Invertible Neural Networks for Graph Prediction

Chen Xu
Georgia Tech
cxu310@gatech.edu

Xiuyuan Cheng
Duke University
xiuyuan.cheng@duke.edu

Yao Xie
Georgia Tech
yao.xie@isye.gatech.edu

Abstract

Graph prediction problems prevail in data analysis and machine learning. The inverse prediction problem, namely to infer input data from given output labels, is of emerging interest in various applications. In this work, we develop *invertible graph neural network* (iGNN), a deep generative model to tackle the inverse prediction problem on graphs by casting it as a conditional generative task. The proposed model consists of an invertible sub-network that maps one-to-one from data to an intermediate encoded feature, which allows forward prediction by a linear classification sub-network as well as efficient generation from output labels via a parametric mixture model. The invertibility of the encoding sub-network is ensured by a Wasserstein-2 regularization which allows free-form layers in the residual blocks. The model is scalable to large graphs by a factorized parametric mixture model of the encoded feature and is computationally scalable by using GNN layers. We study the invertibility of flow mapping based on theories of optimal transport and diffusion process. The proposed iGNN model is experimentally examined on synthetic data, including the example on large graphs, and the empirical advantage is also demonstrated on real-application datasets of solar ramping event data and traffic flow anomaly detection.

1 Introduction

Graph prediction is an important topic motivated by various applications, e.g., protein-protein interaction networks [38], wind power prediction [39], and user behavior modeling in social networks [7]. In the so-called inverse of a graph prediction problem, one would like to infer the input graph nodal features X given an outcome response Y. Such a problem is of interest in various real-world applications, e.g., molecular design [33] and power outage analysis [41, 1]. The inverse graph prediction problem is the focus of the current work, for which we develop an invertible deep model that can be efficiently applied to graph data.

More formally, The forward prediction problem is to learn the conditional probability of $p(Y|X)$. This discriminative task can be done by a conventional classification model. The inverse prediction problem is to learn the conditional probability of $p(X|Y)$ and to *generate* samples X from it. This generative task is challenging when data X is in high dimensional space (e.g., graph nodal features that scales linearly with graph size), where a grid of X can not be efficiently constructed. In addition, in the case of categorical response, the graph label Y assigns one of the K-class labels to each node, which makes the total possible outcomes K^N many. Thus the inverse prediction problem on graph data poses both modeling and computational challenges when scalability to large graphs is needed.

In this work, we develop a deep generative model for the conditional generation task of the inverse prediction problem on graphs. Unlike previous conditional generative models [30, 20, 3, 4, 2], which typically concatenate the prediction label Y with the random code Z as input or rely on curated forms of neural network (NN) layers to ensure invertibility, we propose to encode the input data X one-to-one by an invertible network to an intermediate feature H, from which the label Y

NeurIPS 2022 New Frontiers in Graph Learning Workshop (NeurIPS GLFrontiers 2022).
* Full paper of the work is available at https://arxiv.org/abs/2206.01163.

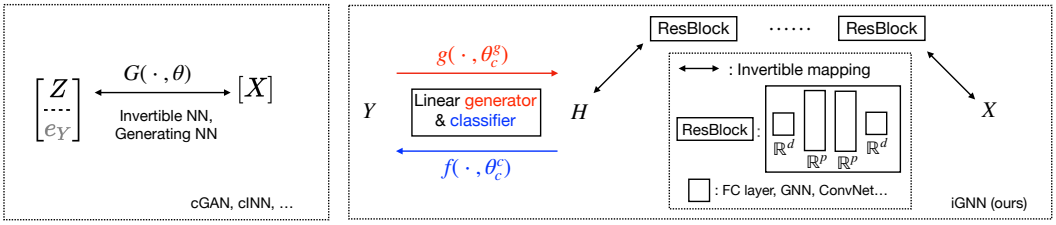


Figure 1: Comparison of existing conditional generative neural network models (left) and the proposed iGNN model (right). Most current approaches concatenate the encoded prediction label Y (e.g. one-hot encoding) as an additional input to the generative network G. Our model takes a two-step approach: a one-to-many mapping g from label Y to intermediate feature H by a Gaussian mixture model (which allows classification from H to Y by f), and a one-to-one mapping from H to input data X.

can be predicted using a linear classifier, and, in the other direction, H can also be generated from Y by a parametric mixture model. The framework of our model is shown in Figure 1. Because the general (non-graph) data case can be seen as a special case of graph data where the graph only has one node, we call the proposed model invertible Graph Neural Network (iGNN), as a unified name. In summary, the contributions of the work are

- We propose a two-step procedure, Y-to-H and H-to-X, to tackle the generative task of inverse prediction problem viewed as a conditional generation problem and develop an invertible flow model consisting of two subnetworks accordingly. The model is made scalable to graph data by a factorized formulation of the parametric mixture model $H|Y$ and the GNN layers in the invertible flow network between H and X.
- We introduce Wasserstein-2 regularization of the invertible flow network, which is computationally efficient and compatible with free-form layer types including the GNN layers. The effect on preserving invertibility is backed by OT theory and verified in practice.
- We theoretically study the invertibility of flowing mapping based on theories of optimal transport and diffusion process.
- The proposed iGNN model is applied to both simulated and real-data examples, showing improved generative performance over alternative conditional generation models.

2 Method

1. Inverse of prediction as conditional generation. The overall framework is to (end-to-end) train a network consisting of two sub-networks: the first sub-network maps invertibly from X to an intermediate representation H, and the second sub-network maps from H to label Y, which is a classifier and loses information. Specifically,

- H-Y classification sub-network. We model $H|Y$ by a Gaussian mixture model. The parametric form of $H|Y$ contains trainable parameter θ_c^g, and the generation of $H|Y$ is by sampling the corresponding mixture component accordingly. The prediction of label Y from H can be conducted by a linear classifier parametrized by θ_c^c. The trainable parameters in this sub-network are denoted as $\theta_c = (\theta_c^g, \theta_c^c)$.
- X-H invertible sub-network. The invertible mapping from X to H is by a flow ResNet in $\mathbb{R}^d$. The sub-network parameters are denoted as $\theta = \{\theta_l, l = 1, \cdots, L\}$, where L is the number of residual blocks, and the network mapping is denoted as F_θ. The generation of $X|Y$ is by inversely mapping $X = F_\theta^{-1}(H)$ once H is sampled according to $p(H|Y)$ parametrized by θ_c^g.

The end-to-end training objective of the proposed network can be written as

$$\min_{\{\theta, \theta_c\}} \mathcal{L}_g + \mu \mathcal{L}_c + \gamma \mathcal{W}, \tag{1}$$

where $\mathcal{L}_g$, $\mathcal{L}_c$ and $\mathcal{W}$ are the generative loss, the classification loss and the Wasserstein-2 regularization (cf. Eq. (7)) , respectively. The scalars $\mu, \gamma \geq 0$ are penalty factors.

