Composing Frankensteins: Data-Driven Design Assemblies through Graph-Based Deep Neural Networks

IMDAT AS
University of Hartford

SIDDHARTH PAL
Raytheon BBN Technologies

PRITHWISH BASU
Raytheon BBN Technologies

Over the last five years, machine learning and AI became exceedingly popular due to significant developments in the study of deep neural networks (DNN) or deep learning. Current research on DNNs focuses mainly on image and audio-based applications, ranging from self-driving cars, to virtual assistants such as Siri, to all kinds of online recommendation systems. Example applications include image classification, prediction of user choices, and generation of new images. In this paper, we present an alternative graph-based DNN approach to generate new conceptual designs. We trained DNNs with residential design represented through attributed graphs. We discovered essential building blocks based on performance criteria and composed them into new user-desired assemblies - aided by learned information about the proximity of various design components in latent vector space.

INTRODUCTION

There have been extensive developments in the field of deep learning over the last few years: Deep neural networks (DNN) have been successfully used on a wide range of real-world applications. In contrast to rule-based systems, DNNs do not need to be programmed upfront, but can decipher rules through examining large amounts of data.[1] For example, one can train a DNN with millions of cat images and use it to label cats in new images. This is especially important for self-driving cars, where the discrimination of objects, such as other cars, trucks, walking/biking people, etc. in real-time video feeds will make the difference between cars safely manoeuvring through traffic or not.

In this paper, we investigate a novel application of deep learning to generate conceptual designs for architectural projects. Unlike current research that focuses mainly on image, audio, or text-based DNNs, we explored graph-based DNNs. We trained DNNs to dissect home designs into essential building blocks and re-compose them into new assemblies. Our early results reveal that DNNs are capable of extracting function-driven building blocks from design data. We merged desirable building blocks into larger structures using tools from graph-theory and proximity information of nodes in latent vector space. While not a focus of the current research, we propose methods of evaluating generated new compositions with human input through crowdsourcing, e.g. Mechanical Turk. The evaluations could be used as training data to employ neural networks to define user-based fitness functions that over time can qualitatively evaluate generated designs.[2]

METHODS AND TECHNIQUES

DNNs are typically trained with significant amounts of real-world data. They do not need to be programmed upfront but can learn patterns and relationships through the training process with a given dataset, i.e. "training set." This is in contrast to traditional methods of generative design, such as shape grammars, where one can generate new compositions through applying certain prescribed design rules. In order to perform such task, however, one has to know all building blocks and set of rules needed to generate design variations.[3,4] For example, a discursive shape grammar has been put forth to mimic Alvaro Siza's style of architecture and generate variations of his Malagueira houses.[5] DNNs, however, do not require one to set up rules beforehand, but can simply detect them through the training process. In other words, if there are enough samples of Siza's work in the training set, a DNN can learn known or latent rules that Siza may have intentionally (or intuitively) applied in his work.

For this study, we used a neural network originally developed to identify essential building blocks of chemical compounds, which are strongly correlated with certain properties of the compound.[6] Here the chemical compounds are represented as graphs, i.e. nodes are atoms and edges are bonds, and the building blocks are subgraphs. We used a supervised graph convolutional neural network to discover these subgraphs based on specific functional criteria. Once we unearthed the subgraphs, we studied methods to merge them into larger assemblies. We then developed techniques to close open-ended nodes resulting during the merging process. We also propose methods to add and potentially augment these assemblies with additional auxiliary connections.

Figure 1 illustrates the DNN architecture that was used to discover the building block subgraphs. In each layer of the DNN, convolution is performed on the attributes of each node in

