uc3m

Deep Learning with Neural Networks

Advanced topics in Deep Probabilistic Modelling

Pablo Martínez Olmos, pamartin@ing.uc3m.es

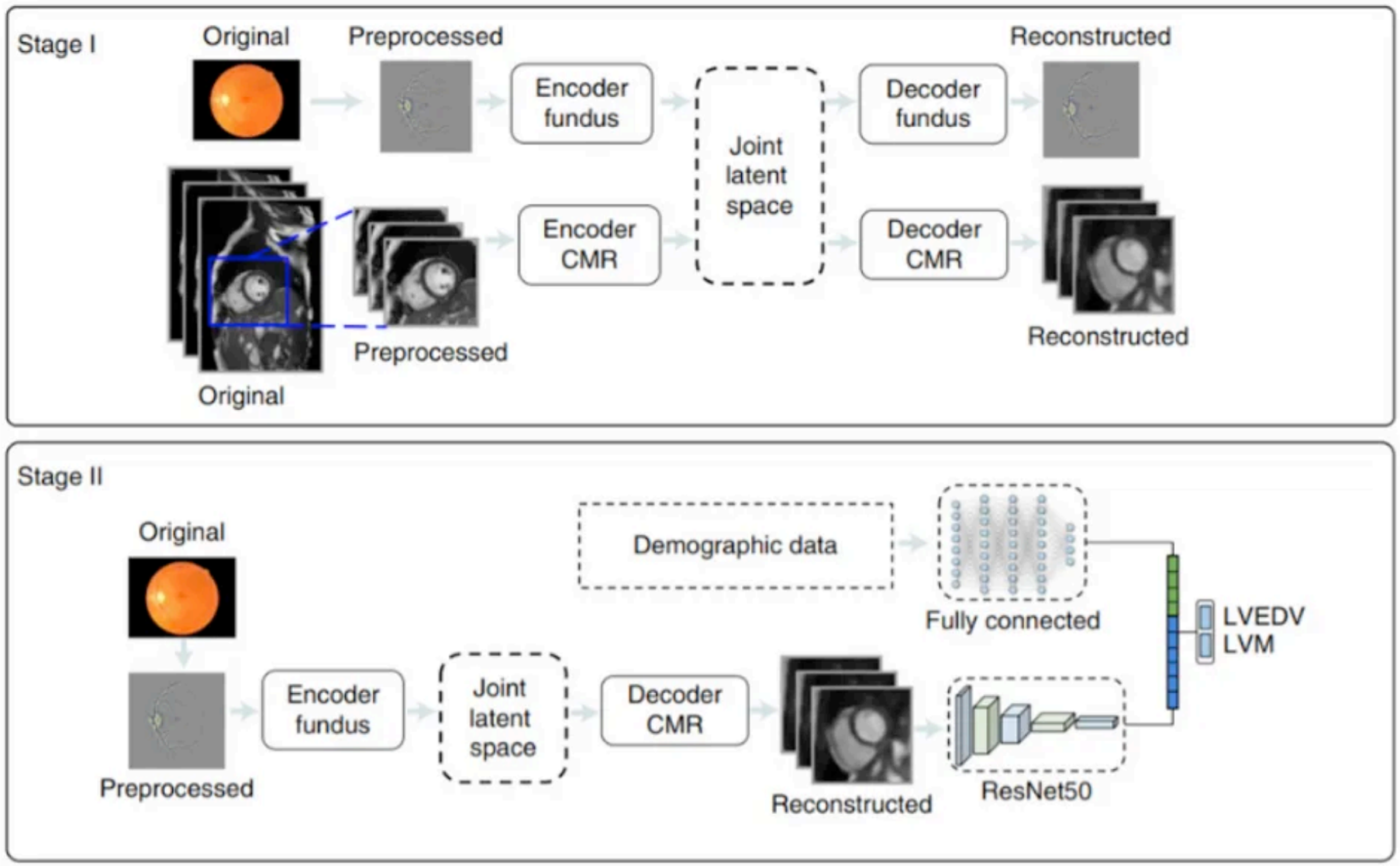
Multimodal generative models

Article | Published: 19 February 2018

Prediction of cardiovascular risk factors from retinal fundus photographs via deep learning

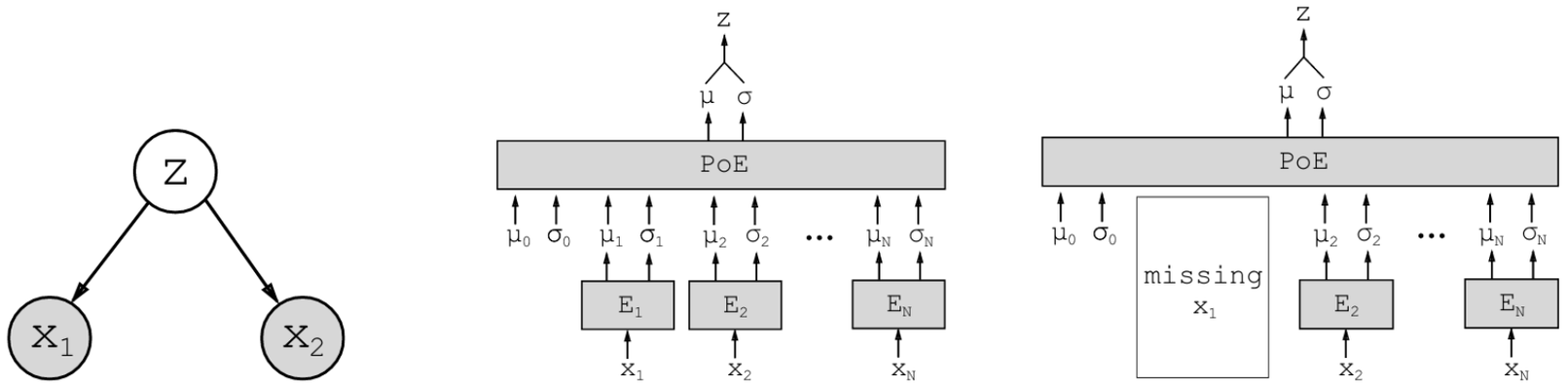
Ryan Poplin, Avinash V. Varadarajan, Katy Blumer, Yun Liu, Michael V. McConnell, Greg S. Corrado, Lily Peng ✉ & Dale R. Webster

Nature Biomedical Engineering **2**, 158–164 (2018) | Cite this article



uc3m

Multimodal Generative Models for Scalable Weakly-Supervised Learning



$$\text{ELBO}(X) \triangleq \mathbb{E}_{q_\phi(z|X)}[\sum_{x_i \in X} \lambda_i \log p_\theta(x_i|z)] - \beta\, \text{KL}[q_\phi(z|X), p(z)].$$

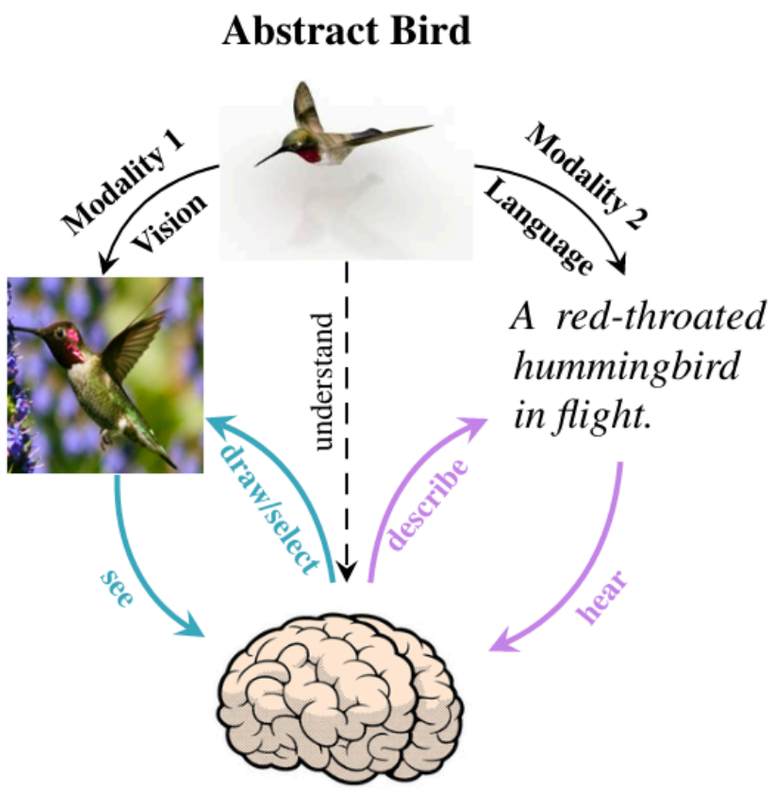
$$p(z|x_1, ..., x_N) \propto \; p(z) \prod_{i=1}^{N} \tilde{q}(z|x_i)$$

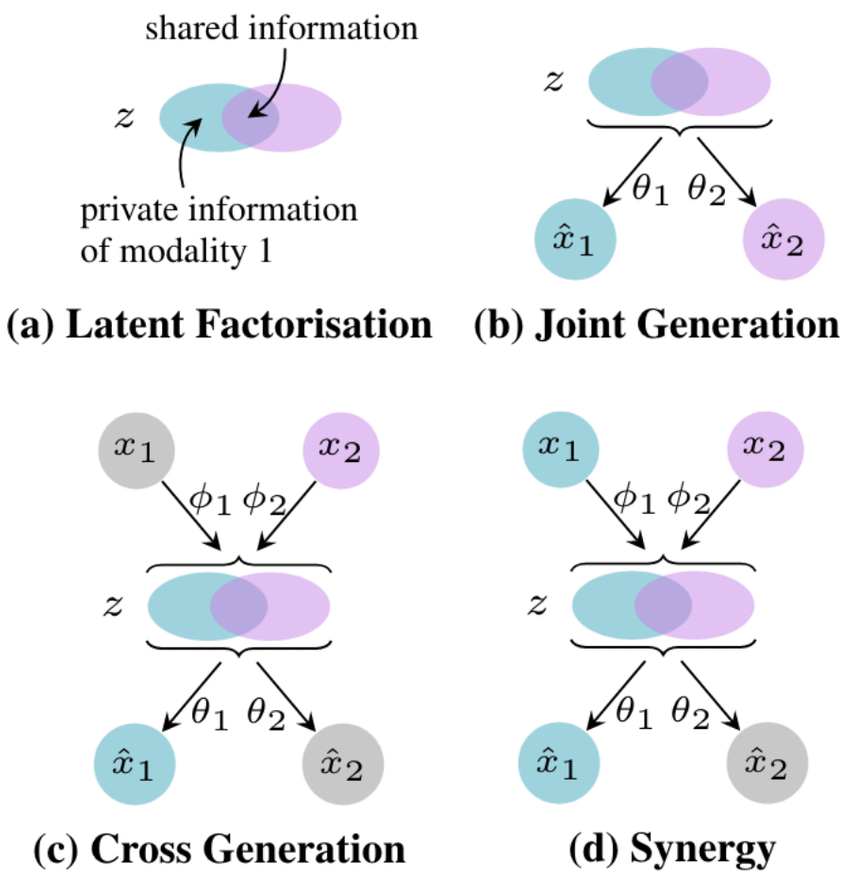
Product-of-experts (MoE)

uc3m

Variational Mixture-of-Experts Autoencoders for Multi-Modal Deep Generative Models

$$\mathcal{L}_{\text{ELBO}}(\boldsymbol{x}_{1:M}) = \mathbb{E}_{\boldsymbol{z}\sim q_\Phi(\boldsymbol{z}|\boldsymbol{x}_{1:M})}\left[\log \frac{p_\Theta(\boldsymbol{z}, \boldsymbol{x}_{1:M})}{q_\Phi(\boldsymbol{z} \mid \boldsymbol{x}_{1:M})}\right]$$





(a) Latent Factorisation (b) Joint Generation

(c) Cross Generation (d) Synergy

$$q_\Phi(\boldsymbol{z} \mid \boldsymbol{x}_{1:M}) = \sum_m \alpha_m \cdot q_{\phi_m}(\boldsymbol{z} \mid \boldsymbol{x}_m)$$

Mixture-of-experts (MoE)

uc3m

Generative classifiers as multimodal VAEs?

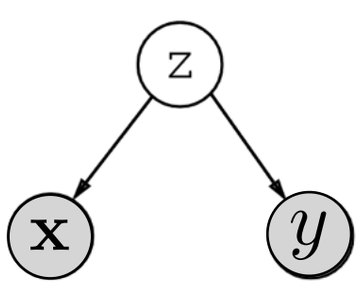
Deep Generative classifiers

$$(\mathbf{x}, y), \quad y \in \{0, 1\} \quad \mathbf{x} \in \mathbb{R}^D, D >> 1$$

Learn $p(\mathbf{x}, y)$ instead $p(y|\mathbf{x})$

- Semi-supervised learning
- Robust against outliers and adversarial attacks

Naive Approach:



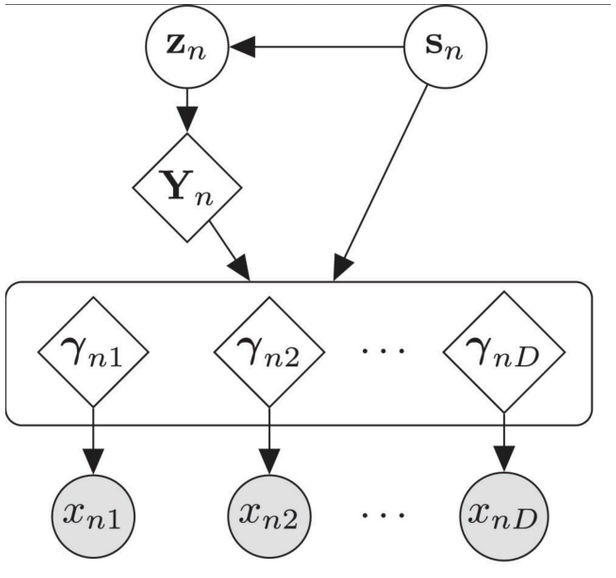
$$\mathbf{ELBO}(X) \triangleq \mathbb{E}_{q_\phi(z|X)}[\sum_{x_i \in X} \lambda_i \log p_\theta(x_i|z)] - \beta\, \mathbf{KL}[q_\phi(z|X), p(z)].$$

The reconstruction term for **x** dominates the ELBO!

