EXPLORING OPTIMAL WAYS TO REPRESENT TOPOLOGICAL AND SPATIAL FEATURES OF BUILDING DESIGNS IN DEEP LEARNING METHODS AND APPLICATIONS FOR ARCHITECTURE

VIKTOR EISENSTADT[1], HARDIK ARORA[2],
CHRISTOPH ZIEGLER[3], JESSICA BIELSKI[4],
CHRISTOPH LANGENHAN[5], KLAUS-DIETER ALTHOFF[6] and
ANDREAS DENGEL[7]
[1,2,6,7]*German Research Center for Artificial Intelligence (DFKI)*
[1,2,6,7]*{viktor.eisenstadt|hardik.arora|klaus-dieter.althoff|andreas.dengel}@dfki.de*
[3,4,5]*Technical University of Munich*
[3,4,5]*{c.ziegler|j.bielski|langenhan}@tum.de*

Abstract. The main aim of this research is to harness deep learning techniques to support architectural design problems in early design phases, for example, to enable auto-completion of unfinished designs. For this purpose, we investigate the possibilities offered by established deep learning libraries such as TensorFlow. In this paper, we address a core challenge that arises, namely the transformation of semantic building information into a tensor format that can be processed by the libraries. Specifically, we address the representation of information about room types of a building and type of connection between the respective rooms. We develop and discuss five formats. Results of an initial evaluation based on a classification task show that all formats are suitable for training deep learning networks. However, a clear winner could be determined as well, for which a maximum value of 98% for validation accuracy could be achieved.

Keywords. Deep learning; spatial configuration; data representation; semantic building fingerprint.

1. Introduction

First approaches for computational support in solving architectural design problems were based on systems theory, which concentrated on design rules, but could not exhaustively represent the complexity of architectural designs (McCullough et al., 1990). Therefore, the second generation of the design methodology movement in the 1970s, represented by Horst Rittel among others, conceived of design not procedurally as the fulfillment of requirements, but rather as an individual process that can only be incompletely described.

In the research project Metis-I (2013-2016), we investigate approaches for the early conceptual design phases of architecture that can cope with the vagueness of architectural design problems. In this context, we investigate deep learning (DL) approaches for auto-completion of spatial configurations building on previous findings on search for design references from the research project Metis-II (2020-2023). After sketching first ideas of the the spatial configuration, rooms and their connections are suggested for completion to support the early ideation process. We envisage auto-completion to be helpful for architects to overcome their own design bias, by offering design options as an inspiration and to evaluate variants of the possible final results for deeper understanding of their own design decisions. Both projects are supported by German Research Foundation (DFG).

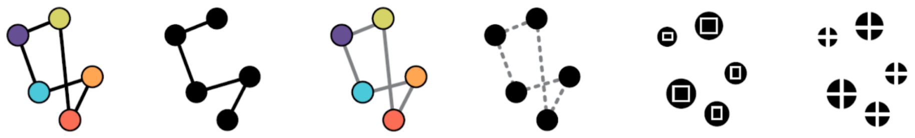


Figure 1. Examples of semantic building fingerprints (SBF). With topology (1-4): semantic spatial graph, through-path-graph, semantic spatial connection graph, spatial distance. Without topology (5, 6): envelope area, center of room.

Spatial information from the architects' sketches needs to be formalized to allow computational methods, such as DL, processing it. One approach for formalization is provided by the Semantic Building Fingerprint (SBF) (Langenhan 2017). It is comparable to the set of characteristic features in friction ridges of a human finger used for identification of human beings. SBFs are graph-based, rooms are mapped as nodes and the relationships between rooms, e.g. direct connections through doors, are represented by edges. Several SBFs with and without typology were developed (see Figure 1). For each SBF, a suitable representation for DL has to be elaborated. In this paper, we investigate such representations based on the common concept ”relation map” for the most complex SBF with the strongest set of semantic information, the so called *semantic spatial connection graph*. This SBF takes rooms and their names as node labels for the graph as well as their connection between each other e.g. by a wall or door as edge labels of the graph into account. In contrast to raster graphics of e.g. floor plans, the semantic information in SBFs is directly accessible, which is especially useful for semantic tasks in DL in the domain of architecture.

