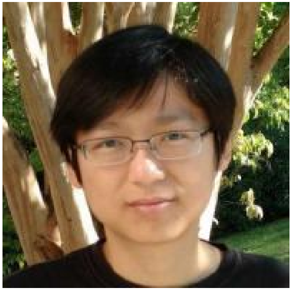
Invertible neural networks for generative models

Chen Xu
in collaboration with Prof. Xiuyuan Cheng and Prof. Yao Xie

H. Milton Stewart School of Industrial and Systems Engineering
Georgia Institute of Technology

Acknowledgement

- Xu, C., Cheng, X., and Xie, Y. (2022). Invertible neural networks for graph prediction. *IEEE Journal on Selected Areas in Information Theory.*
- Xu, C., Cheng, X., and Xie, Y. (2022). Invertible normalizing flow neural networks by JKO scheme. *ArXiv Preprint ArXiv:2212.14424.*



(a) Prof. Xiuyuan Cheng

(b) Prof. Yao Xie

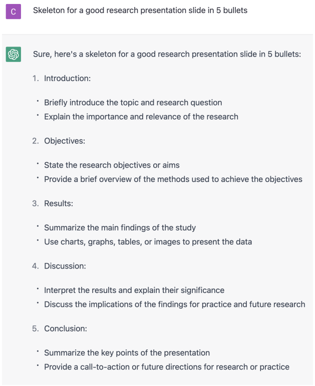
Figure 1: My wonderful and supportive collaborator Prof. Xiuyuan Cheng and advisor Prof. Yao Xie

Outline

- Introduction (motivation, objective, and related works)
- Part I: Conditional generation (method, theory, and experiments)
- Part II: Unconditional block-wise generation (method and experiments)
- Summary and extensions

Motivation

- Generative modeling is fundamental in statistics and machine learning
  - Statistics: Inferring and sample from the population $\mathbb{P}(X, Y)$.
  - Machine learning: Large-language models (e.g., ChatGPT), Image generation (e.g., DALLE-2), etc.

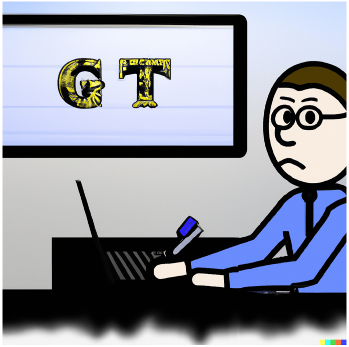


Skeleton for a good research presentation slide in 5 bullets

Sure, here's a skeleton for a good research presentation slide in 5 bullets:

1. Introduction:
   - Briefly introduce the topic and research question
   - Explain the importance and relevance of the research
2. Objectives:
   - State the research objectives or aims
   - Provide a brief overview of the methods used to achieve the objectives
3. Results:
   - Summarize the main findings of the study
   - Use charts, graphs, tables, or images to present the data
4. Discussion:
   - Interpret the results and explain their significance
   - Discuss the implications of the findings for practice and future research
5. Conclusion:
   - Summarize the key points of the presentation
   - Provide a call-to-action or future directions for research or practice

(a) Prompt: "Skeleton for a good research presentation slide in 5 bullets".



(b) Prompt: "A PhD student intently preparing for his research presentation in front of a computer showing the logo GT, cartoon style"

Problem setup

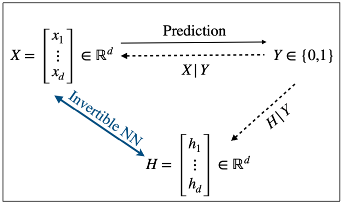

$X = \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix} \in \mathbb{R}^d$
Prediction
$X|Y$
$Y \in \{0,1\}$
Invertible NN
$H|Y$
$H = \begin{bmatrix} h_1 \\ \vdots \\ h_d \end{bmatrix} \in \mathbb{R}^d$

Figure 1: General tabular data

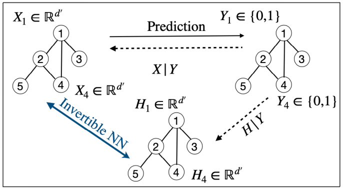
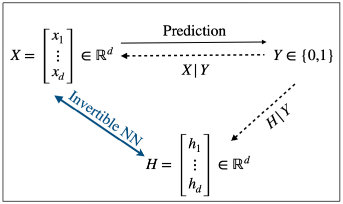

$X_1 \in \mathbb{R}^{d'}$
Prediction
$Y_1 \in \{0,1\}$
$X_4 \in \mathbb{R}^{d'}$
$X|Y$
$H_1 \in \mathbb{R}^{d'}$
$Y_4 \in \{0,1\}$
Invertible NN
$H|Y$
$H_4 \in \mathbb{R}^{d'}$

Figure 2: Graph nodal features

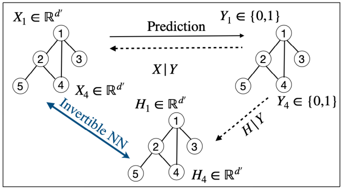
- Construct a shared flow to generate $X|Y$, where Y is categorical.
  - a shared flow $=$ one normalizing flow model (more details later)

Problem setup



$X = \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix} \in \mathbb{R}^d$
Prediction
$Y \in \{0,1\}$
$X|Y$
Invertible NN
$H|Y$
$H = \begin{bmatrix} h_1 \\ \vdots \\ h_d \end{bmatrix} \in \mathbb{R}^d$

Figure 1: General tabular data

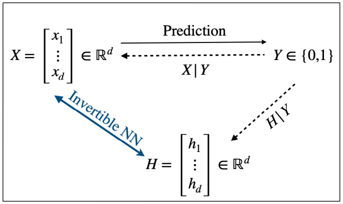


$X_1 \in \mathbb{R}^{d'}$
Prediction
$Y_1 \in \{0,1\}$
$X|Y$
$X_4 \in \mathbb{R}^{d'}$
$H_1 \in \mathbb{R}^{d'}$
$Y_4 \in \{0,1\}$
Invertible NN
$H|Y$
$H_4 \in \mathbb{R}^{d'}$

Figure 2: Graph nodal features

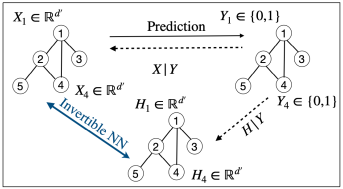
- Construct a shared flow to generate $X|Y$, where Y is categorical.
  - a shared flow = one normalizing flow model (more details later)
- Part I: Using existing normalizing flow frameworks, how to effectively incorporate Y into the model to allow conditional generation.
  - Achieves prediction and generation at once.
  - Scalable to (large) graph data (Fig 2)

Problem setup



$X = \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix} \in \mathbb{R}^d$
Prediction
$Y \in \{0,1\}$
$X|Y$
Invertible NN
$H|Y$
$H = \begin{bmatrix} h_1 \\ \vdots \\ h_d \end{bmatrix} \in \mathbb{R}^d$

Figure 1: General tabular data



$X_1 \in \mathbb{R}^{d'}$
Prediction
$Y_1 \in \{0,1\}$
$X|Y$
$X_4 \in \mathbb{R}^{d'}$
$H_1 \in \mathbb{R}^{d'}$
$Y_4 \in \{0,1\}$
Invertible NN
$H|Y$
$H_4 \in \mathbb{R}^{d'}$
1
2
3
4
5

Figure 2: Graph nodal features

- Construct a shared flow to generate $X|Y$, where Y is categorical.
  - a shared flow $=$ one normalizing flow model (more details later)
- Part I: Using existing normalizing flow frameworks, how to effectively incorporate Y into the model to allow conditional generation.
  - Achieves prediction and generation at once.
  - Scalable to (large) graph data (Fig 2)
- Application: inverse problem, power system, etc...

