ETH zürich

Chair of Circular Engineering for Architecture

Master Thesis

Urban Scale Building Rooftop Material Detection Using Computer Vision

Parham Zendehdel Nobari
30. January 2023

Catherine De Wolf
Deepika Raghu



Institut für Bau- und Infrastruktur-management

D BAUG

cea Circular Engineering for Architecture

Acknowledgements

During my master thesis, I received a lot of support from my friends and family, colleagues and supervisors. First and foremost, I would like to express my gratitude to Prof. Dr. ir. arch. Catherine De Wolf for allowing me to do research under the helm of the Circular Engineering for Architecture (CEA) lab. Next, I would also like to extend my deepest gratitude to Deepika Raghu. She followed my thesis closely and has been very supportive throughout the whole semester. I am also deeply grateful to Dr. Iro Armeni and my colleagues in the computer science field for guiding me through my research. They showed great interest in my work and pointed out things I could improve.

Lastly, I would like to dedicate this thesis to the brave people of my home country, Iran, who at the time of conducting this thesis were engaged in a noble struggle for freedom, equality and justice both within the borders of our nation and beyond. Their unwavering determination and resilience in the face of adversity served as a constant source of inspiration, guiding me to persist in my efforts and to maintain hope for a better future, even in the darkest of hours.

Table of contents

1	**Introduction**	**1**
2	**Theoretical Background**	**2**
2.1	Computer Vision	2
2.2	Semantic Image Parsing	2
2.3	Image Annotation	3
2.3.1	COCO Dataset	3
2.3.2	COCO-Style Annotation	4
2.4	Class Imbalance	4
2.5	Mask R-CNN	4
2.6	Model Evaluation Metrics	5
2.6.1	Precision and Recall	5
2.6.2	Intersection Over Union – IoU	5
2.6.3	COCO Metrics	6
2.7	Hyperparameters	6
2.7.1	Epoch	6
2.7.2	Batch Size	7
2.7.3	Iteration	7
3	**Methodology**	**8**
3.1	Description of Raw Data	8
3.1.1	Geographic Scale of Raw Data	9
3.1.2	Structure of Raw Data	9
3.2	Data Pre-processing	10
3.3	Description of Pre-processed Data	11
3.3.1	Data Quality Issues	11
3.3.2	Façade Renderings	13
3.3.3	Rooftop Renderings	13
3.4	Dataset Creation and Preparation	14
3.4.1	Filtering and Selection of Processed Data	14
3.4.2	Data Value Compared to Other Available Data	14
3.4.3	Dataset Classes and Image Types	15
3.4.4	Dataset Training and Validation Split	17
3.4.5	Dataset Annotation	17
3.4.6	Dataset Augmentation and Class Imbalance	18
3.5	Computer Vision Model	19
3.5.1	Transfer Learning	19
3.5.2	Facebook Research Detectron2 Library	20
3.5.3	Model Training	21
4	**Results and Discussion**	**23**
4.1	Model Performance	23
4.2	Model Inference	24
4.2.1	Inference on Validation Dataset	24
4.2.2	Inference Limitations and Capabilities	25

4.3 Model Evaluation 28
4.4 Creating Material Take-off from Inferences 29

5 Conclusion and Outlook 30

6 Bibliography 32

7 Appendix 36
A1. Loss Calculation in Detectron2 Models 36
A2. Model Inference Output 36
A3. Google Colab Notebook 36

List of Figures

Figure 2.1: Different semantic image parsing methods 3
Figure 2.2: A visual representation of the concept of IoU. 5
Figure 3.1: An example of a captured photo-mesh from Uzufly 8
Figure 3.2: The commune of Collonge-Bellerive to the east of Lake Geneva in the Canton of Geneva. 9
Figure 3.3: The 3D object file of a sample building in the raw data and its corresponding texture file . .9
Figure 3.4: The process flow-chart of the automated data pre-processing 10
Figure 3.5: The Blender interface 11
Figure 3.6: The various categories of quality issues in the processed 12
Figure 3.7: Example of façade and rooftop renderings 14
Figure 3.8: Filtered & processed Uzufly image compared to Swisstopo's best aerial image 15
Figure 3.9: The four roof classes 16
Figure 3.10: The three image types 16
Figure 3.11: The training set annotations in Make Sense AI. 17
Figure 3.12: Number of annotated instances across all roof classes 18
Figure 3.13: Example augmentations of a metal sheet roof image 18
Figure 3.14: Number of annotated instances across all roof classes post-augmentation 19
Figure 3.15: Transfer learning implementation scheme. 20
Figure 3.16: Ground truth images and their annotations as interpreted by the Detectron2 model 21
Figure 4.1: Trained model loss curve diagram 23
Figure 4.2: Model inference on a sample gravel cover roof from the validation dataset 24
Figure 4.3: Model inference on a sample multi-instance image type from the validation dataset 25
Figure 4.4: Model inference on a sample validation image 25
Figure 4.5: Model inference on a sample validation image 26
Figure 4.6: Model inference on a sample validation image 26
Figure 4.7: Model inference on a sample validation image 27
Figure 4.8: Model inference on random images 27
Figure 4.9: The process of calculating an approximate material take-off from a segmented image. 29

List of Tables

Table 4.1: Model Segmentation Evaluation.....28
Table 4.2: Model Segmentation Evaluation Per-Class.....28

Master thesis for Civil Engineering

Urban Scale Building Rooftop Material Detection Using Computer Vision

Author(s) Parham Zendehdel Nobari

E-Mail parhamz@student.ethz.ch

Supervision

Prof. Catherine de Wolf, Professor of Circular engineering for architecture
Deepika Raghu, Doctorate at Circular engineering for architecture

January 30

Abstract

With the emergence of the climate crisis, the urgency to transition to a circular economy is now greater than ever. This urgency is best exemplified in the construction sector which despite consuming most of the world's materials and accounting for around 37% of the world's greenhouse gas emissions, still operates according to a "take-make-waste" model. The planning, design and construction of circular buildings, hinges on accurate data on the available material stock within a district/city, which as of yet is either inaccessible or simply non-existent. The focus of this thesis is to build an urban-scale dataset of high-resolution aerial rooftop images to train a proof-of-concept computer vision algorithm which can accurately detect materials, and create approximate material quantity take-offs of building rooftops in the built environment. With improved data, expanded versions of this image-based algorithm can be used to detect and quantify the material composition of the entire building exterior layer (rooftop and façade) for the purpose of creating a large-scale, comprehensive spatial material database for circular construction and architecture in the future.

Keywords
Circular economy; Data; Material stock; Computer vision; Urban-scale; High-resolution; Aerial images; Detection; Material take-off; Spatial material database

Zusammenfassung

Mit dem Aufkommen der Klimakrise ist die Dringlichkeit des Übergangs zu einer Kreislaufwirtschaft jetzt grösser denn je. Diese Dringlichkeit lässt sich am besten in dem Bausektor veranschaulichen, der trotz des Verbrauchs der meisten Materialien der Welt und eines Anteils von rund 37 % der weltweiten Treibhausgasemissionen immer noch nach einem „Take-Make-Waste“-Modell arbeitet. Die Planung, der Entwurf und der Bau von zirkulären Gebäuden hängen von genauen Daten über die verfügbaren Materialbestände innerhalb eines Bezirks/einer Stadt ab, die bisher entweder nicht zugänglich oder einfach nicht vorhanden sind. Der Schwerpunkt dieser Arbeit liegt auf der Erstellung eines Datensatzes aus hochauflösenden Luftbildern von Dächern im städtischen Massstab, um einen Proof-of-Concept-Computer Vision Algorithmus zu trainieren, der Materialen genau erkennen und ungefähre Materialmengenabschätzungen von Gebäudedächern erstellen kann. Mit verbesserten Daten können erweiterte Versionen dieses bildbasierten Algorithmus verwendet werden, um die Materialzusammensetzung der gesamten Gebäudeaussenschicht (Dach und Fassade) zu erfassen und zu quantifizieren, um eine grossräumige, umfassende räumliche Materialdatenbank für das zirkuläre Bauen und Architektur in der Zukunft zu erstellen.

Schlüsselwörter
Kreislaufwirtschaft; Daten; Materialbestände; Computer Vision; Städtischen Massstab; Hochauflösend; Luftbildern; Erkennung; Materialmengenabschätzungen; Räumliche Materialdatenbank

1 Introduction

The architecture, engineering and construction (AEC) industry currently operates based on a linear "take-make-waste" model. Raw resources are extracted and processed into construction materials, which are then transformed and constructed into buildings. Buildings are usually demolished at the end of their life-cycle, destroying the value of the embedded materials, thereby creating what is known as construction and demolition waste (CDW). This practice results in billions of tons of CDW each year which is often landfilled (Duan 2018), despite the fact that 95% of non-hazardous CDW can be reused or recycled (Ma 2020). Furthermore, mismanagement of CDW exacerbates climate change through water, ground, and air pollution, causing harm to ecosystems and human health. Meanwhile, increasing demand for housing driven by global population growth necessitates the use of newly extracted resources in constructing new buildings. This depletes raw resources and consumes large amounts of energy, making the AEC industry accountable for 37% of process and energy-related CO2 emissions and a major contributor to climate change (United Nations Environment Programme 2022). A shift towards a circular economy paradigm in the AEC industry is therefore critical for maintaining a sustainable built environment. A circular economy prioritizes the re-use and recycling of materials, reduces waste and minimizes the negative impact on the environment.

In line with the goals of a circular economy, recent regulations in the canton of Zurich have been introduced and put into effect. They aim to close material loops throughout the entire supply and value chain, from production to consumption and to waste, across all industry sectors including the AEC industry (Kanton Zürich 2022). However, in order to effectively transition towards a more circular paradigm, the AEC industry needs access to comprehensive data on the available material stock (Rose 2019). This data is necessary to inform design and construction processes, enabling the selection of materials that can be re-used or recycled, thereby reducing the need for new resources to be extracted. The availability of such data on a large scale will help the AEC industry to move towards a more circular economy, where resources are used in a closed loop system, reducing waste and preserving the environment for future generations.

The contribution of this thesis is to research the prospects of using urban-scale, ultra-high-detail 3D digital shadows of buildings and advanced computer vision techniques to mine data on the available material stock in the built environment. Previous research has focused on automatic building detection from remote sensing images (Luo 2021) and classifying roofing materials using satellite images (Trevisiol 2022) or detecting facade material and component compositions through terrestrial imagery (Raghu 2022). However, none have attempted to extract extensive information about the material composition of building exteriors from a single truly scalable source. The current work distinguishes itself by offering scalability in both input images and computer vision algorithms, as well as utilizing ultra-high-detail images for creating datasets which can not only be used for material detection but can also potentially be employed for conducting condition assessment of the detected materials and components.

Furthermore, it is important to note that this thesis is not about inventing new technology but rather about integrating state-of-the-art technology from the rapidly growing fields of computer vision and deep learning into the AEC industry through an innovative and interdisciplinary process. This thesis provides a roadmap for this process by outlining a pipeline for data collection, data pre-processing, data annotation, dataset creation, and data analysis and representation. The ultimate goal of this thesis is to develop the tools that are needed to create an urban-scale spatial material stock database which will assist future stakeholders in designing and managing the built environment with a circular perspective.

2 Theoretical Background

This chapter provides an overview of the key theories and concepts that will be discussed in the subsequent chapters. It presents the foundational knowledge needed to fully grasp the work, with a particular focus on the theoretical aspects of machine learning and computer vision techniques. The explanations are designed to be accessible to a broad audience, with a general and conceptual approach, rather than a highly technical one.

2.1 Computer Vision

Computer vision is a field of artificial intelligence that deals with the development of algorithms and models that allow computers to understand and interpret visual information from the world. This includes tasks such as image and video analysis, object recognition, and scene understanding. By analyzing visual data, computer vision systems can extract useful information, make decisions, and perform actions in a variety of applications.

Recent advances in deep learning have led to significant progress in computer vision, with neural networks achieving state-of-the-art results on a wide range of tasks (Zdziarski 2018). These include image classification and semantic image parsing. In addition to technological progress in neural networks and deep learning, a combination of the following factors has contributed to the recent rise of computer vision:

- Increase in data availability: with the proliferation of cameras and other imaging devices, more visual data is produced as ever before, which could in turn be used to train and test computer vision models (Mihajlovic 2019).
- Advancements in hardware: the availability of powerful graphics processing units (GPUs) and specialized chips that can handle large amounts of data and computational power has enabled the training of complex models for image and video processing tasks (Mihajlovic 2019).
- Emergence of large-scale data sets: the availability of large-scale datasets such as ImageNet and COCO has allowed for the development and training of more accurate models (Zdziarski 2018).
- Increased demand: the growth of industries such as self-driving cars, robotics, and surveillance has increased the demand for computer vision systems (Zdziarski 2018).

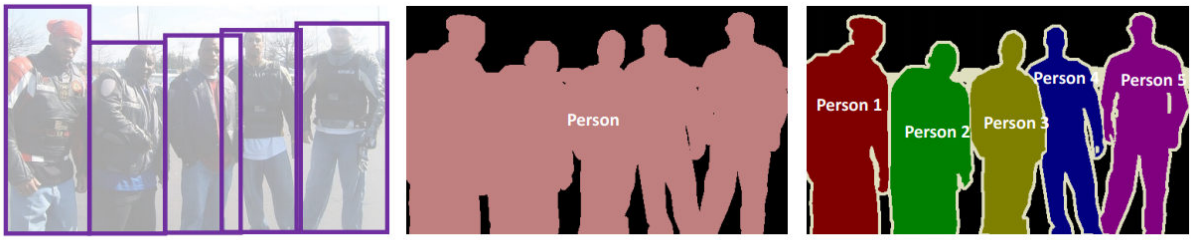
2.2 Semantic Image Parsing

Semantic image parsing is a process whereby an input image is decomposed into semantic regions, representing objects or pixels attributed to a class (Lili Huang 2018). Figure 2.1 depicts the different semantic image parsing methods which are further described in the following.

Object Detection
Object detection is a computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos. With the help of an object detector, it is possible to locate the object of interest in the image, classify it and draw a bounding box around it.

Semantic Segmentation
Semantic segmentation is a process of classifying each pixel in an image into a class. This is different from object detection, which only provides the bounding box coordinates of the object. With semantic

segmentation, we can obtain a more detailed understanding of the image by having a pixel-wise classification of the objects in the image.



Source: (Liu 2020)

Figure 2.1: Different semantic image parsing methods. From left to right: a) object detection, b) semantic segmentation, c) instance segmentation.

