Author/s:
Sun, F;Chang, Y;Tanin, E;Karunasekera, S;Qi, J

Title:
FlexiReg: Flexible Urban Region Representation Learning

Date:
2025

Citation:
Sun, F., Chang, Y., Tanin, E., Karunasekera, S. & Qi, J. (2025). FlexiReg: Flexible Urban Region Representation Learning. KDD '25: Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2, 2, pp.2702-2713. Association for Computing Machinery. https://doi.org/10.1145/3711896.3736965.

Persistent Link:
https://hdl.handle.net/11343/361936

FlexiReg: Flexible Urban Region Representation Learning

Fengze Sun
School of Computing and Information System, University of Melbourne
Melbourne, Australia
fengzes@student.unimelb.edu.au

Yanchuan Chang
School of Computing and Information System, University of Melbourne
Melbourne, Australia
yanchuan.chang@unimelb.edu.au

Egemen Tanin
School of Computing and Information System, University of Melbourne
Melbourne, Australia
etanin@.unimelb.edu.au

Shanika Karunasekera
School of Computing and Information System, University of Melbourne
Melbourne, Australia
karus@.unimelb.edu.au

Jianzhong Qi*
School of Computing and Information System, University of Melbourne
Melbourne, Australia
jianzhong.qi@unimelb.edu.au

Abstract

The increasing availability of urban data offers new opportunities for learning region representations, which can be used as input to machine learning models for downstream tasks such as check-in or crime prediction. While existing solutions have produced promising results, an issue is their fixed formation of regions and fixed input region features, which may not suit the needs of different downstream tasks. To address this limitation, we propose a model named FlexiReg for urban region representation learning that is flexible with both the formation of urban regions and the input region features. FlexiReg is based on a spatial grid partitioning over the spatial area of interest. It learns representations for the grid cells, leveraging publicly accessible data, including POI, land use, satellite imagery, and street view imagery. We propose adaptive aggregation to fuse the cell representations and prompt learning techniques to tailor the representations towards different tasks, addressing the needs of varying formations of urban regions and downstream tasks. Extensive experiments on five real-world datasets demonstrate that FlexiReg outperforms state-of-the-art models by up to 202% in term of the accuracy of four diverse downstream tasks using the produced urban region representations.

CCS Concepts

• **Information systems** → **Location based services**; **Data mining**.

Keywords

Urban region representation, multi-modal learning

ACM Reference Format:
Fengze Sun, Yanchuan Chang, Egemen Tanin, Shanika Karunasekera, and Jianzhong Qi. 2025. FlexiReg: Flexible Urban Region Representation Learning. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '25), August 3–7, 2025, Toronto, ON, Canada.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3711896.3736965

*Corresponding author.

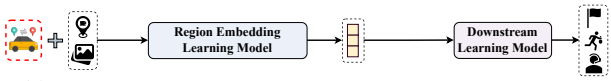
KDD '25, Toronto, ON, Canada

ACM ISBN 979-8-4007-1454-2/2025/08
https://doi.org/10.1145/3711896.3736965

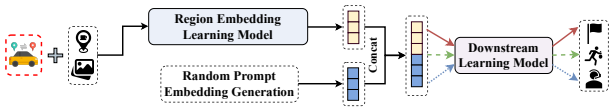
KDD Availability Link:
The source code of this paper has been made publicly available at https://doi.org/10.5281/zenodo.15545697.

1 Introduction

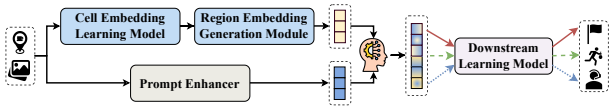
Urban region representation learning has become increasingly popular in the community of urban computing [40, 43, 49, 51, 57], which aims to transform urban regions into vector representations, known as embeddings. These embeddings entail valuable insights on urban structures and properties, facilitating effective urban planning and management, such as designating functionalities for new development areas. They are also useful in various tasks related to daily life, such as crime count prediction [15, 31, 41, 45, 59].



(a) Representation learning for predefined region partitions



(b) Representation learning with prompting



(c) Ours: allowing flexible partitions and task-based prompting

Figure 1: Region representation learning schemes.

Recently, the use of multi-modal data for learning urban region representations has gained attention. A critical aspect of this process is the selection of input data features, often referred to as region features, where each type of features depicts a region from a distinct *view*. Existing studies commonly utilize human mobility data [7, 8, 15, 19, 31, 36, 38, 41, 45, 47, 48, 52–55] and POIs [8, 15, 19, 31, 45, 52–55]. Among these, the ones using human mobility data often demonstrate superior performance, as such data offer critical insights into movement pattern and hence functional relationships between regions.

Moreover, existing studies typically follow a two-stage process, as shown in Fig. 1a, with a generic region embedding learning stage

and a downstream task learning stage. The first stage learns the embeddings for a set of predefined regions with all input region feature data, while the second stage trains a (separate) machine learning model for downstream tasks, e.g., crime count prediction, using the embeddings as the model input.

However, there are three limitations in the existing studies:

Limitation 1. Existing methods heavily depend on mobility data and underutilize publicly accessible data. Human mobility data plays a critical role in learning effective region embeddings [15, 31, 41, 52, 53]. However, their limited availability, particularly in underdeveloped regions, together with privacy issues, prevent models using such data from a wider adoption. Recently, several studies [14, 42, 46, 48] leverage features from publicly accessible data (e.g., POIs from OpenStreetMap) to enhance model applicability. However, these studies suffer from the effectiveness of the learning models. Their learned embeddings have reported lower accuracy for downstream tasks comparing with those learned by the mobility-based models.

Limitation 2. Existing studies lack the flexibility to utilize different urban features for different downstream tasks. Existing studies simply use all input features to learn region embeddings together, without considering their relevance to specific downstream tasks. During the downstream task learning stage, most studies directly use the region embeddings for downstream tasks without any adaptation for task-specific needs, as shown in Fig. 1a. Prompt learning presents opportunities to incorporate task-specific adaptations into the region embeddings. HREP [59] first attempted this idea (see Fig. 1b). It simply applies random prompt embeddings for downstream tasks, which fails to capture the correlation between features and downstream tasks.

Limitation 3. Existing studies lack the flexibility to adapt to different formation of regions Existing studies typically rely on a single, predefined region formation for all downstream tasks, making it difficult to accommodate different downstream tasks with different region formations (or analytical tasks to explore different region formations). For example, population estimation may need to be done at the census tract level, whereas transportation planning concerns more on traffic-related region partitions. As a third example, real estate investors or house seekers may be more interested in regions defined by school zones. Existing studies will need to compute a different set of embeddings for each of these application scenarios, which is costly and less flexible.

We summarize existing works for the issues above in Table 1.

Table 1: Comparison between Region Embedding Learning Methods

Models	Publicly accessible data	Prompting	Adaptive region embeddings
[5, 7, 8, 15, 31, 36] [38, 41, 45, 47, 48, 52–55]			
[59]		✓	
[1, 11, 12, 14, 39, 42, 46]	✓		
[2, 32]	✓		✓
FlexiReg (ours)	✔	✔	✔

To address the issues above, we propose FlexiReg (Fig. 1c), a Flexible model for urban Region representation learning. It takes a three-stage learning process that enables a flexible use of urban features to generate region embeddings tailored for different region formations and downstream tasks. FlexiReg is flexible in all three aspects discussed above:

(1) It leverages urban region features from publicly accessible data, including POIs, land use data, satellite imagery, and street view imagery, which have wider availability than human mobility data. To effectively exploit these features, we partition an area of interest into finer-grained spatial partition units using a hexagonal grid. We propose a novel *multimodal grid cell embedding learning* (GridLearner) module and an environment context-based contrastive learning technique to capture distinctive urban patterns from each type of input features and spatial correlations between different types of features, respectively (addressing Limitation 1).

(2) It takes a three-stage learning process. The first two stages learn fine-grained grid cell embeddings and aggregate them into region embeddings, respectively. We propose an *adaptive region embedding generation* (AdaRegionGen) module for the aggregation stage, which weighs the embeddings for the cells by their overlapping areas with a region. Notably, this aggregation process is flexible, allowing grid cell embeddings to be combined into region embeddings regardless of the region partitioning methods (addressing Limitation 3).

(2) It has a prompt learning process for its third stage, which enables it to flexibly utilize different types of features for different downstream tasks. We propose a novel *prompt enhancer* (PromptEnhancer) module to tailor region embeddings for downstream tasks by integrating textual and street-view imagery features. To capture task-relevant information, PromptEnhancer consists of a *text-region alignment* (T-RAlign) module and a *street view-region alignment* (SV-RAlign) module. T-RAlign incorporates task-specific geographic knowledge into region embeddings using dimension-wise similarity, while SV-RAlign extracts task-relevant visual features through adapted attention mechanisms (addressing Limitation 2).

To summarize, this paper makes the following contributions:

(1) We propose a model named FlexiReg to generate effective and flexible region representations that can be adapted for different downstream tasks by leveraging publicly accessible data.

(2) We propose a multimodal grid cell embedding learning module, followed by an adaptive region embedding generation module to generate region embeddings when a set of regions is given. These two modules capture urban patterns within grid cells and model their correlations to enhance region representation learning.

(3) We propose a prompt enhancer module to tailor region embeddings for downstream tasks by seamlessly integrating task-relevant information from additional features into the embeddings.

(4) We conduct extensive experiments to evaluate FlexiReg on five real-world datasets. The results show that FlexiReg outperforms all competitors, including those utilizing publicly accessible data and those based on human mobility data, across four downstream tasks (crime, check-in, service call, and population count predictions), by up to 202% in term of accuracy.