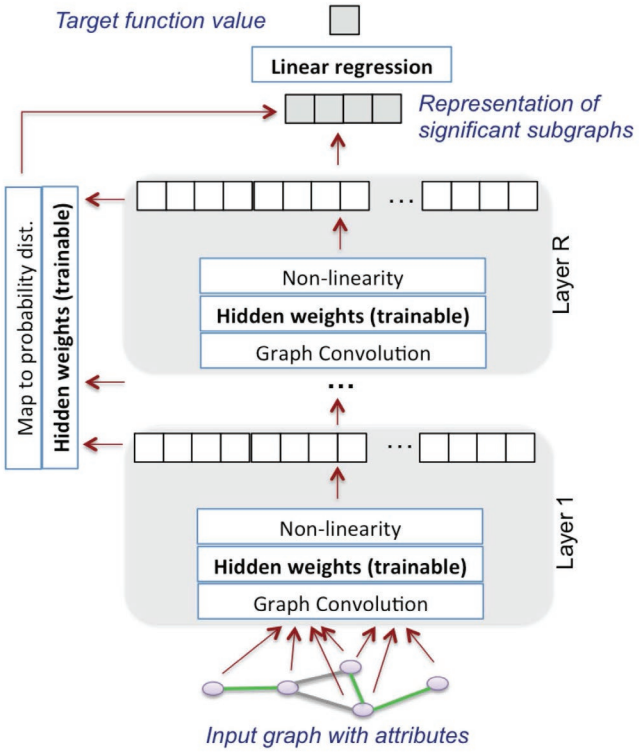

Figure 1. DNN architecture for discovering building blocks, i.e. subgraphs, from complete home designs.

the graph, by essentially summing the attribute values in its neighbourhood in the graph, and then the result is multiplied by hidden weights corresponding to that layer and some non-linearities before being mapped to a probability distribution. The probabilities in each layer contribute to a latent representation vector of the attributes of the specific input graph, which in turn is correlated with the target function score via linear regression. For each increasing layer in the DNN, contributions from larger neighbourhoods of each node are accumulated and remembered within the "hidden weights." Thus, if subgraph patterns are found among nodes with certain types of attributes in an R-hop neighbourhood of a node and these patterns correlate strongly with a high target functional score, the DNN will remember them in the hidden weights of the R-th layer after the training phase converges with a low value of root mean square error (RMSE). Then, for a given r, we output subgraphs of radius r that have a high linear regression coefficient and a high activation score in the r-th layer neurons in the DNN. These subgraphs are expected to correspond to building blocks of commensurate size that correlate strongly with high target functional scores.

Workflow of research: Figure 2 illustrates the research workflow. On top, design data is fed into a DNN (#1), which uncovers latent building blocks (#2) and maps nodes into high-dimensional vector space (#3). These essential building blocks are then merged into new compositions (#4). The resulting designs are put through an evaluation process, where valid designs are identified. In the following section, we discuss a case study.

DECOMPOSING AND RECOMBINING HOME DESIGNS

We trained the DNN with 15 home designs. Traditionally, architecture is represented through drawings, e.g. plans, sections, etc., or through more sophisticated and information-rich building information models (BIM). However, for this study, we represented architectural design using attributed graphs. We focused on the representation of essential elements of architecture, i.e., spaces (or rooms) of various types and their adjacency relationships that tend to occur in real conceptual design. We collected design data from BIM models and converted them into graphs in the following manner:

- nodes represent particular room types, e.g. bedroom, bathroom, etc., with attributes like area, volume, perimeter, etc.
- edges between nodes represent the connection type between rooms, e.g. a door connection, open connection, and vertical connections, e.g. stairs, ramps, etc.

Producing graphs from 3D models: We extracted design data from BIM models through a Python plugin developed for Autodesk Revit. We queried the digital models within the Revit API, and generated graphs displaying available node and edge conditions. We annotated the type of rooms, the type of relationship between rooms, and the evaluation scores based on various functional performance criteria, such as human-provided scores for livability (rating how well the living/family spaces were designed), or sleepability (rating how well the bedroom quarters were designed), and so on. Even though we limited our annotations to this narrow set of attributes, graph representations can be easily expanded with additional data, such as type of furniture, lighting fixtures, color, etc. In order to represent such more detailed information, one would need to create auxiliary nodes that show the containment relationship within a subgraph. In short, graph representations can be expanded to contain more details, if those details are available.

