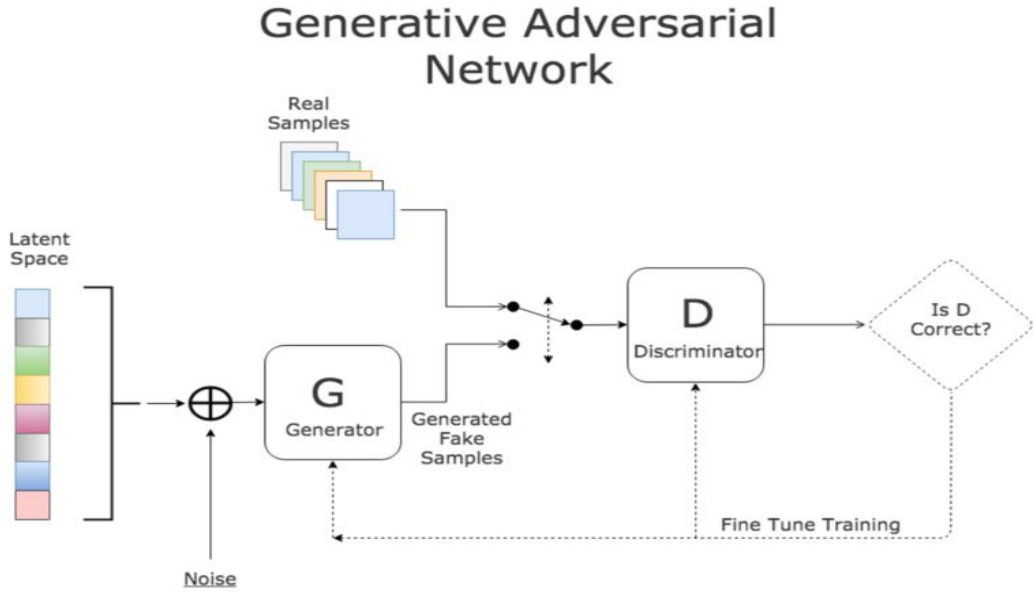
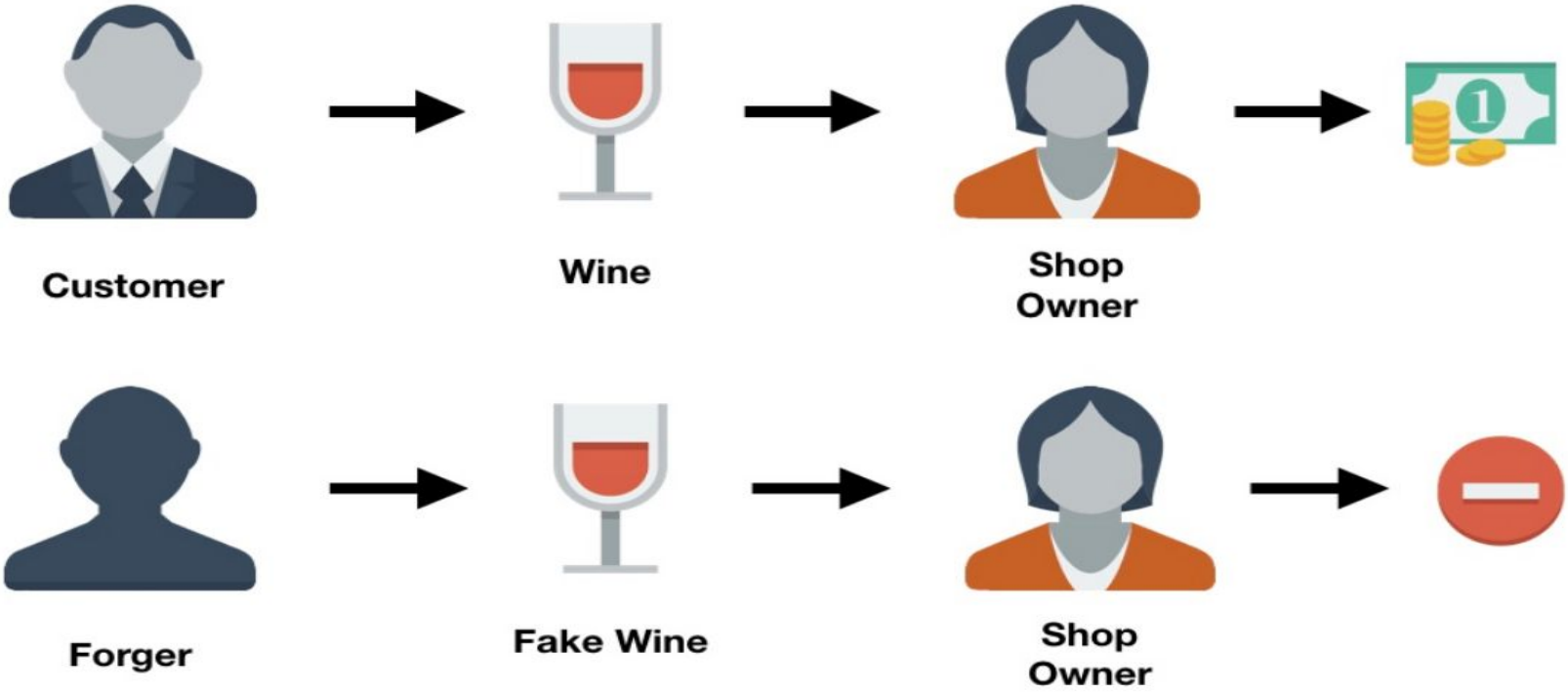
Generative Adversarial Nets (GANs)



