# Deep Generative Models

Ehsan Abbasnejad

THE UNIVERSITY of ADELAIDE

# Outline

- Generative models
- Why using Deep Generative models?
  - Generative vs. Discriminative models
- Existing Methods:
  - Autoregressive Methods
  - Latent Variable Models
    - Variational Autoencoders (VAEs)
    - Generative Adversarial Networks (GANs)
- Problems with existing GANs and our approaches
  - 3D Hand pose Estimation
  - Other divergences (Dudley GANs)
  - Uncertainty in Generation (Uncertainty-aware GAN)
  - Density Estimator Adversarial Network
  - Imitation Learning
  - Causality

# Generative vs. Discriminative

- Generative models learn $p_\theta(\mathbf{x}, y)$
  Placing a **joint** distribution over all dimensions of the data

$$p_\theta(y|\mathbf{x})$$

- Discriminative models learn

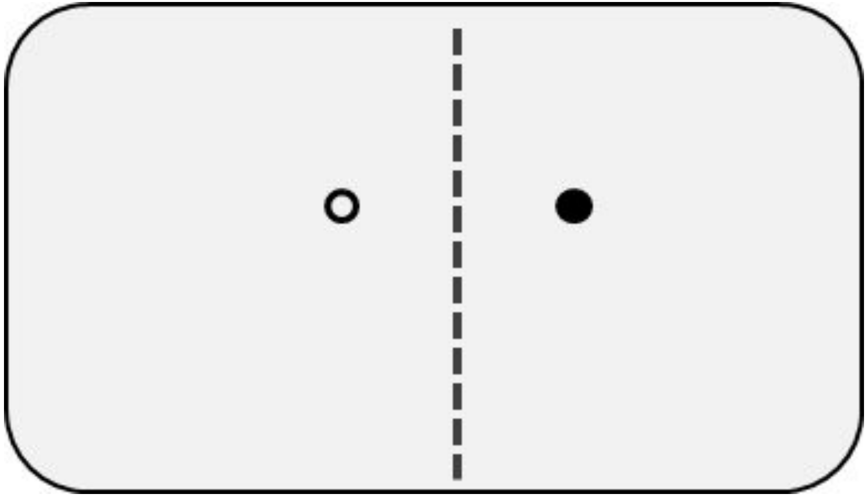
Remember:

$$\min_\theta \; \mathbb{E}_{\mathbf{x},y}[\ell(h(\mathbf{x};\theta), y)] = \int \ell(h(\mathbf{x};\theta), y) dp(\mathbf{x}, y)$$
$$\approx \frac{1}{n}\sum_{i=1}^n \ell(h(\mathbf{x}_i;\theta), y_i) \quad (\mathbf{x}_i, y_i) \sim p(\mathbf{x}, y)$$

and

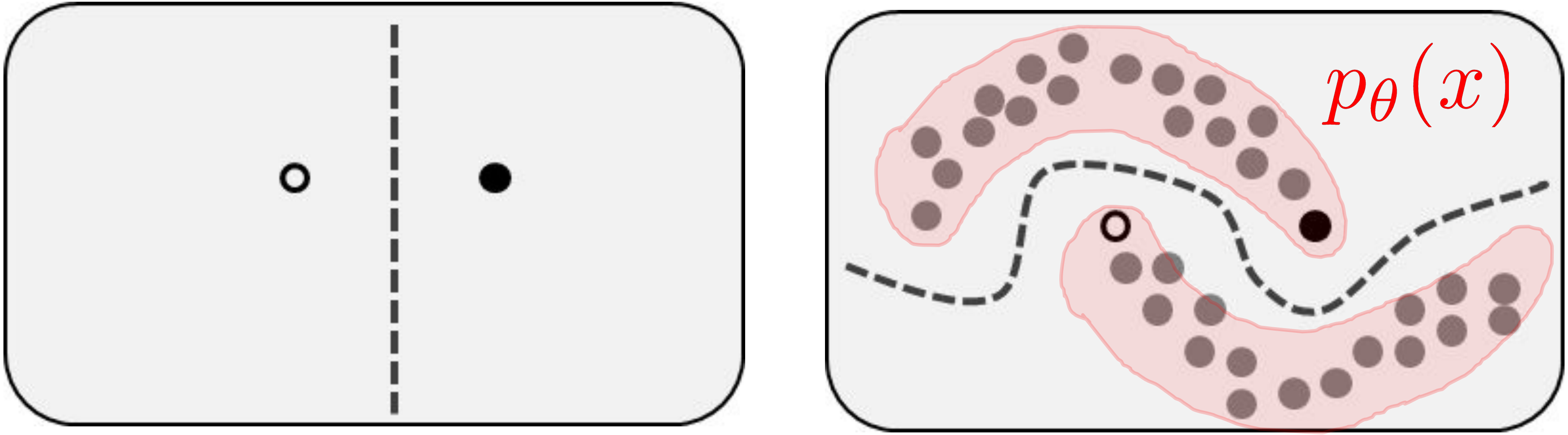
$$p(\mathbf{x}, y) = p(y|\mathbf{x})p(\mathbf{x})$$

# Generative models for *classification*



$$p_\theta(y|x)$$

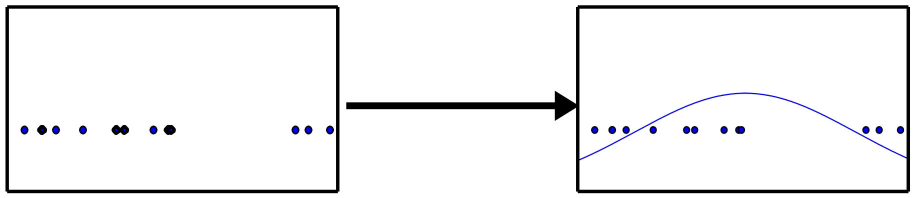
# Generative models for
*classification*




$$p_\theta(y|x) \longrightarrow p_\theta(y,x)$$

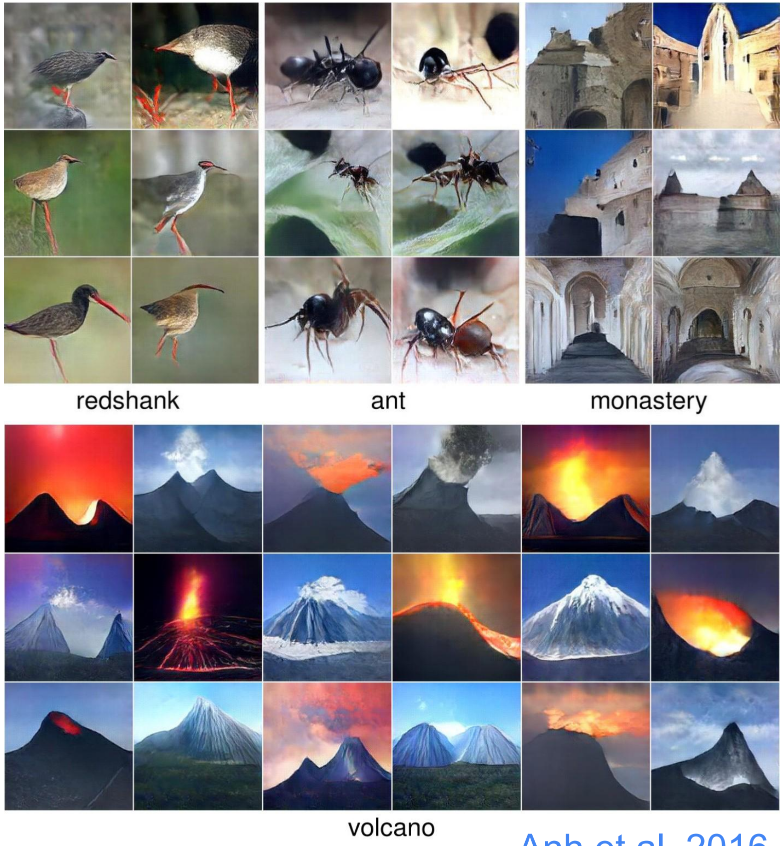
# Generative modeling

- Generative models allows to sample from some data distribution and learn a model that represents that distribution.
- Density estimation:
  - Probability for a sample (conditional or marginal)
  - Compare probability of examples
- Dimensionality reduction and (latent) representation learning
- Generate samples