2 Solution Overview

Problem statement. Given a spatial area of interest with publicly accessible features (detailed in Section 3.1.2) and a set of non-overlapping regions R in this area, we aim to learn an embedding function $f : r_i \rightarrow \mathbf{h}_i$ that maps a region $r_i \in \mathcal{R}$ to a d-dimensional

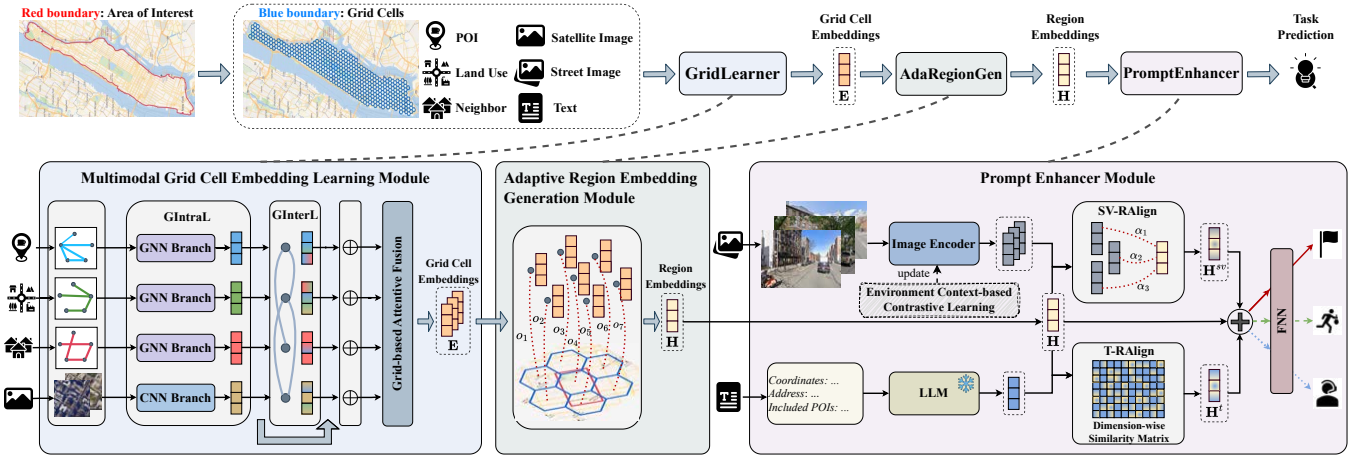


Figure 2: FlexiReg model overview. The model processes a set C of grid cells, each with associated features, through a three-stages learning process to generate flexible region embeddings to accommodate the needs of different downstream tasks with different region formations.: (1) GridLearner computes fine-grained grid cell embeddings E. (2) AdaRegionGen aggregates fine-grained grid cell embeddings to produce region embeddings H, given an input partition of regions. (3) PromptEnhancer further tailors the region embeddings with additional features as guided by given downstream tasks.

vector $\mathbf{h}_i$. The learned embeddings are expected to be applicable in different downstream tasks. Then, for each downstream task (e.g., crime count prediction), we learn a prediction function $g : \mathbf{h}_i \rightarrow y_i$, where $y_i \in \mathbb{R}$ is typically a numerical indicator for the task.

Model overview. Fig. 2 shows the overall structure of our model FlexiReg, which consists of three main learning stages. (1) FlexiReg takes a set of grid cells as the input. The grid cells come from a fine-grained partitioning over the spatial area of interest that we perform as part of data preparation (Section 3.1). FlexiReg learns the embeddings of grid cells across different features through a Multimodal Grid Cell Embedding Learning module (GridLearner, Section 3.2). (2) Then, the Adaptive Region Embedding Generation module (AdaRegionGen) aggregates the fine-grained cell embeddings to generate region embeddings for the input regions (Section 3.3). Starting from the fine-grained cell embeddings allows FlexiReg to flexibly adapt to different sets of regions which may come from different downstream tasks (or analytical tasks to explore different way to form regions). (3) Finally, the Prompt Enhancer module (PromptEnhancer) refines the generic region embeddings with extra features, guided by given tasks (Section 3.4).

3 Proposed Model

This section details the FlexiReg model. We summarize the frequently used symbols in Table 2.

3.1 Data Preparation

3.1.1 Grid Cell Construction. We partition the input spatial area of interest into a set C of grid cells, where c_i denotes the i-th cell. Here, the grid cells are supposed to be finer-grained spatial partitions than the regions, allowing for flexible formations of regions as required by downstream tasks later on. We partition the area using a hexagonal grid, as illustrated in Fig. 2 (the blue grid on the map shown at the top left), which provides several advantages. First, cells (which are of a small size) mitigate spatial heterogeneity by enabling localized feature learning, allowing FlexiReg to effectively capture local variations within a region. Second, hexagonal cells in particular offer more uniform coverage than cells of other shapes (e.g., squares), as each cell is surrounded by six equidistant neighbors. Third, hexagonal cells are easier to approximate natural boundaries, improving spatial coverage and making them ideal for regions with irregular boundaries [4, 37].

Table 2: Frequently Used Symbols

Symbol	Description
S	A spatial area of interest
R	A set of regions (non-overlapping space partitions)
n	The number of regions
C	A set of spatial grid cells (basic space partition units)
m	The number of grid cells
$\mathbf{p}_i$ $\mathbf{l}_i$ $\mathbf{gn}_i$ $\mathbf{si}_i$, $\mathbf{sv}_i$, $\mathbf{t}_i$	POI, land use, geographic neighbor, satellite imagery, street view imagery, and textual feature of cell c_i
$\mathbf{E}$	The embeddings of grid cells
$\mathbf{H}$	Adaptive region embeddings

3.1.2 Feature Preparation. We use six types of features for each cell, which are all publicly accessible, with full details in Appendix A.

POI features. For each cell, we count the number of POIs that belong to one of 15 POI categories from OpenStreetMap [23] as the POI feature. We denote the POI feature of cell c_i as $\mathbf{p}_i \in \mathbb{R}^{15}$.

Land use features. Similar to POI features, we count the numbers of zones that belong to 20 different land use types within a cell. The land use feature of cell c_i is denoted as $\mathbf{l}_i \in \mathbb{R}^{20}$.

Geographic neighbor features. This feature indicates the adjacency relationships between cells. We use $\mathbf{gn}_i \in \mathbb{R}^6$ to denote a vector of the six direct neighboring cells of c_i.

Satellite imagery features. Satellite images capture rich coarse-grained urban patterns. We use $\mathbf{si_i} \in \mathbb{R}^{H \times W \times 3}$ to denote the satellite

imagery feature of c_i, where H and W denote the height and the width of the satellite image of c_i, which has a rectangular shape just covering the cell.

Street view imagery features. Street view images capture finer-grained urban patterns. We use $\mathbf{sv}_i = \{\mathbf{sv}_{i,1}, \mathbf{sv}_{i,2}, \cdots\}$ to denote the set of street view images captured within the area of c_i, where each image is also in the shape of $\mathbb{R}^{H\times W\times 3}$. Note that different cells may have different numbers of street view images.

Textual features. We generate textual features for each cell by describing them from different aspects in text, including geometric properties, addresses, and POIs within them, to enable learning the urban features from a semantic perspective. We denote the textual feature of c_i as $\mathbf{t}_i \in \mathbb{R}^S$, where S refers to the maximum length of a textual description. We elaborate this feature in Section 3.4.1.

We use the first four features to learn cell embeddings, capturing their functionality, spatial structure, correlations, and urban patterns, while the last two features will be used later to tailor region embeddings for downstream tasks. POIs and land use categories reflect the functional roles of urban areas; geographic neighbors reflect spatial relationships; satellite images provide visual insights into the physical layout and urban patterns. These features are generic for urban representation learning. In contrast, textual data captures nuanced details such as the presence of a large number of "entertainment venues", which may correlate with and suit a common downstream task, crime prediction. Street view images offer ground-level context, such as building density, which is crucial for tasks such as population prediction. These features are more suitable for task-relevant adaptation of the region embeddings.

3.2 Multimodal Grid Cell Embedding Learning

The GridLearner module learns cell embeddings through four *views* each corresponding to a type of input features. It learns correlations between views and between cells, forming robust embeddings.

3.2.1 Grid-based Intra-view Feature Learning. We leverage GNNs for POI, land use, and geographic neighbor features, and a CNN for satellite imagary features, to suit the different types of features.

GNN branches. We construct feature-aware grid graphs on the POI (p), land use (l), and geographic neighbor (gn) features of cells, separately, to help capture the correlation between cells based on such features. Let $\mathcal{G}^{\mathcal{X}} = (\mathcal{V}, \mathcal{E}, \mathbf{A}^{\mathcal{X}})$ be a grid graph based on a specific feature $\mathcal{X}$, where $\mathcal{X} \in \{p, l, gn\}$. Here, $\mathcal{V} = \{c_1, c2, \cdots, c_m\}$ denotes the set of m vertices (i.e., m grid cells in the input area of interest); $\mathcal{E}$ denotes the set of edges between vertices; and $\mathbf{A}^{\mathcal{X}}$ is a weighted adjacency matrix, where $\mathbf{A}^{\mathcal{X}}_{i,j}$ is the cosine similarity between feature vectors of c_i and c_j.

Once the feature-aware grid graphs are constructed, we employ Graph Attention Networks (GAT) [35] to produce grid cell embeddings on each view of features. GAT stacks multiple graph attention layers to compute the correlation between vertices and aggregate vertex embeddings based on correlation scores. For a given graph attention layer at the g-th layer, its process is as follows. We omit the superscript $\mathcal{X}$ hereafter for simplicity when the context is clear.

$$\alpha^g_{ij} = \text{Softmax}\left(\sigma\left(\mathbf{a}^\intercal\left(\mathbf{W}\mathbf{z}^g_i || \mathbf{W}\mathbf{z}^g_j || \mathbf{w}\mathbf{A}_{i,j}\right)\right)\right), \tag{1}$$

$$\mathbf{z}^{g+1}_i = \sigma\Big(\sum_{j\in[1,m]} \alpha^g_{ij}\mathbf{z}^g_i\Big). \tag{2}$$

Here, a^g_{ij} denotes the correlation (i.e., the normalized correlation score) between c_i and c_j w.r.t. their embeddings $\mathbf{z}^g_i$ and $\mathbf{z}^g_j$ in the g-th GAT layer. We use $\mathbf{a} \in \mathbb{R}^{3d}$, $\mathbf{W} \in \mathbb{R}^{d\times d}$, and $\mathbf{w} \in \mathbb{R}^d$ to denote learnable parameters, and σ is the LeakyReLU activation function. The input to the 1-st layer, $\mathbf{z}^0_i$, is obtained by random initialization.