Problem setup (cont.)

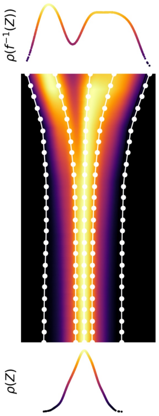


Figure 1:
$X|Y \leftrightarrow Z = H|Y$
with $Y \in \{0\}$.

Part II: Improve training of unconditional normalizing flow models (i.e., single-class Y):

- Be a part of the conditional generation pipeline
- More computational and memory efficient than existing methods

Goal

1. For **conditional generation**, how to effectively incorporate the categorical conditioning variable Y into the framework.
   - Here, we adopt existing unconditional normalizing flow model.
2. How to improve the design and training of **unconditional flow models**. Namely, $X|Y \longleftrightarrow H|Y$ for single-class Y.

Toy Examples

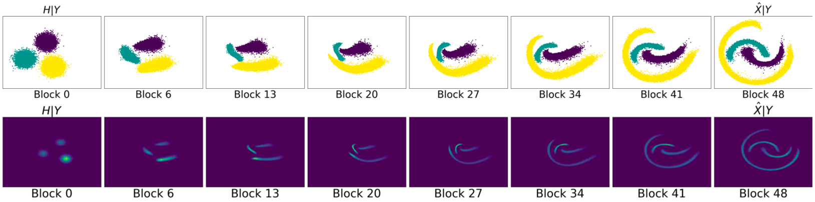


Figure 1: Part I, conditional generation: parametric mixture of $H|Y$ to $X|Y$ having three moons.

Toy Examples

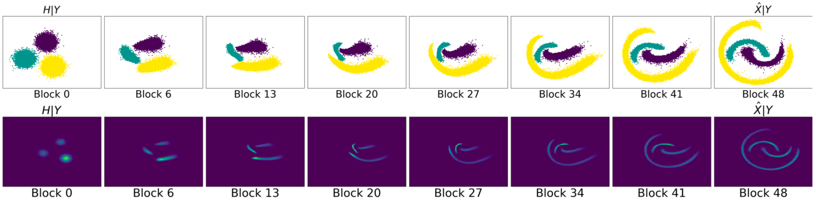
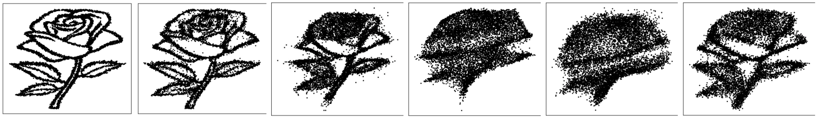


Figure 1: Part I, conditional generation: parametric mixture of $H|Y$ to $X|Y$ having three moons.



(a) True data	JKO-iFlow	(b) FFJORD	(c) OT-Flow	(d) IGNN	(e) ScoreSDE
τ**: 2.79e-4**, MMD-c:	2.73e-4	3.88e-4	1.42e-3	3.14e–3	6.90e-4
NLL	2.64	2.95	3.30	3.35	3.2

Figure 2: Part II, better flow model: proposed JKO-iFlow model.

Related works

- Traditional generative models: Hidden markov models [Baum and Petrie 1966, Rabiner and Juang 1986, Mor et al., 2021], Bayesian network [Heckerman 1996, Koller and Firedman 2009], ...
  Challenges: model assumption and specification, performance in high dimension.

Related works

- Traditional generative models: Hidden markov models [Baum and Petrie 1966, Rabiner and Juang 1986, Mor et al., 2021], Bayesian network [Heckerman 1996, Koller and Firedman 2009], ...
  Challenges: model assumption and specification, performance in high dimension.

- Deep Generative models using neural networks (NN):
  GAN [Goodfellow et al., 2014, Mirza and Osindero 2015, Gulrajani et al., 2017, ...], VAE [Kingma and Welling 2014, 2019, ...]
  Challenges: mode collapse and vanishing gradients [Salimans et al., 2016], posterior collapse [Lucas et al., 2019], and so on. Neither GAN nor VAE provides explicit data density.

Related works (cont.)

- NN-based conditional generation: conditional GAN [Mirza and Osindero 2014, Isola et al., 2017], conditional invertible neural networks (cINN) [Ardizzone et al., 2019(a)(b), 2020, 2021]
  Challenges: concatenated input (Y, Z) into the generator, restricted form of invertible NN.

Related works (cont.)

- NN-based conditional generation: conditional GAN [Mirza and Osindero 2014, Isola et al., 2017], conditional invertible neural networks (cINN) [Ardizzone et al., 2019(a)(b), 2020, 2021]
  Challenges: concatenated input (Y, Z) into the generator, restricted form of invertible NN.

- Normalizing flow models: FFJORD [Grathwohl et al., 2019], graph flow [Liu et al., 2019], OT-Flow [Onken et al., 2021]...
  Challenges: computation and memory efficiency, model regularization.

Related works (cont.)

- NN-based conditional generation: conditional GAN [Mirza and Osindero 2014, Isola et al., 2017], conditional invertible neural networks (cINN) [Ardizzone et al., 2019(a)(b), 2020, 2021]
  Challenges: concatenated input (Y, Z) into the generator, restricted form of invertible NN.

- Normalizing flow models: FFJORD [Grathwohl et al., 2019], graph flow [Liu et al., 2019], OT-Flow [Onken et al., 2021]...
  Challenges: computation and memory efficiency, model regularization.

- Neural SDE-based: score-based generative models [Song and Ermon, 2019, Song et al., 2021, Boffi & Vanden-Eijnden, 2022],
  Challenges: efficient and accurate sampling of SDE trajectory, difficulty in learning the score at all $t \in [0, T]$.

Contributions

1. Conditional generation: Propose a general $X|Y$ framework for categorical Y
   - Compatible with existing flow models.
   - Scalable to high-dimensional data, such as graphs.
   - Incorporate prediction and generation at once.

Contributions

1. Conditional generation: Propose a general $X|Y$ framework for categorical Y
    - Compatible with existing flow models.
    - Scalable to high-dimensional data, such as graphs.
    - Incorporate prediction and generation at once.
2. Normalizing flow: Introduce step-wise training of each invertible residual block
    - Memory and computationally efficient.
    - Easier training and simpler design than non-invertible models
        - Examples: score-matching, variational formulation.
    - Invertibility also allows uncertainty quantification.

Background (unconditional)

- Normalizing flow: density evolution of $\rho(x,t)$, with $\rho(x,0) = p_X$ and $\lim_{t\to\infty}\rho(x,t) = p_Z \sim \mathcal{N}(0, I_d)$.

