Bilim Akademisi - Bilkent Üniversitesi Yapay Öğrenme Yaz Okulu 2020

Gökberk Cinbiş

Department of Computer Engineering

ORTA DOĞU TEKNİK ÜNİVERSİTESİ
MIDDLE EAST TECHNICAL UNIVERSITY

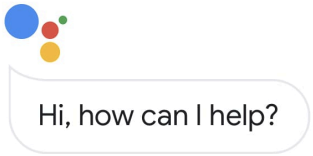
ImageLab

June 2020

Supervised recognition – success stories

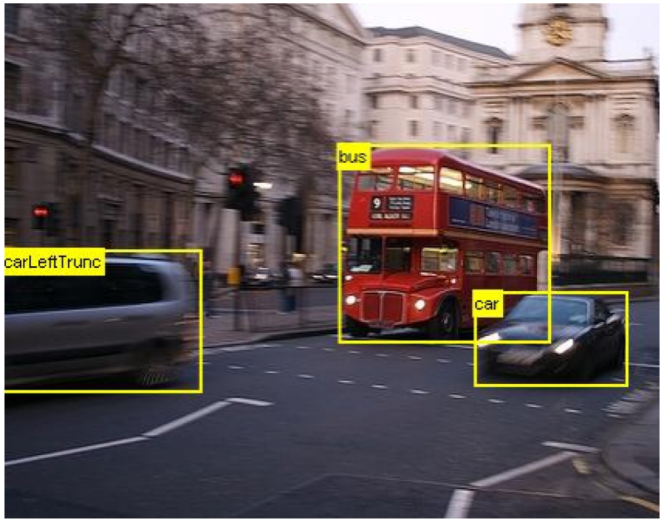
Slide collaged from cs231n of Stanford University

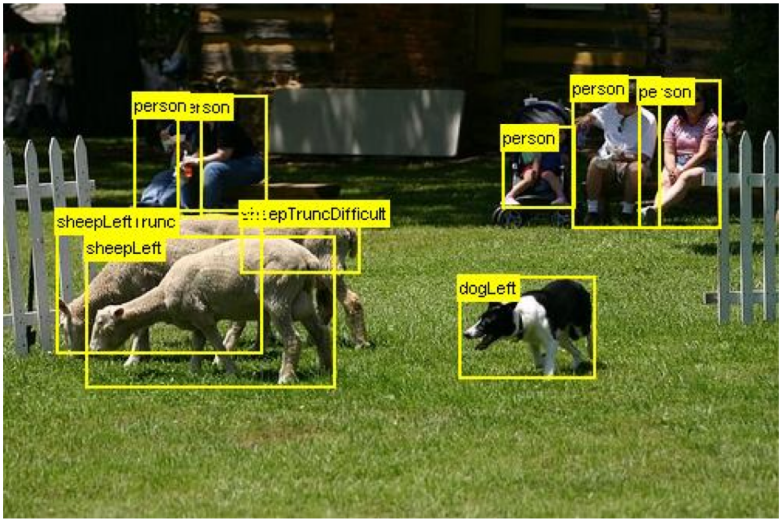
A white teddy bear sitting in the grass



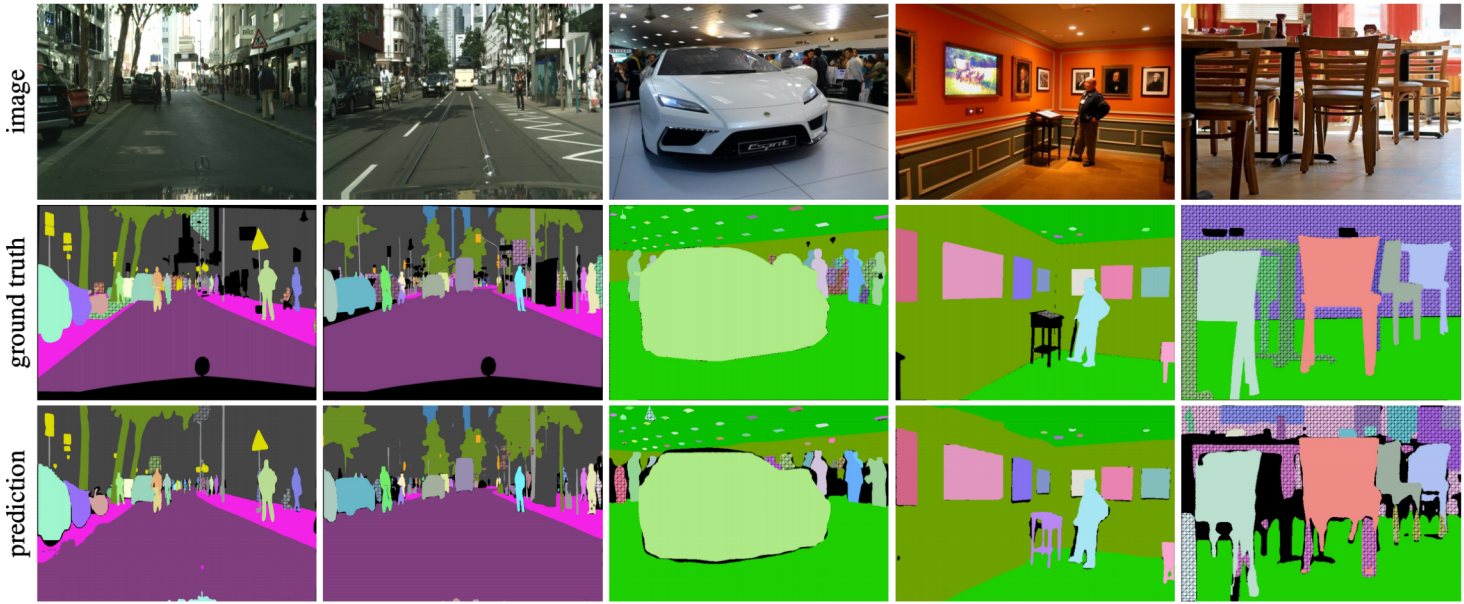
Supervised training – image classification

Supervised training – object detection





Supervised training – panoptic segmentation



https://arxiv.org/pdf/1801.00868.pdf

How many training examples do we need?

* Torralba, et al. 2008.

How many training examples do we need?

- 75.000 non-abstract nouns from WordNet*, some of which are *rare*





* Torralba, et al. 2008.

How many training examples do we need?

