Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Civil and Environmental Engineering

Arne Jakob Eikeng Sterri

Building Boundary Extracting from Pointcloud with a Generative Adversarial Network

Master's thesis in MTING
Supervisor: Hongchao Fan
June 2021

NTNU
Norwegian University of
Science and Technology

Arne Jakob Eikeng Sterri

Building Boundary Extracting from Pointcloud with a Generative Adversarial Network

Master's thesis in MTING
Supervisor: Hongchao Fan
June 2021

Norwegian University of Science and Technology
Faculty of Engineering
Department of Civil and Environmental Engineering

NTNU
Norwegian University of
Science and Technology

NTNU

Kunnskap for en bedre verden

Building Boundary Extracting from Pointcloud with a Generative Adversarial Network

Engineering and ICT

TBA4925 - Geomatics, Master thesis

Submission date: June 11, 2021

Arne Jakob Eikeng Sterri

Supervisor:Prof. Hongchao Fan

Acknowledgements

Big thanks to Hongchao Fan for counselling and Gefei Kong for great discussions. My biggest thanks go to the developers behind Pandas, Seaborn, Geopandas and Shapely, making this research possible. I also want to express gratitude and love to my family and girlfriend. A special thanks to Beanfryd, who always makes me smile, and all members of Fakultetet for a great year.
Knock-knock.

Abstract

This study aimed at extract building boundaries from pointclouds by introducing deep-learning processing by a Generative Adversarial Network (GAN). The research objective was to investigate the extent deep-learning processing solves or improves limitations of established methods that extract boundaries from pointclouds. A total of 903 buildings in Trondheim were manually segmented and served as the dataset for training the network. A statistical analysis found that the network significantly increased Intersection Over Union (IOU) compared to baseline models. Comparing samples before and after processing indicates that GAN increases IOU by more correctly representing concave buildings. From inspecting predicted building boundaries, it is clear that additional geometrical refinements are required to solve the resulting misalignment. More importantly, deep-learning in boundary extraction shows great potential by allowing low-quality input and cheap computational cost.

Sammendrag

I denne masteren har det blitt produsert bygningsomriss fra punktskyer ved å introdusere dyp læring med/av et Generative Adversarial Network (GAN). Forskningsmålet var å undersøke i hvilken grad behandling med dyp læring løser eller forbedrer begrensninger av allerede etablerte metoder rundt ekstrasksjon av bygningsomriss fra punktskyer. Totalt 903 bygninger i Trondheim ble manuelt segmentert og fungerte som datasett for trening av nettverket. En statistisk analyse fant at nettverket produserte en signifikant økning i Snitt over Union (IOU) sammenlignet med baseline-modeller. Sammenligning av bygningsomriss før og etter prosessering indikerer at GAN øker IOU ved å bedre representere konkave bygninger. Ved undersøkelse av de produserte bygningsomrissene er det klart at det kreves ytterligere geometriske forbedringer for å løse den resulterende feiljusteringen. Enda viktigere er at inkluderingen av dyp læring i bygningsomrissekstraksjon viser stort potensial ved å tillate input av lav kvalitet og billige beregningskostnader.

Table of Contents

1 Introduction

The main goal of this thesis is to predict building boundary polygons from pointcloud input.

The task is carried out by improving upon existing methods of building boundary extraction from pointclouds, by introducing deep learning-based processing with a Generative Adversarial Network (GAN).

Put bluntly, knowing how a building is structured and where it is, has great importance in multiple fields. For example, building boundaries is fundamental for many GIS applications and urban planning. Furthermore, as airborne light detection and ranging (LIDAR) data also has a height attribute, it provides the means for 3D building modelling. Examples of other applications are calculations of sunlight duration, disaster management and property-taxes (K. Zhang, Yan, and Chen 2006). Thus, to automatically predict consistent and accurate polygons that represent buildings boundary geometries is of high value.

Traditional approaches using total stations or satellite imagery are either very time-consuming or too inaccurate. Airborne LIDAR data provides high precision at a low cost, but the boundary extraction problem proves difficult due to large amounts of unstructured data.

Existing methods of building boundary extraction from pointclouds have limitations, such as the sizing problem. This revolves around the allowed degree of detail and concavity, and this thesis implements GAN processing to investigate the extent it can improve upon these limitations. An entire pipeline of methods from segmented pointcloud to building boundary polygon is presented. The predicted polygons are compared to the actual buildings by projecting on a map for visual presentation complimented by a statistical analysis of the role the processing with GAN serves. As deep-learning processing can be a black box, two different input images of varying quality were generated.

The main research question naturally becomes if, how, and to what extent the inclusion of a GAN can improve the predicted building polygons from a pointcloud.

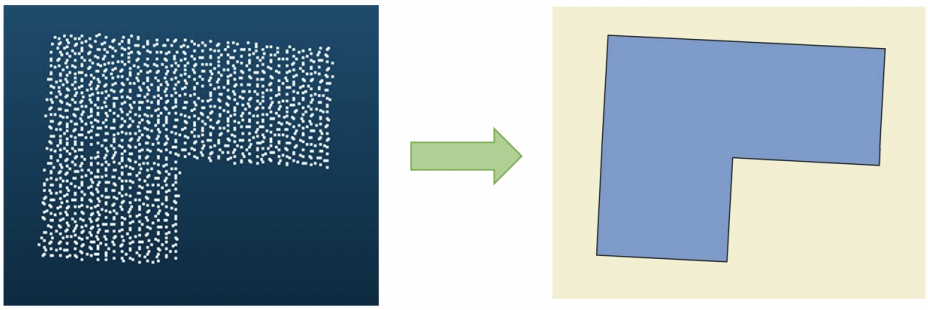


Figure 1.1: Example of extracting building boundary polygon from a pointcloud.

In Figure 1.1, the essence of the problem can be seen. The theoretical side of how difficult it is to solve such a "simple" problem is fascinating. For example, how a young child is able to, without as much as breaking a sweat, draw the lines representing the exterior boundary of a pointcloud. Whereas for a computer to do this requires a whole lot, as shall be seen in the upcoming pages.

2 Theory and Similar work

Many different approaches to identifying building outlines exist, depending on the origin of data, level of user interaction, and preferred method. This section provides an overview of the current methodologies based on the origin of data and the methods' steps. (Awrangjeb 2016) uses LiDAR-data as input and formalizes the steps and difficulties in building boundary extraction with traditional approaches and further proposes possible solutions to these challenges. Supplementary, (Chawda, Aghav, and Udar 2018) present a method for building boundary extraction from satellite images based on a convolutional neural network (CNN). (Su Zhang, Han, and Bogus 2020) combines LiDAR-data and satellite images for extracting building footprints. We start by discussing the case where the data is acquired by airborne LiDAR equipment. The LiDAR-data is structured as a set of points where each point has a list of attributes. The attributes are the points coordinates x-, y- and z- values and other attributes such as intensity and number of returns etc., Depending on the equipment used for data acquisition.

(Awrangjeb 2016) formalizes building boundary extraction from a set of points by three main steps.

1. **Segmentation** - categorizing points into building-points and non-building points.
2. **Trace** identified points to generate building boundary.
3. **Adjust edges** to form regular building footprint.

The first step is, in short, to classify if a point belongs to a building or not. It is possible to do the segmentation step manually or automatically. Manually translates to human labelling the points in applicable software. By automatic means, there are several ways the segmentation can be done. The end goal of this step is to say if a point belongs to a building or not, and multiple ways of achieving this have been introduced. For example, (Sampath and Shan 2007) used a 1D-bi directional filter for separation between ground- and non-ground points before using a clustering-based region-growing method for segmenting points into separate buildings. Similarly, (Ramiya, Nidamanuri, and Krishnan 2017) first separated points into ground- and non-ground points before using a Euclidean distance-based algorithm for segmentation. Another variant is sepa-

rating ground- and non-ground points by using morphological operations and segmenting into individual buildings/roof-planes by using surface-fitting (Mongus, Lukač, and Žalik 2014). Also worth mentioning is (Alharthy and Bethel 2002), where they take advantage of multiple returns in the LiDAR data before applying local statistical analysis based on gradient. In general, these approaches can be grouped by taking advantage of inherent geometric- and topological information such as height, normal vectors of neighbourhoods, etc. Recently, in the era of deep-learning-based approaches, networks that segment and classify 3-dimensional data have also been made (Qi et al. 2017). As was utilized in (Pohle-Fröhlich et al. 2019).

After the segmentation, each point has been labelled as building or non-building, and what proceeds is to generate a building boundary from the categorized points. There are, as in the first step, several ways to do this. (Awrangjeb 2016) divides this step as *direct*- and *indirect* contour extraction, and this thesis will adopt this division as well. The division is based on how the data is processed. In the direct approach, the boundary is found by using mathematical operations directly on the points them-self. Intuitively this step can be thought of as a gift-wrapping problem. The basis of most approaches to this problem is variations of the convex hull algorithm (Jarvis 1977). The convex hull algorithm is, as the name implies, made for convex point sets. It deals poorly with concave outlines, such as Cross-shaped buildings (Edelsbrunner and Mücke 1994).

Most predominantly used is the variation called α-shape. (Dorninger and Pfeifer 2008) defines the α-shape of a set of points as its polygonal boundary, where $\alpha = 0$ yields the convex hull. Thus the main challenge in using the α-shape is to determine the size of α such that the resulting polygon reflects the finer details of the point-set boundary. Especially challenging is the case of uneven point-cloud density distributions (Sampath and Shan 2007). This is made apparent by observing Figure,2.1 and the size of α should be based on the point distribution.(Dorninger and Pfeifer 2008) set the value of α to be twice the mean distance between two points. It should be noted that it is possible to use "brute force" to decide α. This is done by starting with a big circle and iterative reduce the radius until points are left out from the extracted boundary.

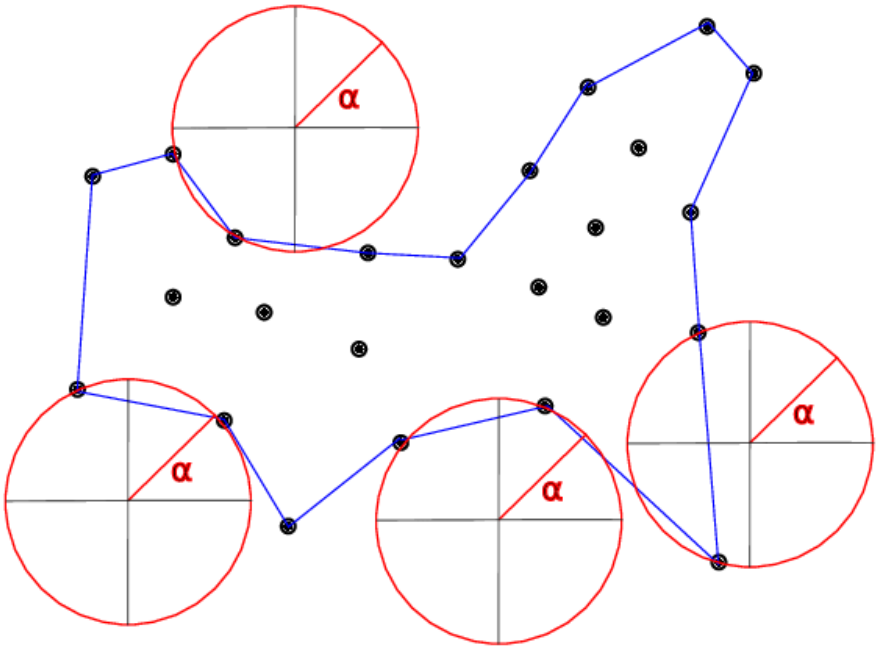


Figure 2.1: Visualisation of the α-**shape** method. The red circle "rolls" around the point set and thus allows for a degree of concavity. Given that α determines the radius, it also determines the allowed concavity. Other geometrical structures, such as squares, can also be used. Figure from (Eich, Dabrowska, and Kirchner 2021).

As the red circle is "rolling" around the points, the order it hits each point is traced, yielding a polygon representation of the exterior boundary of the point-set. Another direct approach is proposed by (Awrangjeb 2016) which also deals with multiple objects inside a point cloud. It is an algorithm that incrementally removes edges longer than a set threshold in the boundary of the Delaunay triangulation.

The *indirect* approach simplifies the problem by transforming the 3D-vector data to a 2D-raster representation before extracting the boundary (Awrangjeb 2016).

Both the *direct-* and *indirect approach*es suffer from a very similar sizing problem. In the direct approach, the main challenge is determining the size of (and the shape of) α. And for the *indirect* approach, the sizing problem becomes determining suitable pixel size. Choosing a fitting grid size for constructing the image depends on desired image characteristics. The most straightforward approach is simply projecting all points down into the x-y- plane based on the dimensions and possibly the density. Following is deciding

the pixels value, where two main approaches exist. First, one can use a binary mask, meaning if a point falls inside a pixel, set the pixel's value to 1. The other approach is constructing an intensity-based image based on how many points fall in each pixel. This transforms data from vector representation to raster, and thus some accuracy is lost in the process (Congalton 1997). Note that each pixel is georeferenced given the point coordinate attribute. After constructing an image representation of the point set, the next step is extracting the boundary pixels. One way to do this is to implement a region growing algorithm or use an edge detector such as the Canny edge detector (Canny 1986).

At this point, both the *indirect methods* have yielded a point representation of the boundary, and what follows is to connect the points. There are several ways to do this. For example, one could start by choosing a random boundary pixel and connecting it to its closest neighbour and then connecting that to its neighbour and so on until all points have been visited. Interestingly, there are many similarities with this problem and the problem of extracting a pointcloud boundary, but now as a 2-dimensional problem with boundary pixels already extracted. (Suzuki and Abe 1985) introduced two border-following algorithms for topological analysis.

After a polygon represents the pointcloud boundary, the next step is **adjusting edges** of the polygon. Before discussing this step in detail, getting to this stage from another type of data input should be briefly touched upon.

An alternative and more easily acquired form for data input is aerial images or satellite images. (Shiqing Wei, Ji, and M. Lu 2020) developed a CNN that segments each pixel to either building- or non-building. After this step, the next required steps of attaining a 2D-polygon representation of the building boundary are similar to the *in-direct* approach. Using an edge detector separates the boundary pixels. Next is connecting the boundary pixels in "correct" order, which yields a polygon representation. (Shiqing Wei, Ji, and M. Lu 2020) used a version of the Marching Cubes algorithm (Lorensen and Cline 1987) for boundary extraction. For a deeper description, the reader is encouraged to read (Shiqing Wei, Ji, and M. Lu 2020). Note that a big disadvantage with data originated from satellite images is the lack of coordinate precision.

Independent of the type of input data and chosen steps, all methods mentioned above yield a 2D-polygonal representation of each building boundary. And all of them now face the third step, namely **edge-adjusting** or, in other words, polygon regularization.

The initial building boundary polygon will, in most cases, not match the building footprint (Awrangjeb 2016). This is due to uncertainty, irregularities, and noise in the point cloud or image. (Chawda, Aghav, and Udar 2018) mentions that polygons often are jagged, and a post-process polygon refinement is necessary to achieve building footprint. Current methods vary in complexity and level of accuracy. (Chawda, Aghav, and Udar 2018) simply implements Douglas-Peucker (Douglas and Peucker 1973) for refining the jagged lines and obtains decent results. (Teh and Chin 1989) introduced an analysis for finding dominant points on lines. Another and more detailed approach is taken by (Shiqing Wei, Ji, and M. Lu 2020) where the first of three steps is the Deuglas-Peucker algorithm. The next steps are categorized as coarse- and fine- adjustment of edges and corners based on calculating one or two principal directions of a building. The idea is that building footprints follow certain rules such as main direction, perpendicular or parallel lines, and threshold values for the area. (Eich, Dabrowska, and Kirchner 2021) uses a Hugh transformation to achieve rectangular shapes from the calculated α shape. (Shen Wei 2008) used an improved Pipe-algorithm for simplifying the jagged edges before implementing a Circumcircle Regularization Algorithm to force irregular shapes to become rectangles. All in all, many methods of simplifying lines that make up the polygon exist, and for an overview and comparison of the methods, the reader is referenced to (Shi and Cheung 2006).

The polygon simplification steps are based on geometrical assumptions such as, i.e. a building can only have two main directions, and all edges must be parallel or perpendicular to these, or all angels must be larger than a given threshold. Whichever geometrical rules imposed, all are included to remove misrepresentations or noise. These are produced either by method of acquisition or imposed by boundary extraction method, i.e. how the $\alpha-$shape does not perfectly describe the pointclouds boundary.