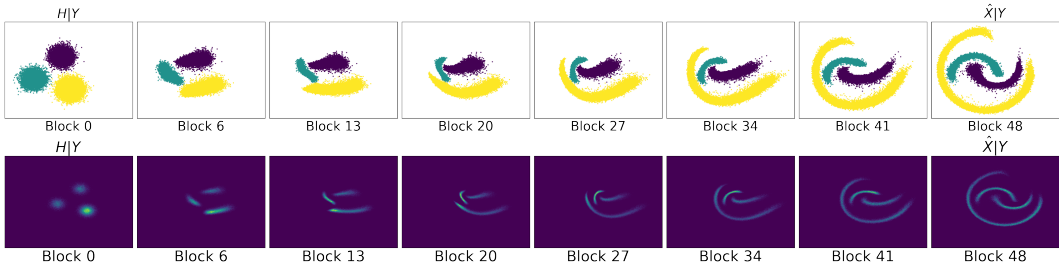


Figure 2: Flow map learned by iGNN model that transports three-class data in $\mathbb{R}^2$ to a three-component Gaussian mixture and back. The distribution $X|Y$ has $(0, 22, 0.22, 0.56)$ fractions in each classes respectively. The ResNet has 48 blocks. The transported data samples (upper panel) and distribution (lower panel) of the three-class data are illustrated along the trained invertible flow network.

Given $n_{\rm tr}$ many data-label pairs $\{X_i, Y_i\}$, we define $\mathcal{L}_g = \frac{1}{n_{\rm tr}} \sum_{i=1}^{n_{\rm tr}} \ell_g(X_i, Y_i)$ where

$$-\ell_g(X_i, Y_i) := \log p_{X|Y_i}(X_i) = \log p_{H|Y_i}(F_\theta(X_i)) + \log|\det J_{F_\theta}(X_i)|. \tag{2}$$

In (2), we parametrize $p_{H|Y}$ by a Gaussian mixture model in $\mathbb{R}^d$ with a prefixed parameter σ as

$$H|Y = k \sim \mathcal{N}(\mu_k, \sigma^2 I_d), \quad k = 1, \cdots, K. \tag{3}$$

In practice, we initialize μ_k to be sufficiently separated to ensure non-overlapping supports of Gaussian components, and we can preserve the separation during training via a barrier penalty on the distances $\|\mu_k - \mu_{k'}\|$. Figure 2 provides an example of the trained flow in $\mathbb{R}^2$.

The classification loss is defined as $\mathcal{L}_c = \frac{1}{n_{\rm tr}} \sum_{i=1}^{n_{\rm tr}} \ell_c(F_\theta(X_i), Y_i)$, where $\ell_c(H_i, Y_i)$ is the per-sample K-class classification loss computed by softmax, upon using a linear classifier on $F_\theta(X_i)$. We find the experimental results insensitive to the choice of penalty factor μ and always use $\mu = 1$.

2. Scalable conditional generation on graph data. In this subsection, the subscript $_v$ indicates graph node v. Suppose the graph (V, E) has N nodes in V and the edge set E. We denote a graph data sample X and a graph label Y as

$$X = [X_1, \cdots, X_N]^T \in \mathbb{R}^{N\times d'},\ X_v \in \mathbb{R}^{d'}, \quad Y = [Y_1, \cdots, Y_N]^T \in \mathbb{R}^N,\ Y_v \in [K], \quad v \in [N],$$

where d' is the dimension of node feature. To specify the Gaussian mixture $H|Y$, we can view X as a vector in $\mathbb{R}^d$, $d = d'N$, and label vector Y taking K^N many possibilities. Yet, doing so requires specifying a K^N-component Gaussian mixture in $\mathbb{R}^d$ that is not scalable when N is large.

To ensure scalability of iGNN, we introduce a factorized form of $H|Y$ as elaborated below, and propose to use GNN layers in the invertible ResNet (e.g., Chebnet [13] and L3net [11]). The scalability of our iGNN approach is demonstrated on a larger graph with $N = 500$ nodes, cf. Figure 3.

Suppose we have a K-component Gaussian mixture in $\mathbb{R}^{d'}$. We specify the graph $H|Y$ as

$$p(H|Y) = \prod_{v=1}^{N} p(H_v|Y_v), \quad H_v|Y_v \sim \mathcal{N}(\mu_{Y_v}, \sigma^2 I_{d'}), \quad \forall v = 1, \cdots, N, \tag{4}$$

that is, the joint distribution of $H|Y$ consists of independent and identical K-component Gaussian mixture distribution of $H_v|Y_v$ in $\mathbb{R}^{d'}$ across the graph. As a result, on the graph sample-label pair $\{X, Y\}$, the $\log p(H|Y)$ term in the generative loss (2) can be computed as

$$\log p_{H|Y}(F_\theta(X)) = \sum_{v=1}^{N} \log p_{H_1|Y_1}((F_\theta(X))_v). \tag{5}$$

Note that the factorized form of $H|Y$ reduces the complexity of modeling $H|Y$ in $\mathbb{R}^{d'N}$ to that of modeling a K-class mixture model in $\mathbb{R}^{d'}$.

3. Invertible flow network with Wasserstein-2 regularization. We use an invertible ResNet with L layers to construct the one-to-one mapping between X and H, following the framework of [5]. Let

the l-th block residual mapping $f(x, \theta_l)$ be parametrized by θ_l. The ResNet mapping from X to H is $H = F_\theta(X) = x_L$ where

$$x_l = x_{l-1} + f(x_{l-1}, \theta_l), \quad l = 1, \cdots, L, \quad x_0 = X. \tag{6}$$

The ResNet architecture is illustrated on the right of Figure 1. The architecture of the residual block is free-form: one can use any layer type inside the residual blocks and even different layer types in different blocks.

The Wasserstein-2 (W_2) regularization used in (1) takes the form

$$\mathcal{W} = \frac{1}{n_{\text{tr}}} \sum_{i=1}^{n_{\text{tr}}} \ell_w(X_i), \quad \ell_w(X_i) = \sum_{l=1}^{L} \|x_l - x_{l-1}\|_2^2, \quad \text{where } x_0 = X_i, \tag{7}$$

where x_l is defined as in (6). In practice, we numerically verify the invertibility of the trained ResNet in experiments, cf. Table 1 and Appendix A. We also find a few tens of residual blocks to be enough, including the large graph experiment.

3 Theory

In this section, we study how the proposed Wassersetein-2 regularization guarantees the invertibility of the flow network. Theoretically, we consider continuous-time flow induced by a velocity field $v(x, t)$, that is, the continuous-time flow is represented by an initial value problem (IVP) of ODE

$$\dot{x}(t) = v(x(t), t), \quad x(0) \sim p. \tag{8}$$

The transport in $\mathbb{R}^d$ is the solution mapping from $x(0) = x$ to $x(t)$ at some time $t > 0$. We refer to our full paper for complete theoretical results and proofs.

