Civil Engineering Infrastructures Journal xx,xx
DOI: 10.22059/ceij.2026.405350.2433

# A Real-World Dataset for Beam and Column Dimension Prediction in Reinforced Concrete Structures Using Graph Neural Networks

Maryam SoltanMohajer[1], Maryam Ahmadvand [2], and Razieh Ghiasi[3]

[1] Department of Civil Engineering, University of Qom, Qom, Iran. Email: MaryamSoltaniMohajer@yahoo.com, , Orcid_id: 0009-0000-2383-396x.

[2] Department of Civil Engineering, Isfahan University of Technology, Isfahan, Iran. Email: m.ahmadvand@alumni.iut.ac.ir, , Orcid_id: 0009-0006-0269-2452.

[3] Department of Computer and Information Technology, University of Qom, Qom, Iran. (Corresponding author) E-mail: raziehghiasi@gmail.com, Orcid_id: 0000-0002-5636-6189

Received: 05/12/2025
Revised: 11/08/2026
Accepted: 20/09/2026

Accepted / Not Edited

## Abstract

Conventional structural design methods are often time-consuming, repetitive, and heavily dependent on designer expertise, requiring significant manual effort. Deep learning models offer a means to accelerate this process and reduce reliance on individual skills; however, their development is limited by the scarcity of large, well-structured datasets. To address this challenge, this paper introduces a standardized real-world dataset, AV_SStruct313, collected from 313 real reinforced concrete projects in Iran. Unlike previous studies that rely on synthetic datasets or limited real-world samples, AV_SStruct313 is derived from fully executed, code-compliant structural projects and explicitly captures practical design and construction constraints. The proposed dataset provides a realistic and reliable benchmark for developing accurate predictive models of beam and column dimensions. To demonstrate its effectiveness, five state-of-the-art

graph neural networks—GCN, GIN, GraphSAGE, GAT, and Graph Transformer—are evaluated on AV_SStruct313. Among, the Graph Transformer achieved the highest accuracy, with an average prediction error of approximately 5 cm, which is consistent with practical design discretization levels in reinforced concrete design, and inference time under one second. These results indicate that graph-based learning can significantly reduce manual design effort and accelerate reinforced concrete design workflows, demonstrating a clear improvement over conventional methods and existing learning-based approaches.

**Keywords:** AI-assisted Civil Engineering, Construction Intelligence Systems, Graph-based Deep Learning Models, RC Element Size Prediction, Dataset Collection.

## 1. Introduction

In the late 19th century, with industrial growth, the increase in building heights, and the introduction of modern materials such as steel and concrete, together with advances in engineering sciences and the development of simple numerical methods, the first codes and regulations were established to ensure structural safety in developed countries(K. K. Gupta and Meek, 1996; Ellingwood et al., 2025; Pandey et al., 2025; Mathewson, 2023). Scientific calculations gradually became an integral part of the construction industry. Structural design based on these calculations typically proceeds in three stages: preliminary design, detailed design, and drawing preparation(Al-Bizri and MacLeod, 2024; R. S. Gupta and Gupta, 2014). During preliminary design, decisions are made primarily based on engineering experience, taking into account factors such as structural importance, architectural constraints, economic and time limitations, material selection, analysis methods, lateral load-resisting system, floor type, column layout, and framing system.

During the design stage, the dimensions of beam and column sections, as well as other elements of the lateral load-resisting system, are determined. The required reinforcement area for each section is calculated to ensure the necessary stiffness and strength under lateral and gravity loads, as required by design codes. Three main objectives guide this process: (1) compliance with codes, (2) adherence to architectural and construction constraints, and (3) minimization of material usage. Designers rely on experience, trial-and-error, and engineering judgment to reach an optimal solution, making the process iterative and time-consuming. The quality of the final design varies depending on the designer's expertise.

Finally, in the drawing preparation stage, the calculated section dimensions and required reinforcement areas are drafted and submitted to the client for construction. The time-consuming nature of this process, particularly the design stage, has long focused the structural engineering community's attention on automating these procedures.

Research in this field has evolved through three major milestones. First, structural design software such as SAP, ETABS, and SAFE automated analysis, although section selection still relied on trial-and-error(Liu et al., 2022). Second, classical AI methods, including knowledge-based approaches and metaheuristic algorithms, aimed to automate decision-making and optimize section selection(Sriram et al., 1985; Mendel, 1995; Adeli and Kumar, 1995; Perez and Behdinan, 2007; Nimtawat and Nanakorn, 2011; Shahrouzi and Pashaei, 2013; Zhang and Mueller, 2017; Kanyilmaz et al., 2022). However, challenges such as the inability to accurately represent engineering considerations, the vast search space in

Accepted / Not Edited

complex 3D models, the diversity of structural systems, and the need to define numerous code-compliant constraints have limited their real-world applications. Third, the rise of machine learning and deep learning in practical applications has shifted research toward these methods, particularly for material strength prediction, structural analysis, design optimization, and damage assessment(Singaravel et al., 2019; Sun et al., 2021; Pizarro and Massone, 2021; Pizarro et al., 2021; Zhao et al., 2022; Fei et al., 2022; Kumar et al., 2025). These approaches leverage models' ability to extract hidden patterns from data, enabling accurate predictions and supporting design decisions.

Despite the remarkable achievements of deep learning, developing practical models in structural engineering still remains challenging. Although several recent studies have successfully applied machine learning and graph-based deep learning to structural layout generation and member size prediction, many of these approaches rely on synthetic datasets, limited numbers of real-world projects, or simplified structural representations. Consequently, their ability to capture practical engineering constraints and generalize to realistic structural design scenarios remains limited. A major obstacle is the lack of comprehensive, reliable, and standardized datasets capturing real-world constraints. This issue is particularly acute for predicting beam and column sections in reinforced concrete structures. The absence of such datasets limits the generalizability of models and hinders accurate performance evaluation.(Liao et al., 2024)

To address this gap, this study introduces AV_SStruct313, a standardized dataset compiled from 313 executed reinforced concrete building projects in Iran. Unlike previous studies, the proposed dataset is derived entirely from real-world, code-compliant engineering projects and incorporates practical design knowledge through a comprehensive feature representation. The proposed feature set consists of four categories: general features, floor-level features, independent element features, and dependent element features. General features describe the overall characteristics of the structure; floor-level features pertain to all elements on a given floor; independent element features are defined for each element individually; dependent element features are meaningful only in relation to other elements. and dependent element features explicitly capture engineering relationships among structural elements, such as alignment and structural dependency, enabling the model to learn practical design regularities and constructability-related patterns. Using these features, a dataset comprising 313 structures with 39 input features was compiled for predicting beam and column sections.

Furthermore, Graph Neural Networks (GNNs) are employed to predict beam and column section dimensions. Due to the inherently graph-like nature of structural frames, GNNs have recently gained significant attention in structural engineering(Chou et al., 2024, 2025). In this approach, structures are represented as graphs, with elements as nodes and connections as edges (Figure 1). This enables the simultaneous utilization of topological information, load paths, and spatial relationships, thereby improving prediction accuracy and computational efficiency. In contrast to previous studies, which have primarily evaluated graph-based models on synthetic or relatively small datasets, this study presents a systematic comparison of five state-of-the-art GNN architectures—GCN, GIN, GraphSAGE, GAT, and Graph Transformer— using a large real-world dataset compiled from completed engineering projects. This evaluation provides a comprehensive assessment of their

Accepted / Not Edited

effectiveness and practical applicability for beam and column dimension prediction. The investigated architectures demonstrate strong performance in modeling structural relationships and accurately predicting beam and column dimensions, highlighting the potential of graph-based learning for real-world structural engineering applications. The main contributions of this study are as follows:

- Identification, categorization, and structured organization of features for predicting beam and column sections in reinforced concrete structures, including practical dependency features that represent engineering design and constructability considerations.
- Compilation of AV_SStruct313, a standardized real-world dataset comprising 313 code-compliant fully aligned with the designed feature hierarchy (general, floor-level, independent element, dependent element).
- Investigation and systematic evaluation of five of state-of-the-art graph neural network architectures to model the defined feature structure and inter-element relationships on the proposed dataset.

The remainder of this paper is organized as follows. Section 2 reviews related prior studies. Section 3 introduces the proposed dataset and explains the rationale for feature selection. Section 4 presents the GNN-based models developed for section prediction. Section 5 reports the experimental results and evaluates model performance. Finally, Section 6 summarizes the findings, highlights key insights, and outlines directions for future research.

