Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Building Block Vectorization of Atlas Municipal

Research Topics in Cartography, Course Project

Emanuel Stüdeli, Chunyang Gao

Supervisor:
Yizi Chen

June 26, 2024

1 Introduction

Atlas Municipal is a comprehensive series of city maps of Paris, spanning from 1866 to 1937. These maps were periodically updated to reflect the changing road networks, providing a historical record of urban development. Today, they are valuable for studying the urban development and transformation of Paris, offering insights for various research fields such as social science and historical studies. To facilitate such research, digitizing these maps is essential.

One major part of the digitization process is the vectorization of building blocks. Building blocks are defined as closed-shape objects of one or multiple houses surrounded by streets, rivers, fortification walls, or other boundaries, and are never directly connected to each other. These building blocks represent the most basic building components of the city's layout and are critical for understanding the structure and evolution of urban spaces.

Manual vectorization of building blocks is very time-consuming, automated methods using deep neural networks suggest a more effective approach. However, the detection process faces several significant challenges. In the Atlas Municipal, there is no designated symbology to mark the building blocks. By default the building blocks are shown as empty areas, only some individual buildings are represented with diagonal hatched areas. This makes color- or texture-based detection algorithms ineffective. Additionally, nested objects within the building blocks require a hierarchy of image filtering techniques. Moreover, building blocks are often obscured by overlapping elements such as texts, underground lines, and graticule lines.

This project aims to address these challenges by developing robust deep neural networks for the accurate vectorization of building blocks. To achieve this, the project will focus on finding a feasible pipeline for vectorizing building blocks, investigating three different types of deep neural network models, and evaluating their performance with pixel- and instance-based metrics. Additionally, the project will aim to improve and optimize both the pipeline and the models to enhance the overall effectiveness of the digitization process. By doing so, it will contribute to the effective digitization of historical maps, enabling more detailed and comprehensive studies of Paris's urban evolution.

2 Data

In this project, the inputs consist of a set of JPEG RGB images of scanned map sheets from the Atlas Municipal, similar to the one illustrated in figure 1. The original maps do not necessarily have a rectangular shape, which results in black pixels at the edges of the scans where no information is available. The images can be quite large, often reaching dimensions of 8000x8000 pixels.

Corresponding mask images are provided to identify non-relevant pixels. In these mask images, non-relevant pixels are assigned a value of 0, while relevant pixels have a value of 255, as shown in the illustration below (figure 3).

Expected output for this task is a binary ground truth mask of the building blocks. The labels are stored in PNG format to ensure lossless quality. The ground truth image is an 8-bit single-channel image where the background is represented by a pixel value of 0, and the building block areas are represented by a pixel value of 255, as shown in Figure 5.

Figure 1: Input image

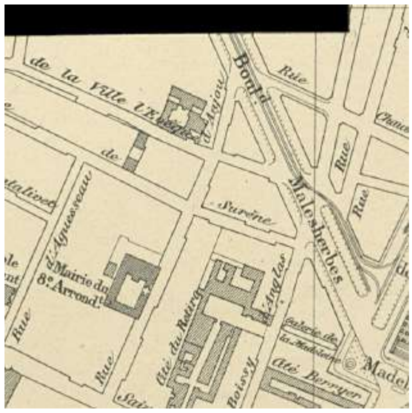

Figure 2: Input image detail

Figure 3: Mask image

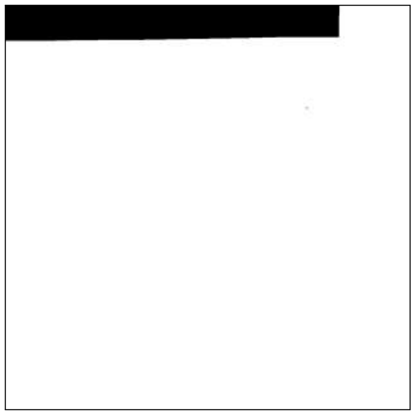

Figure 4: Mask image detail

Figure 5: Ground truth image

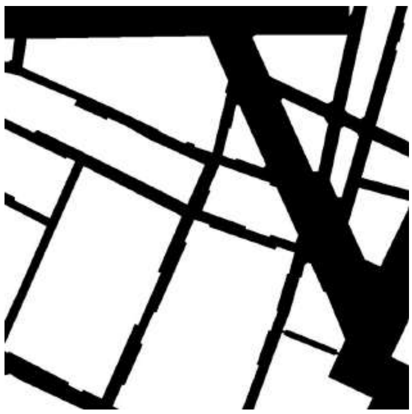

Figure 6: GT image detail

One single input image (file 101) with the corresponding mask and ground truth is used for training, another for validation (file 201). Three map sheets are provided for testing (files 301, 302 and 303). All input images have slightly different background colours and contrasts as a result of the variations in the original map sheet and the scanning conditions.

When taking a closer look at the testing images, it can be observed that file 301 shows the same geographical area as the training image, but from a different map version. On the other hand, file 303 seems to originate from the same map series as the training image, but shows a different geographical extent. Finally, in file 302, both the map version as well as the geographical extent are different.

3 Methods

3.1 Pipeline

The process for this project involves multiple steps to ensure the accurate vectorization of building blocks. The overall workflow can be divided into three main stages: training (including validation), testing and vectorization, as illustrated in figure 7.

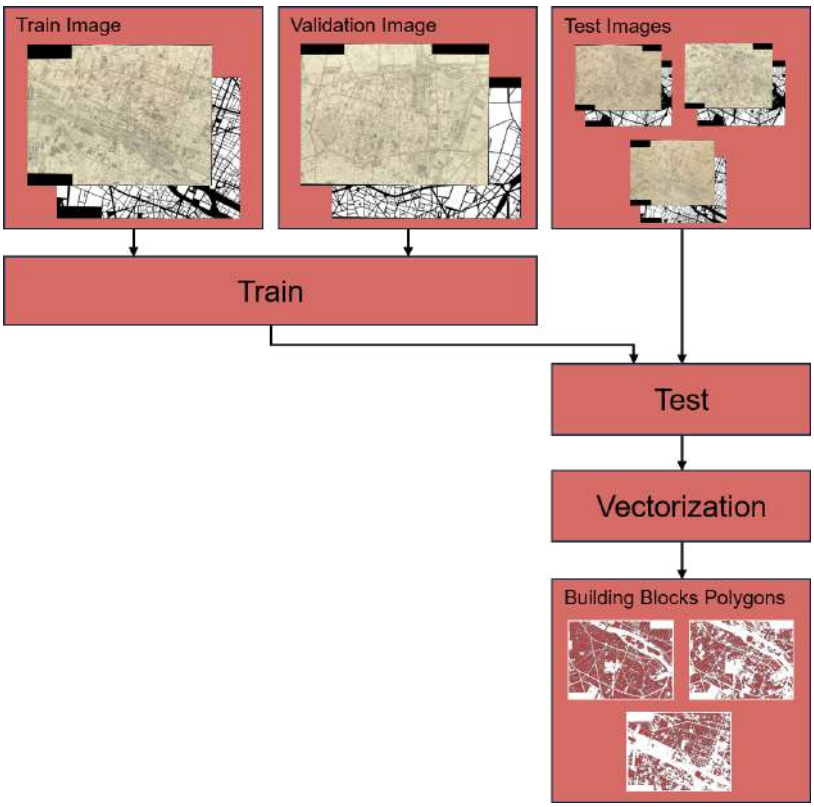

Figure 7: Pipeline for vectorizing building blocks

The training phase consists of two steps, a training step and a validation step. Initially, the input images are divided into smaller tiles to facilitate processing. For the training step, various data augmentation techniques are then applied to address different scan conditions such as skews, warps, and contrast variations. These techniques include random contrast adjustments, random rotations, and random thin plate spline transformations. Following this, three different models, a U-Net, a ResUnet, and a SwinUnet are trained to perform classification task. The training employs the sum of dice loss and binary cross-entropy (BCE) as loss function, while regularization and gradient clipping techniques are used to

reduce overfitting. The model training settings are shown in table 1. After each epoch, a validation step is included to evaluate the performance of the model. As in the training step, the first action in the validation step is to generate individual tiles from the validation image. The tiles are then predicted and and reconstructed into full image. The pixels are assigned to a binary class label based on the sigmoid function with threshold values. Basic post-processing steps, such as opening, closing, and connected components analysis, as well as the mask to mask out irrelevant areas are applied to prepare the predicted image for evaluation. Finally two kinds of evaluation metrics, both pixel-based and instance-based evaluations, are calculated to assess the model performance.

