convolution layer propagates the features of one-hop neighbors. Our approach effectively executes a spatial regularization based on Tobler's first law of geography stating that "near things are more related than distant things", to battle the spatial feature discontinuity caused by the fine granularity of ZIP code area units.

Even though we are able to construct such a multigraph of ZIP code areas from open data, and use NAICS industry sector growth vector for supervised representation learning, the amount of data is still very limited compared with skip-gram models. Specifically, we only have 33119 ZIP codes within US, and we should only use the feature data and growth supervision from recent years to avoid out-of-date growth trend. This poses another challenge to building our embedding model, since the loss objective function of a deep learning model is typically non-convex and requires a lot of data to train the large quantity of weight parameters. In contrast, we only have a moderate-sized multigraph and thus a simple model with less parameters is preferred to avoid model underfitting, as according to the principle of Occam's razor.

We thus adopt a simple GCN design to extract intermediate features from each category of region input features, which are then integrated to predict the final sector-by-sector growth region embeddings. After testing various ways of feature integration, we find that a simple weighted sum works the best (e.g., as compared to adding another dense layer) thanks to its simplicity to train which aligns with Occam's razor.

Contributions. Our main contributions are as follows:

- We identify open data sources (including government data and third-party data) to effectively construct our region-based multigraph for learning sector-by-sector region embeddings from the perspective of economic growth.
- We follow the principle of Occam's razor to design simple GCN based models that are easy to train on our limited data source without model underfitting, including techniques such as separating feature extraction of different categories of input region features, feature integration by weighted sum, and selection of top-k weighted edges for neighbor feature integration (see Section 4 for details).
- We evaluate our method on real data by comparing it extensively with existing alternative solutions to demonstrate its effectiveness.

Paper Organization. The rest of this paper is organized as follows. Section 2 reviews the related work regarding graph neural networks and region embedding. Section 3 then defines our notations and presents how our multigraph of ZIP code areas is constructed from open data. Section 4 introduces our model design and explains why it is more effective than other alternatives. Finally, Section 5 reports our experimental results and Section 6 concludes this paper.

2 RELATED WORK

2.1 Node Embedding & Graph Neural Networks

Topology-Based Node Embedding. Since deep learning layers take fixed-size tensors (or, a mini-batch of fixed-length vectors) as the input, a number of works propose to embed nodes in a graph into vectors so that graph data become admissible to deep learning tasks such as node classification and graph classification. For example, DeepWalk [27] traverses a graph to generate random walks to capture local structures by neighborhood relations, and then uses the skip-gram model to learn the embedding of each node that are most likely to generate its nearby nodes (or their embeddings) in a walk. This is similar to word2vec [25] except that now sentences become walks and words become nodes. Node2vec [14] further introduce Breadth-Fast-Sampling (BFS) and Depth-First-Sampling (DFS) to control the random behavior. BFS reaches immediate neighbors while DFS prefer node away from the source.

A problem with using DeepWalk and node2vec in our setting is that, these methods only consider graph topology and cannot incorporate input node features. However, in our multigraph, edges only reflect the connection strength between different ZIP code areas, and it is the input node features that propagate along these edges to mutually reinforce each other w.r.t. economic factors.

Graph Neural Networks (GNNs). The notion of GNN was initially outlined in [13] and further elaborated by [29] as a form of recurrent neural network (RNN). They propagate neighbor information in an iterative manner until reaching a stable fixed point. This process is expensive and is recently improved by several studies such as [23]. Encouraged by the success of convolutional neural networks (CNNs) in computer vision which extracts higher-level features from images using a sequence of interleaving convolution layers and pooling layers, recent models focus on adapting these layers to work directly on graph input (rather than tensor input).

Graph Convolutional Networks (GCNs). Graph convolution layers can be divided into two categories, spectral graph convolution and localized graph convolution [34].

