Generative Adversarial Nets

Paper by Ian Goodfellow *et. Al, 2014*

Simon Sasse WiSe21/22

Problem Description

- Given: Dataset of similar objects
  - Images of cats, handwritten digits, abstract artworks etc
- Goal: Artificially generate similar objects

What kind of similarity?

- What we don’t want:
  - Objects that have a small distance (euclidian)
- What we want:
  - Objects that come from the same probability distribution

- Example: not a good copy of an artwork but a new unique artwork.

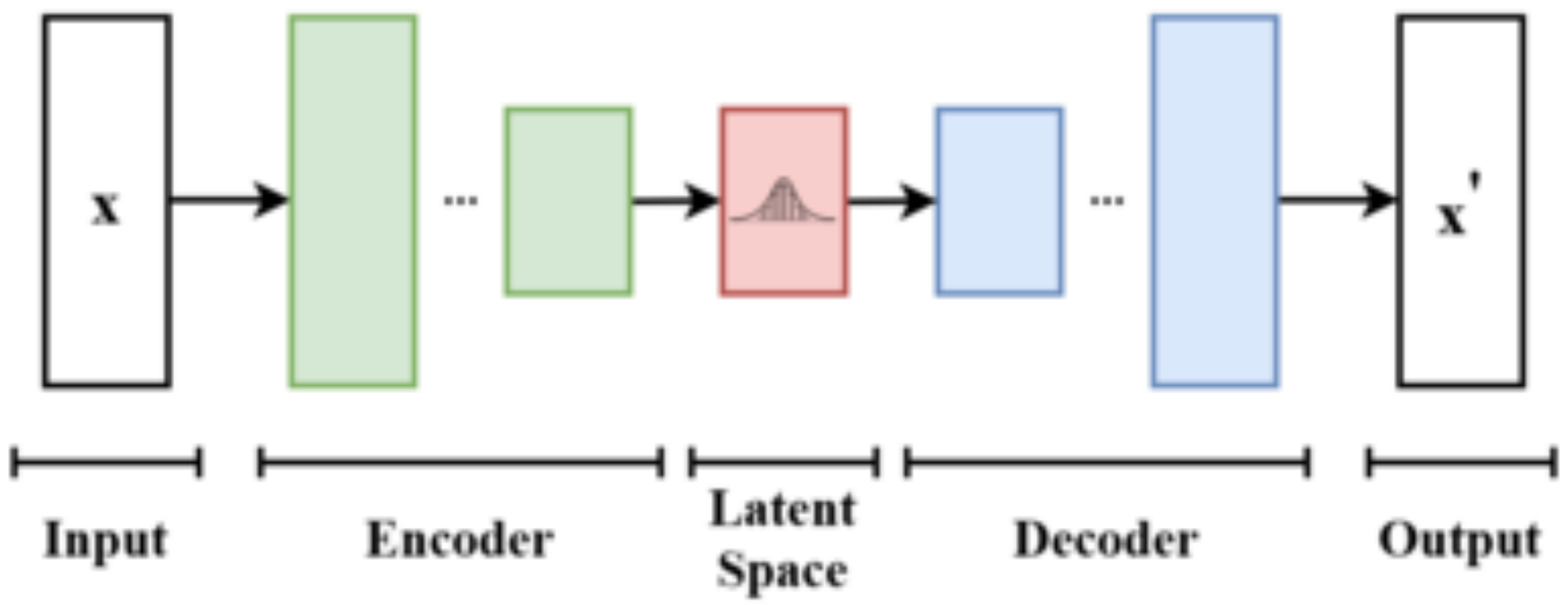
More Formal

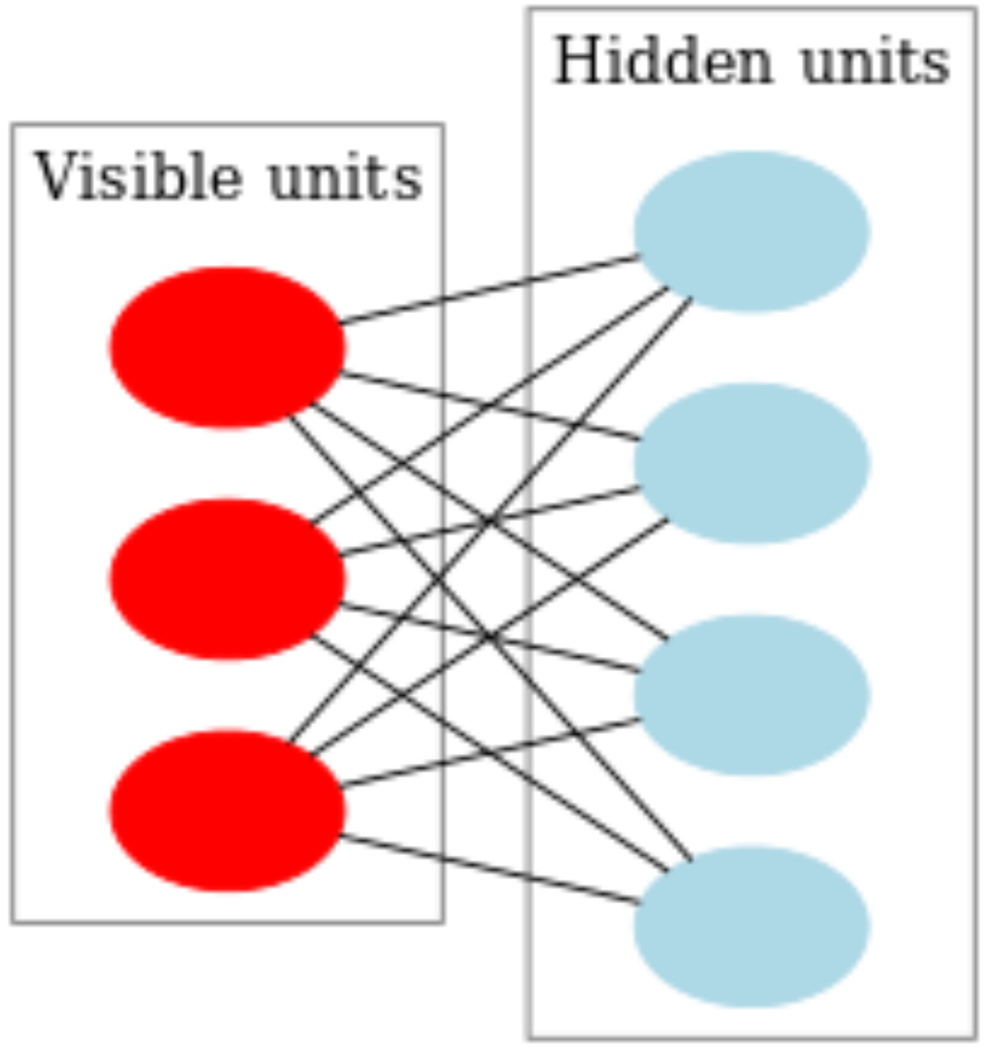
Generative vs. discriminative learning

- Generative learning:
  - Assumption: Samples come from one <u>unknown</u> probability distribution $\mathbf{p_{data}}$
  - Goal: approximate this probability distribution as $\mathbf{p_g}$
    - Drawing from $\mathbf{p_g}$ should give results similar to given dataset
- Discriminative learning:
  - Often multi class problem
  - Not learning full distribution but boundaries between classes