Background (unconditional)

- Normalizing flow: density evolution of $\rho(x,t)$, with $\rho(x,0) = p_X$ and $\lim_{t\to\infty} \rho(x,t) = p_Z \sim \mathcal{N}(0, I_d)$.
- Non-unique flow: we consider flow induced by ODE of $x(t) \sim \rho(x,t)$

$$\begin{aligned} dx(t)/dt &= f(x(t),t) \qquad (1) \\ \to x(t) &= x(0) + \int_0^t f(x(s),s)ds. \qquad (2) \end{aligned}$$

• Example: (multivariate) Ornstein-Uhlenbeck (OU) process with the Fokker-Planck equation.

Background (unconditional)

- Normalizing flow: density evolution of $\rho(x,t)$, with $\rho(x,0) = p_X$ and $\lim_{t\to\infty} \rho(x,t) = p_Z \sim \mathcal{N}(0, I_d)$.

- Non-unique flow: we consider flow induced by ODE of $x(t) \sim \rho(x,t)$

$$dx(t)/dt = f(x(t),t) \tag{1}$$

$$\to x(t) = x(0) + \int_0^t f(x(s),s)ds. \tag{2}$$

  • Example: (multivariate) Ornstein-Uhlenbeck (OU) process with the Fokker-Planck equation.

- Transport regularization: $\mathcal{T} = \int_0^1 \mathbb{E}_{x\sim\rho(\cdot,t)} \|f(x,t)\|^2 dt$.
  • Recovers the Wasserstein-2 optimal transport under the Benamou-Brenier formula [Villani 2009]).

Background (unconditional)

- Normalizing flow: density evolution of $\rho(x,t)$, with $\rho(x,0) = p_X$ and $\lim_{t\to\infty} \rho(x,t) = p_Z \sim \mathcal{N}(0, I_d)$.
- Non-unique flow: we consider flow induced by ODE of $x(t) \sim \rho(x,t)$

$$dx(t)/dt = f(x(t),t) \tag{1}$$

$$\to x(t) = x(0) + \int_0^t f(x(s),s)ds. \tag{2}$$

  • Example: (multivariate) Ornstein-Uhlenbeck (OU) process with the Fokker-Planck equation.

- Transport regularization: $\mathcal{T} = \int_0^1 \mathbb{E}_{x\sim\rho(\cdot,t)} \|f(x,t)\|^2 dt$.
  • Recovers the Wasserstein-2 optimal transport under the Benamou-Brenier formula [Villani 2009]).

- Normalizing flow models learn f using neural networks.

Part I: Generative modeling for $X|Y$

Reference: Xu, C., Cheng, X., and Xie, Y. (2022). Invertible neural networks for graph prediction. *IEEE Journal on Selected Areas in Information Theory.*

Method

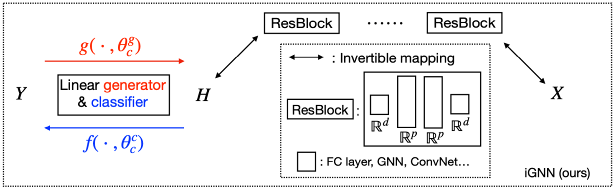


Figure 1: Illustration of framework $Y \rightleftarrows H|Y \longleftrightarrow X|Y$.

$Y \rightleftarrows H|Y$ specification:

- $H|Y = k \sim \mathcal{N}(\mu_k, \sigma^2 I_d)$.
- g part: trainable parameter $\{\mu_k\}$ with log-barrier.
- f part: cross-entropy classification loss (easy to train)

Method (cont.)

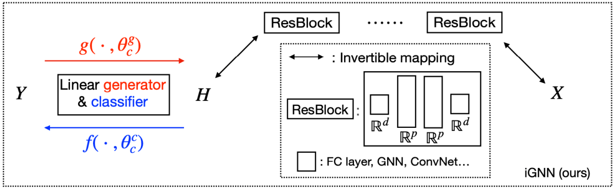


Figure 1: Illustration of framework $Y \rightleftarrows H|Y \longleftrightarrow X|Y$.

$Y \xrightarrow{g} H|Y$ on graph:

- Challenge: $|Y| = K^V$, high-dimensional $H|Y \in \mathbb{R}^{d'V}$.
  - Example: number of nodes $V = 500, K = 2, d' = 2$.

Method (cont.)

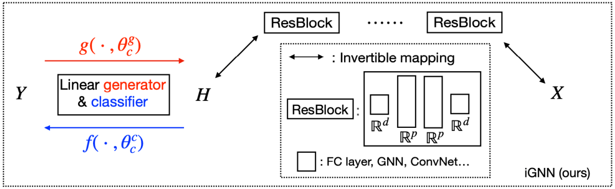


Figure 1: Illustration of framework $Y \rightleftarrows H|Y \longleftrightarrow X|Y$.

$Y \xrightarrow{g} H|Y$ on graph:

- Challenge: $|Y| = K^V$, high-dimensional $H|Y \in \mathbb{R}^{d'V}$.
  - Example: number of nodes $V = 500, K = 2, d' = 2$.
- Factorized idea:

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

  - Separation in $\mathbb{R}^{d'}$ ($\sqrt{\log(KV)}$ apart) enforces separation in $\mathbb{R}^{d'V}$.

Method (cont.)

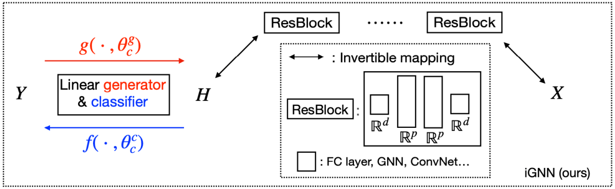


Figure 1: Illustration of framework $Y \underset{\longleftarrow}{\overset{\longrightarrow}{}} H|Y \longleftrightarrow X|Y$.

$Y \xrightarrow{g} H|Y$ on graph:

- Challenge: $|Y| = K^V$, high-dimensional $H|Y \in \mathbb{R}^{d'V}$.
  - Example: number of nodes $V = 500, K = 2, d' = 2$.
- Factorized idea:

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

  - Separation in $\mathbb{R}^{d'}$ ($\sqrt{\log(KV)}$ apart) enforces separation in $\mathbb{R}^{d'V}$.
- Computation: use GNN layers in $X|Y \leftrightarrow H|Y$.

Method (cont.)

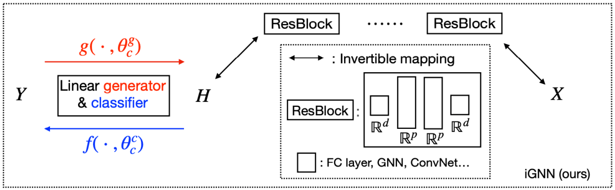


Figure 1: Illustration of framework $Y \rightleftarrows H|Y \longleftrightarrow X|Y$.

$H|Y \longleftrightarrow X|Y$ formulation:

- Residual network/block $x_{l+1} = F(x_l; \theta_l) = x_l + f(x_l; \theta_l)$
  - Euler approximation of ODE $dx(t)/dt = f(x(t), t)$

Method (cont.)

