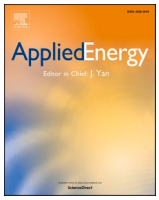
Applied Energy 367 (2024) 123431

Contents lists available at ScienceDirect

Applied Energy

journal homepage: www.elsevier.com/locate/apenergy



EPlus-LLM: A large language model-based computing platform for automated building energy modeling

Gang Jiang [a], Zhihao Ma [a], Liang Zhang [b], Jianli Chen [a,*]

[a] The University of Utah, United States
[b] The University of Arizona, United States

HIGHLIGHTS

- This study represents the pioneering effort in customizing LLM for auto-modeling.
- A large language model is used to digest natural language descriptions for modeling.
- Integration of four prompts enhances the flexibility and versatility of our platform.
- Our platform provides a human-AI interface and reduces building modeling efforts.

ARTICLE INFO

Keywords:
Large language models
Artificial intelligence
Machine learning
Building energy modeling
Automated simulation

ABSTRACT

Establishing building energy models (BEMs) for building design and analysis poses significant challenges due to demanding modeling efforts, expertise to use simulation software, and building science knowledge in practice. These make building modeling labor-intensive, hindering its widespread adoptions in building development. Therefore, to overcome these challenges in building modeling with enhanced automation in modeling practice, this paper proposes Eplus-LLM (EnergyPlus-Large Language Model) as the auto-building modeling platform, building on a fine-tuned large language model (LLM) to directly translate natural language description of buildings to established building models of various geometries, occupancy scenarios, and equipment loads. Through fine-tuning, the LLM (i.e., T5) is customized to digest natural language and simulation demands from users and convert human descriptions into EnergyPlus modeling files. Then, the Eplus-LLM platform realizes the automated building modeling through invoking the API of simulation software (i.e., the EnergyPlus engine) to simulate the auto-generated model files and output simulation results of interest. The validation process, involving four different types of prompts, demonstrates that Eplus-LLM reduces over 95% modeling efforts and achieves 100% accuracy in establishing BEMs while being robust to interference in usage, including but not limited to different tones, misspells, omissions, and redundancies. Overall, this research serves as the pioneering effort to customize LLM for auto-modeling purpose (directly build-up building models from natural language), aiming to provide a user-friendly human-AI interface that significantly reduces building modeling efforts. This work also further facilitates large-scale building model efforts, e.g., urban building energy modeling (UBEM), in modeling practice.

1. Introduction

1.1. Background

Buildings account for about 36% of total global energy [1]. Building energy models (BEMs) have increasingly stood as a crucial tool to simulate building energy use and indoor environment, serving various purposes such as building retrofits, sustainability, and decarbonization [2,3]. However, creating BEMs is time consuming, requiring skills of software usage, especially for large and complex buildings. To build an accurate building simulation model, building modelers have to repeatedly modify design parameters and debug the models, in order to ensure validity of modeling outcomes (e.g., energy use and indoor environment status). A user-friendly and automated building simulation platform is

* Corresponding author.
E-mail address: jianli.chen@utah.edu (J. Chen).

https://doi.org/10.1016/j.apenergy.2024.123431
Received 16 January 2024; Received in revised form 3 May 2024; Accepted 8 May 2024

particularly beneficial for facilitating simulation-intensive applications, e.g., calibration and urban building energy modeling (UBEM).

In recent years, pre-trained large language models (PTMs) have found remarkable success in natural language processing (NLP). Trained on extensive datasets, PTMs exhibit adaptability to a range of downstream tasks and applications through fine-tuning or context learning. Large language models (LLMs), with billions of parameters and unique model architectures involving the attention mechanism, have developed a strong capability for natural language tasks. This enables them to capture subtle nuances and contextual information within language, performing well across various applications. Furthermore, these models exhibit multimodal capabilities, excelling in processing diverse types of information, including text, images, and sound. The versatility of LLMs not only positions them as powerful tools in NLP and computer vision (CV), but also holds the potential to open up new possibilities and opportunities in the fields of building and simulation. The intersection of LLMs with building simulation is anticipated to bring about revolutionary advancements and innovative solutions and will offer fresh insights and prospects for future technological development.

1.2. Existing practice and development of building modeling and LLM

Engineers and researchers begin to model buildings (or calculate building loads) using computers or computing methods ever since 19th century. These early modeling were primarily based on simplified methods (e.g., Degree-day method and Bin method [4]) to estimate the energy demand of heating, ventilation, and air conditioning (HVAC) systems for building design and operation. With development of modeling engines, existing modeling software, such as ANSYS [5], Dymola [6], TRNSYS [7], DeST [8], EnergyPlus [9], ISO 13790 [10], and VDI 6007 [11], are mostly based on heat balance principles and ordinary differential equation (ODE) in the modeling process. These applications use building details, material properties, and climate data as inputs to produce reliable building simulation results. Among them, EnergyPlus, developed by the US Department of Energy (DOE), is one of the most widely used software in the field of building simulation. EnergyPlus adopts the physics-based modeling method, simulating the complex interactions between various components in the building (such as walls, windows, HVAC systems, etc.) of different physical characteristics. The high-fidelity physics principle enables EnergyPlus to reliably capture dynamics of building systems, providing reliable simulation results to support various building and energy applications.

Despite the development, modeling buildings are non-trivial. Accurately characterizing and creating models for complex building systems demands substantial efforts, although graphical user interfaces (GUIs), such as DesignBuilder [12] and OpenStudio [13], were developed to simplify the modeling process. Users still need to grapple with a plethora of options and parameters in modeling through GUIs. To automate the creation and modification of building energy models, other researchers have developed Python libraries like Eppy [14]. However, it is crucial to recognize that using these packages for automated building modeling also requires strong coding ability to effectively utilize these advanced tools.

With significant advancement in computation these years, the emergence of LLMs presents as a promising solution to overcome complexity and accessibility of building modeling. LLMs have achieved remarkable success in NLP, leading to a paradigm shift from supervised learning to pre-training followed by fine-tuning [15]. As one of the pioneering works, Peters et al. [16] proposed embeddings of language models to learn contextual word representations using bi-directional LSTMs, and then applied the pre-trained embeddings to downstream tasks. This approach demonstrated dramatic improvements in a wide range of NLP tasks. Generative Pre-Training (GPT) [17], which leverages transformers [18] for language modeling, also illustrated the great potential of PTMs to support different downstream tasks. After the success of GPT, there has been significant interest in scaling up and exploring pre-trained language models, leading to the development of numerous LLMs. Examples include BERT [19], text-to-text transfer transformers (T5) [20], PaLM [21], and LLaMA [22].

Building on the success of PTMs described above, these models have been increasingly used to support various downstream applications, including but not limited to computer vision [23], speech processing [24], etc. They have also been applied in generative tasks within domains like chemistry [25], geography [26], and code generation [27]. LLMs possess the unique ability to understand and generate human-like text, making them well-suited for interpreting natural language descriptions. In the context of building modeling, this capability holds significant potential to facilitate user's modeling and reduce simulation efforts. By leveraging LLMs in automated building simulation and design, users can describe their simulation visions and requirements in a more intuitive manner: directly using natural language descriptions instead of complex design with simulation software. However, adapting LLMs for building modeling with auto-model generation is challenging due to the following issues:

(1) Precise Model Description

Building modeling is a complex process involving numerous inputs, simulation configurations, and closely coupled sets of equations in calculation. Even a slight error of punctuation (e.g., missing a decimal somewhere) or a small discrepancy of geometry (e.g., the edge of walls defined by coordinates has 0.01 m deviations from its connected walls) will lead to the simulation failure (the modeling engine will report severe errors and terminate the simulation). Achieving this high level of accuracy poses a significant challenge for LLMs, because general LLM outputs typically exhibit varying degrees of randomness. Consequently, LLMs for our building modeling task) must demonstrate remarkable precision during generation. Given that LLMs comprise billions of parameters, training a capable foundation model to effectively and accurately generate building model descriptions demands significant computational resources and time. The current diversity in LLM architectures further complicates the selection of appropriate models and sizes to accurately cater to the requirements of building simulation. Blindly opting for a larger model does not necessarily equate to better performance [28].

(2) Customization of LLM for Building Modeling Tasks

Another critical challenging lies in customizing LLMs for building modeling. This challenge arises from (1) the need for domain-specific knowledge to customize LLMs for building modeling. LLMs have not been pre-trained on building modeling tasks, making it difficult to be directly used for automated building modeling, (2) The high level of required consistency in building modeling tasks. Unlike generic NLP tasks, auto-creation of building modeling files requires precision and consistency for the generated models to be valid. The necessity of understanding diverse language description formats further adds extra complexity to maintain consistency in building modeling. Users may use synonymous terms, varied sentence structures, or even misspellings to convey identical building attributes, requiring LLMs to discern and reconcile these linguistic subtleties to capture user intent accurately.

To overcome these, the "pre-training then fine-tuning" paradigm has emerged as an effective and flexible method for creating sub-models to customize LLMs for specific applications. This paradigm is favored because: (1) Pre-training models involve billions of parameters, advanced computational architectures, and large corpus datasets, endowing LLMs with language flexibility, accuracy performance, and long context window for downstream tasks. (2) Fine-tuning process relies on self-learning and domain-specific data (e.g., building simulation dataset), enabling the model to grasp the relevance and focal points of content within user inputs (i.e., building modeling requirements). Therefore, fine-tuning is an effective way for our purposes in handling

complex and highly customized building modeling demands.

