Automated urban planning aware spatial hierarchies and human instructions

Dongjie Wang¹ · Kunpeng Liu² · Yanyong Huang³ · Leilei Sun⁴ · Bowen Du⁴ · Yanjie Fu¹

Received: 28 February 2022 / Revised: 7 November 2022 / Accepted: 13 November 2022 / Published online: 29 November 2022
© The Author(s), under exclusive licence to Springer-Verlag London Ltd., part of Springer Nature 2022

Abstract
Traditional urban planning demands urban experts to spend much time producing an optimal urban plan under many architectural constraints. The remarkable imaginative ability of deep generative learning provides hope for renovating this domain. Existing works are constrained by: (1) neglecting human requirements; (2) omitting spatial hierarchies, and (3) lacking urban plan samples. We propose a novel, deep human-instructed urban planner to fill these gaps and implement two practical frameworks. In the preliminary version, we formulate the task into an encoder–decoder paradigm. The encoder is to learn the information distribution of surrounding contexts, human instructions, and land-use configuration. The decoder is to reconstruct the land-use configuration and the associated urban functional zones. Although it has achieved good results, the generation performance is still unstable due to the complex optimization directions of the decoder. Thus, we propose a cascading deep generative adversarial network (GAN) in this paper, inspired by the workflow of urban experts. The first GAN
is to build urban functional zones based on human instructions and surrounding contexts. The second GAN will produce the land-use configuration by considering the built urban functional zones. Finally, we conducted extensive experiments and case studies to validate the effectiveness and superiority of our work.

**Keywords** Urban planning · Representation learning · Deep variational autoencoder · Deep generative adversarial network

**1 Introduction**

Urban planning is the study of how urban resources (like parks and schools) are used, which affects how a place looks and how it will grow in the future. Conceptually, urban planning experts plan a multi-year project to renovate a target area in order to develop a promising community, which includes exploration, feasibility studies, master planning, zone planning, land use planning, and implementation. If the created urban plan is inadequate, not only these years but also billions of dollars will be wasted, according to the study in [1]. Therefore, it has drawn a significant amount of attention to how to leverage advanced Artificial Intelligence (AI) technologies for relieving urban experts’ onerous workloads and assisting them in producing unbiased and inspiring urban plans.

Various AI technologies have exhibited the exciting potential to unlock the possibilities of smart urban planning to optimize urban infrastructures and look into the cities of the future worldwide. For example, in the USA, Sidewalks, a Google subsidiary, attempted to improve Toronto’s sustainability, economic opportunity, and new mobility by using AI [2]. Similarly, in China, Baidu signed a strategic collaboration agreement with Hebei Province to develop Xiongan New Area into a smart city via AI techniques [3]. These suggest that developing AI-assisted generative urban design has become a hot direction for revolutionizing traditional urban planning (Fig. 1).

Even though much efforts have been devoted to AI for automated urban planning [4–7], these AI-based planning are limited in addressing the following issues. Issue 1: human instructions. Without regulations, deep generative modeling could generate non-usable land-use configurations that lacks a precise planning objective. The most reliable regulations are human instructions, which are in a textual form of empirical knowledge of domain experts to represent natural intelligence [8]. It is a well-known challenge to develop a human-machine interface so machines can digest and convert human instructions into knowledge to regularize model structures. Issue 2: zone-grid spatial hierarchies. For instance, the study in [7] proposed a GAN-based framework to generate urban plan based on geospatial contexts. However, this method ignores the spatial hierarchies between zone-level planning (i.e., urban functional zones) and grid-level planning (i.e., land-use configuration). Mixed functional zoning intends to regulate built environments to ensure the complementarity and sustainability of the grid-level planning. Issue 3: data sparsity. To generate ideal urban plans, deeply generative urban planners need to learn from numerous urban plan samples. However, realistic urban plan samples are typically sparse and even imperfect in most cases.

In the preliminary work [9], we propose a deep conditional variational encoder–decoder framework to address the aforementioned limitations. Specifically, the encoder is to learn the associations between the planning requirements (e.g., human guidance and surrounding contexts) and the corresponding land-use configurations. The decoder is to generate ideal urban functional zones and land-use configurations based on planning requirements and the
Automated urban planning aware spatial hierarchies... 1339

Fig. 1 Our deep urban planner generates land-use configurations based on the human instructions and numerous planning requirements

learned association relations. This decoding strategy ensures the capture of spatial hierarchies between spatial grids and functional zones. In the meantime, to deal with the lack of data, we propose a variational Gaussian mechanism and use it in the encoder module to increase the variety of inputs.

Even though the preliminary study has yielded positive results, the generation performance of the urban planner is unstable. Spatial hierarchies are captured by simultaneously rebuilding urban functional zones and land-use patterns, leading to ambiguous optimization directions and unstable generation performance. In order to address this issue, we evaluate the workflow of conventional urban experts. As shown in Fig. 2, these experts always first design a rough sketch for a geographical area, then fill in concrete building elements to make it workable. This workflow can establish a clear objective for each step and enhance the rationality of the generated configurations.

Thus, in this journal version, we formulate automated urban planning as a two-step generation task to mimic such workflow. Specifically, we implement a cascading GAN framework to accomplish this task. We first utilize the first GAN to generate a coarse-grained urban plan (i.e., urban functional zones) based on human instructions and surrounding environments. Then, we employ the second GAN to generate the fine-grained urban plan (i.e., land-use configurations) based on the coarse-grained plan, human instructions, and surrounding environments. This model structure naturally captures the spatial hierarchies between spatial grids and functional zones. Additionally, to alleviate the data sparsity issue, we provide a conditioning augmentation module to augment training data samples.

In summary, in this paper, we propose a deep human-instructed generative framework for automated urban planning. The contributions of this paper can be summarized as follows:

- We formulate automated urban planning as a deep generative task and propose practical frameworks.
We uncover that human instructions from urban planners can be formulated as the generated conditions. We capture spatial hierarchies in urban planning to improve the performance of deep urban planners. We propose data augmentation methods to augment training samples in order to enhance generative learning. We conduct extensive experiments and case studies on a realistic dataset to demonstrate the superiority and efficacy of our work.

2 Definitions and problem statement

2.1 Definitions

2.1.1 Target area and surrounding contexts

The target area refers to a geographically square region that is empty and waiting to be planned. The surrounding contexts have the same square shape as the target area and encircle the target area in different directions. Figure 3 illustrates the spatial relationships between the target area and surrounding contexts. There are many socioeconomic activities in surrounding contexts, such as commutes, real estate transactions, and entertainment. These activities affect the urban plan of the target area.
2.1.2 Land-use configuration

Land-use configuration is our generative goal, which refers to grid-level planning for a geographical region. We adopt the quantitative definition in [7] for it. Figure 4 shows its example that has three dimensions: latitude, longitude, and POI category. Specifically, the latitude and longitude coordinates comprise a geographical region, which is divided into $N \times N$ grids. Then, for each POI category, we figure out how many POIs are in each grid to make one layer. Different POI categories will build up different layers, which form the 3D tensor configuration.

