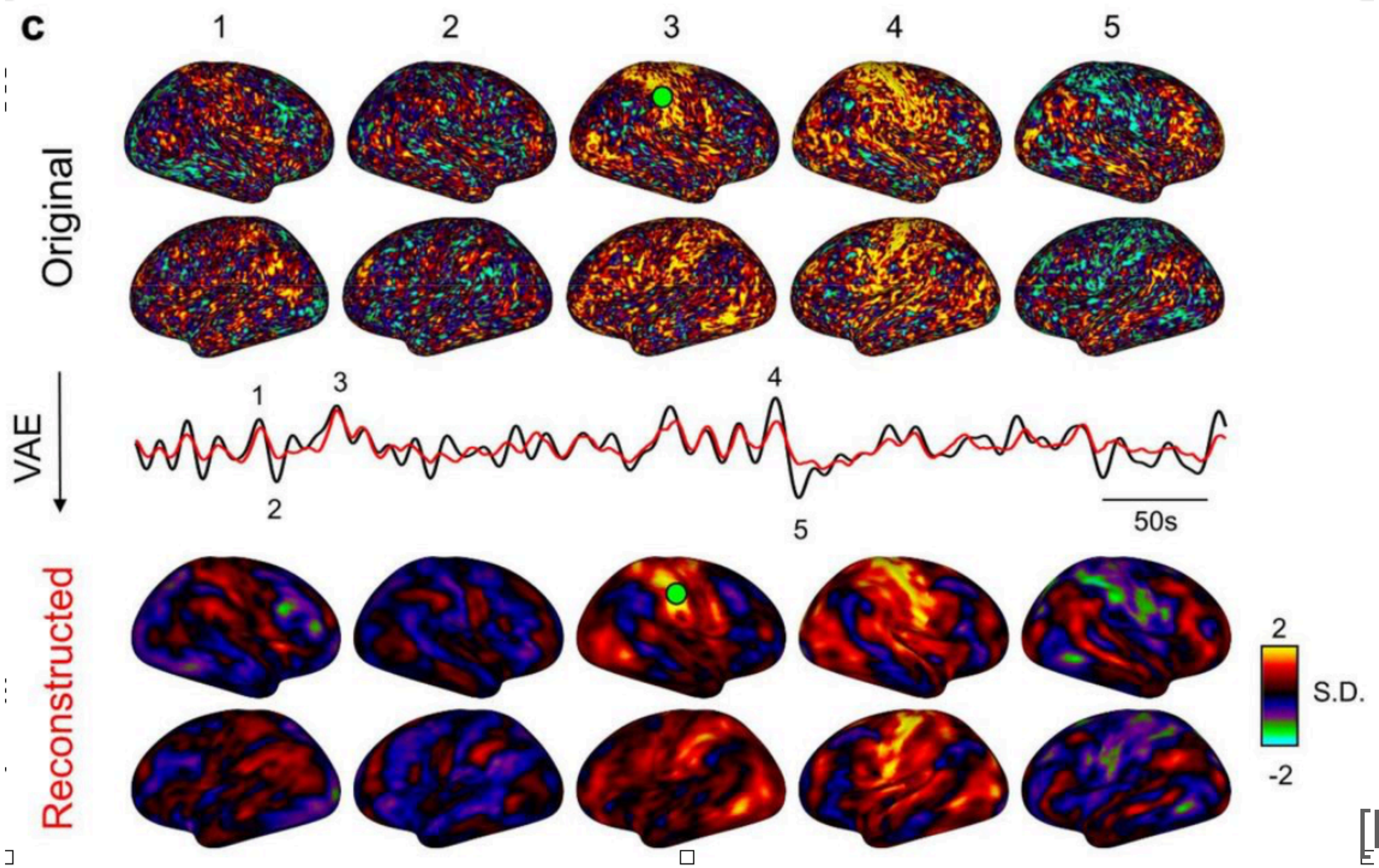
Posterior Collapse and Latent Variable Non-identifiability

Yixin Wang

Joint work with David Blei and John Cunningham

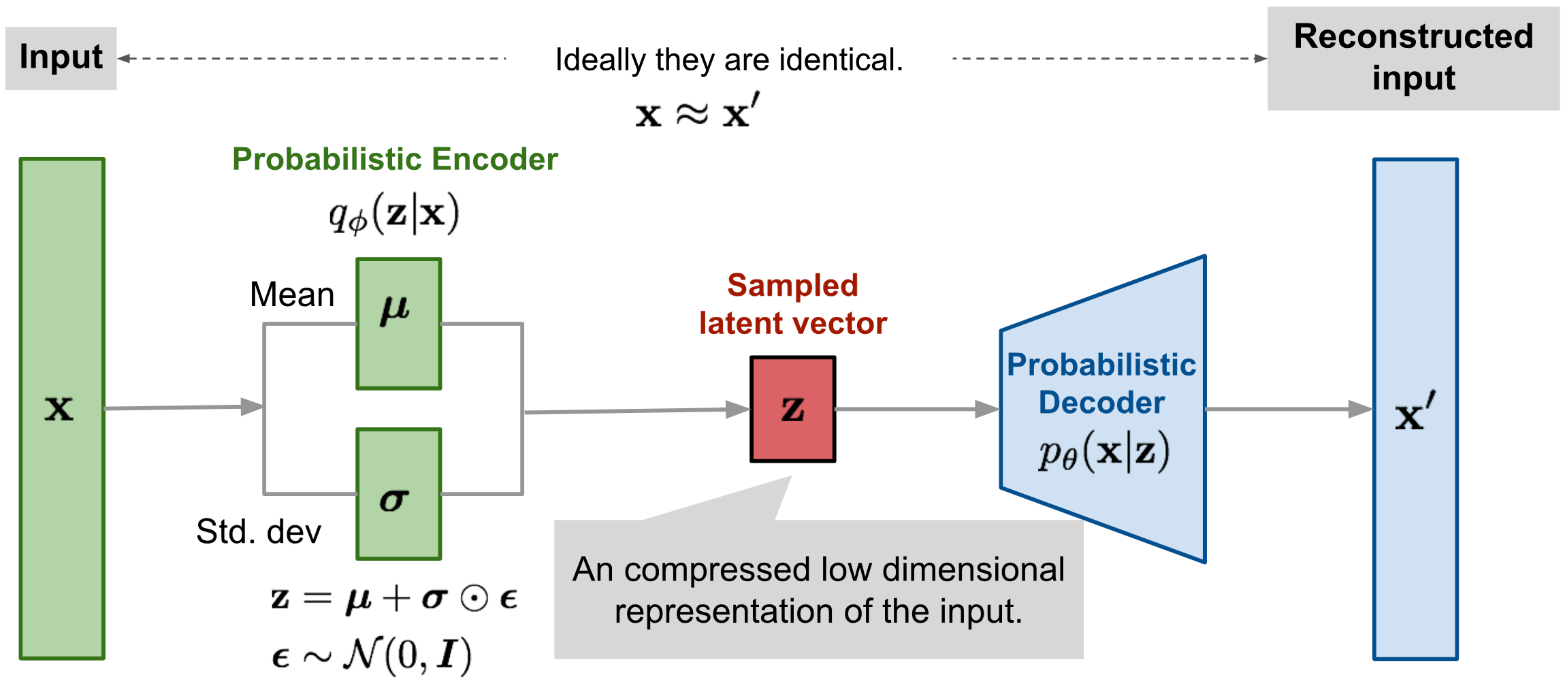
The Power of Deep Generative Models



[Kim et al. (Biorxiv, 2020)]

- **Unsupervised representation learning**: Extract meaningful latent variable
- Density estimation; reconstruct input; generate new samples

