Classifying Building and Ground Relationships Using Unsupervised Graph-Level Representation Learning

Abdulrahman Alymani, Andrea Mujica, Wassim Jabi and Padraig Corcoran
Cardiff University, UK
Alymaniaa@Cardiff.ac.uk

When designing, architects must always consider the ground on which their buildings are supported. Our aim is to use data mining and artificial intelligence (AI) techniques to help architects identify emerging patterns and trends in building design and suggest relevant precedents. Our paper proposes a novel approach to unsupervised building design representation learning that embeds a building design graph in a vector space whereby similar graphs have comparable vectors or representations. These learned representations of building design graphs can, in turn, act as input to downstream tasks, such as building design clustering and classification. Two primary technologies are used in the paper. First, is a software library that enhances the representation of 3D models through non-manifold topology entitled Topologic. Second, an unsupervised graph-level representation learning method is entitled InfoGraph. Result experiments with unsupervised graph-level representation learning demonstrates high accuracy on the downstream task of graph classification using the learned representation.

Introduction

Studies have shown that graphs are an effective representational tool for a significant variety of data, including architecture, urban and planning designs [1], [2]. A graph is a network of nodes and edges [3]. With the use of graphs, explicit information can be obtained not only from the general graph network but also from smaller units inside the same graph. Additionally, graphs provide an intuitive way of assigning properties to nodes and edges. Recent developments have seen a rapid rise in the study of graphs because they can model rich information, which is critical in numerous architectural applications. For example, researchers have

Design Computing and Cognition DCC'22. J.S. Gero (ed),
pp. xx-yy.

presented several graph theoretical approaches to the generation of architectural floor plans [4], [5]. Another application involves graph geometric measures and models for architectural planning to find the 'minimum-path graph', which constitutes a proposed analysis of all shortest-path traversals between any two locations in a network [2]. However, architectural works using machine learning usually consider 2D image-based representations of data. Recent graph research has focused on 3D topological supervised learning [6]. Despite this development, applying supervised learning approaches to graphs presents a challenge in that difficulties often occurs when collecting annotated label data. Furthermore, many data mining methods for tasks such as classification and clustering, demand that elements in the input data be fixed-length feature vectors. A graph in native form does not have such a representation and therefore such methods cannot be directly applied.

Unsupervised learning approaches are a promising method of overcoming the aforementioned limitations. Unsupervised learning focuses on unlabelled data and generates a representation, which can aid future downstream tasks, such as classification. For example, the unsupervised learning approach, like word2vec and Bert in the domain of natural language processing has demonstrated the potential of this approach.

The use of representation learning in the context of graphs corresponding to 3D meshes was examined by [7]. During the workflow, unsupervised deep learning of representation in a latent vector space classified the room types in design samples. Bouritsas and Bokhnyak utilised a research method to introduce a graph convolutional operator that explicitly models the inductive bias of the underlying fixed graph, implemented directly on the 3D mesh. The spiral operator enforces consistent local orderings of the graph's vertices, thus breaking the permutation invariance property existing in all prior work on Graph Neural Networks [8].

This paper aims to design a novel proof of concept workflow that applies unsupervised graph level representation learning to building/ground relationship data. The objective is to provide a vector for each graph to encode the relationship similarity between two 3D building/ground topological graphs. Our workflow is divided into two stages. The first stage uses the Topologic software library that enhances the representation of 3D models through non-manifold topology with embedded semantic information. The plug-in automatically and generatively creates a synthetic dataset of building and ground relationships with respective typological categories. A topological dual graph is then automatically generated by

labelling the geometric models. The dual graph of a building consists of one node for each space or element that are then connected by edges if they share a surface. The ground is segmented into a grid of cells and therefore several 'ground nodes' are created to represent it. Similarly, columns are segmented as well and thus multiple 'column nodes' are created. In the second stage, this dataset acts as input to run the unsupervised graph level representation learning for generating fixed-length feature vector representations of graphs.

By implementing this framework, a similar precedent can be introduced into the design process, which will allow designers to estimate the performance consequences of their choices quickly. As examples of this application, the designer can use this workflow to generate their new design, and then the machine learning models classifier can determine the type of design. This class information could then be used to retrieve other design precedents (from our 500-dataset archived) of the same type, which may inspire the designer.

Building and Ground Relationships

Building and ground relationships have long undergone discussion in the architectural field. For centuries, architects have used the ground as a reliable physical and conceptual support for their work. However, the notion of ground connections has improved with recent technological, philosophical, and geopolitical advances [9]. Modern architects respond to these conditions by inventing formal topologies. Several materials help provide a physical disengagement with the building form and the ground. For instance, the Convent of La Tourette by Le Corbusier facilitated a diverse range of approaches relative to the ground [10]. Other architects deconstructed the architectural objects by integrating the interior space into the surrounding landscape [11]. Contemporary architects have used similar methods to work with the ground; some have disregarded it, while others have focused on the division between landscape and building. However, T. Berlanda, a graphic lexicon, illustrates that buildings touching the ground are divided into three principal categories: Separation, Adherence and Interlock [12] (see Figure 1).