InstructGPT and ChatGPT [28] are fine-tuned with effective prompts, aligning with user intentions on various tasks. Notably, the 1.3B InstructGPT outperforms the 175B GPT-3 despite having 100 times fewer parameters. ChatGPT excels in various downstream tasks post fine-tuning. The fine-tuned T5 checkpoint, Flan-t5 [29], achieves superior performance in evaluation benchmarks, even when compared to larger models. Pavlyshenko [30] fine-tuned Llama 2 for multitask analysis of financial news. CodeT5+ [31] is an encoder-decoder family tailored for diverse downstream code tasks. Surprisingly small and cost-effective, Alpaca [32], fine-tuned from Meta's Llama 7B, achieves superior performance. In conclusion, the superior performance of these fine-tuned models underscores the effectiveness of the fine-tuning approach in customizing language models (LLMs) for specific applications. A fine-tuned model can achieve high accuracy in various downstream tasks with minimal additional training data. By providing different prompts during fine-tuning, LLMs can gather more information tailored to specific tasks, thereby guiding their generation with precision.

1.3. New opportunities for building model design and simulation

For building simulation, despite the existence of numerous physical-based and data-driven modeling methods, practitioners are still plagued by challenges such as the accessibility of software or technologies, data reliability, and technological complexity. Creating a high-fidelity building model is typically time-consuming, even for experienced modelers, not to mention non-tech-savvy users. The emergence of LLMs presents new opportunities to completely automate building design and simulation from natural language description of buildings by modelers. LLMs are designed to comprehend natural language and generate sentences through pre-training. Fine-tuning is then performed based on the pre-trained LLM, utilizing smaller field datasets and fewer computing resources to further enhance its performance and availability in downstream tasks. Integrating LLMs into building design and modeling can significantly reduce the efforts, including knowledge, technologies, data, hardware, manpower, and time requirements, needed to construct physics-based building models. This will further foster the widespread adoption of advanced building modeling in practical applications. However, despite the great potential shown by LLMs in natural language applications, their application with the automation of building modeling is still limited due to the difficulty of reaching the precision requirements and effectively handling customization of building modeling tasks as mentioned above. Hence, there still lacks a customized LLM-based platform for auto-building modeling.

In this study, we proposed Eplus-LLM as the computing platform customized for automated building modeling tasks, capable of directly and automatically translating natural language descriptions of modelers into high-fidelity building models with precision (overcoming two challenges mentioned in Section 1.2). This platform integrates the LLM architecture, i.e., the Text-to-Text Transfer Transformer (T5), with the physics-based modeling engine, EnergyPlus, aiming to create a user-friendly and automated process to support modelers (e.g., from consulting firms, architecture firms, academia, building system design engineers, etc.) in building modeling. Using Eplus-LLM platform is able to (1) make modeling easier with AI-empowered communication between modeling software and modelers. The developed platform aims to reduce burdens and required efforts of modelers in building modeling. Natural language is the natural way of humans in communications, not only with other humans, but also with machines. LLM has demonstrated significant potential to automate tasks under natural instruction of humans in diverse fields. However, this automation and intelligence are not fully leveraged in the building modeling field yet. The developed platform aims to realize the future goal of fully automated building modeling without significant efforts as in the current practice, and (2) handle more flexible inputs. Our platform supports structured and unstructured ways of modeling inputs. For unstructured inputs, our platform can handle natural language with different tones, grammar errors, and omitted words. Users can also use structured data inputs, e.g., specifying dimensions with height: xxx; width: xxx; length: xxx …, without worrying about deviations from expected format.

For validation of the effectiveness of our proposed framework in auto-building modeling, we conducted a total of 152 validation instances to verify the effectiveness and stability of the developed computing platform for auto and accurate building modeling from natural language descriptions of models, achieving 100% accuracy. We also assessed the robustness and anti-interference ability of the platform in generation by introducing various types of noise and unseen prompts.

The paper is organized as follows: Section 2 provides an overview, model architecture, and training details of our proposed Eplus-LLM platform. In Section 3, we demonstrate the effectiveness, robustness, and versatility of our platform through validation and evaluation. Section 4 discusses insights gained during the exploration of LLM-based auto-building modeling, and Section 5 concludes our study.

2. Methodology

2.1. Overview of proposed Eplus-LLM platform

Fig. 1 illustrates the framework of the proposed Eplus-LLM, a fine-tuned LLM capable of rapidly and automatedly building up building models directly from natural language. The objective is to overcome the complexities, necessity of specific knowledge, familiarity with software, and challenges in human-modeling interaction of traditional building simulation and modeling. The Eplus-LLM platform integrates physics-based simulation software (i.e., EnergyPlus) with an LLM architecture (i.e., T5), providing a human-AI interface that comprehends description of building models by natural language and perform building modeling from dialogue.

As the core of the proposed framework, Eplus-LLM primarily consists of three modules: Prompting, LLM Architecture, and Auto-simulation, as shown in the Fig. 1. The Prompting module receives modeling commands of users in natural language descriptions from the interface of human-model interaction. Following tokenization, embedding, and position encoding, the natural language descriptions of specified simulation demands are transformed into numerical and tokenized multi-dimensional features (Fig. 2a). These features are then accepted and comprehended by the LLM, with the attention module serving as its core. By constructing Q (Query), K (Key), and V (Value) matrices (the Q represents the specific information the model is focusing on; the K is the features of the input, determining the content related to the Q; the V is used to provide the actual information content), the LLM is able to effectively focus on the most significant information. In building modeling tasks, Q, K, and V matrices can be understood as multi-dimensional relationships among various building input parameters and interconnected relationships among building sub-systems. They facilitate the mapping of simulation instructions to model descriptions and the generation of simulation files, i.e., EnergyPlus IDF files. Afterward, the API of simulation software is automatically activated through scripting, then outputting modeling results, e.g., indoor temperature and building energy consumptions (Fig. 2 b). In Section 2.2, the foundation model is elaborated in detail, explaining how natural language descriptions are transformed into tokens recognizable by our Eplus-LLM. In Section 2.3, the fine-tuning and customizing process for auto-building modeling tasks are discussed in detail, highlighting our data preparation and processing, the designed prompts, without fine-tuning baseline, and provides insights into the fine-tuning process and details.

2.2. Foundation LLM

In this work, we utilized the Flan-T5 model [29] as a foundation model, fine-tuning it with training data derived from building modeling

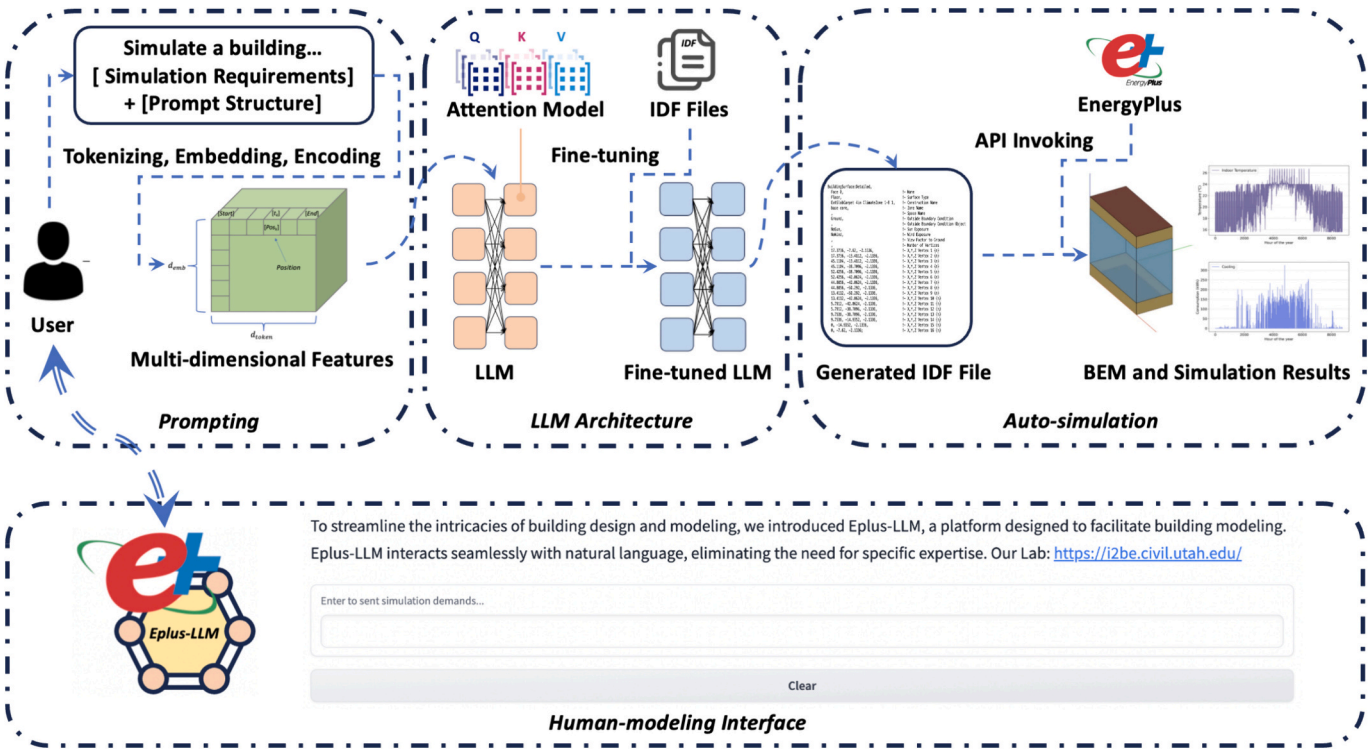


Fig. 1. The framework of our Eplus-LLM platform.

software and domain-specific knowledge (Section 2.3). Flan-T5 is an instruction model fine-tuned on top of T5 architecture (Encoders-Decoders), which has demonstrated advanced performance and superb generalization capability compared to T5 [29]. T5, known as "the transfer learning with a unified text-to-text transformer model" is distinct from previous pre-trained models like BERT or GPT. It is designed with the idea that all tasks are transformed into text-to-text problems through an encoder-decoder transformer structure, enabling it to excel across a variety of tasks. Pre-trained on a large-scale text corpus, T5 learns the natural language representation. These pre-trained weights are then fine-tuned for specific downstream application scenarios. T5 has demonstrated remarkable suitability for generation tasks; notable T5-based code generation models include codeT5 and codeT5+ [27,31]. This performance allows us to create an LLM capable of auto-building modeling. The Foundation T5 mainly have two parts, prompting of natural language to tokens and encoders-decoders with attention mechanism.

