APML - Generative Models

Yedid Hoshen

Dec 2020

Generative Models

- Unconditional
- Conditional
- Unsupervised - conditional

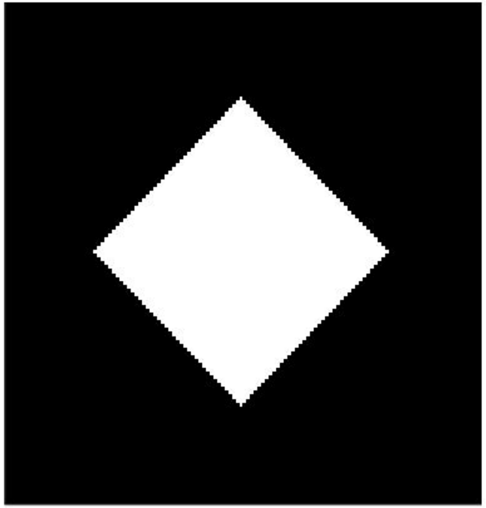
Probabilistic Models

Want to learn a model, telling us how likely an image is:

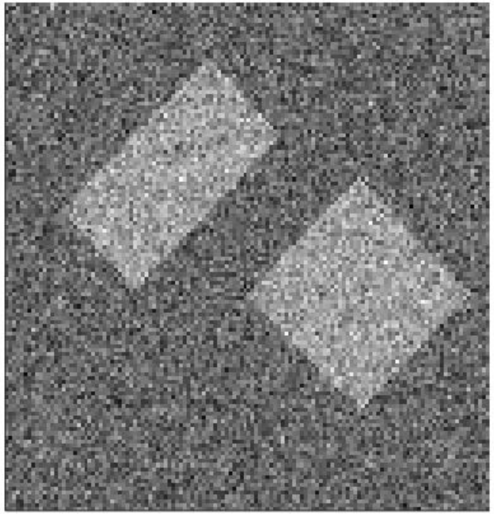
$$\max_{\theta \in \mathbb{R}^d} \frac{1}{m} \sum_{i=1}^{m} \log P_\theta(x^{(i)})$$

This type of estimation is named maximum likelihood - common in statistics

Illustration



Likely



Unlikely

Motivation

1. Generate new images:

   Sample a new image *x* according to PDF: P_θ

1. Modify images to become more natural (use PDF as image prior)

$$p(x|\dot{x}) = \frac{p(\dot{x}|x)p(x)}{p(\dot{x})} \alpha \underset{\text{likelihood}}{p(\dot{x}|x)}\underset{\text{Prior}}{p(x)}$$

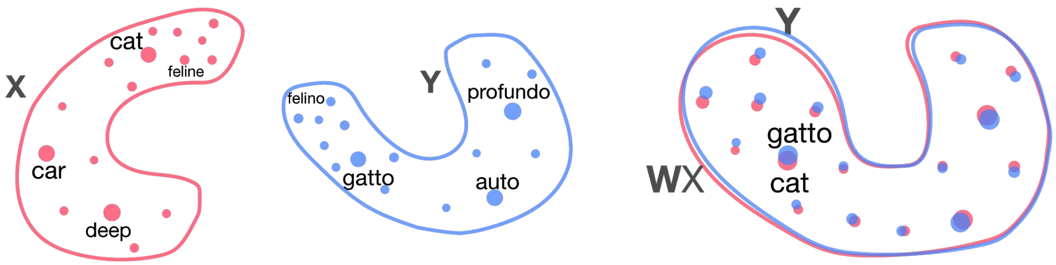
Generative Approaches

- Modeling the PDF of images is hard
- Idea: map a simple parametric distribution to the distribution of images
- Pro: Can easily be used to generated new samples
- Con: Does not give the numerical probability of images

$$g_\theta : \mathcal{Z} \rightarrow \mathcal{X}$$

Main Challenge

How to measure the difference between generated a true images?

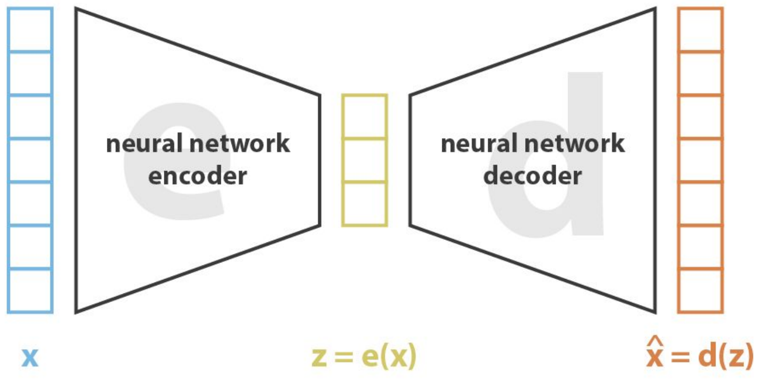


Auto-Encoder

Find a latent space of low-dimensionality

Can generate new images, given small codes

Code not normal



$$\text{loss} = \| x - \hat{x} \|^2 = \| x - d(z) \|^2 = \| x - d(e(x)) \|^2$$

Latent Optimization - Bojanowski et al. ICML'18

Optimize mapping and matching end-to-end:

$$arg \min_{G \in \mathcal{G}, x1, x2, \ldots, x_N \in \mathcal{B}^d} \sum_n d(G(x_n), y_n)$$

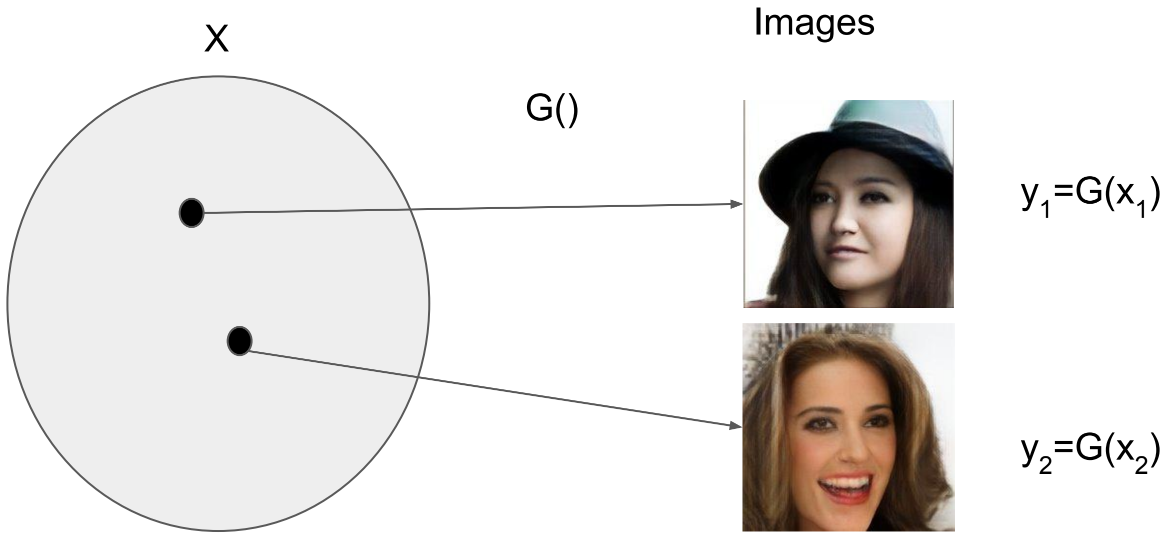
Optimize parameters of G and values of $x_1 \ldots x_N$ (inverse problem) with SGD

