# Advancing Probabilistic Models for Approximate and Exact Inference

By
Robert Giaquinto

University of Minnesota

Advised by Arindam Banerjee

June 30, 2021

# Outline

# Outline

# Motivation

"What I cannot create, I do not understand."

— Richard Feynman

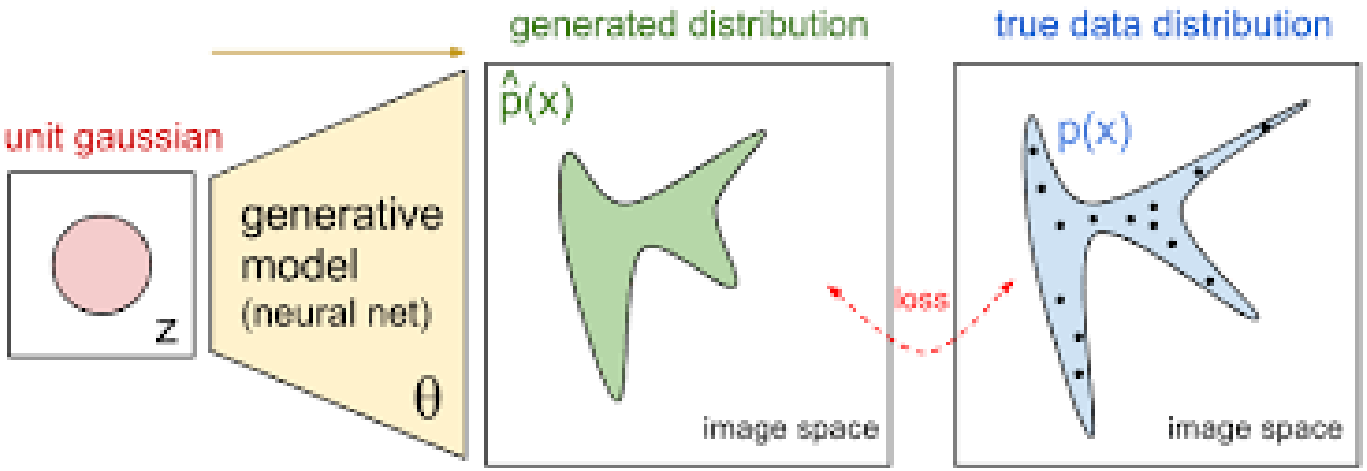



Figure: Seek models that (1) explain all evidence, and (2) are sufficiently flexible.

# Generative Models

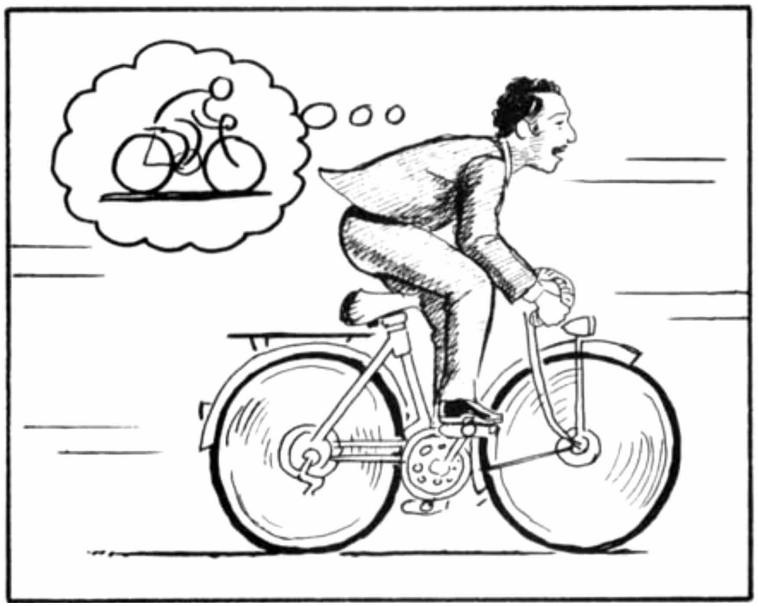


Figure: Finding good internal representations of data.

What: Models full joint distribution

Why:
- Complete model of the data
- Rich features for downstream tasks

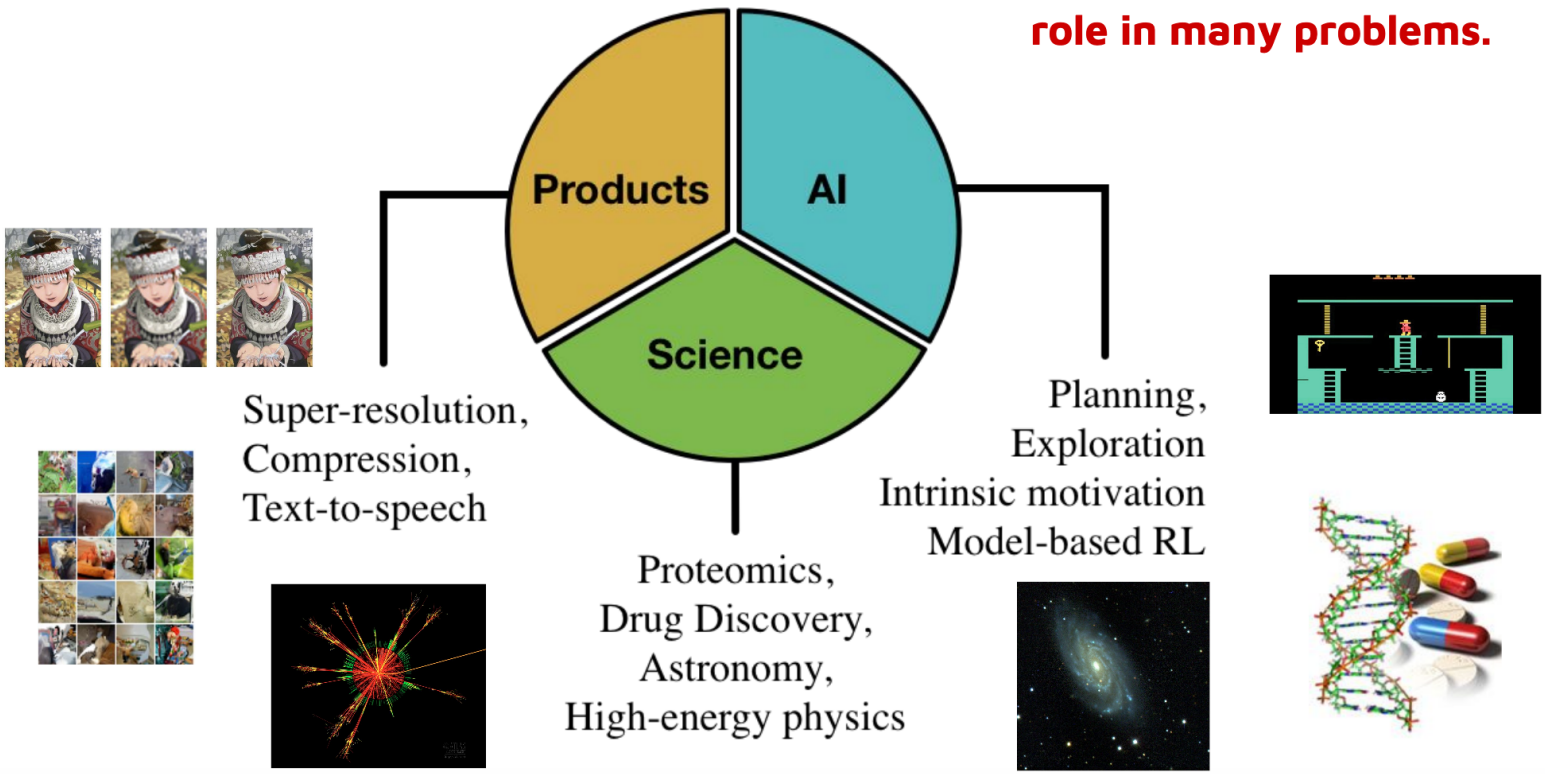
Bonus: Simulate *novel* & *plausible* samples, computing likelihoods, estimate uncertainty

# Problems: Density Estimation & Posterior Approximation

**Use cases:**

- Likelihood of data $p(x)$
- Approximate a posterior with $q(z \mid x)$
- Generating new (plausible) data
- Endless applications[1]:

**Generative models have a role in many problems.**




[1] Image taken from `www.shakirm.com/slides/DeepGenModelsTutorial.pdf`

# Generating Data (Images)

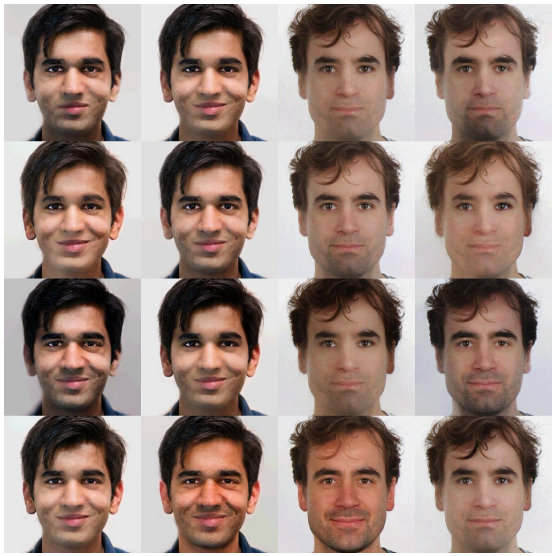


Figure: Interpolation

Figure: Novel synthesis

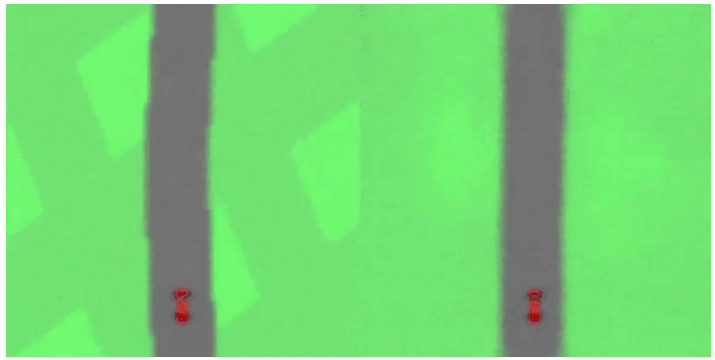


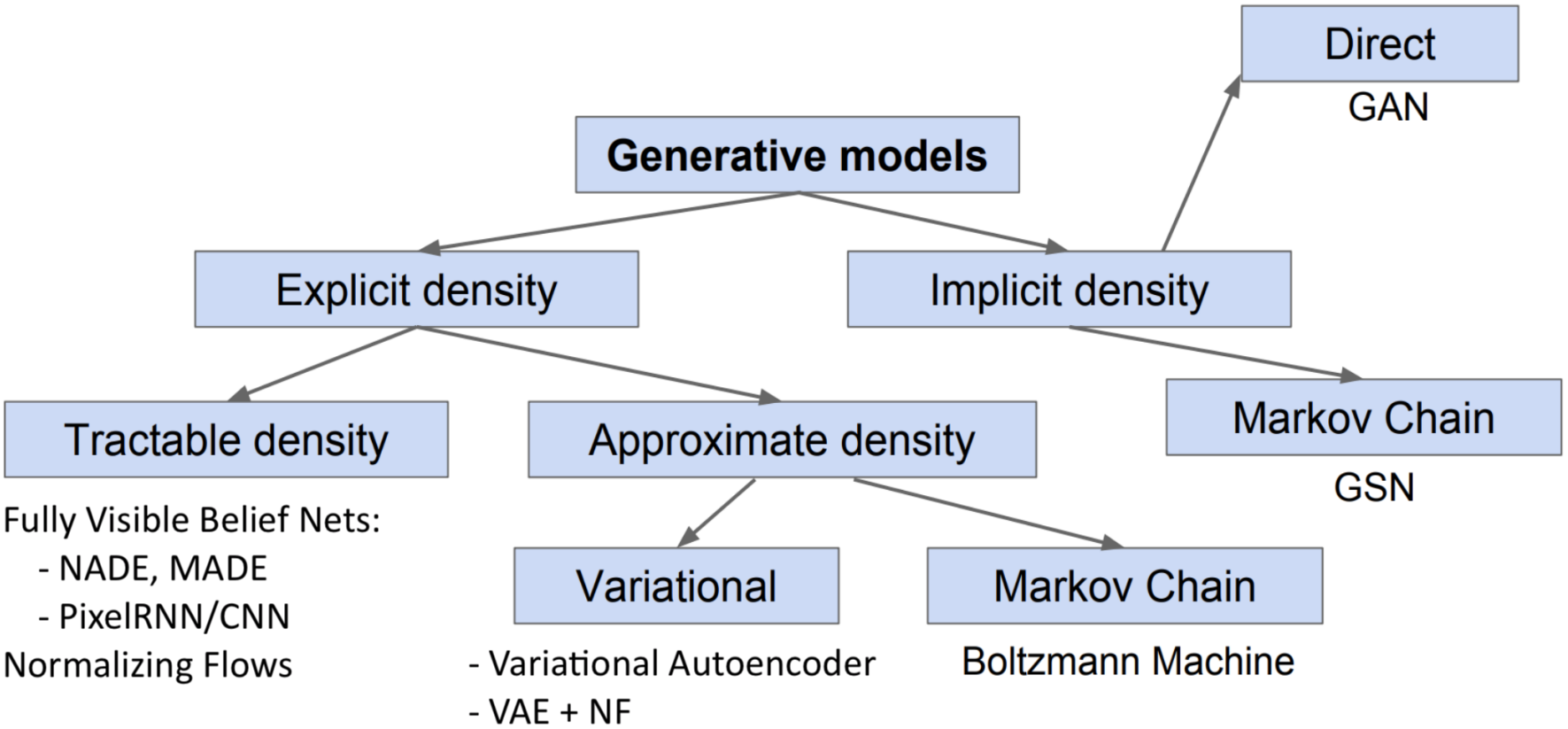
Figure: Agent interacts with *compressed* representation of world

Figure: *Probabilistic* super-resolution

Kingma and Dhariwal 2018; Ha and Schmidhuber 2018

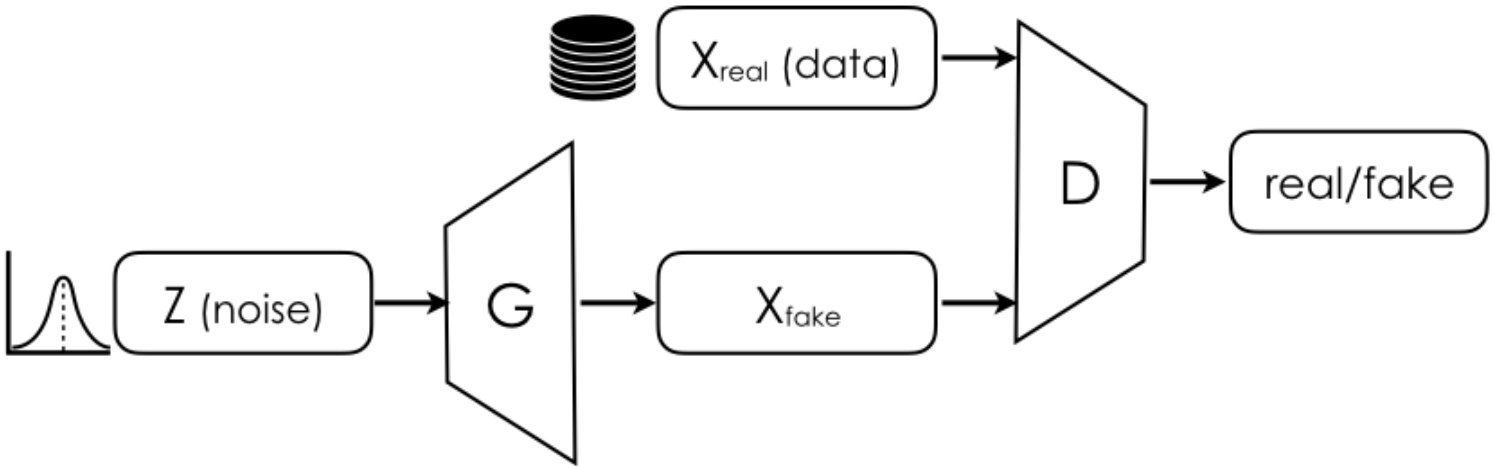
# Outline

# Deep Generative Models: The Larger Context



# Generative Adversarial Networks

- Minimax between Generator $G_\theta$ and Discriminator $D_\phi$




- Generation: Map noise z to data space x: $\mathrm{x} = G_\theta(\mathrm{z}), \mathrm{z} \sim p(\mathrm{z})$
- *Implicit* distribution over x.

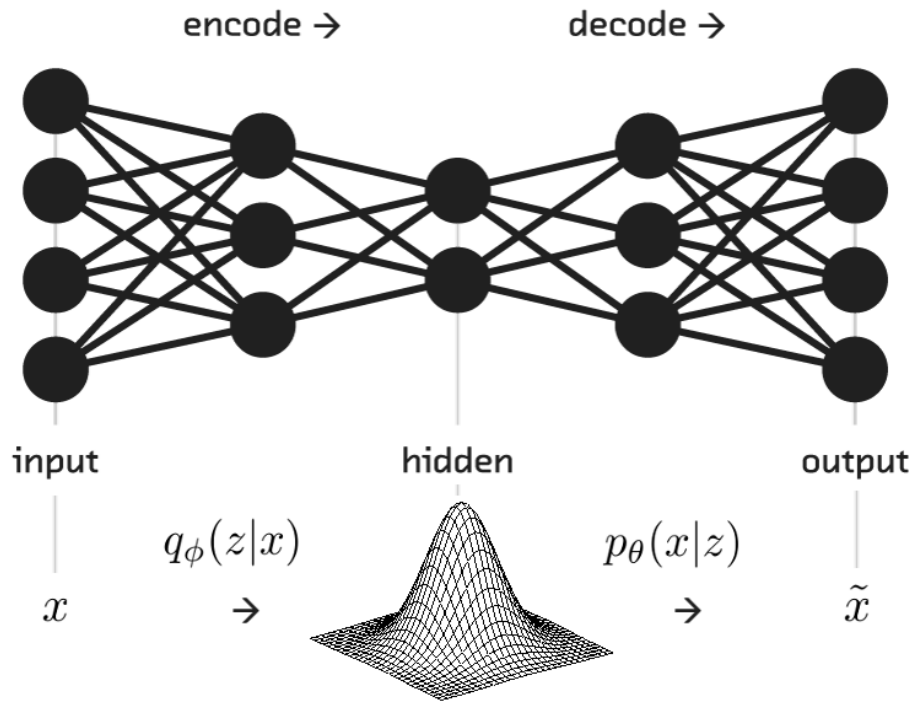
$$\max_D \mathcal{L}_D = \mathbb{E}_{\mathrm{x}\sim p_{data}} [\log D(\mathrm{x})] + \mathbb{E}_{\mathrm{x}\sim G(\mathrm{z}),\mathrm{z}\sim p(\mathrm{z}|y=0)} [\log(1 - D(\mathrm{x}))]$$

$$\min_G \mathcal{L}_G = \mathbb{E}_{\mathrm{x}\sim G(\mathrm{z}),\mathrm{z}\sim p(z)} [\log(1 - D(\mathrm{x}))]$$

Goodfellow et al. (2014), "Generative Adversarial Networks"

# Variational Autoencoder

- VAE: Hidden layer encodes parameters to approximate posterior.
  - E.g. $\mu, \sigma^2 =$ EncoderNetwork(x)
  - *Explicit* distribution.
- Likelihood objective
  - Maximize $\log p(\mathrm{x})$, not just minimizing reconstruction.




Kingma and Welling 2014; Rezende et al. 2014; Kingma and Welling 2019

# Objective for VAE

- Decompose log-likelihood of data x:

$$\begin{aligned}\log p_\theta(\mathrm{x}) &= \mathbb{E}_{q_\phi(\mathrm{z}|\mathrm{x})}\left[\log p_\theta(\mathrm{x})\right] \\ &= \underbrace{\mathbb{E}_{q_\phi(\mathrm{z}|\mathrm{x})}\left[\log\left[\frac{p_\theta(\mathrm{x},\mathrm{z})}{q_\phi(\mathrm{z}\mid\mathrm{x})}\right]\right]}_{\mathcal{L}_{\theta,\phi}(\mathrm{x}) \quad \text{(ELBO)}} + \underbrace{\mathbb{E}_{q_\phi(\mathrm{z}|\mathrm{x})}\left[\log\left[\frac{q_\phi(\mathrm{z}\mid\mathrm{x})}{p^*(\mathrm{z}\mid\mathrm{x})}\right]\right]}_{KL(q_\phi(\mathrm{z}|\mathrm{x})||p^*(\mathrm{z}|\mathrm{x}))} \end{aligned} \tag{1}$$

- z = latent
- Objective: maximize Evidence LOwer Bound (ELBO) term[2]
- KL: how well $q_\phi(\mathrm{z}\mid\mathrm{x})$ approximates true posterior $p^*(\mathrm{z}\mid\mathrm{x})$
- KL is positive $\implies$ ELBO is lower bound on $\log p_\theta(\mathrm{x})$
- What if $q = p$?

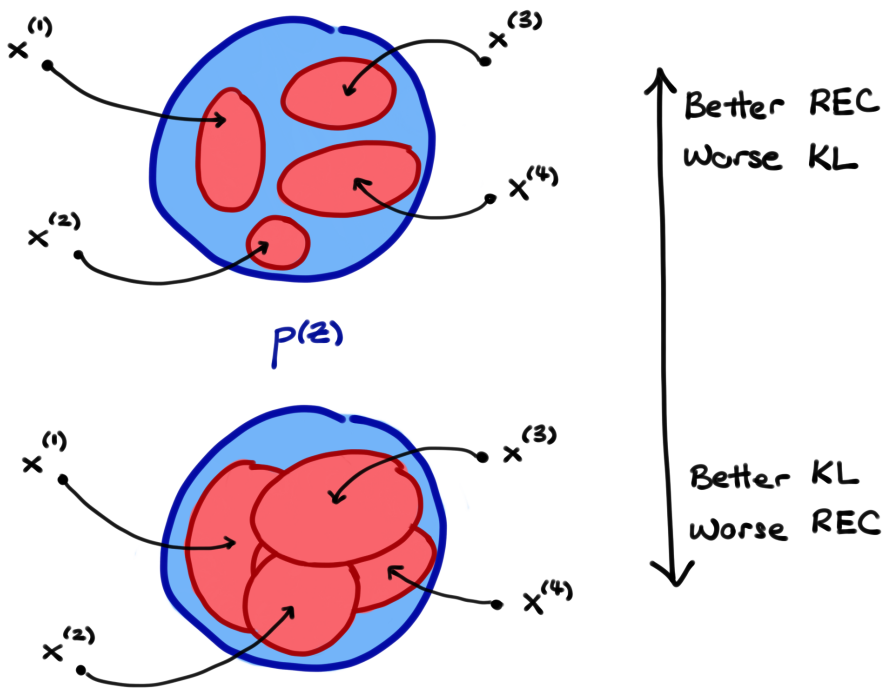
[2] Jordan et al. 1999; Blei et al. 2017; Wainwright and Jordan 2007

# Visual of ELBO Trade-offs

Expanding joint term reveals[3]:

$$\text{negative-ELBO} = \mathcal{F}_{\theta,\phi}(\mathrm{x}) = \mathbb{E}_{q_\phi(\mathrm{z}|\mathrm{x})}\left[\underbrace{-\log p_\theta(\mathrm{x} \mid \mathrm{z})}_{\text{reconstruction error}} \underbrace{-\log p(\mathrm{z}) + \log q_\phi(\mathrm{z} \mid \mathrm{x})}_{\text{regularization terms}}\right]$$

$$= \mathbb{E}_{q_\phi(\mathrm{z}|\mathrm{x})}\left[-\log p_\theta(\mathrm{x} \mid \mathrm{z})\right] + KL\left(\log q_\phi(\mathrm{z} \mid \mathrm{x}) \,||\, p(\mathrm{z})\right)$$

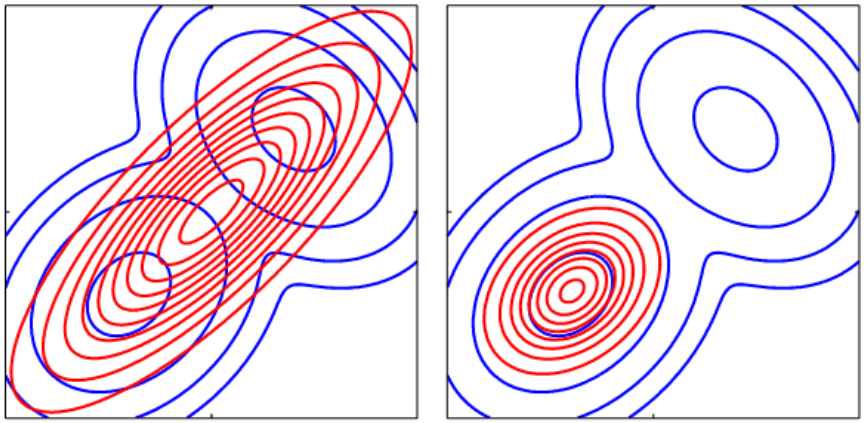



[3]Figure from http://ruishu.io/2018/03/14/vae/

# VAE vs GAN

**GANs and VAEs minimize KL in opposite directions**

- VAE: $min_\theta KL(Q \,||\, P_\theta)$ tends to covering regions with small values of $p_{data}$
- GAN: $min_\theta KL(P_\theta \,||\, Q)$ tends to missing mode
- Very different behavior!



