Tutorial on Deep Generative Models

ADITYA GROVER AND STEFANO ERMON

STANFORD UNIVERSITY

IJCAI-ECAI 2018

URL: HTTPS://BIT.LY/2MHUFNR

Introduction

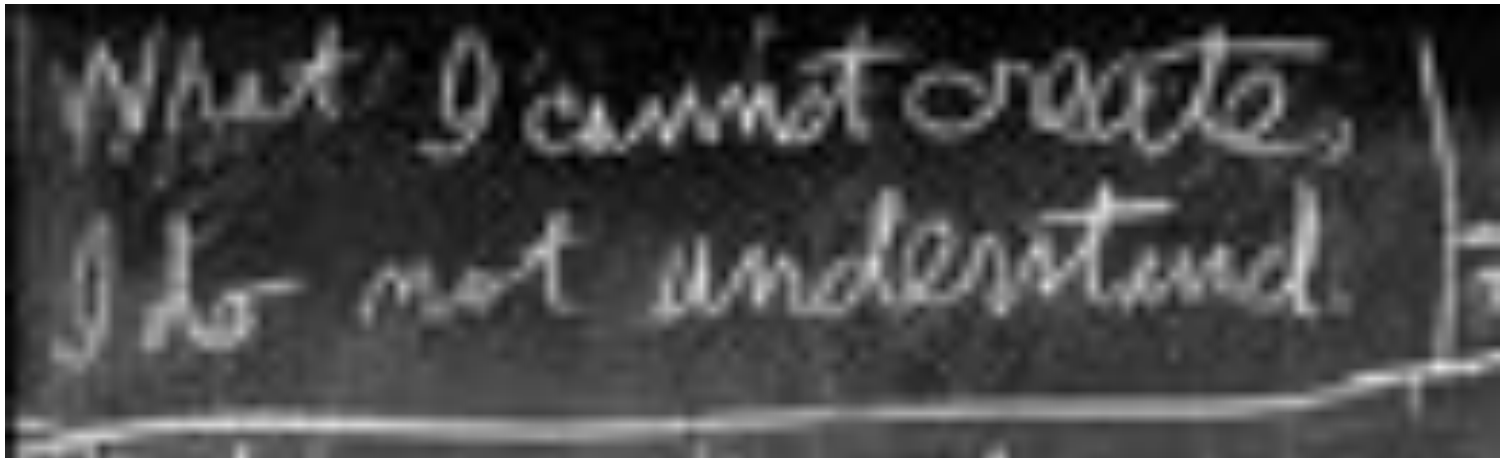
Computer Vision

Computational Speech

Natural Language Processing

Computer Vision / robotics

Introduction



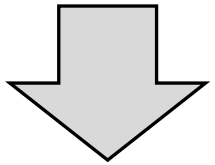
Richard Feynman: "*What I cannot create, I do not understand*“

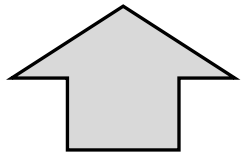
Assumption for today: "*What I understand, I can create*“

Generative Modeling: Computer Graphics

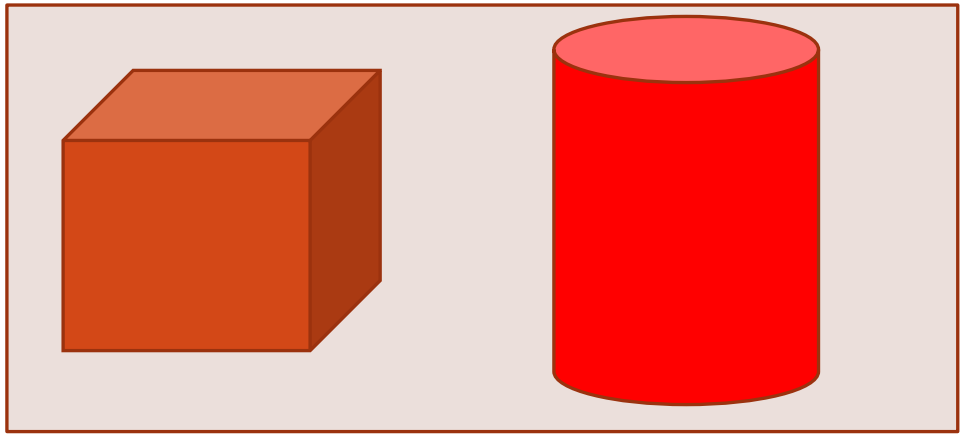
Cube (color=blue, position=, size, …)
Cylinder (color=red, position=, size,..)

Computer Graphics





Computer Vision



Statistical Generative Models

Data
(e.g., images of bedrooms)

+

Prior Knowledge
(e.g., physics, materials, ..)

Learning

Generative model

Data

Prior Knowledge

This tutorial

Graphics

Statistical Generative Models

Data

+

Model family, loss function, optimization algorithm, etc.

Learning

Prior Knowledge

Image x

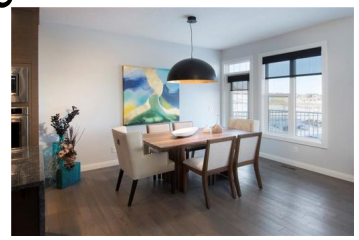
A probability distribution p(x)

probability p(x)

Sampling from p(x) **generates** new images:

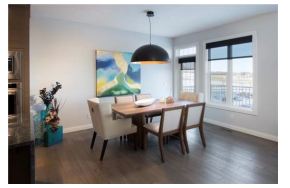
Discriminative vs. generative

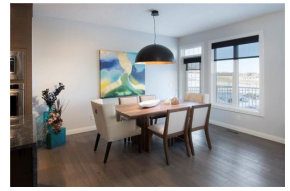
Decision boundary



$P(Y = \text{Bedroom} \mid X = \text{  }) = 0.01$

Example: logistic regression, convolutional net, etc.

Y=B , 

Y=D , 

... ...

Y=B , 

Y=D , 

$P(Y = \text{Bedroom}, X = \text{  })$

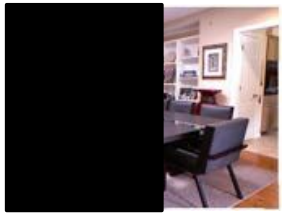
Model of the **joint**

Discriminative vs. generative

Joint and conditional are related by **Bayes Rule**:

$$P(Y = \text{Bedroom} \mid X = \text{  }) = \frac{P(Y = \text{Bedroom}, X = \text{  })}{P(X = \text{  })}$$

Discriminative: X is always given as input, doesn't need to model $P(X = \text{  })$

Cannot handle missing data $P(Y = \text{Bedroom} \mid X = \text{  })$

Conditional Generative Model

Class conditional models:

P(X=  | Y = Bedroom) P(X=  | Y = Dining Room)

P(X=  | Caption = “A black table with 6 chairs”)

A discriminative model is a very simple conditional generative model of Y:

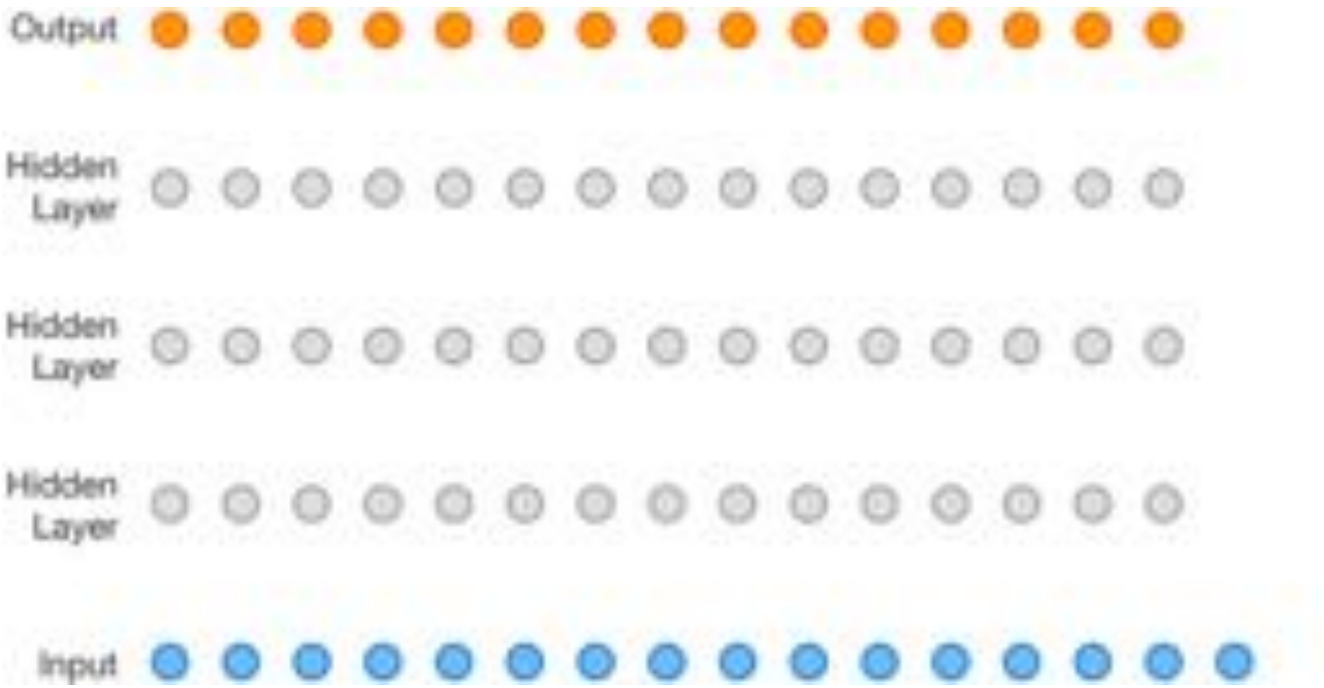
P(Y = Bedroom | X= )

Progressive Growing of GANs

Karras et al., 2018

WaveNet

Generative model of speech signals



Text to Speech

- Parametric
- Concatenative
- WaveNet

- Unconditional
- Music

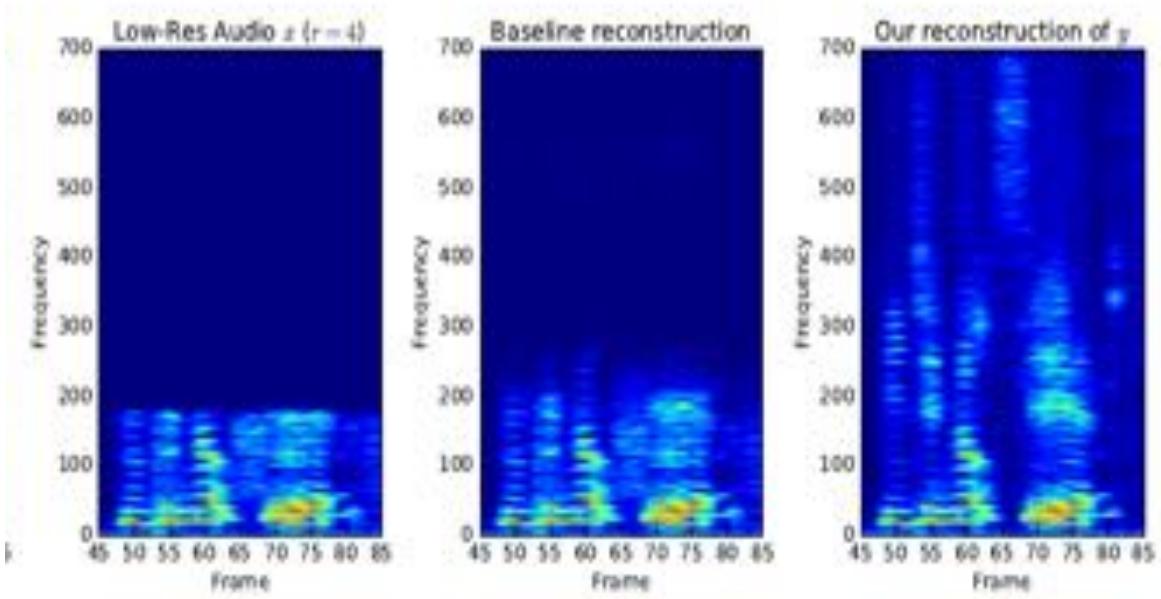
van den Oord et al, 2016

Image Super Resolution

Conditional generative model P(high res image | low res image)

Audio Super Resolution

Conditional generative model P(high-res signal | low-res audio signal)



Low res signal

High res audio signal

Kuleshov et al., 2017

Machine Translation

Conditional generative model P(English text| Chinese text)

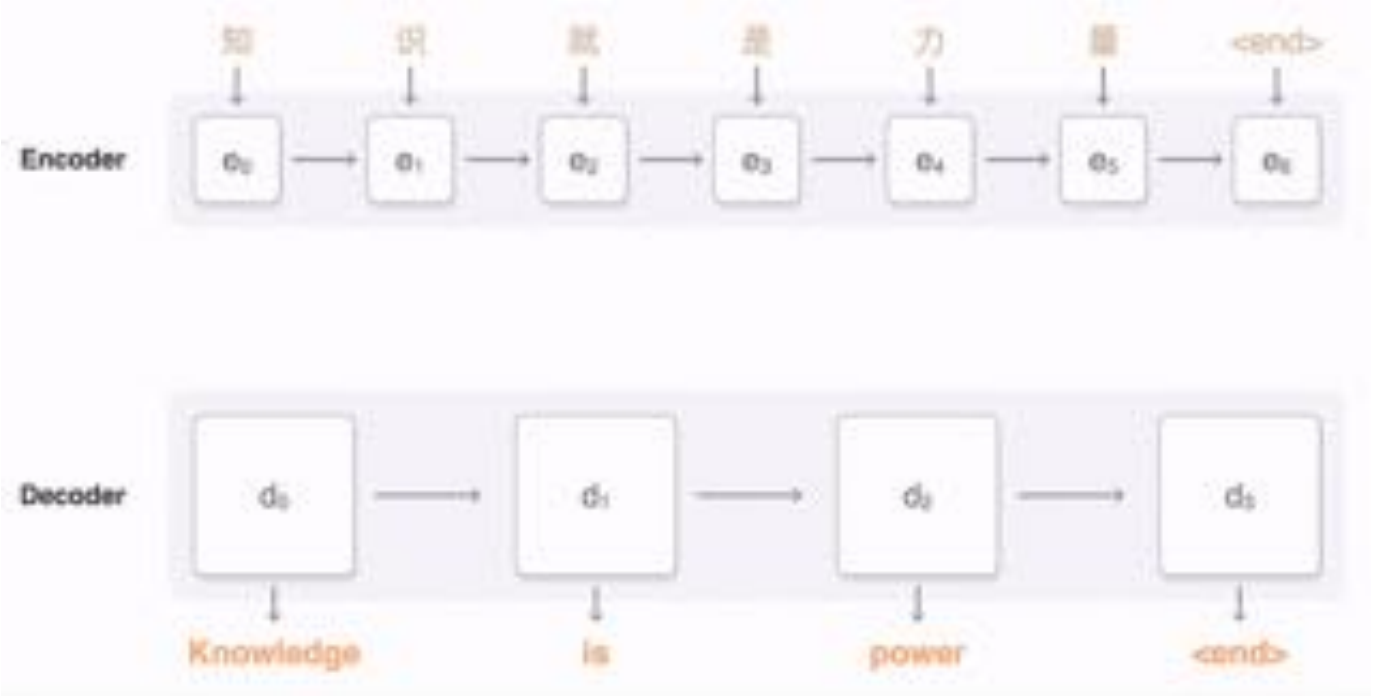


Figure from Google AI research blog.

Image Translation

Conditional generative model P(zebra images| horse images)

Zhu et al., 2017

Imitation Learning

Li et al., 2017

Roadmap

Introduction

Background and Taxonomy

Statistical Generative Models

Model family, loss function, optimization algorithm, etc.

Data

Learn

Generative model

$p(x)$

Desirable properties:

1. Sampling
2. Evaluating p(x)
3. Extracting features

as **efficiently as possible**

Representation - compactness

Suppose x_1,x_2,x_3 are binary variables. $P(x_1,x_2,x_3)$ can be specified with $(2^3-1)=7$ parameters

- P(0,0,0), P(0,0,1), …, P(1,1,0), ~~P(1,1,1)~~ = 1 – (P(0,0,0) + .. P(1,1,0))

Main challenge: **distributions over high dimensional objects**

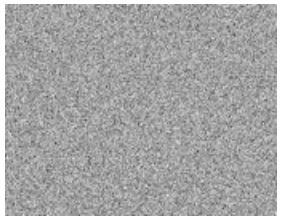
P(X= )

P(X= )

…

P(X= )

P(X= )

P(X= )

Too many possibilities!

N black/white pixels →

2^N parameters

Representation - factorization

Suppose x_1,x_2,x_3 are binary variables. $P(x_1,x_2,x_3)$ can be specified with $(2^3-1)=7$ parameters

- P(0,0,0), P(0,0,1), …, P(1,1,0), ~~P(1,1,1)~~

Definition of conditional probability: $P(x_1,x_2) = P(x_1)\,P(x_2|x_1)$

Chain rule: $P(x_1,x_2,x_3) = P(x_1)\,P(x_2|x_1)\,P(x_3|x_1,x_2)$

Chain rule: $P(x_1,x_2,x_3,x_4) = P(x_1)\,P(x_2|x_1)\,P(x_3|x_1,x_2)\,P(x_4|x_1,x_2,x_3)$

…

Main idea: **write as a product of simpler terms**

Representation - factorization

Suppose x_1,x_2,x_3 are binary variables. $P(x_1,x_2,x_3)$ can be specified with $(2^3-1)=7$ parameters

- P(0,0,0), P(0,0,1), …, P(1,1,0), ~~P(1,1,1)~~

Definition of conditional probability: $P(x_1,x_2) = P(x_1)\,P(x_2|x_1)$

Chain rule: $P(x_1,x_2,x_3) = P(x_1)\,P(x_2|x_1)\,P(x_3|x_1,x_2)$

Chain rule: $P(x_1,x_2,x_3,x_4) = P(x_1)\,P(x_2|x_1)\,P(x_3|x_1,x_2)\,P(x_4|x_1,x_2,x_3)$

Simple! ☺

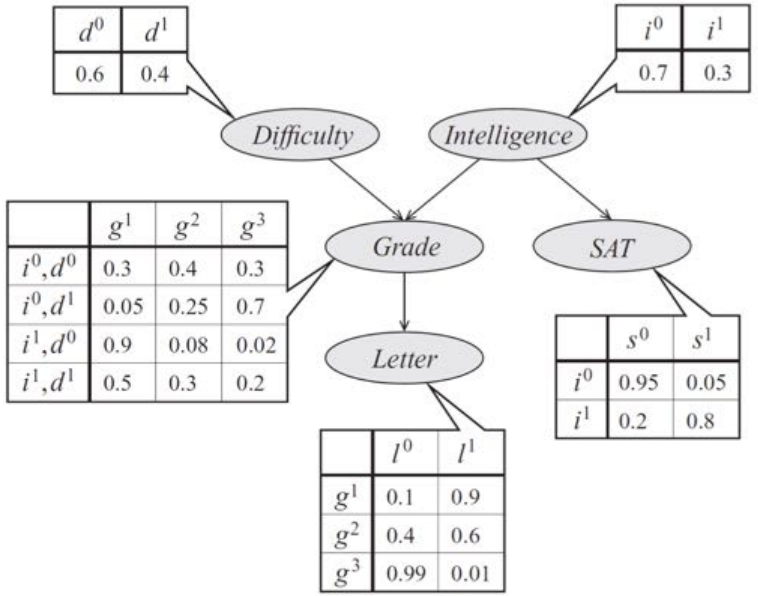
Still complex! ☹
Needs a distribution for **all possible values** of x_1,x_2,x_3

Representation – Bayes Nets

Chain rule: $P(x_1,x_2,x_3,x_4) = P(x_1)\,P(x_2|x_1)\,P(x_3|x_1,x_2)\,P(x_4|x_1,x_2,x_3)$

