UNIVERSITÄT BIELEFELD

GANs - Generative Adversarial Networks

Florian Nolte, Jan-Hendrik Deke, Alexander Symanski

Deep Learning

25.06.2020

Content

1. Introduction to Generative Modelling
2. Motivation and Intuition behind GANs
3. Formalization of GANs
4. Experiments
5. Advantages and Disadvantages
6. Recent Advances and the State of the Art
7. Discussion

Introduction to Generative Modelling

Sources

Unless stated otherwise, all content of the introduction is based on:

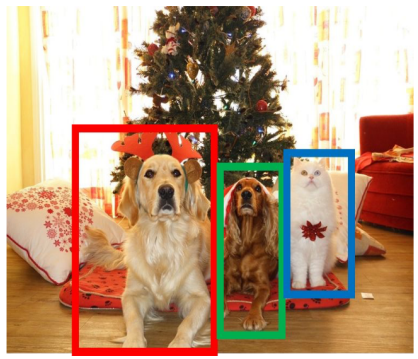
- Dive into Deep Learning, chapter 17 (Zhang et al.)
- NIPS 2016 Tutorial on GANs (Ian Goodfellow)
- Stanford Lecture CS231n, Lecture 13, 2017

Common Tasks in Machine Learning

Supervised learning

- Classification
- Regression
- Object detection

-> learn function $f(x) = y$ that maps label y to data x

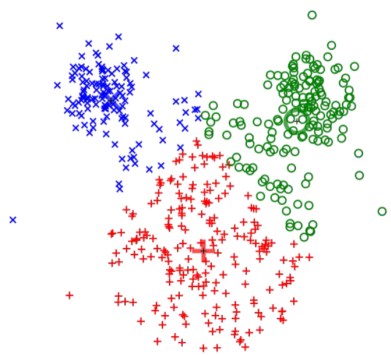


DOG, DOG, CAT

Unsupervised learning

- Clustering
- Dimensionality reduction
- Density estimation

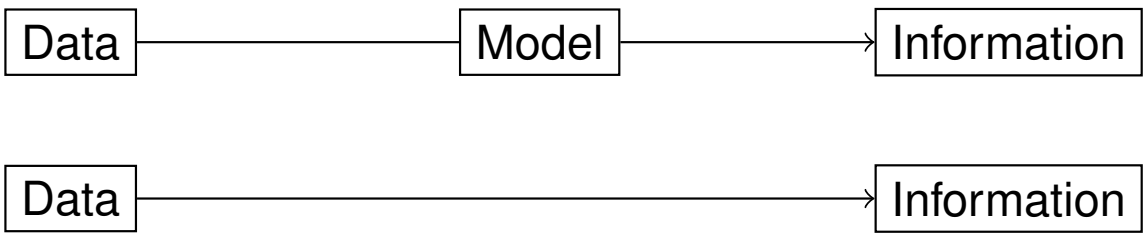
-> learn underlying structure of data



The goal is usually to create a model of the data and extract some information.

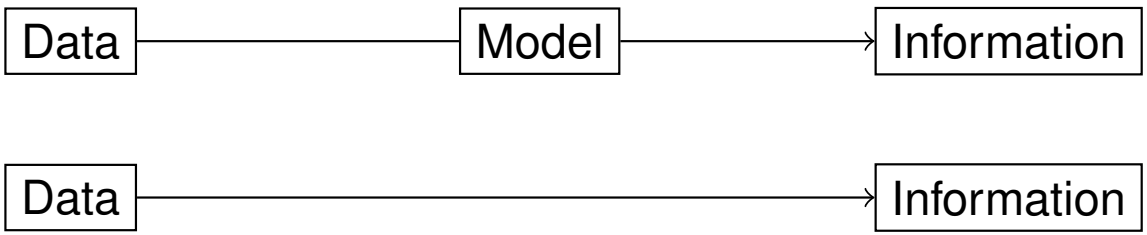
Common tasks in Machine Learning

ML in introductory lectures:

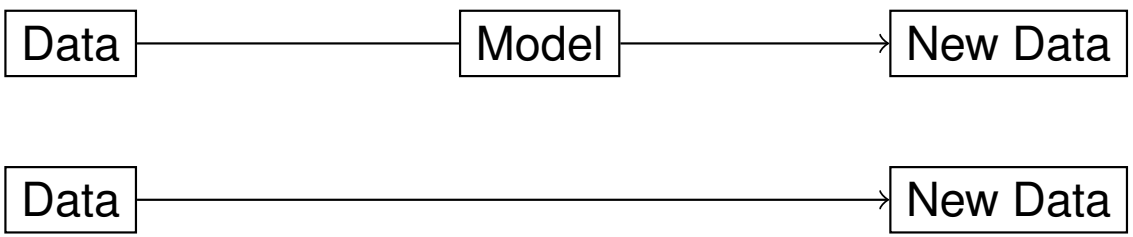


Common tasks in Machine Learning

ML in introductory lectures:



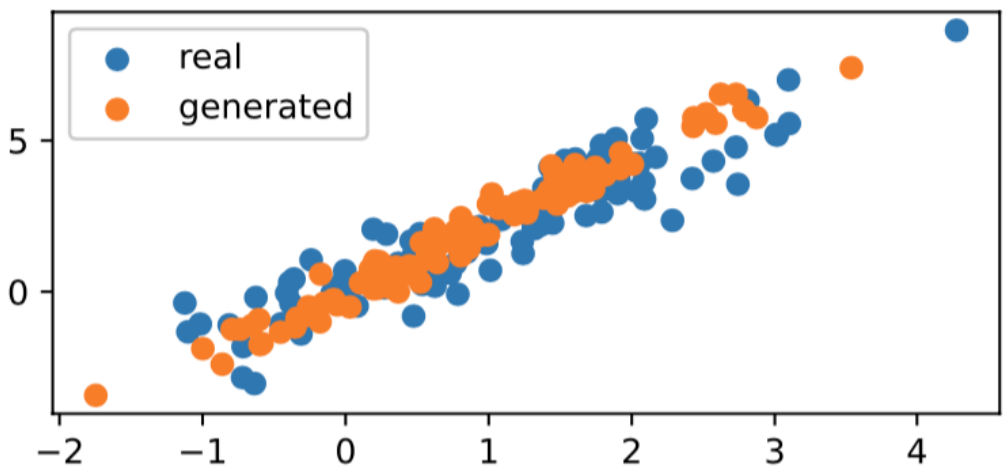
Generative models:



Generative Modelling

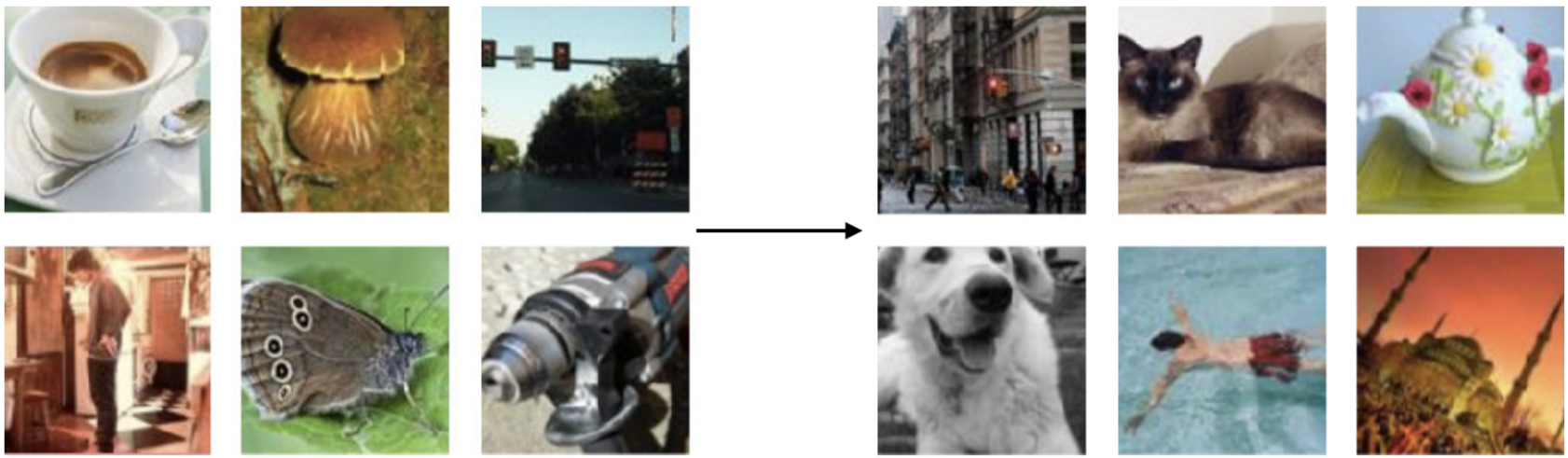
Goal:

> Given some training data, generate new data from same distribution.



Not particularly hard for simple datasets.
But what about more complex data?

Generating Images

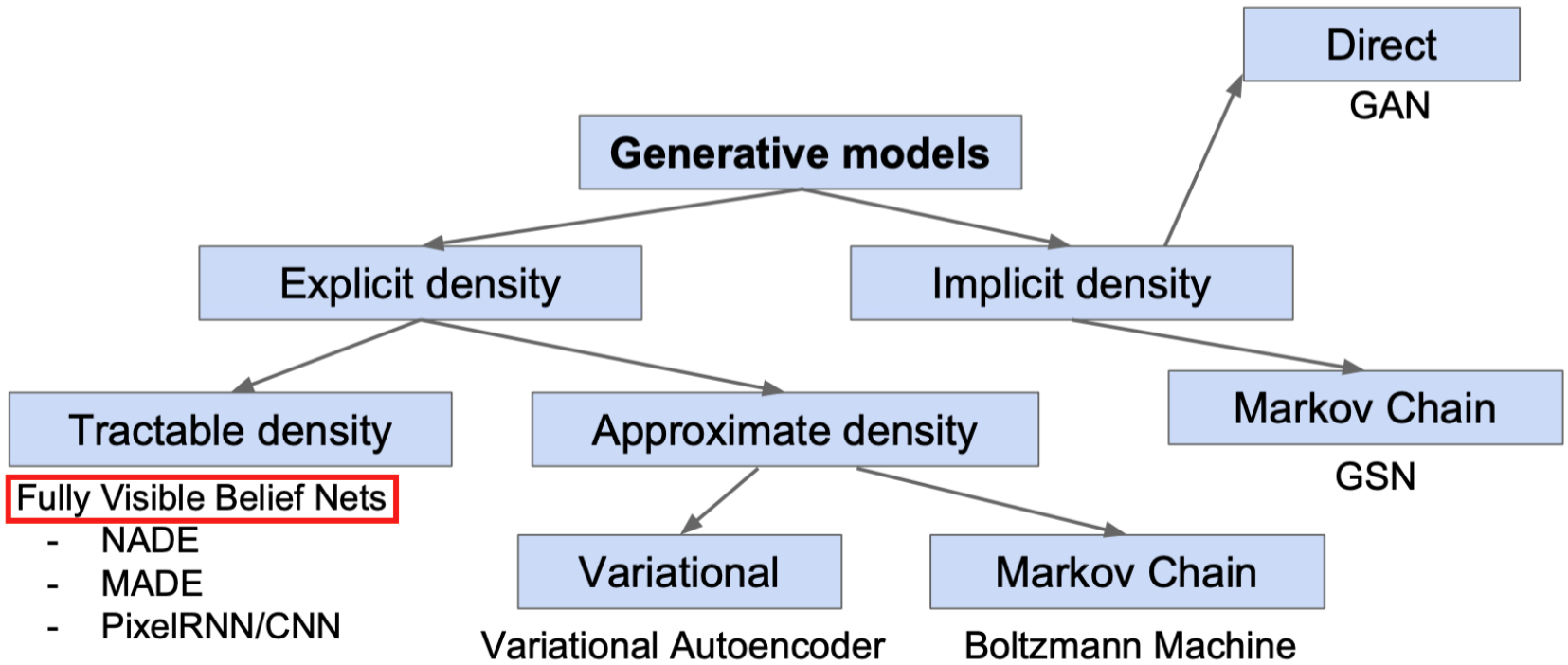


Real dataset

new generated samples

- Requires very complex, nonlinear algorithms (e.g. deep neural networks)
- Huge amounts of data needed
- -> Unsupervised approaches necessary ("cheap data")

Taxonomy of Generative Modelling



Fully Visible Belief Nets

Explicit density model:

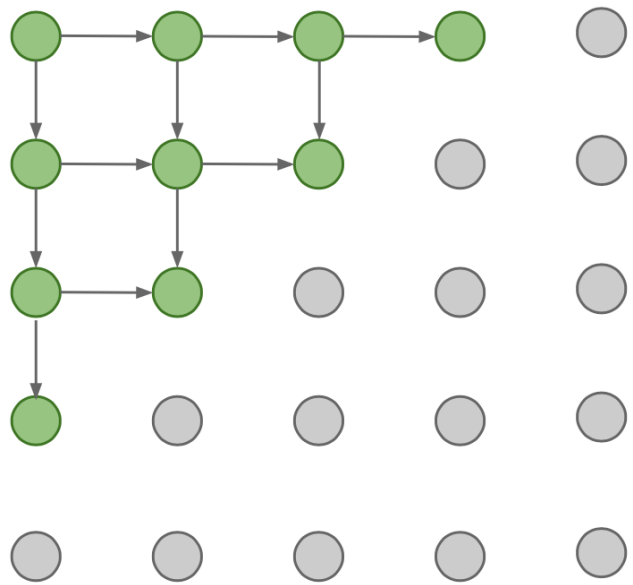
Likelihood of picture x is product of probabilities of pixel values x_i, given all previously drawn pixels.

