Generative AI and LLM

Flow-based generative models
Normalizing Flows
CS5202

Course Instructor : Dr. Nidhi Goyal

Mahindra University
Global Thinkers. Engaged Leaders.

Lecture Plan

- Flow-based generative models
  - Normalizing Flows

Flow-based generative models

- A flow-based generative models are constructed by a sequence of invertible transformations.

Flow based generative models

WIKIPEDIA
25 years of the free encyclopedia

Search Wikipedia | Search

Flow-based generative model

8 languages

Article Talk | Read Edit View history Tools

Contents hide

From Wikipedia, the free encyclopedia

A **flow-based generative model** is a generative model used in machine learning that explicitly models a probability distribution by leveraging **normalizing flow**,[1][2][3] which is a statistical method using the change-of-variable law of probabilities to transform a simple distribution into a complex one.

The direct modeling of likelihood provides many advantages. For example, the negative log-likelihood can be directly computed and minimized as the loss function. Additionally, novel samples can be generated by sampling from the initial distribution, and applying the flow transformation.

In contrast, many alternative generative modeling methods, such as variational autoencoders (VAEs), generative adversarial networks (GANs), or diffusion models, do not explicitly represent the likelihood function.

Method [edit]

Let z_0 be a (possibly multivariate) random variable with distribution $p_0(z_0)$.

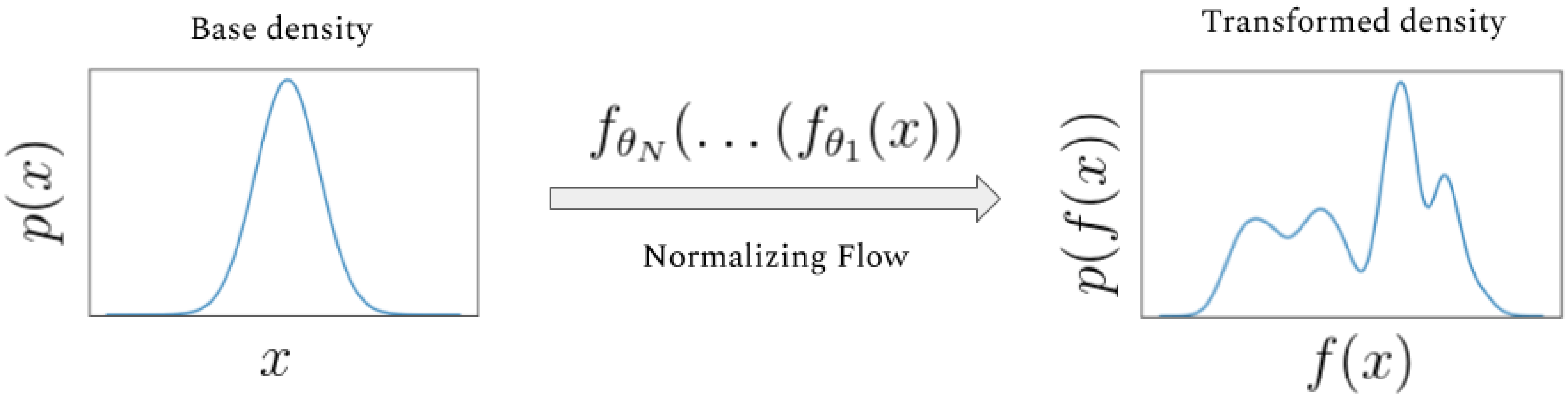
For $i = 1, \ldots, K$, let $z_i = f_i(z_{i-1})$ be a sequence of random variables transformed from z_0. The functions $f_1, \ldots, f_K$ should be invertible, i.e. the inverse function f_i^{-1} exists. The final output z_K models the target distribution.