Early works mainly focus on spectral graph convolutions, as pioneered by [6]. In a follow-up work, ChebyNet [8] defines graph convolutions using Chebyshev polynomials to remove the expensive Laplacian eigendecomposition. Then, GCN [19] as the state of the art, further simplify graph convolution by a localized first-order approximation, leading to a simple form of

$$H^{(\ell+1)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(\ell)}W^{(\ell)}),$$

where $\tilde{A} = A + I$ is the adjacency matrix with self-connections added, $\tilde{D}$ is a diagonal matrix with $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$, $W^{(\ell)}$ is the weight matrix to train for Layer ℓ, σ is an activation function such as ReLU, and $H^{(\ell)}$ is the state matrix where each row keeps the features of a node at Layer ℓ; $H^{(0)} = X$, i.e., the input matrix where each row keeps the features of a node.

However, spectral methods require operating on the entire graph Laplacian during training which is expensive, though some follow-up works endeavors to mitigate the cost as in FastGCN [7] and SGC [33]. To avoid operating on graph Laplacian, GraphSAGE [16] propose to learn a function $f(.)$ that generates node embeddings by aggregating features from a node's local neighborhood. Specifically, each convolution layer performs two transformations on each node v:

$$\mathbf{h}^{(\ell)}_{\mathcal{N}(v)} \leftarrow f(\{\mathbf{h}^{(\ell-1)}_u \mid \forall u \in \mathcal{N}(v)\}),$$

$$\mathbf{h}^{(\ell)}_v \leftarrow \sigma(\mathbf{W}^{(\ell)} \cdot \text{CONCAT}(\mathbf{h}^{(\ell-1)}_v, \mathbf{h}^{(\ell)}_{\mathcal{N}(v)}) + \mathbf{b}^{(\ell)}),$$

where we denote $\mathcal{N}(v)$ as v's neighborhood. Specifically, we first aggregate features of v's neighbors u, i.e., $\mathbf{h}_u$ from the previous layer

by function $f(.)$; then the aggregated features are concatenated with the old features of v itself, which then pass through a dense layer to obtain v's new features at Layer ℓ. GraphSAGE proposes 3 different choices of aggregating function $f(.)$: mean aggregator, LSTM aggregator and pooling aggregator. For example, the mean aggregator simple averages the features of v and its neighbors.

PinSage [34] further improves the performance of localized graph convolution by only incorporating top-k nodes that exert the most influence on node v into $\mathcal{N}(v)$ rather than using the full neighborhood, which is selected based on node visit frequencies of short random walks starting from v. PinSage also adopts a different aggregation function from GraphSAGE:

$$\mathbf{h}_{\mathcal{N}(v)}^{(\ell)} \leftarrow \text{AVG}(\{\text{ReLU}(\mathbf{Q}^{(\ell)} \cdot \mathbf{h}_u^{(\ell-1)} + \mathbf{q}^{(\ell)}) \,|\, \forall u \in \mathcal{N}(v)\}),$$

$$\mathbf{h}_v^{(\ell)} \leftarrow \text{ReLU}(\mathbf{W}^{(\ell)} \cdot \text{CONCAT}(\mathbf{h}_v^{(\ell-1)}, \mathbf{h}_{\mathcal{N}(v)}^{(\ell)}) + \mathbf{b}^{(\ell)}).$$

Here, AVG(.) is weighted average where the weight for neighbor u equals the L_1 normalized visit counts of u by random walks from v. Note that different from GraphSAGE that directly averages neighbor features, PinSage transforms neighbor features $\mathbf{h}_u^{(\ell-1)}$ by a dense layer plus ReLU activation first before taking their weighted average, which introduces an additional parameters $<\mathbf{Q}^{(\ell)}, \mathbf{q}^{(\ell)}>$ besides $<\mathbf{W}^{(\ell)}, \mathbf{b}^{(\ell)}>$ to each graph convolution layer, allowing more model capacity.

Graph Pooling. Besides convolution, pooling has been studied in the context of graph data as input, which reduces graph size for upper layers (hence reduces their parameter size). This is especially true when going from node embeddings to a representation of the entire graph for the task of graph classification, where global pooling is essential as in [9, 12, 23]. SortPool [37] sorts nodes according to their structural roles within a graph, truncates the sorted embeddings to a fixed size k and which are then fed to the next layer.