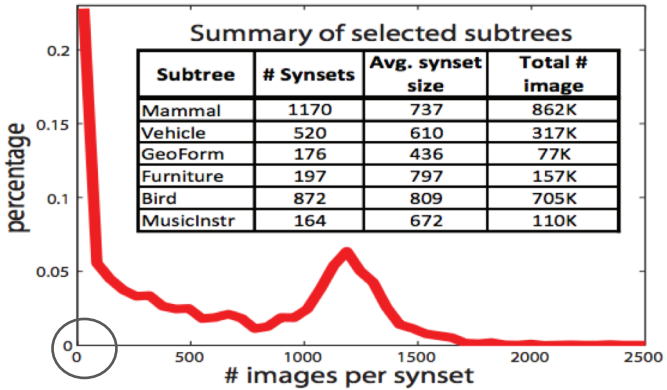
- 75.000 non-abstract nouns from WordNet*, some of which are *rare*





Summary of selected subtrees

Subtree	# Synsets	Avg. synset size	Total # image
Mammal	1170	737	862K
Vehicle	520	610	317K
GeoForm	176	436	77K
Furniture	197	797	157K
Bird	872	809	705K
MusicInstr	164	672	110K



* Torralba, et al. 2008.

How many training examples do we need?

- … plus object combinations, scenes

A tennis player hitting a ball

Fork with spaghetti

Wedding car

- It is not feasible to collect several fully annotated samples per "class"
- (... and *categorization* is a questionable paradigm)

Learning with Incomplete Supervision

- The main goal: minimize the data collection and/or annotation effort.

- Many exciting research directions:
  - Unsupervised representation learning
  - Self-supervised learning (extract pseudo-supervision from data)
  - Semi-supervised learning (unsupervised + supervised)
  - Weakly-supervised localization (training images with labels only)
  - Zero-shot learning (novel classes based on auxiliary knowledge)
  - Few-shot learning (learning from few examples)
  - Partially-supervised learning (learning from few examples)
  - ....

Learning with Incomplete Supervision

- The main goal: minimize the data collection and/or annotation effort.

- Many exciting research directions:
  - Unsupervised representation learning
  - Self-supervised learning (extract pseudo-supervision from data)
  - Semi-supervised learning (unsupervised + supervised)
  - Weakly-supervised localization (training images with labels only)
  - Zero-shot learning (novel classes based on auxiliary knowledge)
  - Few-shot learning (learning from few examples)
  - Partially-supervised learning (learning from few examples)
  - ....

Learning with Incomplete Supervision

- The main goal: minimize the data collection and/or annotation effort.

- Many exciting research directions:
  - Unsupervised representation learning
  - Self-supervised learning (extract pseudo-supervision from data)
  - Semi-supervised learning (unsupervised + supervised)
  - Weakly-supervised localization (training images with labels only)
  - Zero-shot learning (novel classes based on auxiliary knowledge)
  - Few-shot learning (learning from few examples)
  - Partially-supervised learning (learning from few examples)
  - ....

Talk overview

SESSION 1

Lecture

Modern deep generative models
for unsupervised learning

For more material:
METU CENG 796
Deep Generative Models

Talk overview

SESSION 1

Lecture

Modern deep generative models for unsupervised learning

For more material:
METU CENG 796
Deep Generative Models

SESSION 2

Research

- Zero-shot learning with generative & discriminative models
- Partially supervised domain transfer
- Q/A

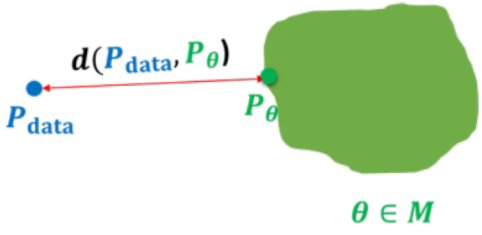
Session I

An overview of contemporary generative models

Learning a generative model

- We are given a training set of examples, e.g., images of dogs

$x_i \sim P_{\text{data}}$
$i = 1, 2, \ldots, n$



- We want to learn a probability distribution $p(x)$ over images x such that
  1. **Generation:** If we sample $x_{new} \sim p(x)$, x_{new} should look like a dog (*sampling*)
  2. **Density estimation:** $p(x)$ should be high if x looks like a dog, and low otherwise (*anomaly detection*)
  3. **Unsupervised representation learning:** We should be able to learn what these images have in common, e.g., ears, tail, etc. (*features*)
- First question: how to represent $p(x)$. Second question: how to learn it.

Slide by Stefano Ermon, Aditya Grover

Coverage

Auto-regressive models

$$p_\theta(\mathbf{x}) = \prod_{i=1}^{N} p_\theta(x_i|\mathbf{x}_{<i})$$

Variational latent models

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}, \mathbf{z}) \mathrm{d}\mathbf{z}$$

Normalizing flow models

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

GANs

$$\min_\theta \max_\phi V(G_\theta, D_\phi) = \mathbb{E}_{\mathbf{x} \sim \mathbf{p}_{\text{data}}}[\log D_\phi(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})}[\log(1 - D_\phi(G_\theta(\mathbf{z})))]$$

Slide by Emre Akbas

Auto-regressive models

Recap: Bayesian networks vs neural models

- Using Chain Rule

$$p(x_1, x_2, x_3, x_4) = p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1, x_2)p(x_4 \mid x_1, x_2, x_3)$$

  Fully General, no assumptions needed (exponential size, no free lunch)

- Bayes Net

$$p(x_1, x_2, x_3, x_4) \approx p_{\text{CPT}}(x_1)p_{\text{CPT}}(x_2 \mid x_1)p_{\text{CPT}}(x_3 \mid \cancel{x_1}, x_2)p_{\text{CPT}}(x_4 \mid x_1, \cancel{x_2, x_3})$$

  Assumes conditional independencies; tabular representations via conditional probability tables (CPT)

- Neural Models

$$p(x_1, x_2, x_3, x_4) \approx p(x_1)p(x_2 \mid x_1)p_{\text{Neural}}(x_3 \mid x_1, x_2)p_{\text{Neural}}(x_4 \mid x_1, x_2, x_3)$$

  Assumes specific functional form for the conditionals. A sufficiently deep neural net can approximate any function.