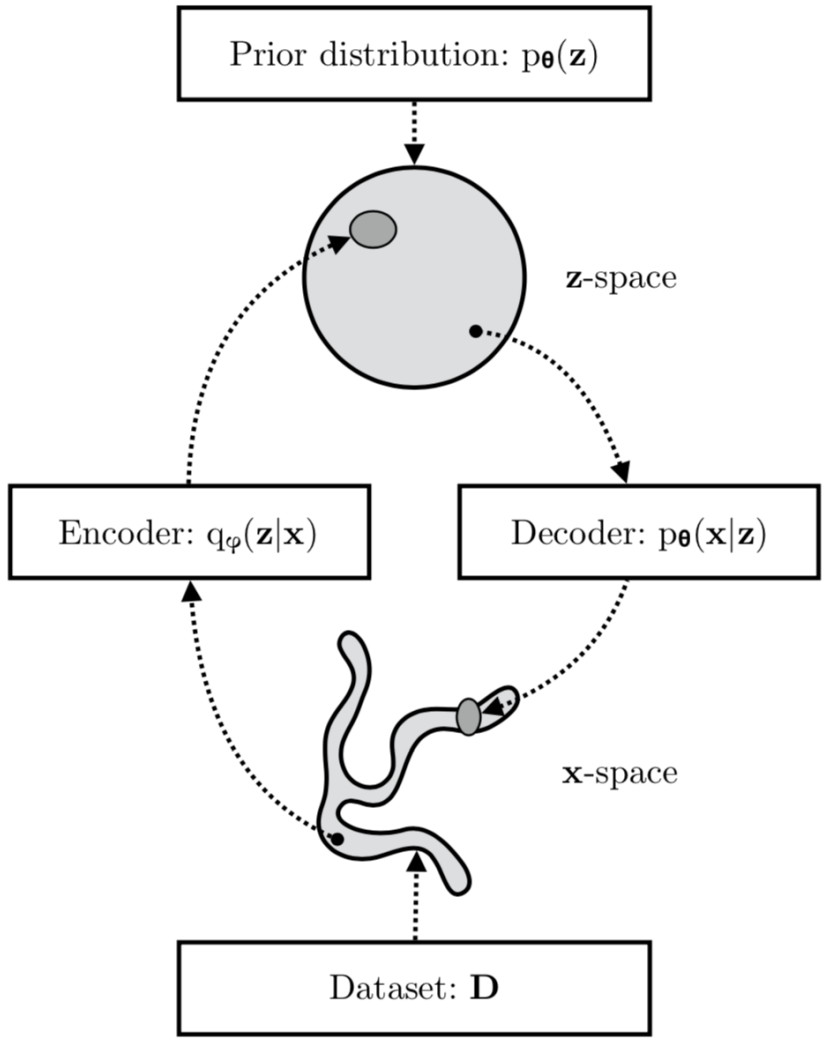
Mode covering Mode missing

| | GANs (InfoGAN) | VAEs |
|---|---|---|
| KLD to minimize | $\min_\theta \text{KL}\,(p_\theta(\boldsymbol{x}\|y) \,\|\|\, q^r(\boldsymbol{x}\|\boldsymbol{z}, y)) \sim \min_\theta \text{KL}(P_\theta \,\|\|\, Q)$ | $\min_\theta \text{KL}(q_\eta(\boldsymbol{z}\|\boldsymbol{x}, y) q^r_*(y\|\boldsymbol{x}) \,\|\|\, p_\theta(\boldsymbol{z}, y\|\boldsymbol{x})) \sim \min_\theta \text{KL}(Q \,\|\|\, P_\theta)$ |

Hu et al. (2018), "On Unifying Deep Generative Models"

# Problems with VAE

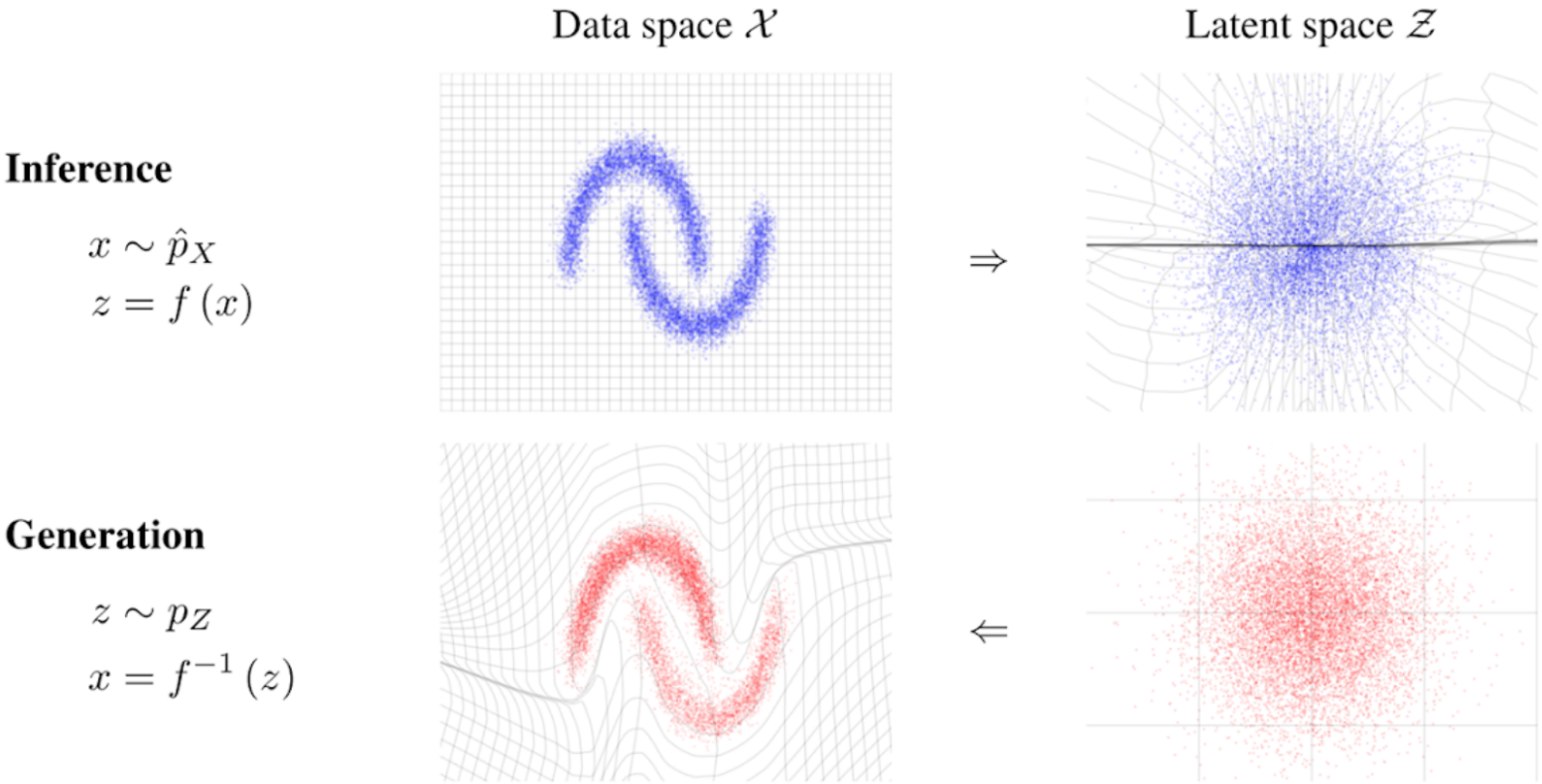
- Factorized Gaussian as posterior $\implies$ constrains flexibility.
- Larger encoder, decoder doesn't eliminate bottleneck.
- No exact density estimation.




**How to create a more flexible posterior?**

Rezende and Mohamed 2015; Kingma et al. 2016; Miller et al. 2017; Chen et al. 2017; Huang et al. 2018b; Tomczak and Welling 2018; Casale et al. 2018; van den Berg et al. 2018

# A Solution: Normalizing Flows




# Normalizing Flows

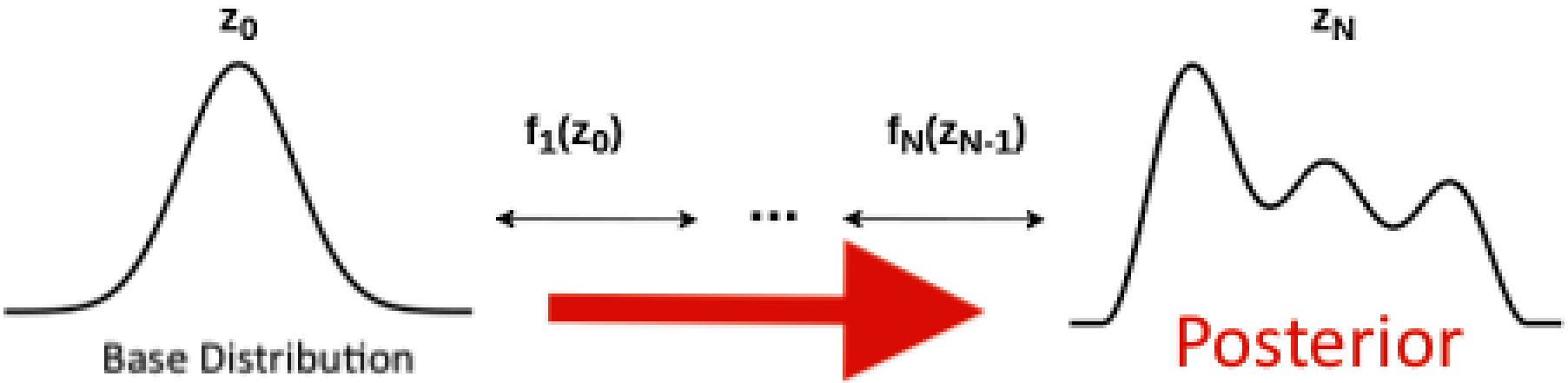
Translating probability distributions

Tabak and Vanden-Eijnden 2010; Tabak and Turner 2013; Dinh et al. 2017; Rezende and Mohamed 2015

# A Solution: Normalizing Flows

**Idea:** Map a complex distribution to a simple

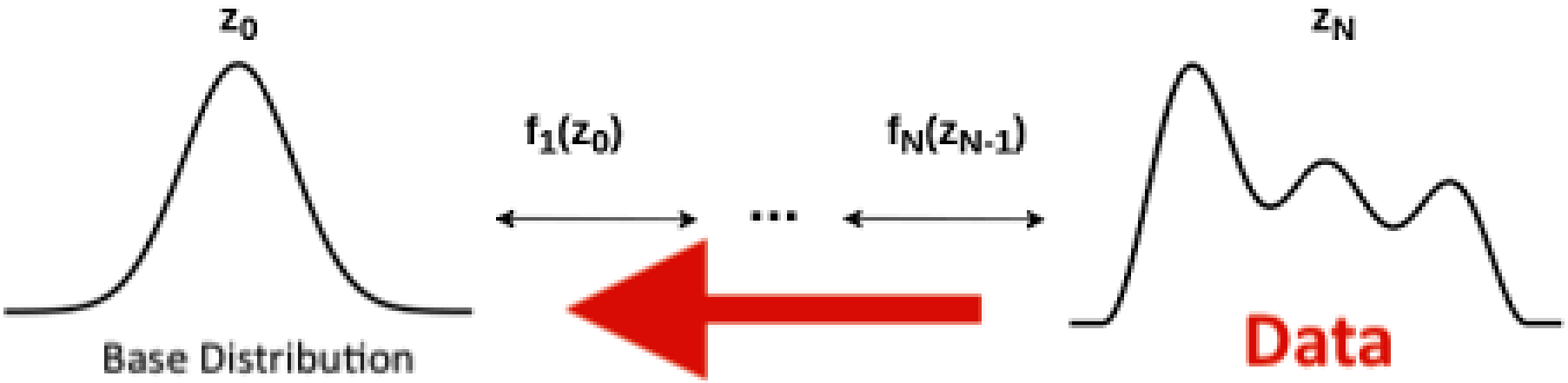
- **Approximate Inference** $KL(q_\phi(z \mid x) \,||\, p(z \mid x))$
-
-
-



# A Solution: Normalizing Flows

**Idea:** Map a complex distribution to a simple

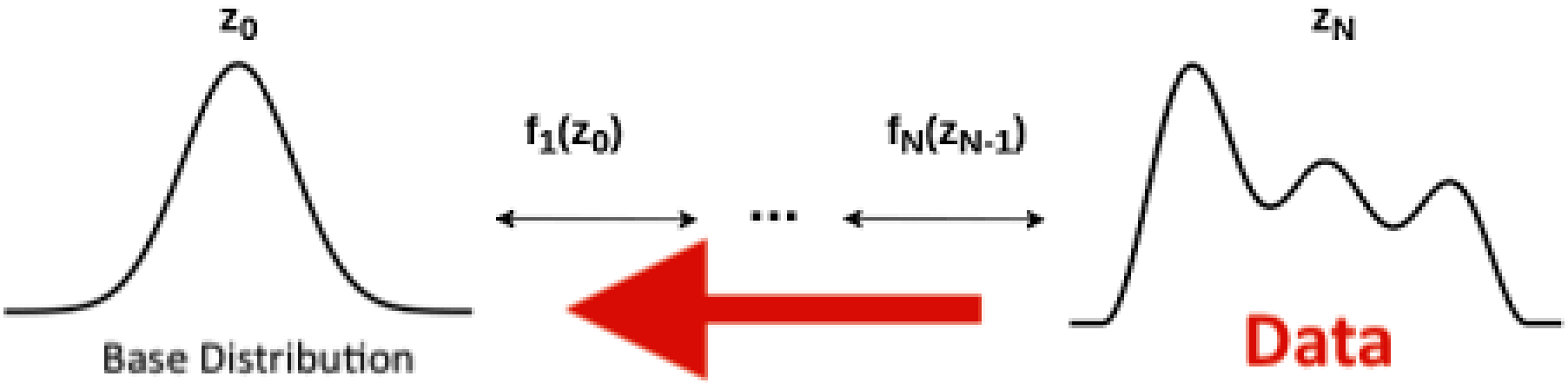
- Approximate Inference $KL(q_\phi(z \mid x) \,||\, p(z \mid x))$
- **Density estimation** $p(x)$
-
-



# A Solution: Normalizing Flows

**Idea:** Map a complex distribution to a simple

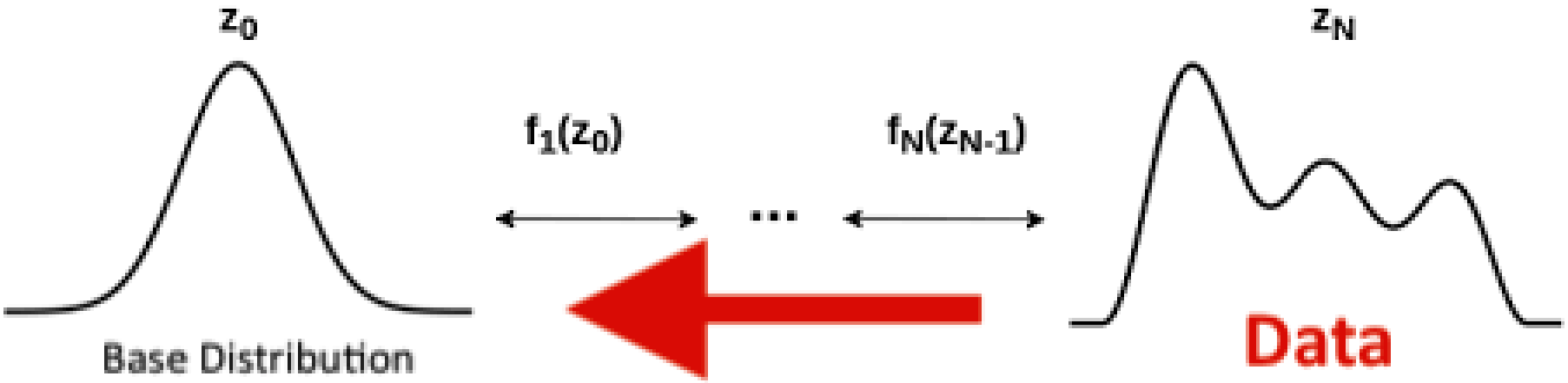
- Approximate Inference $KL(q_\phi(z \mid x) \,||\, p(z \mid x))$
- Density estimation $p(x)$
- **Data Synthesis** Transform noise by $f^{-1}$
-



# A Solution: Normalizing Flows

**Idea:** Map a complex distribution to a simple

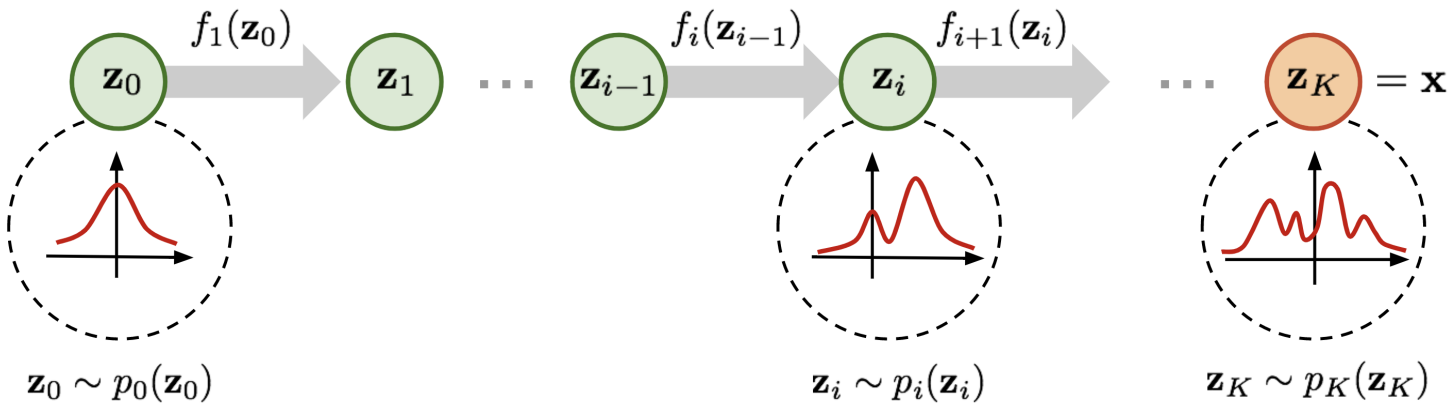
- Approximate Inference $KL(q_\phi(z \mid x) \,||\, p(z \mid x))$
- Density estimation $p(x)$
- Data Synthesis Transform noise by $f^{-1}$
- **Out of Distribution Detection** Evaluate $p(x^*)$



# Flexible Normalizing Flows

Transform density through a sequence of $K$ invertible transforms $f_k(\cdot)$:

$$z_K = f_K \circ \cdots \circ f_2 \circ f_1(z_0)$$




Since $f_k$ is an invertible, smooth mapping $\implies$ computable density:

$$q(z') = q(z) \left| \det \frac{\partial f_k^{-1}}{\partial z'} \right| = q(z) \left| \det \frac{\partial f_k}{\partial z} \right|^{-1} \tag{2}$$

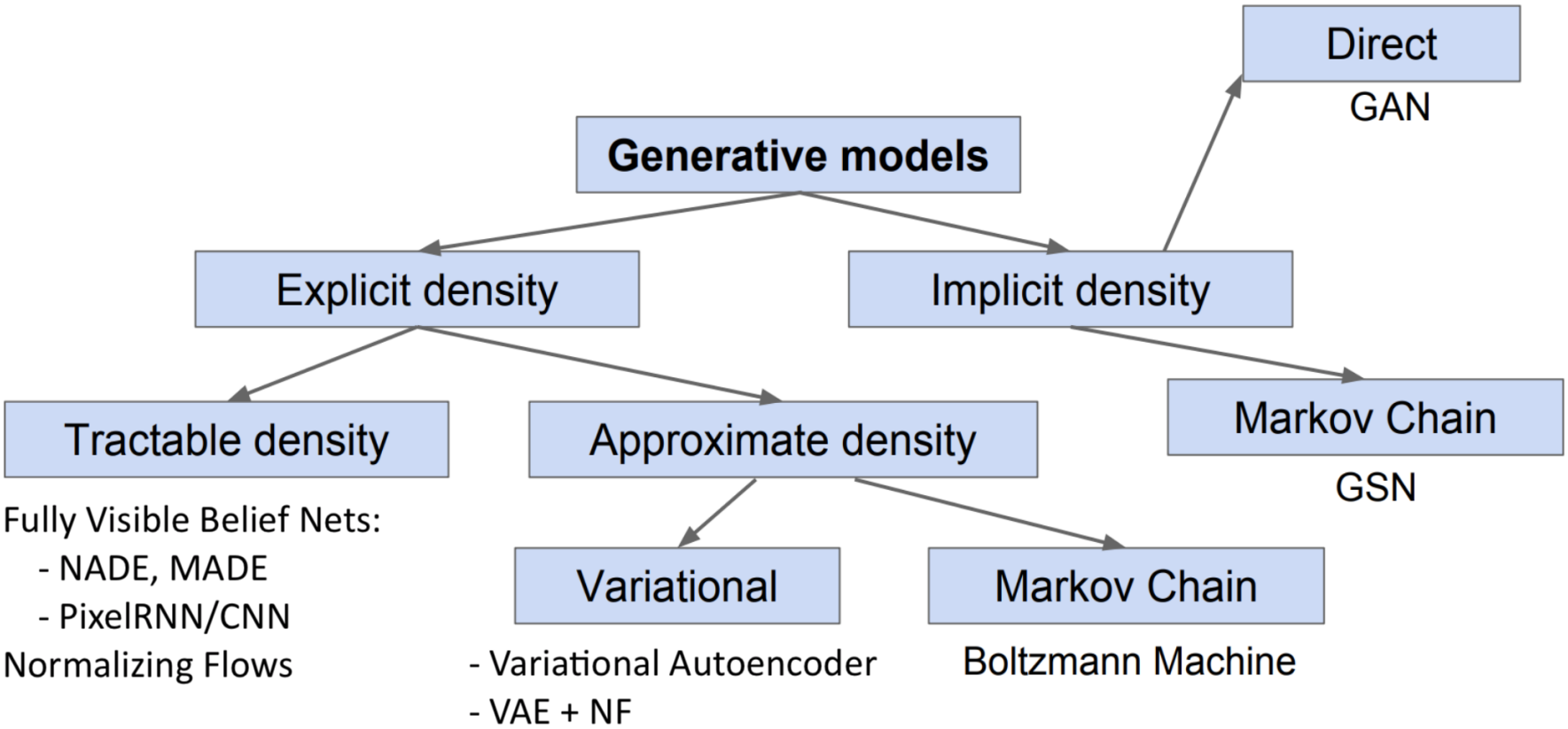
Tabak and Vanden-Eijnden 2010; Tabak and Turner 2013; Rezende and Mohamed 2015

# VAE Objective with Flow Posterior

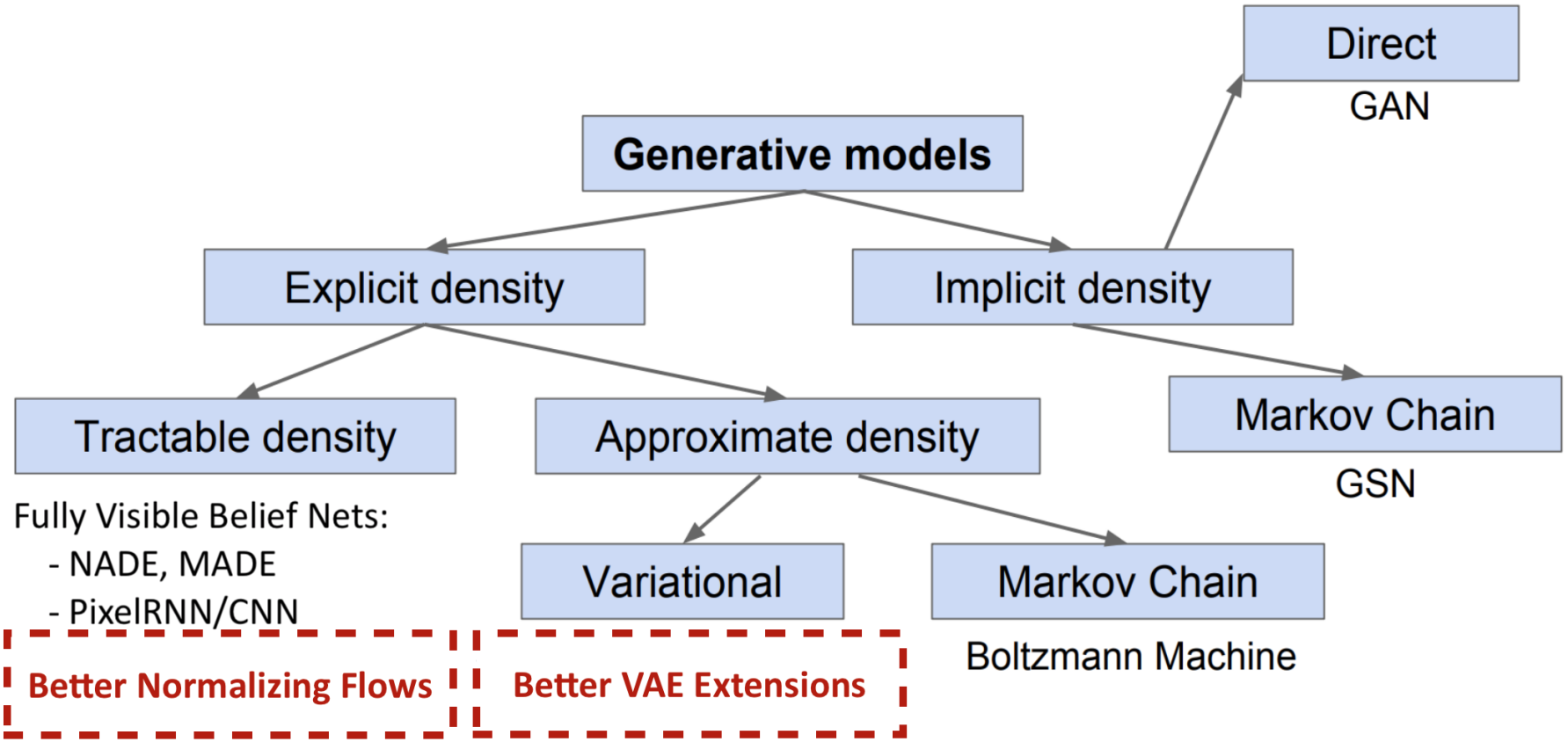
$$\begin{aligned}\mathcal{F}_{\theta,\phi}(\mathsf{x}) &= \mathbb{E}_{q_\phi}\left[-\log p_\theta(\mathsf{x}, \mathsf{z}_K) + \log q_\phi(\mathsf{z}_K \mid \mathsf{x})\right] \\ &= \mathbb{E}_{q_0}\left[-\log p_\theta(\mathsf{x}, \mathsf{z}_K) + \log q_0(\mathsf{z}_0 \mid \mathsf{x}) - \sum_{k=1}^{K} \log \left|\det \frac{\partial f_k}{\partial \mathsf{z}_{k-1}}\right|\right] , \qquad (3)\end{aligned}$$

- where $q_0(\mathsf{z}_0 \mid \mathsf{x})$ is a known base distribution (e.g. standard normal).

# The Larger Context



# The Larger Context



# Outline

# Gradient Boosted Normalizing Flows

Robert Giaquinto and Arindam Banerjee

*Advances in Neural Information Processing Systems*, 2020

# Gradient Boosted Normalizing Flows: Overview

- Deep generative models with flexible latent representations
- *Wider* not deeper models
- Builds on recent work:

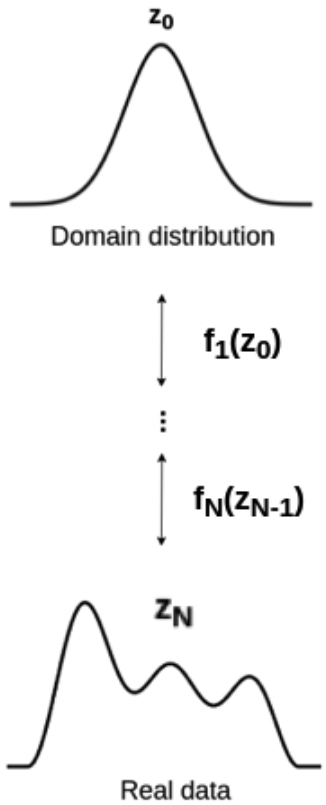



Figure: Normalizing flows

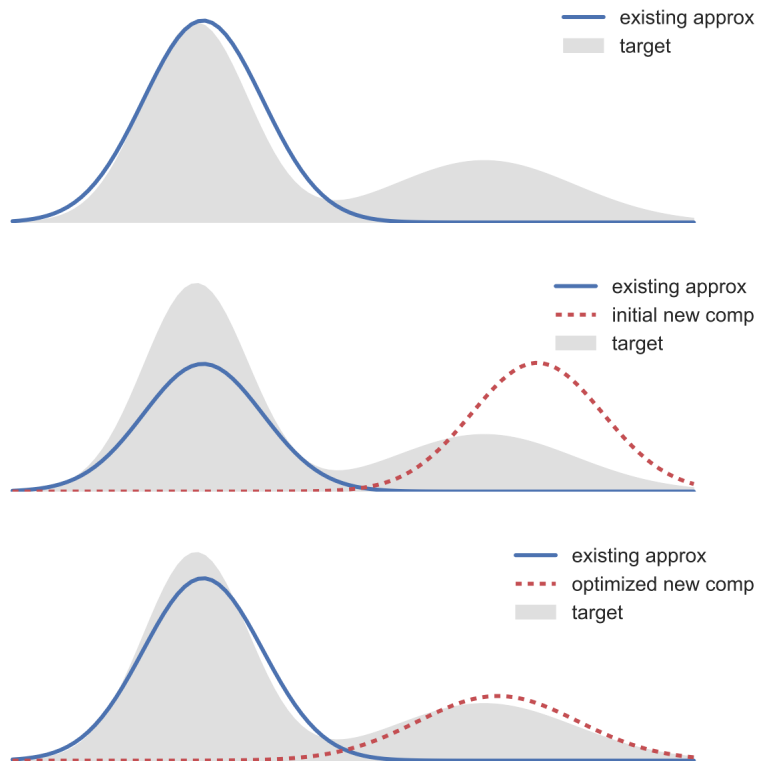
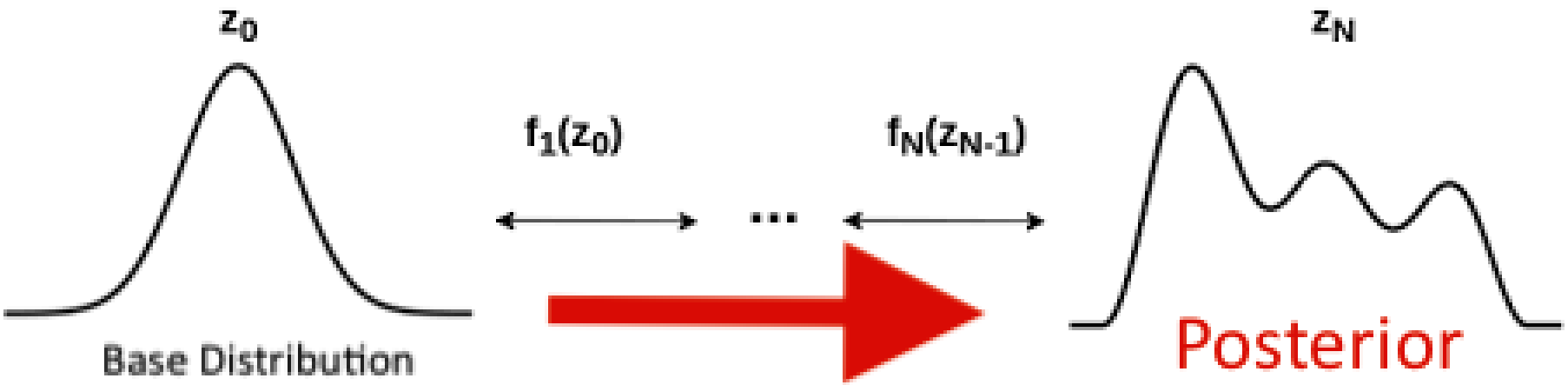



Figure: Unsupervised gradient boosting

Giaquinto and Banerjee (2020), "Gradient Boosted Normalizing Flows"