Other generative models

- (Deep) (Restricted) Boltzmann Machine
- Variational Autoencoders

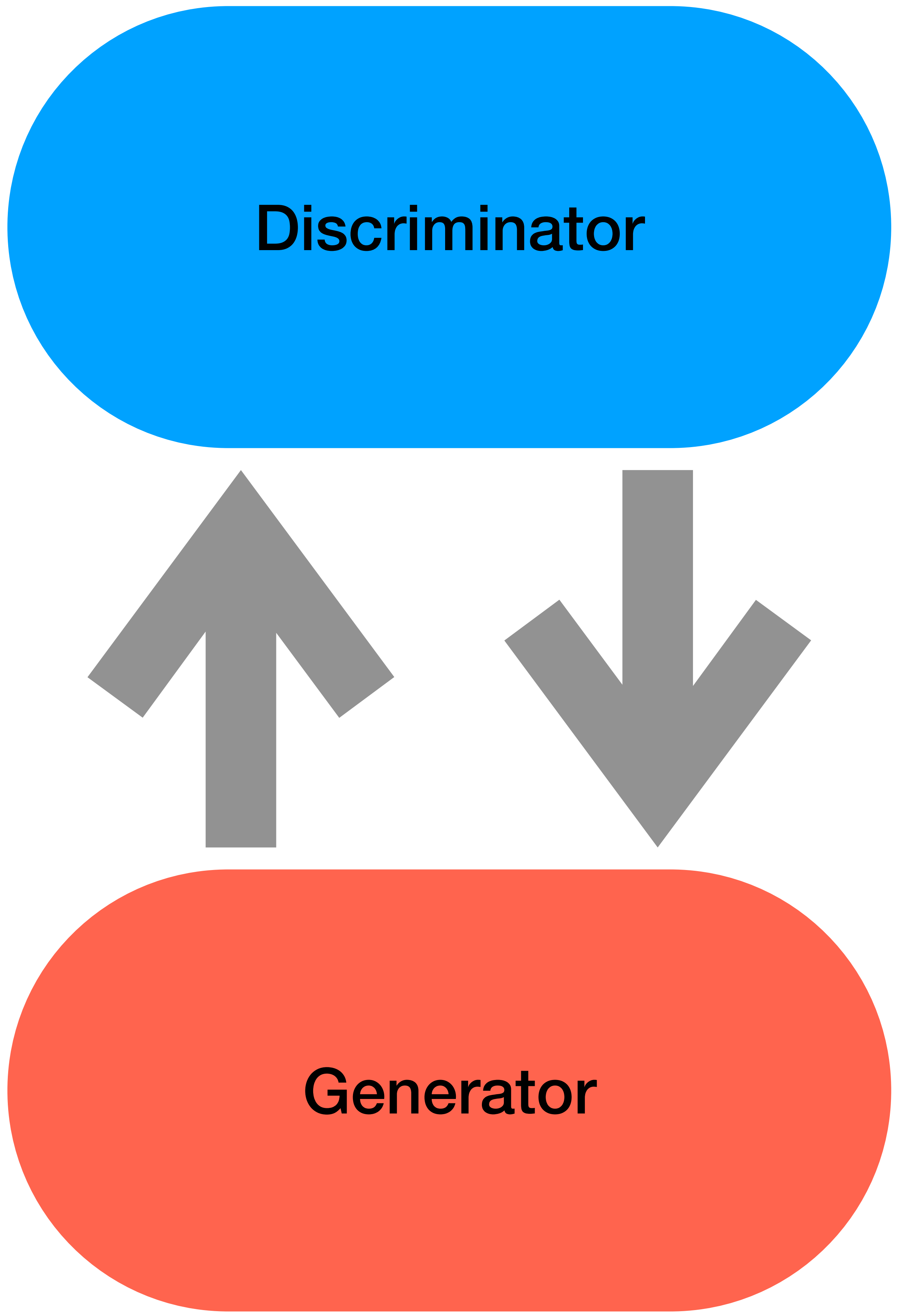




Generative Adversarial Nets

Intuitive Explanation

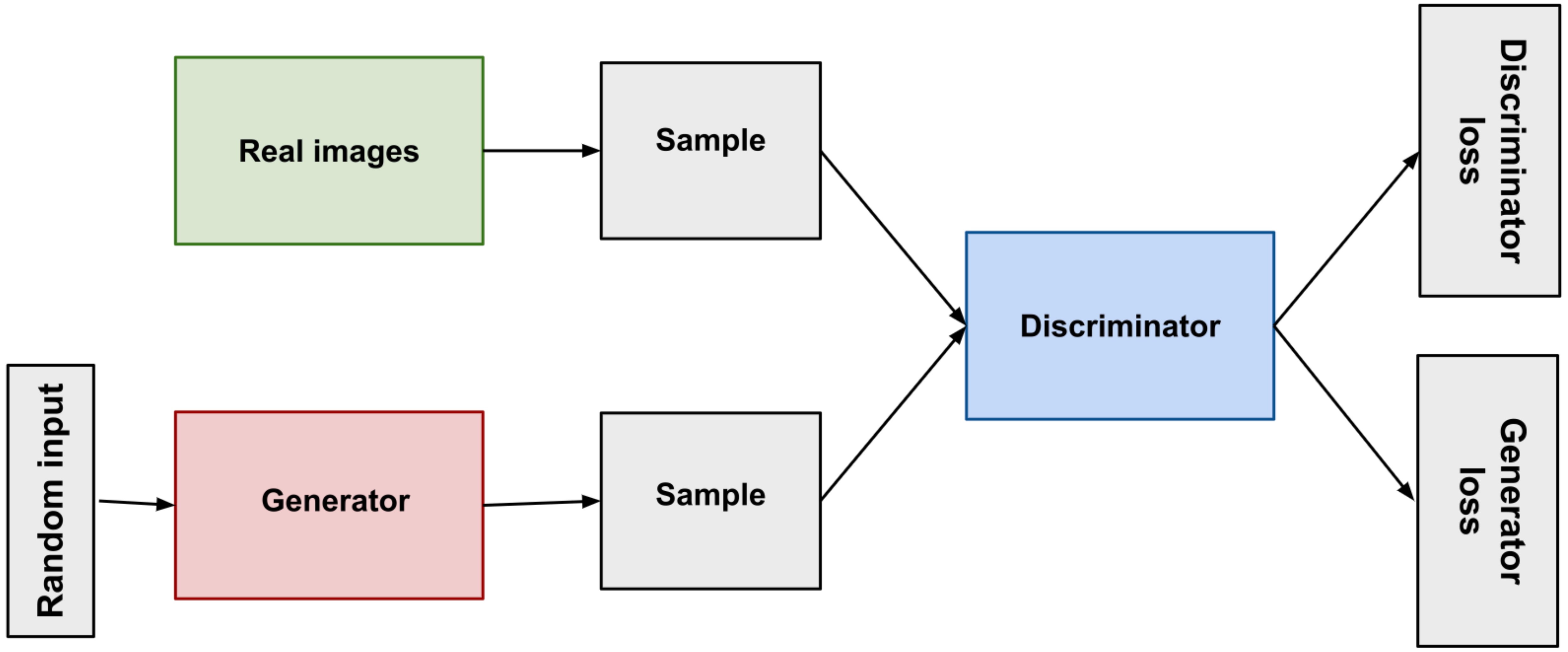
- Discriminator D discriminates between fake and real input
- Generator G generates fake data
- Minimax game
    - One player aims to max. its gain
    - Other player aims to min. its loss
    - Fight about D’s accuracy



Generative Adversarial Nets

Intuitive Explanation

- Sample from real images
- G generates random fake
- D classifies both samples
    - D Improves so that fake is classified as fake and real as real
    - G improves so that fake is classified as real



Discriminator Network

- D(x) is in [0,1]
