University of Zurich[UZH]

Exploring Deep Learning for Deformative Operators in Vector-Based Cartographic Road Generalization

GEO 511 Master's Thesis

Author
Nicolas Beglinger
17-711-722

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

Faculty representative
Prof. Dr. Robert Weibel

31.01.2023
Department of Geography, University of Zurich

Abstract

Cartographic generalisation is the process by which geographical data is simplified and abstracted to increase the legibility of maps at reduced scales. As map scales decrease, irrelevant map features are removed (selective generalisation), and relevant map features are deformed, eliminating unnecessary details while preserving the general shapes (deformative generalisation). The automation of cartographic generalisation has been a tough nut to crack for years because it is governed not only by explicit rules but also by a large body of implicit cartographic knowledge that conventional automation approaches struggle to acquire and formalise. In recent years, the introduction of Deep Learning (DL) and its inductive capabilities has raised hope for further progress. This thesis explores the potential of three Deep Learning architectures — Graph Convolutional Neural Network (GCNN), Auto Encoder, and Recurrent Neural Network (RNN) — in their application on the *deformative* generalisation of *roads* using a *vector*-based approach. The generated small-scale representations of the input roads differ substantially across the architectures, not only in their included frequency spectra but also in their ability to apply certain generalisation operators. However, the most apparent learnt and applied generalisation operator by all architectures is the smoothing of the large-scale roads. The outcome of this thesis has been encouraging but suggests to pursue further research about the effect of the pre-processing of the input geometries and the inclusion of spatial context and the combination of map features (e.g. buildings) to better capture the implicit knowledge engrained in the products of mapping agencies used for training the DL models.

Keywords: Cartographic Generalisation, Deep Learning, Automation, Graph Convolutional Neural Network (GCNN), Autoencoder, Long Short-Term Memory Neural Network (LSTM), Road Generalisation, Line Simplification

Acknowledgements

I would like to express my gratitude to my supervisors Professor Dr. Robert Weibel, Dr. Cheng Fu, and Dr. Zhiyong Zhou, for their invaluable assistance and guidance in tackling this challenging subject matter. It was a privilege to collaborate with them in this new area of exploration.

Additionally, I would like to express my gratitude to swisstopo, and especially to Dr. Roman Geisthövel, for generously providing me with all the necessary cartographic data with unwavering support and cooperation. Their contributions were instrumental in the success of this thesis.

Furthermore, I would like to extend my gratitude to the following individuals for their immense mental and technical support:

- Jan Winkler, with whom I spent hours discussing various implications of cartography and Deep Learning. These discussions were indispensable for the success of this thesis.
- Hannah Sommer, Patrick Luchsinger, Inga Birkhölzer, and Gabriele Durband for providing me with support and encouragement as fellow colleagues during breaks, especially during the final stages of the thesis.
- Colette Mathis, for her continuous encouragement and never-ending support.
- My family and friends for being there when I needed them and for proofreading this thesis.

Contents

Abstract i

Acknowledgements ii

1 Introduction 1

2 Background 3
2.1 A Brief History of Automated Map Generalisation . . . 3
2.2 Introduction of Deep Learning Architectures . . . 3
2.2.1 Rationale . . . 3
2.2.2 Graph Convolutional Neural Networks . . . 4
2.2.3 Auto Encoder . . . 4
2.2.4 Recurrent Neural Networks . . . 5
2.3 Deep Learning Approaches in Road Generalisation . . . 7
2.3.1 Raster-Based Approaches in Road Generalisation . . . 7
2.3.2 Vector-Based Approaches in Road Generalisation . . . 7
2.4 Research Gaps . . . 8
2.5 Research Objective . . . 9

3 Methodology 10
3.1 Data . . . 10
3.2 Preprocessing . . . 10
3.2.1 Preprocessing for GCNN . . . 11
3.2.2 Preprocessing for CAE and LSTM . . . 11
3.2.3 Workflow . . . 12
3.3 Implementation of Models and Training . . . 17
3.3.1 GCNN . . . 18
3.3.2 Auto Encoder . . . 19
3.3.3 RNN . . . 19
3.3.4 Fourier Loss Extension . . . 20
3.4 Evaluation . . . 21
3.4.1 Quantitative Evaluation . . . 21
3.4.2 Qualitative Evaluation . . . 24

4 Results 25
4.1 Quantitative Description . . . 25
4.1.1 GCNN . . . 25
4.1.2 Convolutional Auto Encoder . . . 27
4.1.3 RNN . . . 29
4.1.4 Comparison of Architectures . . . 33
4.1.5 Further Analytics . . . 33
4.2 Qualitative Description . . . 36
4.2.1 Effect of Number of Layers in GCNNs . . . 36
4.2.2 Comparison between simple CAE and ResUNet . . . 37
4.2.3 Comparison of RNN Collapsing Strategies . . . 39
4.2.4 Comparison of Architectures . . . 43
4.3 Loss Curves . . . 46

5 Discussion **48**
5.1 Performance of Individual Architectures 48
5.1.1 GCNN 48
5.1.2 CAE 48
5.1.3 RNN 50
5.2 Observations Regarding All Architectures 51

6 Conclusion **52**
6.1 Contributions 52
6.2 Insights 52
6.3 Limitations 53
6.4 Outlook 53

Chapter 1

Introduction

"Map generalization is one of the central concepts in map design. A map is a generalised, simplified abstraction of reality.": Brassel and Weibel (1988, p. 229) state that the concrete reality is the sum of general and specific aspects. The general aspect of reality is the information that remains after removing the 'random' or the 'unimportant', being left with only the general, crucial elements of reality.

The Map Generalisation Problem The over-arching problem in cartographic generalisation is that by increasing the covered real-world area without increasing the information display size (i.e. transitioning to a smaller map scale), the available space for each map element gets reduced by the square. Thus, decisions on what information to omit have to be made. Information can be omitted in several ways. The most obvious way is to exclude certain map elements (e.g. a small footpath). Other possibilities include reducing precision (e.g. simplification) or accuracy (e.g. displacement). The generalisation of roads is also subject to different specific generalisation operations. Broadly speaking, one could distinguish three different categories: Operations that change the graphical symbology (e.g. enlargement), operations that change the geometry of roads (e.g. smoothing), and operations that select (or deselect) certain roads from being displayed on the generalised map. In this thesis, the two latter are called deformation and selection. The difference between deformation and selection can be seen in Figure 1.1.

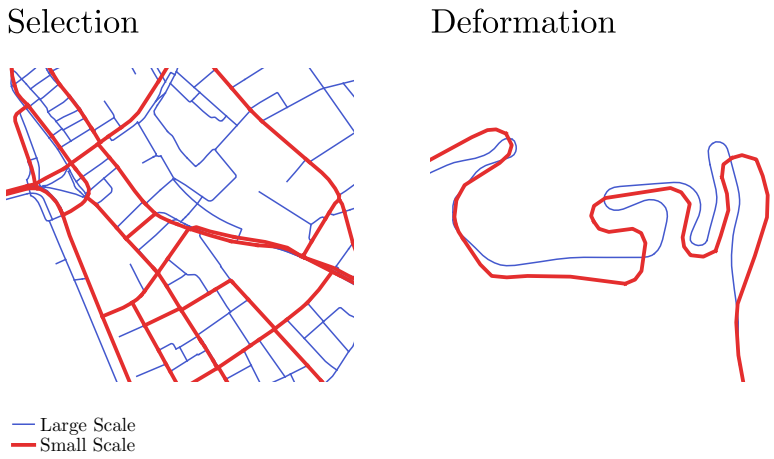


Figure 1.1: Difference between selection and deformation (1:25'000 → 1:100'000 in this example)

Challenges for Automating Map Generalisation Many map providers, including national mapping agencies, produce maps at different scales and times. The earth's surface constantly changes due to natural phenomena and human-made activities. As a result, maps must be updated

frequently. However, the current updating processes are labour-intensive and time-consuming (Lee et al., 2017), which creates the need to automate certain processes. Cartographic generalisation is conducted using explicit and implicit cartographic knowledge. Explicit knowledge, that is, rules (e.g. for smoothing cartographic lines), can be directly stated by trained cartographers and is thus generally translatable into traditional computer programs. Implicit knowledge (e.g. how to best alter the geometry of an object to increase its legibility), on the other hand, influences the work of cartographers in a way they can't describe directly because they have internalised it inductively during their training and practice. As a result, generalisation is often governed by generalisation goals (e.g. Spiess et al. 2003) describing very fundamental principles. For years, the automation of map generalisation has been difficult to achieve. In addition to the challenge of the technical implementation of explicit rules, a large body of implicit knowledge poses a problem that could not be solved entirely by conventional approaches. The above-mentioned generalisation principles are applied by cartographers holistically, and it is the abundance of possible combinations of these principles and the resulting cartographic generalisation operations (e.g. simplification, smoothing, enlargement, aggregation, and displacement of map objects (Weibel, 1995)) that makes it impossible to explicitly state all rules and implement them using traditional computer programs.

After over 50 years of research (with early work already done in the 1960s, e.g. Tobler 1966), digital map generalisation has already developed to a high level. However, as already mentioned, it turned out that conventional approaches cannot go much further in explicitising implicit cartographic knowledge. Deep Learning is a promising new approach to the cartographic generalisation problem that may raise the possibility for further progress on the matter. Although there is a need to explore the applicability of different Deep Learning architectures to all three above-stated generalisation categories, this thesis cannot engage with all of them. Hence, the first restriction of this thesis is its exclusive focus on road map objects. The second restriction is that only vector-based approaches are explored. Finally, the third restriction is the generalisation category: The usability of vector-based Deep Learning approaches for road network *selection* has already been explored and demonstrated by Zheng et al. (2021) (see more in Section 2.3), and *graphical symbology* is highly interconnected with raster-based operations (not with vector-based operations). Thus, this thesis focuses on *deformative operations* (i.e. operations that change the geometry of roads), namely simplification, smoothing, exaggeration, and displacement.

This thesis is written within the framework of the research project "DeepGeneralization" of the Department of Geography at the University of Zurich. Partners of the project are the Swiss Federal Office of Topography swisstopo[1], the French National Institute of Geography IGN[2], and the Institute of Cartography and Geoinformatics IKG at the Leibniz University Hannover[3].

[1] `https://www.swisstopo.admin.ch/`
[2] `https://www.ign.fr/`
[3] `https://www.ikg.uni-hannover.de/`

Chapter 2

Background

2.1 A Brief History of Automated Map Generalisation

First approaches, starting in the 1960s, focused on single, isolated algorithms representing specific steps in the process chain of generalisation (e.g. Tobler 1966), such as the simplification of lines. Later, approaches that separated structural knowledge from procedural knowledge became important (Weibel, 1995). Structural knowledge allows the recognition of essential map elements, procedural knowledge enables the execution of generalisation algorithms. However, as described in Chapter 1, it is challenging to explicitize implicit knowledge that cartographers learn during their education and practice. Because it seemed impossible to create an autonomous algorithm that follows all possible generalisation rules, a paradigm shift towards what Weibel (1991) called "amplified intelligence" took place in the early 1990s. Approaches that correspond to this paradigm imply a shared cognitive workload between humans and computers, where the human is always in control of the process, and the machine acts as decision support. While approaches using neural networks and other machine learning approaches existed in the 1990s (e.g. Werschlein and Weibel 1994), Beard (1991) proposed an influential approach that was based on constraints and soon was adopted by others (e.g. Ruas 1998, Weibel and Dutton 1998). It turned out that the formulation of constraints was more accessible and versatile than the formulation of rules. Nevertheless, these systems proved difficult and time-consuming, especially when many parameters were used (Weibel et al., 1995; Taillandier et al., 2011), which led to the systems being limited by the irrefutable trade-off between completeness and complexity. As a result, smaller solutions that fit well into the generalisation pipeline were often preferred over these heavy systems (Petzold et al., 2006). These more lightweight solutions are now often packaged in interactive environments that support cartographers in their work. This way of working is essentially a recollection of the *amplified intelligence* paradigm. In summary, current conventional approaches for automated map generalisation have come a long way, using powerful generalisation algorithms that facilitate the work of cartographers. However, since the *knowledge acquisition bottleneck* (Weibel et al., 1995) is hard, if not impossible, to overcome, these algorithms still need much human intervention and control.

2.2 Introduction of Deep Learning Architectures

2.2.1 Rationale

Deep Learning is the process by which machines learn to extract specific patterns from data using multi-layer neural networks, which are systems designed to imitate human neural actions and brain activities (Haykin, 2009). They are inspired by the way that neurons are (inter-)connected and control their activation behaviour in a self-organised manner. Deep Learning has been successfully used in computer vision, video and speech recognition, natural language processing, and other applications (Lecun et al., 2015). It enables information systems to learn about data in an inductive way, which means learning about general rules by inducing them from specific examples, circumventing the need to explicitise hidden, implicit knowledge. While the big hurdle in conventional generalisation automation approaches is acquiring knowledge about generalisation rules, which are then applied to specific examples, Deep Learning may be the perfect candidate to overcome the still valid *knowledge acquisition bottleneck*. Neural networks are well adapted to problems where knowledge is implicit in the data (Touya et al., 2019) and has recently been applied to map generalisation problems concerning buildings (e.g. Feng et al. 2019, Yan et al. 2019, Yan et al. 2020a)

and roads (e.g. Courtial et al. 2020, Du et al. 2021, Zheng et al. 2021, Du et al. 2022). In the following sections, three different Deep Learning architectures that this thesis engages with are introduced.

2.2.2 Graph Convolutional Neural Networks

Graph Neural Networks (GNNs) are Deep Learning architectures that are adapted for working on graph data. An excellent overview of different GNN architectures is given by Zhang et al. 2019. Graphs are data structures consisting of a set of nodes (or vertices) V and a set of edges E that connect the nodes. In a GNN, the data points are represented by the nodes in the graph, and the edges between them represent the relationships or dependencies between the data points. A sub-type of the GNN is the Graph Convolutional Neural Network. Its core functionality is to learn meaningful representations of a graph by aggregating information from the neighbourhood $N_v, v \in V$ of each node v in the graph. GCNNs solve problems that have been challenging traditional Convolutional Neural Networks (CNNs), which are commonly used for image classification and recognition tasks, by expanding their functionality to non-grid-like data structures (Wu et al., 2022). Just as CNNs use convolutional filters to extract local features from images, GCNNs use graph convolutional filters to extract local features from the graph structure. The convolutions are conducted using a process called message passing, which refers to the aggregation of information from the neighbourhoods of each node in the graph and using it to update the representation of the node (Wu et al., 2022). Message passing can thus be seen as a form of information exchange between nodes in the graph. Each node communicates with its neighbours and aggregates information from them to form a new, more holistic representation of the data. This process is repeated iteratively, with each node $v^t \in V$ updating its representation based on the representations of its neighbours $n \in N_{v^{t-1}}$ in the previous iteration (Wu et al., 2022). The message passing is implemented using a graph convolutional operator, which is a function that takes as input a node and its neighbours and produces a new feature representation for the node. The graph convolutional operator is defined by a set of trainable parameters learned during the training process. A visualisation of a message passing step is depicted in Figure 2.1. GCNNs have been successfully used in various research fields such as social analysis, fraud detection, traffic prediction, computer vision, and much more (Zhang et al., 2019). However, GCNNs are known for the over-smoothing problem, leading to them being mostly shallow (only few layers) (Chen et al., 2020; Wu et al., 2022) which limits the amount of message passing steps and, therefore, the size of the widest possible "receptive field".

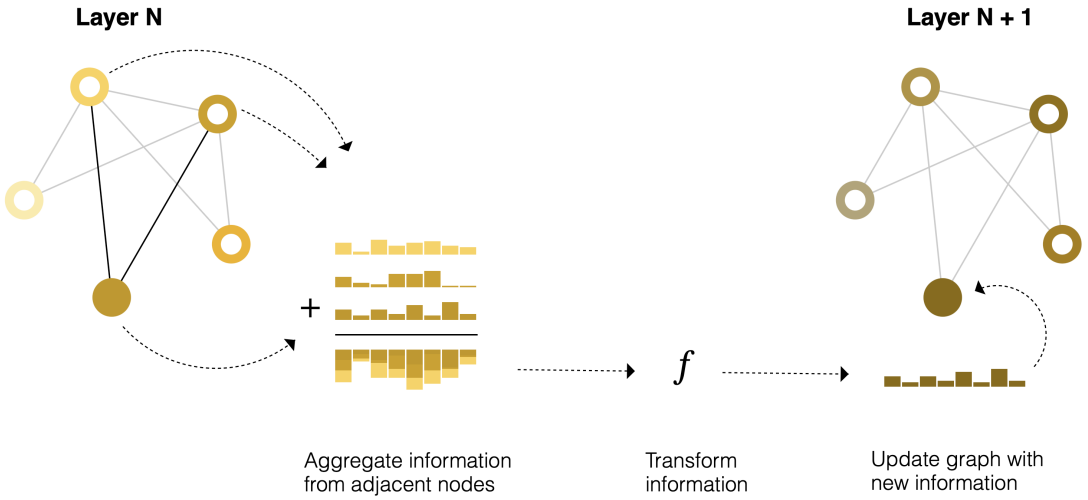


Figure 2.1: Visualisation of a message passing step (source: https://distill.pub/2021/gnn-intro/)

2.2.3 Auto Encoder

"An autoencoder is a neural network trained to attempt to copy its input to its output. Internally, it has a hidden layer h that describes a *code* used to represent the input." (Goodfellow et al., 2016, p. 499) The hidden layer h is also described as latent space (Hinton and Salakhutdinov, 2006). An auto encoder consists of an encoder that transforms the input x into h and a decoder that tries to restore x by decoding h into x'. A schema of an auto encoder is depicted in Figure 2.2. In the case of undercomplete auto encoders, the dimension of h is smaller than the dimension

of the input x, forcing the network to capture the most salient features of the training data. This relates well to the map generalisation problem, where a general representation of the input (i.e. the large-scale map features) has to be found. The usual form of an auto encoder does not require targets/ground truths. Because of this feature, they are usually considered unsupervised neural networks. Auto encoders are applied in different fields such as dimensionality reduction, classification, anomaly detection, and denoising applications (Michelucci, 2022) and have also been combined with Convolutional Neural Networks, leading to Convolutional Auto Encoders (CAE). CAEs allow the creation of abstract representations of the initial inputs by removing noise and redundant information (Pintelas et al., 2021).

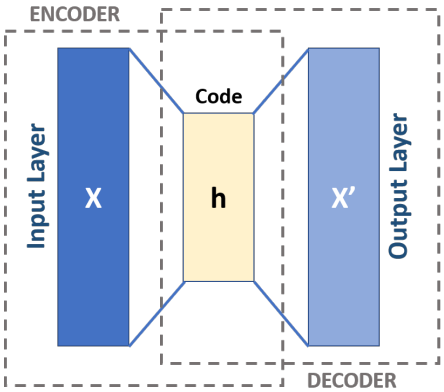


Figure 2.2: Schema of an Auto Encoder (Hinton and Salakhutdinov, 2006)

A specific, well-known example of a CAE is the U-Net architecture, which was initially developed for n-dimensional biomedical image segmentation (i.e. it classifies each pixel of the input picture) by Ronneberger et al. (2015). Its visual structure is depicted in Figure 2.3 and consists of three parts:

1. An *encoder section*, consisting of four convolutional blocks that each halve the spatial dimensions while doubling the number of channels. They do so by using convolutional blocks that aggregate pixel data using a moving window with learnable weights and max pooling the resulting aggregated images.
2. A *bottleneck module*, that represents h.
3. A *decoder section*, that reverses the steps of the encoder section.

The affiliation of the U-Net architecture to the auto encoders is controversial[1]. Although it is composed of the typical auto encoder parts (encoder, bottleneck module, decoder (Yin et al., 2022)), it requires ground truth data that serves as target and its skip-connections bypass the need to find a single holistic, compressed representation of the input in the latent space. In this thesis, however, they are treated as auto encoder. From the original publication by Ronneberger et al. (2015), many further developments have been proposed that modify the convolutional blocks. The ResUNet (Jha et al., 2019) that is used in this thesis adds skip connections to the convolutional blocks.

2.2.4 Recurrent Neural Networks

Recurrent neural networks (RNNs) are a family of neural networks that are adapted to process sequential data. "Much as a convolutional network is a neural network that is specialised for processing a grid of values $\mathbf{X}$ such as an image, a recurrent neural network is a neural network that is specialised for processing a sequence of values $\mathbf{x}^{(1)}, ..., \mathbf{x}^{(\tau)}$." (Goodfellow et al., 2016, p. 367) RNNs, like the convolutional operators of the GCNN and U-Net architectures, are therefore based on the concept of *parameter sharing* (Goodfellow et al., 2016). But unlike the convolutional operators found in the GCNN and in the U-Net architectures, where the output of each member m is a function of a small number of neighbours of the input, in RNNs, each member of the output is a function of all the previous members of the output. Thus, instead of using a consistent

[1] see for example `https://www.researchgate.net/post/Are_U-net_and_encoder-decoder_network_the_same`

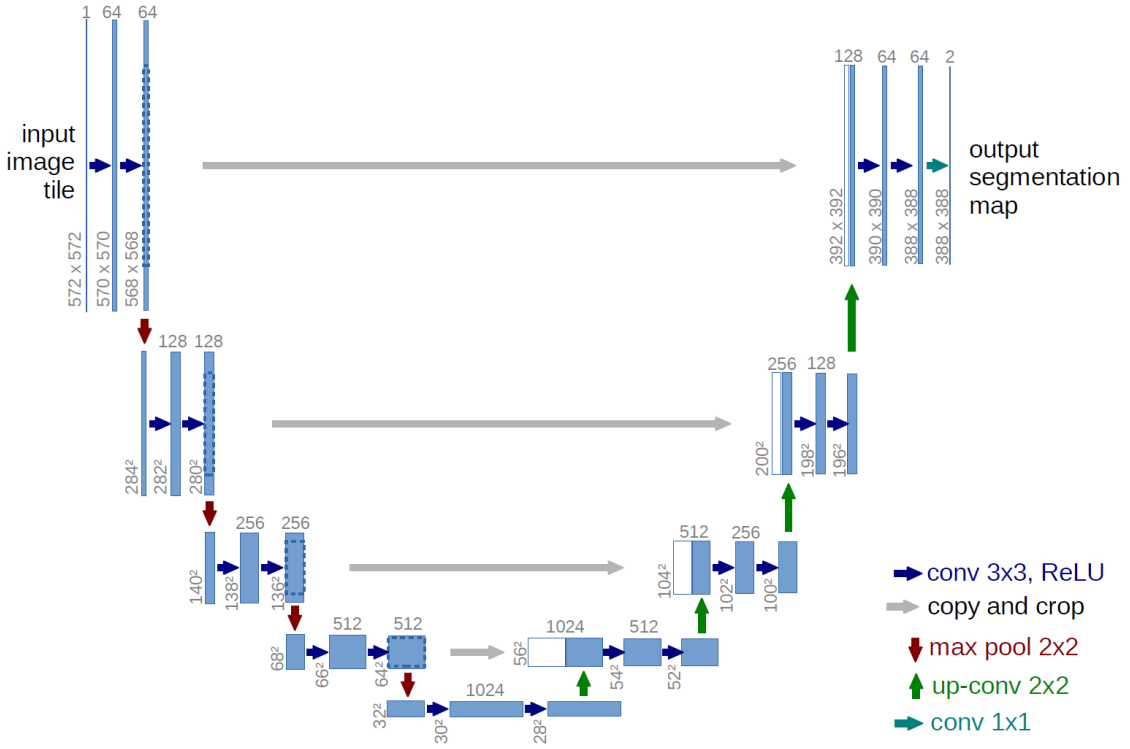


Figure 2.3: Schema of the U-Net from the original paper of Ronneberger et al. (2015)

1D convolution kernel $f(m^{t-1}, m^t, m^{t+1})$ moving over the sequence, the output is produced using a consistent update rule $f(m^{t0}, ..., m^{t-1}, m^t)$. This recurrent formulation allows the sharing of parameters through very deep neural networks (Goodfellow et al., 2016).

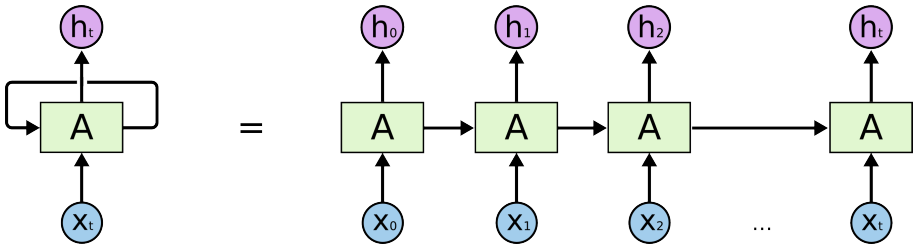


Figure 2.4: Schema of an RNN-Architecture (source: `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`)

The vanishing gradient problem is a common problem in Deep Learning and refers to the error becoming smaller in each layer of a neural network during backpropagation, which leads to an increased learning time (Hochreiter, 1998) and a waste of computational resources. RNNs, as commonly very deep neural networks, are said to suffer especially from the vanishing gradient and the long-distance dependency problem (Le and Zuidema, 2016). To overcome these limitations, further developments on the plain RNN emerged. One of the proposed architectures is the *Long Short-Term Memory* neural network (LSTM) (original publication by Hochreiter and Schmidhuber 1997). As the name suggests, it enhances the network's capability to "remember" information about steps that are further away in time. It does so by introducing a *cell state* and different *gates*. The cell state is like a "conveyor belt"[2] that carries information through the entire network without major interactions. It is represented by the upper horizontal line in Figure 2.5. LSTM cells thereby get information from 1. their input x_t and 2. the information of previous LSTM cells in the form of a hidden state h_{t-1} and the cell state c_{t-1}. The information that an LSTM cell gets is then modified using three different gates and point-wise matrix operations. The three gates are the following (Smagulova and James, 2020):

1. The *Forget Gate* decides what information is removed from the cell state.

[2]Wording from `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`

2. The *Input Gate* decides what information is added to the cell state.

3. The *Output Gate* decides what information from x_t, h_{t-1}, and c_t and their deratives is output.

As proposed by Mai et al. (2022), the location-to-polyline relation can be seen as an analogy of the word-to-sentence relation. Thus, polylines compare well to the sequential input data that usually serves as input for RNNs.