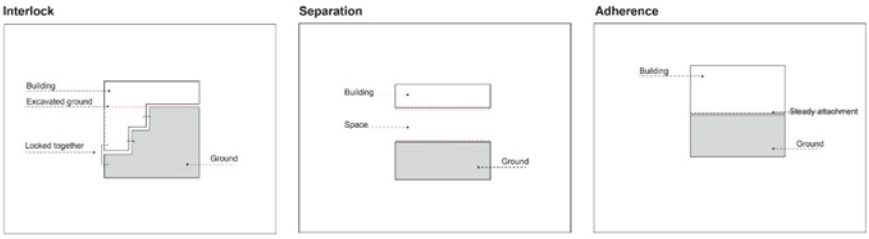

Fig. 1 The three main types of building and ground relationships are displayed.

Graph Representation Learning

Graph embedding represents a method of mapping a graph into a fixed-length vector that captures key features or properties of the graph. This approach means the graph embedding helps to translate a complex graph into a form that many popular machine learning, and data mining methods can use. Graph embedding machine learning models typically learn what is significant in an unsupervised generalised way. There are two major graph embedding types: Monopartite graphs, which have nodes with a single label, such as Deep Walk, and Multipartite graphs, which have nodes with several labels, such as the Knowledge Graph. Moreover, there are three different aspects of the graph that are trying to represent as embedding: 1) vertex embedding, which describes the connectivity of each node; 2) path embedding for traversal across the graph; and 3) graph embedding, to encode the graph into a single vector (see Figure 2) [3].

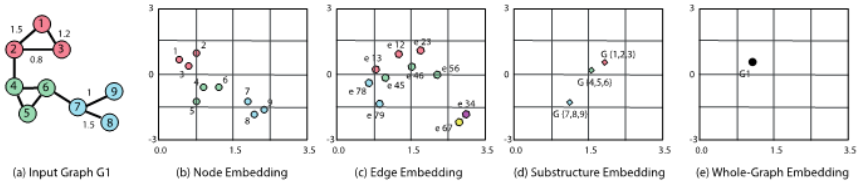

Fig. 2 Example of embedding a graph into 2D space with different granularities (author after [3]).

Unsupervised Graph Level Representation Learning (UGLRL)

In Graph Level Representation Learning, the graph is encoded into a single vector. Two graphs are said to be similar if they are represented by corresponding vectors that are embedded close together. A whole graph embedding provides a straightforward and efficient way of calculating graph similarities, which is essential to graph classification.

In 2020, Yun-Sun et al. introduced InfoGraph, a machine learning model that studies the representations of whole graphs in both unsupervised and semi-supervised scenarios [13]. The unsupervised InfoGraph accomplishes this task by providing both the number of nodes in and all graph matrix representations as The InfoGraph model focuses on graph neural networks (GNNs). Repeated aggregation of local neighbourhood node representations in embedding architectures produces node representations. By aggregating the features of neighbours, one can learn the representations of nodes, called patch representations. In a GNN, the READOUT function summarises all the obtained patch representations into a fixed-length graph representation.

Figure 3 illustrates the **UGLRL** model process: a) with graph convolutions and jumping concatenation, an input graph is encoded into a feature map; b) (global representation, patch representation) pairs are input to the discriminator, which determines whether they belong to the same graph; and c) InfoGraph generates all possible positive and negative samples using a batch-wise fashion. For example, consider the two input graphs in the batch and seven nodes in total (above). The global representation of the graph (A) will apply seven input pairs to the discriminator as well as the graph (B). In this case, the discriminator will take 14 (global representation, patch representation) pairs as input.

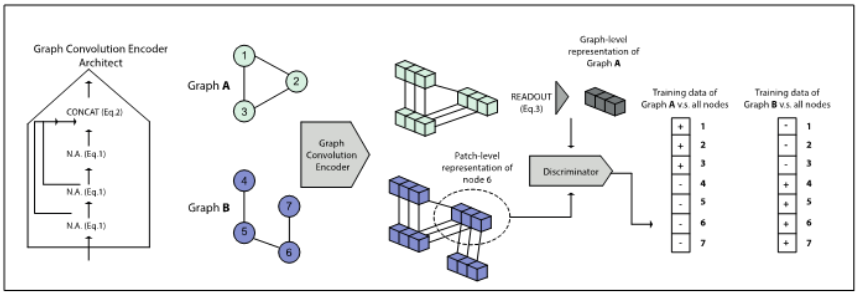

Fig. 3 Unsupervised graph-level representation learning model (author after [13].

Topology

The use of topology can help represent architectural designs, graphs and topological structures. The primary difference between geometry and topology is that the latter abstracts away the concepts of form and physical distance but retains the notion of connectivity. Consequently, one can explore complex designs at a much higher abstraction level than geometry allows. Different topologies can represent a multitude of designs [14].

Recently, Jabi discovered the potential of non-manifold topology in early design stages [15], by modelling architectural spaces using concepts from non-manifold topology [16]. This approach resulted in the Topologic toolkit[17]. Topologic is a software library that enhances the representation of space in 3D parametric and generative modelling environments. To date, the Topologic tool has shown its compatibility with Grasshopper [18], Dynamo [19] and Sverchok [20].

Topologic's classes include Vertex, Edge, Wire, Face, Shell, Cell, CellComplex, Cluster, Topology, Graph, Aperture, Content and Context.