- D(x) = 1, if x is considered real (from p_{data})
- D(x) = 0, if x is considered fake (from p_g)
- Optimal D: $D_G^*(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})}$
- If G is perfect D(x) converges to 0.5
    - No matter the input, D has to guess
    - $p_{data}(x)$ and $p_g(x) = 1$

Generator Network

- G(z) generates object of desired output size
- z is random noise from distribution p_z
- Wants to fool discriminator D

Generative Adversarial Nets

Value function

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

Training Loop

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.

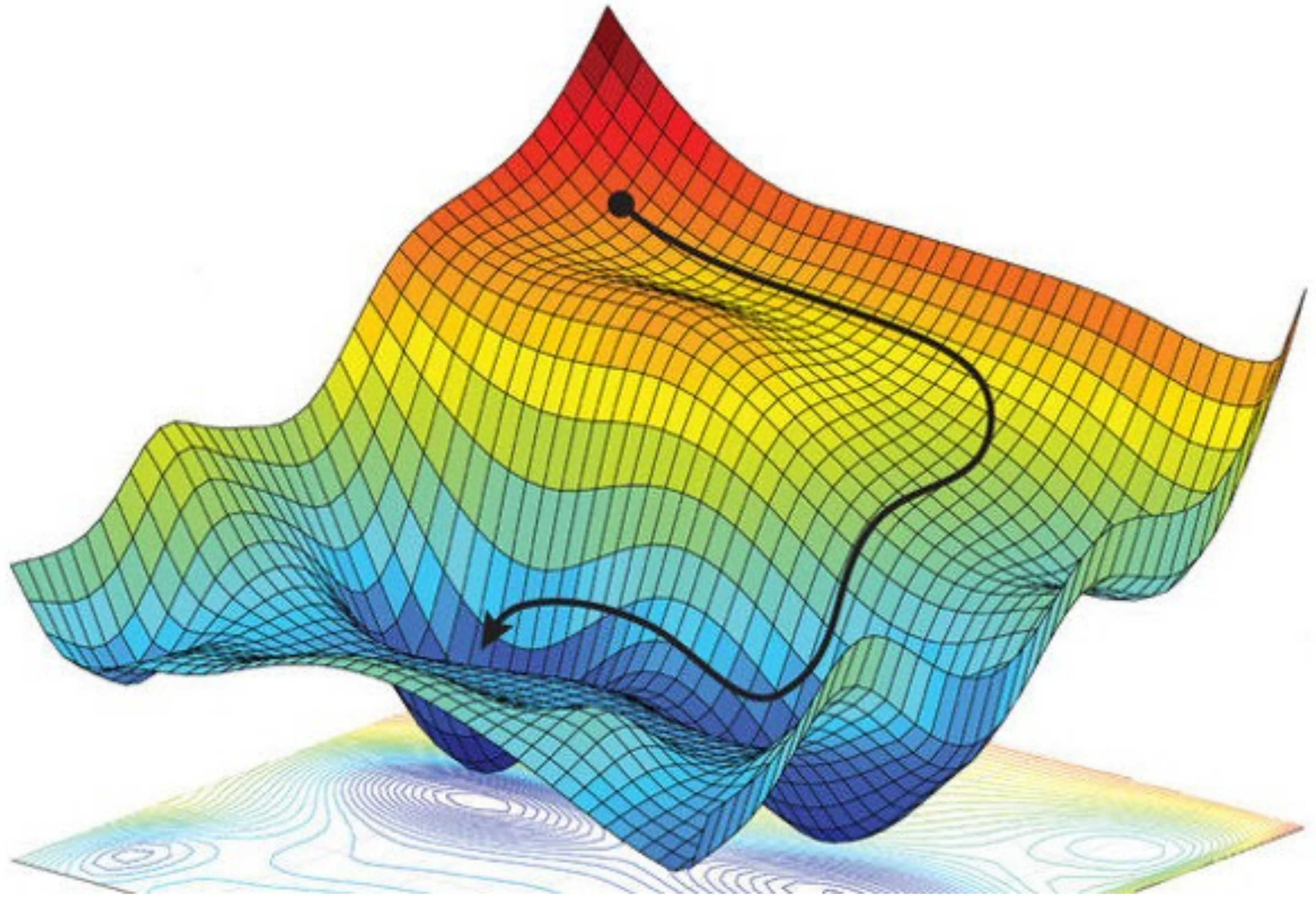
for number of training iterations **do**
  for k steps **do**
  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