This thesis aims at improving the building boundary extraction by introducing deep-learning with a Generative adversarial network. To understand how and why a GAN

could be used for a task like this, some theoretical foundation of the underlying principles is necessary. (Goodfellow 2017) provides a systematic overview of the principles behind as well as use-case examples. The main idea of GAN is illustrated in Figure 2.2. The network architecture is designed such that the Generator and Discriminator are adversaries. Based on whether or not the discriminator is correct, the models' weights are updated through back-propagation. If it correctly distinguishes fake from real, the Generator network is heavily penalized and vice versa in the other case. Put it differently, the generator network attempts to learn the transformation from input-image to "real-image". This thesis aims to use this use by setting a coarse image representation of the pointcloud as input, representing a building pointcloud. The "real" image is made by transforming the ground truth polygon into an image. Thus, the network attempts to learn the transformation from an image made initially from pointcloud to an image that correctly represents the building boundary. The following step is to transform the generated image back to polygon form.

This thesis uses the network architecture named *Pix2Pix*, which was introduced in (Isola et al. 2016). It is important to note that *Pix2Pix* is architecturally based on that input and output differ in appearance but has a similar underlying structure (Isola et al. 2016).

Both the generator- and discriminator networks are CNN's - convolutional neural networks, and this fact is what the network feasible for the presented task. One could think that the generator network matters in this thesis, given it needs to learn the desired transformation. But this is not the full picture, as the structure of the discriminator network serves a crucial role. In *Pix2Pix*, the discriminator network is called PatchGAN. Instead of analyzing pixel for pixel (how the generator network operates), it looks at each image's sections (or patches) when classifying real and fake. As the author puts it, ... *PatchGAN can therefore be understood as a form of texture/style loss (Isola et al. 2016).* This fact indicates a promising potential for extracting building boundary. Also worth mentioning regarding the network architecture is its skip connections. These connections are present so that low-level information can flow from input directly to output (Isola et al. 2016).

Lastly, methods measuring the similarities between polygons should be mentioned. (Zeng, J. Wang, and Lehrbass 2013) introduced a complete system for evaluating extracted building boundaries. The system is divided into three categories. The first evaluates the accuracy of correctly predicted building footprint, which is irrelevant for this thesis as manual segmentation is performed. Secondly, a system for evaluating the shape similarity between the predicted building footprint and ground truth. This step is topological grounded. Thirdly, a metric for positional accuracy is proposed by evaluating distances between certain points to the polygon's centre. Another more simple polygon similarity measure is taking the intersection of the polygons in question divided by their union (IOU - intersection over union) and is an area-based similarity measure, not explicitly enforcing any form of topological relationship.

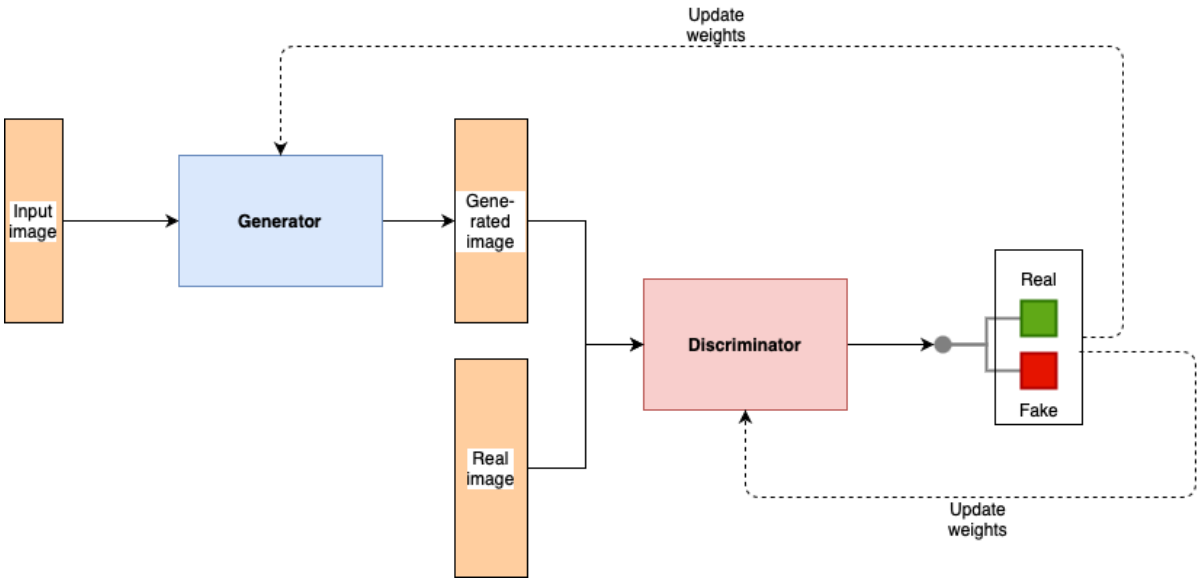


Figure 2.2: The principle of a Generative adversarial network (GAN) is illustrated. Note that GAN is not limited to images as inputs.

3 Establishing data

3.1 Segmentation

The segmentation task for this thesis was done manually. The starting point was roughly 1000 separate las files, representing an area of about 1km^2 acquired by airborne laser scanning done in 2018. The end goal was approximately 1000 different buildings. A complete overview of the files bounding boxes is presented in Figure 3.1.

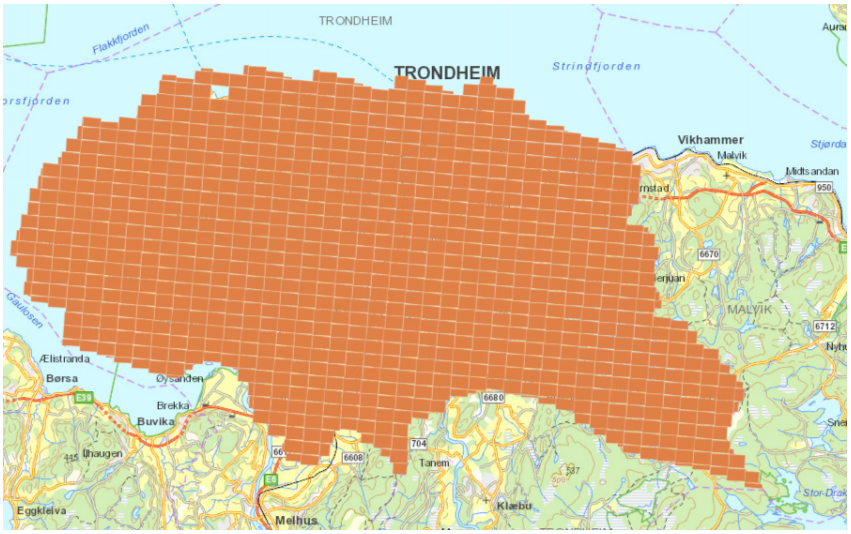


Figure 3.1: Overview of laser-scan coverage over Trondheim. This pointcloud was acquired by airborne LiDAR using a drone in 2018.

The dataset was also to be used in another master thesis, and the segmentation process was, therefore, a collaboration between the author of this thesis alongside two others. As previously mentioned in the theory section, the goal of this step was to separate points belonging to a roof and points that do not lie on roofs. This thesis, and the collaborator's thesis, intended to use the segmented data-set for machine learning tasks regarding buildings and roof-planes. Therefore a well-distributed data-set was set as a goal. Balancing between time consumption, quality of each segmentation and appropriate sizing (number of buildings), a target goal was around 1000 separate buildings. The number was set based on the assumption it would deem a sufficiently large sample size, allowing the deep-learning network to learn aspects of each category. The pointclouds were categorised by the building categories presented in Figure 3.2, as introduced in

(Kada and Mckinley 2009) with the inclusion of a ninth category for combinations of the former. Each separate building segmentation was proven rather time-consuming as some building categories proved quite challenging to find. The ground truth polygon (*FKB-Bygning - Kartkatalogen* 2021) had to be checked before starting segmenting. This was because some building boundaries sometimes included terraces or porches, etc.

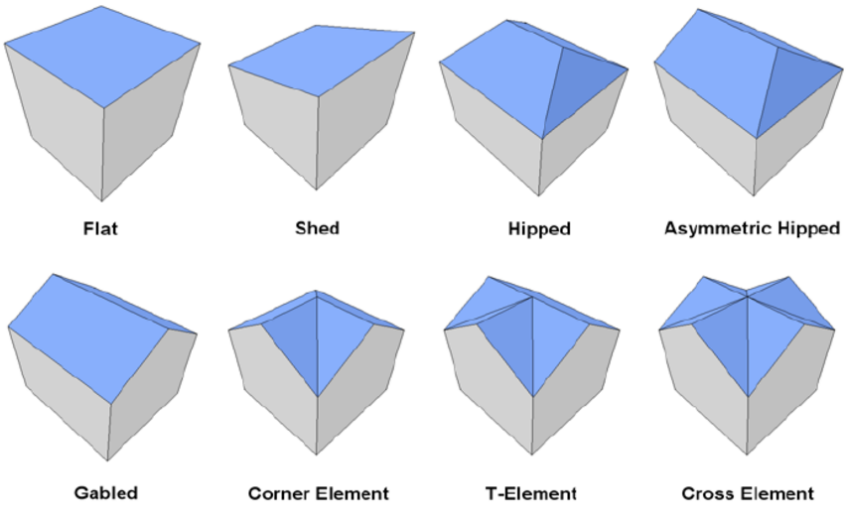


Figure 3.2: The different building categories presented in the pointclouds, introduced by (Kada and Mckinley 2009).

Due to the overrepresentation of *Flat-*,*Hipped-* and *Gabled* buildings, the target number of these buildings was easily acquired. After reaching around 200 of these roofs, the segmentation task solely focused on buildings outside these categories. This task ended up being more time-consuming than initially thought, and as a result, some uneven distribution between types was accepted. A note further justifying this is the distribution in the dataset relates better to the real world (at least in Trondheim) as the distribution between building categories is not uniform. The software used for this task is called CloudCompare and is an open-source program *CloudCompare - Open Source project* 2021. Each pointcloud was segmented by looking at the pointcloud from various angles and manually selecting points laying on the roof.

Each roof was further segmented into its respective planes due to the machine learning task of the other thesis. For instance, the classical gable roof (category 5) is broken down into two different planes. The planes are categorized numerically by geometrical properties like shape and size. A more complex roof structure as T-element and Corner Element roof will consist of 4 different roof planes.

Regarding the distribution between categories, it is important to have sufficient samples of each different building type so that the network can learn the characteristics of each building category. Finally, as the segmentation was done manually, a subjective decision of how much occlusion to allow and the degree of accepted noise and disturbances in the pointcloud was reached.

The dataset was organized by their respective building ID's used by Kartverket *FKB-Bygning - Kartkatalogen* 2021. This allowed for a one-to-one correspondence to other available data relating to the building, and most importantly, the building's footprint. Further, each building was associated with the attribute *Shared-property*. This was done as some building-footprint polygons are divided, which can be observed in Figure 3.3. (For an in-depth description of the FKB-building dataset, the reader is referred to *FKB-Bygning - Kartkatalogen* 2021.) Finally, an overview of the complete data organization in the segmentation step is given in Figure 3.4.

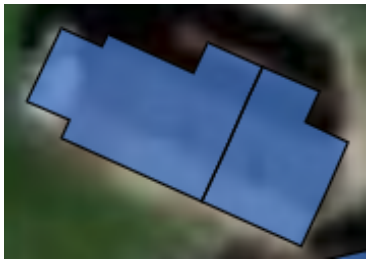


Figure 3.3: Example where footprint of same building is divided in two parts.

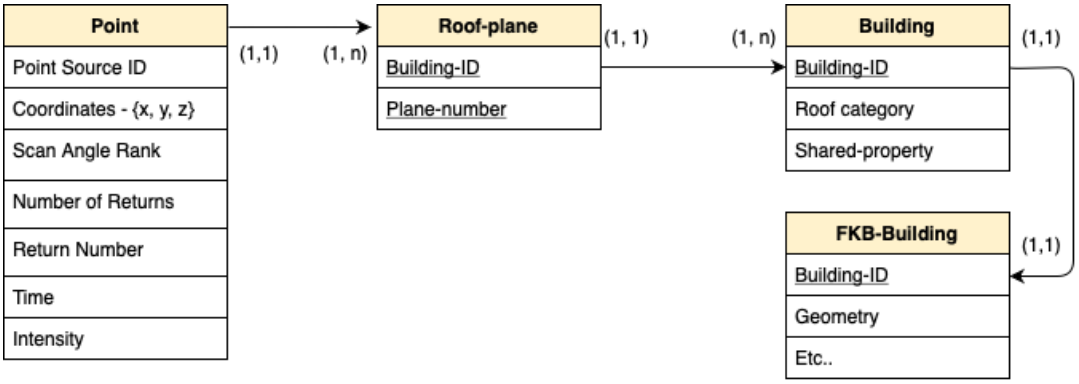


Figure 3.4: Overview of the data-organization. One point must belong to a roof-plane, which belongs to a building. A combination of Building-id and Plane-number is used to uniquely define a roof-plane. The geometry attribute in FKB-building represents the polygon of a building footprint.

3.2 The dataset

This section provides a foundation for the dataset by presenting simple statistics. Firstly the distribution of roof types is shown in Figure 3.5. The total number of segmented roofs is 903, after a data-cleaning step. This process was done in plenary with the other collaborators. Each segmented roof and the ground truth polygon from (*FKB-Bygning - Kartkatalogen* 2021) was visually inspected and discussed. This led to many roofs being re-categorized and some removed due to errors. This step was crucial in agreeing on standards for categorizing roofs which arguably could fit several categories and resulted in a higher degree of objectivity. By observing Figure 3.5, it is clear that the distribution between categories is uneven. For example, only 14 *shed-* and *cross-roofs* were segmented, and only 38 corner-roofs. This uneven distribution results from the few roof types of these categories present in the initial pointcloud and therefore resemble the distribution of roof types in Trondheim. As a result of the uneven distribution, it should be noted that machine-learning algorithms may have difficulties with these categories, given the low population of training samples.

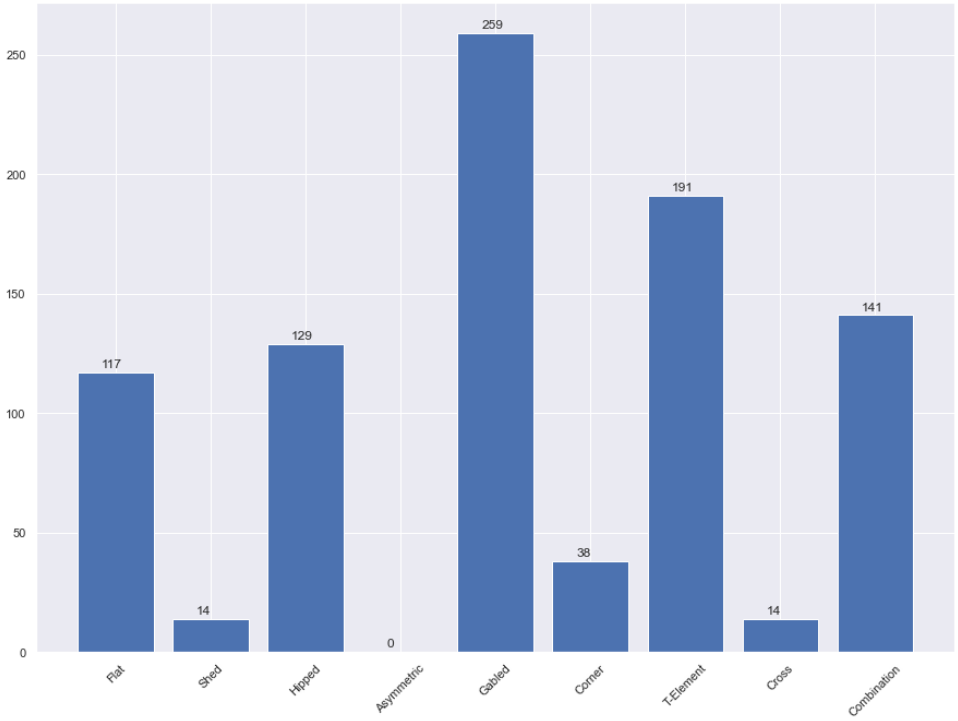


Figure 3.5: Distribution of roof-type categories in the dataset. A total of 903 buildings point-clouds were segmented.

As this thesis focuses on buildings as a whole and not on each separate roof-plane, only the statistics of the combined roof-planes are relevant. To better understand the dataset, researching how many points per roof make up each roof presents an overview and valuable statistic. This is visualized in Figure 3.6.