However, semantic information encoded as an SBF graph needs to be converted for processing with DL, as this AI discipline makes use of uniformly preprocessed data to draw conclusions based on the features detected in this data. That is, for an optimal use within a DL approach, e.g. an artificial neural network (ANN), spatial configurations need to be turned into specific numerical or textual representations. These DL-compatible representations of SBF should be designed to allow reversible conversion from DL to SBF. This makes sure that deep learning results can be visualized for domain users such as architects or urban planners.

We explore how SBF-based spatial configurations can be represented optimally in DL approaches for architectural design, especially in the state-of-the-art ANNs. We propose five types of the DL-compatible representation “relation map” and evaluate three most promising using a straightforward classification task.

2. Research Context and Related Work

Currently, the most applied generic tasks of DL methods include classification (assignment of predefined classes to a new data entity, e.g. recognition of objects in an image), sequence learning (e.g. completion of natural language sentences), and generative methods (e.g. "deep fakes"). Nowadays, deep learning has become a common tool for computer science applications and is used in many knowledge-intensive domains as the solution of choice to help in decision making (e.g., in the financial sector) or security enhancement (e.g. face recognition). However, for the domain of architectural design, deep learning methods and approaches are still not common and not widely applied as for the other domains.

Existing DL approaches in architecture mostly make use of images of floor plans in the form of raster or vector graphics, for example, for retrieval of similar designs (Sharma et al., 2017), design style manipulation (Newton, 2019), or 3D room layout estimation (Sun et al., 2019). While this might be the most obvious way of using floor plans in DL, such pure image-based data might not allow for recognition of the relevant semantic information, such as relational dependencies between the rooms that make up the purpose of the spatial configuration. Thus the images do not allow for easy and flexible selection of semantic features that should be taken into account for learning of a DL model.

This problem became visible during our work on an AI-based prototype for support of early phases in architectural design, for which a number of DL approaches should be developed that make use of relevant semantic information contained in the SBFs. For example, the aforementioned intelligent auto-completion of spatial configurations with ANNs is one of these approaches. The data available for these approaches missed a suitable representation of spatial layouts for DL as the data was put together from different sources and had only GraphML as a common SBF representation format. However, this XML-based format does not provide a numerical tensor format required for the modern DL libraries such as TensorFlow or PyTorch. These libraries implement the standard DL methods that need to be evaluated for the research project goals of auto-completion to either use them, evaluate other libraries, or develop new DL methods for the domain of architecture. For graphs, it is possible to use a dedicated deep learning library, such as *Deep Graph Library* DGL, that functions as an abstraction layer to other DL frameworks extending them with deep learning methods for graphs that have already shown promising results (Zhang et al., 2020). However, DGL does not provide architecture-specific methods or representations as well, making use of generic representations, such as adjacency matrices that also do not allow for selective application of relevant SBF features.

As a practical approach for using GraphML data for DL, methods to construct a floor plan image based on the adjacency graph of the spatial configuration using graph theory tools and rulesets can be used (Shekhawat et al., 2019; Roth, 1982). However, these methods do not make use of semantic information as well and produce rectangular and orthogonal spatial layouts concentrating on the geometry, that is currently not considered by any DL approach of the AI-based prototype (but is encoded in SBFs like envelope area or center of the room, see also Figure 1).

3. Data Representation: Relation Map

In search of an optimal solution to the problem described in the previous section, we developed a number of representations, which are all based on a tensor data structure called “relation map”. Before introducing the representations this chapter introduces the general underlying concept of the relation map.

A relation map is a tensor that represents *the modified adjacency matrix of the topology* of a spatial configuration encoded in a graph. Each row of the matrix represents *a particular room* and *its outgoing relations* to other rooms. Unavailable relations between the existing rooms and geometrical features of the building design are not considered. Initially, relation maps take inspiration from architectural morphospaces (Steadman and Mitchell, 2010) and geometry maps (De Miguel et al., 2019). Adjacency matrices have been already used as representation means of spatial layouts (Hickman and Krolik, 2009), however, the structure proposed by Hickman and Krolik only indicates if direct paths (connections) between the rooms are available (or not), making no use of any other semantic information.