Characteristics	**UNet**	**ResUnet**	**SwinUnet**
Input image size	512	512	224
Data augmentation mode	ctr, rot, tps	ctr, rot, tps	ctr, tps
Batch size	8	16	8
Criterion	dice loss + BCE	dice loss + BCE	dice loss + BCE
Learning rate	$1.0 \cdot 10^{-5}$	$1.0 \cdot 10^{-5}$	$1.0 \cdot 10^{-5}$
Weight decay	$1.0 \cdot 10^{-4}$	$1.0 \cdot 10^{-4}$	$1.0 \cdot 10^{-5}$
Gradient clipping	1.0	1.0	1.0
Pretrained weights	no	no	yes

Table 1: Training settings for different network types. Abbreviations: ctr: random contrast, rot: random rotation, tps: random thin plate spline, BCE: binary cross entropy loss

The test process involves three test images with varying background shades, contrast, and other scan properties to evaluate the models' performance. The test phase encompasses several steps that are equivalent to the validation step: generating tiles, prediction, image reconstruction, basic post-processing, masking, and calculating evaluation metrics. Here as well, both pixel-based and instance-based evaluation metrics are used to assess the models's performance. For the test phase, the epoch checkpoint of the trained model with the best validation scores is used and the test images are predicted based on the trained weights of this specific epoch.

The last phase, the vectorization phase, consists of two parts. First, the predicted building blocs of the binary pixel image are vectorized. Second, the resulting vector polygons are generalized with the Douglas-Peucker algorithm. Thus the pipeline from the RGB input image of the map sheets to the vectorized building block polygons is complete.

3.2 U-Net

The U-Net proposed by Ronneberger et al. 2015 is used as our baseline model. Its architecture is shown in Figure 8. It is a highly popular deep learning architecture specifically designed for semantic segmentation tasks. Initially developed for medical imaging, it achieved significant success in this field. The unique encoder-decoder structure of U-Net, along with its skip connections, allows it to not only extract important features from images but also retain their spatial information. Unlike other convolutional neural networks (CNNs) used for classification and autoencoders used for image compression, U-Net excels at precise segmentation even with limited training data.

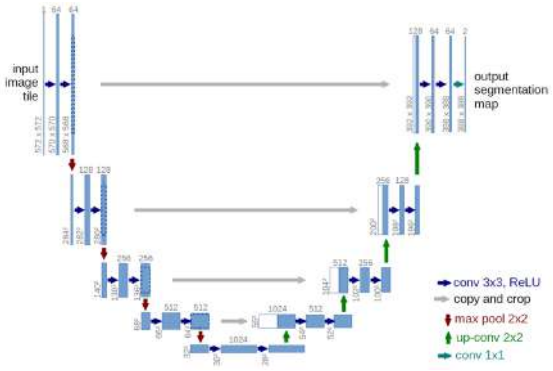

Figure 8: UNet architecture Ronneberger et al. 2015

Our implementation U-Net

Our U-Net model is based on the architecture of milesial 2017, a customized implementation of the U-Net model which we adapted to our problem. Among other adaptions, we use 5x5 convolutional blocks instead of 3x3 because a larger kernel size can capture larger spatial structures, thereby improving model performance.

3.3 ResUnet

The work of Zhang et al. 2018 serves as reference for the deep residual U-Net (ResUnet) model. The authors introduce a neural network which combines the strengths of deep residual learning and U-Net to extract roads from remote sensing images. The architecture of the proposed model is based on U-Net, an u-shaped encoder-decoder structure with its characteristic skip-connections between the encoding and decoding path. In addition, residual units are used instead of plain neural units as basic blocks to build the ResUnet model. This residual units consist of two convolutional layers as well as the identity mapping, a skip connection within this residual block where the input is added to the output of the double-convolution (figure 9). This design is meant to facilitate training and address the degradation problem. In their work, the authors use a seven-level architecture of ResUnet with a bridge between the encoding and decoding path, as shown in figure 10. Instead of using a pooling operation in the encoding part, a stride of 2 is applied to the first convolution of the residual block to downsample the feature map size by half. The resulting ResUnet network was reported to achieve comparable or even better performance on semantic segmentation problems with much fewer training parameters.

Our implementation of ResUnet

In our project, we implemented the proposed ResUNet architecture by Zhang et al. 2018, adapting it to our specific problem. Our findings indicate that factors such as the kernel size and the number of channels in the initial convolutional layer significantly impact model performance. The optimal model architecture for our application consists of a seven-level network, with each level comprising two 5x5 convolutional blocks and 32 channels in the input convolutional layer. Leaky ReLU is utilized as the activation function. The detailed architecture is illustrated in Figure 11.

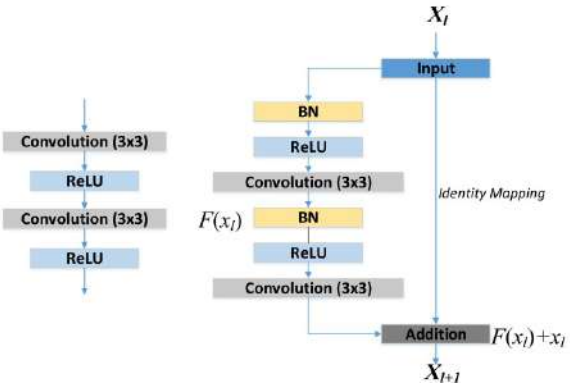

Figure 9: Residual block (Zhang et al. 2018)

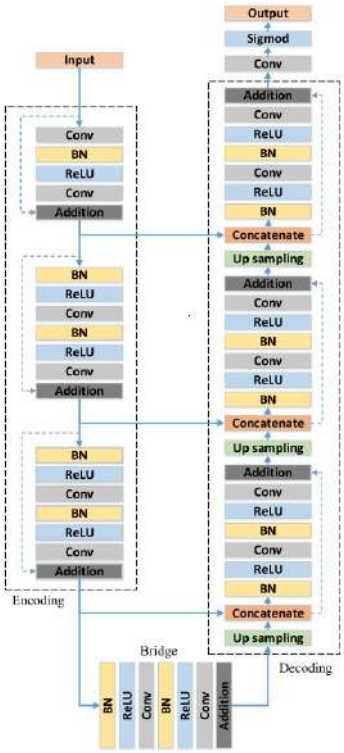

Figure 10: ResUnet architecture (Zhang et al. 2018)

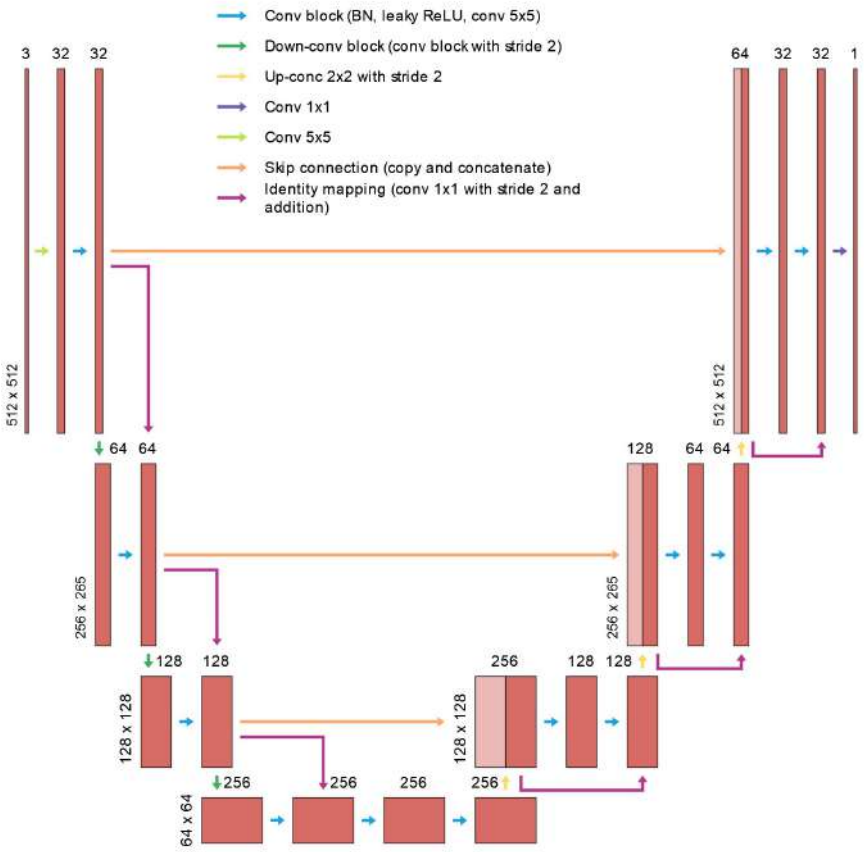

Figure 11: ResUnet architecture adapted to our problem

3.4 SwinUnet

SwinUnet is a U-shaped encoder-decoder network with a bottleneck in between as well as skip connections. The encoder, bottleneck, and decoder are designed based on the Swin Transformer block. The input images are partitioned into non-overlapping patches. Then feature embedding layer on each patch is applied and fed into the encoder. The encoder consists of the transformer-based Swin Transformer blocks and patch merging layers. The patch merging layers are responsible for downsampling, while the Swin Transformer blocks are responsible for learning hierarchical feature representations from the patches. The decoder is composed of the Swin Transformer block and patch-expanding layer. A patch-expanding layer is responsible for up-sampling. The extracted features are fused with multiscale features from the encoder via skip connections to preserve the feature map's resolution. The architecture is presented in figure 12 (Cao et al. 2022).

