ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

IKG
Institut für Kartografie und Geoinformation
Institute of Cartography and Geoinformation

Boosting Semantic Segmentation of Historical Maps with Self-Supervised Vision Transformers

Project 2

Spring Semester 2023

Geomatics Master

Author: Shupeng Wang
shupwang@student.ethz.ch

Professorship: Prof. Dr. Lorenz Hurni
LHurni@ethz.ch

Supervisor: Xue Xia xiaxue@ethz.ch
Chenjing Jiao cjiao@ethz.ch

Submission Date: 16.06.2023

Abstract

Historical maps have long been valuable resources due to the rich amount of information they offer about the built environment and landscapes in the past. And to enjoy the benefits brought by the rich information, efficient ways to extract information from the maps are required. Current mainstream methods originate from the field of computer vision and use techniques such as Convolutional Neural Networks (CNNs) to achieve classification or semantic segmentation with relatively high accuracy. In the meantime, the growing popularity of the transformers and new self-supervised methods such as contrastive learning seem to point out a new direction to go. This project combines a Swin Transformer based model with contrastive learning aiming to boost the performance of semantic segmentation of historical maps and achieve more efficient information extraction from historical maps. The highlight of the project lies in the following perspectives: a) historical map data preprocessing b) pretraining of the backbone with contrastive learning using multi-temporal large-scale unlabeled historical map data c) finetuning of the segmentation head with the small-scale labeled dataset.

Table of Contents

Table of Figures

1. Introduction

1.1. Background and motivation

Historical maps have long been valuable sources of information in the past where they provide references ranging from the natural environment (natural features and topography) to the built environment (buildings and infrastructures). Yet, it was always difficult to extract useful information from these maps due to the great amount of expert knowledge and effort required during the early times. The quick development of deep learning techniques in the field of computer vision such as Convolutional Neural Networks (CNN), however, has been a game changer and brought the use of historical maps to a new dimension. They have significantly reduced the complexity and difficulty to extract information from these maps, as well as achieved relatively high accuracy in tasks such as classification and semantic segmentation.

In recent years, two new emerging techniques have once again revolutionized the field of computer vision and raised the efficiency and accuracy of downstream tasks to a new height. One of them is self-supervised learning (SSL) which explores the inherent structure of non-labeled data to create labels, and based on these generated labels, tries to learn the useful feature representations. The other one is the transformer-based model that adopts the mechanism of attention instead of convolution to promote the learning of the feature representation. While these techniques have been widely applied in computer vision, there's still limited application in the field of historical maps. Would they bring a thorough change to the field just like CNNs did before? How to incorporate these new techniques to the field of historical maps in order to achieve better performance in downstream tasks? All these questions remain open and will form a key focus of this project.

1.2. Objective

The main objective of this project is to explore whether the involvement of self-supervised learning and the transformer-based model can help boost the performance of semantic segmentation, specifically for railway features in historical maps. The focus is given to contrastive learning and the Swin Transformer. The detailed implementation can be summarized as a primary goal and two secondary goals.

1.2.1. Primary goal

The primary goal is to automize the map data preprocessing flow, incorporate contrastive learning into the Swin Transformer framework, and create a full workflow to conduct the semantic segmentation of railway features in historical maps. The main goal covers the basic requirements that a deep learning task should generally fulfill: automatic data preprocessing flow and a runnable model.

1.2.2. Secondary goals

The secondary goals are divided into the following two sessions:

a) Contrastive learning

Explore whether the overall performance of semantic segmentation of railways can be improved by involving additional pretraining of the backbone of the model using contrastive learning and a large unlabeled historical map dataset.

b) Swin Transformer

Explore whether the overall performance of semantic segmentation of railways can be improved by incorporating the Swin Transformer based backbone and decoder compared to the traditional UNet model with ResNet backbone.

2. Related work

2.1. Self-supervised learning

Self-supervised learning (SSL) is a machine learning technique that exploits the inherent information present in the unlabeled data to provide supervision signals for learning feature representations. Most SSL methods choose one of two popular approaches: conducting pretext tasks or utilizing contrastive learning.

For pretext-based methods, different tasks such as context prediction (Doersch et al., 2015), Jigsaw solving (Noroozi & Favaro, 2016), colorization of grey-scale images (Zhang et al., 2016), and cluster prediction (Caron et al., 2020) have been developed.

For contrastive learning based methods, many frameworks have been proposed. The two most popular frameworks are MoCo (He et al., 2019) and SimCLR (T. Chen, Kornblith, Norouzi, et al., 2020). MoCo builds up on the early InstDisc model (Wu et al., 2018) and summarizes contrastive learning as a dictionary look-up task. The idea of queue and momentum encoder helps to build up a large and consistent dictionary on-the-fly that facilitates contrastive learning. SimCLR, on the other hand, chooses a simple structure that uses data augmentation to generate positive pairs. The model can be easily set up with an encoder and a fully connected layer. Learning from the strength of each other, MoCov2 (X. Chen et al., 2020) and SimCLRv2 (T. Chen, Kornblith, Swersky, et al., 2020) come out and lead another wave of contrastive learning. Interestingly, contrastive learning can also be conducted without negative samples while still avoiding model collapse. BOYL (Grill et al., 2020) introduces a predictor and converts the former maximizing agreement problem to a prediction problem where mean squared loss can be used. To reduce the overall complexity of the model picked in this project, SimCLR, as a simple contrastive learning framework, is picked and used for pretraining.

2.2. Vision Transformers

The success of Transformers in the field of natural language processing has sparked interest in applying the same technique in the field of computer vision. Following the trend comes out the Vision Transformer (ViT) (Dosovitskiy et al., 2020). In contrast to CNNs, ViT uses self-attention instead of convolution so that both local and global contextual information can be captured together during the training process. At the same time, ViT is also facing problems such as difficulty in capturing information of entities with different scales, and the requirement of large amount of computation power. To resolve these issues, Swin Transformer is developed by combining a shift-window strategy from CNNs with Transformers (Liu et al., 2021) It divides the image into non-overlapping windows and performs self-attention within each window to improve computational efficiency. The hierarchical structure ensures information from multi-scale can be captured properly. Due to its brilliant performance in various tasks, Swin Transformer is used as the backbone in this project.

2.3. Semantic segmentation

Semantic segmentation is one of the most fundamental downstream tasks in the field of computer vision with the objective to achieve pixel-wise classification for given images. There are many sophisticated CNNs such as UNet (Ronneberger et al., 2015) and FCN (Long et al., 2014) available. Among them, UNet, with its simplicity and robustness has received a lot of attention. One way to improve the performance of the UNet is to replace its encoder with more robust backbones. Attempts have been made by replacing it with ResNet (Diakogiannis et al., 2019), Swin Transformer (Cao et al., 2021), and many other encoders. This project utilizes these improved versions of UNet and combines them with self-supervised learning to help boost the performance in semantic segmentation.

3. Methodology

3.1. Data

3.1.1. Data overview

The data used in this project mainly involve four parts: Siegfried Map data, Old National Map data, railway data for the Siegfried Map, and road data for the Siegfried Map. All the data are provided exclusively by the Chair of Cartography, Institute of Cartography and Geoinformation at ETH Zurich.

3.1.1.1. Siegfried Map data

The Siegfried Map is a series of topographic maps of Switzerland that was created between 1870 and 1926 and continued getting updated until 1949 (Swisstopo, 2023b). The series consists of several thousand sheets providing coverage of the whole country. The map sheets contain a wide range of information, including relief, waterways,

forests, settlements, and transportation networks such as roads, railways, and telegraph lines. In this project, digital versions of Siegfried Map sheets (dimension: 7000*4800*3) are used. The map series is divided into three subsets according to the period: 1880s, 1890s, and 1940s. Sheets from the 1940s are used in the pretraining of the encoder and sheets from the other two time periods are for the finetuning of the segmentation head.

3.1.1.2. Old National Map data

The Old National Map is another series of maps of Switzerland that superseded the earlier Siegfried Map starting in the 1930s (Swisstopo, 2023a). Just like the Siegfried Map series, the Old National Map series provides coverage of the whole country as well as information about natural features, human settlements, and transportation networks. Within the scope of the project, digital versions of Old National Map sheets dating back to the 1950s (dimension: 14000*9600*3) are used for the pretraining of the encoder.

3.1.1.3. Railway data for the Siegfried Map

Digitized railway data from Siegfried Map sheets dating back to the 1880s and 1890s are utilized. This data is available in vector format as polylines and consists of three categories of railways: normal railway, narrow railway, and tunnel railway. The railway features are used as references to extract small railway tiles from the Siegfried Map sheets and generate corresponding labels. The generated small tiles and labels are mainly for the finetuning of the segmentation head.

3.1.1.4. Road data for the Siegfried Map

In order to enhance the model's ability to distinguish railways from roads, additional road data is incorporated into the training process. This dataset consists of digitized road data for four selected Siegfried Map sheets. The road data is available in vector format, specifically as points that either fall directly on the road network or within a certain distance of it. The road networks are used to extract road tiles from Siegfried map sheets. These road tiles, along with the corresponding labels generated from the road data, are utilized for the finetuning of the segmentation head.

3.1.2. Data preprocessing

Data preprocessing is divided into two parts which correspond to the model structure mentioned later in section 3.2: preprocessing for pretraining data and preprocessing for finetuning data.

