Invertible Neural Networks for Graph Prediction

Chen Xu[1], Xiuyuan Cheng[2], and Yao Xie[1]

[1]H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology.
[2]Department of Mathematics, Duke University

Abstract

In this work, we address conditional generation through a new method called *invertible graph neural network* (iGNN). During training, we revise the typically-used loss objective in normalizing flow and consider Wasserstein-2 regularization. The framework can also address prediction and generation through a single model. Theoretically, we analyze the expressiveness of iGNN in learning the true mapping. Experimentally, we show the performance of iGNN on both simulated and real-data datasets.

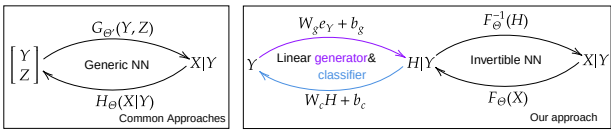


Figure 1: Abstract comparison of (several) current conditional generative models (left) and ours (right). These current approaches inevitably increase the input and output dimensions of the network, and because Y (target) and Z (randon noise) are inherently independent, practical training difficulties often arise. In comparison, our approach maintains the dimensionality of the feature space when training an invertible NN. Details are in Section 2.

1 Introduction

We are interested in the problem of finding an invertible neural network g, such that given a Y subject to $Y = g(X)$ for an input X, we can find the corresponding $X = g^{-1}(Y)$. This is an important question in many domains such as molecular design [15], anomaly detection [18], and market analysis [11], where one wants to infer the most probable features given the outcomes. This one-to-many task is synonymous with conditional generation, where the goal is to generate $X|Y$ based on specific outcomes Y. There have been several streams of works, with an abstract description presented on the left of Figure 1. Some consider conditional versions of generative adversarial networks (cGAN) [14, 9], where the outcome Y and random noise Z are both taken as inputs to the generator. More recently, the work [1] directly builds conditional invertible neural networks (CINN) for analyzing inverse problems. A single invertible NN is trained by minimizing maximum mean discrepancy (MMD) losses in the latent space Z and the original data space X. However, these works take in the outcome Y as an additional input to the networks, thus increasing the dimensionality of the original problem; we will compare our iGNN with these methods in experiments.

Therefore, although conditional generation has been a long-standing and actively researched area, there are still several open questions. In particular, we are concerned with (1) how to design (graph) generative models that do not increase difficulties in training; (2) how to reformulate the training objective in a natural way that also allows for prediction. Specifically, we extend techniques in normalizing flow [3, 17, 10] with the following contributions:

1. For the training objective, our conditional generative loss (see Eq. (1)) allows simple prediction of Y by design and extends to graph observations, without increasing the data dimension.
2. In network design, we impose the Wasserstein-2 regularization. In contrast to the much more expensive spectral normalization [3], this penalty is easy-to-implement and facilitates smoother density transport.
3. Theoretically, we show that graph neural networks with enough (resp. insufficient) expressiveness can (resp. cannot) learn the true invertible mapping.
4. In experiments, we show improved performance of iGNN over competing methods on simulated and real data. We also verify the model expressiveness results.

2 Method

Our method has three components. The first is end-to-end training for conditional generation and prediction, all within a single training objective. The second is the extension into graph data. The last is an effective regularization.

1. End-to-end training. We first consider the case where the random feature $X \in \mathbb{R}^d$. Given an invertible network F_Θ and additional categorical outcome variables $Y \in [K]$ for K classes, we consider the following conditional generative loss:

$$L_g := \log p_{H|Y}(F_\Theta(X)) + \log|\det J_{F_\Theta}(X)|, \quad (1)$$

where is the change-of-variable formula of $\log p_{X|Y}(X)$. The term L_g denotes the generative loss to be minimized and each $H|Y$ is disjoint over the value of Y. Note that conditional generation based on (1) is simple—given different outcomes Y, we can sample from different distributions $H|Y$ and then perform the inverse mapping F_Θ^{-1}. In implementation, we let $H|Y$ be isotropic multivariate Gaussian vectors.