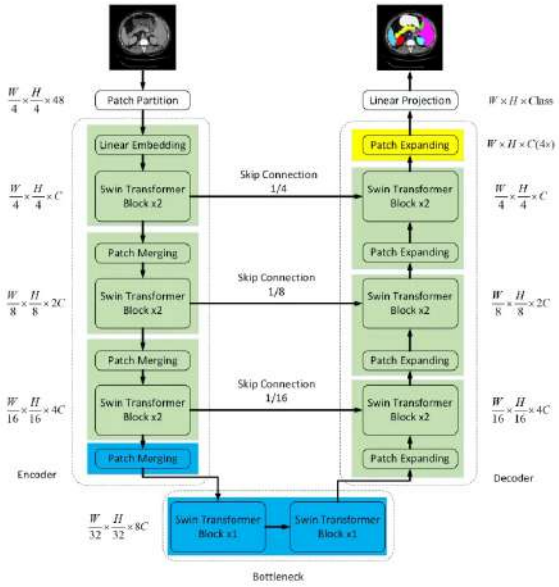

Figure 12: SwinUnet architecture Cao et al. 2022

Compared to traditional U-Net structure, SwinUnet replaces convolutional layers with Swin Transformer blocks. It can capture both local and global semantic features. Regarding its application in historical map segmentation field, Xia et al. 2023 uses SwinUnet as the foundation for both pretraining and downstream railway detection task. The results demonstrate that the method achieves comparable performance to fully supervised approaches.

Our implementation of SwinUnet

In our project, we implemented the proposed SwinUnet architecture by Cao et al. 2022 including the pre-trained weights, which are initialized with ImageNet pre-trained model parameters. In our case, we generate tiles of size 224 × 224 as input image, the same input image size as the original structure. As for the output, the model output class is set to 1. The other parts are the same as the original architecture.

4 Results

The training scores of the three models are shown in figure 13. As the curve of the validation loss is not meaningful, the selection of the best epoch is based on the combination of the F1-score and the panoptic quality of the validation image. The model weights of the following epochs are used to predict the building blocks of the three test images: Epoch 14 for U-Net, Epoch 14 for ResUnet, Epoch 18 for SwinUnet.

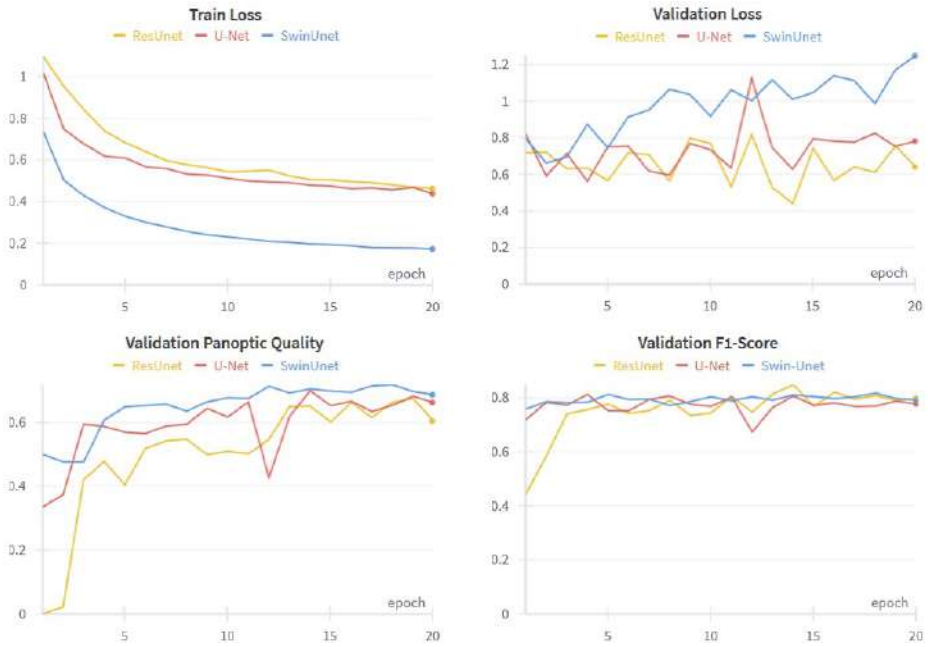

Figure 13: Training scores for each model

The prediction output of the building blocks for each model and test file can be found in the appendix. The same applies to the final results, i.e. the vectorized and generalized building block polygons for each prediction. In the following figures, the results of the intermediate steps as well as the final result are visualised using the example of the SwinUnet model for test file 303, as this combination achieves the best performance (see chapter 5 Evaluation). The result of the segmentation is a binary raster file with the value 255 where a building block is predicted and 0 otherwise (Figure 18). By vectorizing, we obtain a Shapefile with the individual building block polygons, whereby the pixel structure is initially still visible (Figure 20). The final result is then the Shapefile with the generalized polygons (Figure 22).

The results show that the pipeline is effective in predicting and vectorizing building blocks of the Atlas Municipal. Based on the visual assessment, we can observe that the prediction results are generally very good. The greatest inaccuracies tend to occur with parks and squares, which often have very diverse and heterogeneous representations on the map, making it sometimes difficult even for human beings to clearly identify an object. Not all prediction results are as accurate as those of SwinUnet for file 303. Some common problems with other models and/or other test files are interrupted/missing roads (see Figure 24) and black patches instead of building blocks (see Figure 25).