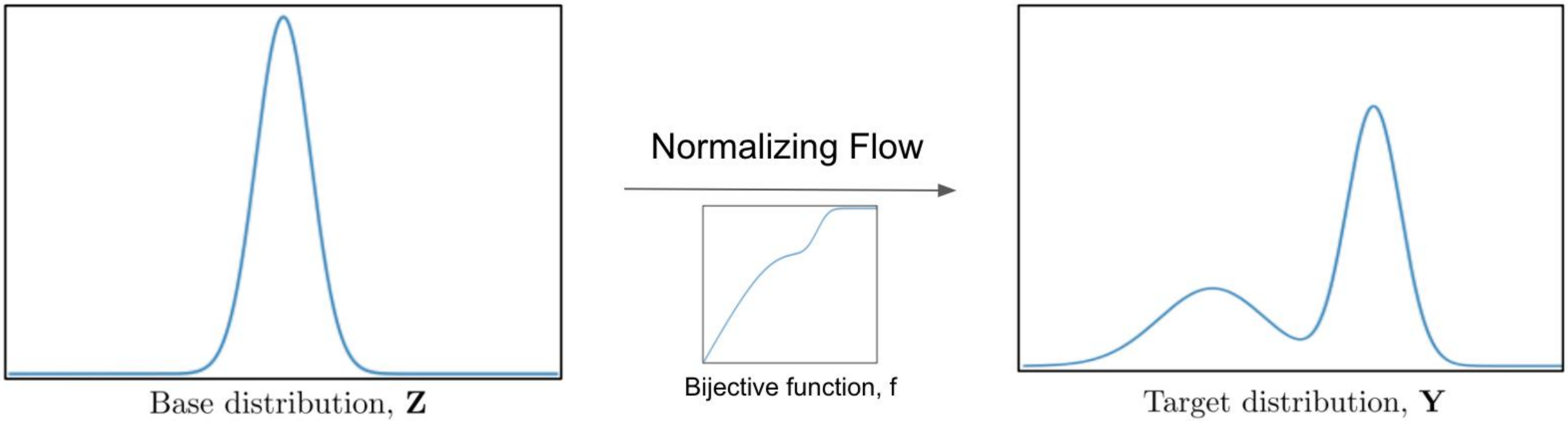
Part of a series on
Machine learning and data mining

Paradigms	[show]
Problems	[show]
Supervised learning (classification • regression)	[show]
Clustering	[show]
Dimensionality reduction	[show]
Structured prediction	[show]
Anomaly detection	[show]
Neural networks	[show]
Reinforcement learning	[show]
Learning with humans	[show]
Model diagnostics	[show]
Mathematical foundations	[show]
Journals and conferences	[show]
Related articles	[show]

Normalizing flows

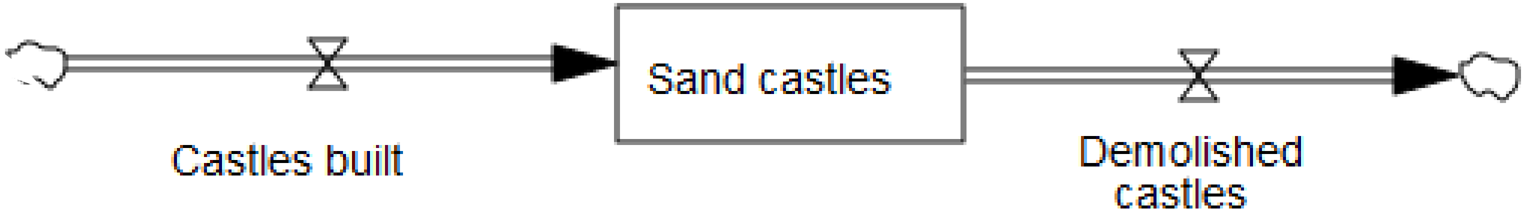
Normalizing flows learn an *invertible* mapping f: X→Z, where X is our data distribution and Z is a chosen latent-distribution.





Normalizing Flow
Base distribution, Z
Bijective function, f
Target distribution, Y

Normalizing flows



Sandcastles

How to create a sandcastle:

Step 1: Take a sandcastle

Step 2: Destroy the sandcastle

Step 3: Remember how you destroyed the sandcastle

Step 4: Reverse the process

Key Idea

Once you know how to reconstruct sandcastles, you can start with some different “sand”, apply this process, and end up with a different “sandcastle”

Why Normalizing Flows?

- **NFs optimize the exact log-likelihood of the data, log(p_X)**
  - VAEs optimize the ______________________
  - GANs _________________________________
- **NFs infer exact latent-variable values z, which are useful for downstream tasks**
  - The VAE infers a distribution over _____________ values
  - GANs _______________________________________
- **Potential for memory savings, with NFs gradient computations scaling constant to their depth**
  - Both VAE’s and GAN’s gradient computations scale _________to their depth
- **NFs require only an encoder to be learned**
  - VAEs require ____________________________________
  - GANs require ___________________________________

Why Normalizing Flows?

