Generative Adversarial Networks

Benjamin Wilson and Muhammad Osama Sakhi

September 2, 2019

Georgia Institute of Technology

Table of contents

1. Motivation

2. Previous Work

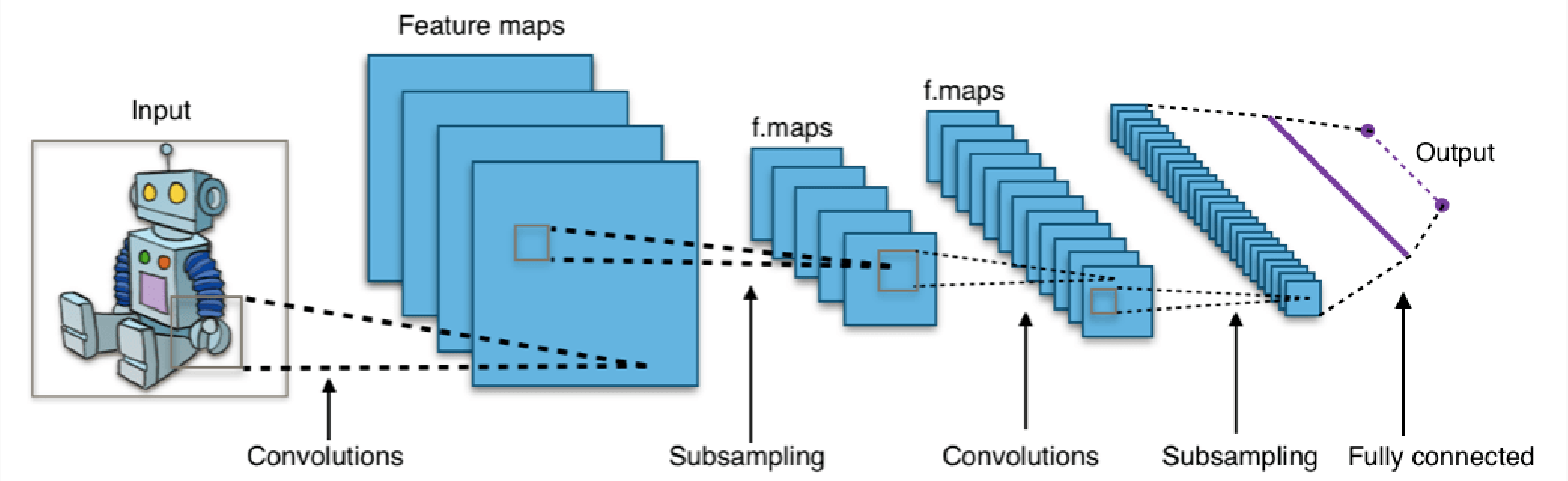
3. Generative Adversarial Networks

4. Conclusion

Motivation

Discriminative Machine Learning Setup

- Estimate a conditional probability distribution function: $P(Y|X)$



- How do we model $P(X)$?

Why Generative Models?

- Important exercise of manipulation high-dimensional probability distributions
- Semi-Supervised Learning
- Generating realistic samples from some distribution
- Plenty of other examples; however, many came after the original paper

Motivation

Previous Generative Models

- We'll focus on generative models that work via **maximum likelihood**:

$$\begin{aligned}\boldsymbol{\theta}^* &= \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{m} P_{model}(\mathbf{x}_i; \theta) \\ &= \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{m} \log P_{model}(\mathbf{x}_i; \theta)\end{aligned}$$

Explicit Density Models

- Define an explicit density function, $P_{model}(\boldsymbol{x}; \theta)$.
- Fully visible belief networks (FBVNs):
- $\boldsymbol{x} \in \mathbb{R}^n$

$$P_{model}(\boldsymbol{x}) = \prod_{i=1}^{n} P_{model}(x_i | x_1, ...x_{i-1})$$

- Pros: Foundation of strong generative models (ex. WaveNet)
- Cons: Samples must be generated one at a time.

Explicit models requiring approximation

- Variational Methods: $\mathcal{L}(x;\theta) \leq \log P_{model}(x;\theta)$
- **Variational Autoencoder (VAE)**
- Pros: Strong control of latent space structure
- Cons: Not asymptotically consistent unless approximate posterior is perfect, low quality sampling

Explicit models requiring approximation

Figure 11: Samples drawn from a VAE trained on the CIFAR-10 dataset. Figure reproduced from Kingma *et al.* (2016).

