Generative Models: Variational Autoencoders

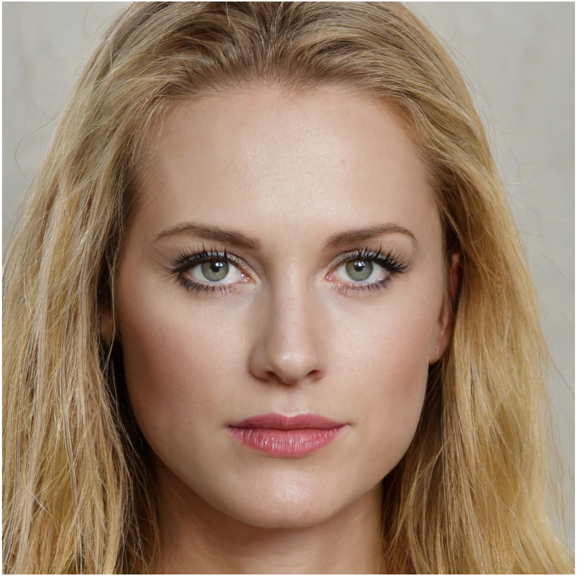
Foundations of Data Analysis

April 28, 2022

These are not real people

Karras et al., CVPR 2020, and `thispersondoesnotexist.com`

These are not real people



Karras et al., CVPR 2020, and thispersondoesnotexist.com

These are not real people

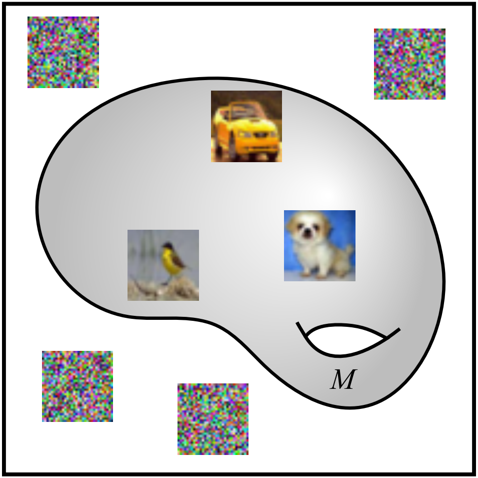
Karras et al., CVPR 2020, and `thispersondoesnotexist.com`

These are not real people

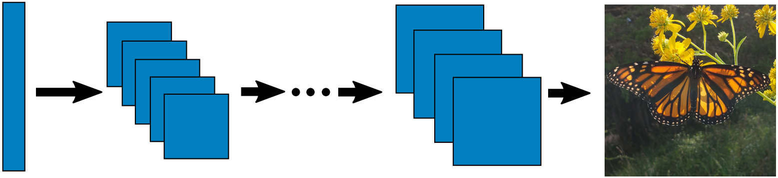
Karras et al., CVPR 2020, and `thispersondoesnotexist.com`

Manifold Hypothesis

Real data lie near lower-dimensional manifolds



Deep Generative Models



Input:

$z \in \mathbb{R}^d$

$z \sim N(0, I)$

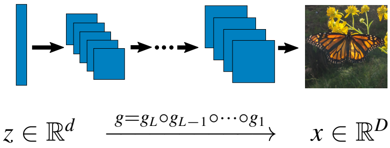
$\xrightarrow{g = g_L \circ g_{L-1} \circ \cdots \circ g_1}$

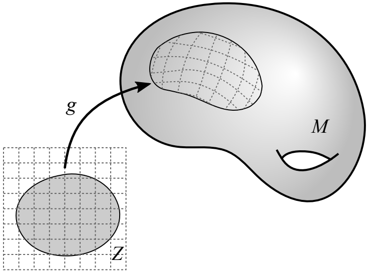
Output:

