Generative Modeling

Generative adversarial networks

Denis Derkach, Artem Ryzhikov, Maxim Artemiev

Laboratory for methods of big data analysis

LAMBDA • HSE

Spring 2022

In this Lecture

- Generative Adversarial Networks
  - algorithm statement;
  - ideal case;
  - shortcomings of vanilla algorithm;
  - proposed shortcuts.

Idea

Reminder: f-divergence

- For convex $f(.)$, P and Q some distributions, we define f-divergence:

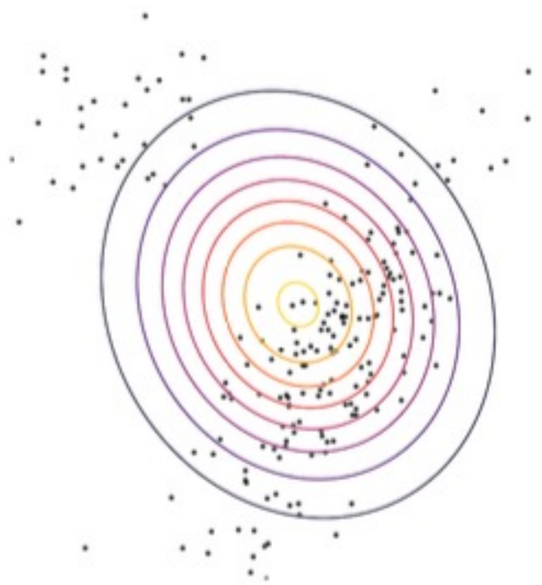
$$\boxed{D_f(P||Q) = \int q(x) f\left(\frac{p(x)}{q(x)}\right) dx.}$$

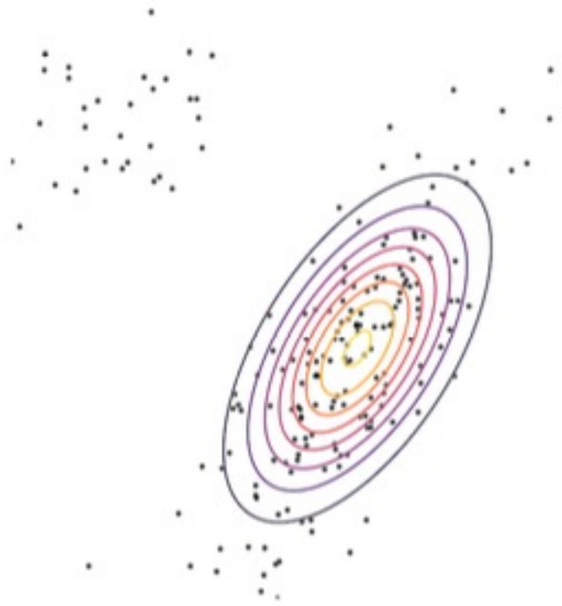
Reminder: f-divergence

- For convex $f(.)$, P and Q some distributions, we define f-divergence:

$$\boxed{D_f(P||Q) = \int q(x) f\left(\frac{p(x)}{q(x)}\right) dx.}$$

$$KL = \int p(X) \log \frac{p(x)}{q_\theta(x)} dx \qquad rKL = \int q(x) \log \frac{q_\theta(x)}{p(x)} dx$$



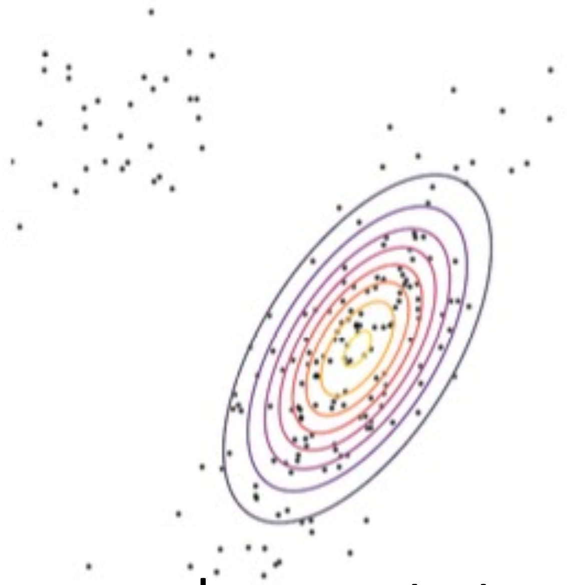


Reminder: f-divergence Convergence

- For convex $f(.)$, P and Q some distributions, we define f-divergence:

$$\boxed{D_f(P||Q) = \int q(x) f\left(\frac{p(x)}{q(x)}\right) dx.}$$

$$rKL = \int q(x) \log \frac{q_\theta(x)}{p(x)} dx$$



$$\theta^* = \underset{\theta}{\operatorname{argmin}} KL(q_\theta(x)||p(x))$$

$$= \underset{\theta}{\operatorname{argmax}}(-\mathbb{E}_{\tilde{x}\sim q_\theta}[\log q_\theta(x)] + \mathbb{E}_{\tilde{x}\sim q_\theta}[\log p(x)])$$

To optimize rKL properly we need access to true PDF, **p(x)**.

Reminder: Variational Lower Bound

- For convex $f(.)$, P and Q some distributions, we define f-divergence:

$$\boxed{D_f(P||Q) = \int q(x) f\left(\frac{p(x)}{q(x)}\right) dx.}$$

- Lower bound can be written:

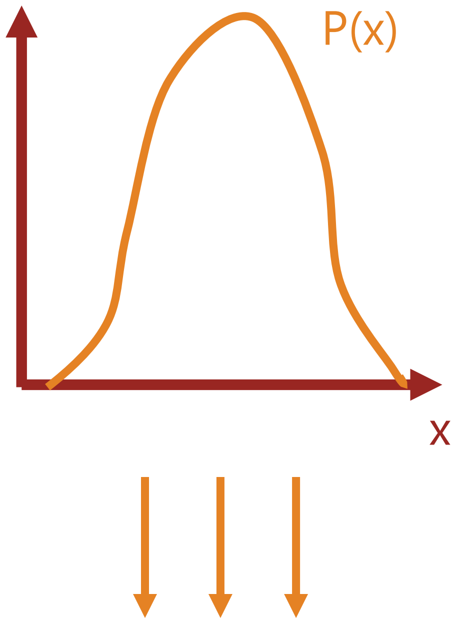
$$D_f(P||Q) \geq \max_{T(x)} \mathbb{E}_{x \sim P} T(x) - \mathbb{E}_{x \sim Q} f^*\big(T(x)\big),$$

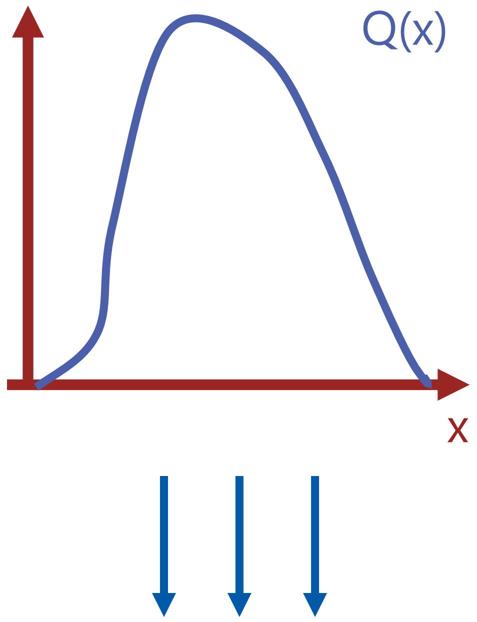
$T(x)$ is some random function.