Slide by Stefano Ermon, Aditya Grover

Motivating Example: MNIST

- **Given:** a dataset $\mathcal{D}$ of handwritten digits (binarized MNIST)

- Each image has $n = 28 \times 28 = 784$ pixels. Each pixel can either be black (0) or white (1).
- **Goal:** Learn a probability distribution $p(x) = p(x_1, \cdots, x_{784})$ over $x \in \{0, 1\}^{784}$ such that when $x \sim p(x)$, x looks like a digit
- Two step process:
  1. Parameterize a model family $\{p_\theta(x), \theta \in \Theta\}$
  2. Search for model parameters θ based on training data $\mathcal{D}$

Autoregressive Models

- We can pick an ordering of all the random variables, i.e., raster scan ordering of pixels from top-left (X_1) to bottom-right ($X_{n=784}$)
- Without loss of generality, we can use chain rule for factorization

$$p(x_1, \cdots, x_{784}) = p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1, x_2) \cdots p(x_n \mid x_1, \cdots, x_{n-1})$$

- Some conditionals are too complex to be stored in tabular form. Instead, we **assume**

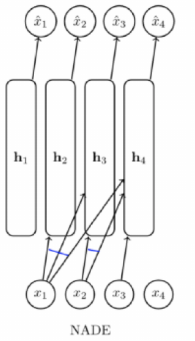
$$p(x_1, \cdots, x_{784}) = p_{\text{CPT}}(x_1; \alpha^1)p_{\text{logit}}(x_2 \mid x_1; \boldsymbol{\alpha}^2)p_{\text{logit}}(x_3 \mid x_1, x_2; \boldsymbol{\alpha}^3) \cdots p_{\text{logit}}(x_n \mid x_1, \cdots, x_{n-1}; \boldsymbol{\alpha}^n)$$

- More explicitly
  - $p_{\text{CPT}}(X_1 = 1; \alpha^1) = \alpha^1$, $p(X_1 = 0) = 1 - \alpha^1$
  - $p_{\text{logit}}(X_2 = 1 \mid x_1; \boldsymbol{\alpha}^2) = \sigma(\boldsymbol{\alpha}_0^2 + \boldsymbol{\alpha}_1^2 x_1)$
  - $p_{\text{logit}}(X_3 = 1 \mid x_1, x_2; \boldsymbol{\alpha}^3) = \sigma(\boldsymbol{\alpha}_0^3 + \boldsymbol{\alpha}_1^3 x_1 + \boldsymbol{\alpha}_2^3 x_2)$
- Note: This is a **modeling assumption**. We are using parameterized functions (e.g., logistic regression above) to predict next pixel given all the previous ones. Called **autoregressive** model.

Slide by Stefano Ermon, Aditya Grover

NADE: Neural Autoregressive Density Estimation

Hugo Larochelle, Iain Murray
The Neural Autoregressive Distribution Estimator
AISTATS 2011



NADE

- To improve model: use one layer neural network instead of logistic regression

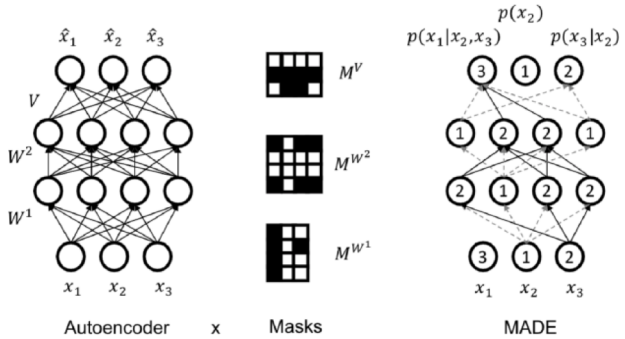
$$\mathbf{h}_i = \sigma(A_i \mathbf{x}_{<i} + \mathbf{c}_i)$$

$$\hat{x}_i = p(x_i | x_1, \cdots, x_{i-1}; \underbrace{A_i, \mathbf{c}_i, \alpha_i, b_i}_{\text{parameters}}) = \sigma(\alpha_i \mathbf{h}_i + b_i)$$

- For example $\mathbf{h}_2 = \sigma\left(\underbrace{\begin{pmatrix}\vdots\end{pmatrix}}_{A_2} x_1 + \underbrace{\begin{pmatrix}\vdots\end{pmatrix}}_{c_2}\right)$ $\mathbf{h}_3 = \sigma\left(\underbrace{\begin{pmatrix}\vdots & \vdots\end{pmatrix}}_{A_3} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \underbrace{\begin{pmatrix}\vdots\end{pmatrix}}_{c_3}\right)$

Slide by Stefano Ermon, Aditya Grover

MADE: Masked Autoencoder for Distribution Estimation

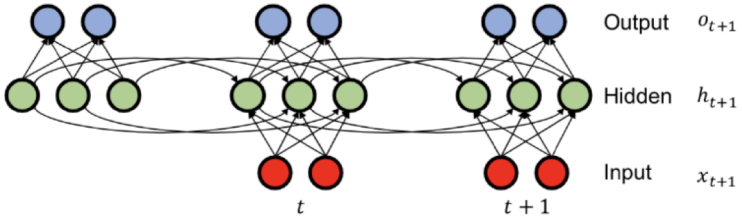


1. **Challenge**: An autoencoder that is autoregressive (DAG structure)
2. **Solution**: use masks to disallow certain paths (Germain et al., 2015). Suppose ordering is x_2, x_3, x_1.
   1. The unit producing the parameters for $p(x_2)$ is not allowed to depend on any input. Unit for $p(x_3|x_2)$ only on x_2. And so on...
   2. For each unit in a hidden layer, pick a random integer i in $[1, n-1]$. That unit is allowed to depend only on the first i inputs (according to the chosen ordering).
   3. Add mask to preserve this invariant: connect to all units in previous layer with smaller or equal assigned number (strictly $<$ in final layer)

Slide by Stefano Ermon, Aditya Grover

RNN: Recurrent Neural Nets

Challenge: model $p(x_t|x_{1:t-1}; \alpha^t)$. "History" $x_{1:t-1}$ keeps getting longer.
Idea: keep a summary and recursively update it

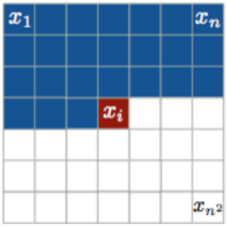


$$\begin{aligned} \text{Summary update rule: } h_{t+1} &= tanh(W_{hh}h_t + W_{xh}x_{t+1}) \\ \text{Prediction: } o_{t+1} &= W_{hy}h_{t+1} \\ \text{Summary initalization: } h_0 &= \boldsymbol{b}_0 \end{aligned}$$

1. Hidden layer h_t is a summary of the inputs seen till time t
2. Output layer o_{t-1} specifies parameters for conditional $p(x_t \mid x_{1:t-1})$
3. Parameterized by $\boldsymbol{b}_0$ (initialization), and matrices W_{hh}, W_{xh}, W_{hy}. Constant number of parameters w.r.t n!