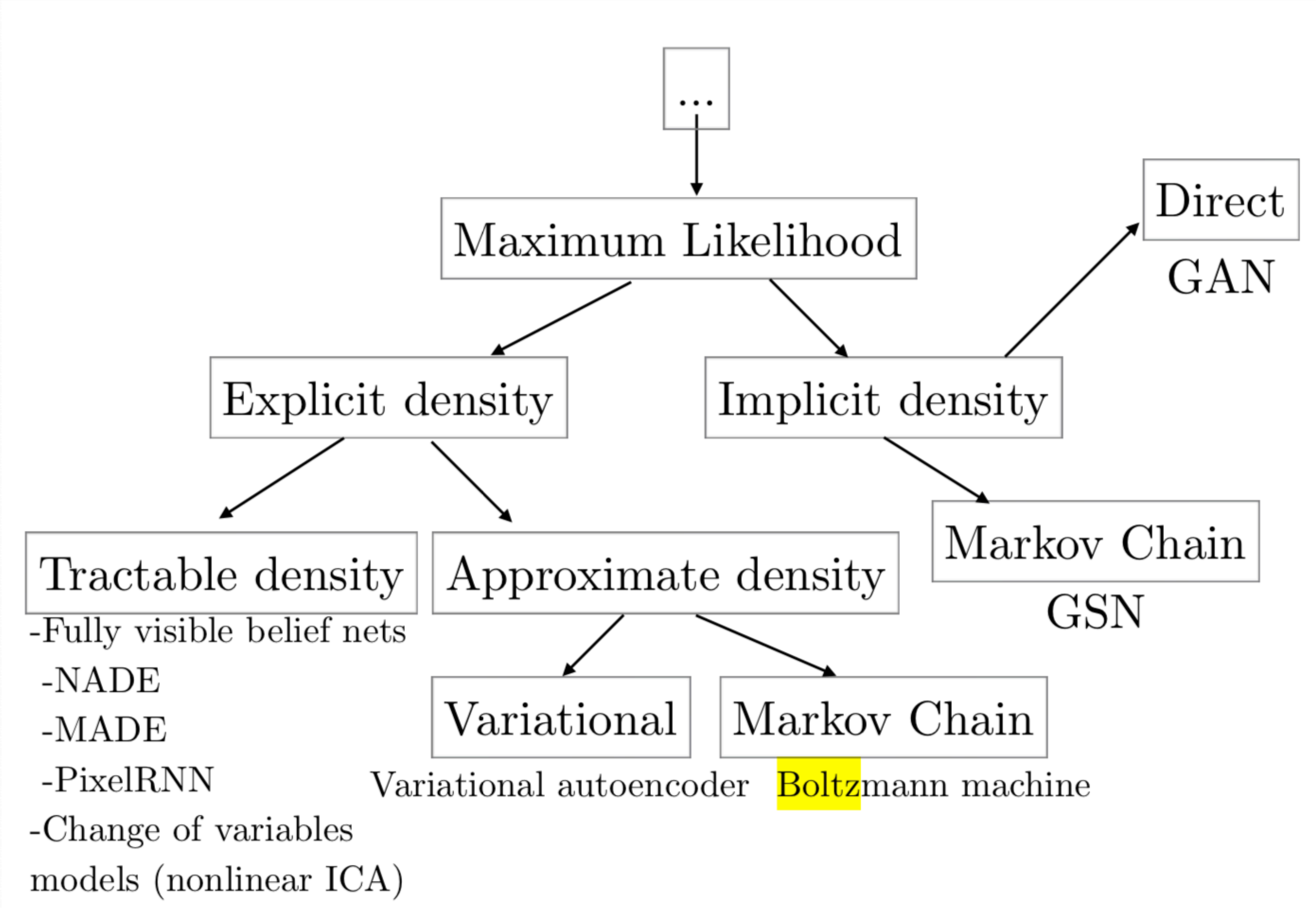
Explicit models requiring approximation

- Boltzmann Machines
- $P(\boldsymbol{x}) = \frac{1}{Z} \exp\left(-E(x, z)\right)$
- $Z = \sum_x \sum_z \exp\left(-E(x, z)\right)$
- Pros: Designed with regard to physical processes
- Cons: Markov chain approximation techniques have not scaled to ImageNet like problems

Implicit density models

- Generative Adversarial Networks (GANs)

