TALLINN UNIVERSITY OF TECHNOLOGY

School of Information Technologies

Majnun Abdurahmanov 172635IVSM

AUTOMATED DETECTION OF OBJECTS OF INTEREST FROM AERIAL IMAGERY USING TRANSFER LEARNING

Master's thesis

Supervisor: Juhan-Peep Ernits
PhD

Tallinn 2020

TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Majnun Abdurahmanov 172635IVSM

HUVIOBJEKTIDE AUTOMAATNE TUVASTAMINE ORTOFOTODELT KASUTADES ÜLEKANDEÕPET

Magistritöö

Juhendaja: Juhan-Peep Ernits
PhD

Tallinn 2020

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature, and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Majnun Abdurahmanov

14.05.2020

Abstract

In computer vision, detection and classification of objects of interest in aerial imagery have numerous applications in areas such as land surveillance, traffic surveillance and tracking, disaster management, smart parking, urban planning, to name a few. However, due to the lower resolutions of the objects and the effect of noise in aerial imageries, there are extra challenges comparing to object detection from ground images since distinguishing features for the objects of interest in aerial imageries are more troublesome to discern. In the current thesis, we address the problem of object detection from aerial imagery using state-of-the-art Convolutional Neural Networks (CNN). To investigate this problem, we first conduct a literature review of the state-of-the-art object detection algorithms and summarize the related work. We then evaluate the execution of three state-of-the-art CNN algorithms by applying transfer learning on a publicly available high-resolution aerial imagery dataset. We analyze the results on three datasets with different resolutions and conduct a robust comparison of the selected algorithms. In the end, we discuss how transfer learning helps us to achieve impressive 33.2 average precision score with mask regional convolutional neural network (Mask R-CNN) that is trained merely on 236 training images and the impact of other key factors such as ground sample distance (GCD), pixel-wise area of the object of interest.

Keywords

Computer vision, aerial imagery, object detection, transfer learning, deep learning, convolutional neural networks.

Annotatsioon

HUVIOBJEKTIDE AUTOMAATNE TUVASTAMINE ORTOFOTODELT KASUTADES ÜLEKANDEÕPET

Aerofotodelt huviobjektide tuvastamisel ja klassifitseerimisel on arvutinägemise valdkonnas arvukalt rakendusi valdkondades, nagu maaseire, liikluse jälgimine, katastroofide ohjamine, nutikas parkimine ja linnaplaneerimine. Objektide madalama eraldusvõime ja müra mõju tõttu aerofotodel, võrreldes objekti tuvastamisega lähedalt tehtud fotodega, on siiski täiendavaid väljakutseid. Aerofotode puhul on huvipakkuvad objektid võrreldes vaateväljaga tihti väikesed ja rakurss on ülevalt alla või nurga all, mida lähedalt pildistades tihti ei esine. Käesolevas magistritöös käsitleme objektide tuvastamise probleemi aerofotodelt, kasutades kaasaegsemaid konvolutsioonilisi närvivõrke (CNN). Selle probleemi uurimiseks viime kõigepealt läbi tipptasemel objektide tuvastamise algoritmide kirjanduse ülevaate. Seejärel hindame kolme moodsa erineva konfiguratsiooniga CNN-algoritmi käitumist, rakendades ülekandeõpet vastloodud ise annoteeritud väikeses andmekogumis. Järgnevalt analüüsime oma katseandmete erineva karakteristikuga tulemusi ja võrdleme nende kolme tipptasemel algoritmi jõulist võrdlust. Lõpuks arutame, kuidas ülekandeõpe aitab meil saavutada muljetavaldava 33,2 keskmise täpsuse skoori Mask R-CNN närvivõrguga, mille treenimiseks kasutati ainult 236 treeningpilti. Lisaks analüüsime ka muude oluliste tegurite, näiteks maapealse valimi kauguse mõju (GCD) ja huvipakkuva objekti pikslite arvu usaldusväärseks tuvastamiseks.

Märksõnad

arvutinägemine, aerofoto, objektide tuvastamine, ülekandeõpe, süvaõpe, konvolutsioonilised närvivõrgud.

Lõputöö on kirjutatud ingilise keeles ning sisaldab teksti 67 leheküljel, 5 peatükki, 25 joonist, 3 tabelit.

List of abbreviations and terms

	Dots per inch
AI	Artificial Intelligence
AP	Average Precision
AR	Average Recall
BN	Batch Normalization
CE	Cross Entropy
ClusDet	Clustered Detection
CNN	Convolutional Neural Network
COCO	Common Objects in Context
CVAT	Computer Vision Annotation Tool
FCN	Fully Convolutional Network
FN	False Negative
FP	False Positive
FPN	Feature Pyramid Network
GOI	Ground Object of Interest
GSD	Ground Sampling Distance
HSC	Histograms of Sparse Codes
IoU	Intersection over Union
KCF	Kernelized correlation filters
LDAP	Lightweight Directory Access Protocol
LVIS	Large Vocabulary Instance Segmentation.
mAP	Mean Average Precision

QGIS	Geographic Information System
R-CNN	Regional Convolutional Neural Network
RoI	Region of Interest
RPN	Region Proposal Network
TalTech	Tallinn University of Technology
TP	True Positive
UAV	Unmanned Aerial Vehicle
UI	User Interface
UX	User Experience
YOLO	You Look Only Once

Table of Contents

List of figures

List of tables

1 Introduction

1.1 General Overview

Computer Vision has become ubiquitous in our day to day life, with applications including but not limited to earth observation, disaster management, autonomous vehicles, urban planning and city management, mobility and infrastructure, border and coastal security, virtual reality, mapping and so on. Many of these applications perform visual recognition tasks such as image classification, localization, and detection. Recent developments in deep learning approaches have significantly advanced the performance of these state-of-the-art visual recognition systems.

Detection and classification of objects of interest in aerial imagery have numerous applications and use cases in the humanitarian, national security, and commercial realms. On the commercial front, businesses have already attempted to infer retail traffic from parking lot density levels, and tracking delivery trucks in near real-time is one of the far-field goals of understanding aerial and satellite imageries [1]. For tax assessment purposes, usually, surveys are conducted manually on the ground, which is essential to calculate the real value of properties, and this can help to identify when having a swimming pool that could increase property prices [1]. Similarly, in a neighborhood or around a store, the count of cars can symbolize the levels of economic activity at that place. By achieving this through understanding aerial imagery by using Artificial Intelligence (AI), can significantly help in these processes by removing the inefficiencies, and the high cost and time required by humans [2]. From national security standpoint, detecting the build-up of war materiel in unstable regions would provide obvious value, as would locating convoys of vehicles vectored towards unmanned border crossings or identifying a large number of vehicles staging just outside the range of terrestrial border monitoring equipment [3]. On the humanitarian front, governments have attempted to infer the scope of natural disasters such as road cracks, damaged buildings, a hurricane from clusters (or absences) of vehicles, or determine optimal travel routes for disaster relief in unknown areas based on observations of local vehicle movements.

1.2 Motivation

One particular application area of computer vision is in unmanned aircraft systems, which are used to do surveillance and control in places like Land and Coastal Borders, emergency cases, military. These systems consist of unmanned aerial vehicles (UAV) and station aircrafts. The UAVs are used for surveillance purposes in-band humans in station aircrafts monitor the taken images or videos to detect and classify target objects.

Unlike well-trained machines, human brains are more error prone. In aircraft stations in one blink of an eye, the human can miss a vital threat or danger caused by objects of interest. If the UAVs are well trained by using computer vision, then these human-introduced problems can be easily eliminated. By doing so, we could offload this kind of more tedious activity to machines and humans can concentrate on some other important stuff.

There are already several companies in the industry that provide UAVs that can perform this task relatively good enough. However, they keep their research privately, and the other industry companies are left with no help from academic research.

1.3 The Hypothesis

Object detection and classification in aerial images is challenging for several reasons. The main reason is that the aerial imagery tends to have a high ratio, and that makes the target objects to look extremely small. When the image is zoomed in, the quality gets lower, thus becoming more challenging to detect and classify the target objects. Secondly, I can say that usually in aerial imagery, the objects of interest are sparsely and non-uniformly distributed in the image, thus making the detection very inefficient.

Previously some researchers have attempted to apply transfer learning from one base task to another target task. For example, Pi et al. [4] evaluated YOLOv2 [5] deep learning model by applying transfer learning on the drone dataset containing merely eight videos (65,000 frames) for natural disaster response, Cisek et al. [6] tested effectiveness of AlexNet [7] on 12,000 car overhead imagery dataset to solve car parking lot problem.

Taking the previous successful transfer learning applied researches on aerial imagery problems into consideration and given that the Republic of Estonia Land Board makes high resolution aerial images available, we want to conduct a research in order to answer the following questions:

- Can we get reasonably good results by applying transfer learning to detect and classify objects of interest in aerial imagery on a relatively small manually annotated custom dataset?
- How do different pre-trained deep learning models perform on the same target dataset?

In this empirical research, first, we will do a thorough literature review on what techniques are used to detect and classify target objects in aerial images. Following, we are going to apply transfer learning method by fine-tuning parameters in pre-trained deep learning algorithms that from detectron2 model zoo [8] to see if we can reasonably good results and try to solve the existing problem with detecting and classifying target objects in aerial imageries.

1.4 Research Objectives

The main objective of this research, as described throughout this thesis, is to evaluate the performance of state-of-the-art deep learning models by applying transfer learning on a small set of high-resolution aerial imagery. Other objectives include:

- Research and review the state-of-the-art deep neural network architectures that are used for the detection and classification of objects of interest in aerial imageries.
- Understand how transfer learning and small dataset can help us to detect and classify objects of interest in aerial imageries.
- Discuss future opportunities.

1.5 Methodology

In order to answer the research questions, I conducted a comprehensive literature review, moreover, I contacted three Estonian startups that are building the UAVs in order to identify the most important categories of objects of interest. Figure 1 illustrates the overall the empirical research methodology [9] which is explained at length in the following chapters and sections.

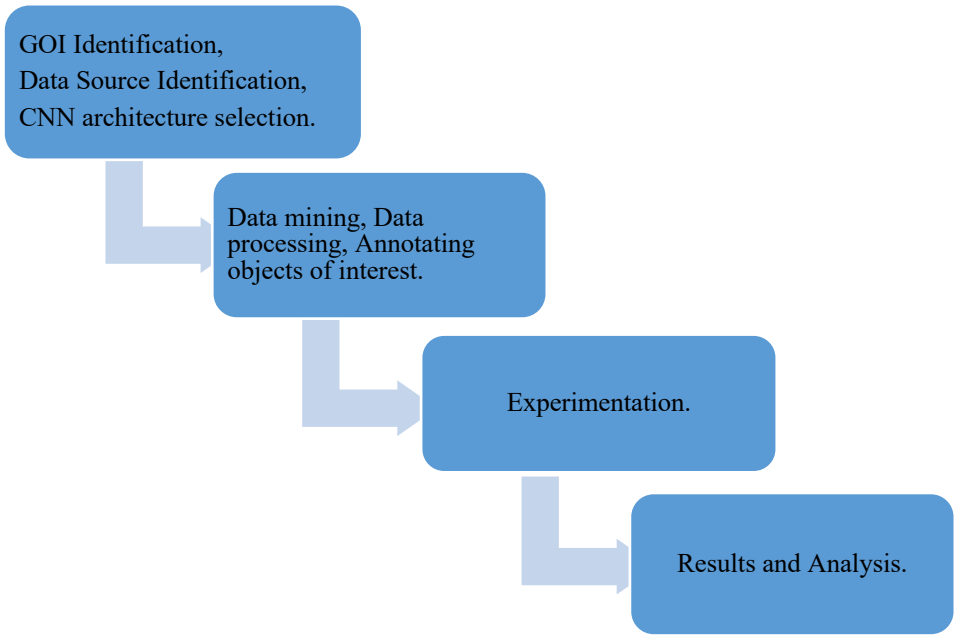

Figure 1. Research Methodology

Web mining was implemented to find publicly available high-resolution aerial imagery [10]. To yield best results while web mining the following key words were used: aerial, imagery, orthophotos, UAV. In the gathered dataset objects of interest were manually labelled then were split into training (85%) and testing (15%) datasets. After a comprehensive literature review of the pretrained publicly available CNN architecture

models from detectron2 model zoo [8] three models namely, Faster R-CNN, Mask R-CNN and RetinaNet were selected. These are pre-trained models on COCO dataset. For environment setup web mining was implemented and Google Collaboratory was chosen [11]. In order to evaluate the deep learning model performance, mean average precision [12] metrics was used.

