Advanced Learning Models

Chapter III - Deep Generative Models

Julien Mairal & Xavier Alameda-Pineda

with the help of Jakob Verbeek and Thomas Lucas

MSIAM/MoSIG – 2019-2020

Course Organisation

Homework & Data challenge

You can do the homework and the data challenge in groups of two people.

If you do so, the two groups must be with different people.

Table of Contents

Introduction

Motivations for unsupervised (deep) learning

1. **Improve supervised learning from few samples**
   - Unlabeled data often abundantly available
   - Learn representations/features from unlabeled data

2. **Generative models for image and other complex data**
   - Unconditional: sandbox research problem (?)
   - Conditional structured prediction: in-painting, colorization, text-to-image, video forecasting, **etc**.

Image colorization [Royer et al., 2017]

(Un)supervised learning and (un)conditional models

- Supervised learning: model conditional distribution $p_\theta(y|\mathbf{x})$
  - For example: $\mathbf{x}$ an image, y a class label

$$\max_\theta \sum_{(\mathbf{x},y)\sim\mathcal{D}} \ln p_\theta(y|\mathbf{x}) \tag{1}$$

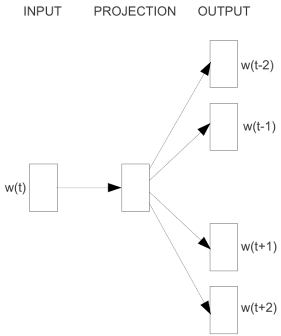
  - $\mathcal{D}$: data generating distribution
  - θ: model parameters

- Unsupervised learning: model unconditional distribution $p(\mathbf{x})$
  - For example: $\mathbf{x}$ an image

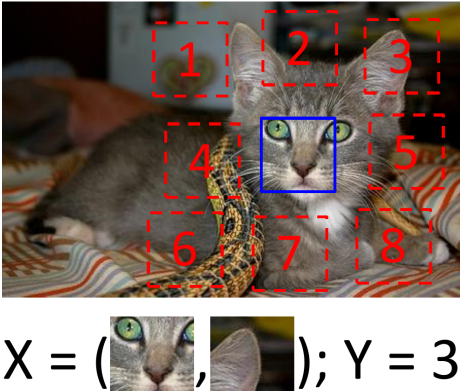
$$\max_\theta \sum_{\mathbf{x}\sim\mathcal{D}} \ln p_\theta(\mathbf{x}) \tag{2}$$

Self-supervised learning

- Learning conditional models $p(y|\mathbf{x})$ from unlabeled data
- Prediction of structural data properties
  - Skip-gram language models (word2vec) [Mikolov et al., 2013]
  - Relative position of image patches [Doersch et al., 2015]
  - Relative ordering of video frames [Fernando et al., 2017]
  - Image inpainting [Pathak et al., 2016]
  - ...



Skip-gram

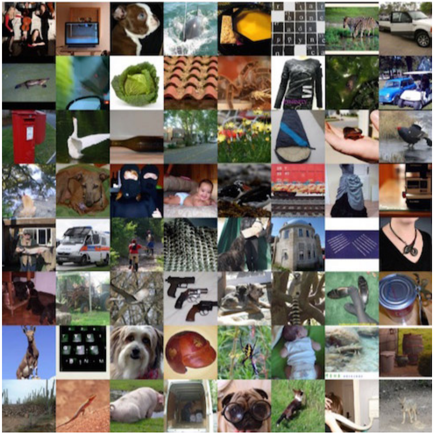


Self-supervised learning to prime supervised learning

- Supervised pre-training of network on proxy-task
- Fine-tune on final task with limited training data
- Unsupervised representation learning
- Does not allow to sample data from model

Generative models

- Unconditional density model $p_\theta(\mathbf{x})$
- Parameters estimated from unlabeled data
- Possible to draw samples from model



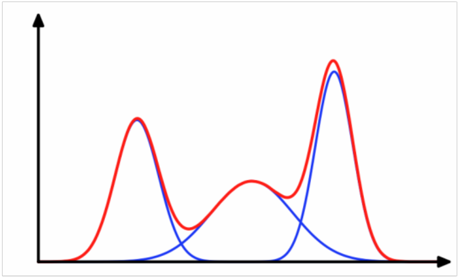

Samples from ImageNet dataset (left) and GAN model (right), figure from OpenAI

My first generative model

- Gaussian mixture model

$$p(z = k) = \pi_k \tag{3}$$
$$p(\mathbf{x}|z = k) = \mathcal{N}(x; \mu_k, \sigma I_D) \tag{4}$$
$$p(\mathbf{x}) = \sum_z p(z)p(\mathbf{x}|z) \tag{5}$$

- Estimation: Expectation-Maximization (EM) algorithm
- Sampling: pick component from prior distribution $p(z)$, then draw sample from conditional distribution $p(\mathbf{x}|z)$



My second generative model

- Probabilistic Principal Component Analysis
  [Roweis, 1997, Tipping and Bishop, 1999]

$$
\begin{aligned}
p(z) &= \mathcal{N}\left(z; 0, I_d\right) && (6)\\
p(\mathbf{x}|z) &= \mathcal{N}\left(\mathbf{x}; \mu + Wz, \sigma I_D\right) && (7)\\
p(\mathbf{x}) &= \int_z p(z)p(\mathbf{x}|z) && (8)
\end{aligned}
$$

- Estimation: SVD or EM algorithm

- Sampling: pick point in subspace from prior $p(z)$,
  then draw sample from conditional distribution $p(\mathbf{x}|z)$

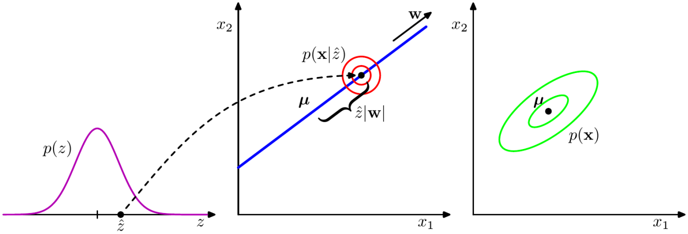


Figure from [Bishop, 2006]

Linear latent variable models

- **Linear transformation** of latent variable
  - PCA: z from unit Gaussian
  - GMM: z random 1-hot vector

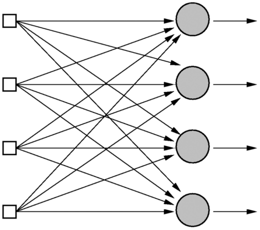
$$\hat{\mathbf{x}} = Wz + \mu \tag{9}$$

- Gaussian noise makes support non-degenerate in data space

$$p(\mathbf{x}|\hat{\mathbf{x}}) = \mathcal{N}\left(\mathbf{x}; \hat{\mathbf{x}}, \sigma I_D\right) \tag{10}$$

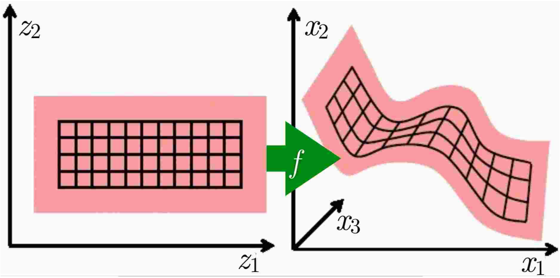
- Negative log-likelihood gives ℓ_2 "reconstruction" loss of PCA and k-means

$$-\ln p(\mathbf{x}|\hat{\mathbf{x}}) = ||\mathbf{x} - \hat{\mathbf{x}}||_2^2 \tag{11}$$



Non-linear latent variable models

- Simple distribution $p(\mathbf{z})$ on latent variable $\mathbf{z}$, **e.g**. standard Gaussian
- Non-linear function $\mathbf{x} = f_\theta(\mathbf{z})$ maps latent variable to data space, for example deep neural net
- Induces complex marginal distribution $p_\theta(\mathbf{x})$



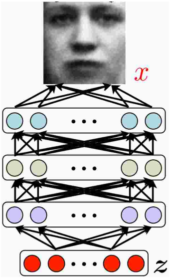


Figure from Aaron Courville

Learning deep latent variable models

- Marginal distribution on $\mathbf{x}$ obtained by integrating out $\mathbf{z}$

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; 0, I), \tag{12}$$

$$p_\theta(\mathbf{x}) = \int_z p(\mathbf{z}) p(\mathbf{x}|f_\theta(\mathbf{z})). \tag{13}$$

- Evaluation of $p_\theta(\mathbf{x})$ intractable due to integral involving non-linear deep net $f_\theta(\cdot)$

- Several approaches to learn deep latent variable models
  1. Avoid integral: Generative adversarial networks (GAN)
  2. Approximate integral: Variational autoencoders (VAE)
  3. Constrain f_θ so that we can compute $p_\theta(x)$ (e.g. Real-NVP)
  4. Do not use latent variables (e.g. PixelCNN)

