Deep Generative Models (Fall 2024)

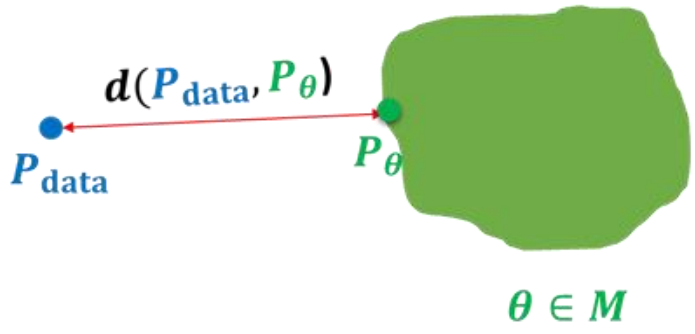
CS HUFS

Normalizing Flows

- Flow Models
- Normalizing Flows
- NICE
- Real-NVP

Recap of likelihood-based learning so far:

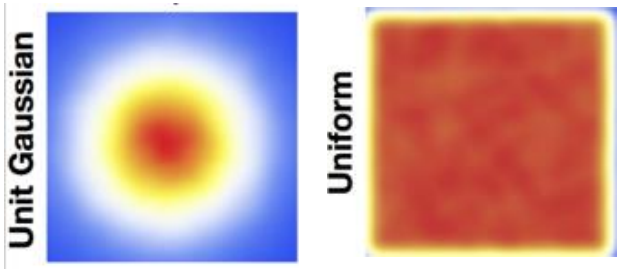
$\mathbf{x}_i \sim P_{\text{data}}$
$i = 1, 2, \ldots, n$



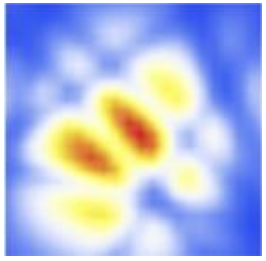
- Model families:
  - Autoregressive Models: $p_\theta(\mathbf{x}) = \prod_{i=1}^{n} p_\theta(x_i | \mathbf{x}_{<i})$
  - Variational Autoencoders: $p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}, \mathbf{z}) d\mathbf{z}$
- Autoregressive models provide tractable likelihoods but no direct mechanism for learning features
- Variational autoencoders can learn feature representations (via latent variables $\mathbf{z}$) but have intractable marginal likelihoods
- **Key question**: Can we design a latent variable model with tractable likelihoods? Yes!

Simple Prior to Complex Data Distributions

- Desirable properties of any model distribution $p_\vartheta(\mathbf{x})$:
  - 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

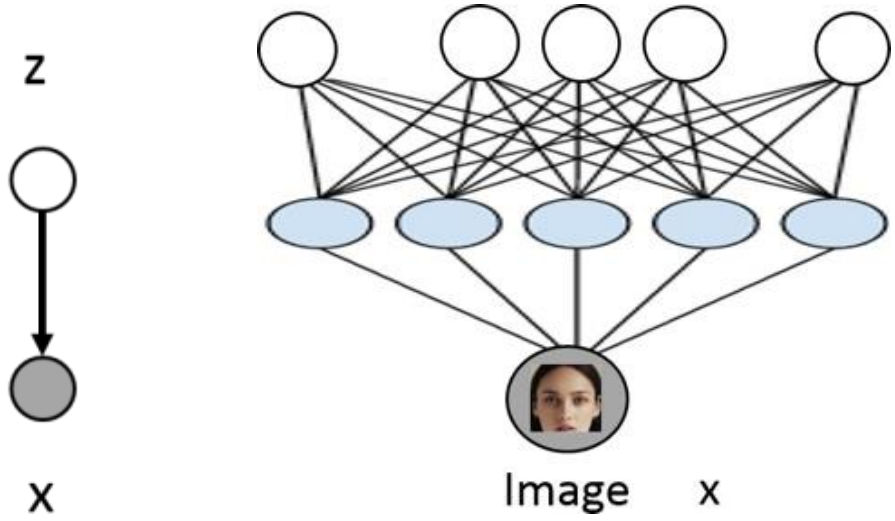


- Unfortunately, data distributions are more complex (multi-modal)