3.1.2.1. Preprocessing for pretraining data

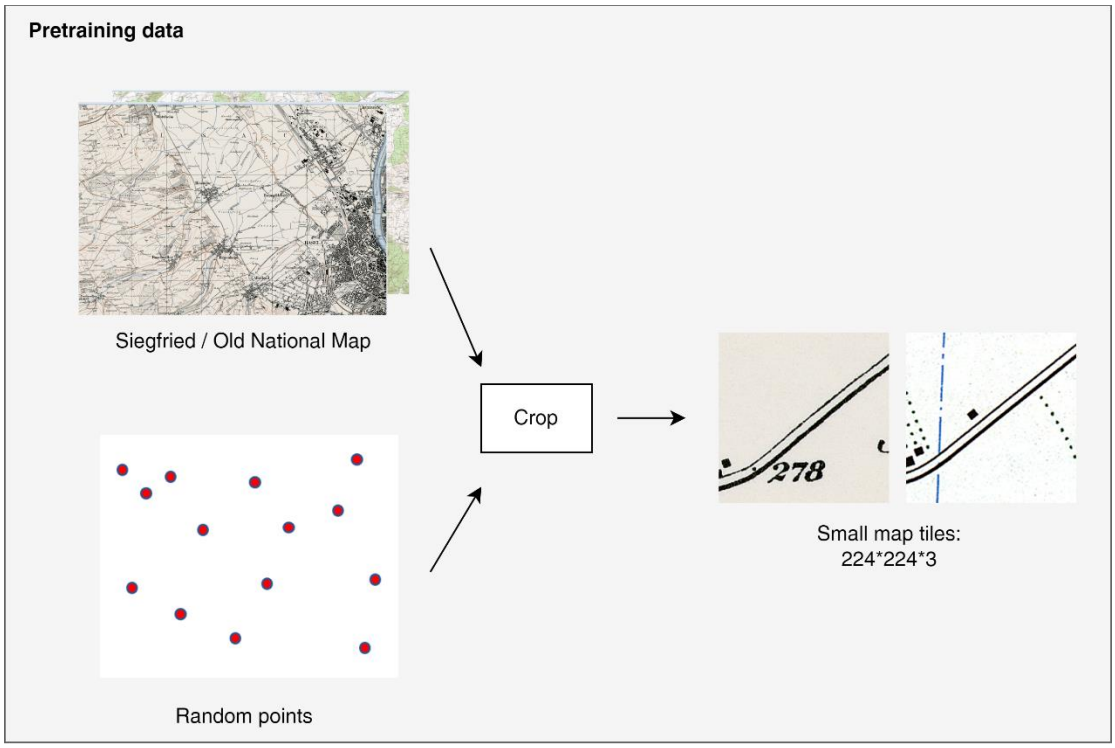

Figure 1: Data preprocessing workflow (pretraining data)

The whole procedure for preprocessing pretraining data is summarized in Figure 1. The input data are Siegfried Map sheets from the 1940s, and Old National Map sheets from the 1950s. For each Siegfried Map sheet, 700 random points are generated within its extent, resulting in a total of 240,800 points. These points are then used to crop both Siegfried and Old National Map sheets, creating the same number of small map tiles with a dimension 224*224*3.

3.1.2.2. Preprocessing for finetuning data

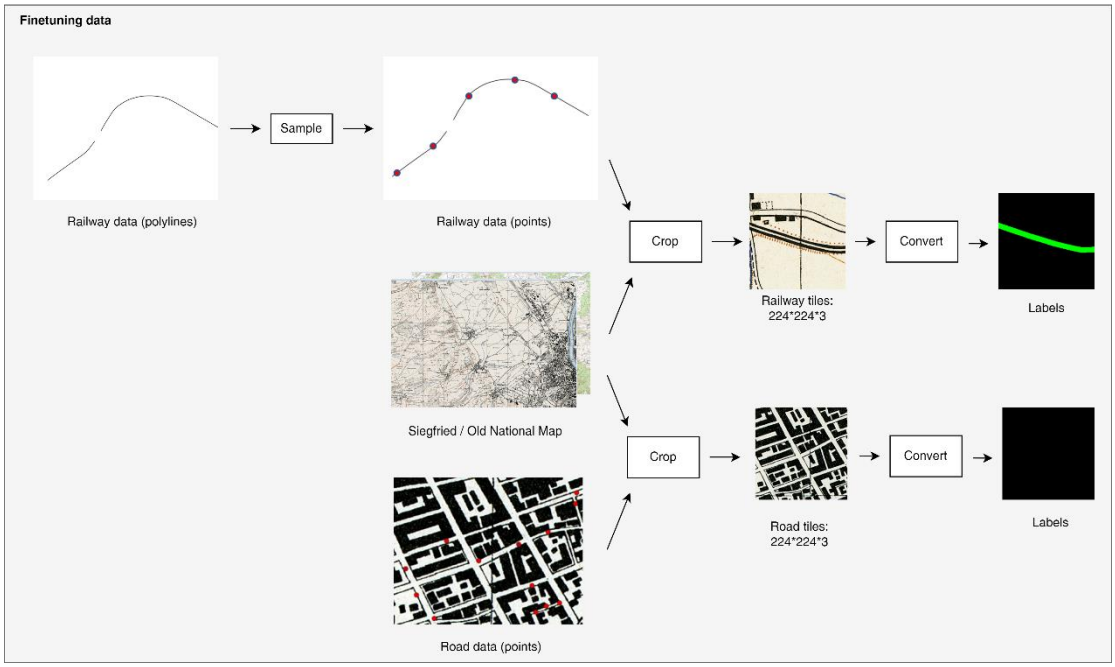

Figure 2: Data preprocessing workflow (finetuning data)

The preprocessing procedure for finetuning data is depicted in Figure 2. For railway data, the preprocessing involves the following steps:

1. Railway Data Sampling: The railway data is sampled as points, ensuring that the distance between consecutive sample points on the same railway is 200 meters. This sampling process generates a total of 25,945 points.
2. Map Sheet Cropping: The sampled railway points are used to crop Siegfried Map sheets from the 1940s and Old National Map sheets from the 1950s. These cropped map sheets result in the creation of the same number of small map tiles that contain railway features.
3. Label Generation: Corresponding labels are generated for each railway tile based on the railway data. The labels are stored as RGB images, where the red channel represents narrow railway labels, the green channel represents normal railways, and the blue channel represents tunnel railways. The value of 1 for a pixel in a given channel means that it belongs to the railway class represented by the channel while 0 means it doesn't.
4. Data split: the railway tiles and corresponding labels are divided into training, validation, and testing sets in an 8:1:1 ratio.

A similar procedure to the railway data is applied to the road data (sampling is unnecessary in this case since road data is already point data). The road data is also sampled as points, and the resulting points are used to generate road tiles and labels. In total, 4,557 road tiles and labels are generated and they are split into three sets in an 8:1:1 ratio

Dataset	Training			Validation			Testing		
	Railway	Road	Total	Railway	Road	Total	Railway	Road	Total
Complete	20756	3645	**24401**	2594	455	**3049**	2595	457	**3052**
Partial (10%)	2075	364	**2439**	259	45	**304**			
Partial (5%)	1037	182	**1219**	129	22	**151**			
Partial (2.5%)	518	91	**609**	64	11	**75**			
Partial (1%)	207	36	**243**	25	4	**29**			

Table 1: Overview of finetuning dataset

Additionally, small finetuning datasets are prepared by taking 10%, 5%, 2.5%, and 1% of the railway and road data generated in the previous steps. Table 1 shows the detailed number of tiles in each dataset. To have a fair evaluation of the model performance, models trained and validated with different datasets are all tested on the testing set for the complete dataset.

3.2. Model

The model proposed in this project is inspired by the SwinUNet SSL model developed by Scheibenreif et al. (2022). The overall structure follows a UNet setting which is an encoder-decoder structure with skip connections. The main blocks, however, are replaced by Swin Transformer blocks instead of typical convolutional layers. In addition, the encoder is first pretrained through contrastive learning, then connect with the decoder to conduct semantics segmentation. The model performance is compared with a baseline model that follows the UNet structure but with a Resnet backbone.

3.2.1. SwinUNet SSL

The SwinUNet SSL model can be divided into three parts: Swin Transformer based backbone, self-supervised pretraining, and finetuning. The details for each part are covered in the following sections.

3.2.1.1. Swin Tranformer based backbone

The Swin Transformer plays a vital role in this project as the model backbone for extracting meaningful features from original historical maps. It is a general-purpose backbone designed to tackle various vision tasks, including image classification, semantic segmentation, and object detection. The Swin Transformer offers two significant advantages. First, it excels at capturing features at multiple scales. By creating hierarchical feature maps starting from small-scale patches and progressively merging adjacent patches, the Swin Transformer achieves an expanded receptive field and a better understanding of features on a larger scale. Another notable advantage of the Swin Transformer is its computational efficiency. The shifted window partitioning technique allows for the computation of self-attention locally, reducing the computational complexity to a linear scale relative to the image size. This efficiency is achieved while still maintaining the crucial interaction between windows in the preceding layers, ensuring comprehensive feature representation (Liu et al., 2021).

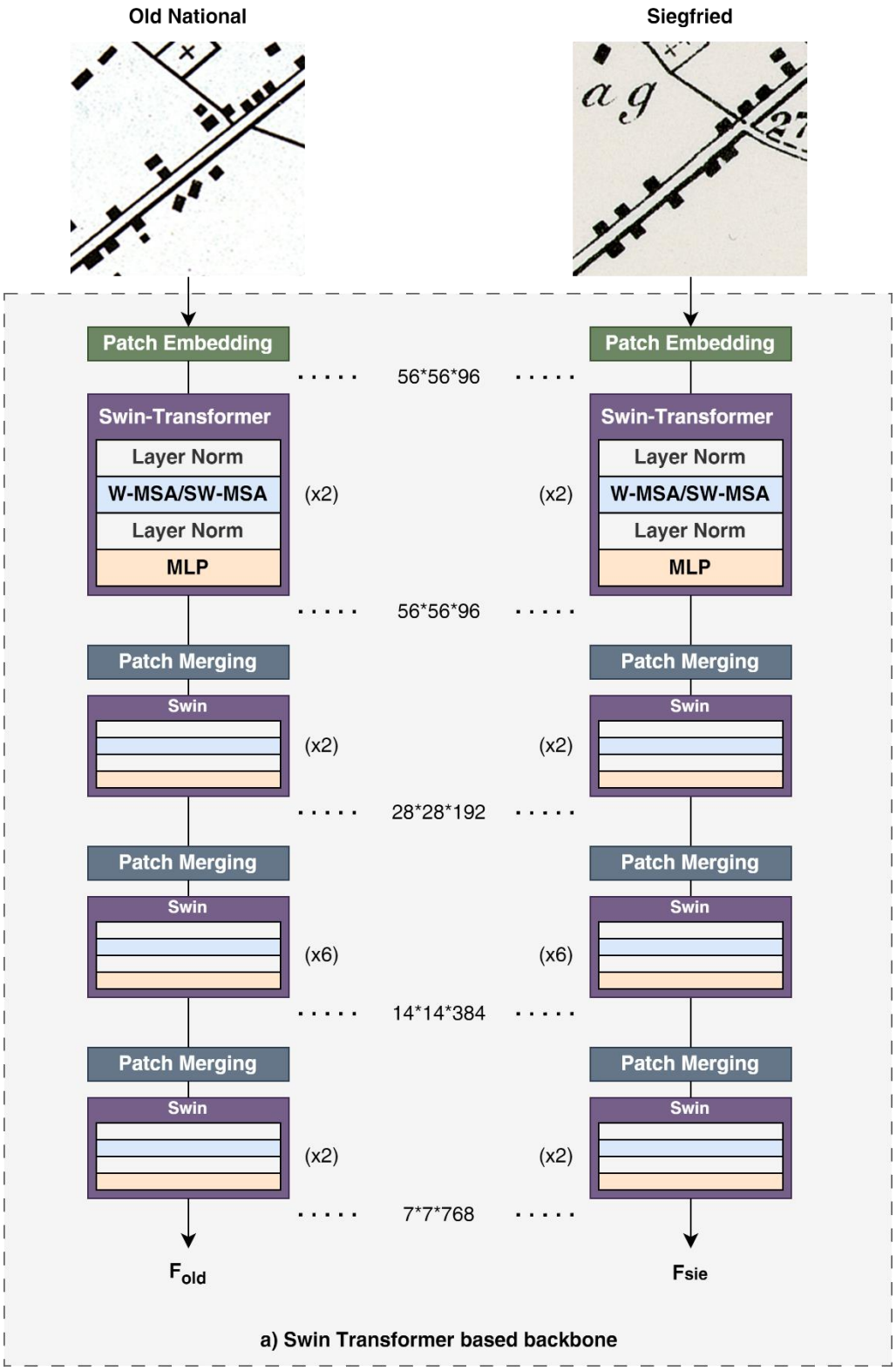

Figure 3: Network architecture for the Swin Transformer based backbone

The backbone of the model consists of several crucial components as indicated in Figure 3: patch embedding, Swin Transformer blocks, and patch merging. These components work together to process the input tiles and extract meaningful features. The overall backbone architecture follows the structure of the tiny version of the Swin