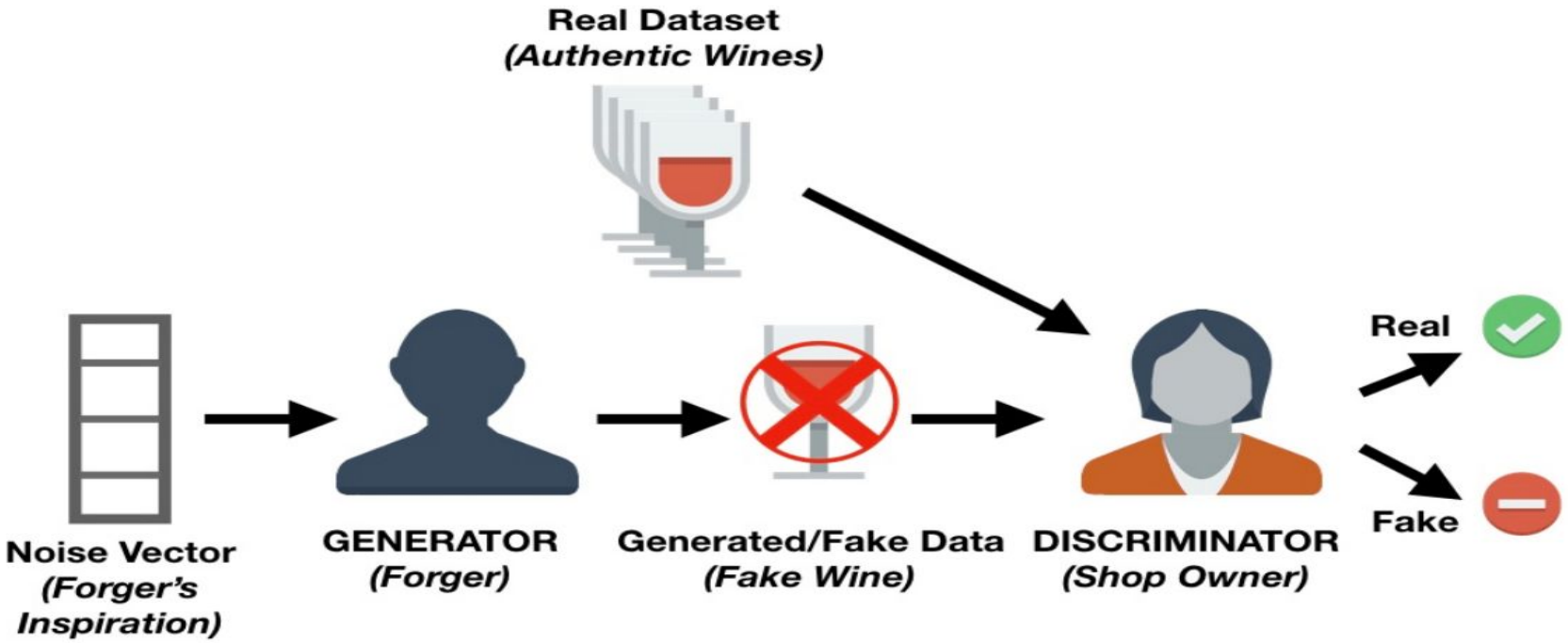
Reading Group - 03

E/15/016 S.Anojan e15016@eng.pdn.ac.lk
E/15/351 S.Thakshajini e15351@eng.pdn.ac.lk
E/15/348 S.Suhail suhailsajahan@eng.pdn.ac.lk
E/15/171 R.Kapilrajh e15171@eng.pdn.ac.lk

