University of Zurich[UZH]

# Exploring the Swin Transformer Architecture for the Generalization of Building Footprints in Binary Cartographic Maps

GEO 511 Master's Thesis

**Author**
Jan Winkler
15-728-686

**Supervised by**
Prof. Dr. Robert Weibel
Dr. Cheng Fu
Dr. Zhiyong Zhou

**Faculty representative**
Prof. Dr. Robert Weibel

20.01.2023
Department of Geography, University of Zurich

# Abstract

This thesis explores using two distinct deep-learning models and three different data models to automate the process of cartographic generalization. Cartographic generalization aims to select essential information, preserve typical elements, and simplify the information content to allow legibility in maps across different scales. Specifically, the thesis focuses on automating the generalization of building footprints. The automation of the described process has proven challenging in the past. The thesis unveils that increased computation power, better computation models, and more data alone will not solve the issue. Moreover, the thesis shows that a better approach is needed to feed data to the computation model. Comparing the performance of U-Net and Swin Transformer computation models reveals that U-Net with convolutions outperforms Swin Transformers, which use attention mechanisms. The thesis suggests that a data model with an artificial attention mechanism rather than a computation model with an attention mechanism is needed to learn the different generalization tasks on a building level. The study then points out its limitations, including a need for more balanced data to train the Transformer model from scratch. Future research could focus on creating the needed more representative training data. Finally, it outlines the possibility of building a purpose-built Transformer model for future use.



**Keywords:** Automating, Cartographic Generalization, Buildings, Deep Learning, Data Model, Computation Model, U-Net, Swin Transformer

## Acknowledgements

First and foremost, I would like to extend my gratitude to Professor Dr. Robert Weibel, Dr. Cheng Fu, and Dr. Zhiyong Zhou for their efficient guidance and support throughout this project. Their expertise, advice, and pertinent feedback challenged me to revisit concepts and critically assess my work.

I am also grateful to Dr. Yu Feng and Professor Dr. Monika Sester for providing unrestricted access to their data and code.

Next, I wanted to thank the team at swisstopo, specifically Dr. Roman Geisthoevel, for providing comprehensive data, ideas, insights, and support.

Also, I want to express my special thanks to Nicolas Beglinger for the countless hours of discussions on the topic, data, and deep-learning models. These discussions were essential to the success of this project.

Finally, I sincerely thank my partner, family, and friends for their unwavering encouragement throughout this journey. This project would not have been possible without their love and support.

# Contents

# Chapter 1

# Introduction

## 1.1 Motivation

Cartography has come a long way. Even in ancient times, humans have generated maps for better orientation or simple navigation and struggled with generalizing earth features (McMaster and Stuart Shea, 1992). Nowadays, maps and their different representations are everywhere. For example, individual navigation, search & rescue, health, city planning, or politics are examples from an exhaustive list of applications that use digital maps. Nonetheless, the fundamental geometric tradeoff of placing as much valuable information as possible on the map while keeping up its legibility remains a challenge (Weibel and Dutton, 1999). Additionally, background map data is crucial for geographical information systems (GIS) and location-based services, but keeping it updated with constantly changing landscapes and built structures is becoming perpetually complex (Lee et al., 2017).

Cartographers have developed specific, standardized procedures to represent the real world with its infinite level of detail in maps. For example, images, landscapes, or topographic models can build a representation of the real world. These, in turn, can then be used as a base for producing cartographic models to scale (Spiess et al., 2002; Bader, 2001). Besides researching philosophical ideals of why, when, and how to generalize a map, cartographers discuss how single elements in maps, such as buildings, roads, or rivers, can be modified by applying various operators, including simplification, aggregation, elimination, or displacement. These operators ensure that specific elements can be represented on different scales, making generalization indispensable (McMaster and Stuart Shea, 1992; Weibel and Dutton, 1999; Weibel, 1995; Lee et al., 2017; Steiniger et al., 2008). In essence, map generalization aims to select essential information, preserve typical elements, and simplify the information content to allow legibility (Steiniger et al., 2008).

The single operators have distinct functions and act on features depicted in the map. Elimination allows the removal of small or isolated structures (i.e., buildings). The displacement operator moves buildings away from roads or separates them so that they are not depicted too close together in the desired target scale. Aggregation comes into play when grouping specific buildings into larger units if the generalization result should not separately show the buildings. Simplification allows generalizing indentations or contorted building geometries (Lee et al., 2017; Spiess et al., 2002). Often, more than one single operator is used for a single structure (e.g., simplification and displacement) (Harrie and Weibel, 2007; Steiniger et al., 2008). For a long time, the procedural method to achieve automated or semi-automated generalization was to acquire explicit knowledge from experts (cartographers), which

allowed defining constraints that drive optimization models. As it turned out, the generalization process's holistic automation proved disreputably challenging.

Current works in artificial intelligence have used neural networks trained for semantic segmentation to solve the described issue with a more technically holistic approach. Specifically, pioneering papers including Sester et al. (2018); Feng et al. (2019); Kang et al. (2020) and Courtial et al. (2021) use single-channeled image data coupled with convolutional neural networks (CNNs) to learn the complicated process of building generalization. All papers show promising results; however, the used computation models, loss functions, and data models show insufficient capabilities to produce meaningful predictions for complicated generalization tasks. A reasonable path to pursue could be introducing new computation models (architectures), loss functions, and data models to automate cartographic generalization. The recent advent in computer vision (CV) of a neural network architecture originating in natural language processing (NLP), called Transformer, could thus be a valid path to pursue. Transformers are gradually replacing state-of-the-art (SOTA) models in computer vision deep learning, more specifically semantic segmentation on single- or multi-channeled images (Mai et al., 2022; Dosovitskiy et al., 2020; Liu et al., 2021; Xu et al., 2021). The Transformer architecture uses a different mathematical procedure to process the input data than the CNNs, called attention mechanisms. These attention mechanisms resemble human attention, capturing more intricate details in input images. Therefore, using Transformer models could be an exciting path to increase models' capabilities to perform building generalization automatically.

## 1.2 Research Aim

This thesis aims to shed new light on automated building generalization using deep learning with an exploratory methodological approach. The main objectives are to discover the possibilities of introducing transformer architecture, task-specific loss functions, and multi-channeled data models into the domain of cartographic generalization and thereby providing solid methodological foundations for further investigating the topic. The thesis is part of the 'DeepGeneralization' project by the Department of Geography (GIUZ) at the University of Zurich in collaboration with swisstopo[1] (CHE) and the IGN[2] (FRA).

[1] `https://www.swisstopo.admin.ch/`
[2] `https://www.ign.fr/`

# Chapter 2

# Related Work

Automating generalization has seen various approaches from different research teams in the last couple of years within the realms of cartography, geoinformatics, and computer science (Sester et al., 2018; Feng et al., 2019). This section summarizes the key aspects related to cartographic generalization, deep learning, and the use of deep learning for cartographic generalization.

## 2.1 Evolution of Map Generalization

The overview provided in Harrie and Weibel (2007) shows the subsequent development of ideas to model the overall generalization process. The overview starts with condition-action models, then shows human interaction modeling, and finally presents constraint-based practices.

**Condition-Action Models:** Condition-action modeling is a two-phase map generalization method involving structural recognition (condition) and execution (action). The structural recognition phase performs the identification of and relationships between objects. The identified conditions then trigger the algorithms for generalization in the execution phase. This approach was popular in the late 1980s and helped to move away from the previous method of using hardwired algorithms to model the generalization process mechanically. However, it has some limitations, most prominently the difficulty of formalizing rules for cartographic knowledge and the need for many rules to cover the possible relationships between map objects. Despite these limitations, some systems using this approach have produced good results for specific generalization tasks (Harrie and Weibel, 2007; Steiniger et al., 2008; Schylberg, 1993; Beard, 1991; Nickerson, 1986, 1988)

**Human Interaction Modeling:** In the early 1990s, there was a focus on developing strategies for map generalization that relied heavily on human interaction. These strategies, known as interactive generalization systems, contained a range of algorithms for generalization that were available from academic research at the time. Nonetheless, these systems lacked automated process modeling to guide the algorithms. To address this, a new paradigm called "amplified intelligence" was proposed by Weibel (1991), in which the computer (generalization software) performs tasks that can be formalized as algorithms and the human guides and control the algorithms, using their ability for holistic judgment. Software vendors have produced several interactive software solutions for map generalization, some of which are toolbox approaches and others are specific interactive generalization systems (Harrie

and Weibel, 2007; Weibel, 1991; Steiniger et al., 2008). However, according to an evaluation of interactive generalization systems, there is a minimal increase in productivity, and the effectiveness of the generalization largely depends on the user's skills (Ruas, 2001). These limitations gave rise to a later instance of constraint-based models, namely agent-based methods.

**Constraint-Based Models:** Constraint-based modeling is a map generalization method involving setting conditions or constraints that the resulting map must satisfy. These constraints may include requirements for the level of detail, representation of certain map features, and distances between features. The generalization process involves finding a solution that satisfies as many constraints as possible, often through a process of compromise, because many constraints may conflict with each other. A cost may be assigned to violations of constraints to find the optimal solution, and the solution with the lowest total cost is chosen. There are three main methods for finding this optimal solution: agent modeling, combinatorial optimization, and continuous optimization. Agent modeling is versatile and can handle a wide range of generalization tasks, but it requires many accurately defined constraints and formalized plans to guide the selection of appropriate algorithms and control parameters. Combinatorial optimization is effective for tasks that can be discretized, such as selection, displacement, and simplification, but it may only be suitable for tasks with a small search space. Finally, continuous optimization is limited to rubber sheet transformations, such as feature displacement, smoothing, simplification, enlargement, and exaggeration, but it cannot handle operators such as aggregation, selection, and typification. In terms of computational efficiency, continuous optimization is the fastest, followed by combinatorial optimization and agent modeling. However, agent modeling has shown the most potential for modeling the complete set of generalization operators and integrating other constraint-based techniques, which is why it acts as state-of-the-art (Harrie and Weibel, 2007; Barrault et al., 2001; Mackaness, 1995; Ruas and Plazanet, 1996).

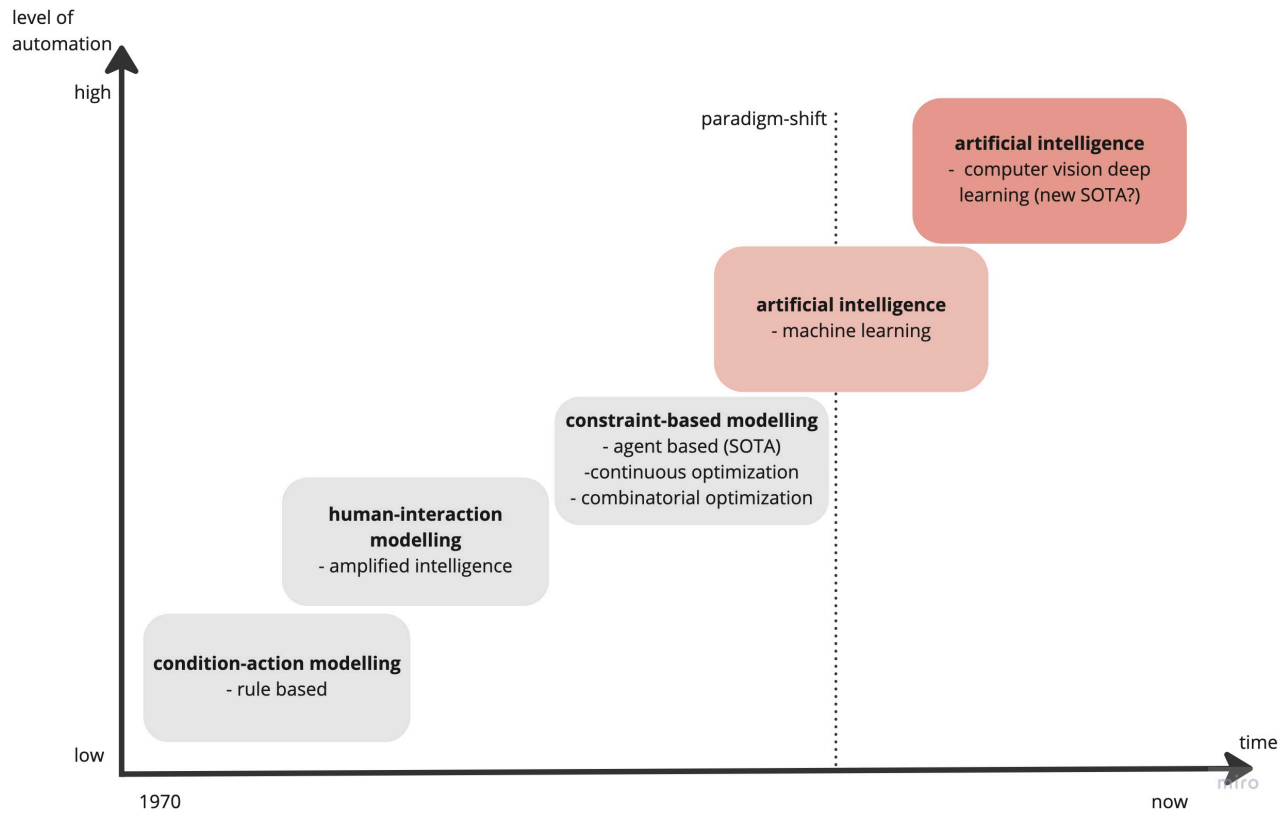



Figure 2.1: The figure provides an intuitive overview of the evolution of systems to automate cartographic generalization. The x-axis represents time, and the y-axis is the level of automation of methods to perform cartographic generalization. The dashed line signifies the paradigm shift induced by introducing artificial intelligence (AI) systems to automate map generalization.

According to the research plan provided by Professor Weibel, these applications still use specific human (expert) knowledge and can be seen as a form of amplified intelligence. Nevertheless, researchers started to use machine learning (ML) techniques (both supervised and unsupervised) and early forms of neural networks, such as self-organizing maps, to further push the ability of generalization algorithms, however, with limited success.

## 2.2 Deep Learning

Deep learning is an approach to artificial intelligence (AI) (Goodfellow et al., 2016). Various teams across domains have tackled many problems using deep learning. Why the name *deep learning*? Deep learning makes use of neural networks. Within a neural network, there are many sequential layers called hidden layers (Figure 2.3), through which the information the model should learn passes. Since there is usually more than one of these hidden layers, these networks are called *deep* neural networks. Since these deep neural networks perform *learning* tasks, the name *deep learning* developed (Goodfellow et al., 2016; Zhang et al., 2021; Chollet, 2017).

### 2.2.1 Domain - Computer Vision (CV):

One domain where deep learning is applied is called computer vision (CV). Generally, CV is a vast field with several overlaps to other disciplines such as pattern recognition, artificial intelligence, maths,

physics, or image processing (Goodfellow et al., 2016). In CV, practitioners make use of the fact that images can be stored in a computer as matrices or tensors. Storing images as matrices allows the application of mathematical computations originating in linear algebra, where matrices can be modified using matrix multiplication, addition, or changed by sub-matrices (Chollet, 2017). All these operations can be used in the image (pre-) processing or image analysis and in connection with making image data useable for neural networks. CV deep learning has seen a strong progression along with the increase in computation power in recent years. The main tasks computer vision deep learning excels can be observed in Figure 2.2 (Goodfellow et al., 2016; Zhang et al., 2021):


Figure 2.2: Three main tasks of CV deep learning. *Image Classification:* The model learns to classify an image, *Object Detection:* The model learns to detect in what region of an image specific items are located, *Image Segmentation:* The model learns to perform a pixel-wise classification. (Graphic: Patel 2020)

### 2.2.2 Deep Learning Architectures

This section provides an overview of the deep learning architectures relevant to this thesis. Specific model architectures are available depending on the task a computer vision deep learning model should learn. These model architectures are a set of algorithms that are designed to recognize patterns in visual data, such as images.

#### 2.2.2.1 Fully Connected Neural Networks - FCNN

One important neural network architecture is the fully connected neural network (FCNN) (Figure 2.3). If a network consists only of fully connected layers, it is a fully connected neural network. According to Ramsundar and Zadeh (2018), FCNNs can be used in many applications. In general, FCNNs learn the data, given enough time. However, their performance is often weaker compared to special-purpose networks tuned to the structure of a problem space. One disadvantage of FCNNs is that they are computationally costly on higher-dimensional, non-tabular data (Zhang et al., 2021).

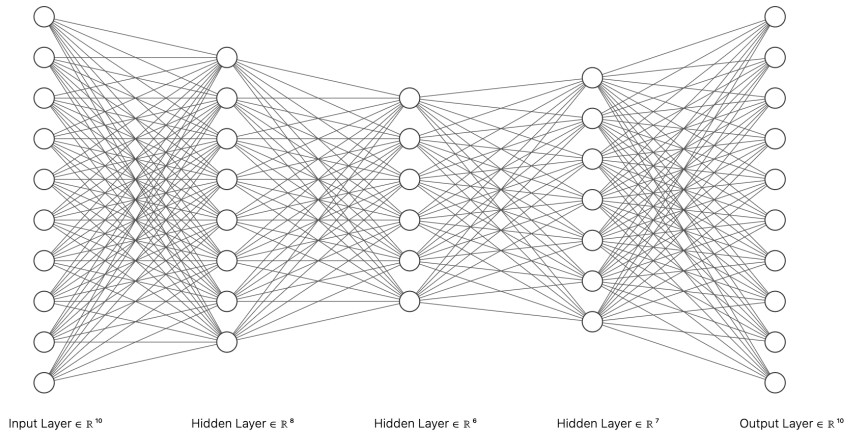



Figure 2.3: Arbitrary architecture of a fully connected neural network (FCNN). In each layer, there are several nodes and vertices. Since all the nodes in the hidden layers are connected to every other, as shown in the visualization above, the *layers* are said to be *fully connected*. If there are only *fully connected* hidden *layers*, the *network* is said to *fully connected* too. Since information can flow to any node, FCNNs are *structure agnostic* (Ramsundar and Zadeh, 2018). (Graphic: LeNail 2019)

.

#### 2.2.2.2 Convolutional Neural Networks - CNN

Another essential network architecture is the convolutional neural network. Convolutional neural networks are used on two-dimensional data, for example, images (Zhang et al., 2021; Chollet, 2017). The name of the convolutional neural network describes what happens inside the network. The hidden layers in the network perform different types of downsampling operations on the data that passes through them. Two essential down-sampling operations, convolution and pooling are described in the following:

**Convolution:** The main reason to use convolutions is to reduce the dimensionality of the data. As mentioned in Section 2.2, high-dimensional data can be stored as n-dimensional matrices. Storing the information as matrices allows for the application of matrix multiplication. When multiplying the image matrix with a pre-defined kernel matrix (Figure 2.4), the said convolution occurs. The latter operation yields a down-sampled result of the input data. Applying the convolution operation multiple times in a network increases the amount of encoded information while the dimensionality decreases (Figure 2.5).

| | | |
|---|---|---|
| 0 | 1 | 2 |
| 2 | 2 | 0 |
| 0 | 1 | 2 |

Figure 2.4: Arbitrary 3x3 kernel matrix used for convolution operations. Graphic: (Dumoulin and Visin, 2018)

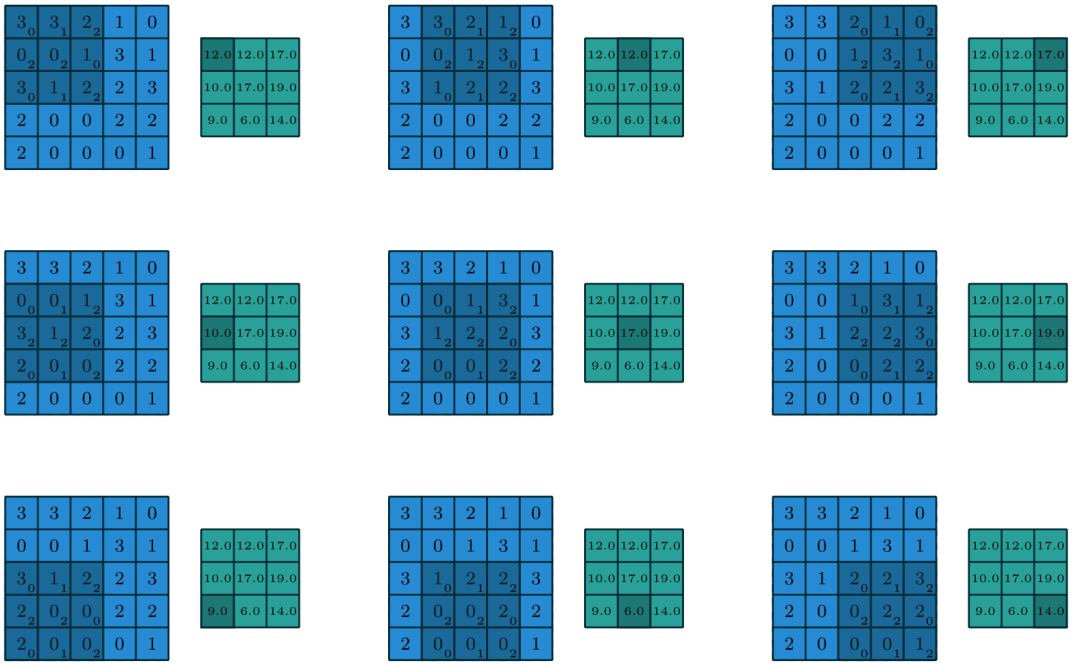


Figure 2.5: Application of convolution on 5x5 matrix with a 3x3 kernel, stride 1, yielding a 3x3 result (Dumoulin and Visin, 2018). In the top left, the algorithm starts. The numbers in the blue matrix are multiplied elementwise with the kernel. The kernel values are shown in the bottom left of each matrix element. The multiplication and summation of all the values of the first iteration lead to the result shown in the green, smaller matrix to the right (e.g., 12). Then the matrix shifts with a stride of 1 horizontally, and the procedure commences again. (Graphic: Dumoulin and Visin 2018)

Important parameters in the convolution operations are the kernel size (number of pixels in height and width) and the stride size (number of pixels to move horizontally or vertically). In the example above, the kernel has a size of 3x3 (3 pixels wide, 3 pixels high) and a stride of 1.

**Pooling:** A further down sampling strategy is pooling. Many convolutional neural networks use one or more pooling layers. There are various ways of pooling data (f.e., average, max, min values). Assuming there is an NxN matrix, pooling works similarly to the convolution operation mentioned in the prior paragraph. However, instead of multiplying the values in the NxN matrix by an MxM kernel, pooling incrementally performs the chosen (average, min, max) operation on a given kernel (see Figure 2.6).

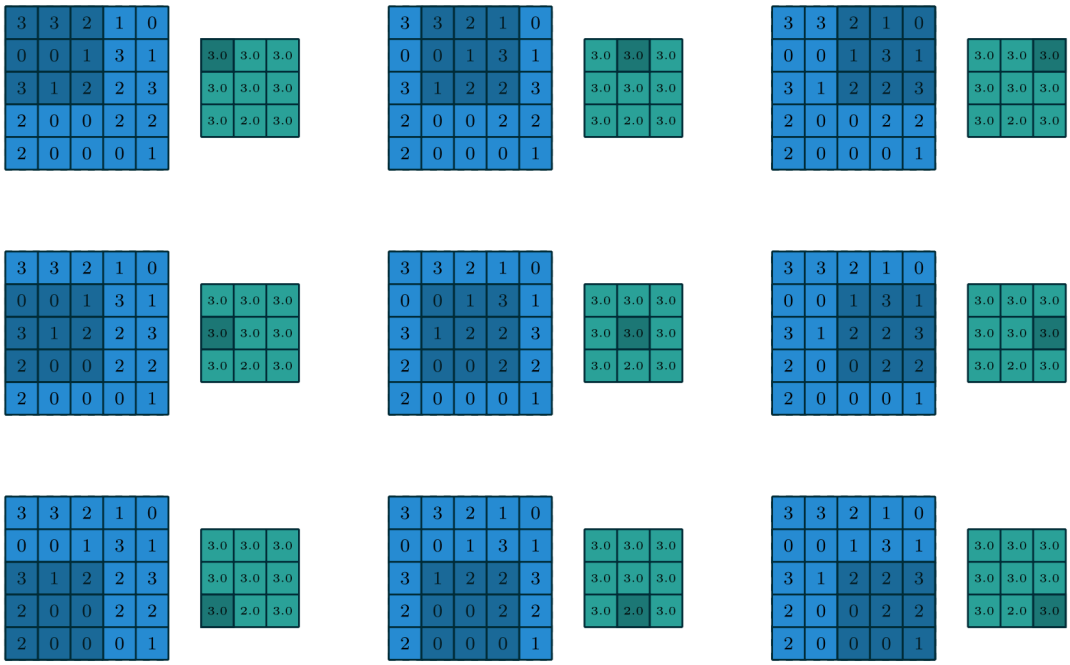


Figure 2.6: Application of max pooling on 5x5 matrix with a 3x3 kernel, stride 1, yielding a 3x3 result (Dumoulin and Visin, 2018). In the top left, the algorithm starts. The numbers in the blue matrix are processed with a 3x3 window. The maximum value is chosen for each submatrix and written to the smaller green matrix. For the first iteration, this result is thus 3. Then the algorithm shifts the kernel with a stride of one, and the procedure starts again. (Graphic: Dumoulin and Visin 2018)

**Feature Extraction** Using the described down-sampling encoding strategies, specific features of an image can be extracted. One of the advantages of convolutional neural networks is that the feature extraction is location invariant, meaning that the models can learn specific features, disregarding their location in the image (Zhang et al., 2021). A visual intuition can be observed in Figure 2.7.

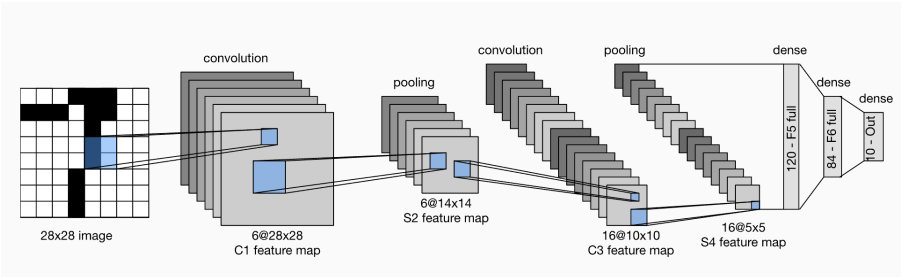



Figure 2.7: Architecture of LeNet (Lecun et al., 1998), the first convolutional neural network. (Graphic: Zhang et al. 2021)

**Further Development:** Over the years, using convolutions on two-dimensional data has been kept, and new architectures emerged. A further landmark was the fully convolutional network, first proposed by Shelhamer et al. (2014). Fully convolutional neural networks consist of only convolutional layers. A significant advantage was that computation speeds increased.