- Normalization to balance reconstruction terms
- Change generative model!

uc3m

Deep Generative classifiers



ELSEVIER

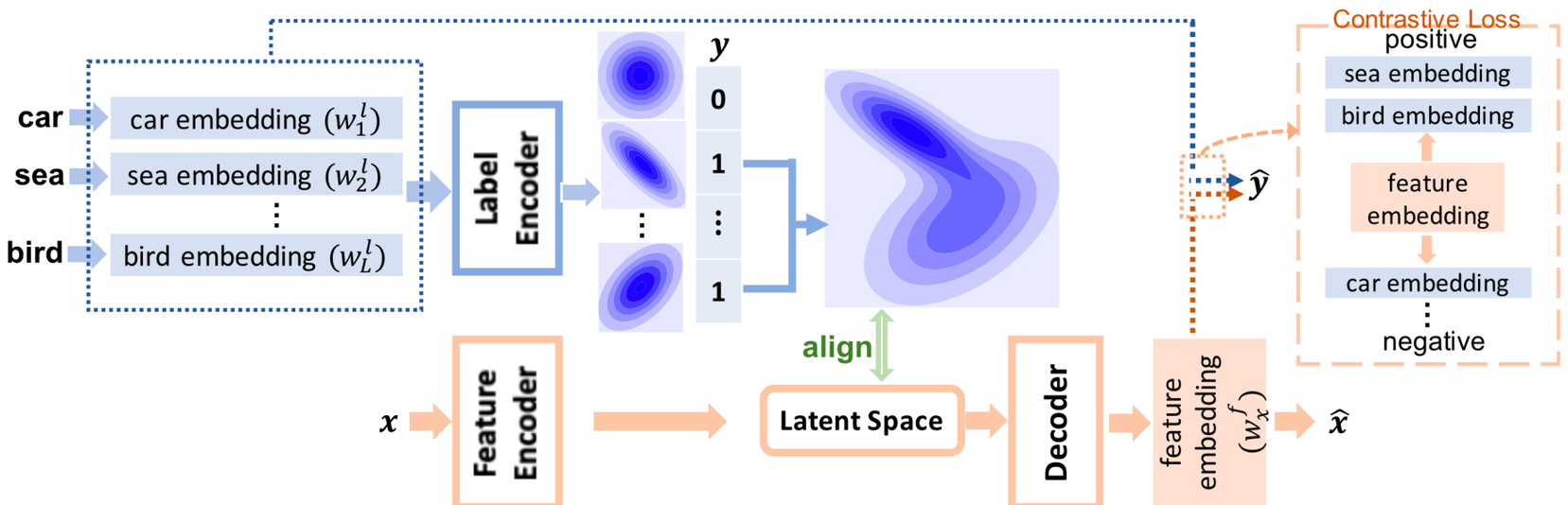
Pattern Recognition

Volume 107, November 2020, 107501

Handling incomplete heterogeneous data using VAEs

Gaussian Mixture Variational Autoencoder with Contrastive Learning for Multi-Label Classification

Junwen Bai [1] Shufeng Kong [1] Carla Gomes [1]



uc3m

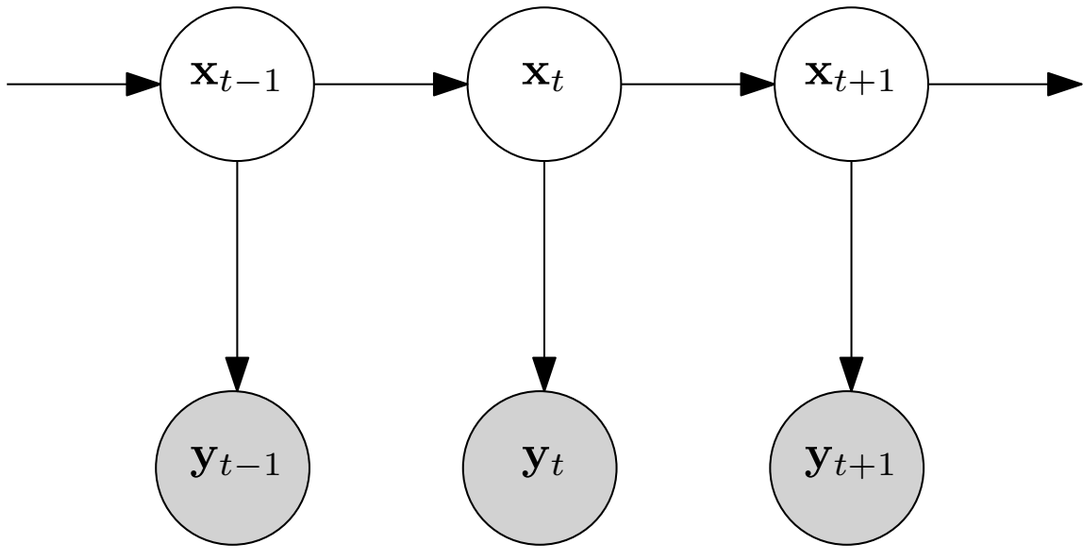
Unsupervised models for time series

uc3m

Hidden Markov Processes

If the observed sequence $\mathbf{y}_{1:T}$ is a noisy version of the (first order) Markov process $\mathbf{x}_{1:T}$

$$p(\mathbf{y}_{1:T}, \mathbf{x}_{1:T}) = p(\mathbf{y}_1|\mathbf{x}_1)p(\mathbf{x}_1) \dots p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{x}_{t-1}) \dots \\ \dots p(\mathbf{y}_T|\mathbf{x}_T)p(\mathbf{x}_T|\mathbf{x}_{T-1})$$



- Discrete x_t (s_t): Hidden Markov Model (HMM)
- Continuous $\mathbf{x}_t$: State Space Model (SSM)

Beyond the Kalman Filter: temporal variational autoencoders

$$p(\mathbf{x}_{\leq T}, \mathbf{z}_{\leq T}) = \prod_{t=1}^{T} p(\mathbf{x}_t \mid \mathbf{z}_{\leq t}, \mathbf{x}_{<t}) p(\mathbf{z}_t \mid \mathbf{x}_{<t}, \mathbf{z}_{<t}).$$

$$\mathbf{z}_t \sim \mathcal{N}(\boldsymbol{\mu}_{0,t}, \mathrm{diag}(\boldsymbol{\sigma}_{0,t}^2)) \text{ , where } [\boldsymbol{\mu}_{0,t}, \boldsymbol{\sigma}_{0,t}] = \varphi_\tau^{\text{prior}}(\mathbf{h}_{t-1}),$$

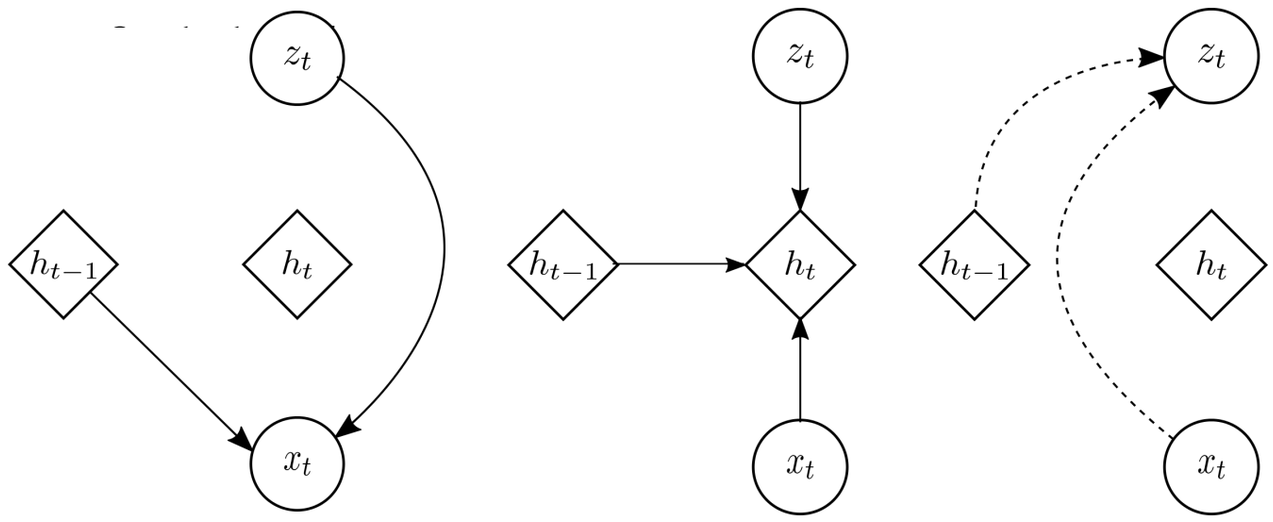
$$\mathbf{x}_t \mid \mathbf{z}_t \sim \mathcal{N}(\boldsymbol{\mu}_{x,t}, \mathrm{diag}(\boldsymbol{\sigma}_{x,t}^2)) \text{ , where } [\boldsymbol{\mu}_{x,t}, \boldsymbol{\sigma}_{x,t}] = \varphi_\tau^{\text{dec}}(\varphi_\tau^{\mathbf{z}}(\mathbf{z}_t), \mathbf{h}_{t-1}),$$

The RNN updates its hidden state using the recurrence equation:

$$\mathbf{h}_t = f_\theta\left(\varphi_\tau^{\mathbf{x}}(\mathbf{x}_t), \varphi_\tau^{\mathbf{z}}(\mathbf{z}_t), \mathbf{h}_{t-1}\right),$$

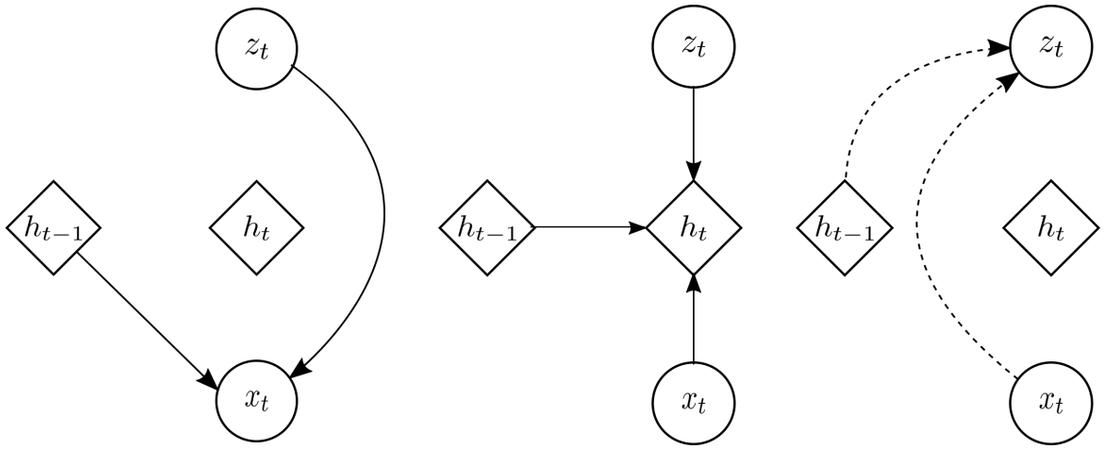
A Recurrent Latent Variable Model for Sequential Data

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron Courville, Yoshua Bengio*
Department of Computer Science and Operations Research
Université de Montréal
*CIFAR Senior Fellow
{firstname.lastname}@umontreal.ca



(b) Generation (c) Recurrence (d) Inference

Beyond the Kalman Filter: temporal variational autoencoders



(b) Generation (c) Recurrence (d) Inference

A Recurrent Latent Variable Model for Sequential Data

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron Courville, Yoshua Bengio*
Department of Computer Science and Operations Research
Université de Montréal
*CIFAR Senior Fellow
{firstname.lastname}@umontreal.ca

$\mathbf{z}_t \mid \mathbf{x}_t \sim \mathcal{N}(\boldsymbol{\mu}_{z,t}, \text{diag}(\boldsymbol{\sigma}_{z,t}^2))$, where $[\boldsymbol{\mu}_{z,t}, \boldsymbol{\sigma}_{z,t}] = \varphi_\tau^{\text{enc}}(\varphi_\tau^{\mathbf{x}}(\mathbf{x}_t), \mathbf{h}_{t-1})$

$$q(\mathbf{z}_{\leq T} \mid \mathbf{x}_{\leq T}) = \prod_{t=1}^{T} q(\mathbf{z}_t \mid \mathbf{x}_{\leq t}, \mathbf{z}_{<t}).$$

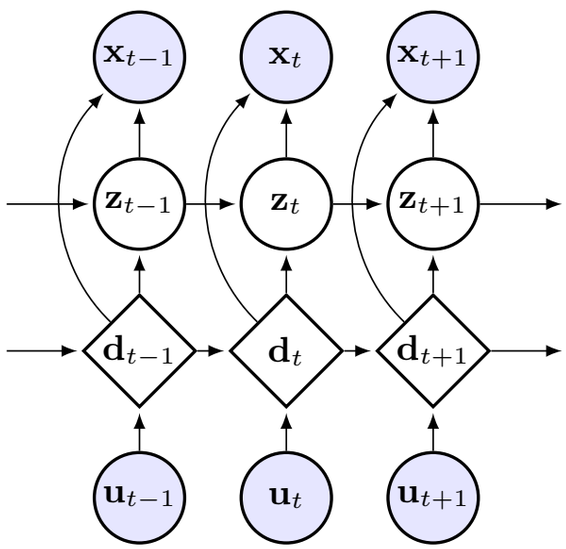
Learning The objective function becomes a timestep-wise variational lower bound using Eq. (8) and Eq. (10):

$$\mathbb{E}_{q(\mathbf{z}_{\leq T}|\mathbf{x}_{\leq T})}\left[\sum_{t=1}^{T}(-\text{KL}(q(\mathbf{z}_t \mid \mathbf{x}_{\leq t}, \mathbf{z}_{<t})\|p(\mathbf{z}_t \mid \mathbf{x}_{<t}, \mathbf{z}_{<t})) + \log p(\mathbf{x}_t \mid \mathbf{z}_{\leq t}, \mathbf{x}_{<t}))\right]. \quad (11)$$

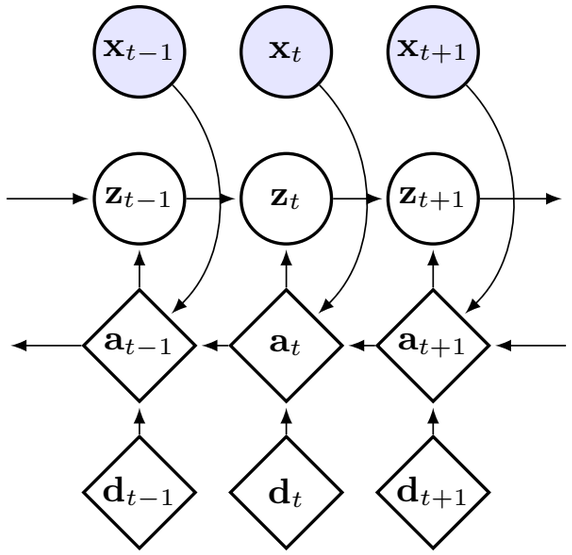
Beyond the Kalman Filter: temporal variational autoencoders