We apply three GATs to POI, land use, and geographic neighbor features separately to obtain representations for each feature view, denoted as $\mathbf{Z}^p$, $\mathbf{Z}^l$, and $\mathbf{Z}^{gn}$, respectively, each in the shape of $\mathbb{R}^{m\times d}$.

CNN Branch. We employ ResNet [10] followed by an MLP to encode the satellite images of grid cells into embeddings:

$$\mathbf{z}^{si}_i = \text{MLP}(\text{ResNet}(\mathbf{si}_i)), \tag{3}$$

where $\mathbf{z}^{si}_i$ denotes the embedding of the satellite image of c_i, and the MLP is an additional projection layer. Further, $\mathbf{Z}^{si} \in \mathbb{R}^{m\times d}$ denotes the embeddings of cells on satellite imagery features.

3.2.2 Grid-based Inter-view Feature Learning. Next, we learn the correlation between different feature views for each grid cell by applying a one-layer self-attention [34] on $\mathbf{Z} \in \mathbb{R}^{4\times m\times d}$, which is obtained by stacking $\mathbf{Z}^p$, $\mathbf{Z}^l$, $\mathbf{Z}^{gn}$ and $\mathbf{Z}^{si}$. The output of the self-attention layer is then added with $\mathbf{Z}$, weighted by a learnable parameter β, as detailed in our technical report [30].

3.2.3 Grid-based Dual-Feature Attentive Fusion. We adopt the dual-feature attentive fusion module (DAFusion) from HAFusion [31] to further refine cell embeddings $\mathbf{Z}$. DAFusion first fuses the representations among different views into an adaptive view representation for the cells. Then, it fuses the representations among cells. We denote the output of this module as $\mathbf{E} \in \mathbb{R}^{m\times d}$. Due to space limit, we elaborate this module in our technical report [30].

3.2.4 Module Training. We leverage a multi-task learning objective $\mathcal{L}$ to learn the cell representations, which consists of four sub-objective functions, each corresponding to a type of features.

Given embeddings $\mathbf{E}$, we first generate feature-oriented cell embeddings $\mathbf{E}^{\mathcal{X}}$ for feature $\mathcal{X}$ (now $\mathcal{X}$ denotes one of the four types of input features above) by adopting an MLP, which can be represented as $\mathbf{E}^{\mathcal{X}} = \text{MLP}_{\mathcal{X}}(\mathbf{E})$. As a result, we obtain four types of feature-oriented embeddings $\mathbf{E}^{\mathcal{X}}$, each using a different objective.

POI-oriented and land use-oriented objectives $\mathcal{L}^p$ and $\mathcal{L}^l$. We use *graph reconstruction* as the learning objective to reconstruct the POI adjacency matrix $\mathbf{A}^p$ and land use adjacency matrix $\mathbf{A}^l$ from their feature-oriented embeddings $\mathbf{E}^p$ and $\mathbf{E}^l$, respectively. These objectives have been used before [31] and are repeated here.

Geographic neighbor-oriented objective $\mathcal{L}^{gn}$. Inspired by the First Law of Geography [33], emphasizing that spatially close cells are likely to share similar embeddings, we use a triplet loss [28] to learn geographic neighbor-oriented embeddings, as detailed in our technical report [30].

Satellite image-oriented objective $\mathcal{L}^{si}$. We model the satellite image-oriented objective as predicting the number of POIs in a cell using the corresponding satellite image, where the ground-truth POI count can be obtained from the POI feature. Such a training objective guides our model to focus on spatial characteristics when learning embeddings from satellite images. The details of this objective can be found in Appendix B.

Finally, the overall objective function is derived by summing up the feature-oriented objective functions as follows:

$$\mathcal{L} = \mathcal{L}^{p} + \mathcal{L}^{\mathcal{L}} + \mathcal{L}^{gn} + \mathcal{L}^{si} \tag{4}$$

3.3 Adaptive Region Embedding Generation

Next, we generate the region embeddings $\mathbf{H} = \{h_i\}_{i=1}^{n}$ by aggregating the embeddings $\mathbf{E}$ of grid cells corresponding to input regions based on their spatial locations. Given a region r_j, we first find a set of grid cells, denoted as $C_{r_j} = \{c_1, \cdots, c_i, \cdots\}$, where each cell c_i either spatially intersects with or is contained within r_j. In addition, we compute the overlapping coefficient between r_j and each $c_i \in C_{r_j}$ based on their areas, which indicates the relative importance of c_i to r_j, as follows:

$$o_{r_j \cap c_i} = \frac{\text{Area}(r_j \cap c_i)}{\text{Area}(c_i)}, \tag{5}$$

where $\cap$ denotes the spatial intersection, and $\text{Area}(\cdot)$ computes the size of a given spatial area. Then, we fuse the cell embeddings with their overlapping coefficients to region r_j and generate the region embedding $\mathbf{h}_j$:

$$\mathbf{h}_j = \sum_{c_i \in C_{r_j}} o_{r_j \cap c_i} \cdot \mathbf{e}_i, \tag{6}$$

Here, $\mathbf{e}_i \in \mathbf{E}$ is the cell embedding of c_i.

3.4 Prompt Enhancer for Task Learning

We propose a *prompt enhancer* (PromptEnhancer) based on prompt learning, which refines the general region embeddings learned above for better adaptability across downstream tasks. PromptEnhancer integrates complementary features–textual descriptions and street view images–to provide rich contextual information, aligning region embeddings with task-specific demands for more accurate predictions. It consists of two modules: the *text-region encoding* module, which encodes semantic insights from textual descriptions, and the *street view-region encoding* module, which incorporates ground-level visual details from street view images.

3.4.1 Text-Region Encoding. The text-region encoding module consists of three main steps: cell description generation, region embedding generation, and text-region embedding alignment.

(1) Cell description generation. We developed a textual description template for cells based on the POI information, which serves as the prompt to effectively extract geographic knowledge from LLMs. The template includes the following key information of a cell: (1) *geometric properties* describing the shape and size of a grid cell. (2) *address* referring to the detailed street address of the POI located at the center of a cell (3) *POI information* including the categories and numbers of POIs within a cell. A sample textual description generated for a cell can be found in Appendix C.

(2) Region embedding generation. After obtaining the cell descriptions, we generate their embeddings using a pre-trained parameter-frozen LLM (we use Llama 3 8B Instruct in our experiments) [18]. The input textual descriptions of all cells $\mathbf{T} \in \mathbb{R}^{m \times S}$ (S refers to maximum length of a textual cell description) are first tokenized and then processed into embeddings. We use the last token embeddings from the last hidden layer of the LLM as the final text embeddings of grid cells, denoted as $\mathbf{E}^t \in \mathbb{R}^{m \times d_{llm}}$, since the last tokens capture information from all preceding tokens [16]. Here, $d_{llm} = 4096$ is the dimensionality of the text embeddings. Note that using the frozen LLM parameters has the benefit of preserving the intrinsic geographic knowledge learned by the LLM.

We use the same cell-to-region embedding aggregation approach as described in Section 3.3 to obtain the region embeddings from textual features, i.e., following Equations 5 and 6. The textual embeddings of regions are denoted as $\mathbf{H}^t \in \mathbb{R}^{n \times d_{llm}}$.

(3) Text-region embedding alignment. To integrate the semantic insights from textual embeddings with the region embeddings learned earlier, we design a *text-region alignment* (T-RAlign) module using dimension-wise similarity computation. This module extracts task-relevant geographic knowledge from the textual embeddings, with guidance given by a downstream task.

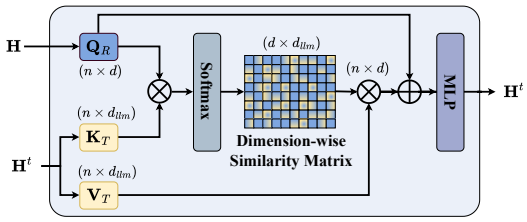


Figure 3: Text-region alignment module

As shown in Fig. 3, we employ linear transformations on $\mathbf{H}^t$ and $\mathbf{H}$ to form three projected matrices in latent space: $\mathbf{Q}_R \in \mathbb{R}^{n \times d} = \mathbf{W}_Q\mathbf{H}$, $\mathbf{K}_T \in \mathbb{R}^{n \times d_{llm}} = \mathbf{W}_K\mathbf{H}^t$, $\mathbf{V}_T \in \mathbb{R}^{n \times d_{llm}} = \mathbf{W}_V\mathbf{H}^t$. Here, $\mathbf{W}_Q$, $\mathbf{W}_K$, and $\mathbf{W}_V$ are learned parameters. Next, we compute the dimension-wise similarity matrix as follows:

$$\mathbf{M}^t = \text{Softmax}\left(\left(\mathbf{W}_Q\mathbf{H}\right)^T \left(\mathbf{W}_K\mathbf{H}^t\right)\right), \tag{7}$$

where $\mathbf{M}^t \in \mathbb{R}^{d \times d_{llm}}$ captures the similarity between the dimensions of the two embeddings.

Then, we compute the retrieved textual embeddings by applying dimension-wise feature aggregation via matrix multiplication between $\mathbf{M}^t$ and $\mathbf{V}_T$. The embeddings are then combined with the input region embeddings $\mathbf{H}$ using element-wise addition. Finally, the result is passed through an MLP to update the output embeddings $\mathbf{H}^t \in \mathbb{R}^{n \times d}$. Formally, this process is expressed as:

$$\mathbf{H}^t = \text{MLP}\left(\left(\left(\mathbf{W}_V\mathbf{H}^t{\mathbf{M}^t}^T\right) + \mathbf{H}\right)\right). \tag{8}$$

Through this text-region embedding alignment, we transfer the geographic knowledge encoded with the LLM into the region embeddings, thereby enhancing FlexiReg's overall performance.

