Unsupervised Representation Learning of Spatial Data via Multimodal Embedding

Porter Jenkins
Penn State University
prj3@psu.edu

Ahmad Farag
Georgia Tech University
afarag7@gatech.edu

Suhang Wang
Penn State University
szw494@psu.edu

Zhenhui Li
Penn State University
jessieli@ist.psu.edu

ABSTRACT

Increasing urbanization across the globe has coincided with greater access to urban data; this enables researchers and city administrators with better tools to understand urban dynamics, such as crime, traffic, and living standards. In this paper, we study the Learning an Embedding Space for Regions (LESR) problem, wherein we aim to produce vector representations of discrete regions. Recent studies have shown that embedding geospatial regions in a latent vector space can be useful in a variety of urban computing tasks. However, previous studies do not consider regions across multiple modalities in an end-to-end framework. We argue that doing so facilitates the learning of greater semantic relationships among regions. We propose a novel method, RegionEncoder, that jointly learns region representations from satellite image, point-of-interest, human mobility, and spatial graph data. We demonstrate that these region embeddings are useful as features in two regression tasks and across two distinct urban environments. Additionally, we perform an ablation study that evaluates each major architectural component. Finally, we qualitatively explore the learned embedding space, and show that semantic relationships are discovered across modalities.

CCS CONCEPTS

• **Computing methodologies** → **Dimensionality reduction and manifold learning**; **Learning latent representations**;

KEYWORDS

Representation Learning, Multimodal Embedding, Satellite Imagery, Spatial Data

ACM Reference format:
Porter Jenkins, Ahmad Farag, Suhang Wang, and Zhenhui Li. 2019. Unsupervised Representation Learning of Spatial Data via Multimodal Embedding. In *Proceedings of The 28th ACM International Conference on Information and Knowledge Management, Beijing, China, November 3–7, 2019 (CIKM '19),* 10 pages.
https://doi.org/10.1145/3357384.3358001

CIKM '19, November 3–7, 2019, Beijing, China

ACM ISBN 978-1-4503-6976-3/19/11…$15.00
https://doi.org/10.1145/3357384.3358001

1 INTRODUCTION

As global urbanization continues to rise, so too does the proliferation of data collected in urban environments. Many cities across the world are collecting and releasing data to the public in an effort to increase urban efficiency and better administer to their citizens. City governments are using diverse data sets to control traffic [28] [22], predict crime [25], and monitor pollution [29]. While these data sets are often large, many spatial data sets have sparse natural representations, which suffer from the curse of dimensionality. High-dimensional feature spaces exponentially increase the need for labelled data, and make models difficult to estimate [24].

In the current work, we study the problem known in the literature as Learning an Embedding Space for Regions (LESR) [7]. LESR refers to the study of methods that project spatial regions into a lower-dimensional feature space, while preserving and highlighting important geospatial semantics. Recent work has shown that learning region embeddings can improve the performance of a variety of downstream prediction tasks. These approaches primarily leverage improved measures of similarity in the latent embedding space for better predictions. For example, Wang et al. [26] approached the LESR problem by considering both temporal dynamics and multi-hop human mobility transitions. Other recent work [7] considers locality-constrained spatial autocorrelations of regions, or learns region representations directly from satellite image data [12].

However, in many real-world applications we have many different channels, or modalities, of data to consider. For example, figure 1 shows a satellite image of downtown San Francisco and three regions of interest, (r_1, r_2, r_3). For each region, we observe its POI's, inter-region spatial distance, inter-region human mobility, and satellite image. A simple spatial measure would indicate that r_2 is most similar to r_3. However, when we look at the regions through all channels of information, we can see that r_2 is most semantically related to r_1. Both are popular tourist destinations with similar POI's. Both encompass residential buildings as well as a park. Additionally, many tourists travel between r_1 and r_2 via taxi. Recent work has demonstrated that capturing a joint view of the multi-modal state of data samples can discover better semantic representations. For instance, learning from both image and text data can improve, efficiency [6] or effectiveness [15] in image classification and word embedding models.

However, learning representations of geospatial regions is non-trivial. For one, it is difficult to jointly learn from many data types without excessively increasing the input dimensionality. Existing multimodal embedding techniques from NLP and computer vision are domain-specific and cannot be directly applied to geospatial settings. One major shortcomings of these models is that they do not account for spatial autocorrelation or spatial heterogeneity. A key characteristic of spatial problems is spatial autocorrelation, or

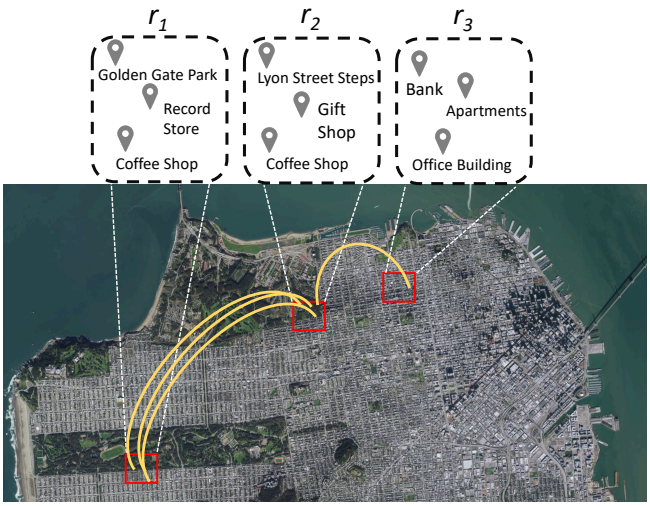


Figure 1: Example of multimodal spatial data in San Francisco. We observe three regions (r_1, r_2, r_3) their Point-of-Interest data (gray dotted box), distances to other regions, satellite images (red box), and human mobility data (yellow lines). Note that r_2 and r_3 are closer in physical space, but r_2 is more semantically related to r_1 in terms of POI's, human mobility, and visual features. Simple measure of similarity may not adequately capture spatial heterogeneity.

the covariance of samples from nearby locations. Additionally, heterogeneity often occurs when the variance of data is non-constant over space. A good spatial embedding technique should capture both of these properties in the latent space. Moreover, state-of-the-art geospatial embedding methods [7][26][12] do not account for spatial autocorrelation of multiple data modalities across regions.

Therefore, in this paper we explore learning region embeddings from multimodal data. To solve this problem we propose the RegionEncoder, a deep learning approach that is well suited for spatial problems. The model takes as input data from satellite images, spatial networks, weighted human mobility graphs, and point-of-interest data and learns a unified embedding of each region through a discriminator function. Spatial autocorrelation of data is considered in two ways: 1) a Graph Convolutional Network that performs approximate, local spectral operations, and 2) a discriminator that learns to distinguish intra-region multimodal embeddings from negative samples. Additionally, spatial heterogeneity is modeled through a loss term that reconstructs a weighted graph of taxi transition data.

In our experiments, we validate RegionEncoder on two prediction tasks, and across two major urban environments: Chicago and New York City. Additionally, we perform an ablation study and qualitatively analyze the nature of the learned region embeddings. We demonstrate the proposed method is effective at learning regional features that can improve the performance of a variety of downstream tasks and discovers interesting semantics among regions.