Sequential Neural Models with Stochastic Layers

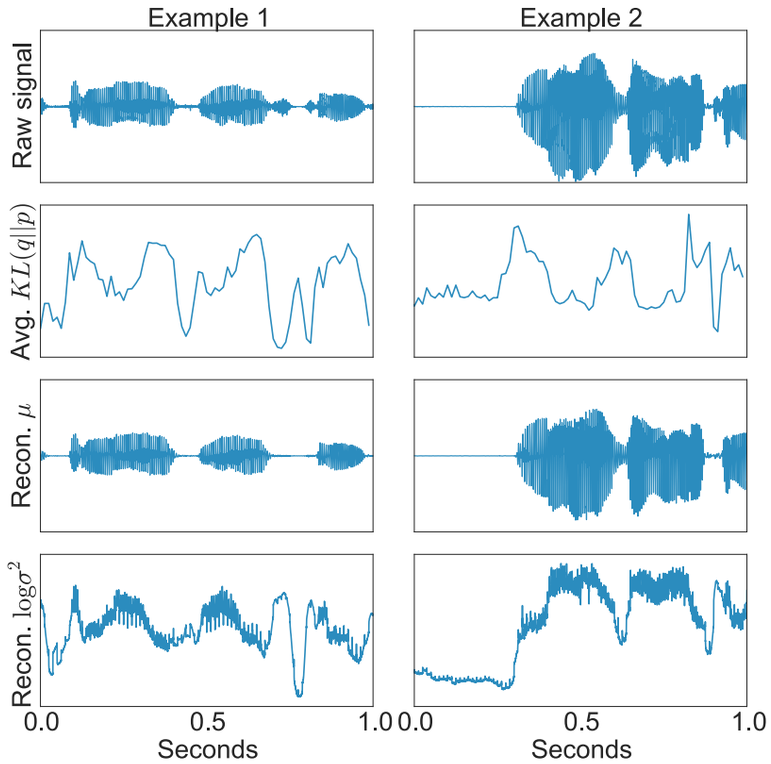
Marco Fraccaro[†] **Søren Kaae Sønderby**[‡] **Ulrich Paquet**[*] **Ole Winther**[†‡]
[†] Technical University of Denmark
[‡] University of Copenhagen
[*] Google DeepMind



(a) Generative model p_θ



(b) Inference network q_ϕ

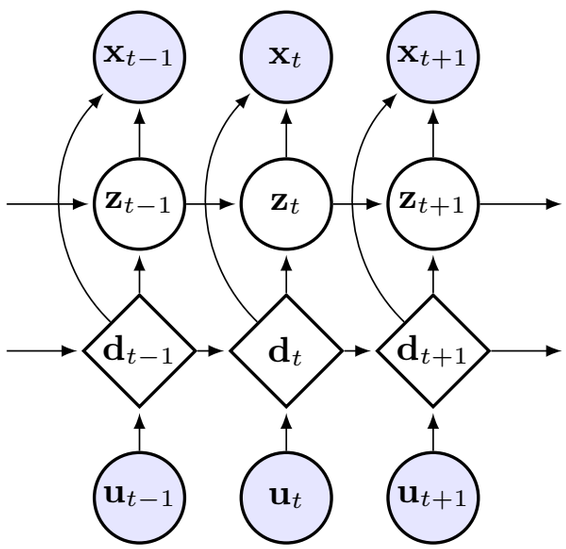


$$\mathcal{F}_i(\theta,\phi) = \mathbb{E}_{q_\phi}\left[\log p_\theta(\mathbf{x}_{1:T}|\mathbf{z}_{1:T},\widetilde{\mathbf{d}}_{1:T})\right] - \mathrm{KL}\left(q_\phi(\mathbf{z}_{1:T}|\widetilde{\mathbf{d}}_{1:T},\mathbf{x}_{1:T},\mathbf{z}_0)\,\middle\|\,p_\theta(\mathbf{z}_{1:T}|\widetilde{\mathbf{d}}_{1:T},\mathbf{z}_0)\right)$$

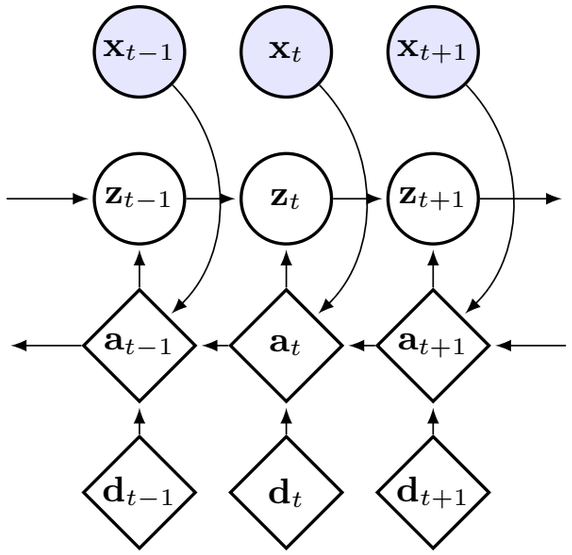
Beyond the Kalman Filter: temporal variational autoencoders

Sequential Neural Models with Stochastic Layers

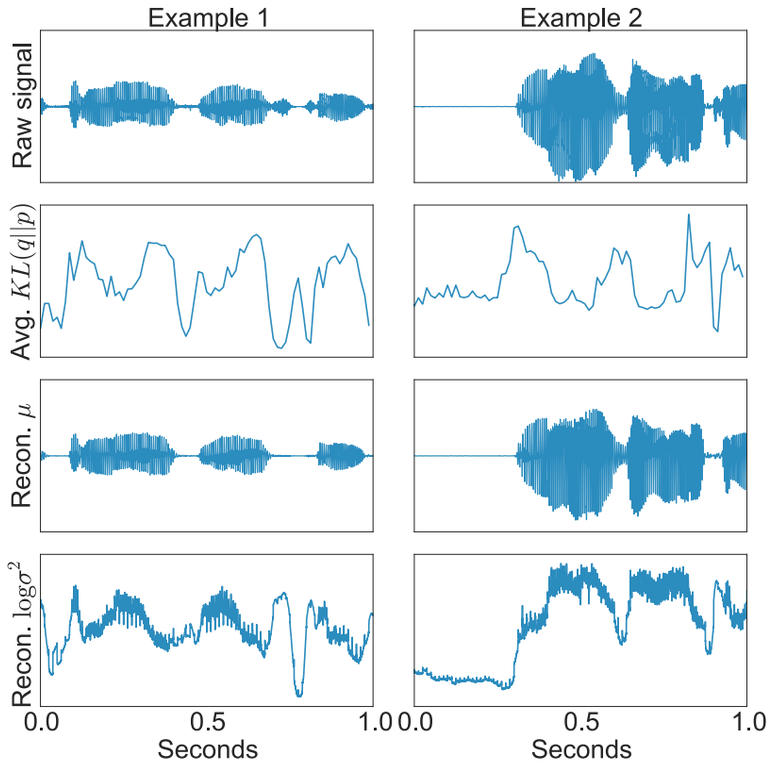
Marco Fraccaro[†] **Søren Kaae Sønderby**[‡] **Ulrich Paquet**[*] **Ole Winther**[†‡]
[†] Technical University of Denmark
[‡] University of Copenhagen
[*] Google DeepMind



(a) Generative model p_θ



(b) Inference network q_ϕ



$$\mathcal{F}_i(\theta,\phi) = \mathbb{E}_{q_\phi}\left[\log p_\theta(\mathbf{x}_{1:T}|\mathbf{z}_{1:T},\widetilde{\mathbf{d}}_{1:T})\right] - \mathrm{KL}\left(q_\phi(\mathbf{z}_{1:T}|\widetilde{\mathbf{d}}_{1:T},\mathbf{x}_{1:T},\mathbf{z}_0)\,\middle\|\,p_\theta(\mathbf{z}_{1:T}|\widetilde{\mathbf{d}}_{1:T},\mathbf{z}_0)\right)$$

Beyond the Kalman Filter: temporal variational autoencoders

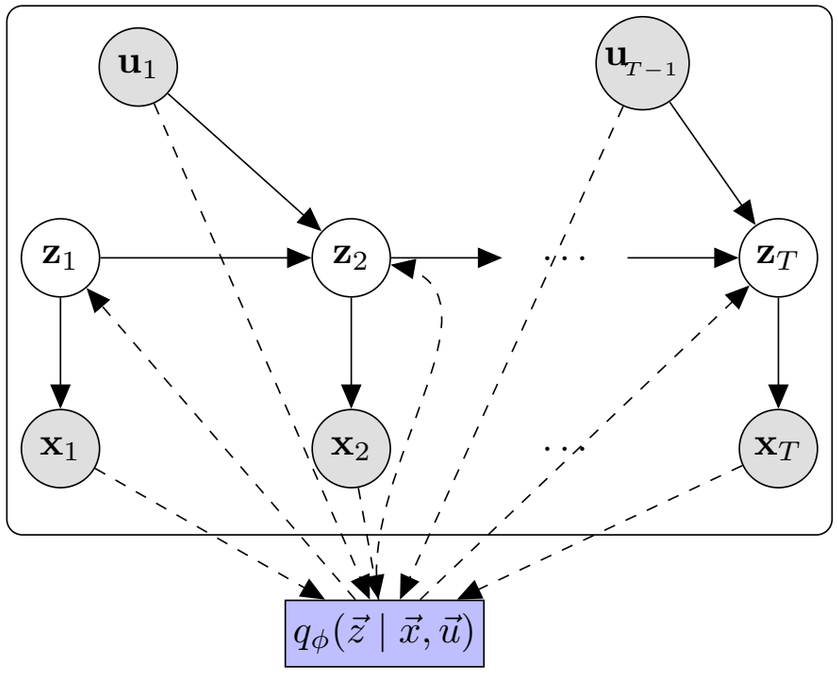
Deep Kalman Filters

Rahul G. Krishnan **Uri Shalit** **David Sontag**
Courant Institute of Mathematical Sciences
New York University

$$z_t = G_t z_{t-1} + B_t u_{t-1} + \epsilon_t \ \textit{(action-transition)} \ , \qquad x_t = F_t z_t + \eta_t \ \textit{(observation)},$$

$$z_1 \sim \mathcal{N}(\mu_0; \Sigma_0)$$
$$z_t \sim \mathcal{N}(G_\alpha(z_{t-1}, u_{t-1}, \Delta_t), S_\beta(z_{t-1}, u_{t-1}, \Delta_t))$$
$$x_t \sim \Pi(F_\kappa(z_t)).$$

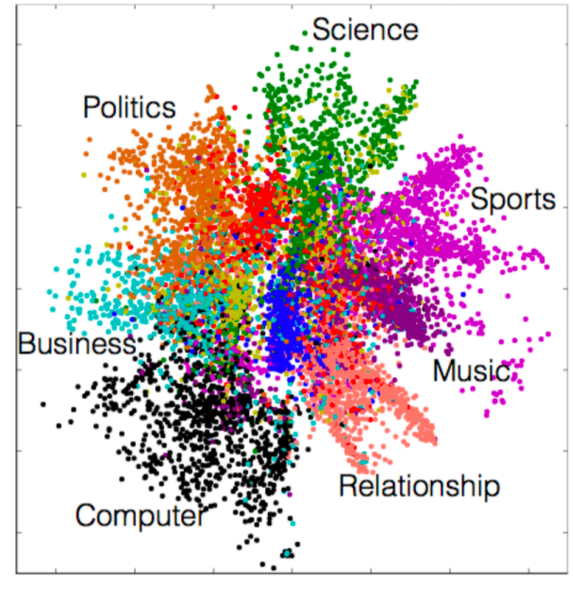
$$\log p_\theta(\vec{x}|\vec{u}) \geq \mathcal{L}(x; (\theta, \phi)) =$$
$$\sum_{t=1}^{T} \mathop{\mathbb{E}}_{q_\phi(z_t|\vec{x},\vec{u})} [\log p_\theta(x_t|z_t)] - \mathrm{KL}(q_\phi(z_1|\vec{x},\vec{u})||p_0(z_1))$$
$$- \sum_{t=2}^{T} \mathop{\mathbb{E}}_{q_\phi(z_{t-1}|\vec{x},\vec{u})} [\mathrm{KL}(q_\phi(z_t|z_{t-1},\vec{x},\vec{u})||p_0(z_t|z_{t-1},u_{t-1}))] \, .$$



Hybrid autoregressive and latent variable models

A Hierarchical Latent Variable Encoder-Decoder Model for Generating Dialogues

Iulian V. Serban*, Alessandro Sordoni ‡, Ryan Lowe◇, Laurent Charlin◇, Joelle Pineau◇,
Aaron Courville* and Yoshua Bengio*



$$P_\theta(w_1, \ldots, w_M) = \prod_{m=2}^{M} P_\theta(w_m \mid w_1, \ldots, w_{m-1}) P(w_1).$$

$$P_\theta(\mathbf{z}_n \mid \mathbf{w}_1, \ldots, \mathbf{w}_{n-1}) = \mathcal{N}(\boldsymbol{\mu}_{\text{prior}}(\mathbf{w}_1, \ldots, \mathbf{w}_{n-1}), \Sigma_{\text{prior}}(\mathbf{w}_1, \ldots, \mathbf{w}_{n-1})),$$

$$P_\theta(\mathbf{w}_n \mid \mathbf{z}_n, \mathbf{w}_1, \ldots, \mathbf{w}_{n-1}) = \prod_{m=1}^{M_n} P_\theta(w_{n,m} \mid \mathbf{z}_n, \mathbf{w}_1, \ldots, \mathbf{w}_{n-1}, w_{n,1}, \ldots, w_{n,m-1}),$$

$$\begin{aligned} Q_\psi(\mathbf{z}_n \mid \mathbf{w}_1, \ldots, \mathbf{w}_N) = Q_\psi(\mathbf{z}_n \mid \mathbf{w}_1, \ldots, \mathbf{w}_n) &= \mathcal{N}(\boldsymbol{\mu}_{\text{posterior}}(\mathbf{w}_1, \ldots, \mathbf{w}_n), \Sigma_{\text{posterior}}(\mathbf{w}_1, \ldots, \mathbf{w}_n)) \\ &\approx P_\psi(\mathbf{z}_n \mid \mathbf{w}_1, \ldots, \mathbf{w}_N), \end{aligned} \tag{5}$$