3.4.2 Street View-Region Encoding. The street view-region encoding module contains two main steps: street view image embedding learning, and street view-region embedding alignment.

(1) Street view image embedding learning. We propose an environment context-based contrastive learning approach to learn the representation of street view images as illustrated in Fig. 4. Motivated by the observation that street view images from the same cell exhibit strong correlations, we aim to maximize the similarity between a street view image and its corresponding cell's environmental context while minimizing its similarity with unrelated cells. This ensures the learned embeddings to effectively capture distinctive environmental patterns, spatial correlations between images and cells, and spatial correlations among the images themselves.

Specifically, given cell c_i and its corresponding street view images $\mathbf{sv}_i = \{\mathbf{sv}_{i,1}, \mathbf{sv}_{i,2}, \cdots\}$, we use ResNet as the image encoder to extract the initial visual embedding $\mathbf{u}_{i,j}$ for each street view image:

$$\mathbf{u}_{i,j} = \text{ResNet}(\mathbf{sv}_{i,j}). \tag{9}$$

Next, we average the visual embeddings for all street view images of the same cell c_i, representing the environmental context embedding of c_i, denoted as $\mathbf{v}_i$:

$$\mathbf{v}_i = \frac{1}{|\mathbf{sv}_i|} \sum_{j=1}^{|\mathbf{sv}_i|} \mathbf{u}_{i,j}, \tag{10}$$

where $|\mathbf{sv}_i|$ is the number of images associated with c_i.

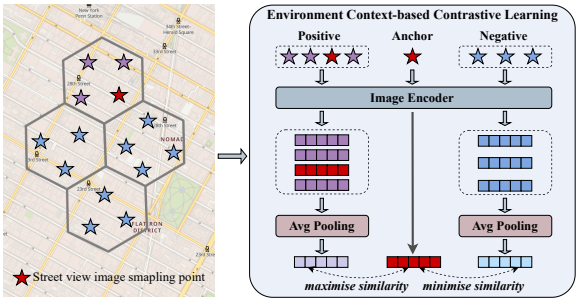


Figure 4: Street view image embedding learning

Then, we adopt the InfoNCE [22] loss as the objective function to optimize the environment context-based image encoder. For a target street view image, the environmental context of the cell it belongs to serves as the positive sample, while the environmental contexts of other grid cells are treated as negative samples. The objective function is defined as follows:

$$\mathcal{L}^{sv} = -\frac{1}{m} \sum_{i}^{m} \sum_{j=i}^{\mathbf{sv}_i} \log \left(\frac{\exp\left(\frac{\mathbf{u}_{i,j}^T \mathbf{v}_i}{\tau}\right)}{\sum_k^m \exp\left(\frac{\mathbf{u}_{i,j}^T \mathbf{v}_k}{\tau}\right)} \right). \tag{11}$$

Recall that m is the total number of cells, and τ is a temperature parameter set to 0.5 in our experiment. We train the environment context-based image encoder by minimizing $\mathcal{L}^{sv}$ to generate the street view image embeddings.

(2) Street view-region embedding alignment. After obtaining the street view image embeddings, we reassign them to their corresponding regions based on the geographical locations of the images. To handle the variability in the number of street view images per region, which can affect the subsequent alignment process, we standardize the data by randomly selecting a fixed number, x, of images for each region ($x = 64$ in our experiments). For a given region r_i, the corresponding street view image embeddings are organized into a matrix $\mathbf{U}_i \in \mathbb{R}^{x \times d_{img}}$, where $d_{img} = 768$.

To effectively integrate the street view image embeddings with the region embeddings learned earlier, we introduce a *street view-region alignment* (SV-RAlign) module. This module extracts task-relevant ground-level visual features from the street view image embeddings, with guidance given by a downstream task.

To enable the adaptive selection of relavant visual information, we employ a cross-attention layer. Given region embedding $\mathbf{h}_i$ and street view image embedding $\mathbf{U}_i$ of region r_i, we define the query matrix $\mathbf{Q}_i = \mathbf{h}_i \mathbf{W}_Q$, key matrix $\mathbf{K}_i = \mathbf{U}_i \mathbf{W}_K$, and value matrix $\mathbf{V}_i = \mathbf{U}_i \mathbf{W}_V$, where $\mathbf{W}_Q \in \mathbb{R}^{d \times d_{proj}}$, $\mathbf{W}_K$ and $\mathbf{W}_V \in \mathbb{R}^{d_{img} \times d_{proj}}$, and d_{proj} denotes the dimension of the projected matrices in latent spaces, set to 256 in our experiments. Then, we use the cross-attention operation followed by an MLP to generate the output embedding for region r_i. Formally, the street view-region embedding $\mathbf{h}_i^{sv} \in \mathbf{H}^{sv}$ is computed as:

$$\mathbf{h}_i^{sv} = \text{MLP}\left(\text{Softmax}\left(\frac{\mathbf{Q}_i \mathbf{K}_i{}^T}{\sqrt{d_{proj}}}\right)\mathbf{V}_i\right), \tag{12}$$

3.4.3 Model Training. After obtaining the region embedding $\mathbf{H}$, the text-region embedding $\mathbf{H}^t$, and the street view-region embedding $\mathbf{H}^{sv}$, we construct the final regin embeddings $\hat{\mathbf{h}}_i \in \hat{\mathbf{H}}$ as $\hat{\mathbf{h}}_i = \mathbf{h}_i \,||\, \mathbf{h}_i^t \,||\, \mathbf{h}_i^{sv}$, where '||' denotes concatenation. We use a feedforward neural network (FNN) for a given downstream (prediction) task, formulated as $\hat{y}_i = \text{FFN}(\hat{\mathbf{h}}_i)$, where $\hat{y}_i$ is the prediction output for region r_i. To optimize the PromptEnhancer module, we adopt the mean squared error loss:

$$\mathcal{L}_{pe} = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2, \tag{13}$$

where y_i is the ground truth for r_i, and n is the number of regions.

4 Experiments

We run experiments to verify: (**Q1**) the embedding quality of our FlexiReg model as compared with the state-of-the-art (SOTA) models on four downstream tasks, (**Q2**) the applicability of FlexiReg across diverse geographic regions, (**Q3**) the adaptability of our cell embeddings to different region formations, (**Q4**) the impact of our model components and input features, (**Q5**) the applicability of FlexiReg to areas of different urban environments (**Q6**) the impact of the grid cell design and key hyper-parameters.

4.1 Experimental Settings

Dataset. We use real data from five cities across the globe: New York City (**NYC**) [20], Chicago (**CHI**) [6], San Francisco (**SF**) [26], Singapore City (**SG**) [29], and Lisbon (**LX**) [17]. We collect data on region division, POI, land use, satellite images, street view images, check-in, and population. Additionally, crime and service call data are collected for NYC, CHI, and SF (unavailable for the other cities). More details about the collected data are in our technical report [30].

Competitors. We compare with models from two categories. The first category uses a subset of the publicly accessible data: **RegionDCL** [14], **UrbanCLIP** [46], and **CityFM** [2] (SOTA). The second category uses human mobility data, which has restricted availability: **MVURE** [53], **MGFN** [41], **HREP** [59], **ReCP** [15], and **HAFusion** [31] (SOTA). This latter category does not apply to SG and LX due to lack of data. A description of these models can be found in our technical report [30]. Implementation details of these models and our FlexiReg can be found in Appendix D.

Evaluation procedure. We use each representation learning model to generate region embeddings for each city separately. The embeddings then serve as input to machine learning models (i.e., downstream models) for downstream tasks. We use four downstream prediction tasks following the baseline models [14, 31]: crime, check-in, service call, and population counts. Since these tasks are regression-based, we employ a ridge regression model for each task, with ten-fold cross-validation.

We evaluate the representation learning models through the downstream models in mean absolute error (**MAE**), root mean square error (**RMSE**), and coefficient of determination (R^2).

4.2 Overall Results (Q1)

Table 3 reports overall model accuracy. Here, we only report R^2 for conciseness, as the performance in MAE and RMSE resembles (same below). Full results can be found in our technical report [30].

Table 3: Overall Prediction Accuracy Results ('↑' indicates that large values are preferred. The best results are in boldface, and the second-best results are underlined. Same for the tables below.)

New York City	Crime	Check-in	Service Call	Population
	R^2 ↑	R^2 ↑	R^2 ↑	R^2 ↑
MVURE [53]	0.591 ± 0.016	0.627 ± 0.019	0.367 ± 0.027	0.545 ± 0.008
MGFN [41]	0.630 ± 0.020	0.690 ± 0.040	0.303 ± 0.069	0.509 ± 0.026
HREP [59]	0.680 ± 0.014	0.703 ± 0.021	0.398 ± 0.021	0.571 ± 0.021
ReCP [15]	0.459 ± 0.022	0.761 ± 0.029	0.356 ± 0.029	0.322 ± 0.047
HAFusion [31]	0.734 ± 0.015	0.844 ± 0.012	0.493 ± 0.014	0.616 ± 0.019
RegionDCL [14]	0.251 ± 0.026	0.471 ± 0.023	0.103 ± 0.026	0.198 ± 0.019
UrbanCLIP [46]	0.267 ± 0.012	0.458 ± 0.005	0.232 ± 0.005	0.276 ± 0.002
CityFM [2]	0.315 ± 0.010	0.471 ± 0.011	0.117 ± 0.013	0.261 ± 0.002
FlexiReg	**0.789 ± 0.009**	**0.876 ± 0.006**	**0.601 ± 0.021**	**0.715 ± 0.010**
Improvement	**7.5%**	**3.8%**	**21.9%**	**16.1%**

