Learning from vector data: enhancing vector-based shape encoding and shape classification for map generalization purposes

Martin Knura

To cite this article: Martin Knura (2024) Learning from vector data: enhancing vector-based shape encoding and shape classification for map generalization purposes, Cartography and Geographic Information Science, 51:1, 146-167, DOI: 10.1080/15230406.2023.2273397

To link to this article: https://doi.org/10.1080/15230406.2023.2273397

CARTOGRAPHY AND GEOGRAPHIC INFORMATION SCIENCE
2024, VOL. 51, NO. 1, 146–167
https://doi.org/10.1080/15230406.2023.2273397

Taylor & Francis
Taylor & Francis Group

OPEN ACCESS Check for updates

Learning from vector data: enhancing vector-based shape encoding and shape classification for map generalization purposes

Martin Knura

Lab for Geoinformatics and Geovisualization (g2lab), HafenCity University Hamburg, Germany

ABSTRACT
Map generalization is a complex task that requires a high level of spatial cognition, and deep learning techniques have shown in numerous research fields that they could match or even outplay human cognition when knowledge is implicitly in the data. First experiments that apply deep learning techniques to map generalization tasks thereby adapt models from image processing, creating input data by rasterizing spatial vector data. Because image-based learning has major shortcomings for map generalization, this article investigates possibilities to learn directly from vector data, utilizing vector-based encoding schemes. First, we enhance preprocessing methods to match essential properties of deep learning models – namely regularity and feature description – and evaluate the performance of Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Graph Convolutional Neural Networks (GCNN) in combination with a feature-based encoding scheme. The results show that feature descriptors improve the accuracy of all three neural networks, and that the overall performances of the models are quite similar for both polygon and polyline shape classification tasks. In a second step, we implement an exemplary building generalization workflow based on shape classification and template matching, and discuss the generalization results based on a case study.

ARTICLE HISTORY
Received 1 December 2022
Accepted 16 October 2023

KEYWORDS
Deep learning; map generalization; vector data; feature descriptor; shape classification; building generalization

1. Introduction

Given the high complexity of map generalization, it is no surprise that a human cartographer can still produce better generalized maps than its automated counterpart. Understanding relationships between map objects, and manipulating them across scale with respect to purpose, function, and granularity (Mackaness et al., 2014), requires a high level of spatial cognition, which is difficult to model in an automated process. Consequently, formalizing a wide range of complex, interplaying scenarios has been the bottleneck for rule-based and constraint-based approaches in recent years (Courtial et al., 2020; Feng et al., 2019; Touya et al., 2019). At this point, deep learning techniques could close this gap, as they have shown in numerous research fields that they could match or even outplay human cognition when knowledge is implicitly in the data, as it is the case in automated map generalization (Touya et al., 2019).

Map generalization is not just a straightforward simplification of geographic features, but a combination of different tasks and cognitive processes, so there are different neural network solutions conceivable for this process. For example, Heterogeneous Graph Neural Networks (Iddianozie & McArdle, 2021) are capable to learn spatial relationships between different types of geometries, while Deep Generative Neural Networks (Ruthotto & Haber, 2021) can create new realistic features from input data. Following the recent advances in automated map generalization, Reinforcement Learning (Arulkumaran et al., 2017) could also be a candidate, as it could use cartographic constraints to evaluate the output of a network.

All of these networks are very different in purpose and design, but they all need to encode spatial data in a hidden embedding space before they are able to utilize it. First experiments with deep learning applications in the field of map generalization adapt models from image processing, creating input data by rasterizing spatial vector data (Courtial et al., 2020; Feng et al., 2019; Touya & Lokhat, 2020). The results accomplished by the authors show that there is a high benefit of using the well-adapted deep learning models for image processing in the field of map generalization, but there are also some major shortcomings that come along with image-based learning.

First, the input data for maps is usually stored in geographic databases, so the workflow of map generalization is mainly applied to the vector format. Therefore, utilizing the advances in Computer Vision requires a realignment of this existing workflow. For

CONTACT Martin Knura martin.knura@hcu-hamburg.de

example, the process of map evaluation is a key component for controlling the generalization process and for assessing the generalization output, and is usually implemented using constraints (Stoter et al., 2014). Applying this to raster-based output requires reworking a lot of measures, and has important limitations (Courtial et al., 2022a), not to mention the problem that rasterization of vector data itself introduces uncertainty and fuzziness into the generalization process (Liao et al., 2012).

Second, learning from raster images is a clear limitation compared to the data richness and information of the underlying vector data (Courtial et al., 2021; Touya et al., 2019). In the aforementioned experiments, tiled image representations of the original vector data were used as the input for model training, so the information was reduced both spatially – while raising new problems like cross-tile continuity and border effects – and in terms of map content, when all explicitly provided object information is only implicitly converted into the image representation.

By contrast, learning from vector data – i.e. encode vector data directly to an embedding space – has the potential to preserve major parts of the well-established workflow of map generalization, and to utilize all information of the underlying data. The current downside is that there is only limited research done in this field until now, and a well-adapted scheme to encode vector data to the embedding space is yet to come for more complex geometries than points (Mai et al., 2022). Within the present research on polyline and polygon embedding, two approaches can be distinguished: ordered sequences and graph structures. Ordered sequences encode geometries as an ordered list of vertex locations, which can be fed into sequential models like Recurrent Neural Networks (RNN) (Veer et al., 2018), or into one-dimensional Convolutional Neural Networks (CNN) (Liu et al., 2021; Veer et al., 2018; Yang et al., 2022). The other approach converts the vertex locations into a graph structure, and feeds this graph into a respective neural network like a Graph Convolutional Autoencoder (GCAE) (Yan et al., 2021).

In this paper, we apply these vector-based approaches to the workflow of map generalization, and discuss their suitability to enable generalization operations. In particular, we propose a workflow to adequately represent map data in deep learning models, and demonstrate the capability of our approach by implementing an exemplary building generalization workflow based on shape classification and template matching.

As the first step, we tackle the two major challenges for encoding vector data: regularity and feature description. The first challenge is to represent irregular spatial formats in a regular manner – which is an essential property of deep learning models – and requires regularity both between and within objects. For comparison, raster data regularity is straightforward: the size of each pixel is fixed (regularity within the object), as well as the size of each image (regularity between objects). On the other side, vector-based lines and polygons need normalization in aspects such as scale, the number of vertices, and the length of the edges between them. Because generalization operations in cartography are mainly evaluated in relation to the original geometry of an object, we introduce a method that minimizes the geometrical manipulation during preprocessing compared to the transformations conducted in the aforementioned studies.

The same variety of solutions can be found for the second challenge, which is feature description. Some models learned directly from point coordinates, while others utilized geometric descriptions for local, regional, and global features such as turning angles, centroid distances, or minimum bounding rectangles. These geometric descriptors are comparable to shape description and representation techniques as part of the constraint-based map generalization workflow (e.g. see Blana & Tsoulos, 2022) and can be used as an interface to include parts of the map generalization knowledge into the learning process. Accordingly, we utilize shape measures from the map generalization domain to expand the method of vertex feature extraction by Yan et al. (2021) and show that feature description not only benefits the learning of Graph Convolutional Neural Networks but also when applied to RNNs and CNNs. Furthermore, we are able to transfer the whole encoding process from polygon to polyline data.

As the second step, we implement our workflow of vector-based deep learning as part of an exemplary building generalization based on shape classification and template matching. First, we train different network models with 5000 buildings of 10 predefined shape classes. Second, the trained models predict the shape class for each building in the study area, and a normalized standard shape of the predicted class is matched to the original building through geometrical operations such as stretching, flattening, rotating, and translating.

The aim of implementing end-to-end map generalization using deep learning is to mimic human behavior and decision-making in complex generalization situations, in which human cartographers have developed rules and skills for abstraction through a set of generalization operations (e.g. simplification, aggregation, exaggeration). These rules are mainly related to the geometries of objects, so representing map data in deep learning models is the first step toward vector-based map generalization (Sester, 2020). Accordingly, this paper focuses mainly on the adequate representation of the vector data, and less on the cartographic decision-making process, which will be the main topic of our future work. In summary, this paper contributes to the ongoing research by:

- Identifying vector-based schemes from the literature that can encode lines and polygons into an embedding space
- Proposing a workflow of vertex feature description which utilizes shape description and representation techniques, and which is applicable to all aforementioned networks (CNN, RNN, GraphCNN) and geometry types (polygons and polylines)
- Investigating the impact that encoding properties such as encoding scheme, normalization, regularity, and feature description have on the outcome of the classification models
- Demonstrating the capability of feature encoding by implementing an exemplary building generalization workflow based on shape classification and template matching

The following work is structured as follows: Based on recent literature, we identify different encoding schemes for vector data (Section 2). We then present our enhancements to the geometry encoding before introducing the different neural networks we utilize (Section 3). Based on this, we conduct experiments to test the performance of these networks and encoding schemes (Section 4) and present the results (Section 5). Next, we conduct a case study for building generalization (Section 6), followed by a discussion of the overall results and its impact on map generalization (Section 7) before summarizing our work (Section 8).

2. Related work

Utilizing machine learning techniques for tasks of the generalization workflow is not a novel phenomenon. Following the work of Weibel et al. (1995) about options for overcoming the knowledge acquisition bottleneck, machine learning in map generalization was mainly explored for data enrichment, knowledge acquisition, and map evaluation (Touya et al., 2019). With the recent advances in the field of deep learning in general, and Computer Vision in particular, a surge of numerous publications employing deep learning techniques to generalization-related tasks have been initiated in the last years, contributing to solve the aforementioned formalization problem of the rule- and constraint-based approaches.

2.1. Deep learning for map generalization tasks

Regarding point generalization, Karsznia and Sielicka (2020) used a decision tree with genetic algorithms for settlement selection, and Karsznia et al. (2022) compared the findings with other methods including a feed-forward neural network with backpropagation. Line selection was done by Zheng et al. (2021) using a deep Graph Convolutional Neural Network (GCNN). Courtial et al. (2021) generalized mountain roads using a U-net architecture for their CNN, and Du et al. (2022) simplified polylines using a Generative Adversarial Network (GAN), and a property-based neural network with controllers (Du et al., 2022). Work on polygon generalization mainly focuses on buildings, also using U-nets and GANs (Feng et al., 2019; Kang et al., 2020). As an alternative, Yang et al. (2022) employed a different approach and utilized a backpropagation network to decide which simplification algorithm to use for the best result.

2.2. Location encoding and encoding schemes

The majority of these end-to-end generalization models created raster images from vector data and learned how to create generalized output from these visual representations of the original data. The spatial location of the objects of interest is thereby encoded through the positions of the respective pixels in the image, and each pixel has one or more vectors that represent its information (e.g. three vectors for RGB images). These 3D-input can be further enriched by adding vectors to the pixels which represent additional information useful for map generalization, as shown by Courtial et al. (2022b).

Compared to the regular grid structure of raster data, the direct encoding of more irregular vector data is more complicated, and there is only limited research done which can be utilized for the process of map generalization. Mai et al. (2022) provide a general framework for understanding the current landscape of location encoding research while focusing on *point set* data. The problem thereby is that this framework is not straightforwardly applicable to other geometries such as lines and polygons, as point cloud data is position invariant, and methods like PointNet (Qi et al., 2016) or Pointer Networks (Vinyals et al., 2017) learn the same representation for a point set, regardless of the order of points within the set. In contrast, the order of points is crucial for vector data representations of lines and polygons, and the approaches to encode these geometries directly to the embedding space can be divided into two directions: ordered sequences and graphs (see Figure 1).

The first approach is to encode a line or the edges of a polygon as an ordered sequence of vertices. Based on the work of Ha and Eck (2017) on neural representations of sketch drawings, Veer et al. (2018) proposed a polygon encoding scheme as a sequence of five-dimensional vectors, concatenating a two-dimensional vector representing the normalized point coordinates and a three-dimensional one-hot vector that describes the function of the vertex in relation to the geometry. The authors then feed this sequence both into an RNN and a 1D-CNN for various classification tasks. Liu et al. (2021) execute a Deep Point Convolutional Network (DPCN) directly on an ordered list of two-dimensional point vectors, using a *TriangleConv* operator which generates a feature representation of the respective point and its local neighbor points, and utilize their model for shape classification. Yang et al. (2022) represent lines as a series of *lixels*, which have a fixed length and contain information about the shape of the respective line segment in the form of a three-dimensional one-hot-vector (i.e. [0 1 0]) based on Grid Context Descriptors (Fan et al., 2021), and used them to segment administrative boundary lines based on their shape characteristics using a 1D-U-net.

The second approach converts polygons into a graph representation, and encodes this graph into the embedding space. It was proposed by Yan et al. (2021) and constructs a graph through connected vertices, which provide information of local, regional, and global shape features. Local features thereby describe characteristics of the vertex and its adjacent neighbors, while regional features also take the geometry's centroid into account. In their study, the authors apply their approach for shape coding and building cognition using a GCAE.