Transformer (Swin-T), where Swin Transformer blocks are divided into four groups and repeated 2, 2, 6, and 2 times within each group.

The first component, path embedding, combines patch partitioning and linear embedding. The input tiles are divided into small patches with dimensions of 56*56*48. These patches are then reprojected to a suitable dimension of 56*56*96, which serves as the input for the Swin Transformer. This patch embedding process is accomplished through a single convolutional layer.

The Swin Transformer blocks form the core of the backbone. Each block comprises four components: two layer norm modules (LN), a multi-head self-attention module with either regular windowing configuration (W-MSA) or shifted windowing configuration (SW-MSA), and a two-layer multilayer perceptron (MLP). To achieve global perception, one Swin Transformer block with W-MSA is always followed by another block with SW-MSA. Importantly, the Swin Transformer blocks maintain the input and output dimensions.

Patch merging, similar to downsampling, reduces the height and width of the input patch by half while doubling its channel dimension. This process helps to capture hierarchical features.

3.2.1.2. Self-supervised pretraining

The use of the contrastive SSL for the pretraining of the Swin Transformer based backbone is also proposed in the project. There are many contrastive learning frameworks available, and the framework picked up is a modified version of SimCLR (Figure 4).

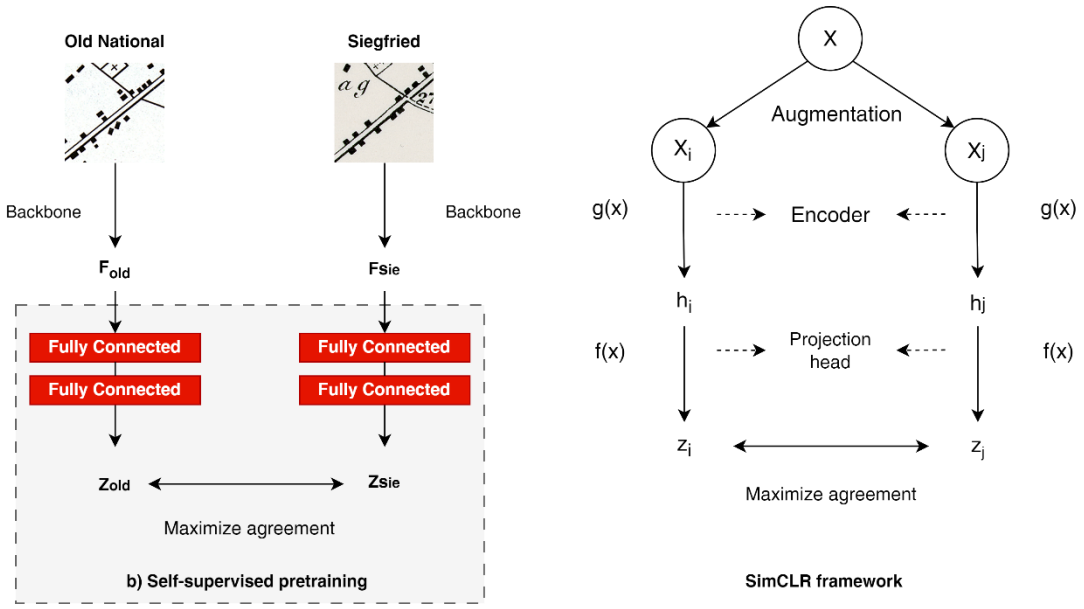

Figure 4: Comparison between the self-supervised pretraining and SimCLR framework

SimCLR adopts the typical idea of contrastive learning by maximizing the agreement

between positive pairs while minimizing negative pairs. It obtains positive pairs X_i and X_j through data augmentation such as image transformation, color jittering, etc. Two augmented views of the input data are generated, then each of them enters an encoder that has shared weights for feature extraction. A further projection is conducted for extracted features h_i and h_j and the InfoNCE loss is calculated between results z_i and z_j to maximize the agreement and update the corresponding weights (T. Chen, Kornblith, Norouzi, et al., 2020).

The modified version of SimCLR in the SwinUNet SSL inherits the idea of the classical SimCLR. The encoder used is the backbone described in section 3.2.1.1, and the projection is achieved by two consecutive fully connected (FC) layers. Two highlights that differentiate it from the classical version are:

1. Data argumentation free: Instead of using data augmentation, tiles from Siegfried Map and Old National Map are directly considered positive pairs. The Old National Map provides roughly the same coverage of the area as the Siegfried Map, despite being created in different years. By selecting the closest possible time frames for the two map series, the dissimilarity of features between the maps can be minimized. In addition to the style differences between the map series, the symbology used in the maps is also different. This difference in symbology can potentially force the model to learn some fundamental representations for specific features such as railways, buildings, or roads.

2. Non-shared weights: The encoder for Siegfried map tiles and the encoder for Old National map tiles are only the same in the structure but not for the weights. The aim of the pretraining in this project is not to train a backbone that can learn a general representation of map features but to better learn the representation for a specific map series. Therefore, each encoder will have a different group of weights that is tailored toward the specific map set.

3.2.1.3. Finetuning

The finetuning, as the main part of the model, follows the SwinUNet architecture. The encoder is the Swin Transformer based backbone mentioned in section 3.2.1.1. After pretraining, meaningful representations will be extracted from Siegfried Map tiles and then used as the input for the segmentation head. The whole finetuning procedure is summarized in Figure 5.

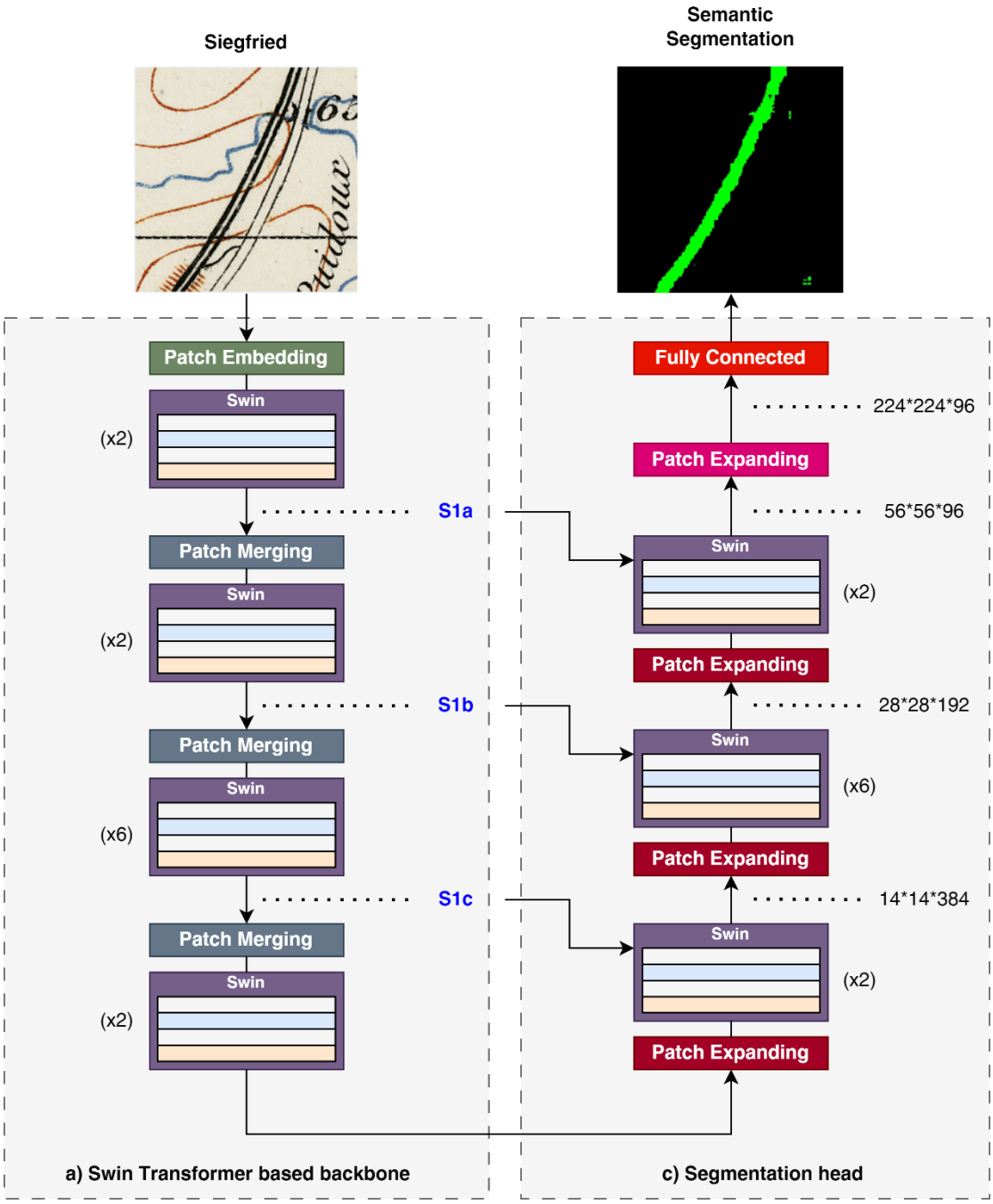

Figure 5: Network architecture for the segmentation head

The segmentation head consists of four components: path expanding, Swin Transformer blocks, skip connection, and a fully connected layer. These components work together to convert the extracted feature into semantic segmentation of railway features. As a part of the SwinUNet, the segmentation also follows the Swin-T structure.