Solution #1 : simply the conditionals (**Bayes Nets**)

E.g., $P(x_1,x_2,x_3,x_4) \approx P(x_1)\,P(x_2|x_1)\,P(x_3|\cancel{x_1},x_2)\,P(x_4|\cancel{x_1},\cancel{x_2},x_3) = P(x_1)\,P(x_2|x_1)\,P(x_3|x_2)\,P(x_4|x_3)$

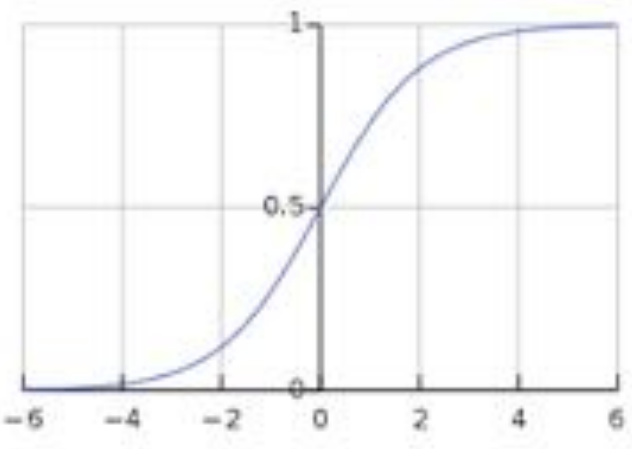


Representation – Deep Generative Models

Chain rule: $P(x_1,x_2,x_3,x_4) = P(x_1)\, P(x_2|x_1)\, P(x_3|x_1,x_2)\, P(x_4|x_1,x_2,x_3)$

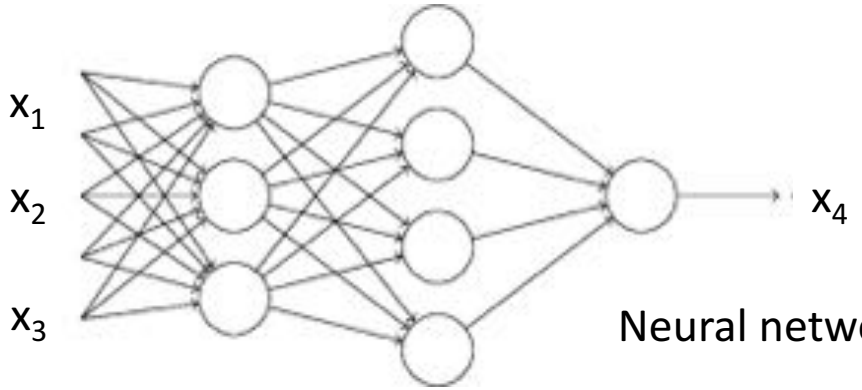
Solution #2a: use simple **functional form** for the conditionals

- $P(x_4|x_1,x_2,x_3) \approx \text{sigmoid}\,(a*x_1 + b*x_2 + c*x_3)$
- Only requires storing 3 parameters
- Relationship between x_4 and (x_1,x_2,x_3) could be too simple



Solution #2a: more complex functional form

- **More flexible**
- **More parameters**



Neural network

Properties of factorized models

Chain rule: $P(x_1,x_2,x_3,x_4) = P(x_1)\,P(x_2|x_1)\,P(x_3|x_1,x_2)\,P(x_4|x_1,x_2,x_3)$

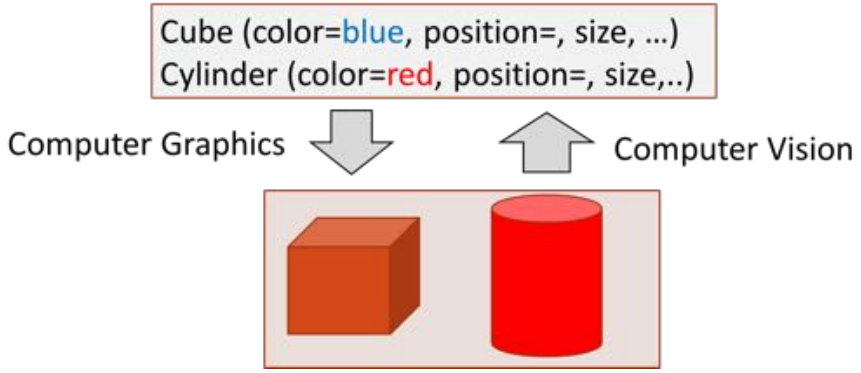
Easy to sample from☺ :

- Sample $x_1 \sim P(x_1)$
- Sample $x_2 \sim P(x_2|x_1)$
- Sample $x_3 \sim P(x_3|x_1,x_2)$
- …

Easy to evaluate $P(x_1,x_2,x_3,x_4)$☺

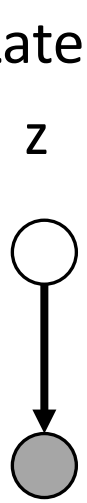
- Just multiply together the terms

Not obvious how to "**learn features**"/representations☹

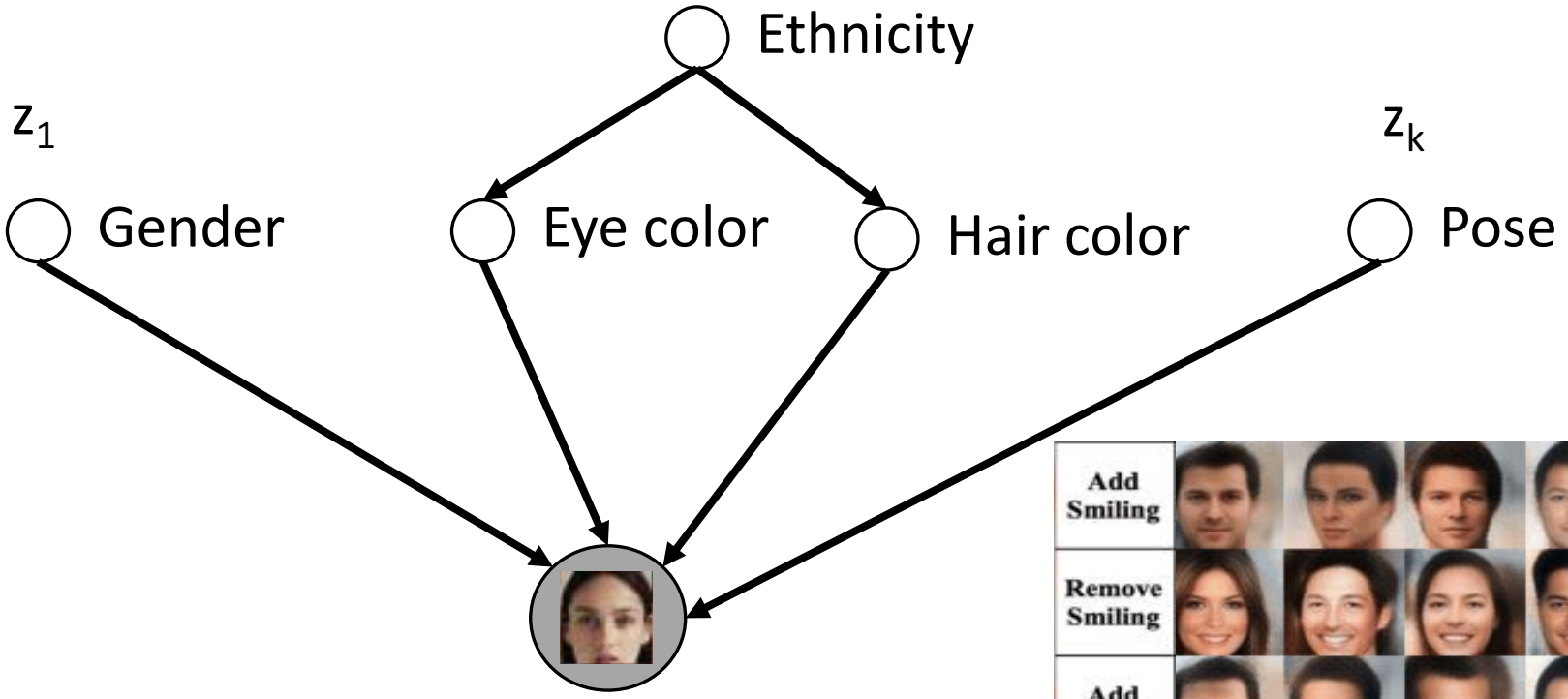


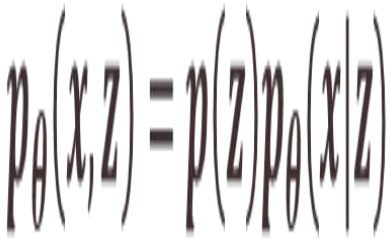
Latent variable models

Latent factors of variation



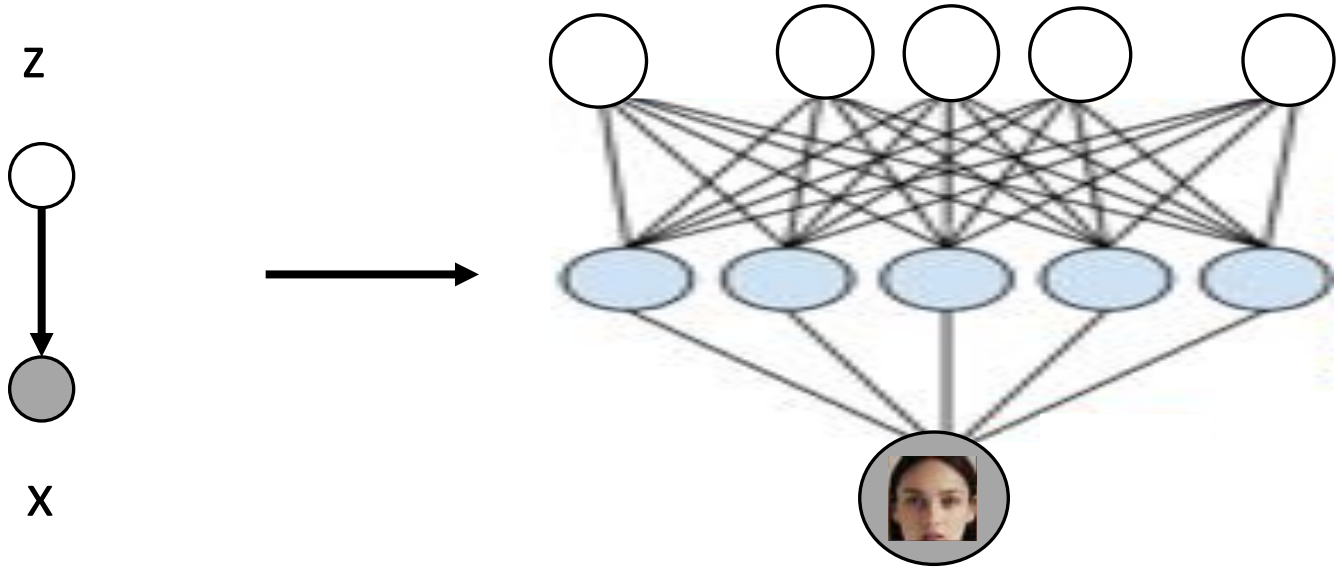
$$p_\theta(x, z) = p(z)p_\theta(x|z)$$



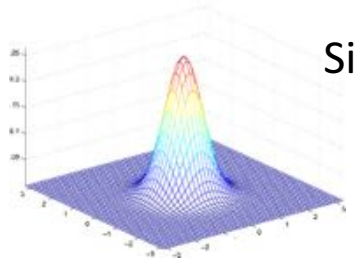


Hou el al., 2016

Deep Latent Variable Models



$p(z)$



Simple prior (Gaussian)

$p_\theta(x|z)$ Parameterized as a neural net

$p_\theta(x, z) = p(z)p_\theta(x|z)$

Image x $\log p_\theta(x) = \log \int p_\theta(x, z)\, dz$

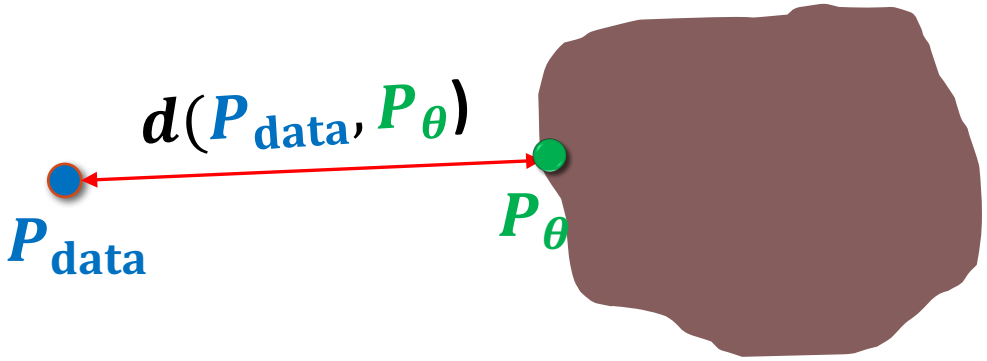
1) Easy to sample from ☺
2) P(x) intractable ☹
3) Enables feature learning ☺

Learning in Generative Models

Given: Samples from a data distribution and **model family**

Goal: Approximate a data distribution as **closely** as possible

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



$$\theta \in M$$

Model family

Learning in Generative Models

Given: Samples from a data distribution

Goal: Approximate a data distribution as **closely** as possible

$$\min_{\theta \in M} d(P_{\text{data}}, P_\theta)$$

Challenge: How should we evaluate and optimize **closeness** between the **data distribution** and the **model distribution**?

Maximum likelihood estimation

Given: Samples from a data distribution

Goal: Approximate a data distribution as **closely** as possible

Solution 1: d = KL divergence

$$\min_{\theta \in M} \mathbb{E}_{x \sim P_{data}} [-\log p_\theta (x)]$$

- Statistically efficient
- Requires the ability to tractably evaluate or optimize likelihoods

Maximum likelihood estimation

Tractable likelihoods: Directed models such as autoregressive models

Bayes Net: $P_\theta(x_1,x_2,x_3,x_4) = P_\theta(x_1)\, P_\theta(x_2|x_1)\, P_\theta(x_3|x_1,x_2)\, P_\theta(x_4|x_3)$

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

$$\min_{\theta \in M} \mathbb{E}_{x \sim P_{\text{data}}} [-\log p_\theta(x)]$$

$$P_{\text{data}}(x_1) P_{\text{data}}(x_2|x_1) P_{\text{data}}(x_3|x_1,x_2) P_{\text{data}}(x_4|x_3)$$

Analytic solution: pick θ so that conditional probabilities match the empirical ones, e.g., $P_\theta(x_1) = P_{\text{data}}(x_1)$

Maximum likelihood estimation

Tractable likelihoods: Directed models such as autoregressive models

Neural Net param.: $\boldsymbol{P_\theta}(x_1,x_2,x_3,x_4) = \boldsymbol{P_\theta}(x_1)\,\boldsymbol{P_\theta}(x_2|x_1)\,\boldsymbol{P_\theta}(x_3|x_1,x_2)\,\boldsymbol{P_\theta}(x_4|x_3)$

where $\boldsymbol{P_\theta}(x_3|x_1,x_2) \approx$ Neural-Net$(x_1, x_2, \boldsymbol{\theta})$

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

$$\min_{\boldsymbol{\theta} \in \boldsymbol{M}} \mathbb{E}_{x \sim \boldsymbol{P}_{\text{data}}} [-\log \boldsymbol{p_\theta}(x)]$$

Approximately solve for $\boldsymbol{\theta}$ using **gradient descent:**

- Gradients computed by **backprograpagation**
- **Stochastic approximations (minibatch)**
- **Non convex,** but works well in practice

Maximum likelihood estimation

Tractable likelihoods: Directed models such as autoregressive models
Intractable likelihoods: Undirected models such as restricted Boltzmann machines (RBM), directed models such as variational autoencoders (VAE)

Alternatives for intractable likelihoods:

- **Approximate inference** using MCMC or variational inference
- **Likelihood-free inference** using adversarial training

Adversarial Training

Given: Samples from a data distribution

Goal: Approximate a data distribution as **closely** as possible

Solution 2: **Two sample test**

d = **integral probability metric** or **f-div** lower bound

For e.g., if the **f-div = Jenson-Shannon**:

$$\min_{\theta \in M} \max_{\phi \in F} \mathbb{E}_{x \sim P_\theta}[\log(1 - D_\phi(x))] - \mathbb{E}_{x \sim P_{data}} \log[D_\phi(x)]$$

Likelihood-free! Requires samples from P_θ

Generative Adversarial Networks

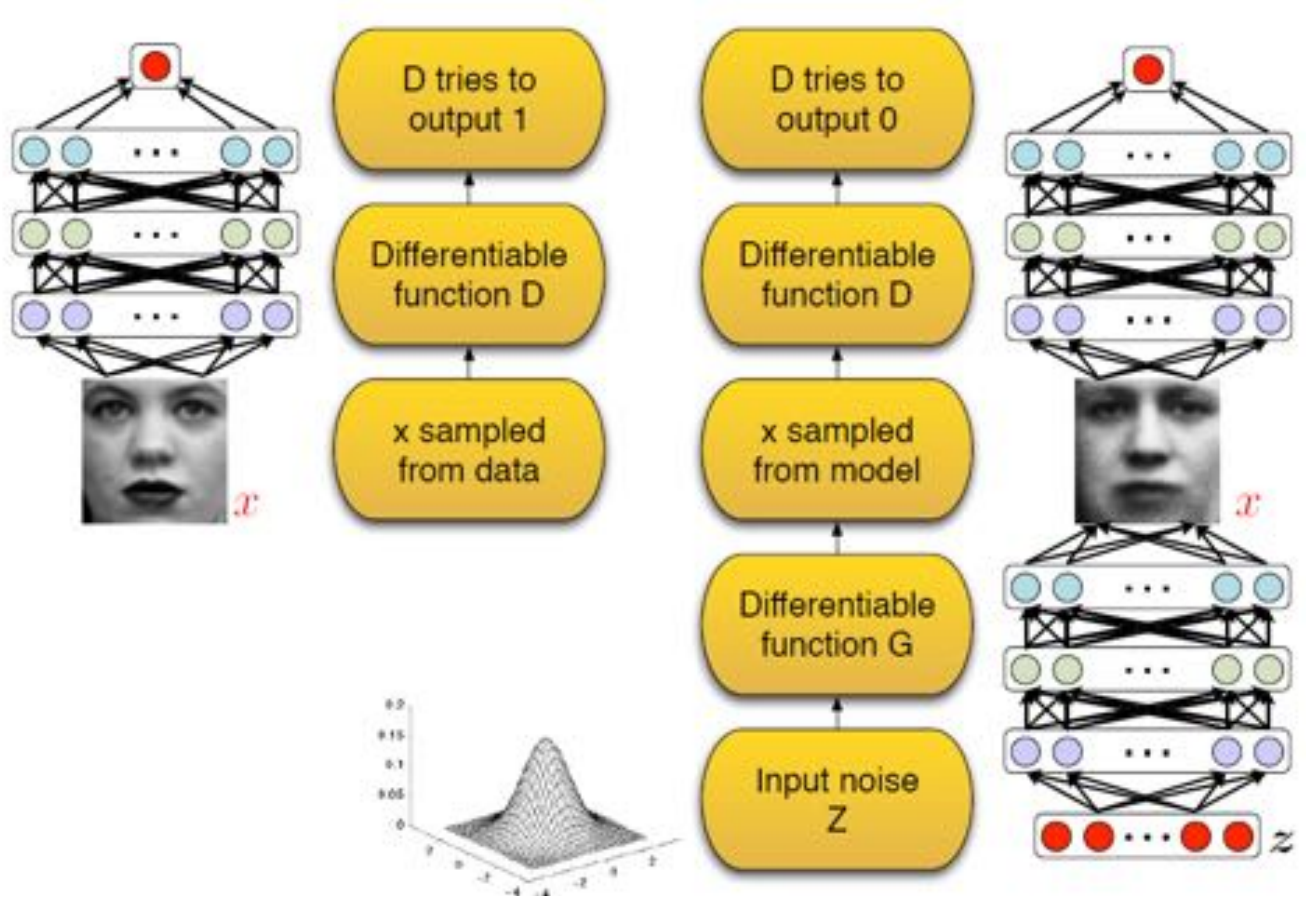


Figure from
Goodfellow et al., 2014

Roadmap

Introduction

Background and Taxonomy

Likelihood-based Generative Models

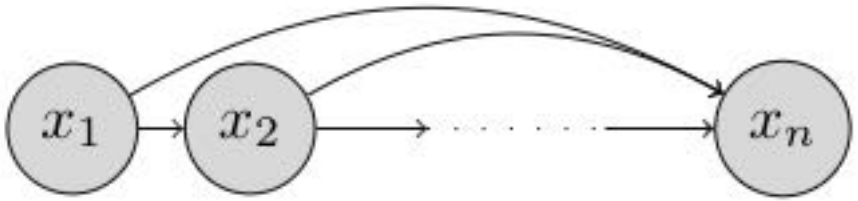
- **Autoregressive Models**

Likelihood-based Generative Models

- Provide an analytical expression for the log-likelihood, i.e., $\log p_\theta(x)$
- Learning involves (approximate) evaluation of the gradients of the model log-likelihood with respect to the parameters θ
- **Key design choices**
  - Directed vs. undirected
  - Fully observed vs. latent variable