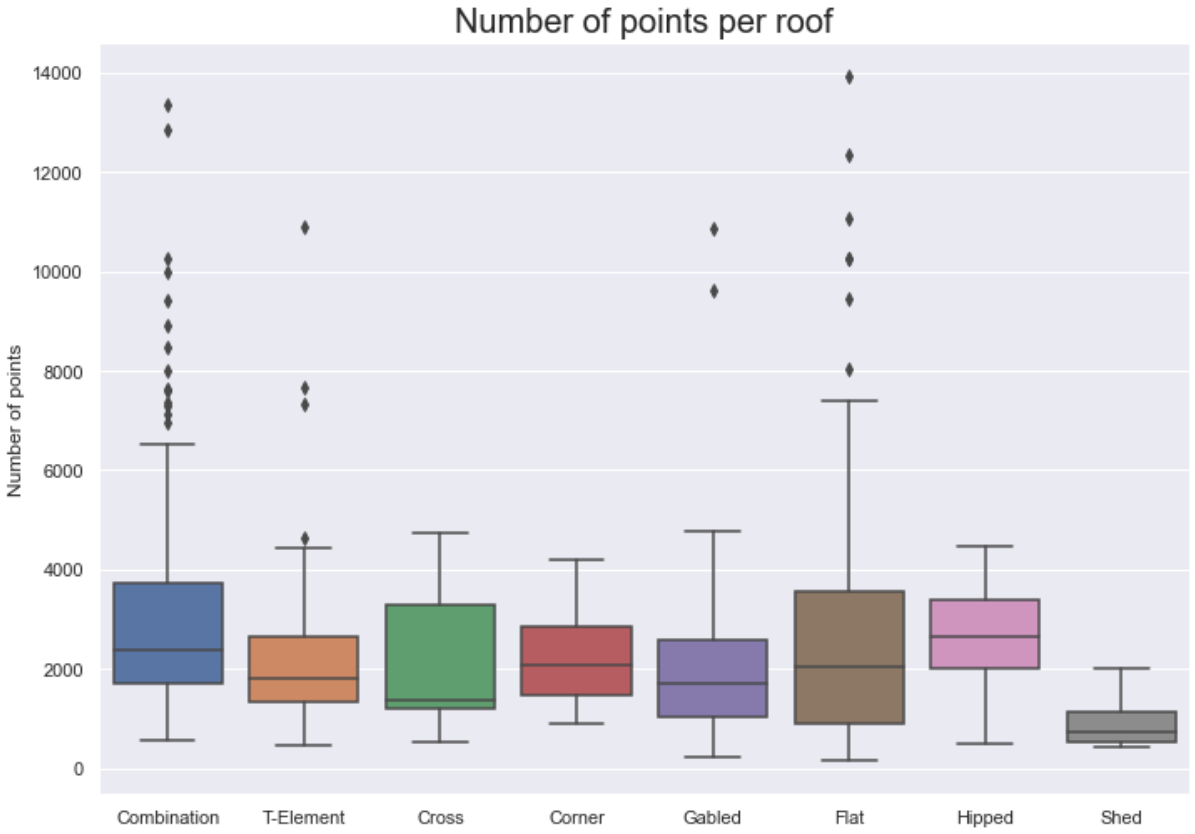


Figure 3.6: The number of points per building for each category present in the dataset. Note that two extreme outliers were removed due to compressing the plot (both in Flat-category).

Important takeaways from Figure 3.6 are how most building-categories, besides shed-category, have a median of around 2000 points. Their 25%− and 75% quartiles (represented by box outlines) are also relatively similar. These takeaways imply that relatively the same amount of information is stored on each roof and checks that everything is normal. If, i.e. some roof categories were very unevenly distributed to others, this would be important to know, and appropriate action would have to be taken. Note that the shed category only consisted of 14 samples and the box-plot column.

Simply looking at the number of points of each segmented roof can be misleading as it says nothing about how the points that make up a building are distributed. For this reason, Figure 3.7 describes the data-set density distribution by the different building categories.

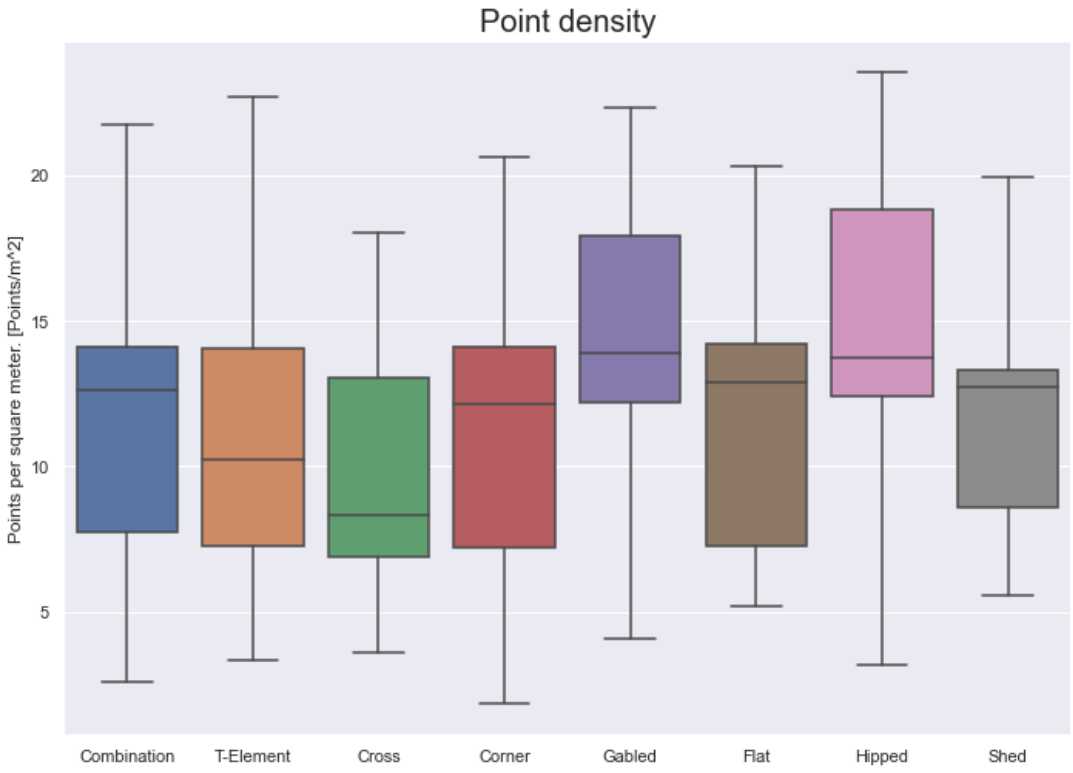


Figure 3.7: Box plot of the point-density for each building category in the dataset.

Note that Figure 3.7 only provides an estimation of the density since the calculated area for each roof is done with a convex-hull algorithm (*The Shapely User Manual — Shapely 1.7.1 documentation* 2021). This results in an estimated value of the actual area and serves as a good foundation for the density distribution. The density of the segmented roofs is important in both *direct-* and *indirect* contour-extraction step, as introduced in Section 2. The critical takeaway from Figure 3.7 is again that values for each category are relatively similar. For example, the median values vary between approximately 7 for *cross-roofs* and 14 for *gabled roofs*. An overview of segmented buildings distributed across Trondheim is shown in Figure 3.8.

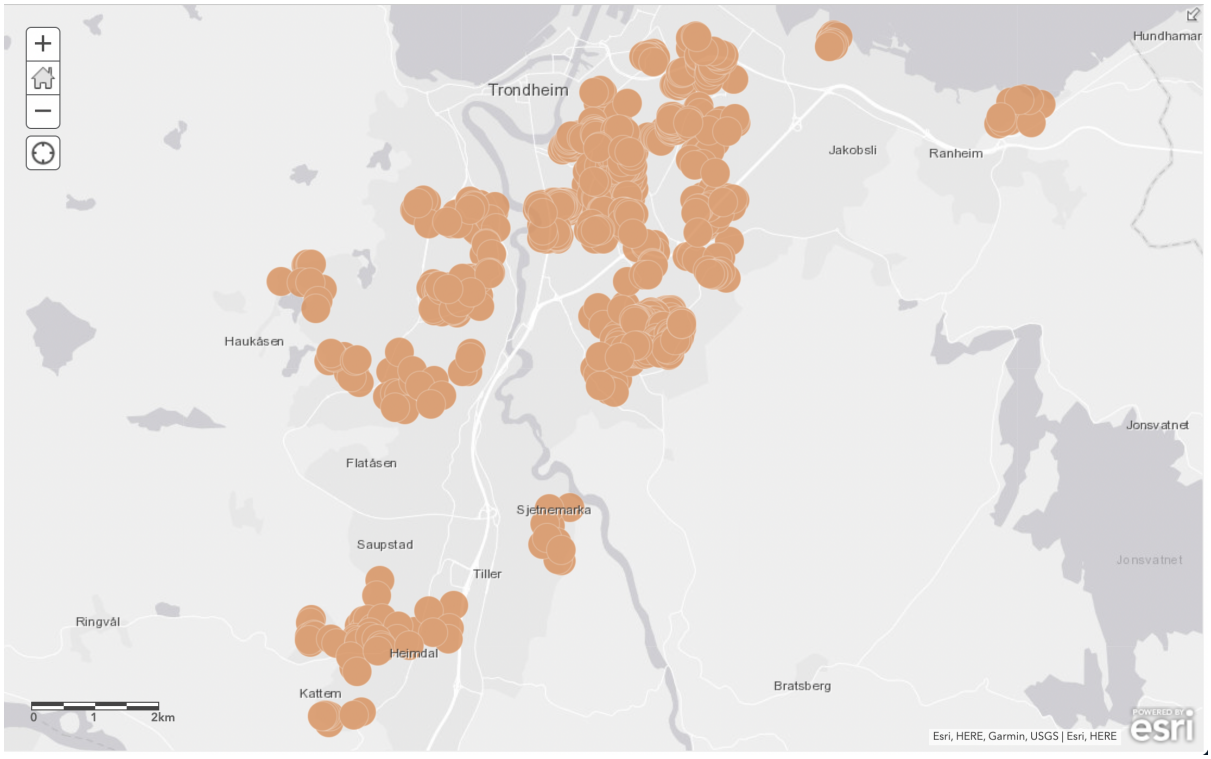


Figure 3.8: Overview of segmented buildings distributed across Trondheim. Each orange circle represent a segmented building pointcloud

It is important to "hold" out some samples to be used for testing in deep-learning experiments. The reasoning behind this is that the network has to predict based on samples that it has not seen before. The validation set is in this thesis used to form the basis for selecting the different epoch models. A visual snippet of the split is presented on a map in Figure 3.9. Dataset is split with ratio $[0.7, 0.2, 0.1]$ into to categories [Train, Validation, Test]. This yields: $Train - 630$ samples, $Validation - 180$ samples, and $Test - 93$ samples.

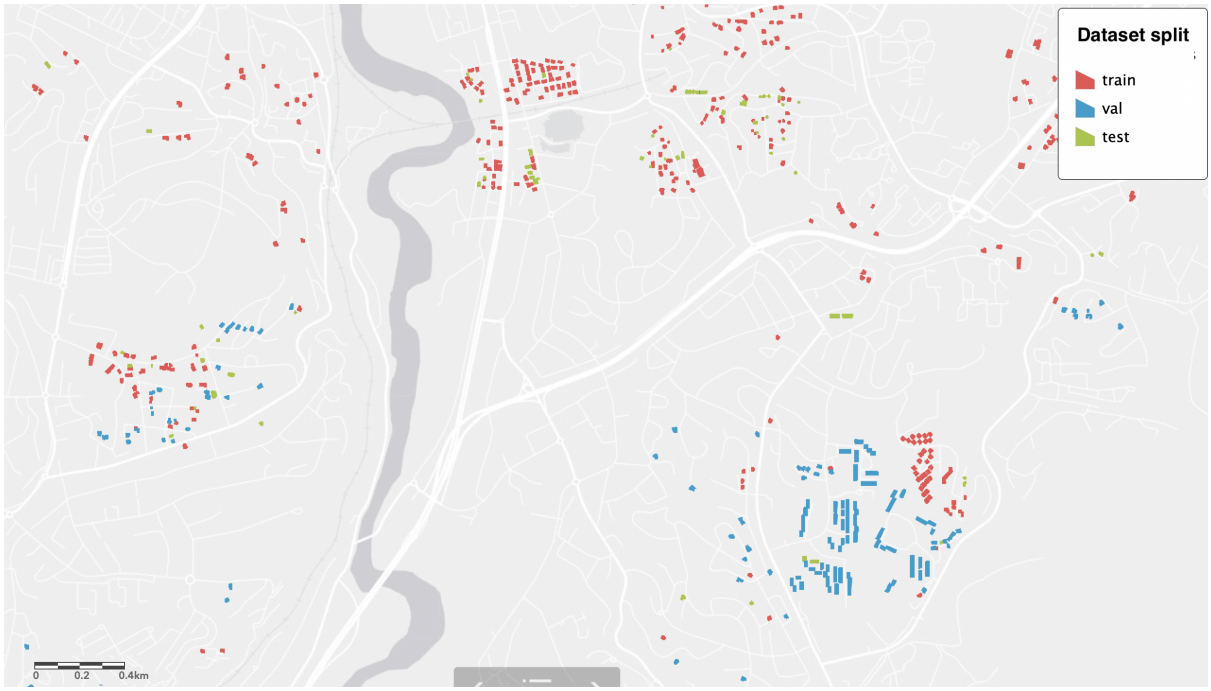


Figure 3.9: A map showing the segmented buildings in their respective splits of train-, test- and validation set.

4 Method

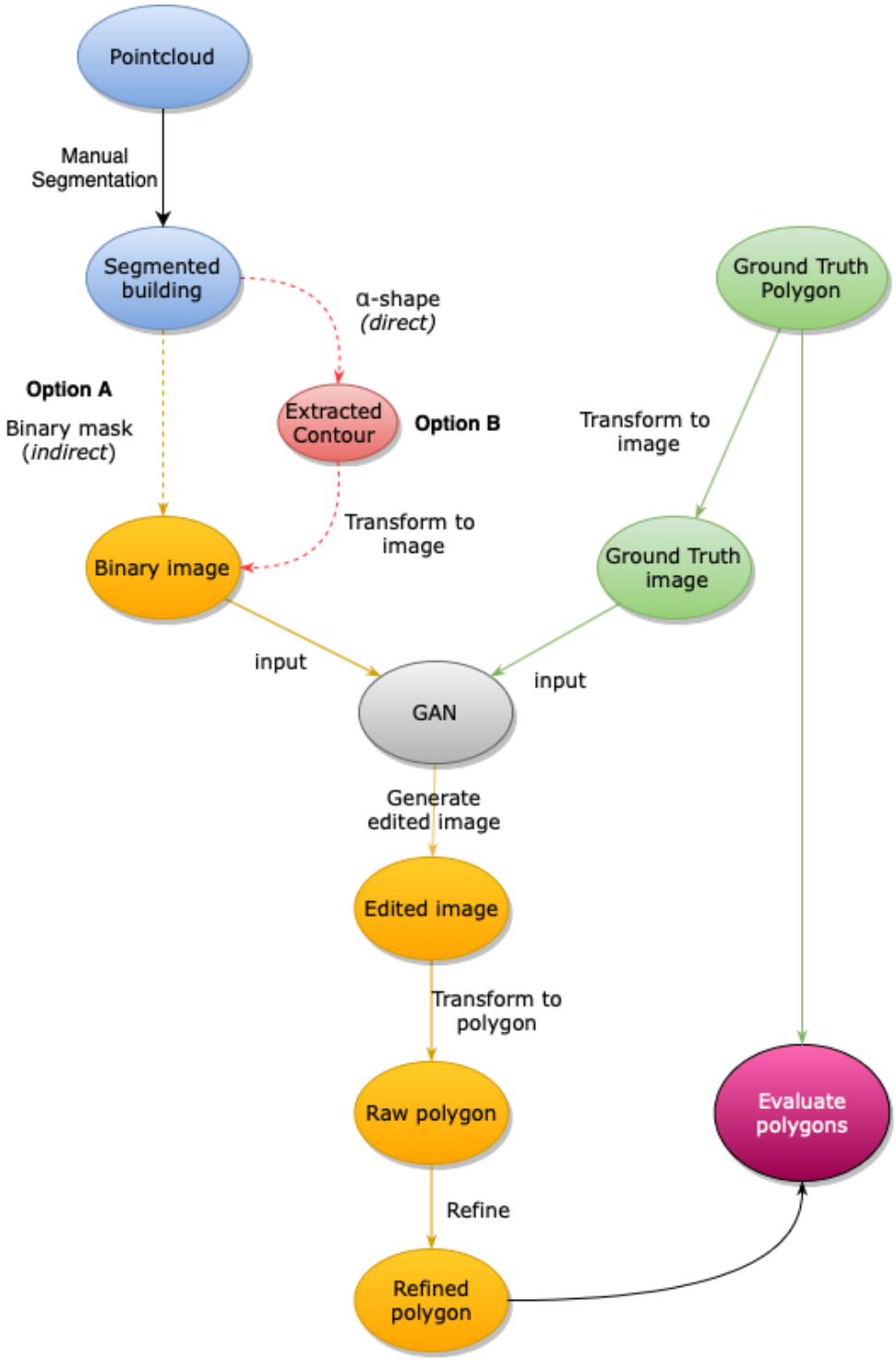


Figure 4.1: The pipeline showing the data flow of the whole method. Starting from Pointcloud and the entire path taken to the end, resulting in the refined polygon.

