Deep Learning based Vehicle Detection in Aerial Imagery

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

M.Sc. Lars Wilko Sommer

aus Bretten

Tag der mündlichen Prüfung:	03.06.2020
Erster Gutachter:	Prof. Dr.-Ing. Jürgen Beyerer
Zweiter Gutachter:	Prof. Dr.-Ing. Stefan Hinz

Abstract

The usage of airborne platforms, such as unmanned aerial vehicles (UAVs), equipped with camera sensors is essential for a wide range of applications in the field of civil safety and security. Amongst others, prominent applications include surveillance and reconnaissance, traffic monitoring, search and rescue, disaster relief and environmental monitoring. However, analyzing the aerial imagery data solely by human operators is often not practicable due to the large amount of visual data and the resulting cognitive overload. In practice, automated processing chains based on appropriate computer vision algorithms are employed to assist human operators in assessing the aerial imagery data. Key component of such processing chains is an accurate detection of all relevant objects inside the camera's field of view, before the scene can be analyzed and interpreted. The low spatial resolution originating from the large distance between camera and ground makes object detection in aerial imagery a challenging task, which is further impeded by motion blur, occlusions or shadows. Although many conventional approaches for object detection in aerial imagery exist in the literature, the limited representation capacity of the utilized handcrafted features often inhibits reliable detection accuracies due to the occurring high variance in object scale, orientation, color, and shape.

In the scope of this thesis, a novel deep learning based detection approach is developed, whereby the focus lies on vehicle detection in aerial imagery recorded in top view. For this purpose, Faster R-CNN is chosen as base detection framework because of its superior detection accuracy compared to other deep learning based detectors. Relevant adaptations to account for the specific characteristics of aerial imagery, especially the small object dimensions, are

systematically examined and resulting issues with respect to real-world applications, i.e., the high number of false detections caused by vehicle-like structures and the poor inference time, are identified. Two novel components have been proposed to improve the detection accuracy by enhancing the contextual content of the employed feature representation. The first component aims at increasing spatial context information by combining features of shallow and deep layers to account for fine and coarse structures, while the latter component leverages semantic labeling – the pixel-wise classification of an image – to introduce more semantic context information. Two different variants to integrate semantic labeling into the detection framework are realized: exploitation of the semantic labeling results to filter out unlikely predictions and inducing scene knowledge by explicitly merging the semantic labeling network into the detection framework via shared feature representations. Both components clearly reduce the number of false detections, resulting in considerably improved detection accuracies. To reduce the computational effort and consequently the inference time, two alternative strategies are developed in the context of this thesis. The first strategy is replacing the default CNN architecture used for feature extraction with a lightweight CNN architecture optimized with regard to vehicle detection in aerial imagery, while the latter strategy comprises a novel module to restrict the search area to areas of interest. The proposed strategies result in clearly reduced inference times for each component of the detection framework. Combining the proposed approaches significantly improves the detection performance compared to the standard Faster R-CNN detector taken as baseline. Furthermore, existing approaches for vehicle detection in aerial imagery, taken from the literature, are outperformed in quantitative and qualitative manner on different aerial imagery datasets. The generalization ability is further demonstrated on a large set of previously unseen data collected from novel aerial imagery datasets with differing properties.

Kurzfassung

Der Einsatz von luftgestützten Plattformen, die mit bildgebender Sensorik ausgestattet sind, ist ein wesentlicher Bestandteil von vielen Anwendungen im Bereich der zivilen Sicherheit. Bekannte Anwendungsgebiete umfassen unter anderem die Entdeckung verbotener oder krimineller Aktivitäten, Verkehrsüberwachung, Suche und Rettung, Katastrophenhilfe und Umweltüberwachung. Aufgrund der großen Menge zu verarbeitender Daten und der daraus resultierenden kognitiven Überbelastung ist jedoch eine Analyse der Luftbilddaten ausschließlich durch menschliche Auswerter in der Praxis nicht anwendbar. Zur Unterstützung der menschlichen Auswerter kommen daher in der Regel automatische Bild- und Videoverarbeitungsalgorithmen zum Einsatz. Eine zentrale Aufgabe bildet dabei eine zuverlässige Detektion relevanter Objekte im Sichtfeld der Kamera, bevor eine Interpretation der gegebenen Szene stattfinden kann. Die geringe Bodenauflösung aufgrund der großen Distanz zwischen Kamera und Erde macht die Objektdetektion in Luftbilddaten zu einer herausfordernden Aufgabe, welche durch Bewegungsunschärfe, Verdeckungen und Schattenwurf zusätzlich erschwert wird. Obwohl in der Literatur eine Vielzahl konventioneller Ansätze zur Detektion von Objekten in Luftbilddaten existiert, ist die Detektionsgenauigkeit durch die Repräsentationsfähigkeit der verwendeten manuell entworfenen Merkmale beschränkt.

Im Rahmen dieser Arbeit wird ein neuer Deep-Learning basierter Ansatz zur Detektion von Objekten in Luftbilddaten präsentiert. Der Fokus der Arbeit liegt dabei auf der Detektion von Fahrzeugen in Luftbilddaten, die senkrecht von oben aufgenommen wurden. Grundlage des entwickelten Ansatzes bildet der Faster R-CNN Detektor, der im Vergleich zu anderen Deep-Learning basierten Detektionsverfahren eine höhere Detektionsgenauigkeit besitzt. Da

Faster R-CNN wie auch die anderen Deep-Learning basierten Detektionsverfahren auf Benchmark Datensätzen optimiert wurden, werden in einem ersten Schritt notwendige Anpassungen an die Eigenschaften der Luftbilddaten, wie die geringen Abmessungen der zu detektierenden Fahrzeuge, systematisch untersucht und daraus resultierende Probleme identifiziert. Im Hinblick auf reale Anwendungen sind hier vor allem die hohe Anzahl fehlerhafter Detektionen durch fahrzeugähnliche Strukturen und die deutlich erhöhte Laufzeit problematisch. Zur Reduktion der fehlerhaften Detektionen werden zwei neue Ansätze vorgeschlagen. Beide Ansätze verfolgen dabei das Ziel, die verwendete Merkmalsrepräsentation durch zusätzliche Kontextinformationen zu verbessern. Der erste Ansatz verfeinert die räumlichen Kontextinformationen durch eine Kombination der Merkmale von frühen und tiefen Schichten der zugrundeliegenden CNN Architektur, so dass feine und grobe Strukturen besser repräsentiert werden. Der zweite Ansatz macht Gebrauch von semantischer Segmentierung um den semantischen Informationsgehalt zu erhöhen. Hierzu werden zwei verschiedene Varianten zur Integration der semantischen Segmentierung in das Detektionsverfahren realisiert: zum einen die Verwendung der semantischen Segmentierungsergebnisse zur Filterung von unwahrscheinlichen Detektionen und zum anderen explizit durch Verschmelzung der CNN Architekturen zur Detektion und Segmentierung. Sowohl durch die Verfeinerung der räumlichen Kontextinformationen als auch durch die Integration der semantischen Kontextinformationen wird die Anzahl der fehlerhaften Detektionen deutlich reduziert und somit die Detektionsgenauigkeit erhöht. Insbesondere der starke Rückgang von fehlerhaften Detektionen in unwahrscheinlichen Bildregionen, wie zum Beispiel auf Gebäuden, zeigt die erhöhte Robustheit der gelernten Merkmalsrepräsentationen. Zur Reduktion der Laufzeit werden im Rahmen der Arbeit zwei alternative Strategien verfolgt. Die erste Strategie ist das Ersetzen der zur Merkmalsextraktion standardmäßig verwendeten CNN Architektur mit einer laufzeitoptimierten CNN Architektur unter Berücksichtigung der Eigenschaften der Luftbilddaten, während die zweite Strategie ein neues Modul zur Reduktion des Suchraumes umfasst. Mit Hilfe der vorgeschlagenen Strategien wird die Gesamtlaufzeit sowie die Laufzeit für jede Komponente des Detektionsverfahrens deutlich reduziert.

Durch Kombination der vorgeschlagenen Ansätze kann sowohl die Detektionsgenauigkeit als auch die Laufzeit im Vergleich zur Faster R-CNN Baseline signifikant verbessert werden. Repräsentative Ansätze zur Fahrzeugdetektion in Luftbilddaten aus der Literatur werden quantitativ und qualitativ auf verschiedenen Datensätzen übertroffen. Des Weiteren wird die Generalisierbarkeit des entworfenen Ansatzes auf ungesehenen Bildern von weiteren Luftbilddatensätzen mit abweichenden Eigenschaften demonstriert.

Contents

1 Introduction

1.1 Motivation

Recording of images or videos with sensors mounted on satellites or airborne platforms, e.g., helicopters, unmanned aerial vehicles (UAVs) etc., afford the coverage of large areas and the capturing of multiple objects and their interactions by a single sensor. Thus, a central limitation of stationary camera networks near ground is overcome. Owing to the increasing technological advancements taking place across imaging systems and airborne platforms and the accompanying decreasing costs, the fields for applications employing aerial imagery, also referred to as *remote sensing imagery*, are growing rapidly [Res17]. While the global aerial imaging market accounted for US $1.56 billion in 2017, it is expected to reach US $6.24 billion by 2026 [Res18]. Amongst others, common applications expected to witness significant market growth include search and rescue tasks [Rud08, Goo09, Qi16], disaster relief [Ada11, Eze14, Erd16], traffic monitoring [Ang03, Len08, Kan15] and surveillance and reconnaissance tasks [Gir04, Hei10, Rei10]. Illustrative examples underlining the utility of aerial imagery for such applications are given in Figure 1.1.

Due to large search areas with often restricted accessibility such as mountains or open seas, helicopters or UAVs equipped with specialized camera sensors, e.g., thermal infrared (IR), are often used to assist in the recovery of missing or injured humans and to assist in the manhunt and apprehension of suspected criminals or fugitives. Studies comparing the effectiveness of airborne assets and ground search teams in terms of success rate and localization time substantiate their benefits for search and rescue tasks [Ham17, Eye18]. It has been shown that the support of human operators by automated algorithms is

crucial for high success rates, as dynamic and complex environments impede the localization of persons that are only visible in a short time range [Goo09].

(a) Search and rescue[1]

(b) Disaster relief[2]

(c) Traffic monitoring[3]

(d) Aerial surveillance[4]

Figure 1.1: Examples for applications based on imagery recorded with aerial sensor platforms.

In the event of natural disasters, such as earthquakes, landslides, tsunamis or floods, the most important issue is to preserve human lives, whereby the first 72 hours are the most critical [Erd16]. Therefore, fast and efficient conduct of search and rescue missions is imperative. While traditional assessment

[1] https://www.thueringer-allgemeine.de/leben/recht-justiz/nach-razzia-in-gierstaedt-spezialeinheit-fahndet-nach-schleusern-und-hintermaennern-id223248397.html

[2] https://ulcrobotics.com/services/gas-utility-unmanned-aerial-inspection/

[3] Recordings from Fraunhofer IOSB over Karlsruhe, Germany

[4] https://verkehrsforschung.dlr.de/en/news/visual-contact-wiesn

methodologies including damage survey vehicles or unmanned ground vehicles possess innate limitations regarding accessibility and timeliness, imagery acquired by airborne platforms facilitates situational awareness and consequently disaster management in real-time [Ada11]. To support human operators suffering from cognitive overload, automatic damage assessment systems are required [Erd16].

The growing traffic volume provoked an increased demand for automated traffic monitoring and management. Besides stationary cameras mounted near ground, induction loops embedded in pavements and pneumatic tubes stretched across roads used to estimate the traffic flow of particular areas, aerial imagery acquired by airborne platforms has proven to be an ideal complement allowing the coverage of large areas [Kan15]. With the rise of autonomous driving, novel applications are the generation of realistic data, i.e., vehicle trajectories extracted from aerial imagery, required as input for simulation tools and the acquisition of aerial imagery as reference for onboard sensors with a rather limited view of the overall scenario [Kur18].

Aerial surveillance tasks range from border patrol to monitoring of large events. In recent years, the fast coverage of large borderlines even in difficult terrains has led to an increased use of airborne systems to prevent unwanted cross-border activities like smuggling or human trafficking, as existing solutions on the ground are often tedious, error-prone, costly and time-consuming [Ber16]. Airborne systems further facilitate continuous monitoring of a particular area as in case of large events, e.g., festivals and sports events, whereby the recorded aerial imagery provides fast access to situational information required by organizers and security teams on the ground in case of emergency or traffic management [Röm16].

All these applications share the need for an automated processing chain to assess the aerial imagery. Assessing huge amounts of data as in case of traffic monitoring or aerial surveillance by human operators is not workable in a time-efficient and cost-effective manner, while assistance for human operators is required in case of search and rescue tasks and disaster relief to counteract cognitive overload caused by dynamic and complex environments. Key component of such processing chains is an accurate detection of all relevant

objects, e.g., vehicles inside the camera's field of view, before the scene can be analyzed and interpreted. This thesis aims at the development of an efficient detection pipeline suited for the task of detecting vehicles in aerial imagery, which are the objects of interest for a wide variety of applications. The focus hereby lies on vehicle detection in images recorded in top view, also referred to as *nadir view*, as it allows the coverage of large areas at uniform detail. For this purpose, deep learning techniques that show promising results in most fields of computer vision are explored to overcome shortcomings of conventional vehicle detection methods based on handcrafted features.

1.2 Challenges

Vehicle detection in aerial imagery captured from sensor platforms like aircraft, UAVs or satellites is a challenging task. The main reasons for this are the high distance between sensor and ground, the capturing conditions and varying scenarios due to different daytimes and regions. Figure 1.2 depicts the challenges in detail, which can be categorized as follows:

1. Challenges arising from image acquisition:
    - **Image noise** is the random deviation from the real pixel intensity values. Main causes are statistical quantum fluctuations, the physics of the camera sensor and intensity quantization.
    - **Blurring** can have different reasons such as objects being out of focus, object motion as well as camera motion or camera shake. Motion blur occurs especially in case of weak illumination leading to longer camera exposure times. Blurring results in weak contrast and reduced sharpness.
    - **Illumination** strongly affects the image quality. Low illumination as in case of twilight requires longer exposure times leading to motion blur or resulting in increased image noise.

Too strong illumination can cause saturation of the camera sensor, which leads to an excessive image brightness and reduced amount of image details.

- **Low spatial resolution** originating from the large distances between camera and ground yields small object dimensions. Objects in the range of only few pixels comprise only little information about their appearance and shape, which impedes the detection and classification task. Varying resolution leads to variation in object dimensions that may result in misclassification, e.g., for classes car and van.

2. Challenges due to object and scene variations:
   - **High intra-class variance** due to the huge variety of vehicle colors, scales and shapes impedes the learning of a robust feature representation used to distinguish between vehicles and non-vehicles. In aerial imagery, the intra-class variance is further increased by arbitrary vehicle orientations due to the camera perspective.
   - **Low inter-class variance** makes it difficult to distinguish between different object categories. Due to the recording perspective, different object categories, e.g., car and van, exhibit similar sizes and shapes, which may cause misclassified objects.
   - **Intricate background** can result in a huge number of false positive detections. In particular, in urban or industrial areas, numerous objects exhibit high similarity in scale and outline compared to vehicles, which may result in only small differences in the feature representations.
   - **Partial occlusion**, e.g., caused by trees or traffic signs, alters the appearance of objects. The reduced amount of visible features impedes the classification between object and background.

- **Shadows** appear when direct light, e.g., sunlight, is obstructed either partially or totally by an object. Especially during morning or afternoon hours, cast shadows can lead to distortion of object shapes and result in weak contrast in shadow areas.

Besides the aforementioned challenges due to image quality as well as object and scene variation, real-world applications impose further requirements on the detection algorithms. These practical requirements comprise:

- **Generality and transferability** of the detection algorithms are required to ensure high robustness against variations in the data. Common detection algorithms make usage of machine learning approaches that learn the appearance of vehicles and non-vehicles from given samples. As these given samples are typically restricted, the learned model should be able to localize and classify vehicles in new, previously unseen data.
- **Real-time requirements** have to be met to assure online processing as required for many applications. As opposed to offline processing, the processing of one image has to be completed before the next image is captured. For instance, if the frame rate is 25 Hz, about 40 ms are available to extract and process the current image information. Note that meeting real-time requirements is generally more challenging with larger image sizes due to the increased computational effort.
- **Hardware constraints** due to the limited payload of airborne platforms affect the computing power and consequently the processing time. Embedded systems such as NVIDIA Jetson platforms[1] that comprise a powerful GPU in addition to a potent processor allow low-power, onboard computing for deep learning and computer vision applications. However, the GPU memory that is often shared with the RAM and the number of parallel processing units are clearly less compared to server setups that may comprise multiple GPUs and thus, restrict the size and complexity of deep learning models.

[1] https://developer.nvidia.com/embedded-computing

Figure 1.2: Illustration of typical challenges occurring in aerial imagery. First row: image noise[1], motion blur[2] and illumination[2]. Second row: low resolution[3], high intra-class variance[1] and low inter-class variance[1]. Third row: intricate background[3], partial occlusion[1] and shadow[1].

[1] Image taken from [Xia18]

[2] Image from Fraunhofer IOSB

[3] Image taken from [Raz16]

1.3 Contributions

The aim of this thesis is the design of a deep learning based detection pipeline for vehicle detection in aerial imagery with low spatial resolution, thus enabling the coverage of large areas. The work presented in this thesis makes the following contributions to the field of deep learning based vehicle detection:

- A thorough analysis of applying deep learning based detection frameworks for the task of vehicle detection in aerial imagery is conducted in detail for the first time. Relevant adaptations to address the characteristics of aerial imagery are systematically examined by means of the popular Faster R-CNN detector [Ren15], which comprises a good trade-off between detection accuracy and inference time. Furthermore, resulting issues with respect to real-world applications are identified, i.e., false alarms caused by objects with vehicle-like structures and time-consuming detection components [Som17c, Som17b, Som18b].
- Two novel approaches to improve the detection accuracy by enhancing the contextual information of the detection framework are introduced. The first approach aims at increasing spatial context information by combining features of shallow and deep layers to account for fine and coarse structures [Som18c], while the latter approach leverages semantic labeling – the pixel-wise classification of an image – to introduce more semantic context information. Two different variants to integrate semantic labeling into the detection framework are realized: exploitation of the semantic labeling results to filter out unlikely predictions [Som17a] and inducing scene knowledge by explicitly merging the semantic labeling network into the detection framework via shared feature representations [Nie18]. The proposed approaches yield improved detection accuracy on publicly available aerial imagery benchmark datasets, as the number of false alarms is considerably reduced.

- A novel semantic labeling aerial imagery dataset is generated by pixel-wise annotation of the DLR 3K dataset, which allows for better integration and evaluation of semantic labeling in the context of vehicle detection in aerial imagery. In cooperation with the German Aerospace Center (DLR), a refined and enhanced version of the dataset is made publicly accessible to researchers [Azi19]. Note that the DLR provides, inter alia, more fine-grained annotations such as lane-markings.
- As vehicle detection in real-time or nearly in real-time is often a prerequisite for real-world applications, two strategies to reduce the computational effort and consequently the inference time are proposed. The first strategy is replacing the default CNN architecture used for feature extraction with a lightweight CNN architecture optimized with regard to vehicle detection in aerial imagery [Rin19]. Making use of the circumstance that vehicles generally cover only a small fraction of aerial imagery, the latter strategy comprises a novel module to restrict the search area prior to the detection modules [Som18a]. The proposed strategies result in clearly reduced inference times for each component of the detection pipeline.
- An extensive evaluation on several publicly available datasets shows the superior detection performance of the detection method proposed in the context of this thesis compared to representative existing work. Furthermore, the transferability of conducted adaptations to account for the characteristics of aerial imagery to more recent deep learning based detection frameworks is demonstrated [Som18d, Aca18] and the generalization ability of the proposed detection method is visualized on unseen data with differing image content and image quality.

1.4 Thesis Outline

This thesis is organized as follows: Chapter 2 provides fundamentals of deep learning that are essential for the remainder of this thesis. Furthermore, a

thorough review about deep learning based detection frameworks and about related work on vehicle detection in aerial imagery is given. In Chapter 3, the concept of the proposed detection pipeline is introduced. Similarities and differences compared to other concepts are identified and discussed. The evaluation protocol and aerial imagery datasets utilized in this thesis are introduced in Chapter 4. In Chapter 5, the base framework of the proposed detection pipeline is presented. Adaptations proposed in order to account for characteristics of aerial imagery are examined and issues with regard to real-world applications are identified. In Chapter 6, two novel components to improve the detection accuracy by enhancing the spatial and semantic content of the employed features are described and evaluated in detail. Chapter 7 provides a detailed description and evaluation of two alternative strategies proposed to improve the inference time. In Chapter 8, an extensive evaluation of the proposed detection pipeline is conducted. Different possibilities to combine the proposed components and strategies are examined. A comparison of the individual and combined approaches to representative existing work in the literature is performed in a qualitative and quantitative manner. Finally, concluding remarks and potential future research are summarized in Chapter 9.

2 Related Work

The goal of this thesis is the design of a deep learning based detection pipeline for vehicle detection in aerial imagery. In the following chapter, an overview about the existing work that covers the relevant modules of the proposed detection pipeline is given. First of all, fundamentals of deep learning, in particular of convolutional neural networks, that are essential for the remainder of this thesis, are introduced in Section 2.1. Section 2.2 gives a thorough review about deep learning based detection frameworks and recent advancements. Related work on vehicle detection in aerial imagery is summarized in Section 2.3. The literature under review focuses on aerial imagery captured from aircraft, UAVs or satellites equipped with visual cameras operating in top view. The considered literature can be distinguished into conventional approaches employing handcrafted features and deep learning based approaches.

2.1 Deep Learning

Neural networks have been applied in computer vision and related search areas for a long time. Since the first attempts in 1943, when McCulloch and Pitts [McC43] created a mathematical model to emulate the neural networks of the human brain, neural networks have progressed through several evolutionary stages. Their most recent form is often termed *deep learning* referring to the large number of layers, which have become feasible with the advancements of the required hardware [Sch15]. The most popular variant of neural networks applied to numerous computer vision tasks is the *convolutional neural network* (CNN). In 2012, CNNs experienced their wide breakthrough in computer vision with the remarkable success of AlexNet [Kri12] in the ImageNet

classification challenge [Den09], which requires the classification of images into one of 1000 diverse classes. AlexNet, the first CNN participating in the challenge, reduced the error rate by a significant margin compared to previous solutions, which was an initial indicator for the ability of CNNs to capture a large and diverse number of image contents.

2.1.1 Multilayer Perceptron

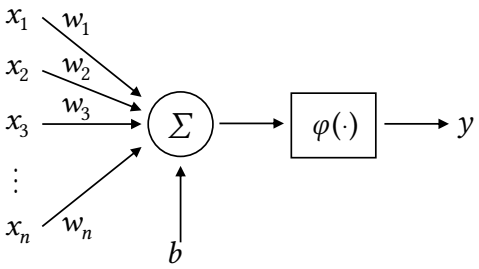


Figure 2.1: Structure of a single perceptron with input vector $\mathbf{x}$, weights $\mathbf{w}$, bias b and output y.

The most common elementary unit of a neural network is the *perceptron* introduced by Frank Rosenblatt in 1957 [Ros58]. Its basic structure is depicted in Figure 2.1. A perceptron takes n scalar inputs, generally provided as an n-dimensional vector $\mathbf{x} \in \mathbb{R}^n$ and has a single scalar output y. The output of the perceptron is defined as the weighted sum of its inputs passed through an activation function:

$$y = \varphi(\mathbf{w}^T \mathbf{x} + b). \tag{2.1}$$

The weight vector $\mathbf{w} \in \mathbb{R}^n$ and the optional bias term b are the free parameters of the neural network learned during training. The activation function $\varphi(\cdot)$ is a non-linear function introduced to facilitate the learning of a non-linear decision function. Common choices are the sigmoid function, the hyperbolic

tangent function and the Rectified Linear Unit (ReLU) function. The ReLU – most frequently applied as activation function in CNNs – is described in more detail in Section 2.1.2.

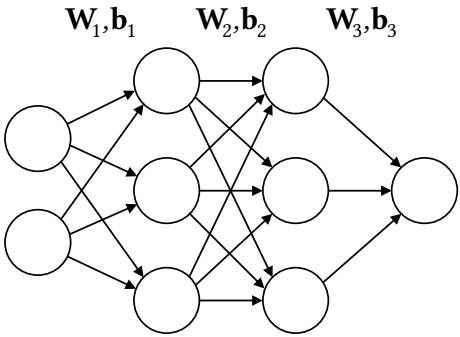


Figure 2.2: Multilayer Perceptron with two input neurons, two hidden layers with three neurons each, and a single output neuron.

While a single perceptron is limited in its ability to approximate decision functions, multiple perceptrons can be combined in a directed acyclic graph, as illustrated in Figure 2.2, in order to allow an accurate approximation of complex decision functions. Within this so-called *Multilayer Perceptron (MLP)*, sets of perceptrons denoted as *neurons* are arranged into three types of layers: an input layer, one or more hidden layers and an output layer. Each neuron of a particular layer i is connected to the outputs of all neurons in the previous layer $i-1$. Thus, these layers are often referred to as *fully connected* layers. The output of the i-th fully connected layer is given by

$$\mathbf{h}_i = \varphi(\mathbf{W}_i \mathbf{h}_{i-1} + \mathbf{b}_i) \tag{2.2}$$

with $\mathbf{h}_0 = \mathbf{x}$ being the input, e.g., image or feature vector, and $\mathbf{h}_n = y$ being the output of an n-layer MLP. $\mathbf{W}_i = [\mathbf{w}_{i1},..., \mathbf{w}_{im})]^T$ is the layer's weight matrix composed of the weight vectors of its m neurons and $\mathbf{b}_i = [b_{i1},..., b_{im})]^T$

is the layer's bias vector. Together, all weight matrices and bias vectors are the trainable parameters of the network. Note that the number of trainable parameters can rise significantly with an increasing number of neurons in the network due to the pairwise connection between neurons of adjacent layers.

2.1.2 Convolutional Neural Networks

CNNs are a specialized kind of neural network designed for processing high-dimensional data with a known grid-like topology, e.g., images or video frames. Typical CNNs consist of three basic types of layers: *convolutional layers*, *pooling layers* and *fully connected layers* [LeC98]. To reduce the number of parameters and consequently the complexity of a neural network, convolutional layers make use of two basic ideas illustrated in Figure 2.3. Instead of connecting every neuron with all neurons of the previous layer, neurons are only connected to neurons of the previous layer within a fixed local neighborhood. Thus, the weight matrix becomes sparse. Furthermore, the weights are shared for all neurons within a layer and thus, become independent of the position of the neuron.

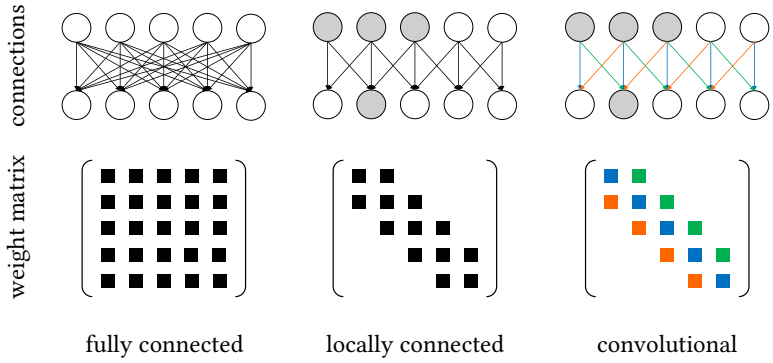


Figure 2.3: Transition from a fully connected layer to a 1D convolutional layer and the respective weight matrix.

In general, convolutional layers consist of a set of learnable filters, also called *channels*, as single convolutional filters do not capture sufficient information from the previous layer. The set of filters is characterized by its kernel size defining the corresponding local neighborhood, which is often referred to as *receptive field*. The weights are not shared between filters, so that each filter is free to learn a different convolutional operation. The output of a convolutional layer is often referred to as *feature map* due to its spatial nature. It spans across the spatial dimensions and contains a feature vector at each location, whose dimension is equal to the number of filters C in the layer. Note that in case of multiple input channels D, each filter is comprised of D kernels with size $k \times k$. Every kernel is shifted across the respective input channel and the resulting outputs are summed together via element-wise addition, yielding a single output channel per filter as illustrated in Figure 2.4. The size of the feature map is affected by stride and padding. The stride parameter specifies the distance between two spatial locations, where filter kernels are applied, while zero padding can be used in order to apply filter kernels at the edges of the input.

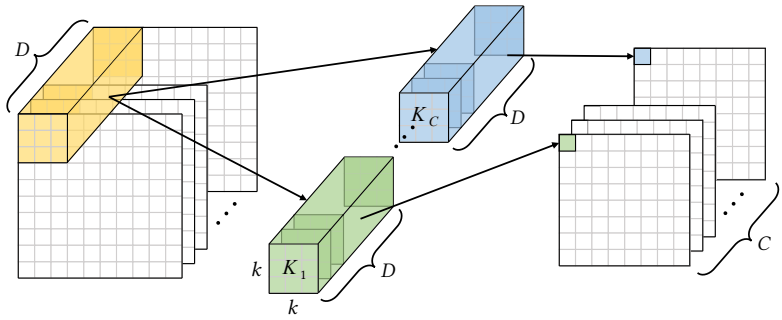


Figure 2.4: Schematic illustration of a convolution with C filters comprised of D kernels with size 3×3. For each filter, every kernel is shifted across the respective input channel and the resulting outputs are summed together via element-wise addition yielding a single output channel per filter.

To approximate complex decision functions, nonlinearities are introduced to a CNN by applying an activation function element-wise to the output of convolutional layers. In practice, ReLU is usually preferred to sigmoidal functions like sigmoid and hyperbolic tangent as activation function [LeC15]. It provides advantages in terms of computational complexity and gradient computation, while obtaining superior results for several tasks across multiple domains [Kri12, Maa13, Glo11]. The ReLU function is defined as

$$\varphi(x) = max(0,x). \tag{2.3}$$

It removes negative values from a feature map by setting them to zero as illustrated in Figure 2.5.

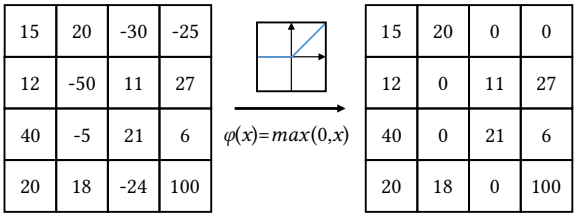


Figure 2.5: Representation of the ReLU functionality and its transfer function.

Pooling layers are inserted periodically after convolutional layers to reduce the spatial size of a feature map and consequently the amount of parameters. This leads to improved computational efficiency, while the invariance to small translations of the input is increased [Sch10]. The pooling operation is defined by its filter size, stride and pooling function. The pooling operation is performed independently on each input channel. Common pooling operations are max pooling with a filter size of 2×2 or 3×3 and a stride of two. Max pooling returns the maximum output within the receptive field defined by the filter size as illustrated in Figure 2.6. Less commonly applied pooling functions are average pooling and ℓ_2-norm pooling as max pooling has been proven to work better in practice [Sch10]. Note that applying convolutions

with stride larger than 1 is an alternative strategy to reduce the spatial size of the feature representation [Spr14].

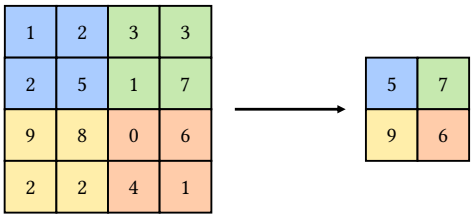


Figure 2.6: Illustration of a max pooling operation with a filter size of 2 × 2 and stride of two.

Fully connected layers that connect each neuron to all neurons of the previous layer like in an MLP are generally used as the last few layers in a CNN. The objective of fully connected layers is to learn non-linear combinations of the high-level features given by the output of the last convolutional layer. The output of the last fully connected layer is used for classification, regression, etc. depending on the particular computer vision task.

2.1.3 CNN Training

The training of a CNN is generally performed in a supervised manner by back-propagation with an objective function termed *loss function L*. The loss function designed for a specific task compares the predicted values of the network to the ground truth (GT) and computes an error measure. In the context of object detection, two types of loss functions, i.e., *classification loss* and *regression loss*, are frequently applied as described in more detail in Section 5.1. Common choice for the classification loss function is the softmax loss [Dud12], while smooth L_1 loss is widely used as regression loss function [Gir15]. A common optimization method is stochastic gradient descent (SGD) where the gradient is calculated for a small, randomly chosen subset of the training data called *batch* $\mathcal{B}$ and then back-propagated through all layers of the network [Mon12].

In each iteration t, the gradients are used to update the current weights of the network as follows:

$$\mathbf{W}^{t+1} = \mathbf{W}^{t} + \Delta\hat{\mathbf{W}}^{t} \tag{2.4}$$

with

$$\Delta\hat{\mathbf{W}}^{t} = \eta \sum_{x \in \mathcal{B}_t} \frac{\partial L_x}{\partial \mathbf{W}^{t}} \tag{2.5}$$

being the gradient over the current batch $\mathcal{B}_t$. The gradient is weighted by the learning rate η that specifies the step size in the gradient descent. To achieve faster convergence, a fraction ω of the previous weight update $\Delta\hat{\mathbf{W}}^{t-1}$ is typically added to the current weight update:

$$\mathbf{W}^{t+1} = \mathbf{W}^{t} + \Delta\hat{\mathbf{W}}^{t} + \omega\Delta\hat{\mathbf{W}}^{t-1}. \tag{2.6}$$

Thus, this method called *momentum* allows to overcome plateaus in the loss function [Dud12]. Besides SGD with momentum, several alternative optimization methods that dynamically adapt the learning rate have been proposed, including AdaDelta [Zei12], AdaGrad [Duc11], RMSprop [Tie12], and Adam [Kin14]. Particularly, the latter optimization method, which is a combination of SGD with momentum and RMSprop, often leads to good results in practice with only little tuning of hyper-parameters. Thus, time-consuming training of several networks with vanilla SGD and various learning rate schedules is avoided.

Overfitting – an over-adaptation to the employed training samples – is a general problem when training CNNs due to their large number of parameters. A common regularization method to address this problem is *weight decay*, which reduces the weights in each iteration by a small fraction δ:

$$\mathbf{W}^{new} = (1 - \delta)\mathbf{W}^{old}. \tag{2.7}$$

This prevents the weights from growing too large, which otherwise dominate the output of the network. Another popular regularization method to avoid

overfitting is *dropout* [Sri14]. During training, dropout deactivates neurons of particular layers with a given probability, forcing the network to learn redundant representations. Thus, formation of critical paths through the network specialized to the training samples is avoided.

To avoid unstable training caused by vanishing or exploding gradients, a proper initialization of the network weights is important. Random initialization, e.g., Xavier initialization [Glo10] or Kaiming initialization [He15], are often applied to address this issue. While random initialization allows more flexibility in the design of the network architecture, it often leads to less optimal results, especially in case of deep networks with many parameters and only a limited amount of available training data. Therefore, the weights of models pre-trained on very large datasets, e.g., the ImageNet classification challenge dataset comprising millions of images [Den09], are commonly used in practice for initialization.

As the gradient signal can become less stable with increasing depth of the network, *batch normalization* has been proposed to make the gradient propagation in the network more stable [Iof15]. In general, batch normalization is applied immediately before the activation function by normalizing the network activations $\tilde{\mathbf{h}}_i = \mathbf{W}_i \mathbf{h}_{i-1} + \mathbf{b}_i$ to zero mean and unit variance:

$$\hat{\mathbf{h}}_i = \frac{\tilde{\mathbf{h}}_i - \mu_{\mathcal{B}} \mathbf{e}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}, \tag{2.8}$$

where $\mu_{\mathcal{B}}$ and $\sigma_{\mathcal{B}}^2$ are the mean and variance calculated for the current batch $\mathcal{B}$, while $\mathbf{e}$ is an all-ones vector with the same length as $\tilde{\mathbf{h}}_i$. Furthermore, a small value ϵ is added to prevent a division by zero. To restore the representation power of the network, a set of learnable parameters γ_i and β_i that scale and shift the normalized input are introduced:

$$\tilde{\mathbf{h}}_i^{BN} = \gamma_i \hat{\mathbf{h}}_i + \beta_i \mathbf{e}. \tag{2.9}$$

As batch normalization prevents activations to become very high or very low, it allows the usage of higher learning rates and thus, accelerates the training process.

2.1.4 CNN Architectures

Since the development of early CNN architectures like LeNet[LeC98] and AlexNet[Kri12], several novel CNN architectures have been proposed to increase the performance and allow for more efficient learning by rearranging the combinations of convolutional layers, pooling layers and fully connected layers. The most notable architectures that have been adapted for a broad range of computer vision tasks can be distinguished into *VGG*, *residual* and *inception* networks.

Similar to prior state-of-the-art networks, e.g., AlexNet, VGG networks (named after the Visual Geometry Group from the University of Oxford) comprise convolutional layers and pooling layers followed by a sequence of fully connected layers arranged in a single path network [Sim14]. While prior state-of-the-art networks made use of computationally expensive 9×9 or 11×11 filters to achieve large receptive fields, VGG networks rely on sequences of 3×3 convolutional layers that emulate the effect of larger receptive fields. The use of sequences of small convolutional layers allowed to increase the network depth up to 19 layers and thus, the extraction of more complex image features. Its variant with 16 layers termed *VGG16* has become a popular choice as feature extractor in many computer vision tasks.

Inception networks rely on a basic unit referred to as *inception module* (see Figure 2.7) [Sze15]. An inception module – first applied in GoogLeNet [Sze15] – consists of parallel branches with different sets of convolutional layers, which are aggregated through concatenation. As each branch possesses a different receptive field, the inception module allows the model to recover both local features via smaller convolutions and more global features through larger convolutions. Bottleneck layers, i.e., 1×1 convolutional layers, are often carried out to restrict the number of channels before larger convolutions.

To reduce the computational complexity, recent variants do not use convolutions larger than 3 × 3. Therefore, large $k \times k$ convolutions are either replaced by sequences of 3 × 3 convolutions or factorized to a combination of $1 \times k$ and $k \times 1$ convolutions [Sze16, Sze17].

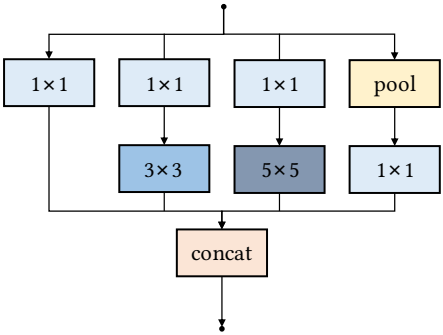


Figure 2.7: Inception module – the basic unit of inception networks – consisting of parallel branches with different sets of convolutional layers, which are aggregated through concatenation.

Residual neural networks [He16a], shortened *ResNets*, overcome the problem of vanishing gradients by adding identity shortcuts to the network, which help to smoothly back-propagate the gradient signal (see Figure 2.8a). These shortcuts further facilitate the learning task, as the intermediate layers only have to learn a residual to its input and no longer an entire feature transformation. Thus, the successful and robust training of deeper network architectures comprising hundreds of layers becomes feasible [He16b]. In practice, bottleneck layers are often applied as first and last layer in a residual block (see Figure 2.8b) to increase and decrease the number of channels, yielding a reasonable number of parameters in case of deep networks. Inspired by the inception architecture, Xie *et al.* proposed a modified variant termed *ResNeXt* [Xie17], which comprises multiple parallel paths with the same topology as depicted in Figure 2.8c. Note that the parameter count is similar to its ResNet counterpart, as the number of filters per path is set to four.

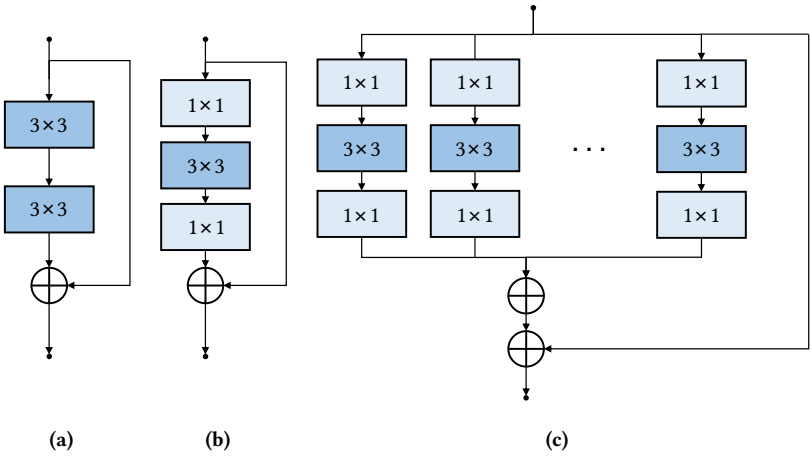


Figure 2.8: Residual block (a), residual block with bottleneck layers (b), and basic unit of ResNeXt networks (c).

2.1.5 Special Layer Types

Though applying CNN architectures with standard convolutions has led to impressive results for a wide range of applications, different types of convolutional layers have been proposed to increase the receptive field, in order to reduce the computational costs compared to standard convolutions and to perform up-sampling within a neural network.

Dilated convolution is a special type of convolution to exponentially increase the receptive field without loss of spatial resolution [Yu15]. Dilated convolution, also referred to as *convolution with dilated filter* or *à-trous convolution*, is characterized by the *dilation coefficient* d, which defines the spacing between the elements of a filter. In practice, the filter elements are matched to distant elements of the input as illustrated in Figure 2.9 yielding a larger receptive field. Note that a dilated convolution with $d = 1$ is equivalent to the standard convolution.

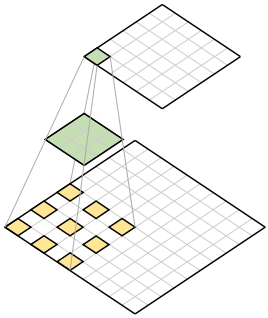
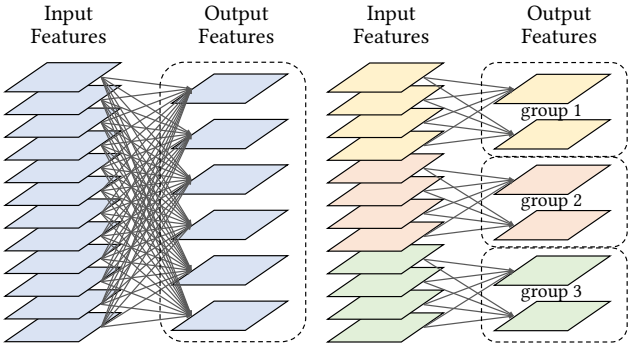


Figure 2.9: Dilated convolution with dilation coefficient $d = 2$.

Group convolutions, first mentioned in [Kri12], have been proposed to reduce the computational costs of standard convolutions. Rather than applying filters with the full channel depth of the input image, i.e., D kernels per filter, the input is split channel-wise into groups and the filters are applied independently for each group (see Figure 2.10b). Letting g denote the number of groups, group convolution filters only consider D/g channels instead of the full channel depth D. Thus, the computational costs are reduced by a factor of g compared to standard convolutions.



(a) Standard convolution **(b)** Group convolution

Figure 2.10: Standard convolution (a) compared to group convolution with three groups (b).

A *depthwise separable convolution* (DSC) performs convolutions independently in spatial and channel domains to reduce the number of parameters and the computational costs [Sif14]. It factorizes the standard convolution into a *depthwise convolution* and a *pointwise convolution* as visualized in Figure 2.11. The depthwise convolution is in essence a special case of group convolution with the same number of groups g and input channels D. Thus, a spatial convolution is performed independently over each channel of the input, while the pointwise convolution, i.e., a 1×1 convolution, is applied across all the D intermediate channels.

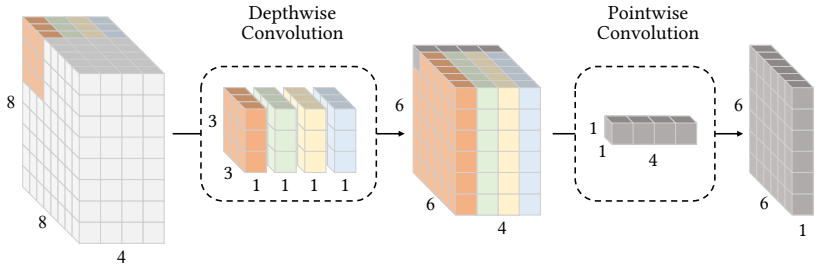


Figure 2.11: Schematic illustration of a depthwise separable convolution that factorizes the standard convolution into a depthwise convolution and a pointwise convolution. Note that the output depth is equal to the number of filters C of the pointwise convolution.

Shuffled group convolutions aim at eliminating the side effect of group convolutions that outputs from a certain channel are only derived from a small fraction of input channels [Zha18b]. Therefore, a novel operation called *channel shuffle* is introduced between two stacked group convolutions. The output channels of each group from the first group convolution are divided into subgroups, which are transposed as depicted in Figure 2.12. The transposed channels are then fed into the second group convolution and thus, allows an information flow across groups. Note that the channel shuffle operation is differentiable and thus, can be directly integrated into CNN architectures for end-to-end training.

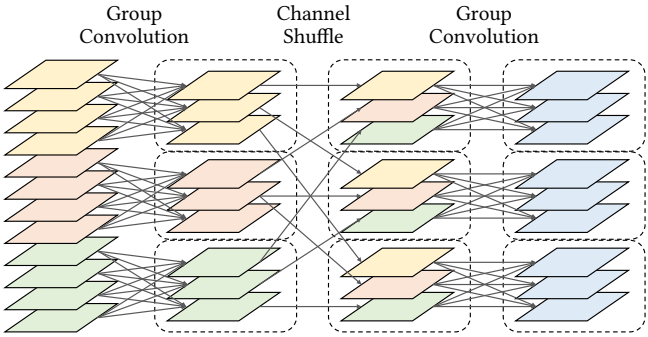


Figure 2.12: Schematic of shuffled group convolutions.

Deconvolution has been proposed to perform up-sampling within a neural network. Note that the deconvolution within the context of CNNs is not the same as the deconvolution defined in signal processing, but a transposed convolution as shown in Figure 2.13. Thus, it is also referred to as *convolution with fractional strides* or *transposed convolution*. Up-sampling with factor f_u can be thought of as convolution with a fractional input stride of $1/f_u$. In practice, deconvolution works by swapping the forward and backward passes of a convolution.

$$\mathbf{w} * \mathbf{x} = \mathbf{W}\mathbf{x} \qquad \mathbf{w} *^{\mathrm{T}} \mathbf{x} = \mathbf{W}^{\mathrm{T}}\mathbf{x}$$

$$\begin{bmatrix} w_0 & w_1 & w_2 & 0 \\ 0 & w_0 & w_1 & w_2 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} w_0 x_0 + w_1 x_1 + w_2 x_2 \\ w_0 x_1 + w_1 x_2 + w_2 x_3 \end{bmatrix} \qquad \begin{bmatrix} w_0 & 0 \\ w_1 & w_0 \\ w_2 & w_1 \\ 0 & w_2 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} w_0 x_0 \\ w_1 x_0 + w_0 x_1 \\ w_2 x_0 + w_1 x_1 \\ w_2 x_1 \end{bmatrix}$$

(a) Convolution **(b)** Transposed Convolution

Figure 2.13: Example of deconvolution in 1D. Convolution with filter $\mathbf{w}$ can be expressed in terms of a matrix multiplication (a). Deconvolution is multiplication with the transposed matrix (b).

2.2 Deep Learning based Object Detection

Object detection, aiming at localizing and classifying object instances from a large number of predefined categories in images, is a fundamental and challenging task in computer vision. With the remarkable success of deep learning based image classification methods, deep learning has been widely adopted to solve the object detection task, yielding significant progress compared to conventional methods relying on handcrafted features [Gir16]. In general, deep learning based detection methods can be distinguished into *two-stage* and *one-stage approaches* [Han18, Liu18, Zha18c]. Two-stage approaches, also referred to as *region proposal based approaches*, comprise two stages. An initial stage generates a set of candidate regions, which are classified in the subsequent stage. In contrast to two-stage approaches, one-stage approaches, also called *regression-based approaches*, predict class probabilities and bounding box offsets with a single feed forward CNN. While two-stage approaches generally outperform one-stage approaches in terms of detection accuracy, one-stage approaches are computationally less expensive and consequently more time-efficient. In the following, an overview about the most important deep learning based detection frameworks and recent extensions yielding improved detection performance is provided. Comprehensive surveys about deep learning based object detection and its advancements are given in [Han18, Liu18, Zha18c].

2.2.1 Two-stage Approaches

Regions with CNN features (R-CNN) proposed by Girshick et al. [Gir14] is one of the first object detection methods that employ CNN features for object detection. Initially, an external region proposal method generally based on handcrafted features, e.g., Selective Search [Uij13], is used to obtain a set of region proposals, also termed *candidate regions*, which are image regions that are likely to contain an object. Region proposal methods, referred to as *object proposal methods*, have been widely applied as alternative to exhaustive sliding window algorithms because of the reduced number of generated candidate regions [Hos15]. Each candidate region is cropped and warped to a

fixed size and used as input for a CNN to extract a fixed-length feature vector. Then, a set of class-specific linear Support Vector Machines (SVMs) is applied to classify the candidate regions. Finally, bounding box regression inspired by [Fel10] is performed for each candidate region to enhance the localization accuracy. Though R-CNN outperforms all previously published works on benchmark datasets by a large margin, it possesses notable drawbacks. The training of this multi-stage detector is complex and time-consuming, since the region proposal method, the CNN used for feature extraction, the SVM classifiers and the bounding box regressor need to be trained and optimized separately. However, the main bottleneck during deployment is the feature extraction, as CNN features are computed for each candidate region separately [Gir15].

Spatial pyramid pooling network (SPPNet) [He14] overcomes this bottleneck by computing convolutional features for the entire input image at once. To account for the fixed-length feature vectors as required for the fully connected layers, the last pooling layer is replaced by a so-called *spatial pyramid pooling layer*. The spatial pyramid pooling extracts features of fixed length for each candidate region from the shared feature map. For this, the spatial pyramid pooling layer partitions the features of the corresponding region of interest into fixed number of spatial bins at multiple levels. Applying the convolutional layers at once for the entire image reduces the runtime during deployment by orders of magnitude. A drawback of SPPNet is that the fine-tuning algorithm is unable to update the convolutional layers that precede the spatial pyramid pooling and consequently limits the accuracy of very deep networks [Gir15].