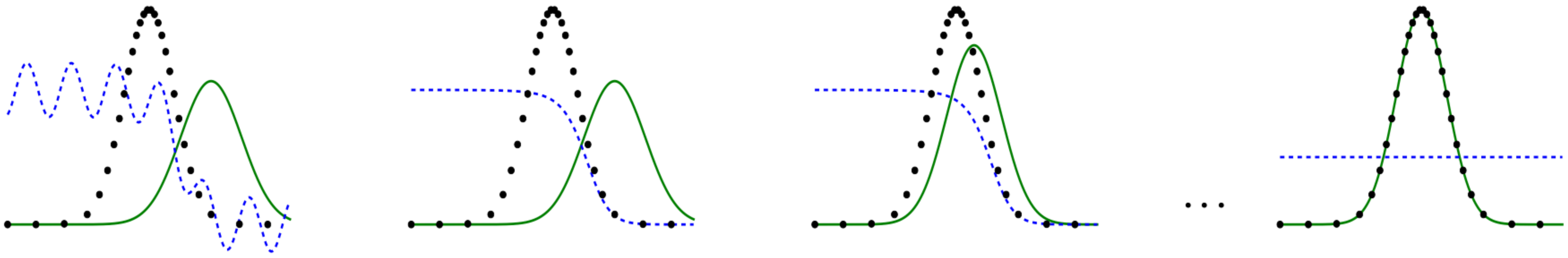
  end for
  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

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

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



Training steps of GANs



1. Training near convergence
2. Discriminator adapted
3. After Generator updated
4. The Discriminator is useless because the Generator is perfect