Figure 14: File 303 Input Image

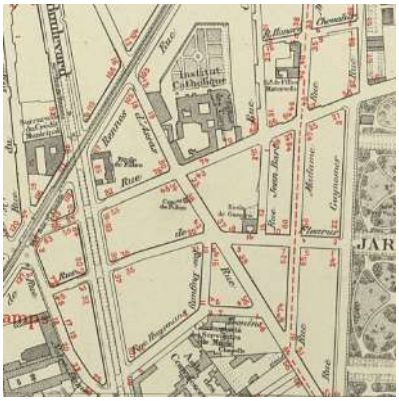

Figure 15: File 303 Input detail

Figure 16: File 303 Ground Truth

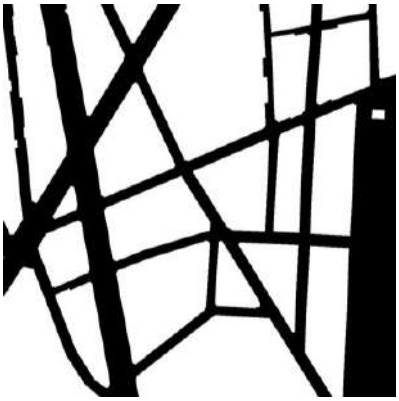

Figure 17: File 303 GT Detail

Figure 18: File 303 Prediction SwinUnet

Figure 19: File 303 Prediction SwinUnet Detail

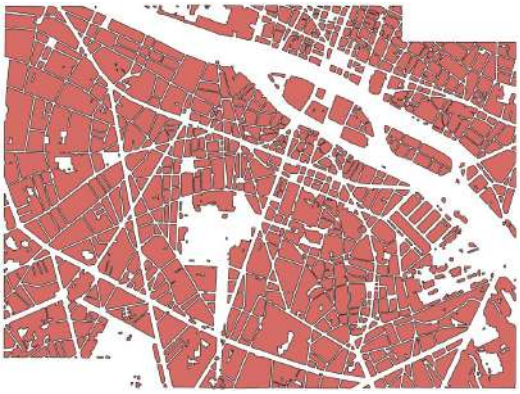

Figure 20: File 303 Vectorized Building Block Polygons

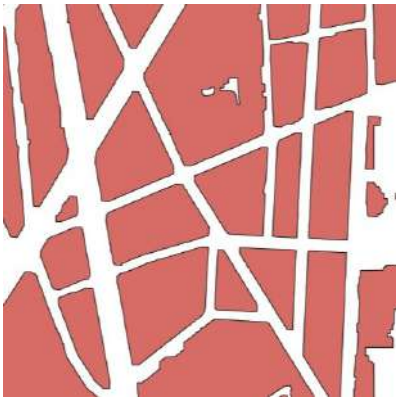

Figure 21: File 303 Vectorized Building Block Polygons Detail

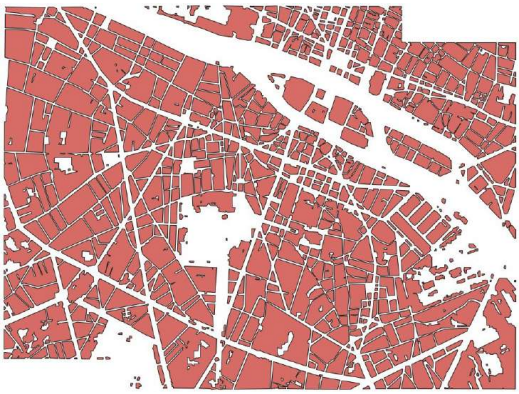

Figure 22: File 303 Generalized Building Block Polygons

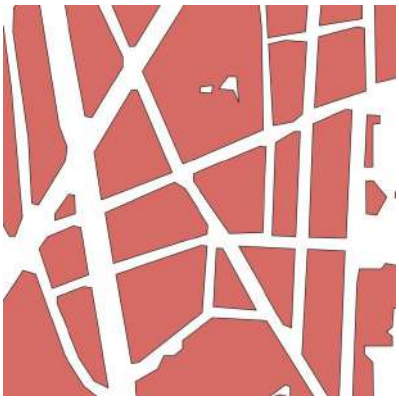

Figure 23: File 303 Generalized Building Block Polygons Detail

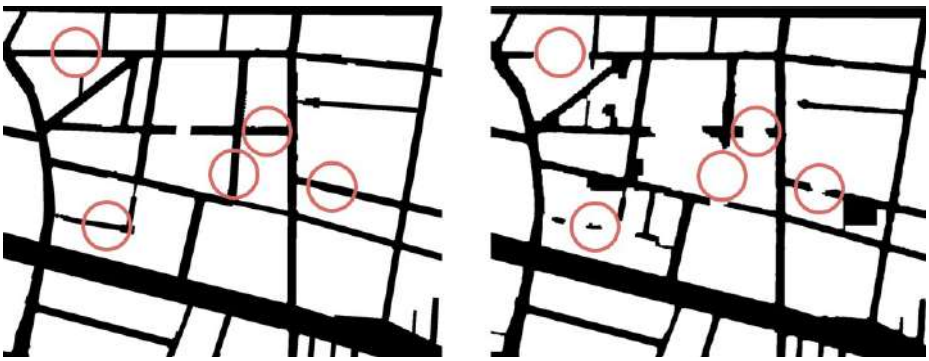

Figure 24: Interrupted and missing roads (red circles) on excerpt of test file 301. Left: ground truth. Right: prediction results of SwinUnet.

Figure 25: Missing building blocks (red circles) on excerpt of test file 301. Left: ground truth. Right: prediction results of U-Net.

5 Evaluation

One very common metric for evaluating binary classification models is the F1 score, which combines the metrics precision and recall into a single value and thus enables a more complete assessment of the model. However, not only the individual pixel is relevant for the evaluation of an image segmentation task, but the entire detected object, which leads us to the instance-based evaluation metrics. These instance based metrics evaluate how closely segments are matched with their ground truths and how effective the model is in getting the segment predictions right. For this reason, both pixel-based and instance-based evaluation metrics are used to asses the model performance.

5.1 Pixel-based evaluation metrics

The pixel-based evaluation of the models was conducted using the macro F1 score. It can be interpreted as a harmonic mean of precision and recall equally averaged over all classes. 0 indicates the worst and 1 the best score.

The specific F1 scores for each model and file are illustrated figure 26. With values ranging from 0.80 to 0.95, all three models generally reach a high performance. It can be

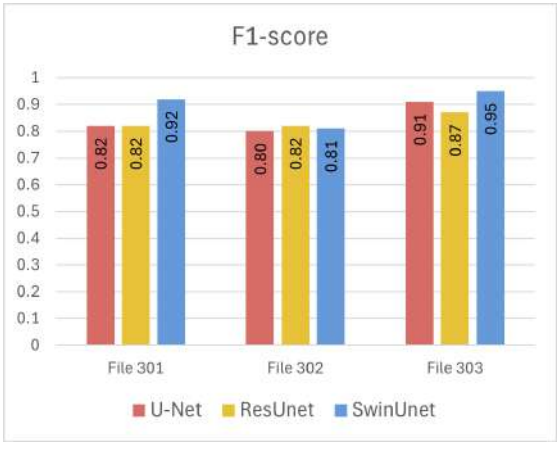

Figure 26: F1 macro score

observed that all models demonstrated their highest performance on file 303. Specifically, SwinUnet achieved the best F1 score of 0.95 on file 303, compared to U-Net's 0.91 and ResUnet's 0.87. On file 301, SwinUnet also performed the best with an F1 score of 0.92, while U-Net and ResUnet scored both 0.82. Only for file 302, ResUnet outperformed the other models with an F1-score of 0.82, the other models follow very close with scores of 0.81 for SwinUnet and 0.80 for U-Net.

5.2 Instance-based evaluation metrics

We utilize three metrics for the instance-based evaluation of our model: recognition quality, segmentation quality, and panoptic quality, which integrates the first two metrics.

The process to calculate these metrics begins with the segment matching. A bipartite graph matching is conducted between the predictions and the ground truth, ensuring that each predicted instance is paired with at most one ground truth instance.

The underlying principle for all three metrics is the Intersection over Union (IoU) (Mechea 2018). IoU is defined as the area of intersection divided by the area of union of the predicted segments and their corresponding ground truth segments.

$$IoU = \frac{\text{Prediction } \cap \text{ GroundTruth}}{\text{Prediction } \cup \text{ GroundTruth}}$$

From IoU, two critical aspects can be derived:

- **Recognition quality**: By selecting a threshold value, IoU can be used to distinguish true positives, false positives, and false negatives. This distinction allows the calculation of segment-based precision and recall, which are subsequently used to determine the recognition quality. The recognition quality lies between 0 and 1, the higher the better.
- **Segmentation quality**: IoU also measures how closely the predicted segments match their ground truths. This assessment, applied to each true positive, determines the segmentation accuracy. A value of 1 would mean that all segments are perfectly matched, 0 is the worst possible score.

The **panoptic quality** (PQ) is then calcualted as a combination of both recognition quality and segmentation quality into a single comprehensive metric. The panoptic quality reaches its best value at 1 and worst score at 0. This composite metric provides the most complete assessment of our model's performance in segmentation tasks, capturing both the ability to correctly identify and accurately segment instances.

$$\text{PQ} = \underbrace{\frac{\sum_{(p,g)\in TP} \text{IoU}(p,g)}{|TP|}}_{\text{segmentation quality (SQ)}} \times \underbrace{\frac{|TP|}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}}_{\text{recognition quality (RQ)}}$$

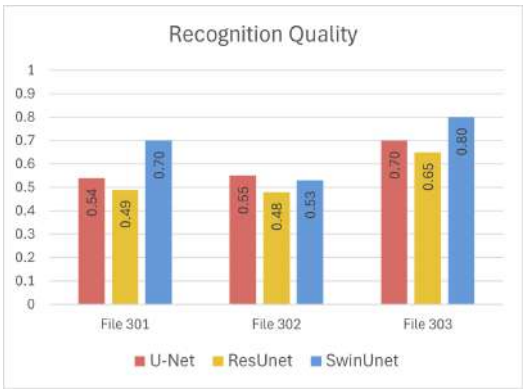

Figure 27: Recognition Quality

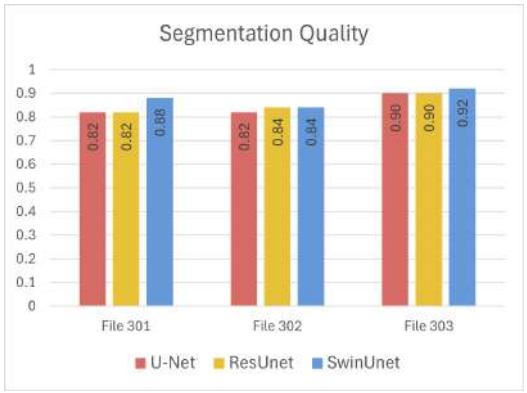

Figure 28: Segmentation Quality

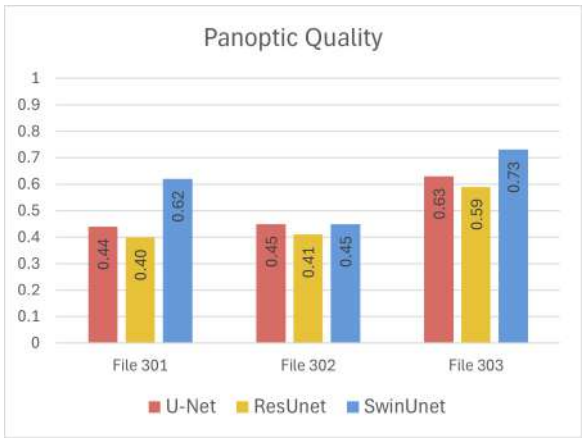

Figure 29: Panoptic quality

The results of the instance-based evaluation metrics for each model and file are shown in the bar charts of figures 27, 28 and 29. All models show a high segmentation quality (between 0.82 and 0.92), while the recognition quality especially for files 301 and 302 is lower (between 0.48 and 0.80). This results in an overall panoptic quality of around 0.40 to 0.45 for files 301 and 302 and 0.59 to 0.63 for file 303. The only exception is SwinUnet which achieves a significantly higher panoptic quality score for file 301 (0.62) and file 303 (0.73) than the other two models. These results demonstrates that SwinUnet generally achieves the highest instance-based scores across different files and on the other hand all models perform best on file 303.

