Improving Optimization of Likelihood-based Generative Models with One Line of Code

Maurizio Filippone

Statistics Program, KAUST

June 10th, 2024

Density Estimation & Generative Modeling

- Random variable $\mathbf{x} \in \mathbb{R}^D$ with distribution $p_{\text{data}}(\mathbf{x})$
- Samples from $p_{\text{data}}(\mathbf{x})$, $\mathbf{x}_i, \ldots, \mathbf{x}_N$
- Build a model $p_{\boldsymbol{\theta}}(\mathbf{x})$ to estimate $p_{\text{data}}(\mathbf{x})$ from samples
- **Density Estimation**: evaluate $p_{\boldsymbol{\theta}}(\mathbf{x})$ for any given $\mathbf{x}$
- **Generative modeling**: generate new samples from $p_{\boldsymbol{\theta}}(\mathbf{x})$

Generative Modeling

- Likelihood-based models:
  - Mixture models [**Mclachlan**, **1988**]
  - Variational Autoencoders [**Kingma**, **2013**] (VAEs, β-VAEs, ...)
  - Normalizing Flows [**Rezende**, **2015**] (Real-NVPs, GLOW, ...)
- Generative Adversarial Networks:
  - GANs [**Goodfellow**, **2014**] (Wasserstein GANs, Conditional GANs, Progressive GANs, ...)
- Diffusion-based models:
  - Score-based DMs [**Hyvarinen**, **2005; Song**, **2019**] (Latent DMs, Functional DMs, ...)

Likelihood-based Generative Modeling - GLOW example

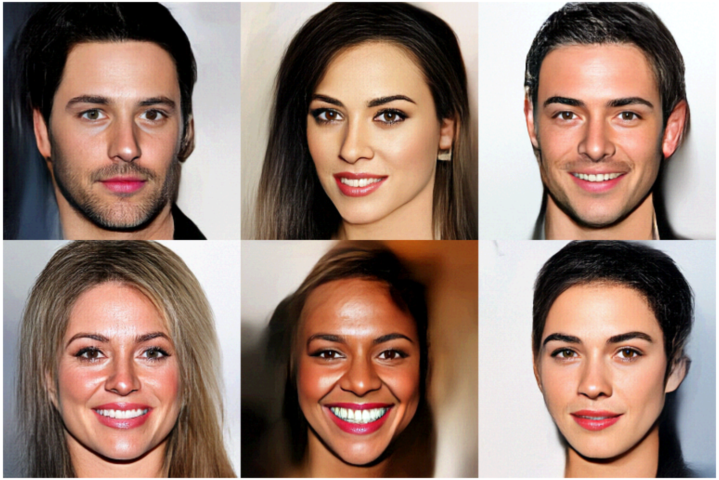


Figure 1: Example of a type of normalizing flow applied to the CelebA data set [**Kingma**, **2018**].

Likelihood-based Generative Modeling - Variational Autoencoders

- Assume (usually low-dim) latent variables $\mathbf{z}$ with $p(\mathbf{z})$ simple and tractable
- **Decoder** $\mathbf{g}_{\boldsymbol{\theta}}(\mathbf{z})$ and likelihood $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) = p(\mathbf{x}|\mathbf{g}_{\boldsymbol{\theta}}(\mathbf{z}))$
- Introduce an approximate $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ and optimize $\boldsymbol{\theta}, \boldsymbol{\phi}$ through variational inference

$$\mathrm{ELBO} = \mathrm{E}_{z \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})] - \mathrm{KL}[q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})||p(\mathbf{z})]$$

- **Encoder** $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ enables amortized inference

Likelihood-based Generative Modeling - Variational Autoencoders

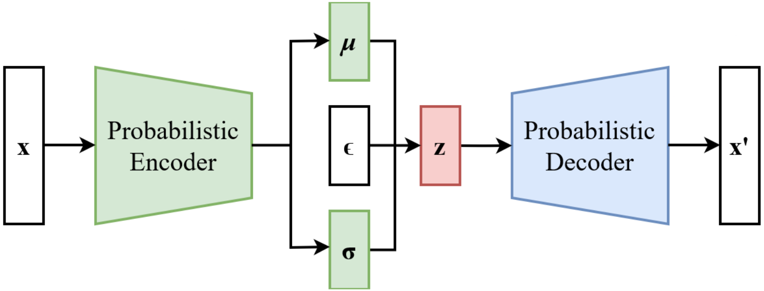


Figure 2: Illustration of the modeling characterizing variational autoencoders.

