Generative Modelling Intro

Matt Smart

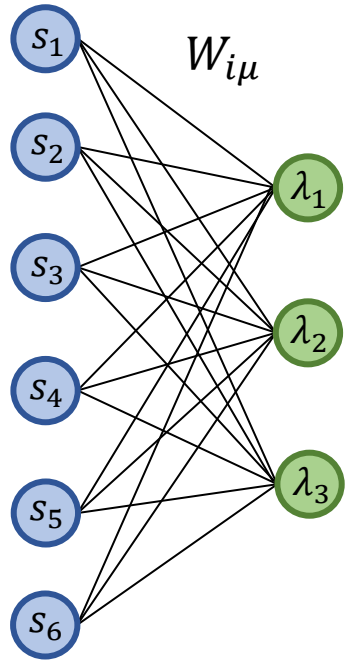
https://thispersondoesnotexist.com/

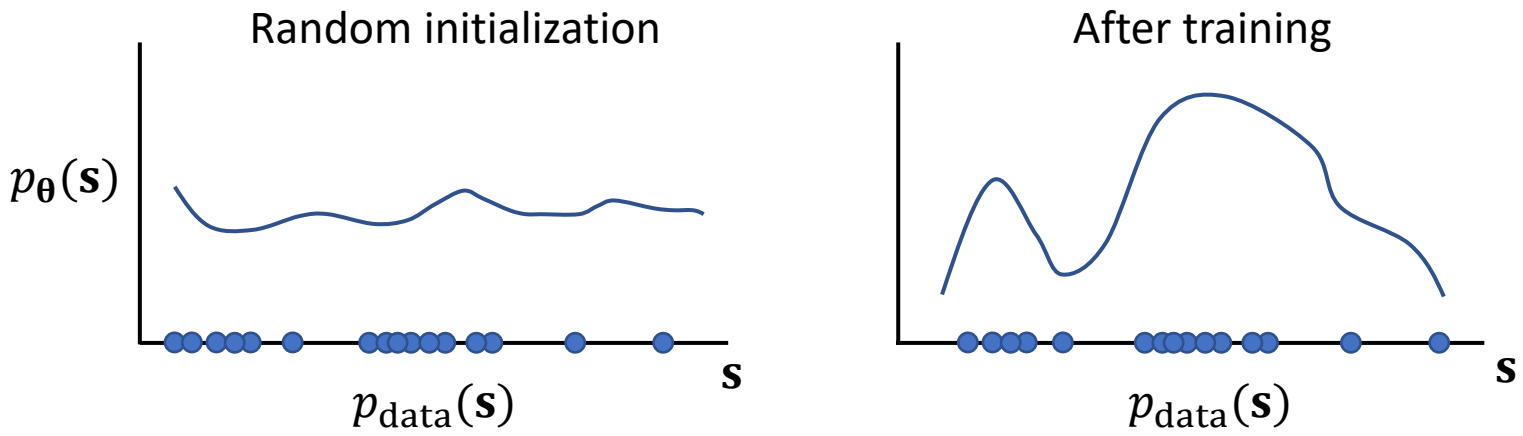
Classical generative modelling with RBMs

- "Energy-based" generative model
- Distinguish "visible" variables s_i from "hidden" variables λ_μ

$$H(\mathbf{s}, \boldsymbol{\lambda}) = -\sum_\mu \sum_i W_{i\mu} s_i \lambda_\mu - \sum_\mu b_\mu(\lambda_\mu)$$

- Classical use: Generative modelling via $p_{\boldsymbol{\theta}}(\mathbf{s}) = Z^{-1} \int e^{-H(\mathbf{s},\boldsymbol{\lambda})} \mathrm{d}\boldsymbol{\lambda}$





Hinton & Salakhutdinov. *Science*. (2006)
Hinton. A practical guide to training restricted boltzmann machines. (2012)

Generative objective

$$\boldsymbol{\theta}^* = \text{argmin}_{\boldsymbol{\theta}} \; D_{\text{KL}}(p_{\text{data}} | p_{\boldsymbol{\theta}})$$

- First note $D_{\text{KL}}(p|q) = \int p(\mathbf{x}) \ln \frac{p(\mathbf{x})}{q(\mathbf{x})} \mathrm{d}\mathbf{x}$, so that

$$\boldsymbol{\theta}^* = \text{argmin}_{\boldsymbol{\theta}} - S(\mathbf{x}) - \int p(\mathbf{x}) \ln q(\mathbf{x}) \, \mathrm{d}\mathbf{x}$$