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

- $p(x_i | x_1, ..., x_{i-1})$ is expressed through neural network (e.g. RNN or CNN).
- Find weights by maximising likelihood of training data

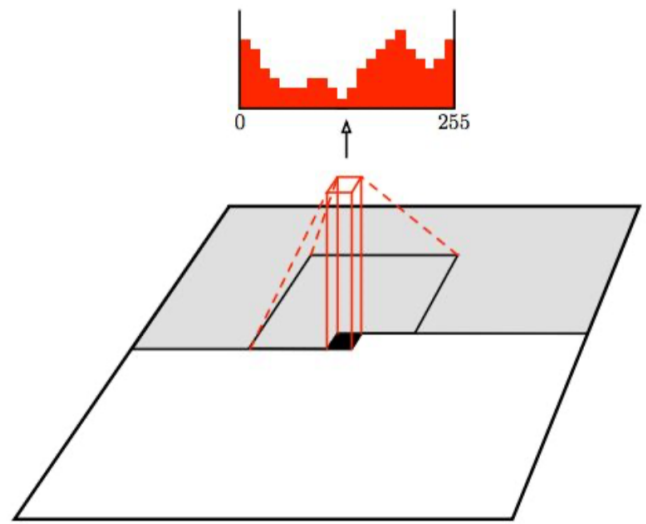
PixelRNN

- Start in top left corner
- Neighbours depend on already drawn pixels
- Dependency modelled through a recurrent neural network
- Major disadvantage: very slow (generation and training)



PixelCNN

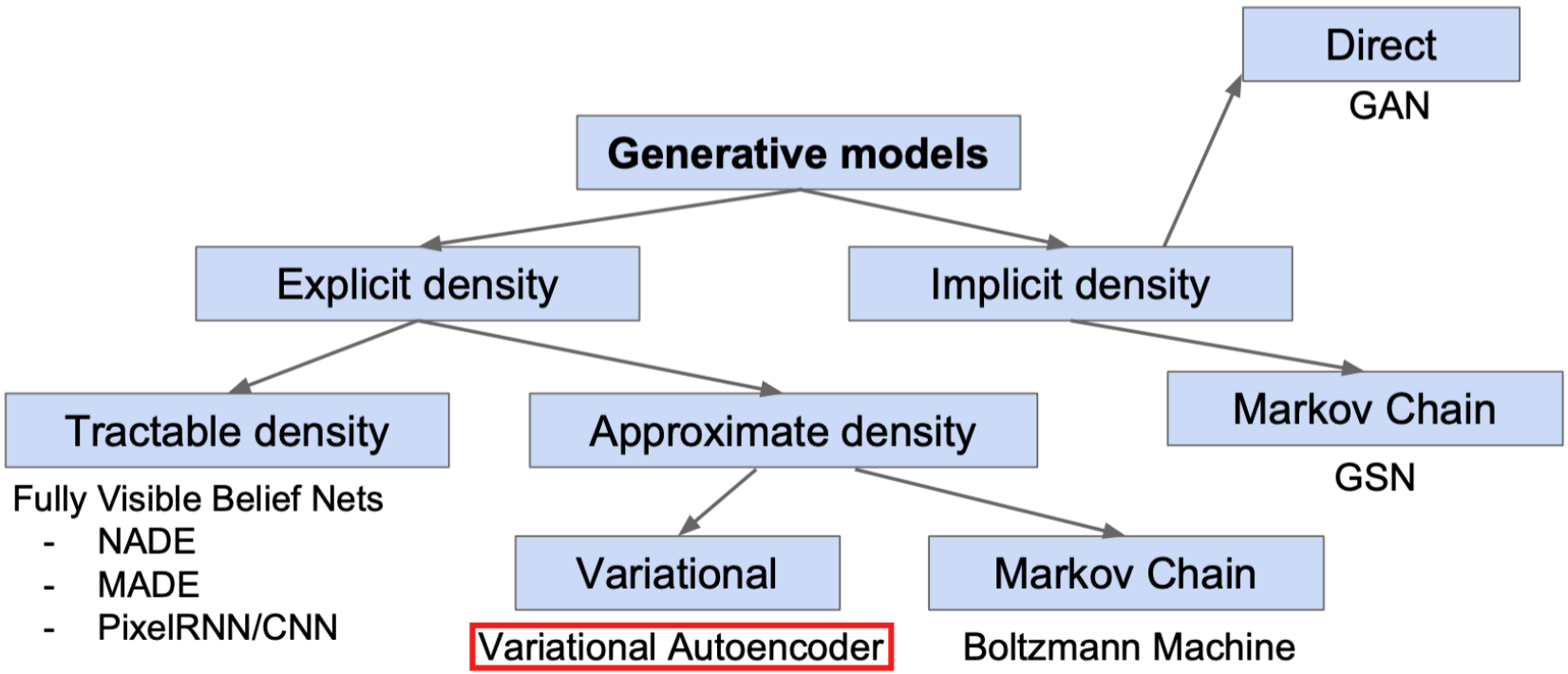
- Replace RNN with convolutional neural network
- Start in top left corner
- Draw next pixels based on context region
- Faster than PixelRNN, but still slow



Results

PixelCNN elephants

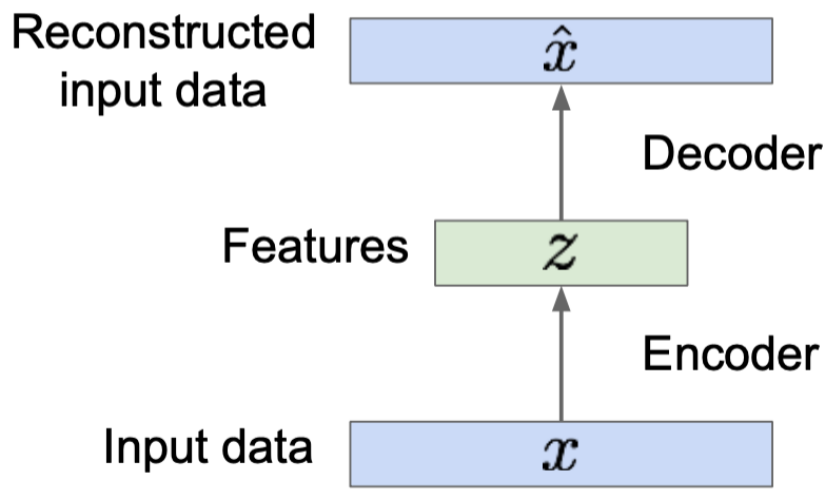
Taxonomy of Generative Modelling



Autoencoders

Unsupervised approach for **dimensionality reduction** and low dimensional feature representation

- **Encoder:** Neural network compresses data x to feature space z
- **Decoder:** Reconstructs data from features z
- Minimise difference $||x - \hat{x}||^2$
- Normally: decoder is thrown away afterwards
- If data can be reconstructed: z is good representation of data



Variational Autoencoder

- **Idea:** Don't throw away the decoder.
  Use it to generate new, unseen data!
- If we can put in some random features z, we might get out sensible new data!

Variational Autoencoder

- **Idea:** Don't throw away the decoder.
  Use it to generate new, unseen data!
- If we can put in some random features z, we might get out sensible new data!
- **Solution:** Construct special probabilistic Autoencoder where we can calculate the distribution of the features z
- Put in some values for z that follow this distribution.
  -> Out comes a new image!

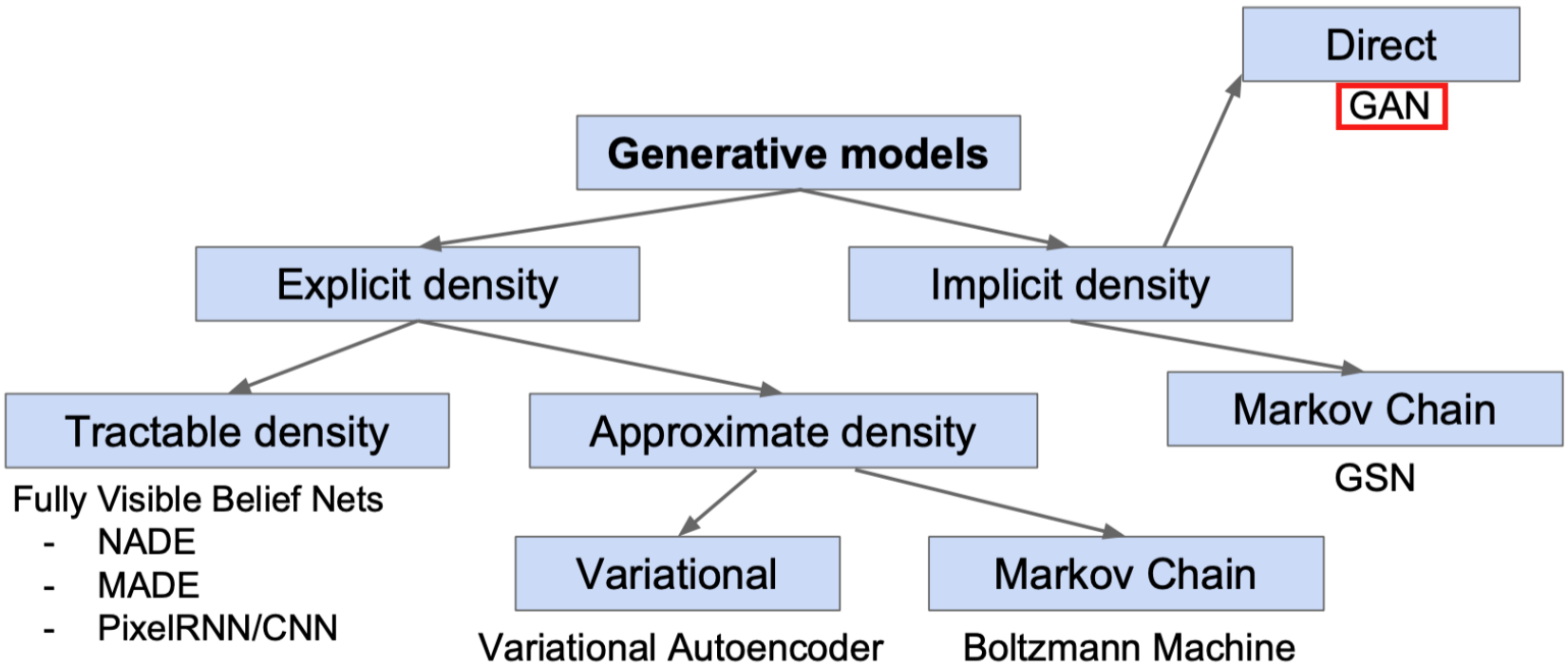
Variational Autoencoders: Results

- Faces are a little bit blurry and not very high quality

Variational Autoencoders: Results

- Dimensions of z correspond to features in the data
  (here: head pose, degree of smile)
- Very useful when trying to generate specific pictures!

Taxonomy of Generative Modelling



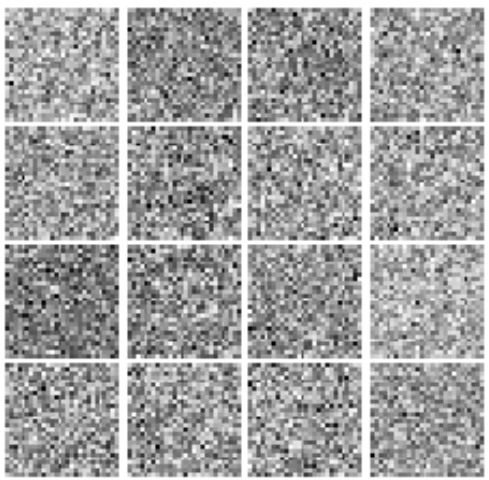
Motivation and Intuition Behind GANs

GANs - Motivation

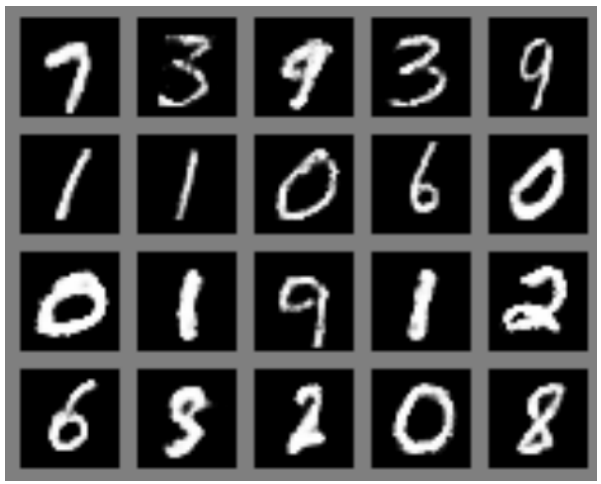
- We don’t care to model these complex distributions!
- **Goal:** Simply produce nice samples
- **Problem:** We can’t easily sample from the data distribution (or any complex distribution!)

GANs - Motivation

- We don't care to model these complex distributions!
- **Goal:** Simply produce nice samples
- **Problem:** We can't easily sample from the data distribution (or any complex distribution!)
- **Solution:** Sample from a very simple distribution (random noise) and transform it to look like the target distribution!



Input noise



Noise after transformation

GANs - Motivation

How to model such a complex, nonlinear transformation?
-> Deep Neural Networks

Generator: Neural network that transforms random noise into sample from data distribution.

- How do we know whether the transformation is good?
  What is the loss function?
- Without this, we cannot train the generator!