2.2.1. *Prompting of natural language for building modeling tasks*

Prompt engineering is to design instructions or inputs for a generative AI model to perform specific tasks. A prompt is typically in a format of natural language sentences that describe detailed task requirements for AI to follow [33]. It serves as a bridge for communication between users and LLMs, determining the direction and content of model outputs. Prompts can take various formats, depending on the user's intentions and task requirements. By inputting a prompt, users can guide the model to generate task results that meet their expectations.

In the building simulation field, prompts from modelers can be simple or complicated. A simple prompt can be a short modeling requirement for generating a simple building model, for example, simulate a building that is 100.00 m long, 50.00 m wide, and 8.00 m high. If complicated, more detailed information and requirements are specified. For example, simulate a building that is 100.00 m long, 50.00 m wide, and 8.00 m high. The window-to-wall ratio is 0.50. The occupancy rate is 10.00 m^2/people, the lighting level is 8.00 W/m^2, and the equipment power consumption is 5.00 W/m^2. In this case, the prompt contains clear task directions and modeling details. The choice of prompt directly affects building modeling outcomes. Users need to consider the nature of the task, the desired output, and the purpose of interacting with the model to choose the prompt appropriately. Appropriate prompts can fully utilize the linguistic capabilities of an LLM, which not only improves the accuracy of model generation, but also effectively guides the LLM to generate content satisfying needs of users. The combination of a reasonable and effective prompt design with an appropriate LLM is the key to achieve satisfactory and accurate results in LLM-based auto building modeling.

2.2.2. *Attention mechanism in LLM-based building modeling*

The self-attention mechanism is at the core of LLMs. In contrast to traditional recurrent neural network (RNN) or long-short-term memory network (LSTM), which rely on sequential processing, the self-attention enables models to assign different attention weights at different locations in the input sequence. This mechanism allows the model to focus on different information while processing the input sequence. The self-attention mechanism permits each word to adjust its importance to the context during the encoding process. It involves the dot-product the encoded input with matrix queries (Q) and keys (K) of dimension d_K, and matrix values (V) of dimension d_V. Applying a *softmax* function to obtain the weights on the values, the output of attention matrix as [18]:

$$Atteention(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_K}}\right)V$$

In building simulation scenarios, the Q, K, V matrices can be understood as representing the impact of different parameters on simulation results in our downstream tasks (i.e., building modeling). For example, the impact of heating, cooling, and electricity on buildings. To further enhance the model's capability, attention is divided into multiple heads. These multi-heads are designed to capture various channels of information in the input, especially when dealing with multiple types of inputs and prompt formats. Each head presents a distinct type of representation. The final outputs are obtained through concatenation or

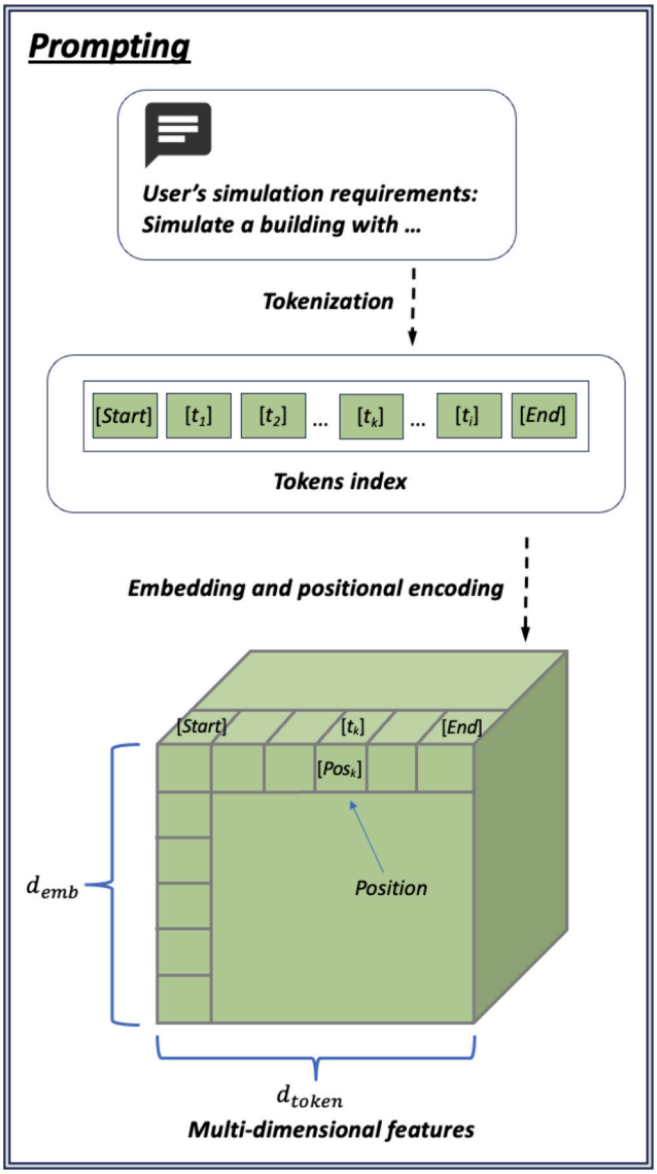


Fig. 2. An overview of Eplus-LLM platform to support auto-building simulation.

weighted averaging all heads. The output of the self-attention layer is passed into a feed-forward neural network, which includes a fully connected layer [34] and an activation function [35]. This helps to introduce the capability of thoroughly capturing nonlinear dynamics in modeling, for example, interrelationships among building system components in building modeling. Additionally, there is a residual connection [36] around each of the two sub-layers, followed by layer normalization [37], to reduce overfitting and gradient problems. These are beneficial for model training.

2.3. Fine-tuning for customizing auto-building modeling tasks

2.3.1. Generating datasets for model fine-tuning

The fine-tuning dataset, utilizing to customize the LLM for auto-building modeling, was generated through two main steps:

(1) Constructing parameters-IDF scenario pairs

The amount of data required for fine-tuning is substantial. To attain a wide range of building modeling scenarios, we utilized a Latin hypercube design [38] to sample diverse combinations of parameters, e.g., different building geometries, window-to-wall ratios, and internal load variations, and generated IDF files corresponding to different parameter combinations, i.e., parameter-to-IDF file correspondence. This step yielded a comprehensive dataset comprising diverse building model descriptions of various parameter settings, as essential to support fine-tuning processes.

(2) Constructing descriptions-IDF sentence pairs

After constructing the parameters-IDF scenario pairs, the next step is to connect these IDF files (as targets) with corresponding natural language descriptions (as prompts) for model fine-tuning. The prompt describes what the model should focus on or generate under user's simulation requirements. For example, during simulation, the prompt includes a description of the building to be simulated, containing the specifications of geometry, window details, and internal loads. The target part corresponds to the building model in IDF format for EnergyPlus. With a Python script, we translated the building model parameters to a description prompt of buildings. Four different prompt formats