Taxonomy of Generative Models

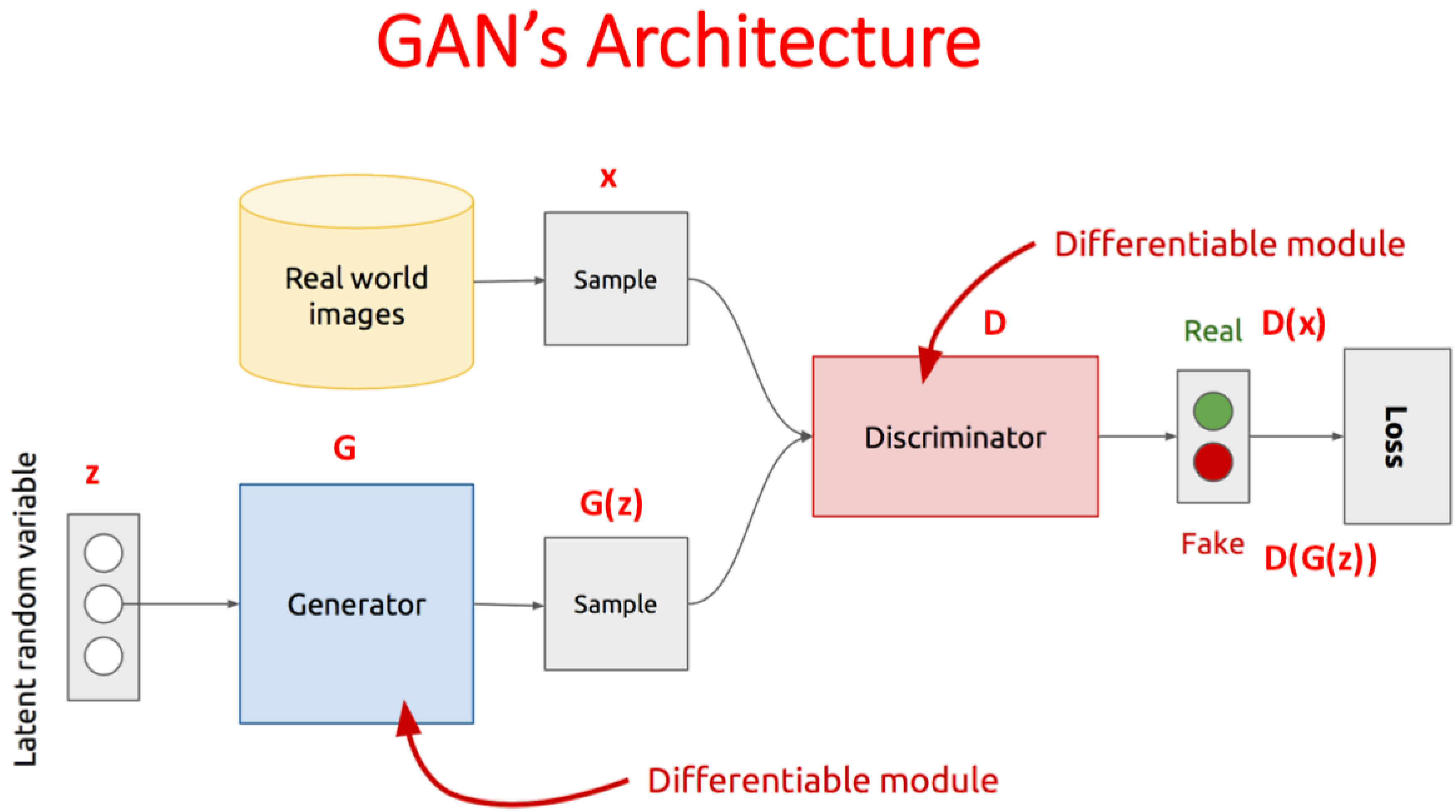


Generative Adversarial Networks

Model Architecture

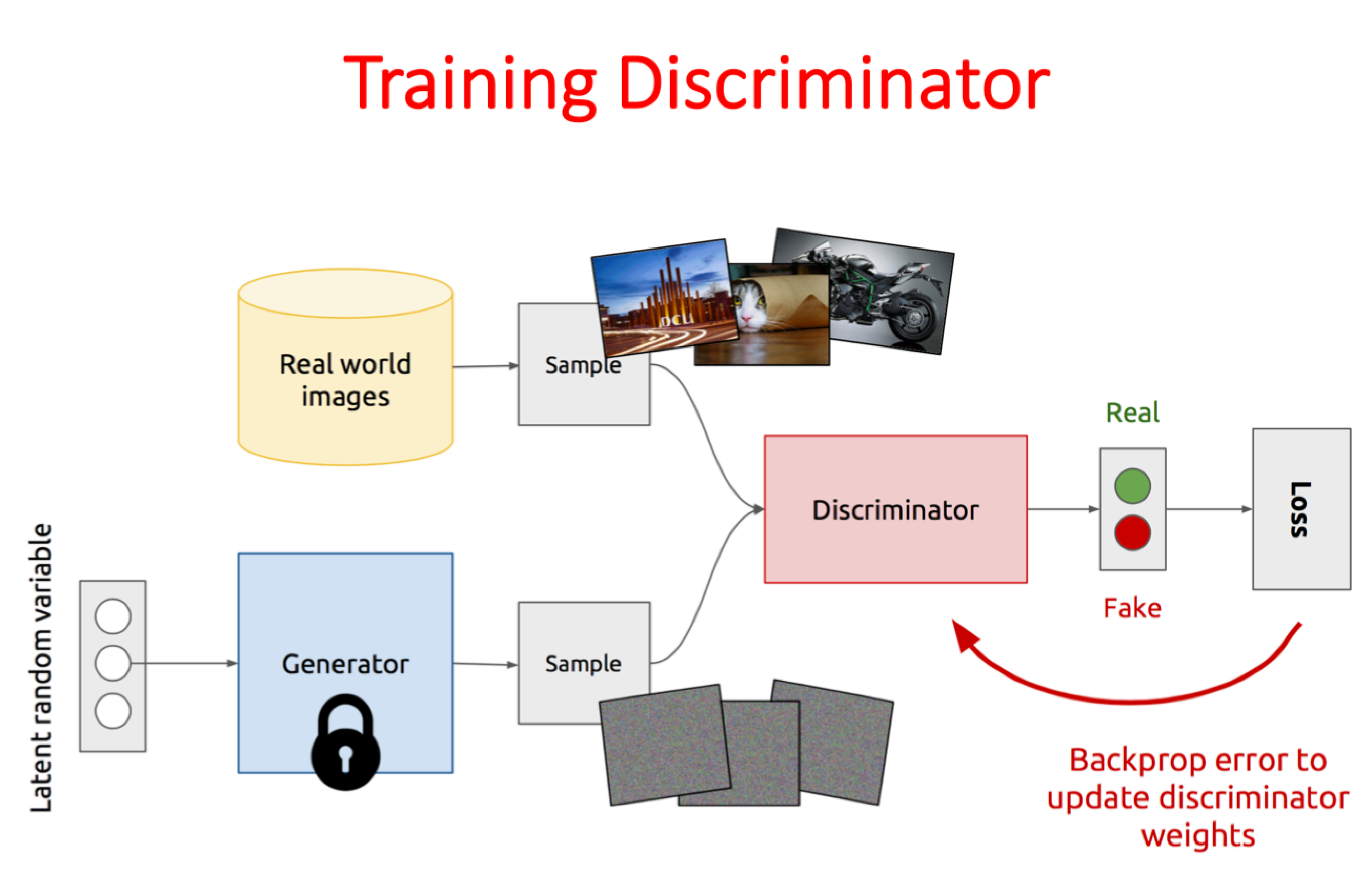
- Setup a game between a **generator** and a **discriminator**.
- The **generator** produces approximations of samples drawn from P_{data}, which we call P_G.
- **Discriminator** is given $x \sim P_G$ and $x \sim P_{data}$. For both samples, it tries to determine whether they were sampled from P_{data}.

