

Deterministic and Stochastic Modeling using Deep Networks

Radu Balan

University of Maryland
Department of Mathematics and the Norbert Wiener Center
College Park, Maryland *rvbalan@umd.edu*

June 7, 2022

Workshop on Neural Networks and Deep Learning at the Fields Institute
June 6-10, 2022

Acknowledgments



This material is based upon work partially supported by the National Science Foundation under grant no. DMS-2108900 and Simons Foundation. “Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.”

Collaborators:

UMD: Chris Dock, Danial Ludwig, Michael Rawson, Dongmian Zou

UMB: Thomas Ernst, Bo Li, Xiaoke Wang, Ze Wang

USC: Paul Bogdan, Gaurav Gupta, Xiongye Xiao, Ruochen Yang

Verisk: Tushar Jain, Sahil Sidheekh, Maneesh Singh

Works:

ICLR 2022: Non-linear Operator Approximations for Initial Value Problems

ISMIRM 2022: Estimating Noise Propagation of Neural Network based Image Reconstruction using Automatic Differentiation

UAI 2022: VQ-Flows: Vector Quantized Local Normalizing Flow

Overview

- 1 Modeling Deterministic Evolution Operators using Deep Networks (Gupta, Xiao, R.B., Bogdan)
 - Problem Formulation: IVP
 - Architecture
 - Lipschitz Analysis
 - Experiments
- 2 Uncertainty Quantification in NN (Wang, Ludwig, Rawson, R.B., Ernst)
 - MRI and NN
 - Uncertainty Propagation through NN
 - Experimental Results
- 3 Chart based Normalizing Flows (Dock, Sidheek, Jain, R.B., Singh)
 - Global normalizing flows
 - Conformal embedding flows
 - Chart based flows - model
 - Chart based flows - performance

Table of Contents

- 1 Modeling Deterministic Evolution Operators using Deep Networks (Gupta, Xiao, R.B., Bogdan)
 - Problem Formulation: IVP
 - Architecture
 - Lipschitz Analysis
 - Experiments
- 2 Uncertainty Quantification in NN (Wang, Ludwig, Rawson, R.B., Ernst)
 - MRI and NN
 - Uncertainty Propagation through NN
 - Experimental Results
- 3 Chart based Normalizing Flows (Dock, Sidheek, Jain, R.B., Singh)
 - Global normalizing flows
 - Conformal embedding flows
 - Chart based flows - model
 - Chart based flows - performance

Problem Formulation

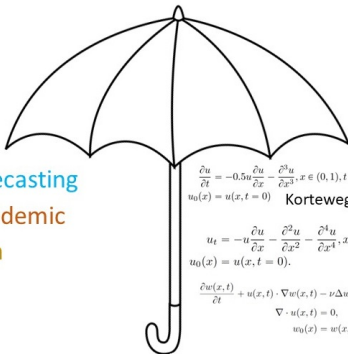
A fundamental problem in machine learning: predict future states using current conditions, $x_0 \in \mathbb{R}^s \mapsto x_T = \Phi(x_0) \in \mathbb{R}^s$.

Examples: Solutions of PDEs , Epidemic Forecasting (COVID19)

Dynamical systems, Forecasting

Kermack-McKendrik- epidemic

Hodgkin-Huxley - neuron



$$\frac{\partial u}{\partial t} = -0.5u \frac{\partial u}{\partial x} - \frac{\partial^3 u}{\partial x^3}, x \in (0, 1), t \in (0, 1] \quad \frac{\partial u}{\partial t} = -v \frac{\partial u}{\partial x} + v \frac{\partial^2 u}{\partial x^2}, x \in (0, 2\pi), t \in (0, 1]$$

$u_0(x) = u(x, t = 0)$ Korteweg-de Vries Burgers'

$$u_t = -u \frac{\partial u}{\partial x} - \frac{\partial^2 u}{\partial x^2} - \frac{\partial^4 u}{\partial x^4}, x \in (0, 1), t \in (0, 1] \quad \text{Kuramoto-Sivashinsky}$$

$u_0(x) = u(x, t = 0)$.

$$\frac{\partial w(x, t)}{\partial t} + u(x, t) \cdot \nabla w(x, t) - \nu \Delta w(x, t) = f(x), \quad x \in (0, 1)^2, t \in (0, T]$$

$\nabla \cdot u(x, t) = 0, \quad x \in (0, 1)^2, t \in [0, T]$ Navier-Stokes

$w_0(x) = w(x, t = 0), \quad x \in (0, 1)^2$

Our problem: How to implement Φ using a Deep Network and a Training data set?

Existing Approaches

- UAP based NN: Use of (Conv.) N.N. (Guo, 2016), (Grohs,2019);
- UAP with Reduced Basis/PCA: Galerkin-like schemes (Santo,2019), sparse networks (Boelcskei,Kutyniok, 2019);
- IVP defined NN: Physics-inspired neural networks (PINNs): (Raissi, Karniadakis, 2019), (Wang, PERdikaris, 2021);
- Reservoir Computing: (Schrauwen, 2007), (Girvan, Hunt, 2020);
- Neural Operators: Data-driven and input-resolution independent: Fourier Neural Op. (FNO) (Li,2020), Graph Nystrom sampling (Li, 2020), Multi Wavelet Transform (MWT) (Gupta, 2021);
- ...

Architecture

As special type of Neural Operator, is the **exponential operator** seen as the evolution operator of a linear (time-invariant) differential equation:

Equation	Solution
$\frac{d\mathbf{u}}{dt} = \mathbf{A}\mathbf{u}, u(t=0) = \mathbf{u}_0$ (Linear ODE)	$\mathbf{u}(t = \tau) = e^{t\mathbf{A}}\mathbf{u}_0$
$u_t = \frac{\partial^2 u}{\partial x^2}, u(x, 0) = u_0(x)$ (Heat equation)	$u(x, \tau) = e^{\tau \frac{\partial^2}{\partial x^2}} u_0(x)$
$u_t = \mathcal{L}u + \mathcal{N}f(u), u(x, 0) = u_0(x)$	$u(x, \tau) = e^{\tau\mathcal{L}}u_0(x) + \int_0^\tau e^{(\tau-t)\mathcal{L}}\mathcal{N}f(u(x, t))dt$

Approach: Learn operator \mathcal{L} while implementing a (nonlinear) version of $e^{\mathcal{L}}$.

Architecture

As special type of Neural Operator, is the **exponential operator** seen as the evolution operator of a linear (time-invariant) differential equation:

Equation	Solution
$\frac{d\mathbf{u}}{dt} = \mathbf{A}\mathbf{u}, u(t=0) = \mathbf{u}_0$ (Linear ODE)	$\mathbf{u}(t = \tau) = e^{t\mathbf{A}}\mathbf{u}_0$
$u_t = \frac{\partial^2 u}{\partial x^2}, u(x, 0) = u_0(x)$ (Heat equation)	$u(x, \tau) = e^{\tau \frac{\partial^2}{\partial x^2}} u_0(x)$
$u_t = \mathcal{L}u + \mathcal{N}f(u), u(x, 0) = u_0(x)$	$u(x, \tau) = e^{\tau\mathcal{L}}u_0(x) + \int_0^\tau e^{(\tau-t)\mathcal{L}}\mathcal{N}f(u(x, t))dt$

Approach: Learn operator \mathcal{L} while implementing a (nonlinear) version of $e^{\mathcal{L}}$.