# Learning to generate

## Images




Anh et al. 2016

# Learning to generate

Images

**Speech**

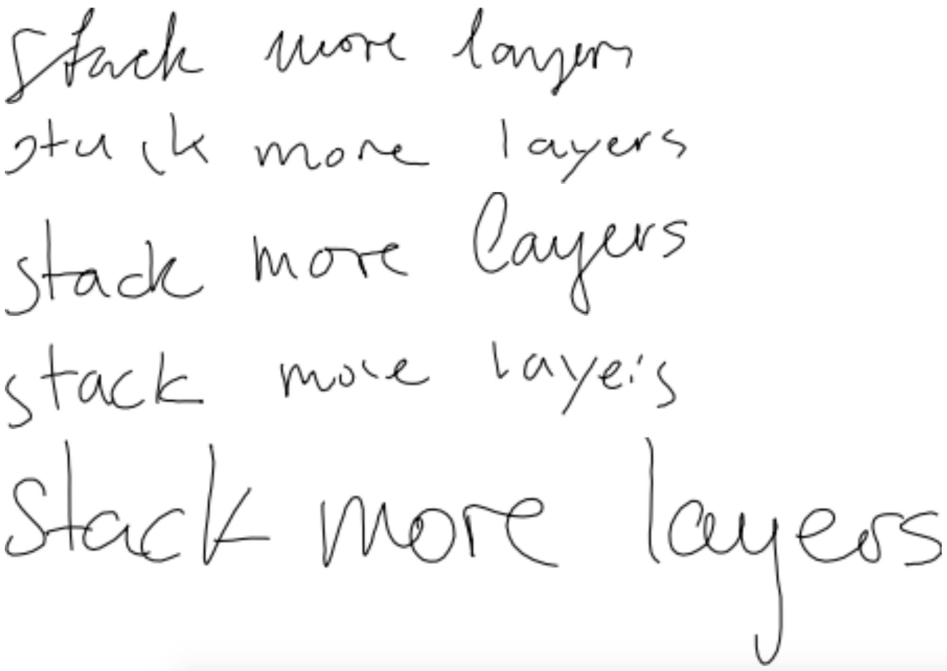
van der Oord et al.
2016

# Learning to generate

Images

Speech

**Handwriting**



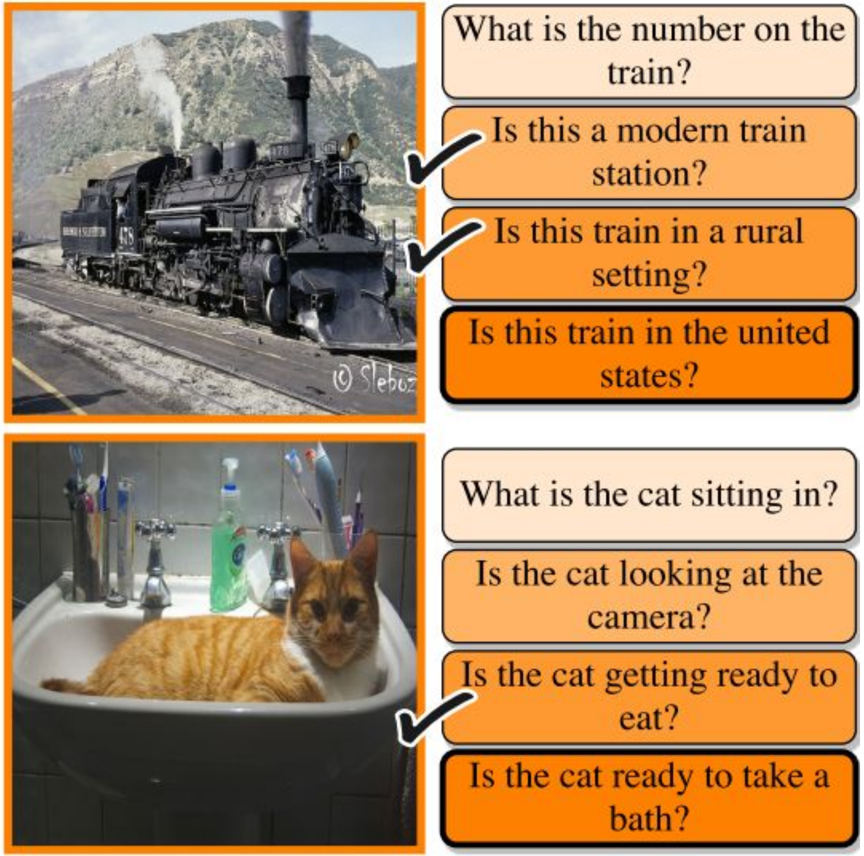

Graves 2013

# Learning to generate

Images
Speech

Handwriting

**Language**




Jain et al. 2017

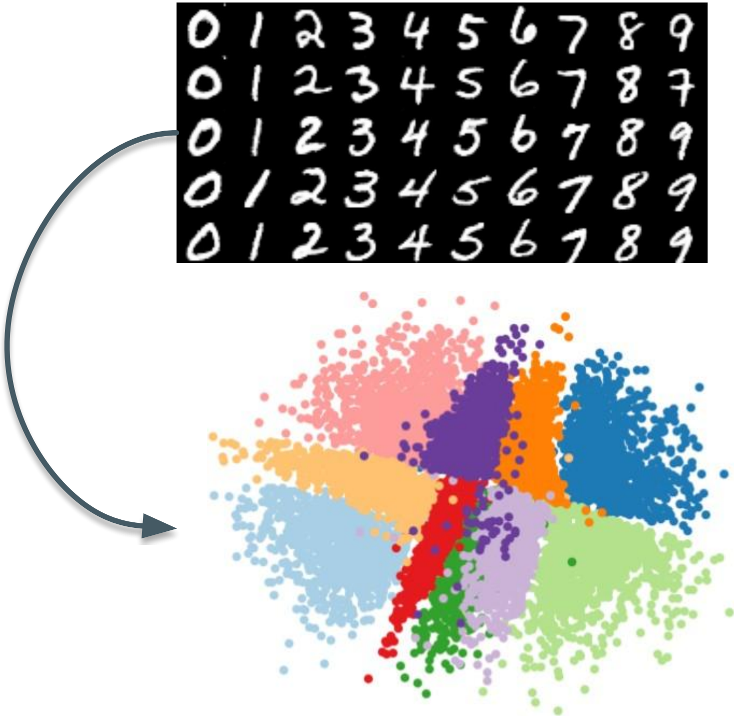
# Learning to generate

Images
Speech

Handwriting

Language

**Representation**



# Learning to generate

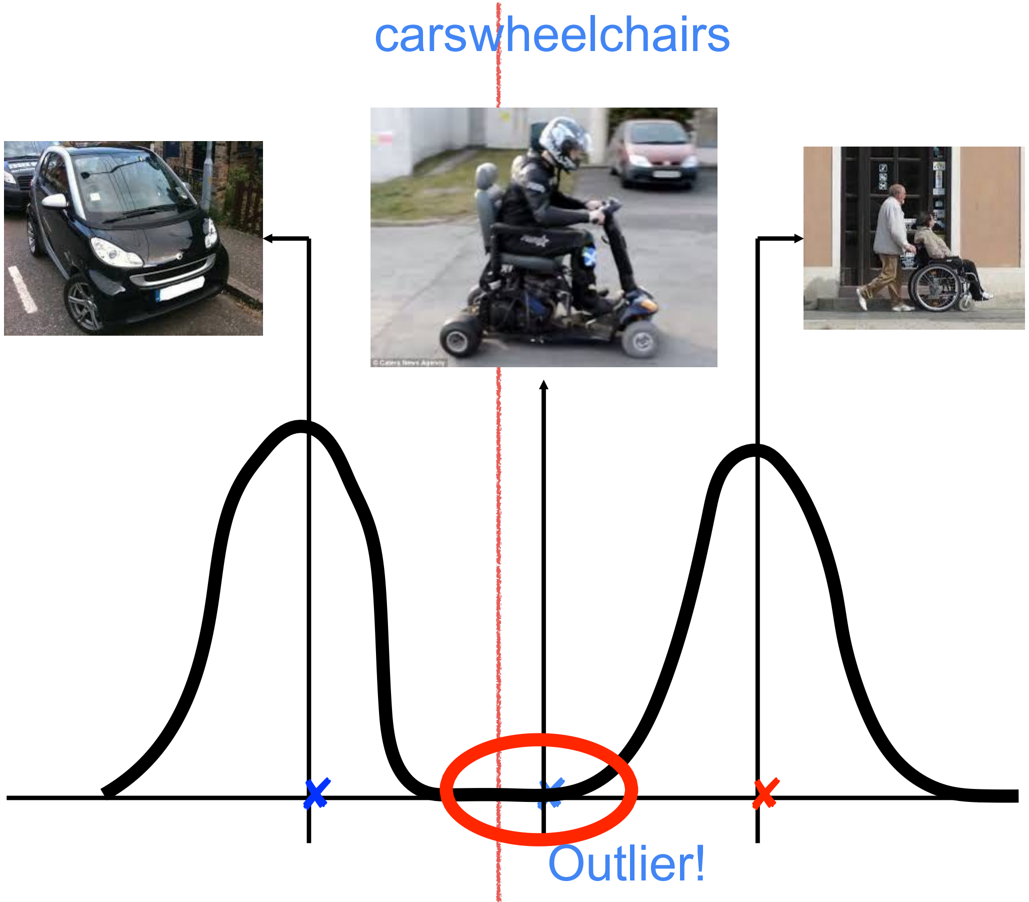
Images
Speech

Handwriting

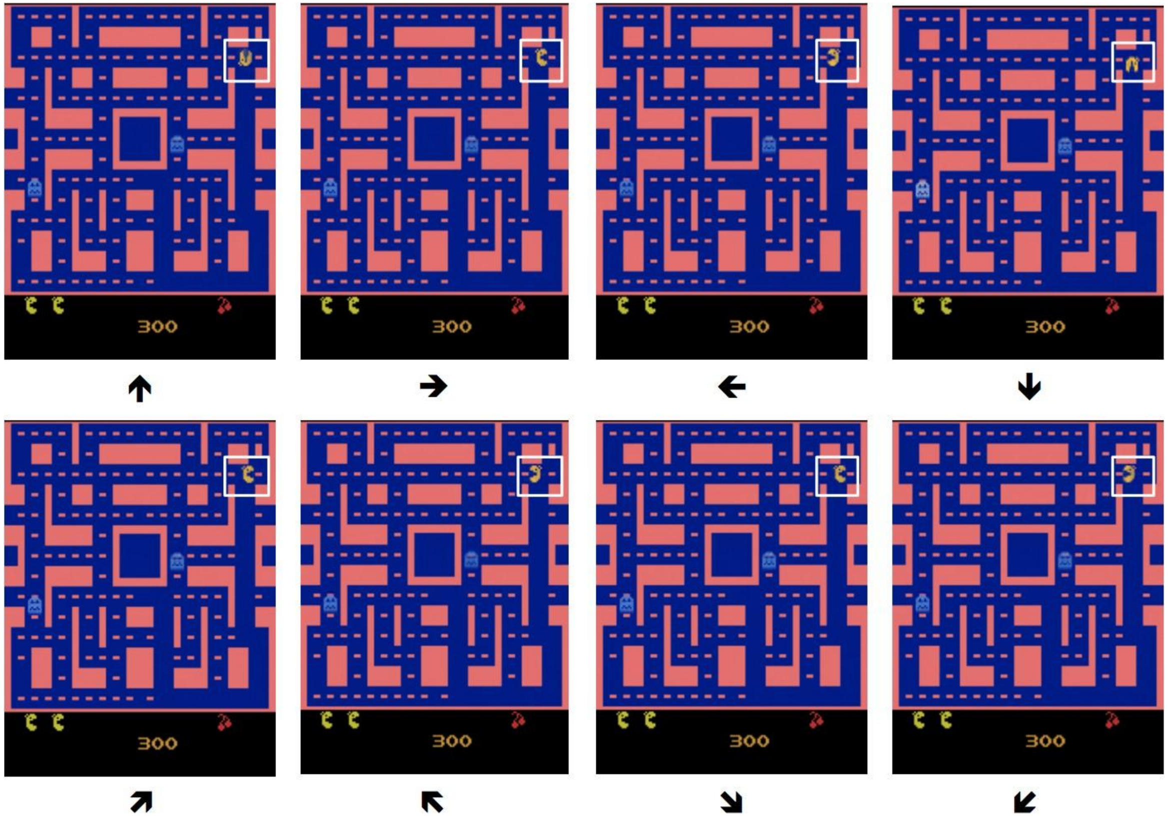
Language

Representation

**Outlier Detection**



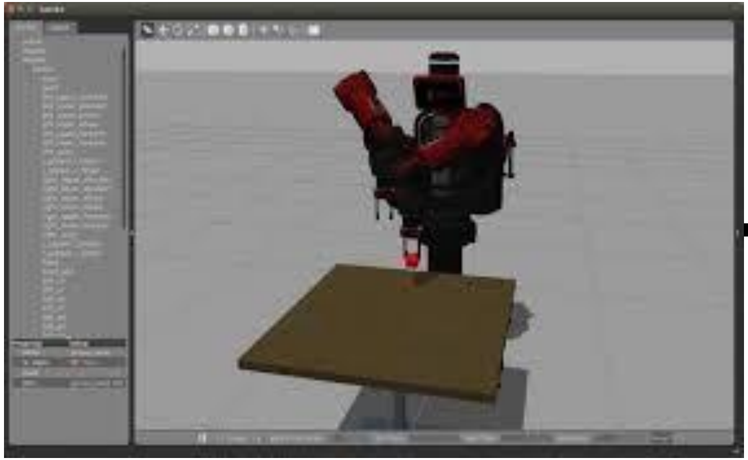
# Simulation, planning, reasoning




Oh et al. 2015

# Generation for Simulation

Supports Reinforcement Learning for Robotics:
Make simulations sufficiently realistic that learned policies can readily transfer to real-world application



Generative model →

Photo from IEEE Spectrum

**idea: learn to** *understand* **data through generation**

# Why generative models?

- Many tasks require structured output for complex data
  - Eg. Machine translation

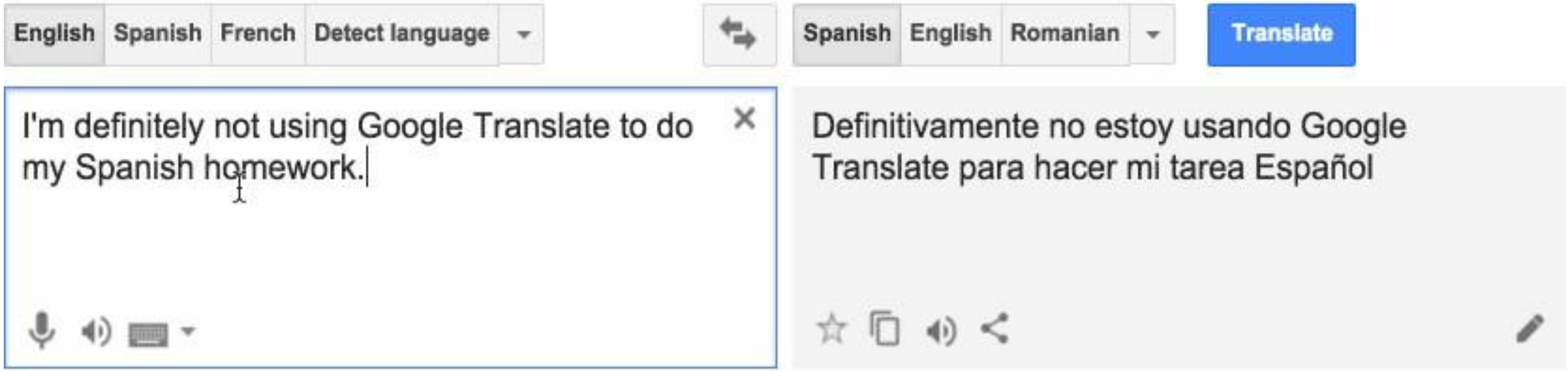



image credit: Adam Geitgey blog (2016) *Machine Learning is Fun.*

# Generative Methods:

Autoregressive models

- Deep NADE, PixelRNN, PixelCNN, WaveNet, Video Pixel  Network, etc.

Latent variable models

- Variational Auto encoders
- Generative Adversarial Networks

**… and Beyond**

# Setup

**Discriminative model**: given *n* examples $(x^{(i)}, y^{(i)})$
learn $h : X \rightarrow Y$

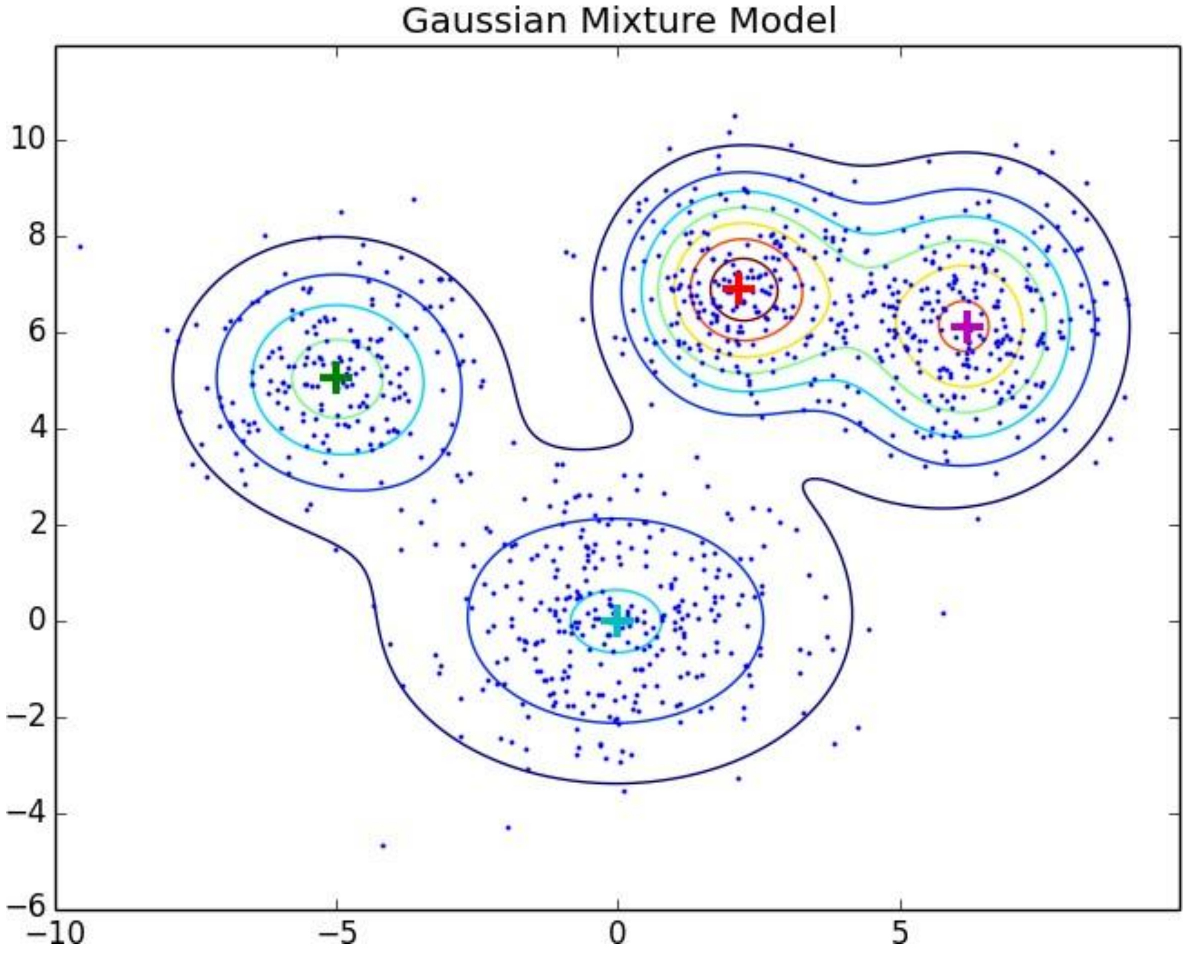
**Generative model**: given *n* examples $x^{(i)}$, recover $p(x)$