The first component, path expanding, is a reverse operation of the patch merging that conducts upsampling tasks. For each patch expanding, it doubles the height and the width of the input feature while reducing the channel by half. The last patch expanding is a little bit special as it quadruples the height and the width instead of doubling.

The Swin Transformer blocks are the same as those in the backbone. Yet, an additional step of skip connection is applied after the Swin Transformer blocks. Shallow features from the backbone are concatenated with deep features in the segmentation head that owns the same dimension to reduce the loss of spatial information caused during the patch merging. To keep the dimension of the concatenated feature the same as the upsampled feature, an additional fully connected layer is applied.

After four patch expanding blocks, the feature now has the same height and width as the input map tile. A final fully connected layer is applied to bring the number of channels to the number of railway classes. In the case of this project, there are three railway classes.

The finetuning process is supervised learning where labels are provided. The binary cross entropy loss (BCE loss) is calculated between each label and the corresponding result of semantic segmentation, and the weights are updated accordingly. One noticeable point is that the weights for the backbone are not frozen, and they can also be updated during the finetuning process.

3.2.2. UNet

The baseline model chosen in this project is the mainstream model: UNet. One issue with classical UNet is that it lacks pretrained weights on large datasets such as ImageNet for its encoder. In addition, it fails to reach roughly the same level of model complexity as the SwinUNet SSL. To ensure a fair comparison between the SwinUNet SSL and the baseline model, the UNet is modified accordingly. The encoder is replaced by the ResNet50 which has sophisticated pretrained weights on large datasets as well as similar complexity as Swin-T (Liu et al., 2021).

3.3. Procedure

The whole procedure can be summarized into pretraining, finetuning, and testing. Before starting any of the above steps, both Old National Map and Siegfried Map sheets have to go through the data preprocessing procedure to get transformed into small tiles with dimensions of 224*224*3 to match the input format of the model.

3.3.1. Pretraining

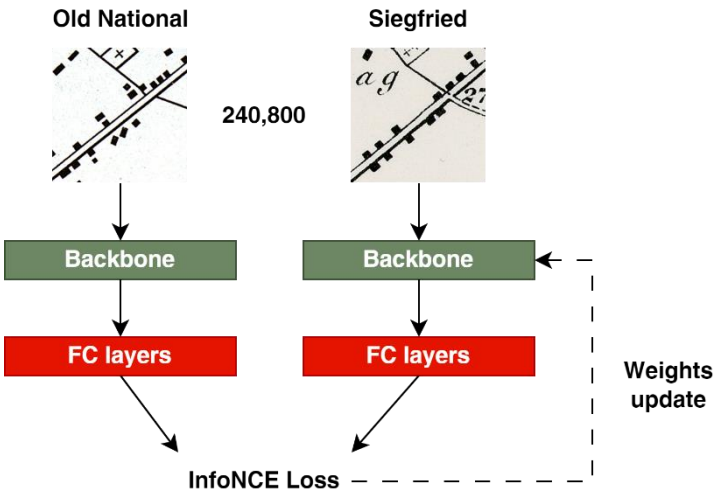

Figure 6: Pretraining workflow

Figure 6 shows that 240,800 Siegfried Map tiles and the same number of Old National Map tiles are used as the input for the pretraining. The tiles go through the backbone and fully connected layers, and the InfoNCE loss is calculated between the generated features. The weights for both the backbone and fully connected layers will be updated accordingly, but only the weights for the backbone will be passed to further steps.

For training parameters, the batch size is set to 96 meaning that 95*2 negative examples will be available per positive pair within a batch. The learning rate is set to be constant at 10^{-4} and the scheduler is a cosine scheduler that starts to work after 30 epochs. To ensure a good start, the pretrained weights for Swin-T are loaded (Liu Z et al., 2023).

3.3.2. Finetuning

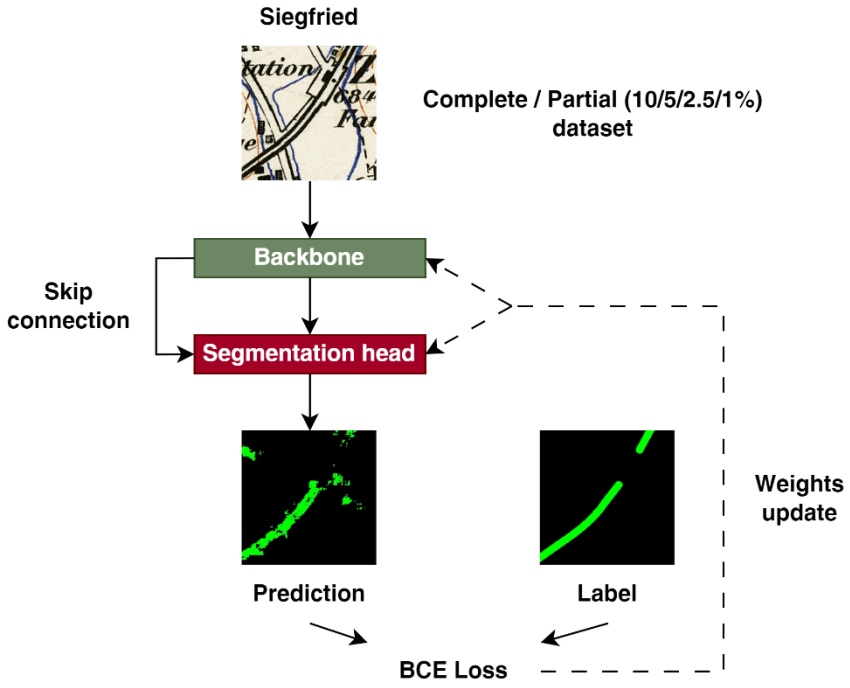

Figure 7: Finetuning workflow

For finetuning, the prepared complete or partial finetuning datasets are utilized. Siegfried tiles will go through the backbone and the segmentation head to generate the semantic segmentation of railway features, and the BCE loss is calculated between the generated features and given labels. The weights for both the backbone and segmentation head will be updated.

For finetuning parameters, the batch size is 48. The learning rate is set to be constant at 10^{-4} and the scheduler starts to work after 50 epochs with a decay rate of 10^{-3}. Experiments are conducted on all five datasets using two different approaches: fine-tuning with pretrained weights for the backbone and training from scratch.

3.3.3. Testing

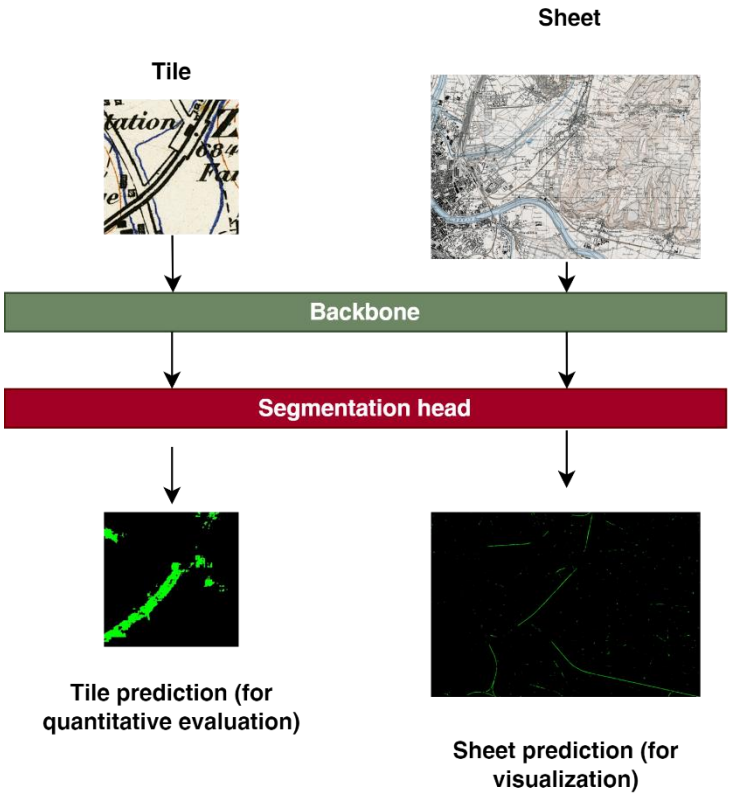

Figure 8: Testing workflow

In the testing phase, the trained model is evaluated on two different levels: map tiles and map sheets. For map tiles, a total of 3052 testing tiles are used to assess the performance of the model. The evaluation focuses on each railway class individually, and a confusion matrix is created to analyze the model's predictions. The confusion matrix includes metrics such as true positive, false positive, true negative, and false negative rates indicating whether a certain type of railway is correctly classified at the pixel level. In addition to the confusion matrix, several evaluation metrics are also calculated. These include accuracy which measures the overall correctness of the model's predictions, F1 score, which considers both precision and recall, and Intersection over Union (IoU), which evaluates the overlap between the predicted and ground truth regions.

For map sheets, the predictions generated by the model are primarily used for visualization purposes. They showcase how the model's capabilities can be applied in real-world applications, where the entire map sheets are processed to highlight the presence and locations of different railway classes.

3.4. Tools

Deep learning tasks are inherently complex and involve a wide range of preparation including data processing, model building, to evaluation. Making informed decisions regarding the tools such as programming language, deep learning framework, and hardware can significantly accelerate the process while ensuring good quality, transferability, and reproducibility.

3.4.1. Hardware

Computers with GPU installed:

Tasks related to deep learning generally require a computer with decent computing power in order to deal with the large amount of data during the training, validation, and testing process. Especially for image data, GPUs with high memory are preferred. Regarding the model training in this project, the pretraining task, due to the large amount of data required, is conducted in a computer with NVIDIA TESLA V100-PCIE which has 32 GB RAM. All other tasks including finetuning and testing are performed in a computer installed with NVIDIA RTX A4000 GPU that has 16 GB RAM.