- So we want $\boldsymbol{\theta}^*$ that *maximizes* the second term (data entropy is constant)
  - In practice, one has finite $M \gg 1$ samples $\{\mathbf{s}_\text{a}\}$ drawn from $p_{\text{data}}(\mathbf{s})$
  - For high dimensional data, usually observe $p_{\text{data}}(\mathbf{s}_\text{a}) = \frac{1}{M}$
- The generative objective we use is

$$\boldsymbol{\theta}^* = \text{argmax}_{\boldsymbol{\theta}} \sum_{a=1}^{M} \ln p_{\boldsymbol{\theta}}(\mathbf{s}_a)$$

Maximizing $L = \sum_{\text{data}} \ln p_{\boldsymbol{\theta}}(\mathbf{s}^{\text{data}})$

Reference:

- $H(\mathbf{s}, \boldsymbol{\lambda}) = \sum_\mu \lambda_\mu^2/2 - \sum_i h_i s_i - \sum_\mu \sum_i W_{i\mu} s_i \lambda_\mu$
- $Z = \int_{\boldsymbol{\lambda}} \sum_{\{\mathbf{s}\}} e^{-H(\mathbf{s}, \boldsymbol{\lambda})} \, d\boldsymbol{\lambda}$
- $p_{\boldsymbol{\theta}}(\mathbf{s}, \boldsymbol{\lambda}) = Z^{-1} e^{-H(\mathbf{s}, \boldsymbol{\lambda})}$

- Note $p_{\boldsymbol{\theta}}(\mathbf{s}) = Z^{-1} \int_{\boldsymbol{\lambda}} e^{-H(\mathbf{s}, \boldsymbol{\lambda})} \, d\boldsymbol{\lambda}$
- Separate $\ln p_{\boldsymbol{\theta}}(\mathbf{s}^{\text{data}}) = \ln \left(\int e^{-H(\mathbf{s}^{\text{data}}, \boldsymbol{\lambda})} \prod_\mu \mathrm{d}\lambda_\mu \right) - \ln Z$
- Define update rule based on gradient with respect to parameters $\boldsymbol{\theta}$:

$$\frac{\partial \ln p_{\boldsymbol{\theta}}(\mathbf{s}^{\text{data}})}{\partial \boldsymbol{\theta}} = \frac{\int \left(-\frac{\partial H}{\partial \boldsymbol{\theta}} \right) e^{-H(\mathbf{s}^{\text{data}}, \boldsymbol{\lambda})} \prod_\mu \mathrm{d}\lambda_\mu}{\int e^{-H(\mathbf{s}^{\text{data}}, \boldsymbol{\lambda})} \prod_\mu \mathrm{d}\lambda_\mu} - \frac{\sum_{\{\mathbf{s}\}} \int \left(-\frac{\partial H}{\partial \boldsymbol{\theta}} \right) e^{-H(\mathbf{s}, \boldsymbol{\lambda})} \prod_\mu \mathrm{d}\lambda_\mu}{Z}$$

- For the interlayer weights, $\frac{\partial H(\mathbf{s}, \boldsymbol{\lambda})}{\partial W_{i\mu}} = -s_i \lambda_\mu$, so

$$\frac{\partial \ln p_{\boldsymbol{\theta}}(\mathbf{s}^{\text{data}})}{\partial W_{i\mu}} = s_i^{\text{data}} \frac{\int \lambda_\mu e^{-H(\mathbf{s}^{\text{data}}, \boldsymbol{\lambda})} \prod_\mu \mathrm{d}\lambda_\mu}{\int e^{-H(\mathbf{s}^{\text{data}}, \boldsymbol{\lambda})} \prod_\mu \mathrm{d}\lambda_\mu} - \frac{\sum_{\{\mathbf{s}\}} \int s_i \lambda_\mu e^{-H(\mathbf{s}, \boldsymbol{\lambda})} \prod_\mu \mathrm{d}\lambda_\mu}{Z}$$

$$= s_i^{\text{data}} \langle \lambda_\mu | \mathbf{s} = \mathbf{s}^{\text{data}} \rangle_{\text{model}} - \langle s_i \lambda_\mu \rangle_{\text{model}}$$

