GAErryChain

Abishrant Panday
abishrantpanday@college.harvard.edu

Amir Shanehsazzadeh
amirshanehsazzadeh@college.harvard.edu

William Zhang
william_zhang@college.harvard.edu

April 28, 2021

Abstract

The practice of gerrymandering, which allows for a populace's votes to be diluted or concentrated unfairly, is prevalent in representative governments where electoral boundaries are redrawn. In the United States, due to the 1986 decision in *David v. Bandemer*, where it was deemed that there was no practicable test for gerrymandering, a prominent avenue of research in the field centers on detection metrics. Work by DeFord et al. [1] utilized MCMC sampling to efficently generate districts, which can then be used to determine whether a specific redistricting plan is unlikely relative to the possibilities. These algorithms, called **Flip** and **ReCom**, have runtimes on the order of minutes or hours per sample, however. In our work, we thus attempt to accelerate this sampling process by training a graph autoencoder (GAE) model, called GAErryChain. We find that GAErryChain is able to generate some degree of contiguous redistricting with significant computation improvements as well. We note that GAErryChain is currently unsuitable for actual sampling and we provide methods which can improve its performance.

1 Introduction

In many forms of representative government, electoral boundaries are imposed in order to divide up a voting population into districts, which can then elect individuals into a Senate or Parliament. In the United States, the process of drawing these boundaries, called redistricting, occurs every ten years and is often handled by each state's legislatures and governors.

Redistricting has major consequences on election outcomes because it alters the voter composition and size of each district, thus possibly concentrating or diluting support for a particular candidate. The act of using this process to establish an unfair advantage is called *gerrymandering* and it has seen frequent use in diluting the voting power of minority groups, increasing support for a preferred party, and maintaining control for incumbent politicians [2].

However, due to the ruling in *David v. Bandemer*, a 1986 Supreme court case—which deemed gerrymandering harmful to norms of fair representation but noted that without a practicable test, the issue itself was not justiciable [3]—a prominent avenue of current research in the field involves the creation and study of metrics that can indicate the existence of a partisan gerrymander.

In this paper, we extend the work of DeFord et al. [1], which provides a method of analyzing different redistricting plans based on their properties relative to a representative sample of possible plans. Specifically, they define a new family of Markov chain Monte Carlo (MCMC) random walks, called **ReCom** Markov Chains, to sample the space of possible district partitions. This sampling, however, is computationally costly and generating districting plans is time-intensive. Our work thus aims to speed up their sampling through the use of neural networks, specifically a graph auto-encoder architecture (GAE).

Section 2 examines the current literature in the field related to our work. Section 3 then introduces our experimental methods. Section 4 gives an overview of important results and is followed by a discussion in Section 5.

2 Related Literature

Within the space of gerrymandering detection—and general legislative fairness—one metric that has seen widespread use is *partisan symmetry*. As described by Gelman and King in 1994 [4], this measure determines whether multiple parties perform identically given the same vote share.

Work by Stephanopoulos et al. in 2015 [5] forwarded the idea of an *efficiency gap* in identifying partisan gerrymanders. This statistic is based on the difference in wasted votes, those which don't affect the outcome, between the parties. The paper argues that with parties of equal support, a redistricting that is non-partisan would have an efficiency gap of zero and proposes a threshold of .08, above which a plan would be considered gerrymandered.

A third way of determining possible gerrymanders is through the geometry of the districts, with multiple definitions of *compactness* being proposed: Reock in 1961 [6] posited using the ratio of a district's area to the area of the smallest circle containing the district. This idea was extended by Niemi et al. in 1990 [7] to the ratio of a district's area to the area of its convex hull. Finally, Polsby and Popper in 1991 [8] utilized the a ratio proportional to district area over the square of perimeter. Although these metrics have been frequently used in confirming the visual intuition of gerrymandering, Barnes and Solomon in 2018 [9] showed that they are affected by variables irrelevant to fairness or civil law.

