

DeepMind

**Want to learn good
generative models?
Don't forget about the
deep learning bits!**

Mihaela Rosca
11/2023



Point of view

A lot of progress in generative models has been made not only due to changes in models and objectives, but a lot due to deep learning progress.

Today we will discuss a few case studies and provide historical context.

Disclaimer

This is a very broad overview of the field, there are many other interesting works that cannot be covered due to time constraints. If I missed anything, this is not a statement about the value of that respective work.

There are slides with additional references at the end of the talk.

Generative modeling

Learn a model of the true underlying data distribution $p^*(x)$ from samples

$$X_1, X_2 \dots X_n$$

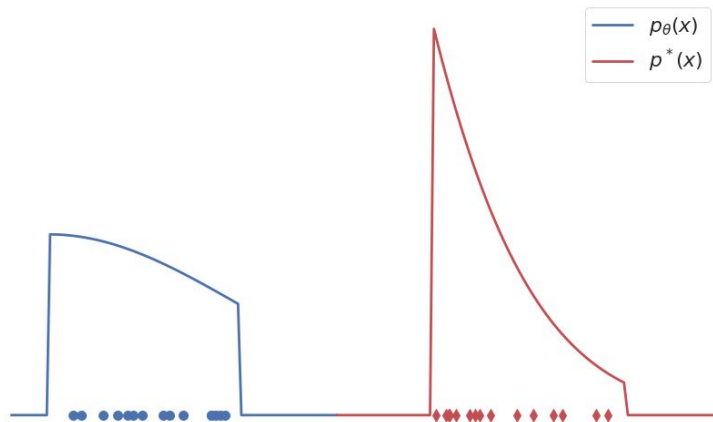


Generative modelling: a recipe

- Find an objective
- Find a model
- Find a way to learn the model using your objective

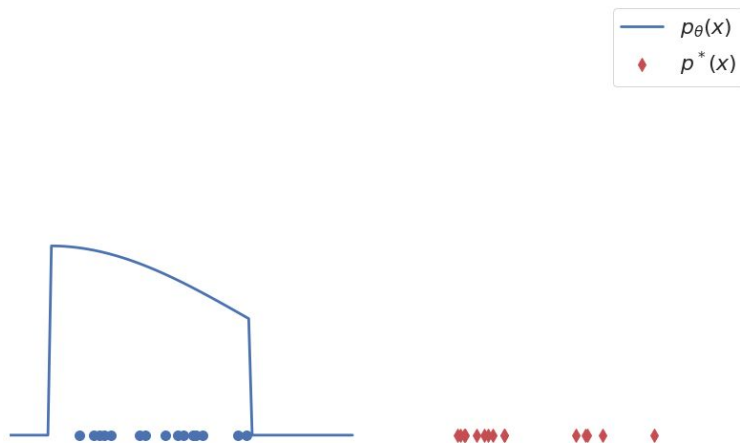
The objective

Measuring distances between distributions



How can we measure the distance between these two distributions?

Measuring distances between distributions



Caveat: we only have samples from the true distribution.

Divergence and distance minimization

- ➔ The objective of generative models is often to minimize a divergence or distance.
- ➔ Most common: Maximum likelihood (KL divergence).

Why divergence/distance minimization?

$$D(p^* || p_\theta) = 0 \implies p_\theta = p^*$$

KL divergence - maximum likelihood

$$\begin{aligned}\text{KL}(p^*(\mathbf{x}) || p_{\theta}(\mathbf{x})) &= \int p^*(\mathbf{x}) \log \frac{p^*(\mathbf{x})}{p_{\theta}(\mathbf{x})} d\mathbf{x} \\ &= C - \int p^*(\mathbf{x}) \log p_{\theta}(\mathbf{x}) d\mathbf{x}\end{aligned}$$

KL divergence - maximum likelihood

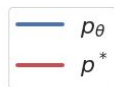
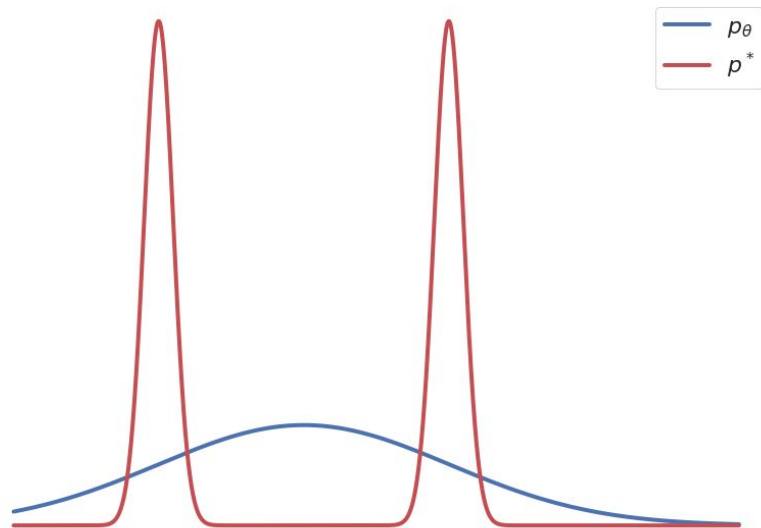
$$\begin{aligned} \min \text{KL}(p^*(\mathbf{x}) || p_{\theta}(\mathbf{x})) &= \int p^*(\mathbf{x}) \log \frac{p^*(\mathbf{x})}{p_{\theta}(\mathbf{x})} d\mathbf{x} \\ &= C - \int p^*(\mathbf{x}) \log p_{\theta}(\mathbf{x}) d\mathbf{x} \end{aligned}$$

$$\max \mathbb{E}_{p^*(\mathbf{x})} \log p_{\theta}(\mathbf{x})$$

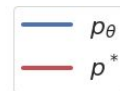
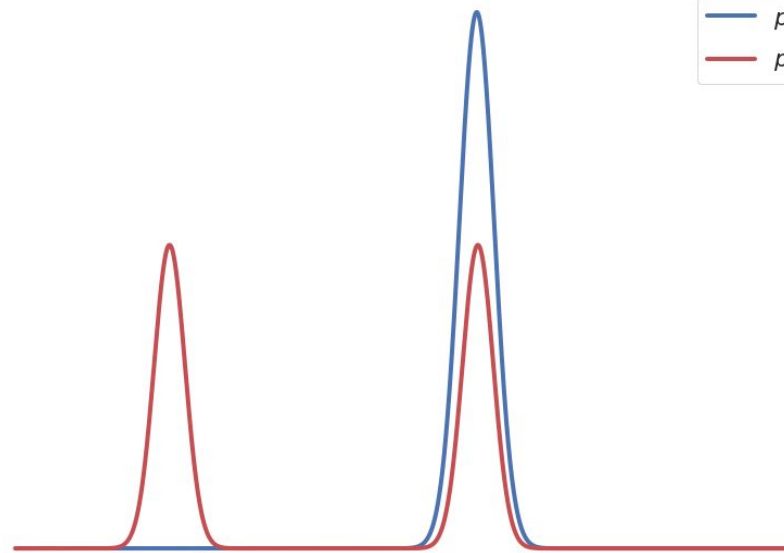