Clip *x* to lie in unit ball.

d() is a perceptual loss - distance between VGG activations

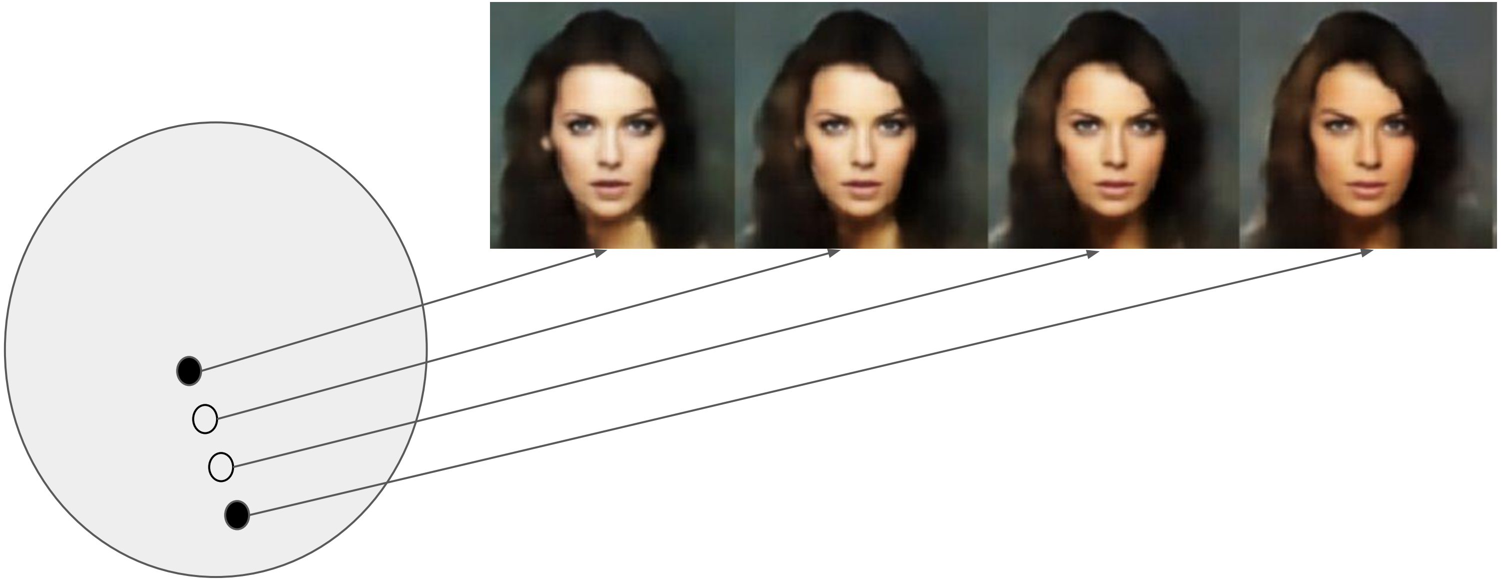
Properties of Generative Latent Optimization

Reconstruction: Given y images, we can typically find x s.t. y = G(x)



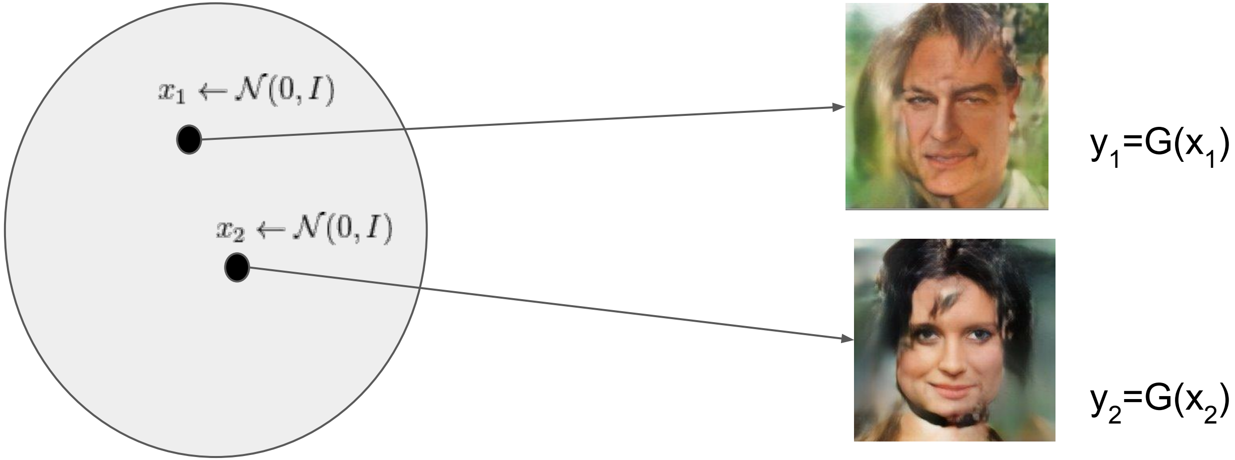
Properties of Generative Latent Optimization

Interpolation: linear combination of training *x* are semantically interpolated



Properties of Generative Latent Optimization

Generation: randomly sampled *x* are not mapped to valid images



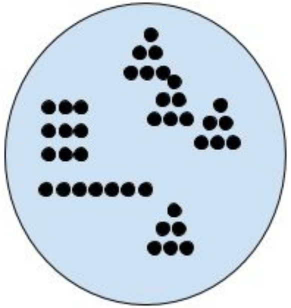
Generative Models

A generative model needs to be:

Sufficient: For every image y there must be x s.t. G(x)=y

Compact: For every x, G(x) should be a valid image in Y

GLO is not compact

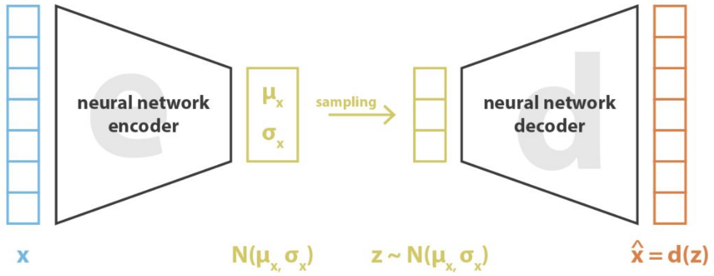


VAE: Make each code normally distributed

For each image, learn latent code mean and standard deviation

Sample new code and measure image reconstruction

Encourage code to be normally distributed



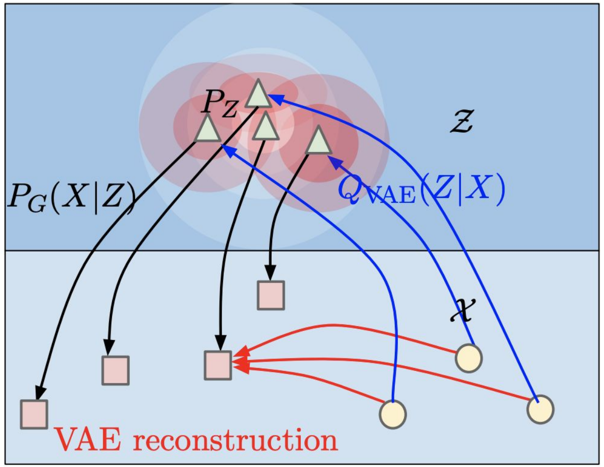
$$\text{loss} = \| x - \hat{x} \|^2 - \mathrm{KL}[\, N(\mu_x, \sigma_x), N(0, I)\,] = \| x - d(z) \|^2 - \mathrm{KL}[\, N(\mu_x, \sigma_x), N(0, I)\,]$$