While the loss in (1) may seem simple, we highlight its benefit over the current CINN-Nflow approach [2], whose loss is also based on the change-of-variable formula:

$$L_g^{\text{CINN}} := \log p_Z(G_{\Theta'}(X,Y)|_Z) + \log|\det J_{G_{\Theta'}}(X,Y)|, \quad (2)$$

$$G_{\Theta'}(X,Y) := [G_{\Theta'}(X,Y)|_Y, G_{\Theta'}(X,Y)|_Z].$$

In (2), Z denotes a standard multivariate Gaussian random vector and G_Θ is the conditional invertible neural network. In contrast to our objective (1), the output $G_{\Theta'}(X,Y)$ in (2) contains predictions for *both* Y and Z, so that the dimension of the original feature space is increased. In addition, because Z and Y are inherently *independent*, the conditional generative model $G_{\Theta'}^{-1}$ must transport the uni-modal random vector Z into multiple $X|Y$, each of which may be multi-modal. Both procedures increase the difficulty in training.

In addition, our conditional formulation in (1) also allows one to perform classification (e.g., estimate $Y|X$) using simple classifiers. This is because by design, distributions $H|Y$ do not overlap and $H := F_\Theta(X)$ loses no information from the feature X. We thus minimize the classification loss using a simple linear classifier on $F_\Theta(X)$:

$$\mathcal{L}_c := -e_Y^T \text{softmax}(W_c F_\Theta(X) + b_c). \quad (3)$$

Therefore, our end-to-end training objective can be written as

$$\min_{\{\Theta, \{W_c, b_c\}, \{W_g, b_g\}\}} \mathbb{E}_{X,Y}[\mathcal{L}_g + W_2(F_\Theta) + \mu \mathcal{L}_c], \quad (4)$$

which is estimated using training data. The term $W_2(F_\Theta)$ will be introduced in Eq. (5) to ensure model invertibility and smoothness in density transport. The term μ is a parameter that controls the relative rate at which different parameters are learnt. Figure 2 illustrates the result on a simple simulated two-moon data. For prediction, both parts of the two-moon data are mapped separately to disjoint parts of the Gaussian mixture. For conditional generation, one simply samples from the Gaussian mixture, and the invertible map F_Θ^{-1} generates the desired $X|Y$.

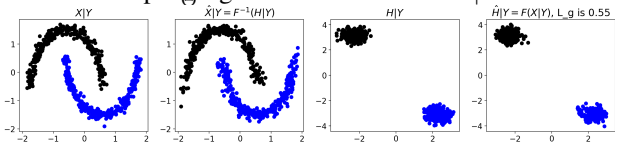


Figure 2: Illustration of iGNN on a toy example. We compare $X|Y$ vs. $\hat{X}|Y := F_\Theta^{-1}(H|Y)$ and $H|Y$ vs. $\hat{H}|Y := F_\Theta(X|Y)$, all of which are close.

2. On graph data. A key emphasis of our paper is on graph data, where we currently focus on node classification and node feature generation. Given a graph $G = (V, E)$ with $|V| = V$, let $X^{(v)} \in \mathbb{R}^C, v = 1, \ldots, V$ be the node feature, and $Y^{(v)} \in [K]$ be the node label. Denote $X \in \mathbb{R}^{V \times C}$ as the matrix of node features and $Y \in \{1, \ldots, K\}^V$ as node labels. For training, we still minimize the end-to-end objective in (4). However, we further simplify L_g in (1) as:

$$\begin{aligned} &\log p_{H|Y}(F_\Theta(X)) + \log\det J_{F_\Theta}(X) \\ \overset{(i)}{=} &\sum_{v=1}^{V} \left[\log p_{H^{(v)}|Y^{(v)}}(F_\Theta(X)^{(v)})\right] + \log\det J_{F_\Theta}(X) \\ \overset{(ii)}{=} &\sum_{v=1}^{V} \left[\log p_{H^{(1)}|Y^{(1)}}(F_\Theta(X)^{(v)})\right] + \log\det J_{F_\Theta}(X), \end{aligned}$$

where in (i), we assume independence of transported node features, and in (ii), we further assume *homogenous* conditional distribution $H^{(v)}|Y^{(v)}$ per node. In practice, requiring (ii) is important as the number of nodes is often significantly larger than the number of node classes. Thus, it is infeasible to specify node-dependent disjoint conditional distributions $H^{(v)}|Y^{(v)}$.