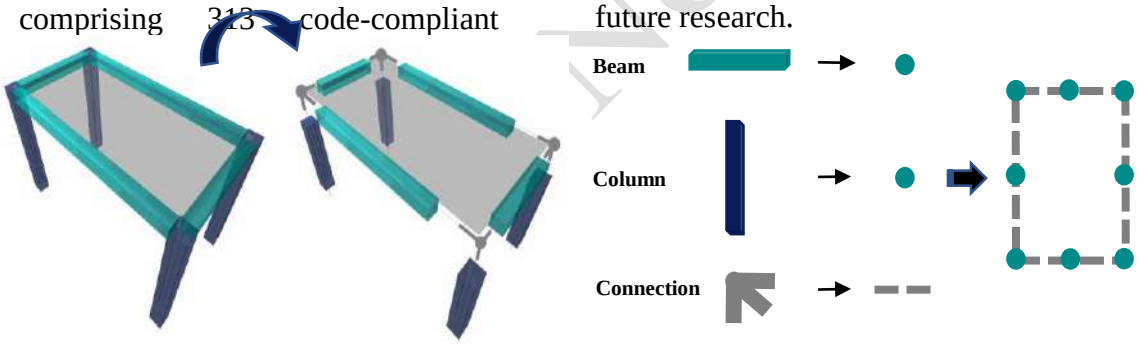



*Figure 1: Graph representation of structural frames, with structural elements defined as nodes and their connections as edges, enabling the integration of topology, load paths, and spatial relationships.*

## 2. Related Work

In recent years, the application of machine learning techniques in structural engineering has attracted growing attention, aiming to improve prediction accuracy and streamline the design process. A wide range of studies has explored various aspects of structural modeling and component sizing, highlighting both the opportunities and the current limitations of these approaches. In general, existing research can be categorized into three main directions: structural layout generation, individual component sizing, and graph-based frame-level prediction models.

Early investigations primarily examined the spatial arrangement of structural elements, such as shear walls and beams. For instance, (Ampanavos et al., 2022) proposed a CNN-based framework to automatically generate preliminary structural layouts from architectural floor plans during the conceptual design stage. Their approach assists architects by recommending feasible structural configurations before detailed

Accepted Manuscript

structural analysis. Similarly, (Soledispa et al., 2024) developed an ANN-based framework to generate preliminary reinforced concrete wall segments and columns from architectural plans using structural neighbor-based features. Their method supports early-stage structural design by proposing structural elements that are not included in the initial architectural layout. In addition, Chang et al. (Zhao, Fei, et al., 2023) and related works (Xia et al., 2025; Zhao et al., 2024; Zhao, Liao, et al., 2023) analyzed the relationships and interactions among structural components to identify optimal layouts. These studies demonstrated the potential of intelligent models to reduce design time and enhance structural performance, although their direct application to real-world construction projects remains limited. Also, these approaches primarily focus on structural topology and layout generation rather than predicting engineering design variables such as beam and column cross-sectional dimensions.

Subsequent research shifted toward predicting the cross-sections of individual beams. Gallet et al. (Gallet et al., 2024) employed deep learning to estimate beam dimensions, focusing on single elements without fully accounting for their interactions within the overall structural frame. While these methods achieved promising results in beam sizing, their scope remained limited to individual components rather than entire frame systems. In contrast, the objective of the present study is to simultaneously predict both beam and column dimensions within a unified structural framework, while explicitly considering inter-element interactions in three-dimensional reinforced concrete frames.

More recent work has adopted graph-based neural network models to predict cross-sections of both beams and columns in reinforced concrete (RC) frames. Chang and Cheng (Chang and Cheng, 2020) introduced an end-to-end pipeline that integrates a differentiable structural simulator (NeuralSim) with a size-prediction network (NeuralSizer). Their study modeled two-dimensional frames and utilized a synthetic dataset of 4,000 building skeletons with randomly sampled cross-sections. This approach enabled gradient-based training and produced designs comparable to those generated by traditional optimization algorithms, validating the effectiveness of graph-based representations in structural design. However, a key limitation of this work is its reliance on synthetic datasets and simplified 2D structural representations, which limits its applicability to real-world engineering scenarios that require compliance with design codes and constructability considerations.

Similarly, Qin et al. (Qin et al., 2025) developed a dataset of 95 real RC frame structures, encompassing 298 standard stories from engineering projects. Using heterogeneous graph neural networks (HetGNNs), they captured diverse frame topologies and accurately predicted beam and column dimensions. Their findings demonstrated that HetGNNs can substantially improve both efficiency and precision in size prediction while maintaining compliance with design standards. Nevertheless, their study is limited by the relatively small dataset size and restricted structural diversity, which may constrain the generalizability of the proposed approach to a wider range of practical engineering scenarios. In addition, the they do not explicitly incorporate constructability-oriented design rules, such as alignment (co-linearity) constraints, which play an important role in practical structural design decision-making.

Accepted / Not Edited

Overall, existing studies have demonstrated the potential of graph-based learning for structural dimension prediction. However, important challenges remain, including the reliance on synthetic datasets, limited dataset scale, restricted structural diversity, insufficient consideration of practical design constraints, and the limited availability of standardized real-world benchmark datasets for evaluating graph-based models. Furthermore, most existing studies focus on layout-level predictions, rather than the simultaneous prediction of beam and column cross-sections in complete three-dimensional reinforced concrete frame systems. These limitations motivate the present study, which aims to provide a large-scale real-world benchmark dataset together with a systematic evaluation of graph neural network models for structural dimension prediction.

## 3. Dataset Description

As noted earlier, a major challenge limiting the effectiveness of predictive models in structural engineering is the lack of suitable datasets, particularly for predicting beam and column sections in reinforced concrete structures. To address this gap, the present study develops a comprehensive and reliable dataset specifically designed for this purpose.

First, a set of features expected to have a significant impact on the prediction of beam and column dimensions is identified. Based on these features, the dataset **AV_SStruct313** is constructed and described in detail. Finally, to provide a clearer understanding of the data distribution, histograms of several key features are presented

### *3.1 Feature Identification*

In machine learning, the quality and relevance of features are crucial to achieving strong model performance. Essentially, these models aim to learn a mapping from the input space (X) to the output space (Y) by adjusting parameters such as weights and biases. In this way, learning models can be viewed as generalized mathematical functions (linear or nonlinear) that uncover hidden relationships and patterns within data. Therefore, carefully designed and domain-specific features play a vital role in enabling models to capture causal relationships accurately and generalize effectively to unseen cases.

In the structural design process, engineers must make a series of key decisions regarding primary design variables. These decisions encompass the choice of structural system (concrete or steel), applicable design codes, structural analysis methods, lateral load-resisting system, floor system type, material properties (e.g., concrete compressive strength, steel yield strength), and the column layout and framing arrangement in accordance with architectural constraints. Furthermore, determining the dimensions of structural sections and the required reinforcement for reinforced concrete members is a fundamental aspect of the design. The objective of these decisions is to achieve an optimal combination of design variables that complies with code requirements, satisfies architectural and construction constraints, and minimizes material usage.

In this study, the focus is on predicting one of the most critical design variables: the dimensions of structural members (width and height) using machine learning models. Accordingly, the other influential design variables discussed above are treated as input features for the model. It should be noted that the determination of required reinforcement, as part of the detailed design process, is performed by the engineer after the section dimensions have been predicted, in accordance with relevant code provisions.

To develop a model capable of accurately predicting the dimensions of structural

Accepted / Not Edited

members while satisfying the three key design objectives—minimizing material usage, complying with code requirements, and adhering to construction and architectural constraints—two fundamental strategies were adopted in data preparation. First, to support material minimization, training samples were selected from optimized designs that meet code requirements with minimal material consumption. Second, to ensure compliance with codes and practical considerations, input features were defined to capture the key parameters influencing structural behavior, enabling the model to effectively learn the governing relationships. Seismic design codes are established to achieve an optimal balance among three fundamental aspects of structural behavior: stiffness, strength, and ductility, ensuring that a structure's vulnerability to earthquakes is controlled according to its importance and usage. Essentially, code provisions provide a systematic framework to regulate and balance these parameters, maintaining reliable seismic performance under extreme loading conditions.

In addition, certain practical and constructability considerations are incorporated into section design, including:

- **Coordination of aligned beams:** Beams with an angular difference of less than 40° should have identical width and height.
- **Coordination of column dimensions along the height:** Upper-story columns should be equal to or smaller than lower-story columns.
- **Geometric compatibility between connected beams and columns:** This ensures optimal force transfer, proper joint behavior, and facilitates detailing and formwork

Accordingly, the features used in the model, as presented in Table 1, are categorized into three main groups:

a. Features reflecting applied loads and the type of analysis.

b. Features characterizing structural stiffness, strength, and ductility.

c. Features related to practical considerations and constructability constraints

*Table 1: Description of the input features*

| Category | Feature | Range | No | Description |
|---|---|---|---|---|
| a | Analysis Type | Static<br>Spectral | 2 | Specification of the method for evaluating applied loads and the structure's nonlinear response. |
| | Element Type | Beam<br>Column | 1 | These features determine which components of the applied forces have the greatest influence on each element. |
| | Angle of Element to Horizontal | $0 < \theta < 180$ | 1 | |
| | Seismic Coefficient | C | 1 | Determination of the intensity of lateral loads applied to the structure and their direct effect on the dimensions and design of the elements |
| | Building Height | $1 < H < 10$ | 1 | The primary indicator for determining the structure's vibration period and the intensity of seismic effects on the entire structure. |
| | Number of Floors | num_story | 1 | Consideration of the effects of upper stories |