As Figure 4.1 breaks down the whole data flow into separate steps, the method will be presented chronologically with the flow direction. Since the manual segmentation step already was explained in Section 3.1, the starting point of this section will be the segmented building pointcloud. For simplicity, the images generated from the *indirect approach* will be named the binary images, and from the *direct approach* will be named alpha-shape images.

4.1 Indirect approach

The *indirect approach* is fairly straightforward and has the intent of transforming the pointcloud into an image representation. Algorithm 1 explains the steps in detail and is exemplified in Figure 4.2.

Algorithm 1 determines grid size based on the max distance of either x- or y-direction. In practice, this means an image representing a small roof will have pixels that translate to a smaller area than the case where a larger roof is represented. This method is utilized due to the relative homogeneous point distribution of the different roofs, as discussed in Section 3.2. Further, as the max distance between two points determines the grid size in both directions, it forces each image to be square.

Note that Algorithm 1 faces the same sizing problem as discussed in Section 2. This problem surfaces as determining a fitting grid size, which is the resolution of the image. If set too high, the resulting image would have holes inside the roofs due to not high enough point density. Contrary, if set too low, the resulting image would be too coarse. After experimenting with different values of initial *resolution*, a mid-point between the above-mentioned challenges was found to be 16×16.

Algorithm 1: *Indirect* binary image generation

input : Segmented roof pointset **S** & initial resolution **res**

output: 256×256 binary image representation of the segmented roof

Init - Create matrix, ***img****, of zeroes with dimension (res,res)*
Project points to $x - y$ plane by removing $z-$ dimension.
Find bounding box of **S**: *xmax, xmin, ymin, ymax*
Let xdiff = xmax − xmin and ydiff = ymax − ymin
if *xdiff* > *ydiff* **then**
    xgrid = linearly spaced array from xmin to xmax with **res** number of steps
    ygrid = linearly spaced array from ymin to ymin+xdiff with **res** number of steps
else
    ygrid = linearly spaced array from ymin to ymax with **res** number of steps
    xgrid = linearly spaced array from xmin to xmin+ydiff with **res** number of steps
end
for *point* $\boldsymbol{p} \in S$ **do**
    Match coordinates of **p** with xgrid and ygrid. matching indexes = *(i,j)*
    $img[i, j] = 1$
end
Upsample image to 256×256
Return **img**

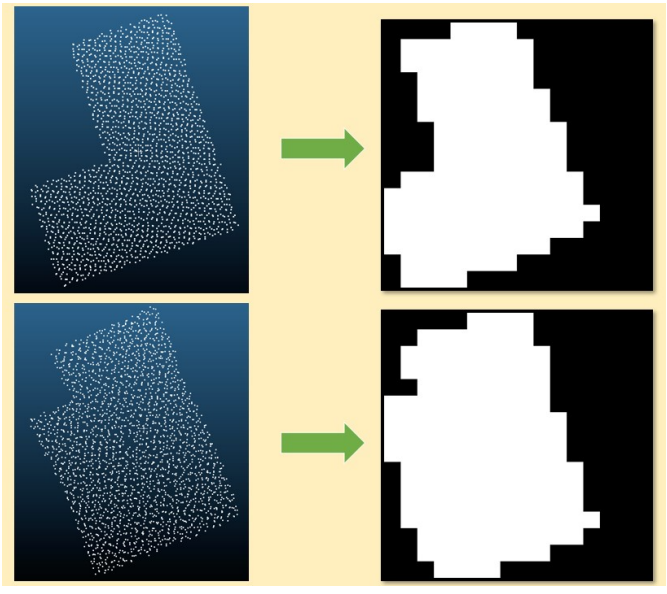


Figure 4.2: Result of *indirect approach*. Two examples visualizing how Algorithm 1 transforms pointcloud to image.

4.2 Direct approach

The *direct* approach works on the points directly but requires some processing before generating the representing image. $\alpha-$Shape was introduced as the main method for *direct* contour extraction, and this thesis will adopt the implementation. Algorithm 2 goes through the steps taken, complemented by a visual example in Figure 4.3.

Algorithm 2: *Direct* binary image generation

input : Segmented roof pointset **S**

output: 256×256 binary image representation of the segmented roof

Init - Create matrix, ***img****, of zeroes with dimension (256,256)*
Project points to $x - y$ plane by removing $z-$ dimension.
Calculate α value based on density
Create $\alpha-$shape polygon, **P**
Find bounding box of **S**: *xmax, xmin, ymin, ymax*
Let xdiff = xmax − xmin and ydiff = ymax − ymin
if *xdiff* > *ydiff* **then**
    xgrid = linearly spaced array from xmin to xmax with 256 number of steps
    ygrid = linearly spaced array from ymin to ymin+xdiff with 256 steps
else
    ygrid = linearly spaced array from ymin to ymax with 256 steps
    xgrid = linearly spaced array from xmin to xmin+ydiff with 256 of steps
end
for $pixel(i, j) \in img$ **do**
    if $(xgrid[i], ygrid[j])$ inside **P then**
    $img[i, j] = 1$
end
Return **img**

At this point, both the *indirect-* and *direct* approach have yielded binary image (256×256) ready for feeding into the GAN.

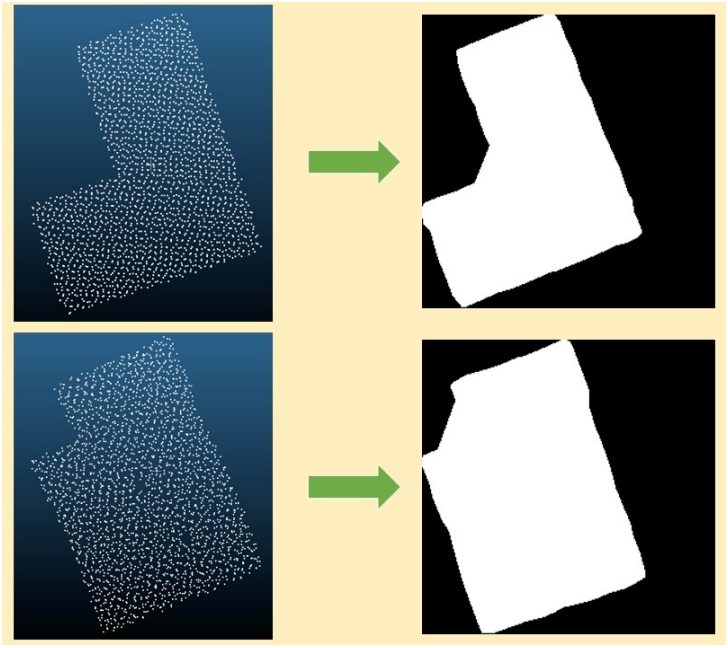


Figure 4.3: Result of *direct approach*. Two examples visualizing how Algorithm 2 transforms pointcloud to image.

4.3 Generative adversarial network

The task of GAN is to transform the images produced by either the *indirect* or *direct* approach into new images that more correctly represent the building boundary. Figure 4.4 shows how images are concatenated for input for the generator network.

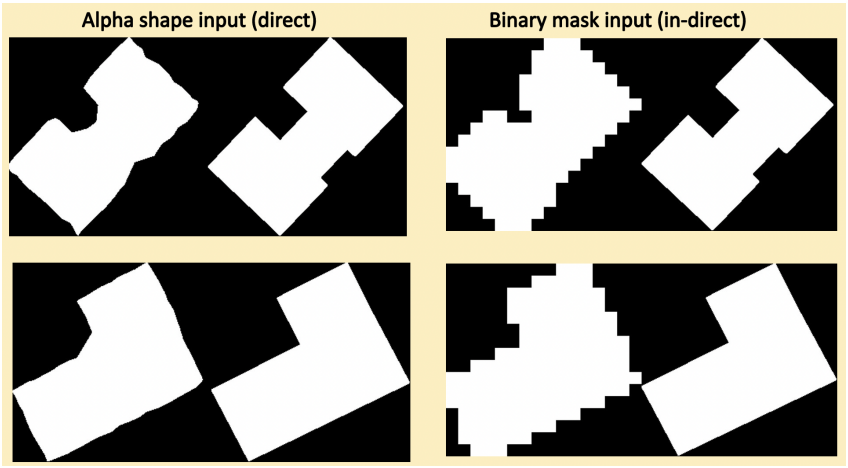


Figure 4.4: Example of input to the generator network. Image is concatenated with the ground truth image.

As previously stated, the name of the framework is *Pix2Pix*. The full architecture of the generator- and discriminator network can be seen in Appendix, Figure A.1 and A.2. The input for the generator network can be seen in Figure 4.4. The dataset is split into *training-*, *validation-* and *test* set with ratio 0.75, 0.15, 0.10. For increasing the number of samples, image augmentation is done by rotating images in fixed directions (both ground truth image and input image) determined by a random probability.

The generator loss is defined by Equations 1, 2 and 3:

$$L_1 = MeanAbsoluteError(G.T - GenOutput) \tag{1}$$

$$GenLoss = BinaryCrossEntropy([[1, 1..]], DiscOutput) \tag{2}$$

$$TotalGenLoss = GenLoss + (\lambda * L_1) \tag{3}$$

In Equation 3 λ is set to 100, as the authors of (Isola et al. 2016) suggests. Further, in Equation 1 G.T is the Ground Truth image.

As for the discriminator the loss is defined as by:

$$real\_loss = BinaryCrossEntropy([[1, 1..]], RealImg) \tag{4}$$

$$generated\_loss = BinaryCrossEntropy([[0, 0, ..]], GeneratedImg) \tag{5}$$

$$DiscLoss = real\_loss + generated\_loss \tag{6}$$

The Discriminator and Generator are adversaries by means that Equation 4 is fighting Equation 2. The network was trained for a total of 150 epochs, testing on the validation set every 25 epoch and saving the current weights. The learning rate is initially set to 0.0002, and Adam is used for optimization. In Figure 4.5 a selected sample output from the generator is shown. The models were trained on a Tesla P100-PCIE-16GB GPU, by using Google Colab PRO (Bisong 2019). The training phase (150 epochs) for each input type took $\sim$ 75 minutes.

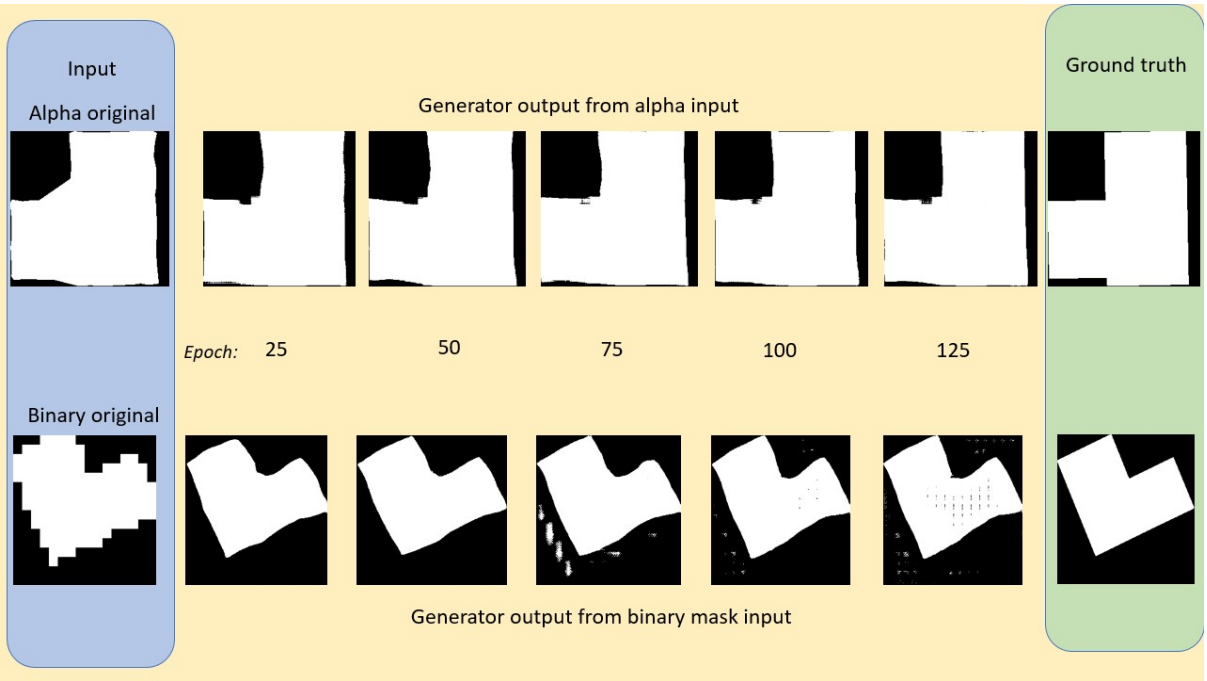


Figure 4.5: Selected sample of output from the generator each 25 epoch. (The generator network start producing strange patters after approximately 100 epochs.)

4.4 Image to polygon

After the network has been trained and generated images, the next step transforms the generated images back to polygon form. This step consists of several operations due to the strange patterns occurring in some of the generated images. The full procedure is described in Algorithm 3. This process is transforming raster- to vector data.

Algorithm 3: Image to polygon

Assumption: the building has only one contour without holes.

input : Generated image, respective building-ID **bid**

output: Raw polygon with georeferenced coordinates

Init: k_size = 5, Define **Kernel**(k_size,k_size), get xgrid & ygrid with *bid*
Add constant padding borders
Morphological closing with square Kernel
Find all connected closed areas, save pixels in the largest region with largest area
Fill all areas expect larges area
Extract boundary pixels with Canny edge detection (Canny 1986)
Find contours with an implemented version of Algorithm proposed in (Suzuki and Abe 1985)
while *number of extracted contours > 1* **do**
    k_size += 1
    Preform morphologic closing on canny img with Kernel(k_size,k_size)
    Extract contour(s)
    if *k_size*/2 >= *padding* **then**
        Return None
    end
end
for *Pixel*$(i, j) \in$ *contour* **do**
    Append geo−coordinates from xgrid and ygrid to a list
end
Create polygon from list
Return polygon

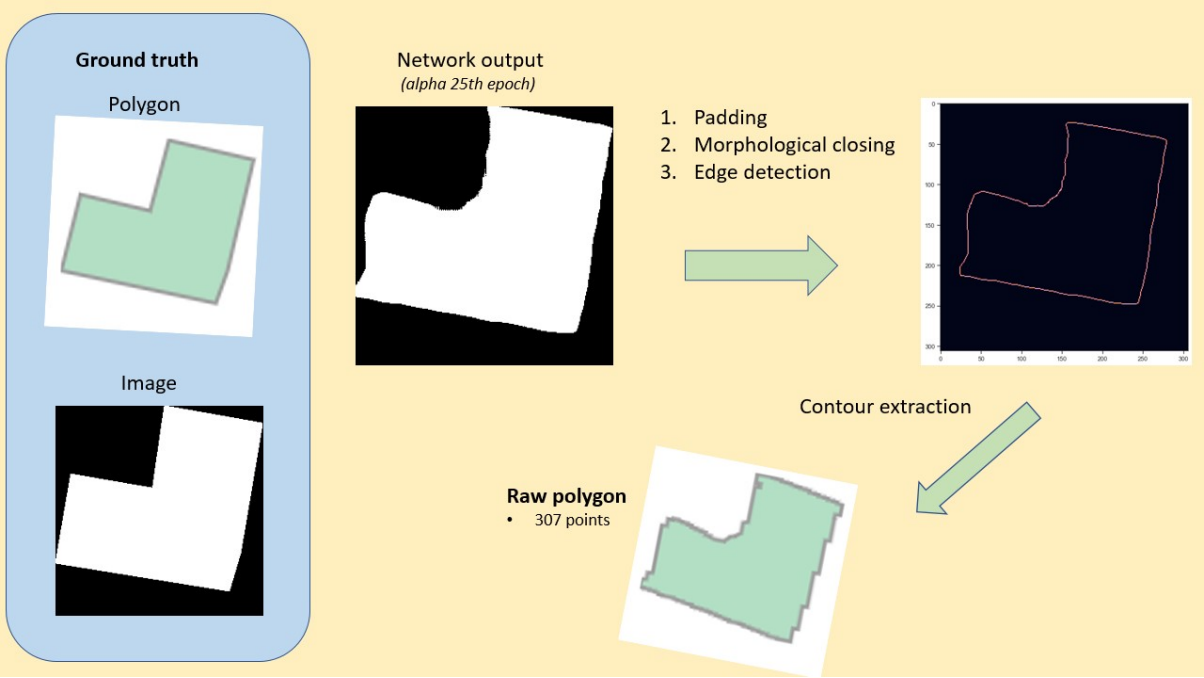


Figure 4.6: Selected sample of output from the generator each 25 epoch.

