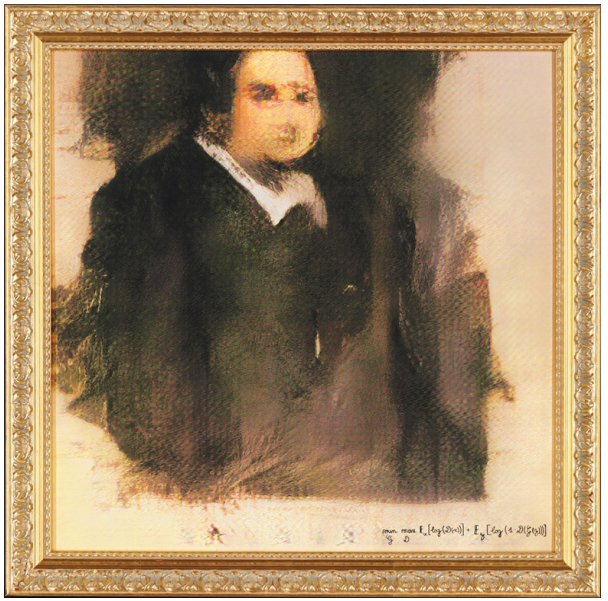
Deep Learning

Lecture 10: Generative adversarial networks

Prof. Gilles Louppe
g.louppe@uliege.be

LIÈGE université



"Generative adversarial networks is the coolest idea in deep learning in the last 20 years."

Yann LeCun, 2018.

Today

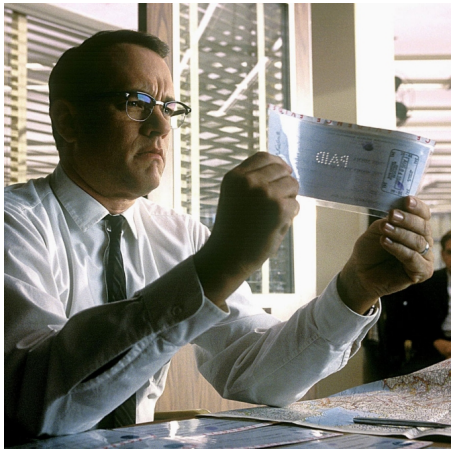
Learn a model of the data.

- Generative adversarial networks
- Wasserstein GANs
- Numerics of GANs
- State of the art
- Applications

Generative adversarial networks

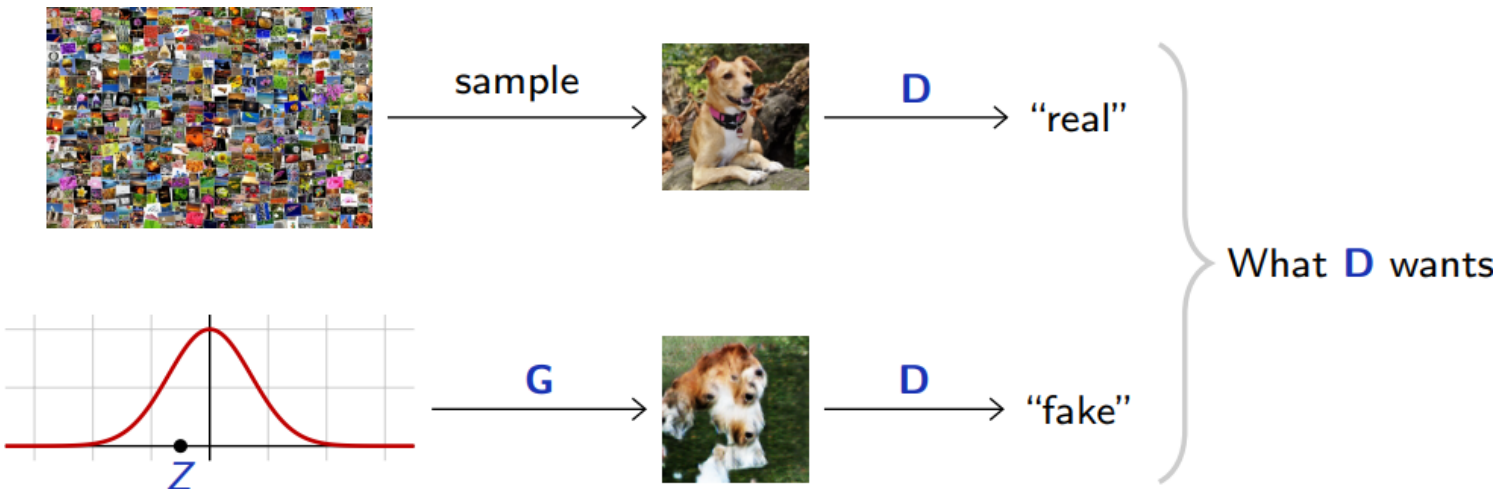

A STEVEN SPIELBERG FILM
catch me
if you can


PAID



A two-player game

In generative adversarial networks (GANs), the task of learning a generative model is expressed as a two-player zero-sum game between two networks.



The first network is a generator $g(\cdot;\theta) : \mathcal{Z} \to \mathcal{X}$, mapping a latent space equipped with a prior distribution $p(\mathbf{z})$ to the data space, thereby inducing a distribution

$$\mathbf{x} \sim q(\mathbf{x};\theta) \Leftrightarrow \mathbf{z} \sim p(\mathbf{z}), \mathbf{x} = g(\mathbf{z};\theta).$$

The second network $d(\cdot;\phi) : \mathcal{X} \to [0,1]$ is a classifier trained to distinguish between true samples $\mathbf{x} \sim p(\mathbf{x})$ and generated samples $\mathbf{x} \sim q(\mathbf{x};\theta)$.

For a fixed generator g, the classifier d can be trained by generating a two-class training set

$$\mathbf{d} = \{(\mathbf{x}_1, y = 1), ..., (\mathbf{x}_N, y = 1), (g(\mathbf{z}_1; \theta), y = 0), ..., (g(\mathbf{z}_N; \theta), y = 0)\},$$

where $\mathbf{x}_i \sim p(\mathbf{x})$ and $\mathbf{z}_i \sim p(\mathbf{z})$, and minimizing the cross-entropy loss

$$\begin{aligned}\mathcal{L}(\phi) &= -\frac{1}{2N} \sum_{i=1}^{N} \left[\log d(\mathbf{x}_i; \phi) + \log\left(1 - d(g(\mathbf{z}_i; \theta); \phi)\right)\right] \\ &\approx -\mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})}\left[\log d(\mathbf{x}; \phi)\right] - \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})}\left[\log(1 - d(g(\mathbf{z}; \theta); \phi))\right].\end{aligned}$$

However, the situation is slightly more complicated since we also want to train g to fool the discriminator, which is equivalent to maximize d's loss.

Game analysis

Let us consider the value function

$$V(\phi, \theta) = \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} \left[\log d(\mathbf{x}; \phi) \right] + \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})} \left[\log(1 - d(g(\mathbf{z}; \theta); \phi)) \right].$$

- For a fixed g, $V(\phi, \theta)$ is high if d is good at recognizing true from generated samples.
- If d is the best classifier given g, and if V is high, then this implies that the generator is bad at reproducing the data distribution.
- Conversely, g will be a good generative model if V is low when d is a perfect opponent.

Therefore, the ultimate goal is

$$\theta^* = \arg \min_{\theta} \max_{\phi} V(\phi, \theta).$$

For a generator g fixed at θ, the classifier d with parameters ϕ_θ^* is optimal if and only if

$$\forall \mathbf{x}, d(\mathbf{x}; \phi_\theta^*) = \frac{p(\mathbf{x})}{q(\mathbf{x}; \theta) + p(\mathbf{x})}.$$

Therefore,

$$
\begin{aligned}
&\min_{\theta} \max_{\phi} V(\phi, \theta) = \min_{\theta} V(\phi^*_\theta, \theta) \\
&= \min_{\theta} \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} \left[\log \frac{p(\mathbf{x})}{q(\mathbf{x}; \theta) + p(\mathbf{x})} \right] + \mathbb{E}_{\mathbf{x} \sim q(\mathbf{x}; \theta)} \left[\log \frac{q(\mathbf{x}; \theta)}{q(\mathbf{x}; \theta) + p(\mathbf{x})} \right] \\
&= \min_{\theta} \mathrm{KL} \left(p(\mathbf{x}) || \frac{p(\mathbf{x}) + q(\mathbf{x}; \theta)}{2} \right) \\
&\qquad + \mathrm{KL} \left(q(\mathbf{x}; \theta) || \frac{p(\mathbf{x}) + q(\mathbf{x}; \theta)}{2} \right) - \log 4 \\
&= \min_{\theta} 2 \, \mathrm{JSD}(p(\mathbf{x}) || q(\mathbf{x}; \theta)) - \log 4
\end{aligned}
$$

where JSD is the Jensen-Shannon divergence.

In summary,

$$\begin{aligned}\theta^* &= \arg\min_\theta \max_\phi V(\phi, \theta) \\ &= \arg\min_\theta \mathrm{JSD}(p(\mathbf{x})||q(\mathbf{x};\theta)).\end{aligned}$$

Since $\mathrm{JSD}(p(\mathbf{x})||q(\mathbf{x};\theta))$ is minimum if and only if

$$p(\mathbf{x}) = q(\mathbf{x};\theta)$$

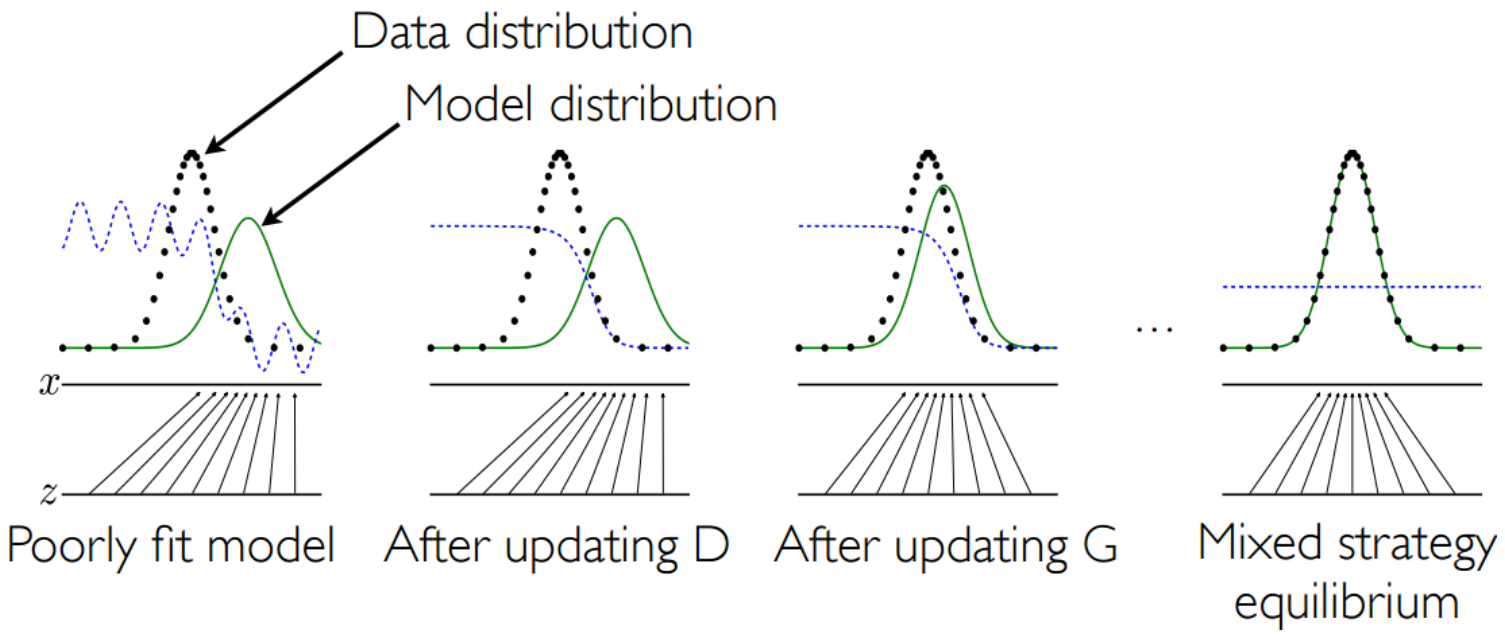
for all $\mathbf{x}$, this proves that the minimax solution corresponds to a generative model that perfectly reproduces the true data distribution.