Accepted / Not Edited

| | | | | |
|---|---|---|---|---|
| | Floor Number | Si | 1 | |
| | Element Gravity Load Share | G | 1 | Dead and live loads applied to each element |
| | Story Shear(X,Y) | Ex,Ey | 2 | Shear developed in each story based on the distribution of seismic forces across the floors |
| | Opening Conditions | Only staircase and small skylight<br>Large symmetric skylight<br>Large asymmetric skylight | 1 | These parameters are defined to describe the distribution of mass and stiffness in the plan and along the height, which affect the torsional coefficient. |
| | Inter-story Height Variation | Does not exist<br>Only at the first floor<br>Exists | 1 | |
| | Inter-floor Area Difference | Does not exist<br>Exists | 1 | |
| | Structural Symmetry(X,Y) | Symmetric<br>Asymmetric | 2 | |
| | Arrangement of Beams | Perpendicular<br>Slightly inclined<br>Inclined<br>Highly inclined | 1 | |
| | Aspect Ratio of Plan | B/L | 1 | The offset between the center of mass and the center of rigidity induces torsion in the structure. This torsion leads to additional shear in the story and also alters the distribution of forces. |
| | Center of Mass(X,Y) | CM(x,y) | 2 | |
| | Center of Stiffness (X,Y) | CR(x,y) | 2 | |
| | CM–CS Offset (X,Y) | ΔL(Cm-CR) | 2 | |
| b | Lateral Load-Resisting System | IMRF<br>SMRF | 1 | Determination of the structure's ductility and the range of permissible deformations |
| | Element Distance from Story CS (X,Y) | Assuming the cross-section dimensions of all elements are constant (width = 30, height = 30), the center of rigidity is calculated, and the horizontal and vertical distances of the midpoint coordinates of each element are reported proportionally. | 2 | By defining this feature, the arrangement of beams and their relative layout is provided to the model. This feature indicates the distribution of stiffness among the elements. |
| | Number of Frames (X,Y) | num_frame(x,y) | 2 | Parameters affecting the structural stiffness |
| | X-to-Y Frame Ratio | Lx/Ly | 1 | |
| | Concrete Compressive Strength | Fc(beam,column) | 2 | |

Accepted / Not Edited

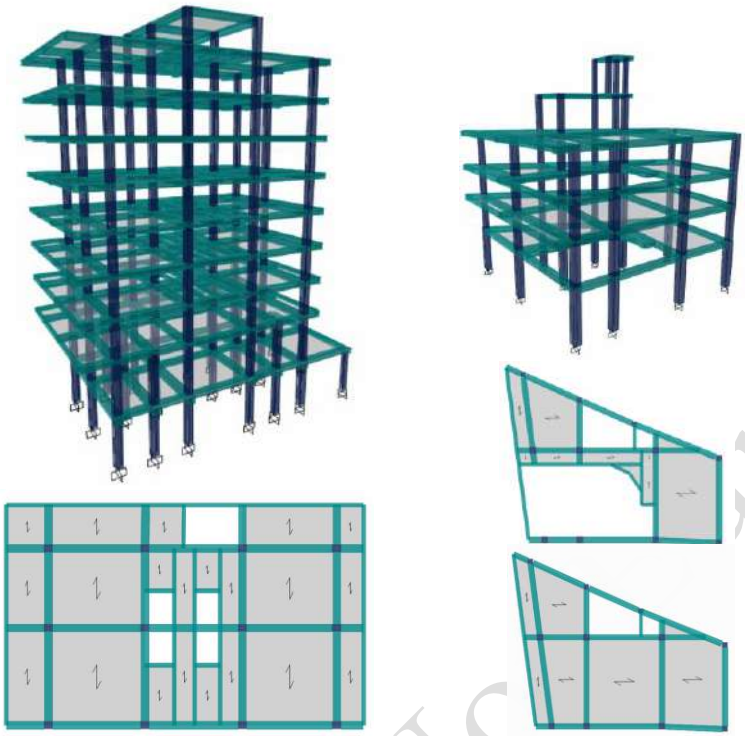
| | | | | |
|---|---|---|---|---|
| | Length of Each Element | Δ(x.y) | 1 | |
| | Element Connection Type | Beam-to-Beam<br>Beam-to-Column | 1 | |
| | Allowable Drift | 0.02/0.025 | 1 | Specification of displacement limits. |
| c | Start and End Coordinates of Each Element | X1,Y1,Z1,X2,Y2,Z2 | 6 | Ensuring proportionality of the dimensions of beams and columns connected at a single joint |
| | Alignment | Elements with an angular difference of less than 40 degrees are considered aligned and are assigned the same number. | 1 | Columns of upper stories should be smaller than or equal to the columns of the lower stories. Preferably, beams aligned on the same plane and in the same direction are considered to have equal width and height |

### *3.2 Dataset Overview*

The proposed dataset includes information from 313 projects designed by experienced engineers, fully complying with the latest editions of Iran's National Building Code—Standard 2800 (Building and Housing Research Center (BHRC), 2014) Chapter 6 (Ministry of Roads and Urban Development, 2019), and Chapter 9 (Ministry of Roads and Urban Development, 2020), and approved by the Iran Engineering Organization. It encompasses a wide range of reinforced concrete residential structures, comprising a total of 1,658 stories and 74,635 structural elements. The selected structures were designed with crack coefficients of 0.35 and 0.7, without shear walls, using one-way joist-block floor systems, and a design ground acceleration of 0.3g.

Each dataset sample represents a single 3D reinforced concrete structure, consisting of a set of beams and columns. The number of elements varies across samples, reflecting the diversity in structural size and complexity of real-world buildings (Figure 2). Based on the design constraints described in the previous paragraph, 42 features were defined for each element as model inputs, while two target variables—member width and height—were considered as outputs.

Accepted / Not Edited



**Sample 1** **Sample 2**

*Figure 2: Two 3D reinforced concrete structures with their floor plans, illustrating the diversity of structural configurations in the dataset*

As shown in Table 2, the input features of each dataset sample are hierarchically organized into four levels:

- **General features:** These features define the overall type of the structure and are identical for all elements within a single structural sample. Their role is to distinguish between different structural samples, e.g., frame type.
- **Floor-level features:** These features are identical for all elements within a specific floor of a structure, e.g., floor shear.
- **Element-level features:** These features describe individual elements and are divided into two subcategories:
    - **Independent element features:** Defined separately for each element, these features can independently characterize a property of the element, e.g., beam length

Accepted / Not edited

*Table 2: Hierarchical organization of input features across four levels and the corresponding labels for each dataset sample*

| | | **Featuer** | | | | | | | | | | | | **Label** | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Story | Element | General Feature | | | | Floor Level Feature | | | Independent element | | | Depended element | | Height | Width |
| | | F(1) | F(2) | … | F(18) | F(19) | … | F(28) | F(29) | … | F(33) | F(34) | … | F(42) | |
| S1 | E1,(S1) | X(1) | X(2) | … | X(18) | Xs1 -(19) | … | Xs1 -(28) | XE1,(S1) -(29) | … | XE1 ,(S1) -(33) | DepXE1,(S1) -(34) | … | DepXE1 ,(S1)-(42) | HE1,(S1) | WE1,(S1) |
| | E2,(S1) | X(1) | X(2) | … | X(18) | Xs1 -(19) | … | Xs1 -(28) | XE2,(S1) -(29) | … | XE2,(S1) -(33) | DepXE2,(S1) -(34) | … | DepXE2 ,(S1)-(42) | HE2,(S1) | WE2,(S1) |
| | …. | X(1) | X(2) | … | X(18) | … | … | … | … | … | … | … | … | … | … | … |
| | Em-1 ,(S1) | … | … | … | … | Xs1 -(19) | … | Xs1 -(28) | XEm-1,(S1) -(29) | … | XEm-1,(S1) -(33) | DepXEm-1,(S1) -(34) | … | DepXEm-1,(S1)-(42) | Hm-1,(S1) | Wm-1,(S1) |
| | Em ,(S1) | … | … | … | … | Xs1 -(19) | … | Xs1 -(28) | XEm,(S1) -(29) | … | XEm,(S1) -(33) | DepXEm,(S1) -(34) | … | DepXEm,(S1)-(42) | Hm,(S1) | Wm,(S1) |
| … | E1,(...) | … | … | … | … | X...-(19) | … | X... -(28) | XE1,(…) -(29) | … | XE1 ,(…) -(33) | DepXE1 ,(…)-(34) | … | DepXE1,(…) -(42) | HE1,(…) | WE1,(…) |
| | E2,(...) | … | … | … | … | X... -(19) | … | X...-(28) | XE2 ,(…)-(29) | … | XE2 ,(…) -(33) | DepXE2,(…) -(34) | … | DepXE2,(…) -(42) | HE2,(…) | WE2,(…) |
| | …. | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … |
| | Em-1,(...) | … | … | … | … | X... -(19) | … | X... -(28) | XEm-1,(…)-(29) | … | XEm-1,(…) -(33) | DepXEm-1,(…)-(34) | … | DepXEm-1,(…)-(42) | Hm-1,(...) | Wm-1,(...) |
| | Em,(...) | … | … | … | … | X... -(19) | … | X... -(28) | XEm,(…)-(29) | … | XEm,(…) -(33) | DepXEm,(…)-(34) | … | DepXEm,(…)-(42) | Hm,(...) | Wm,(...) |
| Sn | E1,(Sn) | … | … | … | … | Xsn -(19) | … | Xsn-(28) | XE1 ,(Sn)-(29) | … | XE1 ,(Sn) -(33) | DepXE1 ,(Sn) -(34) | … | DepXE1,(Sn) -(42) | HE1 ,(Sn) | WE1 ,(Sn) |
| | E2,(Sn) | … | … | … | … | Xsn -(19) | … | Xsn-(28) | XE2,(Sn) -(29) | … | XE2 ,(Sn) -(33) | DepXE2,(Sn) -(34) | … | DepXE2,(Sn) -(42) | HE2 ,(Sn) | WE2 ,(Sn) |
| | …. | X(1) | X(2) | … | X(18) | … | … | … | … | … | … | … | … | … | … | … |
| | Em-1,(Sn) | X(1) | X(2) | … | X(18) | Xsn -(19) | … | Xsn-(28) | XEm-1,(Sn)-(29) | … | XEm-1 ,(Sn)-(33) | DepXEm-1,(Sn) -(34) | … | DepXEm-1,(Sn)-(42) | Hm-1,(Sn) | Wm-1,(Sn) |
| | Em,(Sn) | X(1) | X(2) | … | X(18) | Xsn -(19) | … | Xsn-(28) | XEm,(Sn)-(29) | … | XEm ,(Sn)-(33) | DepXEm,(Sn) -(34) | … | DepXEm,(Sn)-(42) | Hm,(Sn) | Wm,(Sn) |