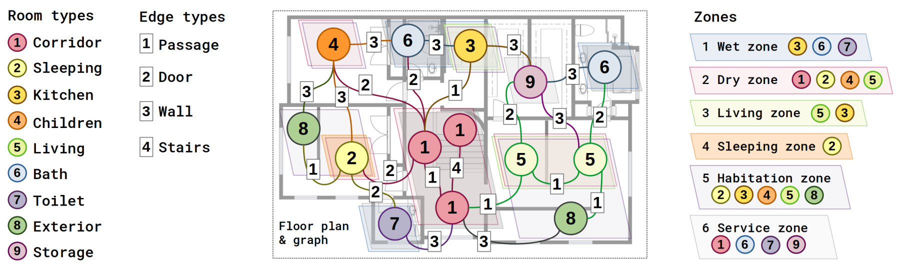


Figure 2. Basic topology for relation maps. An example of an SBF derived from a real spatial configuration graph shows how this typology can represent the semantic features.

As the relation maps make use of the semantic information rather than of the geometrical or raster graphics features, a specific typology of node and edge labels is required that can be used for standardized representation of the features. As mentioned in the introduction to this paper, the paramount features of the SBF semantic spatial connection graph, that we selected to work with, are its room and room connection types that correspond to nodes and edges of the room graph. However, generally, applying semantic typology for data that can be used as a graph it is possible to represent any type of information either as a node, an edge or their properties. In Figure 2, an example of such semantic typology is shown, in its basic version it consists of a number of room and relation types, each encoded with a specific number. The typology is flexible and can be easily complemented, extended, or reduced if required. For example, it can be reworked in a way that only an abstract categorization of rooms (e.g. habitable space) and connections (open or closed) is represented. An additional advantage, which is crucial at least for our research context, is that using the typology the relation map can be reconstructed back to GraphML or other graph representation format.

3.1. SIMPLE AND ZONED RELATION MAPS

The basic type of a relation map is a *simple relation map* that uses three numerical slots to encode the room type relations available in the SBF. Each relation between two rooms can be represented by a specific *relation code* and placed in the row that represents the room from which the connection goes out. For example, in the relation code 562, **living** (5) has an outgoing connection to **bath** (6) using a **door** (2). While such tensors are lightweight and easy to create, there are also limitations of their use caused by the occasionally repeating codes. The simple maps were integrated in a design augmentation approach (Arora et al., 2020) based on generative adversarial nets and related to the architectural design generation (As et al., 2018) and House-GAN (Nauata et al., 2020).

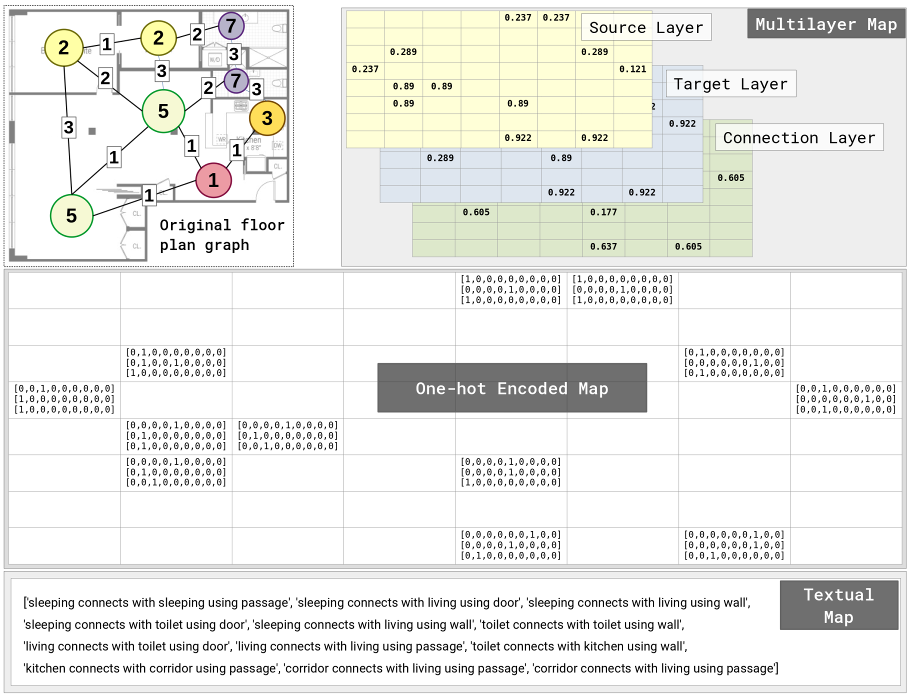
An improvement of the simple map is the *zoned relation map* that adds information on the architectural zones available in the SBF, making the maps more versatile and the relation codes rarely repeating. Architectural zones (Langenhan 2017) represent the taxonomy of categories for room types, grouping them by their functionality (see Figure 1). The room types can be part of multiple zones, for each room type a primary zone can be selected. Using a specific number, the primary zone can be used in the relation code. For example, the code 35162 represents **kitchen** (3) from the **habitation** zone (5) connected to **corridor** (1) from the **service** zone (6) by a **door** (2). Zoned maps were evaluated in the floor plan retrieval pipeline of the aforementioned AI-based prototype to provide contexts for the current search process (Eisenstadt et al., 2020).