3.1 Invertibility of trained flow network

The proposed Wassersetein-2 regularization serves to penalize the transport cost

$$\mathcal{T} := \int_0^1 \mathbb{E}_{x \sim \rho(\cdot, t)} \|v(x, t)\|^2 dt, \tag{9}$$

when the number of residual blocks L is large. In the special case of $K = 1$, we transport from the source density p (the data distribution in $\mathbb{R}^d$) to the target density q, which can be chosen as standard normal $\mathcal{N}(0, I_d)$. Our minimizing objective $\mathcal{L}_g + \gamma \mathcal{W}$ (in population form) is then equivalent to

$$\mathcal{T} + \tilde{\gamma} \text{KL}(\rho(\cdot, 1) || q) \tag{10}$$

with some positive scalar $\tilde{\gamma}$. Compared to the Benamou-Brenier formula [35, 6], the objective (10) relaxes the terminal condition that $\rho(\cdot, 1) = q$ to be the KL divergence, which does not change the solution of the optimal v. The optimal flow induced by the minimizer also gives the Wasserstein geodesic from p to the normal density q, and is smooth when p is a smooth density. More details of the $K = 1$ case are given in Section 3.2. In our problem of conditional generation, when there are $K > 1$ classes, the limiting continuous time flow differs from the Wasserstein geodesic from $\rho(\cdot, 0) = p(X|Y)$ to $\rho(\cdot, 1) = p(H|Y)$ for fixed Y, but is expected to give a shared flow in $\mathbb{R}^d$ from X to H, cf. Figure 2. Under regularity conditions, one would expect the transport-cost regularized continuous-time flow to also be regular.

As a result, the velocity field $v(x, t)$ would have a finite x-Lipschitz constant B on any bounded domain. Divide the time interval $[0, 1]$ into L time steps, $\Delta t = t_{l+1} - t_l = 1/L$. Then, the transport from $x(t_{l-1})$ to $x(t_l)$ induced by $v(x, t)$ on the interval $[t_{l-1}, t_l]$ is invertible when $B\Delta t < 1$, which holds when $L > B$. In this case, also assuming that the solved discrete-time transport map $T_l(x) := x + f(x, \theta_l)$ is close to that induced by $v(x, t)$, one can expect the invertibility of T_l in each residual block, and then the composed transport F_θ over L blocks is also invertible.

3.2 The special case of $K = 1$

For the continuous-time flow, the transport mapping by the IVP (8) is invertible as long as (8) is well-posed. The normalizing flow from p to normal q is typically not unique. We introduce two constructions here that are related to the proposed Wasserstein-2 regularization.

(i) Flow by Benamou-Brenier formula. Consider (8) on $t \in [0,1]$, the flow F maps from $x = x(0)$ to $x(1)$. Let $\rho(\cdot, t)$ be the density of $x(t)$, $\rho(\cdot, 0) = p$ and $\rho(\cdot, 1) = F_{\#}p$, the push-forward final density by F. For the flow induced by the optimal $v(x,t)$ in (10) which is equivalent to the Benamou-Brenier formula, the regularity of v and ρ follows from classical OT theory:

Proposition 3.1 ([9, 36])**.** *Suppose p is smooth on $\mathbb{R}^d$ with finite moments, then the optimal velocity field $v(x,t)$ that minimizes* (10) *is smooth, the induced IVP* (8) *is well-posed on $\mathbb{R}^d \times [0,1]$ and the flow mapping is smooth and invertible.*

(ii) Flow by Fokker-Planck equation. We use the Fokker-Planck equation of stochastic process to provide another theoretical construction of the invertible flow. Because q is standard normal, the stochastic process is an Ornstein-Uhlenbeck (OU) process in $\mathbb{R}^d$, for which the Fokker-Planck equation can be written as

$$\partial_t \rho = \nabla \cdot (\rho \nabla V + \nabla \rho), \quad V(x) = |x|^2/2, \quad \rho(x,0) = p(x), \tag{11}$$

where $\rho(x,t)$ represents the probability density of the OU process at time t. The Liouville equation of (8) is $\partial_t \rho = -\nabla \cdot (\rho v)$, where $\rho(x,t)$ represents the density of $x(t)$. Comparing to (11), we see that the density evolution can be made the same if the velocity field $v(x,t)$ is set to satisfy

$$-v(x,t) = \nabla V(x) + \nabla \log \rho(x,t) = x + \nabla \log \rho(x,t). \tag{12}$$

The smoothness of v follows from the explicit expression of $\rho(x,t)$ as the solution of (11).

Proposition 3.2. *Let $\rho(x,t)$ be the solution to* (11) *from $\rho(x,0) = p$, then the IVP* (8) *induced by velocity field $v(x,t)$ as in* (12) *is well-posed on $\mathbb{R}^d \times (0,T)$ for any $T > 0$ and the flow mapping is smooth and invertible.*

While theoretically the density $\rho(x,t)$ in (11) converges to the normal equilibrium q in infinite time, the convergence is exponentially fast [8]. Thus the transported density $F_{\#}p = \rho(\cdot, T)$ for a finite $T \sim \log(1/\epsilon)$ can be ϵ-close to q.

4 Experiment

We experimentally examine the proposed iGNN model on simulated data and real graph data (solar ramping event data and traffic flow anomaly detection). We also compare spectral and spatial GNN layers on a simulated example. The experimental setup is provided in Appendix A.1, and further details are in Appendix A.2. Code is available at `https://github.com/hamrel-cxu/Invertible-Graph-Neural-Network-iGNN`.

4.1 Simulated examples

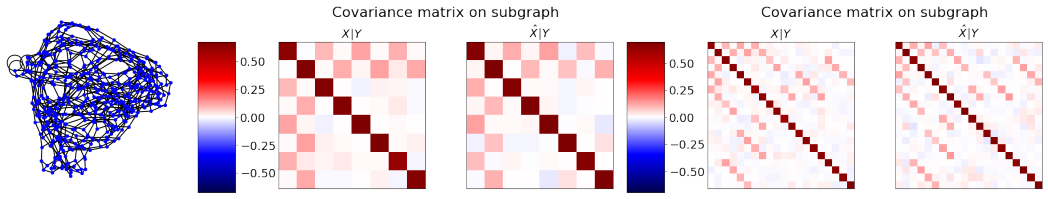
We consider three simulated examples in this section. The first considers non-graph Euclidean vector data, and the second and the third consider graph data on small and large graphs.

1. Non-graph data in $\mathbb{R}^2$. The dataset contains a Gaussian mixture of eight components with four classes, where each $X|Y$ is further divided into two disjoint Gaussian distributions in $\mathbb{R}^2$. Figure A.2 compares our generative results with cINN-MMD, where both methods can generate data that are reasonably close to $X|Y$ at each Y.

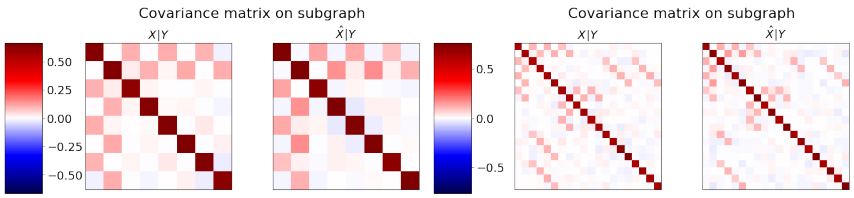
2. Data on a small graph. At each node v, $Y_v \in \{0,1\}$ and features $X_v \in \mathbb{R}^2$, thus the graph node label vector $Y \in \{0,1\}^3$. Detailed data-generating procedures can be found in Appendix A.1. In this example, iGNN yields comparable generative performance with cINN-MMD, see Figure A.4 and more results in Appendix A.2.