$$\Downarrow \qquad \Downarrow$$

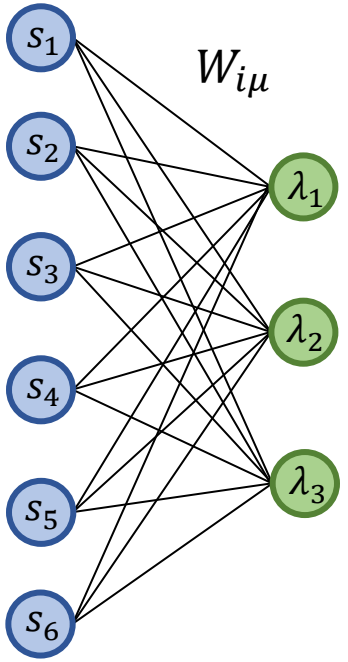
$$s_i^{\text{data}} \sum_j W_{j\mu} s_j^{\text{data}}$$

Second term is intractable. Approach: Monte Carlo simulation of $p_{\boldsymbol{\theta}}$. Exact training very slow!

- Gradient ascent update rule (learning rate η, average first term over training batch B):

$$\eta^{-1}\Delta W_{i\mu} = \langle \Sigma_j W_{j\mu} s_i^a s_j^a \rangle_{a\in B} - \langle s_i \lambda_\mu \rangle_{\text{model}}$$

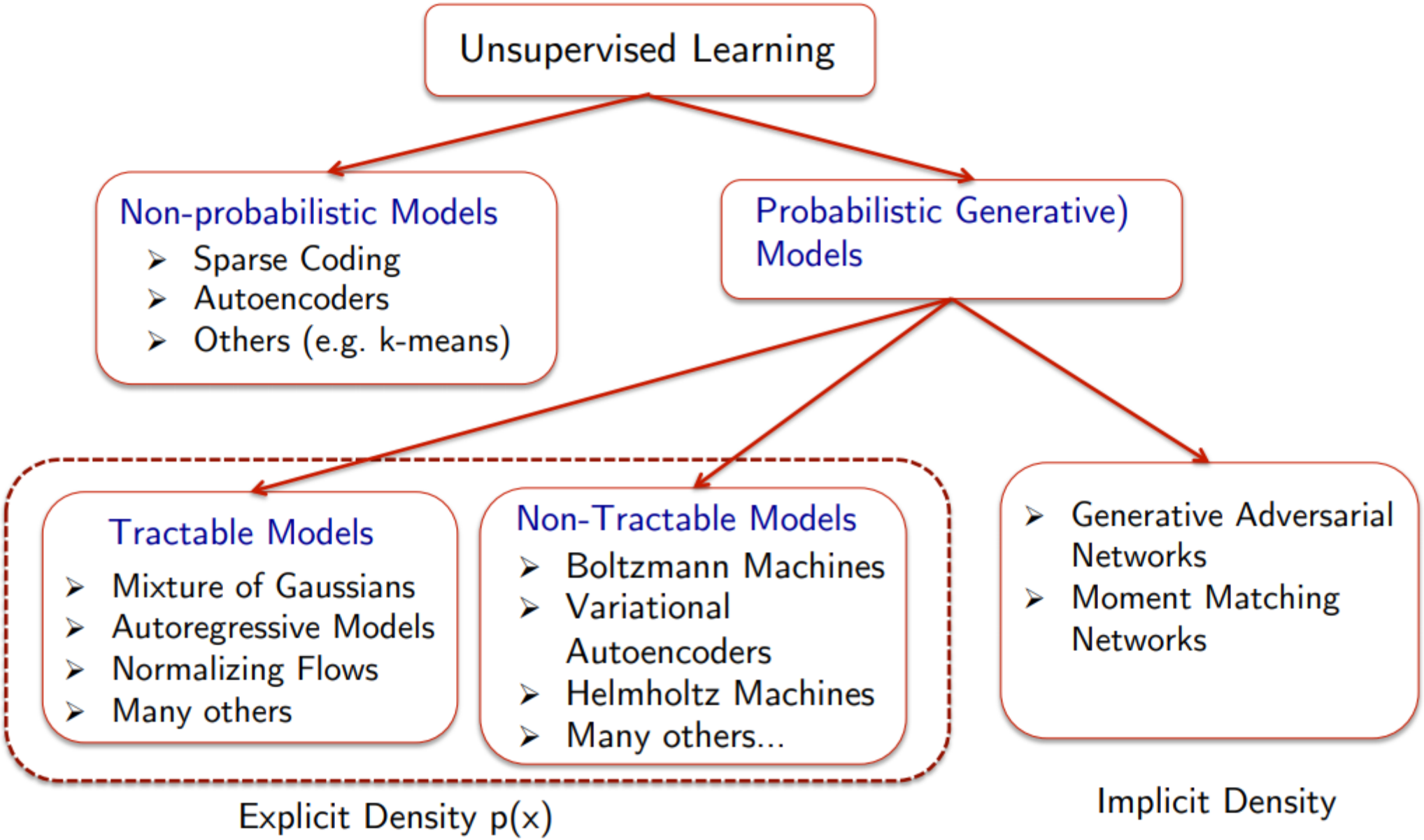
  - can also add regularization (e.g. weight decay) and augmented motion (e.g. momentum)