1.6 Structure of the Thesis Document

In the pages that follow, this thesis document is divided into four primary chapters. Chapter 2 includes a review of existing literature on the subjects of object detection and classification, transfer learning, objection detection approaches, and state-of-the-art deep learning models. Then, in Chapter 3, we will walk through how we processed and labelled the manually collected dataset and discuss about the experimental setup. Following, in Chapter 4, we evaluate the execution of the three state-of-the-art CNN algorithms with different configurations by applying transfer learning and analyze the results on our test dataset with different characteristics and conduct a robust comparison between these three cutting-edge algorithms. Finally, in Chapter 5, we conclude the whole research, discuss the challenges of object detection in aerial images and talk about the future opportunities for improvement.

2 Literature Review and Background theory

2.1 Related work

Aerial imageries contain vast amount of data. That is a prominent reason that understanding manually only by humans is immensely complicated. Therefore, since computing machines entered our lives there has always been continuously ongoing research to teach machines to understand those imageries. As these imageries have a large focal distance to objects of interest in them, Computer Vision face a lot of challenges. The distribution of objects of interest classes are very imbalanced, obviously in an arbitrary orthophoto of a living area we will notice so many cars than busses, trucks or even buildings. Another difficulty is some objects of interest such as passenger vehicles or say motorbikes are often take very tiny pixel wise area inside the image. State-of-the-art Convolutional Neural Network based algorithms often down sample the images, thus resulting in immense information loss for these objects. Hence it becomes extremely difficult to identify those objects of interest.

In the last decade alone, the seamless integration of data into everyday life has resulted in exponentially large, multimodal datasets (e.g., photos, videos, blogs, and tweets) in the public domain such as online content sharing platforms and social networking sites. Within the context of understanding aerial imageries a lot of researches have been conducted.

Several research projects have looked into discovering and detecting ground objects of interest from orthophotos. Since these aerial imageries contain so much data, researchers have challenged these imageries understanding with various ideas and proposals. For UAVs to discover post disaster damages Baker et al. [13] proposed a Monte-Carlo algorithm which was proven work faster than the conventional methods.

Radovic et al. [14] attempted to detect solve discovering airplanes from aerial imagery challenge by applying transfer learning method on YOLO algorithm [15].

Han et al. [16] proposed an embedded system framework called Deep Drone to power drones with vision, int other words letting the drones to do automatic human detection and tracking. For object detection they used Faster R-CNN [17], the accurate but slow

detection algorithm as well as YOLO [15], the less accurate but fast-tracking algorithm to make the system both fast and accurate and additionally, for tracking they used and kernelized correlation filters (KCF).

Farrukh et al. [3] have studied detecting military vehicles and distinguishing them from non-military vehicles on low-altitude aerial imageries by deploying Faster R-CNN [17], SSD [18] and recurrent fully convolutional neural network (R-FCN) [19] algorithms.

Narayanan et al. [20] demonstrated the possibility of utilizing a high-performance cloud computer for real-time aerial objects of interest detection from UAVs.

Guirado et al [21]employed Faster R-CNN algorithm [15] to track and count the number of whales from satellite imagery.

Mundhenk et al. [1] introduced new aerial overhead imagery dataset with a large diverse set of cars to classify, detect and count cars, they have experimented the results with several state-of-the-art CNN architectures.

Pi et al. [4] reviewed YOLOv2 [5] deep learning model by applying transfer learning method on their newly introduced VOLAN dataset containing eight videos of the areas that had natural disasters.

Yang et al. [22] proposed a Clustered Detection (ClusDet) network that unifies object clustering and detection in an end-to-end framework.

Uus et al. [23] reviewed detecting different types of vehicles from aerial imagery by using YOLOv3 [24].

In previous works, researchers have not experimented and documented detecting ground of objects of interest by re training pretrained state-of-the-art models such as Faster R-CNN [15], mask R-CNN [25], RetinaNet [26] on very small dataset that contains super high resolution aerial imager and compared the results. Those researches focused mainly on using one deep learning models and comparing results with previous baselines.

2.2 Object Detection and Classification

Object detection is an essential problem task for computer vision techniques to solve. Here the idea is to determine whether the specific features in image data are present or not. By using computer vision, a detected target object then is classified to a set of pre-defined classes. This operation in computer vision is called object classification. With that being said, object detection and classification are fundamental building blocks of artificial intelligence. We discuss Object Localization, classification and detection tasks in detail further. Computer Vision helps computers to understand the visual world around us. The object detection and recognition in usual images due to recent studies have been performing with pretty high confidence scores.

Computer Vision tries to solve various problems. When we humans see the image what we do is first classify the objects inside the image. For example, for the following image, we would categorize the objects as a car and a building. When we teach the same thing to computers, we teach them how to classify. In other words, the task becomes image classification. However, to make the computer more intelligible, we need to teach them where that particular object is inside the image. In deep learning, we refer to it as object localization.

Image classification is the type of such computer vision task that tries to predict the class of an object in the image. Object Localization (Figure 2) is the type of Computer Vision task such that it tries to identify the location of one or more objects of interest in the given image. Mostly it draws a bounding box around the object it classifies.

Figure 2: Object classification and localization.

On the other hand, Object detection (Figure 3) is the type of Computer vision task that does Object Localization for more than one different type of classes. So, it detects the class of the object and where it is located in the given image.

Figure 3: Object detection.

2.3 Evolution of Object detection

In this section, we briefly discuss how modern object detection algorithms evolved throughout the artificial intelligence history and study the main object detection approaches.

In computer vision, as we mentioned in previous 2.2 the task of object detection problem is to determine where the objects of interest (e.g., cat, dog) are located in a given image (a.k.a. object localization) and to which category they belong to (a.k.a. object classification). The traditional object detection algorithms solved this task in three stages, namely informative region selection, feature extraction, and classification.

In the informative region selection stage, we scan the whole image with a multi-scale sliding window, then to recognize different objects, we extract visual features and finally run a classifier that distinguishes an object of interest from all other classes. In this approach, however, especially region selection is computationally inefficient, not always very accurate, and produces too many redundant windows [27].

Thanks to the emergency of deep neural networks, researchers recently (mostly last decade) proposed more sophisticated object detection algorithms. To learn the generic features in the input visual such as shapes, edges, colors, we apply convolutional filters(kernel). Hence, these deep neural networks are often referred to as convolutional neural networks (CNN). History of neural networks dates back to the 1950s when scientists had the intention of simulating the human brain on machines to solve general problems [28]. Later, in the late 1980s with the birth of backpropagation optimization technique [29], neural networks became even more popular in the computer science research community. However, until the mid-2000s due to huge overfitting, lack of large-scale data, and limited computational power, neural networks remained actual only in the artificial intelligence research community. Following the late 2000s, we began to see the recovery of deep learning, especially when the tech industry started implementing these ideas in their businesses. It was attributed to the emergency of large-scale training data, more powerful GPUS, and significant advancement of neural network architectures [27].

Convolutional neural network (CNN) is the most representative of deep learning in computer vision. Typical CNN architectures consist of convolutional, pooling, and fully connected layers. The state-of-the-art CNN architectures solve the vision problem in two different ways. One follows the traditional object detection approach, namely generating region proposals first, then classifying more fine-grained classes such as cats, cars, etc. The other approach regards object detection problem as a classification task and infer the objects of interest directly. These algorithms are sometimes called as two-stage detectors and one-stage detectors, respectively. To give an example, more widely adapted two-

stage object detectors are R-CNN [30], Fast R-CNN [31], Faster R-CNN [17], Mask R-CNN [25], FPN [17] and one stage detectors are YOLO [15] , SSD [18], YOLOv2 [5]. Figure 4 illustrates the evolution of object detection algorithms based on these two main approaches.

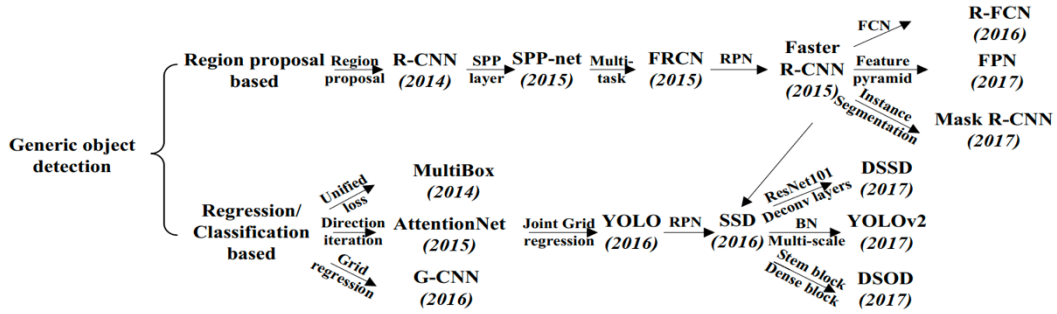

Figure 4. The evolution of the object detection algorithm based on the two main approaches/framework. We adopt this figure from [27].

In the region proposal-based framework side (a.k.a. two-stage detector) one of the first proposed algorithms was R-CNN [30] which adopted selective search to generate about 2k region proposals for each image, then utilizing CNN module to extract 4096-dimensional features and finally running a classifier to distinguish objects of interest form other classes. This algorithm obtained mean average precision (mAP) of 53.3% with more than 30% over the previous best result (HSC [32]) on the PASCAL VOC 2012 benchmark dataset [27]. Despite enormous improvements over traditional object detection methods, it still had several drawbacks. Mainly, these disadvantages were due to redundant region proposals and the training process being computationally costly. To overcome these problems, Fast R-CNN was proposed with the novelty of multi-tasks loss and region of interest (ROI), and later Faster R-CNN was introduced with an additional Region Proposal Network (RPN) [27].

In the regression/classification-based framework side (a.k.a. one-stage detector), a novel object detection algorithm called YOLO [15] was proposed by Redmon et al. The main idea of YOLO [15] was dividing the input image into grids and inferring the objects centered in those grids. An improved version, YOLOv2 [5] that was later proposed, which adopts several impressive strategies, such as batch normalization (BN) [33], anchor

boxes, dimension clusters, and multi-scale training [27]. The difficulty that YOLO [15] faced with was having tiny objects, mainly because of multiple downsampling while dividing into grids. To prevail these challenges, Liu et al. proposed Single Shot MultiBox Detector (SSD) [18] which was inspired by the anchors adopted in MultiBox [34], RPN [17] and multi-scale representation. Unlike YOLO [15] using fixed grids, SSD instead introduces a set of default anchor boxes with different aspect ratios and scales to discretize the bounding boxes. To handle objects with various pixel size areas, the network fuses predictions from multiple feature maps with different resolutions [27].

2.4 Transfer Learning

When deep neural networks are trained, the first layers learn the features which turn out not to be very specific to a particular dataset or task but are often generally useful for various tasks. These learned features eventually transition from being general to being specific to a dataset or task by the last layer of the deep neural network [35]. This behavior of the deep neural networks has given birth to a healthy idea of applying knowledge that is gained from one task(base) to another task(target). To give an example, we can train a neural network to recognize objects like cats, dogs, and then use fully or partially that knowledge to help us do a better job on some other computer vision tasks such as reading x-ray scans.