Autoregressive Generative Models

- Directed, fully-observed graphical models



- **Key idea:** Decompose the joint distribution as a **product of tractable conditionals**

$$p_\theta(\mathbf{x}) = \prod_{i=1}^{n} p_\theta(x_i|x_1, x_2, \ldots, x_{i-1}) = \prod_{i=1}^{n} p_\theta(x_i|x_{<i})$$

Learning and Inference

- **Learning** maximizes the model log-likelihood over the dataset $\mathcal{D}$

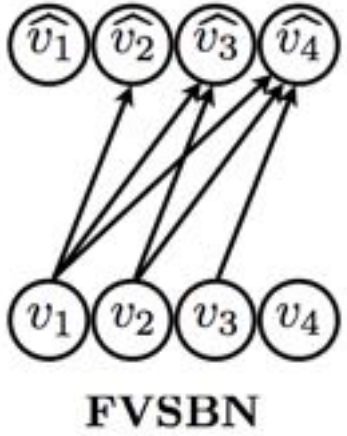
$$\max_{\theta} \log p_\theta(\mathcal{D}) = \sum_{\mathbf{x} \in \mathcal{D}} \log p_\theta(\mathbf{x}) = \sum_{\mathbf{x} \in \mathcal{D}} \sum_{i=1}^{n} \log p_\theta(x_i | x_{<i}))$$

- Tractable conditionals allow for **exact likelihood evaluation**
  - Conditional evaluated in parallel during training
- Directed model permits **ancestral sampling**, one variable at a time

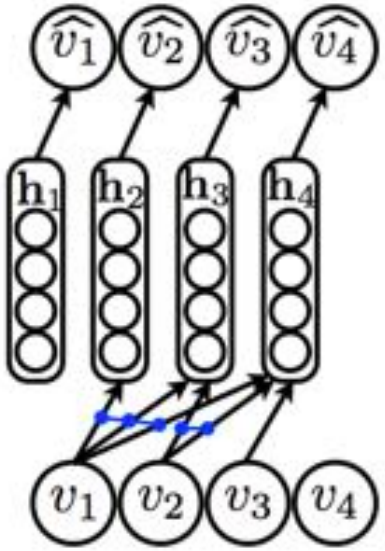
$$x_1 \sim p_\theta(x_1), x_2 \sim p_\theta(x_2|x_1), \ldots, x_n \sim p_\theta(x_n|x_{<n})$$

NN-based parameterizations

- Fully-visible sigmoid belief networks (FVSBN)
- Neural autoregressive density estimator (NADE): 1 hidden layer NN



FVSBN



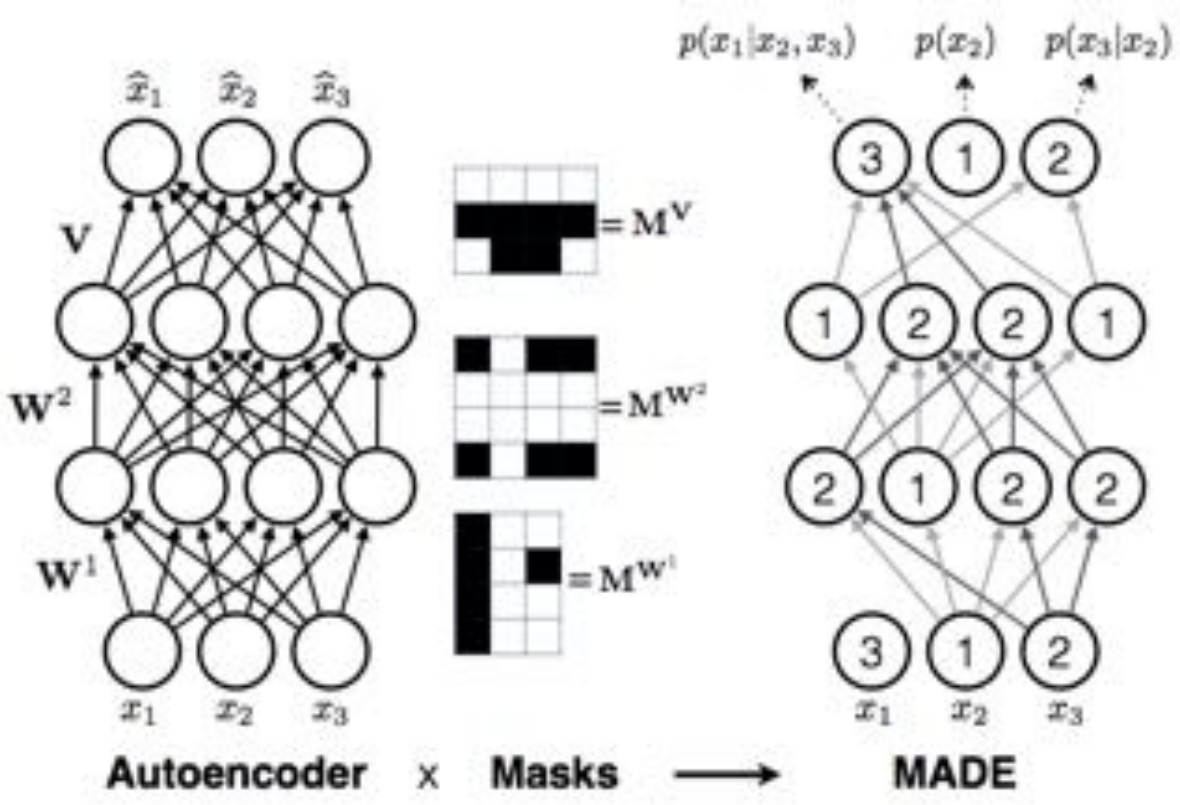
NADE

$$p_\theta(v_i|v_{<i}) = N(\mu(v_{<i}), \sigma(v_{<i}))$$

Neal, 1992, Larochelle & Murray, 2011

MLP-based parameterizations

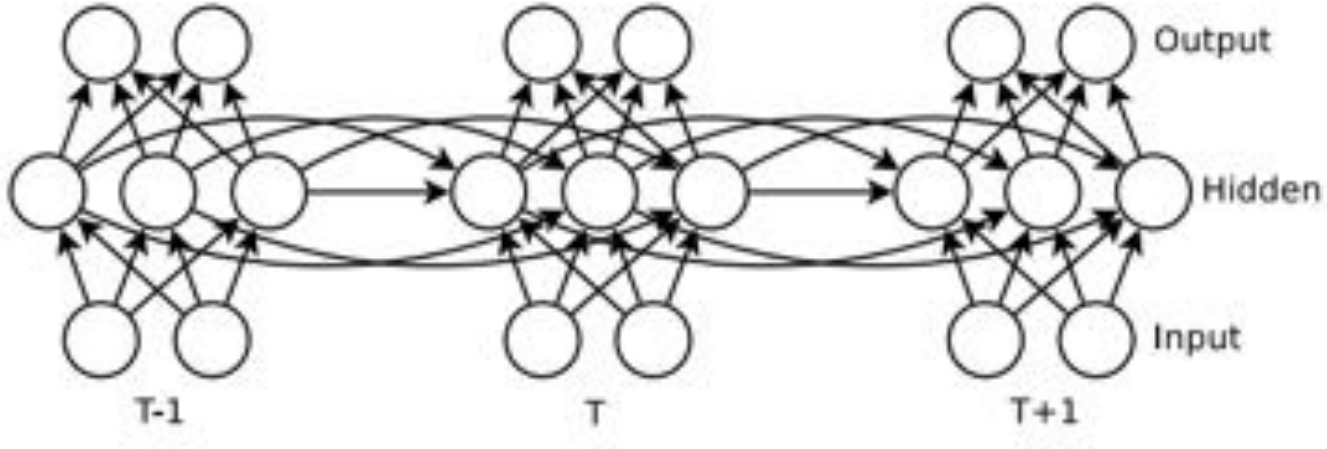
- Masked autoencoder distribution estimation (MADE): multilayer NN



Germain et al., 2015

RNN-based parameterizations

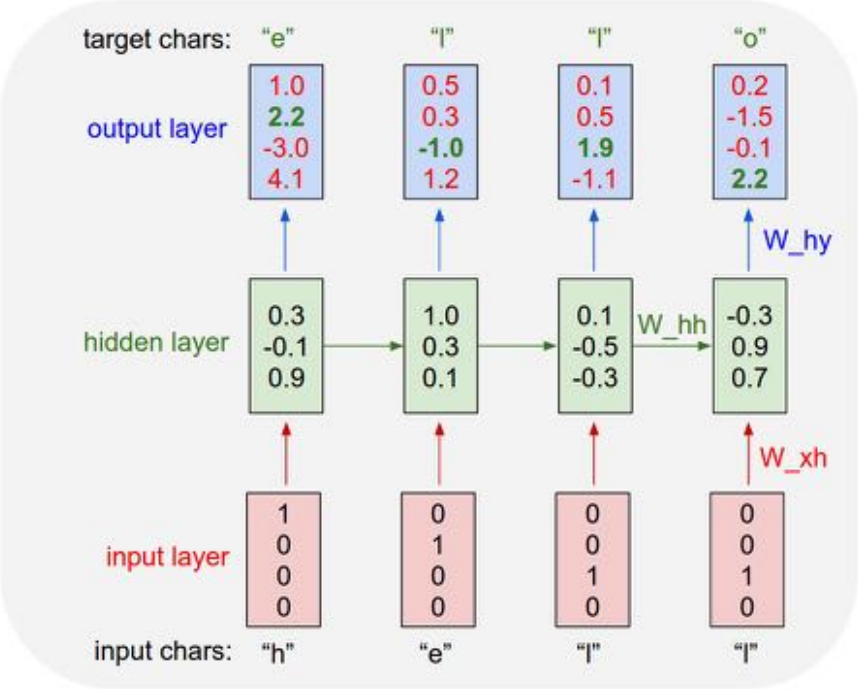
Effective architecture for learning long-range sequential dependencies



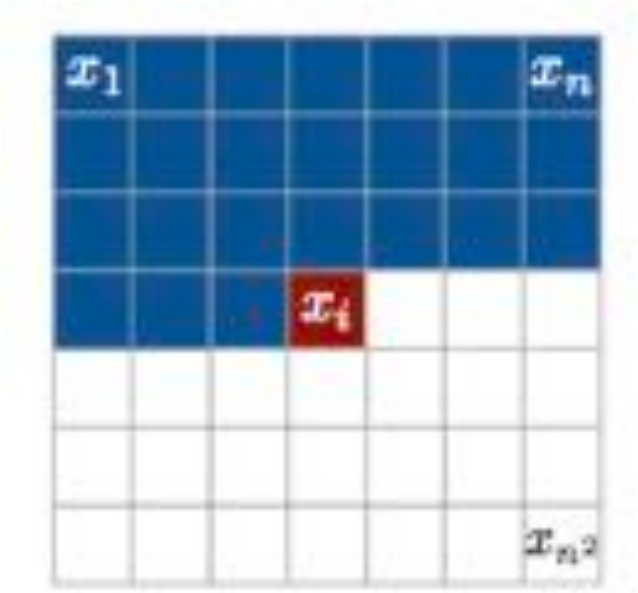
- Hidden layer $h_\theta(t)$ is a summary of the inputs seen till $t-1$
- Output layer $o_\theta(t)$ specifies parameters for conditionals $p_\theta(x_t \mid x_{<t})$

Sutskevar et al., 2011

RNN-based parameterizations



Char-RNN

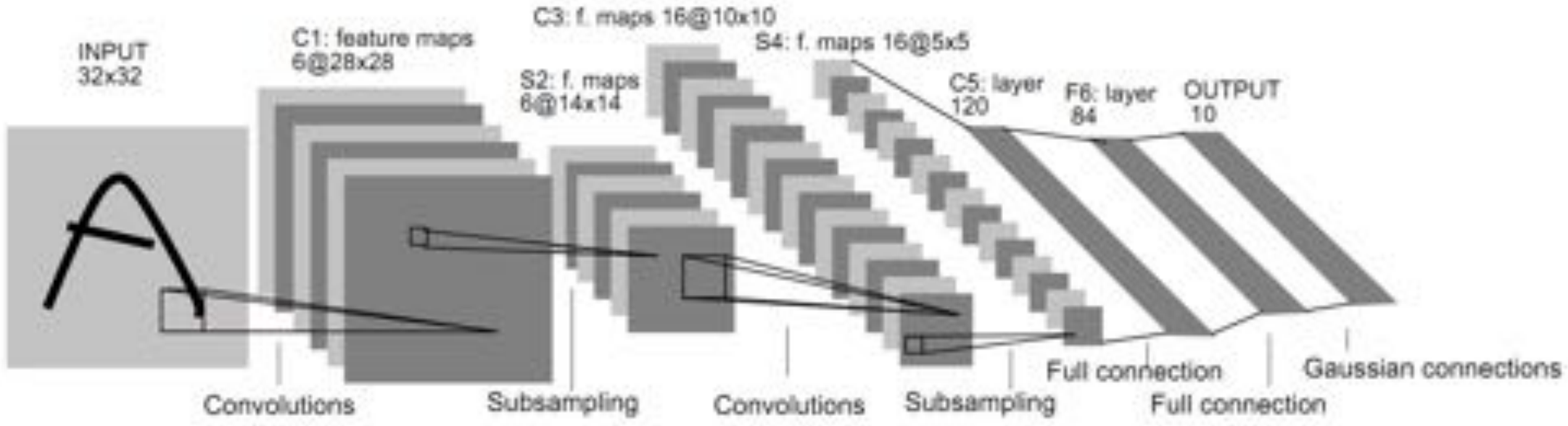


Pixel-RNN

Sutskevar et al., 2011, Karpathy, 2015, Theis & Bethge, 2015, van den Oord, 2016a

CNN-based parameterizations

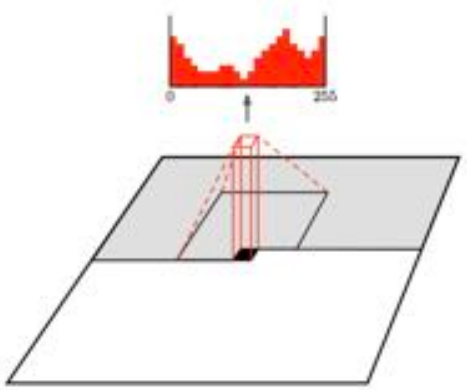
Convolutions are amenable for parallelization on modern hardware

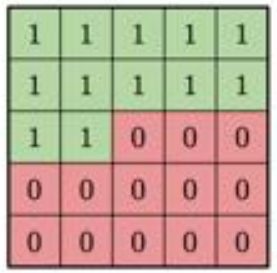


Lecunn et al., 1998

CNN-based parameterizations

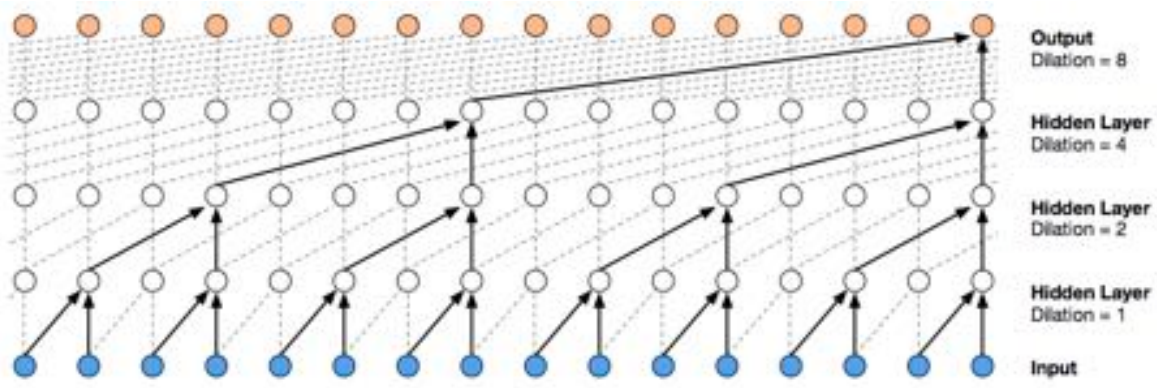
Masked convolutions preserve raster scan order





PixelCNN

Dilated convolutions increase the receptive field



WaveNet

van den Oord et al., 2016a, 2016b, 2016c

Roadmap

Introduction

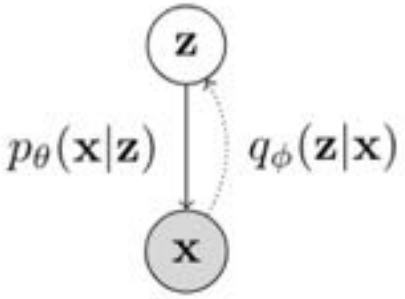
Background and Taxonomy

Likelihood-based Generative Models

- Autoregressive Models
- **Variational Autoencoders**

Variational Autoencoders

Directed, latent-variable models, **with an inference network**



Goal: Maximize the marginal log-likelihood of the dataset

$$\max_\theta \Sigma_{x \in \mathcal{D}} \log p_\theta(\mathbf{x})$$

Variational Autoencoders

Challenge: Marginal log-likelihood of any data point is intractable

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

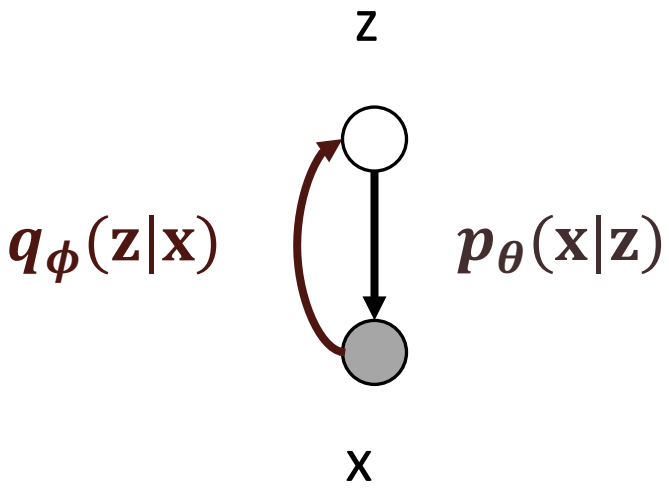
Key ideas

- Approximate the posterior $p_\theta(\mathbf{z}|\mathbf{x})$ with a simpler, tractable distribution $q_\phi(\mathbf{z}|\mathbf{x})$
- Cast **inference as optimization** over the parameters of the model and the approximate posterior

