An introduction to Score-based Generative Models
Lecture 1: Introduction to generative models

Giovanni Conforti, Alain Durmus
École polytechnique
with the help of Valentin De Bortoli, Marta Gentiloni-Silveri, Emmanuel Gobet, Yazid Janati, Éric Moulines, Maxence Noble, Tom Sander, and many others...

February 19, 2024

Outline of the course

- Introduction / Motivations
- Minimum distance estimation (MDE)
- Normalizing flows
- Energy based models – Maximum entropy methods

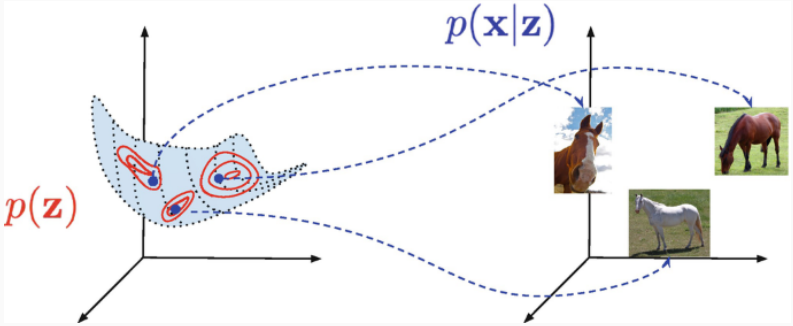
Introduction / Motivation

Let $\mathsf{X} \subset \mathbb{R}^p$ endowed with $\mathcal{X} = \mathsf{X} \cap \mathcal{B}(\mathbb{R}^d)$.

- **Input**
  Data $\{x^i\}_{i=1}^N$: N i.i.d. observations from an unknown $\mu^\star \in \mathcal{P}(\mathsf{X})$.
  Notation: $\mathcal{P}(\mathsf{X})$ space of probability measures on $(\mathsf{X}, \mathcal{X})$.
- **Output**
  New samples from $\mu^\star$

- Consider $\{\mu_\theta : \theta \in \Theta\}$, $\Theta \subset \mathbb{R}^d$.
- We must be able to sample from μ_θ.
- Goal: learn θ which fits the data $\{x^i\}_{i=1}^N$.
- Two approaches: stochastic/statistical approach and transformation approach.

- **Input**
  Data $\{x^i\}_{i=1}^N : N$
  i.i.d. observations.
- New samples from $\mu^\star$
- Consider $\{\mu_\theta : \theta \in \Theta\}$, $\Theta \subset \mathbb{R}^d$.
- We must be able to sample from μ_θ.
- Goal: learn θ which fits the data $\{x^i\}_{i=1}^N$.

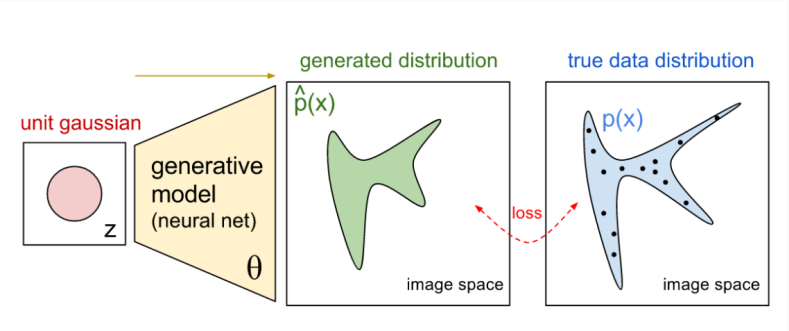


Credit: Tomczak (2022)

- **The stochastic/statistical approach: construct μ_θ which has an explicit density p_θ with respect to a dominating measure λ.**
- p_θ not necessarily tractable!
- In general $\lambda = \mathrm{Leb}$.

- **Input**
  Data $\{x^i\}_{i=1}^N$: N
  i.i.d. observations.
- **Output**
  New samples from $\mu^\star$

- Consider $\{\mu_\theta : \theta \in \Theta\}$, $\Theta \subset \mathbb{R}^d$.
- We must be able to sample from μ_θ.
- Goal: learn θ which fits the data $\{x^i\}_{i=1}^N$.



Credit: https://openai.com/blog/generative-models/

- **The transformation/flows approach:** $\mu_\theta = \mathrm{T}_\theta \sharp \nu$;
  1. for a family of transformations $\{\mathrm{T}_\theta : \theta \in \Theta\}$;
  2. ν is a reference measure we should be able to sample (samplable).
- **Here μ_θ may not admit an explicit density.**

Let $\mathsf{X} \subset \mathbb{R}^p$ endowed with $\mathcal{X} = \mathsf{X} \cap \mathcal{B}(\mathbb{R}^d)$.

- **Input**
  Data $\{x^i\}_{i=1}^N$: N i.i.d. observations from an unknown $\mu^\star \in \mathcal{P}(\mathsf{X})$.
- **Output**
  New samples from $\mu^\star$

- **Consider $\{\mu_\theta : \theta \in \Theta\}$, $\Theta \subset \mathbb{R}^d$.**
- We must be able to sample from μ_θ.
- Goal: **learn θ which fits the data $\{x^i\}_{i=1}^N$.**
- Question: **how to fit the data once $\{\mu_\theta : \theta \in \Theta\}$, $\Theta \subset \mathbb{R}^d$ chosen?**

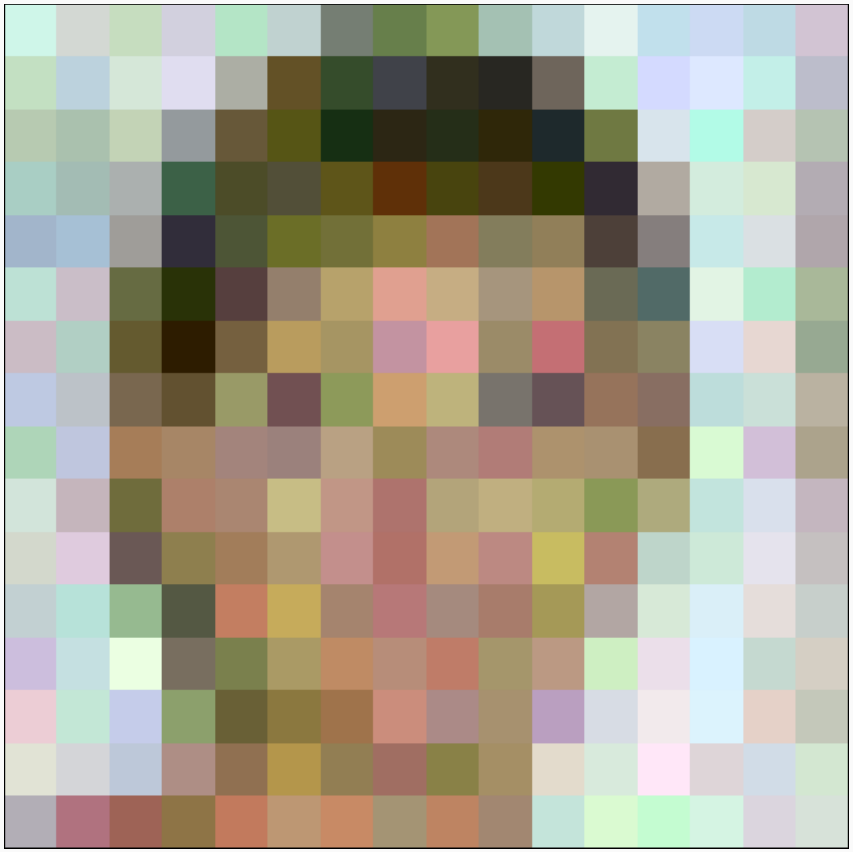
A first useless application

https://thesecatsdonotexist.com/

Foundation models, deep-learning priors I

- We consider a linear inverse problem:

$$y = \mathbf{A}x + G \ .$$



Foundation models, deep-learning priors II

- We consider a linear inverse problem:

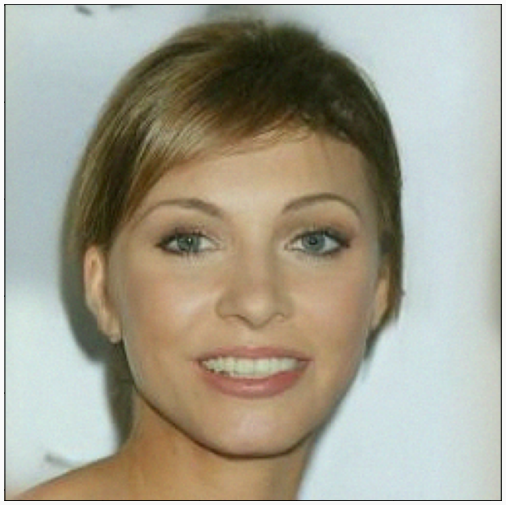
$$y = \mathbf{A}x + G \ .$$

- One "simple" option:

$$\textbf{minimize } x \mapsto \|\mathbf{A}x - y\|^2 - \log p^\star(x) \ ,$$

  - the term $\|\mathbf{A}x - y\|^2$ enforces the constraints associated with the observation y;
  - the second term: $p^\star(x)$ should be able quantify how "plausible" x is.

