Detecting Armed Conflict Damages in Satellite Imagery Using Deep Learning

Master Thesis

A thesis submitted in partial fulfilment of the requirements for the degree of the Master of Social and Economic Data Science at the University of Konstanz.

submitted by

Birke Pfeifle

01/938529

at

Universität Konstanz

Period of Completion: 25 April 2022 - 26 September 2022

1st Supervisor: Prof. Dr. Nils B. Weidmann

2nd Supervisor (external): Prof. Dr. Konrad Schindler (ETH Zurich)

Konstanz, 26 September 2022

Abstract

While in high-risk armed conflict settings it can be difficult to gather information on the ground, satellites offer a non-intrusive form of overviewing the situation. When it comes to the monitoring of urban damage, this is usually associated with a high cost since very high-resolution images have to be purchased. Meanwhile, there are satellites whose images are only of medium resolution, but that cover the world with a high revisit frequency and make their images open to the public. Therefore, this thesis aims to answer the research question of whether it is possible to automate the detection of armed conflict damages in medium-resolution optical satellite images using deep learning. Approaching this task as one of semantic segmentation, a reference dataset of Iraq and Syria has been created consisting of Sentinel-2 images labeled pixel-wise with damage information from UNOSAT. The dataset was used to train an FCN, a U-net as well as a simplified 6-Layers ResNet. While the best-performing models had difficulties with recognizing sparser damage patterns, they performed well when it came to the detection of damage clusters. The experiments show the importance of avoiding to lose resolution, for example, by keeping the stride to 1. Furthermore, the labeling has a strong effect on model performance, indicating that it is better to label too much in favor of avoiding false negative labels in the ground truth. An interdisciplinary approach is proposed for creating and maintaining a database of armed conflict damages which could have major implications for research on the spatial developments of armed conflicts. The inclusion of uncertainty measurements and an approach for verification will be essential for its applicability.

Acknowledgements

I would like to express my sincere gratitude to Prof. Dr. Jan Dirk Wegner and Dr. Valerie Sticher, who entrusted me with joining the RMAC project, taking some of its first steps, and presenting it on several occasions. I would also like to thank Dr. Thao Ton-That Whelan for meaningful insights into her work at the ICRC and how the project could support them. I would furthermore like to thank my supervisors Prof. Dr. Nils B. Weidmann and Prof. Dr. Konrad Schindler for their valuable interdisciplinary academic guidance throughout this project.

This endeavor would not have been possible without the day-to-day supervision provided by Dr. Torben Peters and his support with questions related to technical issues as well as the implementation of my work. I would also like to thank Olivier Dietrich for his feedback and Thomas Gazel-Anthoine for the exchange of ideas on the project and for working on it together. I further wish to acknowledge the help provided by Clémence Lanfranchi, and Andrés C. Rodriguez with technical questions, and the organizational help provided by Monique Berger Lande, Kathrin Krautheimer, and Michael Schuhmacher.

I wish to extend my thanks to Prof. Dr. Anke Hoeffler and Prof. Dr. Karsten Donnay for their encouragement, academic guidance, and support throughout my academic journey.

Finally, I am deeply grateful for the following people and their input and emotional support, which was helping me to finalize this thesis: Rahkakavee Baskaran, Maren Lüdecke, Maik Tran van, Judith Clauß, and my flatmates: Sarah Bastigkeit, Sophia Wild, Marlinde Spira, Hannah Ziehfreund, and Joy Stieger.

Contents

List of Figures

List of Tables

Acronyms

ACLED Armed Conflict Location & Event Data Project 4

AOI area of interest 13

CE complex emergencies 9

CNN convolutional neural networks 6

ESA European Space Agency 2

FCN fully convolutional network 3

HDF hierarchical data format 63

ICRC International Committee of the Red Cross 1

IDP internally displaced persons 6

IHL international humanitarian law 1

IS Islamic State 13

LR learning rate 33

ML machine learning 48

NIR near-infrared 3

RDA rapid damage assessments 9

ResNet residual network 3

RMAC Remote Monitoring of Armed Conflicts 3

SAR synthetic-aperture radar 50

UCDP GED Uppsala Conflict Data Program Geolocated Events Dataset 4

UNITAR United Nations Institute for Training and Research 9

UNOSAT United Nations Operational Satellite Applications Programme 1

VHR very high resolution 1

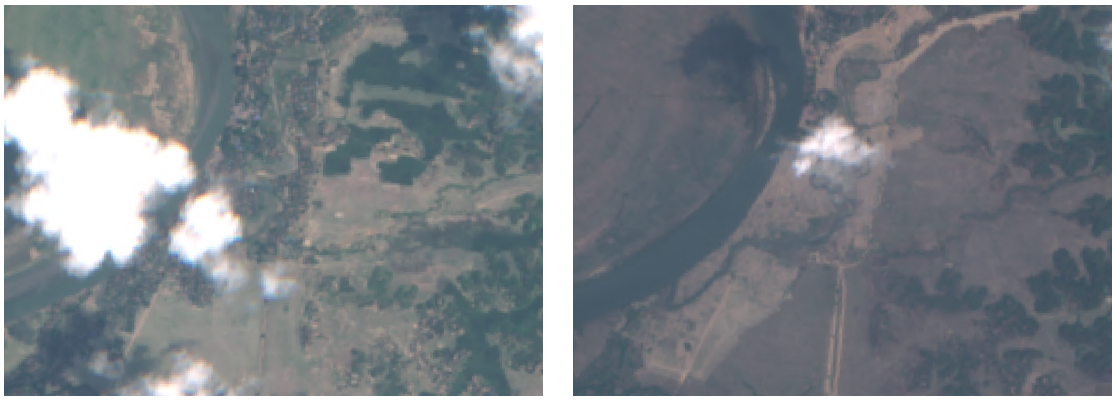
1. Introduction

Despite clear prohibition in international humanitarian law (IHL), civilian infrastructure is often a target in armed conflicts. Research considers the targeting of civilian infrastructure to be both an "increasingly prevalent form of war-making" as well as having "long-term implications for rebuilding states, sustaining livelihoods, and resolving conflicts" (Sowers, Weinthal, & Zawahri, 2017, p. 410). As the war in Ukraine painfully shows as one of the most recent examples, "extensive and repeat urban damage remains a hallmark of modern and increasingly urbanized warfare (Höglund, Melander, Sollenberg, & Sundberg, 2016; Nedal, Stewart, & Weintraub, 2020; Raleigh & Hegre, 2009 as cited in van den Hoek, 2021).

For being able to react appropriately to a crisis, it is essential to know about the current situation on the ground. The extent and time of armed conflict-related damages is critical information for protecting civilians as well as planning the provision of humanitarian aid and subsequent reconstruction. In addition, remote analyses of past conflict events can reveal violations of IHL and constitute evidence of such. Furthermore, geo-referenced time-series information on destruction patterns in armed conflict would help research on actors' warfare behavior and tactics.

While in such high-risk armed conflict settings, it is difficult to gather information, satellites offer a non-intrusive form of observing the situation on the ground. While the military has been using satellite images since a long time, their usage has become more and more popular for inspecting war effects, even by the media. For current crises, organizations like the United Nations Operational Satellite Applications Programme (UNOSAT) or humanitarian organizations themselves like the International Committee of the Red Cross (ICRC) supply experts to analyze satellite images to produce maps that showcase the current state of destruction of conflict-affected areas. Even research has begun to tap into the possibilities that remote sensing data offers. This shows the potential of and interest in the analysis of remote sensing data.

The problem is, that these analyses as well as the images shown in the media are mostly based on a manual analysis of very high resolution (VHR) images. Such VHR remote sensing data is not openly available but can only be purchased from private companies, who own the respective satellites. Furthermore, these satellites do not regularly cover several parts of the world, since they are only focused on current areas of interest. Using such VHR data for the monitoring of vast conflict areas, would therefore be too expensive and often also simply just not possible since not all of the conflict-affected areas are regularly covered by commercial VHR satellites. Furthermore, the manual nature of the analyses and the expert



Sentinel-2 image from 19 April 2017 Sentinel-2 image from 20 March 2018

Figure 1: Example of Sentinel-2 satellite images showing how the town of Hpaw Ti Kaung, Myanmar, has been wiped out; detected by UNOSAT.

knowledge required do not scale up well – especially when wanting to go beyond a rapid assessment of a known hotspot towards a regular monitoring of a larger area. This is also the reason why there is no dataset available currently, that goes beyond single case studies.

A potential solution to the scalability issue is a composite of two things: freely available satellite images that cover most of the world every couple of days as well as a method to automate their analysis. In 2015, the European Space Agency (ESA) launched a twin-satellite system with a high revisit frequency called Sentinel and since has been making its outputs open to the public. While their resolution is considerably lower than what is being used elsewhere for conflict-related damage assessments, one of the mission's goals is to contribute to surveillance related to maritime and border security, indicating its general suitability for monitoring tasks. An example of an armed conflict-related damage, that is very well visible on Sentinel-2 images even for the human eye, can be seen in Figure 1. Based on VHR satellite imagery analysis comparing images from before and after the incident, UNOSAT experts have detected that this whole town in Myanmar has just been obliterated. This also shows in images of less resolution as can be seen here.

When it comes to the automation of the analysis of medium resolution images using deep learning, remote sensing research focusing on topics such as natural disasters has shown that it is possible to do so for certain applications. But the only attempts to automate the detection of conflict-related damages have been working with VHR data (Lee et al., 2020; Mueller, Groeger, Hersh, Matranga, & Serrat, 2021).

Therefore, my research question is: Is it possible to detect armed conflict damages in medium-resolution optical satellite images using deep learning?

My research is closely linked to the Remote Monitoring of Armed Conflicts (RMAC) project by ETH Zurich. In cooperation with the ICRC, its main goal is to develop a deep learning-based monitoring tool to enhance the ICRC's early warning capabilities with respect to conflict events. Another output of the project will be a database on armed conflict damages which can be used for research.

The necessary first step in this endeavor is to train an algorithm that is able to detect relevant damages in satellite images of the given resolution. Once such an algorithm is trained properly, it can be used to revisit past satellite images of known conflict areas, scan those for damages and thereof create a database. The thereby collected information can give new insights in past conflicts, both for research as well as the documentation of human rights violations. Furthermore, the trained algorithm can be applied for near-real time monitoring of conflict-affected areas. The results of this near-real time screening can then feed into both the planning of e.g. the ICRC as well as the database on armed conflict damages.

To answer my research question and to contribute to the development of an appropriate solution to the task of the project, my thesis is showcasing the possibility of solving it by framing it as a semantic segmentation task. This means that I will train several segmentation networks with the goal to classify each pixel in the input images as either damaged by armed conflict or not. The networks tested are a fully convolutional network (FCN), two U-net specifications where I vary the stride, as well as a simplified 6-Layers residual network (ResNet) based on Rodriguez and Wegner (2018). Their performance is compared across different model specifications, where I experiment with the inclusion of an image from *before* the damage happened, the inclusion of the near-infrared (NIR) channel, the size of the labels, and a weighting of the loss function to counteract the class imbalance. In addition, I created a reference dataset to train and test the different model architectures. It consists of satellite images, which are labeled by pixel according to each pixel's damage status. The ground truth on the armed conflict-related damage stems from damage assessments from UNOSAT on Syria and Iraq. The images used for this are multi-spectral optical satellite images from Sentinel-2.

The paper is structured as follows: The first section gives an overview of the related work on both, already available geolocated conflict event data as well as what already has been done when it comes to capturing armed conflict impacts from space, to derive the research gap. The subsequent section presents how the dataset was created and how the

deep learning models were defined and implemented. This is followed by the presentation of the results. In a next step, the results of the different specifications are discussed, alongside potential improvements to the method in future research as well as what they mean for the applicability of this method for social science research. Finally, a conclusion is drawn at the end of the paper.

2. Related Work

2.1. Geolocated Conflict Event Data

So far, conflict researchers working with spatially disaggregated data mainly have two event databases available, i.e. databases where each entry refers to an incident with a given time and place. First, there is the Uppsala Conflict Data Program Geolocated Events Dataset (UCDP GED), which gives information on conflict-related events where the use of armed force by an organized actor resulted in at least 1 direct death (Sundberg & Melander, 2013). There is also the Armed Conflict Location & Event Data Project (ACLED), which tracks events falling under the categories of political violence events, demonstrations, or non-violent political actions (Clionadh, Linke, Hegre, & Karlsen, 2010). In the event-category of violent events, there is the sub-event type "Explosions/Remote violence". With this type, they capture "one-sided violent events in which the tool for engaging in conflict creates asymmetry by taking away the ability of the target to respond" (ACLED, 2020, p. 10). The tools referred to are "explosive devices, including, but not limited to, bombs, grenades, improvised explosive devices (IEDs), artillery fire or shelling, missile attacks, heavy machine gun fire, air or drone strikes, or chemical weapons (ACLED, 2020, p. 10). When these events led to the damaging of a building, they might be in parts comparable to the data on buildings damaged by armed conflicts that is planned to be collected using the remote monitoring tool. However, the geoprecision of ACLED is much lower since it goes up to the level of a town at most. Furthermore, ACLED has been criticized for uneven quality-control issues as well as imprecise geocoding (Eck, 2012).

The information on conflict events in state-of-the-art databases stems mainly from news reports. This affects both the geo-precision as well as the type of information available. Using "satellite imagery [could] generate new insights into conflict dynamics that are distinct from but complementary to geographically disaggregated conflict event data" (van den Hoek, 2021, p. 327), namely those which are visible from a nadir perspective. This can for example be data on newly established settlements indicating a refugee camp or land

use changes. But it could also yield information on damages to buildings and infrastructure. This presupposes the assumption that "spectral reflectance, transmittance, texture, or backscatter" in itself or its change "is an effective and appropriate proxy for conflict damage" (van den Hoek, 2021, p. 329). Information on the actors involved or fatality estimates cannot be derived from satellite images. Nevertheless, in combination with other already existing datasets like the ones from UCDP GED, it could yield interesting insights when revisiting conflicts, investigating events' and actors' relation to building damages, etc. When working with a tool based on machine learning to create such a dataset, this could even yield very timely information on current conflicts and their events. However, how to handle such sensitive data would have to be carefully determined beforehand.

Furthermore, data on conflicts derived from satellites could provide more objective information regardless of population density or reachability. This would be an advantage over the resources available so far, which can be affected by a reporting bias. This bias can lead to an inaccurate or under-representation of smaller events that happened in hard-to-reach and sparsely populated areas, especially with regard to the location and severity of the event (Weidmann, 2015, 2016).

2.2. Remote Sensing for Assessing Armed Conflict Impacts

The impacts of armed conflict potentially visible from space are manifold. Accordingly, research has been using remote sensing data to detect different types of impacts.

One of the topics investigated is *changes in land use* as well as *land cover* as a consequence of armed conflict. These changes come for example in the form of the abandonment of agricultural land (Witmer, 2008; Witmer & O'Loughlin, 2013) and the displacement of agricultural activity (Baumann, Radeloff, Avedian, & Kuemmerle, 2015). While agricultural areas have been abandoned, Gorsevski, Kasischke, Dempewolf, Loboda, and Grossmann (2012) and Stevens, Campbell, Urquhart, Kramer, and Qi (2011) have found natural vegetation to be increasing in both current and former conflict areas in the South Sudan-Uganda border and Nicaragua. In other armed conflict settings, however, the picture can be quite different. For example, when the burning of oil wells is used as a warfare tactic, the terrestrial ecosystem can be severely affected (El-Gamily, 2007). Furthermore, the emergence of vast refugee camps can impact the surrounding environment. Hassan, Smith, Walker, Rahman, and Southworth (2018) have used Sentinel images to track such a camp in Bangladesh and how it affected its surrounding forest land. Changes in both land cover and land use have mainly been investigated using medium-resolution satellite imagery from Landsat and Sentinel, which makes sense since they have been developed exactly for these

applications. Another sign of armed conflict visible on medium-resolution images is the burning of areas or villages and fires (Bromley, 2010; Prins, 2008).

In addition to multi-spectral images, night-time images can also be used for monitoring armed conflict's socio-economic effects (e.g. Li, Chen, & Chen, 2013; Li, Liu, Jendryke, Li, & Wu, 2018; Witmer & O'Loughlin, 2011).

When it comes to the monitoring of anything involving buildings or makeshift settlements, remote sensing research so far has mostly been relying on VHR images. For example research monitoring the development of internally displaced persons (IDP) and *refugee camps* has been carried out mainly based on manual analyses of VHR images (Bjorgo, 2000; Giada, de Groeve, Ehrlich, & Soille, 2003; Kemper, Jenerowicz, Gueguen, Poli, & Soille, 2011; Lang, Tiede, Hölbling, Füreder, & Zeil, 2010).

When it comes to the detection or assessment of building or urban *damages*, the majority of the remote sensing literature has focused on natural disasters rather than conflict (e.g. Bevington, Eguchi, Gill, Ghosh, & Huyck, 2015; Gupta et al., 2019; Kahraman, Imamoglu, & Ates, 2016b; Moradi & Shah-hosseini, 2020; Xu, Lu, Li, Khaitan, & Zaytseva, 2019). On armed conflict-related damages, most of the analyses are still done by manually interpreting satellite images as a review by Avtar et al. (2021) shows. There have been some efforts to take steps towards automation. Jenerowicz, Kemper, Pesaresi, and Soille (2010) for example rely on simple differencing methods between pre- and post-damage images of VHR of Sri Lanka. Kahraman, Imamoglu, and Ates (2016a) use special feature vectors created of VHR images for post-conflict damage assessments of the Gaza Strip. Another approach is to rely on time-series analysis like Braun (2018) or Marx, Windisch, and Kim (2019).

While in the study of impacts of natural disasters it is quite common to use machine learning approaches (e.g. Bai et al., 2017; Ireland, Volpi, & Petropoulos, 2015; Rudner et al., 2019; Vetrivel, Gerke, Kerle, Nex, & Vosselman, 2018; Yang & Cervone, 2019), in the context of remotely assessing armed conflict impacts only very few approaches have been using machine learning. Hassan et al. (2018) worked with a random forest classifier for the assessment of the refugee camp's impact on forest cover. Furthermore, two research efforts applied deep learning techniques to the remote detection of armed conflict damages in Syria. Mueller et al. (2021) deployed a 2-step approach, where they follow up a flat convolutional neural networks (CNN) architecture by a random forest classifier to classify 32×32m patches into destroyed or not. Lee et al. (2020) developed a semi-supervised solution for detecting damaged buildings with limited labeled data. However, they both worked with VHR images.

While the works of Mueller et al. (2021) and Lee et al. (2020) have already shown great

potential in applying deep learning techniques to the automatic detection of armed conflict damages, these methods have only been applied to single cases, and none of them have resulted in the creation of a comprehensive dataset. This is due to the fact that the expansion of their approaches to other cases requires the availability of VHR images. With a few case-specific exceptions, those are not freely available. This significantly limits the possibilities of using these approaches for the creation of a comprehensive database on past armed conflict damages and would make the continuous monitoring of vast conflict areas far too expensive.

Therefore, this paper aims to test whether deep learning solutions can be applied to medium-resolution optical images from Sentinel-2 to detect armed conflict damages. Since they are freely and regularly available, this would allow for (1) the creation of a database on damages both from past as well as current conflicts as well as (2) the monitoring of high-risk areas for potential signs of conflict. To the best of our knowledge, the RMAC project is the first to attempt the detection of damages from armed conflict in medium-resolution images using deep learning.

3. Methods

In the following, I will elaborate on how I created a reference dataset including satellite images of conflict areas, where each pixel is labeled according to its damage status. Further, I will explain my choice and implementation of simple deep learning algorithms to solve the task at hand. The whole process pipeline is visible in Figure 2.[1]

[1]Not all of the steps visible in the process pipeline will be discussed in detail in this report. Some are briefly explained in the appendix, or else visible in the code, which is handed in with this thesis.

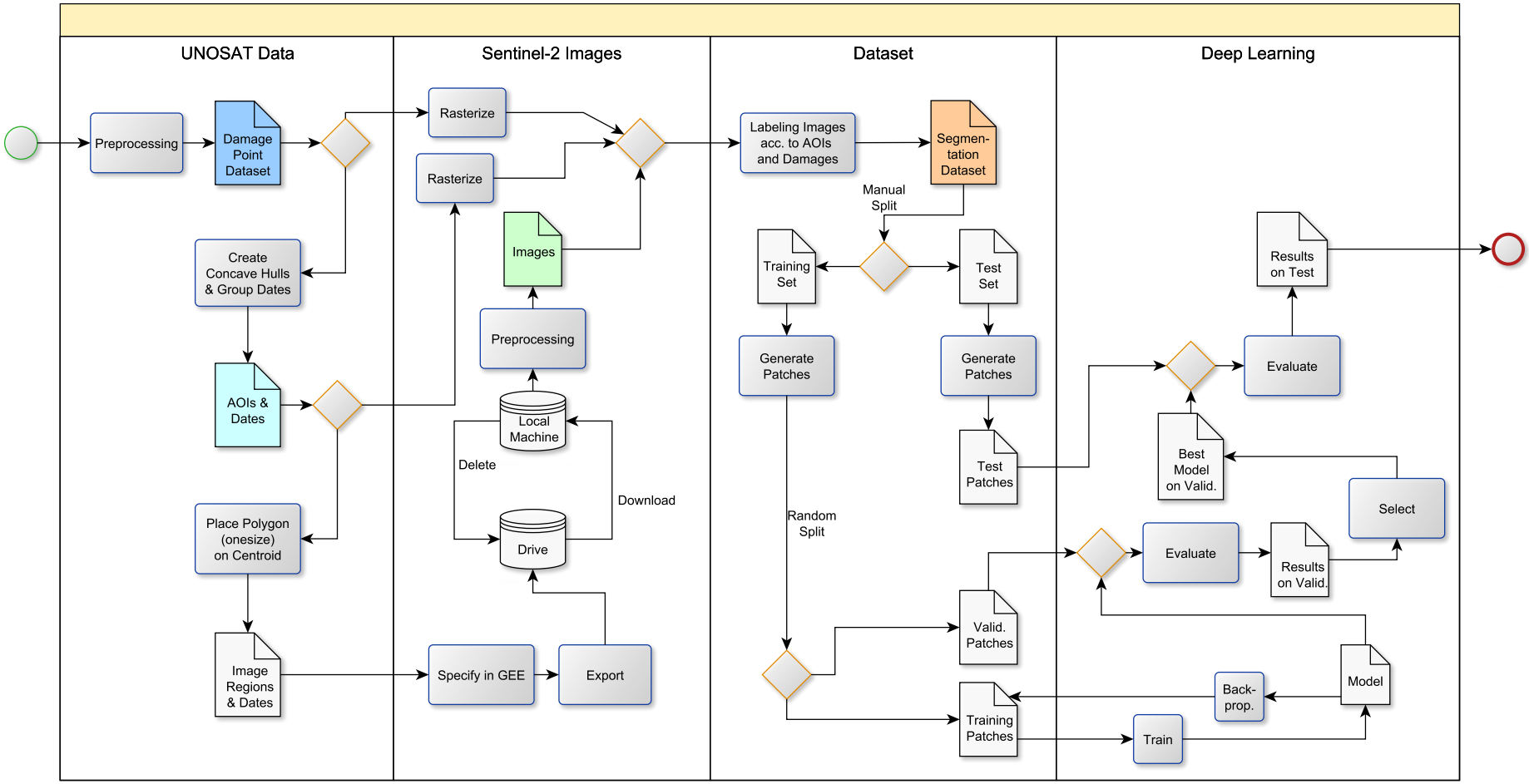


Figure 2: Process pipeline.

3.1. Reference Data: Creation of a Segmentation Dataset

The first output of my master thesis is a reference dataset to train, test, and validate the segmentation model along with the pipeline to create said dataset (see Figure 2). This pipeline can be used as a base for further steps within the project. To create the aforementioned dataset, optical satellite images were labeled by pixel based on georeferenced information about damages inflicted by conflict.

In the following, I will first talk about the respective data sources. Second, I will quickly explain the selection of my study area. Then, I will introduce the pipeline steps for the retrieval and combination of the data, followed by a more in-depth explanation of each step and the decisions I took along the way.

3.1.1. Data Sources

Data on Conflict-Related Damages by The information for locating conflict-related surface changes stems from UNOSAT which is a part of United Nations Institute for Training and Research (UNITAR). On-demand, they deliver maps and GIS products via their Rapid Mapping Service to the humanitarian community. Their maps are based on a manual analysis of VHR satellite images, i.e. with a resolution of 30-70cm per pixel, of the respective area of interest. UNOSAT offers, among other situations, satellite image analyses for complex emergencies (CE). For armed conflicts, which fall within that category, their analyses include, in particular, refugee and IDP camp mapping as well as conflict damage assessments (United Nations Satellite Centre, n.d.). While the project aims to detect different kinds of surface changes related to conflict, I will focus on damages from armed conflict. UNOSAT's information will be my ground truth, i.e. define the areas I want the deep learning model to learn to detect as conflict-related damages.