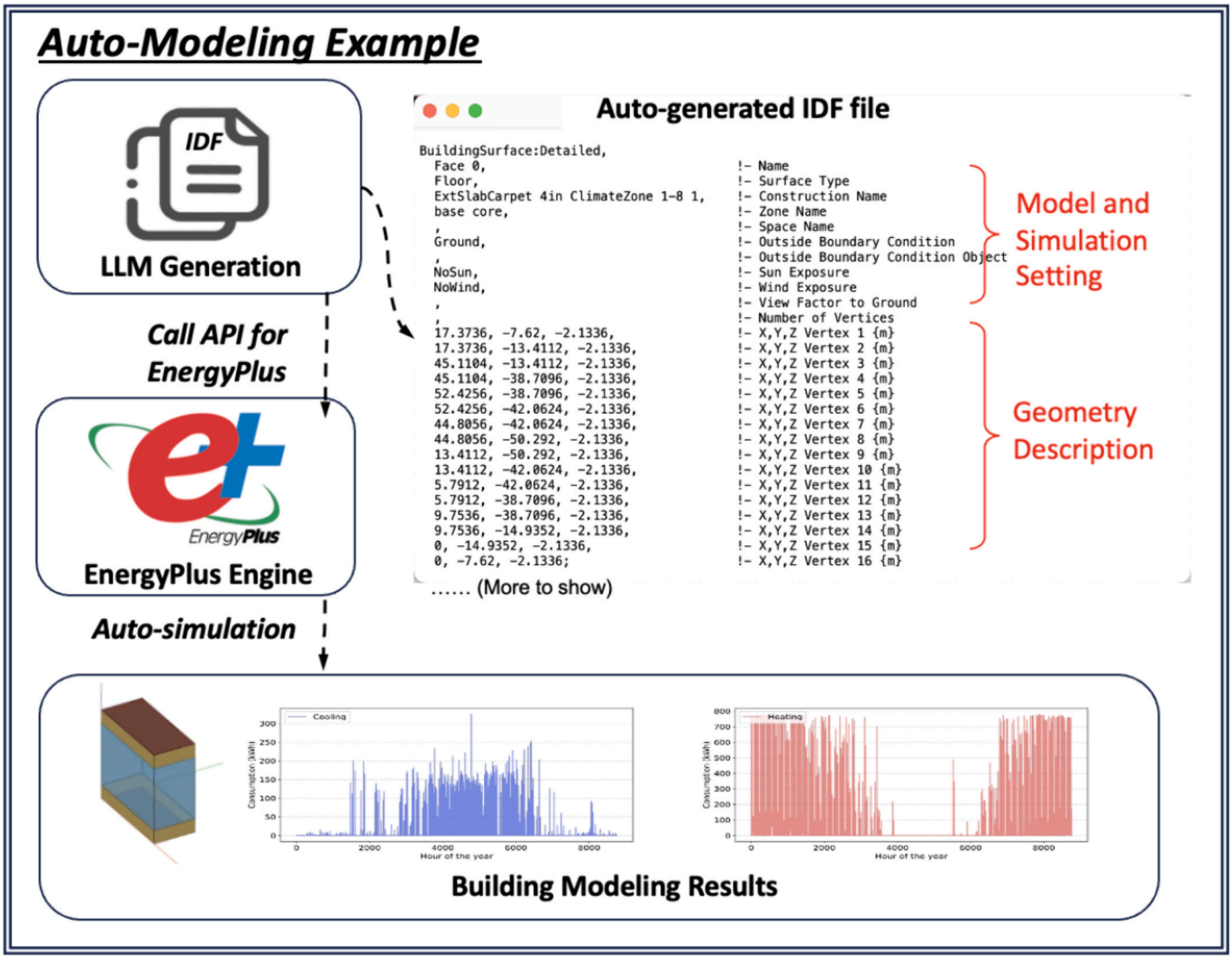


Fig. 2. (*continued*).

are designed to cover different descriptions of model requirements, resulting in a total of 70,000 pairs of prompt-targets for fine-tuning. Table 1 lists our four designed prompt formats with their simulation requirement parameters. Fig. 3 clarifies the window position within our prompt formats. The term "window height" refers to the vertical dimension of the window, while the "window sill" denotes the vertical distance between the window and the wall, influencing the window's vertical placement. The "window jamb" represents the horizontal distance between the window and the wall, affecting the window's horizontal placement. We maintain a constant value of 0.1 across all prompts for the window jamb with a window-wall ratio.

To illustrate the correlation between natural language descriptions (prompts) and IDF files, we provide a simple example (Fig. 4). The geometric configuration of building surfaces is delineated by coordinates, with interconnections among various surfaces. Due to the extent of IDF details, presenting the entire content in the paper is unfeasible. Therefore, we have extracted parts of geometry, window setting, and internal load (e.g., wall, window, and electric equipment) from the IDF and matched them with the corresponding parts in the prompt, as shown in Fig. 4.

2.3.2. *Fine-tuning and auto-simulation*

After obtaining the processed dataset for fine-tuning, these dataset (i.e., processed sentence pairs) were encoded using byte-pair encoding [39], which has a shared source target vocabulary of ~32,000 tokens. Sentence pairs were batched together by the same sequence length and each training batch contained 5 sets of sentence pairs for fine-tuning. The model weights are adjusted through back-propagation to minimize the discrepancies between the generated IDF files and the actual IDF files. Once the model achieves a satisfactory level of performance, it can be deployed with a building energy simulation engine, EnergyPlus, to generate building models and produce results (Fig. 2. b). This fine-tuned LLM can automatically generate building modeling files based on various requirements and input parameters. We fine-tuned our model on one machine with one NVIDIA A100 80G GPU. Each training step took ~1.5 s. We trained the model for a total of 32,000 steps with ~16 h.

3. Model validation and analysis

Prior to fine-tuning, we evaluated the performance of the foundation model (original Flan T5) using direct generation and one-shot learning (without fine-tuning) as a baseline for our building modeling task. Subsequently, following fine-tuning of our model, we conducted a total of 152 validation instances to verify the effectiveness and stability of the developed computing platform for auto and accurate building modeling from natural language descriptions. All auto-generated models run successfully, achieving 100% accuracy with the ground truth. In Section 3.1, we examined the performance of the foundation model using direct generation and one-shot learning. In Section 3.2, we tested the model's capability to generate corresponding outputs by inputting different

Table 1
Four types of prompt formats.

Prompt Format	Simulation requirement parameters	Prompt
1	Length, weight, height	*"Simulate a building that is xx meters long, xx meters wide, and xx meters high."*
2	Length, weight, height, WWR,	*"Simulate a building that is xx meters long, xx meters wide, and xx meters high. The window-to-wall ratio is xx."*
3	Length, weight, height, WWR, window position,	*"Simulate a building that is xx meters long, xx meters wide, and xx meters high. The window-to-wall ratio is xx, the window sill height is xx meters, the window height is xx meters, and the window jamb width is xx meters."*
4	Length, weight, height, WWR, window position, occupant, lighting, equipment	*"Simulate a building that is xx meters long, xx meters wide, and xx meters high. The window-to-wall ratio is xx, the window sill height is xx meters, the window height is xx meters, and the window jamb width is xx meters. The occupancy rate is xx m2/people, the lighting level is xx W/m2, and the equipment power consumption is xx W/m2."*

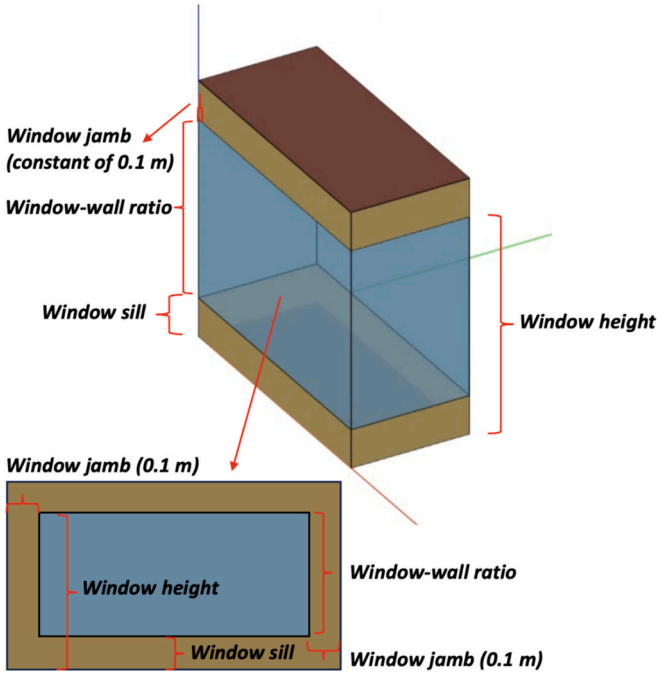


Fig. 3. Explanation of the window position in the prompts.

types of prompts randomly and manually, ensuring its seamless integration into the simulation engine and successful execution. In Section 3.3, we compared the time efficiency of the LLM-based Eplus-LLM platform with manual approaches. In Section 3.4, we assessed the robustness and anti-interference ability of the model in generation by introducing various types of noise and unseen prompts. These steps aim to guarantee the effectiveness, robustness, and versatility of the developed platform in practical applications.

3.1. Comparison of fine-tuning, prompt tuning, and direct generation

To assess performance of the fine-tuned LLM, we compare the accuracy of auto-building modeling under specified prompts between the fine-tuned LLM and two baselines without fine-tuning, i.e., direct generation (original LLM directly generates content based on the input prompt) and one-shot learning (original LLM is fed with one example of prompt-IDF pair, then instructing LLM for similar auto-modeling) [40]. Four prompts were designed for comparison of auto-model generation with-out fine-tuning and the test results are presented in Table 2. For direct generation, we observed that regardless of the prompt complexity, the output results are irrelevant to the task content. This is not surprising since Flan-T5 was not trained with any knowledge of building modeling or EnergyPlus software in the previous pre-training phase. As for the one-shot learning, it appears that the model only generates a few initial words with formatting errors, garbled content, and incomplete comprehension. Furthermore, due to Flan-T5 being trained only on sentences with a maximum length of 512 tokens during the pre-training phase, it is insufficient to produce a complete building description files with sufficient length. However, fine-tuning can overcome the limitations of the foundation model's token constraints. This further demonstrates the effectiveness of our fine-tuning process in model validation (Section 3), which enables the model to adapt to downstream tasks and achieve 100% accuracy in auto-generation of building model description.

3.2. Validation of Eplus-LLM generation

According to the four prompts in Section 2.4, we randomly generated 10 instances and manually generated 10 instances for four types of prompts in this validation process to verify the output produced by the Eplus-LLM, resulting in a total of 80 instances. The randomly and manually generated prompt examples are presented in Table 3. The validation of the accuracy is based on the match of the generated IDF file, the corresponding model, and simulation results, including indoor temperature and energy consumptions (Fig. 5). The validation results indicate that all 80 instances can be correctly invoked by the EnergyPlus engine. Additionally, they perfectly matched the ground truth models and simulation results, achieving 100% accuracy. The first prompts can only generate building models with a default WWR (i.e., WWR = 0.3). The second prompt can generate different WWR but cannot specify the position of the window. The third and fourth prompts allow for specifying the window position.

3.3. Comparison of time efficiency between manual and LLM-based modeling approaches

In this section, to evaluate the time efficiency of the LLM-based modeling method, we compared the modeling time of using the Eplus-LLM platform with two manual modeling approaches, i.e., directly using EnergyPlus IDF Editor and through OpenStudio GUI for BEM. We measured the time taken for modeling across 24 instances covered in Section 3.2, which includes 8 prompts*3 instances of building models in different complexity. Specific experiment results are shown in Table 4.