Accepted / Not Edited

- **Dependent element features:** Also defined separately for each element, these features do not have a standalone meaning; rather, they describe the element's behavior in interaction with other structural components, e.g., the element's distance from the structural coordinate center.

This hierarchical categorization facilitates accurate modeling of inter-level relationships and enables the machine learning model to simultaneously capture global structural behavior and local effects. The extracted features, organized according to this classification, are presented in Table 3.

*Table 3: Extracted features of dataset samples organized according to hierarchical levels*.

| Hierarchical levels | Features |
|---|---|
| **General features** | 1. Lateral Load-Resisting System 2. Opening Conditions 3. Inter-story Height Variation 4. Inter-floor Area Difference 5.Structural Symmetry(X) 6.Structural Symmetry(Y) 7.Arrangement of Beams 8. Aspect Ratio of Plan 9. Number of Frames (X) 10.Number of Frames (Y) 11. X-to-Y Frame Ratio 12.Building Height 13. Number of Floors 14. Seismic Coefficient 15.Allowable Drift 16.Fc(column) 17.Fc(beam) 18.Analysis Type (D) 18.Analysis Type (S) |
| **Floor-level features** | 19. Floor Number 20.Ex 21.Ey 22.CMx 23. CMy 24.CRx 25.CRy 27.CM–CS Offset (X) 28.CM–CS Offset (Y) |
| **Independent element features** | 29. Element Type 30.Element Length 31. Angle of Element to Horizontal 32. Element Gravity Load Share 33. Element Connection Type |
| **Dependent element features** | 34.Alignment 35.X1 36.Y1 37.Z1 38.X2 39Y2 40.Z2 41. Element Distance from Story CS (X) 42. Element Distance from Story CS (Y) |

In this study, a systematic approach was developed to extract structural information using a Python program interfacing with the ETABS API(Computers and Structures Inc. (CSI), 2016). The outputs are stored in three Excel files, containing feature data, label information, and element connectivity data.

Finally, to assess the quality, distribution, and balance of the proposed dataset, histograms were plotted for the key features (**Error! Reference source not found.**). This assessment is important, as data quality and balance directly influence the performance and generalizability of machine learning models. The analysis indicates that most buildings have 4–8 stories, with story heights ranging from 15 to 20 meters. Width-to-length ratios are primarily between 0.5 and 0.7, and design seismic coefficients generally fall within 0.05–0.165. These results indicate that the dataset provides sufficient variability and relative balance, with key feature patterns clearly represented.

Accepted / Not Edited

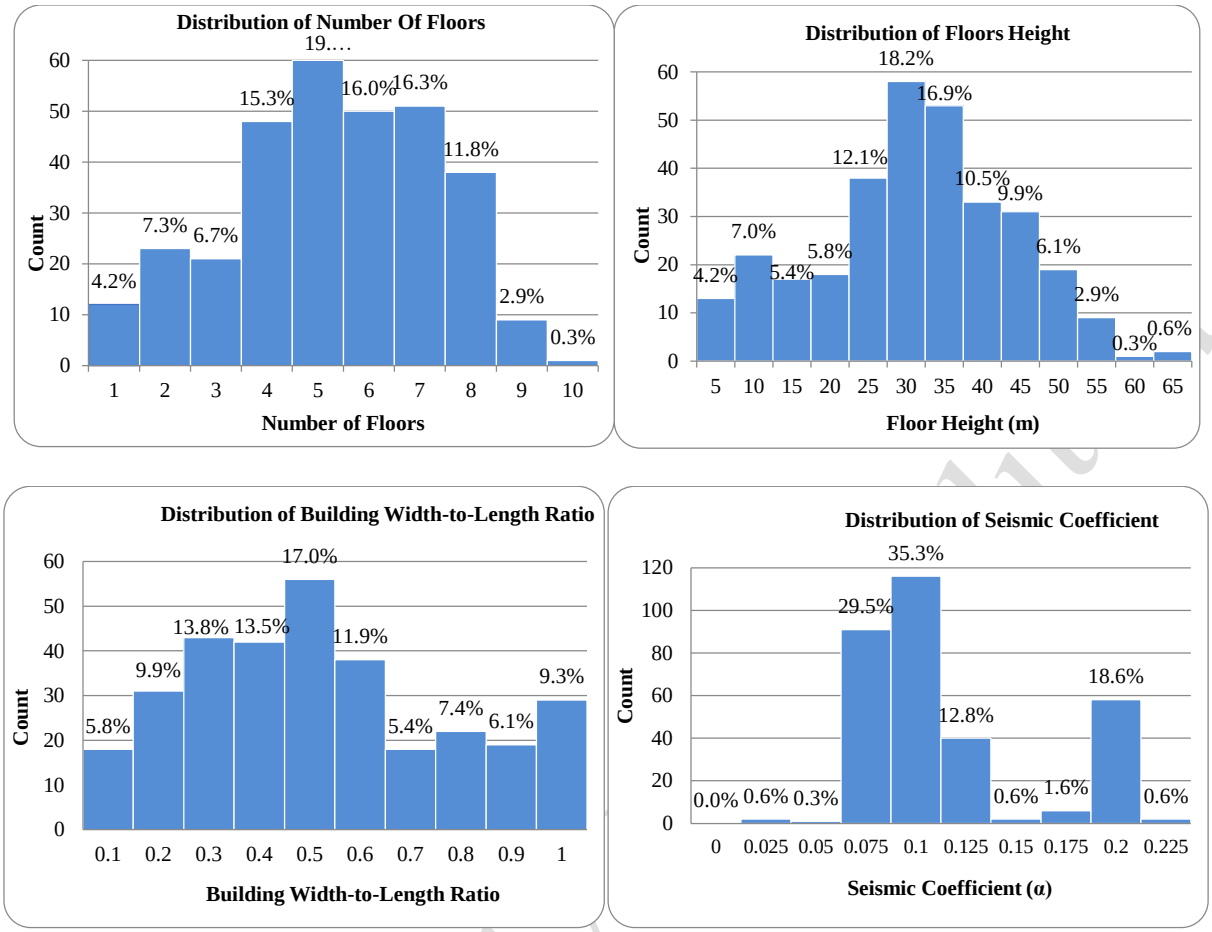



*Figure 3: Histograms of four key features in the proposed dataset*

## 4. Model Introduction

Graph Neural Networks (GNNs) are a new generation of deep learning models capable of directly processing graph-structured data. Unlike classical neural networks, which are designed for regular data structures, GNNs can model complex and irregular relationships among entities through nodes and edges. In these networks, each node aggregates information from its neighbors using a message-passing mechanism and combines it with its own features, enabling the learning of representations that capture both local node attributes and the overall graph structure. Formally, a layer of a GNN can be expressed as:

$$a_v^l = Aggregate^l(\{h_u^{l-1}: u \in \mathcal{N}(v)\}), \quad (1)$$

$$h_v^l = Updater^l\,(h_v^{l-1}, a_v^k),$$

Accepted

where $h_v^l$ is the feature vector of the node $v$ at layer $l$, initially set to the node's input features $h_v^0 = X$ . Also, $\mathcal{N}(v)$ denotes the set of neighbors of node $v$, and $u$ is a member of this neighborhood. Any permutation-invariant function, such as sum, mean, or max, can be used as the aggregation and update function. The choice of aggregation and update functions plays a pivotal role in the performance of GNNs, and extensive efforts have been made to develop effective feature aggregators that improve node representation. This has led to various GNN architectures, including Graph Convolutional Networks (GCNs), Graph Attention Networks (GATs), and, more recently, with the success of transformer models, Graph Transformer Networks (GTNs), which will be described in the following sections.