Learning process

In practice, the minimax solution is approximated using alternating stochastic gradient descent:

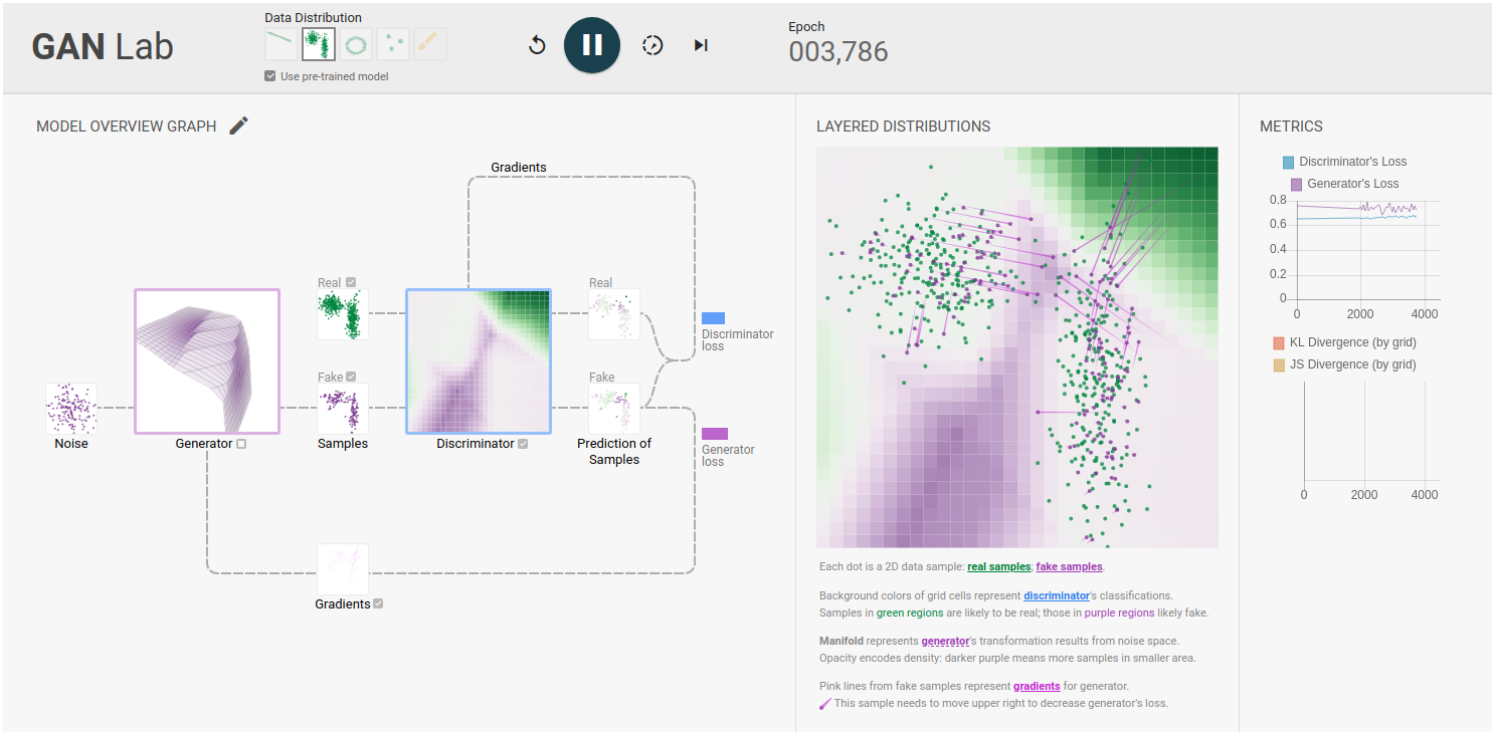
$$\theta \leftarrow \theta - \gamma \nabla_\theta V(\phi, \theta)$$
$$\phi \leftarrow \phi + \gamma \nabla_\phi V(\phi, \theta),$$

where gradients are estimated with Monte Carlo integration.



—

Credits: Goodfellow et al, Generative Adversarial Networks, 2014.

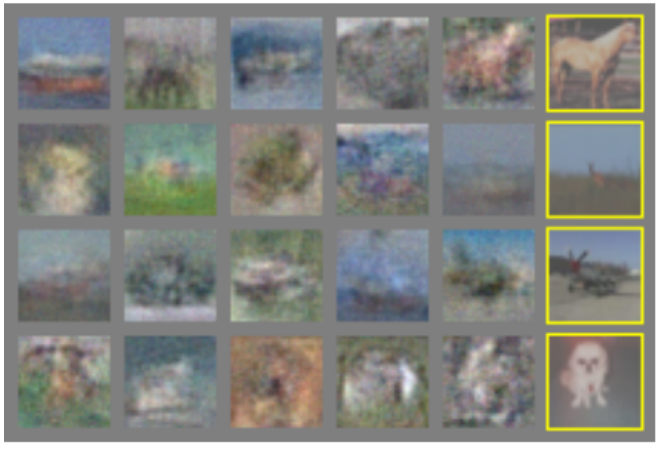

GAN Lab
Data Distribution
Use pre-trained model
Epoch
003,786
MODEL OVERVIEW GRAPH
Gradients
Real
Fake
Noise
Generator
Samples
Discriminator
Prediction of Samples
Discriminator loss
Generator loss
Gradients
LAYERED DISTRIBUTIONS
Each dot is a 2D data sample: real samples, fake samples.
Background colors of grid cells represent discriminator's classifications.
Samples in green regions are likely to be real; those in purple regions likely fake.
Manifold represents generator's transformation results from noise space.
Opacity encodes density: darker purple means more samples in smaller area.
Pink lines from fake samples represent gradients for generator.
This sample needs to move upper right to decrease generator's loss.
METRICS
Discriminator's Loss
Generator's Loss
0.8
0.6
0.4
0.2
0
0
2000
4000
KL Divergence (by grid)
JS Divergence (by grid)
0
2000
4000

[Demo]

Results

a)

b)



c)

d)

—

Credits: Goodfellow et al, Generative Adversarial Networks, 2014.

—

Credits: Radford et al, Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, 2015.

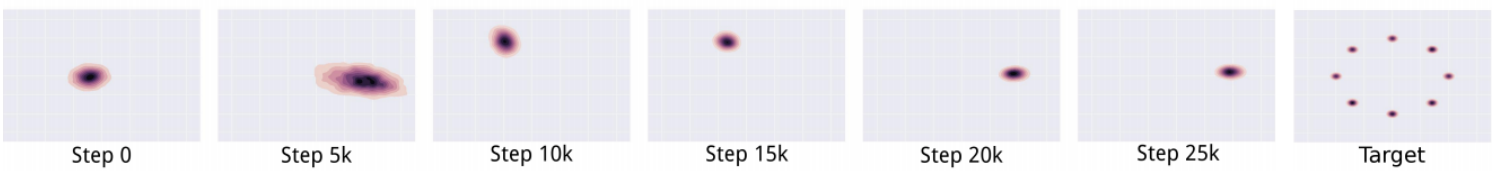
—

Credits: Radford et al, Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, 2015.

Open problems

Training a standard GAN often results in pathological behaviors:

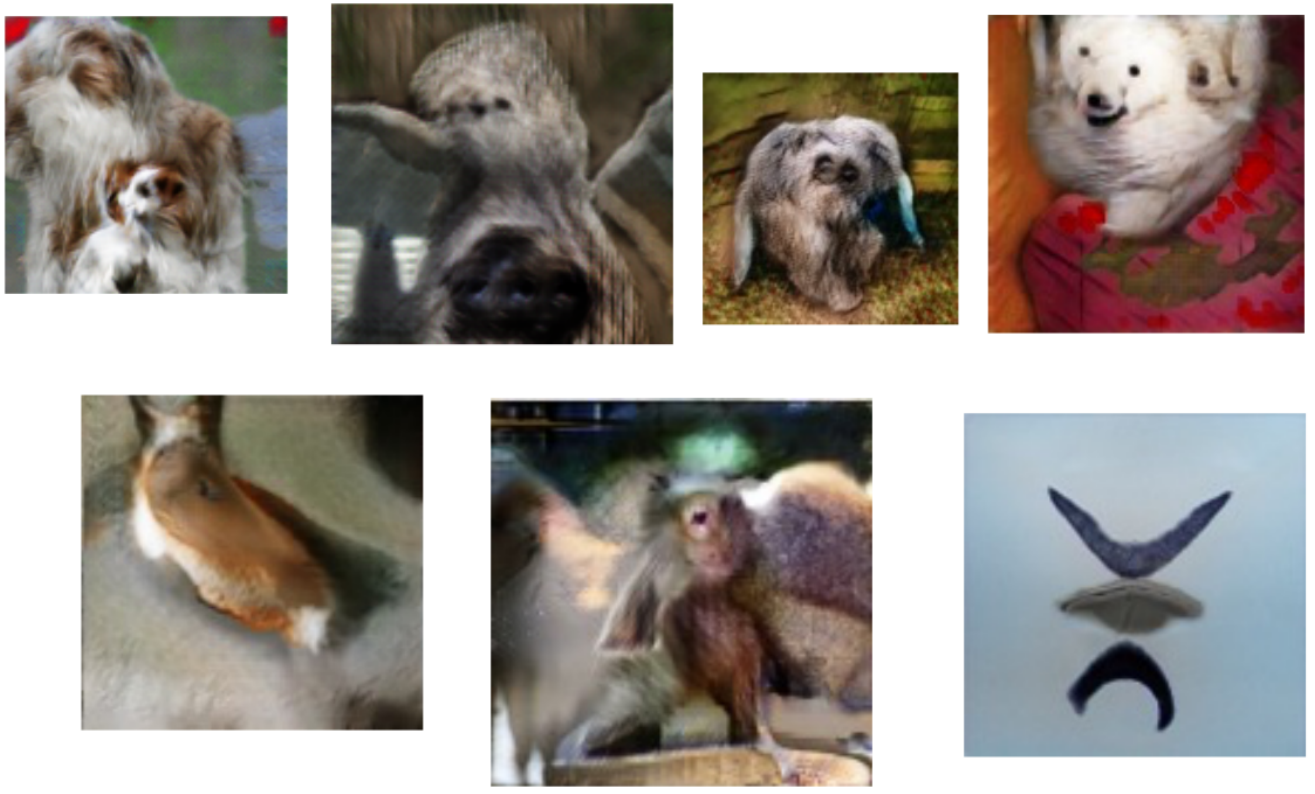
- Oscillations without convergence: contrary to standard loss minimization, alternating stochastic gradient descent has no guarantee of convergence.
- Vanishing gradients: when the classifier d is too good, the value function saturates and we end up with no gradient to update the generator.
- Mode collapse: the generator g models very well a small sub-population, concentrating on a few modes of the data distribution.
- Performance is also difficult to assess in practice.



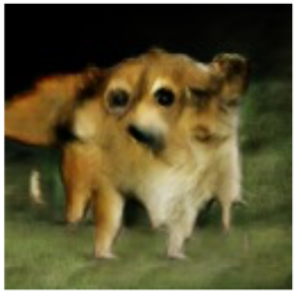
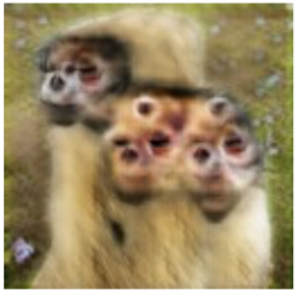
Mode collapse (Metz et al, 2016)

Cabinet of curiosities

While early results (2014-2016) were already impressive, a close inspection of the fake samples distribution $q(\mathbf{x}; \theta)$ often revealed fundamental issues highlighting architectural limitations.



Cherry-picks


Problems with counting

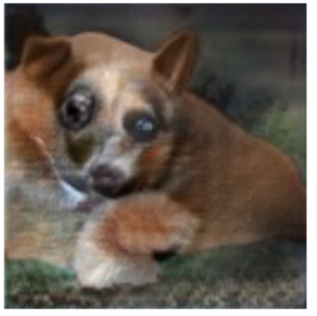
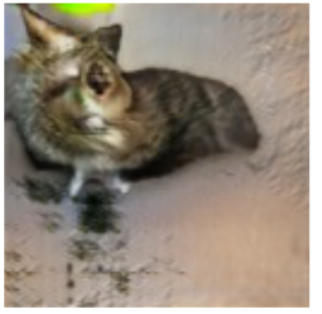
Problems with perspective

—

Credits: Ian Goodfellow, 2016.





Problems with global structures

—

Wasserstein GANs

(optional)

Return of the Vanishing Gradients

For most non-toy data distributions, the fake samples $\mathbf{x} \sim q(\mathbf{x}; \theta)$ may be so bad initially that the response of d saturates.

At the limit, when d is perfect given the current generator g,

$$d(\mathbf{x}; \phi) = 1, \forall \mathbf{x} \sim p(\mathbf{x}),$$
$$d(\mathbf{x}; \phi) = 0, \forall \mathbf{x} \sim q(\mathbf{x}; \theta).$$

Therefore,

$$V(\phi, \theta) = \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} \left[\log d(\mathbf{x}; \phi)\right] + \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})} \left[\log(1 - d(g(\mathbf{z}; \theta); \phi))\right] = 0$$

and $\nabla_\theta V(\phi, \theta) = 0$, thereby halting gradient descent.

Dilemma :

- If d is bad, then g does not have accurate feedback and the loss function cannot represent the reality.
- If d is too good, the gradients drop to 0, thereby slowing down or even halting the optimization.

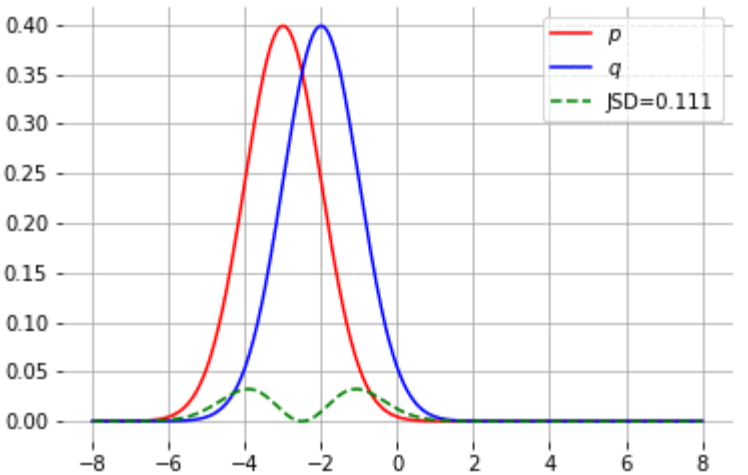
Jensen-Shannon divergence

For any two distributions p and q,

$$0 \leq JSD(p||q) \leq \log 2,$$

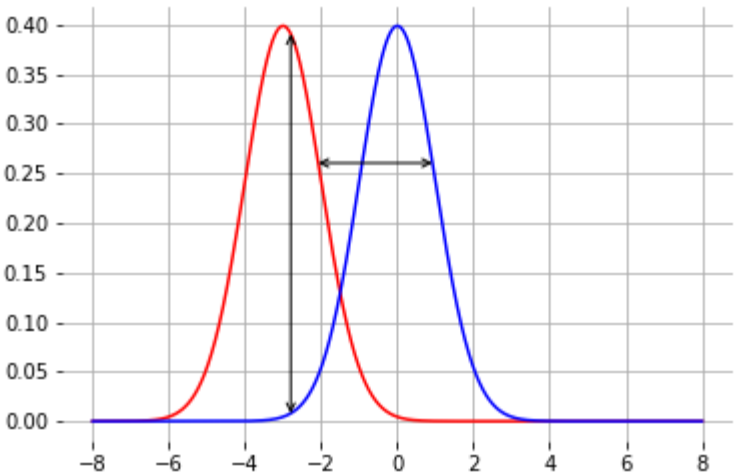
where

- $JSD(p||q) = 0$ if and only if $p = q$,
- $JSD(p||q) = \log 2$ if and only if p and q have disjoint supports.