- Simulation performed via (many) alternating layer updates:
  - $p(s_i = 1|\boldsymbol{\lambda}) = \frac{1}{1+e^{-2x_i}}$ where $x_i \equiv \sum_\mu W_i^\mu \lambda_\mu + h_i$ defines the visible node input
  - $p(\lambda_\mu = \lambda|\mathbf{s}) \sim N(h_\mu, 1)$ where $h_\mu \equiv \sum_i W_i^\mu s_i$ defines the hidden node input

- "Contrastive divergence" CD-k approximation: $\langle s_i \lambda_\mu \rangle_{\text{model}} \approx s_i^{(k)} \lambda_\mu^{(k)}$
  - Terminate after k steps of RBM updates – from which $s_i^{(k)}$, $\lambda_\mu^{(k)}$ are the final state
  - Hinton argues for CD-1 (See e.g. Hinton, 2012, "A Practical Guide to Training RBMs")

Complexity (M samples, N visible, p hidden)	**Per parameter iteration** ($\|B\|$ samples per batch)	**Per epoch** ($M/\|B\|$ loops per epoch)	**Overall estimate** (assume p epochs)
Exact (analytic) training	$> O(2^N)$	N/A	N/A
Hinton's CD-k	$O(kBNp)$	$O(kMNp)$	$O(kMNp^2)$



http://www.cs.toronto.edu/~rsalakhu/ITA_ALT_Part2.pdf

Types of Generative Models

Here is a quick summary of the difference between GAN, VAE, and flow-based generative models:

1. Generative adversarial networks: GAN provides a smart solution to model the data generation, an unsupervised learning problem, as a supervised one. The discriminator model learns to distinguish the real data from the fake samples that are produced by the generator model. Two models are trained as they are playing a minimax game.
2. Variational autoencoders: VAE inexplicitly optimizes the log-likelihood of the data by maximizing the evidence lower bound (ELBO).
3. Flow-based generative models: A flow-based generative model is constructed by a sequence of invertible transformations. Unlike other two, the model explicitly learns the data distribution $p(\mathbf{x})$ and therefore the loss function is simply the negative log-likelihood.