Evaluating the training set: In order to set up the DNN, we divided home designs into two datasets, one for training and the other for testing. The training set consisted of twelve homes and the testing set of the remaining three. We trained the DNN with both design data on homes and their corresponding performance scores on livability. We then implemented a regression test on the remaining three

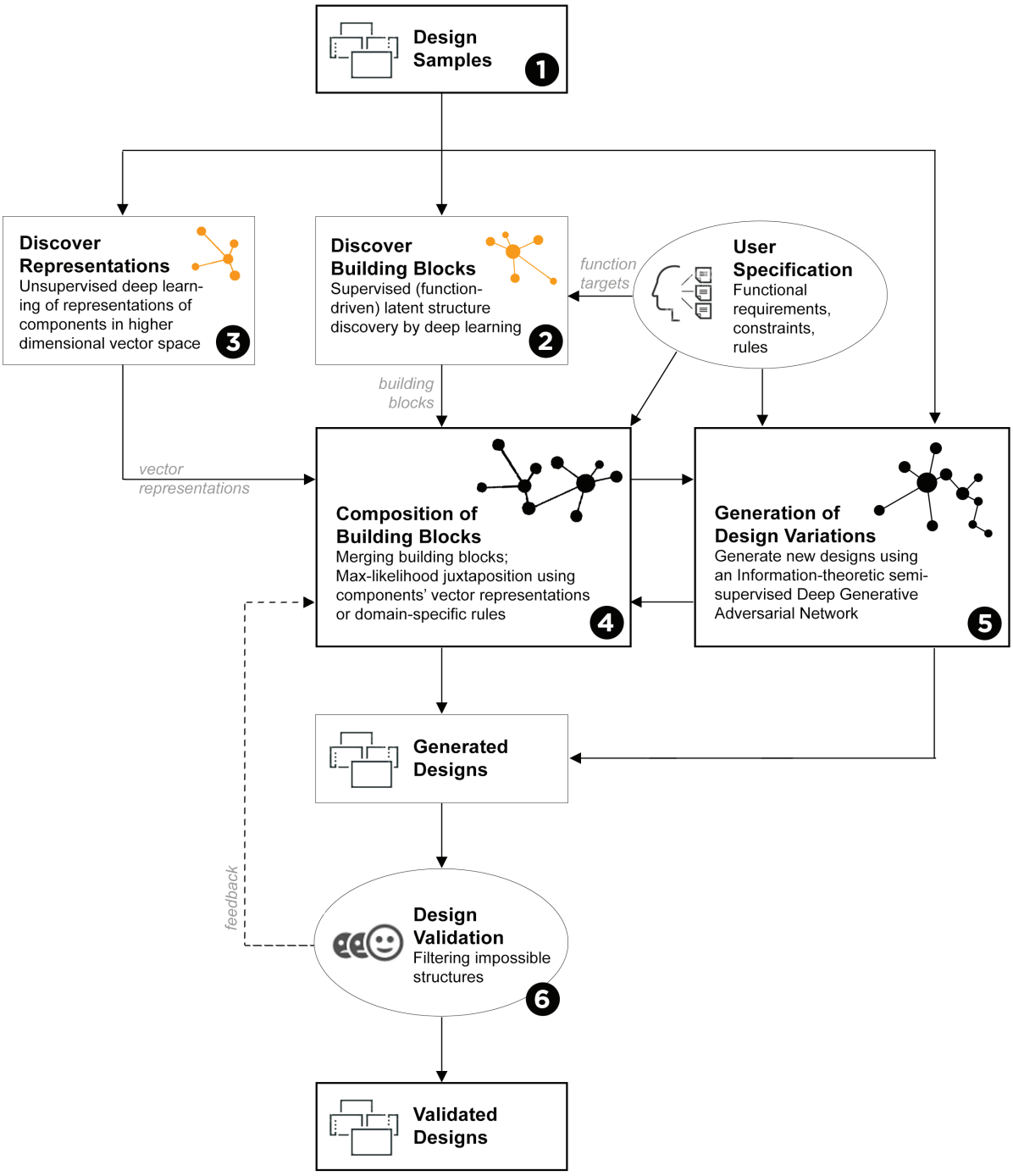

Figure 2. Workflow diagram of the deep learning system.

homes, which the DNN had not encountered before. The original livability scores of the three homes were 51, 32, and 67 (on a scale of 1-100). The DNN predicted them as 51.2, 24.5, and 67.2 in the test. The original scores we gave were based on subjective evaluations given by reviewers, and therefore, it was interesting to see that the DNN was able to predict them with such accuracy.