Table 1: Relative inversion error $\mathbb{E}_X[\|F^{-1}(F(X)) - X\|_2]$ on the solar ramping event test data. Generative quality and data details are described in Section 4.2.

γ	0	0.5	1	2	5	10
Inversion error	4.09e+04	2.74e-06	1.03e-06	3.14e-06	2.61e-06	1.60e-06



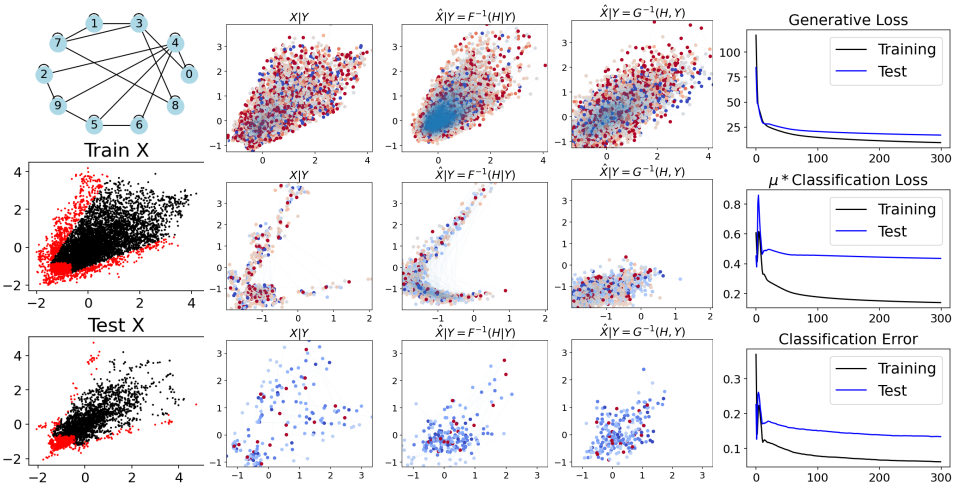
(a) 503-node graph (b) 1-hop neighborhood of node 100 (c) 2-hop neighborhood of node 100



(d) 1-hop neighborhood of node 500 (e) 2-hop neighborhood of node 500

Figure 3: Generating performance by iGNN of graph data $X|Y$ on a 503-node chordal cycle graph, where the node feature dimension $d' = 2$, and the per-node class number $K = 2$. To evaluate the conditional generation quality, we plot the covariance matrix of model-generated data $\hat{X}|Y$ (right in (b)-(e)) restricted to sub-graphs produced by 1 or 2-hop neighborhoods of a graph node in comparison with the ground truth (left in (b)-(e)).

3. Data on a large graph. We consider a 503-node chordal cycle graph [27], which is an expander graph. We design binary node labels and let node features $X|Y \sim N(\mu_Y, \Sigma_Y)$, where $X_v \in \mathbb{R}^2$ and the mean μ_Y and covariance matrix Σ_Y contain graph information. Detailed data-generating procedures can be found in Appendix A.1. Because enumerating all values of Y is infeasible, we randomly choose 50 values of outcome Y, each of which has 50% randomly selected node labels to be 1. To visualize the generative performance of iGNN, we compare the covariance of true and generated data restricted to subgraphs. Specifically, we plot the covariance matrix of model-generated data $\hat{X}|Y$ and true data $X|Y$ on sub-graphs produced by 1 or 2-hop neighborhoods of a graph node. Figure 3 shows the resemblance between learned and true covariance matrices on different neighborhoods on the graph.



(a) Raw data (b) $X|Y$ (left), iGNN $\hat{X}|Y$ (right) (c) cINN $\hat{X}|Y$ (d) iGNN loss

Figure 4: Comparison of iGNN versus cINN-MMD on solar ramping event data. The graph has ten nodes with node features in $\mathbb{R}^2$. Each row in (b)-(c) shows the model generated $\hat{X}|Y$ in comparison to the ground truth $X|Y$ for different values of Y.

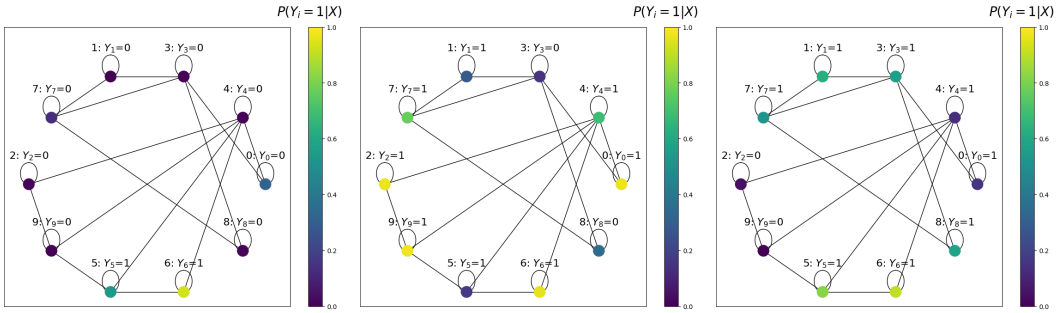


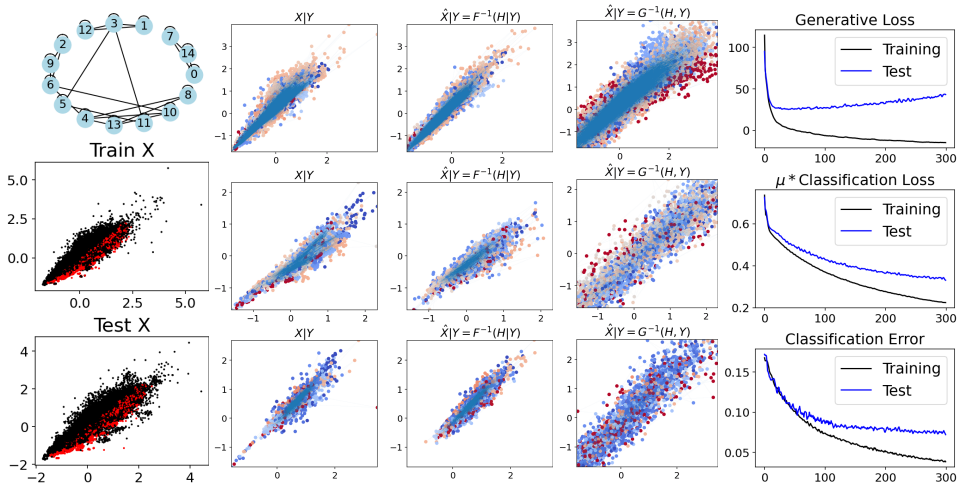
Figure 5: Predicted probabilities of graph labels Y by iGNN model where Y takes different values on graph nodes on test data. Given a node feature matrix X, we compute $\mathbb{P}(Y_i = 1|X)$ on each node using the linear classifier $f(\cdot; \theta_c^c)$ in H-Y sub-network applied to the flow-mapped graph node feature $H = F_\theta(X)$. The true node label Y_i is shown on top of each node in the plot.