Instance Segmentation
Instance segmentation is a more advanced version of object detection, where not only the bounding box of the object is identified but also each instance of the object is labelled separately. This helps in separating out multiple objects of the same class that may be present in an image. This is particularly useful in applications where multiple objects of the same class need to be identified and distinguished from each other. In this thesis, instance segmentation algorithms are utilized so that different roof instances can be identified and distinguished across aerial images of different scales, featuring single or multiple buildings.

2.3 Image Annotation

Image annotation is the process of labeling or tagging images with relevant information, such as object or feature labels, bounding boxes, keypoints or mask segmentations (Pokhrel 2020). The goal of image annotation is to provide a detailed, structured description of the content of an image, which can be used for tasks such as object recognition, image search, and training machine learning models.

There are different types of image annotation, depending on the level of detail and the type of information provided. Some examples include:

- Object annotation: Labeling specific objects or features within an image, such as "car", "person", "stop sign", etc.
- Bounding box annotation: Drawing a box around an object or feature in an image and labeling it.
- Keypoint annotation: Marking specific points of interest, such as the corners of an object or the joints of a person, within an image.
- Semantic annotation: Assigning a semantic class or label to different regions of an image, such as "sky", "road", "building", etc.

Image annotation can be done manually by human annotators or using automated techniques, such as computer vision algorithms. The choice of method depends on the specific task and the available resources. Manual annotation is considered more accurate but it is time-consuming, while automated annotation is faster but may not be as accurate. In this thesis, images are annotated manually using semantic annotation to create a dataset of segmented rooftops.

2.3.1 COCO Dataset

The COCO (Common Objects in Context) dataset is a large-scale image recognition dataset that was developed by the Microsoft Research team (T. e. Lin 2014). It contains over 330,000 images, with more

than 2.5 million object instances annotated across 80 object categories. The dataset is designed to support the development of computer vision models for tasks such as object detection, instance segmentation, and captioning. The COCO dataset is notable for several reasons:

- Scale: It is one of the largest image recognition datasets available, with a diverse set of images that cover a wide range of objects, scenes, and contexts.
- Variety: The images in the dataset were collected from diverse sources such as the Internet, and they contain a wide range of objects and scenes, including people, animals, vehicles, indoor and outdoor environments, and more.
- Quality: The images were carefully selected to be of high quality and resolution, with minimal noise or distortion.
- Annotation: The dataset includes detailed annotation, including object bounding boxes, object segmentation masks, and captions that describe the objects and scenes in the images.

The COCO dataset is widely used in the computer vision community for training and evaluating object recognition models. It is also used as a benchmark for several computer vision challenges, such as the COCO object detection challenge and the COCO captioning challenge. The COCO dataset is publicly available and can be downloaded from the Microsoft COCO website, which also provides a browser-based interface for exploring the dataset.

2.3.2 COCO-Style Annotation

COCO-style annotation is a method of annotation used to label and mark object instances in images and videos. It is used to train object detection, semantic segmentation, and instance segmentation models. The COCO annotation format includes several types of annotations such as bounding boxes, keypoints, and polygons (masks). Each annotation is associated with an object class label and an instance ID. The bounding boxes are represented by four coordinates (x, y, width, height) that enclose the object, keypoints are the locations of specific parts of the object and polygons are used to mask certain parts of the object (Petrosyan 2021).

2.4 Class Imbalance

In machine learning, class imbalance refers to the situation wherein the classes in a dataset are not represented equally. This can occur when one class has significantly more examples than the other class. For example, in a dataset of animals, there may be many more examples of dogs than lions, leading to an imbalance in the class distribution.
When training a machine learning model on a dataset with class imbalance, the model is more likely to predict the majority class, as it has seen more examples of it during training. This can lead to poor performance on the minority class, as the model may not have seen enough examples of it to accurately classify it. Additionally, metrics like accuracy, which only consider the overall performance of the model, can be misleading in the presence of class imbalance as the model might achieve high accuracy by just predicting the majority class.
There are several techniques that can be used to address class imbalance in machine learning. One common approach is to over-sample the minority class by duplicating examples, or to under-sample the majority class by removing examples (Johnson 2019). Another approach is to adjust the class weights in the training loss function to give more importance to the minority class (Johnson 2019). It's important to note that the solution to this problem depends on the specific use case and context of the data, as well as the type of model used. Within the context of this thesis, since we are working with limited data, an over-sampling approach is taken to address class imbalance.

2.5 Mask R-CNN

Mask R-CNN is a deep learning-based model for object instance segmentation (G. G. Kaiming He 2017). It is an extension of the popular Faster R-CNN object detection model, which adds a branch for predicting a segmentation mask for each object detected in the image.

The Mask R-CNN model consist of three main components: a backbone network, a Region Proposal Network (RPN), and a parallel fully connected network. The backbone network is used for feature extraction and is typically a pre-trained convolutional neural network such as ResNet or FPN. The RPN is used to generate object proposals and the parallel fully connected network is used for object classification and mask prediction.
The model is trained end-to-end and uses the same detection mechanisms as Faster R-CNN, but adds a branch for predicting a mask for each object. During training, the mask branch is trained to predict a binary mask for each object, which is then compared with the ground truth mask to compute the loss. During inference, the model generates a set of object proposals, classifies each proposal and generates a binary mask for each object. The result is a set of bounding boxes, class labels, and binary segmentation masks for each object instance in the image. Mask R-CNN is commonly used in applications such as object detection, instance segmentation and medical imaging. Due to its high accuracy, it's often used as a base model for other architectures and is considered as one of the state-of-the-art models for instance segmentation.

2.6 Model Evaluation Metrics

This section covers the fundamental concepts behind the metrics used to evaluate a machine learning model's ability to generalize, or perform well on unseen data. The goal is to determine whether the model has effectively extracted and learned patterns in the training data and can apply them to new, real-world examples.

2.6.1 Precision and Recall

Precision and recall are two commonly used evaluation metrics in machine learning, particularly in the context of classification and object detection tasks (Goutte 2005).
Precision is a measure of how many of the predicted positive instances are actually positive. It is calculated by dividing the number of true positive instances (correctly predicted positive instances) by the total number of predicted positive instances. A high precision means that the model has a low false positive rate, meaning that it doesn't label instances as positive when they are actually negative.
Recall is a measure of how many of the actual positive instances are correctly predicted as positive. It is calculated by dividing the number of true positive instances by the total number of actual positive instances. A high recall means that the model has a low false negative rate, meaning that it detects most of the actual positive instances.

2.6.2 Intersection Over Union – IoU

Intersection over Union (IoU) is a metric used to evaluate the performance of object detection and segmentation models. IoU is defined as the ratio of the area of intersection between the predicted segmentation mask M_{gt} and the ground truth segmentation mask M_p to the area of their union (Rosebrock 2016). Mathematically, it is calculated as follows:

$$IoU = \frac{M_{gt} \cap M_p}{M_{gt} \cup M_p}$$

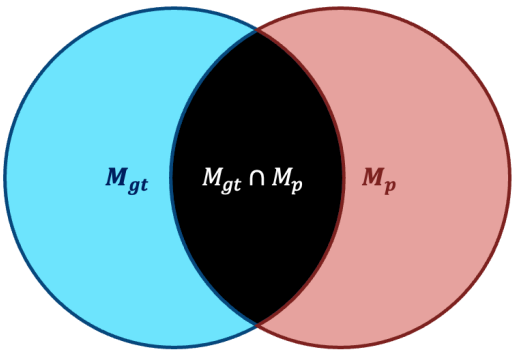


Figure 2.2: A visual representation of the concept of IoU.

As shown in Figure 2.2, the area of intersection is the area of the region where the predicted segmentation mask and the ground truth segmentation mask overlap. The area of union is the total area of both masks, including the overlap. IoU values range from 0 to 1, where a value of 0 indicates no overlap between the predicted segmentation mask and the ground truth segmentation mask, and a value of 1 indicates perfect overlap.

2.6.3 COCO Metrics

COCO (Common Objects in Context) metrics are a set of evaluation metrics used to evaluate object detection, instance segmentation and keypoint detection models (Microsoft 2021). They are defined by the COCO dataset and the COCO evaluation server. The most commonly used COCO metrics are Average Precision (AP), Average Precision at IoU threshold of 50% (AP50), and Average Precision at IoU threshold of 75% (AP75).

AP
Average Precision (AP) is a measure of the model's performance across all object classes and IoU thresholds from 0.5 to 0.95 with a step size of 0.05. It is calculated by averaging the precision-recall curves for each object class. Precision is the number of correctly detected objects divided by the total number of detected objects, and recall is the number of correctly detected objects divided by the total number of ground truth objects. The AP metric is commonly used in object detection and instance segmentation to evaluate the overall performance of a model as it is not biased towards any arbitrary IoU threshold.

AP50
Average Precision at IoU threshold of 50% (AP50) is similar to AP, but it only includes detections that have an IoU (Intersection over Union) score of at least 0.5 with the ground truth segmentation mask. This metric focuses on the detection of objects with a relatively high degree of overlap with the ground truth segmentation mask.

AP75
Average Precision at IoU threshold of 75% (AP75) is similar to AP50, but it only includes detections that have an IoU score of at least 0.75 with the ground truth segmentation mask. This metric focuses on the detection of objects with a very high degree of overlap with the ground truth segmentation mask.
In general, a model with a higher AP, AP50 and AP75 score is considered to be more accurate. However, the specific threshold value (50% or 75%) that is used depends on the application and the desired level of accuracy.

2.7 Hyperparameters

In machine learning, a model has two types of parameters: those that are learned from data, and hyperparameters, which are set by the practitioner. Hyperparameters are settings that are not learned from data and are used to control the behavior of the model during training. In this section we will cover some of the hyperparameters that are relevant for the training of the machine learning model proposed in this thesis.

2.7.1 Epoch

An epoch is a single pass of the neural network through the entire training dataset (Google 2022). During each epoch, the model's parameters are updated based on the error it makes on the training data. The number of epochs is a hyperparameter that controls how many times the model will cycle through the training data. Typically, the more epochs a model trains for, the better it will perform on the training data, but at the cost of potentially overfitting to the training data, which will be later discussed in section 4.1. The number of epochs is also used as a stopping criterion for the training process, when the model's performance on a validation set stops improving, or starts degrading.

2.7.2 Batch Size

In machine learning, a batch size refers to the number of samples used in one iteration of the training process (Google 2022). When training a model, the entire dataset is typically too large to fit in memory, so it must be split into smaller chunks called batches. During each iteration of the training process, a batch of data is passed through the model, and the model's parameters are updated based on the error it makes on that batch.
The batch size is a hyperparameter that can have a significant impact on the performance of the model. A larger batch size can lead to faster training times, but may also cause the model to converge to a suboptimal solution. A smaller batch size allows for more frequent updates to the model's parameters, which can lead to a more accurate solution, but at the cost of longer training times. The optimal batch size depends on the dataset, the model, and the available computational resources.

2.7.3 Iteration

An iteration refers to a single update of the model's parameters (or weights) during the training process (Google 2022). An iteration consists of the following steps:

- A batch of data is selected from the training dataset.
- The model's current parameters are used to make predictions on the selected batch of data.
- The error or loss between the predictions and the true labels is computed.
- The model's parameters are updated based on the loss, using an optimization algorithm (for instance stochastic gradient descent).

The number of iterations is determined by the number of batches in the dataset and the batch size.

3 Methodology

The previous chapter outlined the necessary concepts and definitions that are used throughout this thesis. This chapter will now focus on data acquisition, data preprocessing, dataset creation and the implementation of the aforementioned machine learning and computer vision concepts, and provides a detailed guide on the specific steps needed to achieve the objective of the thesis.

3.1 Description of Raw Data

The data for this thesis was obtained from Uzufly, an EPFL spin-off company in Lausanne that creates ultra-high-detailed[1] 3D digital shadows of buildings, cities, and territories using aerial photogrammetry (Uzufly 2020). Uzufly carries out flight plans over areas of interest and captures high-resolution photo-meshes of the area. An example of such a photo-mesh can be seen in Figure 3.1.

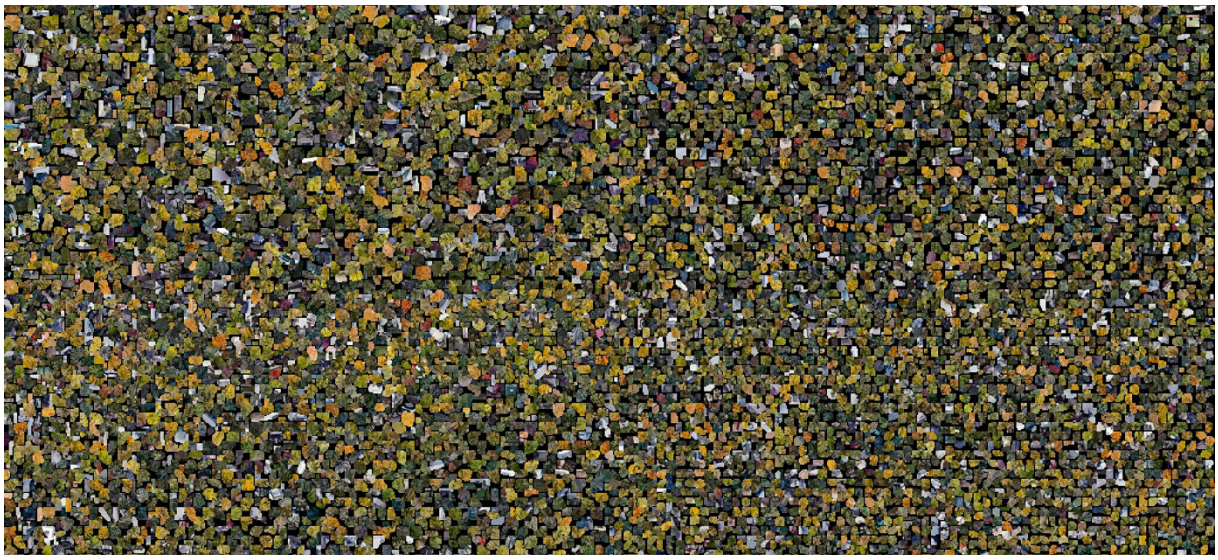


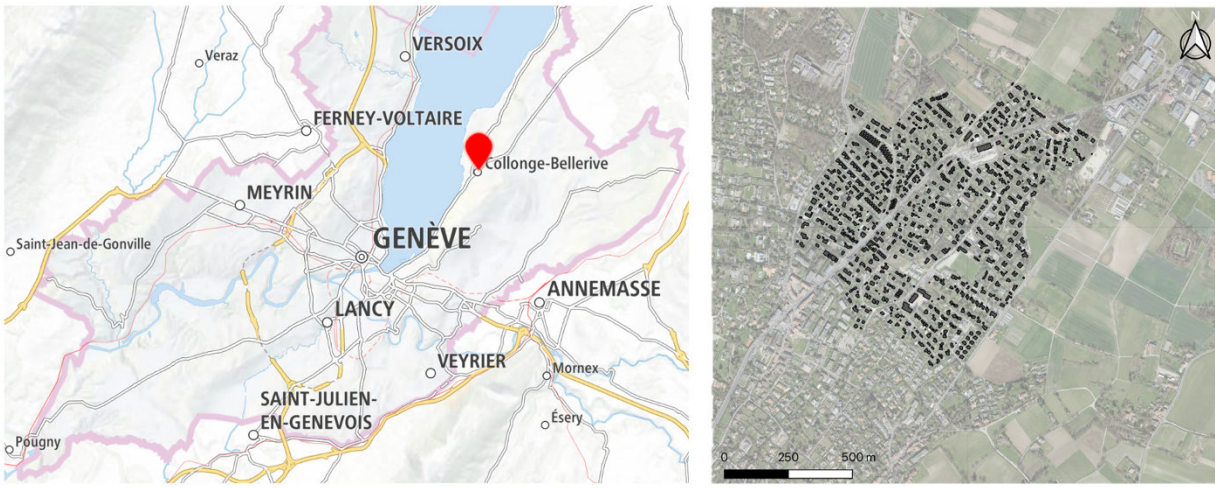
Figure 3.1: An example of a captured photo-mesh from Uzufly

Uzufly uses these photo-meshes and extracts textures for each individual building with the help of building-specific volumetric shape data on available cantonal or federal 3D city GML databases. This process is called baking. The textures are then formatted in a way so that they can be pasted onto the respective mesh faces of a 3D building object to create a textured 3D model of the building.