Inference as Optimization

Goal: Maximize the marginal log-likelihood of the data is **intractable!**

$$\log \int_{\mathbf{z}} p_\theta(\mathbf{x}, \mathbf{z}) = \log \int_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}$$



Inference as Optimization

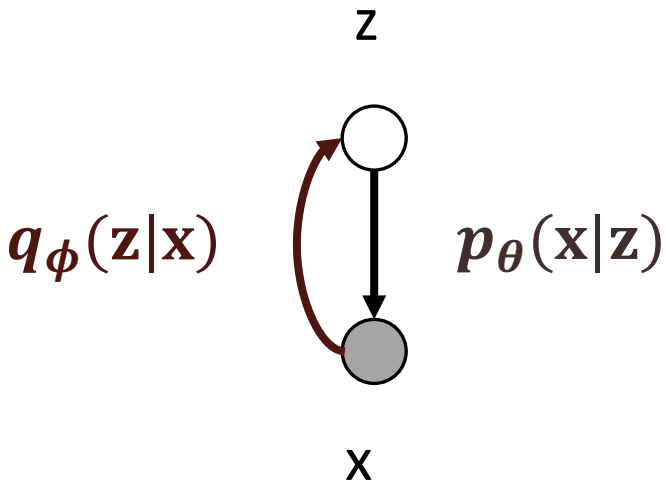
New Goal: Maximize **lower bound** to the marginal log-likelihood of the data is **tractable!**

$$\log \int_{\mathbf{z}} p_\theta(\mathbf{x}, \mathbf{z}) = \log \int_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}$$

$$\geq \underbrace{\int_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}}_{\text{ELBO}(\mathbf{x};\theta,\phi)}$$

Tightness Condition

$q_\phi(\mathbf{z}|\mathbf{x}) = p_\theta(\mathbf{z}|\mathbf{x})$

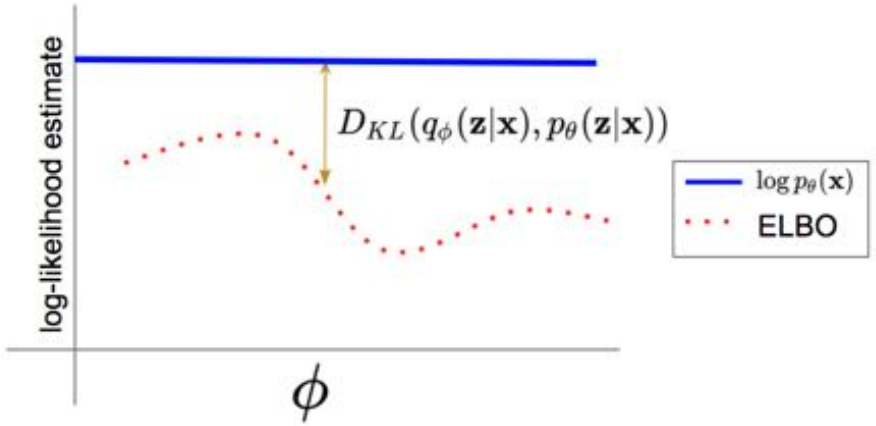


Variational Bayes

- Evidence lower bound (ELBO) for the marginal log-likelihood of $\mathbf{x}$

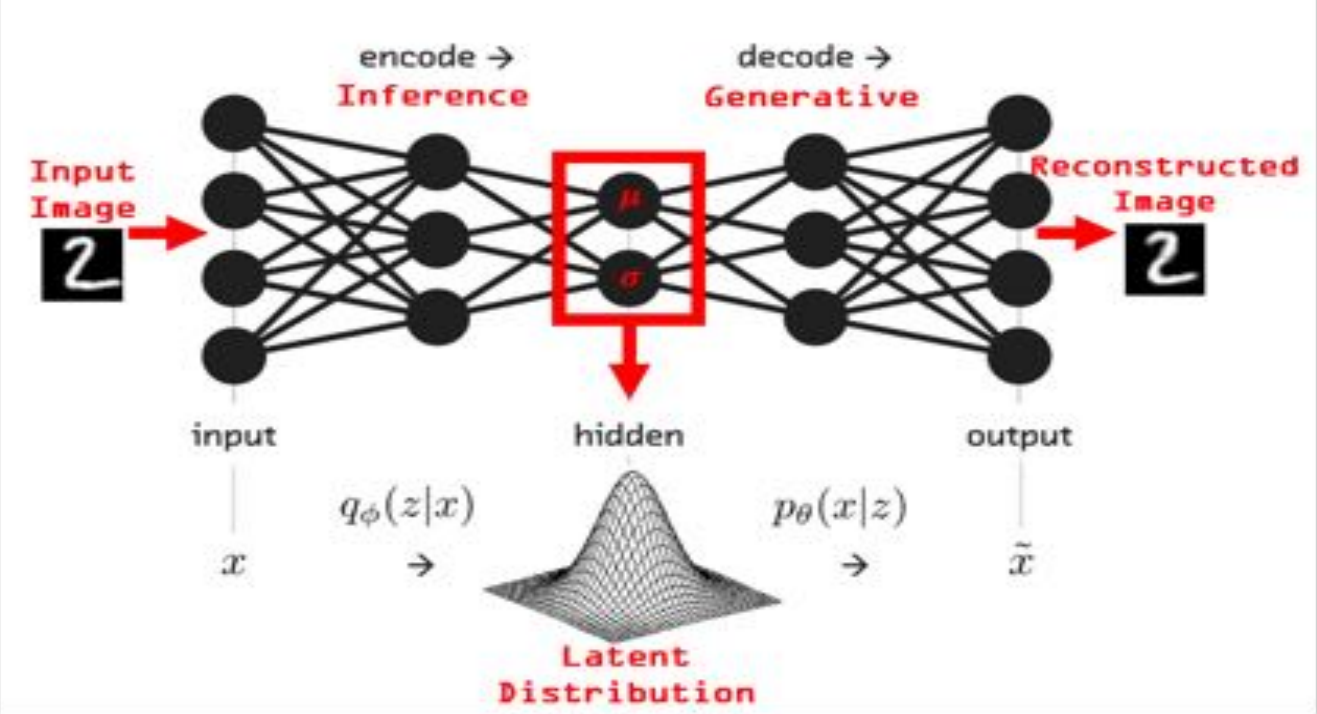
$$\log p_\theta(\mathbf{x}) = \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\left(\frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right)\right]}_{\mathrm{ELBO}(\mathbf{x};\theta,\phi)} + \mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x}))$$

- KL gap depends on the quality of the variational approximation



The Autoencoder Perspective

$$\text{ELBO}(\mathbf{x};\theta,\phi) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\left(\frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right)\right] = \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right]}_{\text{neg. reconstruction error}} - \underbrace{\text{KL}(q_\phi(\mathbf{z}|\mathbf{x}),p_\theta(\mathbf{z}))}_{\text{regularization penalty}}$$



Learning

- Learning maximizes the ELBO for the observed data $\mathcal{D}$
- Gradient estimates with respect to:
  1. model parameters θ: easy, use direct Monte Carlo
  2. variational parameters ϕ: score function estimates

$$\begin{aligned}\nabla_\phi \left[\mathbb{E}_{q_\phi}[f(\mathbf{z})]\right] &= \int \nabla_\phi q_\phi(\mathbf{z}) f(\mathbf{z}) \mathrm{d}\mathbf{z} \\ &= \int q_\phi(\mathbf{z}) \nabla_\phi \log q_\phi(\mathbf{z}) f(\mathbf{z}) \mathrm{d}\mathbf{z} \\ &= \mathbb{E}_{q_\phi}[\nabla_\phi \log q_\phi(\mathbf{z}) f(\mathbf{z})]\end{aligned}$$

Glynn, 1990, Williams, 1992, Fu, 2006

Inference

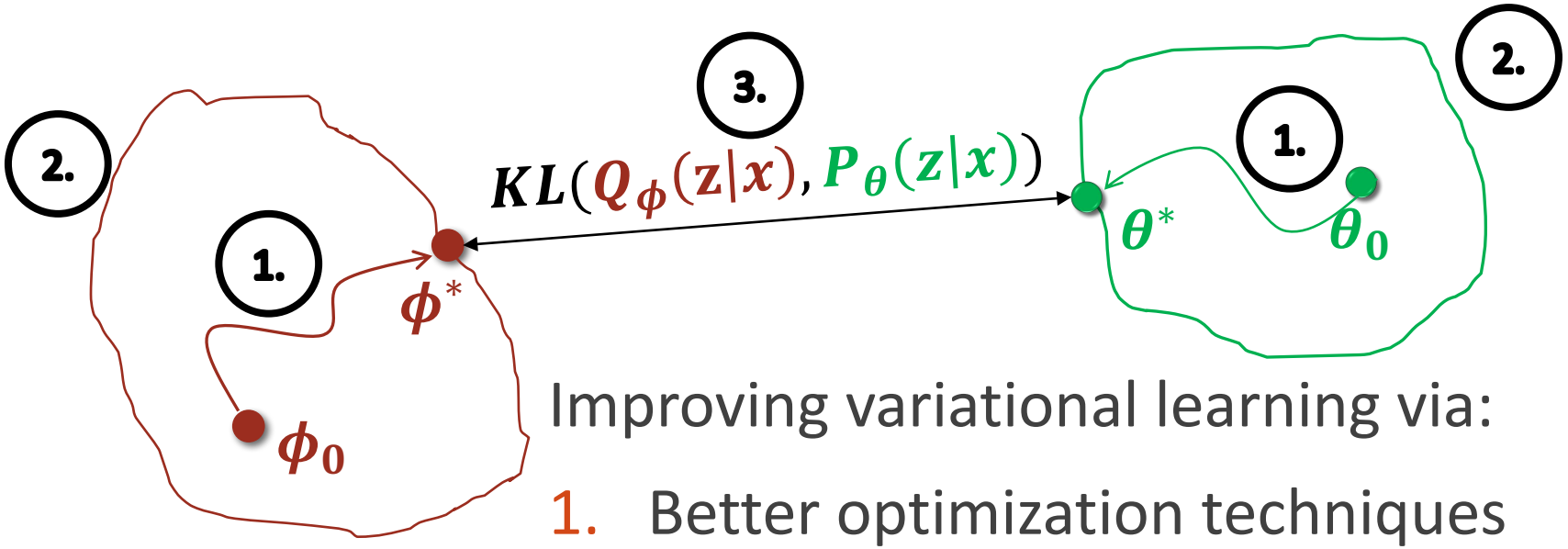
- ELBO and annealed importance sampling give **lower bounds** on true, intractable log-likelihood
- Directed model permits **ancestral sampling**:

$$\mathbf{z} \sim p(\mathbf{z}), \mathbf{x} \sim p_\theta(\mathbf{x} \,|\mathbf{z})$$

- **Latent representations** for any $\mathbf{x}$ can be inferred via $q_\phi(\mathbf{z} \,|\mathbf{x})$

Neal, 1998

Research avenues in a nutshell



Improving variational learning via:

1. Better optimization techniques
2. More expressive approximating families
3. Alternate loss functions

Figure inspired from David Blei's keynote at AISTATS 2018

Optimizing variational objectives

- **Stochastic Variational Inference** (Hoffman et al., 2013)
  - subsample data, fit variational parameters, repeat
- Gradient estimates with respect to variational parameters ϕ
  1. **Score function with control variates** (Wingate & Weber, 2013, Ranganath et al., 2014, Minh & Gregor, 2014, Gu et al., 2016, Mnih & Rezende, 2016)

$$E_{q_\phi}[\nabla_\phi \log q_\phi(z) f(z)] = E_{q_\phi}[\nabla_\phi \log q_\phi(z)(f(z) - b)]$$

  2. **Reparameterization** (Kingma & Welling, 2014, Rezende et al., 2014, Titsias & Lazaro-Gredilla, 2014)

$$z \sim N(\mu, \sigma^2) \;\Leftrightarrow\; \epsilon \sim N(0,1), z = \mu + \epsilon\sigma$$

Optimizing variational objectives

- Gradient estimates with respect to variational parameters ϕ
  1. **Score function with control variates** (Wingate & Weber, 2013, Ranganath et al., 2014, Minh & Gregor, 2014, Gu et al., 2016, Mnih & Rezende, 2016)
  $$E_{q_\phi}[\nabla_\phi \log q_\phi(z) f(z)] = E_{q_\phi}[\nabla_\phi \log q_\phi(z)(f(z) - b)]$$
  2. **Reparameterization** (Kingma & Welling, 2014, Rezende et al., 2014, Titsias & Lazaro-Gredilla, 2014)
  $$z \sim N(\mu, \sigma^2) \;\Leftrightarrow\; \epsilon \sim N(0,1), z = \mu + \epsilon\sigma$$
- Score function estimators can be applied to most continuous and discrete distributions
- Reparameterized estimators are applicable to certain continous distributions
  - Continuous Relaxations to discrete distributions using Gumbel perturbations (Jang et al., 2017, Maddison et al., 2017)

Model families - Encoder

- **Amortization** (Gershman & Goodman, 2015, Shu et al., 2018)
  - Generalization: Variational posterior shares statistical strength across $\mathbf{x}$
  - Scalability: Efficient test-time inference on massive datasets
- **Augmenting variational posteriors**
  - **Monte Carlo methods**: Importance Sampling (Burda et al., 2015), MCMC (Salimans et al., 2015, Hoffman, 2017, Levy et al., 2018), Sequential Monte Carlo (Maddison et al., 2017, Le et al., 2018, Naesseth et al., 2018), Rejection Sampling (Grover et al., 2018)
  - **Normalizing flows** (Rezende & Mohammed, 2015, Kingma et al., 2016)

Model families - Decoder

- **Powerful decoders** such as DRAW (Gregor et al., 2015), PixelCNN (Gulrajani et al., 2016)
- **Parameterized, learned priors** (Nalusnick et al., 2016; Tomczak & Welling, 2018; Graves et al., 2018)

Rethinking variational objectives

Tighter ELBO **does not imply:**

- **Better samples:** Samples and likelihoods are uncorrelated (Theis et al., 2016)
- **Informative latent codes:** Powerful decoders can ignore latent codes due to tradeoff in minimizing reconstruction error vs. KL prior penalty (Bowman et al., 2015, Chen et al., 2016, Zhao et al., 2017, Alemi et al., 2018)
- **Higher signal-to-noise ratio** in gradient estimates (Rainforth et al., 2018)

Rethinking variational objectives

Alternatives to the **reverse-KL variational objective**:

- Renyi's alpha-divergences (Li & Turner, 2016)
- f-divergences (Makhzani et al., 2015; Mescheder et al., 2017)
- integral probability metrics (Dziugaite et al., 2015; Tolstikhin et al., 2018)
- objectives derived from information theoretic principles (Alemi et al., 2018, Zhao et al., 2018)

Roadmap

Introduction

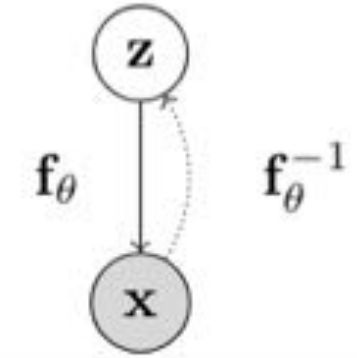
Background and Taxonomy

Likelihood-based Generative Models

- Autoregressive Models
- Variational Autoencoders
- **Normalizing Flow Models**

Normalizing flow models

Directed, latent-variable **invertible** models



Key idea: The mapping between $\mathbf{z}$ and $\mathbf{x}$, given by $\mathbf{f}_\theta: \mathbb{R}^n \rightarrow \mathbb{R}^n$, is **deterministic and invertible** such that $\mathbf{x} = \mathbf{f}_\theta(\mathbf{z})$ and $\mathbf{z} = \mathbf{f}_\theta^{-1}(\mathbf{x})$.

Normalizing flow models

- Use **change-of-variables** to relate densities in the spaces defined by $\mathbf{z}$ and $\mathbf{x}$:

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

- Transformations shape densities via **volume expansions and contractions**
- Determinant quantifies the per-unit change in volume

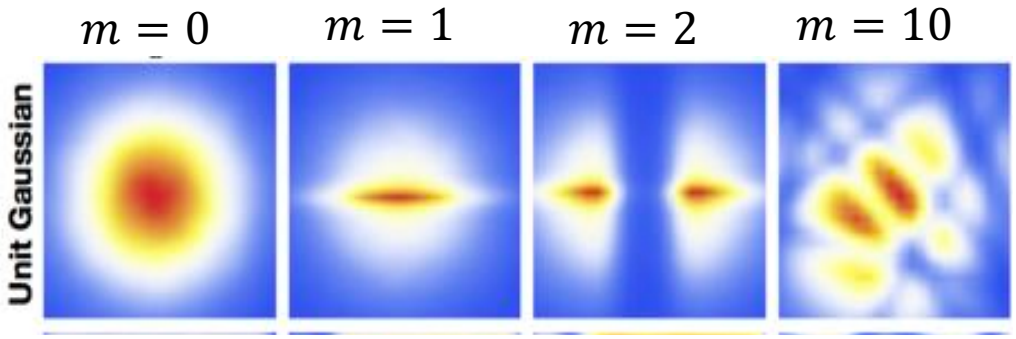
A flow of transformations

Invertible transformations can be composed with each other

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

Planar Flows

$$\mathbf{x} = \mathbf{z} + \mathbf{u} h(\mathbf{w}^T \mathbf{z} + b)$$



Rezende & Mohamed, 2016

Learning and inference

- **Learning** maximizes the model likelihood over the dataset $\mathcal{D}$

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

- **Exact likelihood evaluation** via inverse transformations
- **Ancestral sampling** via forward transformations

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

- **Latent representations** inferred via inverse transformations

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

Desiderata for flow models

- **Simple prior** that allows for sampling and tractable likelihood evaluation E.g., isotropic Gaussian
- **Invertible transformations with tractable evaluation:**
  - Likelihood evaluation requires efficient evaluation of inverse
  - Sampling requires efficient evaluation of inverse
- **Tractable evaluation of determinants** of Jacobian for large n
  - Computing determinants for an $n \times n$ matrix is $O(n^3)$: **prohibitive!**
  - **Key idea:** Determinant of **triangular matrices** is the product of the diagonal entries, i.e., an $O(n)$ operation

Triangular Jacobians

$$\mathbf{x} = \mathbf{f}(\mathbf{z}) \qquad \mathbf{J} = \begin{bmatrix} \frac{\partial \mathbf{f}}{\partial z_1} & \cdots & \frac{\partial \mathbf{f}}{\partial z_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial z_1} & \cdots & \frac{\partial f_1}{\partial z_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial z_1} & \cdots & \frac{\partial f_n}{\partial z_n} \end{bmatrix}$$

Lower triangular
x_i depends on $z_{\leq i}$

$$\begin{bmatrix} \frac{\partial f_1}{\partial z_1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial z_1} & \cdots & \frac{\partial f_n}{\partial z_n} \end{bmatrix} \qquad \begin{bmatrix} \frac{\partial f_1}{\partial z_1} & \cdots & \frac{\partial f_1}{\partial z_n} \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{\partial f_n}{\partial z_n} \end{bmatrix}$$

Upper triangular
x_i depends on $z_{\geq i}$

Triangular Jacobian → **Autoregressive structure**

Flow transformations in practice

- Nonlinear Independent Components Estimation (NICE)
  Additive coupling at intermediate layers and **scaling** at final layer

$$x_{1:d} = z_{1:d}, \quad x_{d+1,n} = z_{d+1,n} + \mu(z_{1:d})$$

$$x_i \leftarrow s_i x_i$$

- Real-valued Non-Volume Preserving (Real NVP)
  Location-scale coupling at all layers

$$x_{1:d} = z_{1:d}, \quad x_{d+1,n} = z_{d+1,n} \odot \sigma(z_{1:d}) + \mu(z_{1:d})$$

- NICE and Real-NVP have same complexity for evaluating forward and inverse transformations

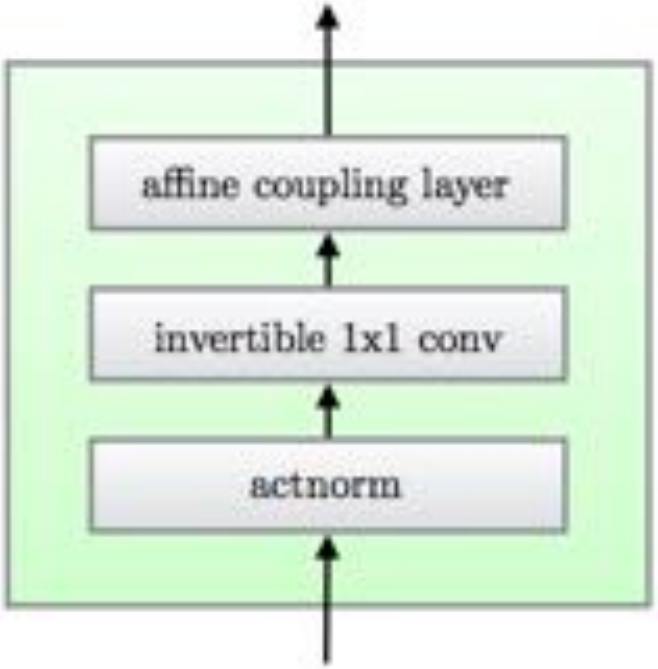
Dinh et al., 2015, 2017

Flow transformations in practice

- Masked and inverse autoregressive flows (MAF, IAF) trade-off computational complexity of forward and inverse transformations
  - **MAF** → **inverse** ☺ → likelihood-based learning E.g., VAE encoder
  - **IAF** → **forward** ☺ → likelihood-free learning E.g., VAE decoder
- Parallel Wavenet
  - **Teacher: MAF-like model** learned using maximum likelihood estimation
  - **Student:** a **distilled IAF model** trained by minimizing KL divergence with teacher model → three orders of magnitude faster generation than teacher!