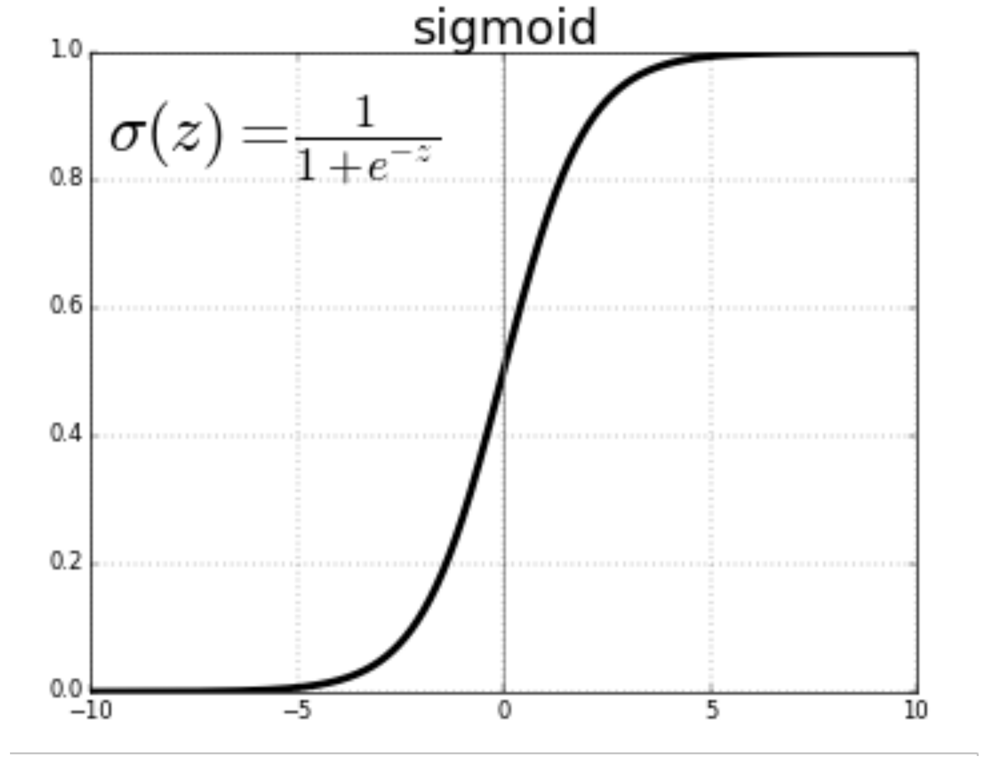
Theoretical results

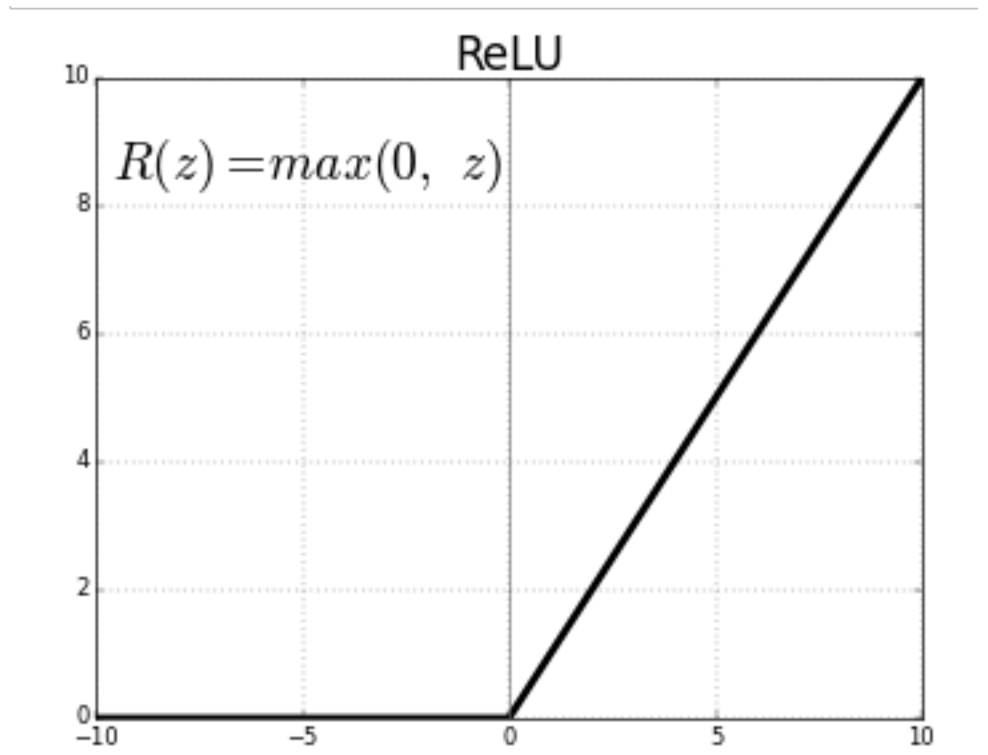
Some proofs

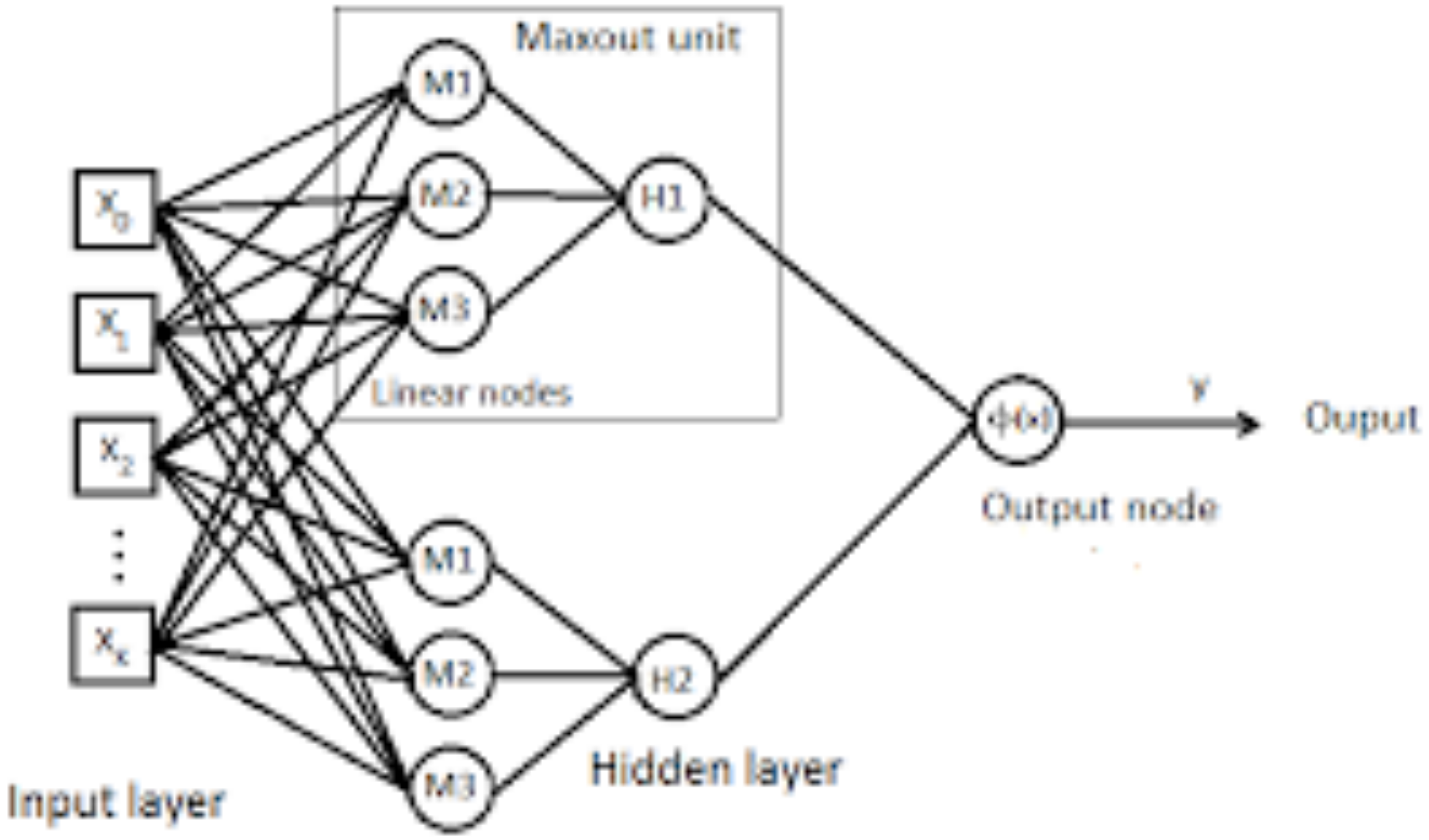
- They proof that:
  - The minimax game has a global minimum and maximum
  - Convergence of the training algorithm