[1] Images with a ground resolution of 1-3 cm.

3.1.1 Geographic Scale of Raw Data

The raw data contains 1204 buildings which are located in a commune in the outskirts of Geneva called Collonge-Bellerive. The buildings cover an area of approximately 2 km^2 and the spatial distribution of these buildings can be seen in Figure 3.2.



Sources: (Swisstopo 2022), (SITG 2022)

Figure 3.2: The commune of Collonge-Bellerive to the east of Lake Geneva in the Canton of Geneva (left), The 1204 3D buildings that were acquired from Uzufly (right).

3.1.2 Structure of Raw Data

The raw data is organized in a predetermined structure. Each building is assigned a unique identification number, known as the EGID, which is created by the Federal Statistical Office in Switzerland and is currently used in all Federal and Cantonal databases (BFS 2022). This EGID number allows us to accurately locate each building from the raw data. In addition, each building in the directory created includes a 3D shape file in .obj format which is obtained from the territorial information system of Geneva (SITG) and a corresponding texture file in .mtl format, which is a high-resolution photo-mesh of the building captured by Uzufly. Figure 3.3 illustrates an example of the object and texture files of one building.

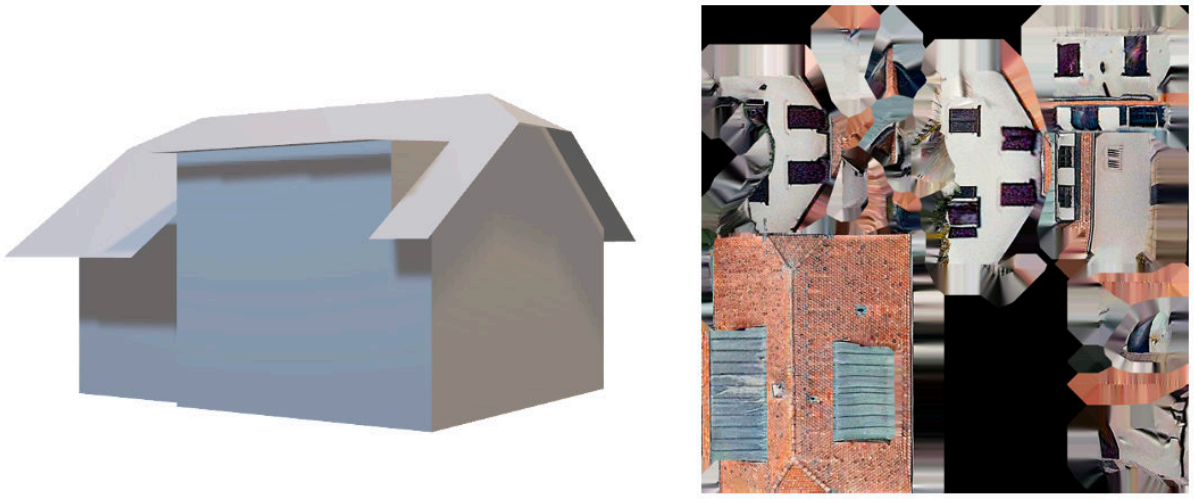


Figure 3.3: The 3D object file of a sample building in the raw data (left) and its corresponding texture file (right).

3.2 Data Pre-processing

In order to extract 2D images from the raw data, Blender, an open-source 3D graphics software, was utilized. Blender uses lighting and camera objects to create 2D rendered frames of 3D objects. When importing a 3D object file into Blender, it automatically recognizes its corresponding texture file in the raw data directory and imports both files together as one textured object. A key advantage of Blender is the ability to use custom scripts through its API to automate tasks. To this effect, a custom script was written that automated the following process shown in Figure 3.4:

- In the first step, a building object is imported into Blender
- Aligning the imported object's local XYZ axes with the global XYZ axes and moving object to origin
- Defining and directing 5 uniform light source objects at the 4 peripheral sides and top of the building respectively
- Defining and directing 5 orthographic[2] cameras with a scale of 40 meters at the 4 peripheral sides and top of the building object respectively
- Rotating the building object according to the normal vectors of the building peripheral meshes so that it fits in all the camera frames
- Rendering the scenes of all 5 cameras and exporting the images in a separate directory under the corresponding building EGID number appended by a numbered suffix which denominates the number of the camera (numbers 1 to 4 refer to the peripheral cameras and the number 5 refers to the top view camera)

After executing this script, five full-HD (1920x1080 pixels) RGBA[3] images per building object are rendered (5120 in total) and stored to a given directory. Figure 3.5 exhibits the custom script and the visual representation of the script syntax in the Blender interface. Furthermore, a separate python code is written to convert the RGBA images to RGB images so that all the processed images can be used for most computer vision algorithms which only take images with 3 channels as inputs.

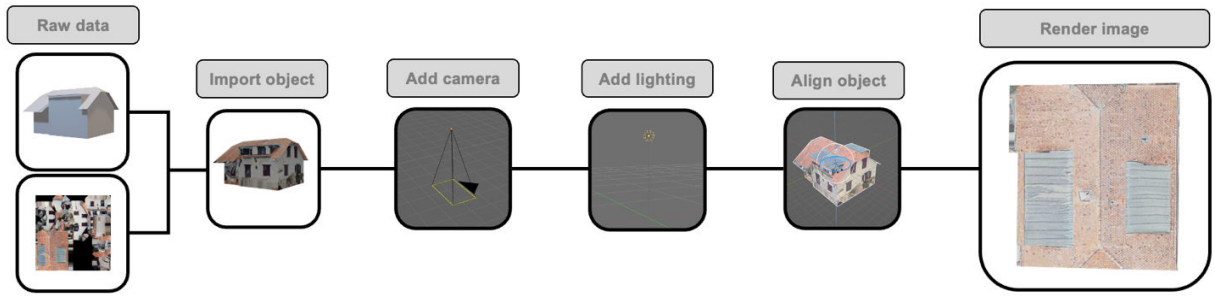


Figure 3.4: The process flow-chart of the automated data pre-processing. The rendered image shown here is the output of the top view camera which renders the rooftop.

[2] An orthographic camera is a camera that uses orthographic projection, due to which an object's size in the rendered image remains constant regardless of how far away it is from the camera.

[3] RGBA stands for Red, Green, Blue and Alpha which are the four channels in images with transparent backgrounds. Images with no transparent backgrounds are only represented by 3 channels, namely RGB.

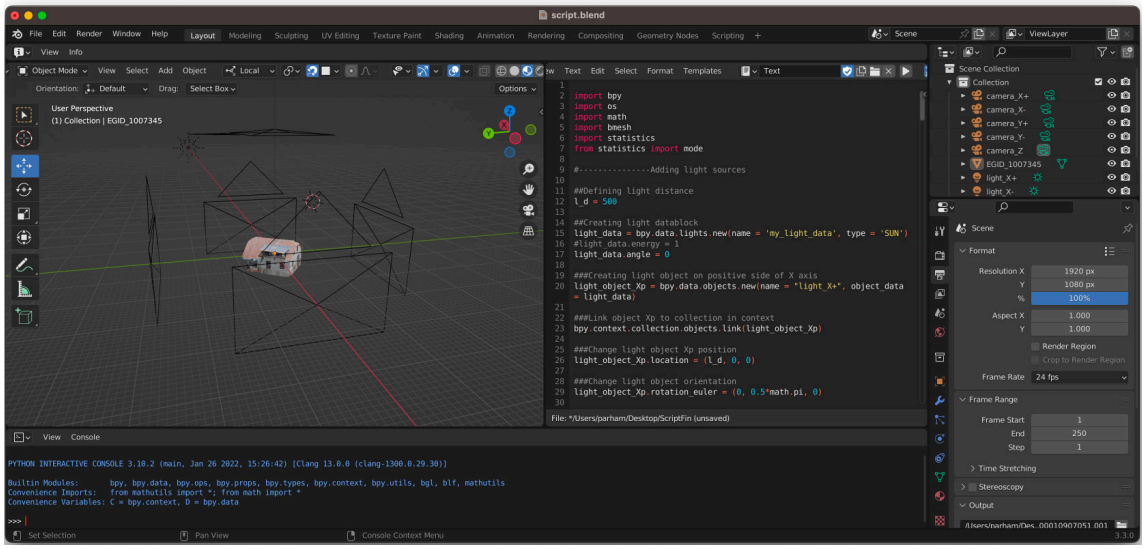


Figure 3.5: The Blender interface, with a panel on the left which shows a graphical representation of the script's syntax, and a panel on the right which displays the script itself. The left panel includes black boxes representing orthographic cameras, dotted spheres representing uniform light objects and the building object.

3.3 Description of Pre-processed Data

After extracting five high-resolution 2D images from the 3D textured building objects, the output images were examined. Upon examining the outputs, it is revealed that most of the output images suffer from poor quality issues which are further described in detail in this section.

3.3.1 Data Quality Issues

The majority of the processed images exhibit different forms of quality issues which are demonstrated in Figure 3.6. These quality issues are a direct result of photogrammetry errors within Uzufly's aerial captures and their data processing pipeline. The observed categories of data quality issues are as follows.

Edge Blur
Edge blur is a common problem in output images, where the background or building outlines appear blurry around the edges of the object. This can be seen in the first image of Figure 3.6, which shows edge blur on the upper and right side of a residential building roof rendering. This issue is often caused by a slight mismatch between the 3D object shapes obtained from the cantonal SITG database and the actual 3D volume of the building.

Warping
Warping is an issue that affects some pixels of the rendered image and is referred to as the relative distortion of the elements in the image. The second image of Figure 3.6 illustrates a rendering of a residential building rooftop with excessive warping, specifically in the bottom left corner of the image. This issue if often a direct result of insufficient overlap between aerial image captures during the flight plan.

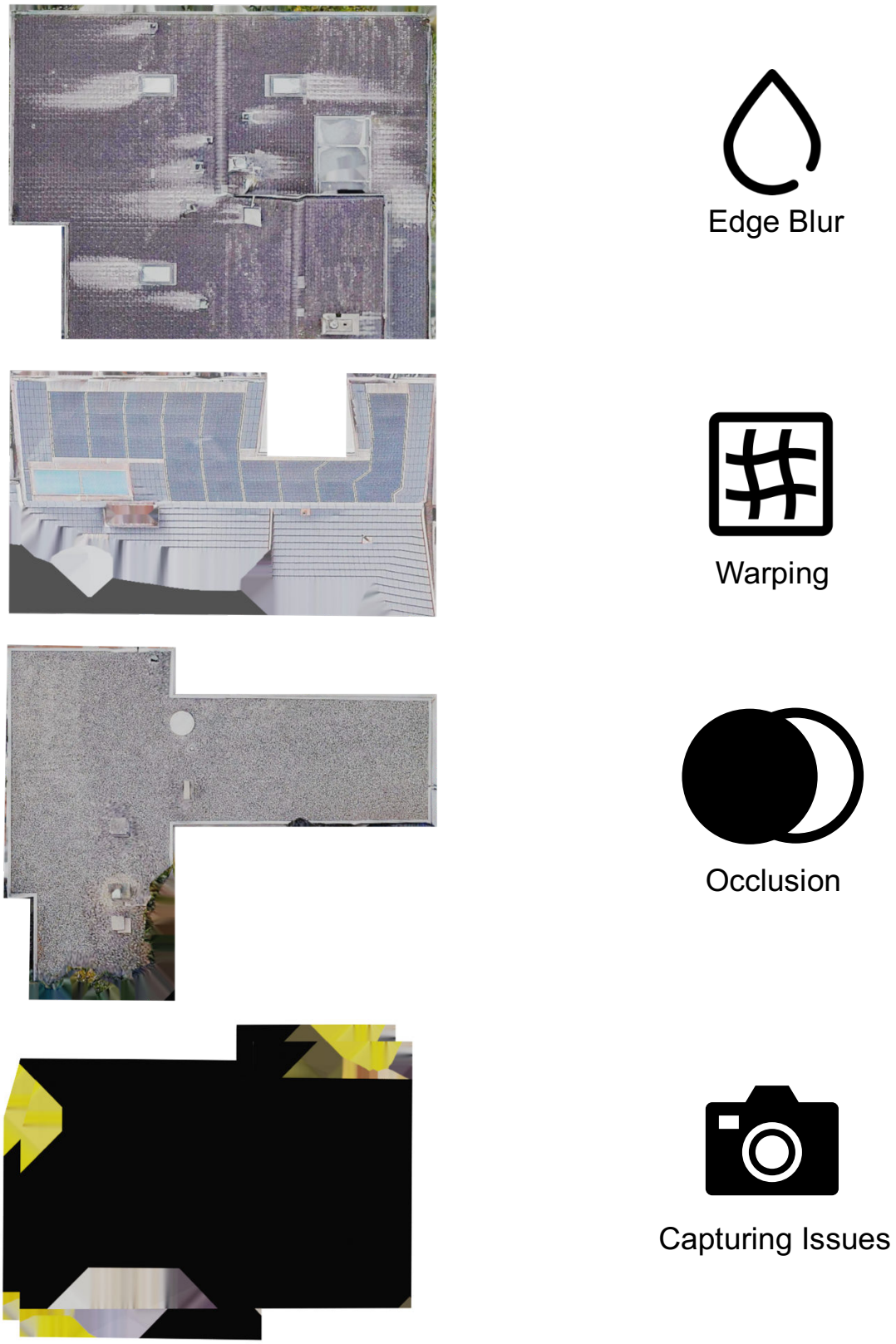


Figure 3.6: The various categories of quality issues in the processed images which can be attributed to Uzufly's photogrammetry practices and data processing pipeline.