Effects of the choice of divergence

$$KL(p^*, p_\theta)$$

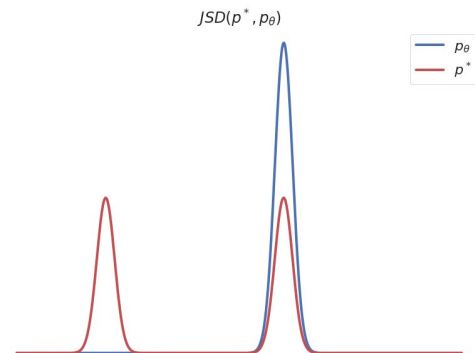
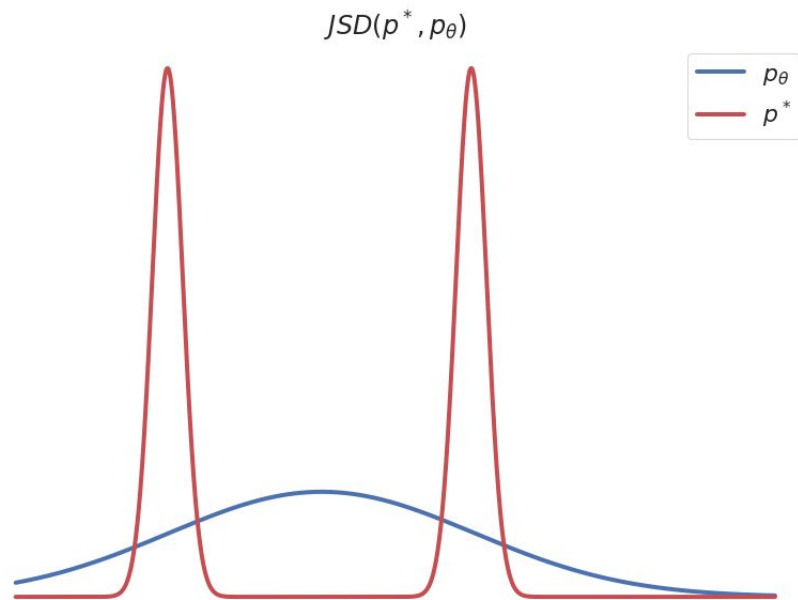


$$KL(p_\theta, p^*)$$



Jensen Shannon divergence

$$\text{JSD}(p_\theta, p^*) = \text{KL}\left(p_\theta, \frac{p_\theta + p^*}{2}\right) + \text{KL}\left(p^*, \frac{p_\theta + p^*}{2}\right)$$



Other divergences and distances

Want to learn more?



Arjovsky, et al Wasserstein GAN
ICML (2017)

Wasserstein Distance

$$W(p^*, p_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{p^*(\mathbf{x})} f(\mathbf{x}) - \mathbb{E}_{p_\theta(\mathbf{x})} f(\mathbf{x})$$

$$|f(x) - f(y)| \leq |x - y|$$

Other divergences and distances

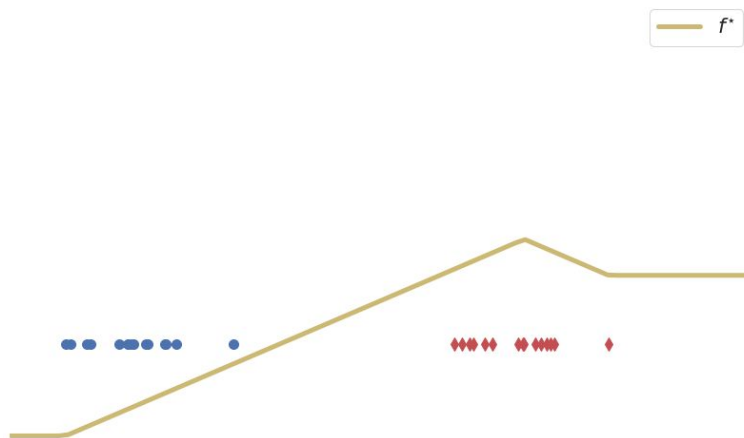
Want to learn more?



Arjovsky, et al Wasserstein GAN
ICML (2017)

Wasserstein Distance

$$W(p^*, p_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{p^*(\mathbf{x})} f(\mathbf{x}) - \mathbb{E}_{p_\theta(\mathbf{x})} f(\mathbf{x})$$



Given a (fixed) model family, different divergences can lead to a vastly different distribution.

Motivation for studying the effect of different divergences and distances and their properties.

Generative modelling: a recipe

- ~~Find an objective~~
- Find a model
- Find a way to learn the model using your objective

The model

Explicit models - autoregressive models

$$p_{\theta}(\mathbf{x}) = \prod_{i=0}^N p_{\theta}(x_i | x_{<i})$$

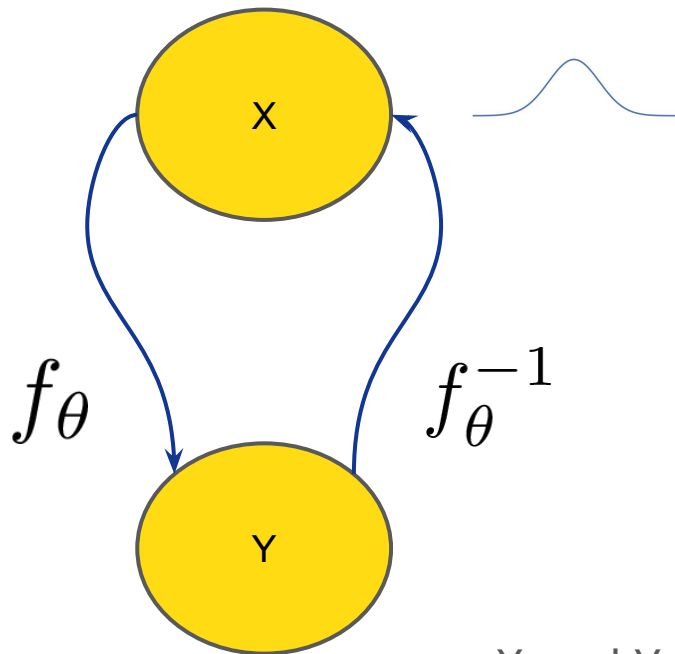
- Pro: Very expressive
- Challenge: Slow at sampling (though can be parallelize)

Explicit models - normalizing flows

Want to learn more?



Rezende et al, Variational Inference with Normalizing Flows, ICML 2015



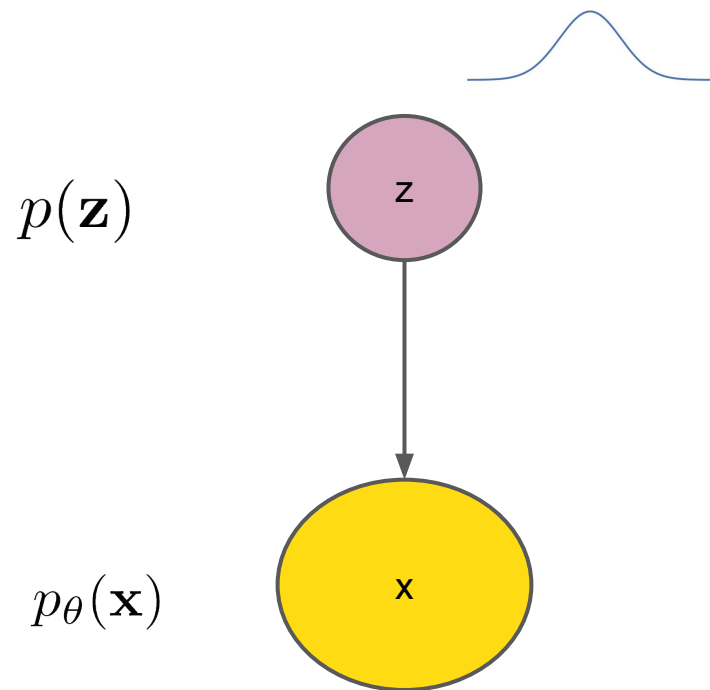
$$p_{\mathbf{y}}(\mathbf{y}) = p_{\mathbf{x}}(f_{\theta}^{-1}(\mathbf{y})) \det \left| \frac{df_{\theta}^{-1}(\mathbf{z})}{d\mathbf{z}} \right|_{\mathbf{z}=\mathbf{y}}$$

X and Y have the same dimension!