**Maximum-likelihood objective**: $\prod_i p_\theta(x) = \sum_i \log p_\theta(x)$

**Generation:** Sampling from $p_\theta(x)$

**Attempt 1:** learn $p_\theta(x)$ directly

$p_\theta(x)$

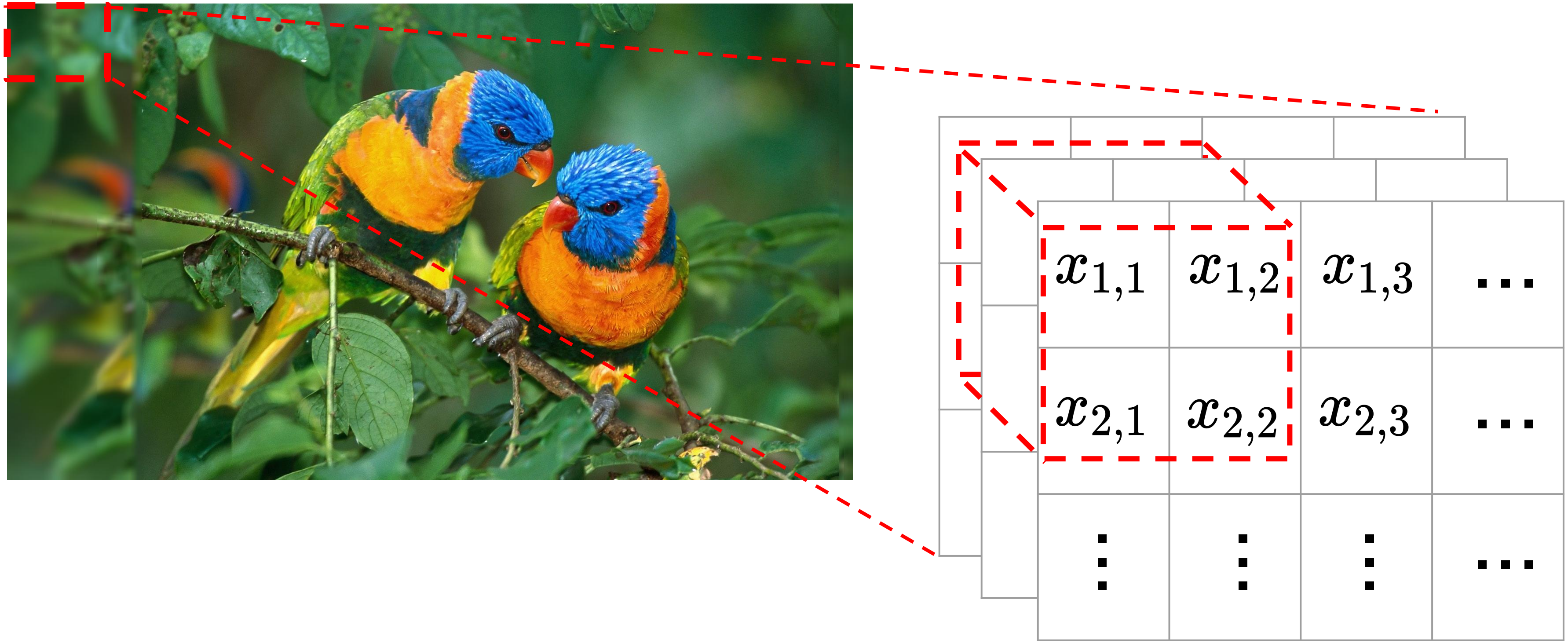


**Attempt 1:** learn $p_\theta(x)$ directly

**Problem:** We need to enforce that $\int_x p_\theta(x)dx = 1$

For most models (i.e. neural networks) this integral is intractable.

# Why Images are difficult?



# Autoregressive Models

**Factorize** dimension-wise:

$$p(x) = p(x_1)p(x_2|x_1)\dots p(x_n|x_1,\dots,x_{n-1})$$

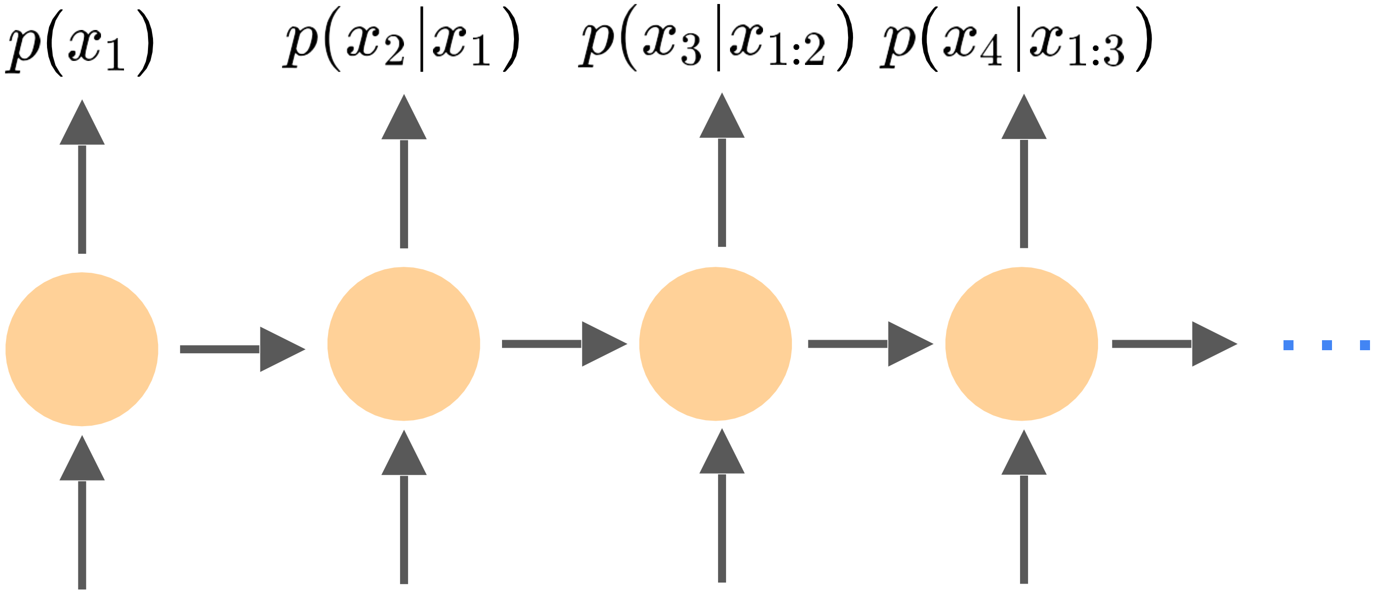
Build a "next-step prediction" model $p(x_n|x_1,\dots,x_{n-1})$

If x is **discrete**, network outputs a probability for each possible value

If x is **continuous**, network outputs parameters of a simple distribution (e.g. Gaussian mean and variance)… *or just discretize!*

**Generation:** sample one step at a time, conditioned on all previous steps

# RNNs for Autoregressive Modeling



# PixelRNN

Autoregressive RNN over pixels in an image

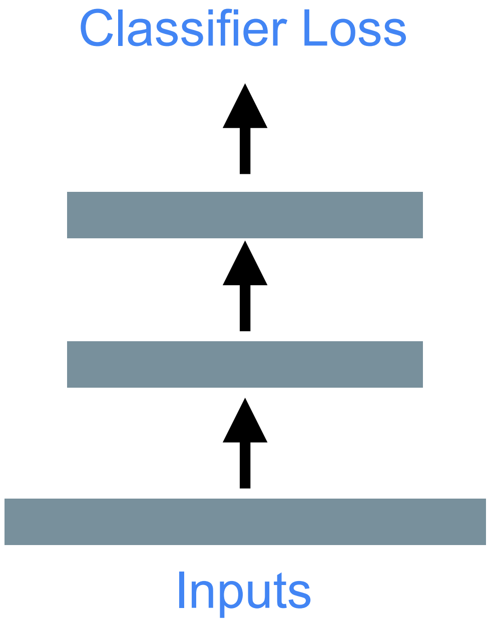
Models pixels as discrete-valued (256-way softmax at each step)

van der Oord et al. 2016

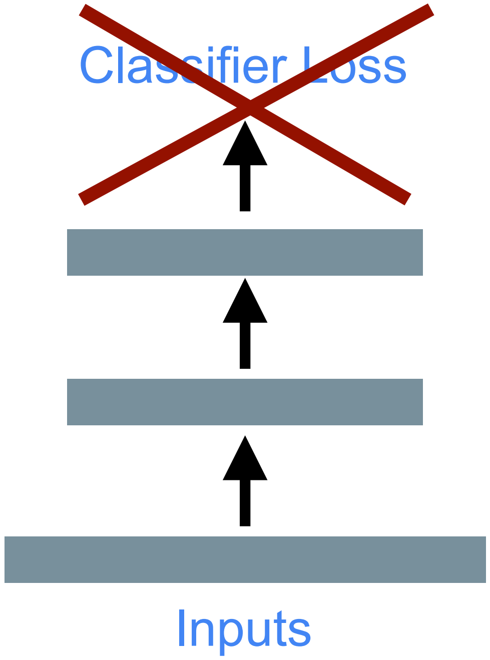
# Autoregressive models

- Autoregressive models are powerful density estimators, *but:*
    - Sequential generation can be slow
    - Doesn’t closely reflect the “true” generating process
    - Tends to emphasize details over global data
    - Not very good for learning representations

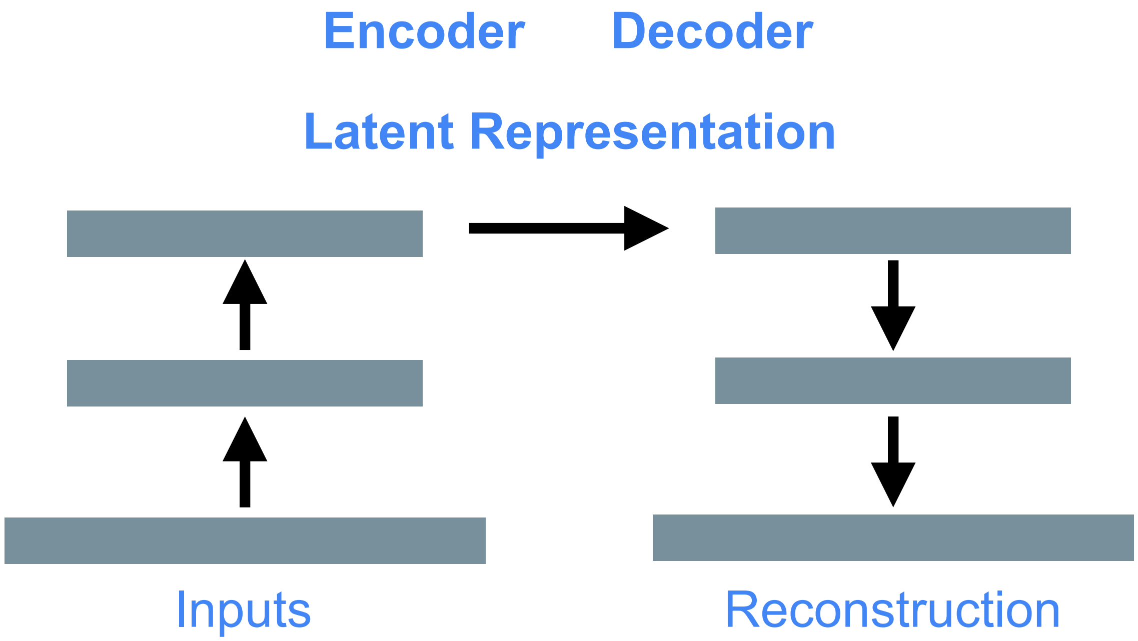
# Autoencoders



# Autoencoders

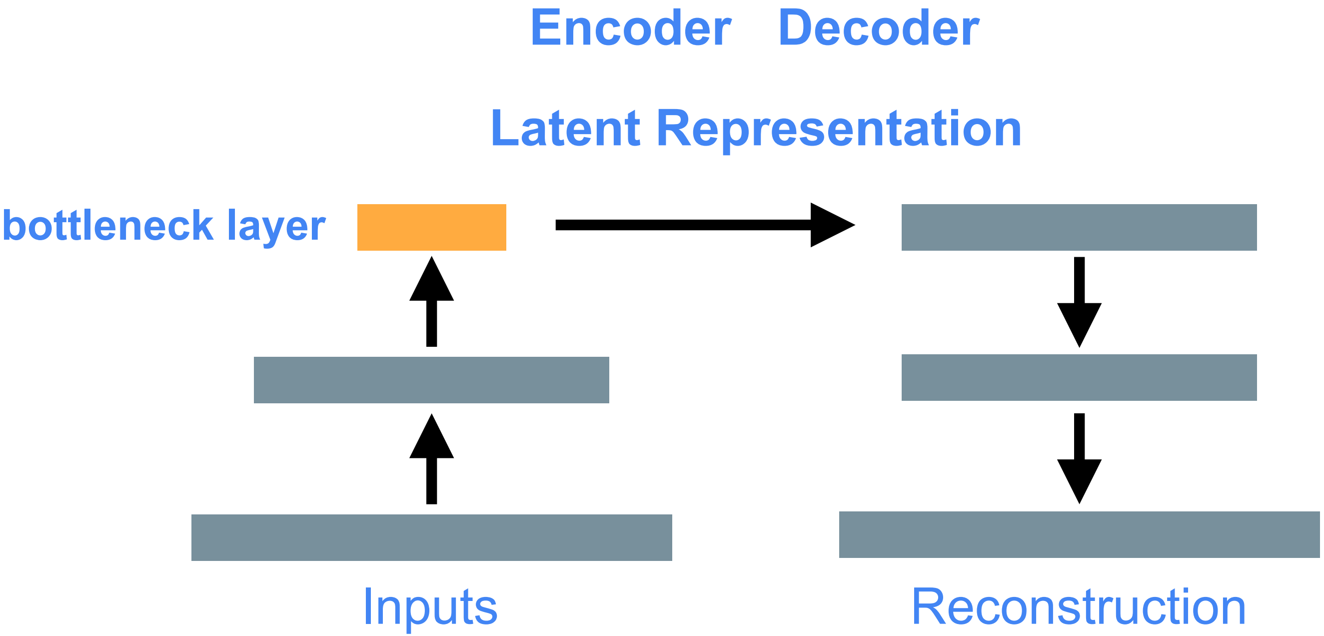


# Autoencoders




$$L = (x - \hat{x})^2$$

# Autoencoders




$$L = (x - \hat{x})^2$$

# Autoencoders

Reconstruction loss forces hidden layer to represent information about the input

Bottleneck hidden layer forces network to learn a compressed latent representation