Generative Adversarial Networks

Generative adversarial networks [Goodfellow et al., 2014]

- Sample $p(\mathbf{z})$, map it using deep net to $\mathbf{x} = G_\theta(\mathbf{z})$
- Instead of evaluating $p^*(\mathbf{x})$, use classifier D_ϕ
  - $D_\phi(\mathbf{x}) \in [0, 1]$ probability $\mathbf{x}$ is real **vs**. synth. image

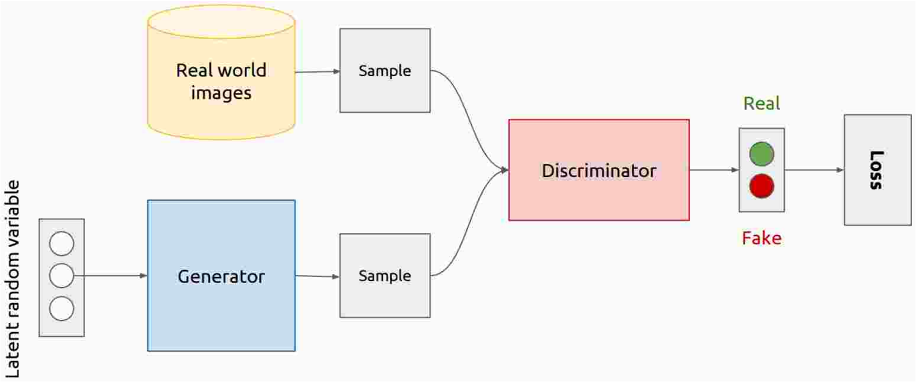


Figure from Kevin McGuinness

Discriminator architecture for images

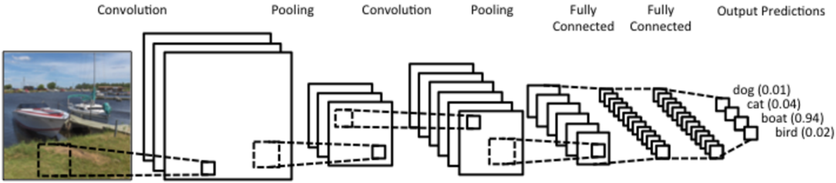


Figure from Kevin McGuinness

- Recognition CNN model, with sigmoid output layer
- Binary classification output: real / synthetic

Generator architecture for images

- Unit Gaussian prior on $\mathbf{z}$, typically 10^2 to 10^3 dimensions
- Up-convolutional deep network (reverse recognition CNN)
  - Replace pooling layers that reduce resolution with upsampling layers (nearest neighbor, bi-linear, or learned)
  - Low-resolution layers induce long-range correlations
  - High-resolution layers induce short-range correlations

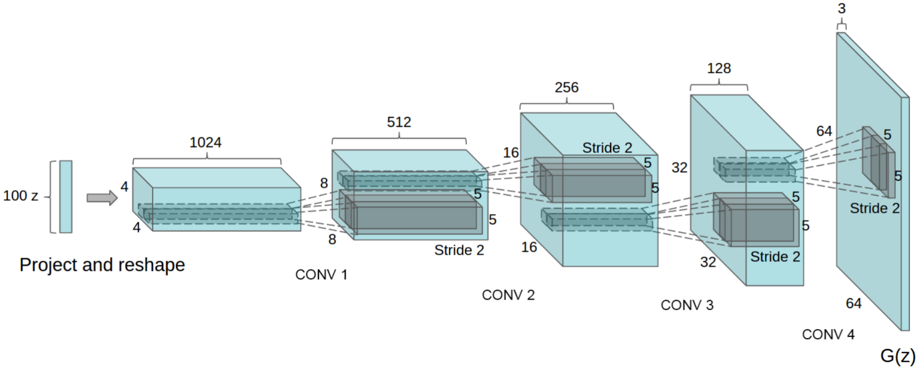
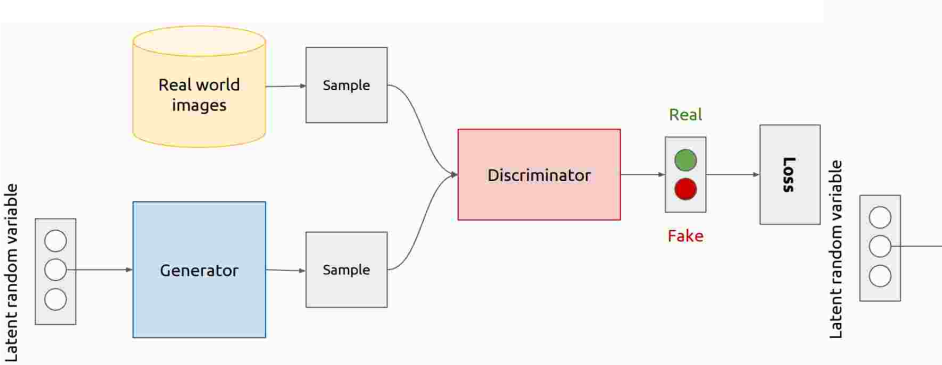


Figure from OpenAI

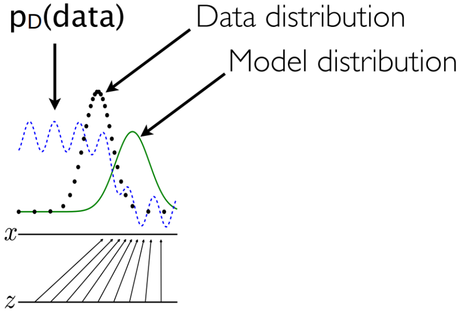
Training GANs



- Discriminator: maximize classification for a given generator
- Generator: degrade classification of a given discriminator

- Samples **z** pass through two differentiable modules
- Discriminator acts as **trainable loss function**

GAN learning process



Poorly fit model

GAN Optimization problem

- Objective function $V(\phi, \theta)$: performance of discriminator

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

$$\min_\theta \max_\phi V(\phi, \theta)$$

- Assuming infinite data and model capacity,
  and reaching optimal discriminator at each iteration
  1. Unique global optimum for G at data distribution
  2. Convergence to optimum guaranteed

Optimal discriminator

- For fixed generator G, the optimal discriminator D is the Bayes classifier

$$D^*_G(\mathbf{x}) = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_G(\mathbf{x})} \tag{14}$$

- Proof: Given generator f, the optimal discriminator maximizes

$$\begin{aligned} V(D,G) &= \mathbb{E}_{x\sim p_{\mathrm{data}}(x)}[\ln D(x)] + \mathbb{E}_{z\sim p(z)}[\ln(1-D(G(z)))] \\ &= \int_x p_{\mathrm{data}}(x)\ln D(x) + p_G(x)\ln(1-D(x))\,dx \end{aligned}$$

  For any $(a,b) \in \mathbb{R}^2 \setminus \{0,0\}$ the function $a\ln(y) + b\ln(1-y)$ achieves its maximum in $[0,1]$ at $y = a/(a+b)$.
  Discriminator only needs to be defined in support of training data and $p_G(x)$.

Link with Jensen-Shannon divergence

- Plugging in the optimal discriminator we obtain

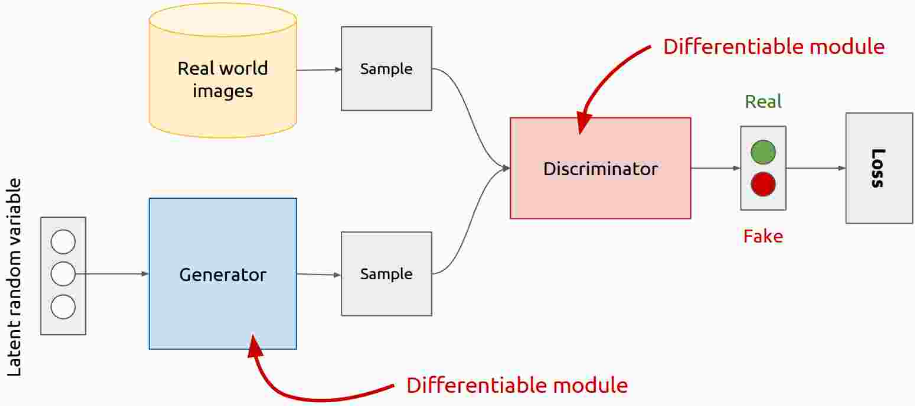
$$\max_D V(D, G) = -\ln 4 + 2D_{JS}(p_{\mathrm{data}}||p_G)$$

with Jensen-Shannon divergence

$$D_{JS}(p||q) = \frac{1}{2}D_{KL}\left(p\middle|\middle|\frac{p+q}{2}\right) + \frac{1}{2}D_{KL}\left(q\middle|\middle|\frac{p+q}{2}\right)$$

- Unique global minimum obtained for $p_{\mathrm{data}} = p_G$
- If D is set to optimum at each iteration, then convexity shows that gradient descent on p_G recovers the global optimum