This paper focuses on two essential features of Topologic for the proposed workflow: 1) the automatic derivation of 3D topological dual graphs using the Cell, CellComplex and Graph classes; and 2) the embedding of semantic information through custom dictionaries. In Topologic, a CellComplex is an enclosed 3D spatial unit (Cell) with a shared Face. The Graph class and associated methods are based on graph theory. A Graph comprises Vertices and Edges that connect Vertices to each other. A graph in Topologic is created with any 3D unit such as a CellComplex as input and outputs its dual graph as a network of labelled edges and vertices. The dual graph connects the centroids of adjacent cells with straight edges.

Dictionary data structures consist of key/value pairs. In computing, a key is any string identifying the data (e.g., 'ID', 'Type', 'Name'). Any type of data can be used as a key-value (e.g., floats, integers, strings). Topologic allows for the embedding of arbitrary dictionaries into any topology. Topologies are modified geometrically (e.g., by subdividing or creating a Cell Complex) so that dictionaries of operand topologies can be transferred to the resultant topologies. Using a topology, one can construct a dual graph by transferring dictionaries from constituent topologies to vertices. Using this capability, we can label the vertices of the dual graph.

Methodology

Recent work on graphs has focused on learning node representations or supervised learning tasks. These include graph analytic tasks, such as graph classification, regression, and clustering. However, such tasks typically require a fixed-length feature vector representing the entire graph. Graph-level representations can be derived implicitly through node-level representations. On the other hand, explicit graph extraction can be more straightforward and advantageous for graph-oriented tasks [13].

In this paper, the experimental workflow leverages two principal technologies. The first is Topologic, a library that enhances the representation of 3D models through non-manifold topology and embedded semantic information. The second is an unsupervised machine learning model that learns a representation of whole graphs as vectors for classification purposes. The experimental workflow involves two stages. In the first stage, an interactive system generates 3D prototypes of building and ground relationships with numerous topological variations. The architectural precedent's geometric models generated a semantically rich topological dual graph. In the second stage, the dual graphs were imported into the unsupervised graph-level representation learning model. The for graph classification. Using the results from the unsupervised embedding, the graphs were passed through a t-SNE (t-distributed Stochastic Neighbour Embedding) plot to visualise the locations and results of the classifications.

We applied the following four classifications methods to the vector representation learned by the InfoGraph method in an unsupervised manner. The performance of these classifications methods acts as a proxy for evaluating the usefulness of the representation in question.

1. Logreg, Logistic regression is used to describe data and its relationships with independent and dependent variables.
2. SVC, which stands for Support Vector Classification.
3. linear SVC, which is similar to SVC, but it generates a linear classifier; and
4. random forest, which is the result of the probability of an ensemble of decision trees.

Experiment Case Study

This experimental case study used Grasshopper and the Topologic plug-in to create 3D parametric models of buildings with different relationships with the ground and their respective dual topological graphs. Five architectural features generated the building/ground relationships: ground plate, building, columns, central core and plinth 'base'. The different features followed a set of rules. The ground plate was fixed in size. The plinth was then sized to be a certain percentage of the ground plate with equal offsets. We then placed the building geometries with the appropriate offsets and spacing. Buildings vary in height, but in one model, all building objects had the same height. A grid of cells also divided the building geometries internally. The models varied in the main building and ground relationship and were divided into three classes: 1) Separation; 2) Adherence; and 3) Interlock.

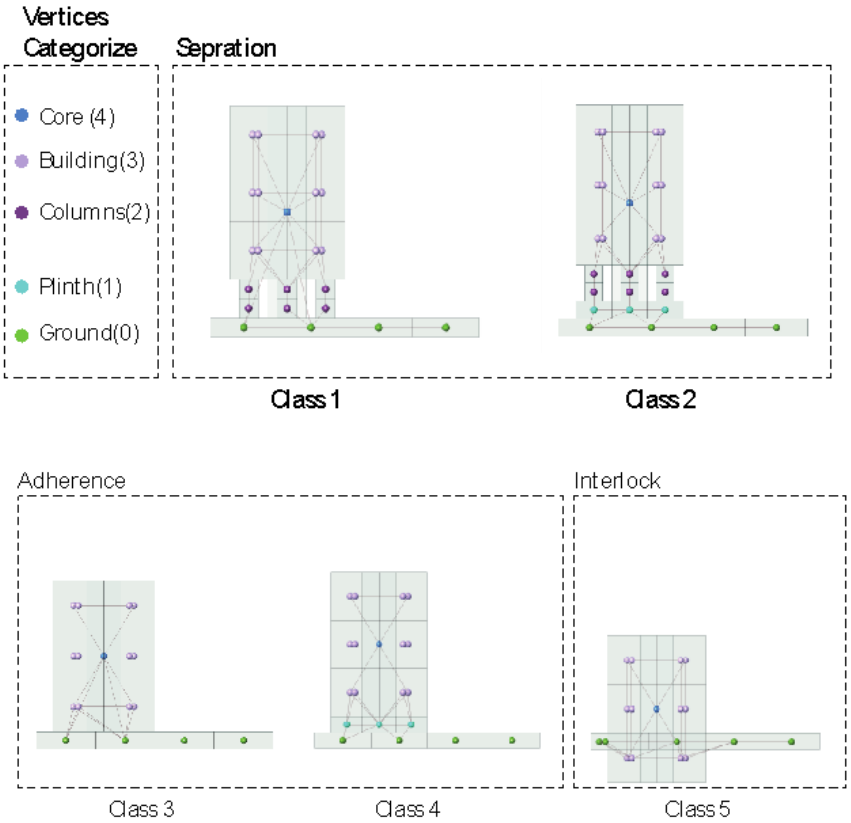

Fig. 4 Examples of the different classes of auto-generated building/ground configuration with associated dual graphs.