GANs - Ideas for Loss Functions

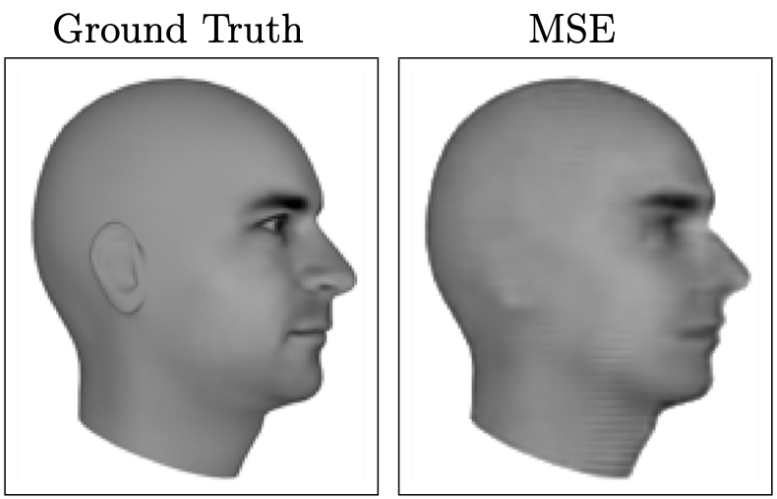
- *Likelihood:*
  We do not model the likelihood -> cannot be optimised! ↯

GANs - Ideas for Loss Functions

- *Likelihood:*
  We do not model the likelihood -> cannot be optimised! ↯
- $Loss = ||\hat{x} - x_i||^2$
  We do not want to reproduce the dataset (i.e. overfit) ↯

GANs - Ideas for Loss Functions

- *Likelihood:*
  We do not model the likelihood -> cannot be optimised! ↯
- $Loss = ||\hat{x} - x_i||^2$
  We do not want to reproduce the dataset (i.e. overfit) ↯
- $Loss = ||\hat{x} - \overline{x}||^2$
  We do not want to produce an "average", blurry sample! ↯



Next video frame prediction - MSE solution is blurry

The average human face
- not a very good sample!

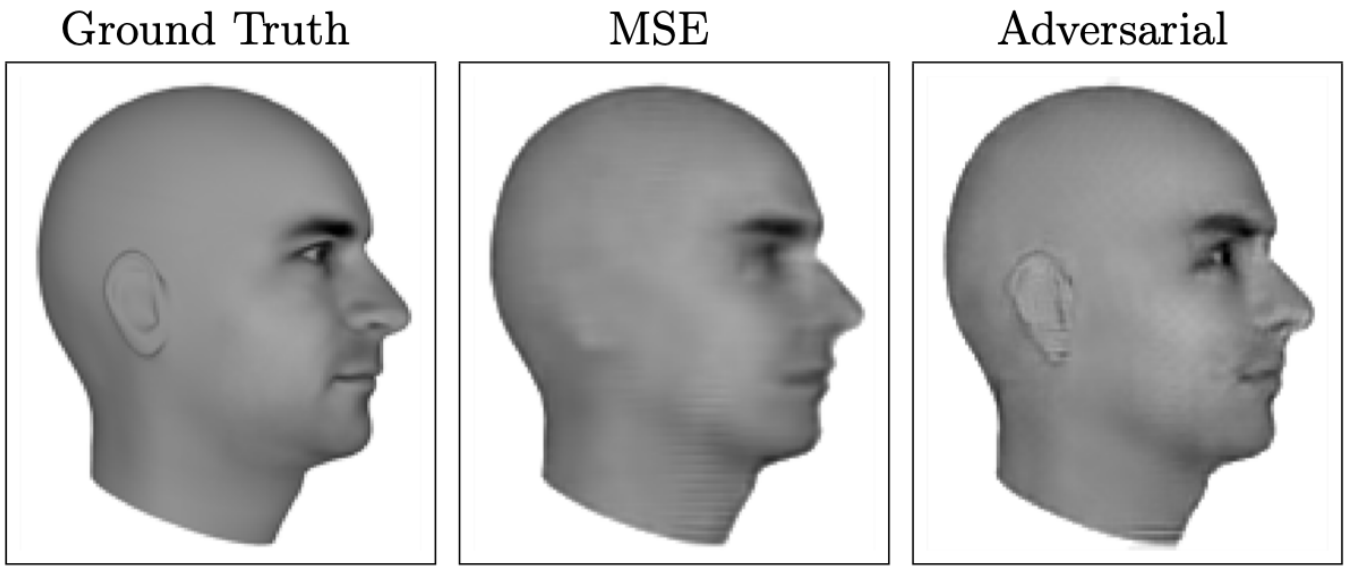
The Adversary

The GAN solution to the loss-function: Game Theory!

Discriminator: Adversarial neural network (opponent) that judges the performance of the generator.

- Normal classifier that predicts if an image is real or "fake"
- Output from the discriminator is used as a loss for the generator
- The two networks "play against each other"
- Common analogy: Police vs counterfeiters

GANs VS MSE



- GAN sample: sharp edges, looks realistic
- MSE solution would not be accepted by the discriminator (no person in the dataset is blurry!)
- There is not one best output for the generator
  -> multiple solutions can be accepted by discriminator

Police VS Counterfeiters

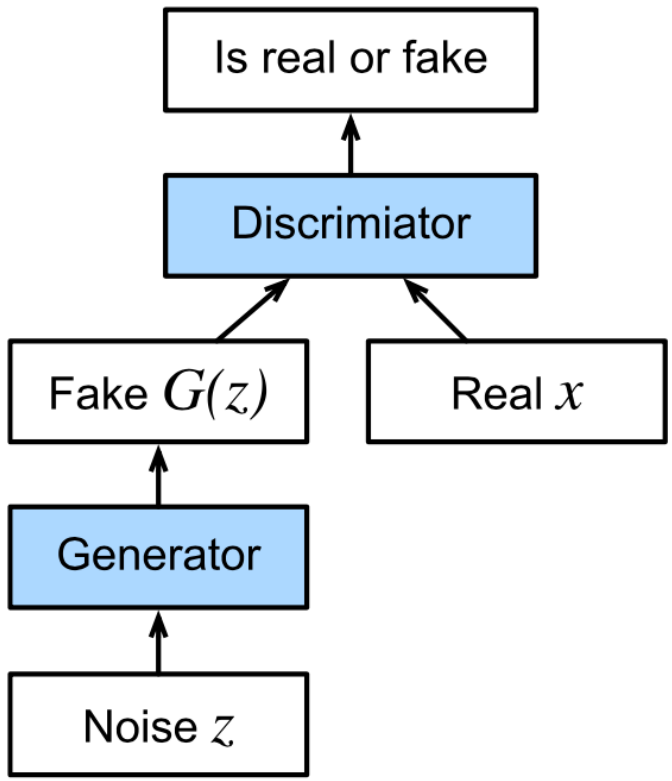
- **Counterfeiters:** Try to produce real looking money
- **Police:** tries to identify the fake money
- Over time, both get better at their tasks
- Police is really good at identifying fake money, but the counterfeiters still manage to fool them?
  => Counterfeiters must have produced really good looking money!

- Counterfeiters = Generator
- Police = Discriminator

GAN - Overview

- **Generator:** Gets better at fooling the discriminator
- **Discriminator:** Gets better at classifying data
- In the end: Fake samples might be indistinguishable from the real data!

- Only the discriminator sees the data!
- Generator can only improve based on the discriminators feedback
  -> Generator does not simply recreate dataset



Formalization of GANs

Sources

Unless stated otherwise, all content of the formalization is based on:

- Goodfellow, Ian, et al. "Generative adversarial nets." Advances in neural information processing systems. 2014.
- NIPS 2016 Tutorial on GANs (Ian Goodfellow)
- https://srome.github.io/An-Annotated-Proof-of-Generative-Adversarial-Networks\-with-Implementation-Notes/
- https://seas.ucla.edu/~kao/nndl/lectures/gans.pdf

A Perfect Generator - Mathematically

- **Goal**: Generate data that is indistinguishable from real data
  - **Mathematically**: Two Random Variables need to be equal in distribution

Equal Probability Density Functions: $p_G(x) = p_{\text{data}}(x)$

- **Strategy**: Let G play a game against D in order to learn!
  - Understand goals and strategies of your opponent
  - Based on the strategy of your opponent, devise your own!

The Discriminator's Goal - Part I

$$\max \mathbb{E}_{\boldsymbol{x} \sim p_{\mathrm{data}}(\boldsymbol{x})} \underbrace{[\log D(\boldsymbol{x})]}_{\substack{\text{Classify real} \\ \text{samples as real}}}$$

- $D(x) \approx 1$ means D classifies data as real
- $D(x) \approx 0$ means D classifies data as fake
- Therefore, $D's$ goal is to maximise this expectation

The Discriminator's Goal - Part II

$$\max \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - \underbrace{D(G(\boldsymbol{z}))}_{\substack{\text{Classify fake} \\ \text{samples as fake}}})]$$

- $D(G(z)) \approx 1$ means D classifies data coming from G as real
- $D(G(z)) \approx 0$ means D classifies data coming from G as fake
- Therefore, $D's$ goal is to maximise this expectation

The Discriminator's Goals Combined

$$\max V(D, G) := \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})} \underbrace{[\log D(\boldsymbol{x})]}_{\substack{\text{Classify real} \\ \text{as real}}} + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})} [\log(1 - \underbrace{D(G(\boldsymbol{z}))}_{\substack{\text{Classify fake} \\ \text{as fake}}})]$$

- The **Value Function** may remind you of the cross-entropy loss as used in logistic regression

The Discriminator's Goals Combined

$$\max V(D,G) := \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\underbrace{[\log D(\boldsymbol{x})]}_{\substack{\text{Classify real}\\ \text{as real}}} + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1-\underbrace{D(G(\boldsymbol{z}))}_{\substack{\text{Classify fake}\\ \text{as fake}}})]$$

- The **Value Function** may remind you of the cross-entropy loss as used in logistic regression
- What will the be the counter strategy of the Generator?

The Generator's Counter Strategy

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

- G wants to minimise what D has maximised: $\min_G V(D^*, G)$
- G and D are thus playing a **minimax game**

Proof Structure

- To Show:

$$p_G = p_{\text{data}}$$

Proof Structure

- To Show:

$$p_G = p_{\text{data}}$$

- Find Optimal

$$D_G^* = \frac{p_{\text{data}}}{p_{\text{data}} + p_G}$$

Proof Structure

- To Show:

$$p_G = p_{\text{data}}$$

- Find Optimal

$$D_G^* = \frac{p_{\text{data}}}{p_{\text{data}} + p_G}$$

- Show

$$p_G = p_{\text{data}} \iff \min V(G, D^*) = -\log(4)$$

Proof Structure

- To Show:

$$p_G = p_{\text{data}}$$

- Find Optimal

$$D_G^* = \frac{p_{\text{data}}}{p_{\text{data}} + p_G}$$

- Show

$$p_G = p_{\text{data}} \iff \min V(G, D^*) = -\log(4)$$

$$V(G, D^*) = -\log(4) + \underbrace{KL\,(...)}_{\text{KL}\geq 0} + \underbrace{KL\,(...)}_{\geq 0}$$

Proof Structure

- To Show:

$$p_G = p_{\text{data}}$$

- Find Optimal

$$D_G^* = \frac{p_{\text{data}}}{p_{\text{data}} + p_G}$$

- Show

$$p_G = p_{\text{data}} \iff \min V(G, D^*) = -\log(4)$$

$$V(G, D^*) = -\log(4) + \underbrace{KL\left(\ldots\right)}_{\text{KL}\geq 0} + \underbrace{KL\left(\ldots\right)}_{\geq 0}$$

$$V(G, D^*) = -\log(4) + \underbrace{2 \cdot JSD\left(p_{\text{data}} | p_G\right)}_{\substack{\text{Jenson Shannon Divergence = 0} \\ \iff p_G = p_d}}$$

Pseudo Code: Iterative Training of D and G

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:

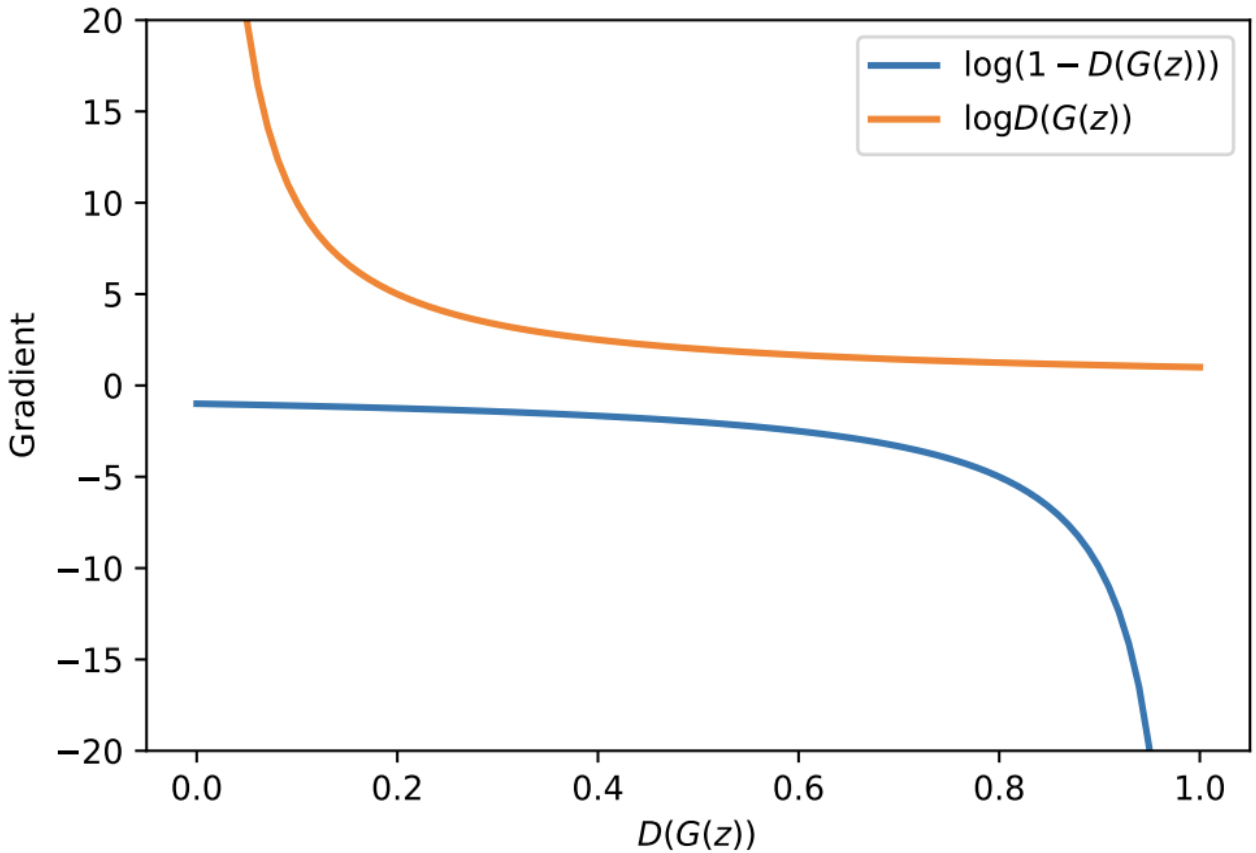
$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  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:

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

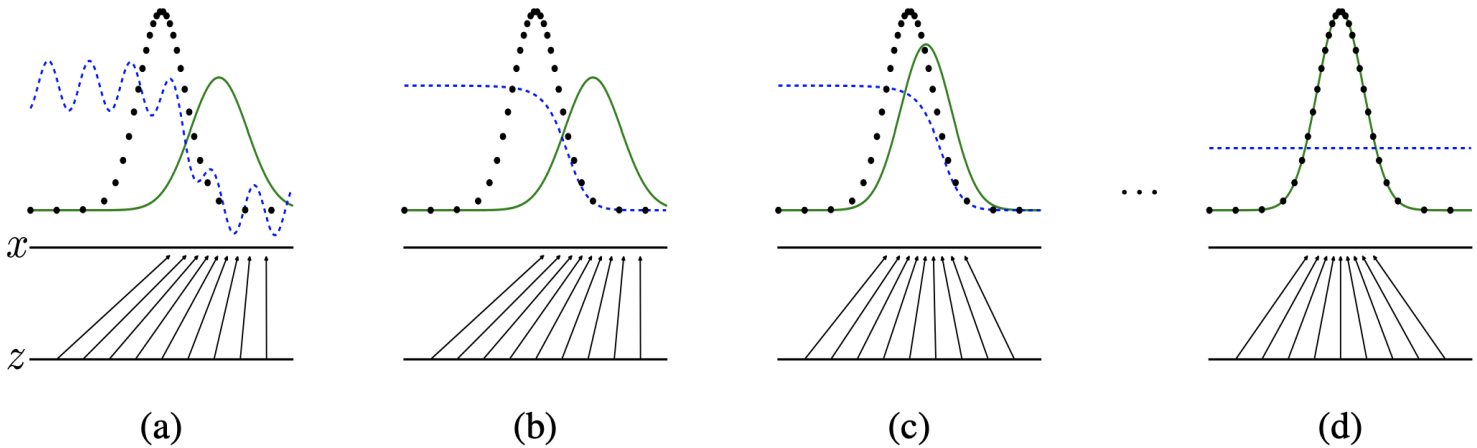
end for

Cost Functions for G



- Non-Saturating Cost Gradient: $\frac{1}{(D(G(z))}$
- Saturating Cost Gradient: $-\frac{1}{1-(D(G(z))}$

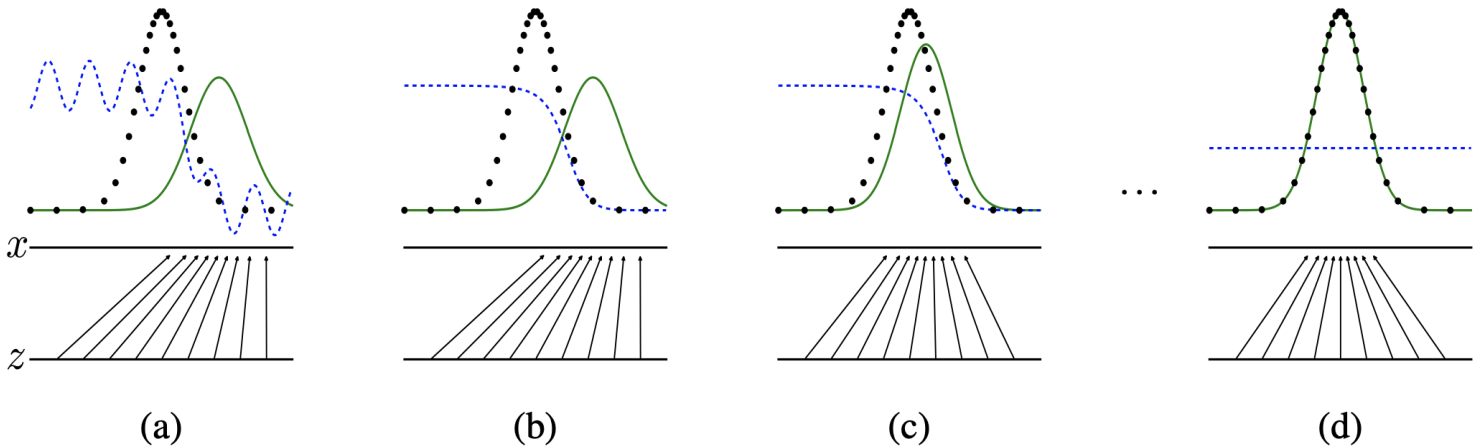
Illustration of Training Procedure - A Distribution Perspective



Iterative Improvement of Generated Probability Distirbution

- Discriminator improves from (a) to (b)

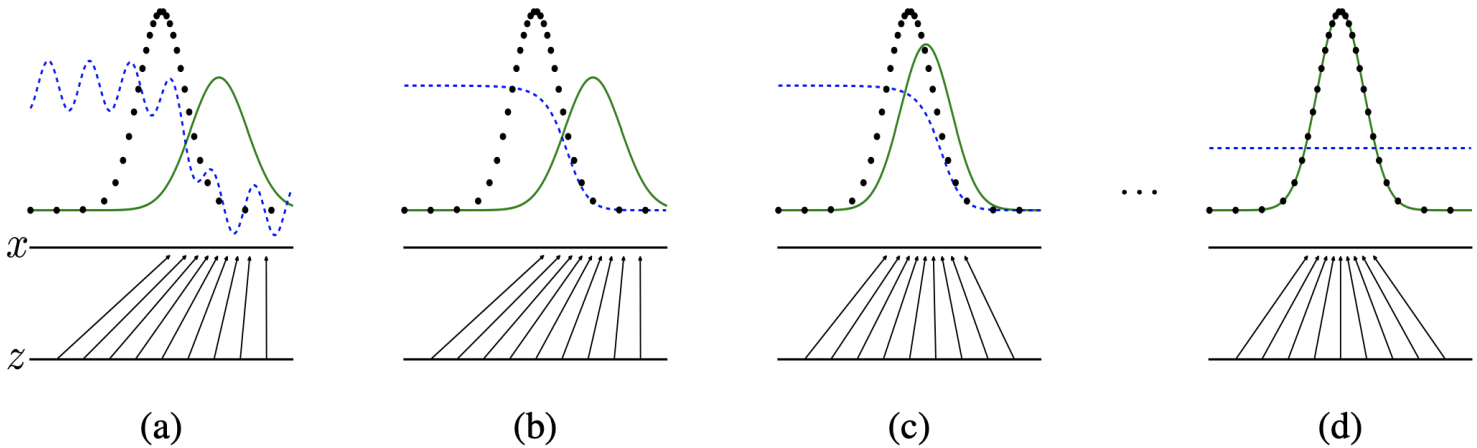
Illustration of Training Procedure - A Distribution Perspective



Iterative Improvement of Generated Probability Distirbution

- Discriminator improves from (a) to (b)
- Generator improves from (b) to (c)

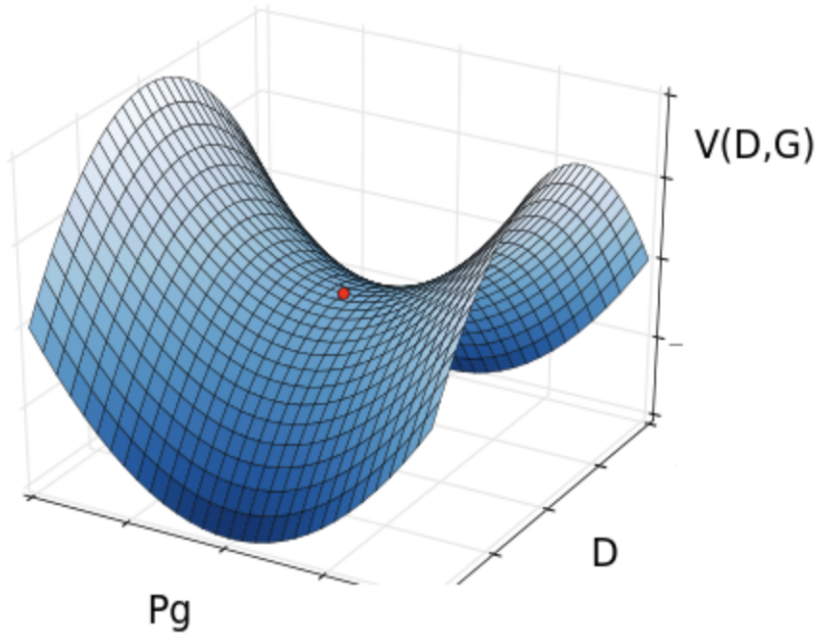
Illustration of Training Procedure - A Distribution Perspective



Iterative Improvement of Generated Probability Distirbution

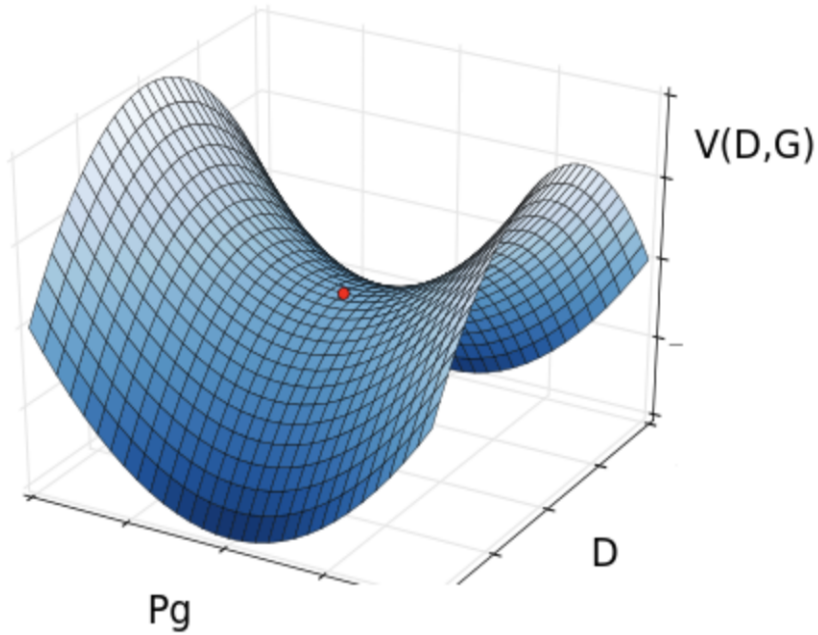
- Discriminator improves from (a) to (b)
- Generator improves from (b) to (c)
- Generator perfectly mimics the true data and the discriminator assigns probability $\frac{1}{2}$ everywhere in (d)

Illustration of Training Procedure - A Function Landscape Perspective



Function Landscape of a Minimax Game

Illustration of Training Procedure - A Function Landscape Perspective



Function Landscape of a Minimax Game

- The red dot is the equilibrium in this minimax game and can be reached with the algorithm

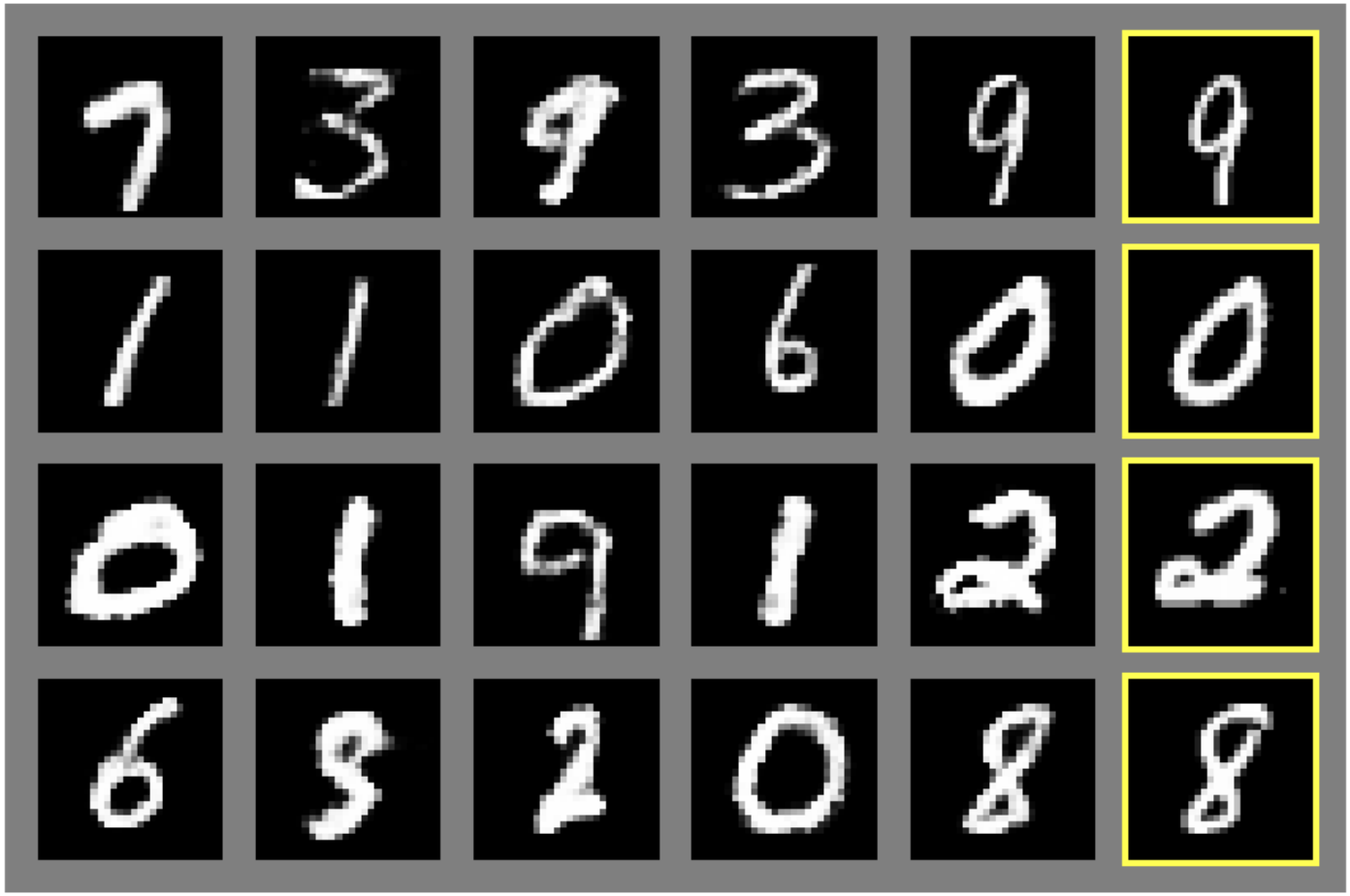
Experiments

Sources

Unless stated otherwise, all content of the experiments is based on:

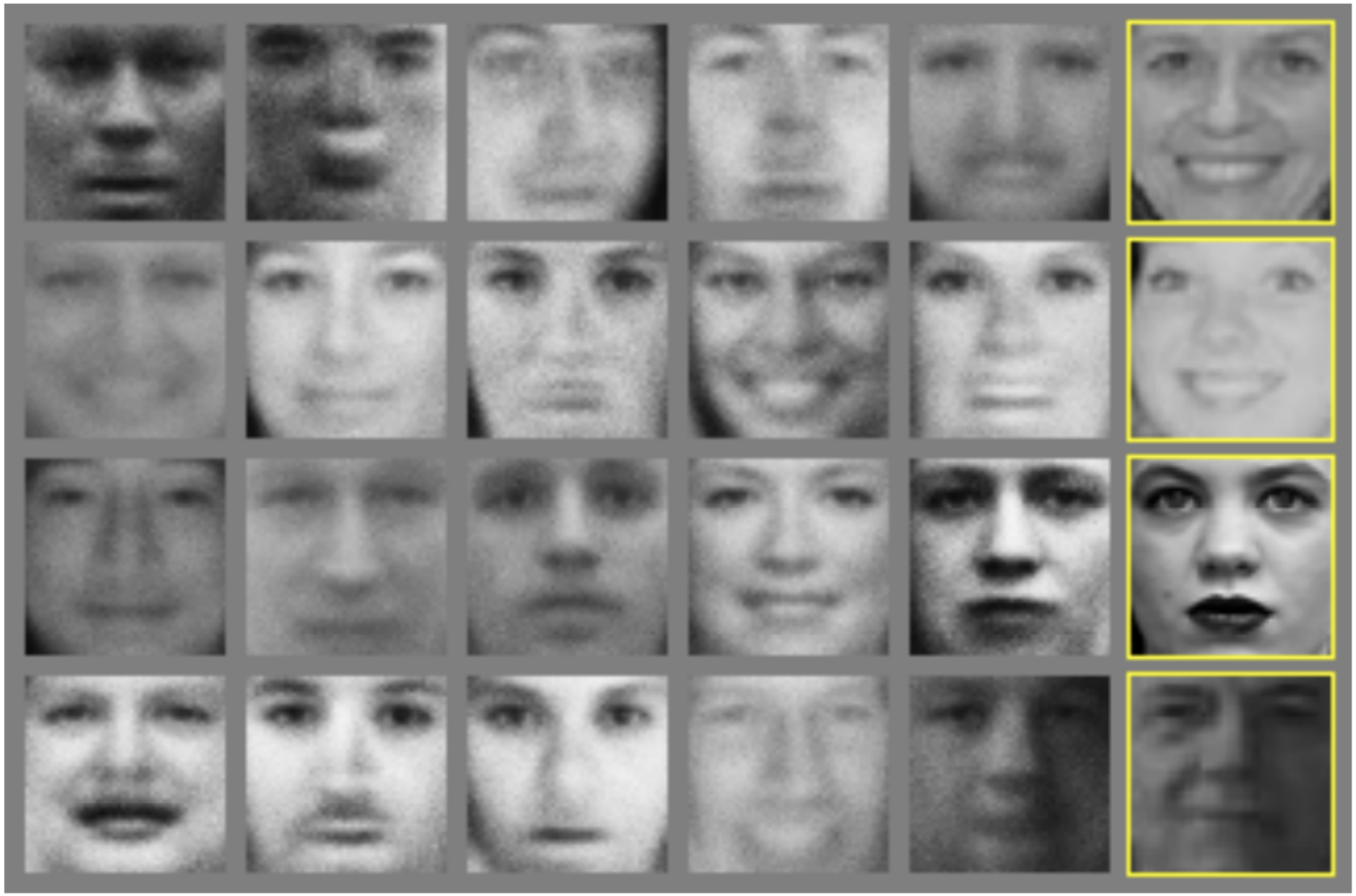
- Goodfellow, Ian, et al. "Generative adversarial nets." Advances in neural information processing systems. 2014.

GANs Can Recover the MNIST Dataset



MNIST Samples with most similar training sample

GANs Recover More Complex Distributions



TFD Samples with most similar training sample

Performance of GANs: Quantified and Compared

Model	MNIST	TFD
DBN	138 ± 2	1909 ± 66
Stacked CAE	121 ± 1.6	2110 ± 50
Deep GSN	214 ± 1.1	1890 ± 29
Adversarial nets	225 ± 2	2057 ± 26

Table: Log-Likelihood Estimates

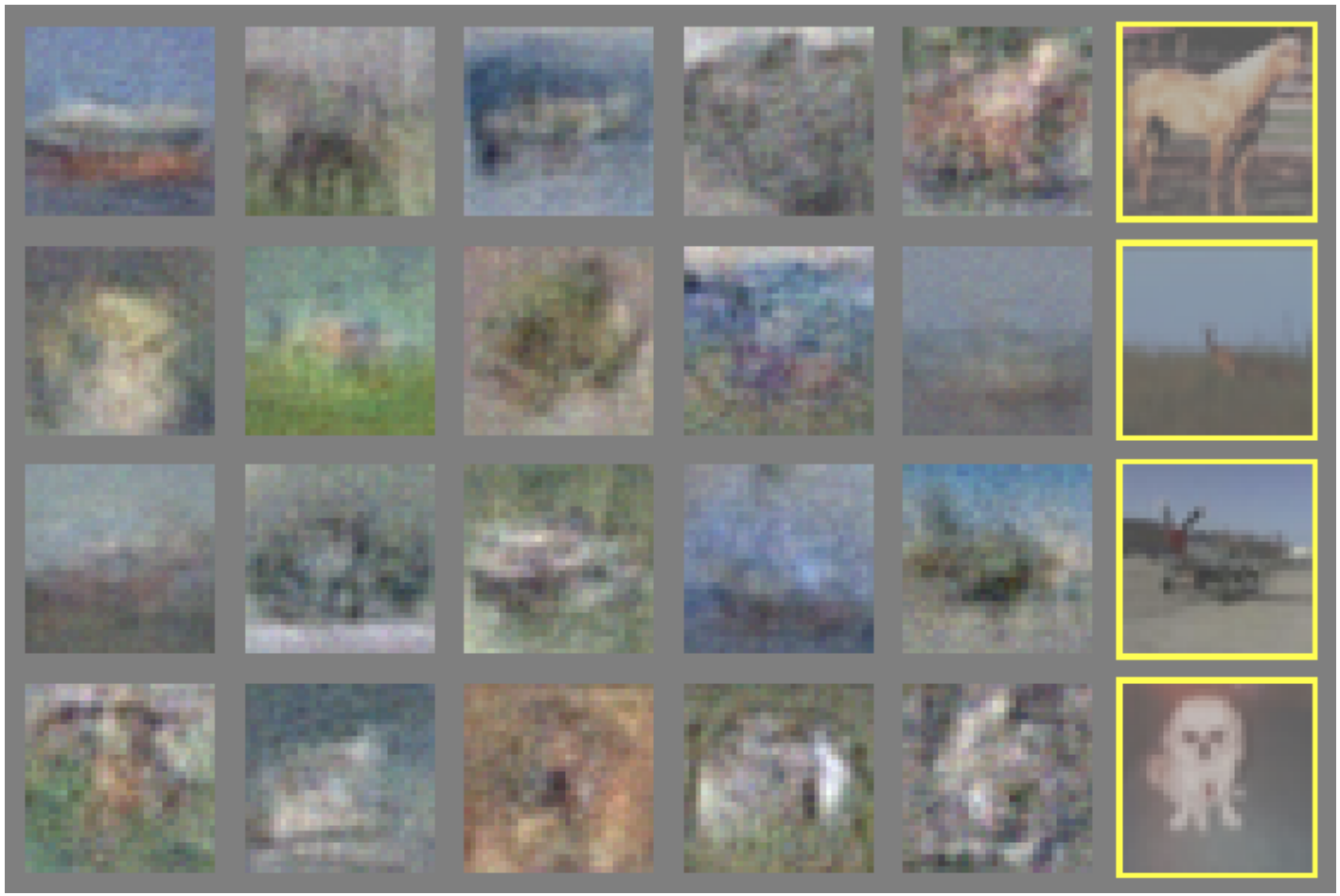
Performance of GANs: Quantified and Compared

Model	MNIST	TFD
DBN	138 ± 2	1909 ± 66
Stacked CAE	121 ± 1.6	2110 ± 50
Deep GSN	214 ± 1.1	1890 ± 29
Adversarial nets	225 ± 2	2057 ± 26

Table: Log-Likelihood Estimates

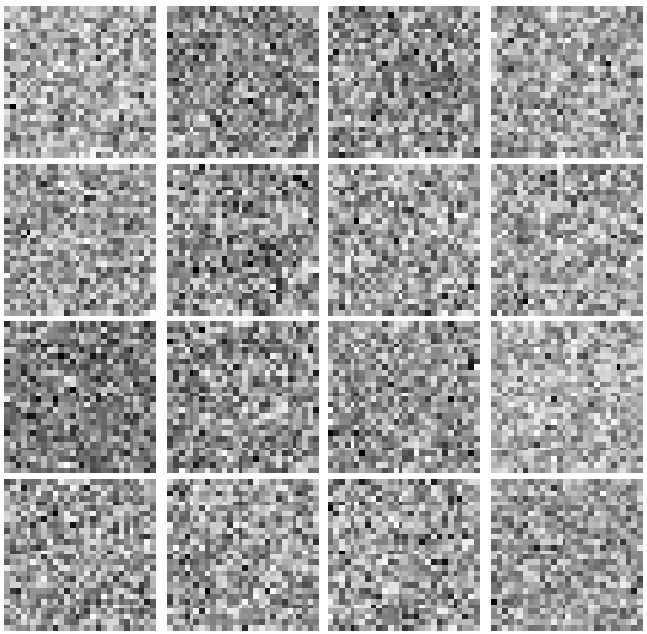
GANs: Almighty!?

Almighty? No!