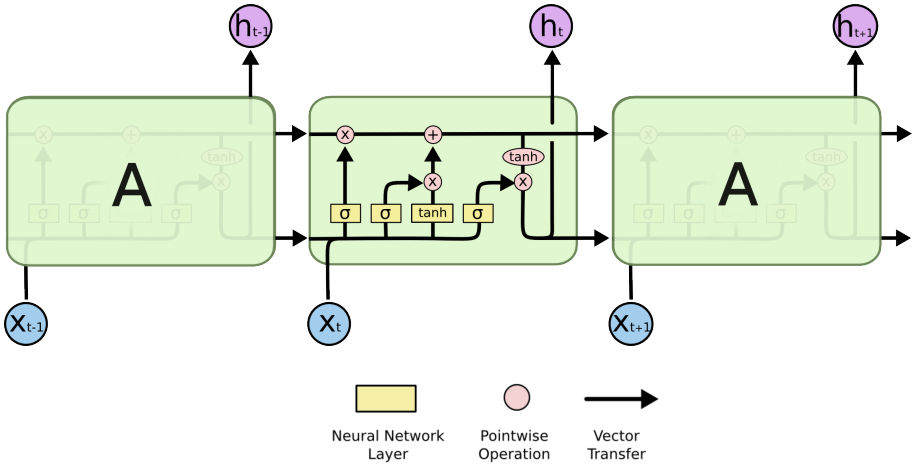


Figure 2.5: Schema of an LSTM-Architecture (source: `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`)

2.3 Deep Learning Approaches in Road Generalisation

The application of Deep Learning in map generalisation is a rather new subject. Although there was some work done about the use of neural networks in the 1990s (e.g. Werschlein and Weibel 1994), the use of Deep Learning has remained largely untouched by map generalisation research for the last 15 years. However, the advances in Deep Learning and pioneering studies such as the work by Sester et al. (2018) or Mai et al. (2022) laid a foundation for subsequent work in this realm. Touya et al. (2019) presents an overview of the possible advantages of Deep Learning in map generalisation. The following two sections elaborate on the few Deep Learning approaches that exist in road generalisation. Broadly speaking, there are two main approaches:

2.3.1 Raster-Based Approaches in Road Generalisation

The first approach is to rasterise the data (that is initially typically stored as vector data) and make use of the many Deep Learning architectures that are adapted to image processing like Convolutional Neural Networks (CNNs, see for example O'Shea and Nash 2015) or Generative Adversarial Networks (GAN's, see for instance Creswell et al. 2018), thereby conceptualising map generalisation as a pure graphical problem. Courtial et al. (2020) explored the potential of such an approach for the generalisation of mountain roads by using the U-Net architecture (Ronneberger et al., 2015). The authors conceptualised the roads as binary images with road pixels and non-road pixels. The neural network then learned what pixels should represent the generalised roads, given the large-scale roads as input. Their approach produced encouraging results in most cases. However, their model did not produce results that were close to the quality of the reference. Among other problems, the generalised roads showed topological inconsistencies, where few black (non-road) pixels split some roads. A few false-negatively misclassified pixels (classified as non-road when they should have been classified as a road) are often enough to split a road but are not enough to cause a big change in the model parameters because the chosen loss function considers the overall performance of the generated image.

2.3.2 Vector-Based Approaches in Road Generalisation

The second approach is to make use of the topological information that is explicitly encoded in vector data (for lines and polygons), thereby ensuring topological integrity, which poses a problem in raster-based models. As introduced in Section 2.2.2, graphs are a suitable format to represent

topological relationships within data. The use of graph theory has also been connected with the map generalisation problem in conventional approaches (e.g. Mackaness and Beard 2013). It further has been introduced above that GCNNs are a suitable technique to apply Deep Learning to graphs. While there are already a few studies that engage with the use of GNNs for map generalisation problems, there is only one study that explores its potential on the road generalisation problem (for buildings, see for example Yan et al. 2020b or Yan et al. 2019): Zheng et al. (2021) used GCNNs for automatic road network selection by conceptualising the selection as a *node classification* problem. First, they created *inverse graphs* from road networks by conceptualising the roads as edges and the intersections as nodes. Then, nodes and edges where inverted. As a result, each node represented a road in the graph. They then compared various kinds of GCNNs in their ability to predict, which roads should be kept during generalisation by classifying the nodes as "keep" or "remove" nodes. As mentioned in Section 2.2.2, the nodes can hold information. Zheng et al. (2021) used this property and enriched the road nodes with three features: The roads' types, lengths, and coordinates. The results indicated that GCNs are an appropriate tool for road network selection and are superior to traditional machine learning models. The work of Yan et al. (2020b), although it is applied to the GNN-based shape coding of *buildings*, gives relevant insights into the engineering and identification of geometric features in the context of graph-based location encoding in general. As depicted in Figure 2.6, they proposed a distinction between local and regional features for each node. Especially the local features may well be applied in the road domain. Very recent work by Yu and Chen (2022) examined the use of auto encoders for polyline simplification. As described in Section 2.2.3, auto encoders usually aim to reconstruct an input after narrowing it down into a bottleneck. Thereby, only the general information is kept. The authors used this functionality to generate simplified versions of input polyline vectors by stacking multiple auto encoders in sequence. Thus, the output of every encoding step provides a more simplified version of the output of the preceding step. Their proposed model succeeds in producing simplified versions of the input roads and compares well or even exceeds the performance of traditional simplification algorithms. However, the training was conducted in an unsupervised setting. Hence, the proposed methodology does not include cartographic knowledge of cartographers in the training of the models, but rather presents an efficient way to produce simplified versions of the input in multiple scales.

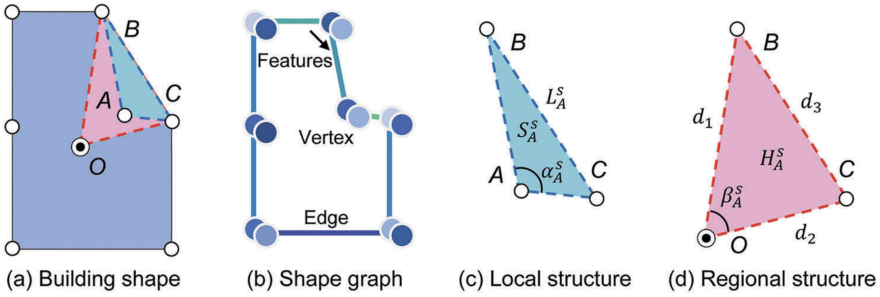


Figure 2.6: Proposed graph structure by Yan et al. (2020b)

2.4 Research Gaps

Based on the above review of the relevant literature, the following research gaps are identified and tackled:

1. **Application of GCNNs in *deformative* vector-based road generalisation**
   Currently, vector-based Deep Learning approaches using GCNNs have only been used to carry out *selection* tasks in road generalisation, but not on generalisation tasks that imply *deformation*.

2. **Application of Auto Encoders in *supervised* deformative road generalisation**
   Auto encoders have only been used for polyline simplification in unsupervised settings, thus their potential on extracting cartographic knowledge from examples has not been explored.

3. **Application of *RNNs* in road generalisation**
   Although proposed by Mai et al. (2022), no attempts to explore the potential of RNNs in the application on road generalisation have been made.

4. **Comparison of Deep Learning Models**
   Little is known about which Deep Learning architectures may contribute the most to deformative road generalisation and what kind of pre-processing could improve the models' performance.

2.5 Research Objective

To address the above-mentioned research gaps, the research objective of this thesis is to explore the potential of the three different Deep Learning architectures GCNN, CAE, and RNN in their application on deformative, vector-based road generalisation. Thereby, relevant thoughts, workflows, and practices are developed and documented. As a result, this thesis should generate valuable first insights into this matter and serve as a "head-start" for possible future work.

Chapter 3

Methodology

3.1 Data

As mentioned above, this thesis only engages with deformative generalisation operations. Thus, a pre-selected road network was needed because the default generalisation workflow assumes selection to happen before deformation (Spiess et al., 2003). *swisstopo*, as partner of the thesis and the *DeepGeneralisation* project, provided the needed road network data. The structure of the data is described in the following section:

Cartographic (intermediate) Products The basis of swisstopo's large scale [k]artographic reference model (KRM25) is the openly available *swissTLM3D* (topographic landscape model). The transition from swissTLM3D to KRM25 is not characterised by geometric generalisation operators but by the selection of cartographically relevant attributes. The KRM25 is the basis for different large-scale digital [k]artographic models DKM25, DKM50, and DKM100. The DKM's scales have substantial implications on the detail reduction level, both on selection and deformative generalisation operators. While the transition from KRM25 to DKM25 causes only slight shifts and no significant generalisation regarding lines, the transitions to smaller scales are characterised by (considerable) positional shifts during the generalisation process (Spiess et al., 2003). The transition to DKM100 causes the greatest deformation of the large-scale generalisations. To simplify the qualitative, graphical evaluation process, this thesis focuses on the transition between the KRM25 and the DKM100.

Matching To assess the difference between the roads at the different scales (or the necessary deformation to transform the road from one scale to the other respectively), the representations of the roads at their different scales had to be matched. Due to *swisstopo*'s data management, no spatial matching algorithms were needed. Instead, each road is indexed with a unique UUID that is stored in a join table, which allows for tracking the roads along the different scale representations. The matching could then be conducted using this join table.

Data Storage and Management To speed up spatial queries and facilitate the filtering, the road network was imported into a PostgreSQL[1] database that was stored on the personal computer. PostgreSQL is an open-source relational database which provides spatial data support with the PostGIS[2] extension. The road network was initially provided by *swisstopo* as *ArcGIS*[3] *geodatabase* and imported into the database using QGIS's[4] PostGIS support. The Database GUI DBeaver[5] was used to interact with the database.

3.2 Preprocessing

As mentioned above, this thesis compares three different deep learning architectures. Each of them has other requirements in terms of the input data structure. GCNNs require graph structures (i.e. a tensor that states the neighbourhood relations), whereas Auto Encoders expect n-D images,

[1] https://www.postgresql.org/about/
[2] http://postgis.net
[3] https://www.esri.com/en-us/arcgis/products/arcgis-pro/overview
[4] https://qgis.org/en/site/about/index.html
[5] https://dbeaver.io/about/

and RNNs need sequential data, similar to sentences. The data preprocessing for the different architectures is described in the following sections.

3.2.1 Preprocessing for GCNN

Conceptualisation of Graph A road network can be conceptualised as a graph in different ways. One method would be to conceptualise roads as edges and crossroads as nodes. Zheng et al. (2021) used this approach to tackle the road network selection problem. Another method is to treat the roads' vertices as nodes and the segments in between as edges. This leads to a much higher spatial resolution of the graph and, arguably, to get the level of detail needed to account for the spatial deformation of roads in the course of being generalised.

Sampling Strategy A neural network trained on road generalisation must be able to generalise different road networks. Thus, it must be trained using different road networks and split into subsets. The sampling strategy could be grid-based or object-based. A grid-based strategy implies cutting the network into subnetworks in a grid-like fashion. The disadvantage of this approach is that roads are cut at meaningless positions and disconnected road segments could be selected alongside otherwise connected subgraphs. An object-based strategy defines a subgraph's spatial extent as its roads' spatial extent, thereby respecting the individual lengths and shapes of roads. For this reason, an object-based sampling strategy was chosen.
The next step was to decide how many connected roads were included in a subgraph. The GCNN, as described in more detail in Section 3.3.1, aggregates the neighbouring nodes' attributes onto each node. A relevant hyper-parameter concerning this step is the number of aggregation steps (or the "receptive field" respectively, see more in detail in Section 2.2.2). To save resources in computational time, only models with 5 and 10 layers have been trained and tested. Therefore, at maximum, each node gets the information from nodes that are 10 steps apart. Thus, a sampling strategy that creates subgraphs containing many roads wouldn't have made much sense because many nodes would have been too far apart to benefit from each other's information. At first, a sampling strategy that includes one focal road and its neighbouring roads in the subgraphs was chosen. Later, to facilitate the learning and evaluation process, the adjacent roads were removed, and only the focal roads were kept.

Expected Data Structure As explained in Section 3.3.1 in more detail, the GCNNs were implemented using the Python framework *pytorch geometric*. Its GCNN implementation expects the input to consist of two 2-D tensors: A node feature matrix and an adjacency matrix. The node feature matrix must have the shape $(N_V * N_A)$ with the number of nodes N_V and the number of features N_A. The adjacency matrix must have the shape $(2 * N_E)$ with the number of edges N_E, where each edge is specified by the two nodes it connects. The numbers of nodes and edges depend on the training data set's size and the graph's connectivity. As the graphs in this thesis' case consist of one line of consecutive nodes, the adjacency matrix has the shape $2 \times N_V - 1$. The feature dimension holds information about the nodes' attributes.

3.2.2 Preprocessing for CAE and LSTM

Sampling Strategy For the Auto Encoder and the LSTM, the same object-based and single-road-based sampling strategy as for the GCNN was chosen.

Expected Data Structure for CAE U-Nets and other CAEs, or CNNs in general, are adapted to process images or image-like data. As a result, the input must be formatted accordingly. The expected shape is $((N_B*)C * H * W)$ with the batch size N_B, the number of channels C, the height H, and the width W. The initial vector data was considered to be 2D: One dimension being the number of vertices N_V and one dimension being the number of the vertices' attributes N_A. There are two ways to treat the 2D vector information as a 3D image. One would be to have an "image" with C = 1, H = N_V, and W = N_A, or to have an "image" with C = N_A, H = N_V, and W = 1. A visualisation of these two options is provided in Figure 3.1. In images, spatial data (or pixels, respectively) is usually given in the dimensions H and W, and different sorts of information (e.g. red, green, and blue values) are given in dimension C. For this reason, the latter option was chosen (the right option on the right hand side in Figure 3.1).

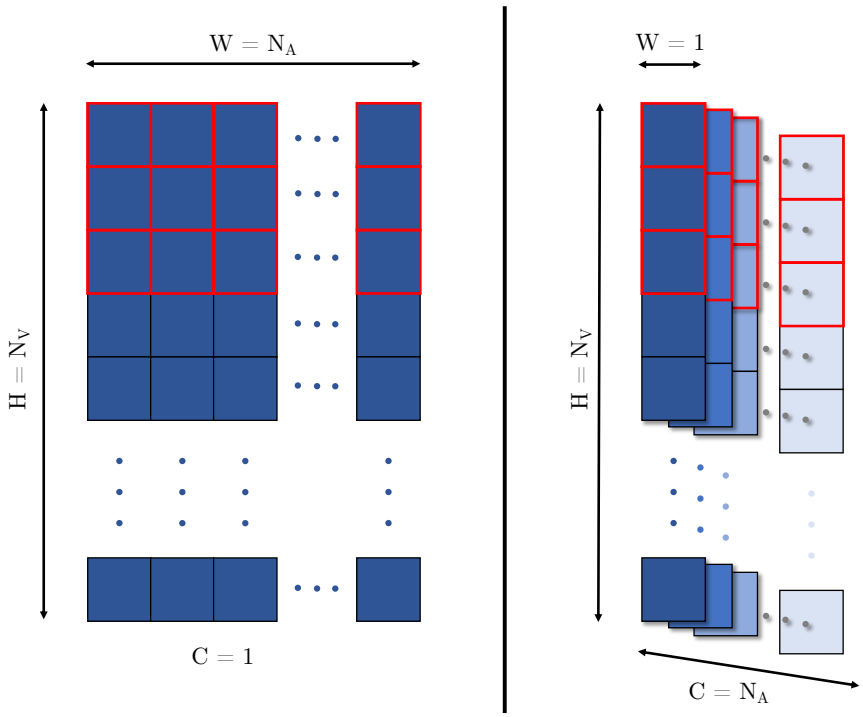


Figure 3.1: Two image-like representations of the road data with convolution kernels depicted in red

Expected Data Structure for LSTM LSTMs are optimised for processing sequential data (Yu et al., 2019), not unlike sentences in natural language. An input tensor for LSTMs has to be of shape $((N_B*)\ L * H_{in})$ with the batch size N_B, the sequence length L, and the input size H_{in}. To carry on with the natural-language analogy: A sentence is a sequence of L words, while a road is a sequence of N_V vertices. A word is described by the H_{in} letters it contains, while a vertex is described by its N_A attributes. Thus, the input has the shape $((N_B*)\ N_V * N_A)$.

3.2.3 Workflow

A visualisation of the preprocessing workflow is depicted in Figure 3.2. Many preprocessing-related steps were the same for all the three architectures. They only began to differ slightly after step 6. Therefore, in the following section, the working steps common to all three architectures are described first, moving to the individual differences as of Step 7. Some of these differences were caused by the different workings of the architectures. However, some were a result of a gradual progression over time.

1. **SQL - Preprocessing**
   As mentioned in Section 3.1, some filtering was conducted using the PostgreSQL database. Due to the selection in the generalisation process, the DKMs consist of fewer roads the smaller the scale gets. Additionally, as part of *swisstopo*'s workflow, sometimes multiple road entities get joined into a single road entity during the generalisation process. As a result, the transition from KRM25 to DKM100 leads to a reduction to about one-third of the roads (*2'011'819* → *651'228*). Therefore, in a first step, only KRM25 roads that point to a DKM100 UUID were selected. In a second step, KRM25 roads that point to the same DKM100 UUID, got joined using the union operator. After the preprocessing using SQL, *643'523* roads remained and were imported into Geopandas[6] GeoDataFrames using *Jupyter Notebooks*.

2. **Omit Multipart Linestrings**
   Some roads of the KRM25 are stored as multipart line strings, which can't be described by a single array of coordinates, but instead by an array of arrays. To compare them to the DKM100 (which consists of solely single-part linestrings), both representations had to be stored in the same way. One possibility would have been to convert the multipart line strings to single part by stringing the individual sequences together. Unfortunately, the individual

[6] `https://geopandas.org/en/stable/about.html`

linestrings are not spatially ordered (i.e. the sequence could "hop" from one ending vertex to the next starting vertex). As a result, multipart linestrings were omitted, which lead to a further reduction to *635'131* roads.

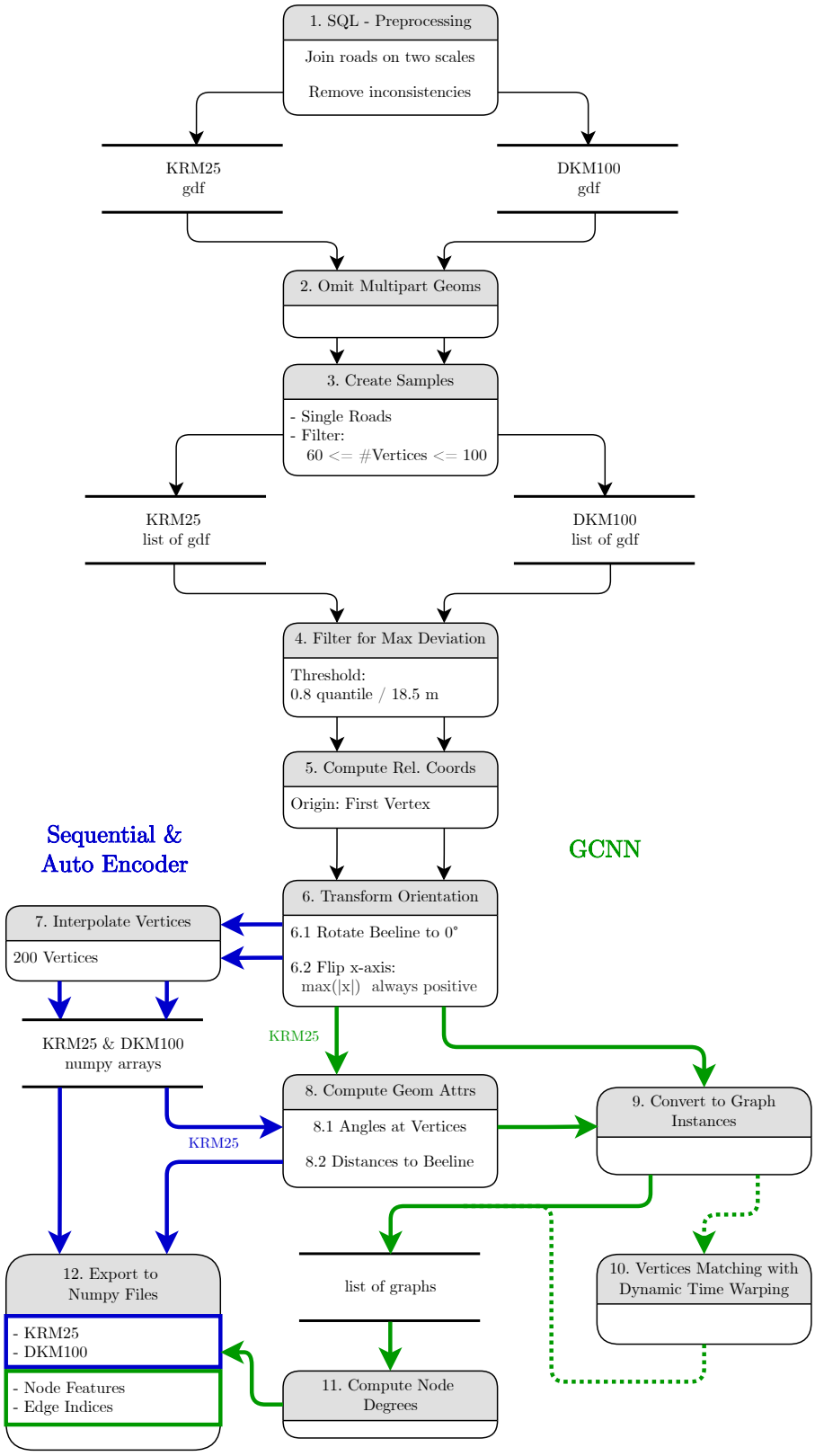


Figure 3.2: Flowchart of the preprocessing workflow

3. **Create Samples**
   As already described in Section 3.2.1, an object-based sampling strategy using single roads was chosen. In addition to breaking up the Swiss road network into single roads, a filtering step according to the roads' number of vertices N_V was conducted. Due to firstly, *swisstopo*'s workflow, leading to a new road entity as soon as one of many attributes changes, and secondly, very short road segments in urban areas, numerous roads consist of very few vertices. A result of these very short roads is that their proposed generalisation of *swisstopo* does not seem meaningful without context, which could be given by adjacent roads that may have been deselected or other map features like buildings or rivers that caused their displacement. A visual impression of these very short roads is given in Figure 3.4. As a result, too short roads were filtered out. The threshold was defined using visual heuristics with the help of a histogram of the N_V's. In step 7 (concerning only the preprocessing for Auto Encoder and LSTM), each road was interpolated onto having a fixed number of vertices $N_V' = 200$. For this reason, such that each road could still be adequately described by its vertices, an upper filter limit of $N_V = 100$ was chosen. After this step *13'366* roads remained.