Finally, the fourth major, although not disjoint, paradigm for gerrymandering detection is the use of sampling to determine the space of possible districting plans and, based on that, determine whether a particular plan is skewed relative to the overall distribution. These methods use Markov chain Monte Carlo (MCMC) sampling, which, as shown by Diaconis in 2009 [10], have been successfully applied in fields from cryptography to physics. DeFord et al. [1] propose the use of MCMC by framing redistricting as a graph partition problem. They examine a natural MCMC approach called **Flip**, which randomly reassigns nodes one at a time. The paper then forwards the **ReCom** Markov chain family of random walks, which combines two districts and forms a new plan by repartitioning them. We use the term **ReCom** to specifically refer to the spanning tree bipartitioning method of recombination. It is shown that **ReCom** provides many desirable properties within the real of redistricting, including the fact that **ReCom** chains tend to produce more compact partitions.

3 Methods

3.1 Data

Data was generated using the Julia implementation of GerryChainJulia[1], which provided a computational advantage over its Python counterpart. Possible redistrictings of Pennsylvania[2] were computed, because of its real-world value as a battleground state in presidential elections and as one with gerymandering concerns. Additionally, Pennsylvania provided a moderately sized example, where its 18 districts were numerous enough to build a robust model while still allowing samples to be computed in a reasonable amount of time.

The **ReCom** algorithm was used to generate the training set of redistrictings, referred to as partitions moving forward. A $10,000$-step **ReCom** chain was computed, in line with the convergence empirics of DeFord et al. [1], 100 times each for 7 different initial seed redistricting plans. Every 100^{th} partition was stored from each run, giving us $7 \cdot 100 \cdot 101 = 70,700$ partitions to use for training and testing our model. These initial redistricting plans were chosen to be 538GOP, 538DEM, 538CMPCT, three plans from FiveThirtyEight favoring Republicans, Democrats, and Compactness respectively, CD_2011, the actual congressional districts from the 2011 enacted map, REMEDIAL, the congressional districts in the 2018 remedial enacted plan, GOV, congressional districts in the Governor's 2018 proposal, and TS, the districts in the Turzai-Scarnati plan, proposed by Pennsylvania House Speaker Mike Turzai and Senate President Pro Tempore Joe Scarnati.

This transformation from a state map to graph, which is required for both the MCMC process and subsequent graph auto-encoder embedding, is made possible by the fact that district plans rarely cut through census blocks and precincts, which allows a state's population to be divided into nodes with sufficient granularity.

For each partition, the data includes characteristics such as population and demographics at the node-level, along with high-level characteristics such as the *consistent advantage* (mean-median difference) metric and the *efficiency gap*.

[1]github.com/mggg/GerryChainJulia
[2]https://github.com/mggg-states/PA-shapefiles

3.2 Model

Our model consists of a graph auto-encoder (GAE) that extends many properties of variational graph auto-encoders (VGAE), introduced by Kipf and Welling in 2016 [11]. These are graphical analogues of variational autoencoders (VAE) [12], which are model frameworks that are used to generate alterations of existing data. A VAE is thus a model that attempts to learn to encode an input distribution into a lower dimensional latent space while simultaneously learn to decode from this latent space to the original distribution. Broadly, this is achieved by having the encoder output a vector of means μ and standard deviations $\ln(\sigma)$, thus allowing the decoder to learn that a range of variations in the latent space correspond to a particular sample.

Our model architecture, which is summarized in Figure 5 in the Appendix, is as follows: Given a particular graph state, the encoder is fed a description of the graph structure, A, and a description of node features, X. The encoder itself is a graph isomorphism network (GIN) [13], a powerful type of graph neural network (GNN) which updates the node representations as

$$x_i^k = h_\Theta\left((1+\epsilon)x_i^{k-1} + \sum_{j\in\mathcal{N}(i)} x_j^{k-1}\right) \tag{1}$$

Where h_Θ is a neural network, which in our case consists of three linear transformations and two $GELU$ activation functions, that maps node features from the input to appropriate output shape. We also experimented with Graph Convlutional Networks [14], defined by

$$x_i^k = \Theta \sum_{j\in\mathcal{N}(i)\cup\{i\}} \frac{e_{ij}}{\sqrt{\hat{d}_i\hat{d}_j}} x_j^{k-1} \tag{2}$$

where $\hat{d}_i = 1 + \sum_{j\in\mathcal{N}(i)} e_{ji}$. However, we found that the GCN model was harder to optimize.

We also leverage residual connections [15], LayerNorm [16], and dropout [17]. Residual connections and LayerNorm were crucial in order for the model to train, and dropout helped reduce the gap between the training and validation loss.