Training GANs in practice



$$V(\phi, \theta) = \mathbb{E}_{x \sim p_{\mathrm{data}}(x)}[\ln D_\phi(x)] + \mathbb{E}_{z \sim p(z)}[\ln(1 - D_\phi(f_\theta(z)))]$$

- Replace expectations with sample average in mini-batch
- Parallel stochastic gradient descent on ϕ and θ

Samples model learned on face images [Radford et al., 2016]

Modern network on ImageNet, class conditional

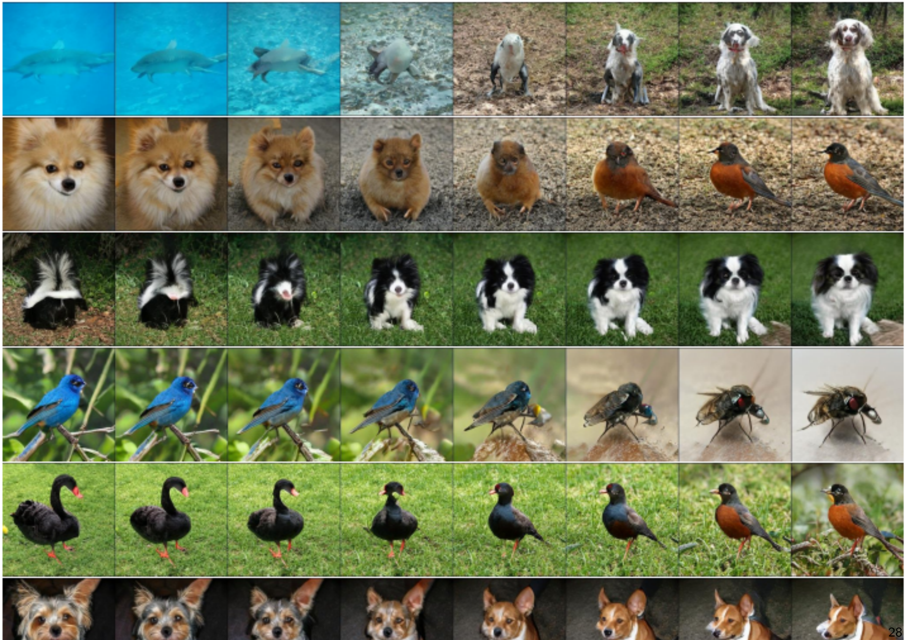
Examples taken from Brock et al. 2019

GAN generalizes beyond training data

- Sample along linear trajectory in latent space $z_1 \to z_2$
- Smooth transitions suggest generalization, sharp transitions would suggest literal memorization

Examples taken from [Radford et al., 2016], trained on LSUN bedroom dataset

GAN generalizes beyond training data



Issues training GANs in practice

- GANs known to be difficult to train in practice
  - Formulated as mini-max objective between two networks
  - Optimization can oscillate between solutions
  - Picking "compatible" generator and discriminator architectures
  - Training fails if the discriminator is 'too good'
- Mode collapse: failure to capture part of training data
- Quantitative evaluation not aligned with objective function

Why is GAN training is difficult in practice?

- Recall divergence measures between distributions
- Kullback-Leibler divergence: maximum likelihood training
  - Infinite if q (model) has a zero in the support of p (data)

$$D_{KL}(p||q) = \int_x p(x) \big[\ln q(x) - \ln p(x) \big] \tag{15}$$

- Jensen-Shannon divergence: idealized GAN training
  - Symmetric KL to mixture of p and q

$$D_{JS}(p||q) = \frac{1}{2} D_{KL}\left(p \middle\| \frac{p+q}{2}\right) + \frac{1}{2} D_{KL}\left(q \middle\| \frac{p+q}{2}\right) \tag{16}$$

Why is GAN training is difficult in practice? [Arjovsky et al., 2017]

1. Strong discriminator leads to vanishing gradients of $\mathbb{E}_{p_z}[\ln(1 - D(G(z)))]$ w.r.t. generator
   - Happens early in training with poor generator
   - Tuning of capacity and training regime of discriminator
   - Generator no longer minimize JS divergence

2. Minimizing $-\mathbb{E}_{p_z}[\ln(D(G(z)))]$ instead to boost gradient
   - Optimizes $KL(p_G || p_{\mathrm{data}}) - 2JS(p_G || p_{\mathrm{data}})$
   - Wrong sign in the JS divergence
   - Direction of KL term leads to mode dropping

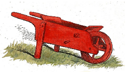
Wasserstein or "earth-mover" distance

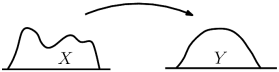
- Consider joint distribution $\gamma(x, y)$
  with marginals $p(x) = \gamma(x)$ and $q(y) = \gamma(y)$

- Conditional $\gamma(y|x)$ "moves mass" to transform $p(\cdot)$ into $q(\cdot)$

- Cost associated with a given transformation

$$T(\gamma) = \int_{x,y} \gamma(x, y)\; ||x - y|| = \int_x p(x) \int_y \gamma(y|x)\; ||x - y||$$



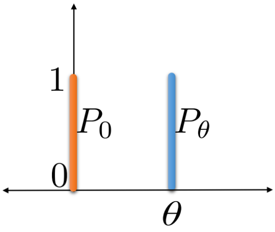


- Wasserstein distance is the cost of optimal transformation

$$D_{WS}(p||q) = \inf_{\gamma \in \Gamma(p,q)} T(\gamma) \tag{17}$$

Distributions with low dimensional support

- Simple example: support on lines in $\mathbb{R}^2$
  - p_0 uniform on $x_2 \in [0, 1]$ for $x_1 = 0$
  - p_θ uniform on $x_2 \in [0, 1]$ for $x_1 = \theta$
- All measures zero for $\theta = 0$, but for $\theta \neq 0$
  - $D_{KL}(p_0 || p_\theta) = \infty$
  - $D_{JS}(p_0 || p_\theta) = \ln 2$
  - $D_{WS}(p_0 || p_\theta) = |\theta|$



- Wasserstein based on proximity of support
- JS and KL based on overlap of support
  - In general measure zero overlap with low dim. supports
  - GAN has support with dimension of latent variable **z**

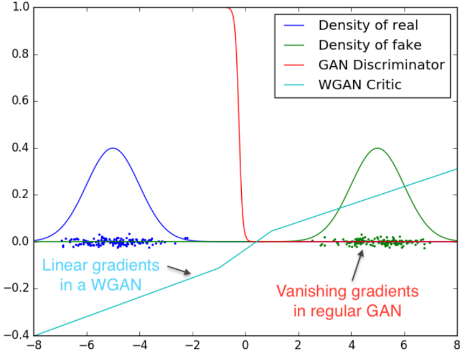
Wasserstein GAN

- Dual formulation of Wasserstein distance

$$D_{WS}(p_{\mathrm{data}}||p_G) = \frac{1}{k} \max_{||D||_L \leq k} \mathbb{E}_{p_{\mathrm{data}}}[D(\mathbf{x})] - \mathbb{E}_{p_z}[D(G(\mathbf{z}))]$$

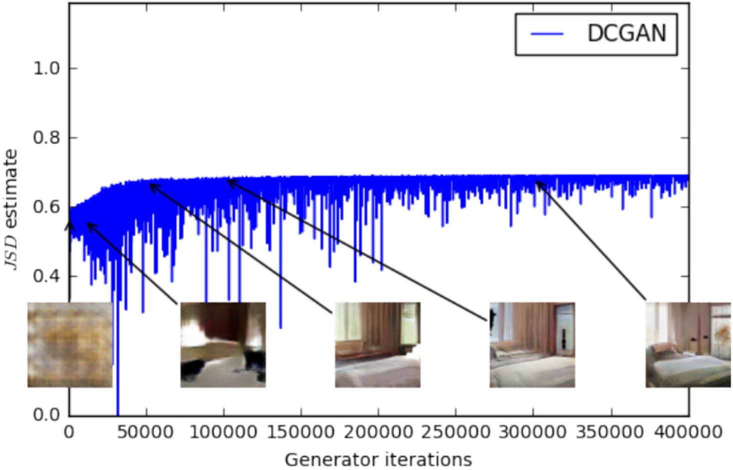
1. Restrict D to some deep net architecture
2. Enforce Lipschitz constraint by clipping discriminator weights or penalty on gradient magnitude [Gulrajani et al., 2017]

- Removes log-sigmoid transformation w.r.t. normal GAN



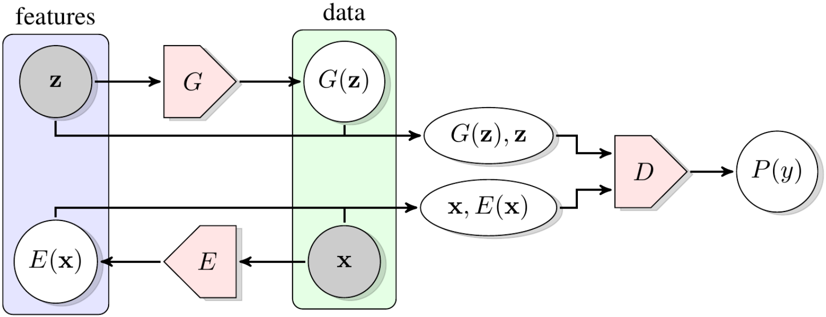
Experimental comparison GAN and WGAN

- GAN loss unstable, and actually increases over iterations!
- WGAN loss deceases in stable manner
- WGAN gives better correlation loss and sample quality