Fast R-CNN [Gir15] addresses the drawback of SPPNet by introducing a *Region of Interest (RoI) pooling layer*. Similar to SPPNet, Fast R-CNN shares the computation of convolutions across region proposals. However, the spatial pyramid pooling layer is replaced by a RoI pooling layer, which enables fine-tuning of all network layers and consequently facilitates the use of very deep networks. By conducting max pooling, the RoI pooling layer converts the features inside any valid region of interest into a feature map with a fixed spatial extent. The feature map with fixed spatial extent is fed into a sequence of

fully connected layers that branch into two sibling output layers. The first layer outputs softmax probabilities for the object categories and the second layer outputs class-specific bounding box regression offsets for proposal refinement. Unlike R-CNN and SPPNet that employ stage-wise training, Fast R-CNN simplifies the training procedure as classification and bounding box regression are trained within the network. Although Fast R-CNN results in improved detection accuracy and speeds up the detection process, using an external region proposal method remains time-consuming [Ren15].

Faster R-CNN [Ren15] extends Fast R-CNN by integrating a sub-network called *Region Proposal Network (RPN)* that generates region proposals directly within the network. The RPN and Fast R-CNN share the first set of convolutional layers, so that the computational overhead of the RPN is small and the time for generating region proposals is significantly reduced. The RPN is applied on the output of the last shared convolutional layer, which is referred to as feature map. The RPN comprises a convolutional layer followed by two sibling output layers: one for classification and one for bounding box regression. Anchor boxes centered at each feature map location are used as bounding box reference for regression. In order to account for various object dimensions, anchor boxes with different scales and aspect ratios are employed. Region proposals that are likely to contain an object are then forwarded to the classification stage, which is in essence the Fast R-CNN detector.

R-FCN [Dai16] is a region-based fully convolutional network based on Faster R-CNN, which makes use of an alternative classification stage to avoid the application of RoI pooling and fully connected layers for each candidate region separately. For this purpose, the standard RoI pooling layer is replaced by a position sensitive RoI pooling layer, which is applied on top of class-specific position sensitive score maps that are shared across the entire image and not computed for each candidate region separately. The position sensitive score maps that are generated by applying a set of convolutional layers encode the relative position of objects, e.g., top-left corner. The position sensitive RoI pooling layer aggregates the responses of the position sensitive score maps

for each candidate region forwarded from the RPN. R-FCN achieves comparable detection results on benchmark datasets, while the runtime is clearly reduced compared to Faster R-CNN.

2.2.2 One-stage Approaches

You only look once (YOLO) [Red16] models the detection as a regression problem. For this, YOLO divides the input image into an $S \times S$ grid and predicts l bounding boxes for each grid cell. Each bounding box comprises the coordinates relative to the cell, its dimensions and a confidence score about the presence of an object within the box. Unlike region proposal based approaches that employ features from local regions, YOLO uses features from the entire image for bounding box prediction. Regardless of the number of bounding boxes, one set of class probabilities is predicted per grid cell. During testing, the class probabilities and confidence scores are multiplied to generate class-specific confidence scores for each box. Predicting bounding boxes and class probabilities at once results in a significant speed-up compared to other deep learning frameworks. However, the coarse grid division and the constraint that only two boxes are predicted per cell make YOLO prone to miss nearby objects.

Single Shot MultiBox Detector (SSD) [Liu16] is a fully convolutional network, whose functional principle is similar to the RPN in Faster R-CNN. In contrast to the original RPN, which is class-agnostic, SSD predicts a fixed number of bounding boxes and confidence scores for each object category. Anchor boxes centered at each feature map location are used as reference for bounding box regression. Besides employing anchor boxes with different scales and aspect ratios, multiple convolutional layers are employed as feature maps to account for different object dimensions. Shallow convolutional layers exhibiting fine spatial resolutions are employed as feature maps to predict small objects, whereas deep convolutional layers with coarse spatial resolutions are used to predict large objects. SSD achieves clearly improved detection accuracy compared to previous single stage approaches.

YOLOv2 [Red17] is an advanced version of YOLO, which adopts different strategies from SSD and Faster R-CNN. Fully connected layers are removed to make the network fully convolutional, which allows multi-scale training. Similar to Faster R-CNN, anchor boxes are employed instead of predicting the width and height outright. Rather than choosing anchor boxes manually, k-means clustering is employed to generate more appropriate anchor boxes. Furthermore, a so-called *passthrough layer* is added to the network, which forwards features of shallower layers to the last convolutional layer that is used as feature map. Adding fine-grained features to the feature map results in better localization of small objects. YOLOv2 exhibits good detection accuracies on benchmark datasets, while the inference time is considerably reduced compared to prior deep learning based detection approaches.

2.2.3 Extensions

The deep learning based detection frameworks discussed above are generally used as basis for further developments. Multiple extensions including exploitation of novel CNN architectures, multi-layer exploitation and adding context information have been proposed to boost the detection performance.

In general, deep learning based detection frameworks employ CNN architectures developed for object classification as feature extractor. The employed CNN architecture is commonly referred to as *base* or *backbone network*. In [He16a], the authors improve the detection accuracy on benchmark datasets by replacing VGG16 with a residual network as base network for Faster R-CNN. Using ResNeXt networks as base network for Faster R-CNN shows slightly improved detection results compared to conventional residual networks [Xie17]. The detection performance is further improved by employing SENets [Hu18] – the winner of the ImageNet2017 classification challenge[1] – as base network for Faster R-CNN. SENets comprise so-called *Squeeze and Excitation blocks* that adaptively recalibrate feature responses so that informative features can be emphasized and less useful features can be suppressed. In

[1] http://image-net.org/challenges/LSVRC/2017/results

[Zop18], the authors recently employ reinforcement learning to learn an optimized network architecture directly on the dataset of interest. Top performing results are achieved by using these so-called NASNets as base network for Faster R-CNN on the MS COCO dataset [Lin14].

Various approaches have been proposed to improve detection accuracy by exploiting multiple layers. These approaches can be roughly divided into combination of features from multiple layers [Bel16, Kon16, Shr16c] and object prediction at different feature maps [Cai16]. Combining features from multiple layers is often performed to enhance the semantic and contextual information. Bell et al. [Bel16] extend Fast R-CNN by applying RoI pooling on multiple layers. The extracted RoI features are concatenated and used to classify the corresponding region proposals. HyperNet [Kon16] – an extension of Faster R-CNN – concatenates deep, intermediate and shallow feature maps to so-called *Hyper Feature maps* used for object prediction. To account for same feature map resolutions, max pooling is conducted to down-sample shallow feature maps, while deconvolution is applied for up-sampling deep layers. Shrivastava et al. [Shr16c] extend Faster R-CNN by introducing a top-down modulation network to combine features from deep and shallow layers. Deep layers are up-sampled via deconvolution and then concatenated with features of shallow layers. Employing multiple layers as feature maps introduced in [Liu16] is often adopted to account for various object dimensions. In [Cai16], region proposals are generated at multiple layers so that receptive fields match objects of different scales. The generated region proposals are then classified in a subsequent stage similar to Faster R-CNN. Several methods take advantage of both extension schemes [Fu17, Woo18, Lin17a, Red18]. Deconvolutional SSD (DSSD) [Fu17] introduces a deconvolution module to SSD. This module comprises deconvolutional layers to up-sample features of deep layers that are merged with features of shallow layers via element-wise product. Woo et al. [Woo18] propose a similar approach based on SSD called StairNet. Lin et al. [Lin17a] propose an extension of Faster R-CNN referred to as *Feature Pyramid Network (FPN)* that utilizes a top-down path to combine features from different layers. Multiple feature maps are used for object prediction at different scales. YOLOv3 [Red18] is a modified version of YOLOv2

that performs object prediction at different scales using a similar concept to [Lin17a].

Adding context information is an alternative approach to enhance the detection accuracy. To increase the spatial context information, Cai et al. [Cai16] enlarge each candidate region by a factor of 1.5. Zhu et al. [Zhu17] extend R-FCN by an additional branch that extracts features from multiple regions surrounding the candidate region. Bell et al. [Bel16] propose spatial recurrent neural networks to explore contextual information across the entire image. In [Shr16a], the authors augment Faster R-CNN with a semantic segmentation network. The output of the semantic segmentation network is fed into the detection branch to introduce semantic information. StuffNet [Bra17] is an extension of Faster R-CNN that exploits an additional semantic segmentation branch to use the local surrounding of an object to identify it. Mask R-CNN [He17] combines Faster R-CNN with semantic segmentation by predicting the class and box offsets in parallel with a segmentation mask for each candidate region.

Recent extensions also focus on novel training strategies, class imbalance handling, cascaded detection and deformable neural networks. Peng et al. [Pen18] introduce a novel Cross-GPU Batch Normalization (CGBN) scheme that allows training with much larger mini-batch sizes. By using CGBN to train FPN, the authors ranked first in the MS COCO 2017[1] detection challenge. In [Sin18], the authors propose a novel scale normalized training scheme to tackle the wide scale spectrum of object instances. Gradients of object instances are selectively back propagated by ignoring gradients arising from objects of extreme scales. To tackle the imbalance between classes, Lin et al. [Lin17b] propose a novel loss function called *focal loss*, which is in essence a dynamically scaled cross entropy loss. As the scaling factor decays to zero with increasing confidence of correct classes, easy examples are down-weighted, while the training focuses on hard examples. Using focal loss to train RetinaNet, which combines SSD and FPN, yields top performing results on the MS COCO dataset. Cascade R-CNN [Cai18], which is widely applicable across detection

[1] https://places-coco2017.github.io/#winners

frameworks, is comprised of a sequence of detection stages trained with increasing IoU thresholds to be sequentially more selective against false alarms. RefineDet [Zha18a] performs two-step cascaded regression to refine the regression and class prediction. In [Dai17], the authors introduce deformable convolution and deformable RoI pooling modules. Both modules model geometric transformations by learning offsets to the grid sampling locations and can readily replace their plain counterparts in existing CNNs.

2.3 Vehicle Detection in Aerial Imagery

Over the last decades, a broad variety of detection methods has been developed for the task of vehicle detection in aerial imagery. In the following, the literature under review is restricted to vehicle detection in single aerial images, whereas methods aiming at moving object detection by exploiting image sequences, e.g., frame differencing [Kum01, Rei06, Xia10], background subtraction [Rei10, Shi12, Lia12] or optical flow based methods [Yao08, Yu09, Sia12], are not considered as static objects such as parked vehicles are missed. In general, vehicle detection in aerial imagery is either performed by conventional detection methods based on handcrafted features or deep learning based methods.

2.3.1 Conventional Vehicle Detection Methods

Before the emergence of deep learning, vehicle detection in single aerial images was generally conducted by extracting handcrafted features followed by a classifier or cascade of classifiers. Therefore, various combinations of feature extraction techniques and classifiers have been explored. Ruskone *et al.* [Rus96] employed an MLP to analyze the intensity values of a pixel's neighborhood for vehicle detection in aerial imagery. Eikvil *et al.* [Eik09] examined a combination of geometric-shape properties, gray level features and Hu moments to detect vehicles in high-resolution satellite images. Moranduzzo and

Melgani [Mor12, Mor13] proposed the use of Scale-Invariant Feature Transform (SIFT) to extract key points, which are discriminated into points belonging to vehicles or background by means of an SVM classifier. In [Mor14b], the authors made use of Histogram of Oriented Gradients (HOG) features followed by SVMs to detect cars in images acquired by UAVs. Tuermer *et al.* [Tue13] adopted HOG features and disparity maps for vehicle detection in dense urban areas. In [Lei10], the authors proposed adaptive boosting (AdaBoost) in combination with Haar-like features for vehicle detection in very high-resolution satellite images. Liu and Mattyus [Liu15] employed Integral Channel Features (ICF) and an AdaBoost classifier in a soft-cascade structure to achieve fast and robust vehicle detection in aerial imagery. Kembhavi *et al.* [Kem10] combined color probability maps, HOG and pairs of pixels followed by partial least squares to project the high-dimensional feature set onto a much lower dimensional subspace for vehicle detection in satellite images taken from Google Earth. In [Sha12], the authors explored the use of multiple visual features, i.e., Local Binary Pattern (LBP), HOG and opponent histogram, and intersection kernel SVM for vehicle detection in high-resolution aerial images. Grabner *et al.* [Gra08] presented a framework for automatic car detection in aerial images, which combines HOG, LBP and Haar-like features and an AdaBoost classifier. In [Klu07], the authors extended the latter approach by incorporating 3D information obtained from a stereo matcher into the training of the online boosting algorithm. Gleason *et al.* [Gle11] examined the use of HOG and Histogram of Gabor coefficients in combination with following classification techniques: nearest neighbor, random forests and SVMs. Liang *et al.* [Lia12] explored a classification scheme for vehicle detection in wide area motion imagery, which combines HOG and Haar-like features with a multiple kernel SVM used to learn the trade-off between HOG and Haar-like features by constructing an optimal kernel with many base kernels. Xu *et al.* [Xu16] proposed a hybrid method for vehicle detection, which adopts an AdaBoost classifier using Haar-like features and a linear SVM using HOG features.

Alternative approaches for vehicle detection in aerial imagery make use of an explicit model representing the shape of vehicles. These approaches either match the model to the image or group extracted image features to construct

structures similar to the model. Burlina *et al.* [Bur97] proposed to combine contours obtained by an edge detector and votes obtained by the generalized Hough transform of the image, calculated using the shape and size of a sample vehicle. In [Moo02], the authors explored a template matching algorithm based on an operator designed to detect 2D shapes for vehicle detection in aerial imagery. Zhao and Nevatia [Zha03] proposed to extract a subset of features from a wire-frame model such as car boundary, windshield and shadow area, which are passed to a Bayesian network with manually selected parameters in order to get a decision of a car's existence. Hinz and Baumgartner [Hin01] proposed a vehicle detection approach based on a hierarchical 3D vehicle model, describing prominent vehicle features on different levels of details. In [Hin03], the authors conducted vehicle detection in high-resolution aerial images by matching an explicit model mainly comprised of geometric features and radiometric properties to the image. Kim and Malik [Kim03] examined the use of a 3D model for fast vehicle detection based on line features extracted by applying oriented edge detectors followed by connected-component analysis for line grouping. In [Cho09], the authors performed vehicle detection by extracting blobs with vehicle-like geometric and radiometric properties using a mean-shift clustering algorithm. A more thorough review on detection methods based on handcrafted features for remote sensing images is summarized in [Che16a].

2.3.2 Deep Learning based Vehicle Detection Methods

In recent years, conventional methods have been largely replaced by deep learning based methods for the task of vehicle detection in aerial imagery due to the limited representation capacity of handcrafted features. The first deep learning based approaches for vehicle detection in aerial imagery employed CNNs as feature extractor and partially as classifier within a sliding window algorithm. Chen *et al.* [Che13] explored a small CNN comprised of three convolutional layers with subsequent max pooling layers followed by one fully connected layer for feature extraction and classification within a sliding window algorithm. The proposed CNN achieved superior results compared to conventional methods based on handcrafted features, i.e., HOG+SVM

and LBP+SVM, on satellite images collected from Google Earth. In [Che14a], the authors modified the CNN architecture proposed in [Che13] by dividing the outputs of the last convolutional layer into multiple blocks of variable receptive fields to allow the extraction of variably scaled features, which further improved the detection accuracy on the same set of satellite images collected from Google Earth. To reduce the computational effort of the detection pipeline, several authors adopted region proposal techniques for vehicle detection in aerial imagery, yielding a smaller set of candidate regions to classify. By applying Binary Normed Gradients (BING) [Che14c] to extract region proposals instead of the exhaustive sliding window paradigm conducted in [Che13, Che14a], Qu *et al.* [Qu16] achieved a clearly reduced inference time without drop in detection accuracy. Zhu *et al.* [Zhu15] adopted the R-CNN detection pipeline for vehicle detection in aerial images. Selective Search [Uij13] - a hierarchical segmentation-based region proposal method - is applied to generate a set of candidate regions. For each candidate region, CNN features are computed by using AlexNet, which are then classified by means of an SVM. Cheng *et al.* [Che16b] exhibited improved detection results compared to [Zhu15] by introducing a new objective function to train the AlexNet model used for feature extraction. To achieve enhanced rotation invariance against the varying orientations of objects in aerial imagery recorded in top view, the new objective function enforces the training samples to share similar features before and after rotating via a regularization term. In [Jia15], the authors proposed the combination of an efficient graph-based superpixel segmentation method [Fel04] to generate candidate regions and a CNN to classify each candidate into vehicle or non-vehicle. Ammour *et al.* [Amm17] conducted car detection in UAV imagery in three stages: candidate regions were initially generated via a mean-shift algorithm followed by the application of a CNN, i.e., VGG16, to extract highly descriptive features, which are classified by means of an SVM. Long *et al.* [Lon17] employed an ensemble of two CNN models, i.e., AlexNet and GoogleNet, to classify candidate regions extracted via Selective Search, yielding improved detection accuracy compared to the single models. Furthermore, the authors proposed an unsupervised bounding box regression algorithm to boost the localization accuracy of objects in remote sensing images. In [Qu17], the authors proposed the combination of

BING for generating a set of candidate regions and a spatial pyramid pooling-based CNN for feature extraction followed by a two-stage cascaded SVM for classification. By replacing the last pooling layer with a spatial pooling layer similar to SPPNet, the spatial pyramid pooling-based CNN facilitates the extraction of a fixed-length feature vector for each candidate region without deformation or cropping of the input, yielding improved detection accuracy compared to its conventional CNN counterpart. Zhong *et al.* [Zho17] made use of two subsequent CNNs for robust vehicle detection in aerial images. The first CNN generates a set of vehicle-like regions similar to the RPN proposed in [Ren15], which are fed into the second CNN for classification. Audebert *et al.* [Aud17] proposed an alternative detection pipeline for vehicle detection in aerial imagery comprised of two separate CNNs. First, semantic segmentation is conducted to extract vehicle candidates, which are classified in the subsequent stage into vehicle types and background.

As computing CNN features for each candidate region separately is computationally expensive, deep learning based detection frameworks that extract CNN features for the entire image at once were adopted for the task of vehicle detection in aerial imagery as well. In [Xu17a], the authors examined the applicability of Faster R-CNN for vehicle detection in images acquired by a UAV at low altitude. No adaptations to account for the characteristics of the aerial imagery were required to outperform conventional detection methods based on handcrafted features due to the low ground sampling distance (GSD), which denotes the distance between pixel centers measured on the ground. Han *et al.* [Han17a] outperformed object detection methods based on handcrafted features on high-resolution remote sensing imagery by utilizing Faster R-CNN with default settings. In order to account for the small dimensions of vehicles in aerial imagery, Carlet *et al.* [Car17] adapted YOLOv2 by removing the last max pooling layer and the associated convolutional layers to increase the resolution of the employed feature map. In [Sak17], the authors adjusted Faster R-CNN for vehicle detection in multimodal aerial imagery to account for the small vehicle dimensions. Top performing results were achieved by modifying the anchor settings of the RPN and by exploiting shallower layers as feature map. Instead of increasing the resolution of the employed feature map, Li *et al.* [Li17] extended the classification stage of

Faster R-CNN by extracting features of candidate regions enlarged by a factor of 1.5 to enhance the contextual information, which led to improved detection accuracy of vehicles in remote sensing images. Xu *et al.* [Xu17b] modified R-FCN for object detection in remote sensing imagery by introducing deformable convolutions [Dai17] that enhance the transformation modeling capability of standard convolutions by adding learnable offsets to the regular grid sampling locations. However, the impact of the deformable convolutions on the detection accuracy for category car was negligible. To improve the detection performance in aerial imagery, several authors made use of recent extensions proposed for deep learning based detection frameworks described in Section 2.2.3. Deng *et al.* [Den17] proposed a modified variant of Faster R-CNN for vehicle detection in aerial images. By combining the features of multiple convolutional layers as input for the RPN and classification stage, as shallower layers are more suitable for localization and deeper layers are more suitable for classification, the authors clearly improved the recall rate compared to the baseline Faster R-CNN. Inspired by the multi-scale scheme employed in SSD, Deng *et al.* [Den18] proposed a multi-scale object detection framework based on Faster R-CNN for vehicle detection in remote sensing imagery. As opposed to [Ren15], the RPN is applied on multiple feature maps to account for various object scales. Furthermore, the features of multiple convolutional layers are combined as input for the classification stage, which resulted in good detection performance for various categories in remote sensing imagery. Guo *et al.* [Guo18] employed a detection framework similar to FPN for object detection in high-resolution satellite images. A top-down architecture with lateral connections to generate multiple high-level semantic feature maps is used to predict objects with various scales. Top performing results were achieved on the publicly available Northwestern Polytechnical University Very-High-Resolution 10-class (NWPU VHR-10) benchmark dataset [Che14b]. In [Azi18], the authors proposed several extensions to the FPN to improve the detection accuracy in remote sensing imagery. In order to extract strong semantic information from different scales, the authors extracted the features for multiple scales of the input image, which are combined in the so-called image cascade network. Though the detection accuracy was improved, the use of multiple image scales as input is often not practicable for real-world

applications. Wang *et al.* [Wan18a] introduced a single stage detection framework similar to DSSD for object detection in remote sensing images. Features of deep layers are up-sampled and combined with shallow layers, yielding feature maps with high-level semantic information. The authors reported improved detection results in aerial images from various sources. Tayara and Chong [Tay18] adopted RetinaNet, which comprises a top-down pathway to combine features of deep and shallow layers and exploits focal loss, for object detection in very high-resolution aerial images. While the top-down pathway results in feature maps with high-level semantic information, the focal loss down-weights the contribution of easy examples to the loss and thus, the training focuses on hard negatives. Yang *et al.* [Yan18] extended Faster R-CNN for vehicle detection in aerial images by adding skip connections from shallow to deep layers in order to learn features with rich detail information. The authors further adopted focal loss as classification loss for the RPN and classification stage to address the issue of easy positive examples and hard negative examples during training. The applicability of the proposed detection method was demonstrated on an own dataset with images recorded in both nadir and oblique view. Ding *et al.* [Din18] performed several modifications to the base network employed in Faster R-CNN. In order to account for small object dimensions, the authors made use of dilated convolutions to increase the feature map resolution. Furthermore, the feature representation is enhanced by combining features from different layers and the fully connected layers are discarded to speed up the inference time. Alternative approaches to address the issue of class imbalance as well as easy and hard examples are examined in [Tan17, Kog18]. Tang *et al.* [Tan17] explored a detection framework for vehicle detection in aerial imagery comprised of the modified RPN proposed in [Den17] and a cascade of boosted classifiers, which replaces the initial CNN classifier, aiming at reducing the number of false detections by negative example mining. Koga *et al.* [Kog18] explored the use of hard example mining in the training process of a CNN, which is applied within a simple sliding window method for vehicle detection in aerial images. Recent developments in the field of vehicle detection in aerial imagery focus on the prediction of oriented bounding boxes [Azi18, Xia18, Bao19, Din19] and the

detection in images and videos recorded from UAVs under varying camera angles [Zhu18, Che19a, Wan19], which are not in the scope of this thesis.

3 Concept

This thesis aims at the design of a deep learning based detection pipeline for vehicle detection in aerial imagery with low spatial resolution. While the low spatial resolution allows the coverage of large areas, the small size of occurring vehicles complicates their detection. Due to its superior detection accuracy compared to other deep learning based detectors, in particular for small object instances, Faster R-CNN is chosen as base detection framework. Faster R-CNN is comprised of two modules: an initial module referred to as RPN that generates a set of candidate regions, which are then forwarded to the subsequent classification stage. The RPN and classification stage share a sequence of convolutional layers serving as the feature extractor, while the output of the last convolutional layer, denoted as feature map, is used as input for both modules. Note that modern deep learning based detection frameworks, such as Faster R-CNN, are designed for benchmark detection datasets clearly differing from aerial imagery. Thus, several adaptations are required to account for the specific characteristics of aerial imagery. Despite the improved detection accuracy, these required adaptations introduce several shortcomings, e.g., poor inference time and low semantic and spatial content of the employed features.

The overall concept of the proposed detection pipeline specifically designed to address these shortcomings is illustrated in Figure 3.1. The RPN and classification stage that remain basically unchanged are highlighted by gray boxes. Modifications to decrease the computational effort and consequently inference time, i.e., restriction of the search area and computation-efficient feature extraction, are emphasized by blue boxes. Green boxes indicate novel components to increase the semantic and spatial context information of the employed features and thus, improve the detection accuracy. Note that the

proposed components are not limited to Faster R-CNN and thus, can be integrated into other deep learning based detection frameworks. Relying on the application or target data, the components can be applied independently or exchanged with other alternatives, e.g., different feature extractors.

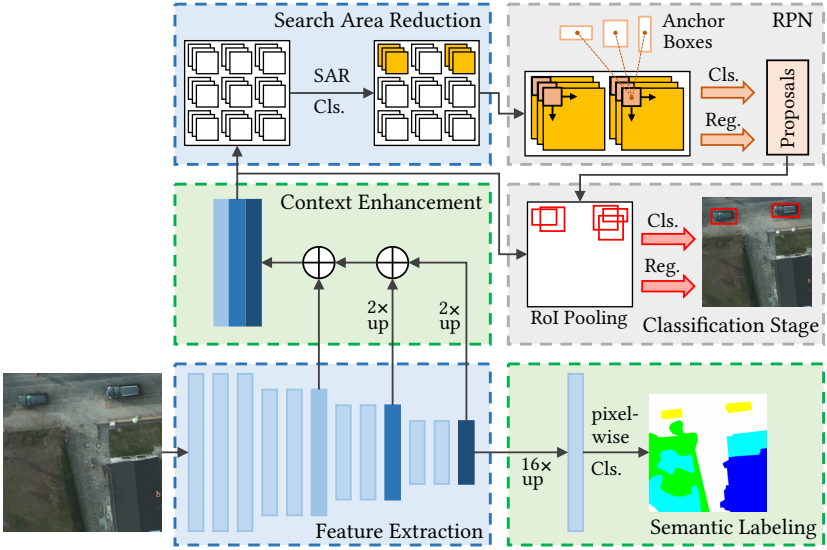


Figure 3.1: Overall concept of the proposed detection pipeline based on the Faster R-CNN detection framework, which comprises an RPN and a classification stage. Two novel components denoted as semantic labeling and context enhancement module are added to improve the detection accuracy by integrating semantic and spatial context information. A search area reduction module and a modified feature extractor are introduced to reduce the computational costs and consequently the inference time.

Detection Framework Adaptation

In the context of this thesis, Faster R-CNN is applied as base detection framework. Compared to other deep learning based detection frameworks, Faster R-CNN achieves superior detection accuracy, especially for small object

instances, and thus, seems most promising for the task of vehicle detection in aerial imagery. However, deep learning based detection frameworks such as Faster R-CNN are typically developed and designed for benchmark datasets that clearly differ from aerial imagery as shown in Figure 3.2 and Figure 3.3. Common benchmark datasets, e.g., PASCAL VOC [Eve10] and MS COCO [Lin14], generally contain one or a few objects per image that are often centered and occupy a high fraction of the image. Thus, even comparatively small objects exhibit a high level of detail such as wheels, license plate, and lights in case of category car. In contrast, aerial imagery datasets like DLR 3K [Liu15] and VEDAI [Raz16], which are typically acquired by sensors mounted on platforms flying at high altitude, generally exhibit low spatial resolutions resulting in small object dimensions, i.e., in the range of a few pixels. Thus, objects that can be randomly located and oriented within the scene often lack in level of detail, which considerably impedes the detection task. Furthermore, the number of objects present in aerial imagery can vary between only a few objects in rural regions and hundreds of objects in urban areas with high traffic volumes or parking lots. Due to these differences, in particular in object dimensions, deep learning based detection frameworks are not directly applicable for vehicle detection in aerial imagery. In order to account for the specific characteristics, several adaptations have been systematically examined with regard to small object dimensions within this thesis [Som17c, Som18b]. Particularly, increasing the feature map resolution considerably improved the detection performance, as the initial resolution is insufficient to precisely localize small vehicle instances in aerial imagery. The transferability of the proposed adaptations to multiple object categories and other detection frameworks are demonstrated in [Som17b, Som18d], respectively. Though the default Faster R-CNN and conventional detection methods are clearly outperformed, the performed adaptations pose drawbacks regarding inference time and semantic and spatial content of the employed features, which are addressed in the following.

Figure 3.2: Example images from benchmark object detection datasets, i.e., PASCAL VOC (left) [Eve10] and MS COCO (right) [Lin14], generally contain one or a few objects that are often centered and occupy a high fraction of the image. Even smaller objects exhibit high level of detail, e.g., wheels, license plate and lights.

Figure 3.3: Example images from aerial imagery datasets, i.e., DLR 3K (left) [Liu15] and VEDAI (left) [Raz16], can contain multiple randomly located objects whose size is in the range of a few pixels.

Integration of Contextual Knowledge

Increasing the feature map resolution as required for an accurate localization of small objects is conducted by exploiting shallower layers as feature maps. However, the lack of semantic and spatial content compared to features from deep layers results in false alarms caused by objects with vehicle-like shapes such as windows or solar panels on buildings (see Figure 3.4). To overcome the lack of semantic and spatial content, the detection pipeline is extended by two novel components.

Figure 3.4: Qualitative detection results on the DLR 3K dataset that highlight the effect of the performed adaptations. False alarms due to the exploitation of shallow layers as feature map are mainly caused by objects with vehicle-like shapes such as solar cells or windows on buildings.

Incorporating spatial context information into deep learning based detectors has led to improved detection results in aerial imagery. While making use of padded GT boxes has been proposed to automatically learn the context of vehicles in aerial imagery [Sak17], adding more context information by increasing the receptive filed via dilated convolution has been recently applied for object detection in aerial imagery such as building detection [Ham18] and

vehicle detection [Din18]. Inspired by semantic labeling networks that combine features of shallow and deep layers to account for fine and coarse structures, e.g., FCN [Lon15], an alternative approach is pursued within this thesis [Som18c]. Therefore, Faster R-CNN is extended by a novel component denoted as context enhancement module (see Figure 3.1). To integrate semantic and contextual information of deep layers, while maintaining a high feature map resolution, the features of deep layers are up-sampled via deconvolution and combined with features of shallow layers, which is a similar concept as in [Lin17a, Woo18]. Adopting the proposed deconvolution module for other detection frameworks confirms the benefit of adding context information from deep layers for vehicle detection [Aca18].

To address false alarms caused in image regions that are unlikely to contain vehicles such as buildings, semantic context information is often applied in conventional detection pipelines. For this, a common procedure is restricting detections to road areas, assuming that vehicles do not appear offside roads [Tue13, Mor14a]. However, the accuracy of road databases is often limited, which causes missed detections due to vehicles parked close to buildings [Tue13]. To cope with this issue, a novel approach that replaces the road database by a semantic labeling mask is developed [Som17a]. As the proposed semantic labeling network accurately predicts roads as well as driveways and parking lots, only detections mainly located on buildings or low vegetation are filtered out. To avoid large computational overhead due to an additional semantic labeling network, an alternative approach that incorporates semantic labeling into the detection framework is introduced in this thesis [Nie18]. Instead of filtering out detections offside roads, semantic labeling is employed to induce scene knowledge into the feature maps used within the detection framework (see Figure 3.1). For this purpose, the semantic labeling and the detection network are merged by sharing a sequence of convolutional layers, which has an implicit effect on the resulting detections. Explicitly adding deep features of the semantic labeling branch to the detection branch further boosts the detection accuracy [Nie18]. Unlike the popular Mask R-CNN [He17] that applies a semantic labeling network for each region proposal, the proposed approach is not limited to scene context within region proposals,

which is beneficial in case of vehicle detection in aerial imagery. The proposed principle is most similar to StuffNet [Bra17], which makes use of the local surroundings of an object to identify it, yielding improved detection accuracy on PASCAL VOC. Furthermore, a novel semantic labeling dataset[1] is created within the context of this thesis to evaluate the effect of semantic labeling on vehicle detection in aerial imagery [Azi19].

Runtime Optimization

Though the performed adaptations to account for the characteristics of aerial imagery considerably improve the detection accuracy, the resulting poor inference time impedes the usage for real-world applications that require vehicle detection in real-time or nearly in real-time. To accelerate the detection pipeline, two different strategies are pursued in the context of this thesis.

As feature extraction is one of the most time-consuming parts in modern deep learning based detection frameworks such as Faster R-CNN, replacing the default CNN architecture with more computational efficient networks is common practice to reduce the inference time. Employing recent architectures developed for use on mobile platforms with limited resources as feature extractor exhibit clearly reduced inference time without large drops in detection accuracy [Wan18b, Zha18b]. The common principle is reducing the number of parameters and computational operations by minimizing the use of expensive 3×3 convolution filters. While these lightweight architectures are generally designed for classification tasks, adopting this principle to vehicle detection in aerial imagery without caution may degrade the detection accuracy, as feature extraction is restricted to shallow layers. In this thesis, the applicability of different lightweight architectures is examined exemplarily for SSD [Rin19], which allows a straightforward exchange of the CNN architecture. In combination with further techniques for runtime optimization, the inference time is considerably reduced, while only sacrificing little to no detection accuracy.

[1] A refined and enhanced version of the dataset is made publicly available in cooperation with DLR, whereby DLR provides more fine-grained annotations: https://www.dlr.de/eoc/en/desktopdefault.aspx/tabid-12760

The most promising architectures are adopted as feature extractor for Faster R-CNN, exhibiting clear speed-up as well.

Restricting the search area to areas of interest, e.g., roads for vehicle detection, is an often applied preprocessing step in conventional object detection pipelines to reduce the inference time. For this, common procedures are the usage of road maps such as OpenStreetMap[1] [Tue13, Lei14] or the extraction of road areas in a preceding area classification stage [Luo12, Mor14a]. In the context of this thesis, a novel component is developed adopting the principle of classifying areas of interest for deep learning based detection frameworks. This novel component termed Search Area Reduction module classifies image areas into regions with or without possibly relevant objects as visualized in Figure 3.1. By filtering out regions that are unlikely to contain at least one object, the computational effort for the subsequent detection stages, i.e., the RPN and classification stage, and consequently the inference time are notably reduced. While existing work employs a separate network for identifying areas of interest, e.g., cascaded application of Faster R-CNN networks [Han17b], the proposed approach is the first that explicitly integrates the search area reduction into the detection framework by sharing convolutional features. Hence, the number of additional parameters and computational costs are minimized.

[1] https://www.openstreetmap.org/

4 Experimental Setup

In this chapter, the experimental setup to evaluate the detection methods proposed in the context of this thesis is introduced. Section 4.1 gives an overview about the employed aerial imagery detection datasets and their main statistics. The evaluation protocols and according metrics are presented in Section 4.2.

4.1 Datasets

An overview about vehicle detection in aerial imagery datasets and their key characteristics is given in Table 4.1. Note that only publicly available datasets are listed. The number of instances is limited to vehicles, as further categories, e.g., buildings, bridges, harbors etc., are not in the scope of this thesis. The DLR 3K Munich Vehicle Aerial Image Dataset [Liu15] is chosen as base dataset to examine the proposed detection pipeline and its components. Its number of annotated vehicles clearly exceeds previous datasets, which is beneficial for learning based detection algorithms as explored in this thesis. The Cars Overhead with Context (COWC) dataset [Mun16], which comprises even more vehicles, is left aside, as annotations are only provided in form of center coordinates. To demonstrate the generalization ability of the proposed detection pipeline, further experiments are performed on the Vehicle Detection in Aerial Imagery (VEDAI) [Raz16] dataset, which is a common benchmark dataset for vehicle detection in aerial imagery, and on recently published datasets, i.e., DOTA [Xia18], ITCVD [Yan18] and xView [Lam18]. Thus, different GSDs, object sizes, object types, backgrounds and varying number of objects per frame are taken into account. Note that the experiments on the latter datasets are

conducted in a qualitative manner to examine the transferability of the proposed detection pipeline. Quantitative experiments are not performed due to the late availability of the datasets and the partially poor annotation quality. The ISPRS 2D Semantic Labeling Challenge Potsdam dataset [ISP] - a common benchmark dataset for semantic labeling of aerial imagery - is chosen as main dataset for evaluating the effect of integrating semantic labeling on the detection performance.

Table 4.1: Vehicle detection in aerial imagery datasets by release year. Center refers to annotations with only the center coordinates of an instance provided, while BB is short for bounding box, which is either axis-aligned or oriented. The number of instances is limited to vehicles, as further categories, e.g., buildings, bridges, harbors etc., are not in the scope of this thesis.

Dataset	Annotation	Number of Images	Image Width (in pixels)	Number of Instances	GSD (in cm)
TAS [Hei08]	a.-a. BB	30	792	1,319	-
NWPU [Che14b]	a.-a. BB	800	~1,000	477	8-200
UCAS-AOD [Zhu15]	orient. BB	910	~1,000	2,819	-
DLR 3K [Liu15]	orient. BB	20	5,616	14,235	~13
VEDAI [Raz16]	orient. BB	1,268	1,024	2,950	12.5
COWC [Mun16]	center	53	2k-19k	32,716	15
DOTA [Xia18]	orient. BB	2,806	800-13k	~180k	10-100
ITCVD [Yan18]	a.-a. BB	173	5,616	29,088	10
xView [Lam18]	a.-a. BB	1,127	700-4k	~250k	30

DLR 3K Munich Vehicle Aerial Image Dataset

The DLR 3K Munich Vehicle Aerial Image Dataset, in the following termed DLR 3K dataset, is acquired at a height of 1000 m above the ground over Munich, Germany and comprises mainly urban and residential areas as visualized in Figure 4.1. The DLR 3K dataset contains 20 images with a resolution

of 5616 × 3744 pixels and a GSD of approximately 13 cm, whereby GT annotations provided in form of oriented bounding boxes for different vehicle types, e.g., car, truck and trailer, are only available for 10 images. For the experiments within this thesis, the images with available GT annotations are split into 8 training and 2 test images. Each image is divided into tiles of 936 × 936 pixels. As the deep learning based detection frameworks employed in this thesis require at least one object per image, images without any object are removed from the training set yielding 140 image tiles for training. Due to the limited number of annotations for most vehicle types, only the classes car and van are considered. Following [Liu15], the two classes are merged into a single vehicle class. Furthermore, all oriented bounding boxes are converted to axis-aligned bounding boxes according to [Sak17, Som17c, Tan17]. On average, the mean bounding box dimensions are 28.2 ± 8.2 × 28.3 ± 8.7 pixels.

Figure 4.1: Illustrative examples of the DLR 3K Munich Vehicle Aerial Image Dataset [Liu15].

Figure 4.2: Illustrative examples of the Vehicle Detection in Aerial Imagery dataset [Raz16].

Vehicle Detection in Aerial Imagery Dataset

The Vehicle Detection in Aerial Imagery (VEDAI) dataset comprises satellite images of the Utah AGRC[1], which are acquired over Utah, US during spring 2012. The raw images have four uncompressed channels (RGB and near IR), whereby only the RGB channels are used in this thesis. As shown in Figure 4.2, the images comprise varying backgrounds such as agrarian, rural and urban areas. The VEDAI dataset comprises in total 1268 images of size 1024 × 1024 pixels and a GSD of 12.5 cm. In addition, a down-scaled version of the images is available with a GSD of 25 cm. The two versions are referred to as large-size color images (LCIs) and small-size color images (SCIs), respectively. GT annotations are provided in form of oriented bounding boxes for nine vehicle types, whereby cars, pick-ups and vans are summarized as small land vehicles. In the following, the first half of images is used as training data and the second half for testing. In accordance with experiments on the DLR 3K dataset, all oriented bounding boxes are converted to axis-aligned bounding boxes. Due to the limited number of annotations for large vehicles such as boats and airplanes, only the small land vehicles are considered similar to

[1] https://gis.utah.gov/

[Sak17]. VEDAI comprises in average 2.0 objects per image, which is considerably less compared to DLR 3K. The mean bounding box dimensions are 33.6 ± 11.7 × 33.6 ± 11.9 and 16.8 ± 5.8 × 16.8 ± 5.9 pixels for LCI and SCI, respectively.

DOTA

Figure 4.3: Illustrative examples of the DOTA dataset [Xia18].

The DOTA dataset is composed of 2806 images that are collected from multiple sensors and platforms, i.e., Google Earth, satellite JL-1 and satellite GF-2 of the China Centre for Resources Satellite Data and Application. The image sizes range from 800 × 800 to 4000 × 4000 pixels and the GSD, which is provided for each image separately, varies between 10 and 100 cm. As depicted in Figure 4.3, the dataset comprises images with differing scenarios and exhibits a wide variety of scales and orientations. GT annotations of 15 categories are provided in form of both oriented and axis-aligned bounding boxes for 1869 images. Note that the poor annotation quality, especially the high number of missing annotations in case of the categories small vehicle and large vehicle, obstruct an expressive quantitative evaluation [Aca18, Waq19].

ITCVD

Figure 4.4: Illustrative examples of the ITCVD dataset [Yan18].

The ITCVD dataset is acquired at a height of 330 m above the ground over Enschede, Netherlands. The dataset comprises in total 173 images with a resolution of 5616 × 3744 pixels whereof 46 images are taken in nadir view with a GSD of 10 cm. The remaining images taken in oblique view with a tilt angle of 45 degrees are not considered for the experiments within this thesis. Similar to DLR 3K, the dataset comprises mainly urban and residential areas as visualized in Figure 4.4. Note that the images exhibit stronger parallax effects compared to the other datasets due to the low acquisition altitude. GT annotations are provided in form of axis-aligned bounding boxes for all vehicles, which are summarized into a single category.

xView

The xView datasets comprises 1127 images from WorldView-3 satellites with a GSD of 30 cm. The image dimensions are in the range between 700 and 4000 pixels. The images are acquired over different continents including Australia,

Africa, Asia, Europe, Middle and South America and thus, include a large variety of different scenarios and objects (see Figure 4.5). GT annotations of 60 categories including buildings, vehicles and mini-scenes, e.g., shipping container lot, are provided in form of axis-aligned bounding boxes for 846 images. The most GT annotations either belong to category building or category small car because of their prevalence in densely populated areas.

Figure 4.5: Illustrative examples of the xView dataset [Lam18].

ISPRS 2D Semantic Labeling Challenge Potsdam dataset

In literature, there exist multiple aerial semantic labeling datasets, whereby most publicly available datasets only provide annotations for an individual class such as roads or building footprints [Mni13, Mag17, Van18, Dem18]. Due to the missing annotations for occurring vehicles, these datasets are not appropriate for the task of vehicle detection. In contrast, the 2018 IEEE GRSS Data Fusion Challenge dataset [GRS18], the ISPRS 2D Semantic Labeling Challenge Vaihingen dataset [ISP], and the ISPRS 2D Semantic Labeling Challenge Potsdam dataset [ISP] provide annotations for multiple categories including vehicles. While the 2018 IEEE GRSS Data Fusion Challenge dataset

and the ISPRS 2D Semantic Labeling Challenge Vaihingen dataset are not considered because of the low GSD, i.e., 50 cm, and used sensor bands, i.e., IR, red and green, respectively, the ISPRS 2D Semantic Labeling Challenge Potsdam dataset is chosen as main dataset for evaluating the effects of integrating semantic labeling on the detection performance.

The ISPRS 2D Semantic Labeling Challenge Potsdam dataset, in the following referred to as Potsdam dataset, consists of 38 patches with a resolution of 6000 × 6000 pixels and a GSD of 5 cm. The Potsdam dataset is split into 24 patches for training and 14 patches for testing. According to [She16], the training set is divided into two subsets: one for training and one for validation. Pixel-wise semantic annotations of six categories are provided as shown in Figure 4.6. These categories are *impervious surface*, *building*, *low vegetation*, *tree*, *car*, and *clutter*. The images collected over Potsdam, Germany, mainly comprise urban areas showing large building blocks, narrow streets and dense settlement structures. For each patch, RGB imagery, IR imagery and a digital surface model (DSM) are provided. As the latter two are generally not available for aerial imagery, only RGB imagery is considered for the experiments in the context of this thesis.

For all experiments, if not stated otherwise, the original image patches are cropped into tiles of size 600 × 600 pixels. As required for Faster R-CNN, GT bounding boxes are generated by fitting axis-aligned boxes around each segment labeled as *car*. To refine the GT annotations, split and merged GT annotations are manually adjusted and GT annotations for missed cars due to inaccurate labeling are added (see Figure 4.7)[1]. Overall, the number of annotated GT instances in the training and validation set is 5022 and 1607, respectively.

[1] The generated GT annotations are made publicly available at s.fhg.de/semseg-avss2017

Figure 4.6: Example patch of the Potsdam dataset (left) and the corresponding semantic labeling mask (right) with pixel-wise semantic annotations of six categories: *impervious surface* (white), *building* (blue), *low vegetation* (cyan), *tree* (green), *car* (yellow), and *clutter* (red).

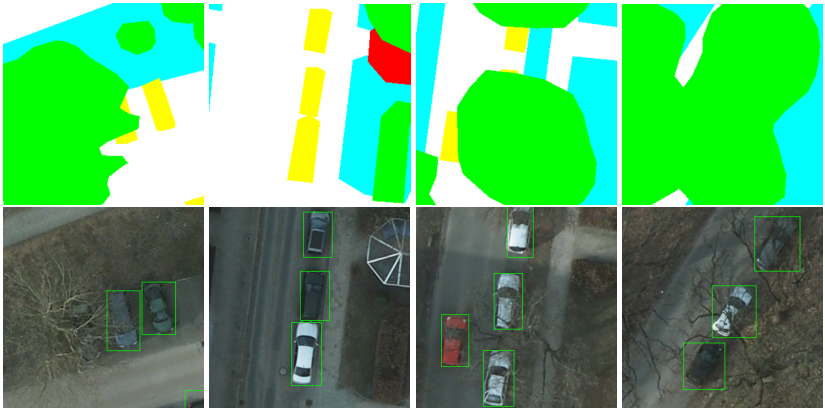


Figure 4.7: Examples for semantic labeling masks (top row) resulting in split (1st column) and merged (2nd column) GT annotations that are manually adjusted (bottom row) and examples for semantic labeling masks with vehicles beneath a tree labeled as category *tree* (3rd and 4th column) whose annotations are manually added.

4.2 Evaluation Metrics and Protocol

In the following, common metrics to evaluate the performance of the proposed detection pipeline, the region proposal generation, i.e., RPN, and the semantic labeling approaches to integrate semantic context information are introduced.

Object Detection

In this thesis, average precision (AP) [Sal83], which is a common metric to measure the detection performance, is applied following the evaluation protocol introduced in [Eve10]. To compute the AP, the detection results are compared against GT. The detection results and GT are provided as bounding boxes with an associated class label. The detection results further comprise for each bounding box the probability for the assigned class, which is denoted as confidence score.

The confidence score and the Intersection over Union (IoU), also referred to as *Jaccard coefficient*, are used as criteria to determine whether a detection is considered as correct detection, termed *true positive* (TP), or not. The IoU is defined as the area of the intersection divided by the area of the union of a predicted bounding box and a GT box:

$$IoU = \frac{A_{pred} \cap A_{GT}}{A_{pred} \cup A_{GT}} . \tag{4.1}$$

A detection is considered as TP, if it satisfies two conditions: the confidence score is higher than a given threshold τ and the IoU is equal to or greater than 0.5, which is referred to as *PASCAL criterion*[1]. Detections that do not fulfill both conditions are termed *false positives* (FPs). Note that, if multiple detections correspond to the same GT instance, only the one with the highest

[1] In case of multiple classes, which is not the case in this thesis, the predicted class has to match furthermore the class label of the corresponding GT.

confidence score counts as a TP, while the remaining detections are considered as FPs. GT instances without an assigned detection are referred to as *false negatives* (FNs).

Precision P - the percentage of correct detections - is defined as the number of TPs divided by the sum of TPs and FPs:

$$P = \frac{|TP|}{|TP| + |FP|} . \tag{4.2}$$

Recall R is the detection rate defined as the number of TPs divided by the sum of TPs and FNs:

$$R = \frac{|TP|}{|TP| + |FN|} . \tag{4.3}$$

Both measures are in the range between 0 and 1. The precision and recall pair for a fixed confidence threshold τ is termed operating point. Higher values for τ generally result in higher precision and lower recall rates and vice versa. By varying τ, different operating points are received, yielding the precision-recall curve (PRC) as exemplarily depicted in Figure 4.8.

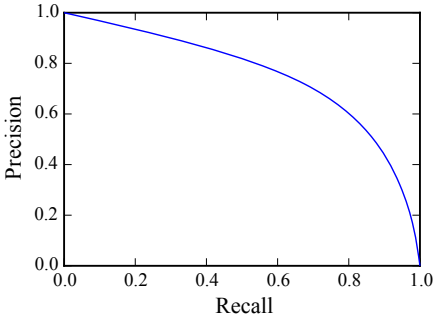


Figure 4.8: Precision-recall curve.

AP is defined as the area under this curve:

$$AP = \int_0^1 P(R)\,\mathrm{d}R\,. \tag{4.4}$$

In practice, AP is computed by summarizing the interval between two successive recall values R_i and R_{i+1} multiplied with the interpolated precision $\tilde{P}(R_{i+1})$:

$$AP = \sum_{i=0}^{n-1} (R_{i+1} - R_i)\tilde{P}(R_{i+1})\,, \tag{4.5}$$

where n is the number of unique recall values arranged in an ascending order. The interpolated precision $\tilde{P}(R_{i+1})$ is given by the maximum precision for any recall $\tilde{R}$ equal to or greater than R_{i+1}:

$$\tilde{P}(R_{i+1}) = \max_{\tilde{R} \geq R_{i+1}} (P(\tilde{R}))\,. \tag{4.6}$$

Object Proposals

Following [Hos15], the effectiveness of object proposals is examined by plotting the recall for the object proposals with respect to various IoU thresholds used to accept GT instances as recalled (see Figure 4.9). To this end, only the n_o object proposals with the highest likelihood of the presence of an object are used, whereby n_o is a hyper-parameter that controls the number of object proposals considered for classification. To measure the localization quality of object proposals, the average best overlap (ABO) is calculated by averaging the best IoU between each GT annotation $a_i \in \mathcal{A}$ and the corresponding set of object proposals $\mathcal{O}$:

$$ABO = \frac{1}{|\mathcal{A}|} \sum_{a_i \in \mathcal{A}} \max_{o_j \in \mathcal{O}} IoU(a_i, o_j)\,. \tag{4.7}$$

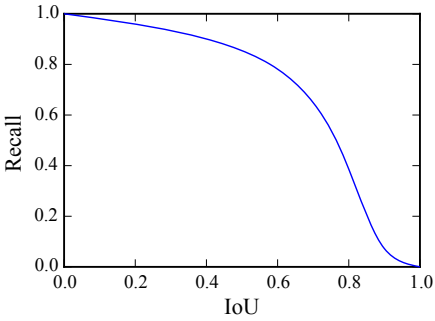


Figure 4.9: Recall versus IoU threshold curve.

Semantic Labeling

To evaluate the semantic labeling results, F1-score computed for each class and overall accuracy showing the percentage of correctly labeled pixels are used as evaluation metrics following the 2D Semantic Labeling Contest protocol[1]. Both metrics are derived from a pixel-based confusion matrix. To this end, the confusion matrices for each tile are accumulated. F1-score is the harmonic mean of precision and recall defined as

$$F1\text{-score} = 2\frac{P \cdot R}{P + R}\,. \tag{4.8}$$

To compute precision and recall (see eq. (4.2) and eq. (4.3)) per class, the number of TP pixels is derived from the main diagonal elements of the confusion matrix, while the number of FP pixels is the sum per column and the number of FN pixels is the sum per row, excluding the main diagonal element. The overall accuracy is computed by normalizing the trace of the confusion matrix. Note that a three-pixel boundary between GT regions with different labels is ignored during evaluation to reduce the impact of uncertain border definitions due to the annotation procedure.