Introduction to GANs

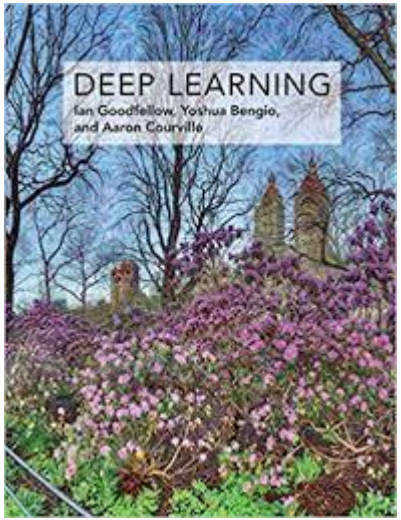


Architecture of GANs



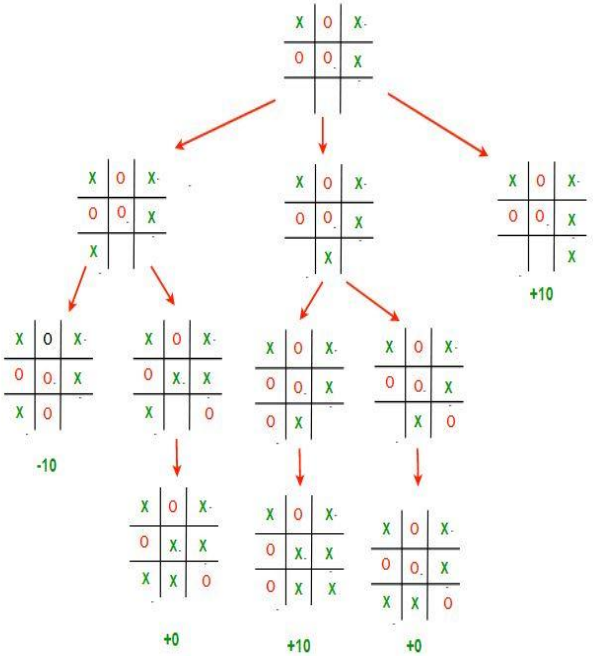
Background

- Authors: Ian J.Goodfellow, Jean-Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio
- Ian J.Goodfellow's book entitled "Deep Learning" is a very good reading material to learn about a broad range of topics related to deep learning.
- They proposed a new framework to estimate generative models using an adversarial process.
- They trained two models simultaneously.
  - Generative model (G)
  - Discriminative model (D) .
- The framework is relative to a minimax two-player game.



Mini-Max Two Player Games

- A backtracking algorithm to find the optimal move for a player.
- Used in decision making and game theory
- Used in two player games such as tic-tac-toe, backgammon, manchala and chess.
- Two players are:
    1. Maximizer - Tries to get the highest score possible.
    2. Minimizer - Tries to get the lowest score possible.
- Every board state has a value associated with it.



Key Points

- Models G and D are defined by multilayer perceptrons and the entire system can be trained with backpropagation.
- No need for any Markov chains or unrolled approximate inference networks during either training or generating samples.
- The experiments demonstrate the potential of the framework through qualitative and quantitative evaluation of the generated samples.
- This framework can yield specific training algorithms for many kinds of model and optimization algorithms.
- G generates samples by passing random noise through a multilayer perceptron and D is also a multilayer perceptron.

Related Work

- Recent deep generative models mainly focuses on providing a parametric specification of a probability distribution function.
- Deep Boltzmann Machine (DBM) is the most successful model among the models which can be trained by maximizing the log likelihood.
- Generative stochastic networks are an example of generative machine that can be trained with exact backpropagation rather than the numerous approximations required for Boltzmann machines.
- Backpropagation of derivatives through generative processes using this observation.

$$\lim_{\sigma \to 0} \nabla_{\boldsymbol{x}} \mathbb{E}_{\epsilon \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I})} f(\boldsymbol{x} + \epsilon) = \nabla_{\boldsymbol{x}} f(\boldsymbol{x}).$$

Related Work

Deep Boltzmann Machines

- Contains input layer and hidden layer.
- Unsupervised Deep Learning Model.
- Generative Deep Learning Model.
- Connections are undirected.

vs

Normal Neural Networks

- Contains input layer, output layer and hidden layers.
- Supervised Deep Learning Model.
- Deterministic Deep Learning Model.
- Connections are directed.

Related Work

- Researchers have used stochastic backpropagation to train Variational AutoEncoders (VAEs).
- VAEs vs GANs
  - GANs require differentiation through the visible units.
  - VAEs require differentiation through the hidden units.
- Previous researches also used a discriminative criterion to train a model.
- Noise-Contrastive Estimation (NCE) trains a generative model by learning the weights which make the model useful for discriminating data from a fixed noise distribution.