For manual modeling using EnergyPlus, users need to define model geometry with coordinates. This necessitates users to pre-calculate position of different points on building surfaces based on the design floorplan and determine window coordinates according to the WWR, taking approximately 70% of the entire modeling time. Particularly when the model precision extends to centimeters, manual calculation not only becomes burdensome but also prone to errors, significantly prolonging the modeling process. Additionally, inputting other required information such as space types and thermal zones in the software to ensure alignment with internal loads occupies ~30% of the modeling time as remaining. Overall, depending on the model complexity, constructing an instance model in EnergyPlus typically takes between 35 and 56 min. To make modeling process easier, OpenStudio is the graphical user interface that enables users to construct geometry directly through drawing, helping reduce the modeling efforts. However, this

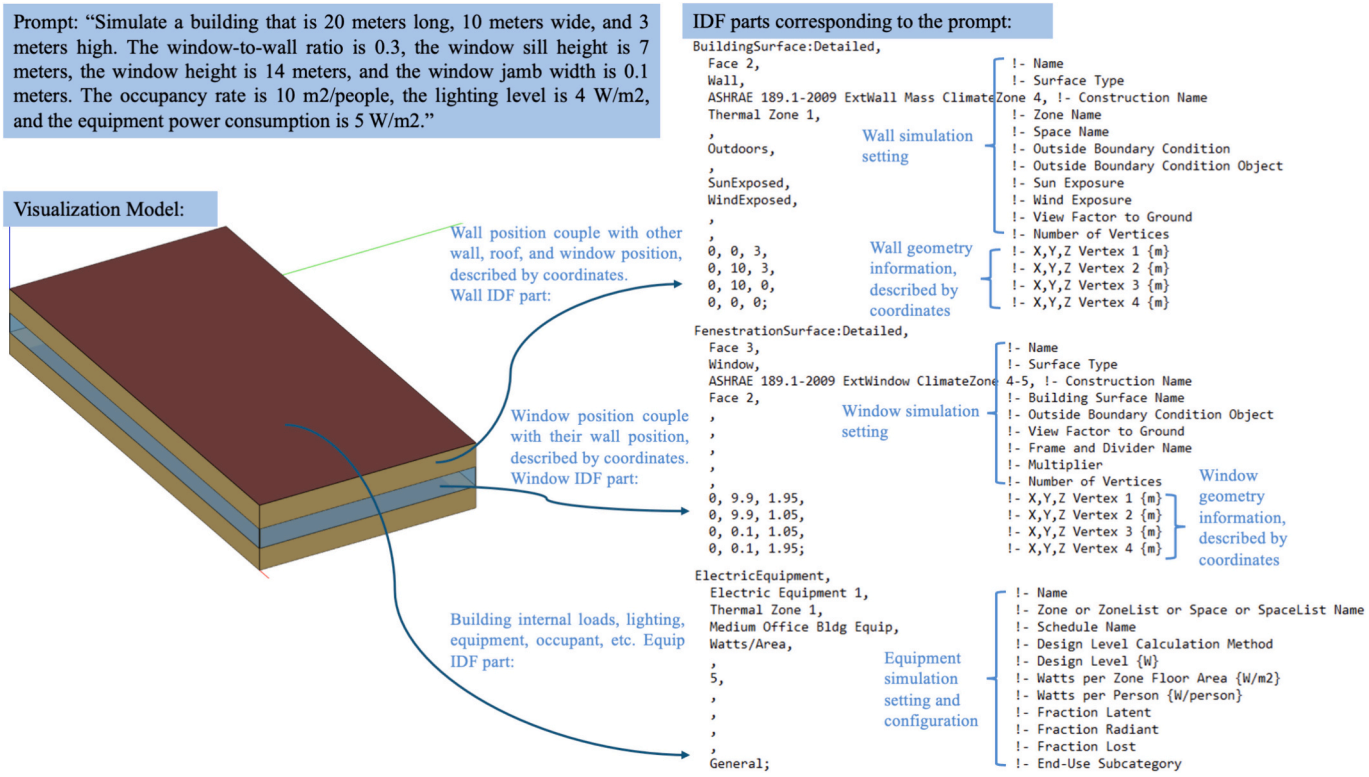


Fig. 4. An example of a specific prompt with detailed IDF.

still requires establishing a spacing grid and drawing (~30% of the time to use OpenStudio in modeling), assigning various information such as stories, thermal zones, space types, constructions, and windows in the geometry interface tab (~30% of the time), and pre-setting space types, thermal zones, and internal loads (~40% of the time). Constructing an instance in OpenStudio takes from 11 to 20 min for different cases. Lastly, establishing a building model in the Eplus-LLM platform only requires users to write a natural language prompt in the platform interface, after which Eplus-LLM automatically generates the corresponding building model in IDF format for modeling (Fig. 6). This process takes approximately 1 min, including about 30 s to write prompts and another 30s for LLM generation.

In conclusion, utilizing Eplus-LLM for auto-modeling through natural language can significantly reduce the modeling efforts by over 95% while ensuring modeling accuracy. Moreover, for beginners, manual modeling of buildings using software programs has a steeper learning curve, including initial model establishment and troubleshooting. It is foreseeable that as building models grow in complexity, the efficiency gains from using LLM-based auto-modeling methods is expected to become more pronounced.

3.4. Robustness evaluation of Eplus-LLM in auto-modeling

In order to evaluate the robustness of our prompts, we introduced four types of noises for every prompt format: user's tone styles, spelling mistakes, omitted words, and extra words. We conducted two tests for each prompt under every noise condition, resulting in a total of 4*4*2 = 32 instances. Table 5 lists the examples of different noises. The test results indicate that our model exhibits high robustness, flexibility, and resistance to the noise of user commands in practice. Even under varying degrees of noise influence, it can still generate results that meet user requirements with 100% accuracy in generation.

In addition, during our testing, we observed that Eplus-LLM exhibits a certain degree of self-learning, specifically, the ability to process previously unseen information (Table 6). Through fine-tuning in training, Eplus-LLM acquired new concise expressions for prompts. These new prompts were not included in the fine-tuning datasets. For Example, for the unseen Prompt 4, Eplus-LLM learned the ability to identify window locations. Consequently, modelers only need to specify the window sill and window height, eliminating the need to additionally specify the WWR and window jamb, as required previously. For the unseen Prompt 1, 2 and 3, Eplus-LLM can adapt to new prompt structures by learning the interrelationships between various prompts (Prompt 3 and 4). This illustrates the versatility, adaptability, and scalability of Eplus-LLM. We conducted 10 tests for each unseen prompt.

4. Discussion and further work

4.1. Model structure and mechanism for LLM-based auto-building modeling

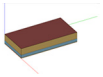
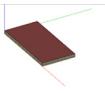
While current LLMs employ various model structures, e.g., decoders-only, encoders-only, or decoders-encoders for diverse tasks, there is no consensus on the optimal model structure for the task of IDF generation. Although PTMs that rely solely on decoders or encoders, such as BERT and GPT, have achieved some success, they are suboptimal in comprehending tasks or processing code due to the inherent constraints and limited flexibility associated with decoder or encoder model structures [27,31]. The auto building modeling task to generate model description in length with specific required structure (as the advantage of decoders). Also, the utilized LLM needs to thoroughly comprehend the user's modeling description (as the advantage of encoders), ensuring a precise understanding and reasoning of the corresponding IDF details, as even minor errors can lead to simulation failure. The encoders-decoders model proves to be particularly advantageous in tasks requiring intricate mapping and capturing of element relationships between input and output sequences; that is why we chose T5 with encoders-decoders structure as the foundation model for our auto-modeling.

While the working mechanism of LLMs remains a black box, posing challenges in understanding how distinct models excel in various

Table 2
Baseline results without fine-tuning.

Prompt format	Input prompt	Output (Direct generation)	Output (One-shot learning)
1	*"Generate IDF using EnergyPlus: Simulate a building that is 20.00 meters long, 40.00 meters wide, and 4.00 meters high."*	"It will use a total of 0.002 kilowatts."	"BuildingSurface: Detailed, face 0, floor, extslabcarpet 4in"
2	*"Generate IDF using EnergyPlus: Simulate a building that is 20.00 meters long, 40.00 meters wide, and 4.00 meters high. The window-to-wall ratio is 0.3."*	"The total number of windows is."	"BuildingSurface: Detailed, face 0, floor, extslabcarpet 4in"
3	*"Generate IDF using EnergyPlus: Simulate a building that is 150.00 meters long, 50.00 meters wide, and 20.00 meters high. The window-to-wall ratio is 0.60, the windowsill height is 4.00 meters, the window height is 16.00 meters, and the window jamb width is 0.01 meters."*	"The total IDF for the building is."	"BuildingSurface: Detailed, face 0, floor, extslabcarpet 4in"
4	*"Generate IDF using EnergyPlus: Simulate a building that is 150.00 meters long, 50.00 meters wide, and 20.00 meters high. The window-to-wall ratio is 0.60, the windowsill height is 4.00 meters, the window height is 16.00 meters, and the window jamb width is 0.01 meters. The occupancy rate is 5.00 m2/people, the lighting level is 10.00 W/m2, and the equipment power consumption is 20.00 W/m2."*	"The building uses a total of.01 m3 of energy per square meter."	"BuildingSurface: Detailed, face 0, floor, extslabcarpet 4in"

Table 3
Examples of model generated for random and manual testing.