#### 2.2.2.3 U-Net

One example of a fully convolutional neural network used for semantic segmentation is the U-Net. The U-Net architecture - initially introduced by Ronneberger et al. (2015) - is a modified version of fully convolutional networks. U-Nets were first used for biomedical image segmentation and later adapted to different image segmentation tasks, such as shape detection in autonomous vehicles (Giurgi et al., 2022; Sugirtha and Sridevi, 2022; Tran and Le, 2019) or landcover detection in remote sensing (Solórzano et al., 2021; Xie et al., 2022; Giang et al., 2020; Wang et al., 2022).

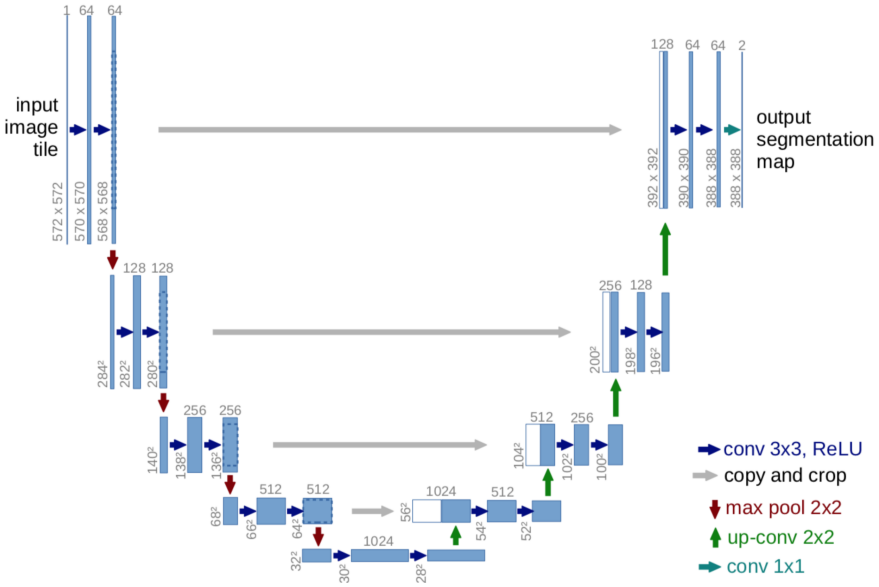



Figure 2.8: The figure shows a diagram of the U-Net architecture as proposed by Ronneberger et al. (2015). (Graphic: Ronneberger et al. 2015)

**Architecture:** Why is it called U-Net? Because their architecture is "U" shaped (Figure 2.8). As Figure 2.8 shows, all the down-sampling on the left-hand side (convolutions followed by max pooling) show the encoding of information. With these operations, the spatial dimension of an image is reduced, encoding the information into a subspace. At the bottom, the information is encoded, and all of the spatial information is lost. On the right-hand side, the information decoding happens through deconvolution (Ronneberger et al., 2015). In these steps, the model unfolds the information into the original dimensions, finally presenting a pixel-wise probability, representing the probability $P$ of a pixel $p$ belonging to a class $c$ (Chollet, 2017).

**Deconvolution:** In order to cast the values of an encoded vector representation into a higher dimensional matrix, deconvolution is applied (Dumoulin and Visin, 2018). There are different types of deconvolution (some authors call it transposed convolution) operations (Dumoulin and Visin, 2018). Deconvolution does the exact opposite of the convolution operation, creating a matrix with more dimensions from a small matrix (Figure 2.9).

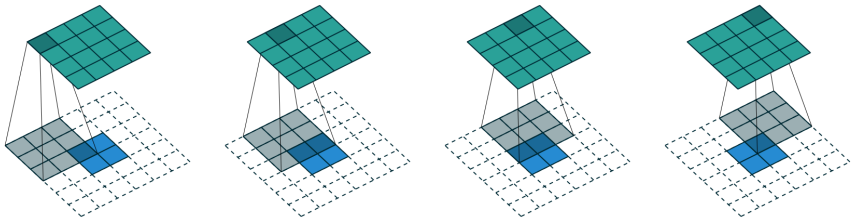


Figure 2.9: Deconvolution, applying a 3x3 kernel on a 2x2 matrix, yielding a 4x4 result (Dumoulin and Visin, 2018). (Graphic: Dumoulin and Visin 2018)

#### 2.2.2.4 Transformer

The following section provides a basic intuition of the core element of the Transformer architecture, the attention mechanism, followed by an introduction to the Swin Transformer architecture.

**Overview:** In recent years, a new neural network architecture called Transformer (Vaswani et al., 2017) has become the model of choice in natural language processing (NLP) (Dosovitskiy et al., 2020). In NLP tasks, the Transformer successfully replaced the former state-of-the-art (SOTA) architectures (recurrent neural networks (RNNs) and long short-term memory networks (LSTM)) with its more straightforward structure, higher computational speed, and better results on various benchmarks (Zhang et al., 2021; Chollet, 2017). In computer vision, the works of Dosovitskiy et al. (2020); Liu et al. (2021) have recently shown that the Transformer is a valid alternative to traditional CNN approaches. In semantic segmentation, Dosovitskiy et al. (2020) and Liu et al. (2021) have outperformed current SOTA methods on the ADE20K dataset.

**Attention Mechanisms** Attention mechanisms are the core of the transformer architecture (Vaswani et al., 2017; Zhang et al., 2021). Attention mechanisms allow the decoder to use the most relevant parts of the input sequence flexibly, by a weighted combination of all the encoded input vectors, with the most relevant vectors being attributed the highest weights (Chollet, 2017).

**Self-Attention:** We briefly focus on a text example similar to Chollet (2017, p.1369) to get an intuition. Text can be represented as an n-dimensional vector. Therefore, every word in a text sequence represents a distinct token. For example, the sequence "Bank in the park" has four tokens (*"Bank", "in", "the", "park"*). Each of these tokens can be represented/encoded numerically. Numerical encoding allows each word to have some meaning. For example, the word "Bank" might have a different meaning in another sentence (e.g., "John robs a bank"). Herefore, it would be helpful if the tokens in a sentence gave each other some context. Giving meaning can be achieved by reweighing each token by a specific weight. It is done by multiplying each token with all other tokens in the sequence and then normalizing the results. These results we now call weights. Finally, each token is multiplied by the respective weight, and all these products are added into new, weighted tokens with added context (Chollet, 2017). Although in CV, the attention mechanism has to be applied to one more dimension, the principle stays the same.

**Multi-Head Attention:** In order to have a system that can learn numerous relationships, it is possible to expand self-attention to multi-head attention. The said expansion adds flexibility to the system and prevents overloading a single attention mechanism. Adding this flexibility is achieved by stacking multiple "query," "key", and "value" (Figure 2.10, "$\mathbf{W}^Q$", "$\mathbf{W}^K$", "$\mathbf{W}^V$") matrices in the system. With the said stacking, the attention calculation is done in parallel. The computational performance of the system remains the same and does not decrease (Vaswani et al., 2017; Zhang et al., 2021; Chollet, 2017).

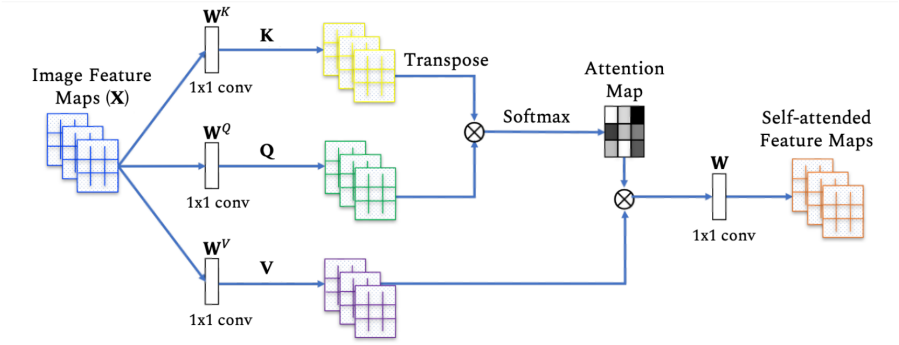



Figure 2.10: This figure shows one possible visual description of self-attention mechanisms as shown in Khan et al. (2021). (Graphic: Khan et al. 2021)

#### 2.2.2.5 Swin Transformer

**Overview:** Liu et al. (2021) introduced the Swin Transformer (Swin stands for **S**hifting **Win**dows), a new general purpose backbone for CV. The advantage of the Swin Transformer is that it can deal with large images within reasonable computation time. A major limitation that the pre-decessing ViT (Vision Transformer, Dosovitskiy et al., 2020) could not handle since the computational complexity increased quadratically with image size (Figure 2.11) (Liu et al., 2021). Moreover, the Swin Transformer uses location embeddings to account for pixel locations when calculating the attention matrices (Liu et al., 2021).

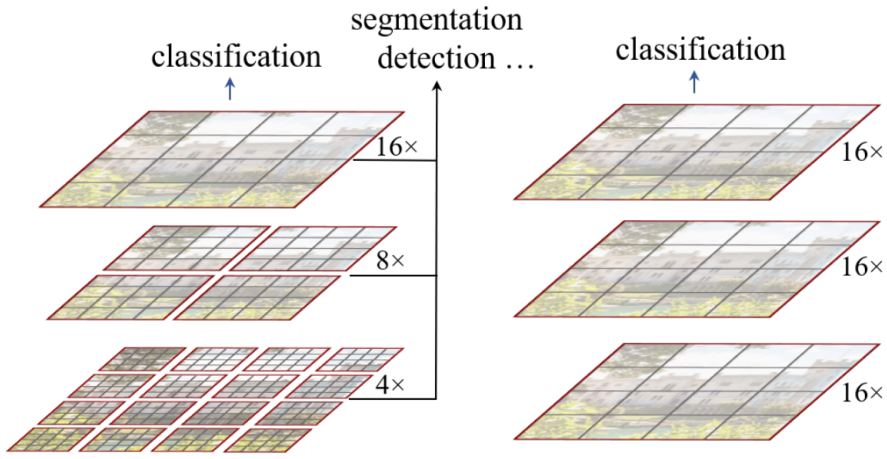



Figure 2.11: Different patching algorithms. On the left, the patching was proposed by Liu et al. (2021) for the Swin Transformer (SwinT). On the right is the patching proposed by Dosovitskiy et al. (2020) for the Vision Transformer (ViT). (Graphic: Liu et al. 2021)

**Architecture:** Swin Transformer takes an input image and splits it into non-overlapping patches (Figure 2.12), which then act as tokens. For each of these tokens, self-attention is calculated. Then, the patches are merged in the following steps, reducing the number of tokens and increasing the field of view for the attention calculation. This token-merging operation is applied in four stages (Figure 2.13).

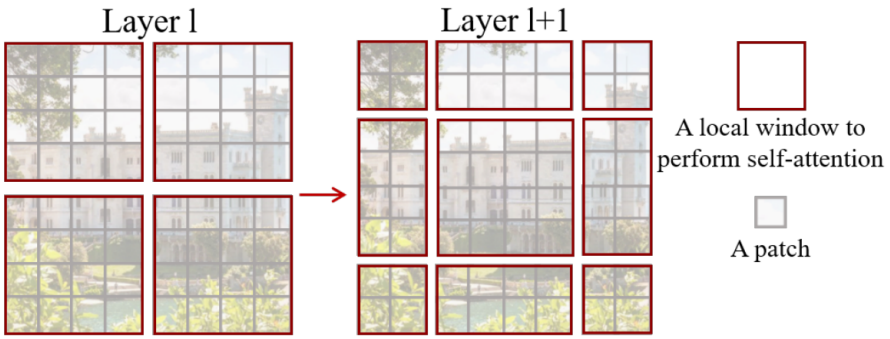



Figure 2.12: Graphical description of how the Swin Transformer approaches patch partition and illustrates how the shifted window approach operates in more detail (Liu et al., 2021). (Graphic: Liu et al. 2021)

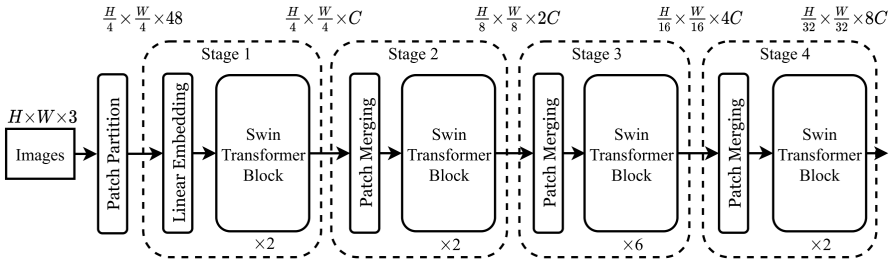



Figure 2.13: This figure shows an overview of the Swin Transformer (tiny version) architecture. More details can be found in Liu et al. (2021). (Graphic: Liu et al. 2021)

.

**Attention-Heads:** As mentioned in Section 2.2.2.5, Swin Transformer architecture can be used for various tasks. Depending on the task, there are a large variety of different algorithms to calculate attention. These algorithms are called attention-heads. The two most notable connections with the Swin Transformer are *uperhead* and *mlphead*. Please refer to Chapter 4 for a detailed description of which attention heads were used for this thesis.

### 2.2.3 Loss Functions

Another essential component of deep learning is the loss function. Loss functions play a vital role in model training (Goodfellow et al., 2016; Chollet, 2017; Zhang et al., 2021). There are a large number of different loss functions for varying tasks. Essentially, loss functions to control how the deviation between a model prediction ($\hat{y}$) and the ground truth label ($y$) is computed. The deviation between the prediction and the label is the target that the neural network aims to optimize. In the backward pass, the respective weights in a model are ideally updated so that the model increases its performance on the prediction tasks. For this thesis, the following existing loss functions were used.

#### 2.2.3.1 Binary Cross Entropy (BCE)

The binary cross entropy loss is one of the most common loss functions for classification tasks. Since semantic segmentation is a pixel-level classification, the binary cross-entropy function works well for binary classification tasks (Jadon, 2020; Yi-de et al., 2004). According to Jadon (2020), the BCE loss function belongs to the category of distribution-based loss functions. Mathematically, the BCE loss function $L_{BCE}$ is defined as:

$$L_{BCE}(y, \hat{y}) = -(y \cdot log(\hat{y}) + (1 - y) \cdot log(1 - \hat{y}))$$

where $y$ is the ground truth label and $\hat{y}$ is the prediction.

#### 2.2.3.2 Focal Tversky Loss

Jadon (2020) mentions the Focal Tversky Loss as a region-based loss function. Abraham and Khan (2019) proposed the Focal Tversky Loss as a generalized focal loss function based on the Tversky Index ($TI$), which specializes in difficult classification examples. With Focal Tversky Loss, it is possible to weigh down easy cases and focus on hard cases (Jadon, 2020; Abraham and Khan, 2019). Mathematically, Focal Tversky Loss ($L_{FT}$) is defined as:

$$L_{FT} = \sum_{c} (1 - TI_c)^{\gamma}$$

where $\gamma$ is a scaling factor in the interval $[1, 3]$ and $TI$ is the Tversky Index which is defined as:

$$TI(y, \hat{y}) = \frac{y \cdot \hat{y}}{y \cdot \hat{y} + \beta \cdot (1 - y) \cdot \hat{y} + (1 - \beta) \cdot y \cdot (1 - \hat{y})}$$

where the $\beta$ coefficient is used to weigh false positives (FP) and false negatives (FN).

#### 2.2.3.3 Dice BCE Loss

This loss function originates from the combination of a widely used metric that calculates the similarity between two images (Dice coefficient) and a modified version of the binary cross entropy loss mentioned above (Jadon, 2020; Taghanaki et al., 2019). According to Taghanaki et al. (2019), this function has been proposed to cope with imbalanced class labeling. Jadon (2020) states that the pure Dice Loss belongs to the region-based loss functions, whereas BCE Loss belongs to the distribution-based loss. Jadon (2020) thus puts Dice BCE Loss into a new category, called compound loss, and calls the loss function itself Combo Loss. Mathematically, Dice BCE (or Combo Loss) $L_{DBCE}$ is defined as:

$$L_{DBCE} = L_{Combo} = \alpha L_{m-BCE} - (1 - \alpha) DL(y, \hat{y})$$

where $\alpha$ controls the contribution of the $L_{m-BCE}$ and $DL$ components respectively and $L_{m-BCE}$ is the modified version of the BCE loss function:

$$L_{m-BCE} = -\frac{1}{N}\sum_{i} \beta(y - log(\hat{y}) + (1-\beta)(1-y)log(1-\hat{y})$$

where $\beta \in [0, 1]$ controls the level of model penalization for false positives (FP) and false negatives (FN), and $DL$ is the Dice Loss defined as:

$$DL(y, \hat{y}) = \frac{2y\hat{y} + 1}{y + \hat{y} + 1}$$

## 2.3 Deep Learning Approaches in Map Generalization

In map generalization, more profound efforts using deep learning to automate the challenging interplay of operators, rules, or constraints mentioned in Chapter 1 have been undertaken recently. Thereby, deep learning approaches produced a new angle on the task while heralding a paradigm shift. According to Sester et al. (2018), the approaches addressed are mainly applied in vector space. However, there are also methods using rasterized representations. The thesis at hand will pursue rasterized image representations of the spatial scene.

**First Attempts:** Sester et al. (2018) were among the first to apply deep learning models in map generalization using image-based modeling. In the study, the researchers applied a simplified U-Net network architecture. Sester et al. (2018) found that pixel-wise comparison of the correct predictions cannot capture the generalization effect. They state that instead, using the IoU leads to more meaningful accuracy values. Also, visual inspection showed that the network has learned to simplify buildings at different scales, but some characteristic features are not necessarily preserved, and minor irregularities may appear. In a subsequent study of the same group, they compared various architectures and found that a derivation of the U-Net architecture, named residual U-Net, performed best (Feng et al., 2019). The main limitation was that some predictions showed a shortcoming regarding the edges of a building that appeared to be more wobbly and distorted in the prediction output (Feng et al., 2019; Kang et al., 2020).

**Possible Data Models:** Touya et al. (2019) mentions that in connection to road data, there are ways of including metadata in an input image to a neural network by adding a dimension (or channel) to the image (section 4.4.2). Similar to RGB data, contextual data could be passed into a neural network by extending single-channel images to multi-channel images. In remote sensing, multi-channel images are often used to enrich the data with geo-relevant context (Ge et al., 2022). Fu (2022) mentions that pursuing a multi-channel data model approach in map generalization could be worthwhile.

**Location Encoding:** Touya et al. (2019) argues that using image-based maps and CNNs could cause a loss of critical spatial information. Moreover, according to Touya et al. (2019) and Courtial et al. (2021), the topological relations inherent in maps need to be sufficiently represented; otherwise, a loss of cartographic quality may. Therefore, it could be essential to consider pixel location encoding closely. A methodological overview of various approaches to pass location embeddings to neural networks recently published by Mai et al. (2022) is noteworthy. This paper shows how two-dimensional location information can be embedded into a high-dimensional vector using a location encoder. The fact that

this information can be stored as a vector allows neural network models downstream to learn this information much more straightforwardly than any other attempt to pass location information to a model. However, Mai et al. (2022) also mentions that the research has yet to develop to a satisfying extent for vector polygon data, which would be crucial for building generalization. The main issue was that the embedding techniques only worked for polygons if there were no holes or overlaps in the vector geometries. Finally, Mai et al. (2022) points out that for rasterized data, the question remains on *how location encoding can be embedded into raster data and what the benefits are.* A method to possibly solve this issue could be the attention mechanisms in the Swin Transformer architecture as introduced in Section 2.2.2.5.

## 2.4 Research Gaps

The outlined related work allows the identification of the following research gaps:

- Account for pixel location embedding in image-based map data
- Apply attention mechanisms, varying receptive field sizes, and location encoding to map generalization by using a Transformer-based model architecture
- Augment map image data with contextual information channel and introduce new data models to change the way how data gets passed into a computation model
- Develop a robust method for quantitative and qualitative evaluation of map generalization

## 2.5 Research Objectives

In order to address the above-stated research gaps, the planned thesis contributes to filling these gaps by pursuing the following research objectives (ROs):

**RO.1** Utilize the work of Feng et al. (2019) as baseline results

**RO.2** Implement Swin Transformer architecture as described in Section 2.2.2.5 (Liu et al., 2021; Xu et al., 2021)

**RO.3** Perform experiments to compare the performance of both computation models (Swin Transformer & U-Net) and the training of these computation models with new data models quantitatively. More specific research questions regarding the single experiments can be found in Section 4.8

**RO.4** Evaluate performance qualitatively

# Chapter 3

# Data

This section gives an overview of the data, the data sources, the spatial extents, and extent-specific statistical information. The data used are topographic map material.

## 3.1 Data Sources

**Topographic Maps:** Topographic maps are detailed maps that depict natural and artificial features on the earth's surface. They typically have scales ranging from 1:5'000 to 1:1'000'000 and are used to accurately represent the terrain through contour lines, elevation points, water bodies, roads, buildings, towns, and other technical features such as boundaries, power lines, and railroads. These geographic objects are represented on the map using a cartographic symbolization system appropriate for the map's scale (Spiess et al., 2002).

**Thesis Specific Data:** The thesis used data from two different sources, namely *OpenStreetMaps (OSM)* and *swisstopo*. The *OSM* data was initially used in the work of Feng et al. (2019); Sester et al. (2018) and then made available for use in this thesis. This thesis marks the first time the *swisstopo* data are used in connection with the application of deep learning to automate the cartographic generalization of buildings. For both sources, the 1:10'000 and 1:25'000 scales were used. Initially, the data is stored as vector files. Then, the vector data was rasterized into images. These images can be observed in Figures 3.1 and 3.2, which show the two map extents from the locations Stuttgart (DE) (*OSM* data) in Figure 3.1 and Zürich (CH) (*swisstopo* data) in Figure 3.2.

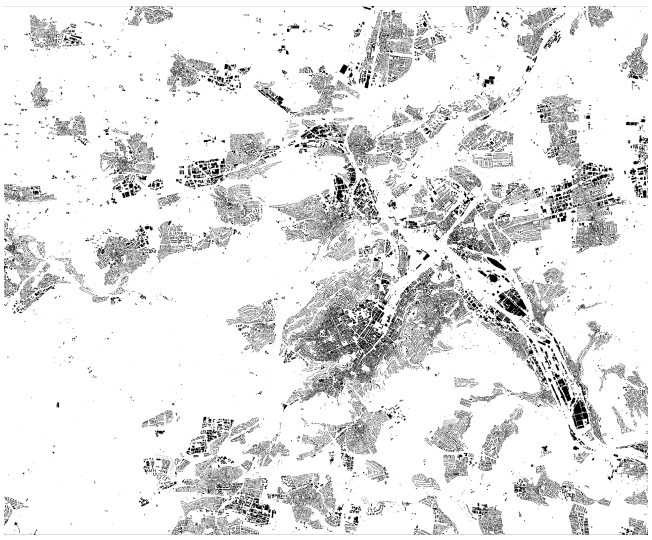


Figure 3.1: The image shows a map extent image of the *OSM* data around the urban area of Stuttgart, with peri-urban and rural settlements. *OSM (1:10'000)*, size: 42'800x35'000 pixels, number of buildings: 89048

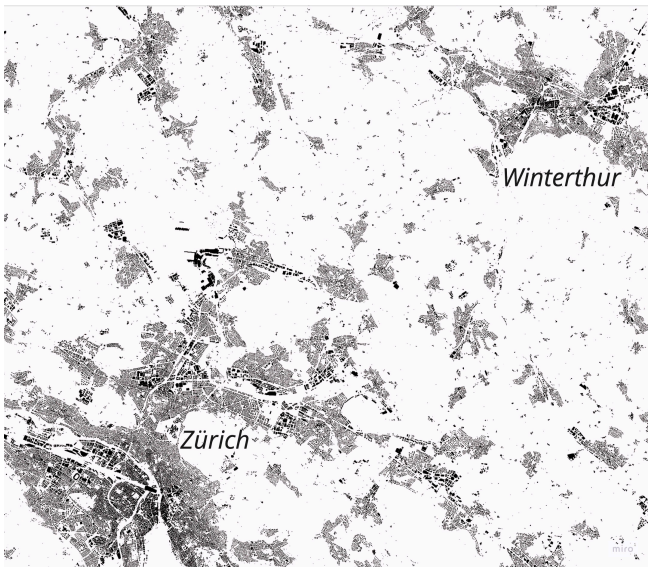



Figure 3.2: The image shows a map extent of the *swisstopo* data around the two Swiss cities, Zürich (bottom left, southwest) and Winterthur (top right, northeast). The countryside between the two cities shows periurban to rural settlements. *swisstopo (1:10'000)*, Size: 46'001x40'001, number of buildings: 72'096

**OSM Data Preparation:** The *OSM* data used in Feng et al. (2019) could be accessed. The data were stored as a .png file for each scale. Each of these files shows the same extent on a different scale. All map elements except building footprints were removed by Feng et al. (2019); thus, the data shows a binary map of buildings. These files were then processed to patches in Python as described in Section 4.3.2.

**Swisstopo Data Preparation:** Swisstopo (Dr. Roman Geisthövel) provided map material for all the buildings of Switzerland stored in vector format in the 1:10000 and 1:25000 scales. Since Dr. Yu Feng provided access to the data and code used in Feng et al. (2019), the swisstopo dataset was

rasterized, matching the resolution mentioned in 3.1 and the procedure described in Section 4.3.2.

## 3.2 Comparison of the Spatial Extents

The following section shows a statistical comparison of the spatial extents shown in Figures 3.1 and 3.2. The chosen extents were selected to have approximately the same amount of pixels (Zürich size in pixels 46'001x40'001, Stuttgart size in pixels 42'800x35'000). The reason why the *swisstopo* dataset is larger than the *OSM* dataset originates in the condition to have approximately the same size of pixels but also approximately the same amount of buildings (89'048 buildings in the *OSM* extent and 72'096 buildings in the *swisstopo* extent). Note that the *swisstopo* extent is more extensive in pixels and, at the same time, contains fewer buildings, whereas the *OSM* data is more minor in pixels but contains more buildings.