# Latent Variable Models: compression as implicit generative modeling

# Variational Autoencoders (VAEs)

Generative extension of autoencoders which allow sampling and estimating probabilities

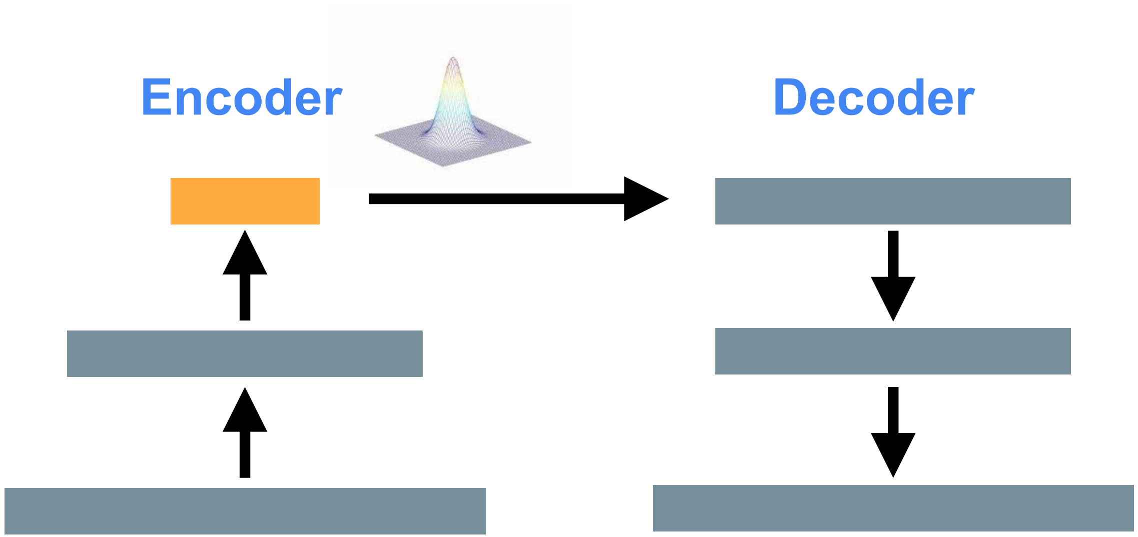
"Latent variables" with fixed prior distribution $p(z)$

Probabilistic encoder and decoder: $q(z|x), p(x|z)$

Trained to maximize a lower bound on log-probability:

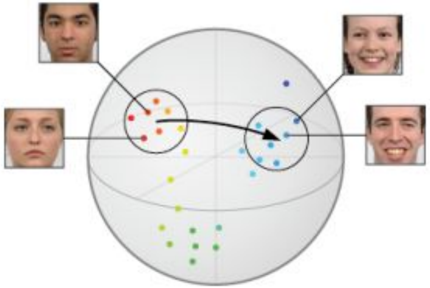
$$\log p(x) \geq \mathbb{E}_{z \sim q(z|x)} [\log p(x|z) + \log p(z) - \log q(z)]$$

# Variational Autoencoders




$$\log p(x) \geq \mathbb{E}_{z \sim q(z|x)} [\log p(x|z) + \log p(z) - \log q(z)]$$

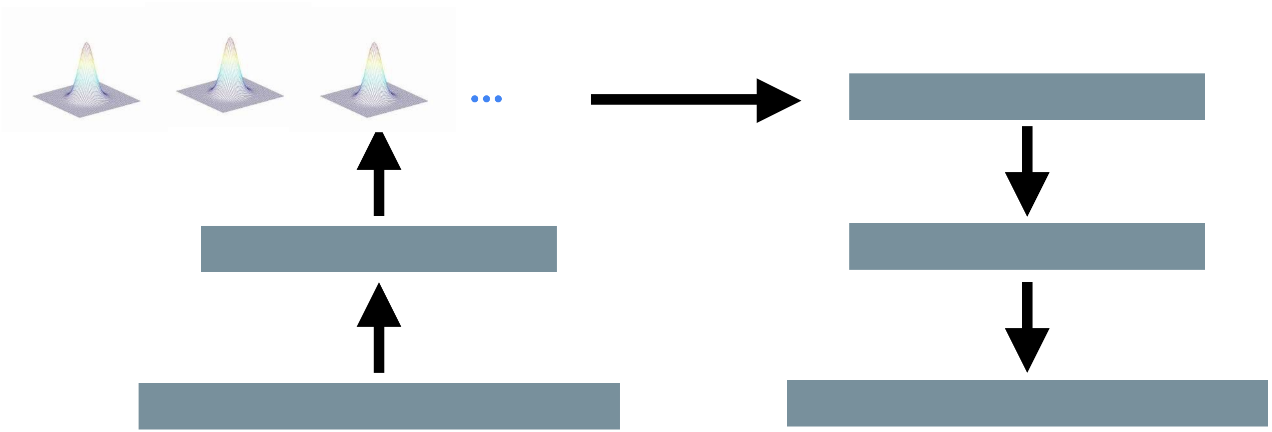
# Interpolation in the Latent Space



Tom White
2016

# Infinite Variational Autoencoders

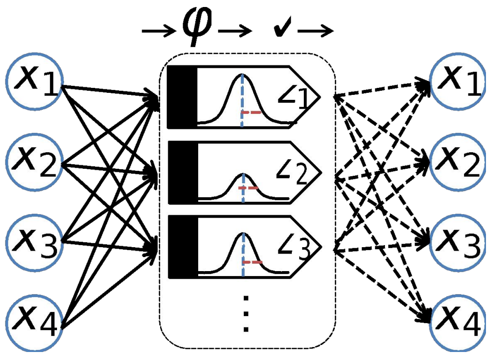
Encoder Decoder



$$\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}_i)}[\log p_{\boldsymbol{\theta}_{\mathbf{c}_i}}(\mathbf{x}_i, \mathbf{z}_{\mathbf{c}_i}) - \log q_\phi(\mathbf{z}_{\mathbf{c}_i}|\mathbf{x}_i)] \approx \frac{1}{L}\sum_{\ell=1}^{L} \log p_{\boldsymbol{\theta}_{\mathbf{c}_i}}(\mathbf{x}_i, \mathbf{z}^{\ell}_{\mathbf{c}_i}) - \log q_\phi(\mathbf{z}^{\ell}_{\mathbf{c}_i}|\mathbf{x}_i)$$

# Infinite VAEs

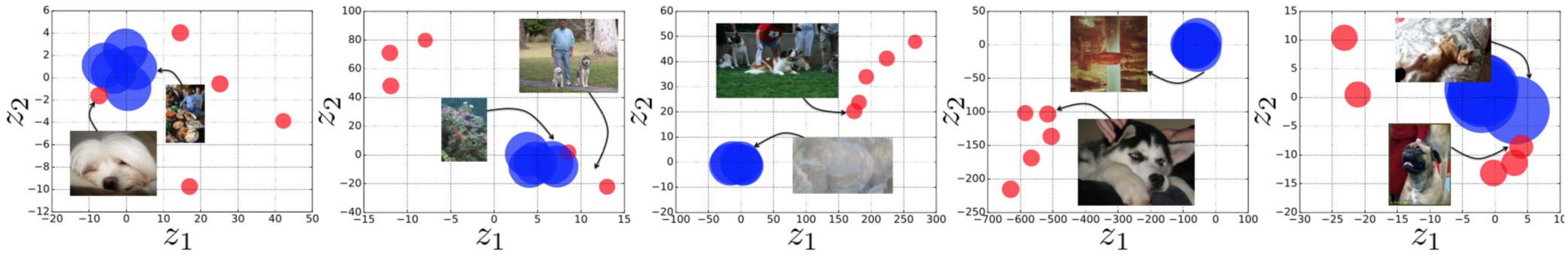
- Infinite dimensional latent space
- Combine potentially infinite number of VAEs
- Infinite dimensional latent space
- Potential to capture all the diversity in the data



# Mixture of Autoencoders

If we train our model with dogs, it is uncertain about other images

Mapping the image to the hidden space



# Problems with VAEs

- Encoder and decoder's output distributions are typically limited (diagonal-covariance Gaussian or similar)

- This prevents the model from capturing fine details and leads to blurry generations

Andrej Karpathy 2015

# Problems with VAEs

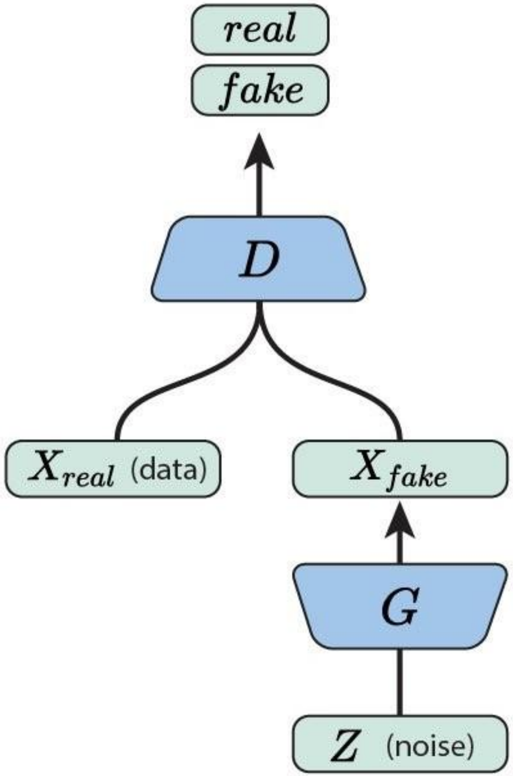
- Encoder and decoder's output distributions are typically limited (diagonal-covariance Gaussian or similar)
- This prevents the model from capturing fine details and leads to blurry generations
- **Solution:** use autoregressive networks in encoder and decoder

# Generative Adversarial Networks (GANs)

Generative Adversarial networks are a way to make a generative model by having two neural networks compete with each other

The **discriminator** tries to distinguish genuine data from forgeries created by the generator.

The **generator** turns random noise into imitations of the data, in an attempt to fool the discriminator



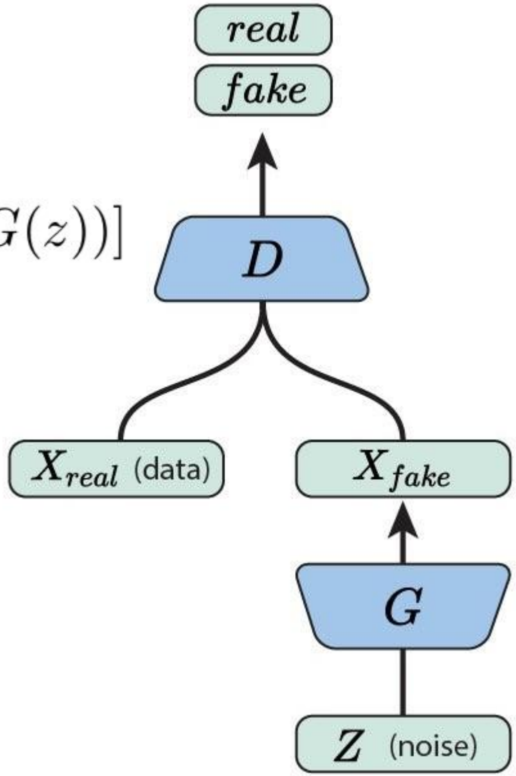
# Generative Adversarial Networks (GANs)

Problem setup:

$$\min_G \max_D \quad E_{x\sim p_X}[\log D(x)] + E_{z\sim p_Z}[\log(1 - D(G(z)))]$$

$P_X$ Data Distribution

$P_G(z)$ Model Distribution



# Generative Adversarial Networks (GANs)

**Algorithm 1** Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, $k$, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

**for** number of training iterations **do**

**for** $k$ steps **do**

- Sample minibatch of $m$ noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of $m$ examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[ \log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right].$$