Random input		Manual input	
Generated model	Prompt	Generated model	Prompt
	*"Simulate a building that is 172.70 meters long, 337.90 meters wide, and 56.60 meters high."*		*"Simulate a building that is 20.00 meters long, 20.00 meters wide, and 5.00 meters high."*
	*"Simulate a building that is 44.60 meters long, 279.20 meters wide, and 90.50 meters high. The window-to-wall ratio is 0.50."*		*"Simulate a building that is 80.00 meters long, 50.00 meters wide, and 10.00 meters high. The window-to-wall ratio is 0.30."*
	*"Simulate a building that is 121.90 meters long, 44.50 meters wide, and 42.10 meters high. The window-to-wall ratio is 0.40, the window sill height is 12.63 meters, the window height is 29.47 meters, and the window jamb width is 0.01 meters."*		*"Simulate a building that is 200.00 meters long, 100.00 meters wide, and 10.00 meters high. The window-to-wall ratio is 0.40, the windowsill height is 3.00 meters, the window height is 7.00 meters, and the window jamb width is 0.01 meters."*
	*"Simulate a building that is 390.00 meters long, 217.90 meters wide, and 35.30 meters high. The window-to-wall ratio is 0.30, the window sill height is 12.35 meters, the window height is 22.94 meters, and the window jamb width is 0.01 meters. The occupancy rate is 25.60 m2/people, the lighting level is 10.50 W/m2, and the equipment power consumption is 5.60 W/m2."*		*"Simulate a building that is 150.00 meters long, 50.00 meters wide, and 20.00 meters high. The window-to-wall ratio is 0.60, the windowsill height is 4.00 meters, the window height is 16.00 meters, and the window jamb width is 0.01 meters. The occupancy rate is 5.00 m2/people, the lighting level is 10.00 W/m2, and the equipment power consumption is 20.00 W/m2."*

downstream tasks and the rationale behind specific outcomes, one certainty prevails: attention is the key to the success of LLMs [41]. In this study, the Eplus-LLM utilizes attention to focus on different parts of the user simulation command, capture complex relationships in natural language descriptions of building models, and map these descriptions to building model files. Improvements of attention mechanism, as listed below, are expected to further boost the LLM applications in auto-building modeling:

(1) Capability to Handle Longer Context in Attention

Current LLMs typically support context lengths of 2048 or 4096 tokens. Longer text content acceptable by LLM indicates its increasing capability to support modeling of more complex buildings and enhance human-AI interaction in the auto modeling process. This further increases the flexibility of auto-building modeling. Block-sparse FlashAttention [42], attention sink [43], and related methods expected to serve as potential techniques for scaling up the acceptable tokens.

(2) Efficient Attention and Model Architecture

Modifying the conventional attention architecture (e.g., through the addition or combination of diverse layers) as exemplified by approaches such as SOLAR [44], MOE [45], and Mistral [46] is anticipated to yield more efficient models. These modifications aim to improve computational efficiency, ultimately facilitating an efficient and precise inference process.

4.2. Prompts and instructions to boost LLMs

In this study, we chose Flan-T5 (encoders-decoders structure) as the foundation model for our Eplus-LLM platform, achieving satisfactory performance in auto-model generation. Besides the model structure, the inherent scaling prompt and instruction process used to produce Flan-T5 is also a key factor contributing to its success. By scaling the number of tasks, scaling the model size, and finetuning on chain-of-thought data, the performance of the model is greatly enhanced, enabling it to attain strong capability even compared to larger LLMs. This makes Flan-T5 more suitable for various downstream tasks that require fine-tuning, such as auto-building modeling in this study.

With limited computational resources, using high-quality prompt and instruction for model fine tuning is an important approach to improve model performance [32]. By skillfully designing prompts and instructions, we can improve model performance by directing the model to focus on important information in the building design and simulation task. In our study, we carefully designed four types of prompts to enhance the flexibility, versatility, and robustness of the fine-tuned LLM

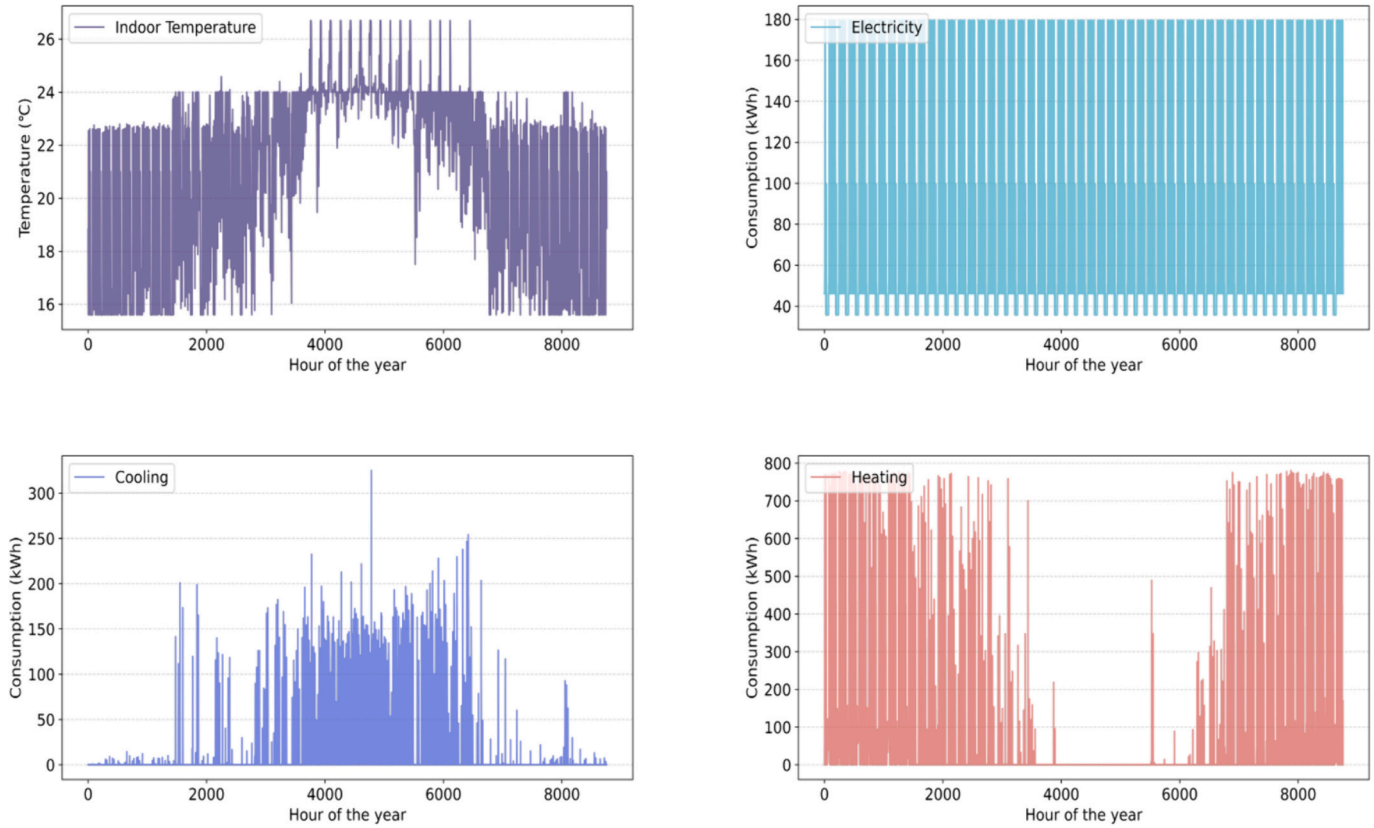


Fig. 5. Simulation results, including indoor temperature, heating, cooling, and electricity consumptions.

Table 4
Estimated time range of manual and LLM-based modeling.

	EnergyPlus	OpenStudio	Eplus-LLM
Modeling time	35–56 min	11–20 min	40–74 s

in auto-building modeling. Prompts usually contain information about the context of the task, as crucial to help the LLM comprehend the requirements of the task. By designing the prompt appropriately, models can successfully focus on key information in the input sequence (e.g., information about the building geometry and internal loads). On the other hand, instructions with more specific and detailed input guidance can tell the model how to handle the task. Instruction can be designed to emphasize specific patterns, regularities, or features that help the model learn key information about the task. For example, in our task, instruction is set as "simulation" to tell the model to perform a simulation task. Providing explicit guidance helps the model to fully utilize the pretrain data to make the model fine-tuning converge faster, learn key aspects of the task, and reduce its sensitivity to noise.

Notably, by flexibly combining and designing prompts and instructions according to the simulation requirements, models become better adaptable to a variety of unseen prompt types rather than being limited to prompts from the training dataset (initial prompts). This flexibility is essential to improve the generality and robustness of the developed auto-modeling platform. For building design and simulation as a downstream task, designing excellent prompts and instructions becomes an effective means to improve model performance and

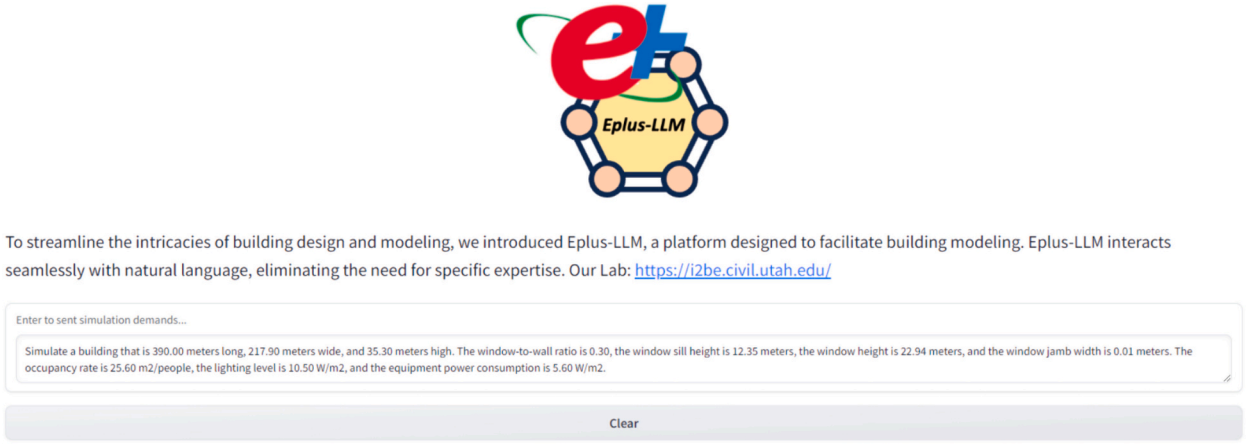


Fig. 6. Input a prompt to Eplus-LLM platform for automated modeling.