Extracting essential building blocks: Afterwards, we used the DNN to identify subgraphs that responded well to the particular functional performance criteria, i.e. to detect essential function-driven building blocks. For example, the system detected the following building block as a high performing subgraph responding to livability: ['2_Kitchen_82', '2_Foyer_39', '2_Pantry_26', '2_Terrace_1951', '2_Bath_26', '2_Living_479', '2_Dining_308']. The string, "2_Foyer_39" means a foyer on the second floor with an area of 39sf. We assume that the DNN classified this building block as high performing due to the fact that it represents a living room that is 479sf large, is situated next to a dining room, and opens up to larger terrace.

Figure 3 shows three building blocks with high scores discovered for each of two separate functional targets, i.e., livability and sleepability. To discover these building blocks, DNNs were trained separately on each functional target but with the same set of 15 design samples; and then a forward pass through the trained DNN was used to identify building blocks of radius one or two with high regression and activation scores.

Merging subgraphs to form new compositions: Next, we merged the discovered building blocks into larger compositions. If, for example, one wants to compose a new home that performs high on both livability and sleepability, the DNN simply discovers essential building blocks based on these function targets, and merges them along edges via graph-merging algorithms.[7] Few illustrative examples are shown in figure 4. If there are nodes or edges that are typical in home designs but are missing in discovered building blocks we can add auxiliary nodes and edges to fill these gaps.

Adding additional nodes and edges through latent vector embeddings: The process of adding auxiliary edges and nodes to new compositions in a mathematically principled manner is based on a method of embedding rooms in various design samples onto a latent vector space while preserving both the similarity of room types across the design samples and the proximity of various types of rooms appearing inside each design (figure 5). This was performed by a DNN-based method for representation learning on attributed graphs. All design graphs with annotated room attributes ("type" in this case) were merged into a single large graph and the latter was served as an input to a DNN. In this way, the DNN learned a multi-dimensional vector representation of each node. Vector representations of nodes depend on their type as well as their relative proximity to other types of nodes. As figure 5 demonstrates, nodes corresponding to each type of room tend to cluster together since their types are identical. More interestingly, however, is that certain clusters of nodes, e.g. Bedrooms, tend to be closer to some clusters of nodes, e.g. Closets, Baths, Balcony and Corridors, and not to other clusters, such as Entrance, Dining etc. Thus, the latent embedded vectors tend to reflect the average proximity of various types of rooms in design samples. Also note that the clusters corresponding to Living rooms, Dining rooms and Terraces are very close to each other and overlapping at times. This is because most of the design samples had these types of rooms adjacent to each other. Vector embedding essentially exposes such latent design rules.

We used vector embedding to discover auxiliary edges or nodes that might be missing in new compositions, which can be added later on. For example, building block H2 for livability and H5 for sleepability shown in figure 4, have the Dining node in common. We merged these two subgraphs to form a larger graph along the Dining node. However, this procedure leaves the Bedroom reachable only through the terrace, which is not ideal. To fix this problem, we computed the probability of connecting various types of rooms in H2 to other types of rooms in H5. Since in the vector embedding reveals that the Bedroom cluster is close to the Corridor cluster, our composition algorithm decides to add an auxiliary edge between Bedroom and Corridor with high probability. DNN based projection of rooms on latent vector space obeys proximities of types of rooms in the given design samples.

Note that in case there are no obvious candidate pairs of rooms to connect by an auxiliary edge, we may need to add new types of rooms in the composed design. The type of rooms to be added can be determined by examining the vector embedding. For example, if the building blocks contain Bedrooms and Living rooms but not "connective" rooms such as Corridors, the latter type of room is needed to connect to the former types. This can be seen as a problem of finding a path from the Bedroom cluster to the Living room cluster. Thus, an intermediate room of type "Corridor" can be added to the composed design in an algorithmic fashion. Or, for example, as seen in the composition of H2 with H5, the resulting graph has no kitchen, in such a situation a Kitchen node can be added through vector embedding - which depicts Kitchen spaces close to Dining, Terrace and Entrance nodes.