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

Flow Models are similar to VAEs

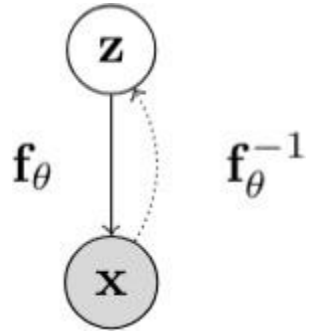


A flow model is similar to a variational autoencoder (VAE):

1. Start from a simple prior: $\mathbf{z} \sim N(0, I) = p(\mathbf{z})$
2. Transform via $p(\mathbf{x} \mid \mathbf{z}) = N(\mu_\vartheta(\mathbf{z}), \Sigma_\vartheta(\mathbf{z}))$
3. Even though $p(\mathbf{z})$ is simple, the marginal $p_\vartheta(\mathbf{x})$ is very complex/flexible. However, $p_\vartheta(\mathbf{x}) = \int p_\vartheta(\mathbf{x}, \mathbf{z}) d\mathbf{z}$ is expensive to compute: need to enumerate all $\mathbf{z}$ that could have generated $\mathbf{x}$
4. What if we could easily "invert" $p(\mathbf{x} \mid \mathbf{z})$ and compute $p(\mathbf{z} \mid \mathbf{x})$ by design? How? Make $\mathbf{x} = f_\vartheta(\mathbf{z})$ a deterministic and invertible function of $\mathbf{z}$, so for any $\mathbf{x}$ there is a unique corresponding $\mathbf{z}$ (no enumeration)

Normalizing flow models

- Consider a directed, latent-variable model over observed variables X and latent variables Z
- In a **normalizing flow model**, the mapping between Z and X, given by $\mathbf{f}_\vartheta : \mathrm{R}^n \rightarrow \mathrm{R}^n$, is deterministic and invertible such that $X = \mathbf{f}_\vartheta(Z)$ and $Z = \mathbf{f}_\vartheta^{-1}(X)$



- Using change of variables, the marginal likelihood $p(\mathbf{x})$ is given by

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

- Note: $\mathbf{x}, \mathbf{z}$ need to be continuous and have the same dimension.

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

$$\mathbf{z}_m = \mathbf{f}_\vartheta^m \circ \cdots \circ \mathbf{f}_\vartheta^1(\mathbf{z}_0) = \mathbf{f}_\vartheta^m(\mathbf{f}_\vartheta^{m-1}(\cdots(\mathbf{f}_\vartheta^1(\mathbf{z}_0)))) \triangleq \mathbf{f}_\vartheta(\mathbf{z}_0)$$

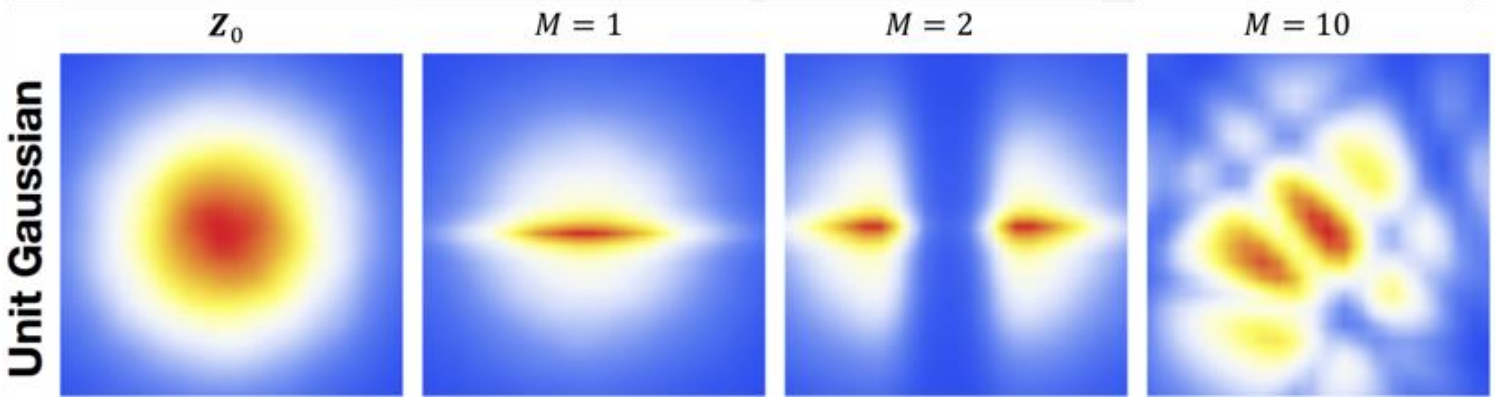
- Start with a simple distribution for $\mathbf{z}_0$ (e.g., Gaussian)
- Apply a sequence of M invertible transformations to finally obtain $\mathbf{x} = \mathbf{z}_M$
- By change of variables