[1] http://www2.isprs.org/commissions/comm3/wg4/semantic-labeling.html

Inference Time

Besides the detection accuracy, the inference time is another key factor to judge the detection performance. In the context of this thesis, two different devices representing a server and a desktop setup are used in order to measure the inference time. The desktop setup is chosen to account that many real-world applications have to do without powerful server setups. The key characteristics of each device are given in Table 4.2. All inference time measurements are conducted through the *pycaffe* Python interface for Caffe. If not stated otherwise, the inference time is reported in milliseconds (ms) averaged over the complete test set of the respective dataset. Note that all time measurements exclude preprocessing steps, e.g., loading the image, as these steps can be done asynchronously, while waiting for the GPU to finish the current forward pass. In advance of each timing, 10 forward passes are performed to warm-up the GPU kernels.

Table 4.2: Overview of devices used for runtime measurements.

	Server	Desktop
CPU	48×Intel Xeon E5-2650 v4 @ 2.20GHz	12×Intel Core i7-7800X CPU @ 3.50GHz
RAM	256GB	32GB
GPU	GTX TITAN X (Pascal), 12GB	GTX 1050 Ti (Pascal), 4GB

5 Base Framework

In this chapter, the functional principle of the utilized base framework, i.e., Faster R-CNN, is introduced. For this, the main detector components and the implementation details are extensively described. Furthermore, adaptations in order to account for the characteristics of aerial imagery are proposed and systematically examined to identify concomitant issues.

5.1 Faster R-CNN

Faster R-CNN is chosen as base framework for the detection pipeline proposed in this thesis, because of its superior detection accuracy compared to other deep learning based detectors, in particular for small object instances. The capability of accurately detecting small object instances is even more essential in case of aerial imagery, which typically comprises objects in the range of a few pixels due to its low spatial resolution as introduced in Section 4.1. The functional principle of Faster R-CNN, which mainly comprises two modules, is schematically depicted in Figure 5.1. The first module termed RPN generates a set of candidate regions, which is forwarded to a subsequent module denoted as classification stage. The two modules are merged into a single network by sharing a sequence of convolutional layers termed feature extractor. Fundamental basics of both modules as well as implementation details are introduced in the following.

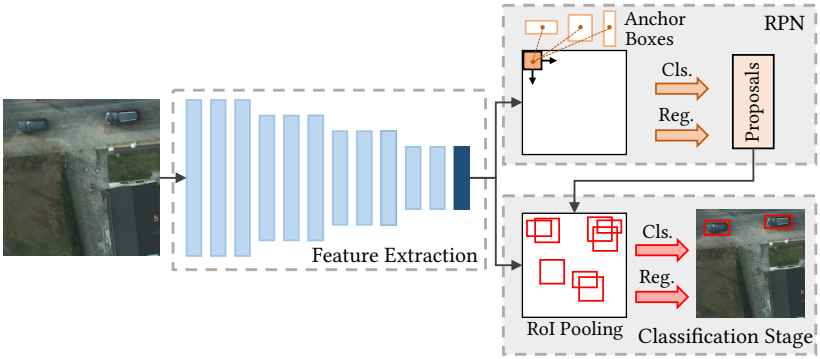


Figure 5.1: Functional principle of the Faster R-CNN, which conducts object detection in two stages: an initial stage that generates a set of candidate regions termed RPN and a subsequent classification stage. Both stages share a sequence of convolutional layers and employ the output of the last convolutional layer as feature map (highlighted in dark blue).

5.1.1 Region Proposal Network

The Region Proposal Network is a deep learning based object proposal method whose functional principle is depicted in Figure 5.2. Purpose of the RPN is the localization of candidate regions that are likely to contain an object. To this end, the RPN comprises a small network, which is shifted in sliding window manner across the output of the last shared convolutional layer used as feature map. The small network is comprised of a 3×3 convolutional layer followed by two sibling fully connected layers: one for bounding box regression (*reg* layer) and one for classification (*cls* layer). In practice, the fully connected layers are implemented with two sibling 1×1 convolutional layers.

To conduct bounding box regression, *anchor boxes* centered at each sliding window location, i.e., feature map location, are utilized as reference boxes. In order to account for various object scales, anchor boxes with different aspect ratios and sizes are employed, yielding k anchor boxes per sliding window location. Thus, the *reg* layer has $4k$ outputs encoding offsets for each anchor box used to predict the coordinates for the respective candidate region, while the *cls* layer outputs $2k$ confidence scores about the presence of an object.

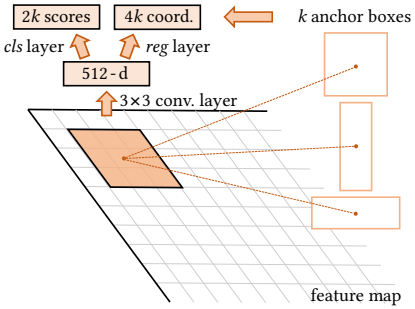


Figure 5.2: Functional principle of the Region Proposal Network. The RPN comprises a small network that is shifted in sliding window manner across the used feature map. The small network is comprised of a 3×3 convolutional layer followed by a classification layer and bounding box regression layer. Anchor boxes centered at each sliding window location are utilized as reference for the bounding box regression.

For training the RPN, a set of anchors termed mini-batch $\mathcal{B}_{RPN}$ is randomly sampled per image, whereby each anchor is assigned by a class label p_i^* indicating the affiliation to an object (positive anchor) or not (negative anchor). The class label p_i^* is 1 in case of positive anchors and 0 in case of negative anchors. Anchors with the highest IoU to a GT box as well as all anchors possessing an IoU above 0.7 are assigned as positive anchors, whereas anchors bearing an IoU below 0.3 are assigned as negative anchors. Anchors being neither positive nor negative are not considered for training.

Both classification and bounding box regression are trained jointly using a multi-task loss defined as[1]

$$L_{RPN} = \sum_{i \in \mathcal{B}_{RPN}} L_{cls}(p_i, p_i^*) + \sum_{i \in \mathcal{B}_{RPN}} p_i^* L_{reg}(\boldsymbol{t}_i, \boldsymbol{t}_i^*), \tag{5.1}$$

where i is the index of an anchor in the current mini-batch, p_i denotes the predicted probability of anchor i is associated to an object, and $\boldsymbol{t}_i$ and $\boldsymbol{t}_i^*$ are vectors representing the parameterized coordinates of the predicted bounding

[1] Note that the given objective function is in accordance with the implementation used in the scope of this thesis and thus, slightly differs from the objective function given in [Ren15].

box and the associated GT box. The classification loss L_{cls} is the logarithmic loss for two classes computed by:

$$L_{cls}(p_i, p_i^*) = -p_i^* log p_i - (1 - p_i^*) log(1 - p_i). \tag{5.2}$$

Note that for simplicity the classification loss is implemented as a two-class softmax layer [Ren15]. To ensure that the regression loss is only activated for positive anchors, it is weighted by the class label p_i^*. The regression loss is given by:

$$L_{reg}(\boldsymbol{t}_i, \boldsymbol{t}_i^*) = \sum_{j \in \{x,y,w,h\}} L_1^{smooth}(t_{ij} - t_{ij}^*), \tag{5.3}$$

whereby smooth L_1 loss [Gir15], which is more robust to outliers compared to L_2 loss, is applied:

$$L_1^{smooth}(x) = \begin{cases} 0.5x^2, & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise.} \end{cases} \tag{5.4}$$

The vectors representing the parameterized coordinates of the predicted bounding box and the associated GT box are defined as follows:

$$\begin{aligned} t_x &= (x - x_a)/w_a, & t_y &= (y - y_a)/h_a, \\ t_w &= log(w/w_a), & t_h &= log(h/h_a), \\ t_x^* &= (x^* - x_a)/w_a, & t_y^* &= (y^* - y_a)/h_a, \\ t_w^* &= log(w^*/w_a), & t_h^* &= log(h^*/h_a). \end{aligned} \tag{5.5}$$

Here, x, y, w, and h are the center coordinates, width and height of the predicted bounding box. The coordinates and dimensions of the corresponding anchor box and GT box are indicated by a subscripted a and a superscripted $*$, respectively.

During deployment, the actual bounding box coordinates of each region proposal are computed in a subsequent layer denoted as *proposal layer* by adding the predicted offsets to the coordinates of the respective anchor box. To generate the final set of region proposals, the region proposals are sorted with

respect to the predicted confidence score for the presence of an object and non-maximum suppression (NMS) is applied to remove redundant region proposals.

5.1.2 Classification Stage

The functional principle of the classification stage, which is in essence the Fast R-CNN detector [Gir15], is illustrated in Figure 5.3. The classification stage takes a pre-defined number of candidate regions, i.e., region proposals with the highest confidence scores, as input. Each candidate region denoted as region of interest (RoI) is projected onto the same feature map as used for the RPN. By conducting max pooling, the RoI pooling layer converts the corresponding features inside the respective candidate region into a feature map with fixed spatial extent. The feature map is then fed into a sequence of fully connected layers branching into two sibling fully connected layers: one for classification and one for bounding box regression similar to the RPN. The *cls* layer outputs *c+1* confidence scores for the *c* classes and the background class, while the *reg* layer outputs 4 values for each class, which encode offsets to the respective candidate region. The applied sub-network is also referred to as *classification head*.

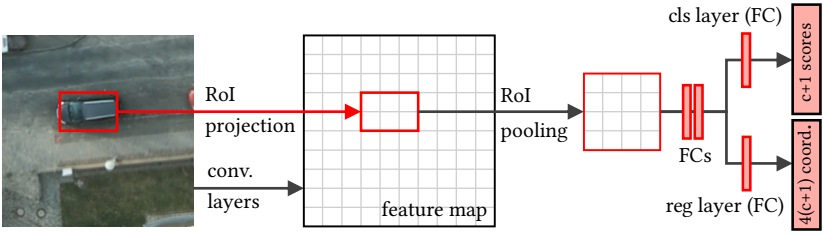


Figure 5.3: Functional principle of the classification stage. Each region of interest, i.e., candidate region, is projected onto the feature map. The corresponding features inside the respective candidate region are converted via RoI pooling into a feature map with fixed spatial extent, which is fed into a sequence of fully connected layers (FCs). The two final sibling fully connected layers output confidence scores and per-class regression offsets.

For training the classification stage, a set of candidate regions also referred to as mini-batch $\mathcal{B}_{CLS}$ is sampled from the set of candidate regions forwarded from the RPN. Candidate regions with an IoU of at least 0.5 to a GT object are associated with the corresponding class label u_i>0, otherwise assigned to the background u_i=0. The ratio between candidate regions associated with an object or assigned to background is set to 1:3 by subsampling the background regions. A multi-task loss analogous to eq. (5.1) is applied to jointly train the classification and bounding box regression of the classification stage:

$$L_{CLS} = \sum_{i \in \mathcal{B}_{CLS}} L_{cls}(p_i, u_i) + \sum_{i \in \mathcal{B}_{CLS}} I(u_i) L_{reg}(\boldsymbol{t}_i^u, \boldsymbol{v}_i). \tag{5.6}$$

The classification loss L_{cls} is logarithmic loss whereby p_i is the probability distribution for candidate region i and u_i is the true class label. The regression loss L_{reg} is smooth L_1 loss (see eq. (5.3)), whereby $\boldsymbol{t}_i^u$ and $\boldsymbol{v}_i$ are vectors representing the parameterized coordinates of the predicted bounding box associated to class u_i and the current GT box v_i. Note that GT boxes are only given for true classes. Thus, the regression loss is weighted with the indicator function $I(u_i)$, which is 1 if u_i is the true class and 0 for all other classes and background.

5.1.3 Implementation Details

As aforementioned, a sequence of convolutional layers is used as feature extractor for Faster R-CNN, whereby the output of the last convolutional layer serves as feature map for both stages. For this purpose, VGG16 [Sim14] is employed by default as base network. Table 5.1 schematically depicts the architecture of VGG16, which is comprised of 13 convolutional layers arranged in sequences of 2 and 3, respectively, followed by 3 fully connected layers. ReLU is applied as activation function after each convolutional and fully connected layer, while spatial pooling is carried out by max pooling layers after the 2^{nd}, 4^{th}, 7^{th}, 10^{th} and 13^{th} convolutional layer. The 13 convolutional layers are shared between both stages as feature extractor and the output of the last convolutional layer, denoted as *conv5_3*, is used as feature map. The fully connected layers are adopted for the classification stage. To this end, the output

dimensions of the RoI pooling layer, which extracts features for each candidate region, is set to 7×7 as required for the first fully connected layer. The last fully connected layer, which comprises 1000 outputs, i.e., one for each class of the ImageNet classification benchmark dataset, is replaced by two sibling fully connected layers. In case of one class, i.e., vehicle, the output dimensions are set to 2 and 8, respectively.

Table 5.1: Schematic structure of VGG16 used by default as feature extractor for Faster R-CNN. $d \times d$ specifies the input image dimension, which is 224×224 in case of ImageNet.

Layer Type	Kernel Size	Stride, Pad	Output Dimension
convolution	$3 \times 3 \times 64$	1, 1	$d \times d \times 64$
convolution	$3 \times 3 \times 64$	1, 1	$d \times d \times 64$
max pooling	2×2	2, 0	$d/2 \times d/2 \times 64$
convolution	$3 \times 3 \times 128$	1, 1	$d/2 \times d/2 \times 128$
convolution	$3 \times 3 \times 128$	1, 1	$d/2 \times d/2 \times 128$
max pooling	2×2	2, 0	$d/4 \times d/4 \times 128$
convolution	$3 \times 3 \times 256$	1, 1	$d/4 \times d/4 \times 256$
convolution	$3 \times 3 \times 256$	1, 1	$d/4 \times d/4 \times 256$
convolution	$3 \times 3 \times 256$	1, 1	$d/4 \times d/4 \times 256$
max pooling	2×2	2, 0	$d/8 \times d/8 \times 256$
convolution	$3 \times 3 \times 512$	1, 1	$d/8 \times d/8 \times 512$
convolution	$3 \times 3 \times 512$	1, 1	$d/8 \times d/8 \times 512$
convolution	$3 \times 3 \times 512$	1, 1	$d/8 \times d/8 \times 512$
max pooling	2×2	2, 0	$d/16 \times d/16 \times 512$
convolution	$3 \times 3 \times 512$	1, 1	$d/16 \times d/16 \times 512$
convolution	$3 \times 3 \times 512$	1, 1	$d/16 \times d/16 \times 512$
convolution	$3 \times 3 \times 512$	1, 1	$d/16 \times d/16 \times 512$
max pooling	2×2	2, 0	$d/32 \times d/32 \times 512$
fully connected			4096
fully connected			4096
fully connected			1000

Joint training of the RPN and classification stage is conducted for all experiments within this thesis, which is 1.5 times faster than alternating optimization at similar detection performance. For this purpose, L_{RPN} and L_{CLS} are equally weighted:

$$L_{Faster\ R\text{-}CNN} = L_{RPN} + L_{CLS}. \tag{5.7}$$

Each model is trained for 60,000 iterations using SGD and an initial learning rate of 0.001. The learning rate is reduced by a factor of 10 every 20,000 iterations. The weight decay and momentum are set to 0.0005 and 0.9, respectively. Weights pre-trained on ImageNet are used to initialize the convolutional layers and the first two fully connected layers in the classification stage. All other layers are randomly initialized by using the Gaussian weight filler method according to [Ren15]. If not stated otherwise, 3 scales and 3 aspect ratios are used for the anchor boxes of the RPN. The aspect ratios are fixed to 1:1, 1:2 and 2:1 to account for the different orientations of vehicles in overhead imagery. The minimum dimension to accept generated region proposals for classification is set to 4 pixels. Note that by default, only candidate regions whose dimensions exceed 16 pixels are considered for classification, which may lead to a high number of missed detections as candidate regions corresponding to small objects, partially occluded objects and objects at image edges are filtered out.

During deployment, NMS is applied on the 10,000 region proposals exhibiting the highest confidence scores. The overlap threshold value used for NMS is set to 0.7. The top-2000 ranked region proposals after NMS are then forwarded to the classification stage. Hence, redundant region proposals are removed and the number of region proposals to classify and consequently the computational costs are reduced. To remove duplicate detections, NMS with an overlap threshold of 0.3 is applied on the final detections. The settings for the NMS are based on preliminary experiments reported in [Som18b].

5.2 Adaptation to Aerial Imagery

Deep learning based detection frameworks, such as Faster R-CNN, are typically developed and designed for benchmark detection datasets that considerably differ from aerial imagery (see Chapter 3). Due to these differences, in particular in object dimensions, deep learning based detection frameworks are not directly applicable for vehicle detection in aerial imagery. In order to account for the characteristics of aerial imagery, several modifications are examined in detail for the first time [Som17b, Som18b]. In the following, the conducted modifications to Faster R-CNN, i.e., reducing the feature map resolution and adapting the anchor box settings, are analyzed and the accompanying effects are discussed.

5.2.1 Feature Map Resolution

By using the original VGG16 as feature extractor for Faster R-CNN, top performing results are achieved on PASCAL VOC indicating the high suitability for benchmark datasets [Ren15]. Due to spatial pooling, the dimensions of the output of the last convolutional layer used as feature map are only 1/16 of the input image (see Table 5.1), which is sufficient for accurately localizing objects in benchmark datasets as shown in Figure 5.4. To this end, the activations of three filters from the employed feature map, i.e., *conv5_3*, and the corresponding detection results are depicted exemplarily for PASCAL VOC. Note that the applied Faster R-CNN model is trained with the settings proposed for PASCAL VOC. During inference, the input image is rescaled so that the shorter size equals 600 pixels analogous to [Ren15]. The activations are normalized to values between 0 and 1, whereby higher values correspond to stronger activations. Multiple feature map pixels overlap with the objects due to the large object dimensions. The activations of the three filters, which respond to different object parts, e.g., windshield, show that the feature map resolution is even sufficient to map the contours of the object parts and consequently yield accurately aligned bounding boxes around the objects.

Figure 5.4: Activations of three filters from *conv5_3* used as feature map and the corresponding detection results on PASCAL VOC indicate that the feature map resolution is sufficient to accurately localize objects. The activations are normalized to values between 0 and 1.

Figure 5.5: Activations of three filters from *conv5_3* used as feature map and the corresponding detection results indicate that the feature map resolution is not sufficient to accurately localize small objects as in case of DLR 3K. The activations are normalized to values between 0 and 1.

In contrast, employing *conv5_3* as feature map in case of aerial imagery, which comprises objects in the range of only a few pixels, results in inaccurately aligned bounding boxes. A reason for the poor alignment is the coarse feature map resolution as exemplarily shown for DLR 3K in Figure 5.5. For this, the applied Faster R-CNN model is trained with identical settings as for PASCAL VOC on DLR 3K. Note that the number of outputs in the classification stage are adapted to 2 and 8 as described in Section 5.1.3. The visualized activations exhibit that only few feature map pixels overlap with the small object in the sample image, whereby several of these feature map pixels mainly cover the background. Hence, the feature map resolution is not sufficient to accurately localize such small objects, which leads to poorly located as well as duplicate detections.

To address this issue, the feature map resolution is systematically increased by removing piecewise sequences of convolutional layers from the original VGG16 network. Using the output of the 10^{th} convolutional layer termed *conv4_3* and 7^{th} convolutional layer termed *conv3_3* results in feature maps whose dimensions are 1/8 and 1/4 of the input image, respectively[1]. The effect of higher feature map resolutions is given by quantitative results in Table 5.2. For this, all experiments are performed on the DLR 3K dataset and AP is used as evaluation metric (see Section 4.2). Each model is trained with the settings specified in Section 5.1.3. Furthermore, the adapted anchor box settings introduced in the subsequent section are adopted. As expected, the AP considerably increases with higher feature map resolutions. The best AP is achieved for *conv3_3*, which outperforms *conv5_3* by almost 30% in AP. However, using the output of *conv2_2* as feature map, which exhibits an even higher resolution, shows no further improvement. Instead, the AP slightly drops due to an increased number of false positive detections, as the semantic context information of the employed feature map becomes less. The observed improvements are in accordance with findings reported in [Sak17]. The best results for vehicle detection on the VEDAI dataset were achieved by using

[1] To increase the number of input channels as required for the first fully connected layer of the classification stage, a 1×1 convolutional layer with 512 channels is applied on the output of *conv3_3* that originally comprises 256 channels (see Table 5.1).

conv3_3 as feature map for Faster R-CNN. Furthermore, the improved detection accuracy in other domains, such as pedestrian detection [Zha16] or logo detection [Egg17], confirm the importance of exploiting shallower layers as feature map for an accurate detection of small objects.

Table 5.2: AP for differing feature map resolutions on DLR 3K. The respective feature map resolutions are given with respect to the used input image size, i.e., 936 × 936 pixels.

Feature Map	Resolution	AP (in %)
conv5_3	59 × 59	65.1
conv4_3	117 × 117	90.3
conv3_3	234 × 234	**94.3**
conv2_2	468 × 468	92.7

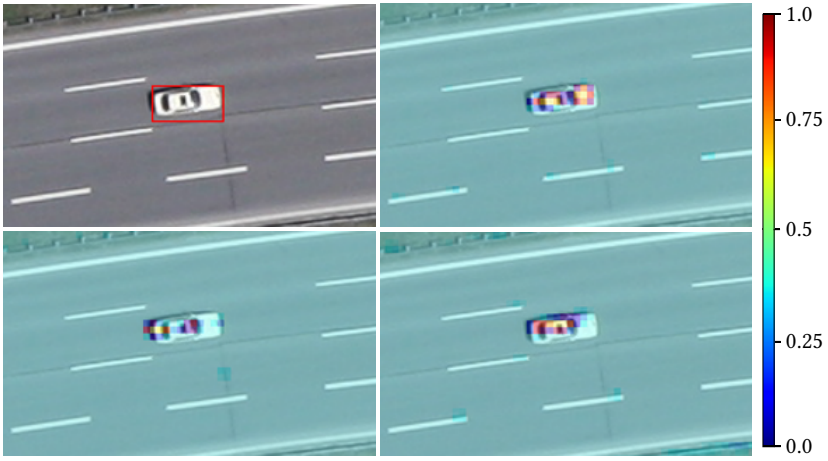
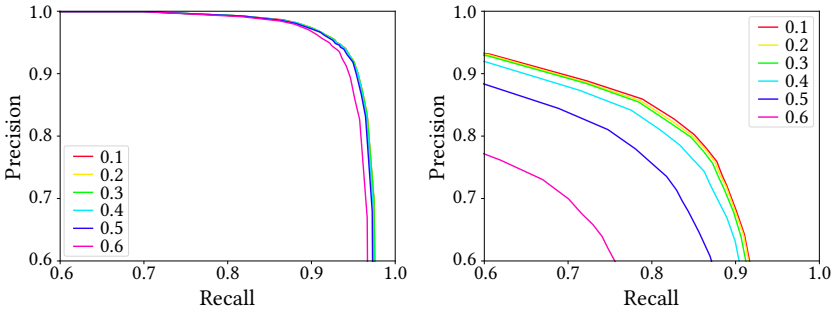


Figure 5.6: Activations of three filters from *conv3_3* used as feature map and the corresponding detection results indicate that the feature map resolution is sufficient to accurately localize even small objects. Note that the activations are normalized to values between 0 and 1.

Visualizing the activations of three filters from *conv3_3* used as feature map and the corresponding detection results qualitatively show that the localization accuracy improves with an increasing feature map resolution (see Figure 5.6). Due to the increased feature map resolution, considerably more feature map pixels overlap with the object in the sample image compared to using *con5_3* as illustrated in Figure 5.5. Thus, even fine object parts such as the windshield are covered by multiple feature map pixels and an accurate prediction of the object boundaries is facilitated.

Analysis of the Localization Accuracy

In the following, a detailed analysis of the detection results is provided to substantiate the impact of the feature map resolution. First, the detection results are examined with respect to the localization accuracy by varying the IoU threshold value used to accept GT objects as recalled. PRCs for the different feature map resolutions and varying IoU thresholds are given in Figure 5.7.



(a) *conv3_3* **(b)** *conv5_3*

Figure 5.7: Precision-recall curves for various IoU thresholds used to accept GT objects as recalled. Exploiting feature maps with higher resolutions results in better localization quality as the performance decreases clearly less with increasing threshold values.

Higher feature map resolutions exhibit fewer variations in detection performance with varying IoU thresholds. For *conv3_3*, the detection performance is only slightly improved with lower IoU threshold values as most detections have a high overlap to the GT annotations. In contrast, the detection performance for *conv5_3* is clearly improved by applying weaker IoU criteria. Hence, employing higher feature map resolutions yields superior localization accuracy.

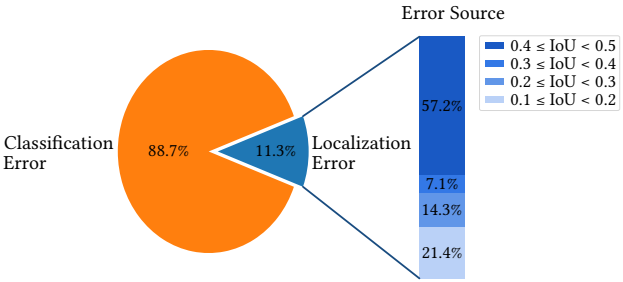


Figure 5.8: Error analysis of false positive detections for *conv3_3*.

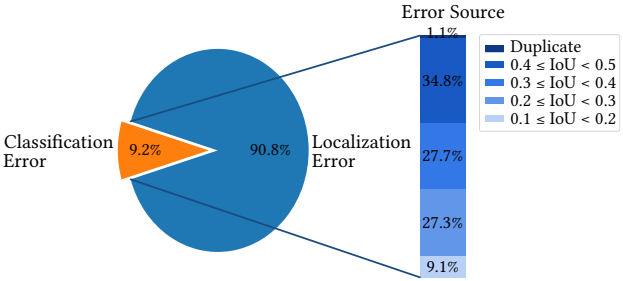


Figure 5.9: Error analysis of false positive detections for *conv5_3*.

The error analysis of false positive detections visualized in Figures 5.8 and 5.9 underlines the improved localization accuracy in case of higher feature

maps. For this, all FPs with a confidence score equal to or greater than 0.5 are distinguished into localization errors and classification errors, respectively. Following [Hoi12], localization errors are duplicate detections and detections with misaligned bounding boxes, i.e., detections possessing an IoU to a GT annotation between 0.1 and 0.5. All other FPs are categorized as classification errors. In case of *conv3_3* only about 10% of all FPs are due to localization errors, whereby most localization errors boast an IoU above 0.4. In contrast, more than 90% of all FPs in case of *conv5_3* are caused by misaligned bounding boxes or duplicate detections. Overall, the number of FPs due to localization errors is reduced by a factor of 46 for *conv3_3*, which confirms the assumption that an increased feature map resolution is necessary to accurately localize tiny objects such as vehicles in aerial imagery.

Impact on the RPN

As described in Section 5.1, the localization of relevant objects is initially done by the RPN, as region proposals that most likely contain a relevant object are identified. The relation between the feature map resolution and the generated region proposals is depicted in Figure 5.10 by means of recall-IoU curves. Therefore, the IoU threshold value used to accept a GT object as covered at least by one region proposal is varied in steps of 0.05 in the range between 0 and 1.0. To compute the recall, only the top-2000 ranked region proposals after NMS are considered, which are equivalent to the region proposals forwarded to the classification stage in the preceding experiments. Using the output of *conv3_3* as feature map results in region proposals exhibiting overall the best overlap to the GT annotations. While the recall values are only marginally lower in case of *conv2_2*, exploiting feature maps with higher resolutions, i.e., using the output of *conv4_3* and *conv5_3*, respectively, delivers candidate regions with clearly worse overlap to the GT annotations. The decline in recall gets more distinctive with higher feature map resolutions, in particular for IoU threshold values above 0.4. Since only region proposals with an overlap above 0.5 are considered as positive samples for the training of the classification stage as described in Section 5.1.2, multiple GT objects are

not adequately covered by region proposals to be classified with high confidence as vehicle. Hence, the probability of missed detections intensifies and consequently the detection performance drops.

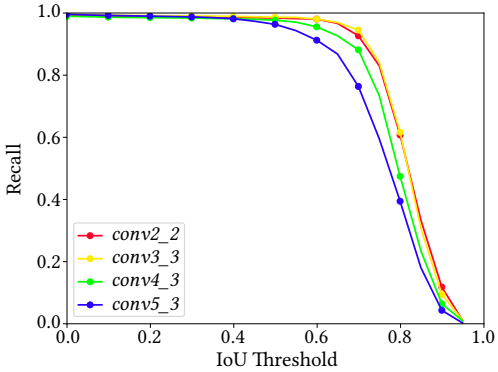


Figure 5.10: Recall-IoU curves for different feature map resolutions. Exploiting feature maps with high resolutions delivers candidate regions with higher overlap to GT annotations.

5.2.2 Anchor Box Settings

The detection accuracy is strongly affected by the quality of candidate regions predicted by the RPN, as only a specified number of candidate regions is forwarded to the classification stage, while all other candidate regions are discarded. Besides the employed feature map (see Section 5.2.1), the quality of candidate regions depends on the configured anchor box settings, i.e., dimensions and aspect ratios of anchor boxes used for bounding box regression [Ren15]. By default, 3 different scales and 3 different aspect ratios are used, resulting in 9 anchor boxes at each location. While the default anchor box scales, yielding box areas of 128^2, 256^2, and 512^2 pixels, are in the range of

objects within the benchmark detection datasets[1], these scales considerably exceed the mean vehicle dimensions in aerial imagery, e.g., 28.2 × 28.3 pixels in case of DLR 3K. Using the default anchor box settings results in clearly lower AP compared to anchor boxes in the range of vehicles within DLR 3K (see Table 5.3). Note that the output of *conv3_3* is employed as feature map due to the findings in Section 5.2.1.

Table 5.3: AP for different anchor box scales and consequently different anchor box areas. Using anchor boxes in the range of vehicles within DLR 3K exhibits clearly improved results compared to the default settings.

Anchor Box Area (in pixels)	AP (in %)
128^2, 256^2, 512^2	92.6
14^2, 28^2, 42^2	**94.3**

To analyze the impact of the anchor box sizes on the detection accuracy, only one anchor box scale is used in the following. The anchor box scale is systematically reduced, yielding box areas in the range between 256^2 and 14^2 pixels. Note that the aspect ratios are retained unchanged due to the different orientations of vehicles in a scene. As given in Table 5.4, the best AP is achieved for an anchor box area of size 28^2, which is roughly equivalent to the mean vehicle dimensions in the DLR 3K dataset. While the AP only slightly decreases for anchor boxes close to the mean vehicle dimensions, the drop in AP considerably increases with anchor boxes clearly exceeding the mean vehicle dimensions. As, in contrast to benchmark datasets, DLR 3K comprises images with a homogenous GSD, the vehicle dimensions exhibit only small variations and thus, using multiple anchor box scales used to account for different object scales is of less importance.

[1] For instance, PASCAL VOC2007 comprises objects with an average size of 143.2 × 148.3 pixels. By default, Faster R-CNN rescales the input images so that the shorter size equals 600 pixels, which results in an effective average object size of 241.3 × 248.7 pixels.

Table 5.4: AP for different anchor box scales and consequently different anchor box areas. Using anchor boxes in the range of vehicles within DLR 3K exhibits the best AP.

Anchor Box Area (in pixels)	AP (in %)
256^2	88.1
112^2	92.8
56^2	93.8
42^2	94.1
28^2	**94.3**
14^2	94.2

Relation between Region Proposal Quality and Detection Accuracy

The impact of the anchor box scales on the quality of generated region proposals is depicted in Figure 5.11 by means of Recall-IoU curves. Using anchor boxes with an area of 28^2 pixels results in region proposals exhibiting overall the best overlap to GT objects. The overlap considerably worsens with anchor boxes that clearly exceed the mean vehicle dimensions. While recall values close to 1 at an IoU threshold of 0.5 are achieved for anchor boxes in the range of the mean vehicle dimensions, about 14% and 38% of the GT objects are not recalled for anchor box areas of 112^2 and 256^2, respectively. As pointed out in Section 5.2.1, only region proposals with an IoU above 0.5 are considered as positive samples for the training of the classification stage. Hence, the number of GT objects that are not adequately covered by region proposals increases with larger margins to the mean vehicle dimensions and thus, the probability of missed detections intensifies.

To analyze the relation between the localization quality of the generated region proposals and detection accuracy more closely, the AP is plotted with respect to the ABO for the different anchor box scales in Figure 5.12. In accordance with above observations, better localization quality is achieved for anchor box areas in the range of the mean vehicle dimensions, while the ABO gets worse with anchor boxes exceeding the mean vehicle dimensions. Similar findings confirming that anchors in the range of present objects yield better detection results are reported in [Ash17].

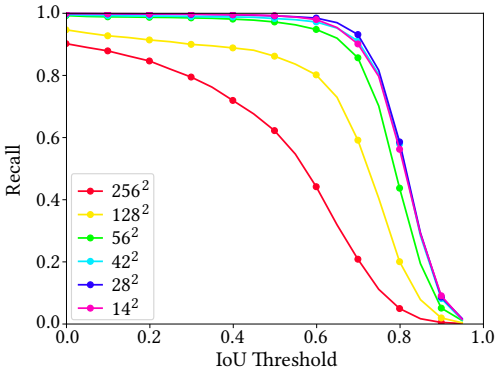


Figure 5.11: Recall-IoU curves for different anchor box scales. Anchor boxes in the range of mean vehicle dimensions result in candidate regions with higher overlap to GT annotations.

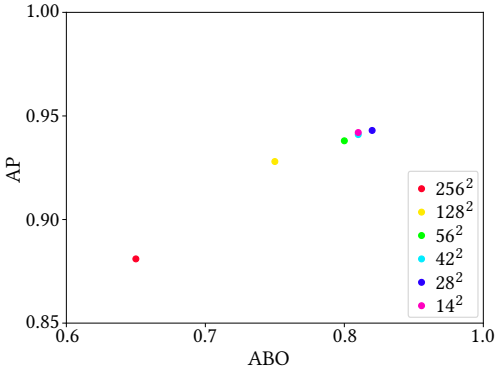
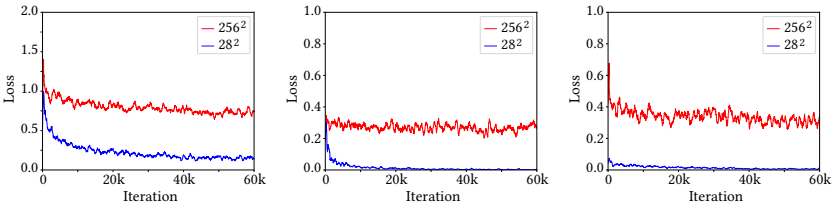


Figure 5.12: Relation between average precision and average best overlap for different anchor box scales. Candidate regions possessing higher ABO result in better AP.

Impact on the Training Behavior

Using anchor box sizes that clearly exceed the mean vehicle dimensions results in poor convergence behavior during training as shown in Figure 5.13. To this end, the overall loss $L_{Faster\ R\text{-}CNN}$ (see eq. (5.7)) and the losses of the RPN, i.e., $L_{cls,RPN}$ and $L_{reg,RPN}$ (see eq. (5.2) and eq. (5.3)), are averaged over

100 iterations. For a mean anchor box area of 28^2 pixels, $L_{Faster\ R\text{-}CNN}$ converges to 0.15, while for a mean anchor box area of 256^2 pixels, $L_{Faster\ R\text{-}CNN}$ only slightly decreases and converges to 0.7. Examining the losses of the RPN shows that in case of a mean anchor box area of 256^2 pixels, both $L_{cls,RPN}$ and $L_{reg,RPN}$ exhibit no convergence behavior, while both losses converge to 0 for a mean anchor box area of 28^2 pixels. A reason for this is the sampling of positive and negative anchors during training as described in Section 5.1.1. To ensure at least one positive anchor per GT instance, anchors with the highest IoU to a GT box are assigned as positive anchors in addition to anchors possessing an IoU above 0.7. However, in case of a mean anchor box area of 256^2 pixels, all anchors exhibit only a small IoU to the nearest GT box as illustrated in Figure 5.14. Thus, all positive anchors and negative anchors possess similar IoUs, which impedes the training of the RPN.



(a) $L_{FasterR-CNN}$ **(b)** $L_{cls,RPN}$ **(c)** $L_{reg,RPN}$

Figure 5.13: Loss curves for a mean anchor box area of 256^2 and 14^2 pixels.

The localization quality of the region proposals that are forwarded to the classification stage is depicted in Figure 5.15. Note that only region proposals with a confidence score above 0.5 are visualized. The generated region proposals (red boxes) confirm the training behavior. In case of an anchor box area of 28^2 pixels (right), the region proposals are located around rectangular structures and all vehicles (green boxes) are covered, which emphasizes that the RPN is capable of correctly identifying regions that are likely to contain an object. In contrast, the region proposals for an anchor box area of 256^2 pixels (left) are

randomly located on road surfaces, as the RPN classifies such areas as regions of interest.

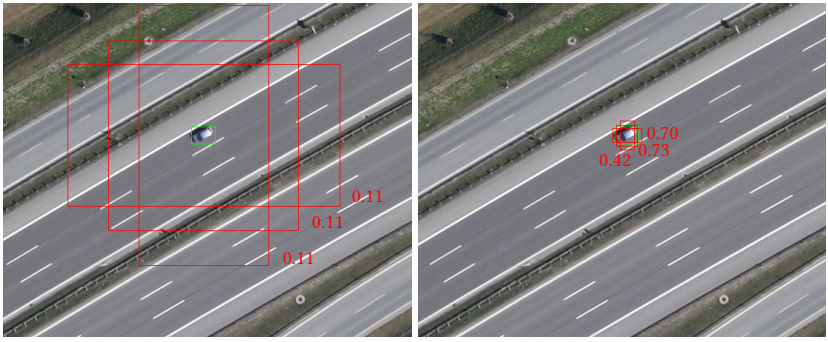


Figure 5.14: Visualization of anchor boxes (red) positioned at the center of a GT object (green). Anchor boxes with a mean anchor box area of 256^2 pixels (left) exhibit a considerably worse IoU to the GT box compared to anchor boxes with a mean anchor area of 28^2 pixels (right).

Figure 5.15: Region proposals (red boxes) for a mean anchor box area of 256^2 pixels (left) and 28^2 pixels (right) that are forwarded to the classification stage as well as the corresponding GT (green boxes). Note that only region proposals with a confidence score above 0.5 are depicted.

During deployment, vehicles that are not adequately covered are likely to yield missed detections. Furthermore, the poorly localized region proposals impede the training of the classification stage, as only region proposals with an IoU of at least 0.5 to a GT object are associated as positive sample (see Section 5.1.2). Using anchor box scales in the range of present objects instead results in region proposals with a high ABO to the GT objects, which facilitates the training of the classification stage and delivers better detection results as reported in Figure 5.12.

5.2.3 Object Dimensions

In the following, the effect of the proposed adaptations with respect to the size of objects present in the aerial imagery is examined by varying the GSD. Because of the uniform GSD of DLR 3K and consequently small variations in vehicle dimensions, the original images with a GSD of 13 cm are rescaled for training and testing by factor 2/3 and 1/2 yielding GSDs of 19.5 and 26 cm, respectively. Hence, the mean object dimensions are reduced to 18.8 × 18.9 and 14.1 × 14.2 pixels. Figure 5.16 shows the object size distributions for the different GSDs.

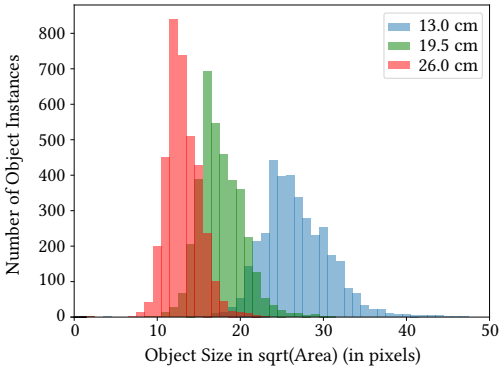


Figure 5.16: Distribution of object instance sizes for different GSDs.

Feature Map Resolution

As described above, increasing the feature map resolution mainly improves the detection accuracy, as coarse feature map resolutions are not appropriate for locating small objects. In the following, the impact of the feature map resolution is examined with respect to the object dimensions. Exploiting higher feature map resolutions shows an improved AP for all GSDs as depicted in Table 5.5. Note that for each GSD, the employed anchor box areas are equivalent to the mean object dimensions (see Table 5.6). Using the output of *conv3_3* as feature map achieves the best AP for all GSDs. Due to smaller object dimensions and consequently fewer feature map pixels that overlap with object instances, the AP decreases with higher GSDs for all feature map resolutions. Note that the effect of decreasing AP with higher GSDs is further intensified by the IoU criterion used to accept GT objects as recalled, which becomes more severe, as even small variations in the predicted bounding box coordinates can lead to a clearly worse IoU to the respective GT box. However, the drop in AP is more pronounced in case of coarser feature map resolutions, which underlines the necessity of high feature map resolutions to detect small objects. In particular for high GSDs, most objects are covered by a single or only a few feature map pixels in case of exploiting *conv5_3* as feature map. Thus, inference on the object location from the feature map to the input images is hindered.

Table 5.5: AP (in %) for differing feature map resolutions with respect to the GSD. Using shallower layers as feature map yielding higher feature map resolutions results overall in improved AP.

Feature Map	GSD (in cm)		
	13	19.5	26
conv5_3	65.1	38.4	21.9
conv4_3	90.3	76.5	59.7
conv3_3	**94.3**	**89.4**	**83.2**

Table 5.6: Anchor box areas employed for the different GSDs.

GSD (in cm)	Anchor Box Area (in pixels)
13	14^2, 28^2, 42^2
19.5	10^2, 18^2, 26^2
26	8^2, 14^2, 20^2

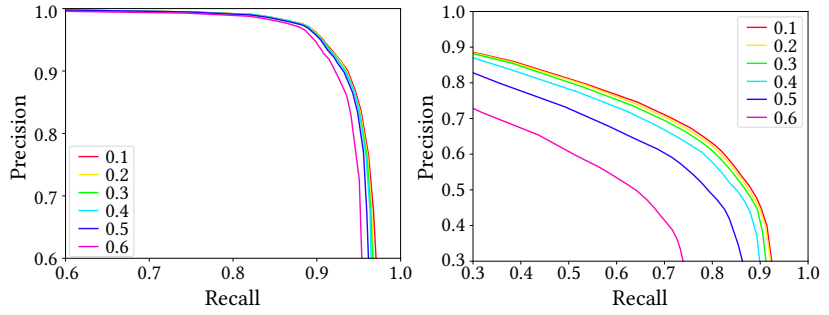
The poor localization accuracy in case of coarse feature map resolutions is confirmed by PRCs (see Figure 5.17). To this end, the precision is plotted with respect to the recall for varying IoU thresholds accordingly to Figure 5.7. Note that diverging axis scales are used for *conv3_3* and *conv5_3*. For both GSDs, coarser feature map resolutions exhibit stronger deviations in detection performance with varying IoU thresholds. While the detection performance for *conv3_3* is only slightly increasing with lower IoU threshold values, the detection performance for *conv5_3* is considerably improved, which is in accordance with the PRCs reported for a GSD of 13 cm (see Figure 5.7). Hence, exploiting higher feature map resolutions allows for detections with generally higher overlap to the GT annotations.

Qualitative detection results for a GSD of 26 cm depicted in Figure 5.18 visualize the considerably improved localization accuracy in case of higher feature map resolutions. Using *conv5_3* as feature map leads to inaccurately located bounding box predictions and numerous duplicate detections. In contrast, almost all objects are accurately detected in case of *conv3_3*, even in scenarios with dense occurrence of vehicles like parking lots. Furthermore, the number of missed detections is clearly reduced with higher feature map resolutions. Remaining false negative detections in case of *conv3_3* are mainly due to heavy occlusions, e.g., caused by trees.

Anchor Box Settings

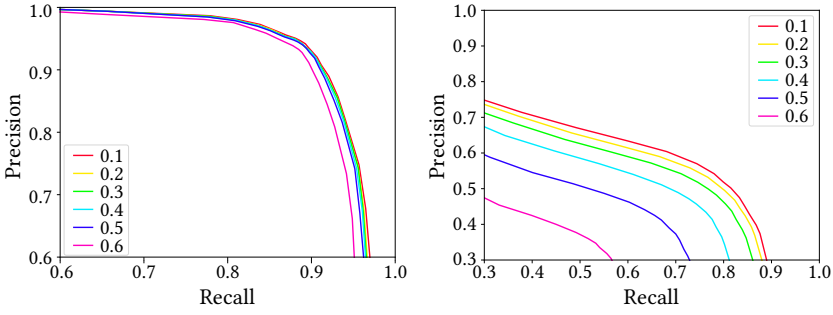
The impact of adapting the anchor box settings is depicted in Table 5.7. For this purpose, only one anchor box scale is employed analogous to Section 5.2.2. The anchor box scale is systematically varied so that the resulting

anchor box areas are in the range between 256^2 and 14^2 pixels. Note that the output of *conv3_3* is used as feature map for all experiments. For each GSD, using anchor box areas that are in the range of the particular mean vehicle dimensions exhibits the best AP. While the drop in AP is comparatively small for a GSD of 13 cm in case of anchor boxes that exceed the mean vehicle dimensions, the drop in AP becomes more distinctive with higher GSDs. Hence, adapting the anchor box settings affects the detection performance steadily with smaller object dimensions.



(a) *conv3_3* and a GSD of 19.5 cm

(b) *conv5_3* and a GSD of 19.5 cm



(c) *conv3_3* and a GSD of 26 cm

(d) *conv5_3* and a GSD of 26 cm

Figure 5.17: PRCs for various IoU thresholds used to accept GT objects as recalled. For each GSD, higher feature map resolutions exhibit better localization quality. Note that the axis scales differ for *conv3_3* and *conv5_3*.

Figure 5.18: Qualitative detection results (red boxes) for Faster R-CNN using *conv5_3* (left) and *conv3_3* (right) as feature map and the corresponding GT (green boxes) for a GSD of 26 cm. The higher feature map resolution results in better localized detections and thus, considerably fewer false positive detections. Furthermore, the number of missed detections is clearly reduced. Remaining false negative detections in case of *conv3_3* are mainly due to heavy occlusions, e.g., caused by trees (bottom row).

Table 5.7: AP (in %) for different anchor box scales and consequently different anchor box areas with respect to the GSD.

Anchor Box Area (in pixels)	GSD (in cm) 13	19.5	26
256^2	88.1	73.2	52.9
112^2	92.8	85.4	71.4
56^2	93.8	88.3	81.1
42^2	94.1	89.0	82.9
28^2	**94.3**	**89.5**	83.1
14^2	94.2	89.4	**83.2**

As pointed out before, the detection accuracy is affected by the localization quality of generated region proposals. The relation between the localization quality of generated region proposals and detection accuracy is shown in Figure 5.19. For each GSD, the highest ABO and consequently best localization quality is achieved for anchor boxes in the range of the particular mean vehicle dimensions, whereby for higher GSDs the ABO decreases steeper with anchor boxes exceeding the mean vehicle dimensions. Hence, using appropriate anchor box scales matters more in case of small object dimensions. For all GSDs, better detection results are achieved for anchor box scales, yielding region proposals with higher ABO. The detection accuracy decreases notably for ABOs close to 0.5, in particular for a GSD of 26 cm. This confirms that larger distances to the IoU threshold used to generate positive and negative training samples lead to better detections, since the corresponding region proposals may be classified with higher confidence as vehicle or background.

5.2.4 Arising Challenges

Despite the considerably improved detection accuracy, in particular for high GSDs, the performed adaptations lead to several shortcomings that have to be addressed for real-world applications. These shortcomings are primarily the weaker semantic and spatial context information of the employed features and the inference time, which are discussed in the following.

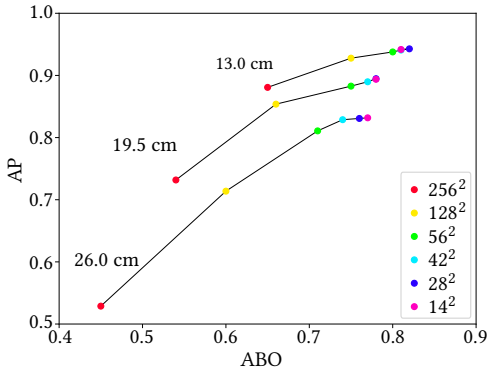


Figure 5.19: Relation between average precision and average best overlap for different anchor box scales and various GSDs. For all GSDs, candidate regions with higher ABO result in better AP.

Semantic and Spatial Context

In general, each convolutional layer within a CNN learns filters of increasing complexity, since features from the previous layers are aggregated and recombined. The first layers learn basic feature representations, e.g., edges and corners, whereas the middle layers learn to respond to object parts and the last layers learn higher representations to recognize full objects with different shapes and positions or even entire scenes [Zho15]. Furthermore, the region of the input space that affects a particular unit of the network, also referred to as *receptive field*, and consequently the spatial context information generally increases with deeper layers [Luo16]. Hence, removing deep layers to increase the feature map resolution required for accurately locating tiny objects leads to less semantic and spatial context information.

Analyzing the erroneous detections indicates that the less semantic and contextual information cause a high number of false positive detections. Figure 5.20 shows the division of the false positive detections into localization and classification errors using the output of *conv3_3* as feature map. For all GSDs, the false positive detections are mainly due to classification errors as background objects are classified as vehicles.

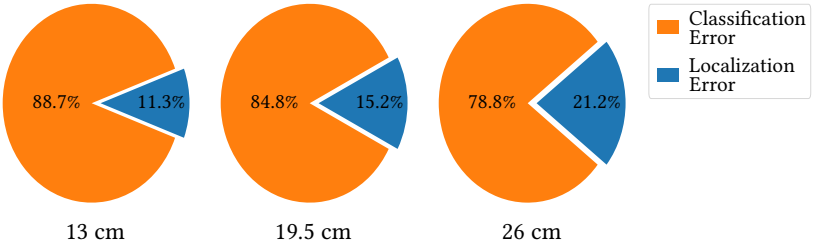


Figure 5.20: Error analysis of false positive detections for various GSDs.

To highlight the effect of the comparatively poor semantic and contextual information on the detection accuracy, false positive detections for a GSD of 13 cm are qualitatively visualized in Figure 3.4. These false positive detections are mainly due to objects with vehicle-like shapes, e.g., rectangular structures on buildings such as solar cells or windows. Note that several of these false positive detections comprise small components that activate filters responding to particular vehicle parts such as windshields or front lights. Figure 5.21 exemplarily depicts activations of four filters that respond to vehicle parts but also to similar structures.

Figure 5.21: Activations of four filters from *conv3_3* that respond to particular vehicle parts but also to similar structures.