4.2 Real-data examples

We apply the iGNN model to two graph prediction data in real applications. The inverse prediction problem is formulated as a conditional generation task.

1. Solar ramping events data. Consider the anomaly detection task on California solar data in 2017 and 2018, which were collected in ten downtown locations representing nodes. Each node records non-negative bi-hourly radiation recordings measured in Global Horizontal Irradiance (GHI). After pre-processing, features $X_t \in \mathbb{R}^{10\times 2}$ denote the average of raw radiation recordings every 12 hours in the past 24 hours, and response vectors $Y_t \in \{0, 1\}^{10}$ contain the anomaly statusy.

Figure 4 shows that the learned conditional distribution $\hat{X}|Y$ by iGNN model closely resembles that of the true data $X|Y$, and outperforms the generation of cINN-MMD. The quantitative evaluation is given in Table 2, which shows that iGNN has comparable or better performance than the alternative approaches (smaller test statistics indicate better generation). The table also shows that cINN-Flow performs significantly worse than cINN-MMD and iGNN on this example, which is consistent with the visual comparison of $\hat{X}|Y$ (not shown). Lastly, Figure 5 shows the predictive capability of iGNN: given a test node feature matrix X, we can compute $\mathbb{P}(Y_i = 1|X)$ for node i using the



(a) Raw data (b) $X|Y$ (left), iGNN $\hat{X}|Y$ (right) (c) cINN $\hat{X}|Y$ (d) iGNN loss

Figure 6: Comparison of iGNN versus cINN-MMD on traffic anomaly detection data. The graph has 15 nodes with node features of dimension $d' = 2$. Each row in (b)-(c) shows the model generated $\hat{X}|Y$ in comparison to the ground truth $X|Y$ for a different value of Y.

Table 2: Two-sample testing statistics under formulas (15) and (17) on test data.

Solar data	MMD	Energy	**Traffic data**	MMD	Energy
iGNN	0.062	**0.341**	iGNN	**0.128**	**0.537**
cINN-MMD	**0.061**	0.344	cINN-MMD	0.152	1.484
cINN-Flow	0.402	3.488	cINN-Flow	0.281	6.183
cGAN	0.572	3.422	cGAN	0.916	4.132

trained linear classifier on $F_\Theta(X)$. The predicted probabilities learned by the model are consistent with the true nodal labels, and provide more information than the binary prediction output.

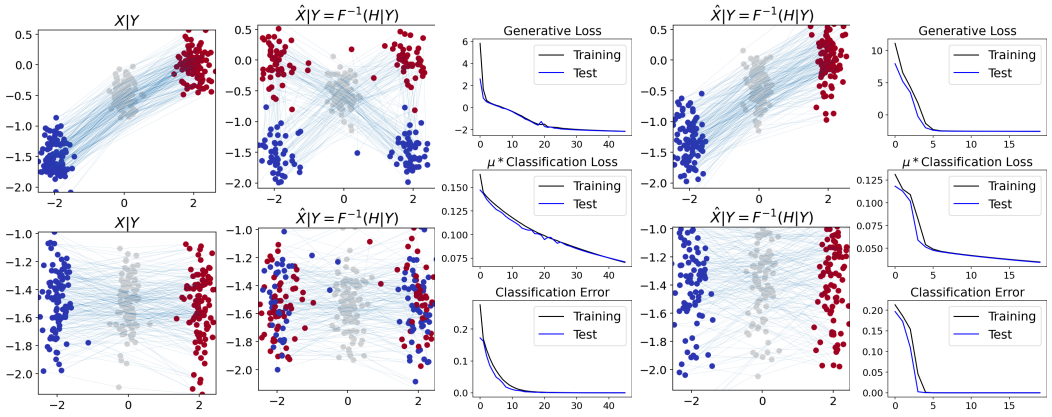
2. *Traffic flow anomalies.* We study the anomaly detection task on Los Angeles traffic flow data from April to September 2019. The whole network has 15 sensors with hourly recordings. Features $X_t \in \mathbb{R}^{15\times 2}$ denote the raw hourly recording in the past two hours, and response vectors $Y_t \in \{0,1\}^{15}$ contain the anomaly status of each traffic sensor. The graph topology is shown in Figure 6(a), along with the raw input features in $\mathbb{R}^2$ (over all graph nodes). The generative performance by iGNN resembles the ground truth, as shown in Figure 6(b), and is better than the generative performance by cINN-MMD in (c). The quantitative evaluation metrics also reveal the better performance of iGNN over alternative baselines, cf. Table 2.

4.3 Comparison of GNN layers

We examine the empirical performance of different GNN layers in learning the flow of graph data. Here we provide an example on a three-node graph where spectral graph filters lack expressiveness due to constructed symmetry.

Example 1 (Insufficient expressiveness of spectral graph filters). Consider a graph with three nodes $\{1,2,3\}$ and two edges $\{(1,2),(2,3)\}$ between nodes. Self-loops at each node are also inserted. For nodal features $X \in \mathbb{R}^3$, assume $X \sim \mathcal{N}(0,\Sigma)$. Let the covariance matrix Σ and permutation matrix π take the form

$$\Sigma := \begin{bmatrix} 1 & \rho & 0 \\ \rho & 0 & \rho_1 \\ 0 & \rho_1 & 1 \end{bmatrix}, \pi = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}.$$



(a) $X|Y$ (b) ChebNet $\hat{X}|Y$ (c) ChebNet losses (d) L3Net $\hat{X}|Y$ (e) L3Net losses

Figure 7: Comparison of using spectral and spatial GNN layer in iGNN model for conditional generation. Data in (a) are generated based on Example 1 as two-dimensional graph node features lying on a three-node graph having binary node labels. We visualize samples generated by iGNN using the spectral GNN layer (ChebNet) in (b) and spatial GNN layer (L3Net) in (d), as well as the losses over training epochs in (c) and (e).

One can see that $\pi\Sigma\pi^T \neq \Sigma$ if $\rho \neq \rho_1$. On the other hand, $\pi A\pi^T = A$ at this chosen permutation π, so that any spectral graph filter $f(A)$ as a matrix function of A satisfies that $\pi f(A)\pi^T = f(A)$. As a result, $f(A)$ for any f will make an $O(1)$ error in approximating Σ and thus fail to generate correct samples. Because (possibly normalized) graph Laplacian is either a polynomial of A or preserves the same symmetry pattern as A, the issue happens with any spectral convolutional filter.

We simulate data based on the three-node graph in Example 1. Figure 7 compares the generative performance of iGNN by using ChebNet and L3Net layers. In this example, iGNN with ChebNet layers fails to learn the conditional distribution $X|Y$. The plot (b) shows that the learned $\hat{X}|Y$ has a symmetry of nodes 1 and 3, which explains the cause of insufficiency. Meanwhile, iGNN with L3Net layers yields satisfactory performance by having sufficient model expressiveness.