- **NFs optimize the exact log-likelihood of the data, $\log(p_X)$**
  - VAEs optimize the lower bound (ELBO)
  - GANs learn to fool a discriminator network
- **NFs infer exact latent-variable values z, which are useful for downstream tasks**
  - The VAE infers a distribution over latent-variable values
  - GANs do not have a latent-distribution
- **Potential for memory savings, with NFs gradient computations scaling constant to their depth**
  - Both VAE's and GAN's gradient computations scale linearly to their depth
- NFs require only an encoder to be learned
  - VAEs require encoder and decoder networks
  - GANs require generative and discriminative networks

How to ensure that we can reverse?

- Use invertible mapping

Bijective function

- Normalizing flows require:
- **f: X→Z**

to be **bijective** because:

  - You must go forward (data → latent)
  - You must go backward (latent → data)

Linear algebra basics

- Jacobian
- Change of variables

Jacobian Matrix

2.1 Jacobian matrix

Given a function of mapping a n-dimensional input vector $\mathbf{x}$ to a m-dimensional output vector, $\mathbf{f} : \mathbb{R}^n \mapsto \mathbb{R}^m$, the matrix of all first-order partial derivatives of this function is called the Jacobian matrix $\mathbf{J}$, where one entry on the i-th row and j-th column is $\mathbf{J}_{ij} = \frac{\partial f_i}{\partial x_j}$.

$$\mathbf{J} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

Source: flow_based_deep_generative_models_report.pdf

Jacobian matrix

$$z = g(x) = f^{-1}(x)$$

$$J_x g(x) = \begin{bmatrix} \frac{\partial z_1}{\partial x_1} & \cdots & \frac{\partial z_1}{\partial x_k} \\ \vdots & & \vdots \\ \frac{\partial z_k}{\partial x_1} & \cdots & \frac{\partial z_k}{\partial x_k} \end{bmatrix}$$

Change of variables theorm

2.2 Change of variable theorem

Given some random variable $z \sim \pi(z)$ and a invertible mapping $x = f(z)$ (i.e., $z = f^{-1}(x) = g(x)$). Then, the distribution of x is

$$p(x) = \pi(z) \left| \frac{dz}{dx} \right| = \pi(g(x)) \left| \frac{dg}{dx} \right| .$$

The multivariate version takes the following form:

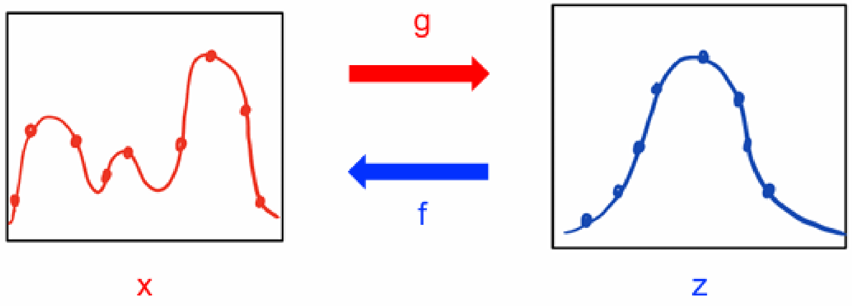
$$p(\mathbf{x}) = \pi(\mathbf{z}) \left| \det \frac{d\mathbf{z}}{d\mathbf{x}} \right| = \pi(g(\mathbf{x})) \left| \det \frac{dg}{d\mathbf{x}} \right| ,$$

where $\det \frac{dg}{d\mathbf{x}}$ is the *Jacobian determinant* of g.

Source: flow_based_deep_generative_models_report.pdf

Intuition/Math

Normalizing Flows – Log likelihood



Bijection (and invertibility) allow us to directly compute the likelihood:

$$\int p_x(x)dx = \int p_z(g(x))dz$$

In multiple dimensions, we generalize to the determinant of the Jacobian

$$p_x(x) = p_z(g(x))\left|\frac{dg(x)}{dx}\right| \rightarrow p_z(g(x))|det.J(g(x))|$$