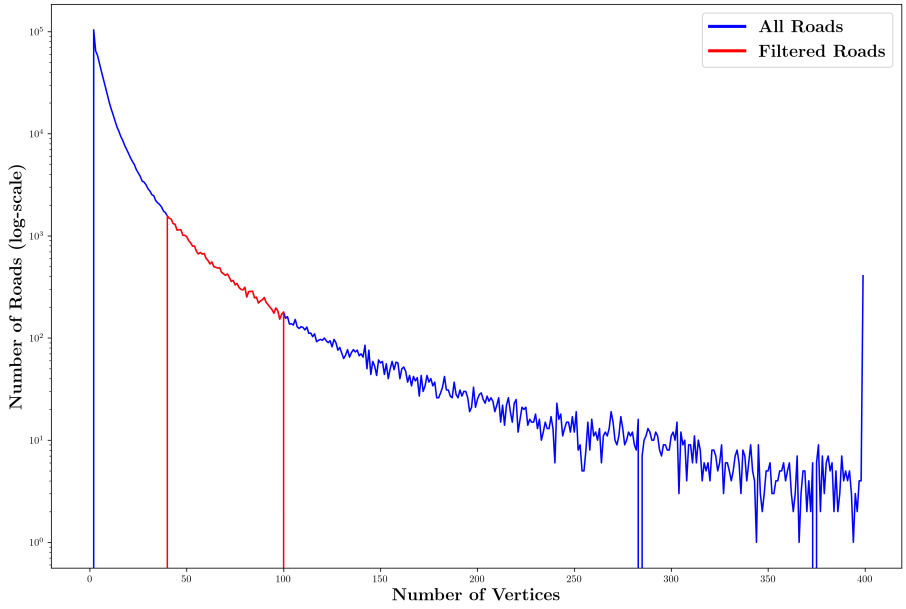


Figure 3.3: Histogram of the number of vertices

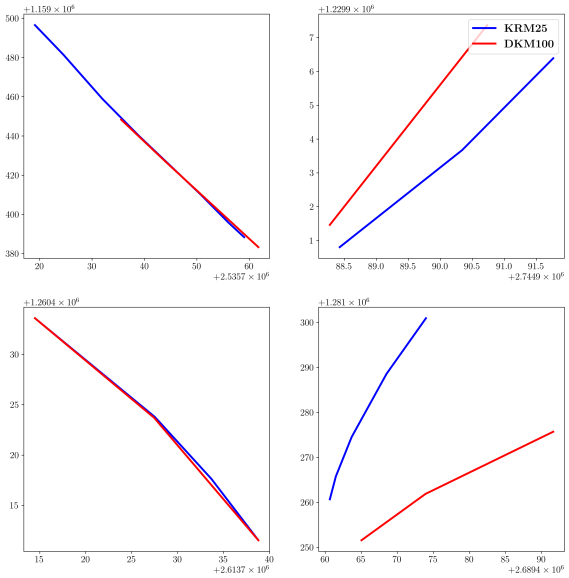


Figure 3.4: Four examples of short roads, x- and y-coordinates are given in the swiss projected coordinate system *CH1903+/LV95*

4. **Filter for Max Deviation**
   Some roads are characterised by a large deviation between the two scales' starting or ending points. The reasons for this are similar to the problem with short roads in the previous step: A lack of context. As a result, roads whose starting or ending points show too large a distance were filtered out. Again, the threshold was defined using a histogram and visual heuristics. The visual judgement resulted in a threshold of about 18.5 meters, which reflects the 20% percentile. The histogram and an example of a road characterised by a large deviation are shown in Figure 3.5. *10'563* roads remained after this filter being applied.

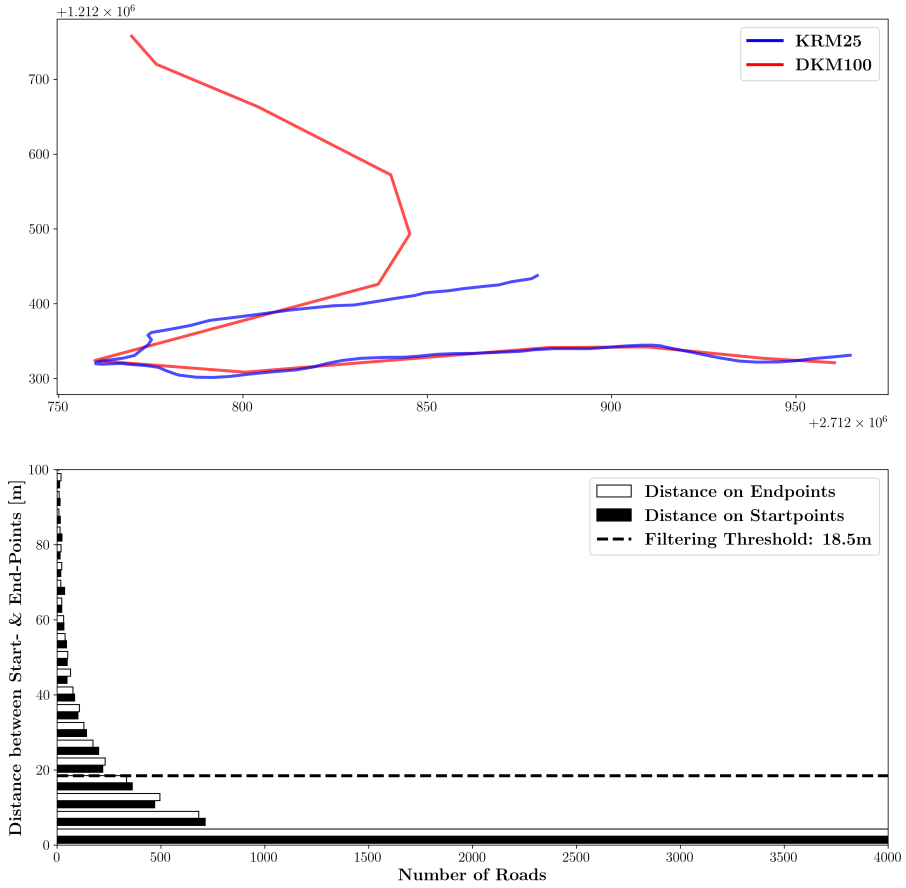


Figure 3.5: Top: Example of a road with a large deviation, Bottom: Histogram of deviations, axes are cut to improve legibility (max(y) $\approx$ 2367 m, max(x) = 10'402 roads)

5. **Compute Relative Coordinates**
   The models should be able to operate independently of the roads' absolute coordinates. Therefore, each road's coordinates were converted into a local coordinate system with the origin being the coordinates of the starting vertex, y being the South-North axis and x being the West-East axis. The absolute origin of each road was stored such that it could be converted to its original coordinates.

6. **Transform Orientation**
   As a further step to make the roads more general and invariant of orientation and polarity, the roads were transformed in two ways:

   6.1. **Rotate Beeline to 0°**: With this transformation, the roads were rotated such that their starting and ending points were on the South-North axis.

   6.2. **Flip X-Axis**: This transformation caused that the largest x-axis deviation of the roads in relation to the Beeline was always east. If this was not the case, the roads were mirrored on the line $x = 0$.

   Three examples of the transformation are depicted in Figure 3.6. The resulting rotation angle and a Boolean variable stating whether the road has been flipped were stored to allow the retrieval of the original geometry.

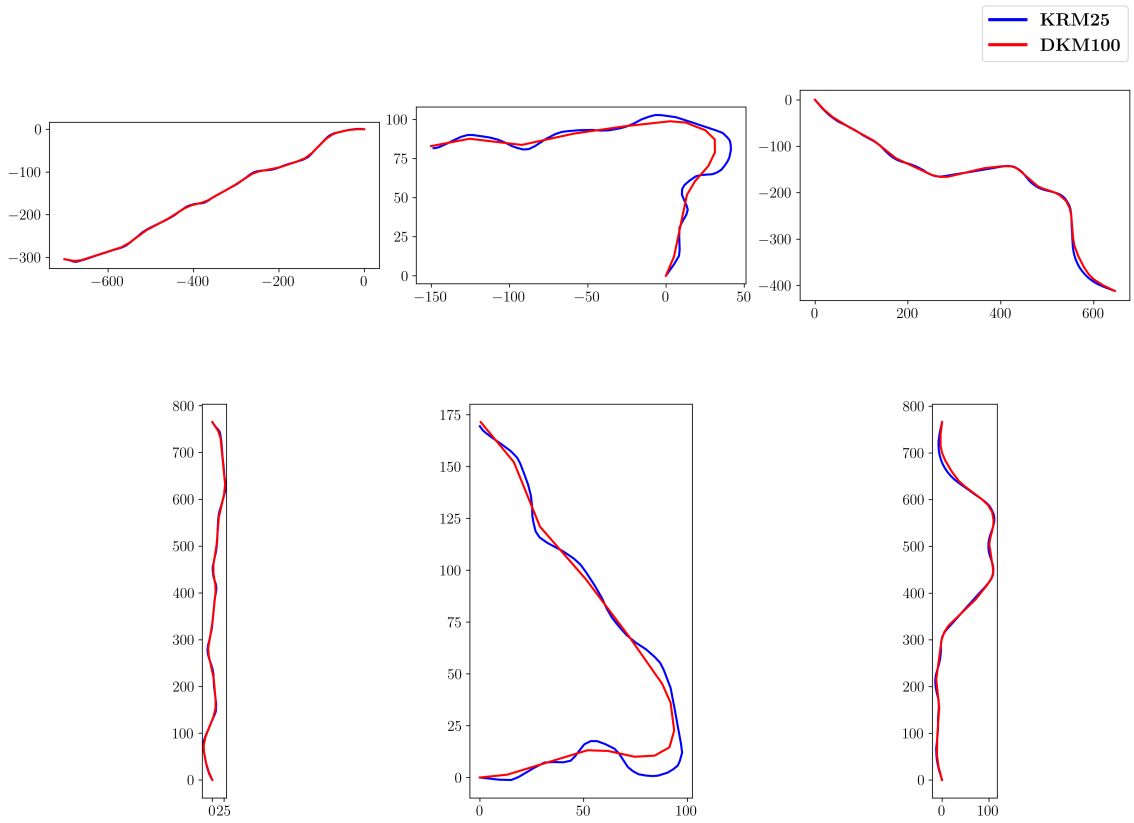


Figure 3.6: 3 examples of transformation → Top: Before Transformation, Bottom: After Transformation

7. **(Only Auto Encoder and LSTM) Interpolate Vertices**
   The chosen loss function *mean squared error* (see more in Section 3.3) compares two lists of values by computing the mean squared difference between all of the corresponding values. In the case at hand in this thesis, it compares the predicted coordinates to the ground truth coordinates of the generalised roads. This means, that the predicted road and the target road must have the same number of vertices. However, the generalised roads can be described by fewer points than the large-scale roads (because they have less detail to describe) and thus they have fewer points to save storage. Two different strategies were used to counter this problem. As already touched upon in step 3, for the Auto Encoder and the LSTM, all the roads, KRM25 and DKM100, were interpolated by exactly 200 vertices. Although the LSTM needed to have a predefined sequence length, this could have been solved more elegantly for the Auto Encoder. The U-Net architecture halves the W and H dimensions in each layer. Therefore, each road could have been interpolated by the nearest power of two. In this thesis, however, this was not implemented. The strategy used for the GCNN is described in Step 9.

8. **Compute Geometric Attributes**
   To enrich the roads' data, two additional geometric attributes were computed:

   8.1. **Angles at Vertices**: If each vertex b of the roads is considered to have a leading vertex a and a trailing vertex c, the intermediate angle ϕ of the segments $\overline{ab}$ and $\overline{bc}$ was computed for each vertex and added to the vertex attributes. This feature corresponds to the local structure in the work of of Yan et al. (2020b) (as described in Section 2.2.2).

   8.2. **Distance to Beeline**: Traditional line simplification algorithms such as the Douglas-Peucker-Algorithm (Douglas and Peucker, 1973) have shown that the distance to a line's beeline (i.e. the direct connection between a line's starting and ending vertices) provides valuable information about the importance of a vertex for the overall shape of a line. For this reason, each vertex's distance to the road's beeline was computed and added to the vertex attributes. This feature embeds each vertex into a global context.

9. **(Only GCNN) Vertices Matching with Dynamic Time Warping**
   The matching of the vertices in the GCNN's case was conducted using *dynamic time warp-*

ing (DTW), which is a technique to find an optimal alignment between two given (time-dependent) sequences (Müller, 2007). A by-product of the algorithm, as it is implemented in the Python library *fastdtw*[7], is the optimal matching of any vertices of the given two sequences, even if the sequences are of different length. DTW was used to create a list of the existing target small-scale coordinates for each large-scale vertex.

10. **(Only GCNN) Convert to Graph Instances**
    To be in control of the topological mapping of the roads (that was, however, more important when at first, the approach using multiple roads per sample was pursued), own *Road* and *Graph* Python classes were created. Each road (-network) thereby was characterised by its vertices (nodes) and its segments (edges).

11. **(Only GCNN) Compute Node Degrees**
    The *Graph* instances have a method that lets them compute the node degrees (= the number of neighbouring nodes) for each of their vertices. Again, this was more important when the multiple-road approach was pursued. However, in the single road approach, the node degrees serve as a "border-vertex / within-vertex" classification, that could help the models in their task.

12. **Export to NumPy Files**
    In the final preprocessing step, the roads were split into a training (80%), validation (10%), and testing set (10%), which led to a training size of 8450, and a validation and testing size of 1056. The sets were the same for all three architectures to enable a consistent comparison. Next, the roads were stored as NumPy[8] binaries on the computer's disk. For the GCNN, the roads were saved as individual NumPy files, with a feature matrix file and an adjacency matrix file per road. For the other architectures, two files in total were saved: One for the KRM25 roads and one for the DKM100 roads.

3.3 Implementation of Models and Training

All the models in this thesis were created using the Deep Learning library "PyTorch" (Paszke et al., 2019). It provides the framework to build all kinds of neural networks and to conduct forward pass, automatic differentiation, and back-propagation. Although most (convolutional) operators are already implemented, the following hyper-parameters had to be specified (at least): The configuration of (convolutional) layers, including the input- and output dimensions for each layer, the loss function, the optimisation algorithm and its learning rate, the batch size, and the number of epochs. The number of layers (i.e. the "deepness" of the network) and their size influence the learning behaviour and learning capability. This configuration controls, how the features can be combined and abstracted (Lecun et al., 2015). However, too many or too large layers may cause problems like the vanishing gradient problem or over-fitting (Li et al., 2019). The loss function assesses the model's performance by comparing the predicted value $\hat{y}$ with the ground truth value y, and returns a value that describes the difference between the two. The lower the value, the better the model. With automatic differentiation, the model adjusts the weight of all its parameters so that the overall loss function is smaller in the next iteration. In PyTorch and PyTorch Geometric, a loss function is an object that also calculates the gradient for each parameter of the model by automatic differentiation. The optimisation algorithm's target is to minimise the loss by adjusting each parameter according to its computed gradient. The learning rate scales the step size of these adjustments per iteration. The batch size is the number of samples considered for the gradients and the parameters to be updated. A small batch size leads to a smoother loss decline but may result in longer training time (Golmant et al., 2018). The number of epochs is the number of complete passes through the training dataset.

The overarching functionality of all the models was to take as input single roads, described by their coordinates and other (geometric) attributes, and output the coordinates of the predicted generalised roads. Each predicted road / mini-batch was compared to its target (mini-batch), and the difference was assessed using the loss function. The architectures thus only differ in their required inputs and their inner workings. In all of the architectures, the loss function *Mean Square Error* (MSE) was used, which is the common loss function when dealing with regressions (Jadon et al., 2022). To update the parameters, the optimisation algorithm *Adam* was used, which is one

[7] `https://github.com/slaypni/fastdtw`
[8] `https://numpy.org/about/`

of the most widely used optimisation algorithms in deep learning (Soydaner, 2020). As *Adam* is an adaptive optimiser, only the initial learning rate had to be specified, which is then adapted automatically for all the weights. This initial learning rate differed between the architectures. A batch size of 16 for the CAEs and the RNNs, and 32 for the GCNNs was used. These sizes emerged after a few tests and seemed like a good trade-off between batch mode (i.e. updating the weights after one entire epoch) and stochastic mode (i.e. updating the parameter after each road). The number of epochs differed between the architectures and is described in Section 4.3 along with samples of the resulting loss curves. Generally, models were trained until no further meaningful decrease or even an increase in the validation loss was registered, which implies over-fitting.

Training on the Science Cluster The Science Cluster of the UZH Science IT facilitated the training of the models. It provides powerful virtual machines that are equipped with GPUs. Thus, much of the training could be conducted on these virtual machines. However, because of an erratic error that occurred when trying to load trained GCNNs back on the personal computer, all GCNNs were trained on the personal computer using only the CPU, which made the training of single models take up to 24 hours.

3.3.1 GCNN

Implementation The GCNN was implemented using the Deep Learning library "PyTorch Geometric" (Fey and Lenssen, 2019) that is built on top of PyTorch. It is specialised for Deep Learning tasks that include graph-structured data. Because the graph convolutional operator is fully implemented within PyTorch Geometric, only the configuration of layers and convolutional operators had to be developed. The basic structure is shown in Figure 3.7 and consists of the following three elements:

1. **Conversion from Feature Space to Embedding Space**
   This step enlarges the feature dimension into a defined size N'_A, which is a hyper-parameter of the model.

2. **Defined number of message-passing steps followed by sequential layers**
   This element is the core of the model. As described in Section 2.2, GCNNs are defined by a number of message-passing steps with subsequent fully connected layers. This number N_L is a hyper-parameter of the model and serves as a measure of complexity and simultaneously as a "receptive field".

3. **Conversion from the embedding space to the x and y coordinates**
   After each vertex has aggregated and transformed its information, its feature space has to be reduced to the x- and y-values of the coordinates, which are the model's output.

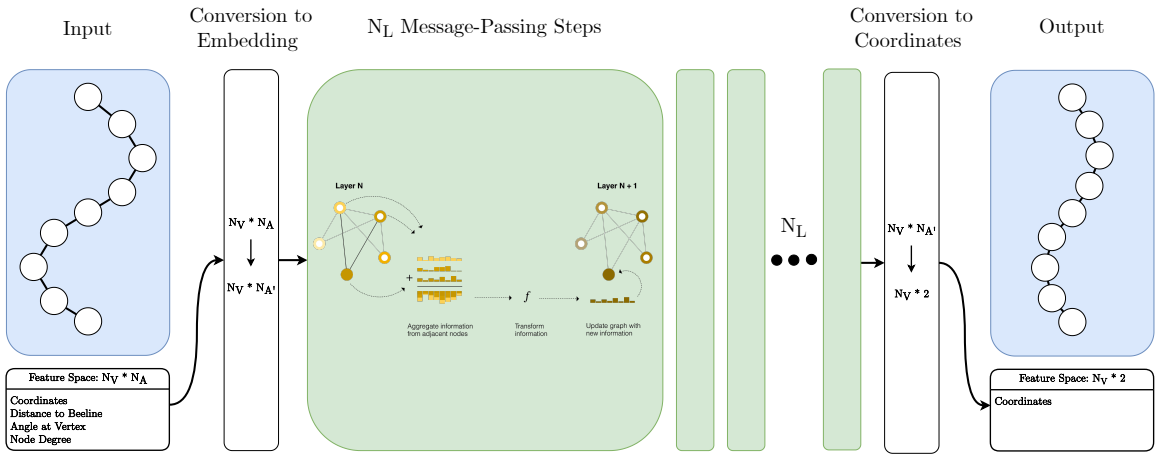


Figure 3.7: GCNN model Structure

Hyper-Parameter Selection The two tunable hyper-parameters N'_A and N_L were compared systematically, meaning that all possible combinations of different settings for those were examined. For N'_A, the values 32, 64, 128, and 256 were used. For N_L, the values 5 and 10 were used. After the identification of the best combination, the effect of a *Fourier Loss Extension* (as described in Section 3.3.4) was examined. The evaluation of all the hyper-parameter configurations is described in Section 3.4.1.

3.3.2 Auto Encoder

Implementation As already touched upon in Section 3.2.2, inputs of Convolutional Auto Encoders have to be image-like. However, the feature-extracting capabilities of CNNs can also be used on sequential (vector) data (Goodfellow et al., 2016). Due to the implementation of CNNs in PyTorch, inputs must be 3D. However, one dimension (the channel-, or one of the spatial dimensions) can be left with length 1. As also mentioned before, the option that collapses the width dimension was chosen. This leads to input tensors of shape $(C * H * W) = (N_V * N_A * 1) = (4 * 200 * 1)$ without the batch dimension, which could be considered a multidimensional 1D-Image. As depicted in Figure 3.1, the kernel sizes were adjusted accordingly: Instead of traditional $(3 * 3)$ kernels, $(3 * 1)$ kernels were used. As described in Section 2.2, auto encoders are said to be unsupervised learning techniques (Sarker, 2021). However, in practice, this is expressed by comparing the predicted output with the input again, instead of comparing it with a target. From this follows that the generalising abilities of CAEs could arguably be used in a supervised setting with specific targets by comparing the output with a target. In total, three CAE architectures were used:

- **Simple Convolutional Auto Encoder**
  This architecture is an auto encoder in the strict sense because it does not have any skip connections, thereby forcing it to store all acquired information in the latent space h. It consists of four convolutional steps and three fully connected layers in both the encoding and decoding steps. The size of the latent space is a hyper-parameter. The structure is depicted in Figure 3.8.
- **Traditional U-Net**
  The implementation of the U-Net architecture was adopted from a *GitHub* repository[9]. Its rough structure is depicted in Figure 2.3: The U-Net consists of an encoder and a decoder, which both consist of 4 convolutional blocks.
- **ResUNet**
  The ResUNet, as it is implemented in this thesis, differs from the traditional U-Net in that its convolutional blocks are extended by skip connections. The overall structure, however, is the same.

Hyper-Parameter Selection The only tunable hyper-parameter of the used architectures is the hidden size h in the simple CAE. 3 different sizes were compared: 64, 128, and 256. Additionally, the effect of the *Fourier Loss Extension* was examined.

3.3.3 RNN

Implementation RNNs are susceptible to the vanishing gradient and long-distance dependency problem, especially when they are to deal with long sequences (for more detail, see Section 2.2). The model inputs, consisting of sequences with a length of 200, are really long. Since the emergence of LSTMs tackled exactly these two problems, the LSTM architecture was used. The implementation using PyTorch was straightforward, as LSTMs are already fully implemented. The relevant hyper-parameters are 1. whether the LSTM should be bidirectional, 2. the number of LSTM layers N_L, and 3. the size of the hidden layers N'_A. As depicted in Figure 2.4, plain RNNs are unidirectional, meaning that a predecessor vertex is unable to gain any information about its successors. To solve this problem, the idea of bidirectional LSTMs implies running two LSTM layers in opposite directions. Besides the parameter selection, other decisions had to be made. After a complete forward pass of the LSTM, an initial tensor of size $((N_B*)\ N_V * N'_A)$ is output. An exemplary initial output y_{LSTM} size would thus be $((16*)\ 200*128)$ with $N_B = 16$, $N_V = 200$, and $N'_A = 128$. This example is used to further illustrate the model. To get to the desired shape of $((16*)\ 200 * 2)$, different strategies to collapse y_{LSTM} were tested:

[9] `https://amaarora.github.io/2020/09/13/unet.html`

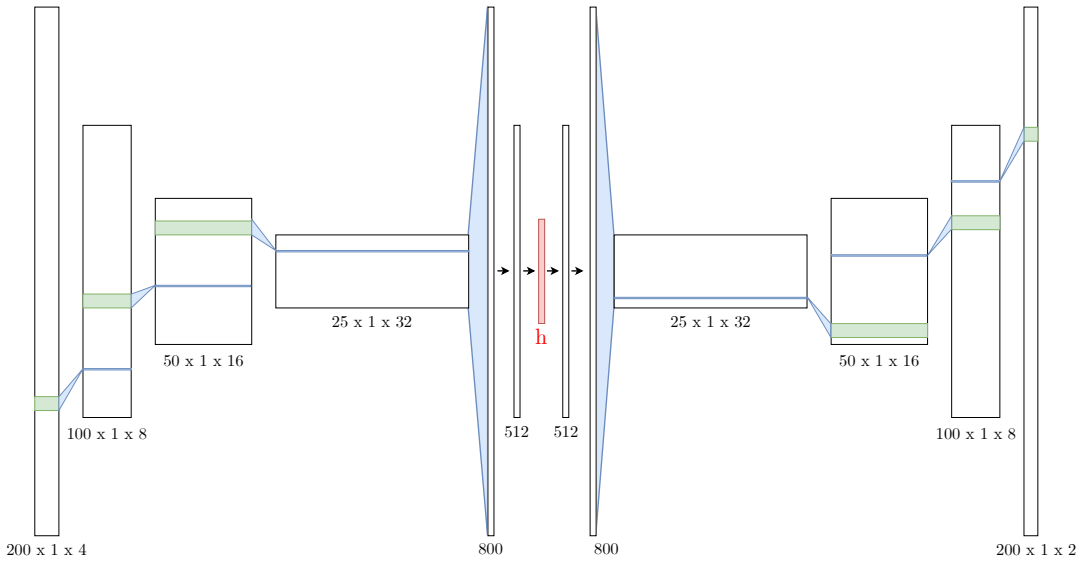


Figure 3.8: Structure of the Simple Convolutional Auto Encoder

1. **Take only the last step**
   The last step of the sequence aggregates all information about the whole sequence. Then, this tensor of size ((16∗) 1∗128) is projected onto the desired output using a fully connected layer with subsequent "unflattening" of the resulting layer onto the desired output. This method is expected to be negatively affected by symmetry effects when dealing with unidirectional LSTMs.

2. **Vertex-Based: Project N'_A onto size 2 in each step**
   Instead of taking the last step and re-projecting only the aggregated information onto the vertices, the hidden size N'_A is reduced to 2 in each time step (128 → 2 in the example). As with the previous collapsing strategy, this method is expected to be negatively affected by symmetry effects in unidirectional settings.

3. **Use all steps**
   The third and most holistic strategy is to flatten y_{LSTM} onto a tensor of size ((${N_B}*$) 25′600) with 25'600 being the product of N_V and N'_A with a subsequent fully connected layer and unflattening onto the desired output.

Hyper-Parameter Selection Two find the best hyper-parameter combination, the following two-step strategy was chosen: In the first step, a systematic comparison between different combinations of N_L and N'_A was conducted with the values 1, 2, and 3 for N_L and 64, 128, 256, and 512 for N'_A. First, the best combination of these hyper-parameters for models that used the third collapsing strategy, was identified. In a second step, this combination was used to test the other strategies in uni- and bidirectional settings. Additionally, a possible enhancement of the results with the Fourier loss extension (Section 3.3.4) was examined.