In the real-world, an entire Convolutional Neural Network is rarely trained from scratch with random parameter initialization. That is because not everyone has a relatively sufficiently huge dataset to train the neural network. Instead, the conventional approach is to pre-train a neural network on a large dataset, for example on ImageNet or COCO that contains millions of images with thousands of different objects of interest, and then use the Convolutional Neural Network either as an initialization or a fixed feature extractor for a particular computer vision task. While applying transfer learning from base task to target task, we usually want to follow one of two existing approaches.

The first approach is called feature extractor, which freezes the features pre-trained on base task. Most of the time, these deep neural networks are pre-trained on large datasets such as ImageNet or COCO dataset [36]. Then the last fully connected layer is removed as the last layer's output is different than what we usually want for our dataset.

Furthermore, we use the resulting neural network as a fixed feature extractor for our dataset. To give an example, in the case of AlexNet [7], the last hidden layer right before the output layer usually outputs a 4096-dimensional vector of an image that contains activations. Once we extract these features for all images, we train a separate linear classifier (e.g., Softmax classifier [37]) for our dataset.

Another transfer learning approach is to train a base network and then copy its first n layers to the first n layers of a target network. The remaining layers of the target network are then randomly initialized and trained toward the target task. One can choose to backpropagate the errors from the new task into the base (copied) features to fine-tune them to the new task, or the transferred feature layers can be left frozen, meaning that they do not change during training on the new task. This approach is motivated by the observation that the earlier features of a CNN contain more generic features such as edge detectors, colors, or blob detectors that is quite generic to many tasks. However, later layers of the CNN become progressively more specific to the details of the classes contained in the original dataset. For example, in the case of ImageNet, which contains many dog breeds, a significant portion of the representational power of the CNN may be devoted to features that are specific to differentiating between dog breeds.

However, there are some caveats that we should consider before applying transfer learning from some base pre-trained models to our target model. If the target dataset is relatively small and the number of parameters is immense, fine-tuning may result in overfitting. That is why these parameters are often left frozen in the early layers of the network. On the other hand, if the target dataset is extensive or the number of parameters is small so that overfitting is not a problem, then the base features can be fine-tuned to the new task to improve the model performance. Of course, if the target dataset is enormous, then there would be little need to transfer because the lower-level filters could just be learned from scratch on the target dataset [35].

3 Experiments

In this Chapter we introduce the aerial imagery dataset that we have built as well as the network architectures of the models that we have used for experiments and transfer-learning scheme we have used for pre-training. Next, three different pre-trained models and their performance and results are displayed. In the end we discuss the key factors that influence model performance, data challenges as well as future opportunities.

3.1 Data

Computer Vision's enormous success in its application areas have been tremendous and almost everyone else know about it. However, it is still such area in Artificial Intelligence that suffers from publicly available labelled data for research. Especially when it comes to aerial imagery.

In this empirical research in order to feed the deep neural networks first we did some web mining and found out some publicly available aerial imageries. One of the few publicly available aerial imageries was orthophoto overage of Estonia provided by Estonian Land Board [10]. Orthophotos are processed aerial photos from which distortions caused by terrain relief, camera tilt relative to the ground at the moment of exposure and camera central projection are removed. These orthophotos are quite diverse and in fact have the orthophotos in three different scales. These imageries are very high-resolution imageries with 1:2000, 1:10000 and 1:20000 scales. It is worth to mention that the spatial data is updated regularly every year, in our case we downloaded the latest imageries from 2019. The aerial imageries are at a nadir view angle such that it resembles satellite imagery.

For our experiments we decided to use imageries from orthophoto collection of densely populated urban areas in Estonia which is 1:2000 scale, and this is because these imageries tend to have smaller Ground Sampling Distance (GSD) [38]. In fact, these orthophotos are produced with GSD of 10-16cm, and the used equipment is aerial camera Leica ADS100-SH100 where the focal length is 120mm and height above ground level is 1250m [39]. Moreover, the ground sample distance (GSD) of these orthophotos are very close to GSD of aerial imageries generated by modern unmanned aerial vehicles (UAV).

For the conducting experiments for our research we downloaded twenty imageries and used sixteen of those as training imagery and the remaining four imagery for testing our model. One of the reasons why we are using a few imageries is because manually annotating them will be so cumbersome and is not in the scope of this thesis. Our training and testing dataset distribution is as follows.

Table 1. Training and testing dataset description

Category	***Training #instances***	***Testing #instances***
Car	11038	2884
Airplane	40	6
Bus	287	50
Watercraft	64	60
Building	1659	532
Truck	354	14
Total	13442	3546

Later, to evaluate final models and to conduct extensive comparison, we introduce two more test datasets by just down sampling the images in the original test dataset. We down sample test images by a factor of 2 and 4 in order to get different data characteristics. By down sampling the image by a factor of 2, we get the new lower resolution image as if it was taken from twice higher altitude. The main idea behind introducing these datasets is to assess the flexibility of the performance of the models on various resolution aerial imagery. The following Table 2 describes our test datasets in detail. We name the test datasets as Dataset followed by their GSD value (Dataset10).

Table 2. Description of test datasets

Name	#Instances	GSD	WidthxHeight
Dataset10	43	10sm	2000x2000
Dataset20	43	20sm	1000x1000
Dataset40	43	40sm	500x500

3.1.1 Data Processing and Annotation

These orthophotos are quite large tiff formatted raster images and to open them for viewing requires special software applications. For this purpose, we have used open source Geographic Information System standalone desktop application (QGIS) [40].(Figure 5)

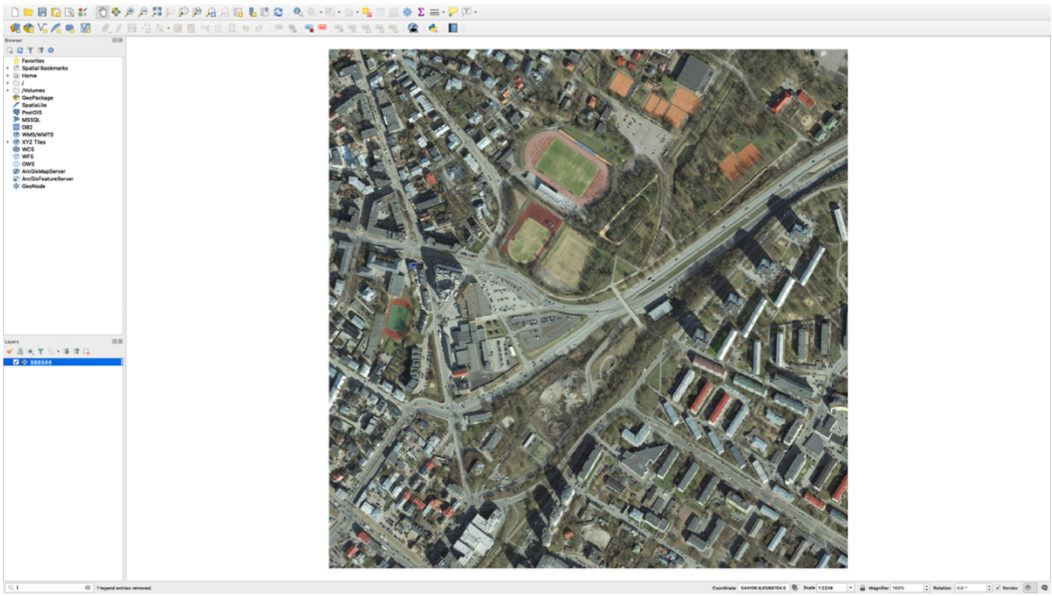

Figure 5. QGIS raster image viewer [40].

As we mentioned in one of the previous sections 1.5 we will have six classes of objects of interest for our experiment. We are generalizing any size of planes as airplanes, all watercrafts such as ferry, ship as watercraft, all mini passenger vehicles as car, and public transportation vehicles of trolleybuses and buses as bus, alongside buildings and trucks.

Data annotation is very essential part while preprocessing data in machine learning for computer vision tasks. In order to annotate the dataset, we decided to use one of the most preferred data annotation tools in the computer vision industry and research community called Computer Vision Annotation Tool (CVAT) [41]. CVAT is a powerful and efficient web-based tool that helps us to annotate videos or images for training Computer Vision models. CVAT has various powerful features such as interpolation of bounding boxes between key frames, shortcuts for most of critical actions, dashboard with a list of annotation tasks, LDAP [42] and basic authorization and so on. It was created with an intention for the usage professional data annotation tasks, hence UX [43] and UI [44] are optimized especially for computer vision tasks. We will set up CVAT on our local machine for data annotation task. Installing CVAT on our local machine is quite simple, we just need to clone the GitHub repository [41] and run a few commands to start up the necessary bits there as it comes with runnable docker [45] images.

We run the command *docker-compose build* in order to build the docker images, this operation usually takes up to three minutes to complete, because it downloads some public docker images such as ubuntu:16.04 and all necessary ubuntu packages to CVAT local server. Then we run the command of *docker-compose up -d* command to start up the container and it takes some time as it is downloading public docker images like postgres [46], Redis [47].

Once we start up CVAT we log in the system by default user credential. Once we login the system, we need to create CVAT annotation task by setting our classes and by adding our images. As we can see from below Figure 6 when we create a new task for annotation, we have several steps to follow, naming our annotation tasks, specify the labels and adding the images. Here labels refer to classes, in other words categories objects of interest.

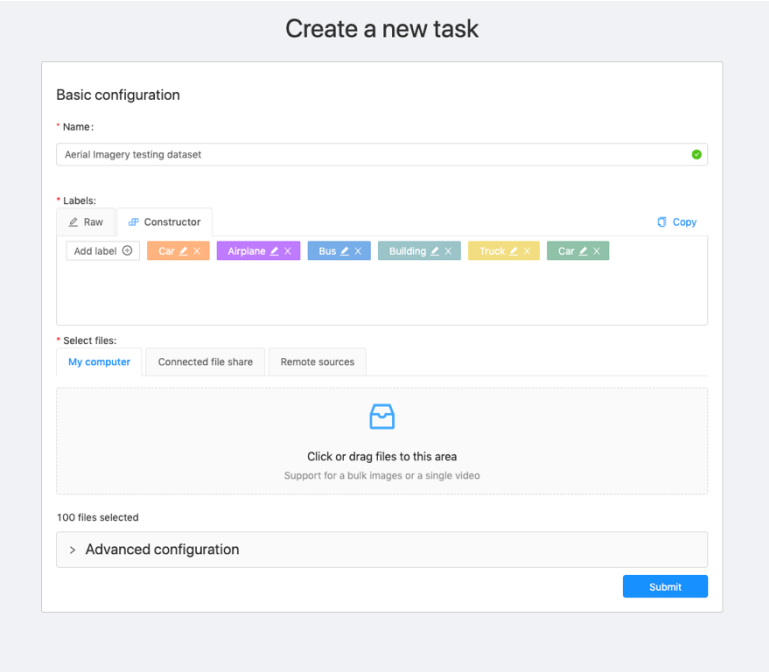

Figure 6. Create new task in CVAT [41].

Since the large tiff raster images are around 100 megabytes of size at the time of our experiment CVAT [41] is not able to handle opening such large images as we can see from the following Figure 7. Therefore, we decided to cut each aerial imagery of 10000x10000 pixel into smaller twenty-five slices each with 2000x2000 pixel [1].