Chicago	Crime	Check-in	Service Call	Population
	R^2 ↑	R^2 ↑	R^2 ↑	R^2 ↑
MVURE [53]	0.461 ± 0.062	0.656 ± 0.029	0.441 ± 0.050	0.313 ± 0.043
MGFN [41]	0.386 ± 0.047	0.817 ± 0.011	0.329 ± 0.077	0.359 ± 0.054
HREP [59]	0.578 ± 0.041	0.664 ± 0.017	0.468 ± 0.022	0.447 ± 0.061
ReCP [15]	0.534 ± 0.057	0.804 ± 0.045	0.284 ± 0.076	0.325 ± 0.044
HAFusion [31]	0.631 ± 0.036	0.870 ± 0.010	0.613 ± 0.067	0.544 ± 0.035
RegionDCL [14]	0.179 ± 0.053	0.402 ± 0.042	0.445 ± 0.041	0.190 ± 0.032
UrbanCLIP [46]	0.416 ± 0.006	0.186 ± 0.024	0.491 ± 0.003	0.288 ± 0.006
CityFM [2]	0.205 ± 0.018	0.614 ± 0.025	0.391 ± 0.027	0.271 ± 0.004
FlexiReg	**0.766 ± 0.022**	**0.891 ± 0.024**	**0.753 ± 0.026**	**0.698 ± 0.014**
Improvement	**21.4%**	**2.4%**	**22.8%**	**28.3%**

San Francisco	Crime	Check-in	Service Call	Population
	R^2 ↑	R^2 ↑	R^2 ↑	R^2 ↑
MVURE [53]	0.594 ± 0.013	0.562 ± 0.021	0.479 ± 0.017	0.093 ± 0.016
MGFN [41]	0.601 ± 0.017	0.708 ± 0.010	0.468 ± 0.021	0.033 ± 0.048
HREP [59]	0.612 ± 0.014	0.629 ± 0.032	0.461 ± 0.029	0.127 ± 0.023
ReCP [15]	0.585 ± 0.075	0.783 ± 0.029	0.301 ± 0.119	0.067 ± 0.044
HAFusion [31]	0.682 ± 0.013	0.813 ± 0.024	0.612 ± 0.018	0.159 ± 0.006
RegionDCL [14]	0.413 ± 0.021	0.437 ± 0.024	0.256 ± 0.024	0.027 ± 0.025
UrbanCLIP [46]	0.283 ± 0.014	0.334 ± 0.002	0.292 ± 0.008	-0.395 ± 0.005
CityFM [2]	0.334 ± 0.008	0.298 ± 0.008	0.396 ± 0.008	0.023 ± 0.003
FlexiReg	**0.732 ± 0.014**	**0.859 ± 0.011**	**0.641 ± 0.021**	**0.480 ± 0.034**
Improvement	**7.3%**	**5.7%**	**4.7%**	**202%**

We make the following observations.

(1) Our model FlexiReg outperforms all competitors including even those using human mobility data in addition, across three cities in the USA (the other two cities will be shown next) and all four downstream tasks, improving R^2 by up to 202% over the best baseline HAFusion. This is attributed to our novel model design: (i) Our grid cell-based embeddings and their adaptive aggregation help capture local variations within regions, ensuring that the learned embeddings accurately reflect nuanced region characteristics. (ii) Our prompt enhanced embeddings extract task-specific information to meet the specific needs of different tasks. FlexiReg excels particularly on population prediction, as the street view images provide information such as building density and types, which strongly correlate with population distribution.

(2) The baseline models using human mobility data (e.g., HAFusion) outperform those based on publicly accessible data (e.g., CityFM) for most datasets and downstream tasks. This is because human mobility reflects population distribution and movement patterns of individuals, which are closely related to the downstream tasks, especially check-in count prediction for which these models perform particularly well. This highlights the difficulties and our technical contributions in designing a model that outperforms the mobility data-based models without using mobility data.

(3) The baseline models using readily accessible data perform poorly for the following reasons. RegionDCL uses only building footprints. It struggles to distinguish the different functionality of regions and hence their crime, check-in, service call, and population counts. UrbanCLIP uses satellite images and their textual descriptions generated by a vision language model (VLM). Satellite images are coarse-grained. Meanwhile, VLMs are prone to generating low-quality textual descriptions due to hallucination, which limits the capability of UrbanCLIP. CityFM was designed to generate embeddings for different geospatial entities (e.g., buildings and roads). Simply concatenating these entity embeddings to form region embeddings can dilute the unique region characteristics.

4.3 Cross-country Applicability (Q2)

We further show the applicability of FlexiReg over cities outside the USA, i.e., Singapore (Asia) and Lisbon (Europe). We compare with the models (RegionDCL, UrbanCLIP, and CityFM) using readily accessible data for the check-in and population count prediction tasks, as there is no mobility or crime/service call count data.

Table 4: Prediction Accuracy over Cities in Different Countries

	Singapore		Lisbon	
	Check-in	Population	Check-in	Population
	R^2 ↑	R^2 ↑	R^2 ↑	R^2 ↑
RegionDCL	0.102 ± 0.033	0.408 ± 0.034	0.457 ± 0.056	0.586 ± 0.039
UrbanCLIP	0.136 ± 0.022	0.305 ± 0.008	0.591 ± 0.012	0.801 ± 0.004
CityFM	0.239 ± 0.039	0.412 ± 0.010	0.195 ± 0.004	0.627 ± 0.003
FlexiReg	**0.309 ± 0.027**	**0.581 ± 0.027**	**0.831 ± 0.022**	**0.934 ± 0.008**
Improvement	**29.3%**	**41.0%**	**40.6%**	**16.6%**

Table 4 only reports R^2 for conciseness (full results are in our technical report [30]). FlexiReg outperforms all competitors consistently, with an improvement of at least 16.6% over the best baseline models. These results confirm FlexiReg's applicability across countries. Among the baseline models, CityFM performs better on Singapore, while UrbanCLIP is more suitable for Lisbon, underscoring the different characteristics of the two cities.

4.4 Adaptability to Region Formations (Q3)

To evaluate the robustness and adaptability of our cell-based embeddings, we form regions of different sizes by recursively merging (with random region selection) the initial 180 regions ("**180r**") of

New York City with their randomly selected neighboring regions to form sets of 150 ("**150r**"), 120 ("**120r**"), and 90 ("**90r**") regions. We repeat the experiments like above over each set of regions. We omit results on the other cities as the patterns resemble (same below).

Note that FlexiReg only needs to learn the cell embeddings once, which can be reused to form embeddings for the different sets of regions. In contrast, existing models, except for CityFM, require data reprocessing and model retraining for each set of regions.

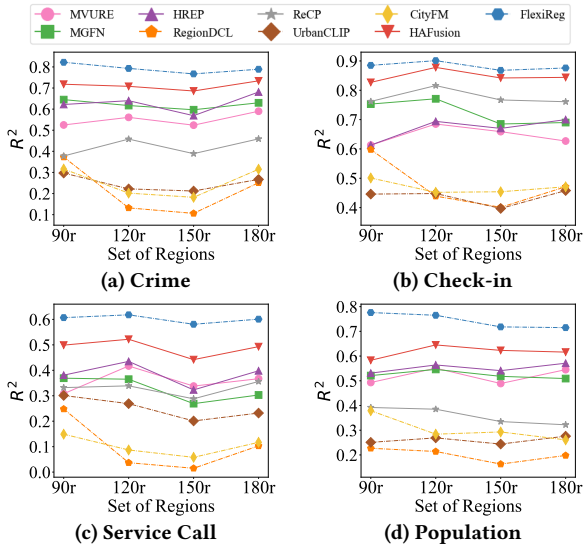


Figure 5: Adaptability to regions of varying sizes (NYC).

As Fig. 5 shows, FlexiReg again outperforms all competitors, with small performance variation (4%) across different sets of regions. In comparison, the only existing model flexible to region formations, CityFM, struggles across all four sets of regions, outperformed by almost all the other baseline models that relearn the embeddings for each set of regions. These observations demonstrate the adaptability of our model and the effectiveness of AdaRegionGen to accommodate different region formations.

4.5 Ablation Study (Q4)

We study the effectiveness of FlexiReg model components with the following variants: (1) **FlexiReg-w/o-PE** removes the prompt enhancer from the downstream task learning stage. (2) **FlexiReg-w/o-TAlign** replaces the text-region alignment module with a direct concatenation of text and region embeddings. (3) **FlexiReg-w/o-SVAlign** replaces the street view-region alignment module with the summation of all street view image embeddings in a region, followed by concatenating these with the region embeddings. (4) **FlexiReg-w/o-EC** removes the environment context-based contrastive learning and directly use ResNet to generate visual embeddings for each street view image. (5) **FlexiReg-w/o-CNN** replaces the CNN branch with a GNN branch to process satellite images. (6) **FlexiReg-w/o-Grid** directly learns region embeddings without using the grid cells. (7) **FlexiReg-w/o-FS** uses all features (including textual descriptions and street view images) during cell embedding learning. (8) **FlexiReg-w/o-WS** replaces the weighted summation of cell embeddings with a direct summation. (9) **FlexiReg-w/o-LT** uses the average of all token embeddings as the text embeddings instead of the last token embeddings.

We again repeat the experiments, and Fig. 6 presents the results. As expected, FlexiReg consistently outperforms all variants, highlighting the contribution of each model component to the overall effectiveness of FlexiReg. There are further observations:

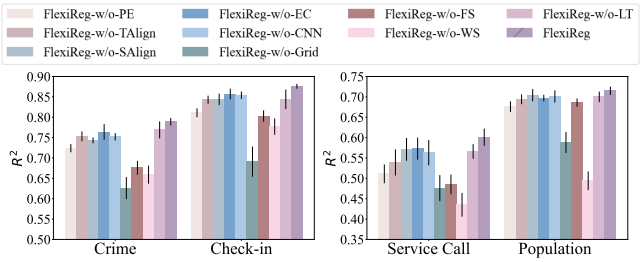


Figure 6: Ablation study results (NYC).