In summary, the key contributions of our paper are:

- We expand on the Learning an Embedding Space for Regions (LESR) problem by exploring multimodal data for region embedding (Section 2).
- We propose a novel deep learning model that learns representations of regions across a variety of modalities, including image, POI, spatial networks, and human mobility data (Section 3).
- We empirically validate RegionEncoder on multiple downstream prediction tasks in two distinct urban environments (Section 4).

2 PROBLEM DEFINITION

We define the problem of unsupervised representation learning of discrete regions as a joint embedding learning problem of four disparate data objects: a spatial graph, an inter-region human mobility ('flow') graph, a set POI locations, and satellite images.

- **Spatial Proximity** is a natural indicator of regional similarity and autocorrelation. Tobler's first law of geography posits that "everything is related to everything else, but near things are more related than distant things" [23].
- **Human Mobility Data** has been shown to be a useful measure of similarity between geospatial regions [26] [25]. Intuitively, a large volume of people travelling from one region to another should indicate an important semantic relationship.
- **Point-of-Interest (POI) Data** is an important measure of the typical functions and activities of a region. Regions with similar functions share a semantic relationship even if the two regions are not spatially adjacent. POI data are often high-dimensional due to the large variety of locations in an urban environment.
- **Satellite Image data** has shown to be useful for a variety of geospatial computing tasks [12] [2]. Recent improvements in remote sensing and satellite image technology have created easy access to high-resolution, and high-frequency imagery. The promise of such easy-to-access data is that of more automated analysis of urban environments.

2.1 Data Types

In what follows, we provide formal definitions for the various data objects we utilize in this work.

2.1.1 Urban Area Window. Prior to training the model, we define a geo-spatial window that defines our larger region of interest. The window is defined by a set of tuples that characterize the latitude and longitude ranges, $W = \{(lat_{min}, lat_{max}), (lon_{min}, lat_{max})\}$.

2.1.2 Regions. We then partition the space defined by our window, W, into a set of n non-overlapping regions, $\mathcal{R} = \{r_1, r_2, ..., r_n\}$, where each region has dimensions (w, h). We typically construct our regions by evenly splitting the latitude and longitude spaces such that $w = \frac{lat_{max} - lat_{min}}{m}$, and $h = \frac{lon_{max} - lon_{min}}{m}$, where m is the number of partitions on each dimension. This gives a total number of regions, $n = m^2$. Each region $r_i = \langle lat^{(i)}_{min}, lat^{(i)}_{max}, lon^{(i)}_{min}, lat^{(i)}_{max} \rangle$ is defined by four bounding spatial points

2.1.3 Spatial Graph. As is done many geospatial applications, we have clear access to a spatial graph, $\mathcal{G}_s = (\mathcal{V}, \mathcal{E}_s)$. The spatial graph input has vertices, $\mathcal{V}$ which is equal to the set of regions $\mathcal{R}$. The edges, $e^{(s)}_{i,j} \in \{0, 1\}$, denote that regions r_i and r_j are adjacent.

From the spatial graph $\mathcal{G}_s$ we can construct the adjacency matrix $\mathbf{A}$, and the degree matrix, $\mathbf{D}$.

2.1.4 Mobility Flow Graph. We collect a set of taxi mobility data wherein we observe a set $\mathcal{T} = \{t_1, t_2, ..., t_T\}$ of trips. Each trip t_i has the format $\langle l_s, l_e \rangle$, where l_s and l_e are the start and end locations coded by latitude and longitude coordinates. We map each trip in $\mathcal{T}$ to the corresponding start and end regions, r_i and r_j respectively. We can then construct the flow graph $\mathcal{G}_f = (\mathcal{V}, \mathcal{E}_f)$, where the vertices $\mathcal{V}$ are the same as above, and the weighted edges, $\mathcal{E}_f = \{e_{ij}^{(f)}\}$, are normalized counts of inter-region taxi flow. We input these data into our model by means of a weighted edge matrix, Ω, where each component, $\omega_{ij} = e_{ij}^{(f)}$.

2.1.5 POI data. Additionally, we input a set of Point-of-Interest (POI) data, $\mathcal{P} = \{p_1, p_2, ..., p_P\}$, where each point-of-interest, $p_i = \langle l_{lat}, l_{lon}, c \rangle$ is defined by it's latitude point, l_{lat}, longitude point, l_{lon} and POI category, $c \in C$. We map each observed point-of-interest, p_i to its corresponding region, r_i, and compute each region's POI distribution. From the POI data we construct a matrix of nodal features, $\mathbf{X} \in \mathbb{R}^{(nxp)}$ where p is equal to the number of categories. In other words, the matrix $\mathbf{X}$ describes POI distribution in each region, r_i.

2.1.6 Image Data. The final raw input to our problem is a set of n of ground satellite images, $\mathcal{I} = \{I_1, I_2, ..., I_n\}$. Each image, $I_i = \langle lat_{min}^{(i)}, lat_{max}^{(i)}, lon_{min}^{(i)}, lat_{max}^{(i)} \rangle$ is defined by four bounding spatial points, such that $r_i = I_i$. We observe exactly one satellite image per region. Additionally, each image, I_i, is a color image with three input channels.

2.2 Region Representation Learning

The goal, or output, of the region representation learning problem is to learn a function that maps disparate data types to a single, latent euclidean space:

$$\Gamma : \mathcal{X} \rightarrow \mathbf{H} \in \mathbb{R}^{(n \times d)} \tag{1}$$

Where $\mathcal{X} = \langle \mathcal{R}, \Omega, \mathbf{A}, \mathbf{X}, \mathbf{D}, \mathcal{I} \rangle$ and each row in $\mathbf{H}$, $\mathbf{h}_i$, is a distributed vector representation of region, r_i and $\Gamma(\cdot)$ is a function that encodes information from all data sources into a single latent space. This vector representation should discover interesting semantics among regions across the input data types, and be useful as features in downstream machine learning tasks such as classification or regression.

3 THE PROPOSED FRAMEWORK – REGIONENCODER

In this section, we provide details of the proposed framework RegionEncoder, a deep learning approach for learning low-dimensional distributed representations of discrete spatial regions. An overview of RegionEncoder is shown in Figure 2. It consists of three main parts: (i) a denoising convolutional autoencoder extracting visual features from satellite images; (ii) a graph convolutional network (GCN) for learning representations from spatially distributed point-of-interest (POI) data and taxi transition volume; and (iii) a discriminator for fusing the visual and spatial features into coherent feature space. Next, we introduce the details of each component.