WAE: Make all codes normally distributed

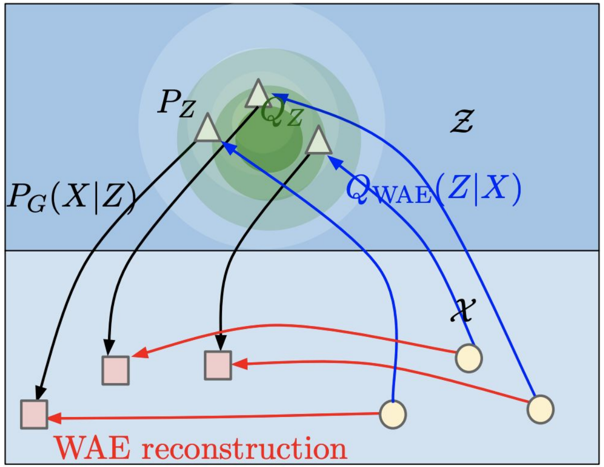
Each image is encoded into a code, and decoded back into the images

Use different distribution distance measures to ensure codes are together normal

(a) VAE



(b) WAE



VAE - Approximating the Data Distribution

We would like to approximate (where z is a hidden variable):

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

We can specify p(z), the prior of z. However computing $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ is not feasible

VAE attempt to approximate this term

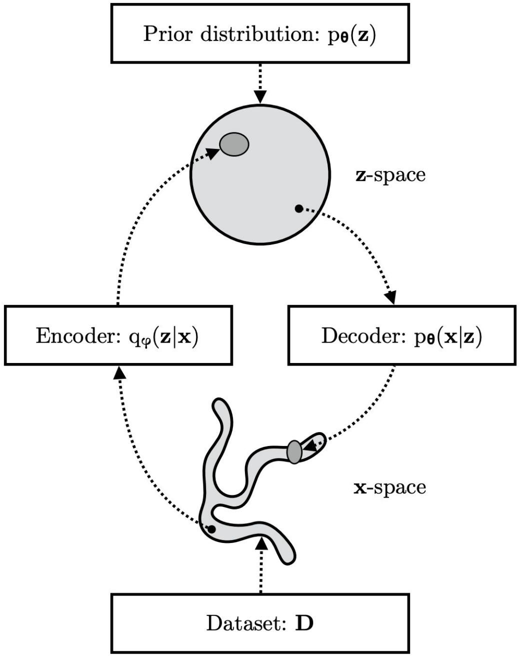
Approximating the Posterior

Idea: train encoder model $q_\phi(\mathbf{z}|\mathbf{x})$ to approximate the posterior of the decoder:

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

Train encoder such that prior of z, p(z)

P(z) is a simple distribution (Gaussian, GMM)



Evidence Lower Bound (ELBO)

We approximate the evidence using the encoder-decoder:

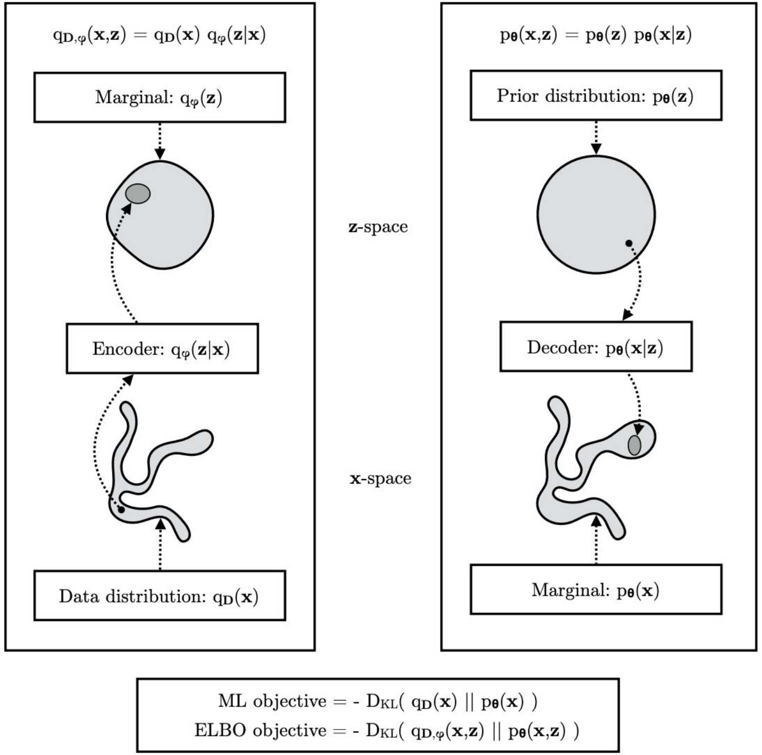
$$\begin{aligned}
\log p_{\boldsymbol{\theta}}(\mathbf{x}) &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x})\right] && (2.5)\\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}\right]\right] && (2.6)\\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}\right]\right] && (2.7)\\
&= \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right]\right]}_{\substack{=\mathcal{L}_{\boldsymbol{\theta},\phi}(\mathbf{x})\\ \text{(ELBO)}}} + \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}\right]\right]}_{=D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}))} && (2.8)
\end{aligned}$$

The second term cannot be computed, but as p(x) is constant

Improving ELBO improves the bound

Another Way of Looking at VAEs

$$\tilde{\mathcal{L}}_{\boldsymbol{\theta},\phi}(\mathbf{x};\boldsymbol{\epsilon}) = \underbrace{\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})}_{\text{Negative reconstruction error}} \underbrace{+ \log p_{\boldsymbol{\theta}}(\mathbf{z}) - \log q_{\phi}(\mathbf{z}|\mathbf{x})}_{\text{Regularization terms}}$$