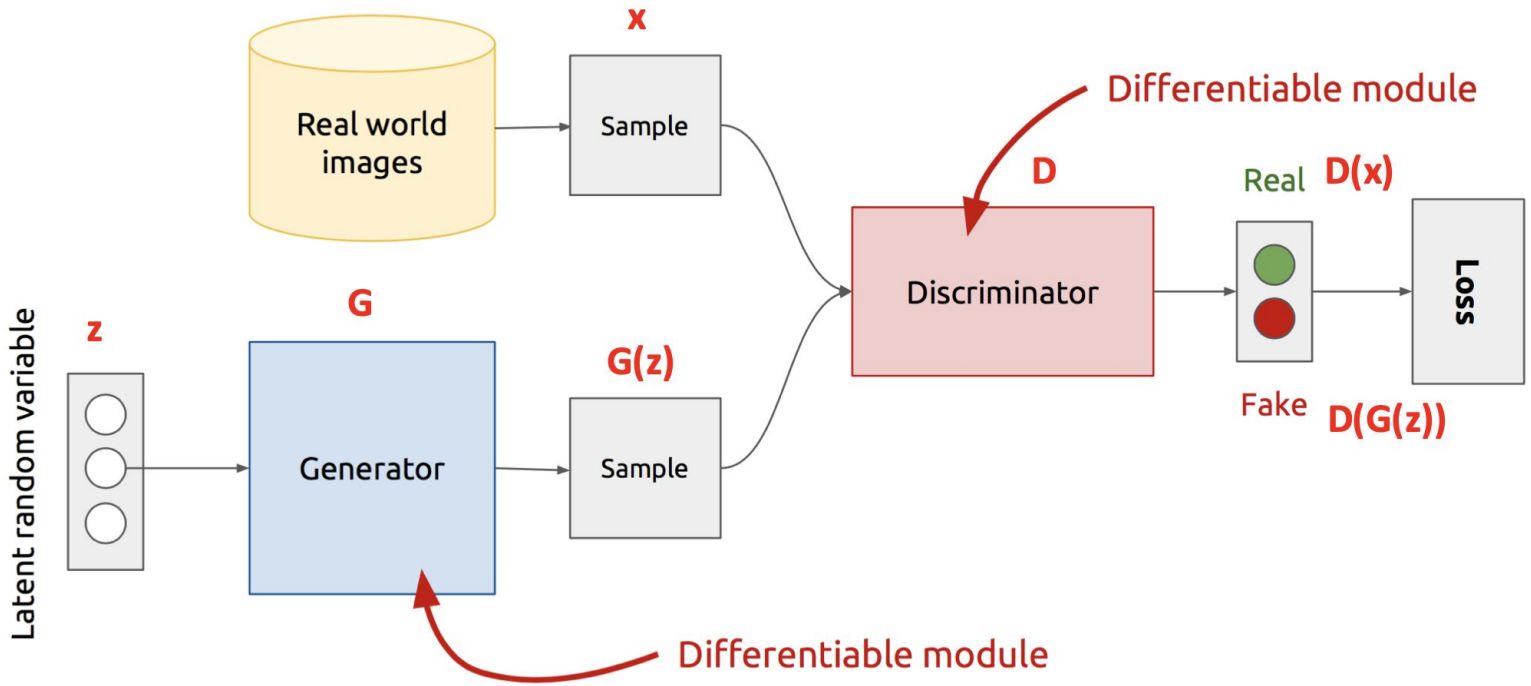
Related Work

- Some of the previous researches has used the competition between two neural networks.
  - The most relevant work - Predictability Minimization