5 Related works

Generative deep models and normalizing flow. At present, generative adversarial networks (GAN) [15, 19] and variational auto-encoders (VAE) [23, 24] are two of the most popular frameworks that have achieved various successes [29, 40, 26]. However, they also suffer from clear limitations such as notable difficulties in training, such as mode collapse [32] and posterior collapse [28]. On the other hand, normalizing flows (see [25] for a comprehensive review) estimate arbitrarily complex densities via the maximum likelihood estimation (MLE), and they transport original random features X into distribution that are easier to sample from (e.g., standard multivariate Gaussian) through invertible neural networks. Flow-based models can be classified into two broad classes: the discrete-time models (some of which include coupling layers [14], autoregressive layers [37] and residual networks [5, 10]), and the continuous-time models as exemplified by neural ODE [17, 31]. Most normalizing flow methods focus on unconditional generation with little development in a conditional generation. In addition, to achieve numerically reliable training, regularization of the density transport trajectories in flow networks are necessary but remain a challenge.

Conditional generation networks. The conditional generation versions of GAN (cGAN) have been studied in several places [30, 20], where the prediction outcome Y (one-hot encoded as e_Y) are typically concatenated with random noise Z and taken as input to the generator network, as illustrated in the left of Figure 1. The one-hot encoding of categorical Y concatenated with Gaussian Z poses challenges in training cGAN models, in addition to known issues of their unconditional counterparts, such as mode collapse, posterior collapse, and failures to provide exact data likelihood. When the prediction label Y is lying on a graph having N nodes, the one-hot coding will increase up to $O(N)$ more coordinates to the input (Z, e_Y), which significantly increases the model complexity and computational load with large graphs. Conditional invertible neural network (cINN) model was developed in [3] for analyzing inverse problems. The model inherits the approach of cGAN models to concatenate one-hot encoded prediction label e_Y with normal code Z while using Real-NVP layers [14] to ensure neural network invertibility. In terms of training objective, [3] proposed to use maximum mean discrepancy (MMD) losses to encourage both the matching of the input data distribution and the independence between label Y and normal code Z. We call the method in [3] cINN-MMD. Replacing the MMD losses with a flow-based objective, [4, 2] extended the invertible network approach in [3] and applied to image generation problems. In the models in [4] and [2], which we call cINN-Flow and cINN-Flow+ respectively, the inputs to the Real-NVP layers contain encoded information of the prediction label Y so as to learn label-conditioned generation.

Comparison to spectral normalization. iResNet [5] proposed spectral normalization to ensure the invertibility of each residual block. Given a weight matrix $W \in \mathbb{R}^{C\times C'}$ in a fully-connected layer, the method first computes an estimator $\tilde{\sigma}$ of the spectral norm $\|W\|_2$ by power iteration [16], and then modify the weight matrix W to be $cW/\tilde{\sigma}$ if $c/\tilde{\sigma} < 1$, where $c < 1$ is a pre-set scaling parameter. It was proposed to apply the procedure to all weight matrices in all ResNet blocks in every stochastic gradient descent (SGD) step with mini-batches. When the number of blocks L is large, this involves expensive computation, especially if the hidden layers are wide, i.e., C and C' being large. In addition, while spectral normalization of fully-connected layers together with contractive nonlinearities (e.g., ReLU, ELU, Tanh) ensures invertibility, it may not be directly applicable to other layer types, e.g., GNN layers. In contrast, the proposed Wasserstein-2 regularization are obtained from the forward passes of the residual blocks on training samples without additional computation. It is also generally compatible with free-form neural network layer types.

Acknowledgement

The work is supported by NSF DMS-2134037. C.X. and Y.X. are supported by an NSF CAREER Award CCF-1650913, NSF DMS-2134037, CMMI-2015787, DMS-1938106, and DMS-1830210. X.C. is partially supported by NSF, NIH and the Alfred P. Sloan Foundation.

References

[1] Abdullah M. Al-Shaalan. Reliability evaluation of power systems. 2019.

[2] Lynton Ardizzone, Jakob Kruse, Carsten Lüth, Niels Bracher, Carsten Rother, and Ullrich Köthe. Conditional invertible neural networks for diverse image-to-image translation. In *DAGM German Conference on Pattern Recognition*, pages 373–387. Springer, 2020.

[3] Lynton Ardizzone, Jakob Kruse, Sebastian J. Wirkert, Daniel Rahner, Eric Pellegrini, Ralf S. Klessen, Lena Maier-Hein, Carsten Rother, and U. Köthe. Analyzing inverse problems with invertible neural networks. *ArXiv*, abs/1808.04730, 2019.

[4] Lynton Ardizzone, Carsten Lüth, Jakob Kruse, Carsten Rother, and Ullrich Köthe. Guided image generation with conditional invertible neural networks. *arXiv preprint arXiv:1907.02392*, 2019.

[5] Jens Behrmann, Will Grathwohl, Ricky TQ Chen, David Duvenaud, and Jörn-Henrik Jacobsen. Invertible residual networks. In *International Conference on Machine Learning*, pages 573–582. PMLR, 2019.

[6] Jean-David Benamou and Yann Brenier. A computational fluid mechanics solution to the monge-kantorovich mass transfer problem. *Numerische Mathematik*, 84(3):375–393, 2000.

[7] Alex Beutel, Leman Akoglu, and Christos Faloutsos. Graph-based user behavior modeling: from prediction to fraud detection. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 2309–2310, 2015.

[8] François Bolley, Ivan Gentil, and Arnaud Guillin. Convergence to equilibrium in wasserstein distance for fokker–planck equations. *Journal of Functional Analysis*, 263(8):2430–2457, 2012.

[9] Luis A Caffarelli. Boundary regularity of maps with convex potentials–ii. *Annals of mathematics*, 144(3):453–496, 1996.

[10] Ricky TQ Chen, Jens Behrmann, David K Duvenaud, and Jörn-Henrik Jacobsen. Residual flows for invertible generative modeling. *Advances in Neural Information Processing Systems*, 32, 2019.

[11] Xiuyuan Cheng, Zichen Miao, and Qiang Qiu. Graph convolution with low-rank learnable local filters. In *International Conference on Learning Representations*, 2021.

[12] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[13] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016.

[14] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *ArXiv*, abs/1605.08803, 2017.

[15] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

[16] Henry Gouk, Eibe Frank, Bernhard Pfahringer, and Michael J Cree. Regularisation of neural networks by enforcing lipschitz continuity. *Machine Learning*, 110(2):393–416, 2021.

[17] Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Kristjanson Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *ArXiv*, abs/1810.01367, 2019.

[18] Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alex Smola. A kernel two-sample test. *J. Mach. Learn. Res.*, 13:723–773, 2012.

[19] Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved training of wasserstein gans. In *NIPS*, 2017.

[20] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5967–5976, 2017.

[21] Herman Kahn. Use of different monte carlo sampling techniques. 1955.

[22] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[23] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2014.

[24] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

[25] Ivan Kobyzev, Simon JD Prince, and Marcus A Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE transactions on pattern analysis and machine intelligence*, 43(11):3964–3979, 2020.

[26] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew P. Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 105–114, 2017.

[27] Alexander Lubotzky. Discrete groups, expanding graphs and invariant measures. In *Progress in mathematics*, 1994.

