Generation of Floor Plan Variations with Convolutional Neural Networks and Case-based Reasoning

An approach for transformative adaptation of room configurations within a framework for support of early conceptual design phases

Viktor Eisenstadt[1], Christoph Langenhan[2], Klaus-Dieter Althoff[3]
[1] Institute of Computer Science, University of Hildesheim, Germany [2] Chair of Architectural Informatics, Faculty of Architecture, Technical University of Munich, Germany [3] German Research Center for Artificial Intelligence (DFKI), Germany
[1] ayzensht@uni-hildesheim.de [2] langenhan@tum.de [3] kalthoff@dfki.de

We present an approach for computer-aided generation of different variations of floor plans during the early phases of conceptual design in architecture. The early design phases are mostly characterized by the processes of inspiration gaining and search for contextual help in order to improve the building design at hand. The generation method described in this work uses the novel as well as established artificial intelligence methods, namely, generative adversarial nets and case-based reasoning, for creation of possible evolutions of the current design based on the most similar previous designs. The main goal of this approach is to provide the designer with information on how the current floor plan can evolve over time in order to influence the direction of the design process. The work described in this paper is part of the methodology FLEA (Find, Learn, Explain, Adapt) whose task is to provide a holistic structure for support of the early conceptual phases in architecture. The approach is implemented as the adaptation component of the framework MetisCBR that is based on FLEA.

Keywords: *room configuration, adaptation, case-based reasoning, convolutional neural networks, conceptual design*

INTRODUCTION

During the early phases of conceptual design in architecture, designers often look for past references in collections of printed or digitally created floor plans in order to stimulate creativity and inspiration and assess the building design at hand or find explicit solutions. The outcome of the early phases has a big impact on the future direction of the currently developed design. The most obvious way to support these phases with computer-aided means is to provide a retrieval method (e.g., in the form of a specific software solution) that is able to find similar references in a collection of previously created building designs. A multitude of approaches was developed for this task in the past, for example, using case-based retrieval (Richter 2011) or deep learning (Sharma et al. 2017). However, not only the retrieval of similar designs can help find inspiration: a generation of different design variations, based on the original and the similar designs, can show how the design would evolve over time, providing the designer with a choice of how to influence the current design direction. In this paper, we present an approach for generation of such design evolutions. The approach is based on the currently widely used machine learning techniques, such as *convolutional neural networks* (ConvNet) and *case-based reasoning* (CBR) (Kolodner 2014).

While a number of architectural design evolution approaches was developed in the past decades as well – mostly using genetic evolutional algorithms (Flack and Ross 2011; Nisztuk and Myszkowski 2019), but also, for example, mobile crowdsourcing and motion sensor data (He et al. 2017) or bayesian networks (Merrell et al. 2010) – none of them was created specifically for the early design phases. Furthermore, none of the currently existing methods is an autonomous part of a framework or methodology that implements the evolution approach as one of its collaborative components.

The implementation of our approach is part of the *FLEA* (Find, Learn, Explain, Adapt) methodology for support of early conceptual phases in architectural design. Building designs in FLEA are represented by the *room configurations* of floor plans in the form of graphs with rooms as nodes and room connections as edges (see figure 1). FLEA was derived from the original 4R (Retrieve, Reuse, Revise, Retain) cycle of CBR (Aamodt and Plaza 1994). The components in FLEA are autonomous and can be combined in order necessary for the current task. Below, a short overview of the components is provided.

- **Find** applies methods of CBR-based retrieval to find the most similar designs for the current room configuration by means of applying the *semantic fingerprints of architecture*, the graph-based patterns for representation and search of floor plans (Langenhan and Petzold 2010). The fingerprints can be selected by the user to personalize the retrieval process and make it more precise.
- **Learn** suggests the next steps for continuation of the current room configuration design process. The steps represent the most common actions, such as *add* (including the room type and position suggestion), *remove*, *reshape*, or *change type* of a room. The mode of operation of this component is based on the specific contextual recurrent neural networks that represent automatically detected floor plan contexts based on their features.
- **Explain** applies *explanation pattern-based* methods for (contextual) explanation of retrieval results achieved during the execution of the Find step. The main goal of Explain is to build trust between the user and the system by justifying the returned search results and make the system behavior during the search more transparent (Eisenstadt et al. 2018).
- **Adapt** generates variations of the current room configuration and shows how it can evolve in the future. Described in this work.