(1) FlexiReg-w-o-Grid is the worst across all tasks. This suggests that learning cell embeddings and then aggregating them into region embeddings contribute significantly to the overall model accuracy, as these steps enable the embeddings to better reflect local variations within regions. FlexiReg-w-o-WS also has low accuracy, implying that individual cell embeddings contribute differently to the region embeddings. A direct summation of cell embeddings could introduce noise and be even worse than not using cell embeddings at all (e.g., for service call and population count prediction).

(2) FlexiReg-w-o-PE is also less accurate than FlexiReg, which emphasizes the need for the prompt enhancer module to tailor region embeddings to meet task-specific requirements.

(3) The low performance of FlexiReg-w-o-FS and FlexiReg-w-o-CNN highlights the importance of using appropriate methods to handle different features. Notably, FlexiReg-w-o-FS, which leverages all six features, performs worse than FlexiReg-w-o-PE that uses only four features. Simply incorporating more features does not necessarily enhance the embedding quality.

4.6 Impact of Input Features (Q4)

To study the impact of input features, we exclude each of the POI, land use, geographic neighbor, satellite image, textual description, and street view image features, forming six variants: **FlexiReg-w/o-P**, **FlexiReg-w/o-L**, **FlexiReg-w/o-N**, **FlexiReg-w/o-SI**, **FlexiReg-w/o-T**, and **FlexiReg-w/o-SV**, respectively.

We repeat the experiments and report results in Fig. 7. The model variants are less accurate than FlexiReg that uses all input features, indicating the necessity of all these feature to achieve optimal embedding quality. FlexiReg-w/o-P has the lowest accuracy for crime, check-in, and service call count prediction. This is because POIs strongly correlate with human activities, which is a critical factor in these tasks. On the other hand, the land use feature contributes the most to the population count prediction task, because it explicitly identifies areas for residential and other purposes, explaining for the low accuracy of FlexiReg-w/o-L for the task.

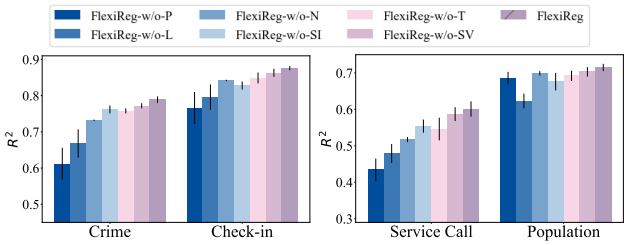


Figure 7: Impact of input features (NYC).

4.7 Applicability to Suburban Areas (Q5)

We also evaluate the applicability of our model to areas of different urban environments using regions in Staten Island, which is the largest borough of New York City by land area yet the least densely populated. This contrasts the area of the NYC dataset used above that covers Manhattan, which is the smallest borough by land area but the most densely populated.

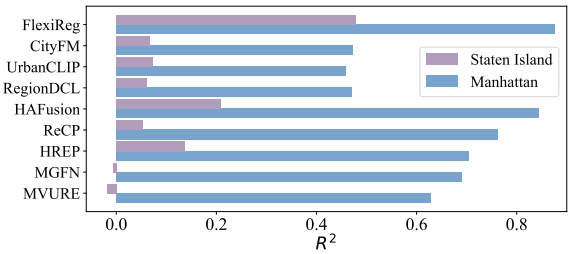


Figure 8: Model applicability to suburban areas (NYC).

We report in Fig. 8 the R^2 results for the regions in these two boroughs, focusing on the check-in count prediction task for conciseness. All models report lower accuracy (i.e., smaller R^2) on Staten Island than on Manhattan. This is expected, as Staten Island is less densely populated with smaller variations in the urban features to help the models learn distinctive embeddings for different regions. The mobility data-based models suffer the most, as the mobility data become more sparse. FlexiReg again outperforms all competitors, now with an even larger performance gap. The model does not rely on mobility data, while its adaptive aggregation module enables it to adaptively fuse different input features of grid cells to accommodate areas of different urban characteristics.

4.8 Additional Results (Q6)

We conducted additional experiments. In Appendix E, we include results on model efficiency, impact of shape and size of the grid cells, and the robustness to the prompt template for text-region encoding. In the technical report [30], we further include results on scalability issues and the impact of model hyper-parameters.

5 Related Work

We categorize existing works on urban representation learning into different groups from three perspectives as presented in Table 1. These works are summarized below and discussed in more detail in our technical report [30].

Human mobility data have been extensively utilized in existing studies. Some studies [38, 41, 47, 54] rely solely on human mobility data, limiting their ability to capture more comprehensive urban region characteristics from different perspectives. In contrast, most studies [5, 7, 8, 15, 31, 36, 45, 48, 52, 53, 55] integrate human mobility data with other types of features. For these studies, the core issue is that human mobility data are only available for certain regions.

Recently, studies [1, 2, 11, 12, 14, 32, 39, 42, 44, 46] have shifted focus to learning region embeddings from publicly accessible data, such as POIs and satellite images, to enhance model applicability. However, these models fall behind mobility-based models on downstream task, as they fail to capture underlying spatial correlations between different features. *We address this issue by proposing a novel, multimodal grid cell embedding learning module and an environment context-based contrastive learning approach to effectively explore different features by capturing spatial correlations and distinctive environmental characteristics.*

These existing studies primarily focus on the general region embedding learning stage, while paying limited attention to the downstream task learning phase. Prompt learning provides a promising solution for downstream task-based learning, which can guide the general region embeddings to adapt to specific tasks. This approach has already achieved promising results in fields such as natural language processing [3, 9, 27] and computer vision [13, 58]. Recently, prompt learning has also been introduced into urban computing [50, 56]. However, the urban tasks targeted in these works differ from those in our study.

In the context of region embedding learning, only one prior attempt — HREP [59] — exists, which generates prompt embeddings randomly, without conditioning on any input features. While the prompt embeddings may be correlated with the downstream tasks, they fail to capture the correlation between features and downstream tasks. To address this limitation, *we propose a prompt enhancer to extract and integrate task-specific information from textual and street-view imagery features into generic region embeddings to tailor for diverse downstream tasks.*

6 Conclusion

We proposed an urban region representation learning model named FlexiReg towards generating flexible representations to accommodate the needs of different downstream tasks with different region formations. FlexiReg only requires publicly accessible data. It learns fine-grained grid cells independent of the region partitions of downstream tasks, achieving region formation flexibility. FlexiReg comes with a multi-modal grid cell embedding learning module and an adaptive region embedding generation module to learn cell and region embeddings, respectively. It further incorporates a prompt enhancer to extract task-specific information and integrate such information into region embeddings to achieve downstream task flexibility. Extensive experiments on data from cities in different countries show that FlexiReg significantly outperforms all SOTA models across downstream tasks in diverse geographic regions.

Acknowledgments

This work is in part supported by the Australian Research Council (ARC) via Discovery Projects DP230101534 and DP240101006. Jianzhong Qi is supported by ARC Future Fellowship FT240100170. We thank the anonymous reviewers for their valuable feedback.

References

[1] Lubin Bai, Weiming Huang, Xiuyuan Zhang, Shihong Du, Gao Cong, Haoyu Wang, and Bo Liu. 2023. Geographic Mapping with Unsupervised Multi-modal Representation Learning from VHR Images and POIs. *ISPRS Journal of Photogrammetry and Remote Sensing* (2023).

[2] Pasquale Balsebre, Weiming Huang, Gao Cong, and Yi Li. 2024. City Foundation Models for Learning General Purpose Representations from OpenStreetMap. In *CIKM*.

[3] Eyal Ben-David, Nadav Oved, and Roi Reichart. 2022. PADA: Example-based Prompt Learning for on-the-fly Adaptation to Unseen Domains. *Transactions of the Association for Computational Linguistics* (2022).

[4] Colin P.D. Birch, Sander P. Oom, and Jonathan A. Beecham. 2007. Rectangular and Hexagonal Grids Used for Observation, Experiment and Simulation in Ecology. *Ecological Modelling* (2007).

[5] Meng Chen, Zechen Li, Hongwei Jia, Xin Shao, Jun Zhao, Qiang Gao, Min Yang, and Yilong Yin. 2025. MGRL4RE: A Multi-Graph Representation Learning Approach for Urban Region Embedding. *ACM Transactions on Intelligent Systems and Technology* (2025).

[6] Chicago Dataset. 2020. *https://data.cityofchicago.org/*.

[7] Jiadi Du, Yunchao Zhang, Pengyang Wang, Jennifer Leopold, and Yanjie Fu. 2019. Beyond Geo-First Law: Learning Spatial Representations via Integrated Autocorrelations and Complementarity. In *ICDM*.

[8] Yanjie Fu, Pengyang Wang, Jiadi Du, Le Wu, and Xiaolin Li. 2019. Efficient Region Embedding with Multi-View Spatial Networks: A Perspective of Locality-Constrained Spatial Autocorrelations. In *AAAI*.

[9] Xu Han, Weilin Zhao, Ning Ding, Zhiyuan Liu, and Maosong Sun. 2022. PTR: Prompt Tuning with Rules for Text Classification. *AI Open* (2022).

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *CVPR*.

[11] Tianyuan Huang, Zhecheng Wang, Hao Sheng, Andrew Y. Ng, and Ram Rajagopal. 2021. Learning Neighborhood Representation from Multi-Modal Multi-Graph: Image, Text, Mobility Graph and Beyond. *arXiv preprint arXiv:2105.02489* (2021).

[12] Weiming Huang, Daokun Zhang, Gengchen Mai, Xu Guo, and Lizhen Cui. 2023. Learning Urban Region Representations with POIs and Hierarchical Graph Infomax. *ISPRS Journal of Photogrammetry and Remote Sensing* (2023).

[13] Muhammad Uzair Khattak, Hanoona Rasheed, Muhammad Maaz, Salman Khan, and Fahad Shahbaz Khan. 2023. MaPLe: Multi-modal Prompt Learning. In *CVPR*.