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

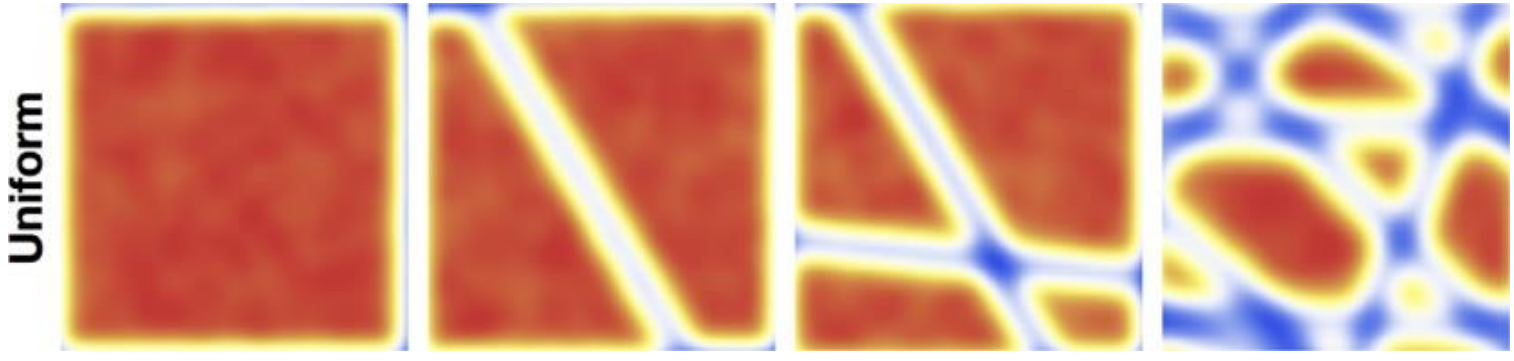
(Note: determininant of product equals product of determinants)

Planar flows (Rezende & Mohamed, 2016)

- Base distribution: Gaussian



- Base distribution: Uniform



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

Nonlinear Independent Components Estimation (NICE)

- NICE is a normalizing flow model, which uses additive coupling layer and rescaling layer to train $p_\theta(\mathbf{x})$

(a) Model trained on MNIST

(b) Model trained on TFD

Nonlinear Independent Components Estimation (NICE)

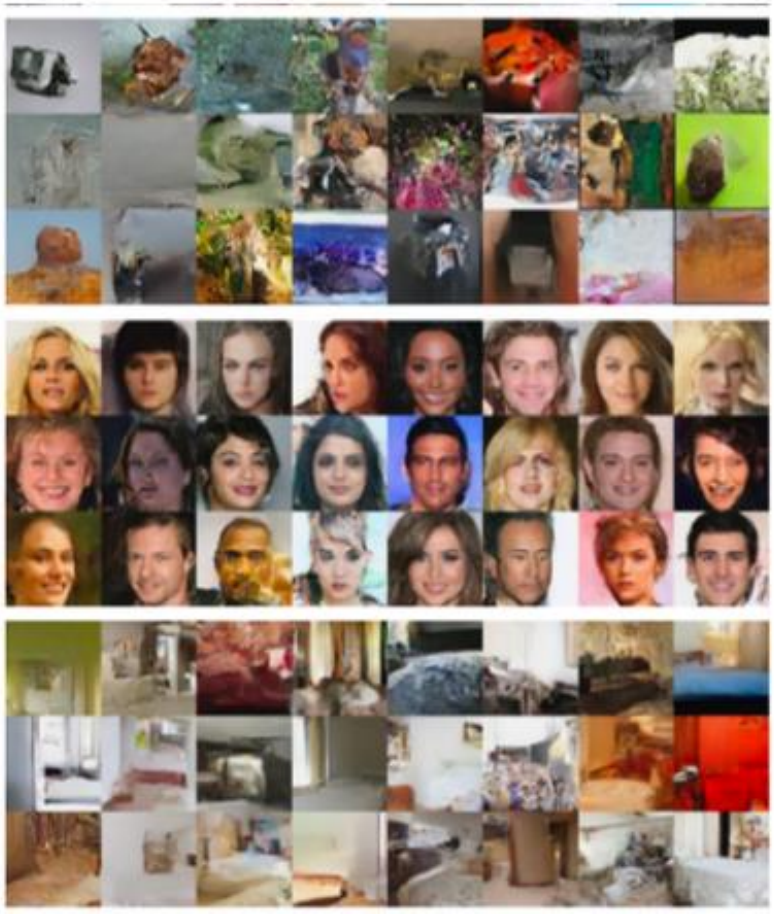
- Because NICE represents the relationship between x and z as a deep computable structure, it can produce okay images, even for datasets with high-dimensional data like MNIST and TFD.
- However, the Normalizing Flow model sets the dimensionality of x and z to be the same, which makes it computationally intensive.