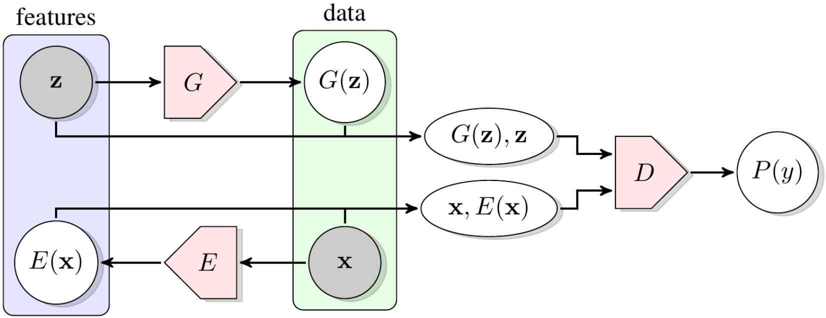
Latent variable inference in GANs [Donahue et al., 2017]

- Vanilla GAN lacks a mechanism to infer **z** from **x**



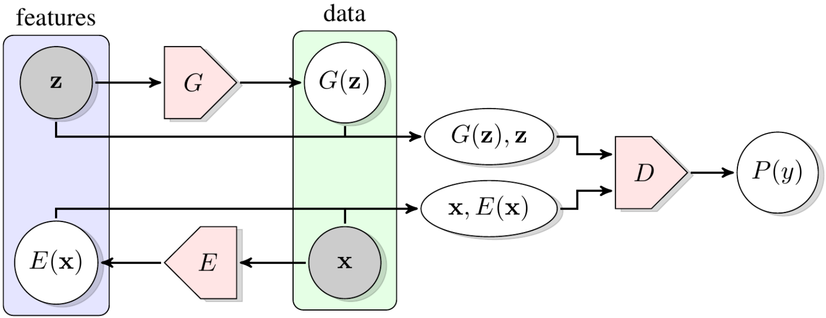
- Generator: maps latent variable **z** to data point **x**
- Encoder: infers latent representation **z** from data point **x**

Induced joint distributions over $(\mathbf{x}, \mathbf{z})$



- Generator: $p_G(\mathbf{x}, \mathbf{z}) = p_{\mathbf{z}}(\mathbf{z})\, \delta(\mathbf{x} - G(\mathbf{z}))$
- Encoder: $p_E(\mathbf{x}, \mathbf{z}) = p_{\text{data}}(\mathbf{x})\, \delta(\mathbf{z} - E(\mathbf{x}))$
- Discriminator: pair $(\mathbf{x}, \mathbf{z})$ completed by generator or encoder?

Bidirectional GANs [Donahue et al., 2017]



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

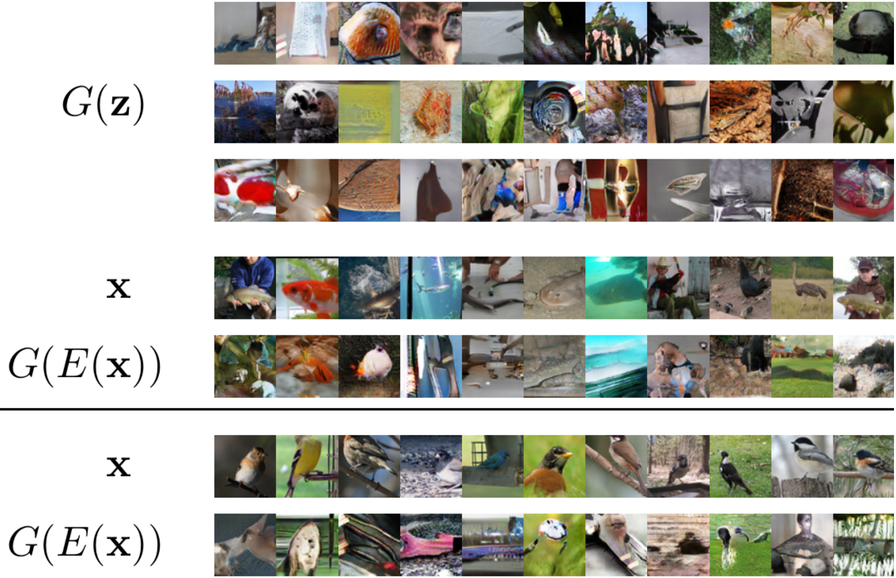
$$\min_{G,E} \max_{D} V(D, E, G)$$

- For optimal discriminator objective equals JS divergence

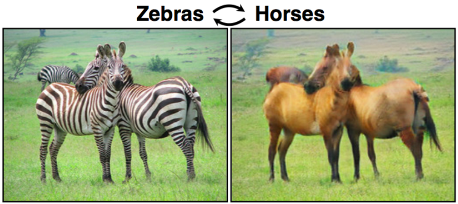
$$\max_{D} V(D, E, G) = 2D_{JS}\left(p_E(\mathbf{x}, \mathbf{z}) || p_G(\mathbf{x}, \mathbf{z})\right) - \ln 4$$

- At optimum G and E are each others inverse

BiGAN samples, ImageNet 64 $\times$ 64

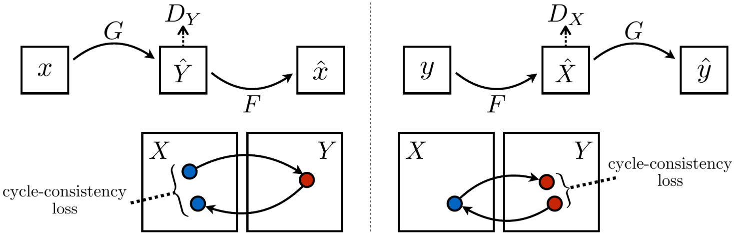


Unpaired image-to-image translation [Zhu et al., 2017]



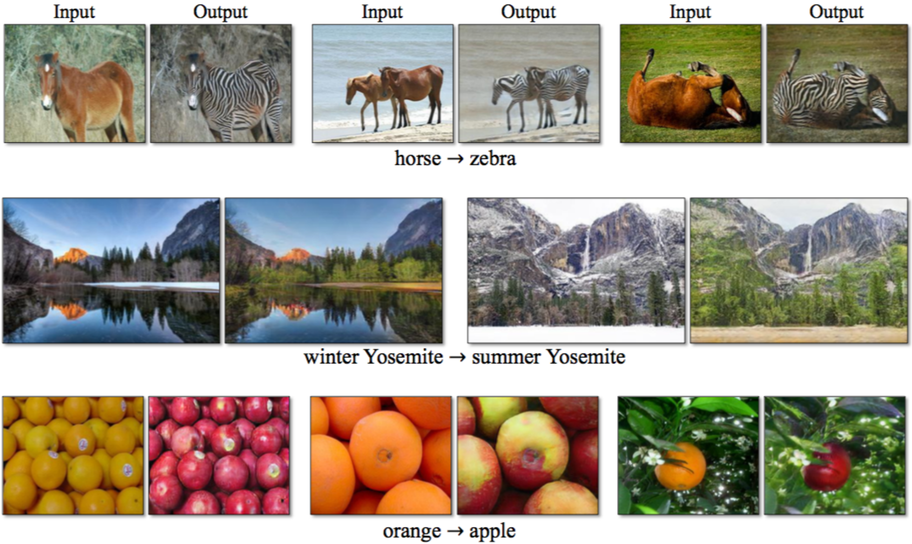
- Learn 2-way mapping between different image domains
- Without using supervised aligned training samples

1. Discriminator ensures realistic samples in each domain
2. Cycle-consistency loss ensures alignment



Some successful examples

- Without using any supervised/aligned examples!



horse → zebra

winter Yosemite → summer Yosemite

orange → apple

And a failure case

Conditional image generation

- We may want to condition the generation by a certain input vector.
- Example: Action Unit conditioned face generation.