Foundation models, deep-learning priors III



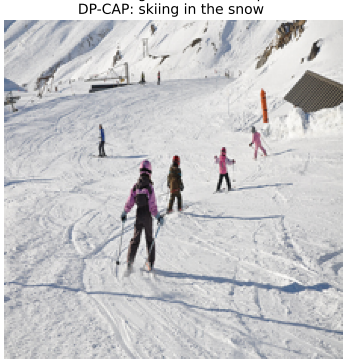
Conditional generative models

- In many applications, we are not only interested in sampling from a single distribution $\mu^\star$...
- Goal: sample from conditional distributions $\{\mu^\star(\cdot|y) : y \in \mathsf{Y}\}$
- Simple/complex adaptations of non-conditioned generative models
- Not discussed in this course...

Text-to-image generation

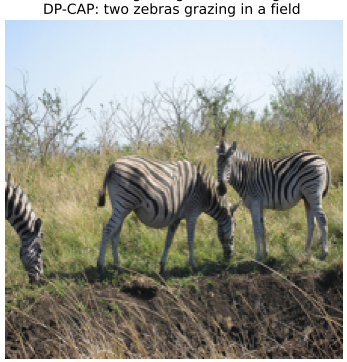
Captioning

True: a picture of skiers skiing on the snow
CAP: skiing in the french alps
DP-CAP: skiing in the snow



True: a dog wearing a scarf and shirt on a leash
CAP: a dog dressed as a cat and a cat
DP-CAP: a dog is sitting on a leash

True: three zebras grazing in a grassy area near shrubs
CAP: zebras grazing in the serengeti
DP-CAP: two zebras grazing in a field



True: blue and white bird standing on a branch
CAP: osprey perched on a dead branch
DP-CAP: bald eagle perched on a branch

True: a person walking along the beach with a surfboard
CAP: surfer walking on the beach
DP-CAP: a man walking on the beach with a dog

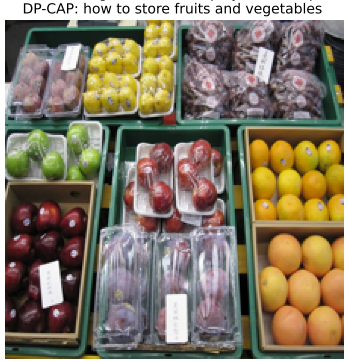
True: a herd of cows lay down on some grass
CAP: cows resting in the grass
DP-CAP: a herd of cows grazing in a field

True: a cat sitting on a chair looking down.
CAP: cat playing in the leaves
DP-CAP: a white cat sitting on a bench

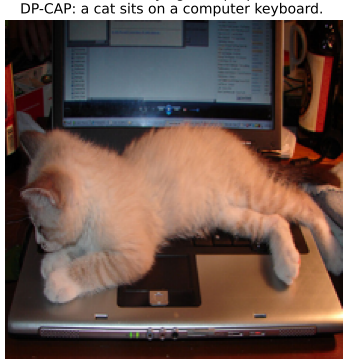
True: a cat wearing a hat on its head
CAP: cat wearing a hat
DP-CAP: cat wearing a pink hat

True: a woman is holding a slice of red and white cake
CAP: woman holding a plate of cake
DP-CAP: a woman holding a plate of red velvet cake

True: a bunch of fruits and vegetables for sale on disp
CAP: fruit and vegetables are displayed at a supermark
DP-CAP: how to store fruits and vegetables



True: a kitty cat lies down on a computer keyboard.
CAP: cat lying on laptop
DP-CAP: a cat sits on a computer keyboard.



True: a herd of sheep grazing in a field.
CAP: sheep grazing in a field
DP-CAP: sheep grazing in the meadow

Minimum Distance Estimation

Minimum Distance Estimation Wolfowitz (1957)

- **Input** For $\mathsf{X} \subset \mathbb{R}^d$.
  Data $\{x_i\}_{i=1}^N$: N i.i.d. observations from $\mu^\star \in \mathcal{P}(\mathsf{X})$ unknown
- **Output**
  New samples from $\mu^\star$

- **Consider a parametric family of distributions $\{\mu_\theta : \theta \in \Theta\}$.**
- Minimum distance estimation :
  - minimize $\theta \mapsto \mathbf{D}(\mu_\theta | \mu^\star)$ where $\mathbf{D}$ is a divergence/metric over the space of probability measure on X.
  - Sample a new observation from $\mu_{\theta^\star}$.

Metric/distance on $\mathcal{P}(X)$

Divergence over $\mathcal{P}(\mathsf{X})$

- A divergence on $\mathcal{P}(\mathsf{X})$, is a function $\mathbf{D} : \mathcal{P}(\mathsf{X})^2 \to \mathbb{R}_+$ which satisfies *the most important*[1] axiom of a distance: for μ, ν two probability measures

$$\mathbf{D}(\mu|\nu) = 0 \text{ if and only if } \mu = \nu \quad .$$

- Any distance on $\mathcal{P}(\mathsf{X})$ is a divergence.
- Do not satisfy the other axioms of a distance
- Important example: **the Kullback Leibler and the Fisher information.**

[1] from the ML viewpoint...

Minimum Distance Estimation: an ideal?

- **Input** For $\mathsf{X} \subset \mathbb{R}^d$.
  Data $\{x_i\}_{i=1}^N$: N i.i.d. observations from $\mu^\star \in \mathcal{P}(\mathsf{X})$ unknown
- **Output**
  New samples from $\mu^\star$

- Consider a parametric family of distributions $\{\mu_\theta : \theta \in \Theta\}$.
- Minimum distance estimation :
  - minimize $\theta \mapsto \mathbf{D}(\mu_\theta | \mu^\star)$ or $\mathbf{D}(\mu_\theta | \hat{\mu}_N)$ where $\mathbf{D}$ is a divergence over the space of probability measure on X.
  - Sample a new observation from $\mu_{\theta^\star}$.

MDE: challenges

- Minimum distance estimation :
  - minimize $\theta \mapsto \mathbf{D}(\mu_\theta | \mu^\star)$ or $\mathbf{D}(\mu_\theta | \hat{\mu}_N)$ where $\mathbf{D}$ is a divergence over the space of probability measure on X.
  - Sample a new observation from $\mu_{\theta^\star}$.
- **Some problems appear**:
  - Choice
    - for the family $\{\mu_\theta \ : \ \theta \in \Theta\}$?
    - for the divergence $\mathbf{D}$?
  - Minimization for $\theta \mapsto \mathbf{D}(\mu_\theta | \mu^\star)$?

MDE: challenges

- In the infinite case $\mathrm{Card}(\mathsf{X}) = \infty$, we can still use $\hat{\mu}_N$.
- **From a statistical viewpoint, it is hard to obtain a better estimator than the empirical measure**: for some class of measure M_0 and divergence $\mathbf{D}_0$, we have there exists $\alpha > 0$, $C_1, C_2 > 0$, for any $n \in \mathbb{N}^*$,

$$C_1 n^{-\alpha} \leq \sup_{\mu \in \mathsf{M}_0} \inf_{\hat{\nu}_N : \mathsf{X}^N \to \mathsf{M}_0} \mathbf{D}_0(\hat{\nu}_N, \mu) \,, \quad \mathbf{D}_0(\hat{\mu}_N, \mu) \leq C_2 n^{-\alpha} \,.$$

- However, sample from $\hat{\mu}_N$ are just training samples: you want to avoid such degenerate situations and introduce some diversity.
- The notion of diversity is still not well-defined and hard to evaluate empirically...