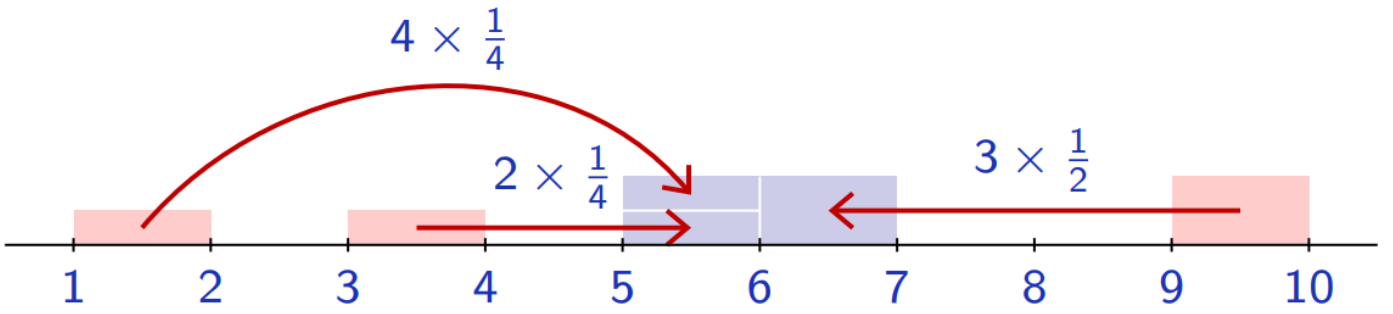
Notice how the Jensen-Shannon divergence poorly accounts for the metric structure of the space.

Intuitively, instead of comparing distributions "vertically", we would like to compare them "horizontally".



Wasserstein distance

An alternative choice is the Earth mover's distance, which intuitively corresponds to the minimum mass displacement to transform one distribution into the other.



- $p = \frac{1}{4}\mathbf{1}_{[1,2]} + \frac{1}{4}\mathbf{1}_{[3,4]} + \frac{1}{2}\mathbf{1}_{[9,10]}$
- $q = \mathbf{1}_{[5,7]}$

Then,

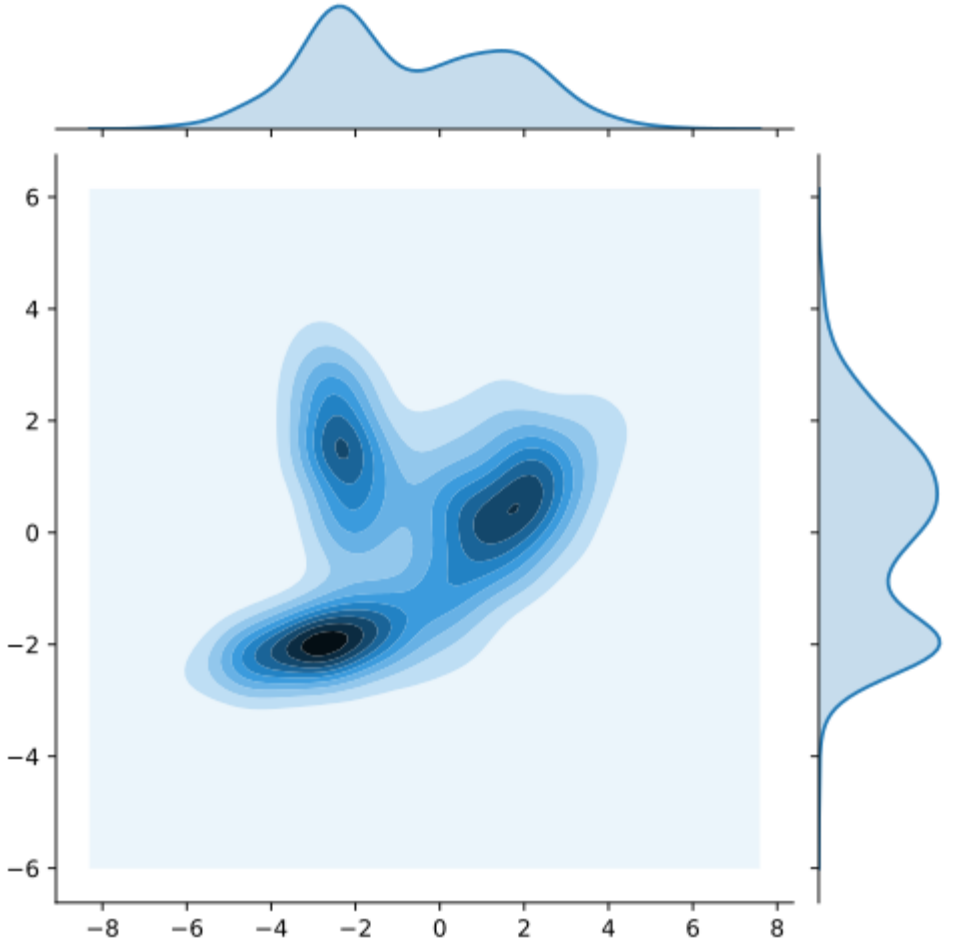
$$\mathrm{W}_1(p, q) = 4 \times \frac{1}{4} + 2 \times \frac{1}{4} + 3 \times \frac{1}{2} = 3$$

The Earth mover's distance is also known as the Wasserstein-1 distance and is defined as:

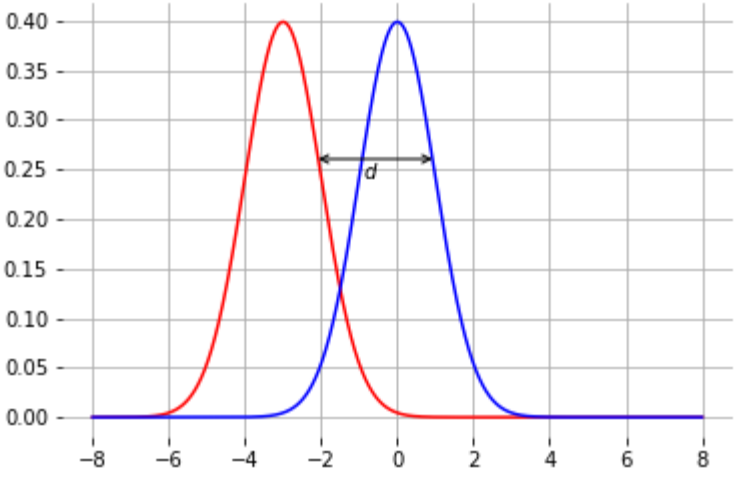
$$\mathbf{W}_1(p, q) = \inf_{\gamma \in \mathbf{\Pi}(p,q)} \mathbb{E}_{(x,y)\sim\gamma} \left[||x - y|| \right]$$

where:

- $\mathbf{\Pi}(p, q)$ denotes the set of all joint distributions $\gamma(x, y)$ whose marginals are respectively p and q;
- $\gamma(x, y)$ indicates how much mass must be transported from x to y in order to transform the distribution p into q.
- $||\cdot||$ is the L1 norm and $||x - y||$ represents the cost of moving a unit of mass from x to y.


6
4
2
0
−2
−4
−6
−8
−6
−4
−2
0
2
4
6
8

Notice how the $\mathbf{W}_1$ distance does not saturate. Instead, it increases monotonically with the distance between modes:



$$\mathbf{W}_1(p, q) = d$$

For any two distributions p and q,

- $W_1(p, q) \in \mathbb{R}^+$,
- $W_1(p, q) = 0$ if and only if $p = q$.

Wasserstein GANs

Given the attractive properties of the Wasserstein-1 distance, Arjovsky et al (2017) propose to learn a generative model by solving instead:

$$\theta^* = \arg\min_\theta \mathrm{W}_1(p(\mathbf{x})||q(\mathbf{x};\theta))$$

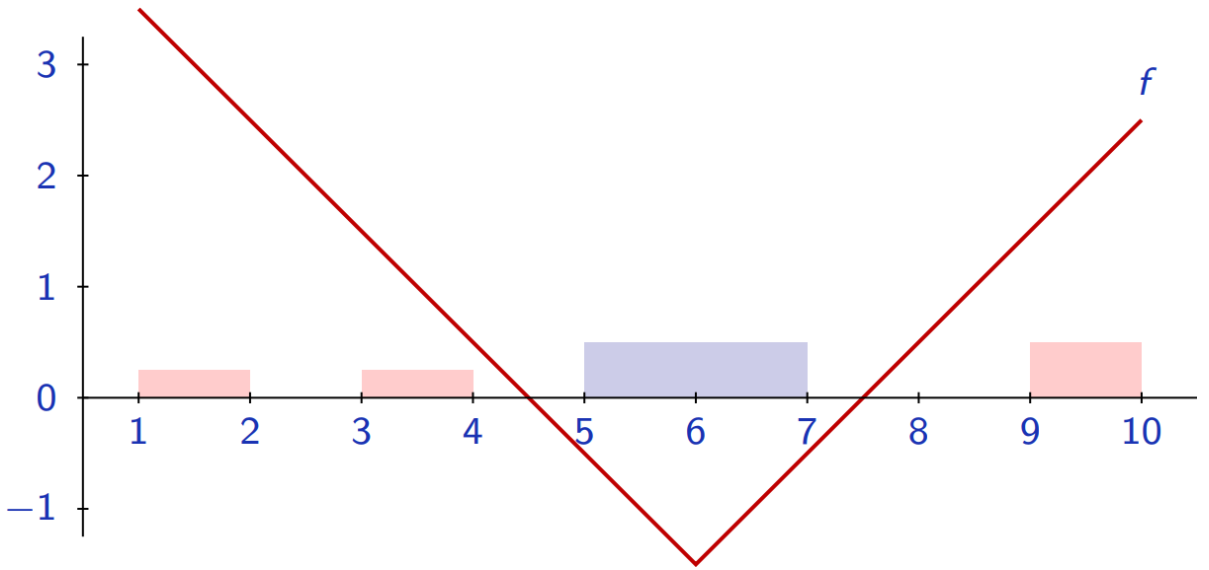
Unfortunately, the definition of W_1 does not provide with an operational way of estimating it because of the intractable $\inf$.

On the other hand, the Kantorovich-Rubinstein duality tells us that

$$\mathrm{W}_1(p(\mathbf{x})||q(\mathbf{x};\theta)) = \sup_{||f||_L \leq 1} \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}\left[f(\mathbf{x})\right] - \mathbb{E}_{\mathbf{x}\sim q(\mathbf{x};\theta)}\left[f(\mathbf{x})\right]$$

where the supremum is over all the 1-Lipschitz functions $f : \mathcal{X} \to \mathbb{R}$. That is, functions f such that

$$||f||_L = \max_{\mathbf{x},\mathbf{x}'} \frac{||f(\mathbf{x}) - f(\mathbf{x}')||}{||\mathbf{x} - \mathbf{x}'||} \leq 1.$$



For $p = \frac{1}{4}\mathbf{1}_{[1,2]} + \frac{1}{4}\mathbf{1}_{[3,4]} + \frac{1}{2}\mathbf{1}_{[9,10]}$ and $q = \mathbf{1}_{[5,7]}$,

$$\mathrm{W}_1(p,q) = 4 \times \frac{1}{4} + 2 \times \frac{1}{4} + 3 \times \frac{1}{2} = 3$$

$$= \underbrace{\left(3 \times \frac{1}{4} + 1 \times \frac{1}{4} + 2 \times \frac{1}{2}\right)}_{\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}[f(\mathbf{x})]} - \underbrace{\left(-1 \times \frac{1}{2} - 1 \times \frac{1}{2}\right)}_{\mathbb{E}_{\mathbf{x}\sim q(\mathbf{x};\theta)}[f(\mathbf{x})]} = 3$$

Using this result, the Wasserstein GAN algorithm consists in solving the minimax problem:

$$\theta^* = \arg\min_{\theta} \max_{\phi:||d(\cdot;\phi)||_L \leq 1} \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}\left[d(\mathbf{x};\phi)\right] - \mathbb{E}_{\mathbf{x}\sim q(\mathbf{x};\theta)}\left[d(\mathbf{x};\phi)\right]$$

Note that this formulation is very close to the original GANs, except that:

- The classifier $d : \mathcal{X} \to [0, 1]$ is replaced by a critic function $d : \mathcal{X} \to \mathbb{R}$ and its output is not interpreted through the cross-entropy loss;
- There is a strong regularization on the form of d. In practice, to ensure 1-Lipschitzness,
  - Arjovsky et al (2017) propose to clip the weights of the critic at each iteration;
  - Gulrajani et al (2017) add a regularization term to the loss.
- As a result, Wasserstein GANs benefit from:
  - a meaningful loss metric,
  - improved stability (no mode collapse is observed).