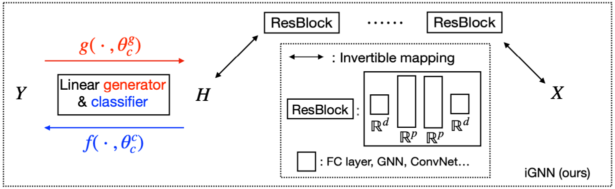


Figure 1: Illustration of framework $Y \underset{\longleftarrow}{\overset{\longrightarrow}{}} H|Y \longleftrightarrow X|Y$.

$H|Y \longleftrightarrow X|Y$ formulation:

- Residual network/block $x_{l+1} = F(x_l; \theta_l) = x_l + f(x_l; \theta_l)$
  - Euler approximation of ODE $dx(t)/dt = f(x(t), t)$
- Conditional log-likelihood via change-of-variable:

$$\log p_{X|Y}(X) = \log p_{H|Y} F(X;\theta) + \log |\text{det} J_F(X)|$$

  - $\log \det$ evaluation [Chen et al., 2020].
  - Generation $X|Y$ from $H|Y$ via fixed-point iteration [Behrmann et al., 2019]

Method (cont.)

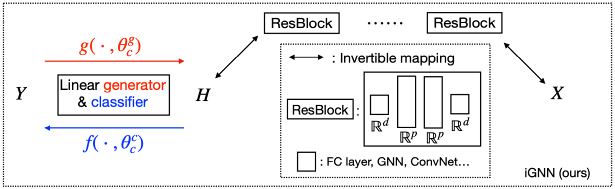


Figure 1: Illustration of framework $Y \rightleftarrows H|Y \longleftrightarrow X|Y$.

$H|Y \longleftrightarrow X|Y$ formulation:

- Transport cost/W_2 regularization: $\mathcal{W} = \sum_{l=1}^{L} \|f(X;\theta_l)\|^2$
  - • In theory, ensures invertibility via $L \to \infty$ and the smoothness of the transport-cost regularized continuous-time flow.
  - • In practice, achieves invertibility even for small L (i.e., $L = 5$).

Method (cont.)

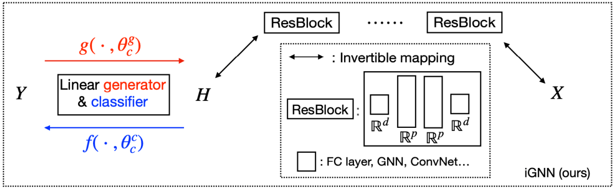


Figure 1: Illustration of framework $Y \rightleftarrows H|Y \longleftrightarrow X|Y$.

$H|Y \longleftrightarrow X|Y$ formulation:

- Transport cost/W_2 regularization: $\mathcal{W} = \sum_{l=1}^{L} \|f(X;\theta_l)\|^2$
  • In theory, ensures invertibility via $L \to \infty$ and the smoothness of the transport-cost regularized continuous-time flow.
  • In practice, achieves invertibility even for small L (i.e., $L = 5$).
  • Benefits over spectral normalization (smooth trajectory and computation).

Method (cont.)

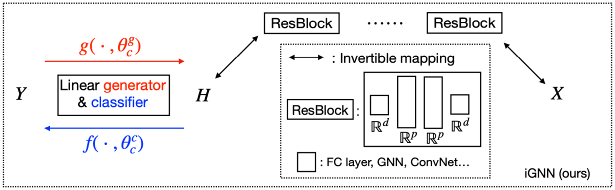


Figure 1: Illustration of framework $Y \rightleftarrows H|Y \longleftrightarrow X|Y$.

Final objective (averaged over n pairs of (X_i, Y_i)):

$$\min_{\theta,\theta_c} \mathcal{L}_g + \mathcal{L}_c + \gamma\mathcal{W}.$$

- $\mathcal{L}_g$ denotes the negative log-likelihood of (X_i, Y_i).
- $\mathcal{L}_c$ denotes cross-entropy loss in estimating $Y_i|X_i$.
- $\mathcal{W}$ denotes the W2 regularization of blocks $f_l, l = 1 \ldots, L$, under penalty $\gamma > 0$.

Theoretical analyses (Flow on graph)

- Consider $x(0) \sim \mathcal{N}(0, \Sigma), \Sigma \in \mathbb{R}^{V \times V}$ is PSD and invertible.

Theoretical analyses (Flow on graph)

- Consider $x(0) \sim \mathcal{N}(0, \Sigma), \Sigma \in \mathbb{R}^{V \times V}$ is PSD and invertible.
- We can show $f(x, t) = T_t x$, with T_t depends on Σ, Σ^{-1}.

Theoretical analyses (Flow on graph)

- Consider $x(0) \sim \mathcal{N}(0, \Sigma), \Sigma \in \mathbb{R}^{V \times V}$ is PSD and invertible.
- We can show $f(x,t) = T_t x$, with T_t depends on Σ, Σ^{-1}.
- Under assumptions on Σ, Σ^{-1}, we have:
  (Spectral)[1] $\|T_t - p_t(L)\|_2 \in \mathcal{O}(\exp\{-n\})$.
  (Spatial)[2] $\|T_t - \sum_{k=0}^{n+1} c_k(t) B_k\|_2 \in \mathcal{O}(\exp\{-n\})$.

[1] $p_t(L)$ is a polynomial with degree at most n.
[2] B_k are local filters with locality depending on that of Σ, Σ^{-1}.

Theoretical analyses (Flow on graph)

- Consider $x(0) \sim \mathcal{N}(0, \Sigma), \Sigma \in \mathbb{R}^{V \times V}$ is PSD and invertible.
- We can show $f(x,t) = T_t x$, with T_t depends on Σ, Σ^{-1}.
- Under assumptions on Σ, Σ^{-1}, we have:
  (Spectral)[1] $\|T_t - p_t(L)\|_2 \in \mathcal{O}(\exp\{-n\})$.
  (Spatial)[2] $\|T_t - \sum_{k=0}^{n+1} c_k(t) B_k\|_2 \in \mathcal{O}(\exp\{-n\})$.
- Expressiveness of GNN: there exists Σ with spatial properties and cannot be approximated by any spectral-based GNN layer (e.g., GCN, Chebnet, etc.).
  - In other words, residual blocks with GNN layers lacking expressiveness can never generate $X|Y$ on graph as desired.

[1] $p_t(L)$ is a polynomial with degree at most n.
[2] B_k are local filters with locality depending on that of Σ, Σ^{-1}.

Theoretical analyses (Flow on graph)

- Consider $x(0) \sim \mathcal{N}(0, \Sigma), \Sigma \in \mathbb{R}^{V \times V}$ is PSD and invertible.
- We can show $f(x,t) = T_t x$, with T_t depends on Σ, Σ^{-1}.
- Under assumptions on Σ, Σ^{-1}, we have:
  (Spectral)[1] $\|T_t - p_t(L)\|_2 \in \mathcal{O}(\exp\{-n\})$.
  (Spatial)[2] $\|T_t - \sum_{k=0}^{n+1} c_k(t) B_k\|_2 \in \mathcal{O}(\exp\{-n\})$.
- Expressiveness of GNN: there exists Σ with spatial properties and cannot be approximated by any spectral-based GNN layer (e.g., GCN, Chebnet, etc.).
  - In other words, residual blocks with GNN layers lacking expressiveness can never generate $X|Y$ on graph as desired.
- Theoretical details and additional results in [Xu et al., 2022]

[1] $p_t(L)$ is a polynomial with degree at most n.
[2] B_k are local filters with locality depending on that of Σ, Σ^{-1}.