**Building Pixels vs. Background Pixels:** Comparing the portion of building pixels present in the two scenes shows that there is a difference of roughly 1.2% in building densities (Figure 3.3). With 7.1%, the map extent around Zürich shows a lower density of buildings compared to the building density in Stuttgart (8.3 %).

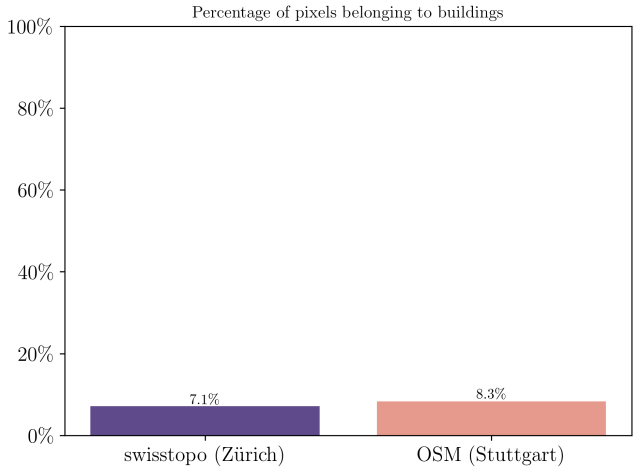



Figure 3.3: The figure shows the ratio of pixels that belong to buildings for the swisstopo (Zürich) dataset and the OSM (Stuttgart) dataset. The y-axis represents the percentage of pixels belonging to buildings. In order to show that the density of buildings is moderately low, the y-axis ranges from 0-100%. The x-axis represents the respective data set.

**Building Area Distribution:** Comparing the distribution of building area pixel count reveals fundamental differences in the distribution of building areas (number of pixels that make up a building footprint area) between the input and the target scale for each data source. For example, in Figure 3.4, we can observe that the count distribution of the target map pixels shows that the swisstopo building areas are generally larger (counts of buildings with more pixels are higher than in the input image). On the other hand, in Figure 3.5, the distribution's visual comparison shows that the OSM buildings' sizes are similar in the input and target maps.

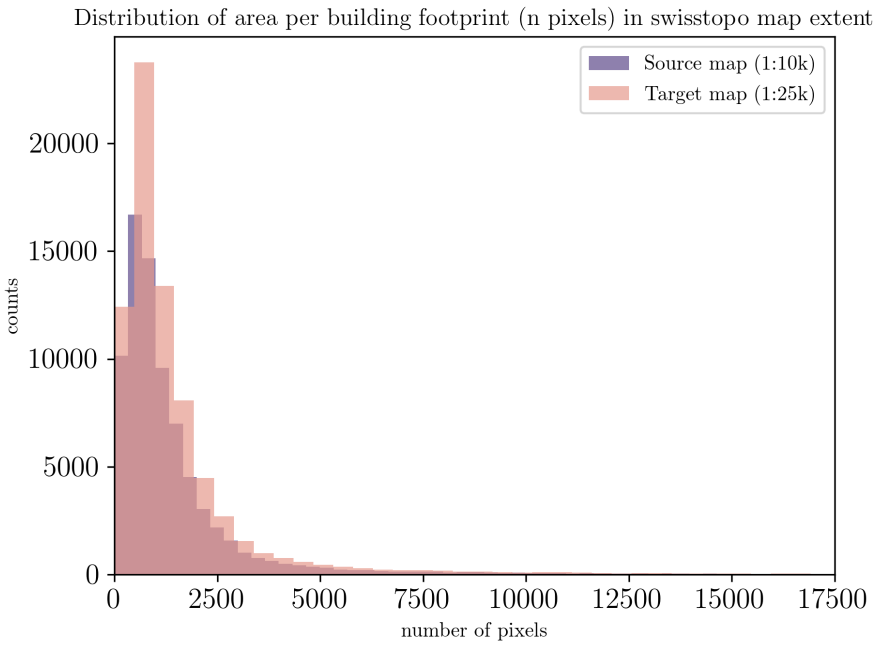



Figure 3.4: The histogram shows the distributions of the area per building footprint in the number of pixels for the swisstopo map extent. In this histogram plot, the x-axis represents the range of areas, and the y-axis represents the frequency of the areas. The histogram bars show the areas' distribution, with each bar's height representing the number of data points that have areas within that bin range. The purple histogram shows the source map, whereas the coral-colored histogram shows the target map.

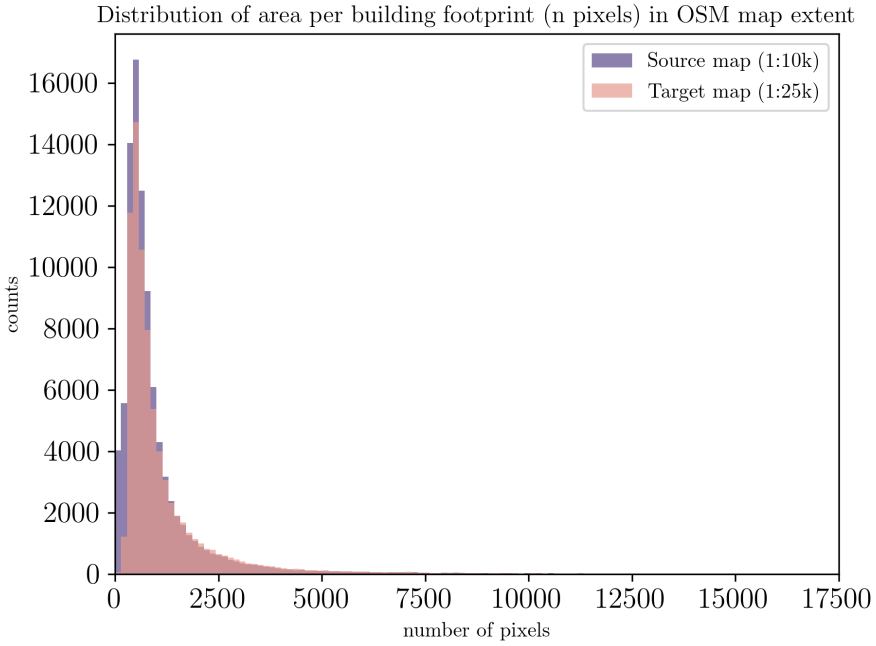



Figure 3.5: The figure shows the distribution of the area per building footprint in the number of pixels for the OSM map extent. In this histogram plot, the x-axis represents the range of areas, and the y-axis represents the frequency of the areas. The histogram bars show the areas' distribution, with each bar's height representing the number of data points that have areas within that bin range. The purple histogram shows the source map, whereas the coral-colored histogram shows the target map.

**Building Perimeter Distribution:** Comparing the distribution of building perimeter pixel count reveals fundamental differences in the distribution of building perimeters (the number of pixels that make up the perimeter of a building) between the input and the target scale for each data source. Figures 3.6 and 3.7 show the distribution of building perimeters in the input and target maps. In Figure 3.6, the target map has more buildings with larger perimeters than the input map, suggesting that there might be a rule in the swisstopo dataset that requires buildings to be a specific size. However, in 3.7, the difference between the two maps is not as significant, which means that the building sizes in both maps are more similar.

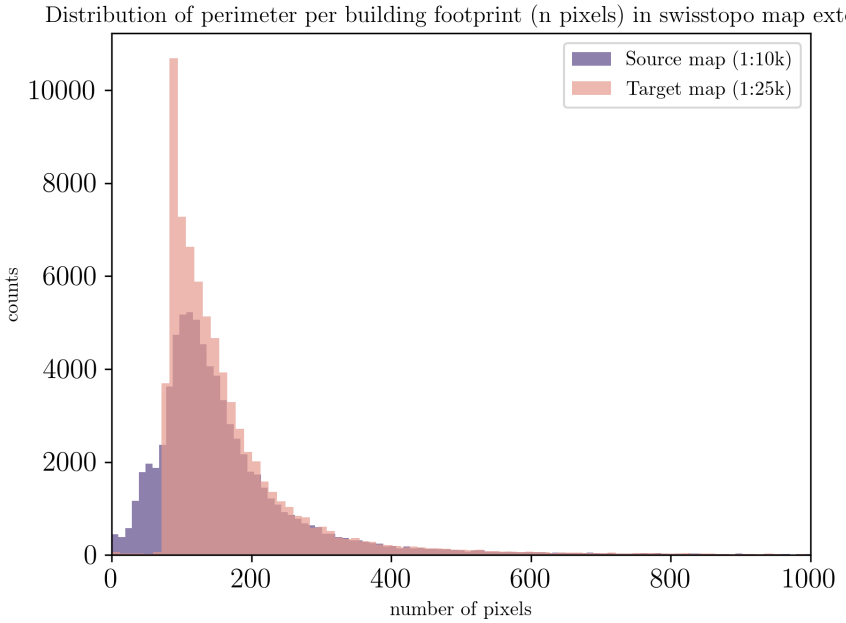



Figure 3.6: The figure shows the distribution of the perimeter per building footprint in the number of pixels for the swisstopo map extent. In this histogram plot, the x-axis represents the range of perimeters, and the y-axis represents the frequency of the perimeters. The bars in the histogram show the distribution of the perimeters, with the height of each bar representing the number of data points that have perimeters within that bin range. The purple histogram shows the source map, whereas the coral-colored histogram shows the target map.

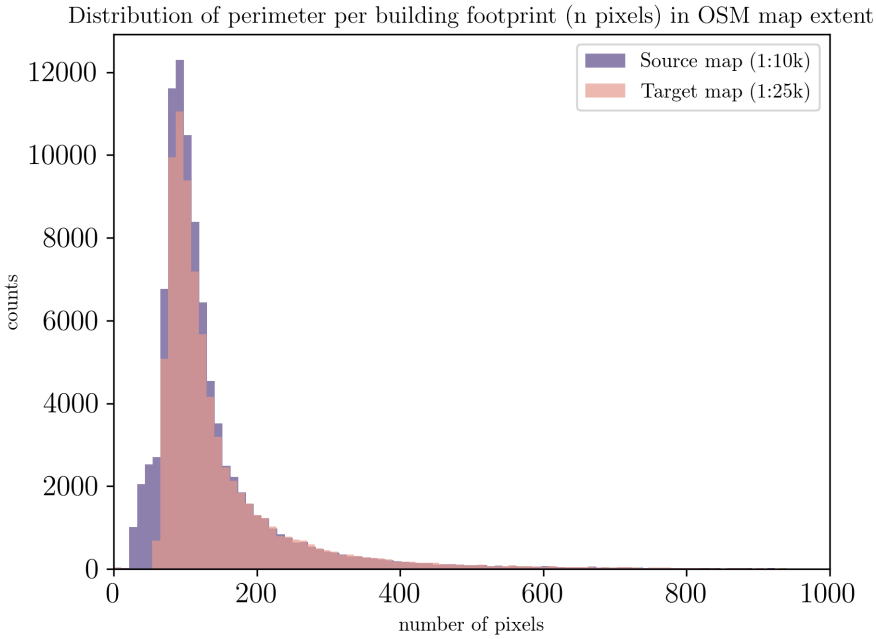



Figure 3.7: The figure shows the distribution of the perimeter per building footprint in the number of pixels for the OSM map extent. In this histogram plot, the x-axis represents the range of perimeters, and the y-axis represents the frequency of the perimeters. The bars in the histogram show the distribution of the perimeters, with the height of each bar representing the number of data points that have perimeters within that bin range. The purple histogram shows the source map, whereas the coral-colored histogram shows the target map.

**Dataset Complexity:** For both sources, the Intersection over Union (IoU) (check Section 4.6.1) was calculated to determine how the input (1:10'000) and target (1:25'000) map extents differ. For example, in Figure 3.8, it can be observed that the IoU between the input and target for the *swisstopo* data is smaller compared to the IoU for the *OSM* data. Since there is more change from the input to the target map in the *swisstopo* map extent, we assume more operators are inherently present in the data. If more operators are present in the data, we could infer that the data is more challenging to learn for a model. Thus, we call the *swisstopo* dataset to be the *more complicated task*, whereas the *OSM* dataset is the *less complicated task*.

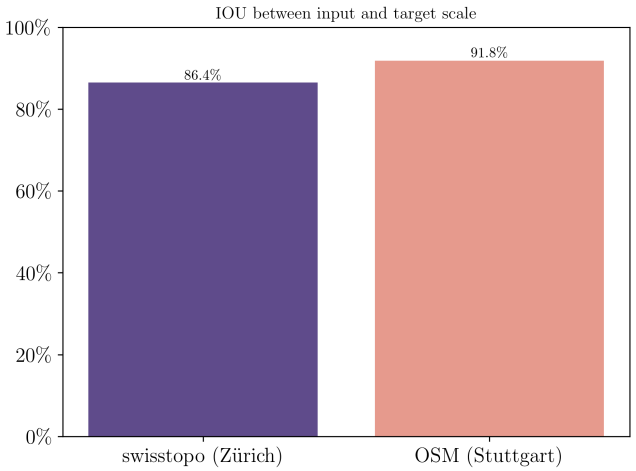



Figure 3.8: The figure shows the comparison of the similarity between two map datasets (Swisstopo from Zürich and OSM from Stuttgart) by showing their IoU values on the y-axis. The x-axis represents the different datasets. The higher the IoU value, the more similar the maps are; the lower the IoU value, the more different the maps are.

**Operators:** The two data sets also differ in terms of how they were generalized. On the one hand, professional cartographers generalized the *swisstopo* dataset. On the other hand, the *OSM* data which was also used in Sester et al. (2018) and Feng et al. (2019), was presumably generalized with the CHANGE software (Powitz, 1993). Therefore, a fundamental difference in the two data sets regarding the number of operators for a single building can be assumed. For example, according to Sester et al. (2018), the *OSM* data contains operators such as simplification and aggregation, whereas displacement and multiple operator combinations are not present in data generalized with CHANGE (Powitz, 1993). On the other hand, since the *swisstopo* data was produced by cartographers, all possible operators with various combinations can be found in the data.

## 3.3 Extensive Data

For a specific experiment described in Section 4.8.1, a comprehensive swisstopo dataset was used. The aim was to thoroughly test the overall capabilities of the Transformer model by providing a massive dataset and checking how the Transformer handles it. Figure 3.9 depicts Switzerland, divided by a 30km x 30km grid. The red squares were the selected grid cells for the experiment. In order to capture as many different settlements and building types as possible, the grid cells were selected across Switzerland. The focus, however, lay on selecting grid cells with a comparably high density of buildings. As a result, the extensive dataset is approximately twenty-five times larger than the datasets used for the other experiments described in section 4.8.1.

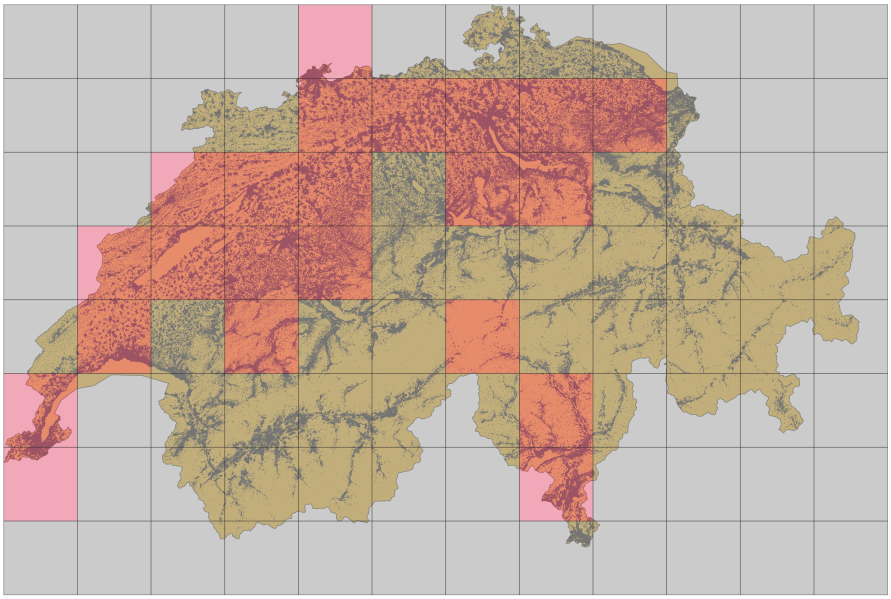


Figure 3.9: This figure exhibits a map of Switzerland divided into grid cells. The grid cells displayed in red symbolize the grid cells containing the buildings used for the experiments conducted on the extensive *swisstopo* data.

# Chapter 4

# Methodology

The following section covers all the relevant technical and procedural details. Table 4.1 provides a crisp overview of the key terms and their meaning discussed in the following.

Table 4.1: The glossary table provides an overview for better orientation in the overall workflow.

| Term | Content |
|---|---|
| **Computation Model** | U-Net (Section 2.2.2.3), Transformer (Section 2.2.2.4) |
| **Data Source** | swisstopo (Section 3.1), OSM (Section 3.1) |
| **Data Model** | single-channel (Section 4.4.1),<br>two-channel (random (Section 4.4.2), centered (Section 4.4.2)) |

## 4.1 Technical Setup

### 4.1.1 Hardware

The main workhorse for the computations done for the thesis was a MacBook (M1, 2020) with 16 GB of RAM. All of the more extensive computations, such as deep learning model training and large-scale data engineering, were performed on the UZH S3IT Science Cluster's[1] CPUs and GPUs (Tesla V100).

### 4.1.2 Software

Most of the visual GIS data exploration was done on the personal computer in QGIS (LTR Bialovieca 3.22). All data-related computations were done using Python 3 (3.9). For sequential workflows, JupyterNotebooks were used, whereas, for auxiliary code, such as definitions for object classes and functions, python files were used. A more detailed overview of used software packages can be seen in Table 4.2:

[1] `https://docs.s3it.uzh.ch/`

Table 4.2: Package overview for with a precise description of the used software.

| Task | Library (version) |
|---|---|
| **Data Engineering** | numpy (1.22.3), GDAL (3.5.0), matplotlib (3.5.1), sci-kit-image (0.18.3), open-cv (4.6.0.66), pandas (1.4.2), os, sys |
| **Deep Learning** | PyTorch (1.11.0), PyTorchLightning (1.5.8), TensorBoard (2.9.1) |

## 4.2 Computation Model Configurations

This section briefly describes which computation models were used to perform the experiments. As described in Chapter 2, the computation models chosen for this thesis are U-Net and Swin Transformer.

### 4.2.1 U-Net

This thesis used a vanilla U-Net model proposed by Ronneberger et al. (2015). With the vanilla architecture and an input image size of 224x224 pixels, the model contained roughly 30 million parameters.

### 4.2.2 Swin Transformer

The Swin Transformer model allows one to choose from a large variety of attention heads and four model architectures. For this thesis, the Swin Transformer with the setting

- attention head: *mlphead*
- architecture: *base*

was used for most of the experiments. Liu et al. (2021) proposed to use *uperhead* attention after having performed a set of experiments on the ADE20K dataset. However, for geospatial applications, a more lightweight, efficient attention mechanism called *mlphead* was proposed by Xu et al. (2021). Since cartographic generalization closely resembles the geospatial application from Xu et al. (2021), the *mlphead* attention mechanism was also used for this thesis. The architecture setting directly impacts the computation model's number of parameters (model complexity). The *base* architecture setting with the *mlphead* yielded roughly 130 million parameters. For the brute force experiments, the architecture parameter *tiny* was selected (yielding roughly 85 million parameters) to reduce training time.

## 4.3 Data

A detailed description of the data used for the thesis can be found in Chapter 3.

### 4.3.1 Data Preparation

The map data is initially stored as vector data for both data sources (Chapter 3). Then, using the GDAL library, the vector data were rasterized with a pixel size of 0.5x0.5 meters. After the

rasterization, the data were stored as images in binary format (0: background pixel, 1: building pixel). This format allows reading the images with Python and the seamless transformation between Python *arrays* and PyTorch *tensors*. Figure 4.1 provides an intuitive overview of the data processing used in the thesis.

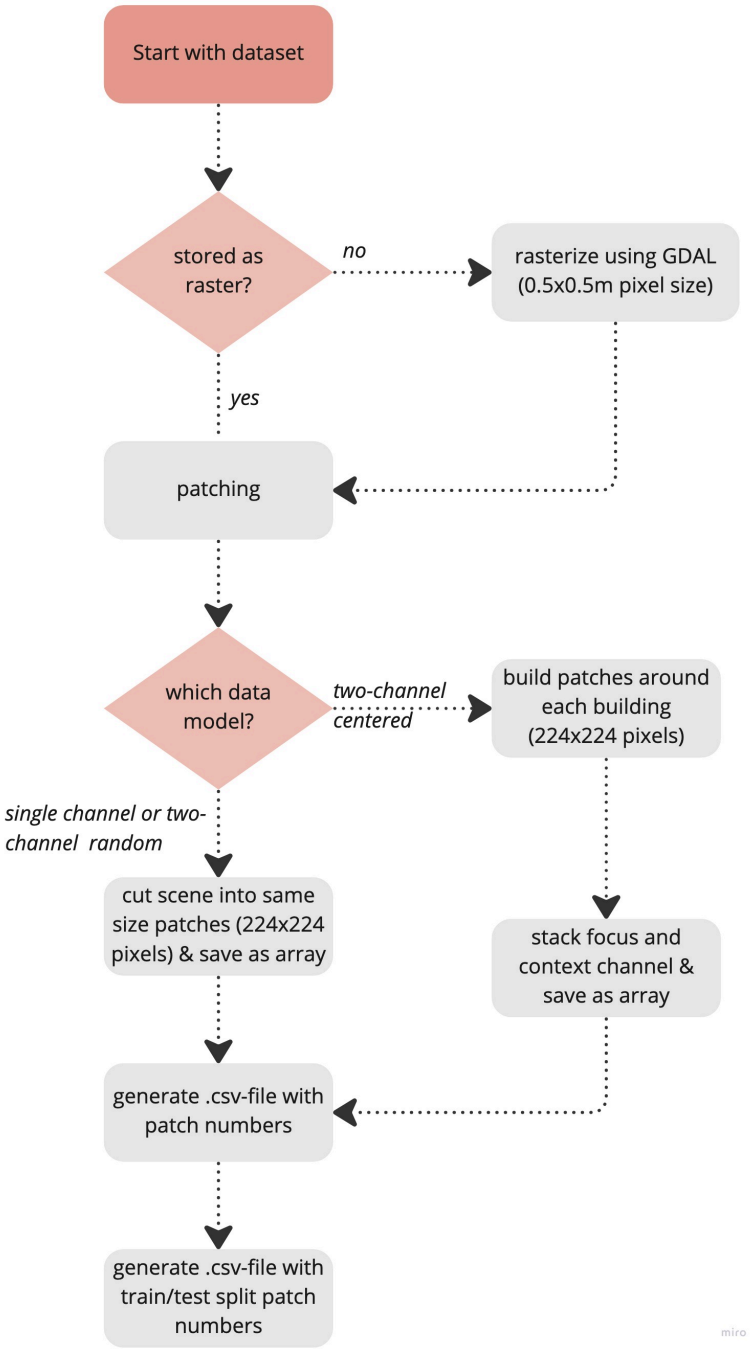



Figure 4.1: Detailed overview of how the map image data were pre-processed into smaller patches. The workflow describes the procedure used to generate the training data for all three data models used in the thesis experiments.

### 4.3.2 Patches

The data was fed to the computation model after cutting up the scenes depicted in Figures 3.1 and 3.2 into smaller patches of 224x224 pixels. For the single-channel (Section 4.4.1) and two-channel random (Section 4.4.2) data model, the patches were generated without any overlaps. For the two-channel

centered data model (Section 4.4.2), patches were generated differently (refer to Section 4.4.2). For all the patches, there exists an input and a target patch (Figure 4.2). The first (input) patch shows a map extent of the original scale (1:10000), whereas the second target patch shows the same map extent on a more minor (target) scale (1:25000).

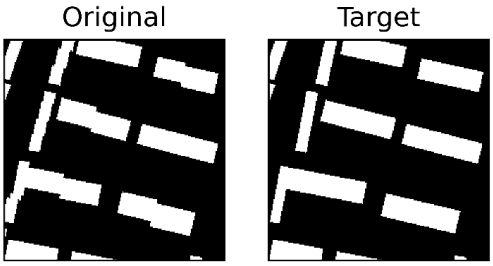



Figure 4.2: Original patch (1:10'000 scale) and target patch (1:25'000 scale) of the same size (224x224 pixel). The original patch shows a higher level of detail, whereas the target shows a generalized view of the map extent

## 4.4 Data Models

In the following, we discuss the different data models used to perform the experiments discussed in Section 4.7. Figure 4.3 provides a visual intuition of the data models used in the thesis.

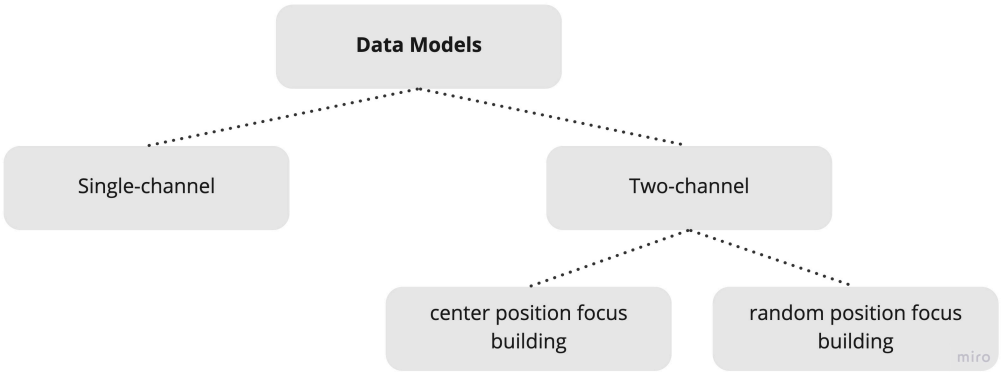



Figure 4.3: Schematic overview of the three different data models. The main difference between the data models is the number of channels. Either there was one single channel containing all the information or two channels. The two channels were split according to information content. The focus building was stored in the first channel, and the context buildings were stored in the second channel. Both two-channel data models differ in how the focus building is placed. The focus building is always placed in the center using the first data model. The focus building is placed randomly in the image in the second data model.