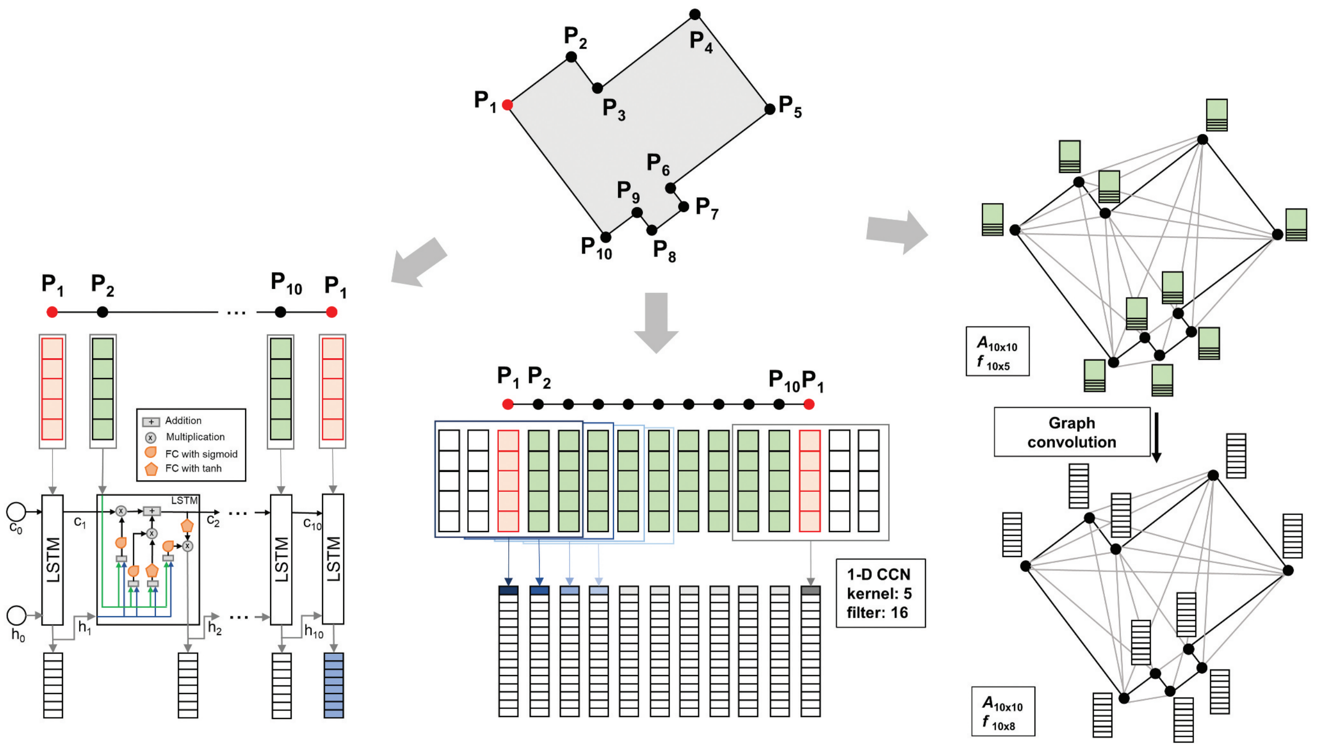

Figure 1. Exemplary approaches for encoding a polygon with ten vertices and five features per vertex, with the first layers of the respective neural networks. For the detailed architectures of the whole networks, see section 3.2. Left: encoding with an ordered sequence using a LSTM-RNN (see Veer et al., 2018). Starting with point P_1, the features of each point pt are fed as a 5D-vector into the LSTM cell, together with the state of the memory (c_{t-1}) and the output (h_{t-1}) of the previous step. For classification tasks, the output of the last state is the overall output of the layer. Center: encoding with an ordered sequence using a 1D-CNN (see Veer et al., 2018). Using a kernel size of five and padding, the sliding window moves over the sequence and produces the output (in blue shades) until the end of the sequence (gray output), before the same process is repeated for the next filter. Right: polygon encoding using a graph representation (see Yan et al., 2021). Each geometry is represented as a single graph, where the vertices of the geometry are the nodes (with the respective features) and their connecting lines are the edges. The adjacency matrix A describes if two nodes are actually connected (in black) or not (gray) as parts of the geometry boundary.

3. Method

Representing map data in neural networks is one of the major challenges in cartographic generalization when implementing deep learning. The previous section revealed different approaches to encode geometries, namely ordered sequences and graphs. We show how the different approaches can complement each other and introduce an enhanced geometry encoding (Section 3.1) suitable for different network types such as 1D-CNN, RNN, and GraphCNN (Section 3.2).

3.1. Geometry encoding

In this section, we propose an approach to encode polygons and polylines using vertex feature description, which is based on the work on graph representations of Yan et al. (2021). We enhance the encoding process in a way that it can be best utilized in automated map generalization by fulfilling certain requirements:

- The encoding process is reversible and encode spatial information about the map object, e.g. in the form of coordinates
- Geometric manipulation during preprocessing is minimized
- Preprocessing does not conduct any vector-raster-transformation

Figure 2 presents the individual steps of the encoding workflow, which we will describe in detail in this section.

3.1.1. Regularity of input data

Deep learning models require a regularly structured and normalized input to perform in the best possible way. Spatial data usually does not have this regularity, and so it is key to preprocess the input geometries appropriately. To ensure that all inputs have the same number of input features, Veer et al. (2018) use a *padding* approach: They identify the geometry with the largest number of points and add zeros to all other geometries until they reach the same size. While this is the best way to maintain the original geometries, adding a large number of zeros often has a negative impact on the performance of neural networks. Yan et al. (2021) and Liu et al. (2021) use an *interpolation* method for preprocessing: They first perform the Douglas-Peucker algorithm to simplify the original data before using an equally spaced interpolation, resulting in a geometry with a predefined number of vertices and the same distance between them. As this harms our second requirement for map generalization utility – minimizing geometry manipulation during preprocessing —, we introduce an alternative method that maintains the original shape of the geometry by *iteratively dividing* the current longest edge with an extra vertex until the target

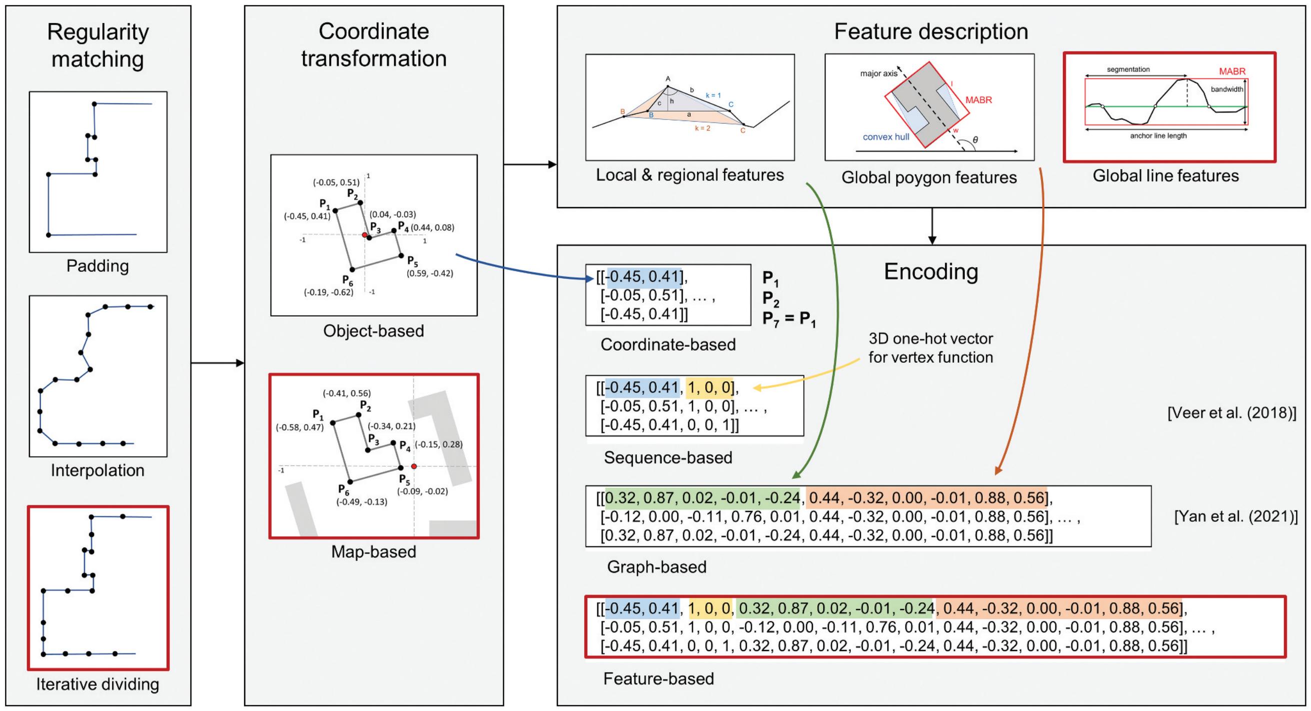

Figure 2. Steps of the encoding workflow with alternative methods and the resulting encoding sequence, which are described in detail in section 3.1. The *sequence-based* encoding refers to the work of Veer et al. (2018), while the *graph-based* encoding was proposed by Yan et al. (2021). The novel contributions of this paper are marked in red.

number of vertices is reached (see Figure 3). The downside is that this method does not ensure the inner regularity of the representation, as the distance between vertices could differ extremely. Because there is no optimal solution between these approaches, we will test them all in our experiments.

3.1.2. Coordinate normalization

The usual workflow for coordinate normalization can be described as object-based normalization. It consists of two steps: First, the whole geometry is relocated to the point of origin, and then scaled to a variance of about one. So for each point vector p, the normalized point vector p' can be calculated as:

$$p' = \frac{p - \bar{p}}{s} \qquad (1)$$

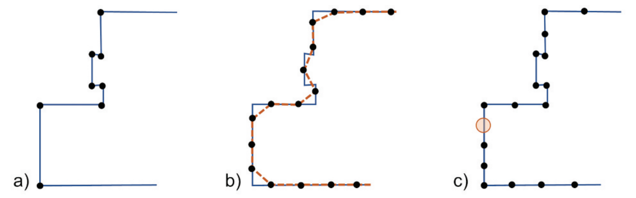

Figure 3. Describing a part of a geometry with 16 vertices. a) Original shape with eight vertices. b) Interpolation approach with equal distances between vertices, manipulating the geometry and resulting in the orange shape. c) Approach of iteratively dividing the longest edges until the target number of vertices is reached. Distances between vertices could differ; the next vertex would be placed at the position marked in orange, as this is currently the longest edge.

where $\bar{p}$ is the centroid of the geometry and s is the scale factor, which can be based on the z-score method (see Liu et al. (2021); Yan et al. (2021)) or the standard deviation over all objects' bounding boxes (see Veer et al. (2018)).

For the application of map generalization, it could be useful to use a normalization approach which contributes to the fact that all objects are located within a close range, and so we propose an alternative approach by introducing a map-based normalization. The main difference compared to the object-based method is that the centroid $\bar{p}$ used in Equation 1 is related to the whole map area (see Figure 4). We expect a negative impact on the performance of coordinate-based learning models such as the ones of Veer et al. (2018) and Liu et al. (2021) and will test the performance with map-based normalization in our experiments.