4.5 Polygon refinement

A raw polygon has been created from the generated images, and the next step is to refine it. An illustration of this is shown in Figure 4.7. But before that is possible, each polygon has to be checked for validity. Due to the complex structure of the generated images, some polygons have self-intersecting edges. Algorithm 4 shows how this is dealt with.

Algorithm 4: Validate and simply raw polygon

Assumption: A building polygon can not intersect itself.

input : Raw polygon

output: Refined and simplified polygon

```
buffer_size = 0
while polygon not valid do
    buffer_poly = polgyon.buffer(buffer_size)
    polygon = exterior boundary coordinates of buffer_poly
    buffer_size =+ 0.05
    if buffer_size >= threshold then
        Return None
    end
end
Simplify polygon with Douglas-Peucker algorithm
Return simplified polygon
```

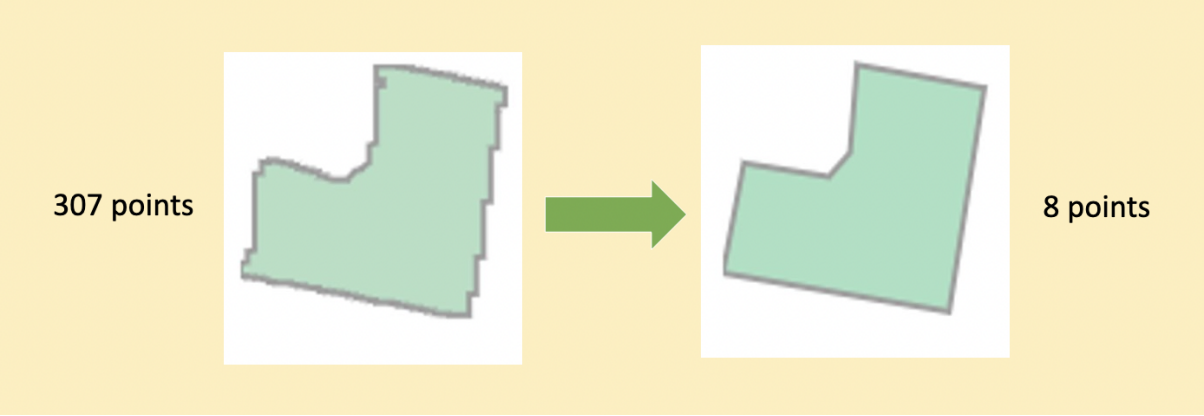


Figure 4.7: Example of polygon refinement with Douglas-Peucker.

5 Experimental results

5.1 Visual results

This section provides a visual presentation of the finished predicted polygons after being subjected to all the steps in the pipeline, which was presented in Figure 4.1. Predicted polygons and ground truth polygons are presented on a map, and this is shown in Figure 5.1 - Figure 5.7. Ground truth polygons are filled with green color. Predicted polygons from alpha-shape are visualised with red boundaries, and predicted polygons from binary images have blue boundaries. Note that all predicted polygons come from validation and test-set.

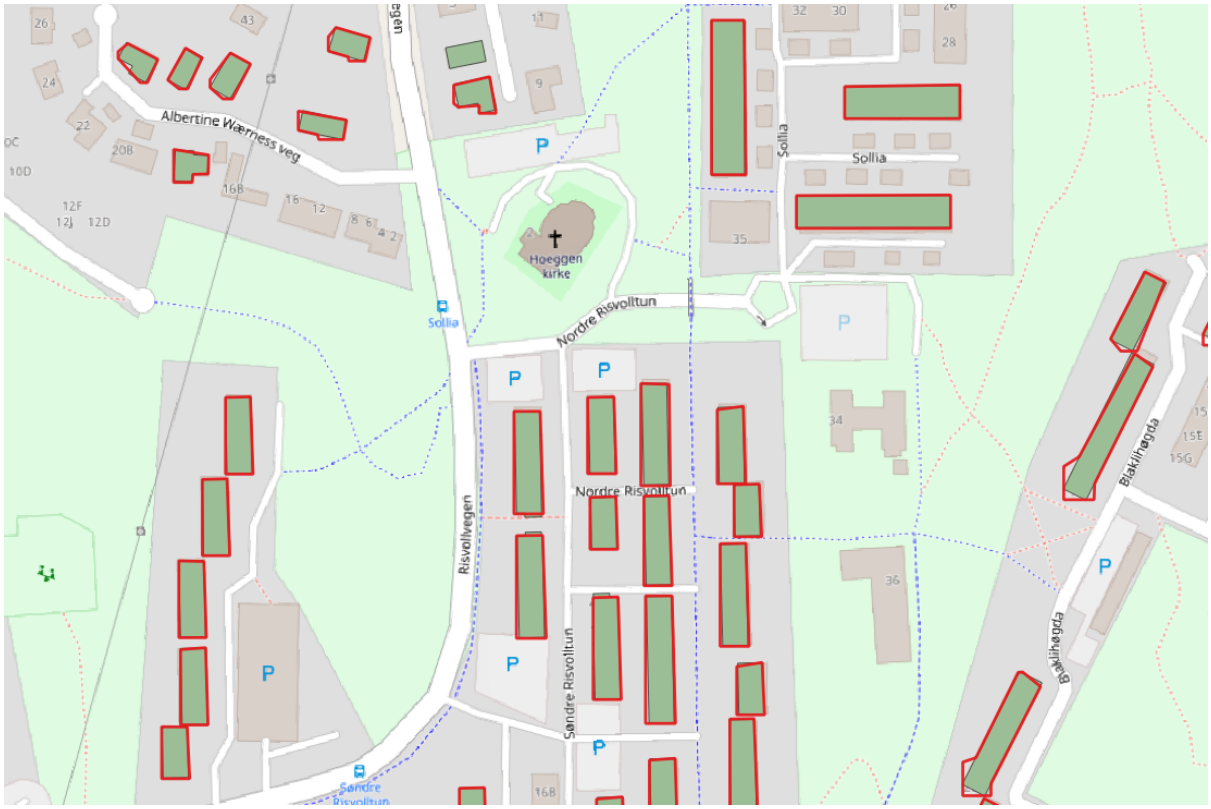


Figure 5.1: A map providing overview of predicted polygons and ground truth polygons. Red boundary is predicted building boundaries based on alpha-shape input, and ground truth polygons are filled with colour [RGB]0,128,0 green.

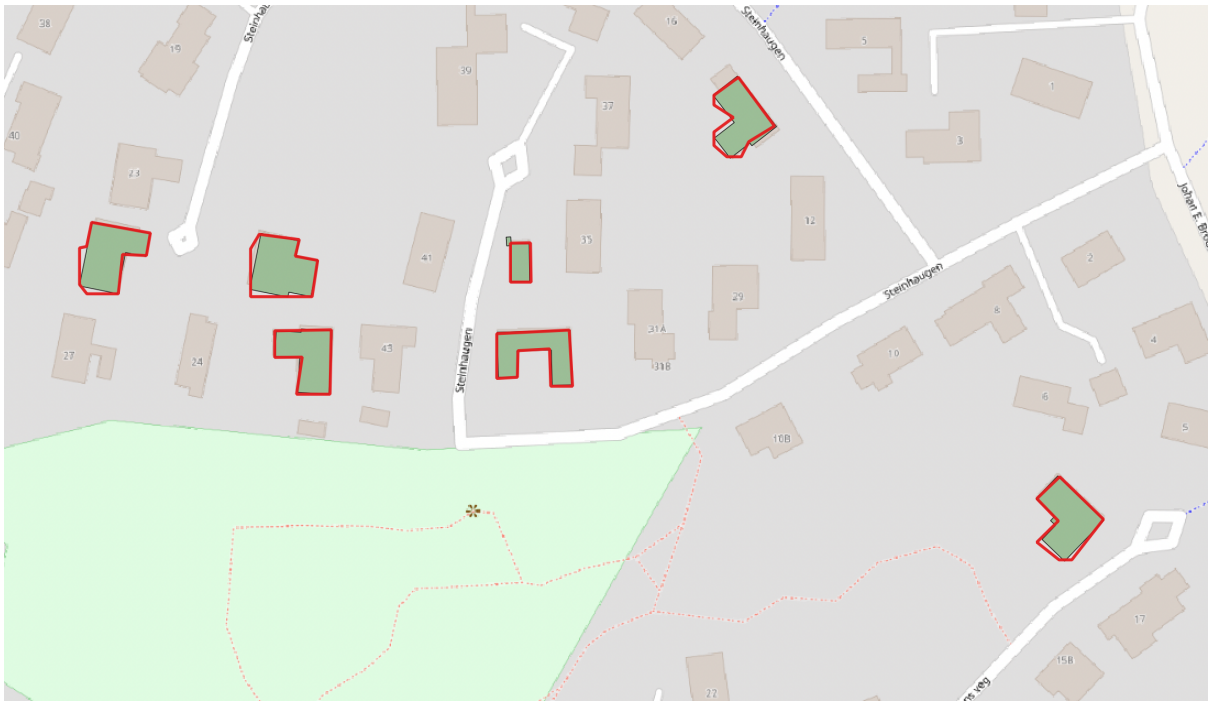


Figure 5.2: A map representing a variety of complex and more simple polygons.

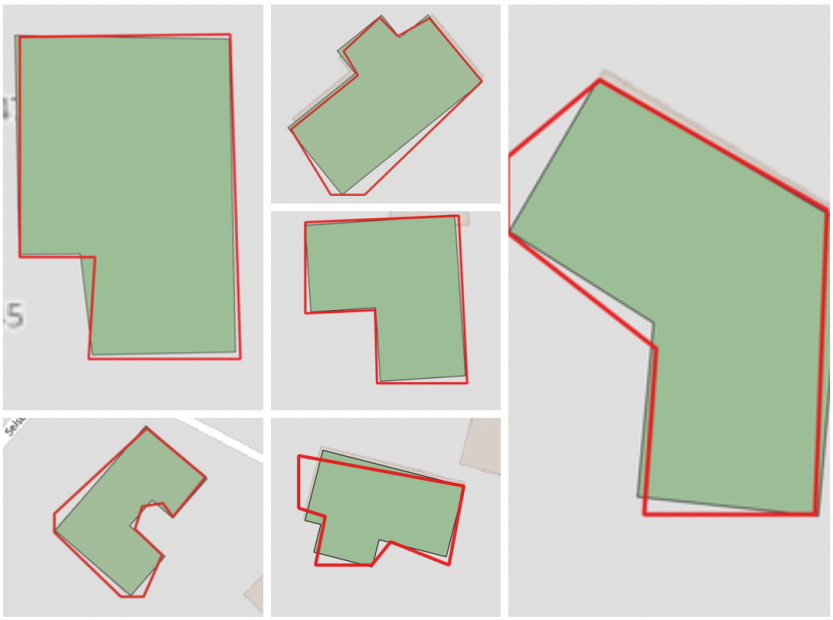


Figure 5.3: Close up of samples of polygons with different geometrical properties. Note the misalignment of edges.

Figure 5.5: Predicted polygons from alpha-shape input presented with the ground truth polygons.

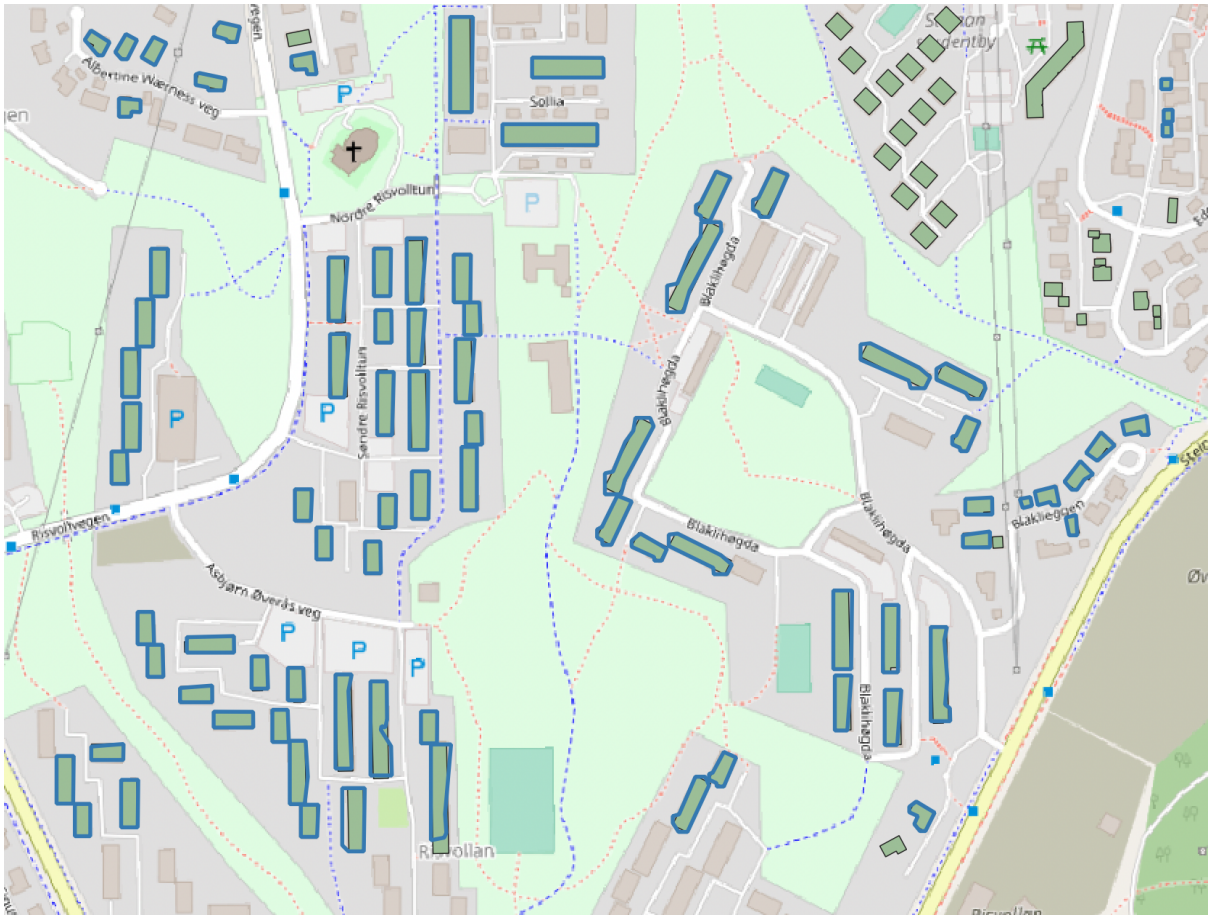


Figure 5.6: A map providing overview of predicted polygons and ground truth polygons. Blue boundary is predicted building boundaries based on binary mask image input, and ground truth polygons are filled with colour [RGB]0,128,0 green.

Figure 5.7: A close up of samples providing overview of polygons from binary mask image input. Noticeable inferior to alpha-shape input in Figure 5.2

5.2 Model selection in GAN

After training the GAN, the best[1] epoch model must be selected. To do this, a scoring between $[0 \rightarrow 1]$ is given to each epoch. This scoring is Intersection over Union (IOU), a measure of area similarity between two polygons. Thus, each generated image can be transformed into a polygon and then compared to the ground-truth polygon to score each image. The mean IOU for each validation set thus provides the basis for model selection. Each epoch mean IOU score is shown in Figure 5.11 and Table 5.2.1. For example, the best epoch for alpha-shape input is epoch 90, and for binary input, the best epoch is 50. All following analysis and calculations will be done on predicted polygons from the test set, with the model weights from the best epoch. The test set consists of 93 samples.