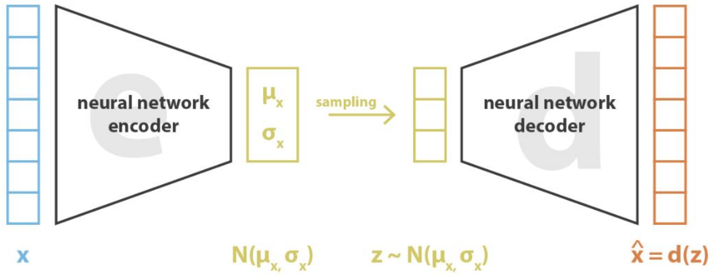


VAE: Make each code normally distributed

For each image, learn latent code mean and standard deviation

Sample new code and measure image reconstruction

Encourage code to be normally distributed

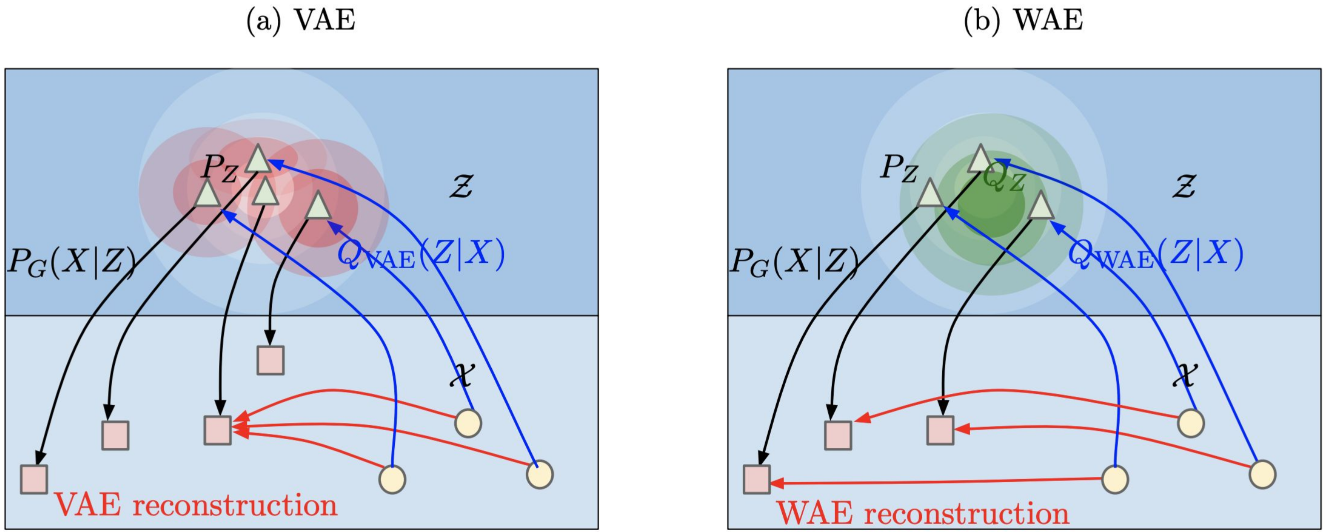


$$\text{loss} = \| x - \hat{x} \|^2 - \text{KL}[\, N(\mu_x, \sigma_x), N(0, I) \,] = \| x - d(z) \|^2 - \text{KL}[\, N(\mu_x, \sigma_x), N(0, I) \,]$$

WAE: Make all codes normally distributed

Each image is encoded into a code, and decoded back into the images

Use different distribution distance measures to ensure codes are together normal



Maximum Mean Discrepancy (MMD)

Measure distance between distributions

Distance between all moment of distribution

$$\mathrm{MMD}_k(P_Z, Q_Z) = \left\| \int_{\mathcal{Z}} k(z,\cdot)\,dP_Z(z) - \int_{\mathcal{Z}} k(z,\cdot)\,dQ_Z(z) \right\|_{\mathcal{H}_k},$$

$$\frac{1}{n}\sum_{i=1}^{n} c\big(x_i, G_\theta(\tilde{z}_i)\big) + \frac{\lambda}{n(n-1)}\sum_{\ell\neq j} k(z_\ell, z_j) + \frac{\lambda}{n(n-1)}\sum_{\ell\neq j} k(\tilde{z}_\ell, \tilde{z}_j) - \frac{2\lambda}{n^2}\sum_{\ell,j} k(z_\ell, \tilde{z}_j)$$

Maximum Mean Discrepancy (MMD)

Measure distance between distributions

Distance between all moment of distribution

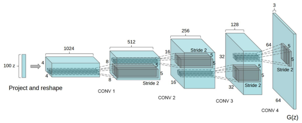
$$\mathrm{MMD}_k(P_Z, Q_Z) = \left\| \int_{\mathcal{Z}} k(z, \cdot) dP_Z(z) - \int_{\mathcal{Z}} k(z, \cdot) dQ_Z(z) \right\|_{\mathcal{H}_k},$$

$$\frac{1}{n} \sum_{i=1}^{n} c\big(x_i, G_\theta(\tilde{z}_i)\big) + \frac{\lambda}{n(n-1)} \sum_{\ell \neq j} k(z_\ell, z_j)$$
$$+ \frac{\lambda}{n(n-1)} \sum_{\ell \neq j} k(\tilde{z}_\ell, \tilde{z}_j) - \frac{2\lambda}{n^2} \sum_{\ell, j} k(z_\ell, \tilde{z}_j)$$

Implicit Maximum Likelihood Estimation (IMLE)

Li and Malik, 2018

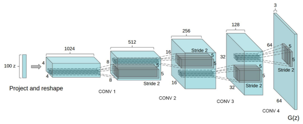
Initialize generator G()

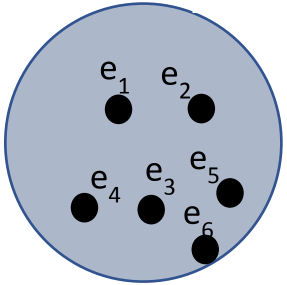


Implicit Maximum Likelihood Estimation (IMLE)

Li and Malik, 2018

Generate random noise vectors

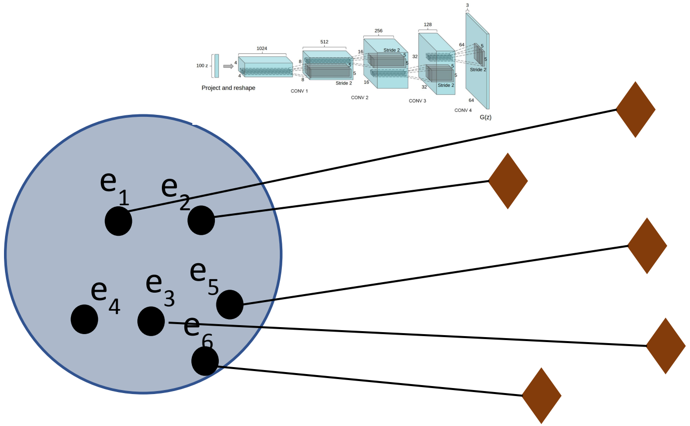




Implicit Maximum Likelihood Estimation (IMLE)

Li and Malik, 2018

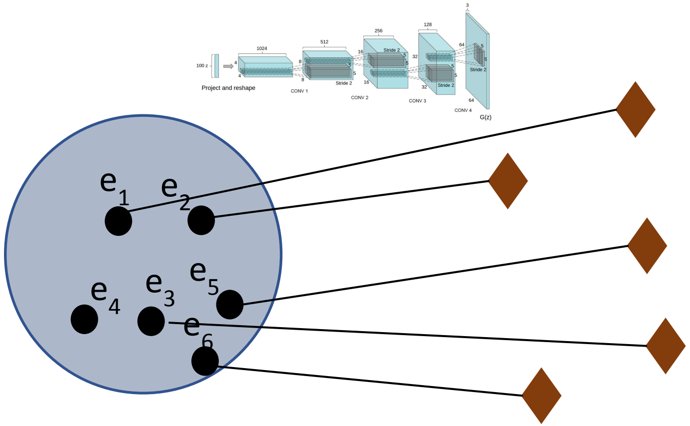
Map codes to images using generator e -> G(e)