[28] James Lucas, G. Tucker, Roger B. Grosse, and Mohammad Norouzi. Understanding posterior collapse in generative latent variable models. In *DGS@ICLR*, 2019.

[29] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, and Ian J. Goodfellow. Adversarial autoencoders. *ArXiv*, abs/1511.05644, 2015.

[30] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *ArXiv*, abs/1411.1784, 2014.

[31] Derek Onken, S Wu Fung, Xingjian Li, and Lars Ruthotto. Ot-flow: Fast and accurate continuous normalizing flows via optimal transport. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 2021.

[32] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *ArXiv*, abs/1606.03498, 2016.

[33] Benjamín Sánchez-Lengeling and Alán Aspuru-Guzik. Inverse molecular design using machine learning: Generative models for matter engineering. *Science*, 361:360 – 365, 2018.

[34] Gábor J. Székely and Maria L. Rizzo. Energy statistics: A class of statistics based on distances. *Journal of Statistical Planning and Inference*, 143:1249–1272, 2013.

[35] Cédric Villani. *Optimal transport: old and new*, volume 338. Springer, 2009.

[36] Cédric Villani. *Topics in optimal transportation*, volume 58. American Mathematical Soc., 2021.

[37] Antoine Wehenkel and Gilles Louppe. Unconstrained monotonic neural networks. *ArXiv*, abs/1908.05164, 2019.

[38] Fang Yang, Kunjie Fan, Dandan Song, and Huakang Lin. Graph-based prediction of protein-protein interactions with attributed signed graph embedding. *BMC bioinformatics*, 21(1):1–16, 2020.

[39] Mei Yu, Zhuo Zhang, Xuewei Li, Jian Yu, Jie Gao, Zhiqiang Liu, Bo You, Xiaoshan Zheng, and Ruiguo Yu. Superposition graph neural network for offshore wind power prediction. *Future Generation Computer Systems*, 113:145–157, 2020.

[40] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2242–2251, 2017.

[41] Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Dae ki Cho, and Haifeng Chen. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *ICLR*, 2018.

A Additional experimental details

A.1 Experimental set-up

A.1.1 Computation of $\log\det$

To compute the log determinant in (2), we adopt the following unbiased log determinant approximation technique as proposed in [10]. Let $F_\theta(x) = x + f_\theta(x)$ denote the output from a generic ResNet block with parameter θ. First, observe that for any input x, $\log|\det J_{F_\theta}(x)| = \text{tr}(\log J_{F_\theta}(x))$ because the matrix $J_{F_\theta}(x)$ is non-singular. We thus have $\text{tr}(\log J_{F_\theta}(x)) = \text{tr}(\log(I + J_{f_\theta}(x)))$. As a result, the trace of the matrix logarithm can be expressed as

$$\text{tr}(\log(I + J_{f_\theta}(x))) = \text{tr}\left(\sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k}[J_{f_\theta}(x)]^k\right). \tag{13}$$

Based on (13), which takes infinite time to compute, we can obtain an unbiased estimator in finite time based on the "Russian roulette" estimator approach [21, 10]. For a ResNet as a concatenation of L ResNet blocks, the approximation is applied to each block and summed over all blocks. Lastly, to speed up gradient computation of the approximation, we further adopt memory efficient backpropagation through the early computation of gradients [10].

A.1.2 Baselines and evaluation metric.

We consider three competing conditional generative models:

- Conditional generative adversarial network (cGAN) [20].
- Conditional invertible neural network with maximum mean discrepancy (cINN-MMD) [3].
- Conditional invertible neural network using normalizing flow (cINN-Flow) [4].

To quantify performance, we measure the difference between two distributions ($X|Y$ versus $\hat{X}|Y$ at different Y) by kernel maximum mean discrepancy (MMD) [18] and energy statistics [34]. Details of the MMD and energy statistics metrics are contained in Appendix A.1.4. We also provide qualitative comparison by visualizing the generated data samples.

A.1.3 Data and ResNet architecture.

In the examples of graph data, the number of graph nodes ranges from 3 to 500. All graphs are undirected and unweighted, with inserted self-loops. Regarding the ResNet block layer type: for non-graph data, we use 2 fully-connected hidden layers of 64 neurons in all ResNet blocks. For graph data, in each residual block, we replace the first hidden layer to be a GNN layer (either ChebNet or L3Net layer); the second layer is a shared fully-connected layer that applies channel mixing across all the graph nodes (which can be viewed as a GNN layer with identity spatial convolution). The activation function is chosen as ELU [12] or LipSwish [10], which have continuous derivatives. The network is trained end-to-end with the Adam optimizer [22].

A.1.4 Model evaluation metrics

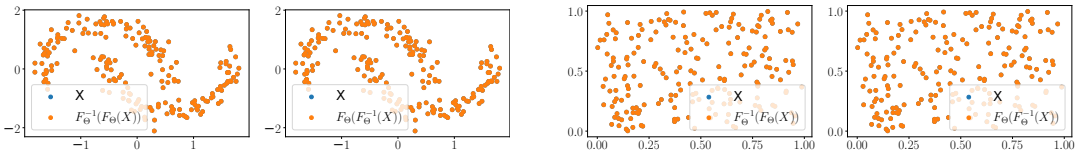
MMD statistics metric. Given two sets of samples $\boldsymbol{X} = \{x_1, \ldots, x_n\}, \boldsymbol{X'} = \{x'_1, \ldots, x'_n\}$ of same sample size n, the MMD two-sample statistic between $\boldsymbol{X}$ and $\boldsymbol{X'}$ is defined as

$$\text{MMD}(\boldsymbol{X}, \boldsymbol{X'}) := \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} k(x_i, x_j) + \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} k(x'_i, x'_j) - \frac{2}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} k(x_i, x'_j), \tag{14}$$

where we use the radial basis kernel $k(x, x') = \exp(-\alpha\|x - x'\|^2)$ with $\alpha = 0.1$.

For the K-class conditional distribution $\boldsymbol{X}|Y$, where we denote by $\{\boldsymbol{X}|Y = k\}$ the set of samples $\{x_i : y_i = k\}_{i=1}^n$, the overall MMD statistic is defined using (14) as

$$\text{MMD} = \sum_{k=1}^{K} w_k \text{MMD}(\{\boldsymbol{X}|Y = k\}, \{\boldsymbol{X'}|Y = k\}), \quad w_k = \frac{\sum_{i=1}^{n}\mathbf{1}(y_i = k)}{n}. \tag{15}$$



(a) In-distribution sample X (b) Out-of-distribution sample X

Figure A.1: The invertibility of iGNN on non-graph data in $\mathbb{R}^2$. We visualize $F_\theta^{-1}(F_\theta(X))$ (forward then invert) and $F_\theta(F_\theta^{-1}(X))$ (invert then forward) on in-distribution data (i.e., X as a part of two-moon data) and out-of-distribution data (i.e., X having random $U[0,1)$ entries).

Note that on graph data where Y concatenates all nodal labels, the summation is over all types of Y (up to K^N many).