Maximum likelihood estimation

Maximum likelihood estimation

- Consider the case $\Theta \subset \mathbb{R}^d$.
- Choice for the family $\{\mu_\theta : \theta \in \Theta\}$?

$$\{\mu_\theta : \mu_\theta \ll \lambda\,, \quad p_\theta = \mathrm{d}\mu_\theta/\mathrm{d}\lambda\}\ .$$

Example:

$$p_\theta : \text{ density w.r.t. Leb of } \mathsf{N}(\mathtt{m}, \sigma^2)\ .$$

- We should be able to sample from p_θ for any θ...
- We leave explicit choice of $\{p_\theta : \theta \in \Theta\}$ for the next section...
- **Choice for the divergence D?**

Maximum likelihood estimation

- Consider the case $\Theta \subset \mathbb{R}^d$.
- Choice for the family $\{\mu_\theta : \theta \in \Theta\}$?

$$\{\mu_\theta : \mu_\theta \ll \lambda\,, \quad p_\theta = \mathrm{d}\mu_\theta/\mathrm{d}\lambda\}\ .$$

- Choice for the divergence **D**?
- **D** = KL: problem $\mathrm{KL}\left(\mu_\theta \,\|\, \hat{\mu}_N\right) = \infty$...
- Recall that ideally if **D** = KL, we would like to minimize

$$\mathrm{KL}\left(\mu^\star \,\|\, \mu_\theta\right) = -\int \mathrm{d}\mu^\star \log\left(\frac{\mathrm{d}\mu_\theta}{\mathrm{d}\mu^\star}\right)\ .$$

- This is equivalent to maximize if $\mu^\star \ll \lambda$,

$$\theta \mapsto \int \mathrm{d}\mu^\star \log\left(\frac{\mathrm{d}\mu_\theta}{\mathrm{d}\lambda}\right)\ .$$

- Solution: replace the integral by an empirical version

$$\theta \mapsto N^{-1} \sum_{i=1}^{N} \log p_\theta(x_i)\ .$$

Density estimation

- How to optimize?

$$\theta \mapsto \int \mathrm{d}\mu^\star \log\left(\frac{\mathrm{d}\mu_\theta}{\mathrm{dLeb}}\right) \text{ or } \theta \mapsto \sum_{i=1}^{N} \log p_\theta(x_i)\ . \tag{1}$$

- Stochastic gradient descent:

$$\theta_{k+1} = \theta_k + \gamma_{k+1} \sum_{x_i \in \mathsf{B}_{k+1}} \nabla_\theta [\log p_{(\cdot)}(x_i)](\theta_k)\ ,$$

where

$(\mathsf{B}_k)_k$ **is a sequence of random batch of data points** ,

$(\gamma_k)_k$ **is a sequence of stepsizes/learning rates** .

- Under some assumptions, it can be shown that almost surely $(\theta_k)_{k\in\mathbb{N}}$ converges to some minimizers of (1).

Density estimation

- Choice for the family $\{\mu_\theta : \theta \in \Theta\}$?
  $\{\mu_\theta : \mu_\theta \ll \mathrm{Leb}\,, \quad p_\theta = \mathrm{d}\mu_\theta/\mathrm{dLeb}\}$.
- First solution:

$$\mu_\theta = (\mathrm{T}_\theta)_\sharp \nu_0\,,$$

  where

$$\nu_0 \in \mathcal{P}(\mathbb{R}^p) \text{ with density } q_0\,, \qquad \mathrm{T}_\theta : \mathbb{R}^p \to \mathbb{R}^p$$

- What people thought it was hard:

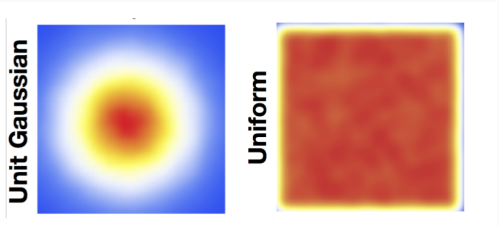
$$\text{find } \{\mathrm{T}_\theta : \theta \in \Theta\} \text{ such that } \theta \mapsto \sum_{i=1}^{N} \log p_\theta(x_i) \text{ easy to optimize...}$$

- It turns out that such constructions are now possible using neural networks, normalizing flows Rezende and Mohamed (2015)!

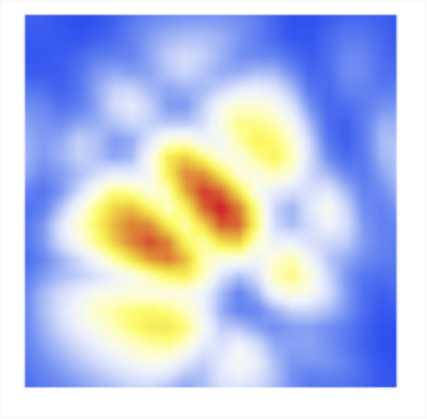
Normalizing flows

Simple Prior to Complex Data Distributions

- Desirable properties of any model distribution p_θ:
  - Easy-to-evaluate, closed form density (useful for training)
  - Easy-to-sample (useful for generation)
- Many simple distributions satisfy the above properties e.g., Gaussian, uniform distributions



- Unfortunately, **data distributions are more complex** (multi-modal)



- **Key idea** behind flow models: Map simple distributions (easy to sample and evaluate densities) to complex distributions through an invertible transformation.

Normalizing flow models

- In a normalizing flow model, the mapping between Z and X, given by $\mathrm{T}_\theta : \mathbb{R}^p \mapsto \mathbb{R}^p$, **is deterministic and invertible** such that $X = \mathrm{T}_\theta(Z)$ and $Z = \mathrm{T}_\theta^{\leftarrow}(X)$.
- Using the d-dimensional **change of variable** we have for any $f \in \mathrm{C}_\mathrm{c}(\mathbb{R}^d, \mathbb{R})$

$$\mathbb{E}_\theta\left[f(X)\right] = \text{?}$$

- The marginal likelihood $p_\theta(x)$ is given by

$$p_\theta(x) = \text{?}$$

- Note: we assume that T_θ is a **diffeomorphism** (not necessary, one can use the co-area/area formula **?**)

A Flow of transformations

- Normalizing: change of variables gives a normalized density after applying an invertible transformation.
- Flow: invertible transformations can be composed with each other

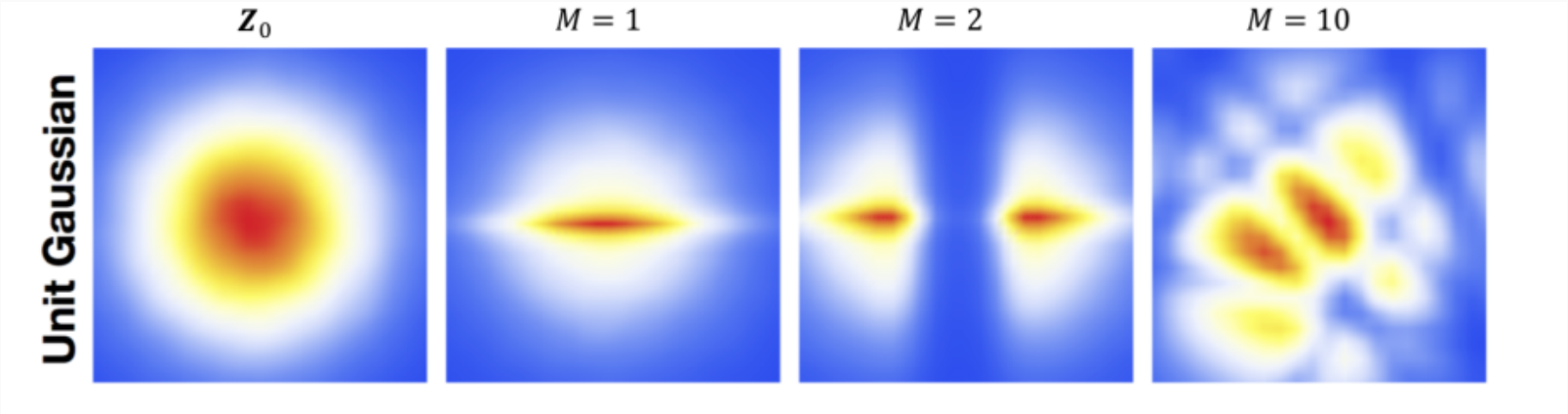
$$\mathrm{T}_\theta^{1:m}(z_0) = \mathrm{T}_\theta^m \circ \cdots \circ \mathrm{T}_\theta^1(z_0) = \mathrm{T}_\theta^m\left(\mathrm{T}_\theta^{m-1}\left(\cdots\left(\mathrm{T}_\theta^1(z_0)\right)\right)\right) \ .$$

- Start with a simple distribution for z_0 (e.g., Gaussian).
- Apply a sequence of M invertible transformations