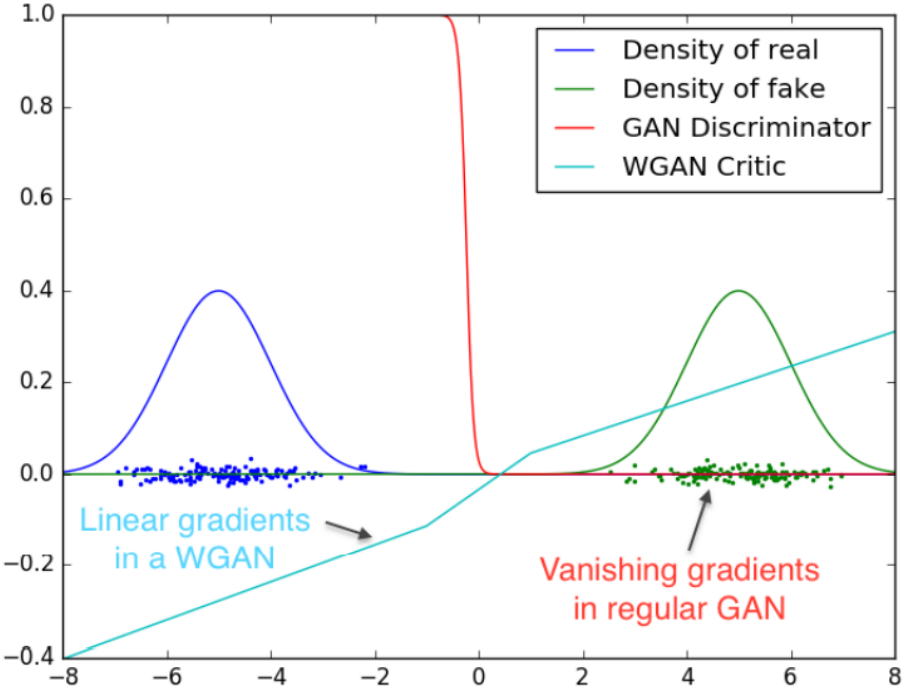
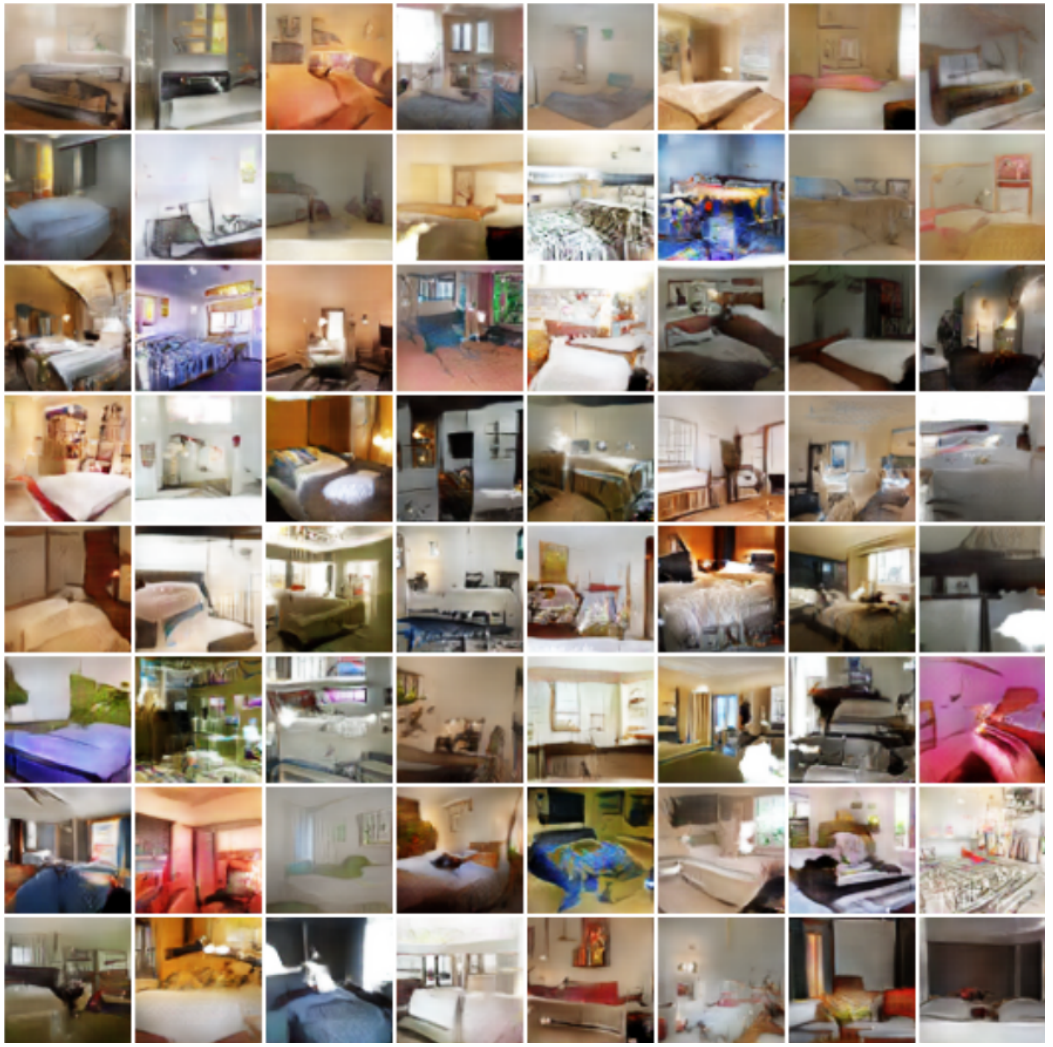


Figure 2: Optimal discriminator and critic when learning to differentiate two Gaussians. As we can see, the discriminator of a minimax GAN saturates and results in vanishing gradients. Our WGAN critic provides very clean gradients on all parts of the space.

—

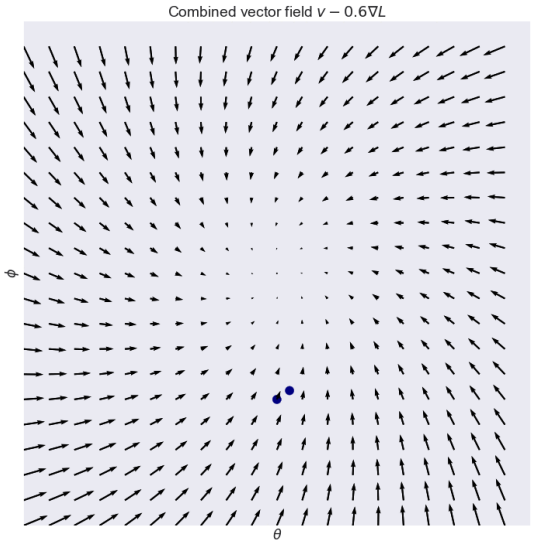
Credits: Arjovsky et al, Wasserstein GAN, 2017.

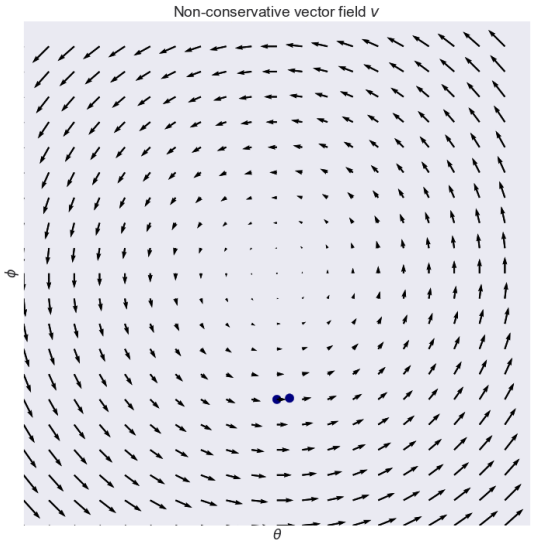


—

Credits: Arjovsky et al, Wasserstein GAN, 2017.

Numerics of GANs





Solving for saddle points is different from gradient descent.

- Minimization of scalar functions yields conservative vector fields.
- Min-max saddle point problems may yield non-conservative vector fields.

—

Credits: Ferenc Huszár, GANs are Broken in More than One Way, 2017.

Following the notations of Mescheder et al (2018), the training objective for the two players can be described by an objective function of the form

$$L(\theta, \phi) = \mathbb{E}_{p(\mathbf{z})}\left[f(d(g(\mathbf{z}; \theta); \phi))\right] + \mathbb{E}_{p(\mathbf{x})}\left[f(-d(\mathbf{x}; \phi))\right],$$

where the goal of the generator is to minimizes the loss, whereas the discriminator tries to maximize it.

If $f(t) = -\log(1 + \exp(-t))$, then we recover the original GAN objective (assuming that d outputs the logits).

Training algorithms can be described as fixed points algorithms that apply some operator $F_h(\theta, \phi)$ to the parameters values (θ, ϕ).

- For simultaneous gradient descent,

$$F_h(\theta, \phi) = (\theta, \phi) + hv(\theta, \phi)$$

  where $v(\theta, \phi)$ denotes the gradient vector field

$$v(\theta, \phi) := \begin{pmatrix} -\frac{\partial L}{\partial \theta}(\theta, \phi) \\ \frac{\partial L}{\partial \phi}(\theta, \phi) \end{pmatrix}$$

  and h is a scalar stepsize.

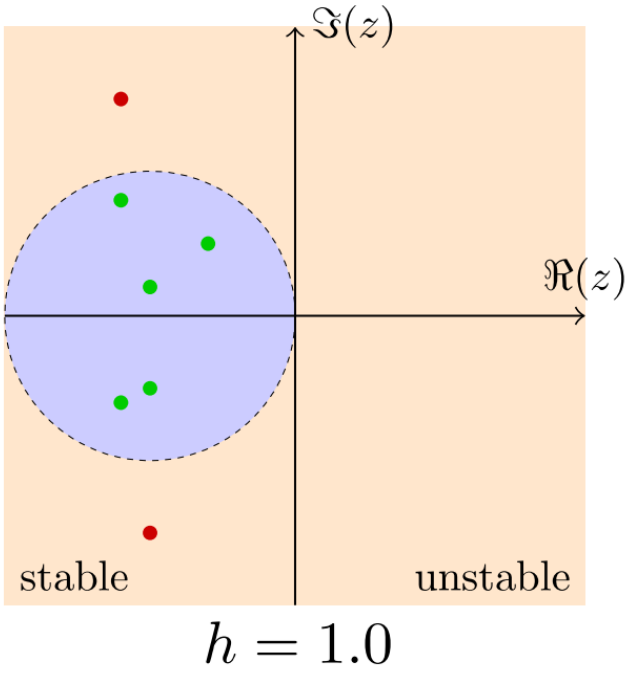
- Similarly, alternating gradient descent can be described by an operator $F_h = F_{2,h} \circ F_{1,h}$, where $F_{1,h}$ and $F_{2,h}$ perform an update for the generator and discriminator, respectively.

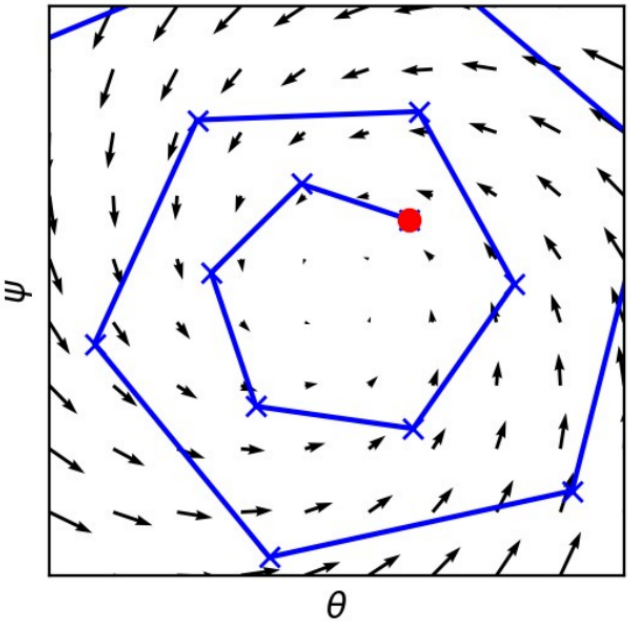
Local convergence near an equilibrium point

Let us consider the Jacobian $J_{F_h}(\theta^*, \phi^*)$ at the equilibrium (θ^*, ϕ^*):

- if $J_{F_h}(\theta^*, \phi^*)$ has eigenvalues with absolute value bigger than 1, the training will generally not converge to (θ^*, ϕ^*).
- if all eigenvalues have absolute value smaller than 1, the training will converge to (θ^*, ϕ^*).
- if all eigenvalues values are on the unit circle, training can be convergent, divergent or neither.

Mescheder et al (2017) show that all eigenvalues can be forced to remain within the unit ball if and only if the stepsize h is made sufficiently small.

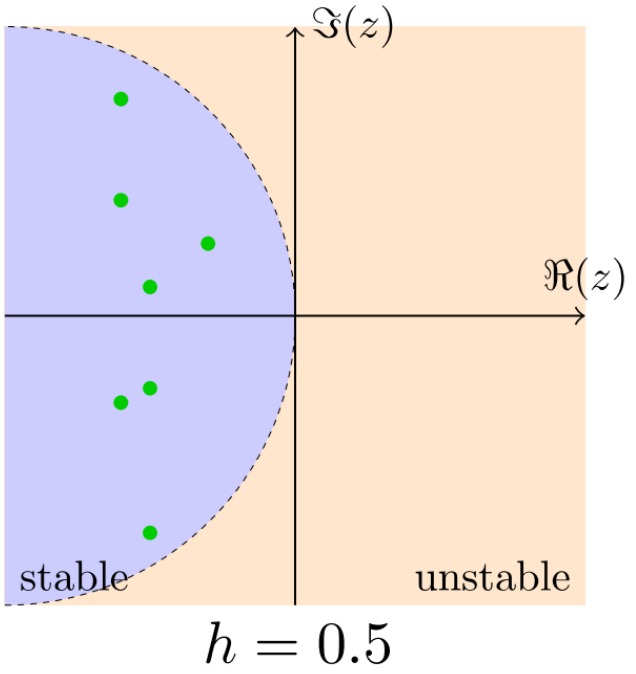


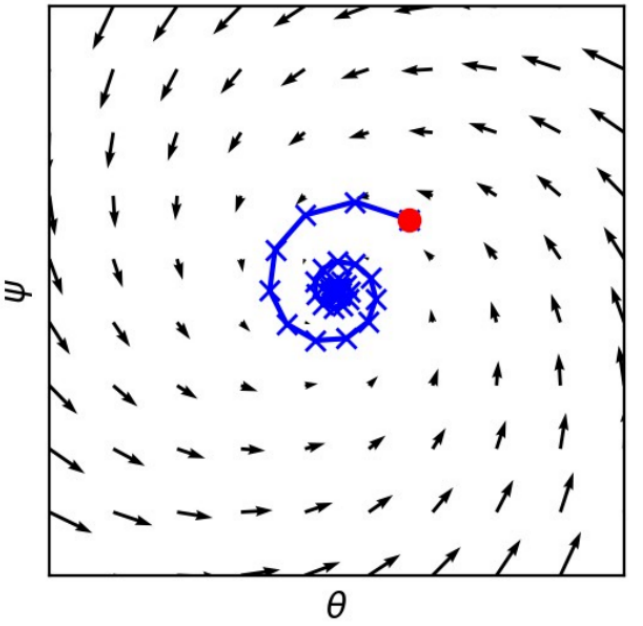


Discrete system: divergence ($h = 1$, too large).

—

Credits: Mescheder et al, Which Training Methods for GANs do actually Converge?, 2018.





Discrete system: convergence ($h = 0.5$, small enough).