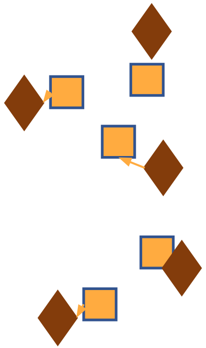


Implicit Maximum Likelihood Estimation (IMLE)

Li and Malik, 2018

For each training set image x: find nearest generated image G(e_j)



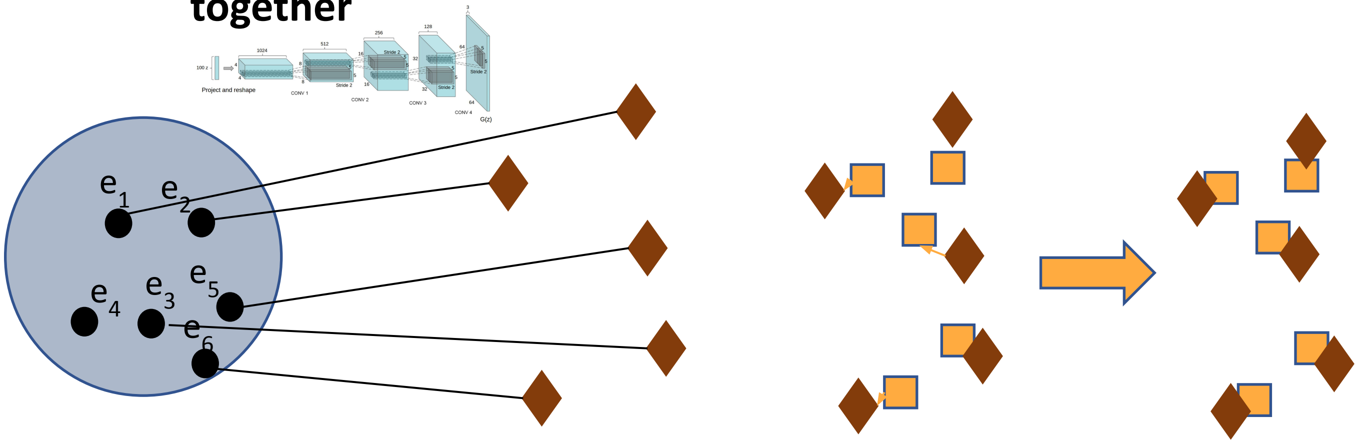


$$e_i \;=\; arg\,\min\nolimits_{e_j} \|G(e_j), x_i|_2^2$$

Implicit Maximum Likelihood Estimation (IMLE)

Li and Malik, 2018

Train generator G() so that nearest neighbor pairs are closer together



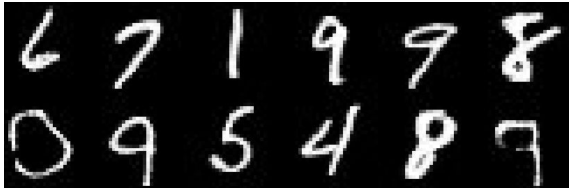
$$G = arg\min_{\tilde{G}} \sum_i ||\tilde{G}(e_i), x_i||_2^2$$

Properties of IMLE

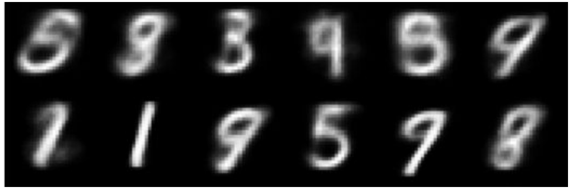
+ Fills up the entire latent space
+ No-mode dropping
  + Li and Malik: Maximum Likelihood solution (under some assumptions)

- Sensitive to metric used (like all other NN methods)
  - Blurry generated images

GAN



IMLE

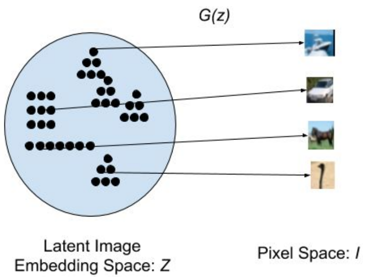


Generative Latent Nearest Neighbors (GLANN)

With J Malik

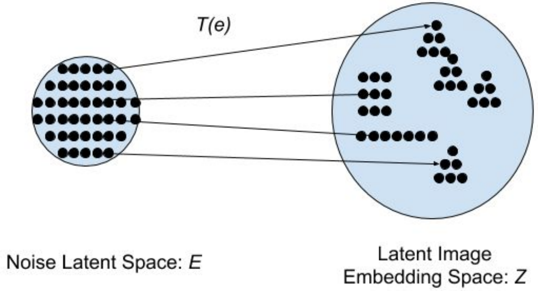
- Decompose images to a semantic latent space (GLO)
- Good latent space and generator but not compact

$$arg \min_{\tilde{G},\{z_i\}} \sum_i \ell_{perceptual}(\tilde{G}(z_i), x_i) \quad s.t. \quad \|z_i\| = 1$$



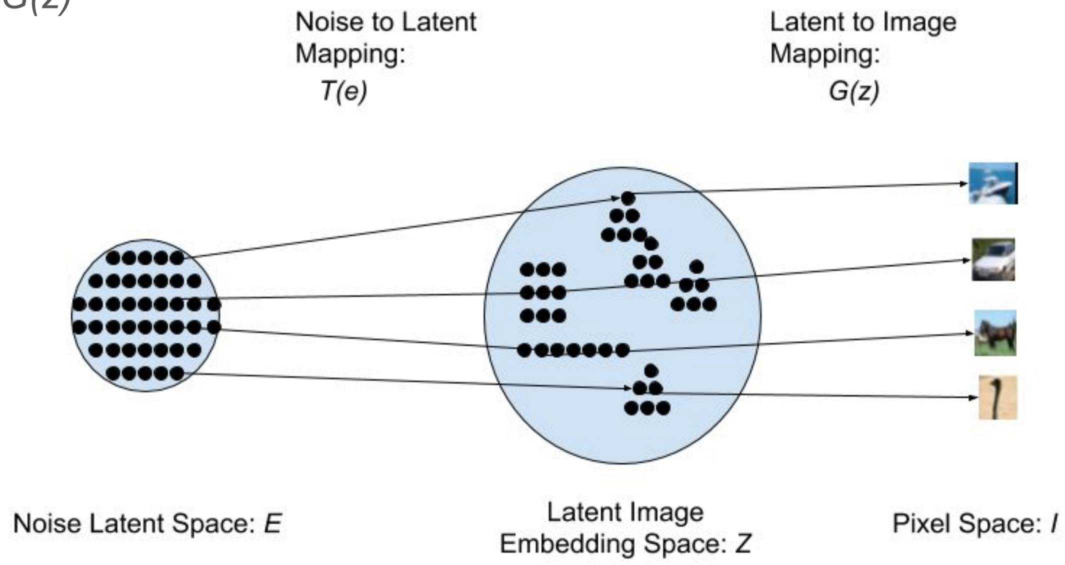
- Map noise distribution to latent code distribution (IMLE)
- Latent space is semantic and linear:
  - Euclidean metric is sufficient

$$T = \arg\min_{\tilde{T}} \sum_t \|z_t - \tilde{T}(e_t)\|_2^2$$



GLANN: Novel Image Generation

- To generate a new image:
  - Sample noise vector: *e*
  - Generate latent code: *z = T(e)*
  - Generate new image: *I = G(z)*

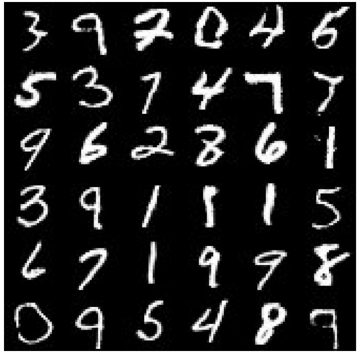


Qualitative Evaluation - MNIST

IMLE

GLO

GAN



GLANN

Real

Evaluating Image Generation Models
(Lucic et al.)