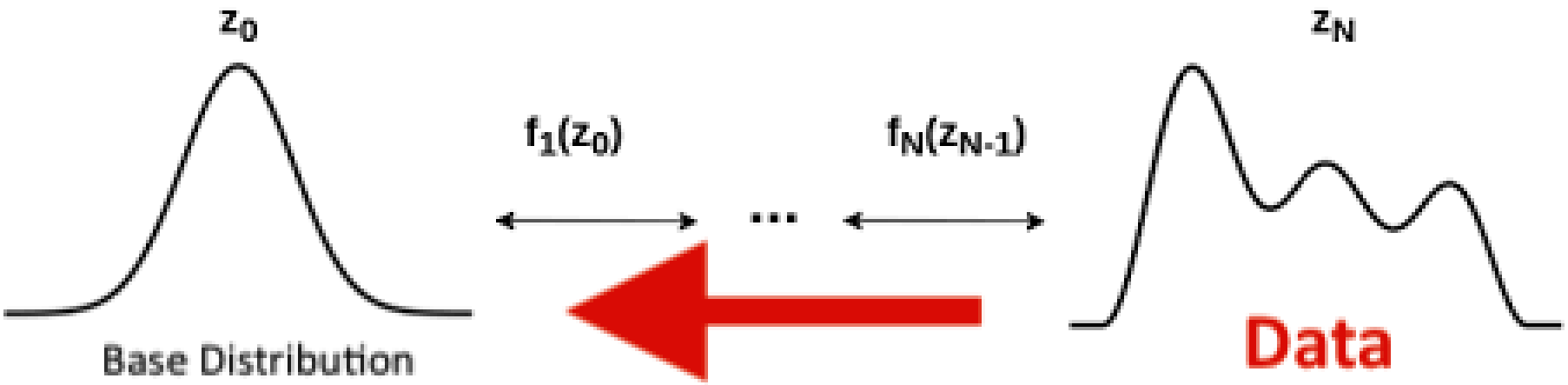
# Scope of Work on GBNF

- Approximate Inference $KL(q_\phi(z \mid x) \,||\, p(z \mid x))$
-



# Scope of Work on GBNF

- Approximate Inference $KL(q_\phi(z \mid x) \,||\, p(z \mid x))$
- Density estimation $p(x)$



# Gradient Boosted Normalizing Flows

- A *wider* alternative
- Iteratively add new flow components with gradient boosting
- New component $g_K^{(c)}$ fit to residuals of previous fixed components $G_K^{(c-1)}$

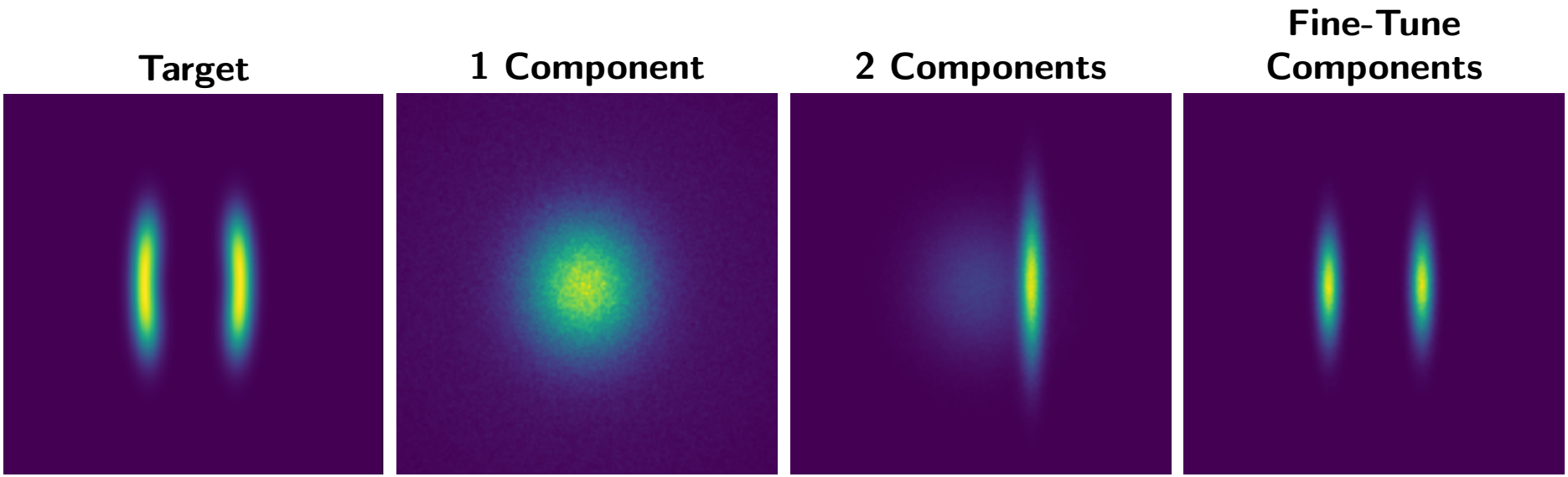



Figure: Simple affine flows trained with GBNF fit a multi-model density.

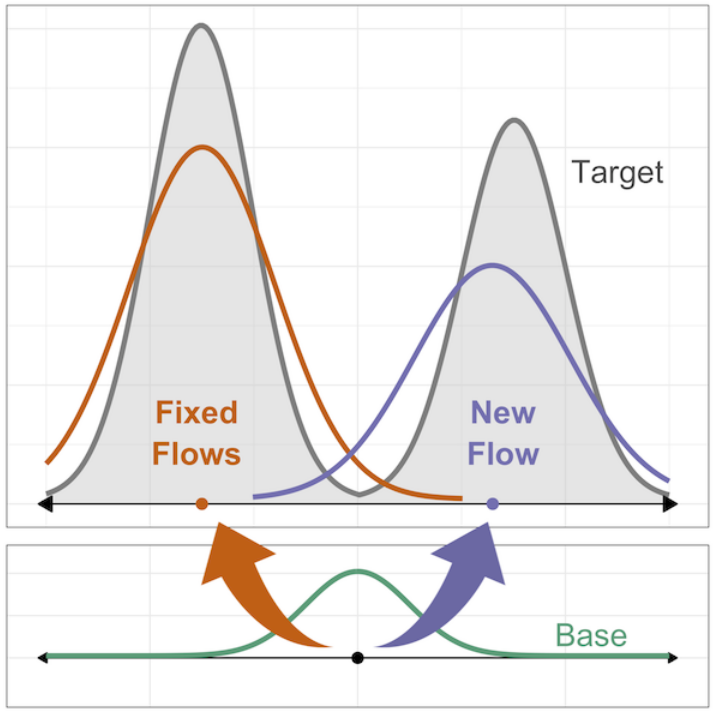
# Density Estimation with Multiplicative GBNF

Weighted combination of fixed and new components.

$$\log G_K^{(c)}(x) = \left(\log G_K^{(c-1)}(x) + \rho_c \log g_K^{(c)}(x)\right) - \log \Gamma_{(c)} \ .$$

- $c$ components in mixture
- $K$-step flows
- Assign components weight $\rho_c \in [0, 1]$
- $\Gamma_{(c)}$ partition function

- See paper for *additive* mixture formulation



# Training GBNF

Maximum likelihood:

$$\text{Loss: } \mathcal{F} = -\frac{1}{N}\sum_{i=1}^{N} \log G_K^{(c-1)}(x_i) + \rho_c \log g_K^{(c)}(x_i) - \log \Gamma_{(c)}$$

Component c=1 : Fit with traditional objective — no boosting!

Component $c > 1$: Have fixed components $G_K^{(c-1)}$, and new component $g_K^{(c)}$

1. Train $g_K^{(c)}$ to convergence
2. Optimize weight $\rho_c \in [0, 1]$

**How to fit new component?**

# Optimizing a New Component

- How do we improve model?

$$\underset{g_K \in \mathcal{G}_K}{\arg\max} \; KL(p^* \| \mathrm{G}_K^{(c-1)}) - KL(p^* \| \mathrm{G}_K^{(c)})$$

# Optimizing a New Component

- How do we improve model?

$$\underset{g_K \in \mathcal{G}_K}{\arg\max} \; KL(p^* \| \mathrm{G}_K^{(c-1)}) - KL(p^* \| \mathrm{G}_K^{(c)})$$

- Fit $g_K^{(c)}$ based on *functional* gradient descent, yields lower bound of:

$$g_K^{(c)} = \underset{g_K \in \mathcal{G}_K}{\arg\max} \; \mathbb{E}_{p^*} \left[\log g_K(\mathrm{x})\right] - \log \mathbb{E}_{G_K^{(c-1)}} \left[g_K(\mathrm{x})\right]$$

- Solution given by:

$$g_K^{(c)}(\mathrm{x}) = \frac{p^*(\mathrm{x})}{G_K^{(c-1)}(\mathrm{x})}$$

# Optimizing a New Component

- How do we improve model?

$$\underset{g_K \in \mathcal{G}_K}{\arg\max}\ KL(p^* \| \mathrm{G}_K^{(c-1)}) - KL(p^* \| \mathrm{G}_K^{(c)})$$

- Fit $g_K^{(c)}$ based on *functional* gradient descent, yields lower bound of:

$$g_K^{(c)} = \underset{g_K \in \mathcal{G}_K}{\arg\max}\ \mathbb{E}_{p^*}\left[\log g_K(\mathrm{x})\right] - \log \mathbb{E}_{G_K^{(c-1)}}\left[g_K(\mathrm{x})\right]$$

- Solution given by:

$$g_K^{(c)}(\mathrm{x}) = \frac{p^*(\mathrm{x})}{G_K^{(c-1)}(\mathrm{x})}$$

- How will new $g_K^{(c)}$ behave?
  - Large $G_K^{(c-1)} \implies$ ignore, already covered
  - Small $G_K^{(c-1)} \implies$ attractive!

# Experiments

**Density Estimation:**

- Toy Density estimation: Given data, model the likelihood
- Density estimation on tabular datasets

**Variational Inference:**

- Toy Density matching: Match analytic energy function, model generates data
- VAE + normalizing flows: model images, variational inference

# Density Estimation

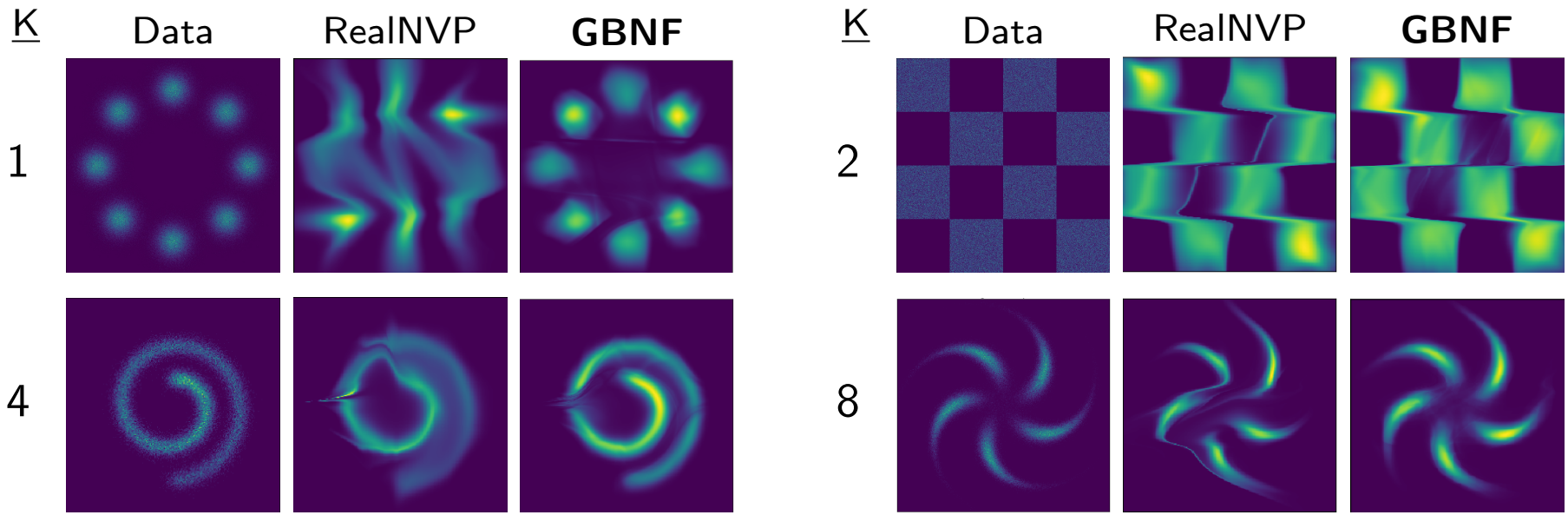



Figure: Density estimation for 2D toy data. The **GBNF** columns shows results for a gradient boosted model where each component is a RealNVP flow with $K = 1, 2, 4$ or 8 flow steps, respectively. **RealNVP** is equivalent to GBNF's first component.

# Density Estimation Animation

https://github.com/robert-giaquinto/gradient-boosted-normalizing-flows

# Density Estimation on Real Data

| **Model** | POWER↑ $d$=6;$n$=2,049,280 | GAS↑ $d$=8;$n$=1,052,065 | HEPMASS↑ $d$=21;$n$=525,123 | MINIBOONE↑ $d$=43;$n$=36,488 | BSDS300↑ $d$=63;$n$=1,300,000 |
|---|---|---|---|---|---|
| RealNVP | $0.17_{\pm .01}$ | $8.33_{\pm .14}$ | $-18.71_{\pm .02}$ | $-13.55_{\pm .49}$ | $153.28_{\pm 1.78}$ |
| **Boosted RealNVP** | $0.27_{\pm 0.01}$ | $9.58_{\pm .04}$ | $-18.60_{\pm 0.06}$ | $-10.69_{\pm 0.07}$ | $154.23_{\pm 2.21}$ |
| Glow | $0.17_{\pm .01}$ | $8.15_{\pm .40}$ | $-18.92_{\pm .08}$ | $-11.35_{\pm .07}$ | $155.07_{\pm .03}$ |
| **Boosted Glow** | $0.24_{\pm 0.01}$ | $9.95_{\pm 0.11}$ | $-17.81_{\pm 0.12}$ | $-10.76_{\pm 0.02}$ | $154.68_{\pm 0.34}$ |

Table: Log-likelihood on the test set (higher is better) for 4 datasets from UCI machine learning (Dua and Taniskidou, 2017) and BSDS300 (Martin et al., 2001). GBNF models include $c = 4$ components. Mean/stdev are estimated over 3 runs.

# Generative Models of Images

Experimental setup:

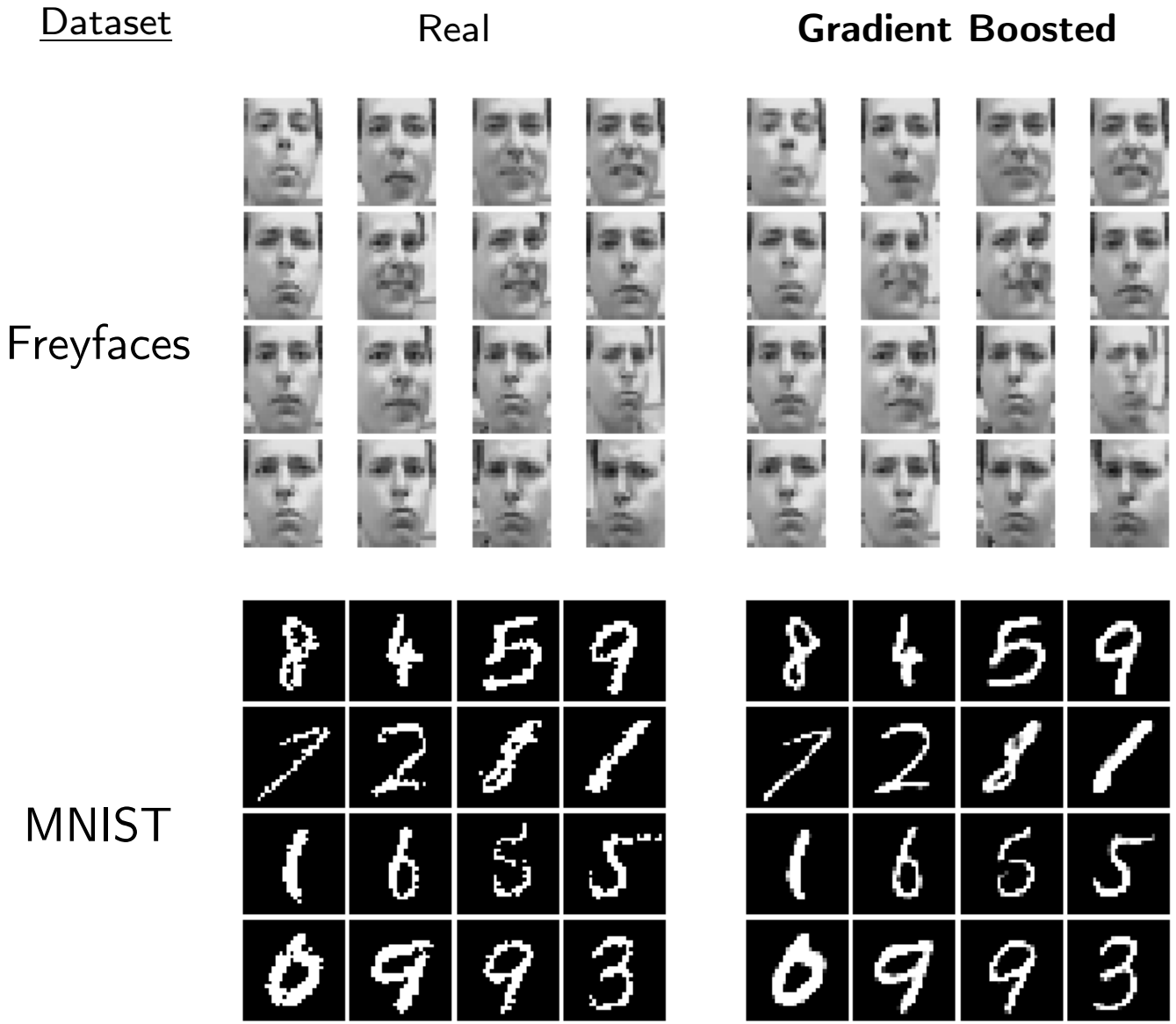
- VAE with various normalizing flows
- Only boosting RealNVP
- Dataset training sizes:
  - Freyfaces: 1,500
  - Caltech 101 Silhouettes: 4,100
  - Omniglot: 23,000
  - static MNIST 50,000

# Generative Models of Images

| **Model** | **MNIST** ↓ | **Freyfaces** ↓ | **Omniglot** ↓ | **Caltech 101** ↓ |
|---|---|---|---|---|
| VAE | $84.97_{\pm 0.01}$ | $4.78_{\pm 0.07}$ | $103.16_{\pm 0.01}$ | $108.43_{\pm 1.81}$ |
| Planar | $83.16_{\pm 0.07}$ | $\mathbf{4.60}_{\pm 0.04}$ | $\mathbf{100.18}_{\pm 0.01}$ | $\mathbf{104.23}_{\pm 1.60}$ |
| Sylvester | $\mathbf{81.99}_{\pm 0.02}$ | $\mathbf{4.49}_{\pm 0.03}$ | $\mathbf{98.54}_{\pm 0.29}$ | $\mathbf{100.38}_{\pm 1.20}$ |
| IAF | $\mathbf{83.14}_{\pm 0.06}$ | $4.70_{\pm 0.05}$ | $100.97_{\pm 0.07}$ | $108.41_{\pm 1.31}$ |
| RealNVP | $83.36_{\pm 0.09}$ | $4.62_{\pm 0.16}$ | $100.43_{\pm 0.19}$ | $113.00_{\pm 1.70}$ |
| GBNF | $\mathbf{82.59}_{\pm 0.03}$ | $\mathbf{4.41}_{\pm 0.01}$ | $\mathbf{99.09}_{\pm 0.17}$ | $\mathbf{106.40}_{\pm 0.54}$ |

Table: Variational inference. Negative log-likelihood for image data.

# Image Reconstructions



# Advantages of Gradient Boosted Normalizing Flows

- Compliments existing flow models[4]
- Resembles mixture
  - Easier! Just optimize $g_K^{(c)}$
- Exchange flexibility for training cycles
- Prediction/sampling in parallel

[4] Note: Variational inference requires *analytically* invertible flows (see paper).

# Outline

# Compressive Normalizing Flows

Robert Giaquinto and Arindam Banerjee

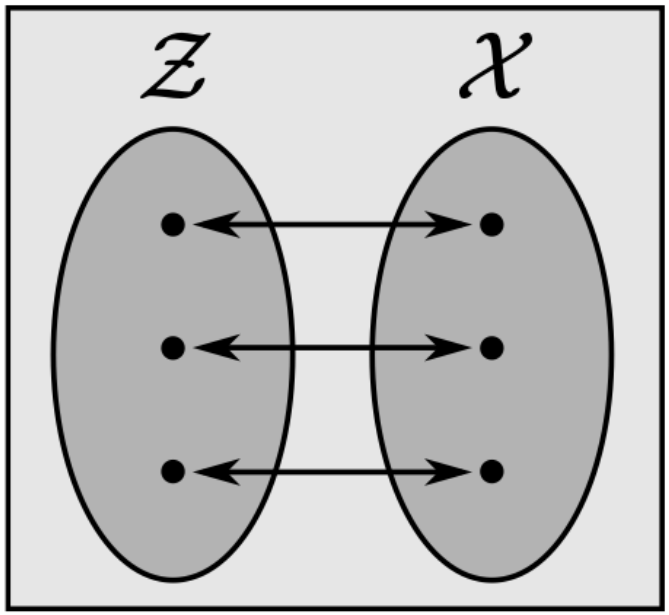
*Under Review*, 2021

# Motivation for Compressing

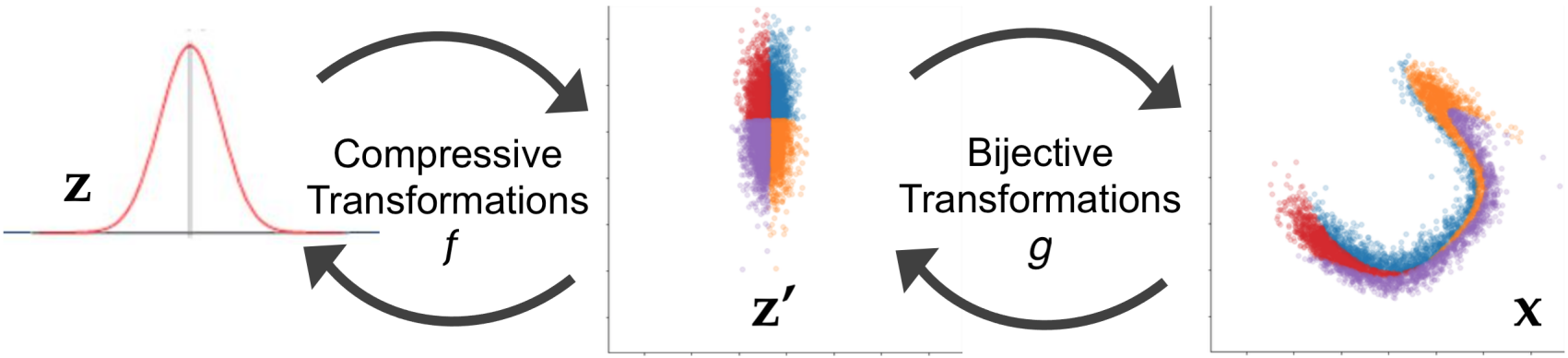
**Problem:** Flows are bijective. High-dimensional data are a challenge.

**Why Change Dimensions?**

- Architectural, memory, and computational costs.
- Discard noise, learn semantic information.
- Better downstream features



# Compressive Normalizing Flows




Flow transformations based on:

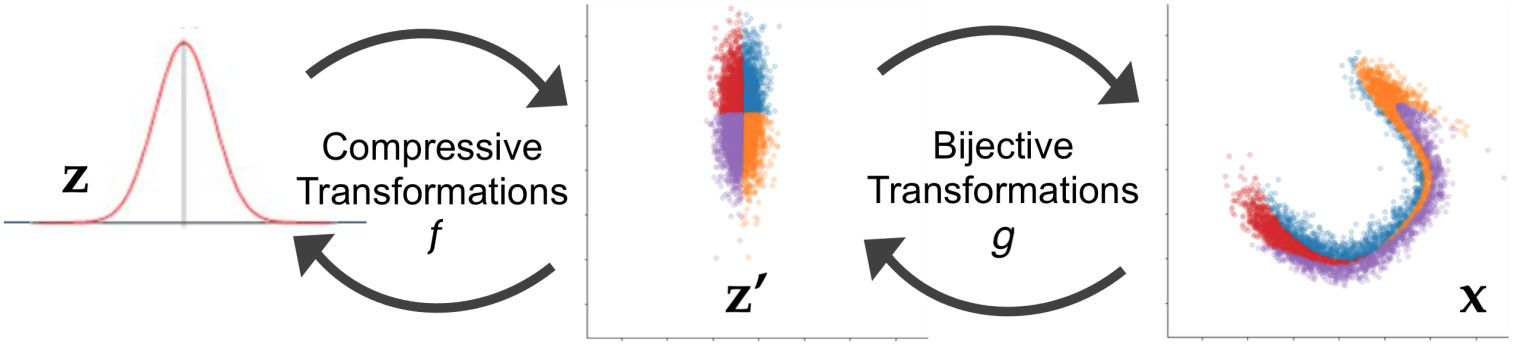
1. Probabilistic PCA[5]
2. Linear VAEs[6]
3. Deep VAEs[7]

[5] Tipping and Bishop 1999

[6] Lucas et al. 2019; Rolinek et al. 2019; Bao et al. 2020

[7] Kingma and Welling 2014; Rezende and Mohamed 2015

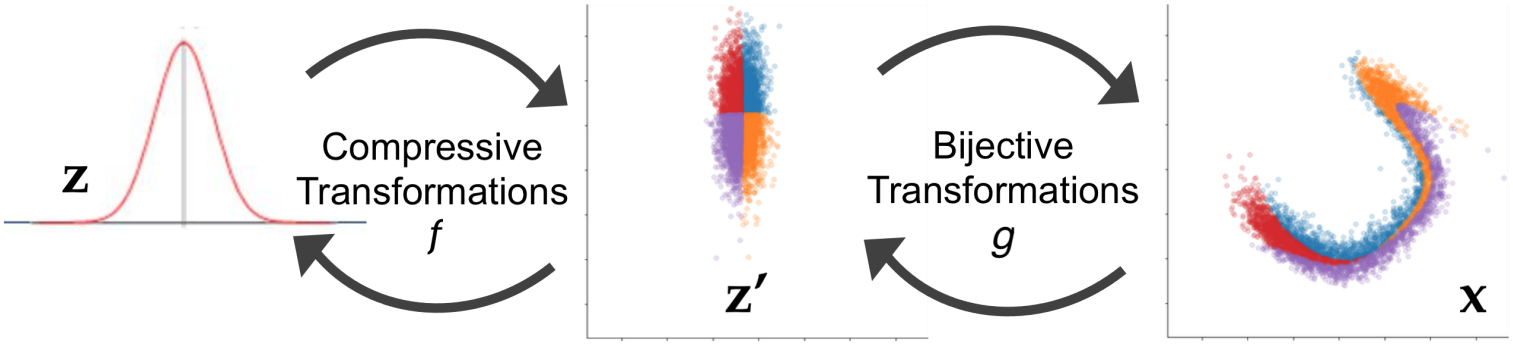
# PPCA-Base Flow




**PPCA**[8]**:** $p(z' \mid z) = \mathcal{N}(Az + \mu, \sigma^2 \mathbb{I}_D)$

[8] Tipping and Bishop 1999

# PPCA-Base Flow




**PPCA**[9]**:** $p(z' \mid z) = \mathcal{N}(Az + \mu, \sigma^2 \mathbb{I}_D)$

**PPCA-based Flow Objective:**

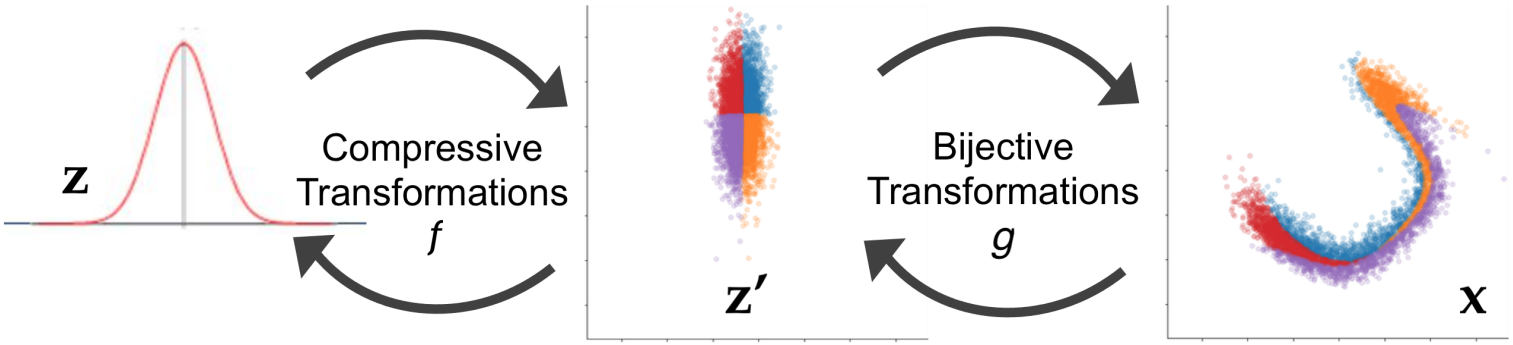
$$\log p(x) = \underbrace{\log p_{z'}(g_\beta^{-1}(x))}_{\text{High-Dim Base}} + \underbrace{\log |\det J_g|}_{\text{Bijective Flow Term}}$$

$$= -\log |C| - \frac{1}{2} \operatorname{Tr}(C^{-1} S) + \log |\det J_{g_\beta}|$$

- C is covariance matrix of marginal $z' \sim \mathcal{N}(\mu, C)$
- S sample covariance for $z'$
- PPCA $\implies$ posterior available analytically

[9] Tipping and Bishop 1999

# VAE-Base Flows




- Linear VAE: recovers MLE of PPCA[10]
- Deep VAE: maximizes a lower-bound[11].

