WHY DO WE NEED DEEP GENERATIVE MODELING?

Jakub M. Tomczak

24 November 2019

ML in PL Conference

VU VRIJE UNIVERSITEIT AMSTERDAM

Introduction

IS GENERATIVE MODELING IMPORTANT?

We learn a neural network to classify images:

IS GENERATIVE MODELING IMPORTANT?

We learn a neural network to classify images:

IS GENERATIVE MODELING IMPORTANT?

We learn a neural network to classify images:

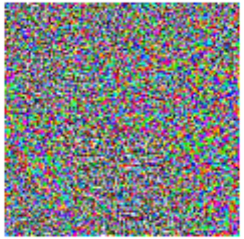
p(**panda**|x)=0.99

...

IS GENERATIVE MODELING IMPORTANT?

We learn a neural network to classify images:

+



=

p(**panda**|x)=0.99

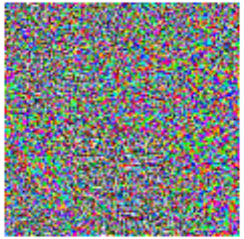
...

noise

IS GENERATIVE MODELING IMPORTANT?

We learn a neural network to classify images:

+



=

p(**panda**|x)=0.99

...

noise

p(**panda**|x)=0.01

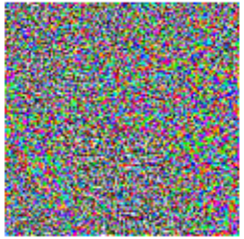
...

p(**dog**|x)=0.9

IS GENERATIVE MODELING IMPORTANT?

We learn a neural network to classify images:

+



=

p(**panda**|x)=0.99
...

noise

p(**panda**|x)=0.01
...
p(**dog**|x)=0.9

There is no semantic understanding of images.

IS GENERATIVE MODELING IMPORTANT?

This simple example shows that:

- A discriminative model is (probably) **not enough**.
- We need a notion of **uncertainty**.
- We need to **understand** the reality.

IS GENERATIVE MODELING IMPORTANT?

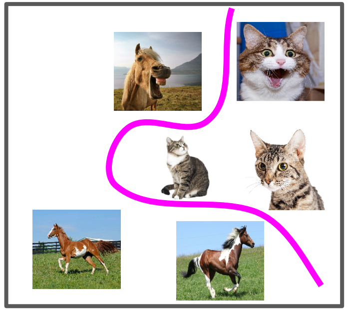
This simple example shows that:

- A discriminative model is (probably) **not enough**.
- We need a notion of **uncertainty**.
- We need to **understand** the reality.

A possible solution is **generative modeling**.

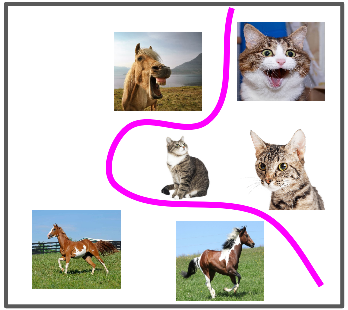
IS GENERATIVE MODELING IMPORTANT?

IS GENERATIVE MODELING IMPORTANT?

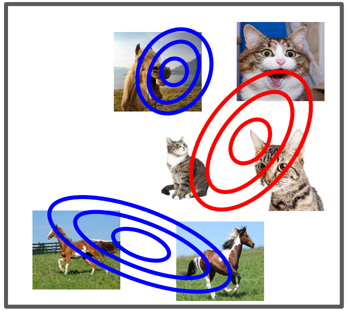


$$p_\theta(y|x)$$

IS GENERATIVE MODELING IMPORTANT?

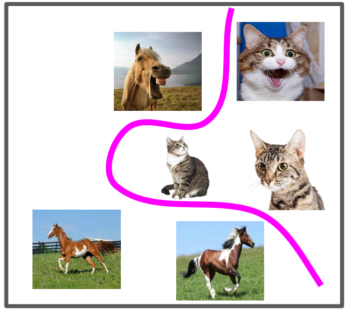


$$p_\theta(y|x)$$

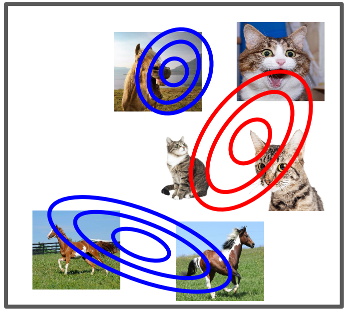


$$p_\theta(x, y) = p_\theta(y|x)\ p_\theta(x)$$

IS GENERATIVE MODELING IMPORTANT?

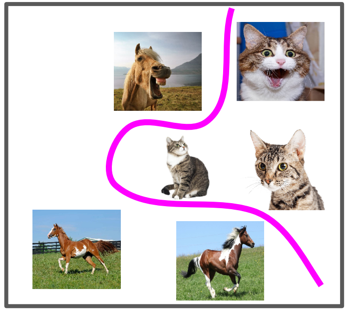


$$p_\theta(y|x)$$



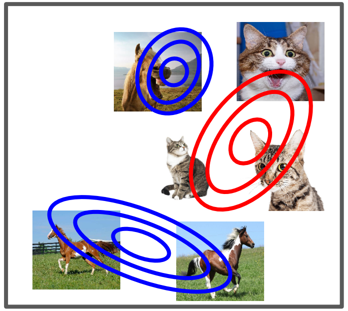
$$p_\theta(x, y) = p_\theta(y|x)\ p_\theta(x)$$

IS GENERATIVE MODELING IMPORTANT?



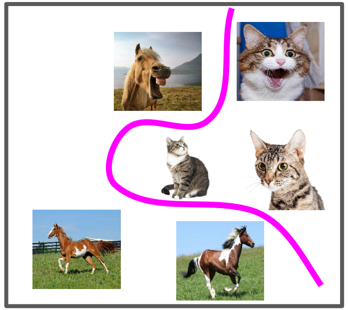
$$p_\theta(y|x)$$

High probability
of a **horse.**
=
Highly probable
decision!



$$p_\theta(x,y) = p_\theta(y|x)\ p_\theta(x)$$

IS GENERATIVE MODELING IMPORTANT?

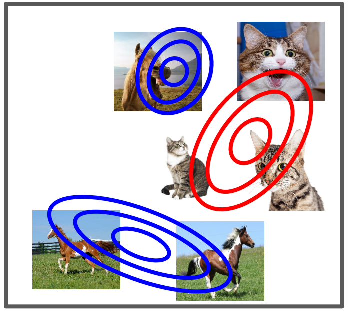


$$p_\theta(y|x)$$

High probability of a **horse.**

=

Highly probable decision!



$$p_\theta(x, y) = p_\theta(y|x)\ p_\theta(x)$$

High probability of a **horse.**

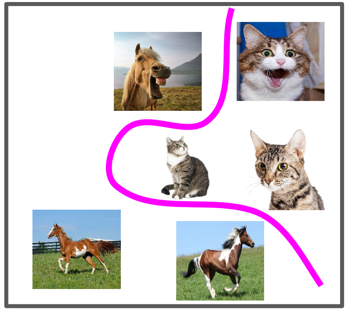
x

Low probability of the **object**

=

Uncertain decision!

IS GENERATIVE MODELING IMPORTANT?

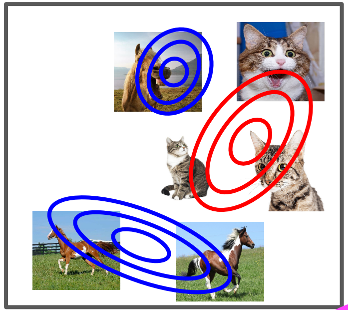


$$p_\theta(y|x)$$

High probability of a **horse.**

=

Highly probable decision!



$$p_\theta(x, y) = p_\theta(y|x)\, p_\theta(x) \quad !$$

High probability of a **horse.**

x

Low probability of the **object**

=

Uncertain decision!

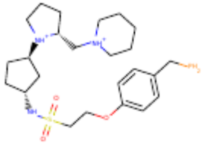
WHERE DO WE USE DEEP GENERATIVE MODELING?

" i want to talk to you . "
"i want to be with you . "
"i do n't want to be with you . "
i do n't want to be with you .
she did n't want to be with him .

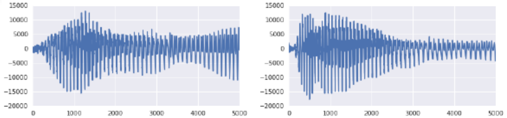
he was silent for a long moment .
he was silent for a moment .
it was quiet for a moment .
it was dark and cold .
there was a pause .
it was my turn .

Text analysis

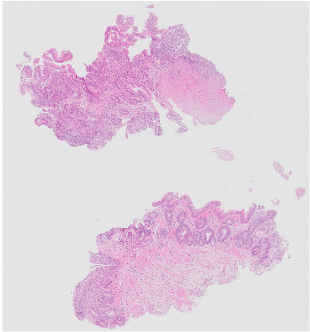
Image analysis



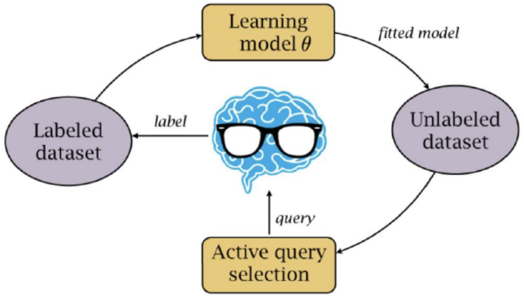
Graph analysis



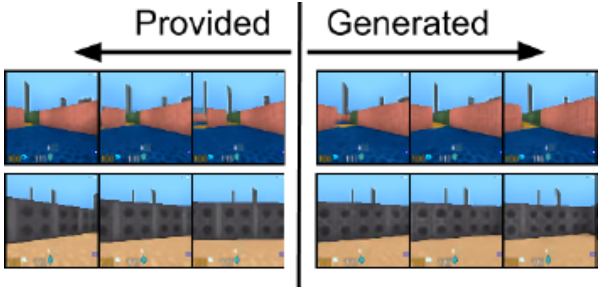
Audio analysis



Medical data



Active Learning



Reinforcement Learning

and more...

HOW TO FORMULATE GENERATIVE MODELS?

Generative model

Autoregressive (*e.g.*, PixelCNN)

Flow-based (e.g., RealNVP, GLOW)

Latent variable models

Implicit models (*e.g.*, GANs)

Prescribed models (*e.g.*, VAE)

HOW TO FORMULATE GENERATIVE MODELS?

	Training	Likelihood	Sampling	Compression
Autoregressive models (e.g., PixelCNN)	Stable	Exact	Slow	No
Flow-based models (e.g., RealNVP)	Stable	Exact	Fast/Slow	No
Implicit models (e.g., GANs)	Unstable	No	Fast	No
Prescribed models (e.g., VAEs)	Stable	Approximate	Fast	Yes

HOW TO FORMULATE GENERATIVE MODELS?

	Training	Likelihood	Sampling	Compression
Autoregressive models (e.g., PixelCNN)	Stable	Exact	Slow	No
Flow-based models (e.g., RealNVP)	Stable	Exact	Fast/Slow	No
Implicit models (e.g., GANs)	Unstable	No	Fast	No
Prescribed models (e.g., VAEs)	Stable	Approximate	Fast	Yes

GENERATIVE MODELS AS (SPHERICAL) COWS

GENERATIVE MODELS AS (SPHERICAL) COWS

flow-based models

GENERATIVE MODELS AS (SPHERICAL) COWS

flow-based models

latent variable models

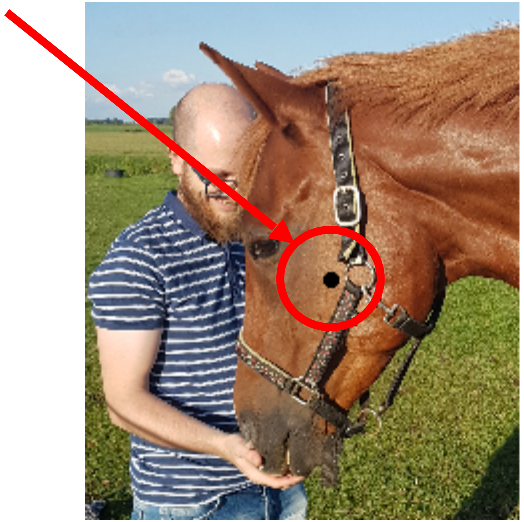
Deep latent variable models

GENERATIVE MODELING

Modeling in high-dimensional spaces is difficult.

GENERATIVE MODELING

Modeling in high-dimensional spaces is difficult.



GENERATIVE MODELING

Modeling in high-dimensional spaces is difficult.

Modeling **all dependencies** among pixels:

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{c=1}^{C} \psi_c(\mathbf{x}_c)$$

GENERATIVE MODELING

Modeling in high-dimensional spaces is difficult.

Modeling **all dependencies** among pixels:

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{c=1}^{C} \psi_c(\mathbf{x}_c)$$

problematic

GENERATIVE MODELING

Modeling in high-dimensional spaces is difficult.

Modeling **all dependencies** among pixels:

$$p(\mathbf{x}) = \frac{1}{Z}\prod_{c=1}^{C}\psi_c(\mathbf{x}_c)$$

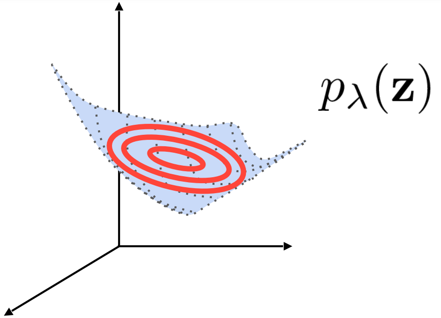
problematic

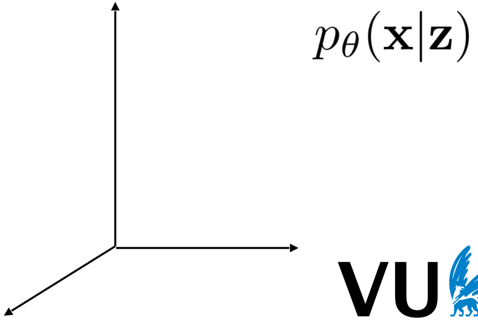
A possible **solution**: **Latent Variable Models**!