Slide by Stefano Ermon, Aditya Grover

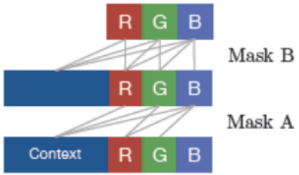
Pixel RNN (Oord et al., 2016)



1. Model images pixel by pixel using raster scan order
2. Each pixel conditional $p(x_t \mid x_{1:t-1})$ needs to specify 3 colors

$$p(x_t \mid x_{1:t-1}) = p(x_t^{red} \mid x_{1:t-1}) p(x_t^{green} \mid x_{1:t-1}, x_t^{red}) p(x_t^{blue} \mid x_{1:t-1}, x_t^{red}, x_t^{green})$$

   and each conditional is a categorical random variable with 256 possible values
3. Conditionals modeled using RNN variants. LSTMs + masking (like MADE)

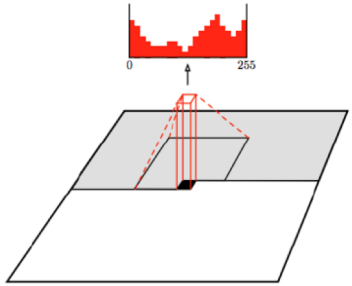


Slide by Stefano Ermon, Aditya Grover

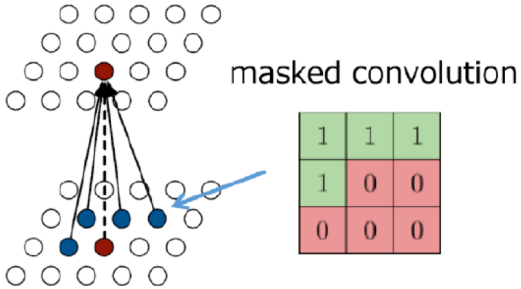
Pixel RNN

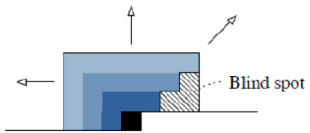
Results on downsampled ImageNet. Very slow: sequential likelihood evaluation.

PixelCNN (Oord et al., 2016)



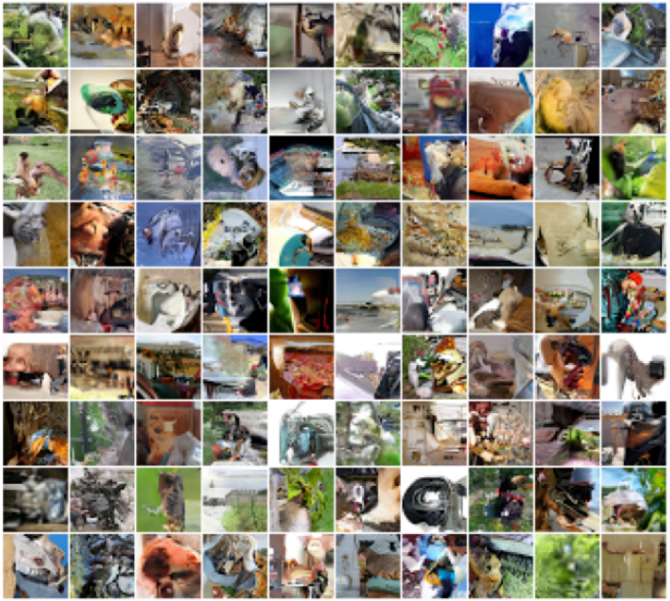
Idea: Use convolutional architecture to predict next pixel given context (a neighborhood of pixels).
Challenge: Has to be autoregressive. Masked convolutions preserve raster scan order. Additional masking for colors order.





Slide by Stefano Ermon, Aditya Grover

PixelCNN



Samples from the model trained on Imagenet (32×32 pixels). Similar performance to PixelRNN, but much faster.

Slide by Stefano Ermon, Aditya Grover

Summary of the auto-regressive models

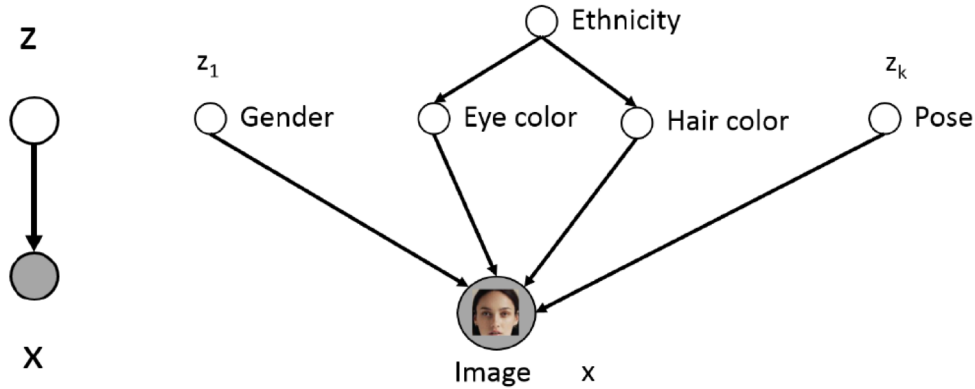
1. Autoregressive models:
   - Chain rule based factorization is fully general
   - Compact representation via *conditional independence* and/or *neural parameterizations*
2. Autoregressive models Pros:
   - Easy to evaluate likelihoods
   - Easy to train
3. Autoregressive models Cons:
   - Requires an ordering
   - Generation is sequential
   - Cannot learn features in an unsupervised way

Latent variable models

Latent Variable Models: Motivation

1. Lots of variability in images **x** due to gender, eye color, hair color, pose, etc. However, unless images are annotated, these factors of variation are not explicitly available (latent).
2. **Idea**: explicitly model these factors using latent variables **z**

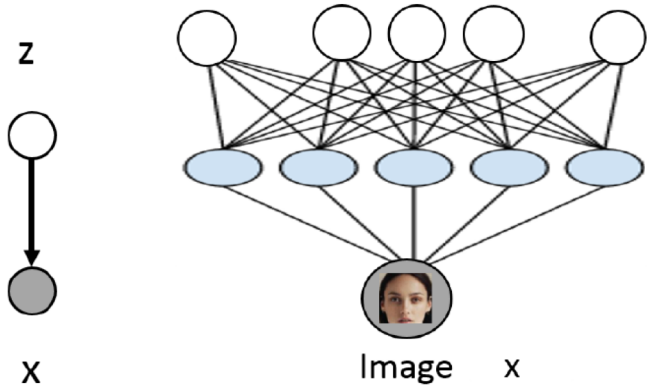
Latent Variable Models: Motivation



1. Only shaded variables $\mathbf{x}$ are observed in the data (pixel values)
2. Latent variables $\mathbf{z}$ correspond to high level features
   - If $\mathbf{z}$ chosen properly, $p(\mathbf{x}|\mathbf{z})$ could be much simpler than $p(\mathbf{x})$
   - If we had trained this model, then we could identify features via $p(\mathbf{z} \mid \mathbf{x})$, e.g., $p(EyeColor = Blue|\mathbf{x})$
3. **Challenge:** Very difficult to specify these conditionals by hand