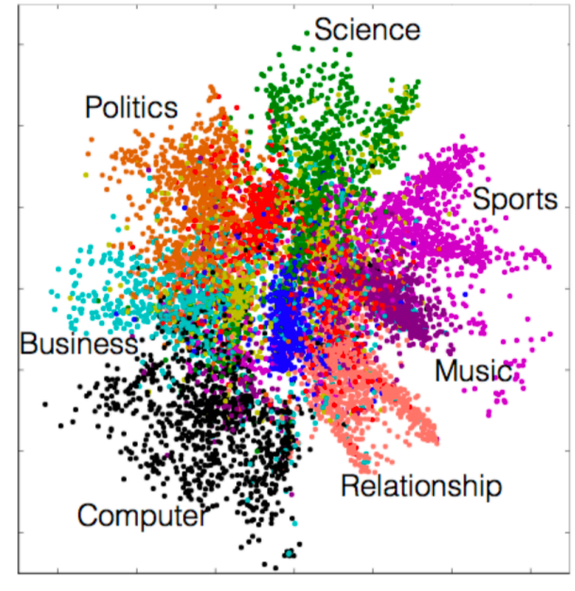
$$\begin{aligned} \log P_\theta(\mathbf{w}_1, \ldots, \mathbf{w}_N) \geq \sum_{n=1}^{N} & -\text{KL}\left[Q_\psi(\mathbf{z}_n \mid \mathbf{w}_1, \ldots, \mathbf{w}_n) || P_\theta(\mathbf{z}_n \mid \mathbf{w}_1, \ldots, \mathbf{w}_{n-1})\right] \\ & + \mathbb{E}_{Q_\psi(\mathbf{z}_n | \mathbf{w}_1, \ldots, \mathbf{w}_n)}\left[\log P_\theta(\mathbf{w}_n \mid \mathbf{z}_n, \mathbf{w}_1, \ldots, \mathbf{w}_{n-1})\right], \end{aligned}$$

Hybrid autoregressive and latent variable models

A Hierarchical Latent Variable Encoder-Decoder Model for Generating Dialogues

Iulian V. Serban*, Alessandro Sordoni ‡, Ryan Lowe◇, Laurent Charlin◇, Joelle Pineau◇,
Aaron Courville* and Yoshua Bengio*

$$P_\theta(w_1, \ldots, w_M) = \prod_{m=2}^{M} P_\theta(w_m \mid w_1, \ldots, w_{m-1}) P(w_1).$$



Latent variable adds topic coherence and improved long-term memory

$$P_\theta(\mathbf{z}_n \mid \mathbf{w}_1, \ldots, \mathbf{w}_{n-1}) = \mathcal{N}(\boldsymbol{\mu}_{\text{prior}}(\mathbf{w}_1, \ldots, \mathbf{w}_{n-1}), \Sigma_{\text{prior}}(\mathbf{w}_1, \ldots, \mathbf{w}_{n-1})),$$

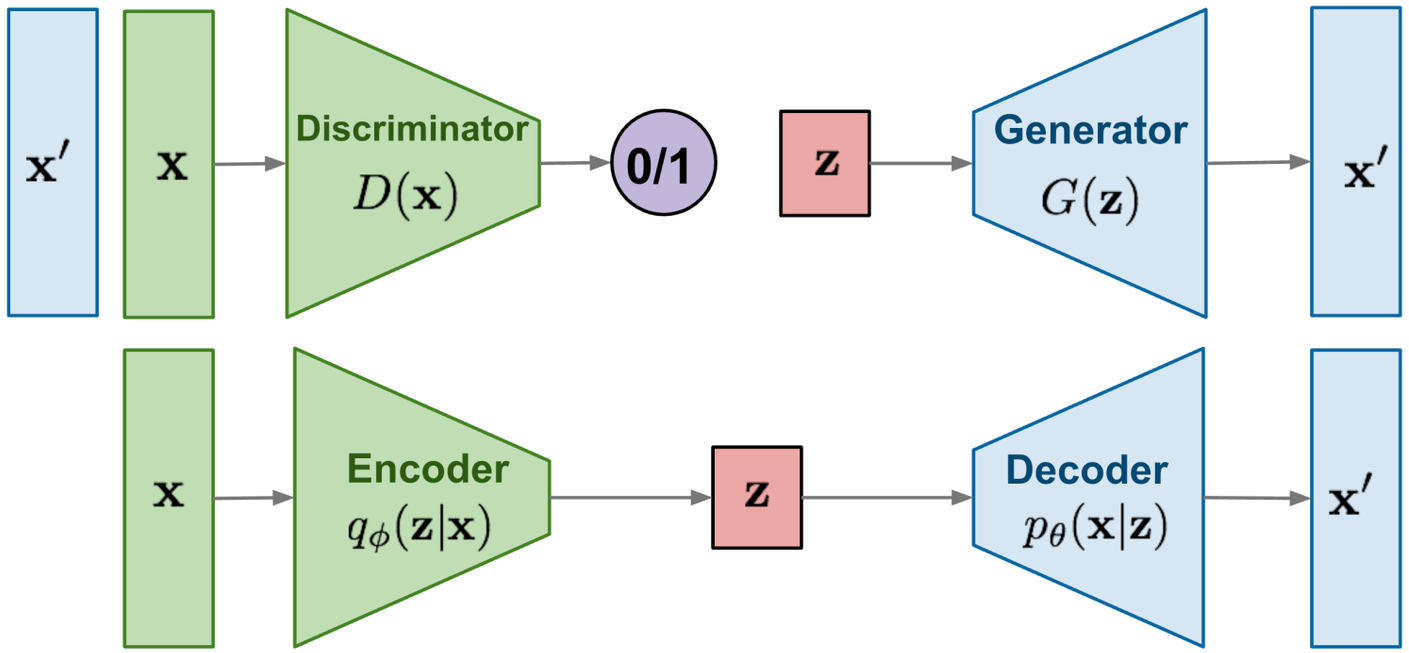
$$P_\theta(\mathbf{w}_n \mid \mathbf{z}_n, \mathbf{w}_1, \ldots, \mathbf{w}_{n-1}) = \prod_{m=1}^{M_n} P_\theta(w_{n,m} \mid \mathbf{z}_n, \mathbf{w}_1, \ldots, \mathbf{w}_{n-1}, w_{n,1}, \ldots, w_{n,m-1}),$$

$$\begin{aligned} Q_\psi(\mathbf{z}_n \mid \mathbf{w}_1, \ldots, \mathbf{w}_N) = Q_\psi(\mathbf{z}_n \mid \mathbf{w}_1, \ldots, \mathbf{w}_n) &= \mathcal{N}(\boldsymbol{\mu}_{\text{posterior}}(\mathbf{w}_1, \ldots, \mathbf{w}_n), \Sigma_{\text{posterior}}(\mathbf{w}_1, \ldots, \mathbf{w}_n)) \\ &\approx P_\psi(\mathbf{z}_n \mid \mathbf{w}_1, \ldots, \mathbf{w}_N), \end{aligned} \tag{5}$$

$$\begin{aligned} \log P_\theta(\mathbf{w}_1, \ldots, \mathbf{w}_N) \geq \sum_{n=1}^{N} & -\text{KL}\left[Q_\psi(\mathbf{z}_n \mid \mathbf{w}_1, \ldots, \mathbf{w}_n) || P_\theta(\mathbf{z}_n \mid \mathbf{w}_1, \ldots, \mathbf{w}_{n-1})\right] \\ & + \mathbb{E}_{Q_\psi(\mathbf{z}_n \mid \mathbf{w}_1, \ldots, \mathbf{w}_n)}\left[\log P_\theta(\mathbf{w}_n \mid \mathbf{z}_n, \mathbf{w}_1, \ldots, \mathbf{w}_{n-1})\right], \end{aligned}$$

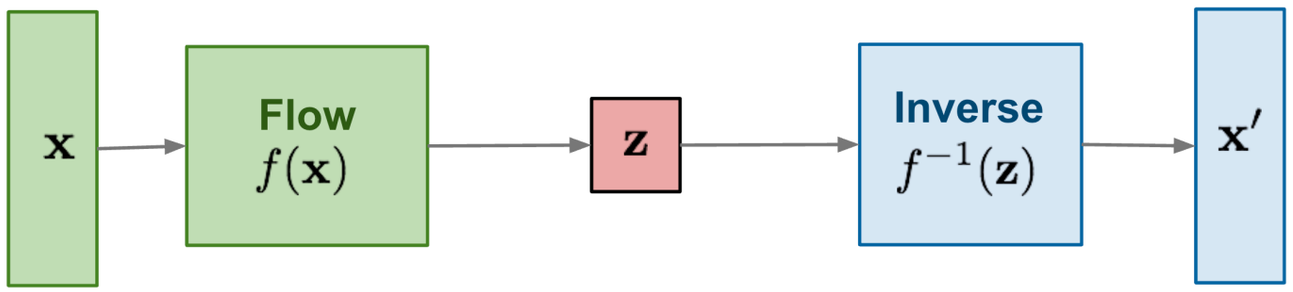
Normalizing Flows

GAN: Adversarial training

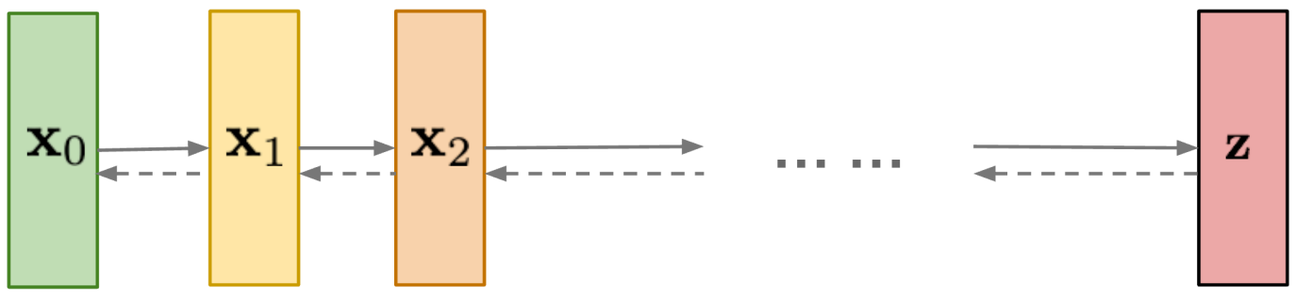


VAE: maximize variational lower bound

Flow-based models: Invertible transform of distributions



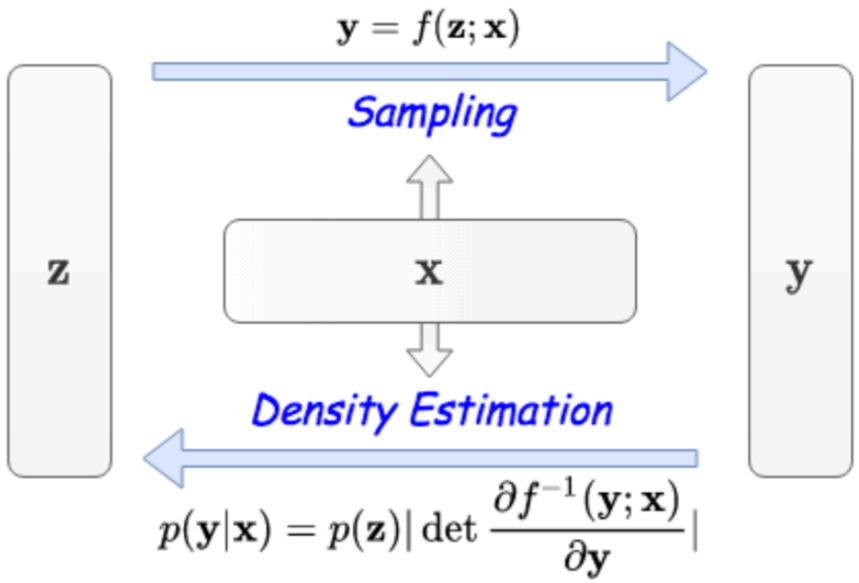
Diffusion models: Gradually add Gaussian noise and then reverse



uc3m

Normalizing Flows Models

- Invertible Functions with Neural Networks.
- Same input-output dimension!
- No dimensionality reduction



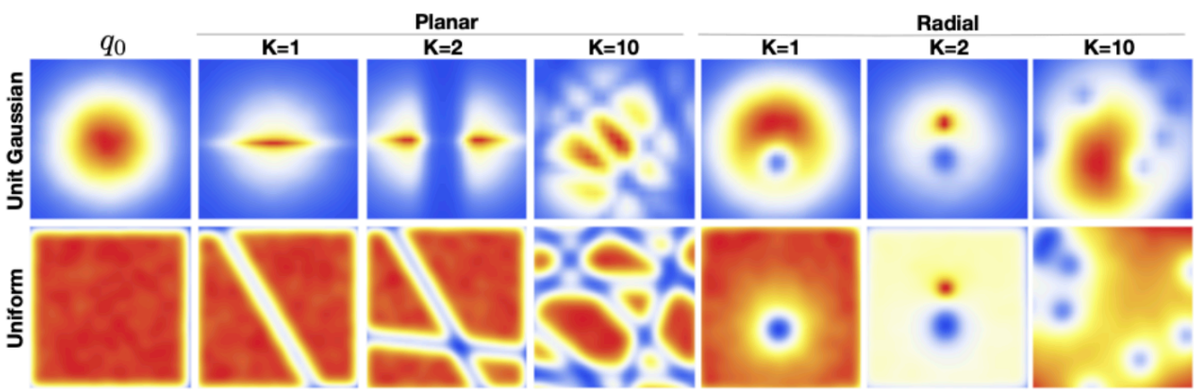
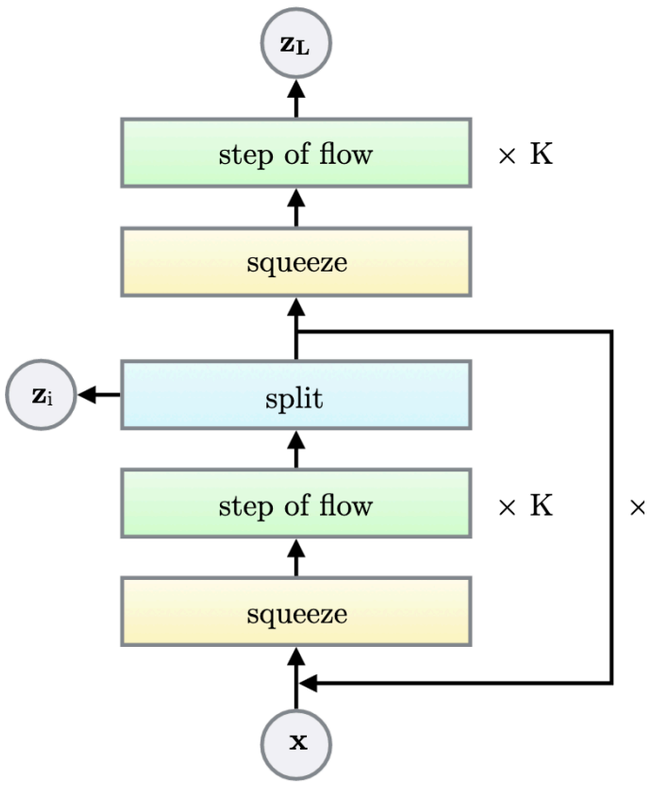


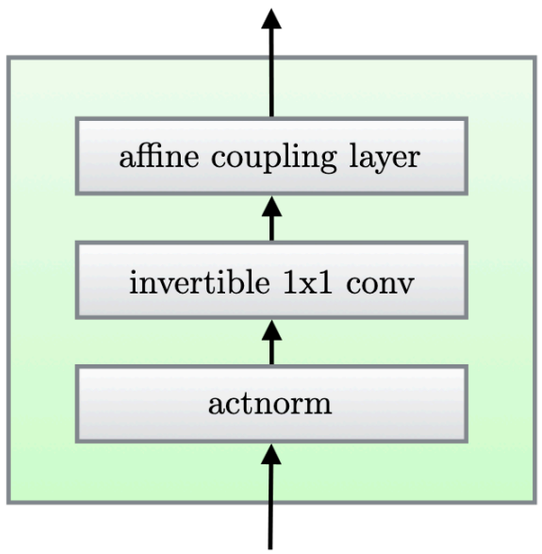
Figure 1. Effect of normalizing flow on two distributions.