3.2. MULTILAYER MAP

While the simple and the zoned type of relation map make use of a (basic) typology, in some cases the situation can occur when the dataset typology is not available. To use such datasets without typology, an approach is required that nevertheless is able to extract and encode semantic information to get relation maps for DL. To solve this case, we developed a specific type of relation map, the *multilayer map*, whose map conversion methods guarantee that the semantic information on room types and relations can be properly transformed into a relation map.

The basic approach of the multilayer map is the conversion based on hashing of values detected as possible room and relation types (e.g., via an attribute) rather than making a lookup in the typology. Based on cryptology techniques, hashing decodes byte information into a string of symbols of predefined size, e.g. to ensure that two files are identical, i.e. produce the same hash using the same hashing function. Applying the hashing methods to the room and relation semantics, given that the data is properly unified or was created with the same method, will result in the same string hashes for the same semantic entities. The hashes can then be decoded into integer or float values. For example, the MD5 hashing function always produces a hexadecimal string that can be converted into a decimal fraction.

A simple example of such conversion is the connection **living** <- **door** -> **bath** whose MD5 hash and the subsequent conversion from hexadecimal to decimal float will result in an array [0.89, 0.277, 0.637]. For the use in DL, a multi-layered 3D map can be constructed where the source room layer, target room layer, and the relation type layer provide a separate matrix for each of these semantic entity types. This relation map type is inspired by the color scheme channels in an RGB image. For DL purposes, the original 3D, as well as 2D representation of the map (where source, target, and connection values are accumulated into one value) can be used. Figure 3 shows an example of conversion of an SBF to a multilayer map.



				[1,0,0,0,0,0,0,0,0,0] [0,0,0,0,1,0,0,0,0,0] [1,0,0,0,0,0,0,0,0,0]	[1,0,0,0,0,0,0,0,0,0] [0,0,0,0,1,0,0,0,0,0] [1,0,0,0,0,0,0,0,0,0]		
	[0,1,0,0,0,0,0,0,0,0] [0,1,0,0,1,0,0,0,0,0] [1,0,0,0,0,0,0,0,0,0]					[0,1,0,0,0,0,0,0,0,0] [0,0,0,0,0,0,0,1,0,0] [0,1,0,0,0,0,0,0,0,0]	
[0,0,1,0,0,0,0,0,0,0] [1,0,0,0,0,0,0,0,0,0] [1,0,0,0,0,0,0,0,0,0]							[0,0,1,0,0,0,0,0,0,0] [0,0,0,0,0,0,0,1,0,0] [0,0,1,0,0,0,0,0,0,0]
	[0,0,0,0,1,0,0,0,0,0] [0,1,0,0,0,0,0,0,0,0] [0,1,0,0,0,0,0,0,0,0]	[0,0,0,0,1,0,0,0,0,0] [0,1,0,0,0,0,0,0,0,0] [0,0,1,0,0,0,0,0,0,0]					
	[0,0,0,0,1,0,0,0,0,0] [0,1,0,0,0,0,0,0,0,0] [0,0,1,0,0,0,0,0,0,0]			[0,0,0,0,1,0,0,0,0,0] [0,0,0,0,1,0,0,0,0,0] [1,0,0,0,0,0,0,0,0,0]			
				[0,0,0,0,0,0,0,1,0,0] [0,0,0,0,1,0,0,0,0,0] [0,1,0,0,0,0,0,0,0,0]		[0,0,0,0,0,0,0,1,0,0] [0,0,0,0,0,0,0,1,0,0] [0,0,1,0,0,0,0,0,0,0]	

['sleeping connects with sleeping using passage', 'sleeping connects with living using door', 'sleeping connects with living using wall', 'sleeping connects with toilet using door', 'sleeping connects with living using wall', 'toilet connects with toilet using wall', 'living connects with toilet using door', 'living connects with living using passage', 'toilet connects with kitchen using wall', 'kitchen connects with corridor using passage', 'corridor connects with living using passage', 'corridor connects with living using passage']

Figure 3. Original floor plan SBF and its conversions into multilayer map, one-hot encoded map, and textual map.