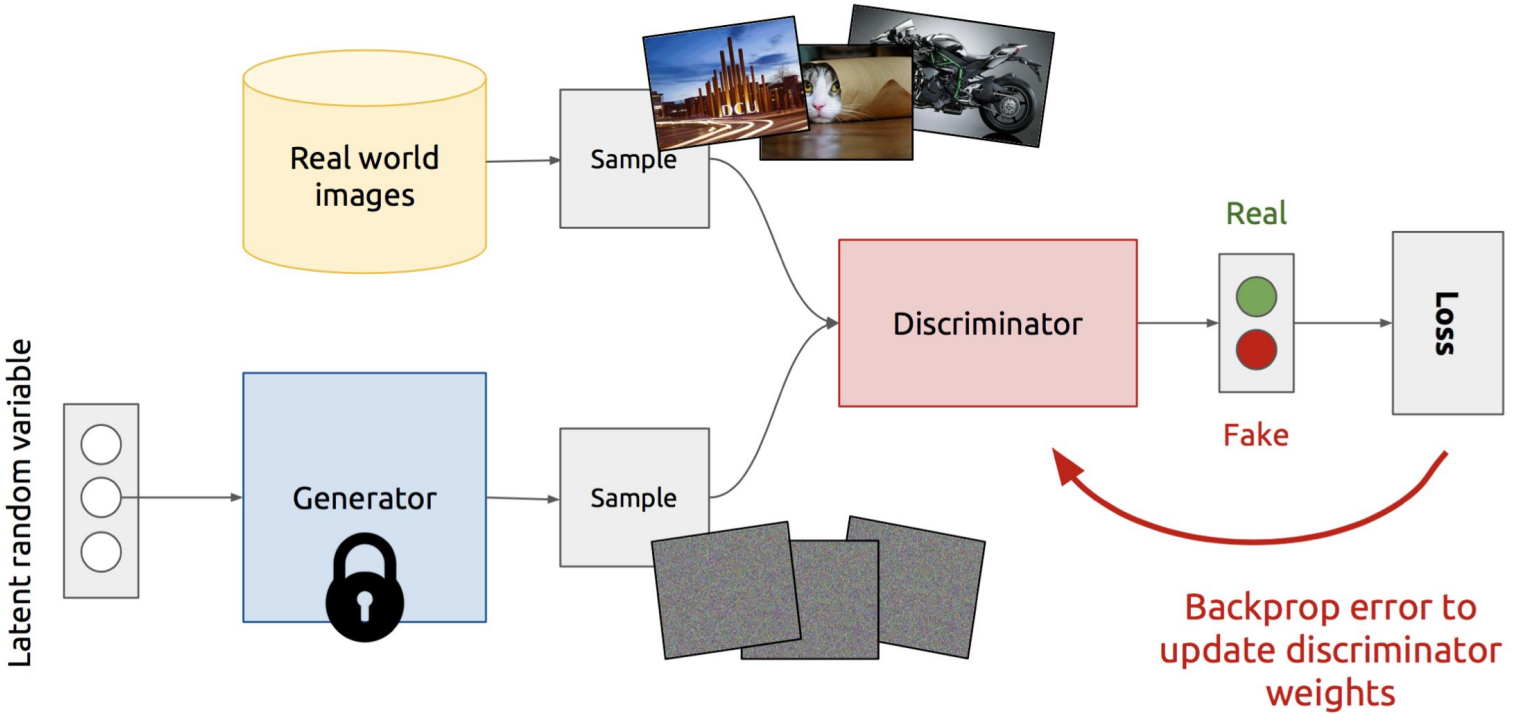
- GANs vs Predictability Minimization
  - The competition and the sole training criterion.
  - The nature of the competition.
  - The specification of the learning process.