The datasets are publicly available and provided in the form of geospatial vector data (geodatabases and/or shapefiles) on their website separately for each case. Damage assessments are mainly available in the form of points indicating the coordinates of damages on the ground. With their damage assessments, UNOSAT also provides information on the severity of the damage. They distinguish destruction, severe damage, moderate and possible damage as well as impact craters on fields and roads (see also Table 2). The difference between destruction, severe and moderate damage is visualized in Figure 3.

Besides the damage assessments, UNOSAT also provides rapid damage assessments (RDA)s for some situations. These come in the form of grids laid over an area of interest and an indication of the percentage of damaged buildings per grid cell. As their name indicates,

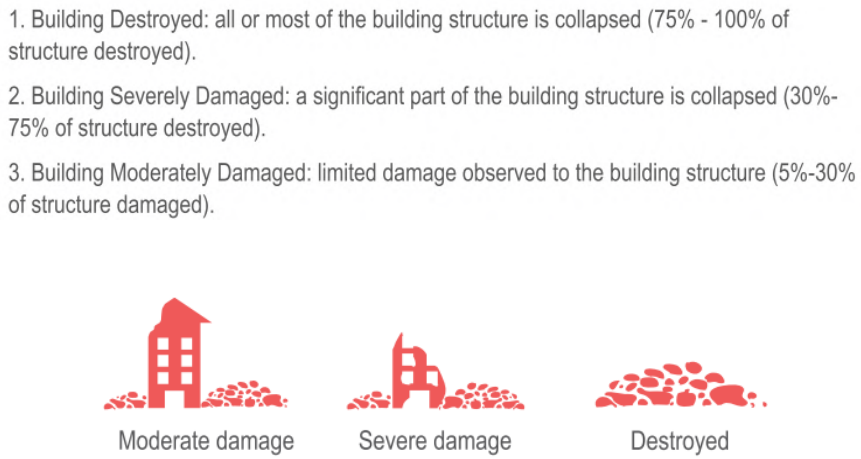


Figure 3: Definition and visualization of the damage categories distinguished by UNOSAT.[2]

however, these are for the purpose of giving an overview of the situation as rapidly as possible. Hence, I consider the data quality to not be sufficient to serve as ground truth for the purpose of this project. I could imagine them to be used in later stages of the project as additional information or for testing. I, however, will only work with the damage assessments available in the form of coordinates of damages, since they promise higher data quality, which is essential when wanting an algorithm to learn the respective patterns.

In addition to the vector data, UNOSAT also publishes annotated maps laid over either a satellite image of high resolution or a topographic map in PDF format. These PDFs often also contain information on the dates of the images they used for analyses and sometimes some more meta data. However, this information is not always given and not uniformly formatted, wherefore it was not used in this thesis, apart from a manual inspection of the dates of images UNOSAT used as reference for the cities pre-damage. Besides from the image sometimes displayed on the PDF, they do not disclose the original images they used for their analyses.

Satellite Images by Sentinel-2 The second component for the reference dataset is satellite imagery by Sentinel-2. The two satellites from the ESA Copernicus Programme cover all continental land surfaces (including inland waters) between latitudes 56° South and 82.8° North (and more) every three to five days and make images with a ground sampling distance

[2]The visualization stems from United Nations Satellite Centre and REACH (2019).

of down to 10 meters freely available (European Space Agency, n.d.-d, n.d.-g). From the launch of the first satellite, Sentinel-2A, in June 2015 until the launch of the second satellite, Sentinel-2B, in March 2017, however, the revisit frequency was only every 10 days (European Space Agency, n.d.-e). This allows for matching the conflict assessments from UNOSAT with satellite images for the respective dates in a very small time window – if the area has not been covered by clouds for longer periods of time at least. Training a deep-learning algorithm with Sentinel-2 images has the further advantage that it does not only allow for a retrospective analysis of conflicts but will also allow for global monitoring of high-risk areas in near-real time, once the monitoring tool has been developed. Furthermore, the Sentinel-2 satellites take multi-spectral images, i.e. cover RGB bands, several infrared bands as well as a quality assessment channel with an estimate of cloud coverage.

There are two versions of Sentinel-2 images available: The first is the rawest version ESA published, i.e. the Top-Of-Atmosphere (TOA) product called Level-1C. Some preprocessing steps, namely radiometric and geometric corrections (including orthorectification and spatial registration) have already been applied to this product. Cloud masks have been generated as well (European Space Agency, n.d.-b). The second published version is called Level-2A, and atmospheric correction has been applied to those images (Main-Knorn et al., 2017). The images I used for my analysis were, however, only available in the 1C-version since the "Sen2Cor" processor, which generates 2A images, has only been published in 2017 and therefore the respective corrected images have also only been published for images taken from then on. Hence, I just applied image-wise min-max normalization and manually selected images, which looked appropriate and did not show any obvious discolorations.[3] Nevertheless, not correcting for the respective atmospheric effects might have impacted the quality of my data basis.

The Sentinel-2 mission was designed mainly for "users interested in thematic areas such as spatial planning, agro-environmental monitoring, water monitoring, forest and vegetation monitoring, land carbon, natural resource monitoring, and global crop monitoring" (European Space Agency, n.d.-c). However, next to land and maritime monitoring, the Sentinel-2 data, also feeds into applications in the Copernicus priority areas of emergency management (which includes natural disasters, man-made emergency situations and humanitarian crises) and security, which mainly refers to satellite-based surveillance (European Space Agency, n.d.-a). The latter two application areas indicate the suitability of the Sentinel-2 products

[3] Future research could apply the Sentinel-2 toolbox to those Level-1C images, however, which performs the pre-processing steps necessary to get from Level 1C to 2A and correct those potential errors (European Space Agency, n.d.-f). Unfortunately, it was not possible to implement this for this thesis due to time constraints.

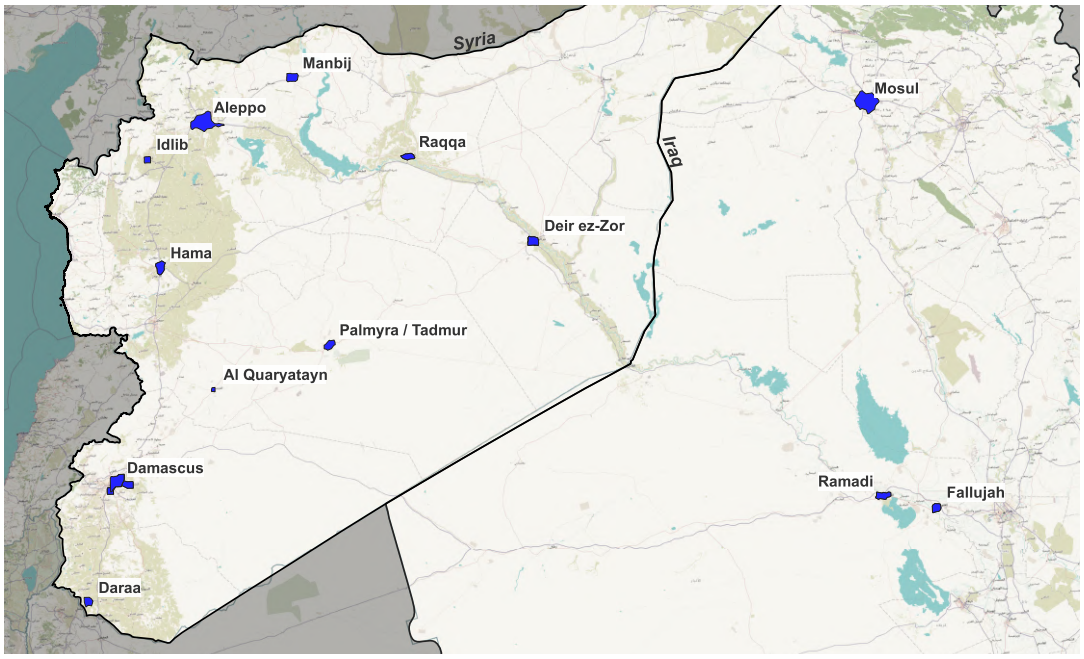


Figure 4: Overview of areas included in the dataset.

not only for land monitoring but also for the monitoring of issues related to armed conflicts.

3.1.2. Study Area

The regions, I focused on and created reference data for, are several urban areas in Iraq and Syria, which are depicted in Figure 4. There are three main reasons for this. First and most importantly, the UNOSAT maps contain information related to armed conflicts for these two countries. Furthermore, the information on these areas goes beyond 2016, which is relevant insofar as Sentinel-2 images are available only from June 2015 onward. Second, these two countries are quite arid regions, which makes remote monitoring using optical satellite images easier, since the chances of clouds limiting the view are much lower than in more humid regions.[4] Third, the two countries should be very similar semantically, wherefore the model should not struggle to learn from both at the same time.

The wars in Syria and Iraq both heavily included the targeting of buildings. "[T]hroughout the course of the conflict in Syria, about three-quarters of injurious attacks there occurred in

[4]In humid regions, when clouds block the view on a conflict site for too long, the next time the ground is visible via satellite images, the damage might have changed or been fixed by then.

populated areas" (Overton & Dathan, 2019). By 2017, about a third of residential buildings as well as half of the basic social infrastructure had been damaged or destroyed (Overton & Dathan, 2019). In Iraq, the war against the Islamic State (IS) has also left several cities severely damaged; the city of Mosul for example, whose old town was destroyed by 65% (Minority Rights Group International, 2020, January 21). This is why these two countries provide many examples of conflict-related building destruction to train a supervised semantic segmentation algorithm. At the same time, since Iraq and Syria have both been heavily affected by war, I will only be able to make statements about the performance of the models on conflicts involving large-scale infrastructure destruction in the end. However, the damage patterns vary between cities as well as within the cities included in the data, therefore the algorithm will not only be trained on almost completely destructed areas of a city but also on lower levels of damage, which should help the generalizability to other types of armed conflicts. In order for the tool to be able to monitor different types of conflict areas, however, future developments of this project will need to incorporate data from different climatic conditions, different urban patterns, and different types of conflict.

3.1.3. Data Generation Pipeline

In the following, I will give a brief overview of the pipeline necessary for creating a dataset usable for training the deep learning algorithm with to detect armed conflict-related damages in medium-resolution satellite images.

As the first step in my pipeline, which is also visible in more detail in Figure 2, a comprehensive *Damage Point dataset* was created. On the basis of this dataset, image regions and dates were defined for retrieving the satellite images on which said damage should be visible. These images were then labeled by pixel with the damage information as well as information on the area of interest (AOI)s to compile the final dataset.

In the following, I will go over the different steps involved in the creation of the reference dataset and visible in Figure 2 in more detail.

Creating a Damage Point Dataset Since the data provided by UNOSAT is not provided in a comprehensive dataset, but for each map separately, the first task was to create one.

Since there is no API available, I scraped the information on available datasets and metadata like their title and the links to download them.[5] UNOSAT already categorizes their maps into several categories, and armed conflict settings fall under the category of CE.

[5]Shortly before the hand-in date, their website structure has changed. In future developments of this project, the script would need to be adapted to that.

To differentiate damage assessments from other maps falling under the CE category, I also created a sub-categorization of the maps based on their titles on the website using regular expressions.[6] Since the shapefiles showed some major inconsistencies in their variable naming, I worked with the geodatabases instead. I only included data published after 2015, since Sentinel-2 was launched in June 2015, which is why for damage assessments from 2015, I would not have been able to include meaningful pre-damage images. Once filtering for damage assessments only, it also became easier to combine them into a comprehensive dataset, since the variable naming is consistent.[7] Furthermore, the handling of dates required some preprocessing as well to get them all into one dataset in long format.

The variables present in the finished comprehensive damage point dataset based on UNOSAT's data are visible in Table 1. The "geometry" column contains information about the location of the damage entry. The "Main_Damage_Site_Class" column contains an integer referring to the damage category that was assigned by UNOSAT. Which integer refers to which category is listed in Table 2. Similar to Lee et al. (2020), I found some inconsistencies in the variable coding, namely the coding of categories 4-7 is not always consistent within UNOSAT's data. However, since I use binary variables, coding either any damage or only destruction as 1 (see the categories "damage" and "destruction" in Table 1), this does not affect my analyses either way. The "SensorDate" column by UNOSAT is referring to the date the sensor took the satellite image on which UNOSAT analysts detected the damage. The "prevSensorDate" column contains information on the date (if available) at which that damage entry was not identified as damaged yet. This is only available for points in areas that have been analyzed by UNOSAT several times. This allowed the creation of this variable for points that have been detected as damaged only in a later analysis of the area. I did not end up using this information, however. In a future version of this project, one could think about training only with this information on newly damaged points. Finally, the last two columns contain information about the geodatabase and the layer within that geodatabase which the damage entry was retrieved from.

The dataset contains 303,386 damage point-dates of which about a quarter (82,454 points) refers to destruction, see Table 2. These damage point-dates refer to 124,612 different affected structures, i.e. point locations. Of these point locations, 99,700 lie within Syria and 24,912 in Iraq. On the map in Figure 14 you can see the cities that have been analyzed and where they lie. Most of the data points included stem from analyses from the year 2016

[6] These titles followed the most consistent naming as compared to e.g. the filenames.

[7] It was done so using the `geopandas` package (Jordahl et al., 2020). The final dataset was stored in a feather file.

Variable	Content	Data type	Created by
geometry	point coordinates of the location	geometry	UNOSAT
Main_Damage_Site_Class	type of damage	integer, 1 – 7	UNOSAT
damage*	1 if Main_Damage_Site_Class > 0, else 0	integer	own
destruction*	1 if Main_Damage_Site_Class = 1 (i.e. destroyed), else 0	integer	own
SensorDate	date the damage was detected	datetime	UNOSAT
prevSensorDate	date of when the point has not been damaged yet, if available	datetime	own
gdb	name of the geodatabase the point was retrieved from	String	own
layer	name of the layer in the geodatabase the point was retrieved from	String	own

*Since the whole project team came to be working with the same dataset basis, these variables are not included in the according feather file but are only added in a later script by me, shortly before the rasterization of everything.

Table 1: Variables in the damage point dataset.

Category	Definition	Count
1	Destroyed	82,454
2	Severe Damage	105,171
3	Moderate Damage	110,990
4	Possible Damage	101
5	Impact Crater (Damage to Road)	385
6	Damaged Road Segments	3,399
7	Impact Crater (Damage to Field)	886

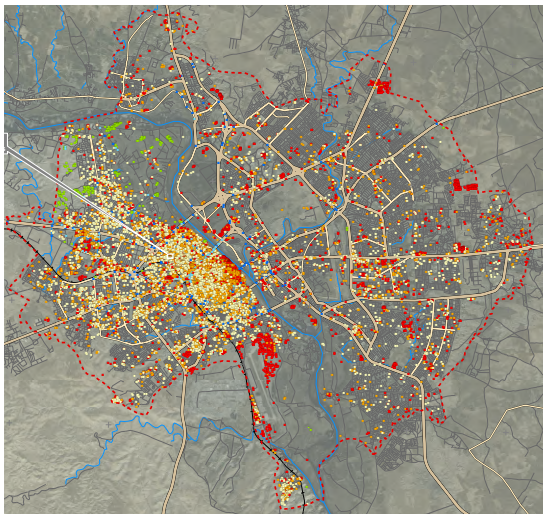
Table 2: Categories of damage intensity defined by UNOSAT.

(about four-fifths).

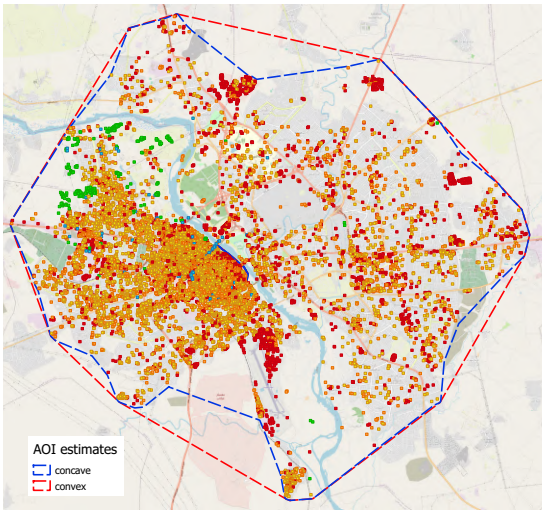
Defining Areas of Interest For the dataset, it is important to know, about which areas of an image there is information available. To avoid adding false negative labels to the dataset, it is critical that no training data is generated outside of UNOSAT's labeled areas. In addition to the damage assessment data in the form of points, UNOSAT provides a separate layer with a polygon depicting their AOI, i.e. the area they analyzed, however, only for a few maps. Therefore, the concave hull over all labeled points from the same geodatabase layer was defined as AOI. In Figure 5, you can see that the concave hull AOI estimation is closer to the actual AOI (visible on UNOSAT's PDF map (United Nations Satellite Centre, 2017)) than the convex hull would have been. For an overview of all AOI estimations included in this thesis see Figure 4.

As for the size of the images that get downloaded to create the dataset later, it makes sense to work with the same across all maps. To achieve that, first, all the AOIs that intersected with each other were merged to create one AOI-cluster for each city that has been analyzed by UNOSAT. Around each AOI-cluster bounding boxes were drawn. The size of the images to download later was chosen a bit bigger than the largest of these bounding boxes. Polygons of that size were placed on each centroid of the different AOI clusters to make sure that all the AOIs are included in the images I download, but not downloading larger images than necessary. The images thereby downloaded comprised 3451×3451 pixels.

To differentiate the AOIs from what is covered on the downloaded images, I call the area covered in an image "region". Areas that are lying outside of an AOI in an image region get discarded in the labeling process later.



Extract from UNOSAT's map for their analysis of Mosul from 04. August 2017



Own visualization of damage points plus the two different AOI estimates

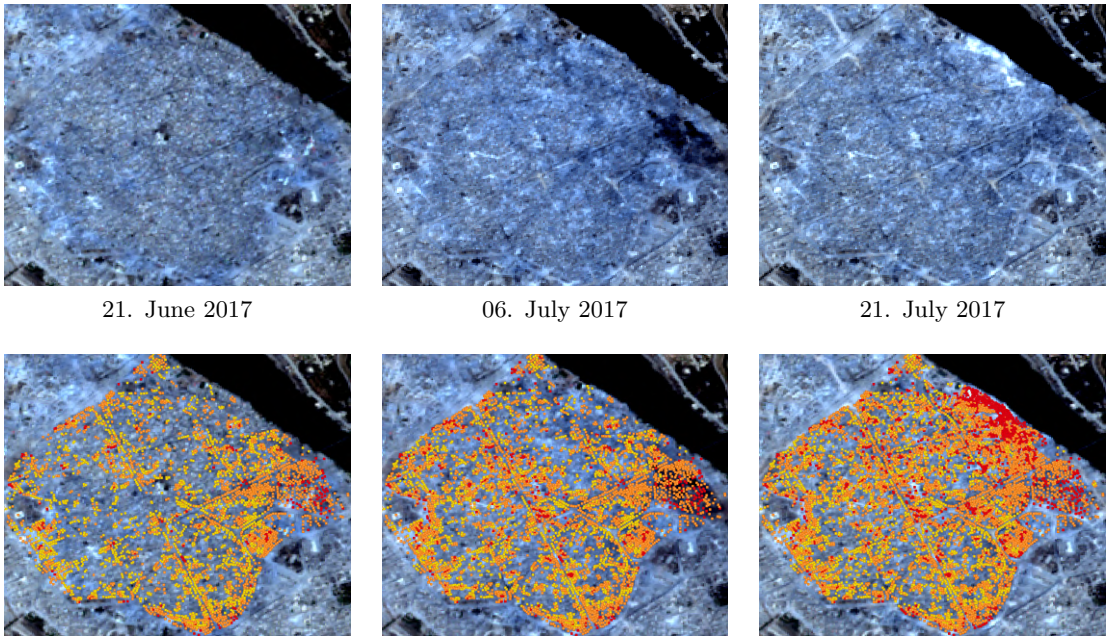
Figure 5: Comparison of actual AOI to AOI estimates – concave and convex.

Defining Dates of Interest A non-trivial issue is how to actually choose the dates of the satellite images in which the damages are supposed to be detected. The UNOSAT analyses are mostly based on manual comparisons of high-resolution images from before and after the supposed conflict events.

I assume that damaged samples at time t_i also remained damaged at subsequent times $t_j > t_i$, similar to Mueller et al. (2021). Thereby, I cannot only choose an image from the exact same date as UNOSAT's analysis as the *after image* but can also resort to one after that if no image from the exact same date was available. With the term *after image* I refer to the image in which the algorithm is supposed to detect the respective damage. In Figure 6, you can see an example of such Sentinel-2 images that were taken shortly after a damage assessment by UNOSAT. In the bottom row, they are overlayed with the damage points detected in an analysis by UNOSAT shortly before those images.

While it is quite straightforward to choose images from at or shortly after UNOSAT's date of assessment, it is not as clear how to go about choosing an appropriate image for the *before*-damage situation. "With months between images, UNOSAT damage maps effectively convey a multi-month cumulative damage map since the specific timing of damage events remains unknown" (van den Hoek, 2021, p. 328). In addition, the dates of the *before image*[8]

[8]With the term *before image* I refer to the image on which the damage to be detected on the respective *after image* should not be present yet.



21. June 2017 | 06. July 2017 | 21. July 2017

16. June 2017 | 30. June 2017 | 18. July 2017

Figure 6: Example of Sentinel-2 satellite images of the Old City of Mosul, Iraq. In the second row, they are overlayed with the respective damage assessments by UNOSAT.

are not even provided as such with the datasets but only on the respective map's website or PDF in an inconsistent format. A manual analysis of those *before* dates indicated in the PDF maps showed that most of the before dates of images used by UNOSAT for comparison lie around or before the launch of the Sentinel satellites.[9] Therefore, simply the earliest available Sentinel-2 image for each image region was chosen as the *before image* of that region for the best possible chance of the damage not being visible on the *before image* already. This is similar to Mueller et al. (2021), who also chose one image as the *before image* for all of the *after images* of an area.[10]

Since it is likely that this is not the perfect choice for each case though, I will also run the models without a *before image* and compare the performances.

Retrieving Images of Interest The Sentinel-2 images were retrieved via the Google Earth Engine (GEE) (Gorelick et al., 2017) instead of Sentinel's Scientific Data Hub, since GEE allows for script-based interaction.[11] In terms of channels, I downloaded all the bands available at 10m resolution. These are: the three RGB channels (B2-B4) plus the NIR channel (B8) plus the quality assessment layer (QA10) European Space Agency (n.d.-g).

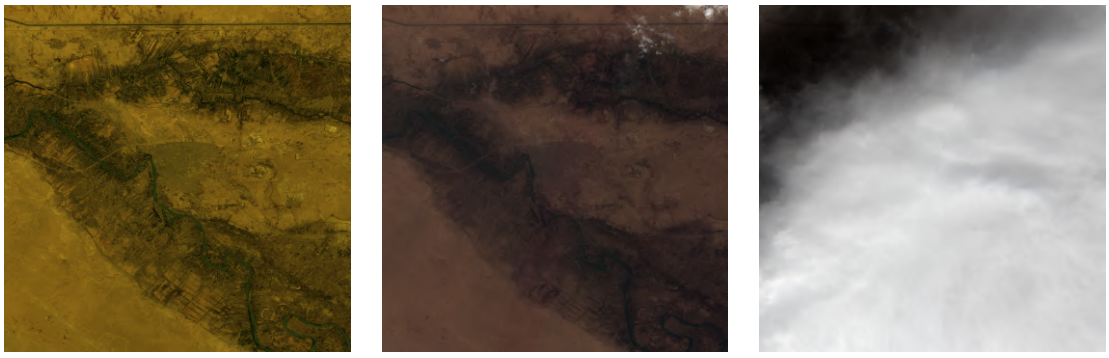
As for the *before images*, I just downloaded all images in between the launch of Sentinel-2 up to six months after that date for each image region. I inspected the available images within quite a long time span here, since images were not taken as frequently back then (every 10 days at best) and the quality of the images was also suffering from some discolorations every now and then, so I wanted to make sure to choose a *before image* of acceptable quality since this will heavily influence the results.

Another issue I had to deal with was cloud coverage present on some images. A manual inspection of the quality assessment layer provided by Sentinel showed, that these layers often exhibit inaccuracies. In addition, this layer does not sufficiently solve the issue of detecting cloud shadows, which could skew the results as well. This is why I downloaded all the images available for the first couple of months after Sentinel's launch for each AOI. Then I went through them manually and chose the first image of good quality as the *before image* for that area. The quality was determined by how cloud-free an image was as well as whether the colors seemed distorted, which was the case for some images, which is why they got ruled out. Examples of images that got ruled out are depicted in Figure 7.

[9] Only a few *before image* dates are after that and at these dates, the respective cities have already been damaged as is evident by UNOSAT having analyzed the respective area before.

[10] However, the before dates they chose were even longer before. They were able to do so since they worked with a different image source (of higher resolution as well).