3.3.4 Fourier Loss Extension

A line theoretically consists of an infinite number of points. Thus, the roads are only described by a certain number of sample points. These sample points are not spaced evenly across the roads but are spaced such that the road can be described in the needed detail but with as few points as possible to save on storage. Especially the generalised roads consist of very few points. However, it is essential that the generalised line is accurately described by the predicted points, not that the

predicted points are the same as the (to some extent) arbitrarily placed target points. But the MSE loss function optimises for exactly the latter. Therefore, after the hyper-parameter selection, the effect of an extension of the loss function by the *Fourier Transform* was examined. The idea of the Fourier Transform was initially developed to model heat diffusion (Lawford, 2007) and its principle is that every signal can be approximated by a series of simple periodic functions (sine and cosine), which is described as the *Fourier Series*. During the Fourier Transformation, a signal is broken down into the individual frequencies that make it up and described by a series of parameters for the simple periodic functions (Lawford, 2007), thereby transforming it into the *Fourier Domain*. This transformation can be inverted by the *Inverse Fourier Transformation*, leading to the input signal again. A line (or road), that is described by a set of coordinates, can be transformed into the *Fourier Domain* too. In the Fourier Domain, the road's shape is then described holistically and continuously. The Fourier Loss Extension works by transforming the predicted roads and the target road into the Fourier Domain before comparing them using the MSE. Thereby, the parameters for the simple periodic functions are compared instead of the discrete points. Besides providing a more holistic representation of the roads, the Fourier Loss Extension could penalise an occurrence of too high or too low frequency spectra in the predicted roads.

3.4 Evaluation

The performance of the models is evaluated quantitatively and qualitatively. The evaluation strategy is described in the following two sections.

3.4.1 Quantitative Evaluation

The quantitative evaluation is conducted using two different metrics that both describe the similarity (or difference, respectively) between two lines. Both metrics are implemented in the Python package *similaritymeasures*[10].

- **Fréchet Distance FD**
  "An intuitive definition of the Fréchet distance is to imagine that a dog and its handler are walking on their respective curves. Both can control their speed but can only go forward. The Fréchet distance of these two curves is the minimal length of any leash necessary for the dog and the handler to move from the starting points of the two curves to their respective endpoints." (Aronov et al., 2006, p. 1) This intuitive definition, however, assumes two continuous lines. But as mentioned above, the lines in this thesis are approximated by discrete points. As a result, not the true FD, but the discrete FD was computed, which deviates most from the true DF by the length of the longest edge along the polygonal curves (Jekel et al., 2019).

  The true FD can formally be described as (Eiter and Mannila, 1994):

  > We define a curve as a continuous mapping $f : [a, b] \to V$, where $a, b \in R$ and $a \leq b$ and (V, d) is a metric space. Given two curves $f : [a, b] \to V$ and $g : [a', b'] \to V$, their Fréchet distance is defined as
  >
  > $$\delta(f, g) = \inf_{\alpha,\beta} \max_{t\in[0,1]} \left\{ d\Big(f(\alpha(t)),\, g(\beta(t))\Big) \right\}$$
  >
  > where α (resp. β) is an arbitrary continuous non-decreasing function from $[0, 1]$ onto $[a, b]$ (resp. $[a', b']$).

  This metric is an aggregating function that is highly susceptible to local events. Thus, if there is one significant deviation between the compared roads in one place, this deviation sets the Fréchet Distance, and the similarity of the rest of the road does not have any influence. Nevertheless, it provides a natural and intuitive measure for computing the similarity of two (polygonal) curves (Bringmann et al., 2019).

- **Area between Curves ABC**
  This metric describes the area spanned between two lines and, as such, is a global measure that considers all portions of the roads equally. As shown in Figure 3.9, it has a moderate

[10] `https://github.com/cjekel/similarity_measures`

positive correlation with the length of the roads (Pearson-Coefficient ≈ 0.62 with a p-value ≈ 0). For this reason, the areas were normalised by dividing them by the large-scale roads' lengths, leading to the relative ABC (rABC). The ABC was computed according to the work of Jekel et al. (2019). By defining the two curves as an array of consecutive quadrilaterals, the ABC can be calculated as a sum of the areas of these quadrilaterals. For this to work, the two curves must have the same vertices. The chosen interpolation method to ensure this constraint is depicted in Algorithm 1, Lines 12 - 17. The area A of the quadrilaterals can be calculated using the Gauss/Shoelace equation:

$$A = \frac{1}{2}|x_1y_2 + x_2y_3 + x_3y_4 + x_4y_1 - x_2y_1 - x_3y_2 - x_4y_3 - x_1y_4|$$

However, this equation only works for simple (non-self intersecting) quadrilaterals. The method to "simplify" self-intersecting quadrilaterals is depicted in Algorithm 1, Lines 23 - 25.

Algorithm 1 Pseudo code to calculate the area between two curves

```
1: function areaBetweenCurves (curveA, curveB)
2: Input: Data of curveA and curveB.
3: Output: Area between curve and curveB.
4: # the length function returns the number of data points
5: if length(curveA) < length(curveB) then
6:     A = curveA
7:     B = curveB
8: else
9:     B = curveA
10:    A = curveB
11: end if
12: while length(A) < length(B) do
13:    Compute distance between every two consecutive points of A;
14:    Find the two points that generate the max distance;
15:    Create a point that bisects these two points;
16:    Add the bisecting point to A in between the two points;
17: end while
18: n = length(A) - 1; # compute the number of quadrilaterals;
19: areas = zeros(n); # initiate zeros array for areas;
20: for i = 1 to n do
21:    # Assemble quadrilateral;
22:    quad = [A[i], A[i+1], B[i+1], B[i]];
23:    if quad is not simple then
24:        Rearrange the order of vertices until quad is simple;
25:    end if
26:    # Calculate the Gauss/shoelace area of the quadrilateral;
27:    areas[i] = gaussArea(quad);
28: end for
29: # Return the summation of quadrilateral areas; return ∑(areas);
```

Effect of N_V on the Fréchet Distance The discrete FD reacted very strongly to N_V. Although the input and target lines of the GCNN and the other models describe the same roads, they consist of a different number of vertices. Since vertices that do not contribute decisively to the shape of the roads have been removed by swisstopo to save storage space, there are huge distances between vertices in some instances. The FD reacts to these large distances, which do not occur in the interpolated lines. As a result, to keep the architectures comparable, the predicted roads, which consist of the same number of vertices as the input roads, were interpolated by 200 vertices in a post-processing step. By doing so, the described effect should be mitigated.

Evaluation of Performance For each model, the two metrics were computed to show the similarity of all target and predicted roads of the test set. After the computation of the measures, they were aggregated using an averaging function. Whether the mean or the median was used

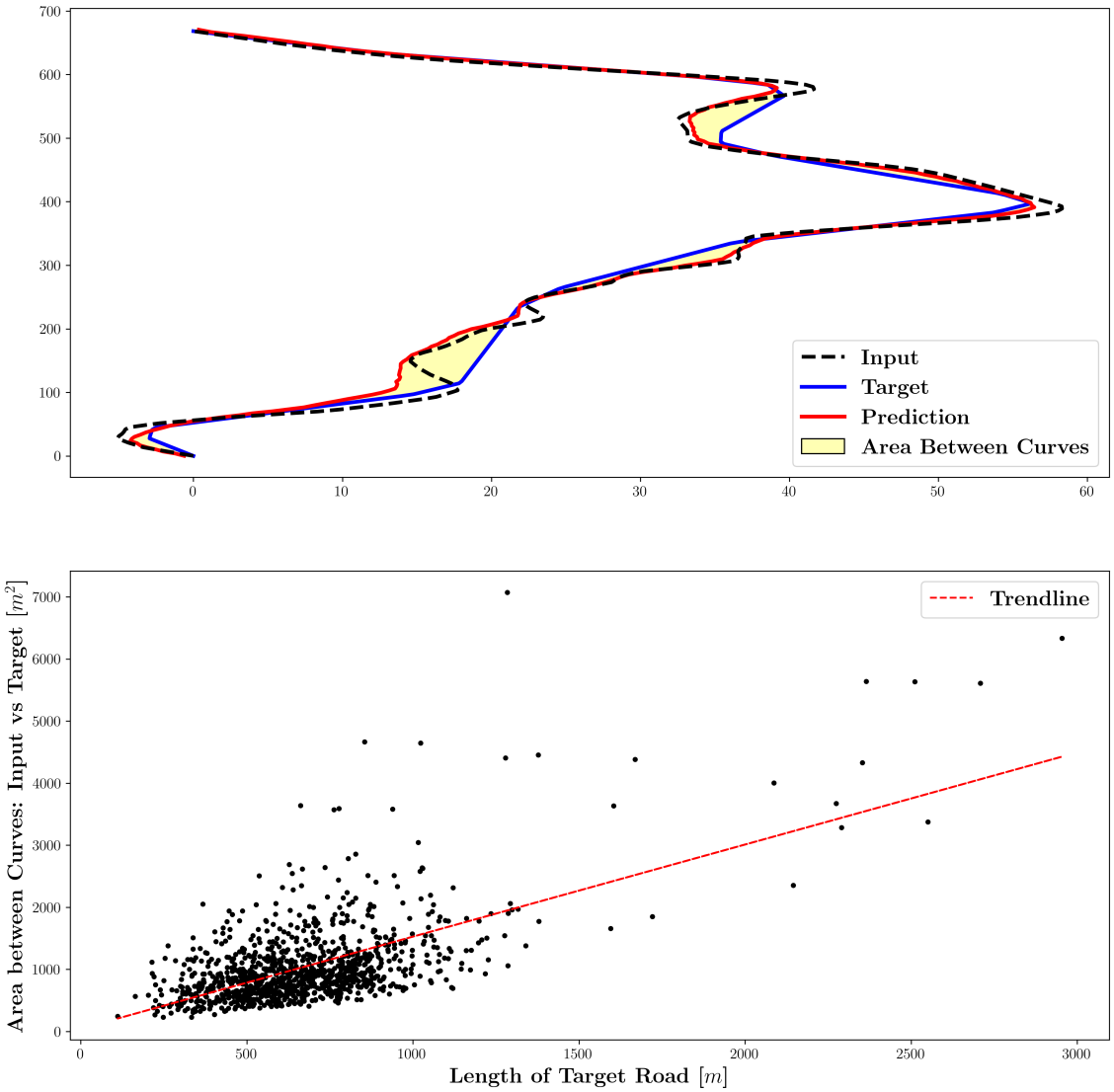


Figure 3.9: Top: Visualisation of the Area between Curves, Bottom: Scatterplot showing the relation between the lengths of the target roads and the Area between the input and the target roads

depended on the distribution of the metrics. If the distributions had looked reasonably normally distributed, the mean would have been used. However, because the distributions were characterised by many outliers and were very skewed, the median was used (more on that in Section 4.1).
The aggregated metrics are used to describe the performance of the models. High values indicate a bad performance, while low values indicate a good performance for both metrics. After identifying and selecting the models with the optimal hyper-parameter configurations within their respective architectures, the resulting aggregated metrics of the best models of the different architectures are compared. If two different models within an architecture have excelled at the two different metrics, both are compared with the other architectures.

Further Analytics For the three different lines per road (i.e. input, target, and prediction), the metrics can be used for three comparisons:

1. **Target ↔ Prediction**
   The difference between the target and the predicted roads, as outlined above, describe a model's *performance*.

2. **Input ↔ Target**
   The difference between the input and the target roads describes how much deformation is needed to transform a road from its large-scale to its small-scale representation. As such,

it can be seen as the *difficulty* for the model to perform its generalisation or the *actual dissimilarity*, respectively.

3. **Input ↔ Prediction**
   The difference between the input and the predicted roads describes how much the model assumes that the large-scale representation of the road has to be deformed to be generalised. This comparison can thus be considered as the *predicted dissimilarity*. In combination with Comparison 1, it can be used to describe the model's ability to judge the amount of deformation needed.

The two additional comparisons were computed for all the models. For the best-performing models within the three architectures then, the correlation between the difficulty and the performance, as well as between the actual dissimilarity and the predicted dissimilarity, were assessed using scatter plots and the *Pearson Correlation Coefficient*.

3.4.2 Qualitative Evaluation

For the qualitative evaluation, the roads were first classified according to their *difficulty* (as described in Section 3.4.1). As the rABC is less susceptible to single deviation events than the FD, the rABC is chosen to quantify the *difficulty*. The classification was done using three quantiles: The lower third quantile includes the "easy" roads, the middle third quantile the "moderate" roads, and the upper third quantile the "difficult" roads. For these three difficulties, random samples were extracted of the three best-performing models of the different architectures and visually evaluated in terms of their ability to perform different generalisation operations, as they are described in Chapter 1 (only the operators depicted in **bold** are used in the evaluation):

- **Simplification** is the removal of vertices in lines, which leads to a reduction of the level of precision and the removal of small bends. The most recognised simplification algorithm is the Douglas-Peucker-Algorithm (Douglas and Peucker, 1973).
- **Smoothing** relates to the removal of high-frequency information in the roads. Smoothing operators thus act as low-pass filters. Often used smoothing algorithms are the Gauss-Filter or the Savitzky-Golay-Filter (Savitzky and Golay, 1964).
- **Exaggeration** implies the deformation of a road to make some aspect of a feature larger than it is, to make it more visible. In road generalisation, it is often-used to emphasise certain bends.
- Displacement operators are applied when two map elements would collide in smaller map scales. As a result, the map elements are moved further apart. Often, there are hierarchical rules regarding which map element classes are moved in case of collisions (Spiess et al., 2003). Unfortunately, this operator can't be considered because it relies on the spatial context of the large-scale roads, which the proposed models do not receive during the training and evaluation process.

Chapter 4

Results

4.1 Quantitative Description

In the following section, the results of the quantitative evaluation of the models are described. In the tables, the models are described with a systematic code. “hs” thereby stands for the hidden size of the model, “l” stands for the number of layers, “fourierloss” stands for a model trained with the Fourier Loss Extension and “coordloss” for being trained without it. The numeric differences between the metrics of the different models is often very small. However, for the sake of simplicity, the models with the lowest metrics were selected for the evaluation of the next hyper-parameter, irrespective of the differences’ statistical significance. The FDs are described using their unit [m] (meters). Although the rABC was initially computed as $[m^2]$ and divided by the length [m] and thus, the resulting unit would be [m], this unit does not intuitively makes sense as it does in the FD’s case. As a result, the rABCs are described unit-less.

4.1.1 GCNN

Removal of the Last Vertex Predicted roads of all of the GCNN models describe a “tick” at the end. The last vertex is always remarkably off the target road. An example of this behaviour is depicted in Figure 4.1. As this “tick” often affects the two computed measures severely, the last vertices of all the predicted roads have been removed for all GCNN models.

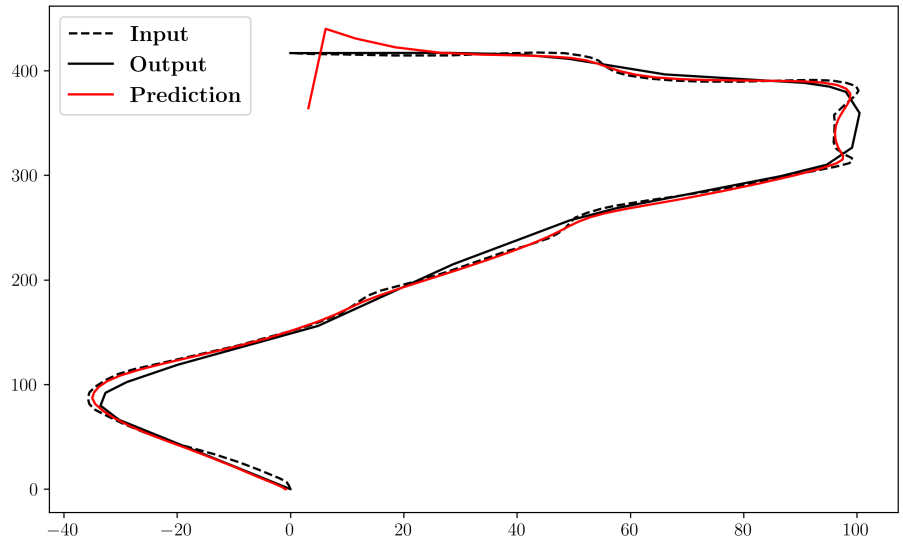


Figure 4.1: Example of a GCNN’s prediction without the removal of the last vertex

Distribution of Metrics As depicted in Figure 4.2, the distributions of both metrics describing the difference between the target and the prediction are strongly skewed and consist of many outliers. Especially when considering the rABC, the deviation between the median and the mean becomes apparent. Thus, to aggregate the computed metrics for the predicted roads, the median

was used. Although the depicted histograms and boxplots only show one exemplary GCNN, similar distributions emerge for all the different hyper-parameter configurations.

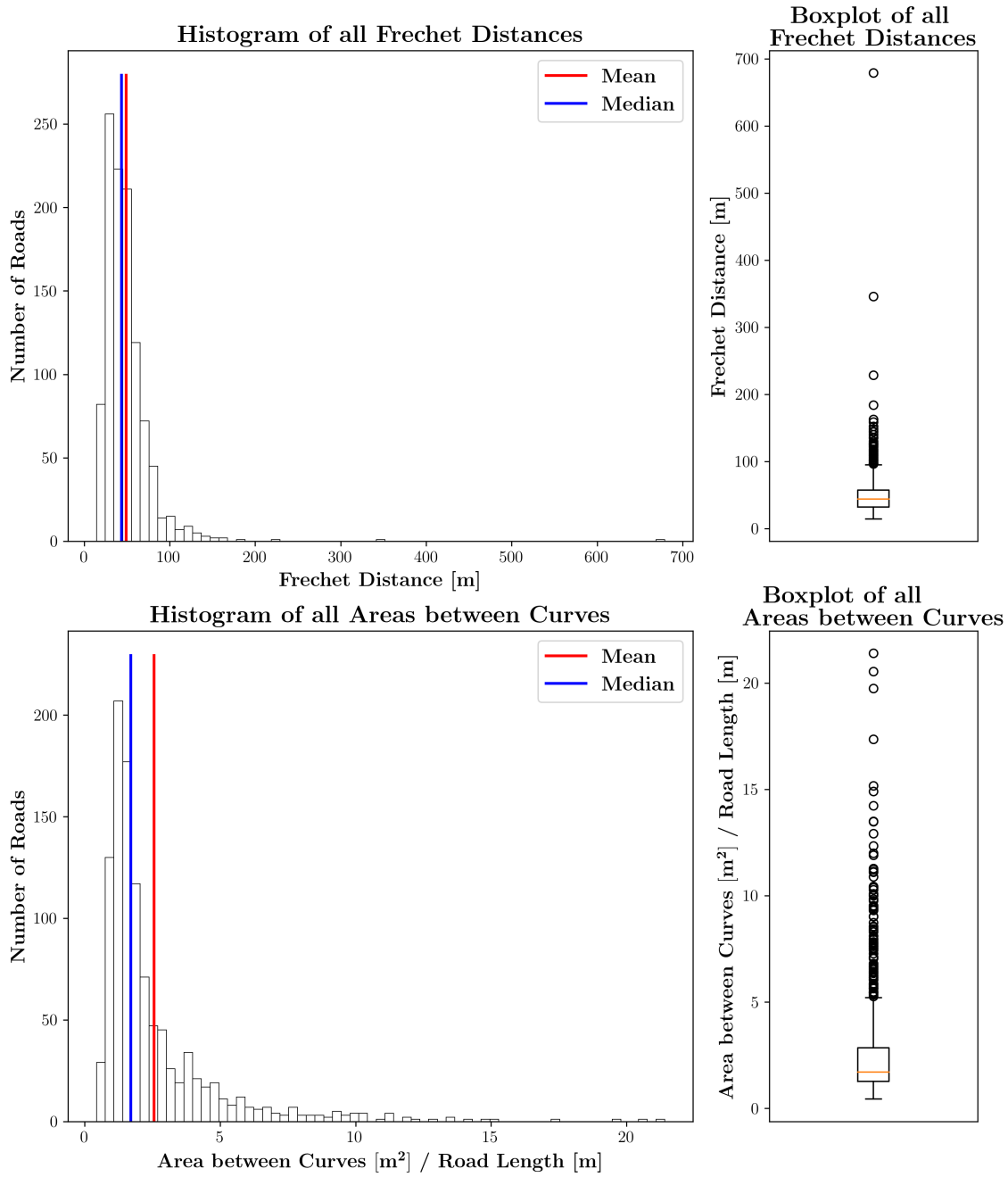


Figure 4.2: Histogram and boxplot of the computed performance metrics of an exemplary GCNN with the hyper-parameter configuration (hidden size: 32, 5 Layers, coordloss)

Description of Aggregated Metrics The median values of the two metrics for the different GCNN models are shown in Table 4.1. The mean median of the FDs of all the models is 22m (sd = 4.78m) and the mean standard deviation is 25.48m (sd = 1.94m). The mean median of the rABCs of all the models is 2.96 (sd = 0.36) and the mean standard deviation is 3.12 (sd = 0.13). The model that best performed in terms of the FD was the GCNN with a hidden size of 256 and 5 message passing layers. The model that best performed in terms of the rABC was the GCNN with a hidden size of 256 and 10 layers. The Fourier Loss Extension didn't improve the performance of these two models (FD: $13.47m \rightarrow 40.98m$, rABC: $2.6 \rightarrow 3.31$). Increased hidden sizes generally produce lower FDs but do not seem to have a consistent effect on the rABCs. The number of layers has a similar effect, with the exception of the model with 10 layers that performed significantly worse than the model with 5 layers regarding the models with a hidden size of 256. The rABC does not reflect this exception.

	Median Fréchet Distance	Median Area between Curves
gcnn_hs32_5layers_coordloss	28.21	2.71
gcnn_hs32_10layers_coordloss	22.93	2.97
gcnn_hs64_5layers_coordloss	27.08	3.03
gcnn_hs64_10layers_coordloss	20.69	2.72
gcnn_hs128_5layers_coordloss	26.5	3.22
gcnn_hs128_10layers_coordloss	19.46	3.83
gcnn_hs256_5layers_coordloss	13.74	2.64
gcnn_hs256_5layers_fourierloss	14.87	2.84
gcnn_hs256_10layers_coordloss	24.95	2.6
gcnn_hs256_10layers_fourierloss	17.84	3.31

Table 4.1: Computed performance metrics of the GCNNs, orange: Best configurations

4.1.2 Convolutional Auto Encoder

Smoothing with the Savitzky-Golay Filter Many predictions of the CAEs are characterised by very noisy lines. An example is depicted in Figure 4.3. While the predictions of the simple CAEs are severely noisy, the predictions of the U-Nets and the ResUNets only show a slight "zig-zag" pattern. To smooth the lines before computing the metrics, a Savitzky-Golay Filter (Savitzky and Golay, 1964) is used. This filter works by fitting low-degree polynomials on successive subsets of the data using a moving window. The relevant parameters are the window size, which refers to the number of data points that the polynomials are fitted on, and the polynomial degree. As the simple CAEs are in need of more smoothing, a large window size of 35 vertices was chosen with a low polynomial degree of 2. Because the U-Net variants should be smoothed less, a smaller window size of 10 vertices was chosen with a larger polynomial degree of 3.

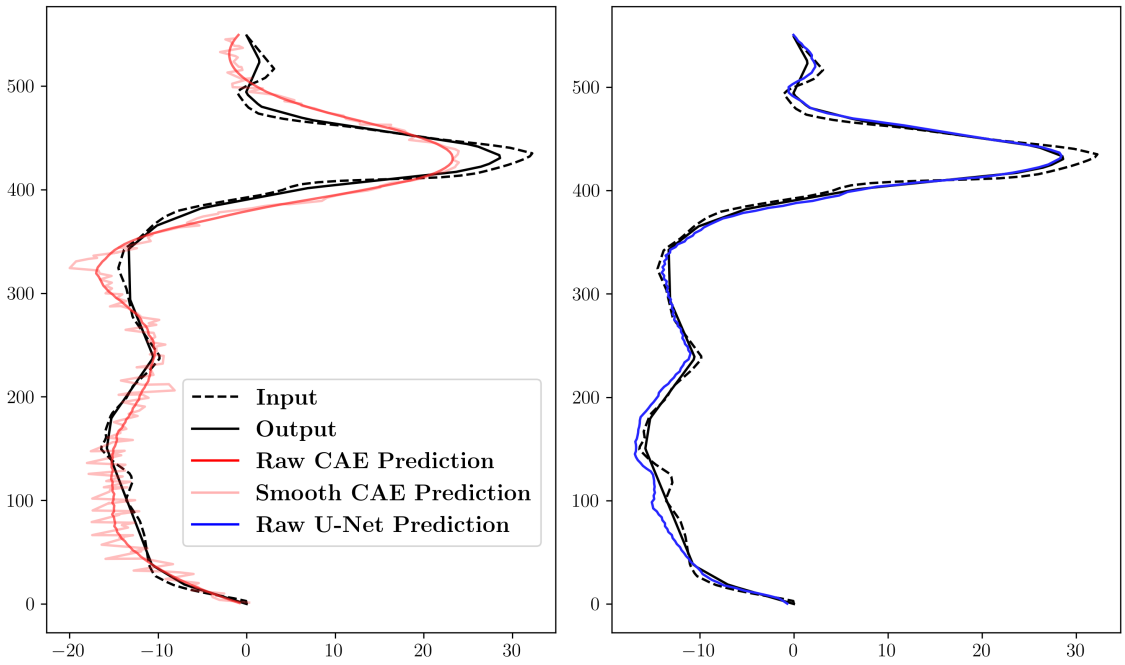


Figure 4.3: Example of noisy and smoothed predictions of a simple CAE (left, window size = 35, polynomial order = 2) and U-Net (right, window size = 10, polynomial order = 3)

Distribution of Metrics As shown in Figure 4.4, the distributions of both metrics describing the difference between the target and the prediction are strongly skewed and consist of many outliers. The mean is considerably higher than the median for both metrics. Thus, again, the median is used as aggregation function. Similar distributions arise for all the CAEs.