GENERATIVE MODELING WITH LATENT VARIABLES

Generative process:

1. $\mathbf{z} \sim p_\lambda(\mathbf{z})$
2. $\mathbf{x} \sim p_\theta(\mathbf{x}|\mathbf{z})$

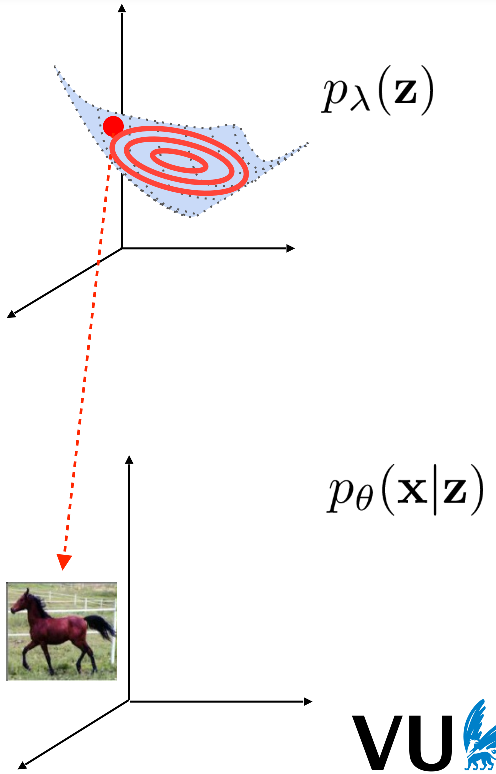




GENERATIVE MODELING WITH LATENT VARIABLES

Generative process:

1. $\mathbf{z} \sim p_\lambda(\mathbf{z})$
2. $\mathbf{x} \sim p_\theta(\mathbf{x}|\mathbf{z})$

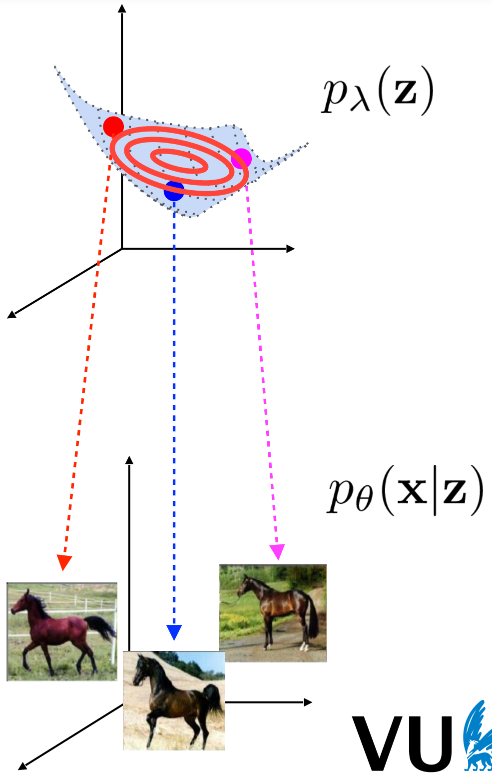


GENERATIVE MODELING WITH LATENT VARIABLES

Generative process:

$$1.\ \mathbf{z} \sim p_\lambda(\mathbf{z})$$

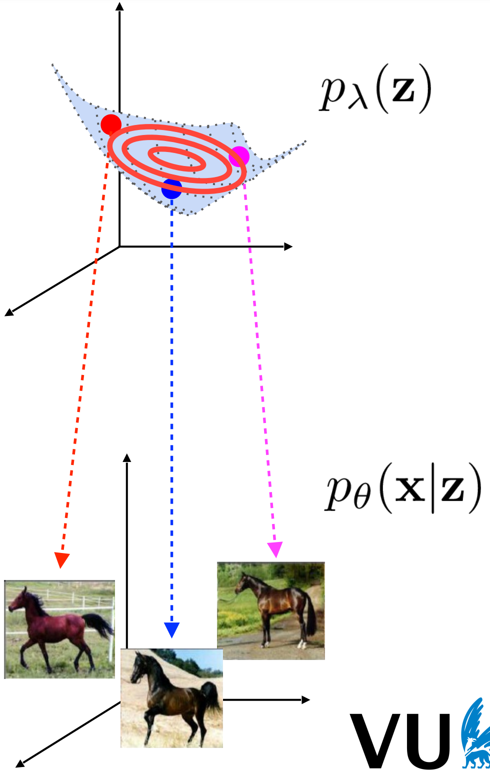
$$2.\ \mathbf{x} \sim p_\theta(\mathbf{x}|\mathbf{z})$$



GENERATIVE MODELING WITH LATENT VARIABLES

Generative process:

$$1.\ \mathbf{z} \sim p_\lambda(\mathbf{z})$$
$$2.\ \mathbf{x} \sim p_\theta(\mathbf{x}|\mathbf{z})$$



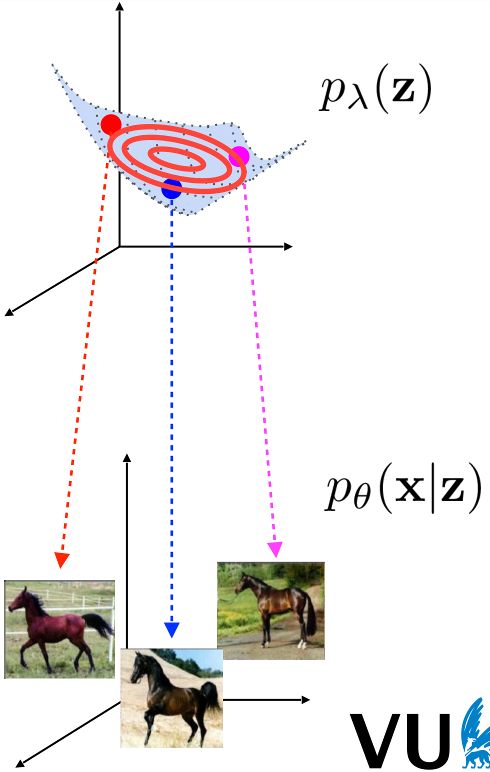
Log of marginal distribution:

$$\log p_\vartheta(\mathbf{x}) = \log \int p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z}) \mathrm{d}\mathbf{z}$$

GENERATIVE MODELING WITH LATENT VARIABLES

Generative process:

$$1.\ \mathbf{z} \sim p_\lambda(\mathbf{z})$$
$$2.\ \mathbf{x} \sim p_\theta(\mathbf{x}|\mathbf{z})$$



Log of marginal distribution:

$$\log p_\vartheta(\mathbf{x}) = \log \int p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})\mathrm{d}\mathbf{z}$$

How to train such model efficiently?

VARIATIONAL INFERENCE FOR LATENT VARIABLE MODELS

$$\begin{aligned}
\log p_{\vartheta}(\mathbf{x}) &= \log \int p_{\theta}(\mathbf{x}|\mathbf{z})\ p_{\lambda}(\mathbf{z})\mathrm{d}\mathbf{z} \\
&= \log \int \frac{q_{\phi}(\mathbf{z}|\mathbf{x})}{q_{\phi}(\mathbf{z}|\mathbf{x})} p_{\theta}(\mathbf{x}|\mathbf{z})\ p_{\lambda}(\mathbf{z})\mathrm{d}\mathbf{z} \\
&\geq \int q_{\phi}(\mathbf{z}|\mathbf{x}) \log \frac{p_{\theta}(\mathbf{x}|\mathbf{z})\ p_{\lambda}(\mathbf{z})}{q_{\phi}(\mathbf{z}|\mathbf{x})} \mathrm{d}\mathbf{z} \\
&= \mathbb{E}_{\mathbf{z}\sim q_{\phi}(\mathbf{z}|\mathbf{x})}\Big[\log p_{\theta}(\mathbf{x}|\mathbf{z})\Big] - \mathrm{KL}\Big(q_{\phi}(\mathbf{z}|\mathbf{x})||p_{\lambda}(\mathbf{z})\Big)
\end{aligned}$$

VARIATIONAL INFERENCE FOR LATENT VARIABLE MODELS

Variational posterior

$$\begin{aligned}
\log p_{\vartheta}(\mathbf{x}) &= \log \int p_{\theta}(\mathbf{x}|\mathbf{z})\ p_{\lambda}(\mathbf{z}) \mathrm{d}\mathbf{z} \\
&= \log \int \frac{q_{\phi}(\mathbf{z}|\mathbf{x})}{q_{\phi}(\mathbf{z}|\mathbf{x})} p_{\theta}(\mathbf{x}|\mathbf{z})\ p_{\lambda}(\mathbf{z}) \mathrm{d}\mathbf{z} \\
&\geq \int q_{\phi}(\mathbf{z}|\mathbf{x}) \log \frac{p_{\theta}(\mathbf{x}|\mathbf{z})\ p_{\lambda}(\mathbf{z})}{q_{\phi}(\mathbf{z}|\mathbf{x})} \mathrm{d}\mathbf{z} \\
&= \mathbb{E}_{\mathbf{z}\sim q_{\phi}(\mathbf{z}|\mathbf{x})}\Big[\log p_{\theta}(\mathbf{x}|\mathbf{z})\Big] - \mathrm{KL}\Big(q_{\phi}(\mathbf{z}|\mathbf{x}) || p_{\lambda}(\mathbf{z})\Big)
\end{aligned}$$

VARIATIONAL INFERENCE FOR LATENT VARIABLE MODELS

$$
\begin{aligned}
\log p_\vartheta(\mathbf{x}) &= \log \int p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})\mathrm{d}\mathbf{z} \\
&= \log \int \frac{q_\phi(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})} p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})\mathrm{d}\mathbf{z} \\
&\geq \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\mathrm{d}\mathbf{z} \\
&= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\Big[\log p_\theta(\mathbf{x}|\mathbf{z})\Big] - \mathrm{KL}\Big(q_\phi(\mathbf{z}|\mathbf{x})||p_\lambda(\mathbf{z})\Big)
\end{aligned}
$$

Jensen's inequality

VARIATIONAL INFERENCE FOR LATENT VARIABLE MODELS

$$
\begin{aligned}
\log p_{\vartheta}(\mathbf{x}) &= \log \int p_{\theta}(\mathbf{x}|\mathbf{z})\ p_{\lambda}(\mathbf{z}) \mathrm{d}\mathbf{z} \\
&= \log \int \frac{q_{\phi}(\mathbf{z}|\mathbf{x})}{q_{\phi}(\mathbf{z}|\mathbf{x})} p_{\theta}(\mathbf{x}|\mathbf{z})\ p_{\lambda}(\mathbf{z}) \mathrm{d}\mathbf{z} \\
&\geq \int q_{\phi}(\mathbf{z}|\mathbf{x}) \log \frac{p_{\theta}(\mathbf{x}|\mathbf{z})\ p_{\lambda}(\mathbf{z})}{q_{\phi}(\mathbf{z}|\mathbf{x})} \mathrm{d}\mathbf{z} \\
&= \underbrace{\mathbb{E}_{\mathbf{z}\sim q_{\phi}(\mathbf{z}|\mathbf{x})}\Big[\log p_{\theta}(\mathbf{x}|\mathbf{z})\Big]}_{\text{Reconstruction error}} - \underbrace{\mathrm{KL}\Big(q_{\phi}(\mathbf{z}|\mathbf{x})||p_{\lambda}(\mathbf{z})\Big)}_{\text{Regularization}}
\end{aligned}
$$

VARIATIONAL INFERENCE FOR LATENT VARIABLE MODELS

$$
\begin{aligned}
\log p_\vartheta(\mathbf{x}) &= \log \int p_\theta(\mathbf{x}|\mathbf{z})\; p_\lambda(\mathbf{z})\mathrm{d}\mathbf{z} \\
&= \log \int \frac{q_\phi(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})} p_\theta(\mathbf{x}|\mathbf{z})\; p_\lambda(\mathbf{z})\mathrm{d}\mathbf{z} \\
&\geq \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{p_\theta(\mathbf{x}|\mathbf{z})\; p_\lambda(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\mathrm{d}\mathbf{z} \\
&= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - \mathrm{KL}\left(q_\phi(\mathbf{z}|\mathbf{x}) || p_\lambda(\mathbf{z})\right)
\end{aligned}
$$

decoder

encoder

marginal

VARIATIONAL INFERENCE FOR LATENT VARIABLE MODELS

$$\log p_\vartheta(\mathbf{x}) = \log \int p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z}) d\mathbf{z}$$

$$= \log \int \frac{q_\phi(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})} p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z}) d\mathbf{z}$$

$$\geq \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} d\mathbf{z}$$

$$= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - \mathrm{KL}\left(q_\phi(\mathbf{z}|\mathbf{x}) || p_\lambda(\mathbf{z})\right)$$

decoder

encoder

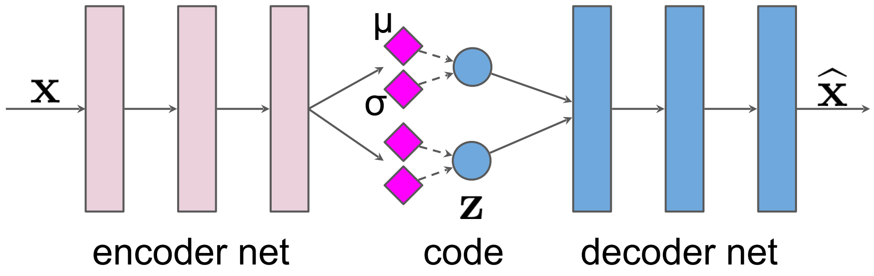
marginal

= Variational Auto-Encoder

VU

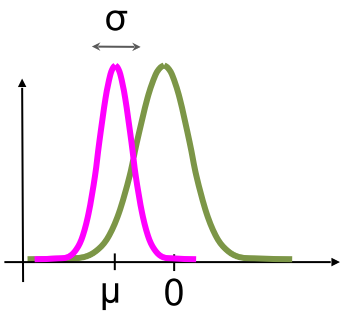
VARIATIONAL AUTO-ENCODERS

Variational posterior (**encoder**) and likelihood function (**decoder**) are parameterized by neural networks.



Reparameterization trick:
move the stochasticity to independent random variables

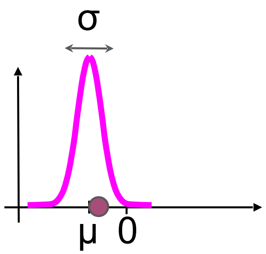
$$\mathbf{z} = f(\boldsymbol{\mu}, \boldsymbol{\sigma}; \boldsymbol{\varepsilon}), \text{ where } \boldsymbol{\varepsilon} \sim p(\boldsymbol{\varepsilon})$$

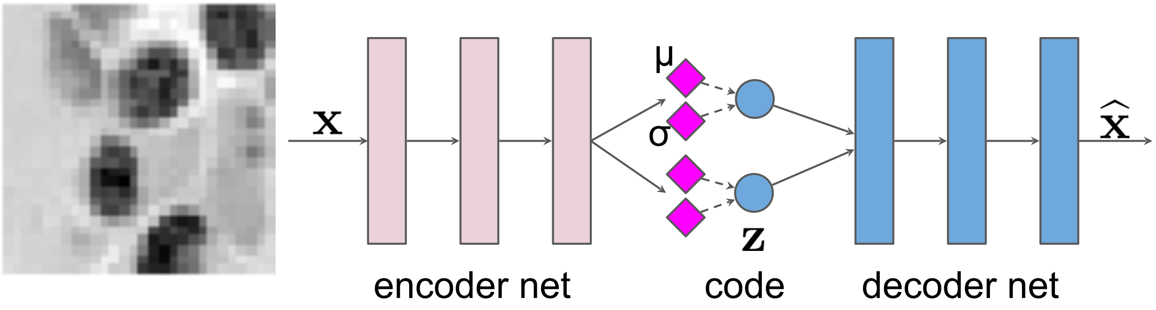


VARIATIONAL AUTO-ENCODERS

VAE copies input to output through a **bottleneck**.

VAE learns a **code** of the data.

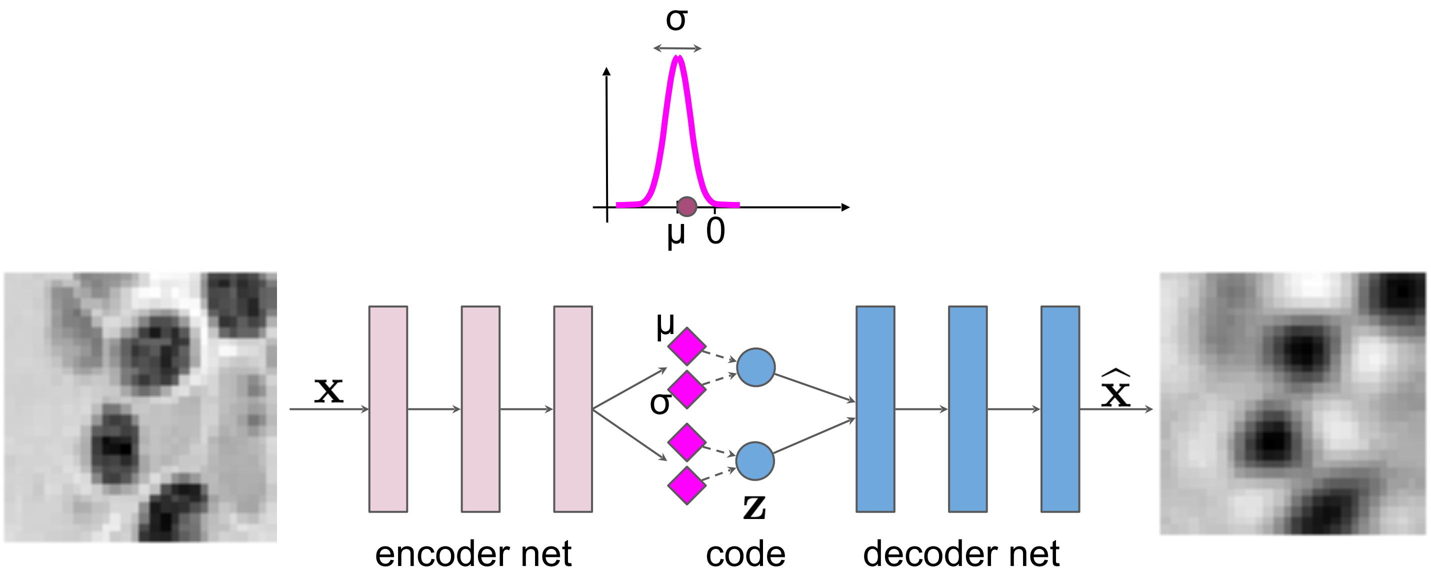




VARIATIONAL AUTO-ENCODERS

VAE copies input to output through a **bottleneck**.

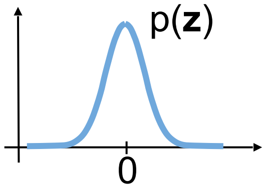
VAE learns a **code** of the data.

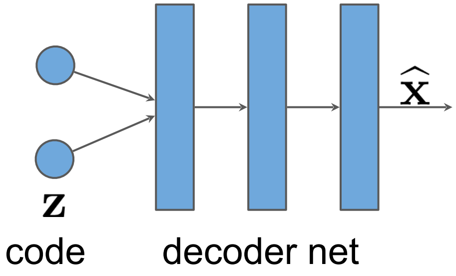


VARIATIONAL AUTO-ENCODERS

VAE has a **marginal** on the latent code.