- The tight boundary can be estimated for each f-divergence ($T^*(x)$).
- For JS-divergence $T^*(x) = \log \frac{2p(x)}{p(x)+q(x)}$, $f^*(x) = -\log(2 - \exp(t))$.

Lower Bound for JS

$$\boxed{JS(p,q) = \frac{1}{2}\left(KL(p(x)||\frac{p(x)+q_\theta(x)}{2}) + KL(q_\theta(x)||\frac{p(x)+q_\theta(x)}{2})\right)}$$





$$\mathrm{JS}(P||Q) \geq \quad \mathbb{E}_{x\sim P} \log \frac{2p(x)}{p(x)+q(x)} \quad + \quad \mathbb{E}_{x\sim Q} \log \frac{2q(x)}{p(x)+q(x)}$$

It would be interesting to construct something close.

Adversarial optimization

Rationale

- Need to optimize the model q_θ without the direct access to the $p(x)$.

$$\theta^* = \arg\min_\theta \max_\phi \mathbb{E}_{x\sim p, \tilde{x}\sim q_\theta} V(f_\phi(x), f_\phi(\tilde{x}))$$

- Instead of minimizing over some analytically defined divergence with parameter ϕ, one could minimize over "learned divergence".

Generator

- G_θ is a **generator**. It should sample from a random noise:

$$z_j \sim N(0;1);$$

$$x_j = \mathrm{G}_\theta\left(z_j\right).$$

- Our aim is G_θ as a neural network.
- We thus have a sample:

$$\{x_j\} \sim q_\theta(x)$$

- G_θ can be defined in many ways. For example, physics generator.

Borisyak M et al. *PeerJ Computer Science* 6:e274 (2020)

Discriminator

- Add a classifying neural network, **discriminator D_ϕ**, to distinguish between the real and generated samples.
- Optimize:

$$\max_{\phi}\left(\underbrace{\mathbb{E}_{x\sim p(x)}}_{\text{Real samples}} \log\left(D_\phi(x)\right) + \underbrace{\mathbb{E}_{x\sim q_\theta(x)}}_{\text{Generated samples}} \log(1 - D_\phi(x))\right)$$

G+D recap

We can now put together generator and discriminator.

- objective of discriminator:

$$\max_{\phi}(\mathbb{E}_{x\sim p(x)} \log\left(D_{\phi}(x)\right) + \mathbb{E}_{x\sim q_{\theta}(x)} \log(1 - D_{\phi}(x)))$$

- objective of generator:

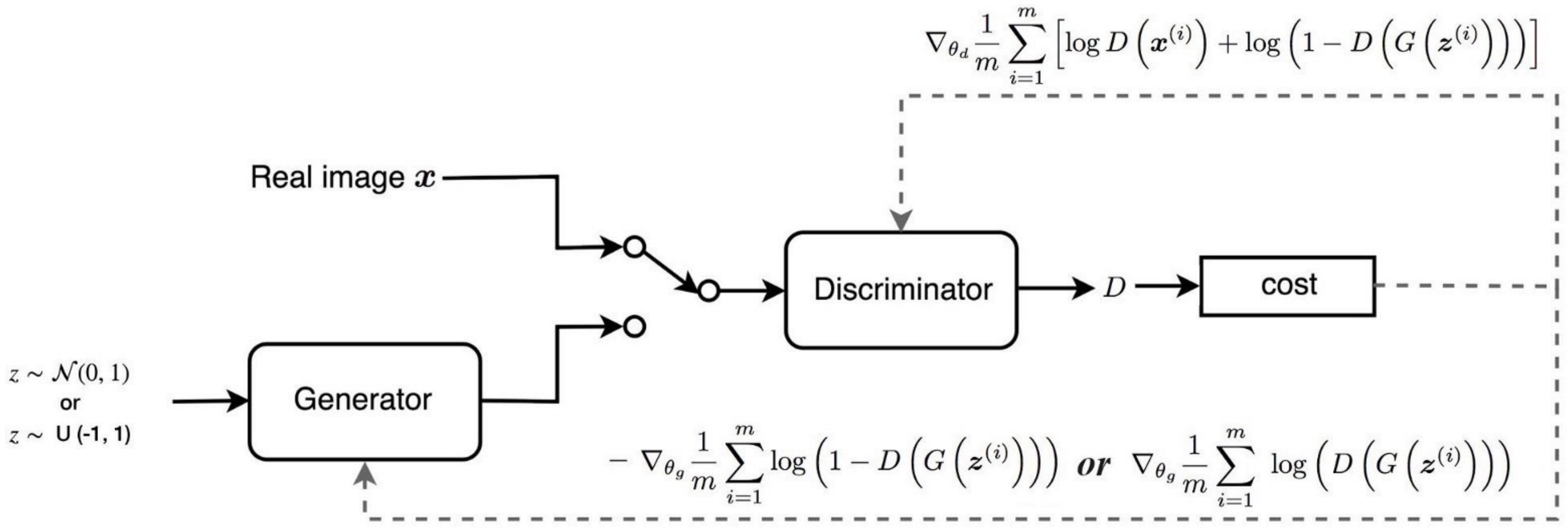
$$\min_{\phi} \mathbb{E}_{z\sim N(0;1)} \log(1 - D_{\phi}(G_{\theta}(z)))$$

We thus defined a minimax game:

$$\min_{\theta} \max_{\phi} \mathbb{E}_{x\sim p, \tilde{x}\sim q_{\theta}} V(f_{\phi}(x), f_{\phi}(\tilde{x})).$$

In exactly the way we wanted.

Training at a Glance


$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right]$
Real image $\boldsymbol{x}$
Discriminator
D
cost
$z \sim \mathcal{N}(0, 1)$
or
$z \sim$ U (-1, 1)
Generator
$-\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)$ *or* $\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)$

For D and G defined as neural networks, we can use backpropagation.

Optimal Solution

- For a given generator, the optimal discriminator is:

$$D^*_\phi(G) = \frac{p(x)}{p(x) + q_\theta(x)}.$$

- Incorporating that into the minimax game to yield virtual training criterion:

$$C(G) = \max_D V(G, D) =$$

$$= \mathbb{E}_{x\sim p(x)} \log(D^*_\phi(x)) + \mathbb{E}_{x\sim q_\theta(x)} \log(1 - D^*_\phi(x)) =$$

$$= \mathbb{E}_{x\sim P} \ \log \frac{p(x)}{p(x) + q(x)} + \mathbb{E}_{x\sim q_\theta(x)} \log \frac{q(x)}{p(x) + q(x)}$$

Lower Bound Reminder

- In case of ideal discriminator:

$$C(G) = \mathbb{E}_{x\sim P} \log \frac{p(x)}{p(x) + q(x)} + \mathbb{E}_{x\sim Q} \log \frac{q(x)}{p(x) + q(x)}$$

- This can be compared to variational bound:

$$\text{JS}(P||Q) \geq \mathbb{E}_{x\sim p(x)} \log \frac{2p(x)}{p(x)+q(x)} + \mathbb{E}_{x\sim q(x)} \log \frac{2q(x)}{p(x)+q(x)}$$

- Difference is only in $\log 4$

Optimal Solution

- In an optimal case $p = q_\theta$, we have $C(G) = -\log(4)$.
- We thus can write out:

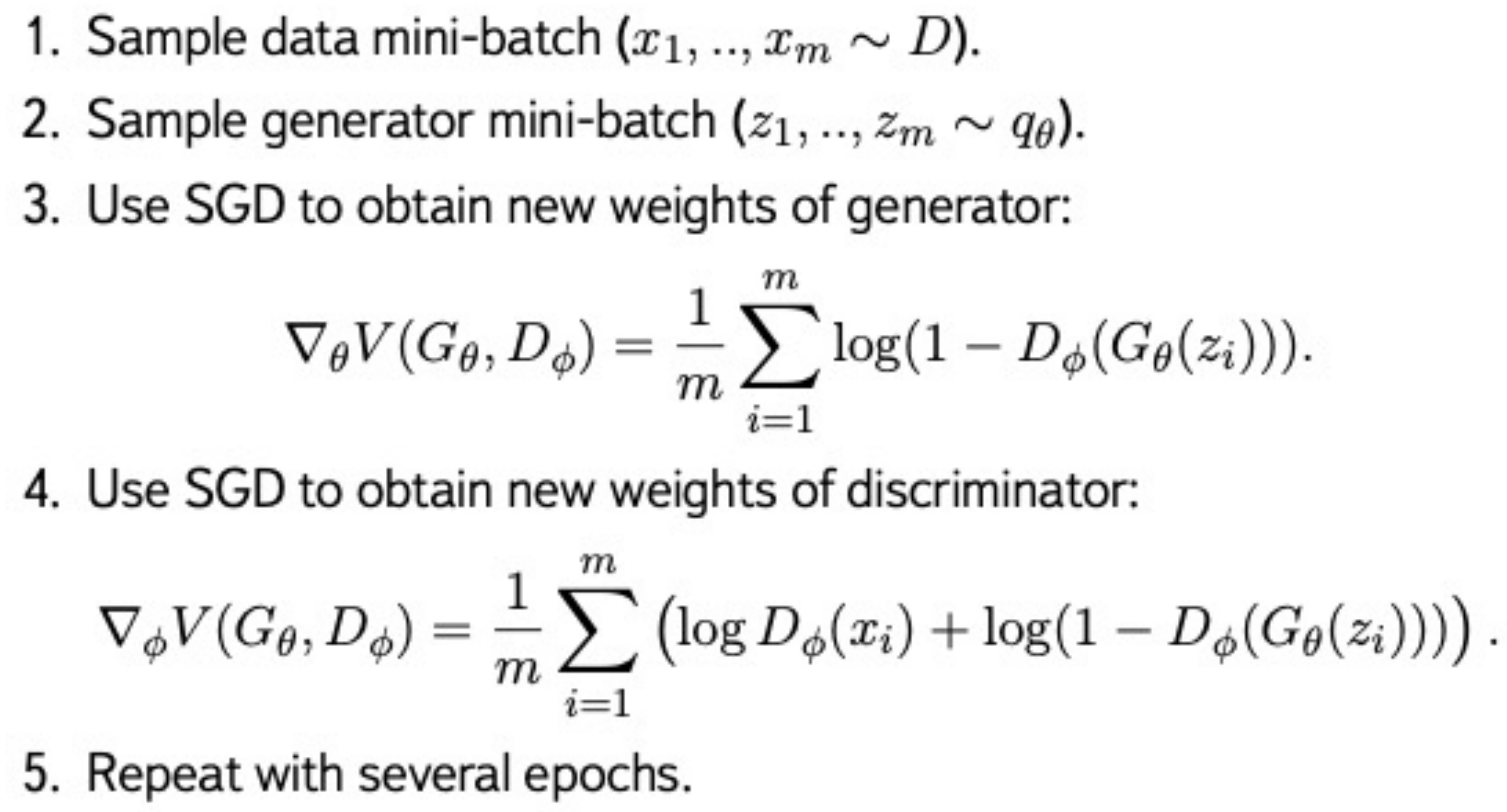
$$C(G) = -\log(4) + \frac{1}{2}KL(p(x)||\frac{p(x)+q_\theta(x)}{2}) + \\ + \frac{1}{2}KL(q_\theta(x)||\frac{p(x)+q_\theta(x)}{2}).$$

- In other words, we effectively optimize Jensen-Shannon divergence:

$$C(G) = -\log(4) + JS(p(x)||q_\theta(x)).$$

- Reminder: we did it without access to $p(x)$.

GAN algorithm



1. Sample data mini-batch $(x_1, .., x_m \sim D)$.
2. Sample generator mini-batch $(z_1, .., z_m \sim q_\theta)$.
3. Use SGD to obtain new weights of generator:

$$\nabla_\theta V(G_\theta, D_\phi) = \frac{1}{m} \sum_{i=1}^{m} \log(1 - D_\phi(G_\theta(z_i))).$$

4. Use SGD to obtain new weights of discriminator:

$$\nabla_\phi V(G_\theta, D_\phi) = \frac{1}{m} \sum_{i=1}^{m} \left(\log D_\phi(x_i) + \log(1 - D_\phi(G_\theta(z_i)))\right).$$

5. Repeat with several epochs.

GAN results

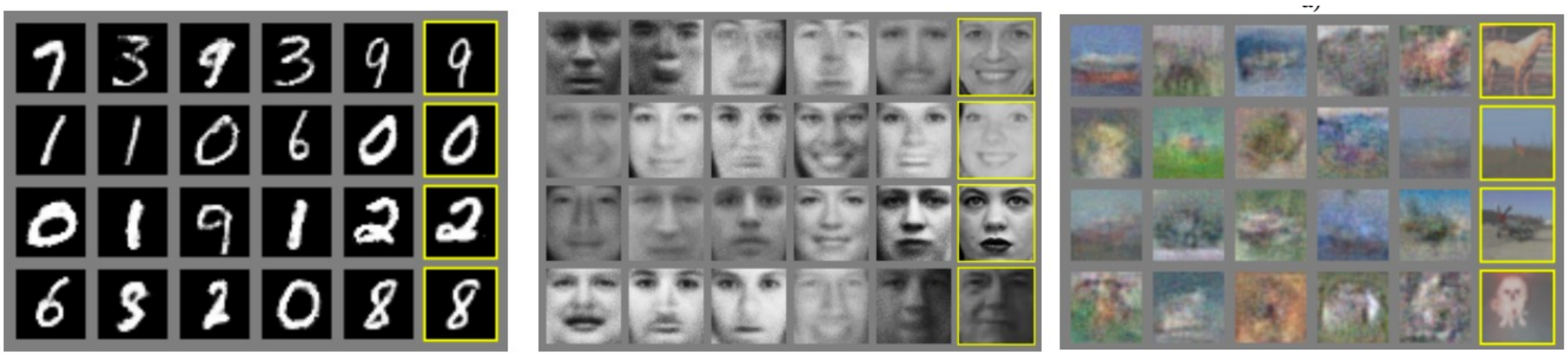


Figure 2: Visualization of samples from the model. Rightmost column shows the nearest training example of the neighboring sample, in order to demonstrate that the model has not memorized the training set. Samples are fair random draws, not cherry-picked. Unlike most other visualizations of deep generative models, these images show actual samples from the model distributions, not conditional means given samples of hidden units. Moreover, these samples are uncorrelated because the sampling process does not depend on Markov chain mixing. a) MNIST b) TFD c) CIFAR-10 (fully connected model) d) CIFAR-10 (convolutional discriminator and “deconvolutional” generator)