3.3 Training

The model was trained for 15 epochs using AMSGrad [18] with a learning rate of 0.001. The loss function is a combination of the standard VAE reconstruction loss, the Kullback-Leibler divergence between the standard normal distribution and the learned distribution, and a discontiguity penalty term:

$$L(x) = \mathbb{E}_x[\log p(\hat{x})] + \lambda_1 \cdot \mathrm{KL}(\mathcal{N}(0,I);\mathcal{N}(\mu_k,\sigma_k)) + \lambda_2 \cdot L_{\text{cont}}(x) \tag{3}$$

where L_{cont} is defined as a sum over the edges in the graph E:

$$L_{\text{cont}}(x) = \sum_{(v_1,v_2)\in E} L_1(\log p_x(v_1), \log p_x(v_2)) \tag{4}$$

with $\log p_x(v_1) \in \mathbb{R}^{64}$ representing the logits vector that node v_1 is mapped to by the model. The intuition behind this penalty is that adjacent nodes are likely to be mapped to similar assignment distributions in a contiguous assignment. We tried a few combinations of loss hyperparameters, specifically with $\lambda_1 \in [\frac{1}{64}, \frac{1}{4}]$ and $\lambda_2 \in [3, 20]$. A plot of the loss curves for a specific selection of loss hyperparameters is shown in Figure 1.

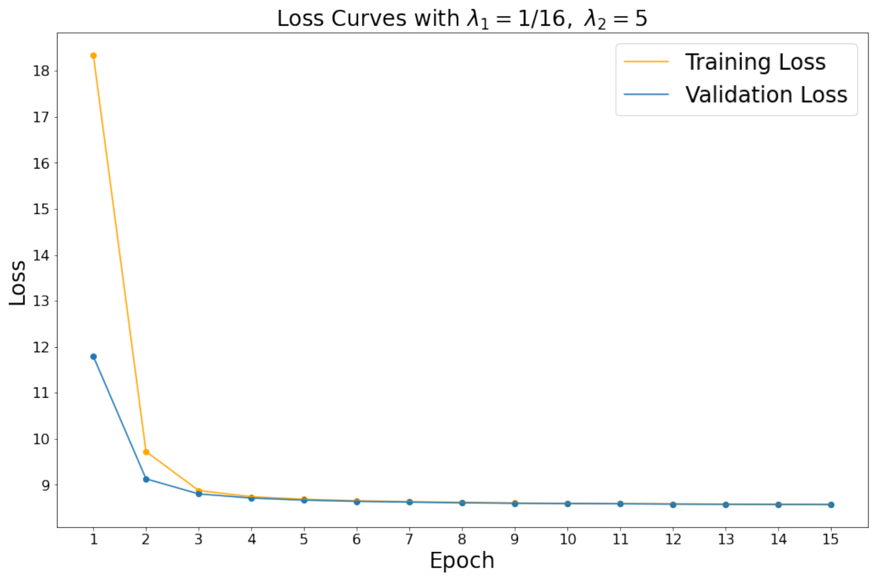


Figure 1: Training and validation loss curves with $\lambda_1 = \frac{1}{16}$ and $\lambda_2 = 5$.

4 Results

In order to determine whether our model would be useful as a sampling augmentation, we sought to determine if it exhibited the state-level aggregation into contiguous districts that is expected of a districting plan. As seen in Figure 2, our baseline model, (b) showed district aggregation in dense census tracts that was significant relative to a random assignment (a). Moreover, incorporating the discontiguity penalty at varying levels, which promoted partitioning the graph into more contiguous districts, resulted in a strict improvement, as shown in (c,d). We believe that further fine-tuning this penalty, as well as incorporating constraints on population deviation between districts, would result in improved data generation.