Energy statistic metric. Given two sets of samples $\boldsymbol{X} = \{x_1, \ldots, x_n\}, \boldsymbol{X'} = \{x'_1, \ldots, x'_n\}$, The energy statistic under ℓ_2 norm is defined as

$$\text{Energy}(\boldsymbol{X}, \boldsymbol{X'}) := \frac{2}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\|x_i - x'_j\|_2 - \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\|x_i - x_j\|_2 - \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\|x'_i - x'_j\|_2, \quad (16)$$

For K-class conditional distribution $\boldsymbol{X}|Y$, the weighted energy statistics is defined using (16) as

$$\text{Energy} = \sum_{k=1}^{K} w_k \text{Energy}(\{\boldsymbol{X}|Y=k\}, \{\boldsymbol{X'}|Y=k\}), \quad w_k = \frac{\sum_{i=1}^{n}\mathbf{1}(y_i = k)}{n}. \quad (17)$$

Computation of the weighted statistics on graph data is identical to that of the weighted MMD statistics on graph.

Model invertibility error. We see from Figure A.1 that iGNN under the Wasserstein-2 regularization ensures model invertibility up to very high accuracy.

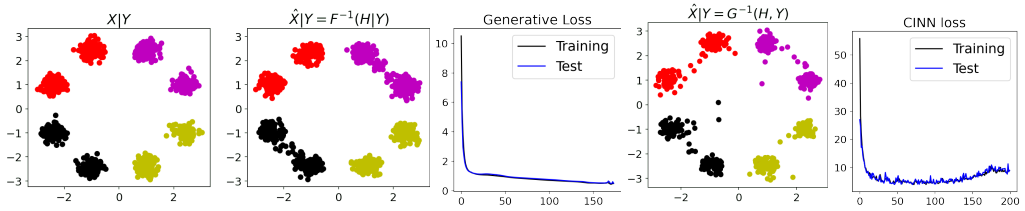
A.1.5 Construction of simulated graph data

We describe the construction of simulated graph data on small graphs (corresponding to Figures 7 and A.4). Each node has a binary label so that $Y \in \{0,1\}^3$ is a binary vector. Conditioning on a specific binary vector Y out of the eight choices, the distribution of $X|Y$ is defined as

$$X|Y := P_A\left(Z|Y + \begin{bmatrix} -4 & 0 & 4 \\ 0 & 0 & 0 \end{bmatrix}\right),$$

where $P_A := D_A^{-1}A$ is the graph averaging matrix. The distribution of $Z|Y$ is independent but non-identically distributed over nodes. On node 0 and 2, we let it be rotated and shifted noisy two moons, whereas on node 1, we let it be a Gaussian mixture in $\mathbb{R}^2$.

On large graphs (corresponding to Figure 3), each node has a binary label so that $Y \in \{0,1\}^{503}$ is a binary vector. Conditioning on a specific binary vector Y out of the eight choices, the distribution



(a) $X|Y$ (left) and iGNN $\hat{X}|Y$ (right) (b) iGNN loss (c) cINN $\hat{X}|Y$ (d) cINN loss

Figure A.2: Compare iGNN vs. cINN-MMD on simulated data in $\mathbb{R}^2$, the data observes a mixture model having eight components but are attributed to four classes (indicated by color).

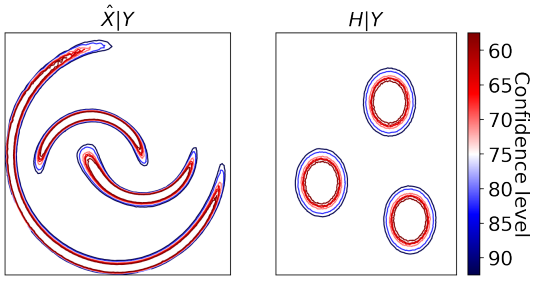


Figure A.3: Confidence region of three moons by conditional-INN. The setup is identical to that in Figure 2, where we visualize the confidence region of $X|Y$ based on that of $H|Y$. The confidence region in the input space of X can be computed from that in the feature space H based on the parametric mixture model of H.

of $X|Y$ is defined as

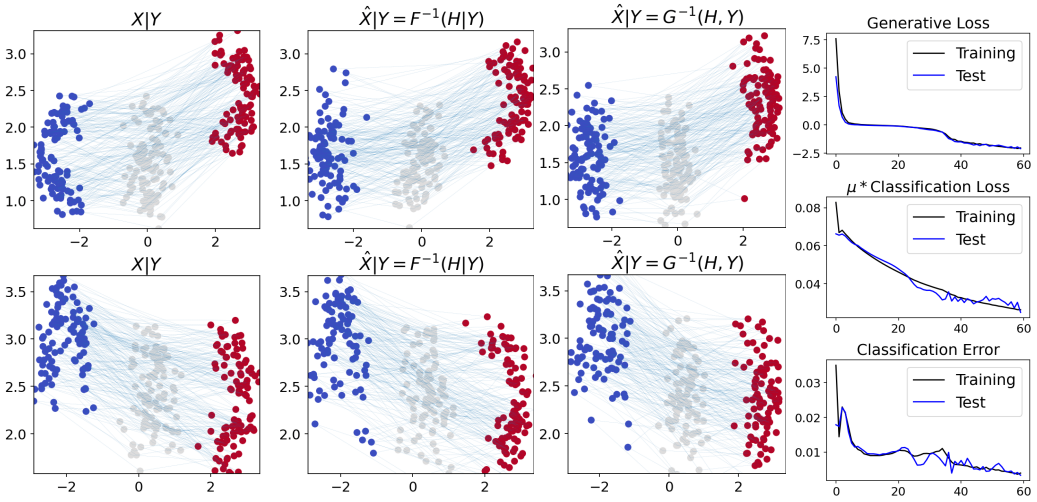
$$X|Y := R((1-\delta)I + \delta P_A)Z|Y \text{ s.t. } Z_v|Y_v \sim \begin{cases} \mathcal{N}((0,12)^T, I_2) & \text{if } Y_v = 0 \\ \mathcal{N}((0,0)^T, I_2) & \text{if } Y_v = 1 \end{cases},$$

where R denotes a counter-clockwise rotation matrix for 90 degrees and $P_A := D_A^{-1}A$ is the graph averaging matrix. We choose $\delta = 0.2$ so that a soft graph averaging is applied to the hidden variables $Z|Y$.

A.2 Additional experimental results

This subsection contains experimental results to augment those in the main text. In particular,

- We show conditional generation performance by iGNN and cINN-MMD on non-graph data in Figure A.2.
- We show the confidence region on the three-moon non-graph conditional generation data in Figure A.3. The generative quality is presented in Figure 2 in the main text.
- We compare the generative quality of iGNN and cINN-MMD on simulated three-node graph conditional generation data in Figure A.4.



(a) $X|Y$ iGNN $\hat{X}|Y$ (b) cINN-MMD $\hat{X}|Y$ (c) iGNN losses

Figure A.4: Compare iGNN vs. cINN-MMD on generating two-dimensional graph node features on the three-node graph. Color indicates node index and rows are determined by two different values of $Y \in \{0,1\}^3$. We connect the two-dimensional node features belonging to the same 3-by-2 feature matrix X by light blue lines to illustrate the distribution of $X|Y$.