Likelihood-based Generative Modeling - Normalizing Flows

- Assume latent variables $\mathbf{z}_0$ with $p(\mathbf{z}_0)$ simple and tractable
- Model $\mathbf{x} = \mathbf{g}_{\boldsymbol{\theta}}(\mathbf{z}_0)$ with bijective composition $\mathbf{g}_{\boldsymbol{\theta}}(..)$

$$\mathbf{z}_0 \overset{\mathbf{f}_1}{\longleftrightarrow} \mathbf{z}_1 \overset{\mathbf{f}_2}{\longleftrightarrow} \mathbf{z}_2 \cdots \overset{\mathbf{f}_K}{\longleftrightarrow} \mathbf{x}$$

- Change of variables induces a change of measure through the Jacobians:

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}) = \log p(\mathbf{z}_0) + \sum_{i=1}^{K} \log |\det(d\mathbf{z}_{i-1}/d\mathbf{z}_i)|$$

- **Key idea**: choose transformations whose Jacobian $d\mathbf{z}_{i-1}/d\mathbf{z}_i$ is a triangular matrix

$$\log|\det(d\mathbf{z}_{i-1}/d\mathbf{z}_i)| = \mathtt{sum}(\log|\mathtt{diag}(d\mathbf{z}_{i-1}/d\mathbf{z}_i)|)$$

Likelihood-based Generative Modeling - Normalizing Flows

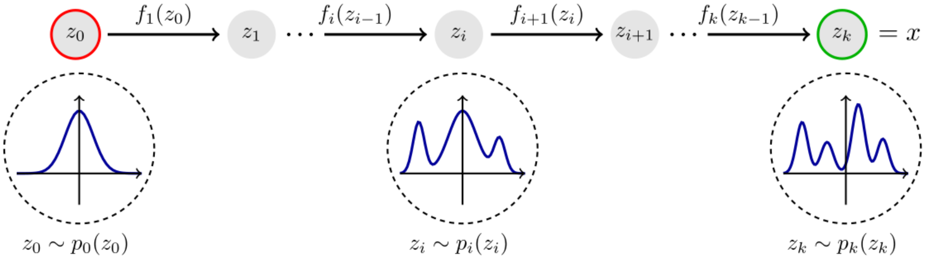


Figure 3: Illustration of the modeling characterizing normalizing flows.

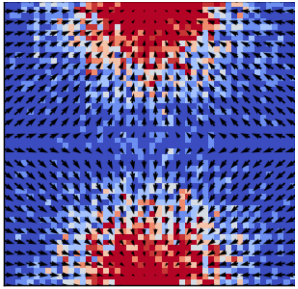
Challenge #1 Density Estimation in Low-Density Regions

Manifold hypothesis: ***High-dimensional data lie on a much lower-dimensional manifold***

The true data density in the input space requires models with high Lipschitz constants

$\longrightarrow$ Models cannot estimate accurately the true density in the tails!

Data distribution $p_{\text{data}}(\mathbf{x})$



Estimated distribution $p_{\theta}(\mathbf{x})$

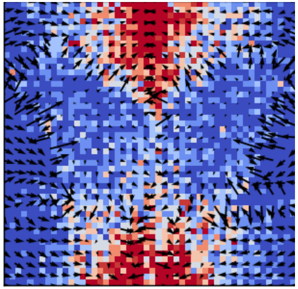


Figure 4: Left: Histogram of samples from $p_{\text{data}}(\mathbf{x})$ and its true scores $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$; **Right:** Histogram of of samples from $p_{\theta}(\mathbf{x})$ and its scores $\nabla_{\mathbf{x}} \log p_{\theta}(\mathbf{x})$.

Challenge #2 Manifold Overfitting

Even when the model distribution $p_{\theta}(\mathbf{x})$ poorly approximates $p_{\text{data}}(\mathbf{x})$, it may reach a high likelihood value by concentrating the density around the correct manifold [**Loaiza-Ganem**, **2022**][1]

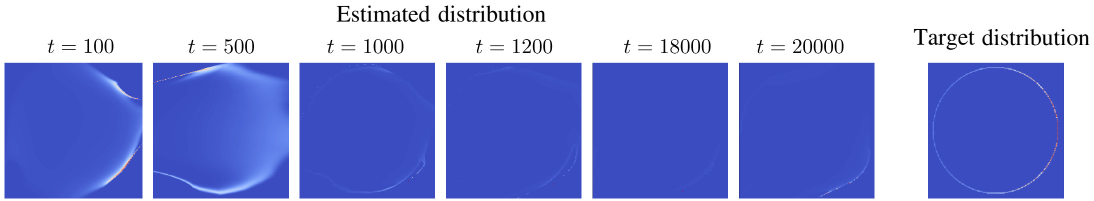


Figure 5: The progression of the estimated densities for the von Mises distribution.