In addition, although our currently generated graphs are inadequate for actual sampling, the GAE implementation shows significant computation improvements over the **ReCom** algorithm. As seen

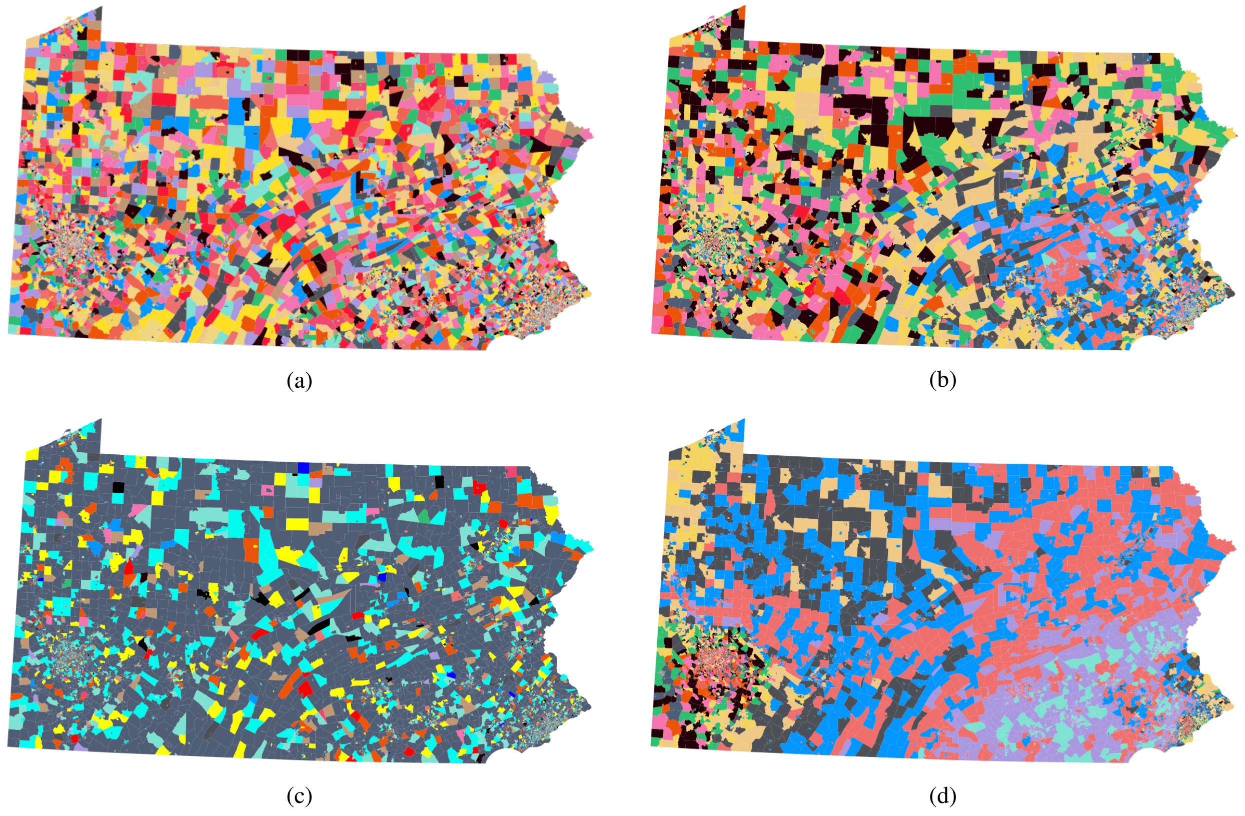


Figure 2: Partitions of Pennsylvania: (a) Random assignment, (b) Base model result, (c, d) Model results with discontiguity penalty.

in Figure 3, the time to generate redistricting plans using **ReCom** was linear in the number of plans generated, with each requiring approximately 6 minutes for Pennsylvania. Our model however, even when including the fixed time cost of training, is faster than **ReCom** at around 25 generated maps, meaning for applications at a real-world level, where the sampling requirement is much larger, we believe there to still exist a performance advantage to a better augmented GAE approach.

5 Discussion

5.1 Engineering Challenges

We believe that there are three key data and model-level engineering hurdles, which, if addressed, would improve the performance of our model. First, we note that a major reason why our output graphs did not show extensive higher-level structuring, such as contiguous districts, is because the $2-$layer GIN models used for the encoder and decoder are not deep enough to learn higher order propagation. This shallow structure inhibits the model from effectively 'communicating' between nodes that are far away from each other, and thus leads to small local clusters as opposed to larger ones. Second, our biggest concern with the initial results was the discontiguity of the output maps, where district labels were not as clustered as hoped. While our inclusion of the discontiguity penalty