Occlusions
Occlusions happen when an object blocks another object from being seen. The third image in Figure 3.6 shows a building rooftop with occlusions at the bottom. This problem often occurs with building rooftop renderings when nearby trees are towering over the building. The same issue can occur with building façade renderings when there are parked cars, bushes or hedges nearby.

Capturing Issues
Capturing issues result in undiscernible image captures, like the fourth image in Figure 3.6, and are rarely found in the processed image outputs. This problem may be attributed to camera-malfunctions during the flight plan.

3.3.2 Façade Renderings

The façade renderings are generally poor quality due to excessive warping and occlusions, as shown in Figure 3.7. These renderings cannot be used in any computer vision algorithm since these algorithms perceive and detect objects by extracting visual patterns and features from the images which in this case are distorted beyond recognition. It seems that for the time being, terrestrial imagery produces better large-scale façade captures compared to aerial imagery, due to camera technology and practical limitations.

3.3.3 Rooftop Renderings

As shown in Figure 3.7, the rooftop renderings show considerably better image quality compared to the façade images which conforms with the initial ultra-high-detail capture quality claims made by Uzufly. This is not surprising since the rooftops are directly visible to photogrammetry aircraft, thus resulting in close to no occlusions. However, the same cannot be said for building façades which are only visible from certain angles with multiple occlusions. It should be noted that edge blur, warping and occlusions are still present in the rooftop renderings, which necessitates filtering the images further so that they meet a certain standard.



Figure 3.7: Example of façade renderings (top) and a rooftop rendering (bottom) of the same building object.

3.4 Dataset Creation and Preparation

Creating a dataset is the most crucial and valuable step in any machine learning project as it directly impacts the performance and accuracy of the machine learning model. The dataset should have sufficient quantities of training data for the machine learning model, so it can learn to extract meaningful patterns and features from the data. In addition to the quantity of the data, the quality of the dataset will also determine how well the model can generalize to new, unseen data. A high-quality dataset should be representative of the problem domain, be balanced, and not contain any biases.
A representative dataset, is a dataset that contains a diverse set of examples that accurately reflect the problem domain. For example, if the task is to create a model that can identify objects in images, the dataset should contain a diverse set of images with different objects, backgrounds, and lighting conditions (as discussed in section 2.4).
A balanced dataset is one where the number of examples for each class is roughly the same. This is important as it ensures that the model does not have a bias towards any particular class. For example, if the task is to create a model that can classify images of cats and dogs, the dataset should contain an equal number of images of cats and dogs.
Bias in a dataset can lead to a model that performs well on the training data but poorly on new, unseen data. This can occur when the dataset does not accurately reflect the problem domain, or when certain groups are underrepresented in the dataset. For example, if the task is to create a model that can recognize faces, and the dataset only contains images of people with light skin, the model will likely perform poorly on images of people with dark skin.

3.4.1 Filtering and Selection of Processed Data

As mentioned in sub-sections 3.3.2 and 3.3.3, almost all façade renderings cannot be used for computer vision purposes, hence the scope of this thesis is limited to rooftop renderings only. Furthermore, in order to ensure that the dataset contains only the best quality images, the rooftop renderings were handpicked so they show limited signs of warping and occlusions. This process reduces the number of potentially useable images from 5120 to just 197.

3.4.2 Data Value Compared to Other Available Data

Although the filtered data may not be flawless, they are still more valuable than other comparable satellite or aerial images from other sources, as they have a much higher pixel density that allows them to paint a more accurate picture of the intricate textures of rooftops. Figure 3.8 shows a comparison between the filtered and processed Uzufly data and the most high-resolution publicly available alternative from Swisstopo.

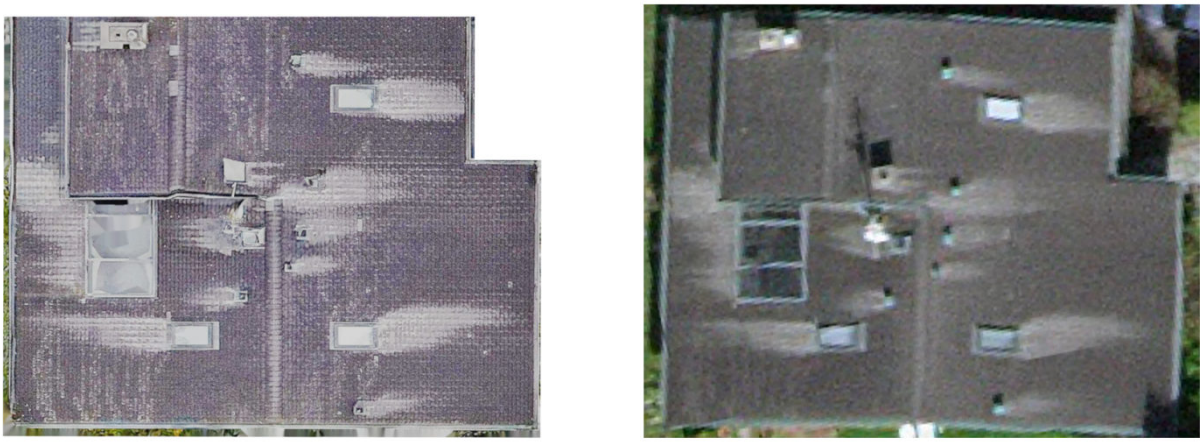


Figure 3.8: Filtered & processed Uzufly image (left) and Swisstopo's best aerial image[4] (right) of an example building in the dataset. The image on the left has 6 times more pixels than the image on the right and has richer textures.

3.4.3 Dataset Classes and Image Types

The following roof classes shown in Figure 3.9 are observed in the dataset: clay tile roofs, gravel cover roofs, roofs with bituminous sheathing and metal roofs.

Clay Tile Roofs
Clay tile roofs are sloped roofs made up of ceramic shingles which come in a myriad of shapes and colors and are the most common roofing material in Switzerland and in our dataset as well. Clay tile roofs are prone to efflorescence, flaking and soiling which deteriorate the visual and physical condition of the material (Franziska Gladis 2011). Typically, building components such as windows, chimneys and solar panels are observed on clay tile roofs.

Gravel Cover Roofs
Gravel cover roofs are flat roofs which are covered by a layer of loose gravel or ballast to protect the tar roof substrate from weathering and UV exposure. Gravel cover roofs are prone to biological growth (mostly moss growth) and mold buildup which affect the material (Paul Berdahl 2008). Building components such as vents, exhausts, HVAC outlets and piping systems are observed on a typical gravel cover roof and they are usually surrounded by parapets which run along the building outline.

Roofs with Bituminous Sheathing
Roofs with bituminous sheathing are almost identical to gravel cover roofs but without an extra gravel layer on top. In these types of roofs, a bituminous waterproofing system is used to protect the underlying roofing structure. Bitumen roofing can show signs or erosion, tears and water pooling in depressions which can stress and expose the roof structure. The same building components that are observed on gravel cover roofs can also be seen on roofs with bituminous sheathing.

[4] This image was taken from the SWISSIMAGE 10cm federal database which is a collection of roughly 427'000 ortho-photomosaic tiles of 1 km by 1 km with a ground resolution of 10 cm.

Metal Roofs
Metal roofs are sloped roofs which are made of galvanized metal sheets. Metal roofs are quite resilient and can last for a long time, showing only signs of surface rust throughout an extended period of time. Building components such as solar panels and exhausts are most frequently observed on metal roofs.

It's worth mentioning that while other types of roofs, like glass ceilings, were present in the dataset, their numbers were too small to be categorized as a distinct class.

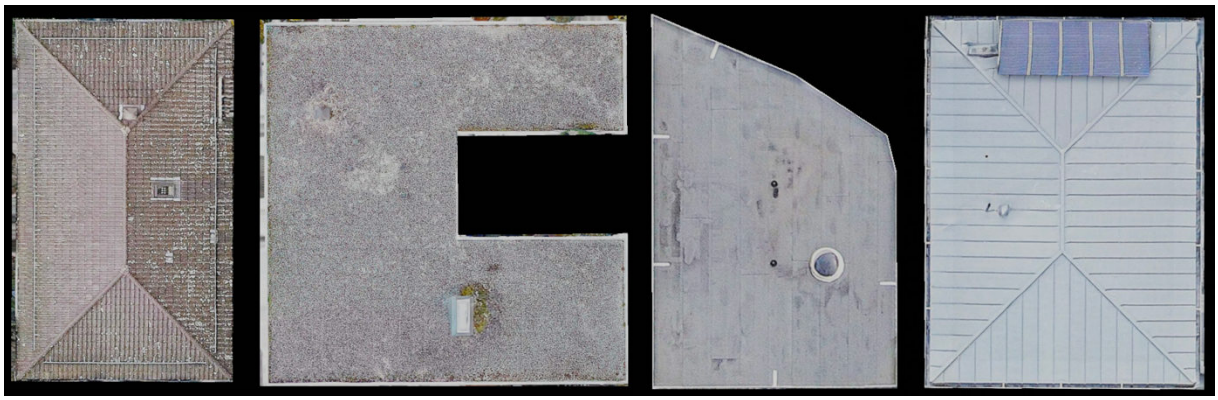


Figure 3.9: The four roof classes, from left to right: a) clay tiles, b) gravel cover, c) bituminous sheath, d) metal sheets.

Image Types Based on Number of Instances
There are different image types within the dataset with regards to the number of different roof classes in the image. The various image types are shown in Figure 3.10 and are as follows:

- Single instance images: images with only one roof class instance in the picture
- Double instance images: images with two roof class instances in the picture
- Multi-instance images: images with three or more roof class instances in the picture

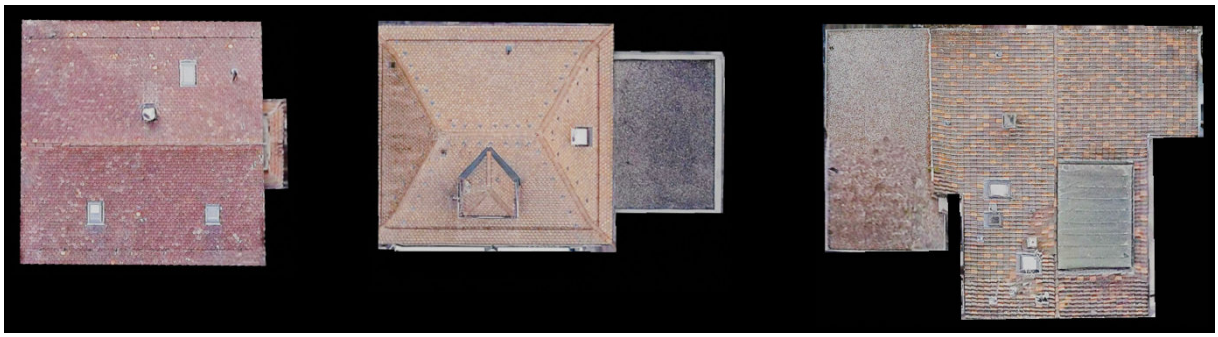


Figure 3.10: The three image types, from left to right: a) a single instance image consisting of a clay tile roof, b) a double instance image consisting of a clay tile and a gravel cover roof, c) a multi-instance image consisting of a clay tile, gravel cover and a metal roof.

The majority of the images are of the single instance type (77%) followed by double instance images (21%) and only a small amount of multi-instance roofs (2%).

3.4.4 Dataset Training and Validation Split

A training-validation split is a way to divide a dataset into two subsets: a training set and a validation set. The training set is used to train a machine learning model, while the validation set is used to evaluate the model's performance. This allows for an estimate of how well the model will perform on unseen data, which is important for assessing its generalization capabilities.
The ratio of the split can vary depending on the size of the dataset and the specific application, but a common ratio is 80/20, where 80% of the data is used for training, and 20% is used for validation. It's important to keep in mind that the validation set is not used for training the model in any way, it's only used to evaluate the performance of the trained model. In our specific case, since the model is going to detect roof instances, the 80/20 split is based on the total number of instances in our dataset and not the total number of images. Additionally, we will ensure that the class and image type distribution in the training and validation datasets match.

3.4.5 Dataset Annotation

The next step in creating the dataset is to annotate the training and validation sets so that they can be read and interpreted by computer vision models. The annotations are carried out using Make Sense AI which is a free online annotation tool (Skalski 2019). During the annotation, segmentation mask polygons are carefully drawn on each roof instance and the associated roof class is manually attributed to the segmentation mask. Instances that were not relevant, such as balconies and glass ceilings, were not included in the annotation. Additionally, edge blur and occlusions were not included in the segmentation masks to prevent the computer vision model from interpreting them as part of the material texture. Make Sense AI automatically creates and adds the bounding boxes of each instance to the annotation. Figure 3.11 shows the Make Sense AI interface and some annotations of the training set. Both the training and validation set annotations can be exported as COCO style annotations (discussed in sub-section 2.3.2) and are saved as two separate JSON files. By using the COCO-style annotation format, we ensure that the dataset can be readily used with state-of-the-art object detection and instance segmentation models.