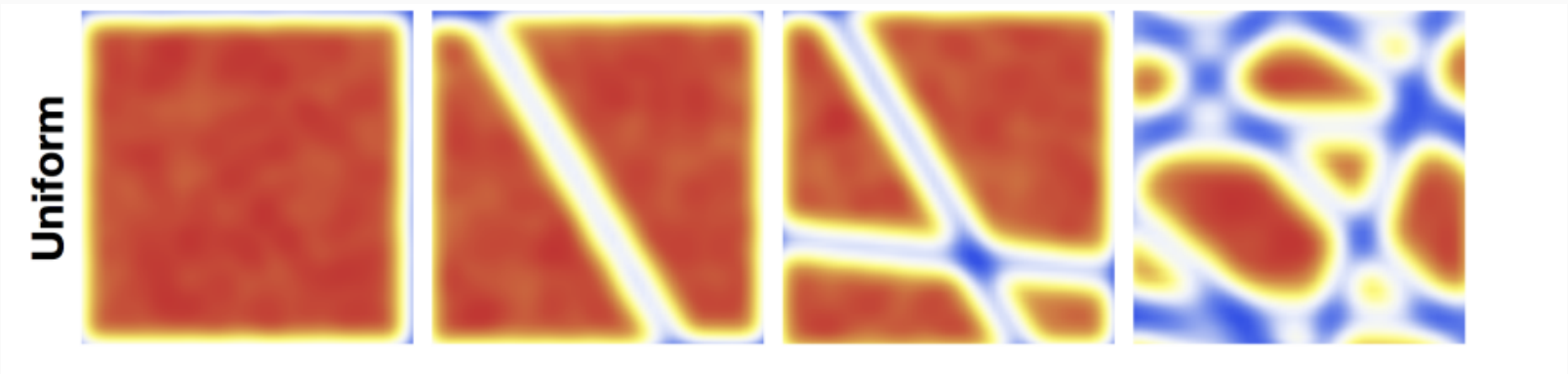
$$p_\theta(x) = \ ?$$

Planar flows Rezende and Mohamed (2015)

- Base distribution: Gaussian



- Base distribution: Uniform



- 10 planar transformations can transform simple distributions into a more complex one

Learning and inference

- Learning via maximum likelihood over the dataset $\{x^i\}$
- Hence, maximizing the **log-likelihood** is equivalent to maximizing

$$\boxed{\ell(\theta) = N^{-1} \sum_{i=1}^{N} \log(p(\mathrm{T}_\theta^{\leftarrow}(x^i))) + \log(|\det \mathbf{Jac}_\theta(\mathrm{T}_\theta^{\leftarrow}(x^i))|) \; .}$$

- **Exact likelihood evaluation via inverse tranformation** $x \mapsto \mathrm{T}_\theta^{\leftarrow}(x)$ and change of variables formula
- **Sampling via forward transformation** $z \mapsto \mathrm{T}_\theta(z)$:

$$Z \sim \nu_0 \; , \quad X = \mathrm{T}_\theta(Z) \; .$$

- Latent representations inferred via inverse (normalizing) transformation (no inference network required!)

$$z = \mathrm{T}_\theta^{\leftarrow}(x) \; .$$

Desiderata for flow models

- Simple prior ν_0 (with density q_0) that allows for efficient sampling and tractable likelihood evaluation.
- E.g., isotropic Gaussian
- **Invertible transformations with tractable evaluation**
- Likelihood evaluation requires efficient evaluation of $\mathrm{T}_\theta^{\leftarrow}$
- Sampling requires efficient evaluation of T_θ

Desiderata for flow models

- Computing likelihoods also requires the **evaluation of determinants of $p \times p$ Jacobian matrices**, where p is the data dimensionality
- Computing the determinant for an $p \times p$ matrix is $\mathcal{O}\left(p^3\right)$: prohibitively expensive within a learning loop!
- Key idea: Choose tranformations so that the resulting Jacobian matrix has special structure.
- For example, the determinant of a triangular matrix is the product of the diagonal entries, i.e., an $\mathcal{O}(p)$ operation.

Planar flows Rezende and Mohamed (2015)

- Invertible transformation

$$\boxed{\mathrm{T}_\theta(z) = z + uh\left(w^{\mathrm{T}} z + b\right)}$$

  parameterized by $\theta = (w, u, b)$ where h is a non-linearity.

- Absolute value of the determinant of the Jacobian is given by

$$|\det \mathbf{Jac}_\theta \mathrm{T}_\theta(z)| = \text{?}$$

- Need to restrict parameters and non-linearity for the mapping to be invertible. For example, $h = \tanh$ and $h'\left(w^{\mathrm{T}} z + b\right) u^{\mathrm{T}} w \geq -1$.
- No closed expression for the inverse.

Designing invertible transformations

- NICE or Nonlinear Independent Components Estimation (Dinh et al., 2014) composes two kinds of invertible transformations: additive coupling layers and rescaling layers
- Real-NVP (Dinh et al., 2017)
- Inverse Autoregressive Flow (Kingma et al., 2016) - Masked Autoregressive Flow (Papamakarios et al., 2017)
- I-resnet (Behrmann et al, 2018)
- Glow (Kingma et al, 2018)
- MintNet (Song et al., 2019)
- And many more...

NICE and Real-NVP

NICE Nonlinear Independent Components Estimation ?

- Partition the variables z into two disjoint subsets, say $z_{1:r}$ and $z_{r+1:p}$ for any $1 \leq r < p$.
- Forward mapping T_θ :

$$\mathrm{T}_\theta(z)_{1:r} = z_{1:r} \ , \quad \mathrm{T}_\theta(z)_{r+1:p} = z_{r+1:p} + \mathtt{m}_\theta\left(z_{1:r}\right) \ ,$$

  $\mathtt{m}_\theta$ **is a neural network** with parameters θ and r input units, and $p - r$ output units.

- Inverse mapping ?
- Jacobian of forward mapping: ?

NICE Nonlinear Independent Components Estimation ?

- Additive coupling layers are composed together (with arbitrary partitions of variables in each layer).
- Final layer of NICE applies a rescaling transformation (linear diagonal flow).
- Forward mapping T_θ :

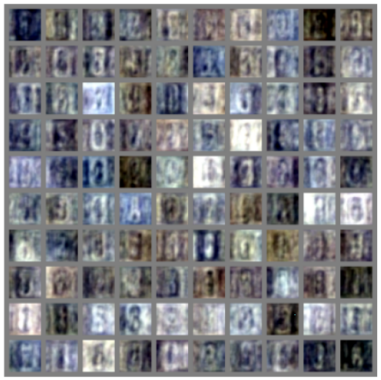
$$\mathrm{T}_\theta(z)_i = s_i z_i \ .$$

  where $s_i > 0$ is the scaling factor for the i-th dimension.
- $\theta = \{s_i\}_{i=1}^p$ here
- Inverse mapping ?
- Jacobian of forward mapping: ?

Samples generated by NICE

(a) Model trained on MNIST

(b) Model trained on TFD



(c) Model trained on SVHN

(d) Model trained on CIFAR-10

- Limitation: too much volume preserving...

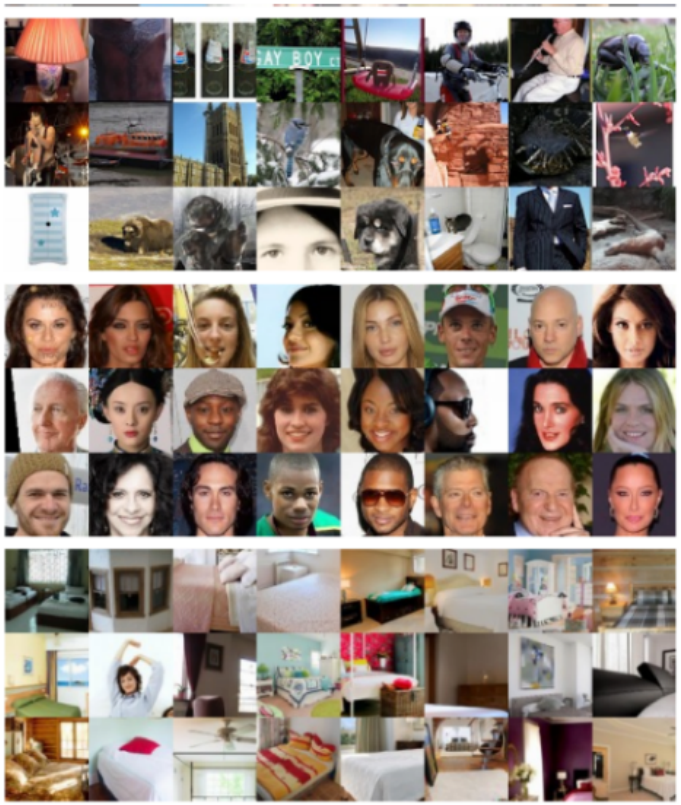
Real-NVP: Non-volume preserving extension of NICE ?