Slide by Stefano Ermon, Aditya Grover

Deep Latent Variable Models

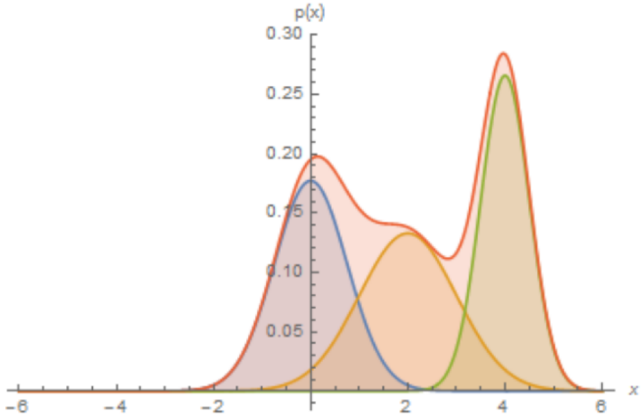


1. $\mathbf{z} \sim \mathcal{N}(0, I)$
2. $p(\mathbf{x} \mid \mathbf{z}) = \mathcal{N}\left(\mu_\theta(\mathbf{z}), \Sigma_\theta(\mathbf{z})\right)$ where μ_θ,Σ_θ are neural networks
3. *Hope* that after training, $\mathbf{z}$ will correspond to meaningful latent factors of variation (*features*). Unsupervised representation learning.
4. As before, features can be computed via $p(\mathbf{z} \mid \mathbf{x})$

Slide by Stefano Ermon, Aditya Grover

Mixture models

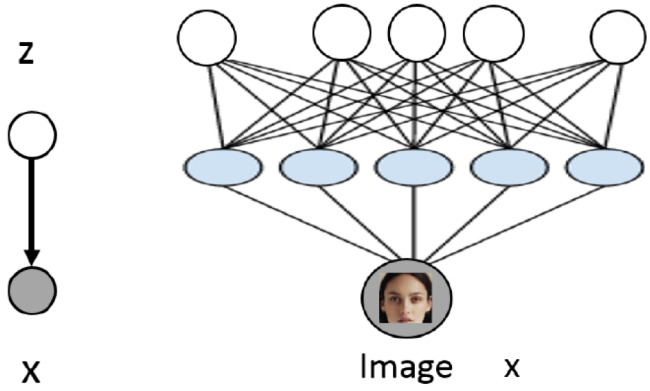
Combine simple models into a more complex and expressive one



$$p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{x}, \mathbf{z}) = \sum_{\mathbf{z}} p(\mathbf{z}) p(\mathbf{x} \mid \mathbf{z}) = \sum_{k=1}^{K} p(\mathbf{z} = k) \underbrace{\mathcal{N}(\mathbf{x}; \mu_k, \Sigma_k)}_{\text{component}}$$

Slide by Stefano Ermon, Aditya Grover

Variational Autoencoder



A mixture of an infinite number of Gaussians:

1. $\mathbf{z} \sim \mathcal{N}(0, I)$
2. $p(\mathbf{x} \mid \mathbf{z}) = \mathcal{N}\left(\mu_\theta(\mathbf{z}), \Sigma_\theta(\mathbf{z})\right)$ where μ_θ,Σ_θ are neural networks
3. Even though $p(\mathbf{x} \mid \mathbf{z})$ is simple, the marginal $p(\mathbf{x})$ is very complex/flexible

Slide by Stefano Ermon, Aditya Grover

Variational Autoencoder Marginal Likelihood

A mixture of an infinite number of Gaussians:

1. $\mathbf{z} \sim \mathcal{N}(0, I)$
2. $p(\mathbf{x} \mid \mathbf{z}) = \mathcal{N}\left(\mu_\theta(\mathbf{z}), \Sigma_\theta(\mathbf{z})\right)$ where μ_θ,Σ_θ are neural networks
3. $\mathbf{Z}$ are unobserved at train time (also called hidden or latent)
4. Suppose we have a model for the joint distribution. What is the probability $p(\mathbf{X} = \bar{\mathbf{x}}; \theta)$ of observing a training data point $\bar{\mathbf{x}}$?

$$\int_{\mathbf{z}} p(\mathbf{X} = \bar{\mathbf{x}}, \mathbf{Z} = \mathbf{z}; \theta) d\mathbf{z} = \int_{\mathbf{z}} p(\bar{\mathbf{x}}, \mathbf{z}; \theta) d\mathbf{z}$$

Slide by Stefano Ermon, Aditya Grover

First attempt: Naive Monte Carlo

Likelihood function $p_\theta(\mathbf{x})$ for Partially Observed Data is hard to compute:

$$p_\theta(\mathbf{x}) = \sum_{\text{All values of } \mathbf{z}} p_\theta(\mathbf{x}, \mathbf{z}) = |\mathcal{Z}| \sum_{\mathbf{z} \in \mathcal{Z}} \frac{1}{|\mathcal{Z}|} p_\theta(\mathbf{x}, \mathbf{z}) = |\mathcal{Z}| \mathbb{E}_{\mathbf{z} \sim Uniform(\mathcal{Z})} [p_\theta(\mathbf{x}, \mathbf{z})]$$

We can think of it as an (intractable) expectation. Monte Carlo to the rescue:

1. Sample $\mathbf{z}^{(1)}, \cdots, \mathbf{z}^{(k)}$ uniformly at random
2. Approximate expectation with sample average

$$\sum_{\mathbf{z}} p_\theta(\mathbf{x}, \mathbf{z}) \approx |\mathcal{Z}| \frac{1}{k} \sum_{j=1}^{k} p_\theta(\mathbf{x}, \mathbf{z}^{(j)})$$

Works in theory but not in practice. For most $\mathbf{z}$, $p_\theta(\mathbf{x}, \mathbf{z})$ is very low (most completions don't make sense). Some are very large but will never "hit" likely completions by uniform random sampling. Need a clever way to select $\mathbf{z}^{(j)}$ to reduce variance of the estimator.

Slide by Stefano Ermon, Aditya Grover

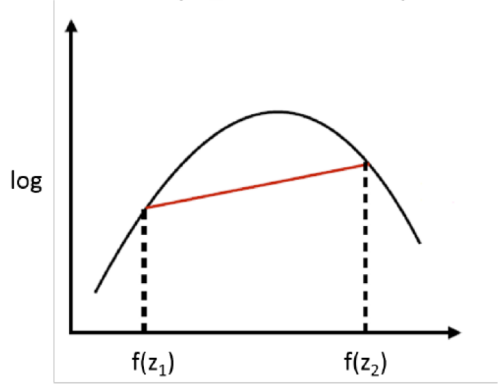
Evidence Lower Bound

Log-Likelihood function for Partially Observed Data is hard to compute:

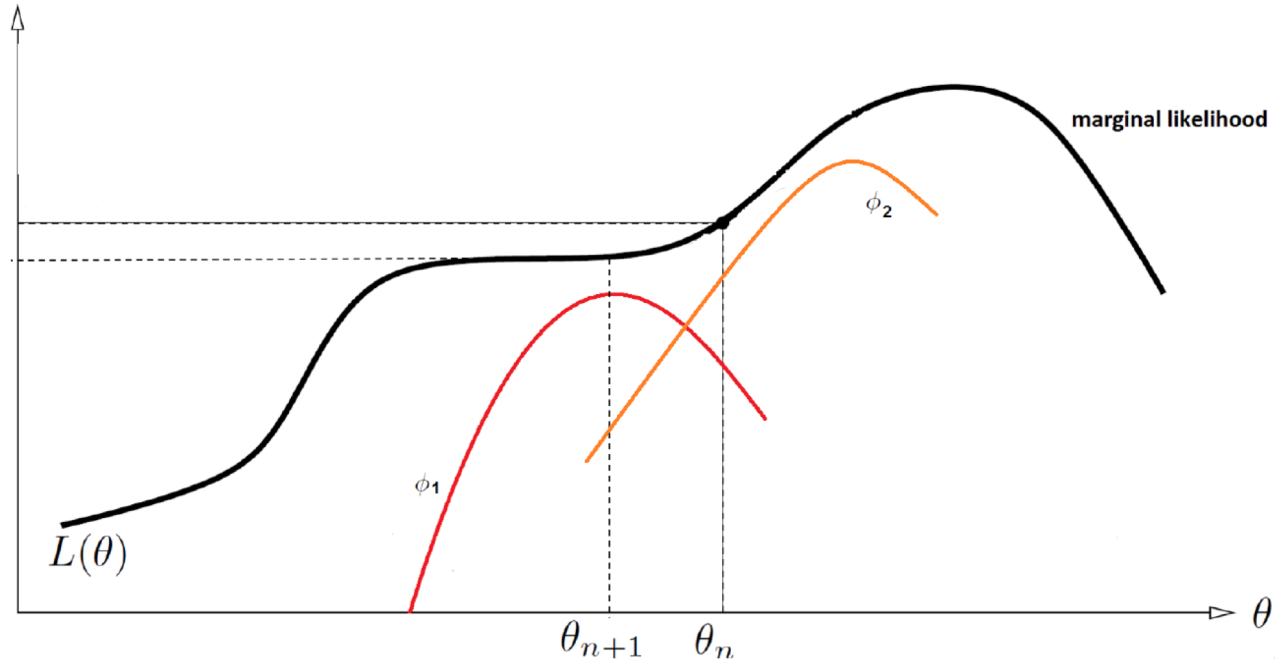
$$\log\left(\sum_{\mathbf{z}\in\mathcal{Z}} p_\theta(\mathbf{x},\mathbf{z})\right) = \log\left(\sum_{\mathbf{z}\in\mathcal{Z}} \frac{q(\mathbf{z})}{q(\mathbf{z})} p_\theta(\mathbf{x},\mathbf{z})\right) = \log\left(\mathbb{E}_{\mathbf{z}\sim q(\mathbf{z})}\left[\frac{p_\theta(\mathbf{x},\mathbf{z})}{q(\mathbf{z})}\right]\right)$$

- log() is a concave function. $\log(px + (1-p)x') \geq p\log(x) + (1-p)\log(x')$.
- Idea: use Jensen Inequality (for concave functions)

$$\log\left(\mathbb{E}_{\mathbf{z}\sim q(\mathbf{z})}\left[f(\mathbf{z})\right]\right) = \log\left(\sum_{\mathbf{z}} q(\mathbf{z}) f(\mathbf{z})\right) \geq \sum_{\mathbf{z}} q(\mathbf{z}) \log f(\mathbf{z})$$



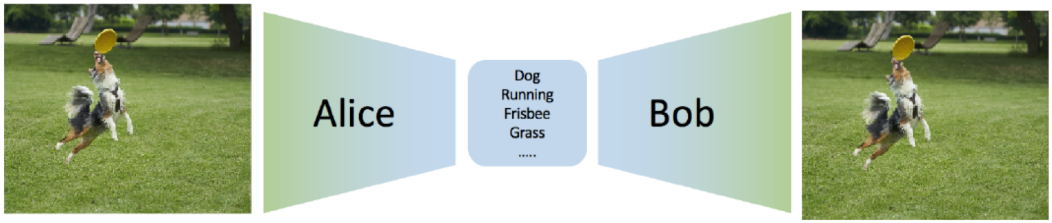
Variational learning



and $\mathcal{L}(\mathbf{x}; \theta, \phi_2)$ are both lower bounds. We want to jointly optimize θ and ϕ

Slide by Stefano Ermon, Aditya Grover

Learning Deep Generative models



1. Alice goes on a space mission and needs to send images to Bob. Given an image $\mathbf{x}^i$, she (stochastically) compresses it using $\hat{\mathbf{z}} \sim q_\phi(\mathbf{z}|\mathbf{x}^i)$ obtaining a message $\hat{\mathbf{z}}$. Alice sends the message $\hat{\mathbf{z}}$ to Bob
2. Given $\hat{\mathbf{z}}$, Bob tries to reconstruct the image using $p(\mathbf{x}|\hat{\mathbf{z}};\theta)$

- This scheme works well if $E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x}|\mathbf{z};\theta)]$ is large
- The term $D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))$ forces the distribution over messages to have a specific shape $p(\mathbf{z})$. If Bob knows $p(\mathbf{z})$, he can generate realistic messages $\hat{\mathbf{z}} \sim p(\mathbf{z})$ and the corresponding image, as if he had received them from Alice!

Slide by Stefano Ermon, Aditya Grover

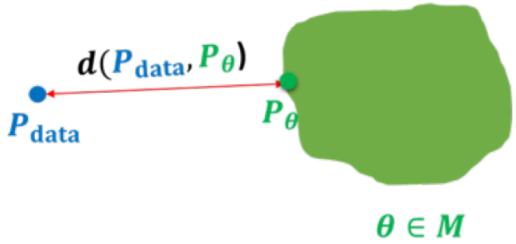
Summary

- Latent Variable Models Pros:
  - Easy to build flexible models
  - Suitable for unsupervised learning
- Latent Variable Models Cons:
  - Hard to evaluate likelihoods
  - Hard to train via maximum-likelihood
  - Fundamentally, the challenge is that posterior inference $p(\mathbf{z} \mid \mathbf{x})$ is hard. Typically requires variational approximations
- Alternative: give up on KL-divergence and likelihood (GANs)

Normalizing Flows

Recap of likelihood-based learning so far:

$x_i \sim P_{\text{data}}$
$i = 1, 2, \ldots, n$



- 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!