**end for**

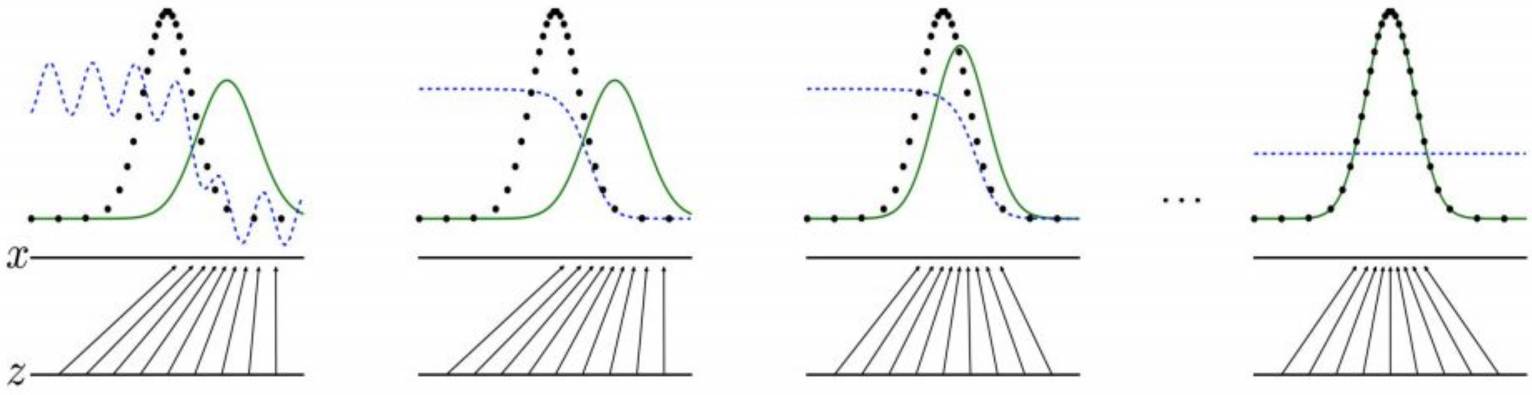
- Sample minibatch of $m$ noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

**end for**

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

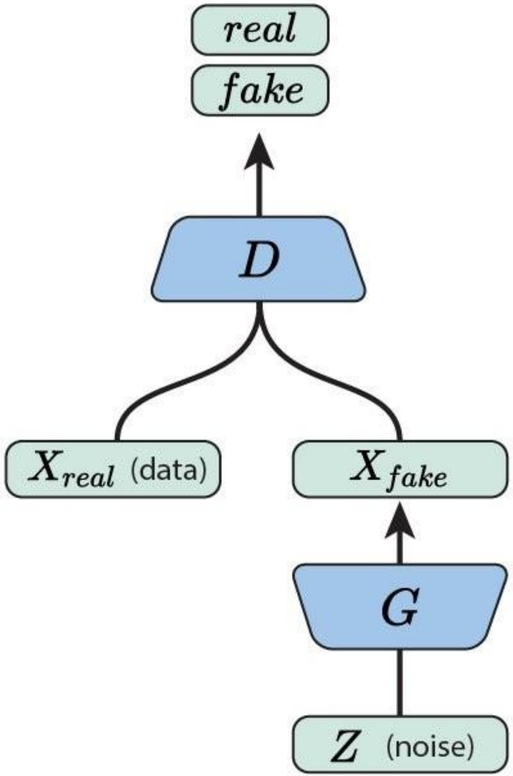
# Generative Adversarial Networks (GANs)



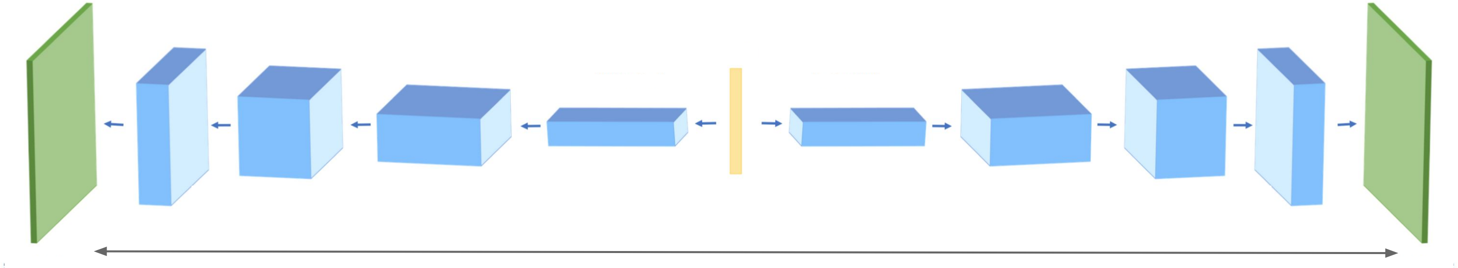
# Generative Adversarial Networks (GANs)

Why GANs?

- Sampling is straightforward
- Robust to overfitting since Generator never sees training data
- GANs are good at capturing the modes
- No need for Markov Chain
- No variational bound
- Produce better samples

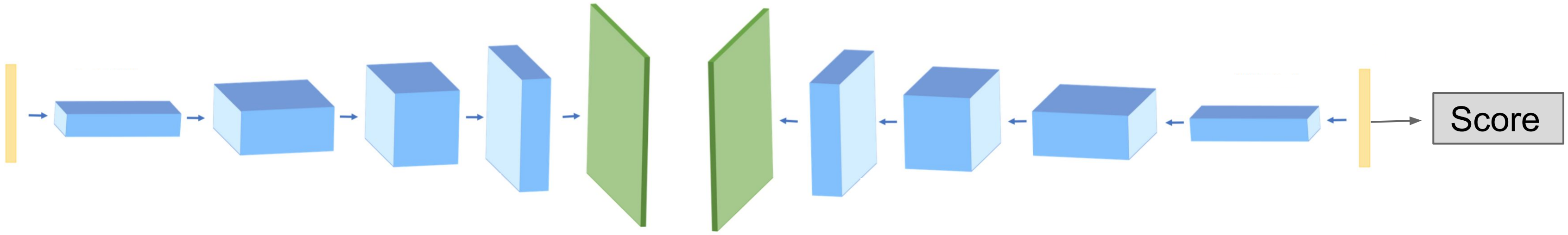


# VAEs



# vs

# GANs



# VAEs vs GANs

- VAEs maximize a lower bound on the likelihood of the data
- GANs maximize a "learned score" for the generated samples

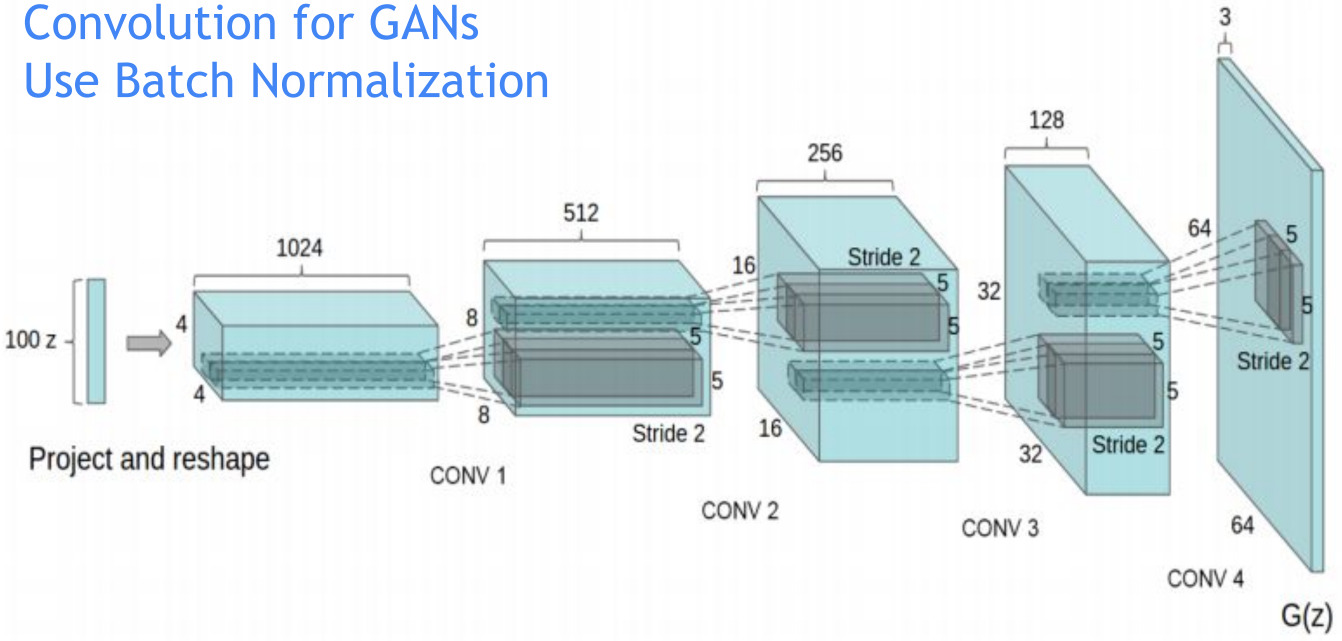
- VAEs minimize the KL-divergence between the generated and real data
- GANs learn the loss based on the divergence between real/fake examples (will discuss)

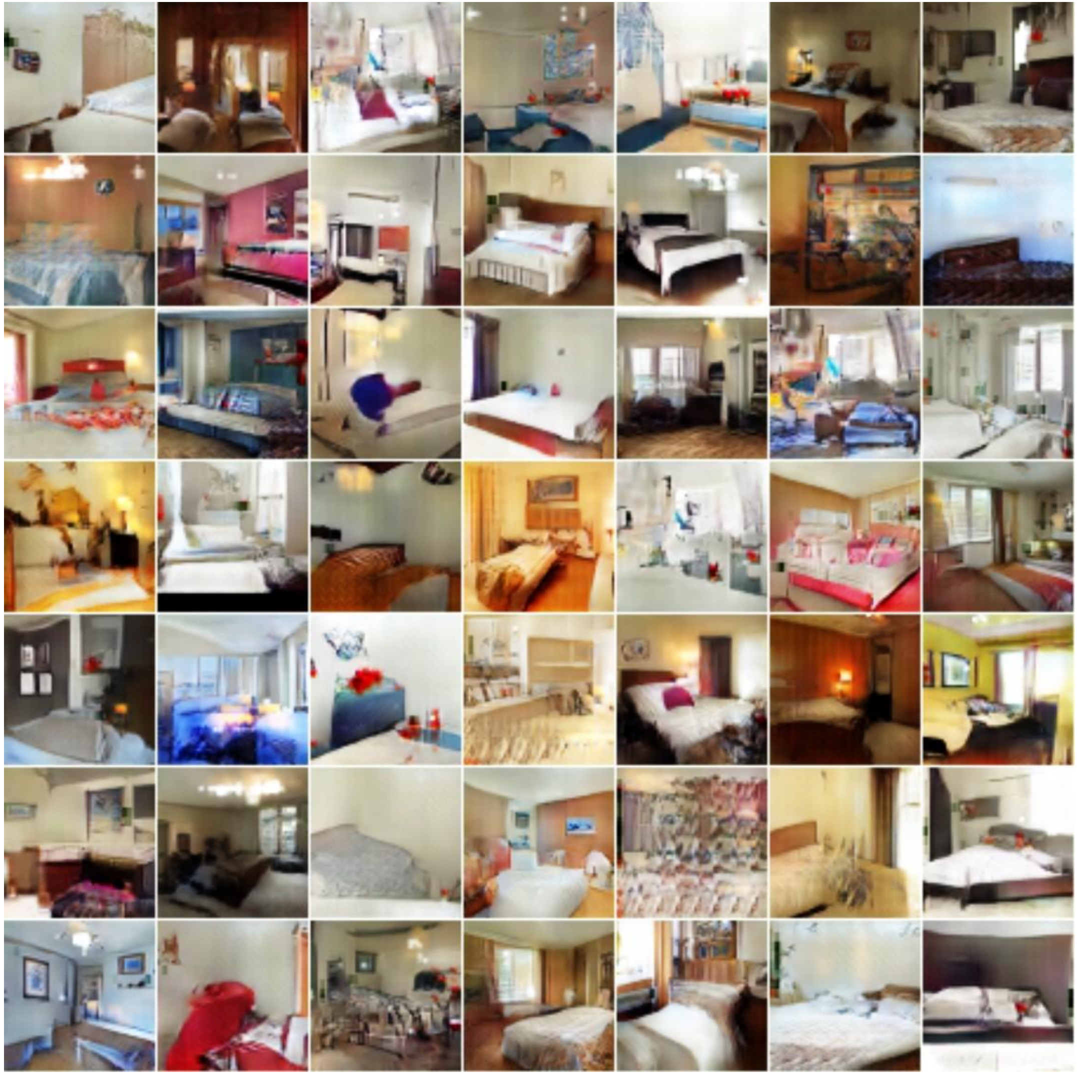
- VAEs generate data conditioned on the input
- GANs' generator never sees real data

- VAEs are trained as a single function (neural network)
- GANs are trained as two individual network with their own loss
- (Both follow a symmetric design)