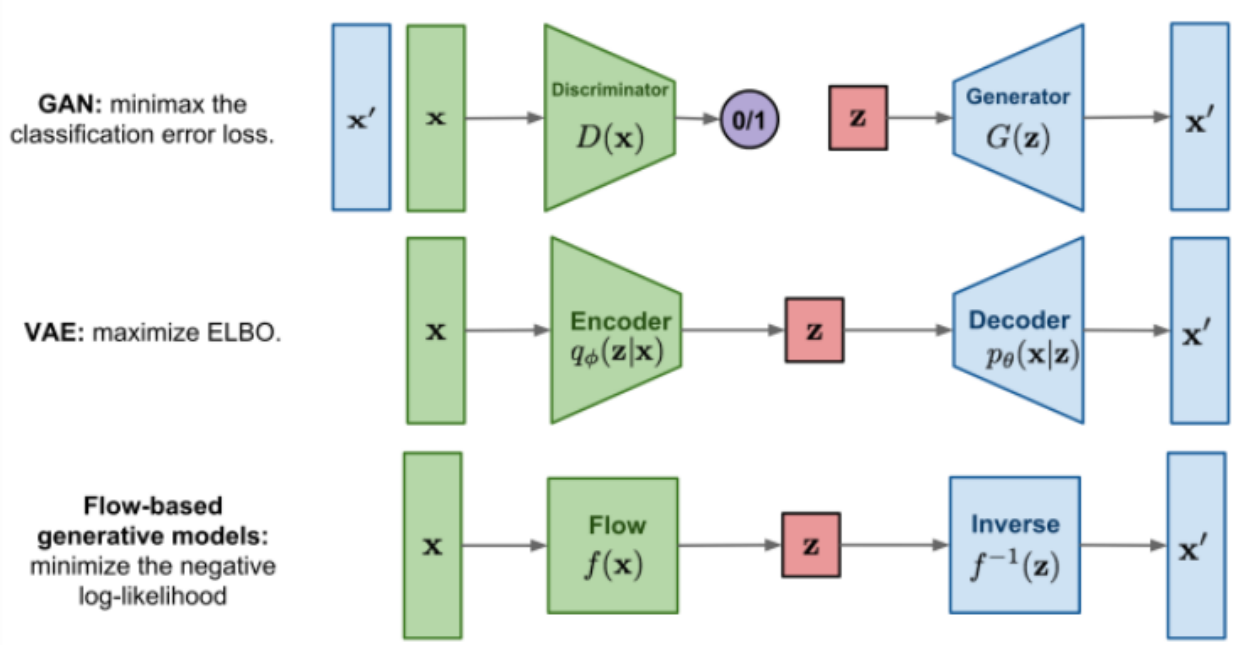
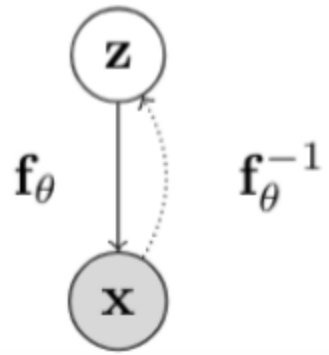


Fig. 1. Comparison of three categories of generative models.

https://lilianweng.github.io/lil-log/2018/10/13/flow-based-deep-generative-models.html

Normalizing Flows

- Directed Latent Variable Invertible models



- The mapping between x and z is deterministic and invertible:

$$\mathbf{x} = \mathbf{f}_\theta(\mathbf{z})$$
$$\mathbf{z} = \mathbf{f}_\theta^{-1}(\mathbf{x})$$

- Use change-of-variables to relate densities between z and x

$$p_X(\mathbf{x};\theta) = p_Z(\mathbf{z}) \left| \det \frac{\partial \mathbf{f}_\theta^{-1}(\mathbf{x})}{\partial X} \right|_{X=\mathbf{x}}$$

Grover and Ermon DGM Tutorial, NICE (Dinh et al. 2014), Real NVP (Dinh et al. 2016)

http://www.cs.toronto.edu/~rsalakhu/ITA_ALT_Part2.pdf

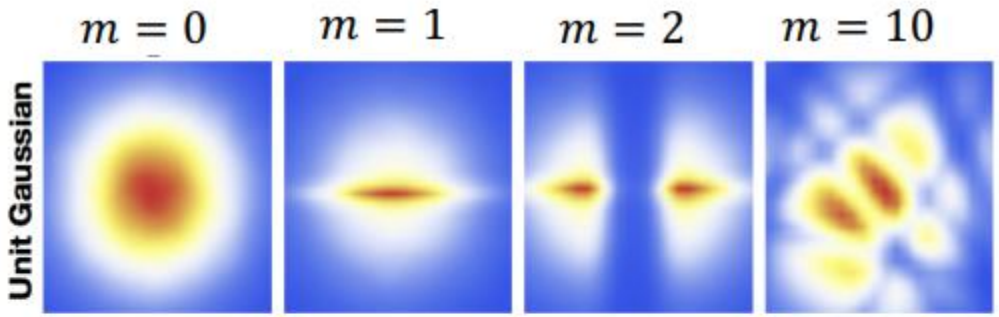
Normalizing Flows

- Invertible transformations can be composed:

$$\mathbf{x} = \mathbf{f}_\theta^M \circ \cdots \circ \mathbf{f}_\theta^1(\mathbf{z}^0); \qquad p_X(\mathbf{x};\theta) = p_{Z^0}(\mathbf{z}^0) \prod_{m=1}^{M} \left| \det \frac{\partial (\mathbf{f}_\theta^m)^{-1}}{\partial Z^m} \right|_{Z^m = \mathbf{z}^m}$$