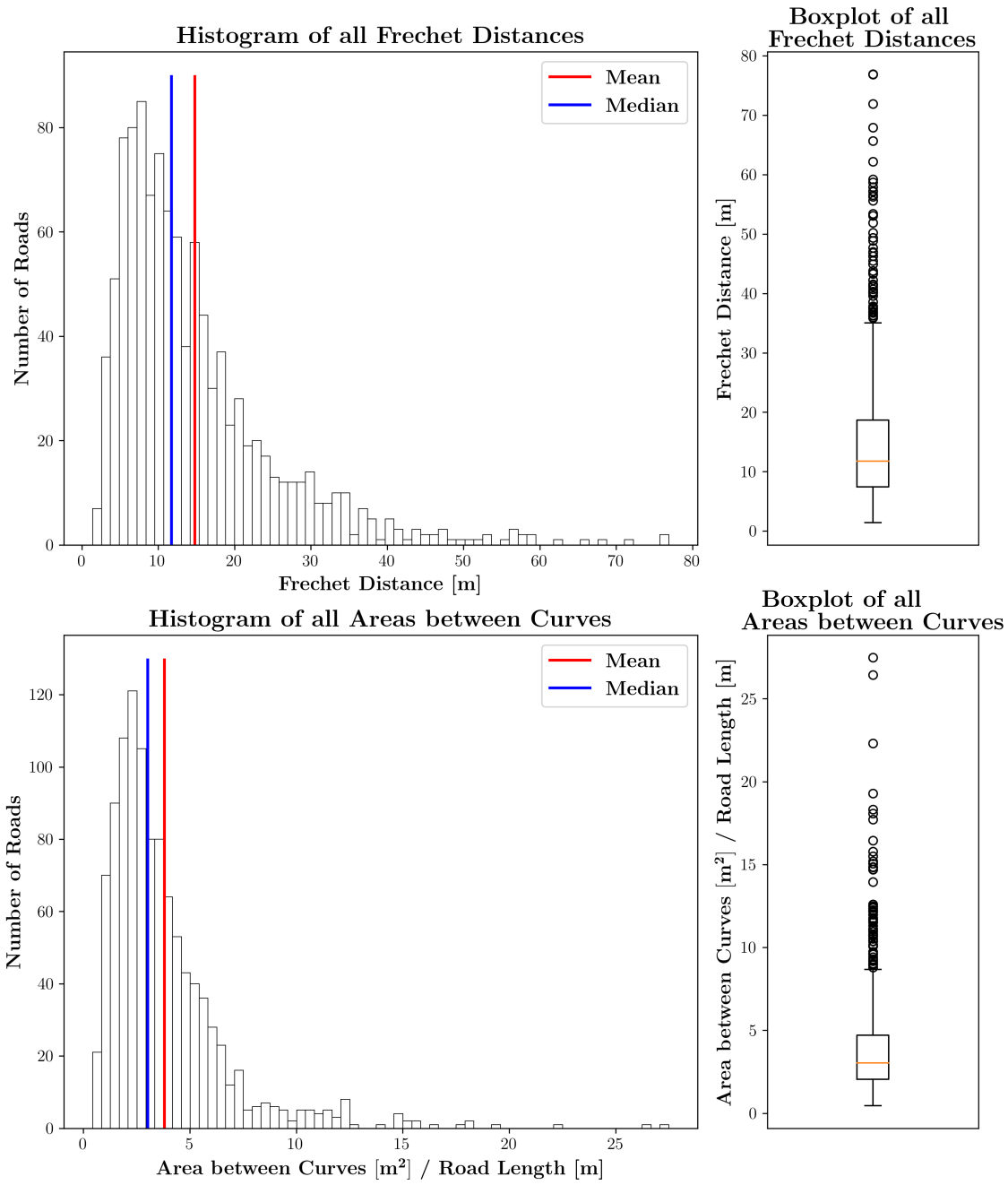


Figure 4.4: Histogram and boxplot of the computed performance metrics of an exemplary simple CAE with the hyper-parameter configuration (hidden size: 128, coordloss)

Description of Aggregated Metrics The median values of the two metrics for the different CAE models are shown in Table 4.2. The mean median of the FDs of all the models is 9.94m (sd = 2.67m) and the mean standard deviation is 10.26m (sd = 0.78m). The mean median of the rABCs of all the models is 2.5 (sd = 0.86) and the mean standard deviation is 2.9 (sd = 0.02). The CAE that performed best was the coordloss ResUNet with a median FD of 6.45m and a median rABC of 1.40. It outperforms the coordloss U-Net by 0.32m regarding the FD and 0.01 regarding the rABC, while outperforming the best simple CAE (fourierloss and hidden size of 256) by 5m regarding the FD and 1.6 regarding the rABC. The two U-Nets in general performed much better than the simple CAEs. The difference within the U-Net variants is very small with a standard deviation of 0.14m regarding the FD and 0.03 regarding the rABC. The standard deviation between the medians of the simple CAEs is 0.55m regarding the FD and 0.19 regarding the rABC, which does not indicate large fluctuations as well.

The effect of the Fourier Loss Extension is the following. A double arrow means an improvement in performance:

- Simple CAE (hs256): FD: $12.1m \Rightarrow 11.45m$, rABC: $3.07 \Rightarrow 2.99$
- U-Net: FD: $6.77m \Rightarrow 6.68m$, rABC: $1.41 \rightarrow 1.49$
- ResUNet: FD: $6.45m \rightarrow 6.83m$, rABC: $1.40 \rightarrow 1.55$

	Median Fréchet Distance	**Median Area Between Curves**
cae_hs64_coordloss	11.79	3.18
cae_hs64_fourierloss	13.15	3.54
cae_hs128_coordloss	11.74	3.03
cae_hs128_fourierloss	12.42	3.32
cae_hs256_coordloss	12.1	3.07
cae_hs256_fourierloss	11.45	2.99
unet_coordloss	6.77	1.41
unet_fourierloss	6.68	1.49
resunet_coordloss	6.45	1.40
resunet_fourierloss	6.83	1.55

Table 4.2: Computed performance metrics of the CAEs, yellow: Best model of simple CAEs, orange: Overall best CAE

4.1.3 RNN

Distribution of Metrics As depicted in Figure 4.5, the distributions of both metrics describing the difference between the target and the prediction are strongly skewed and consist of many outliers. The mean is considerably higher than the median for both metrics. Thus, again, the median is used as aggregation function. Similar distributions arise for all the RNNs.

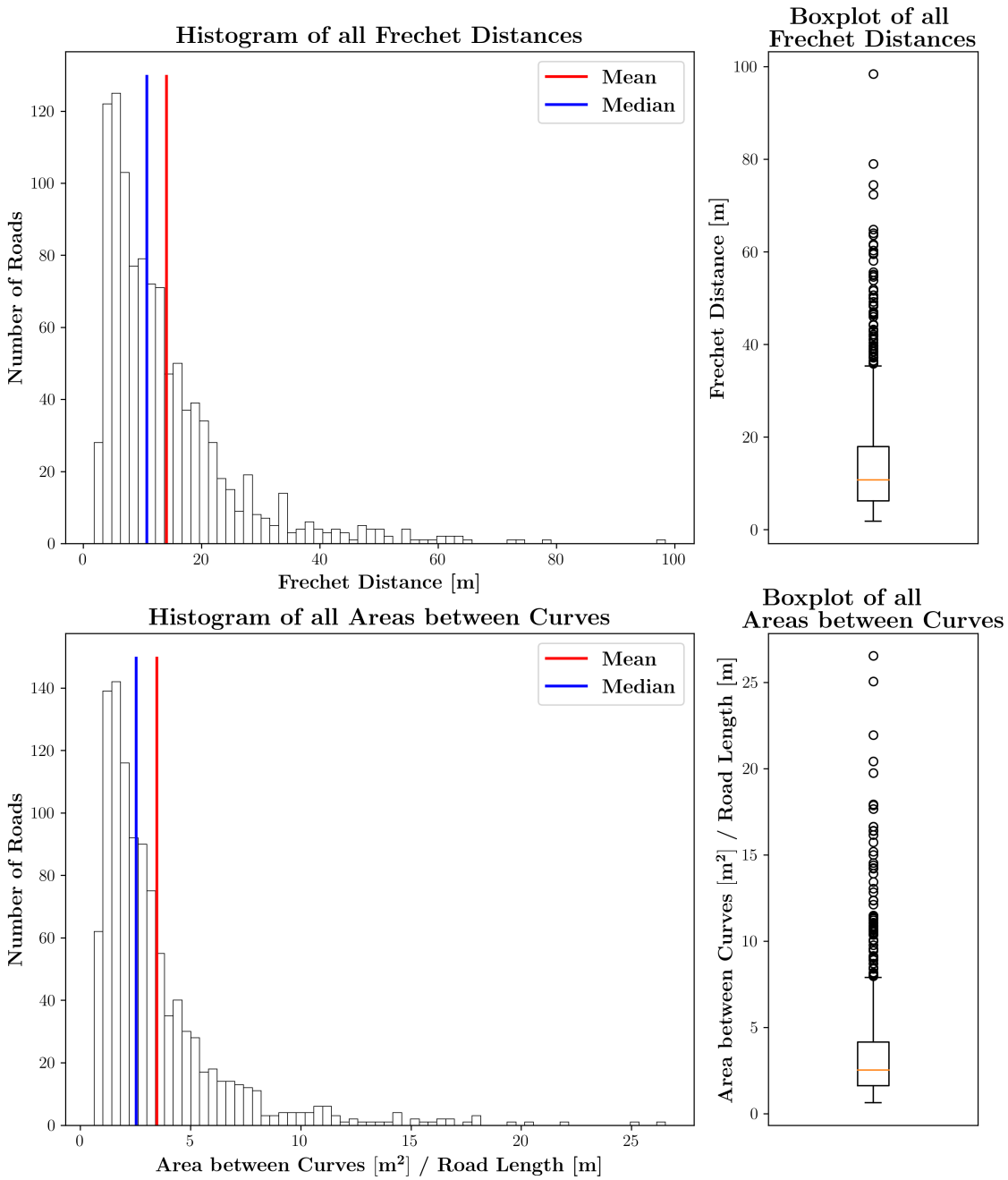


Figure 4.5: Histogram and boxplot of the computed performance measures of an exemplary LSTM with the hyper-parameter configuration (hidden size: 256, 1 layer, collapsing strategy: "allsteps", coordloss). The uppermost outlier isn't included in the plots (FD: 508m, rABC: 102)

Description of Aggregated Metrics The mean median of the FDs of all the models is 18.74m (sd = 21.5m) and the mean standard deviation is 10.26m (sd = 60.92m). The mean median of the rABCs of all the models is 4.22m (sd = 4.02m) and the mean standard deviation is 6.58m (sd = 3.92m).

The RNNs have an additional slot in their names, referring to the collapsing strategies that are described in Section 3.3.3: "allsteps" for the first strategy, "vertexbased" for the second, and "laststep" for the third. The aggregated metrics of the LSTMs are split corresponding to the 4 steps in the description of the hyper-parameter selection in Section 3.3.3 and are shown in Tables 4.3 - 4.5. The first step identifies the best configurations in terms of hidden size and the number of layers, while the three subsequent steps explore the effect of the collapsing strategy, bi-directionality, and the Fourier Loss Extension on these configurations:

1. **Table 4.3: Systematic Comparison of hidden sizes and number of layers**
   Keeping the collapsing strategy ("allsteps"), the loss function ("coordloss"), and the direc-

tionality ("unidirectional") stable, the RNN that performed best in terms of the FD is the one with a hidden size of 512 and 1 layer and in terms of the rABC the one with a hidden size of 128 and 2 layers. There is a clear improvement in performance when increasing the hidden size from 64 to 128 regarding both metrics. With lower hidden sizes, increasing the number of layers caused an improvement of the performance. However, both effects were mitigated with higher hidden sizes. To exclude the possibility that more resources (larger/more layers) would cause a better performance in the subsequent experiments, the models with a hidden size of 512 and 3 layers were taken to Steps 2, 3, and 4 too.

	Median Fréchet Distance	**Median Area Between Curves**
lstm_hs64_1l_allsteps_coordloss	15.44	3.94
lstm_hs64_2l_allsteps_coordloss	13.85	3.43
lstm_hs64_3l_allsteps_coordloss	12.33	2.95
lstm_hs128_1l_allsteps_coordloss	12.66	3.00
lstm_hs128_2l_allsteps_coordloss	10.46	2.37
lstm_hs128_3l_allsteps_coordloss	11.02	2.56
lstm_hs256_1l_allsteps_coordloss	10.78	2.54
lstm_hs256_2l_allsteps_coordloss	10.70	2.57
lstm_hs256_3l_allsteps_coordloss	10.37	2.67
lstm_hs512_1l_allsteps_coordloss	9.92	2.52
lstm_hs512_2l_allsteps_coordloss	10.45	2.76
lstm_hs512_3l_allsteps_coordloss	10.93	2.73

Table 4.3: Computed performance metrics comparing different hidden sizes and number of layers, orange: Best configurations

2. **Table 4.4: Comparison of Collapsing Strategies**
   The "vertexbased" model outperformed its "allsteps" counterpart in the "hs512_1l" setting, being the best performing model in this step but performing worse in the other two settings "hs128_2l" and "hs512_3l". The "laststep" models performed the worst in all three tested hyper-parameter configurations.

	Median Fréchet Distance	Median Area Between Curves
lstm_hs128_2l_allsteps_coordloss	10.46	2.37
lstm_hs128_2l_vertexbased_coordloss	54.61	5.4
lstm_hs128_2l_onlylast_coordloss	111.86	22.11
lstm_hs512_1l_allsteps_coordloss	9.92	2.52
lstm_hs512_1l_vertexbased_coordloss	9.29	2.11
lstm_hs512_1l_onlylast_coordloss	25.55	8.07
lstm_hs512_3l_allsteps_coordloss	10.93	2.73
lstm_hs512_3l_vertexbased_coordloss	11.27	2.81
lstm_hs512_3l_onlylast_coordloss	20.72	6.22

Table 4.4: Computed performance metrics of three LSTM configurations comparing the collapsing strategies, orange: Best configurations

3. **Table 4.5: Compare Effect of Bi-Directionality**
   The hyper-parameter configurations that the three collapsing strategies best performed on were taken to this step to exclude the possibility that the effect of bi-directionality could have an especially great effect on a certain collapsing strategy. The best-performing model in this step is the same as in step 2. The introduction of bi-directionality only had an improving effect on the collapsing strategy "laststep"

	Median Fréchet Distance	Median Area Between Curves
lstm_hs512_1l_allsteps_coordloss	9.92	2.52
lstm_hs512_1l_allsteps_coordloss_bidirectional	10.17	2.77
lstm_hs512_1l_vertexbased_coordloss	9.29	2.11
lstm_hs512_1l_vertexbased_coordloss_bidirectional	9.34	2.3
lstm_hs512_3l_onlylast_coordloss	20.72	6.22
lstm_hs512_3l_onlylast_coordloss_bidirectional	17.48	5.06

Table 4.5: Computed performance metrics comparing the effect of bi-directionality on best configurations of different collapsing strategies, orange: Best configurations

4. **Table 4.6: Compare Effects of the Fourier Loss Extension**
   The better configurations for each collapsing strategy in Step 3 were taken to this step, where they were tested on the effect of the Fourier Loss Extension, which had only a positive effect on the rABC of the "vertexbased" LSTM, being the best-performing LSTM in general in terms of the rABC.

	Median Fréchet Distance	Median Area Between Curves
lstm_hs512_1l_allsteps_coordloss	9.92	2.51
lstm_hs512_1l_allsteps_fourierloss	10.3	2.58
lstm_hs512_1l_vertexbased_coordloss	9.29	2.11
lstm_hs512_1l_vertexbased_fourierloss	11.25	2.09
lstm_hs512_3l_onlylast_coordloss_bidirectional	17.48	5.06
lstm_hs512_3l_onlylast_fourierloss_bidirectional	18.88	5.77

Table 4.6: Computed performance metrics comparing the effect of the Fourier Loss Extension on best configurations of different collapsing strategies and bi-directionality, orange: Best configurations

4.1.4 Comparison of Architectures

The aggregated metrics of the best-performing models of the three architectures are shown in Table 4.7. The best model in terms of both metrics is the coordloss ResUNet with a median FD of 6.45 and a median rABC of 1.4. The difference is less pronounced with regard to the rABC (GCN: 2.6 and LSTM: 2.09). For further comparisons in the next paragraph and Section 4.2, the following hyper-parameter configurations for the different model architectures are used: *gcnn_hs256_10layers_coordloss*, *resunet_coordloss*, and *lstm_hs512_1l_vertexbased_fourierloss*.

	Median Fréchet Distance	Median Area Between Curves
gcnn_hs256_5layers_coordloss	13.74	2.64
gcnn_hs256_10layers_coordloss	24.95	2.6
resunet_coordloss	6.45	1.4
lstm_hs512_1l_vertexbased_coordloss	9.29	2.11
lstm_hs512_1l_vertexbased_fourierloss	11.25	2.09

Table 4.7: Comparison of performance metrics of the best-performing models of the different architectures, orange: Best model

4.1.5 Further Analytics

As described in Section 3.4.2, two additional comparisons have been made using the two metrics. They are described separately and are visualised in Figure 4.6:

- **Difficulty vs. Performance**
  The mean difference of the large-scale roads and the proposed generalisations by swisstopo (difficulty) is 6.41m regarding the Fréchet Distance (FD), and 1.61 regarding the relative Area between the Curves (rABC). The performance of all architectures show a strong negative correlation with the difficulty (Pearson coefficient > 0.7, p-value ≈ 0).

- **Actual vs. Predicted Dissimilarity**
  The predicted dissimilarities of the different architectures are shown in Tables 4.8 - 4.10. Because the standard deviation of the medians of the CAEs and the RNNs is comparatively high, these architectures are broken down into sub-groups. The CAEs are broken down into the two sub-architectures simple CAE and U-Net variants and the RNNs into the collapsing strategies. The means of the medians within the CAE sub-architectures seem to be rather stable with standard deviations of 0.6m and 0.3m regarding the FD, and 0.2 and 0.08 regarding the rABC. Regarding the FD, the breakdown of the RNNs only reduced the standard deviation of the "allsteps" collapsing strategies (1.74m), the standard deviation of the other two remained high (18.12m and 36.77m). Regarding the rABC, the breakdown of the RNNs

reduced the standard deviation of the "allsteps" and the "vertexbased" models (0.42 and 0.78) but not of the "laststep" model (6.31). The predicted dissimilarity of most of the (sub-)architectures is higher than the actual dissimilarity that is suggested by swisstopo regarding both metrics. Only the U-Net variants predict lower dissimilarities than swisstopo.

Regarding the scatterplots in Figure 4.6, the predicted dissimilarity of the GCNN and the RNN show a low correlation with the actual dissimilarity (Pearson coefficient ≈ 0.2) the correlation is moderate regarding the predictions of the CAE (Pearson coefficient ≈ 0.4). All correlations are statistically significant (p-values $< 1e-6$).

	swisstopo	**GCNN**	**CAE**	**RNN**
FD: mean of medians	**6.41**	**20.04**	**9.47**	**18.13**
FD: sd of medians	/	6.27	3.25	21.84
rABC: mean of medians	**1.61**	**2.84**	**2.55**	**4.08**
rABC: sd of medians	/	0.36	0.95	4.05

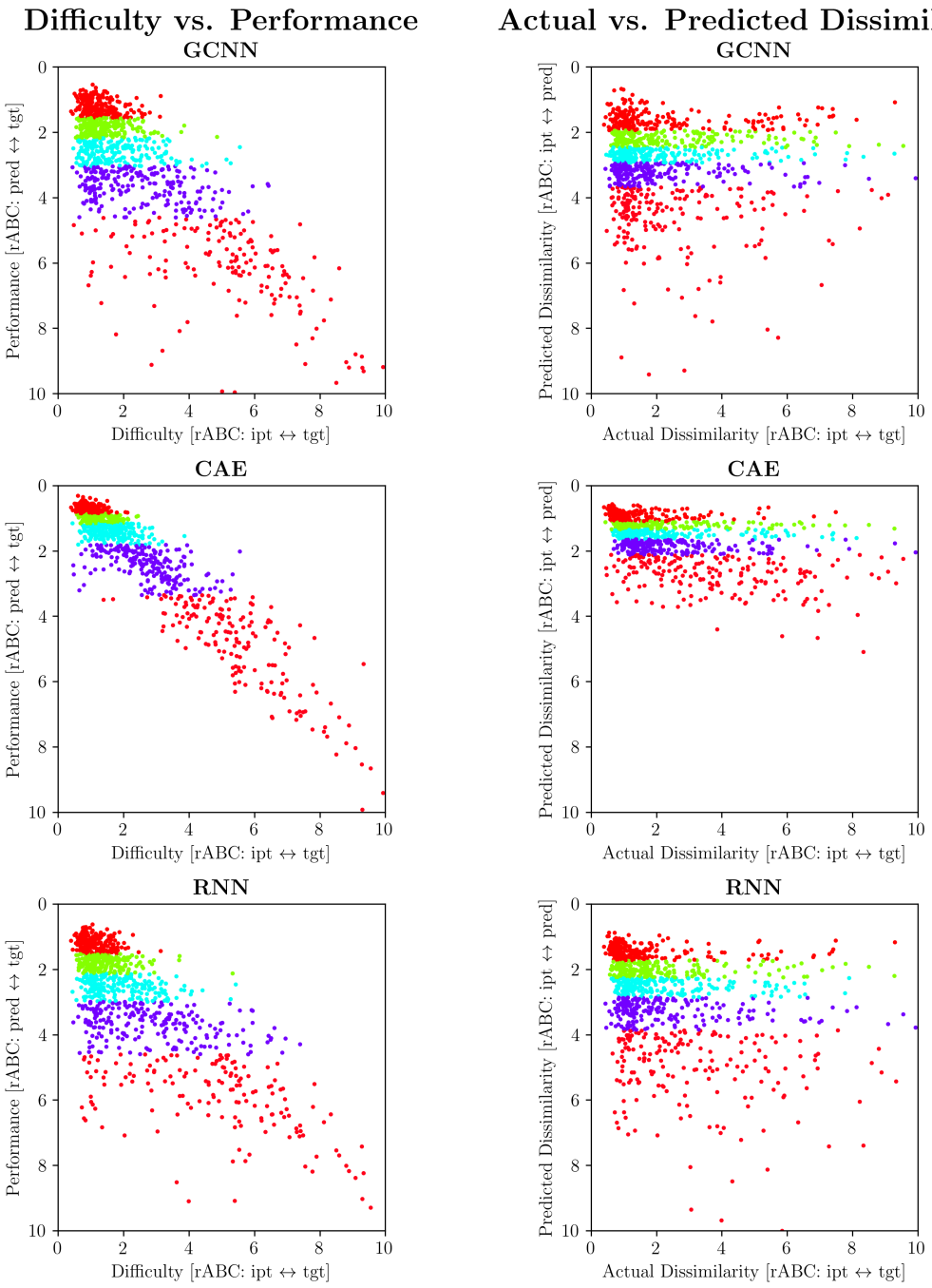
Table 4.8: Actual dissimilarities by swisstopo and predicted dissimilarities by the architectures (medians are computed per model, means per architecture)

	Simple CAEs	**U-Net Variants**
FD: mean of medians	**12.1**	**5.53**
FD: sd of medians	0.6	0.3
rABC: mean of medians	**3.31**	**1.41**
rABC: sd of medians	0.2	0.08

Table 4.9: Predicted dissimilarities of the simple CAEs and the U-Net variants (medians are computed per model, means per sub-architecture)

	allsteps	**vertexbased**	**onlylast**
FD: mean of medians	**10.87**	**17.2**	**39.4**
FD: sd of medians	1.74	18.12	36.77
rABC: mean of medians	**2.84**	**2.13**	**9.52**
rABC: sd of medians	0.42	0.78	6.31

Table 4.10: Predicted dissimilarities of the different RNN collapsing strategies (medians are computed per model, means per collapsing strategy)



(a) Comparison between the difficulty of the roads and the performance of the models

(b) Comparison between the actual and the predicted dissimilarity

Figure 4.6: Further analysis of the models using the relative Area between Curves. The roads are coloured according to the quintile that their performances, or their predicted dissimilarities are in. (values that are higher than 10 are not depicted in the scatter plots)

4.2 Qualitative Description

4.2.1 Effect of Number of Layers in GCNNs

The number of message passing layers has an influence on the smoothing of the predicted roads. An example is shown in Figure 4.7. GCNNs with 10 layers show a more pronounced smoothing of the input roads than GCNNs with 5 layers.