3.4.2. Software

The Software used in the project mainly involves two parts: a) QGIS which is mainly used as a data viewer or to perform simple subset tasks, b) Python which is the major programming language for all data preprocessing, model building and testing, and result evaluation.

a) QGIS:

QGIS is an open-source GIS platform that allows easy working with geospatial data. The platform not only supports numerous formats of data including vector, raster, and image but also provides a great variety of functions for geoprocessing and spatial analysis. In the scope of this project, QGIS is mainly used to select the road data for finetuning. Also, it serves as a data viewer to visualize the railway and road data and to locate the generated tiles.

b) Python:

Python has long been a popular programming language for data science and machine learning. There are plenty of well-developed packages such as Numpy and Pandas which allow easy data reading and processing, OpenCV and Pillow that deal with image processing and computer vision, and PyTorch and TensorFlow that offer machine learning frameworks.

Among all libraries utilized in this project, PyTorch is one of the most important libraries. PyTorch significantly simplifies deep learning workflow by decomposing it

into multiple components such as datasets, dataloaders, and neural network modules. Built-in transformations, optimizers, schedulers, and loss functions are also provided eliminating the need for manual implementation (PyTorch Foundation, 2023). The whole model in this project is built on the PyTorch framework, and the features mentioned above have greatly accelerated both model building and the training process.

4. Result

4.1. Loss change

Figure 9: Loss (pretraining)

In the pretraining phase, both the training loss and validation loss exhibit a rapid drop in the initial epochs, as shown in Figure 9. After around epoch 50, the drop speed of the loss starts to slow down, yet it's worth noticing that the trend of decreasing loss continues even up to epoch 400.

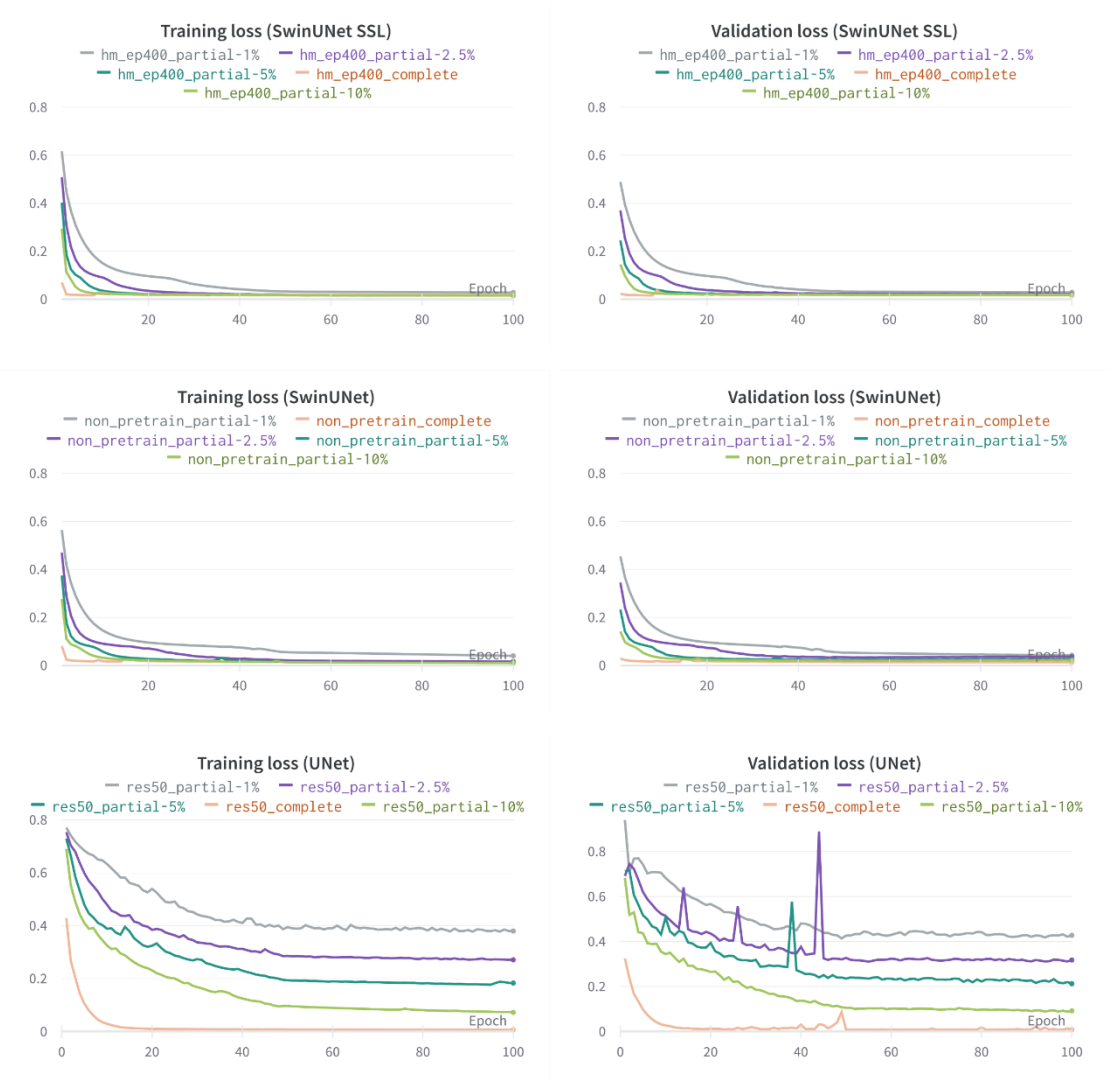

Figure 10: Loss (finetuning)

For SwinUNet SSL, the weights trained for 400 epochs during the pretraining is loaded into the backbone. The SwinUNet-based models consistently exhibit similar performances in terms of both training loss and validation loss. During the initial epochs of training, losses tend to decrease rapidly. As training progresses, the loss values reach a plateau or show only a gradual decrease. The size of the finetuning dataset also has an impact on the starting point and the rate of decrease in the loss values. When a larger finetuning dataset is used, the model starts with a lower initial loss and the drop speed of the loss is generally faster.

The UNet model shows similar behavior in terms of the starting point and rate of decrease in the loss as the SwinUNet-based models. However, one notable difference is that the final loss values for different finetuning datasets are not at the same level. In general, larger datasets tend to lead to lower final loss values. Another observation is the presence of abnormal peaks in the plot of validation loss. But the loss usually recovers quickly and continues to decrease as the training progresses

4.2. Semantic segmentation of map sheets

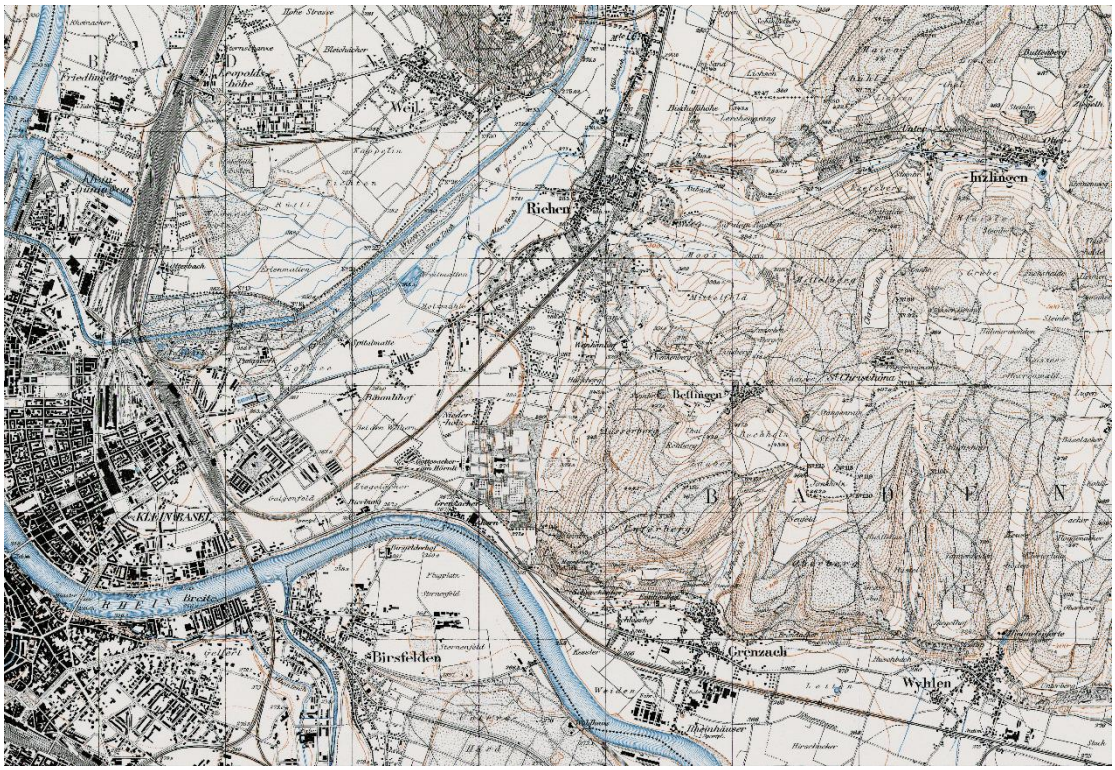

Figure 11: Sample map sheet

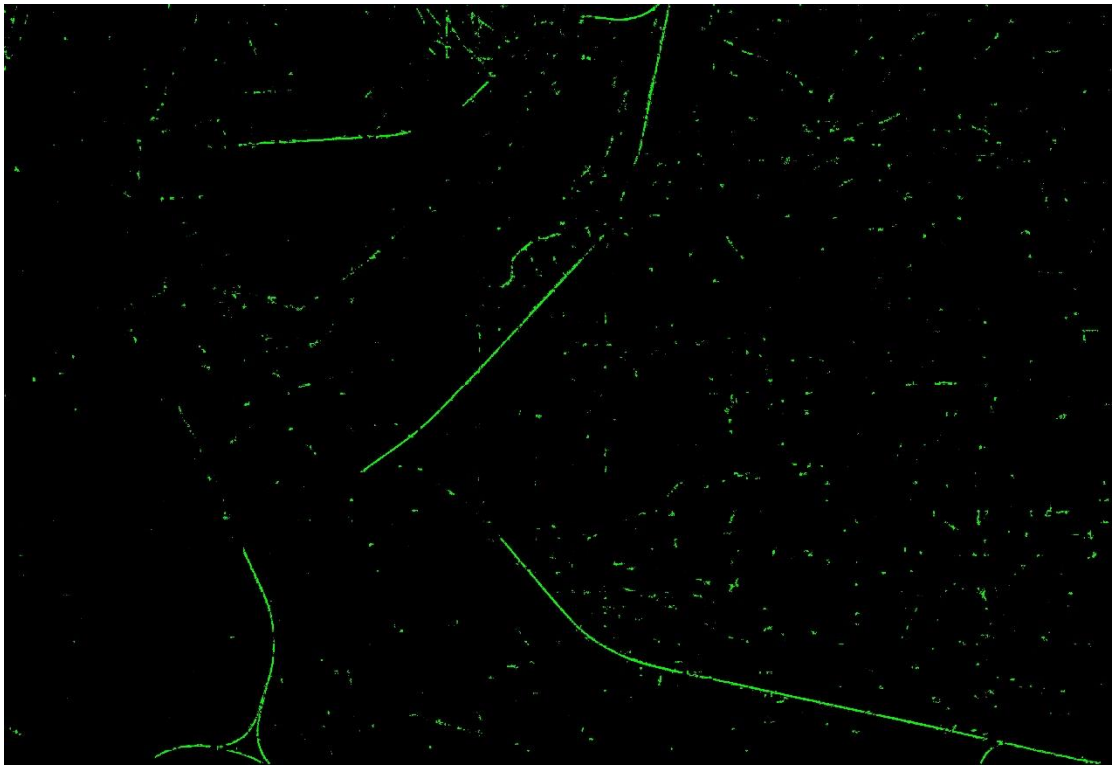

Figure 12: Semantic segmentation of the sample map sheet (SwinUNet SSL)