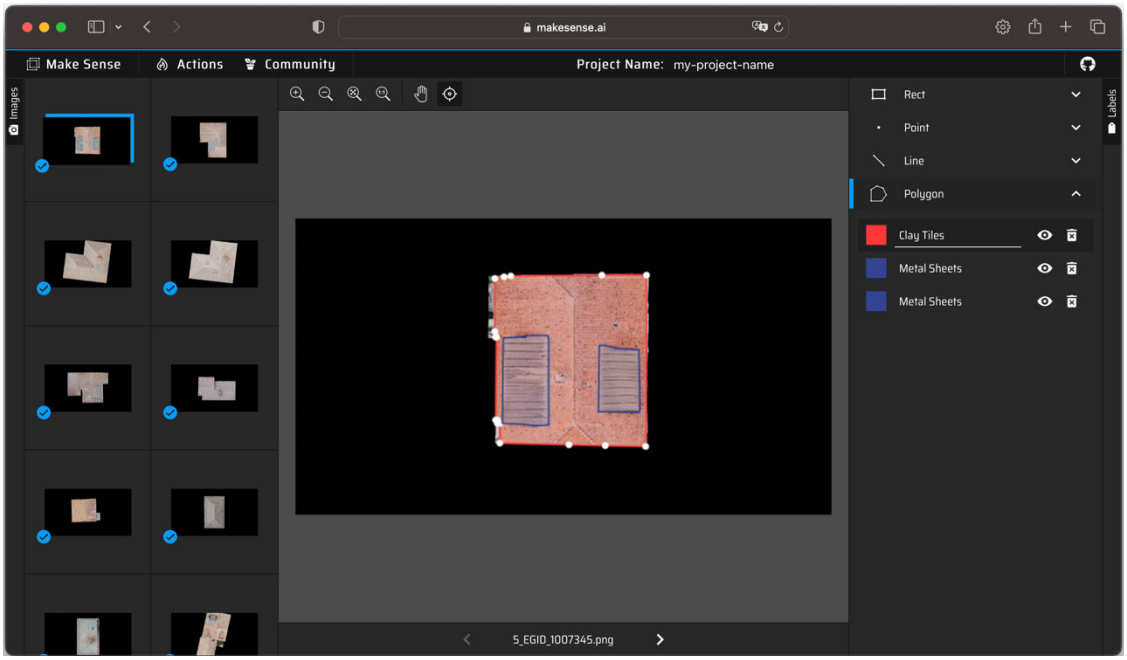


Figure 3.11: The training set annotations in Make Sense AI. The annotations are a series of polygons which mask every roof instance and classify the roof instances according to its respective class.

3.4.6 Dataset Augmentation and Class Imbalance

Figure 3.12 shows the distribution of classes in both the training and validation sets by counting the number of annotated instances. It is observed that both sets are afflicted by class imbalance (discussed in section 2.4) and are biased towards the clay tile roof class which makes up most of the instances in the dataset, while there are very few instances of bituminous sheath and metal sheets classes. In order to remedy this issue, the underrepresented classes were oversampled by augmenting single instance images from the minority classes to prevent the addition of any other unwanted classes.

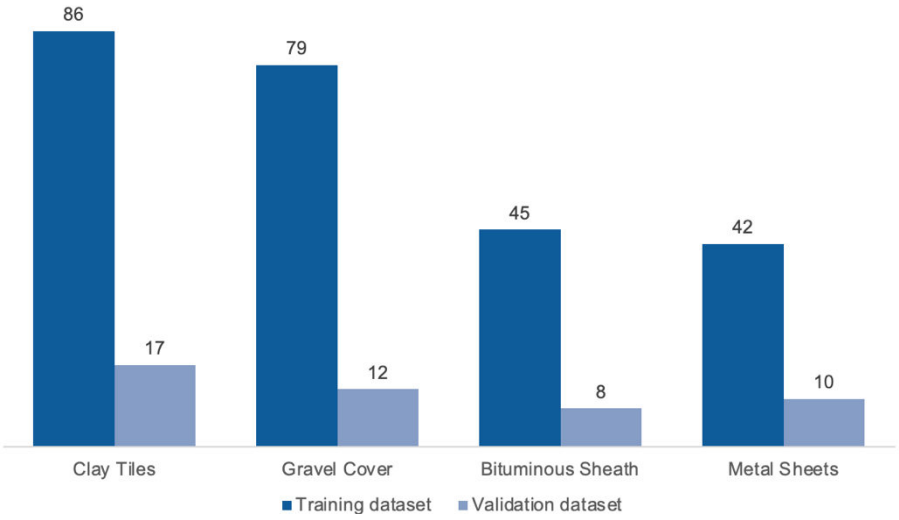


Figure 3.12: Number of annotated instances across all roof classes in both the training and validation sets.

The Albumentations image augmentation library (Buslaev 2020) was utilized to augment the images, by altering the contrast, gamma, saturation and color jitter (Figure 3.13). This ensures that the model is exposed to rooftop images under different lighting conditions (by changing contrast and gamma), color schemes (saturation and color jitter) and images with some background noise (saturation). These augmentations make the trained model more robust and less biased towards any particular class.

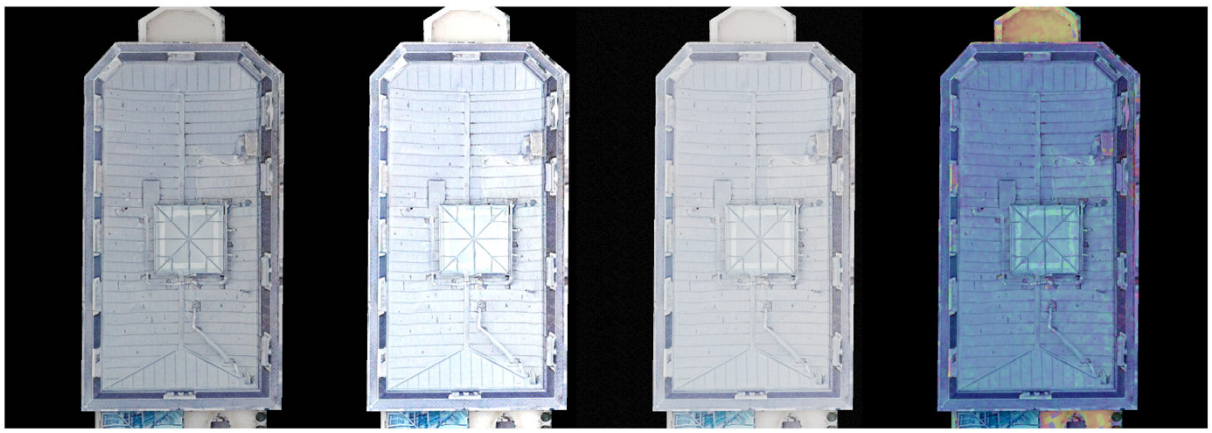


Figure 3.13: Example augmentations of a metal sheet roof image, from left to right: a) original image b) image with altered contrast c) image with altered gamma d) image with altered saturation and color jitter.

Subsequently, the augmented images are appended to the respective JSON annotation files together with their corresponding bounding boxes and segmentation masks. Figure 3.14 shows the resulting equal distribution of the roof classes in the augmented training and annotation set. It should be noted that by augmenting the dataset, the total number of images in the dataset increased from 197 to 310.

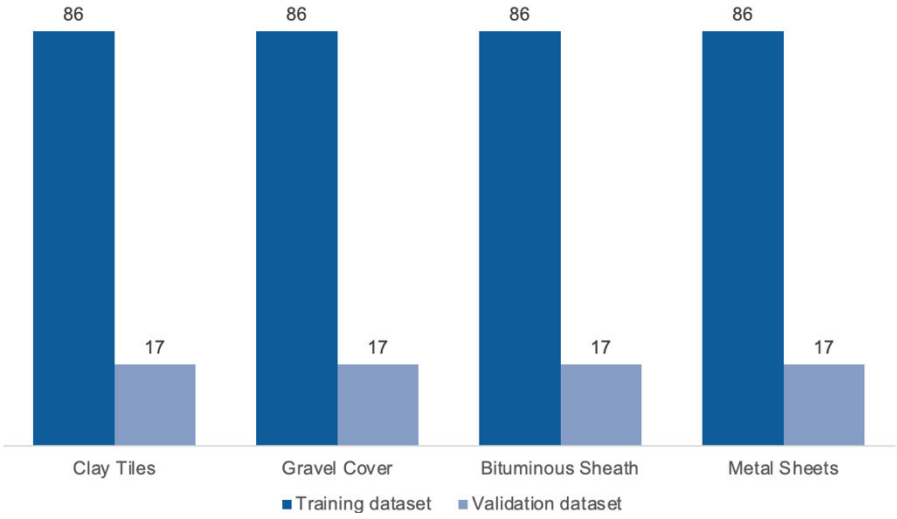


Figure 3.14: Number of annotated instances across all roof classes in both the training and validation sets post-augmentation.

3.5 Computer Vision Model

This section covers the specifics of training a state-of-the-art instance segmentation computer vision model on our custom dataset which was described in the previous section.

3.5.1 Transfer Learning

Transfer learning is a technique that allows for reusing a pre-trained model on a related task, instead of training a model from scratch. Many state-of-the-art large and complex computer vision models with millions of parameters have already been engineered to learn and abstract general features from large image datasets. These models can be customized and fine-tuned to our own specific task. However, it is important to note that the pre-trained model's training data should be similar to the new task's data for this technique to be effective. In this thesis, pre-trained models trained on COCO dataset were used as a starting point for training, since this dataset contains many buildings, making these trained models ideal choices for the task at hand.

Figure 3.16 shows how transfer learning is carried out more specifically. Typically, pre-trained instance segmentation model architectures are comprised of complex inner layers and a final layer in their architecture which is called the head or top layer. The inner layers of the model architecture extract features from regions of interest within the input image and the head layer classifies those regions.
The head layer contains a number of classification nodes which correspond to the number of classes within the training dataset. In transfer learning, the inner layers are frozen and the head layer is replaced by a customized classification layer containing nodes which correspond to our own custom dataset classes. This customized model is then iteratively trained on our custom dataset, so the head layer weights are fine-tuned to the task of rooftop material detection.

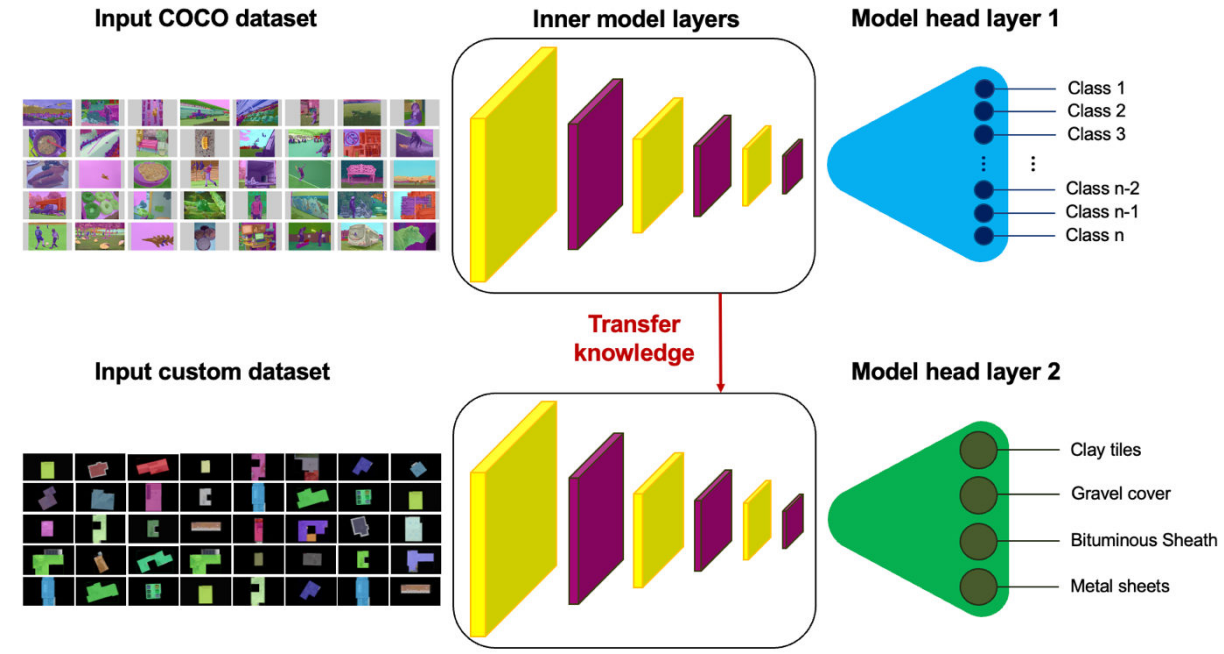


Figure 3.15: Transfer learning implementation scheme.

3.5.2 Facebook Research Detectron2 Library

This thesis will utilize Detectron2 (A. K.-Y. Yuxin Wu 2019), which is an implementation of Mask-RCNN developed by Facebook Research. Detectron2 is a popular state-of-the-art open-source machine learning library for computer vision tasks, particularly object detection and instance segmentation. It is built on the PyTorch[5] framework and provides a high-level API for training and evaluating models. It also provides support for a wide range of pre-trained models and allows for easy customization and experimentation.

One of the other key features of Detectron2 is its modular design, which allows for easy swapping of different components such as the backbone network, the heads, and the data input pipeline. This allows users to easily test different architectures and configurations, and compare their performance. It also includes a wide range of built-in metrics for evaluating the performance of models, such as the popular COCO evaluation metric discussed in 2.6.3 which is used in this thesis. Moreover, Detectron2 also uses a permissive licensing scheme[6] under which pretrained and customized models can be used for research and even commercial purposes. Detectron2 also includes a number of additional tools and features, such as support for training with distributed computing, and the ability to export models in the ONNX[7] format for deployment in other environments. It also provides an easy-to-use visualization tool for inspecting the results of object detection and instance segmentation models, including an interactive web-based interface. This makes it a powerful tool for researchers and practitioners working in the field of computer vision.

[5] PyTorch is an open-source machine learning library for Python which provides a flexible and high-performance platform for building and deploying deep learning models.

[6] Detectron2 uses the Apache 2.0 license.

[7] Open Neural Network Exchange Format (ONNX) is a format that allows for the representation of various ML models and supports multiple frameworks like PyTorch, TensorFlow, Keras, SAS, Matlab and others, making it easier to convert models between frameworks.

3.5.3 Model Training

The model training was carried out on a Google Colab[8] notebook with access to free standard GPUs since the pre-trained models from the Detectron2 library are highly efficient and fine-tuning them can be done with low computation cost.

After cloning and setting up of the required libraries and importing relevant modules and utilities, the first step of the model training is to load and register the training and validation datasets as COCO-like datasets, so they can be read and interpreted by the models. The registered datasets are then visualized using built-in visualization modules to check whether the models are interpreting the annotated images correctly. Figure 3.16 shows that the images in both datasets are loaded and interpreted correctly.