2.1.3 Urban functional zone

An urban functional zone is a coarse-grained urban plan that provides a rough sketch and high-level guidance for producing a land-use configuration. Following the idea in studies [10], we utilize geographical data and human mobility to extract urban functional zones. Specifically, we first divide a geographical area into $N \times N$ grids. Then, we consider the
Urban functional zones is a 2D matrix, which provides a high-level guidance for generating a land-use configuration. The data structure of urban functional zones is shown in Figure 5, where different colors indicate different functional zones.

### 2.1.4 Human instruction

In this paper, human instruction is used to guide the generation process of our planning framework. We quantify the semantic meaning of such instructions into different levels to allow our model to perceive them. For instance, the range of the "green rate" (i.e., the coverage of green plants in a geographical area) is \([0 \sim 1]\). We divide the green rate into multiple green rate levels. Human experts can input the label of a green rate level into our planner to indicate their planning requirements. So, these labels of "green rate level" refer to human instructions.

### 2.2 Problem statement

In this study, we intend to develop an automated urban planner capable of generating configurations based on human instructions and the socioeconomic features of surrounding contexts. Formally, given a list of land-use configurations \(\tilde{X} = \{\tilde{X}^{(1)}, \tilde{X}^{(2)}, \ldots, \tilde{X}^{(K)}\}\), socioeconomic features of surrounding contexts \(S = \{s^{(1)}, s^{(2)}, \ldots, s^{(K)}\}\), human instruction \(I = \{i^{(1)}, i^{(2)}, \ldots, i^{(K)}\}\), our purpose is to find the mapping function \(f : (S, I) \rightarrow \tilde{X}\) that takes socioeconomic features \(S\) and human instruction \(I\) as input, and outputs the corresponding land-use configurations \(\tilde{X}\).

### 3 Proposed method

In this section, we introduce the technical details of the proposed framework.
3.1 Framework overview

Figure 6 shows the proposed framework, which includes four main steps: (1) We collect the land-use configurations and the corresponding urban functional zones of a series of target areas; (2) We preserve the information of surrounding contexts and human instructions into embedding vectors. Specifically, for surrounding contexts, we utilize a spatial attributed graph to organize their features and employ a graph embedding model to convert the graph into embedding. For the human instruction, we regard the one-hot vector of human instruction texts as its embedding; (3) We build up an automated urban planner based on deep generative learning. Specifically, we first concatenate the embedding of surrounding contexts and human instructions as the condition embedding. Then, we input this condition embedding into a generation model to learn the relationship between the input and land-use configurations. (4) We employ the well-trained generation model as the automated urban planner to generate ideal land-use configurations. During this process, we need to capture the spatial hierarchies between the coarse-grained urban plan (i.e., urban functional zones) and the fine-grained urban plan (i.e., land-use configurations).

3.2 Collecting target area and surrounding contexts

To train our deep urban planner model, we must collect data about the target area and its surrounding contexts. Specifically, we begin by collecting a list of geographical points with longitude and latitude coordinates. Then, we regard each geographical point as the central point to draw the corresponding target area (i.e., one square shape with 1 km² area). Later, we find all surrounding contexts for each target area according to the spatial relationships depicted in Sect. 2.1.1. Following that, we gather all of the target areas and surrounding contexts and pair them together. In the following paragraphs, to make it convenient, we employ the $k$-th target area and the corresponding surrounding contexts to introduce denotations and all calculation processes.

3.3 Collecting land-use configurations and discovering urban functional zones

To build an AI-based model for generating land-use configurations, we need a lot of land-use configuration samples to train the model. The collecting process for the $k$-th target area is as follows: We divide the $k$-th target area into $N \times N$ squares based on latitude and longitude. Next, we determine the number of POIs belonging to each POI category in each square and arrange them in a matrix based on the position of the squares. After that, we
stack different matrices together to form a 3D tensor. Figure 4 shows the data structure of the tensor. The $k$-th land-use configuration is denoted by $\hat{X}^{(k)} \in \mathbb{R}^{N \times N \times M}$, where $M$ is the number of POI category. In addition, urban functional zones refer to the coarse-grained urban plan, which provides a rough sketch for a land-use configuration. Figure 5 shows the data structure of urban functional zones. The format of the $k$-th urban functional zones is denoted by $F^{(k)} \in \mathbb{R}^{N \times N}$, where each value in $F^{(k)}$ indicates the urban function of the associated geographical square.

3.4 Extracting features of the surrounding contexts

The socioeconomic features of surrounding contexts affect the urban planning design of the target area. For example, if there are a lot of apartments in the area, entertainment POIs (e.g., a gym or club) should be built in the target area to make the whole area more lively. In this paper, we utilize a spatial attributed graph to embrace all features of surrounding contexts and employ a graph embedding model to convert these features into an embedding.

First, Sect. 2.1.1 shows the spatial correlations among surrounding contexts, which can be formulated as a graph. The $k$-th surrounding contexts can be denoted by $G^{(k)} = (V^{(k)}, E^{(k)})$, where $V^{(k)}$ indicates the vertex set, and one vertex is one surrounding context; $E^{(k)}$ is the edge set, and one edge reflects the spatial connectivity between two associated vertices.

Second, we extract explicit features of each vertex from three perspectives: (1) house price change, which reflects living attractions for a geographical area. Specifically, for each vertex, we first collect the house price in $T$ months. Then, we obtain the house price change by using the house price of each month to deduct the previous one. Finally, we collect the house price change of all contexts, denoted by $V = \{v_1, v_2, \ldots, v_8\}$, where $V \in \mathbb{R}^{8 \times (T-1)}$ and $v$ is the house price change of one context. (2) POI Ratio, which describes urban function varieties of a geographical area. Specifically, for each vertex, we first count the number of POIs belonging to each POI category. Then, we divide the value by the total POI numbers in the area to obtain a POI ratio vector. Finally, we collect all POI ratio vectors, denoted by $R = \{r_1, r_2, \ldots, r_8\}$, where $R \in \mathbb{R}^{8 \times M}$ and $r$ is the POI ratio vector of a context. (3) Transportation shows the traffic condition of a geographical area. We extract two kinds of transportation features: public transportation and private transportation. For public transportation, in each context, we collect 5 values: (a) the leaving volume of buses per day; (b) the arriving volume of buses per day; (c) the transition volume of buses per day; (d) the number of bus stops in per square meters; (e) the average price of each bus trip. The five values are organized as the public transportation feature vector, denoted by $o$. Finally, we collect the vectors of all contexts, denoted by $O = \{o_1, o_2, \ldots, o_8\}$, where $O \in \mathbb{R}^{8 \times 5}$. For private transportation, in each context, we also collect 5 values: (a) the leaving volume of taxis per day; (b) the arriving volume of taxis per day; (c) the transition volume of taxis per day; (d) the average velocity of taxis per day; (e) the average commute distance of taxis per day. The five values are organized as the private transportation feature vector, denoted by $u$. Finally, we collect all contexts’ vectors, denoted by $U = \{u_1, u_2, \ldots, u_8\}$, where $U \in \mathbb{R}^{8 \times 5}$.