3.1.3. Feature extraction for encoding geometries

The most straightforward approach to represent a geometry is to use its exterior vertex coordinates (e.g. see Liu et al. (2021)). The disadvantage is that this encoding can only describe simple polygons. Veer et al. (2018) also use these point coordinates, but in combination with one-hot vectors marking the end of either the vertex [1 0 0], the sub-geometry [0 1 0], or the whole geometry [0 0 1]. This approach can also encode complex polygons with holes and multipolygons and can be applied to polylines. However, using only the point coordinates to represent vertices falls short of the rich cognitive information a shape boundary can provide, not mentioning the important role that Gestalt

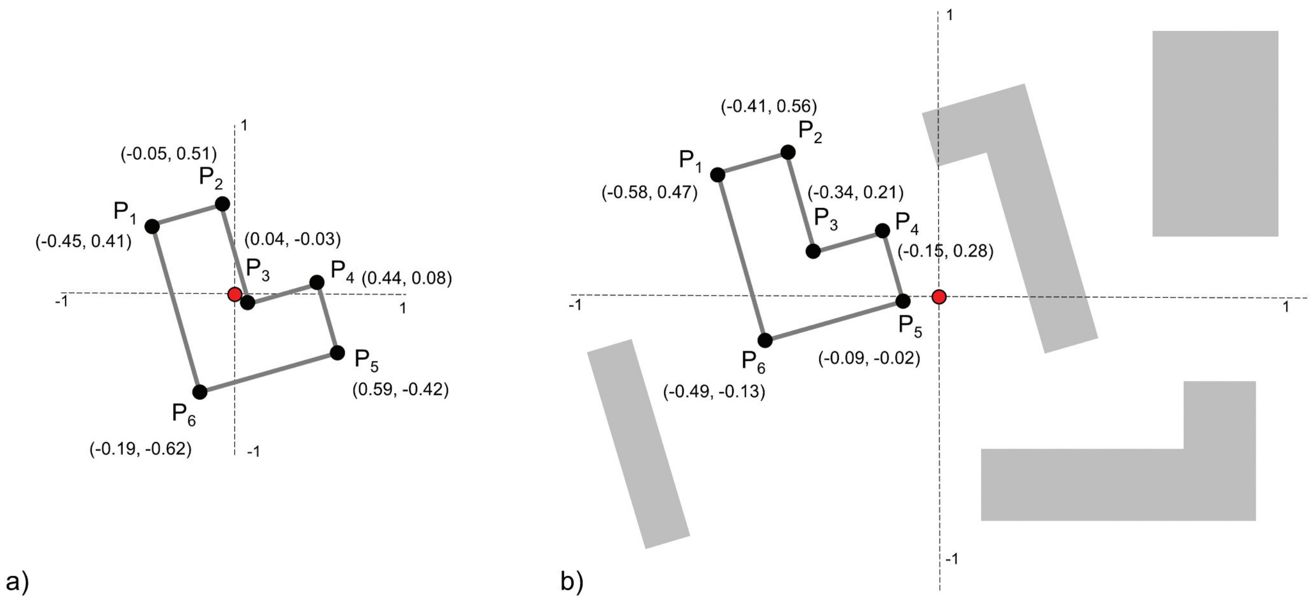

Figure 4. a) Object-based and b) map-based normalization. Respective centroids in red, map frame and other map objects in gray.

principles play in map generalization. For that reason, Yan et al. (2021) used specific indicators that describe local, regional, and global features instead of point coordinates as an input for their GCAE, and Liu et al. (2021) utilize local feature extraction as a part of their *TriangleConv* operator. Both approaches show improved results compared to conventional inputs in shape classification tasks. In the following subsections, we compose lists from the domain knowledge of map generalization with different basic features that can be used to describe the shapes of lines and polygons.

Features of the local structure describe the characteristics that appear at a certain point A and its adjacent neighbors B, C with a distance of k that form the triangle ABC (see Figure 5). These features can be described for all vertices in a polygon, and for all but the k-first and k-last vertices of a polyline. From the basic local features shown in Table 1, Yan et al. (2021) use triangle area, length of a and the turning angle as vertex features, while Liu et al. (2021) use all edge lengths and turning angles for their *TriangleConv* operator.

Features of the regional structure describe the general characteristics around the point of interest in relation to a significant regional point of the geometry (e.g. the centroid) while using the same feature calculations as for local features. For most lines, the centroid may not be the optimal reference point, and it could be beneficial to segment the line based on anchor line concurrences, inflection points, pattern types (Samsonov & Yakimova, 2017) or bend segments (Du et al., 2022; García Balboa & Ariza López, 2008). As an example, Yang et al. (2022) clip the lines for their *Unet* into smaller units, and describe each lixel based on the pattern type as either *smooth irregular schematic*, *sharp regular schematic* or *sharp irregular non-schematic*.

Features that describe the global structure of a geometry are calculated once, and the same value is applied to every vertex of the geometry. Although there are some features that can be applied to both lines and polygons, it is useful to differentiate between the two geometry types (see Figure 6). Table 2 lists the basic global line features, while Table 3 lists the basic global features for polygons. Parametric description of entire geometric features is thereby not new, but an elementary technique of map generalization to assess shape preservation Buttenfield (1991). Utilizing these features as information in the encoding process is therefore not only benefiting the learning process for encoder-decoder models (see Yan et al., 2021) but could also trigger and evaluate generalization operations in future end-to-end generalization models based on vector data.

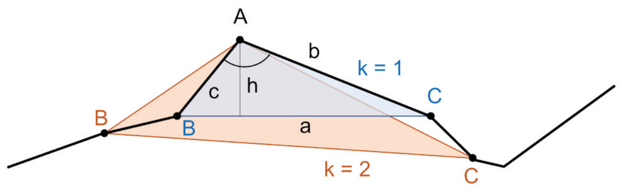

Figure 5. Local features at point A of a line or a part of a polygon. The blue triangle is formed with neighbors of $k = 1$, the red triangle with neighbors of $k = 2$.

Table 1. Basic local features based on a triangle ABC and A as the point of interest.

Feature	Description
Triangle area	Area of the triangle ABC
Length of c	Length of adjacent line BA
Length of b	Length of adjacent line AC
Angle	Turning angle at point A.
Length of a	Length of opposite line from point A (BC)
Height	Height of the triangle ABC

3.1.4. Encoding schemes

In summary, all encoding schemes have their advantages regarding their usability for map generalization. Using coordinates is mandatory when the results should be used in combination with map objects that are not generalized in the same model. Utilizing the three one-hot-vectors proposed by Veer et al. (2018) offers the possibility to encode complex polygons, and encoding additional features helps to describe specific characteristics of the geometries. Table 4 presents the three encoding schemes. The feature encoding is thereby an adapted version from the one of Yan et al. (2021), where we combine all the aforementioned properties in one scheme. We will investigate the performance of each scheme in the following experiments

3.2. Deep learning models

The aforementioned publications propose CNNs, RNNs, and GCNNs in combination with the different encoding schemes, and so we introduce each network in this section. Based on the findings of the recent literature, we adopt well-established models and the best setups available for each network type.

3.2.1. Convolutional neural networks (CNNs)

t'Veer et al. (2019) and Liu et al. (2021) both fed ordered sequences into one-dimensional CNNs to solve classification tasks. We follow this approach and utilize two CNN-based models. The first one is the original version of the CNN proposed by Veer et al. (2018). Figure 7 shows the framework of the *tVeerCNN* model. The input of the model is a sequence of length N, which is the number of vertices of the geometry, and with size E,

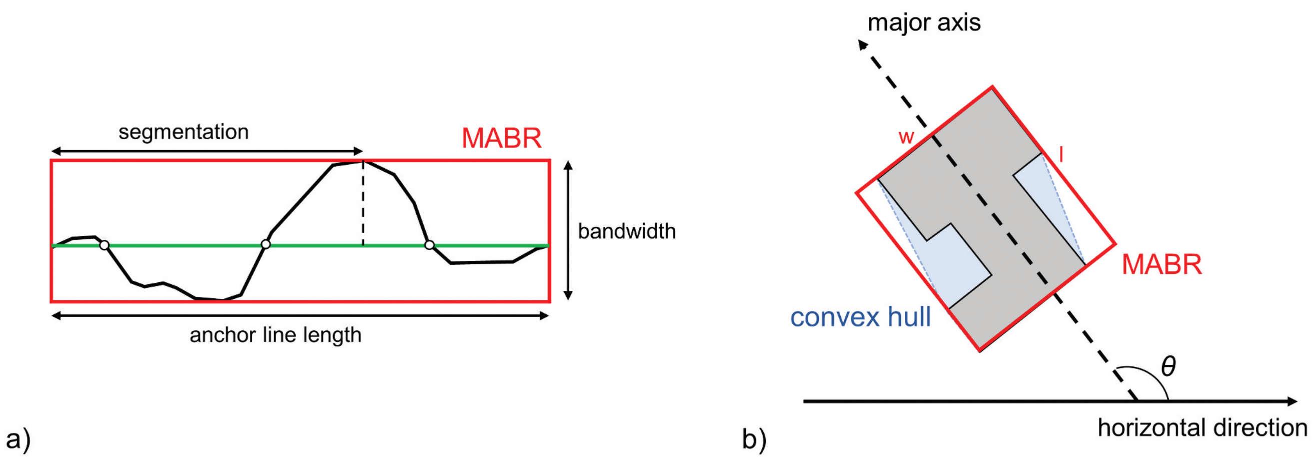

Figure 6. Selected global features for a) a line and b) a polygon. MABR is the minimum area bounding rectangle defining length (l) and width (w), or respectively anchor line length and bandwidth. Concurrence points are in white. Adapted from Buttenfield (1991) and Wang and Burghardt (2020).

Table 2. Basic global line features (adapted from Buttenfield, 1991; Du et al., 2022; García Balboa & Ariza López (2008); Samsonov & Yakimova, 2017).

Feature	Description
MABR	The minimum area bounding rectangle of the line
MABR area	The area of the MABR of the line
Anchor line length	The length of the MABR
Bandwidth	The width of the MABR
Segmentation	Distance from the beginning to the location on the anchor line where the maximum deviation occurs
WidthPos	Maximum deviation on one side of the anchor line
WidthNeg	Maximum deviation on other side of the anchor line
Concurrence	Number of times line crosses anchor line
Error variance	Discrete approximation of total discrepancy between line and anchor line
Bendings	Number of bends
Sinuosity	Number of inflection points
Directionality	Deviation between line length and anchor line length

Table 3. Basic global polygon features (adapted from Basaraner & Cetinkaya, 2017; Blana & Tsoulos, 2022; Wang & Burghardt, 2020; Yan et al., 2019).

Feature	Description
Area	Area of the polygon
MABR	Minimum area bounding rectangle
Elongation	Length to width ratio of the MABR
Circularity	Deviation between polygon and its equal-perimeter circle
Rectangularity	Deviation between polygon and its MABR
Squareness	Deviation between polygon and its equal-area square
Convexity	Deviation between polygon and its convex hull
Fractality	Edge roughness or smoothness
Orientation	Angle between major axis of the MABR and the horizontal direction

which reflects the number of vectors used by the encoding scheme. Each rectangle in the figure visualizes the shape of the output tensor after the respective layer. The representation learning part of the *tVeerCNN* is composed of two sequential one-dimensional convolutional layers with ReLU activation and a pooling layer, while the downstream part is composed of two fully

Table 4. Encoding schemes where x and y are the point coordinates, v_1, v_2, and v_3 are one hot vectors to describe the function of the vertex as of Veer et al. (2018), and f_l*, f_r* and f_g* are local, regional, and global feature descriptors of various size (*), adapted from Yan et al. (2021).

Abbr.	Description	Encoding Scheme
c	Coordinate encoding	[x y]
s	Sequence encoding	[x y v_1 v_2 v_3]
f	Feature encoding	[x y v_1 v_2 v_3 f_l* f_r* f_g*]

connected layers. The output of the model is a probability prediction for each of the K classes.

The second model we design adapts some parts of the *LiuDPCN* model of Liu et al. (2021), adding more depth to the layers. The overall framework of this model, which we will call *dCNN* ("deep CNN") in this paper, is the same as of the *tVeerCNN*, but with dimensions of 64-1024-256 instead of 32-64-32. Furthermore, we will use the *LiuDPCN* as a benchmark for the building shape classification task, although the design of the network prohibits the use of other inputs than point coordinates.

3.2.2. Recurrent neural networks (RNNs)

Similar to the one-dimensional CNNs, RNNs can process ordered sequences of vertices, and the basic model architecture we utilize is the one described by Veer et al. (2018). Figure 8 presents the framework of the *tVeerRNN*. The input is thereby fed into a bidirectional Long-Short Term Memory (LSTM), which is designed to process sequential data. Feature representation learning is done by passing the vertex input information, the cell state, and the output from one vertex to the next, while a trainable forget gate regulates which data are retained. Because of the bidirectionality, the network learns the representation of a vertex in relation to the vertices on

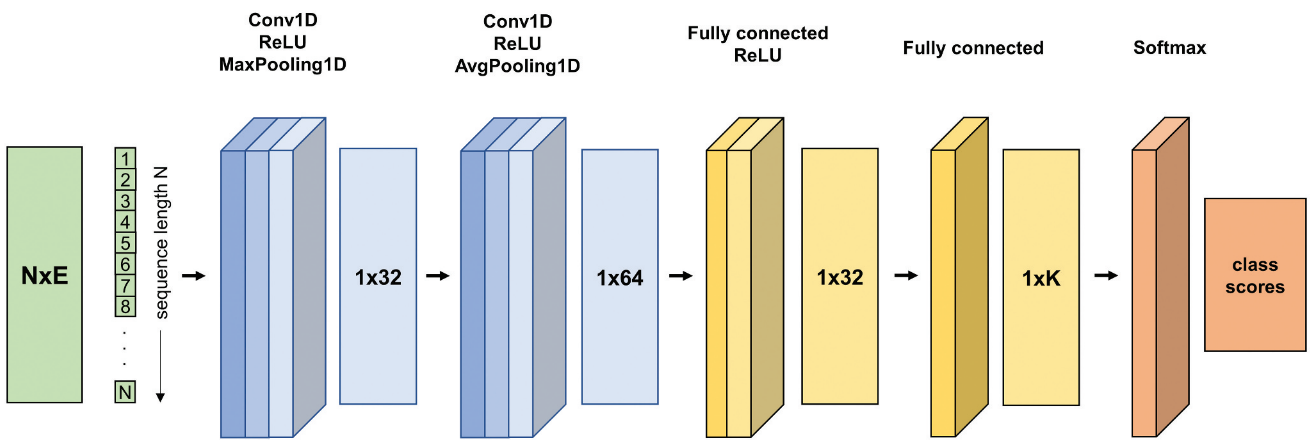

Figure 7. Framework of *tVeerCNN*. The input sequence of length *N* describes the vertex information using an encoding scheme of size *E*. Feature representation learning is done using two sequential 1-D-Convolutional layers with ReLU activation and Pooling (in blue color). The downstream part (in yellow) uses two fully connected layers and Softmax activation to produce a probability output for each of the *K* classes. Adapted from Veer et al. (2018) and Liu et al. (2021)).