**VAE-based Flow Objective:**

$$\log p(\mathrm{x}) = \underbrace{\log p_{Z'}(g_\beta^{-1}(\mathrm{x}))}_{\text{High-Dim Base}} + \underbrace{\log |\det \mathrm{J}_g|}_{\text{Bijective Flow Term}}$$

$$= \underbrace{\mathbb{E}_q\left[\log p(\mathrm{z}' \mid \mathrm{z})\right] - KL\left(q(\mathrm{z} \mid \mathrm{z}') \| p(\mathrm{z})\right)}_{\text{VAE Objective}} + \log |\det \mathrm{J}_{g_\beta}|$$

[10] Lucas et al. 2019; Rolinek et al. 2019; Bao et al. 2020

[11] Nielsen et al. 2020; Kingma and Welling 2014

# Deep VAE-based Compressive Flows

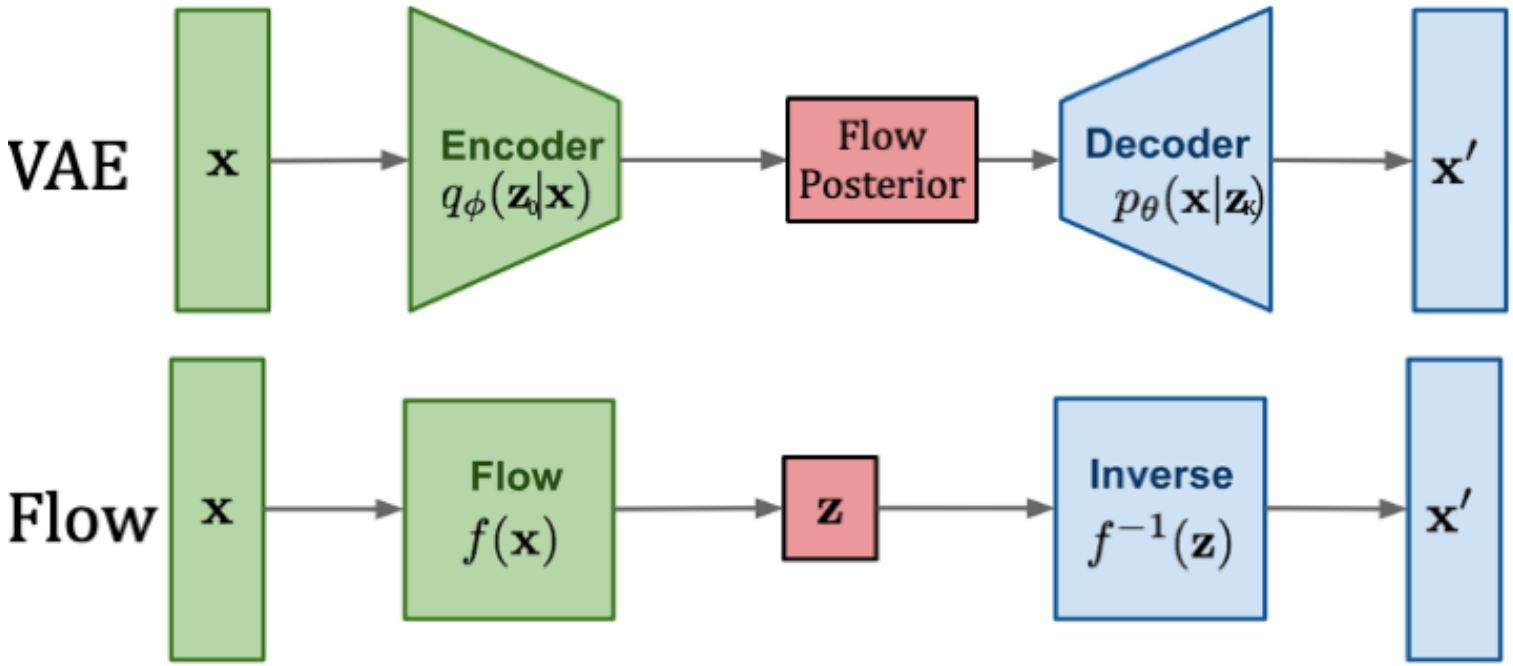
**Is it worth losing exact likelihood calculations?**

**Deep VAE Compression:**

- Powerful compression step
- Scalable optimization (amortized VI)
- Intermixing lower-bounds is not uncommon[12]

[12] Nielsen et al. 2020; Ho et al. 2019; Nielsen and Winther 2020

# Comparing to VAEs with Flow Posteriors




**Advantages of Flows:**

- Invertible mappings are parameter *efficient*
- Modular and composable
- Exact likelihoods (often)

---

**Adapted from** `lilianweng.github.io/lil-log/2018/10/13/flow-based-deep-generative-models`

# Architectures

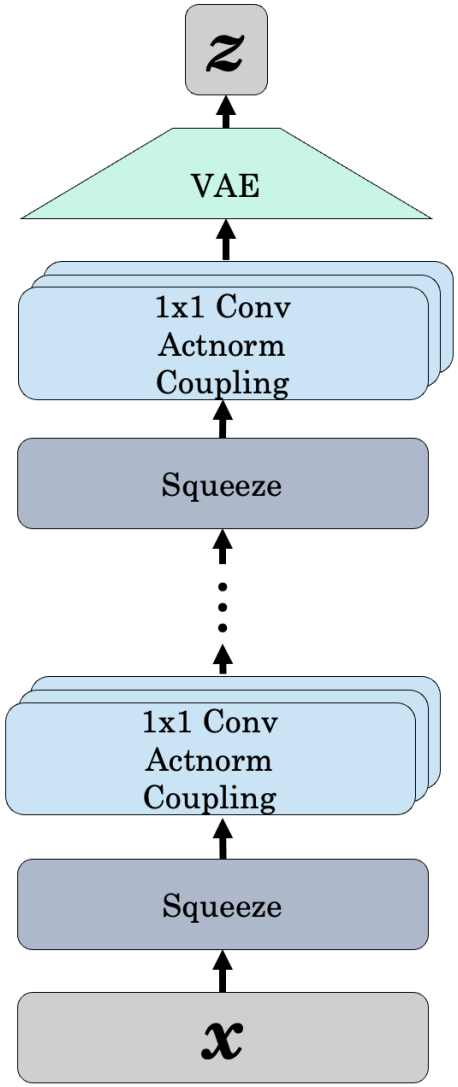



Figure: Compress Once

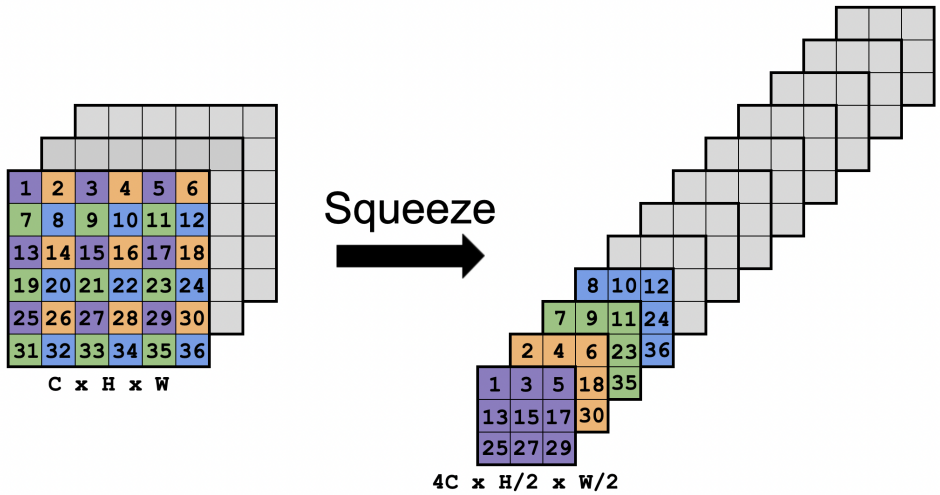



Figure: "Squeeze" operation

# Architectures

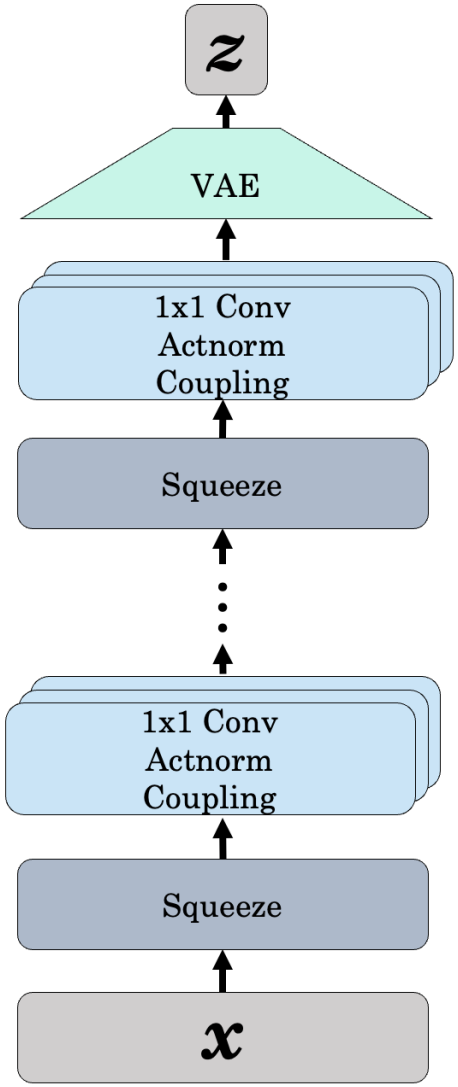



Figure: Compress Once

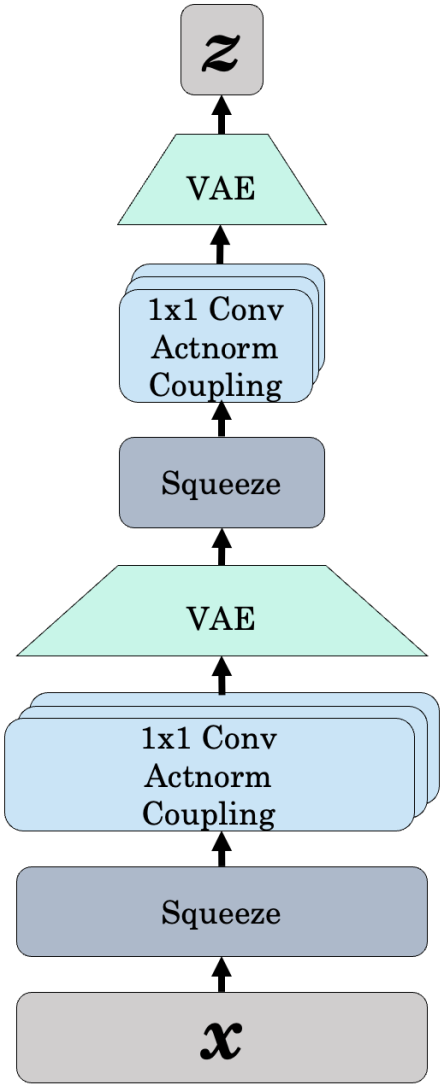
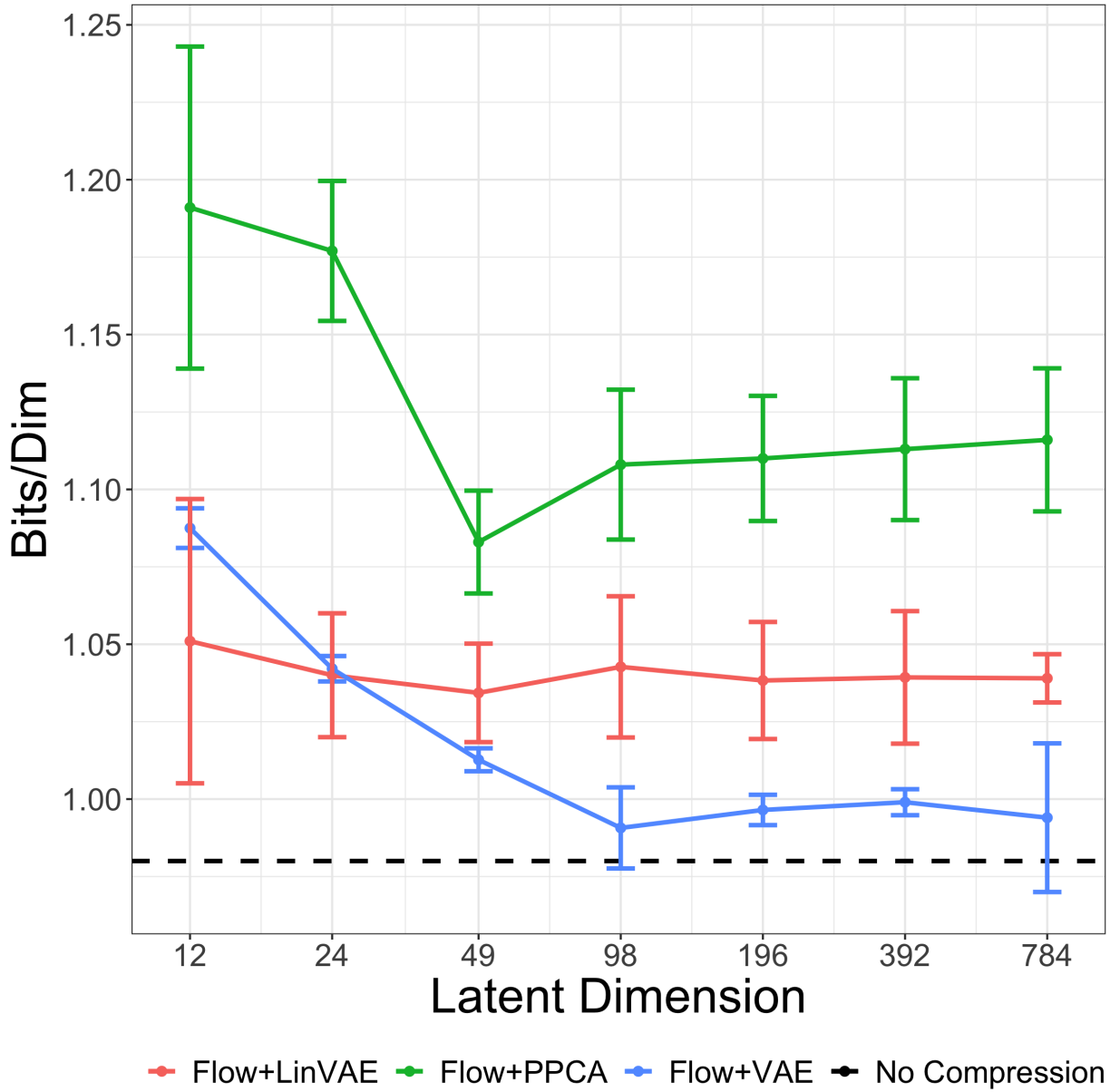



Figure: Intermix compressive and bijective

# Does Compression Hurt Performance?



# What is the best way to compress?

Table: Average bits/dim (lower is better) for compression models. Models substantially compress the latent space by ≈94%, but we only observe a a small drop in performance compared to a purely bijective flow (No compression).

| **Model** | **MNIST** ↓ | **CIFAR-10** ↓ |
|---|---|---|
| No compression | 0.98 | 3.27 |
| MaxPool Flow Nielsen et al. (2020) | 1.10 | 3.41 |
| PPCA Compression | 1.08 | 3.37 |
| Linear VAE Compression | 1.07 | 3.35 |
| Intermixed Compression | 1.04 | 3.35 |
| VAE Compression | **1.01** | **3.30** |

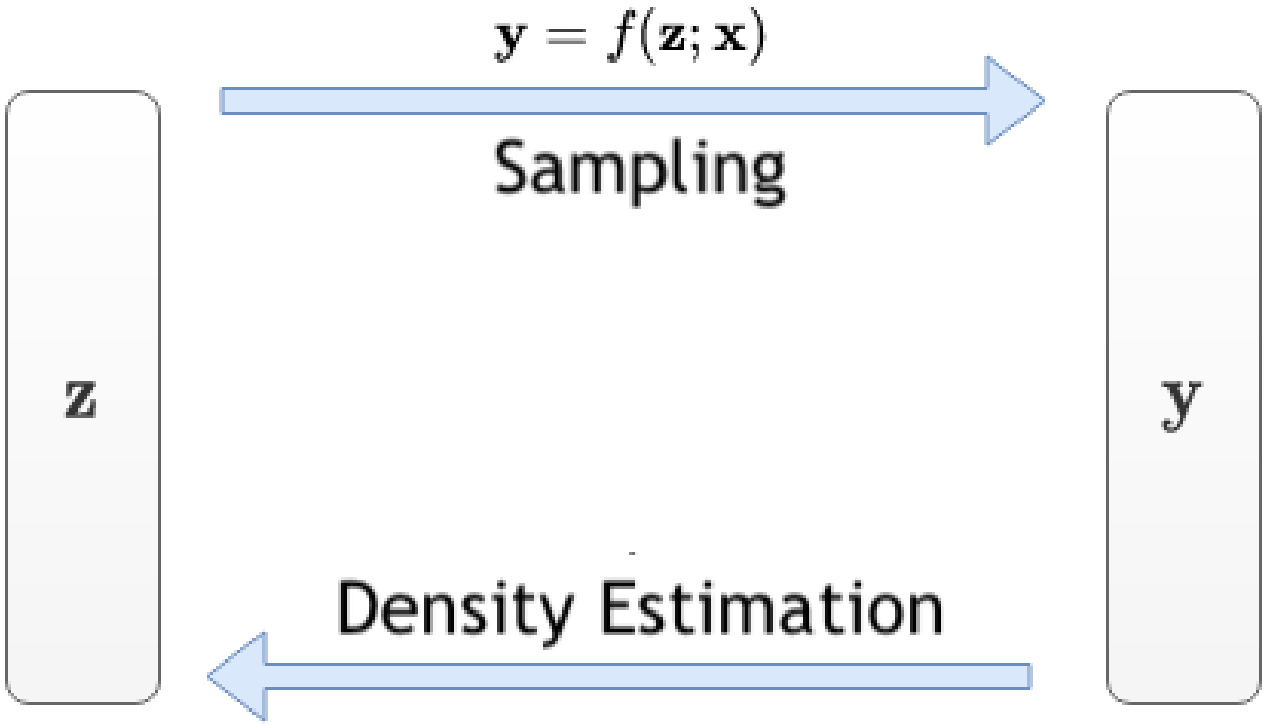
# Outline

# Probabilistic Super-Resolution with Normalizing Flows

Robert Giaquinto and Arindam Banerjee

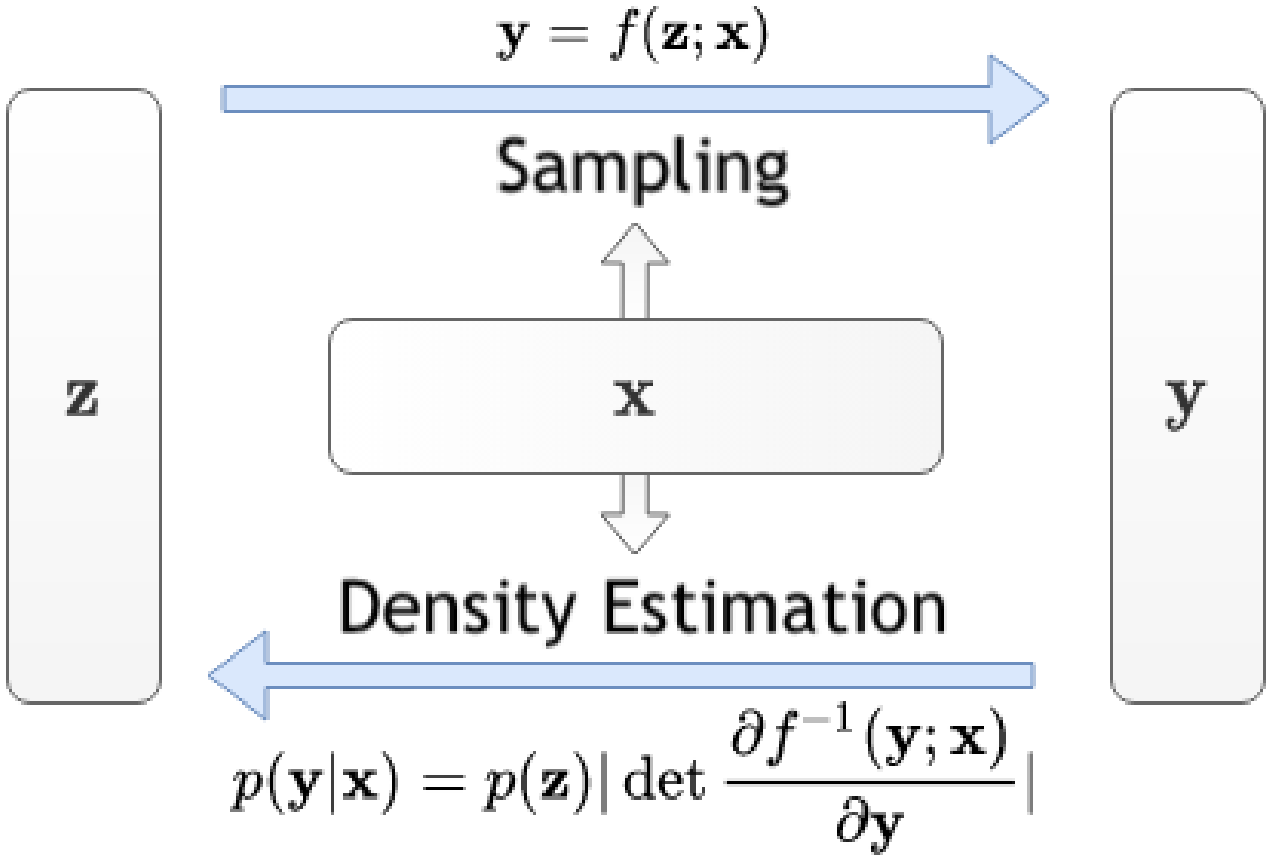
*Under Review*, 2021

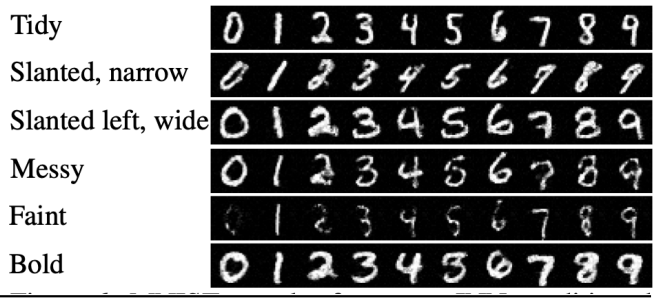
# Conditional Normalizing Flows

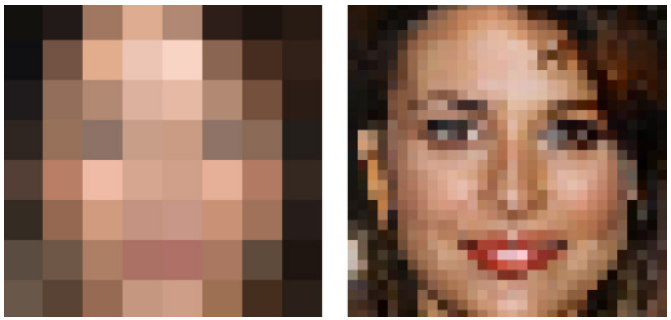



Winkler et al. 2019; Ardizzone et al. 2019; Lugmayr et al. 2020

# Conditional Normalizing Flows



Winkler et al. 2019; Ardizzone et al. 2019; Lugmayr et al. 2020

# Why Super-Resolution (SR) with Normalizing Flows?

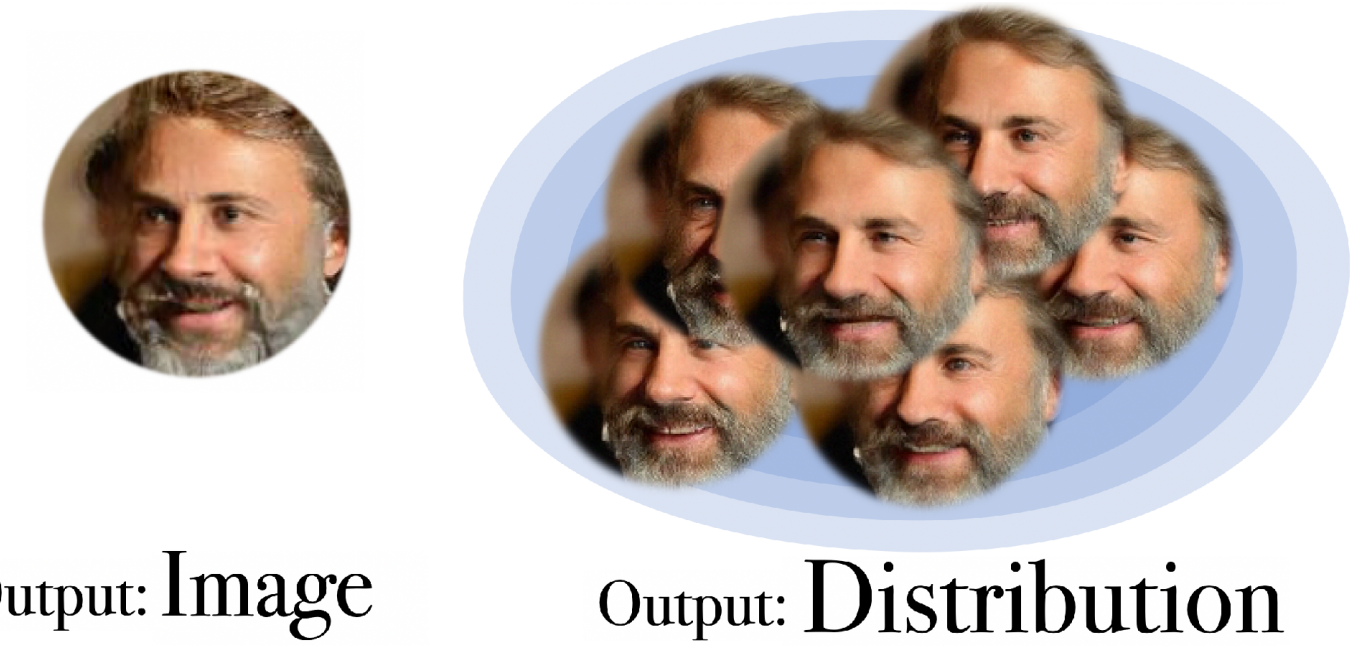



Figure: Traditional SR is ill posed[13]. Goal: learn a distribution over the outputs.

- Normalizing flows excel with high-dimensional data

[13] Lugmayr et al. 2020

# Our Focus

1. Challenges with flows:
   - Information bottleneck[14]
   - Bias towards learning local information[15]

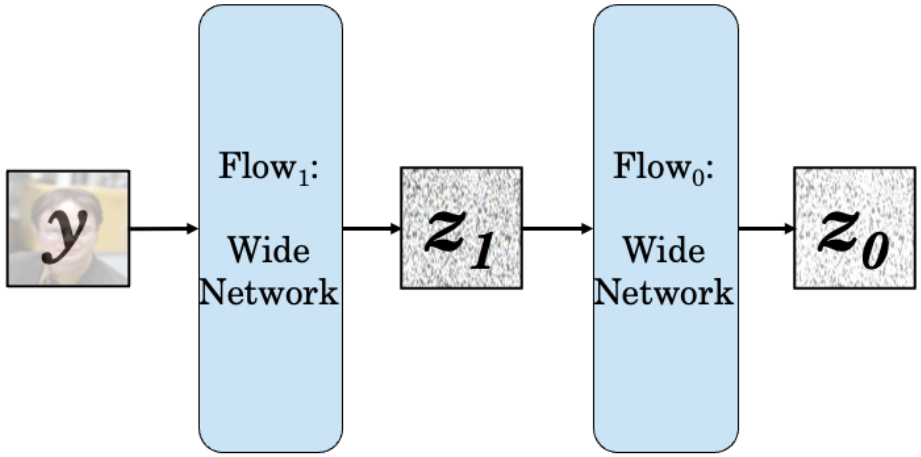



Figure: Information bottleneck from dimension constraints.