I. Goodfellow,et al. Generative Adversarial Networks, NIPS 2014

GAN First Paper Disclaimer

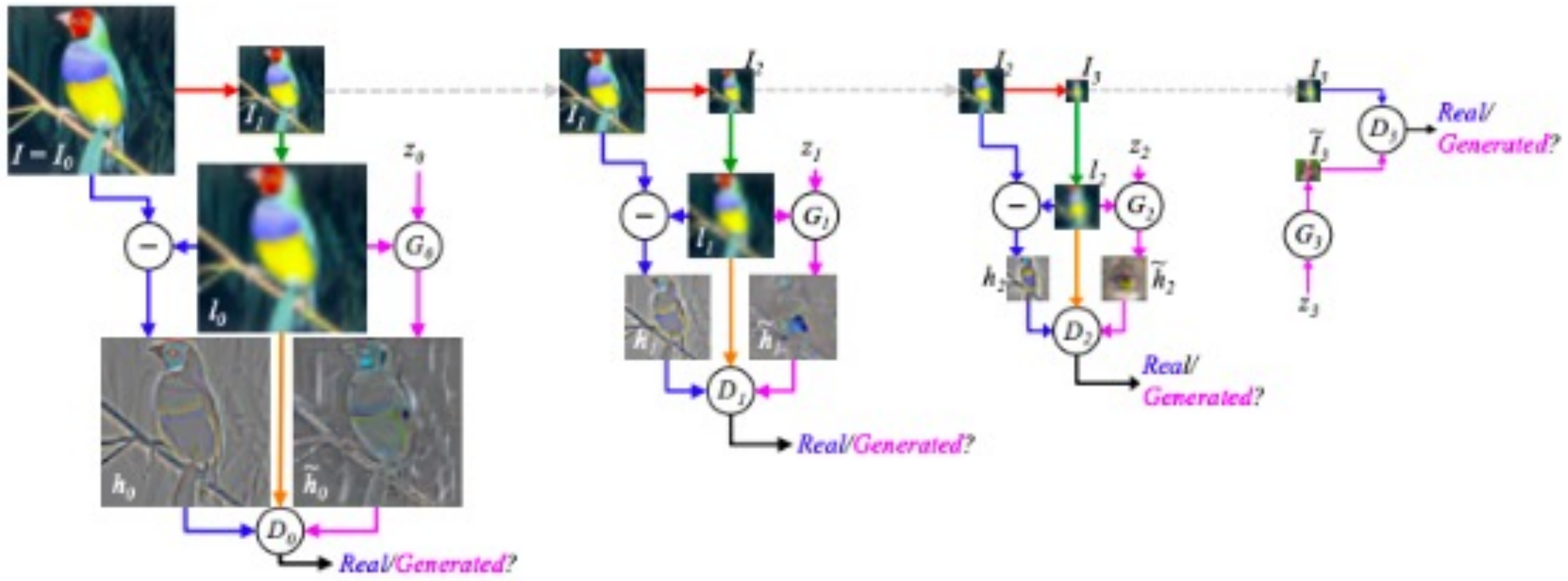
While we make no claim that these samples are better than samples generated by existing methods, we believe that these samples are at least competitive with the better generative models in the literature and highlight the potential of the adversarial framework.

I. Goodfellow, Generative Adversarial Networks, NIPS 2014

Enhancing GAN

Unsupervised Feature Learning

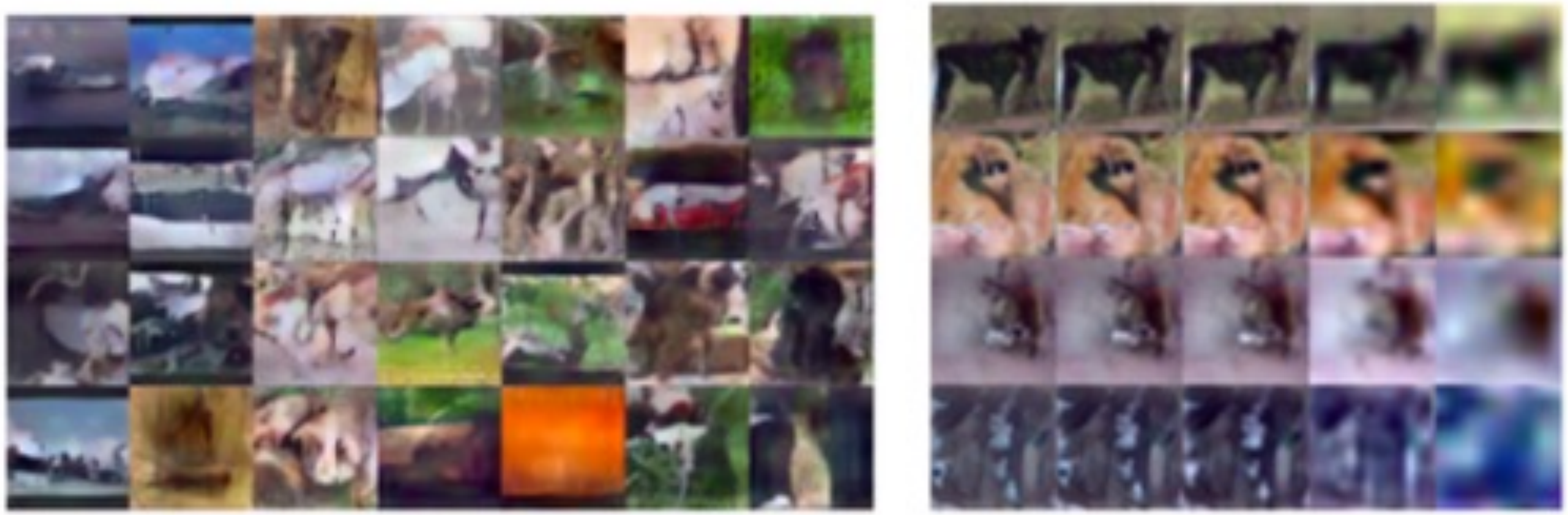
We need to decrease the size of images to make a faster convergence.
Let us make a Laplacian pyramid of images.



The representation is learned gradually.