FLEA is implemented in MetisCBR (Ayzenshtadt et al. 2016), a distributed AI framework that assigns the execution of the four steps described above to the autonomous entities, i.e., agents, combining them into a multi-agent system (MAS).

FLOOR PLAN ADAPTATION APPROACH OF FLEA

The generation of different variations of a room configuration can also be seen as *adaptation of the original design* to the current *task context* (e.g., a revision of design of an apartment for an elderly married couple). Similarly, we can assume that the task context is part of the specific *problem space* (e.g., design of apartments). The component Adapt of FLEA makes use of these two assumptions, i.e., it is based on the Reuse step of the 4R CBR cycle, whose basic premise

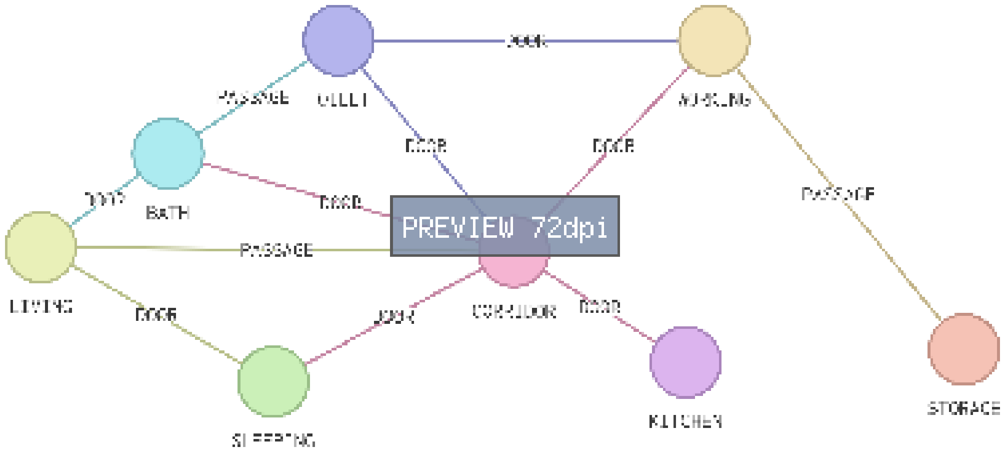


Figure 1
An example of a room configuration used in the implementation of the FLEA methodology.

is that a problem that fits to the given problem space can be solved by a solution taken and adapted from the most similar case (problem + solution) from the database of the past cases (the *case base*).

In general, two basic approaches of adaptation exist in CBR: *transformation-based* and *generative* (Wilke and Bergmann 1998). The transformation-based approach requires knowledge about adaptation process, i.e., the exact algorithm, but does not require knowledge about acceptable solutions and does not guarantee the correctness of the produced solution. The generative approach requires knowledge about acceptable solutions, but must contain the solution path within the case. FLEA applies the transformation-based approach, as knowledge about acceptable solutions is not available in the architectural design domain per se, as every designer has her/his own criteria and requirements on quality of the designs. Specifically for design cases, it is also not feasible to include the exact solution path into the case, as the number of these paths for the same design is as high as the number of possible continuations of the floor plan configuration. Therefore, the transformation-based adaptation and the corresponding generation of variations appears to be a more suitable approach for the floor plan cases.

To accomplish the task of adaptation of graph-based floor plans, FLEA's Adapt makes use of combination of both subtypes of the transformation-based adaptation: *substitution* and *structural modification*. The substitutional adaptation replaces the features (e.g., available connections of the room configuration) of the case with those of the selected solutions that are represented by the most similar room configurations from the case base. The subsequent structural modification phase is then used for addition of new or removal of existing features (e.g., rooms or their types).