3. Wasserstein-2 regularization. We remark that the invertible NN F_Θ used in iGNN (see right of Figure 1) is chosen as an invertible residual network [3], which is a concatenation of residual blocks:

$$\begin{aligned} F_{b,\Theta_b}(x) &:= x + g_{b,\Theta_b}(x), \\ g_{b,\Theta_b} &:= W_{b_l} \circ \phi_{b_l} \circ \ldots \phi_{b_1} \circ W_{b_1}, \end{aligned}$$

To ensure network invertibility, we consider the following regularization on the movement of each residual block:

$$W_2(F_\Theta) := \frac{1}{2} \sum_{b=1}^{B} \|g_{b,\Theta_b}(X)\|_2^2. \quad (5)$$

Note that (5) is the Wasserstein-2 regularization [4].

There are several benefits of 5 compared to the spectral normalization technique [3]. First, implementation is easy via a single forward pass. In contrast, spectral normalization relies on the power iteration of *all* weight matrices at *each training epoch*. Doing so is highly expensive for complex residual networks. In addition, the Wasserstein-2 penalty is agnostic of the residual block design—each block can consist of only fully-connected (FC) layers or generic graph filters [6, 5]. In contrast, spectral normalization is applicable when each residual block concatenates fully-connected layers (FC) with contractive nonlinearities (e.g., ReLU, ELU, Tanh). It can be insufficient if each block consists of more complex structures.

3 Theory

We utilize the Fokker-Planck equation of an Ornstein-Uhlenbeck (OU) process to establish the expressiveness of our iGNN for Gaussian processes in two examples that depend on graph topology. Throughout this section, we assume that $X_t \in \mathbb{R}^d$, where graph features $X_t \in \mathbb{R}^{V\times C}$ can be equivalently interpreted as random vectors in $\mathbb{R}^{V\cdot C}$.

We reduce the problem to the construction of invertible flow for the unconditional case. The conditional case can be viewed as having a *joint* flow which coincides with each individual flow upon restricting to the support of $p_{X|Y}$. In particular, let the desired flow transport data distribution p_X in $\mathbb{R}^d$ to a normal distribution p_H which is $\mathcal{N}(0, I_d)$. The continuous-time flow is represented by an initial value problem (IVP) of ODE

$$\frac{dx_t}{dt} = \mathbf{f}(x_t, t), \tag{6}$$

with initial value $x_0 \sim p_X$. The solution mapping $x_t := P_t(x_0)$ for $t > 0$ is invertible as long as (6) is well-posed. There may be more than one $\mathbf{f}(x, t)$ which induces such a flow towards p_H, and to show existence, we construct $\mathbf{f}$ to correspond to the following Fokker-Planck equation of an OU process (∇ denotes ∇_x)

$$\partial_t \rho = \nabla \cdot (\rho \nabla V + \nabla \rho), V(x) = \frac{|x|^2}{2}, \tag{7}$$

where $\rho(x, t)$ represents the probability density of the OU process at time t starting from $\rho(x, 0) = p_X(x)$. Comparing (7) to the Liouville equation of (6), we see that the density transportation under the flow can be made the same as in (7) if we set the force $\mathbf{f}$ as

$$-\mathbf{f}(x, t) = x + \nabla \log \rho(x, t), \tag{8}$$

where $x = \nabla V(x)$. Using (8), we have the following analytic form of $\mathbf{f}(x, t)$ in (8), which applies to general vector data in $\mathbb{R}^d$:

Lemma 3.1. *If $X_0 \sim N(0, \Sigma)$ and $X_\infty \sim N(0, I_d)$, then*

$$\mathbf{f}(x, t) = -(I_d - \Sigma_t^{-1})x, \tag{9}$$

where $\Sigma_t := (1 - \exp(-2t))I_d + \exp(-2t)\Sigma$ is the covariance matrix that converges to I_d exponentially fast.

Proof. We only need to derive the distribution of X_t in order to find its density function $\rho(x, t)$. Because the Ornstein–Uhlenbeck process is a Gaussian process, it is evident that $X_t \sim N(0, \Sigma_t)$.