Glow: Generative Flow with Invertible 1×1 Convolutions

Figure 1: Synthetic celebrities sampled from our model; see Section 3 for architecture and method, and Section 5 for more results.

Normalizing Flows Models





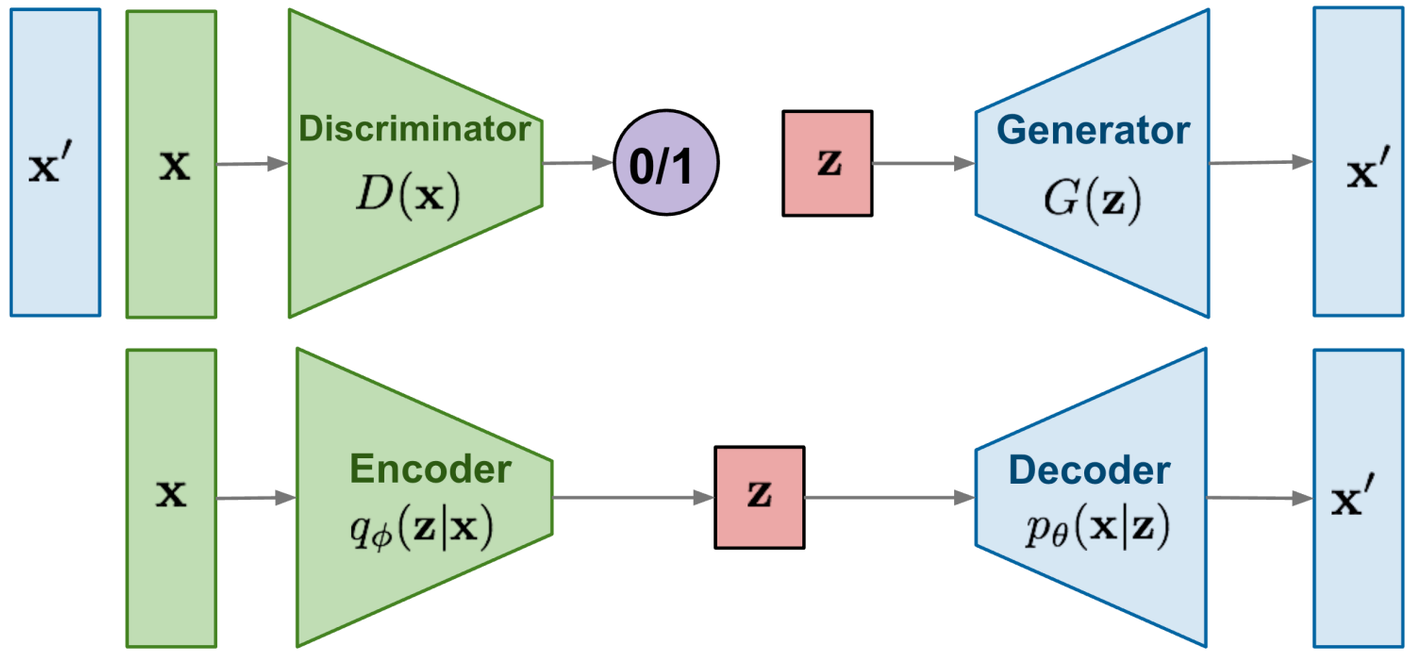
$$\mathbf{x} \overset{\mathbf{f}_1}{\longleftrightarrow} \mathbf{h}_1 \overset{\mathbf{f}_2}{\longleftrightarrow} \mathbf{h}_2 \cdots \overset{\mathbf{f}_K}{\longleftrightarrow} \mathbf{z}$$

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

Description	Function	Reverse Function	Log-determinant
Actnorm. See Section 3.1.	$\forall i,j : \mathbf{y}_{i,j} = \mathbf{s} \odot \mathbf{x}_{i,j} + \mathbf{b}$	$\forall i,j : \mathbf{x}_{i,j} = (\mathbf{y}_{i,j} - \mathbf{b})/\mathbf{s}$	$h \cdot w \cdot \mathtt{sum}(\log \lvert\mathbf{s}\rvert)$
Invertible 1×1 convolution. $\mathbf{W} : [c \times c]$. See Section 3.2.	$\forall i,j : \mathbf{y}_{i,j} = \mathbf{W}\mathbf{x}_{i,j}$	$\forall i,j : \mathbf{x}_{i,j} = \mathbf{W}^{-1}\mathbf{y}_{i,j}$	$h \cdot w \cdot \log \lvert \det(\mathbf{W}) \rvert$ or $h \cdot w \cdot \mathtt{sum}(\log \lvert\mathbf{s}\rvert)$ (see eq. (10))
Affine coupling layer. See Section 3.3 and (Dinh et al., 2014)	$\mathbf{x}_a, \mathbf{x}_b = \mathtt{split}(\mathbf{x})$ $(\log \mathbf{s}, \mathbf{t}) = \mathtt{NN}(\mathbf{x}_b)$ $\mathbf{s} = \exp(\log \mathbf{s})$ $\mathbf{y}_a = \mathbf{s} \odot \mathbf{x}_a + \mathbf{t}$ $\mathbf{y}_b = \mathbf{x}_b$ $\mathbf{y} = \mathtt{concat}(\mathbf{y}_a, \mathbf{y}_b)$	$\mathbf{y}_a, \mathbf{y}_b = \mathtt{split}(\mathbf{y})$ $(\log \mathbf{s}, \mathbf{t}) = \mathtt{NN}(\mathbf{y}_b)$ $\mathbf{s} = \exp(\log \mathbf{s})$ $\mathbf{x}_a = (\mathbf{y}_a - \mathbf{t})/\mathbf{s}$ $\mathbf{x}_b = \mathbf{y}_b$ $\mathbf{x} = \mathtt{concat}(\mathbf{x}_a, \mathbf{x}_b)$	$\mathtt{sum}(\log(\lvert\mathbf{s}\rvert))$

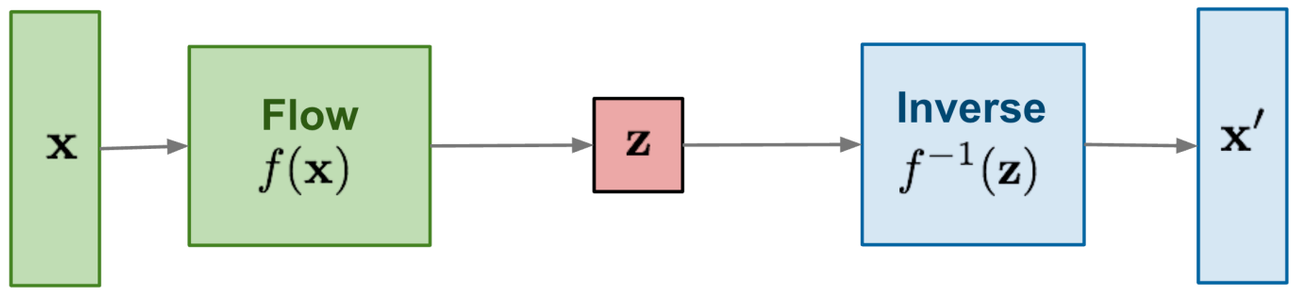
Diffusion Models

GAN: Adversarial training

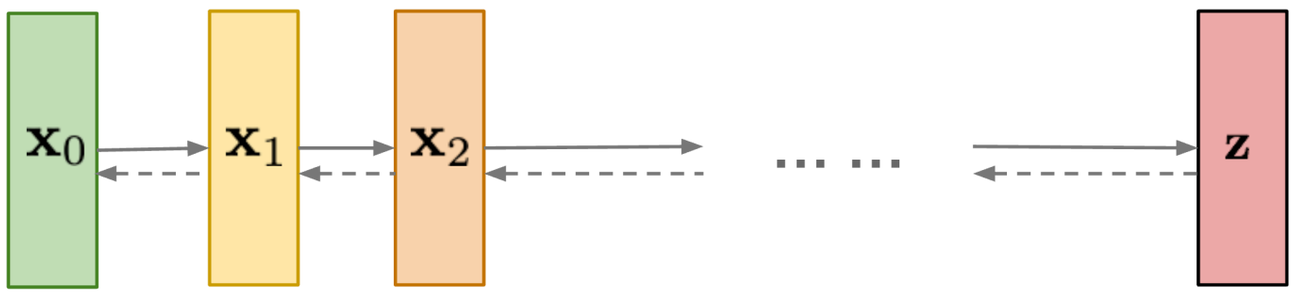


VAE: maximize variational lower bound

Flow-based models: Invertible transform of distributions



Diffusion models: Gradually add Gaussian noise and then reverse



uc3m

Diffusion models

Don’t miss this blog!

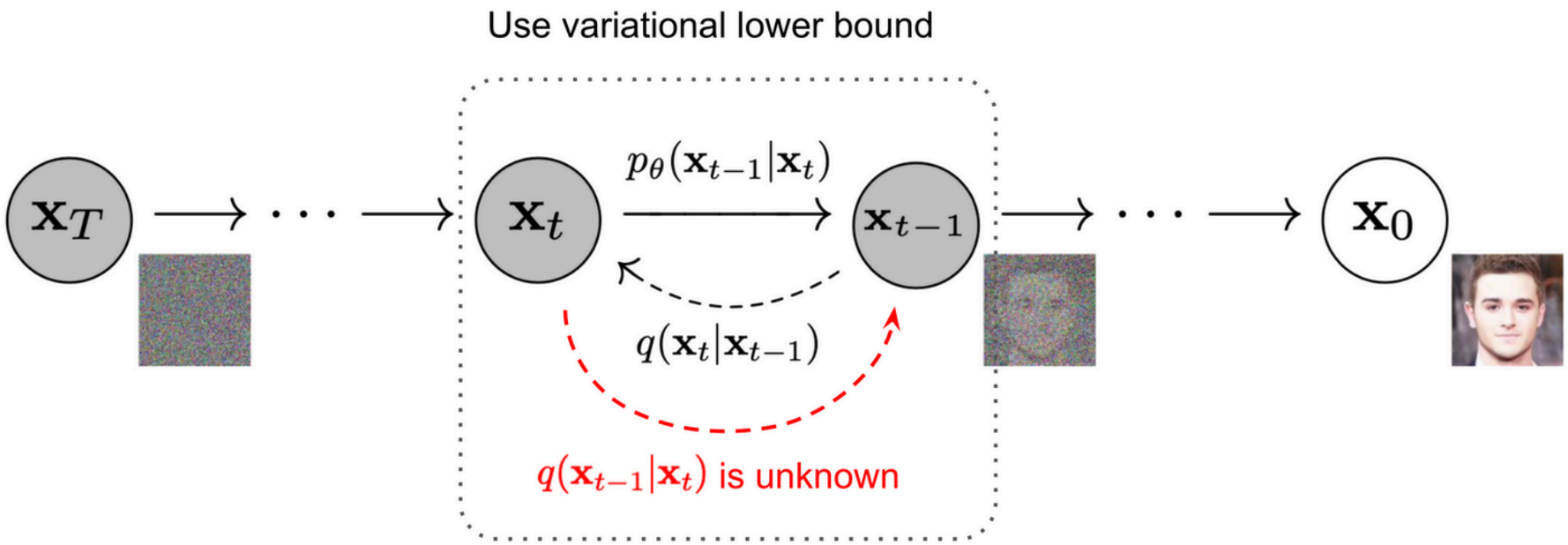
Denoising Diffusion Probabilistic Models

Jonathan Ho
UC Berkeley
jonathanho@berkeley.edu

Ajay Jain
UC Berkeley
ajayj@berkeley.edu

Pieter Abbeel
UC Berkeley
pabbeel@cs.berkeley.edu

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \quad q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$$



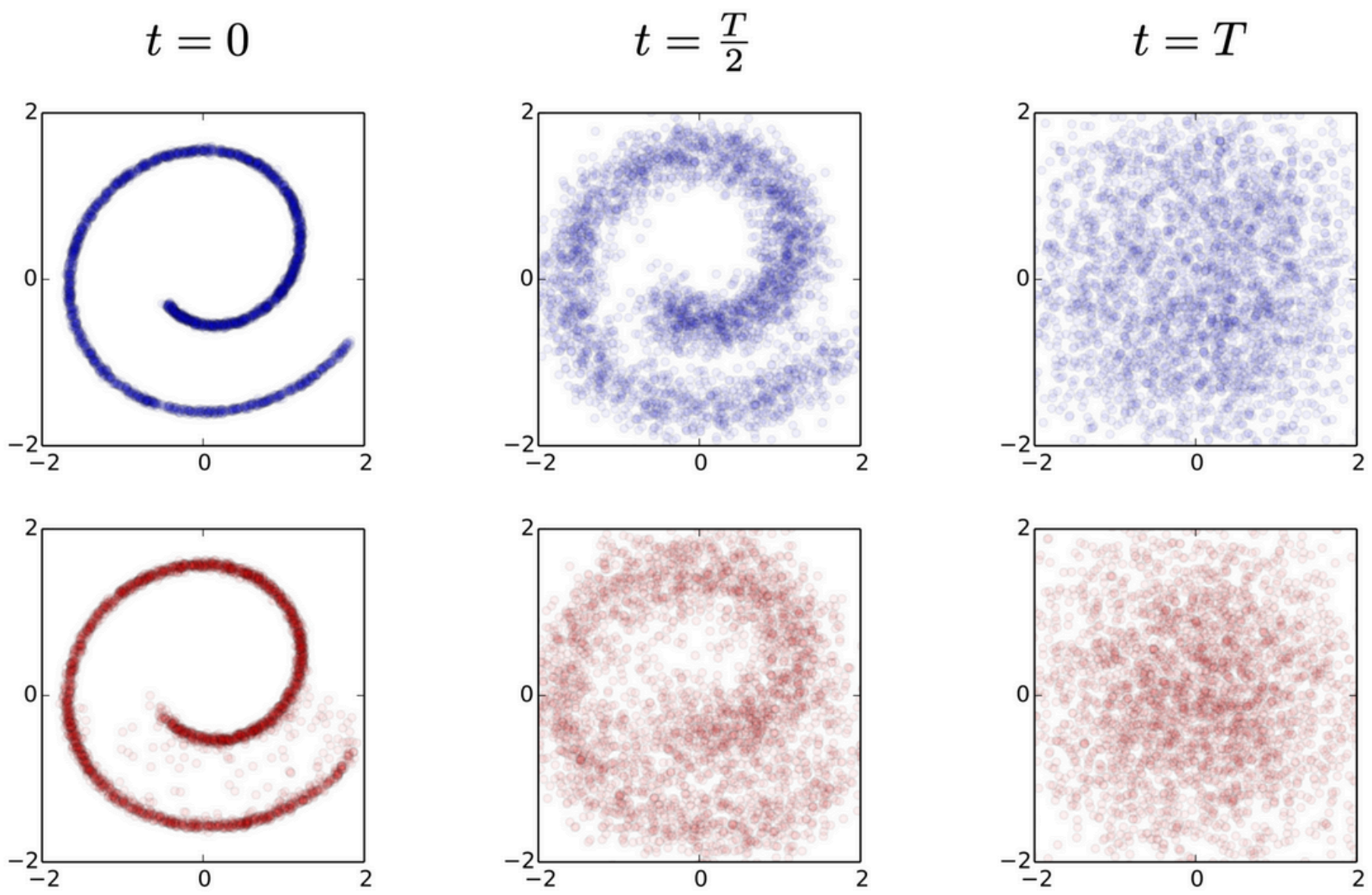
$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \quad p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