However, looking at the surrounding areas of the false positive detections exhibits that most false positive detections are located on regions that are unlikely to contain a vehicle such as buildings or vegetation. Though the training data comprises no such positive samples for classification, the learned representation is not sufficient to distinguish between relevant and non-relevant

surrounding areas. Hence, increasing the spatial or semantic context information of the employed features to better learn the representations of relevant surrounding areas like road surfaces or parking lots can reduce the number of false positive detections. Approaches to enrich the spatial and semantic content of employed feature maps aiming at improving the detection accuracy are presented in Chapter 6.

Inference Time

Besides the detection accuracy, the inference time is another essential factor for most applications, e.g., search and rescue tasks. In Table 5.8, the impact of the performed adaptations, in particular the increased feature map resolution, which exhibits the largest gain in detection accuracy, is regarded with respect to the inference time. Following the protocol for time measurements introduced in Section 4.2, the inference time is reported for the Faster R-CNN detector as well as for each detector component in milliseconds (ms) averaged over the complete test set including 240 image tiles of size 936 × 936 pixels. For both the server and the desktop setup, the overall inference time gets worse with increasing feature map resolutions. While the inference time for the classification stage remains unchanged, as for all models the same number of region proposals are processed, the inference time considerably increases for the RPN because of the higher feature map resolution. As described in Section 5.1.1, the RPN comprised of a small network is applied on each feature map location, whereby anchor boxes centered at the respective feature map location are utilized for bounding box regression. Using *conv3_3* instead of *conv4_3* or *conv5_3* results in 4 and 16 times more feature map locations that have to be processed and consequently, the number of region proposals to compute the bounding box coordinates for and that have to be sorted increases by factor 4 and 16, respectively. Hence, the computational effort clearly increases. However, the increase in inference time is not linear, as NMS is applied on the 10,000 region proposals exhibiting the highest confidence score for each model.

Table 5.8: Comparison of the inference time of Faster R-CNN using different feature maps. The overall inference time gets worse with higher feature map resolutions due to the RPN.

Feature Map	Component	Time (in ms)	
		Server	Desktop
conv3_3	Feature Extractor	58.9	177.4
	RPN	139.8	212.8
	Classification Stage	87.5	273.3
	Total	286.2	663.5
conv4_3	Feature Extractor	59.1	230.2
	RPN	40.1	58.7
	Classification Stage	86.9	275.2
	Total	186.1	564.1
conv5_3	Feature Extractor	62.0	255.6
	RPN	20.3	23.6
	Classification Stage	86.8	273.7
	Total	169.1	552.9

Comparing the different device setups indicates that clearly less inference time is spent on more powerful devices, i.e., server setup. Especially the inference times for the base network used for feature extraction and the classification stage are considerably worse. Though multiple convolutional layers are discarded in case of *conv3_3*, the inference time for *conv5_3* only slightly increases for the server setup, whereas for the desktop setup, the inference time for feature extraction notably rises. Note that in case of *conv3_3*, the inference time for feature extraction includes the time spent for the auxiliary convolutional layer to account for the required number of inputs channels of the fully connected layers. In order to speed up the detector for different devices, optimization of the feature extraction as well as of the RPN and classification stage are required. Strategies to address these issues are introduced in Chapter 7.

6 Integration of Contextual Knowledge

As discussed in Section 5.2.4, exploiting shallower layers as feature map for Faster R-CNN to account for the characteristics of aerial imagery results in a high number of FPs. These FPs are mainly caused by rectangular structures such as windows or solar panels, whose appearance is similar to vehicles in overhead imagery. Thus, post-processing or even human interactions are requisite to ensure an accurate detection as required for a broad range of applications.

To circumvent the demand for post-processing or human interactions, two different strategies to improve the detection accuracy are proposed in the context of this thesis. In the remainder of this chapter, both strategies aiming at enhancing the contextual information of the detection framework are presented and discussed in detail. The goal of the first strategy introduced in Section 6.1 is to increase the spatial context information by combining features of shallow and deep layers to account for fine and coarse structures. The latter strategy presented in Section 6.2 employs semantic labeling to introduce more semantic context information. For this, two different approaches to integrate semantic labeling into the detection framework are realized. The work presented in this chapter is mainly based on three of the author's publications [Som17a, Som18c, Nie18].

6.1 Spatial Context

Exploiting shallower layers as feature map for Faster R-CNN is crucial for vehicle detection in aerial imagery, as the resolution of deeper layers is often not sufficient to accurately localize tiny objects. To overcome the lack of semantic and contextual content resulting in false alarms, an extension to the original Faster R-CNN architecture called Multi Feature Deconvolutional (MFD) Faster R-CNN is proposed in the context of this thesis. The schematic structure of the MFD Faster R-CNN is illustrated in Figure 6.1. Inspired by semantic labeling networks aiming at the prediction of fine and coarse structures, features from various layers (indicated by different blue tones) offering different semantic and contextual information are combined. For this purpose, a context enhancement module (CEM) highlighted in green is introduced to allow the combination of these features, while the feature map resolution is kept sufficiently high to localize tiny objects. In the following, the CEM and the implementation details are presented followed by experiments and discussion of the results to highlight the enhanced detection accuracy.

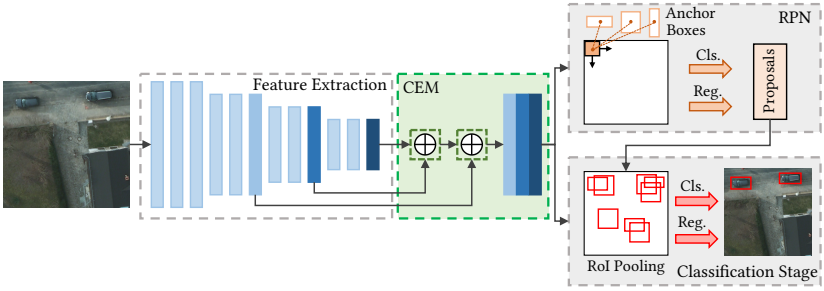


Figure 6.1: Schematic structure of the proposed Multi Feature Deconvolutional Faster R-CNN, which extends the original Faster R-CNN by a CEM to combine features of shallow and deep layers, while the resolution of the feature map is kept sufficient to localize tiny objects.

6.1.1 Context Enhancement Module

The main purpose of the context enhancement module is to create a high-resolution feature map appropriate for the localization of tiny objects, which comprises rich sematic features from deep layers. As shown in Section 5.2.1, using the output of *conv3_3* as feature map yields better detection performance compared to *conv4_3* and *conv5_3* though the corresponding features possess less semantic and contextual information. Essential for this is the coarse feature map resolution in case of *conv4_3* and *conv5_3*, i.e., 1/8 and 1/16 of the input image dimensions, which impedes the accurate localization of tiny objects. As deeper layers generally comprise features with more semantic and contextual content in comparison to shallower layers [Zho15, Luo16], the proposed CEM up-samples the low-dimensional feature maps of deep layers, i.e., *conv4_3* and *conv5_3*, which are then combined with the features of *conv3_3*. For this purpose, a deconvolutional sub-module depicted in Figure 6.2 is introduced. To combine feature maps of different size, a deconvolutional layer with kernel size 4×4 is used to up-sample the lower-dimensional feature map by a factor of 2. Instead of using nearest neighbor up-sampling as proposed in [Lin17a], the application of deconvolutional layers facilitates the learning of a non-linear up-sampling, which empirically showed superior results in preliminary experiments. The up-sampled features are then combined with the features of the shallower layer via concatenation, which is equivalent to the lateral connections proposed in [Lin17a] to ensure more precise locations of up-sampled features. To effectively fuse the information from the concatenated feature maps, a 1×1 convolutional layer is applied after the concatenation. Note that the number of channels for each feature map is further adjusted to 256, which is the minimum number of channels of the employed feature maps, to allow similar level of influence of the combined feature maps. For this, 1×1 convolutions are applied on the output of *conv4_3* and *conv5_3*.

As shown in Figure 6.1, the deconvolutional sub-module is iteratively applied. At first, the deconvolutional sub-module up-samples the features from *conv5_3* and combines the up-sampled features with the features from *conv4_3*. Then, the combined features are up-sampled and merged with the output from *conv3_3*. The resulting features are then employed as feature

map for the RPN and classification stage. Thus, the contextual content of the deeper layers are propagated to the feature map.

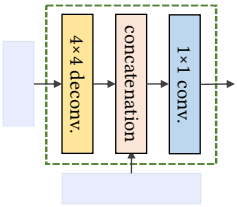


Figure 6.2: Schematic illustration of the deconvolutional sub-module used to combine features from shallow and deep layers. A deconvolutional layer with kernel size 4 × 4 and stride 2 is applied to up-sample the features from the deep layer by a factor of two, which are then combined via concatenation with the features from the shallow layer. Finally, a 1 × 1 convolutional layer is appended to effectively fuse the information from the concatenated feature maps.

6.1.2 Stage-wise Training Scheme

To train the proposed MFD Faster R-CNN, staged fine-tuning is performed inspired by [Lon15] in order to improve the weight initialization. For this purpose, Faster R-CNN using only *conv3_3* as feature map is initially trained end-to-end for a total of 60,000 iterations with an initial learning rate of 0.001. Weights pre-trained on ImageNet are used for initialization. Next, *conv4_1* through *conv4_3* and a deconvolutional sub-module are added to the network and the combination of *conv3_3* and the up-sampled feature of *conv4_3* are used as feature map. The model is then trained end-to-end for 20,000 iterations with a learning rate of 0.001. For this, the added layers are initialized by using the Gaussian weight filler method, while all other weights are initialized by the initially trained Faster R-CNN. In the last stage, *conv5_1* through *conv5_3* and a deconvolutional sub-module to up-sample the features of *conv5_3* are added to the network. The combination of *conv3_3* and the up-sampled features of *conv4_3* and *conv5_3* are used as feature map. The model is then trained end-to-end for further 20,000 iterations with a learning rate

of 0.001. While the added layers are initialized by using the Gaussian weight filler method, all other weights are initialized by the previously fine-tuned Faster R-CNN.

6.1.3 Ablation Experiments

The proposed MFD Faster R-CNN aims at enhancing the semantic and contextual information of the employed feature map by combining features from shallow and deep layers and thus, reducing the number of false alarms. The impact of enhancing the employed feature map by adding features from deeper layers on the detection accuracy is examined on the DLR 3K dataset. Table 6.1 shows the detection accuracy of the MFD Faster R-CNN for three different GSDs. For this, the original images are rescaled for training and testing as described in Section 5.2.3 and the used anchor box scales are adopted accordingly.

Table 6.1: Comparison between the proposed MFD Faster R-CNN, which combines the outputs from *conv3_3*, *conv4_3*, and *conv5_3* as feature map, and the baseline Faster R-CNN using only *conv3_3* as feature map by means of AP (in %) for various GSDs.

Feature Map	GSD (in cm)		
	13	19.5	26
conv3_3	94.3	89.4	83.2
conv3_3 $\oplus$ *conv4_3*	95.0	90.8	84.6
conv3_3 $\oplus$ *conv4_3* $\oplus$ *conv5_3*	**95.1**	**91.4**	**85.3**

Combining the features of *conv3_3* with up-sampled features from *conv4_3* exhibits superior results for all GSDs compared to the baseline Faster R-CNN merely employing *conv3_3* as feature map. The proposed MFD Faster R-CNN that combines the features of *conv3_3* with the up-sampled features from *conv4_3* and *conv5_3* achieves overall the best detection performance for all

GSDs, which indicates that adding features comprising more contextual information results in better detection performance. A reason for the gain in detection performance is an improved classification accuracy, yielding fewer false positive detections. For instance, for a GSD of 26 cm, the number of false positive detections is reduced by a factor of 27.8% compared to the baseline Faster R-CNN, while the number of false negative detections remains almost unaffected. For this, a classification threshold value of 0.5 is used. Qualitative results on DLR 3K for a GSD of 26 cm depicted in Figure 6.3 exhibit that the number of false positive detections caused by objects with shapes similar to vehicles is clearly reduced. This shows that integrating features of deeper layers with more semantic information leads to better classification accuracy and thus, results in better detection performance.

Figure 6.3: Qualitative detection (red boxes) and corresponding GT (green boxes) for Faster R-CNN using *conv3_3* as feature map (top row) and MFD Faster R-CNN (bottom row) for a GSD of 26 cm. The number of FPs due to objects with shapes similar to vehicles are reduced by integrating spatial context information from deeper layers.

The impact of the training scheme is shown in Table 6.2 exemplarily for the GSD of 26 cm. For comparison with the proposed training procedure comprised of three stages, two alternative variants are considered. The first variant is training the MFD Faster R-CNN in a single stage. For this, weights pre-trained on ImageNet are used to initialize all layers of the VGG16 backbone from *conv1_1* through *conv5_3*. The additional layers are initialized by

using the Gaussian weight filler method. The MFD Faster R-CNN is trained end-to-end for a total of 60,000 iterations with an initial learning rate of 0.001. The latter variant is comprised of two stages. In the first stage, Faster R-CNN using only *conv3_3* as feature map is initially trained end-to-end for 60,000 iterations with an initial learning rate of 0.001 analogously to the stage-wise training procedure. Next, the layers to integrate features from deeper layers are appended to the network, which is then trained for further 20,000 iterations. For this, the added layers are initialized by using the Gaussian weight filler method, while all other weights are initialized by the previously fine-tuned Faster R-CNN. Both alternative training schemes exhibit improved AP compared to the baseline Faster R-CNN, which confirms the improved detection performance in case of employing more semantic and contextual information. Nevertheless, using the three-stage training procedure results in clearly better detection performance. As the network learns an accurate localization in the first stage, the enhancement of the employed feature map with features from deeper layers via learned up-sampling and combination is facilitated in the subsequent stages compared to the single stage training.

Table 6.2: Impact of the training procedure on the detection performance. Successively adding features from deeper layers results in better AP.

Training scheme	AP (in %)
One-stage	84.0
Two-stage	84.8
Three-stage	**85.3**

6.2 Semantic Context

An alternative strategy to overcome the lack of contextual information in deep learning based detectors adapted for vehicle detection in aerial imagery is the exploitation of semantic labeling networks. Semantic labeling is essentially a

pixel-wise classification of an input image. Integrating semantic labeling into the detection framework allows the introduction of semantic context information, i.e., local surrounding of an object to detect, and thus, seems promising to reduce the number of false positive detections, which are often caused by rectangular structures in image regions that are unlike to contain vehicles such as buildings. For this purpose, two differing methods to integrate the semantic labeling network into the detection pipeline are proposed.

The remainder of this section is organized as follows. First, auspicious architectures for semantic labeling of aerial imagery are presented. Then, the two proposed methods are described in detail and ablation experiments on the ISPRS 2D Semantic Labeling Challenge Potsdam dataset are provided. Finally, an evaluation on a novel semantic labeling dataset based on DLR 3K is performed to emphasize the effect of integrating semantic labeling on the detection performance.

6.2.1 Semantic Labeling Approaches

Several CNN architectures have been proposed for the task of semantic labeling. In the following, four promising architectures that are examined within the context of this thesis are introduced. Note that all considered architectures are based on the VGG16 architecture to facilitate the fusion of the semantic labeling network with the Faster R-CNN detection network proposed in Section 6.2.3.

FCN-32s

FCN-32s [Lon15] depicted in Figure 6.4 is a fully convolutional network (FCN) designed for the task of semantic labeling. For this, the fully connected layers of the default architecture, i.e., VGG16 (see Table 5.1), are converted into convolutional layers. As fully connected layers can be viewed as convolutions covering the entire input dimensions [Lon15], *fc6*, whose input dimensions are $7 \times 7 \times 512$, is cast into a convolutional layer with kernel size 7×7 and the subsequent fully connected layer *fc7* is transformed into a convolutional layer

with kernel size 1 × 1. The last fully connected layer originally used as classification layer is discarded and a 1 × 1 convolutional layer with 6 channels is appended in case of the Potsdam dataset to predict scores for each category at each of the coarse feature map locations. Finally, pixel-wise predictions for the input image are achieved by up-sampling the coarse predictions via a single deconvolutional layer. Per-pixel softmax loss is used to train the semantic labeling network.

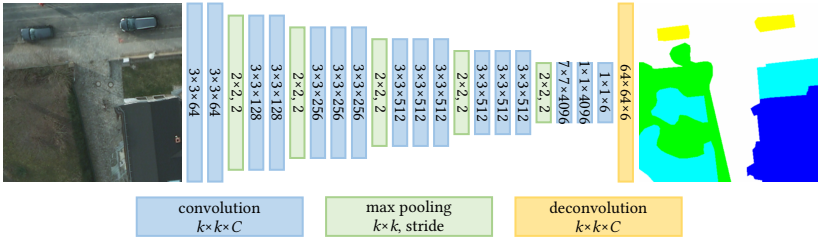


Figure 6.4: Schematic structure of FCN-32s. The first two fully connected layers of VGG16 are cast into convolutional layers with kernel size 7 × 7 and kernel size 1 × 1, respectively, while the last fully connected layer is replaced by a 1 × 1 convolutional layer with 6 channels. To up-sample the output of the last convolutional layer by a factor of 32, a deconvolutional layer with kernel size 64 × 64 and stride 32 is appended.

FCN-16s

As FCN-32s outputs semantic labeling results with fuzzy boundaries due to the coarse prediction layer, FCN-16s [Lon15] extends FCN-32s by combining features from a deep, coarse layer with features from a shallow, fine layer to overcome this drawback (see Figure 6.5). Instead of up-sampling the output of the prediction layer on top of *fc7* by a factor of 32, the prediction layer is up-sampled by a factor of 2 and fused with the output of *pool4* via element-wise addition. Note that a 1 × 1 convolutional layer with 6 channels is applied on

pool4 to adjust the number of channels before fusing. Then, the combined features are up-sampled to the input image dimensions by applying an additional deconvolutional layer.

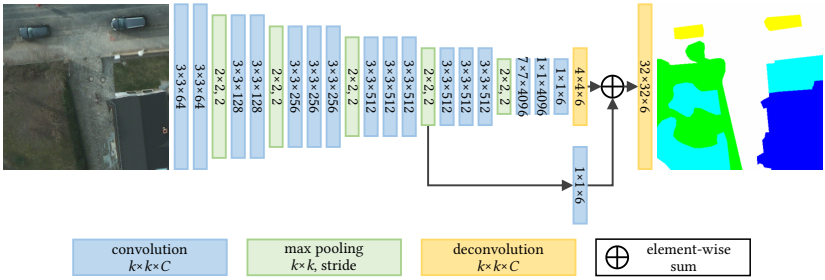


Figure 6.5: Schematic structure of FCN-16s. As opposed to FCN-32, the output of the last convolutional layer is up-sampled by a factor of two via deconvolution with kernel size 4 × 4 and stride 2 and then combined via element-wise addition with the output of the 4$^{\text{th}}$ pooling layer. Note that the number of channels is adjusted by an additional 1 × 1 convolutional layer. A deconvolutional layer with kernel size 32 × 32 and stride 16 is used to up-sample the combined features by a factor of 16.

FCN-D16

FCN-D16 is a modification of FCN-32s proposed within the context of this thesis in order to account for small objects and fine structures present in aerial imagery. FCN-D16 illustrated in Figure 6.6 is based on VGG16 with all fully connected layers converted to convolutional layers such as FCN-32s. In contrast to FCN-32s, dilated convolutions are introduced to increase the receptive field without reducing the spatial resolution, which has shown to lead to clearly improved semantic labeling results on benchmark datasets [Yu15]. For this purpose, the stride of the last pooling layer, i.e., *pool5*, is set to 1, so that the down-sampling factor is reduced from 32 to 16. Furthermore, the convolutional layer *fc6* is replaced by dilated convolutions with dilation factor 2. While the reduced down-sampling factor retains more fine details, i.e., local information, the use of dilated convolutions increases the receptive field and

thus, yielding more contextual information. Analogous to FCN-32s, a 1×1 convolutional layer with 6 channels is appended to predict scores for each category at each of the coarse feature map locations and a single deconvolutional layer is then applied to achieve pixel-wise predictions for the input image.

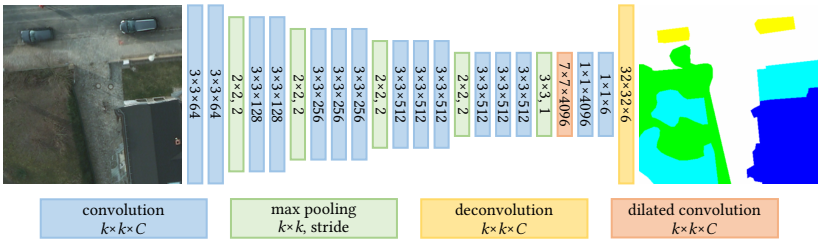


Figure 6.6: Schematic structure of FCN-D16. In contrast to FCN-32s, the stride of the last pooling layer is set to 1 to reduce the down-sampling factor from 32 to 16. The subsequent convolutional layer, i.e., fc6, is replaced by dilated convolutions with dilation factor 2 to maintain the size of the receptive field of its counterpart in FCN-32s. A deconvolutional layer with kernel size 32×32 and stride 16 is applied to up-sample the output of the last convolutional layer by a factor of 16.

SegNet

Badrinarayanan *et al.* [Bad17] proposed an encoder-decoder architecture for semantic labeling called SegNet, which showed promising semantic labeling results on benchmark datasets. As depicted in Figure 6.7, SegNet is composed of an encoder and a decoder network. The architecture of the encoder is topologically identical to the convolutional part of the VGG16 architecture, while the topology of the decoder is in essence the mirrored encoder. Note that the number of channels for the last convolutional layer is set to 6 in order to provide pixel-wise predictions for each category. Instead of applying deconvolutional layers to up-sample low resolution feature maps, the decoder uses max pooling indices to perform non-linear up-sampling. For this, the pooling indices, which indicate the positions with maximum values within the region

defined by the pooling size, are stored for each max pooling step of the corresponding encoder. Up-sampling is then performed by mapping the values of the particular feature map to the stored positions as shown in Figure 6.8, while the remaining pixels are filled with zeros. In this manner, the need for learned up-sampling is eliminated. Per-pixel softmax loss is used to train the semantic labeling network according to FCN-32s and its extensions.

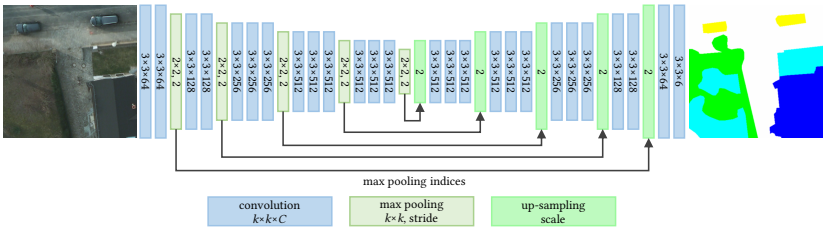


Figure 6.7: Schematic structure of SegNet, which is comprised of an encoder and decoder network. The topology of the encoder is identical to the convolutional part of VGG16, while the topology of the decoder is in essence the mirrored encoder. Non-linear up-sampling is performed by using max pooling indices that encode the positions where to map the values of the preceding feature map as shown in Figure 6.8.

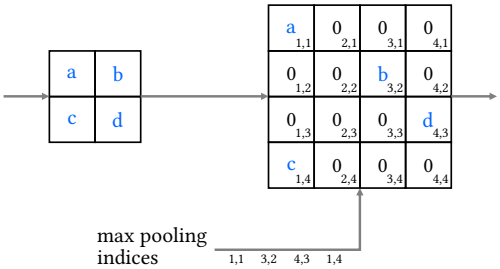


Figure 6.8: Illustration of the non-linear up-sampling conducted in SegNet. The positions with maximum values within the region defined by the pooling size are stored for each max pooling step in pooling indices, which are used to map the values of the preceding feature map to the stored positions. The remaining pixels are filled with zeros.

Ablation Experiments

In the following, the differing semantic labeling architectures are evaluated on the Potsdam dataset introduced in Section 4.1 to assess the potential of each architecture for integration into the Faster R-CNN detection framework. For training, the original image patches are cropped into tiles of size 256 × 256 pixels with an overlap of 50%. In addition, data augmentation is performed through applying vertical and horizontal flipping as well as rotation in steps of 90 degrees, so that the number of training samples is increased by a factor of 8. At test time, image tiles of size 512 × 512 pixels with 50% overlap between tiles are used. For each tile, the central 384 × 384 pixels are selected for the final semantic labeling mask, which exhibited slightly better performance in preliminary experiments compared to averaging the overlapping regions. The performed stitching strategy further reduces artefacts at image borders and stitching borders, respectively. Each model is trained for 100,000 iterations with a batch size of 6 using the Adam solver with an initial learning rate of $1e\text{-}8$. Stage-wise training as proposed in [Lon15] is conducted to train FCN-16s. Thus, weights of FCN-32s pre-trained on the Potsdam dataset are used for initialization. Otherwise, weights pre-trained on ImageNet are used for initialization.

The semantic labeling results for the different architectures are given in Table 6.3. For this, F1-score computed for each class and overall accuracy are used as evaluation metrics following the 2D Semantic Labeling Contest protocol (see Section 4.2). All architectures achieve an overall accuracy around 88%, which is clearly higher compared to the baseline results solely on RGB imagery reported in [She16]. In particular, the F1-score achieved for category *car* is considerably improved. SegNet, FCN-16s and FCN-D16 even outperform the baseline results, which additionally rely on IR and DSM information. High F1-scores are achieved for each category except for the category *clutter*. A reason for the by far lowest accuracy is the large variance of objects and concepts aggregated in this category, ranging from water areas through tennis courts to small structures such as outdoor furniture. Compared to the F1-scores achieved for the categories *impervious surface* and *building*, the categories *low vegetation* and *tree* exhibit slightly lower F1-scores. These lower

F1-scores are mainly due to confusion between both categories, as even the borders between both categories are often not unambiguous. While SegNet achieves overall the highest accuracy slightly outperforming FCN-16s and FCN-D16, FCN-32s exhibits the lowest overall accuracy and F1-scores for each category especially for category *car*, whose instances possess the smallest dimensions. This indicates the importance of adding features of shallower layers or maintaining finer feature maps by applying dilated convolutions to accurately label small instances.

Table 6.3: Semantic labeling results of the examined semantic labeling architectures compared to the baseline results reported in [She16]. F1-scores are provided for the categories *impervious surface* (IS), *building* (B), *low vegetation* (LV), *tree* (T), *car* (Ca) and *clutter* (Cl).

Sem. Labeling Approach	F1-score (in %)						Overall Accuracy
	IS	B	LV	T	Ca	Cl	
RGB only [She16]	88.96	92.49	83.84	82.11	86.13	73.09	86.05
RGB+IR+DSM [She16]	**90.01**	**93.83**	**86.15**	83.59	92.97	**75.87**	87.84
FCN-32s	89.42	92.14	83.80	83.61	90.76	54.95	87.68
FCN-16s	89.92	92.44	84.21	84.32	94.58	56.48	88.09
FCN-D16	89.69	92.93	84.54	84.54	93.34	57.35	88.19
SegNet	89.93	93.70	85.05	**84.57**	**95.26**	55.54	**88.46**

Qualitative examples depicted in Figure 6.9 emphasize the good semantic labeling results of the examined architectures. In particular, the high accuracy for the categories *car*, *impervious surface* and *building* is notable, which is essential for the application within a detection framework aiming at suppressing false alarms often caused by vehicle-like structures on buildings. On the other hand, the comparably poor accuracy in case of category *clutter* is apparent throughout the examined architectures. The qualitative results for FCN-32s indicate the slightly worse accuracy in case of instances possessing small dimensions. For instance, the boundaries for segments labeled as category *car* are less accurate resulting in merged segments.

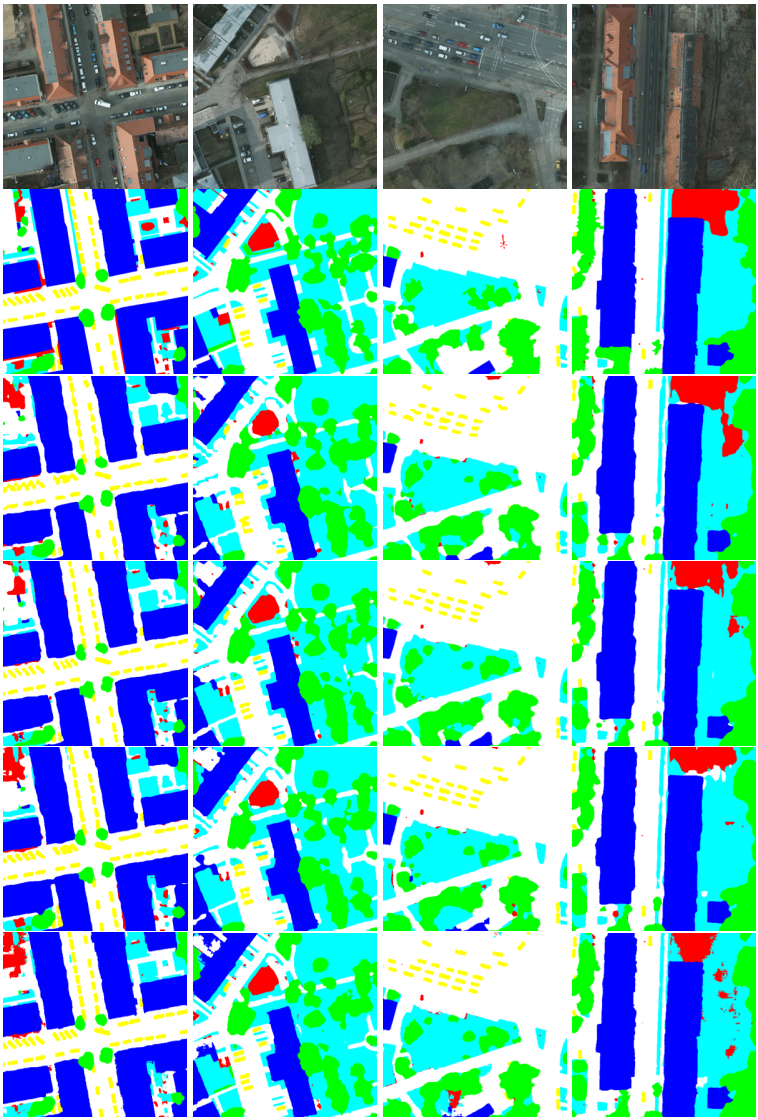


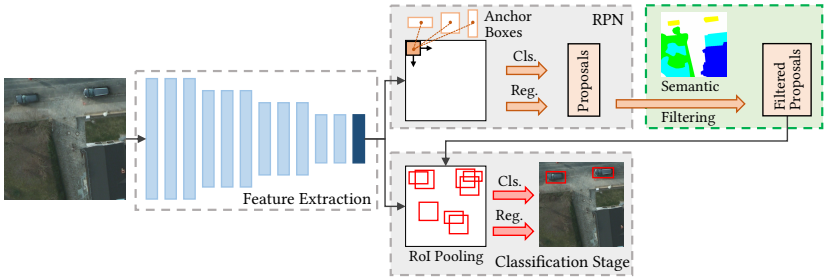
Figure 6.9: Qualitative semantic labeling results of FCN-32s (3rd row), FCN-16s (4th row), FCN-D16 (5th row) and SegNet (6th row) and corresponding GT (2nd row).

6.2.2 Semantic Labeling based Filtering

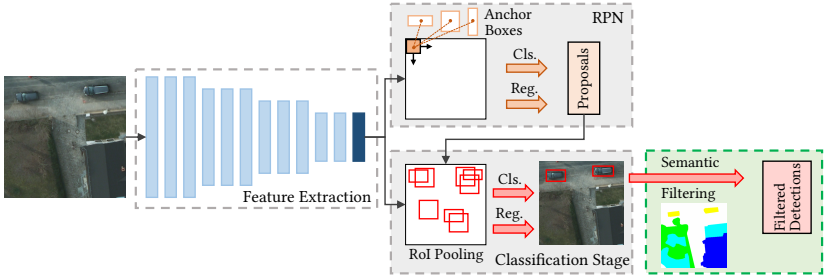
In the following section, the first strategy to improve the detection performance by integrating semantic labeling into the detection framework is introduced. As a large number of false positive detections is caused by vehicle-like structures located offside roads, e.g., solar cells on buildings, semantic labeling masks that exhibit accurate predictions of roads as well as driveways and parking lots (see Section 6.2.1) are used to filter out detections that are mainly located on regions unlikely to contain vehicles. Note that a separate semantic labeling network is employed to generate the semantic labeling masks. Two different positions to integrate the filtering step into the Faster R-CNN detection pipeline are investigated. The first position is directly after the RPN (see Figure 6.10a) and the latter after the classification stage (see Figure 6.10b). While filtering the final detections after the classification stage only yields a reduced number of false alarms, filtering region proposals may additionally reduce the inference time because a smaller set of candidate regions is passed to the classification stage of the detection network.

Filtering Scheme

The proposed filtering scheme is straightforward as illustrated in Figure 6.11. A separate CNN network is employed to generate a semantic labeling mask, which is then used to compute the category distribution within each region proposal or detection. The characteristics of the category distribution are lastly employed to accept or reject the corresponding region proposal or detection. Note that simply accepting region proposals or detections that are mainly labeled as category car would decrease the detection accuracy due to inaccurate labeling, e.g., high number of vehicles beneath a tree are labeled as category *tree* (see Figure 4.7), which influences the training of the semantic labeling networks. Thus, a filter criterion is applied, which is designed to maximize the detection accuracy.



(a) Filtering of region proposals



(b) Filtering of detections

Figure 6.10: Schematic illustration of the semantic labeling based filtering. For this purpose, semantic labeling masks generated by a separate network are used to either filter out region proposals (a) or detections (b) that are mainly located on regions unlikely to contain vehicles.

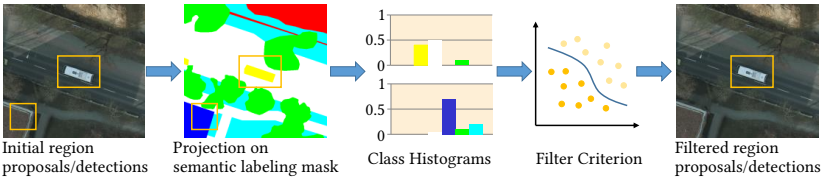


Figure 6.11: Schematic of the semantic labeling based filtering scheme. Class histograms computed for each initial region proposal or detection are used to filter out region proposals or detections based on a filter criterion.

For this purpose, the effect of rejecting region proposals or detections based on a single semantic labeling category on the detection performance is analyzed in Table 6.4. Region proposals and detections, whose pixels are labeled

at least 50% as *impervious surface*, *building*, *low vegetation*, *tree* and *clutter*, respectively, are removed. For this, the provided semantic labeling GT is exploited. The Faster R-CNN model is trained end-to-end for 60,000 iterations and an initial learning rate of 0.001. To account for the characteristics of the Potsdam dataset, in particular the lower GSD yielding larger vehicle dimensions, the output of *conv4_3* is used as feature map. Furthermore, the anchor base size is set to 4 and the anchor scales are set to 8, 16, and 24. The top-100 ranked region proposals after NMS are forwarded to the classification stage. Removing region proposals or detections that are mainly labeled as *building* or *clutter* clearly improves the detection accuracy as the number of false alarms caused by objects with shapes similar to vehicles, like solar cells on roofs, are filtered out. Instead, removing region proposals or detections mainly labeled as category *tree* yields a drop in AP as numerous vehicles are missed due to the semantic labeling GT. As most vehicles are surrounded by impervious surfaces, removing region proposals or detections with at least 50% labeled as *impervious surface* also results in worse AP, whereby the drop in AP is more distinct in case of filtering detections. On the other hand, filtering out region proposals or detections mainly labeled as category *low vegetation* exhibits only minor impact on the detection accuracy. Note that performing the filtering step directly after the RPN results in fewer region proposals per image that have to be classified and consequently is less computationally expensive. Thus, only filtering of region proposals is conducted in the following.

As using a filter criterion based on a single semantic labeling category only results in a small gain in AP, only region proposals that fulfill following equation are considered for classification:

$$max(\frac{N_{car}}{N_{bg}}, \frac{N_{is}}{N_{bg}}, \frac{N_{tree}}{N_{bg}}) > 1, \tag{6.1}$$

where N_i is the number of pixels corresponding to class i within a region proposal and N_{bg} is the sum of all pixels labeled as *building* or *low vegetation*. Thus, only region proposals that are mainly labeled as *car*, *impervious surface*, or *tree* are considered for classification.

Table 6.4: Impact of filtering out detections or region proposals (⁺) based on a single semantic labeling category on the Potsdam dataset. Note that detections or region proposals whose pixels are at least 50% labeled as the current category are removed and the semantic labeling GT is used to compute the category distribution.

Filter Criterion	AP (in %)	# Proposals/Image
-	92.9	100
50% Imp. Surface	86.4	100
50% Building	93.5	100
50% Low veg.	92.6	100
50% Tree	85.5	100
50% Clutter	**93.6**	100
50% Imp. Surface⁺	92.3	82.4
50% Building⁺	93.5	84.8
50% Low veg.⁺	93.1	**77.7**
50% Tree⁺	87.9	89.8
50% Clutter⁺	**93.6**	95.1

Ablation Experiments

The proposed filter criterion aims at maximizing the number of removed false alarms and at minimizing the number of region proposals to classify. The effect of applying the proposed filter criterion is evaluated in Table 6.5. By using the semantic labeling GT to compute the category distribution, the detection performance is improved by 1.4% in AP compared to the baseline Faster R-CNN, while the number of region proposals forwarded to the classification stage is almost halved. Note that the proposed filter criterion empirically showed the largest gain in AP compared to further filter criteria examined in preliminary experiments. Employing the masks generated by the semantic labeling networks introduced in Section 6.2.1 leads to slightly worse AP due to incorrectly predicted labels, e.g., confusion between *clutter* and *car*, while the number of region proposals considered for classification remains roughly unaffected. Nevertheless, the baseline Faster R-CNN is still outperformed, whereby using FCN-D16 to generate the semantic labeling masks exhibits the

highest detection accuracy amongst the examined semantic labeling architectures. Compared to the baseline Faster R-CNN, the number of false alarms caused by objects with shapes similar to vehicles, e.g., windows on buildings, is reduced as illustrated in Figure 6.12. For this, the output of FCN-D16 is used and only detections (red boxes) with a confidence score above 0.5 are accepted. Note that using the semantic labeling results as detections themselves is not practical on the Potsdam dataset, as multiple missed and split detections occur due to inaccurate semantic labeling annotations (see Figure 4.7) [Som17a].

Table 6.5: Impact of the semantic labeling mask employed for filtering out region proposals on the Potsdam dataset. For this, eq. (6.1) is used as filter criterion.

Semantic Labeling	AP (in %)	# Proposals/Image
GT	**94.3**	**50.3**
FCN-32s	93.7	50.9
FCN-16s	93.8	51.1
FCN-D16	93.9	51.1
SegNet	93.7	51.0

As aerial imagery are often recorded with higher GSDs, the impact of semantic labeling based filtering on the detection performance is examined for various GSDs. For this, the original images are down-scaled by factor 2, 3, 4, and 5, respectively. FCN-D16 trained on the original image resolution is used to generate semantic labeling masks instead of training a model for each resolution separately. During testing, the down-scaled images are up-scaled to the original image resolution. The corresponding semantic labeling results are reported in Table 6.6. The F1-score decreases for all categories with higher GSDs and lower ground resolutions, whereby the drop in accuracy is only minor for a GSD of 10 cm. Category *tree*, which is due to the season only represented by thin tree branches, undergoes the strongest decrease with lower ground resolutions, as such fine structures are eliminated during down-scaling and consequently, assigned to incorrect categories. The highest F1-scores are achieved for category *car*, which even exhibits a F1-score above 78% for a GSD of 25 cm.

The categories *impervious surface*, *building*, and *low vegetation* show good F1 scores around 70%, which are essential to apply the proposed filter criterion.

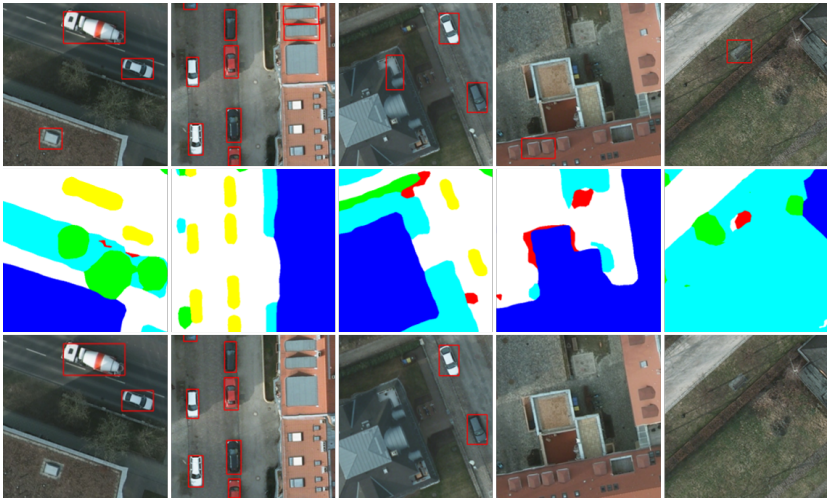


Figure 6.12: Qualitative detection examples before (top row) and after (bottom row) filtering the region proposals using the semantic labeling mask outputted by FCN-D16 indicate that false alarms located on regions that are unlikely to contain vehicles, e.g., windows on buildings, are removed.

The detection performance for the different GSDs is given in Table 6.7. For this, a Faster R-CNN model is trained on the down-scaled images for each GSD separately. The respective anchor box scales are adapted for each GSD, so that the employed anchor box areas are equivalent to the mean object dimensions. All further settings remain unchanged. The semantic labeling masks generated for the corresponding resolution and the proposed filter criterion (see eq. (6.1)) are used to filter the region proposals. The detection performance is clearly improved for all GSDs, whereby the gain in AP increases with higher GSDs even though the semantic labeling accuracy gets worse for higher GSDs. This indicates that the importance of employing semantic context information increases with higher GSDs and consequently smaller object dimensions.

Table 6.6: Semantic labeling results for different GSDs using FCN-D16 on the Potsdam dataset. F1-scores are provided for the categories *impervious surface* (IS), *building* (B), *low vegetation* (LV), *tree* (T), *car* (Ca) and *clutter* (Cl).

GSD	F1-score (in %)						Overall
(in cm)	IS	B	LV	T	Ca	Cl	Accuracy
5	89.69	92.93	84.54	84.54	93.34	57.35	88.19
10	89.35	92.30	83.67	83.08	93.04	55.09	87.45
15	84.43	89.65	75.54	44.99	91.24	37.36	77.03
20	76.08	84.93	70.86	18.25	87.79	28.40	68.60
25	70.10	75.75	69.76	11.60	78.01	19.29	63.04

Table 6.7: Average precision (in %) of Faster R-CNN with and without semantic labeling based filtering using FCN-D16 on the Potsdam dataset for different GSDs.

Filter	GSD (in cm)				
Criterion	5	10	15	20	25
-	92.9	92.2	90.1	82.7	62.9
eq. (6.1)	**93.9**	**93.3**	**91.7**	**85.8**	**70.7**

6.2.3 Joint Semantic Labeling and Detection

Though the semantic labeling based filtering strategy demonstrates the utility of semantic labeling for vehicle detection in aerial imagery, the inference time doubles due to the application of two separate networks: one for detection and one for semantic labeling. To overcome this drawback, an alternative strategy is proposed that incorporates semantic labeling into the detection framework by merging both networks. Thus, semantic labeling directly induces scene knowledge into the feature maps used within the detection framework instead of filtering out region proposals or detections, respectively. In the following, two different variations to induce scene knowledge are introduced and the effect of the detection performance is discussed.

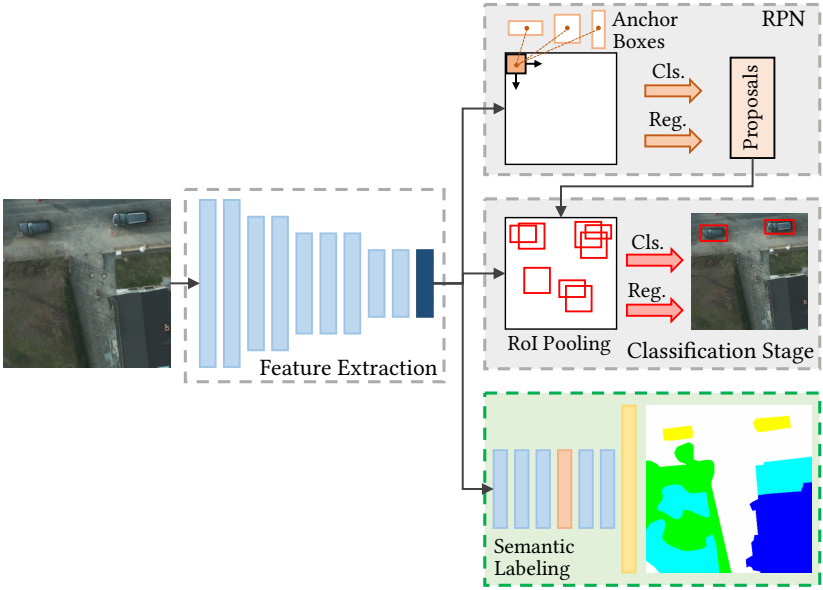


Figure 6.13: Schematic illustration of the proposed Implicit Multi-Task Faster R-CNN. An auxiliary branch for semantic labeling is added to the network, which shares the first four sequences of convolutional layers with the detection branch, i.e., the RPN and the classification stage. Thus, the semantic labeling branch has an implicit effect on the detections through the resulting shared feature map.

Implicit Multi-Task Faster R-CNN

The first variant, which is in the following referred to as Implicit Multi-Task (IMT) Faster R-CNN, is depicted in Figure 6.13. The proposed architecture comprises two branches: one for detection (top branch) and one for semantic labeling (bottom branch). The detection branch includes the RPN and the classification stage, while FCN-D16 is exemplarily employed for the semantic labeling branch. Note that the semantic labeling network can be replaced by the alternative architectures described in Section 6.2.1. Both branches are merged by sharing the first four sequences of convolutional layers. The prerequisite for this is that both branches are based on the same base architecture, i.e., VGG16. This allows the network to learn a shared global feature map, i.e.,

conv4_3, through which the semantic labeling branch implicitly affects the detection branch. As the semantic labeling branch is only required for training, it can be discarded during deployment.

To allow end-to-end training of both tasks, a joint multi-task loss L_{MT} comprised of five losses is proposed:

$$L_{MT} = \lambda_1 L_{SL} + \lambda_2 L_{RPN,cls} + \lambda_3 L_{RPN,reg} + \lambda_4 L_{CLS,cls} + \lambda_5 L_{CLS,reg}. \quad (6.2)$$

The semantic labeling loss L_{SL} is the normalized sum of the pixel-wise softmax loss, while the further losses are the classification losses $L_{RPN,cls}$ and $L_{CLS,cls}$ and regression losses $L_{RPN,reg}$ and $L_{CLS,reg}$ of the Faster R-CNN introduced in Section 5.1. The weighting factor λ_1 is set to 4 and the weighting factors λ_2, λ_3, λ_4, and λ_5 are set to 1, so that the semantic labeling branch and the detection branch are weighted equally.

Explicit Multi-Task Faster R-CNN

The latter variant termed Explicit Multi-Task (EMT) Faster R-CNN further extends the proposed IMT Faster R-CNN by explicitly employing additional features of the semantic labeling branch for detection as visualized in Figure 6.14. For this purpose, the output of *conv5_3* of the semantic labeling branch is employed as auxiliary feature map for the classification stage. The set of generated region proposals is projected onto the auxiliary feature map as well and an additional RoI pooling layer extracts the corresponding features. The output of the additional RoI pooling layer has the same dimensions, i.e., 7×7, and number of channels, i.e., 512, as the output of the RoI pooling layer of the detection branch. The outputs of both RoI pooling layers are fused via element-wise addition. The fused features are then fed into the sequence of fully connected layers of the classification stage. Outputs of deeper convolutional layers of the semantic labeling branch are not considered as feature map because the number of output channels exceeds 512 as required for the element-wise addition. The EMT Faster R-CNN is trained analogously to the IMT Faster R-CNN using the joint multi-task loss given in eq. (6.2). Thus,

semantic context information is induced twofold: implicitly by shared convolutional layers and joint learning and explicitly by exploiting features of the semantic labeling branch for the classification stage.

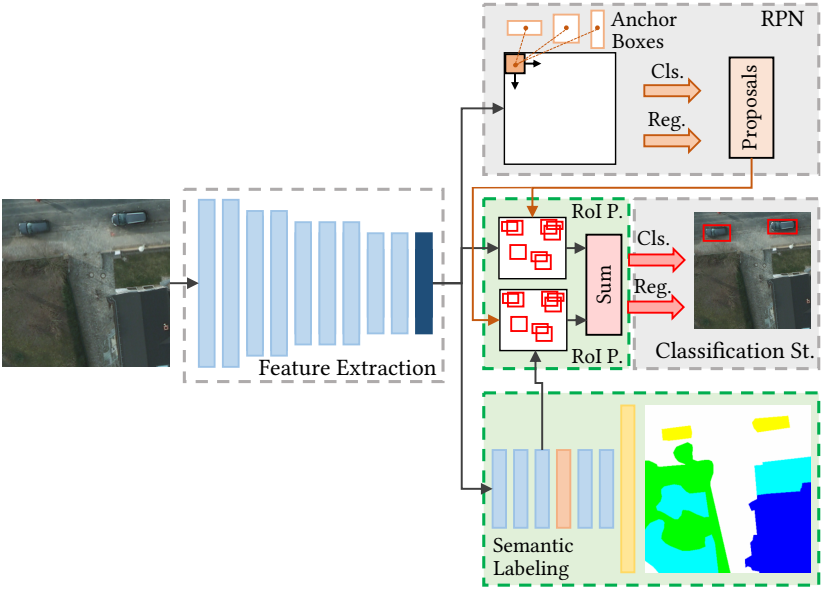


Figure 6.14: Schematic illustration of the proposed Explicit Multi-Task Faster R-CNN. In addition to the shared sequences of convolutional layers, features from the semantic labeling branch are explicitly added for each region proposal by way of an additional RoI pooling layer and element-wise addition.

Ablation Experiments

In the following, the impact of both variants to induce scene knowledge into the detection framework are evaluated. For this, each model is trained end-to-end for 70,000 iterations with an initial learning rate of 0.001. VGG16 pre-trained on ImageNet is used to initialize the weights of the shared convolutions as well as the weights of both branches. Note that data augmentation, which results in better semantic labeling results [She16], is performed through applying vertical and horizontal flipping as well as rotation in steps of 90 degrees. To account for the characteristics of the Potsdam dataset, the settings introduced in Section 6.2.2 are adopted, i.e., exploitation of *conv4_3* as feature map, setting the anchor base to 4 and the anchor scales to 8, 16, and 24, and forwarding of the top-100 ranked region proposals after NMS to the classification stage.