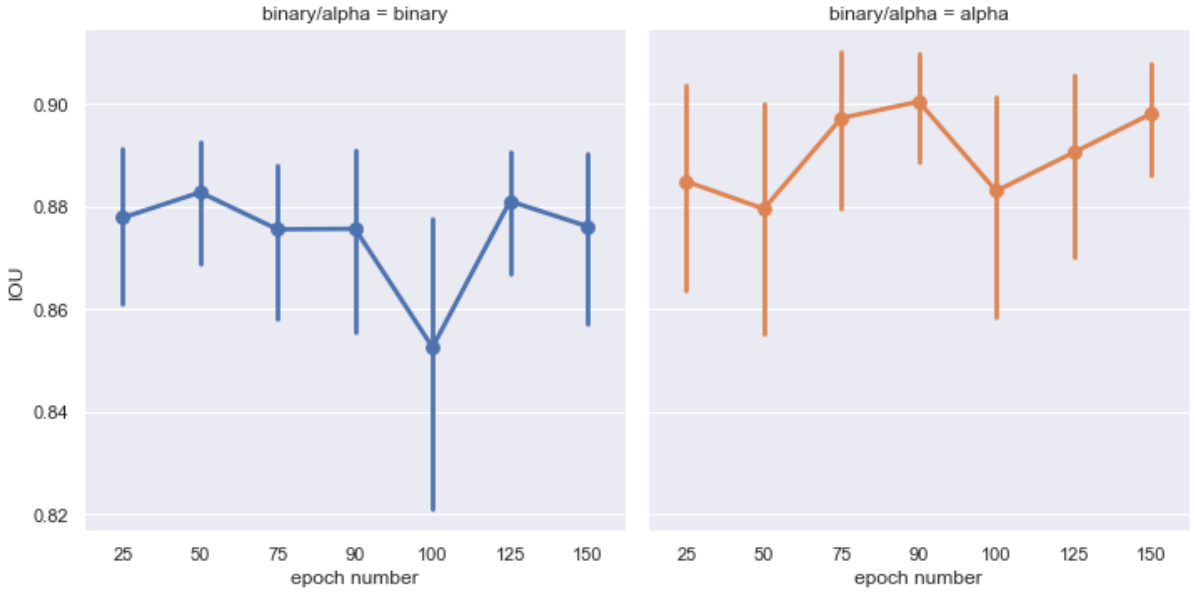


Figure 5.8: Intersection over union on the validation set for each 25th epoch. Note that the vertical lines on each points shows the standard deviation of the mean. Also, note that IOU scores are calculated on the raw, unrefined polygons.

[1]Which model that is "best" is solely based upon which criteria is set for evaluation

Table 5.2.1: Table overview of model scores on different epochs. Scores are calculated on validation set.

Input type	**Epoch number**	**IOU**	**Number of buildings**
Alpha	25	0.885	180 of 180
Alpha	50	0.880	180 of 180
Alpha	75	0.897	180 of 180
Alpha	**90**	**0.901**	**180 of 180**
Alpha	100	0.884	180 of 180
Alpha	125	0.891	179 of 180
Alpha	150	0.898	177 of 180
Binary	25	0.878	177 of 180
Binary	**50**	**0.882**	**176 of 180**
Binary	75	0.876	177 of 180
Binary	90	0.876	107 of 180
Binary	100	0.853	154 of 180
Binary	125	0.881	177 of 180
Binary	150	0.876	159 of 180

5.3 Statistical results

5.3.1 Alpha-shape based image input

To quantify the effect the GAN network has on building footprint generation, respective baseline models are introduced. The baseline models follow the same pipeline as presented in Figure 4.1, but skips the processing of the GAN. Figure 5.9 shows how the IOU scores for a baseline- and real model.

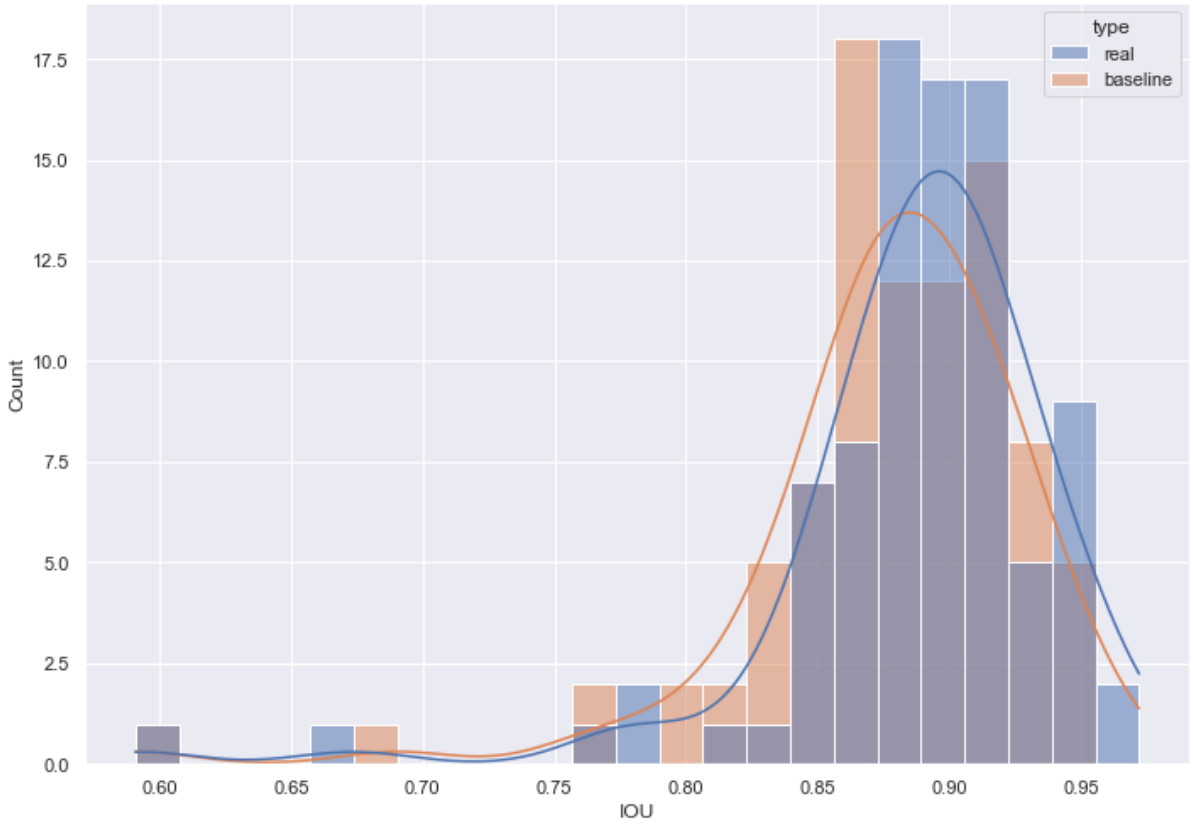


Figure 5.9: Showing distribution of calculated IOU from alpha-shape images. Simple visual inspection hints at skewed normal distribution of the calculated IOU scores. Note that IOU scores are calculated from test-set.

Many statistical tests assume that the data is normally distributed; therefore, checking for normality is the first step of the analysis. Next, the threshold value is set to $\alpha = 0.05$ for comparing against *p-values*.

When testing for normality with Shapiro test (Shapiro and Wilk 1965), the null hypothesis is H_0*: data were drawn from a normal distribution.*

Shapiro Wilk normality test (Alpha)

	Real IOU	Baseline IOU
Statistics	0.781	0.831
P-value	3.06×10^{-10}	1.24×10^{-8}

As *p-value*, > 0.05, the initial null hypothesis is rejected, meaning there is evidence that the data is **not** normally distributed. Thus to analyze if the produced IOU scores differ from the baseline approach Wilcoxon-Signed-Rank Test is used. Wilcoxon-Signed-Rank Test differs from paired t-test in that it does not assume normally distributed data (Rey and Neuhäuser 2011). Further, the null hypothesis is stated as

... H_0 : $\Omega = 0$, i.e., the distribution of the differences is symmetric about zero corresponding to no difference in location between the two samples (Rey and Neuhäuser 2011).

In other words: H_0: the median difference between the datasets are 0.

Wilcoxon-Signed-Rank Test

IOU real vs IOU alpha baseline	
Statistics	P-value
877.0	2.48×10^{-6}

With a *p-value* < 0.05, this yield the rejection of the null hypothesis and thus means the median of the data sets differ. Although note that the *p-value* is only slightly less than 0.05.

5.3.2 Binary image based input

The following section is the same statistical analysis between IOU scores from binary images with GAN processing on the test set and the baseline binary method.

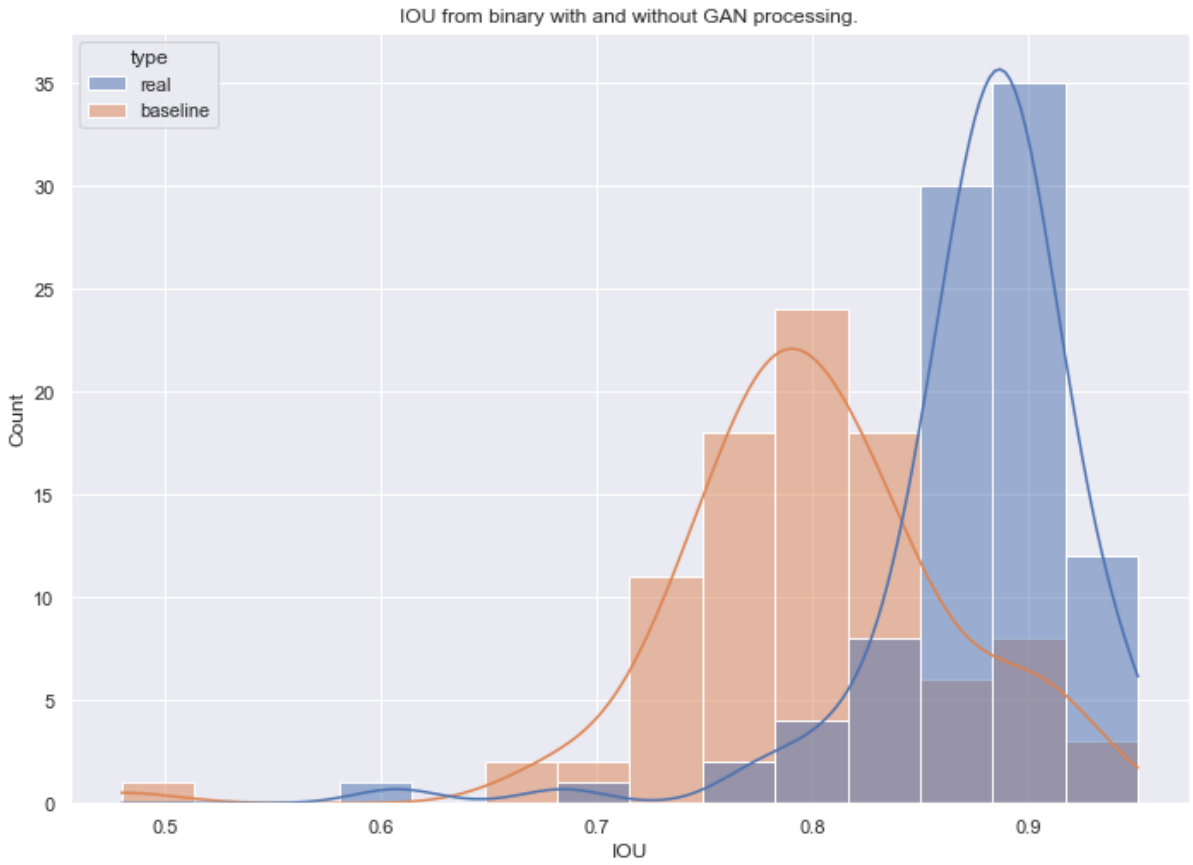


Figure 5.10: Showing distribution of calculated IOU from binary images. Simple visual inspection hints at skewed normal distribution of the calculated IOU scores.

Shapiro Wilk normality test (Binary)

	Real IOU	Baseline IOU
Statistics	0.783	0.919
P-value	2.25×10^{-10}	2.58×10^{-5}

As *p-value*, > 0.05, the initial null hypothesis is rejected, meaning there is evidence that the data is **not** normally distributed. Again Wilcoxon-Signed-Rank Test is utilized for comparing the results from the baseline approach with real results.

Wilcoxon-Signed-Rank Test

IOU Binary vs IOU BinaryBaseline	
Statistics	P-value
377	4.23×10^{-12}

With a *p-value* < 0.05, it can be stated that the initial hypothesis, H_0, is rejected and thus that binary baseline results differ significantly from real results.

Baseline IOU	Mean	Std
Alpha-baseline	0.874	0.0530
Binary-baseline	0.796	0.0651

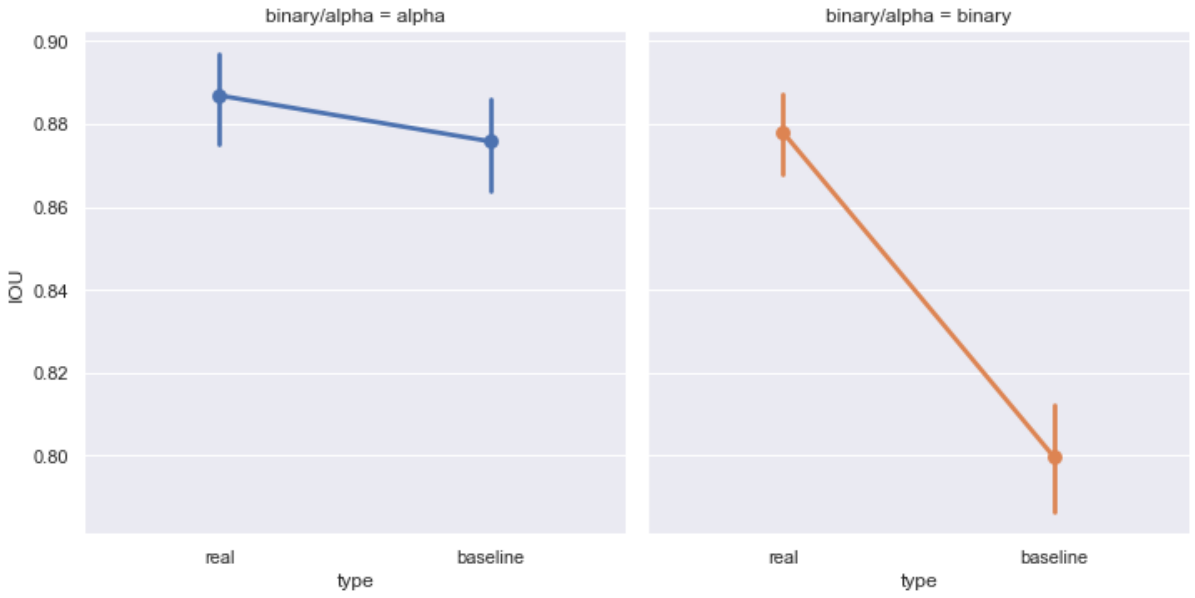


Figure 5.11: Comparing IOU on test-set from model with best performance to baseline IOU. Baseline IOU is without image processing by GAN.

5.4 IOU relation to building category

As the dataset is categorized into building categories, it is interesting to see how each of the different building categories is affected by GAN processing. Figure 5.12 shows the IOU scores for different building categories. Note that the "real" model is superior for categories with concave building outlines (*T-element, Combination* and *Corner*).

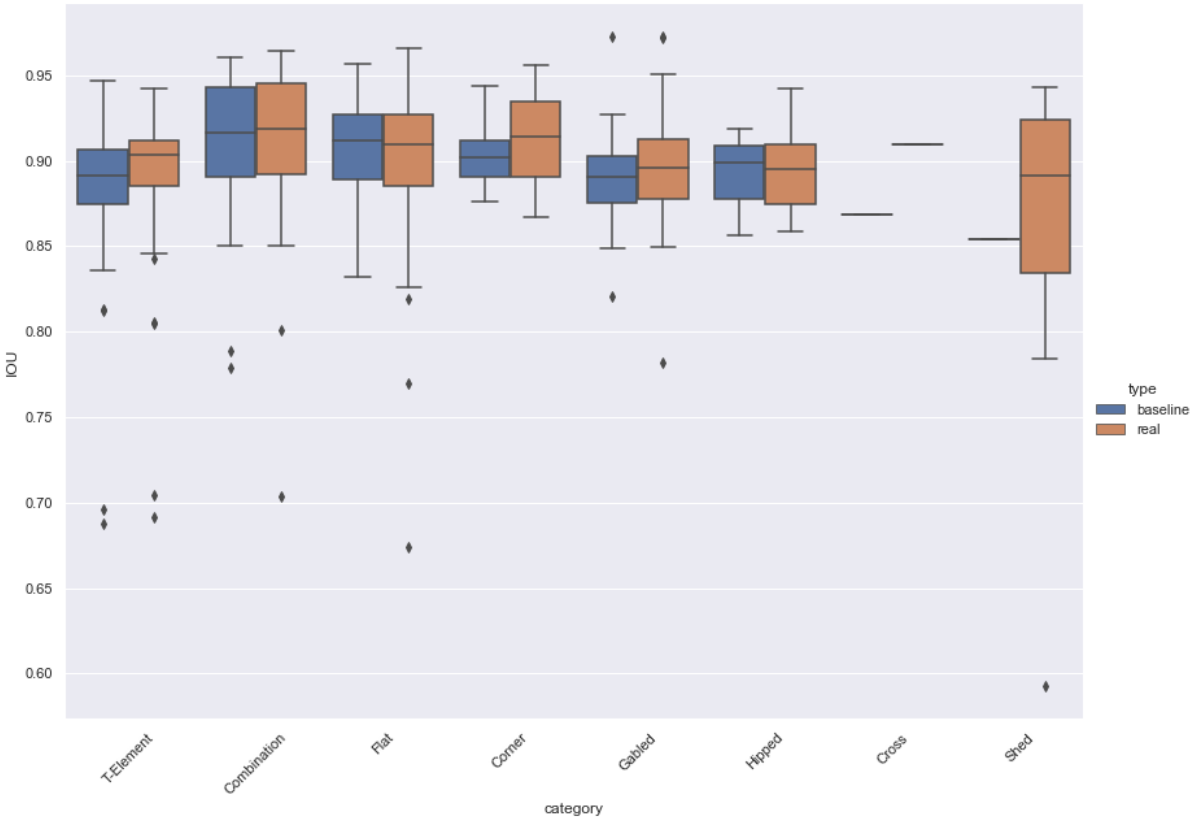


Figure 5.12: Comparing IOU on validation set from the model with best performance to the baseline IOU with respect to building categories.