VAE can **generate** new data.

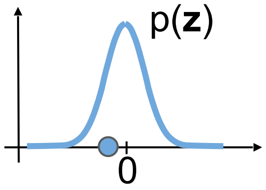


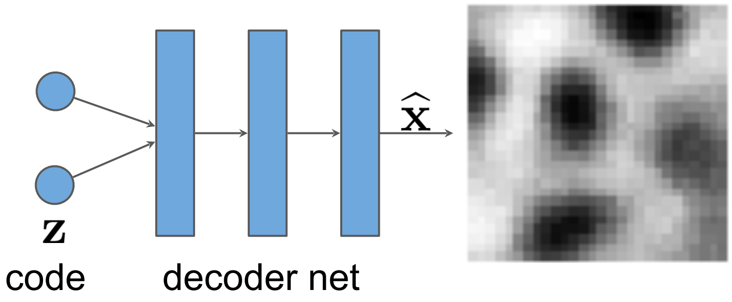


VARIATIONAL AUTO-ENCODERS

VAE has a **marginal** on the latent code.

VAE can **generate** new data.

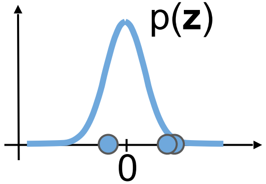


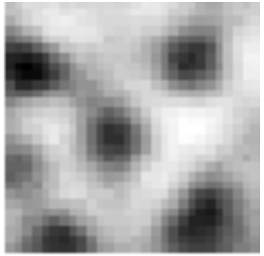


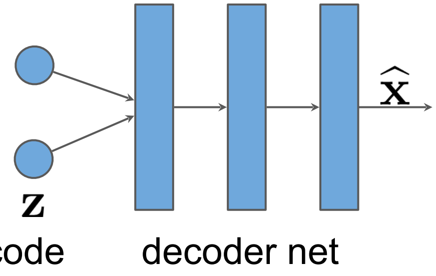
VARIATIONAL AUTO-ENCODERS

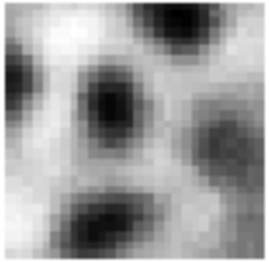
VAE has a **marginal** on the latent code.

VAE can **generate** new data.









COMMON ISSUES WITH VAES

$$q_{\phi}(\mathbf{z}|\mathbf{x}) \propto p_{\theta}(\mathbf{x}|\mathbf{z})\ p_{\lambda}(\mathbf{z})$$

Weak decoders → bad generations/reconstructions

Weak encoders → bad latent representation

Weak marginals → bad generations

Variational posteriors → what family of distributions?

Others…

COMPONENTS OF VAES

$$q_{\phi}(\mathbf{z}|\mathbf{x}) \propto p_{\theta}(\mathbf{x}|\mathbf{z})\ p_{\lambda}(\mathbf{z})$$

Resnets
DRAW
Autoregressive models
Normalizing flows

Autoregressive models
Normalizing flows
VampPrior
Implicit prior

COMPONENTS OF VAES

$$q_\phi(\mathbf{z}|\mathbf{x}) \propto p_\theta(\mathbf{x}|\mathbf{z})\; p_\lambda(\mathbf{z})$$

Normalizing flows
Discrete encoders
Hyperspherical dist.
Hyperbolic-normal dist.
Group theory

Resnets
DRAW
Autoregressive models
Normalizing flows

Autoregressive models
Normalizing flows
VampPrior
Implicit prior

COMPONENTS OF VAES

$$q_\phi(\mathbf{z}|\mathbf{x}) \propto p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})$$

Normalizing flows
Discrete encoders
Hyperspherical dist.
Hyperbolic-normal dist.
Group theory

Resnets
DRAW
Autoregressive models
Normalizing flows

Autoregressive models
Normalizing flows
VampPrior
Implicit prior

$$\mathrm{ELBO}(\mathbf{x};\theta,\phi,\lambda)$$

Adversarial learning
MMD
Wasserstein AE

COMPONENTS OF VAES

$$q_\phi(\mathbf{z}|\mathbf{x}) \propto p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})$$

Normalizing flows
Discrete encoders
Hyperspherical dist.
Hyperbolic-normal dist.
Group theory

Resnets
DRAW
Autoregressive models
Normalizing flows

Autoregressive models
Normalizing flows
VampPrior
Implicit prior

$$\mathrm{ELBO}(\mathbf{x};\theta,\phi,\lambda)$$

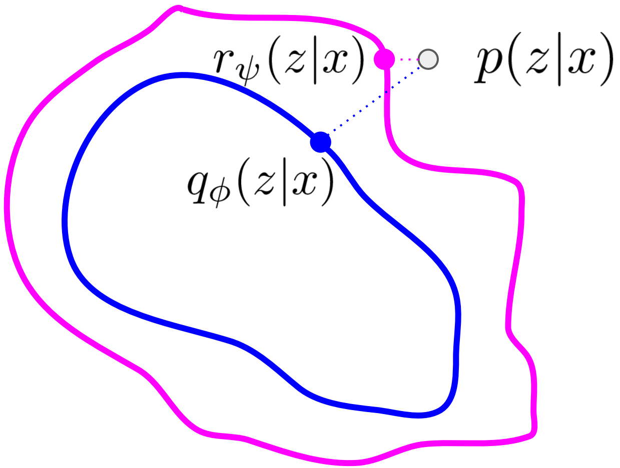
Adversarial learning
MMD
Wasserstein AE

VARIATIONAL POSTERIOR IN VAES

Question: How to minimize the KL(q||p)?

$$\mathrm{ELBO}(\mathbf{x};\theta,\phi,\lambda) = \log p_{\vartheta}(\mathbf{x}) - \mathrm{KL}\Big(q_{\phi}(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})\Big)$$

In other words: *How to formulate a more flexible family of approximate (variational) posteriors?*



Using Gaussian is not sufficiently **flexible**.

We need a **computationally efficient tool**.

Arvanitidis, G., Hansen, L. K., & Hauberg, S. (2017). Latent space oddity: on the curvature of deep generative models. *arXiv preprint arXiv:1710.11379*.

VARIATIONAL INFERENCE WITH NORMALIZING FLOWS

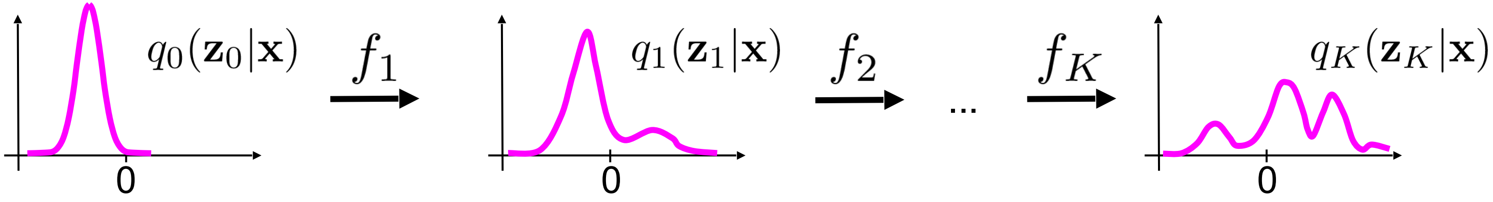
Sample from a “simple” distribution: $\mathbf{z}_0 \sim q_0(\mathbf{z}|\mathbf{x}) = \mathcal{N}\left(\mathbf{z}|\mu(\mathbf{x}), \mathrm{diag}(\sigma^2(\mathbf{x}))\right)$

Rezende, D. J., & Mohamed, S. (2015). Variational inference with normalizing flows. ICML 2015

VARIATIONAL INFERENCE WITH NORMALIZING FLOWS

Sample from a "simple" distribution: $\mathbf{z}_0 \sim q_0(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}|\mu(\mathbf{x}), \mathrm{diag}(\sigma^2(\mathbf{x})))$

Apply a sequence of *K* invertible transformations: $f_k : \mathbb{R}^M \to \mathbb{R}^M$

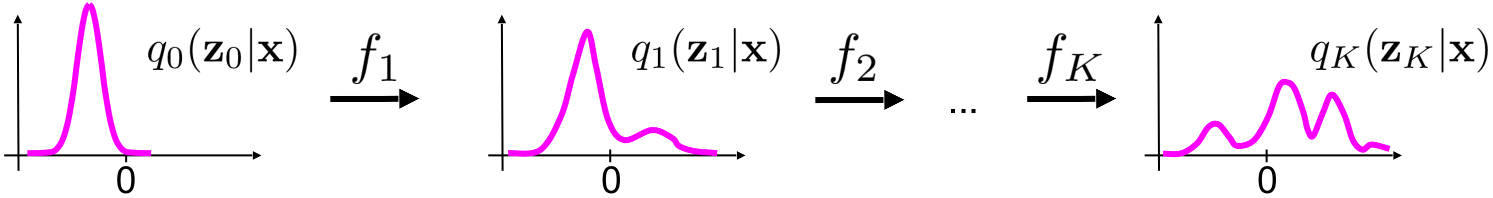


Rezende, D. J., & Mohamed, S. (2015). Variational inference with normalizing flows. ICML 2015

VARIATIONAL INFERENCE WITH NORMALIZING FLOWS

Sample from a "simple" distribution: $\mathbf{z}_0 \sim q_0(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}|\mu(\mathbf{x}), \mathrm{diag}(\sigma^2(\mathbf{x})))$

Apply a sequence of *K* invertible transformations: $f_k : \mathbb{R}^M \rightarrow \mathbb{R}^M$



and the change of variables yields:

$$q_K(\mathbf{z}_K|\mathbf{x}) = q_0(\mathbf{z}_0|\mathbf{x}) \prod_{k=1}^{K} \left| \det \frac{\partial f_k(\mathbf{z}_{k-1})}{\partial \mathbf{z}_{k-1}} \right|^{-1}$$

Rezende, D. J., & Mohamed, S. (2015). Variational inference with normalizing flows. ICML 2015

VARIATIONAL INFERENCE WITH NORMALIZING FLOWS

The learning objective (ELBO) with normalizing flows becomes:

$$\mathrm{ELBO}(\mathbf{x};\theta,\phi,\lambda) = \mathbb{E}_{\mathbf{z}_0\sim q_0(\mathbf{z}_0|\mathbf{x})}\Big[\log p_\theta(\mathbf{x}|\mathbf{z}_K)\Big] - \mathrm{KL}\Big(q_0(\mathbf{z}_0|\mathbf{x})||p_\lambda(\mathbf{z}_K)\Big)+$$

$$+\mathbb{E}_{\mathbf{z}_0\sim q_0(\mathbf{z}_0|\mathbf{x})}\Big[\sum_{k=1}^{K}\log\Big|\det\frac{\partial f_k(\mathbf{z}_{k-1})}{\partial \mathbf{z}_{k-1}}\Big|\Big]$$

Rezende, D. J., & Mohamed, S. (2015). Variational inference with normalizing flows. ICML 2015

VARIATIONAL INFERENCE WITH NORMALIZING FLOWS

The learning objective (ELBO) with normalizing flows becomes:

$$\mathrm{ELBO}(\mathbf{x};\theta,\phi,\lambda) = \mathbb{E}_{\mathbf{z}_0\sim q_0(\mathbf{z}_0|\mathbf{x})}\Big[\log p_\theta(\mathbf{x}|\mathbf{z}_K)\Big] - \mathrm{KL}\Big(q_0(\mathbf{z}_0|\mathbf{x})||p_\lambda(\mathbf{z}_K)\Big) +$$

$$+\mathbb{E}_{\mathbf{z}_0\sim q_0(\mathbf{z}_0|\mathbf{x})}\Big[\sum_{k=1}^{K}\log\Big|\det\frac{\partial f_k(\mathbf{z}_{k-1})}{\partial \mathbf{z}_{k-1}}\Big|\Big]$$

The difficulty lies in calculating the Jacobian determinant:

Rezende, D. J., & Mohamed, S. (2015). Variational inference with normalizing flows. ICML 2015

VARIATIONAL INFERENCE WITH NORMALIZING FLOWS

The learning objective (ELBO) with normalizing flows becomes:

$$\mathrm{ELBO}(\mathbf{x};\theta,\phi,\lambda) = \mathbb{E}_{\mathbf{z}_0 \sim q_0(\mathbf{z}_0|\mathbf{x})}\Big[\log p_\theta(\mathbf{x}|\mathbf{z}_K)\Big] - \mathrm{KL}\Big(q_0(\mathbf{z}_0|\mathbf{x})||p_\lambda(\mathbf{z}_K)\Big) +$$
$$+\mathbb{E}_{\mathbf{z}_0 \sim q_0(\mathbf{z}_0|\mathbf{x})}\left[\sum_{k=1}^{K} \log \left|\det \frac{\partial f_k(\mathbf{z}_{k-1})}{\partial \mathbf{z}_{k-1}}\right|\right]$$

The difficulty lies in calculating the Jacobian determinant:

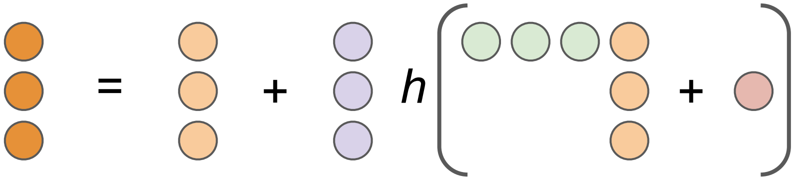
Volume-preserving flows: $\left|\det \frac{\partial f_k(\mathbf{z}_{k-1})}{\partial \mathbf{z}_{k-1}}\right| = 1$

General normalizing flows: $\left|\det \frac{\partial f_k(\mathbf{z}_{k-1})}{\partial \mathbf{z}_{k-1}}\right|$ is "easy" to compute.

Rezende, D. J., & Mohamed, S. (2015). Variational inference with normalizing flows. ICML 2015

PLANAR FLOWS

First, let us take a look at **planar flows** (Rezende & Mohamed, 2015):

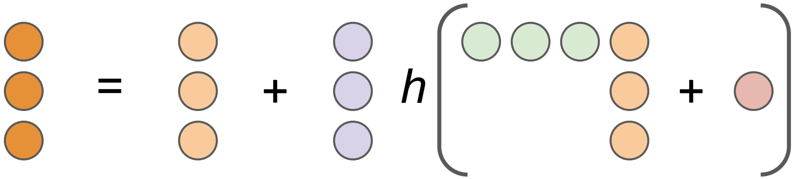


$$\mathbf{z}_k = \mathbf{z}_{k-1} + \mathbf{u}\, h(\mathbf{w}^\top \mathbf{z}_{k-1} + b)$$

This is equivalent to a residual layer with a **single** neuron.

PLANAR FLOWS

First, let us take a look at **planar flows** (Rezende & Mohamed, 2015):



$$\mathbf{z}_k = \mathbf{z}_{k-1} + \mathbf{u}\; h(\mathbf{w}^\top \mathbf{z}_{k-1} + b)$$

This is equivalent to a residual layer with a **single** neuron.

Can we calculate the Jacobian determinant efficiently?

PLANAR FLOWS

We can use the **matrix determinant lemma** to get the Jacobian determinant:

$$\det \frac{\partial \mathbf{z}'}{\partial \mathbf{z}} = 1 + \mathbf{u}^\top h'(\mathbf{w}^\top \mathbf{z} + b)\mathbf{w}$$

which is **linear** wrt the number of $\mathbf{z}$'s.

PLANAR FLOWS

We can use the **matrix determinant lemma** to get the Jacobian determinant:

$$\det \frac{\partial \mathbf{z}'}{\partial \mathbf{z}} = 1 + \mathbf{u}^\top h'(\mathbf{w}^\top \mathbf{z} + b)\mathbf{w}$$

which is **linear** wrt the number of **z**'s.

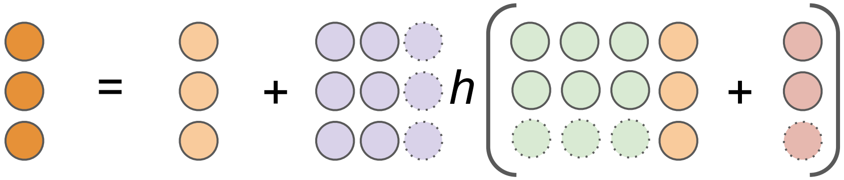
The bottleneck requires many steps, so how can we improve on that?

1. Can we **generalize** planar flows?

2. If yes, how can we compute the Jacobian determinant **efficiently**?

GENERALIZING PLANAR FLOWS: SYLVESTER FLOWS

We can control the bottleneck by generalizing **u** and **w** to **A** and **B**.