2. Broader applications:
   - $\mathrm{x} \in \mathbb{R}^d, \mathrm{y} \in \mathbb{R}^D$, where $d < D$
   - $x = \pi(y)$ for some (unknown) function $\pi : \mathbb{R}^D \mapsto \mathbb{R}^d$

3. Learn $p(\mathrm{y} \mid \mathrm{x})$ and $p(\mathrm{x}, \mathrm{y})$

[13] Chen et al. 2020; Huang et al. 2020

[14] Kirichenko et al. 2020

# Probabilistic Super-Resolution with Normalizing Flows

Learn joint distribution over high- and low-dimensional samples

$$p_{(X,Y)}(\mathrm{y},\mathrm{x};\phi,\theta) = \underbrace{p_{Y|X}(\mathrm{y}\mid\mathrm{x},\phi)}_{\text{Super-Resolution Conditional}} \cdot \underbrace{p_X(\mathrm{x};\theta)}_{\text{Low-Resolution Marginal}}$$

$$= p_Z\left(f_\phi^{-1}(\mathrm{y};\mathrm{x})\right)\left|\det\frac{\partial f_\phi}{\partial \mathrm{y}}\right| \cdot p_\epsilon\left(g_\theta^{-1}(\mathrm{x})\right)\left|\det\frac{\partial g_\theta}{\partial \mathrm{x}}\right|$$

- Two base-distributions control output

# Probabilistic Super-Resolution with Normalizing Flows

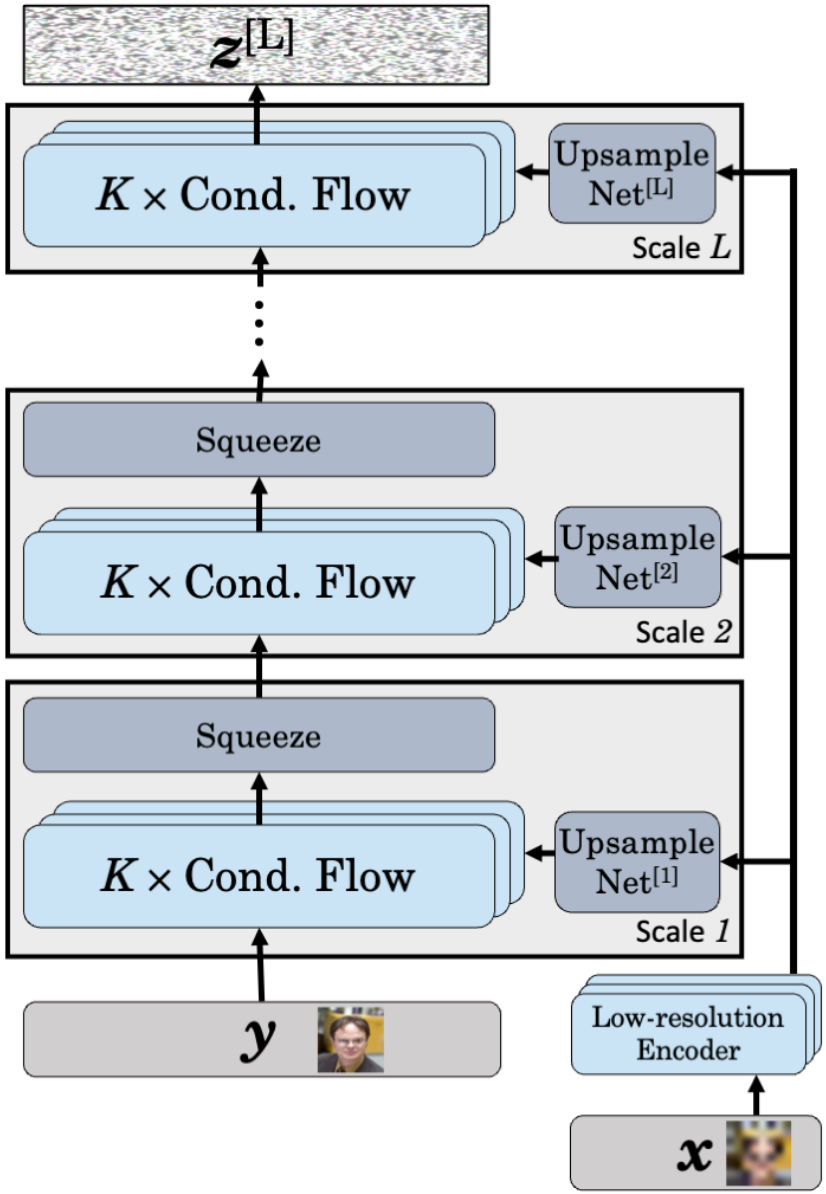
Learn joint distribution over high- and low-dimensional samples

$$p_{(X,Y)}(\mathrm{y},\mathrm{x};\boldsymbol{\phi},\boldsymbol{\theta}) = \underbrace{p_{Y|X}(\mathrm{y}\mid\mathrm{x},\boldsymbol{\phi})}_{\text{Super-Resolution Conditional}} \cdot \underbrace{p_X(\mathrm{x};\boldsymbol{\theta})}_{\text{Low-Resolution Marginal}}$$
$$= p_Z\left(f_{\boldsymbol{\phi}}^{-1}(\mathrm{y};\mathrm{x})\right)\left|\det\frac{\partial f_{\boldsymbol{\phi}}}{\partial \mathrm{y}}\right| \cdot p_{\boldsymbol{\epsilon}}\left(g_{\boldsymbol{\theta}}^{-1}(\mathrm{x})\right)\left|\det\frac{\partial g_{\boldsymbol{\theta}}}{\partial \mathrm{x}}\right|$$

- Two base-distributions control output
- Generative process:

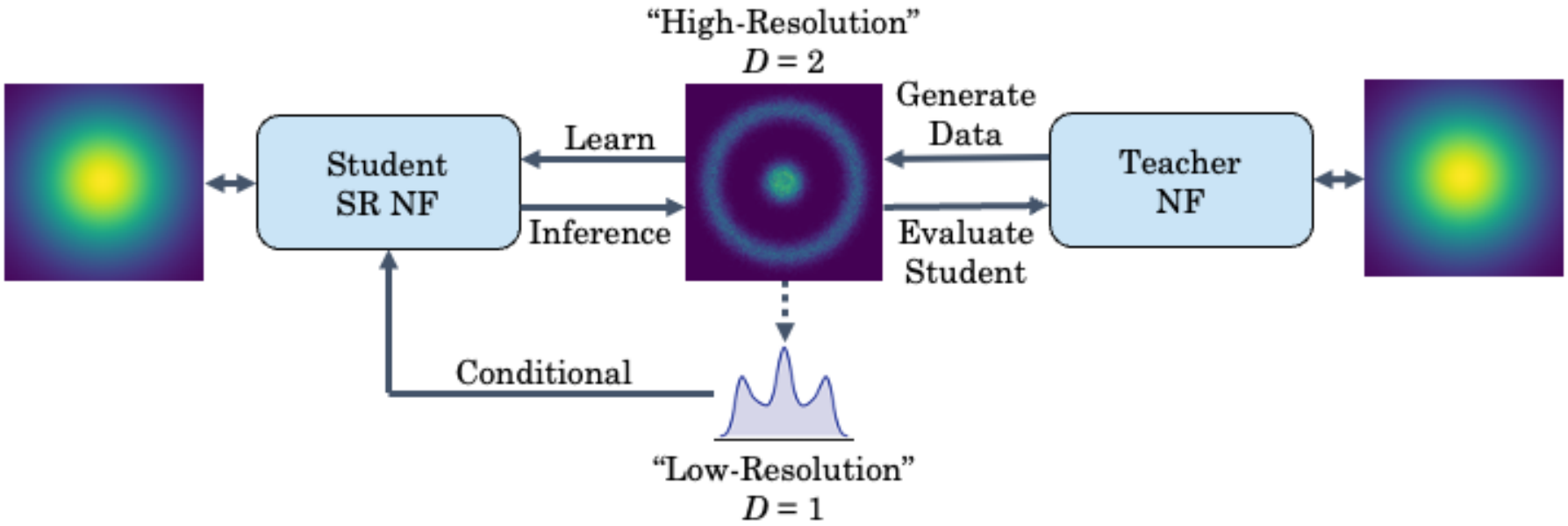
$$\begin{aligned} \boldsymbol{\epsilon} &\sim \mathcal{N}(0,\mathbb{I}_d)\ , & \mathrm{z} &\sim \mathcal{N}(0,\mathbb{I}_D)\ , \\ \mathrm{x} &= g_{\boldsymbol{\theta}}(\boldsymbol{\epsilon})\ , & \mathrm{y} &= f_{\boldsymbol{\phi}}\left(\mathrm{z};g_{\boldsymbol{\theta}}(\boldsymbol{\epsilon})\right)\ . \end{aligned} \tag{4}$$

# Advantages of PSR Flows

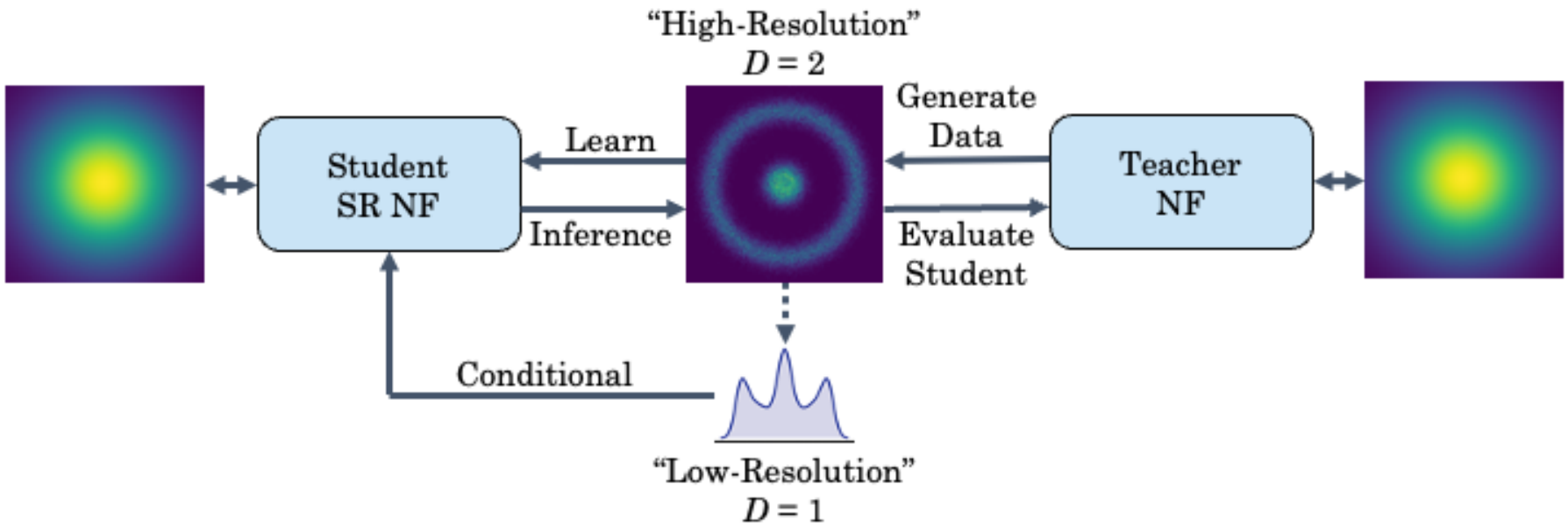



1. Conditional guides semantics
2. "Residual connections" help bottleneck
3. Applicable to any dataset
4. Joint model $\Longrightarrow$ more applications

# Student-Teacher Evaluation



# Student-Teacher Evaluation

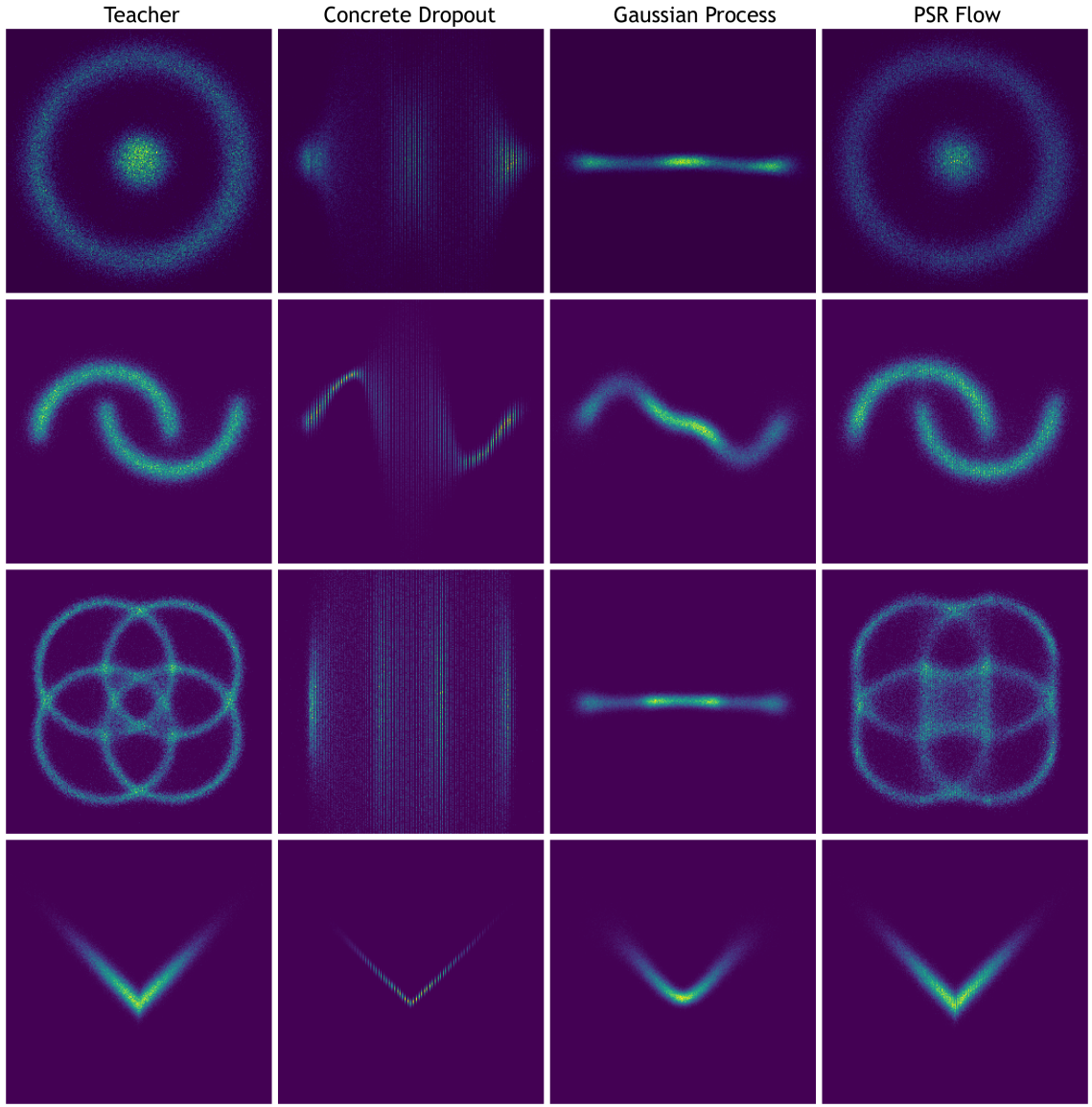



| Student | Circle & Cluster | Two Moons | Four Circles | Abs |
|---|---|---|---|---|
| Dropout[16] | 1.59 | 0.46 | 1.40 | 0.05 |
| Gaussian Process | 1.58 | 0.41 | 1.38 | 0.05 |
| PSR Flow | **0.05** | **0.02** | **0.05** | **0.01** |

Table: Earth Mover's Distance (lower is better) between teacher and student models.

[16] Gal et al. (2017), "Concrete Dropout"

# Learning a Distribution Over Samples



# Super-Resolution Faces

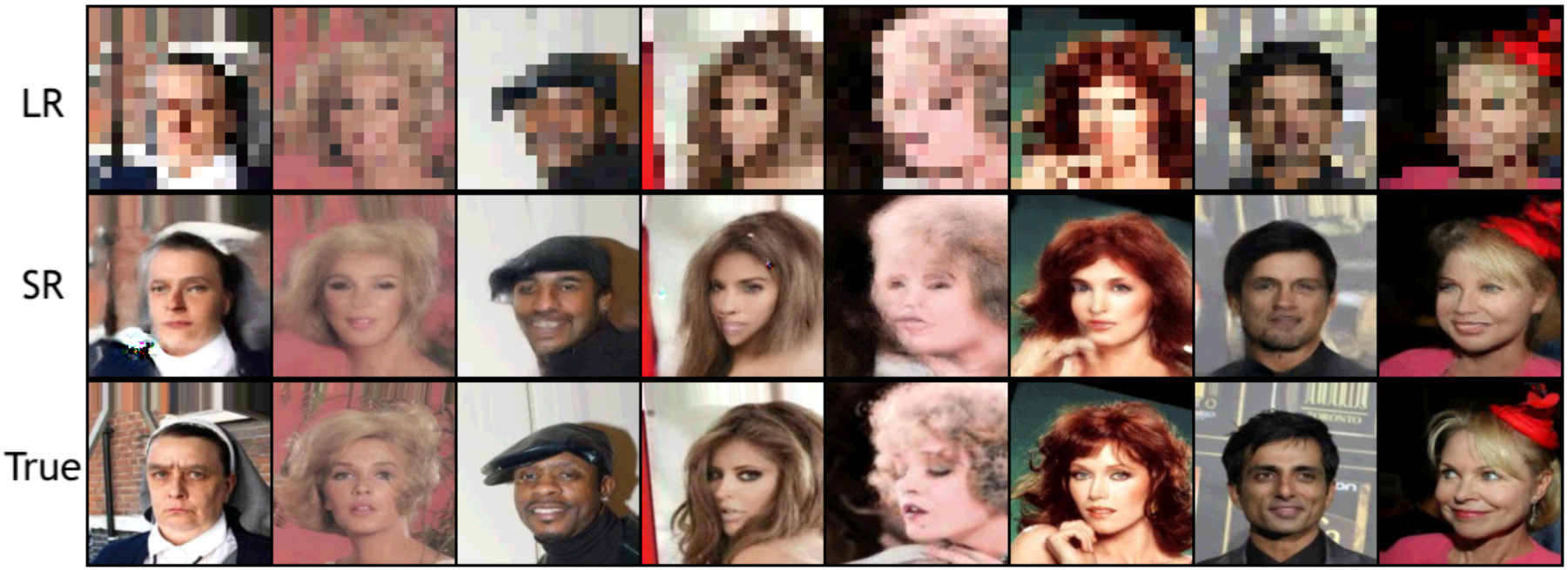
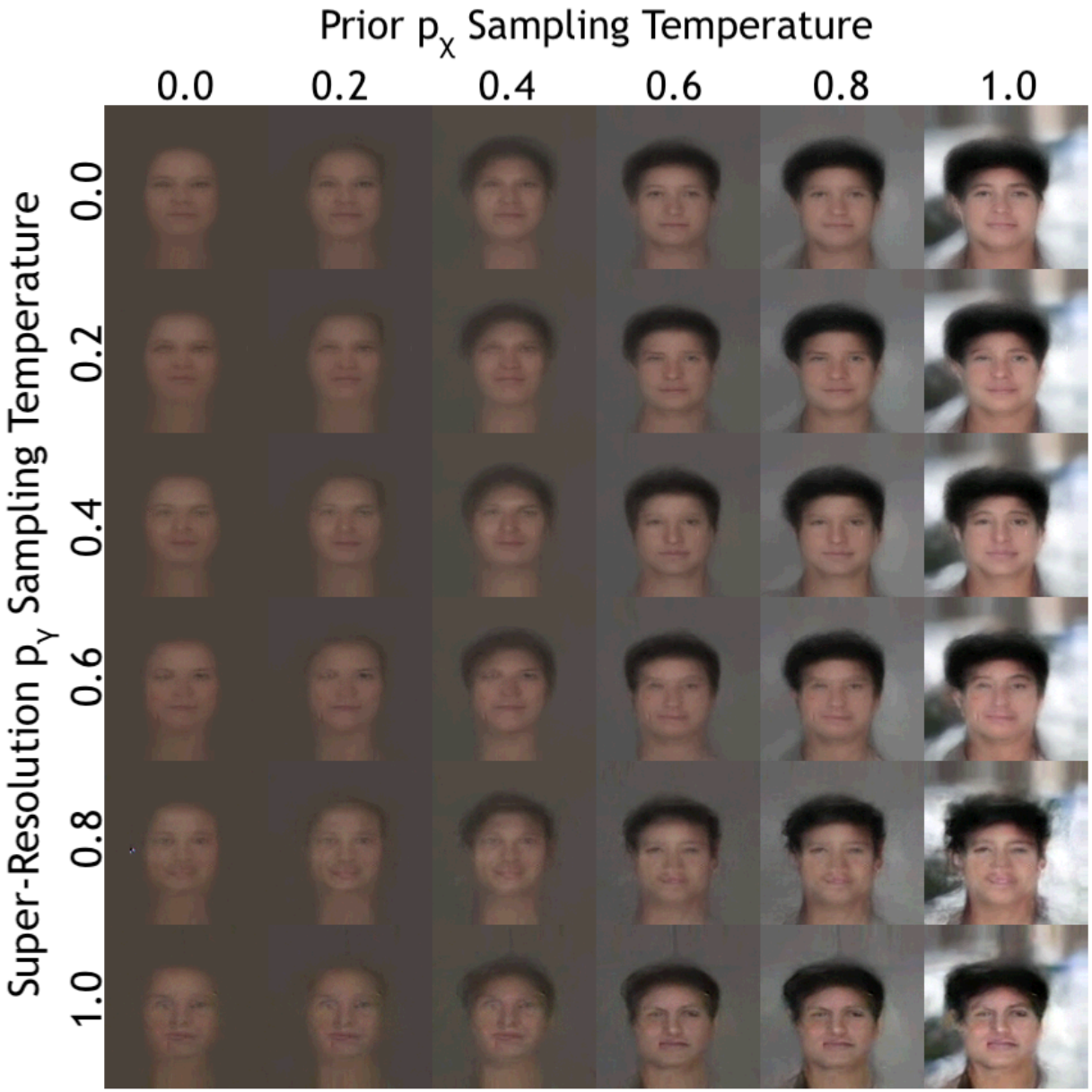



Figure: 8x Super-resolution on celebrity faces.

# Diversity in Learned Distribution: 4x vs. 8x SR

# Interpolation on Joint PSR Flow



# Semantic Structure vs. Stylistic Features

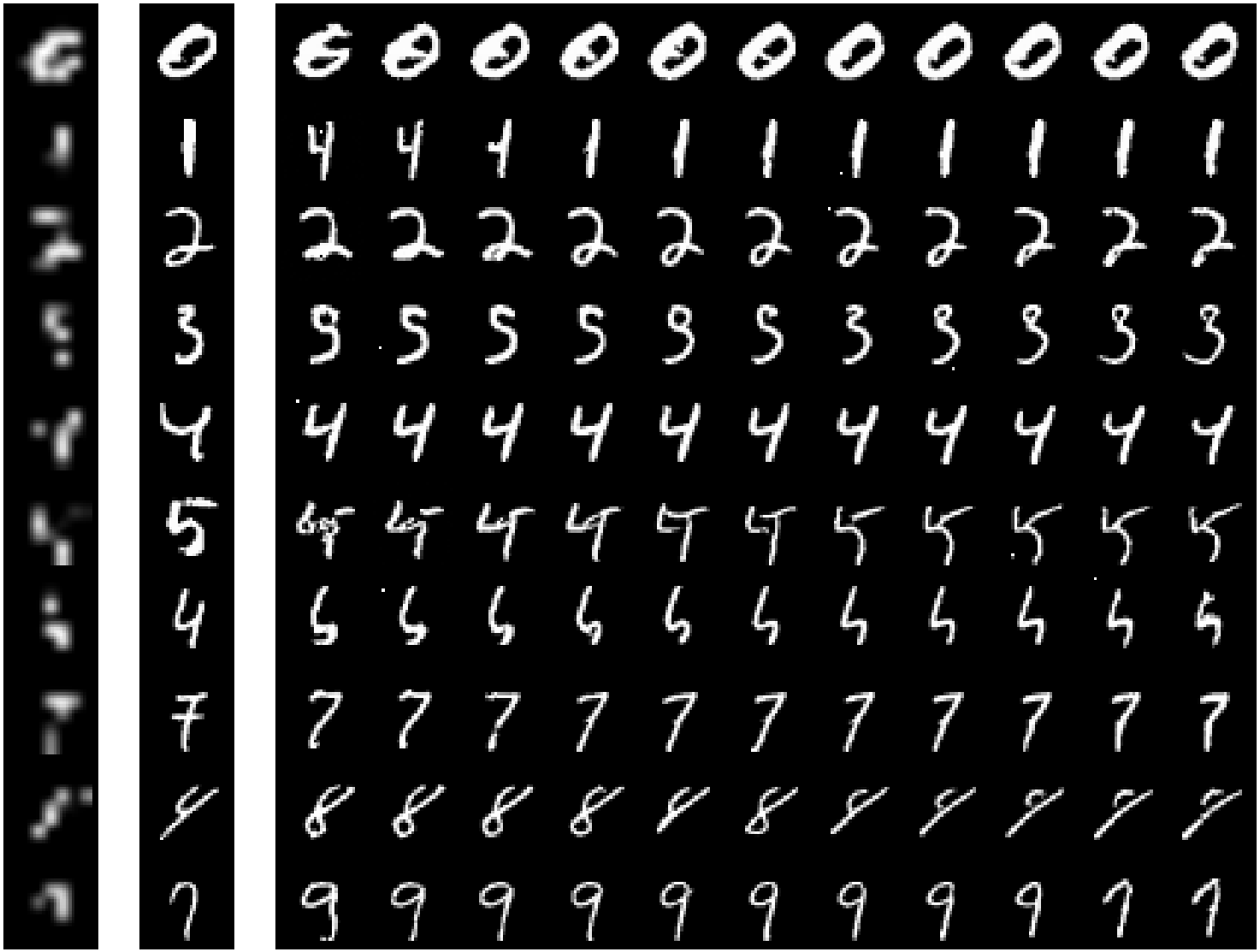


Figure: *Left:* LR input. *Center:* True. *Right:* PSR interpolations.

# Summary

PSR:

- Retains advantages of flow-based models
- Addresses the information bottleneck and inductive bias problems
- Learn diverse outputs, useful beyond image modeling.

# Outline

# Summary of Contributions

- Graphical models and variational inference
  - New topic model and scalable inference algorithm
  - Regularized variational inference

- Normalizing flows
  - Gradient boosted: flexibility
  - Compressive: compact representations with minimal degredation
  - PSR: learn high-d distribution from a low-d

# Questions?

Thank you!!!

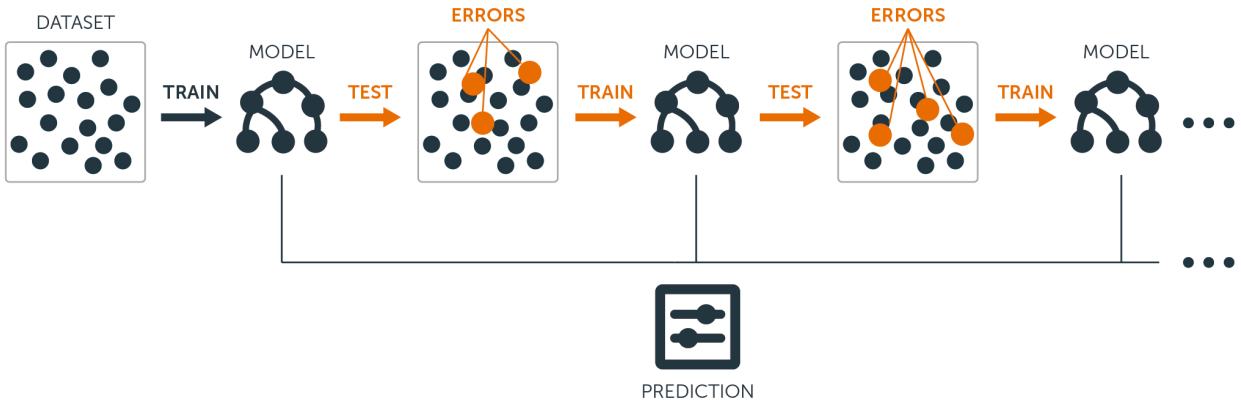
Link to slides, written dissertation, and supplemental material:
`https://github.com/robert-giaquinto/Advancing-Probabilistic-Models-for-Approximate-and-Exact-Inference`

# Acknowledgements

This dissertation was supported by NSF grants OAC-1934634, IIS-1908104, IIS-1563950, IIS-1447566, IIS-1447574, IIS-1422557, CCF-1451986. We thank the University of Minnesota Supercomputing Institute for technical support.