# DCGAN

- Convolution for GANs
- Use Batch Normalization



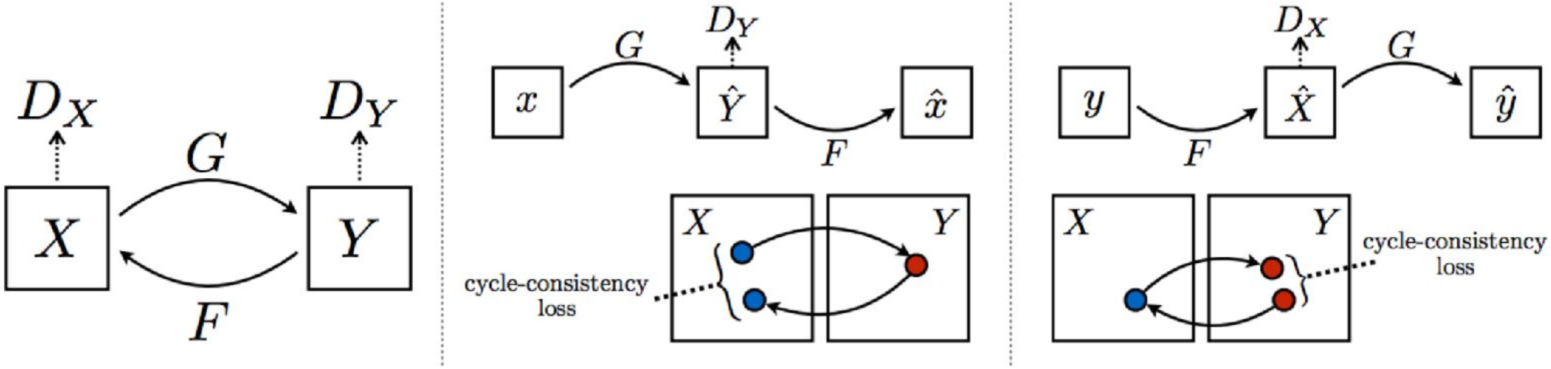


redshank ant monastery

volcano

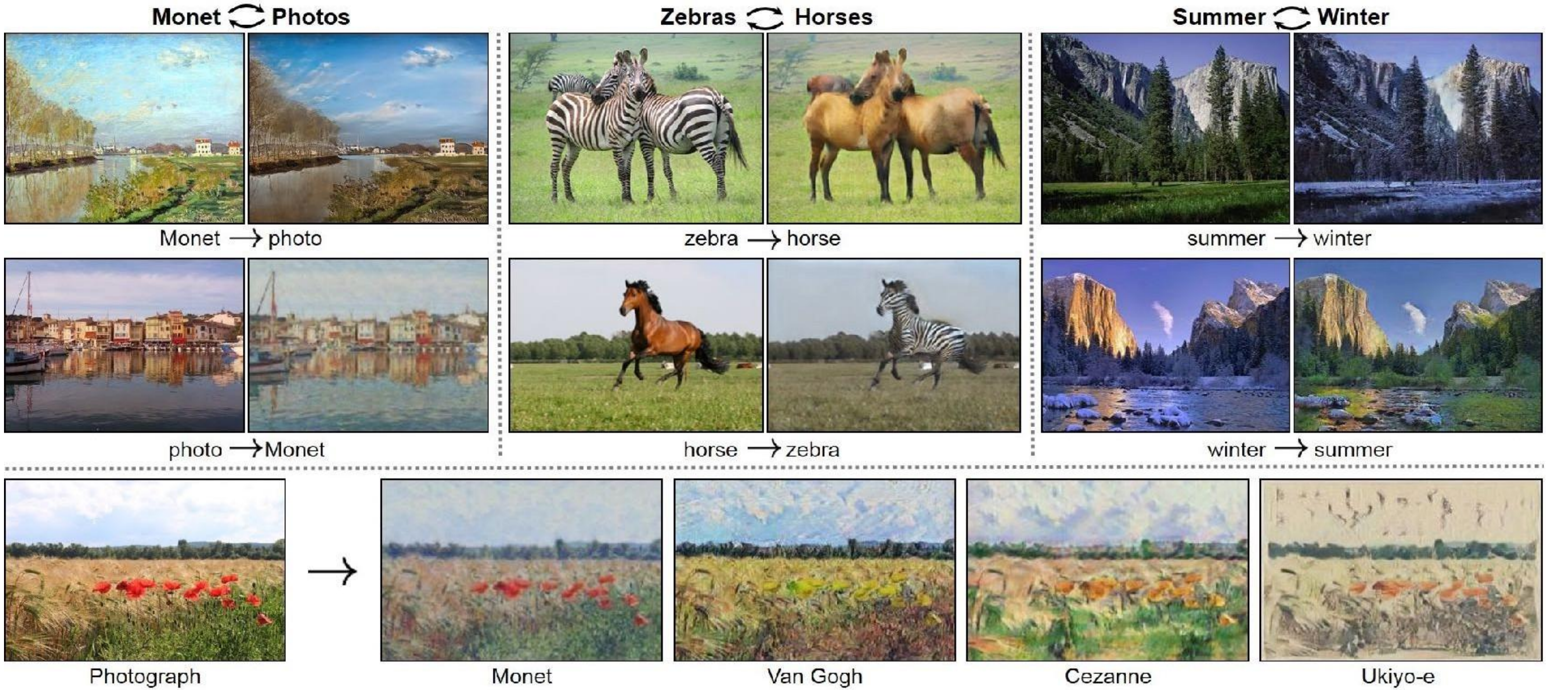
# cycleGAN: Adversarial training of domain transformations

- CycleGAN learns transformations across domains with unpaired data.
- Combines GAN loss with “cycle-consistency loss”: L1 reconstruction.

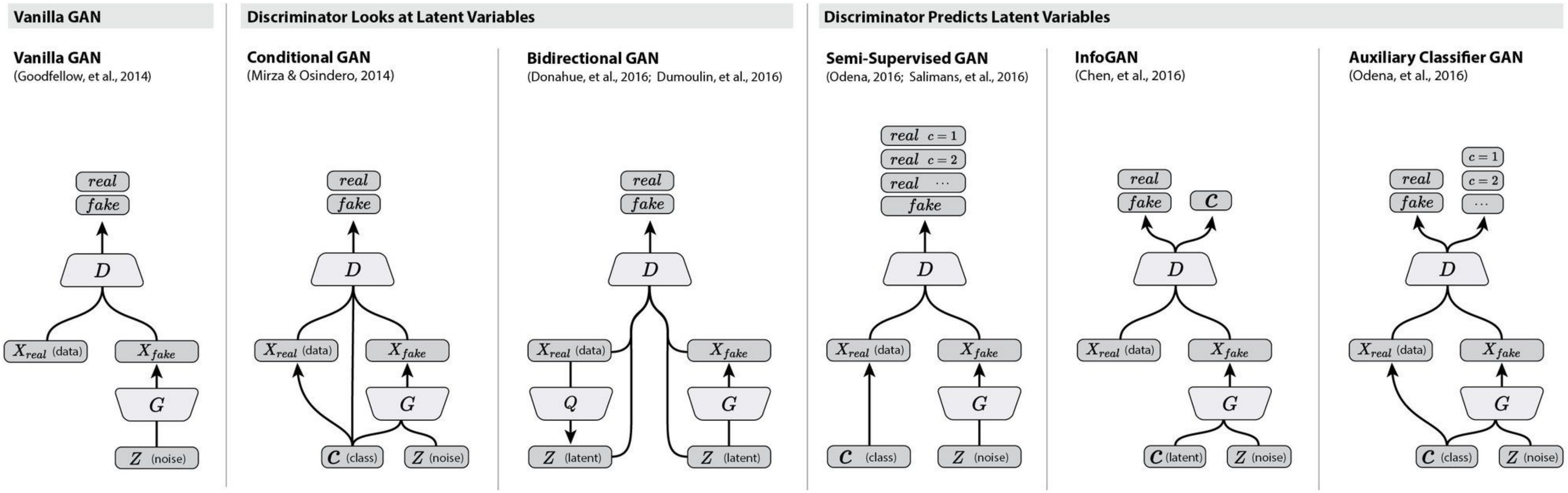



Zhu et al 2017.

# CycleGAN for unpaired data



# Various Architectures




Chris Olah, 2016

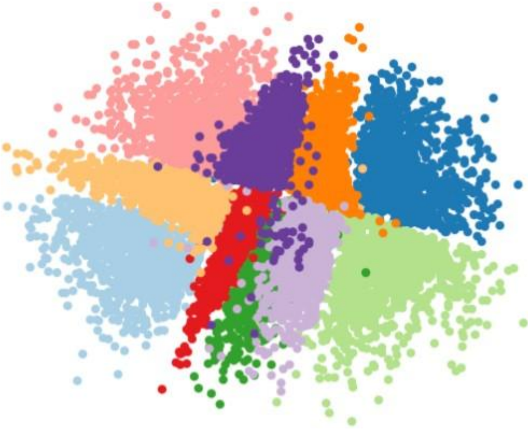
# Representation learning

(c) Varying $c_2$ from $-2$ to 2 on InfoGAN (Rotation)

(a) Varying $c_1$ on InfoGAN (Digit type)

(d) Varying $c_3$ from $-2$ to 2 on InfoGAN (Width)

Chen et al.
2016



Sønderby et al. 2016

# High dimensional faces using GANs

- Samples of 1024 * 1024

# Fake videos


# Fake videos

# Fake videos

# Text to Image

a red and white bird with a small beak

a yellow and black bird with long beak

a small yellow and black bird with red beak

a small bird with blue crown, back and white belly

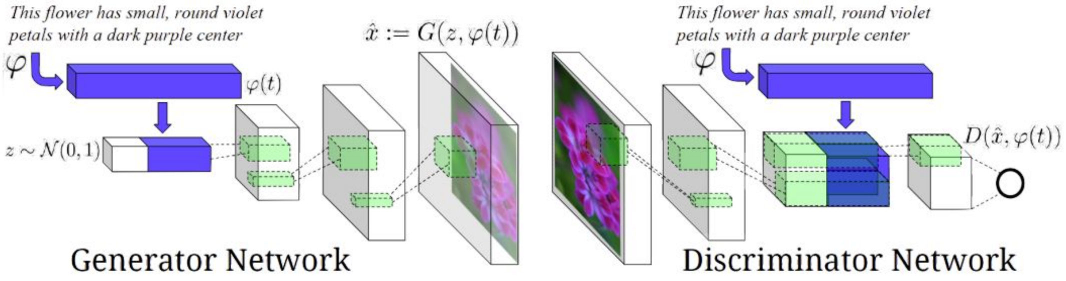
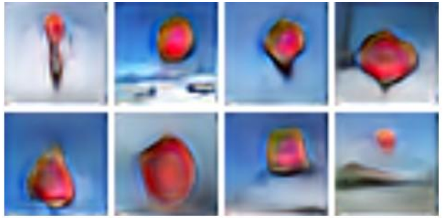
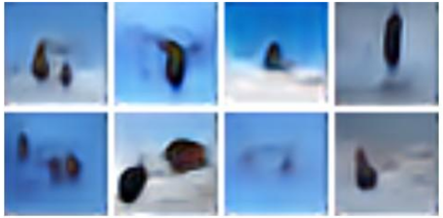



Figure 2 in the original paper.

# Generating Implausible Scenes from Captions

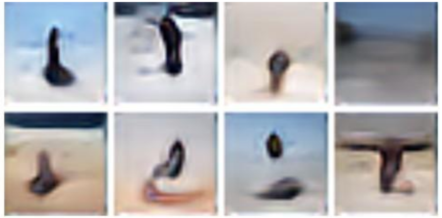


A stop sign is flying in blue skies.



A herd of elephants flying in the blue skies.

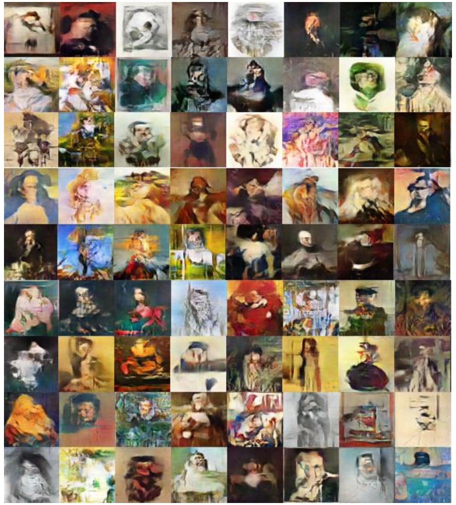
A toilet seat sits open in the grass field.



A person skiing on sand clad vast desert.

Mansimov et al. 2015

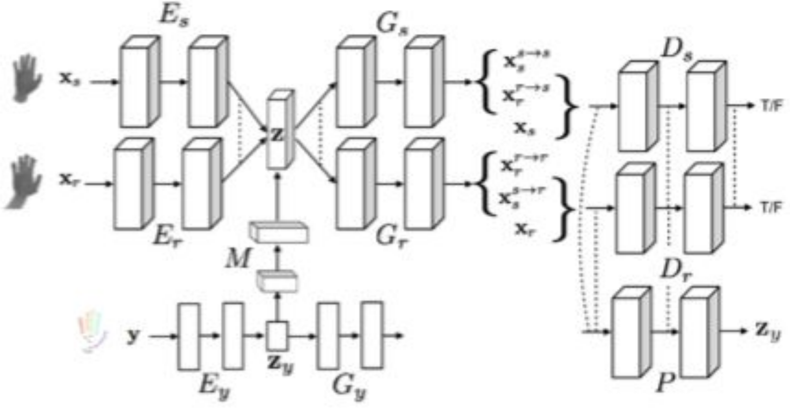
# GANs for art: GANGogh

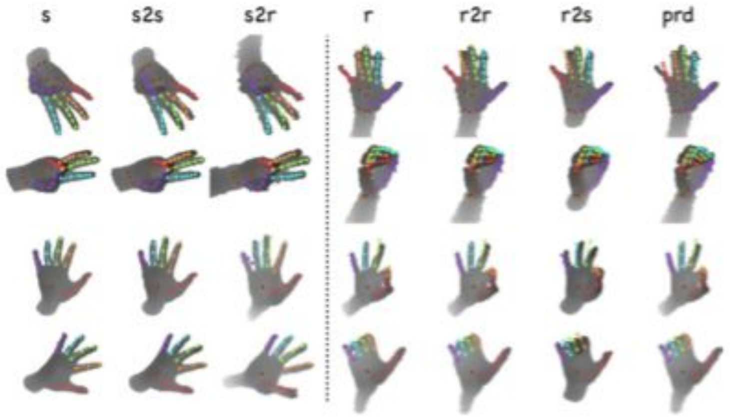


# GANs for Games: Fully computer Generated Game

We have many orders of magnitude more data than labels; **unsupervised learning is important**.