$x \in \mathbb{R}^D$

$$d << D$$

Generative Models as Immersed Manifolds





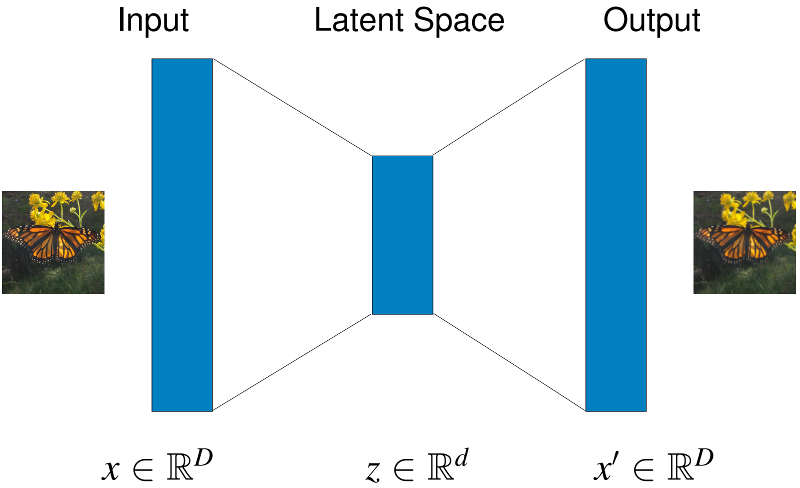
1. g should be differentiable
2. Jacobian matrix, Dg, should be full rank

Shao, Kumar, Fletcher, The Riemannian Geometry of Deep Generative Models, DiffCVML 2018.

Talking about this paper:

Diederik Kingma and Max Welling, Auto-Encoding Variational Bayes, In *International Conference on Learning Representation (ICLR)*, 2014.

Autoencoders



$$d << D$$

Autoencoders

- Linear activation functions give you PCA

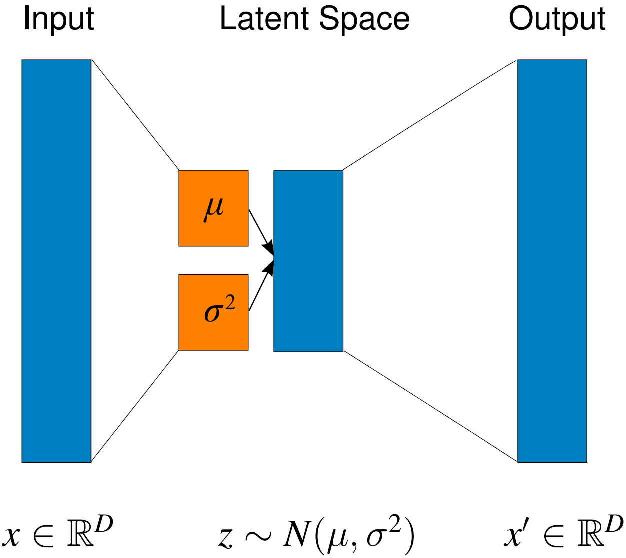
Autoencoders

- Linear activation functions give you PCA
- Training:
    1. Given data x, feedforward to x' output
    2. Compute loss, e.g., $L(x, x') = \|x - x'\|^2$
    3. Backpropagate loss gradient to update weights

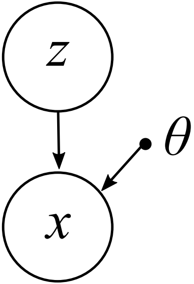
Autoencoders

- Linear activation functions give you PCA
- Training:
  1. Given data x, feedforward to x' output
  2. Compute loss, e.g., $L(x, x') = \|x - x'\|^2$
  3. Backpropagate loss gradient to update weights
- **Not** a generative model!

Variational Autoencoders



Generative Models

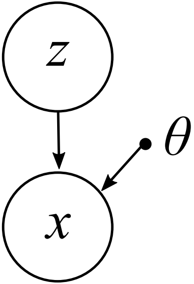


Sample a new x in two steps:

Prior: $p(z)$

Generator: $p_\theta(x \mid z)$

Generative Models



Sample a new x in two steps:

Prior: $p(z)$
Generator: $p_\theta(x \mid z)$

Now the analogy to the "encoder" is:

Posterior: $p(z \mid x)$

Bayesian Inference

Posterior via Bayes' Rule:

$$p(z \mid x) = \frac{p_\theta(x \mid z)p(z)}{p(x)}$$
$$= \frac{p_\theta(x \mid z)p(z)}{\int p_\theta(x \mid z)p(z)dz}$$

Integral in denominator is (usually) intractable!