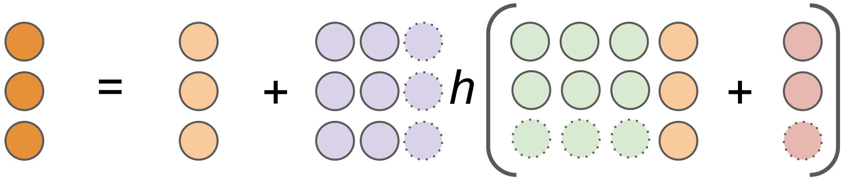


$$\mathbf{z}_k = \mathbf{z}_{k-1} + \mathbf{A}\ h(\mathbf{B}^\top \mathbf{z}_{k-1} + \mathbf{b})$$

How to calculate det of Jacobian?

GENERALIZING PLANAR FLOWS: SYLVESTER FLOWS

We can control the bottleneck by generalizing **u** and **w** to **A** and **B**.



$$\mathbf{z}_k = \mathbf{z}_{k-1} + \mathbf{A}\ h(\mathbf{B}^\top \mathbf{z}_{k-1} + \mathbf{b})$$

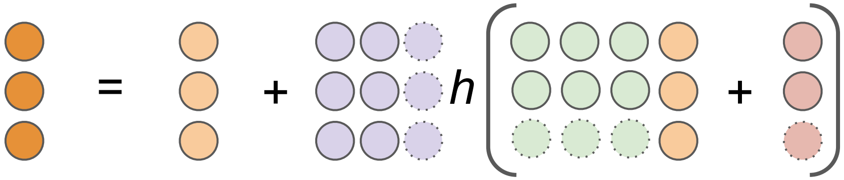
How to calculate det of Jacobian? Use **Sylvester Determinant Identity**:

$$\det \frac{\partial \mathbf{z}'}{\partial \mathbf{z}} = \det\Big(\mathbf{I} + \mathrm{diag}\big(h'(\mathbf{B}\mathbf{z} + \mathbf{b})\mathbf{B}\mathbf{A}\big)$$

v.d. Berg, R., Hasenclever, L., Tomczak, J.M., & Welling, M. (2018). Sylvester Normalizing Flows for Variational Inference. UAI 2018

GENERALIZING PLANAR FLOWS: SYLVESTER FLOWS

We can control the bottleneck by generalizing **u** and **w** to **A** and **B**.



$$\mathbf{z}_k = \mathbf{z}_{k-1} + \mathbf{A}\ h(\mathbf{B}^\top \mathbf{z}_{k-1} + \mathbf{b})$$

How to calculate det of Jacobian? Use **Sylvester Determinant Identity**:

$$\det \frac{\partial \mathbf{z}'}{\partial \mathbf{z}} = \det\Big(\mathbf{I} + \mathrm{diag}\big(h'(\mathbf{B}\mathbf{z} + \mathbf{b})\mathbf{B}\mathbf{A}\big)$$

OK, but it's very expensive! Can we simplify these calculations?

v.d. Berg, R., Hasenclever, L., Tomczak, J.M., & Welling, M. (2018). Sylvester Normalizing Flows for Variational Inference. UAI 2018

SYLVESTER FLOWS

Use of **Sylvester Determinant Identity** yields:

$$\det \frac{\partial \mathbf{z}'}{\partial \mathbf{z}} = \det\Big(\mathbf{I} + \mathrm{diag}\big(h'(\mathbf{B}\mathbf{z} + \mathbf{b})\mathbf{B}\mathbf{A}\big)$$

Next, we can use **QR decomposition** to represent **A** and **B**:

$$\begin{aligned}\det \frac{\partial \mathbf{z}'}{\partial \mathbf{z}} &= \det\Big(\mathbf{I} + \mathrm{diag}\big(h'(\mathbf{R}_B\mathbf{Q}^\top\mathbf{z} + \mathbf{b})\mathbf{R}_B\mathbf{Q}^\top\mathbf{Q}\mathbf{R}_A\big) \\ &= \det\Big(\mathbf{I} + \mathrm{diag}\big(h'(\mathbf{R}_B\mathbf{Q}^\top\mathbf{z} + \mathbf{b})\mathbf{R}_B\mathbf{R}_A\big)\end{aligned}$$

$\mathbf{Q}$ columns are orthonormal vectors

$\mathbf{R}_A,\ \mathbf{R}_B$ triangular matrices

SYLVESTER FLOWS

Use of **Sylvester Determinant Identity** yields:

$$\det \frac{\partial \mathbf{z}'}{\partial \mathbf{z}} = \det\Big(\mathbf{I} + \mathrm{diag}\big(h'(\mathbf{B}\mathbf{z} + \mathbf{b})\mathbf{B}\mathbf{A}\big)$$

Next, we can use **QR decomposition** to represent **A** and **B**:

$$\det \frac{\partial \mathbf{z}'}{\partial \mathbf{z}} = \det\Big(\mathbf{I} + \mathrm{diag}\big(h'(\mathbf{R}_B\mathbf{Q}^\top\mathbf{z} + \mathbf{b})\mathbf{R}_B\mathbf{Q}^\top\mathbf{Q}\mathbf{R}_A\big)$$
$$= \det\Big(\mathbf{I} + \mathrm{diag}\big(h'(\mathbf{R}_B\mathbf{Q}^\top\mathbf{z} + \mathbf{b})\mathbf{R}_B\mathbf{R}_A\big)$$

$\mathbf{Q}$ columns are orthonormal vectors **How to keep an orthogonal matrix?**

$\mathbf{R}_A,\ \mathbf{R}_B$ triangular matrices

SYLVESTER FLOWS: LEARNING ORTHOGONAL MATRIX

1.(**O-SNF**) Iterative orthogonalization procedure (e.g., Kovarik, 1970):

a. We can **backpropagate** through this procedure.

$$\mathbf{Q} := \mathbf{Q}\Big(\mathbf{I} + \frac{1}{2}(\mathbf{I} - \mathbf{Q}^\top\mathbf{Q})\Big)$$

b. We can control the bottleneck by changing the number of columns.

2.(**H-SNF**) Use Householder transformations to represent **Q**.

a. H-SNF is a non-linear **extension** of the Householder flow.

$$\mathbf{Q} = \mathbf{I} - 2\frac{\mathbf{v}\mathbf{v}^\top}{\mathbf{v}^\top\mathbf{v}}$$

b. No bottleneck!

3.(**T-SNF**) Alternate between identity matrix and a fixed permutation matrix.

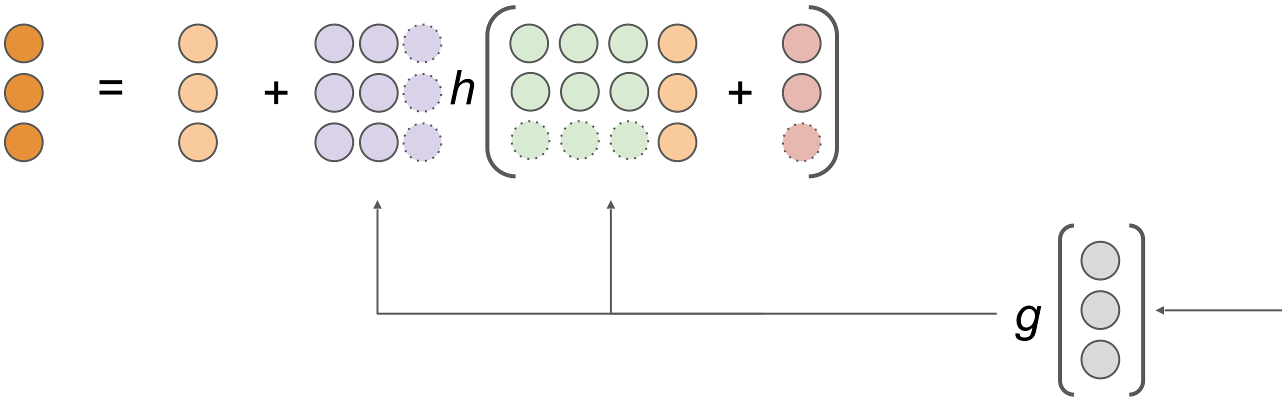
a. Used also in RealNVP and IAF.

SYLVESTER NORMALIZING FLOWS

A single step: $\mathbf{z}_k = \mathbf{z}_{k-1} + \mathbf{Q}\mathbf{R}_A\ h(\mathbf{R}_B\mathbf{Q}^\top \mathbf{z}_{k-1} + \mathbf{b})$

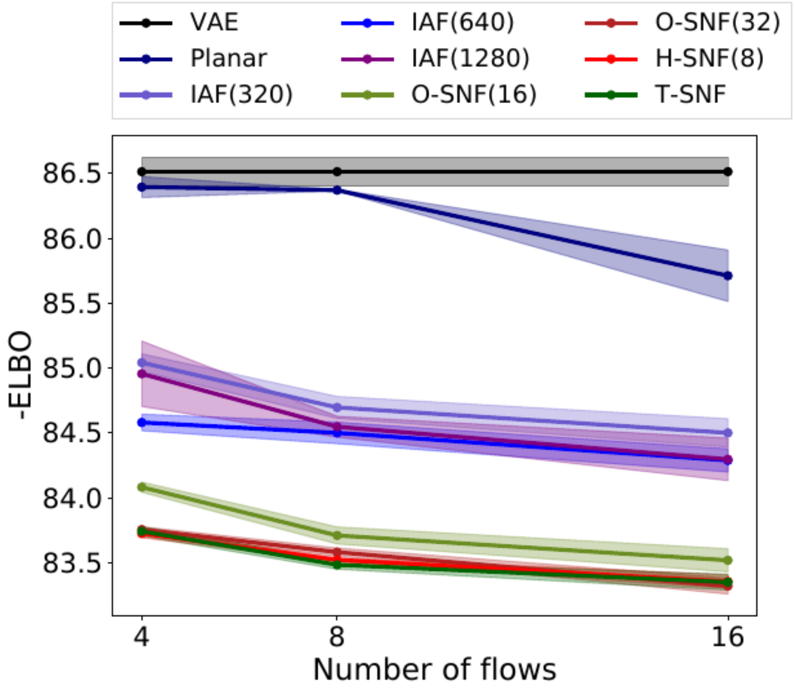
Keep **Q** **orthogonal**: (i) bottleneck: O-SNF, (ii) w/o: H-SNF, T-SNF.

Use **hypernets** to calculate **Q** and **R**'s:



VU

SYLVESTER FLOWS: RESULTS ON MNIST



Model	-ELBO	NLL
VAE	86.55 ± 0.06	82.14 ± 0.07
Planar	86.06 ± 0.31	81.91 ± 0.22
IAF	84.20 ± 0.17	80.79 ± 0.12
O-SNF	$\mathbf{83.32 \pm 0.06}$	$\mathbf{80.22 \pm 0.03}$
H-SNF	83.40 ± 0.01	80.29 ± 0.02
T-SNF	83.40 ± 0.10	80.28 ± 0.06

SYLVESTER FLOWS: RESULTS ON OTHER DATA

Model	Freyfaces		Omniglot		Caltech 101	
	-ELBO	NLL	-ELBO	NLL	-ELBO	NLL
VAE	4.53 ± 0.02	4.40 ± 0.03	104.28 ± 0.39	97.25 ± 0.23	110.80 ± 0.46	99.62 ± 0.74
Planar	$\mathbf{4.40 \pm 0.06}$	$\mathbf{4.31 \pm 0.06}$	102.65 ± 0.42	96.04 ± 0.28	109.66 ± 0.42	98.53 ± 0.68
IAF	4.47 ± 0.05	4.38 ± 0.04	102.41 ± 0.04	96.08 ± 0.16	111.58 ± 0.38	99.92 ± 0.30
O-SNF	4.51 ± 0.04	4.39 ± 0.05	99.00 ± 0.29	93.82 ± 0.21	106.08 ± 0.39	94.61 ± 0.83
H-SNF	4.46 ± 0.05	4.35 ± 0.05	$\mathbf{99.00 \pm 0.04}$	$\mathbf{93.77 \pm 0.03}$	$\mathbf{104.62 \pm 0.29}$	$\mathbf{93.82 \pm 0.62}$
T-SNF	4.45 ± 0.04	4.35 ± 0.04	99.33 ± 0.23	93.97 ± 0.13	105.29 ± 0.64	94.92 ± 0.73

No. of flows: 16
IAF: 1280 wide MADE, **no hypernets**
Bottleneck in O-SNF: 32
No. of Householder transformations in H-SNF: 8

COMPONENTS OF VAES

$$q_\phi(\mathbf{z}|\mathbf{x}) \propto p_\theta(\mathbf{x}|\mathbf{z})\; p_\lambda(\mathbf{z})$$

Normalizing flows
Discrete encoders
Hyperspherical dist.
Hyperbolic-normal dist.
Group theory

Resnets
DRAW
Autoregressive models
Normalizing flows

Autoregressive models
Normalizing flows
VampPrior
Implicit prior

$$\mathrm{ELBO}(\mathbf{x};\theta,\phi,\lambda)$$

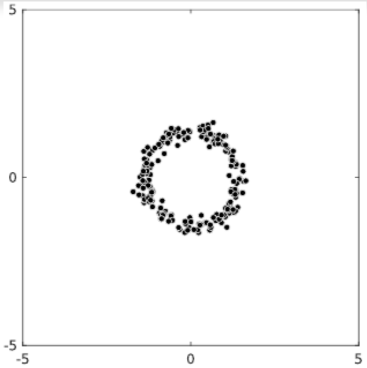
Adversarial learning
MMD
Wasserstein AE

GEOMETRIC PERSPECTIVE ON VAES

Question: Is it possible to recover the true Riemannian structure of the latent space?

In other words:

Will geodesics follow data manifold?



Arvanitidis, G., Hansen, L. K., & Hauberg, S. (2017). Latent space oddity: on the curvature of deep generative models. *arXiv preprint arXiv:1710.11379*.

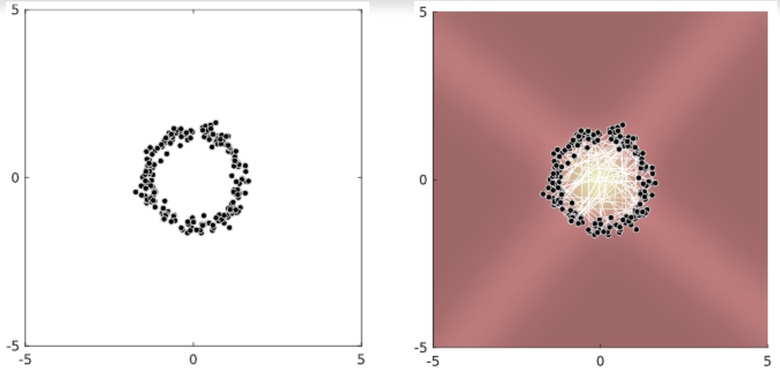
GEOMETRIC PERSPECTIVE ON VAES

Question: Is it possible to recover the true Riemannian structure of the latent space?

In other words:

Will geodesics follow data manifold?

For Gaussian VAE: **No**.



Arvanitidis, G., Hansen, L. K., & Hauberg, S. (2017). Latent space oddity: on the curvature of deep generative models. *arXiv preprint arXiv:1710.11379*.

GEOMETRIC PERSPECTIVE ON VAES

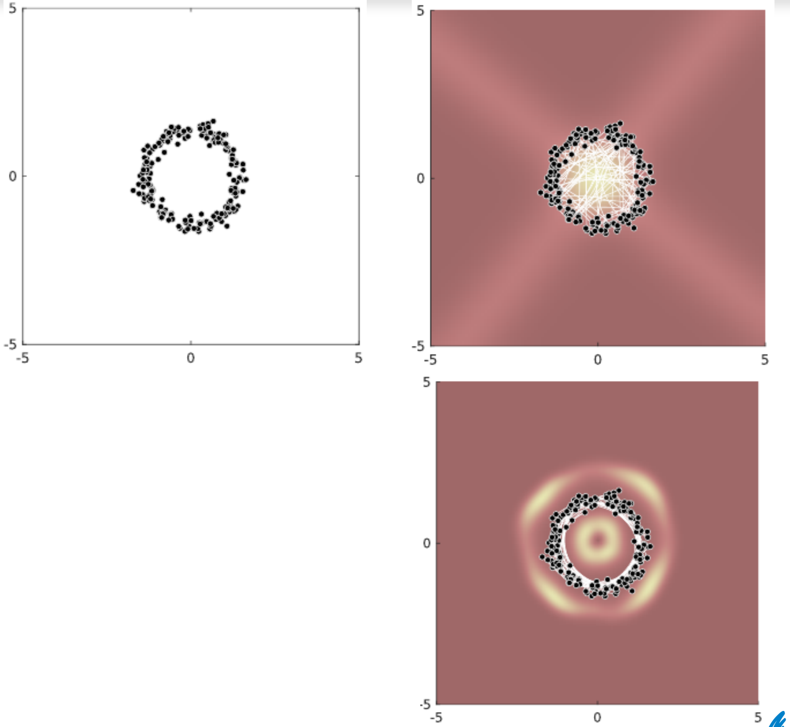
Question: Is it possible to recover the true Riemannian structure of the latent space?