Experiments–simulation

- No graph, imbalanced $X|Y$ samples

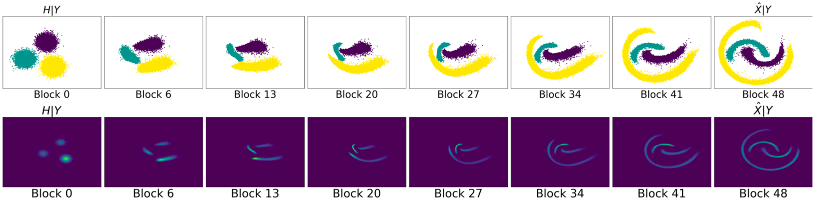


Figure 1: Three-moon dataset, $Y \in \{0, 1, 2\}$

Experiments–simulation (cont.)

- Small-graph (expressiveness of GNN layers)
- **Takeaway:** Due to symmetry in the graph design, Chebnet lacks expressiveness when L3Net correctly generates.

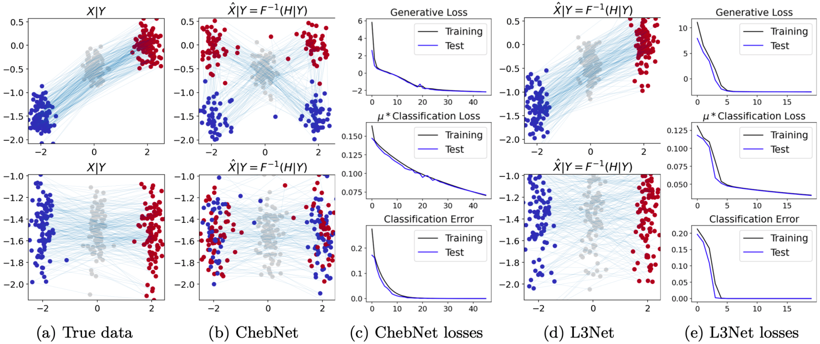
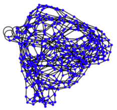
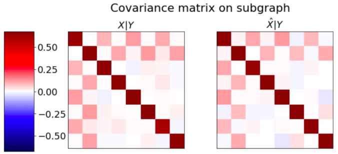


Figure 1: Spectral vs. spatial layer comparison: The graph has three nodes with binary nodal labels and 2D nodal features.

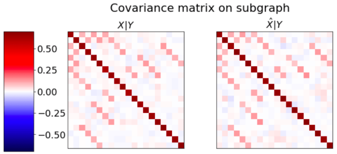
Experiments–simulation (cont.)

- Large graph with Gaussian $X|Y$
- **Takeaway:** The closeness between estimated and true covariance matrices restricted to subgraphs reflect the ability of iGNN to generate $X|Y$.

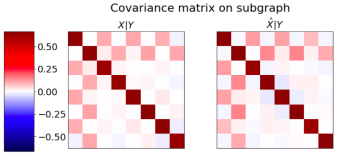

(a) Graph



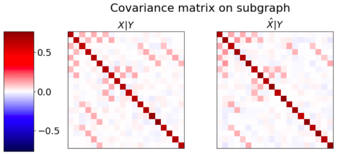
(b) 1-hop neighborhood of node 100 (4 nodes)



(c) 2-hop neighborhood of node 100 (10 nodes)



(d) 1-hop neighborhood of node 500 (4 nodes)



(e) 2-hop neighborhood of node 500 (10 nodes)

Figure 1: A chordal graph with 503 nodes with binary nodal labels and 2D nodal features. Due to high-dimensionality of $X|Y$, we visualize covariances of sub-graphs using true and generated samples.

Experiments–real data

- Baselines: compare with cGAN [Isola et al., 2017] and CINN [Ardizzone et al., 2019(a)(b).]
- These methods
  - Either concatenate Y as additional input to $Z \sim \mathcal{N}(0, I_d)$
  - Or encode Y into residual blocks with special architecture (e.g., Real-NVP).
  - Doing so increases training difficulty and yields worse performance than iGNN.

Experiments–real data

- Baselines: compare with cGAN [Isola et al., 2017] and CINN [Ardizzone et al., 2019(a)(b).]
- These methods
  - Either concatenate Y as additional input to $Z \sim \mathcal{N}(0, I_d)$
  - Or encode Y into residual blocks with special architecture (e.g., Real-NVP).
  - Doing so increases training difficulty and yields worse performance than iGNN.
- Datasets: Solar ramping event and traffic anomaly detection, with $V \approx 10$ or 15 nodes, binary nodal labels Y_v, and two-dimensional nodal feature X_v.

Experiments–real data (cont.)

- Solar ramping dataset
- **Takeaway:** The distribution of generated $\hat{X}|Y$ matches with that of the true $X|Y$ over difference Y.

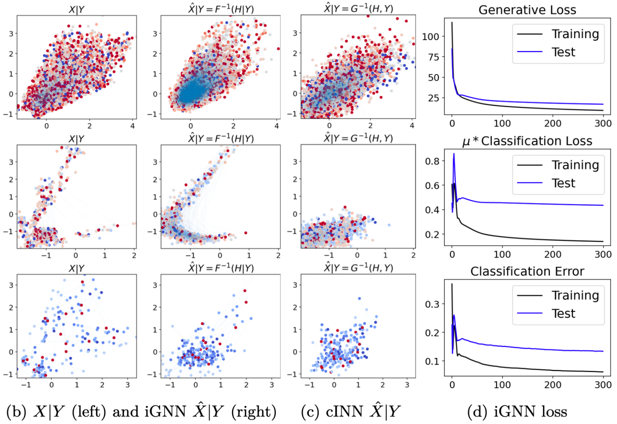


Figure 1: Scatter plot of conditionally generated nodal features. Colors indicate empirical variance of features over nodes, and we connect features of same graph by light-blue lines.

Experiments–real data (cont.)

- Traffic anomaly detection dataset

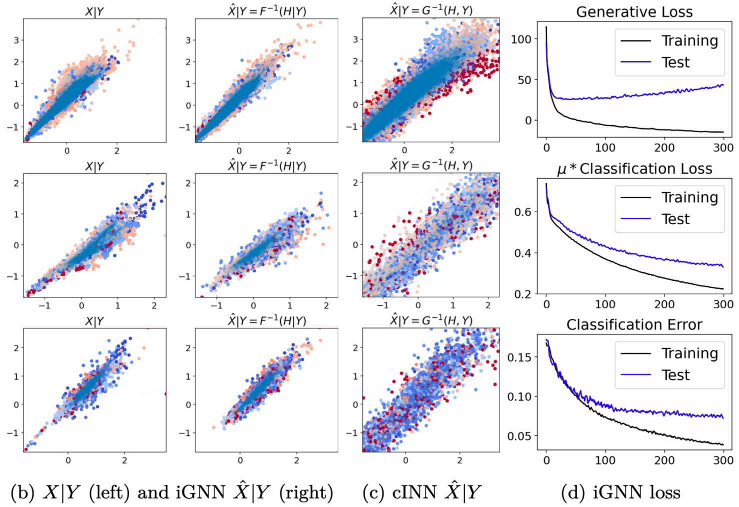


Figure 1: Scatter plot of conditionally generated nodal features. Same plot arrangement as in solar ramping event data.