6 Discussion

When looking at the results of the evaluations, two points stand out. First, all models perform remarkably better on file 303 than on the two other files. Second, SwinUnet almost always has the highest performance across differnt files compared to the other two models, thus indicating its robustness and accuracy in detecting building blocks.

As stated in the data section, file 303 has the most similar background shade and contrast conditions to the training file, because it originates from the same map series while the other two test images come from a different map version. This most likely explains why the models perform better on file 303 than on the other two files. From this, we can conclude that the models are much more capable of generalizing with respect to the geographical extent than different map styles, background shades and contrast conditions, even if the same map extent was used for training. This generalization problem has been addressed with various data augmentation techniques that have already significantly increased performance and helped to achieve satisfying results. However, the evaluation metrics show that there is still potential in this filed. One possible way to overcome the generalization problem would be to use training input data from different map series or to use more elaborate data augmentation techniques.

The results and evaluation metrics demonstrate that SwinUNet outperforms the other two models and exhibits better generalization capabilities. However, it is important to note that SwinUnet is the only model among the three where a version with pretrained weights has been implemented for the classification task. This makes it challenging to attribute the superior performance solely to the model's architecture, the pretrained weights, or a combination of both factors. To ensure a fair comparison, it would be necessary to implement pretrained weights for both the U-Net and ResUNet models as well.

7 Conclusion and Outlook

Our pipeline has demonstrated feasibility for the vectorization of building blocks of the Atlas Municipal. We implemented three different model types, a U-Net, a ResUnet and a SwinUnet model and investigated their performance in detecting building blocks using pixel- and instance-based evaluation metrics. After optimizing the models and their parameters for our specific problem, all three showed good results, proving their suitability for the segemntation of building blocks. Among the three models, SwinUnet outperformed ResUnet and U-Net across both pixel-based and instance-based metrics.

Given the limited availability of input data, data augmentation has proven effective in enhancing model performance and improving generalization capabilities. For both U-Net and ResUNet, increasing the kernel size led to marginal performance improvements. Using pretrained weights was investigated based on the SwinUnet model and has further improved the model performance.

To further enhance the generalization capabilities of the models, it is crucial to integrate additional data sources. This includes using training data from different map series or incorporating more elaborate data augmentation methods.

Given the superior performance of SwinUnet, which benefited from pretrained weights, future work should include applying pretrained weights to U-Net and ResUnet models. This could potentially enhance their performance and provide a fair comparison among the models.

Beyond building block vectorization, the pipeline and models developed in this project can be adapted for other segmentation tasks relevant to the Atlas Municipal. This includes tasks like road network extraction, text detection, and segmentation of other urban features.

8 Code, Data and License

All project files, code and data (including input data, final results and model weights) are available for internal use at ETH Zurich under https://polybox.ethz.ch/index.php/s/YXjprMnyTUKXja7. The software is distributed under the MIT License.

MIT License

9 Reference

Cao, Hu et al. (2022). "Swin-unet: Unet-like pure transformer for medical image segmentation". In: *European conference on computer vision*. Springer, pp. 205–218.

Mechea, Daniel (Feb. 2018). "Panoptic Segmentation — The Panoptic Quality Metric". In: *Medium*. URL: https://medium.com/@danielmechea/panoptic-segmentation-the-panoptic-quality-metric-d69a6c3ace30.

milesial (2017). *U-Net: Semantic segmentation with PyTorch Pytorch-UNet*. https://github.com/milesial/Pytorch-UNet?tab=readme-ov-file.

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox (2015). "U-net: Convolutional networks for biomedical image segmentation". In: *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*. Springer International Publishing, pp. 234–241.

Xia, Xue, Chenjing Jiao, and Lorenz Hurni (Nov. 2023). "Contrastive Pretraining for Railway Detection: Unveiling Historical Maps with Transformers". In: pp. 30–33. DOI: 10.1145/3615886.3627738.

Zhang, Zhengxin, Qingjie Liu, and Yunhong Wang (2018). "Road extraction by deep residual u-net". In: *IEEE Geoscience and Remote Sensing Letters* 15.5, pp. 749–753.