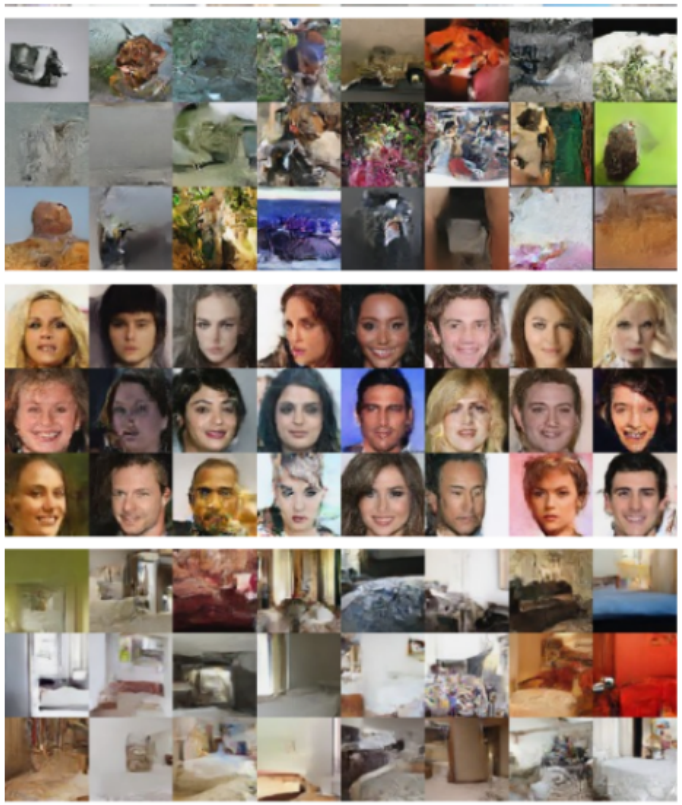
- Forward mapping T_θ:

$$\mathrm{T}_\theta(z)_{1:r} = z_{1:r}\ , \quad \mathrm{T}_\theta(z)_{r+1:p} = z_{r+1:p} \odot \exp\left(\alpha_\theta\left(z_{1:r}\right)\right) + \mathtt{m}_\theta\left(z_{1:r}\right)$$

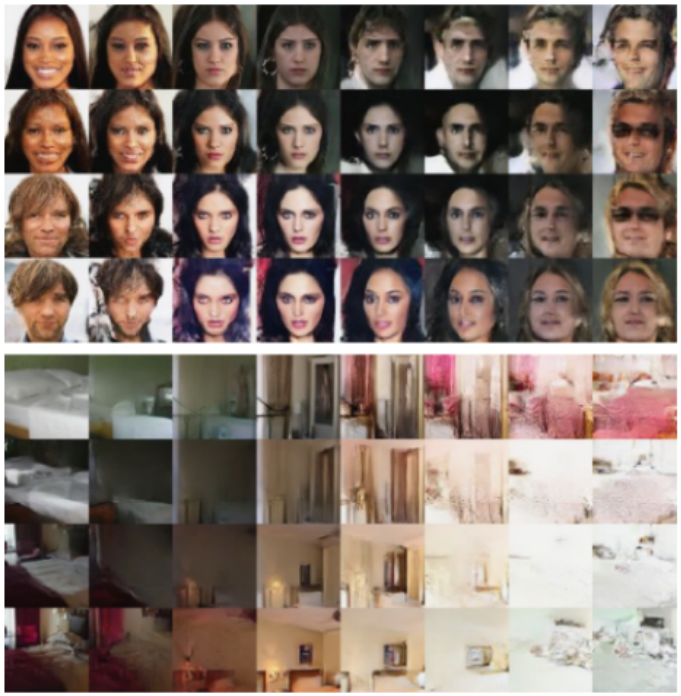
- $\mathtt{m}_\theta$ **and** α_θ **are both neural networks** with parameters θ, r input units, and $p - r$ output units ($\odot$ denotes elementwise product)
- Inverse mapping ?
- Jacobian of forward mapping ?
- Non-volume preserving transformation in general since determinant can be less than or greater than 1.

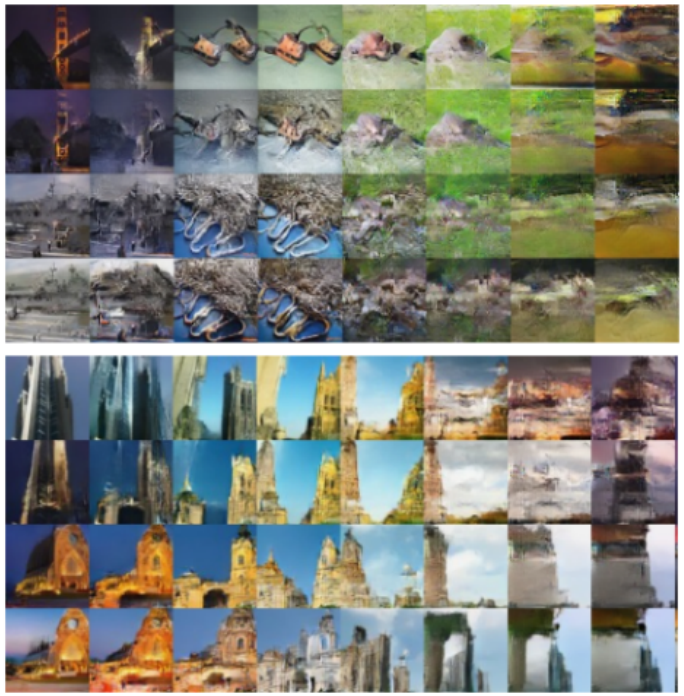
Samples from Real-NVP





Latent space interpolations via Real-NVP





Using with four validation examples $z^{(1)}, z^{(2)}, z^{(3)}, z^{(4)}$, define interpolated z as:

$$z = \cos\phi \left(z^{(1)} \cos\phi' + z^{(2)} \sin\phi' \right) + \sin\phi \left(z^{(3)} \cos\phi' + z^{(4)} \sin\phi' \right)$$

parameterized by ϕ and ϕ'.

A detour by autoregressive models

- Another **generative modeling** approach: **autoregressive models**
  - **M**asked **A**utoencoder for **D**istribution **E**stimation (autoregressive autoencoder), **?**.
  - **PixelRNN** (autoregressive LSTM), **?**.
  - Both models are trained by maximizing the **log-likelihood**.

- Both models assume the following **raster-scan** decomposition.

$$p_\theta(x) = \prod_{i=1}^{p} p_\theta(x_i | x_{1:i-1}) \ .$$

- **Problems**:
  - As many predicitions as the **dimension**.
  - Can be parallelized for **training but not for sampling**.

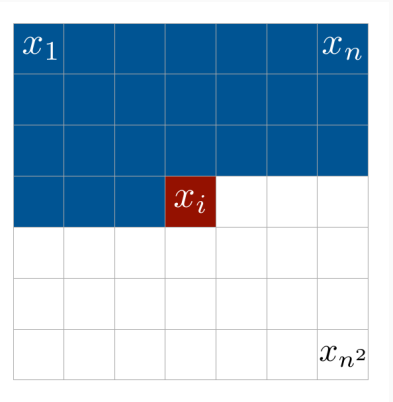


Figure 1: Raster scan order. Image extracted from **?**.

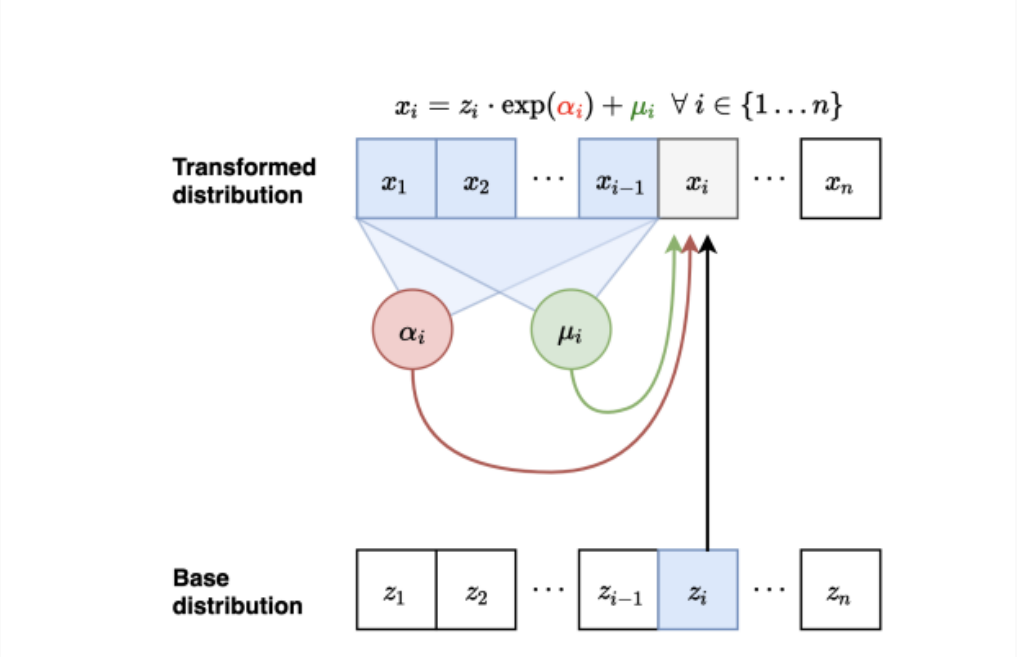
Continuous Autoregressive models as flow models