Papamakarios et al., 2017, Kingma et al., 2016, van den Oord et al., 2018

Glow



Kingma and Dhariwal, 2018

Roadmap

Introduction

Background and Taxonomy

Likelihood-based Generative Models

- Autoregressive Models
- Variational Autoencoders
- Normalizing Flow Models

Likelihood-free Generative Models

- **Generative Adversarial Networks**

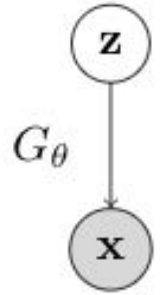
Likelihood-free generative models

- Optimal generative model: best samples and highest log-likelihoods
- For imperfect models, **log-likelihoods and samples are uncorrelated**
  - **Great held-out log-likelihoods, poor samples.** E.g., For a noise mixture model $p_\theta(\mathbf{x}) = 0.01\, p_{\text{data}}(\mathbf{x}) + 0.99 p_{\text{noise}}(\mathbf{x}),\ \log p_\theta(\mathbf{x}) \geq \log p_{\text{data}}(\mathbf{x}) - \log 100$
  - **Great samples, poor held-out log-likelihoods.** E.g., Memorizing training set
- Likelihood-free learning considers objectives that do not depend directly on the likelihood function E.g., 2-sample test objectives

Theis et al., 2016, Mohamed & Lakshminarayanan, 2016

Generative Adversarial Networks

- Directed, latent variable generative models **with a discriminator**



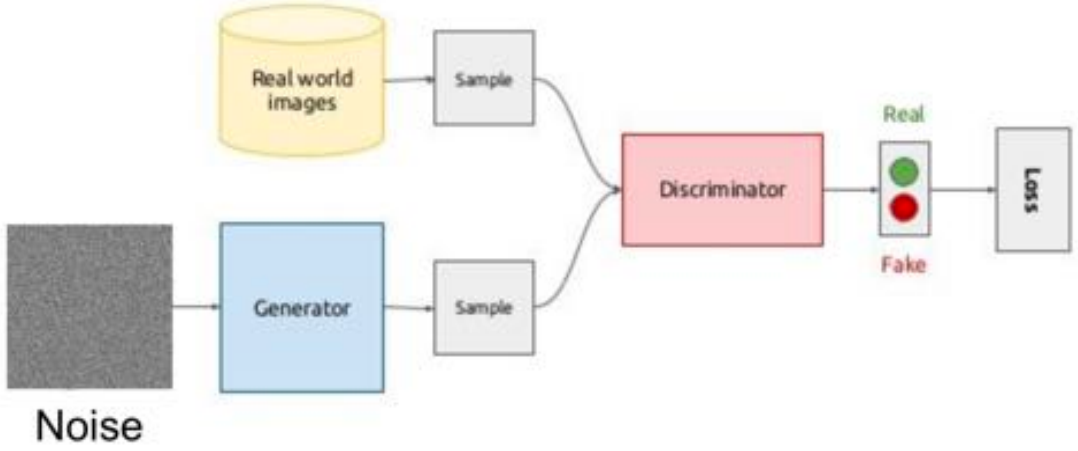
Generative Model

- The mapping between $\mathbf{z}$ and $\mathbf{x}$, given by G_θ, is **deterministic**

Generative Adversarial Networks

Key idea: A **two player generator-discriminator game**

- **Discriminator** distinguishes between real dataset samples and fake samples from the generator
- **Generator** generates samples that can fool the discriminator



Training procedure

Distributional perspective - Discriminator

$$\min_G \max_D V(G,D) = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\log(1 - D(G(\mathbf{z}))]$$

- For a fixed generator, discriminator is maximizing negative cross entropy
- Optimal discriminator is given by:

$$D^*_G(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})}$$

Goodfellow et al., 2014

Distributional perspective - Generator

$$\min_G \max_D V(G,D) = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\log(1 - D(G(\mathbf{z}))]$$

- For the optimal discriminator, generator is minimizing (shifted and scaled) Jenson-Shannon divergence (JSD) between p_g and p_{data}

$$C(G) = -\log(4) + \left\{ KL\left(p_{\text{data}} \middle\| \frac{p_{\text{data}} + p_g}{2}\right) + KL\left(p_g \middle\| \frac{p_{\text{data}} + p_g}{2}\right) \right\}$$

Jenson-Shannon Divergence

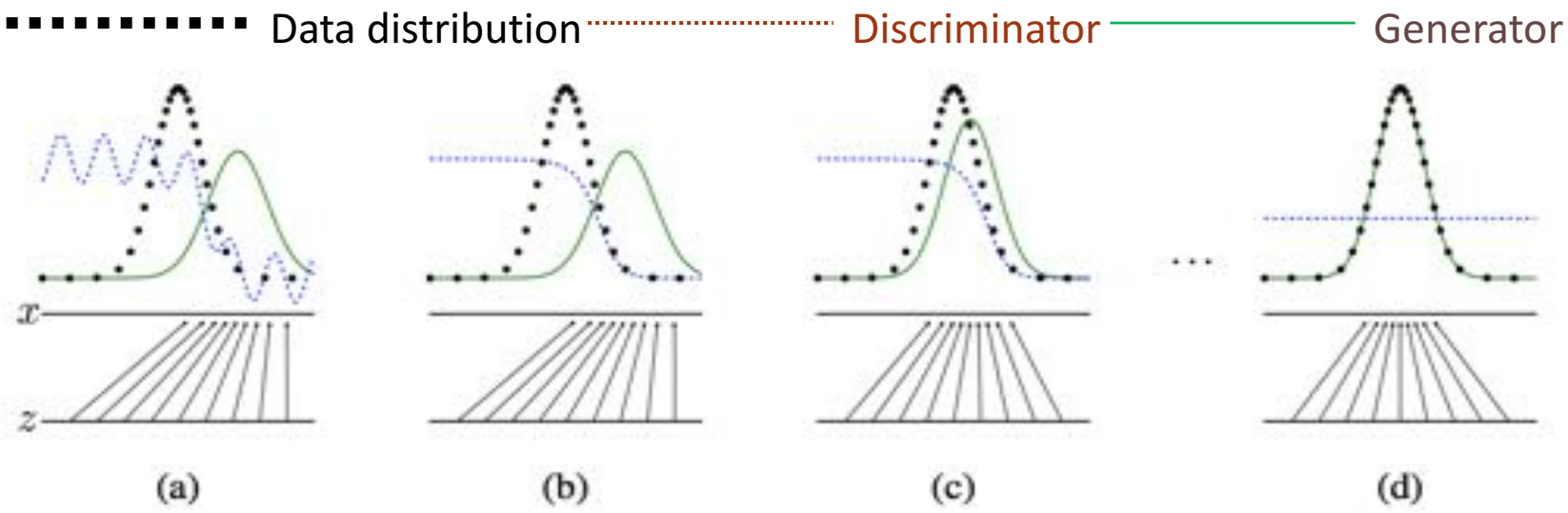
- Optimal generator is given by $p_g = p_{\text{data}}$

Goodfellow et al., 2014

A minimax learning objective

During learning, generator and discriminator are updated alternatively

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



Goodfellow et al., 2014

Evaluation

- **Likelihoods** may not be defined or tractable
  - Approximate likelihood evaluation using kernel density estimation and annealed importance sampling → **con:** unreliable estimates
  - Flow-GANs: Exact likelihood evaluation for normalizing flow generators trained adversarially→ **con:** constrains generator architecture
- Directed model permits **ancestral sampling**
  - For labelled datasets, metrics such as inception scores and Frechet inception distance quantify sample diversity and quality using pretrained classifiers

Wu et al., 2017, Grover et al., 2018, Salimans et al., 2016, Heusel et al., 2018

Training dynamics

If the generator updates are made in function space and discriminator is optimal at every step, then the generator is guaranteed to converge to the data distribution ← **unrealistic assumptions!**

In practice, GANs suffer from **mode collapse**

Arjovsky et al., 2017

Stabilizing training

- **Theory of GAN convergence:** Arjovsky & Bottou, 2017, Arora et al., 2017, Mescheder et al., 2017, Roth et al., 2017, Nagarajan & Kolter, 2018, Li et al., 2018
- JSD can be replaced **f-divergences and integral probability metrics** (Nowozoin et al., 2016, Li et al., 2017, Arjovsky et al., 2017)
- **Practical architectures, regularizers, optimization algorithms:** Radford et al., 2015, Poole et al., 2016, Salimans et al., 2016, Gulrajani et al., 2017, Metz et al., 2017, Miyato et al., 2018, Karras et al., 2018 and many more!

https://github.com/soumith/ganhacks

- *How to Train a GAN? Tips and tricks to make GANs work* by Soumith Chintala

Inferring latent representations

Prefinal layer of discriminator (Salimans et al., 2016)

Controlled generation: InfoGAN (Chen et al., 2016)

Inferring latent representations

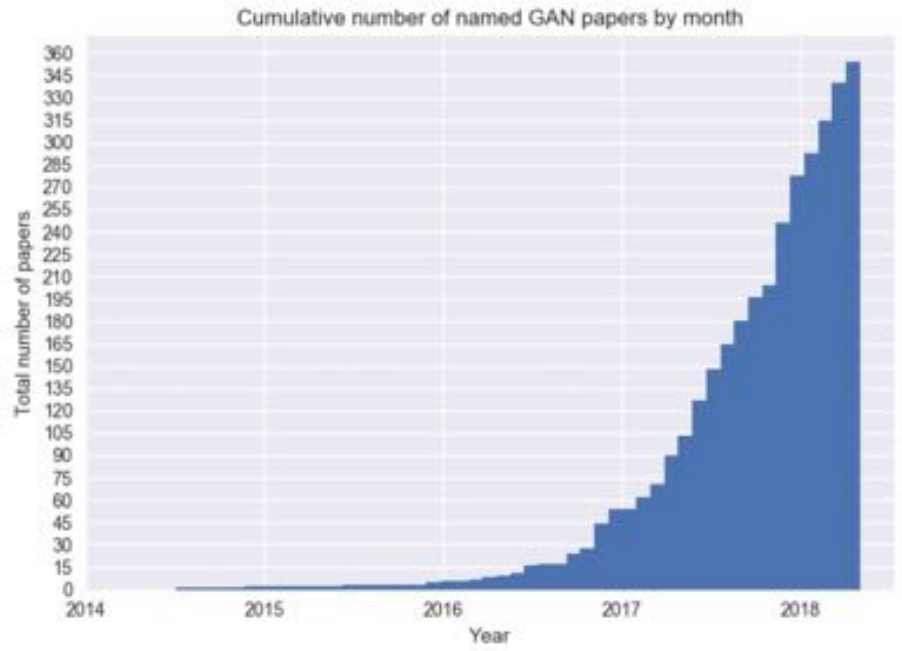
Learn **explicit encoders** along with generators: Adversarially Learned Inference(Dumoulin et al., 2017), BiGAN (Donahue et al., 2017)

Conditional GANs for image translation: Pix2Pix (Isola et al., 2017), CycleGAN (Zhu et al., 2017)

The GAN Zoo

https://github.com/hindupuravinash/the-gan-zoo

- *A list of all named GANs* by Avinash Hindupur



Roadmap

Introduction

Background and Taxonomy

Likelihood-based Generative Models

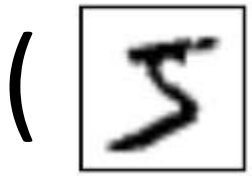
- Autoregressive Models
- Variational Autoencoders
- Normalizing Flow Models

Likelihood-free Generative Models

- Generative Adversarial Networks

Applications: Semi Supervised learning, imitation learning, adversarial examples, compressed sensing

Semi Supervised Learning



$= (x_1 , y_1)$

…

$= (x_N , y_N)$

Labeled data

$= (x_{N+1} , ?)$

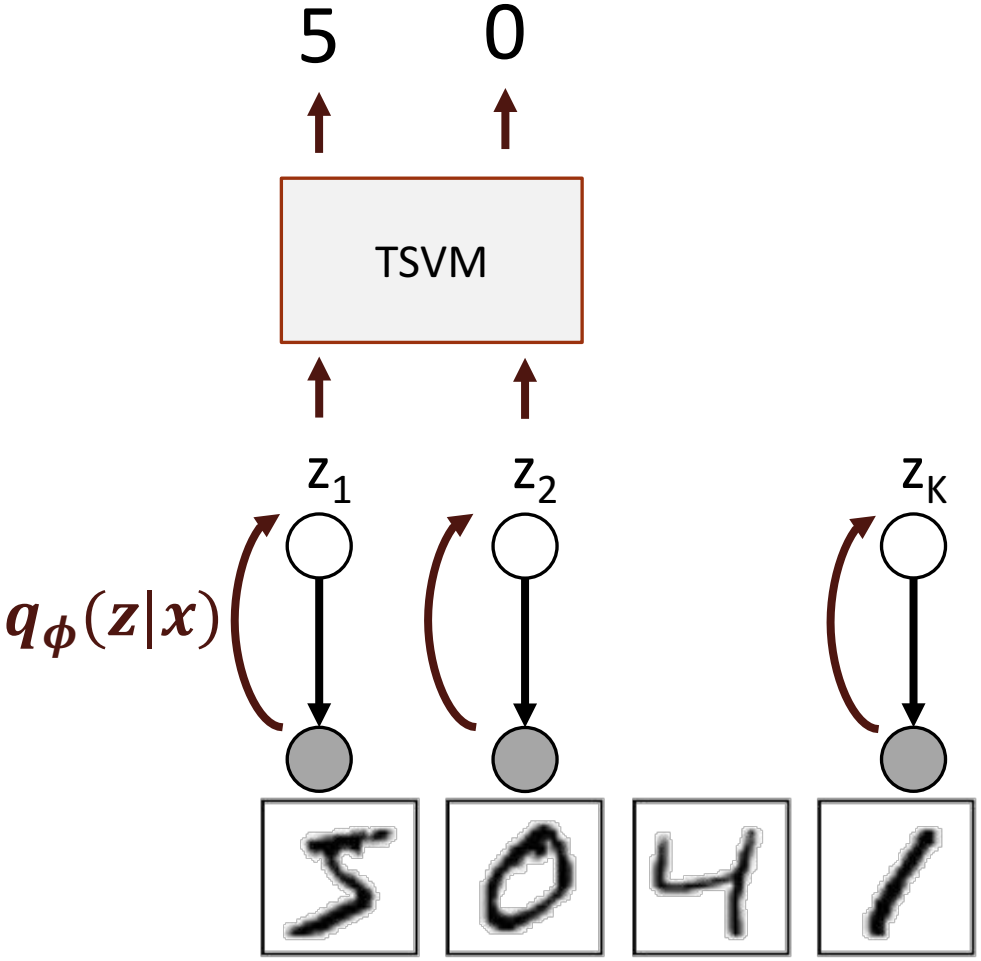
…

$= (x_{N+M} , ?)$

Unlabeled data

How can we leverage unlabeled data?

Latent-feature discriminative model (M1)



Step 2: train classifier (e.g., SVM), using z as features, only using the labeled part

Step 1: learn a latent var. generative model on **labeled and unlabeled data**

Kingma et al, 2014

Generative model (M2)

( , 5) = (x_1 , y_1)

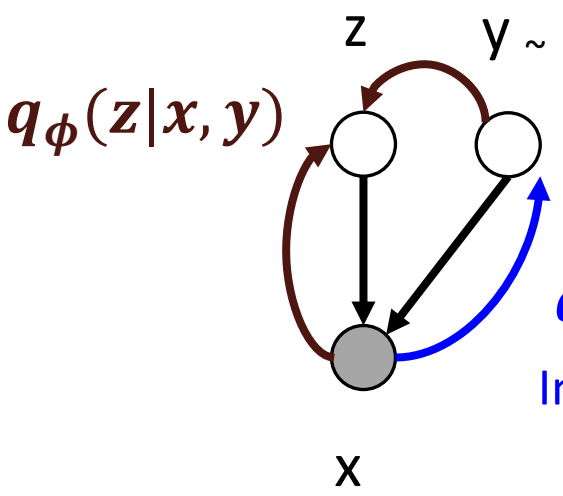
...

( , 0) = (x_N , y_N)

( , ?) = $(x_{N+1} , ?)$

...

( , ?) = $(x_{N+M} , ?)$



Inference net is a classifier!

Class-conditional VAE: a fully probabilistic model where the label (y) is another latent variable

Kingma et al, 2014

Generative model (M2)


, 5) = (x_1 , y_1)

…


, 0) = (x_N , y_N)

Labeled data

$$\log p_\theta(x_1, y_1) = \log \sum_z p_\theta(x_1, y_1, z)$$

≥ ELBO

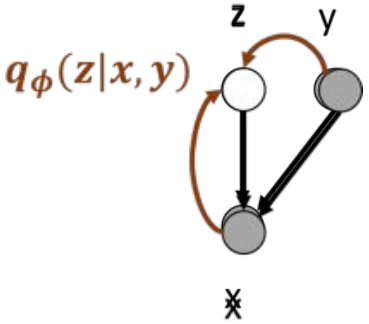

, ?) = (x_{N+1} , ?)

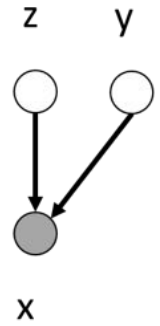
…


, ?) = (x_{N+M} , ?)