In other words:

Will geodesics follow data manifold?

For Gaussian VAE: **No**.

We need a better notion of **uncertainty**



Arvanitidis, G., Hansen, L. K., & Hauberg, S. (2017). Latent space oddity: on the curvature of deep generative models. *arXiv preprint arXiv:1710.11379*.

GEOMETRIC PERSPECTIVE ON VAES

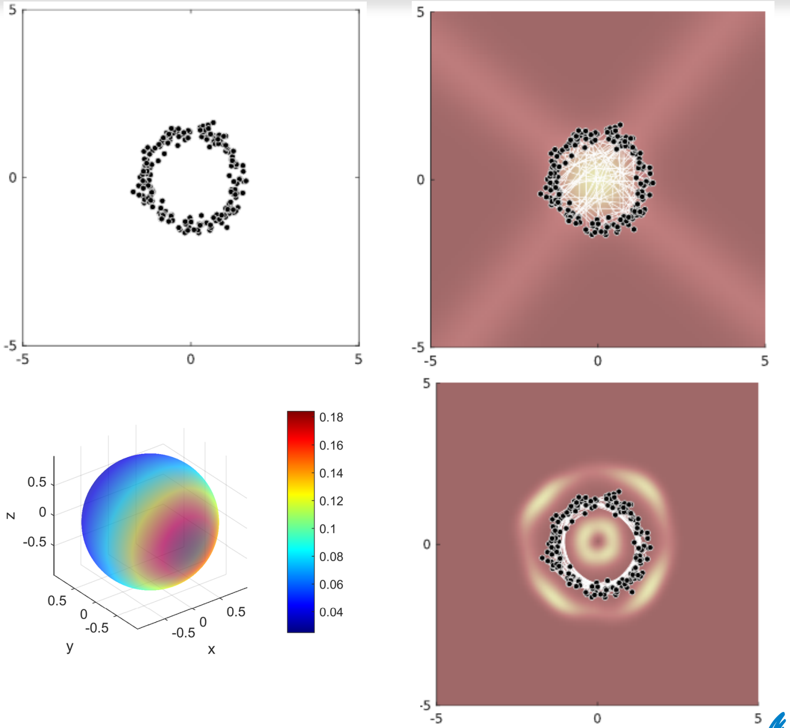
Question: Is it possible to recover the true Riemannian structure of the latent space?

In other words:

Will geodesics follow data manifold?

For Gaussian VAE: **No**.

We need a better notion of **uncertainty** or **different models**.



Arvanitidis, G., Hansen, L. K., & Hauberg, S. (2017). Latent space oddity: on the curvature of deep generative models. *arXiv preprint arXiv:1710.11379*.

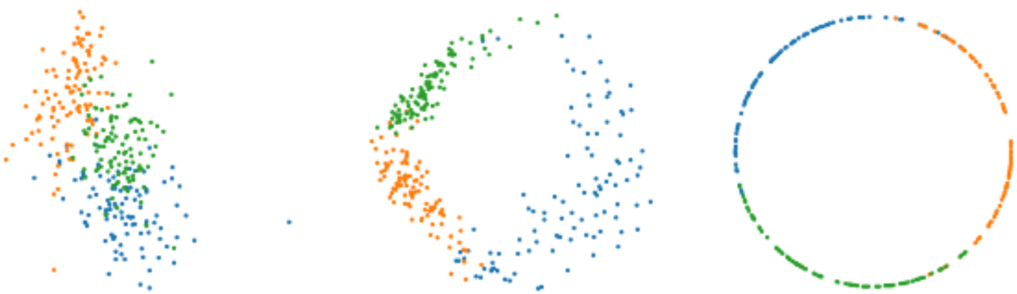
POTENTIAL PROBLEMS WITH GAUSSIANS

A common assumption in VAEs: distributions are Gaussians.

POTENTIAL PROBLEMS WITH GAUSSIANS

A common assumption in VAEs: distributions are Gaussians. But:

The Gaussian distr. is concentrated around the origin $\longrightarrow$ possible **bias**.



(a) Original (b) Autoencoder (c) $\mathcal{N}$-VAE (d) $\mathcal{N}$-VAE, $\beta = 0.1$ (e) $\mathcal{S}$-VAE

POTENTIAL PROBLEMS WITH GAUSSIANS

A common assumption in VAEs: distributions are Gaussians. But:

The Gaussian distr. is concentrated around the origin $\longrightarrow$ possible **bias**.

In high-dim, Gaussians concentrate on a hypersphere $\longrightarrow$ ℓ_2 norm **fails**.

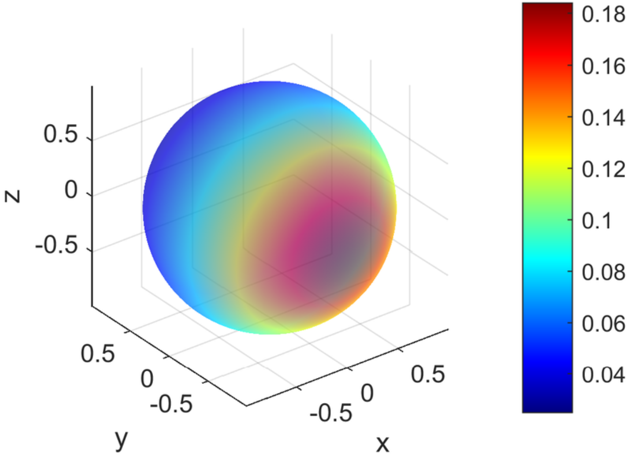
(a) Original (b) Autoencoder (c) $\mathcal{N}$-VAE (d) $\mathcal{N}$-VAE, $\beta = 0.1$ (e) $\mathcal{S}$-VAE

A HYPERSPHERICAL LATENT SPACE

Since in high-dim the Gaussian distribution concentrates on a **hypersphere**, we propose to use **von-Mises-Fisher** distribution defined on the hypersphere $\mathcal{S}^{m-1} \subset \mathbb{R}^m$

$$q(\mathbf{z}|\mu,\kappa) = \mathcal{C}_m(\kappa)\exp(\kappa\mu^\top\mathbf{z})$$

$$\mathcal{C}_m(\kappa) = \frac{\kappa^{m/2-1}}{(2\pi)^{m/2}\mathcal{I}_{m/2-1}(\kappa)}$$



where $||\mu||^2 = 1,\ \mathcal{I}_v$ is the modified Bessel function of the first kind of order *v*.

Davidson, T. R., Falorsi, L., De Cao, N., Kipf, T., & **Tomczak, J. M.** (2018). Hyperspherical Variational Auto-Encoders. *UAI 2018*

HYPERSPHERICAL VAES

The variational dist. is the **von-Mises-Fisher**, and the marginal is **uniform**, *i.e.*, von-Mises-Fisher with $\kappa = 0$. Then the **KL term** is as follows:

$$\mathrm{KL}(\mathrm{vMF}(\mu, \kappa) || \mathrm{U}(\mathcal{S}^{m-1})) = \kappa \frac{\mathcal{I}_{m/2}}{\mathcal{I}_{m/2-1}(\kappa)} + \log \mathcal{C}_m(\kappa) - \log \left(\frac{2\pi^{m/2}}{\Gamma(m/2)} \right)^{-1}$$

HYPERSPHERICAL VAES

The variational dist. is the **von-Mises-Fisher**, and the marginal is **uniform**, *i.e.*, von-Mises-Fisher with $\kappa = 0$. Then the **KL term** is as follows:

$$\mathrm{KL}(\mathrm{vMF}(\mu, \kappa) || \mathrm{U}(\mathcal{S}^{m-1})) = \kappa \frac{\mathcal{I}_{m/2}}{\mathcal{I}_{m/2-1}(\kappa)} + \log \mathcal{C}_m(\kappa) - \log \left(\frac{2\pi^{m/2}}{\Gamma(m/2)} \right)^{-1}$$

There exist an efficient **sampling procedure** using Householder transformation (Ulrich, 1984).

HYPERSPHERICAL VAES

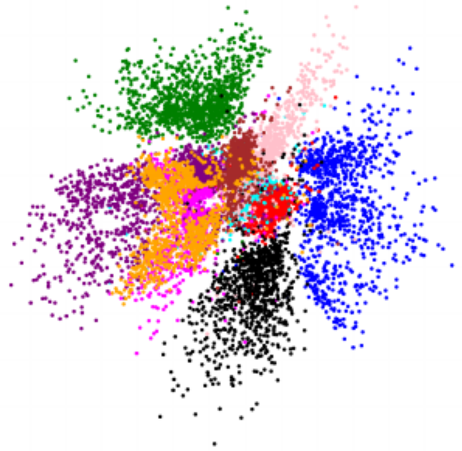
The variational dist. is the **von-Mises-Fisher**, and the marginal is **uniform**, *i.e.*, von-Mises-Fisher with $\kappa = 0$. Then the **KL term** is as follows:

$$\mathrm{KL}(\mathrm{vMF}(\mu,\kappa)||\mathrm{U}(\mathcal{S}^{m-1})) = \kappa \frac{\mathcal{I}_{m/2}}{\mathcal{I}_{m/2-1}(\kappa)} + \log \mathcal{C}_m(\kappa) - \log \left(\frac{2\pi^{m/2}}{\Gamma(m/2)} \right)^{-1}$$

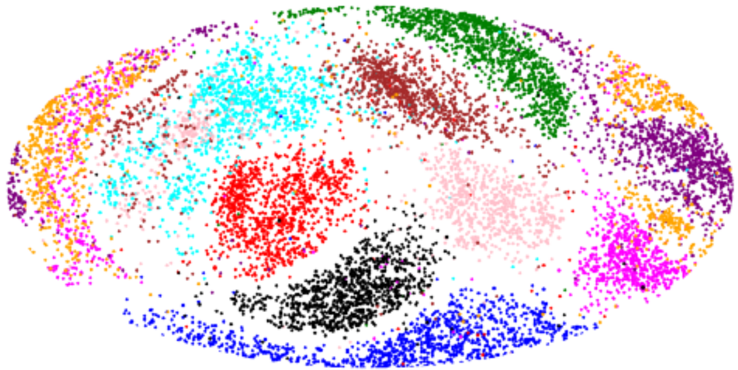
There exist an efficient **sampling procedure** using Householder transformation (Ulrich, 1984).

The reparameterization trick could be achieved by using the **rejection sampling** (Naesseth et al., 2017).

HYPERSPHERICAL VAE: RESULTS ON MNIST



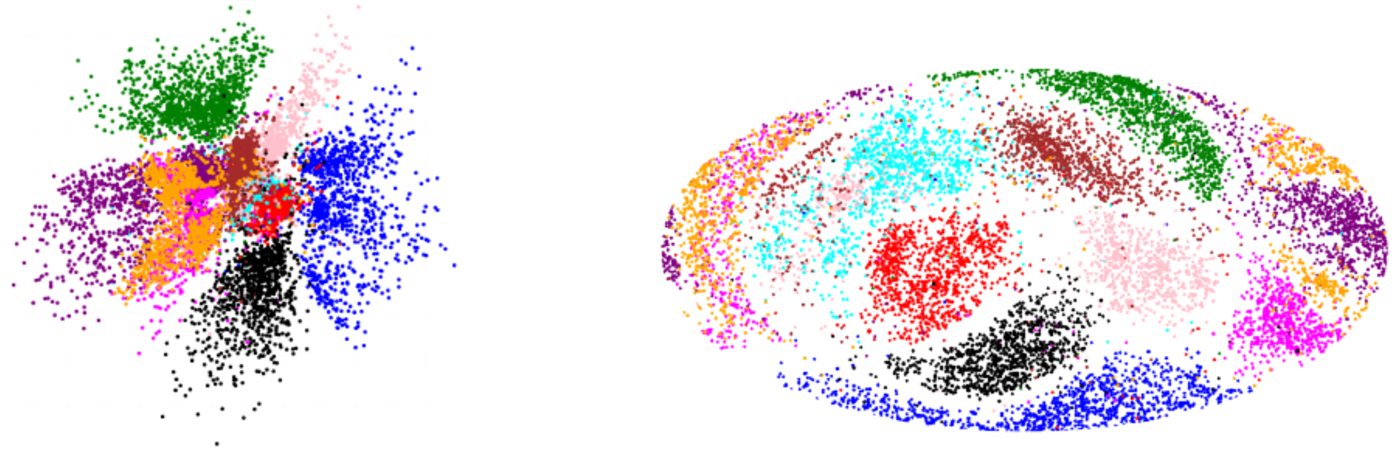
(a) $\mathbb{R}^2$ latent space of the $\mathcal{N}$-VAE.



(b) Hammer projection of $\mathcal{S}^2$ latent space of the $\mathcal{S}$-VAE.

Method	$\mathcal{N}$-VAE				$\mathcal{S}$-VAE			
	LL	$\mathcal{L}[q]$	RE	KL	LL	$\mathcal{L}[q]$	RE	KL
$d=2$	-135.73$\pm$.83	-137.08$\pm$.83	-129.84$\pm$.91	7.24$\pm$.11	**-132.50**$\pm$.73	-133.72$\pm$.85	-126.43$\pm$.91	7.28$\pm$.14
$d=5$	-110.21$\pm$.21	-112.98$\pm$.21	-100.16$\pm$.22	12.82$\pm$.11	**-108.43**$\pm$.09	-111.19$\pm$.08	-97.84$\pm$.13	13.35$\pm$.06
$d=10$	-93.84$\pm$.30	-98.36$\pm$.30	-78.93$\pm$.30	19.44$\pm$.14	**-93.16**$\pm$.31	-97.70$\pm$.32	-77.03$\pm$.39	20.67$\pm$.08
$d=20$	-88.90$\pm$.26	-94.79$\pm$.19	-71.29$\pm$.45	23.50$\pm$.31	-89.02$\pm$.31	-96.15$\pm$.32	-67.65$\pm$.43	28.50$\pm$.22
$d=40$	**-88.93**$\pm$.30	-94.91$\pm$.18	-71.14$\pm$.56	23.77$\pm$.49	-90.87$\pm$.34	-101.26$\pm$.33	-67.75$\pm$.70	33.50$\pm$.45

HYPERSPHERICAL VAE: RESULTS ON MNIST

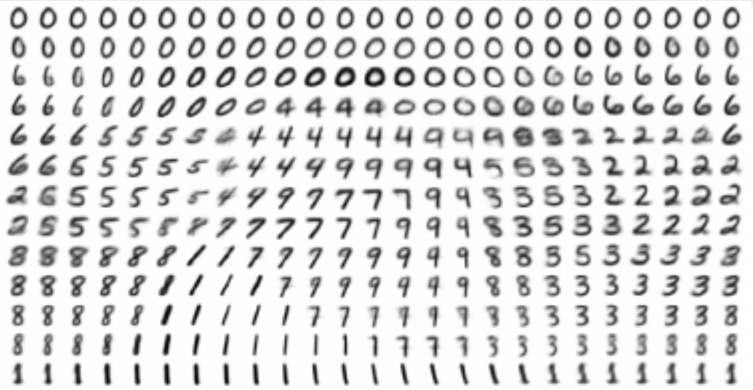


(a) $\mathbb{R}^2$ latent space of the $\mathcal{N}$-VAE. (b) Hammer projection of $\mathcal{S}^2$ latent space of the $\mathcal{S}$-VAE.