Technically, our approach is based on a combination of *Generative Adversarial Nets* (GAN) (Goodfellow et al. 2014), and *case-based retrieval* (De Mantaras et al. 2005) (see section "Generator" for usage details). GAN is a methodology for neural network-based generation of new objects (e.g., images). GAN received much acceptance in the machine learning community during the last years and usually consists of two separate networks that compete against each other: the *Generator* network creates/generates objects, whereas the *Discriminator* network rates these objects and decides if they correspond to the criteria of a *"real"* object (e.g., an image that would appear real to a human). Our approach modifies the original GAN by adding a new network, the *Classificator* (see section "Classificator" for more details). The three modules and their common data representation are described in the next sections. In figure 2, the complete process of GAN- and CBR-based adaptation of room configurations is shown in detail.

Data representation

The room configurations in our adaptation approach are represented by a *connection map* of the floor plan, an approach similar to architectural morphospaces (Steadman and Mitchell 2010). Each map represents a *modified adjacency matrix* of the room configuration's graph, where instead of weights, a specific *connection code* is used to represent relations between two rooms. Each code entity is a *numerical signature of the relation*: the first two numbers stand for the room types of the connected rooms and the last number represents the connection type (e.g., Door, Stairs, or Passage). For example, the connection code **621** stands for *'Working and Living rooms connected by a Door'*. Both directions, e.g., *Living–>Working* and *Working–>Living* can be decoded, if they are connected with two different room connection types. For use in the GAN, which was originally conceptualized to work with image matrices, these numbers are then converted into the *grayscale intensity* values, e.g., **621** to **0.621**. Figure 2 shows an example visualization of a connection map.

Classificator

This new network module extends the original GAN structure with a specific pre-generation step, the *adaptation complexity classification*. During the complexity classification process, the ConvNet of the Classificator determines the complexity grade of adaptation: *0* (weak), *1* (middle), or *2* (heavy). This process is necessary in order to estimate the user's design modification direction, i.e., to match her/his expectation on the floor plan's evolution. The classification basis of this network are the complexity labels from the previously adapted connection maps. In figure 2, step 1 denotes the classification process.