Unlabeled data

$$\log p_\theta(x_N) = \log \sum_y p_\theta(x_N, y)$$

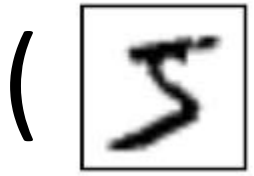




Experimental Results on MNIST

N	NN	CNN	TSVM	CAE	MTC	AtlasRBF	M1+TSVM	M2	M1+M2
100	25.81	22.98	16.81	13.47	12.03	8.10 ($\pm$ 0.95)	11.82 ($\pm$ 0.25)	11.97 ($\pm$ 1.71)	**3.33** ($\pm$ 0.14)
600	11.44	7.68	6.16	6.3	5.13	–	5.72 ($\pm$ 0.049)	4.94 ($\pm$ 0.13)	**2.59** ($\pm$ 0.05)
1000	10.7	6.45	5.38	4.77	3.64	3.68 ($\pm$ 0.12)	4.24 ($\pm$ 0.07)	3.60 ($\pm$ 0.56)	**2.40** ($\pm$ 0.02)
3000	6.04	3.35	3.45	3.22	2.57	–	3.49 ($\pm$ 0.04)	3.92 ($\pm$ 0.63)	**2.18** ($\pm$ 0.04)

Semi-supervised Learning with Deep Generative Models, 2014

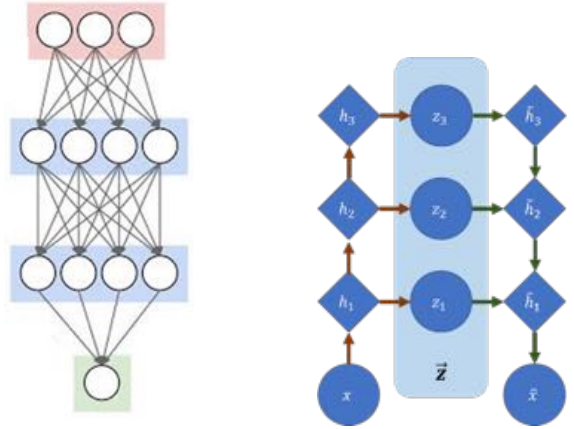


Generative vs. discriminative

Favorite disc. model Favorite gen. model

Generative approach. Maximize $\log p_\theta(x, y)$

Discriminative approach. Maximize $\log p_\theta(y|x)$

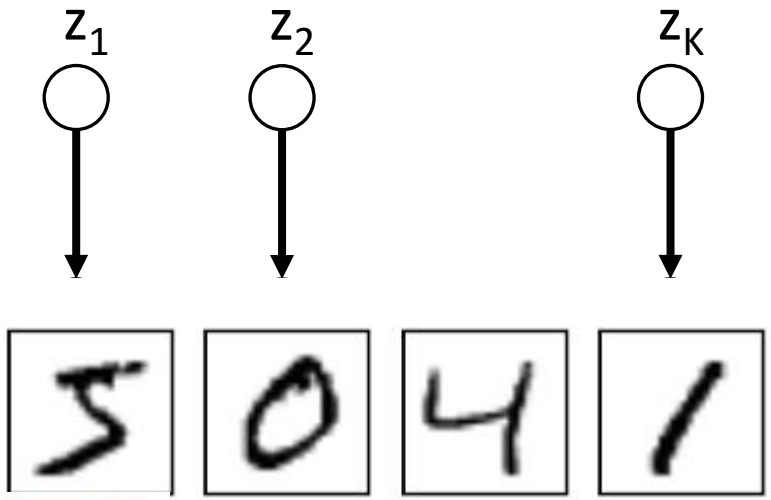


Multi-conditional (McCallum). Maximize $a \log p_\theta(y|x) + b \log p_\theta(x)$

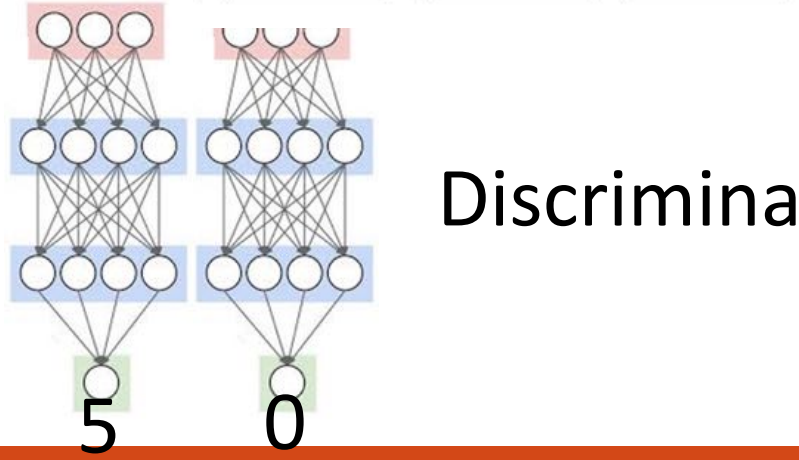
- a=1 , b=0 → Discriminative
- a=1=b → Generative

They **need** to share parameters: from same model!

Attempt 1: Discriminative + Generative



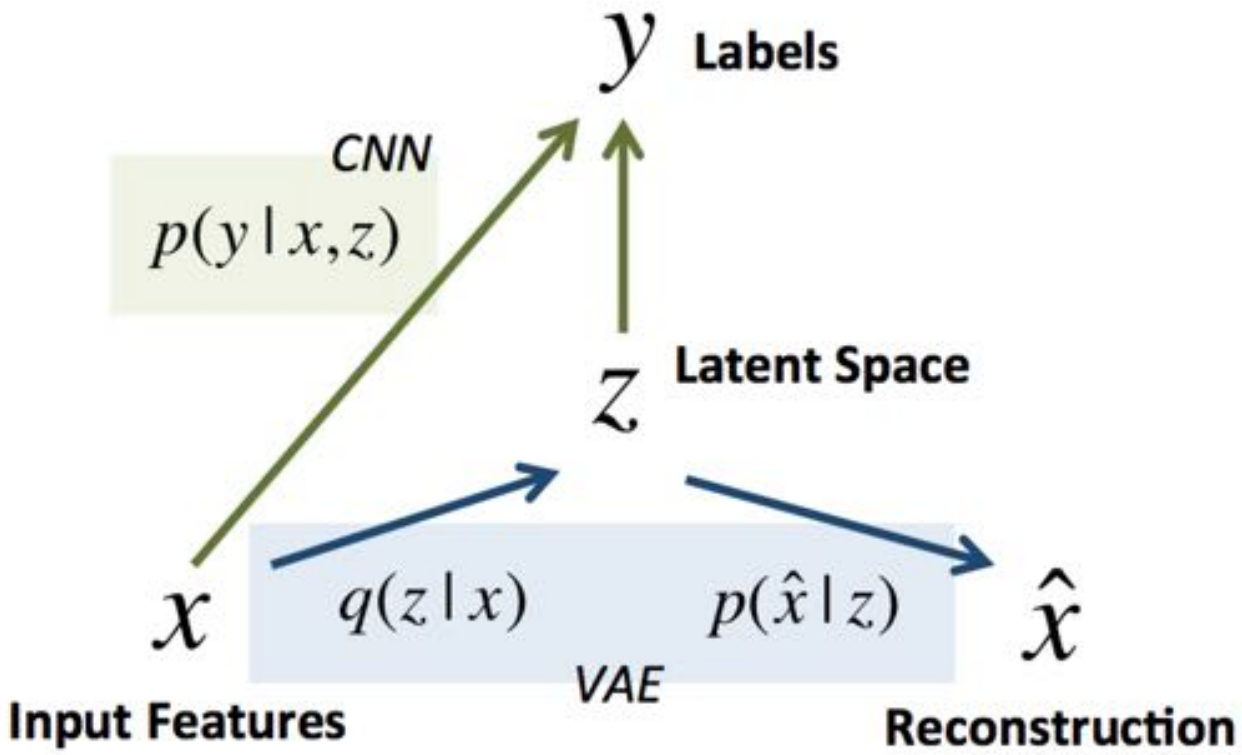
Generative model



Discriminative model

Issue: The two components are independent of each other!
Can be optimized separately

Semi Supervised Learning



Idea 1: couple any discriminative model and generative one through a shared latent space

Idea 2: train using a generalized multi-conditional objective for divergences (adversarial)

$$\alpha \cdot L_D\left[q(x,y,z), p(y|x,z)\right] + \beta \cdot L_G\left[q(z,x), p(x,z)\right]$$

Kuleshov et al, 2017

Results on Semi Supervised Learning

Method	Accuracy
VAE (Kingma et al., 2014)	$3.33 \pm 0.14\%$
SDGM (Maaløe et al., 2016)	$1.91 \pm 0.10\%$
Ladder Network (Rasmus et al.)	$1.06 \pm 0.37\%$
ADGM (Maaløe et al., 2016)	$0.96 \pm 0.02\%$
Improved GAN (Salimans et al., 2016)	$0.93 \pm 0.07\%$
Implicit DHM (ours)	$0.91 \pm 0.06\%$

MNIST (100 labels)

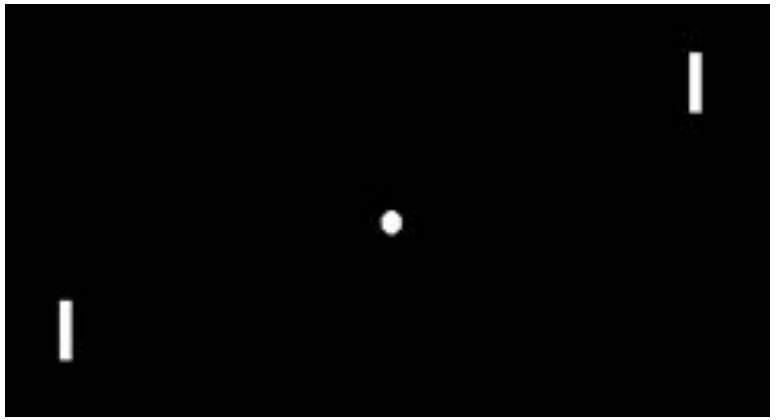
Method	Accuracy
VAE (Kingma et al., 2014)	$36.02 \pm 0.10\%$
SDGM (Maaløe et al., 2016)	$16.61 \pm 0.24\%$
Improved GAN (Maaløe et al., 2016)	$8.11 \pm 1.3\%$
ALI (Dumoulin et al., 2016)	$7.42 \pm 0.65\%$
Π-model (Laine & Aila, 2016)	$5.45 \pm 0.25\%$
Implicit DHM (ours)	$4.45 \pm 0.35\%$

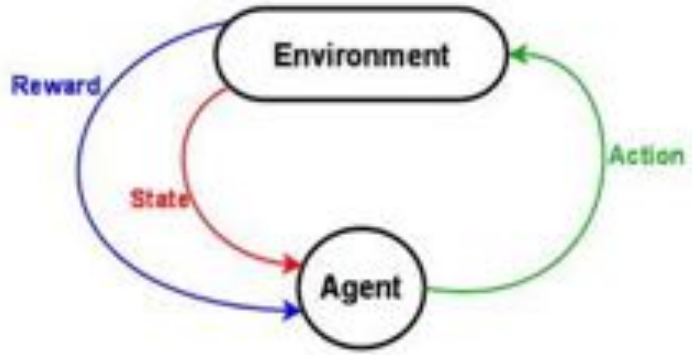
SVHN (1000 labels)

Imitation Learning

Input: demonstrations provided by *an expert*

$$\{(s_0^i, a_0^i, s_1^i, a_1^i, \dots)\}_{i=1}^n \sim \pi_E$$



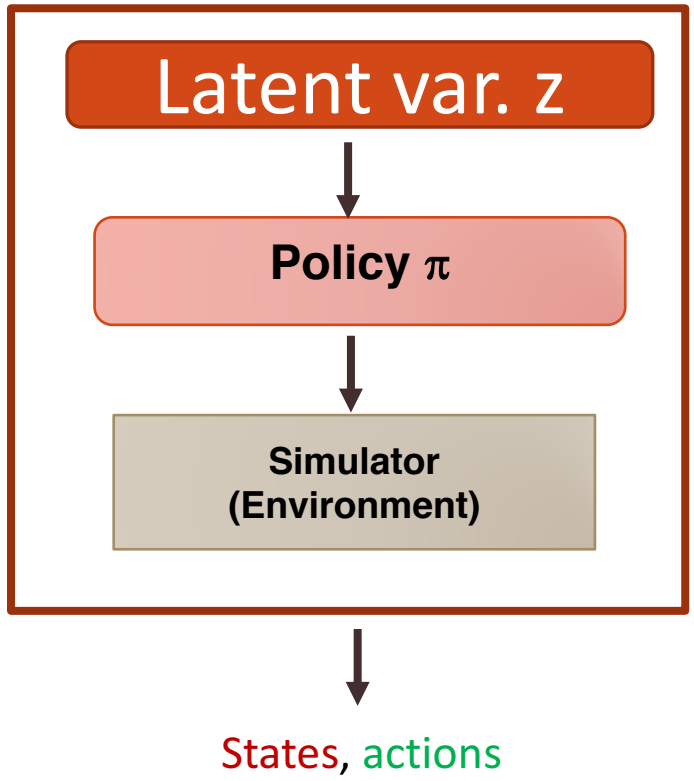


Goal: find policy that *generates* a similar behavior

Imitation Learning

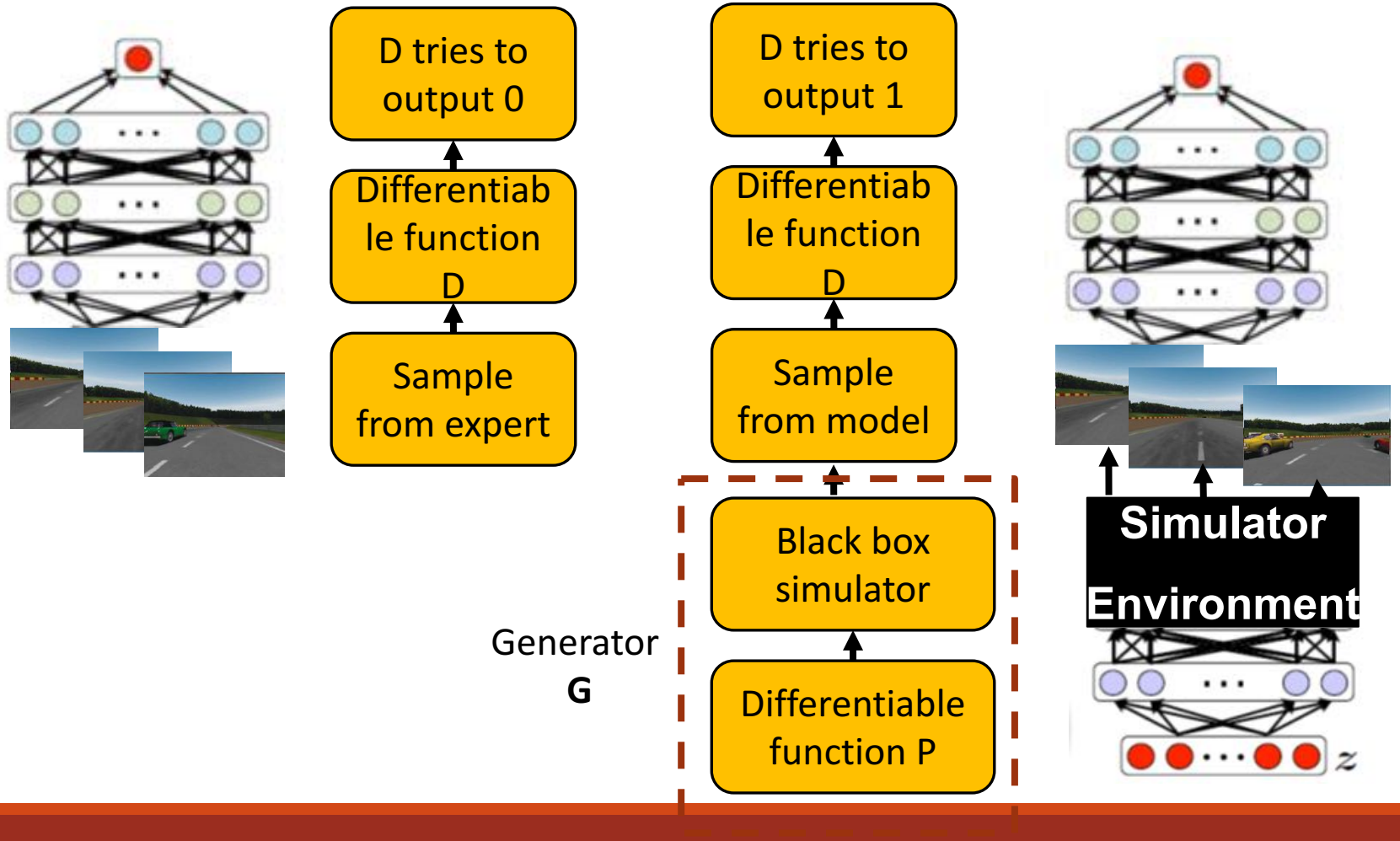
Several existing approaches:

- Behavioral cloning
- Inverse reinforcement learning
- Apprenticeship learning



Our approach: a generative **latent variable model**

Generative Adversarial Imitation Learning



Ho and Ermon, *2016*

How to optimize the objective

Previous Apprenticeship learning work:

- Full dynamics model
- Small environment
- Repeated RL

We propose: gradient descent over policy parameters (and discriminator)

Trust region policy optimization

J. Ho, J. K. Gupta, and S. Ermon, 2016.

Results

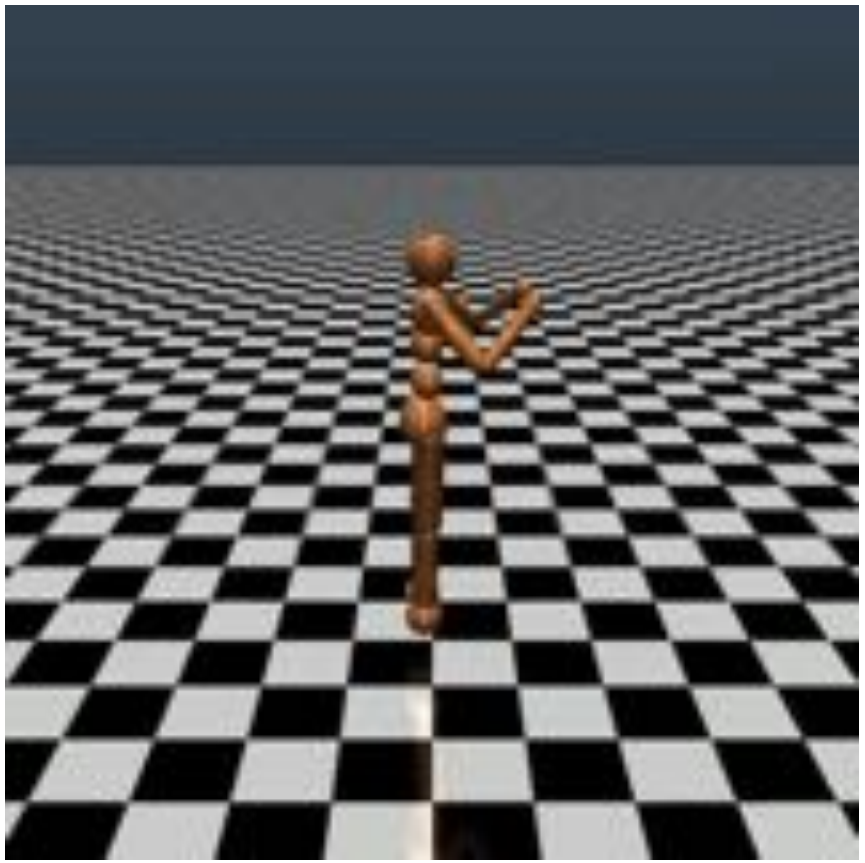
Input: driving demonstrations (Torcs)

Output policy:

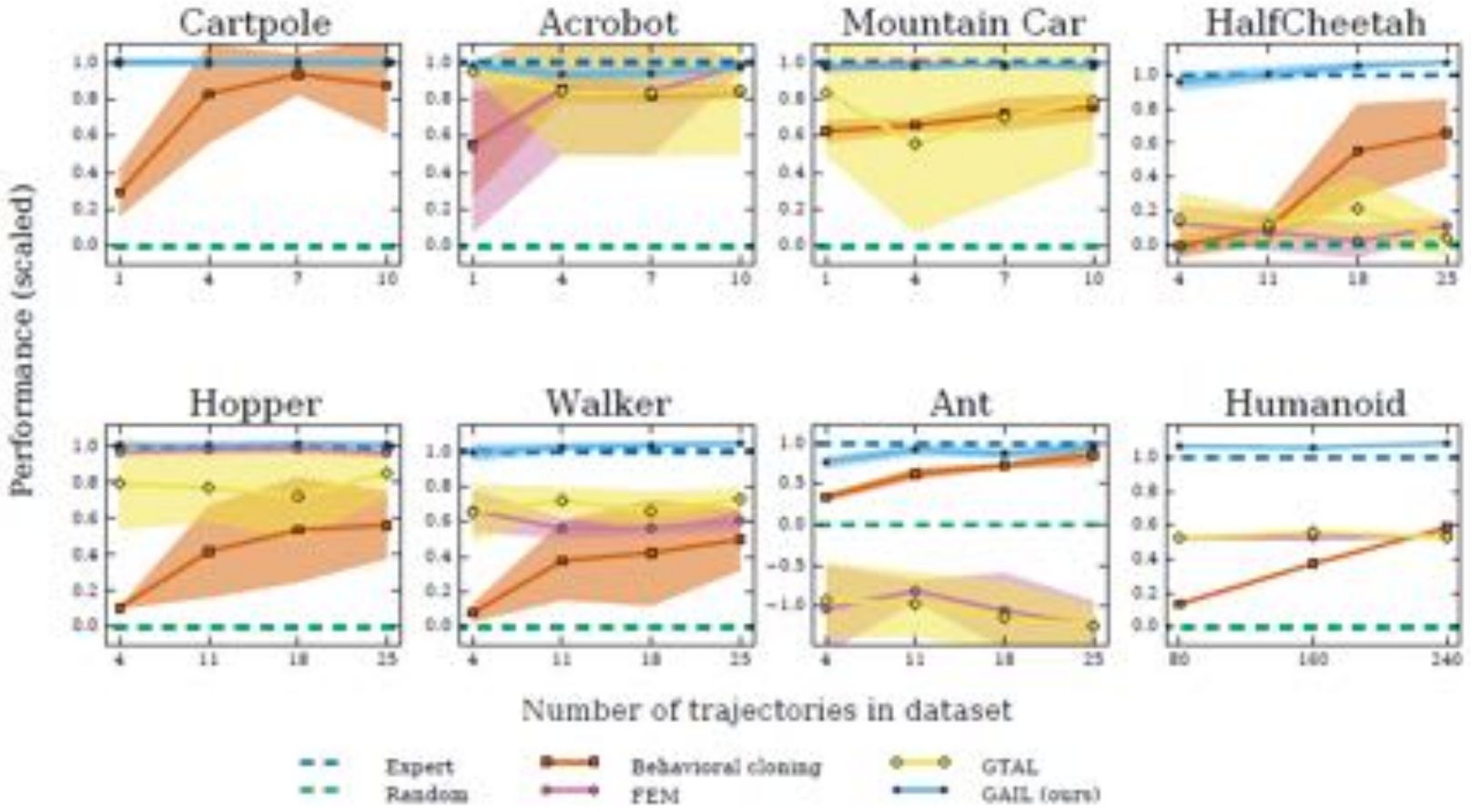
From raw visual inputs

Li et al. Inferring The Latent Structure of Human Decision-Making from Raw Visual Inputs, 2017

Results

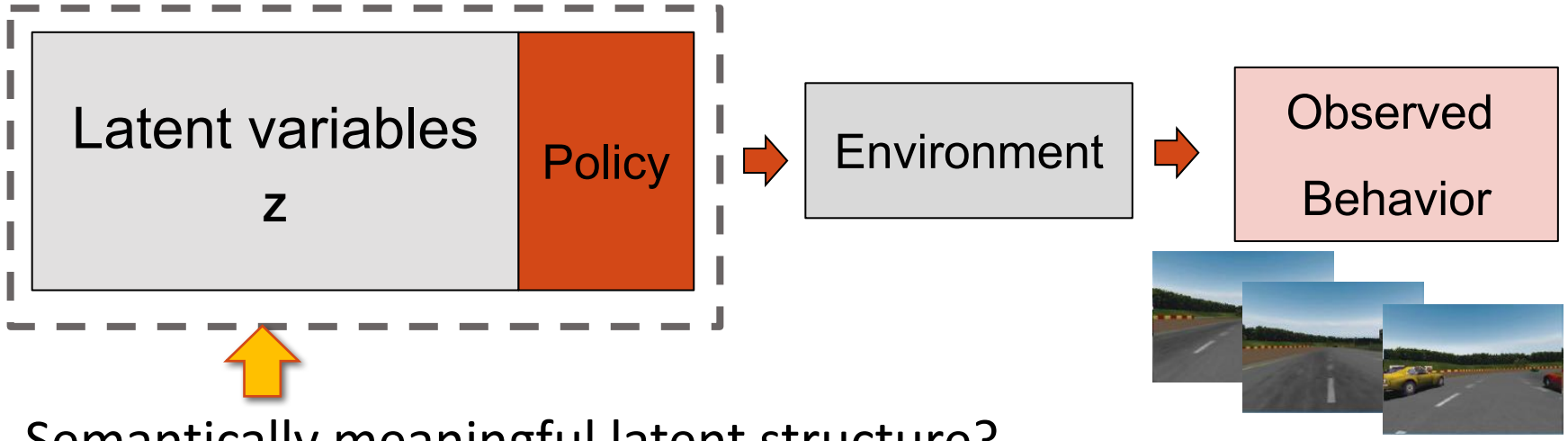


Experimental results

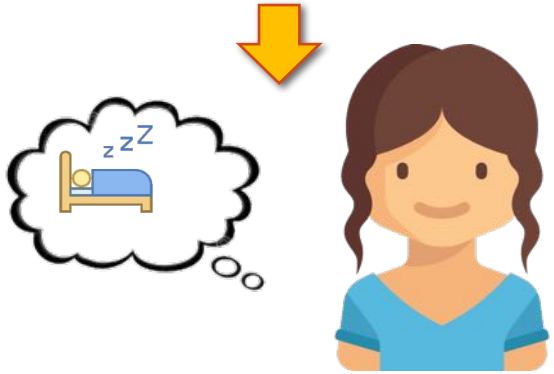


Latent structure in demonstrations

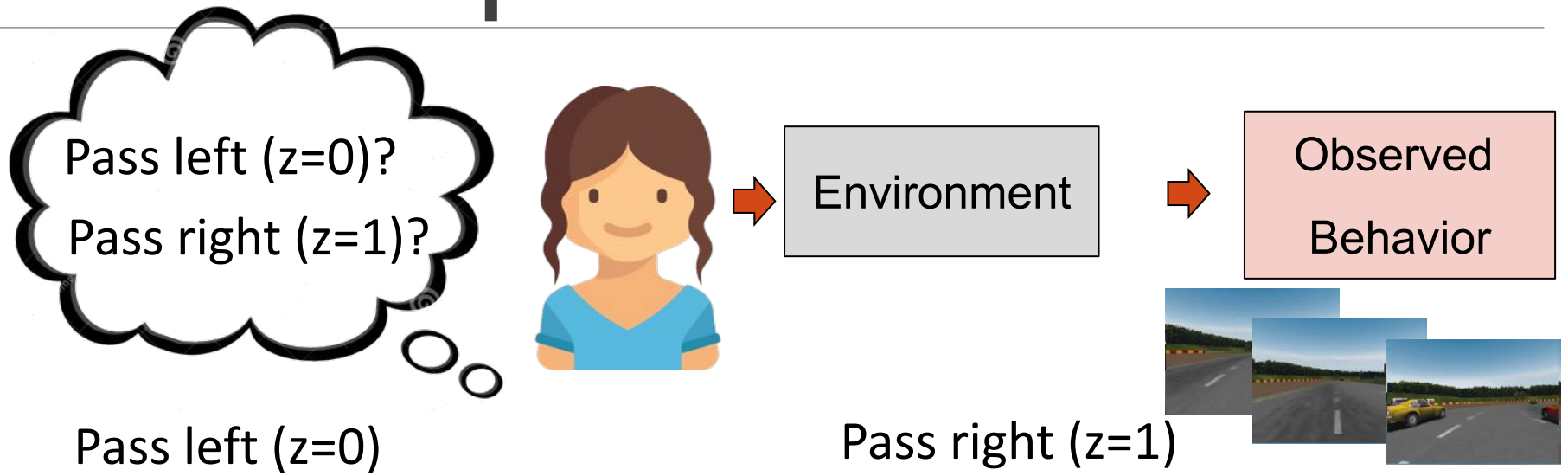
Human model



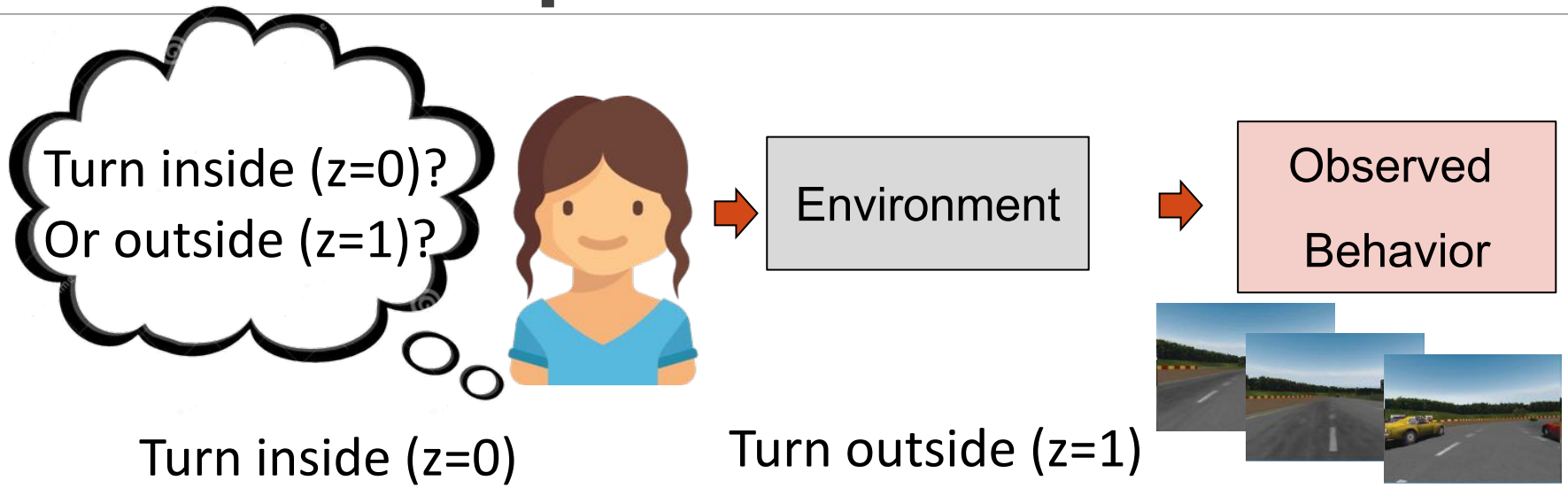
Semantically meaningful latent structure?



InfoGAIL Example



InfoGAIL Example

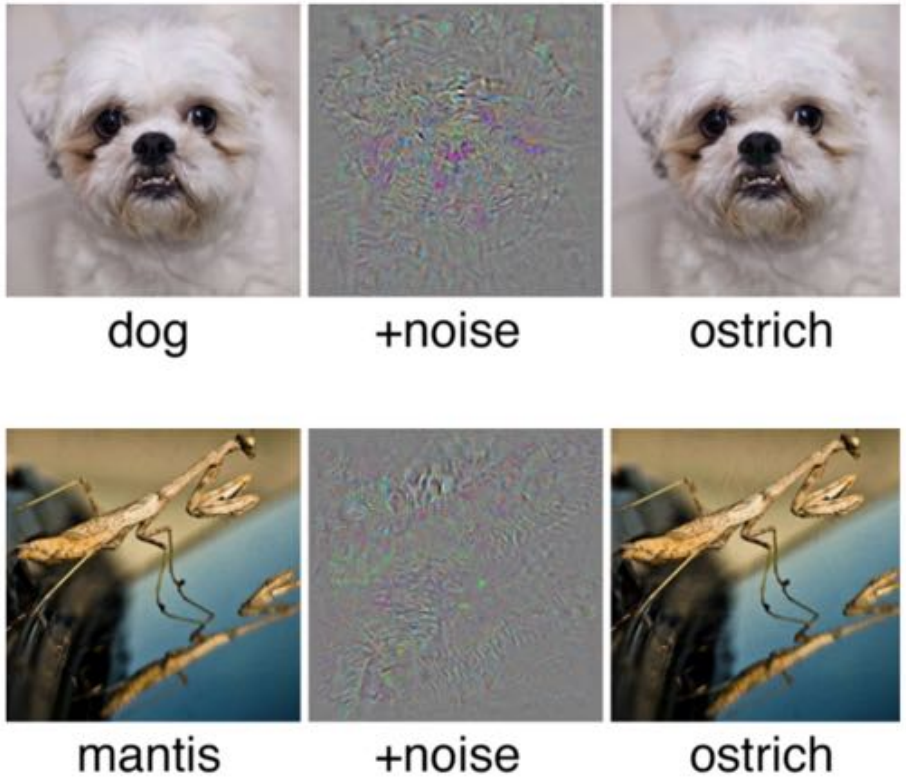


Turn inside (z=0)

Turn outside (z=1)

Anomaly detection - Adversarial examples

- State-of-the-art classifiers can be fooled by adding imperceptible noise



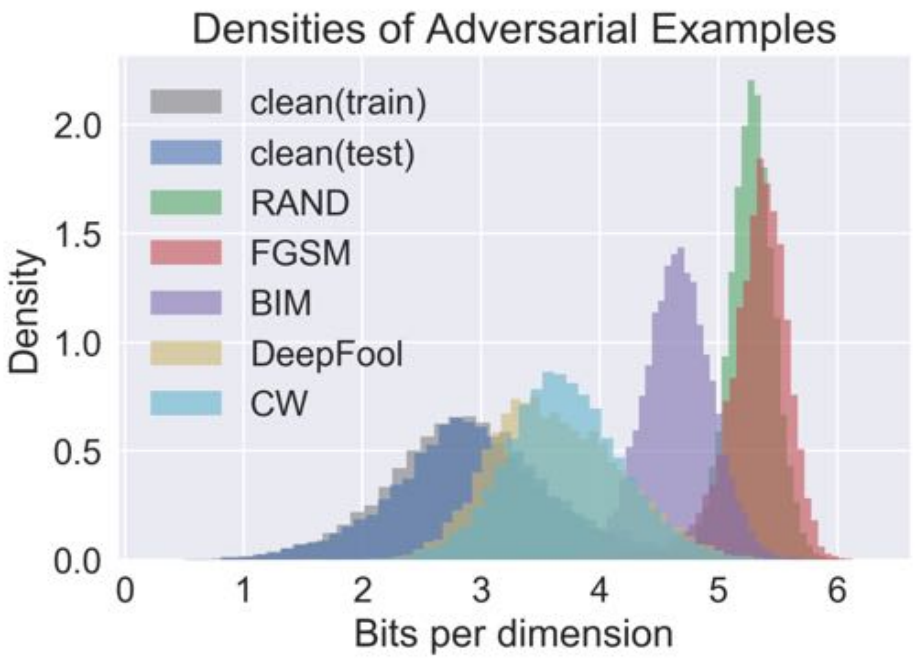
Detecting adversarial examples

- **Observation:** The PixelCNN density of an adversarial example is usually significantly lower than that of an clean example. Therefore, *$p(X)$ can be used as a test statistic to detect adversarial examples.*

- **Statistical test:** Given an input $X' \sim q(X)$ and training images $X_1, X_2, \cdots, X_N \sim p_t(X)$. The null hypothesis is $H_0: p_t(X) = q(X)$ while the alternative is $H_1: p_t(X) \neq q(X)$. The p-value is computed as

$$\text{p-value} = \frac{1}{N+1}\left(\sum_{i=1}^{N} \mathbb{I}\left[p(X_i) \leq p(X')\right] + 1\right)$$

Detecting adversarial examples



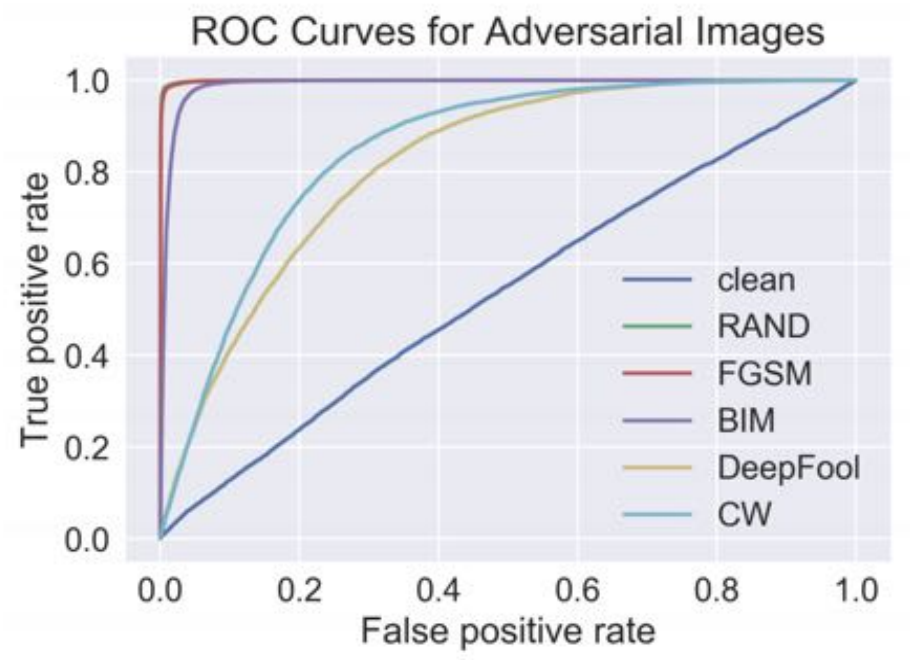


Figure: (Left) *Likelihoods of different adversarial examples.* (Right) *ROC curves for detecting various attacks.*

Song et al., 2018

PixelDefend

- **Intuition:** The harm of adversarial examples might be reduced if they can be modified to have higher likelihood.