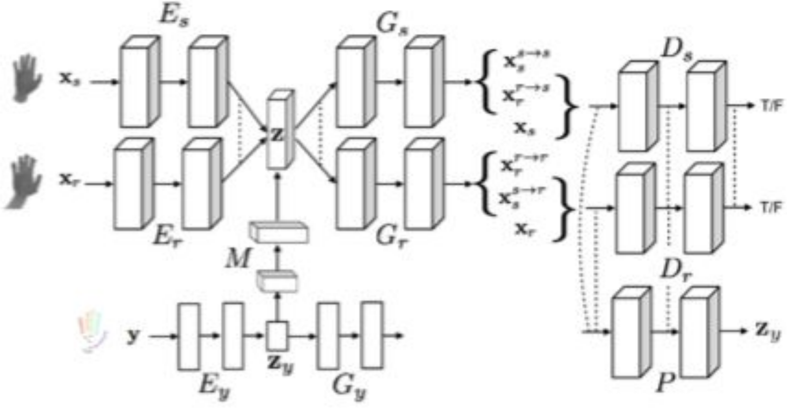
# 3D Hand pose estimation

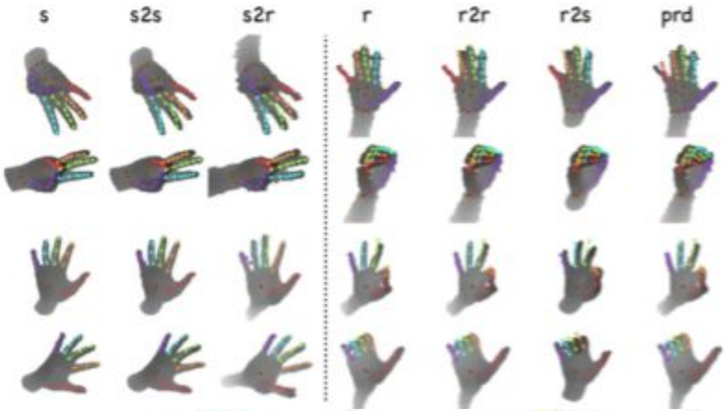


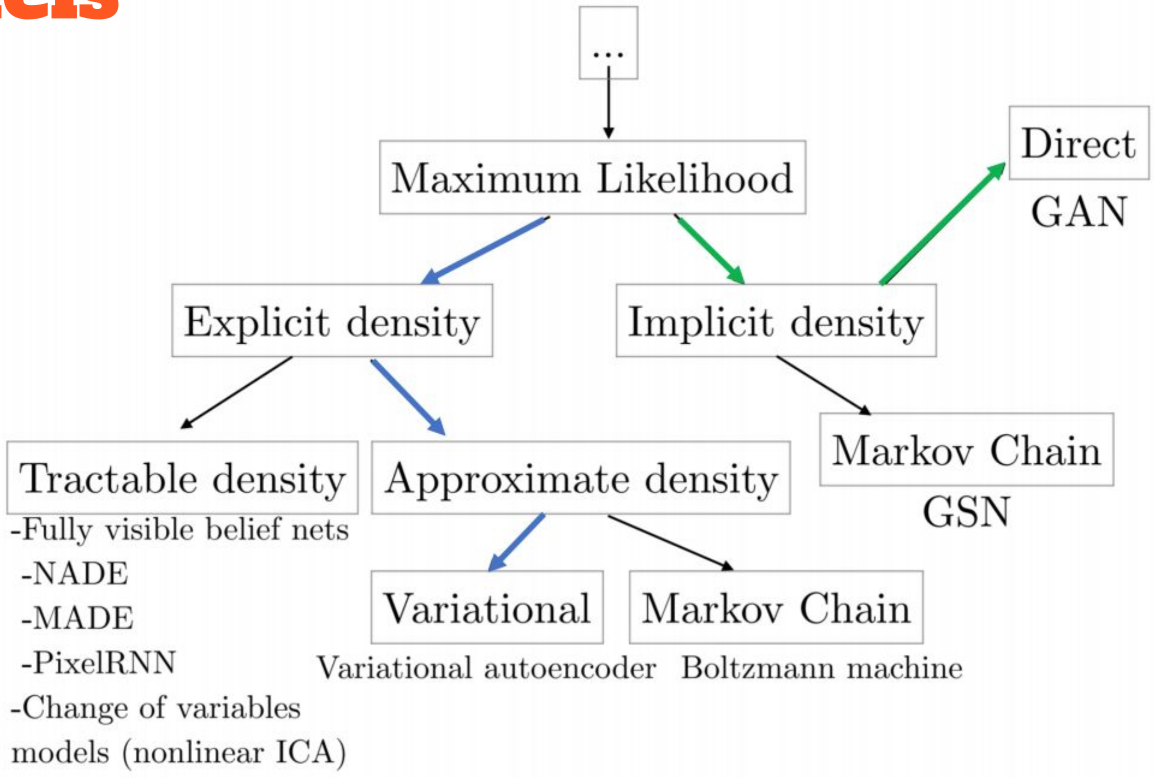

- VAEs and GANs can be combined to learn a better latent space
- Hand pose from the depth images
- Combination of synthetic and real depth images
- From unpaired images

# 3D Hand pose estimation



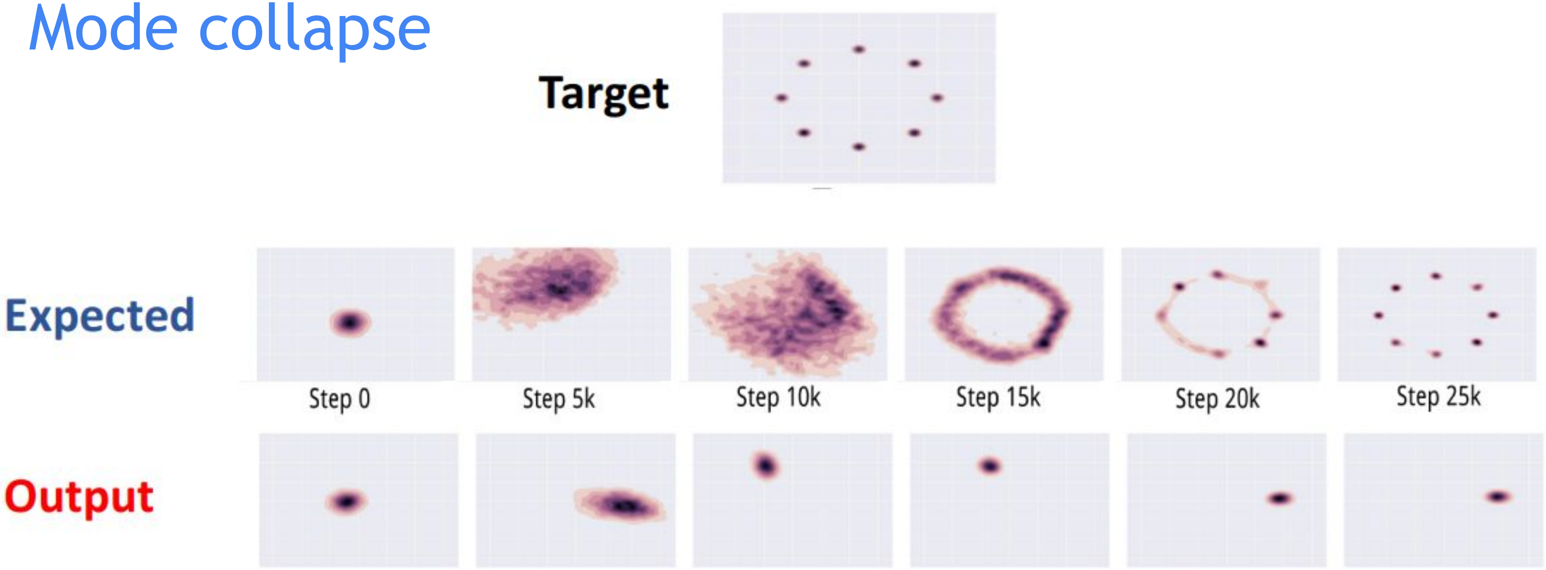




# Taxonomy of Generative models



# Problems with GANs

- non-convergence:
  - it might not converge to the Nash Equilibrium (e.g. minmax xy)
- Mode collapse



# Problems with GANs

- In Vanilla GAN:
  - The divergence measure used is not always continuous
  - The discriminator may not provide sufficient gradients
  - For the optimal discriminator, the gradient is zero for generator
  - Generator collapses too many values to the same sample (mode-collapse)

# Divergences in GANs

Remember:

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

Then,

$$C(G) = \max_D V(G,D)$$

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

In practice, however,

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

# Divergences in GANs

Discriminator measures the divergence between the two distributions

A general class of divergences

$$D_f(P\|Q) = \int_{\mathcal{X}} q(x) f\left(\frac{p(x)}{q(x)}\right) \mathrm{d}x,$$

which is the upper bound for

$$\sup_{T \in \mathcal{T}} \left(\mathbb{E}_{x\sim P}\left[T(x)\right] - \mathbb{E}_{x\sim Q}\left[f^*(T(x))\right]\right)$$

and

$$F(\theta, \omega) = \mathbb{E}_{x\sim P}\left[g_f(V_\omega(x))\right] + \mathbb{E}_{x\sim Q_\theta}\left[-f^*(g_f(V_\omega(x)))\right]$$

# Divergences in GANs

| Name | Output activation $g_f$ | $\mathrm{dom}_{f^*}$ | Conjugate $f^*(t)$ | $f'(1)$ |
|---|---|---|---|---|
| Total variation | $\frac{1}{2}\tanh(v)$ | $-\frac{1}{2} \leq t \leq \frac{1}{2}$ | $t$ | $0$ |
| Kullback-Leibler (KL) | $v$ | $\mathbb{R}$ | $\exp(t-1)$ | $1$ |
| Reverse KL | $-\exp(v)$ | $\mathbb{R}_-$ | $-1-\log(-t)$ | $-1$ |
| Pearson $\chi^2$ | $v$ | $\mathbb{R}$ | $\frac{1}{4}t^2 + t$ | $0$ |
| Neyman $\chi^2$ | $1-\exp(v)$ | $t < 1$ | $2-2\sqrt{1-t}$ | $0$ |
| Squared Hellinger | $1-\exp(v)$ | $t < 1$ | $\frac{t}{1-t}$ | $0$ |
| Jeffrey | $v$ | $\mathbb{R}$ | $W(e^{1-t}) + \frac{1}{W(e^{1-t})} + t - 2$ | $0$ |
| Jensen-Shannon | $\log(2) - \log(1+\exp(-v))$ | $t < \log(2)$ | $-\log(2-\exp(t))$ | $0$ |
| Jensen-Shannon-weighted | $-\pi\log\pi - \log(1+\exp(-v))$ | $t < -\pi\log\pi$ | $(1-\pi)\log\frac{1-\pi}{1-\pi e^{t/\pi}}$ | $0$ |
| GAN | $-\log(1+\exp(-v))$ | $\mathbb{R}_-$ | $-\log(1-\exp(t))$ | $-\log(2)$ |
| $\alpha$-div. ($\alpha < 1, \alpha \neq 0$) | $\frac{1}{1-\alpha} - \log(1+\exp(-v))$ | $t < \frac{1}{1-\alpha}$ | $\frac{1}{\alpha}(t(\alpha-1)+1)^{\frac{\alpha}{\alpha-1}} - \frac{1}{\alpha}$ | $0$ |
| $\alpha$-div. ($\alpha > 1$) | $v$ | $\mathbb{R}$ | $\frac{1}{\alpha}(t(\alpha-1)+1)^{\frac{\alpha}{\alpha-1}} - \frac{1}{\alpha}$ | $0$ |

# Divergences in GANs

To solve vanishing gradient we use other divergences:

We use **Integral Probability Measure (IPM)**:

$$\rho(p, q_\theta) = \sup_{f_w \in \mathcal{F}} \left| \int_{\mathcal{X}} f_w dp - \int_{\mathcal{X}} f_w dq_\theta \right|$$

- Where f is the critic function (a deep neural net)
- Function f is chosen from a class of functions F
- Distribution of real data p
- Distribution of fake data q (it is learnt)

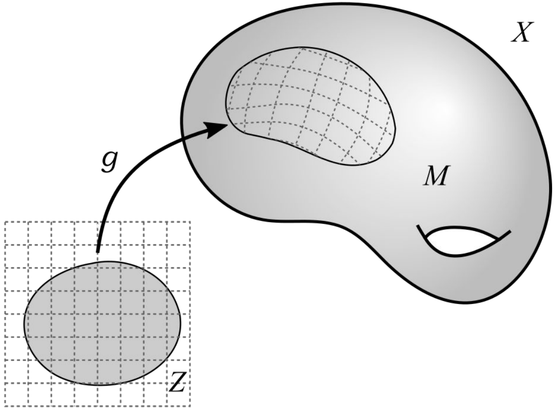
# Dudley GAN

- We propose to use $\|f_{\mathbf{w}}\|_{BL} \leq 1$ where

$$\|f_{\mathbf{w}}\|_{BL} = \|f_{\mathbf{w}}\|_{\infty} + \|f_{\mathbf{w}}\|_{L}$$

$$\|f\|_L := \sup\left\{\frac{|f(\mathbf{x}_1) - f(\mathbf{x}_2)|}{|\mathbf{x}_1 - \mathbf{x}_2|} : \mathbf{x}_1 \neq \mathbf{x}_2 \forall \mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X}\right\}$$

$$\|f_{\mathbf{w}}\|_{\infty} = \sup\{|f_{\mathbf{w}}(\mathbf{x})| : \mathbf{x} \in \mathcal{X}\}$$



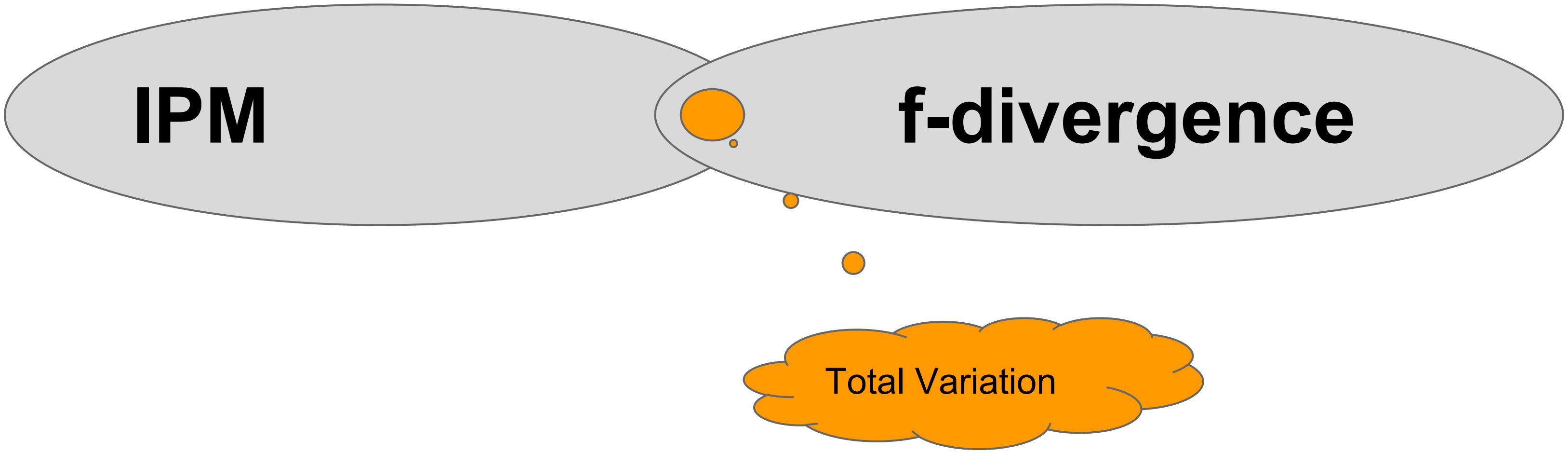

- Bounded: Limits the critic's values
- Lipschitz: Limits the rate of change in the critic's compared to the change in the input
  - How to define Lipschitz neural networks?