—

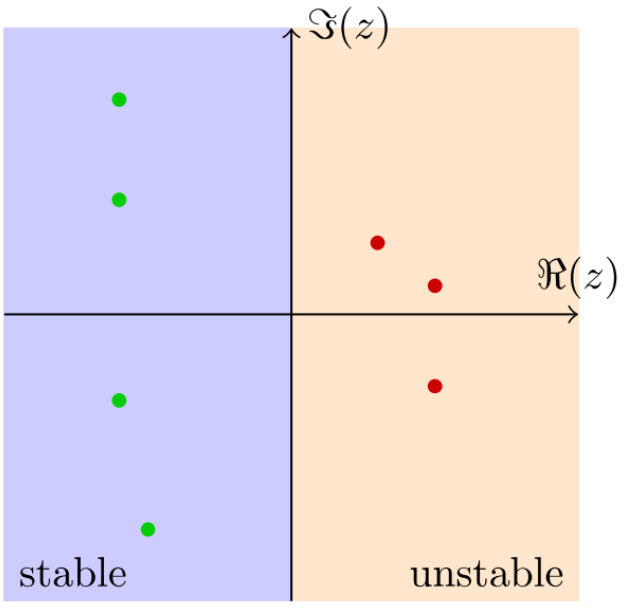
Credits: Mescheder et al, Which Training Methods for GANs do actually Converge?, 2018.

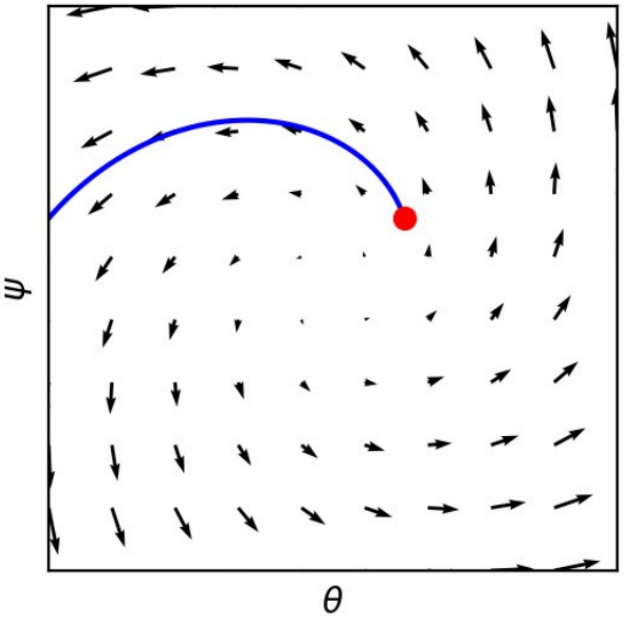
For the (idealized) continuous system

$$\begin{pmatrix} \dot{\theta}(t) \\ \dot{\phi}(t) \end{pmatrix} = \begin{pmatrix} -\frac{\partial L}{\partial \theta}(\theta, \phi) \\ \frac{\partial L}{\partial \phi}(\theta, \phi) \end{pmatrix},$$

which corresponds to training GANs with infinitely small learning rate $h \to 0$:

- if all eigenvalues of the Jacobian $v'(\theta^*, \phi^*)$ at a stationary point (θ^*, ϕ^*) have negative real-part, the continuous system converges locally to (θ^*, ϕ^*);
- if $v'(\theta^*, \phi^*)$ has eigenvalues with positive real-part, the continuous system is not locally convergent.
- if all eigenvalues have zero real-part, it can be convergent, divergent or neither.

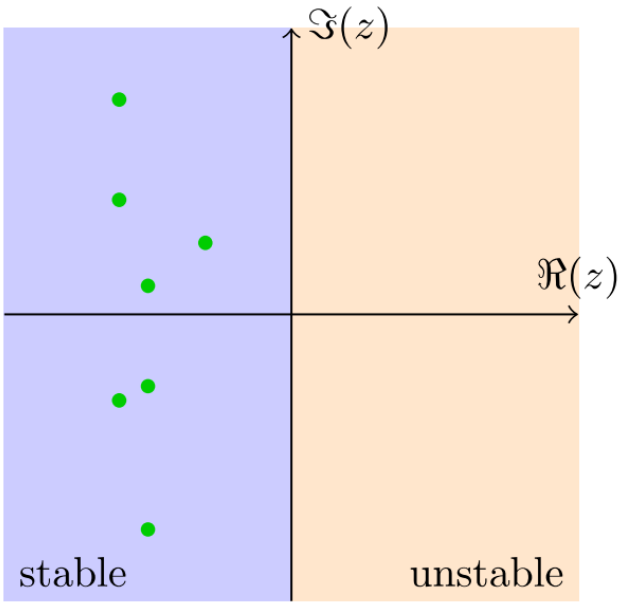


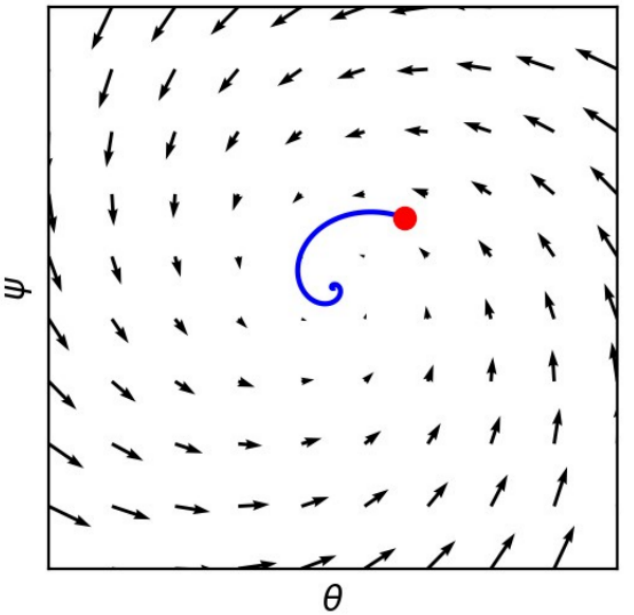


Continuous system: divergence.

—

Credits: Mescheder et al, Which Training Methods for GANs do actually Converge?, 2018.



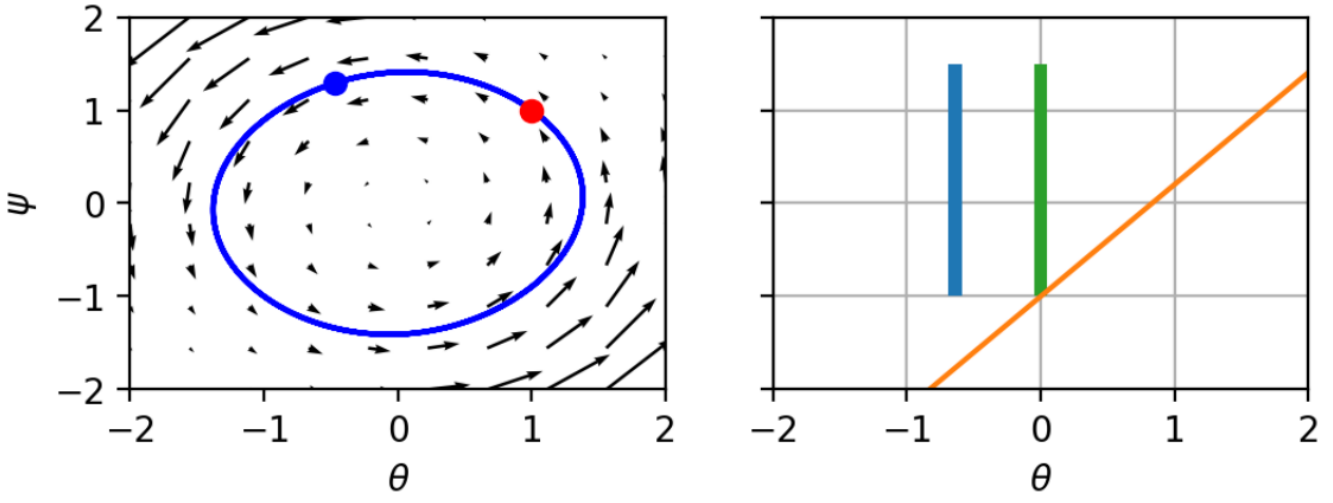


Continuous system: convergence.

—

Credits: Mescheder et al, Which Training Methods for GANs do actually Converge?, 2018.

Dirac-GAN: Vanilla GANs

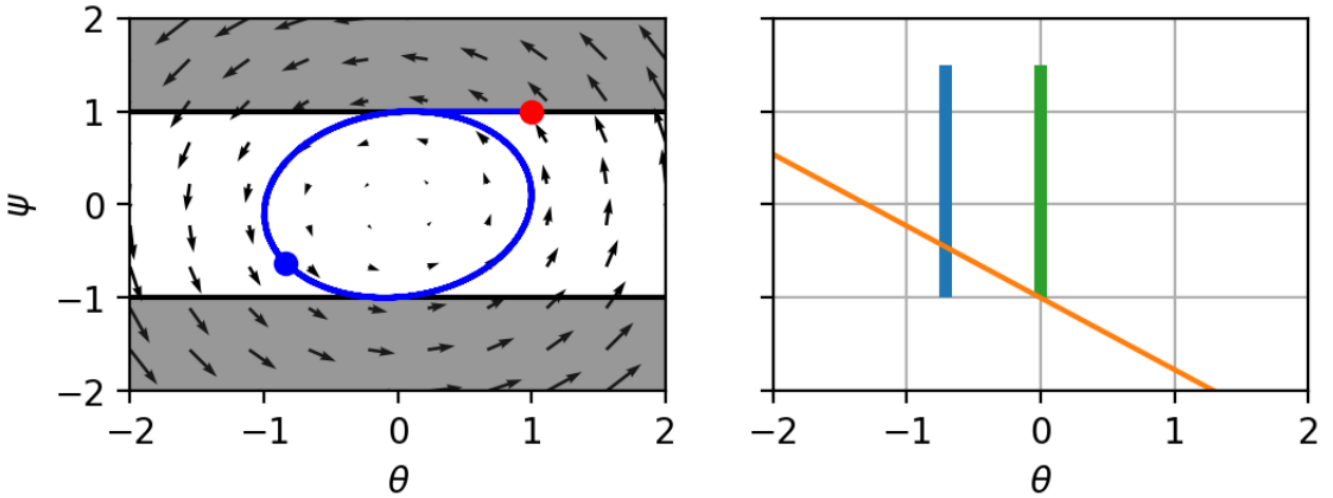


On the Dirac-GAN toy problem, eigenvalues are $\{-f'(0)i, +f'(0)i\}$. Therefore convergence is not guaranteed.

—

Credits: Mescheder et al, Which Training Methods for GANs do actually Converge?, 2018.

Dirac-GAN: Wasserstein GANs

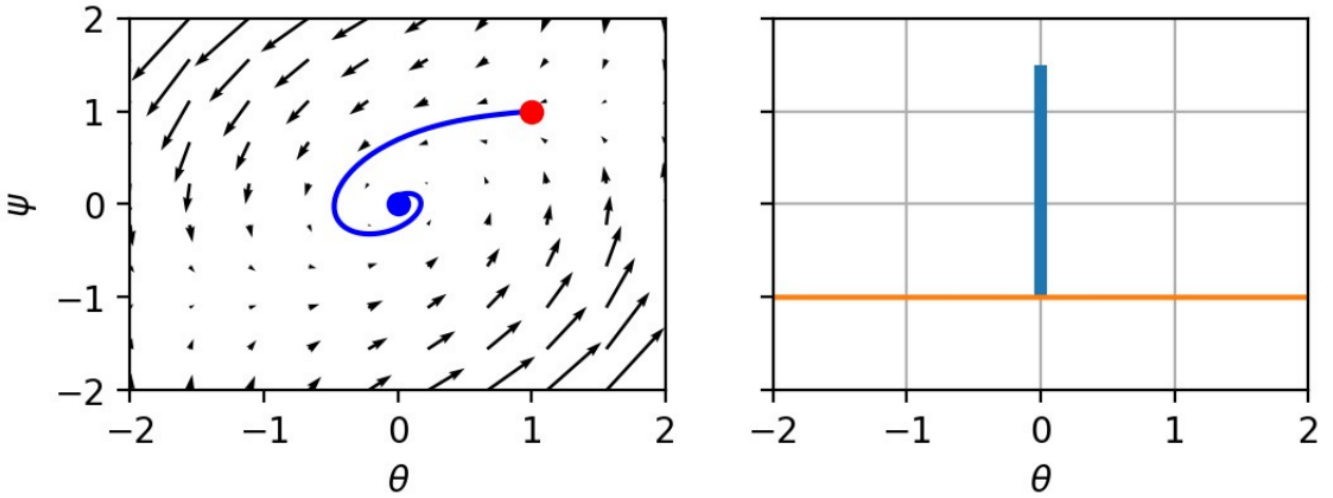


Eigenvalues are $\{-i, +i\}$. Therefore convergence is not guaranteed.

—

Credits: Mescheder et al, Which Training Methods for GANs do actually Converge?, 2018.