Experiments–real data (cont.)

- Quantitative metrics on test data via two-sample testing methods: MMD [Gretton et al., 2012] and Energy statistics [Székely and Rizzo 2013].

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

Figure 1: Quantitative metrics of empirical performances. Smaller indicates a closer match between the empirical distributions.

Experiments–additional results

- Prediction in addition to conditional generation
  - Achieved by the $H|Y$ designs and accurate flow
- **Takeaway:** The value of $\hat{\mathbb{P}}(Y_i = 1|X)$ approximately matches the actual label of Y_i.

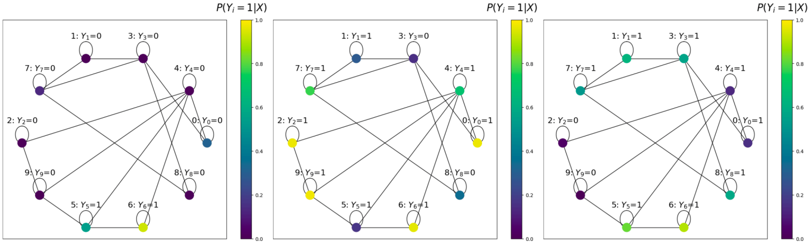


Figure 1: Predicted $\mathbb{P}(Y_v = 1|X)$ at three different values of Y containing graph nodal labels

Experiments–additional results (cont.)

- Multi-dimensional uncertainty quantification
  - Achieved through invertibility of the shared flow
- **Takeaway:** IGNN has the potential to quantify uncertainty in multi-dimensional prediction.

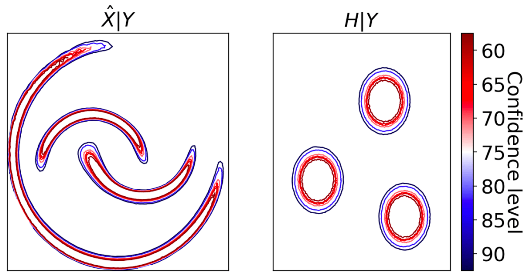


Figure 1: Uncertainty sets for three moon, based on confidence regions of $H|Y$.

Part II: Improved normalizing flow framework

Reference: Xu, C., Cheng, X., and Xie, Y. (2022). Invertible normalizing flow neural networks by JKO scheme. *ArXiv Preprint ArXiv:2212.14424.*

Motivation

- Previously, IGNN uses *discrete-time* invertible residual networks:

$$F(x_l;\theta_l) = x_l + f(x_l;\theta_l), x_0 \sim p_X.$$

Motivation

- Previously, IGNN uses *discrete-time* invertible residual networks:

$$F(x_l;\theta_l) = x_l + f(x_l;\theta_l), x_0 \sim p_X .$$

- In practice, *continuous-time* flows yield better approximation of the ODE flow $\int_0^T f(x(s),s)ds$ [Grathwohl et al., 2019, Onken et al., 2021]

Motivation

- Previously, IGNN uses *discrete-time* invertible residual networks:

$$F(x_l;\theta_l) = x_l + f(x_l;\theta_l), x_0 \sim p_X.$$

- In practice, *continuous-time* flows yield better approximation of the ODE flow $\int_0^T f(x(s),s)ds$ [Grathwohl et al., 2019, Onken et al., 2021]
- Most existing continuous flows *pre-specify* the number of blocks L to be trained [Ibid.]
  - Namely, the integral from $[0,T]$ is broken into a sequence of L sub-blocks $f(\cdot;\theta_l)$.

Motivation

- Previously, IGNN uses *discrete-time* invertible residual networks:

$$F(x_l;\theta_l) = x_l + f(x_l;\theta_l), x_0 \sim p_X.$$

- In practice, *continuous-time* flows yield better approximation of the ODE flow $\int_0^T f(x(s),s)ds$ [Grathwohl et al., 2019, Onken et al., 2021]
- Most existing continuous flows *pre-specify* the number of blocks L to be trained [Ibid.]
    - Namely, the integral from $[0,T]$ is broken into a sequence of L sub-blocks $f(\cdot;\theta_l)$.
- Yet, challenges are
    - Design: how to specify L.
    - Computation: joint training of all L blocks.
    - Memory: samples are flowed through all L blocks.

Method

- Our JKO-iFlow is inspired by the Jordan-Kinderleherer-Otto (JKO) scheme [Jordan et al., 1998]: starting at $p_0 = \rho_0 \in \mathcal{P}$, with step size $h > 0$, the JKO scheme at the $k-$th step is

$$p_{k+1} = \arg\min_{p\in\mathcal{P}} \mathsf{KL}(\rho\|\|p_Z) + \frac{1}{2h} W_2^2(p_k, \rho).$$

Method

- Our JKO-iFlow is inspired by the Jordan-Kinderleherer-Otto (JKO) scheme [Jordan et al., 1998]: starting at $p_0 = \rho_0 \in \mathcal{P}$, with step size $h > 0$, the JKO scheme at the $k-$th step is

$$p_{k+1} = \arg\min_{p \in \mathcal{P}} \mathsf{KL}(\rho \| \| p_Z) + \frac{1}{2h} W_2^2(p_k, \rho).$$

- Using the instantaneous change-of-variable formula [Chen et al., 2018], we derive the step-wise objective in JKO-iFlow as:

$$\min_{\theta_k} \mathbb{E}_{x(t_k) \sim p_k} \| x(t_{k+1}) \|^2 - \int_{t_k}^{t_{k+1}} \nabla \cdot f_{\theta_k}(x(s), s) ds + \frac{1}{2h} \| x(t_{k+1}) - x(t_k) \|^2,$$

where $x(t_{k+1}) = x(t_k) + \int_{t_k}^{t_{k+1}} f_{\theta_k}(x(s), s) ds$.

Method

- Our JKO-iFlow is inspired by the Jordan-Kinderleherer-Otto (JKO) scheme [Jordan et al., 1998]: starting at $p_0 = \rho_0 \in \mathcal{P}$, with step size $h > 0$, the JKO scheme at the $k-$th step is

$$p_{k+1} = \arg\min_{p \in \mathcal{P}} \mathsf{KL}(\rho \| \| p_Z) + \frac{1}{2h} W_2^2(p_k, \rho).$$

- Using the instantaneous change-of-variable formula [Chen et al., 2018], we derive the step-wise objective in JKO-iFlow as:

$$\min_{\theta_k} \mathbb{E}_{x(t_k) \sim p_k} \|x(t_{k+1})\|^2 - \int_{t_k}^{t_{k+1}} \nabla \cdot f_{\theta_k}(x(s), s) ds + \frac{1}{2h} \|x(t_{k+1}) - x(t_k)\|^2,$$

where $x(t_{k+1}) = x(t_k) + \int_{t_k}^{t_{k+1}} f_{\theta_k}(x(s), s) ds$.