Model Architecture

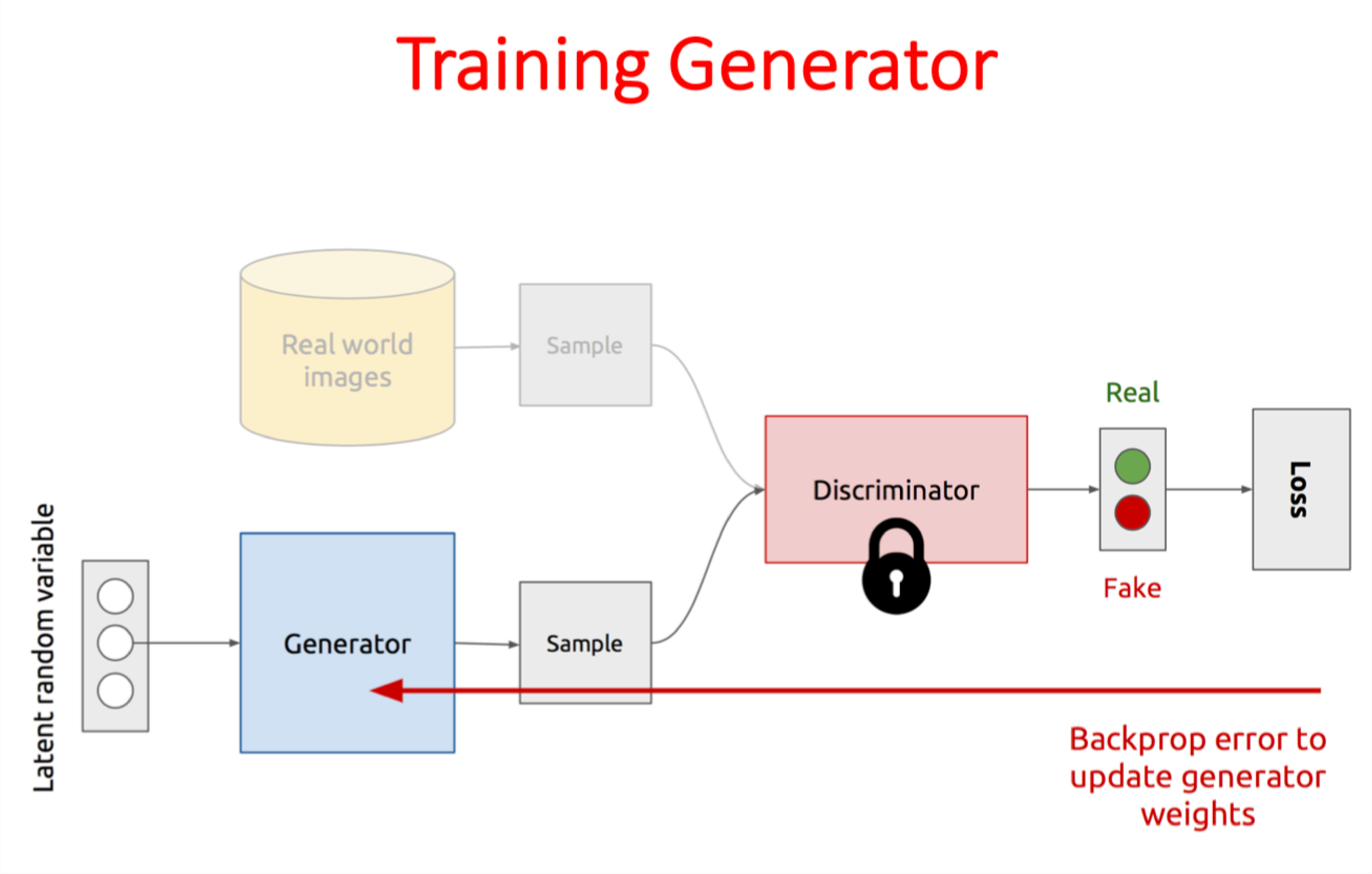


- **Z** is some random noise (Gaussian/Uniform).
- **Z** can be thought as the latent representation of the image.

Model Architecture



Model Architecture



Model Objective

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

- Authors show both convergence and optimality under certain assumptions using the above objective.

Model Objective

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:

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

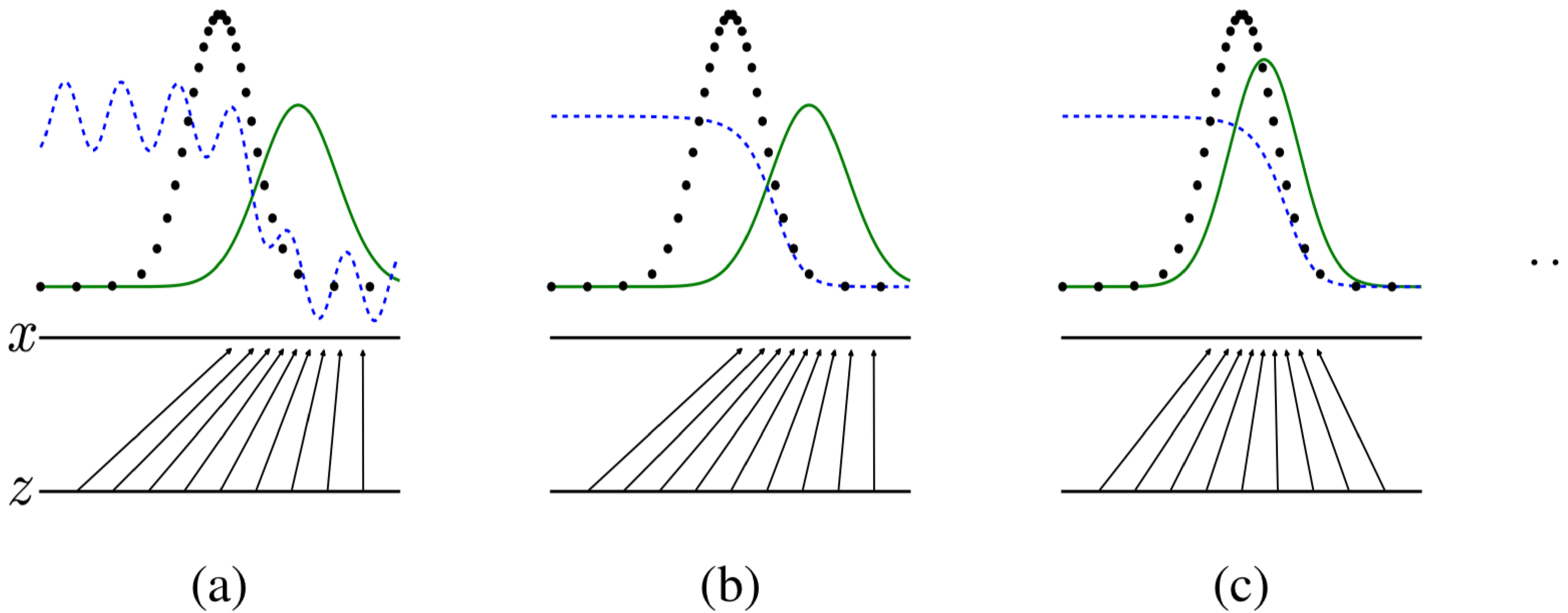
end for

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

Pros

- **Context and Differentiating From Other Work:** The authors explain how their approach is a departure from existing generative approaches since they do not use a DNN to obtain the parameters that maximize the likelihood of the data and then use those parameters to sample from a distribution. Rather, they explain how the networks themselves, G and D, are sufficient for producing results that, although are not necessarily maximizing the likelihood, are still generating samples that trick even the most optimal discriminator.
- **Illustrations:** Clear explanations and illustrations to show how training an optimal discriminator, then an optimal generator eventually results in both G and D being optimized (Ex: Fig 1).