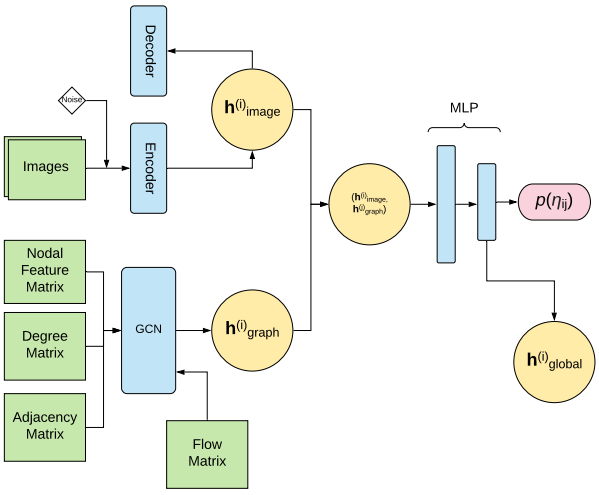


Figure 2: RegionEncoder architecture. We input data from multiple modalities in two simultaneous steps. 1) We feed raw visual input from satellite images into a denoising autoencoder. 2) We embed the spatial graph (adjacency, and degree matrices), a "flow" graph derived from human mobility data, and nodal features using a graph convolutional network. We input the hidden states from the autoencoder and GCN to a discriminator (multi-layer perceptron) to get the global hidden state, $\mathbf{h}_{global}^{(i)}$ for region r_i.

3.1 Denoising Autoencoder

Generally, an autoencoder is an unsupervised neural network that projects samples from the original features space to a lower dimensional latent space via series of non-linear mappings [3] [7]. Typically, the autoencoder model tries to map its input to its output, by minimizing the reconstruction error, $L(\mathbf{I}, g(f(\mathbf{I})))$ [8]. Here f and g respectively denote the encoder and decoder functions and $\mathbf{I}$ is the input image tensor. In minimizing the reconstruction loss the model learns an internal representation of the data. However, given enough capacity an autoencoder can easily learn the identity function. This motivates the use of other classes of stochastic autoenocoders such as denoising and variational autoencoders. In this paper, we rely on the denoising autoencoder to provide a latent space mapping for satellite images. We first perturb the input image with a small amount of Gaussian noise, $\tilde{I}_i = I_i + N(0, 1)$ and try and reconstruct the original image from the noisy input by minimizing $L(\mathbf{I}, \hat{\mathbf{I}})$ where,

$$\mathbf{h}_{image} = f(g(\mathbf{h}_{image})) \tag{2}$$

3.1.1 Reconstruction Loss. To train the RegionEncoder model, we minimize the mean squared error of the reconstructed image to its input as in equation 3.

$$L_I = \frac{1}{n} \sum_{i=1}^{n} (\mathbf{I}_i - \hat{\mathbf{I}}_i)^2 \tag{3}$$

Where $\mathbf{I}_i$ is the ground truth image, and $\hat{\mathbf{I}}_i$ is its denoised reconstruction. In minimizing the loss in equation 3, the autoencoder produces a hidden state of each image, $\mathbf{h}_{image}$.

3.1.2 Architecture. In many cases, the encoder and decoder functions chosen are simple feedforward networks. However, given the grid-like structure of image data, we rely on a Convolutional Neural Network (CNN) to project the image data down at each layer to construct a hidden representation. Specifically, we use two convolutional layers in the encoder, followed by three fully connected layers. In the decoder, we use three linear layers, followed by two convolutional layers.

3.2 Graph Convolutional Network

Recent work has extended Convolutional Neural Networks from low-dimensional, grid-like data to more high-dimensional, flexible datasets such as graphs [11] [13] [4]. GCN's typically rely on filter parameters that are shared over all locations in the graph to project nodal features at each layer. Recent work [13] uses a convolutional architecture that approximates first-order spectral graph convolutions by propagating the input data through augmented adjacency and degree matrices (equation 4).

3.2.1 GCN Architecture. To learn a distributed representation of each node in our regional graph, we follow this layer-wise propagation rule [13]:

$$\mathbf{Z}_l = \sigma(\hat{\mathbf{D}}^{-\frac{1}{2}}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-\frac{1}{2}}\mathbf{Z}_{l-1}\mathbf{W}_{l-1}) \tag{4}$$

Where $\mathbf{Z}_l$ is the network's hidden state at layer l, $\hat{\mathbf{A}}$ is the augmented adjacency matrix, $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, and $\hat{\mathbf{D}}$ is the degree matrix derived from $\hat{\mathbf{A}}$. For an activation function, $\sigma(\cdot)$ we choose ReLU. In the context of the proposed RegionEncoder model, the GCN component takes as input a nodal feature matrix, $\mathbf{X} \in \mathbb{R}^{n \times p}$, where p is the number of nodal features. More specifically, in equation 4 we input $\mathbf{X}$ for $\mathbf{Z}_0$, The GCN component then outputs a hidden state, $\mathbf{H}_{graph} \in \mathbb{R}^{n \times d_{graph}}$.

In our experiments we let the number of layers, $l = 2$. We then extract the hidden state from the GCN at the final layer, Z_2, and use that as our graph embedding, $\mathbf{Z}_2 = \mathbf{H}_{graph}$.

3.2.2 GCN Loss. The GCN component of the model is crafted to minimize two objectives simultaneously. These two carefully selected loss components have the effect that the GCN layer of the RegionEncoder model learns to distinguish true contextual regions from a noise distribution, as well as encoding knowledge from the trajectory similarity as in [26].

SkipGram Loss with Negative Sampling: First, RegionEncoder minimizes the well-known SkipGram loss with negative sampling in equation 5. In this loss term, the graph representation is constrained to values that distinguish regions that are spatially adjacent from those that are not.

$$\begin{aligned} L_s = &-\frac{1}{n}\sum_{i=1}^{n}\sum_{c\in C_i}\log p(r_c|r_i) = \\ &-\frac{1}{n}\sum_{i=1}^{n}\sum_{c\in C_i}[\log\sigma(v_{r_c}^\top v_{r_i}) + \sum_{j=1}^{k}\mathbb{E}[\log\sigma(-v_{r_j}^\top v_{r_i})]] \end{aligned} \tag{5}$$

where k is the number of negative samples, v_{r_i} is the vector representation for region, r_i, and the set C_i is the set of context regions for region, r_i. We construct the context from nodes adjacent to r_i. To get negative samples for r_i, we randomly select k regions not adjacent to r_i with probabilities proportional to their degree.

Transition Reconstruction Loss: Second, we incorporate a loss component for reconstructing a weighted representation of the graph based on taxi mobility flow in equation 7. This has the effect that the latent graph embedding is also trained to preserve the first order proximity of the mobility flow graph. We model the reconstruction of the weighted edge graph using KL-divergence as in [19]. We define the joint probability of any two regions, r_i and r_j:

$$p(r_i, r_j) = \frac{1}{1 + e^{-v_{r_i}^\top v_{r_j}}} \tag{6}$$

where v_{r_i}, is the vector representation for region, r_i. The distance between the empirical sample probability, $\hat{p}(r_i, r_j) = \frac{w_{ij}}{\sum_{(i,j)\in\mathcal{R}} w_{ij}}$, and the estimated probability, $p(r_i, r_j)$ is a natural choice of loss function to reconstruct the weighted flow graph. Using KL-divergence [19], we get the transition reconstruction loss:

$$L_f = d(\hat{p}(r_i, r_j), p(r_i, r_j)) = \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\log p(r_i, r_j) \tag{7}$$

3.3 Discriminator

The final component of the RegionEncoder model uses a feedforward network, or Multilayer Perceptron (MLP), that acts to unify all disparate data sources into a single latent space embedding. The discriminator layer takes as input the hidden state produced by the autoencoder, $\mathbf{H}_{image}$, and the hidden state from the GCN, $\mathbf{H}_{graph}$. Using a parameterized function, $f(\cdot)$, the upper-level of the RegionEncoder model learns to predict whether two hidden states $\mathbf{h}_{image}^{(i)}$, and $\mathbf{h}_{graph}^{(j)}$ belong to the same region as in equation 8. In our experiments, $f(\cdot)$ is a Multilayer Perceptron with a single hidden layer. We concatenate the vectors, $\mathbf{h}_{image}^{(i)}$ and $\mathbf{h}_{graph}^{(j)}$, as they are fed into the MLP.