- Benefits are thus
  - Use stopping criterion to determine number of blocks
  - No sampling (e.g., SDE-based score matching [Song et al., 2021]) nor variational learning (e.g., min-max formulation [Fan et al., 2021])

Method (cont.)

- We further proposed reparametrization-and-refine techniques to improve training. In short,
  - Reparametrization adjusts width of intervals $[t_k, t_{k+1}]$ to encourage even W_2 movement per block, in light of exponential convergence by JKO theory.

Method (cont.)

- We further proposed reparametrization-and-refine techniques to improve training. In short,
  - Reparametrization adjusts width of intervals $[t_k, t_{k+1}]$ to encourage even W_2 movement per block, in light of exponential convergence by JKO theory.
  - Refinement interpolates between $[t_k, t_{k+1}]$ to increase accuracy (i.e., double $\#$ blocks)

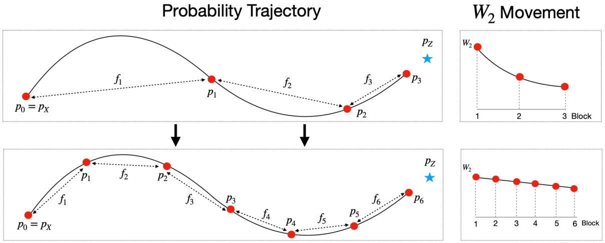
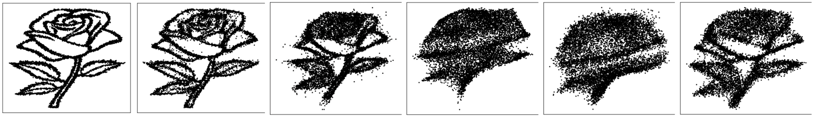


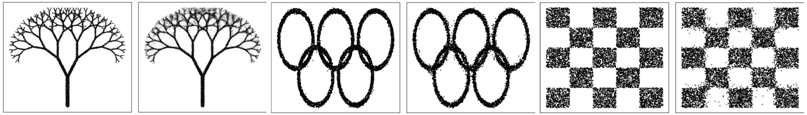
Figure 1: Before and after reparametrization and refinement.

Experiments–simulation

- Baselines: two discrete-time flow [Berhmann et al., 2019, Xu et al., 2022], two continuous-time flow [Grathwohl et al., 2019, Onken et al., 2021], and one diffusion model [Song et al., 2021].
- **Takeaway:** JKO-iFlow yields a closer match of $\hat{X}$ vs. X.



(a) True data	JKO-iFlow	(b) FFJORD	(c) OT-Flow	(d) IGNN	(e) ScoreSDE
τ: **2.79e-4**, MMD-c:	2.73e-4	3.88e-4	1.42e-3	3.14e–3	6.90e-4
NLL	2.64	2.95	3.30	3.35	3.2

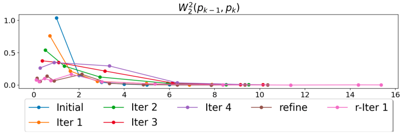


(f) Fractal tree		(g) Olympic rings		(h) Checkerboard	
τ: **3.12e-4**, MMD-c:	2.17e-4	τ: **3.16e-4**, MMD-c:	2.36e-4	τ: **3.09e-4**, MMD-c:	2.70e-4
NLL	2.20	NLL	1.66	NLL	3.59

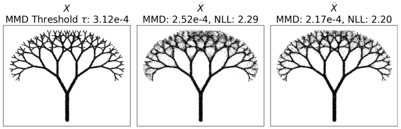
Figure 1: Two-dimensional datasets visualized as scatter plots.

Experiments–simulation (cont.)

- Benefits of reparametrization + refinement.
- **Takeaway:** improved performance on edges, at which we have few samples.



(a) Per-block W_2^2 over reparameterization iterations and refinement ('r-Iter 1' means one reparameterization iteration after refinement).

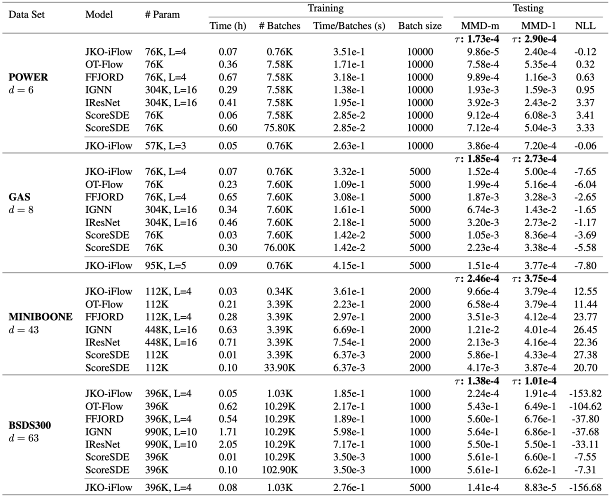


(b) Results at Iter 4 (middle) and r-Iter 1 (right). MMD and NLL values are shown in the title.

Figure 1: W_2 movement before and after reprametrization and refinement, as well as the generated samples.

Experiments–real data

- High-dimensional tabular daatsets ($d = 6, 8, 43, 63$).
- **Takeaway:** competitive or better performance under much less number of mini-batch SGD.