Performance metrics:

- 1 Approximation error MSE (for training), MAE (for testing);
- 2 Model complexity, expressed by number of parameters to be learned; Important especially when the training data set is relatively small.

Architecture (2)

The exponential operator $\mathcal{L} \mapsto e^{\mathcal{L}}$ has been used in deep learning:

- 1 exponential function used to model NN non-linearity (Andoni, 2014);
- 2 Taylor polynomial as a truncation of the Taylor series (Hoogeboom, 2020) - particularly in the context of convolutive operators, and (Sylvester) normalizing flow;
- 3 **Our first contribution**: Use Padé approximation as a more compact polynomial form than the Taylor polynomial. **Padé Neural Operator**.

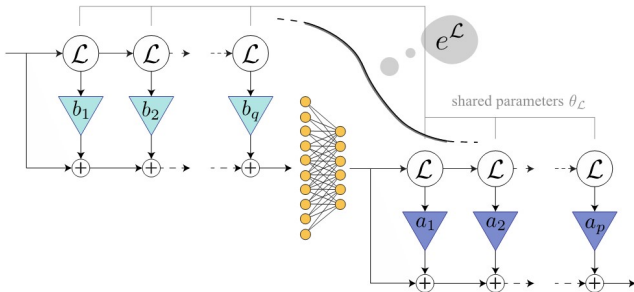
Padé Approximation of the exponential function, $x \mapsto e^x$ is denoted by

$[p/q] = \frac{A_{pq}(x)}{B_{pq}(x)}$, with $p, q \geq 0$ integers and:

$$e^x \approx [p/q] := \frac{\sum_{j=0}^p a_j x^j}{\sum_{j=0}^q b_j x^j}, \quad a_j = \frac{(p+q-j)! p!}{(p+q)! j! (p-j)!}, \quad b_j = (-1)^j \frac{(p+q-j)! q!}{(p+q)! j! (q-j)!}$$

Architecture (3)

- Padé Neural Operator $[p/q]e^{\mathcal{L}}$ with a single layer nonlinear block:



- "Our" second contribution: Decompose \mathcal{L} using a Multi-Wavelet basis (Gupta, 2021), $e^{\mathcal{L}} = \sum_{i=1}^L (Q_i e^{\mathcal{L}} Q_i + Q_i e^{\mathcal{L}} P_i + P_i e^{\mathcal{L}} Q_i) + P_L e^{\mathcal{L}} P_L$. Then apply the Padé Neural Operator for each term of this decomposition:

$$\Phi(x_0) = \sum_{i=1}^L (Q_i [p/q] e^{A_i} Q_i + Q_i [p/q] e^{B_i} P_i + P_i [p/q] e^{C_i} Q_i) + P_L [p/q] e^{\mathcal{L}_L} P_L$$

Overall we obtain: the **Multiwavelet Padé Exponential Model**.

Lipschitz Analysis of the Padé Neural Operator

Theorem

Given a linear operator $\mathcal{L} = \mathcal{L}(\theta_{\mathcal{L}})$ (or, a Lipschitz operator with Lipschitz constant $\|\mathcal{L}\|$ and $\mathcal{L}(0) = 0$), a non-linearity layer $v = \sigma(Wu + b)$, and $p, q \in \mathbb{N}$, at points of differentiability, the gradients of the operation $x \mapsto y = F(x; \theta_{\mathcal{L}}, W, b) := [p/q]e^{\mathcal{L}}(x)$ using the $[p/q]$ Padé neural operator are bounded in operator norm by

$$\left\| \frac{\partial y}{\partial \theta_{\mathcal{L}}} \right\| \leq \exp(\|\mathcal{L}\|) (\|b\|_2 + \|W\| \|x\|_2) \left(\sum_{j=1}^{n_{\theta}} \left\| \frac{\partial \mathcal{L}}{\partial \theta_j} \right\|^2 \right)^{1/2}, \quad (1)$$

$$\left\| \frac{\partial y}{\partial W} \right\| \leq \exp(\|\mathcal{L}\|) \|x\|_2, \quad (2)$$

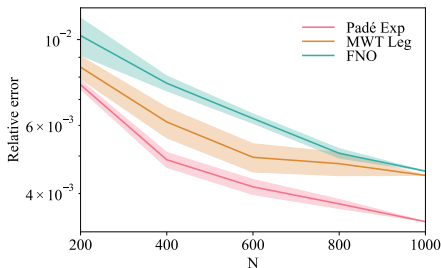
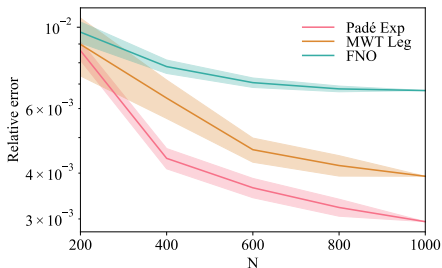
$$\left\| \frac{\partial y}{\partial b} \right\| \leq \exp\left(\frac{p}{p+q} \|\mathcal{L}\|\right). \quad (3)$$

Remarks: The polynomials $A_{pq}(\mathcal{L})$ and $B_{pq}(\mathcal{L})$ are implemented as recurrent networks. This theorem guarantees the gradients do not explode with $p, q \Rightarrow \infty$.

Testing on PDEs

Data Efficiency

How fast the training error decays (“data efficiency”) w.r.t. number N of training samples, for Korteweg - de Vries (KdV, left), and Kuramoto-Shivashinski (SV, right):



Number of training samples N vs performance (relative L2 error) for neural operators evaluated on the KdV equation with $s=1024$. For $N < 1000$, each smaller dataset is sampled uniformly randomly 5 times from the complete dataset ($N = 1000$) and mean \pm std.dev (shaded region) results are shown across the sampling experiments. (Right) Same analysis for KS equation with $s=1024$.

Testing on PDEs

Sensitivity to Input Resolution

Korteweg-de Vries (KdV) equation benchmarks for different input resolution s . The relative L2 errors are shown for each model.

Networks	$s = 64$	$s = 128$	$s = 256$	$s = 512$	$s = 1024$
Padé Exp	0.00301	0.00308	0.00311	0.00298	0.00295
MWT Leg	0.00372	0.00369	0.00391	0.00408	0.00392
FNO	0.00663	0.00676	0.00657	0.00649	0.00672
MGNO	0.12507	0.13610	0.13664	0.15042	0.13666
LNO	0.04234	0.04764	0.04303	0.04465	0.04549
GNO	0.13826	0.12768	0.13570	0.13616	0.12521

GNO: Graph Neural Operator (Li, 2020); *MGNO*: Multi-level version of GNO (Li, 2020); *LNO*: low-rank representation of the integral operator kernel, à la DeepONet (Lu, 2020); *FNO*: Fourier Neural Operator (Li, 2020); *MWT Leg*: MWT with Legendre OPs;

Epidemic Forecasting (COVID19)

Problem Specifications

Epidemic Forecasting is an example where the dynamical system is unknown (or it may not be deterministic). Neural operators provide an entirely data-driven approach and are capable to learn PDE agnostic maps.

Dataset COVID-19 from April 12, 2020 to August 28, 2021 provided by JHU.