### *4.1 Graph Convolutional Networks (GCNs)*

Graph convolutional networks (GCNs) are one of the most widely used families of deep learning models for graph-structured data, inspired by convolutional neural networks in computer vision. The core idea is that each node can compute its representation by aggregating information from its neighbors and updating its features accordingly. In this method, convolution is defined over the graph structure, and the learning process is performed layer by layer by combining node features with their neighbors. Prominent models in this category include GCN, Graph SAGE, and GIN.

The GCN model, proposed by Kipf et al.(Kipf and Welling, 2017), seeks to generalize convolution to graph-structured data by using the Fourier transform. In this method, the new representation of each node is obtained using the following formula:

$$h^{l+1} = \sigma\left(\left(\widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}}\right)h^l w\right), \quad (2)$$

where $h^l$ is the node representation at layer $l$, $\tilde{A} = A + I$ is the adjacency matrix with self-loops, $\widetilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ is the corresponding degree matrix $\tilde{A}$, $w$ is a learnable weight matrix, and $\sigma$ is a nonlinear activation function such as ReLU. Also, normalized averaging $\widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}}$ is chosen as the aggregator function, while the linear transformation (learnable weights) followed by the activation function is selected as the update function.

The formula updates each node's representation via a linear transformation of its own features and a weighted aggregation of its neighbors' features, as defined by $\bar{A}$. Although this model has limitations in learning large graphs due to the need to load the entire graph into memory and learn long-range dependencies, it is used as a basic model in many applications due to its simplicity and efficiency.

The GraphSAGE method, proposed by Hamilton et al.(Hamilton et al., 2017), addresses the limitations of the scalability of GCNs by locally sampling each node's neighborhood and applying appropriate aggregation functions:

$$a_v^l = Aggregate^l(\{h_u^{l-1} : u \in \mathcal{N}(v)\}),$$
$$h_v^{(l+1)} = \sigma(w^l \cdot Concat(h_v^l, a_v^l))), \quad (3)$$

while the authors employ three types of aggregation functions: mean, pooling (using either maximum or average), and a long short-term memory (LSTM) network. The update step is performed by concatenating the node's feature vector with the aggregated features of its neighbors, followed by a linear transformation and an activation function. A key advantage of this model lies in its inductive capability, enabling it to generate predictions for previously unseen graphs without requiring retraining. This property makes GraphSAGE a powerful and efficient

Accepted / Not Edited

method for analyzing large and diverse datasets.

Xu et al. (Xu et al., 2019) demonstrated that existing graph neural networks lack sufficient discriminative power in distinguishing isomorphic graphs. To address this limitation, they proposed the Graph Isomorphism Network (GIN), which is theoretically as powerful as the Weisfeiler–Lehman test in detecting graph isomorphism, thereby offering significantly greater structural discrimination than earlier models. In this approach, the updated representation of each node at layer $l$ is computed as follows:

$$a_v^l = \sum_{u \in \mathcal{N}(v)} h_u^{l-1}, \qquad h_v^{(l+1)} = MLP((1+\varepsilon^l)h_v^l + a_v^l), \tag{4}$$

while the aggregation function is summation, the learnable parameter ε balances the influence between the central node and its neighbors, and the update function is implemented by applying a Multi-Layer Perceptron (MLP) combined with a summation function.

### *4.2 Graph Attention Networks (GATs)*

Graph attention networks dynamically assign different weights to neighbors using self-attention. Unlike previous methods that consider all neighbors with the same weight, these networks can focus on nodes and connections with greater importance. This feature is highly efficient in heterogeneous structures where some connections play a more important role. Among the common models in this category is the GAT method proposed by Vaswani et al.(Veličković et al., 2018). In this approach, the updated representation of the node $v$ at layer $l$ is computed as a weighted sum of its neighbors' features, where the weights are the normalized attention coefficients $\alpha_{vu}$ as follows:

$$\alpha_{vu} = softmax\left(leakyReLU\left(a^T[wh_v || wh_u]\right)\right), \qquad h_v^{l+1} = \sigma\left(\sum_{u \in N(v)} \alpha_{vu}^l w^l h_u^l\right), \tag{5}$$

while, $wh_v$ is the linear transform feature vector $v$, $\|$ is concatenation, $a^T$ is a learnable vector, leaky ReLU is the activation function, and softmax normalizes weights to [0,1]. Additionally, the authors utilized multi-head attention to enhance learning stability. Consequently, this approach employs a weighted mean as the aggregation function, incorporates a linear transformation (learnable weights) and a nonlinear activation function for updating node representations.

### *4.3 Graph Transformer Networks*

Graph transformer networks extend the transformer architecture (Vaswani et al., 2023) to graph data. Classical graph networks, including Graph Convolutional Networks, primarily rely on message passing and local neighborhood aggregation. However, these approaches exhibit inherent limitations in effectively capturing long-range dependencies and modeling complex graph structures(Chen et al., 2025). To address this challenge, Graph Transformer Networks integrate the self-attention mechanism with explicit structural information from the graph(Chen et al., 2025). This architecture enables each node to attend not only to its direct neighbors but to all nodes in the graph with varying importance weights.

One well-known implementation of the Graph Transformer is the TransformerConv layer, introduced in the PyTorch Geometric library (PyTorch Geometric, n.d.). In this layer, three fundamental vectors are defined for each

Accepted / Not Edited

node: a query vector, representing the node's information needs; a key vector, representing distinctive features of neighboring nodes; and a value vector, carrying the actual feature content. These vectors play a foundational role in this method. Attention coefficients between nodes are then computed based on the similarity between query and key matrices, along with topological information from the graph. Finally, the new representation of each node $v$ is obtained through a weighted combination of the value vectors of its neighbors. This process can be formally expressed as follows:

$$\alpha_{vu} = softmax_u\left(\frac{(W_Q h_v)^T (W_K h_u)}{\sqrt{d_k}}\right),$$
$$h_v^{l+1} = \sigma_u\left(\sum_{u \in N(v)} \alpha_{vu} W_V h_u\right), \quad (6)$$

where $W_Q$, $W_K$, and $W_V$ are the learned weight matrices for the query, key, and value projections, respectively. In this formulation, the transformer's attention function serves as the aggregation mechanism, while a linear transformation (parameterized by learnable weights) followed by a nonlinear activation function acts as the update function. Furthermore, these methods employ positional encoding to capture relative positions within the graph, enhancing structural discrimination.

## 5. Experiments

### *5.1 Data Preparation and Hyperparameter Settings*

As mentioned earlier, the dataset in this study comprises 313 real-world reinforced concrete frame structures. Each structure is modeled as a homogeneous graph, with nodes representing structural elements (beams and columns) and edges denoting their physical connections. Figure 1 illustrates an example of how a structure is converted into a graph. For each node, 42 input features—including general, floor-level, and cross-sectional properties—are considered, while two continuous variables, the width and height of the cross-section, serve as prediction targets. Accordingly, the task is formulated as a supervised regression problem. Although the model predicts continuous width and height values, a practical post-processing step is applied during inference, whereby the predicted dimensions are rounded to the nearest multiple of 5 cm to ensure compatibility with standard construction practices and commonly adopted structural dimensions.

Before training, all input features were standardized to zero mean and unit variance. Of the total dataset, 254 samples were used for training and model evaluation, while 59 samples were reserved for final testing. Model selection employed five-fold cross-validation on the training set to ensure robust performance estimation and avoid overfitting.

To optimize hyperparameters, a grid search was performed on the validation set. The initial learning rate was chosen from {0.1, 0.01, 0.001}, the number of layers from {2, 3}, and the number of neurons in hidden layers from {16, 32, 64, 128}. For attention-based models, the number of attention heads was selected from {1, 2}, and the batch size from {4, 8}.

Training was performed using the Adam optimizer with a ReduceLROnPlateau learning rate scheduler, employing a decay factor of 0.1. The maximum number of epochs was set to 250. For TransformerConv models, Laplacian positional encoding was applied to incorporate relative node positions in the graph. Model performance was evaluated using Mean Absolute Error (MAE) and the coefficient of determination ($R^2$).

A total of 96 hyperparameter combinations were evaluated for each model.

Accepted / Not Edited

The combination yielding the lowest MAE and the highest $R^2$ on the validation set was selected as the optimal configuration. Table 4 presents the best hyperparameter settings for each model. The model with this configuration was then retrained on the entire training set (254 samples) without cross-validation, and the resulting weights were saved for evaluation on the independent test set (59 samples). All implementations were performed using the PyTorch Geometric and PyTorch libraries, with training conducted on a GPU-enabled environment to accelerate computation.