E Denton et al. Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks

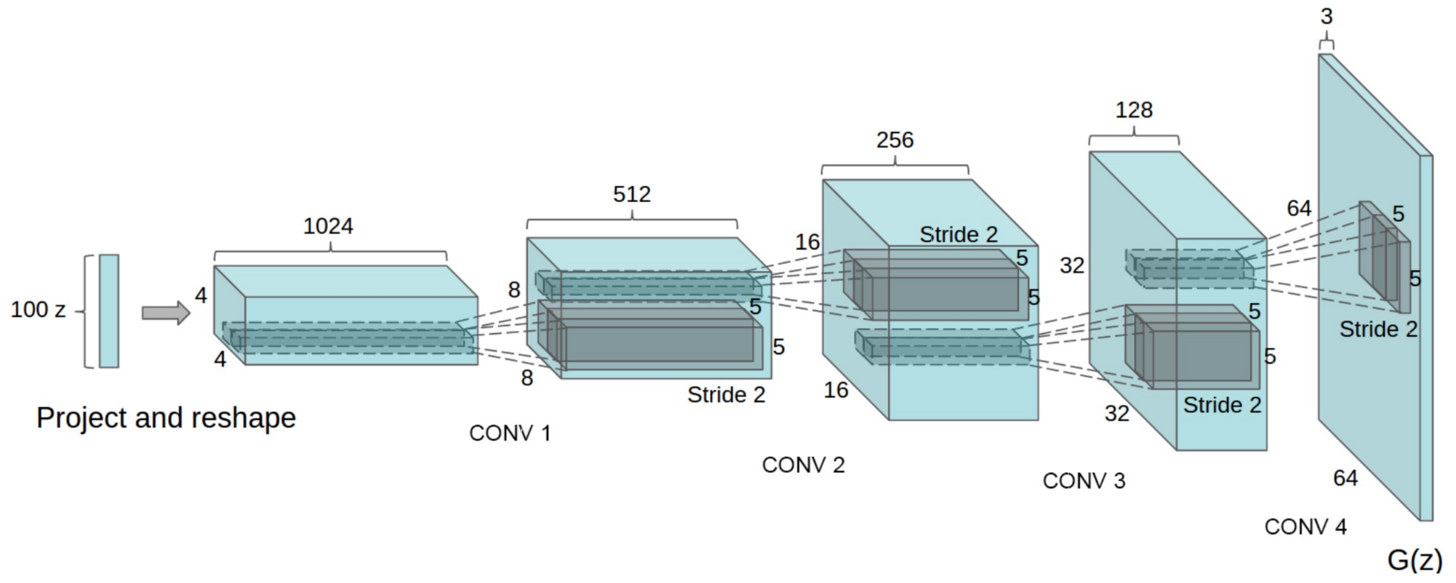
LAPGAN: results



(a) (b)

Figure 4: STL samples: **(a)** Random 96x96 samples from our LAPGAN model. **(b)** Coarse-to-fine generation chain.

Convolutional Layers are Here to Help



- pooling layers → convolution layers.
- Use batchnorm.
- No fully connected hidden layers.
- ReLU activation in generator.
- LeakyReLU activation in the discriminator for all layers.

A. Radford et al. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, ICLR 2016

DCGAN: results

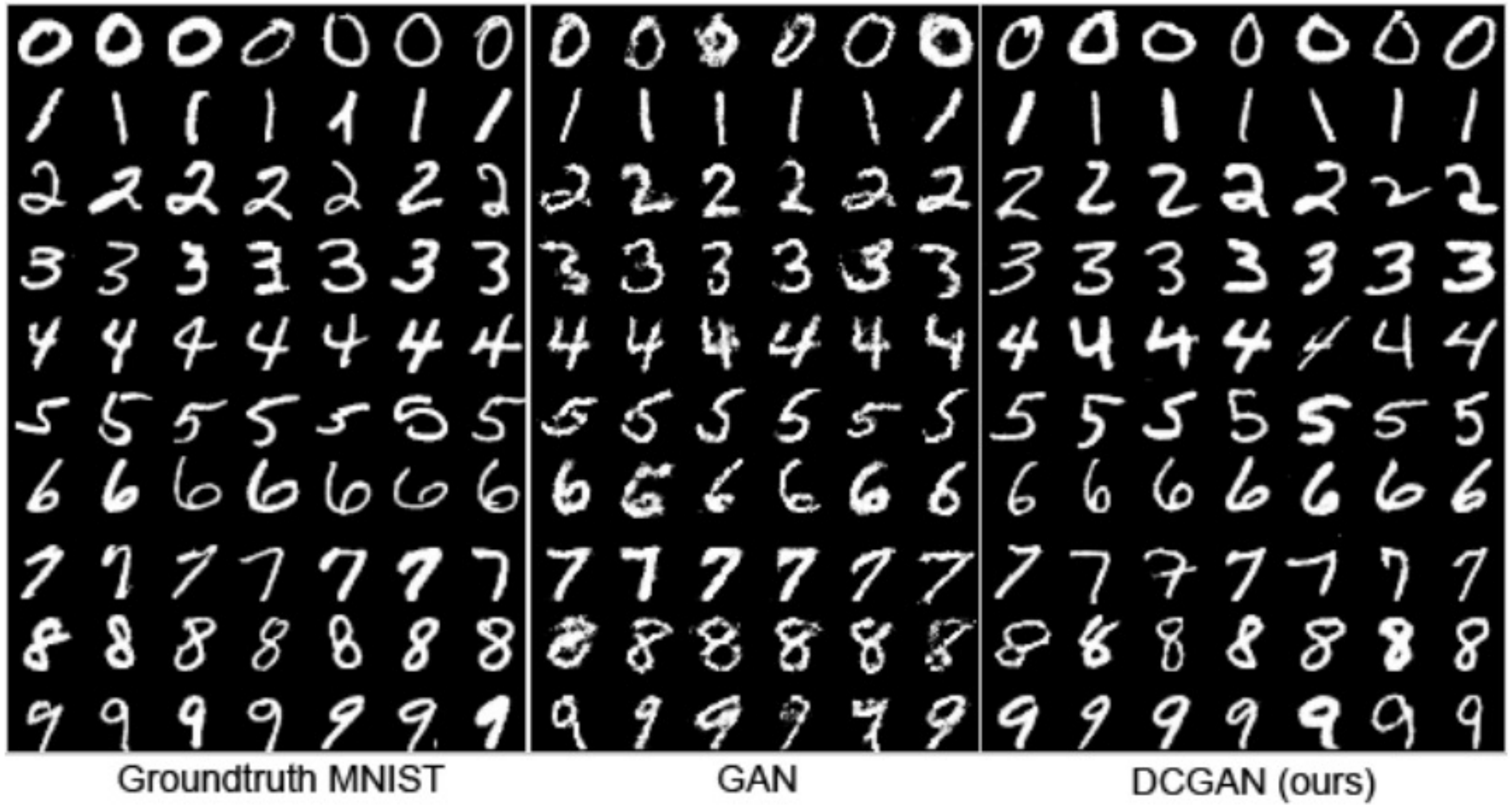
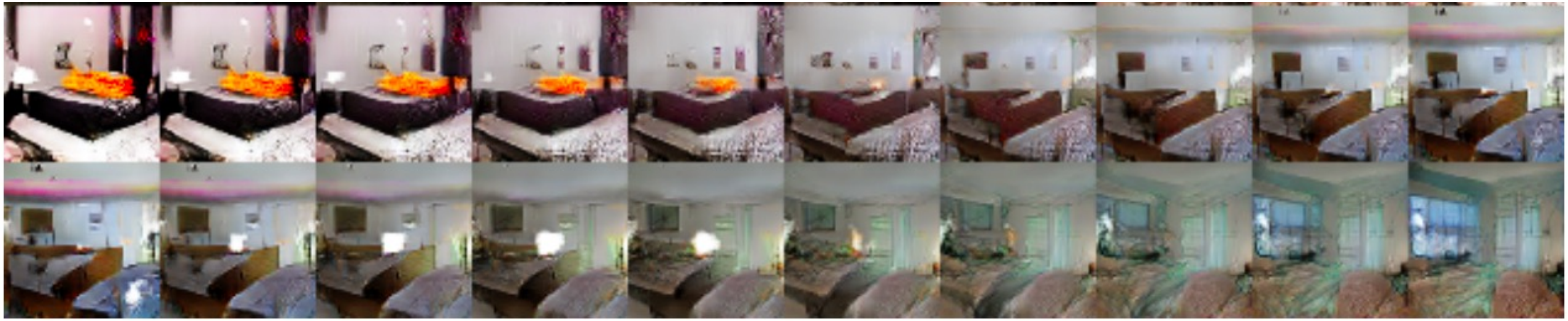