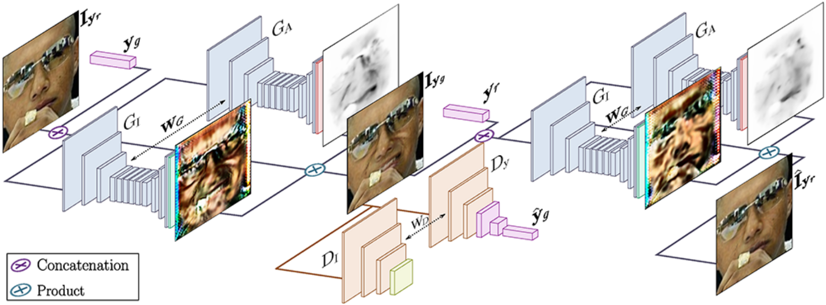
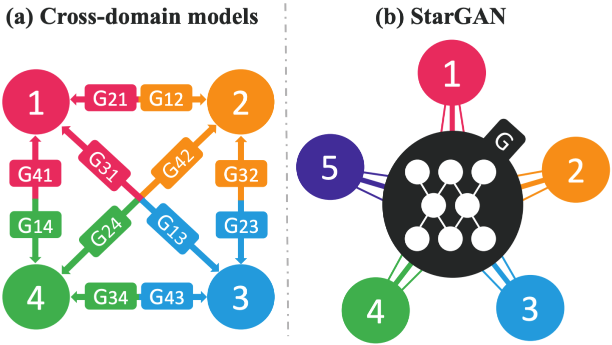


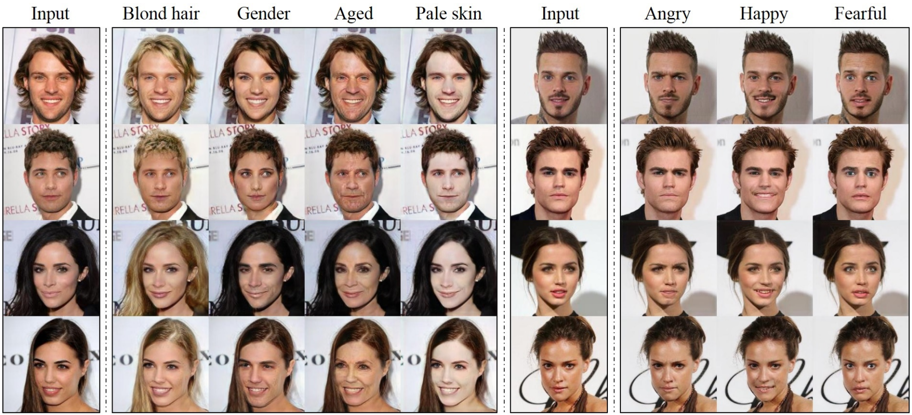
Image from [Pumarola et al., 2018].

Multi-domain conditional generation

- We may want to translate between multiple domains.
- With previous methods, we need to learn a pair encoder-generation for every two domains $\rightarrow$ highly undesirable.
- StarGAN: use a central latent representation space.
- Learn a domain2central encoder and a central2domain generator for each domain.



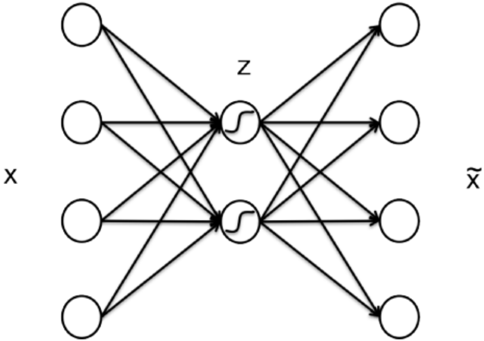
Samples


Input
Blond hair
Gender
Aged
Pale skin
Input
Angry
Happy
Fearful

Variational Auto-Encoders

Autoencoders

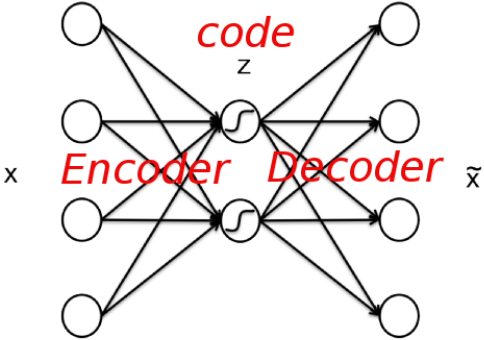
- Learn latent representation $\mathbf{z}$ via reconstruction of data $\mathbf{x}$
- Neural network where output $\sim$ input
  - Encoder: maps data $\mathbf{x}$ to latent code $\mathbf{z}$
  - Decoder: maps latent code $\mathbf{z}$ to reconstruction $\tilde{\mathbf{x}}$
- Loss minimizes discrepancy between $\mathbf{x}$ and $\tilde{\mathbf{x}}$



Relation autoencoders and PCA [Baldi and Hornik, 1989]

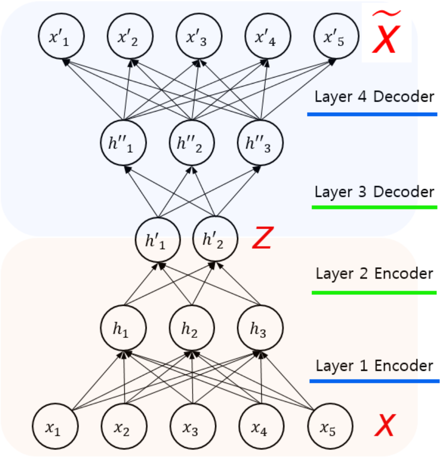
- Autoencoder recovers PCA if
  1. Encoder and decoder are both linear
  2. Optimizing ℓ_2 reconstruction loss

$$\min_{V,W} \frac{1}{2N} \sum_{n=1}^{N} ||x_n - VWx_n||^2 \tag{18}$$



Deep non-linear autoencoders

- Stack many non-linear layers in encoder and decoder
- Non-linear representation learning
- Does not provide a generative model that can be sampled



Autoencoding variational Bayes [Kingma and Welling, 2014]

- Decoder f implements generative latent variable model
  - Maps latent code $\mathbf{z}$ to observation $\mathbf{x}$

$$p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; f_\theta^\mu(\mathbf{z}), f_\theta^\sigma(\mathbf{z})) \tag{19}$$

- Encoder g compute approximate posterior distribution
  - Maps data $\mathbf{x}$ to latent code $\mathbf{z}$

$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; g_\phi^\mu(\mathbf{x}), g_\phi^\sigma(\mathbf{x})) \tag{20}$$

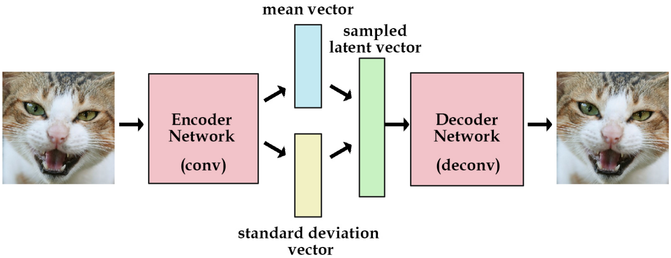


Figure from kvfrans@github

Objective function: Evidence lower bound (ELBO)

- Quantity of interest: marginal likelihood or "evidence"

$$p_\theta(\mathbf{x}) = \int_{\mathbf{z}} p(\mathbf{z}) p_\theta(\mathbf{x}|\mathbf{z}) \tag{21}$$

- Idea 0: Monte-Carlo estimation. Problem: high dimensional
- Idea 1: Weighted sampling

$$p_\theta(\mathbf{x}) = \int_{\mathbf{z}} q_\phi(z|x) p_\theta(\mathbf{x}|\mathbf{z}) \frac{p(\mathbf{z})}{q_\phi(z|x)} dz \tag{22}$$

Objective function: Evidence lower bound (ELBO)

- Idea 2: Efficient estimation with the ELBO

$$\ln(p_\theta(\mathbf{x})) = \ln\left(\int_{\mathbf{z}} q_\phi(z|x)p_\theta(\mathbf{x}|\mathbf{z})\frac{p(\mathbf{z})}{q_\phi(z|x)}dz\right) \quad (23)$$

$$\geq \int_{\mathbf{z}} q_\phi(z|x)\ln\left(p_\theta(x|z)\frac{p(\mathbf{z})}{q_\phi(z|x)}\right)d\mathbf{z} \quad (24)$$

$$= \mathbb{E}_{q_\phi(z|x)}[\ln(p_\theta(x|z))] - D_{KL}(q_\phi(z|x)||p(z)) \quad (25)$$

- ELBO becomes function of inference net and generative net

$$F(\theta, \phi) = \mathbb{E}_{q_\phi}[\ln p_\theta(\mathbf{x}|\mathbf{z})] - D_{KL}\big(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})\big) \quad (26)$$

Objective function: Evidence lower bound (ELBO)

Comments on the ELBO:

$$F(\theta, \phi) = \underbrace{\mathbb{E}_{q_\phi(z|x)}[\ln p_\theta(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction}} - \underbrace{D_{KL}\big(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})\big)}_{\text{Regularization}} \tag{27}$$

- Has an auto-encoder interpretation.
- Efficient computations, at the cost of approximation.
- KL divergence: non-negative, and zero if and only if $q = p$. Balance between both terms

Objective function: Evidence lower bound (ELBO)

Re-writing the ELBO:

$$F(\theta,\phi) = \mathbb{E}_{q_\phi}[\ln p_\theta(\mathbf{x}|\mathbf{z})] - D_{KL}\big(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})\big) \tag{28}$$