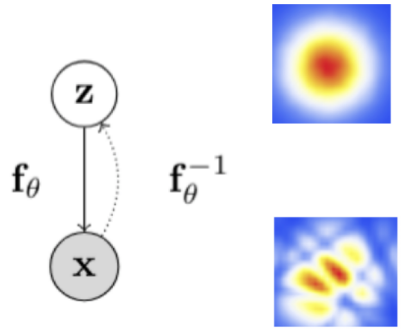
Slide by Stefano Ermon, Aditya Grover

Simple Prior to Complex Data Distributions

- Desirable properties of any model distribution:
  - Analytic density
  - Easy-to-sample
- Many simple distributions satisfy the above properties e.g., Gaussian, uniform distributions
- Unfortunately, data distributions could be much more complex (multi-modal)
- **Key idea**: Map simple distributions (easy to sample and evaluate densities) to complex distributions (learned via data) using **change of variables**.

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 $\mathbf{f}_\theta : \mathbb{R}^n \mapsto \mathbb{R}^n$, is deterministic and invertible such that $X = \mathbf{f}_\theta(Z)$ and $Z = \mathbf{f}_\theta^{-1}(X)$



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

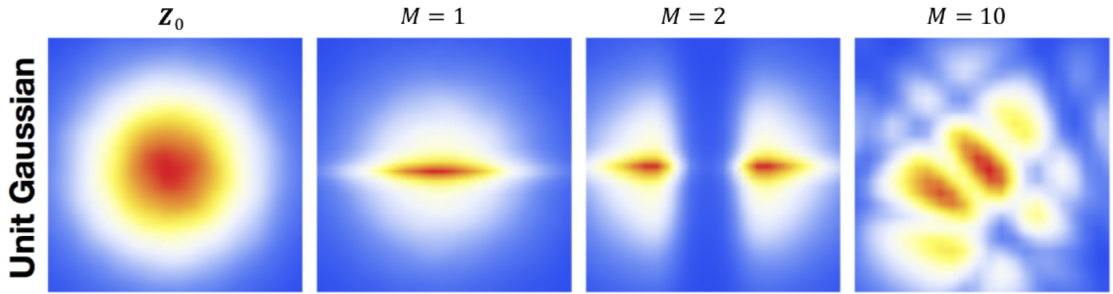
$$p_X(\mathbf{x};\theta) = p_Z\left(\mathbf{f}_\theta^{-1}(\mathbf{x})\right)\left|\det\left(\frac{\partial \mathbf{f}_\theta^{-1}(\mathbf{x})}{\partial \mathbf{x}}\right)\right|$$

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

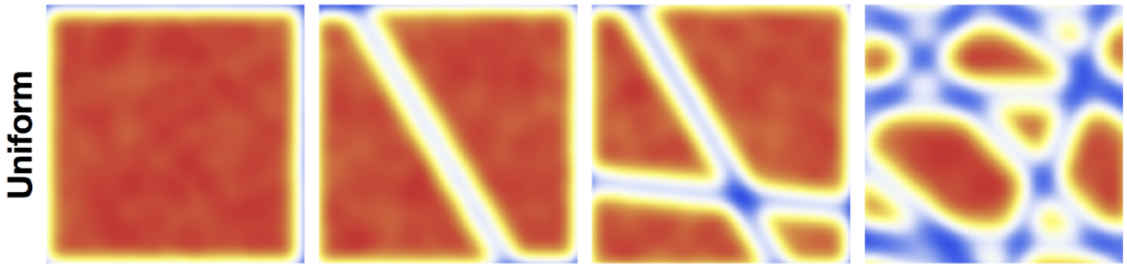
Slide by Stefano Ermon, Aditya Grover

Planar flows (Rezende & Mohamed, 2016)

- Base distribution: Gaussian



- Base distribution: Uniform



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

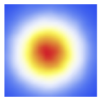
Slide by Stefano Ermon, Aditya Grover

Learning and Inference

- Learning via **maximum likelihood** over the dataset $\mathcal{D}$

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

- **Exact likelihood evaluation** via inverse tranformation $\mathbf{x} \mapsto \mathbf{z}$ and change of variables formula
- **Sampling** via forward transformation $\mathbf{z} \mapsto \mathbf{x}$



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

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

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

Slide by Stefano Ermon, Aditya Grover

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)

Summary of Normalizing Flow Models

- Transform simple distributions into more complex distributions via change of variables
- Jacobian of transformations should have tractable determinant for efficient learning and density estimation
- Computational tradeoffs in evaluating forward and inverse transformations

Generative Adversarial Networks

So far…

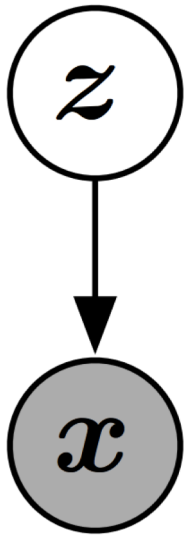
- Autoregressive models
- Normalizing Flows
- Latent Variable Models, VAEs

- All these models are likelihood-based.

- Now GANs: focus on sample quality, directly.

Implicit Models

- Sample z from a fixed noise source distribution (uniform or gaussian).
- Pass the noise through a deep neural network to obtain a sample x.
- Sounds familiar? Right:
  - Flow Models
  - VAE
- *What's going to be different here?*
  - *Learning the deep neural network **without** explicit density estimation*



Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

Motivation for GANs

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

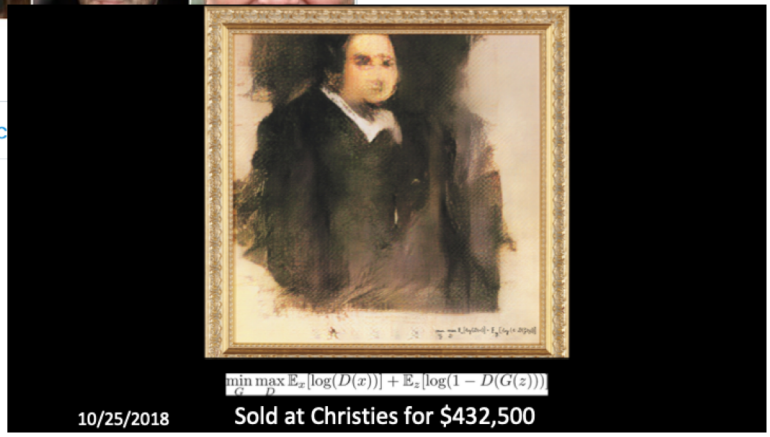
Motivation for GANs

[BigGAN, Brock, Donahue, Simonyan, 2018]

Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

Motivation for GANs

[BigGAN, Brock, Donahue, Simonyan, 2018]



Slide originally by Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas, Alex Li, Wilson Yan