Generator

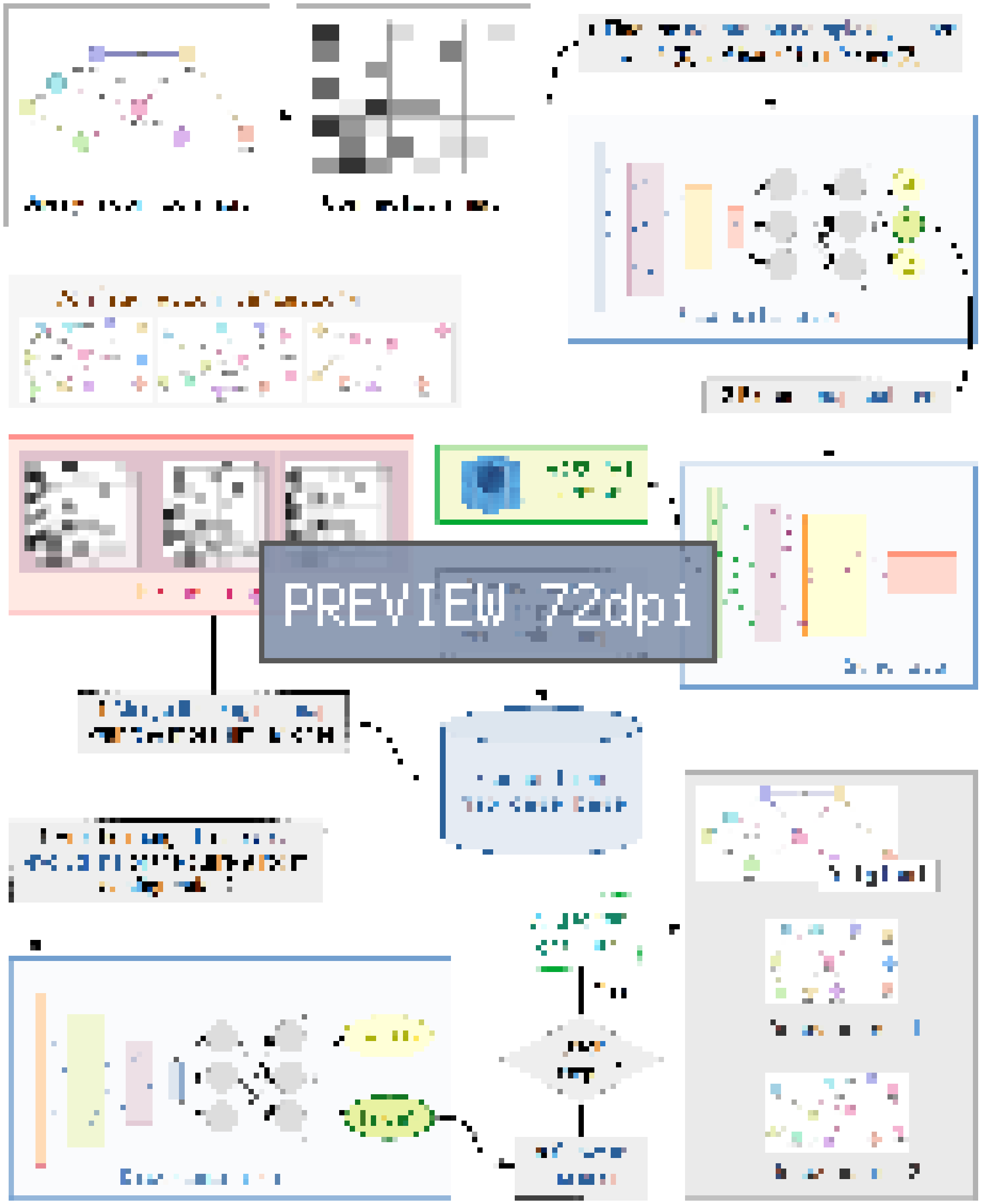
Depending on the outcome of the complexity classification, the Generator module adapts the incoming map, i.e., the query room configuration, using the following two-step approach:

1. **Feature extraction with a ConvNet** - During the first step (see also figure 2, step 2), the query map (more exactly: its grayscale image representation) is fed into the Generator's ConvNet to undergo a process of feature extraction. The outcome of this process is an *n*-dimensional tensor that represents the image features. Currently, a VGG 16-based ConvNet (Simonyan et al. 2014) is in use.
2. **Case-based merge with the case maps** - The query map is then merged with a number of the most similar previous connection maps (case maps) saved in a specific case base. Each previous map represents a *case with extracted features as attributes* used to *determine the similarity with the query map*. The merge algorithm then performs the substitutional and structural modification according to the *requirements of the complexity grade*: weak adaptation only adds connections that do not exist in the query map, heavy adaptation adds connections from the case map and replaces the 'overlapping' connections with its own, middle replaces the overlappings only partly. See also figure 2, steps 3 and 4.

Discriminator

Finally, after the adaptation merge has been performed, the Discriminator network, trained on the previously merged examples, classifies the created variations of the original map as **true** or **false**, i.e., decides if a map is a possible real continuation of the current room configuration. If the user accepts the suggested continuation, it is then added to the common case base and marked as a successor of the current room configuration in its case tree. See also figure 2, steps 5 and 6.

Figure 2
FLEA adapation approach for room configurations. The numbers in bold font denote the particular steps of the variation generation process.