[1] Loaiza-Ganem et al. *Diagnosing and Fixing Manifold Overfitting in Deep Generative Models*. TMLR, 2022.

Motivation - Inspiration from Score-based Diffusion Models

- Score-based diffusion models currently dominate the state-of-the-art of generative models

- Other likelihood-based GMs achieve lower sample quality, but sampling is fast

  Research question: Can we borrow strengths of score-based diffusion models to improve other likelihood-based generative models?

- One distinctive element of score-based diffusion models is *data mollification* [**Song**, **2019**][2]

[2]Song et al. *Generative Modeling by Estimating Gradients of the Data Distribution*. NeurIPS, 2019.

Motivation - Diffusion Models do Great on the Tails!

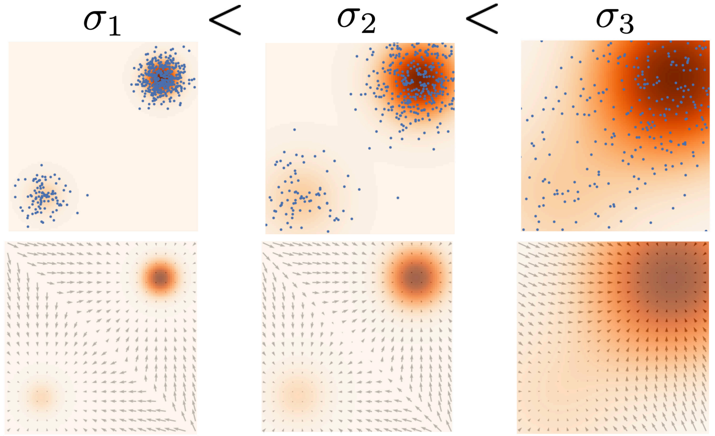


Figure 6: An illustration of the scores obtained by score-based diffusion models for toy mixture densities with increasing variance [**Song**, **2019**].

Data Mollification with Gaussian Noise

Main idea – Data Mollification: Adding Gaussian noise to the data throughout training and gradually reducing its variance until recovering the original data

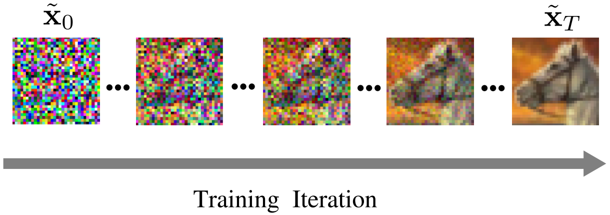


Figure 7: Illustration of Data Mollification.

Data Mollification with Gaussian Noise

- The distribution of the mollified data $\tilde{\mathbf{x}}_t$ at iteration t is defined as follows

$$q(\tilde{\mathbf{x}}_t \mid \mathbf{x}) = \mathcal{N}(\tilde{\mathbf{x}}_t; \alpha_t \mathbf{x}, \sigma_t^2 \mathbf{I})$$

- We adopt the *variance-preserving* formulation used for diffusion models:

$$\alpha_t = \sqrt{1 - \sigma_t^2} \qquad \text{and} \qquad \sigma_t^2 = \gamma(t/T)$$

- Here, $\gamma(\cdot)$ is a noise schedule, which is a monotonically decreasing function

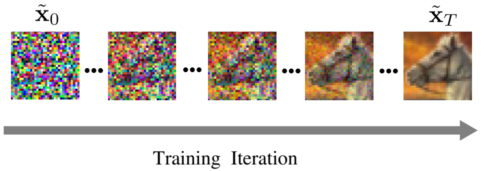


Figure 8: Illustration of Data Mollification.

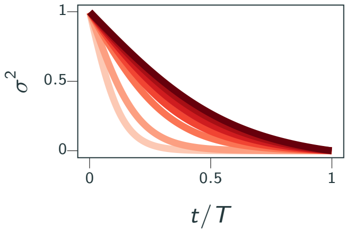


Figure 9: Sigmoid schedule $\gamma(\cdot)$ with different temperatures τ

One-Line-of-Code Data Mollification

Readily applicable to any likelihood-based generative model with a single line of code!