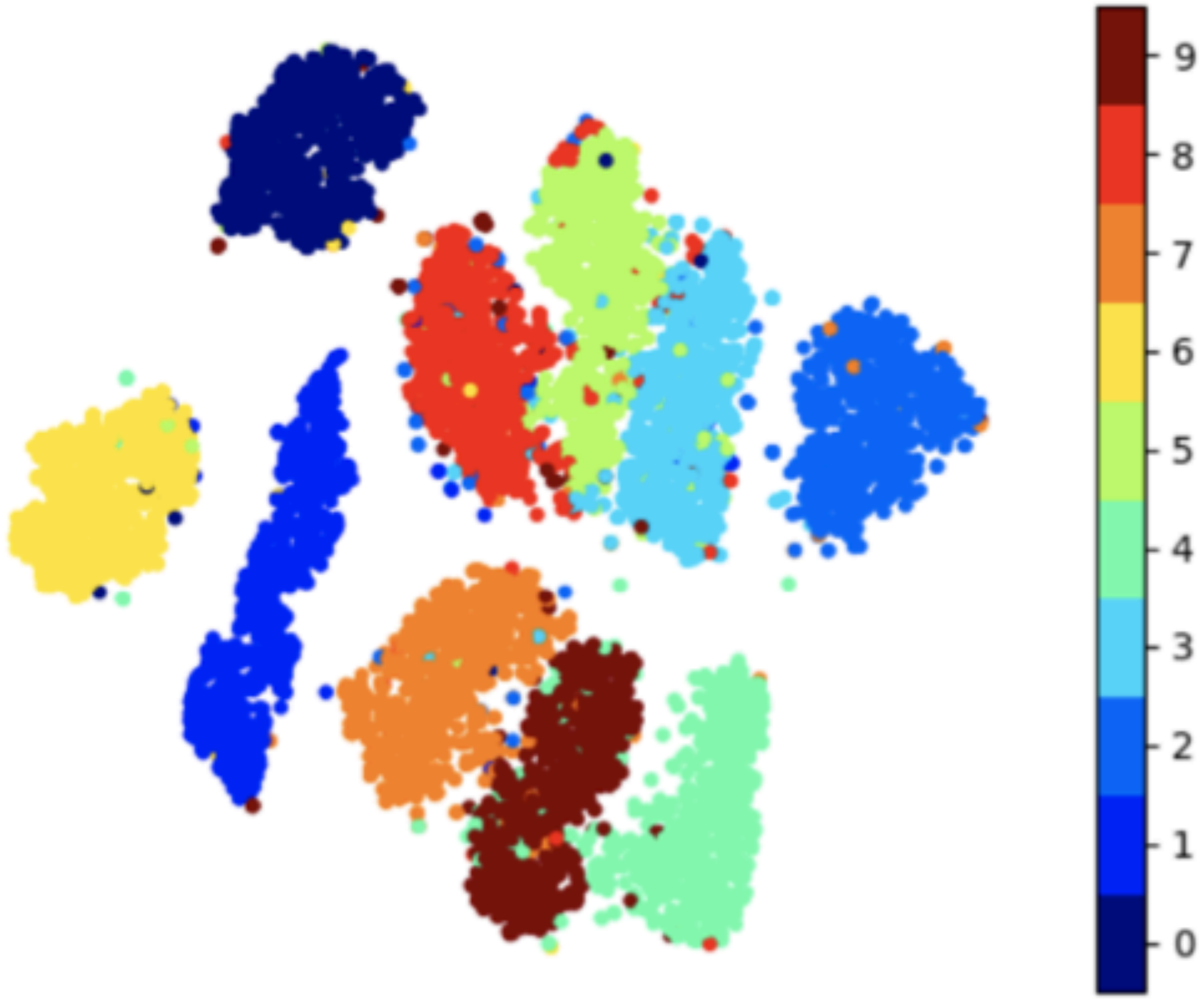
Variational Autoencoders



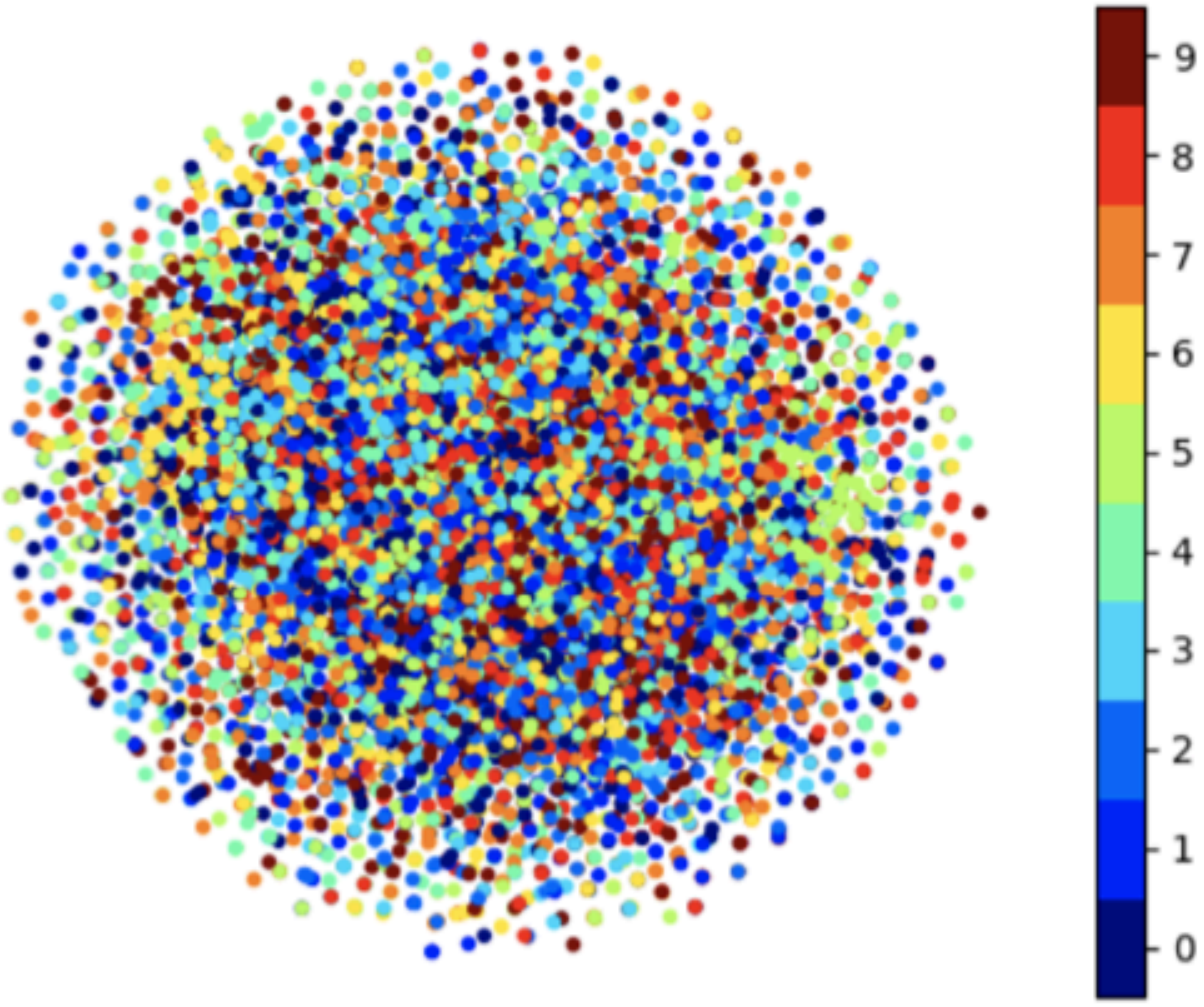
$$z_i \sim p(z_i), \qquad x_i \,|\, z_i \sim p(x_i \,|\, z_i\,; \theta) = \mathrm{EF}(x_i \,|\, f_\theta(z_i))$$

Weng (Lil'Log, 2018)

Posterior Collapse



Ideal



Reality

Jha et al. (CVPR, 2018)

- The model **fits** well: Good predictive likelihood; Generate good new samples.
- **Posterior is equal to the prior**: Non-informative; useless as representations.

We have blamed many aspects of VAE for collapse

- Decoder is too powerful (Li+ 2019)
- The prior biases us (Higgins+ 2016)
- Approximate inference (Bowman+ 2015; Kingma+ 2016; Sønderby+ 2016)
- Training procedure; the order of parameter updates (He+ 2019)
- Local minima of optimization (Lucas+ 2019)
- Information preference (Chen+ 2016)

We have invented many ways to try to fix it

- Beta VAE (Higgins+ 2016)
- VampPrior (Tomczak+ 2017)
- Lagging inference (He+ 2019)
- Semi-amortized training (Kim+ 2018)
- Threshold the KL to prior (Li+ 2019)

Posterior Collapse and Latent Variable Identifiability

- What is it? Why it happens? Is it new?
- Can we fix it? Do we pay a price? Does it work?

Posterior collapse is a problem of latent variable non-identifiability.

Takeaways first

- **Posterior collapse** is a problem of **latent variable non- identifiability**.
- It is **not** specific to the use of neural networks or variational inference algorithms in VAE. Rather, it is an **intrinsic** issue of the model and the dataset.
- We propose a class of **latent-identifiable variational autoencoders (LIDVAE)** via Brenier maps to resolve latent variable non-identifiability and mitigate posterior collapse.
- **Identifiability** used to be mostly of theoretical interest, but it turns out to have important **practical implications** in modern machine learning.