- Consider a Gaussian autoregressive model:

$$\boxed{p_\theta(x) = \prod_{i=1}^{p} p_\theta(x_i | x_{1:i-1}) \ .}$$

  such that $p_\theta\left(x_i \mid x_{1:i-1}\right) = \mathrm{N}\left(\mathtt{m}_i\left(x_1, \cdots, x_{i-1}\right), \exp\left(\alpha_i\left(x_1, \cdots, x_{i-1}\right)\right)^2\right)$.
- Here, $\mathtt{m}_i = \mathtt{m}_{\theta_i, i}$ and $\alpha_i = \alpha_{\theta_i, i}$ are NN for $i > 1$ and constants for $i = 1$.
- Sampler for this model:
- Sample $z_i \sim \mathrm{N}(0, 1)$ for $i = 1, \cdots, n$ and set $z = z_{1:p}$.
- Let $x_1 = T_\theta(z)_1 = \exp\left(\alpha_1\right) z_1 + \mathtt{m}_1$. Compute $\mathtt{m}_2\left(x_1\right), \alpha_2\left(x_1\right)$
- Let $x_i = T_\theta(z)_i = \exp\left(\alpha_i(x_{1:i-1})\right) z_i + \mathtt{m}_i(x_{1:i-1})$. Compute $\mathtt{m}_{i+1}\left(x_{1:i}\right), \alpha_{i+1}\left(x_{1:i}\right)$, for $i = 2 \ldots p$.
- Flow interpretation: transforms samples from the standard Gaussian $(z_1, z_2, \ldots, z_n)$ to those generated from the model $(x_1, x_2, \ldots, x_n)$ via invertible transformations (parameterized by $\mathtt{m}_i, \alpha_i$).

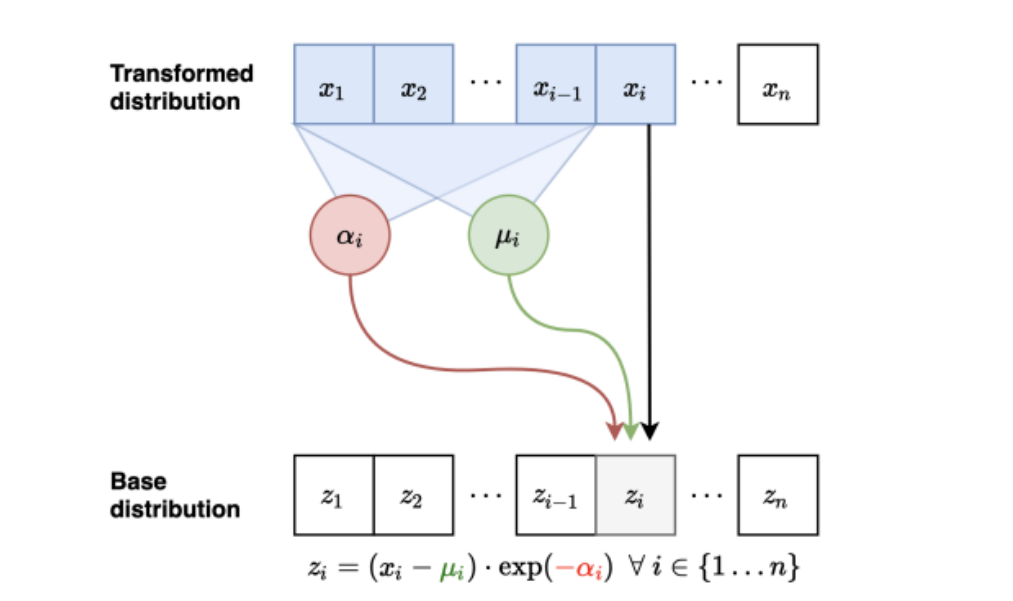
Masked Autoregressive Flow (MAF) ?



Credit: Eric Jang's blog

- Forward mapping from T_θ:
- Let $x_1 = T_\theta(z)_1 = \exp(\alpha_1) z_1 + \mathtt{m}_1$. Compute $\mathtt{m}_2(x_1), \alpha_2(x_1)$
- Let $x_i = T_\theta(z)_i = \exp(\alpha_i(x_{1:i-1})) z_i + \mathtt{m}_i(x_{1:i-1})$. Compute $\mathtt{m}_{i+1}(x_{1:i}), \alpha_{i+1}(x_{1:i})$, for $i = 2 \dots p$.
- Sampling is sequential and slow (like autoregressive): $O(p)$ time

Masked Autoregressive Flow (MAF)



Credit: Eric Jang's blog

- Inverse mapping from $\mathrm{T}_\theta^{\leftarrow}$: ?
- Jacobian is lower diagonal, hence efficient determinant computation
- Likelihood evaluation is easy and parallelizable (like MADE)
- Layers with different variable orderings can be stacked

The autoregressive layer

- **?** introduces the **autoregressive layer**:
  - $z = \{z_i\}_{i=1}^p$
  - $\sigma_i(z_{1:i-1}) = \text{sigmoid}(\alpha_i(z_{1:i-1}))$
  - **Forward** transform $\mathrm{T}_\theta(z)_i = \sigma_i(x_{1:i-1})z_i + (1 - \sigma_i(x_{1:i-1}))\mathrm{m}_i(x_{1:i-1})$ with

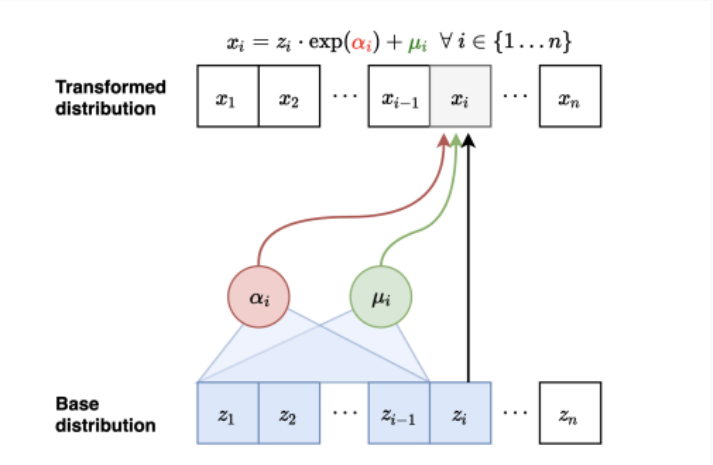
$$x_{1:i-1} = \mathrm{T}_\theta(z)_{1:i-1} \ .$$

  - **Reverse** transform ?
  - **Log-Jacobian** ?
- The Jacobian is **triangular** (easy computation of the determinant).
- Parameterization with the sigmoid is numerically stable (inspired by LSTM **?**).
- Between each autoregressive layer the ordering is **reversed**.
- More involved autoregressive models in practice:
  - **Convolutional** autoregressive models **?**.

Sampling vs estimation for MAF

- MAF:
- Sampling $O(p)$;
- Parallel estimation $O(1)$.
- Can we have a sampling in $O(1)$?

Inverse Autoregressive Flow (IAF)