Data from 50 US states, and for each state, total counts of daily reported confirmed (C), recovered (R), and deaths (D). Data in each state is normalized by the respective state total population. Total data: array of $50 \times 3 \times 484$ numbers.

Task: The forecasting problem is to learn the map between 14 consecutive counts (C,R,D) to next 7 days data for each of the 50 US state. Let d_t denote the 50×3 array on day t . Then the operator map can be written as:

$$T(\underbrace{(d_{-14}, d_{-13}, \dots, d_{-1})}_{u_0(x)}) = (\underbrace{(d_0, d_1, \dots, d_6)}_{u(\tau, x)}).$$

Challenge: Due to data scarcity, we do a 10-fold resampling of the dataset for additional training/testing samples.

Epidemic Forecasting (COVID19)

Prediction Benchmarks

COVID-19 prediction benchmarks for different networks using 10-fold resampling with mean \pm std. dev. across folds. The Mean Average Error (MAE) is presented for Confirmed (C), Recovered (R), and Deaths (D) counts averaged across 7 days of prediction for 50 US states. The relative L2 error is the test error for each model. The last column compares each network vs FC (auto-regressive fully connected network) in terms of the total MAE improvement and total model parameters difference.

Networks	MAE			Relative L2 error	Net. vs FC
	C	R	D		
Padé Exp	1219 \pm 130	1752 \pm 666	211 \pm 31	0.0155 \pm 0.0034	82.14% (+652K)
MWT Leg	3554 \pm 1157	2928 \pm 1338	284 \pm 209	0.0245 \pm 0.0043	62.0% (+18M)
FNO 3D	4213 \pm 391	3391 \pm 1233	592 \pm 157	0.0301 \pm 0.0045	54.0% (+1.02M)
LNO 3D	28502 \pm 12698	6586 \pm 3442	1465 \pm 965	0.1056 \pm 0.0394	-105.0% (+238K)
Neural ODE	4339 \pm 1174	3443 \pm 1408	443 \pm 192	0.0310 \pm 0.0069	53.8% (+172K)
Seq2Seq	2798 \pm 456	3317 \pm 1690	346 \pm 83	0.0273 \pm 0.0058	63.7% (+1.8M)
Transformer	7087 \pm 972	6613 \pm 2853	1722 \pm 320	0.0501 \pm 0.0094	13.4% (+15.2K)
FC	10305 \pm 2818	5885 \pm 1609	1634 \pm 686	0.0609 \pm 0.0111	(37.2K)

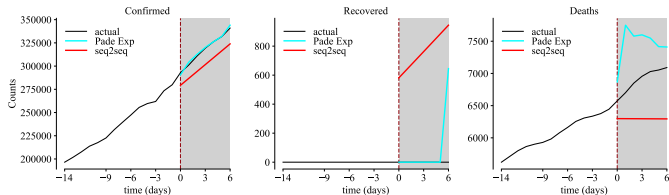
seq2seq, transforms, FC are non-neural operators.

Epidemic Forecasting (COVID19)

A Tale of Two States

COVID19 Forecasting. Confirmed, Recovered, and Deaths count forecasting results for the 07/07/20 – 07/13/20 (chosen arbitrarily) using previous 2 weeks as the input. The Padé Exp prediction and the best non-neural operator scheme from previous table (seq2seq) is shown.

California:
39.77M population.



Massachusetts:
6.89M population.

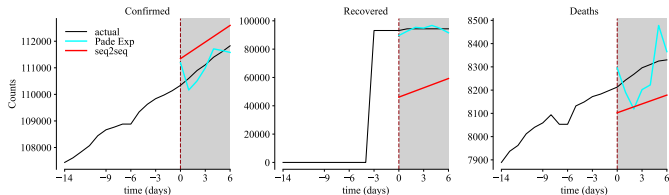
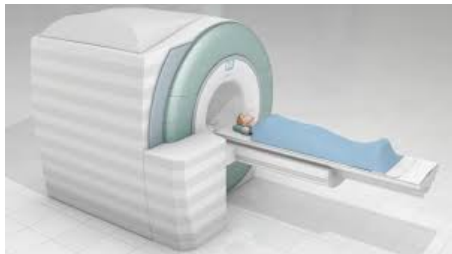


Table of Contents

- 1 Modeling Deterministic Evolution Operators using Deep Networks (Gupta, Xiao, R.B., Bogdan)
 - Problem Formulation: IVP
 - Architecture
 - Lipschitz Analysis
 - Experiments
- 2 Uncertainty Quantification in NN (Wang, Ludwig, Rawson, R.B., Ernst)
 - MRI and NN
 - Uncertainty Propagation through NN
 - Experimental Results
- 3 Chart based Normalizing Flows (Dock, Sidheek, Jain, R.B., Singh)
 - Global normalizing flows
 - Conformal embedding flows
 - Chart based flows - model
 - Chart based flows - performance

MRI Model



The measurement model. For coil $k \in [N_c]$,

$$x_k = \mathcal{F}(S_k z) + \nu_k$$

where \mathcal{F} is the Fourier acquisition matrix, S_k is the diagonal matrix with the coil k sensitivity map, ν_k is measurement noise, and z is the brain signal.

Figure: Credits: hopkinsmedicine.org

Knowns: F, x_1, \dots, x_{N_c} . Unknowns: $S_1, \dots, S_{N_c}, \nu_1, \dots, \nu_{N_c}, z$. Target: z .

Lots of research, lots of Nobel prizes, lots of companies (Siemens, GE, Philips), lots of techniques (compressive sampling, GRAPPA, SENSE, ...) to solve the inverse problem: $z = G(\text{measurements})$.

More recent: Use of Deep Neural Networks.

The MRI Inverse Problem

At an abstract level, the forward model, $z \mapsto x$ and the reconstruction (inverse) model, $x \mapsto \hat{z}$ are:

$$x = F(z) + \nu \quad , \quad \hat{z} = G(x).$$

To fix notations: the target (brain) signal $z \in \mathbb{R}^d$, the measured (acquired) signal $x \in \mathbb{R}^n$.

The DNN approach proposes to implement G using certain Neural Network architectures. Out of many architectures out there, we focused on a specific network, namely the *end-to-end variational neural network (E2E-VarNet)* introduced by Sriram, et al, at MICCAI 2020.

Our problem: Given a trained network that implements a reconstruction algorithm G , quantify the level of uncertainty per reconstructed pixel.

Assumption: We assume the network has been trained well enough so that $G(F(z)) = z$, i.e., perfect reconstruction in the absence of noise.

Uncertainty Propagation through NN

CRLB and FIM

The standard way of quantifying uncertainty is through the Cramer-Rao Lower Bound (CRLB). The CRLB has been used many times for experimental design in Medical Imaging and elsewhere. Fisher Information Matrix $I(z)$ and $CRLB$:

$$I(z) = \mathbb{E} \left[(\nabla_z \log(p(x; z))) (\nabla_z \log(p(x; z)))^T \right], \quad CRLB = (I(z))^{-1}$$

Interpretation: Covariance of any *unbiased* estimator of z is lower bounded $CRLB$. Assume further, the noise is AWGN with variance σ^2 . A simple computation yields:

$$CRLB = \sigma^2 (J_F^T J_F)^{-1}, \quad J_F = \left[\frac{\partial F_k}{\partial z_j} \right]_{(j,k) \in [n] \times [d]} \in \mathbb{R}^{n \times d}$$

where J_F denotes the Jacobian matrix of the forward model.