Experiments

- Datasets:
  - MNIST (handwritten digits)
  - Toronto Face Database (TFB)
  - CIFAR-10 (tiny images)
- G and D are Multilayer Perceptrons
- Hyperparameters:
  - Activation Functions:
    - G: Sigmoid, rectifier linear (ReLU)
    - D: maxout







(Dis)advantages of GANs

Pros:

- Backprop is possible
- G never sees the data
  - No copying

Cons:

- No explicit representation of p_g
- Synchronization of D and G (Unstable training)
- (Mode collapse) can occur
  - G models only part of p_{data}

Results

Comparison

- No results for CIFAR-10
- Parzen window estimate
    - Estimating p_g
    - Calculate likelihood of test data under p_g
    - Likelihood(model | data)
- Not the best evaluation

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	$\mathbf{225 \pm 2}$	$\mathbf{2057 \pm 26}$

Conclusion and future work

- Proof of concept of GANs
- Conditional GAN
  - Adding input C to G and D
  - C could be label of digit (0..9)
- Semi-supervised learning
  - Use Discriminator's feature to improve classifiers
  - Make D predict all labels and fake (2 modes)
- Improve coordination of G and D for efficiency
- (Improve model evaluation)

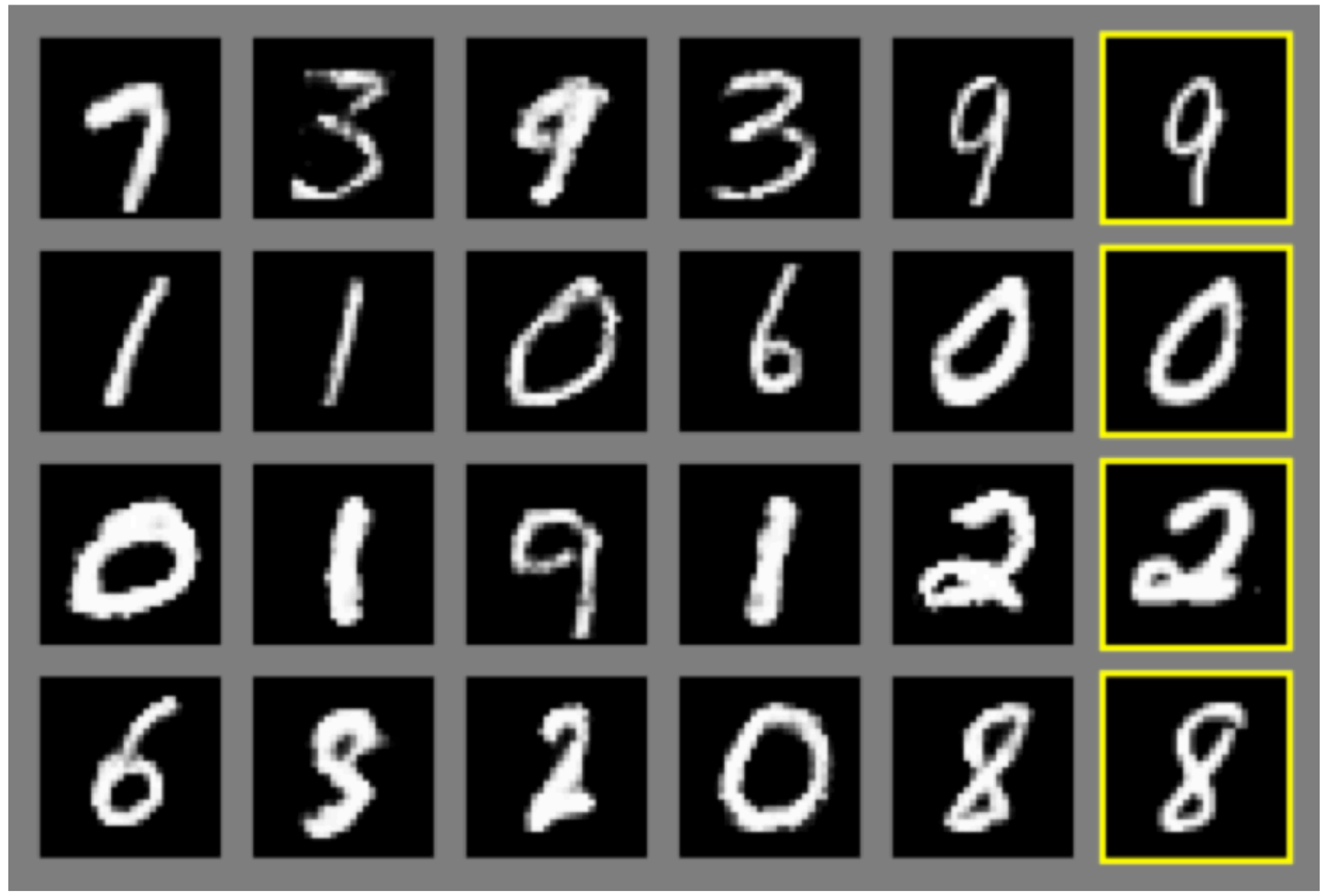
Evaluating GANs

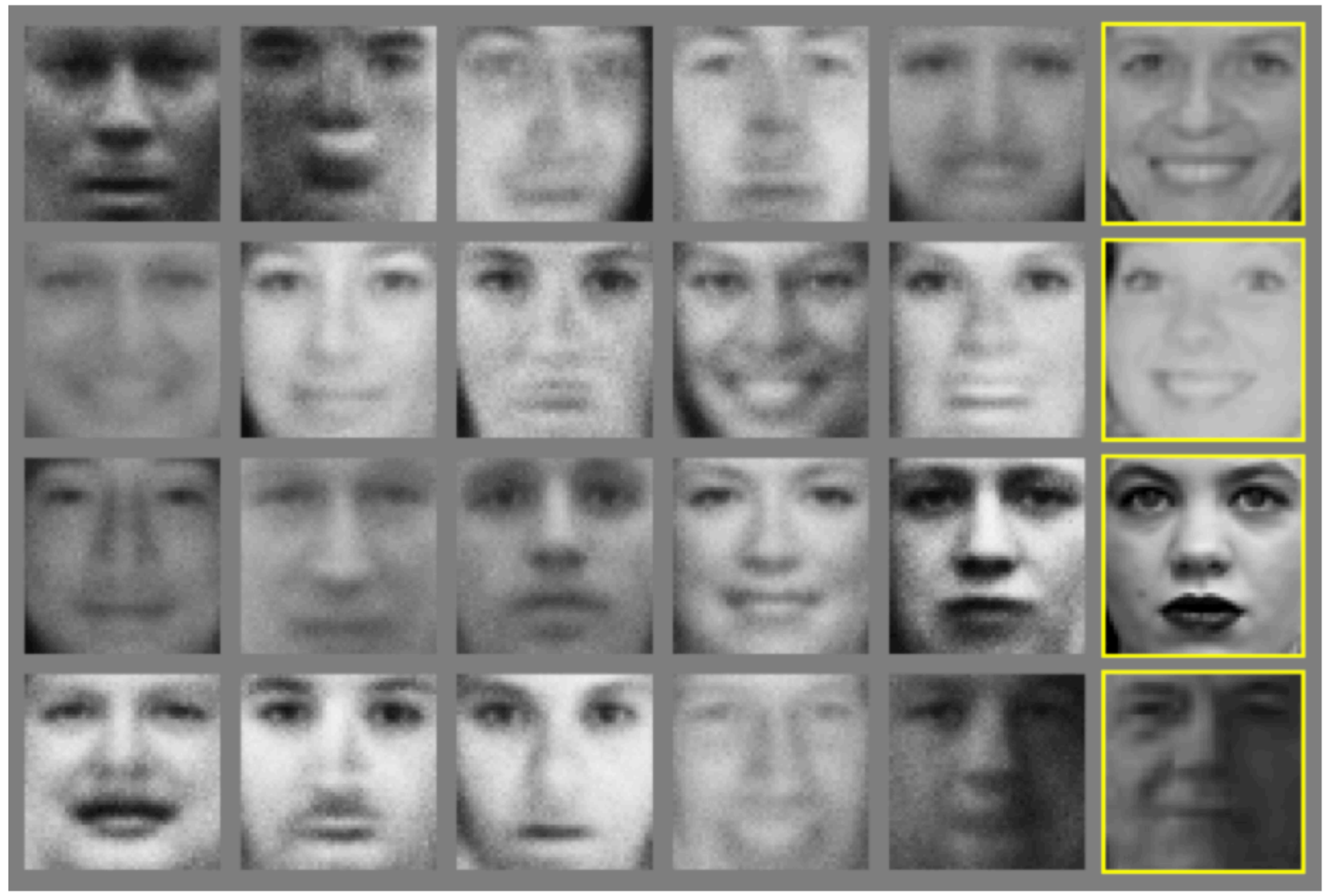
- Cannot measure accuracy like in discriminitive models
- Do not want Distance Evaluation
- Want likelihood of p_g
- Parzen Window Estimates
  - Cases where p_g performs better than p_{data}
- Manual inspection
  - Subjective and ineffective
- FID
  - Compares p_g and p_{data}
  - Using image analysis CNN Inception v3

Results

2014

- Yellow framed = from train-data
- Digits look ok (only 27x27px)
- Faces blurry and noisy





Results

Moving through output space

- Linearly interpolating between coordinates in input-space of Generator

Thats it

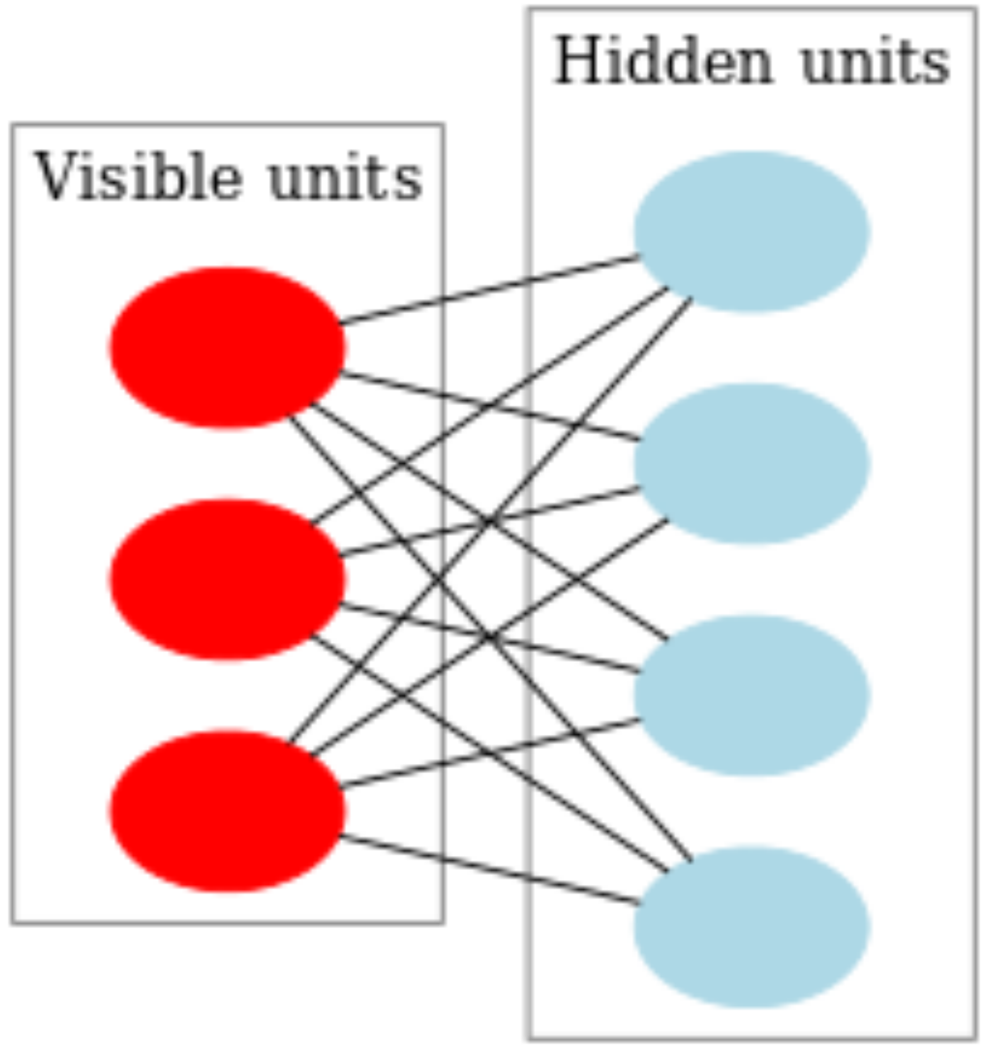
- My longest presentation ever.
- I hope it was informative.