6 Discussion

All in all, the predicted building polygons are quite similar to the ground truth but not perfect. Why this is the case will be discussed in this section. Further, the initial hypothesis and research questions will also be addressed. These are:

- Can image-editing with a GAN-network *better* the extracted polygon in relation to the real world building?
- To what extent, and how may it improve upon existing boundary extraction methods?
- How does the network perform on various input-types?

6.1 On the sizing problem

One of the initial hypothesis regarding the GAN was if it can help solve the sizing problem occurring when extracting boundary from pointclouds. The sizing problem was introduced in Section 2 as determining the α-shape circle in the direct approach and setting grid-size for the in-direct approach. As this thesis extracted contour with both approaches, the sizing problem was also faced. Thus the reader might be confused about how it can help in this step. One can observe the conceptual idea in Figure 4.5, where comparing the input images to Ground Truth images shows the sizing problem's imposed error. With alpha-shape input in Figure 4.5, the sizing problem produces a misrepresentation of the concave interior angle of the L-shape. In more general terms, it represents the lack of concavity. The alpha-shape image lacking concavity is fed as input to the generator network. The network modifies the image to an image more related to the ground truth. Conceptually it is clear that it improves the degree of concavity. For the in-direct approach (binary mask input), the effects of the sizing problem are represented by the difference between the ground truth and input image. In this case, the concavity is more correctly represented, but it lacks sufficient resolution to allow for high precision. Therefore, in this case, the network has to increase the image's resolution to match the ground truth image.

Examples in the Figure 4.5 indicates that GAN-editing reverts much of the imposed lack of concavity for alpha-shapes and improves the low resolution in binary mask images. To research whether this is the general case or just a fluke, a statistical analysis based on the baseline- and real approach were performed5.3.

In short, the statistical analysis states that a significant increase in IOU scores is achieved by including the GAN-processing for both alpha-shape- and binary mask input.

To understand if this is the same thing as improving the sizing problem, a fundamental breakdown of what IOU *actually* is and what it implies needs addressing.

6.2 On IOU and results

From testing different models on the validation set, the best epoch had as a mean IOU of 0.901, as seen in Table 5.2.1. The visual presentation in Section 5.1, shows what this value can correspond to and how well the predicted polygons actually represent the real-world building. In most cases, the predicted polygons have similar topological features as ground truth, but lines are often slightly misaligned, which can be seen clearly in Figure 5.3. As the selection of the model and the statistical analysis was based on IOU scores, it is important to discuss the implications of this. Simply choosing the model with the best IOU score rests on the following assumption:
. Given two polygons ρ_1, ρ_2, and a ground truth polygon GT:

$$IOU(GT, \rho_1) > IOU(GT, \rho 2) \Leftrightarrow \rho_1 > \rho_2 \tag{7}$$

Where $\rho_1 > \rho_2$ means that ρ_1 is better than ρ_2. This assumption can easily be shown false and is done so in Figure 6.1. Here the polygon on the left more "correctly" represents the green polygon but has a lower IOU score than the example on the right. Although Figure 6.1 shows how IOU can be a "misleading" score, it can still serve as a good evaluation metric, justified by three reasons.

1. Data originated from LiDAR data and represent the actual building. Albeit possible to construct mathematical violations of the assumption, buildings in the real world usually does not inhabit such abnormal geometrical shapes.

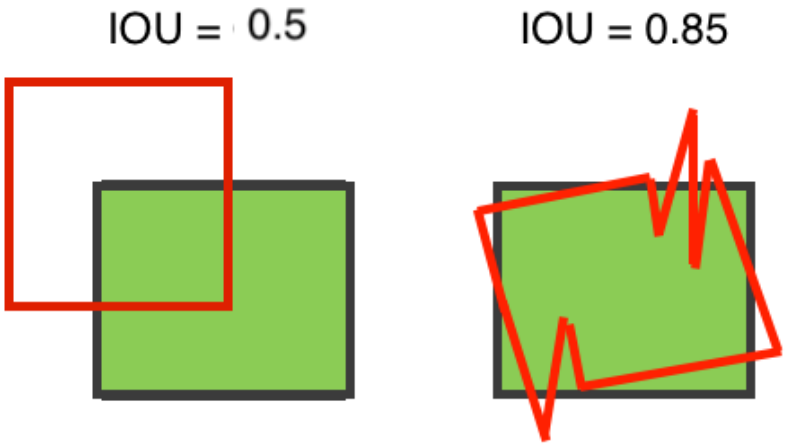


Figure 6.1: An example of how IOU can be a misleading metric when comparing generated polygons and ground truth polygons.

2. The higher the IOU score, the more constricted is the topology of the polygon. (From the fact that IOU=1 translates to identical polygons.)

3. Polygons are subjected to simplification process which, i.e. would remove the jagged lines in right side example in Figure 6.1

Further, when dealing with IOU in this experiment, it is rarely observed scores < 0.75, as seen in Figure 5.9, which implies the assumption is valid in **most** cases of this thesis. Although this is by no means proven true in all cases, it is experimentally shown to yield good results, as seen in Section 5.1.

Weakness and limitations from solely using IOU as a scoring metric for the predicted polygons have now been established. As the statistical analysis performed in Section 5.3 also is founded on the IOU metric, the outcome is also grounded by limitations of IOU scoring on the assumption that a higher IOU yields a better polygon. Thus comparing two polygons by their IOU proves a difficult task and will not suffice. From Figure 5.12 the IOU scores for the different building categories are presented. For all categories with the most concave building boundary, the IOU scores are higher with GAN processing compared to the baseline. Although not statistically proven significant, it strongly indicates that the GAN reverts a portion of the imposed error by the $\alpha-$shape method. Lastly, in combination with observing the processing in Figure 4.5, there is a

strong indication that the GAN increases IOU by more correctly representing concave buildings boundaries.

In many ways, the discussion boils down to one question: does the GAN network yield more informational gain than what is lost when transforming data back and forth between vector and raster. This brings up the level of accuracy. All steps in the proposed pipeline, presented in Figure 4.1, introduces uncertainties in the data. Even the data acquisition is not perfect and introduces uncertainties in the measurements. Thus, as long as the methods operate with (information-loss $\approx$ uncertainty), little information is actually lost in the process.

6.3 On other aspects of GAN processing

A way to minimize the imposed error from the sizing problem in alpha-shapes is to decide α for each pointcloud iteratively. This can be done by decreasing the radius of the rolling ball until it starts missing points. However, this is a computationally heavy process and was deemed not advantageous enough for this experiment. Additional benefits from GAN is the one-time cost of training the network. After the network is trained, processing with it comes computationally cheap.

The biggest potential for improving the proposed method is to include explicitly defined topological based geometrical rule by exploiting the fact that buildings are categorized. As images are binary, they require only one channel to be described (as most images have $[R, G, B]$ colour channels). Thus, one could utilize the available channels for simultaneously classifying building category.

It would also be possible to define a loss function for the GAN to minimize. In this thesis, binary cross-entropy was used as suggested in (Isola et al. 2016), but, i.e. changing it to IOU could yield potential improvement, especially if one could, i.e. take a weighted mean between a topological similarity metric and IOU.

6.4 On alpha-shape images versus binary mask images

A big backside in the field of deep learning is the black-box problem. It is very complex and difficult to justify why the network does what it does, i.e. it started to generate spots and weird patterns around epoch 100. For this reason, this experiment looked at two different types of images through the pipeline. Binary images were "worse" and, in many ways, served as a control group. By comparing the IOU scores (Best-alpha: 0.901 vs Best-binary-mask: 0.882) and visual results in section 5.1, the binary is inferior, but stills perform reasonably well. Combined with the fact that the baseline approach was improved ($0.796 \rightarrow 0.882$), it suggests the GAN can enhance on low-quality images. A result of this is how optimizing grid size no longer becomes as important. But, seen in Table 5.2.1 some epochs edited the images to the extent where producing polygons from them were not possible. For example, epoch 90(binary mask image) resulted in only 107 valid polygons. This fact point to weaknesses in dealing with low-quality data input.

6.5 On re-occurring sizing problem

A re-occurring theme in this thesis is the sizing problem. This problem will always be present when transforming data between vector- and raster representation. In the step of going from image to polygon, another variant of this problem re-surfaces. This stems from how the GAN creates sharp edges and strange patterns, which require further processing before extracting contour. The processing is morphological closing, which is dilution followed by erosion. Dilution is a way of smoothing the edge pixels by striding a kernel on the image. This is, ironically, the same thing as alpha-shape contour extracting, and now the problem is determining the kernel size. As the operation is performed on image data,which is 2-dimensional, it simplifies the problem, and with high-resolution images, it is easier to implement a low sized kernel. I.e. if a buildings max distance= 20m, one-pixel(width×height) will correspond to ∼(8cm×8cm), thus allowing for higher precision when smoothing the edge on a few pixels.

But a question then becomes: is it possible for deep learning processing to achieve such high precision to avoid the sizing problem? Well, of course, if we could yield images with perfect boundary pixels and straight edges (IOU = 1), this would make the process of

going back to vector representation much easier. But as data originates from real-world LiDAR, it deems impossible to include all intricate details and complexities required to represent the real-world counterpart accurately. But as deep learning is rapidly developing and networks that operate on vector data has been developed, chances are deep-learning methods in the future can fully automate the building boundary extraction process.

7 Conclusion

This research set out to investigate potential improvements in building boundary extraction by including processing by the GAN network. Quantitative analysis showed a significant increase in the IOU score of the predicted building polygons. Furthermore, qualitative observation of how the increase in IOU manifests in the processing implies the GAN is reverting some of the imposed error from the $\alpha-$shape method. In many ways, the ultimate questions boil down to if the advantage of including GAN-processing yield more informational gain and computational efficiency versus the negatives of the extra required transformations from vector$\rightarrow$raster and raster$\rightarrow$vector.

It was shown that after processing by the GAN network and transforming back to vector representation, the IOU score was higher than the initial alpha-shape. But as further discussed, IOU scores does not mean everything. If it's not possible to optimize IOU scores to the extent of a nearly perfect representation of the predicted polygon with GAN processing and geometrical rule-based constrictions are necessary to achieve this, it is debatable whether the inclusion is needed. But, assuming it is easier to correctly simplify a polygon with higher IOU, it is of high value and relevance to include deep-learning processing.

8 Further work

A big advantage to working with such a conceptually and visually straightforward problem is observing what could further improve the result by looking at the result. So the natural step forward is to introduce explicit geometrical rules in the polygon simplification step based on topological features from the misalignment of the predicted polygon and the disadvantages of scoring polygons based on their IOU score. A way to do this could be to simultaneously train the GAN to output what category the building belongs to.[2] And from knowing what type of building enforce, i.e. the number of edges or how many main directions, etc...

Of course, the process of editing edges is not constricted to the image domain, and the extra steps back and forth between vector and raster data is not advantageous. Therefore, a natural improvement is finding a vector-based network, meaning it can operate directly on the point set itself. Promising proposed ways of doing this is presented by (Sheng Zhang, Song, and W. Lu 2021) and (H. Wang et al. 2017).

Other aspects that would be interesting to include are deep learning-based edge detection ("Xie and Tu 2015),

[2]This is possible as binary images only utilize one colour channel, and two is available for passing other information.

References

"Xie, Saining and Zhuowen" Tu (2015). “Holistically-Nested Edge Detection”. In: *Proceedings of IEEE International Conference on Computer Vision*.

Alharthy, Abdullatif and James Bethel (2002). “Heuristic filtering and 3D feature extraction from LIDAR data”. In: *International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences* 34.3/A. Publisher: NATURAL RESOURCES CANADA, pp. 29–34.

Awrangjeb, M. (2016). “Using point cloud data to identify, trace, and regularize the outlines of buildings”. In: *International Journal of Remote Sensing* 37.3. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/01431161.2015.1131868, pp. 551–579. DOI: 10.1080/01431161.2015.1131868. URL: https://doi.org/10.1080/01431161.2015.1131868.

Bisong, Ekaba (2019). “Google Colaboratory”. In: *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*. Berkeley, CA: Apress, pp. 59–64. ISBN: 978-1-4842-4470-8. DOI: 10.1007/978-1-4842-4470-8_7. URL: https://doi.org/10.1007/978-1-4842-4470-8_7.

Canny, John (1986). “A Computational Approach to Edge Detection”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-8.6, pp. 679–698. DOI: 10.1109/TPAMI.1986.4767851.

Chawda, Chandan, Jagannath Aghav, and Swapnil Udar (2018). “Extracting Building Footprints from Satellite Images using Convolutional Neural Networks”. In: pp. 572–577. DOI: 10.1109/ICACCI.2018.8554893.

CloudCompare - Open Source project (2021). URL: https://www.danielgm.net/cc/ (visited on 05/18/2021).

Congalton, Russell (1997). “Exploring and Evaluating the Consequences of Vector-to-Raster and Raster-to-Vector Conversion”. In: *Photogrammetric Engineering and Remote Sensing* 63, pp. 425–434.

Dorninger, Peter and Norbert Pfeifer (2008). "A Comprehensive Automated 3D Approach for Building Extraction, Reconstruction, and Regularization from Airborne Laser Scanning Point Clouds". In: *Sensors* 8.11, pp. 7323–7343. ISSN: 1424-8220. DOI: 10.3390/s8117323. URL: https://www.mdpi.com/1424-8220/8/11/7323.

Douglas, David H. and T. Peucker (1973). "ALGORITHMS FOR THE REDUCTION OF THE NUMBER OF POINTS REQUIRED TO REPRESENT A DIGITIZED LINE OR ITS CARICATURE". In: *Cartographica: The International Journal for Geographic Information and Geovisualization* 10, pp. 112–122.

Edelsbrunner, Herbert and Ernst P. Mücke (Jan. 1994). "Three-Dimensional Alpha Shapes". In: *ACM Trans. Graph.* 13.1. Place: New York, NY, USA Publisher: Association for Computing Machinery, pp. 43–72. ISSN: 0730-0301. DOI: 10.1145/174462.156635. URL: https://doi.org/10.1145/174462.156635.

Eich, Markus, Malgorzata Dabrowska, and Frank Kirchner (2021). "Semantic Labeling: Classification of 3D Entities Based on Spatial Feature Descriptors". In: URL: http://www.best-of-robotics.org/pages/BRICS-events/icra2010/SemanticLabeling2010Eich.pdf.

FKB-Bygning - Kartkatalogen (2021). URL: https://kartkatalog.geonorge.no/metadata/fkb-bygning/8b4304ea-4fb0-479c-a24d-fa225e2c6e97 (visited on 05/19/2021).

Goodfellow, Ian (2017). *NIPS 2016 Tutorial: Generative Adversarial Networks*. _eprint: 1701.00160. URL: https://arxiv.org/abs/1406.2661.

Isola, Phillip et al. (2016). "Image-to-Image Translation with Conditional Adversarial Networks". In: *arxiv*.

Jarvis, RA (1977). "Computing the shape hull of points in the plane". In: *Proceedings of the IEEE Computing Society Conference on Pattern Recognition and Image Processing*. New York, pp. 231–241.

Kada, Martin and Laurence Mckinley (2009). "3D building reconstruction from LiDAR based on a cell decomposition approach". In: *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 38.