Completing three tasks created the dataset classes. The first task involves labelling the overall graph on the Grasshopper script. Firstly, separation follows five rules: ground, building, core, columns and plinth. Separation classes occur when the building is elevated from the ground with columns class (0) or when the building is elevated from the ground on the plinth and columns class (1). Secondly, Adherence follows four rules: ground, building, core and plinth. Interlocking follows three rules: ground, building and core. Adherence classes occur when the building is set directly on the ground class (2) or when the building is set on the plinth, which is set on the ground directly in class (3). Thirdly, interlock follows three rules: ground, building and core. Interlock classes occur when the building integrates and overlaps with the topology of the ground class (4). The second task involves the vertices. In our dataset, the vertices were classified into five categories: 0) Ground, 1) Plinth, 2) Columns, 3) Building, and 4) Core (see Figure 4). The last step involves integrating the visual dataflow definition with a custom Python script to convert Topological 3D dual graphs into text files in the InfoGraph format.

The dataset produced consisted of 900 graphs, as follows (see Figure 5):

- A total of 720 separation graphs comprising 90 building graphs separated from the flat ground with small, medium, and large columns; 90 building graphs separated from the flat ground with small, medium, and large columns and set into the plinth; 270 building graphs separated from the sloping ground with small, medium, and large columns; 270 building graphs separated from the flat ground with small, medium, and large columns and set into the plinth.
- A total of 96 adherence graphs comprising 12 building graphs set directly into the flat ground; 12 building graphs set on the plinth then into the flat ground; 36 building graphs set directly into the sloping ground; 36 building graphs set on the plinth then into the sloping ground.
- A total of 108 interlock graphs comprised 36 building graphs interlocked with the flat ground and 72 building graphs interlocked with the sloping ground.

Utilizing grasshopper, topologic and python script, we created our workflow that was used to develop the building and ground relationship iterations. The workflow contains five stages (see Figure 6). The first stage was to create the building and ground geometry. The second stage was to slice the created geometry with curves into different cells. The third stage was to feed the geometry into the Topologic tool, so the geometry transferred from geometry to topology. The fourth stage was to implement a dual graph to the created topology. Finally, we created two python scripts, one to repeat all the iterations and the other to transfer the dual graph to the required format.

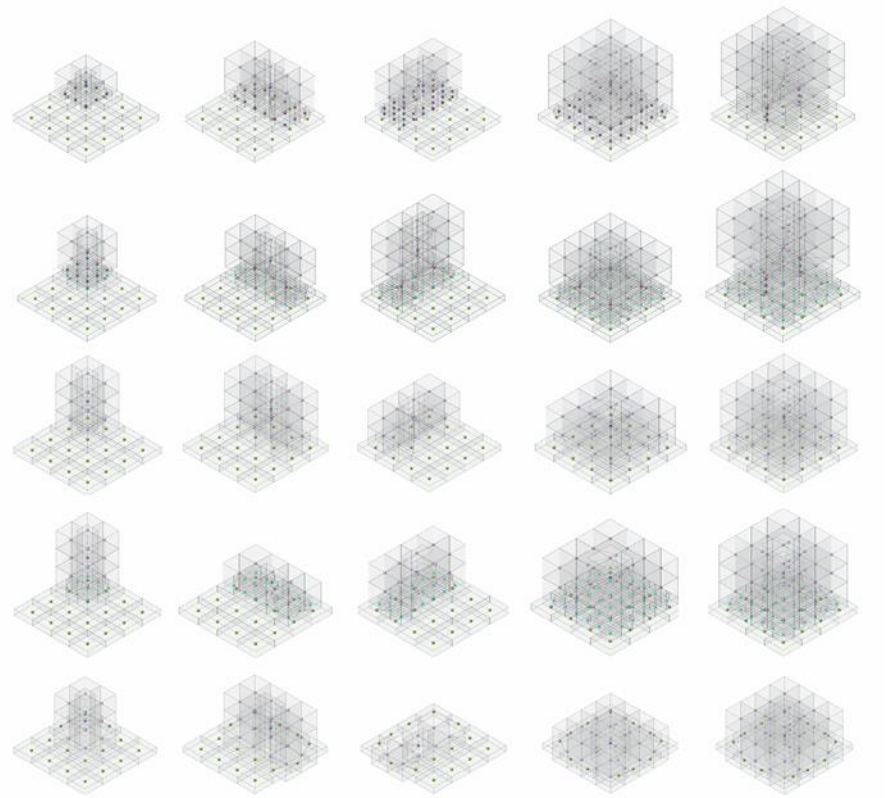

Fig. 5 Sample of automatically generated building/ground relationship typologies.