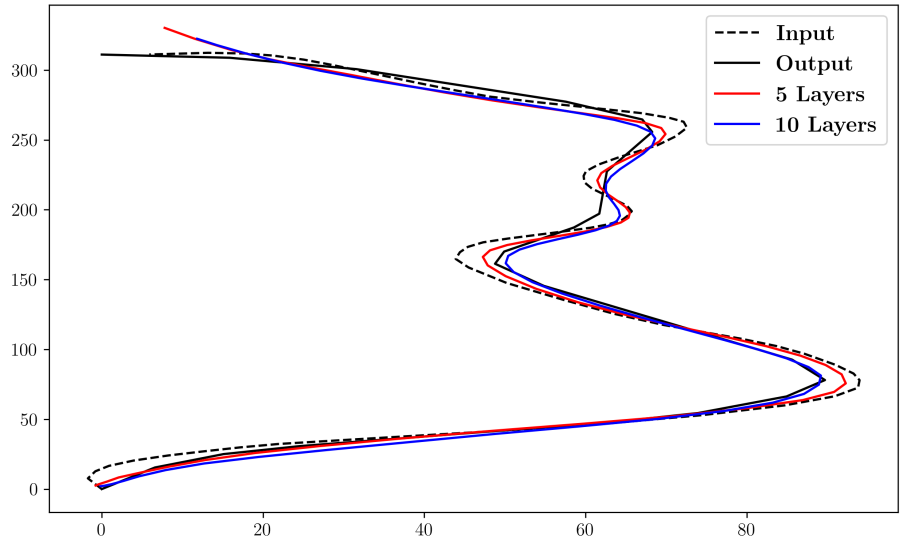


Figure 4.7: Exemplary predictions of two GCNNs with a hidden size of 32, and 5 or 10 layers

4.2.2 Comparison between simple CAE and ResUNet

The predictions of the simple CAEs and the U-Net differ substantially in their included frequency spectra. The predictions of the best-performing models of the two sub-architectures on two examples of easy, intermediate, and hard roads each (classification as described in Section 3.4.2) are shown in Figures 4.8 - 4.10. The predicted roads of the simple CAEs consist of only very low frequencies and over-smooth the input severely, which can be clearly seen in Figure 4.8, Road 1 and Figure 4.9, Road 2. Narrow bends are oftentimes straightened (see Figure 4.9, Road 2). The U-Net variants, represented by the ResUNet, remove high frequencies too, but to a lesser extent.

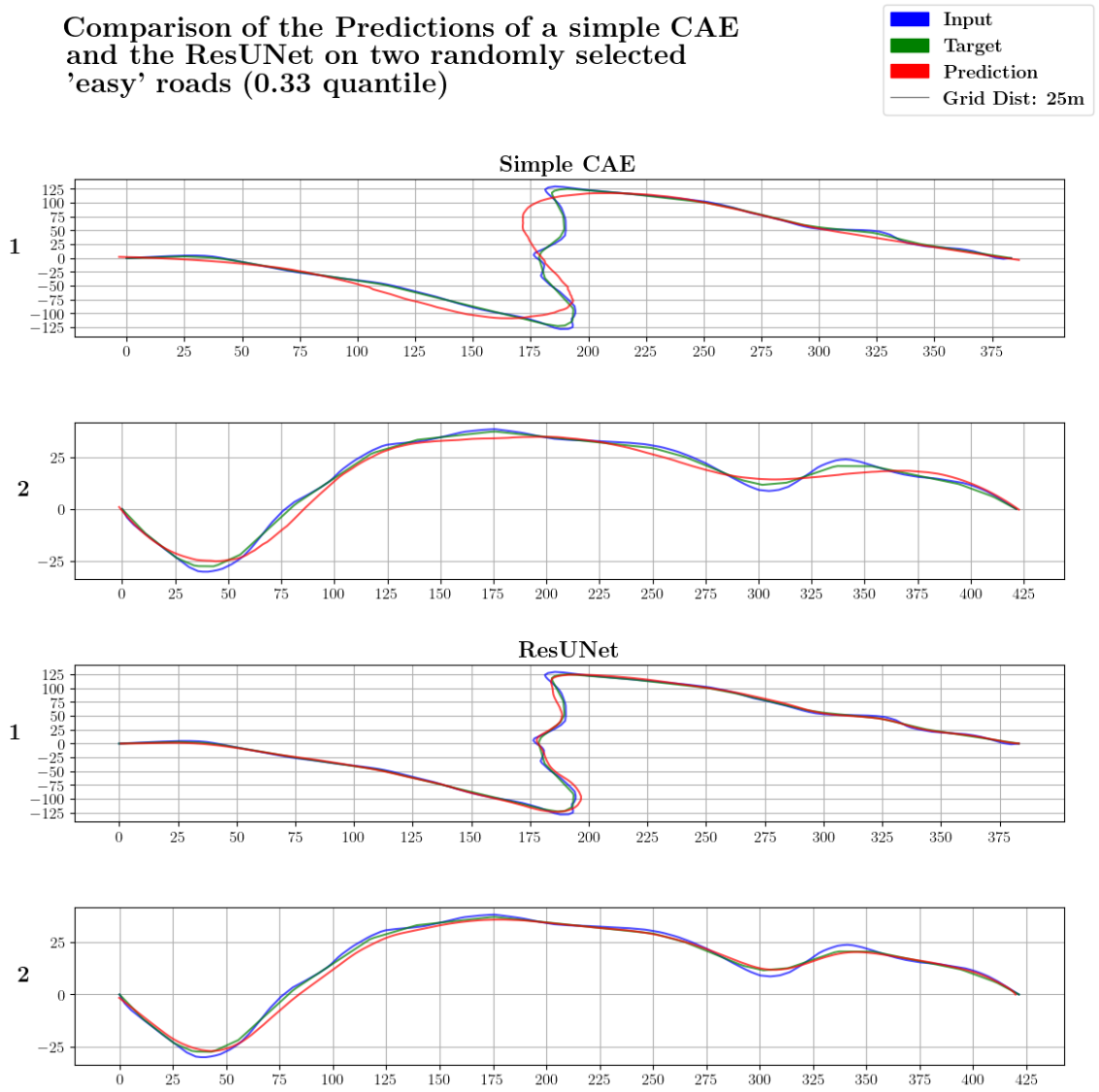


Figure 4.8: Comparison between the predictions of a simple CAE and the ResUNet on two randomly selected "easy" roads

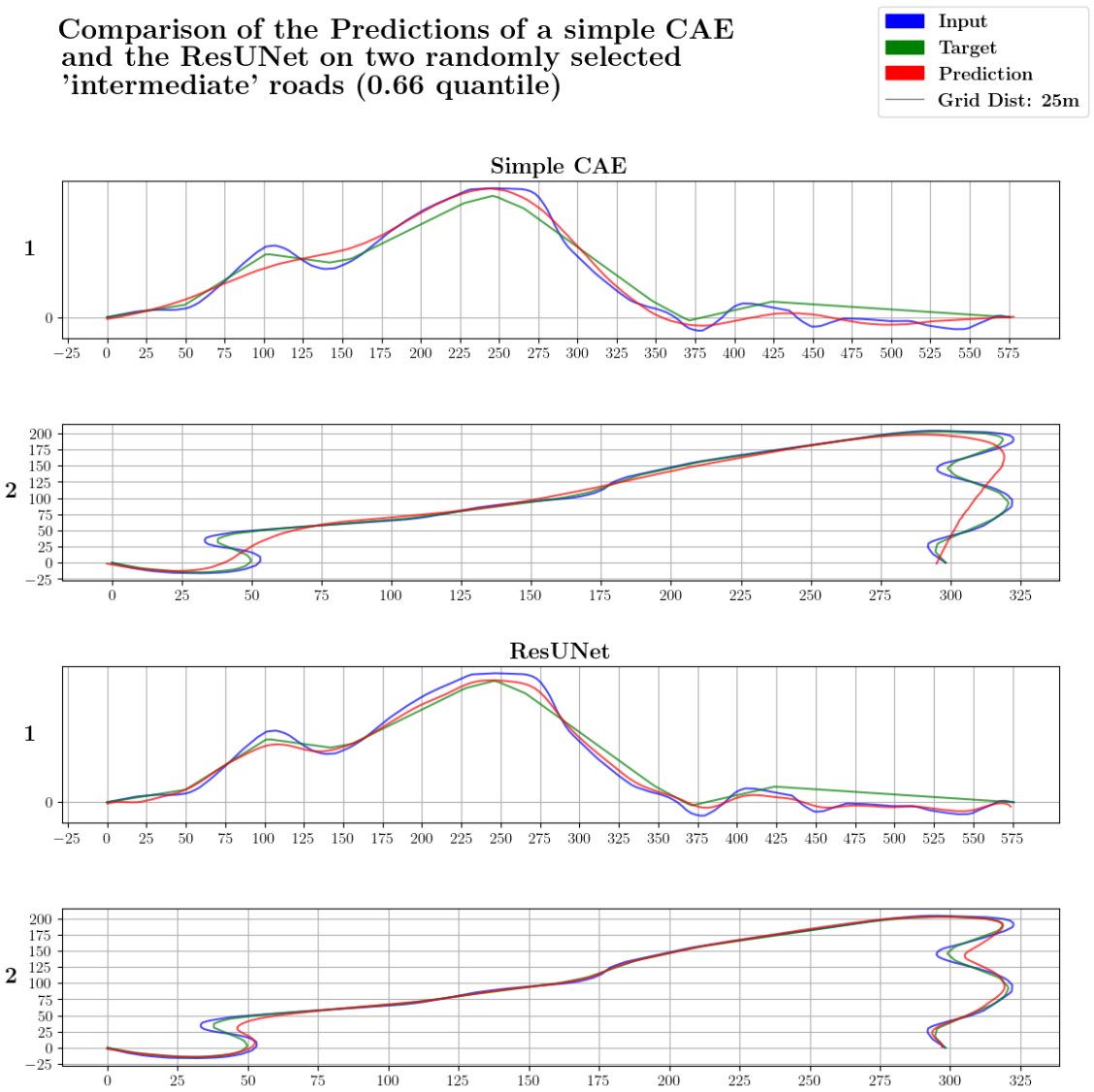


Figure 4.9: Comparison between the predictions of a simple CAE and the ResUNet on two randomly selected "intermediate" roads

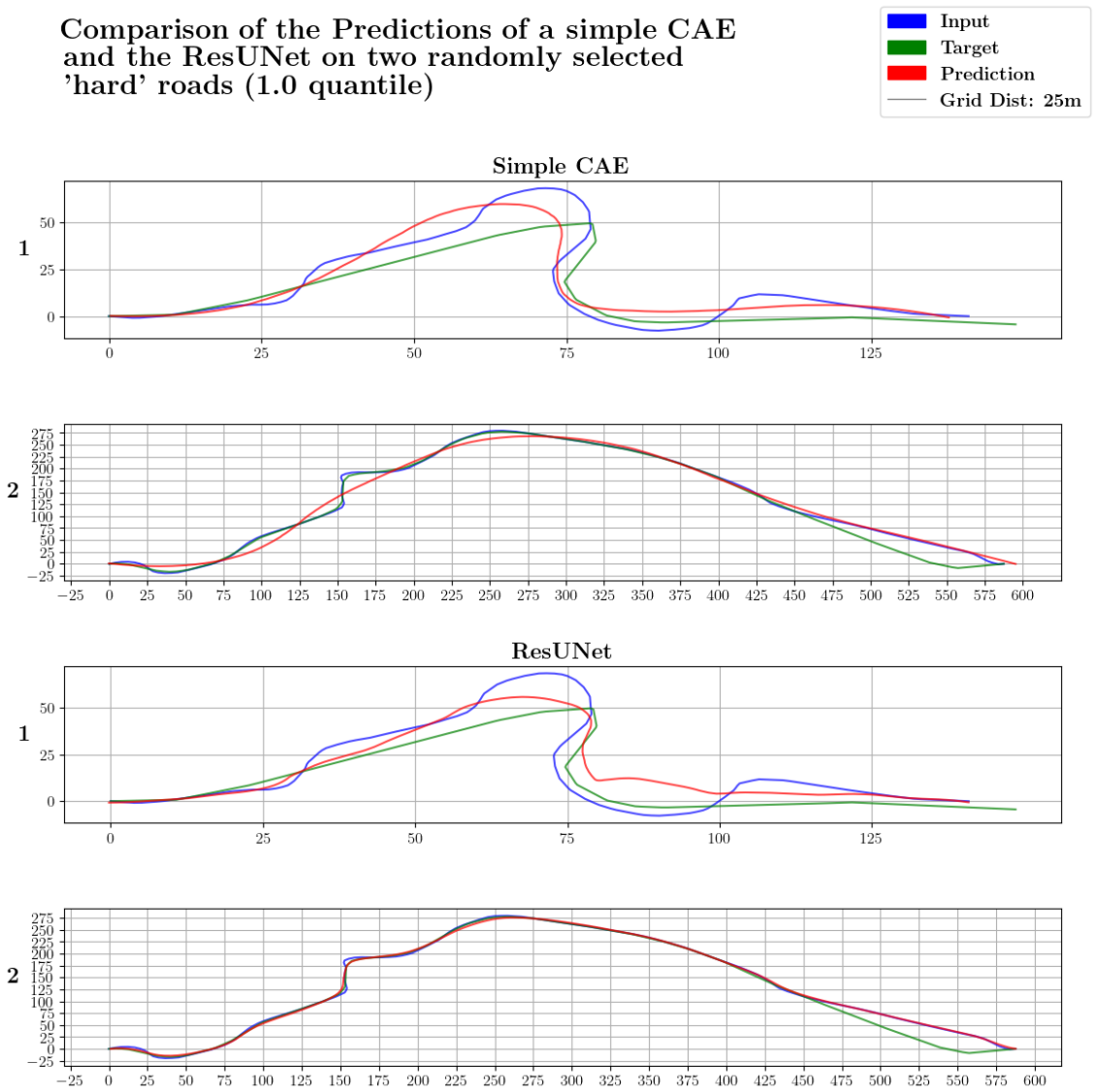


Figure 4.10: Comparison between the predictions of a simple CAE and the ResUNet on two randomly selected "hard" roads

4.2.3 Comparison of RNN Collapsing Strategies

As described in Section 4.1.3, the LSTMs with a "vertexbased" collapsing strategy perform the best in terms of FD and rABC. However, there is a discrepancy between the quantitative and the qualitative observations. In Figure 4.11 - 4.13, predictions of the different strategies on two randomly selected examples of easy, intermediate, and hard roads each are shown. For each strategy, the model that best performed in terms of the rABC has been included in this visual analysis. The strategies' predictions differ quite substantially. The "vertexbased" strategy's predictions follow the input lines very strictly and even introduce additional high frequencies and noise (Figure 4.13, Road 1). The other two strategies seem to only pick up the very low frequencies, especially the "laststep", that only outputs a rough, very smooth shape of the road (Figure 4.13, Road 1 and 2). Although the "vertexbased" strategy performs best in terms of the two metrics, it introduces much noise and sometimes very erratic predictions. Thus, for further visual analysis in the comparisons between the architectures, the "allsteps" strategy is chosen.

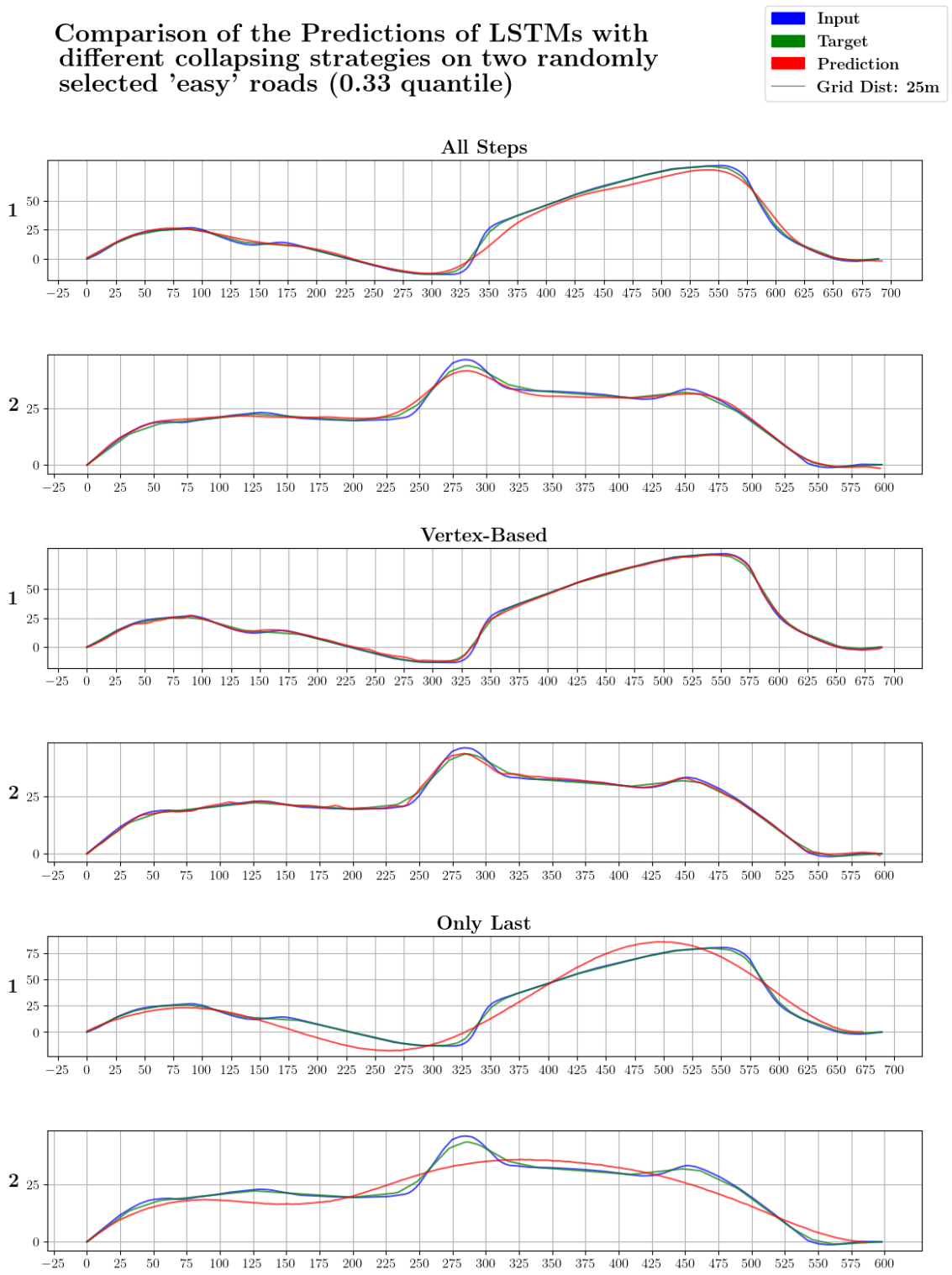


Figure 4.11: Predictions of RNNs with the three different collapsing strategies on two randomly selected "easy" roads

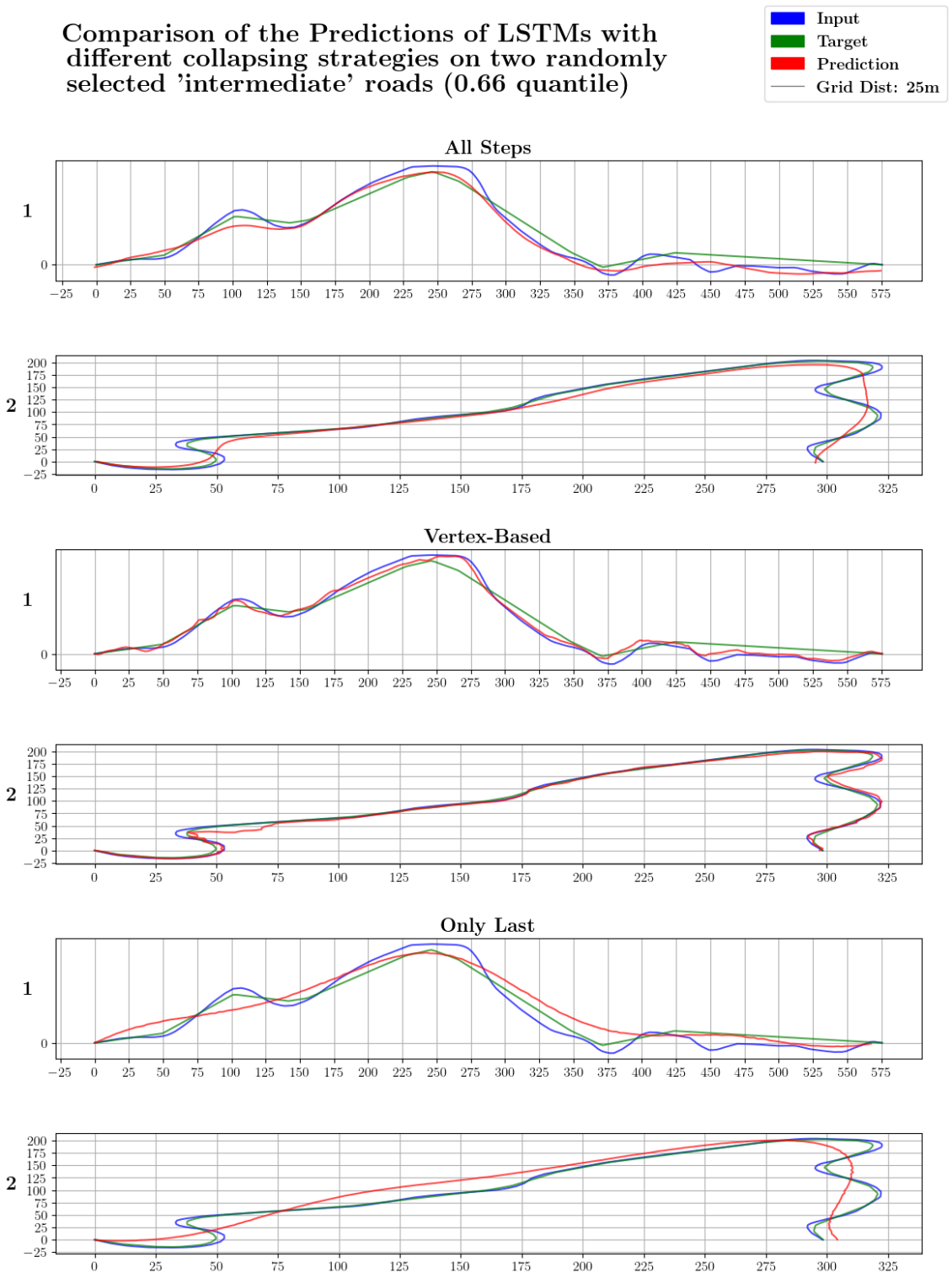


Figure 4.12: Predictions of RNNs with the three different collapsing strategies on two randomly selected "intermediate" roads

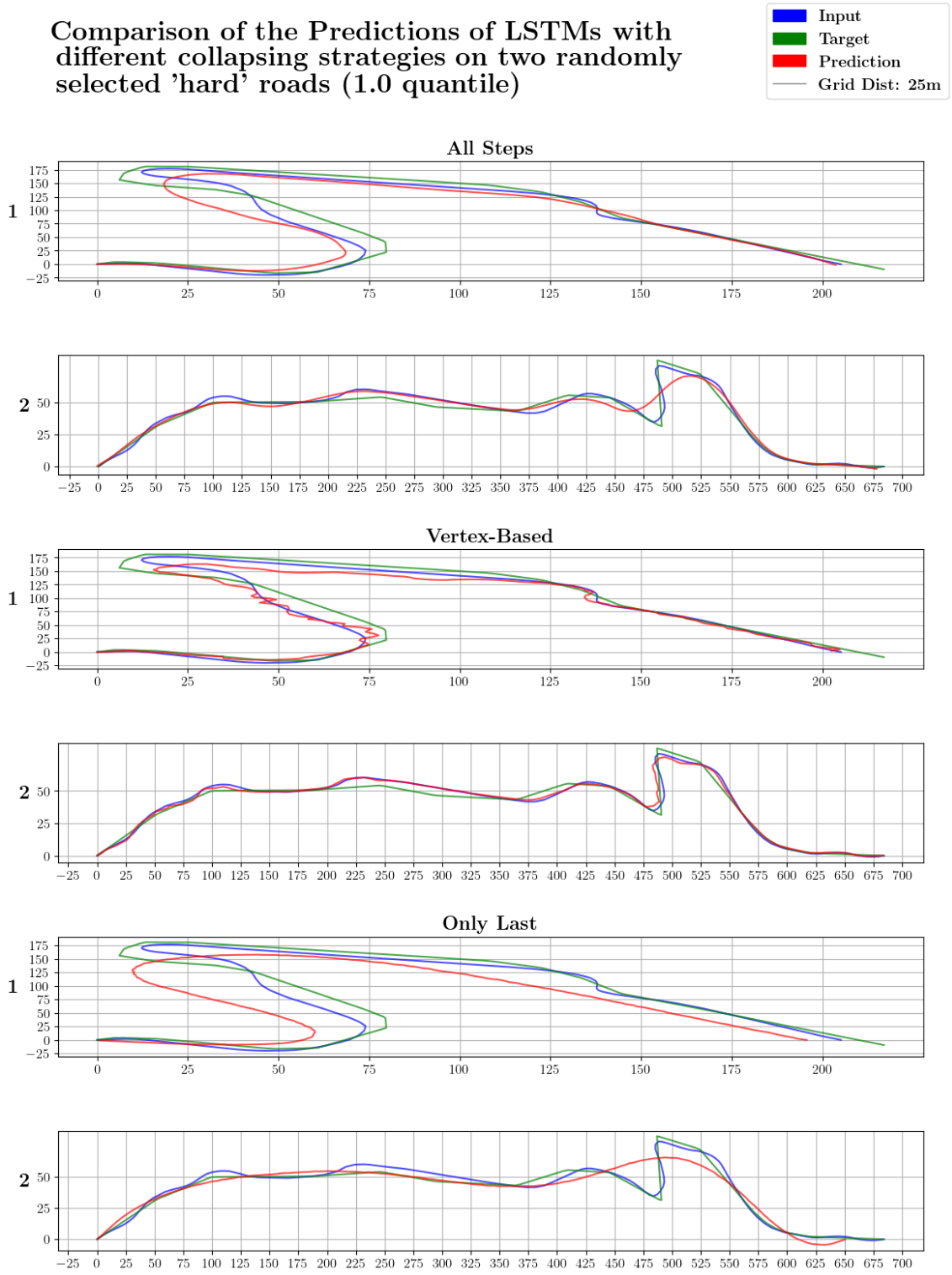


Figure 4.13: Predictions of RNNs with the three different collapsing strategies on two randomly selected "hard" roads

4.2.4 Comparison of Architectures

Easy Roads The roads with a small rABC between the input and target roads often show smooth curves without significant interruptions. The architectures' predictions follow these low-frequency targets and input roads well. All the architectures reduce the bend at $x \approx 375$ in Road 2 and stay smooth throughout. The large bend at the left side of Road 1 produces different outputs across the architectures. While the output predicted from the GCNN follows the input very strictly, the CAE and the RNN exaggerate this bend to a high degree.