By the transition probability, also known as the Green's function

$$K(X, Y; t) := (2\pi\sigma_t^2)^{-d/2} \exp\left(-(2\sigma_t^2)^{-1}\|X - \exp(-t)Y\|_2^2\right),$$

where $\sigma_t^2 := 1 - \exp(-2t)$, we know that $X_t|X_0 \overset{d}{=} N(0, \sigma_t^2 I_d)$. When $X_0 \sim N(0, \Sigma)$, we thus have

$$X_t = \exp(-t)X_0 + \mathcal{E}, \mathcal{E} \sim N(0, \sigma_t^2 I_d), X_0 \perp \mathcal{E}.$$

Hence, $\Sigma_t = \exp(-2t)\Sigma + (1 - \exp(-2t))I_d$.

As a result, $\rho(x, t) \propto \exp\left(-1/2 X_t^T \Sigma_t^{-1} X_t\right)$, with $\nabla_X \ln \rho(x, t) = -\Sigma_t^{-1} X_t$. □

Because the force $\mathbf{f}(x, t)$ in (9) is linear, it can always be approximated by residual blocks of FC layers, which are universal approximators [8, 12].

On the other hand, assume the random vector X_0 is the concatenation of node features with Σ encoding feature dependency. Then, it is not always true that *any* graph filters can approximate Σ and thus Σ_t^{-1} in (9) that depends on Σ; one example that spectral-based GNN fails to do so will be given in Example 1. In particular, we have the following two propositions on the properties of Σ_t under different assumptions of Σ. For notation consistency with previous discussion on graph data, we change the dimensionality d to V. For simplicity, all analyses onward implicitly assume node features have dimension $C = 1$.

Case I. Spectral-based presentation of Σ: We first show the closed-form of Σ_t when Σ is a polynomial expression of the graph Laplacian L. Then spectral-based GNN can in theory correctly approximate the true force in (9).

Proposition 3.2 (Spectral-based Σ)**.** *Denote $L \in \mathbb{R}^{V\times V}$ as the graph Laplacian. Suppose there exists a polynomial P such that $\Sigma = P(L)$. Then*

$$\Sigma_t^{-1} = \sum_{i=1}^{V} ((1-\exp(-2t))+\exp(-2t)P(\lambda_i(L)))^{-1} U_i U_i^T,$$

where $\lambda_i(L)$ denotes the i-th largest eigenvalue of $L = U\Lambda U^T$ under eigen-decomposition.

Proof. Note that by the spectral decomposition $L =$

$U\Lambda U^T$ and the assumption $\Sigma = P(L)$, we have

$$\begin{aligned}\Sigma &= \sum_{i=1}^{V} P(\lambda_i(L))U_i U_i^T,\\ \Sigma_t &= (1-\exp(-2t))I_V + \exp(-2t)\Sigma\\ &= U[(1-\exp(-2t))I_V + \exp(-2t)P(\Lambda)]U^T,\end{aligned}$$

whereby the expression of Σ_t^{-1} easily follows □

The implication is that under regularity conditions on P, there exists another polynomial P_t that can approximate Σ_t^{-1} up to arbitrary precision. For instance, when P denotes the Chebyshev polynomial of order k, then this approximation can be achieved by (possibly) raising the degree of polynomial above k.

Case II. Spatial-based presentation of Σ*:*

We next show that under spatial properties of Σ, Σ_t^{-1} satisfies similar properties and can thus be approximated by spatial-based GNN filters such as the L3net [5]. However, it may *not* be approximated by spectral-based GNN such as Chebnet [6] as shown in Example 1.

Proposition 3.3 (Local Σ and Σ^{-1})**.** *Suppose Σ and Σ^{-1} are v_1-local and v_2-local, where v-locality of a covariance matrix Σ means that $\Sigma_{ij} = 0$ if node j is not in the v-neighborhood of node i. Then Σ_t^{-1} can be expressed by power series of Σ or Σ^{-1}. If the power series are truncated at order k, Σ_t^{-1} can be approximated by a $\max(kv_1, kv_2)$-local covariance matrix.*

Proof. Recall that $\Sigma_t := (1-\exp(-2t))I_V + \exp(-2t)\Sigma$. We thus analyze the locality of Σ_t based on the magnitude of t.

When $t = 0$, $\Sigma_t = \Sigma$ so that $\Sigma_t = \Sigma^{-1}$.