The forward trajectory

$q(\mathbf{x}_{0:T})$

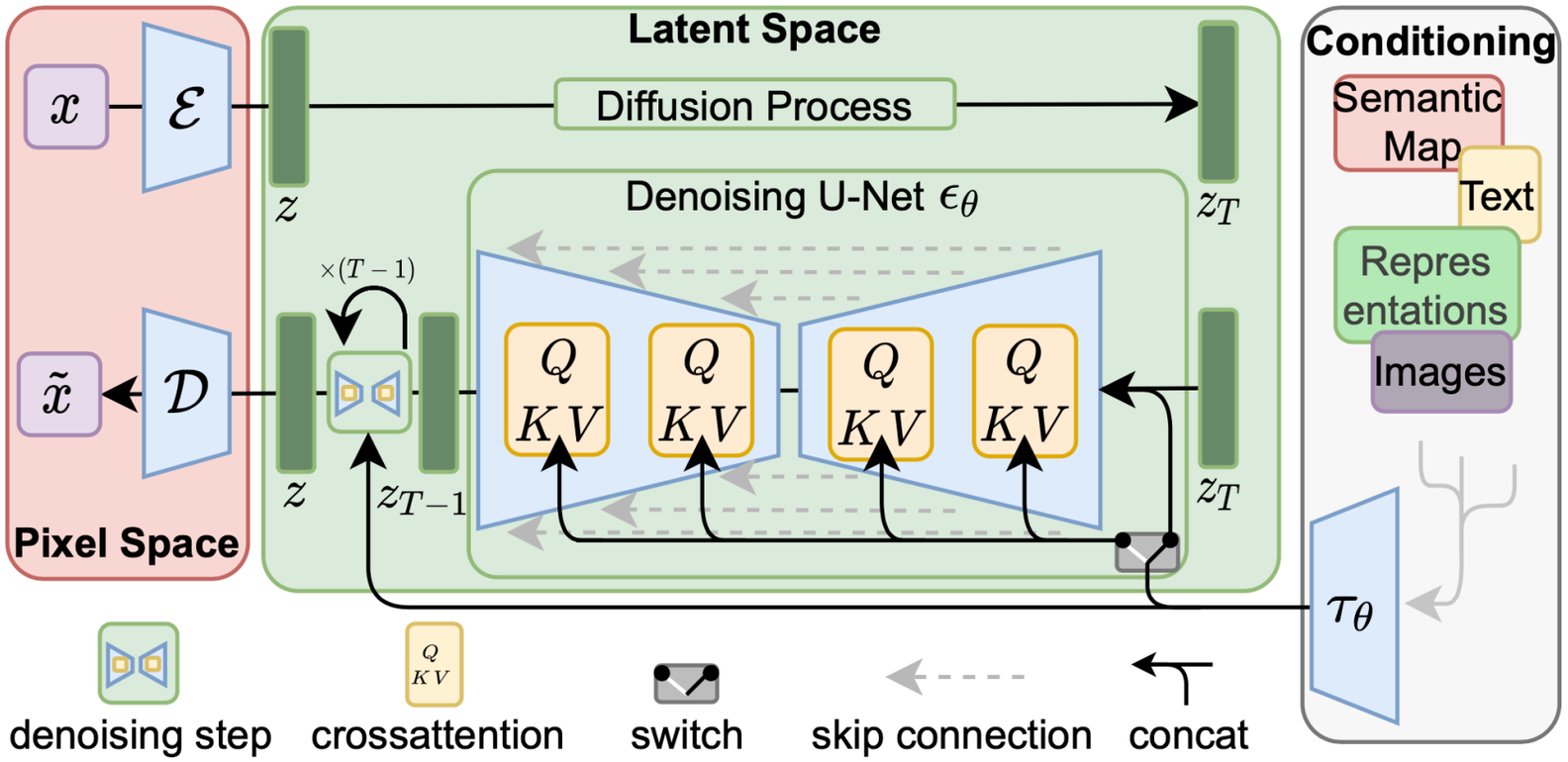
The reverse trajectory

$p_\theta(\mathbf{x}_{0:T})$



$$-\log p_\theta(\mathbf{x}_0) \leq -\log p_\theta(\mathbf{x}_0) + D_{\mathrm{KL}}(q(\mathbf{x}_{1:T}|\mathbf{x}_0)||p_\theta(\mathbf{x}_{1:T}|\mathbf{x}_0))$$

Stable Diffusion!

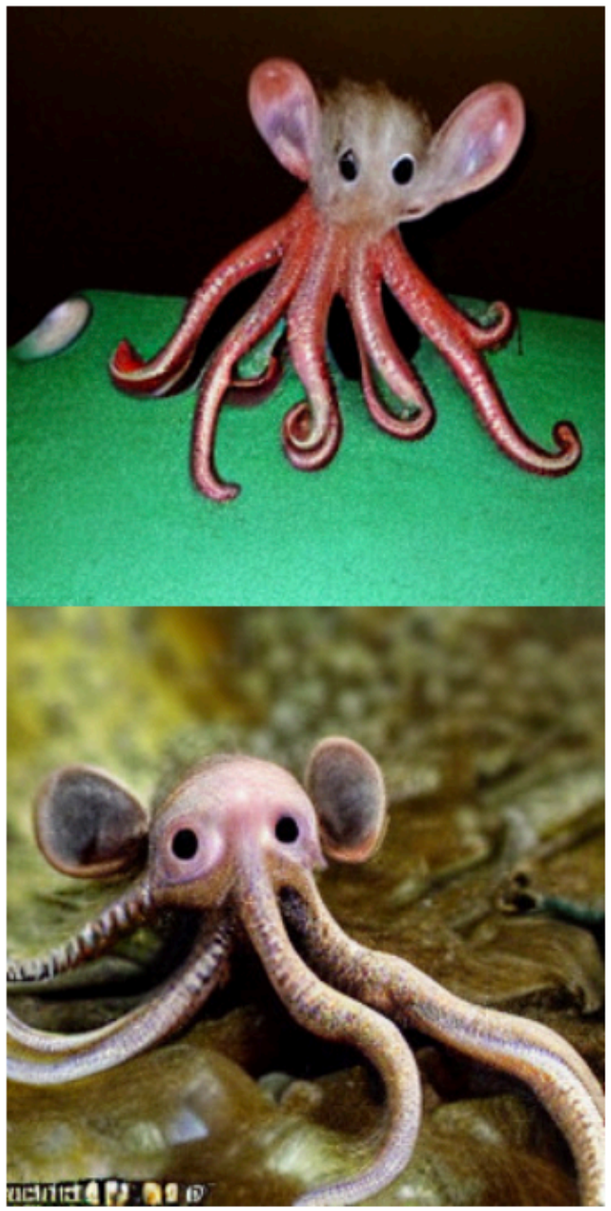


$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right) \cdot \mathbf{V}$$

$$\text{where } \mathbf{Q} = \mathbf{W}_Q^{(i)} \cdot \varphi_i(\mathbf{z}_i),\ \mathbf{K} = \mathbf{W}_K^{(i)} \cdot \tau_\theta(y),\ \mathbf{V} = \mathbf{W}_V^{(i)} \cdot \tau_\theta(y)$$

$$\text{and } \mathbf{W}_Q^{(i)} \in \mathbb{R}^{d \times d_\epsilon^i},\ \mathbf{W}_K^{(i)}, \mathbf{W}_V^{(i)} \in \mathbb{R}^{d \times d_\tau},\ \varphi_i(\mathbf{z}_i) \in \mathbb{R}^{N \times d_\epsilon^i},\ \tau_\theta(y) \in \mathbb{R}^{M \times d_\tau}$$

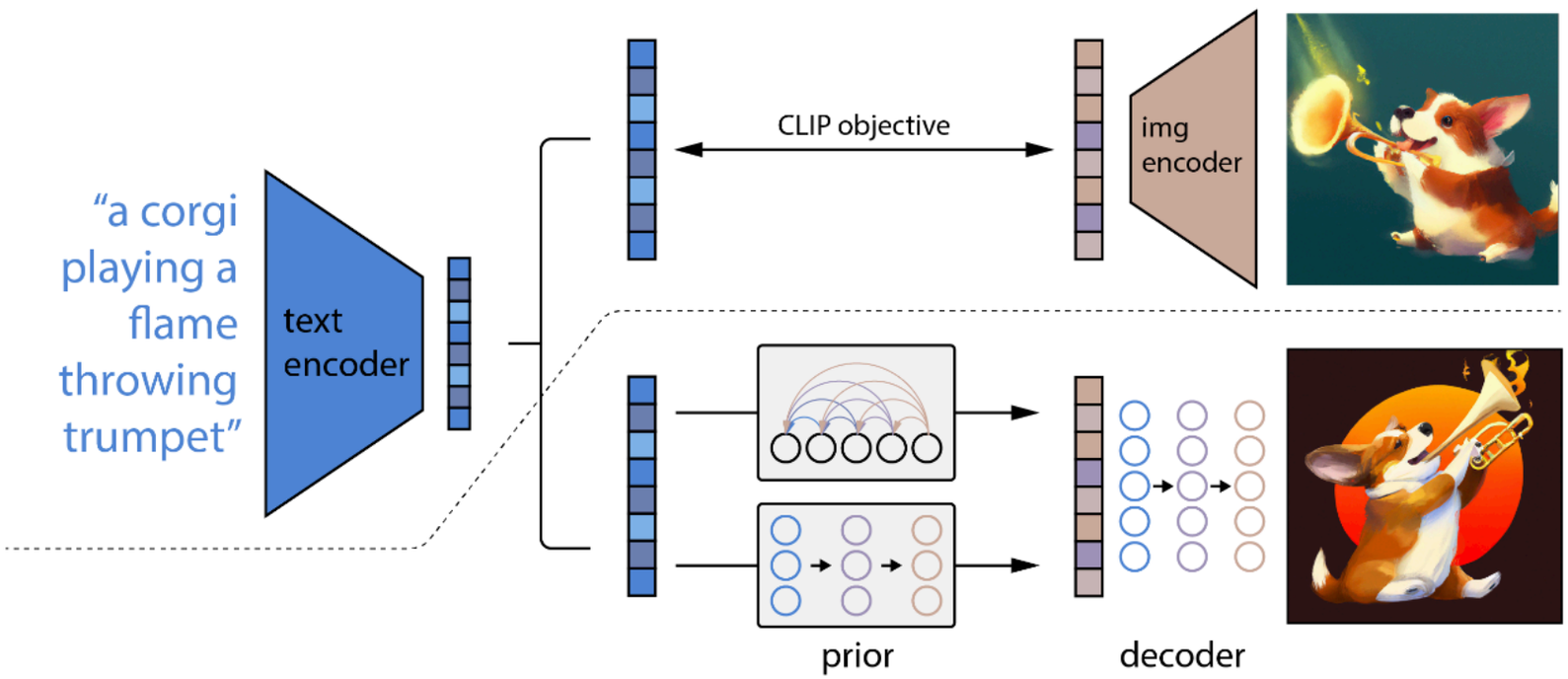
Stable Diffusion!

'A zombie in the style of Picasso'	*'An image of an animal half mouse half octopus'*
	

Try me!

DALL-E-2

See this Blog!



Try me!

Two kinds of prior distributions are proposed with roughly similar performance:

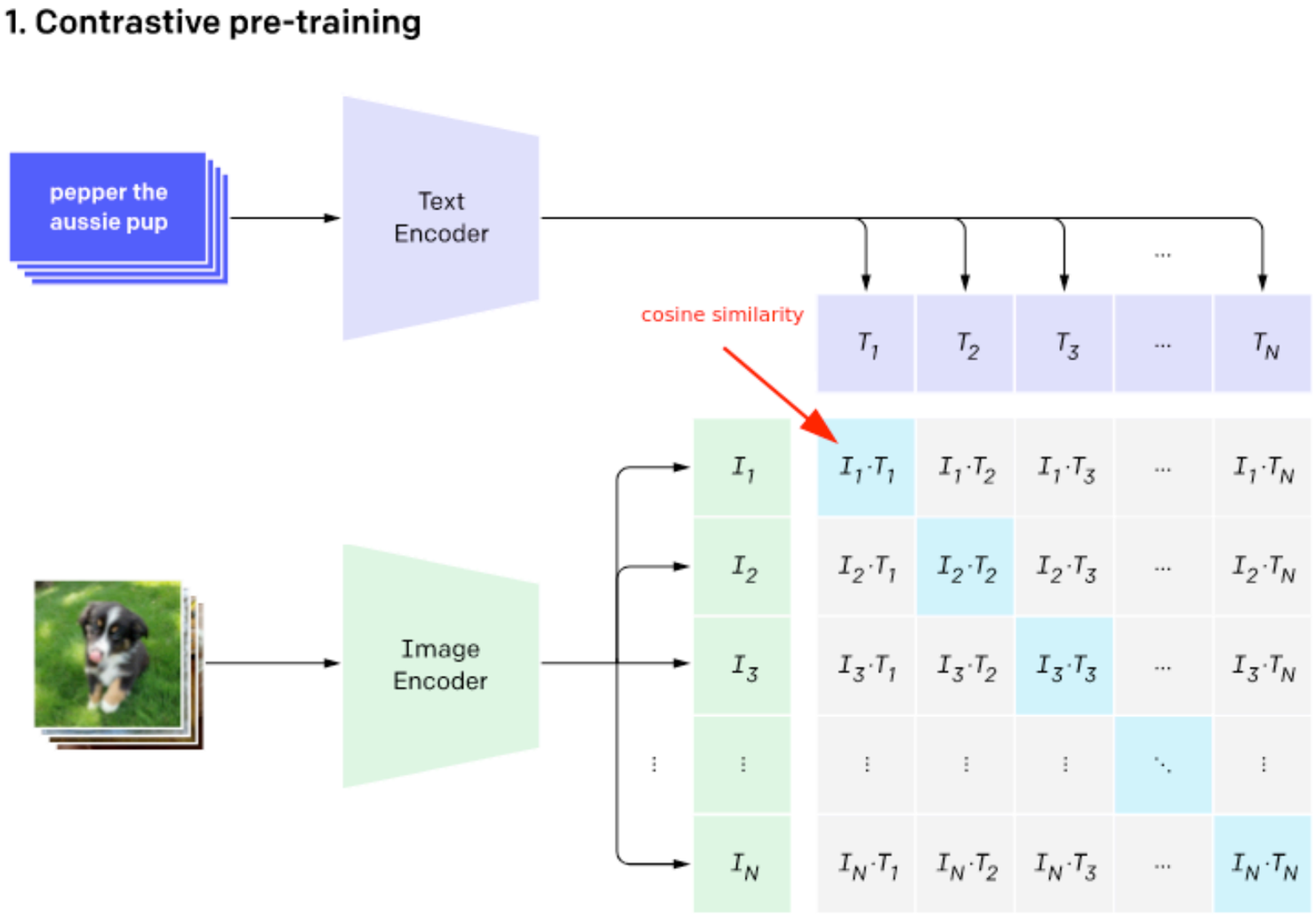
- An autoregressive prior over a discrete latent space (VQ-VAE)
- **Diffusion prior** just like in Stable Diffusion

CLIP pretraining to jointly encode images/text

CLIP pretraining in DALL-E-2

See this Blog!