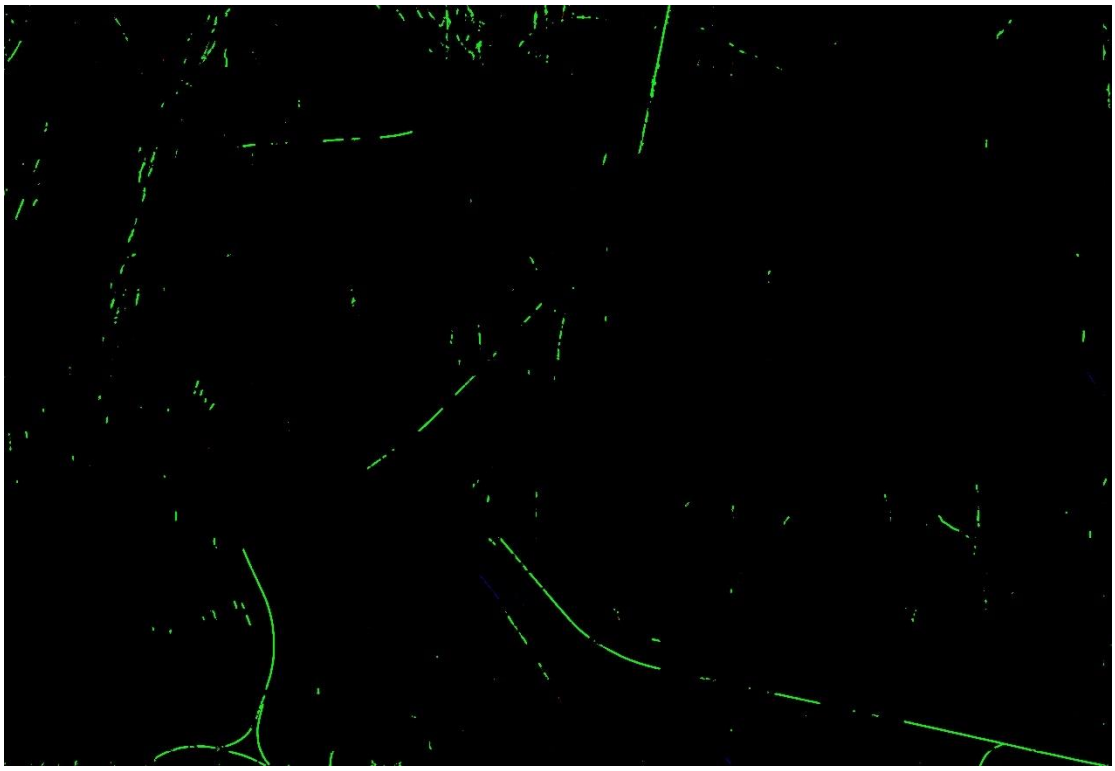

Figure 13: Semantic segmentation of the sample map sheet (UNet)

Figure 12 and 13 show the semantic segmentation results obtained by applying the SwinUNet SSL and UNet models to the map sheet in Figure 11. Just like the labels generated before, the color channels in the segmented images correspond to specific railway categories: the red channel represents narrow railways, the green channel represents normal railways, and the blue channel represents tunnel railways.

The results obtained by the SwinUNet SSL model demonstrate its capability to capture a greater number of railway features within the designated area as well as to maintain the continuity property of railway networks. However, it is worth noting that the resulting features exhibit a lack of smoothness and contain a significant amount of noise, particularly in mountainous regions where contour lines are prevalent.

In contrast, the results produced by the UNet model depict fewer railway features as well as less noise. Nevertheless, the identified features demonstrate a smoother appearance, presenting a cleaner representation of the railway network though more fragmented.

4.3. Semantic segmentation of map tiles

For semantic segmentation of map tiles, SwinUNet SSL, SwinUNet (SwinUNet SLL without pretrain), and the UNet model trained on different finetuning datasets are applied to the test sets. The semantic segmentation or normal railways are visualized in the following sections.

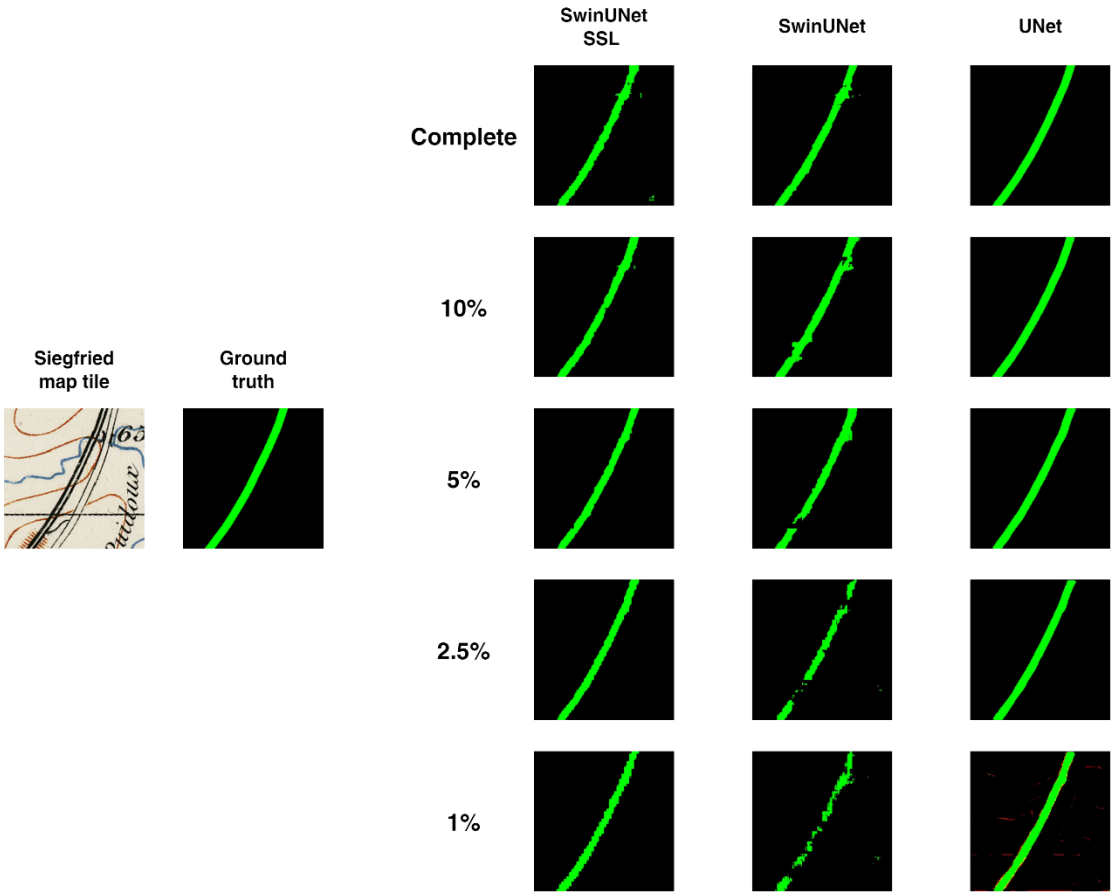

Figure 14: Semantic segmentation of the normal map tile

For a common map tile, both SwinUNet SSL and UNet model generate results that visually resemble the ground truth regardless of the size of the dataset they use during the finetuning (some misclassification occurs in the 1% set for the UNet model). For SwinUNet, the performance quickly diminishes as the size of the finetuning dataset becomes smaller. The problem of rough features still exists in SwinUNet-based models, while for the UNet, the results are generally nice and clean.