Hierarchical pooling aims to learn hierarchical representations to capture structural information. DiffPool [35] coarsens each input graph by learning a differentiable soft assignment of nodes at Layer ℓ to clusters which are essentially nodes of Layer $(\ell + 1)$. gPool [11] achieves performance comparable to DiffPool with less cost. It uses a trainable projection vector $\mathbf{p}$ to project all node features to 1D, and then pick top-k nodes with the largest projected values to form a smaller graph. A gUnpool operation is also proposed so that a U-Net style encoder-decoder network can be constructed for node classification and graph classification. SAGPool [20] uses a self-attention layer to select top-ranked nodes to form a smaller graph, and the layer has the same form as a spectral convolution layer of GCN [19] though the activation function adopted is tanh.

2.2 Region Embedding

The problem of city similarity comparison was studied by [28] which simply model a city as a vector of counts of different venues, which are obtained from Foursquare check-in data. Two cities are compared using the cosine similarity of their vectors. This work find that directly aggregate amenity counts in a city gives slightly different similarity than if amenity counts are first aggregated by subareas in a city and than averaged, which hints that a city-level representation is too coarse to be stable. In fact, different regions in a city can have different economic features and functions. For example, the GDP of Manhattan Borough in New York City is one order of magnitude larger than that of Queens Borough, and [36] identifies different functional zones in Beijing through a topic model over human mobility trajectories. We thus consider the finer granularity of ZIP code areas in this paper.

The work of [31] goes beyond modeling a region simply with a frequency vectors of PoI categories, to incorporate features that represent the spatial distribution of PoIs in the region. Specifically, the features are computed based on the distances of each object to five predefined reference points in the region. Since such hand-crafted spatial features do not consider the relative locations between objects, [24] addresses this problem by leveraging a deep learning model that is similar to FaceNet [30] to learn a ranking metric for comparing rectangular regions that are modeled as images with objects in grid cells (regarded as pixels).

The work of [10] use Foursquare check-in data to derive features of venues (e.g., number of check-ins, number of likes) so that each venue is associated with a 30-dimensional feature vector. The work considers neighborhoods as regions, and each neighborhood is represented by its set of venues (and their feature vectors). Different neighborhoods are compared using Earth mover's distance, where the distance between venues are computed by a learned Mahalanobis distance of their feature vectors. Another work, [18], learns a generative process for the geotags inside a neighborhood from Flicker photos, and use the distribution of sampling local tags of a neighborhood as its feature vector.

Our problem setting is different from those of the above works which utilize objects (e.g., PoIs, check-ins, geotags) inside a region to extract features, as our input features for each region are directly available from ACS which are related to economy rather than PoI distribution or visiting preference. Also, we capture region relationships using a multigraph so that GCN can learn node (i.e., region) embeddings, while previous works directly consider distance between region representations without resorting to a graph data model.

3 NOTATIONS AND THE MULTIGRAPH

A key difference of our work from those reviewed in Section 2.2 is that we not only consider the content information of each individual region, but also consider the structural relationships between regions. In this section, we define our notations and explain how we construct our multigraph over ZIP code areas; these definitions will provide sufficient context for understanding our methods to be described in Section 4.

We consider the geographic unit of ZIP code areas. Specifically, we denote the set of all ZIP code areas within US by $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{33119}\}$, where $\mathbf{x}_i$ is the input feature vector of the i-th ZIP code area. Recall that we collected 4 categories of features from American Community Survey (ACS) including 82 demographic features, 150 social features, 114 economic features, 142 housing features. We organize features in $\mathbf{x}_i$ accordingly into 4 segments:

$$\mathbf{x}_i = \text{CONCAT}(\mathbf{x}_{i,c_1}, \mathbf{x}_{i,c_2}, \mathbf{x}_{i,c_3}, \mathbf{x}_{i,c_4}),$$

where $\mathbf{x}_{i,c}$ refers to the segment of all features of the i-th ZIP code area that belong to Category c. We denote the set of all feature categories by $C = \{c_1, c_2, c_3, c_4\}$.