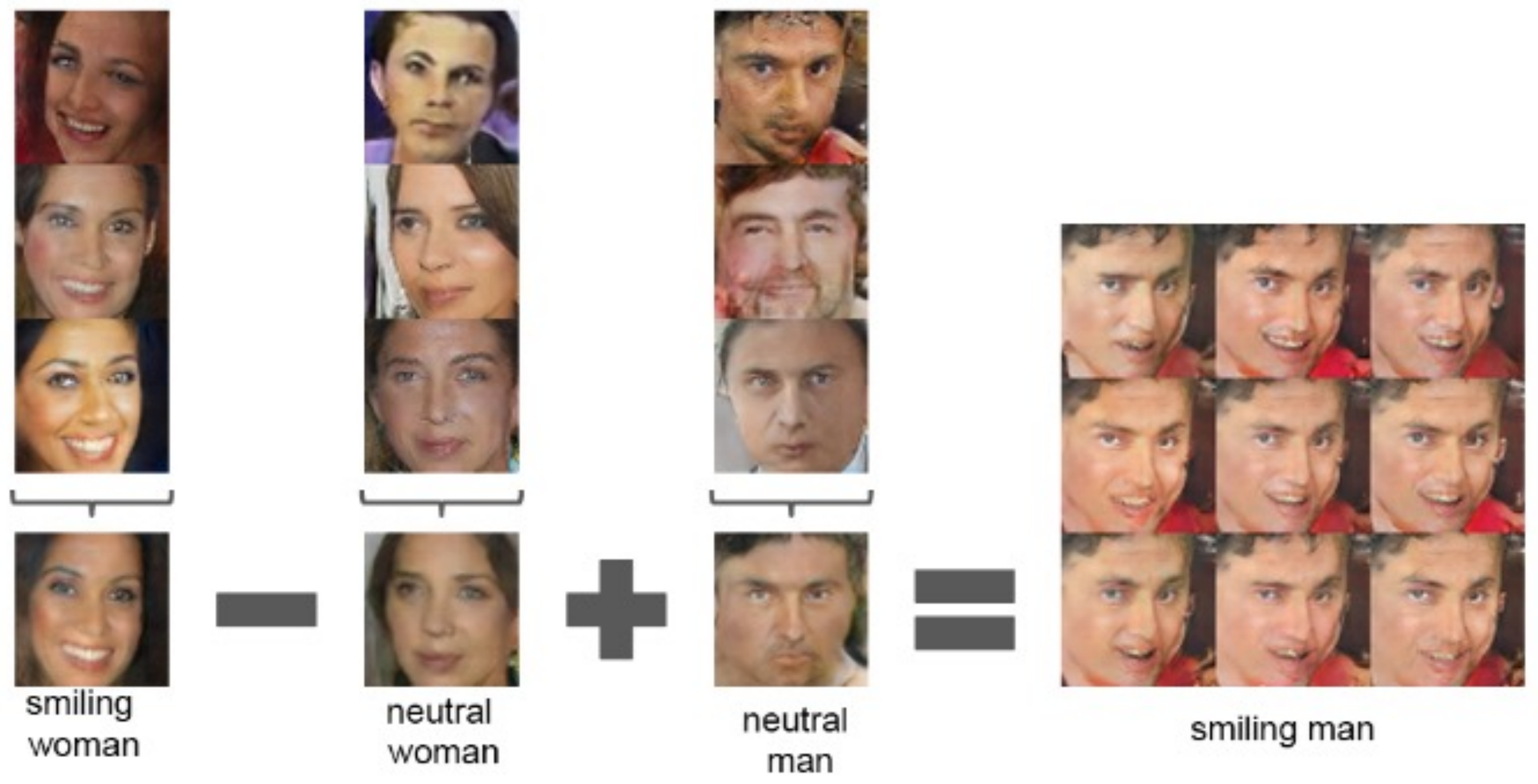
Figure 9: Side-by-side illustration of (from left-to-right) the MNIST dataset, generations from a baseline GAN, and generations from our DCGAN .

Walking in the Latent Space



TV $\longrightarrow$ Window

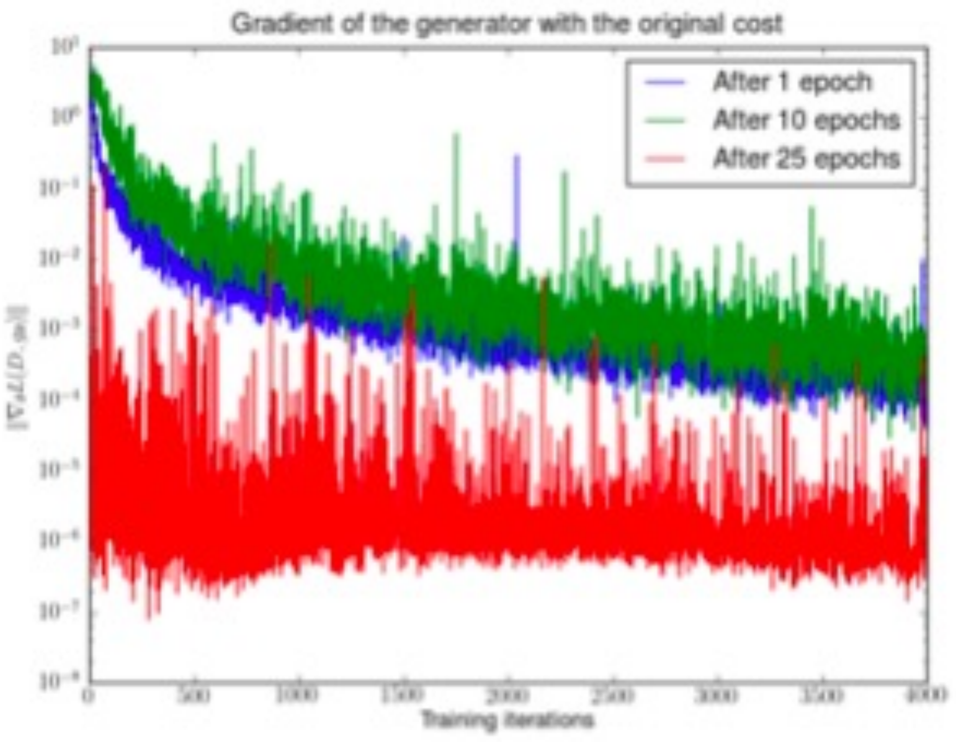
Arithmetic in the Latent Space



GAN problems

Game Approach Problems

- Discriminator must be optimal on every step of convergence.
- This is not true, you should not overtrain discriminator.
- Loss-function can be quite noisy.