Translating graphs into conventional modes of architectural representation: Afterwards we inspected whether the new compositions breaks any geometric constraints. The subgraphs in themselves may work well, but when put together they may form impossible assemblies. Therefore, we applied techniques to determine the fitness of the generated designs, such as planarity constraints.[8] Once a solution has been validated it can

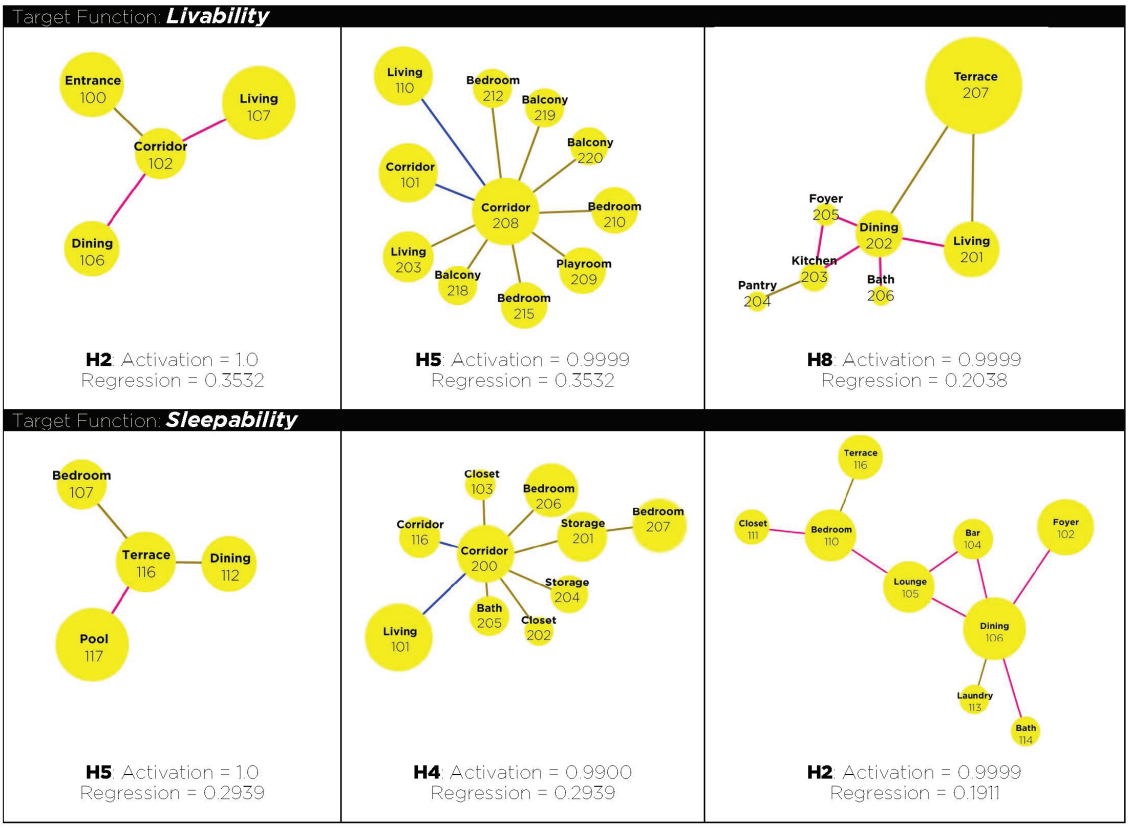

Figure 3. Discovered building blocks specific to the following target functions: livability and sleepability.