[11] For more details on the image retrieval, see Appendix Section A.



discolored (21. August 2015) | foggy and discolored (10. September 2015) | heavily covered by clouds (20. October 2015)

Figure 7: Examples of Sentinel-2 images of the image region covering Fallujah, Iraq, that got discarded due to heavy discolorations or clouds.

Furthermore, some images did not cover the whole AOI, which is why they got ruled out as well. [12] Another common solution in remote sensing is to take the median for each color band across several subsequent dates. However, I have ruled out this option, since it did not seem suitable for my application. Taking the average could diminish the exact changes I want to detect.

To normalize the images, I turned each channel of each image into integer values between 0 and 255, which is common for images, by applying min-max normalization to each image separately.[13]

Rasterizing the Damage Information To bring everything together, I needed to turn the point-wise information on damages into information per pixel of the respective satellite images. To test the performance of the simplest solution, only the pixels wherein there lies at least one damage coordinate were labeled as damaged.[14] The labeling was carried out in a binary manner, i.e. pixels with damage present were denoted with 1 and others with 0, instead of varying the labeling by the count of damage points within a pixel or according to their damage category. Instead, I implemented two different definitions of this binary labeling. In the first one, pixels are labeled with 1 simply when any damage lies within that pixel's borders. As per my second definition, only pixels with destruction present are

[12] In future adaptions of the download workflow, one could include an option to indicate already at download, that for the time frame at hand, only the images with the least clouds or with a cloud cover percentage below some threshold should be downloaded in order to speed up this selection process.

[13] When feeding the images to the models later, they were turned into floats between 0 and 1.

[14] For a discussion of this issue, see Section 3.1.4.

labeled with 1. When working with the UNOSAT damage assessment data, (Mueller et al., 2021, p. 2) found that even in the high-resolution images they used, damages labeled as moderate or severe damage "were not always clearly visible in the satellite images". Hence, it is reasonable to assume that when working with less resolution, it is even more likely that only destruction is visible in the satellite images. This is why I also ran all the models with this definition only based on destroyed buildings.

All the damage information that lies geographically within an image and was assessed before the said image was taken got rasterized.[15] Furthermore, I added information on my estimate of UNOSAT's AOI. To only include the appropriate AOI, the latest available one before the image date that lies completely within the image region was selected. This is a reasonable choice since this area is the last one that UNOSAT analyzed, i.e. within that AOI one can be the surest that the damage information is up to date. Anyway, when UNOSAT analyzed a region several times in a row, if anything, the respective AOI increased over time. An example of that is Mosul, where at first only the old town was analyzed but at a later analysis the AOI was widened to include all of Mosul, as you can see in Figure 8. Hence, I will not lose any information by ignoring everything outside of the latest AOI before an *after image* was taken.

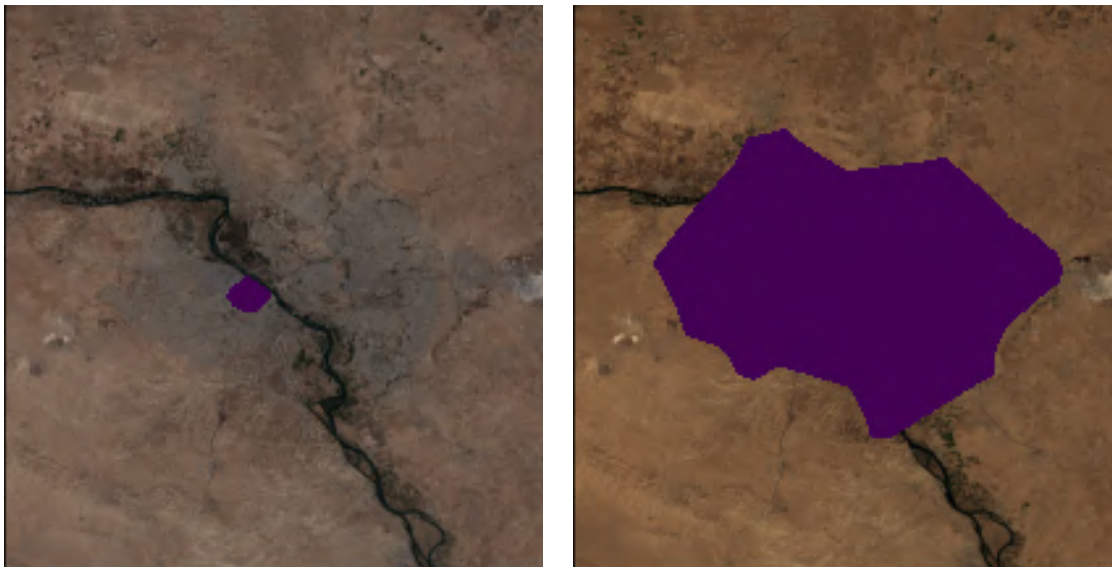
Everything outside of my estimate of UNOSAT's AOI was labeled with 99, in order to be able to ignore those pixels in all later steps easily. Any small clouds potentially remaining even after my qualitative manual selection of suitable images, were labeled with 99 as well based on the quality assessment layer provided by Sentinel.

Finally, the segmentation dataset consisted of the following information:

- an *after image* (incl. RGB and NIR channels)
- a segmentation mask including all the damage that happened
  - geographically: within that image region
  - in terms of time: before that image was taken
- the *before image* for that area (incl. RGB and NIR channels)
- an AOI mask based on the latest available AOI within that image region before the *after image* was taken

In this dataset, of 15,080,939 labeled pixels in total there are 156,632 labeled as damaged and 39,233 labeled as destroyed.

[15] For more implementation details, see Appendix Section B.



AOI estimate based on UNOSAT's damage assessment from 11. June 2017

AOI estimate based on UNOSAT's damage assessment from 04. August 2017

Figure 8: Visualization of AOI increase over time.

3.1.4. Dealing with Label Uncertainty

Semantic segmentation requires training data where each pixel is labeled. When only having point coordinates, which refer to the centroid of a damage, available, this gets difficult. When only labeling pixels, where a damage coordinate lies within, as positive (as I did in my "vanilla" solution, see Section 3.1.3), probably not all pixels that actually experienced damage are captured. The damages that the labels by UNOSAT are referring to can be of different extent. Therefore, it is unlikely that the damage identified relies exactly within the pixel's borders. It is likely that said damage is also visible in one or several of the neighboring pixels. At the same time, we, therefore, do not know for certain which pixels actually have not been damaged. Hence, we are dealing with incomplete supervision, since we only know the labels of a few single points within whole images and we do not know their extent.

To deal with this issue of label uncertainty, several approaches have been discussed in the literature. One option is to work with the pool of methods that were developed particularly for tasks where you are dealing with incomplete supervision: semi-supervised learning. Another approach often discussed particularly for working with the data from UNOSAT's damage assessments is to link the point coordinates to the respective building footprints

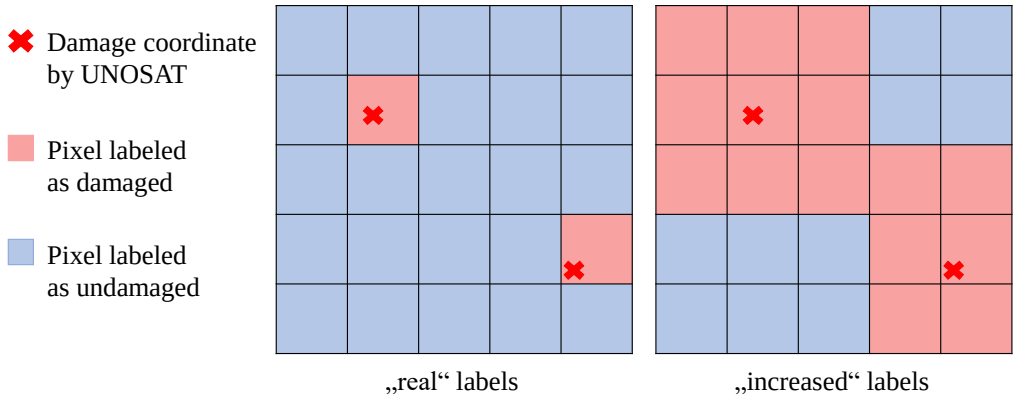


Figure 9: Schematic visualization of the label increase performed.

(e.g. Hasnat & Faisal, 2015; Kahraman et al., 2016a, 2016b; Lee et al., 2020; Xu et al., 2019).

In light of the time constraints faced with the implementation of this thesis, I rather thought about a simple strategy to improve the labeling rule-based. In my "vanilla" models, I just assumed that all of the pixels with a damage coordinate within are damaged, and all the others are not. However, buildings are likely to often be bigger than 10 by 10 meters and, in addition, likely not to follow the arbitrary sizes of the respective pixels the respective coordinate lies within. Hence, I also ran the models with a simple version of increasing the labels. To do so, I simply increased the labels by one pixel "circle" around each "real" label.[16] This means that in the "increased" label version, a damage point coordinate was matched not only with one but with the nine pixels around that coordinate. This scheme is visualized in Figure 9.

This led to more pixels being labeled in general and also more continuous areas being labeled as damaged in the segmentation mask, as is visible in Figure 10, which is an example image tile from the test data.

3.2. Deep Learning Model for Detecting Damages Related to Armed Conflicts

In the following, I will explain how I approached the implementation of a solution to this deep learning problem. First, I will define my task in computer vision-/ deep learning terms and give a brief overview of the relevant architectures. Second, I will explain how I defined

[16] To contrast the two labeling methods in the following, I differentiate them by referring to "real" labels, as opposed to the "increased" labels.

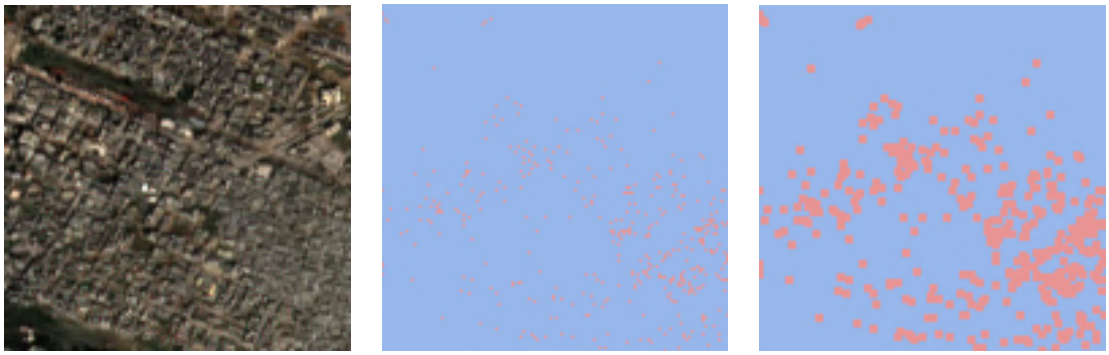


Image tile: 128×128 pixels | "real" labels | "increased" labels

Figure 10: Example image tile with segmentation masks with and without a simple label increase.

the different models in Section 3.2.2. Then I will discuss how class imbalance could affect the success of the models and how I approach minimizing that issue (see Section 3.2.3. Finally, in Section 3.2.4, I will give a few details on the training process.

3.2.1. Semantic Segmentation Using Deep Learning

The task this thesis tries to solve falls under the category of **semantic segmentation**. In semantic segmentation, the input is an image, consisting of one or more channels. The goal is that the model classifies each pixel x of the input image $I(x)$ into a class. Since I have pre-defined classes the model is supposed to learn to predict, this falls under supervised classification. The ground truth for training the model is of the same height and width as the input image and labels each pixel with its corresponding class. The output is referred to as a (segmentation) mask $\hat{M}(x)$, which consists of one binary channel per class.

Since the task at hand is a binary classification task, the goal is to label each pixel of an input satellite image as either 1 (positive class) or 0 (negative class or background), which refers to "damaged through armed conflict" and "undamaged through armed conflict", respectively.

While classical machine learning methods can be used to achieve semantic segmentation, deep learning has performed much better when it comes to most image-related classification tasks. In particular, methods based on CNN have become the norm for semantic segmentation problems (Bressan et al., 2022), which is why I focus on simple implementations of a few frequently applied CNN-based architectures, which I will detail in the following.

CNNs as the Basis CNNs are a special form of neural networks, which have become very common to solve pattern recognition tasks in images in general since they allow "to encode image-specific features into the architecture" (O'Shea & Nash, 2015, p. 2). A CNN consist of a convolutional layer, a pooling layer and a non-linear activation function. In the convolutional layer (e.g. 3×3) kernels, which are as deep as their input[17], slide over the input image and multiply the kernel values with the values from their receptive field. Each kernel in a convolutional layer produces a different feature map, potentially extracting a different feature of the input. These kernel values are learnable, i.e. their values can adapt through backpropagation to extracting the features that are appropriate for distinguishing the different classes. After an activation function is applied to every feature map, "[p]ooling layers aim to gradually reduce the dimensionality of the representation, and thus further reduce the number of parameters and the computational complexity of the model" (O'Shea & Nash, 2015, p. 8). Typical CNNs then have one or several fully connected layers at the end to combine the information from the features that comprise an image into a prediction of the class label for the whole input image. By calculating and backpropagating the loss in the end, like with classical neural networks, the kernels can be trained to extract appropriate features.

FCN While CNNs are used to predict e.g. the label of the whole image, FCNs were developed specifically to solve tasks like semantic segmentation end to end. FCNs take an input of arbitrary size and produce an output of the same size. To achieve this there needs to be an upsampling part (= decoder) after the convolution (= downsampling or encoder) layers, since they reduce the size of the image. However, since the features got reduced so much during the downsampling steps, the result of upsampling using only interpolation techniques would be quite rough. Therefore, Long, Shelhamer, and Darrell (2015) introduced an idea to circumvent this issue and make FCNs a suitable choice for semantic segmentation tasks. They introduce a connection between higher and lower layers so that the net can "learn to combine coarse, high layer information with fine, low layer information" (Long et al., 2015, p. 3435). Like this, the net can combine the higher-level features that recognize the *what* with the information from the beginning when the input still had a higher resolution to keep information on the *where*. Long et al. (2015) showed that you could also augment image classification networks into FCNs to benefit from their already learned representations. To do so, they "decapitate[d] each net by discarding the final classifier layer, and convert[ed]

[17] This means that when the input is, for example, an RGB image, i.e. has 3 channels, the depth of the kernel will be 3 as well.

all fully connected layers to convolutions. [They] append[ed] a 1×1 convolution [...] to predict scores for each of the [...] classes (including background) at each of the coarse output locations, followed by a deconvolution layer to bilinearly upsample the coarse outputs to pixel-dense outputs" (Long et al., 2015, p. 3435).

U-Net The U-net is a modification of an FCN, developed by Ronneberger, Fischer, and Brox (2015) for fast and precise segmentation of images. It also contains an encoder and a decoder part, but while a FCN only contains one upsampling layer as the decoder, a U-net is symmetrical in the sense that it has as many upsampling as downsampling layers. These upsampling layers are also called transposed convolutions and can be trained as well. Furthermore, it introduces skip connections between the down- and upsampling layers. By combining higher resolution features with the upsampled output, features can be localized better. "A successive convolution layer can then learn to assemble a more precise output based on this information" (Ronneberger et al., 2015, p. 235). On top as the final layer comes a 1×1 convolution, which is used to map each feature vector to the desired number of classes (Ronneberger et al., 2015).

ResNet The idea of ResNet came to be to solve the vanishing gradient problem when working with very deep nets (He, Zhang, Ren, & Sun, 2016). For deeper networks, where layers are simply stacked on top of each other it can become difficult to backpropagate the error appropriately up until the first layers. So the idea of He et al. (2016) was to introduce shortcut connections, i.e. every couple of layers a possible skip in the form of an identity function is introduced, which adds to a higher overall derivative of a residual block. Thereby, unimportant layers can be skipped easily if they do not contribute to the layer performance.

3.2.2. Network Architectures

In the following, I will detail how I implemented the four different network architectures I chose to apply to this task.

Pretrained FCN with ResNet Backbone For my application, I selected the pre-trained version of the **FCN** that PyTorch offers. To do so, I just needed to adapt the first layer to fit the number of input dimensions of the images I feed into the net as well as the number of output classes I wanted it to predict by replacing the final layer (see Figure 11). Choosing a pre-trained model, in this case, means that the returned model has been pre-trained on

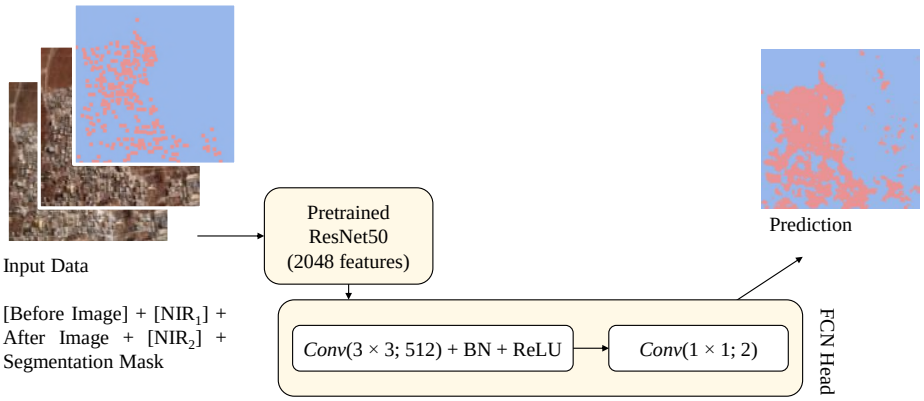


Figure 11: Visualization of FCN with pre-trained ResNet50 backbone.

COCO train2017, which is a large image dataset, among else, for semantic segmentation (Lin et al., 2014). It contains the same classes as Pascal VOC, which is the dataset Long et al. (2015) used in their paper. Working with a pre-trained version means that this model is not initialized with random weights but weights that have been trained on COCO. Therefore, it already has some understanding of image features in general and will only be fine-tuned with the data I feed it. This so-called "transfer learning" is common practice since it is quite rare to have a dataset of sufficient size (Chilamkurthy, 2022), which is the case for my project as well. The FCN Pytorch provides has a ResNet-50 backbone, which means that the encoder part of the FCN consists of a 50-layer ResNet.

U-net While a pre-trained network could be used for the encoding part of the network, U-net has the advantage that "such a network can be trained end-to-end from very few images" (Ronneberger et al., 2015, p. 234), which is why I initialized it without any pre-trained weights. Furthermore, instead of alternating convolution and pooling layers, I only used strided convolutions and did not include any pooling steps similar to Springenberg, Dosovitskiy, Brox, and Riedmiller (2014), who showed that convolutions with an increased stride outperform max-pooling with regards to several image-recognition benchmarks. Since I am working with a rather small dataset, I kept the net quite small with four convolutional and four deconvolutional layers to prevent it from overfitting and also losing less resolution (see Figure 12).

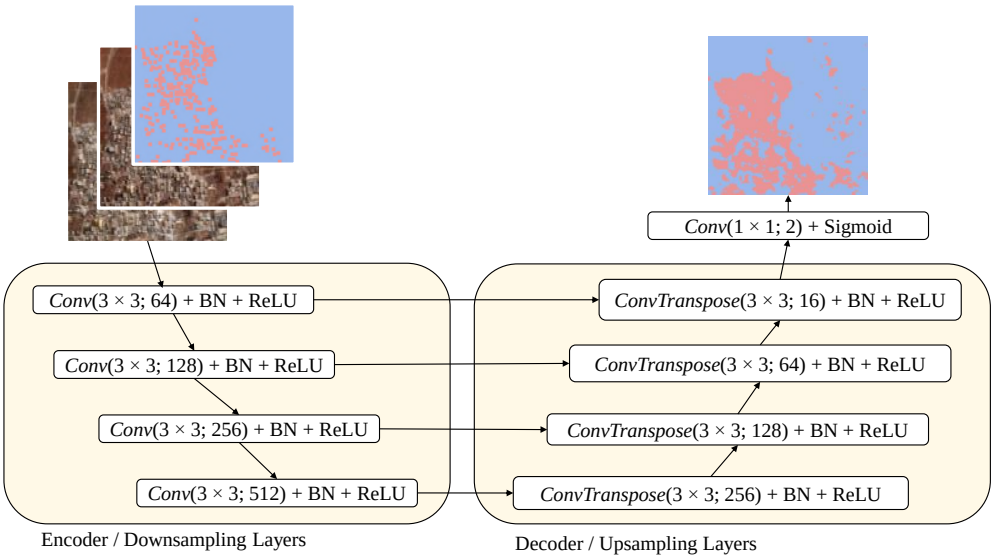


Figure 12: Visualization of simplified U-Net implementation.

Simplified 6-Layers ResNet Finally, I also implemented a simplified 6-layers ResNet as suggested by Rodriguez and Wegner (2018) (see Figure 13) in their attempt to detect objects of sub-pixel size. I will refer to the model under the name **ResNet6** in the following. In their application, it outperformed other state-of-the-art computer vision models, which hints that this model could prove to be a sensible approach in my case as well since I am working with images of medium resolution, wherefore damages or destruction are likely to at least partly be smaller than the pixel-size of 10×10 meters. Since they wanted "to retain as many details as possible", (Rodriguez & Wegner, 2018, p. 6) set all the striding operations within the net to 1[18]. Like this, the "method moves away from lower dimensional representations but instead keeps details" (Rodriguez & Wegner, 2018, p. 5). This is sensible, especially for remote sensing tasks like the ones at hand, where the objects to be detected are more likely of sub-pixel size or only a few pixels large. This is different from more classical computer vision images (which most of the semantic segmentation models have been designed for), where the objects consist of > 100 pixels.

This is why I also tried out a second U-net, similar to the one explained above but with a stride of 1 instead of 2 throughout the net. In the following, I will refer to them by **stride-1**

[18]The stride defines the step size of the kernel when traversing the image. A stride higher than 1 would result in downsampling of the image.

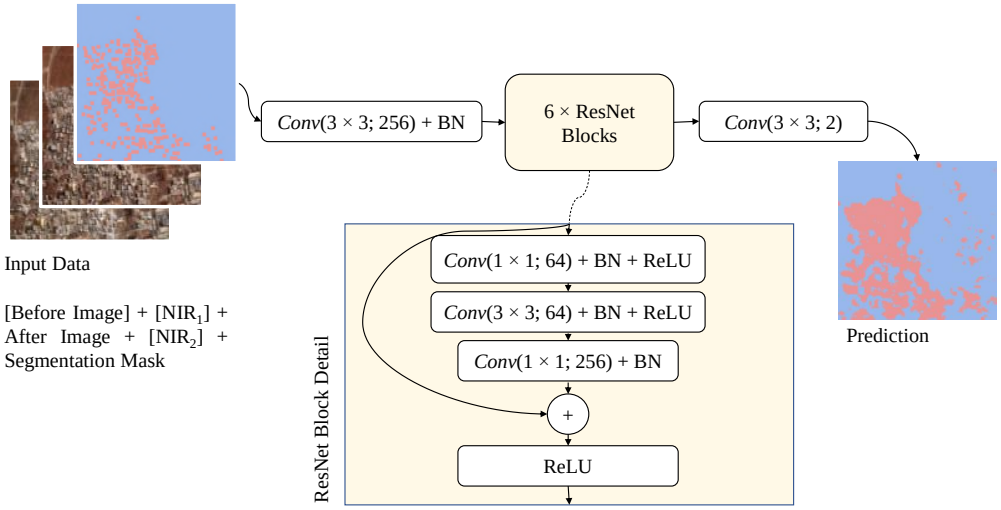


Figure 13: Visualization of simplified 6-layer ResNet.

U-net and **stride-2 U-net** respectively.

3.2.3. Accounting for Class Imbalance

When going for a pixel-wise solution to the problem at hand, it is inherently an imbalanced one. A lot more pixels of each satellite image will remain undamaged than not, leading to "undamaged through armed conflict" being by far the dominant class. This needs to be compensated for as otherwise, it is very likely that the model will only learn to predict the dominant class and not the damages, which are much rarer.

There are several approaches in the literature on how to handle class imbalance when training a classifier. One could apply other loss functions, developed specifically for imbalanced issues, like the focal loss function (as opposed to a global loss function) to give higher weight to the underrepresented "damage" events (Lin, Goyal, Girshick, He, & Dollár, 2017). Another possibility for dealing with class imbalance is to artificially increase the rarer class for the training or undersample the majority class. However, these approaches come with their own pitfalls, since they change the distribution of the data (Dal Pozzolo, Caelen, & Bontempi, 2015). Lastly, one could apply other models like random forest, however, CNN-based architectures tend to perform better in applications on satellite images (Boston, van

Dijk, Larraondo, & Thackway, 2022; Yoo, Han, Im, & Bechtel, 2019).

The simplest option that I can add to the "vanilla" version I already have is to weigh the loss function (which in my case is the cross-entropy loss) to give more emphasis to the positive labels.