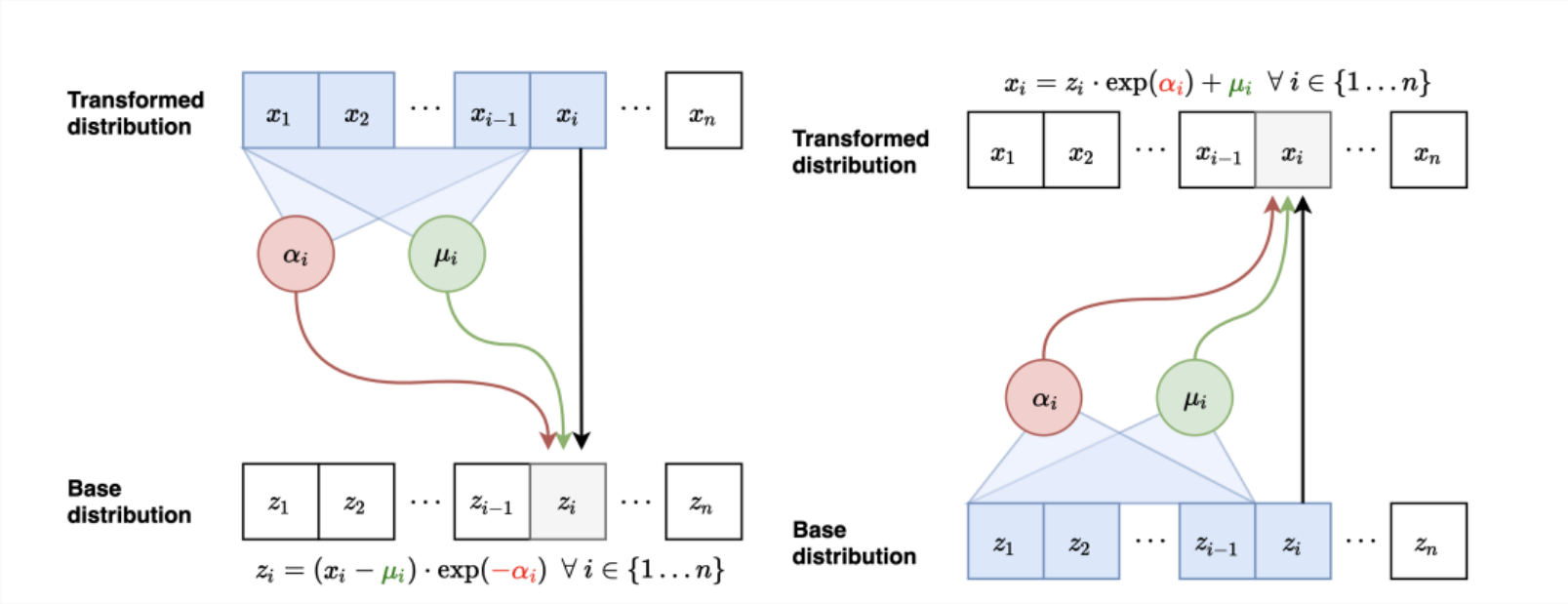
Credit: Eric Jang's blog

- Forward mapping from T_θ (parallel).
- Sample $z_i \sim \mathrm{N}(0,1)$ for $i = 1, \cdots, p$ and set $z = z_{1:p}$.
- Compute all $\mathtt{m}_i(z_{1:i-1}), \alpha_i(z_{1:i-1})$ (can be done in parallel)
- Again $\mathtt{m}_i = \mathtt{m}_{\theta_i,i}$ and $\alpha_i = \alpha_{\theta_i,i}$ are NN for $i > 1$ and constants for $i = 1$.
- Let $x_1 = \mathrm{T}_\theta(z)_1 = \exp(\alpha_1)\, z_1 + \mathtt{m}_1$
- Let $x_i = \mathrm{T}_\theta(z)_i = \exp(\alpha_i(z_{1:i-1}))\, z_i + \mathtt{m}_i(z_{1:i-1})$ for $i > 1$
- Sampling is fast now (in parallel)!
- Note: Fast to evaluate likelihoods of a generated point using $(z_1, z_2, \dots, z_n)$, just Gaussian $p_\theta(x) = p_\theta(\mathrm{T}_\theta(z)) =$?

Inverse Autoregressive Flow (IAF)

- Inverse mapping from $\mathrm{T}_{\theta}^{\leftarrow}$ (sequential).
- Let $z_1 = T_\theta(x)_1^{\leftarrow} = \exp\left(-\alpha_1\right)\left(z_1 - \mathtt{m}_1\right)$. Compute $\mathtt{m}_2\left(z_1\right), \alpha_2\left(z_1\right)$
- Let $z_i = T_\theta(x)_i^{\leftarrow} = \exp\left(-\alpha_i(x_{1:i-1})\right)\left(z_i - \mathtt{m}_i(x_{1:i-1})\right)$. Compute $\mathtt{m}_{i+1}\left(z_{1:i}\right), \alpha_{i+1}\left(z_{1:i}\right)$, for $i = 2 \ldots p$.
- Sampling is sequential and slow (like autoregressive): $O(p)$ time
- Fast to sample from, slow to evaluate likelihoods of data points (train)

IAF is inverse of MAF



Credit: Eric Jang's blog

Inverse pass of MAF (left) vs. Forward pass of IAF (right)

- Interchanging z and x in the inverse transformation of MAF gives the forward transformation of IAF
- Similarly, forward transformation of MAF is inverse transformation of IAF

IAF vs. MAF

- Computational tradeoffs
- MAF: Fast likelihood evaluation, slow sampling
- IAF: Fast sampling, slow likelihood evaluation
- MAF more suited for training based on MLE, density estimation
- IAF more suited for real-time generation
- Can we get the best of both worlds?

Parallel Wavenet

- Two part training with a teacher and student model
- Teacher is parameterized by MAF. Teacher can be efficiently trained via MLE
- Once teacher is trained, initialize a student model parameterized by IAF.
- Student model cannot efficiently evaluate density for external datapoints but allows for efficient sampling
- Key observation: IAF can also efficiently evaluate densities of its own generations

Parallel Wavenet

- Given a teacher with density t_θ parametrized by θ (MAF)
- Probability density distillation: student distribution is trained to minimize the KL divergence between student s_ψ, parametrized by ψ (IAF), and teacher t_θ:

$$\psi \mapsto \int s_\psi(x) \log \frac{s_\psi(x)}{t_\theta(x)} \mathrm{d}x = \int q_0(z) \log \frac{s_\psi(\mathrm{T}_\psi(z))}{t_\theta(\mathrm{T}_\psi(z))} \mathrm{d}x \ .$$

- Evaluating and optimizing Monte Carlo estimates of this objective

$$N^{-1} \sum_{i=1}^{N} \log \frac{s_\psi(\mathrm{T}_\psi(Z^i))}{t_\theta(\mathrm{T}_\psi(Z^i))}$$

- This requires:
- Forward transformation T_ψ (IAF)
- Density of x assigned by student model, s_ψ
- Density of x assigned by teacher model (MAF), t_θ
- All operations above can be implemented efficiently

Parallel Wavenet

- Training
- Step 1: Train teacher model (MAF) via MLE
- Step 2: Train student model (IAF) to minimize KL divergence with teacher
- Test-time: Use student model for testing

Energy-based models

Maximum likelihood estimation (reminder...)

- Consider the case $\Theta \subset \mathbb{R}^d$.
- Choice for the family $\{\mu_\theta : \theta \in \Theta\}$?

$$\{\mu_\theta : \mu_\theta \ll \lambda\,, \quad p_\theta = \mathrm{d}\mu_\theta/\mathrm{d}\lambda\}\ .$$

- Choice for the divergence **D**?
- $\mathbf{D} = \mathrm{KL}$: problem $\mathrm{KL}\left(\mu_\theta \,\|\, \hat{\mu}_N\right) = \infty$...
- Recall that ideally if $\mathbf{D} = \mathrm{KL}$, we would like to minimize

$$\mathrm{KL}\left(\mu^\star \,\|\, \mu_\theta\right) = -\int \mathrm{d}\mu^\star \log\left(\frac{\mathrm{d}\mu_\theta}{\mathrm{d}\mu^\star}\right)\ .$$

- This is equivalent to maximize if $\mu^\star \ll \lambda$,

$$\theta \mapsto \int \mathrm{d}\mu^\star \log\left(\frac{\mathrm{d}\mu_\theta}{\mathrm{d}\lambda}\right)\ .$$

- Solution: replace the integral by an empirical version

$$\theta \mapsto N^{-1} \sum_{i=1}^{N} \log p_\theta(x_i)\ .$$

EBMs rationale

- Consider the case $\lambda = \mathrm{Leb}$ and $\mathsf{X} = \mathbb{R}^d$.
- EBM consists in defining a family $\{\mu_\theta \,:\, \theta \in \Theta\}$ directly from a family of potential/energy functions $\{U_\theta \,:\, \theta \in \Theta\}$: for $x \in \mathbb{R}^d$

$$p_\theta(x) = (\mathrm{d}\mu_\theta/\mathrm{dLeb})(x) = \exp[-U_\theta(x)]/\mathfrak{Z}(\theta)\ ,$$
$$\mathfrak{Z}(\theta) = \textstyle\int_{\mathbb{R}^d} \exp[-U_\theta(\tilde{x})]\mathrm{d}\tilde{x}\ .$$

- U_θ is typically a **neural network** ($\theta \in \Theta$ is a set of parameters).
- The **likelihood** is then:

$$\boxed{\hat{\ell}_N(\theta) = -(1/N)\textstyle\sum_{k=1}^N U_\theta(x^k) - \log \mathfrak{Z}(\theta)\ .}$$

- The first term in the right-hand side has **an explicit gradient** (computed by auto-diff in practice...).
- The second term unfortunately is untractable... but we can write its gradient w.r.t. θ as integral w.r.t. p_θ.