Challenge: modeling invertible functions using neural networks

Explicit latent variable models

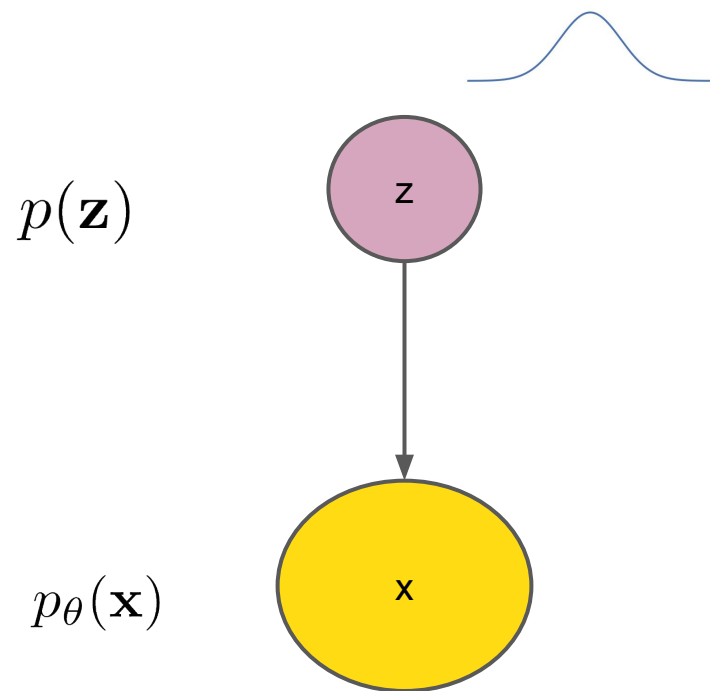
$$p_{\theta}(\mathbf{x}) = \int p_{\theta}(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}$$



Explicit latent variable models

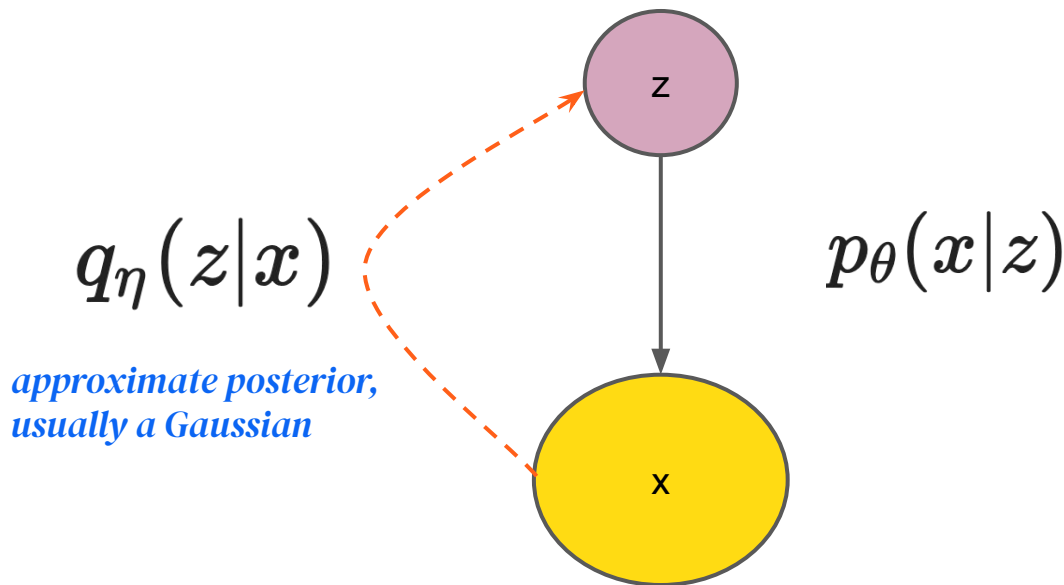
$$p_{\theta}(\mathbf{x}) = \int p_{\theta}(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}$$

$$\max \mathbb{E}_{p^*(\mathbf{x})} \log p_{\theta}(\mathbf{x})$$





Variational autoencoders need expressive posteriors

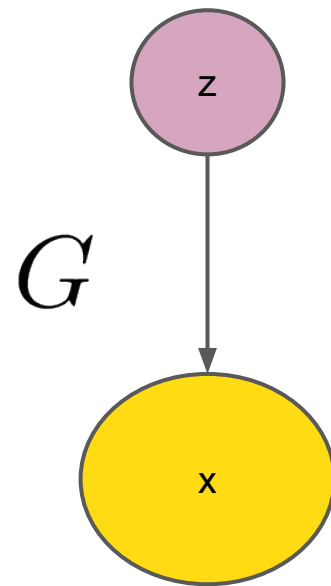


$$\log p_{\theta}(\mathbf{x}) \geq \mathbb{E}_{q_{\eta}(\mathbf{z}|\mathbf{x})} \log p_{\theta}(\mathbf{x}|\mathbf{z}) - \text{KL}(q_{\eta}(\mathbf{z}|\mathbf{x}) || p(\mathbf{z}))$$

Implicit models - latent variable models

Directly the sampling path, without require likelihoods explicitly (no need for the sum rule).

Caveat: you only get samples from the model.



Generative modelling: a recipe

- ~~Find an objective~~
- ~~Find a model~~
- Find a way to learn the model using your objective

$$\min_{\theta} E(\theta)$$

$$\min_{\theta} E(\theta)$$

Done: we know everything we need to care about when learning about generative models.

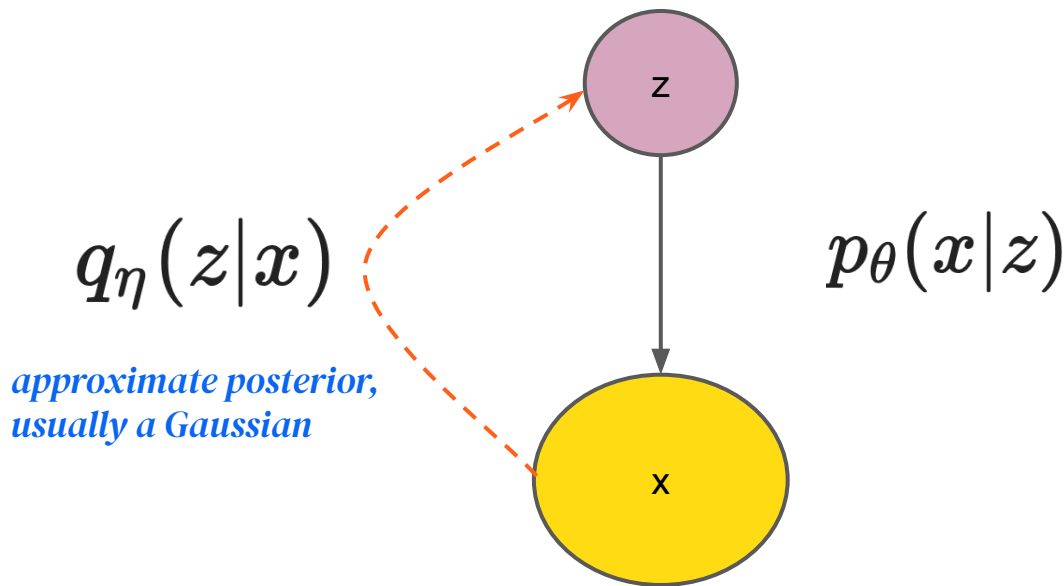
The secret sauce: training stability

Historical context

A lot of progress in generative modelling has been made due to improvements in training stability and optimisation, as well as architectural changes.