Table 6.8: Detection results for IMT Faster R-CNN and EMT Faster R-CNN with different semantic labeling architectures employed as semantic labeling branch on the Potsdam dataset.

Semantic Labeling Architecture	IMT Faster R-CNN	EMT Faster R-CNN
-	95.7	95.7
FCN-32s	**96.3**	96.6
FCN-16s	95.7	96.4
FCN-D16	96.1	**96.8**
SegNet	96.1	96.3

Table 6.8 shows the detection accuracy for the IMT Faster R-CNN and EMT Faster R-CNN with different semantic labeling architectures employed as semantic labeling branch. The best detection accuracy for IMT Faster R-CNN is achieved by employing FCN-32s as semantic labeling architecture, which slightly outperforms FCN-D16 and SegNet. The IMT Faster R-CNN exhibits an improved AP compared to the baseline Faster R-CNN except for using FCN-16s as semantic labeling architecture, which indicates that implicitly inducing

scene knowledge by sharing features results in improved detection accuracy. Note that the baseline Faster R-CNN is trained on augmented data as well. Thus, the achieved AP is higher compared to the AP reported in Table 6.4. One reason for the absent gain in AP in case of FCN-16s may be the training procedure as no staged training as proposed in [Lon15] is performed. In [Som17a], notably worse semantic labeling results are achieved on the Potsdam dataset by training FCN-16s at once compared to FCN-32s. EMT Faster R-CNN shows better detection results for all semantic labeling architectures compared to its IMT Faster R-CNN counterparts and clearly improved detection results compared to the baseline Faster R-CNN. Hence, explicitly adding features from the semantic labeling branch and consequently more semantic context information boosts the detection accuracy. Overall, the best AP is achieved for FCN-D16, which outperforms the baseline Faster R-CNN by 1.1% in AP.

Table 6.9: Impact of varying the weighting factor λ_1 of the semantic labeling loss exemplarily for EMT Faster R-CNN with FCN-D16.

Weighting factor λ_1	AP (in %)
1	96.5
2	96.5
3	96.7
4	**96.8**

The impact of varying the weight ratio between the semantic labeling loss and the detection loss is given in Table 6.9 exemplarily for EMT Faster R-CNN using FCN-D16 for the semantic labeling branch. For this, the weighting factor λ_1 of the semantic labeling loss (see eq. (6.2)) is varied in the range between 1 and 4, while all other weighting factors are kept fixed at 1. Weighting all losses equally ($\lambda_1 = 1$) results in the lowest AP, while the best AP is achieved by weighting the semantic labeling branch and the detection branch comprising four losses equally ($\lambda_1 = 4$). This indicates that the impact of the semantic

labeling branch on the shared features increases with higher weighting factors. Thus, the enhanced adaptation of the employed feature map with respect to the semantic labeling categories yields reduced false alarms mainly labeled as *building* or *low vegetation*. Note that experiments with even higher values for λ_1 showed no further improvements.

Qualitative detection examples shown in Figure 6.15 illustrate that EMT-Faster R-CNN exhibits fewer FPs due to vehicle-like structures compared to the baseline Faster R-CNN. Overall, the number of FPs is reduced by 58.8% for a confidence score of 0.5, while the number of FNs remains almost unchanged. The corresponding semantic labeling masks still show good results of the overall scene though the training settings are chosen with respect to the detection task, e.g., batch size or number of iterations. Good semantic labeling results are essential for improving the detection performance by enhancing the scene knowledge, otherwise the detection performance may decrease due to distraction caused by the semantic labeling branch.

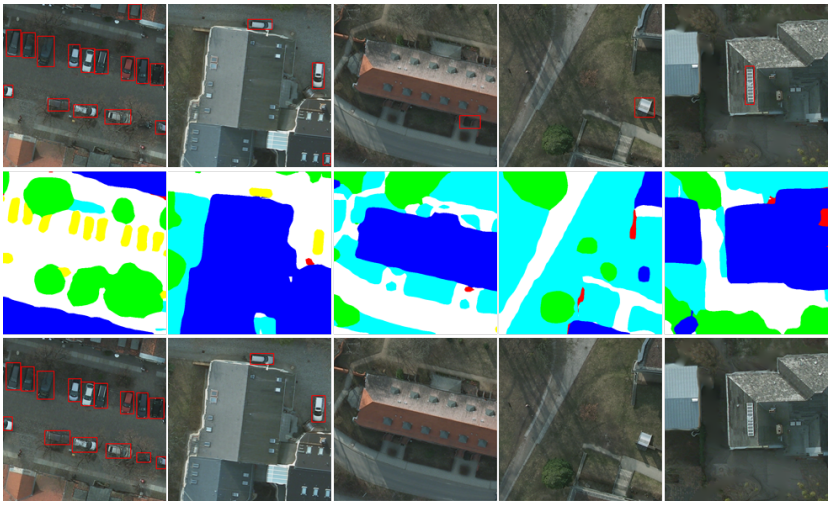


Figure 6.15: Qualitative detection examples of the baseline Faster R-CNN (top row) and EMT Faster R-CNN with FCN-D16 (bottom row) indicate that false alarms located on regions that are unlikely to contain vehicles are reduced.

Analyzing the remaining FPs underlines that almost no FPs on buildings remain. Qualitative examples of remaining FPs given in Figure 6.16 illustrate that most FPs can be attributed to vehicle-like structures positioned near roads or beneath trees such as trailers or garbage containers. Examining the semantic labeling masks exhibits that the corresponding pixels of the FPs are mainly labeled as category *car* or *tree* as well. Hence, the semantic labeling results are in good accordance with the generated detections.

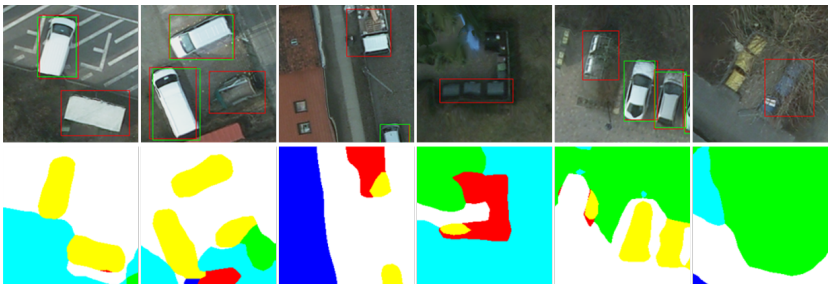


Figure 6.16: Qualitative examples of remaining FPs for EMT Faster R-CNN with FCN-D16. Most FPs can be attributed to vehicle-like structures positioned near roads or beneath trees, while almost no FPs on buildings remain.

As aerial imagery often exhibits higher GSDs, the detection performance of IMT Faster R-CNN and EMT Faster R-CNN are evaluated for various GSDs. Therefore, the image tiles used for training and testing are down-scaled by factor 2, 3, 4, and 5 yielding GSDs of 10 cm, 15 cm, 20 cm, and 25 cm, respectively. For each GSD, models are trained separately, whereby FCN-D16 is employed for the semantic labeling branch. The anchor box scales are adapted for each GSD, so that the respective anchor box areas are equivalent to the mean object dimensions. As shown in Table 6.10, EMT Faster R-CNN achieves the best AP for all GSDs. The gain in AP compared to the baseline Faster R-CNN increases with higher GSDs, which shows that additional semantic context information due to the proposed architecture boosts the detection performance even for tiny objects as in case of high GSDs. IMT Faster R-CNN outperforms Faster R-CNN for low GSDs, while the detection accuracy is almost similar for high

GSDs. This indicates that the impact of implicitly adapting the features of the detection branch by adding an additional semantic labeling branch decreases for high GSDs. The decreasing impact is not unexpected, as the semantic labeling results decrease as well for high GSDs, which may be enforced in this case by the training settings conceived for the detection task. Note that the used joint detection and semantic labeling architectures are designed for low GSDs due to the original GSD of 5 cm. For instance, employing coarser feature maps as demonstrated in Section 5.2 can improve the detection performance for high GSDs.

Table 6.10: Average precision of the baseline Faster R-CNN, IMT Faster R-CNN and EMT Faster R-CNN on the Potsdam dataset for different GSDs.

Approach	GSD (in cm)				
	5	10	15	20	25
Faster R-CNN	95.7	93.0	90.5	83.2	67.5
IMT Faster R-CNN	96.1	95.0	91.5	83.2	67.6
EMT Faster R-CNN	**96.8**	**95.8**	**92.4**	**85.7**	**71.2**

Finally, the proposed IMT Faster R-CNN and EMT Faster R-CNN are compared to Faster R-CNN with semantic labeling based filtering introduced in Section 6.2.2 and two further baselines. For the semantic labeling based filtering, FCN-D16 is used to generate semantic labeling masks and eq. (6.1) is used as filter criterion. As further baselines, Faster R-CNN with hard negative mining and an extension of Faster R-CNN termed Multi-Feature Faster R-CNN are considered. Online hard example mining (OHEM) [Shr16b], which yields improved detection results on benchmark datasets, is used for hard negative mining. Hence, an alternative strategy to influence the learning of the employed feature map is examined. The Multi-Feature Faster R-CNN is a straightforward alternative to increase the semantic context information. For this, the output of *conv5_3* is used as additional feature map for the classification stage. For each region proposal, the corresponding features are extracted and fused with features from *conv4_3* via element-wise addition analogously

to EMT Faster R-CNN. Note that all models are trained on augmented data analogous to IMT and EMT Faster R-CNN. For both additional baselines, the settings are adopted from the baseline Faster R-CNN.

The best detection accuracy is achieved for EMT Faster R-CNN that clearly exceeds the AP of the baseline approaches and of Faster R-CNN with semantic labeling based filtering (see Table 6.11). The relatively poor AP achieved for the Multi-Feature Faster R-CNN shows that adding features from the semantic labeling branch, which explicitly learns scene knowledge, is superior compared to simply adding features from deeper layers. IMT Faster R-CNN slightly outperforms Faster R-CNN with OHEM, which indicates that inducing semantic context information by adding an auxiliary semantic labeling branch is a useful way to affect the learning of the feature map employed for detection. Though Faster R-CNN with semantic labeling based filtering improves the baseline Faster R-CNN by 0.3% in AP showing the benefit of the semantic labeling based filtering, the impact on the detection accuracy is minor compared to incorporating semantic labeling directly into the detection framework.

Table 6.11: Average precision and inference time of the IMT Faster R-CNN and EMT Faster R-CNN compared to baseline approaches on the Potsdam dataset.

Approach	AP (in %)	Time (in ms)
Faster R-CNN	95.7	**58**
Multi-Feature Faster R-CNN	95.8	65
Faster R-CNN + OHEM	96.0	**58**
Faster R-CNN + Semantic Filtering	96.0	138
IMT Faster R-CNN	96.1	**58**
EMT Faster R-CNN	**96.8**	65

At last, the inference time is evaluated for all approaches by averaging over the complete test set including 700 image tiles of size 600 × 600 pixels (see Table 6.11). All time measurements are performed on a single GTX TITAN X

GPU using the server setup introduced in Section 5.2.4. Faster R-CNN with semantic labeling based filtering shows by far the worst inference time, because the detection network and the semantic labeling network are computed separately. The inference time of IMT Faster R-CNN is identical to the baseline Faster R-CNN, as the additional semantic labeling branch of IMT Faster R-CNN can be disabled for deployment so that no computational overhead emerges. As features from the semantic labeling branch are used for the classification stage, EMT Faster R-CNN results in a slightly increased runtime at a notable improvement in detection accuracy. Thus, the proposed joint semantic labeling and detection architectures are an efficient way to integrate semantic labeling into Faster R-CNN without notably worsening the inference time.

6.2.4 Adaptation to the DLR 3K Dataset

As shown in Section 6.2.2 and Section 6.2.3, the proposed approaches achieve good detection results on the Potsdam dataset. However, the comprised training data is comparatively poor because of the utilized semantic labeling procedure and the use of ortho-rectified RGB images, which may impede the learning of optimal feature representations. The main issue regarding the utilized semantic labeling procedure is the labeling of vehicles beneath trees as category *tree* though the vehicles are clearly visible (see Figure 4.7), while the employed RGB images comprise distinctive artifacts (see Figure 6.17). To overcome these issues, semantic labeling masks are generated for the DLR 3K dataset within the context of this thesis. A refined and enhanced version of the dataset is made publicly available as aforementioned in cooperation with the DLR. As depicted in Figure 6.18, the six categories of the Potsdam dataset are adopted. In contrast to the Potsdam dataset, vehicles beneath trees that are clearly visible are labeled as category *car*. In addition, the DLR 3K dataset extended by semantic labeling masks allows the comparison of the proposed approaches with further vehicle detection methods. Note that the Potsdam dataset originally designed for the task of semantic labeling of high-resolution aerial imagery comprises no bounding box annotations and thus, is only rarely considered for the task of vehicle detection in literature.

Figure 6.17: Artifacts of the RGB images of the Potsdam dataset due to ortho-rectification.

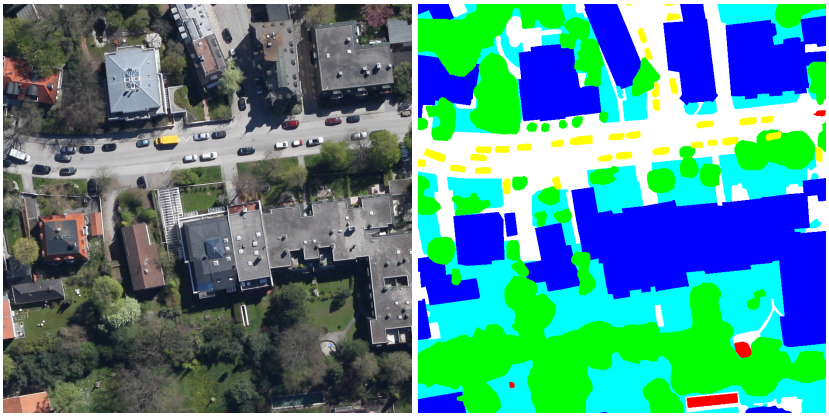


Figure 6.18: Example of the DLR 3K dataset (left) and the corresponding semantic labeling mask (right) with pixel-wise semantic annotations of six categories: *impervious surface* (white), *building* (blue), *low vegetation* (cyan), *tree* (green), *car* (yellow), and *clutter* (red).

The detection results of the proposed semantic labeling based filtering, the IMT Faster R-CNN and the EMT-Faster R-CNN are given in Table 6.12. To account for the smaller object dimensions in case of DLR 3K, the output of *conv3_3* is used as feature map instead of *conv4_3*. Hence, the feature map resolution is increased as required for accurately localizing tiny objects. Furthermore, the anchor base is set to 2 and the anchor scales are set to 7, 14, and 21 analogous to Section 5.2.2. For the semantic labeling based filtering, FCN-D16 is used to generate semantic labeling masks and eq. (6.1) is used as filter criterion. For the EMT Faster R-CNN, the outputs of *conv4_3* and *conv5_3* of the semantic labeling branch are employed as auxiliary feature maps for the classification stage and RoI pooling layers are added to extract the corresponding features for each candidate region. The extracted features of the semantic labeling branch and *conv3_3* are combined via element-wise addition. The further settings are adopted from Section 6.2.2 and Section 6.2.3, respectively. Image patches of size 936 × 936 are used for training and evaluation. Furthermore, data augmentation is conducted during training by applying vertical and horizontal flipping as well as rotation in steps of 90 degrees.

Table 6.12: Average precision of Faster R-CNN with and without semantic labeling based filtering, IMT Faster R-CNN and EMT Faster R-CNN on DLR 3K.

Approach	AP (in %)
Faster R-CNN	95.0
Faster R-CNN + Semantic Filtering	95.2
IMT Faster R-CNN	95.8
EMT Faster R-CNN	**96.2**

The semantic labeling based filtering, the IMT Faster R-CNN and the EMT-Faster R-CNN outperform the baseline Faster R-CNN. Note that the AP of the baseline Faster R-CNN is higher compared to the AP reported in Table 5.2 due to the performed data augmentation. The gain in AP achieved for semantic labeling based filtering is only 0.2% though the employed FCN-D16, which is trained accordingly to Section 6.2.2, exhibits good semantic labeling results

(see Table 6.13). While the overall accuracy and the F1-scores for categories *impervious surface*, *building*, *low vegetation*, and *tree* are in the range of the semantic labeling results achieved on the Potsdam dataset, the F1-score for category *car* is worse due to the high GSD. As depicted in Figure 6.19, predictions of fine structures such as vehicles comprise inaccurate boundaries, which indicates limitations of the applied semantic labeling architecture in case of tiny structures. Inducing semantic context information by adding an additional semantic labeling branch to the detection network exhibits an improvement of 0.8% in AP. Figure 6.20 illustrates that the employed feature maps are affected by the semantic labeling branch. For this, the detections and corresponding activations of four filters are exemplarily depicted for Faster R-CNN (top row) and IMT Faster R-CNN (bottom row). In case of Faster R-CNN, the filters respond to vehicle parts but also to similar structures such as solar cells. In contrast, the filters of IMT Faster R-CNN responding either to structures on buildings or to vehicle parts are more discriminative. EMT Faster R-CNN, which improves the baseline Faster R-CNN by 1.2% in AP, exhibits overall the best detection accuracy. For a confidence threshold of 0.5, the number of false positive detections is reduced by a factor of 32.3%, while the number of false negative detections decreases by 10.6%. Qualitative results given in Figure 6.21 indicate that the number of false positive detections caused by vehicle-like structures on buildings are reduced. Overall, the results achieved on DLR 3K are in good accordance with the results observed on Potsdam dataset. Integrating semantic labeling into the detection framework results in an improved detection accuracy, whereby the largest gain in AP is achieved by explicitly employing features from the semantic labeling branch for the detection task.

Table 6.13: Semantic labeling results for FCN-D16 on DLR 3K. F1-scores are provided for the categories *impervious surface* (IS), *building* (B), *low vegetation* (LV), *tree* (T), *car* (Ca) and *clutter* (Cl).

Sem. Labeling Approach	F1-score (in %)						Overall Accuracy
	IS	B	LV.	T	Ca	Cl	
FCN-D16	86.84	91.38	82.85	87.03	71.82	69.18	86.02

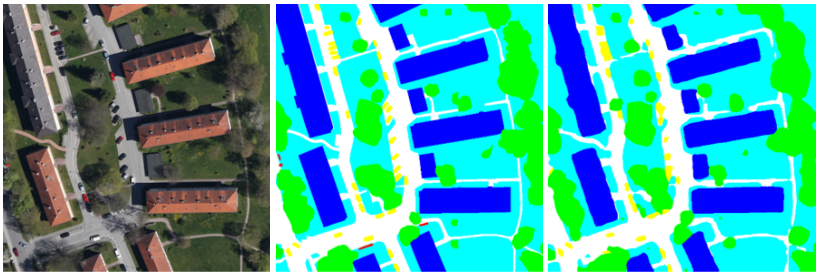


Figure 6.19: Qualitative semantic labeling results of FCN-D16 (left column) on DLR 3K and corresponding GT (middle column) indicate limitations of the employed semantic labeling architecture in case of tiny structures such as vehicles, i.e., inaccurate boundaries.

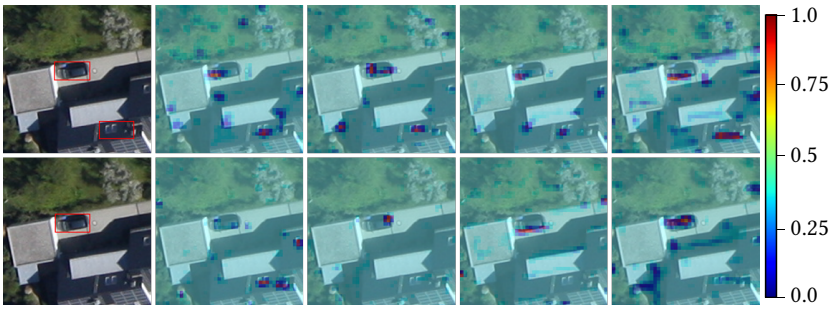


Figure 6.20: Detection results and activations of four filters from *conv3_3* for Faster R-CNN (top row) and IMT Faster R-CNN (bottom row) exhibit that inducing semantic labeling results in more discriminative filters.

Finally, the detection performance of the proposed IMT Faster R-CNN and EMT Faster R-CNN are evaluated for various GSDs. For this, the original images with a GSD of 13 cm are rescaled for training and testing by factor 2/3 and 1/2 yielding GSDs of 19.5 cm and 26 cm, respectively. For each GSD, the employed anchor box areas are equivalent to the mean object dimensions (see Table 5.6). Both IMT Faster R-CNN and EMT Faster R-CNN exhibit an improved detection accuracy compared to the baseline Faster R-CNN for each

GSD, whereby explicitly adding features from the semantic labeling branch results in stronger gain in AP. The improved AP in case of high GSDs shows that the proposed architectures affect the detection performance even for low spatial resolutions.

Figure 6.21: Qualitative detection examples of the baseline Faster R-CNN (top row) and EMT Faster R-CNN with FCN-D16 (bottom row) indicate that false alarms located on regions that are unlikely to contain vehicles, e.g., buildings, are reduced.

Table 6.14: Average precision of the baseline Faster R-CNN, IMT Faster R-CNN and EMT Faster R-CNN on DLR 3K for different GSDs.

Approach	GSD (in cm)		
	13	19.5	26
Faster R-CNN	95.0	91.9	86.0
IMT Faster R-CNN	95.8	93.0	87.3
EMT Faster R-CNN	**96.2**	**93.3**	**88.9**

7 Runtime Optimization

Detecting relevant objects in real-time or nearly in real-time is of great importance across a broad range of applications. In general, deep learning based object detectors, such as Faster R-CNN, are employed for most applications due to their superior detection accuracy compared to conventional counterparts. However, applying Faster R-CNN as object detector is not appropriate for a multitude of applications due to its poor inference time, which is in the context of this thesis further impaired by adapting Faster R-CNN to the characteristics of aerial imagery (see Section 5.2.4). Regarding the inference time for the feature extraction, region proposal generation, and classification stage separately exposes that each component notably contributes to the overall inference time and thus, is often not directly practicable within the detection pipeline. To allow the use of Faster R-CNN for different applications depending on vehicle detection in aerial imagery, acceleration of each component is required.

In the remainder of this chapter, two approaches addressing the optimization of the inference time are presented and discussed in detail. The first approach introduced in Section 7.1 aims at reducing the computational costs for the feature extraction by replacing the default CNN architecture with a more computational efficient CNN architecture. To minimize the computational effort of both the region proposal generation and classification stage, the second approach introduced in Section 7.2 restricts the detection area by a novel Search Area Reduction module. Hence, the number of feature map locations used for region proposal generation is reduced and fewer region proposals need to be classified. The presented approaches in this chapter are mainly based on two of the author's publications [Rin19, Som18a].

7.1 Lightweight Feature Extraction

In the past, most research on CNN architectures has primarily focused on improving the accuracy, yielding deeper and more complex network architectures often unpractical for real-world applications due to their model size and inference time. With the increasing demand for online processing on mobile platforms with limited resources, novel CNN architectures aiming at reducing the number of parameters and computational operations have recently been proposed. Replacing the default CNN architectures used as base network with these computation-efficient architectures bears the potential to reduce the inference time of deep learning based detection frameworks, as feature extraction is one of the most time-consuming parts of these frameworks.

In this thesis, the applicability of such lightweight architectures is examined exemplarily for SSD, which allows a straightforward exchange of the CNN architecture. For this purpose, four promising architectures are adapted as base network for the task of vehicle detection in aerial imagery. In the following, the functional principle of SSD and the structures of the employed CNN architectures are introduced. Furthermore, auxiliary techniques for runtime optimization applied in this thesis are described followed by experiments and discussion of the results. The most promising architectures are then adopted for Faster R-CNN.

7.1.1 Single Shot MultiBox Detector

As described in Section 2.2, SSD is a fully convolutional network whose functional principle is similar to the RPN in Faster R-CNN (see Figure 7.1). In contrast to the class-agnostic RPN introduced in Section 5.1.1, SSD predicts a fixed number of bounding boxes and confidence scores for each object category. For this, $(c + 4)k$ convolutional filters with kernel size 3×3 referred to as classification head are applied at each feature map location to predict four bounding box offsets relative to the anchor boxes termed *default boxes* and c confidence scores, where c is the number of object classes including the

background class and k is the number of anchor boxes. By default, the output of multiple convolutional layers is used as feature map to account for various object scales.

Figure 7.1: Schematic structure of SSD.

For training, the objective loss is equivalent to the multi-task loss of the RPN given in eq. (5.1). The classification loss is the softmax loss over multiple class confidences, while smooth L_1 loss is used as regression loss. Note that in case of vehicle detection with only one category, the classification loss is equivalent to the two-class softmax of the RPN. Regression to the anchor boxes is performed analogously to the RPN by using eq. (5.5).

In the context of this thesis, only a single feature map is exploited as the examined aerial imagery datasets comprise images with a homogenous GSD and thus, the vehicle dimensions exhibit only small variations. As observed in [Rin19], using multiple layers may even yield a worse detection accuracy. Moreover, truncating all layers after the exploited feature map also reduces the computational costs. To further account for the characteristics of the aerial imagery, the anchor box scales are adopted according to the vehicle size distribution, i.e., the anchor box scale is set to 28 pixels and the aspect ratios are set to 1:1, 1:2 and 2:1.

In the following, VGG16, which is used by default, is employed as base network for the SSD detector baseline. VGG16 pre-trained on ImageNet is used

to initialize the weights. The baseline model is trained for 20,000 iterations using the SGD optimizer with a batch size of 32 and an initial learning rate of 0.0001. All other hyper-parameters remain unchanged from the original settings[1]. During deployment, the 2500 predictions with the highest confidence score are considered for NMS. The top-200 predictions after NMS are used as final detections.

7.1.2 Computation-Efficient CNN Architectures

Several computation-efficient CNN architectures have recently been proposed, achieving comparable accuracies compared to conventional CNN architectures, while the inference time is considerably reduced. In the following, the most promising of these architectures that are examined in this thesis are presented and adaptations for the usage as base network within a deep learning based vehicle detector are described.

MobileNet

MobileNet [How17] is a recently proposed network architecture specially developed for mobile and embedded vision applications. Instead of standard 3×3 convolutions, depthwise separable convolutions (see Figure 2.11) are used as main building block. Due to the factorization of a standard 3×3 convolution into a 3×3 depthwise convolution followed by a 1×1 pointwise convolution, the computational costs and the number of parameters are considerably reduced compared to the standard 3×3 convolution. Except for an initial 3×3 convolution and the final fully connected layer used for classification, MobileNet only comprises stacked DSCs as depicted in Table 7.1. Strided convolutions are used for down-sampling instead of pooling, which provides a cheap way to decrease the input size. All convolutional layers are followed by batch normalization and ReLU nonlinearity to stabilize and speed up the training.

[1] https://github.com/weiliu89/caffe/tree/ssd

Table 7.1: Schematic structure of MobileNet that is mainly comprised of depthwise separable convolutions. g denotes the number of groups for the respective depthwise (dw) convolution. Note that the initial network used for classification is designed for input images of size 224 × 224 pixels. Both hyper-parameters, i.e., ξ and ρ, are set by default to 1.

Layer Type	Kernel Size	Stride, Pad	Output Dim.
convolution	3 × 3 × 32	2, 1	112 × 112 × 32
dw convolution	3 × 3 × 32, g=32	1, 1	112 × 112 × 32
convolution	1 × 1 × 64	1, 0	112 × 112 × 64
dw convolution	3 × 3 × 64, g=64	2, 1	56 × 56 × 64
convolution	1 × 1 × 128	1, 0	56 × 56 × 128
dw convolution	3 × 3 × 128, g=128	1, 1	56 × 56 × 128
convolution	1 × 1 × 128	1, 0	56 × 56 × 128
dw convolution	3 × 3 × 128, g=128	2, 1	28 × 28 × 128
convolution	1 × 1 × 256	1, 0	28 × 28 × 256
dw convolution	3 × 3 × 256, g=256	1, 1	28 × 28 × 256
convolution	1 × 1 × 256	1, 0	28 × 28 × 256
dw convolution	3 × 3 × 256, g=256	2, 1	14 × 14 × 256
convolution	1 × 1 × 512	1, 0	14 × 14 × 512
5× dw convolution	3 × 3 × 512, g=512	1, 1	14 × 14 × 512
5× convolution	1 × 1 × 512	1, 0	14 × 14 × 512
dw convolution	3 × 3 × 512, g=512	2, 1	7 × 7 × 512
convolution	1 × 1 × 1024	1, 0	7 × 7 × 1024
dw convolution	3 × 3 × 1024, g=1024	1, 1	7 × 7 × 1024
convolution	1 × 1 × 1024	1, 0	7 × 7 × 1024
avg pooling	7 × 7	-	1 × 1 × 1024
fully connected			1000

Two global hyper-parameters are introduced in order to adjust the speed/accuracy trade-off: the width multiplier ξ and the resolution multiplier ρ. The width multiplier $\xi \in (0,1]$ reduces the number of input channels D_{in} to ξD_{in} and the number of output channels D_{out} to ξD_{out} for each layer, which leads

to a thinner network with considerably less parameters and reduced computational costs. By applying the resolution multiplier $\rho \in (0,1]$ on the input image, the width W and height H of the input image and each subsequent layer are reduced to ρW and ρH, respectively. Though reducing the input image and the internal representation of every layer reduces the computational costs by ρ^2, it is not practicable for the task of vehicle detection in aerial imagery due to the already small object dimensions.

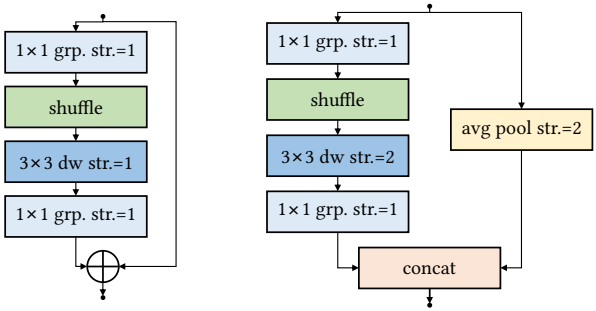
For the usage of MobileNet as base network within the SSD detector, only the output of the 5$^{\text{th}}$ DSC layer is exploited as feature map, as deeper layers would result in coarser feature map resolutions that are unsuited to accurately localize small objects. Thus, all deeper layers are truncated, yielding a clearly reduced model size. The official MobileNet weights pre-trained on ImageNet are used for initialization[1]. Therefore, the weights are converted from the TensorFlow format to the Caffe format, as no pre-trained weights are directly available for Caffe. Due to findings in preliminary experiments, each model is trained for 45,000 iterations using the Adam optimizer with a batch size of 16 and an initial learning rate of 0.001.

ShuffleNet

ShuffleNet [Zha18b] designed especially for mobile devices with very limited computing power employs DSCs to reduce the computational costs. In contrast to MobileNet, 1 × 1 convolutional layers are inserted before each DSC. Thus, the number of input channels and consequently computational operations for the 3 × 3 depthwise convolutions are reduced. By replacing conventional 1 × 1 convolutions with cheaper 1 × 1 group convolutions, the number of parameters and computational costs are further reduced. Depending on the group count g, a group convolution filter only considers the D/g channels of its respective group as input instead of the full channel depth D. Channel shuffling (see Figure 2.12) is applied after the auxiliary 1 × 1 group convolutions

[1] https://github.com/tensorflow/models/blob/master/research/slim/nets/mobilenet_v1.md

to overcome the side effect caused by group convolutions that outputs from a certain channel are only derived from a small fraction of input channels.



(a) ShuffleNet Unit A **(b)** ShuffleNet Unit B

Figure 7.2: Main building blocks of ShuffleNet.These so-called ShuffleNet Units are composed of depthwise (dw) convolutions, group convolutions and channel shuffling. Note that the initial 1×1 group convolution and the 3×3 depthwise convolution comprise the same number of channels C, while the last 1×1 group convolutional layer has $4C$ channels.

ShuffleNet mainly comprises two building blocks called *ShuffleNet Units* based on depthwise convolutions, group convolutions and the novel channel shuffle operation (see Figure 7.2). Both building blocks comprise a sequence of two group convolutions, a channel shuffle layer, and depthwise convolution and a residual branch. Batch normalization is applied for each convolutional layer. ReLU is used as activation function after the initial group convolution, while no activation function is applied after depthwise convolution as suggested by [Cho17]. Building blocks with stride 2 are modified by adding 3×3 average pooling to the shortcut path and replacing the element-wise addition with concatenation, which allows to enlarge the channel dimension with little extra computational costs. ShuffleNet $1\times(g{=}3)$ provides the best speed/accuracy trade-off according to the authors [Zha18b]. The overall architecture shown in Table 7.2 consists of an initial 3×3 convolution and a subsequent max pooling layer with stride 2 followed by a stack of ShuffleNet Units and a final fully connected layer used for classification. The first building block in each stage is

applied with stride 2 to down-sample the input dimensions. Note that a standard 1 × 1 convolution is applied as first convolution in the initial ShuffleNet Unit and the subsequent channel shuffle operation is discarded.

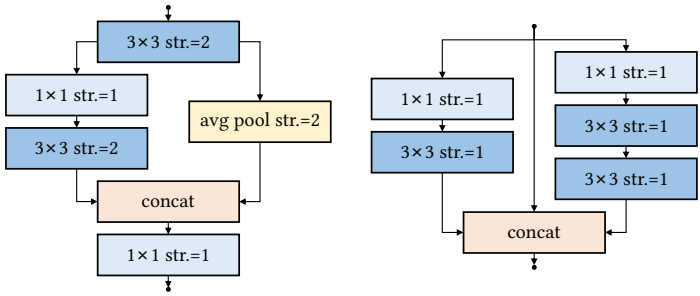
Table 7.2: Schematic structure of ShuffleNet 1×(g=3). For each ShuffleNet Unit, the kernel size, number of channels, stride and padding are given for the depthwise convolution. The initial 1 × 1 group convolution has the same number of channels, while the last 1 × 1 group convolution comprises 4 times the number of channels. The original network used for classification is designed for input images of size 224 × 224 pixels.

Layer Type	Kernel Size	Stride, Pad	Output Dim.
convolution	3 × 3 × 24	2, 1	112 × 112 × 24
max pooling	3 × 3	2	56 × 56 × 24
ShuffleNet Unit B	3 × 3 × 54	2, 1	28 × 28 × 240
3× ShuffleNet Unit A	3 × 3 × 60	1,1	28 × 28 × 240
ShuffleNet Unit B	3 × 3 × 60	2, 1	14 × 14 × 480
7× ShuffleNet Unit A	3 × 3 × 120	1, 1	14 × 14 × 480
ShuffleNet Unit B	3 × 3 × 120	2, 1	7 × 7 × 960
3× ShuffleNet Unit A	3 × 3 × 240	1, 1	7 × 7 × 960
avg pooling	7 × 7	-	1 × 1 × 960
fully connected			1000

Within this thesis, ShuffleNet 1×(g=3) is used as base network for the SSD detector. To this end, the output of the 4$^{\text{th}}$ ShuffleNet Unit is used as feature map, while all deeper layers are removed from the network. Weights pre-trained on ImageNet are used to initialize the model. Each model is trained for 30,000 iterations using the Adam optimizer with a batch size of 16 and an initial learning rate of 0.001.

PeleeNet

Unlike MobileNet and ShuffleNet, PeleeNet [Wan18b] foregoes the use of special layers, such as depthwise convolutions or channel shuffling, due to the lack of efficient implementations in most deep learning frameworks and only comprises conventional convolutions instead. PeleeNet, which is inspired by DenseNet [Hua17], is comprised of two building blocks termed *Stem Block* and *Two-Way Dense Layer*, respectively (see Figure 7.3). Both building blocks utilize 1×1 convolutions to restrict the amount of channels for subsequent 3×3 convolutions and thus, reduce the computational costs. The Two-Way Dense Layer comprises two ways with various kernel sizes in order to achieve different scales of receptive fields and consequently to account for different object dimensions.



(a) Stem Block **(b)** Two-Way Dense Layer

Figure 7.3: Main building blocks of PeleeNet.

The overall architecture given in Table 7.3 is composed of an initial Stem Block used to enhance the feature expression ability in a computational efficient manner followed by a sequence of stacked Two-Way Dense Layers and a final fully connected layer used for classification. Note that the number of kernels remains constant throughout the network for all 3×3 convolutional layers to save computational costs. Therefore, 1×1 convolutions with increasing

number of kernels are added after the last Two-Way Dense Layer of each stack in order to improve the representational abilities. Except for the last stack, average pooling with stride 2 is applied after these 1 × 1 convolutions to reduce the input dimensions for the subsequent stack. Batch normalization and ReLU nonlinearity are applied for each convolutional layer.

Table 7.3: Schematic structure of PeleeNet. The convolutions of the initial Stem Block comprise 32 channels except for the first 1 × 1 convolution that possesses 16 channels. For each Two-Way Dense Layer, the kernel size, number of channels, stride and padding are given for the 1 × 1 convolutions, as the number of channels, i.e., 16, remains constant throughout the network for all 3 × 3 convolutions. Note that the initial network is designed for input images of size 224 × 224 pixels.

Layer Type	Kernel Size	Stride, Pad	Output Dim.
Stem Block			56 × 56 × 32
3×Dense Layer	1 × 1 × 16	1, 1	56 × 56 × 128
convolution	1 × 1 × 128	1, 0	56 × 56 × 128
avg pooling	2 × 2	2	28 × 28 × 128
4×Dense Layer	1 × 1 × 32	1, 1	28 × 28 × 256
convolution	1 × 1 × 256	1, 0	28 × 28 × 256
avg pooling	2 × 2	2	14 × 14 × 256
8×Dense Layer	1 × 1 × 64	1, 1	14 × 14 × 512
convolution	1 × 1 × 512	1, 0	14 × 14 × 512
avg pooling	2 × 2	2	7 × 7 × 512
6×Dense Layer	1 × 1 × 64	1, 1	7 × 7 × 704
convolution	1 × 1 × 704	1, 0	7 × 7 × 704
avg pooling	7 × 7	-	1 × 1 × 704
fully connected			1000

For the usage of PeleeNet as base network within the SSD detector, only the output of the 1 × 1 convolutional layer after the 2nd stack of Two-Way Dense Layers is exploited as feature map. All deeper layers are truncated from the network. In contrast to the other computation-efficient networks, weights

pre-trained on PASCAL VOC and MS COCO are used for initialization[1], as no weights pre-trained on ImageNet were publicly available at that time. Each model is trained for 30,000 iterations using the Adam optimizer with a batch size of 16 and an initial learning rate of 0.001.

SqueezeNet

SqueezeNet [Ian16] does not make use of special layers and instead only leverages its building block called *Fire module*, which only consists of standard convolutions. The Fire module visualized in Figure 7.4 is comprised of an initial 1 × 1 convolutional layer referred to as *squeeze layer* followed by a combination of 1 × 1 and 3 × 3 convolutions termed *expand layer*[2]. The squeeze layer decreases the number of input channels for the subsequent expand layer, which reduces in particular the computational effort for its 3 × 3 convolutions. By increasing the number of channels, the expand layer improves the representational abilities. Note that the computational costs of the expand layers are reduced due to the combination of 3 × 3 with less computationally expensive 1 × 1 convolutions.

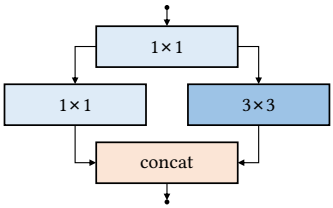


Figure 7.4: Main building block of SqueezeNet denoted as Fire module.

[1] https://github.com/Robert-JunWang/Pelee

[2] Note that the Caffe framework does not natively support convolutional layers with multiple kernel sizes. Thus, separate convolutional layers with different kernel sizes are used and their outputs are concatenated.

Table 7.4: Schematic structure of SqueezeNet v1.0 that mainly consists of so-called *Fire modules* (see Figure 7.4). Each *Fire module* comprises initial 1 × 1 convolutions followed by parallel 1 × 1 and 3 × 3 convolutions that are combined via concatenation. The original network used for classification is designed for input images of size 227 × 227 pixels.

Layer Type	Kernel Size	Stride, Pad	Output Dim.
convolution	7 × 7 × 96	2, 3	114 × 114 × 96
max pooling	3 × 3	2	57 × 57 × 96
2× convolution	1 × 1 × 16	1, 0	57 × 57 × 16
convolution	1 × 1 × 64 | 3 × 3 × 64	1, 0 | 1, 1	57 × 57 × 128
convolution	1 × 1 × 32	1, 0	57 × 57 × 32
convolution	1 × 1 × 128 | 3 × 3 × 128	1, 0 | 1, 1	57 × 57 × 256
max pooling	3 × 3	2	28 × 28 × 256
convolution	1 × 1 × 32	1, 0	28 × 28 × 32
convolution	1 × 1 × 128 | 3 × 3 × 128	1, 0 | 1, 1	28 × 28 × 256
2× convolution	1 × 1 × 48	1, 0	28 × 28 × 48
convolution	1 × 1 × 192 | 3 × 3 × 192	1, 0 | 1, 1	28 × 28 × 384
convolution	1 × 1 × 64	1, 0	28 × 28 × 64
convolution	1 × 1 × 256 | 3 × 3 × 256	1, 0 | 1, 1	28 × 28 × 512
max pooling	3 × 3	2	14 × 14 × 512
convolution	1 × 1 × 64	1, 0	14 × 14 × 64
convolution	1 × 1 × 256 | 3 × 3 × 256	1, 0 | 1, 1	14 × 14 × 512
convolution	1 × 1 × 1000	1, 0	14 × 14 × 1000
avg pooling	14 × 14	-	1 × 1 × 1000

The SqueezeNet v1.0 network architecture given in Table 7.4 is comprised of an initial 7 × 7 convolutional layer followed by 8 Fire modules. Max pooling with a stride of 2 is performed after the initial convolutional layer, the 3rd and 7th Fire module. The relatively late placements of pooling layers lead to relatively large feature maps, referred to as *activation maps*, throughout the network, which is motivated by the intuition that large activation maps result

in higher accuracy. The output of the final average pooling layer is used for classification.

Table 7.5: Schematic structure of SqueezeNet v1.1, which mainly consists of so-called *Fire modules* similar to SqueezeNet v1.0. Note that the original network used for classification is designed for input images of size 227x227 pixels.

Layer Type	Kernel Size	Stride, Pad	Output Dim.
convolution	3 × 3 × 64	2, 1	113 × 113 × 64
max pooling	3 × 3	2	56 × 56 × 64
2× convolution	1 × 1 × 16	1, 0	56 × 56 × 16
convolution	1 × 1 × 64 | 3 × 3 × 64	1, 0 | 1, 1	56 × 56 × 128
max pooling	3 × 3	2	28 × 28 × 128
2× convolution	1 × 1 × 32	1, 0	28 × 28 × 32
convolution	1 × 1 × 128 | 3 × 3 × 128	1, 0 | 1, 1	28 × 28 × 256
max pooling	3 × 3	2	14 × 14 × 256
2× convolution	1 × 1 × 48	1, 0	14 × 14 × 48
convolution	1 × 1 × 192 | 3 × 3 × 192	1, 0 | 1, 1	14 × 14 × 384
2× convolution	1 × 1 × 64	1, 0	14 × 14 × 64
convolution	1 × 1 × 256 | 3 × 3 × 256	1, 0 | 1, 1	14 × 14 × 512
convolution	1 × 1 × 1000	1, 0	14 × 14 × 1000
avg pooling	14 × 14	-	1 × 1 × 1000

Instead of computationally expensive 7 × 7 convolutions, SqueezeNet v1.1 (see Table 7.5), which is built on SqueezeNet v1.0, utilizes less expensive 3 × 3 convolutions to improve the inference speed. By performing max pooling after earlier layers, i.e., the initial convolutional layer, the 2nd and 4th Fire module, the computational costs are further reduced due to smaller input volumes for some convolutional layers. Based on SqueezeNet v1.1, Gschwend [Gsc16] proposed ZynqNet (see Table 7.6), which improves the classification accuracy by an alternating usage of 3 × 3 and 1 × 1 convolutions for the squeeze layers.

Furthermore, convolutions with stride 2 are used for down-sampling instead of max pooling. In contrast to the other lightweight CNN architectures presented in this section, no batch normalization is conducted. ReLU is used as activation function for each convolutional layer except for the last convolutional layer.

Table 7.6: Schematic structure of ZynqNet, which alternatingly employs 3 × 3 and 1 × 1 convolutions as *squeeze layers* in the *Fire modules*. The stride of these 3 × 3 convolutions is set to 2 to conduct down-sampling, while max pooling layers are discarded. The initial network used for classification is designed for input images of size 256 × 256 pixels.

Layer Type	Kernel Size	Stride, Pad	Output Dim.
convolution	3 × 3 × 64	2, 1	128 × 128 × 64
convolution	3 × 3 × 16	2, 1	64 × 64 × 16
convolution	1 × 1 × 64 | 3 × 3 × 64	1, 0 | 1, 1	64 × 64 × 128
convolution	1 × 1 × 16	1, 0	64 × 64 × 16
convolution	1 × 1 × 64 | 3 × 3 × 64	1, 0 | 1, 1	64 × 64 × 128
convolution	3 × 3 × 32	2, 1	32 × 32 × 32
convolution	1 × 1 × 128 | 3 × 3 × 128	1, 0 | 1, 1	32 × 32 × 256
convolution	1 × 1 × 32	1, 0	32 × 32 × 32
convolution	1 × 1 × 128 | 3 × 3 × 128	1, 0 | 1, 1	32 × 32 × 256
convolution	3 × 3 × 64	2, 1	16 × 16 × 64
convolution	1 × 1 × 256 | 3 × 3 × 256	1, 0 | 1, 1	16 × 16 × 512
convolution	1 × 1 × 64	1, 0	16 × 16 × 64
convolution	1 × 1 × 192 | 3 × 3 × 192	1, 0 | 1, 1	16 × 16 × 384
convolution	3 × 3 × 112	2, 1	8 × 8 × 112
convolution	1 × 1 × 256 | 3 × 3 × 256	1, 0 | 1, 1	8 × 8 × 512
convolution	1 × 1 × 112	1, 0	8 × 8 × 112
convolution	1 × 1 × 368 | 3 × 3 × 368	1, 0 | 1, 1	8 × 8 × 736
convolution	1 × 1 × 512 | 1 × 1 × 512	1, 0	8 × 8 × 1024
avg pooling	8 × 8	-	1 × 1 × 1024

The output of the 4th Fire module is chosen as feature map for using SqueezeNet v1.0, SqueezeNet v1.1, and ZynqNet as base network. Weights pre-trained on ImageNet are used to initialize the SqueezeNet models[1] and ZynqNet[2]. Each model is trained for 35,000 iterations using the Adam optimizer with a batch size of 16 and an initial learning rate of 0.001.

7.1.3 Auxiliary Techniques for Runtime Optimization

Besides the exploitation of computation-efficient CNN architectures, further techniques that are conducted in the context of this thesis to optimize the inference time are presented in the following.

Filter Pruning

As base networks are generally pre-trained for classification on benchmark datasets comprising a large number of classes, e.g., ImageNet with 1000 classes, these networks are often over-parameterized for detection tasks with only a few classes. In literature, there exist several techniques to reduce the number of parameters and consequently the model size such as PCA decomposition [Wen17], random pruning [Li16, Ani17] and one-shot pruning [Li16], whereby latter proved to outperform the other mentioned techniques [Ani17].

In this thesis, the one-shot pruning strategy proposed by Li *et al.* [Li16] is applied to remove filters with redundant information, which are dispensable for the task of vehicle detection in aerial imagery. In an initial stage, the one-shot pruning, which is applied on an already trained network, measures the relative importance of each filter f_i within a convolutional layer by calculating its ℓ_1-norm $\|f_i\|_1$. $\|f_i\|_1$ represents the average magnitude of its kernel weights and thus, gives an expectation of the magnitude of the output feature map, i.e., smaller kernel weights tend to produce output feature maps with weak

[1] https://github.com/DeepScale/SqueezeNet
[2] https://github.com/dgschwend/zynqnet

activations. For each filter f_i within a convolutional layer, $\|f_i\|_1$ is calculated as $\sum_{j=0}^{n_i}|f_{ij}|$ where f_{ij} is the j^{th} kernel weight of filter f_i. For every convolutional layer, the filters are then sorted according to the ℓ_1-norm, which proved to be a good criterion to judge the usefulness of a particular filter [Li16]. Next, a fixed percentage of filters with the lowest ℓ_1-norm is removed from the particular convolutional layer. Finally, the condensed network is re-trained in order to regain any knowledge lost during the pruning process.

Merged Batch Normalization

As described in Section 2.1.3, batch normalization is used to accelerate the training and to make the gradient propagation in the network more stable. Therefore, the network activations are normalized to zero mean and unit variance and transformed through the learned parameters γ_i and β_i as defined in eq. (2.8) and eq. (2.9), respectively, which can be rewritten as:

$$\tilde{\mathbf{h}}_i^{BN} = \gamma_i \frac{(\mathbf{W}_i\mathbf{h}_{i-1} + \mathbf{b}_i) - \mu_{\mathcal{B}}\mathbf{e}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} + \beta_i\mathbf{e}. \tag{7.1}$$

As μ, σ^2, γ_i and β_i are constant during inference, they can be merged into the weights and biases of the previous convolutional layer as proposed in [Fu17]. Note that μ and σ^2 are the average of the mean and variance values computed for each batch during training. The new convolutional layer can then be written as eq. (7.4), where $\hat{\mathbf{W}}_i$ and $\hat{\mathbf{b}}_i$ are the rescaled weights and biases of layer i given in eq. (7.2) and eq. (7.3), respectively.

$$\hat{\mathbf{W}}_i = \gamma_i\left(\frac{\mathbf{W}_i}{\sqrt{\sigma^2 + \epsilon}}\right) \tag{7.2}$$

$$\hat{\mathbf{b}}_i = \gamma_i\left(\frac{\mathbf{b}_i - \mu\mathbf{e}}{\sqrt{\sigma^2 + \epsilon}}\right) + \beta_i\mathbf{e} \tag{7.3}$$

$$\tilde{\mathbf{h}}_i^{BN} = \hat{\mathbf{W}}_i\mathbf{h}_{i-1} + \hat{\mathbf{b}}_i \tag{7.4}$$

Merging batch normalization related variables into the weights and biases of the previous convolutional layers leads to improved inference time, as the additional computational costs of the batch normalization layers are removed.

Computation-Efficient Classification Heads

As described in Section 7.1.1, $(c + 4)k$ convolutional filters with kernel size 3×3 are applied at each feature map location to predict c confidence scores and four bounding box offsets relative to the anchor boxes. In the context of this thesis, the number of categories c is set to 2, i.e., background and category vehicle, and the number of anchor boxes is set to 5, yielding in total 30 convolutional filters. Motivated by the modifications proposed in [San18, Wan18b], the parameter count and computational costs can be reduced by replacing the expensive 3×3 convolutions with more lightweight building blocks. In the following, three different building blocks are considered as classification head, i.e., DSCs, 1×1 convolutions and 1×1 group convolutions.