Dirac-GAN: Zero-centered gradient penalties



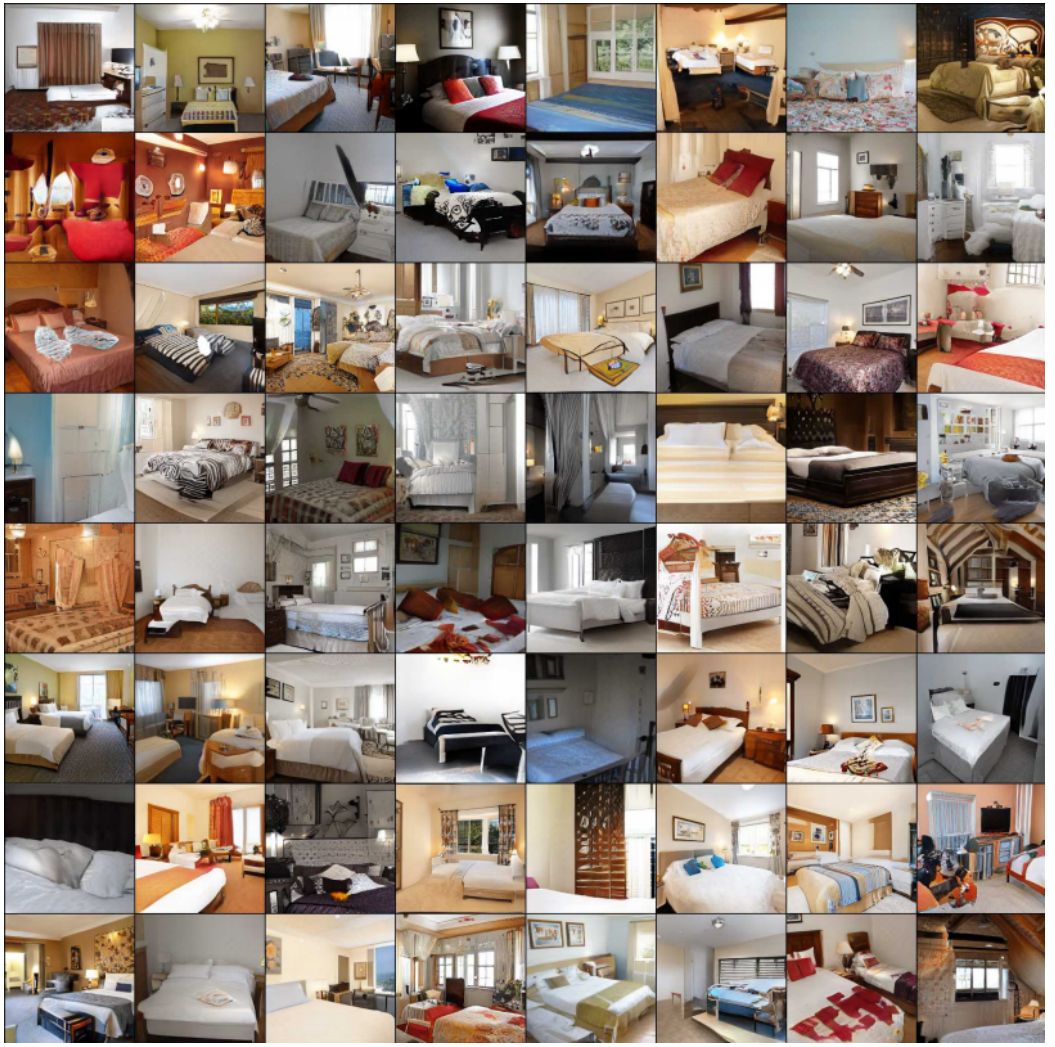
A penalty on the squared norm of the gradients of the discriminator results in the regularization

$$R_1(\phi) = \frac{\gamma}{2}\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}\left[||\nabla_{\mathbf{x}} d(\mathbf{x};\phi)||^2\right].$$

The resulting eigenvalues are $\left\{-\frac{\gamma}{2} \pm \sqrt{\frac{\gamma}{4} - f'(0)^2}\right\}$. Therefore, for $\gamma > 0$, all eigenvalues have negative real part, hence training is locally convergent!

—

Credits: Mescheder et al, Which Training Methods for GANs do actually Converge?, 2018.



—

Credits: Mescheder et al, Which Training Methods for GANs do actually Converge?, 2018.

—

Credits: Mescheder et al, Which Training Methods for GANs do actually Converge?, 2018.

—

Credits: Mescheder et al, Which Training Methods for GANs do actually Converge?, 2018.

State of the art

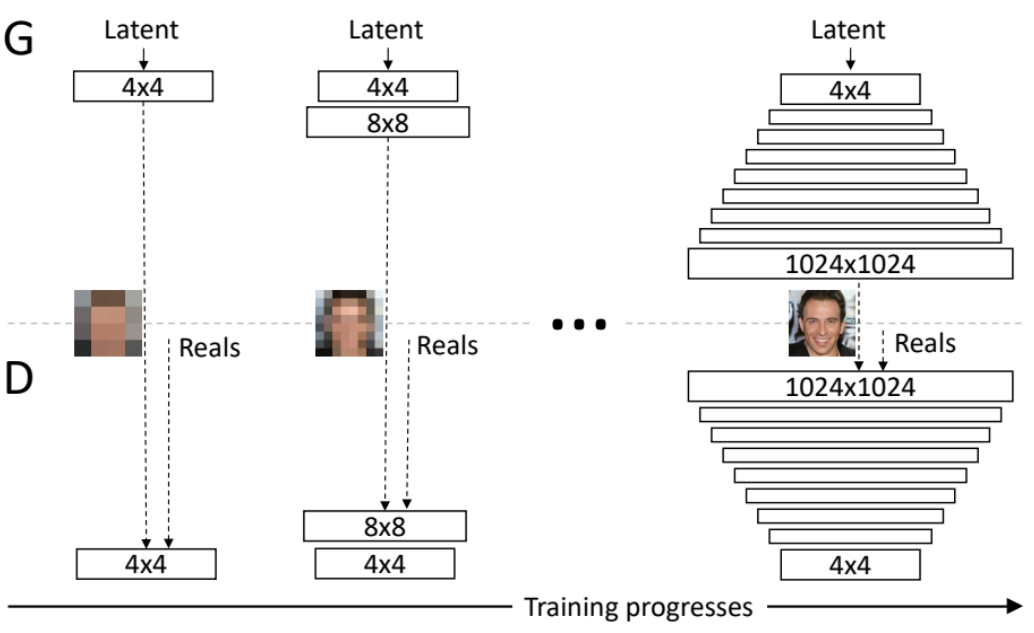
Ian Goodfellow
@goodfellow_ian

4.5 years of GAN progress on face generation. arxiv.org/abs/1406.2661
arxiv.org/abs/1511.06434
arxiv.org/abs/1606.07536
arxiv.org/abs/1710.10196
arxiv.org/abs/1812.04948

Progressive growing of GANs

Wasserstein GANs as baseline (Arjovsky et al, 2017) +
Gradient Penalty (Gulrajani, 2017) + (quite a few other tricks)

+



(Karras et al, 2017)

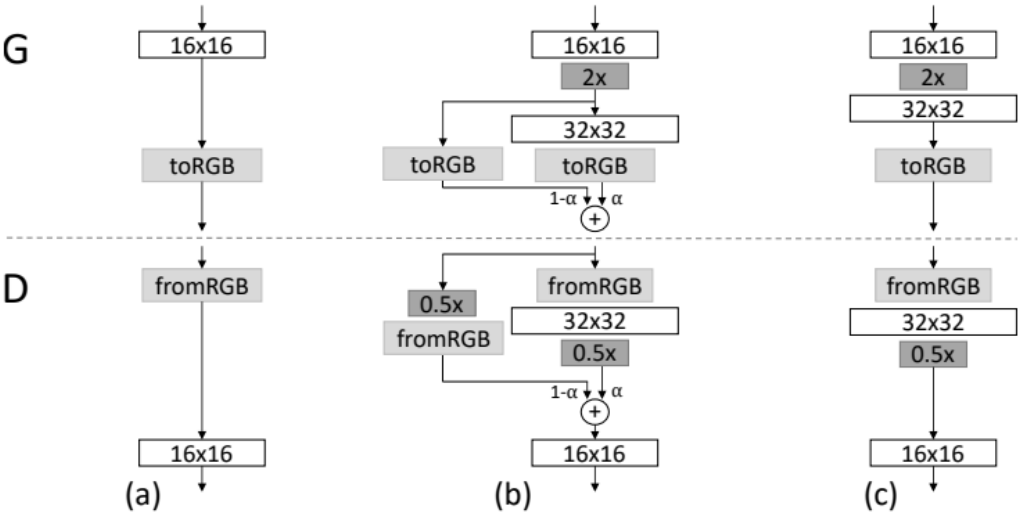
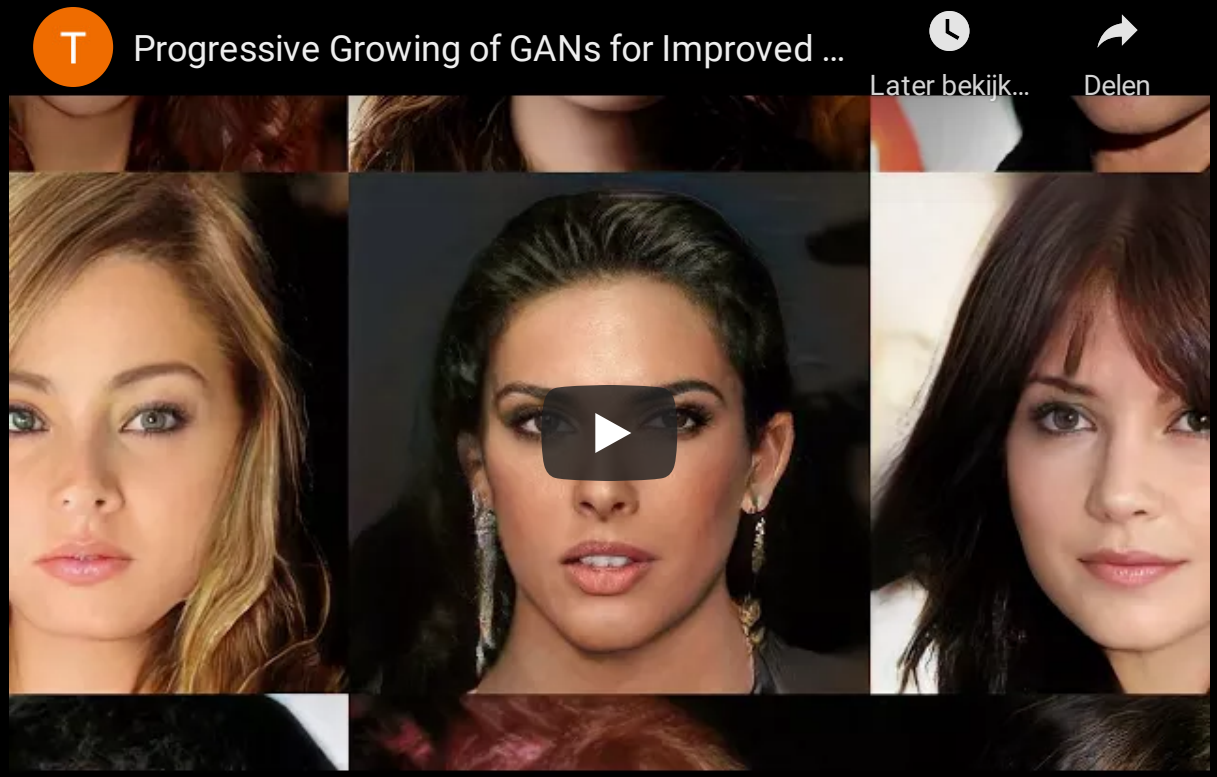


Figure 2: When doubling the resolution of the generator (G) and discriminator (D) we fade in the new layers smoothly. This example illustrates the transition from 16×16 images (a) to 32×32 images (c). During the transition (b) we treat the layers that operate on the higher resolution like a residual block, whose weight α increases linearly from 0 to 1. Here $\boxed{2\times}$ and $\boxed{0.5\times}$ refer to doubling and halving the image resolution using nearest neighbor filtering and average pooling, respectively. The $\boxed{\text{toRGB}}$ represents a layer that projects feature vectors to RGB colors and $\boxed{\text{fromRGB}}$ does the reverse; both use 1×1 convolutions. When training the discriminator, we feed in real images that are downscaled to match the current resolution of the network. During a resolution transition, we interpolate between two resolutions of the real images, similarly to how the generator output combines two resolutions.

(Karras et al, 2017)



(Karras et al, 2017)

BigGANs

Self-attention GANs as baseline (Zhang et al, 2018) + Hinge loss objective (Lim and Ye, 2017; Tran et al, 2017) + Class information to g with class-conditional batchnorm (de Vries et al, 2017) + Class information to d with projection (Miyato and Koyama, 2018) + Half the learning rate of SAGAN, 2 d-steps per g-step + Spectral normalization for both g and d + Orthogonal initialization (Saxe et al, 2014) + Large minibatches (2048) + Large number of convolution filters + Shared embedding and hierarchical latent spaces + Orthogonal regularization + Truncated sampling + (quite a few other tricks)

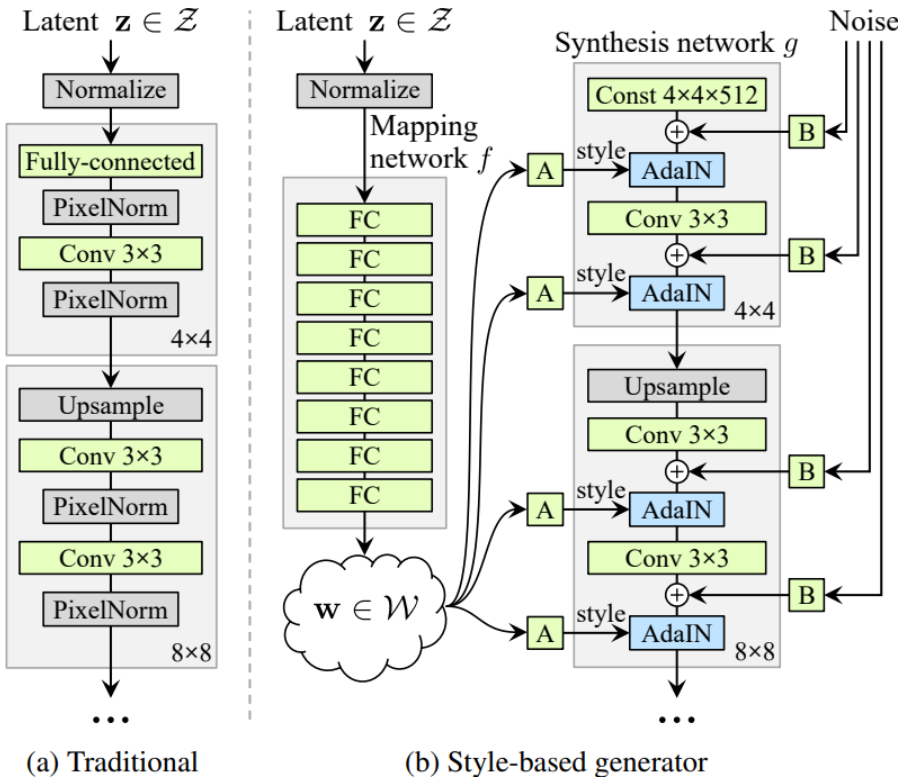
(Brock et al, 2018)

(Brock et al, 2018)