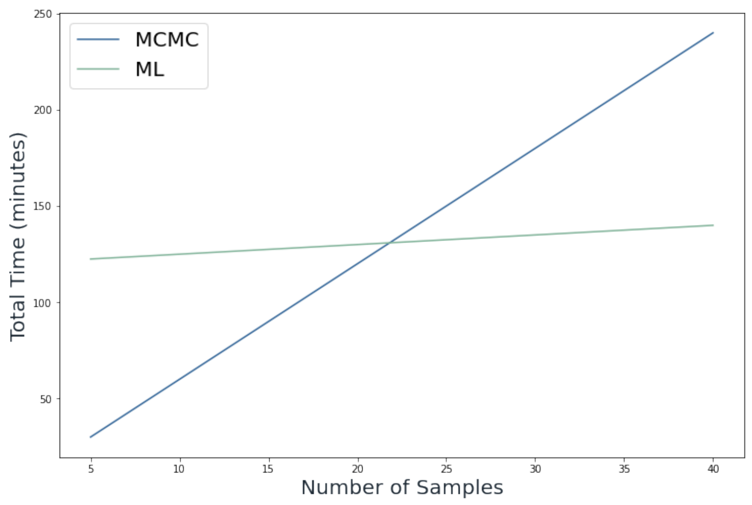


Figure 3: Time to generate samples for **ReCom** (6 minutes per sample using GerryChainJulia) and our model (fixed training cost + $\sim$0.165 seconds per sample).

showed effectiveness, as noted in Section 4, we believe a more nuanced application of this loss can lead to further improvements, since currently, increasing the weighing of this penalty, λ_2 eventually results in a single prediction for all nodes, as shown in Figure 4 below. Finally, the third concern

Figure 4: Result of increasing λ_2 arbitrarily, which leads to all nodes being classified into the same district.

was that our model did not preserve permutation-invariance of the labels. District assignments are fundamentally permutation-invariant, since swapping the labels of districts i and j has no actual effect on election outcomes. This property was preserved in the data-generation process and we believe incorporating it into our model would improve clustering performance.

5.2 Future Directions

There are multiple avenues of exploration that we believe would improve our sampling performance. The first is on the model front, where the size of our GIN encoder and decoder were limited by compute constraints on personal hardware. As noted above, increasing the depth of these models should lead to a higher level of dependence between distant nodes. On the same front, we are currently implementing a virtual node into our graph network, as studied in Pham et al. [19], which is an abstract node with connections to all nodes in the graph. This thus decreases the distance between nodes in the network and allows for a more complex sample space to be studied. Finally, adding population-level constraints should implicitly help aggregation of nodes into contiguous districts. Thus, one can directly add a penalty for deviating from a district population mean, which leads districts to be of similar population and may improve clustering behavior.

References

[1] D. DeFord, M. Duchin, and J. Solomon, “Recombination: A family of markov chains for redistricting.,” *arXiv preprint arXiv:1911.05725*, 2019.

[2] M. Bernstein and M. Duchin, “A formula goes to court: Partisan gerrymandering and the efficiency gap.,” *Notices of the AMS*, vol. 64, no. 9, pp. 1020–1024, 2017.

[3] *Davis v. Bandemer*. 478 U.S. 109 1986.

[4] A. Gelman and G. King, “Enhancing democracy through legislative redistricting.,” *American Political Science Review*, pp. 541–559, 1994.

[5] N. Stephanopoulos and E. McGhee, “Partisan gerrymandering and the efficiency gap.,” *University of Chicago Law Review*, no. 82, pp. 831–900, 2015.

[6] E. Reock, “A note: Measuring compactness as a requirement of legislative apportionment.,” *Midwest Journal of Political Science*, vol. 5, no. 1, p. 70–74, 1961.

[7] R. Niemi, B. Grofman, C. Carlucci, and T. Hofeller, “Measuring compactness and the role of a compactness standard in a test for partisan and racial gerrymandering,” *The Journal of Politics*, vol. 52, no. 4, p. 1155–1181, 1990.

[8] D. Polsby and R. Popper, “The third criterion: Compactness as a procedural safeguard against partisan gerrymandering.,” *Yale Law Policy Review*, vol. 9, no. 2, p. 301–353, 1991.

[9] R. Barnes and J. Solomon, “Gerrymandering and compactness: Implementation flexibility and abuse.,” *arXiv preprint arXiv:1803.02857*, 2018.