7.1.4 Experiments and Discussion

The effect of the employed base network on the detection performance is evaluated on the DLR 3K dataset. Table 7.7 gives the detection accuracy by means of AP and the inference time benchmarked on the server and desktop setup introduced in Section 5.2.4. The inference speed reported in frames per second (FPS) is averaged over 500 forward passes on images with size 936×936 pixels. As for Faster R-CNN timings, all time measurements exclude preprocessing steps and 10 forward passes are performed to warm-up the GPU kernels. Note that the benchmarks do not include the NMS stage to better judge the impact of architectural changes.

Using VGG16 as base network exhibits the best AP, which is roughly on par with Faster R-CNN (see Section 5.2). However, the inference time, in particular on the desktop setup, may not be satisfactorily for real-world applications often requiring processing of even larger images, e.g., full HD. Replacing VGG16 with the computation-efficient CNN architectures leads to a vast gain

in inference speed. For MobileNet, ShuffleNet and PeleeNet that are trained with batch normalization layers, the inference times are reported with and without these layers. Using the merging process described in Section 7.1.3 results in considerably improved inference times, while the inference times with batch normalization layers are only slightly better compared to VGG16. Note that the merging process does not affect the detection accuracy, as it is only an identity transformation. Hence, all further results are reported with merged batch normalization layers.

Table 7.7: Comparison of the inference time in FPS for different computation-efficient CNN architectures used as base network for the SSD detector. Models marked with BN include the batch normalization layers during deployment.

Base Network	AP (in %)	Time (in FPS)	
		Server	Desktop
VGG16	**93.8**	17.9	4.1
MobileNet^{BN}	93.6	27.5	11.4
MobileNet	93.6	81.2	30.0
ShuffleNet^{BN}	92.7	33.6	17.6
ShuffleNet	92.7	52.1	29.5
PeleeNet^{BN}	**93.8**	23.2	11.4
PeleeNet	**93.8**	96.1	31.3
SqueezeNet v1.0	93.7	88.8	28.1
SqueezeNet v1.1	93.6	119.8	40.9
ZynqNet	93.7	**121.2**	**41.6**

ZynqNet exhibits overall the best inference time without large drop in detection accuracy. Compared to the VGG16 baseline the inference time is speeded up by a factor of 6.8 and 10.1 on the server and desktop setup, respectively. Among the computation-efficient CNN architectures, using PeleeNet as base network yields the best detection accuracy reaching VGG16-level AP, while outperforming the inference time for MobileNet and ShuffleNet by 14.9 and

44.0 FPS on the server setup. ShuffleNet exhibits overall the worst detection accuracy due to its fast down-sampling strategy to reduce computational costs. For this reason, ShuffleNet is not considered for further experiments.

Impact of Filter Pruning

To further accelerate the inference time, the number of parameters are reduced by performing the filter pruning strategy described in Section 7.1.3. For this, two different pruning thresholds are considered, removing either 25% or 50% of the filters with the lowest ℓ_1-norm. In case of PeleeNet, convolutional layers of the initial Stem Block and of the Two-Way Dense Layers comprising only maximal 16 channels are saved from pruning. In case of ZynqNet, squeeze layers that possess already a small number of channels are saved from pruning. Preliminary experiments showed that pruning layers with an already small number of channels results in worse detection accuracy, as too small number of channels are not enough to retain all the information. In case of MobileNet, the width multiplier ξ is set to 0.5 and 0.75, respectively, which is an alternative strategy to reduce the number of channels. Note that for training, each MobileNet model is initialized with the official weights pretrained on ImageNet for the particular ξ.

The impact of reducing the number of parameters on the detection performance is given in Table 7.8. On both setups, using ZynqNet with 50% removed filters as base network results in the highest number of FPS, which is roughly increased by a factor of 1.5 compared to the unpruned network. Compared to the VGG16 baseline, the inference time is even speeded up by a factor of 10.2 and 15.0 on the server and desktop setup. The vast gain in inference speed by removing filters is achieved for MobileNet, as in contrast to PeleeNet and ZynqNet the number of channels is reduced for all convolutional layers. Removing filters in case of PeleeNet only slightly improves the inference time, while the detection accuracy remains almost constant. The small gain in inference time is due to the limited number of layers that are pruned. Note that typically the computational effort for these layers is already reduced by a preceding 1×1 convolutional layer minimizing the amount of input channels. The

almost constant detection accuracy indicates that even computation-efficient CNN architectures are over-parameterized for the task of vehicle detection in aerial imagery and can be further condensed. This is confirmed by the only slightly decreasing AP for MobileNet$_{\xi=0.75}$ and ZynqNet$_{\alpha=0.75}$.

Table 7.8: Comparison of the inference time in FPS for condensed base networks. Therefore, the number of parameters of MobileNet is reduced by setting the width multiplier ξ to 0.5 and 0.75, respectively, while one-shot pruning is applied on PeleeNet and ZynqNet keeping the α% of channels with the highest ℓ_1-norm.

Base Network	AP (in %)	Time (in FPS)	
		Server	Desktop
MobileNet$_{\xi=1.00}$	93.6	81.2	30.0
MobileNet$_{\xi=0.75}$	93.2	98.6	37.3
MobileNet$_{\xi=0.50}$	92.3	131.0	51.0
PeleeNet$_{\alpha=1.00}$	**93.8**	96.1	31.3
PeleeNet$_{\alpha=0.75}$	93.7	99.6	31.8
PeleeNet$_{\alpha=0.50}$	93.6	106.9	35.6
ZynqNet$_{\alpha=1.00}$	93.7	121.2	41.6
ZynqNet$_{\alpha=0.75}$	93.6	143.9	49.5
ZynqNet$_{\alpha=0.50}$	93.1	**181.7**	**61.6**

Impact of Computation-Efficient Classification Heads

The effect of replacing the standard classification head comprised of 3 × 3 convolutions with more lightweight building blocks is evaluated in Table 7.9. For this purpose, PeleeNet and ZynqNet are used as base networks, exhibiting so far the best AP and best inference time, respectively. The number of groups is set to 2 for the experiments involving group convolutions, so that the number of input and output channels are evenly divided. Two variants employing DSCs are evaluated, as the convolutional filters for the classification and

regression are natively split into two separate branches, i.e., one for classification and one for regression. The first variant pursues the native procedure and employs separate DSCs for each branch. The second variant referred to as shared DSC shares the first 3 × 3 depthwise convolutions for both branches, while the subsequent pointwise convolutions are divided for each branch.

Table 7.9: Comparison of the inference time in FPS for different building blocks used as classification head.

Base Network	Classification Head	AP (in %)	Time (in FPS)	
			Server	Desktop
PeleeNet	3 × 3 conv.	**93.8**	96.1	31.3
PeleeNet	1 × 1 conv.	**93.8**	98.5	32.1
PeleeNet	1 × 1 group conv.	**93.8**	98.6	32.1
PeleeNet	separated DSC	**93.8**	86.4	29.6
PeleeNet	shared DSC	**93.8**	92.7	30.9
ZynqNet	3 × 3 conv.	93.7	121.2	41.6
ZynqNet	1 × 1 conv.	93.7	124.3	43.0
ZynqNet	1 × 1 group conv.	93.7	**124.8**	**43.1**
ZynqNet	separated DSC	**93.8**	106.5	38.3
ZynqNet	shared DSC	93.7	113.8	41.3

The results indicate that every building block could replace the standard 3 × 3 convolutions as classification head without loss of accuracy, which is in accordance with observations on detection benchmark datasets [San18, Wan18b]. In terms of inference time, exploiting 1 × 1 convolutions or 1 × 1 group convolutions leads to a slightly increased number of FPS, while models involving depthwise separable convolutions experience a slightly decreased number of FPS. A reason for this small drop in inference speed is the additional memory access by employing a classification head comprised of two layers, i.e., depthwise and pointwise convolutions, instead of a single layer. Thus, using the shared variant reduces this effect and results in an increased number of FPS compared to its counterpart.

Finally, the different architectural modifications from the experiments above are combined for PeleeNet and ZynqNet. For this, one-shot pruning is applied with α=0.5 and 1 × 1 convolutions are employed as classification head, replacing the computational more expensive 3 × 3 convolutions. Note that 1 × 1 group convolutions are not applied due to their slightly worse AP in case of ZynqNet (see Table 7.9). The resulting inference times and detection accuracies are given in Table 7.10. The final PeleeNet exhibits a considerably improved inference speed compared to the VGG16 baseline, i.e., by factor 6.1 and 8.8 on the server and desktop setup, while the detection accuracy is on par with the VGG16 baseline. The final ZynqNet achieves the highest inference speed, outperforming the final PeleeNet and the VGG16 baseline by factor 1.7 (1.8) and 10.5 (15.5), respectively, on the server setup (desktop setup). However, the detection accuracy is slightly worse compared to PeleeNet and VGG16. In addition to a speed-up, the performed architectural modifications lead to considerably reduced numbers of parameters, in particular, due to the application of the lightweight CNN architectures. Filter pruning and replacing the classification head further decrease the parameter count of PeleeNet and ZynqNet by 47.1% and 63.3%. The clearly improved inference time and significantly smaller model size allow now the usage on mobile platforms with limited resources.

Table 7.10: Comparison of the parameter count and inference time for selected CNN architectures. The final CNN architectures involve architectural modifications from the experiments above, i.e., one-shot pruning with α=0.5 and 1 × 1 convolutions as classification head.

Base Network	Parameter Count	AP (in %)	Time (in FPS)	
			Server	Desktop
VGG16	7,774,046	**93.8**	17.9	4.1
PeleeNet	278,558	**93.8**	96.1	31.3
PeleeNet$_{\alpha=0.50;\ 1\times1\ \text{conv.}}$	147,422	93.6	108.3	36.1
ZynqNet	230,782	93.7	121.2	41.6
ZynqNet$_{\alpha=0.50;\ 1\times1\ \text{conv.}}$	84,734	93.0	**187.1**	**63.5**

7.1.5 Adoption to Faster R-CNN

In the following, ZynqNet is adopted as base network for Faster R-CNN, as it possesses the largest speed-up for the SSD detector. For this purpose, the output of the 4$^{\text{th}}$ Fire module is chosen as feature map analogous to the SSD detector. Note that the stride of the initial 3×3 convolutional layer is set to 1 in the 3$^{\text{rd}}$ Fire module. Thus, the employed feature map is only down-sampled by a factor of four as required for an accurate localization of small objects in case of high GSDs (see Section 5.2.3). In contrast to SSD, Faster R-CNN comprises an additional classification stage, which has to be adapted for the new base network. As described in Section 5.1.2, the classification stage is composed of a RoI pooling layer, which converts for each region proposal the corresponding features into a feature map with fixed spatial extent, followed by a sequence of fully connected layers branching in two fully connected layers: one for classification and one for bounding box regression. Note that by default the sequence of fully connected layers is adopted from the VGG16 base network. Due to the absence of fully connected layers in ZynqNet, different combinations of layers are examined as so-called classification head. The first examined classification head, in the following termed ZynqNet-Head, comprises all layers from ZynqNet after the 5$^{\text{th}}$ Fire module through the 9$^{\text{th}}$ Fire module followed by the two sibling fully connected layers for classification and regression, which remain unchanged for all setups. Furthermore, the default classification head termed VGG-Head is employed, whereby the fully connected layers are randomly initialized by using the Gaussian weight filler method. To retain a small parameter count, two variations of the VGG-Head classification head comprising fewer parameters are examined. For this, the number of outputs of each fully connected layer is reduced by factor 2 and 4, respectively.

The impact of replacing the base architecture of Faster R-CNN on the detection accuracy and inference time is depicted in Table 7.11. For this, the time measurements are performed in accordance to Section 5.2.4. All models employing ZynqNet as base network are trained for 40,000 iterations using the training settings introduced in Section 5.1.3. Weights pre-trained on ImageNet are used for initializing the base network. Employing ZynqNet as

base network results in a significantly reduced inference time on both setups. Using the ZynqNet-Head as classification head exhibits slightly worse AP in comparison to the VGG-Head, while the inference time is notably reduced especially on the desktop setup. Employing the $VGG_{0.5}$-Head, whose fully connected layers comprise only 2048 outputs instead of 4096, yields inference times similar to the ZynqNet-Head, while the detection accuracy is on par with the VGG-Head. Decreasing the number of outputs further to speed up the inference time results in worse AP.

Table 7.11: Comparison of the detection accuracy and the inference time for Faster R-CNN with different base networks on the DLR 3K dataset. For the ZynqNet architecture, different combinations of layers are examined as classification head due to the absence of fully connected layers.

Base Network	Classification Head	AP (in %)	Time (in ms)	
			Server	Desktop
VGG16	VGG-Head	**94.3**	286.2	663.5
ZynqNet	ZynqNet-Head	93.8	164.9	227.8
ZynqNet	VGG-Head	94.1	188.0	313.2
ZynqNet	$VGG_{0.5}$-Head	94.1	166.7	227.0
ZynqNet	$VGG_{0.25}$-Head	93.3	**156.9**	**196.1**

The impact of replacing the base network on the inference time for each component is given in Table 7.12. Using ZynqNet, which is more computation-efficient than VGG16, considerably reduces the time spent for feature extraction, i.e., by 63% on the server and 78% on the desktop setup. Besides the considerably reduced time spent for feature extraction, the inference time of the RPN and classification stage are improved as well, as the employed feature map comprises fewer channels and consequently, the number of input channels is reduced for both stages. Employing the ZynqNet-Head instead of the VGG-Head results in roughly halved inference time for the classification stage. Similar inference times are achieved on both setups by halving

the number of outputs of the fully connected layers of the VGG-Head. Thus, $VGG_{0.5}$-Head is applied as classification head in the following.

Table 7.12: Comparison of the inference time for each component of the Faster R-CNN with different base networks.

Base Network	Component	Time (in ms) Server	Desktop
VGG16	Feature Extractor	58.9	177.4
	RPN	139.8	212.8
	Classification Stage - VGG-Head	87.5	273.3
ZynqNet	Feature Extractor	21.7	38.9
	RPN	122.4	119.8
	Classification Stage - ZynqNet-Head	20.8	69.1
	Classification Stage - VGG-Head	43.9	154.5
	Classification Stage - $VGG_{0.5}$-Head	22.6	68.3
	Classification Stage - $VGG_{0.25}$-Head	12.8	37.4

Table 7.13 shows the detection accuracy for Faster R-CNN using ZynqNet as base network with respect to various GSDs. Compared to VGG16, employing ZynqNet as base network results only in slightly worse AP, which indicates the applicability of the proposed computation-efficient CNN architecture even for high GSDs.

Table 7.13: AP (in %) for Faster R-CNN using VGG16 and ZynqNet as base network with respect to the GSD. $VGG_{0.5}$-Head is applied as classification head in case of ZynqNet.

Base Network	GSD (in cm) 13	19.5	26
VGG16	**94.3**	**89.4**	**83.2**
ZynqNet	94.1	89.1	82.0

Impact of Filter Pruning

To further reduce the inference time, the one-shot pruning strategy described in Section 7.1.3 is applied. The effect of removing filters from the base network on the detection accuracy and inference time is given in Table 7.14. For this, either 75% or 50% of the filters with the highest ℓ_1-norm are kept. According to Section 7.1.4, squeeze layers are saved from pruning due to their already small number of channels. Each condensed network is then re-trained to regain any knowledge lost during the pruning process. The inference time clearly decreases by removing filters from the base network especially on the desktop setup. Removing 25% of the filters reduces the time spent for feature extraction by roughly 20% on both setups, while the detection accuracy is almost on par with the unpruned network. The AP slightly drops when 50% of the filters are removed, while the time spent for feature extraction decreases by 36% and 43% on the server and desktop setup, respectively. The inference time for the RPN and classification stage are also reduced because of the smaller number of input channels.

Impact of Modified RPN

So far, only the classification head, i.e., layers of the classification stage, has been modified to speed up the inference time without loss in detection accuracy. Modifying the prediction layers of the RPN further reduces the inference time as shown in Table 7.15. For this purpose, the initial 3×3 convolutional layer comprising 512 channels is removed, so that the RPN is only composed of two sibling 1×1 convolutional layers. Thus, the structure of the RPN is equivalent to the prediction layers of the SSD detector, which is in essence a class-specific RPN. The modified RPN decreases the time spent for generating candidate regions and consequently, the overall inference time. The speed-up is more distinctive in case of the unpruned network due to the higher number of input channels for the RPN. The AP remains almost unchanged, which confirms the results achieved for SSD with different classification heads, namely, that 1×1 convolutions are adequate as prediction layers for vehicle detection in aerial imagery.

Table 7.14: Comparison of the overall inference time and the time spent for each component of the Faster R-CNN with pruned base networks. Therefore, the α% of channels with the highest ℓ_1–norm are kept.

Base Network	Component	AP (in %)	Time (in ms)	
			Server	Desktop
$\text{ZynqNet}_{\alpha=1.00}$	Feature Extractor	94.1	21.7	38.9
	RPN		122.4	119.8
	Classification Stage		22.6	68.3
	Total		166.7	227.0
$\text{ZynqNet}_{\alpha=0.75}$	Feature Extractor	94.0	17.5	30.4
	RPN		120.9	113.6
	Classification Stage		15.7	50.3
	Total		154.1	194.3
$\text{ZynqNet}_{\alpha=0.50}$	Feature Extractor	93.7	13.8	22.3
	RPN		117.9	109.4
	Classification Stage		12.4	34.9
	Total		144.1	166.6

Table 7.15: Comparison of the overall inference time and the time spent for the RPN in case of adapted prediction layers for the RPN.

Base Network	Component	AP (in %)	Time (in ms)	
			Server	Desktop
$\text{ZynqNet}_{\alpha=1.00}$	RPN	94.1	117.8	109.2
	Total		162.1	216.4
$\text{ZynqNet}_{\alpha=0.75}$	RPN	94.0	116.5	106.1
	Total		149.7	186.9
$\text{ZynqNet}_{\alpha=0.50}$	RPN	93.5	114.7	103.5
	Total		140.9	160.7

The performed adaptations to the base network and prediction layers yield significantly decreased inference times, whereby in particular the time spent for the feature extraction and for the classification stage are clearly reduced on both setups. The bottleneck in terms of inference time is the RPN. While

the inference time of its prediction layers is relatively small, the most time-consuming parts of the RPN are the mapping of the predictions to the image coordinates and the subsequent filtering of the candidate regions by means of their confidence score via NMS. In the next section, a procedure is proposed aiming at reducing the inference time of the RPN, amongst others.

7.2 Search Area Reduction

Figure 7.5: Examples of DLR 3K comprising large areas without vehicles.

Restricting the search area to areas of interest, e.g., roads for vehicle detection, is an often applied preprocessing step in conventional object detection pipelines. In particular for vehicle detection in aerial imagery, both the computational costs for extracting feature descriptors as well as the subsequent classification and the number of false positive detections are reduced. While benchmark detection datasets typically comprise objects that are almost image filling, only a small fraction of aerial imagery datasets is occupied by objects, i.e., vehicles, due to their small dimensions in the range of only a few

pixels. For instance, images of PASCAL VOC 2007 are occupied by more than 50% with GT annotations, whereas in case of DLR 3K only about 1% of the images are covered by GT annotations. Therefore, the majority of the input image is no area of interest and is not relevant for predicting objects (see Figure 7.5). In case of real-world applications, this is often more distinct as the image fraction covered by vehicles often further decreases especially in rural areas that may contain no vehicle at all.

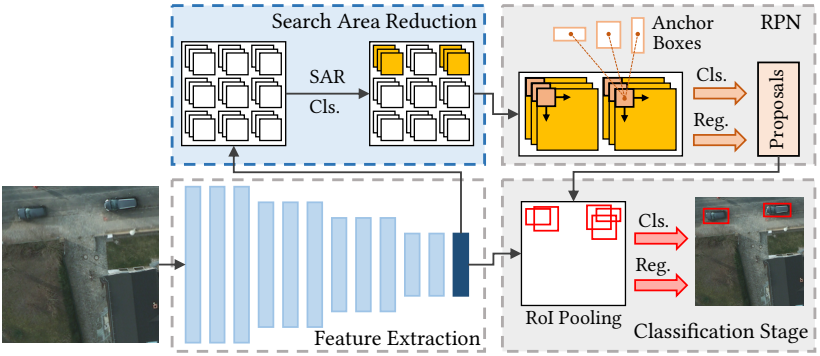


Figure 7.6: Schematic illustration of Faster R-CNN extended by an integrated Search Area Reduction (SAR) module to adaptively reduce the search area. The SAR module divides the input image into tiles and predicts a confidence score of how likely a tile contains at least one object. As only tiles that likely contain at least one object are forwarded to the components of the detection module, the inference time of the actual detection module is reduced.

As conventional object detectors generally compute features for each candidate window separately, the restriction only to areas of interest is straightforward to realize, e.g., by prior filtering out of non relevant candidate windows. In the context of this thesis, a novel Search Area Reduction (SAR) module is proposed to reduce the search area within deep learning based detection frameworks such as Faster R-CNN. The extension of Faster R-CNN by the proposed SAR module is schematically illustrated in Figure 7.6. In essence, the SAR module divides the employed feature map into tiles that correspond to certain areas of the input image and predicts a confidence score of how

likely a tile and consequently the respective area of the input image contains at least one object. Based on this confidence score, an adaptive set of tiles is forwarded to the components of the detection module, i.e., the RPN and classification stage. By filtering out tiles that do not contain any vehicles, the inference time of the actual detection module is reduced. In contrast to previous approaches, the SAR module is directly integrated into the detection pipeline. For this purpose, the SAR module and Faster R-CNN are realized within a single network by sharing the convolutional features. In the following, the SAR module and the implementation details are presented in detail. Ablation experiments are further conducted to demonstrate the effect of the proposed SAR module on the inference time.

7.2.1 Search Area Reduction Module

The main principle of the proposed approach to reduce the search area is the division of the input image into small tiles that are then classified into tiles containing at least one object or none. To realize this within a deep learning based detection framework, a SAR module comprised of three components is developed as depicted in Figure 7.7. Furthermore, the proposal layer is modified to map the generated region proposals that are given inside each tile to their position in the input image.

The first component divides the input image into equally sized tiles. For this purpose, a novel implemented layer called *sub-division-SAR layer* is introduced. Instead of dividing the input image directly into tiles and computing the convolutional features for each tile separately, the sub-division-SAR layer is applied on the output of the last convolutional layer, i.e., *conv5_3*. Thus, the computational effort is reduced, as the computation of convolutional features is shared between adjacent tiles. Note that adjacent tiles slightly overlap in order to avoid split objects at tile edges, which may result in misclassified objects. The convolutional features for each tile are cropped and reorganized in a batch by stacking the cropped features. The batch is then processed by the subsequent *SAR classifier*.

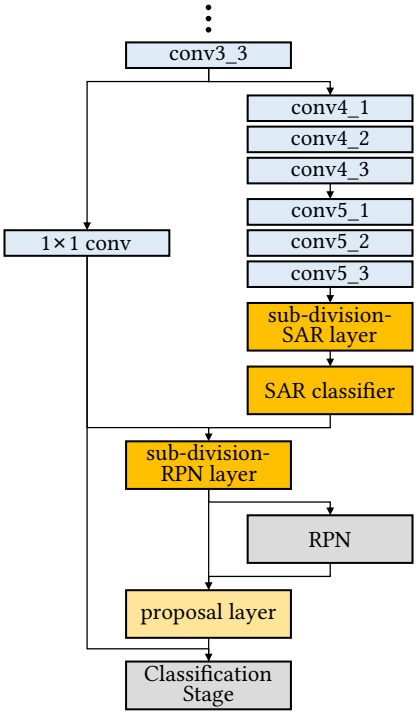


Figure 7.7: Schematic structure of the SAR module. Novel components introduced to perform the SAR are highlighted in dark orange. Furthermore, the proposal layer is modified to map the generated region proposals to their position in the input image.

The whole network employed for SAR classification is in essence the default VGG16 classifier comprised of five sequences of convolutional layers followed by three fully connected layers and a final softmax layer (see Table 5.1). The last fully connected layer comprising 1000 outputs trained for 1000-way ImageNet classification is replaced by a new fully connected layer with 2 outputs for the two classes. The subsequent softmax layer outputs the corresponding probability distribution. As described above, all convolutional features are computed at once for the entire image and the output of the last convolutional layer is divided into tiles. To classify each tile into tiles containing at least

one object or none, the sequence of fully connected layers as well as the softmax layer denoted as SAR classifier are applied on the corresponding features. Note that the fully connected layers require fixed input dimensions. Therefore, the cropped features are of size 14 × 14 × 512 (width × height × channels), which complies to the dimensions for input images of size 224 × 224 pixels as in case of the VGG16 classification network.

To restrict the search area, i.e., area processed by the detector components, a novel *sub-division-RPN layer* is applied on the output of *conv3_3*[1], which is used as feature map for the RPN. The sub-division-RPN layer divides the output of *conv3_3* into tiles of size 56 × 56 × 512. The offsets for cropping the tiles are set in a way so that the corresponding input image regions are equivalent for both the sub-division-RPN layer and sub-division-SAR layer. Besides the output of *conv3_3*, the sub-division-RPN layer takes as additional input the classification results of the SAR classifier. Based on the classification scores, tiles that are likely to contain at least one object are passed to the RPN, while all other tiles are filtered out. Note that the passed tiles are stacked together into one batch, so that the RPN can generate region proposals for all relevant tiles at once.

The original proposal layer computes the actual coordinates of each region proposal by adding the predicted offsets to the respective reference bounding box. As the reference bounding box is given in tile coordinates, the proposal layer of the RPN is modified in order to map the region proposals to their position in the input image. Therefore, a position vector generated by the sub-division-RPN layer is taken as an additional input. The position vector encodes the position of each tile that is forwarded from the sub-division-RPN with regard to its position in the input image. The actual coordinates of each region proposal are then computed by adding the offsets for the respective tile to its position given in tile coordinates. All mapped region proposals are then forwarded to the classification stage.

[1] To adapt the number of channels required for the fully connected layers, a 1 × 1 convolutional layer is applied on the output of *conv3_3* prior to the *sub-division-RPN layer*.

7.2.2 Implementation Details

To train the extended Faster R-CNN, a training strategy comprised of two stages is performed. Note that training both Faster R-CNN and the SAR classifier at once is not possible, as Faster R-CNN requires at least one GT object per training image, whereas the SAR classifier requires images with and without GT objects. In the first stage, Faster R-CNN is trained end-to-end for 60,000 iterations. For this, the training settings described in Section 5.1.3 are adopted. Then, the SAR classifier is trained in the second stage. To account for the input image dimensions of the default VGG16 classifier used as base architecture, the images of the DLR 3K dataset are split into sub-images of size 224 × 224 pixels. Adjacent sub-images exhibit an overlap of 46 pixels. The sub-images are divided into two categories: the first comprises all sub-images without any GT annotation and the second contains all sub-images with at least one GT annotation. For each sub-image, only GT annotations with an overlapping area of 10 or more pixels to the current sub-image are considered. Furthermore, data augmentation, i.e., horizontal and vertical flipping, is performed for each sub-image yielding a total of 19,200 training images, whereof 4,656 images belong to the category with at least one object. Example images of the training set are depicted in Figure 7.8.

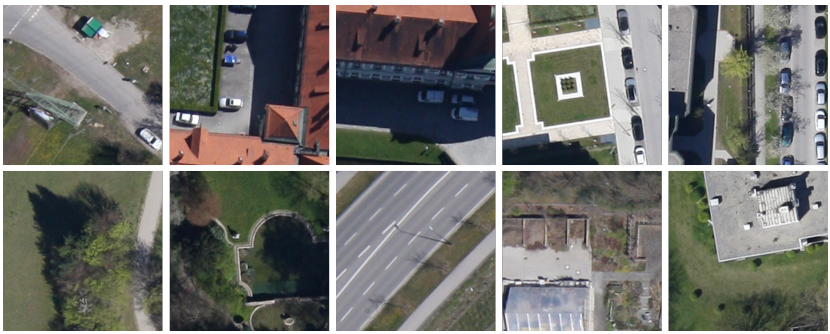


Figure 7.8: Example images of the training set used to train the SAR classifier for DLR 3K. The top row shows examples for the category with at least one object, while the bottom row shows examples for the category without objects.

The SAR classifier is trained for 10,000 iterations with a batch size of 8. The initial learning rate is set to 0.001 and decreased after 5,000 iterations by a factor of 10. Note that all weights of Faster R-CNN are kept fixed in the second training stage.

For deployment, the sub-division-SAR layer, sub-division-RPN layer, and the modified proposal layer are added to the framework to perform the search area reduction and the detection in a single network as depicted in Figure 7.7. For each tile, NMS is applied on the 400 region proposals exhibiting the highest confidence score. The top-80 ranked proposals after NMS are then forwarded to the classification stage. As the input images are divided into 25 tiles, the maximum number of proposals before and after NMS are 10,000 and 2,000, respectively, which is equivalent to the numbers for Faster R-CNN without the SAR module (see Section 5.1.3). The further settings are analogous to the settings employed for Faster R-CNN without the SAR module.

7.2.3 Ablation Experiments

The impact of the proposed SAR module on the inference time is evaluated on the DLR 3K dataset and the VEDAI LCI dataset in order to account for different area categories with varying vehicle distributions, as the speed-up in inference time depends on the number of filtered out image regions. In case of the DLR 3K test set, 36.4% of the image tiles include at least one object, since DLR 3K comprising mainly urban and residential areas exhibits a high traffic volume. In contrast, VEDAI comprises more rural and nature areas with a low traffic volume, so that only 8.2% of the image tiles in the VEDAI test set contain at least one object.

Comparison of the inference time for Faster R-CNN with and without the SAR module on the DLR 3K dataset is given in Table 7.16. For this, all timings are performed analogous to the timings in Section 5.2.4 on two different devices. The SAR module aims at restricting the search area and consequently reducing the inference time without worsening the detection accuracy. Preliminary experiments showed that the best trade-off between runtime and detection accuracy is achieved for a confidence score threshold of 0.5 used to accept tiles

for the RPN and the subsequent classification stage. Using higher threshold values results in false negative detections due to vehicles present in tiles that are filtered out. Thus, the threshold value is set to 0.5 in the following.

Table 7.16: Comparison of the inference time for Faster R-CNN with and without SAR on the DLR 3K dataset. The overall inference time and in particular the inference time for the RPN and the classification stage are reduced by applying the SAR module, while the AP remains unaffected.

Approach	Component	Time (in ms)	
		Server	Desktop
Faster R-CNN	Feature Extractor	58.9	177.4
	RPN	139.8	212.8
	Classification Stage	87.5	273.3
	Total	286.2	663.5
Faster R-CNN + SAR	Feature Extractor	78.6	284.2
	sub_SAR	4.9	5.1
	SAR	3.1	3.4
	sub_RPN	32.3	28.5
	RPN	92.2	159.8
	Classification Stage	15.2	75.1
	Total	226.3	556.1

The inference time for the RPN is reduced by approximately 34% and 25% for the server and desktop setup, respectively, due to the clearly reduced number of feature map locations used to predict candidate regions and the clearly reduced number of region proposals that have to be processed in the subsequent proposal layer. The inference time for the classification stage is even reduced by about 83% and 73%, respectively, as the number of candidate regions to classify is considerably reduced. However, the overall inference time is only reduced by 21% and 16%, respectively, because of the auxiliary components of the SAR module and the inherent computational costs for its feature extraction, which takes about 27% of the overall inference time. The additional

costs are mainly due to the cropping and reorganization of convolutional features in the sub-division-SAR and sub-division-RPN layer. Note that computational costs for the sub-division-RPN layer depend on the number of tiles considered for detection, as only the corresponding convolutional features are rearranged. Furthermore, additional costs depending on the number of tiles considered for detection result from the auxiliary computational operations in the proposal layer.

Table 7.17: Comparison of the inference time for Faster R-CNN with and without SAR on the VEDAI dataset. The overall inference time and in particular the inference time for the RPN and the classification stage are reduced by applying the SAR module.

Approach	Component	Time (in ms)	
		Server	Desktop
Faster R-CNN	Feature Extractor	66.8	214.1
	RPN	155.1	255.6
	Classification Stage	92.9	258.5
	Total	314.8	728.2
Faster R-CNN + SAR	Feature Extractor	93.1	331.5
	sub_SAR	5.4	5.5
	SAR	3.3	3.2
	sub_RPN	15.2	12.3
	RPN	29.3	50.2
	Classification Stage	8.7	22.5
	Total	155.0	425.2

The impact of the SAR module on the inference time for VEDAI is given in Table 7.17. The overall inference time is reduced by 51% and 42% on the server and desktop setup, respectively, while the AP only marginally decreases from 97.4% to 97.3%. Note that the feature extraction for VEDAI is more time-consuming due to the larger input image dimensions, i.e., 1024×1024 vs. 936×936 pixels in case of DLR 3K. The speed-up is more distinctive compared to DLR 3K due to the considerably lower traffic volume and consequently reduced number of tiles considered for the RPN and the classification stage.

In particular, the inference time for the RPN is notably reduced, i.e., by 81% and 80%, respectively.

Figure 7.9: Classification results of the SAR module on DLR 3K. Highlighted regions are classified as region that contain at least one object and are considered for detection. Regions labeled in blue contain vehicles, whereas regions labeled in red contain no vehicles.

Qualitative visualizations of the reduced search area are depicted in Figure 7.9 and Figure 7.10. Highlighted regions possess a confidence score about the presence of at least one vehicle above 0.5 and are considered for detection. Regions labeled in blue are correctly classified and contain at least one vehicle,

whereas regions labeled in red contain no vehicle. On both datasets, the SAR module is able to reliably identify tiles that contain at least one object even in case of housing areas with complex backgrounds and in case of areas off paved roads that typically contain no vehicles. Tiles without vehicles that are misclassified and accepted for detection generally comprise structures with shapes similar to vehicles.

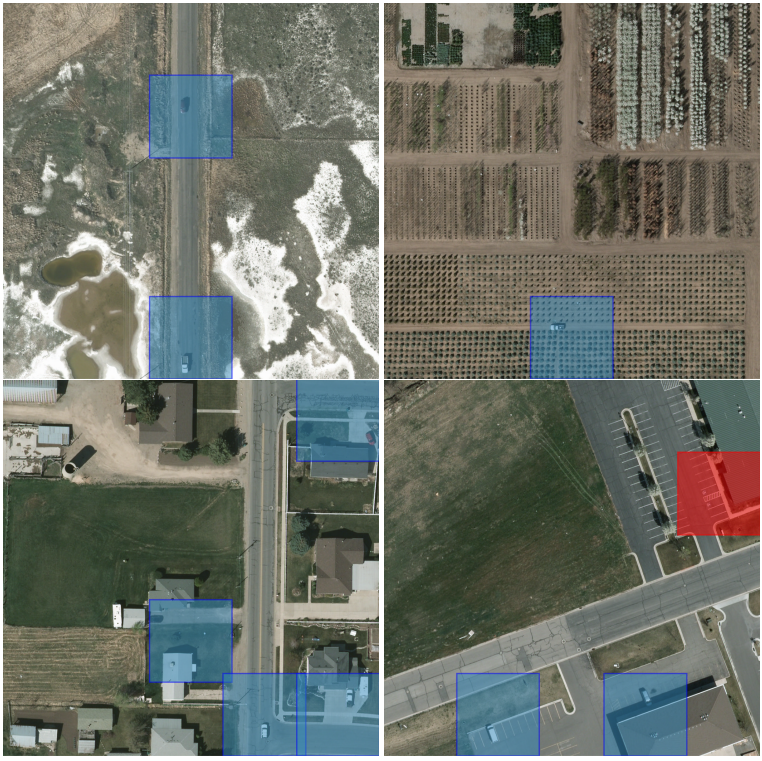


Figure 7.10: Classification results of the SAR module on VEDAI. Highlighted regions are classified as region that contain at least one object and are considered for detection. Regions labeled in blue contain vehicles, whereas regions labeled in red contain no vehicles.

The visualizations also indicate the potential as well as the limitation of the proposed SAR module to reduce the inference time. While in case of rural areas (see Figure 7.10) the majority of the tiles contain no vehicle and are filtered out, DLR 3K comprises images with high traffic volume across the entire scene, so that the search area is only marginally reduced (see Figure 7.9 bottom row). As the computational costs for the RPN and classification stage and consequently the inference time depend on the number of tiles without vehicles that are filtered out, the speed-up increases with fewer tiles considered for detection. Figure 7.11 shows the relation between speed-up compared to Faster R-CNN without SAR and the number of tiles filtered out exemplarily for DLR 3K on the server setup. For this, the number of tiles considered for detection is fixed for each image and not based on the confidence score about the presence of at least one vehicle. The timings are performed accordingly to Section 5.2.4. Due to the auxiliary computational costs of the SAR module, the removed search area has to comprise at least 10 tiles, which is 40% of the image, to reduce the inference time. However, in case of images that comprise only a few tiles that contain at least one vehicle, the inference time decreases considerably, e.g., more than 125 ms if only 20% of the image regions are considered for detection. Note that the time spent for the auxiliary components and the proposal layer may be further reduced by implementing these layers in CUDA to allow computation on a GPU.

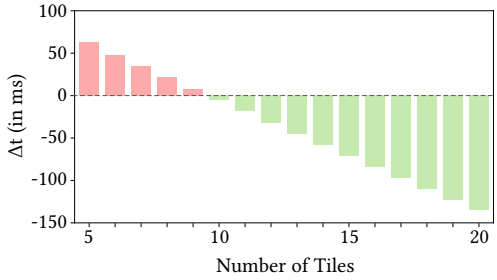


Figure 7.11: Speed-up in inference time between Faster R-CNN with and without SAR with respect to the number of tiles filtered out. Due to the additional computational costs of the SAR module, at least 10 tiles, which is 40% of the image, have to be filtered out to reduce the inference time.

8 Evaluation

In this chapter, the methods proposed within the context of this thesis are extensively evaluated following the evaluation protocol given in Section 4.2. First, different combinations of the methods proposed for improving the detection accuracy (see Chapter 6) and the methods proposed for runtime optimization (see Chapter 7) are examined in Section 8.1. Comparison of the combined methods to representative existing work in the literature with respect to detection accuracy and inference time is provided in Section 8.2. To demonstrate the generalization ability of the proposed methods, additional experiments are conducted in qualitative manner on different aerial imagery datasets in Section 8.3. Finally, a summary is given in Section 8.4.

8.1 Combined Methods for Improved Detection and Inference Time

As real-world applications generally require high detection accuracies at real-time or near real-time, the methods proposed for improving the detection accuracy by integrating contextual knowledge and the methods proposed for runtime optimization are combined in this section. For this, all experiments are conducted on the DLR 3K dataset. Note that data augmentation is performed for all trainings by applying vertical and horizontal flipping as well as rotation in steps of 90 degrees.

Combined Integration of Spatial and Semantic Context

As demonstrated in Section 6.1 and Section 6.2, integrating spatial and semantic context information results in improved detection accuracy. In the following, the effect of combining methods that either integrate spatial or semantic information are examined. To this end, MFD Faster R-CNN and EMT Faster R-CNN are merged into a single network as depicted in Figure 8.1. Instead of employing the outputs of *conv4_3* and *conv5_3* from the semantic labeling branch as auxiliary feature maps for the classification stage by extracting the corresponding features for each candidate region via RoI pooling, the CEM introduced in Section 6.1.1 is inserted to create a single high-resolution feature map. At first, the features from *conv5_3* are up-sampled and combined with the features from *conv4_3*. Then, the combined features are up-sampled and merged with the output from *conv3_3*. Thus, the semantically enriched features of deeper layers from the semantic labeling branch are propagated to the resulting feature map. Note that the proposed combination facilitates the use of the resulting feature map for both the RPN and classification stage, so that the candidate region generation is not only implicitly but also explicitly affected by the semantic labeling branch. The resulting network is trained end-to-end using the joint multi-task loss L_{MT} given in eq. (6.2). Weights pre-trained on ImageNet are used for initialization, while all further settings are adopted from Section 6.2.3.

Table 8.1: AP (in %) of the combined EMT-MFD Faster R-CNN for various GSDs. Compared to its individual components and the baseline Faster R-CNN, the detection accuracy is improved for all GSDs.

Base Network	GSD (in cm)		
	13	19.5	26
Faster R-CNN	95.0	91.9	86.0
MFD Faster R-CNN	95.9	93.2	88.5
EMT Faster R-CNN	96.2	93.3	88.9
EMT-MFD Faster R-CNN	**96.3**	**94.1**	**90.4**

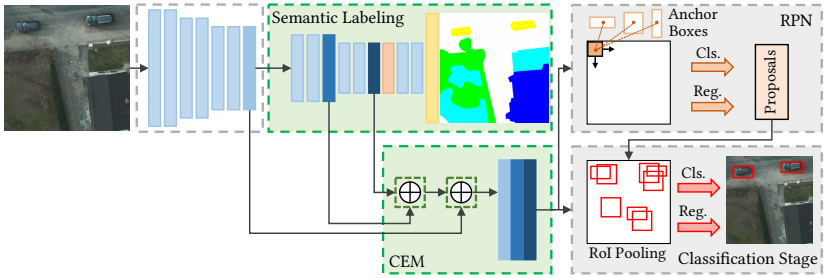


Figure 8.1: Schematic illustration of the merged MFD Faster R-CNN and EMT Faster R-CNN. Note that the outputs of *conv4_3* and *conv5_3* from the semantic labeling branch are used as inputs for the CEM to create a single high-resolution feature map.

Table 8.1 reports the detection accuracy of the merged MFD Faster R-CNN and EMT Faster R-CNN for various GSDs. Compared to its plain counterparts and Faster R-CNN, the detection accuracy is improved for all GSDs. Note that the AP for MFD Faster R-CNN is higher in comparison with the AP reported in Table 6.1 due to the performed data augmentation. While the gain in AP is minor for a GSD of 13 cm, the improvement becomes more notable with higher GSDs. For a GSD of 26 cm, MFD Faster R-CNN and EMT Faster R-CNN are outperformed by 1.9% and 1.5% in AP, respectively. This shows that inducing scene knowledge via semantic labeling yields more distinctive feature representations in deep layers as expected. Furthermore, it indicates that combining features from different layers via the CEM and the use of the resulting feature map for both stages are beneficial towards the simple integration of features from deep layers for classification via RoI pooling as carried out in EMT Faster R-CNN.

Integration of Semantic Context and Lightweight Feature Extraction

The effect of combining the integration of semantic context and the lightweight feature extraction is examined by replacing the default base network of EMT Faster R-CNN, i.e., VGG16, with the computation-efficient ZynqNet architecture. The outputs of the 7^{th} and 9^{th} Fire module are considered as auxiliary feature maps for the classification stage to explicitly

integrate features from the semantic labeling branch. Additional RoI pooling layers are applied to extract the corresponding features for each candidate region, which are fused with the respective output of the RoI pooling layer of the detection branch via element-wise addition as described in Section 6.2.4. $VGG_{0.5}$-Head is employed as classification head, as it exhibits the best trade-off between detection accuracy and inference time (see Table 7.11). Joint multi-task loss L_{MT} (see eq. (6.2)) and the settings specified in Section 7.1.5 are adopted for the training.

Table 8.2: Comparison of the detection accuracy and inference time for EMT Faster R-CNN with different base networks.

Base Network	AP (in %)	Time (in ms)	
		Server	Desktop
VGG16	**96.2**	317.2	824.5
$ZynqNet_{\alpha=1.00}$	96.0	172.5	264.2
$ZynqNet_{\alpha=0.75}$	95.1	156.0	225.6
$ZynqNet_{\alpha=0.50}$	94.1	**149.8**	**187.6**

The AP and inference times for EMT Faster R-CNN with $ZynqNet_{\alpha=1.00}$ as base network are given in Table 8.2. In comparison to EMT Faster R-CNN with VGG16, the inference time is reduced by 46% and 68% for the server and desktop setup, respectively, while the detection accuracy only marginally decreases. This demonstrates that even lightweight architectures are suited as base network for EMT Faster R-CNN and thus, allow improved detection accuracy by integration of semantic context with clearly decreased inference time compared to the original Faster R-CNN. Removing filters from the base network following the pruning strategy introduced in Section 7.1.3 further boosts the inference speed. However, the detection accuracy clearly drops with fewer filters per convolutional layer. Table 8.3 illustrates the applicability of EMT Faster R-CNN with $ZynqNet_{\alpha=1.00}$ for higher GSDs. Though the gap in AP compared to its counterpart using VGG16 as base network increases with higher GSD, the AP is still high for a GSD of 26 cm.

Table 8.3: AP (in %) for EMT Faster R-CNN using VGG16 and ZynqNet as base network with respect to various GSDs.

Base Network	GSD (in cm)		
	13	19.5	26
VGG16	**96.2**	**93.3**	**88.9**
ZynqNet$_{\alpha=1.00}$	96.0	92.7	87.8

Integration of Semantic Context and Search Area Reduction

Figure 8.2 depicts the integration of the SAR module proposed in Section 7.2 into EMT Faster R-CNN. To minimize the auxiliary computational costs for the SAR module, the convolutional layers are shared between the semantic labeling branch and the SAR module. Stage-wise training is performed according to Section 7.2.2, as Faster R-CNN requires at least one GT object per training image, whereas the SAR classifier requires images with and without GT objects. In the initial stage, EMT Faster R-CNN is trained using the settings specified in Section 6.2.3, while the SAR classifier is trained in the second stage as described in Section 7.2.2. Note that all convolutional layers are kept fixed during the second training stage. For inference, the further components of the SAR module, i.e., the sub-division-SAR layer and the sub-division-RPN layer, and the modified proposal layer are added to the framework. All settings are adopted from Section 7.2.2.

As shown in Table 8.4, the times spent for region proposal generation and the classification stage as well as the overall inference time are clearly reduced by restricting the search area. Note that the detection accuracy marginally drops by 0.1% in AP, as the number of false negatives increases due to incorrectly classified image areas. Compared to EMT Faster R-CNN with ZynqNet as base network, the detection accuracy is on par, while the gain in inference time is smaller because of the high traffic volume in the DLR 3K dataset.

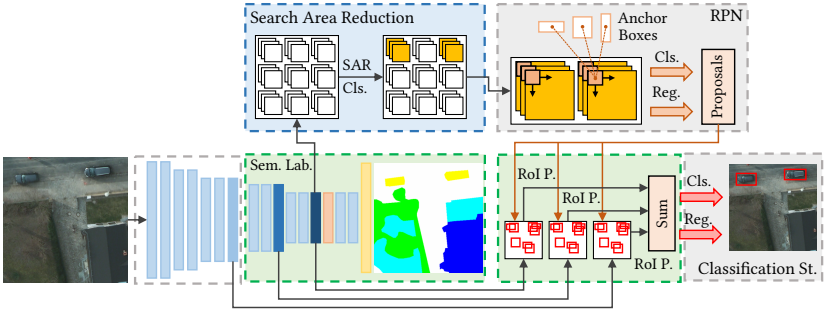


Figure 8.2: Schematic illustration of the EMT Faster R-CNN with SAR. Features from the semantic labeling branch are explicitly added for each region proposal by way of additional RoI pooling layers and element-wise addition.

Table 8.4: Comparison of the inference time for EMT Faster R-CNN with and without SAR. The overall inference time and in particular the inference time for the RPN and the classification stage are reduced by applying the SAR module.

Feature Map	Component	Time (in ms)	
		Server	Desktop
EMT Faster R-CNN	Feature Extractor	80.4	286.1
	RPN	135.0	209.7
	Classification Stage	101.8	328.7
	Total	317.2	824.5
EMT Faster R-CNN + SAR	Feature Extractor	81.5	286.0
	sub_SAR	5.0	4.8
	SAR	3.2	3.8
	sub_RPN	33.1	30.9
	RPN	93.7	161.1
	Classification Stage	21.7	95.3
	Total	238.2	581.9

Integration of Spatial Context and Lightweight Feature Extraction

To combine the integration of spatial context and the lightweight feature extraction, ZynqNet is applied as base network for MFD Faster R-CNN. The integration of spatial context from deeper layers is conducted in a stage-wise manner by applying the CEM introduced in Section 6.1.1. The output from the 9^{th} Fire module is up-sampled and combined with the output from the 7^{th} Fire module. The combined features are up-sampled and combined with the output from the 5^{th} Fire module. The resulting feature map enriched with features that comprise more contextual information is then used as input for the RPN and classification stage. Due to its good trade-off between detection accuracy and inference time in case of Faster R-CNN with ZynqNet as base network, $\text{VGG}_{0.5}$-Head is employed as classification head. For training, the staged fine-tuning scheme proposed in Section 6.1.2 is adopted.

The AP and inference times for MFD Faster R-CNN with $\text{ZynqNet}_{\alpha=1.00}$ as base network are given in Table 8.5. The inference time is significantly reduced on both setups by replacing the default base network with $\text{ZynqNet}_{\alpha=1.00}$, while the decrease in AP is small. Applying the pruning strategy introduced in Section 7.1.3 to remove redundant filters yields a further acceleration as expected. The drop in AP with fewer filters per convolutional layer is smaller compared to EMT Faster R-CNN with ZynqNet (see Table 8.2).

Table 8.5: Comparison of the detection accuracy and inference time for MFD Faster R-CNN with different base networks.

Base Network	AP (in %)	Time (in ms)	
		Server	Desktop
VGG16	**95.9**	316.3	815.2
$\text{ZynqNet}_{\alpha=1.00}$	95.8	167.8	283.6
$\text{ZynqNet}_{\alpha=0.75}$	95.3	162.5	233.3
$\text{ZynqNet}_{\alpha=0.50}$	94.5	**142.3**	**190.9**

Table 8.6: AP (in %) for MFD Faster R-CNN using VGG16 and ZynqNet as base network with respect to various GSDs.

Base Network	GSD (in cm)		
	13	19.5	26
VGG16	**95.9**	**93.2**	**88.5**
ZynqNet$_{\alpha=1.00}$	95.8	92.5	88.1

The applicability of MFD Faster R-CNN with ZynqNet$_{\alpha=1.00}$ for higher GSDs is demonstrated in Table 8.6. While the detection accuracy drops by 0.7% in AP compared to MFD Faster R-CNN with VGG16 for a GSD of 19.5 cm, the drop in AP is only 0.4% for a GSD of 26 cm.

Integration of Spatial Context and Search Area Reduction

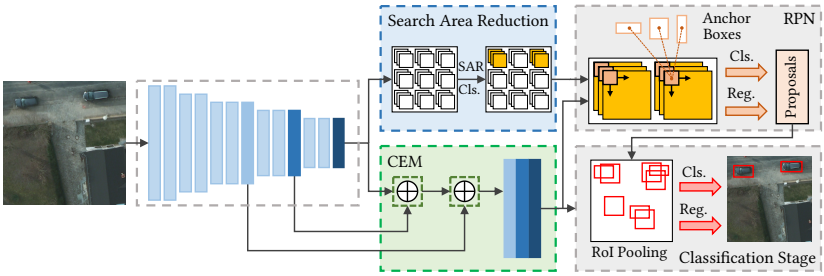


Figure 8.3: Schematic illustration of the MFD Faster R-CNN with SAR. Note that the combined feature map is divided into tiles and region proposals are only generated for tiles that are likely to contain at least one object based on the SAR classification.