3.3.1 Binary Cross Entropy Loss.

$$p(\eta_{i,j} = 1) = f(\mathbf{h}_{image}^{(i)}, \mathbf{h}_{graph}^{(j)}) \tag{8}$$

where $\eta_{i,j}$ is a binomial random variable such that:

$$\eta_{i,j} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} \tag{9}$$

When training the RegionEncoder model, as the outputs from the lower layers are fed into the discriminator, we produce k negative samples. To construct the set of negative samples, we uniformly sample vectors $\mathbf{h}_{image}^{(i)}$ and $\mathbf{h}_{graph}^{(j)}$, where $i \neq j$. We then compute the loss in equation 10 over the n ground truth samples and k negative samples

$$\begin{aligned} L_d = &-\sum_{i=1}^{n+k}\eta_{i,j}\log p(\eta_i|\mathbf{h}_{image}^{(i)}, \mathbf{h}_{graph}^{(j)}) + \\ &(1-\eta_{i,j})\log(1 - p(\eta_i|\mathbf{h}_{image}^{(i)}, \mathbf{h}_{graph}^{(j)})) \end{aligned} \tag{10}$$

Finally, we extract the hidden layer from the discriminator MLP and treat it as our global embedding vector, $\mathbf{h}^{(i)}_{global}$

3.4 Training Objective

We jointly train all components of our model by minimizing the global loss function in equation 11. The vector $\boldsymbol{\lambda}' = [\lambda_s, \lambda_f, \lambda_I]$ is a hyperparameter, where each λ_i controls the contribution of L_i to the global loss function. We tune $\boldsymbol{\lambda}$ via random search. Additionally, In our experiments, we incorporate an L-2 norm weight decay term on each model parameter vector, which we omit here for brevity.

$$\mathcal{L} = L_d + \lambda_s L_s + \lambda_f L_f + \lambda_I L_I \tag{11}$$

4 EXPERIMENTS

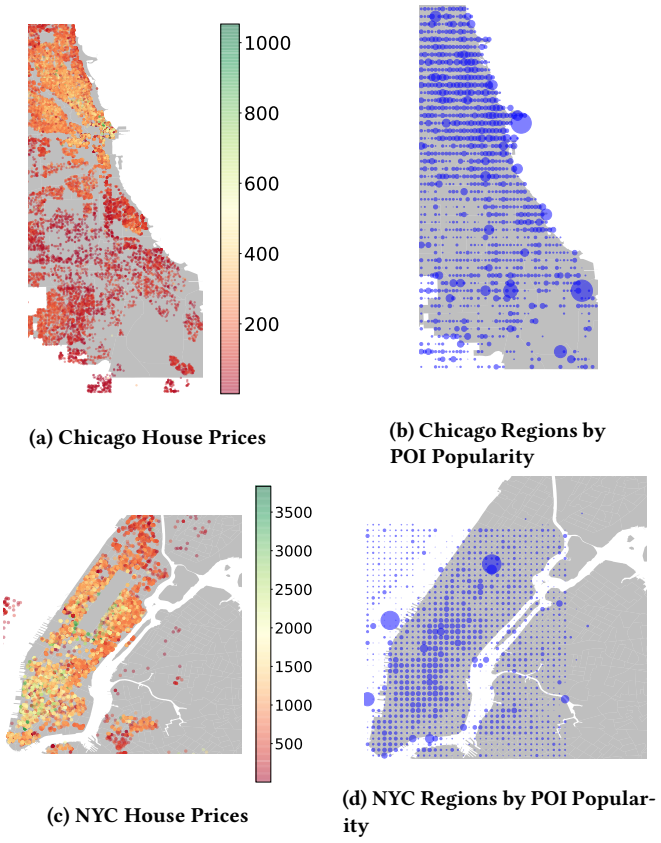
In the following section [1] we describe the data, and the strategies used to evaluate the RegionEncoder method. We perform two quantitative evaluations across urban environments: Chicago and New York City. First, we recreate the experiment in [7]; we use learned region embeddings directly as features in a regression model to predict region popularity from mobile checkin-ins. Second we use a similar strategy to predict house prices. For each task, we test learned feature vectors produced by various embedding techniques. We vary the complexity of the embedding technique as well as the amount of information used. Our hypothesis is that learning across modalities under the RegionEncoder framework produces regional features that yield improvement in downstream prediction tasks. We then perform an ablation study wherein we evaluate the effect that each major architectural component has on the prediction error of both tasks. Finally, we qualitatively analyze the latent space produced by RegionEncoder, and what semantics are captured. In this experiment, we look at a variety of query regions in Chicago and discuss what semantics they share with their nearest neighbors in the embedding space.

4.1 Region Popularity Prediction

We first evaluate the RegionEncoder model by predicting region popularity. We recreate the experiment in [7] and use a linear regression to predict regional check-in counts. Let p_i denote the popularity of region r_i. Then the goal is to learn a regression model $p_i = \boldsymbol{\beta}^\top \mathbf{x}_i + \epsilon$, where $\mathbf{x}_i$ is a vector of region-level features. We use linear regression with an L-1 penalty (lasso) with 5-fold cross-validation [24]. Additionally, we define regional popularity, p_i as the average POI check-in count in each region.

4.2 House Price Prediction

Our second task is house price prediction in both Chicago and New York city. In this experiment, we train the model, $\hat{y}_i = f(\langle \mathbf{x}_i, \mathbf{z}_i \rangle)$, where $\hat{y}_i$ is the predicted price per square foot for house i, $\mathbf{x}_i$ is a vector of region-level features, and $\mathbf{z}_i$ is a vector home-level features. We use the same home-level features in all models: number of bedrooms, number of bathrooms, square footage. In our experiment we train and evaluate a random forest regression model [18] using 5-fold cross-validation for all feature baselines and report the results in table 2.



(a) Chicago House Prices

(b) Chicago Regions by POI Popularity

(c) NYC House Prices

(d) NYC Regions by POI Popularity

Figure 3: House prices and region popularity in Chicago and New York City. (a,c) House prices are colored by relative price per square foot: expensive homes are given green values and less expensive homes red values. (b,d) Each region's centroid is plotted. The size of a region's bubble is scaled by it's popularity.

4.3 Experimental Settings

4.3.1 Dataset Description. All of the data sets used in our experiments are publicly available from government entities or open API's.