*Table 4: Best hyperparameter settings for each model*

| GNN | Learning rate | Batch size | Num_layer | Num_neurons in hidden layers | Num_head | Positional encoding dimension |
|---|---|---|---|---|---|---|
| **GCN** | 0.01 | 8 | 2 | 16 | - | - |
| **Graph SAGE** | 0.1 | 8 | 2 | 16 | - | - |
| **GIN** | 0.01 | 8 | 3 | 128 | - | - |
| **GAT** | 0.01 | 8 | 3 | 64 | 2 | - |
| **Transformer Conv** | 0.01 | 8 | 3 | 64 | 2 | 15 |

### *5.2 Results Analysis*

As mentioned earlier, the models in this study were trained using the Mean Absolute Error (MAE) loss function. **Error! Reference source not found.** illustrates the evolution of the loss function over the epochs. The continuous decrease of the curve indicates proper learning and convergence of the model. Moreover, no signs of overfitting are observed on the evaluation dataset.

After confirming proper training, the final performance of the models was evaluated on the test set. The results of applying the graph-based models to the proposed dataset are presented in Table 5. As shown, all examined models captured structural patterns; however, the accuracy and stability of their predictions varied. Although classical models achieved satisfactory performance, using more advanced architectures, such as graph transformer networks, resulted in significant improvements in evaluation metrics. This comparison suggests that the graph transformer model, by leveraging transformer mechanisms, is better able to capture structural dependencies and section-level features of elements. The findings also indicate that, for modeling the behavior of reinforced concrete frame structures—where element relationships can be both local and non-local—using more powerful architectures provides a substantial advantage.

Accepted / Not Edited

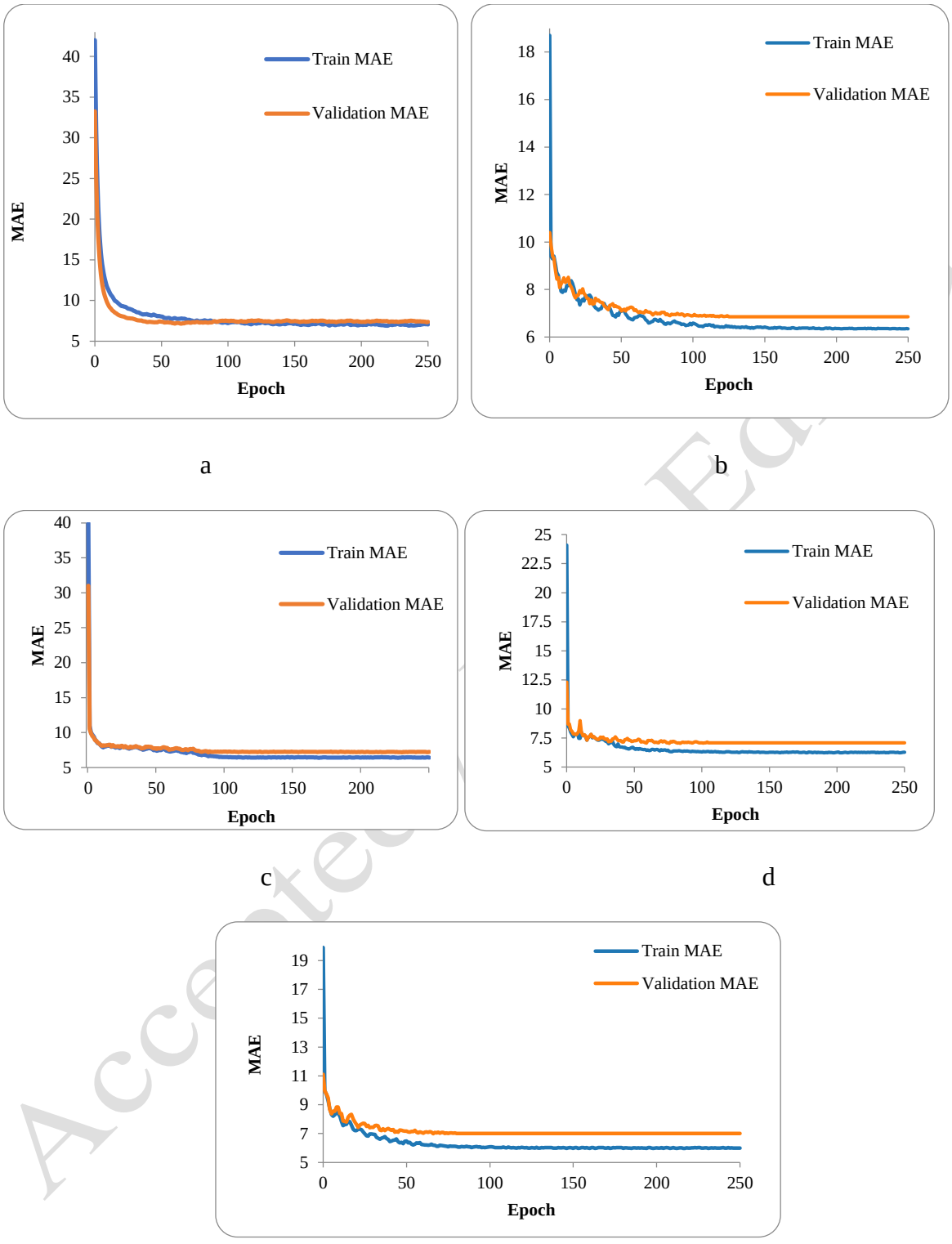



*Figure 4: Training and evaluation loss curves for each of the graph-based models: a) GCN, b) GraphSAGE, c) GIN, d) GAT, e) Transformer Conv*

Accepted Manuscript

*Table 5: Comparison of graph-based model performance based on $R^2$ for predicting cross-section width and height (best results highlighted in bold).*

| GNN | R2 _W | R2 _H |
|---|---|---|
| GCN | 0.46 | 0.56 |
| Graph SAGE | 0.59 | 0.66 |
| GIN | 0.6 | 0.65 |
| GAT | 0.58 | 0.6 |
| **Transformer Conv** | **0.7** | **0.74** |

To further evaluate model performance on individual structures, nine buildings with 1 to 9 stories were randomly selected from the test dataset. The predicted section dimensions of elements in these structures were compared to the actual values. For each building, the mean error, standard deviation of the error, and mean percentage error across all elements were calculated and are reported in Table 6. This structural-level analysis indicates that the Transformer Conv model, besides high element-level accuracy, exhibits robust stability and effectiveness in predicting section dimensions across buildings with varying configurations.

*Table 6: Structural-level evaluation of predicted section dimensions for selected test buildings. Mean error, standard deviation, and mean percentage error across all elements are reported.*

| Number of stories | Exec. Time | MAE width (cm, mean ± SD) | MAE height (cm, mean ± SD) | Mean % error width | Mean % error height |
|---|---|---|---|---|---|
| 1 | 0.34 | 5+-4.2 | 3.9+-4.4 | 0.14 | 0.11 |
| 2 | 0.35 | 4.7+-6.3 | 4+-5.2 | 0.1 | 0.1 |
| 3 | 0.36 | 3.2+-5 | 2.9+-4.99 | 0.08 | 0.08 |
| 4 | 0.37 | 5+-5.2 | 4.2+-4.2 | 0.1 | 0.1 |
| 5 | 0.39 | 4.5+-3.9 | 3.9+-3.2 | 0.12 | 0.09 |
| 6 | 0.39 | 5.8+-6.4 | 4.9+-5.1 | 0.12 | 0.12 |
| 7 | 0.4 | 4+-4.2 | 4.5+-4.4 | 0.08 | 0.11 |
| 8 | 0.52 | 4.5+-3.6 | 5.2+-3 | 0.08 | 0.1 |
| 9 | 0.73 | 5.7+-4.85 | 4.2+-3.38 | 0.1 | 0.09 |

### *5.3 Hyperparameter Analysis*

To obtain a more detailed understanding and a comprehensive evaluation of the Graph Transformer network, additional experiments were conducted by varying key hyperparameters. Specifically, the influence of the learning rate, number of layers, hidden layer size, and number of attention heads on model performance was systematically investigated. The results of these experiments are summarized in Tables 7–10. Table 7 specifically reports the impact of the learning rate, while Tables 8–10 present the effects of the number of layers, hidden neurons, and attention heads, respectively. As shown in Table 7, a lower learning rate (0.01) led to more stable convergence behavior, yielding higher $R^2$ and lower MAE values.

*Table 7: Effect of initial learning rate variation on model performance*

| Learning rate | MAE (cm) | R2 _W | R2_H |
|---|---|---|---|
| 0.1 | 7.22 | 0.37 | 0.51 |
| **0.01** | **6.86** | **0.43** | **0.56** |
| 0.001 | 7.03 | 0.39 | 0.52 |
| 0.0001 | 7.22 | 0.38 | 0.51 |

Next, to examine the effect of the number of layers, the model was evaluated with configurations ranging from two to five layers, and the results are reported in Table 8. The findings show that increasing the depth from two to three layers yielded a noticeable improvement in performance. However, adding additional layers (four or five) did not

Accepted / Not Edited

provide further benefits and, in some cases, even degraded the results. This suggests that increasing network depth beyond the optimal point is not only ineffective but may also lead to issues such as over-smoothing and a diminished ability to distinguish node features. Consequently, the three-layer configuration was adopted as the optimal architecture.