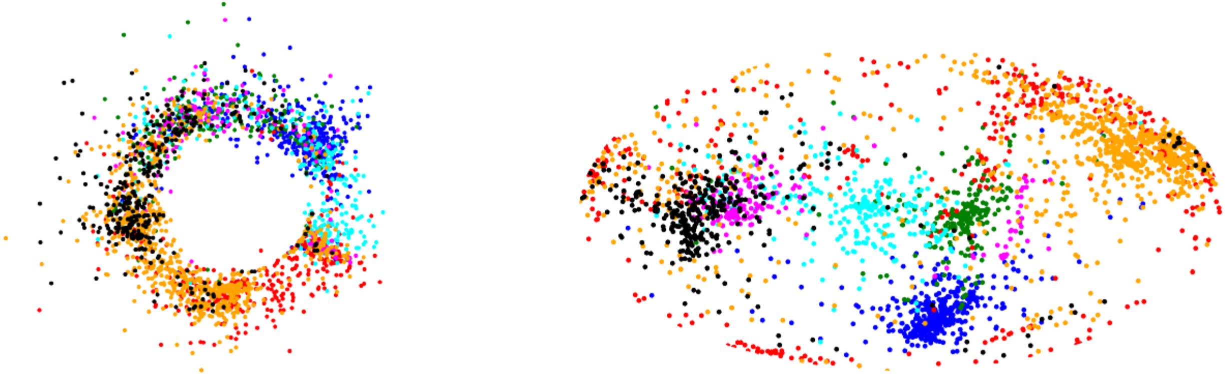
Method	$\mathcal{N}$-VAE				$\mathcal{S}$-VAE			
	LL	$\mathcal{L}[q]$	RE	KL	LL	$\mathcal{L}[q]$	RE	KL
$d=2$	-135.73$\pm$.83	-137.08$\pm$.83	-129.84$\pm$.91	7.24$\pm$.11	**-132.50**$\pm$.73	-133.72$\pm$.85	-126.43$\pm$.91	7.28$\pm$.14
$d=5$	-110.21$\pm$.21	-112.98$\pm$.21	-100.16$\pm$.22	12.82$\pm$.11	**-108.43**$\pm$.09	-111.19$\pm$.08	-97.84$\pm$.13	13.35$\pm$.06
$d=10$	-93.84$\pm$.30	-98.36$\pm$.30	-78.93$\pm$.30	19.44$\pm$.14	**-93.16**$\pm$.31	-97.70$\pm$.32	-77.03$\pm$.39	20.67$\pm$.08
$d=20$	-88.90$\pm$.26	-94.79$\pm$.19	-71.29$\pm$.45	23.50$\pm$.31	-89.02$\pm$.31	-96.15$\pm$.32	-67.65$\pm$.43	28.50$\pm$.22
$d=40$	**-88.93**$\pm$.30	-94.91$\pm$.18	-71.14$\pm$.56	23.77$\pm$.49	-90.87$\pm$.34	-101.26$\pm$.33	-67.75$\pm$.70	33.50$\pm$.45

HYPERSPHERICAL VAE: RESULTS ON SEMI-SUPERVISED MNIST

Method		**100**		
dim $\mathbf{z}_1$	*dim* $\mathbf{z}_2$	$\mathcal{N}$+$\mathcal{N}$	$\mathcal{S}$+$\mathcal{S}$	$\mathcal{S}$+$\mathcal{N}$
5	5	90.0±.4	**94.0**±.1	93.8±.1
	10	90.7±.3	94.1±.1	**94.8**±.2
	50	90.7±.1	92.7±.2	**93.0**±.1
10	5	90.7±.3	91.7±.5	**94.0**±.4
	10	92.2±.1	**96.0**±.2	**95.9**±.3
	50	92.9±.4	95.1±.2	**95.7**±.1
50	5	92.0±.2	91.7±.4	**95.8**±.1
	10	93.0±.1	95.8±.1	**97.1**±.1
	50	93.2±.2	94.2±.1	**97.4**±.1



HYPERSPHERICAL GRAPHVAE: LINK PREDICTION



(a) $\mathbb{R}^2$ latent space of the $\mathcal{N}$-VGAE.

(b) Hammer projection of $\mathcal{S}^2$ latent space of the $\mathcal{S}$-VGAE.

Method		**$\mathcal{N}$-VGAE**	**$\mathcal{S}$-VGAE**
Cora	AUC	92.7$\pm$.2	**94.1**$\pm$.1
	AP	93.2$\pm$.4	**94.1**$\pm$.3
Citeseer	AUC	90.3$\pm$.5	**94.7**$\pm$.2
	AP	91.5$\pm$.5	**95.2**$\pm$.2
Pubmed	AUC	**97.1**$\pm$.0	96.0$\pm$.1
	AP	**97.1**$\pm$.0	96.0$\pm$.1

COMPONENTS OF VAES

$$q_\phi(\mathbf{z}|\mathbf{x}) \propto p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})$$

Normalizing flows
Discrete encoders
Hyperspherical dist.
Hyperbolic-normal dist.
Group theory

Resnets
DRAW
Autoregressive models
Normalizing flows

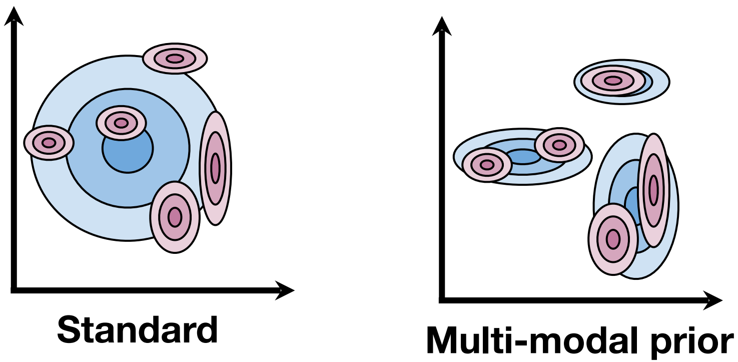
Autoregressive models
Normalizing flows
VampPrior
Implicit prior

$$\mathrm{ELBO}(\mathbf{x};\theta,\phi,\lambda)$$

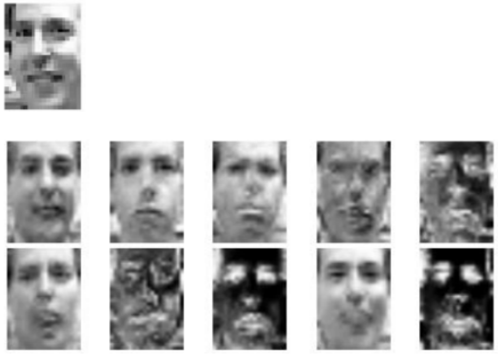
Adversarial learning
MMD
Wasserstein AE

PROBLEMS OF *HOLES* IN VAES

There is a discrepancy between posteriors and the Gaussian prior that results in regions that were never "seen" by the posterior (**holes**). → **multi-modal prior**



Sampling process could produce **unrealistic** samples.



Rezende, D.J. and Viola, F., 2018. Taming VAEs. *arXiv preprint arXiv:1810.00597*.

LOOKING FOR THE OPTIMAL PRIOR

Let's rewrite ELBO over the training data:

$$\mathbb{E}_{\mathbf{x}\sim p_{\mathcal{D}}(\mathbf{x})}\left[\log p_{\vartheta}(\mathbf{x})\right] \geq \mathbb{E}_{p_{\mathcal{D}}(\mathbf{x})q_{\phi}(\mathbf{z}|\mathbf{x})}\left[\log p_{\theta}(\mathbf{x}|\mathbf{z})\right] - \mathbb{I}_{\mathcal{D}}(\mathbf{x};\mathbf{z}) - \mathrm{KL}\left(q_{\phi,\mathcal{D}}(\mathbf{z})\|p_{\lambda}(\mathbf{z})\right)$$

Tomczak, J.M., Welling, M. (2018), VAE with a VampPrior, AISTATS 2018

LOOKING FOR THE OPTIMAL PRIOR

Let's rewrite ELBO over the training data:

$$q_{\phi,\mathcal{D}}(\mathbf{z}) = \frac{1}{N}\sum_{n} q_{\phi}(\mathbf{z}|\mathbf{x}_n)$$

$$\mathbb{E}_{\mathbf{x}\sim p_{\mathcal{D}}(\mathbf{x})}\left[\log p_{\vartheta}(\mathbf{x})\right] \geq \mathbb{E}_{p_{\mathcal{D}}(\mathbf{x})q_{\phi}(\mathbf{z}|\mathbf{x})}\left[\log p_{\theta}(\mathbf{x}|\mathbf{z})\right] - \mathbb{I}_{\mathcal{D}}(\mathbf{x};\mathbf{z}) - \mathrm{KL}\left(q_{\phi,\mathcal{D}}(\mathbf{z})\|p_{\lambda}(\mathbf{z})\right)$$

Tomczak, J.M., Welling, M. (2018), VAE with a VampPrior, AISTATS 2018

LOOKING FOR THE OPTIMAL PRIOR

Let's rewrite ELBO over the training data:

$$q_{\phi,\mathcal{D}}(\mathbf{z}) = \frac{1}{N}\sum_{n} q_{\phi}(\mathbf{z}|\mathbf{x}_n)$$

$$\mathbb{E}_{\mathbf{x}\sim p_{\mathcal{D}}(\mathbf{x})}\left[\log p_{\vartheta}(\mathbf{x})\right] \geq \mathbb{E}_{p_{\mathcal{D}}(\mathbf{x})q_{\phi}(\mathbf{z}|\mathbf{x})}\left[\log p_{\theta}(\mathbf{x}|\mathbf{z})\right] - \mathbb{I}_{\mathcal{D}}(\mathbf{x};\mathbf{z}) - \mathrm{KL}\left(q_{\phi,\mathcal{D}}(\mathbf{z})\|p_{\lambda}(\mathbf{z})\right)$$

KL = 0 iff $q_{\phi,\mathcal{D}}(\mathbf{z}) = p_{\lambda}(\mathbf{z})$, then the optimal prior = **aggregated posterior**.

Tomczak, J.M., Welling, M. (2018), VAE with a VampPrior, AISTATS 2018

LOOKING FOR THE OPTIMAL PRIOR

Let's rewrite ELBO over the training data:

$$q_{\phi,\mathcal{D}}(\mathbf{z}) = \frac{1}{N}\sum_{n} q_{\phi}(\mathbf{z}|\mathbf{x}_n)$$

$$\mathbb{E}_{\mathbf{x}\sim p_{\mathcal{D}}(\mathbf{x})}\left[\log p_{\vartheta}(\mathbf{x})\right] \geq \mathbb{E}_{p_{\mathcal{D}}(\mathbf{x})q_{\phi}(\mathbf{z}|\mathbf{x})}\left[\log p_{\theta}(\mathbf{x}|\mathbf{z})\right] - \mathbb{I}_{\mathcal{D}}(\mathbf{x};\mathbf{z}) - \mathrm{KL}\left(q_{\phi,\mathcal{D}}(\mathbf{z})\|p_{\lambda}(\mathbf{z})\right)$$

KL = 0 iff $q_{\phi,\mathcal{D}}(\mathbf{z}) = p_{\lambda}(\mathbf{z})$, then the optimal prior = **aggregated posterior**.

Summing over all training data is infeasible and since the sample is finite, it could cause some additional instabilities.

Tomczak, J.M., Welling, M. (2018), VAE with a VampPrior, AISTATS 2018

LOOKING FOR THE OPTIMAL PRIOR

Let's rewrite ELBO over the training data:

$$q_{\phi,\mathcal{D}}(\mathbf{z}) = \frac{1}{N}\sum_{n} q_{\phi}(\mathbf{z}|\mathbf{x}_n)$$

$$\mathbb{E}_{\mathbf{x}\sim p_{\mathcal{D}}(\mathbf{x})}\left[\log p_{\vartheta}(\mathbf{x})\right] \geq \mathbb{E}_{p_{\mathcal{D}}(\mathbf{x})q_{\phi}(\mathbf{z}|\mathbf{x})}\left[\log p_{\theta}(\mathbf{x}|\mathbf{z})\right] - \mathbb{I}_{\mathcal{D}}(\mathbf{x};\mathbf{z}) - \mathrm{KL}\left(q_{\phi,\mathcal{D}}(\mathbf{z})\|p_{\lambda}(\mathbf{z})\right)$$

KL = 0 iff $q_{\phi,\mathcal{D}}(\mathbf{z}) = p_{\lambda}(\mathbf{z})$, then the optimal prior = **aggregated posterior**.

Summing over all training data is infeasible and since the sample is finite, it could cause some additional instabilities. Instead we propose to use:

$$p_{\lambda}(\mathbf{z}) = \frac{1}{K}\sum_{k=1}^{K} q_{\phi}(\mathbf{z}|\mathbf{u}_k)$$

Tomczak, J.M., Welling, M. (2018), VAE with a VampPrior, AISTATS 2018

LOOKING FOR THE OPTIMAL PRIOR

Let's rewrite ELBO over the training data:

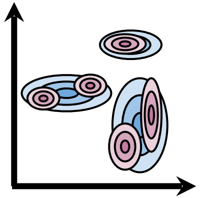
$$q_{\phi,\mathcal{D}}(\mathbf{z}) = \frac{1}{N}\sum_{n} q_{\phi}(\mathbf{z}|\mathbf{x}_n)$$

$$\mathbb{E}_{\mathbf{x}\sim p_{\mathcal{D}}(\mathbf{x})}\left[\log p_{\vartheta}(\mathbf{x})\right] \geq \mathbb{E}_{p_{\mathcal{D}}(\mathbf{x})q_{\phi}(\mathbf{z}|\mathbf{x})}\left[\log p_{\theta}(\mathbf{x}|\mathbf{z})\right] - \mathbb{I}_{\mathcal{D}}(\mathbf{x};\mathbf{z}) - \mathrm{KL}\left(q_{\phi,\mathcal{D}}(\mathbf{z}) \| p_{\lambda}(\mathbf{z})\right)$$

KL = 0 iff $q_{\phi,\mathcal{D}}(\mathbf{z}) = p_{\lambda}(\mathbf{z})$, then the optimal prior = **aggregated posterior**.

Summing over all training data is infeasible and since the sample is finite, it could cause some additional instabilities. Instead we propose to use:

$$p_{\lambda}(\mathbf{z}) = \frac{1}{K}\sum_{k=1}^{K} q_{\phi}(\mathbf{z}|\mathbf{u}_k)$$



Multi-modal

Tomczak, J.M., Welling, M. (2018), VAE with a VampPrior, AISTATS 2018

VU

LOOKING FOR THE OPTIMAL PRIOR

Let's rewrite ELBO over the training data:

$$q_{\phi,\mathcal{D}}(\mathbf{z}) = \frac{1}{N}\sum_{n} q_{\phi}(\mathbf{z}|\mathbf{x}_n)$$

$$\mathbb{E}_{\mathbf{x}\sim p_{\mathcal{D}}(\mathbf{x})}\left[\log p_{\vartheta}(\mathbf{x})\right] \geq \mathbb{E}_{p_{\mathcal{D}}(\mathbf{x})q_{\phi}(\mathbf{z}|\mathbf{x})}\left[\log p_{\theta}(\mathbf{x}|\mathbf{z})\right] - \mathbb{I}_{\mathcal{D}}(\mathbf{x};\mathbf{z}) - \mathrm{KL}\left(q_{\phi,\mathcal{D}}(\mathbf{z})\|p_{\lambda}(\mathbf{z})\right)$$

KL = 0 iff $q_{\phi,\mathcal{D}}(\mathbf{z}) = p_{\lambda}(\mathbf{z})$, then the optimal prior = **aggregated posterior**.