3.3. ONE-HOT ENCODED MAP

The conversion methods presented so far encode categorical data into numerical data and simple and zoned maps always summarize several pieces of semantic information in one numerical value. Depending on the position further left or further right in that number, the coded information has a larger or smaller influence on the numerical value. For example, in the current order of encoding in a simple relation map, the most significant digit encodes the source room, and the least significant digit encodes the relation. A change of the source room has a greater influence on the numerical value than a change of the relation. Also, there exists a risk that the assignment of categories to digits can be interpreted by the DL network as a ranking and a similarity measure for categories. In this case, using the typology shown in Figure 2, the difference between **storage** (9) and **corridor** (1) would be greater than between **toilet** (7) and **kitchen** (3), i.e. 9-1=8 vs. 7-3=4.

We introduce another representation, which is based on the classic *one-hot encoding* technique for categorical data (see Figure 3). In its simplest form, it is based on the simple relation map. Each field of the simple relation map is represented by three vectors, each of which encodes the category of the source room type, the target room type, and the relation type, respectively. Each vector has as many elements as there are categories for the entity type (room or relation type) with the highest amount of categories as required by DL. Each position in the vector corresponds to a category and can have either the value 1 (belongs to the category) or 0 (does not belong to the category). In each vector, the value 1 can appear exactly once because rooms and connections are uniquely assigned to categories. For this representation, the discussed influences do not play a role.

3.4. TEXTUAL RELATION MAP

The deep learning representations for spatial configuration presented in the previous sections try to encode the semantic information and varied room types as numerical values or as one-hot encoded vectors. The above-stated representations are usable as input tensors for a deep learning approach, but at the same time, these representations can be highly sparse and unadaptable. The text-based representation is designed to be adaptable to change in the number of room type or edge type categories while preserving the original shape of the tensors. Since the representation is text-based, different deep learning approaches such as text classification and sequence learning can also be applied to these representations.

The textual relation map is inspired by the simple relation map. In this representation, instead of encoding the semantic information using numerical values, sentences are used to represent a connection between two different rooms. For example, if two rooms are connected through a wall, then the textual formulation would be, **A connects with B using wall**. This representation also solves the problem of sparsity as for representing unavailable connection between two rooms instead of using 0 as the encoding, **‘no connection’** is used. Figure 3 shows an example conversion of a spatial configuration into a textual relation map (‘no connection’ omitted due to limited space).

4. Evaluation

To explore optimal data representations in deep learning for topological and semantic features of a SBF, a preliminary *two-phase evaluation* using the straightforward classification task was performed. Using classification of data samples was selected for the evaluation due to an array of reasons. It is easier to perform classification, instead of more complex tasks such as data generation or sequence learning, and also easier to evaluate the performance of the models due to the presence of a labeled dataset and the built-in evaluation methods in the DL libraries. Additionally, the results can be used to help reduce the computational cost of searching for similar floor plan contexts in the AI-based prototype.

A total of 200 housing floor plans were initially at our disposal, while for training a DL model, larger amounts of data are required. Due to the lack of such amounts of data, it was decided to generate variations of the already present spatial configurations using total of 15 different variation rules suggested by architecture domain experts. For example, the room types 'sleeping', 'living' and 'room' (the generic flexible type) can be freely exchanged in a one room apartment. The variationally generated floor plans were then checked by a specifically developed rule-based floor plan consistency checker to ensure that they adhere to the architectural rules removing from the final dataset those that do not pass the check. A total of 2544 housing floor plans made it to the final dataset.