The works of Sester et al. (2018); Feng et al. (2019); Kang et al. (2020) use single-channel images of maps (Chapter 2). The single-channel data model served as the base for the thesis at hand. Developing the ideas from Touya et al. (2019) further led to the introduction of two-channel data models, first discussed by Fu (2022). The latter was achieved by adding contextual information to a map extent by adding a tensor dimension to the input image, yielding two-channel images rather than one-channel images. A conceptual schema can be observed in Figure 4.4.

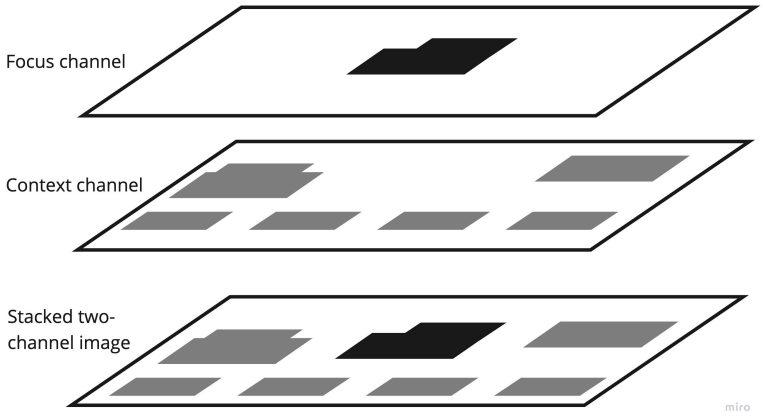



Figure 4.4: Schematic overview of the two-channel data model, stacking the single channels into one single image

### 4.4.1 Single-Channel Data Model

The single channel data model is the original data model proposed by Feng et al. (2019). The computation model is given a single patch with one channel with a predefined size (224x224 pixels in this work). With this approach, the model has to predict all the buildings visible in the given scene. Besides, all empty patches (no building pixels) were omitted during the data preparation. Furthermore, each patch had to contain at least 20 percent of building pixels.

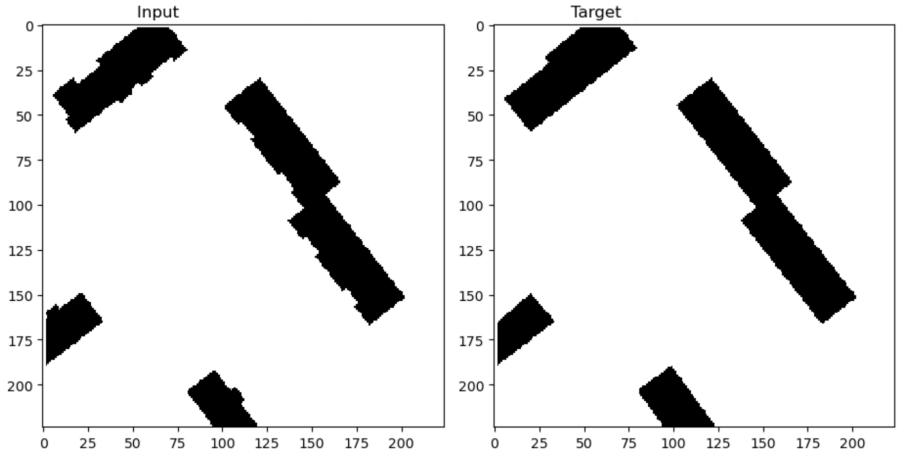



Figure 4.5: Single channel data model, input, and target patch. This is the same data model as proposed by Feng et al. (2019)

### 4.4.2 Two-Channel Data Models

In the two-channeled approach, the input data is split into two channels (Figure 4.4). One channel contains the focus building; the other contains the context (surrounding) buildings. In this approach, the model predicts only the focus building. There are two types of two-channeled data models: one with random placement of the focus building and one with the central placement of the focus building. The following describes the procedures used to process the data.

**Random Focus Building:** This approach randomly selects a building from a given input-target patch pair. This transformation happens on the fly whenever a patch is passed to the computational model. Again, the constraint was that the patches could not be empty nor contain less than 20 percent of building pixels.

**Algorithm 1** Pseudocode for generating input and target patches with random placement of focus building

**procedure** RANDOMSELECTION(input, target)
    Generate empty tensor ($t$)
    Label all buildings in input and target patch
    Calculate region properties, extract coordinates of pixels belonging to a building
    Generate masks for each building (one mask per building)
    Check which buildings are fully contained in patch
    **if** there are fully contained buildings **then**
        Calculate IoU between fully contained input building masks and target patches
        Create list of fully contained buildings (*primary list*)
    **else**
        Calculate IoU between randomly selected buildings in map
        Create list of partially contained buildings (*backup list*)
    **end if**
    **if** Length of *primary list* is larger than 0 **then**
        Select a fully contained building from *primary list* at random
        Add building to focus channel (dimension 0) of tensor $t$
    **else**
        Select partially contained building from *backup list* at random
        Add building to focus channel (dimension 0) of tensor $t$
    **end if**
    Subtract focus building from original scene
    Set result as context channel (dimension 1)
    Stack tensor with focus channel as dimension 0, context channel as dimension 1
    Pass stacked tensor to model
**end procedure**

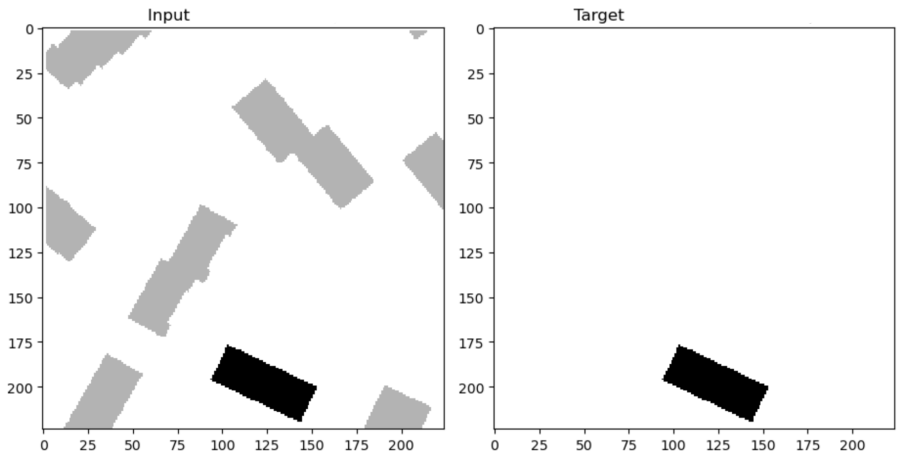



Figure 4.6: Example of a patch with two channels and random focus building placement. The focus building is shown in black, and context buildings are shown in gray. Note how the input patch shows both channels, whereas the target patch only shows the focus channel. Computation models trained with this data model have to predict only a single building at a time.

**Centered focus building:** In this approach, the focus building is always placed in the center of a patch. However, contrary to the random building placement, the patches are generated before passing them to the computation model. After the patches are generated, they are saved.

**Algorithm 2** Pseudocode for generating input and target patches with centered placement of focus building

```
procedure CenteredSelection(input, target)
    Select building from selected input map extent
    Generate bounding box and centroid around selected building
    Identify missing height and width pixels to generate patch of predefined size (e.g. 224x224 pixels)
    Identify top left of patch to generate
    Copy missing row and col pixels from original input map
    Set focus building in the center (dimension 0),
    Set other buildings in the context channel (dimension 1)
    Select building with most overlap from target map
    Generate array with corresponding input/target pairs
end procedure
```

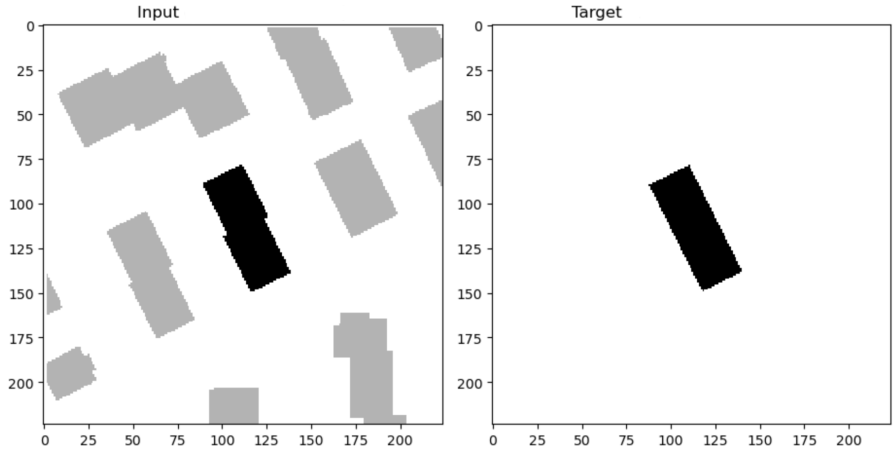



Figure 4.7: Example of a patch with two channels and centered focus building placement. The focus building is shown in black, and context buildings are shown in gray. Note how the input patch shows both channels, whereas the target patch only shows the focus channel. Computation models trained with this data model have to predict only a single building at a time.

### 4.4.3 Dataset size overview

As mentioned in Section 4.3.2, the patches were generated without overlaps for the single-channel and two-channel random data models. Therefore, the number of patches can be calculated by dividing the original image into 224x224 pixel patches. Conversely, the patches were constructed around each building for the two-channel-centered approach. Thus, the number of patches represents the number of buildings in the original image scenes. The exact numbers can be observed in Table 4.3. Moreover, in Table 4.3, the swisstopo dataset for the brute force experiments (extensive data) is significantly larger than the other datasets. Detailed information can be found in Section 4.8.1.

Table 4.3: Overview of the dataset sizes split by source and data model. The indicated number of patches represents the respective total amount of patches. The constraint on the ratio between the background and building pixels in each patch is ignored.

| **Source** | **Data Model** | **Number Patches** | **Patch Size [pixel x pixel]** |
|---|---|---|---|
| OSM (Stuttgart) | two-channel centered | 89'048 | 224 x 224 |
| OSM (Stuttgart) | two-channel random | 29'854 | 224 x 224 |
| OSM (Stuttgart) | single-channel | 29'854 | 224 x 224 |
| swisstopo (Zürich) | two-channel centered | 72'096 | 224 x 224 |
| swisstopo (Zürich) | two-channel random | 36'672 | 224 x 224 |
| swisstopo (Zürich) | single-channel | 36'672 | 224 x 224 |
| swisstopo (extensive data) | single-channel | ∼1'000'000 | 224 x 224 |

## 4.5 Deep Learning Training

The deep learning models were trained over a varying number of epochs. Usually, a model was trained past overfitting, logging the results every five epochs (Section 4.7). Thus, the point where the model was optimally trained could be obtained.

**Overfitting One Batch:** Whenever a new loss function or hyperparameter was applied, the model would be pushed into overfitting on one single batch to run a sanity check if any errors exist in the computation as a first step (Persson, 2020; Tobin, 2019).

### 4.5.1 Training, Validating, and Testing Data

The training and validation data for all the experiments performed were randomly selected using a 90% / 10% training/validation split. A priori, the test set was decoupled from this process. The test set was chosen to be 5% of the original data.

**Random Selection:** Both Sester et al., 2018 and Feng et al., 2019 have used an approach where the testing area was predefined as a small but coherent extent on the map. In this thesis, the test area per se does not exist, as the process was chosen to be truly random. Thus, all the testing patches have a random position in the original input map. Therefore, putting together the test set would lead to a random representation of the input map with random positions of the testing patches.

## 4.6 Evaluation Metrics

The following details describe the metrics logged during training and evaluate model performance. In this thesis, instance segmentation was applied to binary data. Therefore, the following confusion matrix for binary classification can be applied, as shown in Table 4.4.

Table 4.4: Binary classification confusion matrix

| | | **Ground Truth** | |
|---|---|---|---|
| | | positive | negative |
| **Prediction** | positive | True Positive (TP) | False Positive (FP) |
| | negative | False Negative (FN) | True Negative (TN) |

### 4.6.1 Intersection over Union (IoU)

The most important metric used in this thesis is Intersection over Union (IoU) or Jaccard-Index (JI). Essentially, the IoU measures the similarity of sets. In geometry, the IoU can be used to measure how similar geometries are. In the case where two geometries are perfectly congruent, the IoU is 1. If the shapes are incongruent, the IoU is 0. In CV, the IoU is often used to measure the similarity of images where a pixel-by-pixel comparison is applied. Recent works performing instance segmentation for geospatial applications such as Sester et al. (2018), Xie et al. (2022), or Giang et al. (2020), but also researchers from other fields (Liu et al. (2021) or Yang and You (2018)) have used the IoU to compare images to determine the difficulty of the learning tasks, or to evaluate the prediction performance. In this thesis, we want to train models, for instance, segmentation tasks, and compare the similarity of geometric shapes in images; therefore, the IoU is used. Mathematically, the IoU is defined as:

$$IoU(A, B) = JI(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Referring to the confusion matrix for binary classification, the IoU can also be defined as:

$$IoU = JI = \frac{TP}{TP + FP + FN}$$

### 4.6.2 Precision

Precision measures the ability of a binary classification model to predict the true positive cases. Mathematically, precision is the number of true positives divided by the number of true positives plus the number of false positives or:

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

### 4.6.3 Recall

Recall measures the number of true positive predictions made out of all positive predictions. Therefore, it can also be called *true positive rate*. The mathematical definition of recall is the number of true positives divided by the number of true positives plus the number of false negatives or:

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

### 4.6.4 F1-Score

The F1 score is the harmonic mean of precision and recall, taking both metrics into account in the following equation:

$$\mathrm{F1} = 2 * \frac{precision * recall}{precision + recall}$$

### 4.6.5 Test Set Evaluation

The model performance was evaluated based on the capability of a trained computation model to predict random examples from a test set. The model never saw specific examples from the test set during training. As mentioned in Section 4.6.1, various teams of researchers also used the IoU to evaluate the model performance. It is important to note that this measure was not explicitly developed to determine the results of cartographic generalization. Therefore this metric is merely a proxy for model performance, and a high IoU value does not necessarily indicate a good generalization. For the test set evaluation shown in Chapter 5, there are two types of IoU values reported:

- IoU (single case IoU): indicating the IoU between two specific images (Chapter 3) or two specific patches.
  - The IoU between an input and target (IPT-TGT) image shows how similar the two images are. The assumption is that if the IoU is lower, the scene is generally more difficult to predict since there is more change that the model has to predict.
  - The IoU between a target and prediction (TGT-PRED) image shows how similar the predicted image is to the target. In the ideal case, the predicted and target images have perfect congruency.
- mIoU (median IoU): indicating the median prediction performance of a model on the whole test set. This metric represents the median value of the pairwise comparison of all targets and all predictions in the test set.

## 4.7 Experimental Setup

### 4.7.1 Logging

During model training, the TensorBoard implementation provided in the PyTorchLightning framework was used for logging the training and validation loss and the evaluation metrics. The model was saved with the validation loss value every five epochs. When testing the model, the best model (lowest loss value) was selected for the predictions on the test set.

### 4.7.2 Loss Functions

The initial objective of the study was to investigate the impact of utilizing various purpose-built loss functions described in Section 2.2.3 on the outcomes of model training. However, initial experiments

with the brute force approach (see Section 4.8.1) led to the realization that the two different loss functions did not significantly change the model's prediction output. Therefore, the more straightforward binary cross-entropy (BCE) loss function was ultimately employed in the subsequent experiments.

## 4.8 Experiments

This thesis's approach toward applying deep learning in map generalization is exploratory. Therefore, several experiments were conducted to try and grasp interactions and influences between the manifold of parameters that can be changed in deep learning. As the set of combinations is large, and the possibilities of changing parameters are almost infinite, the following section provides a conceptual overview of the landmark experiments carried out in this thesis. The research objectives (ROs) listed in the following are more detailed parts of the main objective to run experiments (Section 2.5, RO.3).

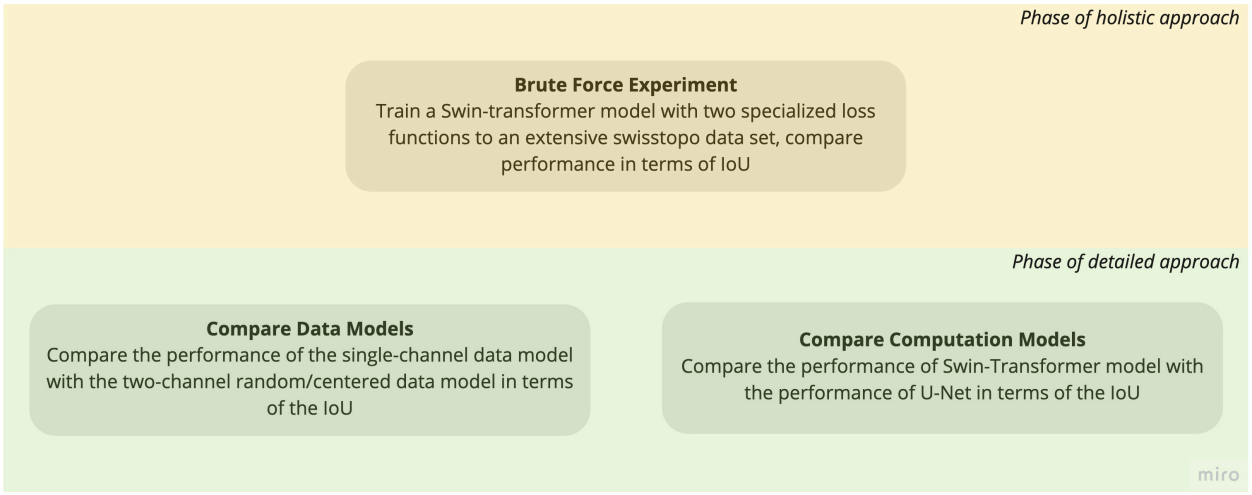



Figure 4.8: Conceptual experiment plan with two phases, first the holistic approach, followed by the detailed approach

### 4.8.1 Brute Force Approach

In the first step, the general capabilities of the Transformer computation model were thoroughly tested. The data for this experiment is described in Section 3.3. In this approach, the data model was chosen to be single-channeled, and therefore the same as used in Feng et al. (2019) and Sester et al. (2018). Two different, task-specific loss functions were tried: *Focal Tversky Loss*, and *Dice BCE Loss*. According to Vaswani et al. (2017), Dosovitskiy et al. (2020), Liu et al. (2021) and Xu et al. (2021) Transformers are data-hungry architectures, and therefore the training of a purpose-built Transformer model is challenging. A further demurral is that the semantic segmentation task used to perform cartographic generalization differs from usual semantic segmentation tasks. Therefore, pre-trained models could not be used. Attempting to split up this challenge into manageable parts, the following research questions (RQ) for this experiment were defined as follows:

**RQ.1** How does the model performance differ when the Transformer is trained with *Focal Tversky Loss*, or *Dice BCE Loss*?

**RQ.2** How is the performance compared to results from Feng et al. (2019)?

**RQ.3** How do training times scale?

**RQ.4** Is it possible to train a purpose-built Transformer model for automatic building generalization from the used training data?

### 4.8.2 Comparing Data Models

This set of experiments tried to capture any performance difference in the data models described in Section 4.4. The primary purpose was to determine whether the model training performance could be improved by combining two-channels (focus and context) and feeding them to the computation of a model context channel, compared to a single-channel model. Moreover, it was crucial to determine whether a random selection of the focus building would outperform the centered placement of the focus building. Therefore, the following research questions (RQ) were defined:

**RQ.1** What is the effect of random vs. centered building selection for the focus channel on the training?

**RQ.2** What is the effect of random vs. centered building selection for the focus channel on the test prediction evaluation?

### 4.8.3 Comparing Computational Models

Ultimately, experiments comparing the U-Net and Transformer architecture were carried out. Here, the focus lies on two critical aspects regarding the architecture of the computation models. First, as described in Chapter 2, the two models are fundamentally different. On the one hand, U-Net performs multiple convolutions, whereas the Swin Transformer computes an attention map. The initial idea was that the Transformer could capture the complex situations in patches where multiple operators are at play better due to the sequential attention calculations. The reason being that, compared to the convolutions in the U-Net architecture, intuitively, one could think that calculating multiple attentions yields better results than convolving all the information in an image into a one-dimensional vector. Comparing the computation models while holding the parameters and datasets constant, the main research questions (RQ) for these experiments were:

**RQ.1** How do Transformer models compare to the U-Net architecture in training performance?

**RQ.2** How do both computation models perform on the *swisstopo* data set which is more difficult to learn compared to the *OSM* data set (see Chapter 3)?

**RQ.3** What are the differences in prediction capabilities of the two computation models if there are multiple operators at play?

# Chapter 5

# Results

This section presents the findings of the experiments performed in the study. The results are organized in the same order as described in Section 4.8. It is important to state that the evaluation of the predictions of generalization results lays a stronger focus on the quantitative results of the IoU metrics compared on the qualitative description. Nonetheless, the qualitative description is briefly noted for each illustrated example.

## 5.1 Brute Force Approach on Extensive Swisstopo Data

Table 5.1 shows an overview of the experimental setup for the brute force experiments on the *swisstopo* data. All the parameters except for the loss function and the learning rates were held constant. The learning rate was defined with the PyTorch Lightning learn rate finder functionality in order to get the optimal value.

Table 5.1: Setup for the brute force approach experiments, showing stable hyperparameters except the learning rate.

| **Epochs** | **Loss Function** | **Learning Rate** | **Computation Model** | **Batch Size** | **Optimizer** | **Attention Head** |
|---|---|---|---|---|---|---|
| 300 | Focal Tversky | 8.32e-05 | Swin tiny | 32 | AdamW | mlp |
| 300 | Dice BCE | 2.29e-05 | Swin tiny | 32 | AdamW | mlp |

The results of the two experiments in Table 5.2 show similar metric values for both trials. In all of the logged metrics the deviation of the results, the absolute difference is smaller than one percent. Note that the shown values represent mean values on the prediction performance on the test set.

Table 5.2: Results of the brute force approach experiments indicating similar performance of both Swin Transformer models trained with the Focal Tversky Loss and the DiceBCE Loss functions. All the metrics represent mean values on the prediction performance on the test set.

| **Experiment** | **Accuracy** | **F1** | **mIoU** | **Precision** | **Recall** |
|---|---|---|---|---|---|
| Focal Tversky Loss | 0.94514 | 0.8691 | 0.86921 | 0.8595 | 0.87902 |
| Dice BCE Loss | 0.94498 | 0.8679 | 0.86714 | 0.86401 | 0.87194 |

In Figure 5.1 we can observe the loss curves of the training and validation loss for the experiments carried out with the Focal Tversky Loss and Dice BCE Loss. In both experiments, the models were trained for 300 epochs. The curves look similar, with a slight tendency to overfitting in the Dice BCE Loss experiment. Interestingly, the curves for the training loss are still pointing downwards even after 300 epochs while the validation loss is more or less stable. The latter, indicates that the models would be able to learn more from the data, if training was continued. However, since the data intake for these two models was significant, the training was stopped after roughly seventy hours.

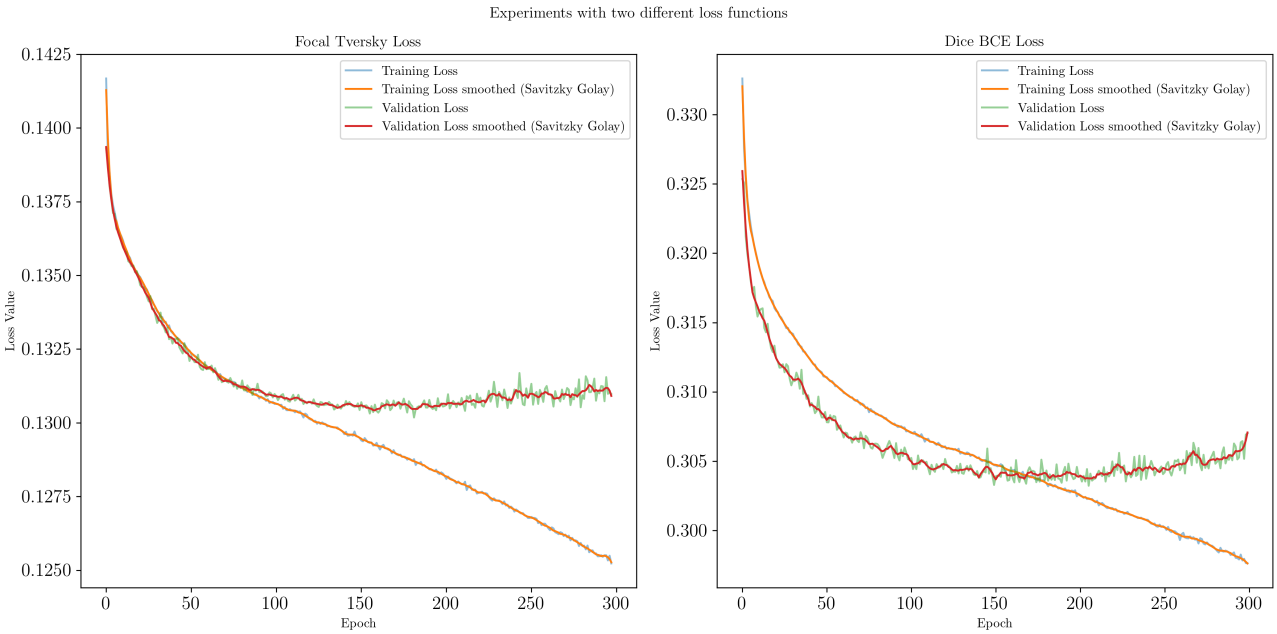



Figure 5.1: Logged training and validation loss for the experiments with the Focal Tversky and the Dice BCE Loss on the extensive swisstopo dataset. Both plots show the originally logged values, and a smoothed curve.