- Encodes images and text to similar embeddings
- Text and image have separate transformer encoders
- Visual encoder is **ViT (vision transformer)**
- Text encoder is **GPT-2 transformer**
- Trained on 256 GPUs for 2 weeks

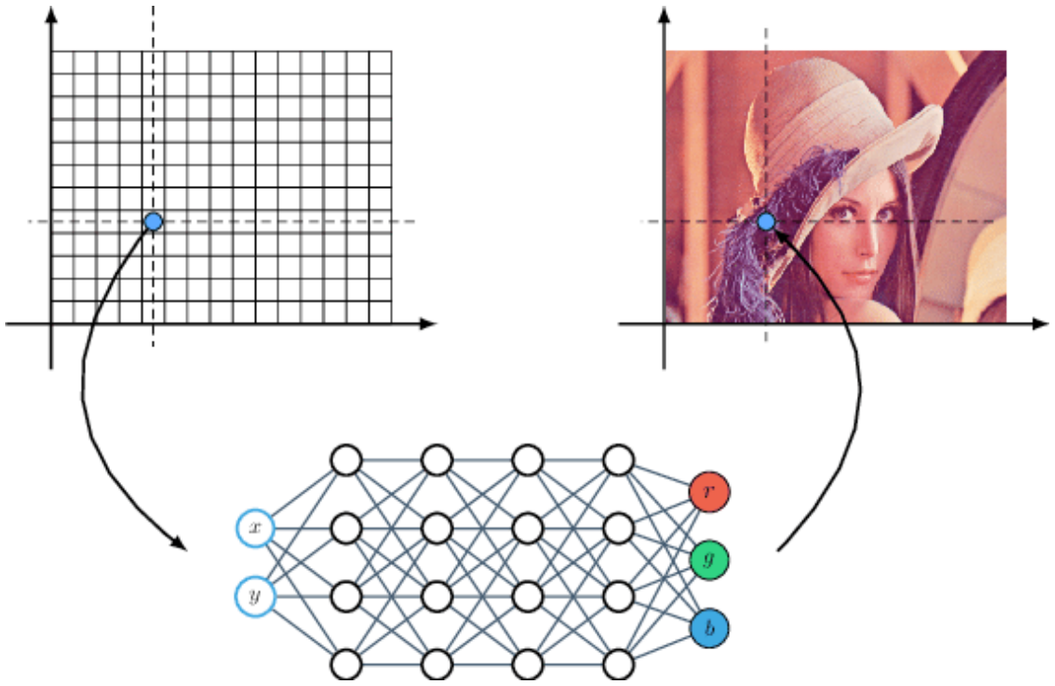


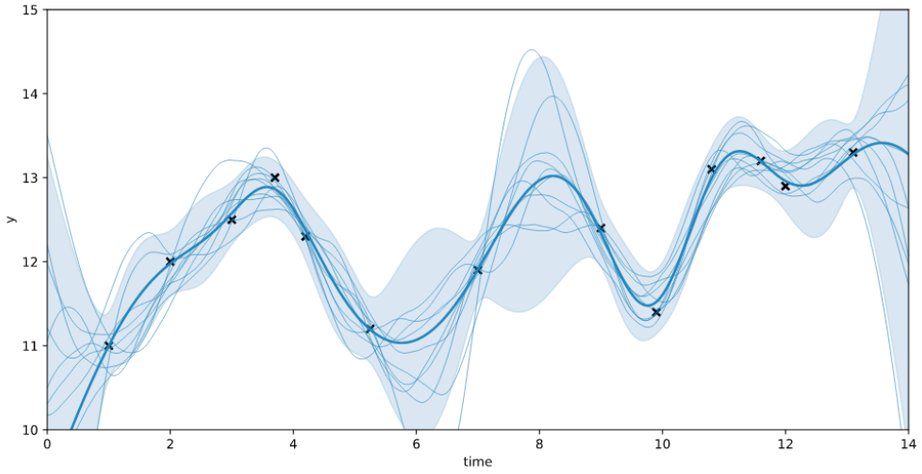
Neural Implicit Representations

uc3m

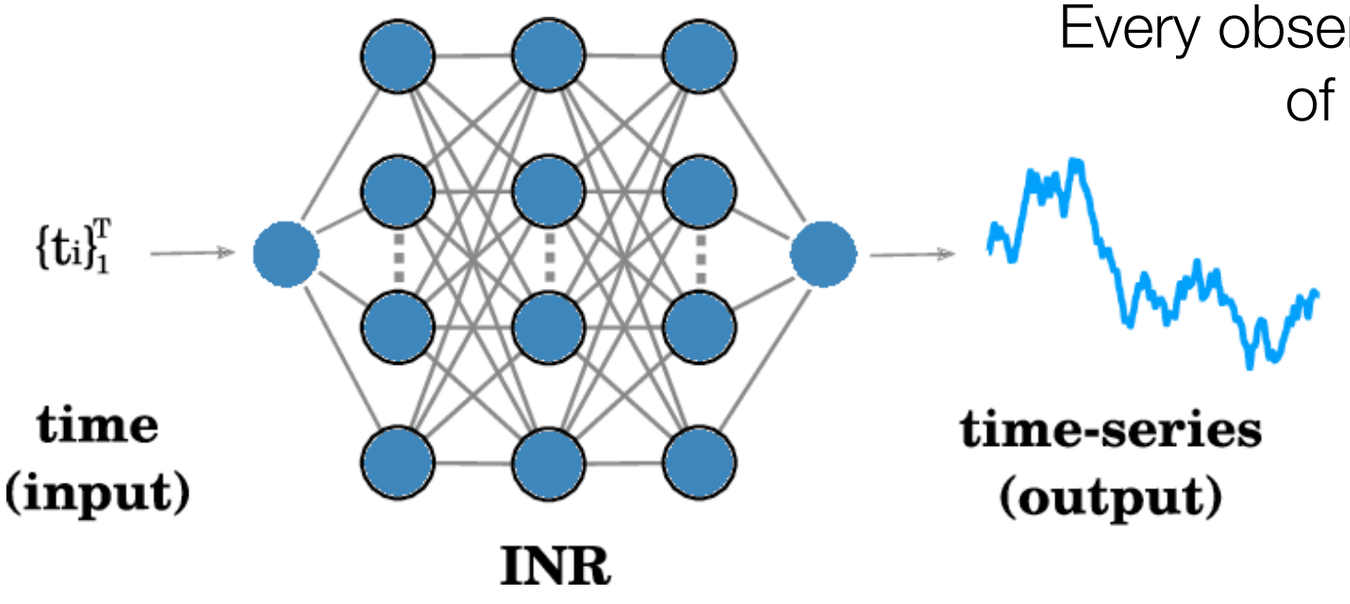
Basic idea

- Implicit neural representations is about **parameterizing a continuous differentiable signal** with a neural network



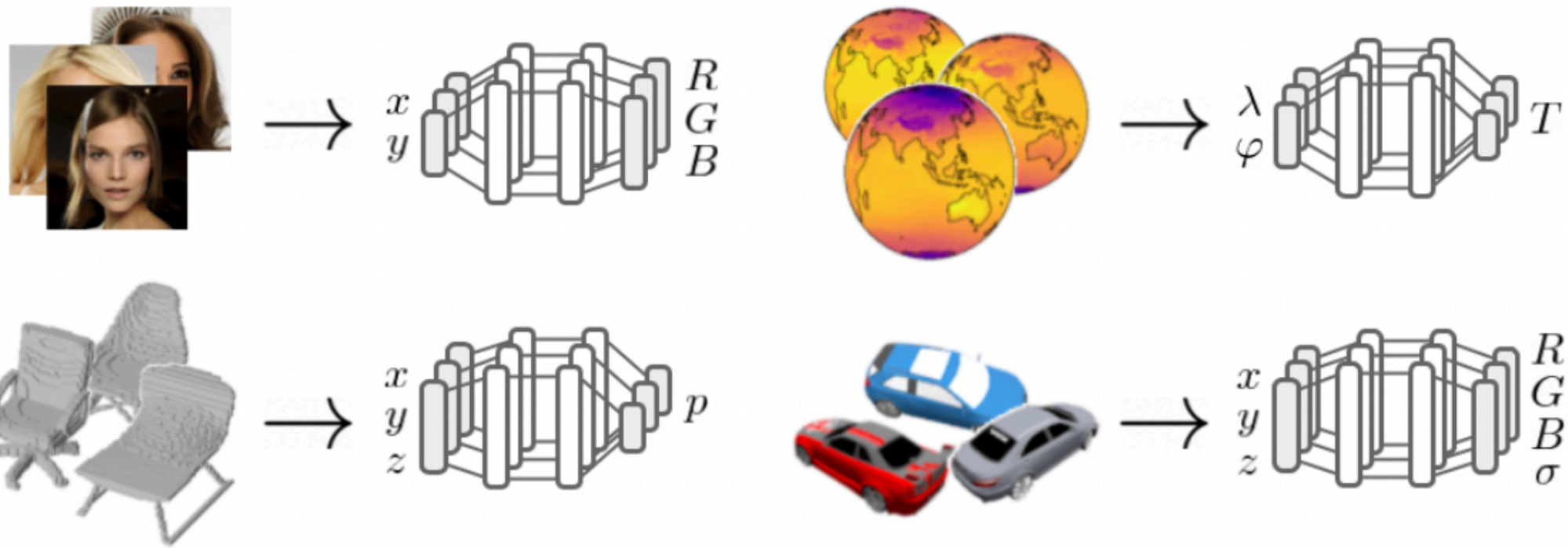


We learn parametric models for functions!
Every observation (pixel in an image) is an evaluation of such function at a given location



Basic idea

- No matter what kind of data you have, the NN nature is not changed!!
- It scales much better with resolution than array representations!



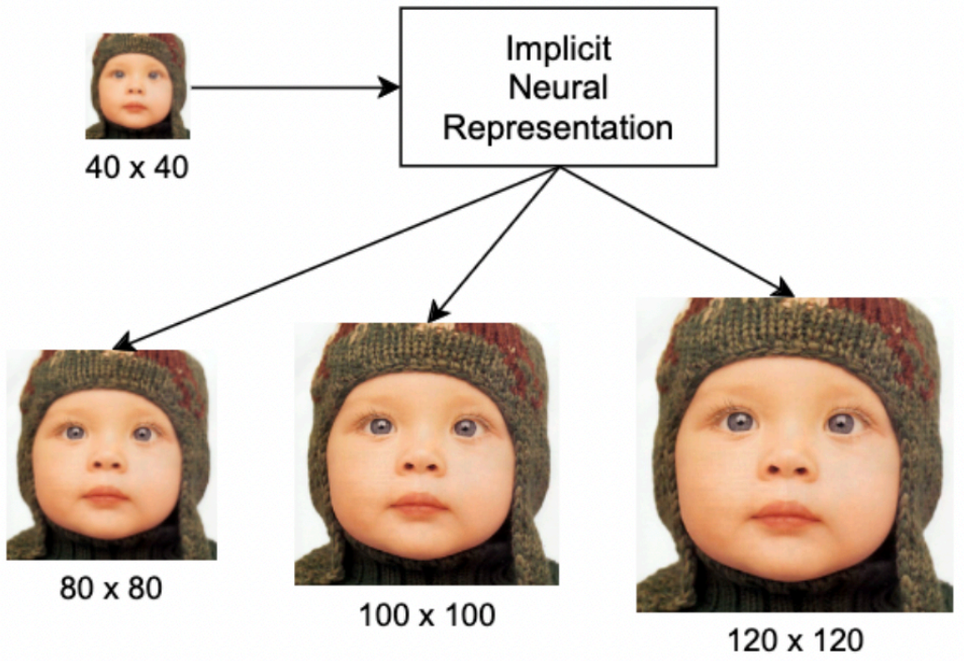
Implicit Neural Representations with Periodic Activation Functions (SIREN)

$$\Phi(\mathbf{x}) = \mathbf{W}_n \left(\phi_{n-1} \circ \phi_{n-2} \circ \ldots \circ \phi_0\right)(\mathbf{x}) + \mathbf{b}_n, \quad \mathbf{x}_i \mapsto \phi_i(\mathbf{x}_i) = \sin\left(\mathbf{W}_i \mathbf{x}_i + \mathbf{b}_i\right)$$

$$\tilde{\mathcal{L}} = \sum_i \|\Phi(\mathbf{x}_i) - f(\mathbf{x}_i)\|^2$$

Reconstruction

Super-resolution



DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation

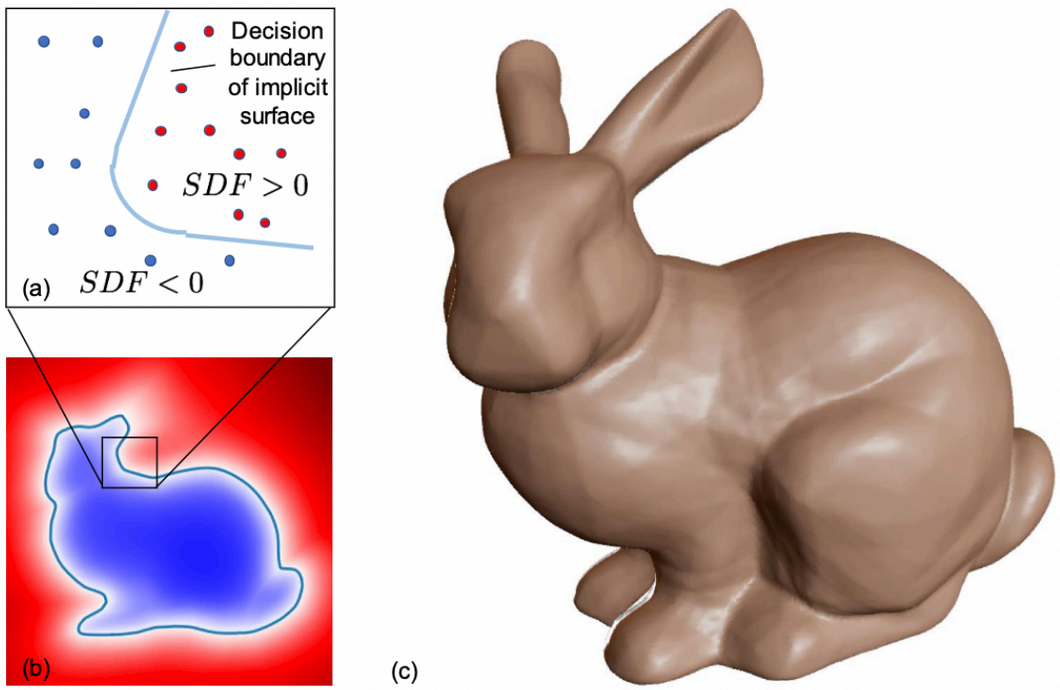


Figure 2: Our DeepSDF representation applied to the Stanford Bunny: (a) depiction of the underlying implicit surface $SDF = 0$ trained on sampled points inside $SDF < 0$ and outside $SDF > 0$ the surface, (b) 2D cross-section of the signed distance field, (c) rendered 3D surface recovered from $SDF = 0$. Note that (b) and (c) are recovered via DeepSDF.