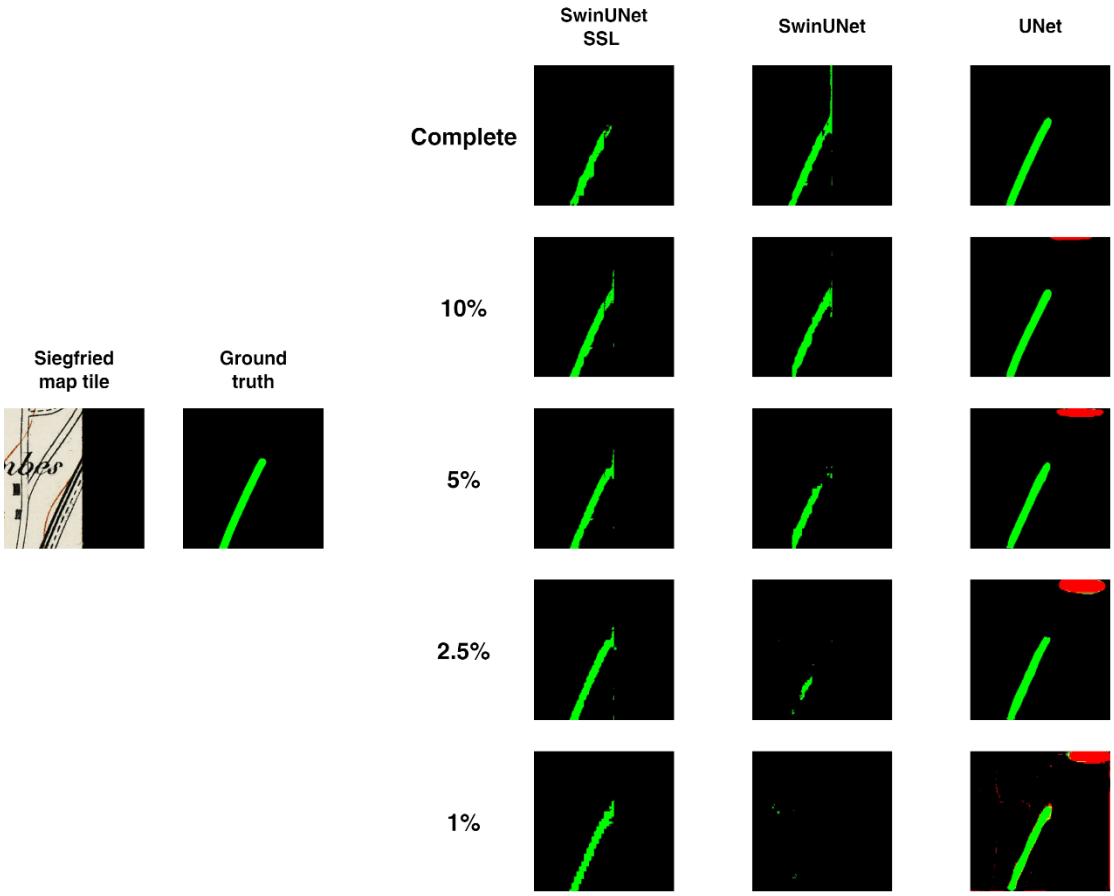

Figure 15: Semantic segmentation of the boundary map tile

For a selected map tile that is close to the boundary, the situation is different. SwinUNet SSL continues to perform well in all five scenarios, though features remain rough. For UNet, the railway pixels are well predicted, however, some uncommon misclassifications occur for pixels outside the boundary when using small finetuning datasets. For SwinUNet, only starting from the 5% partial dataset that the model begins to perform meaningful semantic segmentation. Yet, there exist some kind of boundary effects where it's still possible to identify the rough location of the boundary of the features in the input map tile.

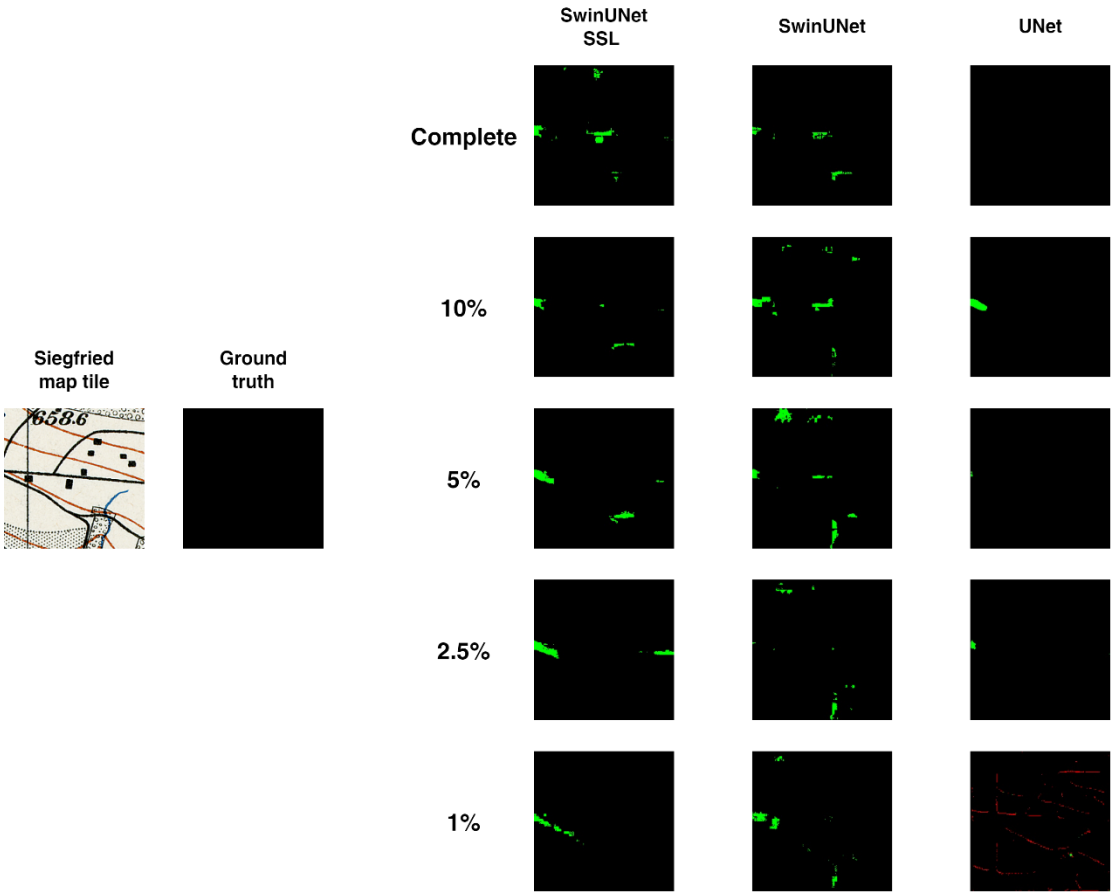

Figure 16: Semantic segmentation of the road map tile

For a selected map tile that contains road features, SwinUNet-based models only show an average level of performance. The misclassification occurs especially in areas where contour lines are close to the road networks. Interestingly, increasing the size of the finetuning dataset does not yield significant improvements in performance. For UNet, while misclassification still occurs when only small finetuning datasets are used, the general performance is brilliant.

4.4. Quantitative evaluation

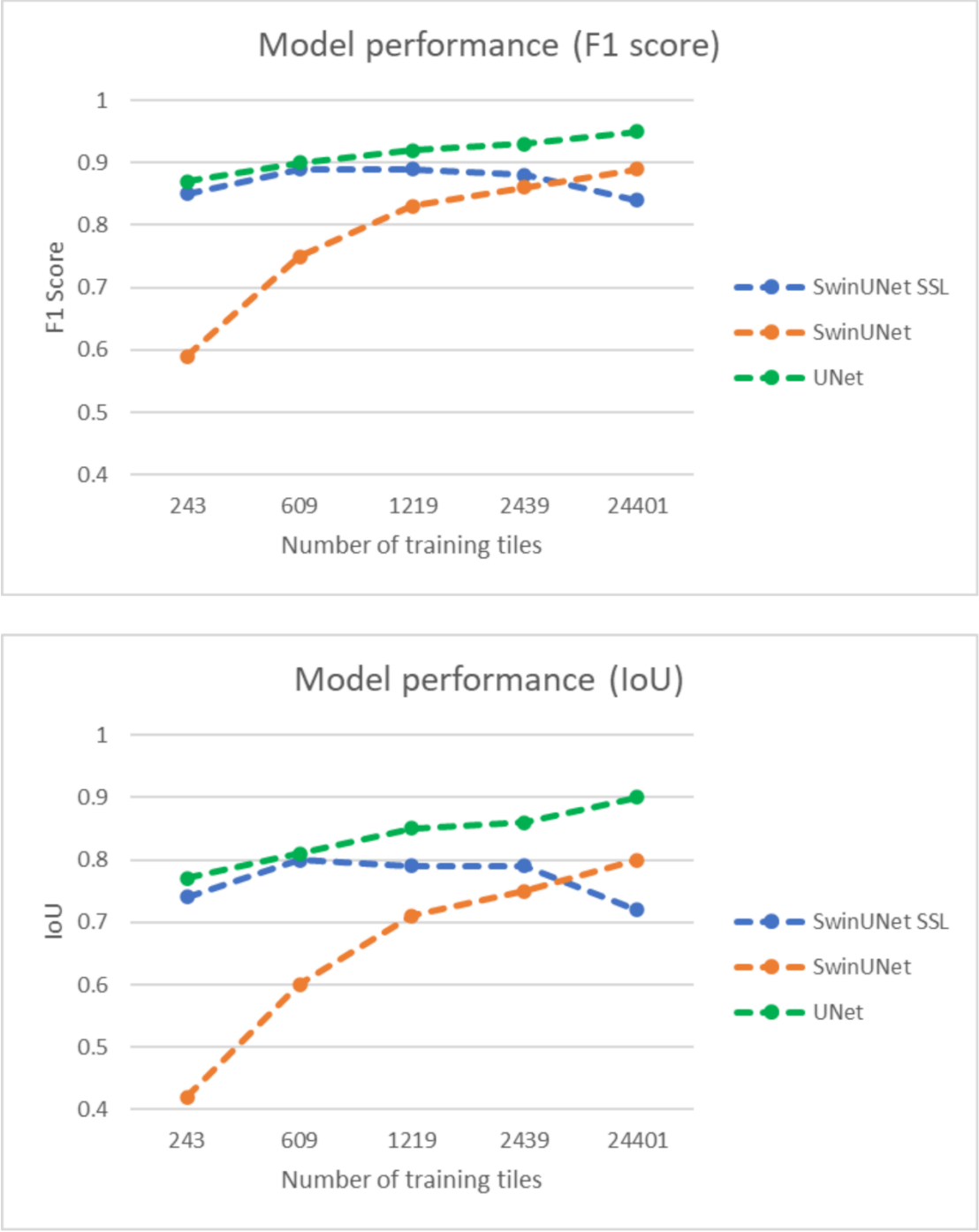

Figure 17: Quantitative evaluation of the model performance

In general, the UNet model outperforms the SwinUNet-based models, as evidenced by higher F1 scores and Intersection over Union (IoU) values across all datasets. All three models still follow the general rule that the model performance tends to improve with the availability of more training data, with the exception of the SwinUNet SSL model trained with the complete dataset. The model performances for other railway classes are not evaluated since the UNet trained with the complete dataset is the only model that is capable to identify some pixels from those classes.

5. Discussion

5.1. Interpretation of results

a) Loss change

The pretraining part behaves like most deep learning models where the loss drops quickly in the beginning and slows down over time. The decreasing trend indicates that continue training might be beneficial as long as the validation loss is not increasing. For the finetuning part, the convergence of losses for SwinUNet-based models shows the signal of stable performance.

b) Semantic segmentation of map sheets

While the SwinUNet SSL model may introduce noise in the final result of semantic segmentation, it exhibits the ability to capture and represent continuous railway features within a given map sheet. This characteristic allows for a quick understanding of the location and extent of railway networks. On the other hand, the UNet model produces segmentation results that are visually clean and nearly free of noise. However, it may generate discontinuous railway features, which can present challenges when interpreting the location of railways.