$$\log p_x(x) = \log p_z(g(x)) + \log|det.J(g(x))|$$

Intuitively

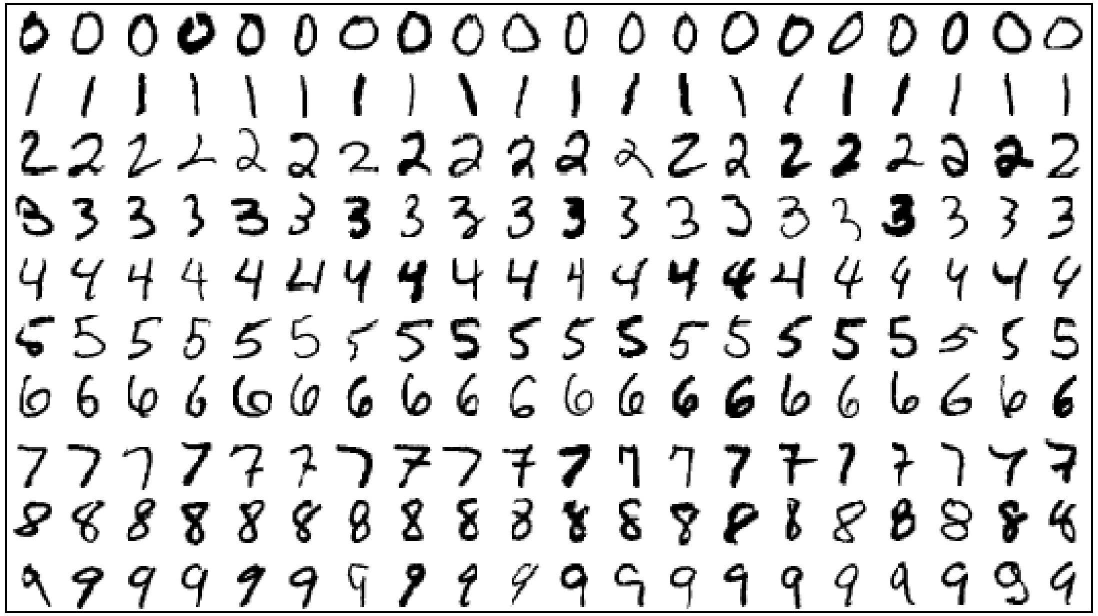
z = g(x) determines where a point in x-space maps to z-space (where to move grains of sand)

|det. J(g(x))| describes how much probability mass (sand) gets moved in a local neighborhood.

Math and Code

- Going with the Flow: An Introduction to Normalizing Flows | Brennan Gebotys

Implementation



[Tutorial 9: Normalizing Flows for Image Modeling — PyTorch Lightning 2.6.1 documentation]


THERE IS NO
SUCH THING
AS A
FREE LUNCH

Downside of NFs

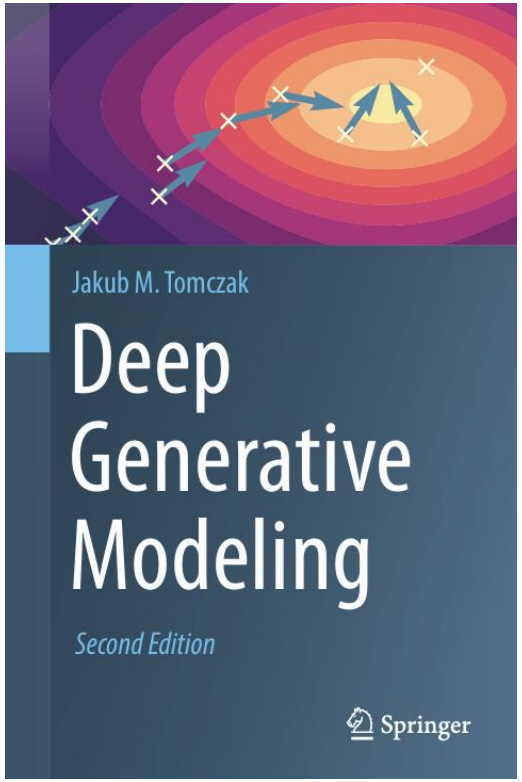
- The requirements of invertibility and efficient Jacobian calculations restrict model architecture
- NFs generative results are still behind VAEs and GANs

References

- https://hermandong.com/pdf/flow_based_deep_generative_models_report.pdf
- Flow-based Deep Generative Models | Lil'Log
- Going with the Flow: An Introduction to Normalizing Flows | Brennan Gebotys

Books and lecture notes

Deep Generative Modeling



GitHub

Generative AI & LLMs · Fall 2026