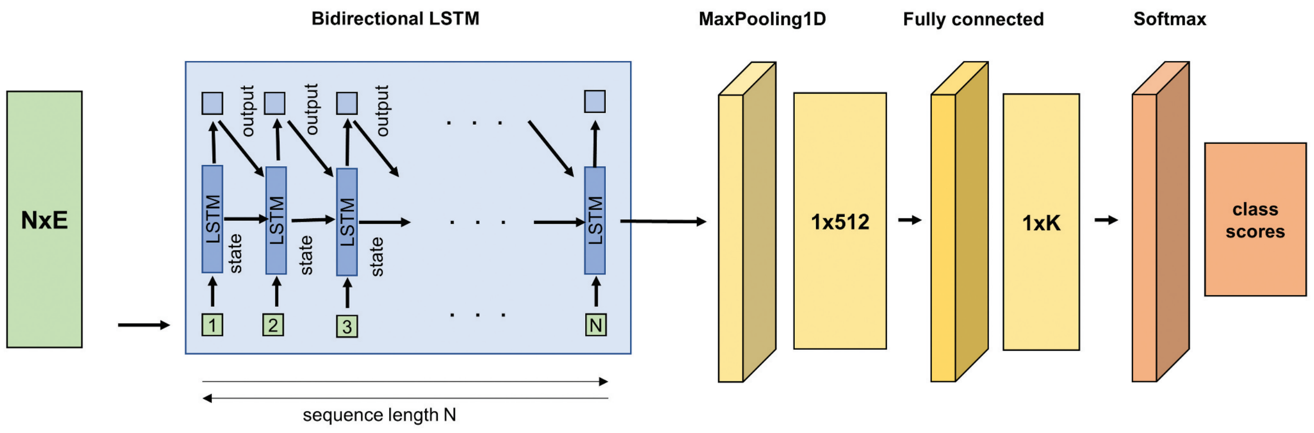

Figure 8. Framework of the *tVeerRNN*. The core of the model is a bidirectional LSTM, which is the feature representation learning module (in blue) and is sequentially fed with *N* inputs of size *E* twice – clockwise from 1 to *N* and anticlockwise from *N* to 1. The downstream part (yellow) is the same as in the *tVeerCNN* framework, with *K* class scores as the model output (red). Adapted from Veer et al. (2018).

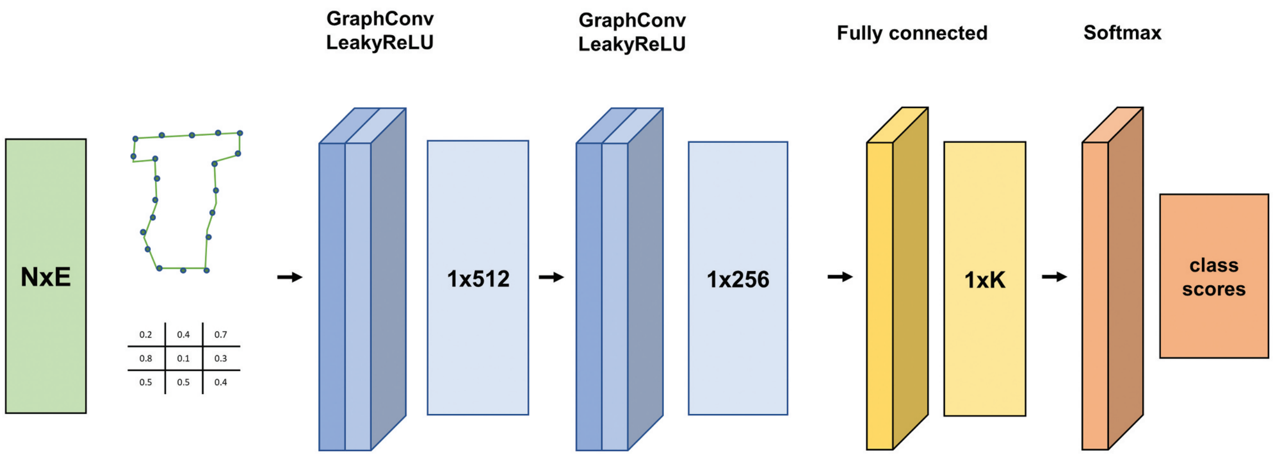

Figure 9. Framework of the GCNN with the dimension we use for our experiments. The input information *NxE* is processed into a graph structure, and the graph and its adjacency matrix is the input for the feature representation learning module (blue), which contains of two graph Convolutional layers with LeakyReLU activation. The downstream module (yellow) is the same as for the CNNs and RNNs, as well as the output of *K* class scores (red).

both sides. The output of the LSTM layer is its hidden state after the last input vertex is processed, which in our case has a combined dimension of 512. The downstream part of the *tVeerRNN* is the same as of the respective CNNs described above.

3.2.3. Graph convolutional neural networks (GCNNs)

The third network type we will test is the GCNN. Yan et al. (2021) proposed this network, and we adapt their model framework for our experiments. Figure 9 shows the architecture of the GCNN. The first step is to construct a graph and a respective adjacency matrix from the input features, which are fed into two sequential Graph Convolutional layers with a LeakyReLU activation. The remainder of the model is the same as for our CNNs and RNNs, with fully connected layers and a Softmax activation for the final class scores. Because the input dimensions of the different encoding schemes do not match the dimension of the proposed model of the authors, we will use the *YanGCNN* with the originally proposed graph features as our second benchmark model.

4. Experiments

In the previous section, we discussed the different approaches for preprocessing and presented the different encoding schemes as well as the neural networks. In this section, we conduct a row of experiments with the aim to:

- Evaluate the performance of the different encoding schemes in combination with CNNs, RNNs, and GCNNs
- Examine the impact of each preprocessing step (regularity matching, coordinate normalization) on the classification results
- Examine the impact of feature selection and encoding order on the classification results
- Apply the encoding techniques to a polyline data set

4.1. Task and data

To compare and evaluate the performance of the proposed encoding workflow in combination with different neural networks, we conduct experiments with shape classification as the downstream task for both polygon and polyline data. In both cases, the models have to predict the correct shape class of the input geometry. We selected this task for three reasons: First, learning from (encoded) geometries – which is the main topic of this paper – is essential for this task. Second, the results of this task can be quantitatively evaluated when labeled in the data set, but also qualitatively by human eye, which is particularly beneficial for classification edge cases. Third, this task in combination with the data set of Yan et al. (2021) has become a benchmark for shape coding, and so we want to utilize this setup for our work. This data set consists of 5010 buildings extracted from OpenStreetMap, which were manually labeled into classes according to their shape and their similarity to one of ten letters in the alphabet, namely E, F, H, I, L, O, T, U, Y, and Z. Figure 10 shows a variation of examples for the ten classes, which contain of 501 buildings each.

For polylines, we utilized a subset of the European Marine Observation and Data Network (EMODnet) data on coastal types. The data set consists of more than 20 different classes of coastal types, such as beaches, erodible rocks, estuaries, harbors, polders, and salt marshes. We selected three of these classes, namely "erodible rock and/ or cliff", "harbor area" and "sand beach fronting upland (> 1 km long)", and manually extract the most representative geometries for each type. Next, we cut longer elements at a maximum length of 8 km to make the classes more comparable, which finally leads to a total of 247 lines of type "harbor", 284 lines of type "erodible rock/cliff" and 329 lines of type "beach". We selected these three types because their shapes are visually distinguishable from a human perspective and can be classified at different corners of the line-shape-cube (Samsonov & Yakimova, 2017).

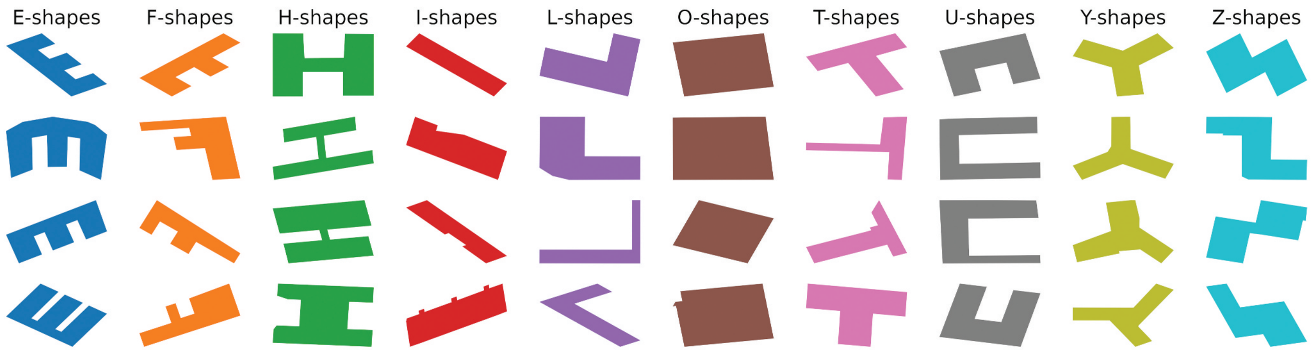

Figure 10. Examples for each class of building shapes in the data set. The first row shows the template shape before normalization, rows 2–4 show variations of each class within the training data. Data generated and published by Yan et al. (2021).

Figure 11 shows an area where all three types are present on the island of Mallorca, Spain, and examples for each class.

4.2. Model implementation and hyperparameter

We test the capability of the three encoding schemes in combination with the three aforementioned types of neural networks: CNNs, RNNs, and GCNNs. In our experiments, each model type will be fed with input from each encoding scheme (c, s, and f; see Table 4), and we compare the performance between the network types and between the encoding schemes.

We implemented all models in Python using the PyTorch framework (Paszke et al., 2019). We trained every network-encoding combination for 250 epochs with a batch size of 16. We used the cross-entropy loss function and the ADAM optimizer with a learning rate of 0.001. After training was done, we selected the epoch with the best accuracy and used this model state for performance evaluation.

4.3. Preprocessing and evaluation

We set the target size for each polygon length at 64 to make sure that every geometry from the experimental dataset could be used, and used interpolation, iteratively dividing and padding to extend shorter geometries to this size. We randomly split the dataset for training and testing at a ratio of 80:20. We then encoded every building using all encoding schemes and trained all models with the same set of training and test data to ensure comparability and the fairness of the competition. As global feature descriptors, we used elongation, circularity, rectangularity, squareness and convexity, and the same local and regional features as Yan et al. (2021). Feature normalization was done using the Z-score method. The whole workflow was repeated four times, and the final result was calculated as the mean performance over five iterations for each net and experimental design.

We did the same workflow for our polyline data, but we set the target sequence length to 128. After splitting the set into train and test set, we enlarged the training set by mirroring the geometries on both axes, leading to a total of 2620 training and 164 test lines. For global line features, we selected anchor line length, elongation, bandwidth, directionality, and error variance and used the same local and regional features as for polygons but with the first and last point as regional reference points.

To evaluate the performance of the different models, we used two metrics which are commonly used for multi-classification tasks: accuracy and f1-score. Accuracy is the ratio of correct predicted samples compared to all samples and is calculated as:

$$accuracy = \frac{1}{m}\sum_{i=1}^{m}(f(x_i) = y_i) \qquad (2)$$

where m is the number of samples, $f(x_i)$ is the label predicted for the i-th sample, and y_i is the true label for the i-th sample.