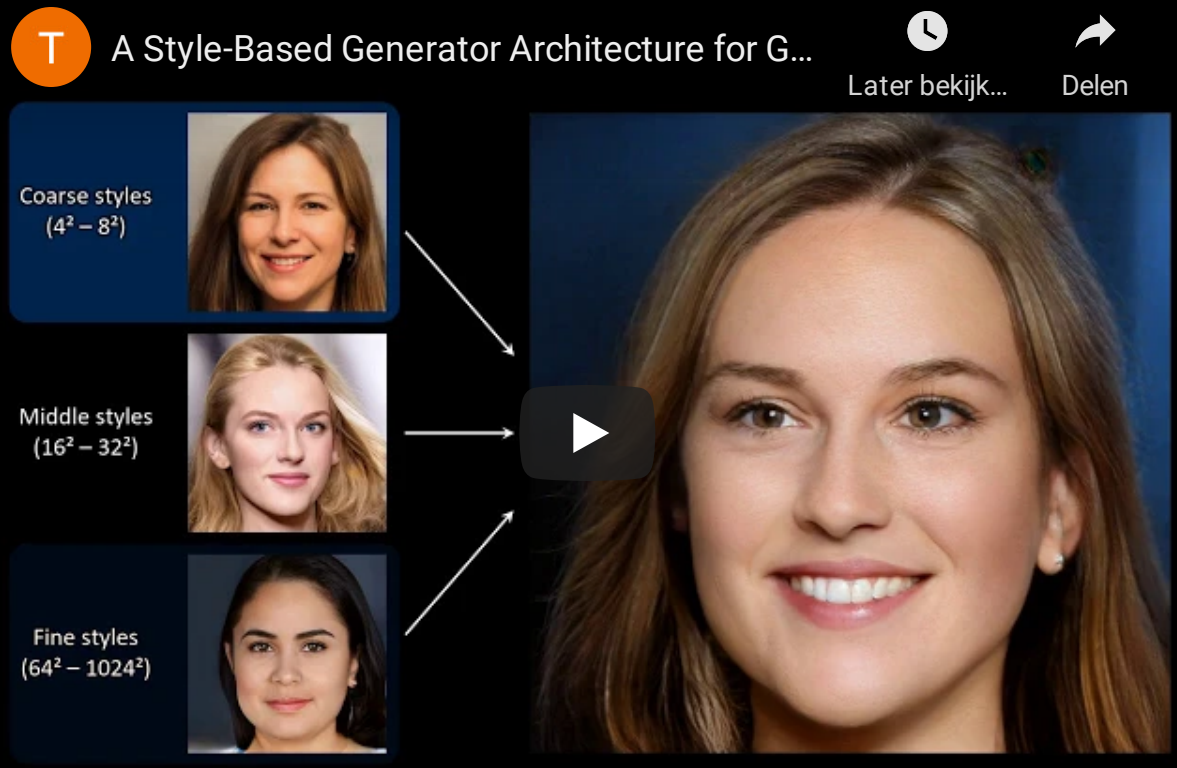
StyleGAN (v1)

Progressive GANs as baseline (Karras et al, 2017) + Non-saturating loss instead of WGAN-GP + R_1 regularization (Mescheder et al, 2018) + (quite a few other tricks)

+



(a) Traditional (b) Style-based generator



(Karras et al, 2018)

The StyleGAN generator g is so powerful that it can re-generate arbitrary faces.

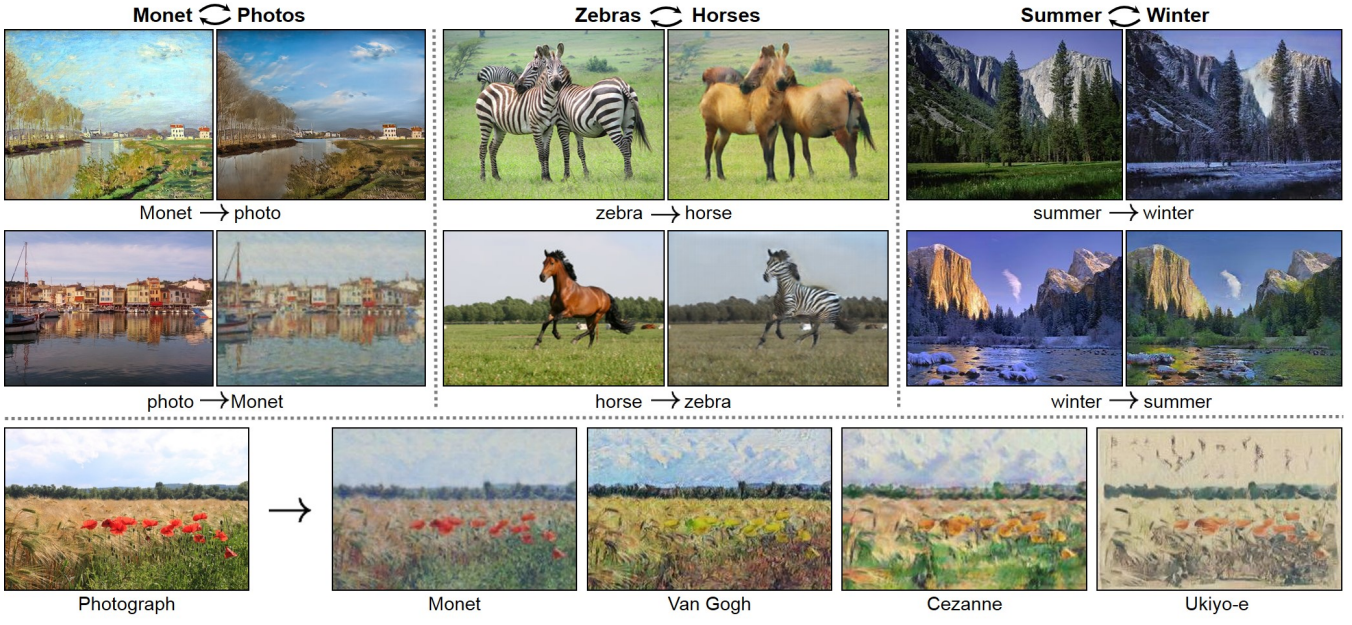
StyleGAN (v2)

(Karras et al, 2019)

Applications

$p(\mathbf{z})$ need not be a random noise distribution.

Image-to-image translation



CycleGANs (Zhu et al, 2017)

High-resolution image synthesis (Wang et al, 2017)

GauGAN: Changing sketches into photorealistic masterpieces (NVIDIA, 2019)

Living portraits / deepfakes

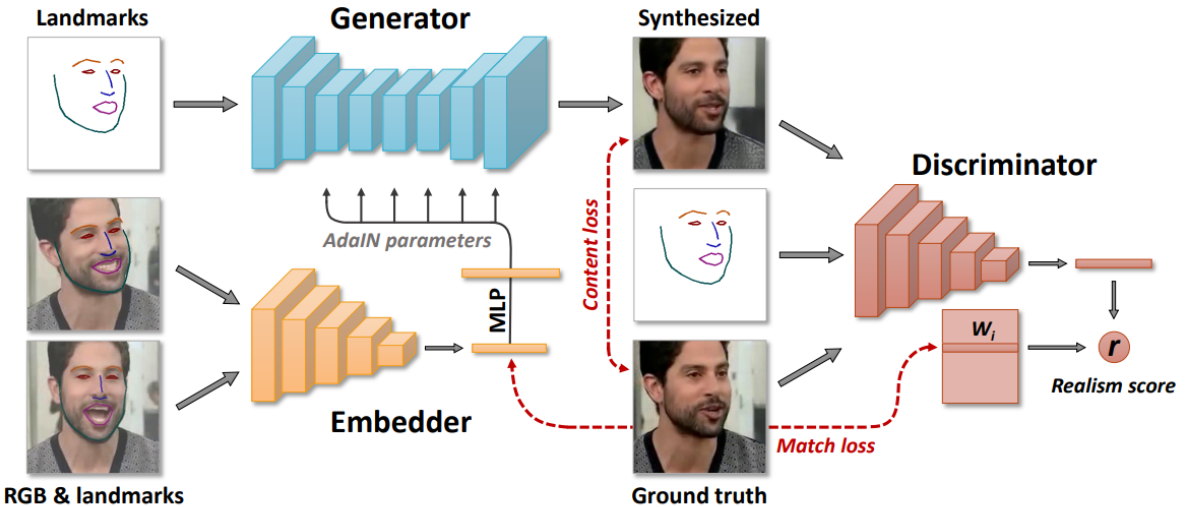
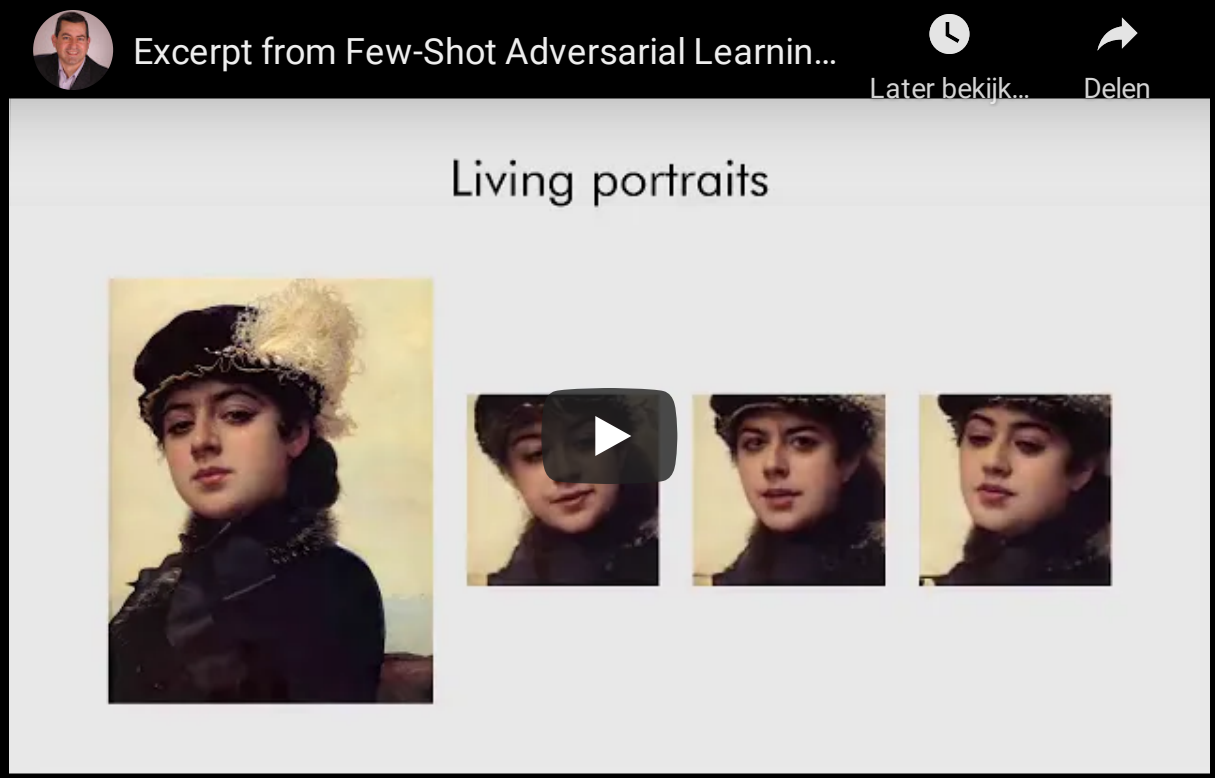


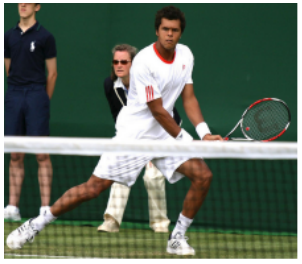
Figure 2: Our meta-learning architecture involves the embedder network that maps head images (with estimated face landmarks) to the embedding vectors, which contain pose-independent information. The generator network maps input face landmarks into output frames through the set of convolutional layers, which are modulated by the embedding vectors via adaptive instance normalization. During meta-learning, we pass sets of frames from the same video through the embedder, average the resulting embeddings and use them to predict adaptive parameters of the generator. Then, we pass the landmarks of a different frame through the generator, comparing the resulting image with the ground truth. Our objective function includes perceptual and adversarial losses, with the latter being implemented via a conditional projection discriminator.

Few-Shot Adversarial Learning of Realistic Neural Talking Head Models (Zakharov et al, 2019)



Few-Shot Adversarial Learning of Realistic Neural Talking Head Models
(Zakharov et al, 2019)

Captioning



a tennis player gets ready to return a serve

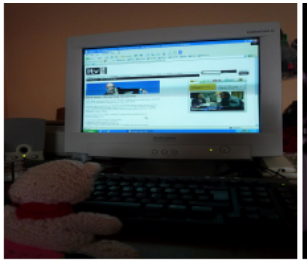
two men dressed in costumes and holding tennis rackets

a tennis player hits the ball during a match

a male tennis player in action on the court

a man in white is about to serve a tennis ball

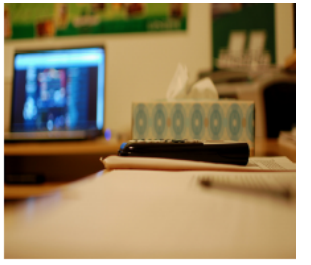
a laptop and a desktop computer sit on a desk



a person is working on a computer screen

a cup of coffee sitting next to a laptop

a laptop computer sitting on top of a desk next to a



a picture of a computer on a desk

(Shetty et al, 2017)

Text-to-image synthesis

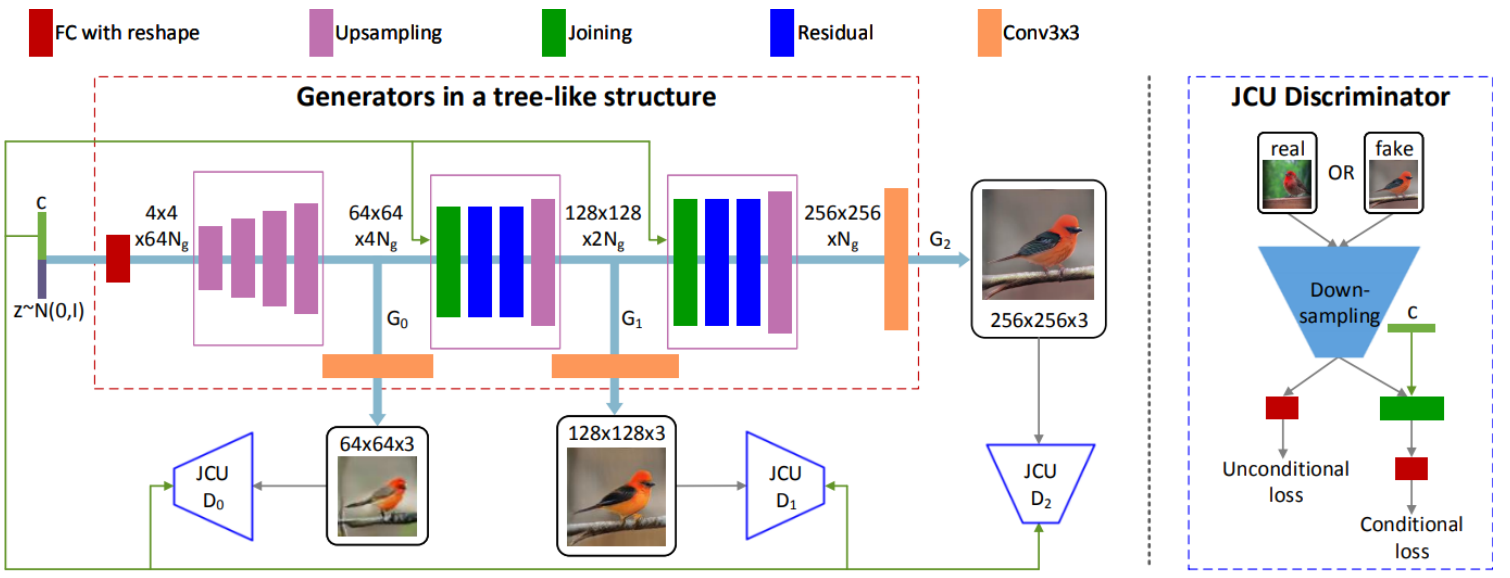


Fig. 2: The overall framework of our proposed StackGAN-v2 for the conditional image synthesis task. c is the vector of conditioning variables which can be computed from the class label, the text description, *etc*.. N_g and N_d are the numbers of channels of a tensor.