[1] https://github.com/MajnunMan/python-image-slicer

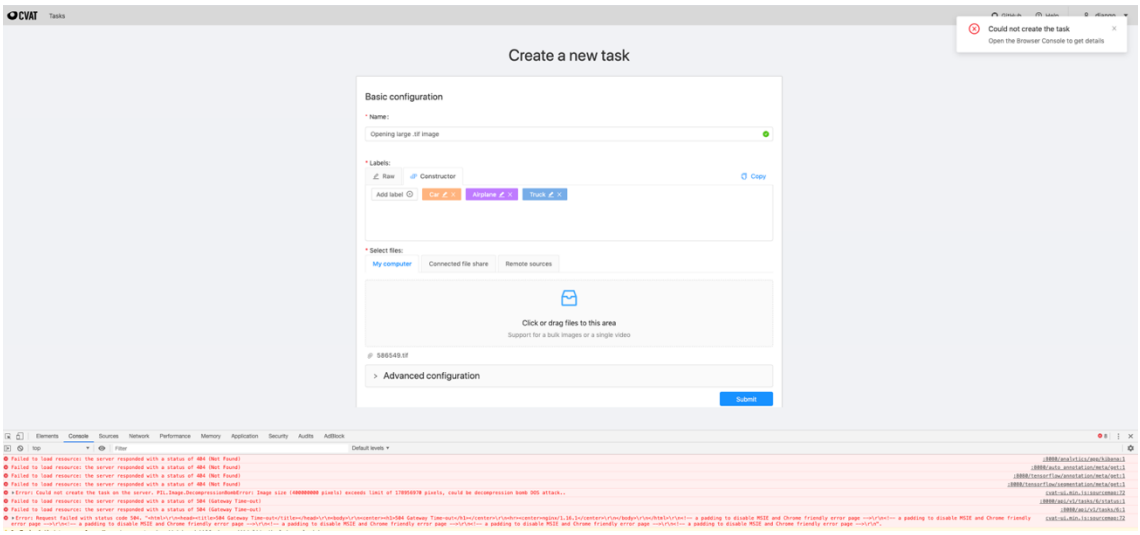

Figure 7. CVAT [41] cannot handle processing large aerial imagery

As a result, we converted our original twenty 10000x10000 pixel aerial imageries into smaller five hundred 2000x2000 images, thus having four hundred training dataset images and hundred testing dataset images.

Once we create our tasks, they appear in the CVAT home page as follows.

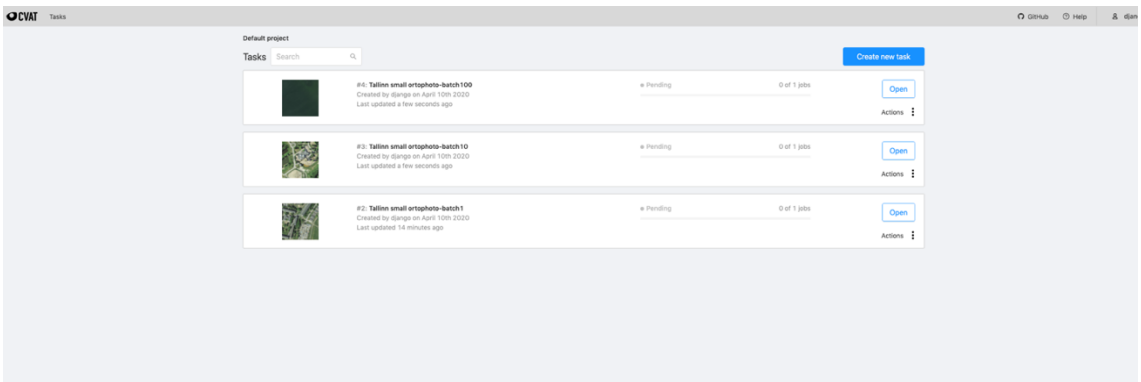

Figure 8. CVAT local server homepage

Once we set up, we can start annotating our images manually. In the following figure 9 we can see how the Task details page looks like.

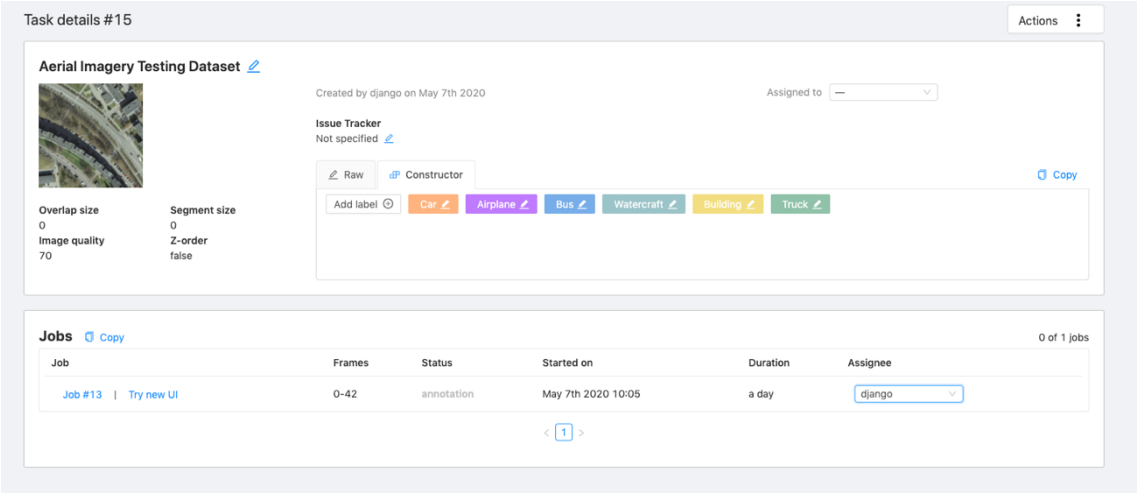

Figure 9. CVAT task details page

When we press the Job button it will take us to another page where we will actually annotate images Figure 10. We will annotate our images by drawing bounding boxes around each of object of interest and assign them their corresponding label.

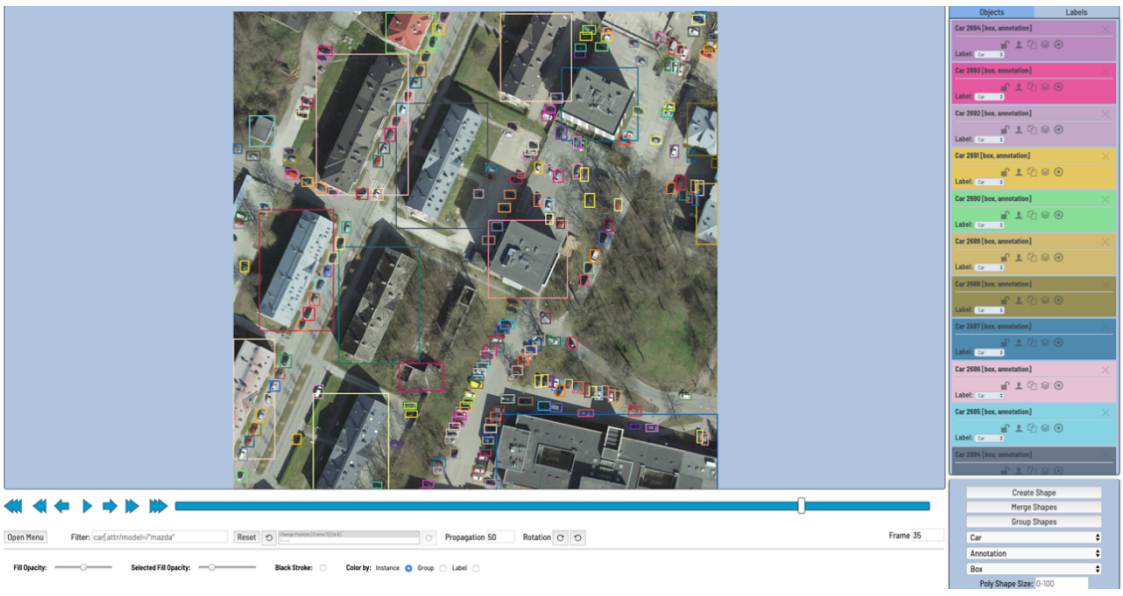

Figure 10. CVAT data annotation page

- Cars: We draw bounding boxes around the edges of each car even if they are not very clearly seen due to shadow in the image individually.

- Airplanes: Bounding boxes are drawn around the edges of every airplane individually. See Figure 11
- Buildings: This ground object of interest (GOI) is one of the challenging GOIs to draw a bounding box around. Because buildings are in various shapes in the imagery, for example top down (only roof, easy to draw bounding box around), from top and side, damaged. See Figure 12
- Trucks: Bounding boxes are drawn around the edges of every truck individually. See Figure 13
- Buses: As we mentioned previously in section 1.5 we consider public transport means of buses and trolleybuses as bus. Bounding boxes are drawn around the edges of every bus and trolleybus individually. See Figure 12
- Watercraft: We draw bounding boxes around the edges of each watercraft (ship, ferry etc.) individually. The imageries contain just a few such GOIs. See Figure 14

Figure 11. Annotated airplane, cars and roof-only building

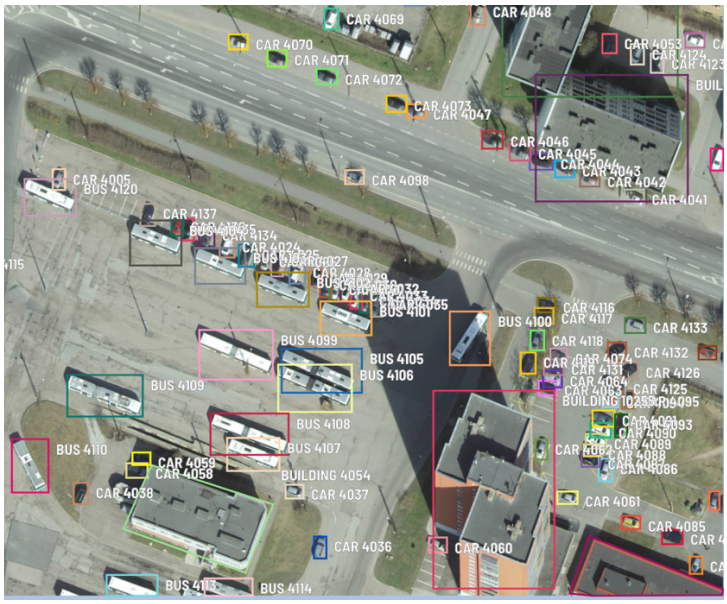

Figure 12. Annotated buildings (walls, roofs), cars and busses

Figure 13. Annotated cars and trucks

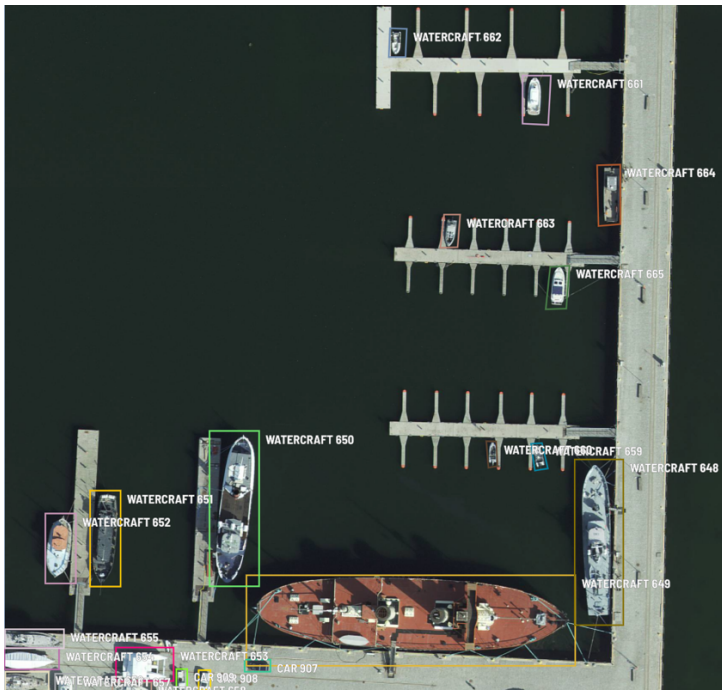

Figure 14. Annotated watercrafts

Once we annotate our image, we should save our work and dump the final annotation. There are various formats that we could dump our annotations such as COCO, PASCAL VOC and for our use case we will choose COCO. This is because we are feeding this data to training deep learning models from detectron2 model zoo **Error! Reference source not found.** and those models data input format is same as COCO data format.