# Lipschitz Neural Nets

- We can mathematically prove certain operations are Lipschitz:
  - Convolutional operations with normalized weights
  - Linear operations with normalized weights
  - Certain activation functions: sigmoid, relu, linear, tanh.

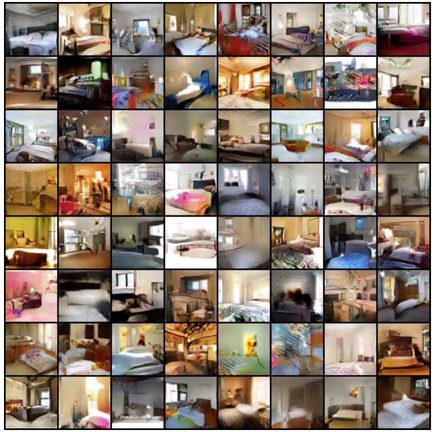
- We can then regularize the function to remain bounded

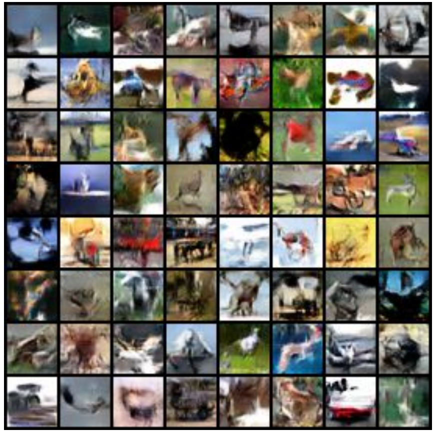
# f-divergence vs IPM



# What can we Do?

Generate Random real looking samples





# What can we Do?

Analyze latent space

# Adversarially Learned Inference

- Learn inference network
- Consider the following joint distribution

$$q(x,z) = q(x)\, q(z|x) \qquad \text{encoder distribution}$$

$$p(x,z) = p(z)\, p(x|z) \qquad \text{generator distribution}$$

# Adversarially Learned Inference

- Learn inference network
- Consider the following joint distribution

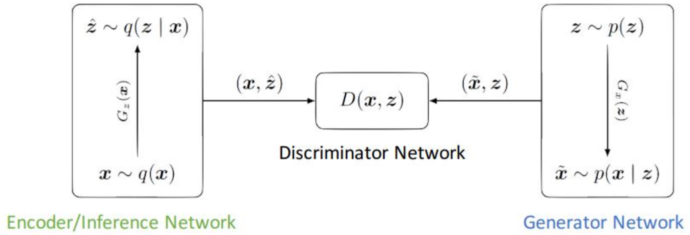



Figure 1 in the original paper.

Nash Equilibrium is

$$\min_G \max_D \; \mathbb{E}_{q(x)}[\log(D(x, G_z(x))] + \mathbb{E}_{p(x)}[\log(1 - D(G_x(z), z))]$$

$p(x,z) \sim q(x,z)$

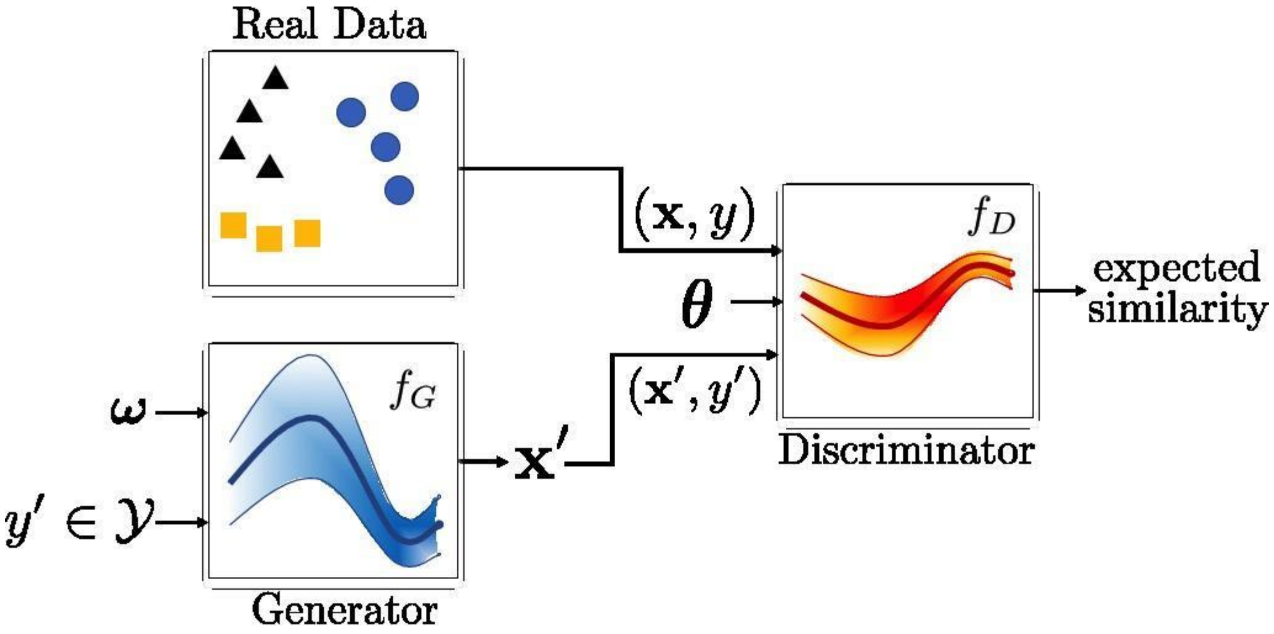
$p(x) \sim q(x)$ and $p(z) \sim q(z)$

$p(x|z) \sim q(x|z)$ and $p(z|x) \sim q(z|x)$

# Uncertainty-aware GAN

- GANs are generated from a function evaluation
- We don’t know if the sample generated
  - Is mapped to a good quality sample
  - Is from the dense region or not
- Let’s treat generator and discriminator as “random functions”

# Uncertainty-aware GAN



# Uncertainty-aware GAN

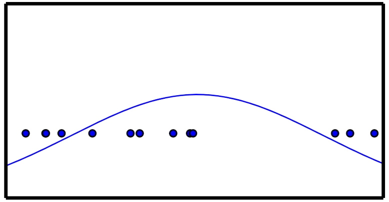
- We perform a Monte Carlo estimate to the expected sample/score
- Training is more stable, due to averaging
- We can compute the uncertainty in the discriminator score

# Outline

- Generative models
- Why using Deep Generative models?
  - Generative vs. Discriminative models
- Existing Methods:
  - Autoregressive Methods
  - Latent Variable Models
    - Variational Autoencoders (VAEs)
    - Generative Adversarial Networks (GANs)
- Problems with existing GANs and our approaches
  - 3D Hand pose Estimation
  - Other divergences (Dudley GANs)
  - Uncertainty in Generation (Uncertainty-aware GAN)
  - **Density Estimator Adversarial Network**
  - Imitation Learning
  - Causality

# Generative Adversarial Density estimator

- GANs are likelihood-free, we don't have a density function
- We can't evaluate probability of a sample in GANs



- Let's consider a likelihood model

$$p(\mathbf{x}|\mathbf{w}) \quad = \quad \exp(\mathbf{w}^\top \phi(\mathbf{x}) - A(\mathbf{w})),$$

$$\boxed{A(\mathbf{w}) = \log \left( \int \exp(\mathbf{w}^\top \phi(\mathbf{x})) d\mathbf{x} \right)}$$

# Generative Adversarial Density estimator

- We can find approximate this normalizer as

$$A_q(\mathbf{w}) \quad = \sup_q \quad \left\{ \int \mathbf{w}^\top \phi(\mathbf{x}) q(\mathbf{x}|\boldsymbol{\theta}) d\mathbf{x} + H_{\mathbf{x}}(q) \right\}$$

- Introducing a generator, we have

$$A_q(\mathbf{w}) \quad = \sup_q \quad \left\{ \int \mathbf{w}^\top \phi(g_{\boldsymbol{\theta}_g}(\mathbf{z})) q(g_{\boldsymbol{\theta}_g}(\mathbf{z})) p_z(\mathbf{z}|\boldsymbol{\theta}_z) d\mathbf{z} + H_{\mathbf{z}}(q) \right\}$$

# Generative Adversarial Density estimator

$$A_q(\mathbf{w}) = \sup_q \left\{ \boxed{\int \mathbf{w}^\top \phi(g_{\boldsymbol{\theta}_g}(\mathbf{z}))q(g_{\boldsymbol{\theta}_g}(\mathbf{z}))p_z(\mathbf{z}|\boldsymbol{\theta}_z)d\mathbf{z}} + H_{\mathbf{z}}(q) \right\}$$

Expectation of
Generated samples

$\updownarrow$

$$\int \mathbf{w}^\top \phi(\mathbf{x})q(\mathbf{x}|\boldsymbol{\theta})d\mathbf{x}$$

Expectation of
Real samples

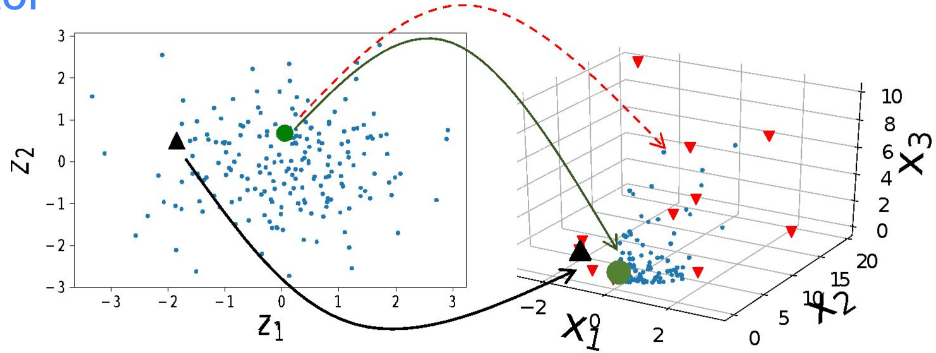
The expectation of the data and the generated samples should match in our density model **(Moment matching property)**

# Generative Adversarial Density estimator

$$A_q(\mathbf{w}) \quad = \sup_q \quad \left\{ \int \mathbf{w}^\top \phi(g_{\boldsymbol{\theta}_g}(\mathbf{z})) q(g_{\boldsymbol{\theta}_g}(\mathbf{z})) p_z(\mathbf{z}|\boldsymbol{\theta}_z) d\mathbf{z} + \boxed{H_{\mathbf{z}}(q)} \right\}$$

Entropy of

- the latent space (e.g. larger variance),
- the curvature of the generator



# Applications

Visual dialog: a sequence of question-answers about an image

**Q**: is the train old?
**A**: i think so
**Sample A**: ~~yes~~ $(0.03 \pm 0.2)$
**Q**: is the train moving?
**A**: no but it does have some steam coming
**Sample A**: no $(0.02 \pm 0.1)$

**Q**: is this in color ?
**A**: yes
**Sample A**: yes $(0.1 \pm 0.05)$
**Q**: how old does the boy look?
**A**: he is not facing me, but maybe
**Sample A**: ~~3~~ $(0.1 \pm 0.4)$

**Q**: what vegetables are there?
**A**: carrots, cauliflower, broccoli
**Sample A**: carrots , carrots , and cucumbers $(0.1 \pm 0.01)$
**Q**: what color is the table ?
**A**: dark brown
**Sample A**: white $(0.1 \pm 0.02)$

# Imitation learning and Inverse Reinforcement Learning

- Reinforcement learning requires a “reward function”

Expert Demonstration → Learning Algorithm → Policy

- An expert drives a car and an autonomous car imitates
- Sampling trajectories to simulate the expert’s behavior

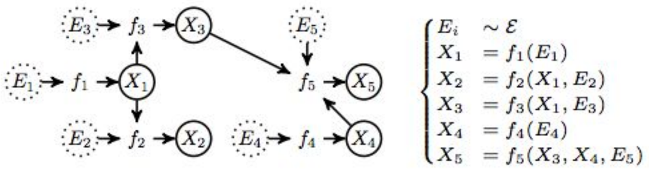
# Imitation learning and Inverse Reinforcement Learning

# Causality

Rather than $p(y|x)$ as the conditional distribution

We consider the causal expression $p(y|\mathrm{do}(x))$

$p(y|\mathrm{do}(x))$ represents the *intervention* expression




Goudet et al. 2018

# Conclusion

- Generative models explain the data, likelihood of samples
- There is a simpler hidden space where complex data may be generated from
- If we create something, we understand it
- Capturing the uncertainty in the model and data
- Deep generative models can be used for
    - Data generation (generate various samples of road)
    - Human imitation (use them instead of real drivers)
    - Environment simulation
    - Use of unlabeled data (rather than expensive, often unavailable real one)