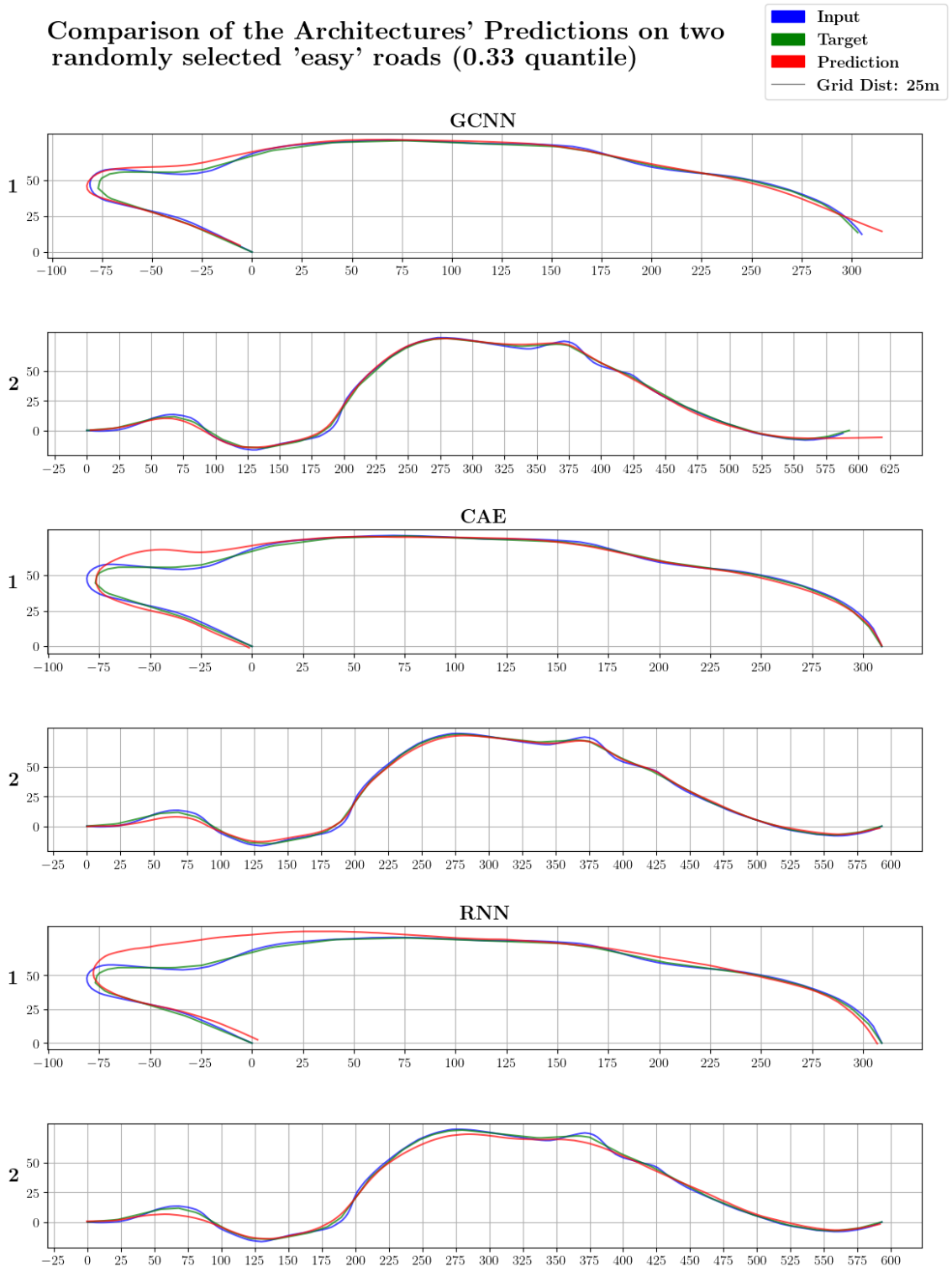


Figure 4.14: Predictions of the three models on two randomly selected "easy" roads

Intermediate Roads The intermediate roads are characterised by a larger rABC between the input and target roads introduced by generalisation operators. The most apparent operator applied by swisstopo in the intermediate roads is the smoothing operator. All three architectures smooth the input roads to a certain degree. The GCNN's predictions generally stay "inside" the curves of the input roads. The CAE exaggerates and smooths the bend at the right side of Road 1, while the RNN's prediction shows over-smoothing in that bend.

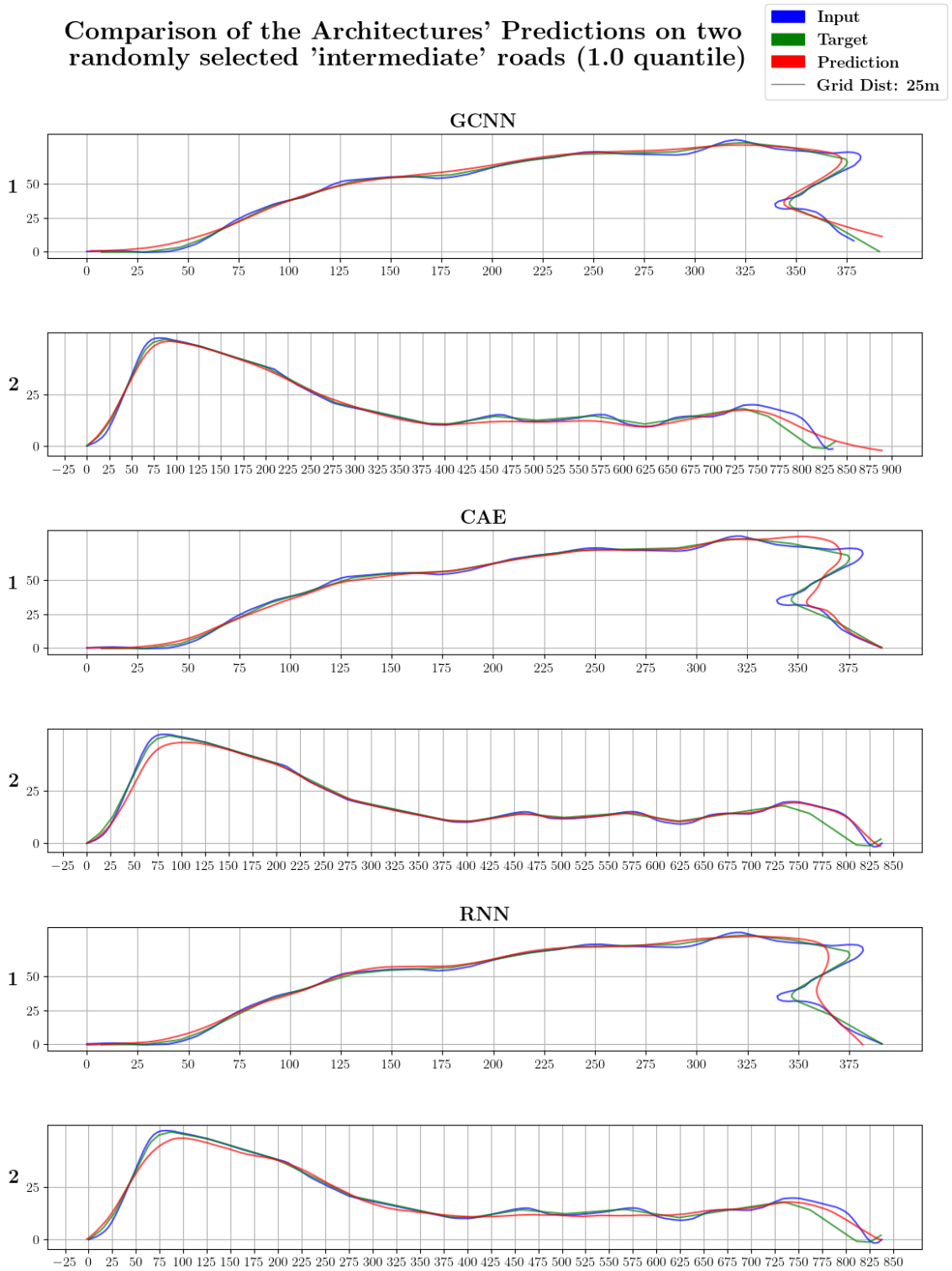


Figure 4.15: Predictions of the three models on two randomly selected "intermediate" roads

Hard Roads Hard roads sometimes consist of very high frequencies with much deformation in the ground-truth generalisations. Again, the GCNN smooths both roads to a moderate degree but does not introduce more deformation. The CAE and the RNN also predict smoothed versions of the input in Road 1. The RNN removes the sharp bends on the right side of Road 1 and the left side of Road 2. The large bend on the left side of Road 2 is exaggerated by both the CAE and the RNN, although the exaggerated bend of the CAE is less narrow.

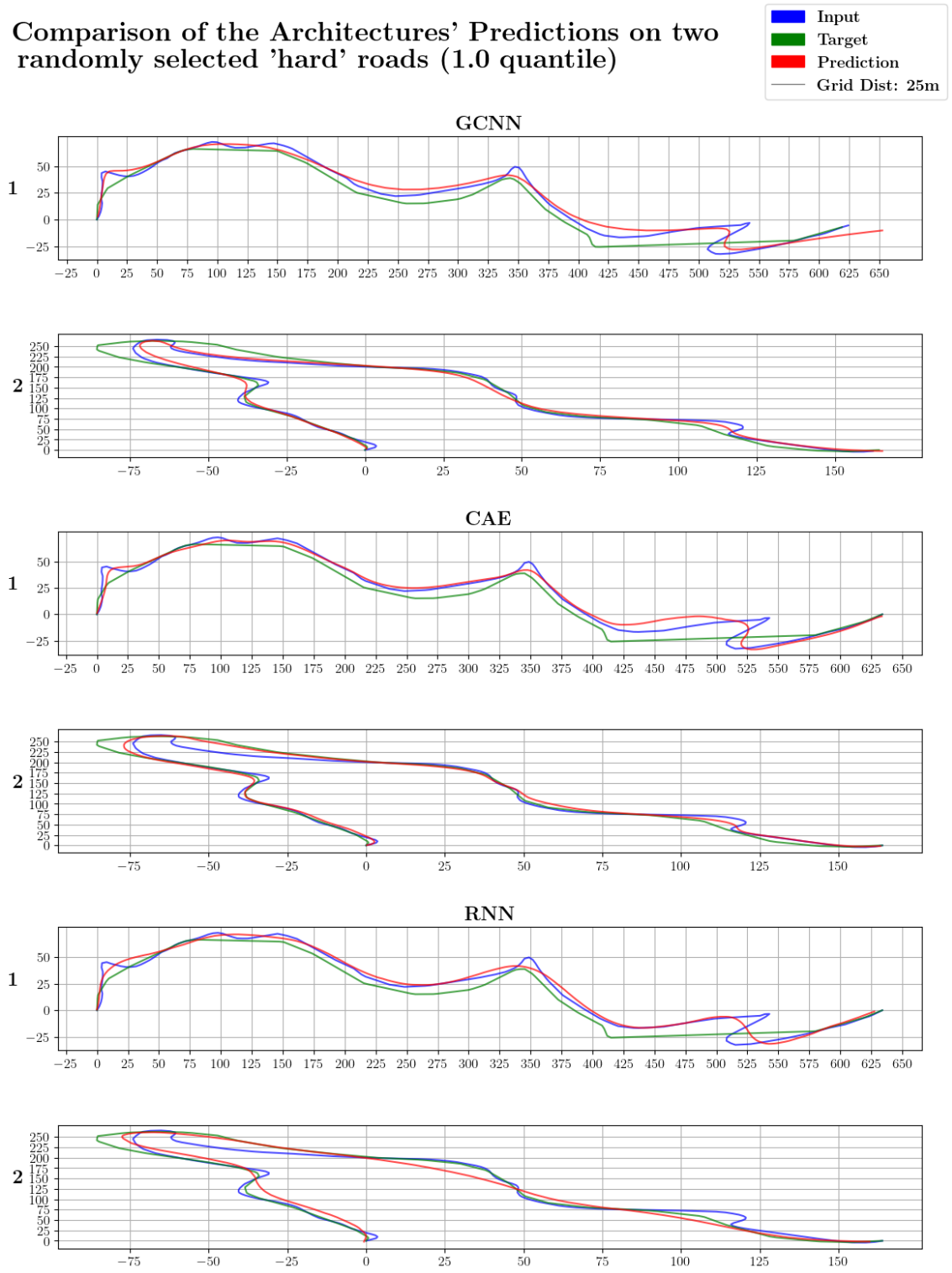


Figure 4.16: Predictions of the three models on two randomly selected "hard" roads

4.3 Loss Curves

As described in Section 3.3, the models have been trained until no further meaningful decrease in the evaluation loss could be detected. A sample of the resulting loss curves is depicted in Figures 4.17 - 4.18. The curves of the CAEs and the RNNs are displayed using logarithmic y-scales to better show the resulting loss values. Due to time constraints, not all models could be trained until their loss values converged fully. The GCNNs have been trained for 3200 epochs, the CAEs for 1600 epochs, and the RNNs for 800 epochs. Although the description of the models' training process is not part of the focus of this thesis, a few interesting observations can be described:

- The loss curves of GCNNs with hidden sizes higher than 128 begin to be very noisy after a certain number of epochs.
- When training the GCNNs and the "vertexbased" and the "laststep" LSTMs, the validation loss was lower than the training loss.
- The only models with a clear sign of over-fitting are all the U-Net variants.
- The U-Net variants converge the fastest across all architectures.
- The LSTMs with "vertexbased" and "laststep" collapsing strategies converge the slowest.

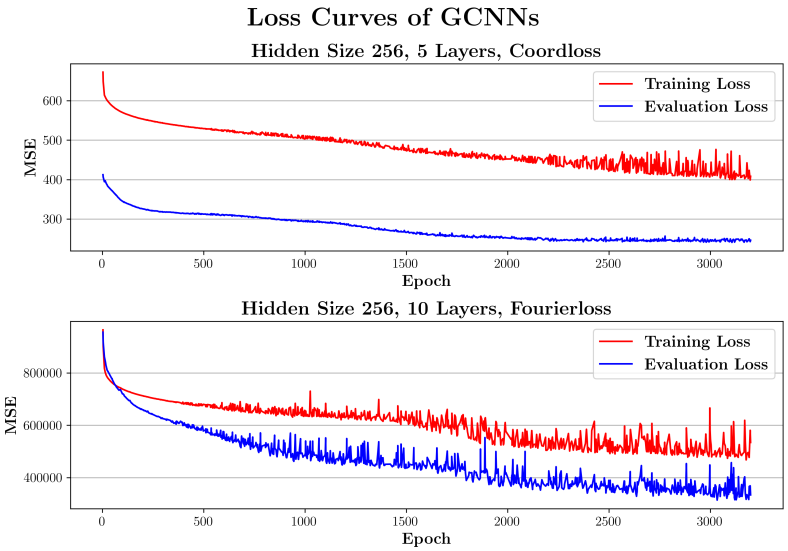


Figure 4.17: Loss Curves of two GCNN models

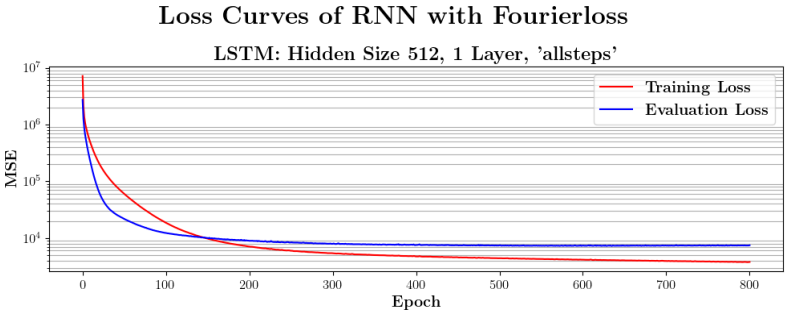


Figure 4.18: Loss Curves of three RNN models that have been trained using fourierloss

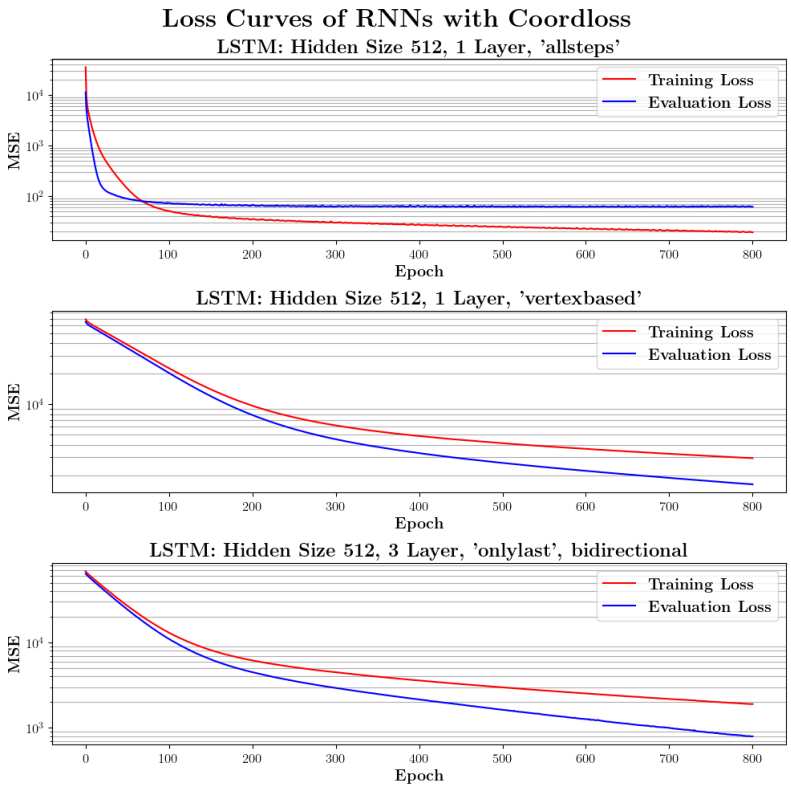


Figure 4.19: Loss Curves of three RNN models that have been trained using coordloss

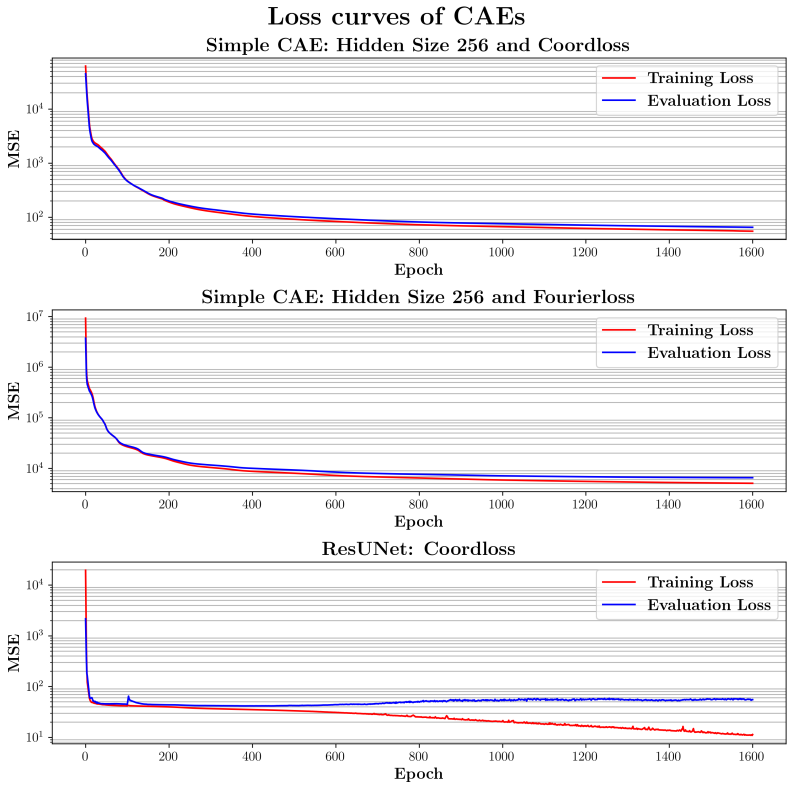


Figure 4.20: Loss Curves of three CAE models

Chapter 5

Discussion

5.1 Performance of Individual Architectures

5.1.1 GCNN

General Description As described in Section 4.1.1, increasing the hidden size of the models increases their performance with regard to the FD but not regarding the rABC. Thus, the models do not seem to differ much regarding the large single-event deviations but differ regarding the overall accuracy. No explanation of the large value for the FD of the GCNN with a hidden size of 256 and 10 layers could be found.

In low-frequency settings, the predictions of GCNNs follow their input roads very strictly (see Roads 1 and 2 in Figure 4.14). In high-frequency settings, GCNNs act like a moderate smoothing filter. Smoothing is also the only generalisation operator that can be identified in the predictions of the GCNNs. Unfortunately, no explanation for the "tick" in the predictions of the GCNNs, as described in Section 4.1.1, could be found.

Predicted Deformation The GCNN's ability to deform the input roads is limited. Thus, the GCNN performs well on "easy" roads because they are characterised by a low deformation of the ground truth. However, as soon as swisstopo's generalisation suggests the usage of an exaggeration operator, the quality of the GCNN's predictions decreases. However, this qualitative observation does not match the large predicted deformation depicted in Table 4.8. Possibly, the explanation for this is the effect of the "tick" described in Section 4.1.1. Although the last vertex hasn't been taken into account to compute the evaluation metrics, also the previous few vertices often show the most significant deviation of the input and the target, which can be clearly seen in Figure 4.7. To solve this problem in post-processing, the first and last vertices of the predictions could have been moved to the first and last vertices of the inputs. The remaining vertices of the predicted roads could then be moved accordingly by a "rubber-sheeting" transformation; for example by treating them like pebbles on a rubber sheet (Saalfeld, 1985). However, due to time constraints, this was not implemented.

Noisy Loss Curves As described in Section 4.3, the loss curves of the GCNNs with a hidden size of 128 and more begin to be noisy after a certain number of epochs. Unfortunately, no well-founded explanation for this was found. A hypothesis emerged that moderate smoothing is the basic working of the generalisation conducted by GCNNs, which can be learnt with hidden sizes of 128 and more. With larger hidden sizes, the models may try to learn additional deformation. As a result, after these models have learnt moderate smoothing, the loss gets noisy, because the additional deformation tasks are much harder to acquire. However, despite the noise, the general trend of the loss still points downwards.

General Remarks Generally, the GCNNs act like a smoothing filter with the kernel size defined by the number of layers. No signs of simplification or exaggeration operators can be observed.

5.1.2 CAE

The simple CAEs and the U-Net variants differ much regarding their performance (Section 4.1.2), their predicted dissimilarity (Section 4.1.5), and their qualitative assessment (Section 4.2.2). Thus, in this section, they are discussed separately.

Simple CAEs The quantitative performance does not seem to depend much on the hidden size regarding the median of either of the two metrics. As described in Section 4.2.2, the simple CAEs only manage to pick up the very general shapes of the input roads. However, not only do they over-smooth the input roads, but their output is always low-frequency, which means that in some cases, they apply an exaggeration operator (see Figure 4.8, Road 1). An additional effect of this low-pass filter is that narrow bends are removed, which could indicate a simplification operator (see Figure 4.9, Road 2). However, the ground truth by swisstopo consists of much higher frequencies than the predictions. Generally, it was expected that the Fourier Loss Extension could penalise this mismatch regarding frequency spectra. Although the best-performing simple CAE (which is also depicted in Figures 4.8 - 4.10) has been trained with the Fourier Loss Extension, the frequency spectra of the targets and the predictions do not match. A possible explanation could be that the bottleneck **h** forces the simple CAEs to discard the high-frequency information during the encoding step. A similar effect is visible in the outputs of auto encoders trained to encode and decode images (see Figure 5.1).

