Generative Adversarial Networks

Ian J. Goodfellow , Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio

Presentation by: Alan Zheng

Supervised vs. Unsupervised

- Supervised
  - Given labels
  - Classification
  - Learns to map a function y'=f(x), given labeled data y
- Unsupervised
  - Model left to figure out the underlying structure of the data
  - Clustering
  - Generative models
  - Learns the intrinsic distribution function of the input data p(x) (or p(x,y) if there are multiple targets/classes in the dataset), allowing them to generate both synthetic inputs x' and outputs/targets y', typically given some hidden parameters

Previous Work

- Models that provided a parametric specification of a probability distribution function
  - Deep Boltzmann machines
- Require Markov chains
- Discriminative models have several key limitations
  - Can't model p(x), i.e. the probability of seeing a certain image
  - Thus, can't sample from p(x), i.e. can't generate new images

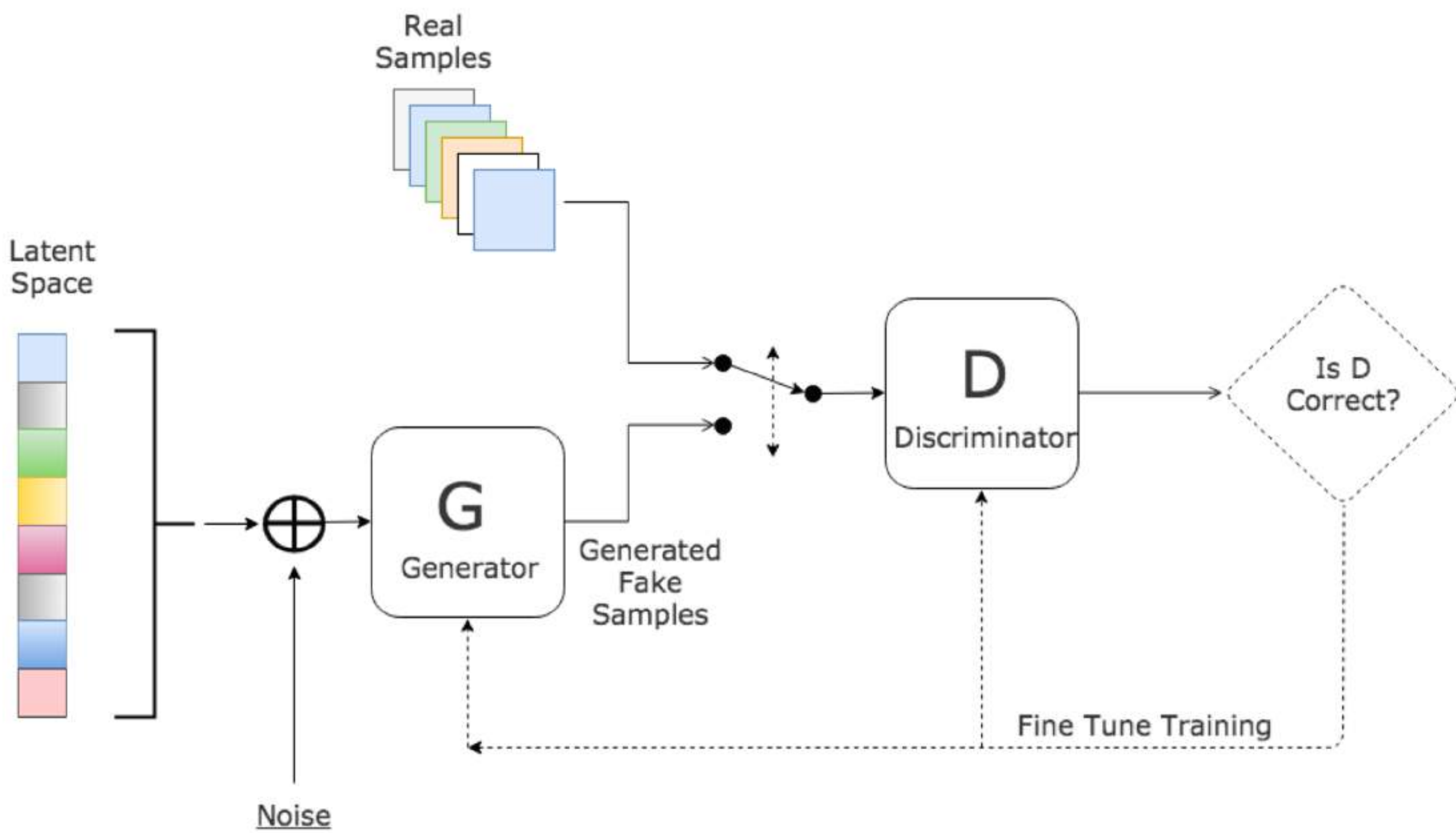
GANs

- Generative
  - Learn a generative model
- Adversarial
  - Trained in an adversarial setting
- Networks
  - Uses deep neural networks

Architecture

- Uses 2 neural networks
  - Generator
    - Takes in random noise as input (latent space)
    - Generates fake images
    - Needs to learn how to create data in such a way that the Discriminator isn't able to distinguish it as fake anymore
    - Human art counterfeiter
  - Discriminator
    - Tries to distinguish between real images and generated images
    - Art expert who tries to detect works as truthful or fraud
- The competition is what makes them both improve

Generative Adversarial Network



Math

- Generator: G(z, θ1)
  - Maps random input noise to desired data space x
  - Tries to mimic x = G(z)
  - The loss maximizes D(G(z))
- Discriminator: D(x, θ2)
  - Outputs probability that x is from the real dataset in (0,1)
  - The loss maximizes D(x) and minimizes D(G(z))
- Log of the probability is used in loss functions to heavily penalize confident errors
- Ideally, the networks reach a Nash equilibrium where neither can improve anymore
  - $P_{data}(x) = P_{gen}(x) \ \forall x$
    $\mathrm{D}(x) = \frac{1}{2} \ \forall x$