(Zhang et al, 2017)

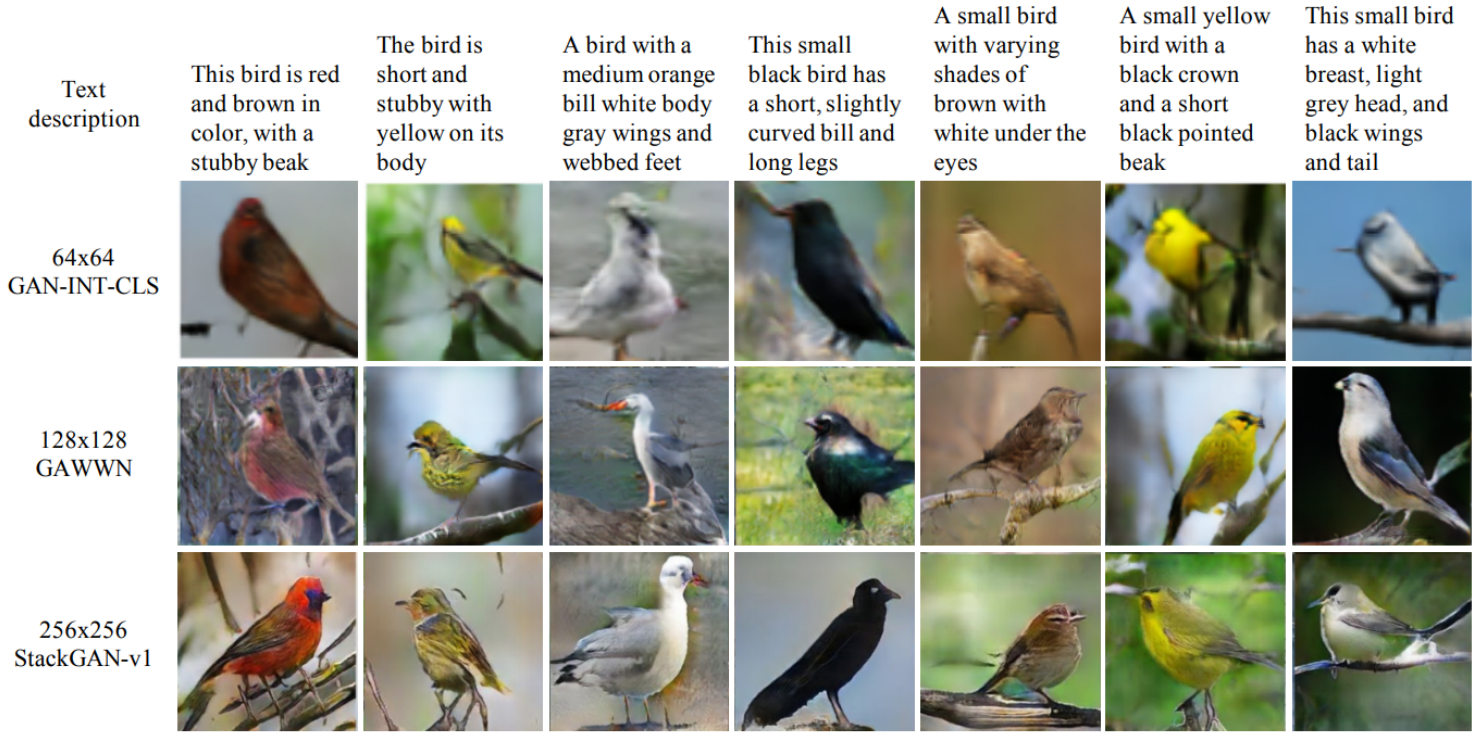


Fig. 3: Example results by our StackGAN-v1, GAWWN [29], and GAN-INT-CLS [31] conditioned on text descriptions from CUB test set.

(Zhang et al, 2017)

Music generation

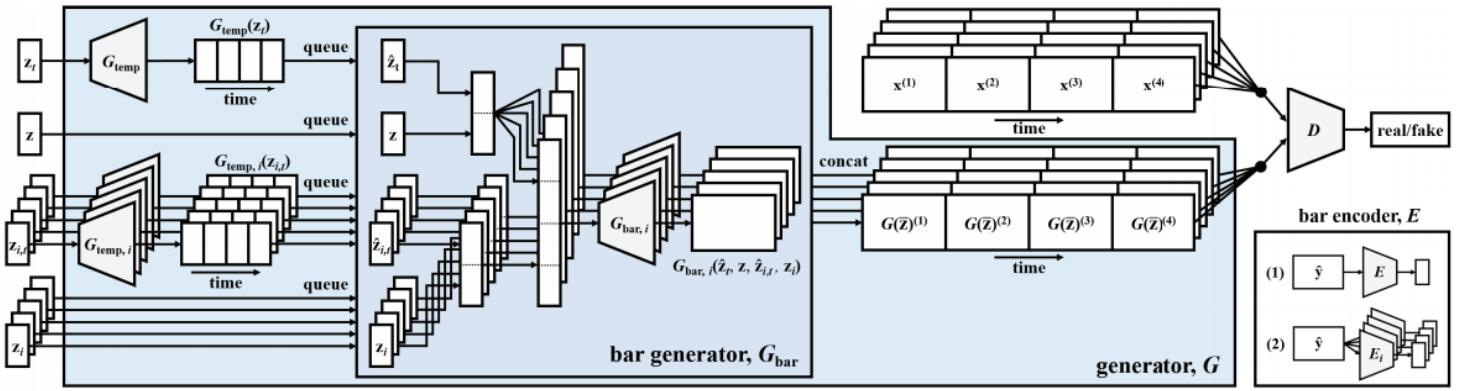


Figure 5: System diagram of the proposed MuseGAN model for multi-track sequential data generation.

0:00 / 3:15

MuseGAN (Dong et al, 2018)

Accelerating scientific simulators

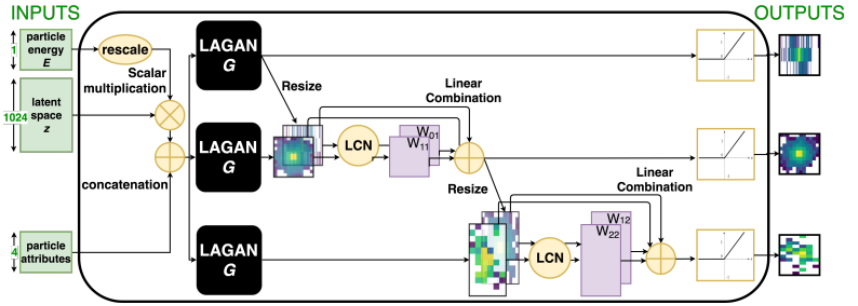


Figure 8.37: Composite conditional CaloGAN generator G, with three LAGAN-like streams connected by attentional layer-to-layer dependence.

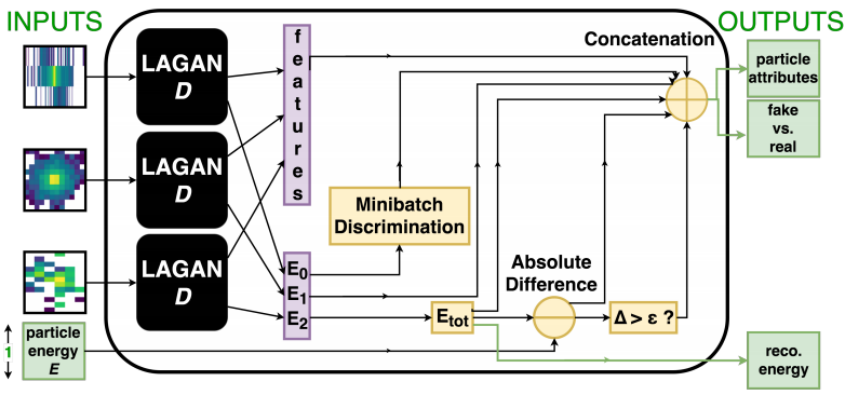
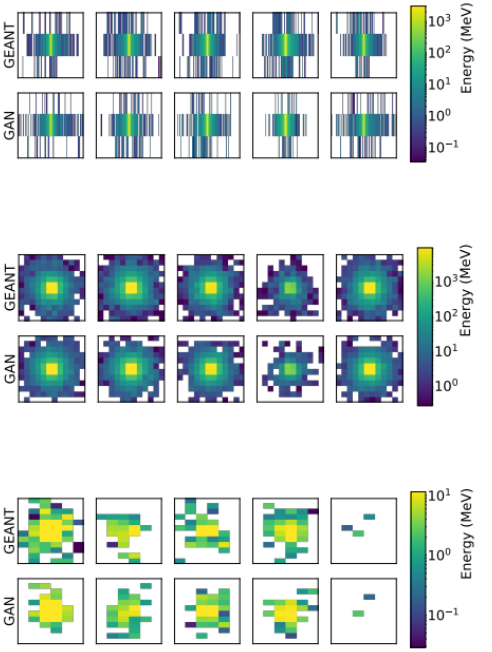


Figure 8.38: Composite conditional CaloGAN discriminator D, with three LAGAN-like streams and additional domain-specific energy calculations included in the final feature space.



Learning particle physics (Paganini et al, 2017)

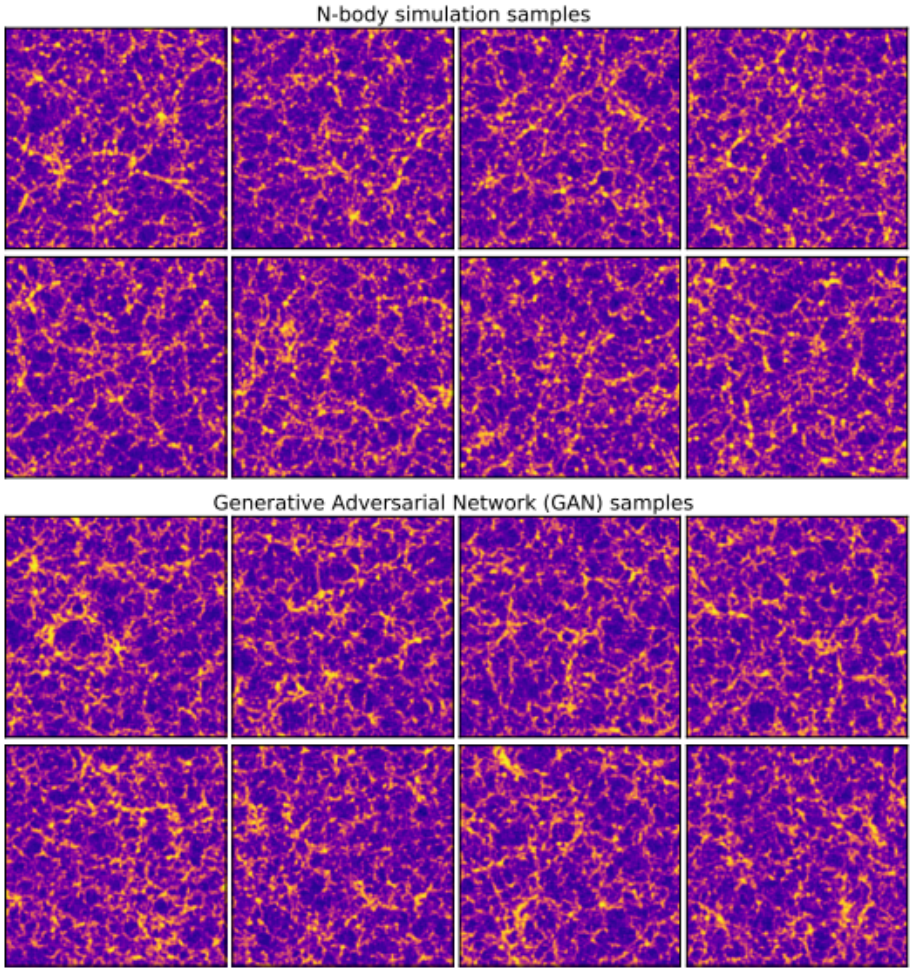


Figure 1: Samples from N-body simulation and from GAN for the box size of 500 Mpc. Note that the transformation in Equation 3.1 with $a = 20$ was applied to the images shown above for better clarity.

Learning cosmological models (Rodriguez et al, 2018)

The end.