Table 1: **Fashion MNIST** ($\epsilon_{\text{attack}} = 8/25$, $\epsilon_{\text{defend}} = 32$)

NETWORK	TRAINING TECHNIQUE	CLEAN	RAND	FGSM	BIM	DEEP FOOL	CW	STRONGEST ATTACK
ResNet	Normal	**93/93**	89/71	38/24	00/00	06/06	20/01	00/00
VGG	Normal	92/92	91/87	73/58	36/08	49/14	43/23	36/08
ResNet	Adversarial FGSM	**93/93**	**92**/89	85/85	51/00	63/07	67/21	51/00
	Adversarial BIM	92/91	**92/91**	84/79	76/63	82/72	81/70	76/63
	Label Smoothing	**93/93**	91/76	73/45	16/00	29/06	33/14	16/00
	Feature Squeezing	84/84	84/70	70/28	56/25	83/83	83/83	56/25
	Adversarial FGSM + Feature Squeezing	88/88	87/82	80/77	70/46	86/82	84/85	70/46
ResNet	Normal + ***PixelDefend***	88/88	88/89	85/74	83/76	87/87	87/87	83/74
VGG	Normal + ***PixelDefend***	89/89	89/89	87/82	85/83	88/88	88/88	85/82
ResNet	Adversarial FGSM + ***PixelDefend***	90/89	91/90	**88**/82	**85**/76	**90**/88	**89/88**	**85**/76
	Adversarial FGSM ***+Adaptive PixelDefend***	91/91	91/**91**	**88/88**	**85/84**	89/**90**	**89**/84	**85/84**

Table 2: **CIFAR-10** ($\epsilon_{\text{attack}} = 2/8/16$, $\epsilon_{\text{defend}} = 16$)

NETWORK	TRAINING TECHNIQUE	CLEAN	RAND	FGSM	BIM	DEEP FOOL	CW	STRONGEST ATTACK
ResNet	Normal	**92/92/92**	**92**/87/76	33/15/11	10/00/00	12/06/06	07/00/00	07/00/00
VGG	Normal	89/89/89	89/88/80	60/46/30	44/02/00	57/25/11	37/00/00	37/00/00
ResNet	Adversarial FGSM	91/91/91	90/**88**/84	**88/91/91**	24/07/00	45/00/00	20/00/07	20/00/00
	Adversarial BIM	87/87/87	87/87/86	80/52/34	74/32/06	79/48/25	76/42/08	74/32/06
	Label Smoothing	**92/92/92**	91/**88**/77	73/54/28	59/08/01	56/20/10	30/02/02	30/02/01
	Feature Squeezing	84/84/84	83/82/76	31/20/18	13/00/00	75/75/75	78/78/78	13/00/00
	Adversarial FGSM + Feature Squeezing	86/86/86	85/84/81	73/67/55	55/02/00	**85/85/85**	83/83/83	55/02/00
ResNet	Normal + ***PixelDefend***	85/85/88	82/83/84	73/46/24	71/46/25	80/80/80	78/78/78	71/46/24
VGG	Normal + ***PixelDefend***	82/82/82	82/82/84	80/62/52	80/61/48	81/76/76	81/79/79	80/61/48
ResNet	Adversarial FGSM + ***PixelDefend***	88/88/86	86/86/**87**	81/68/67	**81/69/56**	**85/85/85**	**84/84/84**	**81/69/56**
	Adversarial FGSM + ***Adaptive PixelDefend***	90/90/90	86/87/**87**	81/70/67	**81/70/56**	82/81/82	81/80/81	**81/70/56**

PixelDefend Experiments

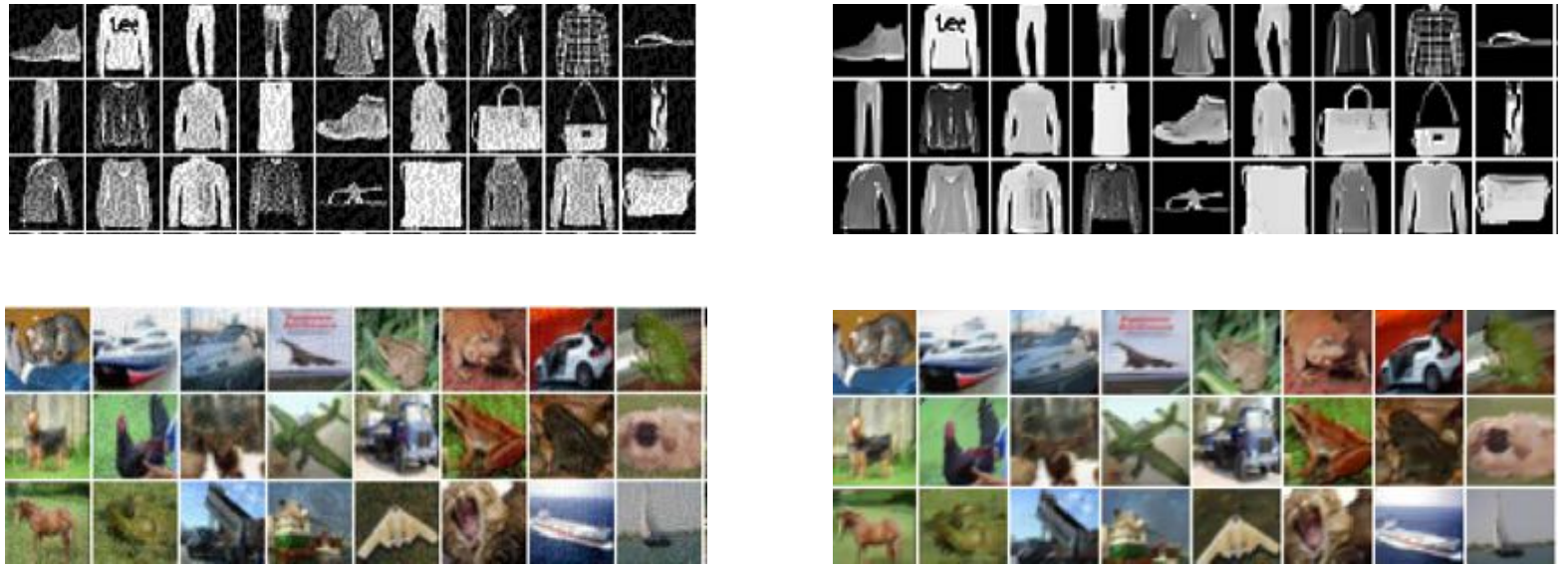
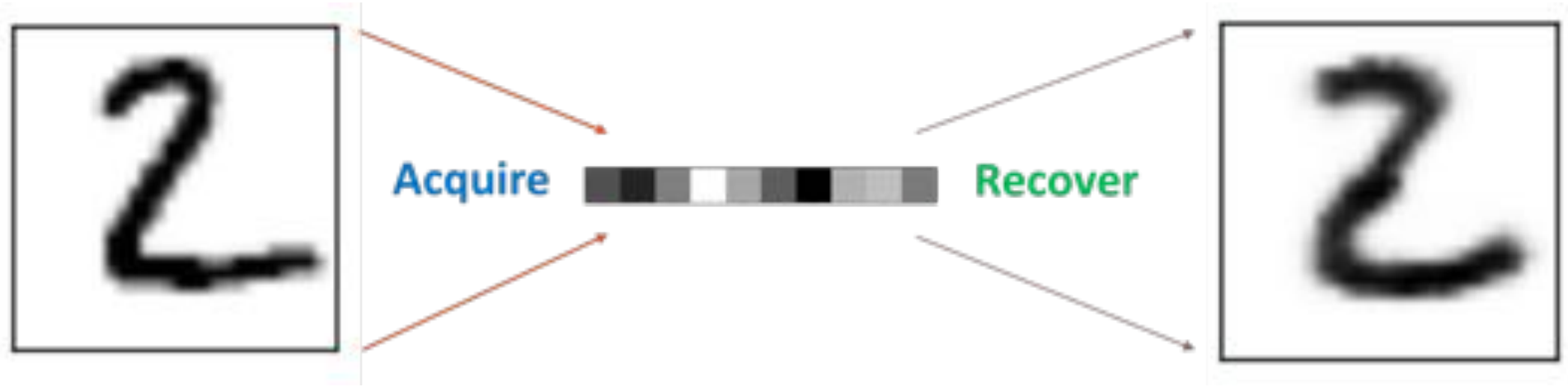


***Figure**: Adversarial images (left) and purified images after PixelDefend (right).*

Compressed Sensing

Goal: Efficient acquisition and recovery of high dimensional signals



High dimensional (unobserved)

Low dimensional (observed)

High dimensional (recovered)

Acquisition: Low dimensional random projections

Recovery: Inverse optimization with structural assumptions

Setup

Solve for an n dimensional signal $\mathbf{x}$ using m linear measurements $\mathbf{y}$.

$$\mathbf{y} = \mathrm{A}\mathbf{x} + \boldsymbol{\epsilon}$$

Compressed sensing: $m << n$

Underdetermined system of equations have *infinite solutions*

… unless we impose **structure** on $\mathbf{x}$

Candes & Tao, 2005; Donoho, 2006; Candes et al., 2006

Structural Assumptions

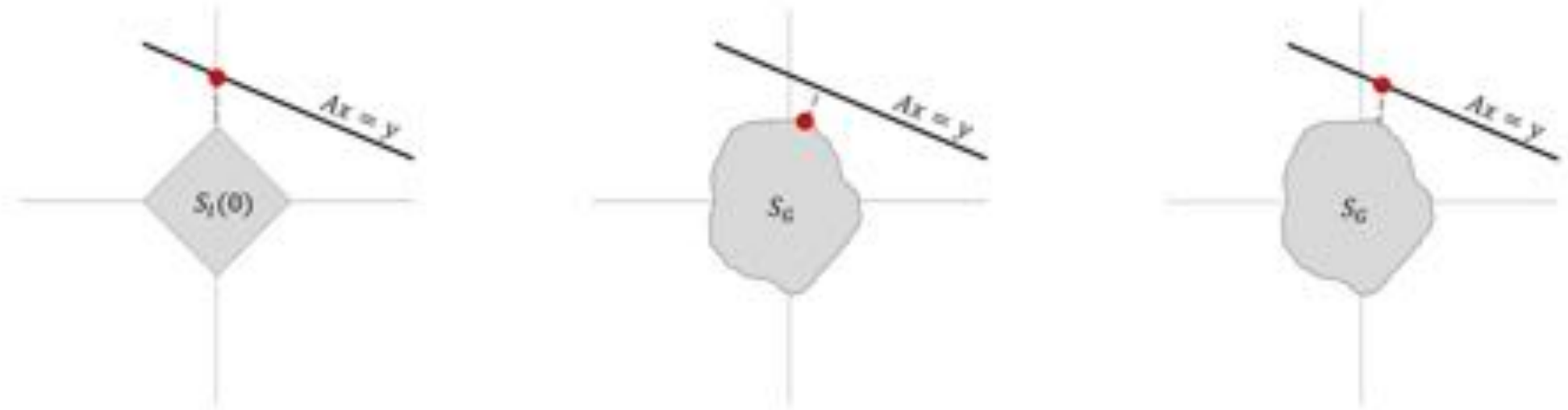
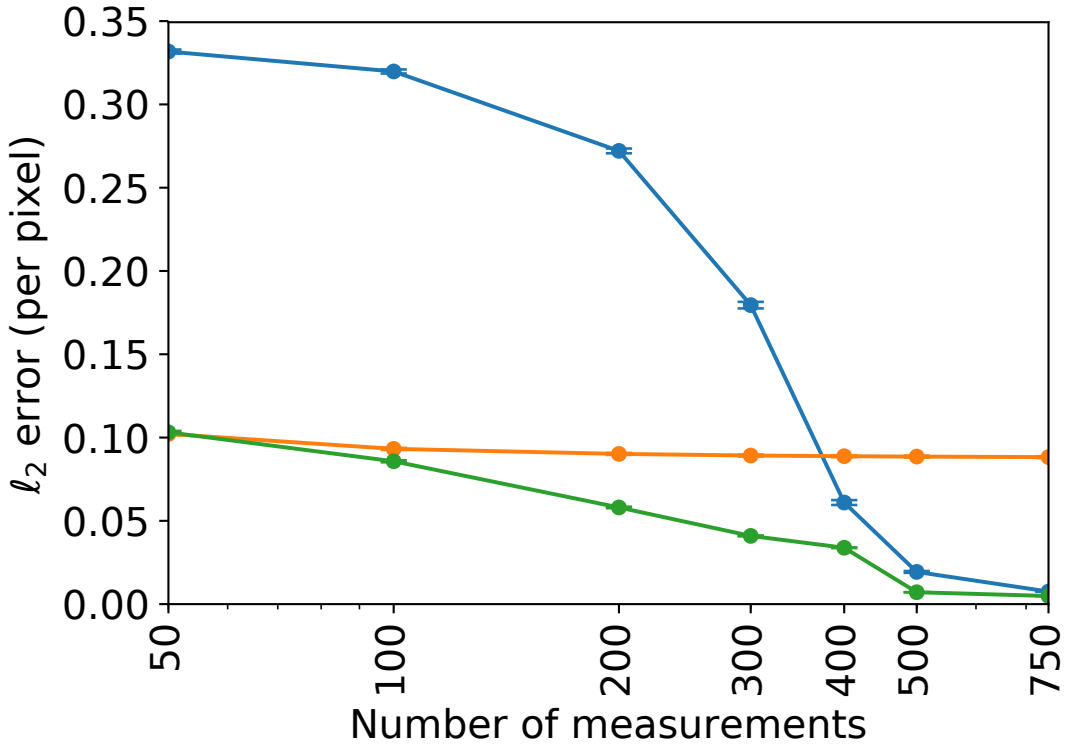


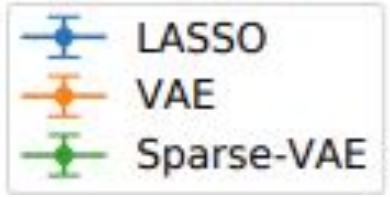
Figure 1. LASSO vs. Generative model vs. Sparse-Gen recovery. Unlike LASSO, Sparse-Gen imposes a stronger prior on the signals being sensed (**shaded grey regions**). Unlike generative model, the recovered signals are not constrained to lie on the range of the generator function (**red points**). Aspects of visualization due to eigenvalue conditions not shown for simplicity.

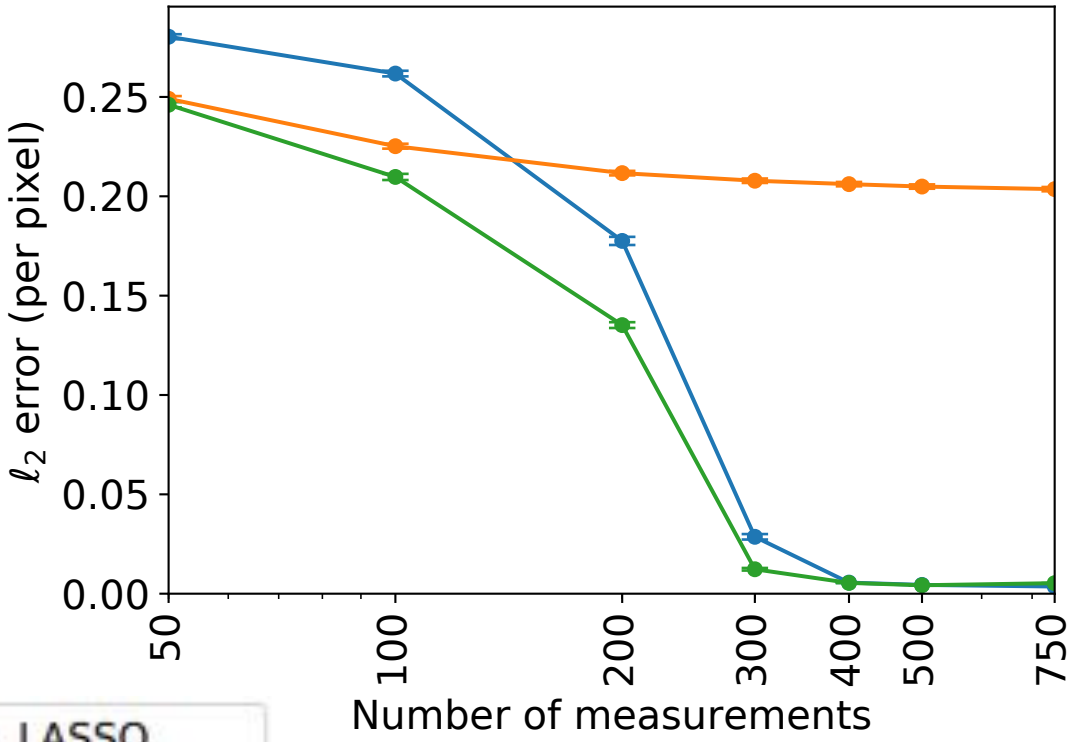
Bora et al., 2017, Dhar et al., 2018

Results



MNIST

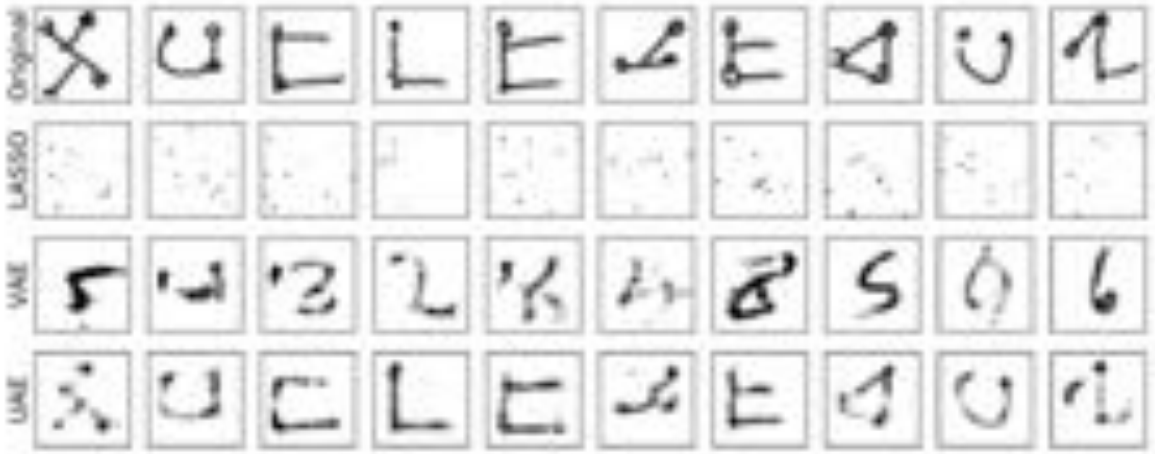




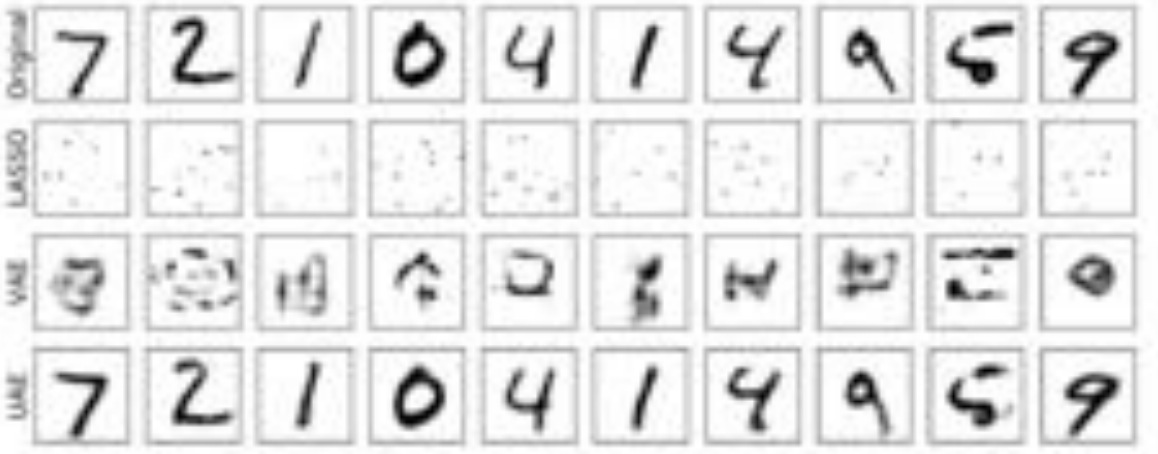
Omniglot

Transfer Compressive Sensing

Transferring from source, data-rich to target, data-hungry domains



(a) Source: MNIST, Target: Omniglot



(b) Source: Omniglot, Target: MNIST

Grover & Ermon, 2018

Beyond images, audio, text

Videos: Extremely high-dimensional spatiotemporal reasoning (Ranzato et al., 2014, Finn et al., 2018)

Graphs: Scalable, permutation-invariant architectures (Kipf et al., 2016, Hu et al., 2017, Bojchevski et al., 2018, Grover et al., 2018, Li et al., 2018, Wang et al., 2018, You et al., 2018)

Reinforcement learning: Online learning with respect to drifting distributions (Pfau & Vinayls, 2018, Kansky et al., 2017, Ostrovski et al., 2017)

Outlook

1. What is the killer application of generative models?
   - Model based-RL?
2. What is the right evaluation metric?
   - Fundamentally, it is unsupervised learning. Ill-defined.
3. Are there fundamental tradeoffs in inference ?
   - Sampling
   - Evaluation
   - Latent features
4. Provable guarantees?