Modeling high-dimensional data with VAE

- A **variational autoencoder (VAE)** assumes each datapoint x_i is generated by the latent variable z_i with parameters θ

$$z_i \sim p(z_i), \qquad x_i \mid z_i \sim p(x_i \mid z_i\,;\,\theta) = \mathrm{EF}(x_i \mid f_\theta(z_i))\,.$$

- Infer θ and posterior $p(z_i \mid x_i; \theta)$ by **maximum (marginal) likelihood** with variational approximation

$$\theta^* = \mathrm{argmax} \quad p(\mathbf{x} \mid \theta),$$

$$q(z_i \mid x_i; \theta) = \mathrm{argmin}_{\mathcal{Q}} \mathrm{KL}(q(z_i \mid x_i; \theta) \,||\, p(z_i \mid x_i; \theta))$$

Examples of Variational Autoencoders

- Variational Autoencoder (VAE)

$$Z_i \sim p(z_i), \qquad X_i \,|\, Z_i \sim p(x_i \,|\, z_i \,;\, \theta),$$

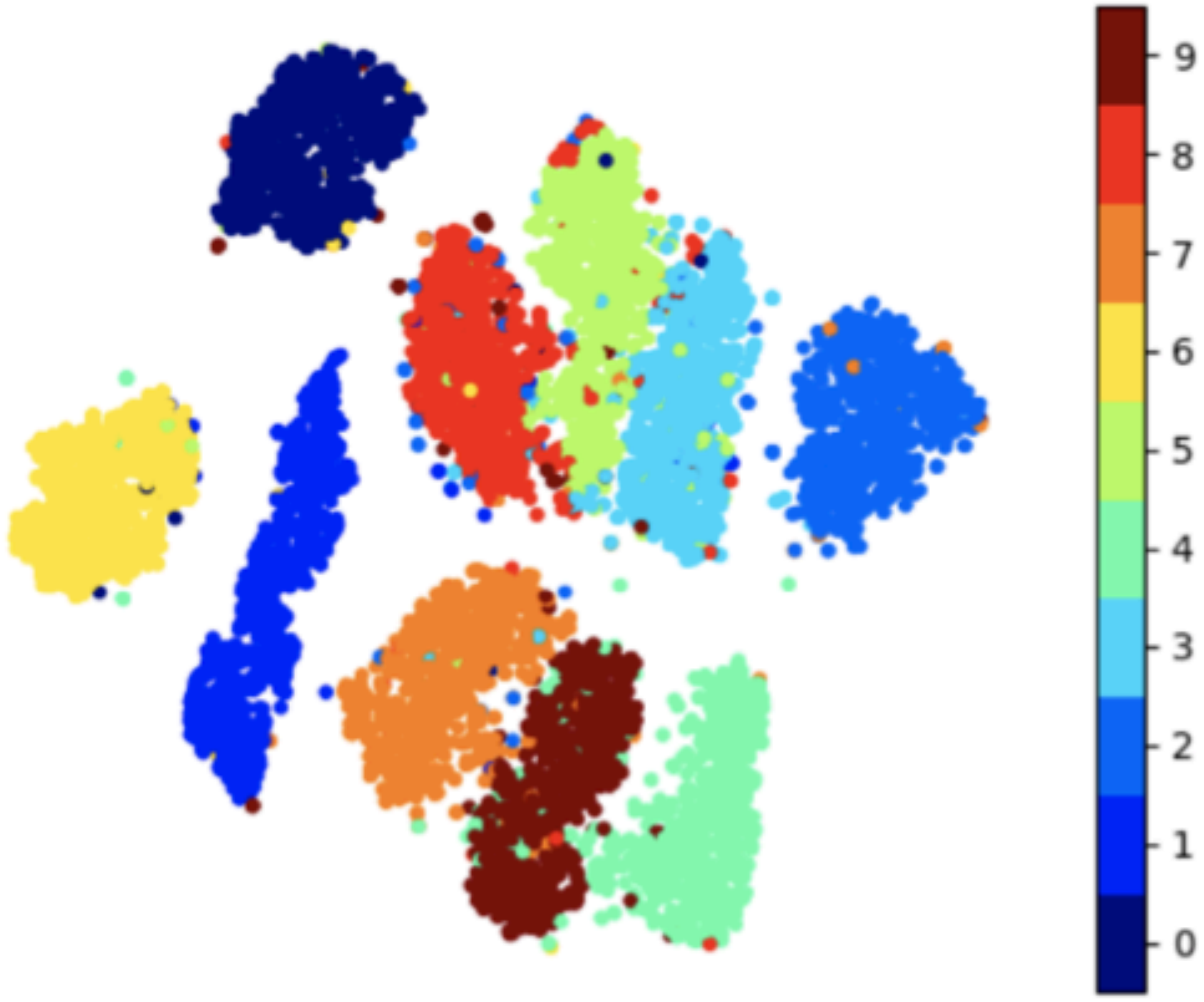
- Example: Gaussian VAE

$$Z_i \sim \mathcal{N}(0, I_K), \qquad X_i \,|\, Z_i \sim \mathcal{N}(f_\theta(z_i), \sigma_\theta^2 \cdot I_m).$$

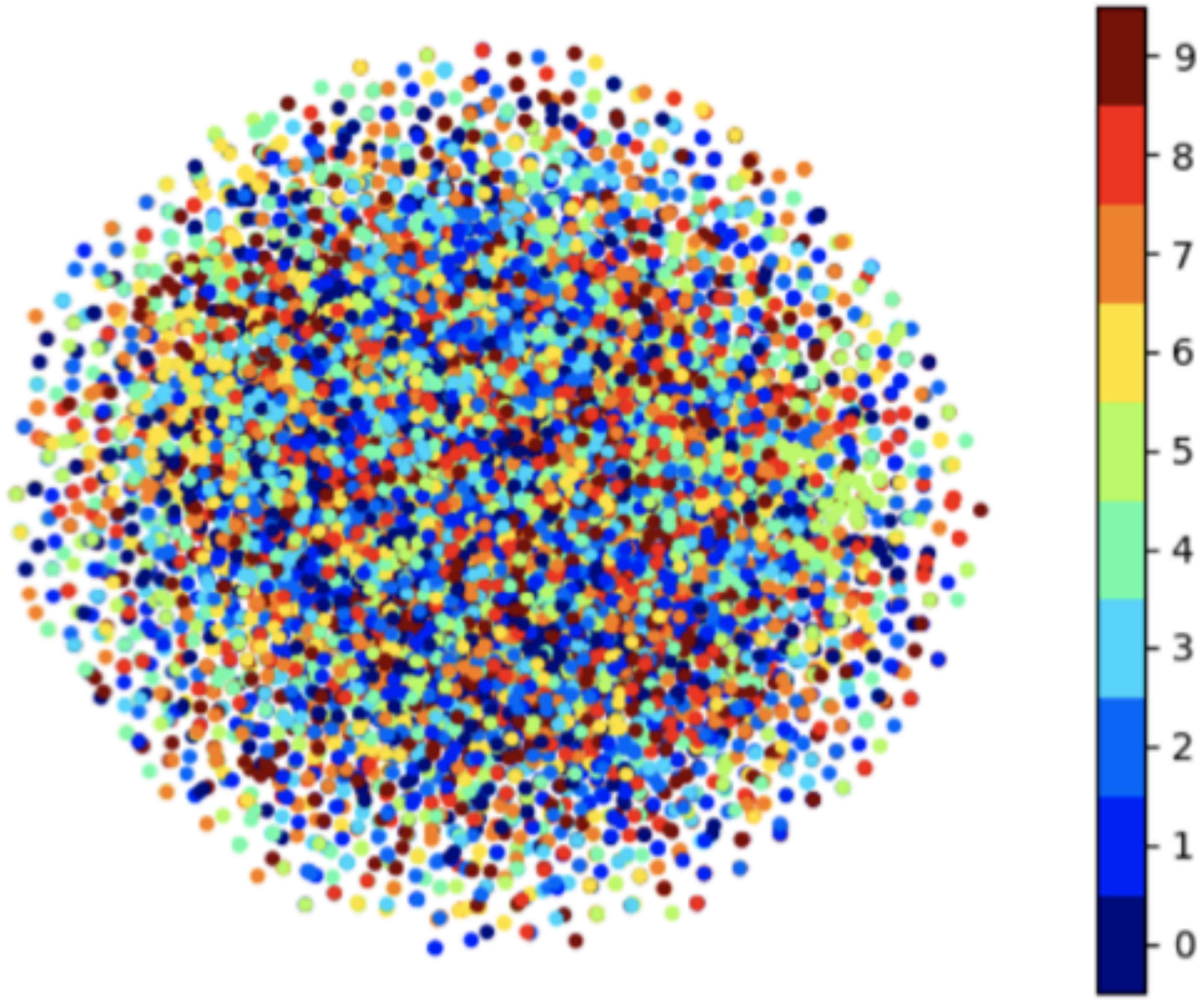
- Example: Bernoulli mixture VAE

$$Z_i \sim \mathrm{Categorical}(1/K), \qquad X_i \,|\, Z_i \sim \mathrm{Bernoulli}(\mathrm{sigmoid}(f_\theta(\mathcal{N}(\mu_{z_i}, \Sigma_{z_i})))),$$

Posterior Collapse: What is it?