- The confusion between GANs and Adversarial Examples

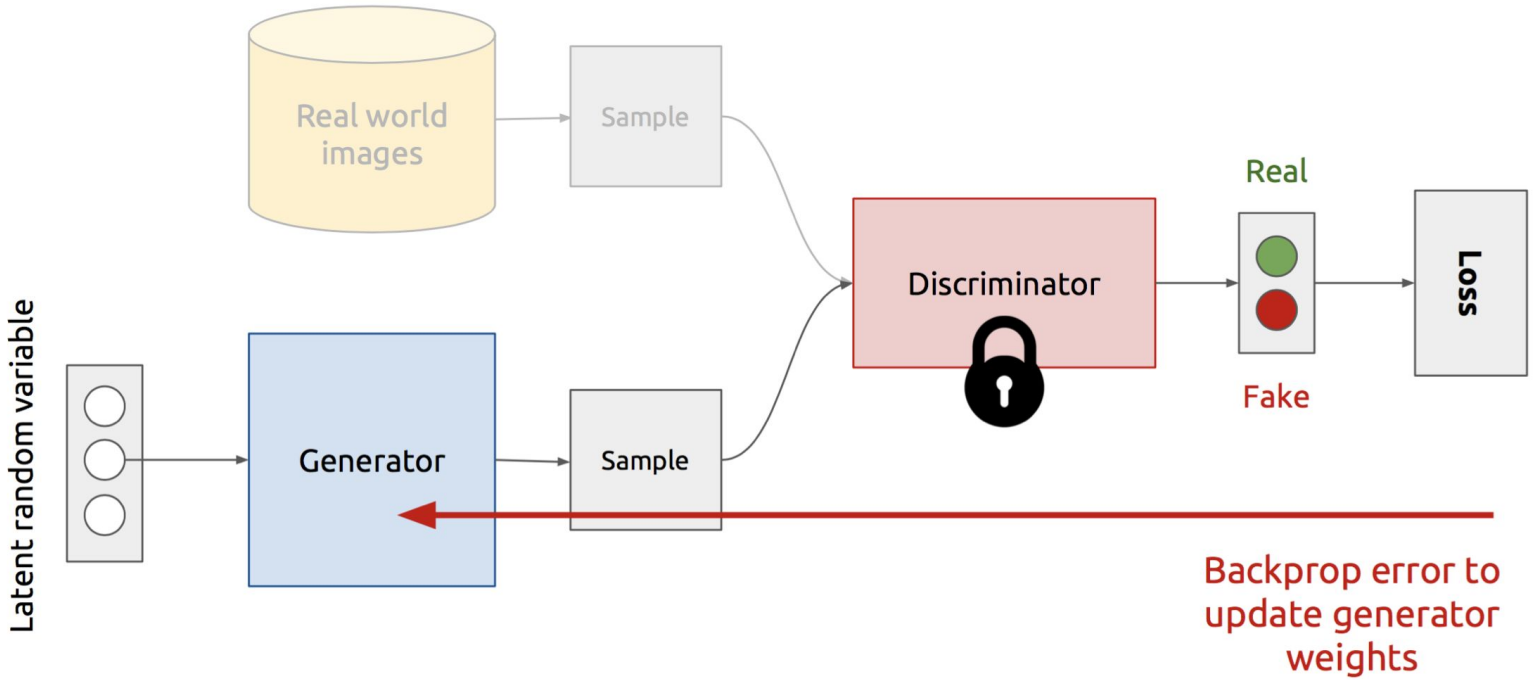
Classic GAN Framework



Training Discriminator



Training Generator



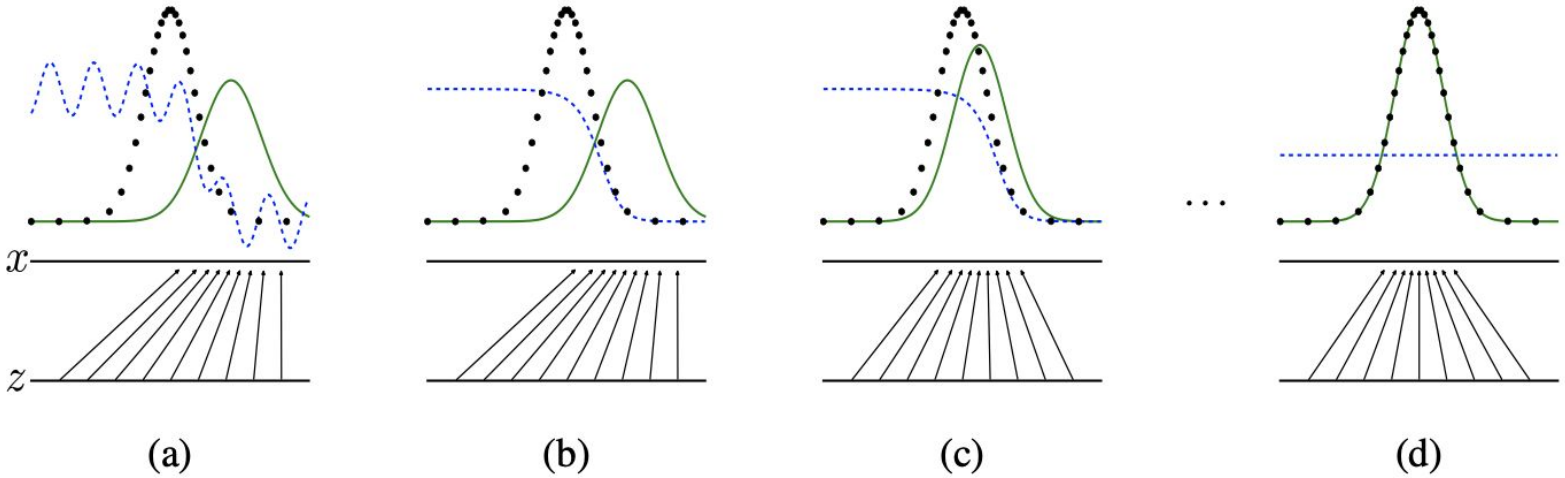
GAN's Formulation

Equation (1) =>

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

- It is formulated as a minimax game, where:
  - The Discriminator is trying to maximize its reward V(D, G)
  - The Generator is trying to minimize Discriminator's reward

Theoretical Results



Discriminative Distribution - Blue dashed line
Generative Distribution (P_g) - Green solid line
Data Generating Distribution (P_{data}) - Black dotted line
Global optimum reached in (d) where $P_g = P_{data}$

Theoretical Results

- **Algorithm I**
  - 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 step to do
    - Sample minibatch of m noise samples {z(1), . . . , z(m)} from noise prior $p_g(z)$.
    - Sample minibatch of m examples {x(1), . . . , x(m)} from data generating distribution $p_{data}(x)$.
    - 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 {z(1), . . . , z(m)} from noise prior $p_g(z)$.
    - Update the generator by descending its stochastic gradient:
- end for

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

Theoretical Results

- **Global Optimality of $p_g = p_{data}$**
  - Considering the optimal discriminator D for any given generator G.
  - Proposition 1 => For G fixed, the optimal discriminator

Equation (2) => $$D_G^*(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})}$$

  - Proof
  - The training criterion for the discriminator D, given any generator G, is to maximize the quantity V (G, D)

Equation (3) => $$V(G,D) = \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x}))dx + \int_{z} p_{\boldsymbol{z}}(\boldsymbol{z}) \log(1 - D(g(\boldsymbol{z})))dz$$
$$= \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x})) + p_g(\boldsymbol{x}) \log(1 - D(\boldsymbol{x}))dx$$

- For any $(a, b) \in \mathbb{R}\times\mathbb{R} \setminus \{0, 0\}$, the function $y \rightarrow a \log(y) + b \log(1 - y)$ achieves its maximum in [0, 1] at $\frac{a}{a+b}$. The discriminator does not need to be defined outside of $\text{Supp}(p_{data}) \cup \text{Supp}(p_g)$. End of proof.