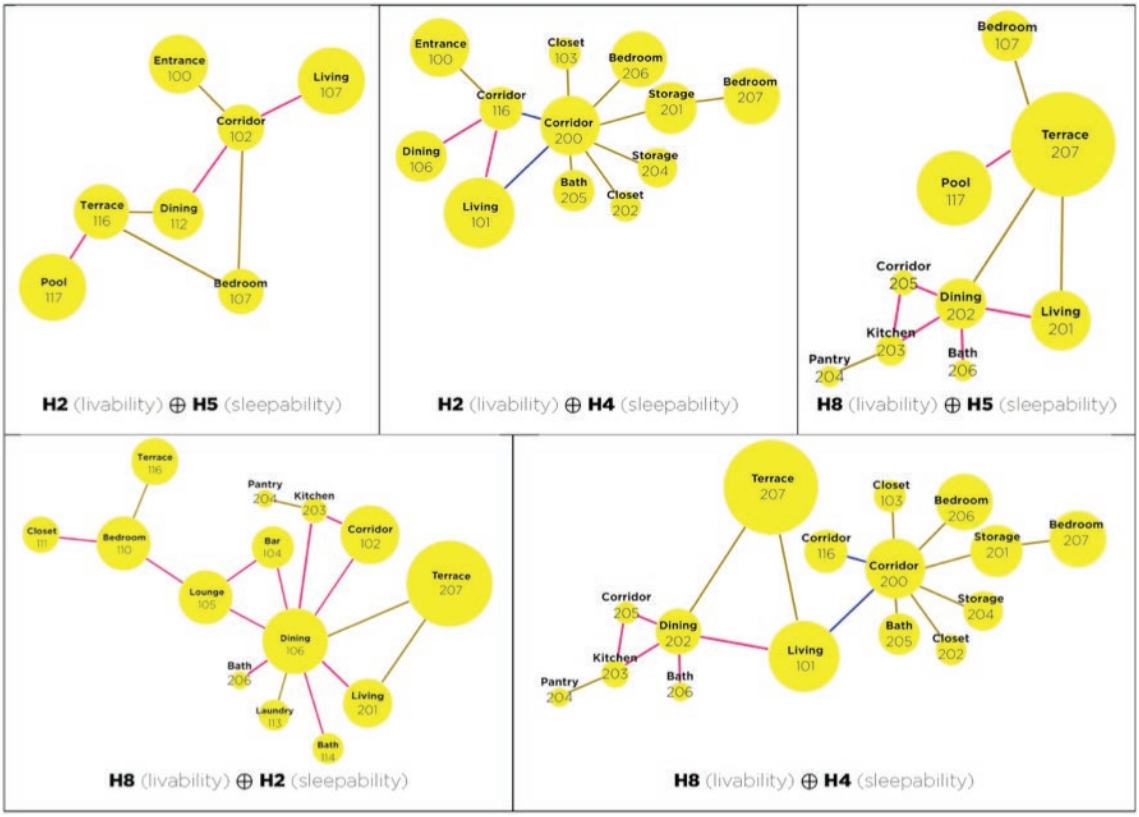

Figure 4. Composing subgraphs into larger assemblies.