The SAR module is added to the MFD Faster R-CNN as illustrated in Figure 8.3. The model is trained in a stage-wise manner. MFD Faster R-CNN is initially trained according to Section 6.1.2. In the second stage, the SAR classifier is trained as specified in Section 7.2.2, whereby all convolutional layers are kept fixed. For deployment, the further components of the SAR module and the

modified proposal layer are added to the framework and the settings stated in Section 7.2.2 are adopted.

Table 8.7: Comparison of the inference time for MFD Faster R-CNN with and without SAR. The overall inference time and in particular the inference time for the RPN and the classification stage are reduced by applying the SAR module.

Feature Map	Component	Time (in ms)	
		Server	Desktop
MFD Faster R-CNN	Feature Extractor	91.3	332.3
	RPN	134.6	213.6
	Classification Stage	90.4	269.3
	Total	316.3	815.2
MFD Faster R-CNN + SAR	Feature Extractor	92.8	334.0
	sub_SAR	5.7	5.5
	SAR	3.2	3.4
	sub_RPN	32.9	28.0
	RPN	92.2	154.4
	Classification Stage	21.9	95.3
	Total	248.7	620.6

The impact of the SAR module on the inference time is reported in Table 8.7. The time spent for the RPN and classification stage on the server setup are reduced by 32% and 76%, respectively, yielding an overall decrease of 21%, while the detection accuracy slightly drops by 0.1% in AP, which is on par with the detection accuracy achieved for MFD Faster R-CNN with a lightweight base network (see Table 8.5).

Lightweight Feature Extraction and Search Area Reduction

The usage of lightweight architectures as base network as well as the restriction of the search area exhibit an improved inference time. To further speed up the inference time, the SAR module is inserted into Faster R-CNN and

the default base network is replaced by $\text{ZynqNet}_{\alpha=1.00}$. To this end, the output from the 5$^{\text{th}}$ Fire module is considered as feature map for the RPN and classification stage. The auxiliary Fire modules, i.e., the 6$^{\text{th}}$ through the 9$^{\text{th}}$ Fire module, followed by max pooling, a sequence of fully connected layers and a softmax layer are employed as SAR classifier. For this, the sequence of fully connected layers is adopted according to Section 7.2.1. $\text{VGG}_{0.5}$-Head is employed as classification head, which exhibits the best trade-off between inference time and AP. The model is trained in two stages as described in Section 7.2.2. The detection part is first trained with the settings specified in Section 7.1.5, while the SAR classifier is trained as described in Section 7.2.2. The further components of the SAR module and the modified proposal layer are inserted for deployment as shown in Figure 7.7. Furthermore, the settings for deployment proposed in Section 7.2.2 are applied.

Table 8.8 shows the overall inference time and the time spent for each component of Faster R-CNN with SAR and lightweight feature extraction. Inserting the SAR module into Faster R-CNN with ZynqNet as base network achieves overall the best inference time. The usage of the lightweight base network results in clearly reduced time spent for feature extraction. Furthermore, the time spent for classification decreases due to the more lightweight classification head. Inserting the SAR module into Faster R-CNN with ZynqNet as base network results in clearly less time spent for generating region proposals and classification compared to its counterpart without SAR, while the detection accuracy only worsens by 0.1% in AP. In comparison to the baseline Faster R-CNN, the overall inference time is speeded up by 53% and 75% on the server and the desktop setup, respectively, as the times spent for each component are clearly reduced, whereas the detection accuracy only drops by 0.2% in AP. Note that all timings are repeated for the models pre-trained on augmented data and thus, the reported inference times may vary slightly compared to their counterparts without augmented data given in previous chapters. The most time-consuming component of Faster R-CNN with SAR and lightweight feature extraction is the RPN. A reason for this is the proposal layer implemented in Python that computes the final coordinates and sorts all proposals by the predicted confidence score on the CPU.

Table 8.8: Overall inference time and the time spent for each component of Faster R-CNN with SAR using ZynqNet as base network compared to the baseline Faster R-CNN and the separate approaches proposed to improve the inference time.

Approach	Base Network	Component	AP (in %)	Time (in ms) Server	Desktop
Faster R-CNN	VGG16	Feature Extractor	95.0	59.0	176.7
		RPN		140.2	214.1
		Classification Stage		87.7	273.6
		Total		286.9	664.4
Faster R-CNN	ZynqNet	Feature Extractor	94.9	21.8	39.0
		RPN		122.7	120.2
		Classification Stage		22.1	68.3
		Total		166.6	227.5
Faster R-CNN + SAR	VGG16	Feature Extractor	95.0	78.7	283.9
		sub_SAR		5.0	5.1
		SAR		3.1	3.5
		sub_RPN		32.4	28.2
		RPN		92.7	159.1
		Classification Stage		15.0	75.4
		Total		226.9	555.2
Faster R-CNN + SAR	ZynqNet	Feature Extractor	94.8	21.6	39.9
		sub_SAR		6.5	6.3
		SAR		2.9	3.2
		sub_RPN		24.9	27.0
		RPN		64.8	68.4
		Classification Stage		13.8	19.1
		Total		134.5	163.9

Combination of all proposed Approaches

Finally, all proposed approaches are merged into a single detector as visualized in Figure 3.1. For this, ZynqNet is used as base network. Note that all layers from the initial convolutional layer through the 9th Fire module are shared between the EMT Faster R-CNN, the MFD Faster R-CNN and the SAR classifier to minimize the computational costs. The output from the 9th Fire

module is up-sampled and combined with the output from the 7th Fire module. The combined features are up-sampled and combined with the output from the 5th Fire module. The resulting feature map is then used as input for the RPN and classification stage as aforementioned. The final model is trained in two stages. First, EMT Faster R-CNN and MFD Faster R-CNN are trained jointly using the proposed multi-task loss L_{MT} (see eq. (6.2)) and the settings specified in Section 7.1.5. In the second stage, the SAR classifier is trained as described in Section 7.2.2, while the layers shared with EMT Faster R-CNN and MFD Faster R-CNN are kept fixed. For deployment, the further components of the SAR module are inserted analogous to Figure 7.7.

Table 8.9: Detection accuracy and inference times for the final model that combines all proposed approaches.

Approach	Base Network	AP (in %)	Time (in ms) Server	Desktop
Faster R-CNN	VGG16	95.0	286.9	664.4
EMT-MFD Faster R-CNN	VGG16	**96.3**	317.0	818.3
Faster R-CNN + SAR	ZynqNet	94.8	**134.5**	**163.9**
EMT-MFD Faster R-CNN + SAR	VGG16	**96.3**	248.1	619.3
EMT-MFD Faster R-CNN + SAR	ZynqNet	96.1	136.9	218.8

The detection accuracy and the inference times for the final model are reported in Table 8.9. In comparison with the baseline Faster R-CNN, the detection accuracy is improved by 1.1% in AP, while the inference time is reduced by 52% and 67% on the server and desktop setup, respectively. The improved detection accuracy is mainly due to the reduced number of FPs caused by vehicle-like structures (see Figure 8.4). Compared to other combinations of the proposed approaches, the final model exhibits the best trade-off between detection accuracy and inference time. The inference time is considerably reduced compared to the joint EMT-MFD Faster R-CNN with and without SAR, while the detection accuracy only drops by 0.2% in AP. On the other hand,

the detection accuracy is improved by 1.3% in AP compared to Faster R-CNN with SAR and ZynqNet as base network. The additional computational costs caused by the modules proposed to integrate spatial and semantic information result in an only marginally increased inference time on the server setup, whereas the inference time increases by roughly 33% on the desktop setup.

Figure 8.4: Qualitative detection results (red boxes) and corresponding GT (green boxes) for Faster R-CNN with VGG16 (left column), EMT-MFD Faster R-CNN + SAR with VGG16 (middle column) and EMT-MFD Faster R-CNN + SAR with ZynqNet (right column) on DLR 3K demonstrate that the proposed approaches are more robust to false alarms due to vehicle-like structures.

8.2 Comparison to Related Work

The combined EMT-MFD Faster R-CNN with SAR is in the following compared to representative existing work in the field of object detection in aerial imagery. As most of these approaches adopt deep learning based detection frameworks for the task of vehicle detection in aerial imagery, recently proposed deep learning based detection frameworks are further considered for the comparison. The detection performance for EMT-MFD Faster R-CNN with SAR and the detection methods from literature are given in Table 8.10. The considered vehicle detection methods either adapt the size of the employed feature map [Car17, Aca18, Din18], exploit multiple feature maps [Guo18, Tay18], combine features from different layers [Den17, Din18, Yan18] or make use of a top-down architecture [Aca18, Guo18, Tay18] to account for the characteristics of aerial imagery. Note that the detection methods proposed for vehicle detection in aerial imagery are adopted unmodified. All the additionally used deep learning based detection frameworks have a top-down architecture and exploit multiple feature maps except for Faster R-CNN with OHEM. For each detection framework, the shallowest feature map has been selected in such a way that its dimensions are 1/4 of the input image in order to account for small-sized vehicles. Furthermore, the anchor box scales are adopted according to Table 5.6. To ensure a fair comparison, each model is trained on the identical training data with the same data augmentation settings.

Amongst the vehicle detection methods, the best AP on DLR 3K is achieved for DYOLO and Adapted RetinaNet, which both comprise a top-down architecture similar to MFD Faster R-CNN in order to obtain feature maps enriched with more context information. Shallow YOLOv2, AVPN Faster R-CNN and DFL-CNN exhibit considerably worse AP compared to the other vehicle detection methods. Though both Shallow YOLOv2 and DFL-CNN specifically adapt the resolution of the employed feature map, the resulting resolution, i.e., 1/16 of the input image, which is the same for AVPN Faster R-CNN, is not sufficient to accurately detect small-sized vehicles as in case of DLR 3K. Overall, none of the examined vehicle detection methods accomplishes an AP that is on par with the AP achieved for EMT-MFD Faster R-CNN with SAR.

One reason for this is the coarser feature map resolutions applied throughout the different methods, i.e., 1/8 or 1/16 of the input image, which may result in poorly located as well as duplicate detections (see Section 5.2.1). However, it is to mention that some of these approaches are designed for object detection in aerial imagery that comprise larger object instances such as NWPU.

Table 8.10: Average precision and inference time for EMT-MFD Faster R-CNN + SAR with VGG16 and ZynNet, respectively, compared to representative existing work on the DLR 3K dataset.

Method	Base Network	AP (in %)	Time (in FPS)
Shallow YOLOv2 [Car17][1]	Darknet-19	73.4	6.4
Adapted YOLOv2 [Aca18][1]	Darknet-19	94.5	11.5
DYOLO [Aca18][1]	Darknet-19	94.9	5.8
AVPN Faster R-CNN [Den17][2]	ZF	69.2	12.4
Multi-Scale CNN [Guo18][2]	VGG16	93.7	4.4
Modified Faster R-CNN [Din18][2]	VGG16	93.6	10.8
DFL-CNN [Yan18][2]	ResNet50	85.1	6.6
Adapted RetinaNet [Tay18][3]	ResNet50	95.0	**14.8**
Faster R-CNN + OHEM [Shr16b][2]	VGG16	95.2	3.5
FPN [Lin17a][3]	ResNet50	95.4	13.4
FPN - DCNv1 [Dai17][3]	ResNet50	95.3	12.2
FPN - DCNv2 [Zhu19][3]	ResNet50	95.4	12.2
Cascade R-CNN [Cai18][3]	ResNet50	95.7	10.5
Libra R-CNN [Pan19][3]	ResNet50	95.6	12.7
YOLOv3 [Red18][1]	Darknet-53	96.1	14.2
DSSD [Fu17][2]	ResNet101	94.7	2.9
RefineDet [Zha18a][2]	VGG16	96.1	6.7
EMT-MFD Faster R-CNN + SAR	VGG16	**96.3**	4.0
EMT-MFD Faster R-CNN + SAR	ZynqNet	96.1	7.3

[1] Using the Darknet framework (https://github.com/pjreddie/darknet)
[2] Using the Caffe framework
[3] Using the MMDetection [Che19b] toolbox based on PyTorch (https://github.com/open-mmlab/mmdetection)

Faster R-CNN with OHEM outperforms the baseline Faster R-CNN by 0.2% in AP (see Table 8.9), which demonstrates the advantage of hard negative mining to learn more robust feature representations. However, the detection accuracy is generally worse compared to the adapted deep learning based detection frameworks that make use of a top-down architecture. Hence, enhancing the context information by combining features of shallow and deep layers possesses a larger impact on the detection accuracy. FPN, which extends Faster R-CNN by a top-down path that combines features from different layers, outperforms the baseline Faster R-CNN by 0.4% in AP. Inserting different variants of deformable convolutions (FPN - DCNv1 and FPN - DCNv2) results in no gain in AP. While Libra R-CNN that aims at reducing the imbalance at sample, feature, and objective level only slightly improves the detection accuracy of the FPN, applying a cascaded training scheme (Cascade R-CNN) in order to increase the localization accuracy outperforms the FPN by 0.3% in AP. The best AP amongst the recently proposed deep learning based detection frameworks is achieved for RefineDet and YOLOv3, which is almost on par with the proposed detection methods. In comparison with the examined vehicle detection methods, the recently proposed deep learning frameworks generally exhibit higher AP values. This shows that the conducted adaptations proposed in Section 5.2 are transferable to more recent detection frameworks and are essential to achieve state-of-the-art detection accuracies in case of tiny objects.

The best inference time is achieved for Adapted RetinaNet followed by YOLOv3 and FPN. Both RetinaNet and YOLOv3 perform detection in a single stage, which is in general less computationally expensive than two-stage approaches. A reason for the high number of FPS in case of FPN, although FPN is an extended version of Faster R-CNN that comprises more computational operations, is the more efficient implementation of the proposal generation step. While the Caffe implementation performs the proposal generation on the CPU, the MMDetection [Che19b] toolbox implementation runs completely on the GPU. Hence, implementing the components proposed in this thesis with MMDetection in future work is an opportunity to further speed up the inference time of EMT-MFD Faster R-CNN with SAR. AVPN Faster R-CNN achieves the best inference time amongst the methods implemented

in Caffe, which is mainly due to the coarse feature map resolution and consequently, the reduced number of region proposals that have to be processed. However, employing such a coarse feature map resolution is not practicable as discussed above. The best inference time for methods implemented in Caffe, which exhibit a fine feature map resolution, is achieved for the proposed detection method. Even single-stage approaches are outperformed, which demonstrates the benefits of the proposed components.

Comparison of EMT-MFD Faster R-CNN with SAR to existing work from literature for various GSDs on DLR 3K is given in Table 8.11. For this, Adapted RetinaNet, which showed the best AP for a GSD of 13 cm amongst the vehicle detection methods, and the adapted detection frameworks with the highest AP are considered. Note that the anchor box scales are adopted according to Table 5.6. EMT-MFD Faster R-CNN with SAR using VGG16 as base network exhibits the best detection accuracy for all GSDs followed by RefineDet and YOLOv3. Using ZynqNet as base network yields slightly worse AP values for higher GSDs, though ZynqNet comprises considerably fewer parameters compared to the other employed base networks. Adapted RetinaNet exhibits the largest decrease in AP with higher GSDs, which verifies the importance of fine feature map resolutions especially in case of tiny objects.

Table 8.11: AP (in %) for EMT-MFD Faster R-CNN + SAR with VGG16 and ZynqNet, respectively, compared to representative existing work on the DLR 3K dataset for various GSDs.

Method	Base Network	GSD (in cm)		
		13	19.5	26
Adapted RetinaNet[Tay18]	ResNet50	95.0	92.2	84.1
FPN [Lin17a]	ResNet50	95.4	93.0	88.6
Cascade R-CNN [Cai18]	ResNet50	95.7	93.4	89.0
YOLOv3 [Red18]	Darknet-53	96.1	93.9	89.7
RefineDet [Zha18a]	VGG16	96.1	94.0	90.0
EMT-MFD Faster R-CNN + SAR	VGG16	**96.3**	**94.1**	**90.3**
EMT-MFD Faster R-CNN + SAR	ZynqNet	96.1	93.6	89.3

Qualitative detection examples (red boxes) and corresponding GT (green boxes) on DLR 3K with a GSD of 26 cm are visualized in Figure 8.5. While all detection methods show a good localization accuracy, applying the proposed detection method results in fewer false alarms caused by vehicle-like structures on buildings. However, for all methods, remaining false alarms are mainly due to objects located on asphalted areas with vehicle-like shapes.

To demonstrate the transferability of the detection methods proposed in this thesis, MFD Faster R-CNN with SAR is compared to existing work from literature on VEDAI LCI and VEDAI SCI (see Table 8.12). Note that exploitation of semantic information is not conducted because of the missing semantic labeling annotations. Each model is trained with the same data augmentation settings, i.e., vertical and horizontal flipping as well as rotation in steps of 90 degrees. The proposed MFD Faster R-CNN with SAR and VGG16 as base network achieves the best AP on both versions of the dataset. Using ZynqNet as base network exhibits the same detection accuracy on VEDAI LCI, while the detection accuracy slightly drops in case of the higher GSD as observed for DLR 3K as well. Amongst the adapted detection frameworks, RefineDet shows overall the best detection accuracy. Similar to DLR 3K, Adapted RetinaNet exhibits the poorest detection accuracies due to the coarse feature map resolution. Qualitative detection examples (red boxes) and corresponding GT (green boxes) on VEDAI SCI are visualized in Figure 8.6.

Table 8.12: AP (in %) for MFD Faster R-CNN + SAR with VGG16 and ZynqNet, respectively, compared to representative existing work on the VEDAI dataset.

Method	Base Network	VEDAI LCI	VEDAI SCI
Adapted RetinaNet[Tay18]	ResNet50	95.5	89.6
FPN [Lin17a]	ResNet50	97.1	93.2
Cascade R-CNN [Cai18]	ResNet50	97.1	93.3
YOLOv3 [Red18]	Darknet-53	96.7	92.7
RefineDet [Zha18a]	VGG16	97.2	93.8
MFD Faster R-CNN + SAR	VGG16	**97.7**	**94.3**
MFD Faster R-CNN + SAR	ZynqNet	**97.7**	94.0

Figure 8.5: Qualitative detection results (red boxes) and corresponding GT (green boxes) for EMT-MFD Faster R-CNN + SAR with VGG16 and ZynqNet, respectively, and representative existing work on DLR 3K with a GSD of 26 cm.

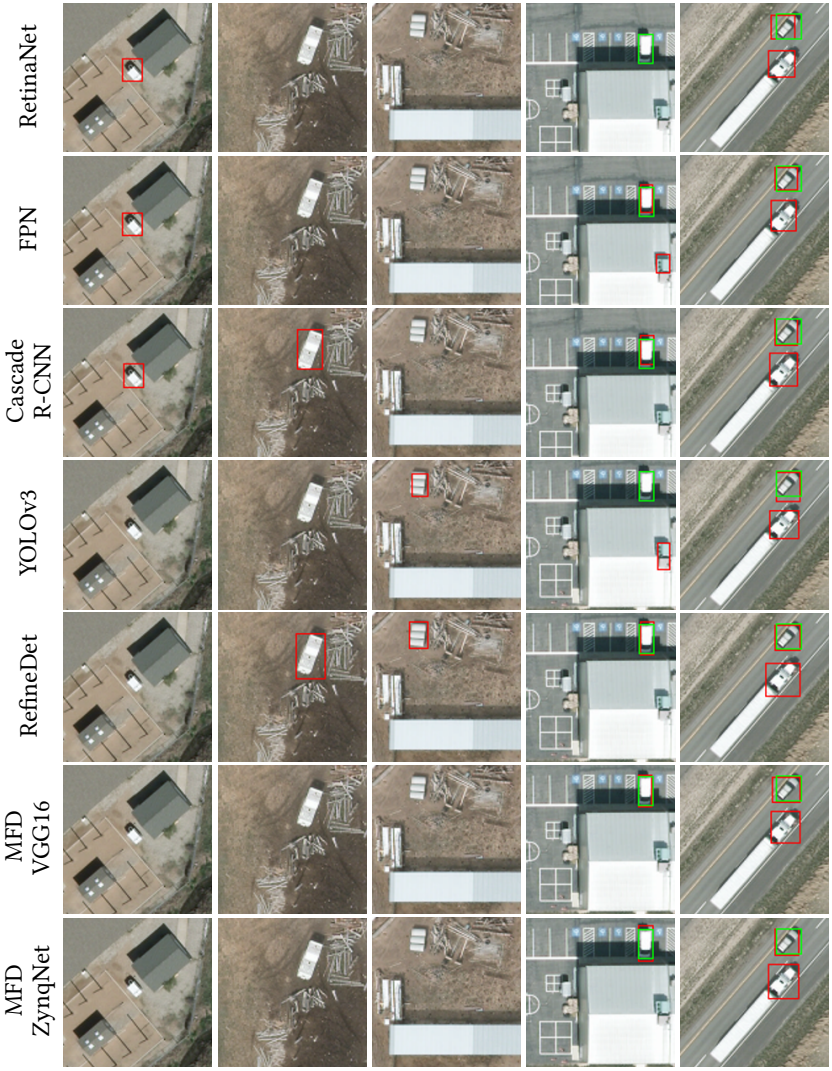


Figure 8.6: Qualitative detection results (red boxes) and corresponding GT (green boxes) for EMT-MFD Faster R-CNN + SAR with VGG16 and ZynqNet, respectively, and representative existing work on VEDAI SCI.

8.3 Generalization to Unseen Aerial Imagery

In the following, the generalization ability of the proposed detection method is demonstrated by auxiliary experiments on three recently published aerial imagery datasets, i.e., ITCVD, DOTA and xView (see Section 4.1). Note that models pre-trained on DLR 3K are employed for all experiments.

ITCVD

Amongst the three datasets mentioned above, ITCVD is most similar to DLR 3K in terms of image quality and content. Both datasets are acquired over Western European cities and thus, mainly comprise urban and residential areas with comparable structures and objects. Figure 8.7 shows qualitative detection results (red boxes) and corresponding GT (green boxes) for EMT-MFD Faster R-CNN with VGG16 and EMT-MFD Faster R-CNN with ZynqNet on images from the ITCVD test set, whereby images taken in oblique view with a tilt angle of 45 degrees are not considered. During deployment, the test images are down-scaled by a factor of 1.3, so that the GSD is similar to DLR 3K. The proposed methods achieve a good classification and localization accuracy even in case of a weak contrast between vehicle and background. Note that even multiple vehicles are correctly detected, which are missed during the annotation process especially in backyards and entrance areas. Hence, the proposed methods are well transferable to unseen data that comprise similar characteristics compared to the employed training data, i.e., the DLR 3K dataset. As depicted in Figure 8.8, EMT-MFD Faster R-CNN with VGG16 and ZynqNet, respectively, exhibit considerably fewer false alarms caused by vehicle-like structures on buildings compared to Faster R-CNN with VGG16 taken as baseline. Though ITCVD comprises strong parallax effects due to the low acquisition altitude, the proposed methods are robust to false alarms caused by the accompanying disturbing structures that are hardly or non-existent in the training data. However, remaining false positive detections are caused by vehicle-like structures on asphalted areas, whereas missed detections are mainly due to partial occlusion, e.g., by trees, or due to vehicles located in shadowed areas (see Figure 8.9), which is similar to observations on DLR 3K.

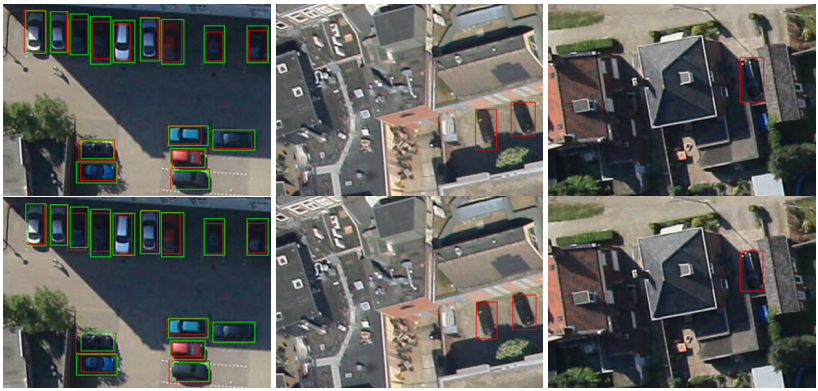


Figure 8.7: Qualitative detection results (red boxes) and corresponding GT (green boxes) for EMT-MFD Faster R-CNN with VGG16 (top row) and with ZynqNet (bottom row) on the ITCVD dataset show the good detection accuracy. Note that even multiple vehicles with missing annotations are correctly detected.

Figure 8.8: Qualitative detection results (red boxes) and corresponding GT (green boxes) for Faster R-CNN with VGG16 (left column), EMT-MFD Faster R-CNN with VGG16 (middle column) and EMT-MFD Faster R-CNN with ZynqNet (right column) on the ITCVD dataset demonstrate that the proposed approaches are more robust to false alarms due to vehicle-like structures in unseen data, while vehicles are correctly detected.

Figure 8.9: Qualitative examples of missed detections and false alarms for EMT-MFD Faster R-CNN with VGG16 (top row) and ZynqNet (bottom row) on the ITCVD dataset. Missed detections are mainly due to objects that are partially occluded, e.g., by trees, or that are located in shadowed areas, while false alarms are mostly caused by structures located on asphalted areas.

DOTA

In contrast to DLR 3K, the DOTA dataset comprises images from multiple sensor platforms and consequently, exhibits varying image qualities, differing scenarios and a larger variety of object appearances. Figure 8.10 depicts qualitative detection results (red boxes) and corresponding GT (green boxes) for EMT-MFD Faster R-CNN with VGG16 and EMT-MFD Faster R-CNN with ZynqNet on images from the DOTA validation set. For this, only GT annotations for vehicle categories, i.e., small vehicles and large vehicles, are visualized, whereas further categories such as *baseball diamond*, *harbor*, *bridge*, etc. are not considered. Note that all images are scaled during deployment in order to exhibit a similar GSD as in case of DLR 3K. Both EMT-MFD Faster R-CNN with VGG16 and ZynqNet, respectively, exhibit a good localization and classification accuracy, which indicates the good transferability in case of differing scenarios and poor image quality. Furthermore, issues regarding the provided GT, i.e., poorly aligned bounding boxes and missing annotations, are illustrated, which impedes the validity of a quantitative analysis. Compared to Faster R-CNN with VGG16, applying the proposed methods results in clearly fewer false alarms caused by vehicle-like structures on buildings (see Figure 8.11), which confirms observations on the ITCVD dataset.

Figure 8.10: Qualitative detection results (red boxes) and corresponding GT (green boxes) for EMT-MFD Faster R-CNN with VGG16 (top row) and with ZynqNet (bottom row) on the DOTA dataset indicate the good detection accuracy in unseen images from different sensors. Note that even multiple vehicles with missing annotations are correctly detected.

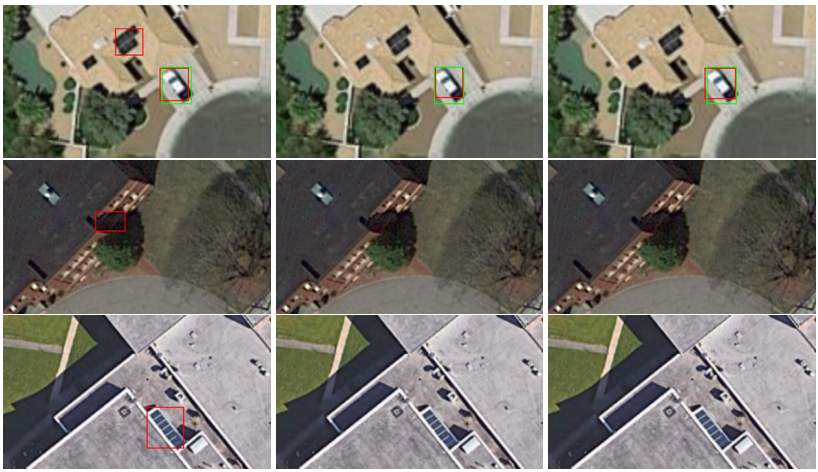


Figure 8.11: Qualitative detection results (red boxes) and corresponding GT (green boxes) for Faster R-CNN with VGG16 (left column), EMT-MFD Faster R-CNN with VGG16 (middle column) and EMT-MFD Faster R-CNN with ZynqNet (right column) on the DOTA dataset demonstrate that the proposed approaches are more robust to false alarms due to vehicle-like structures in unseen data.

Figure 8.12 shows the main reasons for remaining false alarms and missed detections. While, similar to results on the DLR 3K dataset, missed detections are mainly due to objects that are partially occluded, e.g., by trees, false alarms are mostly caused by shadows or structures located on asphalted areas. Additional false alarms stem from split detections caused by vehicle types that are not represented in the training data.

Figure 8.12: Qualitative examples of missed detections and false alarms for EMT-MFD Faster R-CNN with VGG16 (top row) and ZynqNet (bottom row) on the DOTA dataset. Missed detections are mainly due to objects that are partially occluded, e.g., by trees, while false alarms are mostly caused by shadows or structures located on asphalted areas. Furthermore, false alarms are caused by split detections due to vehicle types that are not represented in the training data.

xView

The xView dataset comprises images acquired over different continents and thus, exhibits a larger variety of scenarios and objects compared to DLR 3K. Qualitative detection results (red boxes) and corresponding GT (green boxes) for EMT-MFD Faster R-CNN with VGG16 and ZynqNet on the xView dataset are depicted in Figure 8.13. For this, only GT annotations for categories belonging to the meta class *passenger vehicle* are visualized. In contrast to the experiments performed on ITCVD and DOTA, models pre-trained on DLR 3K

down-scaled by a factor of 2 are employed due to the low GSD of approximately 30 cm. Both EMT-MFD Faster R-CNN with VGG16 and EMT-MFD Faster R-CNN with ZynqNet exhibit almost no false alarms even in scenes with complex backgrounds, while detections are accurately aligned around occurring vehicles. However, multiple vehicles are not detected because of the relatively large differences to the training data, in particular the poorer image quality and larger variety of occurring objects. This indicates that the generalization ability gets worse in case of higher GSDs and consequently smaller object dimensions.

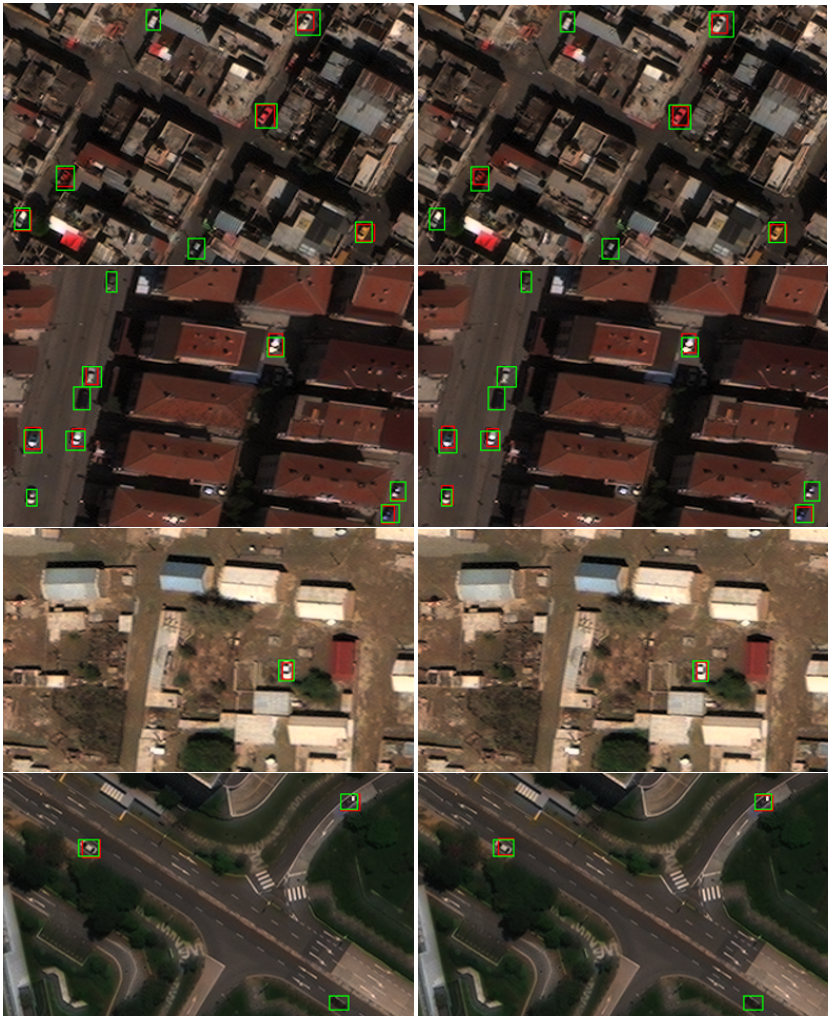


Figure 8.13: Qualitative detection results (red boxes) and corresponding GT (green boxes) for EMT-MFD Faster R-CNN with VGG16 (left column) and with ZynqNet (right column) on the xView dataset. Despite the low spatial resolution, the proposed approaches are robust to false alarms due to vehicle-like structures in unseen data with differing scenarios and backgrounds. However, multiple missed detections occur due to the relatively large differences to the training data.

8.4 Summary

Finally, it is worth discussing the strengths and weaknesses of the proposed components and the examined combinations with respect to real-world applications. The CEM introduced to enhance the spatial context information of the employed features is straightforward to integrate into deep learning frameworks, as essentially the feature extraction is altered. Its benefits to the feature representation are demonstrated in the performed experiments by means of clearly reduced false alarms caused by vehicle-like structures in unlikely areas, which yields an improved detection accuracy. The auxiliary computational costs on the other hand only slightly increase the inference time, which may be tolerable for most applications in combination with techniques to optimize the inference time. Implementations similar to the CEM are a major part of the most recent deep learning based detectors that achieve state-of-the-art results in various domains. This further emphasizes the importance of such a component. Inducing scene knowledge via semantic labeling improves the feature representation as well. The detection accuracy is clearly improved, as the number of false alarms caused by vehicle-like structures in unlikely areas is reduced. However, in contrast to the CEM, the training requires semantic labeling annotations, whose generation is time-consuming. Thus, semantic labeling annotations are often not available and the training is restricted to a few aerial imagery datasets. Consequently, the applicability of the semantic labeling based approach may be limited to images similar to these datasets. Since architectures for semantic labeling in aerial imagery are typically developed for low GSDs, novel architectures have to be explored to better account for extremely fine structures as in case of GSDs above 20 cm. Otherwise, the potential of the semantic content information may be not fully exploited. Using computation-efficient CNN architectures considerably reduced the time spent for feature extraction without large drops in detection accuracy, which is essential for applications that have to run in real-time or near real-time. As training deep learning based detection frameworks with such CNN architectures does not depend on specific datasets, its applicability is not restricted. However, the drop in detection accuracy compared to heavyweight CNN architectures increases with higher GSDs and thus, more

complex scenarios due to the smaller object dimensions. As one reason for this is the clearly reduced number of parameters used for feature representation, the computation-efficient CNN architectures could be modified to address this issue. The proposed module to restrict the search area to areas of interest results in a large speed-up of the region proposal generation and the classification stage, while the detection accuracy remains almost unaffected. Prerequisite for this speed-up is that only a small fraction of an aerial image is occupied by relevant objects and thus, most image regions are not considered for detection. Therefore, the proposed module is less appropriate for aerial imagery recorded over urban areas with dense traffic volumes. Since a classifier is trained on the respective data to identify areas that are unlikely to contain a vehicle, the generalization ability is further affected by the quality of the classifier, which may limit its applicability to unseen data. A data independent alternative is the integration of referenced road maps to identify areas of interest, whereby road maps are not always provided and vehicles offside roads are missed.

While each component already outperforms the baseline Faster R-CNN either in terms of detection accuracy or inference time, combining the proposed components further boosts the detection performance. Combining both alternative approaches to enhance the feature representation shows an additional gain in detection accuracy, especially in case of more complex scenarios such as higher GSDs. Hence, this combination is particularly of interest for applications relying on very accurate detection. As the computation-efficient CNN architecture and the SAR module reduce the computational costs for different stages of the detection pipeline, their combination further speeds up the overall inference time. Thus, this combination is better suited than the individual approaches for applications with harsher time constraints. All combinations of components to enhance the detection accuracy and components to decrease the inference time yield an improved trade-off between inference time and detection accuracy compared to the individual components, so that these combinations are good alternatives to fulfill time and accuracy constraints. The best trade-off between inference time and detection accuracy is achieved for integrating all components into the detection pipeline, whereby aforemtioned

limitations may restrict its applicability. Nevertheless, the stand-alone character of the proposed components allows the usage of themselves or in different combinations in order to meet the specific requirements of an application.

In summary, the proposed detection pipeline comprised of the components to enhance the detection accuracy and inference time outperforms representative existing work from literature on different aerial imagery. Furthermore, a good generalization ability is demonstrated on unseen data with differing scenarios, which is essential for most real-world applications. Compared to the baseline Faster R-CNN especially the number of false alarms is considerably reduced due to the more robust feature representation.

9 Conclusions and Outlook

9.1 Conclusions

In this thesis, a novel deep learning based detection pipeline is proposed for the task of vehicle detection in aerial imagery. For this, Faster R-CNN, which is chosen as base framework, is systematically adapted with respect to the specific characteristics of aerial imagery. Especially increasing the resolution of the feature map employed for region proposal generation and classification clearly improves the detection accuracy, as localization issues in case of small-sized objects are solved. Despite the improved detection accuracy, the performed adaptations yield several shortcomings, i.e., low semantic and spatial content of the employed features and a poor inference time, which impede the usage in real-world applications.

Two novel approaches are proposed to overcome the lack of semantic and spatial content and thus, reduce the number of false alarms caused by objects with vehicle-like shapes. The first approach enhances the spatial context information by combining features from different layers to account for fine and coarse structures, while maintaining a high feature map resolution. For this purpose, Faster R-CNN is extended by the proposed CEM, which utilizes deconvolutional layers to up-sample features of deep layers. The latter approach leverages semantic labeling to increase the semantic context information, whereby two different variants to integrate semantic labeling into the detection framework are realized. Inducing scene knowledge by explicitly merging the semantic labeling network into the detection framework via

shared feature representations outperforms the alternative variant that exploits the semantic labeling results to filter out unlikely predictions. The proposed approaches exhibit clearly improved detection results, in particular for high GSDs and consequently smaller object dimensions. The reason for the improved detection results is the reduced number of false alarms caused by vehicle-like structures located on regions that are unlikely to contain vehicles, e.g., buildings.

In order to reduce the computational effort and consequently, the inference time, two different strategies are pursued in this thesis. The first strategy aims at optimizing the time spent for feature extraction by replacing the default CNN architecture with a lightweight CNN architecture. In combination with further techniques for runtime optimization, i.e., filter pruning, merged batch normalization and exploitation of computation-efficient classification heads, the inference time is considerably reduced, while the detection accuracy remains almost unaffected. The second strategy restricts the search area to areas of interest by identifying and removing areas that are unlikely to contain at least one vehicle. For this, a novel module to classify image areas is explicitly integrated into the detection framework by sharing the convolutional features. As vehicles generally cover only a small fraction of aerial imagery, the computational efforts for the region proposal stage and the classification stage and consequently the inference time are clearly reduced.

To ensure high detection accuracies at real-time or near real-time, the proposed approaches and strategies are combined into a single detection pipeline. The standard Faster R-CNN detector taken as baseline is significantly improved in terms of detection accuracy and inference time. Furthermore, the proposed method outperforms representative existing work from literature on different aerial imagery datasets. Finally, the generalization ability of the proposed method is demonstrated by auxiliary experiments on unseen data with differing scenarios.

9.2 Outlook

Although the proposed method exhibits good results regarding the detection accuracy of vehicles in aerial imagery, further enhancements and extensions are often necessary to meet the requirements of real-world applications.

Though the components proposed to reduce the computational effort result in a large speed-up compared to the baseline Faster R-CNN, the inference time of the entire detection pipeline may not be sufficient for some applications. The main bottleneck is an inefficient proposal generation, which can be addressed by transferring the corresponding processing steps to the GPU, as implemented in novel frameworks like the MMDetection toolbox based on PyTorch [Che19b]. Integrating the components proposed in this thesis into single-stage detection frameworks offers an alternative to overcome issues with regard to the proposal generation. A promising way to decrease the overall inference time is making use of the NVIDIA TensorRT[1] library, which facilitates high-performance inference of different deep learning frameworks such as Caffe. By combining layers and optimizing kernel selection for improved latency, throughput, power efficiency and memory consumption, TensorRT optimizes the inference time of a given pre-trained network. Moreover, TensorRT offers out-of-the-box INT8 quantization and FP16 precision implementations of common layers as further options to accelerate the inference time.

So far, the proposed detection pipeline is limited to a single vehicle class in aerial imagery recorded in top-down view due to the low availability of annotated training data. However, distinguishing between vehicle classes, e.g., car and truck, is highly relevant for applications such as traffic monitoring and management, while an accurate detection in images recorded in oblique view is often prerequisite for applications like disaster relief and search and rescue tasks. As deep learning is largely data-driven, the trend of getting bigger and more extensive datasets, emerging in different computer vision domains including object detection in aerial imagery, facilitates the learning

[1] https://developer.nvidia.com/tensorrt

of more complex tasks. The VisDrone dataset [Zhu18] for instance comprises more than 2.6 million annotations for different object categories, e.g., pedestrian, car, truck, etc., in images recorded by UAVs with different perspectives. Besides the new extension options, novel challenges arise that have to be addressed. Especially imbalanced data that may yield biased rules in favor of the majority class and the large variety of object scales ranging from a few to hundreds of pixels complicate the detection task. Furthermore, recent developments in the field of object detection in aerial imagery focus on the transition from axis-aligned bounding boxes to oriented bounding boxes, which is especially of interest for tracking based applications that depend on orientation information. For this, most components of the base detection pipeline, e.g., proposal generation, RoI pooling, NMS, etc., and the objective function have to be extended regarding the orientation.

In general, training data is limited in its diversity to particular areas and recording conditions, which impairs the generality and transferability of learned models. Though the proposed detection pipeline exhibits good detection results on unseen data with differing scenarios, limitations of its transferability become obviously recognizable especially in case of high GSDs, poor image quality and large variety of occurring objects. To overcome challenges of cross-domain differences, domain adaptation aims at transferring knowledge learned by a particular network on a source domain to a new related target domain. Common techniques attempt to match the distributions of the source and target datasets by minimizing some divergence criterion or make use of generative adversarial networks (GANs) to generate synthetic target data which are somehow related to the source domain. Few-shot learning – the ability to learn from only few labeled samples – is another promising research direction to address these issues, which may allow re-training on target data on the flight.

To improve the detection accuracy, exploiting temporal context across consecutive frames has recently drawn increasing attention in the computer vision community. Besides established approaches to integrate temporal context like recurrent neural networks (RNNs), applying 3 dimensional convolutions is a popular strategy to learn discriminative features along both spatial

and temporal dimensions. Since the data recorded for most applications based on aerial imagery generally comprises image sequences, such approaches are promising to improve the detection accuracy in aerial imagery, in particular in case of tiny or partially occluded objects. While data-driven object detection in aerial imagery has been limited to single images in the past, recent datasets such as the VisDrone dataset and the UAVDT dataset [Du18] providing annotations for image sequences allow the usage of multiple consecutive frames.

Bibliography

[Aca18] Acatay, Oliver; Sommer, Lars; Schumann, Arne and Beyerer, Jürgen: "Comprehensive Evaluation of Deep Learning based Detection Methods for Vehicle Detection in Aerial Imagery". In: *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. 2018.

[Ada11] Adams, Stuart M and Friedland, Carol J: "A survey of unmanned aerial vehicle (UAV) usage for imagery collection in disaster research and management". In: *9th International Workshop on Remote Sensing for Disaster Response*. Vol. 8. 2011.

[Amm17] Ammour, Nassim; Alhichri, Haikel; Bazi, Yakoub; Benjdira, Bilel; Alajlan, Naif and Zuair, Mansour: "Deep learning approach for car detection in UAV imagery". In: *Remote Sensing* 9.4 (2017).

[Ang03] Angel, Alejandro; Hickman, Mark; Mirchandani, Pitu and Chandnani, Dinesh: "Methods of analyzing traffic imagery collected from aerial platforms". In: *IEEE Transactions on Intelligent Transportation Systems* 4.2 (2003), pp. 99–107.

[Ani17] Anisimov, Dmitriy and Khanova, Tatiana: "Towards lightweight convolutional neural networks for object detection". In: *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. 2017.

[Ash17] Ashraf, Khalid; Wu, Bichen; Iandola, Forrest N; Moskewicz, Mattthew W and Keutzer, Kurt: "Shallow networks for high-accuracy road object-detection". In: *International Conference on*

Vehicle Technology and Intelligent Transport Systems (VEHITS) (2017).

[Aud17] Audebert, Nicolas; Le Saux, Bertrand and Lefèvre, Sébastien: "Segment-before-detect: Vehicle detection and classification through semantic segmentation of aerial images". In: *Remote Sensing* 9.4 (2017).

[Azi18] Azimi, Seyed Majid; Vig, Eleonora; Bahmanyar, Reza; Körner, Marco and Reinartz, Peter: "Towards multi-class object detection in unconstrained remote sensing imagery". In: *Proceedings of the Asian Conference on Computer Vision (ACCV)*. Springer. 2018, pp. 150–165.

[Azi19] Azimi, Seyed Majid; Henry, Corentin; Sommer, Lars; Schumann, Arne and Vig, Eleonora: "SkyScapes Fine-Grained Semantic Understanding of Aerial Scenes". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2019.

[Bad17] Badrinarayanan, Vijay; Kendall, Alex and Cipolla, Roberto: "Segnet: A deep convolutional encoder-decoder architecture for image segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.12 (2017), pp. 2481–2495.

[Bao19] Bao, Songze; Zhong, Xing; Zhu, Ruifei; Zhang, Xiaonan; Li, Zhuqiang and Li, Mengyang: "Single Shot Anchor Refinement Network for Oriented Object Detection in Optical Remote Sensing Imagery". In: *IEEE Access* 7 (2019), pp. 87150–87161.

[Bel16] Bell, Sean; Lawrence Zitnick, C; Bala, Kavita and Girshick, Ross: "Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 2874–2883.

[Ber16] Berrahal, Sarra; Kim, Jong-Hoon; Rekhis, Slim; Boudriga, Noureddine; Wilkins, Deon and Acevedo, Jaime: "Border surveillance monitoring using quadcopter UAV-aided wireless sensor networks". In: *Journal of Communications Software and Systems* 12 (2016).

[Bra17] Brahmbhatt, Samarth; Christensen, Henrik I and Hays, James: "StuffNet: Using 'Stuff' to Improve Object Detection". In: *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2017, pp. 934–943.

[Bur97] Burlina, P; Parameswaran, V and Chellappa, R: "Sensitivity analysis and learning strategies for context-based vehicle detection algorithms". In: *Proceedings DARPA Image Understanding Workshop*. 1997, pp. 577–584.

[Cai16] Cai, Zhaowei; Fan, Quanfu; Feris, Rogerio S and Vasconcelos, Nuno: "A unified multi-scale deep convolutional neural network for fast object detection". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer. 2016, pp. 354–370.

[Cai18] Cai, Zhaowei and Vasconcelos, Nuno: "Cascade r-cnn: Delving into high quality object detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 6154–6162.

[Car17] Carlet, Jennifer and Abayowa, Bernard: "Fast vehicle detection in aerial imagery". In: *arXiv preprint arXiv:1709.08666* (2017).

[Che13] Chen, Xueyun; Xiang, Shiming; Liu, Cheng-Lin and Pan, Chun-Hong: "Vehicle detection in satellite images by parallel deep convolutional neural networks". In: *Proceedings of the IEEE IAPR Asian Conference on Pattern Recognition (ACPR)*. 2013, pp. 181–185.

[Che14a] Chen, Xueyun; Xiang, Shiming; Liu, Cheng-Lin and Pan, Chun-Hong: "Vehicle detection in satellite images by hybrid deep convolutional neural networks". In: *IEEE Geoscience and Remote Sensing Letters* 11.10 (2014), pp. 1797–1801.

[Che14b] Cheng, Gong; Han, Junwei; Zhou, Peicheng and Guo, Lei: "Multi-class geospatial object detection and geographic image classification based on collection of part detectors". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 98 (2014), pp. 119–132.

[Che14c] Cheng, Ming-Ming; Zhang, Ziming; Lin, Wen-Yan and Torr, Philip: "BING: Binarized normed gradients for objectness estimation at 300fps". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014, pp. 3286–3293.

[Che16a] Cheng, Gong and Han, Junwei: "A survey on object detection in optical remote sensing images". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 117 (2016), pp. 11–28.

[Che16b] Cheng, Gong; Zhou, Peicheng and Han, Junwei: "Learning rotation-invariant convolutional neural networks for object detection in VHR optical remote sensing images". In: *IEEE Transactions on Geoscience and Remote Sensing* 54.12 (2016), pp. 7405–7415.

[Che19a] Chen, Changrui; Zhang, Yu; Lv, Qingxuan; Wei, Shuo; Wang, Xiaorui; Sun, Xin and Dong, Junyu: "RRNet: A Hybrid Detector for Object Detection in Drone-Captured Images". In: *Proceedings of the IEEE International Conference on Computer Vision Workshops (ICCVW)*. 2019.

[Che19b] Chen, Kai; Wang, Jiaqi; Pang, Jiangmiao; Cao, Yuhang; Xiong, Yu; Li, Xiaoxiao; Sun, Shuyang; Feng, Wansen; Liu, Ziwei; Xu, Jiarui, et al.: "MMDetection: Open MMLab Detection Toolbox and Benchmark". In: *arXiv preprint arXiv:1906.07155* (2019).

[Cho09] Choi, Jae-Young and Yang, Young-Kyu: "Vehicle detection from aerial images using local shape information". In: *Pacific-Rim Symposium on Image and Video Technology*. Springer. 2009, pp. 227–236.

[Cho17] Chollet, François: "Xception: Deep learning with depthwise separable convolutions". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 1251–1258.

[Dai16] Dai, Jifeng; Li, Yi; He, Kaiming and Sun, Jian: "R-fcn: Object detection via region-based fully convolutional networks". In: *Advances in Neural Information Processing Systems (NIPS)*. 2016, pp. 379–387.

[Dai17] Dai, Jifeng; Qi, Haozhi; Xiong, Yuwen; Li, Yi; Zhang, Guodong; Hu, Han and Wei, Yichen: "Deformable convolutional networks". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 764–773.