Goal: Determine $CRLB$ and use it to measure the confidence in the reconstructed image \hat{z} .

Challenge: The exact form of F is unknown!

The CRLB and the Jacobian of the NN

Our main theoretical result is to connect $CRLB = (I(z))^{-1}$ and the Jacobian of G , J_G .

A simple lemma:

Lemma

Assume $A \in \mathbb{R}^{n \times d}$ is full rank with $n \geq d$.

- ① For any $B \in \mathbb{R}^{d \times n}$ such that $BA = I_d$ (i.e., a left inverse), $BB^T \geq (A^T A)^{-1}$.
- ② If $B_0 = (A^T A)^{-1} A^T$ is the pseudo-inverse of A then, $B_0 B_0^T = (A^T A)^{-1}$.

Consequence:

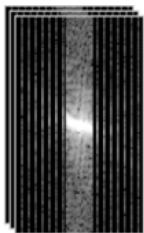
$$CRLB = \sigma^2 J_{G_0} J_{G_0}^T, \quad G_0 = \operatorname{argmin}_{G: G(F(z))=z} \operatorname{trace}(J_G J_G^T)$$

Remarks:

1. The objective function above can provide an additional regularization term in the loss function used by the neural network training.
2. The importance of Jacobians has been shown by (Antun et al, 2020), “On instabilities of deep learning in image reconstruction ...”

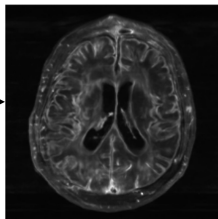
Architecture

Input k-space
fastMRI¹



Uses U-Net for
some computational
steps

End-to-end
Variational Network²



Acceleration Factor: 6~12, ACS: 24, Matrix Size: 320×320

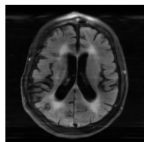
1. Zbontar J, Knoll F, Sriram A, et al. *fastMRI: An Open Dataset and Benchmarks for Accelerated MRI*. Published online November 21, 2018. Accessed November 10, 2021. <https://arxiv.org/abs/1811.08839v2>
2. Sriram, Anuroop, et al. "End-to-end variational networks for accelerated MRI reconstruction." *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, Cham, 2020. (Facebook AI Research and NYU)

Results (1)

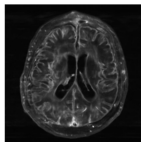
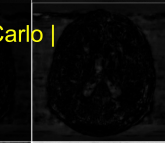
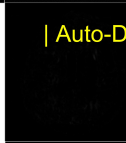
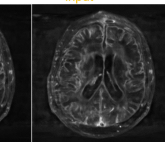
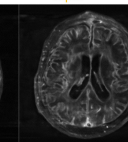
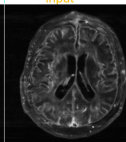
Results

Auto-Diff

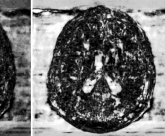
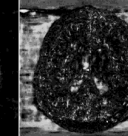
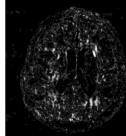
Monte-Carlo Simulation



$$\frac{\text{RMSE}}{\sigma_{\text{input}}}$$


 $\sigma_{\text{input}} = 1\%$
 $\sigma_{\text{input}} = 8\%$
 $\sigma_{\text{input}} = 16\%$


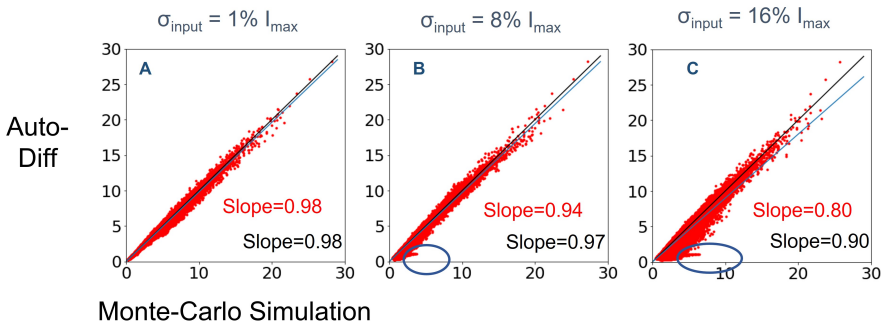
| Auto-Diff - Monte-Carlo |



- Noise amplification is structured: generally higher at sharp edges
- Auto-Diff agrees well with Monte-Carlo
- Agreement poorer at higher noise levels → non-linearity of NN?
- Deviations more pronounced in regions of low signal intensity (e.g., background and in ventricles)

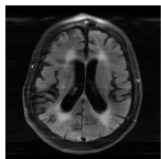
Results (2)

Pixel-by-pixel $\frac{\text{RMSE}}{\sigma_{\text{input}}}$: Auto-Diff versus MC

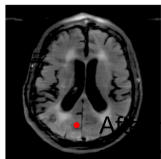


- Auto-Diff (Linear Model) agrees well with Monte-Carlo Simulation
- Agreement is less strong with higher noise level
- The outlying voxels are mostly in background and ventricles

Results (3)



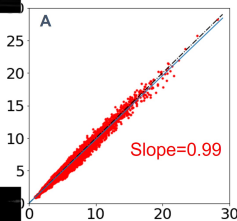
↓
×Auto-diff



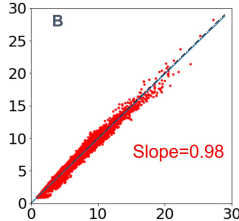
$\frac{\text{RMSE}}{\sigma_{\text{input}}}$ with masking: Auto-Diff versus MC



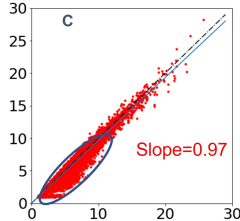
$\sigma_{\text{input}} = 1\% I_{\text{max}}$



$\sigma_{\text{input}} = 8\% I_{\text{max}}$



$\sigma_{\text{input}} = 16\% I_{\text{max}}$



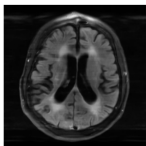
Monte-Carlo Simulation

After masking, very good agreement between MC and the Auto-Diff even at very high noise levels

- However, Auto-Diff still tends to underestimate noise

Results (4)

Reconstruction
without Noise



$|\text{Bias}| / \sigma_{\text{input}}$

$\text{RMSE} / \sigma_{\text{input}}$

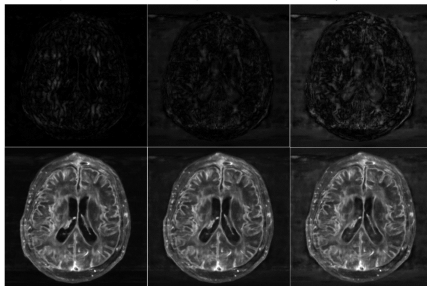
($\text{RMSE} = |\text{Bias}|^2 + \text{Variation}$)