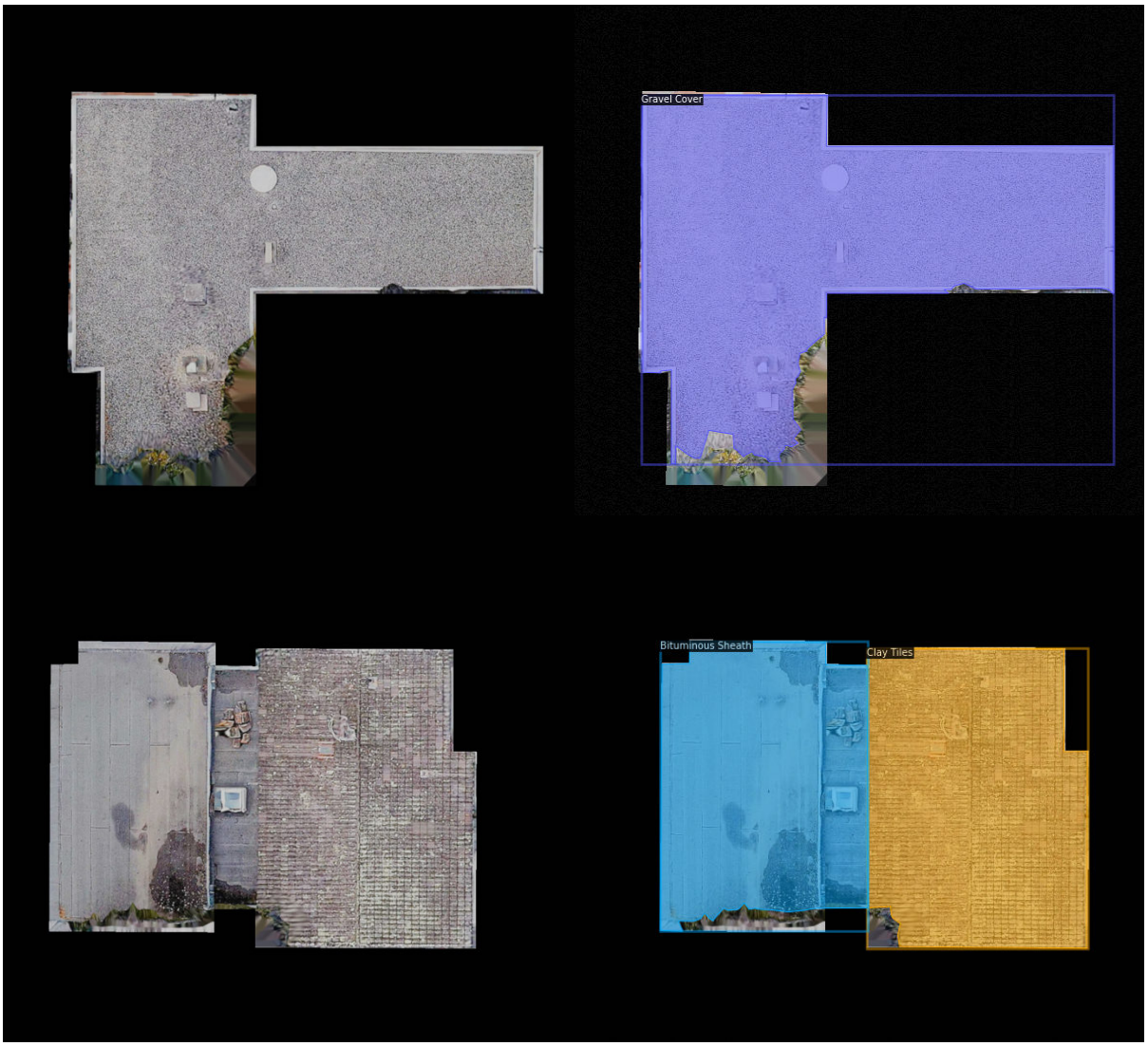


Figure 3.16: Input ground truth images (left) and their annotations as interpreted by the Detectron2 models (right). The images on the top are of a rooftop sample from the training dataset and the images on the bottom are of a rooftop sample from the validation dataset.

[8] Google Colab is a free cloud-based platform that allows users to write and execute Python code in a web browser, using the Jupyter notebook interface. It provides access to computational resources, including high-performance GPUs, making it well-suited for machine learning and deep learning tasks.

The next step is to define a training configuration which contains information on transfer-learning parameters such as number of classes and also information on the specific pre-trained model, its initial weights and hyperparameters. This configuration is then used by a trainer module to train the model on the training dataset. Throughout training, batches of training data are passed to the customized model architecture in multiple iterations, throughout which the model adjusts its weight parameters so it minimizes the training loss function. The default Detectron2 trainers only calculate and log the training losses in each iteration, therefore modifications were made to the training configuration so it calculates and logs the validation losses as well (Ortega 2020). Depending on the hardware, pre-trained model and number of training iterations the training could take anywhere between 20 minutes to 2 hours. In our case, two pre-trained instance segmentation models are trained for 3000 iterations on Google Colab's standard Nvidia T4 GPUs. As we will see in the next chapter, training the models for more than 3000 iterations results in minute performance improvements since the model losses stabilize after 3000 iterations, indicating that the model losses have reached an optimum. A short description of the two pre-trained models that were used is provided below.

Mask R-CNN R50-FPN-3X

This pre-trained model uses a ResNet-50[9] and FPN[10] backbone with standard convolution and fully connected heads for mask and bounding-box prediction. The model was trained for approximately 37 epochs on the COCO dataset. The model is light and capable and training takes 0.209 seconds per iteration on average (R. G. Yuxin Wu 2021). Since training on this model is fast, it is used to experiment with different configurations and datasets.

Mask R-CNN X101-FPN-3X

This pre-trained model uses a ResNeXt-101[11] and FPN backbone with standard convolution and fully connected heads for mask and bounding-box prediction. It also has been trained for approximately 37 epochs on the COCO dataset. This model is a heavy high-performance model and training takes 0.638 seconds per iteration on average (R. G. Yuxin Wu 2021). According to the fact that this model is the best-performing model of all the instance segmentation pre-trained models, it is used as the main model for this thesis.

[9] ResNet50 and ResNet101 are deep convolutional neural network architecture that are 50 and 101 layers deep respectively. They were developed by Microsoft Research and won the first place in the Large-Scale Visual Recognition Challenge (ILSVRC) in 2015 (X. Z. Kaiming He 2015).

[10] FPN stands for Feature Pyramid Network, which is a backbone architecture for object detection and instance segmentation in computer vision (T.-Y. a. Lin 2016).

[11] ReXt-Net101 is a variant of the ResNet101 architecture developed by Facebook's AI research team in 2017, that utilizes the ResNeXt module, which is an improvement on the traditional Residual block used in ResNet architectures (Saining Xie 2016).

4 Results and Discussion

In this chapter the trained instance segmentation model performance is examined and the model is evaluated to gauge its precision and generalization. Additionally, we discuss the capabilities and limitations of the model and provide insight into how and why detection errors occur. Lastly, we explore potential usecases for the model's output data.

4.1 Model Performance

Figure 4.1 shows the model loss curve diagram which is typically used in any machine learning project to determine whether the model has actually learned and abstracted features from the training data or not. The diagram shows the losses of the model on the training and validation datasets with each training iteration. Throughout its training, the model updates its weights to minimize a certain loss-function, thereby optimizing its cognition of the task at hand. In our specific case, the loss function is designed to capture the losses of classifying the materials, the losses associated with regressing a bounding box around the detected instance and the losses of the created binary mask for each instance all at the same time (for more information on the losses see Appendix A1).

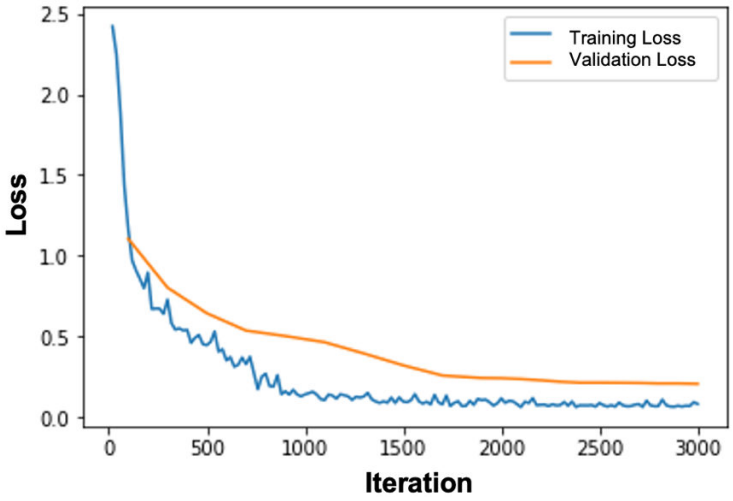


Figure 4.1: Trained model loss curve diagram. The model doesn't show any signs of underfitting or overfitting.

Both loss curves show a steady downward decline with minimal fluctuations, plateauing at the end with higher iterations. Moreover, the training and validation losses closely follow each other. These are all indications that the model has indeed learned from the training dataset to detect and segment rooftops. The loss curve is typically used to check whether the trained model shows signs of underfitting or overfitting which are described in detail below.

Underfitting

Underfitting in machine learning occurs when a model is not able to capture the underlying patterns in the data (Brownlee 2019). This can happen when the model is too simple or when it is not trained for enough iterations. As a result, the model will have poor performance on both the training and validation sets. Underfitting can be detected by the training loss curve only and is characterized by a training loss that remains relatively constant or a training loss that continues to decrease until the end of training.

Overfitting

Overfitting in machine learning occurs when a model is too complex and captures noise or random variations in the training data instead of the underlying patterns (Brownlee 2019). As a result, the model performs well on the training data but poorly on unseen data (validation data). Overfitting is problematic because as the model becomes more specialized to the training data, its ability to generalize to new data decreases, leading to a higher generalization error. Overfitting can be detected by the training and validation loss curves and is characterized by a training loss that keeps decreasing and a validation loss that keeps increasing with higher iterations.

4.2 Model Inference

After the trained model is checked for underfitting and overfitting, the model weights are stored in an output directory and can be used to compile the same trained model in inference mode, so the model can make inferences on out-of-sample images. It is important to note that the model is set to only output detections that have a confidence score of at least 70% during inference. In other words, if the model detects an instance with a probability of being correctly labeled below 70%, it will be discarded.

4.2.1 Inference on Validation Dataset

The model outputs the following in inference mode:

- Segmentation masks over detected instances
- Bounding boxes surrounding the segmentation masks of detected instances
- Instance classes of the detected instances and their confidence score

Inferences are made on the complete validation dataset and are compared to the original validation dataset annotations. In general, the model demonstrates outstanding performance in both classification and segmentation tasks. Figure 4.2 shows that the model has learned to classify and segment single instance images quite well. The model has even learned to exclude occlusion errors just as intended.



Figure 4.2: Model inference on a sample gravel cover roof from the validation dataset. From left to right: a) ground truth, b) annotation, c) inference. The model has successfully excluded a tree occlusion on the bottom of the image from the segmentation mask.

Figure 4.3 shows that the model is capable of classifying and segmenting more complicated multi-instance images with an acceptable level of precision as well.

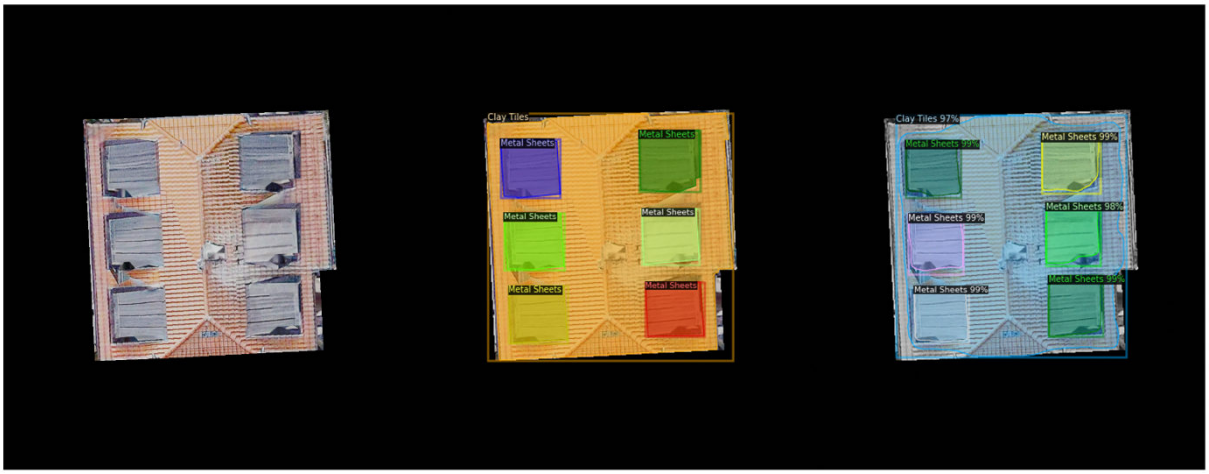


Figure 4.3: Model inference on a sample multi-instance image type from the validation dataset. From left to right: a) ground truth, b) annotation, c) inference

4.2.2 Inference Limitations and Capabilities

The model exhibits an exceptional performance when it comes to detecting and classifying roof instances. Figures 4.4, 4.5 and 4.6 show some of the wrong predictions the model has made. In most mispredictions, the model has confused a glass ceiling or a balcony for a metal roof or bituminous sheath. This error mainly occurs due to the close resemblance between the striped patterns in some balcony pavements, glass ceiling mullions and the metal sheets and bituminous layers.

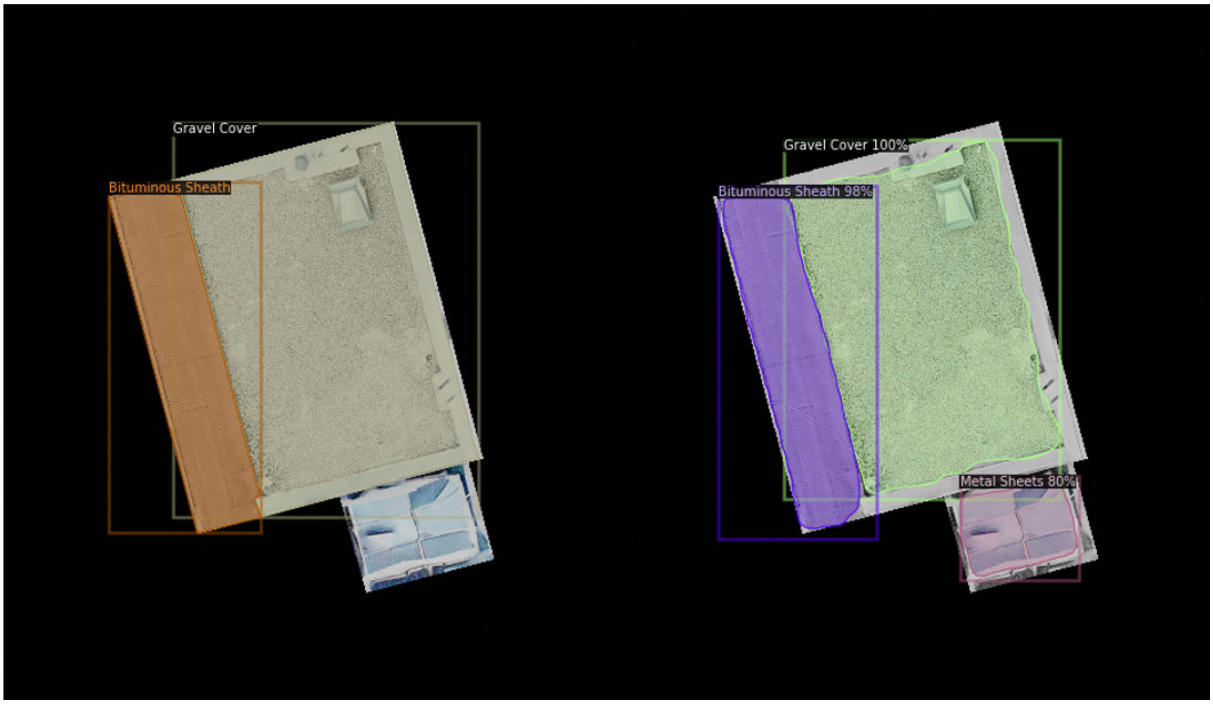


Figure 4.4: Model inference on a sample validation image, from left to right: a) annotation, b) inference. The model has mistakenly identified the bottom glass ceiling as a metal roof.