This is the basic cross-entropy loss function where y_i is the ground truth label for the ith training example (which is either 0 or 1 in the binary case), and $\hat{y}_i$ is the respective predicted probability (between 0 and 1):

$$CrossEntropyLoss = y_i \cdot -log(\hat{y}_i) + (1 - y_i) \cdot -log(1 - \hat{y}_i)$$

This formula gives equal emphasis to both classes and increases the more the predicted probability differs from the actual label. Therefore, it penalizes the most, when the model was confident in its prediction but did so wrongly.

To adapt this, one can simply add weights to the formula. In PyTorch's implementation of the CrossEntropyLoss, you are supposed to give each of the classes a weight (PyTorch Contributors, 2022). I gave the classes decimal weights adding up to 1. For example, when I gave class 1 a weight of 0.8, the weight of class 0 was $1 - 0.8 = 0.2$. Formally written:

$$CrossEntropyLoss_{weighted} = y_i \cdot -log(\hat{y}_i) \cdot w + (1 - y_i) \cdot -log(1 - \hat{y}_i) \cdot (1 - w)$$

In the following, I will refer to those weights by their ratio. For example, when class 0 was weighted with 0.2 and class 1 was weighted with 0.8, I will refer to that weighting as 2:8. By introducing these weights, false negatives are more heavily penalized, which should help to increase the recall, i.e. help to increase the model's ability to actually recognize positive cases.

3.2.4. Training

In the following, I will go over some details on my training procedure. This includes a reasoning for the split of my data into training-, validation- and test set. Further, I will give on overview of which model specifications I varied across runs and for how long I let them train.

Split into Training-, Validation- and Test-Set In machine learning it is usual practice to split the dataset into a training, a validation, and a test set. The training data is the only

one that is used for training the model, i.e. only the loss of the results of the training data is backpropagated. The thereby adapted model is evaluated on the validation set each epoch, but that does not have an influence on the model itself. The results on the validation set are just used for selecting the model from the epoch which performed best on validation data, which did not influence the model. The performance of the best model of each model specification (based on the results of the model on the validation set) is then tested on the test data once.

Before starting the training, I split my data into two separate datasets – one for training and one for testing. I assigned whole image regions to either of the two in order to be able to test the models on cities they have not seen before. You can see an overview of the different cities present in the data as well as their assignment to either test or training data in Figure 14. In doing so, however, I still kept the different landscapes present in my data in mind and chose three cities for testing that cover a different type of landscape each. This is also important since the different landscapes affect the images. I divided them into three different landscape types:

- **Landscape type 1** is characterized by desert in the surrounding of the city, and the images are quite bright. This holds true for cities in the Syrian desert (Al Quaryatayn, Palmyra, Deir ez-Zor) as well as Mosul. From this landscape type, I selected Deir Ez-Zor for testing.

- **Landscape type 3** is characterized by very dark images. The cities I assigned to this category are Daraa and Damascus, both in the South-West of Syria. From this category, I chose Daraa for testing, since the available images were of higher quality.

- Under **Landscape type 2**, I grouped all the other image regions, whereof the images are neither very bright nor dark. Their surroundings show a mixture of mountains and agricultural areas. The image region I assigned to the test set is Raqqa. The other Syrian cities in this category are Hama, Manbij, Aleppo, and Idlib, which all lie in the North of Syria. The Iraqi cities are Ramadi and Fallujah, which lie in the middle of Iraq, East of the Syrian desert.

This allows me to properly evaluate the models and see how well they perform in areas they have not seen during training while still giving them a chance to learn from data on the different landscape types during the training phase. In the Appendix in Section D, you can see an overview of all the different image regions with exemplary satellite images and to which landscape type I assigned them.

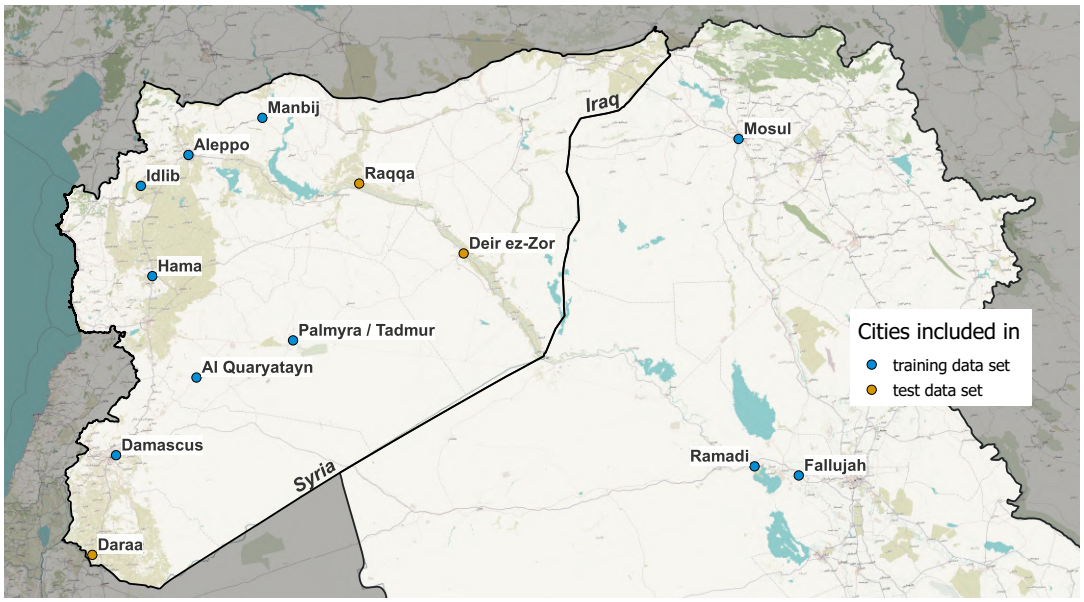


Figure 14: Overview of cities included in the dataset, colored by inclusion in test or training data.

Another reason that made those three cities a good choice for testing is that their distribution of damaged and destroyed areas resembles the one in the rest of the data; the damage percentage is even slightly higher. In the test data, 1.17% of the labeled image pixels are damaged and 0.25% destroyed. This resembles the percentages in the training data, which are at 1.01% and 0.26% respectively. In total, the cities I selected for the testing account for about 20% of all the positive labels.

The AOI with by far the highest percentage of damaged pixels is the inner city of Mosul with 13.57% in July 2017. In August, UNOSAT significantly increased the radius of analysis, which is why then the percentage of damaged pixels lies only at 0.77% since the inner city was the most destroyed. Mosul is part of the training dataset. The AOI with the second highest damage percentage is Raqqa with 2.62% in October 2017. Raqqa is one of the regions selected for testing. The AOI of Palmyra and Tadmur has the lowest damage percentage with 0.13% and is in the training set.

For validation, I randomly selected 20% of the tiles from the training data. So these image tiles were not used for training the models per se but were sections from within the images used for training.[19]

[19]The split was performed using PyTorch's `random_split()` function. Since I set a seed, the tiles selected

Varying Model Specifications In addition to testing the performance of the different model architectures, several different model specifications were evaluated (see Appendix Section 7 for an overview of all combinations). As for the image basis fed to the model, it was varied whether only to include the RGB channels or also the NIR channel. Similarly, the model specifications differed as to whether they only included the *after* (damage) image or whether the *after image* was stacked on top of the *before image*. As discussed in Section 3.2.3, the specifications also varied as to whether the loss function included a weighting. The weighting parameter furthermore varied between a weighting of 2:8 and 1:9. In addition, the label definition varied across specifications: per damage coordinate from UNOSAT, either one ("real" labels) or nine pixels ("increased" labels) around that coordinate were labeled as damaged, as discussed in Section 3.1.4. Finally, the definition of the positive label itself was varied to refer to either any damage or only destruction.

For the optimization, the gradient-based Adam algorithm (Kingma & Ba, 2014) was used. The learning rate (LR) varied between 0.01, 0.001, and 0.0001 for each specification and the best performing one was chosen.

The batch size was set at 32 for all models, except the U-net with stride 1 and the 6-Layers ResNet model. For these, the batch size was set to 8 since they would have consumed too much memory otherwise.

Training Time Furthermore, I let the runs go for 400 epochs, but included early stopping in a way that when the validation loss was not better than the previously best loss for 20 epochs in a row, the run would stop, since this indicates overfitting to the training data. For some runs, the stopping criterion did not work as intended, since they got stopped even though the F1-scores of training and validation still were on an upwards trend. These runs were identified by a manual selection and reran without a stopping criterion in order to be sure not to stop them prematurely. When a model was rerun, the earlier version which stopped prematurely was disregarded in the following steps. Which of the model specifications was run without the stopping criterion is visible in the Appendix in Section 7.

for validation were the same for all different model specifications.

4. Results

4.1. Evaluation Metrics

When assessing and comparing the performance of different models, it is essential to choose appropriate metrics, since that choice influences model evaluation and thereby model selection (Ferri, Hernández-Orallo, & Modroiu, 2009). Therefore, I will briefly justify my selection of evaluation metrics in the following. I will focus on threshold metrics (Ferri et al., 2009).

The confusion matrix is the basis of most performance metrics (Luque, Carrasco, Martín, & de las Heras, 2019). In confusion matrices, elements m_{ij} are subdivided into different groups, dependent on their actual class i and the class they were classified into j. For the binary case, it can be formally written as:

$$CM = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix}$$

or as

$$CM = \begin{bmatrix} m_{PP} & m_{PN} \\ m_{NP} & m_{NN} \end{bmatrix},$$

whereby one of the classes is referred to as the "positive" and the other as the "negative" class (Luque et al., 2019).

This results in four different sub-cases (see Table 3), which build the basis for the metrics in question: m_{PP}, which are commonly called the "True Positives" (TP), i.e. elements that belong to the positive class and have also been predicted as such; m_{PN}, the "False Negatives" (FN), are the ones that should have been predicted as positive but have not been; and similarly m_{NP}, the "False Positives" (FP), have been predicted as positive but should not have been; and m_{NN}, the "True Negatives" (TN), have been predicted correctly as negative.

	Predicted Class	
Actual Class	**P**	**N**
P	TP	FN
N	FP	TN

Table 3: Confusion matrix.

Among the most common metrics, using the confusion matrix values is the Overall Accuracy (OA), which is a measure of how well the classifier was able to predict the classes correctly across all classes. For a binary classification problem, this can formally be written as:

$$OA = \frac{TP + TN}{TP + FN + TN + FP}.$$

The best possible accuracy score is 1, which means that all examples are correctly classified. When no example was correctly classified, the accuracy score is 0. While it is a widely used metric, it is evident that this measure is highly biased toward the majority class (Berthold, Borgelt, Höppner, Klawonn, & Silipo, 2020). With a majority class percentage of 99%, the accuracy score would yield an almost perfect accuracy score of 0.99, when the model only predicted the dominant class. The large imbalance inherent to the task at hand is the reason why accuracy is not a good evaluation metric for this use case. Therefore, the accuracy metric is not sufficient to get a comprehensive picture of the performance of the algorithms.

Other commonly used metrics, are Precision, Recall, and the harmonic mean of the two, which is called the F1-score. Precision is a measure of how many of the examples predicted as positive are actually positive. Recall measures how many of the actual positive examples, the classifier was able to "catch". They can be written formally as:

$$Precision = \frac{TP}{TP + FP}$$

and

$$Recall = \frac{TP}{TP + FN}.$$

Since the focus of these measures is on the positive class, they are better suited for selecting a model which is actually able to detect examples from the positive class, which in my case is in the minority. From these two metrics, recall is the one that shows no bias when applied on imbalanced datasets (Luque et al., 2019) since it only considers examples that are actually positive. Considering only the recall, however, could lead to a model, which is predicting the positive class excessively or even only predicts the positive class in the worst case, since recall only captures how many of the positive examples got predicted as such. A higher precision, however, often means that the recall is going down since it penalizes false positive predictions. This trade-off between precision and recall is well-known (Buckland &

Gey, 1994).

While for some applications it is clear, which of the two metrics is more desirable and which false prediction, i.e. FP or FN, is easier to tolerate, it is not that clear for the task I want to solve. A model, which is catching all of the actual armed conflict events on the ground (high recall), is desirable for an early-warning tool since you do not want to miss any events. If this comes, however, with a lot of false alarms, i.e. false negatives (low precision), this would mean that a lot of alarms by the tool need to be checked on the ground or using images of higher resolution, leading to a lot of unnecessary additional workload. Furthermore, when the results of such a model should be used for the creation of a dataset for research purposes, false positive predictions would make the dataset almost unusable, since these predicted events would simply be wrong. Hence, a high precision score is really important, both for research and practice. If this results, however, in too low of a recall value, i.e. catches almost no events, this would make the tool obsolete.

Hence, I will base my model selection on the F1-score, which is based on the harmonic mean of the two and is thereby best suited for indicating the real model performance (Jeni, Cohn, & de La Torre, 2013) even for skewed data. It can be formally written as:

$$\textit{F1-score} = 2 \cdot \frac{\textit{Precision} \cdot \textit{Recall}}{\textit{Precision} + \textit{Recall}}.$$

However, I will also measure precision, recall, and accuracy throughout my runs since they are important for thoroughly assessing the performance of the models.

4.1.1. Two Different Metric Definitions for Specifications with Increased Labels

When working with the "increased" labels, I measure the evaluation metrics in two ways:

Metric Type A First, for being able to compare the results of the model specifications to each other no matter whether they worked with "increased" or "real" labels, I calculate the evaluation metrics based solely on the "real" labels. Since I would not want to punish damage predictions in the vicinity of the "real" labels, I ignore the pixels that got only labeled due to the increase when forming the respective metrics (see Figure 15). Thereby, the recall calculation basis is equal as with specifications without an increase since the recall is only based on the TP and the FN values. For the precision value, there are fewer potential pixels "available" that the model could predict falsely as positive since I ignored the ones around the pixels with the "real" labels. However, this is just as intended, since I would not want to rate a model less based on those specific FPs (which were positive just because

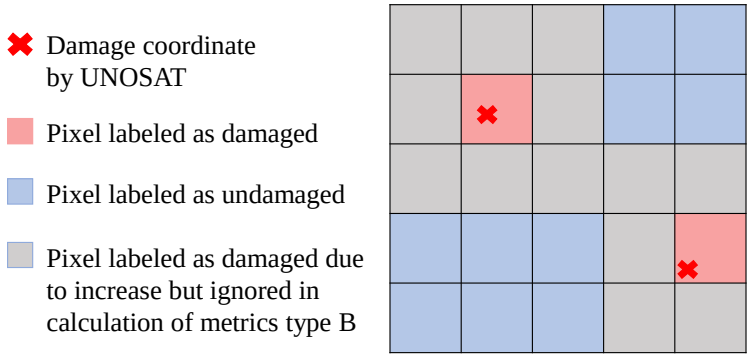


Figure 15: Schematic visualization of Metric Type B for specifications based on "increased" labels.

of the label increase), since those were the ones it was trained to detect as well. Since the F1-score is based on precision and recall, it will perform as intended as well.

Metric Type B Second, I also calculate the metrics based on the "increased" labels and assume those to be the ground truth. The F1-score based on the "increased" labels is also the one that is used for selecting the epoch with the best weights from each model specification (which includes the "increased" labels) since it best measures how well a model worked on those "increased" labels. The precision score based on the "increased" labels is particularly interesting, since it shows how many of the predictions fell into the area of the "increased" labels. For recall, the metric based on the "real" labels is more relevant, since of these, I am more sure that they are actually positive.

4.2. Experimental Results

In this section, the results of the different model specifications will be discussed.

4.2.1. Baseline

As a primitive baseline reference, I recorded the evaluation results when predicting at random, when only predicting the majority class (undamaged) as well as when only predicting the minority class (damaged). As expected, the random prediction recalled about half of the positive labels by chance since it equally predicted 0s and 1s as you can see in Figure 16. As you can see in Table 4, the F1-score of predicting either at random or only the

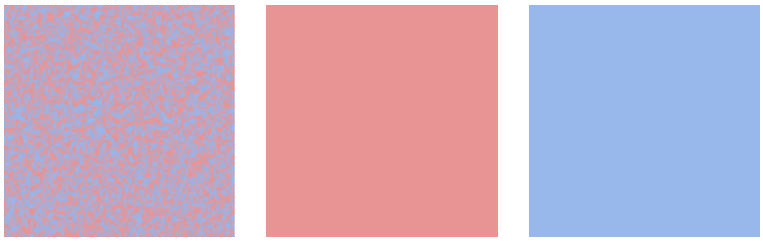


Figure 16: Example predictions of baseline references.

Baseline	Precision	Recall	F1-score	Accuracy
Random	0.012	0.502	**0.023**	0.500
Majority (0)	0	0	0	**0.988**
Minority (1)	0.012	**1**	**0.023**	0.012
1×1 convolution, no weight	**0.017**	0.001	0.003	0.987
1×1 convolution, weight 2:8	0.011	**1**	**0.023**	0.012

Bold font indicates the best score for that metric.

Table 4: Results of primitive baseline on the test set.

positive class is similar at 0.023. The metrics when only predicting the majority class are all 0, except for the accuracy, which is at 0.988. This showcases why the accuracy is an important metric to include in the evaluation of the overall results but is not appropriate for model selection, since it achieves almost perfect results when the model only predicts a region as undamaged.

A simple pixel-wise classification based on an n-channeled input image can be achieved by a 1×1 convolutional filter. This results in an output image of the same height and width as the input image but only one channel with, in this case, the class predictions for each pixel. After three epochs at most, this pixel-wise classification began to only predict 0s during training, so the best model was from epoch 1. When applied to the test data, there were a few positive predictions, which resulted in a F1-score of 0.003, which is even worse than the results from the random or minority prediction. When introducing a weighting[20] it almost only predicted 1s.

These results are helpful in the sense that I can directly discard model specifications with results equal to or worse than those achieved by the baseline. In the following, I will present a selection of results from the different model specifications. The complete list of results can

[20]The weighting applied was 2:8 for the labels 0 and 1 respectively.

be found in the Appendix Section 8. I will mainly focus on the results of the test set but mention the results of the validation and training set if they are helpful for understanding the performance.

4.2.2. Vanilla Model Specifications

First, I want to introduce the results of the "vanilla" models, i.e. the models with the least amount of information, and no adjustments to the labels or the loss function. This means that these models are working only with:

- the *after* images (no *before* images),
- the RGB channels (not the NIR channel),
- all damage categories included in the specification of label 1 (as opposed to only destruction),
- the "real" labels (as opposed to the "increased" labels),
- and no weighting applied to the loss function (as opposed to including a weighting of either 2:8 or 1:9 to counteract the class imbalance).

As you can see in the first row of Table 5, none of the models really learned to detect any damage. While at first, it seemed like all of the models would just predict all pixels to be undamaged, after more than 200 epochs, the stride-2 U-Net at least achieved an almost perfect F1-score on the training set, and in the 306th epoch its best F1-score on the validation set with 0.293. However, when applying this model to the test set, the F1-score is only 0.010, i.e. even worse than if the model only predicted the minority class or at random. The best-performing architecture in the "vanilla" model round was ResNet6, which at least predicted 635 pixels correctly as damaged. Nevertheless, the results are still only very slightly higher than that of the random or minority prediction baseline. The other architectures either predict no damaged pixels at all or only very few and the majority of them falsely.

4.2.3. Introducing One Variation at a Time

Next, I will describe how the results change when adapting single parameters as compared to the "vanilla" specification. The results can also be found in Table 5.

Including the **NIR** channel into the data helped the ResNet6 model to increase its F1-score slightly to 0.031. Again, the best epoch was 396, so training for even longer might

	FCN			stride-2 U-Net			stride-1 U-Net			ResNet6		
	Pr.	Rec.	F1	Pr.	Rec.	F1	Pr.	Rec.	F1	Pr.	Rec.	F1
Vanilla	0.003	0.001	0.001	0.015	0.008	0.010	0.006	0.001	0.001	**0.035**	**0.021**	**0.026**
NIR	0.098	0.000	0.000	0.022	0.010	0.013	0.006	0.011	0.007	**0.038**	**0.027**	**0.031**
Bef.	0.025	0.060	**0.036**	0.030	0.005	0.009	**0.042**	0.021	0.028	0.024	**0.071**	**0.036**
2:8	0.020	0.008	0.011	0.016	0.008	0.010	0.039	**0.070**	0.050	**0.047**	0.060	**0.053**
1:9	0.031	0.016	0.021	0.021	0.010	0.014	**0.050**	0.125	**0.071**	0.046	**0.140**	0.070
Incr.	0.039	0.137	0.061	0.021	0.091	0.034	0.056	**0.247**	**0.091**	**0.059**	0.130	0.081

Pr.: precision, Rec.: recall, F1: F1-score; bold font indicates the best score for that metric per specification.

Table 5: Results from model specifications using the "vanilla" architecture or when changing one parameter as compared to the "vanilla" version.

help with model performance. Models built from the other architectures also saw a slight increase in performance but the scores (except for the accuracy of course) were still very low since they mostly just predicted pixels to be undamaged. On the validation set, the stride-2 U-Net performed best with an F1-score of 0.354.

By **including** a ***before image***, the two models working with a stride of 1, i.e. the stride-1 U-Net and ResNet6, as well as the FCN achieved an F1-score slightly higher than the baseline. However, the curve of both the validation as well as the training F1-score indicates that the performance might have increased when they had trained for even longer than 400 epochs. While the U-Net with a stride of 2 performed the best in the "vanilla" round, and in this specification also performed best on the validation set its results on the test set were the worst of all the models and failed to surpass the baseline results.

Introducing a **weighting** of 2:8 into the loss function, helped both the stride-1 models to achieve an F1-score of over 0.05. The other two architectures performed below the baseline levels. With a weighting of 0.9, however, all models achieved scores above the baseline. ResNet6 and stride-1 U-net manage to recall about 13% of all damaged pixels, but only about 5% of their damage predictions are true. The stride-2 U-Net performed the lowest on the test set again. However, on the validation set, the stride-2 U-Net achieved the best results again, followed by the FCN.

When working with **"increased" labels**, all models achieve the best results from the different specifications so far. The stride-1 models perform best again, with the U-Net achieving an F1-score of 0.091 and ResNet6 getting a score of 0.081. while the stride-2 U-Net scores worst again. These were the results based on the "real" labels, i.e. the pixels that only got labeled due to the label increase, got ignored in the calculation of these metrics (see Section 4.1.1, Metric Type A). When looking at the results based on the "increased" labels instead (see Section 4.1.1, Metric Type B), the scores achieved are significantly better

	Specification	Real labels (Metric Type A) Precision	Recall	F1-score	Increased labels (Metric Type B) Precision	Recall	F1-score
best performing on the "real" labels							
ResNet6	85	**0.089**	0.306	**0.137**	**0.377**	0.279	0.321
Stride-1 U-net	13	0.084	0.318	0.133	0.365	0.293	0.325
FCN	29	0.055	0.231	0.089	–	–	–
Stride-2 U-net	14	0.037	0.187	0.062	0.198	0.176	0.186
best performing on the "increased" labels							
ResNet6	50	0.060	0.471	0.106	0.290	0.443	**0.350**
Stride-1 U-net	86	0.055	**0.489**	0.100	0.272	**0.456**	0.341
FCN	32	0.047	0.391	0.085	0.245	0.374	0.296
Stride-2 U-net	14	0.037	0.187	0.062	0.198	0.176	0.186

Bold font indicates the best score for that metric – one time based on the "real" labels and one time based on the "increased" labels. For a more detailed definition of the difference between these two metrics, see Section 4.1.1.

Table 6: Selection of the best performing results for each architecture.

– the stride-1 models, for example, achieve F1-scores of about 0.25. On the validation set, the FCN and the stride-2 U-Net both achieved the highest F1-score with with about 0.2.

When using the label definition of **destruction instead of damage**, all of the previously laid out model specifications perform below the baseline.

4.2.4. Best Performing Variations

While the introduction of each of the variations compared to the "vanilla" specification has already improved the results at least a little, there is still much room for improvement. Therefore, I will go into more detail about the results of the different combinations of variations and the respective results. I will only give examples of some of the combinations, most of which can be found in Table 6. The complete results of all four architectures with all 71 model specifications can be found in the Appendix in Section 8.

The best performing model is the ResNet6 specification no. 85 with an F1-score of 0.149. It recalled about a third of the "real" labels, and about 9% of its damage predictions (precision score for the "real" labels) were correct. It did not include a weighting but the NIR channel as well as the "increased" labels. These were the only adaptions from the "vanilla" specification. The training time was quite long since the selected model (based on the F1-score on the validation set) was from epoch 261. When looking at the metrics based on the "increased" labels (see Metric Type B), the recall was a bit lower (as expected) but the precision score was a lot higher with about 38% of the damage predictions having been true.