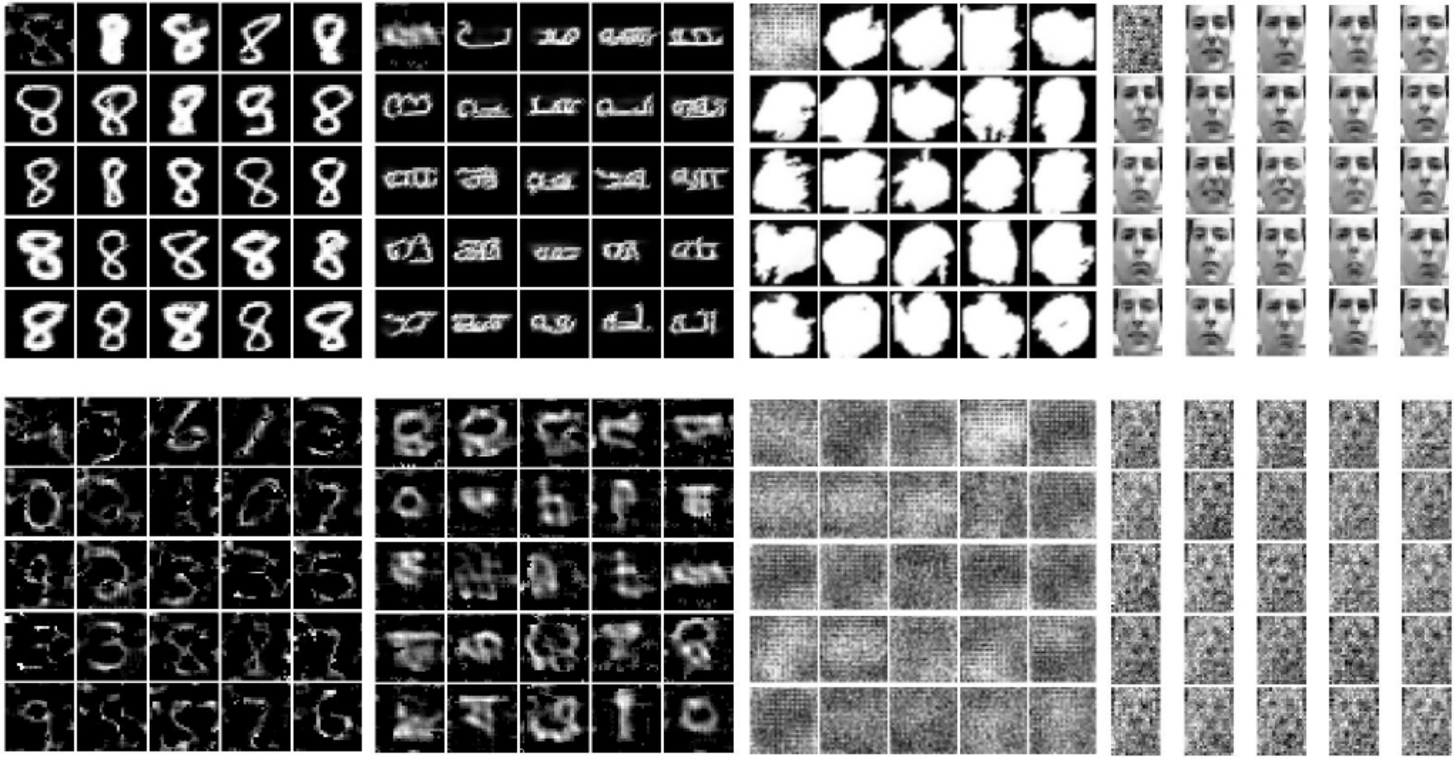
Summing over all training data is infeasible and since the sample is finite, it could cause some additional instabilities. Instead we propose to use:

$$p_{\lambda}(\mathbf{z}) = \frac{1}{K}\sum_{k=1}^{K} q_{\phi}(\mathbf{z}|\mathbf{u}_k)$$

pseudoinputs are trained from scratch by SGD

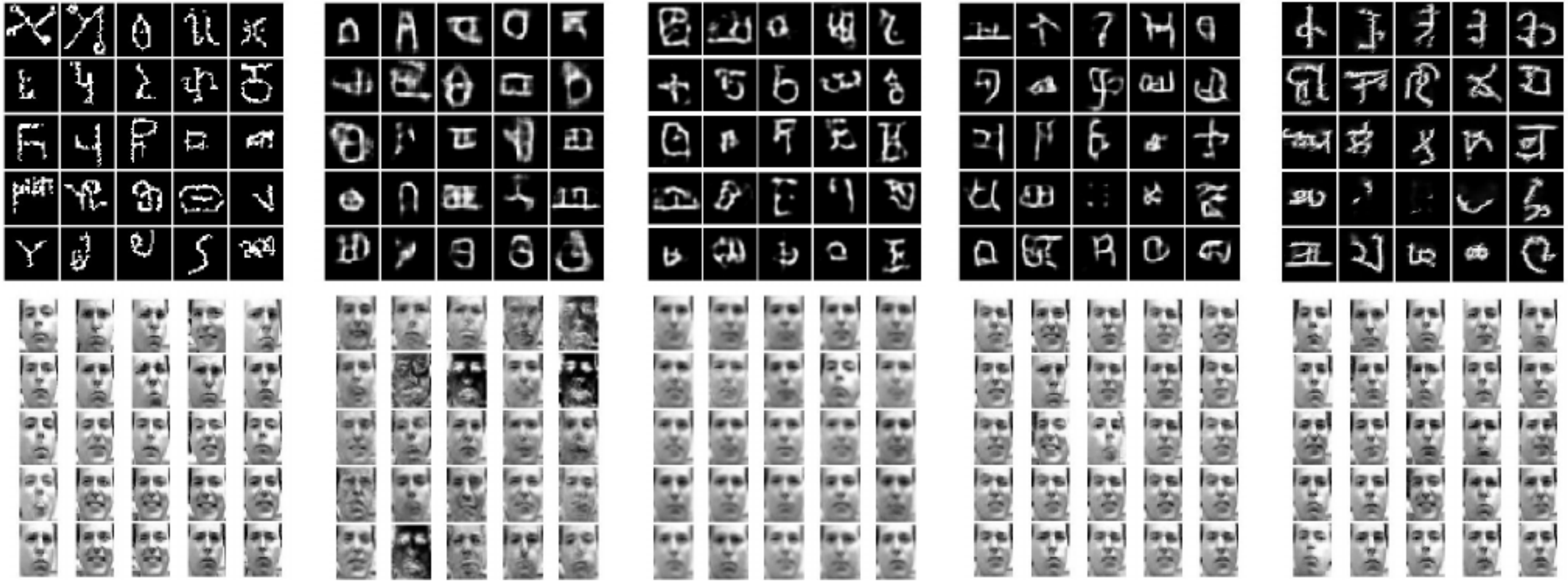
Tomczak, J.M., Welling, M. (2018), VAE with a VampPrior, AISTATS 2018

VAMPPRIOR: EXPERIMENTS (PSEUDOINPUTS)

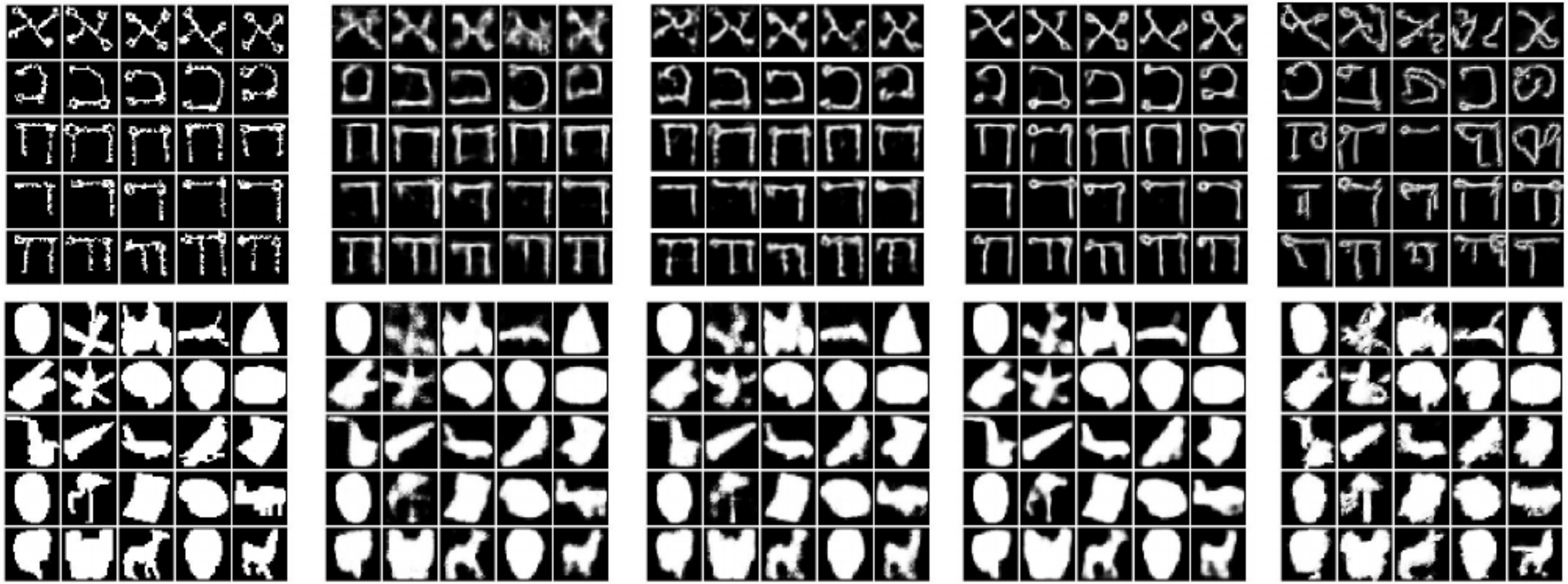


MNIST Omniglot Caltech 101 Silhouettes Frey Faces

VAMPPRIOR: EXPERIMENTS (SAMPLES)


(a) real data (b) VAE (c) HVAE + VampPrior (d) convHVAE + VampPrior (e) PixelHVAE + VampPrior

VAMPPRIOR: EXPERIMENTS (RECONSTRUCTIONS)



(a) real data (b) VAE (c) HVAE + VampPrior (d) convHVAE + VampPrior (e) PixelHVAE + VampPrior

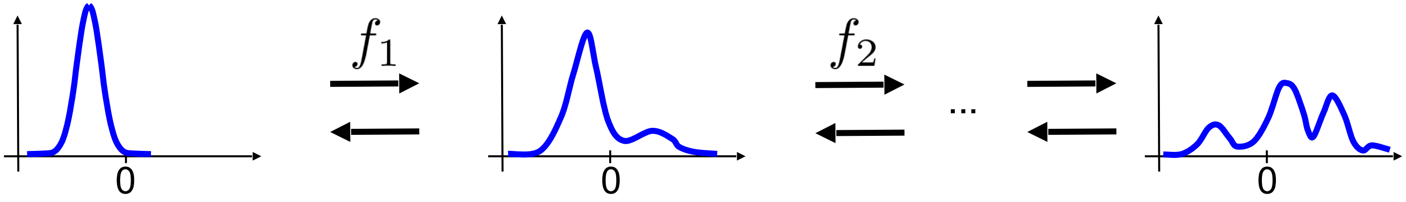
Flow-based models

THE CHANGE OF VARIABLES FORMULA

Let’s recall the change of variables formula with invertible transformations:

$$p(\mathbf{x}) = \pi_0\left(\mathbf{z}_0\right) \prod_{i=1}^{K} \left| \det \frac{\partial f_i(\mathbf{z}_{i-1})}{\partial \mathbf{z}_{i-1}} \right|^{-1}$$

We can think of it as an invertible neural network:



latent space

pixel space

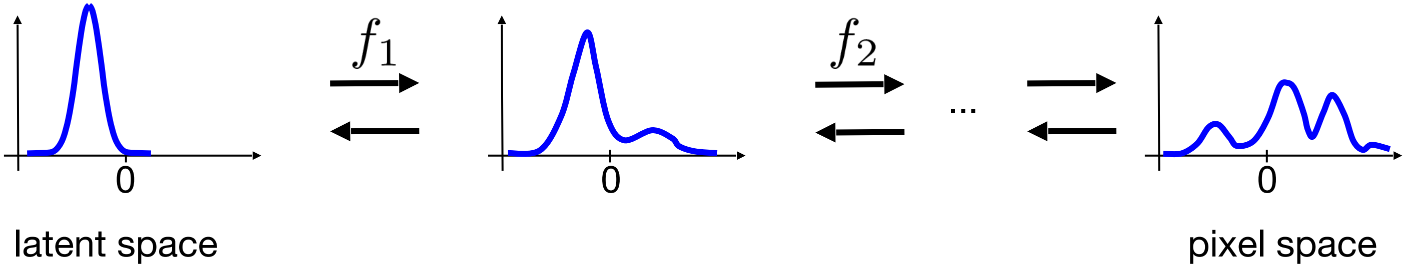
Rippel, O., & Adams, R. P. (2013). High-dimensional probability estimation with deep density models. arXiv preprint arXiv:1302.5125.

THE CHANGE OF VARIABLES FORMULA

Let's recall the change of variables formula with invertible transformations:

$$p(\mathbf{x}) = \pi_0\left(\mathbf{z}_0\right) \prod_{i=1}^{K} \left|\det \frac{\partial f_i(\mathbf{z}_{i-1})}{\partial \mathbf{z}_{i-1}}\right|^{-1}$$

We can think of it as an invertible neural network:



Rippel, O., & Adams, R. P. (2013). High-dimensional probability estimation with deep density models. arXiv preprint arXiv:1302.5125.

REALNVP

Design the invertible transformations as follows:

$$\mathbf{y}_{1:d} = \mathbf{x}_{1:d}$$
$$\mathbf{y}_{d+1:D} = \mathbf{x}_{d+1:D} \odot \exp\left(s\left(\mathbf{x}_{1:d}\right)\right) + t\left(\mathbf{x}_{1:d}\right)$$

Dinh, L., Sohl-Dickstein, J., & Bengio, S. (2016). Density estimation using real nvp. arXiv preprint arXiv:1605.08803.

REALNVP

Design the invertible transformations as follows:

$$\mathbf{y}_{1:d} = \mathbf{x}_{1:d}$$

$$\mathbf{y}_{d+1:D} = \mathbf{x}_{d+1:D} \odot \exp\left(s\left(\mathbf{x}_{1:d}\right)\right) + t\left(\mathbf{x}_{1:d}\right)$$

Invertible by design:

$$\begin{cases} \mathbf{y}_{1:d} &= \mathbf{x}_{1:d} \\ \mathbf{y}_{d+1:D} &= \mathbf{x}_{d+1:D} \odot \exp\left(s\left(\mathbf{x}_{1:d}\right)\right) + t\left(\mathbf{x}_{1:d}\right) \end{cases} \Leftrightarrow \begin{cases} \mathbf{x}_{1:d} &= \mathbf{y}_{1:d} \\ \mathbf{x}_{d+1:D} &= \left(\mathbf{y}_{d+1:D} - t\left(\mathbf{y}_{1:d}\right)\right) \odot \exp\left(-s\left(\mathbf{y}_{1:d}\right)\right) \end{cases}$$

Dinh, L., Sohl-Dickstein, J., & Bengio, S. (2016). Density estimation using real nvp. arXiv preprint arXiv:1605.08803.

REALNVP

Design the invertible transformations as follows:

$$\mathbf{y}_{1:d} = \mathbf{x}_{1:d}$$
$$\mathbf{y}_{d+1:D} = \mathbf{x}_{d+1:D} \odot \exp\left(s\left(\mathbf{x}_{1:d}\right)\right) + t\left(\mathbf{x}_{1:d}\right)$$

Invertible by design:

$$\begin{cases} \mathbf{y}_{1:d} & = \mathbf{x}_{1:d} \\ \mathbf{y}_{d+1:D} & = \mathbf{x}_{d+1:D} \odot \exp\left(s\left(\mathbf{x}_{1:d}\right)\right) + t\left(\mathbf{x}_{1:d}\right) \end{cases} \Leftrightarrow \begin{cases} \mathbf{x}_{1:d} & = \mathbf{y}_{1:d} \\ \mathbf{x}_{d+1:D} & = \left(\mathbf{y}_{d+1:D} - t\left(\mathbf{y}_{1:d}\right)\right) \odot \exp\left(-s\left(\mathbf{y}_{1:d}\right)\right) \end{cases}$$

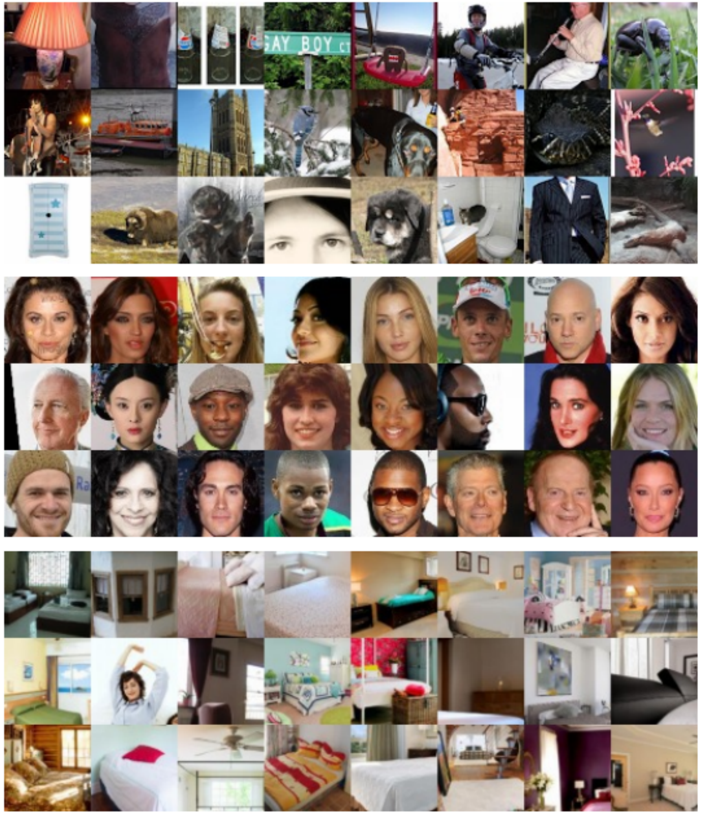
Easy Jacobian:

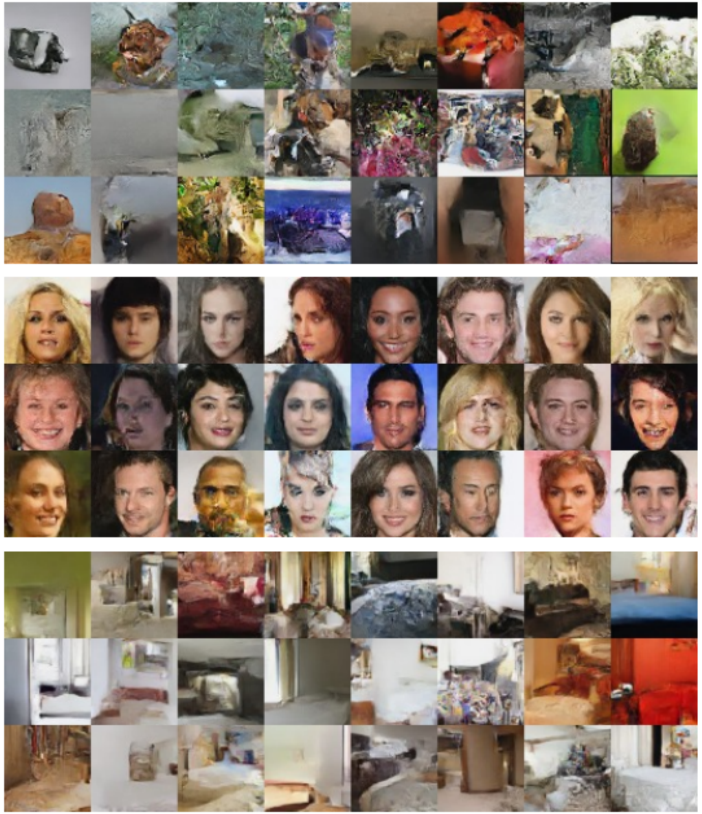
$$\mathbf{J} = \begin{bmatrix} \mathbb{I}_d & \mathbf{0}_{d \times (D-d)} \\ \frac{\partial \mathbf{y}_{d+1:D}}{\partial \mathbf{x}_{1:d}} & \operatorname{diag}\left(\exp\left(s\left(\mathbf{x}_{1:d}\right)\right)\right) \end{bmatrix} \qquad \det(\mathbf{J}) = \prod_{j=1}^{D-d} \exp\left(s\left(\mathbf{x}_{1:d}\right)\right)_j = \exp\left(\sum_{j=1}^{D-d} s\left(\mathbf{x}_{1:d}\right)_j\right)$$

Dinh, L., Sohl-Dickstein, J., & Bengio, S. (2016). Density estimation using real nvp. arXiv preprint arXiv:1605.08803.