# Outline

# Gradient Boosting




Goal: Minimize loss $\mathcal{F}(G)$, where $G(\cdot)$ is current model

Consider: Perturbation $G + \epsilon g$

- $g$ is a new component function
- Taylor expansion as $\epsilon \to 0$:

$$\mathcal{F}(G + \epsilon g) = \mathcal{F}(G) + \epsilon \langle g, \nabla \mathcal{F}(G) \rangle + o(\epsilon^2) \ , \tag{5}$$

$\implies$ **functional gradient $\nabla \mathcal{F}(G)$ is the direction that reduces the loss at the current solution.**

Mason et al. 1999; Friedman et al. 2000; Friedman 2001, 2002

# Unsupervised Boosting

- Weigh distributions $G$ and $g$ by $\rho \in [0, 1]$

  $\Longrightarrow$ boosting outside of classification or regression

- *Boosting variational inference*: improve variational posterior by iteratively adding simple approximations

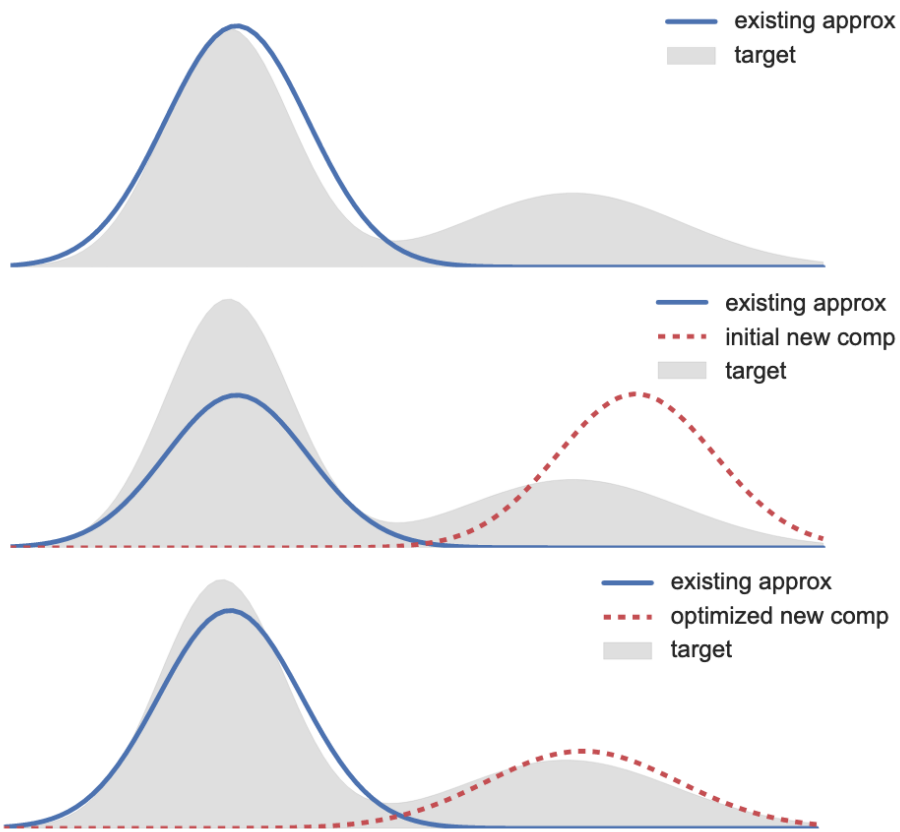



Figure: Trade-off additional computation by adding more components for more flexible posterior.

Rosset and Segal 2002; Miller et al. 2017; Guo et al. 2016

# How to Optimize VAE Objective?

Challenge: Gradients of ELBO objective are non-trivial!

- Gradients w.r.t. decoder $\nabla_\theta \mathcal{L}_{\theta,\phi}(\mathsf{x})$ possible:

$$\begin{aligned} \nabla_\theta \mathcal{L}_{\theta,\phi}(\mathsf{x}) &= \nabla_\theta \mathbb{E}_{q_\phi(\mathsf{z}|\mathsf{x})} \left[\log p_\theta(\mathsf{x},\mathsf{z}) - \log q_\phi(\mathsf{z} \mid \mathsf{x})\right] \\ &= \mathbb{E}_{q_\phi(\mathsf{z}|\mathsf{x})} \left[\nabla_\theta \log p_\theta(\mathsf{x},\mathsf{z}) - \log q_\phi(\mathsf{z} \mid \mathsf{x})\right] \end{aligned} \tag{6}$$

- But, w.r.t. encoder: $\nabla_\phi \mathcal{L}_{\theta,\phi}(\mathsf{x})$ not possible:

$$\begin{aligned} \nabla_\phi \mathcal{L}_{\theta,\phi}(\mathsf{x}) &= \nabla_\phi \mathbb{E}_{q_\phi(\mathsf{z}|\mathsf{x})} \left[\log p_\theta(\mathsf{x},\mathsf{z}) - \log q_\phi(\mathsf{z} \mid \mathsf{x})\right] \\ &\neq \mathbb{E}_{q_\phi(\mathsf{z}|\mathsf{x})} \left[\nabla_\phi \log p_\theta(\mathsf{x},\mathsf{z}) - \log q_\phi(\mathsf{z} \mid \mathsf{x})\right] \end{aligned} \tag{7}$$

***Core problem being solved by VAE***

# Solution: Reparameterization Trick

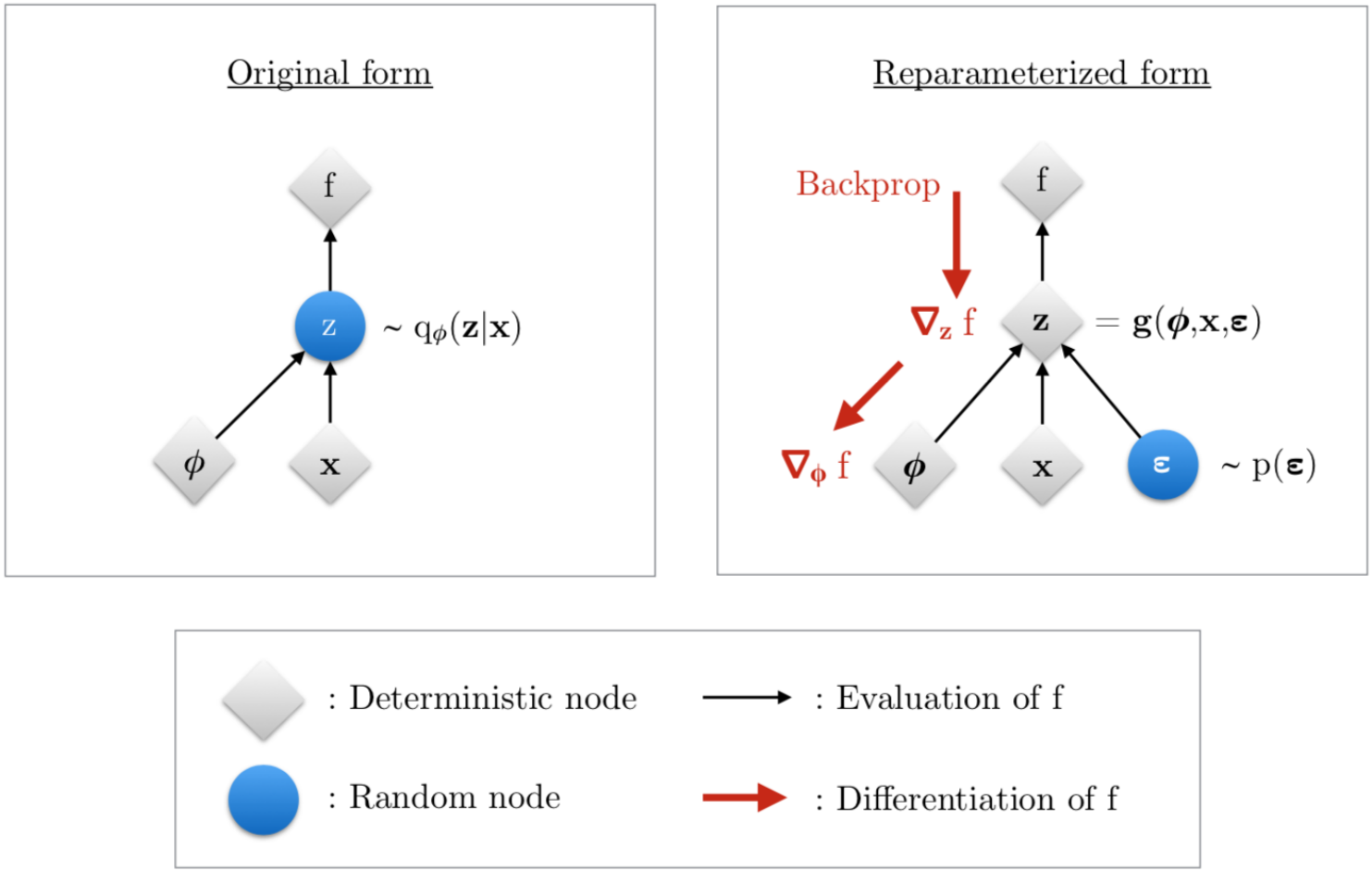
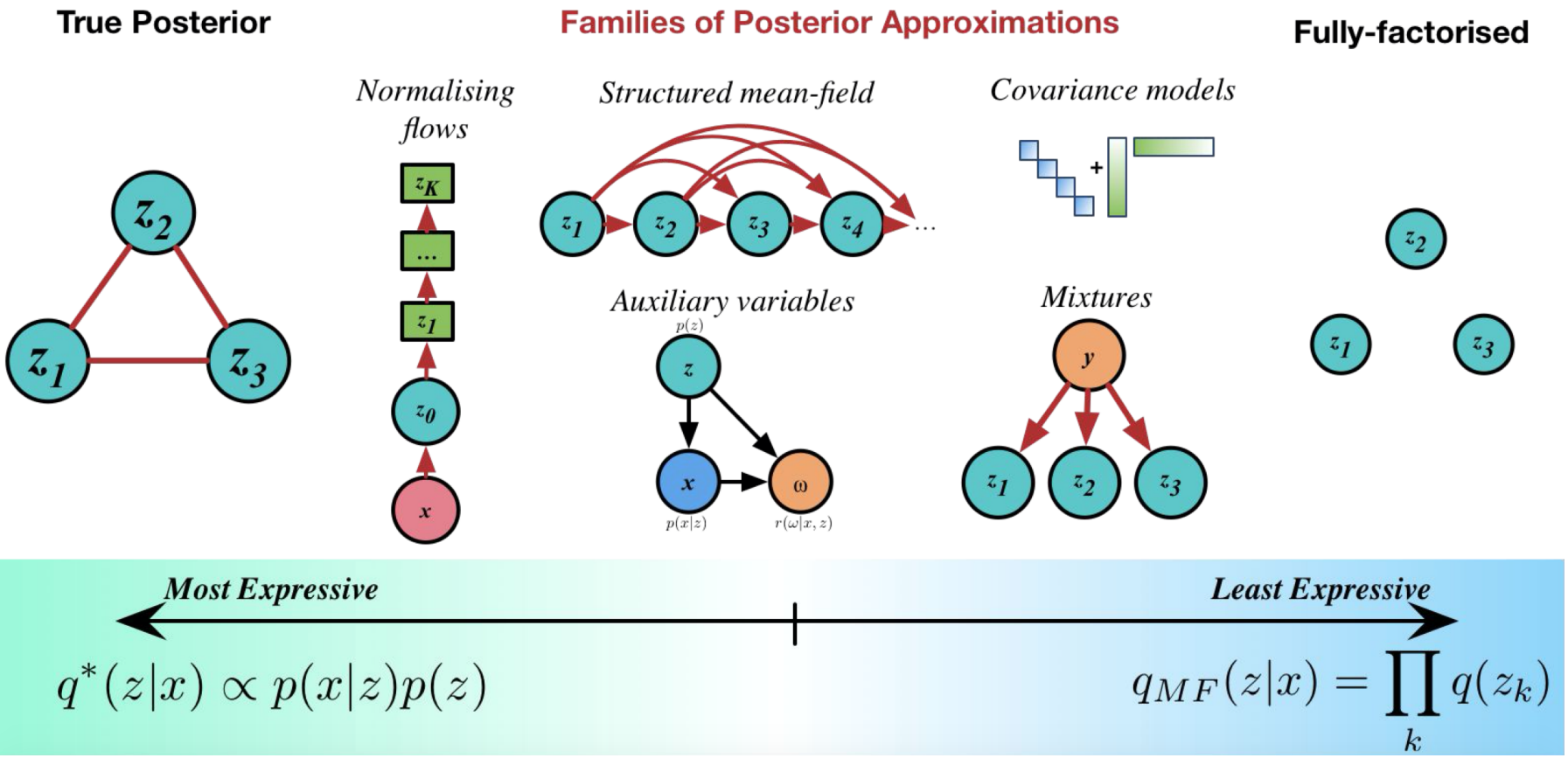



Figure: Variational parameters $\phi$ effect objective $f$ through the random variable $z \sim q_\phi(\mathrm{z} \mid \mathrm{x})$, so cannot compute $\nabla_\phi f$. Reparameterization trick (RT): reparameterize z as deterministic function of $\phi, \mathrm{x}$, and random noise $\epsilon$.

# The Larger Context



# Example: Planar Flow

- Each transformation:

$$f(z) = z + u \cdot h(w^\top z + b)$$

- $u, w, b$ are coefficients
  - Given by encoder for each data-point
- $h(\cdot)$ a non-linear activation function

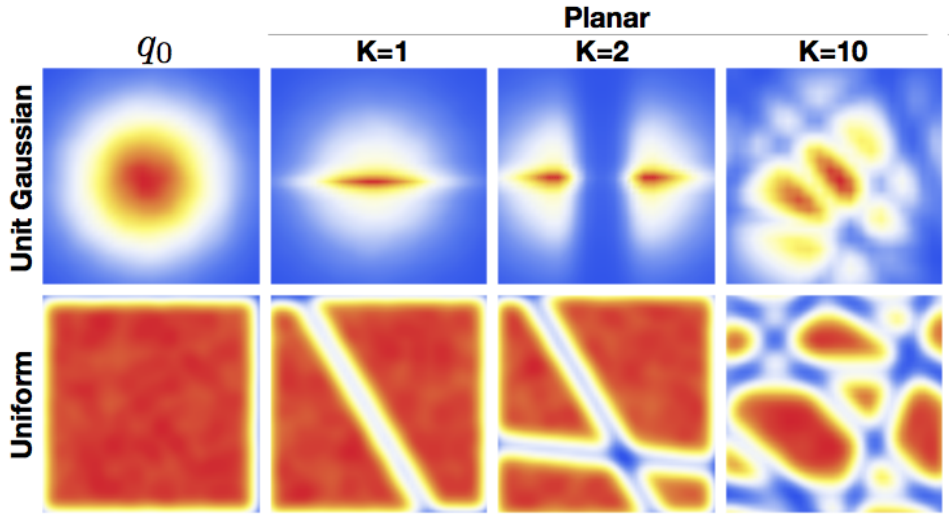



Figure: Normalizing flows transform a simple density into a more complex one.

**Why VAE+NF instead of more layers in the encoder?**

Rezende and Mohamed (2015), "Variational Inference with Normalizing Flows"

# Related Work: Normalizing Flows

## The trend: deeper, more complex transformations

For reference only:

- Sylvester flows (van den Berg et al., 2018) generalize planar flows (Rezende and Mohamed, 2015) into a more expressive framework
- Inverse autoregressive flows (IAF, (Kingma et al., 2016)) and masked autoregressive flows (MAF, (Papamakarios et al., 2017)) scale to higher dimensions
- Neural autoregressive flows (NAF, (Huang et al., 2018a)) and the more compact Block NAF (De Cao et al., 2019), replace affine transformations with autoregressive neural networks.

*Analytically* Invertible Flows:

- Flows based on coupling layers like NICE (Dinh et al., 2015) and successor RealNVP (Dinh et al., 2017).
- Glow replaced RealNVP's permutation operation with a $1 \times 1$ convolution (Kingma and Dhariwal, 2018)
- Neural spline flows provide a method for increasing the flexibility of both coupling and autoregressive transforms using monotonic rational-quadratic splines (Durkan et al., 2019)

# RealNVP

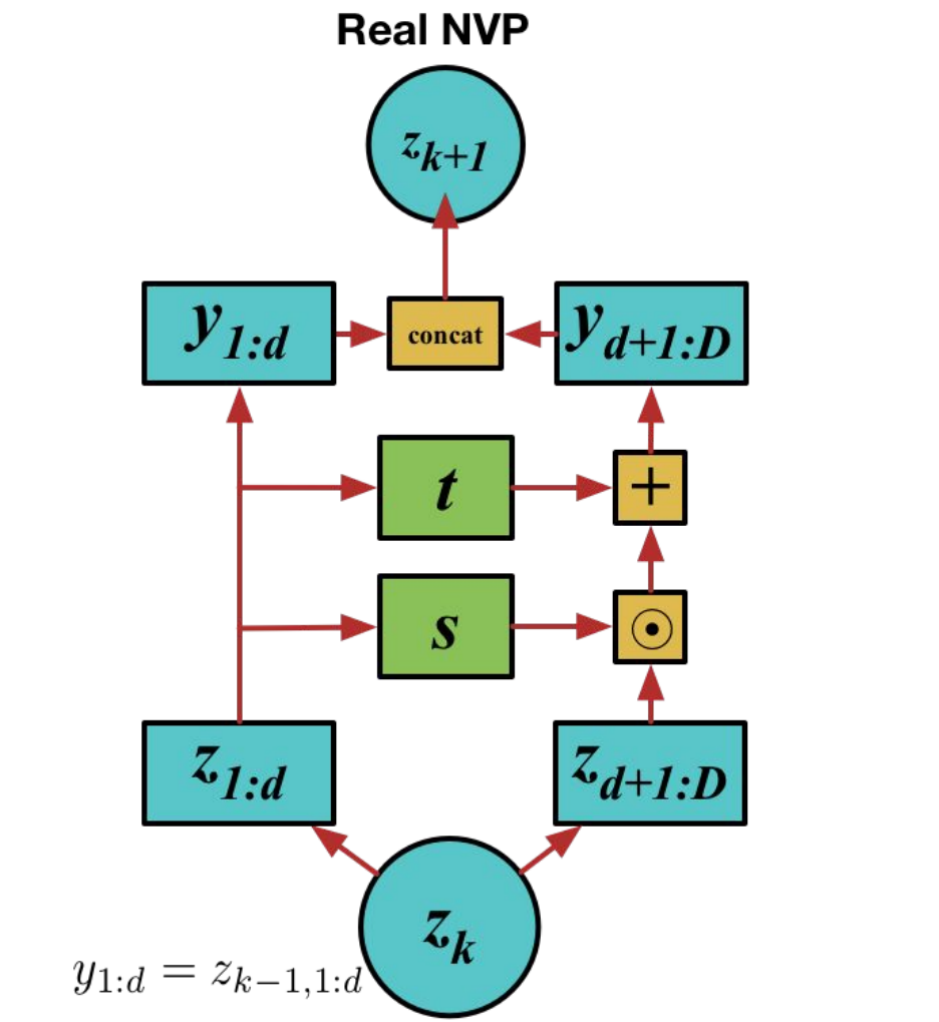



Figure: RealNVP transformations, taken from
`https://www.shakirm.com/slides/DeepGenModelsTutorial.pdf`

# Flows Compatible with GBNF for Variational Inference

- In theory: All flows compatible
- In practice: *Analytically* invertible flows much easier
  - Affine/lower-triangular[17], non-linear squared[18], neural spline[19], NICE and RealNVP[20], Glow[21]
  - IAF and MAF are $D$ times slow to invert [22]

$\rightarrow$ **GBNF compatible with many existing flows!**

[17] Louizos and Welling 2017; Tomczak and Welling 2016, 2017
[18] Ziegler and Rush 2019
[19] Durkan et al. 2019
[20] Dinh et al. 2015, 2017
[21] Kingma and Dhariwal 2018
[22] Kingma et al. 2016; Papamakarios et al. 2017

# What about Variational Inference?

- With GBNF approximate posterior, the VAE objective is:

$$\mathcal{F}^{(VI)}_{\phi,\theta}(\mathrm{x}) = \mathbb{E}_{G^{(c)}_K}\left[\log G^{(c)}_K(\mathrm{z}_K \mid \mathrm{x}) - \log p_\theta(\mathrm{x}, \mathrm{z}_K)\right]$$

# What about Variational Inference?

- With GBNF approximate posterior, the VAE objective is:

$$\mathcal{F}^{(VI)}_{\phi,\theta}(\mathrm{x}) = \mathbb{E}_{G^{(c)}_K}\left[\log G^{(c)}_K(\mathrm{z}_K \mid \mathrm{x}) - \log p_\theta(\mathrm{x}, \mathrm{z}_K)\right]$$

- Consider additive GBNF posterior:

$$G^{(c)}_K(\mathrm{z} \mid \mathrm{x}) = (1-\rho_c)G^{(c-1)}_K(\mathrm{z} \mid \mathrm{x}) + \rho_c g^{(c)}_K(\mathrm{z} \mid \mathrm{x})$$

  - Samples from posterior: $z^{(c)}_K = f^{(c)}_K \circ \cdots \circ f^{(c)}_1(z_0)$
    - Mixture component $(c)$ chosen based on weights $\rho_{1:c}$
  - Base distribution $q_0$ shared by all components

# Adding a New Boosted Component

- Goal: find $g_K^{(c)}$ based on functional gradient descent.

$$\nabla_{G_K^{(c)}} \mathcal{F}_{\phi,\theta}(\mathrm{x})\big|_{\rho_c \to 0} = -\log \frac{p_\theta(\mathrm{x}, \mathrm{z})}{G_K^{(c-1)}(\mathrm{z} \mid \mathrm{x})} + 1$$

# Adding a New Boosted Component

- Goal: find $g_K^{(c)}$ based on functional gradient descent.

$$\nabla_{G_K^{(c)}} \mathcal{F}_{\phi,\theta}(\mathrm{x})\big|_{\rho_c \to 0} = -\log \frac{p_\theta(\mathrm{x},\mathrm{z})}{G_K^{(c-1)}(\mathrm{z} \mid \mathrm{x})} + 1$$

- Boosting: Choose $g_K^{(c)}$ with minimum inner product with $\nabla_{G_K^{(c)}} \mathcal{F}_{\phi,\theta}(\mathrm{x})$:

$$\begin{aligned} g_K^{(c)}(\mathrm{z} \mid \mathrm{x}) &= \arg\min_{g_K \in \mathcal{G}_K} \sum_{i=1}^{n} \mathop{\mathbb{E}}_{g_K(\mathrm{z}|\mathrm{x}_i)} [\nabla_G \mathcal{F}(\mathrm{x}_i)] \\ &= \arg\min_{g_K \in \mathcal{G}_K} \mathop{\mathbb{E}}_{g_K(\mathrm{z}|\mathrm{x})} \left[ -\log \frac{p_\theta(\mathrm{x},\mathrm{z})}{G_K^{(c-1)}(\mathrm{z} \mid \mathrm{x})} \right] \end{aligned}$$

# Adding a New Boosted Component

- Goal: find $g_K^{(c)}$ based on functional gradient descent.

$$\nabla_{G_K^{(c)}} \mathcal{F}_{\phi,\theta}(x)\big|_{\rho_c \to 0} = -\log \frac{p_\theta(x,z)}{G_K^{(c-1)}(z \mid x)} + 1$$

- Boosting: Choose $g_K^{(c)}$ with minimum inner product with $\nabla_{G_K^{(c)}} \mathcal{F}_{\phi,\theta}(x)$:

$$\begin{aligned} g_K^{(c)}(z \mid x) &= \arg\min_{g_K \in \mathcal{G}_K} \sum_{i=1}^{n} \mathop{\mathbb{E}}_{g_K(z|x_i)} [\nabla_G \mathcal{F}(x_i)] \\ &= \arg\min_{g_K \in \mathcal{G}_K} \mathop{\mathbb{E}}_{g_K(z|x)} \left[ -\log \frac{p_\theta(x,z)}{G_K^{(c-1)}(z \mid x)} \right] \end{aligned}$$

- Avoid degenerate $g_K^{(c)}$ by adding entropy regularization $\lambda > 0$:

$$g_K^{(c)}(z \mid x) = \arg\min_{g_K \in \mathcal{G}_K} \mathop{\mathbb{E}}_{g_K(z|x_i)} \left[ -\log \frac{p_\theta(x,z)}{G_K^{(c-1)}(z \mid x)} + \lambda \log g_K(z \mid x) \right]$$

# Comparison to VAE Objective

- VAE + GBNF Objective:

$$\mathcal{F}_{\theta,G}(\mathrm{x}) = \mathbb{E}_{g_K}\left[-\log \frac{p_\theta(\mathrm{x} \mid \mathrm{z})}{G_K^{(c-1)}(\mathrm{z} \mid \mathrm{x})}\right] + \lambda KL\left(g_K(\mathrm{z} \mid \mathrm{x}) \,||\, p(\mathrm{z})\right)$$

# Comparison to VAE Objective

- VAE + GBNF Objective:

$$\mathcal{F}_{\theta,G}(\mathrm{x}) = \mathbb{E}_{g_K}\left[-\log \frac{p_\theta(\mathrm{x} \mid \mathrm{z})}{G_K^{(c-1)}(\mathrm{z} \mid \mathrm{x})}\right] + \lambda KL\left(g_K(\mathrm{z} \mid \mathrm{x}) \,||\, p(\mathrm{z})\right)$$

- VAE Objective:

$$\mathcal{F}_{\theta,\phi}(\mathrm{x}) = \mathbb{E}_{q_K}\left[-\log \frac{p_\theta(\mathrm{x} \mid \mathrm{z})}{const}\right] + KL\left(q_K(\mathrm{z} \mid \mathrm{x}) \,||\, p(\mathrm{z})\right)$$

Similarity: KL-divergence regularization

Difference: Down-weight reconstruction of samples explained by $G_K^{(c-1)}$

# Update Component Weights

Gradient of the loss $\mathcal{F}_{\phi,\theta}(\mathsf{x})$ with respect to $\rho_c$

$$\frac{\partial \mathcal{F}_{\phi,\theta}}{\partial \rho_c} = \sum_{i=1}^{n} \left( \mathop{\mathbb{E}}_{g_K^{(c)}(\mathsf{z}|\mathsf{x}_i)} \left[ \gamma_{\rho_c}^{(t-1)}(\mathsf{z} \mid \mathsf{x}_i) \right] - \mathop{\mathbb{E}}_{G_K^{(c-\mathbf{1})}(\mathsf{z}|\mathsf{x}_i)} \left[ \gamma_{\rho_c}^{(t-1)}(\mathsf{z} \mid \mathsf{x}_i) \right] \right) ,$$

where,

$$\gamma_{\rho_c}^{(t-1)}(\mathsf{z} \mid \mathsf{x}_i) \triangleq \log \left( \frac{(1-\rho_c^{(t-1)}) G_K^{(c-1)}(\mathsf{z} \mid \mathsf{x}_i) + \rho_c^{(t-1)} g_K^{(c)}(\mathsf{z} \mid \mathsf{x}_i)}{p_\theta(\mathsf{x}_i, \mathsf{z})} \right) .$$

- Plug into SGD algorithm
- Update $\rho_c$ only after training $g_K^{(c)}$
- Can "fine-tune" $\rho$ after training all components

# Density Matching

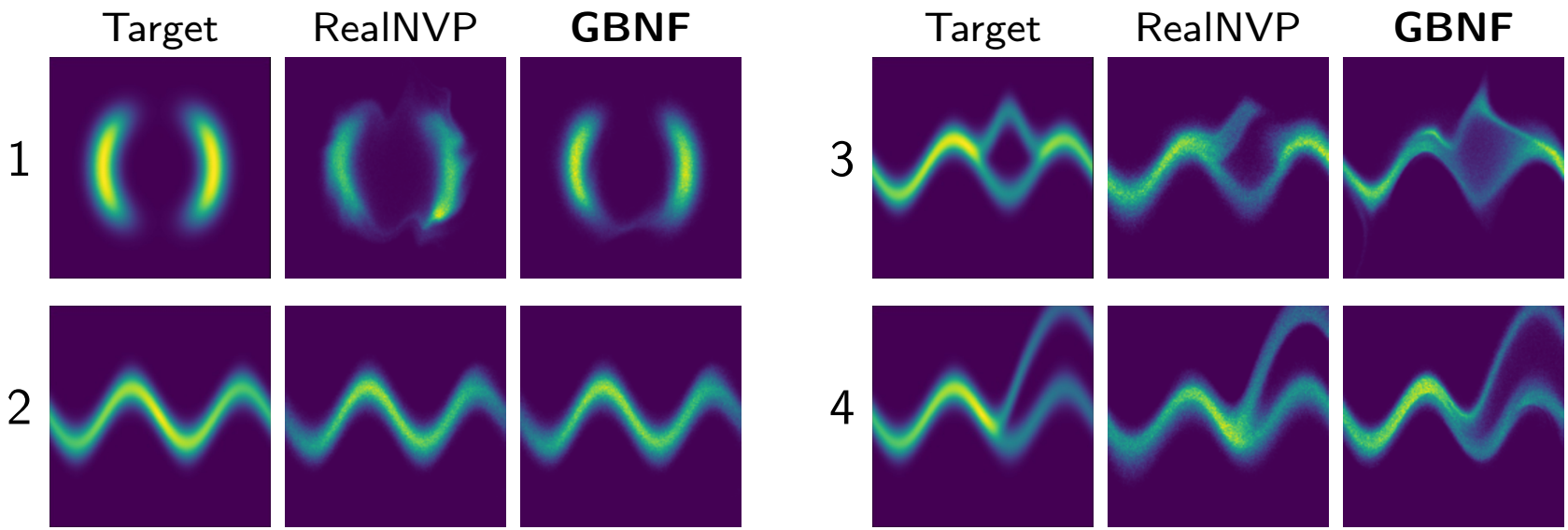



Figure: Matching the energy functions from Table 1 of Rezende and Mohamed (2015). The middle columns show deep RealNVPs with $K = 16$ flows. Gradient boosting RealNVP with $c = 2$ components of length $K = 4$ performs as well or better with **half as many parameters**.