- Planar Flows

$$f(\mathbf{z}) = \mathbf{z} + \mathbf{u} g(\mathbf{w}^\top \mathbf{z} + b)$$



Rezendre and Mohamed, 2016

Rezendre and Mohamed, 2016, Grover and Ermon DGM Tutorial

http://www.cs.toronto.edu/~rsalakhu/ITA_ALT_Part2.pdf

Normalizing Flows

- Maximum log-likelihood objective

$$\max_{\theta} \log p_X(\mathcal{D};\theta) = \sum_{\mathbf{x}\in\mathcal{D}} \left(\log p_Z(\mathbf{z}) - \log \left| \det \frac{\partial (\mathbf{f}_\theta)^{-1}}{\partial X} \right|_{X=\mathbf{x}} \right)$$

- Exact log-likelihood evaluation via inverse transformations
- Sampling from the model

$$\mathbf{z} \sim p_Z(\mathbf{z}), \quad \mathbf{x} = \mathbf{f}_\theta(\mathbf{z})$$

- Inference over the latent representations:

$$\mathbf{z} \quad = \quad \mathbf{f}_\theta^{-1}(\mathbf{x})$$

Rezendre and Mohamed, 2016, Grover and Ermon DGM Tutorial

http://www.cs.toronto.edu/~rsalakhu/ITA_ALT_Part2.pdf

Normalizing flows

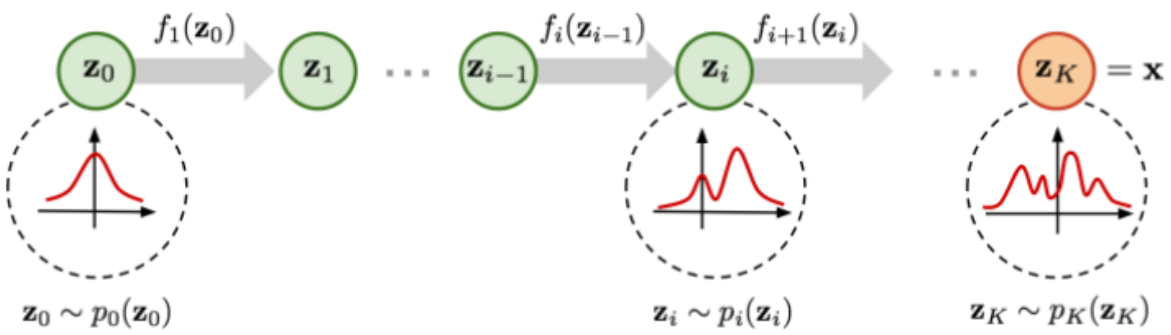


Fig. 2. Illustration of a normalizing flow model, transforming a simple distribution $p_0(\mathbf{z}_0)$ to a complex one $p_K(\mathbf{z}_K)$ step by step.

As defined in Fig. 2,

$$\begin{aligned}\mathbf{z}_{i-1} &\sim p_{i-1}(\mathbf{z}_{i-1}) \\ \mathbf{z}_i &= f_i(\mathbf{z}_{i-1}), \text{ thus } \mathbf{z}_{i-1} = f_i^{-1}(\mathbf{z}_i) \\ p_i(\mathbf{z}_i) &= p_{i-1}(f_i^{-1}(\mathbf{z}_i)) \left|\det \frac{df_i^{-1}}{d\mathbf{z}_i}\right|\end{aligned}$$

Then let's convert the equation to be a function of $\mathbf{z}_i$ so that we can do inference with the base distribution.

$$\begin{aligned}p_i(\mathbf{z}_i) &= p_{i-1}(f_i^{-1}(\mathbf{z}_i)) \left|\det \frac{df_i^{-1}}{d\mathbf{z}_i}\right| \\ &= p_{i-1}(\mathbf{z}_{i-1}) \left|\det \left(\frac{df_i}{d\mathbf{z}_{i-1}}\right)^{-1}\right| && \text{; According to the inverse func theorem.} \\ &= p_{i-1}(\mathbf{z}_{i-1}) \left|\det \frac{df_i}{d\mathbf{z}_{i-1}}\right|^{-1} && \text{; According to a property of Jacobians of invertible func.} \\ \log p_i(\mathbf{z}_i) &= \log p_{i-1}(\mathbf{z}_{i-1}) - \log \left|\det \frac{df_i}{d\mathbf{z}_{i-1}}\right|\end{aligned}$$

Given such a chain of probability density functions, we know the relationship between each pair of consecutive variables. We can expand the equation of the output $\mathbf{x}$ step by step until tracing back to the initial distribution $\mathbf{z}_0$.