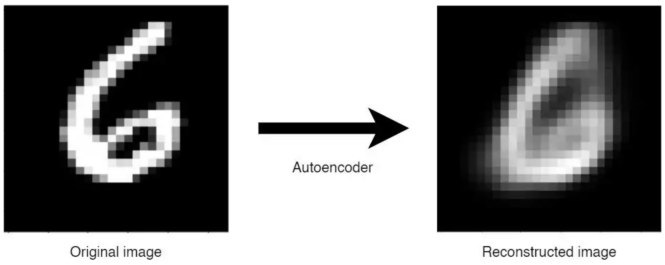


Figure 5.1: Example output of an auto encoder that has been trained on the MNIST dataset (Deng, 2012), source: `https://becominghuman.ai/the-deep-autoencoder-in-action-digit-reconstruction-bf177ccbb8c0`

U-Nets The U-Net variants are the overall best-performing models. The worst U-Net still performs better than the best model of the other architectures. As described in Section 4.1.2, the performance of all variants is similar. The predictions of the ResUNet in Figures 4.14 - 4.16 show examples of the generalisation operators smoothing (Figure 4.9, Road 1 and 2) and exaggeration (Figure 4.14, Road 1). Because it follows the input line more strictly than the simple CAEs, no signs of a simplification operator can be observed. Often, the predictions of the U-Nets seem legitimate, considering that no context information could be taken into account. However, some predictions are characterised by erratic patterns (Figure 4.15, Road 1). All the U-Net variants converged fast compared to the other architectures and over-fitted after about 200 epochs (see Figure 4.20).

Predicted Deformation As described in Section 4.1.5, the two CAE sub-architectures differ in the amount of predicted deformation too, but stay very consistent within the sub-architectures. The simple CAEs, due to their inability to produce high frequencies, overestimates the required deformation by about a factor of two in both metrics. The U-Nets stay within a sensible range around swisstopo's ground truth.

Remarks on Noisy Outputs All models of the CAE architecture predict noisy lines, although the simple CAEs' predictions are more affected than the U-Net variants. Considering the other architectures, this is only the case with the LSTMs using the "vertexbased" collapsing strategy. An explanation in the case of the CAEs could be that these models were developed in a computer vision setting. There is a significant difference between an actual image and the proposed input data. While the input raster gives the spatial context in images, the spatial context of the model inputs in this thesis is provided by the values of the raster. However, the **smoothed** curves sometimes show legitimate generalisation suggestions, and the U-Net variants quantitatively perform best regarding both metrics.

General Remarks CAEs seem to be able to acquire knowledge about the shapes of roads. Both the simple CAEs and the U-Nets predict small-scale representations of the input roads that often seem sensible. However, first and foremost, all CAEs apply the smoothing generalisation operator.

The simplification and exaggeration capabilities seem to be a byproduct of an applied low-pass filter.

5.1.3 RNN

Hidden Sizes and Number of Layers The general image of the comparison of the LSTMs with the "allsteps" collapsing strategies conveys an improvement of the performance with an increased hidden size. However, the effect seems to decrease when the hidden sizes reach a certain level. The LSTMs seem to be saturated with parameters with hidden sizes of 256 or 512. The number of layers tells a similar story. A positive effect of additional layers only seems to exist very clearly with hidden sizes of 64. This suggests that the examined hyper-parameters have covered the optimal range to a large degree.

Collapsing Strategies The collapsing strategy "only last" renders by far the worst results, quantitatively, as well as qualitatively. The introduction of bi-directionality mitigates the difference to some degree, as expected, but by no means removes it. The reason for this probably lies in the number of available parameters. The other two collapsing strategies have a total of $N_V * N'_A$ parameters to extract the N_V coordinates from, while the "laststep" strategy only has the N'_A parameters of the last LSTM step to extract the coordinates from. Especially with values for N'_A that are smaller than or near N_V, this means that there is even less than one neuron available per coordinate that must be computed. The very high value of the "vertexbased" strategy with a hidden size of 128 and 2 layers could be explained by a lack of available parameters, too. As mentioned above, until hidden sizes of 256 or even 512, increasing the hidden size improves the performance in the "allsteps" models. The "vertexbased" strategy limits the LSTMs by removing the capability of using a mixture of all steps to extract the coordinates. As a result, it seems that models with the "vertexbased" strategy need more than hidden sizes of 128 to produce sensible results. As described in Section 4.2.3, the main difference between the "allsteps" and the "vertexbased" models lies in the included frequency spectra of the output roads. The "vertexbased" models do not seem to pick up the general courses of the input roads but follow them very strictly and introduce random noise. Despite weakly pronounced smoothing, no indications of acquired and applied generalisation operators can be observed. The "allsteps" models, on the other hand, show signs of smoothing (Figure 4.13, Road 2), simplification (Figure 4.16, Road 2), and exaggeration (Figure 4.14, Road 1) operators. However, often, the predictions of the "allsteps" models seem rather over-smoothed (Figure 4.13, Road 1). An explanation for the bad predictions of the "vertexbased" and the "allsteps" LSTMs could be that they could not fully converge due to the fact that the training was stopped too early.

Bi-Directionality As described in Section 4.1.3, the introduction of bi-directionality only improves the performance of the models with the "laststep" strategies. This outcome was unexpected, especially regarding the models with the "vertexbased" strategy. Their LSTM cells and thus, their resulting vertices, only receive information from the previous LSTM cells and vertices. Not only are the metrics invariant to the bi-directionality, but there is also no detectable asymmetry regarding the outputs of the uni-directional models.

Predicted Deformation Models with the "laststep" collapsing strategy predicted the highest deformation among the different strategies. The reason for this is likely the same as for the simple CAEs: A lack of high frequencies within the predictions. The two metrics suggest different conclusions about the other two strategies. Regarding the FD, the "allsteps" models predict less deformation, while according to the rABC, the "vertexbased" models predict less deformation. A possible explanation could be that although qualitatively, the "vertexbased" models follow the input lines more closely, their noisy predictions produce local extreme deviations that the FD picks up.

General Remarks The generalised representations of the input roads that are generated by LSTMs are sensible to some degree. However, the smoothing operator adapted by the LSTMs is the first and foremost generalisation operator. High frequencies are successfully removed. Bends that are too narrow to pass the low-pass filter are removed if they are small enough such that their removal does not lead to a significant change in the overall course of the road, which could imply a simplification operator. If narrow bends are too large to be removed, they are often enlarged and exaggerated, which implies an exaggeration operator. However, the output and quality depend

largely on the chosen collapsing strategy. The acquired information about the roads in the last LSTM cells does not suffice to generate results with high frequencies. The outputs generated by models that use all the LSTM cells to produce their respective vertices generate no sensible outputs and introduce random noise. Only the models that allow the LSTM cells to exchange information with fully connected layers achieve reasonable outputs.

5.2 Observations Regarding All Architectures

Fourier Loss Extension The Fourier Loss Extension did not results in any major differences regarding any of the two metrics in all the architectures. In fact, since these differences were so small, the question remains if they emerged by chance. This finding was unexpected, as the comparison of coordinates is vastly different than the comparison of the parameters of simple periodic functions. However, this finding could lead to the important conclusion that the models are not limited by what differences their loss functions are able to detect, but by the inner workings and limits of their architectures. The vertices in the GCNNs, for example, could not gain any information about vertices that were further away than 10 neighbours at most. Thus, they did not have the *general* capability of finding suitable locations to apply the exaggeration operator.

Effect of Difficulty Figure 4.6.a shows that the performance of all models decreases with more difficult roads. Thus, the models seem to perform worse when the ground truth suggests a large deformation. In addition to the general effect, the GCNN and the RNN seem to perform worse on many "easy" roads that the CAE perform well on. Looking at the roads in the Figures 4.7 - 4.16, it becomes apparent that swisstopo's low-deformation generalisations are often only characterised by the smoothing operator, which all models could learn to some degree. As the actual deformations get larger, two effects seem to emerge: First, the operators simplification and exaggeration start to emerge, but second, the differences in the proposed deformations by swisstopo sometimes get erratic even to the human eye. These explainable differences could emerge by the "cartographic license" of swisstopo's cartographers, but more likely in most cases, they are due to a lack of spatial context. For example, when there are already many map elements such as buildings or elevation contours displayed on the map, cartographers may decide to remove more high frequencies than when the roads are the only dominant map features.

Similarities with Smoothing Filters As mentioned above, all the architectures apply low-pass filters to the inputs to remove the high frequencies in the roads. However, there is a distinction between the predictions of the GCNN and the predictions of the CAE and the RNN. The predictions of the GCNN generally stay "within" the bends of the input roads and thus are reminiscent of the products of smoothing algorithms that use the mean of moving kernels. A representative of such a filter is the Gaussian smoothing filter. The predictions of the other two architectures, especially the RNN, seem to prohibit high frequencies in their outputs generally, leading to signs of exaggeration operators if the Gaussian filtered roads' frequencies would consist of too high frequencies. The outputs are similar to the products of the Savitzky-Golay-Filter with a low polynomial order. An example of this difference can be seen in Road 2 of Figure 4.16. The bend on the left side is exaggerated only by the CAE and the RNN.

Discussion of Loss Curves Generally, all the loss curves in Figures 4.17 - 4.18 show that the models have been able to learn. They all show a steep decline at first, with a flattening after a certain number of epochs. Note that the shape of the loss curves is distorted by the logarithmic scales, which makes the incline of the curves at lower magnitudes seem to be greater than it really is. The observation discussed in this paragraph is that the evaluation loss values of the GCNNs and the RNNs with the collapsing strategies "vertexbased" and "laststep" are lower than their training loss values. Usually, the opposite is expected because the models should perform better on the data they have been trained on. From this observation it can be deduced that, due to a too small sample size and the resulting influence of chance, the training and the evaluation sets are not sampled from the same distribution. The evaluation set probably contains a larger proportion of easy cases or easier cases in general. However, not all models perform better on the validation set. The CAEs and the "allsteps" variants of the RNNs perform better on the training set, which would be expected. These three (sub-)architectures are the (sub-)architectures with the lowest predicted dissimilarity. The roads in the validation set thus may be characterised by more significant deformations so that the high-deformation models could obtain a lower loss value.

Chapter 6

Conclusion

6.1 Contributions

This thesis has explored the potential of three different Deep Learning architectures in their application in automated deformative cartographic road generalisation using a vector-based approach. Thereby, relevant procedures and workflows have been developed:

Pre-Processing The chosen methodology proposed a workflow to pre-process multi-scale cartographic road data to make it compatible with the implementations of different Deep Learning architectures of the latest Python Deep Learning frameworks. The pre-processing involved extensive filtering to account for unsuitable input-target combinations, which arise since the used data has been produced with a sole cartographic purpose in mind. The thesis further proposes a method to pre-process the geometry of the roads such that they have regularised orientations and polarities. To enrich the available data beyond mere coordinates, two additional geometric attributes have been computed, which embed each vertex in the local and global context of its road.

Evaluation The results of the Deep Learning models have been evaluated both quantitatively and qualitatively. The thesis proposes a way to quantitatively determine the models' performance using two metrics to assess the difference between two curves (output and target). The application of the *Fréchet Distance* enabled the assessment of occurring local deviation maxima while the *Area between Curves* described the similarities and differences globally. Using these two metrics, the thesis provides valuable insights about adequate hyper-parameter configurations and differences regarding the three used architectures. The qualitative evaluation examined the models' ability to induce three generalisation operators from the cartographic ground truth samples and produced knowledge about the advantages and shortcomings of the different architectures.

6.2 Insights

Applied generalisation knowledge Most of the time, the models generated meaningful small-scale representations of the input roads. However, the outputs vary significantly from swisstopo's ground truth. The most remarkable difference across the architectures lies in the frequency spectra of the output roads, which contribute considerably to the models' ability to apply different generalisation operators. All the models successfully learned and applied the smoothing operator, while only the CAE and the RNN could acquire the simplification and the exaggeration operator.

U-Nets performed best The U-Net variants were the overall best-performing models. The U-Net variant with the worst quantitative performance still performed better than the best model of the other architectures. The loss curves of the U-Nets converged faster than the loss curves of the other models, and the U-Nets were also the only models that had the capacity to over-fit on the training data. In the qualitative evaluation, the predictions of the U-Net variants mimicked swisstopo's generalisations most accurately in terms of their shape and amount of deformation.

Fourier Loss Extension has only minor effect The Fourier Loss Extension did not cause any significant differences regarding any of the two metrics in all the architectures. The small effect size suggests a performance limitation by the model architectures and not by the loss function.

Mismatch between quantitative and qualitative performance Often, there was a substantial difference between the interpretation of the quantitative and the qualitative results. In many cases, there is not only one "correct" generalisation of a road within a given context. However, the quantitative measures only assessed the metric differences to one specific target, which is heavily influenced by the "cartographic license" of swisstopo's cartographers and by non-available spatial context. Thus, a low quantitative performance does not necessarily imply a bad generalisation.

More spatial context is needed A large proportion of the generalisation decisions of the swisstopo cartographers, represented in the target roads, cannot be understood without additional spatial context, even by knowledgeable geographers. The models thus were facing a challenging task when trying to mimic these seemingly erratic targets. A further advantage of introducing more spatial context would be the possibility of accounting for the displacement operator.

6.3 Limitations

The conducted work and the outcomes were limited by several decisions that have been taken to reduce the scope of this thesis. Many of these decisions resulted from a workflow that gradually evolved over time. The following limitations are considered to be the most important:

- **Spatial Context**
  The most considerable limitation of the models was the lack of spatial cartographic context. First, no other map feature class has been taken into account, and second, only single road segments have been sampled, not contiguous road networks.
- **Sample Sizes**
  The conducted filtering steps removed a substantial portion of the roads. The used thresholds, although they were supported by histograms, were arbitrary to some degree. The sample size may have been just above the bottom edge of the necessary size. Larger training and validation sets would more probably have been drawn from the same distribution, thereby mitigating the problem that the validation loss was lower than the training loss in some cases. The extensive filtering of the input roads has further removed much of potential cartographic knowledge. Less conservative thresholds would have increased the sample size to a large degree.
- **Training Resources**
  The model training was also limited in terms of computational and temporal resources. As a result, it cannot with certainty be concluded that the models would not have performed better if more time had been given for the training.
- **Hyper-Parameters**
  It is unclear whether the performance could have been improved with more extensive hyper-parameter configurations. Perhaps more layers, larger hidden sizes, or normalised inputs would have further enhanced the models' performance.
- **Vertex Density**
  The input roads of the GCNN differed vastly from the input roads of the other two architectures. The interpolation of the inputs of the CAEs and the RNNs by a fixed number of vertices led to a much higher vertex density than in the inputs of the GCNNs.
- **Attributes**
  Only geometric attributes solely derived from the coordinates have been used as inputs to the models. More elaborate geometric or qualitative features, such as the road types or surfaces, may have given the models important information to parameterise the deformations.

6.4 Outlook

Spatial Context First, further research about the inclusion of the spatial context of roads in vector-based Deep Learning models is needed. Thereby, questions about how the models react to other map features such as buildings could be addressed. The inclusion of additional map features in raster-based approaches could be achieved by adding channels to the input images. However, vector-based approaches — that this thesis engages with — often imply that the map features

are specified as *entities*, which would add complexity. A strategy to circumvent the need to add discrete entities to the models would be to add features to the vertices that quantify their relative proximity to additional map feature classes. In the case of polygon map features, such as buildings, this could be achieved by reducing the footprints to their centroids and calculating the distance to the nearest building for each vertex. Additional features could indicate the type of the nearest building (e.g. residential building or church) and whether it is located within a group of buildings. In the case of linear map features, such as rail tracks or rivers, the distance to their nearest vertex and the interior angle between the tangents at these vertices could be computed.

Effect of Geometric Transformations Second, more research on the effect of different geometric transformations is needed. This thesis proposes a method to regularise the shapes of the roads. However, no research about the effectiveness of the applied transformations has been conducted. By comparing the performance of differently pre-processed inputs, a deepened understanding of the importance of the input structure could be gained. In addition, further research could include non-cartesian coordinate systems, such as the polar coordinate system. Possible origins for these coordinate systems could be the starting vertices or the centre of mass of the roads.

Bibliography

Aronov, B., Har-Peled, S., Knauer, C., Wang, Y., and Wenk, C. (2006). Fréchet distance for curves, revisited. In *Algorithms–ESA 2006: 14th Annual European Symposium*, pages 52–63. Springer.

Beard, K. (1991). Constraints on rule formation. In Buttonfield, B. P. and McMaster, R. B., editors, *Map Generalization: Making Rules for Knowledge Representation*, pages 121–135. Longman, London.

Brassel, K. E. and Weibel, R. (1988). A review and conceptual framework of automated map generalization. *International Journal of Geographical Information System*, 2(3):229–244.

Bringmann, K., Künnemann, M., and Nusser, A. (2019). Walking the Dog Fast in Practice: Algorithm Engineering of the Fréchet Distance. *Journal of Computational Geometry*, 12(1):70–108.

Chen, M., Wei, Z., Huang, Z., Ding, B., and Li, Y. (2020). Simple and Deep Graph Convolutional Networks. In *Proceedings of the 37th International Conference on Machine Learning*, pages 1725–1735. Proceedings of Machine Learning Research.

Courtial, A., Ayedi, A. E., Touya, G., and Zhang, X. (2020). Exploring the potential of deep learning segmentation for mountain roads generalisation. *ISPRS International Journal of Geo-Information*, 9(5):338–341.

Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., and Bharath, A. A. (2018). Generative Adversarial Networks: An Overview. *IEEE Signal Processing Magazine*, 35(1):53–65.

Deng, L. (2012). The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141 – 142.

Douglas, D. H. and Peucker, T. K. (1973). Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122.

Du, J., Wu, F., Xing, R., Gong, X., and Yu, L. (2021). Segmentation and sampling method for complex polyline generalization based on a generative adversarial network. *Geocarto International*, 37(14):4158–4180.

Du, J., Wu, F., Yin, J., Liu, C., and Gong, X. (2022). Polyline simplification based on the artificial neural network with constraints of generalization knowledge. *Cartography and Geographic Information Science*, 49(4):313–337.

Eiter, T. and Mannila, H. (1994). Computing discrete Frechet distance. *Technical Report CD-TR 94/64, Christian Doppler Laboratory for Expert.*

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):258–258.

Fey, M. and Lenssen, J. E. (2019). Fast Graph Representation Learning with PyTorch Geometric. *CoRR*. arXiv: 1903.02428.

Golmant, N., Vemuri, N., Yao, Z., Feinberg, V., Gholami, A., Rothauge, K., Mahoney, M. W., and Gonzalez, J. (2018). On the Computational Inefficiency of Large Batch Sizes for Stochastic Gradient Descent. *CoRR*.

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

Haykin, S. (2009). *Neural Networks and Learning Machines*. Prentice Hall.

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

Hochreiter, S. (1998). The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 06(02):107–116.

Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Jadon, A., Patil, A., and Jadon, S. (2022). A Comprehensive Survey of Regression Based Loss Functions for Time Series Forecasting. arXiv:2211.02989 [cs].

Jekel, C. F., Venter, G., Venter, M. P., Stander, N., and Haftka, R. T. (2019). Similarity measures for identifying material parameters from hysteresis loops using inverse analysis. *International Journal of Material Forming*, 12(3):355–378.

Jha, D., Smedsrud, P. H., Riegler, M. A., Johansen, D., de Lange, T., Halvorsen, P., and Johansen, H. D. (2019). ResUNet++: An Advanced Architecture for Medical Image Segmentation. arXiv:1911.07067 [cs, eess].

Lawford, G. J. (2007). Fourier Series and the cartographic line. *International Journal of Geographical Information Science*, 20(1):31–52.

Le, P. and Zuidema, W. H. (2016). Quantifying the Vanishing Gradient and Long Distance Dependency Problem in Recursive Neural Networks and Recursive LSTMs. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 87–93. Association for Computational Linguistics.

Lecun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

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):309–309.

Li, H., Li, J., Guan, X., Liang, B., Lai, Y., and Luo, X. (2019). Research on Overfitting of Deep Learning. In *15th International Conference on Computational Intelligence and Security*, pages 78–81. Institute of Electrical and Electronics Engineers Inc.

Mackaness, W. A. and Beard, K. M. (2013). Use of Graph Theory to Support Map Generalization. *Cartography and Geographic Information Systems*, 20(4):210–221.

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*, 36(4):639–673.

Michelucci, U. (2022). An Introduction to Autoencoders. arXiv:2201.03898 [cs].

Müller, M. (2007). Dynamic Time Warping. In *Information Retrieval for Music and Motion*, pages 69–84. Springer, Berlin, Heidelberg.

O'Shea, K. and Nash, R. (2015). An introduction to convolutional neural networks. arXiv:1511.08458.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Wallach, H., Larochelle, H., Beygelzimer, A., d Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Petzold, I., Burghardt, D., and Matthias, B. (2006). Workflow management and generalisation services. In *Workshop of the ICA Commission on Map Generalisation and Multiple Representation*, Portland, USA. International Cartographic Association.

Pintelas, E., Livieris, I. E., and Pintelas, P. E. (2021). A Convolutional Autoencoder Topology for Classification in High-Dimensional Noisy Image Datasets. *Sensors (Basel)*, 21(22):7731.

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. (1998). A method for building displacement in automated map generalisation. *International Journal of Geographical Information Science*, 12(8):789–803. Publisher: Taylor & Francis Group.

Saalfeld, A. (1985). A fast rubber-sheeting transformation using simplicial coordinates. *The American Cartographer*, 12(2):169–173.

Sarker, I. H. (2021). Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions. *SN Computer Science*, 2(6):420.

Savitzky, A. and Golay, M. J. E. (1964). Smoothing and Differentiation of Data by Simplified Least Squares Procedures. *Analytical Chemistry*, 36(8):1627–1639.

Sester, M., Feng, Y., and Thiemann, F. (2018). Building generalization using deep learning. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-4:565–572.

Smagulova, K. and James, A. P. (2020). Overview of Long Short-Term Memory Neural Networks. In *Deep Learning Classifiers with Memristive Networks*, volume 14 of *Modeling and Optimization in Science and Technologies*. Springer, Cham.

Soydaner, D. (2020). A Comparison of Optimization Algorithms for Deep Learning. *International Journal of Pattern Recognition and Artificial Intelligence*, 34(13):2052013.

Spiess, E., Baumgartner, U., Arn, S., and Vez, C. (2003). Topographic Maps: Map Graphics and Generalisation Cartographic. In *Cartographic Publication Series*, volume 17. Swiss Society of Cartography, Wabern BE.

Taillandier, P., Duchêne, C., and Drogoul, A. (2011). Automatic revision of rules used to guide the generalisation process in systems based on a trial and error strategy. *International Journal of Geographical Information Science*, 25(12):1971–1999.

Tobler, W. (1966). Numerical map generalization. Technical report, Michigan Inter-University, Michigan, USA.

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.

Weibel, R. (1991). Amplified intelligence and rule-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 *GIS and Generalization: Methodology and Practice*, pages 56–69. Taylor & Francis, London.

Weibel, R. and Dutton, G. (1998). Constraint-Based Automated Map Generalization. *8th International Symposium on Spatial Data Handling 1998*, pages 214–224.

Weibel, R., Keller, S. F., and Reichenbacher, T. (1995). Overcoming the Knowledge Acquisition Bottleneck in Map Generalization: The Role of Interactive Systems and Computational Intelligence. In *International Conference on Spatial Information Theory*, pages 139–156, Semmering. Springer.

Werschlein, T. and Weibel, R. (1994). Use of Neural Networks in Line Generalization. In *European Conference on Geographical Information Systems*, pages 76–85, Paris.

Wu, L., Cui, P., Pei, J., and Zhao, L., editors (2022). *Graph Neural Networks: Foundations, Frontiers, and Applications*. Springer Nature Singapore, Singapore.

Yan, X., Ai, T., Yang, M., and Tong, X. (2020a). Graph convolutional autoencoder model for the shape coding and cognition of buildings in maps. *International Journal of Geographical Information Science*, 35(3):490–512.

Yan, X., Ai, T., Yang, M., Tong, X., and Liu, Q. (2020b). A graph deep learning approach for urban building grouping. *Geocarto International*, 37(10):2944–2966. Publisher: Taylor & Francis.

Yan, X., Ai, T., Yang, M., and Yin, H. (2019). A graph convolutional neural network for classification of building patterns using spatial vector data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 150:259–273.

Yin, X.-X., Sun, L., Fu, Y., Lu, R., and Zhang, Y. (2022). U-Net-Based Medical Image Segmentation. *Journal of Healthcare Engineering*, 2022.

Yu, W. and Chen, Y. (2022). Data-driven polyline simplification using a stacked autoencoder-based deep neural network. *Transactions in GIS*, 26(5):2302–2325.

Yu, Y., Si, X., Hu, C., and Zhang, J. (2019). A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures. *Neural Computation*, 31(7):1235–1270.

Zhang, S., Tong, H., Xu, J., and Maciejewski, R. (2019). Graph convolutional networks: a comprehensive review. *Computational Social Networks*, 6(1):1–23.

Zheng, J., Gao, Z., Ma, J., Shen, J., Zhang, K., Zheng, C. ., Gao, J. ., Ma, Z. ., Shen, J. ., Zhang, J. ., and Deep Graph, K. (2021). Deep Graph Convolutional Networks for Accurate Automatic Road Network Selection. *ISPRS International Journal of Geo-Information*, 10(11):768–768.

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, January 31, 2023

Nicolas Beglinger