Using Bayes rule yields $p(z) = \frac{p(x)p(z|x)}{p(x|z)}$ and:

$$\begin{aligned} F(\theta,\phi) &= \mathbb{E}_{q_\phi}[\ln p_\theta(\mathbf{x}|\mathbf{z})] - \int_z q_\phi(\mathbf{z}|\mathbf{x}) \log\left(\frac{q_\phi(\mathbf{z}|\mathbf{x})p_\theta(\mathbf{x}|\mathbf{z})}{p(\mathbf{z}|\mathbf{x})p_\theta(\mathbf{x})}\right) && (29) \\ &= \mathbb{E}_{q_\phi}\left[\frac{\ln p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{x})}{\ln p_\theta(\mathbf{x}|\mathbf{z})}\right] - D_{KL}\big(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})\big) && (30) \\ &= \ln p_\theta(\mathbf{x}) - D_{KL}\big(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})\big) && (31) \end{aligned}$$

Objective function: Evidence lower bound (ELBO)

$$
\begin{aligned}
F(\theta, \phi) &= \mathbb{E}_{q_\phi}[\ln p_\theta(\mathbf{x}|\mathbf{z})] - D_{KL}\big(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})\big) && (32)\\
&= \ln p_\theta(\mathbf{x}) - D_{KL}\big(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x})\big) && (33)
\end{aligned}
$$

Comments:

- Second form intractable
- Second form clearly a lower bound
- Second bound is tight if and only if $q_\phi(\mathbf{z}|\mathbf{x}) = p_\theta(\mathbf{z}|\mathbf{x})$

Computation ELBO for variational autoencoder

$$F(\theta, \phi) = \underbrace{\mathbb{E}_{q_\phi}\left[\ln p_\theta(\mathbf{x}|\mathbf{z})\right]}_{\text{Reconstruction}} - \underbrace{D_{KL}\left(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})\right)}_{\textbf{Regularization}} \tag{34}$$

- **Regularization term** keeps q from collapsing to single point $\mathbf{z}$ (Information bottleneck)
- Closed form if both terms are Gaussian, for $p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; 0, I)$

$$D_{KL}\left(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})\right) = \frac{1}{2}\left[1 + \ln g_\phi^\sigma(\mathbf{x}) - g_\phi^\mu(\mathbf{x}) - g_\phi^\sigma(\mathbf{x})\right] \tag{35}$$

- Differentiable function of inference net parameters

Computation ELBO for variational autoencoder

$$F(\theta, \phi) = \underbrace{\mathbb{E}_{q_\phi}[\ln p_\theta(\mathbf{x}|\mathbf{z})]}_{\textbf{Reconstruction}} - \underbrace{D_{KL}\left(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})\right)}_{\text{Regularization}} \tag{36}$$

- **Reconstruction term**: to what extent can $\mathbf{x}$ be reconstructed from $\mathbf{z}$ following approximate posterior $q(\mathbf{z}|\mathbf{x})$
- Use sample approximation of intractable expectation $\mathbf{z_s} \sim q_\phi(\mathbf{z}|\mathbf{x})$

$$\mathbb{E}_{q_\phi}[\ln p_\theta(\mathbf{x}|\mathbf{z})] \approx \frac{1}{S}\sum_{s=1}^{S} \ln p_\theta(\mathbf{x}|\mathbf{z_s}) \tag{37}$$

- Estimator is non-differentiable due to sampling operator

Re-parametrization trick

- Side-step non-differentiable sampling operator by re-parametrizing samples $\mathbf{z_s} \sim q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}\left(\mathbf{z}; g_\phi^\mu(\mathbf{x}), g_\phi^\sigma(\mathbf{x})\right)$
- Use inference net to modulate samples from a unit Gaussian

$$\mathbf{z}_s = g_\phi^\mu(\mathbf{x}) + g_\phi^\sigma(\mathbf{x}) \odot \epsilon_s, \qquad \epsilon_s \sim \mathcal{N}(\epsilon_s; 0, I) \tag{38}$$

- Samples $\mathbf{z}_s$ differentiable function of inference net param. ϕ, given unit Gaussian samples ϵ_s
- Unbiased differentiable approximation of ELBO

$$F(\theta, \phi) \approx \frac{1}{S} \sum_{s=1}^{S} \ln p_\theta\left(\mathbf{x}|g_\phi^\mu(\mathbf{x}) + g_\phi^\sigma(\mathbf{x}) \odot \epsilon_s\right) \tag{39}$$

$$-\frac{1}{2}\left[1 + \ln g_\phi^\sigma(\mathbf{x}) - g_\phi^\mu(\mathbf{x}) - g_\phi^\sigma(\mathbf{x})\right] \tag{40}$$

Re-parametrization trick in a cartoon

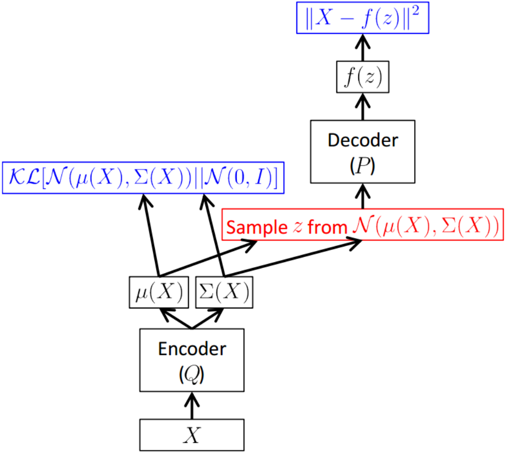


Figure from [Doersch, 2016]

Autoencoding variational Bayes training algorithm

- For each data point $\mathbf{x}$ in a mini-batch
  1. Sample one or multiple values $\{\epsilon_s\}$
  2. Use back-propagation to compute
     $g_\theta = \nabla_\theta F(\theta, \phi, \{\epsilon_s\})$
     $g_\phi = \nabla_\phi F(\theta, \phi, \{\epsilon_s\})$
  3. Gradient-based parameter update

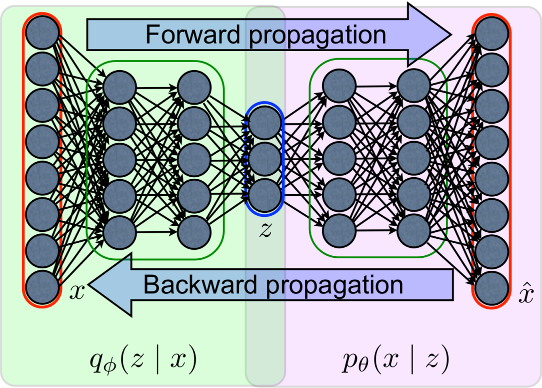


Figure from Aaron Courville

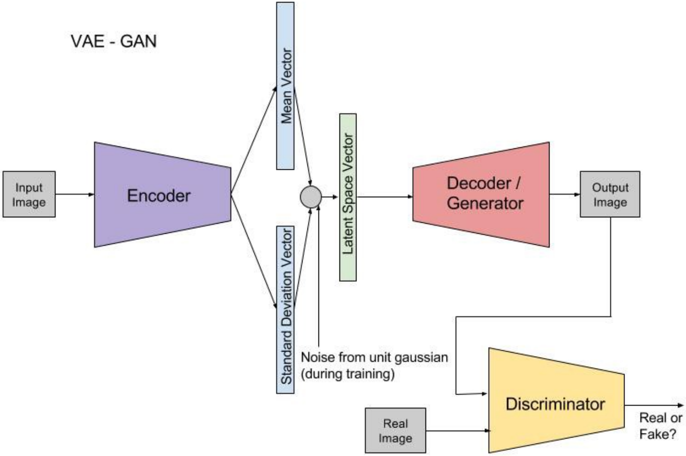
Random samples from VAE and GAN

- Trained from 200k images in CelebA dataset
- VAE samples appear overly smooth / blurred
- GAN samples show more (imperfect) detail

Figure from [Hou et al., 2016]

VAE vs. GAN

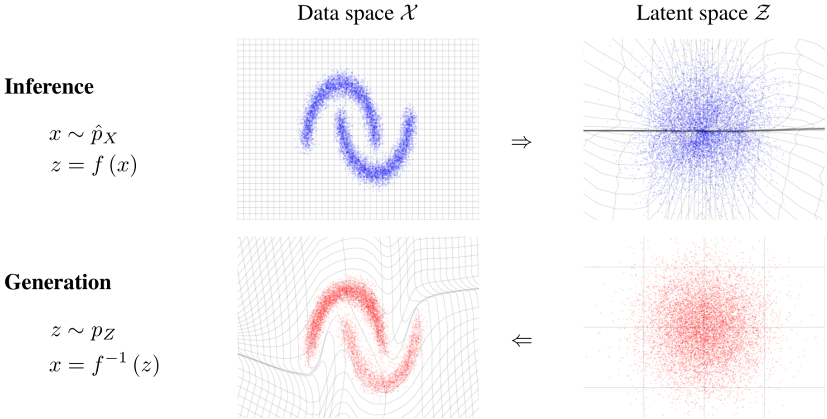
- VAE provide a nice probabilistic generation framework but smooth results.
- GANs are less intuitive but have sharper results.