When taking a look at the actual images from the predictions, however, the pattern of the predictions actually resembles the ground truth (with the "increased" labels) quite well. In Figure 17, the top three rows show such examples, where the predicted distribution is actually close to the ground truth mask. Large-scale destruction was particularly well recognized by the model. In general, the false positive predictions are rarely far off from actual damage in the ground truth. Often they are spatially close to true positive predictions. Moreover, the pattern from the predictions seems to be more continuous, while ground truth labels are often more sparsely distributed. Therefore, the pixel-wise calculation of the metrics applied in this thesis might underestimate the actual model performance. Predictions like the ones shown here would already be very useful for both a monitoring tool as well as a dataset. The bottom two rows in Figure 17 show exemplary image tiles where the model did not perform as well. In most of these cases, the model predicted too little damage. It seemed to be more challenged by images with only sparsely distributed damage as well as images from the city border with fewer settlements. In general, I would say that predicting too little damage, in favor of being more certain about what is predicted is favorable for both applications.

Most of the best-performing specifications of the ResNet6 architecture worked with the "increased" labels. All of them included the NIR channel but almost none of them included *before images*. In contrast to the best model, a slight majority of them included a weight in the loss function.

For the stride-1 U-net, the best specification, which was no. 13, performed very similarly to ResNet6_85 in terms of all metrics. The specification of the model is also similar, except for the LR, which was at 0.01 instead of 0.0001.[21] In general, the label increase was really important for the performance of stride-1 U-nets as well as the inclusion of the NIR channel, while the inclusion of *before images* even tended to worsen the result. The weighting of the loss function was really important for model performance when working with the "increased" labels.

The stride-2 U-net achieved an F1-score of 0.062 on the "real" labels (see Metric Type A) in its best specification. When looking at the metrics based on the "increased" labels (see Metric Type B), the performance on precision, recall, and F1-score is pretty evenly at about 0.2, i.e. it recalls about 20% of the damages and about 20% of its damage predictions are correct. Next to the "increased" labels, this specification also included the NIR channel, and the loss function was weighted by 2:8. In fact, all of the best stride-2 U-net specifications trained on the "increased" labels, and most included the NIR channel. Weighting also

[21] With such a low LR, ResNet6 only correctly detected 17 "real" and 20 "increased" labels.

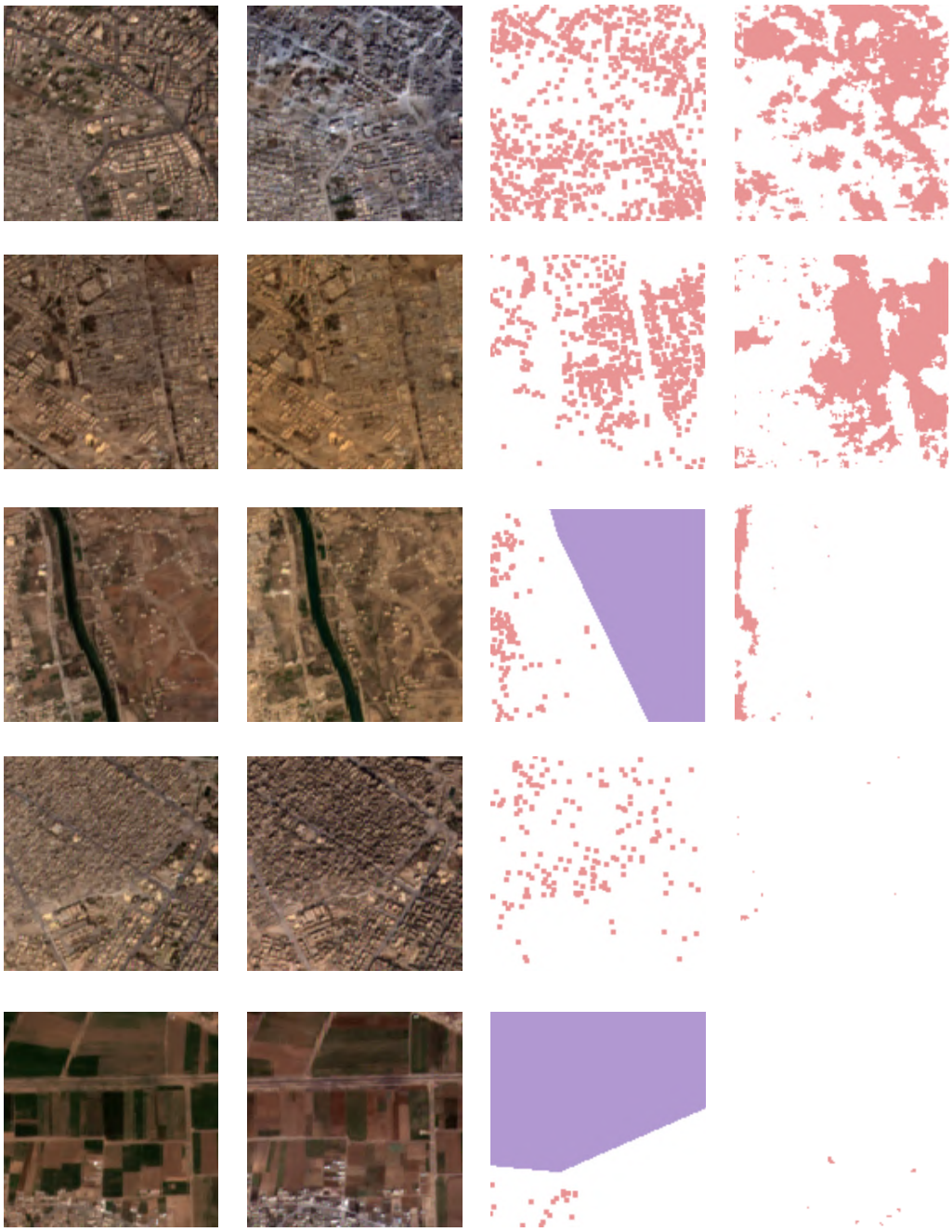


Figure 17: Visualization of the performance of the best model from the experiments: ResNet6_85.

*The *before image* is just shown for illustration purposes. It was not fed into this model specification. As for the *before* and *after image*, only the RGB channel is visualized, not the NIR channel.

tended to be positively correlated. As for the other parameters, there is no clear correlation to the model performance.

When looking at the FCN, the best model (no. 29) achieves only an F1-score of 0.089, which translates into 7032 "real" true positives. It included a *before image*, the NIR channel, and a weighting of 2:8 but only the "real" labels. When investigating the several best performing model specifications, all, except for *the* best, had incorporated the "increased" labels. Including the *before images* and the weighting of the loss function seemed to help the performance in general.

When one looks at the best results on the "increased" labels, i.e. based on the Metric Type B, the stride-1 models performed best as well. The specification of ResNet6 which includes the "increased" labels, a weighting of 2:8, the NIR channels, and a LR of 0.001 (no. 50) performed the best. About a third of its predictions have been true and it recalls about half of both the "real" and the "increased" damage pixels. This translates into 14,356 correctly predicted "real", and 77,670 correctly predicted "increased" damaged pixels, while 225,801 pixels have been wrongly classified as damaged.

5. Discussion

5.1. Discussing Different Model Specifications and their Influence on the Results

When looking at the overall results of the models on the test data, some trends become apparent. The two factors contributing most to a good performance were when a model architecture was set up with a stride of 1 and when they have been trained on the "increased" instead of the "real" labels. The first finding underlines the importance of avoiding losing resolution when applying deep learning methods to satellite images, especially when they are of moderate resolution.

Other parameters slightly positively correlated with model performance were incorporating a weighting in the loss function, including the NIR channel, and working with a lower LR. When looking at the F1-score based on the "increased" labels (see Metric Type B), the inclusion of the weighting is actually highly correlated with a better score though, and the higher it was, the better.

When taking all models into account, including a *before image* was even negatively correlated with the F1-score. If you break the F1-score measure down to its components though, it becomes evident that while including the *before image* worsened the recall and the preci-

sion score based solely on the "real" labels (Metric Type A), it helped with precision when taking into account the "increased" labels as well (Metric Type B). A higher recall was achieved in general with model specifications that included a weighting, "increased" labels as well as a higher LR.

Interestingly, when looking at the results on the validation set, including a *before image*, not applying a label increase as well as working with the stride-2 U-net architecture are the specifications that perform best. The ResNet6 architecture as well as the FCN architecture in particular have been negatively correlated to the F1-score.

So there seem to be two different types of specifications to be working best. One is working with a stride-2 U-net, the "real" labels, and a *before image*, maybe the NIR channel but no weighting. These models are achieving some really good results on the validation set after a while. However, these results do not translate into good results on the test set. Quite the contrary, stride-2 U-net models are performing worst on the test set, and "real" labels as well as *before images* tend to lead to worse results in most specifications. On the test set, the specifications performing best worked with one of the two stride-1 architectures, i.e. the U-net or ResNet6, "increased" labels, only the *after image*, the NIR channel, and maybe a weighting. These best-performing models on the test set were not among the best on the validation set. This indicates that what the models were learning on the training set, did not always transfer to the models' predictive capability in general but only on regions from the training and validation set – especially with longer training times. The long training times in general for all architectures but the FCN, however, probably stem from the fact that these models were trained from scratch with only little training data, and that the task itself is not that easy.

In general, when setting the F1-score from the test set in relation to the metrics on the validation set of that model specification, there are some interesting insights. The best performing models on the test set either had a high recall on the validation set, or precision and recall have been quite balanced at validation, but the precision score was never higher than the recall score. Only the poorly performing models (on the test set) had a higher precision than recall during validation. This relation stems mainly from the recall score on the test set tending to be linked to the recall from the validation round. The relation between the precision of the predictions on the test set to recall and precision during validation is unclear. This indicates that a high recall during training is important when wanting the model to perform well on unseen data as well.

Especially the fact that including a *before image* did not seem to help predictions was unexpected. Since the stride-1 models were the main ones that did not perform well with the

before images hints that the pixel-wise differences between *before* and *after image* were not that indicative of conflict-related damage on the ground. These models seem to have picked up on pixel values indicative of a damaged structure from the *after image* better than when being supplied data from two different dates and learning from the difference between the two. Since the model, which was including more abstraction due to a higher stride (stride-2 U-net) profited from the before image, however, it might be that when loosening the single pixel constraint, the *before images* were actually able to provide some helpful additional information. This inference is supported by the fact that when looking at the metrics based on the "increased" labels (Metric Type B), the inclusion of a *before image* slightly improved the precision score. In general, however, that including *before images* sometimes even worsened the results indicates that they were not perfectly suited. This could either stem from the fact that they were not actually from before the respective conflict, i.e. the structure identified as damaged by UNOSAT might have already been damaged at the time the *before image* was taken. On another note, the *before images* not being helpful could also be because they are from too long before the damage took place. Since I did not do any pre-training on e.g. seasonal differences, there might be a lot of other differences between *before* and *after image*, which are "confusing" the model instead of contributing to its ability to detect the conflict-related damages. In future research, this could maybe be alleviated by working with multi-temporal images, shortly after each other and around the time the damages probably occurred. Like this, the changes between images would be easier attributable to conflict damages. Moreover, it could be worth it to think about working with a different model architecture in general, where before and *after image* are not just stacked on top of each other but where they actually are compared to each other more explicitly because change detection is built into the architecture.

While I originally incorporated a weighting into the loss function to counteract the extreme class imbalance, the results of including it into a specification are not as unambiguous as I thought. When starting off with the "vanilla" model, including a weighting definitely helped all models to achieve slightly better results and especially to not only predict the majority class. When adjusting more parameters in the specification, however, working with a weighted loss function in general still led to slightly better F1-scores due to a better recall (as expected) but tended to lead to a worse precision of the model's predictions. So the weighting's influence on this trade-off has to be kept in mind in future refinements of this project. Experimenting with further different weights might help to find a weighting that supports a good balance between precision and recall. Otherwise, if precision was deemed to be more important, dropping the weight from the loss function completely could

be an option as well. As evident from the results of this thesis' trials, the best-performing model did not include a weighting as well.

The fact that increasing the labels helped train the models to predict damages more correctly, indicates that the assumption, that the damages do not necessarily coincide only with the pixel their coordinate pair lies in, was correct. When there are damages evident on the satellite images that are not labeled as such, that makes it more difficult for the model to correctly learn to detect such damages, when sometimes they are labeled (because the UNOSAT point falls within a pixel) and sometimes they are not (because the point referring to a building as damaged is only in the neighboring pixel). So increasing the labels seems to have helped to alleviate this issue of label uncertainty by decreasing the number of falsely negative labeled pixels in the ground truth at the cost of potentially increasing the number of false positives in the ground truth segmentation mask. Furthermore, it probably has indirectly helped to counteract the extreme class imbalance a bit, since it led to proportionally more pixels being labeled as damaged in the reference data. Interestingly, in contrast to the weighting of the loss function, increasing the labels did not worsen the precision score of the models. It did not lead to more predictions just in general and thereby also more false positives, but it actually improved the precision as well as the recall. This is why increasing the labels might have diminished the need for the weighting of the loss function and why the effect of weighting became less.

5.1.1. Ideas for Enhancements

There are several potential enhancements to the methods I used that were not within the scope of this paper, but that could improve further developments of this project beyond working with more advanced methods in general. I will briefly describe those ideas for future steps in the following.

Tackling the Label Uncertainty In my approach, I tried to solve the issue of task-inherent label uncertainty by increasing the labels. This simple solution already showed great success, since the models predicted a lot of the increases as damage, while not losing precision, which would hint that the models just predict more damage in general. This indicates that the coding of not just one but nine pixels around a damage coordinate from UNOSAT (see Figure 9) in itself comes closer to the actual damage patterns on the ground. In future versions of this project, this label increase could probably be refined though, for example by taking the damage type into account, since destruction is more likely to be visible on more than one pixel than moderate damage. When the label definition is not binary anymore, its

variation could also be included in the weighting of the loss function, giving more weight e.g. to destroyed and less to damaged pixels.

Furthermore, one could incorporate an approach already discussed in the literature when working with UNOSAT's damage assessment data, which is the inclusion of building footprints. See for example Lee et al. (2020) or Xu et al. (2019), who "used a building detection machine learning (ML) model to identify all buildings in the damage assessment area and then filtered out all buildings that were marked by UNOSAT analysts as damaged. This approach allowed [them] to generate a large number of negative examples" (Xu et al., 2019, p. 3). Hasnat and Faisal (2015) ran a segmentation method first, to identify infrastructure in general and only thereafter proceeded with the change detection techniques to identify damages. Kahraman et al. (2016a) and Kahraman et al. (2016b) utilized building footprints from the Open Street Map (OSM) project (OpenStreetMap contributors, 2015) for an accurate assessment of building-level changes. They all used VHR images so far but an implementation using medium-resolution images could be experimented with.

Another possibility to circumvent the issue of having to deal with label uncertainty is switching to a patch-classification approach. When classifying patches instead of single pixels, one could measure the model performance based on its prediction for a whole patch (consisting of several pixels) instead of single pixels, where the labeling is potentially faulty per definition anyhow. When creating the ground truth, one could for example work with a threshold and classify patches with a lot of damaged pixels as severely damaged and the ones with only a few damages as slightly damaged. In the evaluation of the models and when working with its results, one could also tie some sort of uncertainty measurement to the density of damages per patch. This would be based on the assumption that when the model predicted a lot of damage within a patch, it is more likely that there actually is damage in that area. In addition, one could think about solving the task with semi-supervised learning like Lee et al. (2020). Their patches covered one building each, making it possible to classify a patch as containing a damaged building when a damage coordinate by UNOSAT laid within a patch. To generate more labeled data, they used a semi-supervised learning technique.

Improve the Before Images Firstly, the preprocessing of the Sentinel-2 images, in general, should be improved in future developments of this project to correct atmospheric effects.[22] These effects could have decreased the helpfulness of the *before images* when differing be-

[22] For example by applying the toolbox from Sentinel, which performs the pre-processing steps necessary to get from Level 1C to 2A and correct those potential errors (European Space Agency, n.d.-f).

tween *before* and *after image*. Furthermore, removing outliers before the normalization could help to improve the image quality and comparability.

Since the inclusion of the *before images* did not help that much with the performance of the models I tested, I think it could be worthwhile to try to improve their quality. One possibility for this could be to rely on other sources for these images that are available at around the same dates as the *before images* that UNOSAT used for their analyses. The wars in Iraq and Syria started before the launch of Sentinel-2, wherefore the *before images* used by UNOSAT also stem from before that time of course. So in order to be sure to work with images from before the conflict (and its damages) happened one could try to supplement this with images from other sources.

Another option would be, instead of working with one explicit *before image* for each *after image*, working with multiple subsequent images could really improve the tool's performance and help in narrowing down the actual date of when the damage happened. To do so, one could work with all the images in between the *before* and *after* date, feed them into the algorithm and let it detect at which point in time the damage happened at the indicated locations. However, this would result in a lot of images to process since the *before* and *after date* are often months to years apart. Another solution I thought of, is to take the time span in between the *before* and *after date* and narrow the time of actual damage down step by step using a divide and conquer approach. By this, I mean that the algorithm could check for whether a change at the indicated locations happened in several larger time spans and then follow up on the ones where this was true to narrow the actual time down step by step. One more option I could think of was to go back in time from the *after date* until the algorithm finds a change in the pixels it was meant to and thereby find out the actual dates of destruction. While the decision on how to tackle this is tricky, I consider it to be an important issue to narrow the possible date of destruction down to a minimum, since this is essential for later application in the social sciences and potential combinations with other conflict event datasets.

Incorporate Additional Information Since the task itself is not that easy, I think it could help to introduce some more information into the training. This could for example come in the form of context information like case knowledge on the conflict the data is about. Here, information about dates of attacks could help as well as information on when warfare started in general and when it arrived in a certain area. This information could even stem from conflict event databases like the ones mentioned in Section 2.1. Their data is by far not as geoprecise but might help to identify dates to take a closer look at and select images

for, for a specific image region. Adding a layer of nighttime lights to each image could also help to improve the results.

Furthermore, it could help to include synthetic-aperture radar (SAR) images, for example, from Sentinel-1. They have proven useful in previous research efforts related to building detection and damage assessments (e.g. Bai et al., 2017; Xiao, You, Gang, Yu, & Xiao-Ping, 2020) and are superior to optical images in the sense that they are not affected by clouds.

In addition, pre-training a model in general or straight-away working with a pre-trained model could really help to boost its performance if it has been pre-trained on relevant data. Therefore, it should be pre-trained on satellite images. As my trials have shown, a model pre-trained on semantic segmentation tasks unrelated to satellite images, was not really helping in increasing the model's performance, since objects in normal photos are different from what is supposed to be detected on those medium-resolution satellite images we are working with here. Objects in normal images usually span a lot of pixels, while the size of what is supposed to be detected in the task at hand ranges from sub-pixel size up to a few pixels. What could help to improve model performance, in particular, would be data on seasonal changes and non-conflict-related destruction (e.g. from natural disasters), so that the algorithm can differentiate those better from conflict-related damages.

Turning the Results into Comprehensive Maps Again Finally, an important step that is missing in my pipeline so far, is the geo-referencation of the final predictions as well as putting the tiles used for training back into one comprehensive image again. This is necessary for being able to create a) a comprehensive dataset from it that can be used for research and b) a monitoring tool in the form of e.g. a map. In my pipeline, this information and the information on the date gets lost at the rasterization step due to how I implemented it. This should be improved in future versions since it is essential for everything the project is supposed to solve.

5.2. Implications for the Applicability in the Social Sciences

My experiments show that already with rather simple methods, deep learning models are able to detect clusters of armed conflict-related damage in urban areas. And they are able to do so even within medium-resolution images when avoiding the loss of resolution within the model architecture as much as possible. However, it also became apparent, that when damages do not cluster but are more sparsely distributed, the semantic segmentation models have a harder time detecting those. Which in turn has the implication that a dataset created using this method will not be applicable for the study of damages related

to every type of conflict. While it was clear from the beginning, that even a deep learning model is not able to detect damages that are not or only barely visible from above, this finding has the further implication that the detection might be biased depending on the type of warfare. More spread-out single attacks, which caused only little damage are likely to remain unnoticed, whereas the destruction of e.g. a whole village or several buildings close to each other is more likely to be detected by the tool. However, further developments of a deep learning solution to this task might be able to detect even these harder cases, i.e. where the damage is not clustered but more spread out.

Since most of the predictions of the best-performing model tended to be in or at least in the vicinity of the damage hotspots, it might make sense not to stick to pixel-wise values when creating a database. Rather, one could create a polygon database containing areas of damage clusters. These could potentially be enriched with a damage density score.

In either case, this thesis has shown that it is in principle possible to automate the detection of armed conflict damages in freely available satellite images of medium-resolution. This opens up entirely new possibilities for creating data at a much lower cost than before. Previously, damage assessments of areas affected by conflicts from remote sensing required high-resolution satellite images, which are not regularly and freely available. When not working with a machine learning approach, time-intensive manual labeling, which does not scale up well, came on top. In contrast, Sentinel-2 products cover almost the whole world every couple of days. So once properly trained on labeled satellite data, a deep learning-based tool could be applied to different regions of the world to screen them for conflict damages. Different landscapes, buildings and climate conditions might require some fine-tuning for the respective local conditions first though. While there is human expertise necessary for the development of the tool as well as in the verification of its results and potential fine-tuning for different contexts, it is in general much better upscalable and enables the monitoring and data generation of vast conflict areas.

Applying the approach elaborated in this thesis (or a variation thereof) for the creation of a database on armed conflict damages could significantly improve the data availability on conflict. Since data like this is only available for specific cases, a comprehensive dataset would allow for the investigation of new research questions. It could fill important research gaps since "there is little deterministic understanding of the ebbs and flows of urban damage occurrence during armed conflict" (Aas Rustad, Buhaug, Falch, & Gates, 2011; Raleigh & Hegre, 2009 as cited in van den Hoek, 2021, p. 330). One potential avenue of research enabled by this data would be to investigate the targeting of civilian infrastructure in war. While there has been a lot of research on why actors target civilians in armed conflict, there

are only case-specific data collections on civilian infrastructure destruction and only limited theory on why, when, and where actors choose to resort to those targets. While the intent when targeting civilian infrastructure might differ as compared to the targeting of civilians, "the strategy of attacking civilian or dual-use infrastructure with the primary purpose of exerting pressure on civilians [...] is neither consistent with the requirements of international law nor ethically defensible along the lines proposed by its advocates" (Thomas, 2006, p. 32).

Ultimately, I think the main issue needing more thought is how to incorporate a form of uncertainty measurement or precision score into the predictions of the model. This is critical, since without this information it is difficult to actually use the data as research evidence, when not knowing how sure of the result the model itself is. In general, the performance of the tool will need to be closely monitored and if possible verified on the ground or by using high-resolution images to prevent the model from having a bias or predicting wrongly unknowingly. Because an inherent bias could also bias the concurrent research, at least when the topic to be investigated is related to the form of the bias in the data. For the performance evaluation, one could check whether the model detected known war-related destruction to make sure that it recalled these events. What I consider to be even more important, however, will be to pay close attention to the model not giving too many "false alarms". If the model would tend to do that, it could potentially lead to a lot of overhead required for checking those false alarms or in the worst case just faulty research results due to a compromised data basis. However, pre-training the model with data on seasonal changes, and other non-war-related changes could help to minimize such false alarms. Furthermore, including building detection into the pipeline could help reduce the false alarms due to changes in non-building areas (Kahraman et al., 2016b). As my application has shown, the false positives, i.e. the false alarms, have mostly not been far off of the actual damage hotspots anyway. A measure of certainty could therefore also be linked to the density of damage predictions by the model in a certain area. If the model predicted a lot of damage in an area, it could be interpreted as the model being more certain of damage within that area as compared to when it only predicted a few pixels as being damaged. My application has shown that when a lot of damage predictions clustered in an area, the predictions were also more likely to resemble the actual damage in that area. This makes sense, since a larger damage cluster means a bigger pattern, which is easier to detect at this moderate resolution. When the model only predicted a few single damages within an area, they tended not to resemble the ground truth, which indicates that such predictions might require more verification and one could interpret them as more uncertain. It would be a pity though if such more insulated events would not be recognized even in

future versions of the tool, since those are also more likely to be missed by the media.

6. Conclusion

The starting point of this thesis was to find out whether it was possible to fill the data gap on armed conflict damages using deep learning and open satellite images, which cover the world regularly but are only of medium resolution. To examine this question, I created a dataset, consisting of satellite images by Sentinel-2, which I labeled by pixel using armed conflict-related damage assessments by UNOSAT on Syria and Iraq. This was used to train and test several different architectures and vary several parameters in their specification.