Region and Window definitions We first select the window of the two urban environments: Chicago and New York City. We define each urban window via latitude and longitude ranges

- *Chicago* Latitude: [41.6284, 41.9989], Longitude: [-87.6998, -87.5262]
- *New York City* Latitude: [40.6994, 40.8293], Longitude: [-74.0210, -73.8908]

In each city, we partition both the latitude and longitude ranges into 50 equal subspaces. These subspaces define the boundaries of each region within our window, giving us a grid with 2,500 regions. In Chicago, each region is approximately .30 km x .80 km; in New York city each is approximately 0.22 km x 0.28 km.

Mobility Data: As discussed above, human mobility data can be used as a measure of correlation among discrete spatial regions [26]

[1] Code and links to data used can be found on the author's GitHub: https://github.com/porterjenkins/region-encoder

[25]. We use taxi 'flow' data as a measure of mobility between regions, which we collect from open data portals provided by Chicago [21] and New York City [20], respectively.

In Chicago we collect approximately 113 million unique taxi trips data from 2013 to 2018. Similarly, we collect 69 million unique taxi trips in New York city from January 2016 to June 2016.

POI Data: Point-of-Interest data is obtained from the Foursquare API [5]. In total, the data set contains 112,000 POI records in Chicago. In New York, we observe 83,000 POI locations. Each record provides the venue name, category, number of check-ins, latitude, longitude, and number of unique visitors. Some example categories include residential, nightlife, education, and food.

Image Data: Additionally, we collect satellite image data from the Google Static Maps API [9]. The luxury of the static API is that it allows for high resolution images up to .3 m resolution. We query a total of 5,000 images, one for each region in each city. We then compress each image to 50-by-50 pixel ($3 \times 50 \times 50$) resolution for training.

House Price Data Home sale prices are collected via Zillow [30], the popular real estate valuation website. We observe the actual sale price, latitude, longitude, and square footage of each home. In Chicago, we collect approximately 45,000 house price samples. In New York, we obtain more than 54,000 homes in Manhattan, Brooklyn, and Queens.

We plot the spatial distribution of house prices and check-ins in both New York and Chicago in Figure 3.

4.3.2 Embedding and Feature Extraction Baselines. We compare RegionEncoder to the following feature learning baselines. Note we hold constant the hidden dimension size at $d = 32$ for all reported results and all methods.

- **Naive (spatial)** A naive spatial feature vector. For house price prediction, we use a one-hot representation of regions because we observe multiple samples per region. In the region popularity experiment where we observe one sample per region, we use latitude and longitude of the region centroid.
- **Raw Features** We concatenate raw features from the input data. This results in a sparse feature vector of the POI distribution and rows from the mobility transition matrix.
- **Raw Features + Kmeans** We use the same features as the previous baseline, but also concatenate a simple representation for region images [12]. We cluster images from the 2,500 regions in pixel space using kmeans [24]. Each region is represented as its distance to cluster centroids.
- **DeepWalk** The DeepWalk model [17] is an extension of Word2Vec [16] to graph data. We simulate random walks along the spatial graph that act as input sentences to the Word2Vec model. In our experiments, we let walks = 100 and the walk length = 40.
- **Node2Vec** The Node2Vec algorithm [10] uses biased random walks to learn latent features that maximize the likelihood of preserving local networks. We generate random walks on the spatial graph, set the walk length = 80 and let walks = 10.
- **AutoEncoder** We use a denoising autoencoder [1] [8] to learn features for satellite images. We perturb the input images with gaussian noise.
- **Tile2Vec** is an unsupervised representation learning technique for image data that relies on CNN's and triplet loss [12]. Each triplet consists of an anchor, a neighbor, and a distant region. In our experiments, we let the margin, $m = .1$ and define the neighborhood as spatially adjacent regions. We use the same CNN architecture as the image encoder layer of RegionEncoder.
- **Heterogeneous Dynamic Graph Embedding (HDGE)** is a recent spatial embedding approach that jointly embeds a spatial graph and flow graph with temporal dynamics [25]. Specifically, the authors observe taxi trip transitions each day at time $t = 1, ..., T$. They combine this graph with a spatial graph weighted by inverse distance and learn embedding vectors via a skipgram objective. We implement their model using $T = 8$ time periods each day.
- **Multi-view Spatial Network Embedding (MSNE)** The proposed method in [7] relies on multi-view, intra-region POI networks in an autoencoder framework to represent regions. Additionally, the authors incorporate a novel top-k, autocorrelation layer to capture inter-region similarities. We let k = 5 in our implementation.
- **Autoencoder + Deepwalk** We concatenate feature vectors from the Autoencoder and DeepWalk, which accounts for image and spatial graph data.
- **MSNE + Tile2Vec** We concatenate feature vectors from MNSE and Tile2Vec, two cutting edge methods. This accounts for all input data modalities.

4.4 Results

In both our house price and region popularity prediction experiments, we use mean absolute error (MAE) and root mean squared error (RMSE) to evaluate each embedding technique.

Region Popularity Prediction The results from both Chicago and New York City (table 1) demonstrate that using features learned from the RegionEncoder model results in lower cross-validated prediction errors.

In particular, the features learned using RegionEncoder are useful in the New York environment. We see significant improvement over baselines across both metrics. Even the cutting edge approaches, MSNE, and HDGE result in higher test error. Moreover, independently adding features from different modalities (Raw Features + Kmeans, Autoencoder + DeepWalk, and MSNE + Tile2vec) cannot match the performance of learning the features jointly.

In general, embedding techniques that utilize graph information only (DeepWalk, Node2Vec, etc...) achieve better results as compared to those that embed images only (Tile2Vec, AutoEncoder). We see the biggest performance boost as POI and mobility are added (MSNE). We see another boost when all data sources are embedded separately (MSNE + Tile2Vec) and again when all data are embedded jointly (RegionEncoder).

In Chicago, RegionEncoder still outperforms all baselines in terms of both RMSE and MAE. However, the improvement is not as dramatic as what we see in New York. We observe improvement

Table 1: Region Popularity Prediction: 5-fold cross-validated errors for predicting average check-in count per region.

		Chicago		New York City	
Unsupervised Features	Data	RMSE	MAE	RMSE	MAE
Naive (spatial)	-	308.2375	132.5632	1770.9903	539.2173
Raw Features	Graph, Mobility, POI	520.9765	219.2348	2034.5784	545.1958
Raw Features + Kmeans	Graph, Mobility, POI, Image	529.6387	226.5416	2036.0156	570.6420
DeepWalk	Graph	299.9857	121.4501	1782.6428	568.7610
Node2Vec	Graph	298.2287	119.7224	1771.8400	534.5297
Tile2Vec	Image	315.9221	147.5118	1818.9645	606.1170
Autoencoder	Image	307.8026	131.2001	1826.5907	611.7085
Autoencoder + DeepWalk	Image, Graph	300.4517	123.1669	1792.7352	583.4387
HDGE	Graph, Mobility	311.5563	137.3171	1731.7406	469.2052
MSNE	Graph, Mobility, POI	296.4344	109.1206	1749.7983	467.6025
MSNE + Tile2Vec	Graph, Mobility, POI, Image	296.4829	109.0536	1751.3210	468.0732
RegionEncoder	Graph, Mobility, POI, Image	**293.9023**	**107.6857**	**1713.3417**	**421.2943**

Table 2: House Price Prediction: 5-fold cross-validated errors for predicting house price per square foot