Monte Carlo Simulation - Bias

$\sigma_{\text{input}} = 1\%$

$\sigma_{\text{input}} = 8\%$

$\sigma_{\text{input}} = 16\%$



- As the standard deviation of noise increases, so does the bias.
- This may also have contributed to the divergence between the auto-diff and Monte-Carlo simulation
- However, even at the highest noise level, the bias was lower than the RMSE

Table of Contents

- 1 Modeling Deterministic Evolution Operators using Deep Networks (Gupta, Xiao, R.B., Bogdan)
 - Problem Formulation: IVP
 - Architecture
 - Lipschitz Analysis
 - Experiments
- 2 Uncertainty Quantification in NN (Wang, Ludwig, Rawson, R.B., Ernst)
 - MRI and NN
 - Uncertainty Propagation through NN
 - Experimental Results
- 3 Chart based Normalizing Flows (Dock, Sidheek, Jain, R.B., Singh)
 - Global normalizing flows
 - Conformal embedding flows
 - Chart based flows - model
 - Chart based flows - performance

Deep latent variable models (DLVMs)

Given samples $X = (x_n)_{n=1}^N \in \mathcal{X}$ drawn from an unknown distribution $p(x)$, the goal of generative machine learning is to obtain new and “realistic” samples also drawn from $p(x)$. One way to do this is to assume that most of the variation in the unknown distribution arises from a latent variable z that is simply distributed according to $q(z)$ (usually Gaussian), in which case Bayes gives

$$p(x) = \int_{\mathcal{Z}} p(x|z)q(z)dz$$

Once $p(x|z)$ is known, new samples can be generated by first sampling z_0 from $z \sim q(z)$ and then sampling $x \sim p(x|z = z_0)$. LVM's are useful for

- Data Augmentation (by generating new samples that follow the same distribution as the data)
- Domain Adaptation (the latent space provides a common representation between domains)
- Outlier Detection
- Generating realistic samples

Deep latent variable models (DLVMs)

When $p(x|z) := p_\theta(x|z)$ are parameterized as DNNs, such models are termed DLVMs. Maximum likelihood estimation for θ gives:

$$\operatorname{argmax}_\theta \log p_\theta(X) = \operatorname{argmax}_\theta \sum_{n=1}^N \log \int_{\mathcal{Z}} p_\theta(x_n|z) q(z) dz$$

When $p_\theta(x|z)$ is given by a DNN, this objective is intractable to evaluate, let alone optimize.

- VAEs instead optimize the following variational lower bound for $\log p_\theta(X)$ that holds for any distribution $q_\phi(z|x)$, with equality when $q_\phi(z|x) = p(z|x)$:

$$\log p_\theta(X) \geq \sum_{n=1}^N \mathbb{E}_{z \sim q_\phi(\cdot|x_n)} [\log p_\theta(x_n|z)] - D_{KL}(q_\phi(\cdot|x_n) || q(z))$$

- GANs do not directly model $p(x|z)$, instead they sample $Z = (z_n)_{n=1}^N$ from $z \sim q(z)$ and seek to learn a generator function $G_\theta : \mathcal{Z} \rightarrow \mathcal{X}$ that minimizes an adversarial objective $L(X, G_\theta(Z))$. For example Goodfellow et al. take

$$L(X, Y) = \max \sum_{n=1}^N \frac{1}{n} \log D_\phi(y_n) + \frac{1}{n} \log(1 - D_\phi(x_n))$$

Global normalizing flows

Neither VAEs or GANs can provide exact densities $p(x)$, as VAEs replace $\log p(x)$ by a lower bound and GANs do not have an explicit probability model. NFs, however, make the assumption that $p_\theta(x|z)$ is of the form

$$p_\theta(x|z) = \delta(x - g_\theta(z))$$

Where $g_\theta : \mathcal{Z} \rightarrow \mathcal{X}$ is a diffeomorphism with inverse $f_\theta : \mathcal{X} \rightarrow \mathcal{Z}$. In other words, at the level of the random variables x and z it is assumed that

$$x = g_\theta(z) \quad z = f_\theta(x)$$

Note further that NFs are the $\sigma \rightarrow 0$ limit of the VAE given by $p_\theta(x|z) = \mathcal{N}(g_\theta(z), \sigma^2 \mathbb{I})$ and $q_\theta(z|x) = \mathcal{N}(f_\theta(x), \sigma^2 \mathbb{I})$.

Change of variables gives

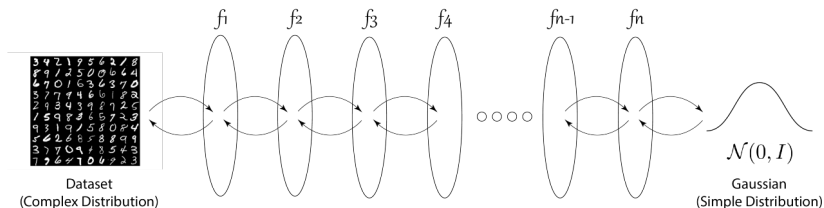
$$\begin{aligned} \log p_\theta(X) &= \sum_{n=1}^n \log \int_{\mathcal{Z}} p_\theta(x|z) q(z) dz \\ &= \sum_{n=1}^n \log |\text{Det}[Jf_\theta(x_n)]| + \log q(f_\theta(x_n)) \end{aligned}$$

Global normalizing flows

To compute θ for a NF one would like to maximize

$$\log p_{\theta}(X) = \sum_{n=1}^n \log |\text{Det}[Jf_{\theta}(z)]| + \log q(f_{\theta}(z))$$

Because computing $\text{Det}[Jf_{\theta}]$ is intractable for an arbitrary deep neural network, one builds f out of compositions $f_{\theta} = f_L^{\theta_L} \circ \dots \circ f_1^{\theta_1}$ where $\text{Det}[Jf_k^{\theta_k}(z)]$ and $g_k^{\theta_k} = (f_k^{\theta_k})^{-1}$ are simple to compute and $\theta = \text{vec}(\theta_1, \dots, \theta_L)$.



In this case the log-likelihood breaks apart to produce a tractable objective:

$$\log p_{\theta}(X) = \sum^N \left\{ \log q(f_{\theta}(x_n)) + \sum^L \log |\text{Det} Jf_{f_j}^{\theta_j}(x_n)| \right\}$$

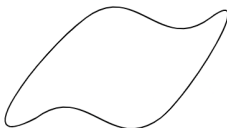
Global normalizing flows

The ability to exactly compute $p(x)$ makes NFs a powerful generative method, but they have an Achilles heel: they are diffeomorphisms. This means that the data manifold must be topologically equivalent to the latent space in order for NFs to get good results. In particular, the data manifold must have *the same dimension* as the latent space. The *manifold hypothesis*, however, suggests that often real data (like images) lies on a much lower dimensional submanifold $\mathcal{M} \subset \mathcal{X}$.

Latent Space \mathbb{R}^d



Topologically Equivalent



Topologically Distinct

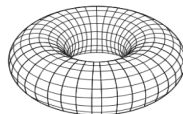


Figure: Topological constraints on an NF. Anything with "non-trivial topology" will cause an NF to struggle.