When $t > 0$, we consider two cases:

(1) *Long-time:* Assume $\exp(-2t) <= 1/2$ and denote $c_t := \exp(-2t)/(1-\exp(-2t))$. Then,

$$\begin{aligned}\Sigma_t &= (1-\exp(-2t)[I_V + c_t\Sigma],\\ \Sigma_t^{-1} &= (1-\exp(-2t))[\sum_{p=0}^{\infty}(-1)^p(c_t\Sigma)^p], \qquad (10)\end{aligned}$$

where (11) holds by the power series expansion $(I + A)^{-1} = \sum_{p=0}^{\infty}(-1)^p A^p$ if the spectral norm $\rho(A) < 1$. We may assume that $\rho(c_t\Sigma)$ is small enough to let this hold, especially as c_t converges to zero exponentially fast. Therefore, if Σ is v_1-local, we can truncate (11) up to the first k summands, which is at most kv_1-local.

(2) *Short-time:* Assume $\exp(-2t) > 1/2$. The proof is nearly the same as in the previous case, where we have

$$\begin{aligned}\Sigma_t &= \exp(-2t)\Sigma[I_V + c_t^{-1}\Sigma^{-1}],\\ \Sigma_t^{-1} &= \exp(-2t)[\sum_{p=0}^{\infty} -c_t^{-p}\Sigma^{-(p+1)}]. \qquad (11)\end{aligned}$$

The remaining steps are identical to above ones. □

Example 1 (Σ_t^{-1} cannot be learned by spectral-based graph filters). Consider a simple graph with three nodes $\{1, 2, 3\}$ and two edges $\{(1, 2), (2, 3)\}$ between nodes. Self-loops at each node are inserted. Note that the adjacency matrix A satisfies the property $\pi A\pi^T = A$, where $\pi \in S_3$ is any permutation matrix over graph nodes. Thus, any graph filter $f[A]$ based on A satisfies $\pi f(A)\pi^T = f(A)$. However, if Σ and π take the form

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

then it is easy to see that $\pi\Sigma\pi^T \neq \Sigma$ if $\rho \neq \rho_1$. Therefore, any $f(A)$ cannot approximate Σ up to arbitrary accuracy.

Table 1: 2D non-convex node feature two-sample test statistics between $p(X|Y)$ and $p(\hat{F}^{-1}(H|Y))$. Statistics are weighted by number of observations at each Y value.

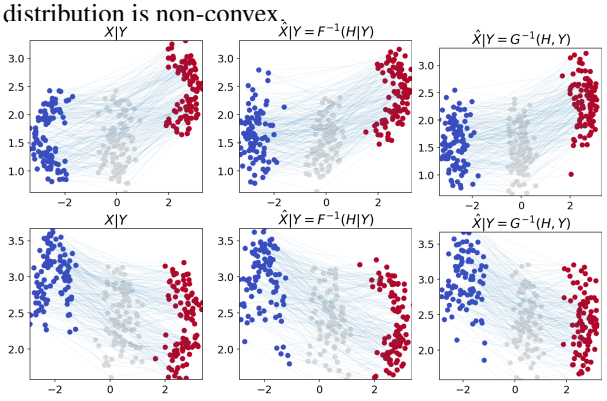
	MMD: $\alpha = 0.1$	MMD: $\alpha = 1.0$	MMD: $\alpha = 5.0$	MMD: $\alpha = 10.0$	Energy
iGNN	0.001	0.006	0.005	0.003	0.008
CINN-MMD	0.003	0.012	0.011	0.007	0.020

4 Experiments

We apply iGNN on simulated and real-data experiments and compare it with three methods: the conditional generative adversarial network (cGAN) [9], the conditional invertible neural network trained with maximum mean discrepancy (CINN-MMD) [1], and the same CINN under the normalizing flow (CINN-Nflow) [2]. Besides visual comparisons, we use two-sample statistics such as kernel MMD [7] and the energy-based test [16] to measure the generative performance. Overall, iGNN yields visibly and quantitatively better performances. In addition, we illustrate the model expressiveness result in Proposition 3.2.

Simulation on graph. We focus on node feature generation on the graph introduced in Example 1. At each node v, features $X^{(v)}$ can be either convex (e.g., linear transform of Gaussian vectors) or non-convex (e.g., a part of the two-moon dataset as in Figure 2). Each node has a binary label so that $Y \in \{0, 1\}^3$ is a three-dimensional

binary vector. For iGNN, we let each of 40 residual blocks be a concatenation of two FC layers and one L3net [5] with neighborhood order (0, 1, 2). Table 1 shows smaller two-sample testing statistics by iGNN over CINN-MMD, which performs much better than both CINN-Nflow and cGAN, whose results are thus not shown. In addition, Figure 3 shows that comparing to CINN-MMD, our method yields better visual performances when the true conditional distribution is non-convex.