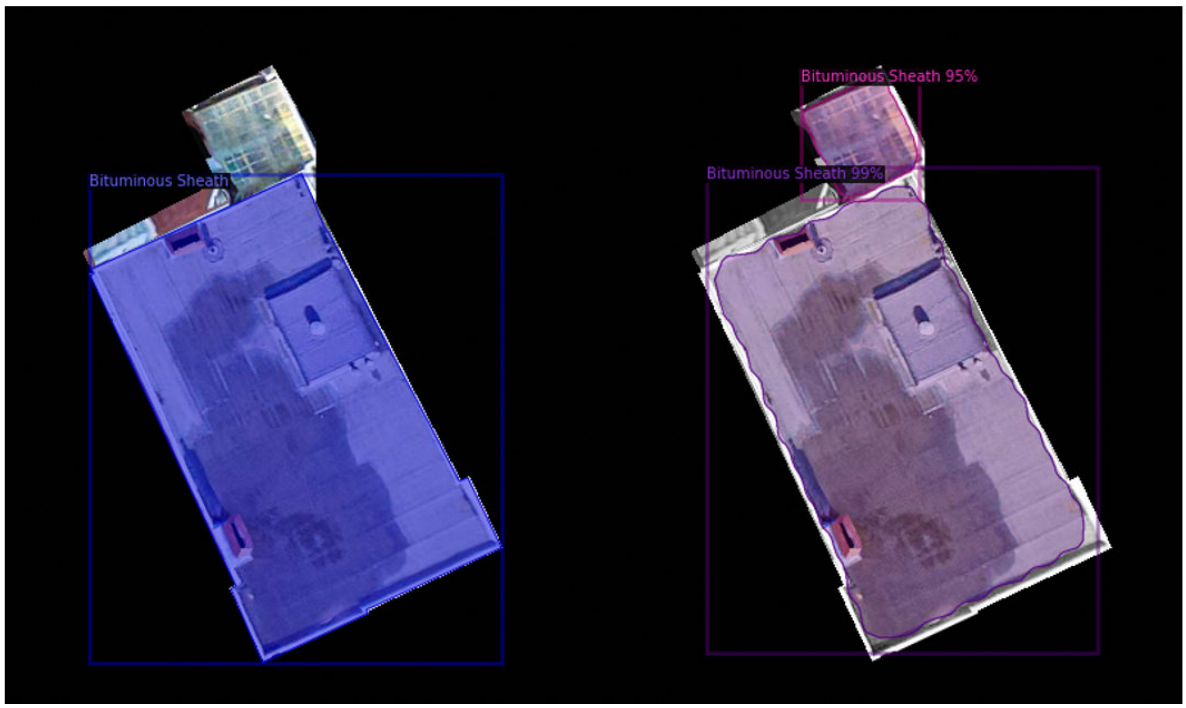


Figure 4.5: Model inference on a sample validation image, from left to right: a) annotation, b) inference. The model has mistakenly identified the glass ceiling at the top as bituminous sheathing.

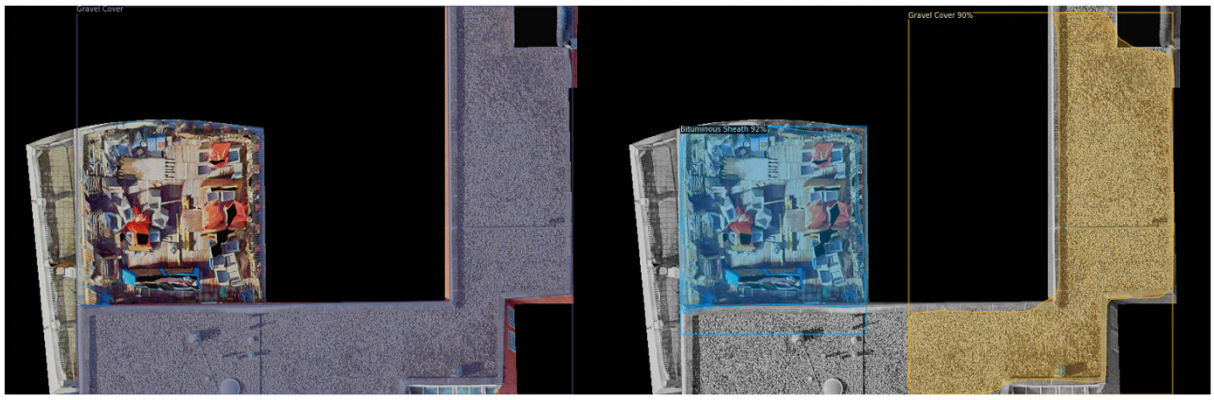


Figure 4.6: Model inference on a sample validation image, from left to right: a) annotation, b) inference. The model has mistakenly identified the balcony on the left as a roof with bituminous sheathing.

While the model excels as a classifier, it is not as successful in generating segmentation masks that accurately outline the detected instances. Figure 4.7 shows an example of such an issue. The root causes of this issue can be traced back to the dataset image quality issues discussed in detail in subsection 3.3.1. Specifically, the presence of edge blur in the majority of images and its subsequent exclusion from image annotations, confuses the model and it can't correctly extend the segmentation mask to completely cover a detected instance.

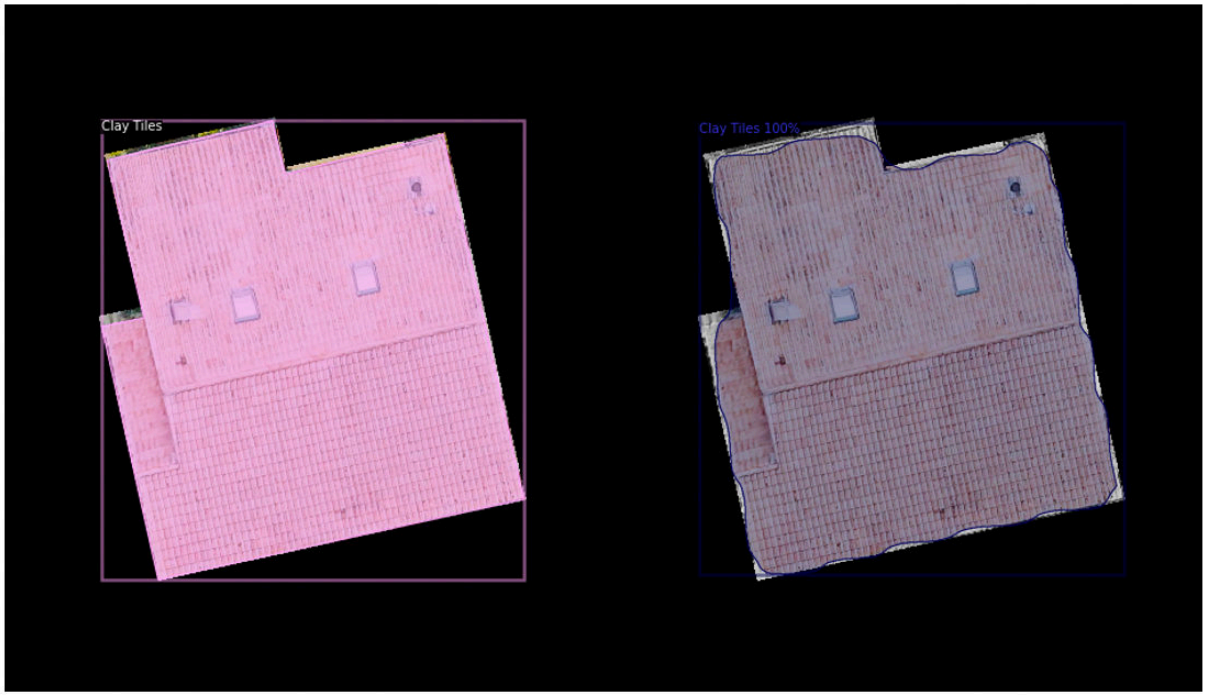


Figure 4.7: Model inference on a sample validation image, from left to right: a) annotation, b) inference. The model has difficulties identifying instance edges and cannot extend masks to cover a detected instance completely.

Nevertheless, despite such limitations the model shows strong generalization on out-of-sample images such as the ones from the validation dataset. To test the model limits and to further drive this point home, a few random images of rooftops from the internet were given to the model for inference. Figure 4.8 shows examples of model inferences on these images.

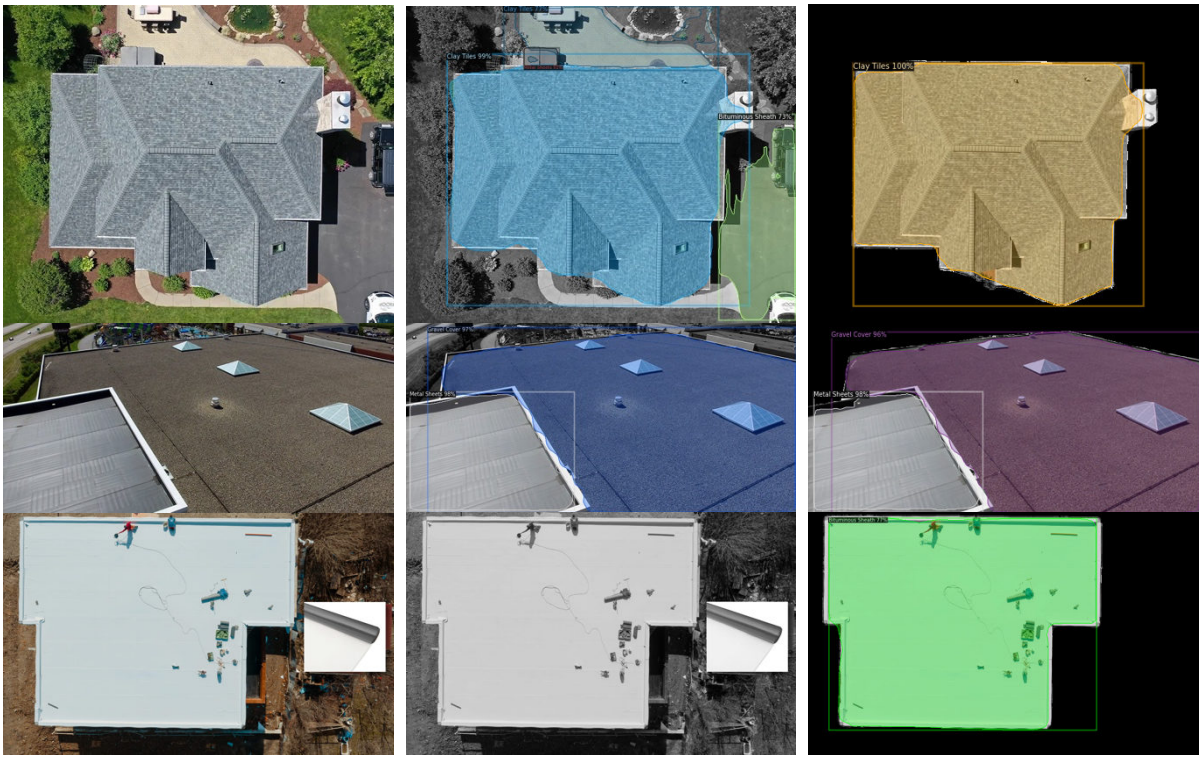


Figure 4.8: Model inference on random images, from left to right: a) ground truth, b) inference on image with background, c) inference on image without removed background

It is observed that the model can detect instances of roof materials not only from the top view, but also from different skewed perspectives. On the top row of Figure 4.8, the model has correctly detected and segmented a clay tile roof along its outlines. It has also identified the backyard pavement as clay tiles and the asphalt drive-way as bituminous sheathing, which shows that the model cannot comprehend backgrounds since it associates every detected material with a roof instance. On the second row, the model has successfully disregarded the background, and has detected and segmented the metal and gravel cover roof instances with acceptable precision. Keep in mind that the model has only been trained on bird-eye view rooftop images and has learned to generalize the patterns it has learned to various perspectives and other uniform image transformations.

It should be noted however, that since the model was trained on roof instance images without any background context, it can also perform much better on images with no background. This can be shown on the third row of Figure 4.8 where the model has failed to detect the bituminous sheathing roof due to the complicated background context of the image but managed to detect it once the background was removed.

4.3 Model Evaluation

Table 4.1 and Table 4.2 show the model evaluation on the validation dataset for the entire model and for each class respectively.

Table 4.1: Model Segmentation Evaluation

AP	AP50	AP75
81.43%	93.78%	88.07%

Table 4.2: Model Segmentation Evaluation Per-Class

Class	AP
Clay Tiles	89.28%
Gravel Cover	73.36%
Bituminous Sheath	77.96%
Metal Sheets	83.11%

Note that these evaluations only account for the generated segmentation masks. The evaluations can also be conducted for the generated bounding boxes which is irrelevant for our current task.

The model has an average precision of 81.43% and an AP50 of 93.78%. This suggests that the model is highly effective as a rooftop material classifier, as it demonstrates a higher precision score on metrics with lower IoU thresholds, which are more lenient in evaluating segmentation. However, the model's precision score is lower in metrics with IoU thresholds higher than 0.75, which is a result of the model's difficulty in accurately detecting instance edges and creating masks that fully cover them, as previously discussed in sub-section 4.2.2.

Furthermore, Table 4.2 shows that the model is not biased towards any particular class. This can be directly attributed to the dataset augmentation carried out in sub-section 3.4.6. The model seems to have the best performance on clay tile roofs, followed closely by metal roofs. The model performs less

well on gravel cover and bituminous sheathing roofs. This can be explained by how edge blur and warping affect each roof type differently. In clay tile and metal roofs the blurs are much more distinct than the patterns in the instance material. However, this cannot be said for gravel cover and bituminous sheathing roofs where any edge blur or warping looks identical to the instance material patterns which exacerbates model confusion even more. Ideally, a better-quality dataset with virtually no edge blur or even slight warping can fill this slight performance gap in detecting different rooftop classes.