### 5.1.1 Prediction Results: Dice BCE Loss Experiments

Three visual examples of the model prediction performance for the Dice BCE Loss experiment are shown in Figures 5.2 - 5.4. The prediction results were randomly selected after being split into percentiles. The first (lowest) percentile shows the worst performance whereas the sixth (highest) percentile shows the best prediction performance of the model. In the leftmost image (Figure 5.2 - 5.4), the input image is shown, followed by the target and finally the predicted patch. As mentioned in Section 4.6.5, ideally, the predicted patch looks the same as the target patch. The IoU value between the input and the target, denoted as *IPT-TGT IoU*, serves as a measure for the difficulty of the prediction task in terms of the IoU (low IoU - more difficult, high IoU - less difficult). The IoU value between the target and the prediction, denoted as *TGT-PRED IoU*, indicates how well the model predicted the task in terms of the IoU (low IoU - bad prediction, high IoU - good prediction).

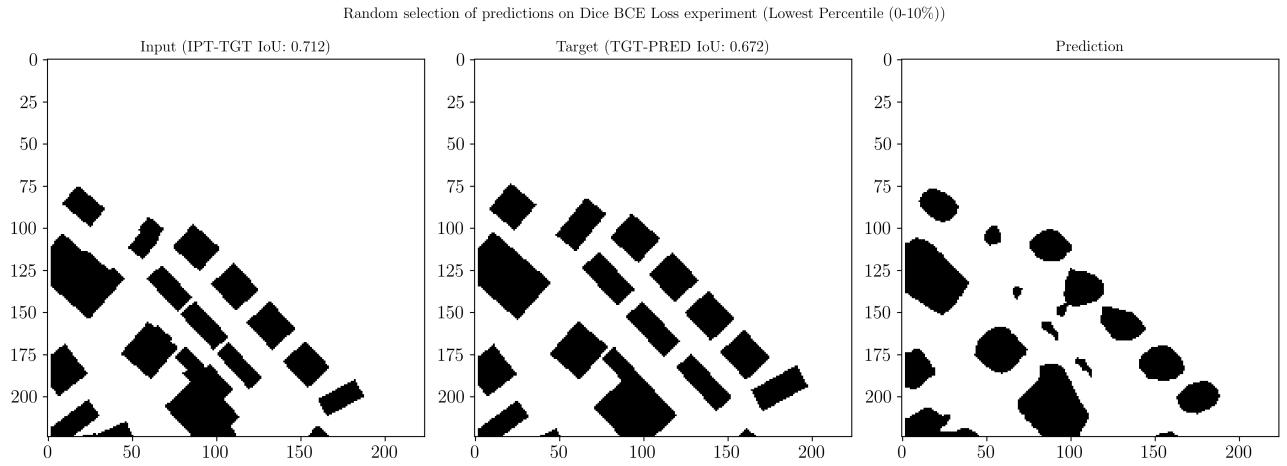



Figure 5.2: Random selection of a prediction result from the lowest percentile with the Swin Transformer computation model and the single channel data model on the extensive swisstopo dataset using the Dice BCE Loss function. Operators at play: *simplification, displacement, enlargement.* The model appears confused about the various generalization tasks at hand and predicts a number of round blobs as buildings. The prediction result is not useable.

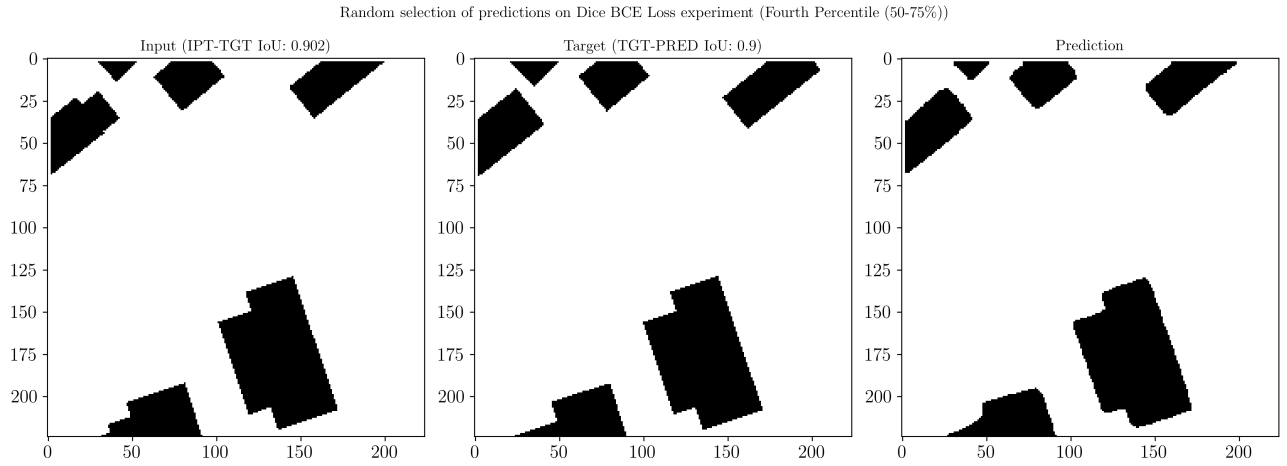



Figure 5.3: Random selection of a prediction result from the mid-range percentile with the Swin Transformer computation model and the single channel data model on the extensive swisstopo dataset using the Dice BCE Loss function. Operators at play: *simplification.* On the upper left corner, the model correctly performs the simplification of one building. The other buildings in the scene stay the same. Also here, the model predicts straight edges. The corners of most of the buildings appear to be rounded.

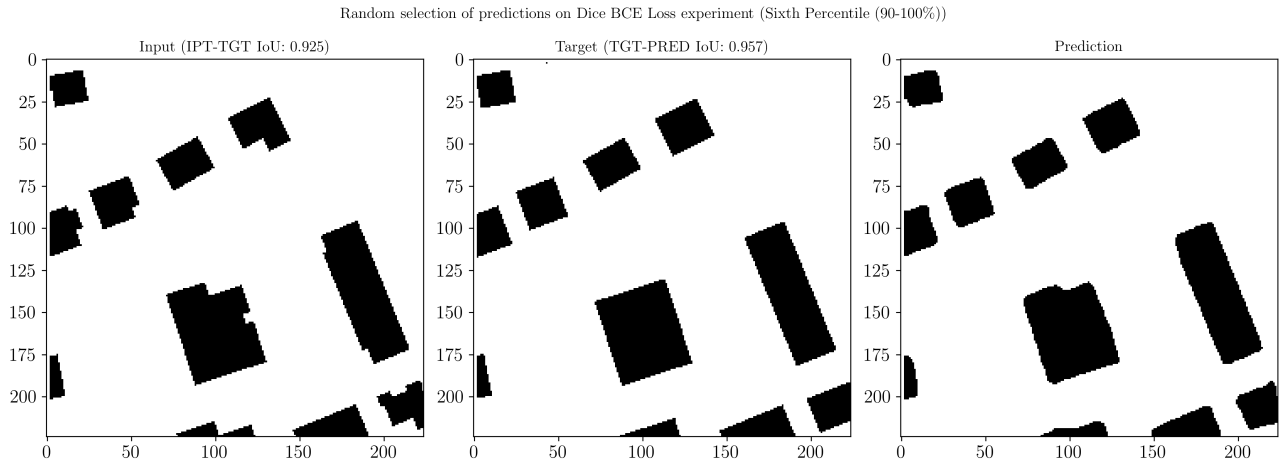



Figure 5.4: Random selection of a prediction result from the highest percentile with the Swin Transformer computation model and the single channel data model on the extensive swisstopo dataset using the Dice BCE Loss function. Operators at play: *simplification*. The model performs multiple simplification operations correctly in the upper half of the patch. Also, the long building on the right side of the patch appears to be generalized well. The large structure in the center shows a result that slightly differs from the target.

### 5.1.2 IoU Value Distribution for Dice BCE Loss Experiments

The IPT-TGT IoU and TGT-PRED IoU appear to have a strong correlation. Also, both value distributions (IPT-TGT IoU and TGT-PRED IoU) shown in Figure 5.5 are similar in shape and form.

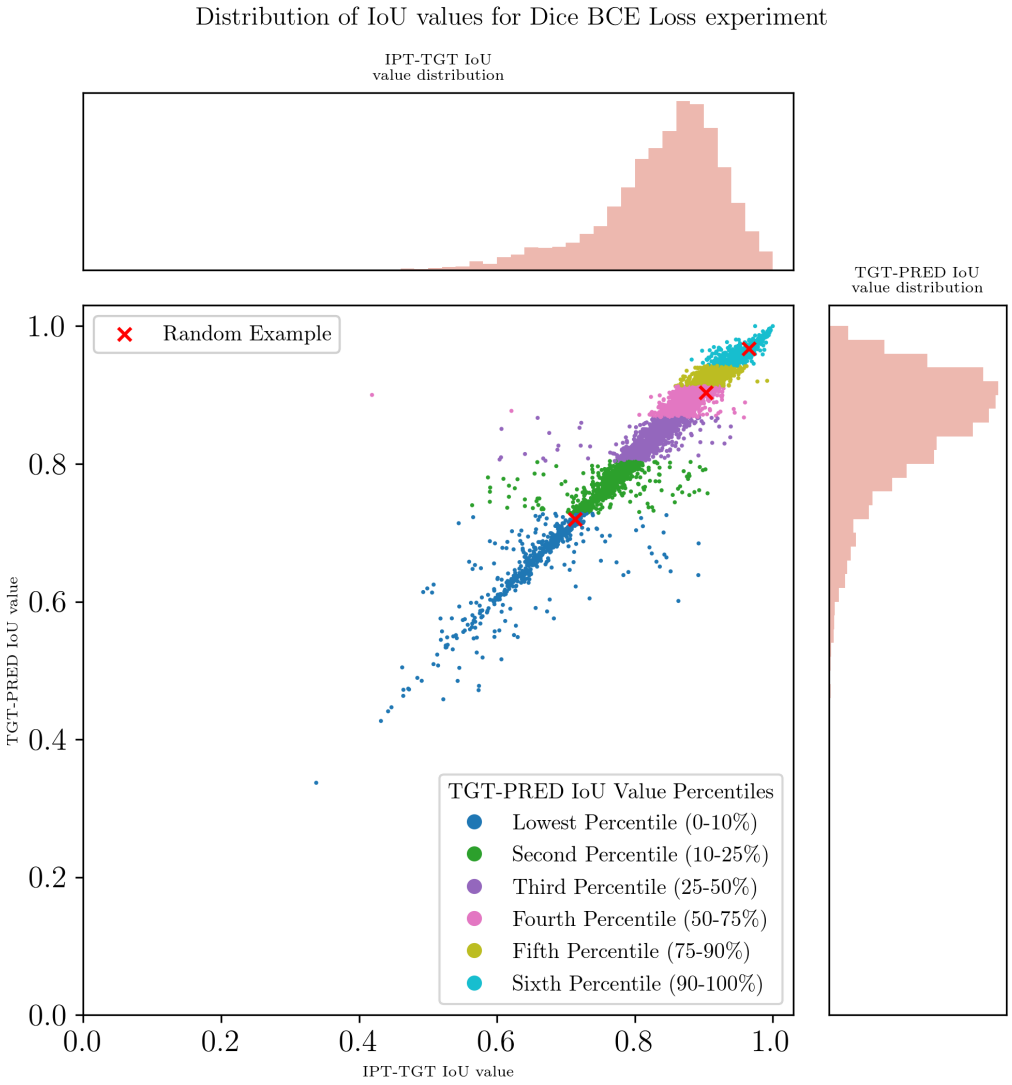



Figure 5.5: IoU distribution of all the predictions on the test set of the Swin Transformer model trained on comprehensive *swisstopo* dataset with single-channel data model and the Dice BCE Loss function. The distributions on the x- and y-Axis appear to be substantially skewed. Between IPT-TGT IoU and TGT-PRED IoU there appears to be a high correlation. The red marks show the position of the randomly sampled examples shown in Figures 5.2 - 5.4

### 5.1.3 Prediction Results: Focal Tversky Loss Experiments

For the experiment carried out using the Focal Tversky loss, the results are in the same structure as described in Section 5.1.1

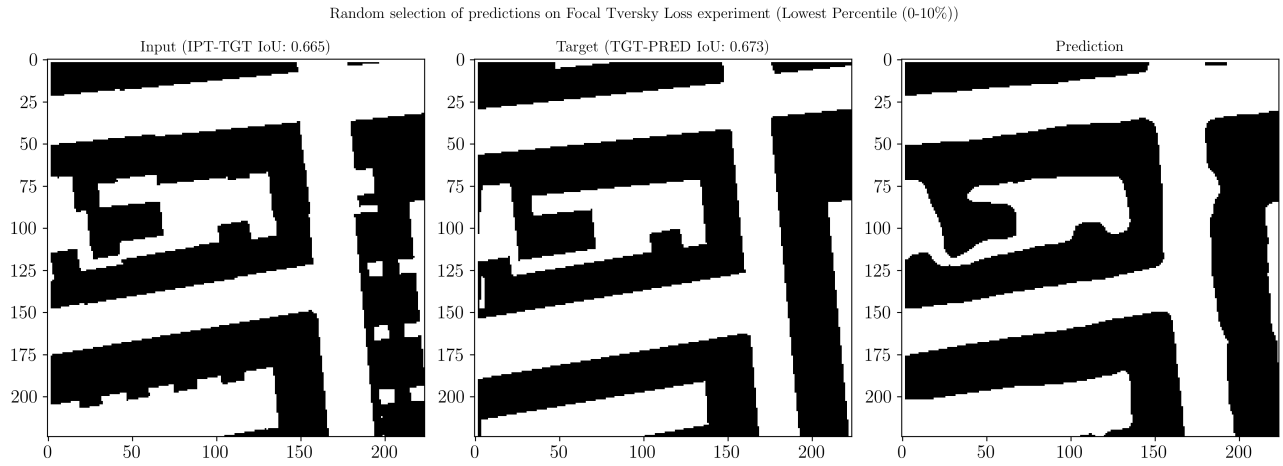



Figure 5.6: Random selection of a prediction result from the lowest percentile with the Swin Transformer computation model and the single channel data model on the extensive swisstopo dataset using the Focal Tversky Loss function. Operators at play: *simplification*. The model fails the generalization task overall. The building edges appear curved and wobbly.

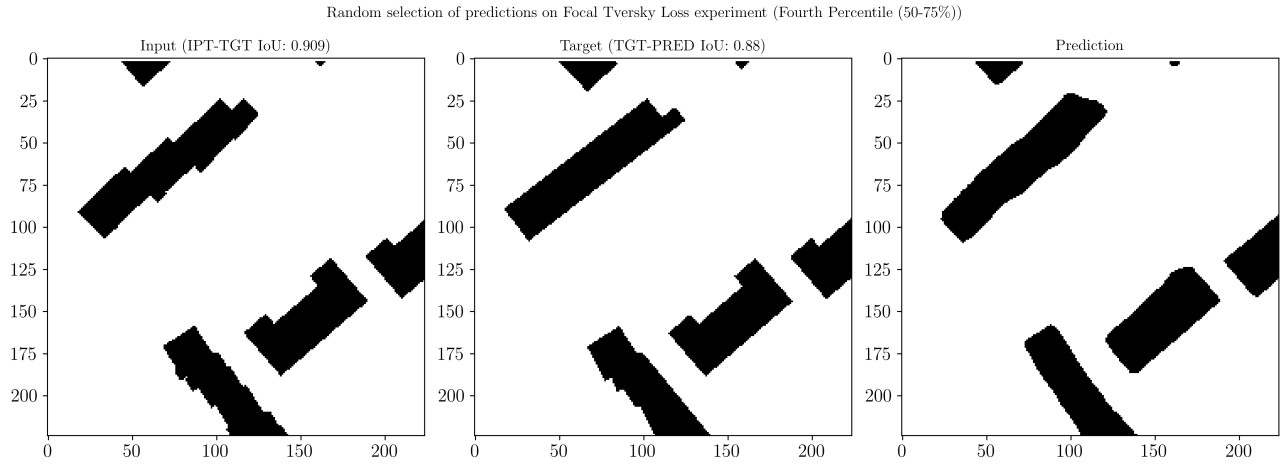



Figure 5.7: Random selection of a prediction result from the mid-range percentile with the Swin Transformer computation model and the single channel data model on the extensive swisstopo dataset using the Focal Tversky Loss function. Operators at play: *simplification*. The model achieves a mediocre prediction performance. The jagged building edges lead to wobbly prediction results.

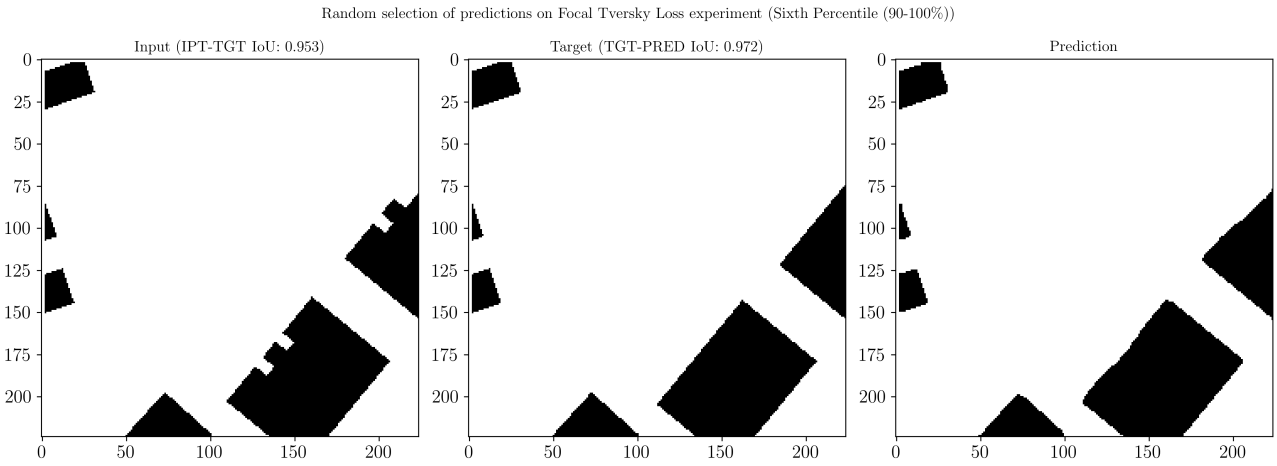



Figure 5.8: Random selection of a prediction result from the highest percentile with the Swin Transformer computation model and the single channel data model on the extensive swisstopo dataset using the Focal Tversky Loss function. Operators at play: *simplification*. The model predicts most of the simplifcation operations correctls. The building in the bottom right shows an acceptably straight edge. On the right edge of the patch, the building has a slanted edge where the model partially filled up intricate features.

### 5.1.4 IoU Value Distribution for Focal Tversky Loss Experiments

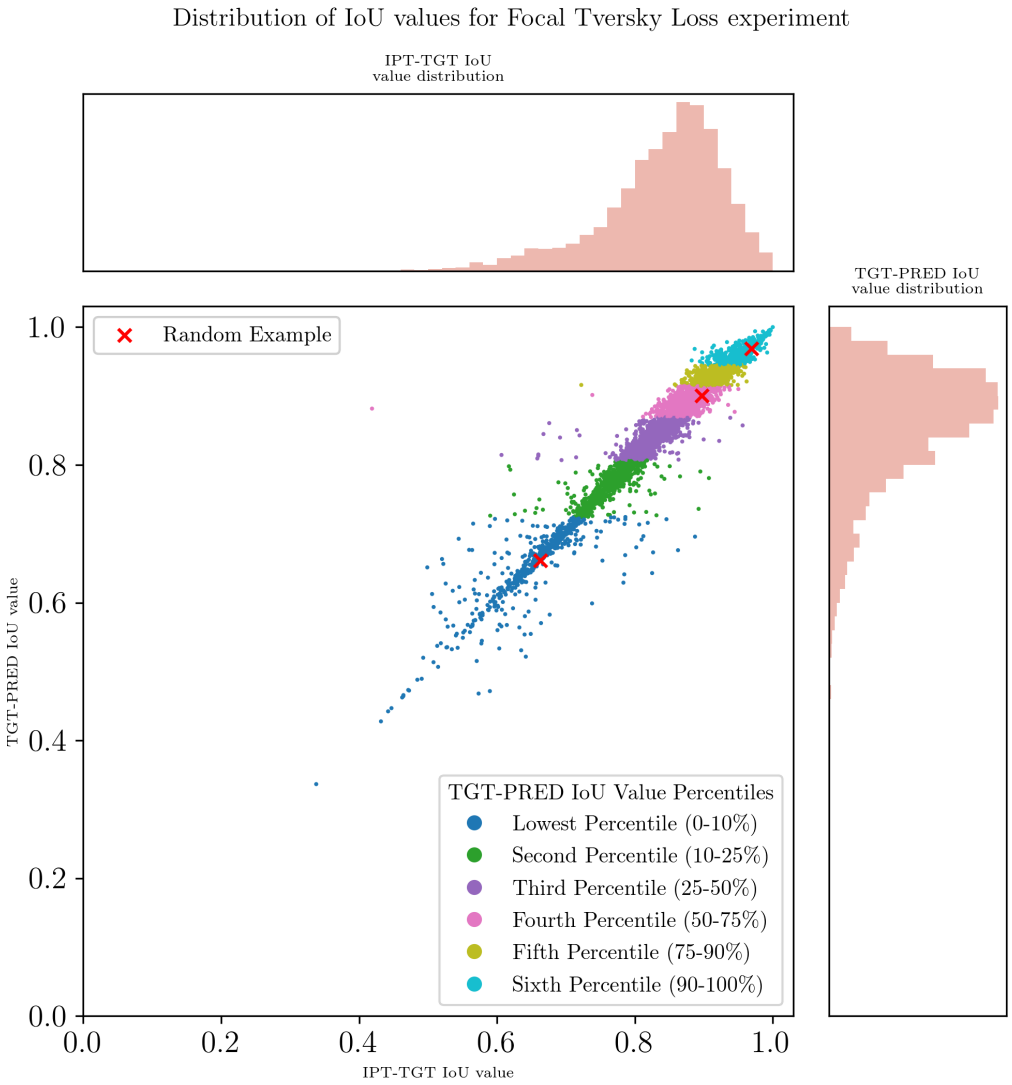



Figure 5.9: IoU distribution of all the predictions on the test set of the Swin Transformer model trained on comprehensive *swisstopo* dataset with single-channel data model and the Focal Tversky Loss function. The distributions on the x- and y-Axis appear to be substantially skewed. Between IPT-TGT IoU and TGT-PRED IoU there appears to be a high correlation. The red marks show the position of the randomly sampled examples shown in Figures 5.6 - 5.8

## 5.2 Comparing Data Models

This section shows the results of the experiments on the application of the three different data models on the Zurich (swisstopo) and Stuttgart (OSM) data sets.

### 5.2.1 Training and Validation Loss Performance

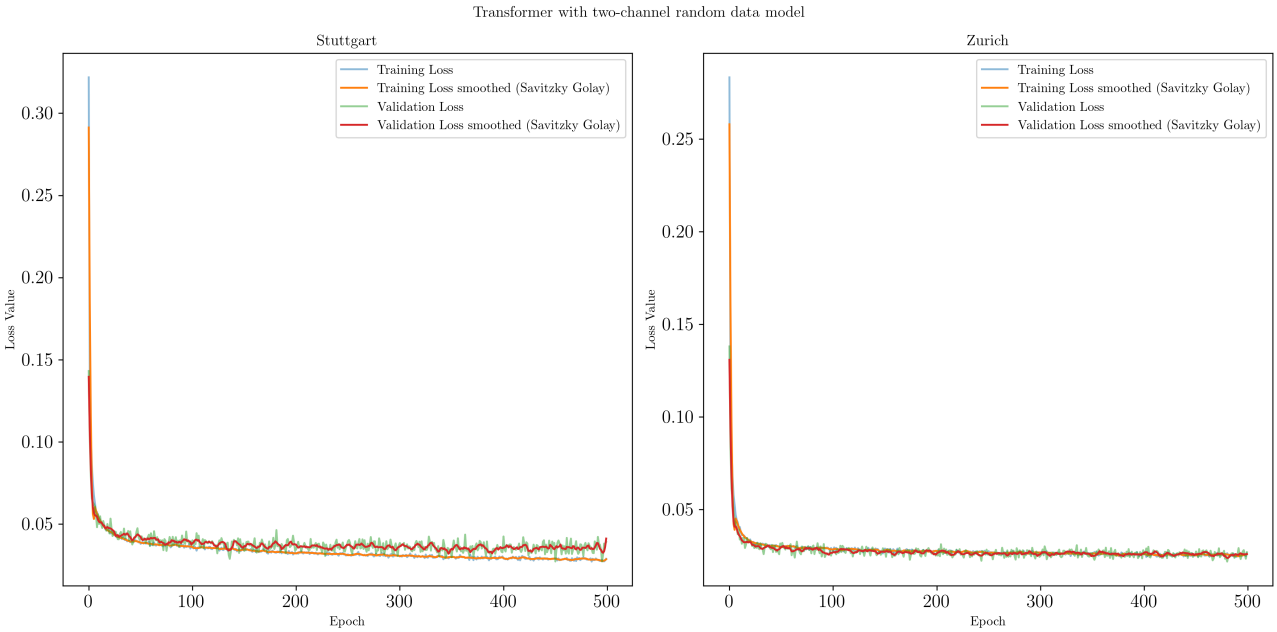



Figure 5.10: Training and validation loss curves of the experiments carried out with the Swin Transformer model on Zurich and Stuttgart with the two-channel random data model. The model trained for 500 epochs. In the case of Stuttgart there appears to be slight overfitting. In the case of Zürich even after the extensive training time, no overfitting seems to occur.