Third, we horizontally concatenate different kinds of features together to form a socioeconomic feature set, denoted by $F^{(k)} = \{V, R, O, U\}$. $F^{(k)} \in \mathbb{R}^{8 \times (M+T+9)}$, where one row reflects socioeconomic characteristics of one surrounding context. Then, we map each row of $F^{(k)}$ to the graph $G^{(k)}$ to form the spatial attributed graph, denoted by $G^{(k)} = (V^{(k)}, E^{(k)}, F^{(k)})$. Finally, a variational graph auto-encoder model [11] is used to convert $G^{(k)}$ into the graph embedding $s^{(k)}$ that contains not only the spatial relationships among surrounding contexts but also all socioeconomic situations.
3.5 Embedding human instruction

Human Instruction reflects the urban planning requirements of human experts. To make our model perceive such requirements, we simplify them into different options that can be selected by urban experts based on their needs. The one-hot vectors of these options are regarded as the embedding of human instructions. The human instruction for the $k$-th target area is denoted by $i^{(k)}$.

Ultimately, for the $k$-th target area, we have obtained the land-use configuration $\hat{X}^{(k)}$, the urban functional zones $F^{(k)}$, the embedding of surrounding contexts’ features $s^{(k)}$, and the embedding of the human instruction $i^{(k)}$. Then, we combine them together to form a data sample that can be learned by our deep urban planner, denoted by $[\hat{X}^{(k)}, F^{(k)}, s^{(k)}, i^{(k)}]$.

3.6 Automated urban planner model

3.6.1 Conditional land-use variational autoencoder

We propose a conditional variational autoencoder-based deep urban planner. Figure 7 shows the encoder–decoder model paradigm. The encoder is to convert the land-use configuration $\hat{X}^{(k)}$ and the concatenated generative conditions ($s^{(k)}, i^{(k)}$) into a latent embedding vector $z^{(k)}$. During the process, we propose a variational Gaussian embedding component to alleviate the data sparsity issue. This component is implemented based on two linear layers, which learns the conditional distribution $p(z^{(k)} | \hat{X}^{(k)}, s^{(k)}, i^{(k)})$. Assuming that the conditional distribution is normal distribution, we only require to estimate the mean $\mu$ and variance $\delta$ of the distribution. After that, we can sample data from the learned distribution by the reparametrization technique and regard it as the output of the encoder. The calculation process can be formulated as follows:

$$\begin{align*}
    c^{(k)} &= \text{Concat}(s^{(k)}, i^{(k)}), \\
    x^{(k)} &= \text{Flatten}(\hat{X}^{(k)}), \\
    \mu^{(k)} &= \text{Linear}_1(x^{(k)}, c^{(k)}), \\
    \delta^{(k)} &= \text{Linear}_2(x^{(k)}, c^{(k)}), \\
    z^{(k)} &= \mu^{(k)} + \delta^{(k)} \times \epsilon.
\end{align*}$$

(1)
The last line of Eq. (1) is the reparameterization trick, where $\epsilon$ is sampled from a standard normal distribution $N(0, 1)$. Next, we input $z^{(k)}$ and $c^{(k)}$ into the decoder for learning generation.

The decoder is to reconstruct $\hat{X}^{(k)}$ and construct the corresponding urban functional zones $F^{(k)}$ based on $z^{(k)}$ and $c^{(k)}$. Inspired by [12], the multi-head decoder in VAE may regularize the decoder for producing more reasonable results. Here, we regard urban functional zones as the generated constraints in the decoder for capturing spatial hierarchies in urban plan. Specifically, we first input $z^{(k)}$ and $c^{(k)}$ a linear layer activated by Relu function to get the reconstructed land-use configuration $\hat{X}^{(k)}$. Next, we input $z^{(k)}$ and $c^{(k)}$ into another linear layer activated by sigmoid function $\sigma$ to construct urban functional zones $\hat{F}^{(k)}$. The calculation process can be formulated as follows:

$$\begin{align}
\hat{X}^{(k)} &= \text{Relu}(\text{Linear}_3(z^{(k)}, c^{(k)})), \\
\hat{F}^{(k)} &= \sigma(\text{Linear}_4(z^{(k)}, c^{(k)})).
\end{align}$$

There are three optimization objectives: (1) Minimizing the difference between $\hat{X}^{(k)}$ and $\hat{X}^{(k)}$, denoted by $\mathcal{L}_X$; (2) Minimizing the difference between $p(z^{(k)} \mid x^{(k)}, c^{(k)})$ and its prior distribution $q(z^{(k)})$ that is a standard normal distribution, denoted by $\mathcal{L}_p$; (3) Minimizing the difference between $\hat{F}^{(k)}$ and $F^{(k)}$, denoted by $\mathcal{L}_F$; The final loss $\mathcal{L}$ is the combination of $\mathcal{L}_X$, $\mathcal{L}_p$, and $\mathcal{L}_F$, which can be formulated as follows:

$$\begin{align}
\mathcal{L}_X &= \frac{1}{K} \sum_{k=1}^{K} (\hat{X}^{(k)} - \hat{X}^{(k)})^2, \\
\mathcal{L}_p &= \frac{1}{K} \sum_{k=1}^{K} \text{KL}[p(z^{(k)} \mid x^{(k)}, c^{(k)}) || q(z^{(k)})] \\
\mathcal{L}_F &= -\frac{1}{K} \sum_{k=1}^{K} \log(F^{(k)}) \\
\mathcal{L} &= \mathcal{L}_X + \mathcal{L}_p + \lambda \times \mathcal{L}_F
\end{align}$$

where KL represents the Kullback–Leibler divergence between $p(.)$ and $q(.)$; $\lambda$ controls the strength of the generated constraints (i.e., urban functional zones), and its value range is $[0 \sim 1]$. When the model converges, the well-trained decoder is the land-use configuration generator. During the testing phase, we first sample a latent embedding from the learned distribution. Then, the decoder takes the sampled embedding and the condition embedding as input, and outputs the generated land-use configuration. The variety of sampled embeddings will generate ideal land-use combinations of different types, hence enhancing the generative diversity of our model.