- Comparison against a baseline of 7 GANs and VAE
- Each was tested on 100 hyperparameters configuration, pick the best
- Evaluated using FID: a standard measure of distribution distance
    - Smaller is better

	Adversarial					Non-Adversarial		
Dataset	MM GAN	NS GAN	LSGAN	WGAN	BEGAN	VAE	GLO	Ours
MNIST	9.8 ± 0.9	6.8 ± 0.5	7.8 ± 0.6	$\mathbf{6.7} \pm 0.4$	13.1 ± 1.0	23.8 ± 0.6	49.6 ± 0.3	8.6 ± 0.1
Fashion	29.6 ± 1.6	26.5 ± 1.6	30.7 ± 2.2	21.5 ± 1.6	22.9 ± 0.9	58.7 ± 1.2	57.7 ± 0.4	$\mathbf{13.0} \pm 0.1$
Cifar10	72.7 ± 3.6	58.5 ± 1.9	87.1 ± 47.5	55.2 ± 2.3	71.4 ± 1.6	155.7 ± 11.6	65.4 ± 0.2	$\mathbf{46.5} \pm 0.2$
CelebA	65.6 ± 4.2	55.0 ± 3.3	53.9 ± 2.8	41.3 ± 2.0	$\mathbf{38.9} \pm 0.9$	85.7 ± 3.8	52.4 ± 0.5	46.3 ± 0.1

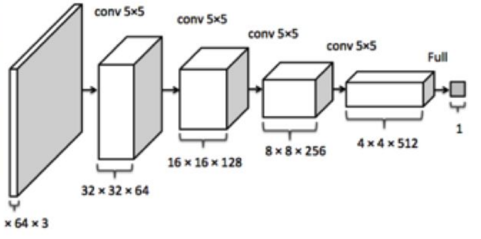
CelebA-HQ 256X256 Face Interpolation



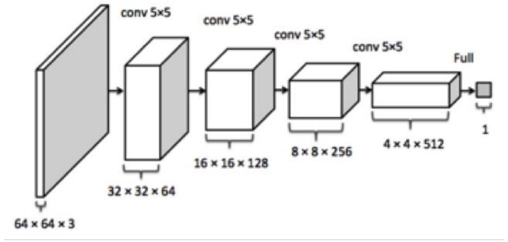

CelebA-HQ 256X256 Face Interpolation

Deep Domain Discriminators

- Goal: Determine if image belongs to an image domain
- Example: Is this a natural image of a shoe?
- Train a deep discriminator between shoe/non-shoe images



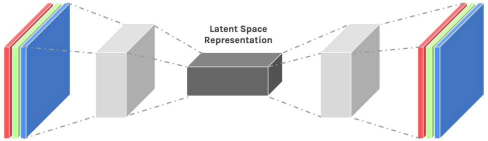
True

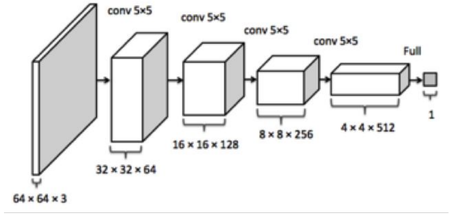


False

Deep Domain Mapping Network

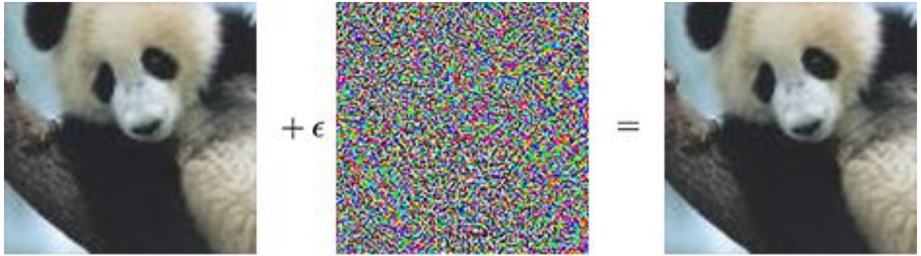
- Train network M() to map images between domain A to B
- Discriminator determines if the mapped image looks like B





- Discriminator only works on natural images on which it was trained
- M() makes small perturbations in input so that discriminator misclassifies

Panda: 57.7%



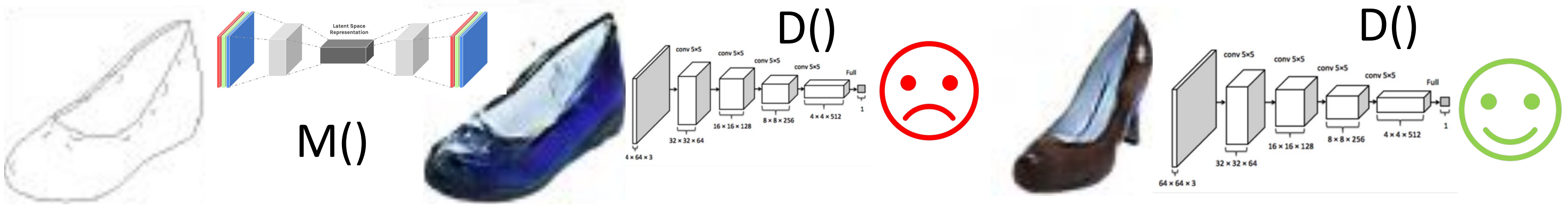
Gibbon: 99.3%

Credit: OpenAI

Adversarial Training

Goodfellow et al. 2014

- Adversarial solution: simultaneously train D() and M()
- Discriminator trained to separate generated and real images

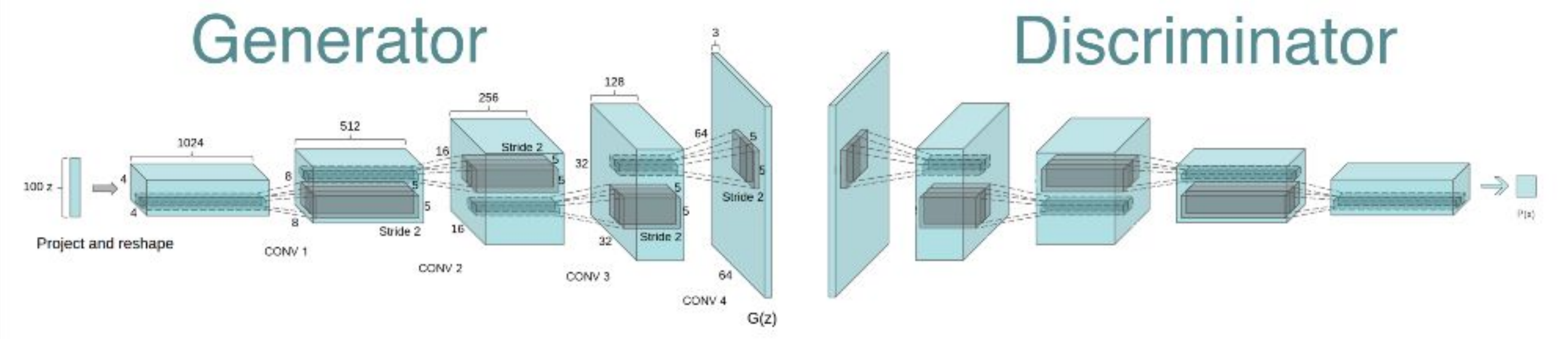


- Mapping is trained to be so realistic as to fool discriminator

Generative Adversarial Networks (GANs)