Kullback-Leibler Divergence

$$
\begin{aligned}
D_{\mathrm{KL}}(q \| p) &= -\int q(z) \log \left(\frac{p(z)}{q(z)} \right) dz \\
&= E_q \left[-\log \left(\frac{p}{q} \right) \right]
\end{aligned}
$$

Kullback-Leibler Divergence

$$\begin{aligned} D_{\mathrm{KL}}(q\|p) &= -\int q(z)\log\left(\frac{p(z)}{q(z)}\right)dz \\ &= E_q\left[-\log\left(\frac{p}{q}\right)\right] \end{aligned}$$

The average *information gained* from moving from q to p

Variational Inference

Approximate intractable posterior $p(z \mid x)$ with a manageable distribution $q(z)$

Variational Inference

Approximate intractable posterior $p(z \mid x)$ with a manageable distribution $q(z)$

Minimize the KL divergence: $D_{\mathrm{KL}}(q(z) \| p(z \mid x))$

Evidence Lower Bound (ELBO)

$$
\begin{aligned}
&D_{\mathrm{KL}}(q(z) \| p(z \mid x)) \\
&\qquad = E_q \left[-\log \left(\frac{p(z \mid x)}{q(z)} \right) \right]
\end{aligned}
$$

Evidence Lower Bound (ELBO)

$$
\begin{aligned}
D_{\mathrm{KL}}(q(z) \| p(z \mid x)) \\
&= E_q\left[-\log\left(\frac{p(z \mid x)}{q(z)}\right)\right] \\
&= E_q\left[-\log\frac{p(z, x)}{q(z)p(x)}\right]
\end{aligned}
$$

Evidence Lower Bound (ELBO)

$$
\begin{aligned}
D_{\mathrm{KL}}(q(z) \| p(z \mid x)) \\
&= E_q\left[-\log\left(\frac{p(z \mid x)}{q(z)}\right)\right] \\
&= E_q\left[-\log\frac{p(z, x)}{q(z)p(x)}\right] \\
&= E_q[-\log p(z, x) + \log q(z) + \log p(x)]
\end{aligned}
$$

Evidence Lower Bound (ELBO)

$$
\begin{aligned}
D_{\mathrm{KL}}(q(z) \| p(z \mid x)) \\
&= E_q\left[-\log\left(\frac{p(z \mid x)}{q(z)}\right)\right] \\
&= E_q\left[-\log\frac{p(z,x)}{q(z)p(x)}\right] \\
&= E_q[-\log p(z,x) + \log q(z) + \log p(x)] \\
&= -E_q[\log p(z,x)] + E_q[\log q(z)] + \log p(x)
\end{aligned}
$$

Evidence Lower Bound (ELBO)

$$
\begin{aligned}
&D_{\mathrm{KL}}(q(z) \| p(z \mid x)) \\
&\quad = E_q\left[-\log\left(\frac{p(z \mid x)}{q(z)}\right)\right] \\
&\quad = E_q\left[-\log\frac{p(z, x)}{q(z)p(x)}\right] \\
&\quad = E_q[-\log p(z, x) + \log q(z) + \log p(x)] \\
&\quad = -E_q[\log p(z, x)] + E_q[\log q(z)] + \log p(x)
\end{aligned}
$$

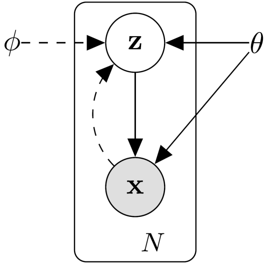
$$\log p(x) = D_{\mathrm{KL}}(q(z) \| p(z \mid x)) + L[q(z)]$$

ELBO: $L[q(z)] = E_q[\log p(z, x)] - E_q[\log q(z)]$

Variational Autoencoder

$q_\phi(z \mid x)$

Encoder Network



$p_\theta(x \mid z)$

Decoder Network

Maximize ELBO:

$$\mathcal{L}(\theta, \phi, x) = E_{q_\phi}[\log p_\theta(x, z) - \log q_\phi(z \mid x)]$$

VAE ELBO

$$\mathcal{L}(\theta, \phi, x) = E_{q_\phi}[\log p_\theta(x, z) - \log q_\phi(z \mid x)]$$