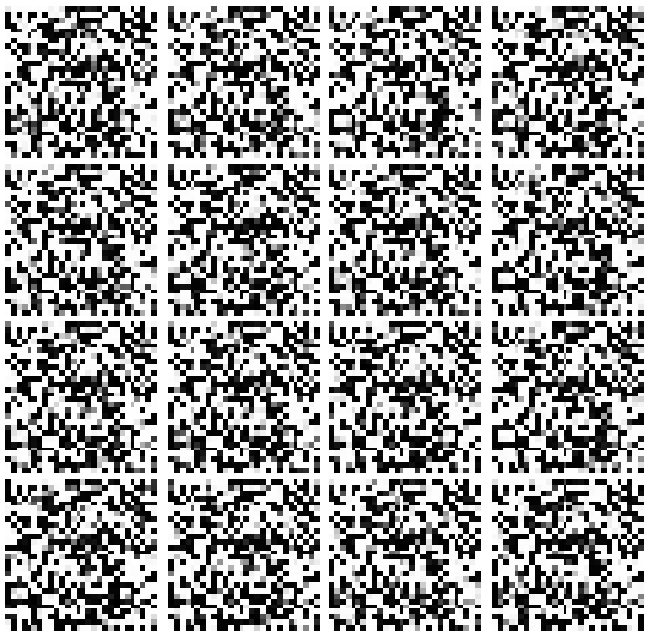


CIFAR-10 Samples with most similar Training Sample

Easy-to-Use? **No!**

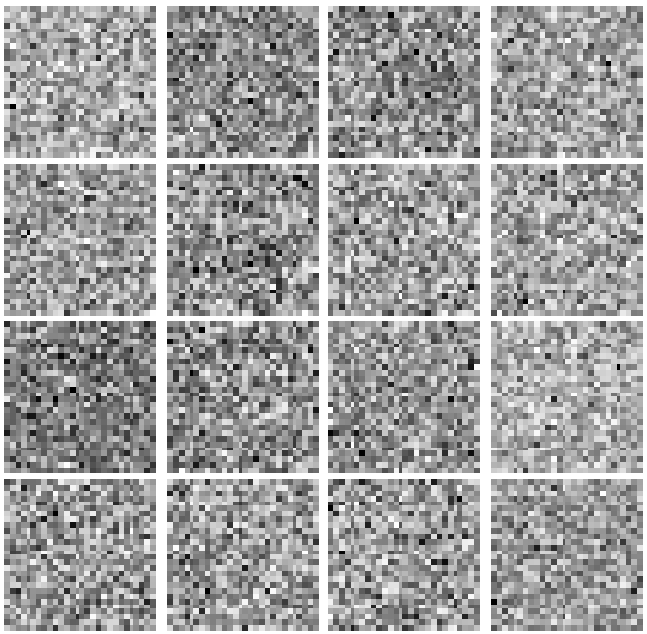


Input Data

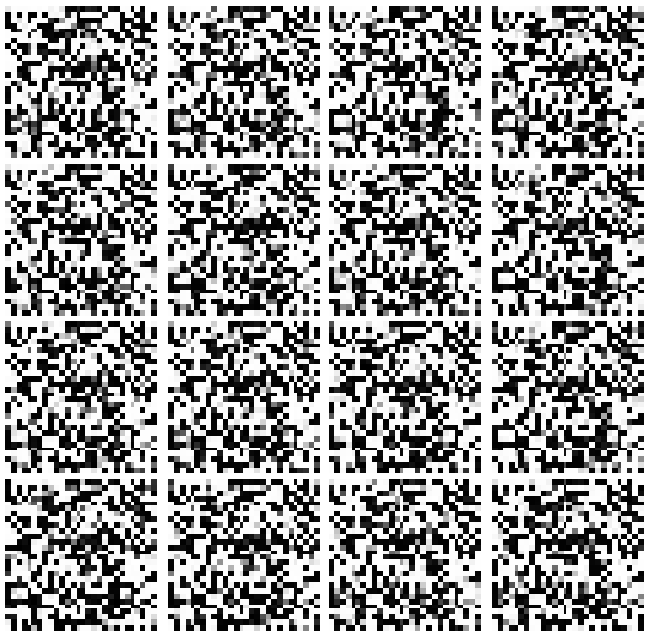


after 2h

Easy-to-Use? **No!**



Input Data



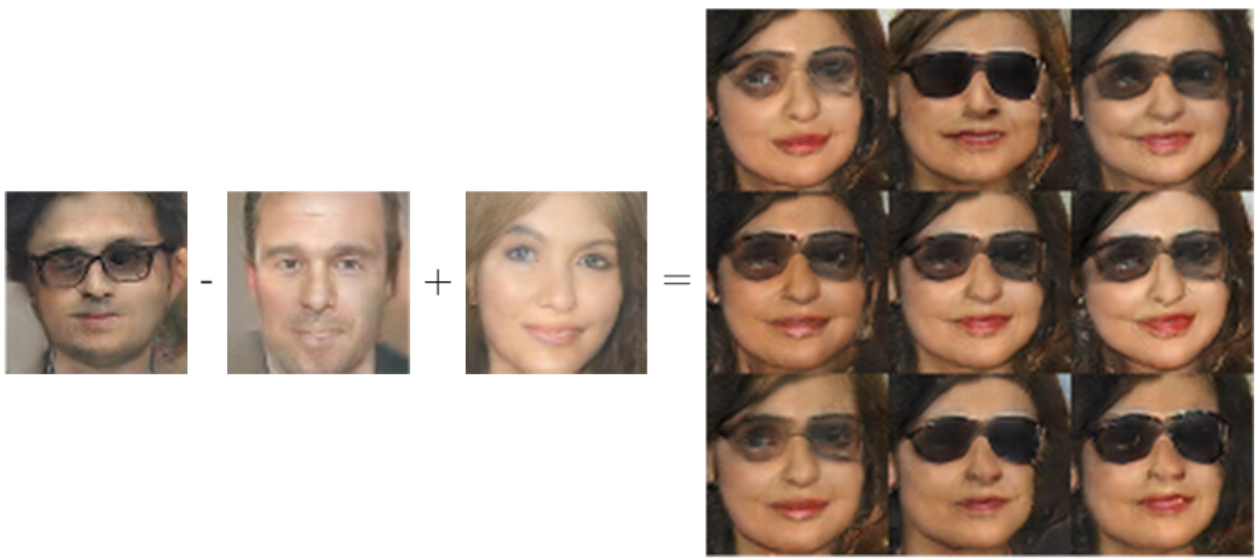
after 2h

"There are so many other problems with GANs that in academia (at least in Harvard), there is a running joke that if you want to train a GAN, you pick an unsuspecting and naive graduate student to do it for you[...]." (Matthew Stewart on towardsdatascience.com)

Advantages and Disadvantages

Advantages

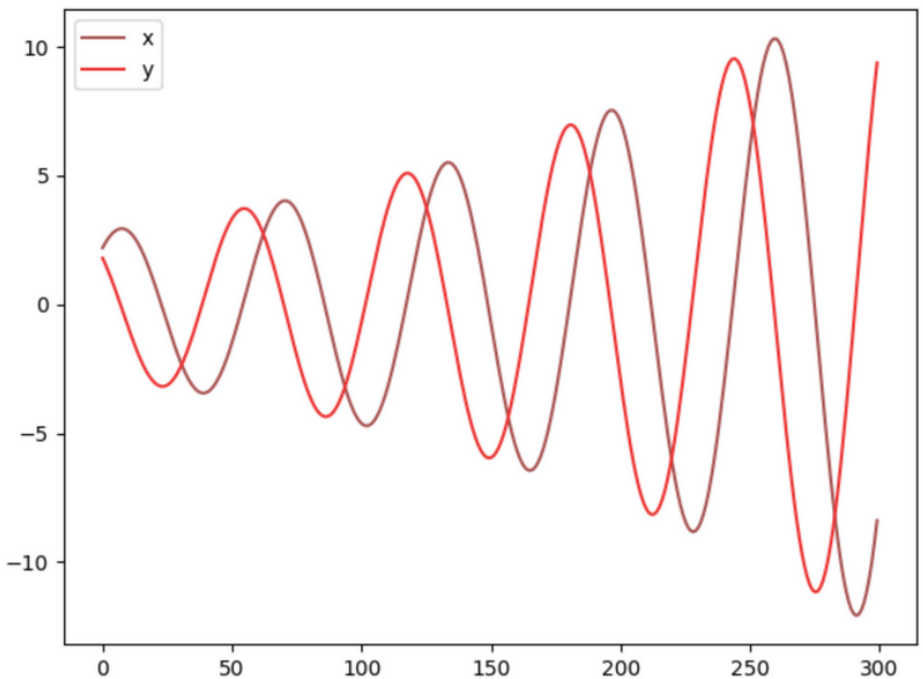
- Only backpropagation is used to obtain gradients
- No Markov chains
- No inference
- Components of the input are not copied directly into the generator's parameters.
- 'Calculation' with images
- Not only for pictures
  - Sound-Example



Source

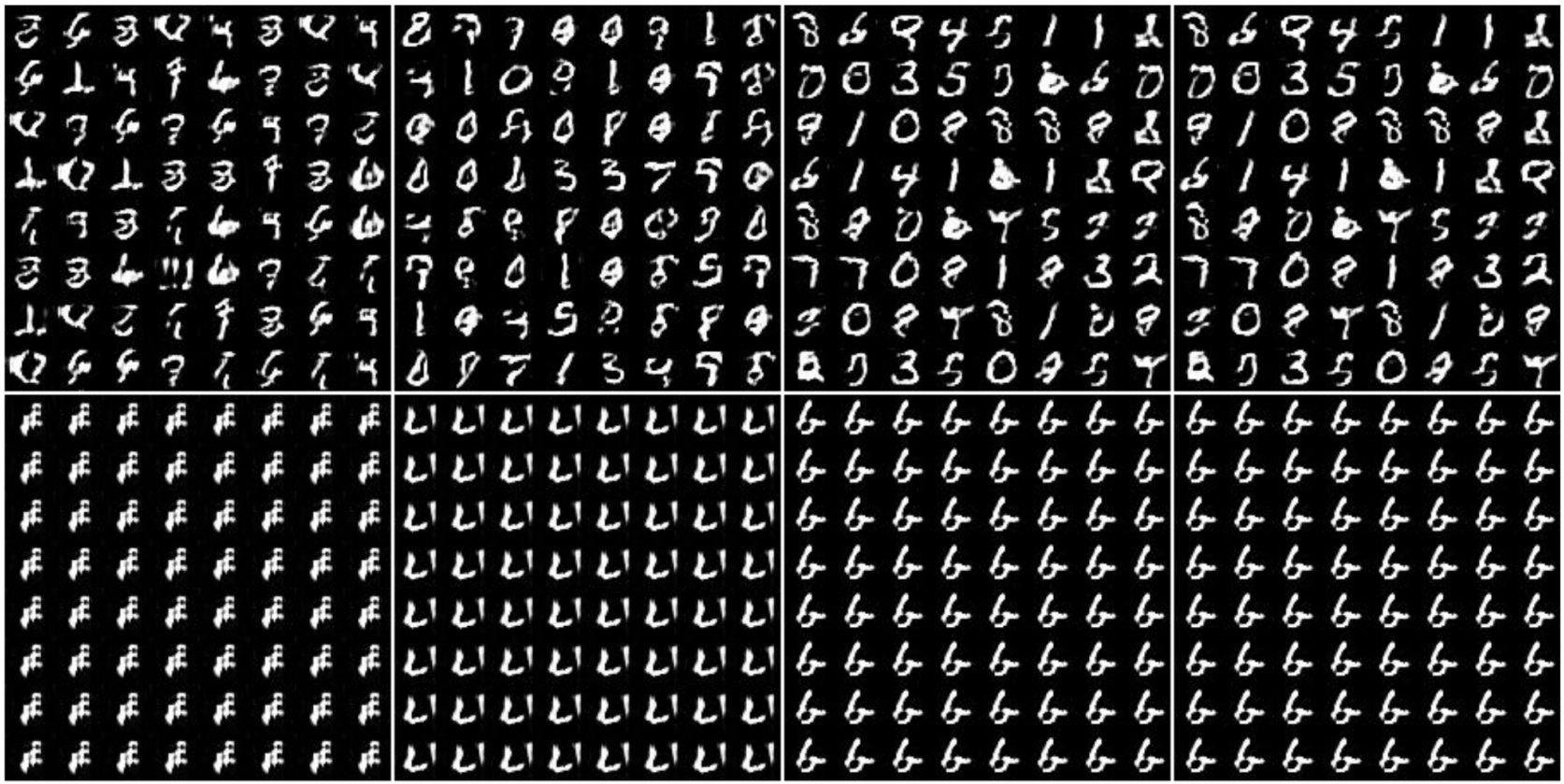
Disadvantages

- No explicit representation of generator distribution p_G
- Hard to train
  - The discriminator and the generator must be 'balanced'
  - Non-Convergence
    - $\min_B \max_A V(D, G) = xy$
    - $\Delta x = \alpha \frac{\delta(xy)}{\delta(x)}, \Delta y = -\alpha \frac{\delta(xy)}{\delta(y)}$