Appendix A: Results File 301

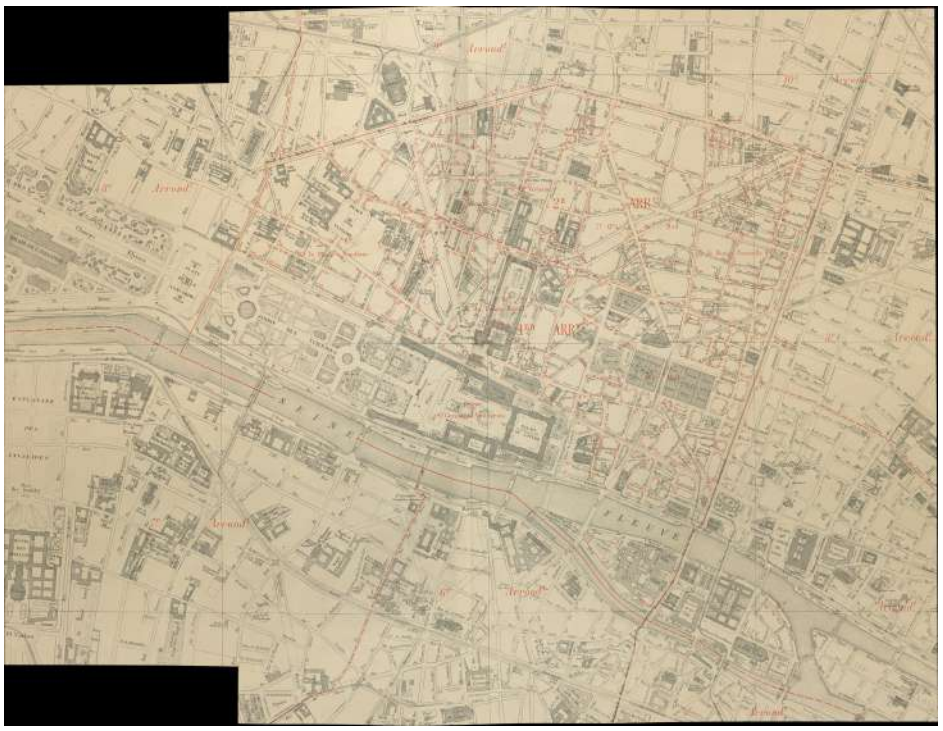

Figure .30: File 301, Input Image

Figure .31: File 301, Ground Truth Image

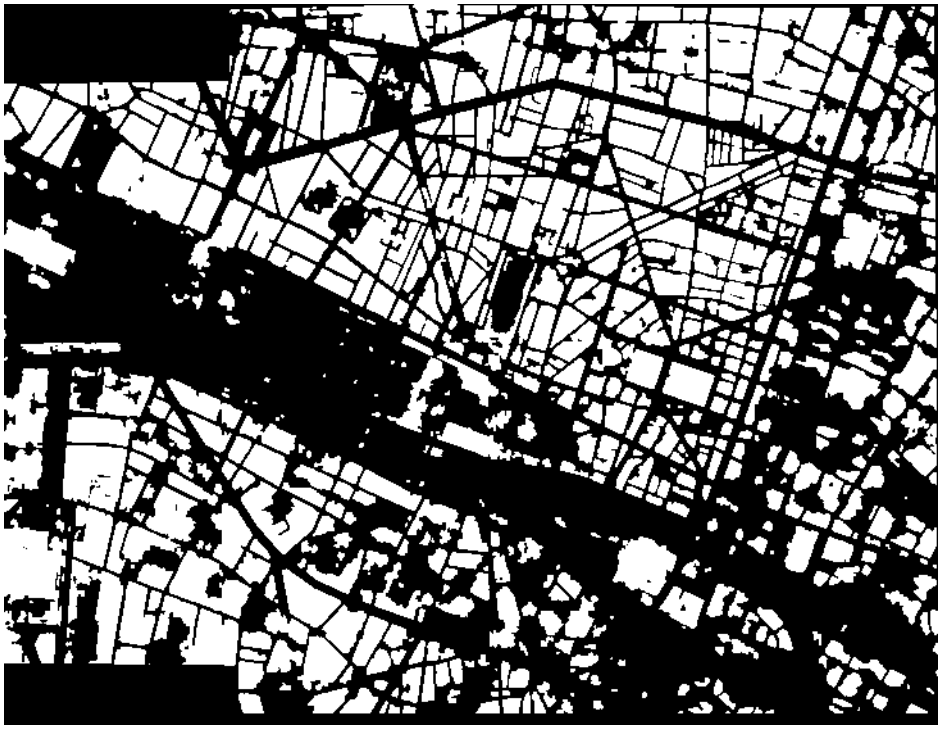

Figure .32: File 301, Prediction U-Net

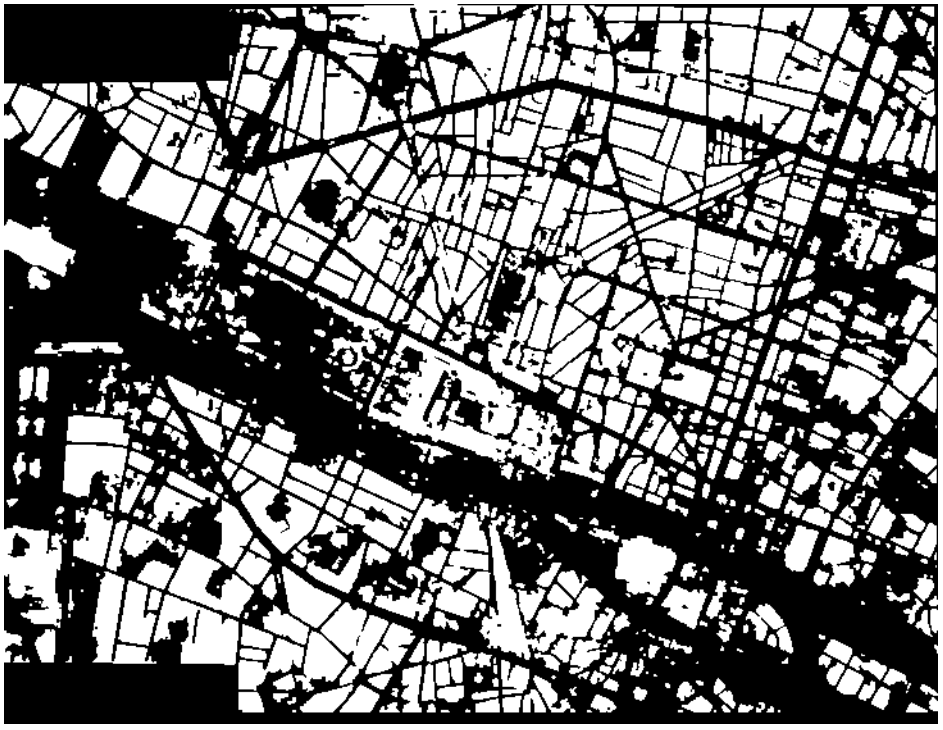

Figure .33: File 301, Prediction ResUnet

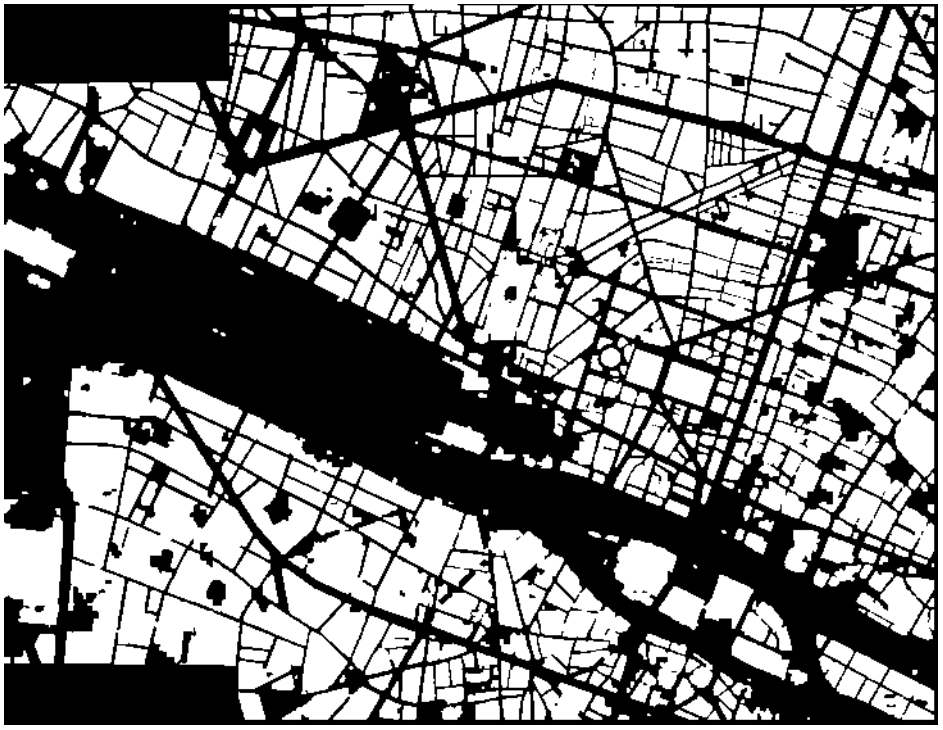

Figure .34: File 301, Prediction SwinUnet

Figure .35: File 301, Vectorized and Generalized Building Block Polygons U-Net

Figure .36: File 301, Vectorized and Generalized Building Block Polygons ResUnet

Figure .37: File 301, Vectorized and Generalized Building Block Polygons SwinUnet