Conformal embedding flows

One way to do dimensionality change using a NF $g_\theta : \mathcal{Z} \rightarrow \mathcal{U}$ is to post-compose it with a dimension raising embedding $h : \mathcal{U} \rightarrow \mathcal{X}$ to form $h \circ g_\theta$. Unfortunately the resulting measure change factor

$$|\text{Det}[(JhJg_\theta)^T(JhJg_\theta)]|^{-\frac{1}{2}} = |\text{Det}[Jg_\theta^T Jh^T JhJg_\theta]|^{-\frac{1}{2}}$$

does not separate into a product. A solution is to restrict h to be conformal [?]:

$$\mathcal{C}(\mathbb{R}^d \rightarrow \mathbb{R}^D) := \{c \in C^1(\mathbb{R}^d \rightarrow \mathbb{R}^D) \mid \exists \lambda \in C^0(\mathbb{R}) : Jc(u)^T Jc(u) = \lambda(u)^2 \mathbb{I}_{d \times d}\}$$

If $g_\theta : \mathcal{Z} \rightarrow \mathcal{U}$ is a NF and $c \in \mathcal{C}(\mathcal{U} \rightarrow \mathcal{X})$ then the measure change factor is:

$$|\text{Det}[(JcJg_\theta)^T(JcJg_\theta)]|^{-\frac{1}{2}} = |\lambda(u)|^{-1} |\text{Det}Jf_\theta| = |\lambda(u)|^{-1} \prod_{j=1}^L |\text{Det}Jf_j^{\theta_j}|$$

In this case the log-likelihood separates nicely as:

$$\log p_\theta(X) = \sum_{n=1}^N \left\{ \log q(f_\theta(x_n)) + \sum_{j=1}^L \log |\text{Det}Jf_j^{\theta_j}(x_n)| - \log |\lambda(c^\dagger(x_n))| \right\}$$

Where c^\dagger is a pseudo-inverse of c (exactly which pseudo inverse depends on how

Conformal embedding flows

$\mathcal{C}(\mathbb{R}^d \rightarrow \mathbb{R}^D)$ is quite rich, but hard to parameterize. In [?] the authors restrict to embeddings of the form:

$$c = c_J \circ \cdots \circ c_1$$

where each c_j is either a trivially conformal zero padding or a dimension preserving conformal map, which for $d > 2$ are Möbius transformations (by Liouville):

$$M(A, a, b, \alpha, \epsilon)(x) = b + \alpha(Ax - a) / \|Ax - a\|^\epsilon$$

where $A \in O(d)$ is an orthogonal matrix, $\alpha \in \mathbb{R}$, $a, b \in \mathbb{R}^d$, and ϵ is either 0 or 2. Unfortunately if $p_s : \mathbb{R}^d \rightarrow \mathbb{R}^{d+s}$ is the zero padding operation, $m_1 = M(A_1, a_1, b_1, \alpha_1, \epsilon_1)$ is a d dimensional Möbius transformation and m_2 is a $d + s$ dimensional Möbius transformation then for $x \in \mathbb{R}^d$:

$$m_2 \circ p_s \circ m_1(x) = (m_2 \cdot \tilde{m}_1)(p_s(x))$$

Where \tilde{m}_1 is the $d + s$ dimensional Möbius transformation:

$$\tilde{m}_1 = \left(\begin{bmatrix} A_1 & 0 \\ 0 & \mathbb{I}_{s \times s} \end{bmatrix}, p_s(a_1), p_s(b_1), \alpha_1, \epsilon_1 \right)$$

Thus the above yields c as a Möbius transformation of \mathbb{R}^D composed with p_s

Chart based flows - motivation

Vanilla NFs don't perform well for data on a low dimensional manifold $\mathcal{M} \subset \mathcal{X}$.
Current extensions of NFs to allow for dimensionality change restrict expressivity.

Idea:

- \mathcal{M} diffeomorphic to $\mathcal{Z} \simeq \mathbb{R}^d$ is too restrictive. Impediment to performance is topological, suggesting a few “cuts” of the data would greatly improve NFs.
- Use a VQAE $(E, D, \{v_k\}_{k=1}^K)$ to learn $(U_k)_{k=1}^K$, a collection of open sets in \mathcal{X} with $X \in \bigcup_{k=1}^K U_k$. With $V_k = U_k \cap \mathcal{M}$, $(V_k, f_{k,\theta}|_{V_k})$ provides an atlas of charts on \mathcal{M} . Model $p(x)$ as a mixture of normalizing flows

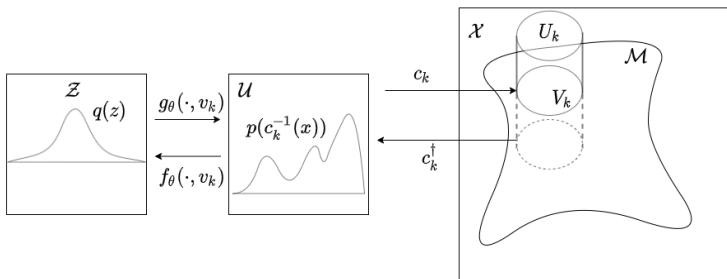
$$p(x|z) = \sum_{k=1}^K p_k \delta(x - g_{k,\theta}(z))$$

Where $g_{k,\theta} : \mathcal{Z} \rightarrow U_k$ is a conformal NF and

$p_k = p(x \in U_k) / \sum_{j=1}^K p(x \in U_j)$. Note $p(x \in V_k) = p(x \in U_k)$ since $p(x)$ is supported on \mathcal{M} . Abusing terminology, we also refer to $(U_k)_{k=1}^K$ as charts.

- Assume that \mathcal{M} is locally conformally flat, and specifically that there exist $D_k \subset \mathcal{U}$ and $c_k : \mathcal{U} \rightarrow \mathcal{X}$ conformal such that $V_k = c_k(D_k)$.

Chart based flows - motivation



Since c_k are Möbius transformations composed with zero paddings, the Riemann measure on V_k is simply a re-scaling of the pullback to the Lebesgue measure on \mathcal{U} . Thus in this sense the invertible normalizing flows $f_\theta(\cdot, v_k)$ are responsible for learning the probability measure $p(x)d_{\mathcal{M}}x$ and the conformal embeddings are responsible for learning the manifold \mathcal{M} .

Chart based flows - probability model

Assume $(U_k)_{k=1}^K$ are known. Let z be a r.v. taking values in \mathcal{Z} and let k be a r.v. taking values in $\{1, \dots, K\}$. Then assume x, z, k are jointly distributed as:

$$p(x, z, k) = \delta(x - g_{k, \theta}(z)) q(z) p_k$$

Where $g_{k, \theta} : \mathcal{Z} \rightarrow U_k$ has (pseudo) inverse $f_{k, \theta} : U_k \rightarrow \mathcal{Z}$. Suppressing θ ,

$$\begin{aligned} p(x, k) &= p_k \int_{\mathcal{Z}} \delta(x - g_k(z)) q(z) dz \\ &= p_k \mathbb{1}_{U_k}(x) \int_{\mathcal{Z}} \delta(z - f_k(x)) |\det[Jg_k(z)]|^{-1} q(z) dz \\ &= p_k \mathbb{1}_{U_k}(x) |\det[Jg_k(f_k(x))]|^{-1} q(f_k(x)) \\ &= p_k \mathbb{1}_{U_k}(x) |\det[Jf_k(x)]| q(f_k(x)) \end{aligned}$$