(c) Model trained on SVHN

(d) Model trained on CIFAR-10

Non-Volume Preserving Extension of NICE (Real-NVP)

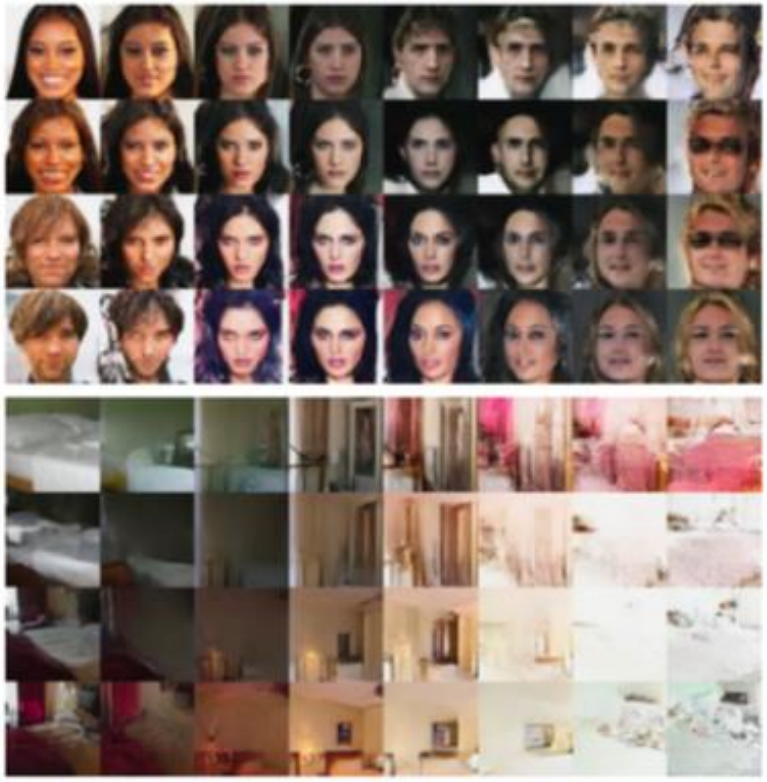
- Real-NVP model is an extension of NICE.
- It adds nonlinearity to the additive coupling layers in NICE using neural networks $\mu_\theta(\mathbf{x})$ and $\alpha_\theta(\mathbf{x})$.



Non-Volume Preserving Extension of NICE (Real-NVP)

- Real-NVP is good at learning complex relationships between data x and z, even for color images.
- Because z can be represented as a formula, it can interpolate to produce an image somewhere between $z^{(1)}$ and $z^{(N)}$.

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



Summary

Why Normalizing Flows Are Useful

Normalizing flows transform a simple distribution (e.g., Gaussian) into a complex one through a sequence of invertible and differentiable transformations.

- **Exact Likelihood Estimation**: Unlike some other generative models (e.g., GAN, VAE), normalizing flows provide an exact computation of the data likelihood, making them useful for density estimation.
- **Efficient Sampling**: Sampling is straightforward because of the invertibility of transformations.
- **Flexible Architectures**: Models like RealNVP, Glow, and Neural Spline Flows allow for expressive generative processes while retaining tractability.

Summary

Applications of Normalizing Flows

- **Density Estimation**: Precise modeling of high-dimensional data distributions.
- **Data Augmentation**: Transforming simple latent spaces into complex manifolds.
- **Bayesian Inference**: Serving as prior or posterior distributions in probabilistic models.
- **Audio and Speech Synthesis**: Models like WaveGlow (flow-based) have been used for efficient and high-quality speech generation.