4.4 Creating Material Take-off from Inferences

Having developed a model capable of segmenting and classifying various rooftop instances, the logical next step is to calculate the surface area of the segmented materials. In order to create a material take-off from the segmented rooftop images, the following process was adopted which is shown schematically in Figure 4.9. First, an additional python script was written in blender which can calculate the rooftop surface area of the 3D building objects (discussed in sub-section 3.1.2). Furthermore, the material detection model outputs the classes of the detected instances and their corresponding binary segmentation mask (for more information see Appendix A2). Subsequently, the proportionate pixel area of each identified instance mask is computed, and then multiplied by the overall rooftop surface area to estimate a material take-off. It is worth noting that as this method calculates pixel areas of segmentation masks on 2D projections of the rooftop, it may introduce inaccuracies in determining the surface area of identified materials on slanted roofs. Additionally, the fact that the segmentation masks do not completely encompass the identified material instances, may also introduce some errors in this calculation.

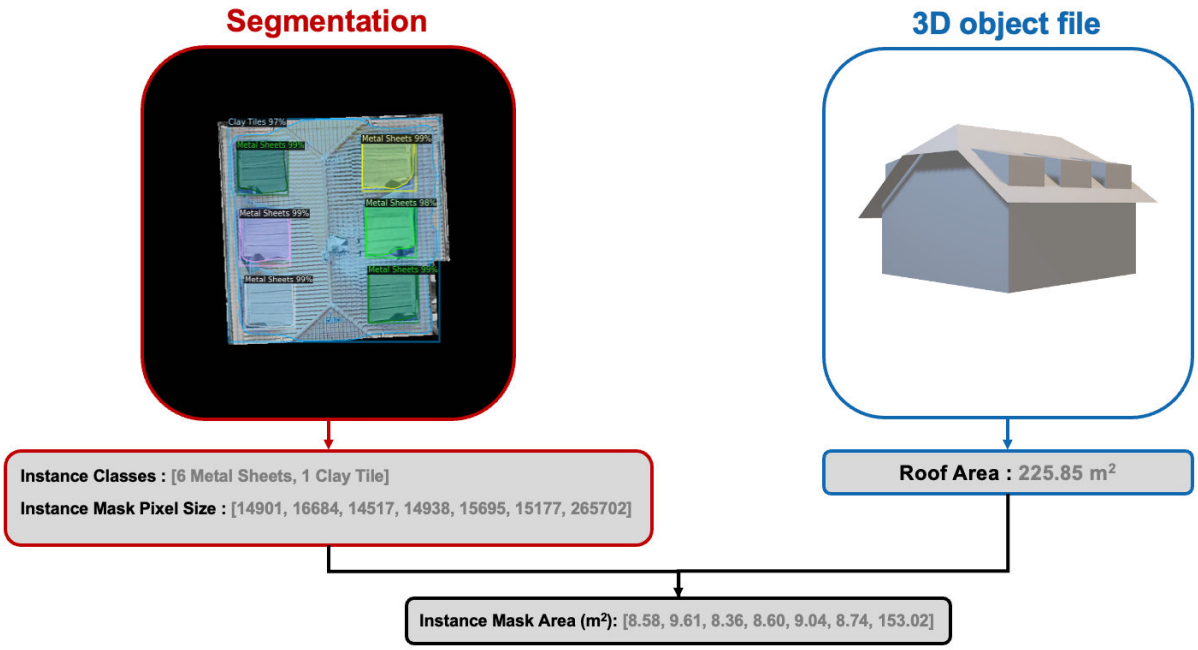


Figure 4.9: The process of calculating an approximate material take-off from a segmented image.

5 Conclusion and Outlook

The current linear take-make-waste approach in the architecture, engineering, and construction (AEC) industry has contributed significantly to resource depletion, greenhouse gas emissions, and adverse environmental impacts. A circular economy is therefore essential for maintaining a sustainable built environment. However, in order to mobilize the industry towards a more circular paradigm, data on the available material stock is needed.

This thesis explored how photogrammetry data and advanced computer vision techniques can be leveraged to fill the material data-gap and to facilitate the reuse of materials in a circular framework within the AEC industry. Large-scale high-detailed 3D digital shadows of buildings were acquired from an innovative aerial imagery startup and a complete methodology was developed to pre-process and annotate the data into a dataset of high-resolution rooftop imagery. This dataset was then later used to train state-of-the-art machine learning models to detect, segment and quantify rooftop materials. With improved photogrammetry data in the future, both in terms of quality and quantity, this model can be scaled up to infer not only building rooftop materials but also build facade materials and components at a city scale. Furthermore, the final models and methods developed in this thesis, lay the foundation for the creation of an urban-scale spatial material stock database which would aid stakeholders in designing and managing the built environment for circularity.

However, more research is necessary to enhance the circularity of building materials through streamlined and optimised reuse. Future research should aim to expand the dataset used to train material detection models, incorporating more material types for improved detection precision. Another viable and valuable area of focus would be to create quantifiable metrics to evaluate the reuse potential of detected materials, and to train a separate computer vision model to assess the condition of available material stock. This would require a large amount of high-resolution data, similar to the ones used in this thesis, but would result in a more comprehensive understanding of the material stock.

Ultimately, despite the recent efforts to transition towards a circular economy, a significant amount of work remains to be done to address the challenges posed by the linear economy. The consequences of the current linear economy only become more pronounced with time, making it imperative to pursue and promote further research in the AEC industry. This research can contribute to developing new technologies, methods, and best practices for efficient and sustainable material management in the industry in order to reduce waste, conserve resources and promote a more sustainable future.

6 Bibliography

BFS, Bundesamt für Statistik. 2022. *Registerharmonisierung - EGID/EWID.* Accessed October 2022. https://www.bfs.admin.ch/bfs/de/home/register/personenregister/registerharmonisierung/egid-ewid.html.

Brownlee, Jason. 2019. *How to use Learning Curves to Diagnose Machine Learning Model Performance.* February 27. Accessed December 2022. https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/.

Buslaev, Alexander and Iglovikov, Vladimir I. and Khvedchenya, Eugene and Parinov, Alex and Druzhinin, Mikhail and Kalinin, Alexandr A. 2020. "Albumentations: Fast and Flexible Image Augmentations." *Information* Article number: 125.

Duan, H., Miller, T. R., Liu, G., & Tam, V. W. Y. 2018. "Construction debris becomes growing concern of growing cities." *Waste management, 83* 1–5.

Franziska Gladis, RhenaSchumann. 2011. "Influence of material properties and photocatalysis on phototrophic growth in multi-year roof weathering." *International Biodeterioration & Biodegradation* 36-44.

Google. 2022. *Machine Learning Glossary.* November 7. Accessed November 2022. https://developers.google.com/machine-learning/glossary#e.

Goutte, Cyril and Gaussier, Eric. 2005. "A Probabilistic Interpretation of Precision, Recall and F-Score, with Implication for Evaluation." *Proceedings of the 27th European conference on Advances in Information Retrieval Research.* Santiago de Compostela: Springer-Verlag. 345–359.

Johnson, J.M., Khoshgoftaar, T.M. 2019. "Survey on deep learning with class imbalance." *Journal of Big Data volume 6* Article number: 27.

Kaiming He, Georgia Gkioxari, Piotr Dollár, Ross Girshick. 2017. "Mask R-CNN." *2017 IEEE International Conference on Computer Vision (ICCV).* Venice: IEEE.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. 2015. "Deep Residual Learning for Image Recognition." *CoRR.*

Kanton Zürich. 2022. *Regierung unterbreitet Gegenvorschlag zur Kreislauf-Initiative.* September 25. Accessed November 2022. https://app.statistik.zh.ch/wahlen_abstimmungen/prod/Archive/Poll/1/1/20220925/233030/Informations.

Lili Huang, Jiefeng Peng , Ruimao Zhang , Guanbin Li , Liang Lin. 2018. "Learning deep representations for semantic image parsing: a comprehensive overview." *Front. Comput. Sci.* 840–857.

Lin, TY. et al. 2014. "Microsoft COCO: Common Objects in Context." *Computer Vision – ECCV 2014.* Zurich: Springer, Cham. pp 740–755.

Lin, Tsung-Yi and Dollár, Piotr and Girshick, Ross and He, Kaiming and Hariharan, Bharath and Belongie, Serge. 2016. "Feature Pyramid Networks for Object Detection." *CoRR.*

Liu, Patrick Langechuan. 2020. *Single Stage Instance Segmentation — A Review.* April 29. Accessed November 2022. https://towardsdatascience.com/single-stage-instance-segmentation-a-review-1eeb66e0cc49.

Luo, Lin, Pengpeng Li, and Xuesong Yan. 2021. "Deep Learning-Based Building Extraction from Remote Sensing Images: A Comprehensive Review." *Energies, 14, no. 23* 7982.

Ma, M., Tam, V. W. Y., Le, K. N., & Li, W. 2020. "Challenges in current construction and demolition waste recycling: A China study." *Waste management, 118* 610–625.

Microsoft. 2021. *COCO Dataset.* August 25. Accessed December 2022. https://cocodataset.org/#detection-eval.

Mihajlovic, Ilija. 2019. *Everything You Ever Wanted To Know About Computer Vision.* April 25. Accessed October 2022. https://towardsdatascience.com/everything-you-ever-wanted-to-know-about-computer-vision-heres-a-look-why-it-s-so-awesome-e8a58dfb641e.

Ortega, Marcelo. 2020. *Trainer with Loss on Validation for Detectron2.* March 22. Accessed December 2022. https://gist.github.com/ortegatron/c0dad15e49c2b74de8bb09a5615d9f6b.

Paul Berdahl, Hashem Akbari, Ronnen Levinson, William A. Miller. 2008. "Weathering of roofing materials – An overview." *Construction and Building Materials* 423-433.

Petrosyan, Tigran. 2021. *Introduction to the COCO Dataset.* October 12. Accessed October 2022. https://opencv.org/introduction-to-the-coco-dataset/.

Pokhrel, Sabina. 2020. *Image Data Labelling and Annotation — Everything you need to know.* March 11. Accessed November 2022. https://towardsdatascience.com/image-data-labelling-and-annotation-everything-you-need-to-know-86ede6c684b1.

Raghu, Deepika and Markopoulou, Areti and Marengo, Mathilde and Neri, Iacopo and Chronis, Angelos and De Wolf, Catherine. 2022. "Enabling Component Reuse From Existing Buildings Through Machine Learning - Using Google Street View to Enhance Building Databases." *Association for Computer-Aided Architectural Design Research in Asia (CAADRIA).* Hong Kong.

Rose, Colin M., Stegemann Julia A. 2019. "Characterising existing buildings as material banks (E-BAMB) to enable component reuse." *Proceedings of the Institution of Civil Engineers - Engineering Sustainability, 172* 1-42.

Rosebrock, Adrian. 2016. *Intersection over Union (IoU) for object detection.* November 7. Accessed December 2022. https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/.

SITG. 2022. *Le territoire genevois à la carte.* Accessed October 2022. https://ge.ch/sitg/sitg_catalog/sitg_donnees?keyword=&geodataid=9810&topic=tous&service=tous&datatype=tous&distribution=tous&sort=auto.

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, Kaiming He. 2016. "Aggregated Residual Transformations for Deep Neural Networks." *CoRR.*

Skalski, Piotr. 2019. *Make Sense.* Accessed November 2022. https://github.com/SkalskiP/make-sense.

Swisstopo. 2022. *Swisstopo, Light Base Map.* Accessed October 2022. https://www.swisstopo.admin.ch/de/geodata/maps/smw/smw_lightbase.html.

Trevisiol, Francesca, Alessandro Lambertini, Francesca Franci, and Emanuele Mandanici. 2022. "An Object-Oriented Approach to the Classification of Roofing Materials Using Very High-Resolution Satellite Stereo-Pairs." *Remote Sensing 14, no. 4* 849.

United Nations Environment Programme. 2022. "2022 Global Status Report for Buildings and Construction: Towards a Zero-emission, Efficient and Resilient Buildings and Construction Sector." Environment programme , United Nations, Nairobi.

Uzufly. 2020. *Uzufly.* October. Accessed September 2022. https://uzufly.com/en/.

Yuxin Wu, Alexander Kirillov, Francisco Massa Wan-Yen Lo, Ross Girshick. 2019. *Detectron2.* Accessed December 2022. https://github.com/facebookresearch/detectron2.

Yuxin Wu, Ross Girshick, Patrick Labatut. 2021. *Detectron2 Model Zoo and Baselines.* July 21. Accessed December 2022. https://github.com/facebookresearch/detectron2/blob/main/MODEL_ZOO.md.

Zdziarski, Zbigniew. 2018. *The Reasons Behind the Recent Growth of Computer Vision.* January 28. Accessed October 2022. https://zbigatron.com/the-reasons-behind-the-recent-growth-of-computer-vision/.

7 Appendix

A1. Loss Calculation in Detectron2 Models

The losses calculated in the Detectron2 instance segmentation models are a weighted average of the following losses:

- loss_cls: Classification SoftMax cross-entropy loss in the ROI (Region of Interest) head. It measures the loss for bounding box classification, i.e., how good the model is at labelling a predicted box with the correct class.
- loss_box_reg: Localization L1 loss in the ROI head. It measures the L1 loss for bounding box localisation (predicted location vs true location).
- loss_rpn_cls: Classification log loss in the Region Proposal Network. It measures the "objectness" loss, i.e., how good the RPN is at labelling the anchor boxes as foreground or background.
- loss_rpn_loc: Localization loss in the Region Proposal Network. It measures the loss for localisation of the predicted regions in the RPN.
- loss_mask: Mask binary cross-entropy with logits loss in the Mask head. It Measures how "correct" the predicted binary masks are.

A2. Model Inference Output

The model outputs the following tensors which are relevant for calculating the surface area of each instance mask:

- pred_classes: A one-dimensional tensor containing the classes for N detected instances ranging from 0 to the number of classes
- loss_mask: A three-dimensional tensor with the shape of (N, H, W) where N is the number of detected instances, H is the y coordinate of every pixel in the image and W is the x coordinate of every pixel in the image. The tensors consist of true/false values indicating whether pixel (H, W) is contained within the segmentation mask of N.

A3. Google Colab Notebook

This thesis is submitted together with the relevant Google Colab Notebooks which were coded for this project. The Google Colab Notebook for the Detectron2 model and the training and validation datasets and their corresponding annotations have been uploaded to GitHub.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

ETH Zürich
Institut für Bau- und Infrastrukturmanagement
Stefano-Franscini-Platz 5
8093 Zürich

www.ibi.ethz.ch

© ETH Zürich, January 30