Appendix B: Results File 302

Figure .38: File 302, Input Image

Figure .39: File 302, Ground Truth Image

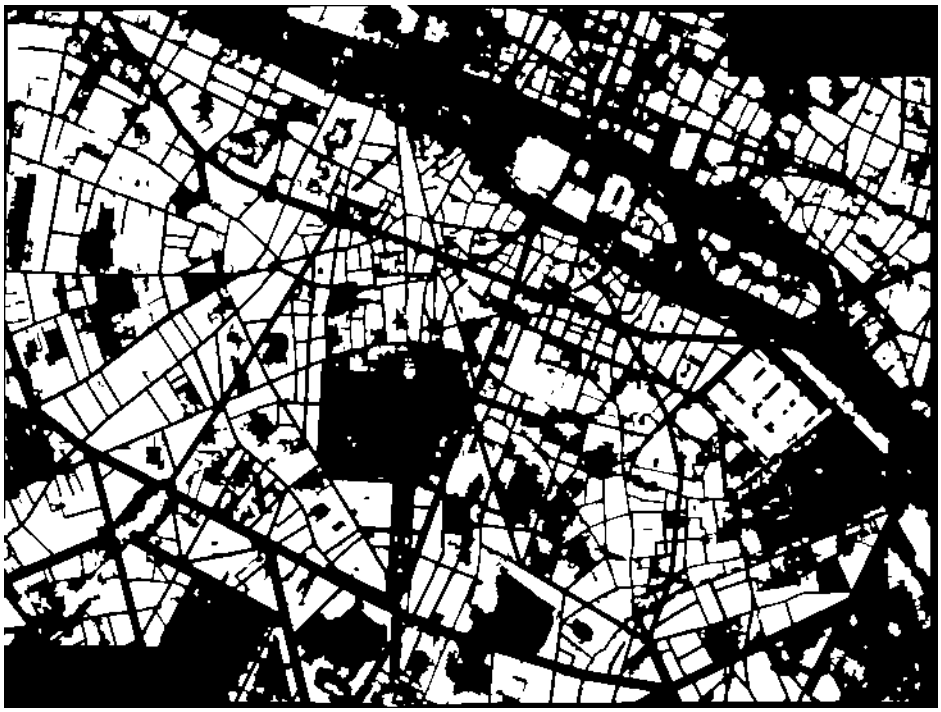

Figure .40: File 302, Prediction U-Net

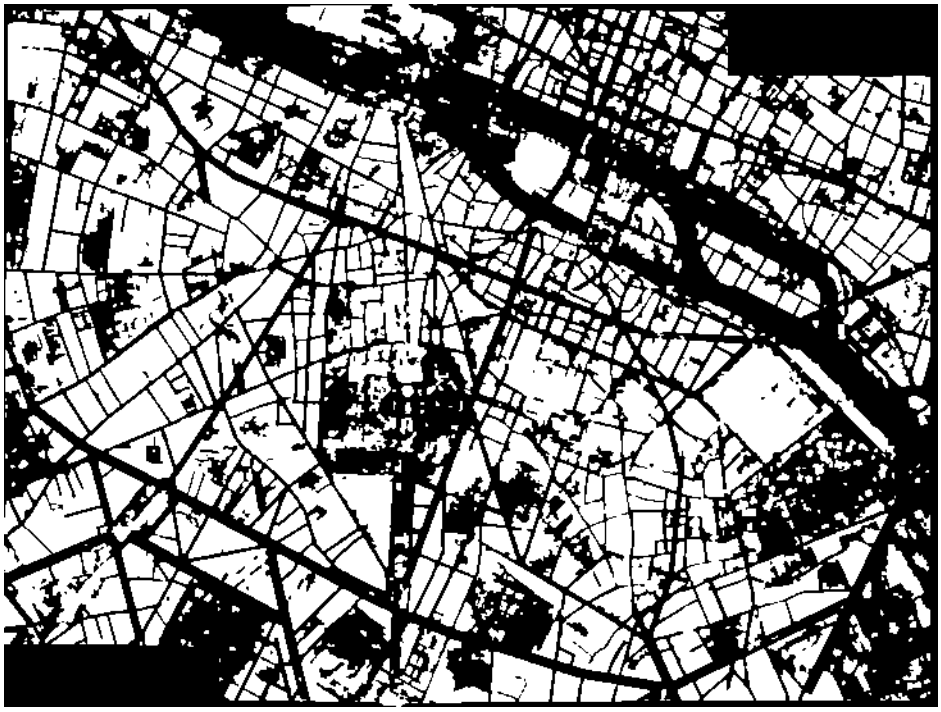

Figure .41: File 302, Prediction ResUnet

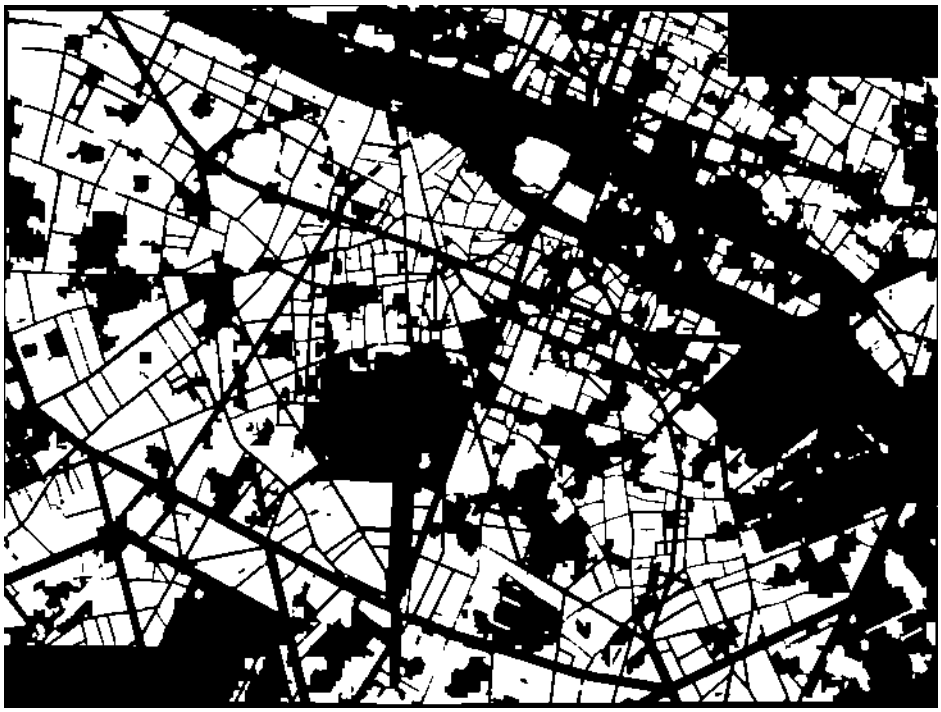

Figure .42: File 302, Prediction SwinUnet

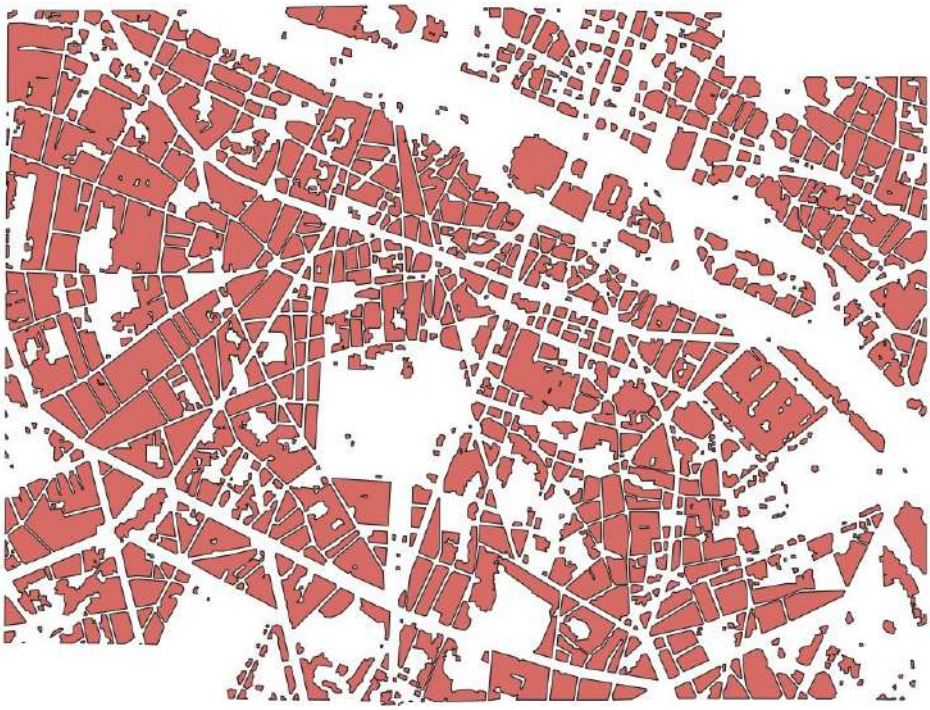

Figure .43: File 302, Vectorized and Generalized Building Block Polygons U-Net

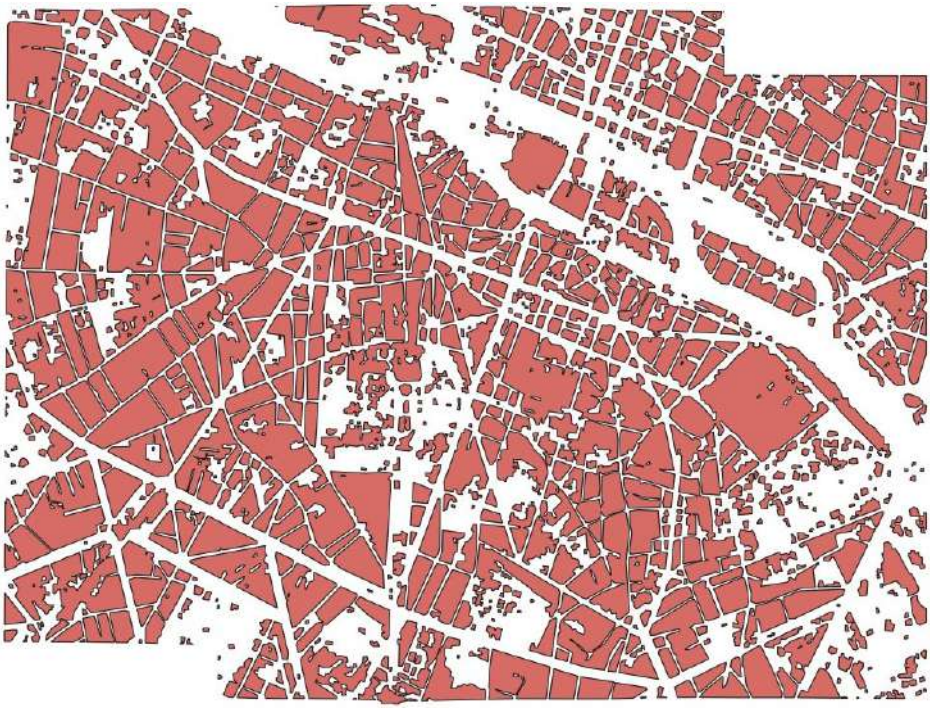

Figure .44: File 302, Vectorized and Generalized Building Block Polygons ResUnet

Figure .45: File 302, Vectorized and Generalized Building Block Polygons SwinUnet