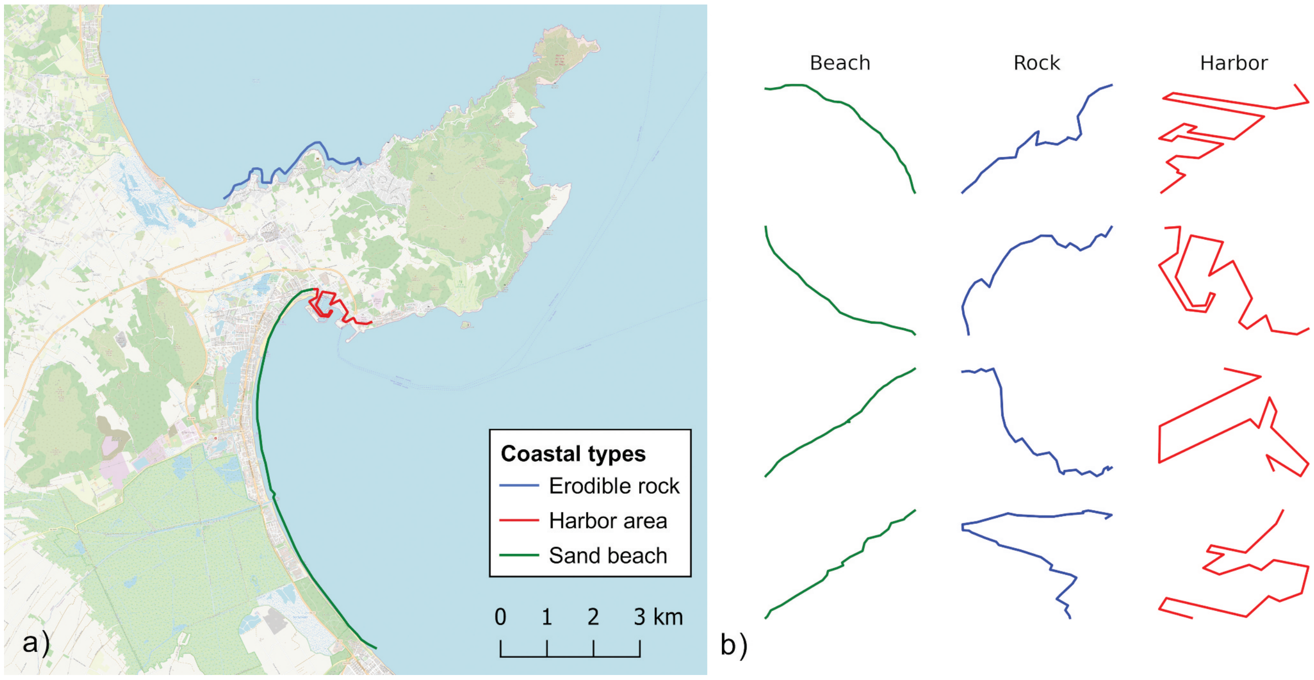

Figure 11. a) Line data sample of coastal types near Alcúdia, Spain. b) Selected examples for each coastal type in the data set. Data from the European marine observation and data network (EMODnet), basemap tiles: OpenStreetMap, under ODbL.

The $f1 - score$ is a metric that combines the values of precision and recall for each class, and $macro - F1$ is the mean $f1 - score$ over all classes. Precision for class k is thereby defined as the ratio of correctly classified samples compared to all samples that are classified as k:

$$precision_k = \frac{TP_k}{TP_k + FP_k} \quad (3)$$

where TP are true positives and FP are false positives for class k. The *recall* for class k is the ratio of correctly classified samples compared to all samples that should have been correctly classified as k:

$$recall_k = \frac{TP_k}{TP_k + FN_k} \quad (4)$$

where FN are false negatives for class k. The $f1 - score$ is now calculated as:

$$f1 - score = \frac{2 * precision_k * recall_k}{precision_k + recall_k} \quad (5)$$

where *precision* and *recall* are the respective metrics for class k. Accuracy and f1-scores both range between 0 and 1. The higher the values, the better is the performance of the model. As long as all classes are equally distributed over the test set and there are no high deviations between the f1-scores of the classes, accuracy and macro-F1 score are in close range to each other. This was the case for our first experiment, and so we renounce to show the f1-score there.

5. Results

In this section, we present the results of our experiments to show the performance of the encoding schemes in combination with CNNs, RNNs, and GCNNs (Section 5.1). Next, we evaluate the impact of the preprocessing step of matching regularity (Section 5.2) and test properties such as orientation and scale (Section 5.3), before conducting a sensitivity study for feature selection (Section 5.4). We then apply the encoding approach on polyline data (Section 5.5).

5.1. Performance of encoding schemes and neural networks

Table 5 lists the accuracy results of the different encoding schemes and neural networks performing the building shape classification task while using interpolation, iterative dividing, and padding during preprocessing. The model name reflects the combination of network and encoding scheme. In general, we can see that the encoding scheme that utilizes feature descriptors outperforms the other two schemes in combination with every type of neural network, and that the respective best performances are quite similar between CNNs, RNNs, and GCNNs.

Table 5. Accuracy results of the experiments on encoding schemes and neural networks. c, s, and f relate to the respective encoding scheme (see also Table 4). The best results per network type are marked (benchmark models are evaluated separately).

	interpolate	iterative dividing	padding
CNN			
tVeerCNN	0.9546	0.8878	0.7371
tVeerCNN	0.9499	0.8784	0.6488
tVeerCNN	0.9563	0.9452	0.9175
dCNN+c	0.9701	0.9261	0.7756
dCNN+s	0.9687	0.9177	0.7776
dCNN+f	0.9835	0.9513	0.9196
LiuDPCN	0.9966	0.9961	0.9732
RNN			
tVeerRNN	0.9664	0.9224	0.78
tVeerRNN	0.9704	0.9204	0.7623
tVeerRNN	0.9956	0.9886	0.947
GCNN			
GCNN+c	0.6828	0.7041	0.6087
GCNN+s	0.6845	0.6922	0.6369
GCNN+f	0.9969	0.9956	0.9724
YanGCNN	0.9968	0.9973	0.9748

Comparing the performances of the different encoding schemes among one another, dCNN+f (98.35%), tVeerRNN+f (99.56%), and GCNN+f (99.69%) accomplish the best accuracies for their respective network types. The second-best accuracy for CNN (97.01%) and GCNN (70.41%) models were reached by the scheme which encodes the vertex coordinates. For RNNs, the sequence-encoding schemes perform slightly better than the coordinate-encoding. However, both results for GCNN+c and GCNN+s are just around 70% and therefore nearly 30% worse than the GCNN +f, which shows that GCNNs without vertex feature descriptors are not converging at a good level during training (see Figure 12).

When comparing the best performances of the different types of neural networks to each other, the results for the GCNN and the RNN were the best, followed by the CNN. GCNN+f performs even better than the two benchmark models for building shape cognition (99.68% and 99.66%), although the margin was very small. Aside from the aforementioned bad performance of the two GCNNs using coordinates and sketch sequences, the overall performance of the different neural network types was promising, as even the worst performing encoding schemes for dCNN (94.99%) and tVeerRNN (96.64%) accomplished accuracies of 95%. As a result, we can state that all types of neural networks were able to reach exceptionally good results when benefiting from feature encoding.

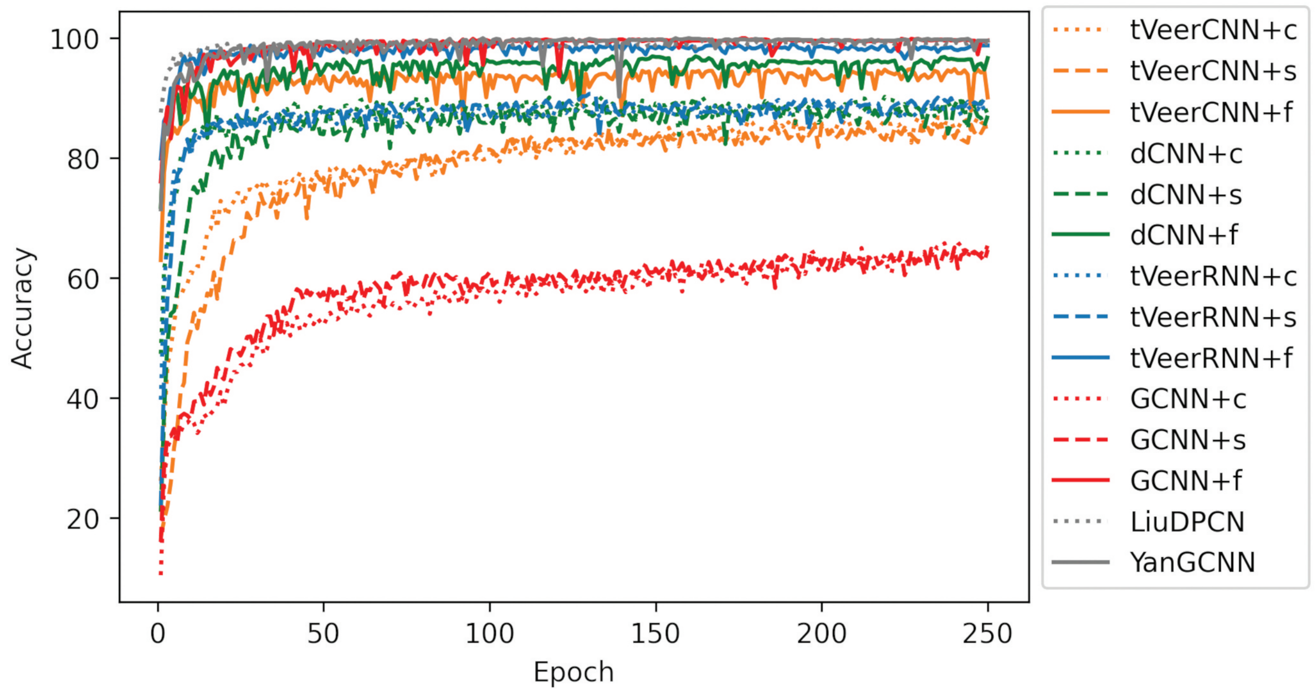

Figure 12. Test accuracy during training for each combination of network and encoding scheme with iterative dividing and object-based normalization during preprocessing. In our pretests with 500 and 1000 epochs, none of the models increase their performance after the 250th epoch.

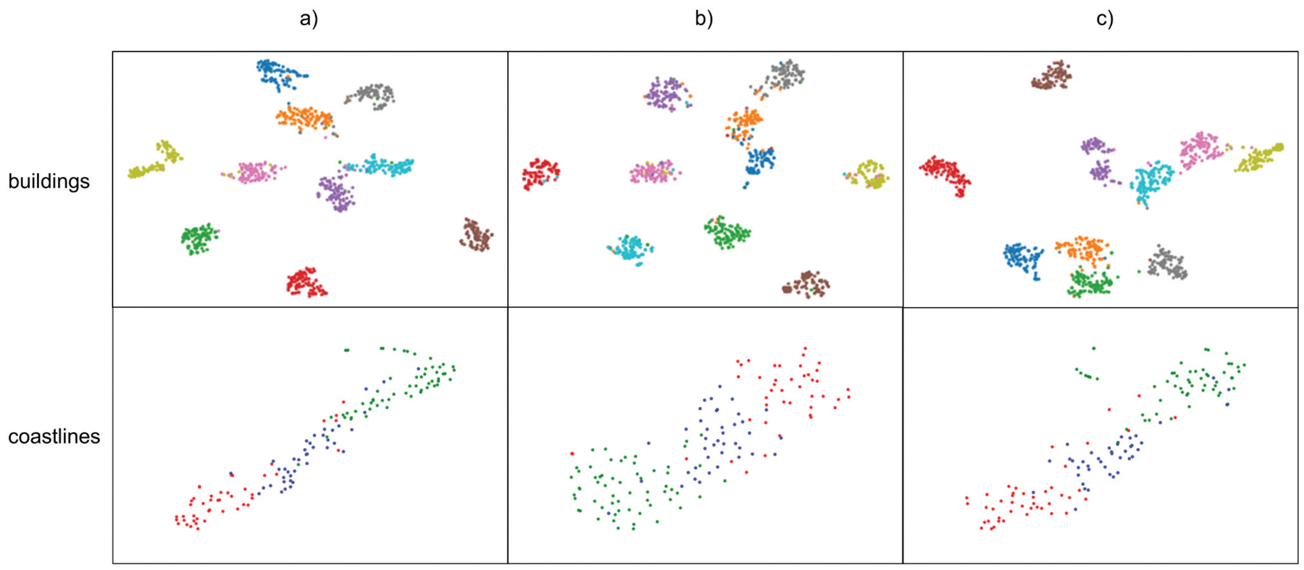

Figure 13. Visualization of the learned shape embeddings for models using feature encoding: a) dCNN+f, b) tVeerRNN+f, and c) GCNN +f. Top: embeddings for the building classification task; colors represent the ground truth class of each sample, as in Figure 10. Bottom: embeddings for the coastline classification task; colors represent the ground truth class of each sample, as in Figure 11.

5.2. Preprocessing approaches to match regularity

We have varied the preprocessing method between three different approaches and compare the impact on the performance of each network, which can be also seen in Table 5. The interpolation method produces the best results for nearly every network and encoding scheme, followed by the iterative dividing approach. The difference between these two methods varied between less than 0.1% (GCNN+f) and 7% (tVeerCNN+s), whereas three of the four GCNN models perform better with the iterative dividing preprocessing. Padding achieved the worst results for the majority of the models, and the shortfall compared to the interpolation method differs depending on the encoding scheme. Feature encoding schemes achieve from 2.5% (GCNN+f) less to 6.4% less (dCNN+f), while the other encodings reached lower accuracies with at least 20% less.

5.3. Property testing

We conduct three experiments of alternative preprocessing and training methods and compare them with the results above (see Table 6). First, we wanted to test if the orientation of a geometry has an influence on the ability of the encoding schemes and networks to classify correctly. Therefore, we rotated the horizontal direction of all geometries in the training data toward the y-axis, based on their major axis of their minimum area bounding rectangle. We then test the models with non-rotated geometries and analyze the accuracy. The results show that networks with feature-based encodings perform only slightly worse, with only dCNN+f declining more than 0.5%. Networks with other encoding schemes were less able to perform when the orientation in the training set was normalized.