The results revealed some interesting trends. It is in fact possible to detect damages stemming from armed conflict even in medium-resolution satellite images. While the model predictions may not always be pixel-accurate, the best-performing models were able to detect clusters of such damages quite well. My analysis revealed that to do so, one should not blindly apply common computer vision algorithms (in this case those for semantic segmentation). Unlike with classical segmentation tasks, the damage "objects" to be detected here are only of sub-pixel size or a few pixels large, since working with medium-resolution remote sensing data. Therefore, it is important to keep as much of the resolution as possible by keeping for example all striding operations to 1 as Rodriguez and Wegner (2018) suggested. This became evident in my analysis since the two models I set up with a stride of 1, which were a 6-Layers ResNet and a U-net, performed by far the best on the test set. Model specifications based on a pre-trained FCN with a 50-layers ResNet as the backbone or a U-net with a stride of 2, which both involved more abstraction, performed considerably worse. Another important factor determining a good model performance in my experiments has been to increase the labels from one to nine pixels per damage coordinate from UNOSAT. This shows the importance of labeling sensibly; and since the problem is inherently imbalanced already, my results suggest that it is better to, when in doubt, rather label too many pixels in favor of avoiding the assignment of false negative labels in the ground truth. Another insight gained through my experiments was that it helps the deep learning models when the NIR band is included in the images, which hints that there are changes in these bands apparent when there has been damage related to armed conflicts.

The inclusion of *before images* did not help the models as much as expected. This possibly stems from the images being from different seasons or simply too much time laying in between the *before* and *after image*. Therefore, I suggest switching from a bi-temporal to a multi-temporal approach in future developments of this project. Thereby, one could

exploit the high revisit rate of Sentinel-2 and use the frequent availability of new images to detect differences indicative of damage between images only a few dates apart. This would also help to narrow down the actual date of damage, which is important for both its use in research as well as by humanitarian or human rights organizations.

While there has been a lot of remote sensing research on methods to detect events of interest in satellite imagery, there has been little social science research using such data. Since the methods are available, I think this mainly stems from the missing link between detection frameworks and the subsequent creation and maintenance of a corresponding dataset. Therefore, I consider this to be an essential next step. A very important part herein is to devise an approach to solving the issue of verification and the inclusion of an uncertainty measurement. To do so, an interdisciplinary understanding is essential. Input from peace and conflict research is needed to define what constitutes a meaningful uncertainty measurement or how damage events would need to be verified in order for the resulting data to be useful for research. On the other hand, the technical expertise of remote sensing experts is necessary for a discussion about what is possible from the technical side and how to interpret the model's predictions.

Because in principle, incorporating data that was collected using methods as proposed in this thesis, offers entirely new possibilities and could fill important research gaps related to the spatial development of armed conflicts. In addition, data based on the analysis of remote sensing data allows for a more objective data basis than newspaper-based data on armed conflicts, which is inherently dependent on which events make it into the news.

"However, developing new insights into conflict processes and humanitarian issues will not be achieved simply with more satellite data, greater processing power, or enhanced image analysis algorithm. Rather, it will only be through a coordinated integration of remote sensing, peace and conflict, and humanitarian scholarship that satellite monitoring of urban conflict damage will be able to tell new narratives of conflict and peace and to expose new vulnerabilities as well as unseen resilience of cities and their communities" (van den Hoek, 2021, p. 330f.).

So to gain meaningful new insights on armed conflicts from satellite images of moderate resolution using deep learning, an interdisciplinary effort is necessary for creating a self-contained database generation pipeline as well as a near-real time early warning tool – just as envisaged within the RMAC project. Especially for the data generated on current armed conflicts, ethical concerns related to the publishing of such data will need to be carefully evaluated.

References

Aas Rustad, S. C., Buhaug, H., Falch, Å., & Gates, S. (2011). All Conflict is Local: Modeling Sub-National Variation in Civil Conflict Risk. *Conflict Management and Peace Science*, *28*(1), 15–40. doi: 10.1177/0738894210388122

ACLED (Ed.). (2020). *Armed Conflict Location & Event Data Project (ACLED) Codebook, 2020.* Retrieved 12.09.2022, from `https://acleddata.com/acleddatanew//wp-content/uploads/2021/11/ACLED_Codebook_v1_January-2021.pdf`

Avtar, R., Kouser, A., Kumar, A., Singh, D., Misra, P., Gupta, A., ... Rimba, A. B. (2021). Remote Sensing for International Peace and Security: Its Role and Implications. *Remote Sensing*, *13*(3). doi: 10.3390/rs13030439

Bai, Y., Gao, C., Singh, S., Koch, M., Adriano, B., Mas, E., & Koshimura, S. (2017). A framework of rapid regional tsunami damage recognition from post-event TerraSAR-X imagery using deep neural networks. *IEEE Geoscience and Remote Sensing Letters*, *15*(1), 43–47.

Baumann, M., Radeloff, V. C., Avedian, V., & Kuemmerle, T. (2015). Land-use change in the Caucasus during and after the Nagorno-Karabakh conflict. *Regional Environmental Change*, *15*(8), 1703–1716. doi: 10.1007/s10113-014-0728-3

Berthold, M. R., Borgelt, C., Höppner, F., Klawonn, F., & Silipo, R. (2020). *Guide to Intelligent Data Science: How to Intelligently Make Use of Real Data* (2nd ed.). Springer Cham. doi: 10.1007/978-3-030-45574-3

Bevington, J. S., Eguchi, R. T., Gill, S., Ghosh, S., & Huyck, C. K. (2015). A Comprehensive Analysis of Building Damage in the 2010 Haiti Earthquake Using High-Resolution Imagery and Crowdsourcing. In C. D. Lippitt, D. A. Stow, & L. L. Coulter (Eds.), *Time-Sensitive Remote Sensing* (pp. 131–145). New York, NY: Springer New York. doi: 10.1007/978-1-4939-2602-2_9

Bjorgo, E. (2000). Using very high spatial resolution multispectral satellite sensor imagery to monitor refugee camps. *International Journal of Remote Sensing*, *21*(3), 611–616.

Boston, T., van Dijk, A., Larraondo, P. R., & Thackway, R. (2022). Comparing CNNs and Random Forests for Landsat Image Segmentation Trained on a Large Proxy Land Cover Dataset. *Remote Sensing*, *14*(14), 3396. doi: 10.3390/rs14143396

Braun, A. (2018). Assessment of building damage in Raqqa during the Syrian civil war using time-series of radar satellite imagery. *Geographic Information Science*, *1*, 228–242.

Bressan, P. O., Junior, J. M., Martins, J. A. C., de Melo, M. J., Gonçalves, D. N., Freitas, D. M., ... Gonçalves, W. N. (2022). Semantic segmentation with labeling uncertainty

and class imbalance applied to vegetation mapping. *International Journal of Applied Earth Observation and Geoinformation*, *108*, 102690. doi: 10.1016/j.jag.2022.102690

Bromley, L. (2010). Relating violence to MODIS fire detections in Darfur, Sudan. *International Journal of Remote Sensing*, *31*(9), 2277–2292. doi: 10.1080/01431160902953909

Buckland, M., & Gey, F. (1994). The relationship between Recall and Precision. *Journal of the American Society for Information Science*, *45*(1), 12–19.

Chilamkurthy, S. (2022). *Transfer Learning for Computer Vision.* Retrieved 18.08.2022, from `https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html#transfer-learning-for-computer-vision-tutorial`

Clionadh, R., Linke, A., Hegre, H., & Karlsen, J. (2010). Introducing ACLED - Armed Conflict Location and Event Data. *Journal of Peace Research*, *47*(5), 651–660.

Dal Pozzolo, A., Caelen, O., & Bontempi, G. (2015). When is Undersampling Effective in Unbalanced Classification Tasks? In A. Appice, P. P. Rodrigues, V. S. Costa, C. Soares, J. Gama, & A. Jorge (Eds.), *Machine Learning and Knowledge Discovery in Databases* (pp. 200–215). Cham: Springer International Publishing.

Eck, K. (2012). In data we trust? A comparison of UCDP GED and ACLED conflict events datasets. *Cooperation and Conflict*, *47*(1), 124–141. Retrieved from `http://www.jstor.org/stable/45084688`

El-Gamily, H. I. (2007). Utilization of multi-dates LANDSAT_TM data to detect and quantify the environmental damages in the southeastern region of Kuwait from 1990 to 1991. *International Journal of Remote Sensing*, *28*(8), 1773–1788.

European Space Agency (Ed.). (n.d.-a). *Applications.* Retrieved 05.09.2022, from `https://sentinel.esa.int/web/sentinel/user-guides/sentinel-2-msi/applications`

European Space Agency (Ed.). (n.d.-b). *Level-1.* Retrieved 05.09.2022, from `https://sentinel.esa.int/web/sentinel/user-guides/sentinel-2-msi/processing-levels/level-1`

European Space Agency (Ed.). (n.d.-c). *Overview.* Retrieved 05.09.2022, from `https://sentinel.esa.int/web/sentinel/user-guides/sentinel-2-msi/overview`

European Space Agency (Ed.). (n.d.-d). *Revisit and Coverage.* Retrieved 05.09.2022, from `https://sentinel.esa.int/web/sentinel/user-guides/sentinel-2-msi/revisit-coverage`

European Space Agency (Ed.). (n.d.-e). *Sentinel-2 operations.* Retrieved 05.09.2022, from `https://www.esa.int/Enabling_Support/Operations/Sentinel-2_operations`

European Space Agency (Ed.). (n.d.-f). *The Sentinel-2 Toolbox.* Retrieved 05.09.2022, from

https://sentinel.esa.int/web/sentinel/toolboxes/sentinel-2

European Space Agency (Ed.). (n.d.-g). *Spatial Resolution.* Retrieved 19.08.2022, from https://sentinels.copernicus.eu/web/sentinel/user-guides/sentinel-2-msi/resolutions/spatial

Ferri, C., Hernández-Orallo, J., & Modroiu, R. (2009). An experimental comparison of performance measures for classification. *Pattern Recognition Letters*, *30*(1), 27–38.

Giada, S., de Groeve, T., Ehrlich, D., & Soille, P. (2003). Information extraction from very high resolution satellite imagery over Lukole refugee camp, Tanzania. *International Journal of Remote Sensing*, *24*(22), 4251–4266.

Gillies, S. (2013). *Rasterio: geospatial raster I/O for Python programmers.* Retrieved from https://github.com/rasterio/rasterio

Gorelick, N. (2021). *Google Earth Engine API (earthengine-api).* Retrieved 10.08.2022, from https://github.com/google/earthengine-api

Gorelick, N., Hancher, M., Dixon, M., Ilyushchenko, S., Thau, D., & Moore, R. (2017). Google Earth Engine: Planetary-scale geospatial analysis for everyone. *Remote Sensing of Environment*. doi: 10.1016/j.rse.2017.06.031

Gorsevski, V., Kasischke, E., Dempewolf, J., Loboda, T., & Grossmann, F. (2012). Analysis of the Impacts of armed conflict on the Eastern Afromontane forest region on the South Sudan—Uganda border using multitemporal Landsat imagery. *Remote Sensing of Environment*, *118*, 10–20.

Gupta, R., Goodman, B., Patel, N., Hosfelt, R., Sajeev, S., Heim, E., ... Gaston, M. (2019). Creating xBD: A dataset for assessing building damage from satellite imagery. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops* (pp. 10–17).

Gwak, J., Blevins, S., & Nabel, R. (2016). *PyDrive.* Google Inc. Retrieved 20.08.2022, from https://pythonhosted.org/PyDrive/

Hasnat, K., & Faisal, K. M. (2015). Segmentation and Classification Using Logistic Regression in Remote Sensing Imagery. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *8*(1), 224–232. doi: 10.1109/JSTARS.2014.2362769

Hassan, M. M., Smith, A. C., Walker, K., Rahman, M. K., & Southworth, J. (2018). Rohingya Refugee Crisis and Forest Cover Change in Teknaf, Bangladesh. *Remote Sensing*, *10*(5). doi: 10.3390/rs10050689

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).

Höglund, K., Melander, E., Sollenberg, M., & Sundberg, R. (2016). Armed conflict and space: exploring urban-rural patterns of violence. In *Spatializing Peace and Conflict* (pp. 60–76). Springer.

Ireland, G., Volpi, M., & Petropoulos, G. P. (2015). Examining the capability of supervised machine learning classifiers in extracting flooded areas from Landsat TM imagery: a case study from a Mediterranean flood. *Remote Sensing*, *7*(3), 3372–3399.

Jenerowicz, M., Kemper, T., Pesaresi, M., & Soille, P. (2010). Post-event damage assessment using morphological methodology on 0.5 m resolution satellite data. *Italian Journal of Remote Sensing*, *42*(3), 37–47.

Jeni, L. A., Cohn, J. F., & de La Torre, F. (2013). Facing Imbalanced Data Recommendations for the Use of Performance Metrics. *International Conference on Affective Computing and Intelligent Interaction and workshops : [proceedings]*, 245–251. doi: 10.1109/ACII.2013.47

Jordahl, K., van den Bossche, J., Fleischmann, M., Wasserman, J., McBride, J., Gerard, J., ... Leblanc, F. (2020). *geopandas/geopandas: v0.8.1.* Zenodo. doi: 10.5281/zenodo.3946761

Kahraman, F., Imamoglu, M., & Ates, H. (2016a). Battle Damage Assessment based on self-similarity and contextual modeling of buildings in dense urban areas. In *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)* (pp. 5161–5164). doi: 10.1109/IGARSS.2016.7730345

Kahraman, F., Imamoglu, M., & Ates, H. (2016b). Disaster Damage Assessment of Buildings Using Adaptive Self-Similarity Descriptor. *IEEE Geoscience and Remote Sensing Letters*, *13*(8), 1188–1192. doi: 10.1109/LGRS.2016.2574960

Kemper, T., Jenerowicz, M., Gueguen, L., Poli, D., & Soille, P. (2011). Monitoring changes in the Menik Farm IDP camps in Sri Lanka using multi-temporal very high-resolution satellite data. *International Journal of Digital Earth*, *4*(sup1), 91–106.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Lang, S., Tiede, D., Hölbling, D., Füreder, P., & Zeil, P. (2010). Earth observation (EO)-based ex post assessment of internally displaced person (IDP) camp evolution and population dynamics in Zam Zam, Darfur. *International Journal of Remote Sensing*, *31*(21), 5709–5731.

Lee, J., Xu, J. Z., Sohn, K., Lu, W., Berthelot, D., Gur, I., ... Kowatsch, B. (2020). Assessing Post-Disaster Damage from Satellite Imagery using Semi-Supervised Learning Techniques. *CoRR*, *abs/2011.14004*.

Li, X., Chen, F., & Chen, X. (2013). Satellite-Observed Nighttime Light Variation as Evidence for Global Armed Conflicts. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *6*(5), 2302–2315. doi: 10.1109/JSTARS.2013.2241021

Li, X., Liu, S., Jendryke, M., Li, D., & Wu, C. (2018). Night-Time Light Dynamics during the Iraqi Civil War. *Remote Sensing*, *10*(6). doi: 10.3390/rs10060858

Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision* (pp. 2980–2988).

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... Zitnick, C. L. (2014). Microsoft COCO: Common Objects in Context. In D. Fleet, T. Pajdla, B. Schiele, & T. Tuytelaars (Eds.), *Computer Vision – ECCV 2014* (pp. 740–755). Cham: Springer International Publishing.

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3431–3440).

Luque, A., Carrasco, A., Martín, A., & de las Heras, A. (2019). The impact of class imbalance in classification performance metrics based on the binary confusion matrix. *Pattern Recognition*, *91*, 216–231. doi: 10.1016/j.patcog.2019.02.023

Main-Knorn, M., Pflug, B., Louis, J., Debaecker, V., Müller-Wilm, U., & Gascon, F. (2017). Sen2Cor for Sentinel-2. In L. Bruzzone (Ed.), *Image and Signal Processing for Remote Sensing XXIII* (Vol. 10427, p. 1042704). SPIE. doi: 10.1117/12.2278218

Marx, A., Windisch, R., & Kim, J. S. (2019). Detecting village burnings with high-cadence smallsats: A case-study in the Rakhine State of Myanmar. *Remote Sensing Applications: Society and Environment*, *14*, 119–125.

Minority Rights Group International. (2020, January 21). *Two years after 'liberation,' civilians in Mosul denied justice, reparations – new report.* Retrieved 09.09.2022, from `https://minorityrights.org/2020/01/21/mosul-after-battle/`

Moradi, M., & Shah-hosseini, R. (2020). Earthquake Damage Assessment Based on Deep Learning Change Detection Method Using VHR Images. In *Environmental Sciences Proceedings* (Vol. 5, p. 8545). doi: 10.3390/IECG2020-08545

Mueller, H., Groeger, A., Hersh, J., Matranga, A., & Serrat, J. (2021). Monitoring war destruction from space using machine learning. *Proceedings of the National Academy of Sciences*, *118*(23). doi: 10.1073/pnas.2025400118

Nedal, D., Stewart, M., & Weintraub, M. (2020). Urban concentration and civil war.

Journal of Conflict Resolution, *64*(6), 1146–1171.

OpenStreetMap contributors (Ed.). (2015). *OpenStreetMap (OSM).* Retrieved 20.09.2022, from `https://www.openstreetmap.org`

O'Shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*.

Overton, I., & Dathan, J. (2019, December 17). *Syria in 2020: the deadly legacy of explosive violence and its impact on infrastructure and health.* Retrieved 09.09.2022, from `https://aoav.org.uk/2019/syria-in-2020-the-deadly-legacy-of-explosive-violence-and-its-impact-on-infrastructure-and-health/`

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d. Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 32* (pp. 8024–8035). Curran Associates, Inc. Retrieved from `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`

Principe, E. R. (2020). *Google Earth Engine tools (gee_tools).* Retrieved 13.08.2022, from `https://github.com/gee-community/gee_tools`

Prins, E. (2008). Use of low cost Landsat ETM+ to spot burnt villages in Darfur, Sudan. *International Journal of Remote Sensing*, *29*(4), 1207–1214. doi: 10.1080/01431160701730110

PyTorch Contributors (Ed.). (2022). *CrossEntropyLoss.* Retrieved 24.08.2022, from `https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html#crossentropyloss`

Raleigh, C., & Hegre, H. (2009). Population size, concentration, and civil war. A geographically disaggregated analysis. *Political Geography*, *28*(4), 224–238. doi: 10.1016/j.polgeo.2009.05.007

Rodriguez, A. C., & Wegner, J. D. (2018). Counting the uncountable: Deep semantic density estimation from space. In *German Conference on Pattern Recognition* (pp. 351–362).

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention* (pp. 234–241).

Rudner, T. G. J., Rußwurm, M., Fil, J., Pelich, R., Bischke, B., Kopačková, V., & Biliński, P. (2019). Multi3Net: segmenting flooded buildings via fusion of multiresolution, mul-

tisensor, and multitemporal satellite imagery. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 33, pp. 702–709).

Sowers, J. L., Weinthal, E., & Zawahri, N. (2017). Targeting environmental infrastructures, international law, and civilians in the new Middle Eastern wars. *Security Dialogue*, *48*(5), 410–430. doi: 10.1177/0967010617716615

Springenberg, J. T., Dosovitskiy, A., Brox, T., & Riedmiller, M. (2014). Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*.

Stevens, K., Campbell, L., Urquhart, G., Kramer, D., & Qi, J. (2011). Examining complexities of forest cover change during armed conflict on Nicaragua's Atlantic Coast. *Biodiversity and Conservation*, *20*(12), 2597–2613. doi: 10.1007/s10531-011-0093-1

Sundberg, R., & Melander, E. (2013). Introducing the UCDP Georeferenced Event Dataset. *Journal of Peace Research*, *50*(4), 523–532. doi: 10.1177/0022343313484347

The HDF Group. (2000-2010). *Hierarchical data format version 5.* Retrieved from `http://www.hdfgroup.org/HDF5`

The pandas development team. (2020). *pandas-dev/pandas: Pandas.* Zenodo. doi: 10.5281/zenodo.3509134

Thomas, W. (2006). Victory by Duress: Civilian Infrastructure as a Target in Air Campaigns. *Security Studies*, *15*(1), 1–33. doi: 10.1080/09636410600666238

United Nations Satellite Centre (Ed.). (n.d.). *UNOSAT Rapid Mapping Service.* Geneva. Retrieved 24.02.2022, from `https://www.unitar.org/maps/unosat-rapid-mapping-service`

United Nations Satellite Centre. (2017). *Damage assessment of Mosul, Ninawa Governorate, Iraq.* Geneva. Retrieved 29.08.2022, from `https://www.unitar.org/maps/map/2738`

United Nations Satellite Centre, & REACH (Eds.). (2019, March 16). *Syrian Cities Damage Atlas - Eight Year Anniversary of the Syrian Civil War: Thematic assessment of satellite identified damage.* Retrieved 09.09.2022, from `https://reliefweb.int/report/syrian-arab-republic/syrian-cities-damage-atlas-eight-year-anniversary-syrian-civil-war`

van den Hoek, J. (2021). The City is the Medium and Satellite Imagery Are a Prism: 15. In X. X. Yang (Ed.), *Urban Remote Sensing* (pp. 325–333). John Wiley & Sons, Ltd. doi: 10.1002/9781119625865.ch15

Vetrivel, A., Gerke, M., Kerle, N., Nex, F., & Vosselman, G. (2018). Disaster damage detection through synergistic use of deep learning and 3D point cloud features derived from very high resolution oblique aerial images, and multiple-kernel-learning. *ISPRS*

Journal of Photogrammetry and Remote Sensing, *140*, 45–59.

Weidmann, N. B. (2015). On the Accuracy of Media-based Conflict Event Data. *Journal of Conflict Resolution*, *59*(6), 1129–1149. doi: 10.1177/0022002714530431

Weidmann, N. B. (2016). A Closer Look at Reporting Bias in Conflict Event Data. *American Journal of Political Science*(60), 206–218. doi: 10.1111/ajps.12196

Witmer, F. D. W. (2008). Detecting war–induced abandoned agricultural land in north-east Bosnia using multispectral, multitemporal Landsat TM imagery. *International Journal of Remote Sensing*, *29*(13), 3805–3831. doi: 10.1080/01431160801891879

Witmer, F. D. W., & O'Loughlin, J. (2011). Detecting the Effects of Wars in the Caucasus Regions of Russia and Georgia Using Radiometrically Normalized DMSP-OLS Nighttime Lights Imagery. *GIScience & Remote Sensing*, *48*(4), 478–500. doi: 10.2747/1548-1603.48.4.478

Witmer, F. D. W., & O'Loughlin, J. (2013). Satellite Data Methods and Application in the Evaluation of War Outcomes: Abandoned Agricultural Land in Bosnia-Herzegovina After the 1992–1995 Conflict. In A. Kobayashi (Ed.), *Geographies of Peace and Armed Conflict* (pp. 222–233). Routledge.

Xiao, J., You, H., Gang, L., Yu, L., & Xiao-Ping, Z. (2020). Building Damage Detection via Superpixel-Based Belief Fusion of Space-Borne SAR and Optical Images. *IEEE Sensors Journal*, *20*(4), 2008–2022. doi: 10.1109/JSEN.2019.2948582

Xu, J. Z., Lu, W., Li, Z., Khaitan, P., & Zaytseva, V. (2019). *Building Damage Detection in Satellite Imagery Using Convolutional Neural Networks.* arXiv. doi: 10.48550/ARXIV.1910.06444

Yang, L., & Cervone, G. (2019). Analysis of remote sensing imagery for disaster assessment using deep learning: a case study of flooding event. *Soft Computing*, *23*(24), 13393–13408.

Yoo, C., Han, D., Im, J., & Bechtel, B. (2019). Comparison between convolutional neural networks and random forest for local climate zone classification in mega urban areas using Landsat images. *ISPRS Journal of Photogrammetry and Remote Sensing*, *157*, 155–170. doi: 10.1016/j.isprsjprs.2019.09.009

A. Details on Implementation of Image Retrieval

To script the retrieval of the satellite images of interest, I worked with the GEE-related Python packages. I worked with the packages `earthengine-api` (Gorelick, 2021), which is the package for interaction with the GEE API, `geetools` (Principe, 2020), which allows downloading a batch of images at once, and `PyDrive` (Gwak, Blevins, & Nabel, 2016), which simplifies many common Google Drive API tasks like download and deletion. To do so, I created a pandas (The pandas development team, 2020) dataframe of all the polygons and respective dates I want images for (see remarks in Section 3.1.3 on date selection). Polygons were provided in the form of a nested list since those can be turned into a polygon format the GEE understands. I wrote the function in a way that you can indicate for how long before or after the indicated dates you want to retrieve images. The function then loops through all the polygon-date pairs and retrieves all the available images. Since it is only possible to export the images to your Google Drive Account, I additionally scripted the immediate download from Drive to my local disc and the subsequent deletion from Drive.

As for the *before images*, I just added the earliest available date, which is the 23rd of June 2015, as a column to the `pandas` dataframe and then downloaded all images up to six months after that date for each image region.