Ideal



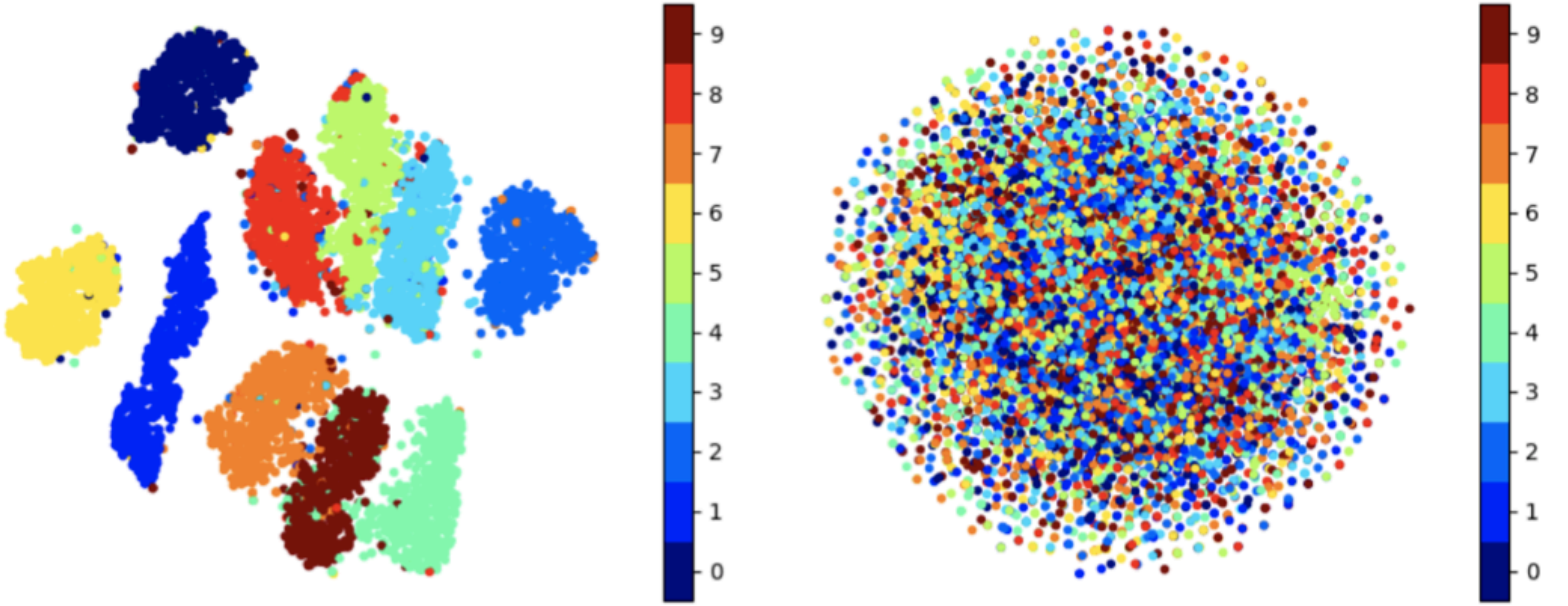
Reality

Jha et al. (CVPR, 2018)

- **Posterior collapse** is a phenomenon where the posterior of the latents in a VAE is equal to its uninformative prior

$$p(\boldsymbol{z} \mid \boldsymbol{x} ; \theta^{*}) = p(\boldsymbol{z}).$$

Posterior Collapse: What are the essential conditions?



- Let’s abstract **away** approximate inference
    - Consider the ideal case **where the variational approximation is exact**.
- Posterior collapse can happen **in the absence of** variational approximation.

Latent Variable Non-identifiability

- **Definition (Latent variable non-identifiability)**
  - Given a likelihood function $p(\mathbf{x}, \mathbf{z}; \theta)$, a parameter value $\theta = \hat{\theta}$, and a dataset $\mathbf{x} = (x_1, \ldots, x_n)$, the latent variable $\mathbf{z}$ is **non-identifiable** if

$$p(\mathbf{x} \,|\, \mathbf{z} = \tilde{\mathbf{z}}'; \hat{\theta}) = p(\mathbf{x} \,|\, \mathbf{z} = \tilde{\mathbf{z}}; \hat{\theta}) \qquad \forall \tilde{\mathbf{z}}', \tilde{\mathbf{z}} \in \mathcal{Z} \; .$$

Posterior Collapse iff Latent Variable Non-identifiability

- **Theorem (Latent variable non-identifiability $\Leftrightarrow$ Posterior collapse)**
  - The latent variables $\mathbf{z}$ are **non-identifiable** at $\hat{\theta}$ **if and only if** the posterior of $\mathbf{z}$ **collapses**, $p(\mathbf{z} \mid \mathbf{x}; \hat{\theta}) = p(\mathbf{z})$.
- **Proof:** One line proof due to the Bayes rule
  - $p(\mathbf{z} \mid \mathbf{x}; \hat{\theta}) \propto p(\mathbf{z})p(\mathbf{x} \mid \mathbf{z}; \hat{\theta}) = p(\mathbf{z})p(\mathbf{x}; \hat{\theta}) \propto p(\mathbf{z})$

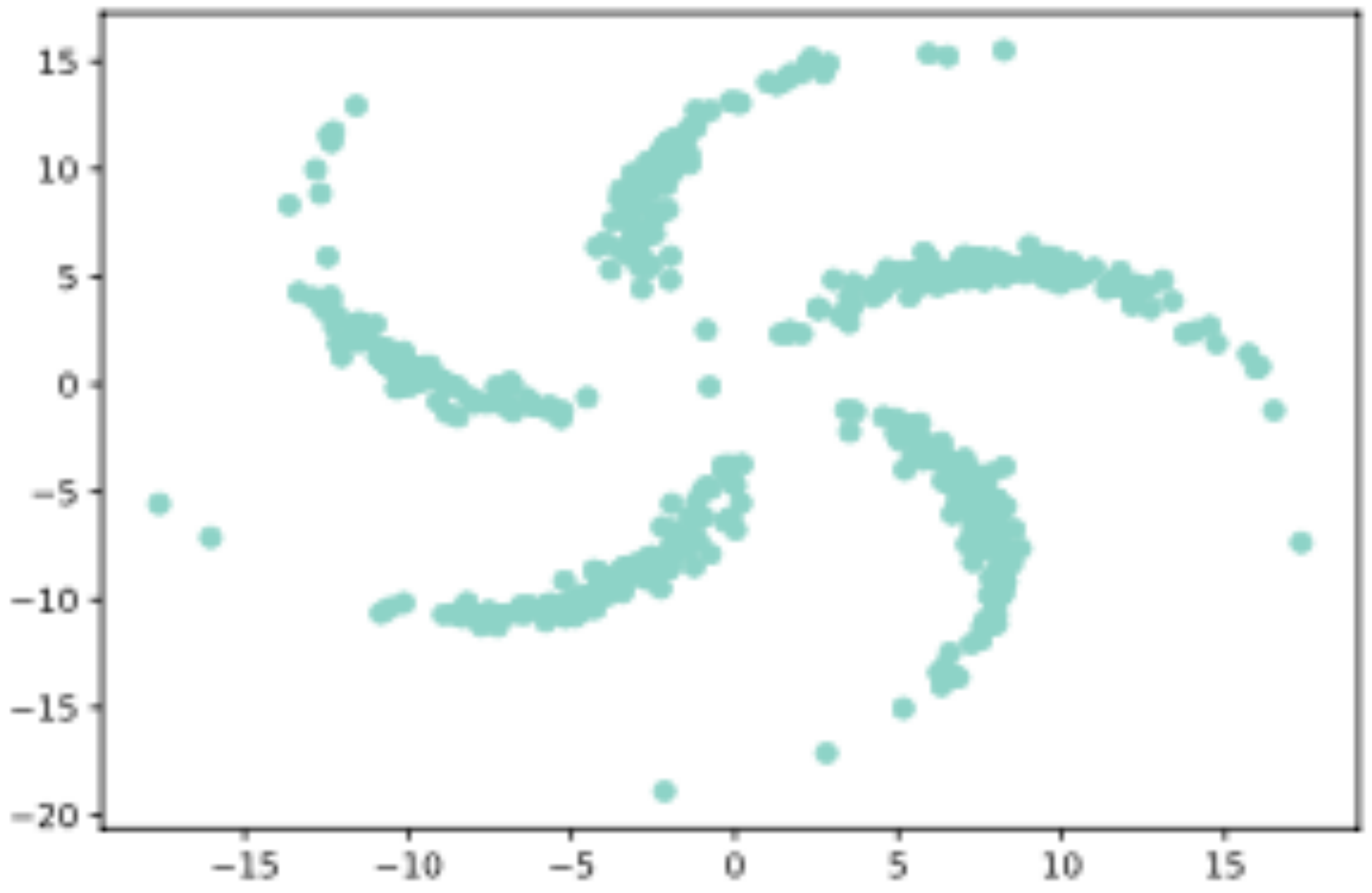
Posterior Collapse iff Latent Variable Non-identifiability

- It happens with exact inference.
- It happens in classical not-so-flexible models.
- It doesn't have to involve neural network.
- It happens with global optima.
- It happens with both local and global latent variables.