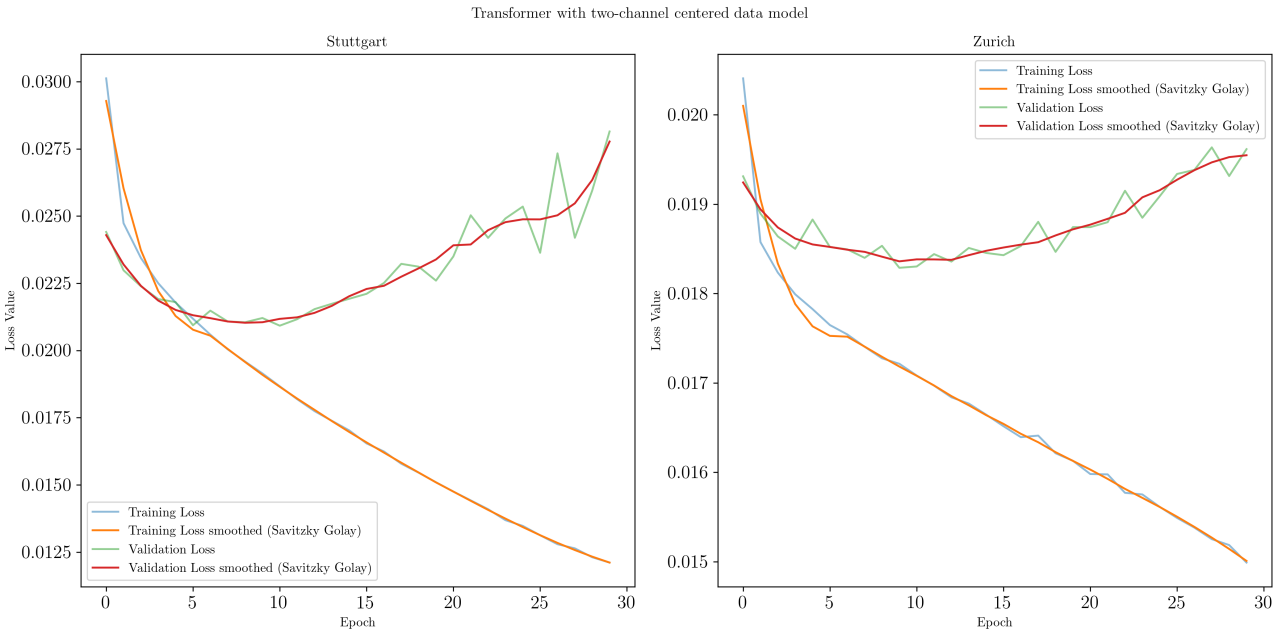



Figure 5.11: Training and validation loss curves of the experiments carried out with the Swin Transformer model on Zurich and Stuttgart with the two-channel centered data model. For Stuttgart, overfitting occurs quicker compared to Zürich. For Stuttgart around epoch 7, for Zürich around epoch 12. Compared to the Swin Transformer models, U-Net appears to run into overfitting much quicker.

Comparing the performance in terms of the training and validation loss curves, it is clear that the two-channel random data model allows for extended training time whereas the two-channel centered approach runs into overfitting quicker.

### 5.2.2 mIOU Performance Overview

Table 5.3: Comparison of the mIoU results on the test set of all Swin Transformer models trained with the different data models on the Zurich (swisstopo) dataset.

| Computation Model | Data Model | Batch Size | mIoU |
|---|---|---|---|
| Swin Transformer | Single-channel | 32 | 0.91491 |
| Swin Transformer | Two-channel random | 32 | **0.95053** |
| Swin Transformer | Two-channel centered | 32 | 0.94127 |
| Swin Transformer | Two-channel centered | 8 | 0.94165 |

The highest mIoU value on the test set was achieved by the two-channel random data model (Table 5.3). The results are not much higher than the results for the two-channel centered data model approach. It is noteworthy that the experiment on the two-channel centered data model showed an increased performance after reducing the batch size from 32 to 8 samples per batch, yielding a convergence towards the performance of the model trained with the two-channel random data model, even after relatively short training.

Table 5.4: Comparison of the mIoU results on the test set of all Swin Transformer models trained with the different data models on the Stuttgart (OSM) dataset.

| Computation Model | Data Model | Batch Size | mIoU |
|---|---|---|---|
| Swin Transformer | Single-channel | 32 | 0.96063 |
| Swin Transformer | Two-channel random | 32 | 0.96475 |
| Swin Transformer | Two-channel centered | 32 | 0.97358 |
| Swin Transformer | Two-channel centered | 8 | **0.97755** |

Table 5.4 shows the IoU values for the experiments performed on the Stuttgart (OSM) dataset. The performance of the computation model trained with the two-channel centered data model and reduced batch size performs best. However, the performance difference is marginal. Similar to the experiments on the Zurich (swisstopo) dataset (Table 5.3), for two-channel centered's data model, when decreasing the batch size from 32 to 8, the model performance increased in terms of the mIoU.

### 5.2.3 Prediction Results: Two-Channel Random Data Model

In the following, three visual examples from predictions on the Zurich dataset are shown. Each example is a randomly sampled prediction result. All the prediction results were classified into percentiles with the same procedure as described in Section 5.1.1.

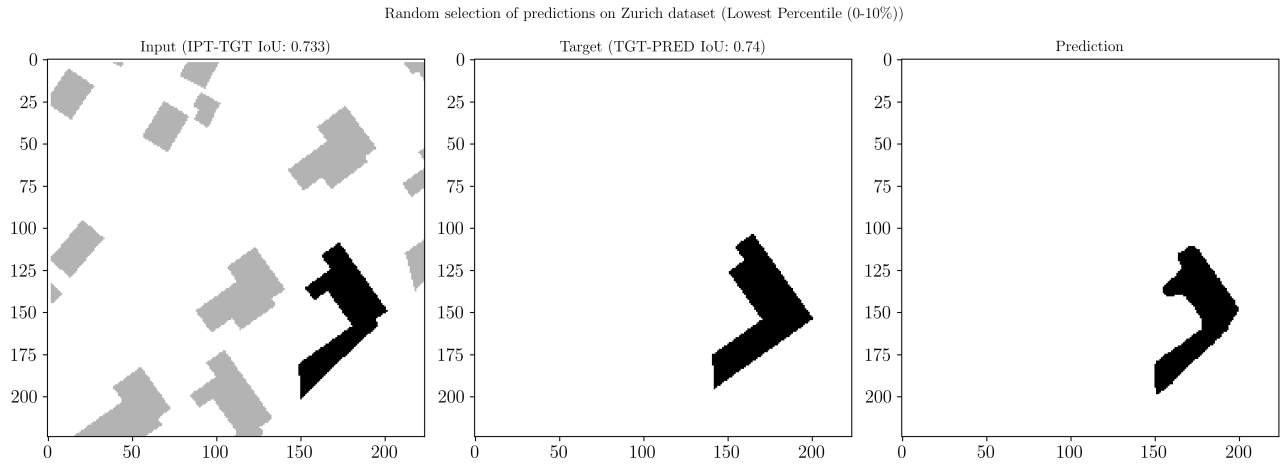



Figure 5.12: Random selection of a prediction result from the lowest percentile with the Swin Transformer computation model and the two-channel random data model on the Zurich (swisstopo) dataset. Operators at play: *simplification*. The model fails to clearly predict straight lines. The blob is predicted even though it should be removed.

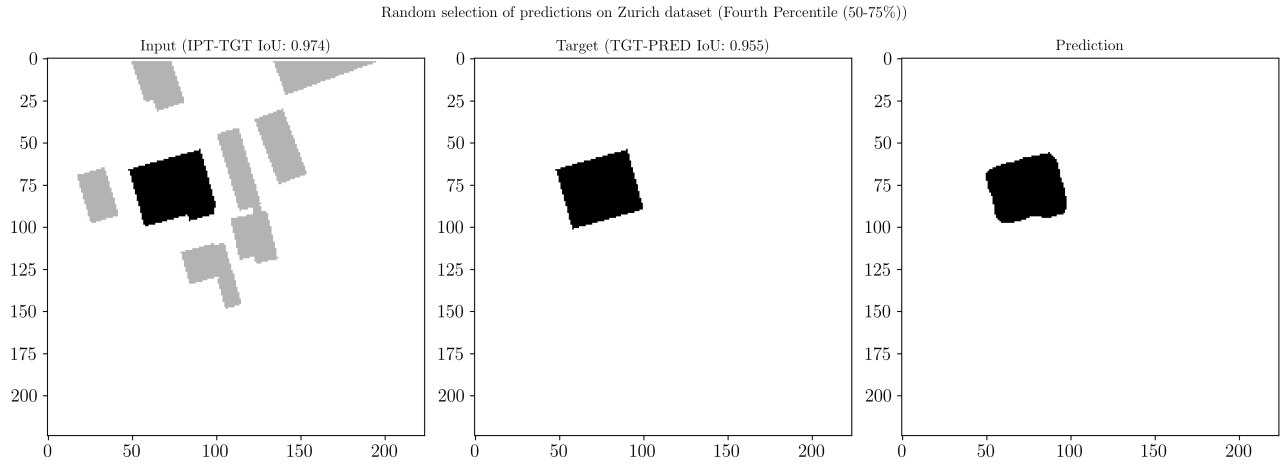



Figure 5.13: Random selection of a prediction result from the mid-range percentile with the Swin Transformer computation model and the two-channel random data model on the Zurich (swisstopo) dataset. Operators at play: *simplification*. There is a minor simplification operation which the model should predit, however, it fails to do so correctly.

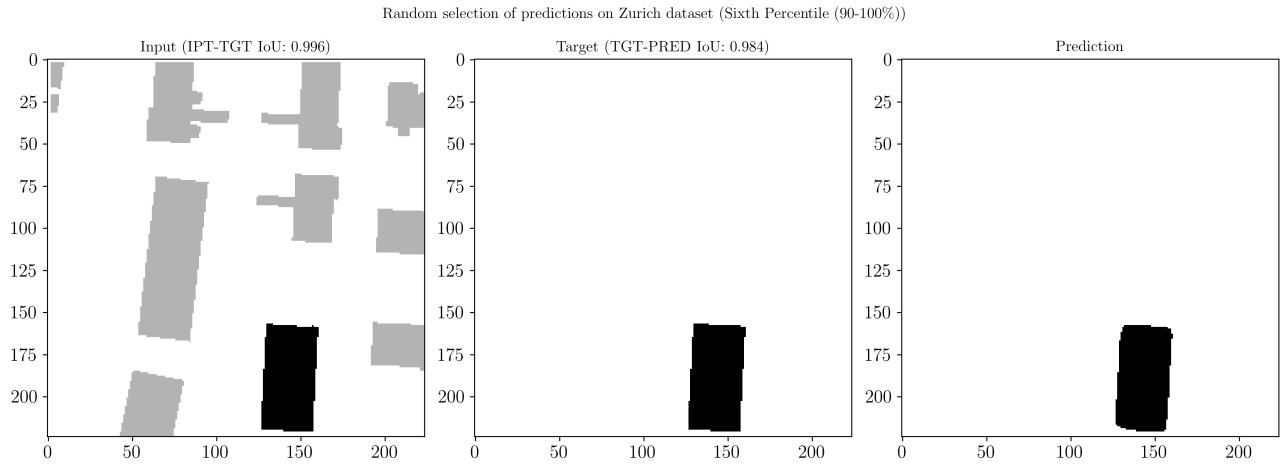



Figure 5.14: Random selection of a prediction result from the highest percentile with the Swin Transformer computation model and the two-channel random data model on the Zurich (swisstopo) dataset. Operators at play: *None*. Since there is no operator at play, the model should predict the input image exactly, which it does.

### 5.2.4 Prediction Results: Two-channel Centered Data Model

In the following, three visual examples from predictions on the Zurich dataset are shown. Each example is a randomly sampled prediction result. All the prediction results were classified into percentiles with the same procedure as described in Section 5.1.1.

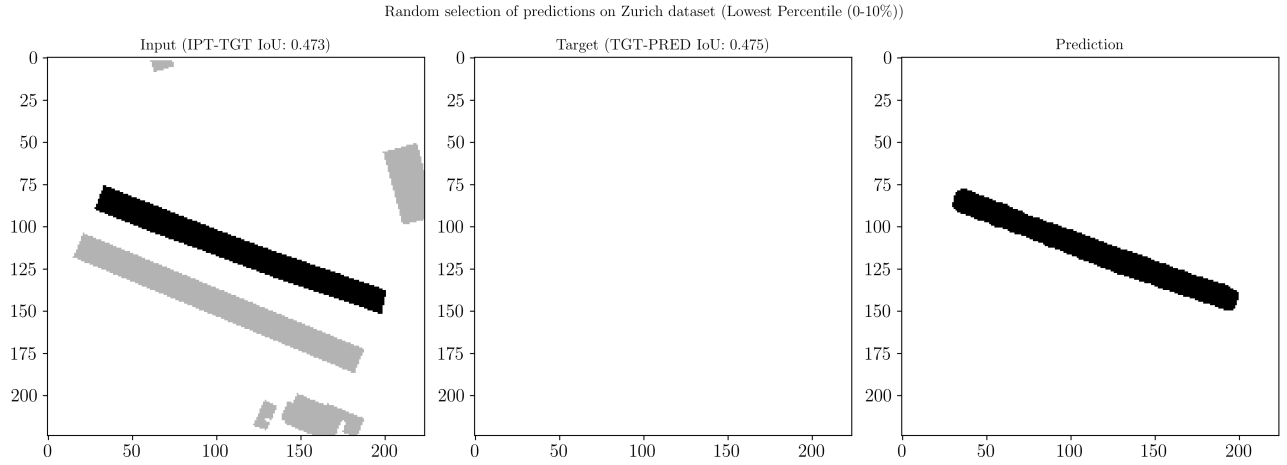



Figure 5.15: Random selection of a prediction result from the lowest percentile with the Swin Transformer computation model and the two-channel centered data model on the Zurich (swisstopo) dataset. Operators at play: *deletion*. Even though the model should predict an empty image, the model predicts a structure similar to the input.

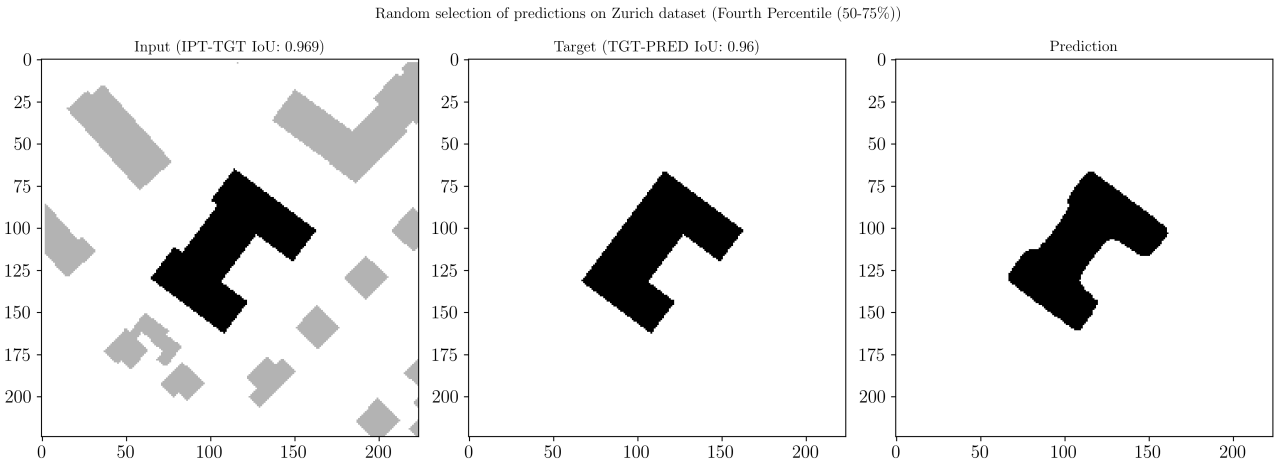



Figure 5.16: Random selection of a prediction result from the mid-range percentile with the Swin Transformer computation model and the two-channel centered data model on the Zurich (swisstopo) dataset. Operators at play: *simplification*. The model performs a minor simplification but not to a satisfying extent. The corners on the left side of the building are still visible in the prediction evn though there should be a straight edge.

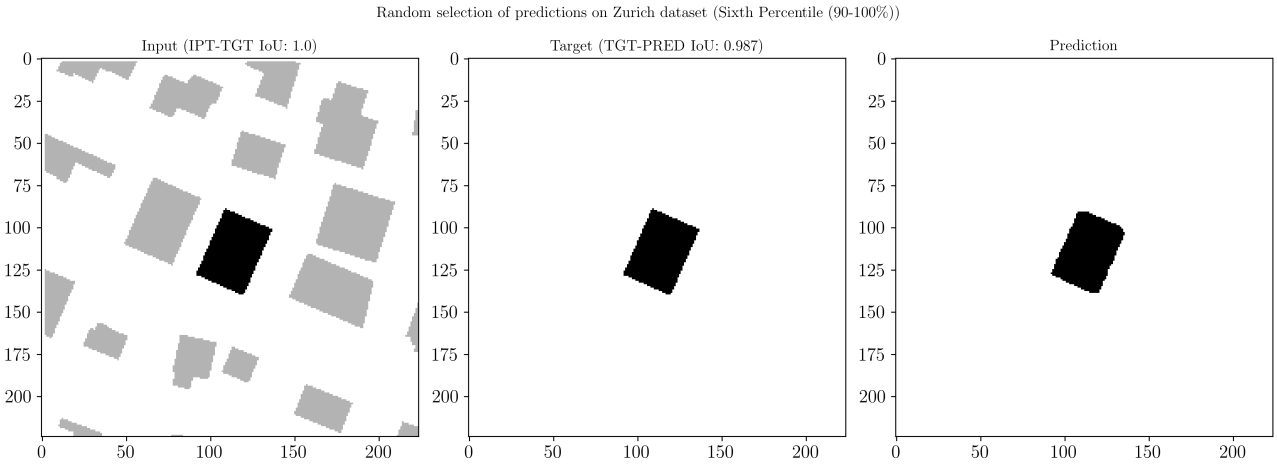



Figure 5.17: Random selection of a prediction result from the highest percentile with the Swin Transformer computation model and the two-channel centered data model on the Zurich (swisstopo) dataset. Operators at play: *None*. There is no operator at play and the model correctly predicts the input structure. However, the corners appear rounded, which is not a perfect result.

### 5.2.5 IoU Value Distribution

This section shows the IoU value distribution of two Swin Transformer models trained on the Zurich test dataset. Figure 5.18 depicts the distribution of all the IoU values between the input and the target (IPT-TGT) plotted against the target-prediction IoU (TGT-PRED) in the test set for the Swin Transformer model trained with the two-channel random data model.

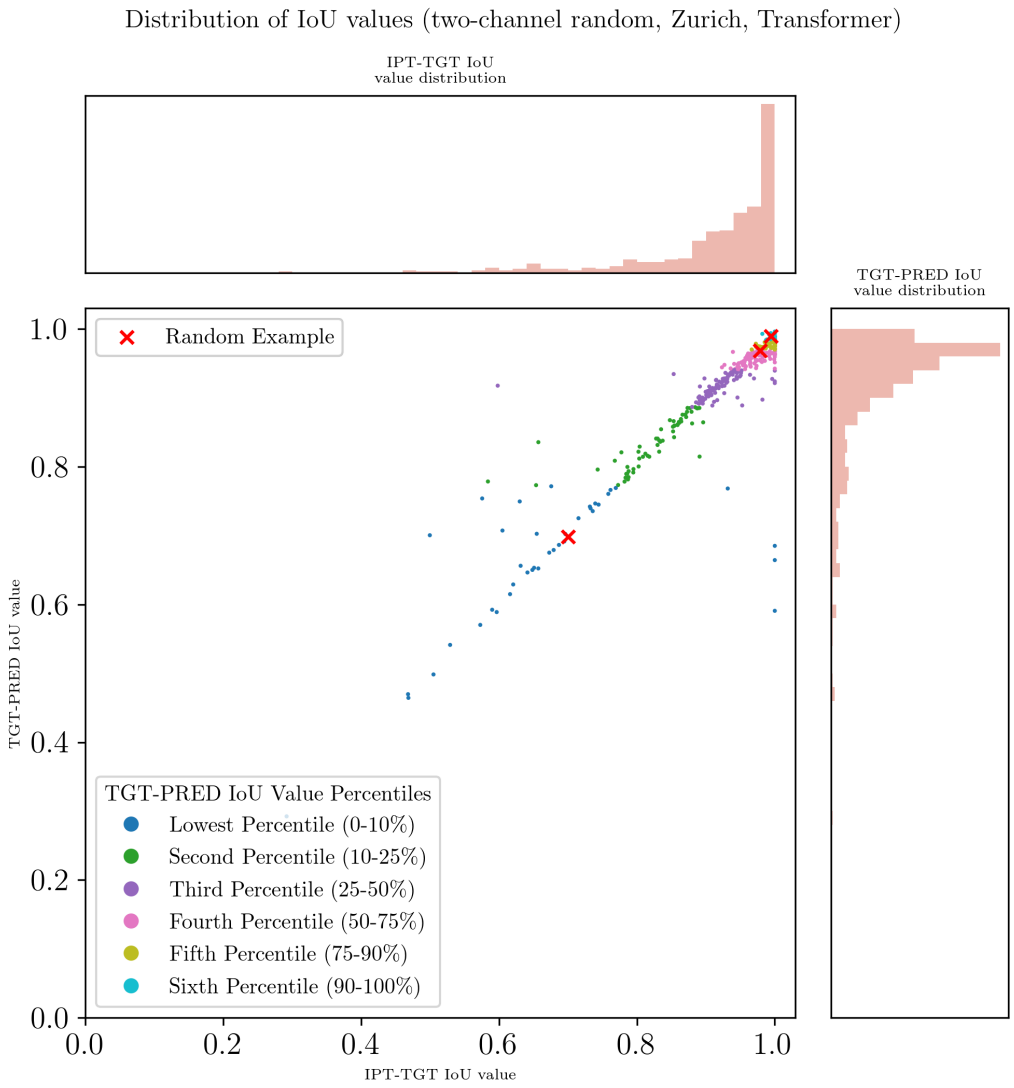



Figure 5.18: IoU distribution of all the predictions on the test set of the Swin Transformer model trained on the Zurich dataset with the two-channel random data model. The distributions on the x- and y-Axis appear to be heavily skewed. Between IPT-TGT IoU and TGT-PRED IoU there appears to be a high correlation. The red marks show the position of the randomly sampled examples shown in Figures 5.12 - 5.14.

Figure 5.19 depicts the distribution of all the IoU values between the input and the target (IPT-TGT) plotted against the target-prediction IoU (TGT-PRED) in the test set for the Swin Transformer model trained with the two-channel centered data model.

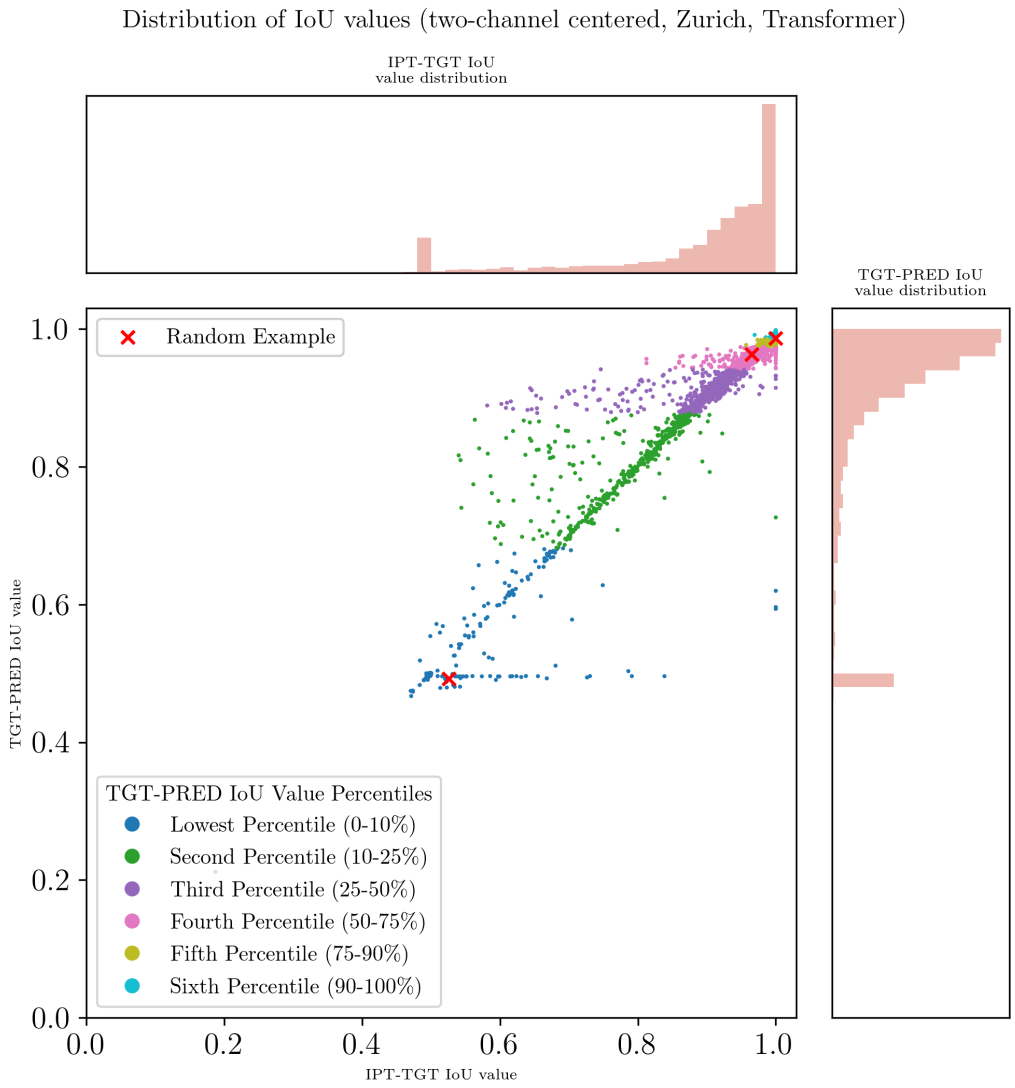



Figure 5.19: IoU distribution of all the predictions on the test set of the Swin Transformer model trained on the Zurich dataset with the two-channel centered data model (batch size 8 experiment). The distributions on the x- and y-Axis appear to be heavily skewed. Between IPT-TGT IoU and TGT-PRED IoU there appears to be a high correlation. The red marks show the position of the randomly sampled examples shown in Figures 5.15 - 5.17.

## 5.3 Comparing Computational Models

This section shows the results of the experiments on the application of the two different computational models on the Zurich (swisstopo) and Stuttgart (OSM) data sets. To keep the result section concise, only the loss performance curves of the U-Net computation models are included. The loss curves of the Swin Transformer model can be observed in Figures 5.10 and 5.11.

### 5.3.1 Training and Validation Loss Performance

Comparing the performance in terms of the training and validation loss curves, it is clear that the two-channel random data model allows for extended training time whereas the two-channel centered approach runs into overfitting quicker.