# Density Matching: Each Component

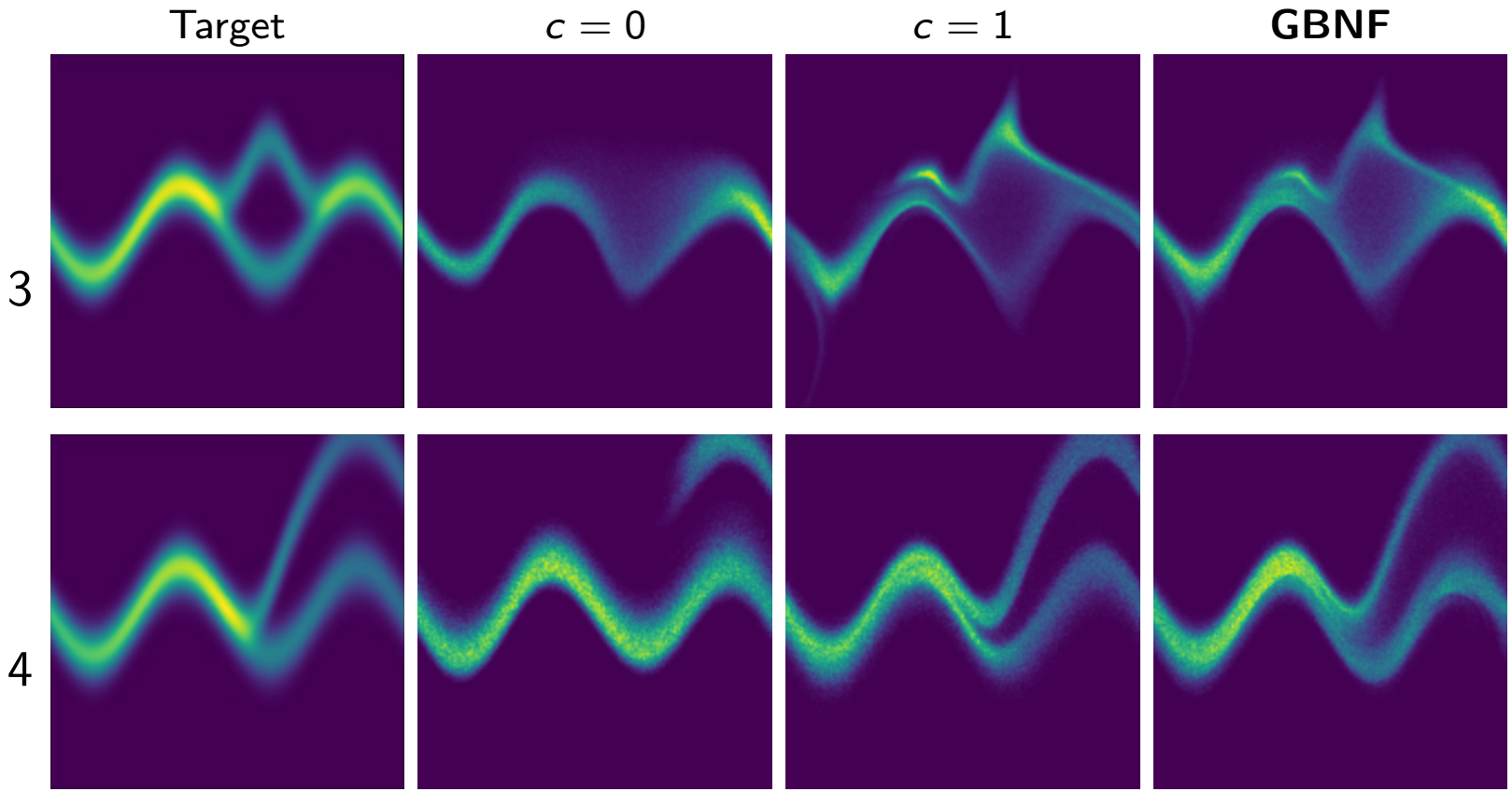
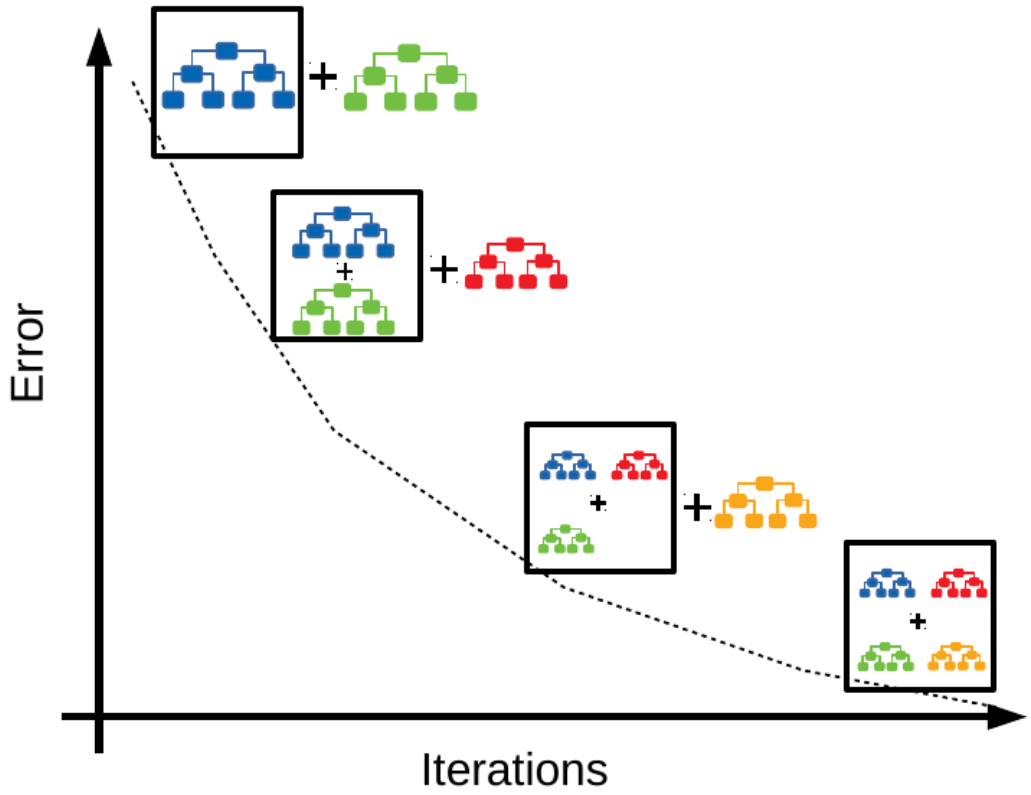



Figure: Individual component results on density matching problem.

# Decoder Shock




- Gradient boosting flows unlike decision trees
- Decoder shared by all components
- What happens when we begin training a new component?

# Decoder Shock: Abrupt Changes to VAE's Posterior

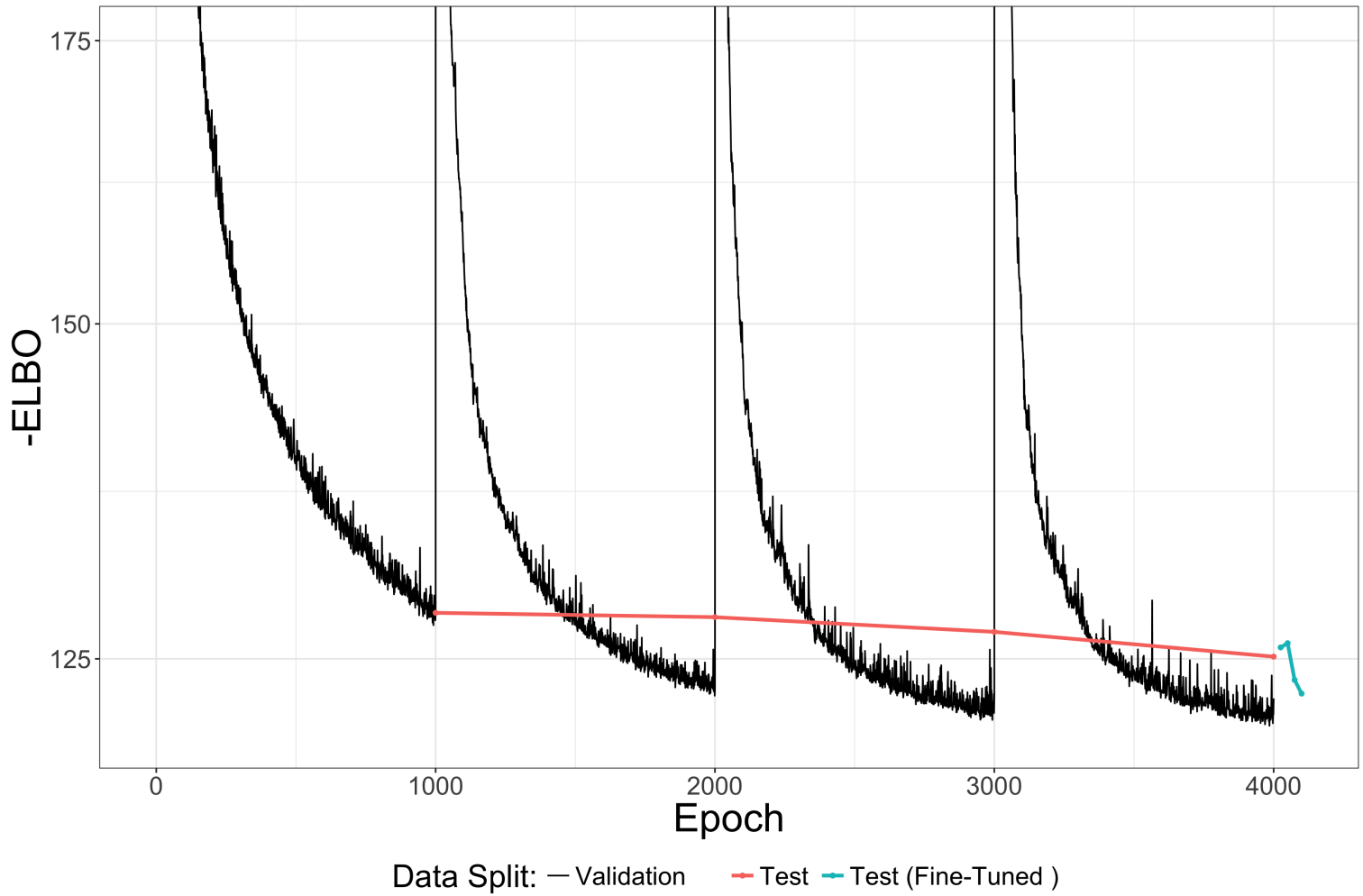



Figure: Loss on the test set decreases steadily as we add new components. The validation loss, however, jumps when a new component is introduced due to a sudden change in the distribution of samples passed to the decoder (aka "decoder shock").

# Reasons for Decoder Shock

New component = sudden shift in distribution of samples, why?

1. Samples from **new** $g_K^{(c)}$
2. Objective: annealing $KL(q(\mathrm{z} \mid \mathrm{x}) \,||\, p(\mathrm{z}))$ from 0 to 1
   $\implies$ No regularization (initially)
   $\implies$ Model is free to find *very* flexible posterior

**Solution:**

# Reasons for Decoder Shock

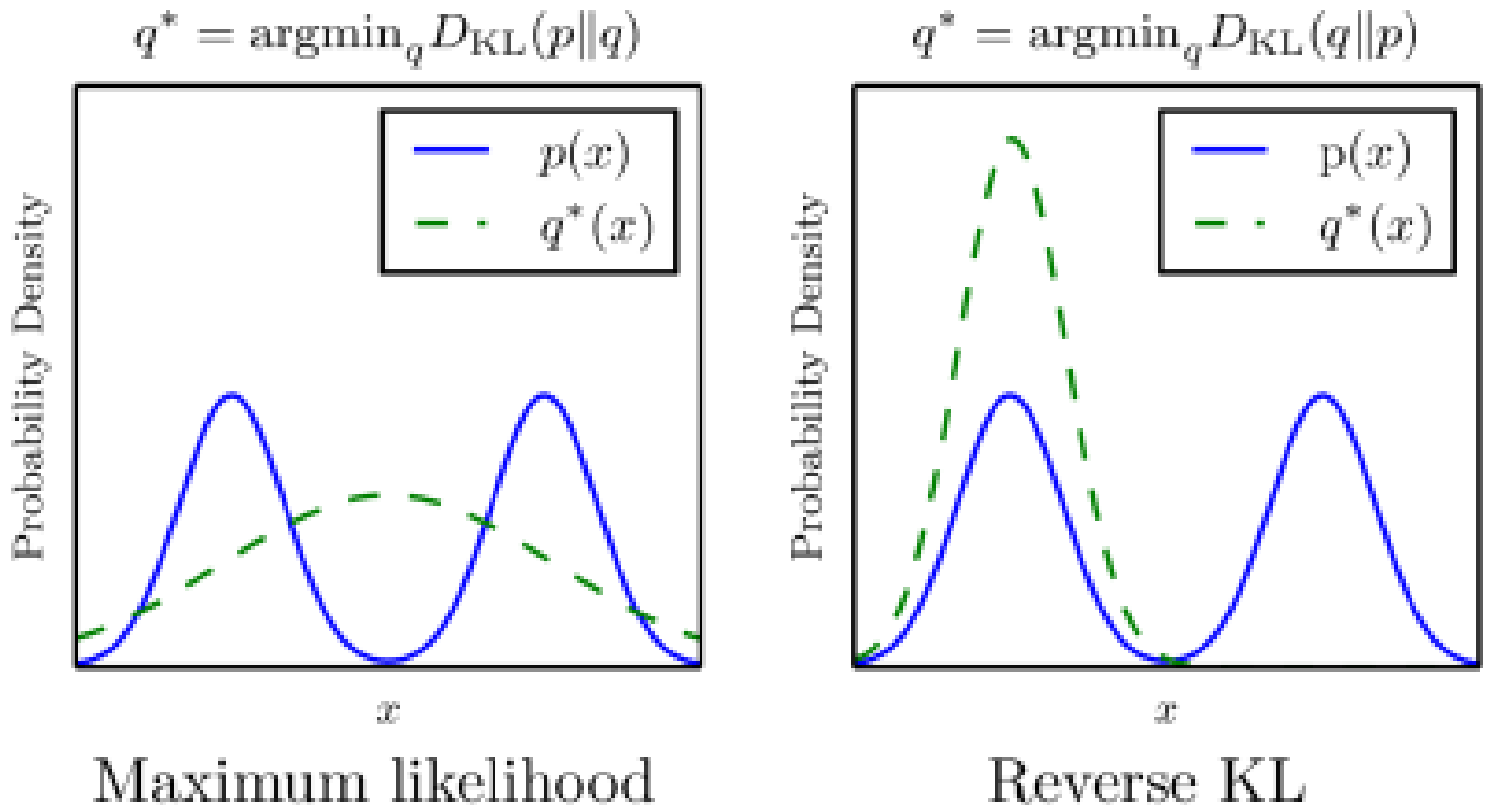
New component = sudden shift in distribution of samples, why?

1. Samples from **new** $g_K^{(c)}$
2. Objective: annealing $KL(q(\mathrm{z} \mid \mathrm{x}) \,||\, p(\mathrm{z}))$ from 0 to 1
   $\implies$ No regularization (initially)
   $\implies$ Model is free to find *very* flexible posterior

**Solution:**

- Blend in samples from fixed components
- Helps "remember" previous components
- $G_K^{(c-1)}$ still fixed

# KL Divergences



# How do we arrive at deep probabilistic models?

**(Classic) Probabilistic Modeling**

- – Mainly conjugate and linear models
- – Computationally expensive, intractable

- \+ Unified framework
  - Model building, inference, prediction, and decision making.
- \+ Model uncertainty and variability in outcomes.
- \+ Robust to overfitting.

**Deep Learning**

# How do we arrive at deep probabilistic models?

**(Classic) Probabilistic Modeling**

- – Mainly conjugate and linear models
- – Computationally expensive, intractable

+ Unified framework
  - Model building, inference, prediction, and decision making.
+ Model uncertainty and variability in outcomes.
+ Robust to overfitting.

**Deep Learning**

- – Deterministic point estimates.
- – Easily overfit.

+ Expressive.
+ Scalable.

# How do we arrive at deep probabilistic models?

**(Classic) Probabilistic Modeling**

- – Mainly conjugate and linear models
- – Computationally expensive, intractable

+ Unified framework
  - Model building, inference, prediction, and decision making.
+ Model uncertainty and variability in outcomes.
+ Robust to overfitting.

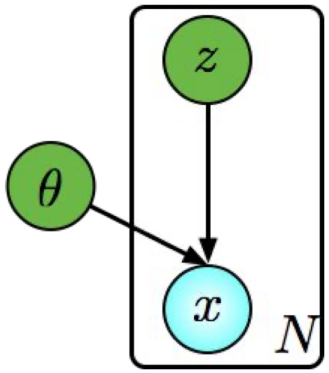
**Deep Learning**

- – Deterministic point estimates.
- – Easily overfit.

+ Expressive.
+ Scalable.

Next: Look back at probabilistic models, connect to modern deep learning

# Training Probabilistic Graphical Models




$$p(x, z, \theta) = \rho(\theta) \prod_{i=1}^{N} p(x_i|z_i, \theta)\pi(z_i)$$

Figure: Bayesian mixture of Gaussians. Evidence $x$, latent cluster assignment $z$, global parameters $\theta$. Box indicates $N$ repetitions for each observation.

- Customized to dataset
- Estimate latent variables with posterior inference. Bayes' Rule:

$$p(\mathrm{z}|\mathrm{x}) = \frac{p(\mathrm{x}|\mathrm{z})p(\mathrm{z})}{p(\mathrm{x})}$$

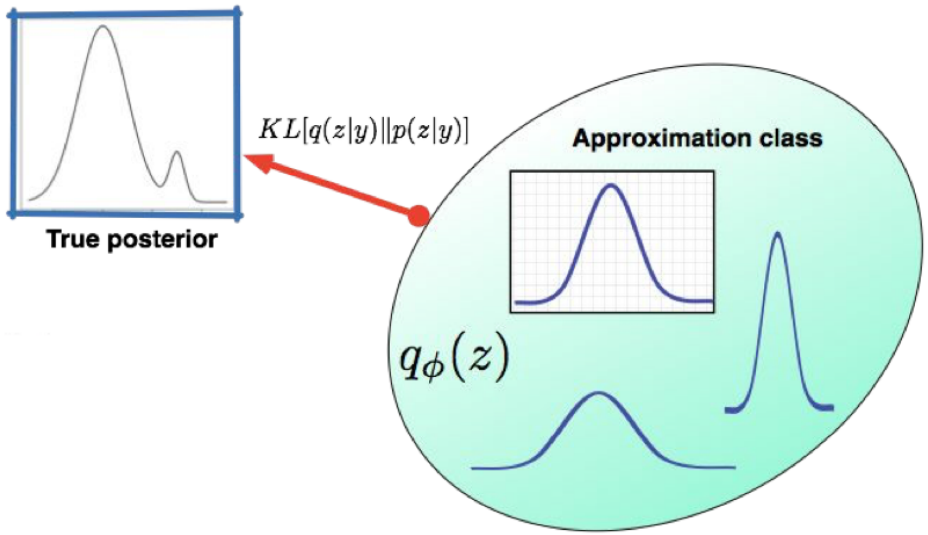
- Intractable due to $p(\mathrm{x})$.
  ...What to do? MCMC? Importance Sampling? Expectation-maximization?

# Variational Inference

- **Idea:** maximize lower bound $\mathcal{L}(\theta, \phi; \mathrm{x})$ on data

$$\log p(\mathrm{x}) = \log \mathbb{E}_{q_\phi(\mathrm{z}|\mathrm{x})} \left[ \frac{\log p_\theta(\mathrm{x} \mid \mathrm{z}) p(\mathrm{z})}{q_\phi(\mathrm{z} \mid \mathrm{x})} \right]$$
$$\geq \mathbb{E}_{q_\phi(\mathrm{z}|\mathrm{x})} [\log p_\theta(\mathrm{x} \mid \mathrm{z})] - KL(q_\phi(\mathrm{z} \mid \mathrm{x}) \,||\, p(\mathrm{z})) := \mathcal{L}(\theta, \phi; \mathrm{x})$$

- **Scalability:** inference into optimization
  - Optimization is easy if closed form solution available*
- **Challenge:** compute stochastic gradients $\nabla_\phi \mathcal{L}(\theta, \phi; \mathrm{x})$ for any model?



# Topic Modeling on Health Journals with Regularized Variational Inference

Robert Giaquinto and Arindam Banerjee
*AAAI*, 2018

# DAPPER: Scaling Dynamic Author Persona Topic Model to Billion Word Corpora

Robert Giaquinto and Arindam Banerjee
*ICDM*, 2018

# Motivation

CaringBridge Data. 14 million journals, 500k authors writing over *time*.

Modeling Challenges. Variable time scales, multiple illnesses.

Goals. (1) Identify common health journeys.

(2) Scale to full CaringBridge dataset.

# Contribution #1: New Topic Model and Inference Algorithm

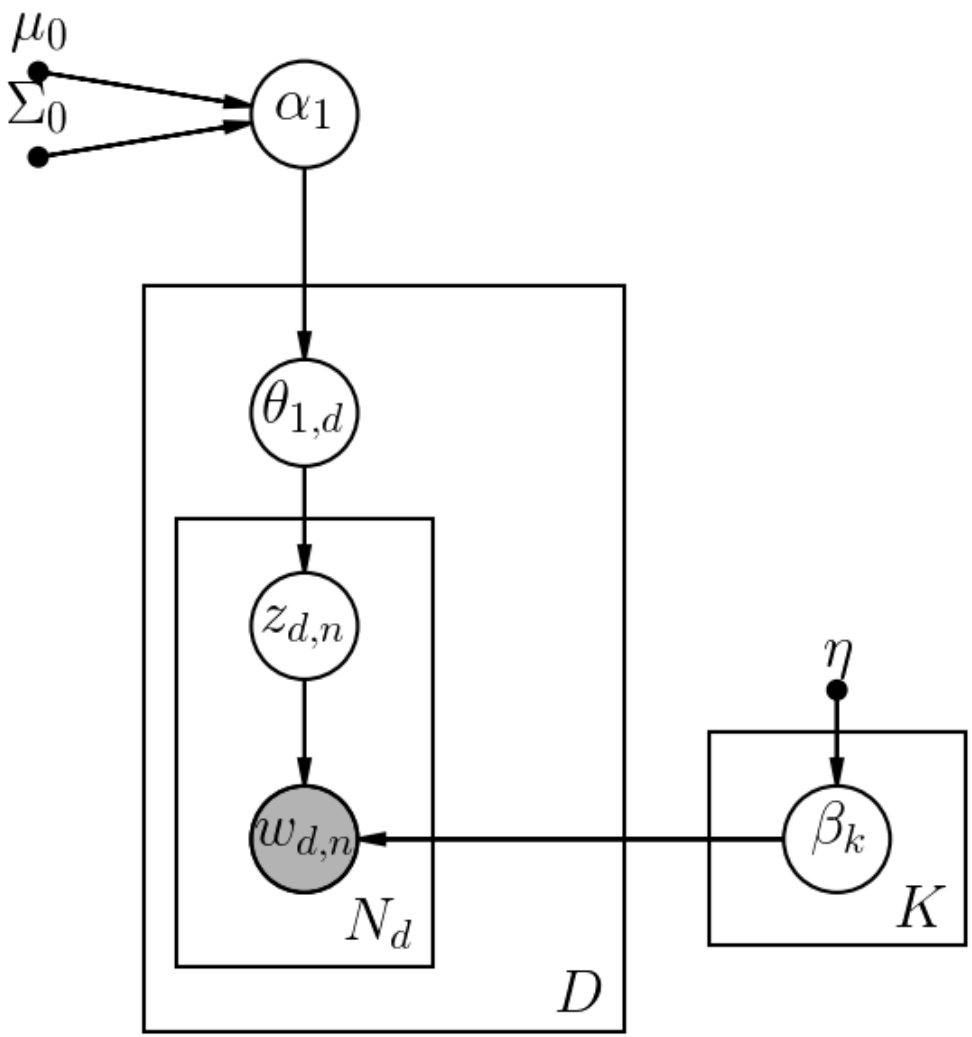
Modeling common narratives?

$\Longrightarrow$ Dynamic Author-Persona (DAP) Topic Model

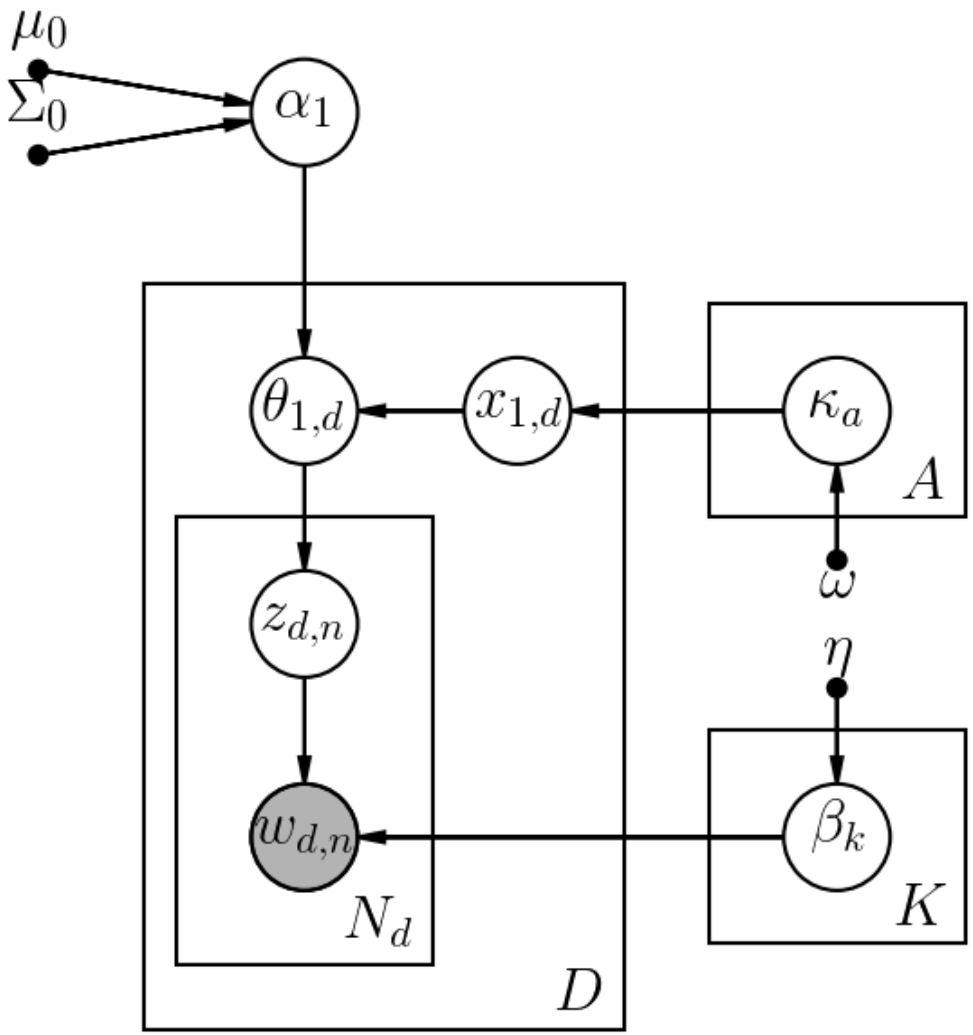
- Assign authors latent *personas*.
  - Soft clustering.
- Persona = propensity to discuss certain *topics over time*.

surgical procedure $\rightarrow$ recovery / PT $\rightarrow$ return to normal life.