Next, we tested if the size of the geometries influences the outcome of the models. Therefore, we scaled the test set with the factor 2 and analyzed the class predictions. Again, our results show that models that learn from feature encoding perform nearly as good as with the normal test set. From the others, dCNN+c and tVeerRNN+s performed better than in the orientation test, while the GCNN+s model does not converge at all under these conditions.

The last column in Table 6 shows the results when we use the map-based coordinate normalization (see Section 3.1.2) during preprocessing instead of the object-based. In this test, tVeerRNN+f had the best results, followed by GCNN+f. dCNN+f lost more than 11.5% of its performance, while all models not utilizing feature encoding lost 30% and more.

5.4. Sensitivity tests for feature encoding

Table 7 presents the results of our test to identify the optimal configuration of neighboring points that will be used in the feature encoding (see Figure 5). For all models, the best configuration was to use three neighbors with the distance of k = [1,2,4]. It shows that with more than three neighbors in consideration, the additional information of each feature is redundant.

Table 8 shows a sensitivity analysis with respect to the different feature categories. The performance of all models decreases if one of the categories was absent in the encoding, with local features having the smallest and regional features the biggest influence on the results. If both local and regional features were missing, the performance was even worse for all models. Between the models, tVeerRNN+f was the most stable model, loosing not more than 2% in any of the experiments, while the GCNN+f looses up to nearly 10% when only utilizing global features.

5.5. Applying the encoding schemes to polyline data

In our last experiment, we apply the encoding schemes on polyline data. Table 9 shows the results of the experiment, including the f1-scores for the three classes in the coastline data set: beaches, rocks/cliffs, and harbors. As a first result, we can state that our iterative dividing method outperforms interpolation for all models due to the fact that the Douglas-Peucker simplification during preprocessing modifies the most important characteristic for this classification task. Furthermore, the accuracy of all models is lower than for the building classification task. The GCNN+f performs best with 87.50%, followed by YanGCNN, dCNN+f and

Table 6. Accuracy results of the experiments when learning from modified properties: a) normalized orientation, b) different scales, c) map-based coordinates. Marked values are within a 1.0% deviation of the standard training results.

experiment		a) oriented	b) scaled	c) map-based
dCNN+c	0.9701	0.903	0.9506	0.2125
dCNN+f	0.9835	0.977	0.9778	0.8597
LiuDPCN	0.9966	0.9724	0.9284	0.6843
tVeerRNN+s	0.9704	0.8837	0.9204	0.2637
tVeerRNN+f	0.9956	0.9954	0.995	0.9944
GCNN+s	0.6845	0.4321	0.1099	0.1361
GCNN+f	0.9969	0.9952	0.9929	0.9821
YanGCNN	0.9966	0.9956	0.997	0.981

Table 7. Experiments for the identification of the optimal configuration of neighbors k. Best performance for each model is marked.

k=	[1]	[1,2]	[1,2,4]	[1,2,3,4]	[1,2,4,8]	[1,2,4,8,16]
dCNN+f	0.9576	0.9606	0.9818	0.9606	0.9627	0.9637
tVeerRNN+f	0.9808	0.9899	0.9949	0.9868	0.9919	0.9929
GCNN+f	0.9879	0.9929	0.9959	0.9959	0.9959	0.9959
YanGCNN	0.9909	0.9939	0.9959	0.9949	0.9949	0.9959

Table 8. Influence of feature categories on the model performance. Best result per feature selection is marked.

features	all	only global	no global	no regional	no local
dCNN+f	0.9667	0.9183	0.9405	0.9173	0.9637
tVeerRNN+f	0.9919	0.9697	0.9889	0.9717	0.9889
GCNN+f	0.9979	0.9002	0.9899	0.9798	0.9919
YanGCNN	0.9979	0.8871	0.9909	0.9727	0.9929

Table 9. Results of the experiment of line classification with interpolation and iterative dividing as preprocessing methods, with the better result per model marked. All f1-scores are calculated for the iterative dividing approach.

	interpolate	**iter. div.**	macro-f1	f1-beach	f1-rock	f1-harbor
dCNN+c	0.6486	0.7681	0.7996	0.8632	0.7593	0.7765
dCNN+f	0.8319	0.8542	0.8637	0.8989	0.8257	0.8667
tVeerRNN+s	0.6278	0.7667	0.7779	0.8283	0.7143	0.7912
tVeerRNN+f	0.8472	0.8667	0.8488	0.8764	0.8257	0.8444
GCNN+s	0.5431	0.6915	0.7878	0.8478	0.7339	0.7816
GCNN+f	0.8167	0.875	0.924	0.9348	0.9126	0.9247
YanGCNN	0.8264	0.8681	0.9244	0.9412	0.9189	0.913

tVeerRNN+f, which all accomplish accuracies between 85% and 87%. By contrast, the models that do not utilize vertex features do not reach more than 77%. This shows that the benefits of vertex feature extraction are also valid for polylines.

We explain the lower performance to a certain amount with the differences between the data sets: While the building dataset contains more than 5000 human-built geometries, the coastline dataset has only 860 geometries from which the majority are naturally occurring shapes of rocks and beaches. These characteristics could lead to less distinguishable – and therefore learnable – classifications, leading to the lower accuracies of the models. To gain further insights into the results of our experiments, we visually compare the shape encodings in the embedding space by utilizing the t-SNE algorithm (van der Maaten & Hinton, 2008). This algorithm reduces the high-dimensional embeddings to just two dimensions, which enables straightforward visualizations. Figure 13 shows the respective encoding visualizations from instances of three models. The building classes are densely clustered and clearly distinguishable from each other, while the embeddings of the coastlines are more scattered and appear in a linear shape, which means that most of the false-positive samples occur between the classes of harbor and rock/cliff and between rock/cliff and beaches. The f1-scores of Table 9 support this assumption: For all but the YanGCNN model, the f1-score of the rock/cliff class is lower than the others.

6. Case study

We implement an exemplary building generalization workflow based on shape classification and template matching. The aim of this case study is to test the suitability of the shape classification models as part of a building generalization workflow, while the visual evaluation of the generalization results allows us to simultaneously discuss and compare the classification results of the three feature-based deep learning models (CNN+f, RNN+f, GCNN+f).

6.1. Workflow

The approach of our case study is suitable for scales between 1:20,000 and 1:50,000, where representational shapes of buildings are more important for the understanding of the map scene than the accuracy of the buildings' detailed boundaries. The workflow is inspired by the works of Rainsford and Mackaness (2002) and Yan et al. (2017) – while replacing shape measurement with our shape classification – and combines the building generalization operators of exaggeration, elimination, and simplification (see also Li et al. (2004)).

We start with the elimination of all buildings which are smaller than a predefined threshold, which we set for our target scale of 1:25.000 at 200 m^2. Next, we select all buildings that are bigger than 15,000 m^2 to retain their initial shape, as we want these landmarks to work as visual anchors in the area. We then preprocess the training data from Yan et al. (2021) as well as the case study data with iterative dividing and object-based normalization and select for k = [1,2,4] and the same features as described in section 4.3. We train our models with the training data set and the same parameter settings as in section 4. After that, we classify each building from our case study area. As the next step, we normalize the template shapes to create a collection of templates. For each classified building in our case study area, we resize the respective template according to the minimum area bounding rectangle of the building of interest, and rotate the template based on the MABR's horizontal direction. As the last step, we apply a translation to match the adjusted template and the location of the building of interest.

6.2. Data

Our case study area is located around the Temple of the Sun in Beijing, China. The area has a variety of different building types regarding their shape, size, and orientation and is shown in Figure 14. We retrieved all buildings available in this area from Open Street Map via the Overpass API and eliminated all geometries that are completely covered by others. This results in 360 distinct buildings, from which 336 buildings are larger than the threshold of 200 m^2.

6.3. Results

Figure 15 shows the resulting buildings using our approach based on the three models. In general, the results are promising for all models. The straight-lined arrangements of buildings on the northern, eastern, and southern border of the quarter are mostly preserved, and the largest buildings – beside those which exceed the threshold – were adequately matched with the correct templates. The same can be said for most of the smaller buildings, which now have less complex boundary shapes. Nevertheless, there are also some shortcomings for all results. Some matched templates seemed to disarrange the overall building arrangement, either because the models predict an unsuitable shape class for the respective building or the result of the matching algorithm is incorrectly rotated because of some individual shape characteristics that influence the MABR of the building. Furthermore, problems arise when building boundaries are adjacent, but the adjacency is not preserved after the template matching, either because of the characteristics of the template shapes or through the different orientations of the respective MABRs.

The aforementioned problems occur for all of the three models without obvious pattern, and there is no model whose result stands out compared to the other two. For that reason, we further discuss the different outcomes of the models based on five examples located in the study area, focusing on two individual buildings and three building ensembles (see Figure 16). The first example shows the shape classifications for a long building with an irregular boundary. Looking at the training data examples in Figure 10, this should be classified as *I* as from the RNN+f, but the predictions of the CNN+f and GCNN+f are also comprehensible. The same can be said for the second example, in which the RNN+f prediction is again different from the other two models, but this time the *F*-shape seems closer to the original geometry than the *E*.

The third example shows two series of similar shaped buildings. Regarding the series on the right side of the image, each model classifies the individual buildings into the same class, but the classes vary between the models: CNN+f predicts *F*, while RNN+f predicts *L* and GCNN+f *Z*. Focusing on the curved series of *Y*-shaped buildings on the left side of the image, the second building is slightly different from the other three, leading to different classifications from RNN+f (*T*) and GCNN+f (*F*). Although this is comprehensible from a geometrical point of view, the results from the CNN+f are preferred for a map generalization due to the more consistent group pattern they built. The same result occurs for the fourth example, in which the CNN+f predicts the same classes for the two respective pairs of buildings in the center of the image, while GCNN +f predicts different classes for one and RNN+f for both pairs of buildings. In contrast, the RNN+f produces the best result for the area in the last example, while the other models struggle mostly with the larger buildings in the lower part of the image.

7. Discussion

Deep learning in the field of map generalization is mainly done with raster-based techniques. We utilized a feature-based encoding scheme to learn

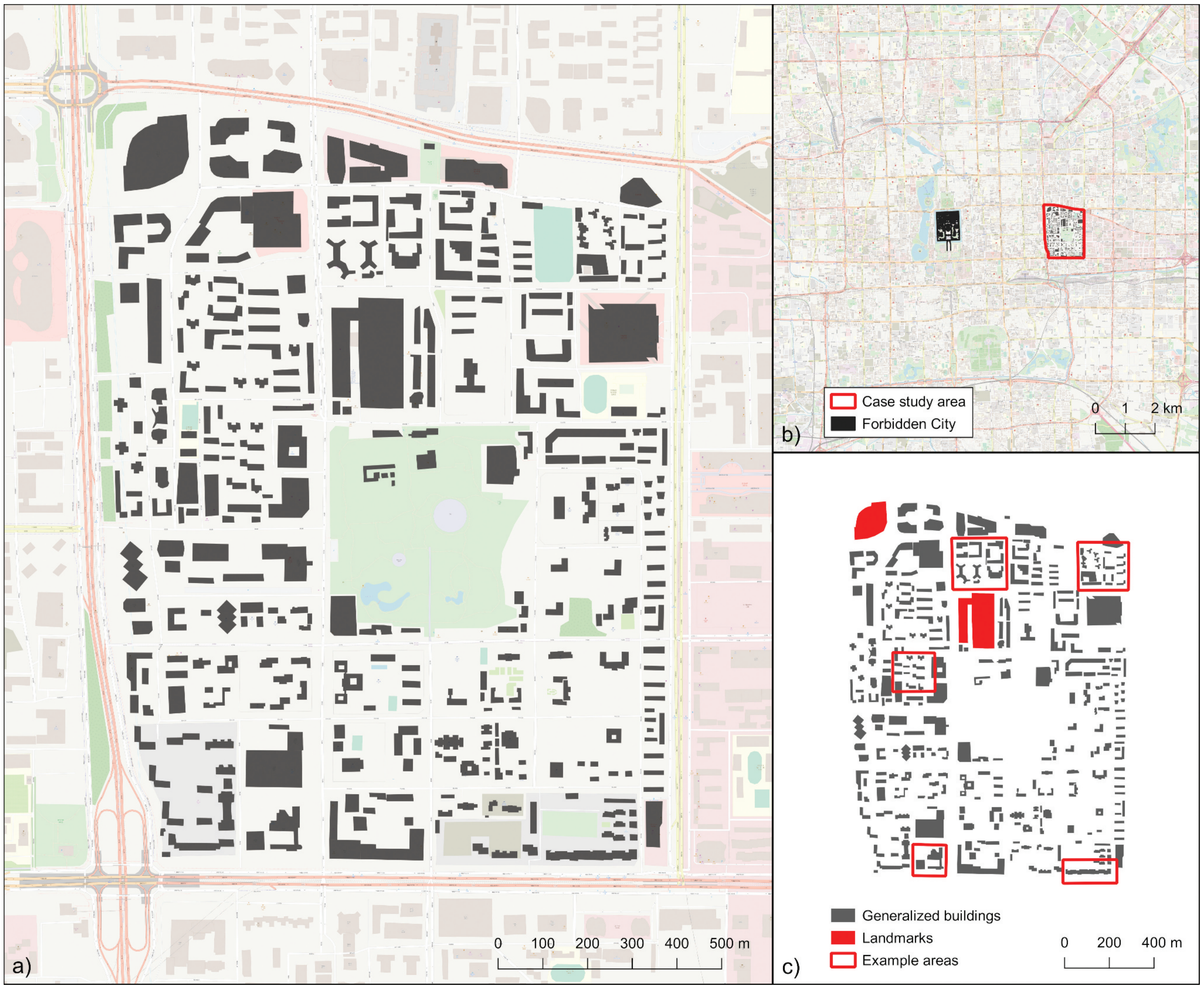

Figure 14. a) Buildings in the area around the temple of the sun, Beijing. b) Location of the case study area in the east of the Forbidden City (in black). c) Locations of the five example areas and the two landmarks with an area above 15,000 m^2 (the Ministry of Foreign Affairs of the People's Republic of China in the north-western corner and the Ritan International Trade Center in the center of the map). Map data and basemap tiles: OpenStreetMap, under ODbL.

directly from vector-based input and implemented a shape classification task as part of a building generalization workflow. In this section, we want to discuss the results and implications of our work.