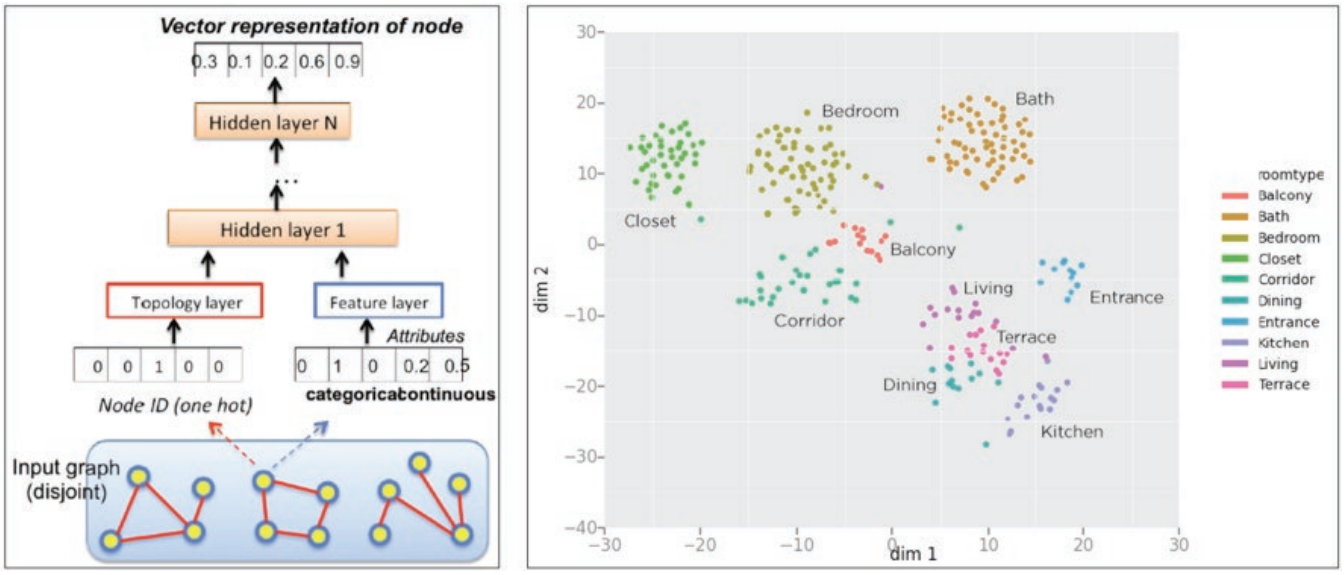

Figure 5. DNN based representation learning of types of rooms in a latent vector space while obeying proximities of types of rooms in design samples.

be converted into two-dimensional orthographic drawings or three-dimensional massing models through an algorithm that stacks nodes by obeying area and volume attributes, proximities, and connection types. The stacking can occur within a constrained building envelope, for example, as would be necessary if a new design had to fit into an existing building; or, can be more loosely stacked, if there are no such spatial restrictions.

CONCLUSION

We showed the potential of training DNNs with graphs to generate novel compositions and unique designs. DNNs have been studied well for image and audio applications, however, graph-based research is limited. We were able to explore various DNNs with a limited number of house designs, evaluated the training set, discerned essential building blocks on functional performance criteria, and merged them into new compositions in a mathematically principled manner. This study demonstrates promising early steps towards automated conceptual design. Undoubtedly, deep learning offers immense opportunities for architecture, and further research needs to be conducted to reach the full potential of neural networks in design exploration.

ENDNOTES

1 Kyle Steinfeld, "Dreams May Come," in *ACADIA 2017: Disciplines & Disruption - Proceedings of the 37th Annual Conference of the Association for Computer Aided Design in Architecture* (Fargo, ND: ACADIA, 2017).

2 C. Sjoberg et al., "Emergent Syntax: Machine Learning for the Curation of Design Solution Space," in *ACADIA 2017: Disciplines & Disruption - Proceedings of the 37th Annual Conference of the Association for Computer Aided Design in Architecture* (Fargo, ND: Acadia Publishing Company, 2017)

3 G. Stiny and J. Gips, "Shape Grammars and the Generative Specification of Painting and Sculpture," *Information Processing* 71, ed., C V Freiman (Amsterdam: North-Holland, 1972): 1460-1465.

4 M. Ruiz-Montiel et al., "Design with Shape Grammars and Reinforcement Learning," *Advanced Engineering Informatics* 27, no. 2 (April 2013): 230-245.

5 J.P. Duarte, "Towards Mass Customization of Housing: The Grammar of Siza's Houses at Malagueira," *Environment and Planning B: Planning and Design* 32 (2005): 347-380.

6 D. Duvenaud et al., "Convolutional Networks on Graphs for Learning Molecular Fingerprints," *NIPS '15: Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2* (Cambridge, MA: The MIT Press, 2015).

7 H. Ehrig and H.J. Kreowski, "Pushout-Properties: An Analysis of Gluing Constructions for Graphs," *Math. Nachr.* 91 (1979): 135–149. doi:10.1002/mana.19790910111.

8 John M. Boyer and Wendy J. Myrvold, "On the Cutting Edge: Simplified O(n) Planarity by Edge Addition," *Journal of Graph Algorithms and Applications* 8, no. 3 (2004): 241–273. doi:10.7155/jgaa.00091.