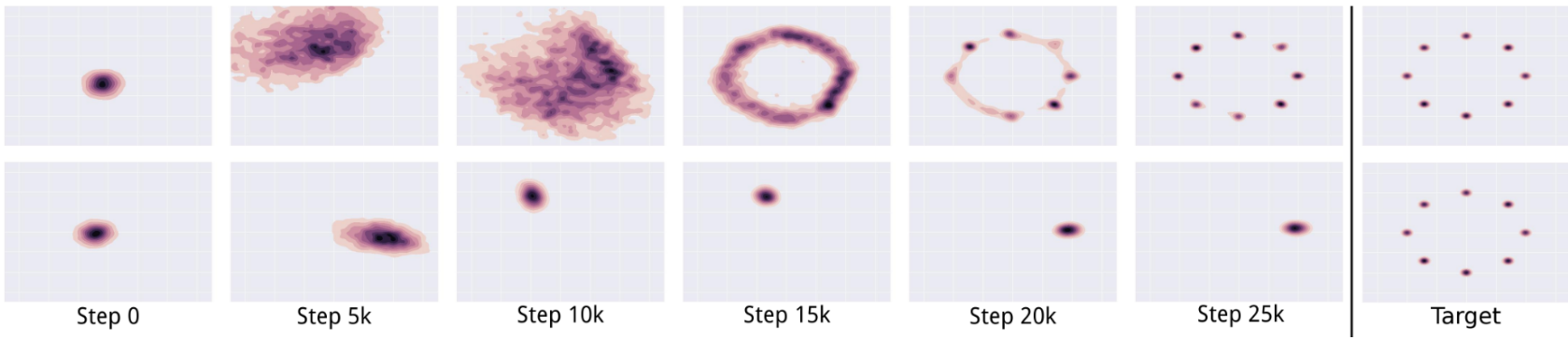
Source

Bad Balancing - Mode Collapse



Source

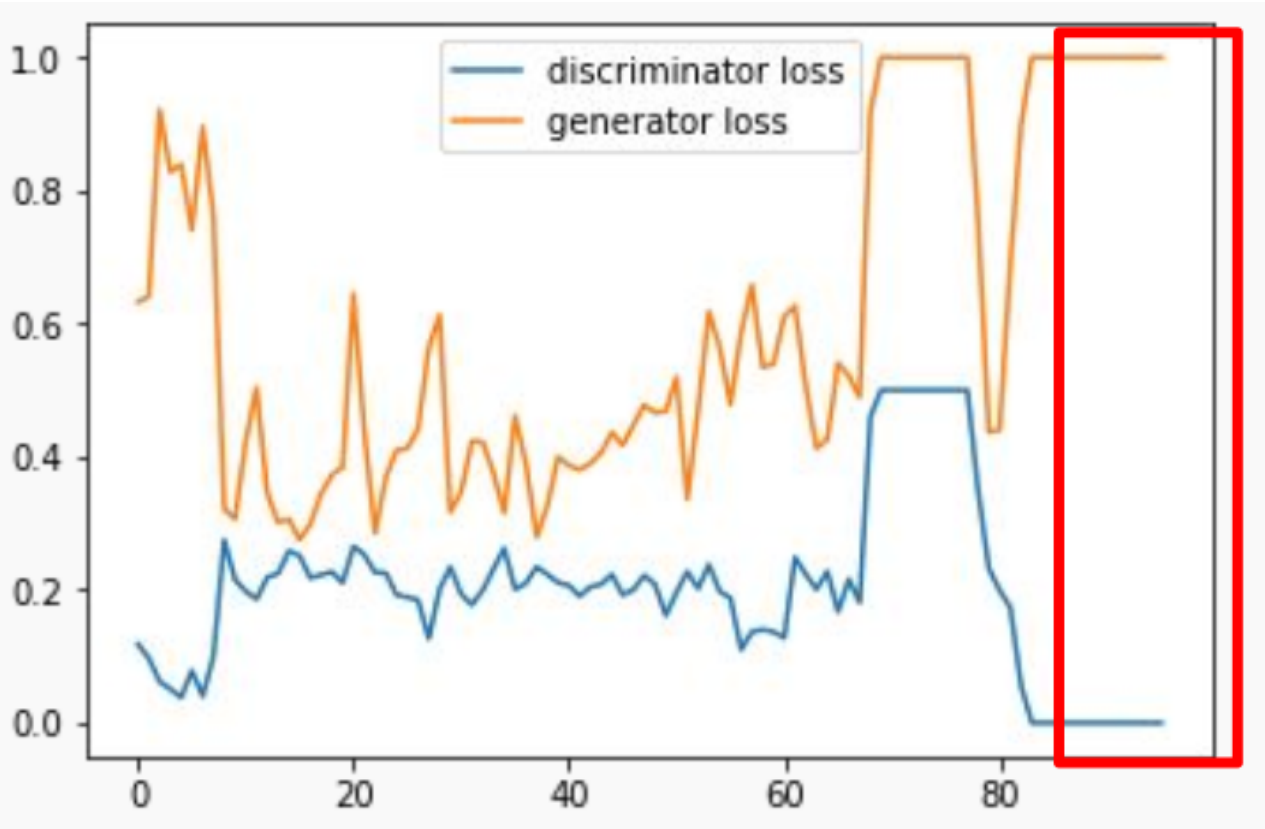
Bad Balancing - Mode Collapse



Source

Bad Balancing - Vanishing Gradients

- Discriminator will get too strong
- Generator never knows success
- Generator will produce bad images



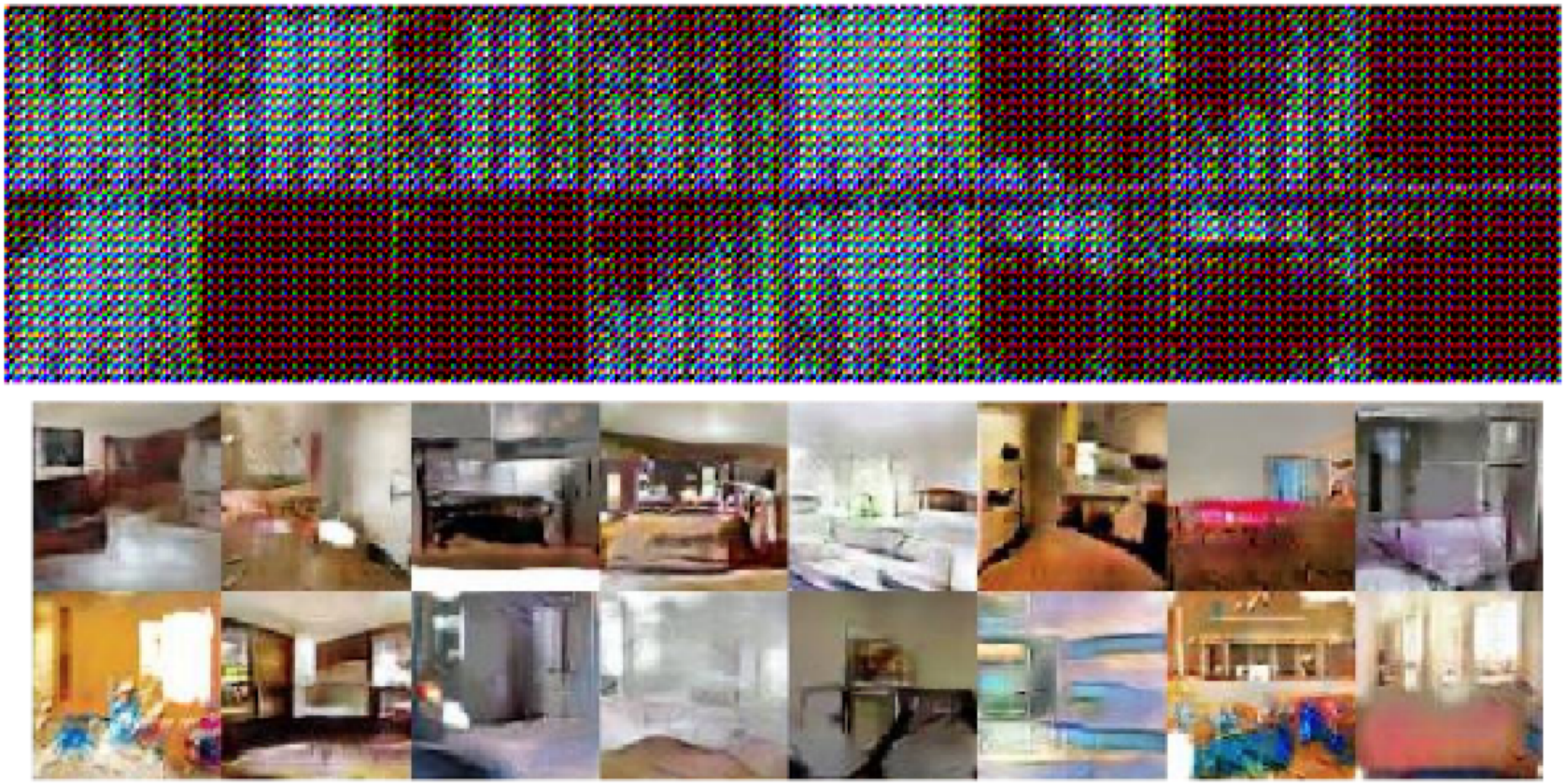
Source

State of the Art Frameworks and Examples

WGANs

- Wasserstein GAN
- New loss function using Wasserstein-1 distance
  $W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \prod(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}[\, ||x - y||\,]$
- Improve the stability of learning
- Get rid of problems like mode collapse and and vanishing gradients
- Provide meaningful learning curves useful for debugging

WGANs

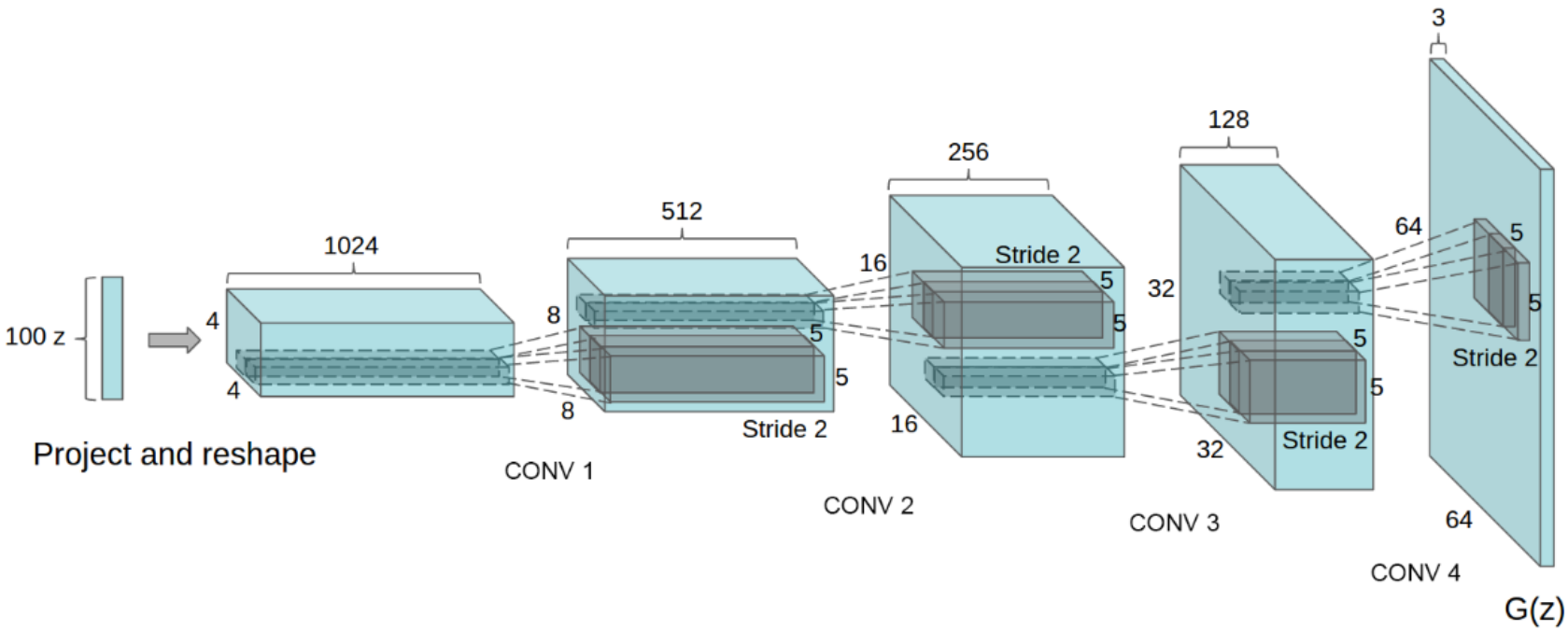


Source

DCGANs

- Deep Convolutional GAN
- Apply the architecture of CNNs on GAN
- Higher stability of learning

DCGANs

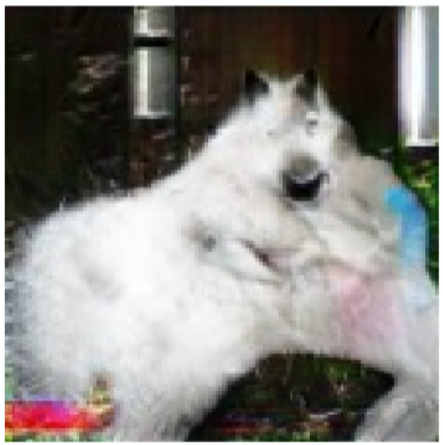


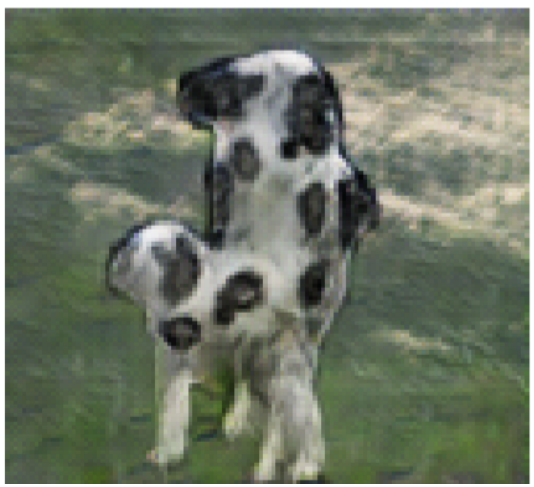
Source

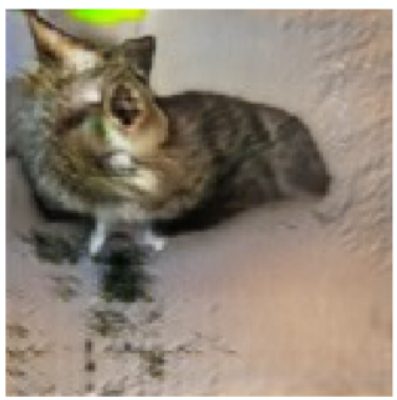
SAGANs

- Self-Attention GAN
- Build on top of the DCGANs
- Use a Self-Attention-Mechanism
  - To model long range dependencies across the image
- Good for structures and geometry