Deep invertible transformations

Non-volume preserving (NVP) transformation [Dinh et al., 2017]

- Learn invertible function from latent to data space
- Latent and data space have same dimensionality
- Unit Gaussian prior on latent variables
- Tractable sampling and exact inference



Change of variable formula for invertible function

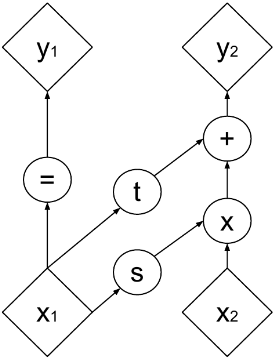
- Using the change of variable formula:

$$p_X(\mathbf{x}) = p_Y(f(\mathbf{x})) \times \left| \det \left(\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}^\top} \right) \right|$$

- Need to ensure efficient computation of (i) $\mathbf{y} = f(\mathbf{x})$ and (ii) determinant

1. Partition variables in two groups
2. Keep one group unchanged
3. Let one group transform the other via translation and scaling

$$\begin{aligned} \mathbf{y}_1 &= \mathbf{x}_1 \\ \mathbf{y}_2 &= t(\mathbf{x}_1) + \mathbf{x}_2 \odot \exp(s(\mathbf{x}_1)) \end{aligned}$$



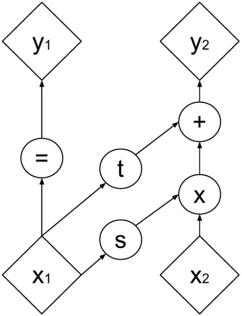
Properties: Efficient inversion

- Inverse transformation

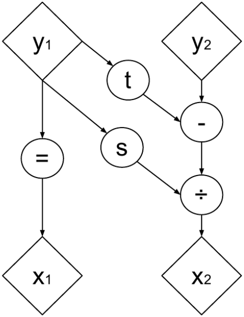
$$\mathbf{x}_1 = \mathbf{y}_1 \tag{41}$$

$$\mathbf{x}_2 = (\mathbf{y}_2 - t(\mathbf{x}_1)) \odot \exp(-s(\mathbf{x}_1)) \tag{42}$$

- No need to invert $s(\cdot)$ and $t(\cdot)$
- Can use complex non-invertible functions, **e.g**. deep CNN



(a) Forward propagation



(b) Inverse propagation

Properties: Efficient determinant computation

- Triangular structure of Jacobian

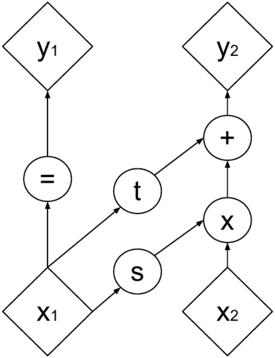
$$\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}^\top} = \begin{bmatrix} I_d & 0 \\ \frac{\partial \mathbf{y}_2}{\partial \mathbf{x}_1^\top} & \mathrm{diag}(\exp(s(\mathbf{x}_1))) \end{bmatrix}$$

- Determinant given by product of Jacobian's diagonal terms

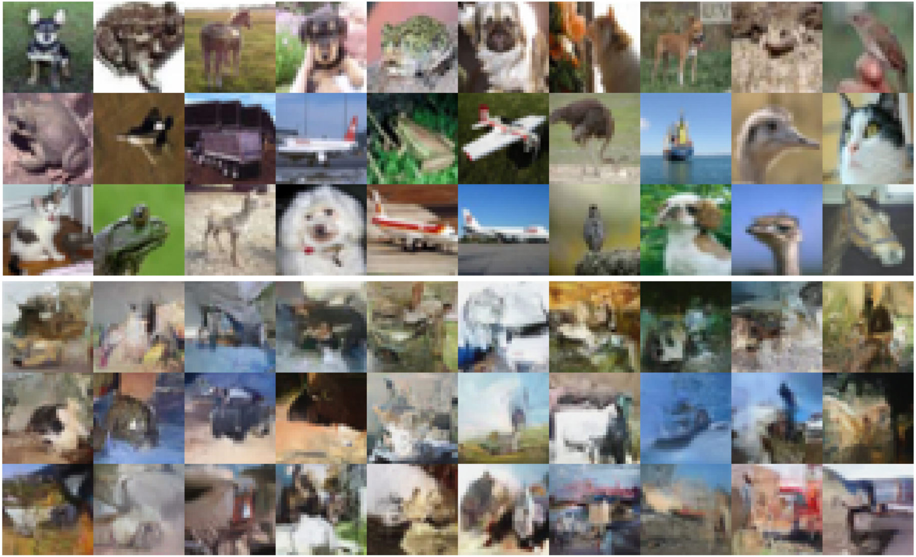
$$\ln \det \left(\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}^\top} \right) = \mathbf{1}^\top s(\mathbf{x}_1)$$

- Log-likelihood easily computed, optimize using stochastic gradient decent

$$\ln p_X(\mathbf{x}) = \ln p_Y(f(\mathbf{x})) + \mathbf{1}^\top s(\mathbf{x}_1)$$



Images & Samples NVP: CIFAR10 Dataset 32 $\times$ 32



Autoregressive Density Estimation

Autoregressive modeling

- Consider generic factorization of joint probability

$$p(\mathbf{x}_{1:D}) = p(x_1) \prod_{i=2}^{D} p(x_i | \mathbf{x}_{<i}) \tag{43}$$

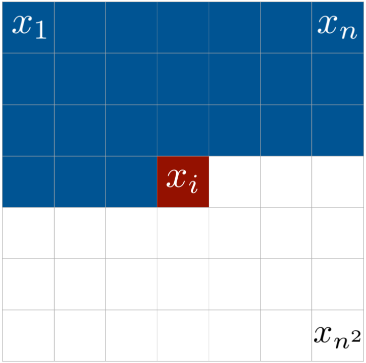
  with $\mathbf{x}_{<i} = \mathbf{x}_1, \ldots, \mathbf{x}_{i-1}$

- Use (deep) neural net to model dependencies in $p(x_i | \mathbf{x}_{<i})$
- Tractable exact likelihood computations
  - No complex integral over latent variables in likelihood
- Slow sequential sampling process
  - Cannot rely on latent variables to couple pixels

Pixel Recurrent/Convolutional Neural Networks

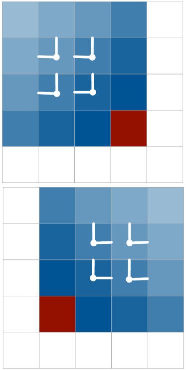
[Oord et al., 2016b]

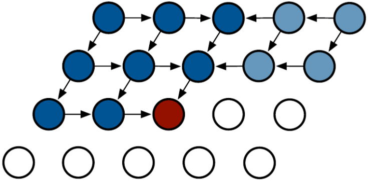
- Predict pixels one-by-one in row-major ordering
- Translation invariant definition of conditionals $p(x_i|\mathbf{x}_{<i})$
- Decouple number of pixels from number of parameters



Pixel RNN: Bi-directional LSTM

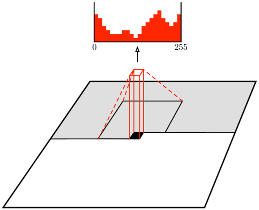
- Two sets of LSTM units, working down-right and down-left
  - Input up and left/right state
  - Input up and left/right pixels
- Receptive field
  - In each stream: all pixels above and to the right/left
  - Combined: all previous pixels
- Slow sequential training process
  - Due to sequential state updates

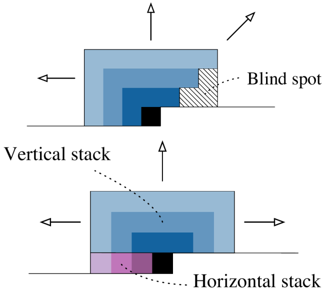




Pixel Convolutional Neural Networks

- Use limited context via CNN layers
  - Only local dependencies per layer
- Masked convolutions to ensure autoregressive property
  - Layers increase receptive field
  - Two stacks to fill blind spot: horizontal stack reads from vertical stack, not vice-versa
- Efficient parallel training, but sampling remains sequential and slow
- Extensions: WaveNet (audio) [Oord et al., 2016a], Video Pixel Networks [Kalchbrenner et al., 2017]





Class-conditional pixelCNN [Oord et al., 2016c]