[14] Yi Li, Weiming Huang, Gao Cong, Hao Wang, and Zheng Wang. 2023. Urban Region Representation Learning with OpenStreetMap Building Footprints. In *KDD*.

[15] Zechen Li, Weiming Huang, Kai Zhao, Min Yang, Yongshun Gong, and Meng Chen. 2024. Urban Region Embedding via Multi-View Contrastive Prediction. In *AAAI*.

[16] Zhenghao Lin, Zhibin Gou, Yeyun Gong, Xiao Liu, Ruochen Xu, Chen Lin, Yujiu Yang, Jian Jiao, Nan Duan, Weizhu Chen, et al. 2024. Not All Tokens Are What You Need for Pretraining. In *NeurIPS*.

[17] Lisbon Region Boundary. 2024. *https://gadm.org/index.html*.

[18] Llama. 2024. *https://www.llama.com/*.

[19] Yan Luo, Fu-lai Chung, and Kai Chen. 2022. Urban Region Profiling via Multi-Graph Representation Learning. In *CIKM*.

[20] New York Dataset. 2020. *https://opendata.cityofnewyork.us/*.

[21] Nominatim. 2024. *https://nominatim.openstreetmap.org/ui/reverse.html*.

[22] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation Learning with Contrastive Predictive Coding. *arXiv preprint arXiv:1807.03748* (2018).

[23] OpenStreetMap. 2024. *https://www.openstreetmap.org/*.

[24] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning Transferable Visual Models from Natural Language Supervision. In *ICML*.

[25] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2016. Faster R-CNN: Towards Real-time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2016).

[26] San Francisco Dataset. 2020. *https://datasf.org/opendata/*.

[27] Timo Schick and Hinrich Schütze. 2021. Few-shot Text Generation with Natural Language Instructions. In *EMNLP*.

[28] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A Unified Embedding for Face Recognition and Clustering. In *CVPR*.

[29] Singapore Dataset. 2019. *https://data.gov.sg*.

[30] Fengze Sun, Yanchuan Chang, Egemen Tanin, Shanika Karunasekera, and Jianzhong Qi. 2025. FlexiReg: Flexible Urban Region Representation Learning. *arXiv preprint arXiv:2503.09128* (2025).

[31] Fengze Sun, Jianzhong Qi, Yanchuan Chang, Xiaoliang Fan, Shanika Karunasekera, and Egemen Tanin. 2024. Urban Region Representation Learning with Attentive Fusion. In *ICDE*.

[32] Nicolas Tempelmeier, Simon Gottschalk, and Elena Demidova. 2021. GeoVectors: A Linked Open Corpus of OpenStreetMap Embeddings on World Scale. In *CIKM*.

[33] Waldo R. Tobler. 1970. A Computer Movie Simulating Urban Growth in the Detroit Region. *Economic Geography* (1970).

[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. In *NeurIPS*.

[35] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.

[36] Hongjian Wang and Zhenhui Li. 2017. Region Representation Learning via Mobility Flow. In *CIKM*.

[37] Lu Wang and Tinghua Ai. 2018. The Comparison of Drainage Network Extraction between Square and Hexagonal Grid-based DEM. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* (2018).

[38] Pengyang Wang, Yanjie Fu, Jiawei Zhang, Xiaolin Li, and Dan Lin. 2018. Learning Urban Community Structures: A Collective Embedding Perspective with Periodic Spatial-Temporal Mobility Graphs. *ACM Transactions on Intelligent Systems and Technology* (2018).

[39] Zhecheng Wang, Haoyuan Li, and Ram Rajagopal. 2020. Urban2Vec: Incorporating Street View Imagery and POIs for Multi-Modal Urban Neighborhood Embedding. In *AAAI*.

[40] Yongfu Wei, Yan Lin, Hongfan Gao, Ronghui Xu, Sean Bin Yang, and Jilin Hu. 2025. Path-LLM: A Multi-Modal Path Representation Learning by Aligning and Fusing with Large Language Models. In *WWW*.

[41] Shangbin Wu, Xu Yan, Xiaoliang Fan, Shirui Pan, Shichao Zhu, Chuanpan Zheng, Ming Cheng, and Cheng Wang. 2022. Multi-Graph Fusion Networks for Urban Region Embedding. In *IJCAI*.

[42] Yanxin Xi, Tong Li, Huandong Wang, Yong Li, Sasu Tarkoma, and Pan Hui. 2022. Beyond the First Law of Geography: Learning Representations of Satellite Imagery by Leveraging Point-of-Interests. In *WWW*.

[43] Ronghui Xu, Hanyin Cheng, Chenjuan Guo, Hongfan Gao, Jilin Hu, Sean Bin Yang, and Bin Yang. 2025. MM-Path: Multi-modal, Multi-granularity Path Representation Learning. In *KDD*.

[44] Ronghui Xu, Weiming Huang, Jun Zhao, Meng Chen, and Liqiang Nie. 2023. A Spatial and Adversarial Representation Learning Approach for Land Use Classification with POIs. *ACM Transactions on Intelligent Systems and Technology* (2023).

[45] Zhuo Xu and Xiao Zhou. 2024. CGAP: Urban Region Representation Learning with Coarsened Graph Attention Pooling. In *IJCAI*.

[46] Yibo Yan, Haomin Wen, Siru Zhong, Wei Chen, Haodong Chen, Qingsong Wen, Roger Zimmermann, and Yuxuan Liang. 2024. UrbanCLIP: Learning Text-Enhanced Urban Region Profiling with Contrastive Language-Image Pretraining from the Web. In *WWW*.

[47] Zijun Yao, Yanjie Fu, Bin Liu, Wangsu Hu, and Hui Xiong. 2018. Representing Urban Functions through Zone Embedding with Human Mobility Patterns. In *IJCAI*.

[48] Xixian Yong and Xiao Zhou. 2024. MuseCL: Predicting Urban Socioeconomic Indicators via Multi-Semantic Contrastive Learning. In *IJCAI*.

[49] Yuan Yuan, Jingtao Ding, Jie Feng, Depeng Jin, and Yong Li. 2024. UniST: A Prompt-empowered Universal Model for Urban Spatio-temporal Prediction. In *KDD*.

[50] Yuan Yuan, Jingtao Ding, Jie Feng, Depeng Jin, and Yong Li. 2024. Unist: A Prompt-empowered Universal Model for Urban Spatio-temporal Prediction. In *KDD*.

[51] Junbo Zhang, Yu Zheng, and Dekang Qi. 2017. Deep Spatio-Temporal Residual Networks for Citywide Crowd Flows Prediction. In *AAAI*.

[52] Liang Zhang, Cheng Long, and Gao Cong. 2023. Region Embedding With Intra and Inter-View Contrastive Learning. *IEEE Transactions on Knowledge and Data Engineering* (2023).

[53] Mingyang Zhang, Tong Li, Yong Li, and Pan Hui. 2020. Multi-View Joint Graph Representation Learning for Urban Region Embedding. In *IJCAI*.

[54] Ruixing Zhang, Liangzhe Han, Leilei Sun, Yunqi Liu, Jibin Wang, and Weifeng Lv. 2023. Regions are Who Walk Them: a Large Pre-trained Spatiotemporal Model Based on Human Mobility for Ubiquitous Urban Sensing. *arXiv preprint arXiv:2311.10471* (2023).

[55] Yunchao Zhang, Yanjie Fu, Pengyang Wang, Xiaolin Li, and Yu Zheng. 2019. Unifying Inter-Region Autocorrelation and Intra-Region Structures for Spatial Embedding via Collective Adversarial Learning. In *KDD*.

[56] Zijian Zhang, Xiangyu Zhao, Qidong Liu, Chunxu Zhang, Qian Ma, Wanyu Wang, Hongwei Zhao, Yiqi Wang, and Zitao Liu. 2023. Promptst: Prompt-enhanced Spatio-temporal Multi-attribute Prediction. In *CIKM*.

[57] Yu Zheng, Licia Capra, Ouri Wolfson, and Hai Yang. 2014. Urban Computing: Concepts, Methodologies, and Applications. *ACM Transactions on Intelligent Systems and Technology* (2014).

[58] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. 2022. Conditional Prompt Learning for Vision-Language Models. In *CVPR*.

[59] Silin Zhou, Dan He, Lisi Chen, Shuo Shang, and Peng Han. 2023. Heterogeneous Region Embedding with Prompt Learning. In *AAAI*.

A Input Cell Features

We illustrate the input features for each cell with Fig. 9.

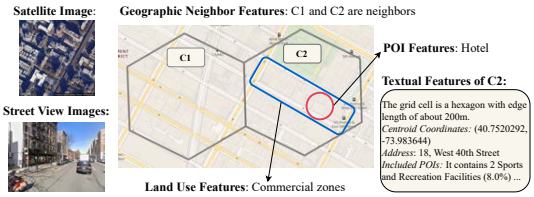


Figure 9: Example of input cell features.

POI features. We use 15 representative POI categories: *educational institutions, commercial and industrial properties, accommodation, cultural and recreational venues, healthcare and medical facilities, entertainment venues, places of worship, food and drink establishments, parking facilities, transportation and transit facilities, residential properties, camping and outdoor recreation sites, sports and recreation facilities, financial services, and others.*

Land use features. We use categories from OpenStreetMap: *grass, park, cemetery, forest, scrub, meadow, farmland, industrial, heath, retail, military, nature reserve, residential, commercial, orchard, farmyard, allotments, recreation ground, vineyard, and quarry.*

Street view imagery features. We sample geocoordinate points along the road network at 100-meter intervals. We remove redundant points within 20-meter of each another. When the number of sampled points is less than 5 for a cell, we randomly sample additional points for that cell. We collect street view images from four different directions (0°, 90°, 180°, 270°) at each sampled point.

B Satellite Image-Oriented Objective

We leverage the object counting task to predict the total number of POIs within the satellite image of each cell. Satellite images contain POI information, such as buildings and roads. This task encourages the model to extract POI features from the satellite images.