Theoretical Results

- The training objective for D can be interpreted as maximizing the log-likelihood for estimating the conditional probability P(Y = y|x), where Y indicates whether x comes from p_{data} (with y = 1) or from p_g (with y = 0). The minimax game in earlier equation can be reformulated as:

Equation (4) =>

$$
\begin{aligned}
C(G) &= \max_D V(G, D) \\
&= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}}[\log(1 - D_G^*(G(\boldsymbol{z})))] \\
&= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x} \sim p_g}[\log(1 - D_G^*(\boldsymbol{x}))] \\
&= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(\boldsymbol{x})}{P_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right] + \mathbb{E}_{\boldsymbol{x} \sim p_g}\left[\log \frac{p_g(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right]
\end{aligned}
$$

JS Divergence formula =>

$$JS(P1||P2) = \frac{1}{2} E_{x \sim P1} \ln\left(\frac{P1}{(P1 + P2)/2}\right) + \frac{1}{2} E_{x \sim P2}\left(\frac{P2}{(P1 + P2)/2}\right)$$

- Theorem 1 => The global minimum of the virtual training criterion C(G) is achieved if and only if $p_g = p_{data}$. At that point, C(G) achieves the value − log 4.

Theoretical Results

- **Proof of Theorem 1:**
  - For $p_g = p_{data}$, $D^*_G(x) = (½)$, (consider Equation (2)). Hence, by inspecting Equation (4) at $D^*_G(x) = (½)$, we find $C(G) = \log(½) + \log(½) = -\log 4$. To see that this is the best possible value of C(G), reached only for $p_g = p_{data}$, observe that

$$\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[-\log 2] + \mathbb{E}_{\boldsymbol{x}\sim p_g}[-\log 2] = -\log 4$$

  - and that by subtracting this expression from $C(G) = V(D^*_G, G)$, we obtain:

Equation (5) => $$C(G) = -\log(4) + KL\left(p_{\text{data}} \middle\| \frac{p_{\text{data}} + p_g}{2}\right) + KL\left(p_g \middle\| \frac{p_{\text{data}} + p_g}{2}\right)$$

  - where KL is the Kullback–Leibler divergence. We recognize in the previous expression the Jensen–Shannon divergence between the model's distribution and the data generating process:

Equation (6) => $$C(G) = -\log(4) + 2 \cdot JSD\left(p_{\text{data}} \| p_g\right)$$

  - Since the Jensen–Shannon divergence between two distributions is always non-negative, and zero iff they are equal, we have shown that $C^* = -\log(4)$ is the global minimum of C(G) and that the only solution is $p_g = p_{data}$, that is the generative model perfectly replicating the data distribution.

Theoretical Results

- **Convergence of Algorithm I**
  - Proposition 2 => If G and D have enough capacity, and at each step of Algorithm 1, the discriminator is allowed to reach its optimum given G, and p_g is updated so as to improve the criterion

$$\mathbb{E}_{\boldsymbol{x}\sim p_{data}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_g}[\log(1 - D_G^*(\boldsymbol{x}))]$$

  - then p_g converges to p_{data}
  - Proof =>
  - Consider V (G, D) = U(p_g, D) as a function of pg as done in the above criterion. Note that U(p_g, D) is convex in p_g. The sub derivatives of a supremum of convex functions include the derivative of the function at the point where the maximum is attained. In other words, if f(x) = $\sup_{\alpha\in A} f_\alpha(x)$ and $f_\alpha(x)$ is convex in x for every α, then $\partial f_\beta(x) \in \partial f$ if $\beta = \arg\sup_{\alpha\in A} f_\alpha(x)$. This is equivalent to computing a gradient descent update for p_g at the optimal D given the corresponding G. $\sup_D$ U(p_g, D) is convex in p_g with a unique global optima as proven in Theorem 1,therefore with sufficiently small updates of p_g, p_g converges to p_x. End of proof.