$$\begin{aligned}\mathbf{x} = \mathbf{z}_K &= f_K \circ f_{K-1} \circ \cdots \circ f_1(\mathbf{z}_0) \\ \log p(\mathbf{x}) = \log \pi_K(\mathbf{z}_K) &= \log \pi_{K-1}(\mathbf{z}_{K-1}) - \log \left|\det \frac{df_K}{d\mathbf{z}_{K-1}}\right| \\ &= \log \pi_{K-2}(\mathbf{z}_{K-2}) - \log \left|\det \frac{df_{K-1}}{d\mathbf{z}_{K-2}}\right| - \log \left|\det \frac{df_K}{d\mathbf{z}_{K-1}}\right| \\ &= \dots \\ &= \log \pi_0(\mathbf{z}_0) - \sum_{i=1}^{K} \log \left|\det \frac{df_i}{d\mathbf{z}_{i-1}}\right|\end{aligned}$$

The path traversed by the random variables $\mathbf{z}_i = f_i(\mathbf{z}_{i-1})$ is the **flow** and the full chain formed by the successive distributions π_i is called a **normalizing flow**. Required by the computation in the equation, a transformation function f_i should satisfy two properties:

1. It is easily invertible.
2. Its Jacobian determinant is easy to compute.

Models with Normalizing Flows

With normalizing flows in our toolbox, the exact log-likelihood of input data $\log p(\mathbf{x})$ becomes tractable. As a result, the training criterion of flow-based generative model is simply the negative log-likelihood (NLL) over the training dataset $\mathcal{D}$:

$$\mathcal{L}(\mathcal{D}) = -\frac{1}{|\mathcal{D}|} \sum_{\mathbf{x} \in \mathcal{D}} \log p(\mathbf{x})$$

https://lilianweng.github.io/lil-log/2018/10/13/flow-based-deep-generative-models.html

Deep Generative Modelling: A Comparative Review of VAEs, GANs, Normalizing Flows, Energy-Based and Autoregressive Models

Sam Bond-Taylor, Adam Leach, Yang Long, Chris G. Willcocks

Abstract—Deep generative modelling is a class of techniques that train deep neural networks to model the distribution of training samples. Research has fragmented into various interconnected approaches, each of which making trade-offs including run-time, diversity, and architectural restrictions. In particular, this compendium covers energy-based models, variational autoencoders, generative adversarial networks, autoregressive models, normalizing flows, in addition to numerous hybrid approaches. These techniques are drawn under a single cohesive framework, comparing and contrasting to explain the premises behind each, while reviewing current state-of-the-art advances and implementations.

Index Terms—Deep Learning, Generative Models, Energy-Based Models, Variational Autoencoders, Generative Adversarial Networks, Autoregressive Models, Normalizing Flows

1 Introduction

Generative modelling using neural networks has its origins in the 1980s with aims to learn about data models, breaking new ground in terms of visual fidelity and sampling speed. In many cases, this has been achieved