For evaluation, the data set was labeled using a set of specific rules for the typology suggested by the architecture domain experts from the research project. The labels were divided in two different categories. The first category indicates the *number of habitable spaces* (i.e., sleeping, living, children, working, and room) in a spatial configuration. Again, the space 'room' was used a generic room to represent flexibility in the data. The second part indicates if the spatial configuration is *open* or *closed*, identified by whether there is *a passage between the kitchen and the living room*. Due to the above-stated labeling approach, it can be concluded that the floor plans have a strong correlation with the classes. The data set was separated into training and testing subset in relation 75% to 25%.

From the data representations presented in this paper, the multilayer map, one-hot encoded map, and the textual map were selected for the evaluation, as they were specifically developed for classification tasks using similar DL models and also make their debut in this paper. Other two map types (simple and zoned) were already evaluated before as mentioned in their respective sections.

In the first phase of the evaluation, a *multi-label classification* was performed using a total of 7 unique classes. Each spatial configuration of the dataset was assigned two labels, a number of habitable spaces, and if the layout is open or closed (e.g., "2 open"). In the second phase, a total of 10 unique classes was used for a *single-label classification* and the categories were combined into one single label for each building graph, resulting in, for example, "3closed". The idea behind performing two runs for the classification using a different number of labels was to observe if the deep learning model can learn different semantic features for different graph types and how the number of labels influences the performance of the corresponding representation.

For both phases, a common deep learning model in the form of a convolutional neural network (CNN) using the out-of-the-box Keras API in the TensorFlow framework was constructed. All representation variants of this common model differed only in the dimensions of the input layer and the rest of the DL model structure was identical for all three of them: 1 Conv2D layer, 1 BatchNormalization, 1 MaxPooling2D, 1 Dropout, 1 Flatten, and 2 Dense layers. However, through some tuning phases, hyperparameters, such as the learning rate, batch size, or the number of epochs, were set optimally for each representation to ensure the best performance. For example, the number of epochs of 100, 1000, 1500 and the batch size of 128, 24, 24 were used in the single-label evaluation for the one-hot encoded map, the multilayer map, and the textual map, respectively.

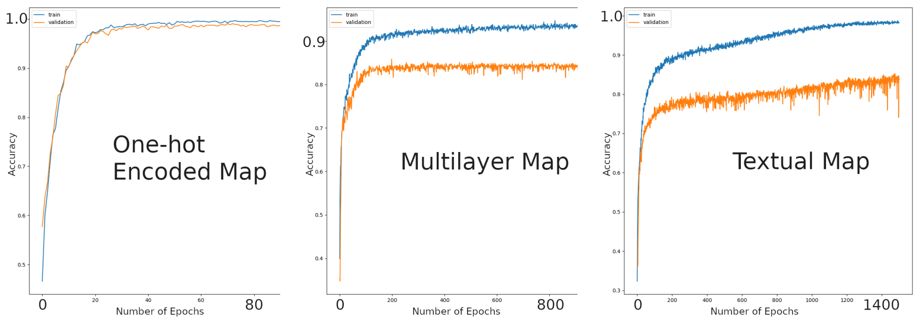


Figure 4. Results of the DL model training for the single-label classification. Blue curves represent the training accuracy, the orange ones the validation accuracy.

To evaluate the results of the model training it was decided to use the built-in evaluation methods implemented in the TensorFlow framework. In particular, the training and validation accuracy, as well as the training and validation loss were selected as the main performance metrics, as they provide the optimal overview of the stability and consistency of the models.

The results of both steps of the experiment revealed that the one-hot encoded map performed best in both classification phases and for all applied metrics. For the multi-label classification it achieved **96%** / **95%** on training and validation accuracy, in the single-label classification these values increased to **99%** and **98%** respectively (see Figure 4). The multilayer and textual maps performed worse in the multi-label classification: **90%** / **84%** for the multilayer and **90%** / **74%** for the textual map. During the single-label classification, the values improved: **93%** / **83%** for the multilayer map and **98%** / **83%** for the textual map (see Figure 4).

In terms of the loss calculation, the comparison of the representations provided a similar picture: the one-hot encoded map performed best and achieved the least loss value in both phases of the experiment. For the multi-label classification, this value was **1.4197** for training and **1.4876** for validation of the model. The multilayer map achieved the loss values of **1.5304** and **2.7472**, the textual map-based model achieved **1.4244** and **2.5508** accordingly. For the single-label evaluation phase, the one-hot encoded map could improve and achieve even lesser values: **0.0085** training loss and **0.0638** validation loss. However, other two models were able to improve as well: **0.0997** / **0.7089** for the multilayer map and **0.0382** / **1.0620** for the textual map.