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

$$\nabla \frac{1}{m} \sum_{i=1}^{m} [\log D(x_i) + \log(1 - D(G(z_i))]$$

$$\nabla \frac{1}{m} \sum_{i=1}^{m} \log(1 - D(G(z_i))$$

Training

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

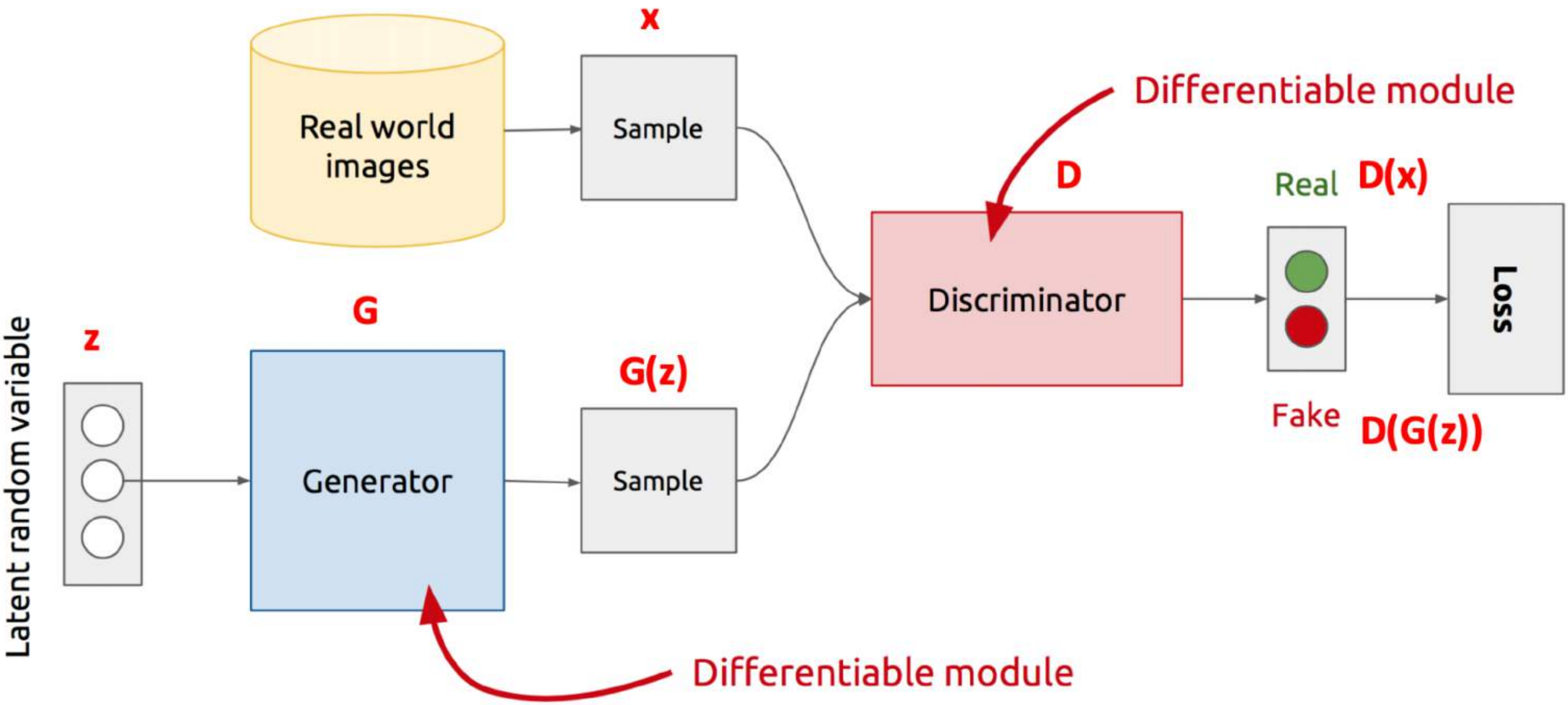
  • Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.