### 3.6.2 Conditional land-use generative adversarial networks

Although the preliminary work has achieved good performance, we find that its generative performance is unstable. The underlying reason is that multi-head decoder-based urban planner considers the spatial hierarchies for generating ideal land-use configurations via optimizing a regularization item. During the learning process, it is hard to determine the contribution of such spatial hierarchies, resulting in unstable generation performance.

Thus, in this journal version, inspired by the workflow of urban experts, we propose a conditional generative adversarial network (GAN) based urban planner. Figure 8 shows the model structure, which three main parts: conditioning augmentation module, zone-level planning generation, grid-level planning generation.
Conditioning augmentation Standard urban plan samples are too rare to train a deep generative model. To alleviate the data sparsity issue, we adopt the conditioning augmentation module depicted in [13]. To be convenient, we adopt the $k$-th empty target area to explain the following calculation process. The concatenated embedding of human instructions and surrounding contexts is denoted by $z^{(k)} \in \mathbb{R}^{1 \times O}$, where $O$ is the size of the feature dimension. Specifically, we first estimate the distribution of $z^{(k)}$. Then, we randomly sample an augmented embedding $c^{(k)}$ from the distribution and regard it as the input of the zone-level generation module. The estimated distribution is assumed to be a normal distribution, denoted by $\mathcal{N}(\mu(z^{(k)}), \delta(z^{(k)}))$, where $\mu(\cdot)$ and $\delta(\cdot)$ indicate the mean and covariance function, respectively. The sampling operation is achieved by the reparameterization technique, which can be formulated as follows:

$$c^{(k)} = \mu(z^{(k)}) + \delta(z^{(k)}) \times \epsilon$$

where $\epsilon$ is a random variable vector sampled from a standard normal distribution $\mathcal{N}(0, 1)$.

Zone-level planning generation The first step is to generate a coarse-grained zone-level planning based on human instructions and surrounding contexts. Specifically, for the $k$-th empty target area, we first concatenate the embedding $c^{(k)}$ and the random variable embedding $\eta^{(k)}$ together, then input it into the generator $G_1$ to generate the urban functional zones (coarse-grained urban plan). Here, $\eta^{(k)}$ is sampled from the standard normal distribution $\mathcal{N}(0, 1)$, which can improve the model robustness. Next, we combine the generated result and the embedding $z^{(k)}$ together, then input it into the discriminator $D_1$. The discriminator is to justify whether the input is the combination of the real urban functional zones $F^{(k)}$ and $z^{(k)}$. We alternatively optimize the generator and the discriminator until model convergence.

When optimizing the generator, we minimize Eq. (5):

$$\mathcal{L}_{G_1} = \sum_{k=1}^{K} \log(1 - D_1(G_1(\eta^{(k)}, c^{(k)}), z^{(k)})) + \alpha \cdot KL[\mathcal{N}(\mu(z^{(k)}), \delta(z^{(k)}))\|\mathcal{N}(0, 1)],$$

where $KL[.]$ indicates the Kullback–Leibler (KL) divergence between the distribution $\mathcal{N}(\mu(z^{(k)}), \delta(z^{(k)}))$ and a standard normal distribution $\mathcal{N}(0, 1)$; $\alpha$ is a scalar, which determines the contribution of the item $KL[.]$ in $\mathcal{L}_{G_1}$. $\mathcal{L}_{G_1}$ can be divided into two parts by “+.” Intuitively, the first part tries to minimize the differences between the generated zone-level planning and the real zone-level planning. This part aims to improve the generation performance of the generator. The second part tries to smooth the distribution $\mathcal{N}(\mu(z^{(k)}), \delta(z^{(k)}))$.
produced by the conditioning augmentation module. This part aims to improve the diversity and quality of the input embedding $c^k$ [14].

When optimizing the discriminator $D_1$, we maximize Eq. (6):

$$
\mathcal{L}_{D_1} = \sum_{k=1}^{K} \log(1 - D_1(G_1(\eta^k, c^k), z^k)) + \log D_1(F^k, z^k).
$$

Intuitively, $\mathcal{L}_{D_1}$ improves the discrimination ability by urging the discriminator to provide lower scores for the generated results and evaluate higher scores for the real golden standards.

Ultimately, the well-trained generator can generate an ideal urban plan rough sketch $\tilde{F}^k \in \mathbb{R}^{N \times N}$ for the $k$-th empty area according to human instructions and surrounding contexts. Each value in $\tilde{F}^k$ indicates the urban functionality label at the associated geographical location.

**Grid-level planning generation** The second step is to generate a fine-grained urban plan under the previous learned urban sketch. Specifically, for the $k$-th empty target area, we first flat the urban functional zones into a one dimensional embedding vector. Then, we concatenate it with the condition embedding $z^k$ to form $b^k$. Moreover, we input $b^k$ into the conditioning augmentation module to augment the input. After that, we concatenate the augmented embedding $a^k$ with a random variable embedding vector $\psi$ as the input of the second generator $G_2$. This vector $\psi$ can increase the diversity of input to improve the learning robustness. Following that, we input the generated land-use configuration and the condition embedding $z^k$ into the second discriminator $D_2$. The discriminator will justify whether it is the combination of the real land configuration $\hat{X}^k$ and condition embedding $z^k$. We alternatively optimize $G_2$ and $D_2$ until model convergence.

The optimization objective of $G_2$ is as follows:

$$
\mathcal{L}_{G_2} = \sum_{k=1}^{K} \log(1 - D_2(G_2(\epsilon^k, a^k), z^k)) + \beta \cdot KL[N(\mu(b^k), \delta(b^k))\|N(0, 1)],
$$

where $\beta$ is a scalar, which determines the contribution of the KL divergence in the total loss. We minimize the Eq. 7 during the optimization process. $\mathcal{L}_{G_2}$ can be divided into two parts by the sign “+”. The first part is to improve the generation ability of the generator $G_2$. The second part is to improve the conditioning augmentation module to produce more diversified data for better model training.