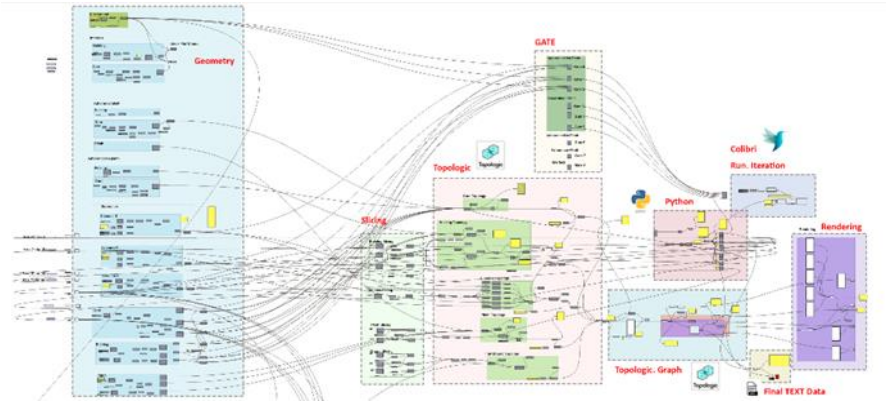

Fig. 6 An example Grasshopper script of 3D models generated using parametric and their associated topological dual graphs.

The process of data creation resulted in numerous findings. The total number of vertices was 55,081. The average number of vertices per graph was 61. The minimum number of vertices per graph was 20. The maximum number of vertices per graph was 197. The total number of ground vertices in the data was 31,772, the total number of building vertices in the data was 18,626, the total number of plinth vertices in the data was 10,689, the total number of columns vertices in the data was 16,923 and the total number of core vertices in the data was 900.

We prepared the following three text files as input to InfoGraph which is the unsupervised graph representation learning model used.

1. DS_A.txt: Adjacency matrix for all graphs, in which each line corresponds to (row, columns) for (node_id, node_id).
2. DS_graph_indicator: The column vector of graph identifiers for all nodes of all graphs, the value in the i-th line is the graph_id of the node with node_id i.
3. DS_node_labels: The column vector of node labels, the value in the i-th line corresponds to the node with node_id I.

It is crucial to mention that the data that worked with InfoGraph comprised undirected graphs, so all the edges were bidirectional. Additionally, the nodes needed to be in a continuous list, starting from 1 to 55,081 (the number of nodes in the dataset). (Figure 7) below explains the different files' connections and what each text file line represents.

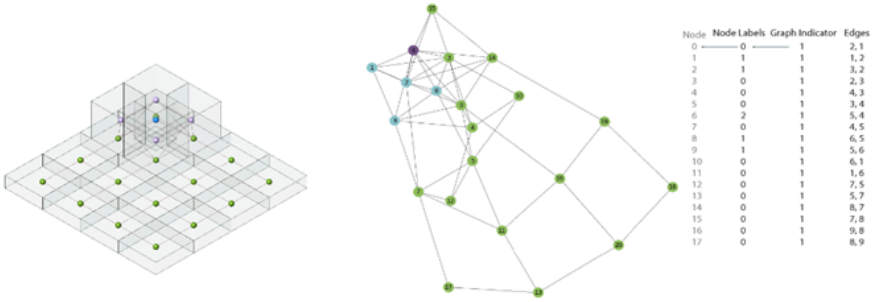

Fig. 7 Graph 1 from data. Visualised with matplotlib. The table shows how an adjacency matrix, node labels and graph indicators work within the text files.

(Figure 7) clarifies the representations of the topological graph within the 3D environment and within the machine learning software. While the nodes in the original dual graph (Left of Figure 7) are placed at the centre of the model elements within 3D space, the topological graph (middle of Figure 7) and its textual representation (Right of Figure 7) is geometry-independent and thus the XYZ coordinates of the original dual graph nodes are not needed.

The code for InfoGraph was created to run using the TUDataset module from Pytorch Geometric. Therefore, the input data required conversion into the same format as all the TUDatasets. The original data containing all 900 graphs were created to work with the format from the DGCNN [22]. Using the dataset MUTAG as an example, a code generated an adjacency matrix for all 900 graphs (see Figure 8).

For our experiment, we maintained the default Graph Isomorphism Network (GIN) model [13].

Experiment Results

Fer the experiment results below, and according to [13], we varied the following hyperparameters: learning rate, number of epochs and batch size. To visualise the graph, a t-SNE plot embedded the whole graph into a 2D space for graph visualisation, where each graph becomes a point. Providing human insights into the dataset facilitates further analysis of the data. The code was run in CPU mode in a Dell XPS Intel Core i7 with 16 GB RAM.