		Chicago		New York City	
Unsupervised Features	Data	RMSE	MAE	RMSE	MAE
Naive (spatial)	-	20072.5992	862.0086	6380.4850	507.7427
Raw Features	Graph, Mobility, POI	17916.6182	774.0934	5902.5004	472.6627
Raw Features + Kmeans	Graph, Mobility, POI, Image	16589.4051	725.4515	6320.5263	487.9767
DeepWalk	Graph	18428.6258	843.6346	5980.6836	479.5409
Node2Vec	Graph	18140.0479	811.8831	6742.3929	514.8081
Tile2Vec	Image	17588.1067	821.0038	6290.1082	586.9296
Autoencoder	Image	19258.4378	846.9709	5486.1961	468.1077
Autoencoder + DeepWalk	Image, Graph	18952.6573	865.8051	5674.6054	464.9848
HDGE	Graph, Mobility	18731.7773	823.4923	6186.7524	473.5029
MSNE	Graph, Mobility, POI	18323.5702	834.6031	5981.0261	502.7883
MSNE + Tile2Vec	Graph, Mobility, POI, Image	18555.0707	829.6533	6274.0676	511.925
RegionEncoder	Graph, Mobility, POI, Image	**14634.5901**	**659.9702**	**5113.3097**	**451.6293**

over the naive and raw feature models, while the MSNE and HDGE methods are more comparable. This is likely due to how these methods explicitly model spatial and network dynamics of regions.

House Price Prediction For house price prediction, the raw feature approaches both perform quite well relative to other embedding methods. The effect of additional data sources is not as pronounced as in the region popularity experiment. However, we again see that jointly embedding all data sources yields superior performance.

Discussion It is noteworthy that the raw feature baselines (Raw Features, Raw Features + Kmeans) perform quite well in the house price prediction task, as opposed to the region popularity prediction. This could be due to the fact that the total sample size (number of homes) in the house price task is much larger than the region task (number of regions). It is likely the larger sample size partially overcomes the dimensionality of the input. In conclusion, both instances of this experiment show that jointly modeling all modalities result in embedding vectors that better capture regional semantics.

4.5 Ablation Study

In order to distill features across multiple channels of information, the RegionEncoder model relies on several core components embedded in the architecture and loss function. As discussed in section 3, these components are a GCN with skipgram loss, a GCN with reconstruction loss, a denoising AutoEncoder, and a Multi-layer Perceptron to fuse data across modalities.

In this section, we report the effect of each of these components on the house price and region popularity predictions tasks. In order to diagnose the effect of each component independently we train an AutoEncoder (`AutoEncoder`) on just the images alone. We then remove the image data and train two graph convolutional networks separately. We train the GCN using only the skipgram loss term (`GCN-SG`). The GCN is trained again only using the loss term that reconstructs the flow matrix (`GCN-flow`) as in equation 7. Finally, we train the GCN with both skipgram and reconstruction loss (`GCN-SG-flow`).

We also wish to know the effect of the MLP fusion component. This final layer jointly combines the latent image and graph embedding vectors into one latent space. We can test the effectiveness of this operation by comparing to a simple concatenation of the `GCN-SG-flow` and `AutoEncoder` embedding vectors. We refer to this concatenated embedding as (`GCN-SG-flow + AE`). Finally, all of these variants are compared to the proposed RegionEncoder model. The results are reported in figure 4.

In general, we see that jointly embedding all data provides the most robust representation. In New York City, we see that adding more information informs the predictive models. The three single loss approaches, `AutoEncoder`, `GCN-SG`, and `GCN-flow` perform

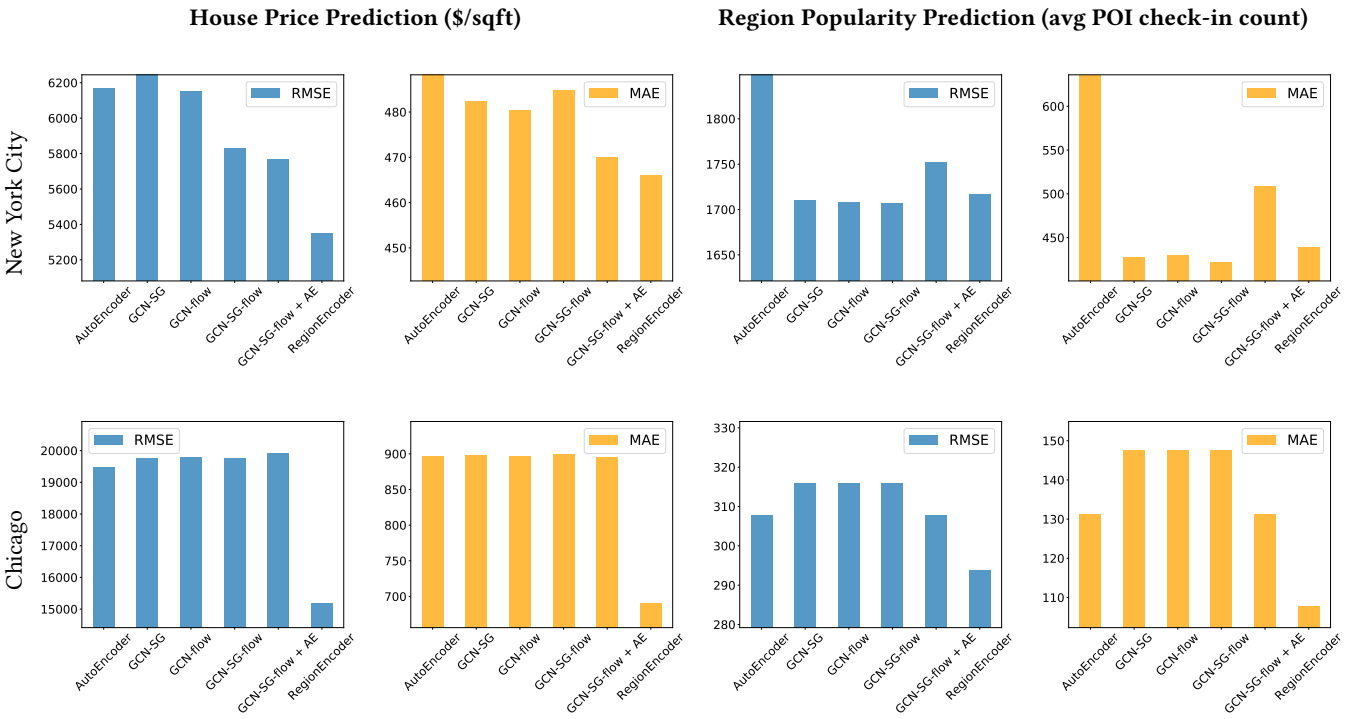


Figure 4: Ablation Study: we hold out each component of the RegionEncoder model and observe the RMSE and MAE of our prediction tasks. (Top row) In the New York City House Price Prediction task, adding information and training jointly improves performance. For region popularity prediction, the image component does no help performance. (Bottom row) In Chicago, training end-to-end across all data types results in superior predictive accuracy on both tasks.

poorly. As more channels of information are added error decreases. For example, both `GCN-SG-flow` and `GCN-SG-flow + AE` outperform the single loss approaches. Furthermore, under both RMSE and MAE jointly learning the features (`RegionEncoder`) provides the best performance.