B. Details on Implementation of Rasterization

I implemented the rasterization of the damage information using the `rasterio.features` (Gillies, 2013) Python module, which allows you to create a raster of the features you want of the size of an input image you specify. The `rasterize` function enables you to match georeferenced images together with georeferenced vector data and automatically lays them on top of each other correctly to rasterize the vector information (in my case: the coordinates of damages). The final dataset was stored as a hierarchical data format (HDF) (The HDF Group, 2000-2010) file, which is a common file format in computer vision and semantic segmentation applications

C. Details on Implementation of Pytorch Dataset

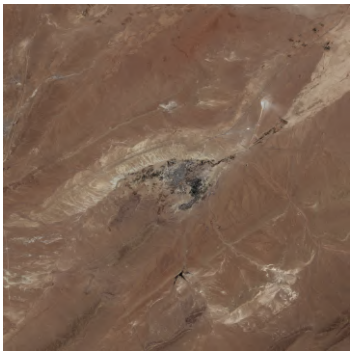
Since I worked with my own dataset, I had to define certain functions for the `torch.utils.data.Dataset` and my own `torch.utils.data.DataLoader` for PyTorch (Paszke et al., 2019). The Dataset I defined in a way, that you can choose a window size, e.g. 128×128 pixels, and

the `__getitem__()` function then returns a sample of that size. When the original image size is not evenly dividable by the indicated window size, the edges of the image get padded, and the segmentation mask gets padded with 99 to indicate the absence of data. The `__len__()` function gets adjusted to reflect the number of such data samples that my custom Dataset can yield. To ignore samples without any information on them, i.e. with only pixels labeled as 99, I create a mapping of indices that skips those samples plus I adjust the number of data samples given out by the `__len__()` function accordingly when first creating the Dataset.

D. Example Images of each Image Region

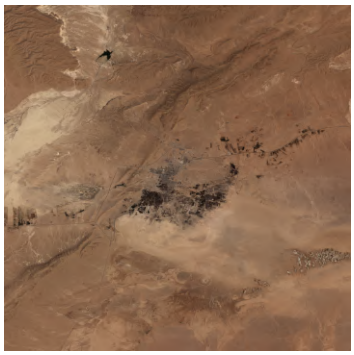
Deir ez-Zor (image region), selected for test

zoomed in



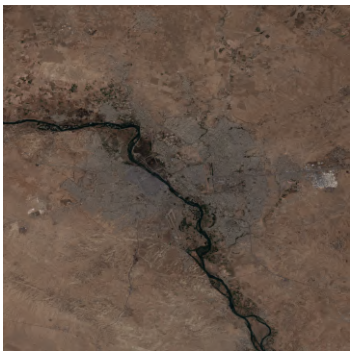
Al Quaryatayn (image region)

zoomed in



Palmyra (image region)

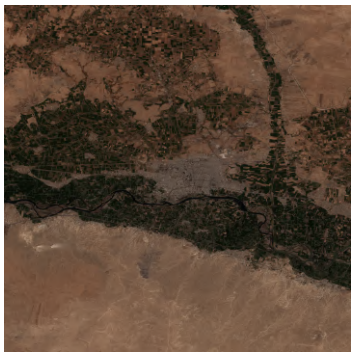
zoomed in



Mosul (image region)

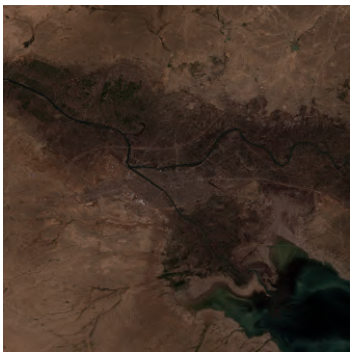
zoomed in

Figure 18: Image regions in landscape type 1; characterized by bright images and desert in its surroundings.



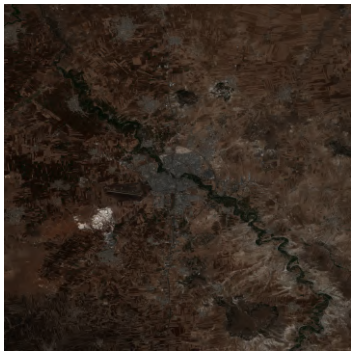
Raqqa (image region), selected for test

zoomed in



Ramadi (image region)

zoomed in

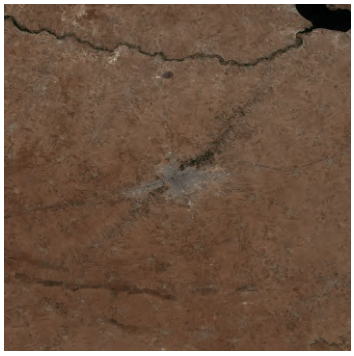


Hama (image region)

zoomed in

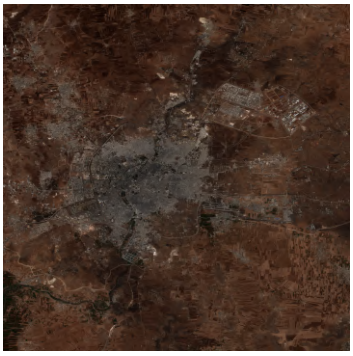
Fallujah (image region)

zoomed in



Manbij (image region)

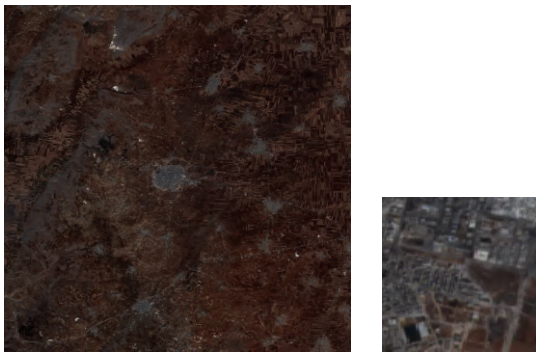
zoomed in



Aleppo (image region)

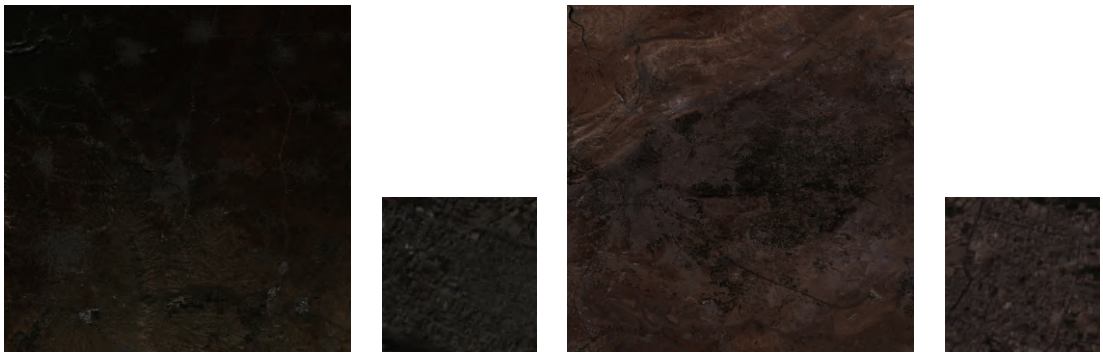
zoomed in

Figure 19: Image regions in landscape type 2; characterized by medium bright images and a mixture of agriculture and mountains in its surroundings.



Idlib (image region) zoomed in

Figure 19: Image regions in landscape type 2; characterized by medium bright images and a mixture of agriculture and mountains in its surroundings (cont.).



Daraa (image region), selected for test zoomed in Damascus (image region) zoomed in

Figure 20: Image regions in landscape type 3; characterized by very dark images.

E. Model Specifications

Table 7: Overview of all model specifications that were experimented with.

Specification Type	Learning Rate	Before Image	Only RGB	Labels	Weighted	Weight Class 1
1	0.0100	False	True	-	False	0.5
2	0.0100	False	True	-	True	0.8
3	0.0100	False	True	-	True	0.9
4	0.0100	False	True	simple increase	False	0.5
5	0.0100	False	True	simple increase	True	0.8
6	0.0100	False	True	simple increase	True	0.9
10	0.0100	False	False	-	False	0.5
11	0.0100	False	False	-	True	0.8
12	0.0100	False	False	-	True	0.9
13	0.0100	False	False	simple increase	False	0.5
14	0.0100	False	False	simple increase	True	0.8
15	0.0100	False	False	simple increase	True	0.9
19	0.0100	True	True	-	False	0.5
20	0.0100	True	True	-	True	0.8
21	0.0100	True	True	-	True	0.9
22	0.0100	True	True	simple increase	False	0.5
23	0.0100	True	True	simple increase	True	0.8
24	0.0100	True	True	simple increase	True	0.9
28	0.0100	True	False	-	False	0.5
29	0.0100	True	False	-	True	0.8
30	0.0100	True	False	-	True	0.9
31	0.0100	True	False	simple increase	False	0.5
32	0.0100	True	False	simple increase	True	0.8
33	0.0100	True	False	simple increase	True	0.9
37	0.0010	False	True	-	False	0.5
38	0.0010	False	True	-	True	0.8
39	0.0010	False	True	-	True	0.9
40	0.0010	False	True	simple increase	False	0.5

Continued on next page

Specification Type	Learning Rate	Before Image	Only RGB	Labels	Weighted	Weight Class 1
41	0.0010	False	True	simple increase	True	0.8
42	0.0010	False	True	simple increase	True	0.9
46	0.0010	False	False	-	False	0.5
47	0.0010	False	False	-	True	0.8
48	0.0010	False	False	-	True	0.9
49	0.0010	False	False	simple increase	False	0.5
50	0.0010	False	False	simple increase	True	0.8
51	0.0010	False	False	simple increase	True	0.9
55	0.0010	True	True	-	False	0.5
56	0.0010	True	True	-	True	0.8
57	0.0010	True	True	-	True	0.9
58	0.0010	True	True	simple increase	False	0.5
59	0.0010	True	True	simple increase	True	0.8
60	0.0010	True	True	simple increase	True	0.9
64	0.0010	True	False	-	False	0.5
65	0.0010	True	False	-	True	0.8
66	0.0010	True	False	-	True	0.9
67	0.0010	True	False	simple increase	False	0.5
68	0.0010	True	False	simple increase	True	0.8
69	0.0010	True	False	simple increase	True	0.9
73	0.0001	False	True	-	False	0.5
74	0.0001	False	True	-	True	0.8
75	0.0001	False	True	-	True	0.9
76	0.0001	False	True	simple increase	False	0.5
77	0.0001	False	True	simple increase	True	0.8
78	0.0001	False	True	simple increase	True	0.9
82	0.0001	False	False	-	False	0.5
83	0.0001	False	False	-	True	0.8
84	0.0001	False	False	-	True	0.9
85	0.0001	False	False	simple increase	False	0.5
86	0.0001	False	False	simple increase	True	0.8

Continued on next page

Specification Type	Learning Rate	Before Image	Only RGB	Labels	Weighted	Weight Class 1
87	0.0001	False	False	simple increase	True	0.9
91	0.0001	True	True	-	False	0.5
92	0.0001	True	True	-	True	0.8
93	0.0001	True	True	-	True	0.9
94	0.0001	True	True	simple increase	False	0.5
95	0.0001	True	True	simple increase	True	0.8
96	0.0001	True	True	simple increase	True	0.9
100	0.0001	True	False	-	False	0.5
101	0.0001	True	False	-	True	0.8
102	0.0001	True	False	-	True	0.9
103	0.0001	True	False	simple increase	False	0.5
104	0.0001	True	False	simple increase	True	0.8
105	0.0001	True	False	simple increase	True	0.9

Models with the architecture ResNet6 and the stride-1 U-net were run with a batch size of 8, and models run with the architecture FCN or a stride-2 U-net were run with a batch size of 32.

F. All Results

	Test ("real" labels)				Test ("increased" labels)				Validation			Best	w/o
Specification	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Epoch	stop
ResNet6_85	0.089	0.306	0.137	0.952	0.377	0.279	0.321	0.906	0.110	0.309	0.162	261	True
UnetStride1_13	0.084	0.318	0.133	0.948	0.365	0.293	0.325	0.903	0.160	0.196	0.176	368	-
ResNet6_32	0.079	0.228	0.118	0.957	0.350	0.210	0.263	0.906	0.058	0.197	0.089	19	-
ResNet6_84	0.089	0.170	0.117	0.970	-	-	-	-	0.136	0.166	0.150	165	True
ResNet6_49	0.104	0.132	0.117	0.975	0.409	0.116	0.180	0.916	0.139	0.162	0.150	351	-
ResNet6_50	0.060	0.471	0.106	0.900	0.290	0.443	0.350	0.869	0.092	0.497	0.155	282	True
UnetStride1_84	0.066	0.252	0.105	0.950	-	-	-	-	0.110	0.210	0.145	139	True
ResNet6_86	0.059	0.409	0.103	0.910	0.283	0.381	0.325	0.874	0.097	0.478	0.161	176	True
UnetStride1_49	0.090	0.113	0.100	0.974	0.378	0.102	0.161	0.915	0.159	0.079	0.105	363	-
ResNet6_83	0.083	0.127	0.100	0.973	-	-	-	-	0.131	0.130	0.130	229	True
UnetStride1_86	0.055	0.488	0.100	0.889	0.272	0.456	0.341	0.859	0.081	0.519	0.140	121	True
ResNet6_51	0.054	0.429	0.096	0.898	0.267	0.401	0.320	0.864	0.075	0.565	0.132	190	True
ResNet6_15	0.052	0.515	0.094	0.876	0.260	0.485	0.338	0.849	0.061	0.638	0.112	389	True
UnetStride1_14	0.052	0.473	0.094	0.886	0.260	0.444	0.328	0.855	0.075	0.509	0.131	373	True
UnetStride1_87	0.051	0.497	0.093	0.878	0.257	0.469	0.332	0.850	0.061	0.669	0.111	141	True
UnetStride1_51	0.051	0.488	0.093	0.880	0.257	0.460	0.330	0.851	0.073	0.551	0.130	150	True
UnetStride1_85	0.062	0.175	0.092	0.957	0.291	0.158	0.205	0.902	0.129	0.274	0.175	352	True
UnetStride1_4	0.056	0.247	0.091	0.938	0.270	0.227	0.247	0.889	0.088	0.195	0.121	380	-
UnetStride1_67	0.062	0.166	0.091	0.958	0.290	0.149	0.197	0.903	0.201	0.453	0.278	397	True

Continued on next page

Specification	Test ("real" labels)				Test ("increased" labels)				Validation			Best	w/o
	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Epoch	stop
ResNet6_67	0.060	0.169	0.089	0.957	0.287	0.155	0.201	0.902	0.204	0.253	0.226	384	True
FCN_29	0.055	0.231	0.089	0.944	-	-	-	-	0.039	0.283	0.069	102	-
UnetStride1_31	0.055	0.226	0.088	0.941	0.268	0.210	0.235	0.891	0.197	0.467	0.277	398	True
ResNet6_103	0.052	0.256	0.087	0.932	0.259	0.239	0.248	0.885	0.124	0.484	0.197	378	True
UnetStride1_105	0.053	0.227	0.086	0.939	0.259	0.208	0.231	0.889	0.181	0.543	0.271	326	-
FCN_95	0.048	0.361	0.085	0.902	0.246	0.342	0.286	0.864	0.109	0.392	0.171	24	-
FCN_32	0.047	0.391	0.085	0.894	0.245	0.373	0.296	0.858	0.098	0.515	0.164	387	True
FCN_50	0.048	0.277	0.082	0.922	0.250	0.268	0.259	0.877	0.111	0.441	0.177	288	True
FCN_58	0.047	0.306	0.081	0.913	0.241	0.290	0.263	0.871	0.123	0.475	0.195	226	True
ResNet6_40	0.059	0.130	0.081	0.963	0.279	0.118	0.166	0.905	0.106	0.141	0.121	367	-
ResNet6_105	0.046	0.327	0.081	0.906	0.236	0.308	0.267	0.865	0.097	0.603	0.167	370	-
FCN_23	0.046	0.275	0.079	0.920	0.239	0.262	0.250	0.875	0.105	0.397	0.166	389	-
FCN_59	0.044	0.348	0.079	0.898	0.233	0.334	0.274	0.859	0.112	0.517	0.184	141	True
ResNet6_31	0.061	0.110	0.078	0.968	0.274	0.094	0.140	0.908	0.049	0.062	0.055	12	-
ResNet6_12	0.044	0.352	0.078	0.903	-	-	-	-	0.046	0.257	0.078	29	-
FCN_86	0.045	0.294	0.078	0.913	0.237	0.283	0.258	0.870	0.094	0.353	0.148	21	-
FCN_87	0.046	0.259	0.078	0.923	0.234	0.240	0.237	0.877	0.093	0.462	0.155	214	True
FCN_24	0.044	0.347	0.078	0.897	0.230	0.331	0.271	0.859	0.091	0.515	0.155	299	True
ResNet6_33	0.043	0.434	0.078	0.871	0.228	0.420	0.295	0.840	0.073	0.716	0.132	384	-
UnetStride1_104	0.043	0.430	0.078	0.872	0.225	0.410	0.291	0.841	0.103	0.548	0.173	140	-
ResNet6_87	0.042	0.521	0.078	0.844	0.221	0.495	0.305	0.820	0.066	0.643	0.119	184	True
FCN_68	0.045	0.296	0.078	0.912	0.236	0.286	0.258	0.869	0.128	0.503	0.205	269	True

Continued on next page

Specification	Test ("real" labels)				Test ("increased" labels)				Validation			Best	w/o
	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Epoch	stop
UnetStride1_68	0.046	0.236	0.077	0.929	0.231	0.216	0.223	0.880	0.192	0.528	0.282	399	True
UnetStride1_77	0.043	0.361	0.077	0.891	0.222	0.338	0.268	0.853	0.095	0.423	0.155	154	True
ResNet6_76	0.044	0.272	0.076	0.917	0.229	0.257	0.242	0.872	0.082	0.214	0.119	393	True
UnetStride1_32	0.043	0.310	0.075	0.904	0.222	0.290	0.252	0.862	0.178	0.517	0.265	382	True
UnetStride1_50	0.040	0.477	0.073	0.849	0.212	0.456	0.290	0.822	0.072	0.521	0.126	182	True
ResNet6_78	0.040	0.387	0.073	0.877	0.213	0.366	0.269	0.842	0.068	0.537	0.121	201	True
ResNet6_96	0.040	0.413	0.073	0.868	0.213	0.396	0.277	0.836	0.088	0.615	0.153	278	-
ResNet6_41	0.040	0.355	0.072	0.885	0.212	0.335	0.260	0.848	0.074	0.489	0.129	294	True
UnetStride1_76	0.045	0.180	0.072	0.942	0.227	0.166	0.192	0.889	0.089	0.203	0.123	345	True
UnetStride1_41	0.039	0.417	0.072	0.864	0.208	0.393	0.272	0.832	0.067	0.501	0.118	300	True
ResNet6_69	0.040	0.364	0.072	0.881	0.211	0.346	0.262	0.845	0.107	0.606	0.181	378	-
UnetStride1_30	0.044	0.189	0.071	0.942	-	-	-	-	0.148	0.266	0.190	385	True
UnetStride1_75	0.049	0.125	0.071	0.961	-	-	-	-	0.094	0.180	0.124	217	True
ResNet6_59	0.038	0.429	0.070	0.856	0.205	0.410	0.273	0.826	0.105	0.585	0.179	259	True
UnetStride1_58	0.043	0.188	0.070	0.937	0.222	0.176	0.196	0.885	0.197	0.463	0.277	400	-
ResNet6_77	0.039	0.353	0.069	0.881	0.206	0.336	0.255	0.844	0.076	0.497	0.132	299	True
UnetStride1_103	0.042	0.194	0.069	0.935	0.218	0.180	0.197	0.883	0.168	0.521	0.255	400	True
ResNet6_75	0.046	0.140	0.069	0.956	-	-	-	-	0.093	0.174	0.121	207	True
UnetStride1_23	0.038	0.417	0.069	0.859	0.206	0.407	0.274	0.828	0.090	0.586	0.156	398	True
UnetStride1_94	0.041	0.228	0.069	0.923	0.211	0.211	0.211	0.874	0.180	0.424	0.253	391	True
ResNet6_95	0.037	0.424	0.069	0.856	0.202	0.406	0.270	0.825	0.104	0.564	0.176	112	True
UnetStride1_22	0.040	0.259	0.069	0.912	0.211	0.246	0.227	0.866	0.134	0.378	0.198	385	True

Continued on next page

Specification	Test ("real" labels)				Test ("increased" labels)				Validation			Best	w/o
	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Epoch	stop
ResNet6_6	0.037	0.404	0.068	0.862	0.200	0.382	0.263	0.829	0.051	0.620	0.094	365	True
FCN_105	0.037	0.489	0.068	0.832	0.201	0.474	0.282	0.808	0.076	0.527	0.132	19	-
UnetStride1_95	0.036	0.461	0.067	0.839	0.199	0.446	0.275	0.812	0.111	0.523	0.183	100	-
UnetStride1_59	0.038	0.270	0.067	0.906	0.205	0.256	0.227	0.862	0.159	0.501	0.242	396	True
FCN_96	0.038	0.283	0.067	0.901	0.204	0.270	0.232	0.858	0.100	0.433	0.163	361	-
ResNet6_58	0.040	0.202	0.067	0.929	0.210	0.189	0.199	0.879	0.136	0.341	0.194	372	True
UnetStride1_48	0.067	0.066	0.066	0.978	-	-	-	-	0.113	0.057	0.076	83	-
UnetStride1_6	0.035	0.511	0.066	0.819	0.193	0.489	0.277	0.796	0.055	0.619	0.101	208	True
UnetStride1_24	0.037	0.264	0.065	0.905	0.200	0.250	0.222	0.861	0.124	0.540	0.202	392	True
ResNet6_68	0.037	0.267	0.065	0.904	0.201	0.254	0.224	0.860	0.133	0.560	0.215	381	True
ResNet6_23	0.035	0.426	0.065	0.846	0.194	0.413	0.264	0.817	0.100	0.538	0.168	351	True
FCN_60	0.038	0.226	0.065	0.918	0.206	0.218	0.212	0.871	0.103	0.457	0.168	389	True
ResNet6_42	0.035	0.436	0.064	0.840	0.189	0.417	0.260	0.811	0.058	0.642	0.106	149	True
FCN_33	0.035	0.357	0.064	0.869	0.192	0.343	0.246	0.833	0.095	0.506	0.159	264	True
UnetStride1_42	0.034	0.474	0.063	0.823	0.185	0.452	0.263	0.798	0.059	0.623	0.109	286	True
ResNet6_5	0.034	0.414	0.063	0.845	0.189	0.403	0.257	0.815	0.032	0.169	0.054	28	-
UnetStride2_14	0.037	0.187	0.062	0.929	0.198	0.176	0.186	0.878	0.140	0.390	0.206	302	True
UnetStride1_15	0.032	0.545	0.061	0.790	0.180	0.523	0.268	0.772	0.063	0.593	0.114	258	True
FCN_40	0.039	0.137	0.061	0.947	0.210	0.131	0.161	0.892	0.119	0.285	0.168	245	True
UnetStride1_60	0.034	0.287	0.061	0.889	0.187	0.274	0.222	0.847	0.174	0.537	0.263	393	True
UnetStride1_33	0.034	0.278	0.061	0.892	0.189	0.267	0.221	0.850	0.121	0.563	0.200	400	True
FCN_47	0.035	0.205	0.060	0.925	-	-	-	-	0.097	0.086	0.091	46	-

Continued on next page

Specification	Test ("real" labels)				Test ("increased" labels)				Validation			Best	w/o
	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Epoch	stop
UnetStride1_96	0.032	0.479	0.060	0.811	0.180	0.467	0.259	0.788	0.073	0.631	0.131	47	-
FCN_69	0.031	0.570	0.060	0.774	0.177	0.554	0.268	0.759	0.068	0.605	0.122	73	-
ResNet6_24	0.031	0.496	0.059	0.802	0.178	0.485	0.260	0.780	0.073	0.654	0.131	316	True
UnetStride2_68	0.038	0.129	0.059	0.948	0.201	0.121	0.151	0.892	0.258	0.443	0.326	382	True
UnetStride2_50	0.035	0.179	0.058	0.928	0.189	0.170	0.179	0.876	0.119	0.418	0.185	381	True
FCN_31	0.035	0.168	0.058	0.932	0.190	0.158	0.173	0.879	0.129	0.363	0.190	339	-
FCN_41	0.033	0.240	0.058	0.902	0.185	0.235	0.207	0.856	0.077	0.406	0.130	39	-
ResNet6_104	0.033	0.222	0.058	0.909	0.181	0.211	0.195	0.861	0.100	0.578	0.170	157	-
UnetStride1_5	0.031	0.416	0.058	0.829	0.176	0.407	0.246	0.801	0.032	0.209	0.055	57	-
UnetStride2_49	0.037	0.125	0.057	0.948	0.198	0.118	0.148	0.891	0.199	0.366	0.258	358	True
UnetStride2_33	0.038	0.115	0.057	0.952	0.202	0.109	0.141	0.895	0.188	0.452	0.265	386	True
ResNet6_60	0.030	0.453	0.056	0.810	0.170	0.441	0.246	0.784	0.094	0.616	0.162	395	True
FCN_66	0.050	0.064	0.056	0.975	-	-	-	-	0.159	0.267	0.200	299	True
FCN_13	0.031	0.262	0.056	0.888	0.174	0.253	0.206	0.845	0.100	0.374	0.158	246	-
ResNet6_94	0.031	0.257	0.055	0.890	0.173	0.247	0.204	0.846	0.123	0.436	0.192	361	True
FCN_51	0.030	0.393	0.055	0.832	0.168	0.380	0.233	0.801	0.076	0.506	0.132	306	True
UnetStride1_83	0.050	0.060	0.055	0.976	-	-	-	-	0.154	0.133	0.142	346	-
FCN_15	0.029	0.491	0.055	0.787	0.165	0.479	0.246	0.766	0.068	0.532	0.121	178	True
UnetStride2_67	0.041	0.081	0.054	0.965	0.216	0.076	0.112	0.904	0.332	0.400	0.363	386	True
UnetStride2_24	0.036	0.112	0.054	0.951	0.193	0.106	0.137	0.893	0.182	0.430	0.256	369	True
UnetStride2_86	0.033	0.158	0.054	0.931	0.181	0.151	0.164	0.878	0.142	0.375	0.206	337	True
UnetStride1_78	0.029	0.506	0.054	0.778	0.162	0.488	0.244	0.759	0.058	0.643	0.107	178	True