The Fisher and Louis identities

Proposition 1
Under appropriate conditions, the following identities hold:

$$\nabla_\theta \log \mathfrak{Z}(\theta) = ?$$
$$\nabla_\theta^2 \log \mathfrak{Z}(\theta) = ? \ .$$

Proposition 2
Under appropriate conditions, the following identities hold:

$$\nabla_\theta \log \mathfrak{Z}(\theta) = -\int \nabla_\theta U_\theta(x) p_\theta(x) \mathrm{Leb}(\mathrm{d}x) \,,$$

$$\nabla_\theta^2 \log \mathfrak{Z}(\theta) = -\int \nabla_\theta^2 U_\theta(x) p_\theta(x) \mathrm{Leb}(\mathrm{d}x) + \int [\nabla_\theta \bar{U}_\theta(x) \nabla_\theta \bar{U}_\theta(x)^{\mathrm{T}}] p_\theta(x) \mathrm{Leb}(\mathrm{d}x) \,,$$

$$\nabla_\theta \bar{U}_\theta(x) = \nabla_\theta U_\theta(x) - \int \nabla_\theta U_\theta(x) p_\theta(x) \mathrm{d}x \,.$$

Training EBMs

- Maximizing the **likelihood**

$$\boxed{\hat{\ell}_N(\theta) = -(1/N)\sum_{k=1}^{N} U_\theta(x^k) - \log \mathfrak{Z}(\theta)\ .}$$

- Taking the gradient of the **log-partition** using Fisher identity:

$$\log \mathfrak{Z}(\theta) = -\int \nabla_\theta U_\theta(x) p_\theta(x) \mathrm{d}x\ .$$

- Taking the gradient of the **empirical likelihood** $\hat{\ell}_N$, we get

$$\boxed{\nabla_\theta \hat{\ell}_N(\theta) = -(1/N)\sum_{k=1}^{N} \nabla_\theta U_\theta(x^k) + \mu_\theta[\nabla_\theta U_\theta]\ .}$$

- We take only a mini-batch of the first term at each iterations of your favorite optimization algorithm.
- At **equilibrium** $\theta^\star$, we cannot distinguish the expectation of $\nabla_\theta U_{\theta^\star}$ w.r.t. $\mu^\star$ and $\mu_{\theta^\star}$.

Training EBMs

- Taking the gradient of the **empirical likelihood** $\hat{\ell}_N$, we get

$$\boxed{\nabla_\theta \hat{\ell}_N(\theta) = -(1/N) \sum_{k=1}^{N} \nabla_\theta U_\theta(x^k) + \mu_\theta[\nabla_\theta U_\theta] \, .}$$

- We take only a mini-batch of the first term at each iterations of your favorite optimization algorithm.
- At **equilibrium** $\theta^\star$, we cannot distinguish the expectation of $\nabla_\theta U_{\theta^\star}$ w.r.t. $\mu^\star$ and $\mu_{\theta^\star}$.
- Approximating $\mu_\theta[\nabla_\theta U_\theta]$, requires **statistical sampling**.
- For this, we consider a family of MCMC algorithm $\{P_\theta \, : \, \theta \in \Theta\}$ such that P_θ targets μ_θ for any θ.

Langevin Monte Carlo for training

- Choice for family of MCMC algorithm $\{P_\theta : \theta \in \Theta\}$:
  - **Markov chains** targeting (approximately) μ_θ.
  - **U**nadjusted **L**angevin **A**lgorithm

$$\boxed{X_{k+1} = X_k - \gamma \nabla_x U_\theta(X_k) + \sqrt{2\gamma} Z_{k+1} \ ,}$$

  - γ is a stepsize, $\nabla_x U_\theta$ is computed with backpropagation.
- In practice:
  - We add some **regularization** to the contrastive divergence.
  - We consider **short runs** of MCMC.
  - The initialization of the MCMC is important: **warm-start** (**persistent contrastive divergence**, see Tieleman (2008)) or not (see Nijkamp et al. (2019)).
  - Tutorial with Pytorch implementation based on Du and Mordatch (2019).

EBM training algorithm using ULA

Algorithm 1 Training of EBM

Input: n_{iter}, K, $\hat{\mu}$, N_{batch}, γ, δ, α, θ_0.
$B \neq \emptyset$.
for $n = 0$ to $n_{\text{iter}} - 1$ **do**
  Sample $X_n^{+,1:N_{\text{batch}}} = \{X_n^{+,k}\}_{k=1}^{N_{\text{batch}}}$ i.i.d. from $\{x^i\}_{i=1}^{N}$.
  if B is not empty **then**
    Sample $X_n^{0,1:N_{\text{batch}}} = \{X_n^{0,k}\}_{k=1}^{N_{\text{batch}}}$ i.i.d. from $(1-\alpha)B + \alpha \mathrm{N}(0, \mathrm{Id})$.
  else
    Sample $X_n^{0,1:N_{\text{batch}}} = \{X_n^{0,k}\}_{k=1}^{N_{\text{batch}}}$ i.i.d. from $\mathrm{N}(0, \mathrm{Id})$.
  end if
  for $k = 0$ to $K - 1$ **do**
    $X_n^{k+1,1:N_{\text{batch}}} = X_n^{k,1:N_{\text{batch}}} + \gamma \nabla_x U_{\theta_n}(X_n^{k,1:N_{\text{batch}}}) + \sqrt{2\gamma} Z_n^{k+1,1:N_{\text{batch}}}$.
  end for
  $X_n^{-,1:N_{\text{batch}}} = X_n^{K,1:N_{\text{batch}}}$.
  $\theta_{n+1} = \theta_n - (\delta / N_{\text{batch}}) \sum_{\ell=1}^{N_{\text{batch}}} \{\nabla_\theta U_{\theta_n}(X_n^{+,\ell}) - \nabla_\theta U_{\theta_n}(X_n^{-,\ell})\}$.
  $B = X_n^{-,1:N_{\text{batch}}}$.
end for

Example-based synthesis

Link with example-based synthesis

- Different density models:
  - In **Energy-Based Models**: $p_\theta(x) = \exp[-U_\theta(x)]/\mathfrak{Z}(\theta)$.
  - In **Maximum Entropy Models**: $p_\theta(x) = \exp[-\langle \theta, f(x) - f(x_0)\rangle]/\mathfrak{Z}(\theta)$.
- Training losses:
  - In **Energy-Based Models**: $\nabla_\theta \hat{\ell}_N(\theta) = -N^{-1} \sum_{i=1}^N \nabla_\theta U_\theta(x_i) + \mu_\theta[\nabla_\theta U_\theta]$.
  - In **Maximum Entropy Models**: $\nabla_\theta \mathfrak{Z}(\theta) = -f(x^0) + \mu_\theta[\nabla_\theta U_\theta]$.
- Some key differences
  - $\hat{\mu}$ is replaced by δ_{x^0}. Only one example to train the model.
  - In EBMs we train a neural network, in Maximum Entropy Models the dependency w.r.t. the parameters is **linear**.
  - More **flexibility** in EBMs but no (trivial) maximum entropy interpretation.
- **Same sampling algorithm**.

Summary of EBMs

- **Advantages**:
  - Model the **potential directly**.
  - Usually allows for model with **less parameters** than VAE, GANs or NFs.
  - Compositionality via Product of Experts Hinton (2002).
- **Problems**:
  - **Training with MCMC is long**. This can be avoided if we replace the Kullback-Leibler objective with a Fisher objective (connection with score-matching Song and Kingma (2021)).
  - **Instabilities** with training Du and Mordatch (2019).
  - Density on $\mathbb{R}^d$. Usually the data is supported on a **low dimensional manifold** Arbel et al. (2020).
- **Links with other methods**:
  - Connection with GANs Che et al. (2020).
  - Connection with VAEs Xiao et al. (2020).
  - Connection with score-matching Song and Kingma (2021); Gao et al. (2020).

References

Michael Arbel, Liang Zhou, and Arthur Gretton. Generalized energy based models. *arXiv preprint arXiv:2003.05033*, 2020.

Tong Che, Ruixiang Zhang, Jascha Sohl-Dickstein, Hugo Larochelle, Liam Paull, Yuan Cao, and Yoshua Bengio. Your gan is secretly an energy-based model and you should use discriminator driven latent sampling. *Advances in Neural Information Processing Systems*, 33:12275–12287, 2020.

Yilun Du and Igor Mordatch. Implicit generation and modeling with energy based models. *Advances in Neural Information Processing Systems*, 32, 2019.

Ruiqi Gao, Yang Song, Ben Poole, Ying Nian Wu, and Diederik P Kingma. Learning energy-based models by diffusion recovery likelihood. *arXiv preprint arXiv:2012.08125*, 2020.

Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

Erik Nijkamp, Mitch Hill, Song-Chun Zhu, and Ying Nian Wu. Learning non-convergent non-persistent short-run mcmc toward energy-based model. *Advances in Neural Information Processing Systems*, 32, 2019.

References ii

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015.

Yang Song and Diederik P Kingma. How to train your energy-based models. *arXiv preprint arXiv:2101.03288*, 2021.

Tijmen Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th international conference on Machine learning*, pages 1064–1071, 2008.

Jakub M Tomczak. Latent variable models. In *Deep Generative Modeling*, pages 57–127. Springer, 2022.

J. Wolfowitz. The minimum distance method. *Ann. Math. Statist.*, 28(1):75–88, 03 1957. doi: 10.1214/aoms/1177707038.

Zhisheng Xiao, Karsten Kreis, Jan Kautz, and Arash Vahdat. Vaebm: A symbiosis between variational autoencoders and energy-based models. *arXiv preprint arXiv:2010.00654*, 2020.