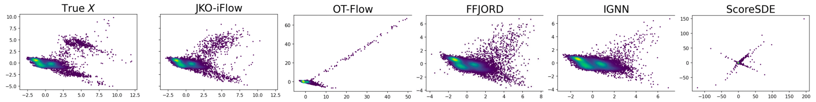


Data Set	Model	# Param	Training				Testing		
			Time (h)	# Batches	Time/Batches (s)	Batch size	MMD-m	MMD-1	NLL
							τ: 1.73e-4	**τ: 2.90e-4**	
POWER $d = 6$	JKO-iFlow	76K, L=4	0.07	0.76K	3.51e-1	10000	9.86e-5	2.40e-4	-0.12
	OT-Flow	76K	0.36	7.58K	1.71e-1	10000	7.58e-4	5.35e-4	0.32
	FFJORD	76K, L=4	0.67	7.58K	3.18e-1	10000	9.89e-4	1.16e-3	0.63
	IGNN	304K, L=16	0.29	7.58K	1.38e-1	10000	1.93e-3	1.59e-3	0.95
	IResNet	304K, L=16	0.41	7.58K	1.95e-1	10000	3.92e-3	2.43e-2	3.37
	ScoreSDE	76K	0.06	7.58K	2.85e-2	10000	9.12e-4	6.08e-3	3.41
	ScoreSDE	76K	0.60	75.80K	2.85e-2	10000	7.12e-4	5.04e-3	3.33
	JKO-iFlow	57K, L=3	0.05	0.76K	2.63e-1	10000	3.86e-4	7.20e-4	-0.06
							τ: 1.85e-4	**τ: 2.73e-4**	
GAS $d = 8$	JKO-iFlow	76K, L=4	0.07	0.76K	3.32e-1	5000	1.52e-4	5.00e-4	-7.65
	OT-Flow	76K	0.23	7.60K	1.09e-1	5000	1.99e-4	5.16e-4	-6.04
	FFJORD	76K, L=4	0.65	7.60K	3.08e-1	5000	1.87e-3	3.28e-3	-2.65
	IGNN	304K, L=16	0.34	7.60K	1.61e-1	5000	6.74e-3	1.43e-2	-1.65
	IResNet	304K, L=16	0.46	7.60K	2.18e-1	5000	3.20e-3	2.73e-2	-1.17
	ScoreSDE	76K	0.03	7.60K	1.42e-2	5000	1.05e-3	8.36e-4	-3.69
	ScoreSDE	76K	0.30	76.00K	1.42e-2	5000	2.23e-4	3.38e-4	-5.58
	JKO-iFlow	95K, L=5	0.09	0.76K	4.15e-1	5000	1.51e-4	3.77e-4	-7.80
							τ: 2.46e-4	**τ: 3.75e-4**	
MINIBOONE $d = 43$	JKO-iFlow	112K, L=4	0.03	0.34K	3.61e-1	2000	9.66e-4	3.79e-4	12.55
	OT-Flow	112K	0.21	3.39K	2.23e-1	2000	6.58e-4	3.79e-4	11.44
	FFJORD	112K, L=4	0.28	3.39K	2.97e-1	2000	3.51e-3	4.12e-4	23.77
	IGNN	448K, L=16	0.63	3.39K	6.69e-1	2000	1.21e-2	4.01e-4	26.45
	IResNet	448K, L=16	0.71	3.39K	7.54e-1	2000	2.13e-3	4.16e-4	22.36
	ScoreSDE	112K	0.01	3.39K	6.37e-3	2000	5.86e-1	4.33e-4	27.38
	ScoreSDE	112K	0.10	33.90K	6.37e-3	2000	4.17e-3	3.87e-4	20.70
							τ: 1.38e-4	**τ: 1.01e-4**	
BSDS300 $d = 63$	JKO-iFlow	396K, L=4	0.05	1.03K	1.85e-1	1000	2.24e-4	1.91e-4	-153.82
	OT-Flow	396K	0.62	10.29K	2.17e-1	1000	5.43e-1	6.49e-1	-104.62
	FFJORD	396K, L=4	0.54	10.29K	1.89e-1	1000	5.60e-1	6.76e-1	-37.80
	IGNN	990K, L=10	1.71	10.29K	5.98e-1	1000	5.64e-1	6.86e-1	-37.68
	IResNet	990K, L=10	2.05	10.29K	7.17e-1	1000	5.50e-1	5.50e-1	-33.11
	ScoreSDE	396K	0.01	10.29K	3.50e-3	1000	5.61e-1	6.60e-1	-7.55
	ScoreSDE	396K	0.10	102.90K	3.50e-3	1000	5.61e-1	6.62e-1	-7.31
	JKO-iFlow	396K, L=4	0.08	1.03K	2.76e-1	5000	1.41e-4	8.83e-5	-156.68

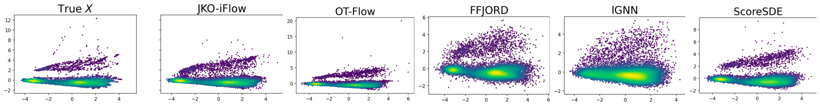
Figure 1: Quantitative metrics (MMD and NLL)

Experiments–real data (cont.)

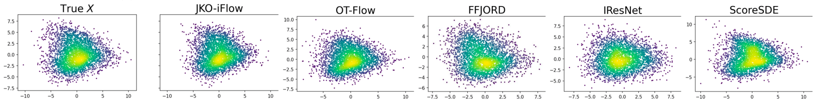
- High-dimensional tabular daatsets ($d = 6, 8, 43, 63$).
- **Takeaway:** A closer visual match between generated and true samples.



(a) Power



(b) Gas



(c) MINIBOONE

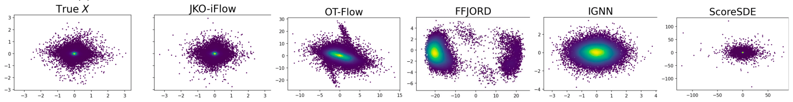


Figure 1: Four real datasets: PCA visualization

Experiments–real data (cont.)

- Image data—MNIST digits via a pre-trained auto-encoder.

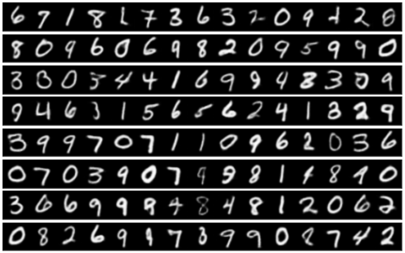
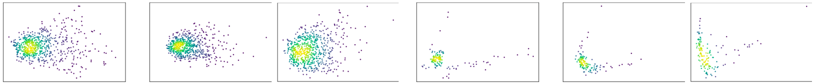


Figure 1: Uncurated MNIST digits.

Experiments–real data (cont.)

- Solar ramping event used in IGNN.
- **Takeaway:** The continuous-time model trains a more accurate invertible flow mapping between $X|Y$ and $H|Y$ than the discrete-time model.



(a) True $X\|Y$	JKO-iFlow	(b) IGNN	(c) True $X\|Y$	JKO-iFlow	(d) IGNN
τ: **1.78e-3**, MMD-m:	2.42e-2	3.25e-2	τ: **7.96e-3,** MMD-m:	5.79e-2	9.71e-2
NLL	-0.32	2.46	NLL	-8.75	-3.80

Figure 1: PCA projection and quantitative metrics.

Summary and Extensions

Part I: Generative modeling for $X|Y$

- Summary:
  - Propose a deep conditional generative model based on invertible residual networks and normalizing flow.
  - The framework is scalable to graph data with interesting implication on GNN expressiveness in generative modeling.

Summary and Extensions

Part I: Generative modeling for $X|Y$

- Summary:
  - Propose a deep conditional generative model based on invertible residual networks and normalizing flow.
  - The framework is scalable to graph data with interesting implication on GNN expressiveness in generative modeling.
- Extensions:
  - $X|Y$ generation for continuous Y (i.e., regression setting).
  - Graph topology and/or edge feature generation.

Summary and Extensions (cont.)

Part II: Improved normalizing flow framework

- Summary:
  - Propose an invertible neural ODE model that trains each residual block in a step-wise fashion.
  - Adaptive reparametrization and refinement of a computed trajectory to improve generative quality and overall computational efficiency.

Summary and Extensions (cont.)

Part II: Improved normalizing flow framework

- Summary:
  - Propose an invertible neural ODE model that trains each residual block in a step-wise fashion.
  - Adaptive reparametrization and refinement of a computed trajectory to improve generative quality and overall computational efficiency.

- Extensions:
  - Continuity in time t to further reduce computation.
  - Other larger-scale examples (e.g., image generation).
  - (Ongoing) Flow between general distributions P and Q given only training samples $X \sim X, Y \sim Y$.