EVALUATION

To evaluate the adaptation approach described in this work, the datasets of *15000* training + *3000* test examples for the Classificator and *10000* training + *2000* test examples for the Discriminator were synthesized with a specific connection map generation algorithm that can create maps with different and randomized density, i.e, the number of connections. Both, Classificator and Discriminator, used a configuration based on 2 convolutional layers and the *Nadam* optimizer (Dozat 2016). Another set of *100* connection map examples with maximum connection count of *144* was generated for querying and modifying, each of these maps was adapted with *10* most similar designs. The classification accuracy of the Classificator could reach approx. **93%**. The Discriminator could classify approx. **95%** of generated variations as **true** with approx. **94%** as classification accuracy. The evaluation has shown that our approach is generally suitable for room configuration evolutions and can be used as a module of the FLEA methodology implementation.

FUTURE WORK

Future plans for the adaptation module of FLEA include an extension of the current merge algorithm with a stepwise evolutionary adaptation of connection maps with densities of large difference and a subsequent user study.

REFERENCES

Aamodt, A and Plaza, E 1994, 'Case-based reasoning: Foundational issues, methodological variations, and system approaches.', *AI communications*, 7.1

Ayzenshtadt, V, Langenhan, C, Bukhari, S, Althoff, KD, Petzold, F and Dengel, A 2016 'Thinking With Containers: A Multi-Agent Retrieval Approach for the Case-Based Semantic Search of Architectural Designs', *ICAART 2016*, Rome, Italy, pp. 149-156

Dozat, T 2016, 'Incorporating nesterov momentum into adam.', *no title given*

Eisenstadt, V, Espinoza-Stapelfeld, C, Mikyas, A and Althoff, KD 2018 'Explainable Distributed Case-Based Support Systems: Patterns for Enhancement and Validation of Design Recommendations', *ICCBR 2018*, Stockholm, Sweden, pp. 78-94

Flack, RWJ and Ross, BJ 2011 'Evolution of architectural floor plans', *European Conference on the Applications of Evolutionary Computation*, pp. 313-322

Goodfellow, I, Pouget-Abadie, J, Mirza, M, Xu, B, Warde-Farley, D, Ozair, S, Courville, A and Bengio, Y 2014, 'Generative adversarial nets.', *Advances in neural information processing systems*

He, Y, Liang, J and Liu, Y 2017, 'Pervasive Floorplan Generation Based on Only Inertial Sensing: Feasibility, Design, and Implementation', *IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS*, 35.5, pp. 1132-1140

Kolodner, J 2014, *Case-based reasoning*, Morgan Kaufmann

Langenhan, C and Petzold, F 2010, 'The fingerprint of architecture-sketch-based design methods for researching building layouts through the semantic fingerprinting of floor plans', *International electronic scientific-educational journal: Architecture and Modern Information Technologies*, 4.13, pp. 1-8

De Mantaras, RL, McSherry, D, Bridge, D, Leake, D, Smyth, B, Craw, S, Faltings, B, Maher, ML, Cox, MT, Forbus, K, Keane, M, Aamodt, A and Watson, I 2005, 'Retrieval, reuse, revision and retention in case-based reasoning.', *The Knowledge Engineering Review*, 20.3, pp. 215-240

Merrell, P, Schkufza, E and Koltun, V 2010, 'Computer-Generated Residential Building Layouts', *ACM Transactions on Graphics (TOG)*, 29.6, p. 181

Nisztuk, M and Myszkowski, B 2019, 'Hybrid Evolutionary Algorithm applied to Automated Floor Plan Generation', *International Journal of Architectural Computing*

Richter, K 2011, *Augmenting designers memory: case based reasoning in der Architektur*, Logos Berlin

Sharma, D, Gupta, N, Chattopadhyay, C and Mehta, S 2017 'DANIEL: A Deep Architecture for Automatic Analysis and Retrieval of Building Floor Plans', *14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*

Simonyan, K and Zisserman, A 2014, 'Very deep convolutional networks for large-scale image recognition', *arXiv:1409.1556*

Steadman, P and Mitchell, LJ 2010, 'Architectural morphospace: mapping worlds of built forms', *Planning and Design*, 37.2, pp. 197-220

Wilke, W and Bergmann, R 1998 'Techniques and knowledge used for adaptation during case-based problem solving', *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*