Lorensen, William E. and Harvey E. Cline (Aug. 1987). "Marching Cubes: A High Resolution 3D Surface Construction Algorithm". In: *SIGGRAPH Comput. Graph.* 21.4. Place: New York, NY, USA Publisher: Association for Computing Machinery, pp. 163–169. ISSN: 0097-8930. DOI: 10.1145/37402.37422. URL: https://doi.org/10.1145/37402.37422.

Mongus, Domen, Niko Lukač, and Borut Žalik (2014). "Ground and building extraction from LiDAR data based on differential morphological profiles and locally fitted surfaces". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 93, pp. 145–156. ISSN: 0924-2716. DOI: https://doi.org/10.1016/j.isprsjprs.2013.12.002. URL: https://www.sciencedirect.com/science/article/pii/S0924271613002840.

Pohle-Fröhlich, Regina et al. (2019). "Roof Segmentation based on Deep Neural Networks". In: *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 4: VISAPP,* Backup Publisher: INSTICC ISSN: 2184-4321. SciTePress, pp. 326–333. ISBN: 978-989-758-354-4. DOI: 10.5220/0007343803260333.

Qi, Charles R. et al. (2017). *PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space.* _eprint: 1706.02413. URL: https://arxiv.org/abs/1706.02413.

Ramiya, Anandakumar M., Rama Rao Nidamanuri, and Ramakrishan Krishnan (2017). "Segmentation based building detection approach from LiDAR point cloud". In: *The Egyptian Journal of Remote Sensing and Space Science* 20.1, pp. 71–77. ISSN: 1110-9823. DOI: https://doi.org/10.1016/j.ejrs.2016.04.001. URL: https://www.sciencedirect.com/science/article/pii/S1110982316300060.

Rey, Denise and Markus Neuhäuser (2011). "Wilcoxon-Signed-Rank Test". In: *International Encyclopedia of Statistical Science.* Ed. by Miodrag Lovric. Berlin, Heidelberg:

Springer Berlin Heidelberg, pp. 1658–1659. ISBN: 978-3-642-04898-2. DOI: 10.1007/978-3-642-04898-2_616. URL: https://doi.org/10.1007/978-3-642-04898-2_616.

Sampath, Ajit and Jie Shan (2007). "Building Boundary Tracing and Regularization from Airborne LIDAR Point Clouds". In: *Photogrammetric Engineering and Remote Sensing* 73. DOI: 10.14358/PERS.73.7.805.

Shapiro, S. S. and M. B. Wilk (1965). "An Analysis of Variance Test for Normality (Complete Samples)". In: *Biometrika* 52.3/4. Publisher: [Oxford University Press, Biometrika Trust], pp. 591–611. ISSN: 00063444. URL: http://www.jstor.org/stable/2333709.

Shi, Wenzhong and ChuiKwan Cheung (2006). "Performance Evaluation of Line Simplification Algorithms for Vector Generalization". In: *The Cartographic Journal* 43.1. Publisher: Taylor & Francis _eprint: https://doi.org/10.1179/000870406X93490, pp. 27–44. DOI: 10.1179/000870406X93490. URL: https://doi.org/10.1179/000870406X93490.

Suzuki, Satoshi and Keiichi Abe (1985). "Topological structural analysis of digitized binary images by border following". In: *Computer Vision, Graphics, and Image Processing* 30.1, pp. 32–46. ISSN: 0734-189X. DOI: https://doi.org/10.1016/0734-189X(85)90016-7. URL: https://www.sciencedirect.com/science/article/pii/0734189X85900167.

Teh, C.-H and Roland T Chin (1989). "On the detection of dominant points on digital curve". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 11, pp. 859–872. DOI: 10.1109/34.31447.

The Shapely User Manual — Shapely 1.7.1 documentation (2021). URL: https://shapely.readthedocs.io/en/stable/manual.html (visited on 05/22/2021).

Wang, Hongwei et al. (2017). *GraphGAN: Graph Representation Learning with Generative Adversarial Nets*. _eprint: 1711.08267. URL: https://arxiv.org/abs/1711.08267.

Wei, Shen (2008). "Building boundary extraction based on LIDAR point clouds data". In: *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* 37.

Wei, Shiqing, Shunping Ji, and Meng Lu (2020). "Toward Automatic Building Footprint Delineation From Aerial Images Using CNN and Regularization". In: *IEEE Transactions on Geoscience and Remote Sensing* 58.3, pp. 2178–2189. DOI: 10.1109/TGRS.2019.2954461.

Zeng, Chuiqing, Jinfei Wang, and Brad Lehrbass (2013). "An Evaluation System for Building Footprint Extraction From Remotely Sensed Data". In: *Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal of* 6, pp. 1640–1652. DOI: 10.1109/JSTARS.2013.2256882.

Zhang, K., J. Yan, and S.-C. Chen (2006). "Automatic Construction of Building Footprints From Airborne LIDAR Data". In: *IEEE Transactions on Geoscience and Remote Sensing* 44.9, pp. 2523–2533. DOI: 10.1109/TGRS.2006.874137.

Zhang, Sheng, Rui Song, and Wenbin Lu (2021). *Graph{CGAN}: Convolutional Graph Neural Network with Generative Adversarial Networks*. URL: https://openreview.net/forum?id=iy3xVojOhV.

Zhang, Su, Fei Han, and Susan M. Bogus (2020). "Building Footprint and Height Information Extraction from Airborne LiDAR and Aerial Imagery". In: *Construction Research Congress 2020*. _eprint: https://ascelibrary.org/doi/pdf/10.1061/9780784482865.035, pp. 326–335. DOI: 10.1061/9780784482865.035. URL: https://ascelibrary.org/doi/abs/10.1061/9780784482865.035.

A Appendix

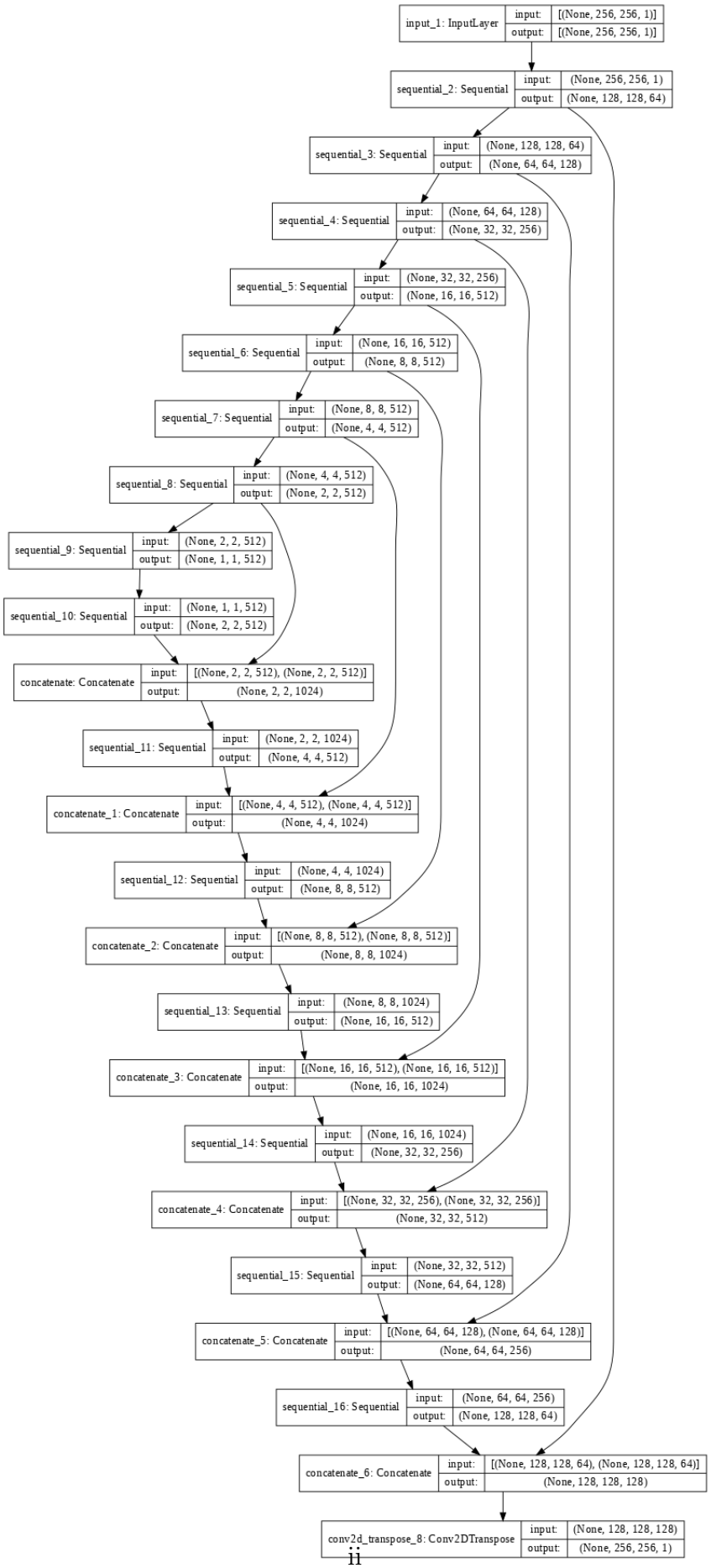


Figure A.1: Showing layers and architecture of Generator network.

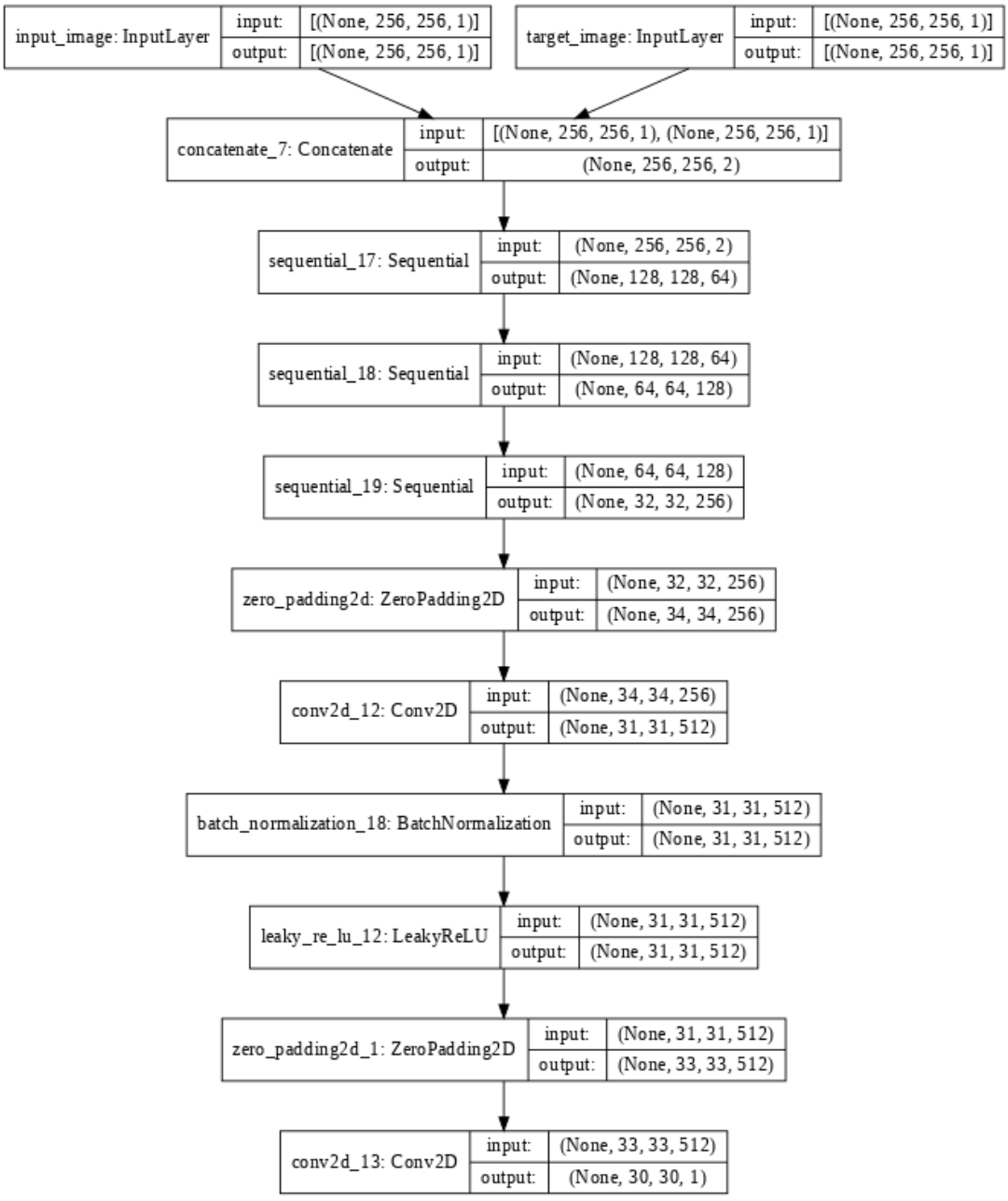
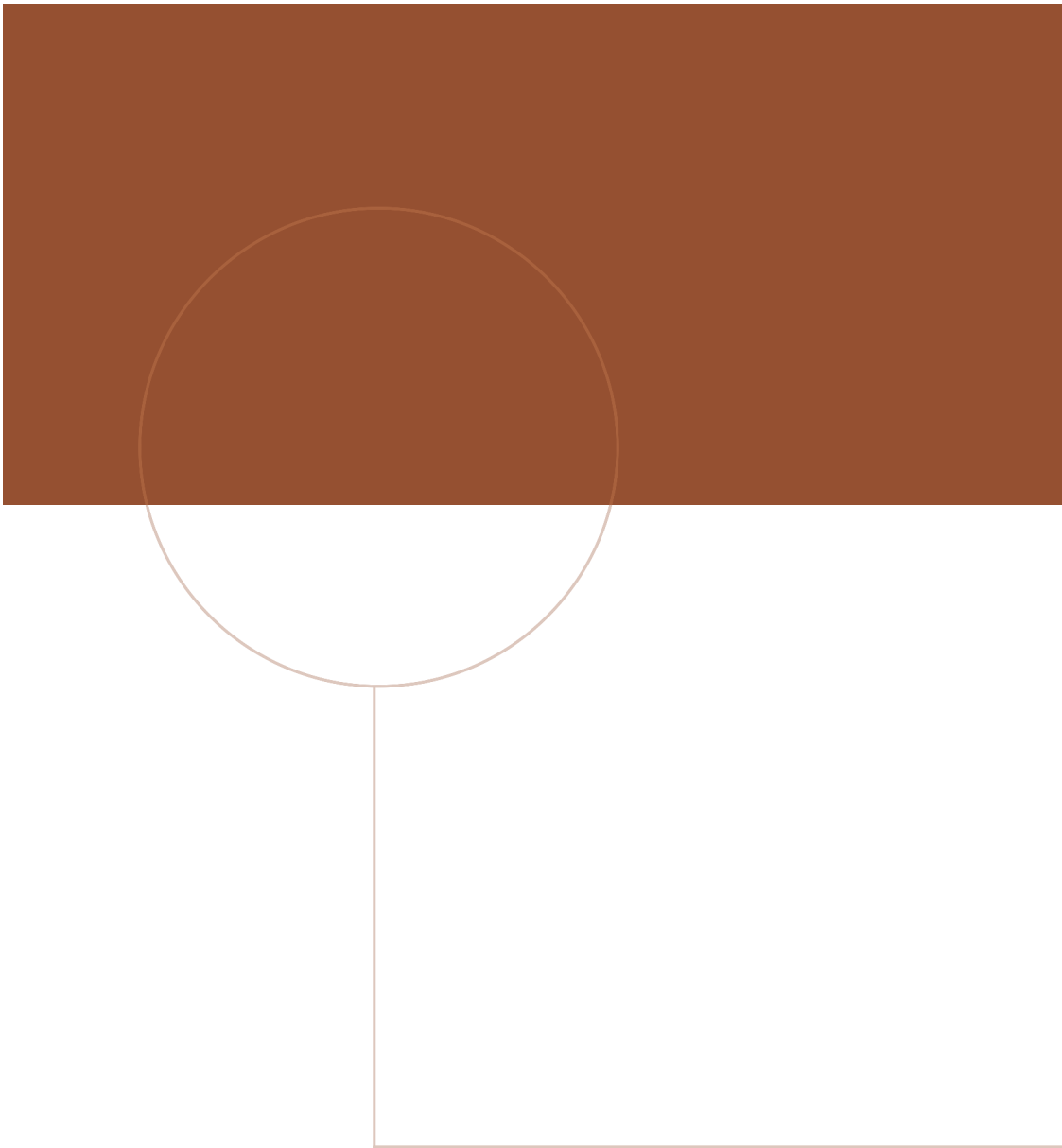


Figure A.2: Showing layers and architecture of Discriminator network.



NTNU
Norwegian University of
Science and Technology