Pros: Figure 1



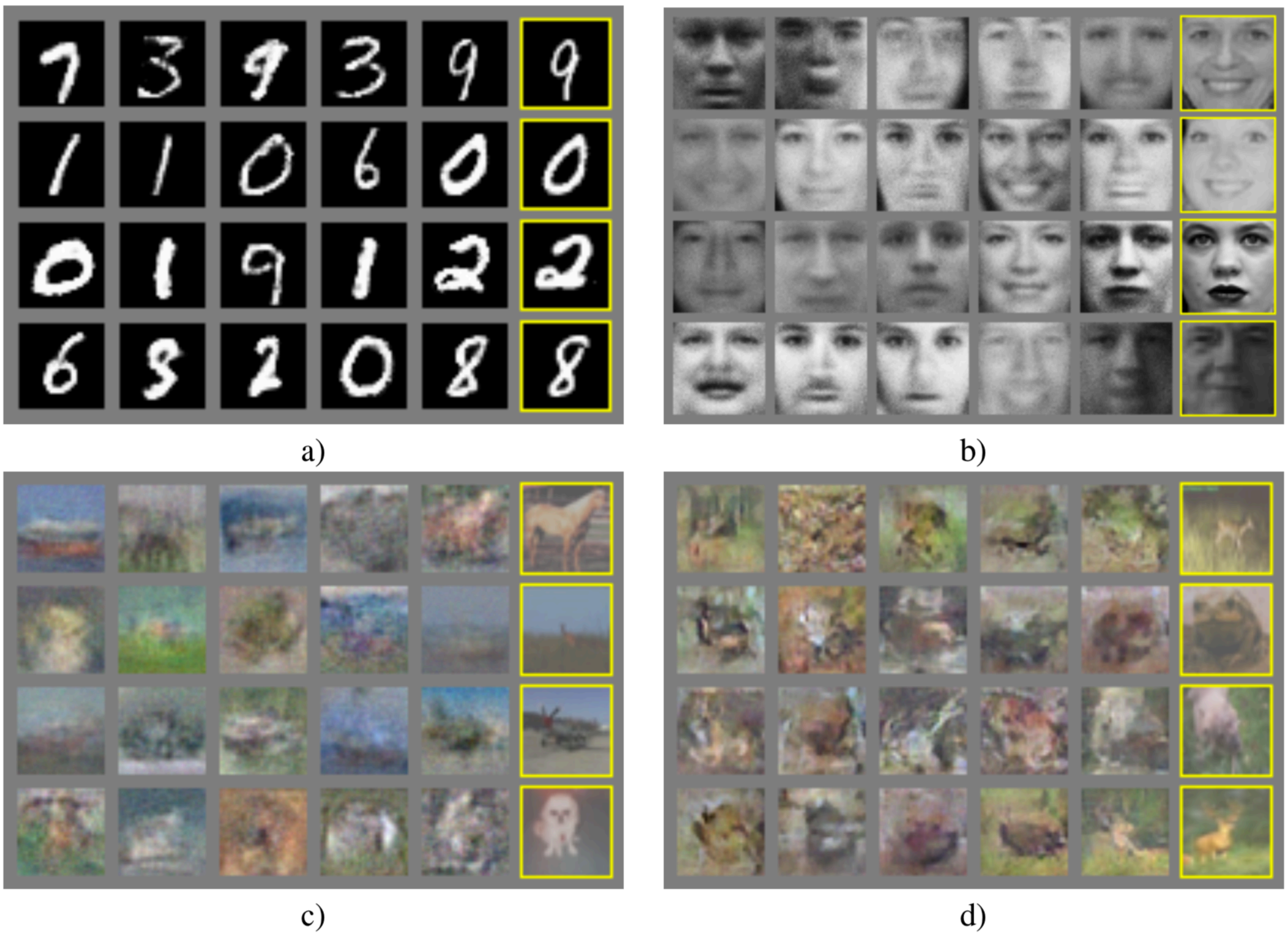
Pros Continued

- **Sampling:** The only time sampling is needed here is to produce the vector $\mathbf{z}$, which is the only thing the generator G needs to produce a synthetic sample $\mathbf{x}$.
- **Practical Guidelines:** The paper specifies how training one network too much without switching to training to the other network can prevent the model from having sufficient diversity to model p_g.

Pros Continued

- **Limitations of GANs and Evaluation Metric:** The authors concede that adversarial networks represent only a subset of p_g distributions, yet still show great performance in practice. They also concede that the *Gaussian Parzen window* approach has limitations (i.e high variance, poor performance in high-dimensional space), but that it at least shows that GANs are competitive in the generative model space.
- **Admitting Pros & Cons:** The authors themselves highlight the pros and cons of their model: GANs are computationally more performant that other generative approaches, no Markov chains needed, no inference step is needed during training

Pros: Figure 2



Cons

- Instability during training with heavy reliance on hyperparameter selection.

Cons

- Instability during training with heavy reliance on hyperparameter selection.
- Estimation of $P_g(x)$ is required to evaluate density function.

Cons

- Instability during training with heavy reliance on hyperparameter selection.
- Estimation of $P_g(x)$ is required to evaluate density function.
- Relies on high variance metric (Parzen log-likelihood estimates) and qualitative samples.

Cons

- Instability during training with heavy reliance on hyperparameter selection.
- Estimation of $P_g(x)$ is required to evaluate density function.
- Relies on high variance metric (Parzen log-likelihood estimates) and qualitative samples.
- May not reach optimal solution

Cons

- Instability during training with heavy reliance on hyperparameter selection.
- Estimation of $P_g(x)$ is required to evaluate density function.
- Relies on high variance metric (Parzen log-likelihood estimates) and qualitative samples.
- May not reach optimal solution
- Mode collapse

Conclusion

Conclusion

Questions?

References i

1. https://media.nips.cc/Conferences/2016/Slides/6202-Slides.pdf
2. https://arxiv.org/pdf/1701.00160.pdf
3. http://slazebni.cs.illinois.edu/spring17/lec11_gan.pdf