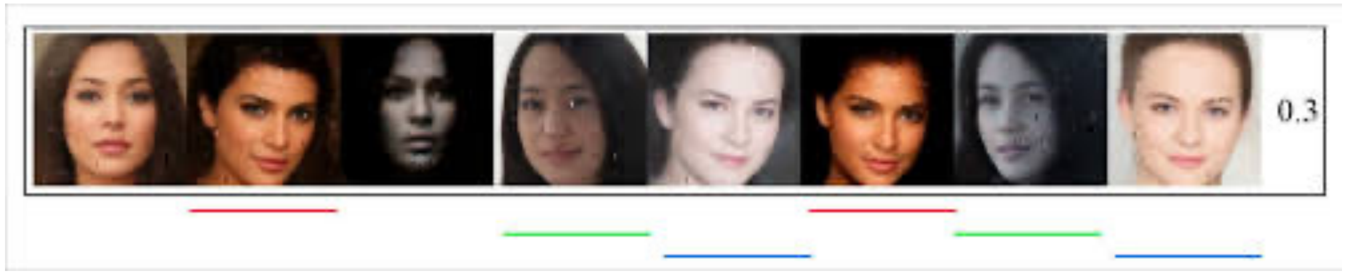


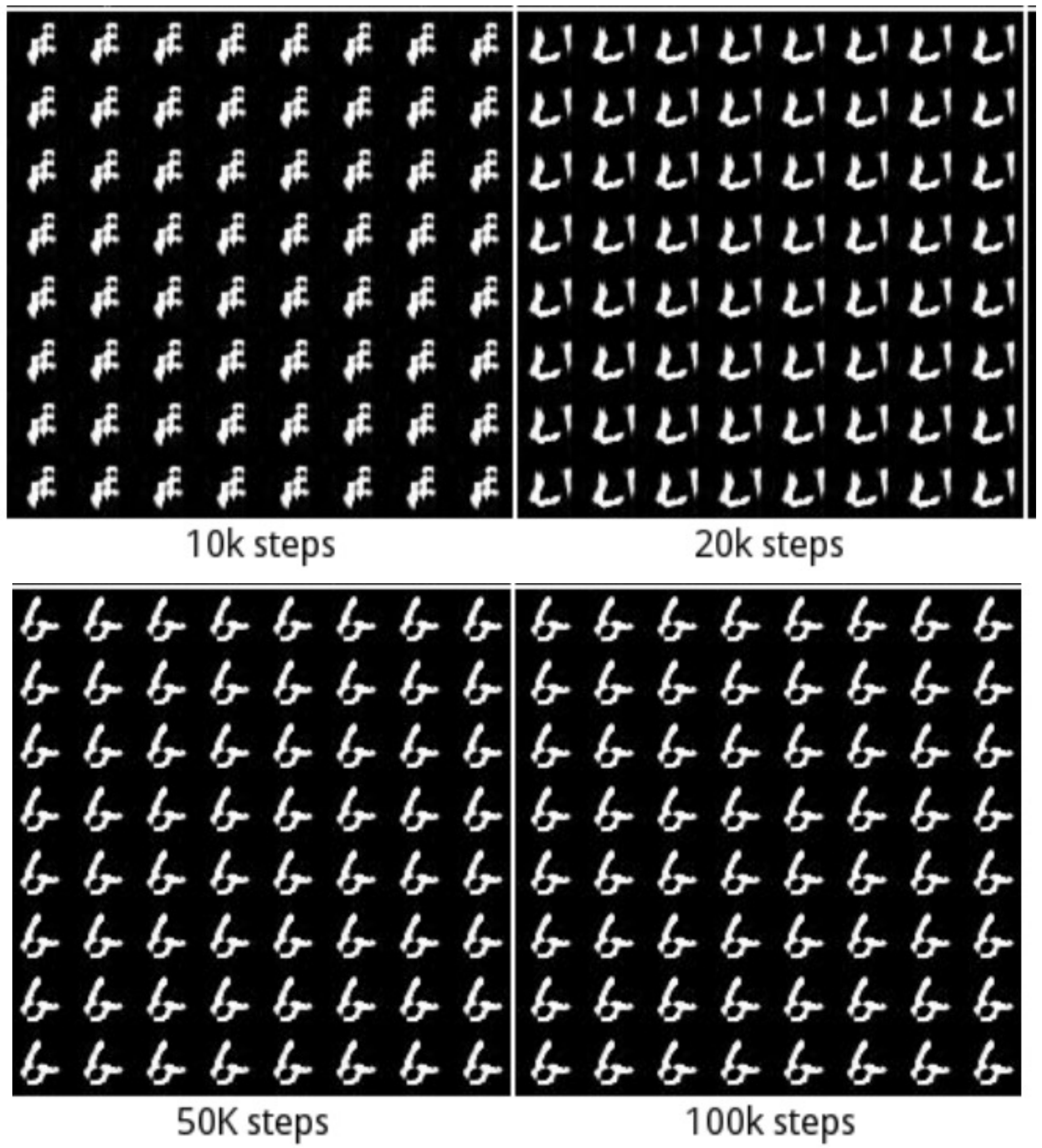
Martin Arjovsky, Towards Principled Methods for Training Generative Adversarial Networks , ICLR17

Mode Collapse

- GANs choose to generate a small number of modes due to a defect in the training procedure, rather than due to the divergence they aim to minimize.