(a) iGNN $\hat{X}|Y$ (b) CINN $\hat{X}|Y$

Figure 3: 2D non-convex node feature visual comparison, iGNN vs. CINN-MMD. The conditional generative performance by iGNN tends to outperform that by CINN-MMD under non-convexity in X.

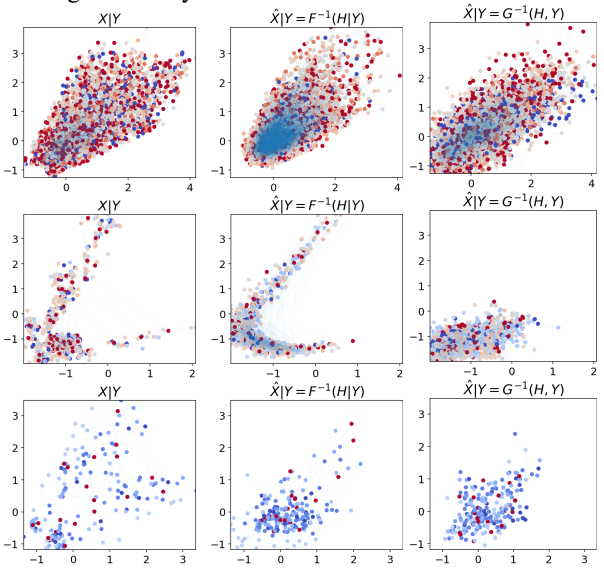
Real graph data. We then consider the anomaly detection task on California solar data in 2017 and 2018. The whole network has ten sensors with bi-hourly recordings. Features $X_t \in \mathbb{R}^{10\times 2}$ denote the average of raw radiation recordings every 12 hours in the past 24 hours, and response vectors $Y_t \in \{0,1\}^{10}$ contain the anomaly status of each city. The first 75% (resp. res 25%) data denote the training (resp. test) data, which have 1094 (resp. 365) graph observations. For iGNN, we let each of 40 residual blocks be a concatenation of two FC layers and one Chebnet of order 2. In particular, Figure 4 shows that when iGNN captures the entire distributional pattern for different $X|Y$, CINN-MMD fails to do so when the underlying distribution is non-convex (see the second row in (c) or (d)). Meanwhile, Table 2 shows that both CINN-Nflow and cGAN yield much larger test statistics than iGNN and CINN-MMD, indicating much poorer generative performance.

IGNN expressiveness. We design one example according to Proposition 3.2 to show that iGNN under graph filters that have enough expressiveness can accurately estimate the covariance matrix Σ of X *and* I_d of H, hence transporting the densities correctly. The case for Proposition 3.3 is similar and thus omitted.

Consider the case in which $\Sigma = \sum_{k=0}^{2} a_k T_k(\widetilde{L})$, where T_k denotes the k-th degree Chebyshev polynomial and $\widetilde{L}$ is the scaled and normalized graph Laplacian. We let $a_0 = 0.5, a_1 = 0.1, a_2 = 0.5$ so that Σ is positive definite with a significant number of non-zero and large off-diagonal entries. We consider a seven-node chordal cycle graph [13], which is one example of expander graphs. To verify the existence statement, we design iGNN to have 40 residual blocks, where each residual block is a single Chebnet filter of order two [6] mapping from $\mathbb{R}$ to $\mathbb{R}$. Figure 5 shows that the correlation matrices Σ and I_V are both estimated with high accuracy.