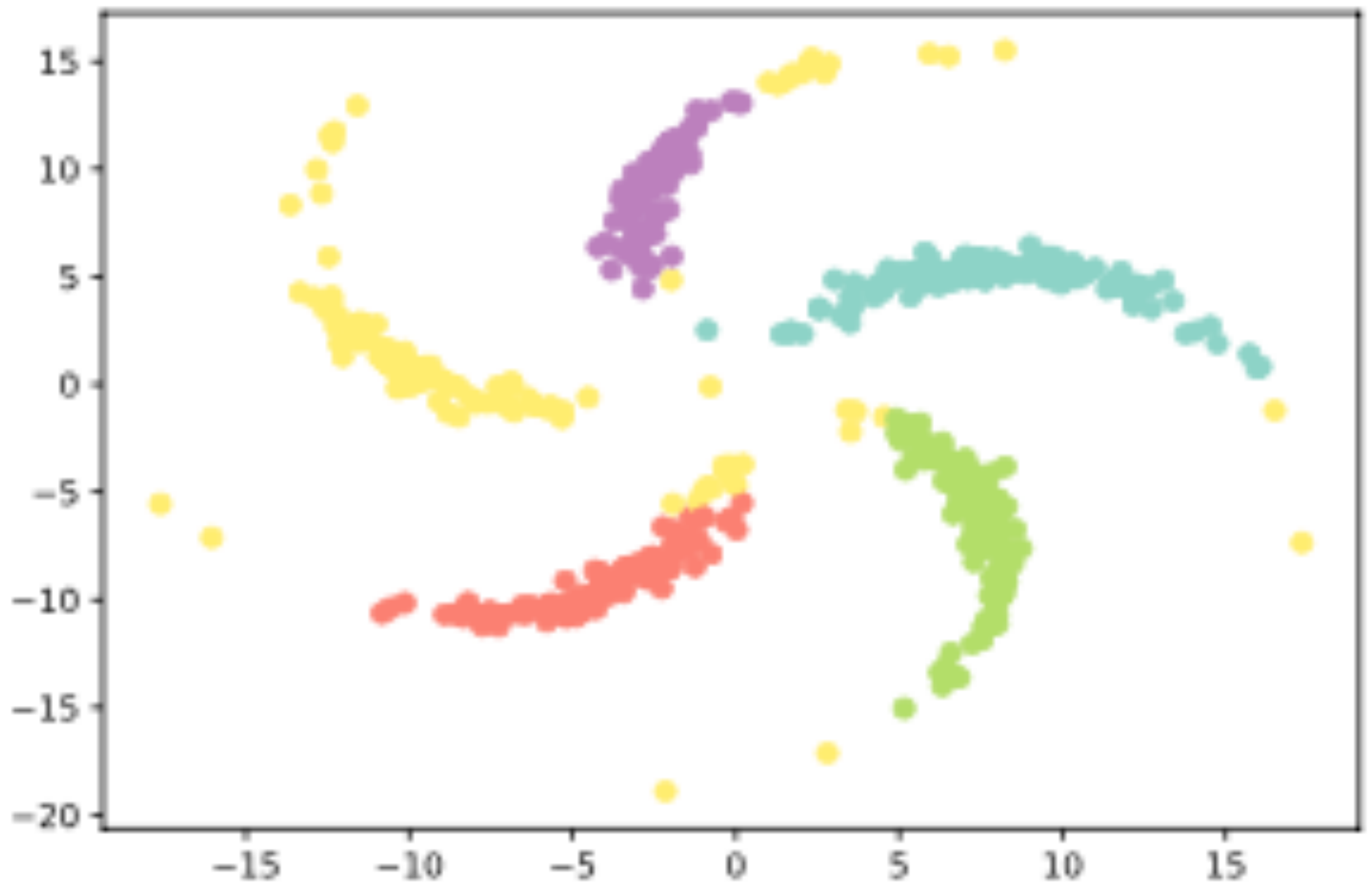
Posterior Collapse in Gaussian Mixture VAE

- Gaussian Mixture VAE (GMVAE)

$$p(z_i) = \text{Categorical}(1/K), \qquad p(w_i \,|\, z_i) = \mathcal{N}(\mu_{z_i}, \Sigma_{z_i}), \qquad p(x_i \,|\, w_i; \theta) = \mathcal{N}(f_\theta(w_i), \sigma^2 \cdot I_m)$$