- Samples single model trained across 1,000 ImageNet classes

Sandbar

Sorrel horse

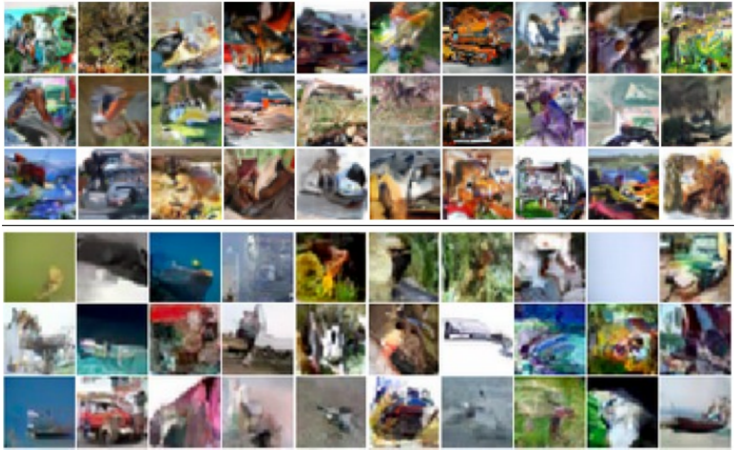
Lhasa Apso (dog)

Lawn mower

Brown bear

Robin (bird)

Images generated by PixelCNNs trained on CIFAR10



[Oord et al., 2016b] (top) and [Salimans et al., 2017] (bottom)

- Models capture texture and details relatively well
- Lacking in global structure / long range dependencies

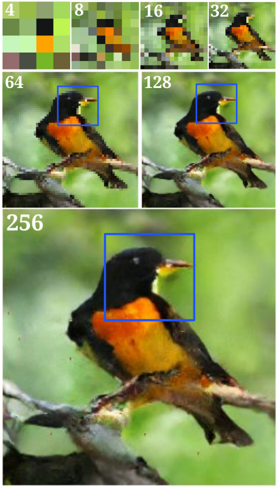
Parallel multiscale autoregressive density estimation

[Reed et al., 2017]

- Address the inherently limited sampling efficiency of autoregressive models

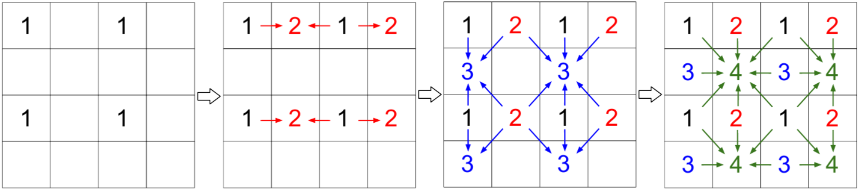
$$p(\mathbf{x}_{1:N}) = \prod_{i=1}^{N} p(x_i|\mathbf{x}_{<i})$$

- Sample image along a scale pyramid
  - Pixel-CNN for base resolution, **e.g**. 4×4
  - Autoregressive upsampling networks
- Impose group structure among pixels
  - Independent sampling within each group
  - Autoregressive sampling across groups

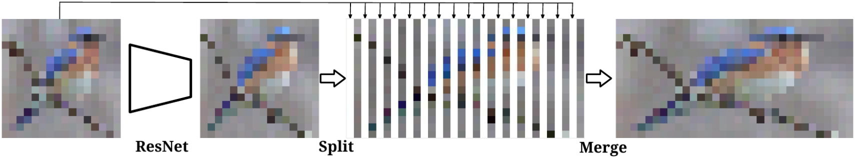


Sampling pixels in groups

- Group pixels along position in 2×2 blocks
  - Group 1 given from previous resolution
  - Sample remaining pixels in three steps

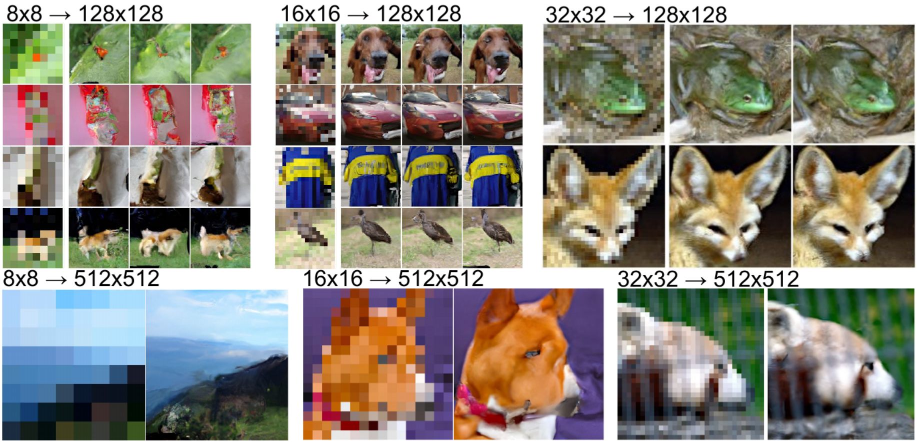


- Example network to predict group 2 from group 1
  - Use CNN without pooling to predict/sample new columns
  - Interleave pixel columns from group 1 and 2



Example results of upsampling real low-resolution images

- About 100× speed-up w.r.t. pixel-CNN sampling



Conclusion & Reminder

Several approaches to learn deep latent variable models

1. Avoid integral: Generative adversarial networks (GAN)
2. Approximate integral: Variational autoencoders (VAE)
3. Constrain the function so that we can compute the marginal (e.g. Real-NVP)
4. Do not use latent variables (e.g. PixelCNN)

Homework & Data challenge

You can do the homework and the data challenge in groups of two people.

If you do so, the two groups must be with different people.

References I

Arjovsky, M., Chintala, S., and Bottou, L. (2017).
Wasserstein generative adversarial networks.
In *ICML*.

Baldi, P. and Hornik, K. (1989).
Neural networks and principal component analysis: Learning from examples without local minima.
Neural Networks.

Bishop, C. (2006).
Pattern recognition and machine learning.
Spinger-Verlag.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2017).
Density estimation using real NVP.
In *ICLR*.

Doersch, C. (2016).
Tutorial on variational autoencoders.
arXiv:1606.05908.

Doersch, C., Gupta, A., and Efros, A. (2015).
Unsupervised visual representation learning by context prediction.
In *ICCV*.

Donahue, J., Krähenbühl, P., and Darrell, T. (2017).
Adversarial feature learning.
In *ICLR*.

Fernando, B., Bilen, H., Gavves, E., and Gould, S. (2017).
Self-supervised video representation learning with odd-one-out networks.
In *CVPR*.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014).
Generative adversarial nets.
In *NeurIPS*.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. (2017).
Improved training of Wasserstein GANs.
In *NeurIPS*.

Hou, X., Shen, L., Sun, K., and Qiu, G. (2016).
Deep feature consistent variational autoencoder.
CoRR, abs/1610.00291.

Kalchbrenner, N., van den Oord, A., Simonyan, K., Danihelka, I., Vinyals, O., Graves, A., and Kavukcuoglu, K. (2017).
Video pixel networks.
In *ICML*.

Kingma, D. and Welling, M. (2014).
Auto-encoding variational Bayes.
In *ICLR*.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013).
Efficient estimation of word representations in vector space.
In *ICLR*.

Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. (2016a).
Wavenet: a generative model for raw audio.
In *ISCA Speech Syntesis Workshop*.

Oord, A. v. d., Kalchbrenner, N., and Kavukcuoglu, K. (2016b).
Pixel recurrent neural networks.
In *ICML*.

References III

Oord, A. v. d., Kalchbrenner, N., Vinyals, O., Espeholt, L., Graves, A., and Kavukcuoglu, K. (2016c).
Conditional image generation with PixelCNN decoders.
In *NeurIPS*.

Pathak, D., Krähenbühl, P., Donahue, J., Darrell, T., and Efros, A. (2016).
Context encoders: Feature learning by inpainting.
In *CVPR*.

Pumarola, A., Agudo, A., Martinez, A. M., Sanfeliu, A., and Moreno-Noguer, F. (2018).
Ganimation: Anatomically-aware facial animation from a single image.
In *ECCV*.

Radford, A., Metz, L., and Chintala, S. (2016).
Unsupervised representation learning with deep convolutional generative adversarial networks.
In *ICLR*.

Reed, S., van den Oord, A., Kalchbrenner, N., Colmenarejo, S. G., Wang, Z., Belov, D., and de Freitas, N. (2017).
Parallel multiscale autoregressive density estimation.
In *ICML*.

Roweis, S. (1997).
EM Algorithms for PCA and SPCA.
In *NeurIPS*.

Royer, A., Kolesnikov, A., and Lampert, C. (2017).
Probabilistic image colorization.
In *BMVC*.

Salimans, T., Karpathy, A., Chen, X., and Kingma, D. (2017).
PixelCNN++: Improving the PixelCNN with discretized logistic mixture likelihood and other modifications.
In *ICLR*.

References IV

Tipping, M. E. and Bishop, C. M. (1999).
Mixtures of probabilistic principal component analysers.
Neural Computation, 11(2):443–482.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. (2017).
Unpaired image-to-image translation using cycle-consistent adversarial networks.
In *ICCV*.