However, the image information may not be beneficial for predicting region popularity in NYC. Figure 4 shows the models that rely on images (`AutoEncoder`, `GCN-SG-flow + AE`) struggle compared those using spatial and traffic flow graphs. RegionEncoder achieves similar results as `GCN-SG-flow`, but perhaps in spite of the AutoEncoder component. Because this task is primarily performed in Manhattan, a small, densely populated island with many skyscrapers, it is possible that the visual features of many regions are the same. Thus it is difficult to find correlations between these features and popular points across the city. This is not case in Chicago.

In Chicago, the proposed RegionEncoder method dramatically reduces the RMSE and MAE of both predictions tasks. Simply adding more information, or only optimizing one single component of the loss has little or no effect on test error. Additionally, the visual features extracted by RegionEncoder play an important role in both tasks. The final MLP fusion layer appears to offer clear benefit. For both house price and region popularity prediction, RegionEncoder significantly reduces error over the concatenation operation, `GCN-SG-flow + AE`. This finding validates our hypothesis that jointly learning representations across modalities better captures important environmental properties.

4.6 Evaluation of the Latent Space

Finally, we explore the latent space learned by RegionEncoder to demonstrate the model has indeed discovered meaningful semantics of the urban environment. We query three regions from across the latent vector space. For each query region, we visually show its three nearest neighbors in the embedding space, compare their POI distributions, and discuss whether they are connected by an edge in the spatial graph.

4.6.1 Downtown and Grant Park. The first query region we discuss (region id: 33-21) covers part of the downtown area and a portion of Grant Park, a popular downtown destination. We visualize the query region and its neighbors (33-20, 34-22, 36-21) in figure 5. We can see obvious visual similarities: all three neighbors capture parts of the downtown area, different views of the park, and the prominent coast of Lake Michigan. Figure 5 also shows the the distribution of the most frequent POI categories. For the query region, as well as its neighbors, the most frequent POI falls in the office category. Additionally, nearly all regions have 'building' and 'taxi' locations, POI's we might expect in a downtown environment. It is worth noting that each of the neighbors of this query region are also adjacent in the spatial graph.

4.6.2 Suburban Residential. The second query region (id: 48-12) covers a residential area near uptown Chicago, the far north side of the city. The three nearest neighbors discovered in the embedding space are regions 47-12, 49-11, 44-10. Figure 5 shows the regions

all share a high degree similarity in terms of POI locations. All are highly residential, while each includes many other locations for transportation (e.g., bus station, rental car, etc.). Additionally, the images in figure 5 all share similar qualities, and are distinct from the other examples previously discussed. Moreover, the third neighbor, 44-10, is southwest of the query region in terms of geospatial distance, but covers a highly residential, suburban area surrounding a park. Notably, this region is not adjacent in the spatial graph.

4.6.3 Southwest Chicago. The final query region (region id: 15-1), covers an area on the southwest side of Chicago and falls on the edge of our pre-defined window. Distant from downtown, this region has no observed POI or mobility data. Additionally, it's neighbors in the latent space (27-42, 32-39, 27-43) have either sparse or completely missing poi and taxi data. However, all of these regions are visually similar, as shown in 5. In the absence of all other data, RegionEncoder is effective at embedding from visual input alone. None of the neighbors discovered in the embedding space are adjacent in the spatial graph.

5 RELATED WORK

Our work largely touches on three core literature streams: 1) network embedding learning, 2) spatiotemporal representation, and 3) multi-modal representation learning.

5.1 Network Embedding

Network embedding methods are inspired by literature from neural language models in natural language processing. For instance, Word2Vec [16] is a neural network that is trained to predict context words conditional on an input word, which produces distributed vector representations for words. In a similar fashion, DeepWalk [17], generates random walks along a graph and treats them as sentences in a Word2Vec paradigm, embedding the nodes in a latent space. LINE [19] relies on a loss function that preserves first- and second-order proximities between nodes. Recently, Graph Convolutional Networks (GCNs) have been shown to learn effective node- and graph-level representations. For instance, the authors of [4] generalize CNN's to graph-structured data based on local spectral filters and effecient pooling. Additionally, [13] propose a layer-wise propagation rule that approximates localized first-order spectral convolutions in a semi-supervised setting. This approach is also attractive because it jointly embeds graph structure and nodal features. However, because all of these methods are developed on graph data, none of them consider other modalities, such as images.

5.2 Spatio-temporal Representation Learning

This stream of research can be thought of as a generalization of network and word embedding to spatial objects. One early work in this domain [26] proposed a method that learns region representations from temporal dynamics and multi-hop transitions of large-scale taxi flow data. Wang et al [27] develop PTARL, an autoencoder-based framework that embeds GPS traces of driving behavior. More recently, [7] proposed a multi-view framework for learning region representations based on intra- and inter-region auto-correlations. Different from the graph-based methods discussed previously, Tile2Vec [12] takes raw images or remote sensing data as input and seeks to learn representations of regions with a convolutional neural network with triplet loss. Albert et al [2] train CNN's on large-scale satellite image data for land-use classification. They show that these models learn similar representations of geospatial regions across cities and countries, and extract high-level concepts about regions. While some of these methods develop multiple views of the same graph, none jointly consider multiple modalties in their formulation.

5.3 Multi-modal Representation Learning

Finally, our approach for learning representations of spatial regions draws on ideas from work in multi-modal representation learning. DeViSE [6] is an object recognition model that mitigates the need for massive labelled training data by learning visual semantics from raw text. Lazaridou et al [14] extend the SkipGram model to a multi-modal context by allowing the model to predict visual and linguistic context jointly. Mao et al [15] propose a weight sharing strategy for Recurrent Neural Networks that learn word embeddings by incorporating visual information from associated images. None of these methods are developed for spatial data, and therefore do not account for spatial autocorrelation across modalities.

6 CONCLUSION

In this paper, we tackle the LESR problem and proposed a novel method, which we call RegionEncoder, for learning representations of discrete spatial regions. Our method jointly embeds data across different modalities, including satellite images, human mobility, point-of-interest locations, and spatial graphs. By doing so we are able to capture a more holistic view of regions, and provide a better notion of semantic similarity in a geopspatial environment. Across two prediction exercises and two urban environments, we empirically demonstrate that the representations learned by RegionEncoder are effective at improving performance of downstream tasks.

ACKNOWLEDGMENTS

The work was supported in part by NSF awards #1652525 and #1618448. The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing any funding agencies.

REFERENCES

[1] Guillaume Alain and Yoshua Bengio. 2013. What Regularized Auto-Encoders Learn from the Data Generating Distribution. In *ICLR'13*. https://arxiv.org/pdf/1211.4246.pdf

[2] Adrian Albert, Jasleen Kaur, and Marta C. Gonzalez. 2017. Using Convolutional Networks and Satellite Imagery to Identify Patterns in Urban Environments at a Large Scale. In *Proceedings of the 23nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM.

[3] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. 2007. Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems (NIPS)*.

[4] Michael Defferrard, Xavier Bressson, and Pierre Vandergheynst. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *Advances in neural information processing systems (NIPS)*.