VAE ELBO

$$\begin{aligned}\mathcal{L}(\theta, \phi, x) &= E_{q_\phi}[\log p_\theta(x, z) - \log q_\phi(z \mid x)] \\ &= E_{q_\phi}[\log p_\theta(z) + \log p_\theta(x \mid z) - \log q_\phi(z \mid x)]\end{aligned}$$

VAE ELBO

$$
\begin{aligned}
\mathcal{L}(\theta, \phi, x) &= E_{q_\phi}[\log p_\theta(x, z) - \log q_\phi(z \mid x)] \\
&= E_{q_\phi}[\log p_\theta(z) + \log p_\theta(x \mid z) - \log q_\phi(z \mid x)] \\
&= E_{q_\phi}\left[\log \frac{p_\theta(z)}{q_\phi(z \mid x)} + \log p_\theta(x \mid z)\right]
\end{aligned}
$$

VAE ELBO

$$
\begin{aligned}
\mathcal{L}(\theta, \phi, x) &= E_{q_\phi}[\log p_\theta(x, z) - \log q_\phi(z \mid x)] \\
&= E_{q_\phi}[\log p_\theta(z) + \log p_\theta(x \mid z) - \log q_\phi(z \mid x)] \\
&= E_{q_\phi}\left[\log \frac{p_\theta(z)}{q_\phi(z \mid x)} + \log p_\theta(x \mid z)\right] \\
&= -D_{\mathrm{KL}}(q_\phi(z \mid x) \| p_\theta(z)) + E_{q_\phi}[\log p_\theta(x \mid z)]
\end{aligned}
$$

VAE ELBO

$$\begin{aligned}
\mathcal{L}(\theta, \phi, x) &= E_{q_\phi}[\log p_\theta(x, z) - \log q_\phi(z \mid x)] \\
&= E_{q_\phi}[\log p_\theta(z) + \log p_\theta(x \mid z) - \log q_\phi(z \mid x)] \\
&= E_{q_\phi}\left[\log \frac{p_\theta(z)}{q_\phi(z \mid x)} + \log p_\theta(x \mid z)\right] \\
&= -D_{\mathrm{KL}}(q_\phi(z \mid x) \| p_\theta(z)) + E_{q_\phi}[\log p_\theta(x \mid z)]
\end{aligned}$$

Problem: Gradient $\nabla_\phi E_{q_\phi}[\log p_\theta(x \mid z)]$ is intractable!

VAE ELBO

$$\begin{aligned}
\mathcal{L}(\theta, \phi, x) &= E_{q_\phi}[\log p_\theta(x, z) - \log q_\phi(z \mid x)] \\
&= E_{q_\phi}[\log p_\theta(z) + \log p_\theta(x \mid z) - \log q_\phi(z \mid x)] \\
&= E_{q_\phi}\left[\log \frac{p_\theta(z)}{q_\phi(z \mid x)} + \log p_\theta(x \mid z)\right] \\
&= -D_{\mathrm{KL}}(q_\phi(z \mid x) \| p_\theta(z)) + E_{q_\phi}[\log p_\theta(x \mid z)]
\end{aligned}$$

Problem: Gradient $\nabla_\phi E_{q_\phi}[\log p_\theta(x \mid z)]$ is intractable!

Use Monte Carlo approx., sampling $z^{(s)} \sim q_\phi(z \mid x)$:

$$\nabla_\phi E_{q_\phi}[\log p_\theta(x \mid z)] \approx \frac{1}{S} \sum_{s=1}^{S} \log p_\theta(x \mid z) \nabla_\phi \log q_\phi(z^{(s)} \mid x)$$

Reparameterization Trick

What about the other term?

$$-D_{\mathrm{KL}}(q_\phi(z \mid x) \| p_\theta(z))$$

Reparameterization Trick

What about the other term?

$$-D_{\mathrm{KL}}(q_\phi(z \mid x) \| p_\theta(z))$$

Says encoder, $q_\phi(z \mid x)$, should make code z look like prior distribution

Reparameterization Trick

What about the other term?

$$-D_{\mathrm{KL}}(q_\phi(z \mid x) \| p_\theta(z))$$

Says encoder, $q_\phi(z \mid x)$, should make code z look like prior distribution