We will now go through a few examples.

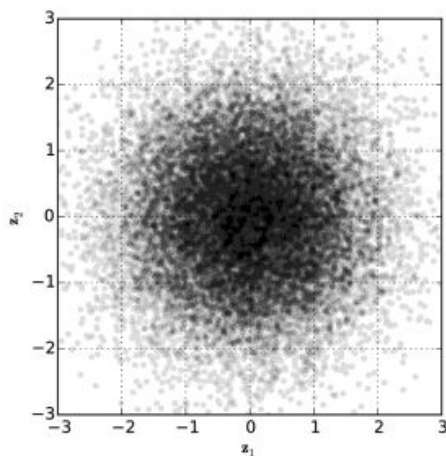
Variational autoencoders need expressive posteriors



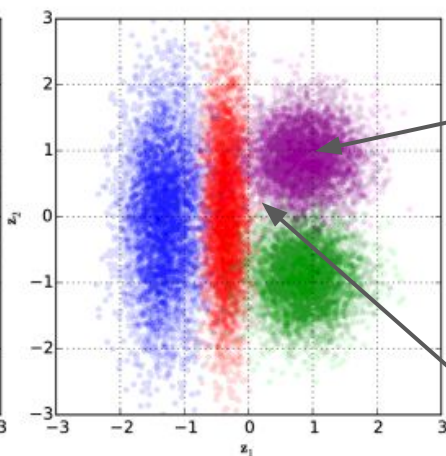
$$\log p_{\theta}(\mathbf{x}) \geq \mathbb{E}_{q_{\eta}(\mathbf{z}|\mathbf{x})} \log p_{\theta}(\mathbf{x}|\mathbf{z}) - \text{KL}(q_{\eta}(\mathbf{z}|\mathbf{x}) || p(\mathbf{z}))$$

Variational autoencoders need expressive posteriors

$$\mathbb{KL} [p^* || p] \leq C - \mathbb{E}_{p^*(x)} [\mathbb{E}_{q_\eta(z|x)} \log p_\theta(x|z) - \mathbb{KL}[q_\eta(z|x) || p(z)]]$$

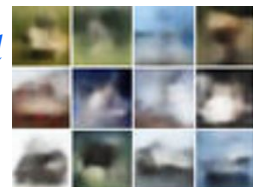


(a) Prior distribution



(b) Posteriors in standard VAE

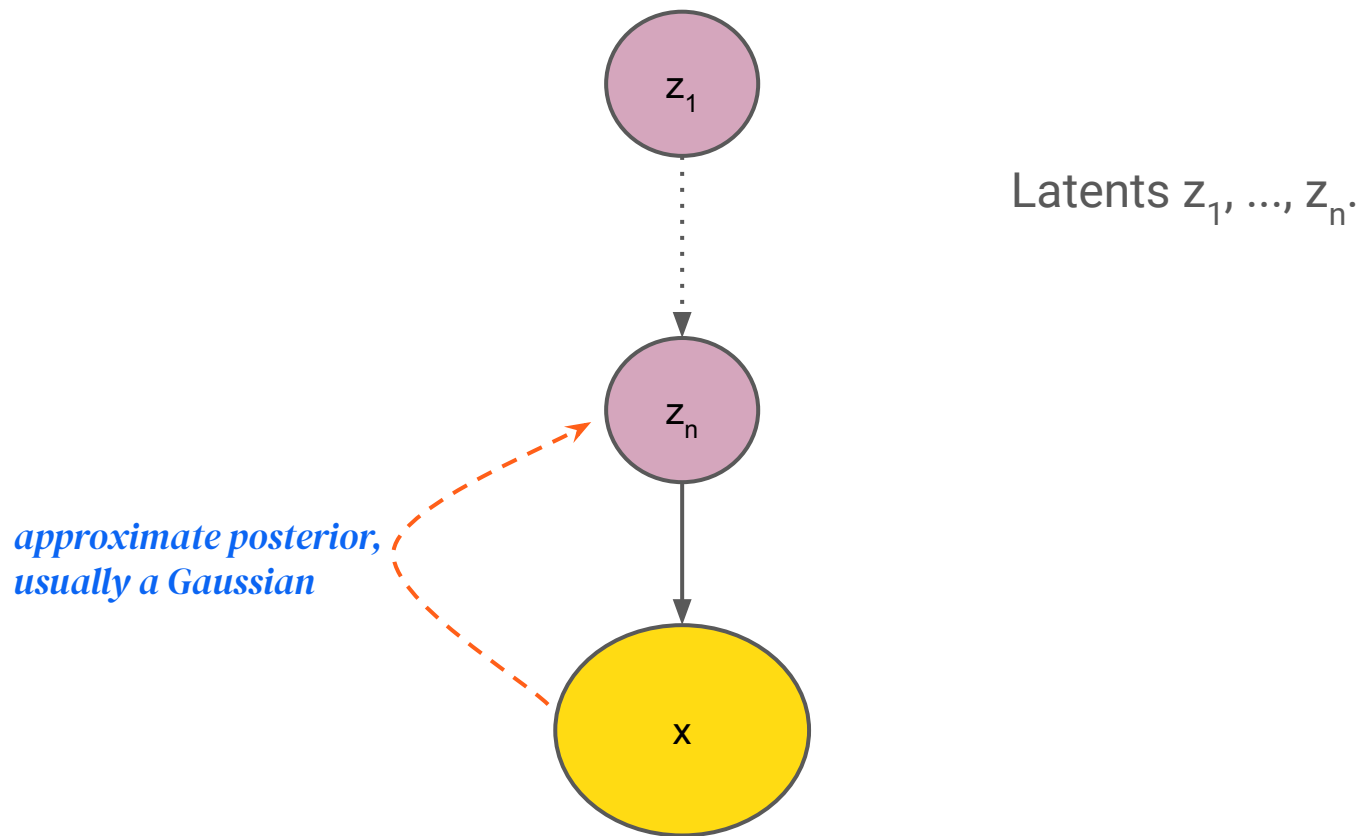
*individual
Gaussian
posteriors*



*prior mass
not covered by
posteriors*



Variational autoencoders need expressive posteriors



Variational autoencoders: flexible posteriors

$$\mathcal{L}_{\text{VAE}}(\mathbf{x}) := \mathbb{E}_{q(\mathbf{z}|\mathbf{x})} [\log p(\mathbf{x}|\mathbf{z})] - \text{KL}(q(\mathbf{z}_1|\mathbf{x})||p(\mathbf{z}_1)) - \sum_{l=2}^L \mathbb{E}_{q(\mathbf{z}_{<l}|\mathbf{x})} [\text{KL}(q(\mathbf{z}_l|\mathbf{x}, \mathbf{z}_{<l})||p(\mathbf{z}_l|\mathbf{z}_{<l}))]$$

Variational autoencoders: flexible posteriors

$$\mathcal{L}_{\text{VAE}}(\mathbf{x}) := \mathbb{E}_{q(\mathbf{z}|\mathbf{x})} [\log p(\mathbf{x}|\mathbf{z})] - \text{KL}(q(\mathbf{z}_1|\mathbf{x})||p(\mathbf{z}_1)) - \sum_{l=2}^L \mathbb{E}_{q(\mathbf{z}_{<l}|\mathbf{x})} [\text{KL}(q(\mathbf{z}_l|\mathbf{x}, \mathbf{z}_{<l})||p(\mathbf{z}_l|\mathbf{z}_{<l}))]$$

Challenge: hard to learn.