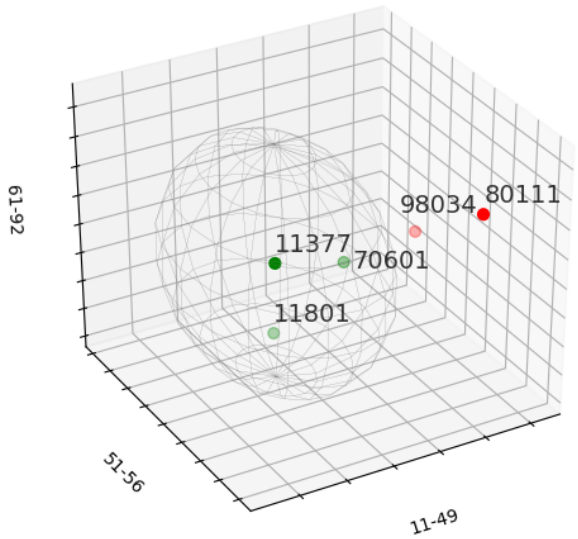

Figure 9: Nearest Neighbors of ZIP Code 11377

difference of the 3 dimensions, we use min-max normalization to normalize ($sum_{11-49}, sum_{51-56}, sum_{61-92}$), and Figure 9 shows the 3D coordinates of 5 ZIP code areas.

Given a query ZIP code 11377, ZIP code areas 11802 and 70601 are more similar to 11377 than 98034 and 80111, and after checking them we find that the sector growth similarity is well captured. The 3D visualization is, however, more intuitive, and sectors can be grouped in different ways depending on the needs. For example, Sectors 44 – 45 (Retail Trade) and 72 (Accommodation and Food Services) can be in the same group, if we are finding a house so that daily life is convenient.

6 CONCLUSION

We described a novel GCN-based model for learning economic growth sector-by-sector predictions from ACS features of ZIP code areas expanded with spatial relations. The design principle is to reduce the number of parameters to train, and to decouple the feature extraction process of different types of region and edge types. Experiments confirm that our learned region embeddings have a high quality than existing baselines.

REFERENCES

[1] Visited in 2020. American Community Survey. https://www.census.gov/programs-surveys/acs.

[2] Visited in 2020. Internal Point. https://www.census.gov/programs-surveys/geography/about/glossary.html.

[3] Visited in 2020. NAICS. https://www.naics.com/business-lists/counts-by-naics-code/#countsByNAICS.

[4] Visited in 2020. ZIP Codes Business Patterns (ZBP) APIs. https://www.census.gov/data/developers/data-sets/cbp-nonemp-zbp/zbp-api.html.

[5] John M. Abowd. 2018. The U.S. Census Bureau Adopts Differential Privacy. In *SIGKDD*, Yike Guo and Faisal Farooq (Eds.). ACM, 2867.

[6] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2014. Spectral Networks and Locally Connected Networks on Graphs. In *ICLR*.

[7] Jie Chen, Tengfei Ma, and Cao Xiao. 2018. FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling. In *ICLR*. OpenReview.net.

[8] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *NIPS*. 3837–3845.

[9] David Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P. Adams. 2015. Convolutional Networks on Graphs for Learning Molecular Fingerprints. In *NIPS*. 2224–2232.

[10] Géraud Le Falher, Aristides Gionis, and Michael Mathioudakis. 2015. Where Is the Soho of Rome? Measures and Algorithms for Finding Similar Neighborhoods in Cities. In *ICWSM*. AAAI Press, 228–237.

[11] Hongyang Gao and Shuiwang Ji. 2019. Graph U-Nets. In *ICML (Proceedings of Machine Learning Research)*, Vol. 97. PMLR, 2083–2092.

[12] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. Neural Message Passing for Quantum Chemistry. In *ICML (Proceedings of Machine Learning Research)*, Vol. 70. PMLR, 1263–1272.