Appendix C: Results File 303

Figure .46: File 303, Input Image

Figure .47: File 303, Ground Truth Image

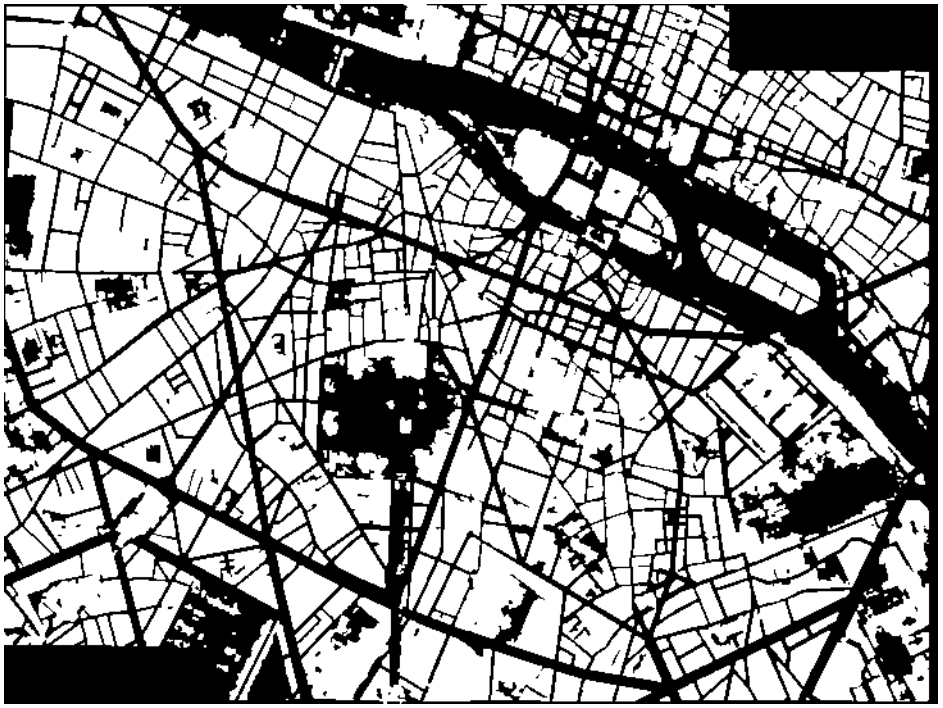

Figure .48: File 302, Prediction U-Net

Figure .49: File 302, Prediction ResUnet

Figure .50: File 302, Prediction SwinUnet

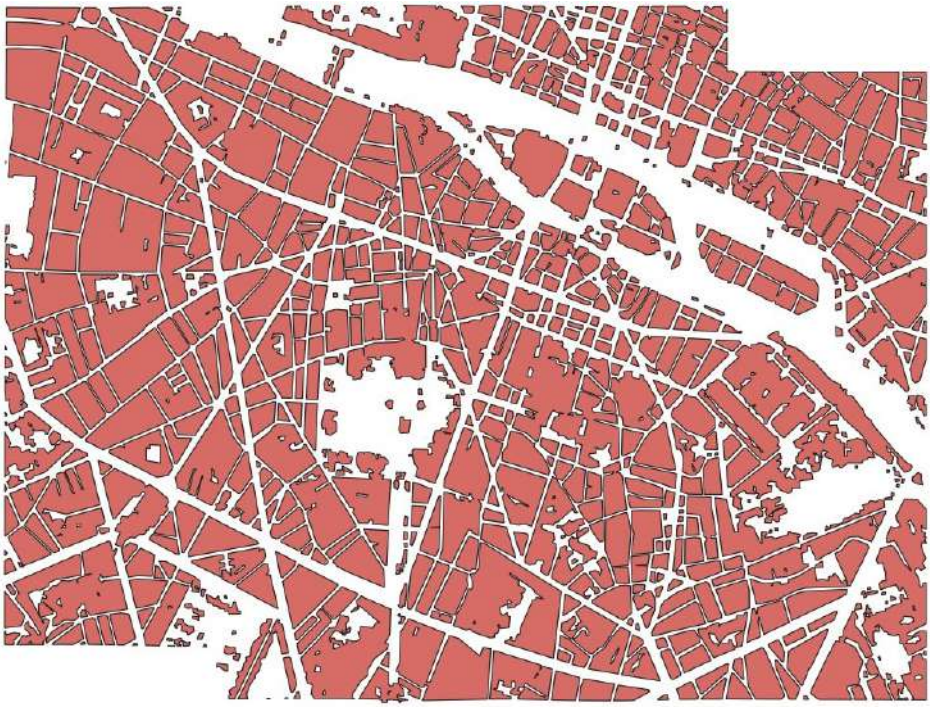

Figure .51: File 303, Vectorized and Generalized Building Block Polygons U-Net

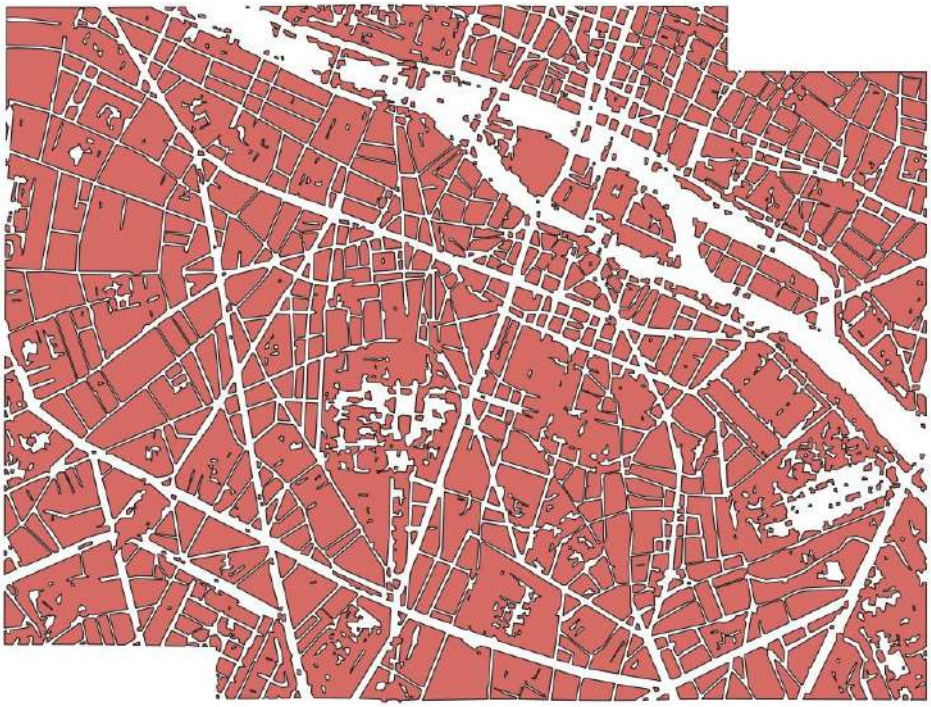

Figure .52: File 303, Vectorized and Generalized Building Block Polygons ResUnet

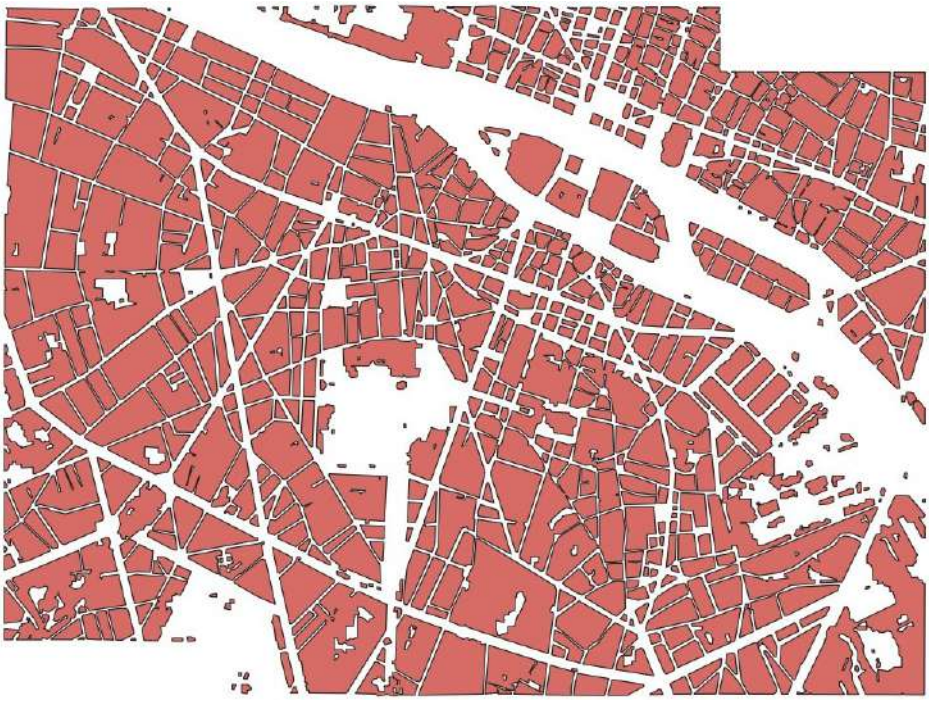

Figure .53: File 303, Vectorized and Generalized Building Block Polygons SwinUnet