$$SDF(\boldsymbol{x}) = s : \boldsymbol{x} \in \mathbb{R}^3,\ s \in \mathbb{R}\,.$$

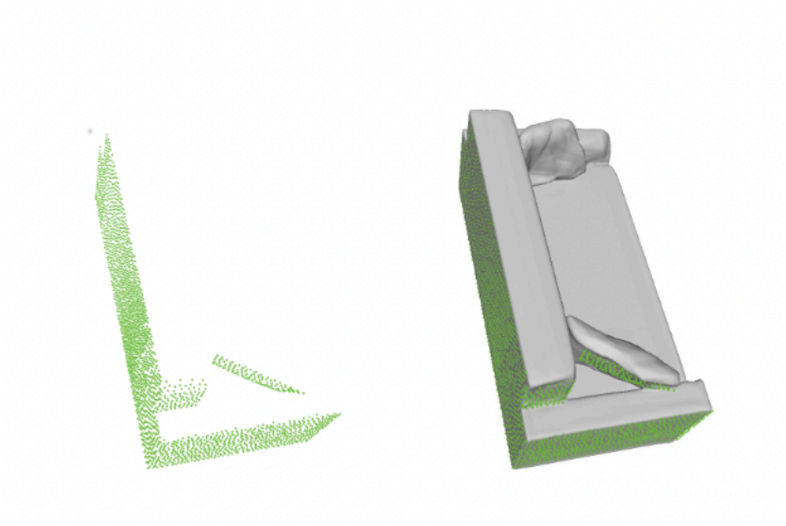
$$SDF(\cdot) = 0 \qquad \text{Surface}$$

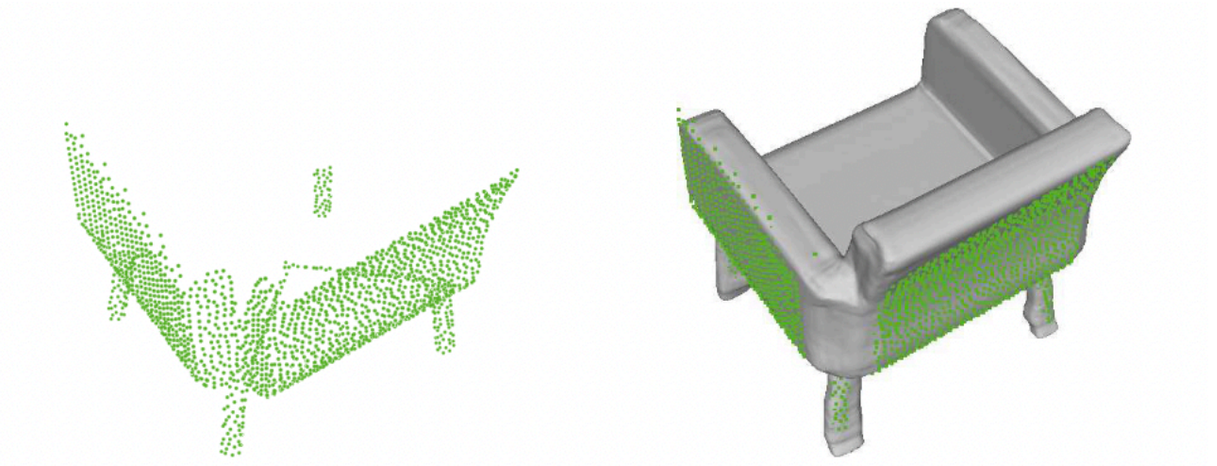
$$f_\theta(\boldsymbol{x}) \approx SDF(\boldsymbol{x}),\ \forall \boldsymbol{x} \in \Omega\,.$$

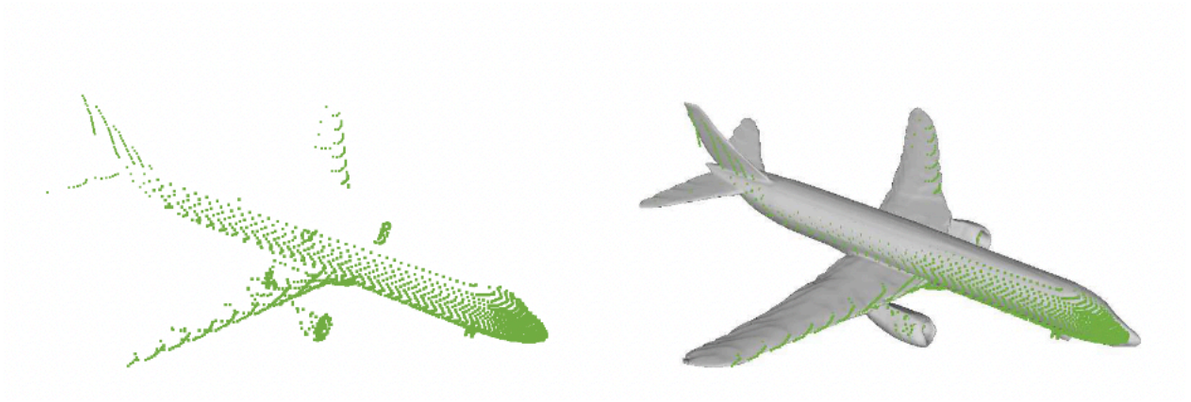
$$X := \{(\boldsymbol{x}, s) : SDF(\boldsymbol{x}) = s\}\,.$$

$$\mathcal{L}(f_\theta(\boldsymbol{x}), s) = |\,\text{clamp}(f_\theta(\boldsymbol{x}), \delta) - \text{clamp}(s, \delta)\,|$$

DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation



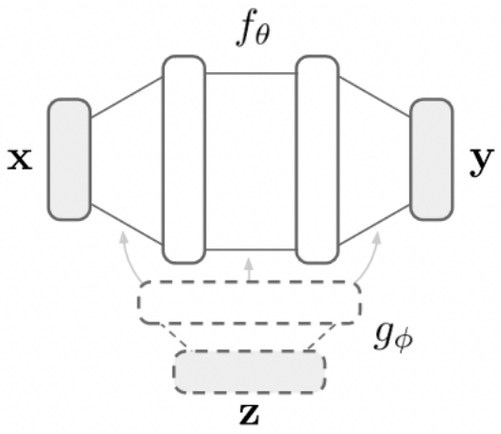


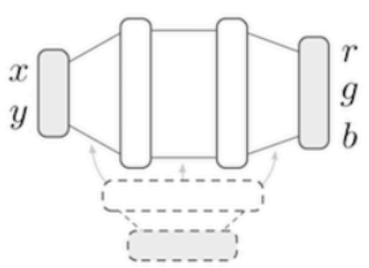


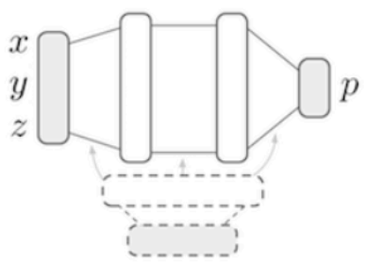
Probabilistic INR models

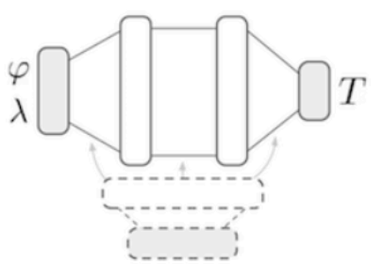
uc3m

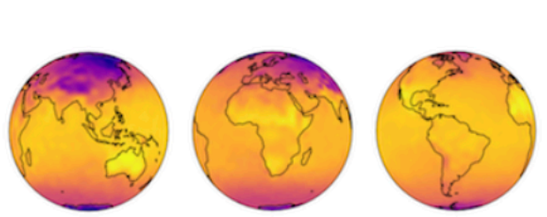
Generative Models as Distributions of Functions











Function generator use a **hypernetwork!**

A **GAN-style loss** function is used to train the generator

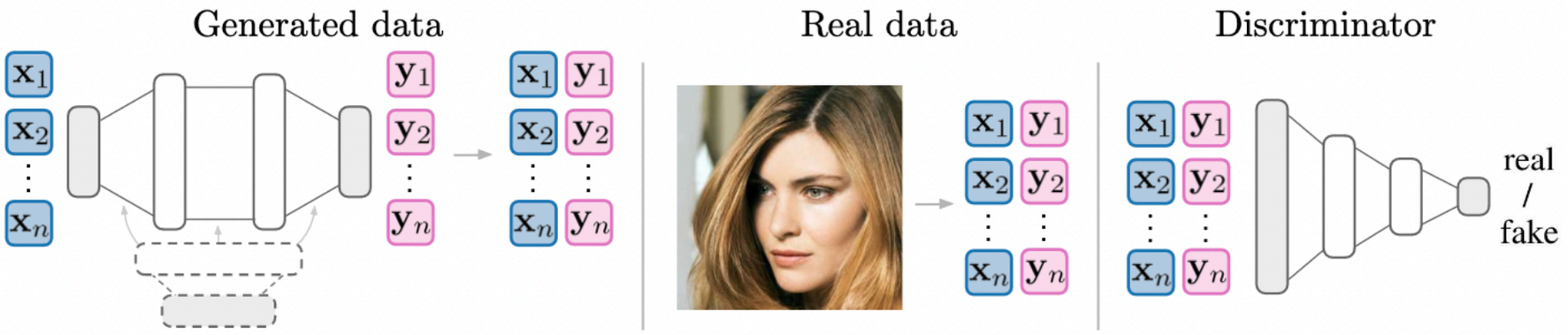


Figure 5: Training procedure for GASP: 1. Sample a function and evaluate it at a set of coordinate locations to generate fake point cloud. 2. Convert real data sample to point cloud. 3. Discriminate between real and fake point clouds.

Generative Models as Distributions of Functions

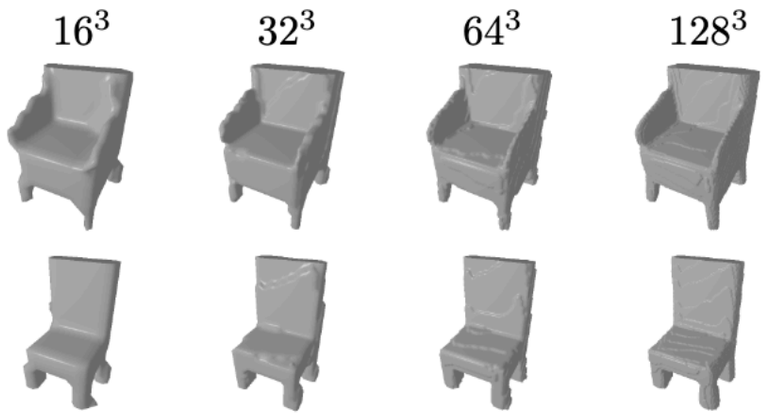


Figure 10: Evaluating the same function at different resolutions. As samples from our model can be probed at arbitrary coordinates, we can increase the resolution to render smoother meshes.

Figure 6: Samples from our model trained on CelebAHQ 64×64 (top) and 128×128 (bottom). Each image corresponds to a function which was sampled from our model and then evaluated on the grid. To produce this figure we sampled 5 batches and chose the best batch by visual inspection.

Generative Models as Distributions of Functions

Still not many things in SOTA. We have an ICML : VAMoH!

VARIATIONAL MIXTURE OF HYPERGENERATORS FOR LEARNING DISTRIBUTIONS OVER FUNCTIONS

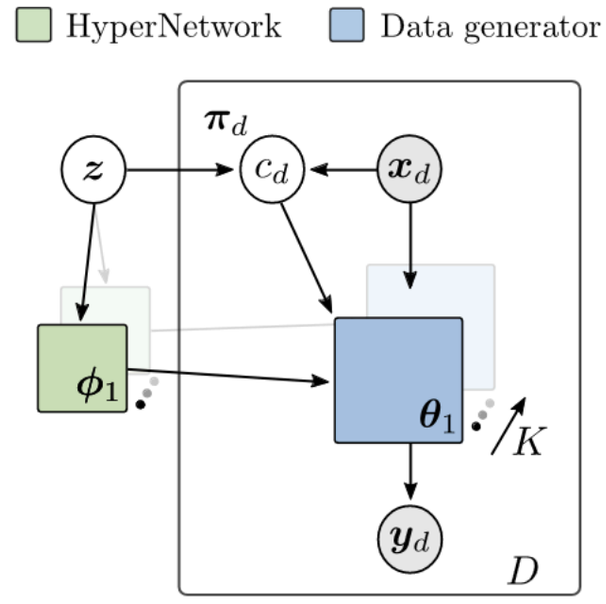
Batuhan Koyuncu[*]
Saarland University
Saarbrücken, Germany

Pablo Sánchez-Martín
Max Planck Institute for Intelligent Systems
Tübingen, Germany

Ignacio Peis
Universidad Carlos III de Madrid
Madrid, Spain

Pablo M. Olmos
Universidad Carlos III de Madrid
Madrid, Spain

Isabel Valera
Saarland University
Saarbrücken, Germany



(a) Generative model