To conclude data section, we would like to mention that despite we have high quality aerial imageries, sometimes even for humans it is difficult to recognize precisely every object of interest in the imagery. To give an example, as we can see from the following Figure 15, separating those buildings from each other is extremely complex task even for humans. Additionally, we notice the generic data imbalance problem in these aerial imageries as well, such that we have way many more cars than trucks or buses.

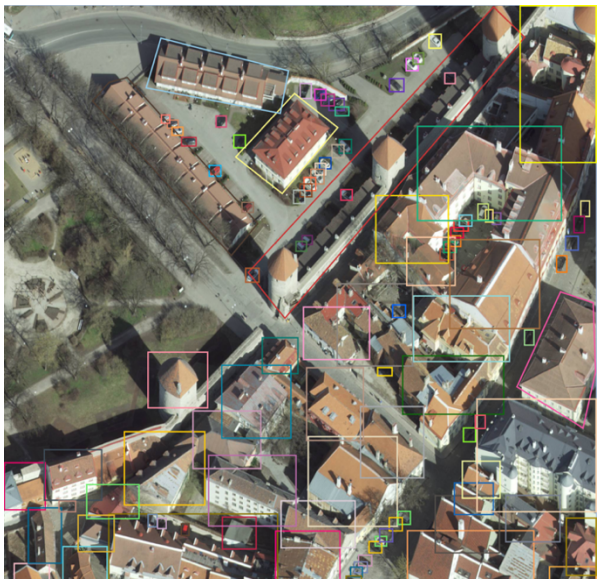

Figure 15. Complexities in aerial images

3.2 Network architectures

In this section we discuss the benchmarks of the state-of-the-art deep learning object detection models and the network architectures of those models that we use for our experiments.

Before proceeding with experiments a literature review was conducted and the state-of-the-art benchmarks were found as shown in the following Table 3 and Figure 16.

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage methods							
Faster R-CNN+++ [17]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [17]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN w G-RMI [61]	Inception-ResNet-v2 **Error! Reference source not found.**	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [57]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	52.1
Mask R-CNN w FPN [17]	Inception-101 FPN	44.3	62.3	43.4	22.1	43.2	51.2
One-stage methods							
YOLOv2 [5]	DarkNet-19 **Error! Reference source not found.**	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [18]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [59]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1

Table 3. Comparison the benchmarks of two-stage and one-stage detectors on COCO [36] dataset.

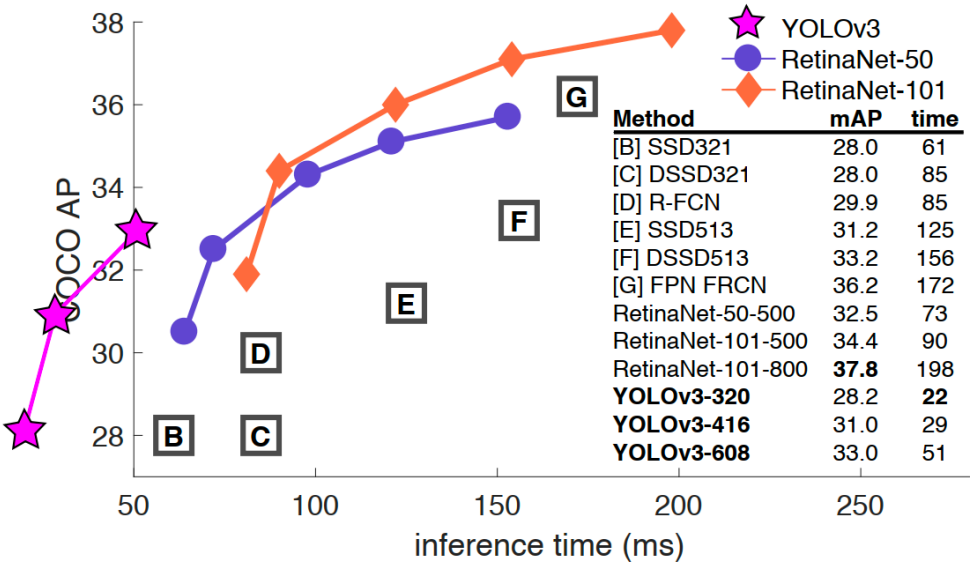

Figure 16. Comparison of the state-of-the-art object detection models. We adopt this figure from [26] and [24].

We adopt the above Table 3 and Figure 16 from [26] and [24] where the benchmarks of state-of-the-art one-stage and two-stage detectors are compared against each other. We can notice that two-stage detectors mostly outperform the one-stage detectors on accuracy, however one-stage detectors are faster. Since our research does not necessarily focus on real-time object detection in aerial imagery, we decided to use best performing two-stage detector algorithm Faster R-CNN [17] and best performing one-stage detector algorithm RetinaNet. Additionally, we also use Mask R-CNN [25] to experiment because of its high accuracy object instance segmentation results. There are two reasons why we are not using YOLO algorithm for our experiments is mainly due to very recently Pi et al. [48] reviewing YOLOv2 [5] deep learning model by applying transfer learning method as well as detetron2 [8] model zoo not having YOLO [15] models pre-trained.

3.2.1 Faster R-CNN

Recently Faster R-CNN is one of the most renowned deep neural network architectures. Faster R-CNN was proposed by Lin et al. [17]. Conceptionally, Faster R-CNN is composed of Feature Pyramid Network (FPN), Region Proposal Network (RPN) and Detection Networks.

Feature Pyramids are just basic components that detect objects at different scales. Prior to this work, Feature Pyramids were avoided to be used mostly due to their computational and memory cost. FPN is a top-down network architecture with lateral connections dev eloped for building high-level semantic feature maps at all scales [17].

The RPN is used to generate bounding boxes called Region of Interests (RoI) that have high probability of containing of any object of interest. Detection Network on the other hand, takes input from both FPN and RPN and performs the final object detection task, meaning that identifying the object of interest class and the bounding box of it.

3.2.2 Masked R-CNN

Mask R-CNN is one of the renowned deep learning models that performs multi tasks such as object detection, instance segmentation and person key point detection [25]. In principle, Mask R-CNN [25] extends Faster-RCNN [17] by adding a branch for predicting segmentation masks on each Region of Interest (RoI), in parallel with existing branch for classification and bounding box regression. The mask branch is a small FCN applied to each RoI, predicting a segmentation mask in a pixel-to-pixel manner. Mask R-CNN is slightly computational expensive just because mask branch only adds a small computational overhead, enabling a fast system and rapid experimentation.

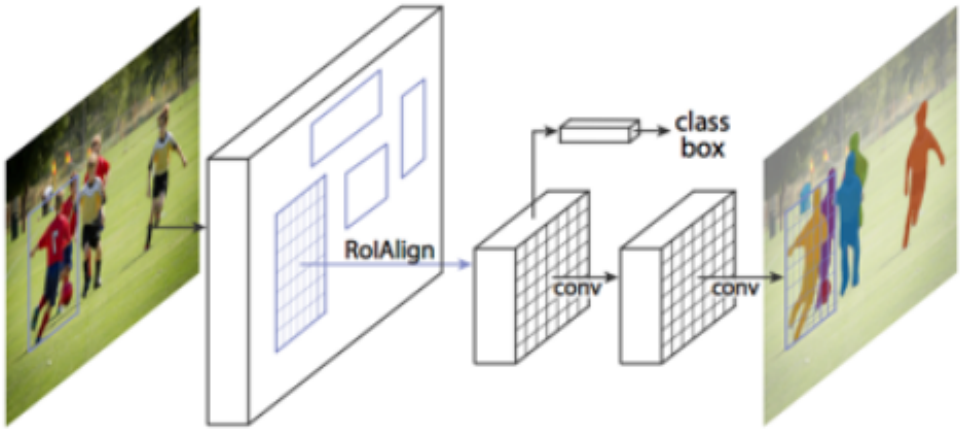

Figure 17. Mask R-CNN framework for Instance Segmentation [25].

3.2.3 Retina Net

As we can see from the Figure 16 single stage detectors have a major insufficiency when it comes to the model accuracy. RetinaNet was proposed by Ross et al. [26] in order to increase the accuracy by making two major improvements over other single stage detectors. In this network architecture there were two major improvements over other single stage objection detection models (e.g. YOLO [15], SSD [18]). Feature Pyramid Networks for Object Detection [49] and Focal Loss for Object Detection [26].

Feature Pyramid networks have been used typically to recognize objects of interest at distinctive scales. A Feature Pyramid Network (FPN) [49] augments CNNs with a top-down pathway and lateral connections so the network efficiently constructs multi-scale pyramid from a single resolution input image (see Figure 18).

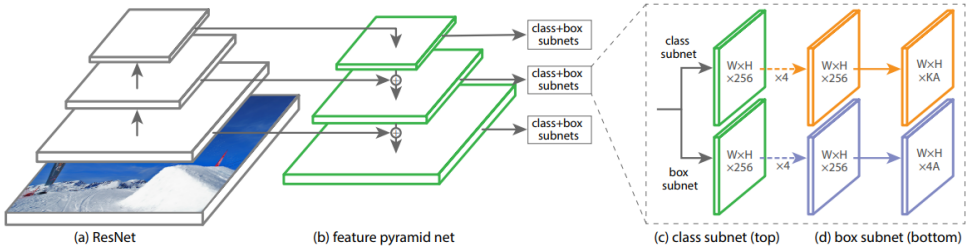

Figure 18. RetinaNet uses a Feature Pyramid Network (FPN) as a backbone on top of a feedforward ResNet architecture [50].

The one-stage RetinaNet network architecture uses a Feature Pyramid Network (FPN) [17] as a backbone on top of a feedforward ResNet architecture [50] (a) to generate a rich, multi-scale convolutional feature pyramid (b). To this backbone RetinaNet attaches two subnetworks, one for classifying anchor boxes (c) and one for regressing from anchor boxes to ground-truth object boxes (d). The network design is intentionally simple, which enables this work to focus on a novel focal loss function that eliminates the accuracy gap between our one-stage detector and state-of-the-art two-stage detectors like Faster R-CNN with FPN while running at faster speeds. Each level of the pyramid can be used for detecting objects at a different scale. FPN improves multi-scale predictions from fully convolutional networks (FCN) [51], as shown by its gains for RPN and DeepMask-style

proposals [51], as well at two-stage detectors such as Fast R-CNN [31] or Mask R-CNN [25].

Focal Loss (see Figure 19) is a more effective alternative to Cross Entropy Loss (CE) which is introduced by Ross et al.**Error! Reference source not found.** [26] to tackle the class imbalance challenge with single stage detection approach. Single stage object detection models usually suffer from foreground and background class imbalance problem mainly because of dense sampling of anchor boxes [26]. Initially, RetinaNet allows thousands of anchor boxes at each pyramid layer, and only a few of them is assigned to ground-truth object in the end, while the vast majority represents background class. Despite detections with high probabilities resulting in smaller loss values. Focal Loss is used to reduce loss contribution when there are easy examples (detections with high confidence scores) and to increase the significance of correcting misclassified examples. Moreover, Focus Loss (FL) is define as following equation (2).

$$FL(p_t) = -\alpha_t(1-p_t)^{\gamma}\log(p_t) \qquad (2)$$

Focal Loss introduces an additional modulating factor $(1-p_t)^{\gamma}$ to the cross-entropy loss [53], with tunable focusing parameter $\gamma \geq 0$.