The optimization objective of $D_2$ is as follows:

$$
\mathcal{L}_{D_2} = \sum_{k=1}^{K} \log(1 - D_2(G_2(\epsilon^k, a^k), z^k)) + \log D_2(\hat{X}^k, z^k).
$$

We maximize Eq. 8 during the optimization process. $\mathcal{L}_{D_2}$ improves the discrimination ability by urging the discriminator to provide lower score for the generated land-use configurations.
and evaluate higher scores for real ones. Finally, when model converges, we can adopt the first generator $G_1$ to produce reasonable urban functional zones and employ the second generator $G_2$ to produce ideal land-use configurations.

### 3.7 Difference from prior literature

With the advancement of artificial intelligence (AI), numerous researchers are interested in how to use AI models to improve and accelerate urban planning. However, the past literature has certain limitations: (1) The automated urban planning process cannot be customized according to human requirements; (2) the generated urban planning solutions are irrational; and (3) standard urban planning samples are limited. Our research intends to address the three restrictions. Human instruction for urban planning is viewed as a conditional input in our approach for generating land-use configurations. We also use spatial hierarchies in urban planning to enhance the quality of developed configurations. In addition, we design a data augmentation module to address the data sparsity issue in automated urban planning.

### 4 Experimental results

We conducted extensive experiments to answer: **Q1.** Is our method effective at generating land-use configurations? **Q2.** What is the impact of the data augmentation module for land-use configuration generation? **Q3.** What is the influence of each technical component in our framework? **Q4.** How robust is our method when confronted with different scaled land-use configuration generation tasks? **Q5.** Compared with real land-use configurations, how is the quality of the generated ones?

#### 4.1 Experimental setup

##### 4.1.1 Data description

Our dataset is composed of land-use configuration, urban functional zones, geospatial contexts, and human instructions. Our research focuses on Beijing. The data collection process is as follows: we first crawled 2990 residential communities from soufun.com and downloaded 328,668 POIs from openstreetmap.org to construct land-use configuration samples referring to [7]. These POIs belong to 20 POI categories that are shown in Table 1. Then, we collected taxi trajectories from T-drive project [15] and downloaded road networks, POIs from openstreetmap.org to discover urban functional zones of each residential community referring to [10]. Next, we used housing price data crawled from soufun.com, mobile checkins crawled from weibo.com, taxi trajectories, and POIs to extract socioeconomic features of surrounding contexts. Moreover, we utilized the green rate including in crawled residential community data to construct human instructions. Finally, according to the location of each residential community, the land-use configurations, urban functional zones, surrounding contexts, and human instruction were paired to form the final dataset.

##### 4.1.2 Evaluation metrics

We divided the collected land-use configurations into five levels according to their green rate. Green level labels are regarded as human instructions in our experiments. The land-use
configurations of different green levels come from different distributions. To evaluate the model performance, we measured the distance between the distribution of generated configurations and the distribution of original configurations under different green level settings. Let $w_j$ denote the weight of the green level $j$, which is the number of land-use configurations belonging to $j$; $Y_j$ denote the distribution of original land-use configurations; $\hat{Y}_j$ denote the distribution of generated land-use configurations. We use four evaluation metrics in our work: (1) Average Kullback–Leibler (KL) Divergence: $AVG_{KL} = \frac{\sum_{j=1}^{5} w_j \cdot KL(Y_j, \hat{Y}_j)}{\sum_{j=1}^{5} w_j}$. (2) Average Jensen–Shannon (JS) Divergence: $AVG_{JS} = \frac{\sum_{j=1}^{5} w_j \cdot JS(Y_j, \hat{Y}_j)}{\sum_{j=1}^{5} w_j}$. (3) Average Hellinger Distance (HD): $AVG_{HD} = \frac{\sum_{j=1}^{5} w_j \cdot HD(Y_j, \hat{Y}_j)}{\sum_{j=1}^{5} w_j}$. (4) Average Cosine Distance (Cos): $AVG_{Cos} = \frac{\sum_{j=1}^{5} w_j \cdot Cos(Y_j, \hat{Y}_j)}{\sum_{j=1}^{5} w_j}$. The lower the metric value is, the better performance the model exhibits.

### 4.1.3 Why use distribution distances as evaluation metrics?

**Data distribution** There are five kinds of human instructions in our dataset: Green0, Green1, Green2, Green3, Green4. The whole dataset is divided into five parts according to these instructions. Figure 9 illustrates the proportion of each part in our dataset. We can find that from the data size perspective, Green1 > Green0 > Green2 > Green3 > Green4.

**Explanation of evaluation metrics** As shown in Fig. 10, the data distribution is divided into 5 clusters. A generated land-use configuration is produced based on a specific human instruction. When compared to others, the generated configuration should be closer to the data cluster that has the same human instructions. Thus, we can utilize the distribution distance metrics to evaluate the model generation performance.

---

1. [https://en.wikipedia.org/wiki/Kullback-Leibler_divergence](https://en.wikipedia.org/wiki/Kullback-Leibler_divergence)
2. [https://en.wikipedia.org/wiki/Jensen-Shannon_divergence](https://en.wikipedia.org/wiki/Jensen-Shannon_divergence)
3. [https://en.wikipedia.org/wiki/Hellinger_distance](https://en.wikipedia.org/wiki/Hellinger_distance)
4. [https://en.wikipedia.org/wiki/Cosine_similarity](https://en.wikipedia.org/wiki/Cosine_similarity)
4.1.4 Baseline methods

We compared our proposed framework against the following baseline models: CGAN: [16] is the conditional version of generative adversarial nets (GAN). Compared with classical GAN, CGAN adds a conditional input to the generator and discriminator, respectively. CVAE: [17]
is another conditional generative model, which is an extension of variational auto encoder. DCGAN: [18] is a classical image generative model, which utilizes convolutional and convolutional transpose layers in the generator and discriminator, respectively. WGAN: [19] improves the stability of learning of traditional GAN and makes the learning curve of GAN become meaningful. WGAN-GP: [20] is an enhanced WGAN, which utilizes gradient penalty to replace clipping weights of WGAN. LUCGAN: [7] is designed for land-use configuration generation, which can generate configurations automatically based on surrounding contexts’ socioeconomic features. CLUVAE: [9] is our preliminary work, which can generate land-use configurations based on human instructions and surrounding contexts under the VAE setting.

Besides, to validate the necessity of each component of the preliminary work, we developed four model variants of CLUVAE: CLUVAE*, which removes the embedding of human instruction in condition embedding; CLUVAE′, which removes the embedding of surrounding contexts in condition embedding. CLUVAE-, which does not force the decoder to produce urban functional zones; CLUVAE#, which removes the variational Gaussian component in CLUVAE. Additionally, we developed four model variants of our journal version work (CLUGAN) to examine the influence of each technical component: CLUGAN*, which removes the embedding of human instruction in condition embedding. CLUGAN′, which removes the embedding of surrounding contexts in condition embedding. CLUGAN-, which removes the zone-level planning generation and generates the grid-level planning directly. CLUGAN#, which removes the conditioning augmentation module of CLUGAN.