Introduced by Goodfellow et al. NIPS’14

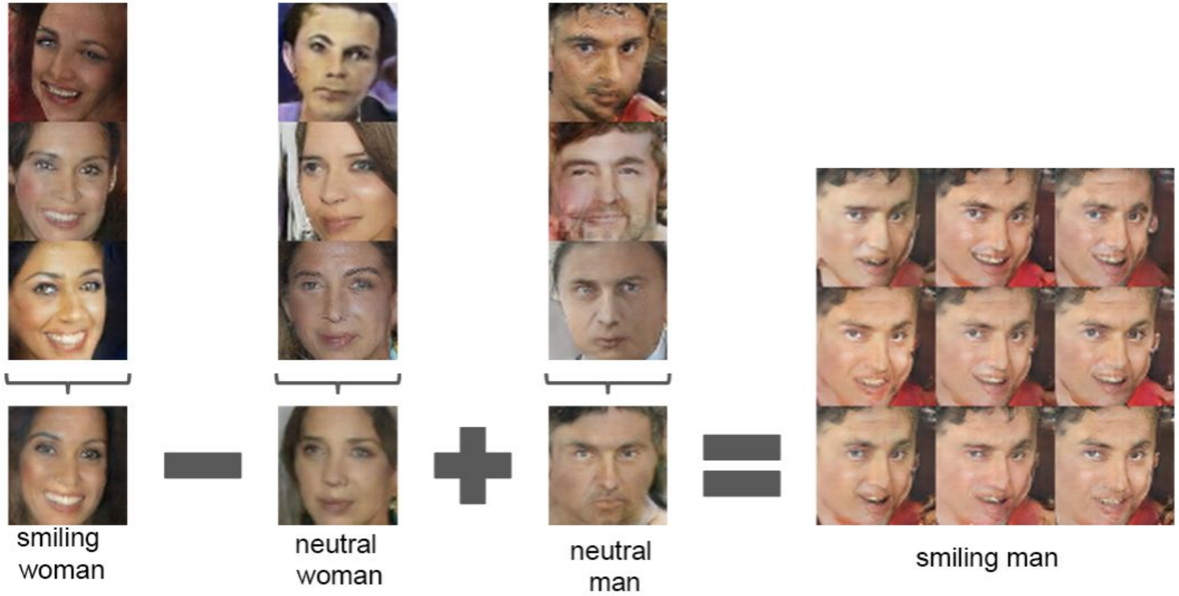
First stable GAN is DCGAN, Radford et al.



DCGAN

High quality generated images on low-res datasets

Vector arithmetics of latent space



Adversarial Training

A discriminator is trained to maximize difference between real and fake

The generator is trained to minimize it

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

Issues with the DCGAN objective

- It can saturate, no more gradients
  - One simple fix: LS-GAN

$$\min_D V_{\mathrm{LSGAN}}(D) = \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\mathrm{data}}(\boldsymbol{x})}\left[(D(\boldsymbol{x})-1)^2\right] + \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z})))^2\right]$$

$$\min_G V_{\mathrm{LSGAN}}(G) = \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z}))-1)^2\right].$$

- Discriminator may overfit, no more gradients

Probability Distance Measures

- The *Total Variation* (TV) distance

$$\delta(\mathbb{P}_r, \mathbb{P}_g) = \sup_{A \in \Sigma} |\mathbb{P}_r(A) - \mathbb{P}_g(A)| \ .$$

- The *Kullback-Leibler* (KL) divergence

$$KL(\mathbb{P}_r \| \mathbb{P}_g) = \int \log\left(\frac{P_r(x)}{P_g(x)}\right) P_r(x) d\mu(x) \ ,$$

- The *Jensen-Shannon* (JS) divergence

$$JS(\mathbb{P}_r, \mathbb{P}_g) = KL(\mathbb{P}_r \| \mathbb{P}_m) + KL(\mathbb{P}_g \| \mathbb{P}_m) \ ,$$

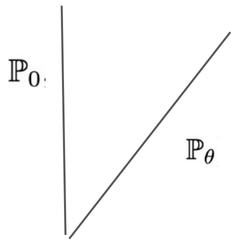
where $\mathbb{P}_m$ is the mixture $(\mathbb{P}_r + \mathbb{P}_g)/2$. This divergence is symmetrical and always defined because we can choose $\mu = \mathbb{P}_m$.

- The *Earth-Mover* (EM) distance or Wasserstein-1

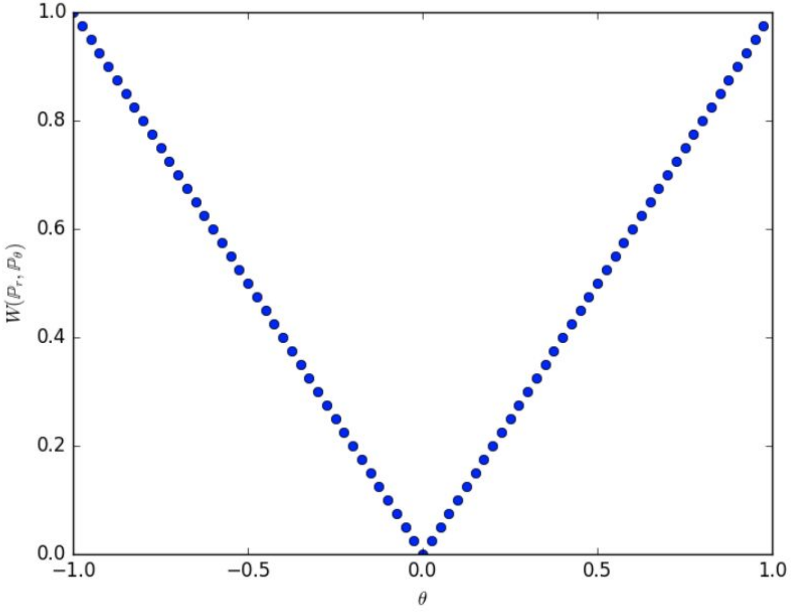
$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y) \sim \gamma} \big[\|x - y\| \big] \ , \tag{1}$$