3.3 Implementation details

For the experimental setup we used Google Collaboratory Professional powered by a Nvidia Tesla P100 GPU with 30GB of random amount memory as our training environment (see Figure 19). We are inheriting the above described pre-trained models from detectron2 model zoo [8]. Detectron2 model zoo offers several pre-trained models that are trained on COCO [36] and LVIS [52] datasets. The training dataset contains 236 images and three test datasets each contains 43 images (see 3.1). One test dataset is original images, while the other two are down sampled versions of it. We conducted a lot of experiments with the above described models mostly in order to fine-tune the hyperparameters. In the end we decided to evaluate the experimentation results of the top

best performing model per each algorithm, namely Faster R-CNN [17], Mask R-CNN [25] and RetinaNet [26].

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 440.82       Driver Version: 418.67       CUDA Version: 10.1     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla P100-PCIE...  Off  | 00000000:00:04.0 Off |                    0 |
| N/A   42C    P0    29W / 250W |      0MiB / 16280MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
```

Figure 19. Detailed training environment description.

The deep neural networks were trained with the hyperparameters of learning rate being $5*10^{-4}$ and the batch size being 4. To tune hyperparameters we followed an approach taken by [53], where we conducted additional experiments with different values of learning rate. Initially we started with values such as 10^{-3}, $25*10^{-3}$, $2*10^{-2,}$ however in the end we figure out that $5*10^{-4}$ was yielding relatively better results for all three models. Additionally, the mini-batch size was another important hyperparameter that we paid attention to. Since we had merely 236 training images, we split them into 59 batches each containing 4 training samples, thus we trained al three models 10000 iterations. In principle, we had limited computational power, therefore we did not manage to train longer than 170 epochs. Perhaps, if we trained longer than 170 epochs, we would get slightly better results (1-3%), although it was not in the scope of this thesis research.

To start training our neural networks, we initialize the weights and biases parameters by loading these parameters from detectron2 model zoo [8] where the parameters of the pre-trained models are stored. We apply second approach of the transfer learning as described above in section 2.4 which is copying first n layer parameters from base pre-trained network to target network. In our case n is the number of layers our neural network architectures have. For all three object detection algorithms we use 101 layered backbone configurations. Since aerial imageries are completely different than the data in COCO [36] and LVIS [53] datasets, we decided to initialize the parameters of our neural networks with the pre-learned parameters and to re-train the whole network.

4 Results

4.1 Evaluation metrics

In this section first we briefly discuss the evaluation metrics that are used to measure the performance of our models, then demonstrate our results, compare them against to state-of-the-art baselines and finally analyze the key factors. Our metrics are average precision (AP), average recall (AR), alongside we consider inference time and training time metrics too.

Object detection evaluation metrics measure to assess how well the model performs on an object detection task. When it comes to evaluating the performance of model in object detection task there is not much of a difference with classification task. In simple classification task in order to calculate the evaluation metrics such as recall, precision or accuracy we need to know about true positive, false positive, true negative and false negative values.

Likewise, in classification task, for object detection task we consider the true positive result when the model correctly draws the bounding box around the corresponding object of interest. A false positive result is when the model detects the wrong object as our own object of interest.

In order to measure how accurate are the detections of our model, we use a metrics called Precision [54] and to measure how good our model detects the objects of interest we use a metrics called Recall [54].

In order to calculate precision and recall metrics we need to have the values for true positive, false positive and false negative values, and these values are based on the metrics called intersection over union (IoU) or sometimes referred as Jaccard Index [55] (see Figure 20). IoU metrics as its name implies computes intersection of the ground truth and predicted bounding boxes of the object of interest over union of the ground truth and predicted bounding boxes of the object of interest. See below equation (1).

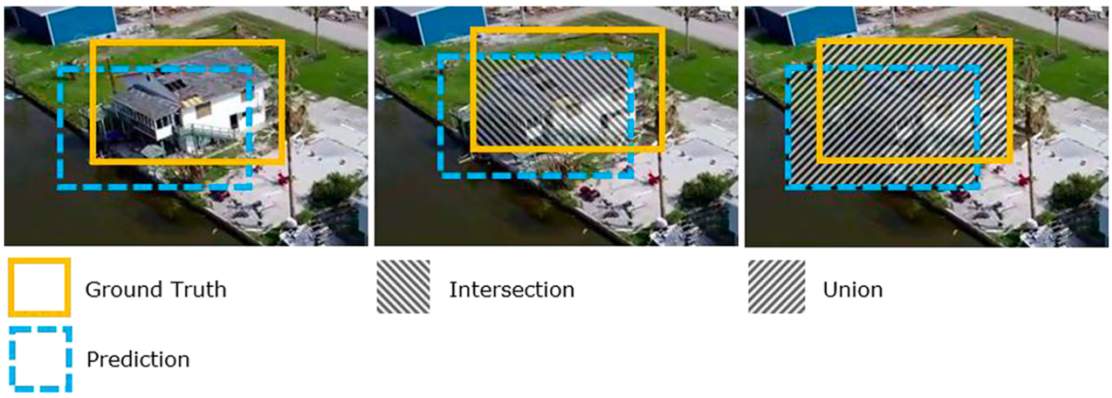

Figure 20. Example of Intersection over Union in aerial object detection [4].

$$Intersection\ over\ Union\ (IOU) = \frac{Intersection\ Area}{Union\ Area} \quad (1)$$

In theory the best object detection result is when we have ground truth and predicted bounding boxes overlap in other words when IoU is 100%. Given limitations of existing object detection algorithms currently it is nearly impossible to have IoU being 100%, thus both in the industry and research community IoU value of 50 to 95% is used.

In this research, taking complexity aerial scenery into consideration, especially due to small objects of interest such as cars, the object detection is deemed true positive (TP) when IoU ≥ 25%. Therefore, we set the threshold for our Jaccard index or IoU as 0.25.

Given the threshold of IoU we classify the object detection result as

- True Positive (TP), when IoU > 0.25.
- False Positive (FP) when IoU < 0.25.
- False Negative (FN), when ground truth object is present in the image, but our model failed to detect it.

Having the above metrics now we can calculate the precision and recall for our model performance.

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \quad (2)$$

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \quad (3)$$

We then calculate the cumulative precision (see Equation 2) and recall (see Equation 3) values for each class and find the are under the precision-recall curve as average precision (AP) for each class respectively. Our model performance evaluation results will be in the format of COCO [36] dataset evaluation format. This is due to us using pre-trained models from detectron2 model zoo (as it is by default supported evaluator type) as well as recently most researchers tend to give results on this data format.

4.2 Analysis and Discussion

Overall, we created three models altogether by feeding Dataset10 to Faster R-CNN [17], Mask R-CNN [25] and RetinaNet [26] algorithms. We chose the top performing backbone network architectures for each algorithm (see Small sized objects, when the pixel-wise area of the object of interest is smaller than 32x32 pixels.

- Medium sized objects, when the pixel-wise area of the object of interest is in the range of 32x32 pixels and 96x96 pixels.

- Large sized objects, when the pixel-wise area of the object of interest is in greater than 96x96 pixels.

Table 4). In general, for all three models the results are good and quite close to the benchmark scores. Overall, we evaluated the performance of the models on our three test datasets that are described in section 3.1 . In this section we will analyze our results on main test dataset and conduct a comprehensive comparison with benchmark scores and previous works.

The training process follows the steps described earlier in section 3.3 and the result metrics are collected in COCO [36] evaluation format. In COCO [36] evaluation format the objects are categorized into three scales in terms of their pixel-wise areas in the given image. The objects are considered as

- Small sized objects, when the pixel-wise area of the object of interest is smaller than 32x32 pixels.
- Medium sized objects, when the pixel-wise area of the object of interest is in the range of 32x32 pixels and 96x96 pixels.
- Large sized objects, when the pixel-wise area of the object of interest is in greater than 96x96 pixels.

Table 4. Average precision results of re-trained deep learning models on the Dataset10.

Method	***Backbone***	***AP***	***AP_{50}***	***AP_{75}***	***AP_S***	***AP_M***	***AP_L***
Faster R-CNN	Inception-101 FPN	29	55.4	27.9	18.7	25	48.4
Mask R-CNN	Inception-101 FPN	**33.2**	**56.4**	**32.3**	**21.2**	**27.1**	**52.4**
RetinaNet	ResNet-101 FPN	26.3	51.9	23.8	5.5	24.9	44.9

Table 5. Average precision results of re-trained deep learning models on the Dataset20.

Method	***Backbone***	***AP***	***AP_{50}***	***AP_{75}***	***AP_S***	***AP_M***	***AP_L***
Faster R-CNN	Inception-101 FPN	28.4	51.6	27.3	18.7	27.3	43.6
Mask R-CNN	Inception-101 FPN	**33.2**	**58.5**	**33.5**	**21.2**	27.2	**61.9**
RetinaNet	ResNet-101 FPN	26.1	51.4	22.8	5.7	**28**	48.3

Table 6. Average precision results of re-trained deep learning models on the Dataset40.

Method	***Backbone***	***AP***	***AP_{50}***	***AP_{75}***	***AP_S***	***AP_M***	***AP_L***
Faster R-CNN	Inception-101 FPN	**28.1**	**50.9**	26.4	**22.6**	**50**	52.9
Mask R-CNN	Inception-101 FPN	25.2	46.3	**35.1**	20.5	46.3	55
RetinaNet	ResNet-101 FPN	24.5	47	22.8	18.4	44.1	**59.6**