Algorithm 1: Gaussian Mollification

```
for $t \leftarrow 1, 2, ..., T$ do
    $\mathbf{x} \sim p_{\text{data}}(\mathbf{x})$ // Sample training data
    $\tilde{\mathbf{x}}_t = \alpha_t \mathbf{x} + \sigma_t \boldsymbol{\varepsilon}$ // Mollify data with $\alpha_t, \sigma_t^2 \leftarrow \gamma(t/T)$, $\varepsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    $\boldsymbol{\theta}_t \leftarrow \text{UPDATE}(\boldsymbol{\theta}_{t-1}, \tilde{\mathbf{x}}_t)$ // Train the model
```

Alternative View as a Continuation Method

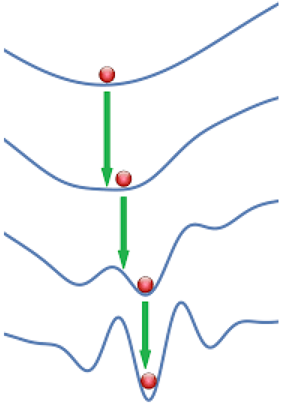


Figure 10: Our approach can be seen as a continuation method, which is a popular way to improve optimization.

Effective Density Estimation in Low-Density Regions

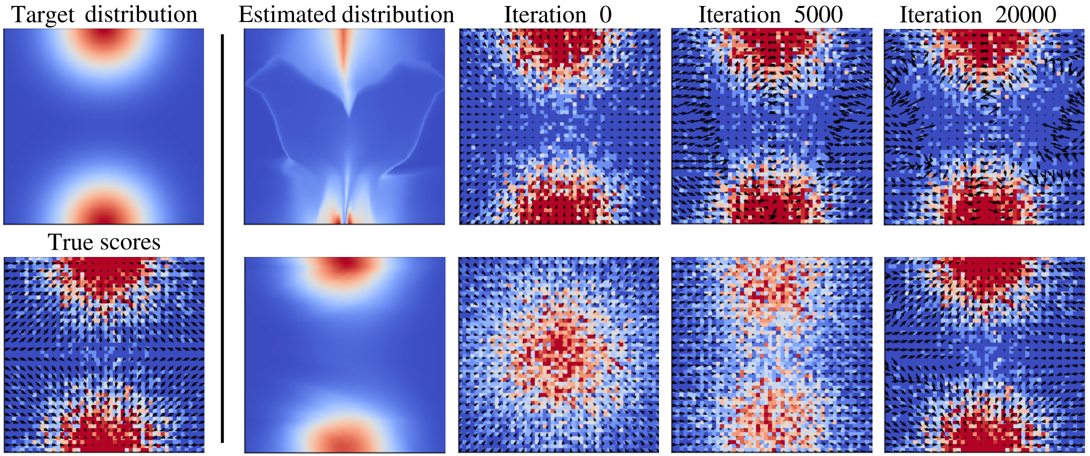


Figure 11: The first column shows the target distribution and the true scores. The second column depicts the estimated distributions for the vanilla (top) and mollification (bottom) training. The remaining columns show histogram of samples from the true (**top row**) and mollified data (**bottom row**), and estimated scores.

Avoid Manifold Overfitting

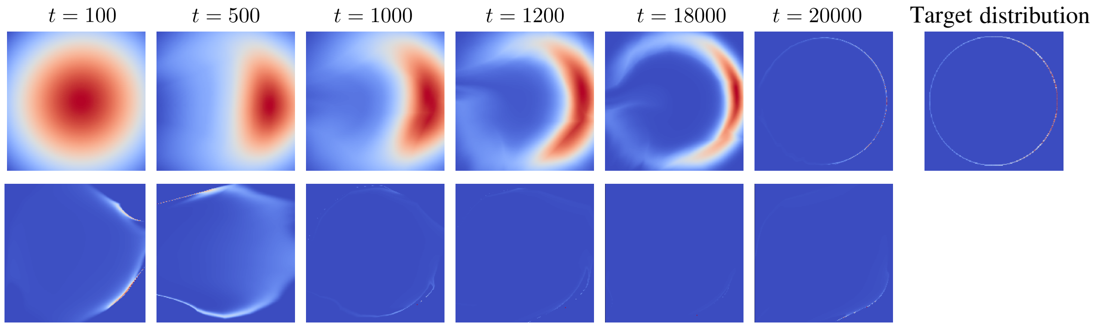


Figure 12: The progression of the estimated densities for the von Mises distribution from the vanilla (**bottom row**) and our mollification (**top row**) approaches.