One of the most significant observations of the evaluation was that the one-hot encoded map performed stable during the entire classification process, while the multilayer and the textual map representations provided different grades of instability during the training process (see Figure 4). Overall, it can be concluded that the one-hot encoded map performed better than the other two evaluated representations, achieving the first place in the first evaluation phase and confirming its performance in the second phase. On the high level, a tendency towards recognition of relevant semantic features within SBF-based relation maps could be observed, indicated by the overall training and validation numbers.

5. Conclusion and Future Work

In this paper, five different data representations for SBFs for use with common DL frameworks, all based on the common tensor-based structure "relation map", were presented. A classification task was performed to evaluate three of them. The results indicate that the DL models in the form of CNNs are able to process and understand the latent structure of the selected relation map types and the information contained in them. A max. validation accuracy of 98% was achieved for the winning type, the one-hot encoding map. For public use, the conversion methods are published under *https:github.com/metis-caad/roomconf-converter*.

In the upcoming experiments (to be submitted, e.g., to eCAADe 2021), relation maps will be much more comprehensively evaluated on further deep learning tasks (e.g., sequence learning or data augmentation) and examined on data not available in training or testing datasets. It is expected that there is no one-fits-all representation for all DL tasks, but the pros and cons of different relation map types for certain DL tasks can be outlined. We also intend to publish the data consistency checking methods used for the evaluation and discuss their limitations.

References

Arora, H., Langenhan, C., Petzold, F., Eisenstadt, V. and Althoff, K.D.: 2020, METIS-GAN: An approach to generate spatial configurations using deep learning and semantic building models, *ECPPM-2020/21*.

As, I., Pal, S. and Basu, P.: 2018, Artificial intelligence in architecture: Generating conceptual design via deep learning, *International Journal of Architectural Computing*, **16**(4), 306-327.

Eisenstadt, V., Langenhan, C., Althoff, K.D. and Dengel, A.: 2020, Improved and Visually Enhanced Case-Based Retrieval of Room Configurations for Assistance in Architectural Design Education, *ICCBR 2020*.

Hickman, G. and Krolik, J.L.: 2009, A graph-theoretic approach to constrained floor plan estimation from radar measurements, *IEEE transactions on signal processing*, **57**(5).

Langenhan, C.: 2017, Datenmanagement in der Architektur, *Doctoral diss., TU München*.

McCullough, M., Mitchell, W. and Purcell, P.: 1990, The Electronic Design Studio: Architectural Knowledge and Media in the Computer Era, *CUMINCAD*.

de Miguel, J.: 2019, Deep Form Finding-Using Variational Autoencoders for deep form finding of structural typologies, *CUMINCAD*.

Nauata, N., Chang, K.H., Cheng, C.Y., Mori, G. and Furukawa, Y.: 2020, House-GAN: Relational Generative Adversarial Networks for Graph-constrained House Layout Generation, *arXiv preprint arXiv:2003.06988*.

Newton, D.: 2019, Deep Generative Learning for the Generation and Analysis of Architectural Plans with Small Datasets, *CUMINCAD*.

Roth, J., Hashimshony, R. and Wachman, A.: 1982, Turning a graph into a rectangular floor plan, *Building and Environment*, **17**(3), 163-173.

Sharma, D., Gupta, N., Chattopadhyay, C. and Mehta, S.: 2017, Daniel: A deep architecture for automatic analysis and retrieval of building floor plans, *2017 14th IAPR ICDAR*, 420-425.

Shekhawat, K., Duarte, J.P. and thers, initials missing: 2019, A Graph Theoretical Approach for Creating Building Floor Plans, *CAAD Futures*, 3-14.

Steadman, P. and Mitchell, L.J.: 2010, Architectural morphospace: mapping worlds of built forms, *Environment and Planning B: Planning and Design*, **37**(2), 197-220.

Sun, C., Hsiao, C.W., Sun, M. and Chen, H.T.: 2019, Horizonnet: Learning room layout with 1d representation and pano stretch data augmentation, *IEEE CVPR*, 1047-1056.

Zhang, Z., Cui, P. and Zhu, W.: 2020, Deep learning on graphs: A survey, *IEEE Transactions on Knowledge and Data Engineering*.