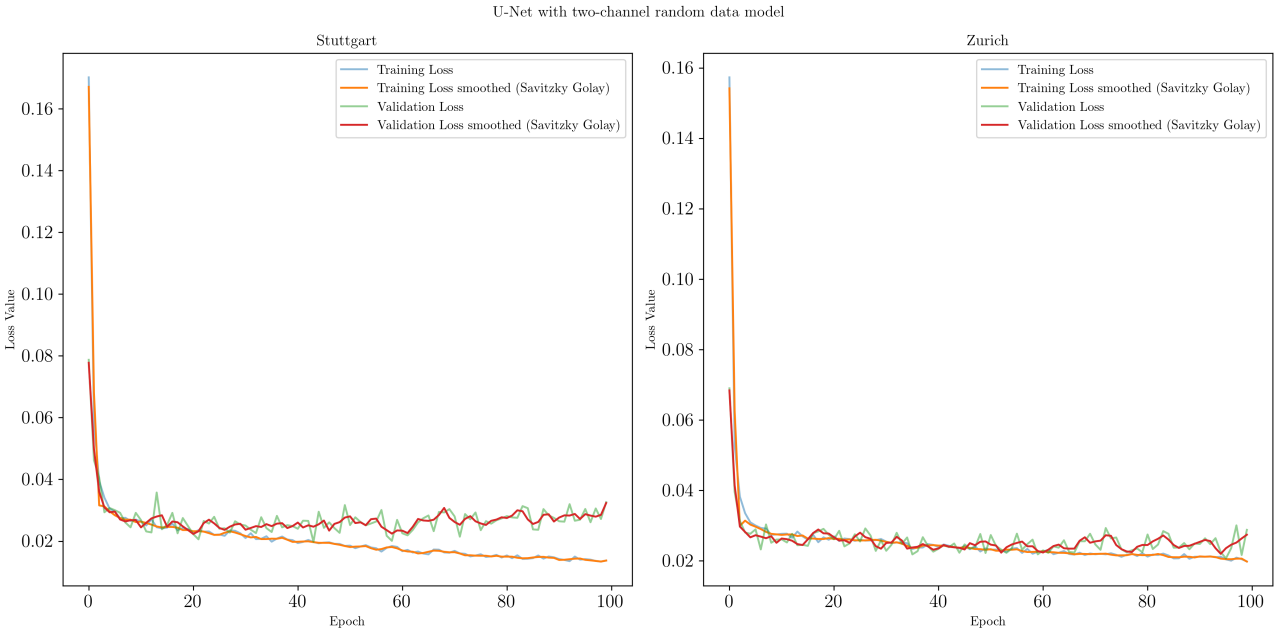



Figure 5.20: Training and validation loss curves of the experiments carried out with the U-Net computation model on Zurich and Stuttgart with the two-channel random data model. The model trained for 100 epochs. In the case of Stuttgart there appears to be slight overfitting. In the case of Zürich, towards the end of the training time, a slight tendency towards overfitting seems to occur.

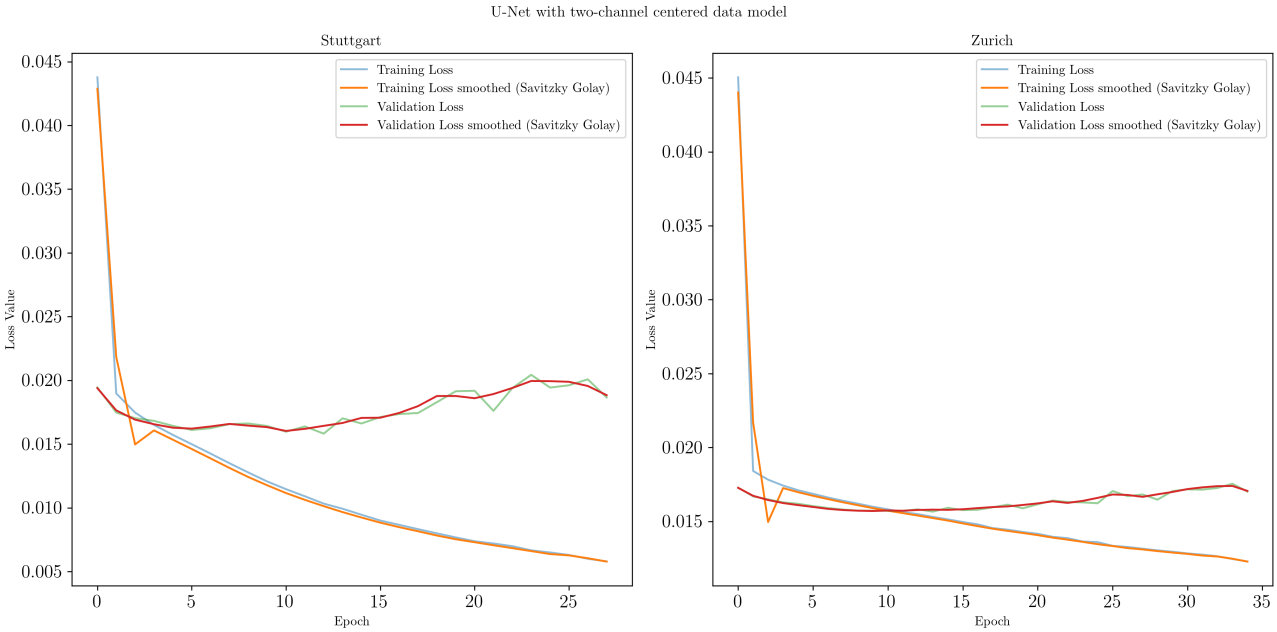



Figure 5.21: Training and validation loss curves of the experiments carried out with the U-Net computation model on Zurich and Stuttgart with the two-channel centered data model. Similar to the results shown in Figure 5.11 for Stuttgart, overfitting occurs quicker compared to Zürich. For Stuttgart around epoch 7, for Zürich around epoch 12. Compared to the Swin Transformer models, U-Net appears to run into overfitting much quicker.

### 5.3.2 mIOU Performance Overview

The comparison of the computational models (U-Net and Swin Transformer) on two different data models (Two-channel random and centered) can be observed in Tables 5.5 and 5.6. The experiments

were performed on the Zurich (swisstopo) and Stuttgart (OSM) dataset. For conciceness, the results in tables 5.5 and 5.6 show only the results for the best performance. In both cases, the computational model showed better performance on the Stuttgart (OSM) dataset. A complete overview with the model performance in terms of the mIoU is shown in Table 5.7.

Table 5.5: Comparison of the model prediction performance on the test set (Stuttgart) in terms of the mIoU. The Tableshows the result for both computational models (Swin Transformer and U-Net), trained with the two-channel random data model.

| **Computation Model** | **Data Model** | **mIoU** |
|---|---|---|
| Swin Transformer | Two-channel random | 0.96475 |
| U-Net | Two-channel random | **0.99943** |

Comparing the Swin Transformer and the U-Net on the two-channel random data model, the U-Net model performed better than the Swin Transformer model in terms of the mIoU. The performance difference amounts to 0.03468.

Table 5.6: Comparison of the model prediction performance on the test set (Stuttgart) in terms of the mIoU. The Table shows the result for both computational models (Swin Transformer and U-Net), trained with the two-channel centered data model.

| **Computation Model** | **Data Model** | **mIoU** |
|---|---|---|
| Swin Transformer | Two-channel centered | 0.97358 |
| U-Net | Two-channel centered | **0.99781** |

Comparing the Swin Transformer and the U-Net on the two-channel centered data model, the U-Net model outperformed the Swin Transformer model in terms of the mIoU by 0.2423.

### 5.3.3 Prediction Results: Two-Channel Random

In the following, the prediction results of the U-Net trained with the two-channel random data model are shown. In Figures 5.22 to 5.24, three different examples from different prediction-performance percentiles are shown. The selection and computation was done as described in Section 5.1.1.

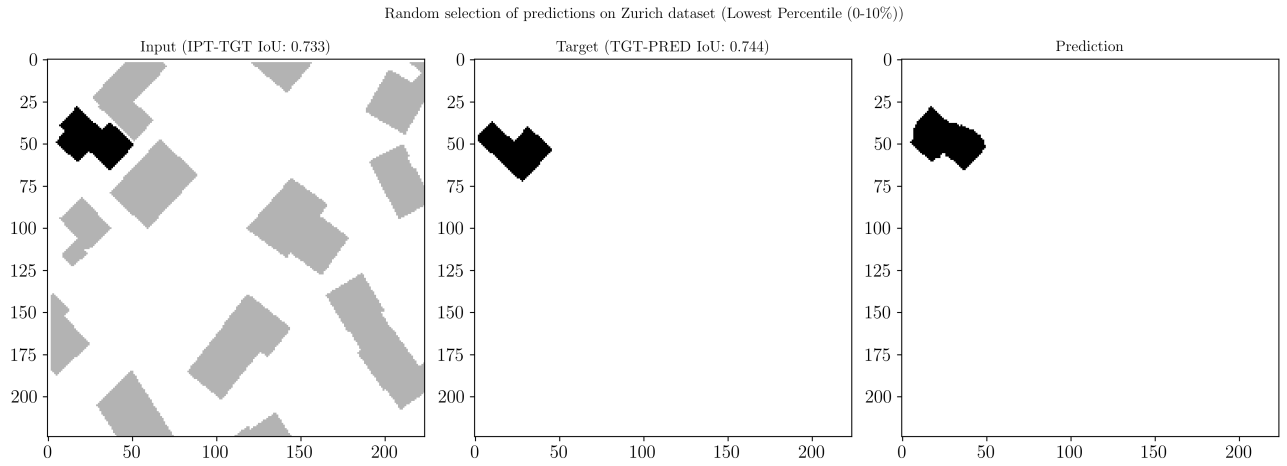



Figure 5.22: Random selection of a prediction result from the lowest percentile with the U-Net computation model and the two-channel random data model on the Zurich (swisstopo) dataset. Operators at play: *simplification*. The model fails to simplify the shown structure. Especially the creation of a rounded building edge is striking. The shape of the target building is not matched.

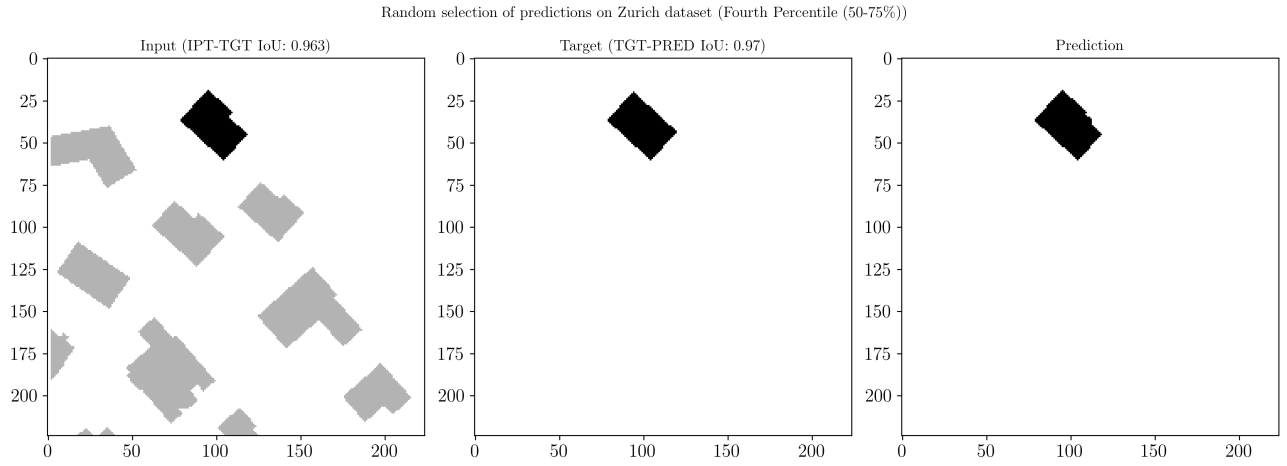



Figure 5.23: Random selection of a prediction result from the mid-range percentile with the U-Net computation model and the two-channel random data model on the Zurich (swisstopo) dataset. Operators at play: *simplification*. The model more or less is able to predict a structure similar to the target. However, there appears to be a slanted building edge, which appears distorted on the output.

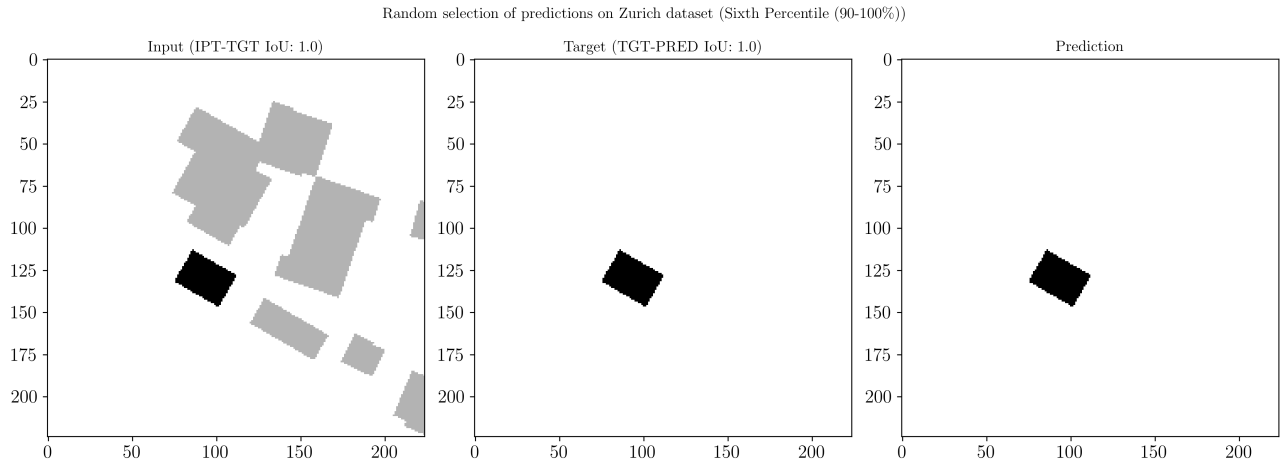



Figure 5.24: Random selection of a prediction result from the highest percentile with the U-Net computation model and the two-channel random data model on the Zurich (swisstopo) dataset. Operators at play: *None*. The model correctly predicts the input structure.

### 5.3.4 Prediction Results: Two-Channel Centered

In the following, the prediction results of the U-Net trained with the two-channel random data model are shown. In Figures 5.25 to 5.27, three different examples from different prediction-performance percentiles are shown. The selection and computation was done as described in Section 5.1.1.

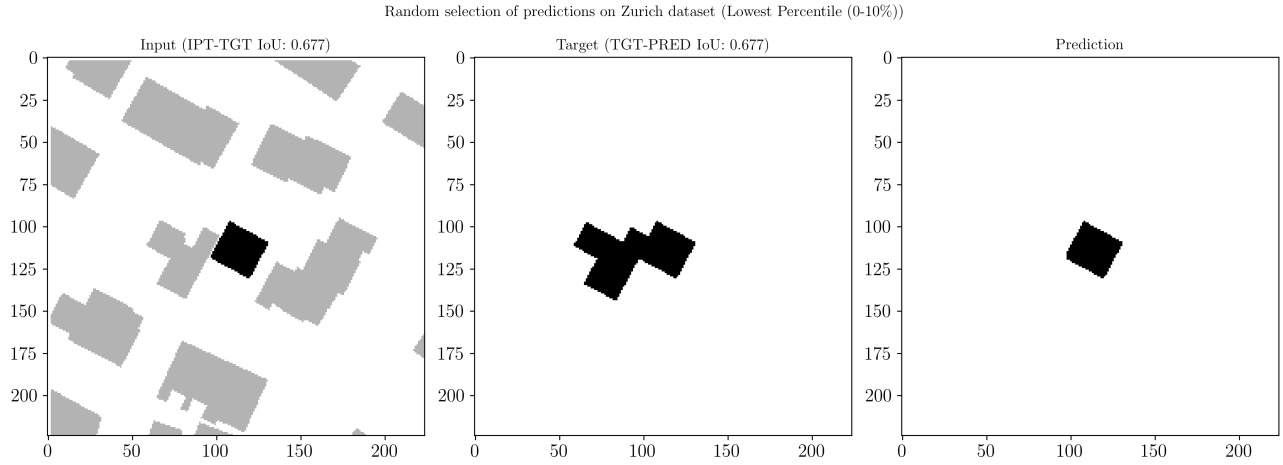



Figure 5.25: Random selection of a prediction result from the lowest percentile with the U-Net computation model and the two-channel centered data model on the Zurich (swisstopo) dataset. Operators at play: *aggregation*. The target building from the input image, is represented in an aggregated form in the target patch. However, the model fails to detect that and predicts the input structure.

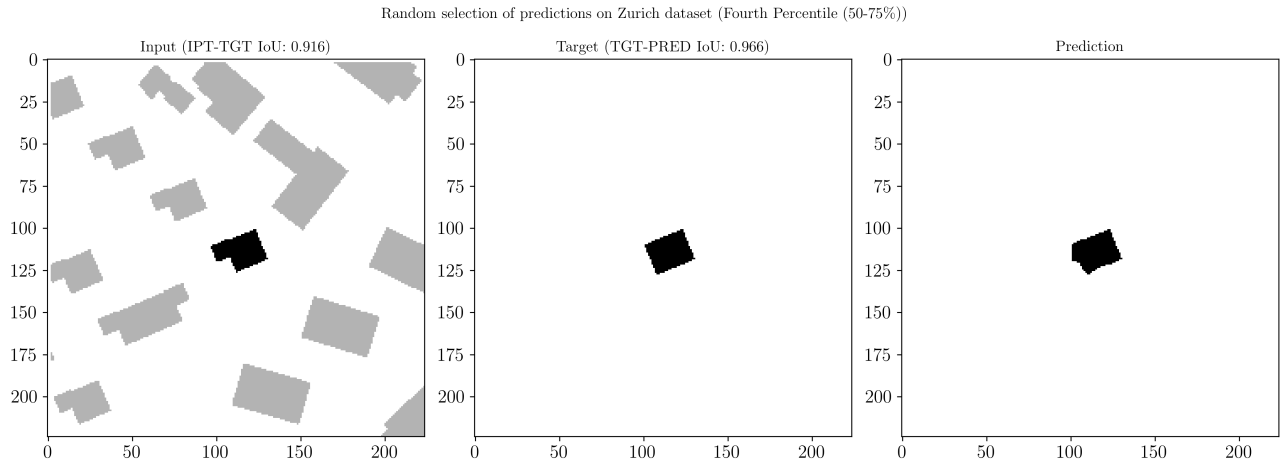



Figure 5.26: Random selection of a prediction result from the mid-range percentile with the U-Net computation model and the two-channel centered data model on the Zurich (swisstopo) dataset. Operators at play: *simplification*. The model achieves a modest result. The simplification is not complete and two out of four edges of the building appear wobbly.

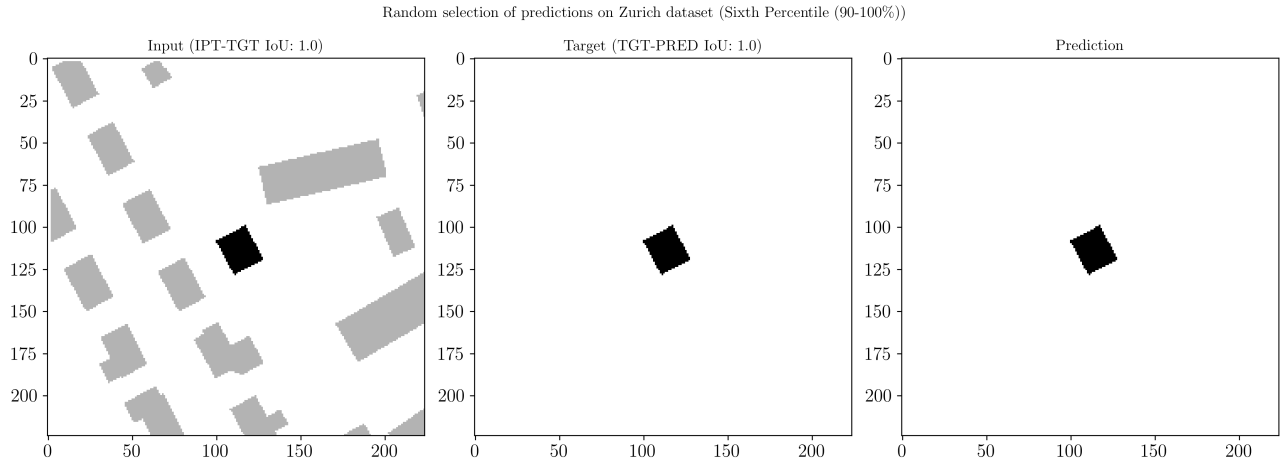



Figure 5.27: Random selection of a prediction result from the highest percentile with the U-Net computation model and the two-channel centered data model on the Zurich (swisstopo) dataset. Operators at play: *None*. The model correctly predicts the input structure.

### 5.3.5 IoU Value Distribution

This section shows the IoU value distribution of two U-Net models trained on the Zurich test dataset. Figure 5.28 depicts the distribution of all the IoU values between the input and the target (IPT-TGT) plotted against the target-prediction IoU (TGT-PRED) in the test set for the U-Net model trained with the two-channel random data model.

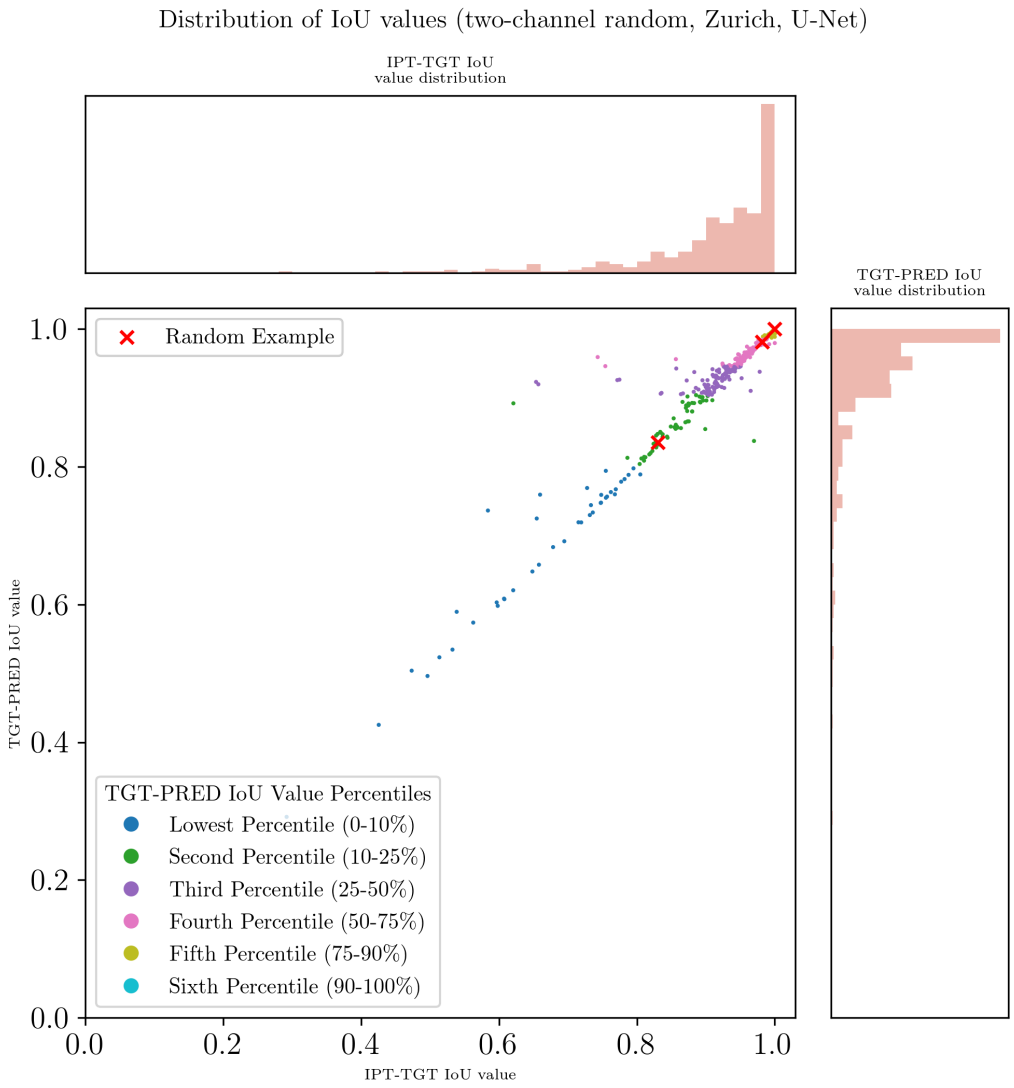



Figure 5.28: IoU distribution of all the predictions on the test set of the U-Net model trained on the Zurich dataset with the two-channel random data model. The distributions on the x- and y-Axis appear to be heavily skewed. Between IPT-TGT IoU and TGT-PRED IoU there appears to be a high correlation. The red marks show the position of the randomly sampled examples shown in Figures 5.22 - 5.24.