*Table 8: Effect of the number of layers on model performance*

| Num_layer | MAE (cm) | R2 _W | R2_H |
|---|---|---|---|
| 2 | 7.02 | 0.39 | 0.53 |
| **3** | **6.86** | **0.43** | **0.56** |
| 4 | 6.9 | 0.42 | 0.56 |
| 5 | 6.9 | 0.39 | 0.54 |

Furthermore, the effect of the number of attention heads was evaluated with values of 1, 2, 3, and 4. As shown in Table 9, increasing the number of heads up to a certain point (two heads) enhances the model's capacity to capture both local and non-local patterns, resulting in a noticeable improvement in performance. However, using additional heads not only increases computational cost but may also introduce training instability and, in some cases, degrade performance. Consequently, two attention heads were selected as the optimal configuration, providing the best balance between predictive accuracy and computational efficiency

*Table 9:* Effect of attention heads on model performance

| Num_head | MAE (cm) | R2 _W | R2_H |
|---|---|---|---|
| 1 | 6.96 | 0.4 | 0.54 |
| **2** | **6.86** | **0.43** | **0.56** |
| 3 | 6.91 | 0.42 | 0.55 |
| 4 | 6.92 | 0.43 | 0.56 |

To investigate the effect of hidden layer dimensions in the Graph Transformer network, the number of neurons in the hidden layers was evaluated with values of 16, 32, 64, and 128. The results presented in Table 10 indicate that increasing the number of neurons from 16 to 64 improves model performance. However, further increasing the number to 128 neurons did not yield significant gains, with performance metrics remaining nearly identical to the 64-neuron configuration. These findings suggest that excessively large hidden layer dimensions do not enhance model efficiency and may increase computational cost and the risk of overfitting. Consequently, 64 neurons were selected as the optimal configuration for the hidden layers.

*Table 10:* Effect of hidden layer neurons on model performance

| Num_neurons in hidden layers | MAE (cm) | R2 _W | R2_H |
|---|---|---|---|
| 16 | 7.03 | 0.39 | 0.53 |
| 32 | 6.96 | 0.39 | 0.53 |
| **64** | **6.86** | **0.43** | **0.56** |
| 128 | 6.94 | 0.42 | 0.56 |

Overall, the experimental results demonstrate that proper tuning of these hyperparameters is critical for enhancing the model's generalization ability and achieving optimal performance. These findings emphasize that designing an efficient model not only depends on selecting an appropriate architecture but also requires careful adjustment of hyperparameters tailored to the characteristics of the data and the specific problem being addressed

## 6. Conclusion

In recent years, deep learning–based approaches have gained increasing attention in structural engineering. However, their effectiveness depends heavily on the

Accepted / Not Edited

availability of comprehensive, realistic, and code-compliant datasets that can fully represent the design domain. To date, no dataset has been available that simultaneously captures the diversity of element arrangements and the variability of typical reinforced concrete buildings for predicting beam and column sections.

In this study, the key features required for predicting the dimensions of reinforced concrete beams and columns were identified, categorized, and structured. Based on these features, a standardized dataset named AV_SStruct313, containing data from 313 real reinforced concrete projects in Iran, was developed and introduced. As the first standardized native dataset in this field, it provides a valuable foundation for intelligent and automated structural design.

The five graph-based neural network models—GCN, GIN, GraphSAGE, GAT, and Graph Transformer—were trained and evaluated using the proposed dataset. The results demonstrated that the Graph Transformer achieved the highest accuracy, with an average prediction error of approximately 5 cm and an inference time of less than one second. From a practical perspective, this error corresponds approximately to one standard design increment commonly adopted in reinforced concrete design, where beam and column cross-sectional dimensions are typically selected in 5 cm intervals. Consequently, the predicted dimensions are generally close to the values that would be considered during preliminary structural design.

These outcomes indicate that graph-based learning can significantly enhance the speed, accuracy, and autonomy of the design process, effectively reduce engineers' manual effort and accelerate the iterative design workflow. The proposed graph-based approach thus represents an important step toward the automation of structural design in reinforced concrete systems.

## 7. Practical Implications

Beyond the quantitative evaluation results, the proposed graph-based approach has several practical implications for structural engineering practice. The Graph Transformer model can be utilized during the preliminary design stage to provide rapid estimates of beam and column cross-sectional dimensions prior to detailed structural analysis. This capability may reduce the conventional trial-and-error process required for selecting initial member sizes, thereby shortening the overall design cycle.

Rather than replacing engineering judgment, the proposed approach is intended to serve as a decision-support tool by providing initial design **estimates** learned from real-world, code-compliant structural projects. The outputs of the model are therefore intended to be used as initial design candidates, which are subsequently verified and refined through conventional structural analysis and design procedures.

Although the current implementation is a research prototype developed in Python, the proposed approach has the potential to be integrated into structural design environments such as ETABS or BIM-based platforms through appropriate software interfaces or plugin development. This integration could support more efficient t, consistent, and data-driven workflows in future structural design practice.

## 8. Limitations and Future work

Despite the promising results, several limitations exist in the current study, which also define directions for future research. First, the dataset is limited to 313 reinforced concrete building projects from a specific regional context. Expanding the dataset to include a wider range of structural

Accepted / Not Edited

typologies, geographical regions, and design practices is expected to improve model generalization capability.

Second, although several graph-based architectures were evaluated, more advanced and hybrid approaches, such as physics-informed graph neural networks and enhanced Graph Transformer variants, may further improve prediction accuracy and robustness.

Third, the current graph-based approach focuses primarily on geometric prediction of beam and column dimensions. Future research may integrate additional design considerations, including seismic performance, reinforcement detailing, and construction cost optimization, to develop a more comprehensive decision-support system.

Finally, validation of the proposed approach on large-scale, high-rise, and industrial structures, as well as its implementation in real engineering design workflows, is necessary to evaluate its practical applicability and readiness for real-world deployment.

## 9. Acknowledgments

The authors sincerely thank the **community of design engineers in Qom** for providing the project samples used in this dataset, and **NiareshGran_Avgooon_Saze** Company for their support in extracting the data from these samples.

The authors also acknowledge the use of artificial intelligence (AI)-based tools for language editing and improving the clarity and readability of the manuscript. The authors take full responsibility for the content and final version of the manuscript.

## 10. Data Availability Statement

From the 313 samples collected, a subset of 59 samples has been made publicly available via the Zenodo repository (DOI:[ https://doi.org/10.5281/zenodo.23076510]). Access to the remaining data is restricted and may be granted only upon request to verified researchers affiliated with recognized universities or research institutions. The data are intended exclusively for non-commercial research purposes. Requesters are required to clearly state their research objectives prior to data access and to share the findings with the data provider upon completion of their study.

## 11. References

Adeli, H. and Kumar, S. (1995) "Distributed Genetic Algorithm for Structural Optimization," *Journal of Aerospace Engineering*, 8(3), 156–163, https://doi.org/10.1061/(ASCE)0893-1321(1995)8:3(156).

Al-Bizri, S. and MacLeod, I. A. (2024) "Structural Design Processes," In *Structural Design of Buildings: Fundamentals in Design, Management and Sustainability*, edited by Feng Fu and David Richardson, Emerald Publishing Limited, https://doi.org/10.1680/978-1-83549-576-620241004.

Ampanavos, S., Nourbakhsh, M. and Cheng, C.-Y. (2022) "Structural Design Recommendations in the Early Design Phase Using Machine Learning," In *Computer-Aided Architectural Design. Design Imperatives: The Future Is Now*, edited by David Gerber, Evangelos Pantazis, Biayna Bogosian, Alicia Nahmad, and Constantinos Miltiadis, Springer, https://doi.org/10.1007/978-981-19-1280-1_12.

Accepted / Not Edited

Building and Housing Research Center (BHRC) (2014) *Iranian Code of Practice for Seismic Resistant Design of Buildings, Standard No. 2800*. 4th ed., Tehran, Iran.

Chang, K.-H. and Cheng, C.-Y. (2020) "Learning to Simulate and Design for Structural Engineering," *Proceedings of the 37th International Conference on Machine Learning*, 119, 1426–1436, https://proceedings.mlr.press/v119/chang20a.html.

Chen, D., Schulz, T. H. and Borgwardt, K. (2025) "Learning Long Range Dependencies on Graphs via Random Walks," *International Conference on Learning Representations*, https://arxiv.org/abs/2406.03386v2.

Chou, Y.-T., Chang, W.-T., Jean, J. G., Chang, K.-H., Huang, Y.-N. and Chen, C.-S. (2024) "StructGNN: An Efficient Graph Neural Network Framework for Static Structural Analysis," *Computers & Structures*, 299, 107385, https://doi.org/10.1016/j.compstruc.2024.107385.

Chou, Y.-T., Kuo, P.-C., Li, K.-Y., Chang, W.-T., Huang, Y.-N. and Chen, C.-S. (2025) "Inductive Graph-Based Long Short-Term Memory Network for the Prediction of Nonlinear Floor Responses and Member Forces of Steel Buildings Subjected to Orthogonal Horizontal Ground Motions," *Earthquake Engineering & Structural Dynamics*, 54(2), 491–507, https://doi.org/10.1002/eqe.4264.