Table 2: Solar ramping event two-sample testing statistics under the same weighting scheme. CINN-Nflow and cGAN yield much larger test statistics than iGNN and CINN-MMD and are thus poorer. The quantitative results by ours and CINN-MMD can be similar, but there are clear visual differences in Figure 4

	MMD: $\alpha = 0.1$	MMD: $\alpha = 1.0$	MMD: $\alpha = 5.0$	MMD: $\alpha = 10$	Energy
iGNN	0.062	0.063	0.014	0.006	0.341
CINN-MMD	0.061	0.056	0.014	0.006	0.344
CINN-Nflow	0.402	0.091	0.015	0.006	3.488
cGAN	0.572	0.938	0.997	1.000	3.422



(a) iGNN $\hat{X}|Y$ (b) CINN $\hat{X}|Y$

Figure 4: Solar ramping event, visual comparison. We show generative performances of the 20-dimensional node feature matrices (10 nodes $\times$ 2D features), where rows are ordered by values of $Y \in \{0,1\}^{10}$ with top three occurrences. The generative performances by CINN-MMD are poor when $X|Y$ is non-convex in the second row.

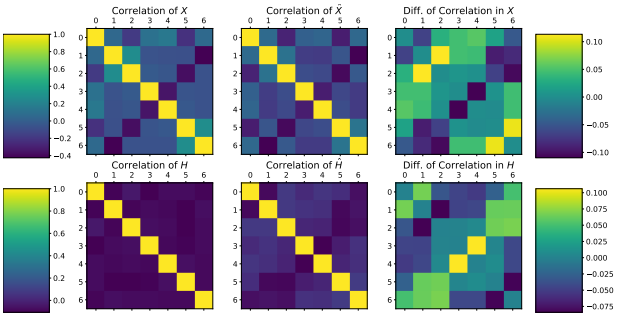


Figure 5: Results for the spectral-based Σ as in Proposition 3.2. Correlation matrices are estimated with high accuracy, revealing that iGNN with enough expressiveness can be trained to estimate the true mapping.

5 Conclusion

We proposed a general framework based on normalizing flow for conditional generation in this work. Although our primary focus is on graph data, the framework is general for any Euclidean data. In particular, our techniques address both prediction and conditional generation through a single invertible residual neural work at once, thus allowing for implementation simplicity and clarity in training. Theoretically, we examined the expressiveness of iGNN in learning the mapping. Experimentally, we compared iGNN with competing conditional generative models on simulated and real-data examples, verifying the improved performance of iGNN, especially in high-dimensional and non-convex graph cases.

References

[1] Ardizzone, L., Kruse, J., Wirkert, S. J., Rahner, D., Pellegrini, E., Klessen, R. S., Maier-Hein, L., Rother, C., and Köthe, U. Analyzing inverse problems with invertible neural networks. *ArXiv*, abs/1808.04730, 2019.

[2] Ardizzone, L., Lüth, C., Kruse, J., Rother, C., and Köthe, U. Guided image generation with conditional invertible neural networks. *arXiv preprint arXiv:1907.02392*, 2019.

[3] Behrmann, J., Grathwohl, W., Chen, R. T., Duvenaud, D., and Jacobsen, J.-H. Invertible residual networks. In *International Conference on Machine Learning*, pp. 573–582. PMLR, 2019.

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

[5] Cheng, X., Miao, Z., and Qiu, Q. Graph convolution with low-rank learnable local filters. In *International Conference on Learning Representations*, 2021.

[6] Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016.

[7] Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. A kernel two-sample test. *J. Mach. Learn. Res.*, 13:723–773, 2012.

[8] Haykin, S. Neural networks: A comprehensive foundation. 1998.

[9] Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. Image-to-image translation with conditional adversarial networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5967–5976, 2017.

[10] Kobyzev, I., Prince, S. J., and Brubaker, M. A. Normalizing flows: An introduction and review of current methods. *IEEE transactions on pattern analysis and machine intelligence*, 43(11):3964–3979, 2020.

[11] Lu, J., Wu, D., Mao, M., Wang, W., and Zhang, G. Recommender system application developments: A survey. *Decis. Support Syst.*, 74:12–32, 2015.

[12] Lu, Z., Pu, H., Wang, F., Hu, Z., and Wang, L. The expressive power of neural networks: A view from the width. In *NIPS*, 2017.

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

[14] Mirza, M. and Osindero, S. Conditional generative adversarial nets. *ArXiv*, abs/1411.1784, 2014.

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

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

[17] Ziegler, Z. M. and Rush, A. M. Latent normalizing flows for discrete sequences. In *ICML*, 2019.

[18] Zong, B., Song, Q., Min, M. R., Cheng, W., Lumezanu, C., ki Cho, D., and Chen, H. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *ICLR*, 2018.