Table 5
Examples of different noised prompts.

Format	Initial prompt	Different Tone styles	Spelling mistakes	Omitted words	Extra words
1	*"Simulate a building that is 30.00 meters long, 40.00 meters wide, and 3.50 meters high."*	*"Create a model representing a building with dimensions of 30.00 meters in length, 40.00 meters in width, and a height of 3.50 meters."*	*"Simulatte a bilding that is 30.00 meteres long, 40.00 meter wide, and 3.50 meters hi."*	*"building 30.00 meters long, 40.00 meters wide, 3.50 meters high."*	*"Simulate a giraffe building that is 30.00 marshmallow meters long, 40.00 kazoo meters wide, and 3.50 trampoline meters high."*
2	*"Simulate a building that is 33.30 meters long, 455.50 meters wide, and 8.80 meters high. The window-to-wall ratio is 0.33."*	*"Generate a simulation for a structure with dimensions of 33.30 meters in length, 455.50 meters in width, and a height of 8.80 meters. The window-to-wall ratio is set at 0.33."*	*"Simulate aa buidling that is 33.30 meters long, 455.50 metrs wide, and 8.80 meters high. The window-to-wll ratio is 0.33."*	*"building 33.30 meters long, 455.50 meters wide, 8.80 meters high. The window wall is 0.33."*	*"Simulate a pineapple trampoline giraffe building that is 33.30 kazoo meters long, 455.50 marshmallow xylophone meters wide, and 8.80 sombrero meters high. The bubblegum window-to-pickle wall ratio is 0.33."*
3	*"Simulate a building that is 36.50 meters long, 326.00 meters wide, and 55.50 meters high. The window-to-wall ratio is 0.44, the window sill height is 8.72 meters, the window height is 49.39 meters, and the window jamb width is 0.01 meters."*	*"Develop a simulation for a structure with dimensions of 36.50 meters in length, 326.00 meters in width, and a height of 55.50 meters. Integrate a window-to-wall ratio of 0.44, with a window sill height of 8.72 meters, a window height of 49.39 meters, and a window jamb width of 0.01 meters."*	*"Simullate a building that is 36.50 meters long, 326.00 meters wide, and 55.50 meters hiegh. The window-to-wal ratio is 0.44, the window sill hieght is 8.72 meters, the window hieght is 49.39 metrs, and the window jamb width is 0.01 meters."*	*"building 36.50 meters long, 326.00 meters wide, 55.50 meters high. Window-wall is 0.44, the window sill is 8.72 meters, the window height is 49.39 meters, window jamb is 0.01 meters."*	*"Simulate an extraordinarily purple building that is 36.50 extremely meters long, 326.00 peculiarly meters wide, and 55.50 fascinatingly meters high. The window-to-wall ratio is 0.44, the astonishing window sill height is 8.72 delightfully meters, the window height is 49.39 mysteriously meters, and the window jamb width is 0.01 excessively meters."*
4	*"Simulate a building that is 83.50 meters long, 55.00 meters wide, and 16.00 meters high. The window-to-wall ratio is 0.35, the window sill height is 3.20 meters, the window height is 12.80 meters, and the window jamb width is 0.01 meters. The occupancy rate is 5.50 m2/ people, the lighting level is 18.00 W/m2, and the equipment power consumption is 10.00 W/m2."*	*"Create a model representing a building measuring 83.50 meters in length, 55.00 meters in width, and 16.00 meters in height. Incorporate specific window features like a window-to-wall ratio of 0.35, a window sill height of 3.20 meters, a window height of 12.80 meters, and a window jamb width of 0.01 meters. Take into account an occupancy rate of 5.50 square meters per person, a lighting level of 18.00 watts per square meter, and equipment power consumption of 10.00 watts per square meter."*	*"Simullate a bilding that is 83.50 meters long, 55.00 metrs wide, and 16.00 meters hiegh. The window-to-wll ratio is 0.35, the window sill hight is 3.20 metrs, the window hieght is 12.80 meters, and the window jamb widht is 0.01 metrs. The occupancy rate is 5.50 m2/people, the lightnng level is 18.00 W/m2, and the equipmnt power consuption is 10.00 W/m2"*	*"Building 83.50 meters long, 55.00 meters wide, 16.00 meters high. The window-to-wall 0.35, the sill height is 3.20 meters, the window height is 12.80 meters, and the window jamb is 0.01 meters. Occupancy 5.50 m2/people, lighting 18.00 W/m2, and equipment is 10.00 W/m2."*	*"Simulate a futuristic building that is 83.50 meters long, 55.00 meters wide, and 16.00 meters high. The window-to-wall ratio, a key element of its design, is meticulously set at 0.35. The window sill height gracefully extends to 3.20 meters, while the soaring window height reaches an impressive 12.80 meters, with a sleek window jamb width of 0.01 meters. The occupancy rate, carefully calculated, stands at 5.50 m2/people, ensuring a harmonious balance within its space. Illuminating the surroundings, the lighting level radiates at 18.00 W/m2, creating a vibrant atmosphere. Furthermore, the building's efficiency is evident as the equipment power consumption is maintained at a sustainable 10.00 W/m2."*

diversity. The key to this design lies in a profound understanding of the relationship between simulation tasks and model predictions, ensuring that the prompt and instruction boost the model's reasoning ability.

4.3. Empowering the entire building life cycle with LLM

This study represents a pioneering effort to automate building model creation directly from natural language descriptions provided by modelers. This is especially useful to assist architects and modelers to assess building design in the conceptual design stage through quick and automated generation of building models. Furthermore, the LLMs have the potential to support building development in its full lifecycle (such as construction and operation), not limited to the model design phase in this paper. For example, during the construction phase, LLM can facilitate interaction between construction personnel and AI to provide real-time decision support, optimize construction schedules, and predict potential risks. In the operation phase, LLMs-based methods will facilitate building control or community-level control through real-time human-AI interaction, maximizing occupant comfort, and the potential for building energy efficiency and decarbonization. Overall, the application of LLMs across the entire building lifecycle has the potential to further improve efficiency, reduce costs, enhance decision support, and steer the building sector toward a smart and sustainable future.

5. Conclusion

In this study, we introduce and demonstrate the successful development of Eplus-LLM as the first LLM-based automated building modeling platform. The platform provides a user-friendly human-AI interface, allowing users to conduct building simulation directly from natural language, without requiring in-depth knowledge of complex building science and simulation software. The Eplus-LLM understands human language through tokenization and embedding techniques, and overcomes the complexity of auto-file generation for building modeling through self-attention. Then, it can output building models and simulation results by invoking EnergyPlus as the simulation engine. This innovation greatly reduces the modeling efforts and dependency of software in modeling. In order to meet the simulation needs of different users, we also designed four different prompts to make Eplus-LLM more adaptable and versatile.

To validate the effectiveness of our developed platform in support modeling practice, we presented a total of 152 test cases. The validation

Table 6
Unseen types of prompt format.

	Format	Initial simulation requirement parameters	Simulation requirement parameters	Unseen prompt
New prompt	1	Length, weight, height	Length, weight, height + window position	*"Simulate a building that is xx meters long, xx meters wide, and xx meters high. The window sill height is xx meters, the window height is xx meters"*
	2	Length, weight, height	Length, weight, height + lighting, occupancy, equipment	*"Simulate a building that is xx meters long, xx meters wide, and xx meters high. The occupancy rate is xx m2/people, the lighting level is xx W/m2, and the equipment power consumption is xx W/m2."*
	3	Length, weight, height	Length, weight, height + WWR, lighting, occupancy, equipment	*"Simulate a building that is xx meters long, xx meters wide, and xx meters high. The window-to-wall ratio is xx. The occupancy rate is xx m2/people, the lighting level is xx W/m2, and the equipment power consumption is xx W/m2."*
Concise prompt	4	Length, weight, height, WWR, window position, occupant, lighting, equipment	Length, weight, height, window position, lighting, occupancy, equipment	*"Simulate a building that is xx meters long, xx meters wide, and xx meters high. The window sill height is xx meters, the window height is xx meters. The occupancy rate is xx m2/people, the lighting level is xx W/m2, and the equipment power consumption is xx W/m2."*

results demonstrate that Eplus-LLM not only achieved 95% time efficiency and 100% accuracy aligning with manual expert modeling, but also exhibited robustness and adaptability to various noises and unseen prompts in application. This robustness underscores the effectiveness of our approach as basis for further applications, such as UBEM and calibration. In the discussion section, we introduce and explore the directions for model selection and attention mechanism in LLMs. Additionally, we discuss the strategies to boost the LLM performance through prompts and instructions. Lastly, we project the future impacts of generative AI with LLMs to support building development for the entire life cycle of buildings.