SAGANs

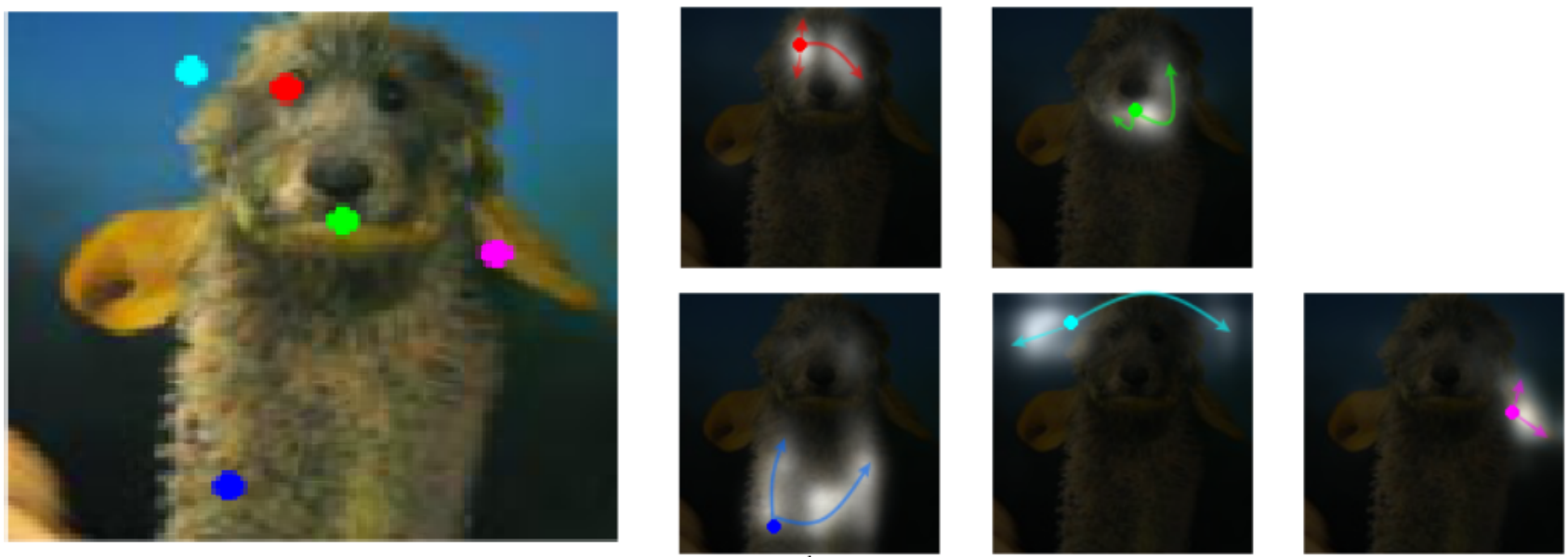




Source

SAGANs



Source

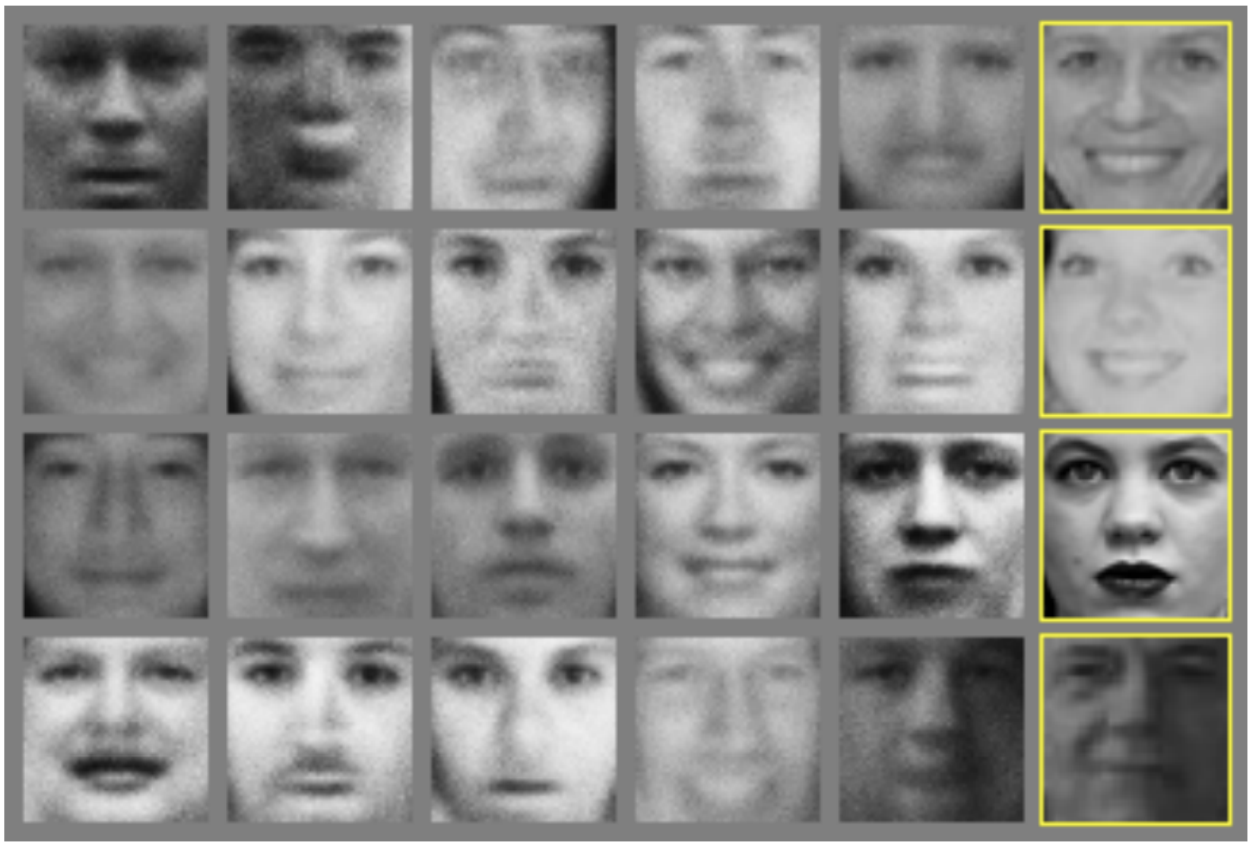
SAGANs

Source

BigGANs

- Original GANs: image resolution up to 64x64 or 128x128
- Scale up the GAN with some design choices, techniques and tricks:
  - Use SAGANs
  - Update discriminator more than generator
  - Truncation trick
  - More model parameters
  - Orthogonal regularization
- Result: Images with resolution of 256x256 and higher

BigGANs



Source

BigGANs

(a) 128×128

(b) 256×256

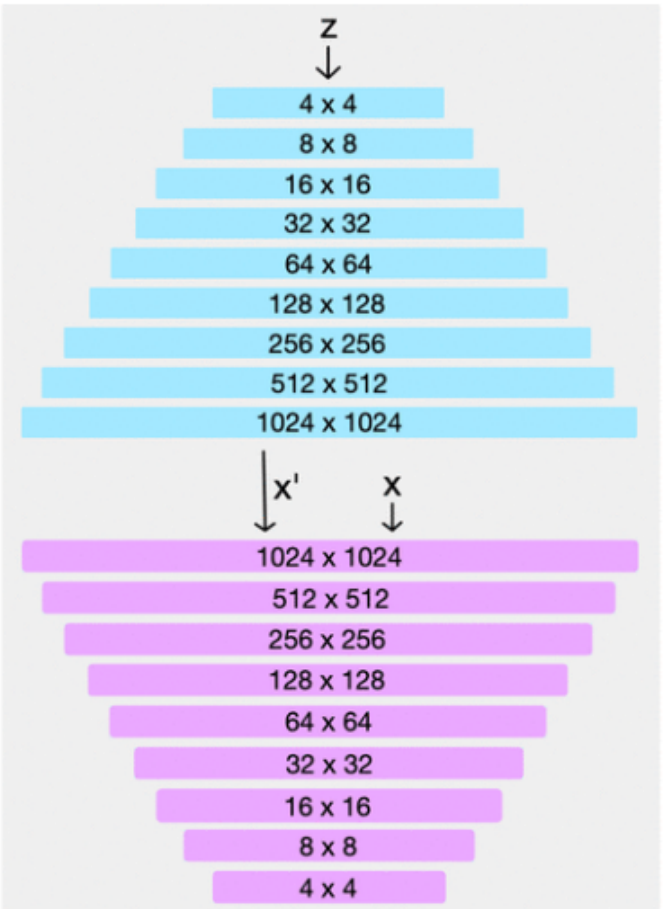
(c) 512×512

Source

Growing GANs

- Changes it's structure while training
- Training contains multiple steps:
  - Reduce the original training data to 4x4
  - Train the GAN until a certain stability is reached
  - Add a new, bigger layer to the GAN
  - Reduce the original training data to 8x8
  - Repeat until original training data resolution is reached
- Training will take a while
- Much more data is seen

Growing GANs



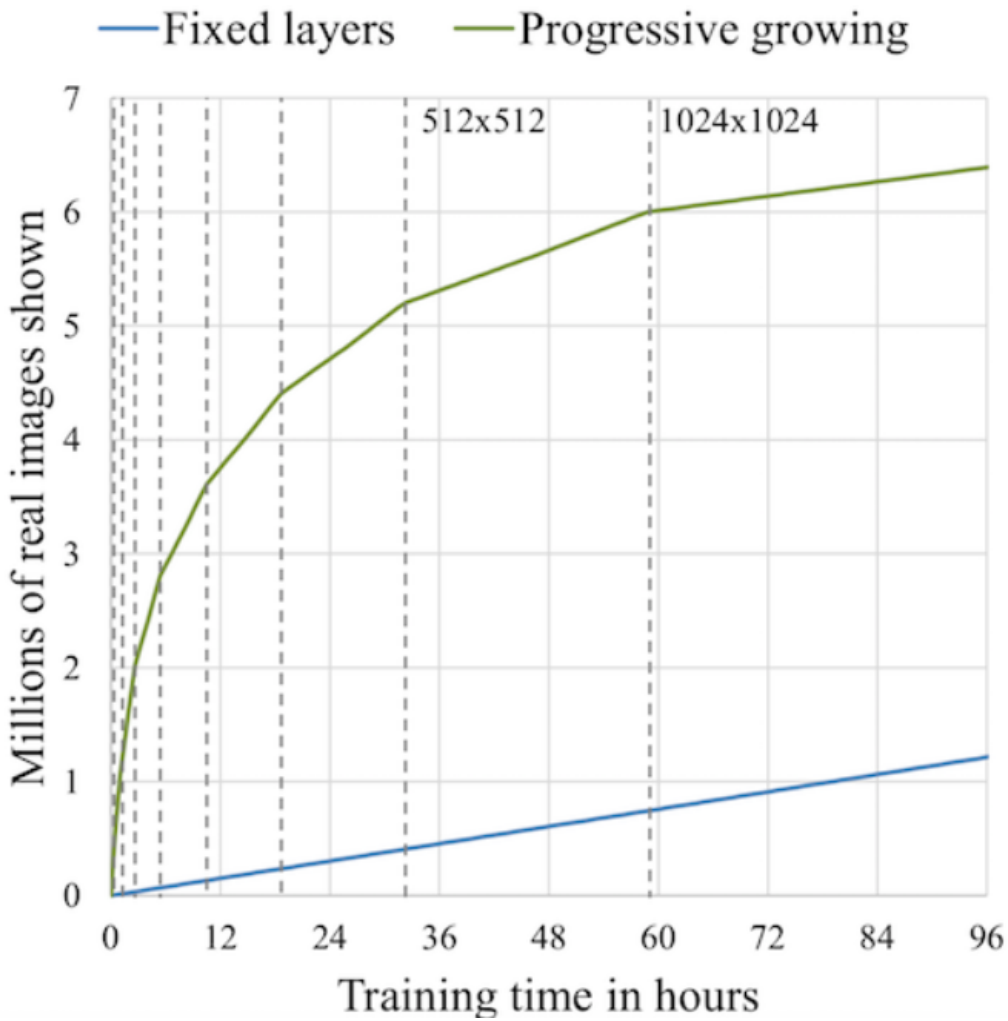
Training time: 13 days
1024x1024 resolution

Generator
Discriminator

z = random code
x = real image
x' = generated image

Source

Growing GANs



Source

Growing GANs

- miro.medium.com/max/586/1*luI7A7HmZpqPUjX04mGavA.gif
- thispersondoesnotexist.com/

More GANs

- DCGAN
- WGAN
- CGAN
- LAPGAN
- SRGAN
- CycleGAN
- WGAN-GP
- EBGAN
- VAE-GAN
- BiGAN
- SGAN
- SimGAN
- VGAN
- iGAN
- 3D-GAN
- CoGAN
- Cat-GAN
- MGAN
- S∧2GAN
- LSGAN
- AffGAN
- TP-GAN
- IcGAN
- ID-CGAN
- AnoGAN
- LS-GAN
- Triple-GAN
- TGAN
- BS-GAN
- MalGAN
- RTTGAN
- GANCS
- SSL-GAN
- MAD-GAN
- PrGAN
- AL-CGAN
- ORGAN
- SD-GAN
- MedGAN
- SGAN
- SL-GAN
- SketchGAN
- GoGAN
- RWGAN
- MPM-GAN
- MV-GAN
- StyleGAN
- GANSynth
- ProGAN
- Context-RNN-GAN

Discussion

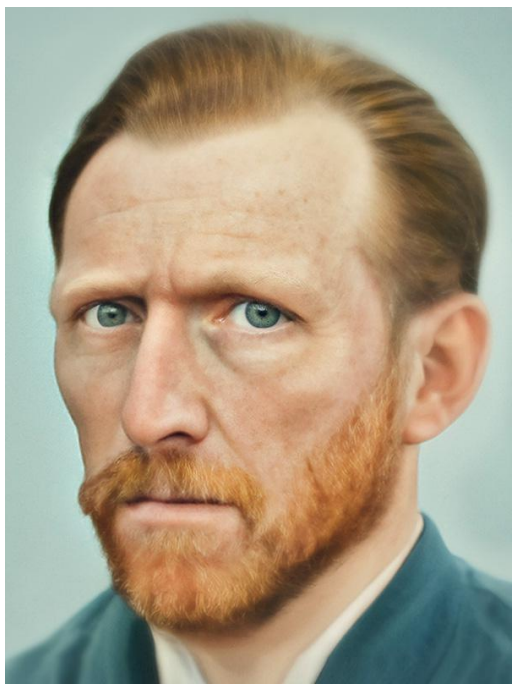
Discussion

In a few years: everyone might be able to produce photo realistic images from their home computer!

This is great for:

- Artists
- Filmmakers
- And all other creative industries!

Sounds fantastic!
Where is the drawback?



Van Gogh by Reddit user basu68

Deepfakes

- GANs or Autoencoders that can recreate faces of specific people
- **Input Dataset:** Videos and photos of target person
- **Output:** Realistic video of target person, based on reference movement.

Trump deepfake: https://www.youtube.com/watch?v=hoc2RISoLWU
The Shining deepfake: https://www.youtube.com/watch?v=AeRofGJ17Sk

Deepfakes

These generated images can (and have been) used for:

- Fake political videos
- Pornographic videos of celebreties
- Pornographic videos of random people from the internet!

If deepfakes get too good, we cannot trust any video to be real anymore!

And on the flipside: e.g. politicians could pretend that real videos are deepfakes

Questions to Discuss

General:

- Are there other kinds of data you would be interested in generating?

Deepfakes:

- What other problems are there with people being able to generate realistic fake images and video?
- Will GANs or other techniques ever even produce perfect pictures?
- How can we fight / prevent deepfakes?
  Through machine learning? Laws?