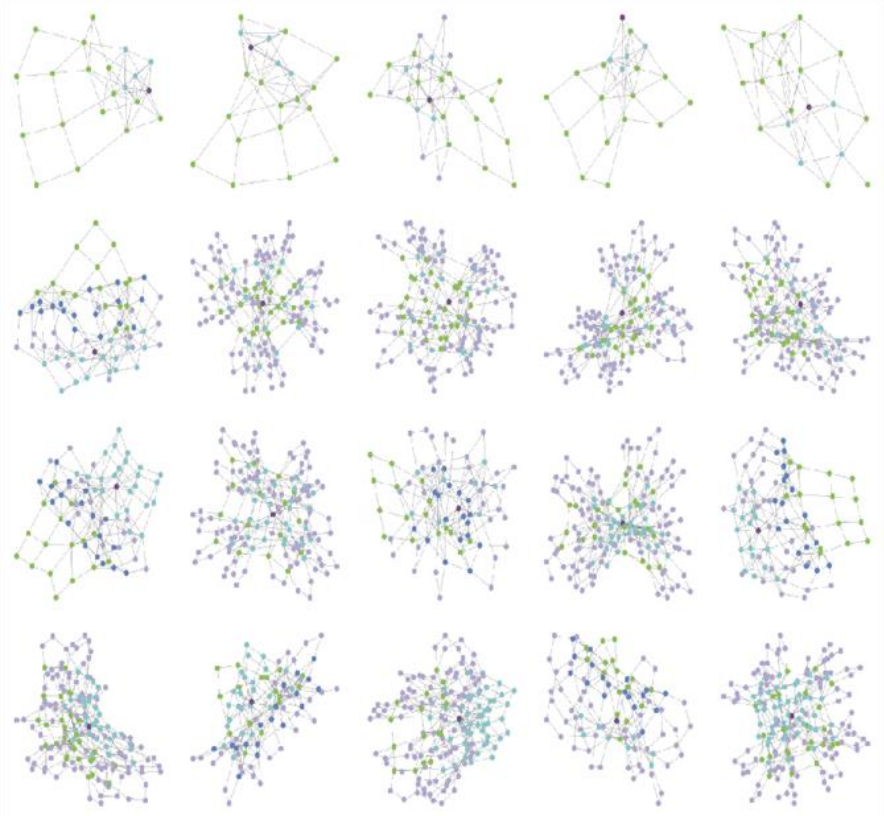

Fig. 8 First 25 graphs of the data were created and visualized using network X and matplotlib.

Learning Rate

In a convolutional neural network, the learning rate represents the rate at which the model parameters are updated each time an optimisation step occurs. Varying the learning rate can affect the model performance. We experimented with four different learning rates (1e-5, 1e-4, 1e-3 and 1e-2) and documented the results (see Table 1). Initially, the test had a fixed batch size of 128 and epochs set at 20. All four runs achieved high representation learning accuracy when used in the downstream classification task. The highest prediction accuracy result (98.4%) was achieved through a learning rate of 1e-4 (see Figure 9). To choose the learning rate for the next experiments, the t-SNE plot for the last epoch of each run underwent examination. All the t-SNE plots have a clear distributed representation of

every group of graphs (see Figure 10). Therefore, the best accuracy (98.4%) with the learning rate of 1e-4 was selected to test other hyperparameters in subsequent experiments.

Table 1 Accuracy results using various learning rates.

Learning rate	**Mutual information loss (MI)**	**Accuracies**			
		logreg	**Svc**	**Linear svc**	**Random forest**
1e-5	20148.611	0.981	0.986	0.980	0.986
1e-4*	3774.354	0.984	0.985	0.987	0.978
1e-3	488.991	0.972	0.985	0.983	0.977
1e-2	142.172	0.963	0.987	0.988	0.975

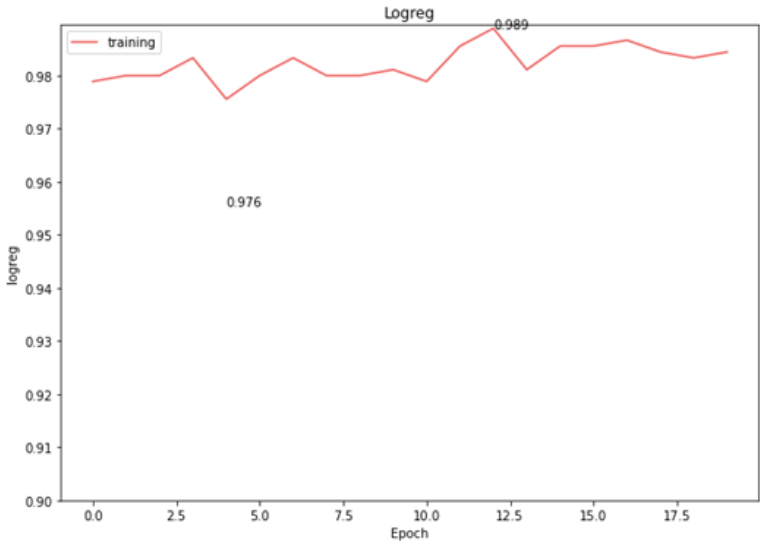

Fig. 9 Best Learning rate (1e-4) and the number of epochs (20) performance.