7.1. Geometry encoding with feature description

We showed that feature description can help to convert shape representation techniques from the constraint-based map generalization workflow to deep learning applications as the first step toward a vector-based, end-to-end generalization model. Furthermore, utilizing (shape) measures from the conventional map generalization workflow helps to regulate the causal mechanisms of the deep learning models. These mechanics are mostly hidden and only visible through the model output, which could be problematic from a cartographic point of view, especially in semi-automatic processes with human editing (Touya et al., 2019). As a solution, we showed in Section 3 that there is a huge body of generalization knowledge available, including various approaches on how to select meaningful measures and how to describe shapes appropriately, and the input for deep learning models should reflect this knowledge to make the output more consistent and correct (Sester, 2020). Feature descriptors can also play an important role to define the loss function during the learning process. Shape measures are commonly used in generalization systems like AGENT (Duchêne et al., 2018) to assess the fulfillment of preservation constraints or similarity measures (Blana & Tsoulos, 2022), and this could be the same when map

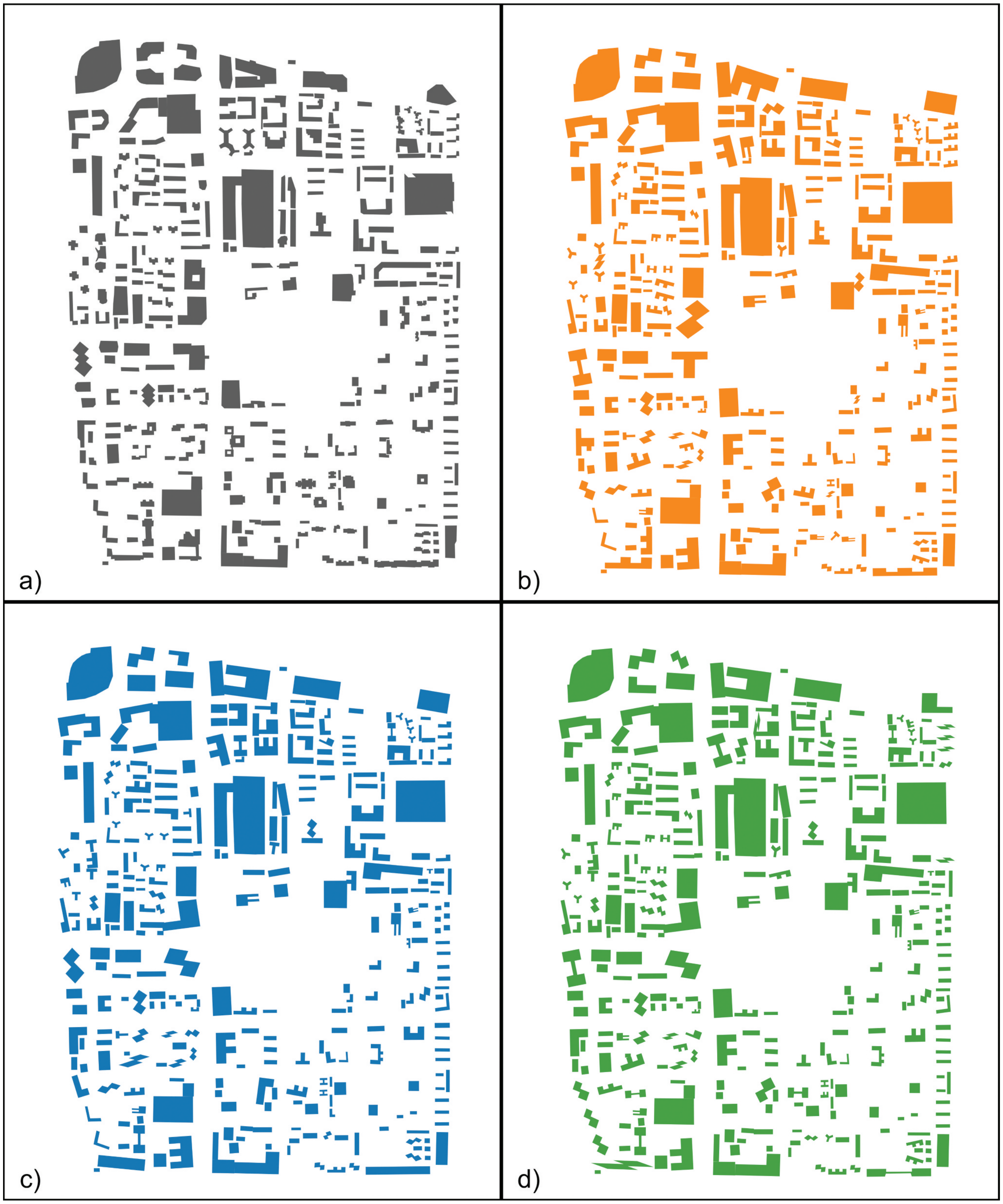

Figure 15. Results of the case study utilizing deep learning models for shape coding and template matching. a) Original data, b) classification with CNN+f, c) classification with RNN+f, d) classification with GCNN+f. Landmarks were not generalized.

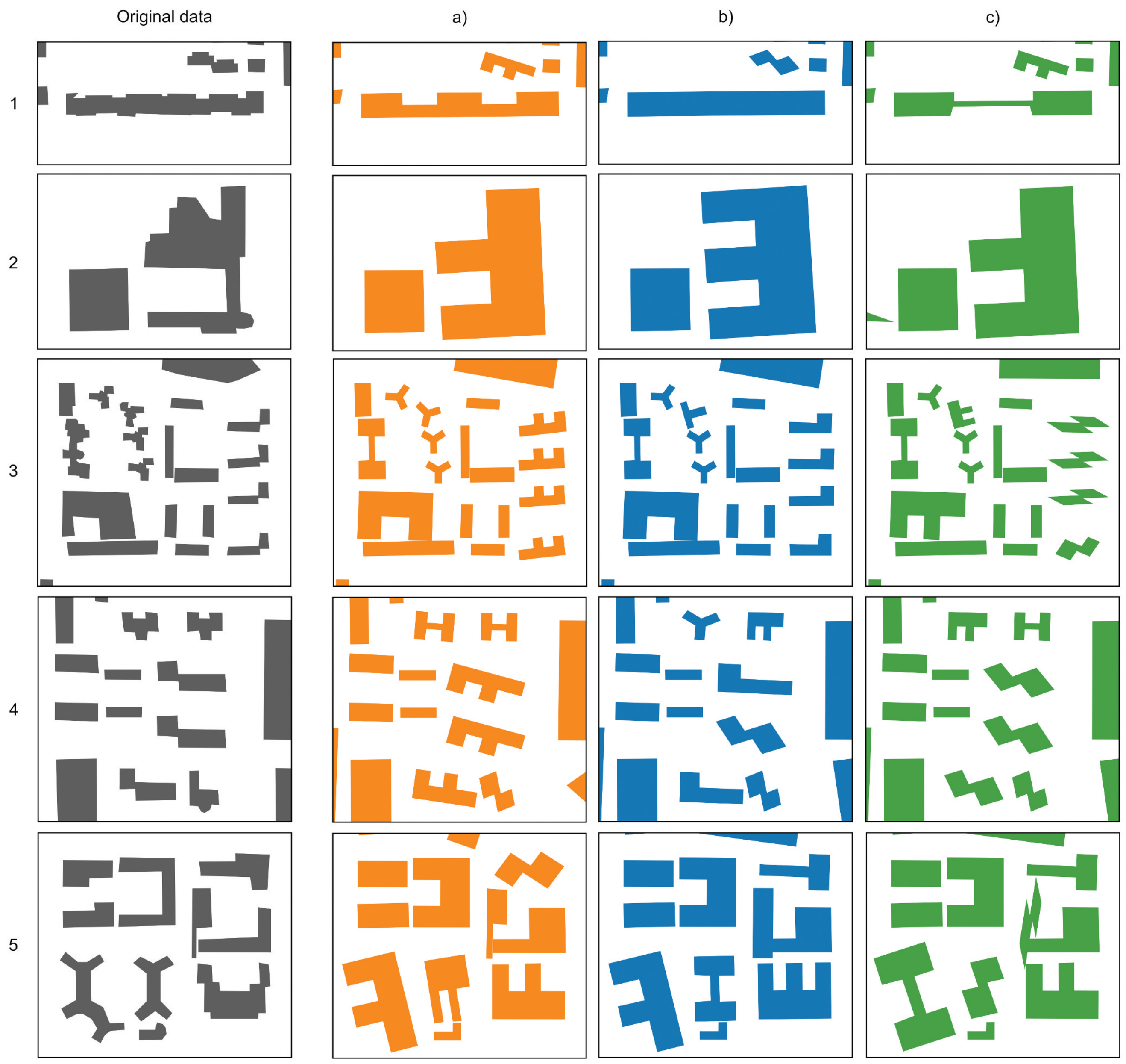

Figure 16. Examples of template matching results for a) CNN+f, b) RNN+f, and c) GCNN+f, focusing on unusual shapes. 1) Long building with irregular boundary. 2) Building with unusual shape similar to an *F*. 3) Area with two series of four similar buildings. 4) A collection of two pairs of unusual building shapes. 5) Buildings in close proximity in combination with unusual shapes.

generalization is done with deep learning models. This can be implemented by defining specific measure values as the targets in supervised learning models or by using respective constraints for reinforcement learning.

7.2. Building generalization using shape classification

We enhanced the feature encoding preprocessing for an optimal use as part of a map generalization workflow and exemplarily implemented an approach for building generalization. Our experiments on the different preprocessing methods showed that our proposed iterative dividing method – preserving the original shape – is preferable for map generalization applications when shape features are of interest, like in our polyline classification task. The results thereby reflect the specific characteristics of the neural networks: For the CNNs, the interpolation method achieved better results due to the regularity that is needed for the convolution operation to work. The architectures of RNNs and GCNNs are more robust against irregular data, and for these models both preprocessing methods achieve similar results when feature descriptors are used for encoding, as this additional input information helps to overcome irregularities in the data. However, the case study showed that these small margins in the model performances for building classification are not emphasized in the results of our building generalization workflow using template matching. For the few buildings in the case study area that had different class predictions between the models, we found no systematic bias in the visual results. The examples shown in Figure 16 revealed that no model was superior compared to the others, as all produced unusual results at certain occasions when the other two did not.

As a result, a combination of all three models with majority voting for the final class prediction could be a straightforward improvement to the workflow. A second approach to achieve better results is to optimize the shape classes we train our models with. While we use the 10 alphabetical characters of Yan et al. (2021), Rainsford and Mackaness (2002) use a slightly different character selection (I, F, P, G, E, L, U, O, T), and Yan et al. (2017) categorize their templates into *simple*, *symbolic,* and *composite shapes* and also extracted shapes automatically from the dataset. Even further, a backpropagation network can be utilized to decide which template is more appropriate (Yang et al., 2022).

7.3. Future research directions

Utilizing the geometric encoding schemes to produce generalized output is the next logical step, and there are several research directions in consideration. We want to discuss them based on two questions.

7.3.1. End-to-end generalization or focus on optimizing subprocesses?

Following the aforementioned work on point generalization, line simplification, or raster-based building generalization, we aim at a vector-based end-to-end object generalization. But for the integration into a more general workflow of map generalization, it could be more advantageous to substitute only selected subprocesses with deep learning techniques, as we did with our approach of building generalization: Building shape classification is utilized as a data enhancement step, and the generalization is done afterward with predefined matching operations according to the predicted shape class. Utilizing feature descriptors for local, regional, and global features was deployed for Graph Convolutional Neural Networks, and we showed that this approach can be transferred to other types of neural networks such as CNNs and RNNs. As map generalization is a combination of different tasks and cognitive processes, having the opportunity to choose between different types of neural networks ensures more task-specific network designs – instead of applying a problem from map generalization to a predetermined type of network, possibly dealing with network-specific limitations – and therefore enabling more applications within the map generalization workflow.