Method	Train Speed	Sample Speed	Param. Effic.	Sample Quality	Relative Divers.	Resolution Scaling	FID	NLL (in BPD)
Generative Adversarial Networks								
DCGAN [162]	★★★★★	★★★★★	★★★★☆	★★★☆☆	★☆☆☆☆	★★☆☆☆	17.70	-
ProGAN [97]	★★☆☆☆	★★★★★	★★★★☆	★★★☆☆	★★☆☆☆	★★★★★	15.52	-
BigGAN [16]	★★★★☆	★★★★★	★★★★☆	★★★★☆	★☆☆☆☆	★★★★☆	14.73	-
StyleGAN2 + ADA [98]	★★★☆☆	★★★★★	★★★★☆	★★★★☆	★★☆☆☆	★★★★★	2.42	-
Energy Based Models								
IGEBM [41]	★★★☆☆	★★☆☆☆	★★★★★	★★★☆☆	★★★☆☆	★★★☆☆	37.9	-
Denoising Diffusion [76]	★★☆☆☆	★★☆☆☆	★★★☆☆	★★★★☆	★★★★☆	★★★★☆	3.17	≤ 3.75
DDPM++ Continuous [182]	★★☆☆☆	★★★☆☆	★★★☆☆	★★★★☆	★★★★☆	★★★★☆	2.92	2.99
Flow Contrastive [50]	★☆☆☆☆	★★★☆☆	★☆☆☆☆	★★☆☆☆	★★★☆☆	★★☆☆☆	37.30	≈ 3.27
VAEBM [215]	★★☆☆☆	★★★☆☆	★★★☆☆	★★★☆☆	★★★☆☆	★★★★☆	12.19	-
Variational Autoencoders								
Convolutional VAE [105]	★★★★★	★★★★★	★★★★☆	★★☆☆☆	★★★★☆	★★☆☆☆	106.37	4.54
Variational Lossy AE [26]	★☆☆☆☆	★☆☆☆☆	★☆☆☆☆	★★☆☆☆	★★★★☆	★☆☆☆☆	-	2.95
VQ-VAE [164], [205]	★★☆☆☆	★★★☆☆	★★☆☆☆	★★★☆☆	★★★★☆	★★★★★	-	4.67
VD-VAE [28]	★★★☆☆	★★★★☆	★★★☆☆	★★★☆☆	★★★★☆	★★★★★	-	≤ 2.87
Autoregressive Models								
PixelRNN [204]	★☆☆☆☆	★☆☆☆☆	★☆☆☆☆	★☆☆☆☆	★★★★★	★★★☆☆	-	3.00
Gated PixelCNN [203]	★☆☆☆☆	★☆☆☆☆	★☆☆☆☆	★★☆☆☆	★★★★★	★★★☆☆	65.93	3.03
PixelIQN [154]	★☆☆☆☆	★☆☆☆☆	★☆☆☆☆	★★☆☆☆	★★★★★	★★★☆☆	49.46	-
Sparse Trans. + DistAug [29], [94]	★★☆☆☆	★☆☆☆☆	★★☆☆☆	★★★★☆	★★★★★	★★★☆☆	14.74	2.66
Normalizing Flows								
RealNVP [38]	★☆☆☆☆	★★★★☆	★☆☆☆☆	★☆☆☆☆	★★★★★	★★★☆☆	-	3.49
Masked Autoregressive Flow [158]	★★☆☆☆	★☆☆☆☆	★★☆☆☆	★☆☆☆☆	★★★☆☆	★★☆☆☆	-	4.30
GLOW [106]	★☆☆☆☆	★★★★☆	★☆☆☆☆	★★☆☆☆	★★★★★	★★★★☆	45.99	3.35
FFJORD [55]	★☆☆☆☆	★★★☆☆	★★★★★	★★☆☆☆	★★★★★	★★☆☆☆	-	3.40
Residual Flow [23]	★★☆☆☆	★★★★☆	★★★★☆	★★☆☆☆	★★★★★	★★★★☆	46.37	3.28

TABLE 1: Comparison between deep generative models in terms of training and test speed, parameter efficiency, sample quality, sample diversity, and ability to scale to high resolution data. Quantitative evaluation is reported on the CIFAR-10 dataset [109] in terms of Fréchet Inception Distance (FID) and negative log-likelihood (NLL) in bits-per-dimension (BPD).

SOTA text-to-image generation with transformers*

- OpenAI blog post: https://openai.com/blog/dall-e/
- Associated paper: https://arxiv.org/abs/2102.12092

* and a bunch of tricks outlined on their blog

TEXT PROMPT

an armchair in the shape of an avocado [...]

AI-GENERATED IMAGES

Edit prompt or view more images↓

TEXT PROMPT

a store front that has the word ‘openai’ written on it [...]

AI-GENERATED IMAGES

Edit prompt or view more images↓

TEXT AND IMAGE PROMPT

the exact same cat on the top as a sketch on the bottom

AI-GENERATED IMAGES

Edit prompt or view more images↓

References

Review

Deep Generative Modelling: A Comparative Review of VAEs, GANs, Normalizing Flows, Energy-Based and Autoregressive Models
https://arxiv.org/abs/2103.04922
2021, Durham University

Learning Deep Generative Models
https://www.cs.cmu.edu/~rsalakhu/papers/annrev.pdf
2015, Salakhutdinov

Other resources (presentations)

http://introtodeeplearning.com/slides/6S191_MIT_DeepLearning_L4.pdf

http://www.cs.toronto.edu/~rsalakhu/ITA_ALT_Part2.pdf

Specific topics

On Unifying Deep Generative Models
https://openreview.net/forum?id=rylSzl-R-
2018, Salakhutdinov++

A note on the evaluation of generative models
https://arxiv.org/pdf/1511.01844.pdf
2015, Bethge++

Variational Inference with Normalizing Flows
https://arxiv.org/abs/1505.05770
2015, Rezende, Mohamed