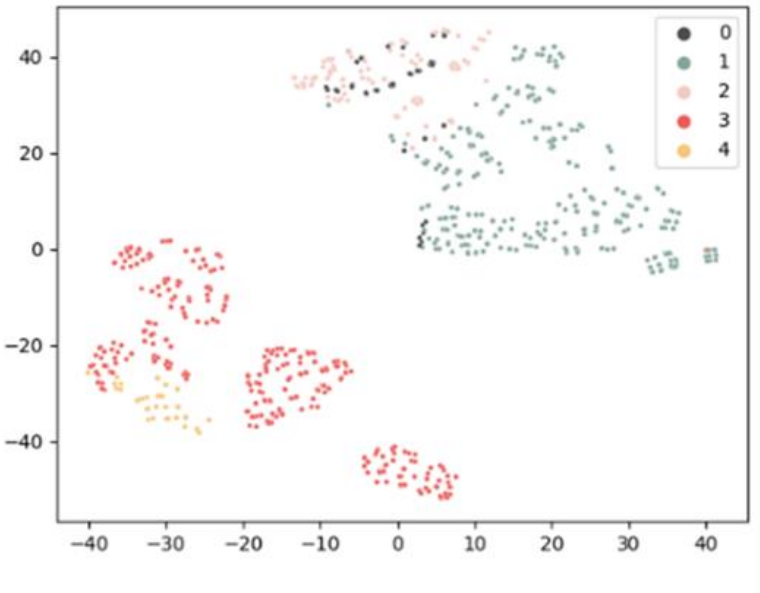

Fig. 10 Best t-SNE plot for learning rates (1e-4) and the number of epochs (20).

Number of Epochs

Epochs refer to the number of complete training dataset iterations. The number of epochs is an important hyperparameter to optimize because too low or too high a value can result in under and overfitting respectively. We experimented with several epochs while maintaining a testing rate of 1e-4 (see Table 2). The representation learning model accuracy results were stabilised with 10, 20 and 50 epochs. Exceeding the 50 epochs resulted in lower accuracy. Therefore, we used the t-SNE plot to examine the distributed representation of every graph. According to the t-SNE plot (see Figure 10), we chose the run with 20 epochs to continue testing other hyperparameters in subsequent experiments.

Table 2 Accuracy results using various numbers of epochs.

Number of epochs	Mutual information loss (MI)	Accuracies			
		logreg	Svc	Linear svc	Random forest
10	5668.682	0.984	0.987	0.99	0.984
20*	3774.354	0.984	0.985	0.987	0.978
50	909.848	0.984	0.988	0.983	0.980

Batch Size

The gradient descent batch size controls the number of training samples to iterate through before updating the model's internal parameters. We maintained a 1e-4 learning rate and 20 epochs for this last hyperparameter experiment. We experimented with three different batch sizes of 32, 64 and 128, respectively (see Figure 11). All the experiments achieved high accuracy; therefore, the t-SNE plot underwent examination to see the best-distributed representation of each graph (see Figure 12). Moreover, we documented the total processing/run time to see how the batch size affects the run time. The total time was neglectable for two reasons. Firstly, the data is limited and secondly, unsupervised representation learning takes less time than its supervised equivalent.

Table 3 Accuracy results using various Batch sizes.

Batch sizes	Accuracies				Total processing time
	logreg	Svc	Linear svc	Random forest	
32*	0.978	0.987	0.984	0.980	1:29:58
64	0.981	0.982	0.986	0.984	1:53:52
128	0.984	0.985	0.987	0.978	01:25:53

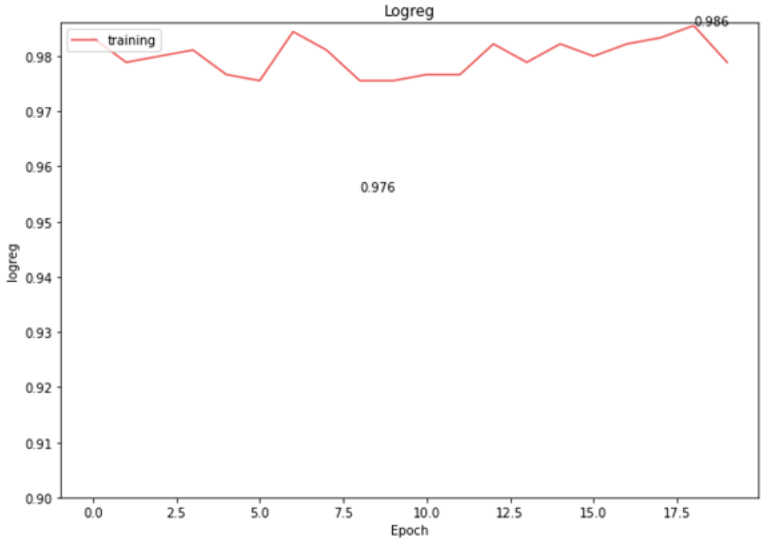

Fig. 11 Best batch size performance (32 batches).

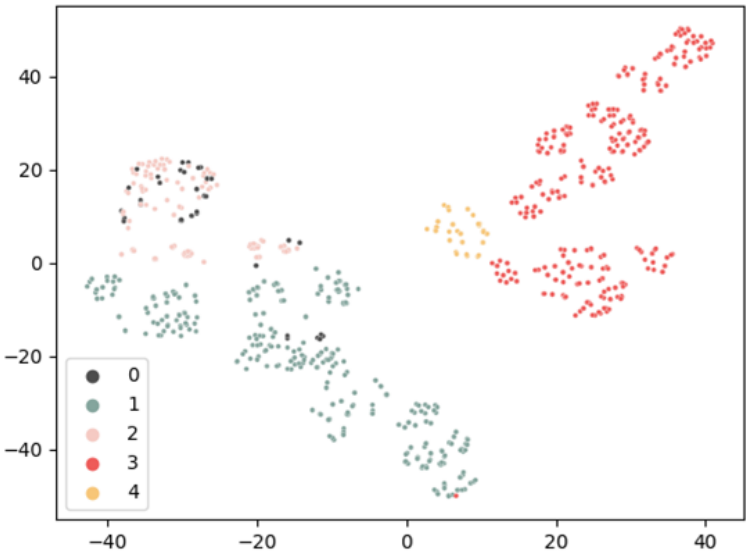

Fig. 12 Best t-SNE plot for 32 batches.