4.1.5 Environmental setting

We conducted all experiments in the Ubuntu 18.04.3 LTS operating system, plus Intel(R) Core(TM) i9-9920X CPU@ 3.50GHz, 1 way SLI Titan RTX and 128GB of RAM, with the framework of Python 3.7.4, Tensorflow 2.0.0.

4.1.6 Hyperparameters and reproducibility

In our experiments, we mixed up our processed dataset at random and divided it into two separate sets based on the index of the target area. The first 90% of the sets are the training set, and the last 10% are the testing set. For the conference work CLUVAE, the encoder part has two fully connected layers; the decoder part has two branches, and each branch is composed of two fully connected layers. The training epochs were set to 50 during the model training process, the Adaptive Moment Estimation (Adam) optimizer was used to optimize the model with learning rate 0.0001, and the $\lambda$ value in the loss function $L$ (Eq. 3) was set to 0.55. For the journal version work CLUGAN, we implemented two generative adversarial networks to resolve this task. The first generator is composed of three transpose convolution layers, and the first discriminator is constituted by two convolution layers with a dropout rate of 0.3. The second generator is composed of three 3D transpose convolution layers, and the second discriminator is constituted by two 3-D convolution layers with a dropout rate of 0.4. We set the value of $\alpha$ contained in the Eq. 5 as 0.8 and the value of $\beta$ used in the Eq. 7 as 0.7. We used Adaptive Moment Estimation (Adam) with a learning rate of 0.0001 over 500 epochs to improve the two GANs. We set hyperparameters in baseline models as the default value.
4.2 Overall performance Q1

We compared our framework with other baseline models in terms of AVG_KL, AVG_JS, AVG_HD, and AVG_Cos. Figure 11 shows the CLUGAN outperforms other baseline methods in terms of all evaluation metrics. This observation indicates that CLUGAN can comprehend the planning requirements and restrictions to generate ideal land-use configurations. Additionally, another interesting observation is that CLUGAN is superior to CLUVAE in terms of all evaluation metrics. One possible reason for this is that CLUGAN’s model structure can naturally capture spatial hierarchies. This gives model learning a clear direction for optimization, which leads to more reasonable generations. In summary, this experiment validates that our method can effectively understand human and surrounding environments to generate ideal urban plans.

4.3 Effectiveness check of variational Gaussian component Q2

To resolve the data sparsity issue, we added a Variational Gaussian module into CLUVAE. Thus, we developed a model variant CLUVAE# by removing this module to observe its influence on land-use configuration generation. We ran CLUVAE and CLUVAE# six times,
Figure 12 shows the changing trend of performance, and its mean and variance value. We can find that CLUVAE has more stable and better generation performance in comparison with CLUVAE#. The underlying driver is that the Variational Gaussian module increases the number and diversity of data samples, resulting in our deep urban planner model can be sufficiently trained to generate ideal urban plans. This experiment validates that the Variational Gaussian model is vital for keeping stable and robust generation performance of CLUVAE.

4.4 Effectiveness check of conditioning augmentation module Q2

In the journal version work, we employed the conditioning augmentation module to resolve the data sparsity issue. We also ran CLUGAN and CLUGAN# six times, respectively. Figure 13 shows the comparison results in terms of the changing trend, mean, and variance of different evaluation metrics. We can find that CLUGAN outperforms CLUGAN# on average level and has more stable generative performances. The underlying reason is that the conditioning augmentation module can enlarge the number and diversity of data samples, allowing CLUGAN to capture additional planning patterns for producing better land-use
configurations. This experiment validates that conditioning augmentation is a key part of keeping generation performance stable and good.

4.5 Ablation study for CLUVAE Q3

We conducted ablation studies and experiments to validate the necessity of each part of CLUVAE. Figure 14 shows the performances of CLUVAE, CLUVAE*, CLUVAE’, and CLUVAE- in terms of all evaluation metrics. We can find that CLUVAE exceeds other baselines by a significant margin. The underlying reason is that, with the integration of various technical components, CLUVAE may interpret human commands and surrounding situations more thoroughly for producing better land-use configurations. Another interesting observation is that CLUVAE is superior to CLUVAE* and CLUVAE’. This observation suggests that understanding people and their environments plays an important role in designing ideal urban planning solutions. Moreover, the superiority of CLUVAE over CLUVAE- indicates that capturing spatial hierarchies can regularize the generation process for producing ideal urban plans. This experiment indicates that each technical component of CLUVAE is indispensable.
4.6 Ablation study for CLUGAN Q3

We conducted ablation study experiments to validate the necessity of each technical component of CLUGAN. We developed three variants: CLUGAN*, CLUGAN’ and CLUGAN- and compared the generation performance in terms of AVG_KL, AVG_JS, AVG_HD, and AVG_Cos. Figure 15 shows the comparison results. We can find that CLUGAN outperforms CLUGAN* and CLUGAN’. This observation indicates that the human-likely two-step generation framework can improve the comprehension of human instructions and surrounding contexts, resulting in good generation performance. Moreover, we can observe that CLUGAN beats CLUGAN-. A potential reason is that the stack GAN structure generates zone-level planning and then, produces grid-level planning. This generation strategy leads each step toward a clear optimization goal, resulting in capturing spatial hierarchies sufficiently to generate better land-use configurations. This experiment validates that each technical component of CLUGAN is significant.
To study the influence of square size, we set $N = 5$, $N = 10$, $N = 25$, $N = 50$, $N = 100$, respectively, to compare the generative performance. The smaller value of $N$ is, the larger size of the square is. Figure 16 shows the comparison results. We can find that as the square size increases, the generation performance decreases. One possible reason for this is that when a square is bigger, it means that land-use configurations are simpler and have less planning detail. So, it is easy for the generative model to learn their patterns, leading to good generation performance. Additionally, another interesting observation is that CLUGAN beats other baseline models in terms of all metrics, regardless of the square size. A potential interpretation for the observation is that CLUGAN captures the human requirements and surrounding planning restrictions to produce regularized and good land-use configurations. This experiment validates the robustness of CLUGAN.