Thus we obtain the density $p(x)$ as

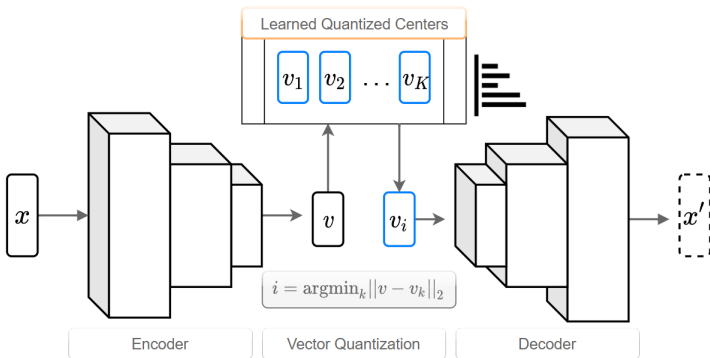
$$p(x) = \sum_{k: x \in U_k} p_k |\det[Jf_k(x)]| q(f_k(x))$$

$$= \sum_k p(k) q(f_k(x)) |\lambda_k(c_k^\dagger(x))|^{-1} \prod_{l=1}^L |\text{Det}[Jf_k^l(f_k^{l+1} \circ \dots \circ f_k^l(x))]|$$

Chart based flows - VQAEs

Given data $\{x\}_{n=1}^N \in \mathcal{X}$ and a latent space \mathcal{V} with $\dim \mathcal{V} \ll \dim \mathcal{X}$ a VQAE seeks to learn an encoder $E : \mathcal{X} \rightarrow \mathcal{V}$, a decoder $D : \mathcal{V} \rightarrow \mathcal{X}$, and a collection of encoded centers $\{v_k\}_{k=1}^K \subset \mathcal{V}$ so that the following loss is minimized:

$$\mathbb{E}_{x \sim p(x)} [\mathcal{L}(D(\arg \min_{v_k} \|v - E(x)\|_2), x)]$$






The number of centers is increased until the reconstruction error is below a   

Chart based flows - chart design

We would like charts $(U_k)_{k=1}^K$ that cover X , that overlap, and that are sparse in the sense that no single $x \in X$ is contained in too many charts.

Definition

Given $v_1, \dots, v_K \in \mathcal{V} \simeq \mathbb{R}^d$ the (m, ϵ) -Voronoi cell of v_k is

$$V_k = \{v \in \mathcal{V} \mid \exists J \subset [K] \mid |J| > K - m \text{ and } \|v - v_k\| \leq (1 + \epsilon) \|v - v_j\| \forall j \in J\}$$

Once a VQAE $(E, D, \{v_k\}_{k=1}^K)$ is trained, we can use the pullback through E of (m, ϵ) -Voronoi cells as charts:

$$U_k := \{x \in \mathcal{X} \mid E(x) \in V_k\}$$

Note that checking whether $x \in U_k$ amounts to computing $d_1 := \|E(x) - v_1\|_2$ through $d_k := \|E(x) - v_k\|_2$ and checking whether $d_k \leq (1 + \epsilon) \tilde{d}_m$ where $\tilde{d}_1 \leq \dots \leq \tilde{d}_K$. Here ϵ and m are hyper-parameters of the model. Note that if $m(x) := |\{k : x \in U_k\}|$ then $m(x) \geq m$ and $\lim_{\epsilon \rightarrow 0} m(x) = m$ almost everywhere.

Chart based flows - implementation

Each conformal normalizing flow $g_{1,\theta}, \dots, g_{K,\theta}$ is trained on only data lying in U_k . Even so, training K separate flows $g_{1,\theta}, \dots, g_{K,\theta}$ becomes infeasibly time consuming as K increases (VQAE produces ~ 120 charts for the MNIST dataset), so instead let $g_\theta : \mathcal{Z} \times \mathcal{V} \rightarrow \mathcal{X}$ be such that $g_\theta(z, v_k) \in U_k$ for all z . Then assume

$$g_{k,\theta}(z) = g_\theta(z, v_k)$$

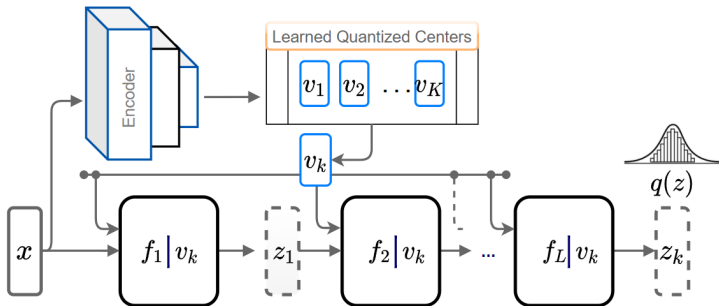


Chart based flows - training

During training the objective function is

$$\begin{aligned} \ln p_{\theta}(X) &= \sum_{n=1}^N \ln p_{\theta}(x_n) = \sum_{n=1}^N \ln \sum_{k: x_n \in U_k} p_k p(x_n | k) \\ &= \sum_{n=1}^N \ln \sum_{k: x_n \in U_k} p_k q(f_k(x_n)) |\lambda_k(c_k^{\dagger}(x_n))|^{-1} \prod_{l=1}^L |\det[Jf_k^l(f_k^{l+1} \circ \dots \circ f_k^L(x_n))]| \end{aligned}$$

Noting that $p(x|k)$ is zero unless $x \in U_k$. The density $p(x)$ can also be written

$$p(x) = \mathbb{E}_{k \sim \tilde{p}_x(k)} [p(x|k)] \underbrace{\sum_{j: x \in U_j} p_j}_{\text{piecewise constant}}$$

Where $\tilde{p}_x(k) = p(k | p(x|k) > 0) = p_k / \sum_{j: x \in U_j} p_j$. During training we replace the expectation $\mathbb{E}_{k \sim \tilde{p}(k)} [p(x|k)]$ with the stochastic quantity $p(x|k)$, $k \sim \tilde{p}(k)$, performing only a single gradient descent pass per data-point as opposed to $m(x)$ passes.

Chart based flows - sampling, inference, and density

- Sampling: Since z and k are independent sample z from $z \sim q(z)$ and $k \sim p_k$ and then compute $x = g_\theta(z, v_k)$.
- Inference: Since z is no longer wholly determined by x , but instead takes values $(f(x, v_k))_{k:x \in U_k}$ with corresponding probabilities $(p(k|x))_{k:x \in U_k}$. One could perform a stochastic inference via sampling $k \sim p(k|x)$ and computing $z = f(x, v_k)$. If deterministic inference is preferred then one may use the expected value of z as $z = \mathbb{E}_{k \sim p(k|x)}[f_k(x)] = \sum_{k:x \in U_k} p(k|x) f_k(x)$ or the most probable value of z as $z = f_s(x)$ where $s = \operatorname{argmax}_{k:x \in U_k} p(k|x)$.
- Density Evaluation: If the exact density $p(x)$ is needed for $x \in \bigcup_{k=1}^K U_k$ it can be computed at the cost of $m(x)$ evaluations of a normalizing flow:

$$p(x) = \sum_{k:x_n \in U_k} p_k q(f_k(x_n)) |\lambda_k(c_k^\dagger(x_n))|^{-1} \prod_{l=1}^L |\det[Jf_k^l(f_k^{l+1} \circ \dots \circ f_k^L(x_n))]|$$

Chart based flows - performance

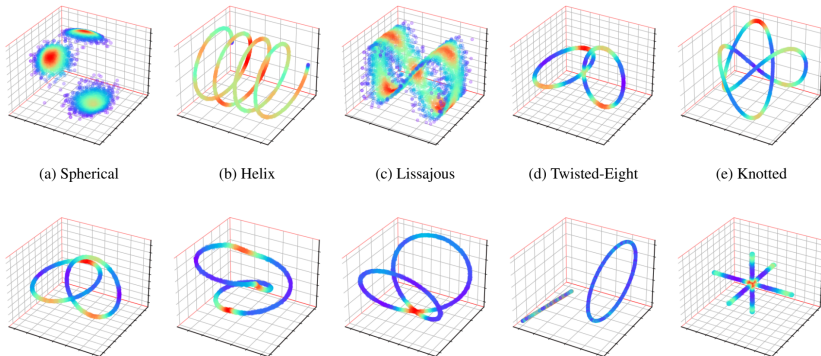


Figure: Toy datasets with various topological features.