Conclusion

This paper aimed to determine the possibility of classifying architectural building/ground relationship forms through a novel workflow that uses unsupervised graph representation learning on 3D graphs rather than on 2D images. We leveraged a sophisticated topology-based 3D modelling environment to develop dual graphs from 3D models and label them automatically. We then fed those graphs to an unsupervised graph representation learning system. To discover the best accuracy rates, we experimented with different hyperparameters, such as learning rates, the number of epochs and the number of batches.

At the conclusion of our experiments, we found that all the experiments achieved high representation learning with more than 98% accuracy for all four different accuracy measures, namely Logreg, Svc, Linear svc and Random Forest. Our approach illustrates strong promise for recognising architectural forms using more semantically relevant and structured data. A novel workflow will undergo comparison to other approaches and will be tested with other datasets and labelling schemes in future work.

The findings have identified several new research areas. We will first try to classify nodes instead of just the overall graph. Secondly, we plan to classify the same dataset with a semi-supervised graph level representation learning approach and compare the results with this paper. Finally, this paper is part of ongoing PhD research devoted to developing a system that may recognise the topological building/ground relationships designers may build in near real-time and suggest precedents from a visual database.

References

1. G. Franz, H. A. Mallot, and J. M. Wiener, “Graph-based models of space in architecture and cognitive science- a comparative analysis,” *Proc. 17th Int. Conf. Syst. Res. Informatics Cybern.*, pp. 30–38, 2005.
2. N. Napong, “The Graph Geometry for Architectural Planning,” *J. Asian Archit. Build. Eng.*, vol. 3, no. 1, pp. 157–164, 2004.
3. H. Cai, V. W. Zheng, and K. C. C. Chang, “A Comprehensive Survey of Graph Embedding: Problems, Techniques, and Applications,” *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 9, pp. 1616–1637, 2018..

4. J. Gilleard, “Layout—A Hierarchical Computer Model for the Production of Architectural Floor Plans,” *Environ. Plan. B Plan. Des.*, vol. 5, no. 2, pp. 233–241, 1978.
5. K. Shekhawat, Pinki, and J. P. Duarte, “A Graph Theoretical Approach for Creating Building Floor Plans,” *Commun. Comput. Inf. Sci.*, vol. 1028, pp. 3–14, 2019.
6. W. Jabi and A. Alymani, “Graph Machine Learning Using 3D Topological Models,” in *SimAUD*, 2020.
7. I. As, S. Pal, and P. Basu, “Artificial intelligence in architecture: Generating conceptual design via deep learning,” *Int. J. Archit. Comput.*, vol. 16, no. 4, pp. 306–327, 2018.
8. G. Bouritsas, S. Bokhnyak, S. Ploumpis, S. Zafeiriou, and M. Bronstein, “Neural 3D morphable models: Spiral convolutional networks for 3D shape representation learning and generation,” *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2019-Octob, pp. 7212–7221, 2019.
9. Z. T. Porter, “Assorted Grounds -.” [Online]. Available: https://www.zacharytateporter.com/assorted-grounds. [Accessed: 25-Dec-2018].
10. F. Samuel, *Sacred concrete : the churches of Le Corbusier*. Basel: Basel : Birkhauser, 2013.
11. D. Leatherbarrow, “Topographical Stories : Studies in Landscape and Architecture.” University of Pennsylvania Press, Philadelphia, 2004.
12. T. Berlanda, *Architectural Topographies: a graphic lexicon of how buildings touch the ground*. 2014.
13. F.-Y. Sun, J. Hoffmann, V. Verma, and J. Tang, “InfoGraph: Unsupervised and Semi-supervised Graph-Level Representation Learning via Mutual Information Maximization,” *ICLR 2020*, no. 2019, pp. 1–22, 2019.
14. W. Jabi, “Linking design and simulation using non-manifold topology,” *Archit. Sci. Rev.*, vol. 59, no. 4, pp. 323–334, 2016.
15. W. Jabi, R. Aish, S. Lannon, A. Chatzivasileiadi, and N. M. Wardhana, “Topologic A toolkit for spatial and topological modelling,” 2018.
16. R. Aish, W. Jabi, S. Lannon, N. Wardhana, and A. Chatzivasileiadi, “Topologic: tools to explore architectural topology,” *AAG 2018 Adv. Archit. Geom. 2018*, no. September, pp. 316–341, 2018.
17. “Topologic.” [Online]. Available: https://topologic.app. [Accessed: 01-Dec-2021].
18. “Grasshopper.” [Online]. Available: https://www.grasshopper3d.com. [Accessed: 01-Dec-2021].
19. “Dynamo.” [Online]. Available: https://dynamobim.org. [Accessed: 01-Dec-2021].
20. “Sverchok.” [Online]. Available: https://www.blender3darchitect.com/modeling-for-architecture/getting-started-with-sverchok-for-3d-modeling/. [Accessed: 01-Dec-2021].