Depending on the use case of the segmentation results, one model could be favored over the other. If the primary goal is to quickly identify the general areas where railways are present, the SwinUNet SSL model may be preferred due to its ability to provide a comprehensive overview of railway networks. Conversely, if the localization of railways is crucial, then the UNet model's cleaner segmentation results may be more suitable.

c) Semantic segmentation of map tiles

Overall, SwinUNet SSL has a decent performance for railway segmentation though not able to outperform UNet. By comparing with the SwinUNet, it is clear that the pretraining weights have played a vital role in helping model learn feature representations in a quick and accurate manner. SwinUNet SSL significantly outperforms the non-pretrained version when both of them are finetuned with the same dataset. The major issue with the SwinUNet SSL remains to be misclassification for imbalanced class as well as rough segmentation results.

d) Quantitative evaluation

Although three metrics are calculated, accuracy alone may not be a reliable indicator in this case. The presence of railway features occupies only a small proportion of pixels in a testing tile, leading to a significant class imbalance between railway and non-railway pixels. Consequently, accuracy can yield high scores even if the model performs poorly on the minority railway class. In contrast, F1 score and Intersection

over Union (IoU) provide more comprehensive evaluations of the model's performance. Based on the current results, UNet remains the preferred choice for semantic segmentation tasks involving railways, as indicated by higher F1 scores and IoU values. However, when the training dataset is small, SwinUNet can achieve comparable performance to UNet. It is important to note that SwinUNet exhibits a counterintuitive behavior as the training dataset size increases. Figure 10 illustrates a small leap in the training and validation loss, suggesting a departure from the local minimum and convergence towards a sub-optimal minimum. The underlying cause for this behavior is still unclear and warrants further investigation.

5.2. Limitations

Limitations in this project can be described from three perspectives:

- Data
- Model
- Evaluation

5.2.1. Data perspective

a) Data

Dissimilarity of features: While the closest possible time frames for Siegfried Map and Old National Map series have been chosen to minimize the dissimilarity of features, the issue is still unavoidable. The dissimilarity may originate from: 1) Development: new buildings or infrastructures can be constructed, boundaries of the forests might change, and the direction of the roads can be modified. 2) Map producing techniques: symbols representing certain features or name tags of a place can be put at totally different places. The dissimilarity in the positive pair during contrastive learning can undermine the model's ability to learn useful feature representations. One possible improvement is to also introduce data augmentation so that stronger positive pairs can be acquired.

Imbalanced class: The majority of railways belong to the normal railway class which naturally leads to a scarcity of samples from narrow or tunnel railways. It is technically hard for a deep learning model, especially a Transformer based model to learn meaningful features for downstream tasks if not enough data is available. Many generative models have already shown promising results in generating new data. Introducing them to this project can potentially solve the imbalanced class issue. Also, switching the loss function to the focal loss or other losses that can better tackle the imbalanced class can be another solution.

b) Data preprocessing

Due to the technical difficulty, it's hard to generate a road tile while also ensuring no railways are included in the same tile. Since the tile size is small, the assumption here is that each road tile contains no railway features. The labels for road tiles are therefore generated by directly creating a black mask. The wrong information provided by the

label might confuse the model during the finetuning. This issue can be relieved by adding additional checks during the preprocessing.

5.2.2. Model perspective

a) Batch size

Batch size is a crucial factor that can affect the performance of contrastive learning as it determines the number of negative examples included during the pretraining for each batch. Due to the limited computing power available for the project, the batch size can only be set to 96 which is much smaller compared to the batch size recommended by the SimCLR framework. Given the current situation, the batch size will continue to pose a significant challenge.

b) Training difficulty

Due to the complexity of the model, it's extremely hard to find out appropriate combination of learning rate, scheduler, and other model parameters. At the same time, identifying specific parameters that lead to unsatisfactory model performance is even more challenging. There is no quick solution to address this issue except investing more time and making more attempts.

5.2.3. Evaluation perspective

Due to the limited amount of time available for the project, the emphasis is given to model building, workflow development, and model testing, leading to a lack of attention to perspectives such as quality control. Measures such as cross-validation can be added to better evaluate the generalization ability of the model. In addition, the random seed has been identified to influence model performance, sometimes to a non-negligible amount. Thus, training with different random seeds and building up error bars for quantitative metrics mentioned in section 4.4 might be helpful to improve model reliability.

6. Conclusion and outlook

6.1. Overall conclusion

In summary, the proposed SwinUNet SSL model in this project has demonstrated several strengths:

+ Good performance with small training dataset: The SwinUNet SSL has exhibited comparable level of performance to mainstream models when data availability is limited.
+ Global perception: Semantic segmentation of map sheets using SwinUNet SSL allows for a quick identification of the location of railway features due to the continuity of features.
+ Efficient training speed: Although not explicitly mentioned previously, the

model has showcased the same level of training speed as mainstream models.

+ Room for improvements: There are various tunable parameters within the model that can be adjusted for further improvement.

However, there are also drawbacks to the model:

- Fair performance with large training dataset: The model's performance diminishes when trained on larger datasets.
- Difficulty in handling imbalanced training classes: The model encounters challenges in learning meaningful features from classes such as tunnel and narrow railways mainly due to scarcity of data.
- Steep learning curve: Due to the model complexity, prior knowledge for CNNs, SSL, and ViT is required. Also, some additional efforts are needed to comprehend the model structure and all available parameters.

6.2. Outlook

There are two possible future directions that the project can go:

a) Perfecting the current model

In Section 5.2, several limitations and challenges have been identified, along with proposed solutions to address them. Moving forward, one promising direction is to focus on implementing these proposed solutions and resolving the current issues within the project.

b) Exploring alternative pretraining strategies

While SimCLR is a simple and straight forward self-supervised learning framework, it requires decent amount of computational power to achieve optimal performance. There are other SSL frameworks such as BEiT or Masked Autoencoders (MAE) that might offer different advantages which are more suitable for Transformer-based backbones while also being computational friendly.

6.3. Achievement of objectives

The workflow for conducting semantic segmentation with self-supervised vision transformers has been successfully established, starting from data preprocessing, model building, pretraining, finetuning, and concluding with model testing. This accomplishment marks the achievement of the project's primary objective. Furthermore, the project has also accomplished its sub-goals, demonstrating that self-supervised pretraining on large unlabeled datasets can effectively enhance the subsequent performance of semantic segmentation on smaller labeled datasets. The introduction of ViT, unfortunately, does not yield significant improvements in the performance of semantic segmentation. Yet, there is still space for improvement as mentioned in section 5.2, and this does not imply the end of utilizing ViT in the field of historical maps. On the contrary, it highlights the need for further exploration and investigation to optimize

the use of ViT for semantic segmentation tasks in this domain.

Reference

Cao, H., Wang, Y., Chen, J., Jiang, D., Zhang, X., Tian, Q., & Wang, M. (2021). *Swin-Unet: Unet-like Pure Transformer for Medical Image Segmentation*. http://arxiv.org/abs/2105.05537

Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., & Joulin, A. (2020). *Unsupervised Learning of Visual Features by Contrasting Cluster Assignments*. http://arxiv.org/abs/2006.09882

Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020). *A Simple Framework for Contrastive Learning of Visual Representations*. http://arxiv.org/abs/2002.05709

Chen, T., Kornblith, S., Swersky, K., Norouzi, M., & Hinton, G. (2020). *Big Self-Supervised Models are Strong Semi-Supervised Learners*. http://arxiv.org/abs/2006.10029

Chen, X., Fan, H., Girshick, R., & He, K. (2020). *Improved Baselines with Momentum Contrastive Learning*. http://arxiv.org/abs/2003.04297

Diakogiannis, F. I., Waldner, F., Caccetta, P., & Wu, C. (2019). *ResUNet-a: a deep learning framework for semantic segmentation of remotely sensed data*. https://doi.org/10.1016/j.isprsjprs.2020.01.013

Doersch, C., Gupta, A., & Efros, A. A. (2015). Unsupervised Visual Representation Learning by Context Prediction. *CoRR*, *abs/1505.05192*. http://arxiv.org/abs/1505.05192

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2020). *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. http://arxiv.org/abs/2010.11929

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P. H., Buchatskaya, E., Doersch, C., Pires, B. A., Guo, Z. D., Azar, M. G., Piot, B., Kavukcuoglu, K., Munos, R., & Valko, M. (2020). *Bootstrap your own latent: A new approach to self-supervised Learning*. http://arxiv.org/abs/2006.07733

He, K., Fan, H., Wu, Y., Xie, S., & Girshick, R. (2019). *Momentum Contrast for Unsupervised Visual Representation Learning*. http://arxiv.org/abs/1911.05722

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., & Guo, B. (2021). *Swin Transformer: Hierarchical Vision Transformer using Shifted Windows*. http://arxiv.org/abs/2103.14030

Liu Z, Lin Y, Cao Y, Hu H, Wei Y, Zhang Z, Lin S, & Guo B. (2023). *Swin-T pretrained models*. https://github.com/microsoft/Swin-Transformer

Long, J., Shelhamer, E., & Darrell, T. (2014). *Fully Convolutional Networks for Semantic Segmentation*. http://arxiv.org/abs/1411.4038

Noroozi, M., & Favaro, P. (2016). Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles. *CoRR*, *abs/1603.09246*. http://arxiv.org/abs/1603.09246

PyTorch Foundation. (2023). *PyTorch Tutorial*. https://pytorch.org/tutorials/

Ronneberger, O., Fischer, P., & Brox, T. (2015). *U-Net: Convolutional Networks for Biomedical Image Segmentation*. http://arxiv.org/abs/1505.04597

Scheibenreif, L. M., Hanna, J., Mommert, M., & Borth, D. (2022). *Self-supervised*

Vision Transformers for Land-cover Segmentation and Classification [Proceeding].

Swisstopo. (2023a). *Background information on the National Map*. https://www.swisstopo.admin.ch/en/knowledge-facts/histcoll/historical-maps/national-map.html

Swisstopo. (2023b). *Background information on the Siegfried Map*. https://www.swisstopo.admin.ch/en/knowledge-facts/histcoll/historical-maps/siegfried-map.html

Wu, Z., Xiong, Y., Yu, S., & Lin, D. (2018). *Unsupervised Feature Learning via Non-Parametric Instance-level Discrimination*. http://arxiv.org/abs/1805.01978

Zhang, R., Isola, P., & Efros, A. A. (2016). *Colorful Image Colorization*. http://arxiv.org/abs/1603.08511