[5] Foursquare. 2015. Foursquare Venues Service. (2015). https://developer.foursquare.com/overview/venues.html

[6] Andrea Frome, Greg Corrado, Jonathon Shlens, Samy Bengio, Jeffrey Dean, Marc'Aurelio Ranzato, and Tomas Mikolov. 2013. DeViSE: A Deep Visual-Semantic Embedding Model. In *Advances in Neural Information Processing Systems (NIPS)*.

[7] Yanjie Fu, Pengyang Wang, Le Wu, and Xiaolin Li. 2019. Efficient Region Embedding with Multi-view Spatial Networks: A Perspective of Locality-Constrained

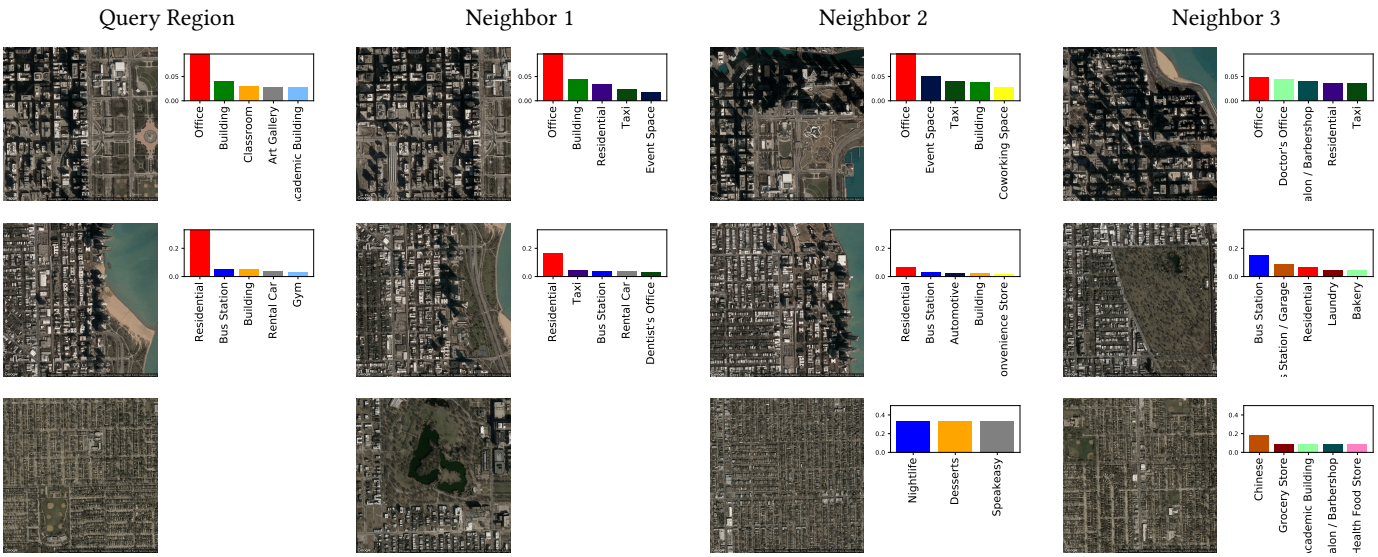


Figure 5: Query regions and nearest neighbors in latent space. For each region we show the satellite image and the distribution of the five most frequent POI categories. (First row) Downtown/Grant Park and neighbors. We see visual similarities of dense urban environments. The POI distributions are similar: high concentration of office and other downtown buildings. (Second row) Suburban residential and neighbors. Each region has a high density of residential and transportation locations. The images show suburban buildings surrounding water or parks. (Third row) Southwest Chicago query and neighbors No observed POI or taxi data for the query region. In the absence of all other data, RegionEncoder learns semantic representations from visual input alone.

Spatial Autocorrelations. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI)*.

[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[9] Google. 2018. Google Static Maps. (2018). https://developers.google.com/maps/documentation/maps-static/intro

[10] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[11] Mikael Henaff, Joan Bruna, and Yann LeCunn. 2015. Deep Convolutional Networks of Graph-Structured Data. In *Advances in neural information processing systems (NIPS)*.

[12] Neal Jean, Sherrie Wang, Anshul Samara, George Azzari, David Lobell, and Stefano Ermon. 2019. Tile2Vec: Unsupervised Representation Learning for Spatially Distributed Data. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI)*.

[13] Thomas Kipf and Max Welling. 2017. Semi-supervised Classification With Graph Convolutional Networks. In *ICLR'17*.

[14] Angeliki Lazaridou, Nghia The Pham, and Marco Baroni. 2015. Combining Language and Vision with a Multimodal Skip-gram Model. In *HLT-NAACL*.

[15] Junhua Mao, Jiajing Xu, Yushi Jing, and Alan Yuille. 2016. Training and Evaluation Multimodal Word Embeddings with Large-scale Web Annotated Images. In *Advances in Neural Information Processing Systems (NIPS)*.

[16] Tomas Mikilov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and Their Compositionality. In *Advances in Neural Information Processing Systems (NIPS)*.

[17] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: Online Learning of Social Representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '14)*. ACM, New York, NY, USA, 701–710. https://doi.org/10.1145/2623330.2623732

[18] Leo Breiman Statistics and Leo Breiman. 2001. Random Forests. In *Machine Learning*. 5–32.

[19] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale Information Network Embedding.. In *WWW*. ACM.

[20] Open Data Team. 2018. New York City Data Dashboard. (2018). https://opendata.cityofnewyork.us/dashboard/

[21] Open Data Team. 2019. Chicago Data Portal. (2019). https://data.cityofchicago.org/

[22] Nicholas Thompson and Ian Bremmer. 2018. Divided we Fail. *Wired* (November 2018).

[23] Waldo Tobler. 1970. A computer movie simulating urban growth in the Detroit region. (1970), 234–240.

[24] Robert Tibshirani Trevor Hastie and Jerome Friedman. 2009. *The Elements of Statistical Learning*. Springer, New York, NY.

[25] Hongjian Wang, Daniel Kifer, Corina Graif, and Zhenhui Li. 2016. Crime rate inference with big data. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 635–644.

[26] Hongjian Wang and Zhenhui Li. 2017. Region Representation Learning Via Mobility Flow. In *Proceedings of CIKM'17*. 10.

[27] Pengyang Wang, Yanjie Fu, Jiawei Zhang, Pengfei Wang, Yu Zheng, and Charu Aggarwal. 2018. You Are How You Drive: Peer and Temporal-Aware Representation Learning for Driving Behavior Analysis. In *Proceedings of the 24nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 2457–2466.

[28] Hua Wei, Guanjie Zheng, Huaxiu Yao, and Zhenhui Li. 2018. IntelliLight: A Reinforcement Learning Approach for Intelligent Traffic Light Control. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2496–2505.

[29] Xiuwen Yi, Junbo Zhang, Zhaoyuan Wang, Tianrui Li, and Yu Zheng. 2018. Deep Distributed Fusion Network for Air Quality Prediction. In *Proceedings of the 24nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 965–973.

[30] Zillow. 2019. Real Estate Valuations in Chicago and New York. (2019). https://www.zillow.com/