When analyzing results from above tables 4, 5 and 6 it appears that generally over all three datasets Mask R-CNN outperforms the other two models. Additionally, it can be clearly seen that two stage detectors namely Mask R-CNN and Faster R-CNN outperform one stage detector namely RetinaNet in all the three datasets. Moreover, we can say that even the ground sampling distance is higher on the Dataset20 and Dataset40, we do not notice significant drops in average precision scores. Hence the models trained on Dataset10 is able to perform decently on Dataset20 and Dataset40. When comparing the average precision results of our re-trained models on our aerial imagery custom datasets versus the benchmarks of the same models on COCO dataset, we observe that Mask R-CNN performs more closely to its corresponding benchmark. Average precision scores comparing the other two models (see Table 3). Nevertheless, Faster R-CNN and RetinaNet scores better average precision than YOLOv2 [5] does on COCO dataset.

Table 7. Granular AP results of re-trained deep learning models by each class on Dataset10.

Method	*Backbone*	*Car*	*Airplane*	*Bus*	*Watercraft*	*Building*	*Truck*
Faster R-CNN	Inception-101 FPN	**39.2**	28.5	53.2	11.8	47.4	18.1
Mask R-CNN	Inception-101 FPN	30.3	**51.2**	**55.4**	**12**	**49.5**	17.5
RetinaNet	ResNet-101-FPN	12.9	26.8	44.6	6.3	46.5	**23.7**

Table 8. Granular AP results of re-trained deep learning models by each class on Dataset20.

Method	*Backbone*	*Car*	*Airplane*	*Bus*	*Watercraft*	*Building*	*Truck*
Faster R-CNN	Inception-101 FPN	**38.8**	25.4	50.2	11.8	47.4	19.7
Mask R-CNN	Inception-101 FPN	30.3	**51.2**	**59.6**	**12.4**	**49.6**	18.1
RetinaNet	ResNet-101-FPN	13.1	30	41.3	7.7	46.7	**20.9**

Table 9. Granular AP results of re-trained deep learning models by each class on Dataset40.

Method	*Backbone*	*Car*	*Airplane*	*Bus*	*Watercraft*	*Building*	*Truck*
Faster R-CNN	Inception-101 FPN	**28.1**	**25.3**	**48.3**	**11.8**	**47.4**	**22.1**
Mask R-CNN	Inception-101 FPN	26	16.9	44.2	10.2	42.6	11.4
RetinaNet	ResNet-101-FPN	12.9	23.6	38.4	5.7	46.5	20.9

The above tables 7, 8 and 9 illustrate that all three models struggle to detect trucks and watercrafts, and this can be explained by the fact that the training dataset had very few samples of each category. Although the training dataset had relatively high number of car instances, RetinaNet does not manage to distinguish the features of the cars well enough. Faster R-CNN outperforms other two models to distinguish the features of car. All three models show similar performances to detect building instances, whereas Mask R-CNN

performs better than the other two models to detect bus instances. Additionally, Mask R-CNN manages to extract features of airplanes with higher precision on Dataset10 and Dataset20 but shows poor performance on Dataset40.

Table 10. AR for a given maximum number of detections where IoU is in range of *0.5: 0.95.* Dataset10.

Method	***Backbone***	$AR^{max=1}$	$AR^{max=10}$	$AR^{max=100}$	AR_S	AR_M	AR_L
Faster R-CNN	Inception-101 FPN	**10.7**	22.8	35.5	**19.8**	30.9	51.8
Mask R-CNN	Inception-101 FPN	9.4	**24.9**	**36.6**	15	29.7	**55.6**
RetinaNet	ResNet-101 FPN	10.3	23.3	33.5	6.8	**31.6**	49.2

Table 11. AR for a given maximum number of detections where IoU is in range of *0.5: 0.95.* Dataset20.

Method	***Backbone***	$AR^{max=1}$	$AR^{max=10}$	$AR^{max=100}$	AR_S	AR_M	AR_L
Faster R-CNN	Inception-101 FPN	**10.6**	22.5	34	19.8	31.7	53.5
Mask R-CNN	Inception-101 FPN	9.4	**24.9**	**36.6**	**21**	29.7	**65**
RetinaNet	ResNet-101 FPN	10.3	23.2	33.6	11	**34.3**	53.5

Table 12. AR for a given maximum number of detections where IoU is in range of *0.5: 0.95.* Dataset40.

Method	***Backbone***	$AR^{max=1}$	$AR^{max=10}$	$AR^{max=100}$	AR_S	AR_M	AR_L
Faster R-CNN	Inception-101 FPN	10	**22.2**	**33.6**	**27.8**	**54.9**	57.9
Mask R-CNN	Inception-101 FPN	9.2	21.2	32.2	27.7	50.1	61.9
RetinaNet	ResNet-101 FPN	**10.3**	21.5	31.8	24.4	51	**67.4**

Table 10, 11 and 12 show the average recall results for a given maximum number of detections respectively on the Dataset10, Dataset20, Dataset40. It can be seen that, RetinaNet and Faster R-CNN are in the race against each other, while Mask R-CNN seems to accept the defeat in this metric. RetinaNet outperforms two stage detectors in this metric with tiny difference. I think it is wiser to take $AR^{max=10}$ and $AR^{max=100}$ metric more seriously, because majority of the images in the dataset (at least ~80%) contain more than 10 objects of interest. Since the images are aerial there is so much data as it can be seen from Figure 12. One significant observation that we can do is, on Dataset40 all three models displays roughly same performance, whereas on Dataset20 and Dataset10 they differ a lot. RetinaNet seems to miss many small objects of interest on Dataset10 and Dataset20.

Additionally, besides evaluating the performance of the models by north star average precision and average recall metrics we also evaluate the results visually. This is not quantifiable way of evaluation, but more of an intuitive way of evaluation. Below, we display the visual performance of the three models on the very same image in three different versions. First three results (Figure 21, Figure 22, Figure 23) reflect the performances of three models respectively on the Dataset10. Following, the next three results (Figure 24, Figure 25, Figure 26) are retrieved by running all the three models on an arbitrary image from Dataset20. Finally the last three results (Figure 27, Figure 28, Figure 29) indicate the visual performances of the models on the same image but from Dataset40.

Figure 21.Faster R-CNN prediction on an arbitrary image from Dataset10.

Figure 22. Mask R-CNN prediction on an arbitrary image from Dataset10.

Figure 23. RetinaNet prediction on an arbitrary image from Dataset10.

Figure 24. Faster R-CNN prediction on an arbitrary image from Dataset20.

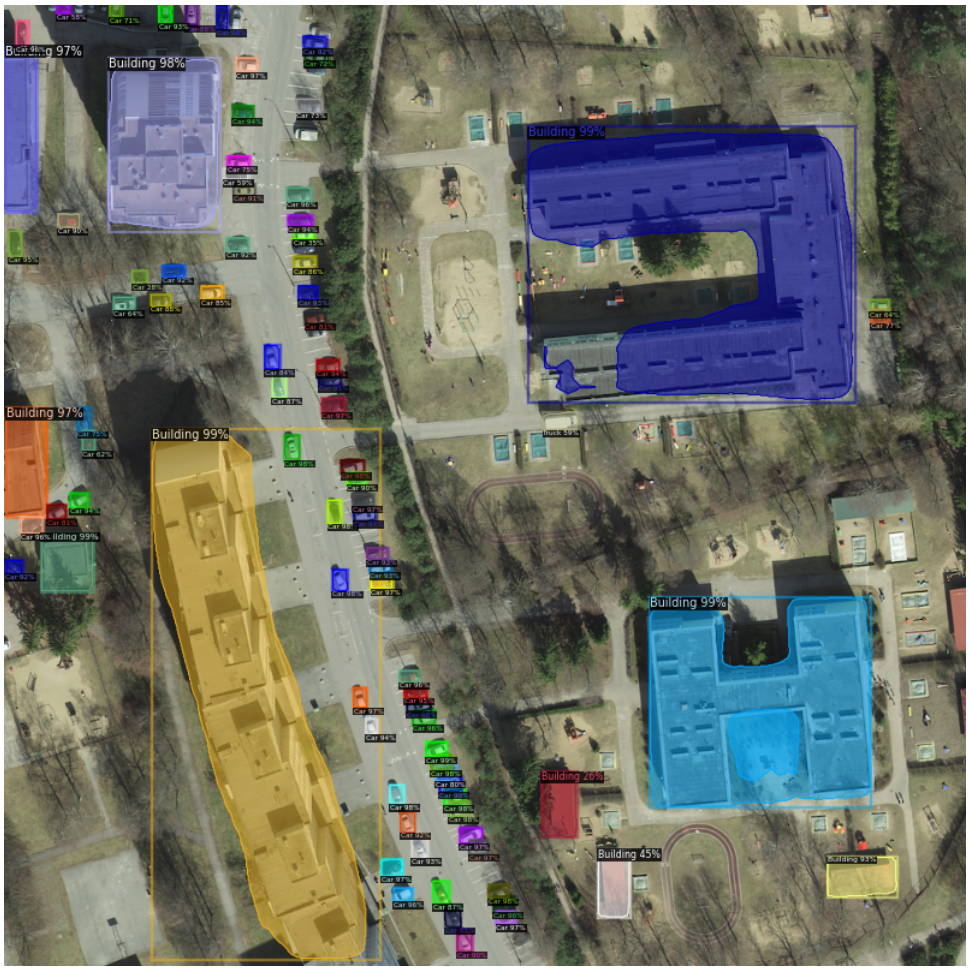

Figure 25. Mask R-CNN prediction on an arbitrary image from Dataset20.

Figure 26. RetinaNet prediction on an arbitrary image from Dataset20.

Figure 27. Faster R-CNN prediction on an arbitrary image from Dataset40.

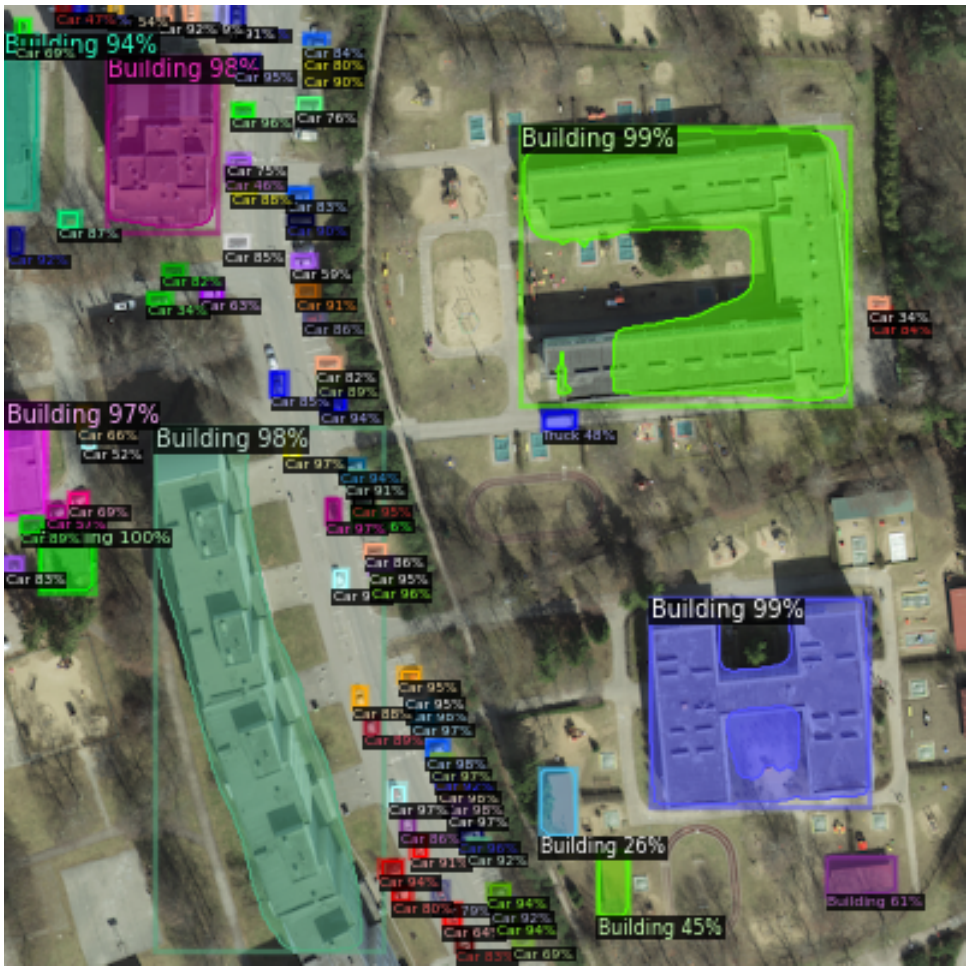

Figure 28. Mask R-CNN prediction on an arbitrary image from Dataset40.

Figure 29. RetinaNet prediction on an arbitrary image from Dataset40.

To conclude this section, we can say that Faster R-CNN displays the most robust performance for detecting smaller objects while Mask R-CNN gives the best performance in high resolution aerial imagery where objects have medium or large size, in other words their pixel-wise areas are greater than at least 32x32 pixels.

4.2.1 Key Factors

It turns out that there are several factors that eventually affect the performance of the models. Obviously, tuning the hyperparameters finely is very essential to get significantly decent results. We started learning rate with 10^{-3}, then ended up with $5*10^{-4}$ which seemed to perform better than other values. The number of batches was also essential hyperparameter to tune for us, initially we started with 2 images per batch (we had 236 training images (see section 3.1) which was not converging the loss we enough. At the end it turned out when we include 4 images per batch, the model started to give better results. One other key factor was the dataset characteristics which is quite visible with

results displayed above tables (see Small sized objects, when the pixel-wise area of the object of interest is smaller than 32x32 pixels.

- Medium sized objects, when the pixel-wise area of the object of interest is in the range of 32x32 pixels and 96x96 pixels.

- Large sized objects, when the pixel-wise area of the object of interest is in greater than 96x96 pixels.

Table 4 and Table 7). The dataset was so imbalanced, this was something that we could not do much about, because the imageries naturally contain more cars and buildings than other objects of interest.

4.2.2 Future Opportunities

In this work we concentrated our research on evaluating two top performing state-of-the-art two-stage detectors and one top state-of-the-art one-stage detector for object detection from aerial imageries. One of the lessons we learned was creating a balanced dataset. This is a known challenge in the field although, we can mitigate the class imbalanced distribution by increasing distribution of some of the classes such as trucks, busses by doing data augmentation. We think that the same evaluation can be done for real-time video analysis for future work.

5 Conclusion

In this research we facilitated objects of interest detection in aerial imagery by applying transfer learning from pre-trained state-of-the-art CNN models. To achieve this, a list of objects of interest were generated by interviewing industrial companies. Following, in order to feed the CNN an in-house high-resolution aerial imagery dataset was introduced by collecting the data from Land Board of Estonian Republic. Totally, three CNN models were trained, validated and tested on several different versions of the original test datasets, whereby down sampling the images ground sampling distance as if the imagery has higher viewpoint altitude. The results for the objects of interest that have medium and large pixel-wise areas were quite satisfactory, in fact nearly beating corresponding benchmark values on COCO dataset. In general, if objects of interest in a given image has greater are than 32x32 pixels, our models lead to reasonably reliable results. The first conclusion that we can draw is that, by applying transfer learning and feeding the deep neural networks with high resolution aerial imagery can yield to decent results that are pretty close to state-of-the-art benchmarks. Additionally, Mask R-CNN slightly outperforms the other two models (Faster R-CNN, RetinaNet) in object detection task from high resolution aerial imagery. To conclude, the all three models that were re-trained on high-resolution aerial imagery (Dataset10) seem to perform almost at the same level in 2- and 4-times lower resolution aerial imagery.