Latent variable is non-identifiable

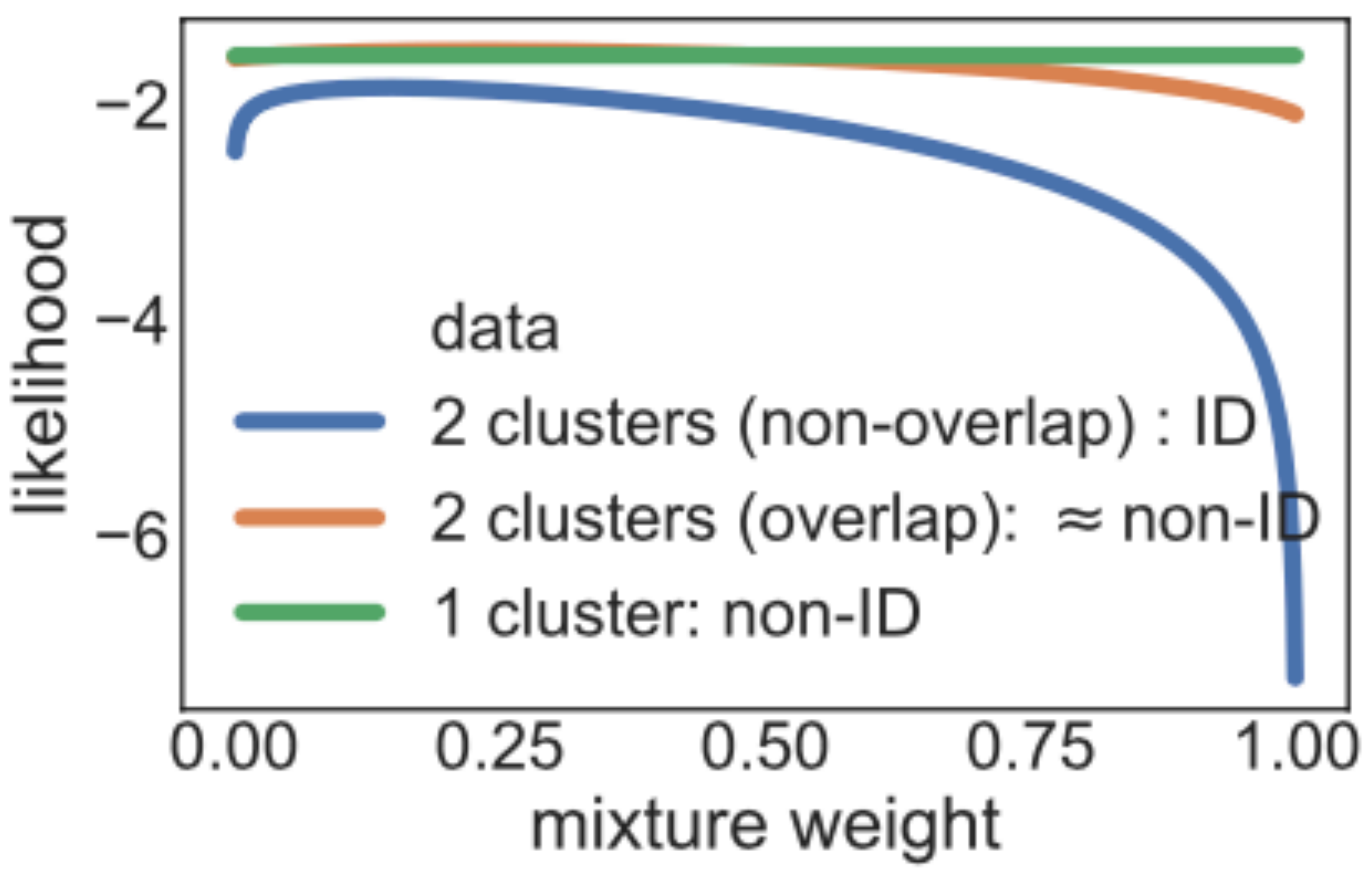


Latent variable is identifiable

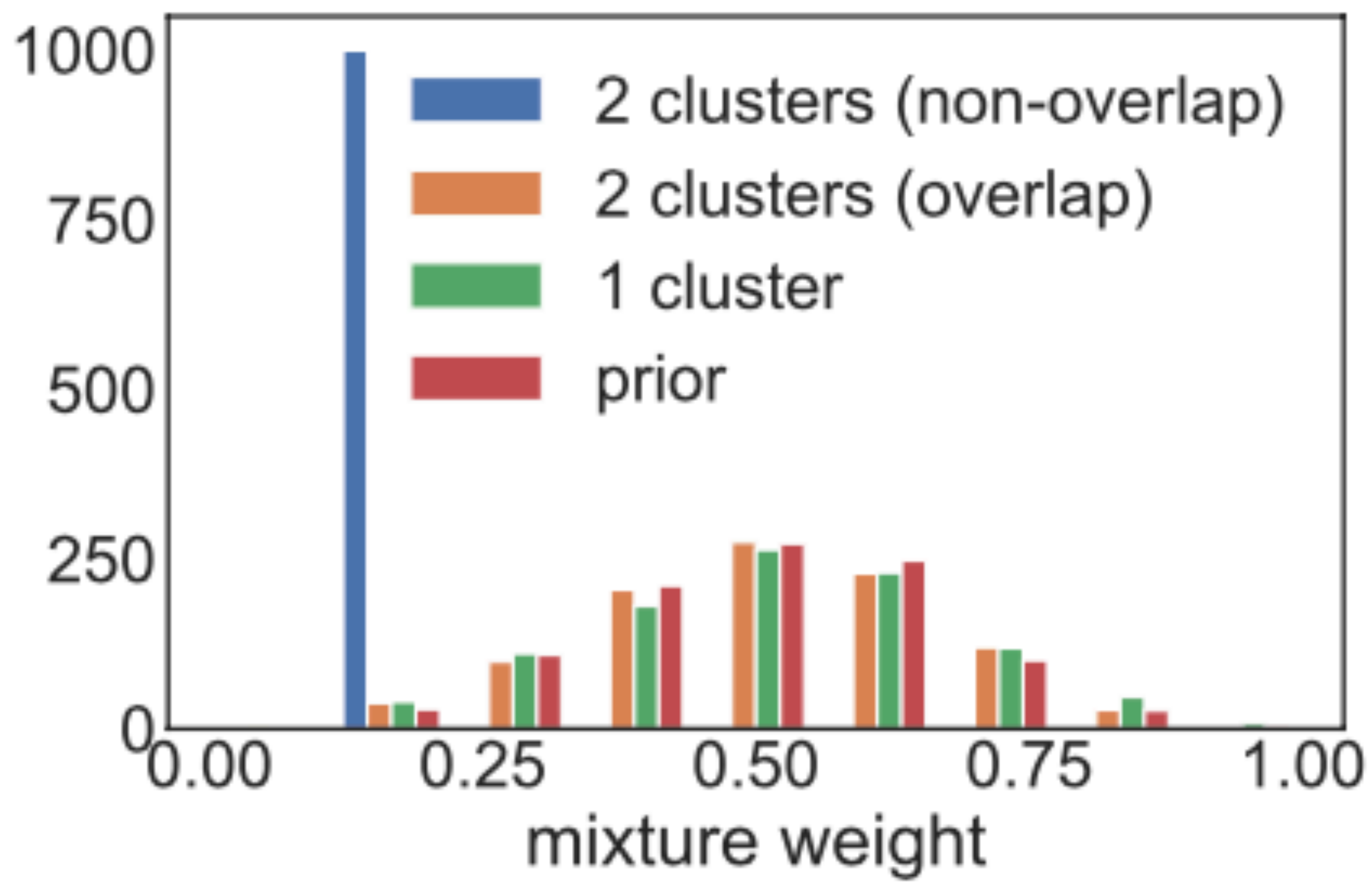
Posterior Collapse in Gaussian Mixture Model

- Gaussian mixture model (GMM)

$$p(\alpha) = \mathrm{Beta}(\alpha; 5{,}5), \quad p(x_i \mid \alpha; \theta) = \alpha \cdot \mathcal{N}(x_i; \mu_1, \sigma_1^2) + (1-\alpha) \cdot \mathcal{N}(x_i; \mu_2, \sigma_2^2)$$



(a) Likelihood function



(b) Posterior histogram

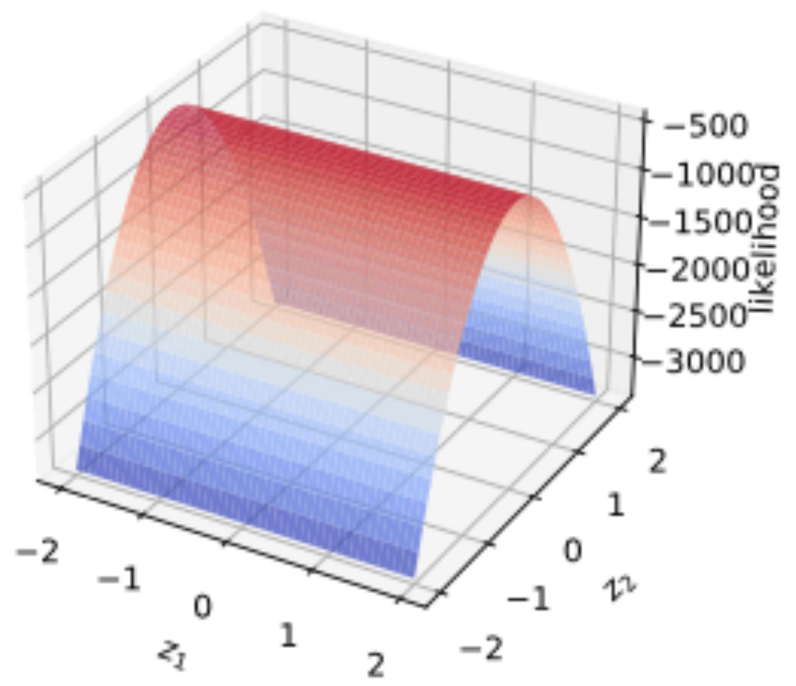
Posterior Collapse in Probabilistic PCA

- Probabilistic PCA (PPCA)