We have a probabilistic solution, but a practical problem.

NVAE: parametrization

The deep learning solutions:

- good activation functions
- batch normalization
- Spectral regularization for the posteriors
- residual parametrization

NVAE: parametrization

The deep learning solutions:

- good activation functions
- batch normalization
- Spectral regularization for the posteriors
 - encourage Lipschitz smoothness
 - posterior parameters should not change too much with changes in input
- residual parametrization for the posteriors

NVAE: parametrization

The deep learning solutions:

- good activation functions
- batch normalization
- Spectral regularization for the posteriors
- residual parametrization for the posteriors
 - gradients flow better (encoder needs to learn the difference to the prior)

NVAE: parametrization

The deep learning solutions:

- good activation functions
- batch normalization
- Spectral regularization for the posteriors
- residual parametrization for the encoder

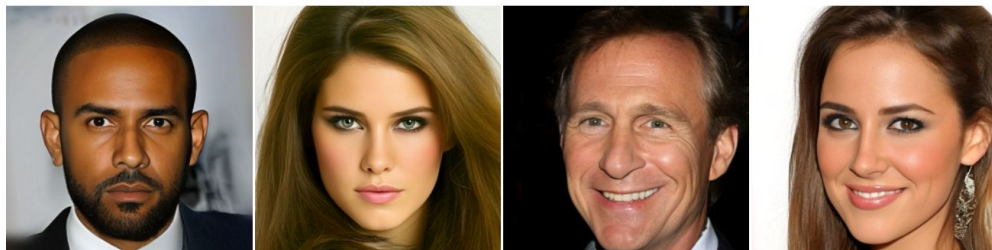
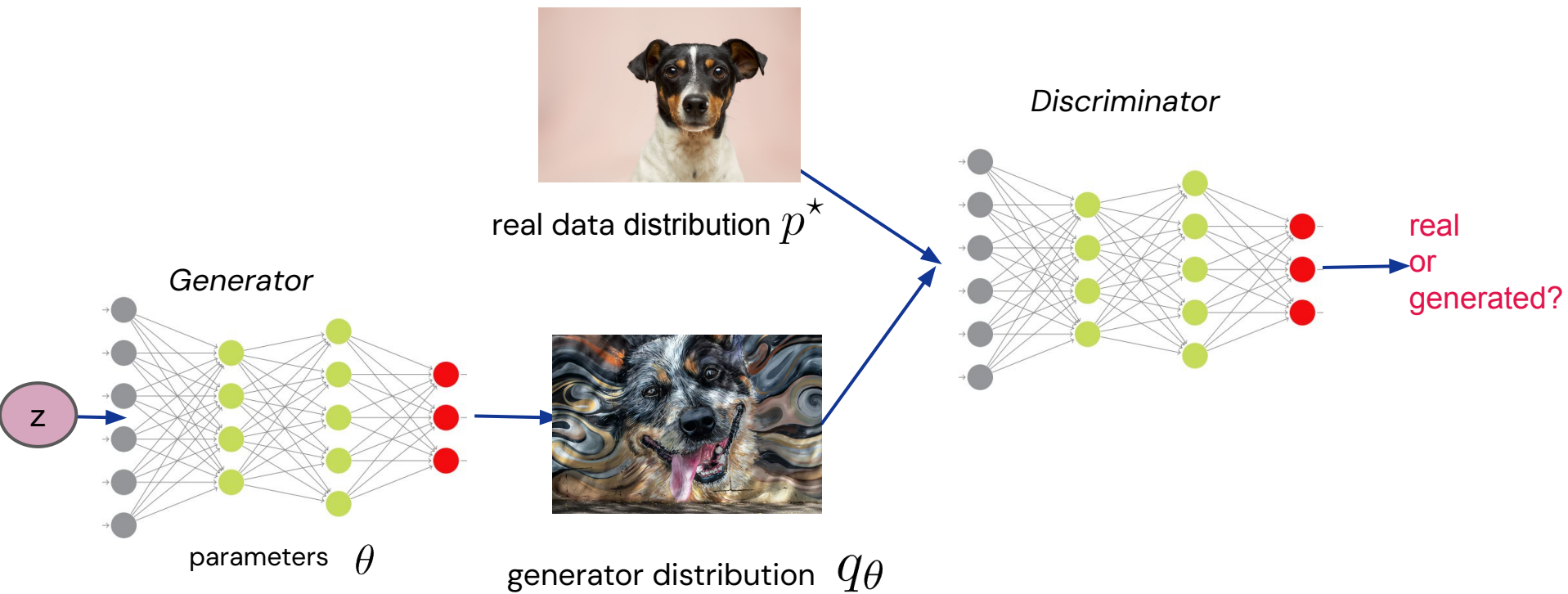


Figure 1: 256×256 -pixel samples generated by NVAE, trained on CelebA HQ [28].

Generative Adversarial Nets



Generative Adversarial Nets

How to answer “*real or generated?*”

Based on divergences (though not exact!)

- Original GAN: JSD
- Wasserstein GAN
- MMD-GAN
- f-GANs
 - f-divergences like the KL and JSD
- many more works!

Generative Adversarial Nets



?



Goodfellow, et al. *Generative adversarial networks*. NIPS (2014)

Karras et al. *A Style-Based Generator Architecture for Generative Adversarial Networks*. CVPR (2019)

Generative Adversarial Nets

A lot of progress in GAN samples was made with:

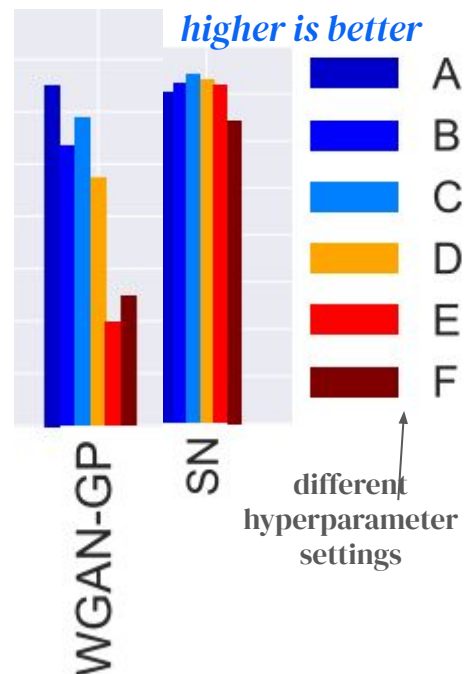
- changes to architecture
- increasing the importance of conditioning
- optimization

Generative Adversarial Nets: Optimization

Tricks to scale GANs:

- Use Spectral Normalization
 - decreases learning rate sensitivity
 - note: model normalization technique but affects optimization
 - (not only for GANs, also in RL)

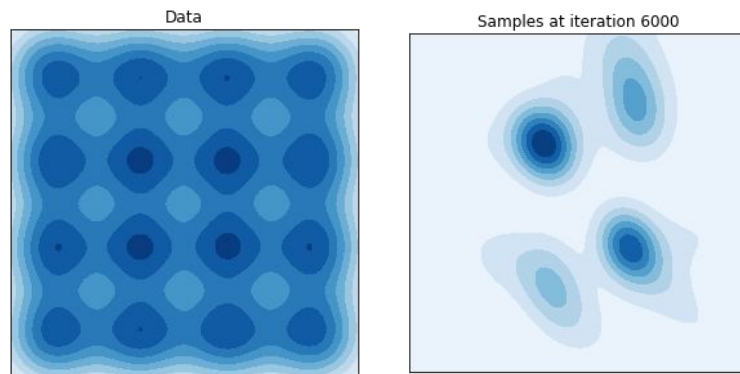
$$W \rightarrow W / \sigma(W)$$



Generative Adversarial Nets: Optimization

Tricks to scale GANs:

- optimization unstable
 - large batch sizes help



Generative Adversarial Nets: Optimization

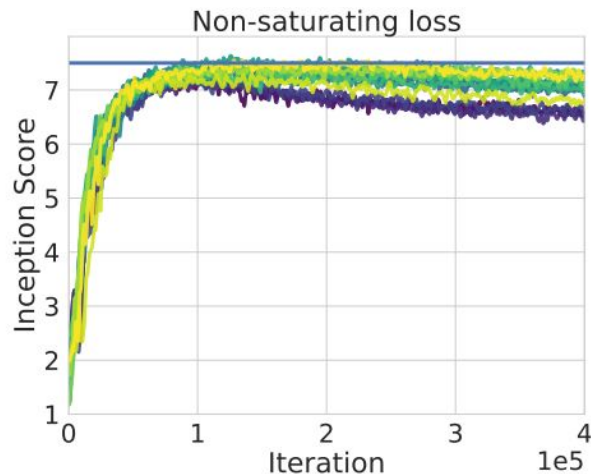
Tricks to scale GANs:

- Adam:
 - important
 - β_1 (momentum) low: 0 or 0.5

Generative Adversarial Nets: Optimization

Tricks to scale GANs:

- Adam:
 - important
 - β_1 (momentum) low: 0 or 0.5



GAN training
Higher is better

still, performance degrades later in training!

Generative Adversarial Nets: Optimization

Understanding optimization in generative models can be theoretical!

Observed problem:

gradient descent and its variants (Adam) struggle to train GANs.

Idea:

use continuous time (ODEs) to understand the training dynamics of the model!

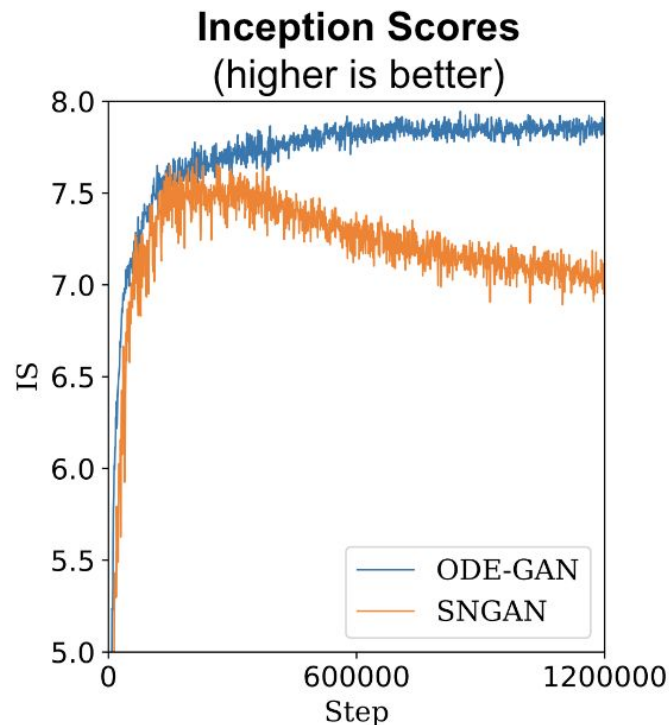
ODE-GAN

Theory:

Continuous dynamics converge to a local Nash equilibrium.

Practical implications:

Do not use gradient descent/Adam, but higher order integrators (RK4) to better follow continuous dynamics.

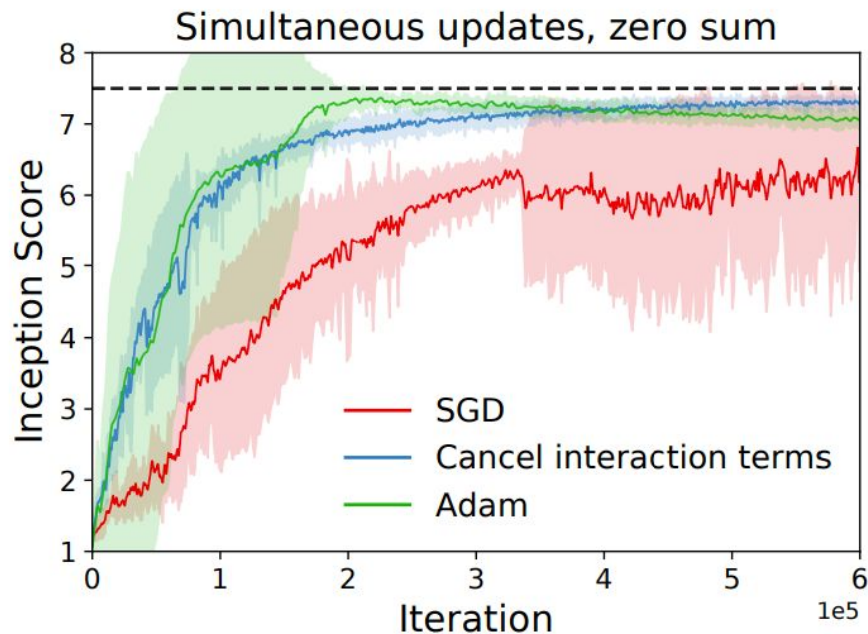


Discretization Drift in Two-Player games

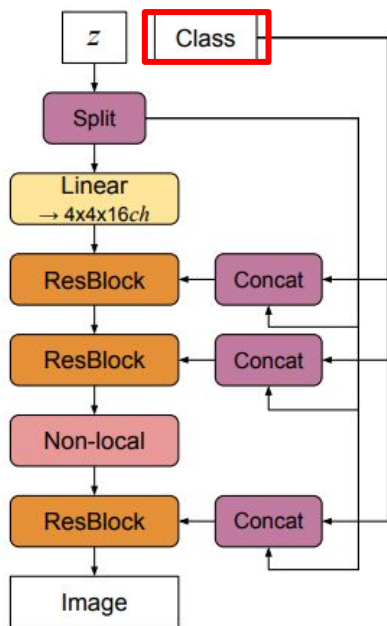
Question:

What makes gradient descent be unstable, when its continuous counterpart is not?

Quantify mathematically and use it to improve the performance of SGD in GANs!