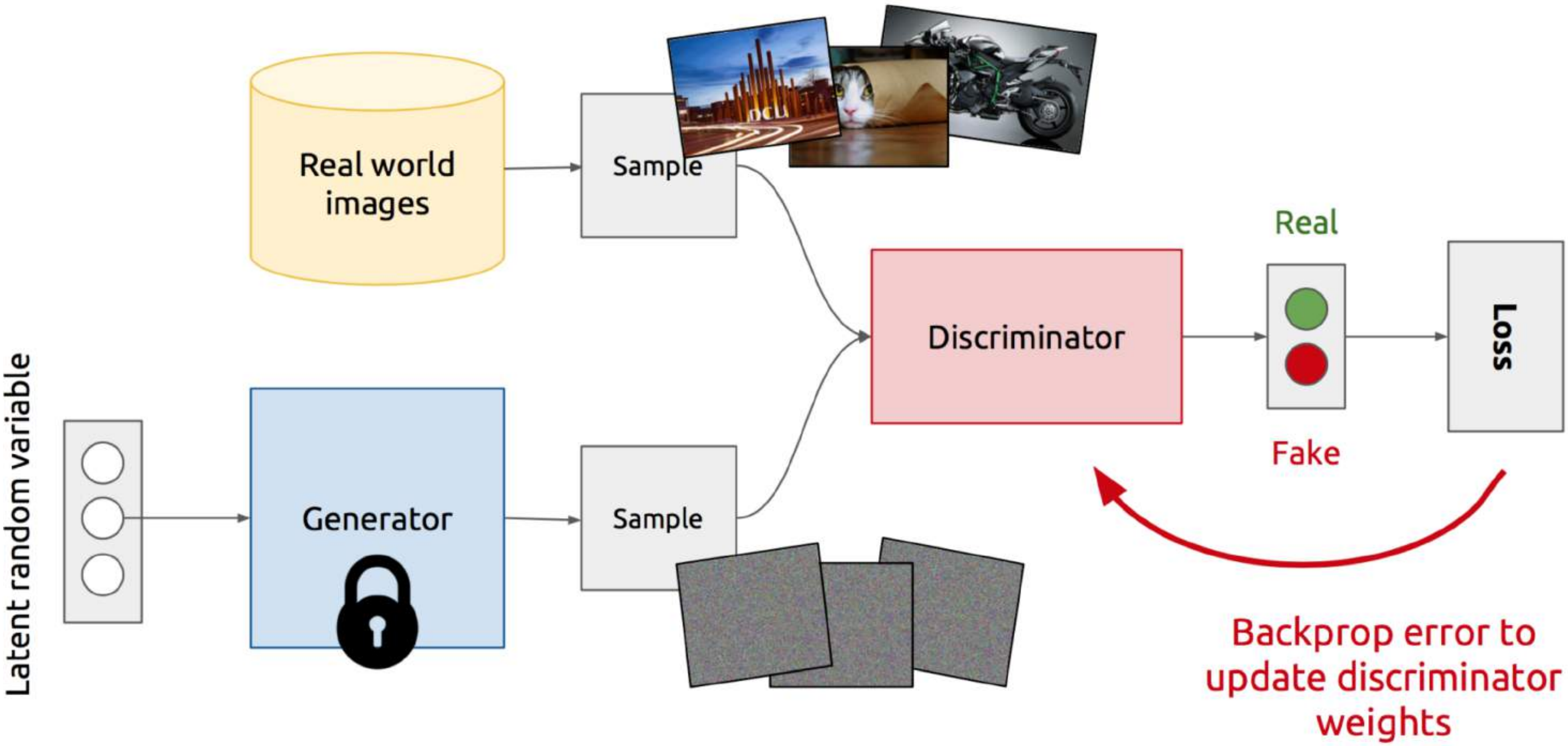
  • Update the generator by descending its stochastic gradient:

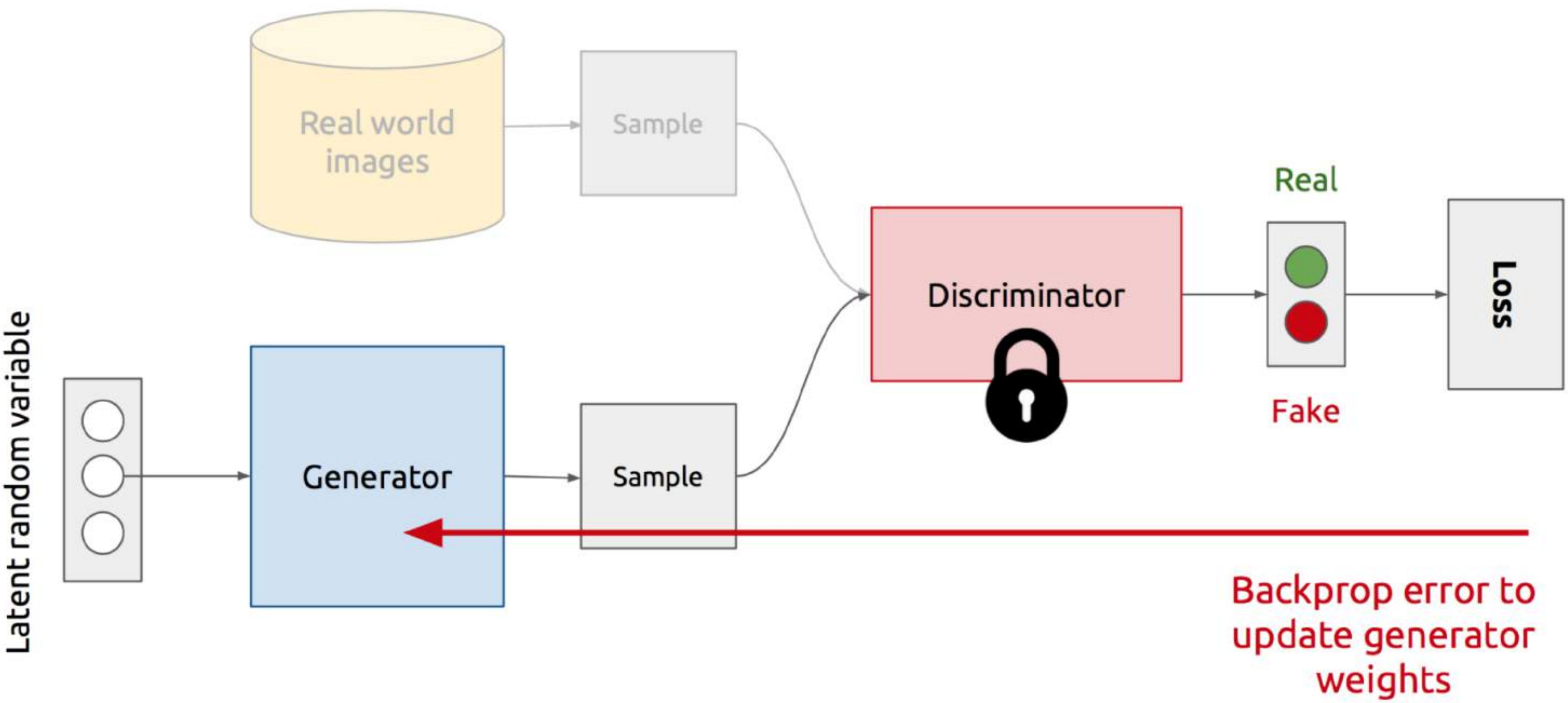
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

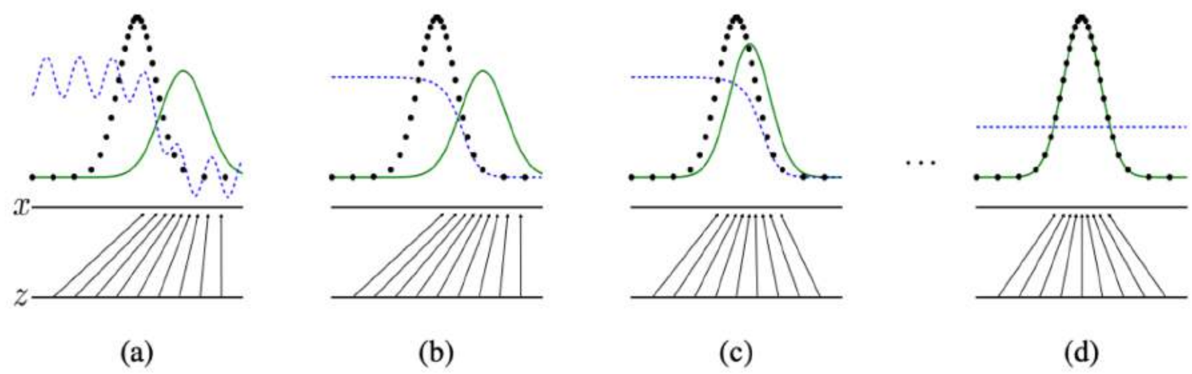
The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.


x
Real world images
Sample
Differentiable module
D
Real
D(x)
Discriminator
Loss
Latent random variable
z
G
Generator
G(z)
Sample
Fake
D(G(z))
Differentiable module