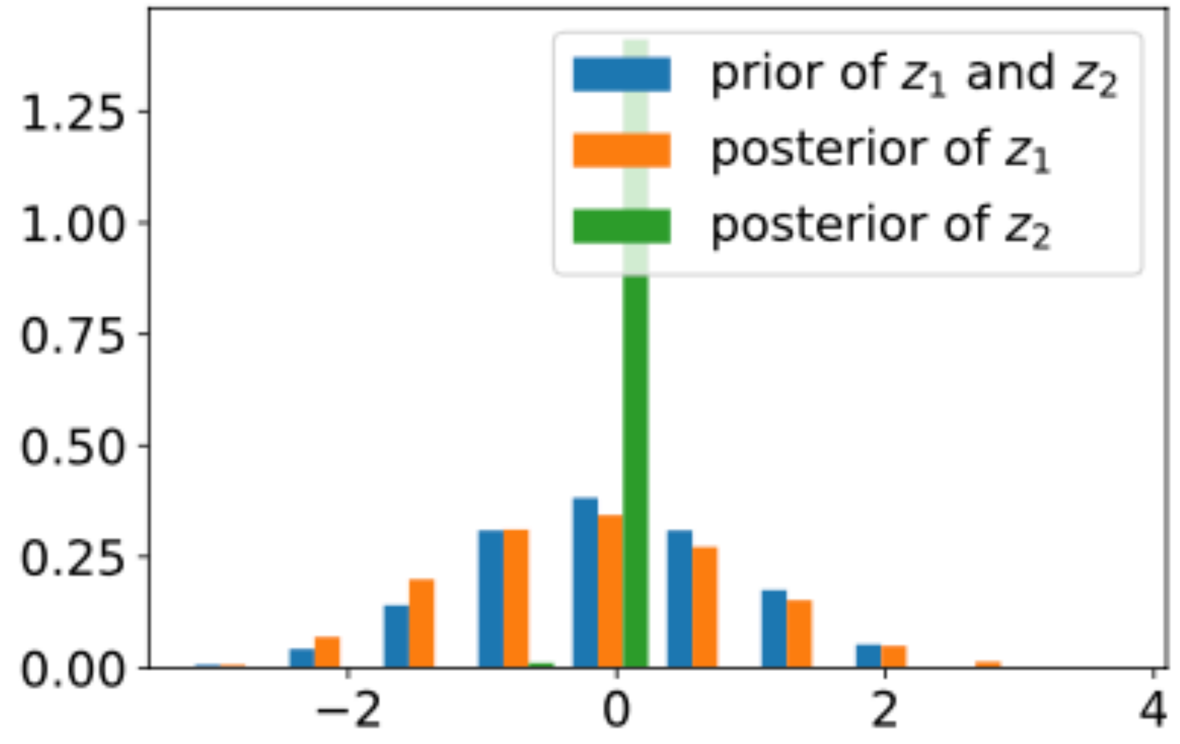
$$p(z_i) = \mathcal{N}(z_i; 0, I_2),$$

$$p(x_i \,|\, z_i; \theta) = \mathcal{N}(x_i; z_i^\top w, \sigma^2 \cdot I_5)$$

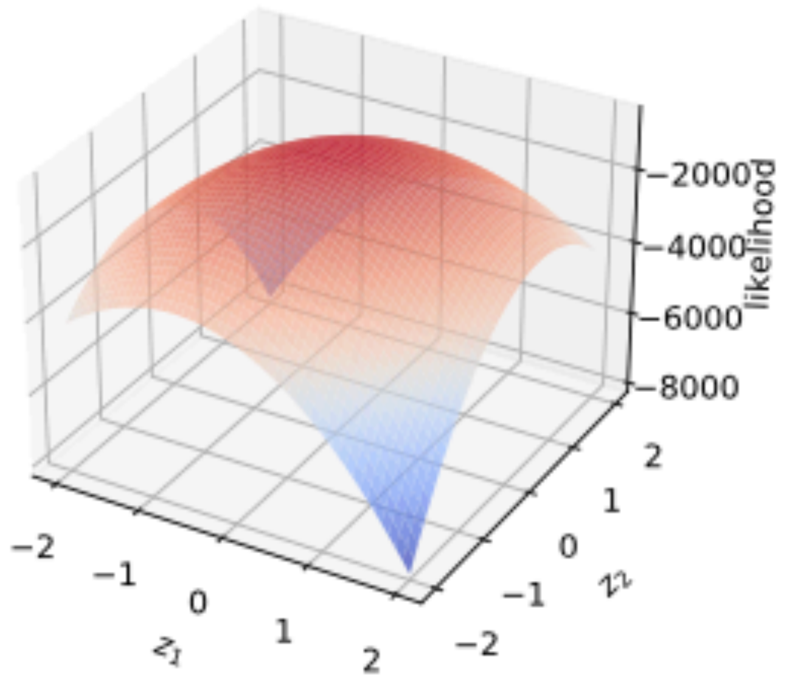
- (Top): z_1 non-identifiable
- (Bottom): z_1 identifiable



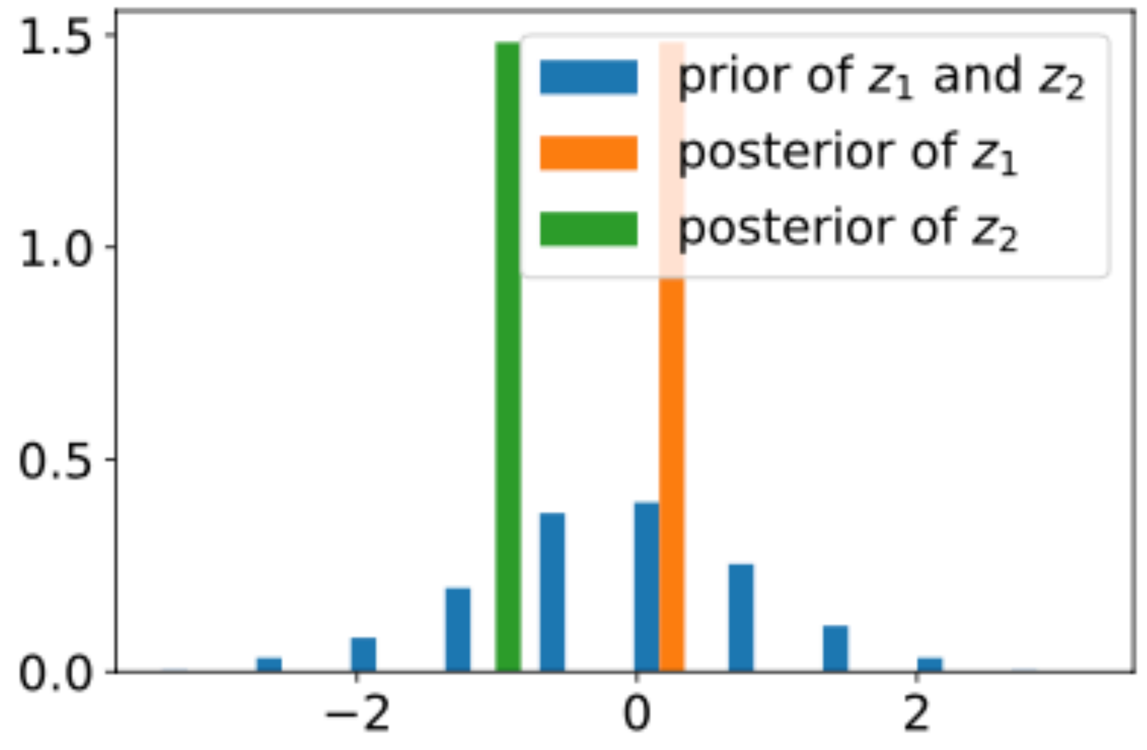
(a) Likelihood (1D PPCA)



(b) Posterior (1D PPCA)

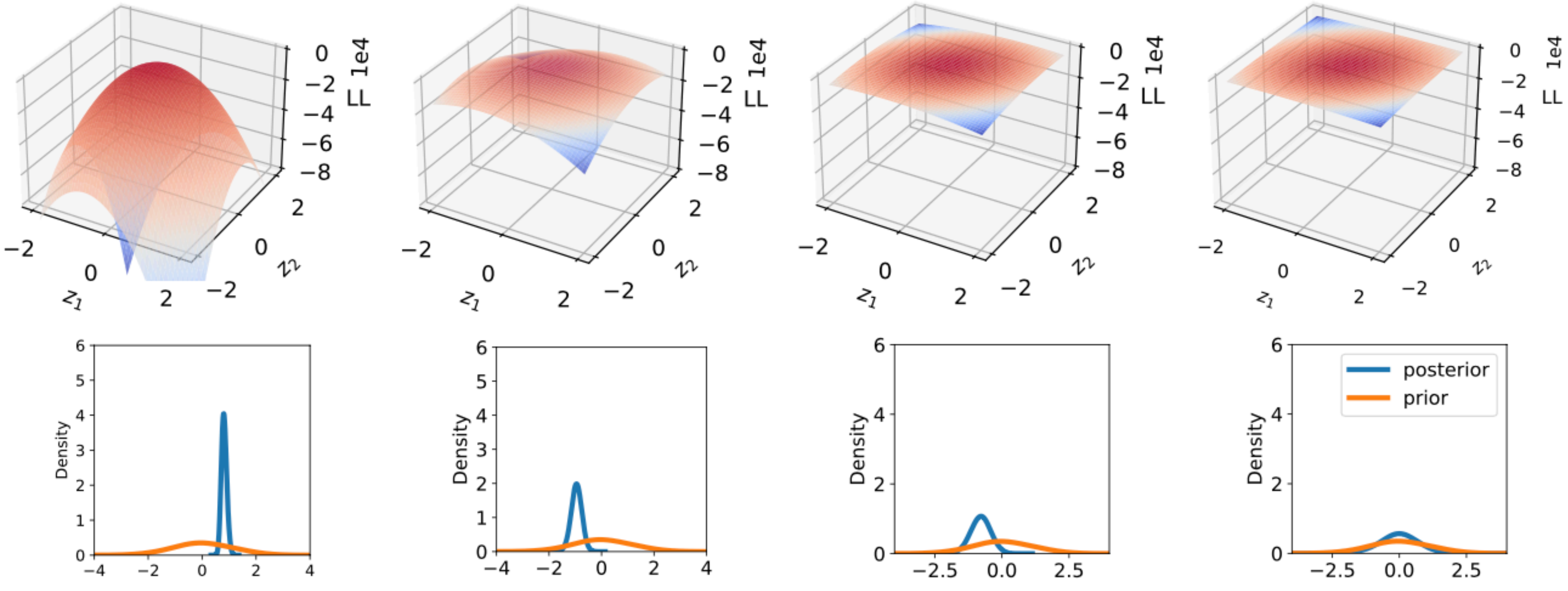


(c) Likelihood (2D PPCA)



(d) Posterior (2D PPCA)

Posterior Collapse in Probabilistic PCA



(a) $\sigma = 0.2$ **(b)** $\sigma = 0.5$ **(c)** $\sigma = 1.0$ **(d)** $\sigma = 1.5$

- Probabilistic PCA (PPCA) $p(z_i) = \mathcal{N}(z_i; 0, I_2), \quad p(x_i \mid z_i; \theta) = \mathcal{N}(x_i; z_i^\top w, \sigma^2 \cdot I_5)$
- The latent variable becomes closer to non-identifiable with larger σ
- The posterior collapses more.

Posterior Collapse: Can we fix it?