[Dem18] Demir, Ilke; Koperski, Krzysztof; Lindenbaum, David; Pang, Guan; Huang, Jing; Basu, Saikat; Hughes, Forest; Tuia, Devis and Raska, Ramesh: "Deepglobe 2018: A challenge to parse the earth through satellite images". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2018, pp. 172–181.

[Den09] Deng, Jia; Dong, Wei; Socher, Richard; Li, Li-Jia; Li, Kai and Fei-Fei, Li: "Imagenet: A large-scale hierarchical image database". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2009, pp. 248–255.

[Den17] Deng, Zhipeng; Sun, Hao; Zhou, Shilin; Zhao, Juanping and Zou, Huanxin: "Toward fast and accurate vehicle detection in aerial images using coupled region-based convolutional neural networks". In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 10.8 (2017), pp. 3652–3664.

[Den18] Deng, Zhipeng; Sun, Hao; Zhou, Shilin; Zhao, Juanping; Lei, Lin and Zou, Huanxin: "Multi-scale object detection in remote sensing imagery with convolutional neural networks". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 145 (2018), pp. 3–22.

[Din18] Ding, Peng; Zhang, Ye; Deng, Wei-Jian; Jia, Ping and Kuijper, Arjan: "A light and faster regional convolutional neural network for object detection in optical remote sensing images". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 141 (2018), pp. 208–218.

[Din19] Ding, Jian; Xue, Nan; Long, Yang; Xia, Gui-Song and Lu, Qikai: "Learning RoI Transformer for Oriented Object Detection in Aerial Images". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 2849–2858.

[Du18] Du, Dawei; Qi, Yuankai; Yu, Hongyang; Yang, Yifan; Duan, Kaiwen; Li, Guorong; Zhang, Weigang; Huang, Qingming and Tian, Qi: "The unmanned aerial vehicle benchmark: Object detection and tracking". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 370–386.

[Duc11] Duchi, John; Hazan, Elad and Singer, Yoram: "Adaptive subgradient methods for online learning and stochastic optimization". In: *Journal of Machine Learning Research* 12.Jul (2011), pp. 2121–2159.

[Dud12] Duda, Richard O; Hart, Peter E and Stork, David G: Pattern classification. John Wiley & Sons, 2012.

[Egg17] Eggert, Christian; Brehm, Stephan; Winschel, Anton; Zecha, Dan and Lienhart, Rainer: "A closer look: Small object detection in faster R-CNN". In: *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*. 2017, pp. 421–426.

[Eik09] Eikvil, Line; Aurdal, Lars and Koren, Hans: "Classification-based vehicle detection in high-resolution satellite images". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 64.1 (2009), pp. 65–72.

[Erd16] Erdelj, Milan and Natalizio, Enrico: "UAV-assisted disaster management: Applications and open issues". In: *Proceedings of the IEEE International Conference on Computing, Networking and Communications (ICNC)*. 2016.

[Eve10] Everingham, Mark; Van Gool, Luc; Williams, Christopher KI; Winn, John and Zisserman, Andrew: "The pascal visual object classes (voc) challenge". In: *International Journal of Computer Vision* 88.2 (2010), pp. 303–338.

[Eye18] Eyerman, J.; Crispino, G.; Zamarro, A. and Durscher, R.: "Drone Efficacy Study (DES): Evaluating the Impact of Drones for Locating Lost Persons in Search and Rescue Events". In: *DJI and European Emergency Number Association* (2018).

[Eze14] Ezequiel, Carlos Alphonso F; Cua, Matthew; Libatique, Nathaniel C; Tangonan, Gregory L; Alampay, Raphael; Labuguen, Rollyn T; Favila, Chrisandro M; Honrado, Jaime Luis E; Canos, Vinni; Devaney, Charles, et al.: "UAV aerial imaging applications for post-disaster assessment, environmental management and infrastructure development". In: *Proceedings of the IEEE International Conference on Unmanned Aircraft Systems (ICUAS)*. 2014, pp. 274–283.

[Fel04] Felzenszwalb, Pedro F and Huttenlocher, Daniel P: "Efficient graph-based image segmentation". In: *International Journal of Computer Vision* 59.2 (2004), pp. 167–181.

[Fel10] Felzenszwalb, Pedro F; Girshick, Ross B; McAllester, David and Ramanan, Deva: "Object detection with discriminatively trained part-based models". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.9 (2010), pp. 1627–1645.

[Fu17] Fu, Cheng-Yang; Liu, Wei; Ranga, Ananth; Tyagi, Ambrish and Berg, Alexander C: "DSSD: Deconvolutional single shot detector". In: *arXiv preprint arXiv:1701.06659* (2017).

[Gir04] Girard, Anouck R; Howell, Adam S and Hedrick, J Karl: "Border patrol and surveillance missions using multiple unmanned air vehicles". In: *Proceedings of the IEEE Conference on Decision and Control (CDC)*. 2004, pp. 620–625.

[Gir14] Girshick, Ross; Donahue, Jeff; Darrell, Trevor and Malik, Jitendra: "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014, pp. 580–587.

[Gir15] Girshick, Ross: "Fast r-cnn". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1440–1448.

[Gir16] Girshick, Ross; Donahue, Jeff; Darrell, Trevor and Malik, Jitendra: "Region-based convolutional networks for accurate object detection and segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.1 (2016), pp. 142–158.

[Gle11] Gleason, Joshua; Nefian, Ara V; Bouyssounousse, Xavier; Fong, Terry and Bebis, George: "Vehicle detection from aerial imagery". In: *Proceedings of the IEEE International Conference on Robotics and Automation*. 2011, pp. 2065–2070.

[Glo10] Glorot, Xavier and Bengio, Yoshua: "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*. 2010, pp. 249–256.

[Glo11] Glorot, Xavier; Bordes, Antoine and Bengio, Yoshua: "Deep sparse rectifier neural networks". In: *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*. 2011, pp. 315–323.

[Goo09] Goodrich, Michael A; Morse, Bryan S; Engh, Cameron; Cooper, Joseph L and Adams, Julie A: "Towards using unmanned aerial vehicles (UAVs) in wilderness search and rescue: Lessons from field trials". In: *Interaction Studies* 10.3 (2009), pp. 453–478.

[Gra08] Grabner, Helmut; Nguyen, Thuy Thi; Gruber, Barbara and Bischof, Horst: "On-line boosting-based car detection from aerial images". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 63.3 (2008), pp. 382–396.

[GRS18] GRSS: 2018 IEEE GRSS Data Fusion Challenge Dataset. http://hyperspectral.ee.uh.edu/?page_id=1075. [Online; accessed 01-October-2019]. 2018.

[Gsc16] Gschwend, David: "Zynqnet: An fpga-accelerated embedded convolutional neural network". In: *Master ETH-Zurich: Swiss Federal Institute of Technology Zurich* (2016).

[Guo18] Guo, Wei; Yang, Wen; Zhang, Haijian and Hua, Guang: "Geospatial object detection in high resolution satellite images based on multi-scale convolutional neural network". In: *Remote Sensing* 10.1 (2018).

[Ham17] Hamilton, Carl; Hughes, S; Perkins, D and Roberts, P: "Exercise Northumberland Research Report". In: (2017).

[Ham18] Hamaguchi, Ryuhei; Fujita, Aito; Nemoto, Keisuke; Imaizumi, Tomoyuki and Hikosaka, Shuhei: "Effective use of dilated convolutions for segmenting small object instances in remote sensing imagery". In: *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2018, pp. 1442–1450.

[Han17a] Han, Xiaobing; Zhong, Yanfei and Zhang, Liangpei: "An efficient and robust integrated geospatial object detection framework for high spatial resolution remote sensing imagery". In: *Remote Sensing* 9.7 (2017).

[Han17b] Han, Zhongxing; Zhang, Hui; Zhang, Jinfang and Hu, Xiaohui: "Fast aircraft detection based on region locating network in large-scale remote sensing images". In: *Proceedings of the IEEE International Conference on Image Processing ICIP*. 2017.

[Han18] Han, Junwei; Zhang, Dingwen; Cheng, Gong; Liu, Nian and Xu, Dong: "Advanced deep-learning techniques for salient and category-specific object detection: a survey". In: *IEEE Signal Processing Magazine* 35.1 (2018), pp. 84–100.

[He14] He, Kaiming; Zhang, Xiangyu; Ren, Shaoqing and Sun, Jian: "Spatial pyramid pooling in deep convolutional networks for visual recognition". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer. 2014, pp. 346–361.

[He15] He, Kaiming; Zhang, Xiangyu; Ren, Shaoqing and Sun, Jian: "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1026–1034.

[He16a] He, Kaiming; Zhang, Xiangyu; Ren, Shaoqing and Sun, Jian: "Deep residual learning for image recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778.

[He16b] He, Kaiming; Zhang, Xiangyu; Ren, Shaoqing and Sun, Jian: "Identity mappings in deep residual networks". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer. 2016, pp. 630–645.

[He17] He, Kaiming; Gkioxari, Georgia; Dollár, Piotr and Girshick, Ross: "Mask r-cnn". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2980–2988.

[Hei08] Heitz, Geremy and Koller, Daphne: "Learning spatial context: Using stuff to find things". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer. 2008, pp. 30–43.

[Hei10] Heinze, Norbert; Esswein, Martin; Krüger, Wolfgang and Saur, Günter: "Image exploitation algorithms for reconnaissance and surveillance with UAV". In: *Airborne Intelligence, Surveillance, Reconnaissance (ISR) Systems and Applications VII*. Vol. 7668. International Society for Optics and Photonics. 2010.

[Hin01] Hinz, Stefan and Baumgartner, Albert: "Vehicle detection in aerial images using generic features, grouping, and context". In: *Joint Pattern Recognition Symposium*. Springer. 2001, pp. 45–52.

[Hin03] Hinz, S; Schlosser, C and Reitberger, J: "Automatic car detection in high resolution urban scenes based on an adaptive 3D-model". In: *2003 2nd GRSS/ISPRS Joint Workshop on Remote Sensing and Data Fusion over Urban Areas*. IEEE. 2003, pp. 167–171.

[Hoi12] Hoiem, Derek; Chodpathumwan, Yodsawalai and Dai, Qieyun: "Diagnosing error in object detectors". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer. 2012, pp. 340–353.

[Hos15] Hosang, Jan; Benenson, Rodrigo; Dollár, Piotr and Schiele, Bernt: "What makes for effective detection proposals?" In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.4 (2015), pp. 814–830.

[How17] Howard, Andrew G; Zhu, Menglong; Chen, Bo; Kalenichenko, Dmitry; Wang, Weijun; Weyand, Tobias; Andreetto, Marco and Adam, Hartwig: "Mobilenets: Efficient convolutional neural networks for mobile vision applications". In: *arXiv preprint arXiv:1704.04861* (2017).

[Hu18] Hu, Jie; Shen, Li and Sun, Gang: "Squeeze-and-Excitation Networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2018), pp. 7132–7141.

[Hua17] Huang, Gao; Liu, Zhuang; Van Der Maaten, Laurens and Weinberger, Kilian Q: "Densely connected convolutional networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 4700–4708.

[Ian16] Iandola, Forrest N; Han, Song; Moskewicz, Matthew W; Ashraf, Khalid; Dally, William J and Keutzer, Kurt: "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size". In: *arXiv preprint arXiv:1602.07360* (2016).

[Iof15] Ioffe, Sergey and Szegedy, Christian: "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *arXiv preprint arXiv:1502.03167* (2015).

[ISP] ISPRS: ISPRS WG III/4. ISPRS 2D Semantic Labeling Contest Dataset. http://www2.isprs.org/commissions/comm3/wg4/semantic-labeling.html. [Online; accessed 30-March-2017].

[Jia15] Jiang, Qiling; Cao, Liujuan; Cheng, Ming; Wang, Cheng and Li, Jonathan: "Deep neural networks-based vehicle detection in satellite images". In: *Proceedings of the IEEE International Symposium on Bioelectronics and Bioinformatics (ISBB)*. 2015, pp. 184–187.

[Kan15] Kanistras, Konstantinos; Martins, Goncalo; Rutherford, Matthew J and Valavanis, Kimon P: "Survey of unmanned aerial vehicles (UAVs) for traffic monitoring". In: *Handbook of unmanned aerial vehicles* (2015), pp. 2643–2666.

[Kem10] Kembhavi, Aniruddha; Harwood, David and Davis, Larry S: "Vehicle detection using partial least squares". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33.6 (2010), pp. 1250–1265.

[Kim03] Kim, ZuWhan and Malik, Jitendra: "Fast vehicle detection with probabilistic feature grouping and its application to vehicle tracking". In: *IEEE*. 2003, p. 524.

[Kin14] Kingma, Diederik P and Ba, Jimmy: "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[Klu07] Kluckner, Stefan; Pacher, Georg; Grabner, Helmut; Bischof, Horst and Bauer, Joachim: "A 3D teacher for car detection in aerial images". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2007.

[Kog18] Koga, Yohei; Miyazaki, Hiroyuki and Shibasaki, Ryosuke: "A CNN-based method of vehicle detection from aerial images using hard example mining". In: *Remote Sensing* 10.1 (2018).

[Kon16] Kong, Tao; Yao, Anbang; Chen, Yurong and Sun, Fuchun: "HyperNet: Towards Accurate Region Proposal Generation and Joint Object Detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 845–853.

[Kri12] Krizhevsky, Alex; Sutskever, Ilya and Hinton, Geoffrey E: "Imagenet classification with deep convolutional neural networks". In: *Advances in Neural Information Processing Systems (NIPS)*. 2012, pp. 1097–1105.

[Kum01] Kumar, Rakesh; Sawhney, Harpreet; Samarasekera, Supun; Hsu, Steve; Tao, Hai; Guo, Yanlin; Hanna, Keith; Pope, Arthur; Wildes, Richard; Hirvonen, David, et al.: "Aerial video surveillance and exploitation". In: *Proceedings of the IEEE* 89.10 (2001), pp. 1518–1539.

[Kur18] Kurz, F; Waigand, D; Pekezou-Fouopi, P; Vig, E; Henry, C; Merkle, N; Rosenbaum, D; Gstaiger, V; Azimi, S; Auer, S, et al.: "DLRAD - A first look on the new vision and mapping benchmark dataset for autonomous driving". In: *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences* 42 (2018).

[Lam18] Lam, Darius; Kuzma, Richard; McGee, Kevin; Dooley, Samuel; Laielli, Michael; Klaric, Matthew; Bulatov, Yaroslav and McCord, Brendan: "xview: Objects in context in overhead imagery". In: *arXiv preprint arXiv:1802.07856* (2018).

[LeC15] LeCun, Yann; Bengio, Yoshua and Hinton, Geoffrey: "Deep learning". In: *Nature* 521.7553 (2015), p. 436.

[LeC98] LeCun, Yann; Bottou, Léon; Bengio, Yoshua and Haffner, Patrick: "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[Lei10] Leitloff, Jens; Hinz, Stefan and Stilla, Uwe: "Vehicle detection in very high resolution satellite images of city areas". In: *IEEE Transactions on Geoscience and Remote Sensing* 48.7 (2010), pp. 2795–2806.

[Lei14] Leitloff, Jens; Rosenbaum, Dominik; Kurz, Franz; Meynberg, Oliver and Reinartz, Peter: "An operational system for estimating road traffic information from aerial images". In: *Remote Sensing* 6.11 (2014), pp. 11315–11341.

[Len08] Lenhart, DOMINIK; Hinz, STEFAN; Leitloff, JENS and Stilla, Uwe: "Automatic traffic monitoring based on aerial image sequences". In: *Pattern Recognition and Image Analysis* 18.3 (2008), pp. 400–405.

[Li16] Li, Hao; Kadav, Asim; Durdanovic, Igor; Samet, Hanan and Graf, Hans Peter: "Pruning filters for efficient convnets". In: *arXiv preprint arXiv:1608.08710* (2016).

[Li17] Li, Ke; Cheng, Gong; Bu, Shuhui and You, Xiong: "Rotation-insensitive and context-augmented object detection in remote sensing images". In: *IEEE Transactions on Geoscience and Remote Sensing* 56.4 (2017), pp. 2337–2348.

[Lia12] Liang, Pengpeng; Teodoro, Gregory; Ling, Haibin; Blasch, Erik; Chen, Genshe and Bai, Li: "Multiple kernel learning for vehicle detection in wide area motion imagery". In: *Proceedings of the IEEE International Conference on Information Fusion*. 2012, pp. 1629–1636.

[Lin14] Lin, Tsung-Yi; Maire, Michael; Belongie, Serge; Hays, James; Perona, Pietro; Ramanan, Deva; Dollár, Piotr and Zitnick, C Lawrence: "Microsoft coco: Common objects in context". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer. 2014, pp. 740–755.

[Lin17a] Lin, Tsung-Yi; Dollár, Piotr; Girshick, Ross; He, Kaiming; Hariharan, Bharath and Belongie, Serge: "Feature pyramid networks for object detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[Lin17b] Lin, Tsung-Yi; Goyal, Priya; Girshick, Ross; He, Kaiming and Dollár, Piotr: "Focal loss for dense object detection". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2980–2988.

[Liu15] Liu, Kang and Mattyus, Gellert: "Fast multiclass vehicle detection on aerial images". In: *IEEE Geoscience and Remote Sensing Letters* 12.9 (2015), pp. 1938–1942.

[Liu16] Liu, Wei; Anguelov, Dragomir; Erhan, Dumitru; Szegedy, Christian; Reed, Scott; Fu, Cheng-Yang and Berg, Alexander C: "Ssd: Single shot multibox detector". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer. 2016, pp. 21–37.

[Liu18] Liu, Li; Ouyang, Wanli; Wang, Xiaogang; Fieguth, Paul; Chen, Jie; Liu, Xinwang and Pietikäinen, Matti: "Deep learning for generic object detection: A survey". In: *arXiv preprint arXiv:1809.02165* (2018).

[Lon15] Long, Jonathan; Shelhamer, Evan and Darrell, Trevor: "Fully convolutional networks for semantic segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 3431–3440.

[Lon17] Long, Yang; Gong, Yiping; Xiao, Zhifeng and Liu, Qing: "Accurate object localization in remote sensing images based on convolutional neural networks". In: *IEEE Transactions on Geoscience and Remote Sensing* 55.5 (2017), pp. 2486–2498.

[Luo12] Luo, Pingting; Liu, Fuqiang; Liu, Xiaofeng and Yang, Yingqian: "Stationary vehicle detection in aerial surveillance with a UAV". In: *Proceedings of the IEEE International Conference on Information Science and Digital Content Technology (ICIDT)*. Vol. 3. 2012, pp. 567–570.

[Luo16] Luo, Wenjie; Li, Yujia; Urtasun, Raquel and Zemel, Richard: "Understanding the effective receptive field in deep convolutional neural networks". In: *Advances in Neural Information Processing Systems (NIPS)*. 2016, pp. 4898–4906.

[Maa13] Maas, Andrew L; Hannun, Awni Y and Ng, Andrew Y: "Rectifier nonlinearities improve neural network acoustic models". In: *International Conference on Machine Learning (ICML)*. Vol. 30. 1. 2013.

[Mag17] Maggiori, Emmanuel; Tarabalka, Yuliya; Charpiat, Guillaume and Alliez, Pierre: "Can semantic labeling methods generalize to any city? the inria aerial image labeling benchmark". In: *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. 2017, pp. 3226–3229.

[McC43] McCulloch, Warren S and Pitts, Walter: "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133.

[Mni13] Mnih, Volodymyr: Machine learning for aerial image labeling. Citeseer, 2013.

[Mon12] Montavon, Grégoire; Orr, Geneviève and Müller, Klaus-Robert: Neural networks: tricks of the trade. Vol. 7700. Springer, 2012.

[Moo02] Moon, Hankyu; Chellappa, Rama and Rosenfeld, Azriel: "Performance analysis of a simple vehicle detection algorithm". In: *Image and Vision Computing* 20.1 (2002), pp. 1–13.

[Mor12] Moranduzzo, Thomas and Melgani, Farid: "A SIFT-SVM method for detecting cars in UAV images". In: *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. 2012, pp. 6868–6871.

[Mor13] Moranduzzo, Thomas and Melgani, Farid: "Automatic car counting method for unmanned aerial vehicle images". In: *IEEE Transactions on Geoscience and Remote Sensing* 52.3 (2013), pp. 1635–1647.

[Mor14a] Moranduzzo, Thomas and Melgani, Farid: "Automatic car counting method for unmanned aerial vehicle images". In: *IEEE Transactions on Geoscience and Remote Sensing* 52.3 (2014), pp. 1635–1647.

[Mor14b] Moranduzzo, Thomas and Melgani, Farid: "Detecting cars in UAV images with a catalog-based approach". In: *IEEE Transactions on Geoscience and Remote Sensing* 52.10 (2014), pp. 6356–6367.

[Mun16] Mundhenk, T Nathan; Konjevod, Goran; Sakla, Wesam A and Boakye, Kofi: "A large contextual dataset for classification, detection and counting of cars with deep learning". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer. 2016, pp. 785–800.

[Nie18] Nie, Kun; Sommer, Lars; Schumann, Arne and Beyerer, Jurgen: "Semantic labeling based vehicle detection in aerial imagery". In: *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2018, pp. 626–634.

[Pan19] Pang, Jiangmiao; Chen, Kai; Shi, Jianping; Feng, Huajun; Ouyang, Wanli and Lin, Dahua: "Libra r-cnn: Towards balanced learning for object detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 821–830.

[Pen18] Peng, Chao; Xiao, Tete; Li, Zeming; Jiang, Yuning; Zhang, Xiangyu; Jia, Kai; Yu, Gang and Sun, Jian: "Megdet: A large mini-batch object detector". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 6181–6189.

[Qi16] Qi, Juntong; Song, Dalei; Shang, Hong; Wang, Nianfa; Hua, Chunsheng; Wu, Chong; Qi, Xin and Han, Jianda: "Search and rescue rotary-wing uav and its application to the lushan ms 7.0 earthquake". In: *Journal of Field Robotics* 33.3 (2016), pp. 290–321.

[Qu16] Qu, Shenquan; Wang, Ying; Meng, Gaofeng and Pan, Chunhong: "Vehicle Detection in Satellite images by incorporating objectness and convolutional neural network". In: *Journal of Industrial and Intelligent Information* 4.2 (2016).

[Qu17] Qu, Tao; Zhang, Quanyuan and Sun, Shilei: "Vehicle detection from high-resolution aerial images using spatial pyramid pooling-based deep convolutional neural networks". In: *Multimedia Tools and Applications* 76.20 (2017), pp. 21651–21663.

[Raz16] Razakarivony, Sebastien and Jurie, Frederic: "Vehicle detection in aerial imagery: A small target detection benchmark". In: *Journal of Visual Communication and Image Representation* 34 (2016), pp. 187–203.

[Red16] Redmon, Joseph; Divvala, Santosh; Girshick, Ross and Farhadi, Ali: "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 779–788.

[Red17] Redmon, Joseph and Farhadi, Ali: "YOLO9000: Better, Faster, Stronger". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 6517–6525.

[Red18] Redmon, Joseph and Farhadi, Ali: "Yolov3: An incremental improvement". In: *arXiv preprint arXiv:1804.02767* (2018).

[Rei06] Reinartz, Peter; Lachaise, Marie; Schmeer, Elisabeth; Krauss, Thomas and Runge, Hartmut: "Traffic monitoring with serial images from airborne cameras". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 61.3-4 (2006), pp. 149–158.

[Rei10] Reilly, Vladimir; Idrees, Haroon and Shah, Mubarak: "Detection and tracking of large number of targets in wide area surveillance". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer. 2010, pp. 186–199.

[Ren15] Ren, Shaoqing; He, Kaiming; Girshick, Ross and Sun, Jian: "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Advances in Neural Information Processing Systems (NIPS)*. 2015, pp. 91–99.

[Res17] Research, Transparency Market: Aerial Imaging Market - Global Industry Analysis, Size, Share, Growth, Trends and Forecast, 2017 – 2025. 2017.

[Res18] Research and Markets: Aerial Imaging - Global Market Outlook (2017-2026). 2018.

[Rin19] Ringwald, Tobias; Sommer, Lars; Schumann, Arne; Beyerer, Jurgen and Stiefelhagen, Rainer: "UAV-Net: A Fast Aerial Vehicle Detector for Mobile Platforms". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2019.

[Röm16] Römer, H; Kiefl, R; Henkel, F; Wenxi, C; Nippold, R; Kurz, F and Kippnich, U: "Using airborne remote sensing to increase situational awareness in civil protection and humanitarian relief - the importance of user involvement". In: *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences* 41 (2016).

[Ros58] Rosenblatt, Frank: "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6 (1958), p. 386.

[Rud08] Rudol, Piotr and Doherty, Patrick: "Human body detection and geolocalization for UAV search and rescue missions using color and thermal imagery". In: *Proceedings of the IEEE Aerospace Conference*. 2008.

[Rus96] Ruskoné, Renaud; Guigues, Laurent; Airault, Sylvain and Jamet, Olivier: "Vehicle detection on aerial images: A structural approach". In: *Proceedings of IEEE International Conference on Pattern Recognition (ICPR)*. Vol. 3. 1996, pp. 900–904.

[Sak17] Sakla, Wesam; Konjevod, Goran and Mundhenk, T Nathan: "Deep Multi-modal Vehicle Detection in Aerial ISR Imagery". In: *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2017, pp. 916–923.

[Sal83] Salton, Gerard and McGill, Michael J: Introduction to modern information retrieval. mcgraw-hill, 1983.

[San18] Sandler, Mark; Howard, Andrew; Zhu, Menglong; Zhmoginov, Andrey and Chen, Liang-Chieh: "Mobilenetv2: Inverted residuals and linear bottlenecks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 4510–4520.

[Sch10] Scherer, Dominik; Müller, Andreas and Behnke, Sven: "Evaluation of pooling operations in convolutional architectures for object recognition". In: *Artificial Neural Networks–ICANN 2010*. Springer, 2010, pp. 92–101.

[Sch15] Schmidhuber, Jürgen: "Deep learning in neural networks: An overview". In: *Neural networks* 61 (2015), pp. 85–117.

[Sha12] Shao, Wen; Yang, Wen; Liu, Gang and Liu, Jie: "Car detection from high-resolution aerial imagery using multiple features". In: *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. 2012, pp. 4379–4382.

[She16] Sherrah, Jamie: "Fully convolutional networks for dense semantic labelling of high-resolution aerial imagery". In: *arXiv preprint arXiv:1606.02585* (2016).

[Shi12] Shi, Xinchu; Ling, Haibin; Blasch, Erik and Hu, Weiming: "Context-driven moving vehicle detection in wide area motion imagery". In: *Proceedings of the IEEE International Conference on Pattern Recognition (ICPR)*. 2012, pp. 2512–2515.

[Shr16a] Shrivastava, Abhinav and Gupta, Abhinav: "Contextual priming and feedback for faster r-cnn". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer. 2016, pp. 330–348.

[Shr16b] Shrivastava, Abhinav; Gupta, Abhinav and Girshick, Ross: "Training region-based object detectors with online hard example mining". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 761–769.

[Shr16c] Shrivastava, Abhinav; Sukthankar, Rahul; Malik, Jitendra and Gupta, Abhinav: "Beyond skip connections: Top-down modulation for object detection". In: *arXiv preprint arXiv:1612.06851* (2016).

[Sia12] Siam, Mennatullah and ElHelw, Mohamed: "Robust autonomous visual detection and tracking of moving targets in UAV imagery". In: *Prcoeedings of the International Conference on Signal Processing*. Vol. 2. 2012, pp. 1060–1066.

[Sif14] Sifre, Laurent and Mallat, Stéphane: "Rigid-motion scattering for image classification". In: *Ph. D. dissertation* (2014).

[Sim14] Simonyan, Karen and Zisserman, Andrew: "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[Sin18] Singh, Bharat and Davis, Larry S: "An analysis of scale invariance in object detection snip". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 3578–3587.

[Som17a] Sommer, Lars; Nie, Kun; Schumann, Arne; Schuchert, Tobias and Beyerer, Jürgen: "Semantic labeling for improved vehicle detection in aerial imagery". In: *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. 2017.

[Som17b] Sommer, Lars W; Schuchert, Tobias and Beyerer, Jürgen: "Deep learning based multi-category object detection in aerial images". In: *Automatic Target Recognition XXVII*. Vol. 10202. International Society for Optics and Photonics. 2017, p. 1020209.

[Som17c] Sommer, Lars Wilko; Schuchert, Tobias and Beyerer, Jürgen: "Fast deep vehicle detection in aerial images". In: *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2017, pp. 311–319.

[Som18a] Sommer, Lars; Schmidt, Nicole; Schumann, Arne and Beyerer, Jürgen: "Search Area Reduction Fast-RCNN for Fast Vehicle Detection in Large Aerial Imagery". In: *Proceedings of the IEEE International Conference on Image Processing (ICIP)*. 2018, pp. 3054–3058.

[Som18b] Sommer, Lars; Schuchert, Tobias and Beyerer, Jürgen: "Comprehensive Analysis of Deep Learning based Vehicle Detection in Aerial Images". In: *IEEE Transactions on Circuits and Systems for Video Technology* (2018).

[Som18c] Sommer, Lars; Schumann, Arne; Schuchert, Tobias and Beyerer, Jurgen: "Multi feature deconvolutional faster r-cnn for precise vehicle detection in aerial imagery". In: *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2018, pp. 635–642.

[Som18d] Sommer, Lars; Steinmann, Lucas; Schumann, Arne and Beyerer, Jürgen: "Systematic evaluation of deep learning based detection frameworks for aerial imagery". In: *Automatic Target Recognition XXVIII*. Vol. 10648. International Society for Optics and Photonics. 2018, p. 1064803.

[Spr14] Springenberg, Jost Tobias; Dosovitskiy, Alexey; Brox, Thomas and Riedmiller, Martin: "Striving for simplicity: The all convolutional net". In: *arXiv preprint arXiv:1412.6806* (2014).

[Sri14] Srivastava, Nitish; Hinton, Geoffrey; Krizhevsky, Alex; Sutskever, Ilya and Salakhutdinov, Ruslan: "Dropout: a simple way to prevent neural networks from overfitting". In: *Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.

[Sze15] Szegedy, Christian; Liu, Wei; Jia, Yangqing; Sermanet, Pierre; Reed, Scott; Anguelov, Dragomir; Erhan, Dumitru; Vanhoucke, Vincent and Rabinovich, Andrew: "Going deeper with convolutions". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 1–9.

[Sze16] Szegedy, Christian; Vanhoucke, Vincent; Ioffe, Sergey; Shlens, Jon and Wojna, Zbigniew: "Rethinking the inception architecture for computer vision". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 2818–2826.

[Sze17] Szegedy, Christian; Ioffe, Sergey; Vanhoucke, Vincent and Alemi, Alexander A: "Inception-v4, inception-resnet and the impact of residual connections on learning". In: *Thirty-First AAAI Conference on Artificial Intelligence*. 2017.

[Tan17] Tang, Tianyu; Zhou, Shilin; Deng, Zhipeng; Zou, Huanxin and Lei, Lin: "Vehicle Detection in Aerial Images Based on Region Convolutional Neural Networks and Hard Negative Example Mining". In: *Sensors* 17 (2017).

[Tay18] Tayara, Hilal and Chong, Kil: "Object detection in very high-resolution aerial images using one-stage densely connected feature pyramid network". In: *Sensors* 18.10 (2018).

[Tie12] Tieleman, Tijmen and Hinton, Geoffrey: "Lecture 6.5-rmsprop, coursera: Neural networks for machine learning". In: *University of Toronto, Technical Report* (2012).

[Tue13] Tuermer, Sebastian; Kurz, Franz; Reinartz, Peter and Stilla, Uwe: "Airborne vehicle detection in dense urban areas using HoG features and disparity maps". In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 6.6 (2013), pp. 2327–2337.

[Uij13] Uijlings, Jasper RR; Van De Sande, Koen EA; Gevers, Theo and Smeulders, Arnold WM: "Selective search for object recognition". In: *International Journal of Computer Vision* 104.2 (2013), pp. 154–171.

[Van18] Van Etten, Adam; Lindenbaum, Dave and Bacastow, Todd M: "Spacenet: A remote sensing dataset and challenge series". In: *arXiv preprint arXiv:1807.01232* (2018).

[Wan18a] Wang, Chen; Bai, Xiao; Wang, Shuai; Zhou, Jun and Ren, Peng: "Multiscale visual attention networks for object detection in VHR remote sensing images". In: *IEEE Geoscience and Remote Sensing Letters* 16.2 (2018), pp. 310–314.

[Wan18b] Wang, Robert J; Li, Xiang and Ling, Charles X: "Pelee: A real-time object detection system on mobile devices". In: *Advances in Neural Information Processing Systems (NIPS)*. 2018, pp. 1963–1972.

[Wan19] Wang, Haoran; Wang, Zexin; Jia, Meixia; Li, Aijin; Feng, Tuo; Zhang, Wenhua and Jiao, Licheng: "Spatial Attention for Multi-Scale Feature Refinement for Object Detection". In: *Proceedings of the IEEE International Conference on Computer Vision Workshops (ICCVW)*. 2019.

[Waq19] Waqas Zamir, Syed; Arora, Aditya; Gupta, Akshita; Khan, Salman; Sun, Guolei; Shahbaz Khan, Fahad; Zhu, Fan; Shao, Ling; Xia, Gui-Song and Bai, Xiang: "iSAID: A Large-scale Dataset for Instance Segmentation in Aerial Images". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2019, pp. 28–37.

[Wen17] Wen, Wei; Xu, Cong; Wu, Chunpeng; Wang, Yandan; Chen, Yiran and Li, Hai: "Coordinating filters for faster deep neural networks". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 658–666.

[Woo18] Woo, Sanghyun; Hwang, Soonmin and Kweon, In So: "Stairnet: Top-down semantic aggregation for accurate one shot detection". In: *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2018.

[Xia10] Xiao, Jiangjian; Cheng, Hui; Sawhney, Harpreet and Han, Feng: "Vehicle detection and tracking in wide field-of-view aerial video". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2010, pp. 679–684.

[Xia18] Xia, Gui-Song; Bai, Xiang; Ding, Jian; Zhu, Zhen; Belongie, Serge; Luo, Jiebo; Datcu, Mihai; Pelillo, Marcello and Zhang, Liangpei: "DOTA: A large-scale dataset for object detection in aerial images". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 3974–3983.

[Xie17] Xie, Saining; Girshick, Ross; Dollár, Piotr; Tu, Zhuowen and He, Kaiming: "Aggregated residual transformations for deep neural networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 5987–5995.

[Xu16] Xu, Yongzheng; Yu, Guizhen; Wang, Yunpeng; Wu, Xinkai and Ma, Yalong: "A hybrid vehicle detection method based on viola-jones and HOG+ SVM from UAV images". In: *Sensors* 16.8 (2016).

[Xu17a] Xu, Yongzheng; Yu, Guizhen; Wang, Yunpeng; Wu, Xinkai and Ma, Yalong: "Car detection from low-altitude UAV imagery with the faster R-CNN". In: *Journal of Advanced Transportation* 2017 (2017).

[Xu17b] Xu, Zhaozhuo; Xu, Xin; Wang, Lei; Yang, Rui and Pu, Fangling: "Deformable convnet with aspect ratio constrained nms for object detection in remote sensing imagery". In: *Remote Sensing* 9.12 (2017).

[Yan18] Yang, Michael Ying; Liao, Wentong; Li, Xinbo and Rosenhahn, Bodo: "Deep learning for vehicle detection in aerial images". In: *Proceedings of the IEEE International Conference on Image Processing (ICIP)*. 2018, pp. 3079–3083.

[Yao08] Yao, Fenghui; Sekmen, Ali and Malkani, Mohan J: "Multiple moving target detection, tracking, and recognition from a moving observer". In: *Proceedings of the IEEE International Conference on Information and Automation*. 2008, pp. 978–983.

[Yu09] Yu, Qian and Medioni, Gérard: "Motion pattern interpretation and detection for tracking moving vehicles in airborne video". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2009, pp. 2671–2678.

[Yu15] Yu, Fisher and Koltun, Vladlen: "Multi-scale context aggregation by dilated convolutions". In: *arXiv preprint arXiv:1511.07122* (2015).

[Zei12] Zeiler, Matthew D: "ADADELTA: an adaptive learning rate method". In: *arXiv preprint arXiv:1212.5701* (2012).

[Zha03] Zhao, Tao and Nevatia, Ram: "Car detection in low resolution aerial images". In: *Image and Vision Computing* 21.8 (2003), pp. 693–703.

[Zha16] Zhang, Liliang; Lin, Liang; Liang, Xiaodan and He, Kaiming: "Is faster r-cnn doing well for pedestrian detection?" In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer. 2016, pp. 443–457.

[Zha18a] Zhang, Shifeng; Wen, Longyin; Bian, Xiao; Lei, Zhen and Li, Stan Z: "Single-shot refinement neural network for object detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 4203–4212.

[Zha18b] Zhang, Xiangyu; Zhou, Xinyu; Lin, Mengxiao and Sun, Jian: "Shufflenet: An extremely efficient convolutional neural network for mobile devices". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 6848–6856.

[Zha18c] Zhao, Zhong-Qiu; Zheng, Peng; Xu, Shou-tao and Wu, Xindong: "Object detection with deep learning: A review". In: *arXiv preprint arXiv:1807.05511* (2018).

[Zho15] Zhou, Bolei; Khosla, Aditya; Lapedriza, Agata; Oliva, Aude and Torralba, Antonio: "Object detectors emerge in deep scene cnns". In: *International Conference on Learning Representations* (2015).

[Zho17] Zhong, Jiandan; Lei, Tao and Yao, Guangle: "Robust vehicle detection in aerial images based on cascaded convolutional neural networks". In: *Sensors* 17.12 (2017).

[Zhu15] Zhu, Haigang; Chen, Xiaogang; Dai, Weiqun; Fu, Kun; Ye, Qixiang and Jiao, Jianbin: "Orientation robust object detection in aerial images using deep convolutional neural network". In: *Proceedings of the IEEE International Conference on Image Processing (ICIP)*. 2015, pp. 3735–3739.

[Zhu17] Zhu, Yousong; Zhao, Chaoyang; Wang, Jinqiao; Zhao, Xu; Wu, Yi; Lu, Hanqing, et al.: "Couplenet: Coupling global structure with local parts for object detection". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2017.

[Zhu18] Zhu, Pengfei; Wen, Longyin; Du, Dawei; Bian, Xiao; Ling, Haibin; Hu, Qinghua; Nie, Qinqin; Cheng, Hao; Liu, Chenfeng; Liu, Xiaoyu, et al.: "Visdrone-det2018: The vision meets drone object detection in image challenge results". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

[Zhu19] Zhu, Xizhou; Hu, Han; Lin, Stephen and Dai, Jifeng: "Deformable convnets v2: More deformable, better results". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 9308–9316.

[Zop18] Zoph, Barret; Vasudevan, Vijay; Shlens, Jonathon and Le, Quoc V.: "Learning Transferable Architectures for Scalable Image Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 8697–8710.

Publications

[1] Cormier, Mickael; Sommer, Lars Wilko and Teutsch, Michael: "Low resolution vehicle re-identification based on appearance features for wide area motion imagery". In: *Proceedings of the IEEE Winter Applications of Computer Vision Workshops (WACVW)*. 2016.

[2] Sommer, Lars W; Schuchert, Tobias and Beyerer, Jürgen: "Generating object proposals for improved object detection in aerial images". In: *Electro-Optical Remote Sensing X*. Vol. 9988. International Society for Optics and Photonics. 2016, 99880N.

[3] Sommer, Lars Wilko; Schuchert, Tobias and Beyerer, Jurgen: "A comprehensive study on object proposals methods for vehicle detection in aerial images". In: *9th IAPR Workshop on Pattern Recogniton in Remote Sensing (PRRS)*. 2016.

[4] Sommer, Lars Wilko; Teutsch, Michael; Schuchert, Tobias and Beyerer, Jürgen: "A survey on moving object detection for wide area motion imagery". In: *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2016.

[5] Coluccia, Angelo; Ghenescu, Marian; Piatrik, Tomas; De Cubber, Geert; Schumann, Arne; Sommer, Lars; Klatte, Johannes; Schuchert, Tobias, et al.: "Drone-vs-Bird detection challenge at IEEE AVSS2017". In: *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. 2017.

[6] Schumann, Arne; Sommer, Lars; Klatte, Johannes; Schuchert, Tobias and Beyerer, Jürgen: "Deep cross-domain flying object classification for robust UAV detection". In: *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. 2017.

[7] Sommer, Lars; Nie, Kun; Schumann, Arne; Schuchert, Tobias and Beyerer, Jürgen: "Semantic labeling for improved vehicle detection in aerial imagery". In: *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. 2017.

[8] Sommer, Lars; Schumann, Arne; Müller, Thomas; Schuchert, Tobias and Beyerer, Jürgen: "Flying object detection for automatic UAV recognition". In: *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. 2017.

[9] Sommer, Lars W; Schuchert, Tobias and Beyerer, Jürgen: "Deep learning based multi-category object detection in aerial images". In: *Automatic Target Recognition XXVII*. Vol. 10202. International Society for Optics and Photonics. 2017, p. 1020209.

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

[11] Acatay, Oliver; Sommer, Lars; Schumann, Arne and Beyerer, Jürgen: "Comprehensive Evaluation of Deep Learning based Detection Methods for Vehicle Detection in Aerial Imagery". In: *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. 2018.

[12] Lyu, Siwei; Chang, Ming-Ching; Du, Dawei; Li, Wenbo; Wei, Yi; Del Coco, Marco; Carcagnì, Pierluigi; Schumann, Arne; Munjal, Bharti; Choi, Doo-Hyun, et al.: "UA-DETRAC 2018: Report of AVSS2018 & IWT4S Challenge on Advanced Traffic Monitoring". In: *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. 2018.

[13] Nie, Kun; Sommer, Lars; Schumann, Arne and Beyerer, Jurgen: "Semantic labeling based vehicle detection in aerial imagery". In: *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2018, pp. 626–634.

[14] Schumann, Arne; Sommer, Lars; Müller, Thomas and Voth, Sascha: "An image processing pipeline for long range UAV detection". In: *Emerging Imaging and Sensing Technologies for Security and Defence III; and Unmanned Sensors, Systems, and Countermeasures*. Vol. 10799. International Society for Optics and Photonics. 2018, 107990T.

[15] Schumann, Arne; Sommer, Lars; Vogler, Max and Beyerer, Jurgen: "Onthology-based Masking Loss for Improved Generalization in Remote Sensing Semantic Image Retrieval". In: *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. 2018.

[16] Sommer, Lars; Acatay, Oliver; Schumann, Arne and Beyerer, Jürgen: "Ensemble of Two-Stage Regression Based Detectors for Accurate Vehicle Detection in Traffic Surveillance Data". In: *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. 2018.

[17] Sommer, Lars; Schmidt, Nicole; Schumann, Arne and Beyerer, Jürgen: "Search Area Reduction Fast-RCNN for Fast Vehicle Detection in Large Aerial Imagery". In: *Proceedings of the IEEE International Conference on Image Processing (ICIP)*. 2018, pp. 3054–3058.

[18] Sommer, Lars; Schuchert, Tobias and Beyerer, Jürgen: "Comprehensive Analysis of Deep Learning based Vehicle Detection in Aerial Images". In: *IEEE Transactions on Circuits and Systems for Video Technology* (2018).

[19] Sommer, Lars; Schumann, Arne; Schuchert, Tobias and Beyerer, Jurgen: "Multi feature deconvolutional faster r-cnn for precise vehicle detection in aerial imagery". In: *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2018, pp. 635–642.

[20] Sommer, Lars; Steinmann, Lucas; Schumann, Arne and Beyerer, Jürgen: "Systematic evaluation of deep learning based detection frameworks for aerial imagery". In: *Automatic Target Recognition XXVIII*. Vol. 10648. International Society for Optics and Photonics. 2018, p. 1064803.

[21] Valev, Krassimir; Schumann, Arne; Sommer, Lars and Beyerer, Jurgen: "A systematic evaluation of recent deep learning architectures for fine-grained vehicle classification". In: *Pattern Recognition and Tracking XXIX*. Vol. 10649. International Society for Optics and Photonics. 2018, p. 1064902.

[22] Zhu, Pengfei; Wen, Longyin; Du, Dawei; Bian, Xiao; Ling, Haibin; Hu, Qinghua; Nie, Qinqin; Cheng, Hao; Liu, Chenfeng; Liu, Xiaoyu, et al.: "VisDrone-DET2018: The Vision Meets Drone Object Detection in Image Challenge Results". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

[23] Zhu, Pengfei; Wen, Longyin; Du, Dawei; Bian, Xiao; Ling, Haibin; Hu, Qinghua; Wu, Haotian; Nie, Qinqin; Cheng, Hao; Liu, Chenfeng, et al.: "VisDrone-VDT2018: The vision meets drone video detection and tracking challenge results". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

[24] Azimi, Seyed Majid; Henry, Corentin; Sommer, Lars; Schumann, Arne and Vig, Eleonora: "SkyScapes Fine-Grained Semantic Understanding of Aerial Scenes". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2019.

[25] Coluccia, Angelo; Fascista, Alessio; Schumann, Arne; Sommer, Lars; Ghenescu, Marian; Piatrik, Tomas; De Cubber, Geert, et al.: "Drone-vs-Bird detection challenge at IEEE AVSS2019". In: *2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. 2019.

[26] Ringwald, Tobias; Sommer, Lars; Schumann, Arne; Beyerer, Jurgen and Stiefelhagen, Rainer: "UAV-Net: A Fast Aerial Vehicle Detector for Mobile Platforms". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2019.

List of Figures

List of Tables

Acronyms

ABO	average best overlap
AdaBoost	adaptive boosting
AGRC	Automated Geographic Reference Center
AP	average precision
AVPN	accurate vehicle proposal network
B	building
BB	bounding box
BING	Binary Normed Gradients
BN	batch normalization
Ca	car
CEM	context enhancement module
CGBN	Cross-GPU Batch Normalization
Cl	clutter

CNN	convolutional neural network
COWC	Cars Overhead with Context
CPU	Central Processing Unit
DCN	deformable convolutional network
DFL-CNN	Double Focal Loss - CNN
DLR	German Aerospace Center
DOTA	Dataset for Object DeTection in Aerial Images
DSC	depthwise separable convolution
DSM	digital surface model
DSSD	Deconvolutional SSD
DYOLO	Deconvolutional YOLO
EMT	Explicit Multi-Task
FC	fully connected layer
FCN	fully convolutional network
FN	false negative
FP	false positive
FPN	Feature Pyramid Network
FPS	frames per second

GAN	generative adversarial network
GPU	Graphics Processing Unit
GSD	ground sampling distance
GT	ground truth
HD	High Definition
HOG	Histogram of Oriented Gradients
ICF	Integral Channel Features
IMT	Implicit Multi-Task
IoU	Intersection over Union
IR	infrared
IS	impervious surface
ISPRS	International Society for Photogrammetry and Remote Sensing
LBP	Local Binary Pattern
LCI	large-size color image
LV	low vegetation
MFD	Multi Feature Deconvolutional
MLP	Multilayer Perceptron

MS COCO	Microsoft Common Objects in Context
NASNet	Neural Architecture Search Network
NMS	non-maximum suppression
NWPU	Northwestern Polytechnical University
OHEM	online hard example mining
PASCAL	Pattern Analysis, Statistical Modelling and ComputAtional Learning
PCA	Principal Component Analysis
PRC	precision-recall curve
RAM	Random-Access Memory
R-CNN	Regions with CNN features
ReLU	Rectified Linear Unit
R-FCN	Region-based Fully Convolutional Network
RNN	recurrent neural network
RoI	Region of Interest
RPN	Region Proposal Network
SAR	Search Area Reduction
SCI	small-size color image

SENet	Squeeze-and-Excitation Network
SGD	stochastic gradient descent
SIFT	Scale-Invariant Feature Transform
SPPNet	spatial pyramid pooling network
SSD	Single Shot MultiBox Detector
SVM	Support Vector Machine
T	tree
TAS	Things and Stuff
TP	true positive
UAV	unmanned aerial vehicle
UAVDT	UAV Detection and Tracking
UCAS-AOD	University of Chinese Academy of Sciences - Aerial Object Detection
VEDAI	Vehicle Detection in Aerial Imagery
VGG	Visual Geometry Group
VHR	Very-High-Resolution
VOC	Visual Object Classes
YOLO	You only look once

Table of Symbols

Calligraphic Symbols

$\mathcal{A}$	set of ground truth annotations
$\mathcal{B}$	batch
$\mathcal{O}$	set of object proposals

Greek Symbols

α	percentage of filters kept after pruning
β	shift parameter
γ	scale parameter
δ	weight decay
ϵ	small value
η	learning rate
λ	weighting factor
μ	mean
$\mu_{\mathcal{B}}$	mean over batch
ξ	width multiplier
ρ	resolution multiplier
σ^2	variance
$\sigma^2_{\mathcal{B}}$	variance over batch

τ	threshold
ω	momentum

Roman Symbols

a	ground truth annotation instance
b	bias
c	number of classes
d	dilation coefficient
$d \times d$	image resolution
f	convolutional filter
f_u	up-sampling factor
g	number of groups
k	number of default anchor boxes
$k \times k$	kernel size
l	number of bounding boxes per grid cell
n_o	number of object proposals considered for classification
o	object proposal instance
p	predicted object probability
p^*	class label
u	class label
x,y,h,w	bounding box center coordinates, height and width
y	perceptron output
A_{GT}	ground truth bounding box area
A_{pred}	predicted bounding box area
B	bounding boxes per grid cell
C	number of filters in a convolutional layer

D	number of channels
H,W	image width and height
L	loss
L_{cls}	classification loss
L_{CLS}	classification stage loss
$L_{Faster\ R\text{-}CNN}$	Faster R-CNN loss
L_{MT}	joint multi-task loss
L_{reg}	regression loss
L_{RPN}	region proposal network loss
L_{SL}	semantic labeling loss
N_{bg}	number of pixels asigned to background classes
N_{car}	number of pixels asigned to class car
N_{tree}	number of pixels asigned to class tree
N_{is}	number of pixels asigned to class impervious surface
P	precision
$\tilde{P}$	interpolated precision
R	recall
$S \times S$	grid size
$\mathbf{b}$	bias vector
$\hat{\mathbf{b}}$	rescaled bias vector
$\mathbf{e}$	all-ones vector
$\mathbf{h}$	local feature vector
$\hat{\mathbf{h}}$	network activations with zero mean and unit variance
$\tilde{\mathbf{h}}$	network activations
$\tilde{\mathbf{h}}^{BN}$	normalized network activations
$\mathbf{t}$	parameterized coordiantes of a predicted bounding box

$\mathbf{t}^*$	parameterized coordiantes of a ground truth bounding box
$\mathbf{t}^u$	parameterized coordiantes of a predicted bounding box associated to class u
$\mathbf{v}$	parameterized coordiantes of a ground truth bounding box
$\mathbf{w}$	perceptron weight vector
$\mathbf{x}$	perceptron input vector
$\mathbf{W}$	neural network layer weight matrix
$\hat{\mathbf{W}}$	rescaled neural network layer weight matrix