Instead of encoding z, encode parameters for a normal distribution, $N(\mu, \sigma^2)$

Reparameterization Trick

$$q_\phi(z_j \mid x^{(i)}) = N(\mu_j^{(i)}, \sigma_j^{2(i)})$$
$$p_\theta(z) = N(0, I)$$

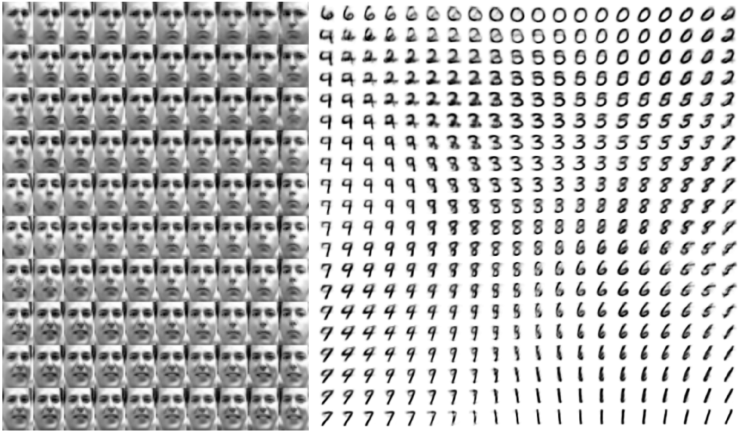
Reparameterization Trick

$$q_\phi(z_j \mid x^{(i)}) = N(\mu_j^{(i)}, \sigma_j^{2(i)})$$
$$p_\theta(z) = N(0, I)$$

KL divergence between these two is:

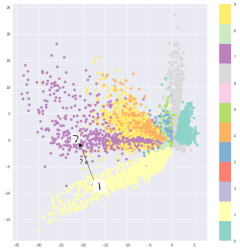
$$D_{\mathrm{KL}}(q_\phi(z \mid x^{(i)}) \| p_\theta(z)) = -\frac{1}{2} \sum_{j=1}^{d} \left(1 + \log(\sigma_j^{2(i)}) - (\mu_j^{(i)})^2 - \sigma_j^{2(i)}\right)$$

Results from Kingma & Welling



Why Do Variational?

Example trained on MNIST:

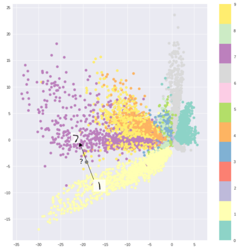


Autoencoder
(reconstruction loss)

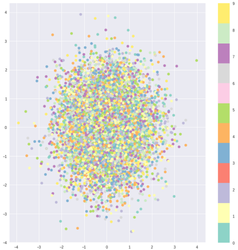
From: this webpage

Why Do Variational?

Example trained on MNIST:



Autoencoder
(reconstruction loss)

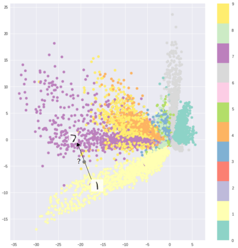


KL divergence only

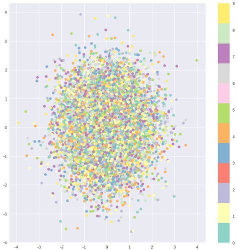
From: this webpage

Why Do Variational?

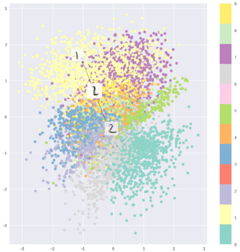
Example trained on MNIST:



Autoencoder
(reconstruction loss)



KL divergence only



VAE
(KL + recon. loss)

From: this webpage

Applications of Autoencoder / VAE Models

Image-to-Image Networks

Instead of trying to reconstruct the original input:

1. Encode input: $z = \mathrm{encode}(x)$
2. Decode **derived** output: $y = \mathrm{decode}(z)$

Image-to-Image Networks

Instead of trying to reconstruct the original input:

1. Encode input: $z = \text{encode}(x)$
2. Decode **derived** output: $y = \text{decode}(z)$

Example: Image Segmentation

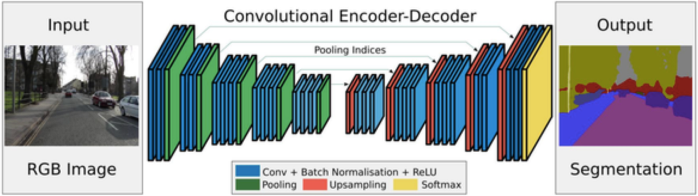
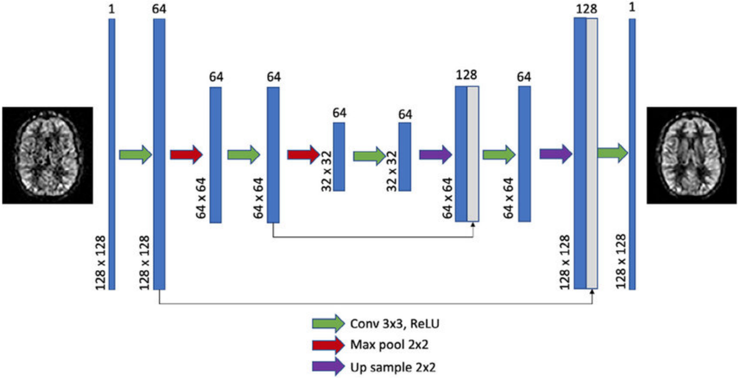


Image Denoising

Learn mapping from noisy inputs → clean outputs



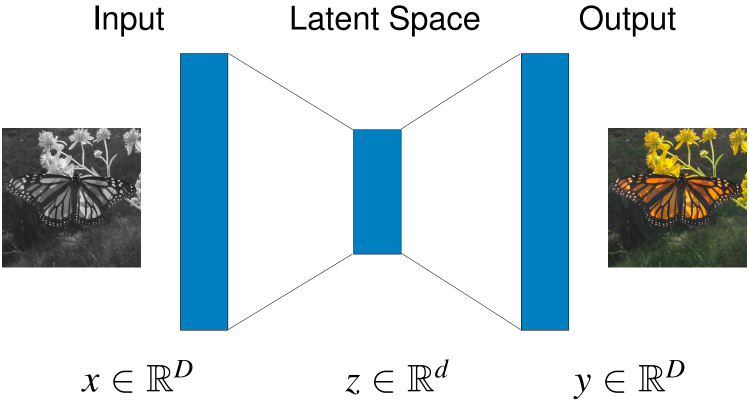
Hales et al., JMRI 2020

Image Super-resolution

Learn mapping from low-res inputs $\rightarrow$ hi-res outputs

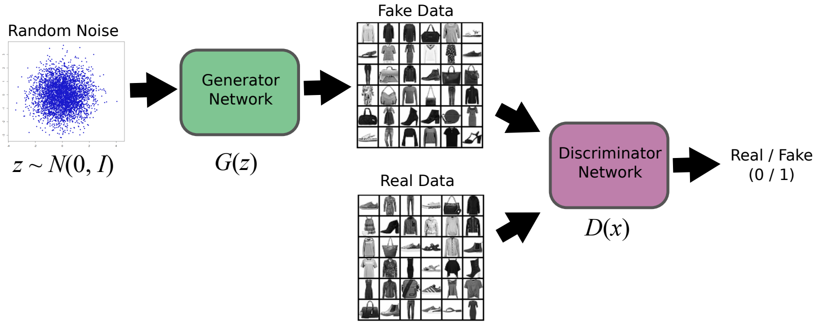
From: this webpage

Image Colorization



Generative Adversarial Networks (GANs)

Generative Adversarial Network



GAN Game Theory

GAN training is framed as a competition where:

1. Discriminator is trying to **maximize** its reward
2. Generator is trying to **minimize** it

$$\min_G \max_D V(D, G)$$

$$V(D, G) = E_{x \sim p(x)} \left[\log D(x)\right] + E_{z \sim N(0,I)} \left[\log(1 - D(G(z))\right]$$

GAN Training Algorithm

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 $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ 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 $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ 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.

Original GAN Faces (2014)

Goodfellow et al., NeurIPS 2014