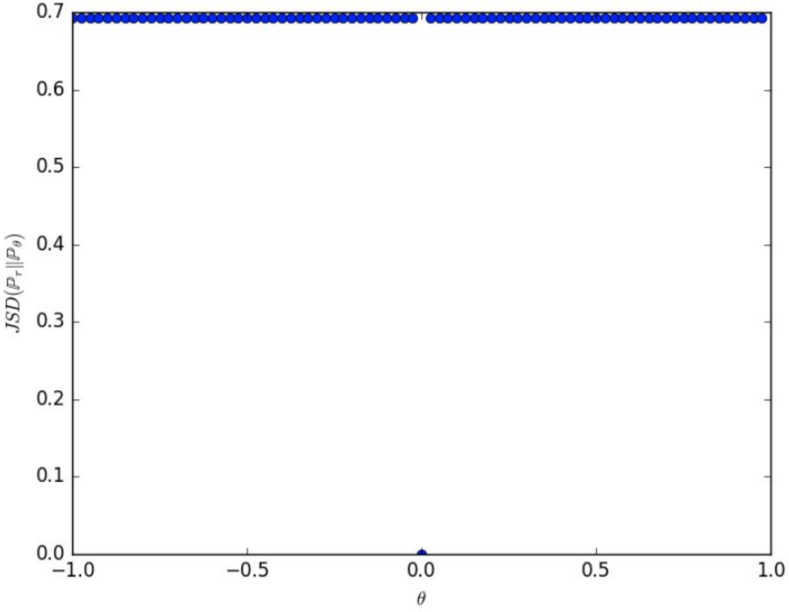
Different distance measures on toy example

- $W(\mathbb{P}_0, \mathbb{P}_\theta) = |\theta|,$
- $JS(\mathbb{P}_0, \mathbb{P}_\theta) = \begin{cases} \log 2 & \text{if } \theta \neq 0 \ , \\ 0 & \text{if } \theta = 0 \ , \end{cases}$
- $KL(\mathbb{P}_\theta \| \mathbb{P}_0) = KL(\mathbb{P}_0 \| \mathbb{P}_\theta) = \begin{cases} +\infty & \text{if } \theta \neq 0 \ , \\ 0 & \text{if } \theta = 0 \ , \end{cases}$
- and $\delta(\mathbb{P}_0, \mathbb{P}_\theta) = \begin{cases} 1 & \text{if } \theta \neq 0 \ , \\ 0 & \text{if } \theta = 0 \ . \end{cases}$



Different distance measures on toy example (2)





Dual of the Wasserstein Distance

Kantorovich-Rubinstein duality gives an another formulation for the W distance

f must be Lipschitz-1

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_\theta}[f(x)]$$

The Wasserstein GAN

Finally this can be seen as an adversarial task:

$$\max_{\|f\|_L \leq 1} \mathbb{E}_{x \sim \mathbb{P}_r}[f(x)] - \mathbb{E}_{x \sim \mathbb{P}_\theta}[f(x)]$$

$$\nabla_\theta W(\mathbb{P}_r, \mathbb{P}_\theta) = -\mathbb{E}_{z \sim p(z)}[\nabla_\theta f(g_\theta(z))]$$

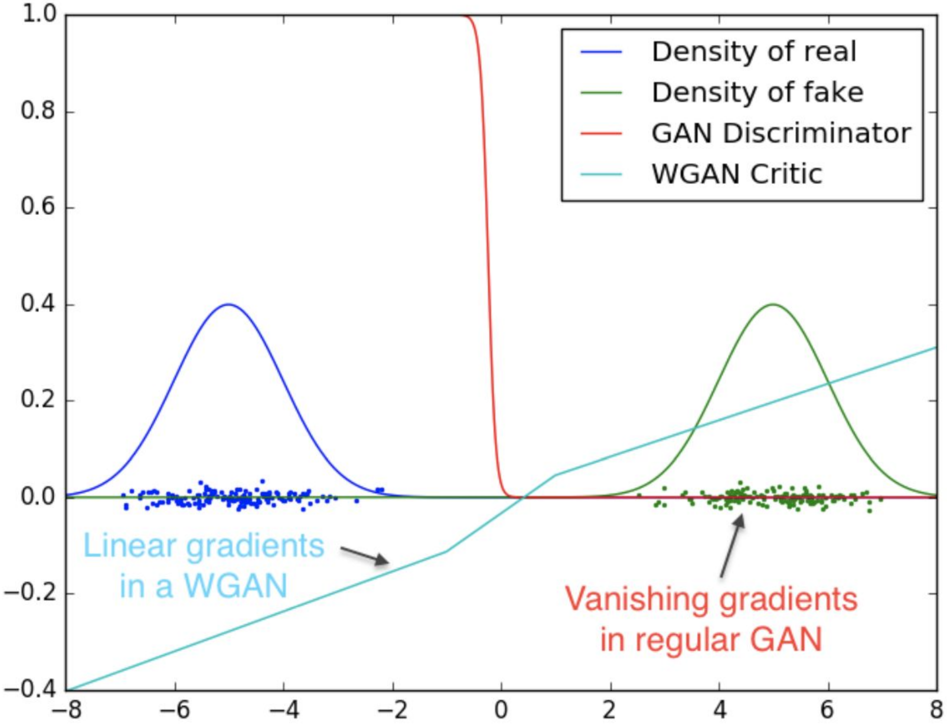
WGAN algorithm

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.
Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

while θ has not converged **do**
    for $t = 0, ..., n_{\text{critic}}$ **do**
        Sample $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$ a batch from the real data.
        Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
        $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))\right]$
        $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
        $w \leftarrow \text{clip}(w, -c, c)$
    end for
    Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
    $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))$
    $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
end while

WGAN vs DCGAN



How to enforce Lipschitz-1 discriminator f

WGAN: weight clipping $\mathrm{clip}(w, -c, c)$

WGAN-GP: $\mathrm{E}_{p_{\mathcal{D}}(x)}\left[|D_\psi(x)|^2 + \|\nabla_x D_\psi(x)\|^2\right]$

Mescheder et al.: $$R_1(\psi) := \frac{\gamma}{2}\mathrm{E}_{p_{\mathcal{D}}(x)}\left[\|\nabla D_\psi(x)\|^2\right]$$

Zhang et al.: $$\min_D L_{cr} = \min_D \sum_{j=m}^{n} \lambda_j \left\|D_j(x) - D_j(T(x))\right\|^2,$$

Spectral Normalization

Ensure that every layer in the network is Lip-1

Fast method for speeding up eigenvalue computation based on power method

$$\sigma(A) := \max_{\boldsymbol{h}:\boldsymbol{h}\neq\boldsymbol{0}} \frac{\|A\boldsymbol{h}\|_2}{\|\boldsymbol{h}\|_2} = \max_{\|\boldsymbol{h}\|_2\leq 1} \|A\boldsymbol{h}\|_2,$$

$$\bar{W}_{\mathrm{SN}}(W) := W/\sigma(W).$$

Inception Score

Network needs to be confident, global class distribution needs to match data

Con: Classes must be related to ImageNet

$$\mathrm{IS}(G) = \exp\left(\mathbb{E}_{\mathbf{x}\sim p_g} D_{KL}(\, p(y|\mathbf{x}) \parallel p(y)\,)\right),$$

$$\boxed{D_{\mathrm{KL}}(P \parallel Q) = -\sum_{x\in\mathcal{X}} P(x)\log\left(\frac{Q(x)}{P(x)}\right) \quad \textbf{\textit{(Eq.1)}}}$$

Frechet Inception Distance

Compute features for 10k real and fake images

Fit a Gaussian to each, compute distance between Gaussians

$$\text{FID} = ||\mu_r - \mu_g||^2 + \text{Tr}(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2}),$$