As to limitations, our developed platform is currently subject to objective conditions (e.g., GPUs, training time, and LLM performance) and is only able to handle relatively simple modeling cases under regular settings (e.g., rectangular building shape with WWR) as representative examples. In practical building modeling, various systems with complex geometries, different zones, and schedules are expected, requiring a further enhancement of the developed platform for automated modeling. Moreover, our platform has not been able to process interdependencies such as "placing this window in the xxx position of the south wall" since it requires the LLM to obtain more semantic information and make corresponding changes.

Future research directions will include further exploring and applying LLMs to enhance their potential in various real-world applications. We advocate for investigations on more flexible and complex modeling scenarios, such as buildings with complex zoning or prompts for auto-modeling containing semantic description of buildings (e.g., south/north wall). These efforts will further advance the development of auto building modeling platform, providing more powerful (AI) tools for future building design and intelligent building management.

CRediT authorship contribution statement

Gang Jiang: Writing – original draft, Validation, Methodology, Investigation, Conceptualization. **Zhihao Ma:** Validation, Software, Methodology. **Liang Zhang:** Writing – review & editing, Software, Formal analysis. **Jianli Chen:** Writing – review & editing, Writing – original draft, Supervision, Resources, Methodology, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The authors are unable or have chosen not to specify which data has been used.

Acknowledgement

We would like to acknowledge the funding provided by the US National Science Foundation (NSF). Award title: Elements: A Convergent Physics-based and Data-driven Computing Platform for Building Modeling (#2311685).

References

[1] IEA – International Energy Agency. IEA. https://www.iea.org; 2024 (n.d. accessed January 14, 2024).
[2] Ding Y, Han S, Tian Z, Yao J, Chen W, Zhang Q. Review on occupancy detection and prediction in building simulation. Build Simul 2022;15:333–56. https://doi.org/10.1007/s12273-021-0813-8.
[3] Zhou X, Liu R, Tian S, Shen X, Yang X, An J, et al. A review of validation methods for building energy modeling programs. Build Simul 2023;16:2027–47. https://doi.org/10.1007/s12273-023-1050-0.
[4] Al-Homoud MS. Computer-aided building energy analysis techniques. Build Environ 2001;36:421–33. https://doi.org/10.1016/S0360-1323(00)00026-3.
[5] Ansys | Engineering Simulation Software. n.d. https://www.ansys.com/. [Accessed 14 January 2024].
[6] Dymola. Dassault Systèmes. https://www.3ds.com/products/catia/dymola; 2023 (accessed January 14, 2024).
[7] Welcome | TRNSYS : Transient System Simulation Tool. n.d. https://www.trnsys.com/. [Accessed 14 January 2024].
[8] Yan D, Xia J, Tang W, Song F, Zhang X, Jiang Y. DeST — An integrated building simulation toolkit part I: fundamentals. Build Simul 2008;1:95–110. https://doi.org/10.1007/s12273-008-8118-8.
[9] EnergyPlus. n.d. https://energyplus.net/. [Accessed 14 January 2024].
[10] 14:00–17:00. ISO 13790:2008. ISO; 2024. n.d. https://www.iso.org/standard/41974.html (accessed January 14, 2024)
[11] VDI 6007 Blatt 1 - Calculation of transient thermal response of rooms and buildings - Modelling of rooms. 2015.
[12] DesignBuilder Software Ltd - Home. n.d. https://designbuilder.co.uk/. [Accessed 14 January 2024].
[13] OpenStudio. n.d. https://openstudio.net/; 2024 (accessed January 14, 2024)

[14] eppy. PyPI. https://pypi.org/project/eppy/; 2022 (accessed January 14, 2024).
[15] Wang H, Li J, Wu H, Hovy E, Sun Y. Pre-trained language models and their applications. Engineering 2023;25:51–65. https://doi.org/10.1016/j.eng.2022.04.024.
[16] Peters ME, Neumann M, Iyyer M, Gardner M, Clark C, Lee K, et al. Deep contextualized word representations. 2018. https://doi.org/10.48550/arXiv.1802.05365.
[17] Radford A, Narasimhan K, Salimans T, Sutskever I. Improving language understanding by generative pre-training. 2024. n.d.
[18] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. 2023. https://doi.org/10.48550/arXiv.1706.03762.
[19] Devlin J, Chang M-W, Lee K, Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 2019. https://doi.org/10.48550/arXiv.1810.04805.
[20] Raffel C, Shazeer N, Roberts A, Lee K, Narang S, Matena M, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. 2023. https://doi.org/10.48550/arXiv.1910.10683.
[21] Chowdhery A, Narang S, Devlin J, Bosma M, Mishra G, Roberts A, et al. PaLM: Scaling Language Modeling with Pathways. 2022. https://doi.org/10.48550/arXiv.2204.02311.
[22] Touvron H, Lavril T, Izacard G, Martinet X, Lachaux M-A, Lacroix T, et al. LLaMA: Open and Efficient Foundation Language Models. 2023. https://doi.org/10.48550/arXiv.2302.13971.
[23] Donahue J, Jia Y, Vinyals O, Hoffman J, Zhang N, Tzeng E, et al. DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. 2013. https://doi.org/10.48550/arXiv.1310.1531.
[24] Schneider S, Baevski A, Collobert R, Auli M. wav2vec: Unsupervised Pre-training for Speech Recognition. 2019. https://doi.org/10.48550/arXiv.1904.05862.
[25] Grisoni F. Chemical language models for de novo drug design: challenges and opportunities. Curr Opin Struct Biol 2023;79:102527. https://doi.org/10.1016/j.sbi.2023.102527.
[26] Zhang Y, Zhang F, Chen N. Migratable urban street scene sensing method based on vision language pre-trained model. Int J Appl Earth Obs Geoinf 2022;113:102989. https://doi.org/10.1016/j.jag.2022.102989.
[27] Wang Y, Wang W, Joty S, Hoi SCH. CodeT5: Identifier-aware Unified Pre-trained Encoder-Decoder Models for Code Understanding and Generation. 2021. https://doi.org/10.48550/arXiv.2109.00859.
[28] Ouyang L, Wu J, Jiang X, Almeida D, Wainwright CL, Mishkin P, et al. Training language models to follow instructions with human feedback. 2022. https://doi.org/10.48550/arXiv.2203.02155.
[29] Chung HW, Hou L, Longpre S, Zoph B, Tay Y, Fedus W, et al. Scaling instruction-finetuned language models. 2022. https://doi.org/10.48550/arXiv.2210.11416.
[30] Pavlyshenko BM. Financial news analytics using fine-tuned Llama 2 GPT Model. 2023. https://doi.org/10.48550/arXiv.2308.13032.
[31] Wang Y, Le H, Gotmare AD, Bui NDQ, Li J, Hoi SCH. CodeT5+: open code large language models for code understanding and generation. 2023. https://doi.org/10.48550/arXiv.2305.07922.
[32] Stanford CRFM. n.d. https://crfm.stanford.edu/2023/03/13/alpaca.html. [Accessed 14 January 2024].
[33] Radford A, Wu J, Child R, Luan D, Amodei D, Sutskever I. Language models are unsupervised multitask learners. 2024. n.d.
[34] Basha SHS, Dubey SR, Pulabaigari V, Mukherjee S. Impact of fully connected layers on performance of convolutional neural networks for image classification. Neurocomputing 2020;378:112–9. https://doi.org/10.1016/j.neucom.2019.10.008.
[35] Agarap AF. Deep Learning using Rectified Linear Units (ReLU). 2019. https://doi.org/10.48550/arXiv.1803.08375.
[36] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. 2015. https://doi.org/10.48550/arXiv.1512.03385.
[37] Ba JL, Kiros JR, Hinton GE. Layer normalization. 2016. https://doi.org/10.48550/arXiv.1607.06450.
[38] Petelet M, Iooss B, Asserin O, Loredo A. Latin hypercube sampling with inequality constraints. 2010. https://doi.org/10.48550/arXiv.0909.0329.
[39] Britz D, Goldie A, Luong M-T, Le Q. Massive exploration of neural machine translation architectures. 2017. https://doi.org/10.48550/arXiv.1703.03906.
[40] Brown TB, Mann B, Ryder N, Subbiah M, Kaplan J, Dhariwal P, et al. Language models are few-shot learners. arXivOrg, https://arxiv.org/abs/2005.14165v4; 2020. [Accessed 31 March 2024].
[41] Guo M-H, Xu T-X, Liu J-J, Liu Z-N, Jiang P-T, Mu T-J, et al. Attention mechanisms in computer vision: a survey. Comput Vis Media 2022;8:331–68. https://doi.org/10.1007/s41095-022-0271-y.
[42] Dao T, Fu DY, Ermon S, Rudra A, Ré C. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. 2022. https://doi.org/10.48550/arXiv.2205.14135.
[43] Xiao G, Tian Y, Chen B, Han S, Lewis M. Efficient streaming language models with attention sinks. 2023. https://doi.org/10.48550/arXiv.2309.17453.
[44] Kim D, Park C, Kim S, Lee W, Song W, Kim Y, et al. SOLAR 10.7B: Scaling large language models with simple yet effective depth up-scaling. 2023. https://doi.org/10.48550/arXiv.2312.15166.
[45] Shen S, Hou L, Zhou Y, Du N, Longpre S, Wei J, et al. Mixture-of-experts meets instruction tuning: a winning combination for large language models. 2023. https://doi.org/10.48550/arXiv.2305.14705.
[46] Jiang AQ, Sablayrolles A, Mensch A, Bamford C, Chaplot DS, Casas D de las, et al. Mistral 7B. 2023. https://doi.org/10.48550/arXiv.2310.06825.