4.8 Comparison study between real and generated land-use configurations under different human instructions Q5

We selected the most representative real and generated land-use configurations under different human instructions to evaluate generation quality. Figure 17 shows the comparison results,
Fig. 16  The influence of square sizes for land-use configuration generation

Fig. 17  Visualizations of real and generated land-use configurations
which are visualized in a 3D space. The left color legend indicates the mapping relations between POI categories and colors; the right 3D part reflects the POI distribution of land-use configurations; the height of each bar indicates the number of POIs at the corresponding position; the label of each subfigure is the corresponding human instruction (i.e., green level label). A careful inspection for Fig. 17a, b shows that the generated configurations are organized and contain enough planning details for implementation in realistic. This observation reflects that our model has remarkable imagination and designing ability to generate land-use configurations considering human requirements and surrounding contexts.

In addition, for the green level of land-use configurations, Green0 < Green1 < Green2 < Green3 < Green4. Another interesting observation is that with the increase in green levels, the POI distribution of land-use configurations becomes more sparse. A potential interpretation for the observation is that the dense POI distribution reflects that the corresponding urban plan does not have enough space to improve the green level. In summary, this experiment shows that our method can capture the characteristics of land-use configurations of different green levels. Thus, urban planners can use our method to customize land-use configurations based on their requirements.

5 Related work

Data fusion refers to the process of combining multiple data sources to extract relevant information for performing significant tasks. In reality, many application scenarios require data fusion in order to integrate information for better modeling. For instance, Wang et al. proposed a user profiling framework based on reinforcement learning by simulating user check-in activities in a geographical region. Liu et al. modeled the interaction coupling of multi-view spatiotemporal contexts to predict shared-bike destination. Wang et al. studied the urban vibrancy of residential communities via analyzing multi-source data such as check-ins, urban geography data, and human mobility. Wang et al. fused the monitoring value of different kinds of sensors together to detect system status of water treatment plants. In this paper, we have more complex data resources than in the aforementioned research. Due to the complexity of urban planning, we use a spatial attributed graph to arrange the socioeconomic variables of surrounding contexts and a graph embedding model to maintain these qualities in an embedding vector.

Deep generative learning There are three kinds of popular approaches in the deep generative learning domain: normalizing flows (NF), variational autoencoders (VAE), and generative adversarial networks (GAN) [26]. NF refers to a set of generative models with tractable distributions where both sampling and density evaluation can be efficient and exact [27]. VAE is capable of learning the latent representations of data and providing deep inference models [28]. GAN can replicate the distribution of real data through the competition of generator and discriminator in a zero-sum game setting [29]. Deep generative learning has been successfully applied to many applications [13, 30]. For instance, Zhang et al. utilized a stack GAN structure to generate realistic images according to text descriptions [13]. Kang et al. built a semi-supervised VAE structure to design new molecules with desired properties [30]. Chentharamakshan et al. proposed CogMol to design the new drug molecules targeting Covid19 [31]. Compared with these existing works, we implement two automated urban planning frameworks based on GAN and VAE, respectively. Our model can produce ideal land-use configurations based on human instructions and surrounding environments.
Urban planning is essential for the future development of a geographic area, as it involves developing the land-use configuration in accordance with human requirements and surrounding contexts. [32, 33]. Plenty of existing works in this domain aim to balance humans and environments to produce ideal urban plans. For instance, Ratcliffe et al. [34] studied the relationship between urban planning and real estate development. Papa et al. [35] studied how to build up a sustainable smart city via urban planning. Recently, with the development of deep learning, many researchers bring deep models to renovate classical urban planning [7, 36, 37]. For instance, Wang et al. [7] utilized GAN-based framework to generate land-use configuration based on surrounding contexts. Compared with these works, CLUVAE and CLUGAN have higher customization abilities and generate personalized land-use configurations.

6 Conclusion remarks

In this paper, we study the problem of automated urban planning. We formulate it as a deep generative learning task and propose practical frameworks. In the preliminary work, we propose a deep conditional VAE-based framework. The encoder part is to learn the relationships between condition embeddings and land-use configurations. The decoder part is to reconstruct land-use configurations and urban functional zones simultaneously in order to capture the spatial hierarchies in urban planning. To alleviate the data sparsity issue, we utilize the Variational Gaussian embedding mechanism to augment the data. Although it has achieved good performance, we found that its generation performance is unstable. The underlying reason is that the way we capture spatial hierarchies may lead to ambiguous optimization directions, resulting in suboptimal generation performance. Thus, in this journal version, we propose a deep stack GAN-based framework after analyzing the workflow of urban experts. We use the first GAN to generate zone-level planning based on human requirements and surrounding contexts. Then, we use the second GAN to generate grid-level planning based on the learned zone-level planning. This generation strategy takes the spatial hierarchies into account naturally and gives each step a clear optimization goal. Additionally, we utilize the conditioning augmentation module to increase the diversity and volume of the data for improving the model learning. Extensive experiments validate that deep generative learning has great potential to renovate traditional urban planning. In the future, we will focus on adding more complex human instructions to our work to make it more useful and practical.

References

1. Litman T (2015) Analysis of public policies that unintentionally encourage and subsidize urban sprawl
2. Public: Sidewalks. [EB/OL] (2022) https://www.sidewalklabs.com/toronto
3. Public: Baidu AI City. [EB/OL] (2017) https://www.globenewswire.com/news-release/2017/12/20/1267217/0/en/Baidu-and-Xiongan-New-Area-Sign-Strategic-Agreement-to-Develop-Smart-City.html
4. Cugurullo F (2020) Urban artificial intelligence: from automation to autonomy in the smart city. Front Sustain Cities 2:38
5. Shen J, Liu C, Ren Y, Zheng H (2020) Machine learning assisted urban filling
6. Ye X, Du J, Ye Y (2021) Masterplangan: facilitating the smart rendering of urban master plans via generative adversarial networks. Environ Plan B: Urban Anal City Sci, 2399803211023516
7. Wang D, Fu Y, Wang P, Huang B, Lu C-T (2020) Reimagining city configuration: automated urban planning via adversarial learning. In: Proceedings of the 28th international conference on advances in geographic information systems, pp 497–506
8. Dong H-W, Hsiao W-Y, Yang L-C, Yang Y-H (2018) Musegan: multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. In: Thirty-second AAAI conference on artificial intelligence

9. Wang D, Liu K, Johnson P, Sun L, Du B, Fu Y (2021) Deep human-guided conditional variational generative modeling for automated urban planning. In: 2021 IEEE international conference on data mining (ICDM), pp 679–688. https://doi.org/10.1109/ICDM51629.2021.00079

10. Yuan NJ, Zheng Y, Xie X, Wang Y, Zheng K, Xiong H (2014) Discovering urban functional zones using latent activity trajectories. IEEE Trans Knowl Data Eng 27(3):712–725

11. Kipf TN, Welling M (2016) Variational graph auto-encoders. arXiv preprint arXiv:1611.07308

12. Ding Z, Xu Y, Xu W, Parmar G, Yang Y, Welling M, Tu Z (2020) Guided variational autoencoder for disentanglement learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 7920–7929