We obtain the ground-truth POI count y_i for grid cell c_i by summing up its POI feature vector $\mathbf{p}_i$, i.e., $y_i = \text{sum}(\mathbf{p}_i)$. Subsequently, the predicted POI count $\hat{y}_i$ is computed by passing the corresponding task embedding $\mathbf{e}_i^{si} \in \mathbf{E}^{si}$ through an MLP, expressed as $\hat{y}_i = \text{MLP}(\mathbf{e}_i^{si})$. Given y_i and $\hat{y}_i$, we employ the smooth L1 loss [25] as the satellite image objective function $\mathcal{L}^{si}$, computed as follows:

$$\mathcal{L}^{si} = \frac{1}{m}\sum_{i=1}^{m} u_i \tag{14}$$

$$u_i = \begin{cases} 0.5 \cdot (\hat{y}_i - y_i)^2 & if|\hat{y}_i - y_i| < \beta, \\ \beta \cdot |\hat{y}_i - y_i| - 0.5 \cdot \beta^2 & otherwise. \end{cases} \tag{15}$$

where β is a threshold hyperparameter ($\beta = 1$ in our experiments). When the absolute difference between the prediction and the ground truth is smaller than β, the loss behaves like the L2 loss (quadratic), ensuring smooth gradients for small errors. Conversely, if the difference exceeds β, the loss becomes an L1 loss (linear), which reduces the influence of outliers and enhances robustness.

C Cell Textual Description Example

We provide an example of the generated textual description for a hexagonal grid cell in Fig. 10. We collect POI information from OpenStreetMap and generate detailed street address using the reverse geocoding functionality of the Nominatim API [21].

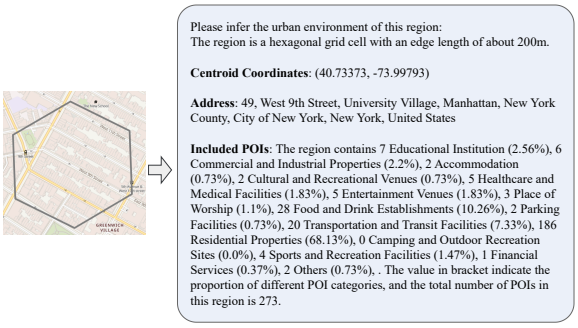


Figure 10: Example of the textual description of a grid cell.

D Model Hyperparameter Settings

All models were trained and tested on a machine equipped with an NVIDIA Tesla V100 GPU and 64 GB of memory.

For the competitor models, we follow parameter settings recommended in their papers as much as possible. We use special settings as described in the HAFusion paper [31] that reduce the model scales on CHI for MGFN, MVURE, HREP, and ReCP, as this dataset has fewer regions. The same applies to LX as it has fewer regions as well. RegionDCL, UrbanCLIP, and CityFM do not require special settings, as their training processes are determined by the number of building groups, image-text pairs, and geospatial entities, respectively, rather than the number of regions.

For our grid cell embedding learning model, the number of layers in the GNN branch of the grid-based intra-view feature learning module is 3 on New York City, and 2 for the other datasets. In the grid-based dual-feature attentive fusion module, the number of layers is 3 for all datasets. We train it for 2,000 epochs in full batches, using Adam optimization with a learning rate of 0.0001. For our downstream task learning model, the dimensionality of the text-region embeddings is 144, and the number of street view images used in the street view-region alignment module is 64. We train this model for 1,000 epochs in full batches, using Adam optimization with a learning rate of 0.0005 and weight decay of 0.0005. These hyperparameter values are set by a grid search.

The region embedding dimensionality d is set as 144 for our model following HAFusion and HREP. The region embedding dimensionalities for MVURE, MGFN, RegionDCL, and ReCP are 96, 96, 64, and 96, respectively, as suggested by their original papers. Our results on the impact of parameters [30] also show that these dimensionality values are optimal for the respective models (i.e., their yielded embeddings are of lower quality when d is 144). The region embedding dimensionalities of UrbanCLIP and CityFM are difficult to change from their default implementation. For UrbanCLIP, $d = 768$ which is determined by the image encoder, as it uses the embeddings of satellite images from a vision language model CLIP [24] as the region embedding. Similarly, for CityFM, $d = 1792$ which is determined by the dimensionality of the embeddings of different geospatial entities enclosed by a region.

E Additional Experimental Results

This section reports additional experimental results on model running time, the impact of the design choice of the grid cell structure, prompt templates, and the region embedding dimensionality.

Table 5: Embedding Learning and Testing Times (seconds)

	Embedding Learning			Downstream Task		
	NYC	CHI	SF	NYC	CHI	SF
MVURE	**35**	**15**	**34**	0.023 (0.001)	**0.053** (0.002)	0.026 (0.001)
MGFN	92	123	47	0.019 (0.001)	0 .061 (0.002)	0.029 (0.001)
HREP	51	45	51	92 (0.003)	146 (0.005)	91 (0.004)
ReCP	178	80	180	0.020 (0.001)	0.056 (0.002)	0.028 (0.001)
HAFusion	79	51	78	0.022 (0.001)	0.061 (0.002)	0.028 (0.001)
RegionDCL	149	1,779	324	**0.017** (0.001)	0.054 (0.002)	**0.023 (0.001)**
UrbanCLIP	1,532	2,497	3,359	86 (0.005)	86 (0.005)	84 (0.005)
CityFM	7,521	8,265	7,339	104 (0.006)	102 (0.005)	104 (0.006)
FlexiReg	208	282	436	137 (0.007)	103 (0.006)	142 (0.007)

Model Running Time. Table 5 reports the running times for embedding learning and downstream task learning (we omit the times on SG and LX as the comparative patterns resemble). The downstream task running times include both model training (prompt learning) and inference, with inference times in parentheses. The downstream tasks share identical input and output sizes.

Our model FlexiReg requires additional time to learn the region embeddings because it starts with learning embeddings for the cells, and there are more cells than regions. Additionally, our model takes additional time for downstream task training, as it integrates textual features and street view visual features into the region embeddings through two alignment modules. Adding these times together, our embedding learning can still be done in less than 10 minutes, which is rather affordable for training a model with deep learning. While FlexiReg takes extra learning times, it significantly reduces prediction errors, as shown in the experiments above.

The models that use readily accessible data, UrbanCLIP and CityFM, take more times for embedding learning than ours. This is because of large training datasets and complex model structures. These two models also take extra time at the training stage for the downstream tasks as they need to extract useful information from embedding by using deep networks with multiple layers. RegionDCL is slow on CHI, because the number of buildings is extremely high compared to other cities. The other baseline models, which use human mobility data, are typically faster in embedding learning 00, as human mobility data provides critical insights into regional interactions and the functional relationships between regions, enabling the model to converge quickly.

The inference times for the downstream tasks are similar across all models, as all prediction models share a simple structure.

Impact of Grid Cell Design. We use the following model variants to study the impact of shape and size of the grid cell: (1) **FlexiReg-Rect** uses a square grid instead of a hexagonal grid, where the edge length of each square is 200m. (2) **FlexiReg-LargeHex** uses a hexagonal grid with an edge length that is three times greater than the default length in FlexiReg. (2) **FlexiReg-SmallHex** uses a hexagonal grid with an edge length that is one-third of the default length in FlexiReg.

As Table 6 shows, FlexiReg produces the best (i.e., largest) R^2 scores across all four downstream tasks, confirming the effectiveness of our default grid cell design.

Table 6: Impact of grid cell design ($R^2 \uparrow$ on NYC)

	Crime	Check-in	Service Call	Population
FlexiReg-Rect	0.741	0.836	0.557	0.699
FlexiReg-LargeHex	0.587	0.784	0.425	0.233
FlexiReg-SmallHex	0.709	0.803	0.586	0.710
FlexiReg	**0.798**	**0.876**	**0.601**	**0.715**

FlexiReg-Rect shares a similar number of cells with FlexiReg. Its lower R^2 scores indicate that the sqaure grid is less effective. We conjecture that FlexiReg's better performance results from the symmetric structure of the hexagonal grid, where each cell has the same distance to all its neighbors, while this does not hold true when rectangular cells are used. Meanwhile, using hexagonal grid cells could better fit regions with irregular boundaries.

FlexiReg-LargeHex has the worst accuracy – it has only 89 cells while FlexiReg has 438. The larger cells can mix the distinctive features in a cell, missing local urban characteristics. When the cells become larger than the target regions, capturing the distinctive urban features within a region becomes even more challenging. In contrast, FlexiReg-SmallHex has 3,201 cells. Now each cell is too small, and urban features become sparse, which makes it difficult to learn meaningful embeddings. These smaller cells also incur more processing times (57, 208, and 1294 seconds for FlexiReg-SmallHex, FlexiReg, and FlexiReg-LargeHex, respectively). These findings ground our choice of using a hexagonal grid.

Table 7: Impact of prompt templates ($R^2 \uparrow$ on NYC)

	Crime	Check-in	Service Call	Population
FlexiReg-ReM	0.786	0.872	0.598	0.709
FlexiReg-RePh	**0.793**	0.869	0.600	0.710
FlexiReg-RePos	0.788	0.872	**0.603**	0.712
FlexiReg	0.789	**0.876**	0.601	**0.715**

Impact of Prompt Templates. As Fig. 10 shows, we use a simple prompt template that contains only a task instruction (i.e., the first sentence) which will be populated with information specific to each related cell (e.g., its address).

We study the impact of prompt templates with three variants: (1) **FlexiReg-ReM** removes the task instruction; (2) **FlexiReg-RePh** rephrases the task instruction with ChatGPT; (3) **FlexiReg-RePos** moves the task instruction to the end.

We repeat the embedding learning and downstream task prediction tasks as above. As Table 7 shows, the model variants achieve similar accuracy to FlexiReg, indicating that our model's effectiveness does not depend on a specific prompt template and is robust to prompt engineering. This robustness stems from our use of last-layer token embeddings from the LLM as textual embeddings, avoiding reliance on the generated text. Additionally, our proposed T-RAlign module effectively extracts task-relevant information from these embeddings.