[13] Marco Gori, Gabriele Monfardini, and Franco Scarselli. 2005. A new model for learning in graph domains. In *IEEE International Joint Conference on Neural Networks*, Vol. 2. IEEE, 729–734.

[14] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *SIGKDD*. ACM, 855–864.

[15] Guimu Guo, Jalal Majed Khalil, Da Yan, and Virginia P. Sisiopiku. 2019. Realistic Transport Simulation: Tackling the Small Data Challenge with Open Data. In *IEEE BigData*. IEEE, 4512–4519.

[16] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NIPS*. 1024–1034.

[17] Elad Hoffer and Nir Ailon. 2015. Deep metric learning using Triplet network. In *ICLR, Workshop Track Proceedings*.

[18] Mohamed Kafsi, Henriette Cramer, Bart Thomee, and David A. Shamma. 2015. Describing and Understanding Neighborhood Characteristics through Online Social Media. In *WWW*. ACM, 549–559.

[19] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

[20] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. 2019. Self-Attention Graph Pooling. In *ICML (Proceedings of Machine Learning Research)*, Vol. 97. PMLR, 3734–3743.

[21] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper Insights Into Graph Convolutional Networks for Semi-Supervised Learning. In *AAAI*. AAAI Press, 3538–3545.

[22] Xia Li, Wei Li, Yuqun Zhang, and Lingming Zhang. 2019. DeepFL: integrating multiple fault diagnosis dimensions for deep fault localization. In *ISSTA*. ACM, 169–180.

[23] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel. 2016. Gated Graph Sequence Neural Networks. In *ICLR*.

[24] Yiding Liu, Kaiqi Zhao, and Gao Cong. 2018. Efficient Similar Region Search with Deep Metric Learning. In *SIGKDD*. ACM, 1850–1859.

[25] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS*. 3111–3119.

[26] Lionel M. Ni, Yunhao Liu, Yiu Cho Lau, and Abhishek P. Patil. 2004. LANDMARC: Indoor Location Sensing Using Active RFID. *Wireless Networks* 10, 6 (2004), 701–710.

[27] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: online learning of social representations. In *SIGKDD*. ACM, 701–710.

[28] Daniel Preotiuc-Pietro, Justin Cranshaw, and Tae Yano. 2013. Exploring venue-based city-to-city similarity measures. In *UrbComp@KDD*. ACM, 16:1–16:4.

[29] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The Graph Neural Network Model. *IEEE Trans. Neural Networks* 20, 1 (2009), 61–80.

[30] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. FaceNet: A unified embedding for face recognition and clustering. In *CVPR*. IEEE Computer Society, 815–823.

[31] Chang Sheng, Yu Zheng, Wynne Hsu, Mong-Li Lee, and Xing Xie. 2010. Answering Top-k Similar Region Queries. In *DASFAA (Lecture Notes in Computer Science)*, Vol. 5981. Springer, 186–201.

[32] Daheng Wang, Meng Jiang, Qingkai Zeng, Zachary Eberhart, and Nitesh V. Chawla. 2018. Multi-Type Itemset Embedding for Learning Behavior Success. In *SIGKDD*. ACM, 2397–2406.

[33] Felix Wu, Amauri H. Souza Jr., Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. 2019. Simplifying Graph Convolutional Networks. In *ICML (Proceedings of Machine Learning Research)*, Vol. 97. PMLR, 6861–6871.

[34] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *SIGKDD*. ACM, 974–983.

[35] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. 2018. Hierarchical Graph Representation Learning with Differentiable Pooling. In *NeurIPS*. 4805–4815.

[36] Nicholas Jing Yuan, Yu Zheng, Xing Xie, Yingzi Wang, Kai Zheng, and Hui Xiong. 2015. Discovering Urban Functional Zones Using Latent Activity Trajectories. *IEEE Trans. Knowl. Data Eng.* 27, 3 (2015), 712–725.

[37] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. 2018. An End-to-End Deep Learning Architecture for Graph Classification. In *AAAI*. AAAI Press, 4438–4445.