Continued on next page

Specification	Test ("real" labels)				Test ("increased" labels)				Validation			Best	w/o
	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Epoch	stop
UnetStride2_51	0.034	0.121	0.053	0.946	0.186	0.114	0.142	0.890	0.156	0.372	0.220	384	True
UnetStride2_85	0.031	0.179	0.053	0.920	0.173	0.171	0.172	0.869	0.127	0.372	0.190	323	True
UnetStride2_87	0.032	0.142	0.053	0.936	0.181	0.138	0.157	0.882	0.140	0.365	0.203	299	True
ResNet6_74	0.047	0.060	0.053	0.975	-	-	-	-	0.099	0.090	0.094	367	-
FCN_49	0.031	0.180	0.052	0.918	0.171	0.173	0.172	0.867	0.158	0.319	0.211	210	-
UnetStride2_15	0.032	0.140	0.052	0.936	0.176	0.133	0.152	0.881	0.153	0.371	0.216	335	True
FCN_104	0.031	0.156	0.052	0.928	0.172	0.148	0.159	0.875	0.136	0.435	0.207	340	True
UnetStride2_13	0.032	0.124	0.051	0.942	0.177	0.119	0.142	0.886	0.158	0.364	0.220	359	-
UnetStride1_74	0.039	0.070	0.050	0.969	-	-	-	-	0.084	0.098	0.090	286	-
ResNet6_22	0.029	0.189	0.050	0.910	0.165	0.185	0.174	0.860	0.150	0.221	0.178	398	-
FCN_42	0.027	0.283	0.049	0.863	0.159	0.283	0.203	0.823	0.076	0.496	0.131	312	True
FCN_67	0.094	0.033	0.049	0.984	0.396	0.031	0.057	0.919	0.099	0.058	0.073	14	-
UnetStride2_103	0.035	0.079	0.048	0.961	0.190	0.076	0.108	0.901	0.300	0.405	0.345	390	True
UnetStride1_69	0.025	0.441	0.048	0.780	0.147	0.429	0.219	0.756	0.075	0.650	0.135	108	-
UnetStride2_69	0.031	0.093	0.047	0.952	0.172	0.088	0.117	0.893	0.232	0.421	0.299	388	-
ResNet6_102	0.034	0.073	0.047	0.965	-	-	-	-	0.177	0.276	0.215	332	-
FCN_14	0.024	0.508	0.047	0.739	0.142	0.494	0.221	0.722	0.072	0.469	0.124	123	True
UnetStride2_59	0.034	0.070	0.046	0.963	0.187	0.067	0.099	0.902	0.226	0.419	0.294	379	True
FCN_48	0.046	0.046	0.046	0.978	-	-	-	-	0.153	0.200	0.174	263	True
FCN_57	0.036	0.059	0.045	0.970	-	-	-	-	0.162	0.249	0.197	373	True
UnetStride2_32	0.030	0.089	0.044	0.952	0.167	0.086	0.114	0.893	0.245	0.421	0.310	395	True
ResNet6_101	0.039	0.052	0.044	0.974	-	-	-	-	0.211	0.212	0.211	380	True

Continued on next page

Specification	Test ("real" labels)				Test ("increased" labels)				Validation			Best	w/o
	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Epoch	stop
UnetStride2_31	0.039	0.051	0.044	0.973	0.205	0.047	0.076	0.910	0.362	0.398	0.379	369	True
UnetStride2_94	0.037	0.054	0.044	0.971	0.195	0.050	0.079	0.908	0.340	0.383	0.360	378	True
UnetStride2_104	0.032	0.068	0.044	0.963	0.180	0.066	0.097	0.902	0.277	0.399	0.327	381	-
UnetStride2_23	0.032	0.061	0.042	0.965	0.182	0.060	0.090	0.904	0.299	0.393	0.340	399	True
FCN_94	0.045	0.037	0.040	0.978	0.233	0.035	0.061	0.914	0.187	0.302	0.231	149	True
ResNet6_14	0.021	0.697	0.040	0.583	0.124	0.684	0.210	0.590	0.030	0.789	0.057	40	-
FCN_77	0.026	0.088	0.040	0.946	0.153	0.089	0.113	0.888	0.099	0.368	0.156	264	True
ResNet6_30	0.118	0.024	0.039	0.987	-	-	-	-	0.095	0.007	0.013	55	-
FCN_6	0.020	0.362	0.039	0.774	0.124	0.360	0.184	0.746	0.063	0.502	0.112	216	True
UnetStride2_5	0.024	0.097	0.038	0.939	0.140	0.094	0.113	0.882	0.146	0.285	0.193	383	True
FCN_5	0.020	0.321	0.038	0.798	0.125	0.322	0.180	0.766	0.062	0.452	0.110	199	-
ResNet6_92	0.026	0.072	0.038	0.958	-	-	-	-	0.167	0.201	0.182	375	True
UnetStride1_57	0.036	0.041	0.038	0.976	-	-	-	-	0.391	0.285	0.330	392	True
UnetStride2_77	0.019	0.397	0.037	0.740	0.118	0.395	0.182	0.717	0.067	0.467	0.117	41	True
UnetStride2_96	0.033	0.042	0.037	0.972	0.178	0.039	0.064	0.909	0.312	0.385	0.345	388	True
UnetStride1_39	0.029	0.049	0.036	0.970	-	-	-	-	0.107	0.166	0.130	373	True
UnetStride2_42	0.024	0.075	0.036	0.950	0.137	0.071	0.093	0.890	0.143	0.320	0.197	391	True
UnetStride1_40	0.028	0.051	0.036	0.965	0.160	0.050	0.077	0.903	0.135	0.074	0.096	383	-
ResNet6_91	0.024	0.071	0.036	0.955	-	-	-	-	0.161	0.148	0.154	388	True
UnetStride2_22	0.032	0.040	0.036	0.973	0.177	0.038	0.062	0.909	0.310	0.393	0.347	381	True
FCN_91	0.025	0.060	0.036	0.962	-	-	-	-	0.011	0.129	0.021	1	-
UnetStride2_60	0.036	0.035	0.035	0.976	0.187	0.032	0.055	0.912	0.297	0.399	0.341	354	True

Continued on next page

Specification	Test ("real" labels)				Test ("increased" labels)				Validation			Best	w/o
	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Epoch	stop
ResNet6_100	0.034	0.037	0.035	0.976	-	-	-	-	0.199	0.182	0.190	394	-
UnetStride2_58	0.028	0.047	0.035	0.968	0.168	0.048	0.075	0.905	0.321	0.400	0.356	400	True
UnetStride2_95	0.034	0.036	0.035	0.975	0.194	0.036	0.061	0.911	0.321	0.386	0.350	391	True
FCN_78	0.025	0.059	0.035	0.959	0.143	0.057	0.082	0.898	0.095	0.396	0.154	381	True
ResNet6_93	0.029	0.043	0.035	0.972	-	-	-	-	0.167	0.216	0.188	377	True
UnetStride1_56	0.037	0.033	0.035	0.979	-	-	-	-	0.313	0.281	0.296	400	-
UnetStride2_41	0.022	0.086	0.035	0.940	0.127	0.083	0.100	0.882	0.128	0.334	0.185	389	True
FCN_103	0.036	0.032	0.034	0.977	0.195	0.031	0.054	0.913	0.241	0.299	0.267	361	True
UnetStride2_6	0.022	0.069	0.034	0.950	0.130	0.066	0.088	0.890	0.126	0.318	0.181	396	True
UnetStride2_40	0.021	0.091	0.034	0.934	0.123	0.089	0.103	0.877	0.132	0.279	0.179	387	-
UnetStride1_102	0.028	0.041	0.034	0.972	-	-	-	-	0.524	0.321	0.399	398	-
UnetStride2_105	0.030	0.038	0.033	0.972	0.167	0.036	0.060	0.909	0.310	0.385	0.343	388	True
UnetStride1_21	0.020	0.096	0.033	0.934	-	-	-	-	0.176	0.254	0.208	395	True
FCN_4	0.018	0.195	0.032	0.853	0.106	0.190	0.136	0.808	0.012	0.182	0.022	1	-
UnetStride1_65	0.039	0.027	0.032	0.981	-	-	-	-	0.477	0.304	0.371	397	True
ResNet6_82	0.038	0.027	0.031	0.981	-	-	-	-	0.075	0.041	0.053	394	-
UnetStride1_20	0.033	0.028	0.030	0.979	-	-	-	-	0.303	0.277	0.289	398	-
UnetStride1_66	0.026	0.036	0.030	0.973	-	-	-	-	0.515	0.361	0.425	400	True
FCN_12	0.016	0.445	0.030	0.667	-	-	-	-	0.012	0.444	0.023	34	-
ResNet6_66	0.017	0.101	0.030	0.923	-	-	-	-	0.162	0.327	0.217	397	True
UnetStride2_4	0.019	0.060	0.029	0.950	0.117	0.059	0.079	0.889	0.128	0.300	0.179	363	-
UnetStride1_29	0.034	0.025	0.029	0.980	-	-	-	-	0.607	0.315	0.415	394	-

Continued on next page

Specification	Test ("real" labels)				Test ("increased" labels)				Validation			Best	w/o
	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Epoch	stop
UnetStride2_78	0.018	0.065	0.029	0.945	0.110	0.063	0.080	0.885	0.132	0.302	0.183	306	True
ResNet6_65	0.020	0.049	0.028	0.961	-	-	-	-	0.211	0.234	0.222	389	-
ResNet6_39	0.029	0.028	0.028	0.978	-	-	-	-	0.049	0.028	0.035	22	-
UnetStride1_91	0.042	0.021	0.028	0.983	-	-	-	-	0.476	0.239	0.318	400	-
ResNet6_56	0.022	0.035	0.027	0.970	-	-	-	-	0.208	0.196	0.202	395	-
UnetStride1_93	0.019	0.044	0.027	0.962	-	-	-	-	0.489	0.275	0.352	378	True
ResNet6_73	0.035	0.021	0.026	0.982	-	-	-	-	0.063	0.034	0.044	396	True
FCN_84	0.029	0.021	0.025	0.980	-	-	-	-	0.148	0.168	0.158	324	True
UnetStride2_76	0.016	0.048	0.024	0.951	0.098	0.047	0.064	0.889	0.153	0.290	0.200	396	True
UnetStride1_92	0.025	0.022	0.023	0.979	-	-	-	-	0.509	0.296	0.374	400	True
UnetStride1_47	0.056	0.014	0.023	0.986	-	-	-	-	0.200	0.027	0.048	148	-
UnetStride1_101	0.041	0.014	0.021	0.985	-	-	-	-	0.627	0.305	0.411	399	True
UnetStride1_100	0.022	0.021	0.021	0.978	-	-	-	-	0.618	0.316	0.418	400	-
FCN_39	0.031	0.016	0.021	0.983	-	-	-	-	0.172	0.181	0.177	245	True
ResNet6_57	0.011	0.104	0.020	0.881	-	-	-	-	0.199	0.149	0.170	128	-
UnetStride2_12	0.029	0.015	0.020	0.983	-	-	-	-	0.435	0.235	0.305	354	True
UnetStride1_64	0.015	0.023	0.018	0.971	-	-	-	-	0.290	0.071	0.113	398	-
UnetStride2_21	0.015	0.022	0.018	0.972	-	-	-	-	0.546	0.359	0.433	399	True
UnetStride2_83	0.021	0.015	0.017	0.980	-	-	-	-	0.240	0.116	0.156	76	-
FCN_65	0.031	0.012	0.017	0.984	-	-	-	-	0.187	0.194	0.190	195	-
UnetStride2_30	0.024	0.012	0.016	0.983	-	-	-	-	0.620	0.364	0.459	368	True
FCN_85	0.031	0.011	0.016	0.983	0.184	0.011	0.021	0.917	0.299	0.246	0.270	376	True

Continued on next page

Specification	Test ("real" labels)				Test ("increased" labels)				Validation			Best	w/o
	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Epoch	stop
FCN_56	0.034	0.011	0.016	0.985	-	-	-	-	0.194	0.180	0.187	341	-
UnetStride2_84	0.028	0.011	0.016	0.984	-	-	-	-	0.583	0.260	0.360	370	True
UnetStride2_48	0.027	0.011	0.015	0.984	-	-	-	-	0.493	0.245	0.328	390	True
ResNet6_48	0.035	0.009	0.015	0.985	-	-	-	-	0.068	0.013	0.021	37	-
UnetStride2_39	0.021	0.010	0.014	0.983	-	-	-	-	0.410	0.222	0.288	375	True
UnetStride2_10	0.022	0.010	0.013	0.983	-	-	-	-	0.395	0.213	0.277	386	-
UnetStride2_20	0.013	0.013	0.013	0.977	-	-	-	-	0.739	0.345	0.471	358	-
UnetStride2_66	0.027	0.008	0.013	0.985	-	-	-	-	0.693	0.361	0.475	358	True
UnetStride2_57	0.027	0.008	0.013	0.985	-	-	-	-	0.723	0.345	0.467	367	True
UnetStride2_65	0.020	0.009	0.013	0.983	-	-	-	-	0.725	0.361	0.482	376	True
UnetStride2_102	0.031	0.008	0.012	0.986	-	-	-	-	0.747	0.354	0.481	376	True
UnetStride2_56	0.018	0.009	0.012	0.983	-	-	-	-	0.753	0.344	0.472	388	True
FCN_93	0.014	0.010	0.012	0.980	-	-	-	-	0.140	0.140	0.140	29	-
UnetStride2_28	0.036	0.007	0.012	0.986	-	-	-	-	0.777	0.360	0.492	377	-
FCN_38	0.020	0.008	0.011	0.984	-	-	-	-	0.199	0.101	0.134	376	-
UnetStride2_47	0.026	0.007	0.011	0.985	-	-	-	-	0.530	0.221	0.312	398	True
FCN_21	0.029	0.006	0.011	0.986	-	-	-	-	0.165	0.214	0.186	398	True
UnetStride2_73	0.015	0.008	0.010	0.982	-	-	-	-	0.439	0.220	0.293	306	True
UnetStride2_74	0.016	0.008	0.010	0.983	-	-	-	-	0.483	0.195	0.278	358	True
UnetStride2_100	0.030	0.006	0.010	0.986	-	-	-	-	0.821	0.349	0.489	363	True
UnetStride2_93	0.035	0.006	0.010	0.987	-	-	-	-	0.766	0.346	0.476	383	True
FCN_20	0.010	0.009	0.010	0.978	-	-	-	-	0.188	0.117	0.144	315	-

Continued on next page

Specification	Test ("real" labels)				Test ("increased" labels)				Validation			Best	w/o
	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Epoch	stop
UnetStride2_75	0.016	0.007	0.010	0.983	-	-	-	-	0.500	0.197	0.283	377	True
UnetStride2_91	0.030	0.005	0.009	0.986	-	-	-	-	0.780	0.323	0.457	384	True
UnetStride2_82	0.017	0.006	0.008	0.985	-	-	-	-	0.620	0.228	0.334	380	-
FCN_30	0.006	0.012	0.008	0.967	-	-	-	-	0.022	0.015	0.018	11	-
FCN_102	0.026	0.005	0.008	0.986	-	-	-	-	0.168	0.249	0.201	315	-
UnetStride2_46	0.026	0.005	0.008	0.986	-	-	-	-	0.683	0.239	0.354	359	-
FCN_101	0.035	0.004	0.008	0.987	-	-	-	-	0.199	0.164	0.180	150	True
UnetStride2_101	0.031	0.004	0.007	0.987	-	-	-	-	0.812	0.353	0.492	393	True
UnetStride1_82	0.006	0.011	0.007	0.966	-	-	-	-	0.002	0.008	0.003	1	-
UnetStride2_64	0.024	0.004	0.007	0.986	-	-	-	-	0.855	0.355	0.502	391	-
UnetStride2_29	0.026	0.004	0.007	0.987	-	-	-	-	0.706	0.360	0.477	397	True
UnetStride2_55	0.024	0.004	0.007	0.987	-	-	-	-	0.828	0.347	0.489	399	-
UnetStride2_92	0.026	0.004	0.006	0.987	-	-	-	-	0.783	0.352	0.486	387	True
ResNet6_21	0.003	0.048	0.006	0.819	-	-	-	-	0.015	0.089	0.025	3	-
ResNet6_20	0.004	0.010	0.006	0.958	-	-	-	-	0.019	0.005	0.008	19	-
UnetStride1_55	0.065	0.003	0.006	0.988	-	-	-	-	0.003	0.001	0.001	6	-
ResNet6_19	0.003	0.022	0.005	0.906	-	-	-	-	0.002	0.007	0.003	48	-
FCN_2	0.004	0.005	0.005	0.975	-	-	-	-	0.003	0.010	0.005	62	-
UnetStride1_38	0.022	0.002	0.004	0.987	-	-	-	-	0.129	0.035	0.055	309	-
FCN_76	0.017	0.002	0.004	0.986	0.109	0.002	0.004	0.919	0.242	0.179	0.206	301	True
FCN_74	0.005	0.002	0.003	0.984	-	-	-	-	0.193	0.104	0.135	259	True
FCN_3	0.006	0.001	0.002	0.985	-	-	-	-	0.004	0.001	0.001	55	-

Continued on next page

Specification	Test ("real" labels)				Test ("increased" labels)				Validation			Best	w/o
	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Epoch	stop
FCN_75	0.022	0.001	0.002	0.988	-	-	-	-	0.174	0.187	0.180	362	True
UnetStride1_3	0.011	0.001	0.002	0.987	-	-	-	-	0.022	0.015	0.018	30	-
ResNet6_11	0.001	0.002	0.001	0.974	-	-	-	-	0.002	0.003	0.002	5	-
ResNet6_13	0.459	0.001	0.001	0.987	0.649	0.000	0.000	0.920	0.001	0.000	0.000	36	-
FCN_92	0.044	0.000	0.001	0.988	-	-	-	-	0.211	0.137	0.166	296	True
ResNet6_38	0.015	0.000	0.001	0.988	-	-	-	-	0.045	0.001	0.003	28	-
ResNet6_2	0.004	0.000	0.001	0.987	-	-	-	-	0.003	0.001	0.001	7	-
FCN_1	0.003	0.000	0.001	0.987	-	-	-	-	0.003	0.002	0.002	34	-
UnetStride1_73	0.006	0.000	0.001	0.988	-	-	-	-	0.006	0.000	0.001	1	-
FCN_83	0.019	0.000	0.000	0.988	-	-	-	-	0.227	0.096	0.135	365	True
FCN_82	0.098	0.000	0.000	0.988	-	-	-	-	0.100	0.014	0.025	164	-
UnetStride2_37	0.005	0.000	0.000	0.988	-	-	-	-	0.014	0.000	0.001	1	-
FCN_55	0.074	0.000	0.000	0.988	-	-	-	-	0.064	0.020	0.031	48	-
ResNet6_29	0.003	0.000	0.000	0.988	-	-	-	-	0.005	0.004	0.004	52	-
ResNet6_3	0.009	0.000	0.000	0.988	-	-	-	-	0.008	0.000	0.000	31	-
ResNet6_47	0.008	0.000	0.000	0.988	-	-	-	-	0.023	0.001	0.002	2	-
UnetStride2_11	0.007	0.000	0.000	0.988	-	-	-	-	0.105	0.003	0.005	124	-
FCN_11	0.000	0.000	0.000	0.987	-	-	-	-	0.003	0.005	0.004	8	-
FCN_19	0.000	0.000	0.000	0.988	-	-	-	-	0.000	0.000	0.000	1	-
ResNet6_4	0.000	0.000	0.000	0.987	0.000	0.000	0.000	0.920	0.005	0.000	0.000	3	-
UnetStride2_1	0.000	0.000	0.000	0.988	-	-	-	-	0.000	0.000	0.000	1	-
UnetStride1_1	0.000	0.000	0.000	0.988	-	-	-	-	0.000	0.000	0.000	1	-

Continued on next page

Specification	Test ("real" labels)				Test ("increased" labels)				Validation			Best	w/o
	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Epoch	stop
FCN_100	0.000	0.000	0.000	0.988	-	-	-	-	0.067	0.000	0.000	7	-
UnetStride2_38	0.000	0.000	0.000	0.988	-	-	-	-	0.232	0.005	0.010	93	-
ResNet6_28	0.000	0.000	0.000	0.988	-	-	-	-	0.000	0.000	0.000	1	-
ResNet6_37	0.000	0.000	0.000	0.988	-	-	-	-	0.000	0.000	0.000	1	-
UnetStride2_19	0.000	0.000	0.000	0.988	-	-	-	-	0.000	0.000	0.000	1	-
FCN_22	0.000	0.000	0.000	0.987	0.000	0.000	0.000	0.920	0.064	0.009	0.015	68	-
UnetStride1_19	0.000	0.000	0.000	0.988	-	-	-	-	0.000	0.000	0.000	1	-
UnetStride1_12	0.000	0.000	0.000	0.988	-	-	-	-	0.000	0.000	0.000	1	-
FCN_46	0.000	0.000	0.000	0.988	-	-	-	-	0.403	0.004	0.008	381	-
FCN_64	0.000	0.000	0.000	0.988	-	-	-	-	0.051	0.018	0.027	48	-
ResNet6_55	0.000	0.000	0.000	0.988	-	-	-	-	0.000	0.000	0.000	1	-
UnetStride1_28	0.000	0.000	0.000	0.988	-	-	-	-	0.000	0.000	0.000	1	-
UnetStride2_3	0.000	0.000	0.000	0.988	-	-	-	-	0.000	0.000	0.000	1	-
ResNet6_10	0.000	0.000	0.000	0.988	-	-	-	-	0.000	0.000	0.000	1	-
FCN_37	0.000	0.000	0.000	0.988	-	-	-	-	0.597	0.002	0.003	68	-
ResNet6_1	0.000	0.000	0.000	0.988	-	-	-	-	0.000	0.000	0.000	1	-
FCN_10	0.000	0.000	0.000	0.988	-	-	-	-	0.013	0.000	0.000	14	-
UnetStride1_37	0.000	0.000	0.000	0.988	-	-	-	-	0.000	0.000	0.000	1	-
UnetStride1_2	0.000	0.000	0.000	0.988	-	-	-	-	0.000	0.000	0.000	1	-
ResNet6_64	0.000	0.000	0.000	0.988	-	-	-	-	0.000	0.000	0.000	1	-
UnetStride1_46	0.000	0.000	0.000	0.988	-	-	-	-	0.000	0.000	0.000	1	-
UnetStride2_2	0.000	0.000	0.000	0.988	-	-	-	-	0.000	0.000	0.000	1	-

Continued on next page

	Test ("real" labels)				Test ("increased" labels)				Validation			Best	w/o
Specification	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Epoch	stop
UnetStride1_10	0.000	0.000	0.000	0.988	-	-	-	-	0.000	0.000	0.000	1	-
FCN_28	0.000	0.000	0.000	0.988	-	-	-	-	0.000	0.000	0.000	1	-
ResNet6_46	0.000	0.000	0.000	0.988	-	-	-	-	0.000	0.000	0.000	1	-
UnetStride1_11	0.000	0.000	0.000	0.988	-	-	-	-	0.088	0.000	0.001	147	-
FCN_73	0.000	0.000	0.000	0.988	-	-	-	-	0.510	0.002	0.004	363	-

Declaration

1. I hereby declare that this thesis entitled "Detecting Armed Conflict Damages in Satellite Imagery Using Deep Learning" is a result of my own work and that no other than the indicated aids have been used for its completion. Material borrowed directly or indirectly from the works of others is indicated in each individual case by acknowledgement of the source and also the secondary literature used. This work has not previously been submitted to any other examining authority and has not yet been published.

2. After completion of the examining process, this work will be given to the library of the University of Konstanz, where it will be accessible to the public for viewing and borrowing. As author of this work, I agree ~~/ do not agree~~ to this procedure.

Konstanz, 26.09.2022 

(Date) (Signature)