Density Estimation on UCI benchmark

Table 1: The average test log-likelihood (*higher is better*) on the UCI data sets. Error bars correspond to the standard deviation over 4 runs.

Dataset	MAF [**Papamakarios**, **2017**]		REAL-NVP [**Dinh**, **2017**]		GLOW [**Kingma**, **2018**]	
	VANILLA	MOLLIFICATION	VANILLA	MOLLIFICATION	VANILLA	MOLLIFICATION
RED-WINE	-16.32 ± 1.88	-11.51 ± 0.44	-27.83 ± 2.56	-12.51 ± 0.40	-18.21 ± 1.14	-12.37 ± 0.33
WHITE-WINE	-14.87 ± 0.24	-11.96 ± 0.17	-18.34 ± 2.77	-12.30 ± 0.16	-15.24 ± 0.69	-12.44 ± 0.36
PARKINSONS	-8.27 ± 0.24	-6.17 ± 0.17	-14.21 ± 0.97	-7.74 ± 0.27	-8.29 ± 1.18	-6.90 ± 0.24
MINIBOONE	-13.03 ± 0.04	-11.65 ± 0.09	-20.01 ± 0.22	-13.96 ± 0.12	-14.48 ± 0.10	-13.88 ± 0.08

Image Experiments

Table 2: Comparisions of FID score between vanilla and mollification training on CIFAR10 and CELEBA datasets

Model	CIFAR10			CELEBA		
	VANILLA	GAUSS.	BLURRING	VANILLA	GAUSS.	BLURRING
REAL-NVP	131.15	121.75	**120.88**	81.25	**79.68**	85.40
GLOW	74.62	**64.87**	66.70	97.59	**70.91**	74.74
VAE	191.98	**155.13**	175.40	80.19	**72.97**	77.29
VAE-IAF	193.58	**156.39**	162.27	80.34	**73.56**	75.67
IWAE	183.04	**146.70**	163.79	78.25	**71.38**	76.45
β-VAE	112.42	**93.90**	101.30	67.78	**64.59**	67.08
HVAE	172.47	**137.84**	147.15	74.10	**72.28**	77.54

Comparison on VAEs – Vanilla vs Mollification – Non-curated Samples

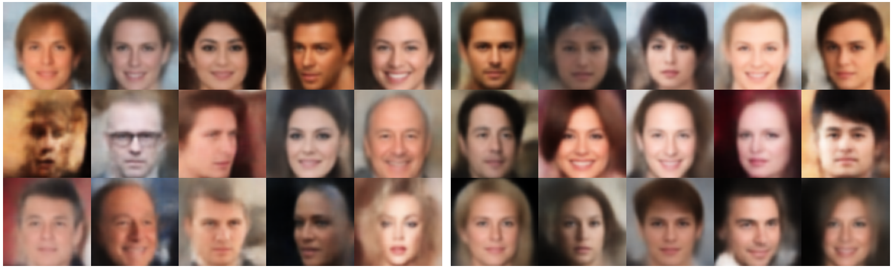


Figure 13: **Left**: samples from a vanilla VAE. **Right**: samples from a VAE trained with mollification.

Conclusions & Ongoing Work

- Performance gains in likelihood-based GMs with a **general**, **cheap**, **simple** code modification!
- However, a gap with diffusion models still exists. . .

Conclusions & Ongoing Work

- Performance gains in likelihood-based GMs with a **general**, **cheap**, **simple** code modification!
- However, a gap with diffusion models still exists. . .
- What about GANs?
- Connections with data augmentation?

References

[1] B.-H. Tran, G. Franzese, P. Michiardi, and M. Filippone. One-Line-of-Code Data Mollification Improves Optimization of Likelihood-based Generative Models. *NeurIPS 2023*.

[2] M. Heinonen, B.-H. Tran, M. Kampffmeyer, and M. Filippone. Robust Classification by Coupling Data Mollification with Label Smoothing. *arXiv:2406.01494* , 2024.

[3] B.-H. Tran, S. Rossi, D. Milios, P. Michiardi, E. V. Bonilla, and M. Filippone. Model Selection for Bayesian Autoencoders. *NeurIPS 2021*.

Thank you!

Questions?