6 References

[1] T. N. Mundhenk, G. Konjevod, W. A. Sakla and K. Boakye, "A Large Contextual Dataset for Classification, Detection and Counting of Cars with Deep Learning," *ArXiv,* 2016.

[2] H. Long, Y. Chung, Z. Liu and S. Bu, "Object Detection in Aerial Images Using Feature Fusion Deep Networks," *IEEE Access,* vol. 7, pp. 30980-30990, 2019.

[3] F. Kamran, M. Shahzad and F. Shafait, "Automated Military Vehicle Detection from Low-Altitude Aerial Images," in *Digital Image Computing: Techniques and Applications (DICTA)*, Canberra, Australia, IEEE, 2018, pp. 1-8.

[4] Y. Pi, N. D. Nath and A. H. Behzadan, "Convolutional neural networks for object detection in aerial imagery for disaster response and recovery," *Advanced Engineering Informatics,* vol. 43, 2020.

[5] A. Farhadi and J. Redmon, "Yolo9000: better, faster, stronger," *arXiv,* vol. 1612.08242, 2016.

[6] D. Cisek, M. Mahajan, J. Dale, S. Pepper, Y. Lin and S. Yoo, "Transfer Learning approach to parking lot classification in aerial imagery," in *Conference: 2017 New York Scientific Data Summit (NYSDS)*, New York, 2017.

[7] K. Alex, I. Sutskeve and G. o. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *NIPS*, 2012.

[8] R. Girshick, Y. Wu, A. Kirillov, F. Mass and W.-Y. Lo, "Detectron2," 2019. [Online]. Available: https://github.com/facebookresearch/detectron2. [Accessed 03 2020].

[9] Wikipedia, "Empirical research," [Online]. Available: https://en.wikipedia.org/wiki/Empirical_research. [Accessed 29 04 2020].

[10] G. Maaamet, "Repulic of Estonia Land Board," [Online]. Available: https://geoportaal.maaamet.ee/index.php?lang_id=2&page_id=662.

[11] Google, "Google Colab," [Online]. Available: https://colab.research.google.com/. [Accessed 04 2020].

[12] S. M. Beitzel, E. C. Jensen and O. Frieder, "MAP," in *Encyclopedia of Database Systems*, Springer US, 2009, pp. 1691--1692.

[13] C. Baker, S. Ramchurn, W. Teacy and N. Jennings, "Planning search and rescue missions for UAV teams," in *Proceedings of the Twenty-second European Conference on Artificial Intelligence*, Netherlands, 2016.

[14] M. Radovic, O. Adarkwa and Q. Wang, "Object recognition in aerial images using convolutional neural networks," *J. Imaging,* pp. 3-21, 2017.

[15] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You only look once: Unified, real-time object detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, 2016.

[16] S. Han, W. Shen and Z. Liu, "Deep Drone : Object Detection and Tracking for Smart Drones on Embedded System," 2016.

[17] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 39, pp. 1137-1149, 2017.

[18] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu and A. C. Berg, "Ssd: Single shot multibox detector," *Computer Vision ECCV,* vol. 9905, pp. 21-37, 2016..

[19] J. Dai, Y. Li, K. He and J. Sun, "Object Detection via Region based Fully Convolutional Networks," *NIPS,* pp. 379-387., 2016..

[20] P. Narayanan, C. Borel-Donohue, H. Lee, H. Kwon and R. Rao, "A real-time object detection framework for aerial imagery using deep neural networks and synthetic training images,," in *Signal Processing, Sensor/Information Fusion, and Target Recognition XXVII*, vol. 10646, SPIE, 2018, pp. 25-33.

[21] E. Guirado, S. Tabik, M. Rivas, D. Alcaraz-Segura and F. Herrera, "Automatic whale counting in satellite images with deep learning," *bioRxiv,* 2018.

[22] F. Yang, H. Fan, P. Chu, E. Blasch and H. Ling, "Clustered Object Detection in Aerial Images," in *IEEE/CVF International Conference on Computer Vision (ICCV): 8310-8319.*, 2019.

[23] J. Uus and T. Krilavičius, "Detection of different types of vehicles from aerial imagery," vol. 3, 2019.

[24] A. F. Joseph Redmon, "YOLOv3: An Incremental Improvement," *ArXiv,* vol. abs/1804.02767, 2018.

[25] K. He, G. Gkioxari, P. Dollár and R. Girshick, "Mask R-CNN," in *IEEE International Conference on Computer Vision (ICCV)*, Venice, 2017.

[26] T. Lin, P. Goyal, R. Girshick, K. He and P. Dollár, "Focal Loss for Dense Object Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 42, pp. 318-327, 2020,.

[27] Z. Zhao, P. Zheng, S. Xu and X. Wu, "Object Detection With Deep Learning: A Review," *IEEE Transactions on Neural Networks and Learning Systems,* vol. 30, no. 11, pp. 3212-3232, 2019..

[28] W. P. a. W. S. McCulloch, "How we know universals the perception of auditory and visual forms," *The Bulletin of Mathematical Biophysics,* vol. 9, no. 3, p. 127–147, 1947 .

[29] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning representations by back-propagating errors.," *Nature,* vol. 323, pp. 533-536, 1986.

[30] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *CVPR*, 2014.

[31] R. Girshick, "Fast r-cnn," in *ICCV*, 2015.

[32] X. Ren and D. Ramanan, "Histograms of sparse codes for object detection," in *CVPR*, 2013.

[33] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *ArXiv,* vol. abs/1502.03167, 2015.

[34] D. Erhan, C. Szegedy, A. Toshev and D. Anguelov, "Scalable object detection using deep neural networks," in *CVPR*, 2014.

[35] J. Yosinski, J. Clune, Y. Bengio and H. Lipson, "How transferable are features in deep neural networks?," in *NIPS*, 2014.

[36] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick and P. Dollár, "Microsoft COCO: Common Objects in Context," *ArXiv,* vol. abs/1405.0312, 2014.

[37] Wikipedia, "Softmax function," [Online]. Available: https://en.wikipedia.org/wiki/Softmax_function. [Accessed 29 04 2020].

[38] Wikipedia, "Ground sample distance," [Online]. Available: https://en.wikipedia.org/wiki/Ground_sample_distance. [Accessed 10 05 2020].

[39] M. O. i. Geoportal, "Geoportal Maaamet," [Online]. Available: https://geoportaal.maaamet.ee/data/files/Leica_ADS100.pdf?t=20160620143616. [Accessed 12 04 2020].

[40] QGIS, "Open Source Geographic Information System," [Online]. Available: https://qgis.org/en/site/. [Accessed 03 2020].

[41] Github, "Computer vision annotation tool," [Online]. Available: https://github.com/opencv/cvat. [Accessed 10 03 2020].

[42] Wikipedia, "Lightweight Directory Access Protocol," [Online]. Available: https://en.wikipedia.org/wiki/Lightweight_Directory_Access_Protocol. [Accessed 10 03 2020].

[43] UX, "User experience," [Online]. Available: https://en.wikipedia.org/wiki/User_experience. [Accessed 04 05 2020].

[44] UI, "User interface," [Online]. Available: https://en.wikipedia.org/wiki/User_interface. [Accessed 04 05 2020].

[45] Wikipedia, "Docker," [Online]. Available: https://en.wikipedia.org/wiki/Docker_(software). [Accessed 21 04 2020].

[46] Wikipedia, "PostgreSQL," [Online]. Available: https://en.wikipedia.org/wiki/PostgreSQL. [Accessed 12 04 2020].

[47] Wikipedia, "Redis," [Online]. Available: https://en.wikipedia.org/wiki/Redis. [Accessed 18 04 2020].

[48] Y. Pi, N. D. Nath and A. H. Behzadan, "Convolutional neural networks for object detection in aerial imagery for disaster response and recovery," *Advanced Engineering Informatics,* vol. 43, no. 1474-0346, 2020.

[49] T.-Y. Lin, P. Dollár, R. Girshick, K. He and B. Hariharan, "Feature Pyramid Networks for Object Detection," *arXiv,* vol. 1612.03144, 2016.

[50] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, 2016, pp. 770-778.

[51] P. O. Pinheiro, R. Collobert and P. Dollar, "Learning to Segment Object Candidates.," in *NIPS*, 2015.

[52] A. Gupta, P. Dollár and R. Girshick, "LVIS: A Dataset for Large Vocabulary Instance Segmentation," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA, 2019.

[53] A. Adel, K. Anis, A. Mohanned and S. Abdulrahman, "Aerial Images Processing for Car Detection using Convolutional Neural Networks: Comparison between Faster R-CNN and YoloV3," *arXiv,* 2020.

[54] Wikipedia, "Precision_and_recall," [Online]. Available: https://en.wikipedia.org/wiki/Precision_and_recall. [Accessed 23 04 2020].

[55] Wikipedia, "Jaccard_index," [Online]. Available: https://en.wikipedia.org/wiki/Jaccard_index. [Accessed 21 04 2020].

[56] A. F. Joseph Redmo, "YOLOv3: An Incremental Improvement," *ArXiv,* vol. abs/1804.02767, 2018.

[57] A. Shrivastava, R. Sukthankar, J. Malik and A. Gupta, "Beyond skip connections: Top-down modulation for object detection," *arXiv,* vol. 1612.06851, 2016.

[58] C. Szegedy, S. Ioffe, V. Vanhoucke and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," *ArXiv,* vol. abs/1602.07261, 2017.

[59] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi and A. C. Berg, "Dssd: Deconvolutional single shot detector," *arXiv,* vol. 1701.06659, 2017.

[60] M. Everingham, L. V. Gool, C. K. I. Williams, J. Winn and A. Zisserman, "The Pascal Visual Object Classes (VOC) challenge," *International Journal of Computer Vision,* Vols. 10.1007/s11263-009-0275-4., pp. 88. 303-338, 2010.

[61] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, "Speed/accuracy trade-offs for modern convolutional object detectors." *IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* pp. 3296-3297, 2017.

[62] E. Mohamed, K. Sirlantzis, and G. Howells, "Application of Transfer Learning for Object Detection on Manually Collected Data," in *Intelligent Systems and Applications*, Springer International Publishing, 2020, pp. 919--931.