13. Zhang H, Xu T, Li H, Zhang S, Wang X, Huang X, Metaxas DN (2017) Stackgan: text to photo-realistic image synthesis with stacked generative adversarial networks. In: Proceedings of the IEEE international conference on computer vision, pp 5907–5915

14. Larsen ABL, Sønderby SK, Larochelle H, Winther O (2016) Autoencoding beyond pixels using a learned similarity metric. In: ICML

15. Yuan J, Zheng Y, Zhang C, Xie W, Xie X, Sun G, Huang Y (2010) T-drive: driving directions based on taxi trajectories. In: Proceedings of the 18th SIGSPATIAL international conference on advances in geographic information systems, pp 99–108

16. Mirza M, Osindero S (2014) Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784

17. Sohn K, Lee H, Yan X (2015) Learning structured output representation using deep conditional generative models. Adv Neural Inf Process Syst 28:3483–3491

18. Radford A, Metz L, Chintala S (2015) Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434

19. Arjovsky M, Chintala S, Bottou L (2017) Wasserstein generative adversarial networks. In: Precup D, Teh YW (eds) Proceedings of the 34th international conference on machine learning. Proceedings of machine learning research, vol 70, pp 214–223

20. Gulrajani I, Ahmed F, Arjovsky M, Dumoulin V, Courville A (2017) Improved training of wasserstein gans. In: Proceedings of the 31st international conference on neural information processing systems, pp 5769–5779

21. Meng T, Jing X, Yan Z, Pedrycz W (2020) A survey on machine learning for data fusion. Inf Fusion 57:115–129

22. Wang D, Wang P, Liu K, Zhou Y, Hughes CE, Fu Y (2021) Reinforced imitative graph representation learning for mobile user profiling: an adversarial training perspective. Proc AAAI Conf Artif Intell 35:4410–4417

23. Liu K, Wang P, Zhang J, Fu Y, Das SK (2018) Modeling the interaction coupling of multi-view spatiotemporal contexts for destination prediction. In: Proceedings of the 2018 SIAM international conference on data mining, pp 171–179. SIAM

24. Wang P, Liu K, Wang D, Fu Y (2021) Measuring urban vibrancy of residential communities using big crowdsourced geotagged data. Front Big Data 4

25. Wang D, Wang P, Zhou J, Sun L, Du B, Fu Y (2020) Defending water treatment networks: Exploiting spatio-temporal effects for cyber attack detection. In: 2020 IEEE international conference on data mining (ICDM), pp 32–41. IEEE

26. Ruthotto L, Haber E. An introduction to deep generative modeling. GAMM-Mitteilungen, 202100008

27. Kobyzev I, Prince S, Brubaker M (2020) Normalizing flows: an introduction and review of current methods. IEEE Trans Pattern Anal Mach Intell

28. Kingma DP, Welling M (2019) An introduction to variational autoencoders. arXiv preprint arXiv:1906.02691

29. Creswell A, White T, Dumoulin V, Arulkumaran K, Sengupta B, Bharath AA (2018) Generative adversarial networks: an overview. IEEE Signal Process Mag 35(1):53–65

30. Kang S, Cho K (2018) Conditional molecular design with deep generative models. J Chem Inf Model 59(1):43–52

31. Chenthamarakshan V, Das P, Padhi I, Strobelt H, Lim KW, Hoover B, Hoffman SC, Mojsilovic A (2020) Target-specific and selective drug design for covid-19 using deep generative models

32. Oliveira V, Pinho P (2010) Evaluation in urban planning: advances and prospects. J Plan Lit 24(4):343–361

33. Naess P (2001) Urban planning and sustainable development. Eur Plan Stud 9(4):503–524

34. Ratcliffe J, Stubbs M (2011) Urban planning and real estate development. Urban Planning and Real Estate Development

35. Fistola R, Papa R (2016) Smart energy in the smart city. Urban planning for a sustainable future
36. Nguyen NH (2020) Simulating the generative process of urban form: an application using opensim. J Plan Educ Res 40(4):393–404
37. Wang D, Fu Y, Liu K, Chen F, Wang P, Lu C-T (2021) Automated urban planning for reimagining city configuration via adversarial learning: quantification, generation, and evaluation. ACM Transactions on Spatial Systems and Algorithms

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.

Dongjie Wang received the BE degree from the Sichuan University, China, 2016, the MS degree from the Southwest Jiaotong University, China, 2017. He is currently working toward the PhD degree at the University of Central Florida (UCF). His research interests include Anomaly Detection, Generative Adversarial Network, Reinforcement Learning and Spatio-temporal Data Mining.

Kunpeng Liu is an Assistant Professor in the Department of Computer Science, Portland State University. Prior to that, he received the Ph.D. degree from the CS Department of the University of Central Florida. He received both his M.E. degree and B.E. degree from the Department of Automation, University of Science and Technology of China (USTC). He has rich research experience in industry research labs, such as Geisinger Medical Research, Microsoft Research Asia, and Nokia Bell Labs. He has published in refereed journals and conference proceedings, such as IEEE TKDE, ACM SIGKDD, IEEE ICDM, AAAI, and IJCAI. He has received a Best Paper Runner-up Award from IEEE ICDM 2021.
Yanyong Huang received his double Ph.D. degrees from the Southwest Jiaotong University and FernUniversität in Hagen in 2018 and 2020, respectively. He is currently a Professor in the School of Statistics, Southwestern University of Finance and Economics. His research interests include data mining, machine learning and granular computing. He has published over 30+ research papers in international journals and conferences.

Leilei Sun received the B.S., M.S. and Ph.D. degree from Dalian University of Technology in 2009, 2012 and 2017, respectively. He is currently an associate professor with the State Key Laboratory of Software Development Environment (SKLSDE) in School of Computer Science, Beihang University. He was a postdoctoral research fellow from 2017 to 2019 in Tsinghua University. His research interests include machine learning and data mining.

Bowen Du (Member, IEEE) received the Ph.D. degree in computer science and engineering from Beihang University, Beijing, China, in 2013. He is currently a Professor with the State Key Laboratory of Software Development Environment, Beihang University. His research interests include AI, multi-source data fusion, and data mining.
Yanjie Fu is an Assistant Professor in the Department of Computer Science at the University of Central Florida. He received his Ph.D. degree from Rutgers, the State University of New Jersey in 2016, the B.E. degree from University of Science and Technology of China in 2008, and the M.E. degree from Chinese Academy of Sciences in 2011. His research interests include Data Mining and Big Data Analytics.