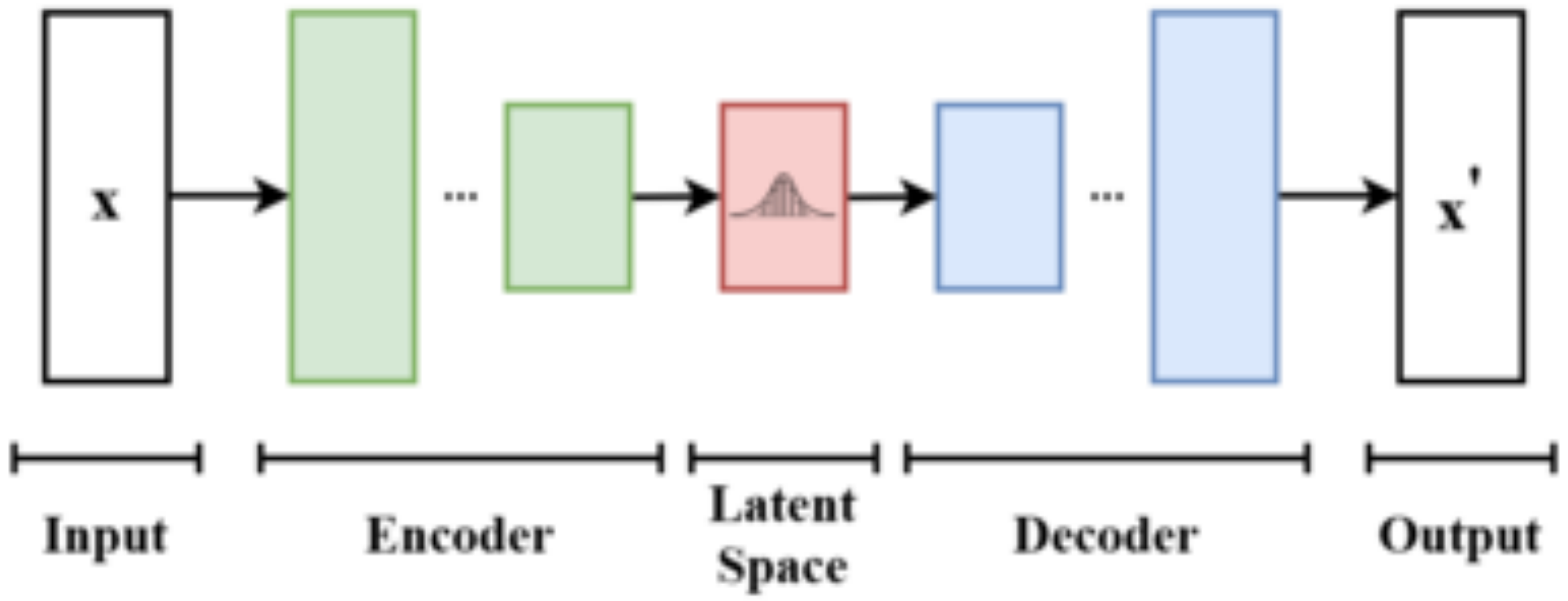
Sources

- Paper: https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf
- cGAN: https://arxiv.org/pdf/1411.1784.pdf
- sGAN: https://arxiv.org/abs/1606.01583
- GSN: https://arxiv.org/pdf/1503.05571.pdf
- DBM: http://proceedings.mlr.press/v5/salakhutdinov09a/salakhutdinov09a.pdf
- VAE Picture: https://en.wikipedia.org/wiki/Variational_autoencoder
- Mathematical explanation: https://arxiv.org/pdf/2009.00169.pdf
- GAN difficulties: https://arxiv.org/pdf/2006.06900.pdf
- https://thispersondoesnotexist.com
- StyleGan2: https://arxiv.org/abs/1912.04958
- StyleGan2 Interpolation: https://www.youtube.com/watch?v=6E1_dgYlifc

Other generative models

- (Deep) (Restricted) Boltzmann Machine
  - Can be stacked (DBM)
  - Stochastic -> ~~backprop~~
- Generative Stochastic Networks (GSN)
  - Based on Variational Autoencoders
  - Learnings transition probabilities of Marko
  - *Reparametrization Trick* enables backprop





Backpropagation

- Calculate partial derivatives of loss function with regards to parameters of last layer
- Continue backwards through the model

Gradient Descent

- Sample n samples
- Forward propagation through network
- Obtain prediction/output
- Evaluate loss function
- Calculate gradient with backprop
  - Layer by layer
- Multiply vector with learning rate
- Subtract/adding vector from model parameters
- Lots of matrix operations