Core GAN formulation

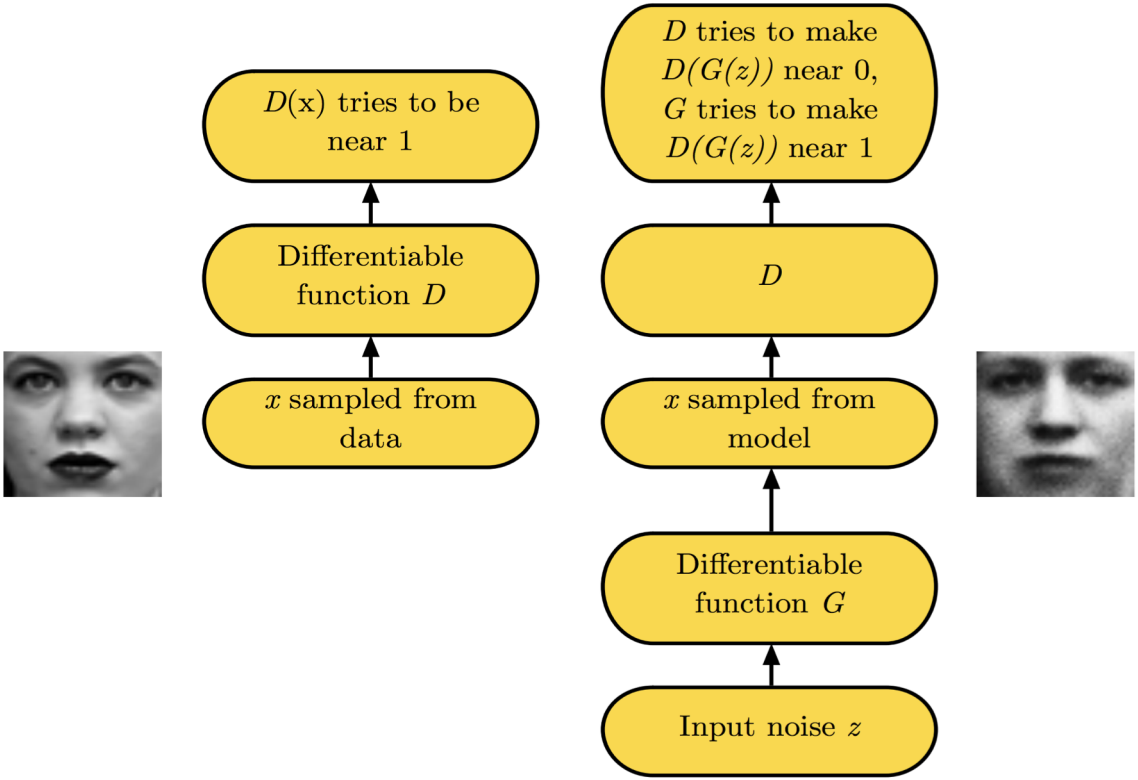


Figure from NeurIPS 2016 GAN Tutorial (Goodfellow)

What G() does (original) GAN optimize after all?

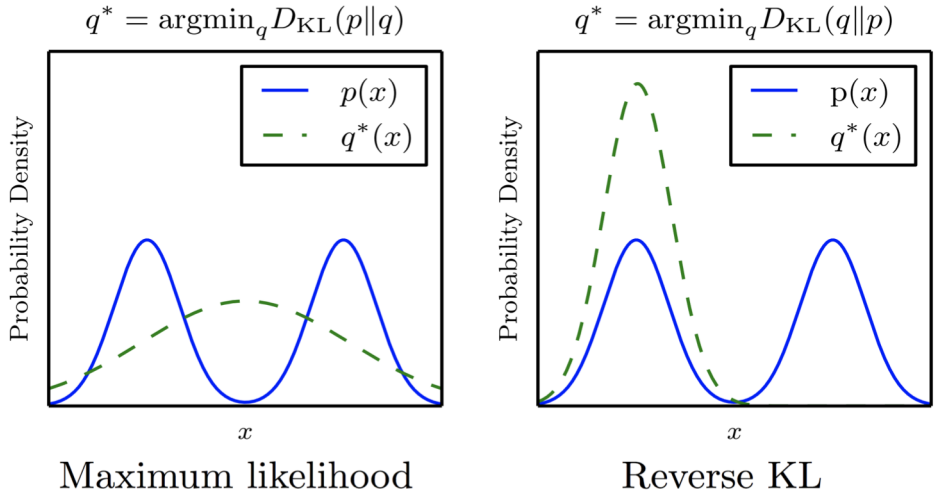
- Assume that D() is Bayes-optimal, then:

$$
\begin{aligned}
V(G, D^*) &= \mathbb{E}_{x \sim p_{\text{data}}} \left[\log D^*(x) \right] + \mathbb{E}_{x \sim p_g} \left[\log(1 - D^*(x)) \right] \\
&= \mathbb{E}_{x \sim p_{\text{data}}} \left[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)} \right] + \mathbb{E}_{x \sim p_g} \left[\log \frac{p_g(x)}{p_{\text{data}}(x) + p_g(x)} \right] \\
&= -\log(4) + \underbrace{KL\left(p_{\text{data}} \| \left(\frac{p_{\text{data}} + p_g}{2} \right) \right) + KL\left(p_{\text{g}} \| \left(\frac{p_{\text{data}} + p_g}{2} \right) \right)}_{\text{(Jensen-Shannon Divergence (JSD) of } p_{\text{data}} \text{ and } p_g) \geq 0}
\end{aligned}
$$

What G() does (original) GAN optimize after all?

- Assume that D() is Bayes-optimal, then:

$$
\begin{aligned}
V(G, D^*) &= \mathbb{E}_{x \sim p_{\text{data}}} \left[\log D^*(x) \right] + \mathbb{E}_{x \sim p_g} \left[\log (1 - D^*(x)) \right] \\
&= \mathbb{E}_{x \sim p_{\text{data}}} \left[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)} \right] + \mathbb{E}_{x \sim p_g} \left[\log \frac{p_g(x)}{p_{\text{data}}(x) + p_g(x)} \right] \\
&= -\log(4) + \underbrace{KL\left(p_{\text{data}} \| \left(\frac{p_{\text{data}} + p_g}{2} \right) \right) + KL\left(p_{\text{g}} \| \left(\frac{p_{\text{data}} + p_g}{2} \right) \right)}_{\text{(Jensen-Shannon Divergence (JSD) of } p_{\text{data}} \text{ and } p_g) \geq 0}
\end{aligned}
$$



Advances in GANs

- Many, many applications (image manipulation, style transfer, guided image generation, etc.)

- Better architectures (eg, StyleGAN), better loss functions (eg, Wasserstein GAN), better regularization techniques (eg, Spectral Norm), better conditioning techniques (eg, Projection Discriminator)

StyleGAN v2
Karras et al 2019

Open problems in GANs

- Better modeling of complex scenes
- Improving training stability
- Improving sample quality
- Better use in representation-learning

Summary

- We have very briefly summarized: auto-regressive models, normalizing flows, VAEs, GANs
- There are several other important ones, such as Energy-based models, hybrid models, moment-matching networks, etc.
- The search for *more principled, more stable, representation-inferring, high-quality sample generating* models continues!

TESEKKURLER