Chart based flows - performance

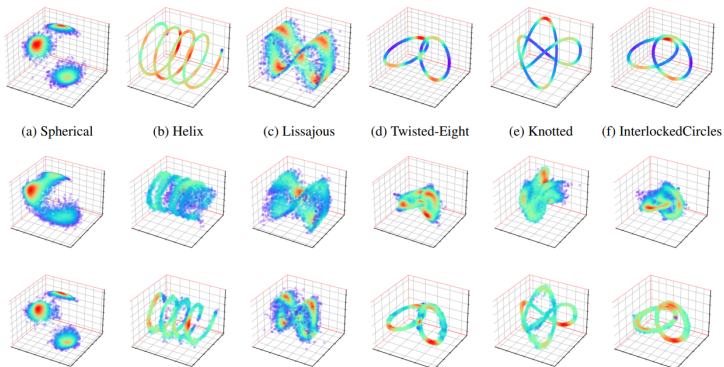


Figure: Qualitative visualization of the samples generated by a classical flow (Middle Row) and its VQ-counterpart (Bottom Row) trained on Toy 3D data distributions (Top Row).

Chart based flows - performance

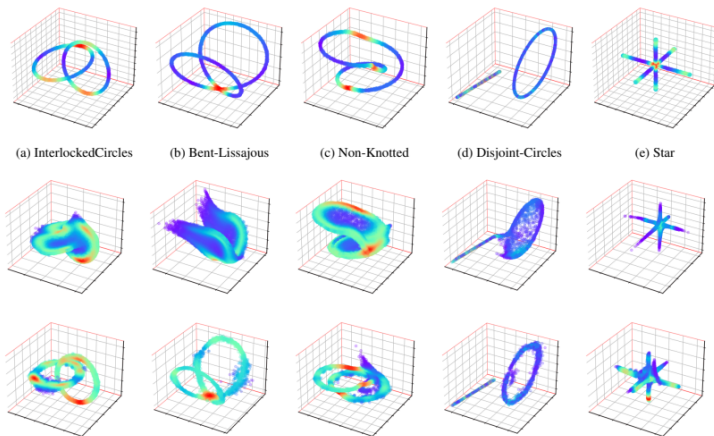


Figure: Qualitative visualization of the samples generated by a classical flow (Middle Row) and its VQ-counterpart (Bottom Row) trained on Toy 3D data distributions (Top Row).

Chart based flows - performance

Model	Spherical	Helix	Lissajous	Twisted-Eight	Knotted	Interlocked-Circles
Real NVP	0.50 ± 0.07	-57.46 ± 2.11	0.18 ± 0.14	-2.72 ± 0.90	-8.65 ± 0.87	-2.18 ± 0.37
VQ-RealNVP	0.99 ± 0.14	-3.85 ± 0.98	0.59 ± 0.08	0.18 ± 0.17	-1.44 ± 0.37	-0.11 ± 0.12
MAF	0.65 ± 0.26	-92.83 ± 5.69	0.12 ± 0.16	-2.77 ± 0.81	-7.04 ± 0.49	-2.49 ± 0.14
VQ-MAF	1.01 ± 0.07	-4.62 ± 0.37	0.59 ± 0.07	-0.32 ± 0.13	-2.44 ± 0.11	-0.15 ± 0.08
CEF	-1.17 ± 0.06	-29.90 ± 2.12	0.38 ± 0.14	-4.03 ± 0.38	-19.40 ± 1.80	-3.42 ± 0.49
VQ-CEF	0.80 ± 3.42	-20.75 ± 2.22	0.49 ± 0.03	-3.51 ± 0.73	-14.44 ± 1.57	-3.23 ± 0.19

Model	Non-Knotted	Bent-Lissajous	Disjoint-Circles	Star
Real NVP	0.53 ± 0.18	1.04 ± 0.22	1.71 ± 0.12	3.33 ± 0.18
VQ-RealNVP	2.39 ± 0.24	2.62 ± 0.13	2.71 ± 0.19	4.23 ± 0.06
MAF	0.73 ± 0.18	1.48 ± 0.11	1.95 ± 0.12	3.53 ± 0.03
VQ-MAF	2.41 ± 0.19	2.06 ± 0.12	2.87 ± 0.07	3.59 ± 0.12
CEF	-0.46 ± 0.13	-0.51 ± 0.16	-0.71 ± 0.21	1.26 ± 0.11
VQ-CEF	-0.15 ± 0.09	-0.54 ± 0.22	0.24 ± 0.15	1.32 ± 0.02

Table: Quantitative evaluation of **Sample Generation** in terms of the log-likelihood of generated samples in nats (higher the better) on the 3D datasets. The values are averaged across 5 independent trials, \pm represents the 95% confidence interval.

Chart based flows - Higher Dimensional Data

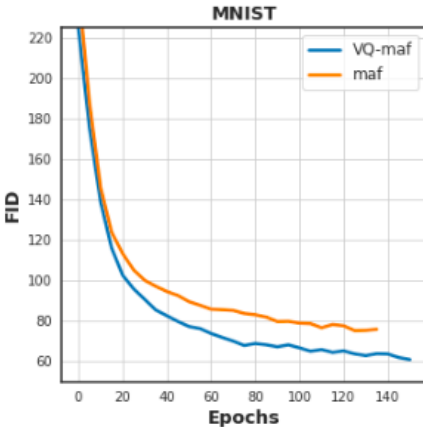
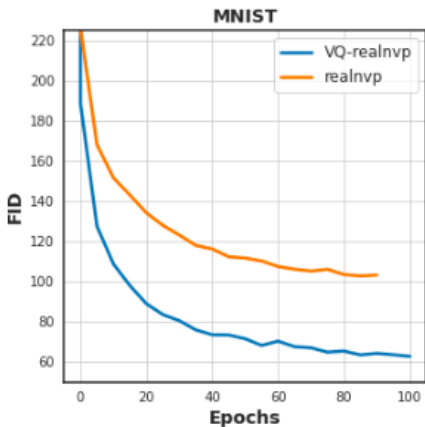


Figure: Seen here are the results of recent experiments on the MNIST dataset. FID is Fréchet Inception Distance (lower is better).

Thank you!

Thank you for listening!
QUESTIONS?

References