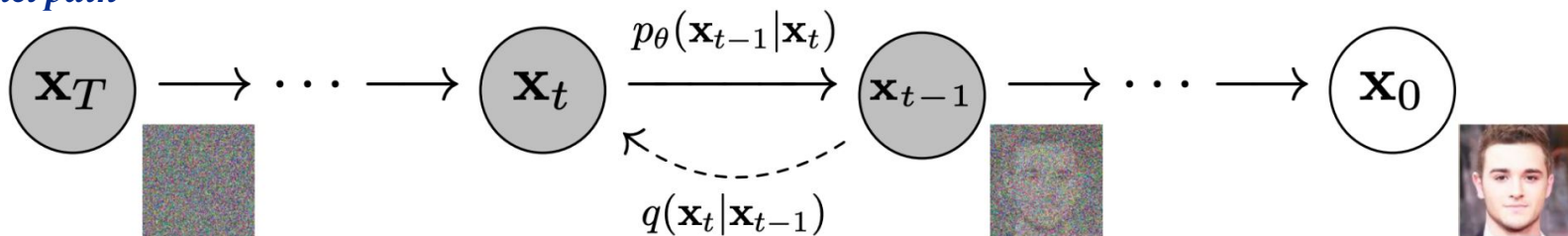
Generative Adversarial Nets: Conditioning



*insert conditioning all through the network
(not only concatenated with the input)*

Diffusion models

model path

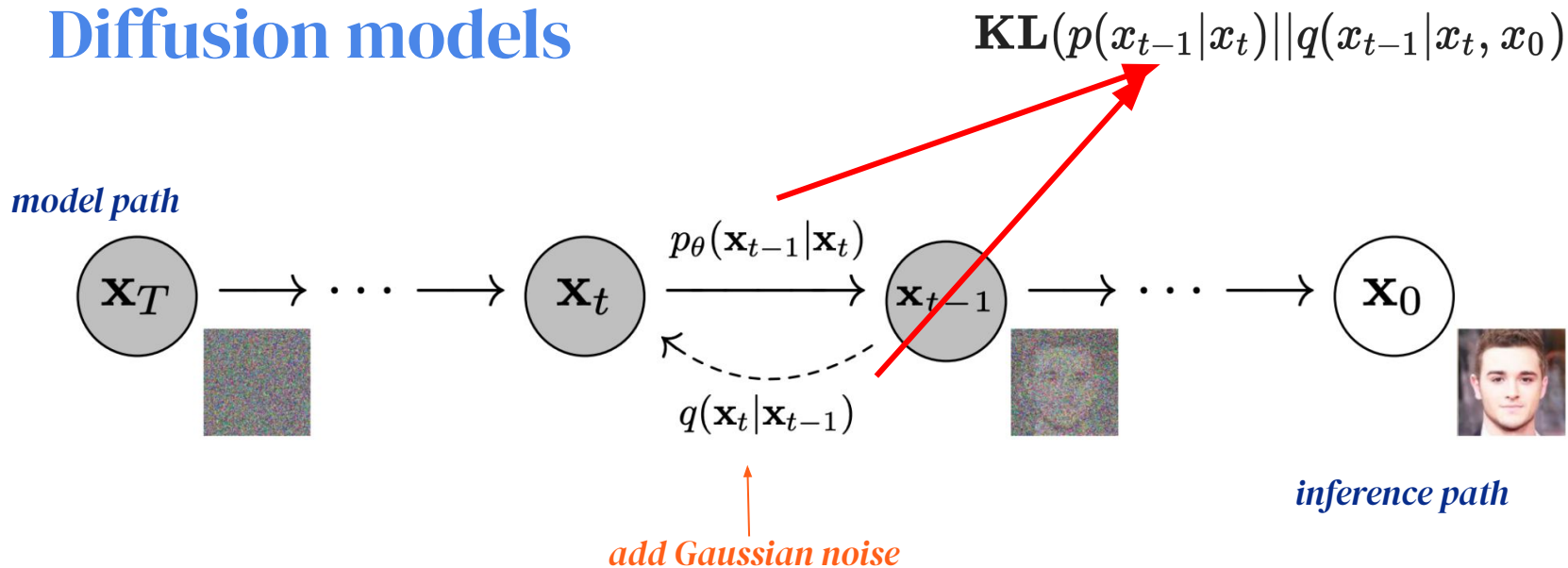


add Gaussian noise

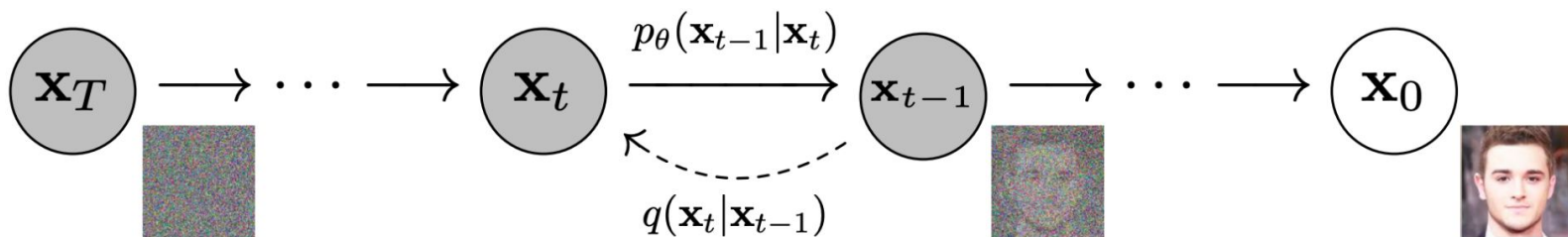
inference path

You can think about diffusion models as a particular formulation of hierarchical VAEs (though latents have same dimension as data)!

Diffusion models

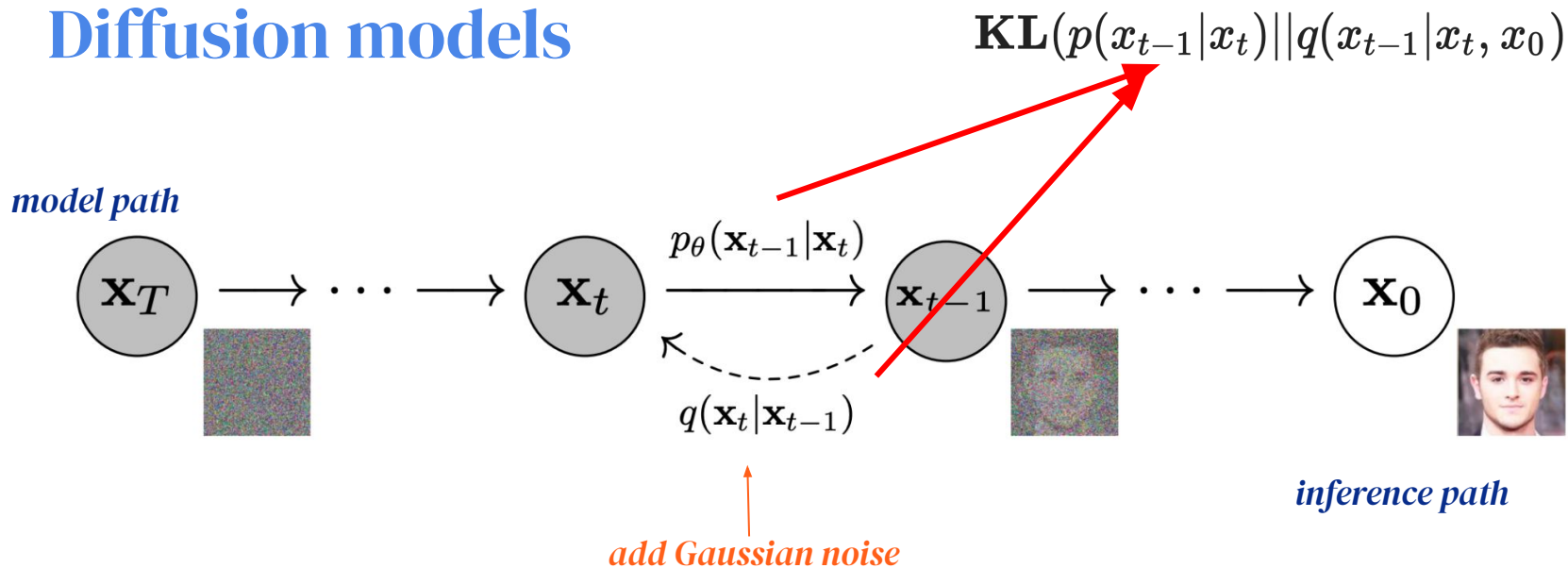


Diffusion models



- Simple optimization criteria, much more stable than GANs
- Conditioning

Diffusion models



Diffusion models - optimisation

Algorithm 1 Training

1: **repeat**

2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$

3: $t \sim \text{Uniform}(\{1, \dots, T\})$


4: $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

5: Take gradient descent step on

$$\nabla_{\theta} \left\| \epsilon - \epsilon_{\theta}(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon, t) \right\|^2$$