[10] P. Diaconis, “The markov chain monte carlo revolution.,” *Bulletin of the American Mathematical Society*, vol. 46, no. 2, pp. 179–205, 2009.

[11] T. N. Kipf and M. Welling, “Variational graph auto-encoders,” *arXiv preprint arXiv:1611.07308*, 2016.

[12] D. P. Kingma and M. Welling, “An introduction to variational autoencoders,” *Foundations and Trends in Machine Learning*, vol. 12, no. 4, pp. 307–392, 2019.

[13] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, “How powerful are graph neural networks?,” 2019.

[14] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” *arXiv preprint arXiv:1609.02907*, 2017.

[15] S. Targ, D. Almeida, and K. Lyman, “Resnet in resnet: Generalizing residual architectures,” *arXiv preprint arXiv:1603.08029*, 2016.

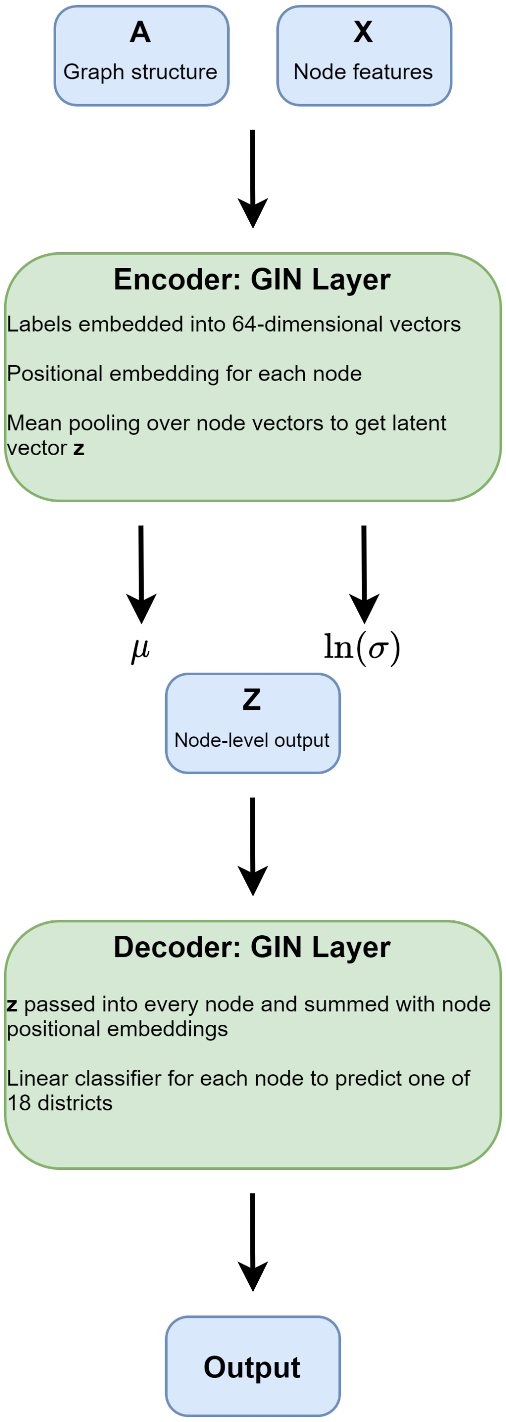
[16] J. L. Ba, J. R. Kiros, and G. E. Hinton, “Layer normalization,” *arXiv preprint arXiv:1607.06450*, 2016.

[17] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[18] S. J. Reddi, S. Kale, and S. Kumar, “On the convergence of adam and beyond,” *arXiv:1904.09237*, 2019.

[19] T. Pham, T. Tran, H. Dam, and S. Venkatesh, “Graph classification via deep learning with virtual nodes,” 2017.

6 Appendix



A
Graph structure
X
Node features
Encoder: GIN Layer
Labels embedded into 64-dimensional vectors
Positional embedding for each node
Mean pooling over node vectors to get latent vector **z**
μ
$\ln(\sigma)$
Z
Node-level output
Decoder: GIN Layer
z passed into every node and summed with node positional embeddings
Linear classifier for each node to predict one of 18 districts
Output

Figure 5: Graphical depiction of the GAE model.