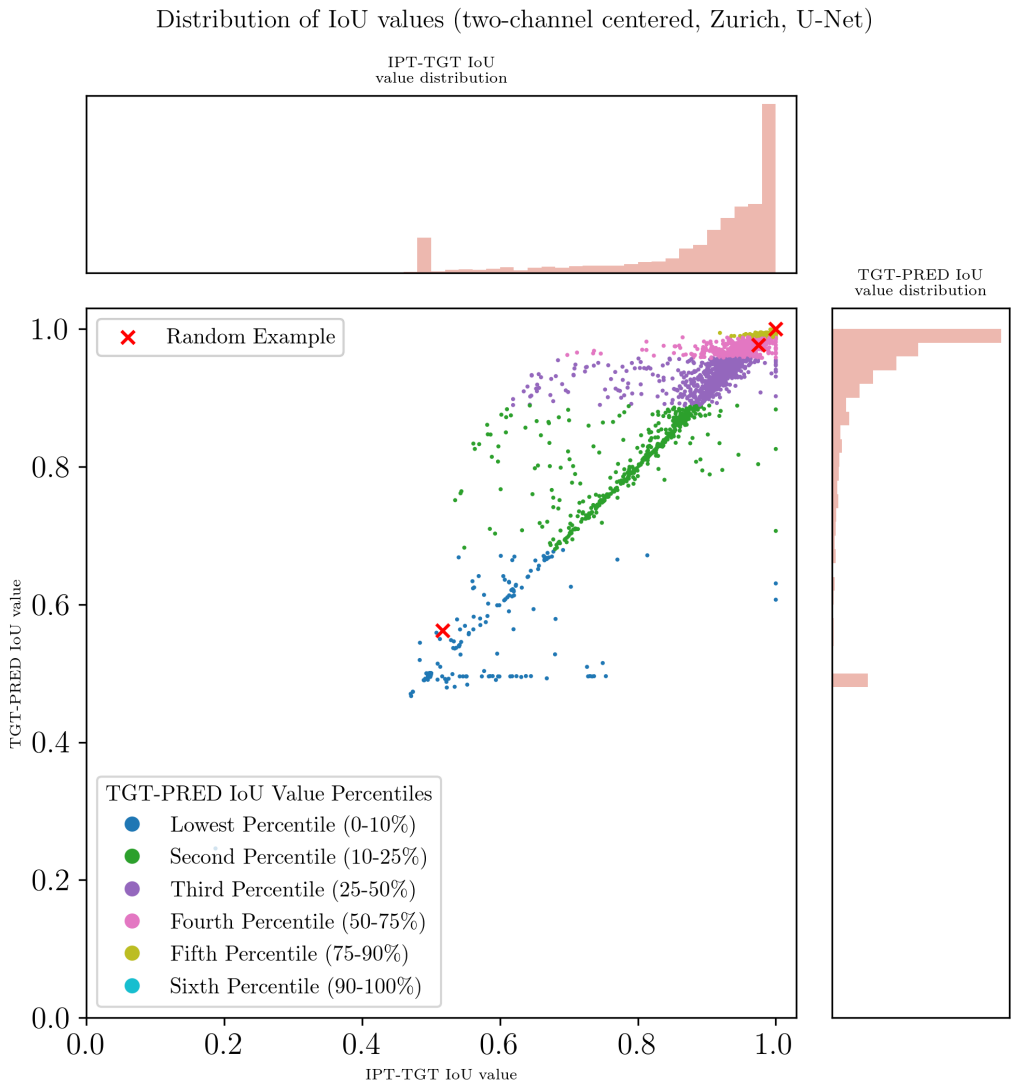



Figure 5.29: IoU distribution of all the predictions on the test set of the U-Net model trained on the Zurich dataset with the two-channel centered data model. The distributions on the x- and y-Axis appear to be heavily skewed. Between IPT-TGT IoU and TGT-PRED IoU there appears to be a high correlation. The red marks show the position of the randomly sampled examples shown in Figures 5.25 - 5.27.

## 5.4 Result Overview

Table 5.7: Results of the relevant results for the carried out experiments regarding data models, computation models on Stuttgart (OSM) and Zurich (swisstopo) dataset. Note that all the reported metrics represent the median IoU (mIoU) of the prediction results on the respective test sets.

| **Computation Model** | **Data Model** | **Batch Size** | **mIoU Stuttgart** | **mIoU Zurich** |
|---|---|---|---|---|
| Swin Transformer | Single-channel | 32 | 0.96063 | 0.91491 |
| Swin Transformer | Two-channel random | 32 | 0.96475 | 0.95053 |
| Swin Transformer | Two-channel centered | 32 | 0.97358 | 0.94127 |
| Swin Transformer | Two-channel centered | 8 | 0.97755 | 0.94165 |
| U-Net | Two-channel random | 8 | **0.99943** | 0.94593 |
| U-Net | Two-channel centered | 8 | 0.99781 | **0.95766** |

# Chapter 6

# Discussion

In the following section, we discuss the results presented in Chapter 5.

## 6.1 Brute Force Experiments

**General:** No significant performance difference between the two tested loss functions could be detected (Section 4.8.1, RQ.1). Thus, the more straightforward Binary Cross Entropy loss function was mainly used for the subsequent experiments comparing data models and computation models. Moreover, neither of the two configurations was capable of significantly outperforming the models used in Feng et al. (2019) (Section 4.8.1, RQ.2).

**Training:** Principally, the Swin Transformer architecture can cope with and learn from a large dataset, given enough time to train (Section 4.8.1, RQ.3). One of the initial ideas of the brute force experiments was to check whether it would be possible to train a purpose-built Swin Transformer for map generalization and achieve better prediction results than Feng et al. (2019) (Section 4.8.1, RQ.2). Figure 5.1 shows both experiments' reported training and validation dataset loss functions. Interestingly, the training loss decreases in both cases even after 300 epochs (roughly 70 hours of training on one Tesla V100 GPU). In the case of the Dice BCE Loss, there might be a slight tendency towards overfitting; however, it would still be interesting to see the results with more extended training given better-engineered training data and different data models. Therefore, it can be stated that building a purpose-built Swin Transformer model failed for now, but changing specific settings discovered with the completion of subsequent experiments could still yield a more successful model (Section 4.8.1, RQ.4).

**Predictions:** Figures 5.2 to 5.8 all show the overall mediocre capability of the model to learn given generalization tasks. For the experiments, a single-channel data model was used, and thus the prediction task for the model incorporates the prediction of every building in one patch. Figure 5.2 shows that if there is a relatively high complexity (IoU value IPT-TGT 0.712), the model has limited capabilities for predictions. Moreover, Figure 5.6 shows a case of strong simplification with which the model expresses difficulties. As the prediction tasks get simpler (Figures 5.3, 5.7 and 5.4, 5.8), the model shows the ability to predict structures to a significant extent. However, the model still shows uncertainty in cases where the input data shows intricate structures, jagged edges, or the removal of

entire building parts. The uncertainty is shown through non-rectangular shapes or slightly wobbly building edges.

**IoU Values:** Figures 5.9 and 5.5 both show that there is a strong correlation between the IPT-TGT IoU and the TGT-PRED IoU. The latter suggests that if the IPT-TGT IoU is high for one prediction, the TGT-PRED IoU is likely to be high too. Furthermore, an investigation of the value distributions for the IPT-TGT IoU and the TGT-PRED IoU indicates skewed distribution towards the higher IoU values. Thus, it can be inferred that simple prediction tasks can be handled, presumably because the simple cases are highly over-represented in the training data. On the other hand, complicated prediction tasks might be underrepresented in the data, and thus the model has issues with predicting them.

## 6.2 Comparing Data Models

**General:** As described in Section 4.8, the performances of the different data models were assessed in detail. These experiments aimed to eventually find an optimal solution to feed the data to the computation models. For example, the single channel data model proposed in Feng et al. (2019) has the disadvantage that the model has to predict a large number of pixels; thus, if the task gets increasingly complex, the model runs into problems as discussed in Section 6.1. Therefore, the data models described in 4.4 with centered building placement as proposed by Fu (2022) and random building placement as proposed by the author of this thesis, respectively.

**Training:** The selection of the data model significantly sways training (Section 4.8.2, RQ.1). It is interesting to see the difference in training possibilities in Figure 5.10 and Figure 5.11. The model trained with the two-channel random data model can train for a long time (roughly 500 epochs) without any significant sign of overfitting. Compared to the same model trained with the two-channel centered data model, severe overfitting occurs after about seven to ten epochs. Overfitting in the latter-mentioned configuration is likely to occur since the building is always placed in the patch's center. In the case of the random selection of the building, the model is forced to learn more because the building position constantly changes, which could be one of the main reasons no overfitting occurs when the building is always placed in the center.

**Predictions:** The prediction performance differs when training a model with a specific data model Section 4.8.2, RQ.2). The predictions displayed in Section 5.2.3 and Section 5.2.4 show varying results. The model trained on the two-channel random data model did not run into overfitting (Figure 5.10 and Figure 5.20). Thus, whenever the model is shown a prediction task that is difficult (e.g., Figure 5.12, *displacement and enlargement*), the model tries to predict a structure but shows high uncertainty. The latter is manifested in the blob-like appearance in the prediction patch in Figure 5.12. On the other hand, the model trained with the two-channel centered data model showed severe overfitting (Figure 5.11 and Figure 5.10) after a short training time. Thus, the model might not be trained very well. It can be observed that when the model is given a complex prediction task (e.g., Figure 5.15, *aggregation and enlargement* or Figure 5.16, *simplification*), the model tends to predict the input. The latter observed behavior could be counteracted by increasing the training data set size.

**IoU values:** The visualizations of the IoU value distribution in Figures 5.18 and 5.19 both show a strong correlation between the difficulty of the given prediction task (IPT-TGT IoU) and the evaluation of the predictions (TGT-PRED IoU). The latter observation indicates a skewed distribution of simple versus complicated examples in the training data, with a clear tendency to over-represented simple and under-represented complex cases. Furthermore, Table 5.3 shows that computation models trained on *swisstopo* data with the two-channel random data model compared to computation models trained with the two-channel centered data model do not differ significantly in terms of performance. For example, the random data model shows slightly better performance (random mIoU **0.95053** vs. centered mIoU 0.94165). On the contrary, comparing the performance of computation models trained with both data models on the *OSM* data, we can observe that in Table 5.4, the centered data model appears to be slightly better than for the random model (centered mIoU **0.97755** vs. random mIoU 0.96475). Another interesting observation is that in all the figures depicting the IoU distribution of models trained with the two-channel centered data model, there appears to be a spike around the IoU value of 0.5. The latter is a phenomenon that requires further investigation. A hypothetical answer could be that the pre-processing of the data with the centered approach detects building deletions better than the other data models. The fact that this spike is also visible in the distribution of the prediction IoUs could further point to increased learning capability of the deletion operator of models trained with the centered approach.

## 6.3 Comparing Computation Models

**General:** Ultimately, the comparison between the computation models Swin Transformer and U-Net sheds light on the capability of the higher (Swin Transformer) or lower (U-Net) complexity model to learn building generalization. The model complexity is expressed in the number of model parameters (see Section 4.2). The models described in more detail in Chapter 2 have their respective advantages and disadvantages. One of the main issues with the Swin Transformer is the overall need for more data. The main theoretical advantage was thought to be the attention mechanisms. On the other hand, U-Net was thought to be too simple for learning the task of building generalization. The comparison shown in Table 5.5 and Table 5.6 show the performance of both computation models trained with the same data model.

**Training:** In terms of training performance, no computation-model specific could be detected (Section 4.8.2, RQ.1). Analysing the single loss curves shown in Figures 5.10 and 5.20 yields that the Swin Transformer and U-Net can train long, and show good learning capabilities without any severe overfitting. On the contrary, consulting the loss curves in Figures 5.11 and 5.21 shows that both computation models ran into severe overfitting after a low amount of epochs. Therefore, the main effect revealing a difference in training capability cannot be allocated to the computation model selection but to the data model selection.

**Predictions:** All the test predictions on the *OSM* test sets showed a better performance than the results on the *swisstopo* test sets (Section 4.8.3, RQ.2). The prediction results shown in Sections 5.3.3 and 5.3.4 show similar prediction capabilities of the models as discussed in sections 5.2.3 and 5.2.4. The models predict simple cases, but whenever the task appears to be more complex (more than one operator at play, significant structural change), all models run into trouble. This manifests in the

prediction results being either wobbly (uncertainty, after long training) or similar to the input patch (undertrained model, overfitting). Considering that the *swisstopo* dataset was assumed to be more difficult to learn compared to the *OSM*, it can be said that this assumption is reflected in the test prediction results (Section 4.8.3, RQ.2). Finally, when inspecting the prediction results visually, it is impossible to determine whether the prediction comes from a U-Net or a Swin Transformer model. Therefore, it can be stated that in terms of the qualitative evaluation, both models show similarly weak performances when it comes to more difficult prediction tasks (Section 4.8.3, RQ.3).

**IoU Values:** The results in Table 5.7 clearly show that for both data models on both data sources, the U-Net performs better than Swin Transformer in terms of the mIoU (Section 4.8.3, RQ.1).

# Chapter 7

# Conclusion

## 7.1 Contributions

The thesis has shed light on using new deep-learning computation architecture and data models to automate cartographic generalization. On the one hand, the thesis has shown that increased computation power, seemingly better or high-complexity computation models, and more data alone will not solve the issue. For example, the training of a purpose-built Swin Transformer model failed. However, it led to a series of more detailed experiments. These experiments revealed a need for further improvement on how the data is fed to the computation models. More specifically, the thesis showed that a random building selection per patch allows for a long training but not persistently increased model performance. Thus, selecting a data model with a centered placement of the focus building might be the ideal solution. The latter fact points towards a data model that simulates an attention-like mechanism. The said mechanism can be achieved artificially, leveraging what the model should focus on. Thus, the model learns the different generalization tasks on a building level, as shown in the experiments. Moreover, a simple computation model architecture based on mathematical convolutions such as the U-Net might suffice to tackle building generalization. This was shown by comparing the U-Net and Swin Transformer architecture in specifically designed experiments, where the U-Net model outperformed the Swin Transformer. Additionally, the influence of heavily imbalanced training data could be shown in all the experiments. Without exception, all the test sets were randomly sampled prior to defining the training and validation sets. Therefore, it can be assumed that respective test sets represent the data on which the models were trained. Therefore, the fact that all the IoU value distributions from the performed model evaluations are heavily skewed towards a high value of IoU allows the further assumption that simple generalization tasks are significantly over-represented in the data sets, whereas complicated tasks fall short. Therefore the models tend to predict simple tasks quickly and have trouble with more complex cases.

## 7.2 Limitations

It must be stated that a comparison between the U-Net and Swin Transformer computation model performance is not entirely reasonable when the amount of data fed to the model is low, as was the case in the detailed approach phase of the thesis. In order to train a Swin Transformer model from the ground up, much more data is needed. One of the main reasons is the lower inductive bias the Swin

Transformer has compared to U-Net. However, since the task of cartographic generalization differs substantially from usual semantic segmentation tasks, no pre-trained model could be used. Therefore, the possibility of training a purpose-built Swin Transformer model still exists. Moreover, the random selection of the test-set patches might corrupt the performance evaluation. With the random selection, it is impossible to reconstruct a structurally coherent testing site which could prove to be an issue going further in the topic. Lastly, the number of possibilities to perform experiments with various hyperparameters, loss functions, and data models is infinite. Hence, this thesis represents one possible set of results within the maximum time frame of a master's thesis and not a comprehensive collection of experiments.

## 7.3 Learnings

Starting the methodological exploration with a time-intensive and complicated experiment, namely the brute force approach, was a mistake. The better approach would have been to start with small and short experiments regarding the data, data models, and computation models and then finally scale up the best working solution. With the described bottom-up approach, there could have been a possibility of training a purpose-built Swin Transformer model for future use.

## 7.4 Future Research

First, there is a need for further creation of meaningful, balanced training data. The latter could be achieved by accessing a large amount of data and then labeling the single operators at play that generalize single building shapes. Moreover, it would be helpful to have more sophisticated data engineering before training the deep learning models. Specifically, in addition to having operator labels on the building level, it could help to track the geometric complexity of the building shapes. With the help of these added meta-data variables, a yet more comprehensive training set with a better balance of simple and complex generalization examples could be generated. Then, the computation models of choice could be unleashed, and a further attempt to learn cartographic generalization holistically could be undertaken. Second, at some point, the Swin Transformer architecture can outperform U-Net, given enough data. Therefore, a different path could be to run experiments, repeatedly comparing the U-Net and Swin Transformer performance on more extensive and better-engineered training data. Third, regarding the evaluation of the model performance, a substantial effort should be made regarding the performance metric. The IoU has proven helpful as a quantitative metric but lacks to capture qualitative aspects of the prediction results. Therefore, an in-depth search for a more meaningful, purpose-built generalization metric could be a different path to pursue. Specifically, a first step to such an approach could be to look at cases where the IoU between the target and the prediction (TGT-PRED) is low (i.e., weak model performance), but the generalization task for the same sample is simple (high IPT-TGT IoU). Then, it might be possible to address specific issues of the IoU metric or even systematically discover examples where the models struggle. This could improve how the IoU is used for evaluating the prediction performance of building generalization tasks, paving the way to a better understanding of the issue or even developing a new metric.

# Bibliography


Abraham, N. and Khan, N. M. (2019). A Novel Focal Tversky Loss Function With Improved Attention U-Net for Lesion Segmentation. In *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, pages 683–687. ISSN: 1945-8452.

Bader, M. (2001). *Energy minimization methods for feature displacement in map generalization*. Dissertation, Departement of Geography, University of Zürich, Zürich.

Barrault, M., Regnauld, N., Duchene, C., Haire, K., Baeijs, C., Demazeau, Y., Hardy, P., Mackaness, W., Ruas, A., and Weibel, R. (2001). Integrating multi-agent, object-oriented, and algorithmic techniques for improved automated map generalization. In *Proceedings 20th International Cartographic Conference*, pages 2210–2216, Bejing, China.

Beard, K. (1991). Constraints on rule formation. *Map generalization: making rules for knowledge representation*, pages 121–135.

Chollet, F. (2017). *Deep Learning with Python*. Manning, New York, NY, United States, 2021 edition.

Courtial, A., Touya, G., and Zhang, X. (2021). Generative adversarial networks to generalise urban areas in topographic maps. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, 43(B4-2021):15–22.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2020). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *arXiv preprint*.

Dumoulin, V. and Visin, F. (2018). A guide to convolution arithmetic for deep learning. *arXiv preprint*, pages 1–31.

Feng, Y., Thiemann, F., and Sester, M. (2019). Learning cartographic building generalization with deep convolutional neural networks. *ISPRS International Journal of Geo-Information*, 8(6).

Fu, C. (2022). Datamodels for DL in Building Generalization. Departement of Geography, University of Zürich.

Ge, Y., Zhang, X., Atkinson, P. M., Stein, A., and Li, L. (2022). Geoscience-aware deep learning: A new paradigm for remote sensing. *Science of Remote Sensing*, 5(April):100047–100047. Publisher: Elsevier B.V.

Giang, T. L., Dang, K. B., Toan Le, Q., Nguyen, V. G., Tong, S. S., and Pham, V.-M. (2020). U-Net Convolutional Networks for Mining Land Cover Classification Based on High-Resolution UAV Imagery. *IEEE Access*, 8:186257–186273. Conference Name: IEEE Access.

Giurgi, D.-V., Josso-Laurain, T., Devanne, M., and Lauffenburger, J.-P. (2022). Real-time road detection implementation of UNet architecture for autonomous driving. In *2022 IEEE 14th Image, Video, and Multidimensional Signal Processing Workshop (IVMSP)*, pages 1–5, Nafplio, Greece. IEEE.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press, Cambridge, MA, United States.

Harrie, L. and Weibel, R. (2007). Modelling the Overall Process of Generalisation. *Generalisation of Geographic Information*, pages 67–87.

Jadon, S. (2020). A survey of loss functions for semantic segmentation. *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology, CIBCB 2020*.

Kang, Y., Rao, J., Wang, W., Peng, B., Gao, S., and Zhang, F. (2020). Towards Cartographic Knowledge Encoding with Deep Learning: A Case Study of Building Generalization. *The 23rd International Research Symposium on Cartography and GIScience*, (2019):1–6.

Khan, S., Naseer, M., Hayat, M., and Zamir, S. W. (2021). Transformers in Vision: A Survey. *ACM Computing Surveys (CSUR)*, pages 1–30.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lee, J., Jang, H., Yang, J., and Yu, K. (2017). Machine Learning classification of buildings for map generalization. *ISPRS International Journal of Geo-Information*, 6(10).

LeNail, A. (2019). NN-SVG: Publication-Ready Neural Network Architecture Schematics. *Journal of Open Source Software*, 4(33):747–747.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. (2021). Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision.*

Mackaness, D. W. A. (1995). A Constraint Based Approach to Human Computer Interaction in Automated Cartography. In *Proceedings of the 17th International Cartographic Conference*, pages 1423–1432, Barcelona, Spain.

Mai, G., Janowicz, K., Hu, Y., Gao, S., Yan, B., Zhu, R., Cai, L., and Lao, N. (2022). A review of location encoding for GeoAI: methods and applications. *International Journal of Geographical Information Science*, 00(00):1–35. Publisher: Taylor & Francis.

McMaster, R. B. and Stuart Shea, K. (1992). Generalization in Digital Cartography. *Association of American Geographers*.

Nickerson, B. G. (1986). Development of a rule-based system for automatic map generalization. In *Proceedings of the 2nd International Symposium on Spatial Data Handling*, Seattle, WA.

Nickerson, B. G. (1988). Automated cartographic generalization for linear features. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 25(3):15–66.

Patel, D. (2020). Image classification vs Object detection vs Image Segmentation | Deep Learning Tutorial 28.

Persson, A. (2020). Pytorch Common Mistakes - How To Save Time.

Powitz, B. M. (1993). Computer-Assisted Generalization - An Important Software-Tool in GIS. *International Archives of Photogrammetry and Remote Sensing*, 29:664–672.

Ramsundar, B. and Zadeh, R. B. (2018). *TensorFlow for Deep Learning: From Linear Regression to Reinforcement Learning*. O'Reilly Media, Inc., 1st edition.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In Navab, N., Hornegger, J., Wells, W. M., and Frangi, A. F., editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham. Springer International Publishing.

Ruas, A. (2001). Automatic Generalisation Project: Learning Process from Interactive Generalisation. Technical Report 39.

Ruas, A. and Plazanet, C. (1996). Strategies for automated generalization. In *Proceedings of 7th International Symposium on Spatial Data Handling*, volume 1. Issue: 6.

Schylberg, L. (1993). *Computational methods for generalization of cartographic data in a raster environment*. Dissertation, Department of Geodesy and Photogrammetry, Royal Institute of Technology, Stockholm.

Sester, M., Feng, Y., and Thiemann, F. (2018). Building generalization using deep learning. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, 42(4):631–637.

Shelhamer, E., Long, J., and Darrell, T. (2014). Fully Convolutional Networks for Semantic Segmentation. *arXiv preprint*, arXiv:1411.

Solórzano, J. V., Mas, J. F., Gao, Y., and Gallardo-Cruz, J. A. (2021). Land Use Land Cover Classification with U-Net: Advantages of Combining Sentinel-1 and Sentinel-2 Imagery. *Remote Sensing*, 13(18):3600. Number: 18 Publisher: Multidisciplinary Digital Publishing Institute.

Spiess, E., Baumgartner, U., Arn, S., and Vez, C. (2002). Swiss Society of Cartography Topographic Maps. *Cartographic Publication Series*, (17).

Steiniger, S., Lange, T., Burghardt, D., and Weibel, R. (2008). An approach for the classification of urban building structures based on discriminant analysis techniques. *Transactions in GIS*, 12(1):31–59.

Sugirtha, T. and Sridevi, M. (2022). Semantic Segmentation using Modified U-Net for Autonomous Driving. In *2022 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*, pages 1–7.

Taghanaki, S. A., Zheng, Y., Zhou, S. K., Georgescu, B., Sharma, P., Xu, D., Comaniciu, D., and Hamarneh, G. (2019). Combo Loss: Handling Input and Output Imbalance in Multi-Organ Segmentation. arXiv:1805.02798 [cs].

Tobin, J. (2019). Why you should always overfit a single batch to debug your deep learning model.

Touya, G., Zhang, X., and Lokhat, I. (2019). Is deep learning the new agent for map generalization? *International Journal of Cartography*, 5(2-3):142–157.

Tran, L.-A. and Le, M.-H. (2019). Robust U-Net-based Road Lane Markings Detection for Autonomous Driving. In *2019 International Conference on System Science and Engineering (ICSSE)*, pages 62–66. ISSN: 2325-0925.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Plolsukhin, I. (2017). Attention Is All You Need. In *Advances in Neural Information Processing Systems*. Issue: 30.

Wang, J., Yang, M., Chen, Z., Lu, J., and Zhang, L. (2022). An MLC and U-Net Integrated Method for Land Use/Land Cover Change Detection Based on Time Series NDVI-Composed Image from PlanetScope Satellite. *Water*, 14(21):3363. Number: 21 Publisher: Multidisciplinary Digital Publishing Institute.

Weibel, R. (1991). Amplified Intelligence and Knowledge-Based Systems. In Buttenfield, B. P. and McMaster, R. B., editors, *Map Generalization: Making Rules for Knowledge Representation*, pages 172–186. Longman, London.

Weibel, R. (1995). Three essential building blocks for automated generalization. In Müller, J.-C., Lagrange, J.-P., and Weibel, R., editors, *GIS and Generalization*, pages 56–69. Taylor & Francis, London.

Weibel, R. and Dutton, G. (1999). Generalising spatial data and dealing with multiple representations. In Longley, P., Goodchild, M., Maguire, D., and Rhind, D., editors, *Geographical Information Systems: Principles, Techniques, Management and Applications*, pages 125–155. John Wiley, Chichester, second edi edition.

Xie, X., Ye, L., Kang, X., Yan, L., and Zeng, L. (2022). Land Use Classification Using Improved U-Net in Remote Sensing Images of Urban and Rural Planning Monitoring. *Scientific Programming*, 2022:e3125414. Publisher: Hindawi.

Xu, Z., Zhang, W., Zhang, T., and Yang, Z. (2021). Efficient Transformer for Remote Sensing Image Segmentation. *Artificial Intelligence Algorithm for Remote Sensing Imagery Processing*, pages 1–24.

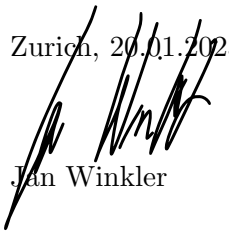
Yang, L. and You, C. (2018). Instance U-Net and Watershed: Improved Segmentations for breast cancer cells. 12353505, Stanford.

Yi-de, M., Qing, L., and Quan, Z.-b. (2004). Automated image segmentation using improved PCNN model based on cross-entropy. In *Proceedings of 2004 International Symposium on Intelligent Multimedia, Video and Speech Processing, 2004.*, pages 743–746, Hong Kong, China. IEEE.

Zhang, A., Lipton, Z. C., Li, M., and Smola, A. J. (2021). *Dive into Deep Learning*, volume arXiv:2106. arxiv.

**Personal declaration** I hereby declare that the submitted Thesis is the result of my own, independent work. All external sources are explicitly acknowledged in the Thesis.

Zurich, 20.01.2023



Jan Winkler