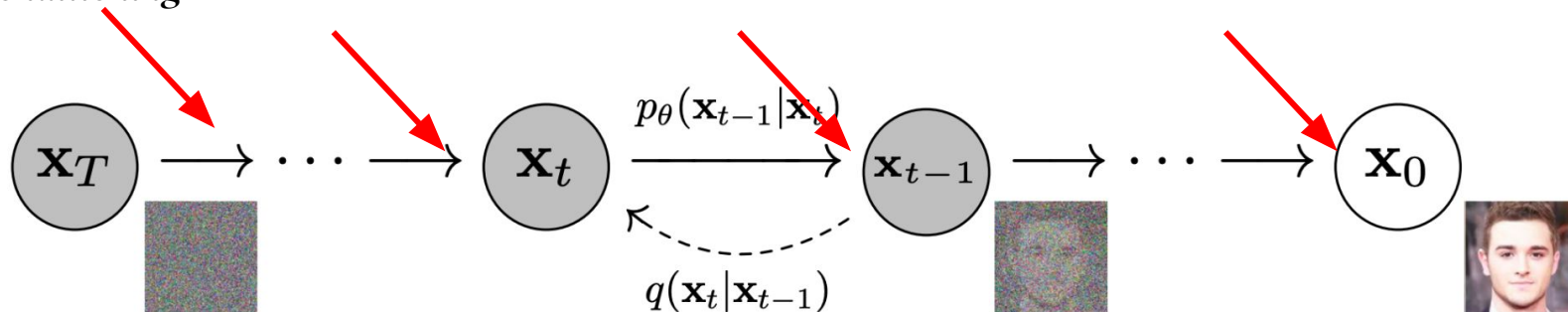
6: **until** converged

*regression loss to unit gaussian sample
dense learning signal due to specific
parametrisation of Gaussian mean*



Diffusion models - conditioning

conditioning



Iterative process: natural way to insert conditioning into the sampling process.

See: classifier guidance, classifier free guidance.

Maximum likelihood text LLMs

Great success with text generation done with transformers trained to maximize likelihood. Here too, *conditioning* has been crucial.

Challenge: how to stabilize transformers at larger scale (entropy collapse of attention layers).

Unlike other models, transformers cannot easily be trained with SGD with momentum, and need Adam.

Conclusion



Thanks!

References

Variational Autoencoders

Auto-Encoding Variational Bayes, Kingma et al, ICLR 2014

Variational Inference with Normalizing Flows, Jimenez Rezende et al, ICML 2015

Improved Variational Inference with Inverse Autoregressive Flow, Kingma et al, Neurips 2016

Draw: A recurrent neural network for image generation, Gregor et al, ICML 2015

Auxiliary deep generative models, Maaløe et al, ICML 2016

Hierarchical variational models, Ranganath et al, ICML 2016

NVAE: A Deep Hierarchical Variational Autoencoder, Vahdat et al, Neurips 2020

GANs

GANs introduced based on divergences and distances:

Generative adversarial nets, Goodfellow et al, Neurips 2014

f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization, Nowozin, et al, Neurips (2016)

Wasserstein GAN, Arjovsky, et al, ICML 2017

MMD GAN: Towards Deeper Understanding of Moment Matching Network, Li, et al, Neurips 2017

Improved Training of Wasserstein GANs, Gulrajani et al, Neurips 2017

Demystifying mmd gans, Bińkowski et al, ICL 2018

Learning in implicit generative models, Mohamed, et al arxiv (2016)

GANs

Understanding the effect of discretisation

ODE-GAN: Training Generative Adversarial Networks by Solving Ordinary Differential Equations, Qin et al, Neurips 2020

Implicit competitive regularization in gans, Schäfer et al, ICML 2021

Discretization Drift in Two-Player Games, Rosca et al, ICML 2021

The limit points of (optimistic) gradient descent in min-max optimization, Daskalakis et al, Neurips 2018

Which divergence and distance to use?

Towards Principled Methods for Training Generative Adversarial Networks, Arjovsky et al, ICLR 2017

Many Paths to Equilibrium: GANs Do Not Need to Decrease a Divergence At Every Step, Fedus et al, ICLR 2018

Which divergence and distance to use?

Large Scale GAN Training for High Fidelity Natural Image Synthesis, Brock et al, ICLR 2018

A Style-Based Generator Architecture for Generative Adversarial Networks, Karras et al, CVPR 2019

Diffusion models

Deep Unsupervised Learning using Nonequilibrium Thermodynamics, Sohl-Dickstein et al, ICML 2015

Generative Modeling by Estimating Gradients of the Data Distribution, Song et al, Neurips 2019

Denoising Diffusion Probabilistic Models, Ho et al, Neurips 2020

Improved Denoising Diffusion Probabilistic Models, Nichol et al, 2021

High-Resolution Image Synthesis with Latent Diffusion Models, Rombach et al, 2022

Diffusion Models Beat GANs on Image Synthesis, Dhariwal et al, Neurips 2021

Classifier-Free Diffusion Guidance, Ho et al, NeurIPS workshop on DGMs and Applications”, 2021.

Maximum Likelihood Training of Score-Based Diffusion Models, Song et al, 2021

Hierarchical Text-Conditional Image Generation with CLIP Latents, Dhariwal et al, 2021

Blog posts:

[What are Diffusion Models? by Lilian Weng](#)

[Guidance: a cheat code for diffusion models, Sander Dieleman](#)

Transformer LLMs and optimisation

Large language models can suffer from optimisation and training related challenges

A Theory on Adam Instability in Large-Scale Machine Learning, Molybog et al, 2023

Stabilizing Transformer Training by Preventing Attention Entropy Collapse, Zhai et al, 2023

Palm: Scaling language modeling with pathways, Chowdhery et al, 2022

Toward Understanding Why Adam Converges Faster Than SGD for Transformers, Pan et al, 2023

Diffusion - classifier guidance and classifier-free guidance

Classifier guidance:

- does not need adjusting the training procedure, only the sampling procedure
- challenge: relying on classifier output for out of distribution, noisy data

$$p(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{c}) \propto p(\mathbf{x}_{t-1} | \mathbf{x}_t) \underbrace{p(\mathbf{c} | \mathbf{x}_{t-1})}_{\text{pretrained classifier}}$$

Classifier-free guidance:

- update the training procedure
- train both conditional and unconditional diffusion model

$$p(\mathbf{c} | \mathbf{x}_{t-1}, \mathbf{x}_t) \propto \frac{p(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{c})}{p(\mathbf{x}_{t-1} | \mathbf{x}_t)}$$

Normalisation and regularisation techniques

Spectral Normalization for Generative Adversarial Networks, Miyato et al, ICLR 2018

Spectral Norm Regularization for Improving the Generalizability of Deep Learning, Yoshida et al, 2017

Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, Ioffe et al, ICML 2015

Layer Normalization, Ba et al, 2016

Group Normalization, Wu et al, 2018

A case for new neural network smoothness constraints, Rosca et al, 2020