7.3.2. How to define the generalization problems?

The majority of the end-to-end generalization models used Deep Generative Networks, as generalization can be seen as a creative process to recreate new objects. The proposed encoding scheme using feature descriptors can be utilized in the same way as input images for generative models, and the implementation of this approach is a major part of our future work. The challenge is thereby to link generalization operations with suitable deep learning tasks to achieve the best results. For example, among many other solutions, simplification can be implemented as a classification task, where the vertex points are classified to decide which ones will be removed (Zhou et al., 2022), or with an autoencoder architecture, where the model learns an efficient representation of the input model – and therefore a simplified shape.

8. Conclusion

Learning directly from vector data has several advantages, and this work can be seen as the first step toward the deployment of a vector-based deep learning model that performs end-to-end object generalization in the future. We identified vector-based schemes from the literature, which can encode lines and polygons to a hidden embedding space and enhance the preprocessing workflow to adequately represent map data that can be utilized for map generalization tasks. Next, we compared the performance of CNNs, RNNs, and GCNNs in combination with different preprocessing approaches and encoding schemes. Our results show that feature descriptors improve the accuracy of all three types of neural networks significantly, and that the overall performances of CNNs, RNNs, and GCNNs are quite similar while solving shape classification task for polylines and polygons. Furthermore, we implemented the classification task as part of an exemplary building generalization workflow, and show that all three models achieve considerable generalization results. The next step for the future is the implementation of an end-to-end generalization model, where the feature descriptors we deployed for geometry encoding could further play an important role for learning and evaluation.

Disclosure statement

No potential conflict of interest was reported by the author(s).

ORCID

Martin Knura http://orcid.org/0000-0002-1678-866X

Data availability statement

The data and code that support the findings of this study are available in GitHub at: https://github.com/geo-mart/Vector-Shape-Encoder.

References

Arulkumaran, K., Deisenroth, M. P., Brundage, M., & Bharath, A. A. (2017). Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, *34*(6), 26–38. https://doi.org/10.1109/MSP.2017.2743240

Basaraner, M., & Cetinkaya, S. (2017). Performance of shape indices and classification schemes for characterising perceptual shape complexity of building footprints in gis. *International Journal of Geographical Information Science*, *31*(10), 1952–1977. https://doi.org/10.1080/13658816.2017.1346257

Blana, N., & Tsoulos, L. (2022). Generalization of linear and area features incorporating a shape measure. *ISPRS International Journal of Geo-Information*, *11*(9), 489. https://doi.org/10.3390/ijgi11090489

Buttenfield, B. 1991, 1). A rule for describing line feature geometry. In B. Buttenfield & R. B. McMaster. (Eds.), *Map generalization: Making rules for knowledge representation* (pp. 150–171). Longman Scientific & Technical London.

Courtial, A., El Ayedi, A., Touya, G., & Zhang, X. (2020). Exploring the potential of deep learning segmentation for mountain roads generalisation. *ISPRS International Journal of Geo-Information*, *9*(5). https://doi.org/10.3390/ijgi9050338

Courtial, A., Touya, G., & Zhang, X. (2021). Generative adversarial networks to generalise urban areas in topographic maps. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *XLIII-B4-2021*, 15–22. https://doi.org/10.5194/isprs-archives-XLIII-B4-2021-15-2021

Courtial, A., Touya, G., & Zhang, X. (2022a, May 7). Constraint-based evaluation of map images generalized by deep learning. *Journal of Geovisualization and Spatial Analysis*, *6*(1), 13. https://doi.org/10.1007/s41651-022-00104-2

Courtial, A., Touya, G., & Zhang, X. (2022b). Representing vector geographic information as a tensor for deep learning based map generalisation. *AGILE: GIScience Series 3*, 32. https://doi.org/10.5194/agile-giss-3-32-2022

Duchêne, C., Touya, G., Taillandier, P., Gaffuri, J., Ruas, A., & Renard, J. (2018, January). *Multi-agents systems for cartographic generalization: Feedback from past and on-going research* (Research Report). IGN (Institut National de l'Information Géographique et Forestière); LaSTIG, équipe COGIT. https://hal.archives-ouvertes.fr/hal-01682131

Du, J., Wu, F., Xing, R., Gong, X., & Yu, L. (2022). Segmentation and sampling method for complex polyline generalization based on a generative adversarial network. *Geocarto International*, *37*(14), 4158–4180. https://doi.org/10.1080/10106049.2021.1878288

Du, J., Wu, F., Yin, J., Liu, C., & 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. https://doi.org/10.1080/15230406.2021.2013944

Fan, H., Zhao, Z., & Li, W. (2021). Towards measuring shape similarity of polygons based on multiscale features and grid context descriptors. *ISPRS International Journal of GeoInformation*, *10*(5), 279. https://doi.org/10.3390/ijgi10050279

Feng, Y., Thiemann, F., & Sester, M. (2019). Learning cartographic building generalization with deep convolutional neural networks. *ISPRS International Journal of Geo-Information*, *8*(6), 258. https://doi.org/10.3390/ijgi8060258

García Balboa, J. L., & Ariza López, F. J. (2008, September 1). Generalization-oriented road line classification by means of an artificial neural network. *GeoInformatica*, *12*(3), 289–312. https://doi.org/10.1007/s10707-007-0026-z.

Ha, D., & Eck, D. (2017). A neural representation of sketch drawings. *arXiv*. https://doi.org/10.48550/ARXIV.1704.03477

Iddianozie, C., & McArdle, G. (2021). Towards robust representations of spatial networks using graph neural networks. *Applied Sciences*, *11*(15), 6918. https://doi.org/10.3390/app11156918

Kang, Y., Rao, J., Wang, W., Peng, B., Gao, S., & Zhang, F. (2020). Towards cartographic knowledge encoding with deep learning: A case study of building generalization. In *Autocarto 2020, the 23rd International Research Symposium on Cartography and Giscience*. https://cartogis.org/docs/autocarto/2020/docs/abstracts/2b%20Towards%20Cartographic%20Knowledge%20Encoding%20with%20Deep%20Learning%20A.pdf

Karsznia, I., & Sielicka, K. (2020). When traditional selection fails: How to improve settlement selection for small-scale maps using machine learning. *ISPRS International Journal of GeoInformation*, *9*(4), 230. https://doi.org/10.3390/ijgi9040230

Karsznia, I., Weibel, R., & Leyk, S. (2022). *May ai help you? Automatic settlement selection for small-scale maps using selected machine learning models*. https://cartogis.org/docs/autocarto/2022/docs/abstracts/Session3Karsznia5316.pdf

Liao, S., Bai, Z., & Bai, Y. (2012, December 1). Errors prediction for vector-to-raster conversion based on map load and cell size. *Chinese Geographical Science*, *22*(6), 695–704. https://doi.org/10.1007/s11769-012-0544-y.

Liu, C., Hu, Y., Li, Z., Xu, J., Han, Z., & Guo, J. (2021). Triangleconv: A deep point convolutional network for recognizing building shapes in map space. *ISPRS*

International Journal of Geo-Information, *10*(10), 687. https://doi.org/10.3390/ijgi10100687

Li, Z., Yan, H., Ai, T., & Chen, J. (2004). Automated building generalization based on urban morphology and gestalt theory. *International Journal of Geographical Information Science*, *18*(5), 513–534. https://doi.org/10.1080/13658810410001702021

Mackaness, W., Burghardt, D., & Duchêne, C. (2014). Map generalisation: Fundamental to the modelling and understanding of geographic space. In D. Burghardt, C. Duchêne, & W. Mackaness, (Eds.), *Abstracting geographic information in a data rich world: Methodologies and applications of map generalisation*. (pp. 1–15). Springer International Publishing. https://doi.org/10.1007/978-3-319-00203-3_1

Mai, G., Janowicz, K., Hu, Y., Gao, S., Yan, B., Zhu, R., L. Cai, & Lao, N. (2022). A review of location encoding for geoai: Methods and applications. *International Journal of Geographical Information Science*, *36*(4), 639–673. https://doi.org/10.1080/13658816.2021.2004602

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen T., Lin Z. ,Gimelshein N. ,Antiga L. , Desmaison A. ,Köpf A. ,Yang E. ,DeVito Z. ,Raison M. , Tejani A. ,Chilamkurthy S. ,Steiner B. ,Fang L, . . . Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. *arXiv*. https://doi.org/10.48550/ARXIV.1912.01703

Qi, C. R., Su, H., Mo, K., & Guibas, L. J. (2016). Pointnet: Deep learning on point sets for 3d classification and segmentation. *arXiv*. https://doi.org/10.48550/ARXIV.1612.00593

Rainsford, D., & Mackaness, W. (2002). Template matching in support of generalisation of rural buildings. In D. E. Richardson & P. van Oosterom (Eds.), *Advances in spatial data handling* (pp. 137–151). Springer.

Ruthotto, L., & Haber, E. (2021). An introduction to deep generative modeling. *CoRr Abs/2103.05180*, *44*(2). https://doi.org/10.48550/arXiv.2103.05180

Samsonov, T. E., & Yakimova, O. P. (2017). Shape-adaptive geometric simplification of heterogeneous line datasets. *International Journal of Geographical Information Science*, *31*(8), 1485–1520. https://doi.org/10.1080/13658816.2017.1306864

Sester, M. (2020 12). Cartographic generalization. *Journal of Spatial Information Science*, (21), 5–11. https://doi.org/10.5311/JOSIS.2020.21.716

Stoter, J., Zhang, X., Stigmar, H., & Harrie, L. (2014). Evaluation in generalisation. In D. Burghardt, C. Duchêne, & W. Mackaness (Eds.), *Abstracting geographic information in a data rich world: Methodologies and applications of map generalisation* (pp. 259–297). Springer International Publishing.

Touya, G., & Lokhat, I. (2020, April). Deep learning for enrichment of vector spatial databases: Application to highway interchange. *ACM Transactions on Spatial Algorithms and Systems*, *6*(3), 1–21. https://doi.org/10.1145/3382080

Touya, G., Zhang, X., & Lokhat, I. (2019). Is deep learning the new agent for map generalization? *International Journal of Cartography*, *5*(2–3), 142–157. https://doi.org/10.1080/23729333.2019.1613071

van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, *9*(86), 2579–2605. http://jmlr.org/papers/v9/vandermaaten08a.html

Veer, R. V. T., Bloem, P., & Folmer, E. (2018). Deep learning for classification tasks on geospatial vector polygons. *arXiv*. https://doi.org/10.48550/ARXIV.1806.03857

Vinyals, O., Fortunato, M., & Jaitly, N. 2017. Pointer networks. *arXiv*. https://doi.org/10.48550/arXiv.1506.03134

Wang, X., & Burghardt, D. (2020). Using stroke and mesh to recognize building group patterns. *International Journal of Cartography*, *6*(1), 71–98. https://doi.org/10.1080/23729333.2019.1574371

Weibel, R., Keller, S., & Reichenbacher, T. (1995). Overcoming the knowledge acquisition bottleneck in map generalization: The role of interactive systems and computational intelligence. In A. U. Frank & W. Kuhn (Eds.), *Spatial information theory a theoretical basis for gis* (pp. 139–156). Springer.

Yan, X., Ai, T., Yang, M., & Tong, X. (2021). Graph convolutional autoencoder model for the shape coding and cognition of buildings in maps. *International Journal of Geographical Information Science*, *35*(3), 490–512. https://doi.org/10.1080/13658816.2020.1768260

Yan, X., Ai, T., Yang, M., & Yin, H. (2019). A graph convolutional neural network for classification of building patterns using spatial vector data. *IS- PRS Journal of Photogrammetry and Remote Sensing 150*, 259–273. https://doi.org/10.1016/j.isprsjprs.2019.02.010

Yan, X., Ai, T., & Zhang, X. (2017). Template matching and simplification method for building features based on shape cognition. *ISPRS International Journal of Geo-Information*, *6*(8), 250. https://doi.org/10.3390/ijgi6080250

Yang, M., Huang, H., Zhang, Y., & Yan, X. (2022). Pattern recognition and segmentation of administrative boundaries using a one-dimensional convolutional neural network and grid shape context descriptor. *ISPRS International Journal of Geo-Information*, *11*(9), 461. https://doi.org/10.3390/ijgi11090461

Yang, M., Yuan, T., Yan, X., Ai, T., & Jiang, C. (2022). A hybrid approach to building simplification with an evaluator from a backpropagation neural network. *International Journal of Geographical Information Science*, *36*(2), 280–309. https://doi.org/10.1080/13658816.2021.1873998

Zheng, J., Gao, Z., Ma, J., Shen, J., & Zhang, K. (2021). Deep graph convolutional networks for accurate automatic road network selection. *ISPRS International Journal of GeoInformation*, *10*(11), 768. https://doi.org/10.3390/ijgi10110768

Zhou, Z., Fu, C., & Weibel, R. (2022). Building simplification of vector maps using graph convolutional neural networks. *Abstracts of the ICA*, *5*(86), 1–2. https://doi.org/10.5194/ica-abs-5-86-2022