- Make latent variables **identifiable** in VAE.
- A variational autoencoder (VAE) assumes each datapoint x_i is generated by the latent variable z_i,

$$x_i \sim p(z_i), \qquad x_i \,|\, z_i \sim p(x_i \,|\, z_i\,;\,\theta) = \mathrm{EF}(x_i \,|\, f_\theta(z_i))\,.$$

- Constructing **latent-identifiable VAE** thus amounts to constructing an **injective likelihood function** for VAE.
  - The construction is based on a few building blocks of linear and nonlinear injective functions, then composed into an injective likelihood $p(x_i \,|\, z_i; \theta)$ mapping from $\mathscr{Z}^d$ to $\mathscr{X}^m$.

The building blocks of LIDVAE: Injective functions

- **Linear injective functions**
  - Left multiplication by matrix $\beta^\top$ where β has **full column rank**
- **Nonlinear injective function**
  - **Brenier map (aka monotone transport map)**: gradient of a convex function
    - Guaranteed to be bijective: derivative is the Hessian of a convex function (positive semidefinite and has a nonnegative determinant)
    - Parametrizable by neural networks using input convex neural networks (ICNN)

Latent-Identifiable VAE (LIDVAE)

- We construct **injective likelihoods** for LIDVAE by **composing injective functions.**
- Vanilla VAE
  $z_i \sim p(z_i), \qquad x_i \mid z_i \sim p(x_i \mid z_i\,;\,\theta) = \mathrm{EF}(x_i \mid f_\theta(z_i))\,.$
- **Latent-Identifiable VAE**
  $z_i \sim p(z_i), \qquad x_i \mid z_i \sim p(x_i \mid z_i\,;\,\theta) = \mathrm{EF}(x_i \mid g_{2,\theta}(\beta^\top g_{1,\theta}(z_i)))$
  - $g_{1,\theta} : \mathbb{R}^K \to \mathbb{R}^K$ and $g_{2,\theta} : \mathbb{R}^D \to \mathbb{R}^D$ are continuous Brenier maps. (Nonlinear injective)
  - The matrix β is a $K \times D$-dimensional matrix $(D \geq K)$ with full row rank. (Linear injective)

Properties of LIDVAE

- **Latent-identifiable VAE (LIDVAE)**
  $z_i \sim p(z_i), \qquad x_i \,|\, z_i \sim p(x_i \,|\, z_i\,;\,\theta) = \mathrm{EF}(x_i \,|\, g_{2,\theta}(\beta^\top g_{1,\theta}(z_i)))$

- Properties

  - **(Identifiability)** The latent variable z_i is identifiable in LIDVAE i.e. for all $i \in \{1,\ldots,n\}$, we have $p(x_i \,|\, z_i = \tilde{z}'; \theta) = p(x_i \,|\, z_i = \tilde{z}; \theta) \qquad \Rightarrow \qquad \tilde{z}' = \tilde{z}, \qquad \forall \tilde{z}', \tilde{z}, \theta\,.$

  - **(Flexibility)** For any VAE-generated data distribution, there exists an LIDAVE that can generate the same distribution.

Inference in LIDVAE

- Inference in LIDVAE is **identical** to the classical VAE, as they differ only in parameter constraints.
- LIDVAE is a **drop-in replacement** for VAE.
  - Both have the same capacity and share the same inference algorithm, but LIDVAE is identifiable and does not suffer from posterior
- The price we pay for LIDVAE is **computational.**
  - The generative model (i.e. decoder) is parametrized using the gradient of a neural network
  - Its optimization thus requires calculating **gradients of the gradient of a neural network**,
  - It increases the computational complexity and can sometimes challenge optimization.

Example: Identifiable Mixture VAE

- We replace the neural network mapping $p(x_i \mid z_i; \theta)$ with its injective counterpart, i.e. a composition of two Brenier maps and a matrix multiplication $g_{2,\theta}(\beta_2^\top g_{1,\theta}(\,\cdot\,))$

- Identifiable Mixture VAE (IDMVAE)

$$
\begin{aligned}
w_i &\sim \text{Categorical}(1/K),\\
z_i \mid w_i &\sim \text{EF}(\beta_1^\top w_i; \gamma_\theta),\\
x_i \mid z_i &\sim \text{EF}(g_{2,\theta}(\beta_2^\top g_{1,\theta}(z_i)))
\end{aligned}
$$

Example: Identifiable Sequential VAE

- We replace the neural network mapping $p(x_i \mid z_i; \theta)$ with its injective counterpart, i.e. a composition of two Brenier maps and a matrix multiplication $g_{2,\theta}(\beta_2^\top g_{1,\theta}(\,\cdot\,))$
- Identifiable Sequential VAE (IDSVAE)

$$
\begin{aligned}
z_i &\sim p(z_i), \\
x_i \mid z_i, x_{<i} &\sim \mathrm{EF}(g_{2,\theta}(\beta_2^\top g_{1,\theta}([z_i, f_\theta(x_{<i})])))
\end{aligned}
$$

LIDVAE: It works!

	Fashion-MNIST				Omniglot			
	AU	**KL**	**MI**	**LL**	**AU**	**KL**	**MI**	**LL**
VAE [28]	0.1	0.2	0.9	-258.8	0.02	0.0	0.1	-862.1
SA-VAE [25]	0.2	0.3	1.3	-252.2	0.1	0.2	1.0	-853.4
Lagging VAE [18]	0.4	0.6	1.6	-248.5	0.5	1.0	3.6	-849.4
β-VAE [19] (β=0.2)	0.6	1.2	2.4	-245.3	0.7	1.4	5.9	-842.6
LIDGMVAE (this work)	**1.0**	**1.6**	**2.6**	**-242.3**	**1.0**	**1.7**	**7.5**	**-820.3**

	Synthetic				Yahoo				Yelp			
	AU	**KL**	**MI**	**LL**	**AU**	**KL**	**MI**	**LL**	**AU**	**KL**	**MI**	**LL**
VAE [28]	0.0	0.0	0.0	-46.5	0.0	0.0	0.0	-519.7	0.0	0.0	0.0	-635.9
SA-VAE [25]	0.4	0.1	0.1	-40.2	0.2	1.0	0.2	-520.2	0.1	1.9	0.2	-631.5
Lagging VAE [18]	0.5	0.1	0.1	-40.0	0.3	1.6	0.4	**-518.6**	0.2	3.6	0.1	**-631.0**
β-VAE [19] (β=0.2)	**1.0**	0.1	0.1	**-39.9**	0.5	4.7	0.9	-524.4	0.3	**10.0**	0.1	-637.3
LIDSVAE	**1.0**	**0.5**	**0.6**	-40.3	**0.8**	**7.2**	**1.1**	-519.5	**0.7**	9.1	**0.9**	-634.2

Table 1: Across image and text datasets, LIDVAE outperforms existing VAE variants in preventing posterior collapse while achieving similar goodness-of-fit to the data.

Takeaways

- **Posterior collapse** is a problem of **latent variable non- identifiability**.
- It is **not** specific to the use of neural networks or variational inference algorithms in VAE. Rather, it is an **intrinsic** issue of the model and the dataset.
- We propose a class of **latent-identifiable variational autoencoders (LIDVAE)** via Brenier maps to resolve latent variable non-identifiability and mitigate posterior collapse.
- **Identifiability** used to be mostly of theoretical interest, but it turns out to have important **practical implications** in modern machine learning.

Thank you!

- Wang, Y., Blei, D.M., and Cunningham, J.P. (2021) Posterior Collapse and Latent Variable Non-identifiability. *NeurIPS 2021.*
- https://github.com/yixinwang/lidvae-public

Input Convex Neural Networks (ICNN)

An L-layer ICNN is a neural network mapping from $\mathbb{R}^d$ to $\mathbb{R}$. Given an input $u \in \mathbb{R}^d$, its lth layer is

$$z_0 = u, \qquad z_{l+1} = h_l(W_l z_l + A_l u + b_l), \qquad (l = 1, \ldots, L-1), \tag{6}$$

where the last layer z_L must be a scalar, $\{W_l\}$ are non-negative weight matrices with $W_0 = \mathbf{0}$, and $\{h_l\}$ are convex and non-decreasing functions. A common choice of h_0 is the square of a leaky RELU, $h_0(x) = (\max(\alpha \cdot x, x))^2$ with $\alpha = 0.2$; the remaining h_l's are set to be a leaky RELU, $h_l(x) = \max(\alpha \cdot x, x)$. This neural network is called "input convex" because it is guaranteed to be a convex function.

Input convex neural networks can approximate any convex function on a compact domain in sup norm (Theorem 1 of Chen et al. [9].) Given the neural network parameterization of convex functions, we can parametrize the Brenier map $g_\theta(\cdot)$ as its gradient with respect to the input $g_\theta(u) = \partial z_L / \partial u$. This neural network parameterization of Brenier map is a universal approxiamtor of all Brenier maps on a compact domain, because input convex neural networks are universal approximators of convex functions [9].