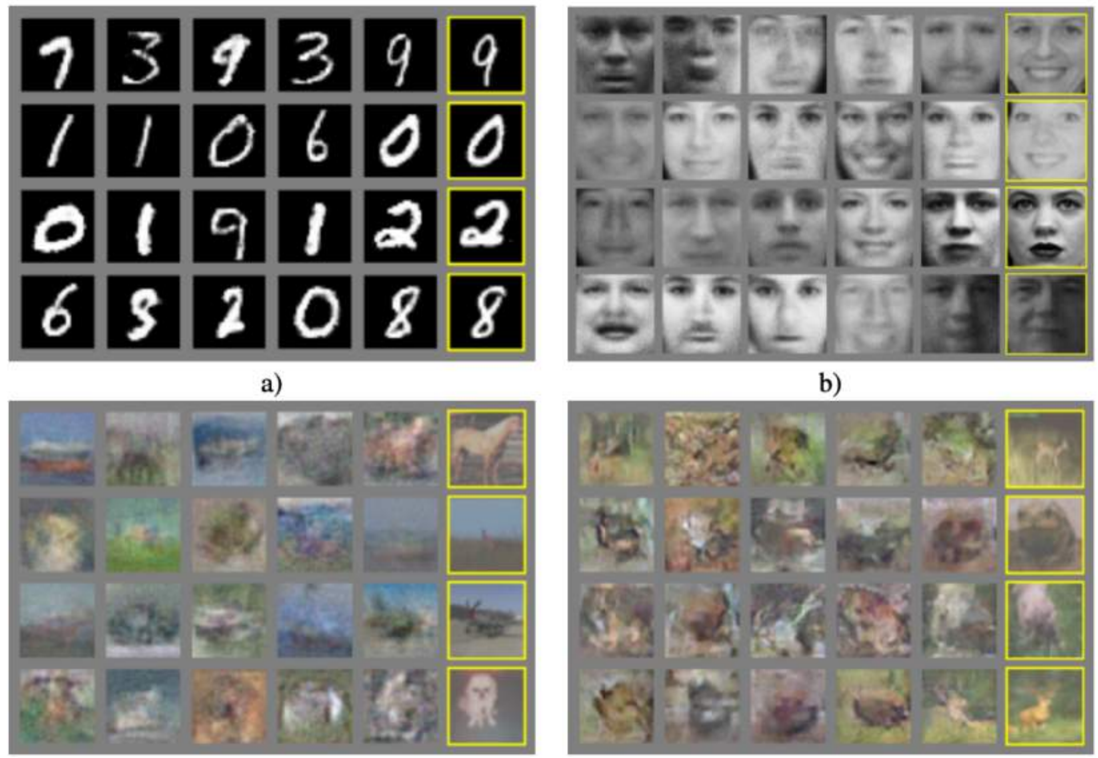

Latent random variable
Real world images
Sample
Generator
Sample
Discriminator
Real
Fake
Loss
Backprop error to update discriminator weights


Real world images
Sample
Real
Discriminator
Loss
Fake
Latent random variable
Generator
Sample
Backprop error to update generator weights

Theoretical Results



Results



a) b) c) d)

	Deep directed graphical models	Deep undirected graphical models	Generative autoencoders	Adversarial models
Training	Inference needed during training.	Inference needed during training. MCMC needed to approximate partition function gradient.	Enforced tradeoff between mixing and power of reconstruction generation	Synchronizing the discriminator with the generator. Helvetica.
Inference	Learned approximate inference	Variational inference	MCMC-based inference	Learned approximate inference
Sampling	No difficulties	Requires Markov chain	Requires Markov chain	No difficulties
Evaluating $p(x)$	Intractable, may be approximated with AIS	Intractable, may be approximated with AIS	Not explicitly represented, may be approximated with Parzen density estimation	Not explicitly represented, may be approximated with Parzen density estimation
Model design	Models need to be designed to work with the desired inference scheme — some inference schemes support similar model families as GANs	Careful design needed to ensure multiple properties	Any differentiable function is theoretically permitted	Any differentiable function is theoretically permitted