Experiments

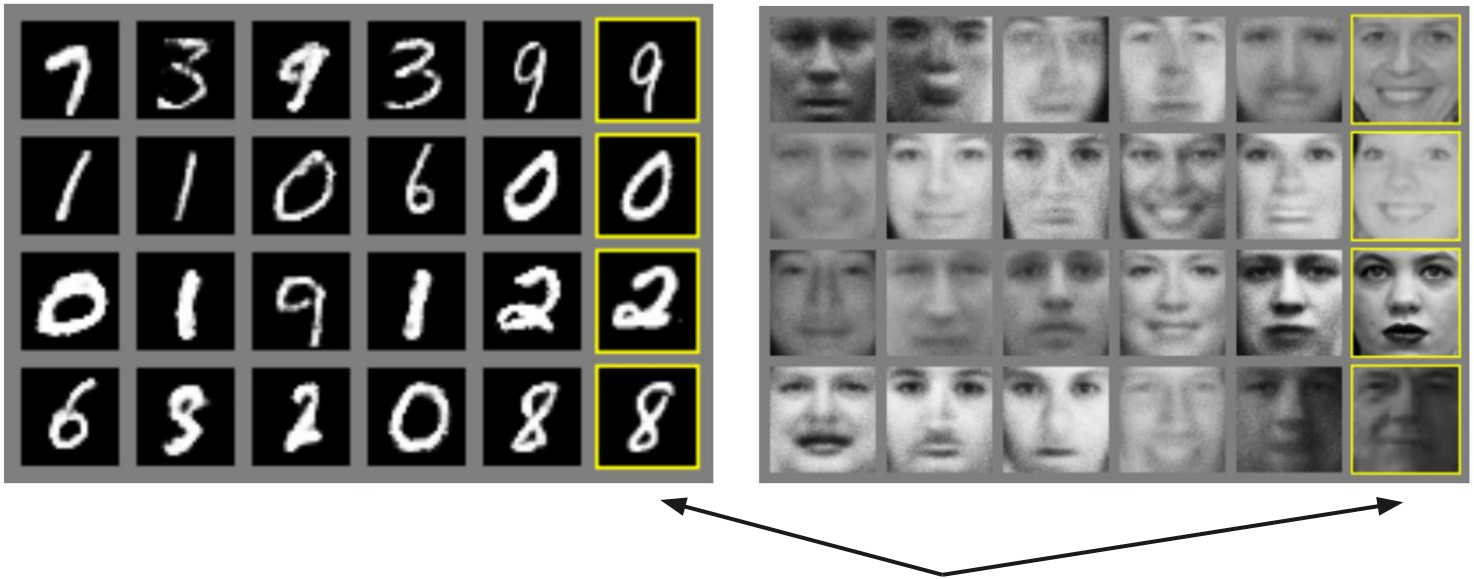
- They trained adversarial nets an a range of datasets including MNIST, the Toronto Face Database (TFD) and CIFAR-10.
- The generator nets used a mixture of rectifier linear activations and sigmoid activations, while the discriminator net used maxout activations.
- Dropout was applied in training the discriminator net. While our theoretical framework permits the use of dropout and other noise at intermediate layers of the generator.
- They used noise as the input to only the bottommost layer of the generator network.

Experiments

- They estimate probability of the test set data under p_g by fitting a Gaussian Parzen window to the samples generated with G and reporting the log-likelihood under this distribution.
- The σ parameter of the Gaussians was obtained by cross validation on the validation set.
- The following table shows Parzen window-based log-likelihood estimates.

Model	MNIST	TFD
DBN [3]	138 ± 2	1909 ± 66
Stacked CAE [3]	121 ± 1.6	$\mathbf{2110 \pm 50}$
Deep GSN [5]	214 ± 1.1	1890 ± 29
Adversarial nets	$\mathbf{225 \pm 2}$	$\mathbf{2057 \pm 26}$

Samples drawn from the generator net after training



Nearest neighbor from training set

Samples drawn from the generator net after training

Nearest neighbor from training set

Advantages and Disadvantages

- The new framework comes with advantages and disadvantages relative to previous modeling frameworks.
    - Disadvantages:
        - There is no explicit representation of $p_g(x)$.
        - D must be synchronized well with G during training.
        - The negative chains of a Boltzmann machine must be kept up to date between learning steps.
    - Computational Advantages:
        - Markov chains are never needed.
        - Only backprop is used to obtain gradients.
        - No inference is needed during learning.
        - A wide variety of functions can be incorporated into the model.

Comparative Analysis

- This table shows the comparison between GAN and the other modelling approaches.
- Training, Inference, Sampling, Evaluating p(x) and the design model are the main criteria used in this comparative analysis.
- It mainly describes the difficulties encountered by different deep generative modelling approaches for every operation involving a model.

	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

Conclusions and Future Work

- This framework admits many straightforward extensions =>
    - A conditional generative model p(x | c) can be obtained by adding c as input to both G and D.
    - Learned approximate inference can be performed by training an auxiliary network to predict z given x.
    - One can approximately model all conditionals $p(x_S \mid x_{\$})$ where S is a subset of the indices of x by training a family of conditional models that share parameters.
    - Semi-supervised learning.
    - Efficiency improvements.

QUESTIONS?