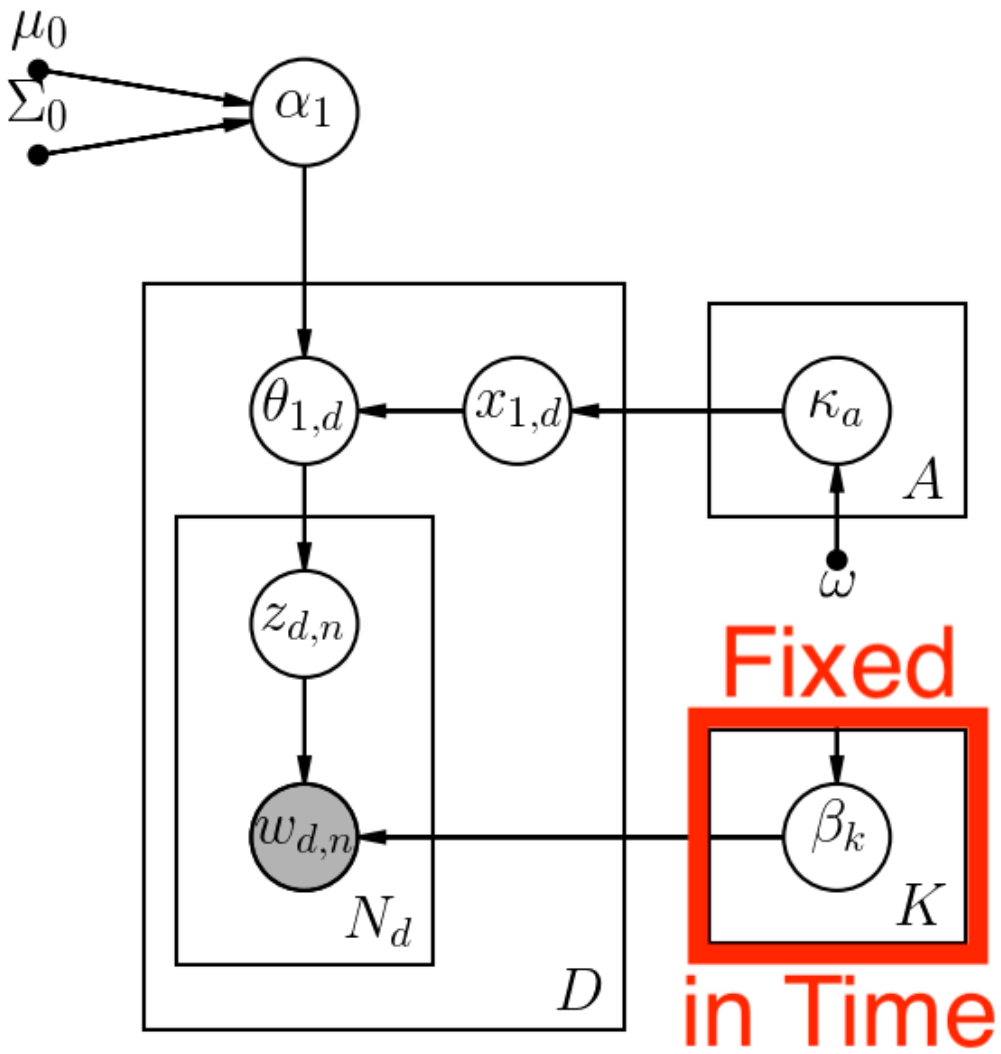
# DAP: Similar to Latent Dirichlet Allocation



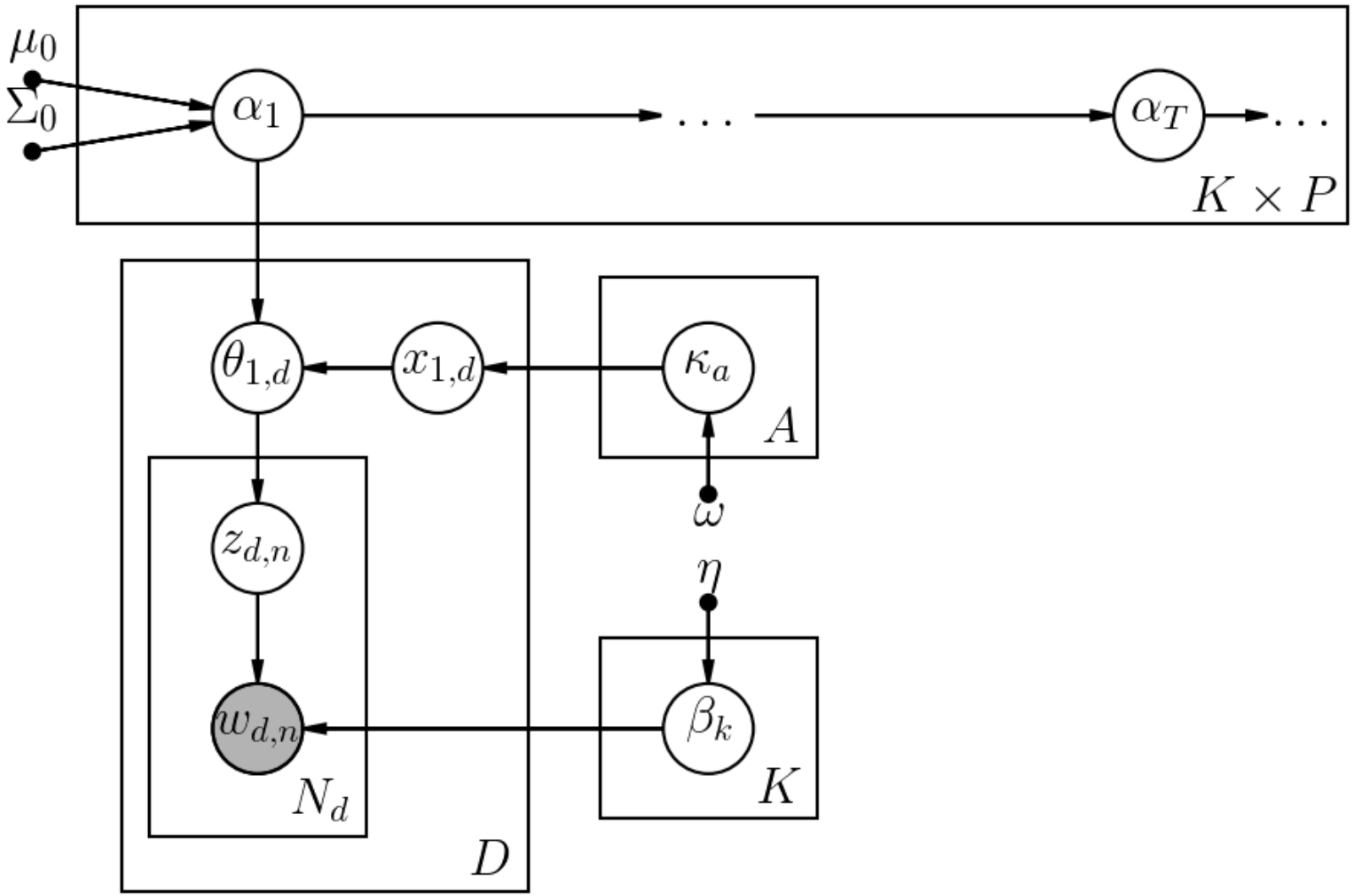
# DAP: Assigns Persona to Each Author



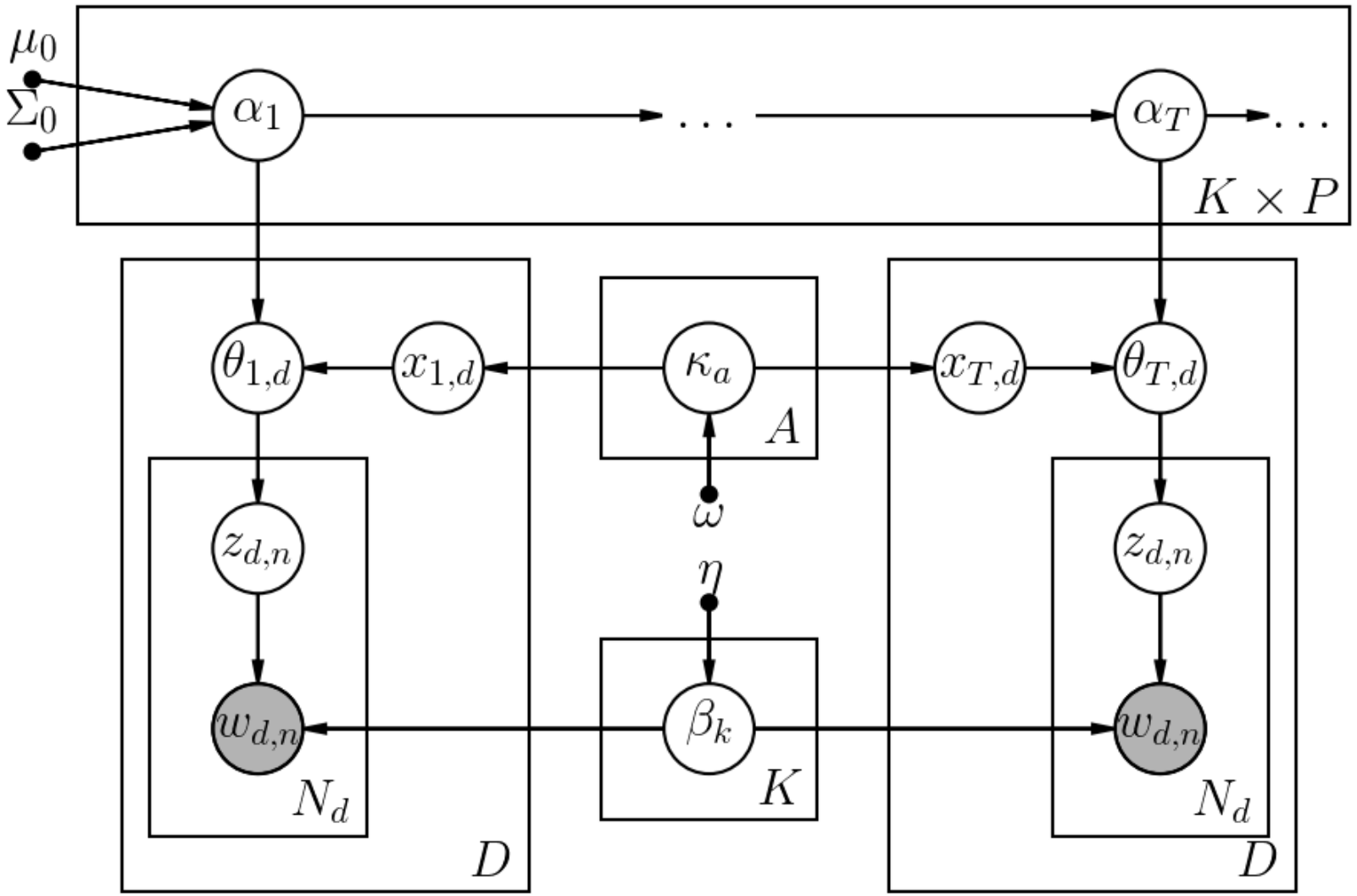
# DAP: Words in Topic are Fixed in Time



# DAP: Model Topics Over Time



# Dynamic Author Persona Topic Model



# Contribution #2: Regularized Variation Inference

RVI := encourage *certain* solutions

- RVI + DAP $\implies$ encourage distinct personas

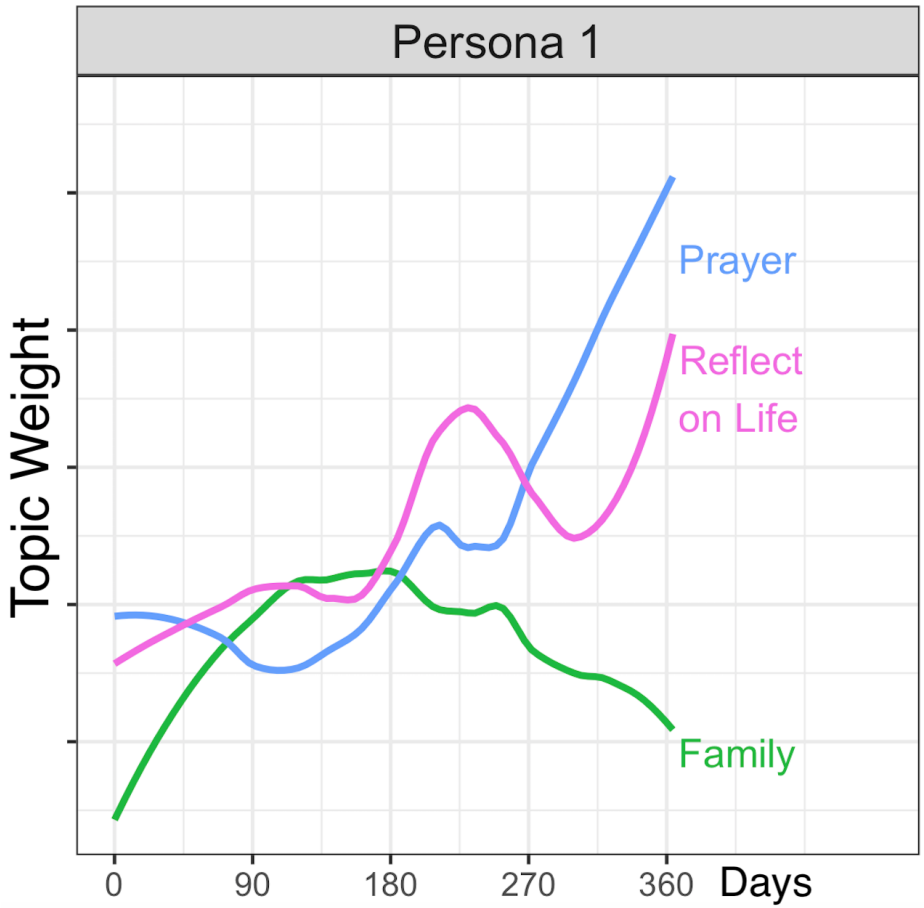



Figure: One persona found by unregularized DAP. Seems fine, right?

# Why Regularize DAP's Personas?


Figure: Unregularized DAP: **boring**! Personas are **homogenous**.

# Compelling Shared Narratives

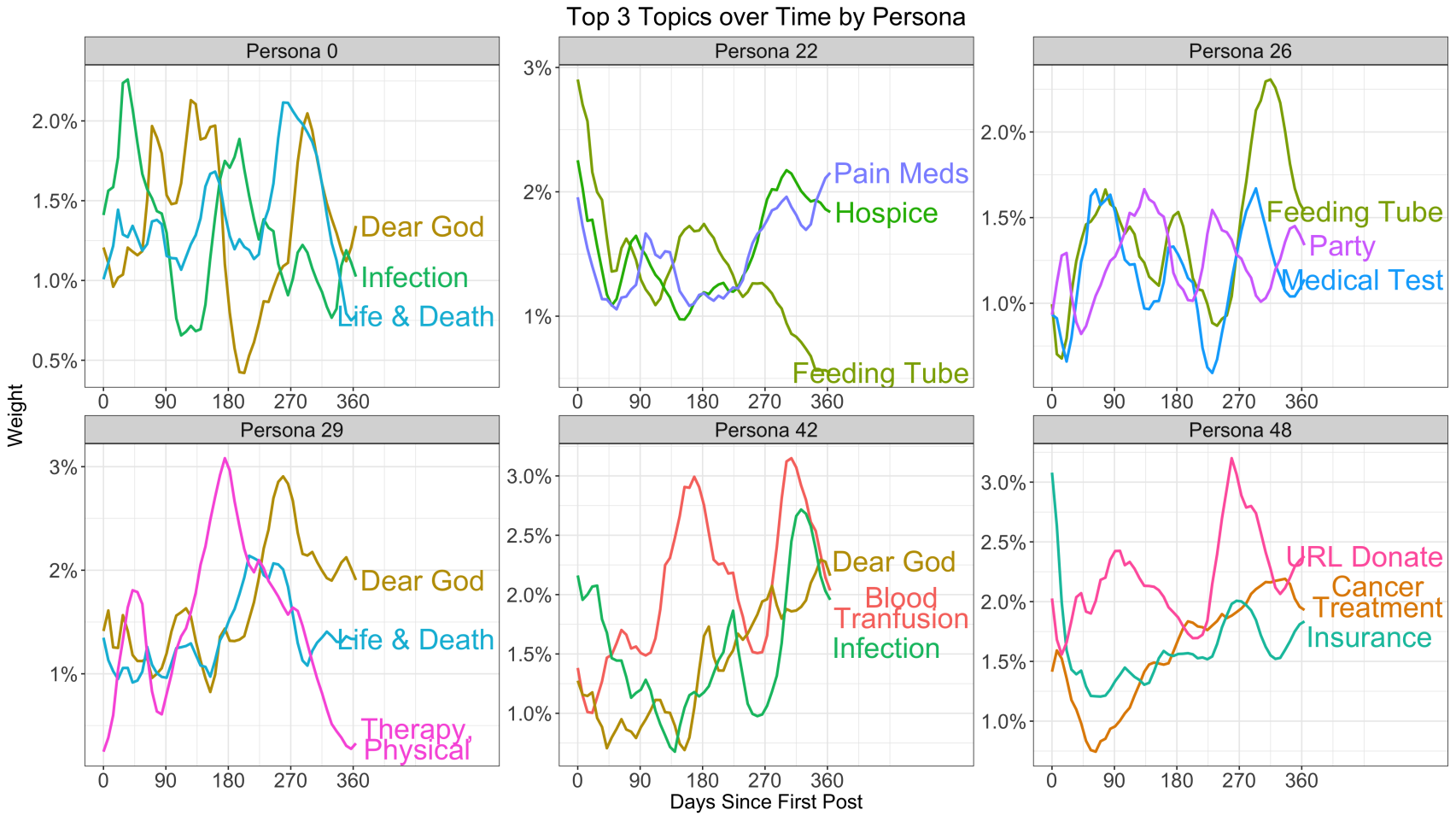
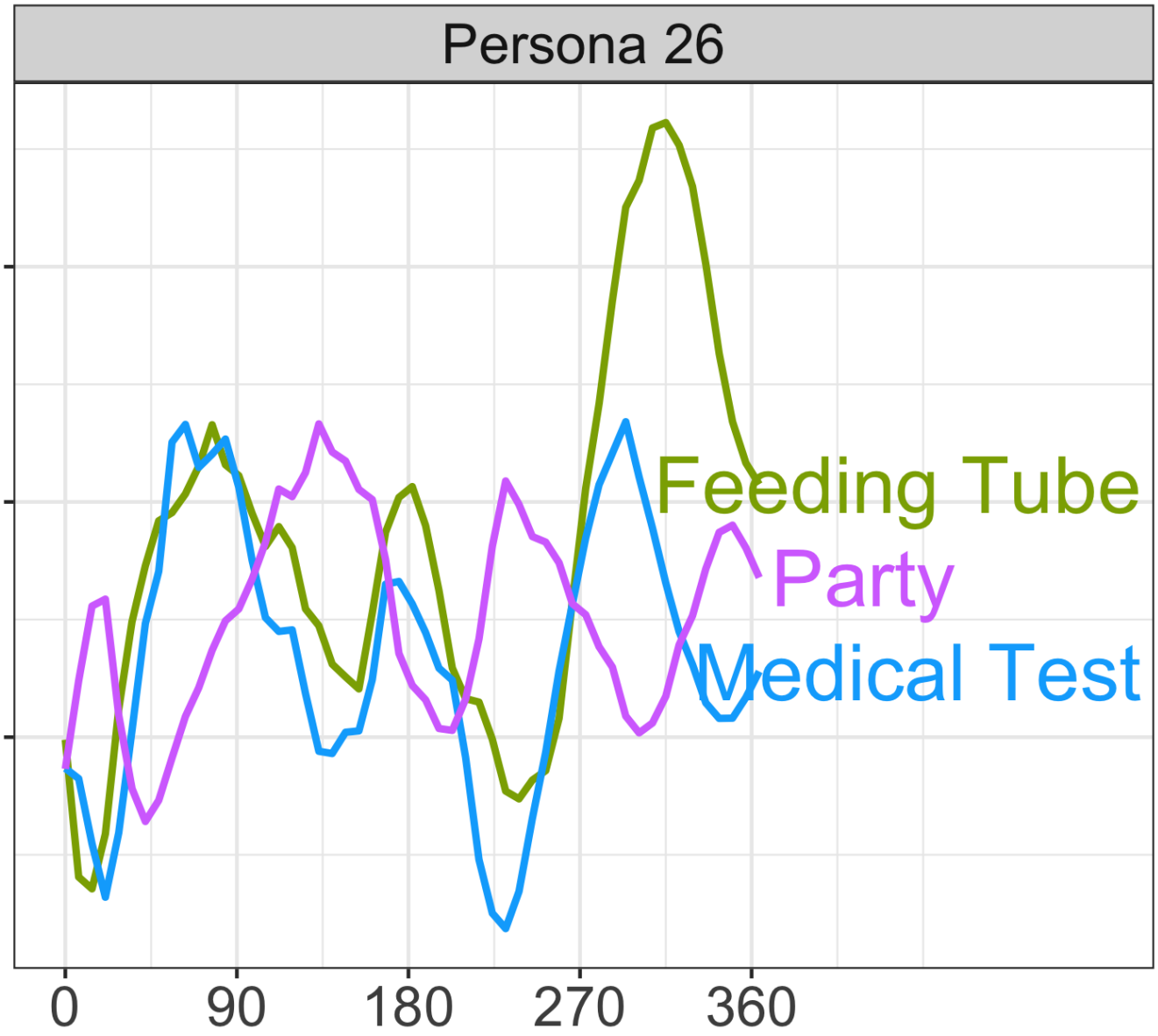



Figure: DAPPER finds personas that reflect common "health journeys" shared by CaringBridge users. Each plot shows how the 3 topics most associated with a persona evolve over time. Topic labels are hand-selected based on top words in each topic.

# Compelling Shared Narratives



# Outline

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

X. Bao, J. Lucas, S. Sachdeva, and R. Grosse. Regularized linear autoencoders recover the principal components, eventually. *arXiv:2007.06731 [cs, stat]*, July 2020.

D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational Inference: A Review for Statisticians. *Journal of the American Statistical Association*, 112(518): 859–877, 2017.

F. P. Casale, A. V. Dalca, L. Saglietti, J. Listgarten, and N. Fusi. Gaussian Process Prior Variational Autoencoders. *NIPS*, page 11, 2018.

J. Chen, C. Lu, B. Chenli, J. Zhu, and T. Tian. VFlow: More Expressive Generative Flows with Variational Data Augmentation. *arXiv:2002.09741 [cs, stat]*, Feb. 2020.

X. Chen, D. P. Kingma, T. Salimans, Y. Duan, P. Dhariwal, J. Schulman, I. Sutskever, and P. Abbeel. Variational Lossy Autoencoder. *ICLR*, 2017.

N. De Cao, I. Titov, and W. Aziz. Block Neural Autoregressive Flow. *35th Conference on Uncertainty in Artificial Intelligence (UAI19)*, Apr. 2019.

L. Dinh, D. Krueger, and Y. Bengio. NICE: Non-linear Independent Components Estimation. In *International Conference on Learning Representations*, 2015.

L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density Estimation using Real NVP. In *International Conference on Learning Representations*, 2017.

D. Dua and E. K. Taniskidou. UCI machine learning repository. https://archive.ics.uci.edu/ml/index.php, 2017.

C. Durkan, A. Bekasov, I. Murray, and G. Papamakarios. Neural Spline Flows. In *Advances in Neural Information Processing Systems*, 2019.

J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: A statistical view of boosting. *The annals of statistics*, 28(2):337–407, 2000.

J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

J. H. Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378, 2002.

Y. Gal, J. Hron, and A. Kendall. Concrete Dropout. In *Advances in Neural Information Processing Systems*, 2017.

R. Giaquinto and A. Banerjee. Gradient Boosted Normalizing Flows. In *Advances in Neural Information Processing Systems*, June 2020.

I. Goodfellow, J. Pouget-Abadie, and M. Mirza. Generative Adversarial Networks. *arXiv preprint arXiv: \ldots*, pages 1–9, 2014.

F. Guo, X. Wang, K. Fan, T. Broderick, and D. B. Dunson. Boosting Variational Inference. *arXiv:1611.05559 [cs, stat]*, Nov. 2016.

D. Ha and J. Schmidhuber. World Models. *arXiv:1803.10122 [cs, stat]*, Mar. 2018. doi: 10.5281/zenodo.1207631.

J. Ho, X. Chen, A. Srinivas, Y. Duan, and P. Abbeel. Flow++: Improving Flow-Based Generative Models with Variational Dequantization and Architecture Design. In *International Conference on Machine Learning*, 2019.

Z. Hu, Z. Yang, R. Salakhutdinov, and E. P. Xing. On Unifying Deep Generative Models. In *ICLR*, pages 1–19, 2018.

C.-W. Huang, D. Krueger, A. Lacoste, and A. Courville. Neural Autoregressive Flows. *ICML*, page 10, 2018a.

C.-W. Huang, S. Tan, A. Lacoste, and A. Courville. Improving Explorability in Variational Inference with Annealed Variational Objectives. In *Advances in Neural Information Processing Systems*, page 11, Montréal, Canada, 2018b.

C.-W. Huang, L. Dinh, and A. Courville. Augmented normalizing flows: Bridging the gap between generative flows and latent variable models. *arXiv:2002.07101 [cs, stat]*, Feb. 2020.

M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. Introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

D. P. Kingma and P. Dhariwal. Glow: Generative Flow with Invertible 1x1 Convolutions. In *Advances in Neural Information Processing Systems*, Montréal, Canada, July 2018.

D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, pages 1–14, Dec. 2014.

D. P. Kingma and M. Welling. An Introduction to Variational Autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, June 2019.

D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. Improving Variational Inference with Inverse Autoregressive Flow. *NIPS*, 2016.

P. Kirichenko, P. Izmailov, and A. G. Wilson. Why normalizing flows fail to detect out-of-distribution data. *arXiv:2006.08545 [cs, stat]*, June 2020.

C. Louizos and M. Welling. Multiplicative Normalizing Flows for Variational Bayesian Neural Networks. *arXiv:1703.01961 [cs, stat]*, Mar. 2017.

J. Lucas, G. Tucker, R. Grosse, and M. Norouzi. Don't Blame the ELBO! A Linear VAE Perspective on Posterior Collapse. *arXiv:1911.02469 [cs, stat]*, Nov. 2019.

A. Lugmayr, M. Danelljan, L. Van Gool, and R. Timofte. SRFlow: Learning the Super-Resolution Space with Normalizing Flow. In *European Conference on Computer Vision*, 2020.

D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 2, pages 416–423,

Vancouver, BC, Canada, 2001. IEEE Comput. Soc. ISBN 978-0-7695-1143-6. doi: 10.1109/ICCV.2001.937655.

L. Mason, J. Baxter, P. L. Bartlett, and M. R. Frean. Boosting algorithms as gradient descent. In *Advances in Neural Information Processing Systems*, pages 512–518, 1999.

A. C. Miller, N. Foti, and R. P. Adams. Variational Boosting: Iteratively Refining Posterior Approximations. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 2420–2429. PMLR, 2017.

D. Nielsen and O. Winther. Closing the Dequantization Gap: PixelCNN as a Single-Layer Flow. *arXiv:2002.02547 [cs, stat]*, Oct. 2020.

D. Nielsen, P. Jaini, E. Hoogeboom, O. Winther, and M. Welling. SurVAE flows: Surjections to bridge the gap between VAEs and flows. *arXiv:2007.02731 [cs, stat]*, July 2020.

G. Papamakarios, T. Pavlakou, and I. Murray. Masked Autoregressive Flow for Density Estimation. In *Advances in Neural Information Processing Systems*, 2017.

D. J. Rezende and S. Mohamed. Variational Inference with Normalizing Flows. *ICML*, 2015.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. *arXiv:1401.4082 [cs, stat]*, Jan. 2014.

M. Rolinek, D. Zietlow, and G. Martius. Variational Autoencoders Pursue PCA Directions (by Accident). In *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12406–12415, 2019.

S. Rosset and E. Segal. Boosting density estimation. In *Advances in Neural Information Processing Systems*, page 8, 2002.

E. G. Tabak and C. V. Turner. A Family of Nonparametric Density Estimation Algorithms. *Communications on Pure and Applied Mathematics*, 66(2): 145–164, Feb. 2013. ISSN 00103640. doi: $10.1002/\mathrm{cpa}.21423$.

E. G. Tabak and E. Vanden-Eijnden. Density estimation by dual ascent of the log-likelihood. *Communications in Mathematical Sciences*, 8(1):217–233, 2010. ISSN 15396746, 19450796. doi: $10.4310/\mathrm{CMS}.2010.\mathrm{v8.n1.a11}$.

M. E. Tipping and C. M. Bishop. Probabilistic Principal Component Analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61 (3):611–622, Aug. 1999. ISSN 1369-7412, 1467-9868. doi: $10.1111/1467\text{-}9868.00196$.

J. Tomczak and M. Welling. VAE with a VampPrior. In *International Conference on Artifi- Cial Intelligence and Statistics (AISTATS)*, volume 84, Lanzarote, Spain, 2018.

J. M. Tomczak and M. Welling. Improving Variational Auto-Encoders using Householder Flow. In *Bayesian Deep Learning Workshop (NIPS 2016)*, 2016.

J. M. Tomczak and M. Welling. Improving Variational Auto-Encoders using convex combination linear Inverse Autoregressive Flow. *arXiv:1706.02326 [stat]*, June 2017.

R. van den Berg, Leonard Hasenclever, Jakub M. Tomczak, and Max Welling. Sylvester Normalizing Flows for Variational Inference. *Uncertainty in Artificial Intelligence (UAI)*, 2018.

M. J. Wainwright and M. I. Jordan. Graphical Models, Exponential Families, and Variational Inference. *Foundations and Trends® in Machine Learning*, 1(1–2): 1–305, 2007.

C. Winkler, D. Worrall, E. Hoogeboom, and M. Welling. Learning likelihoods with conditional normalizing flows. *arXiv:1912.00042 [cs, stat]*, Nov. 2019.

Z. M. Ziegler and A. M. Rush. Latent Normalizing Flows for Discrete Sequences. In *Advances in Neural Information Processing Systems*, 2019.