RESULTS

RESULTS

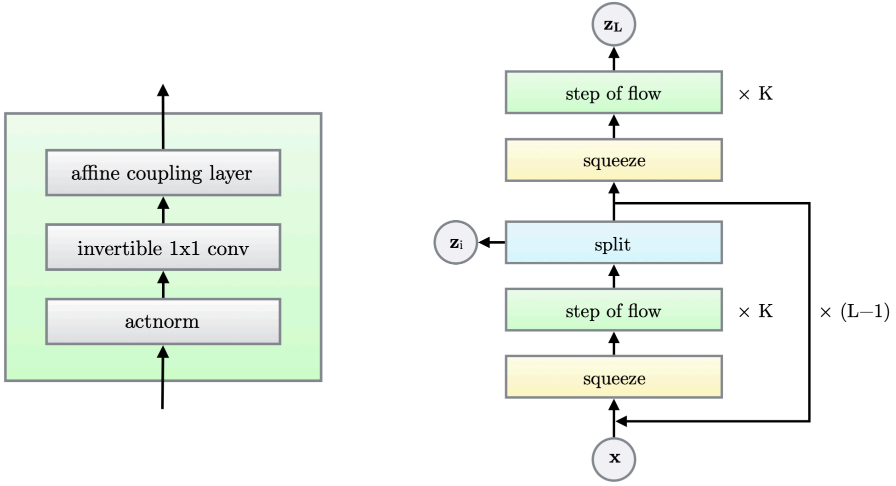




GLOW: REALNVP WITH 1X1 CONVOLUTIONS

A model contains ~1000 convolutions.

A new component: 1x1 convolution instead of a permutation matrix.

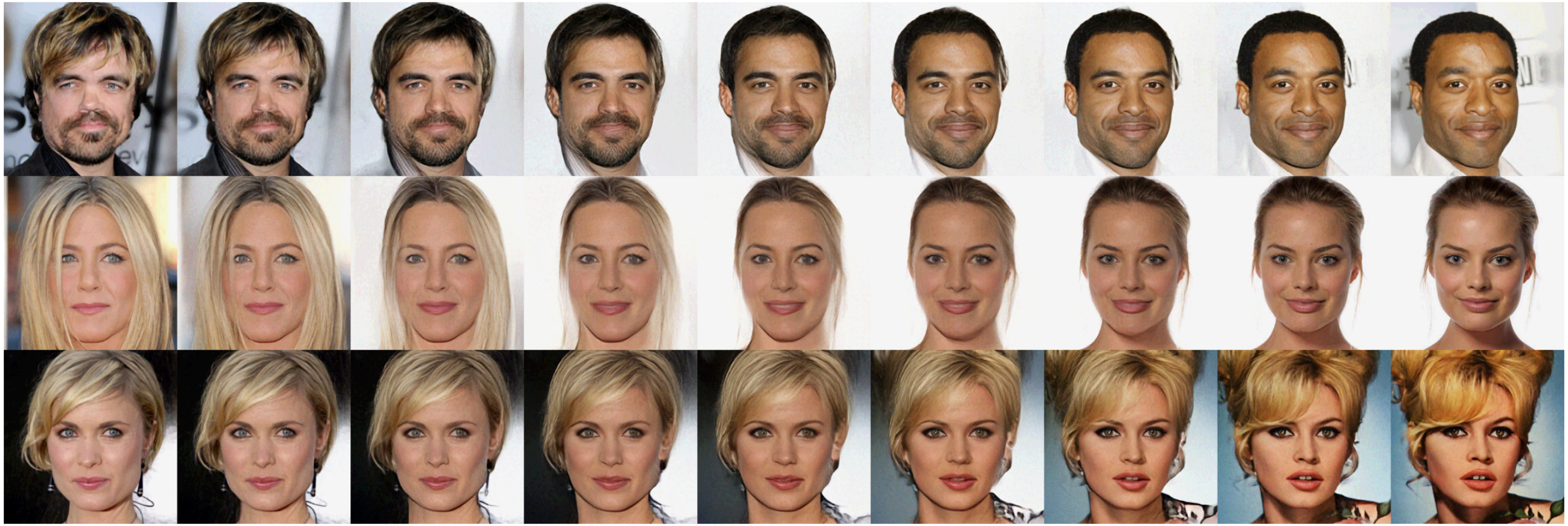


(a) One step of our flow. (b) Multi-scale architecture (Dinh et al., 2016).

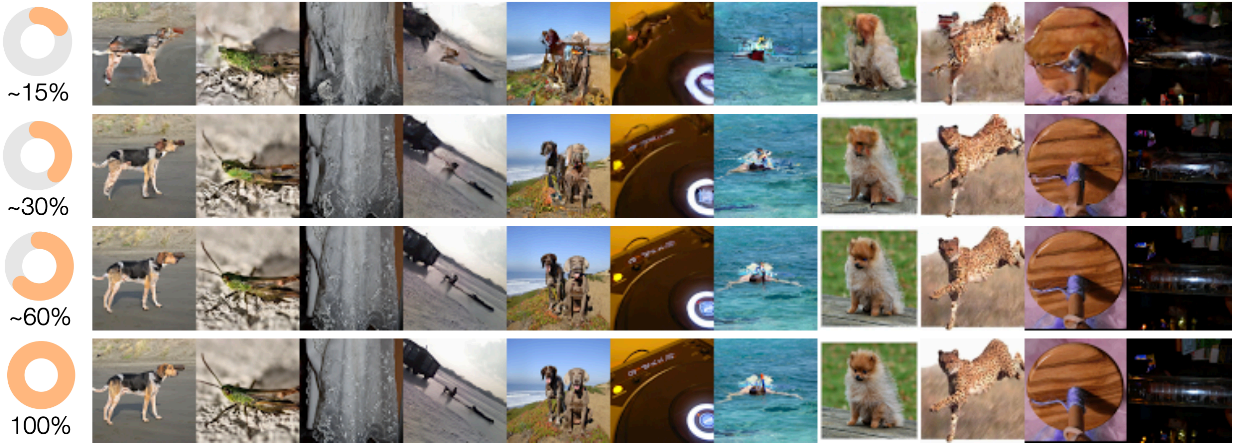
Kingma, D. P., & Dhariwal, P. (2018). Glow: Generative flow with invertible 1x1 convolutions. *NeurIPS 2018*

GLOW: SAMPLES

GLOW: LATENT INTERPOLATION



INTEGER DISCRETE FLOW: NO NEED TO CALCULATE JACOBIAN!



Hoogeboom, E., Peters, J. W., Berg, R. V. D., & Welling, M. (2019). Integer Discrete Flows and Lossless Compression. *NeurIPS 2019*

Future directions

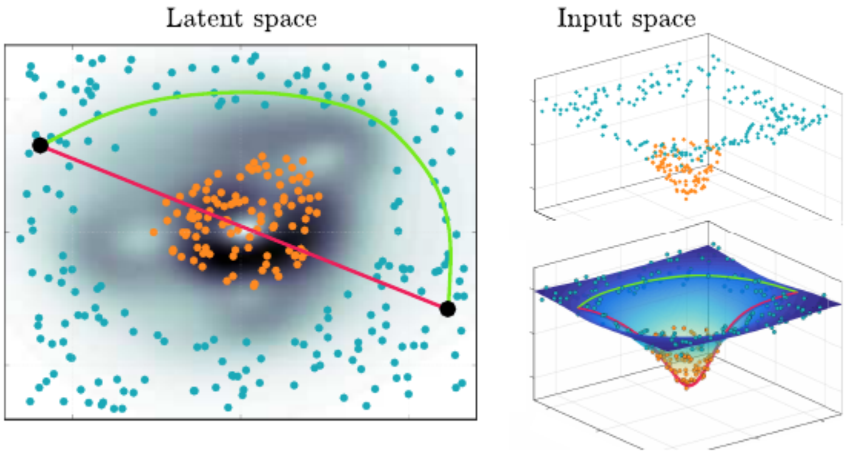
BLURRINESS AND SAMPLING IN VAES

How to avoid sampling from **holes**?

Should we follow **geodesics in the latent space**?

How to use **geometry** of the latent space to build better **decoders**?

How to build **temporal decoders**?



COMPRESSION AND VAES

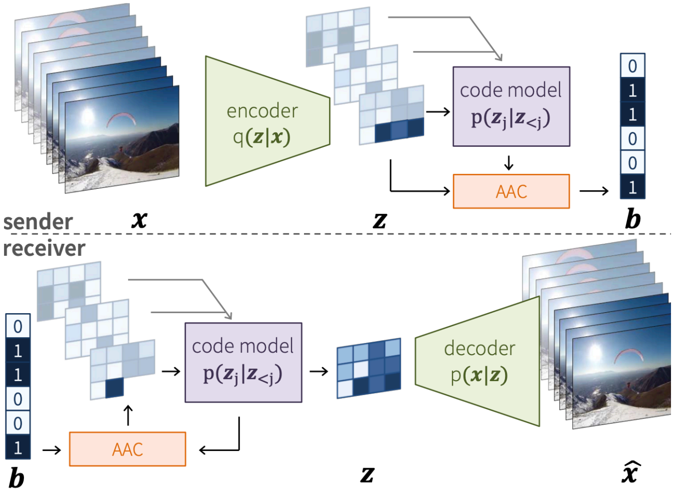
Taking a **deterministic & discrete encoder** allows to simplify the objective.

$$
\begin{aligned}
\mathrm{RE}(x|z) - \mathrm{H}[q(z|x)] - \mathrm{CE}[q(z)||p(z)] \\
= \mathrm{RE}(x|z) - \mathrm{CE}[q(z)||p(z)]
\end{aligned}
$$

It is important to learn a **powerful prior**. This is challenging!

Is it **easier** to learn a prior with **temporal dependencies**?

Can we alleviate some dependencies by using **hypernets**?



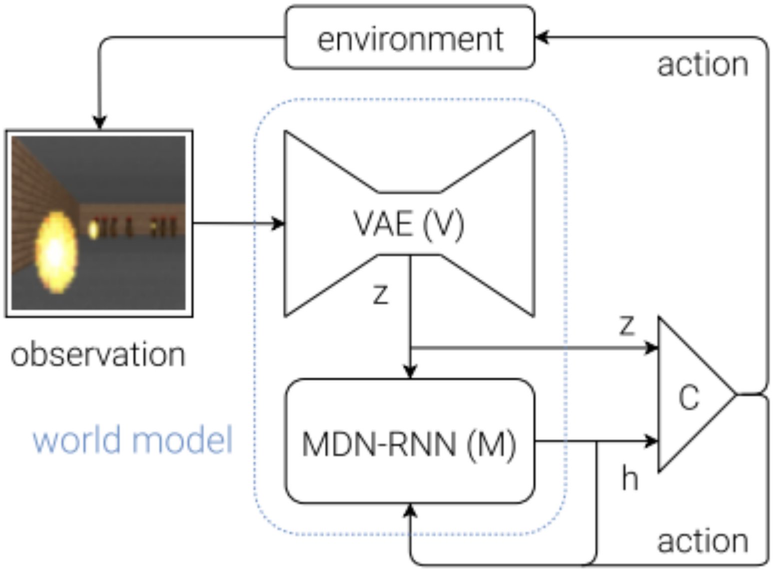
Habibian, A., van Rozendaal, T., Tomczak, J.M., & Cohen, T.S. (2019), Video Compression with Rate-Distortion Autoencoders, ICCV 2019

ACTIVE LEARNING/RL AND VAES

Using **latent representation** to navigate and/or quantify uncertainty.

Formulating **policies** in the latent space entirely.

Do we need a better notion of **sequential dependencies**?



Ha, D., & Schmidhuber, J. (2018). World models. *arXiv preprint arXiv:1803.10122*.

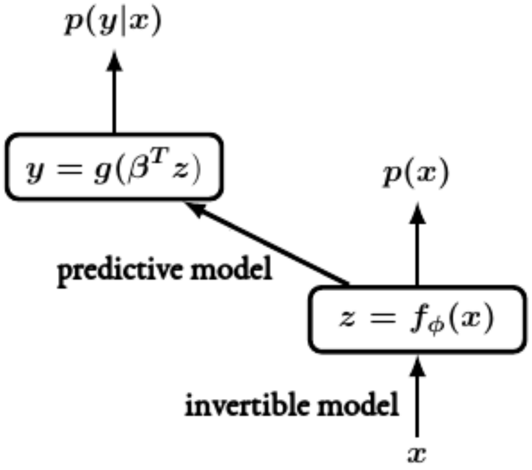
HYBRID AND FLOW-BASED MODELS

We need a **better understanding** of the latent space.

Joining an **invertible model** (flow-based model) with a **predictive model**.

Isn't this model an **overkill**?

How would it work in the **multi-modal learning** scenario?



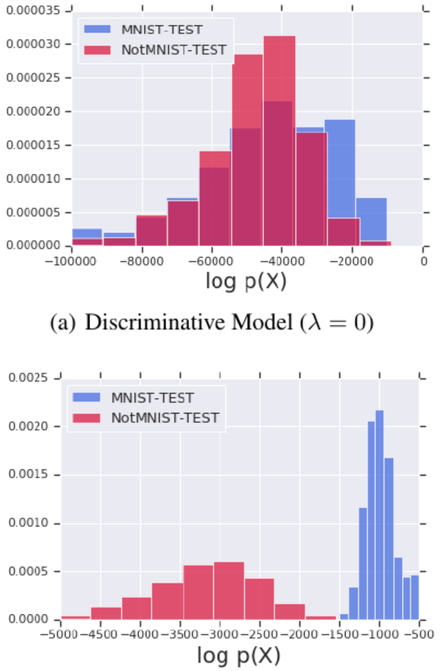
Nalisnick, E., et al. (2019). Hybrid models with deep and invertible features. *arXiv preprint arXiv:1902.02767*.

HYBRID MODELS AND OOD SAMPLE

Going back to first slides, we need a good notion of **p(x)**.

Distinguishing **out-of-distribution (OOD)** samples is very important.

Crucial for **decision making**, **outlier detection**, **policy learning**…



(a) Discriminative Model ($\lambda = 0$)

(b) Hybrid Model

Nalisnick, Eric, et al. "Hybrid models with deep and invertible features." *arXiv preprint arXiv:1902.02767* (2019).

Thank you!

VRIJE
UNIVERSITEIT
AMSTERDAM

Webpage:
https://jmtomczak.github.io/

Code on github:
https://github.com/jmtomczak/

Contact:
jmk.tomczak@gmail.com