Computers and Structures Inc. (CSI) (2016) *ETABS: Integrated Building Design Software*, Berkeley, California, USA.

Ellingwood, B., Maes, M., Bartlett, M. F., et al. (2025) "Development of Methods of Structural Reliability," *Structural Safety*, 113, 102474, https://doi.org/10.1016/j.strusafe.2024.102474.

Fei, Y., Liao, W., Huang, Y. and Lu, X. (2022) "Knowledge-Enhanced Generative Adversarial Networks for Schematic Design of Framed Tube Structures," *Automation in Construction*, 144, 104619, https://doi.org/10.1016/j.autcon.2022.104619.

Gallet, A., Liew, A., Hajirasouliha, I. and Smyl, D. (2024) "Machine Learning for Structural Design Models of Continuous Beam Systems via Influence Zones," *Inverse Problems*, 40(5), 055011, https://doi.org/10.1088/1361-6420/ad3334.

Gupta, K. K. and Meek, J. L. (1996) "A Brief History of the Beginning of the Finite Element Method," *International Journal for Numerical Methods in Engineering*, 39(22), 3761–3774, https://doi.org/10.1002/(SICI)1097-0207(19961130)39:22%3C3761::AID-NME22%3E3.0.CO;2-5.

Gupta, R. S. and Gupta, R. S. (2014) *Principles of Structural Design*, CRC Press, https://doi.org/10.1201/b16857.

Hamilton, W. L., Ying, R. and Leskovec, J. (2017) "Inductive Representation Learning on Large Graphs,"

Accepted / Not Edited

*Advances in Neural Information Processing Systems 30*, 1024–1034.

Kanyilmaz, A., Tichell, P. R. N. and Loiacono, D. (2022) "A Genetic Algorithm Tool for Conceptual Structural Design with Cost and Embodied Carbon Optimization," *Engineering Applications of Artificial Intelligence*, 112, 104711, https://doi.org/10.1016/j.engappai.2022.104711.

Kipf, T. N. and Welling, M. (2017) "Semi-Supervised Classification with Graph Convolutional Networks," *International Conference on Learning Representations*, https://doi.org/10.48550/arXiv.1609.02907.

Kumar, P., Sharma, S. and Pratap, B. (2025) "Prediction of Compressive Strength of Geopolymer Fiber Reinforced Concrete Using Machine Learning," *Civil Engineering Infrastructures Journal*, 58(1), 173–182, https://doi.org/10.22059/ceij.2024.364871.1956.

Liao, W., Lu, X., Fei, Y., Gu, Y. and Huang, Y. (2024) "Generative AI Design for Building Structures," *Automation in Construction*, 157, 105187, https://doi.org/10.1016/j.autcon.2023.105187.

Liu, W. K., Li, S. and Park, H. (2022) "Eighty Years of the Finite Element Method: Birth, Evolution, and Future," *Archives of Computational Methods in Engineering*, 29(6), 4431–4453, https://doi.org/10.1007/s11831-022-09740-9.

Mathewson, G. (2023) "A History of U.S. Building Codes," *Fine Homebuilding*, https://www.finehomebuilding.com/2023/07/19/a-history-of-u-s-building-codes.

Mendel, J. M. (1995) "Fuzzy Logic Systems for Engineering: A Tutorial," *Proceedings of the IEEE*, 83(3), 345–377, https://doi.org/10.1109/5.364485.

Ministry of Roads and Urban Development (2019) *Part 6: Loads on Buildings*. 4th ed., Road, Housing and Urban Development Research Center.

Ministry of Roads and Urban Development (2020) *Part 9: Design and Construction of Reinforced Concrete Buildings*, Road, Housing and Urban Development Research Center.

Nimtawat, A. and Nanakorn, P. (2011) "Simple Particle Swarm Optimization for Solving Beam-Slab Layout Design Problems," *Procedia Engineering*, 14, 1392–1398, https://doi.org/10.1016/j.proeng.2011.07.175.

Pandey, M., Viljoen, C., Way, A., et al. (2025) "Life Safety in the Reliability-Based Design and Assessment of Structures," *Structural Safety*, 113, 102453, https://doi.org/10.1016/j.strusafe.2024.102453.

Perez, R. E. and Behdinan, K. (2007) "Particle Swarm Approach for Structural Design Optimization," *Computers & Structures*, 85(19), 1579–1588, https://doi.org/10.1016/j.compstruc.2006.10.013.

Accepted / Not Edited

Pizarro, P. N. and Massone, L. M. (2021) "Structural Design of Reinforced Concrete Buildings Based on Deep Neural Networks," *Engineering Structures*, 241, 112377, https://doi.org/10.1016/j.engstruct.2021.112377.

Pizarro, P. N., Massone, L. M., Rojas, F. R. and Ruiz, R. O. (2021) "Use of Convolutional Networks in the Conceptual Structural Design of Shear Wall Buildings Layout," *Engineering Structures*, 239, 112311, https://doi.org/10.1016/j.engstruct.2021.112311.

PyTorch Geometric (n.d.) "Torch_geometric.Nn.Conv.TransformerConv," https://pytorch-geometric.readthedocs.io/en/2.5.2/generated/torch_geometric.nn.conv.TransformerConv.html.

Qin, S., Liao, W., Huang, Y., et al. (2025) "Intelligent Design for Component Size Generation in Reinforced Concrete Frame Structures Using Heterogeneous Graph Neural Networks," *Automation in Construction*, 171, 105967, https://doi.org/10.1016/j.autcon.2025.105967.

Shahrouzi, M. and Pashaei, M. (2013) "TunnedTTStochastic Directional Search: An Efficient Heuristic for Structural Optimization of Building Frames," *Scientia Iranica*, 20(4), 1124–1132.

Singaravel, S., Suykens, J. and Geyer, P. (2019) "Deep Convolutional Learning for General Early Design Stage Prediction Models," *Advanced Engineering Informatics*, 42, 100982, https://doi.org/10.1016/j.aei.2019.100982.

Soledispa, C. E., Pizarro, P. N. and Massone, L. M. (2024) "Optimizing Reinforced Concrete Walls and Columns through Artificial Neural Networks with Structural Neighbor-Based Features," *Journal of Building Engineering*, 89, 109223, https://doi.org/10.1016/j.jobe.2024.109223.

Sriram, D., Maher, M. L. and Fenves, S. J. (1985) "Knowledge-Based Expert Systems in Structural Design," *Computers & Structures*, 20(1), 1–9, https://doi.org/10.1016/0045-7949(85)90047-1.

Sun, H., Burton, H. V. and Huang, H. (2021) "Machine Learning Applications for Building Structural Design and Performance Assessment: State-of-the-Art Review," *Journal of Building Engineering*, 33, 101816, https://doi.org/10.1016/j.jobe.2020.101816.

Vaswani, A., Shazeer, N., Parmar, N., et al. (2023) "Attention Is All You Need." arXiv:1706.03762, https://doi.org/10.48550/arXiv.1706.03762.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P. and Bengio, Y. (2018) "Graph Attention Networks." arXiv:1710.10903, https://doi.org/10.48550/arXiv.1710.10903.

Xia, J., Liao, W., Han, B., Zhang, S. and Lu, X. (2025) "Intelligent Co-Design of Shear Wall and Beam Layouts Using a Graph Neural Network," *Automation in Construction*, 172,

Accepted / Not Edited

106024, https://doi.org/10.1016/j.autcon.2025.106024.

Xu, K., Hu, W., Leskovec, J. and Jegelka, S. (2019) "How Powerful Are Graph Neural Networks?" arXiv:1810.00826, https://doi.org/10.48550/arXiv.1810.00826.

Zhang, Y. and Mueller, C. (2017) "Shear Wall Layout Optimization for Conceptual Design of Tall Buildings," *Engineering Structures*, 140, 225–240, https://doi.org/10.1016/j.engstruct.2017.02.059.

Zhao, P., Fei, Y., Huang, Y., Feng, Y., Liao, W. and Lu, X. (2023) "Design-Condition-Informed Shear Wall Layout Design Based on Graph Neural Networks," *Advanced Engineering Informatics*, 58, 102190, https://doi.org/10.1016/j.aei.2023.102190.

Zhao, P., Liao, W., Huang, Y. and Lu, X. (2023) "Intelligent Beam Layout Design for Frame Structure Based on Graph Neural Networks," *Journal of Building Engineering*, 63, 105499, https://doi.org/10.1016/j.jobe.2022.105499.

Zhao, P., Liao, W., Huang, Y. and Lu, X. (2024) "Beam Layout Design of Shear Wall Structures Based on Graph Neural Networks," *Automation in Construction*, 158, 105223, https://doi.org/10.1016/j.autcon.2023.105223.

Zhao, P., Liao, W., Xue, H. and Lu, X. (2022) "Intelligent Design Method for Beam and Slab of Shear Wall Structure Based on Deep Learning," *Journal of Building Engineering*, 57, 104838, https://doi.org/10.1016/j.jobe.2022.104838.

Accepted / Not Edited