I. Goodfellow NIPS 2016 Tutorial: Generative Adversarial Network



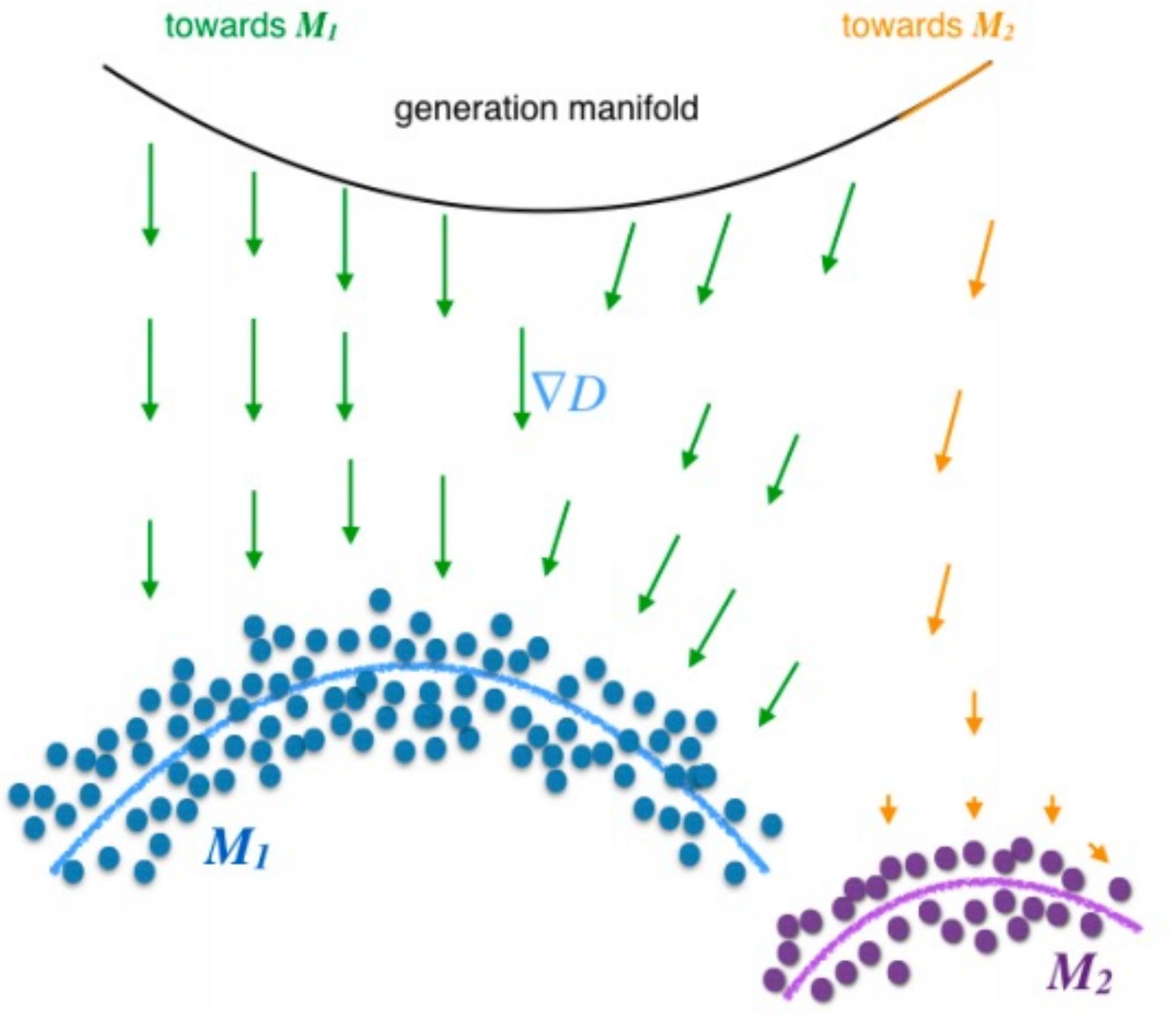


Luke Metz et al Unrolled Generative Adversarial Networks ICLR 2017

Mode Collapse

- For fixed D:
  - G tends to converge to a point x^* that fools D the most.
  - In extreme cases, G becomes independent on z.
  - Gradient on z diminishes.

- When D restarts:
  - Easily finds this x^*.
  - Pushes G to the next point x^{**}.

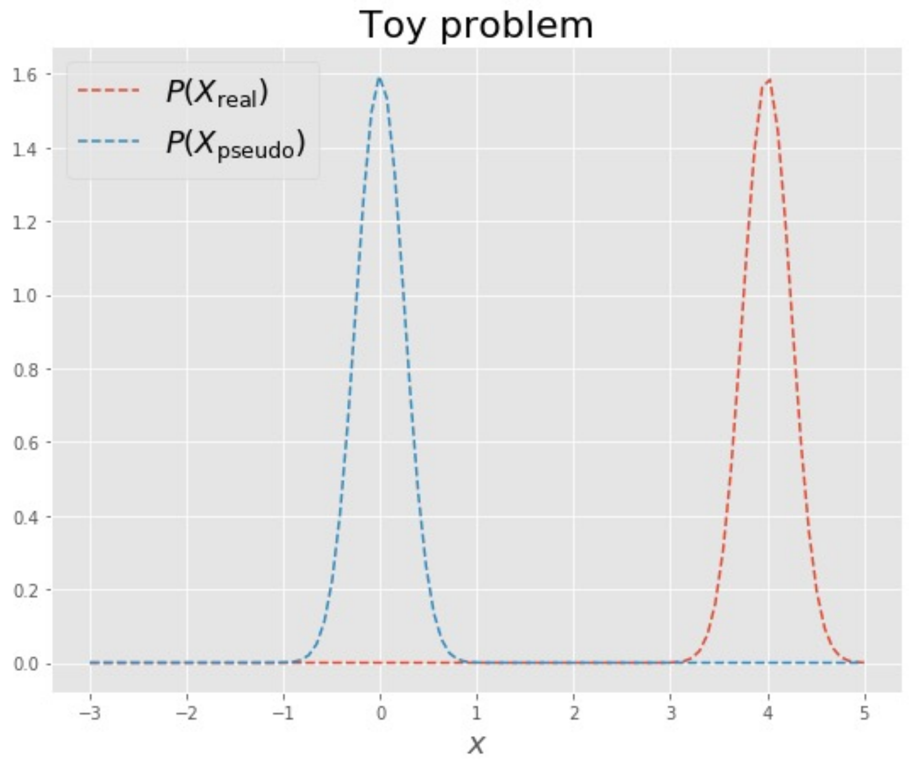


T. Che Mode Regularized Generative Adversarial Networks ICLR 2017

Vanishing Gradients

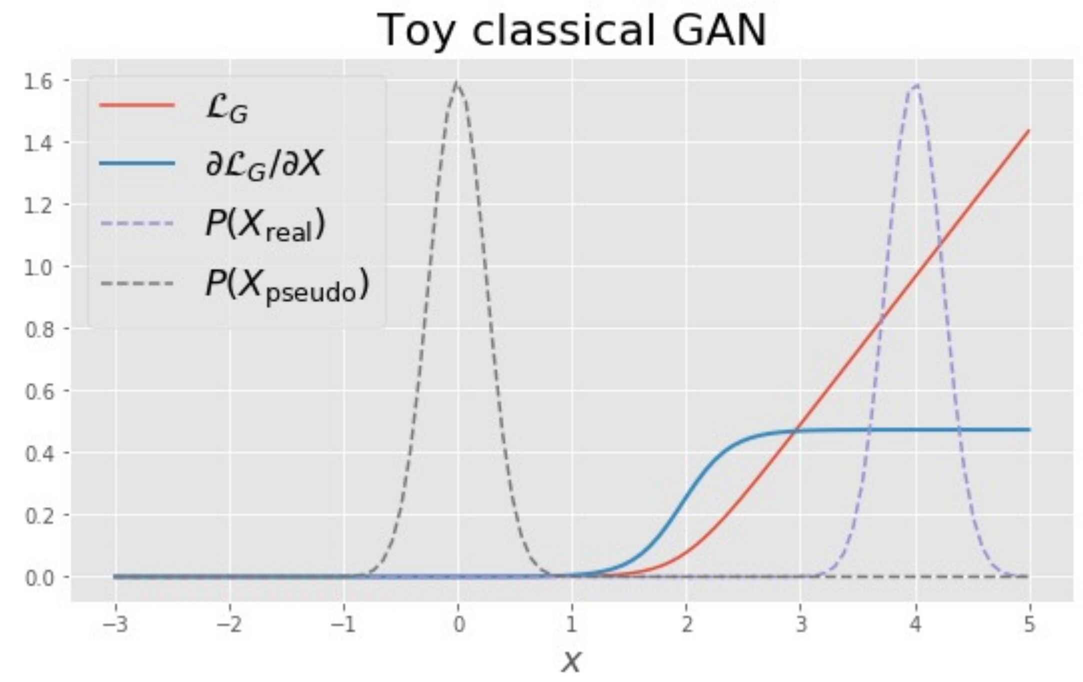
- For disjoint support of real and generated data

- An ideal discriminator can perfectly tell the real and generated data apart:
$$D(G(z)) \approx 0$$



Vanishing Gradients

- $L_{\mathrm{G}} = -\log D(G(z))$
- ${}^{dD(x)}/_{dx} \approx 0$ for generated x
- ${}^{dL_{\mathrm{G}}(x)}/_{dx} \approx 0$ for generated x
- Generator can't train!
- Need to start closer (how?)
- Problem is further enhanced due to noisy estimate from data.



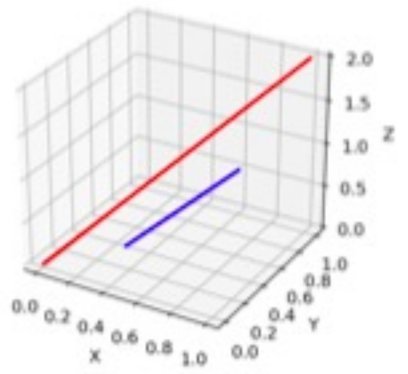
Summary so Far

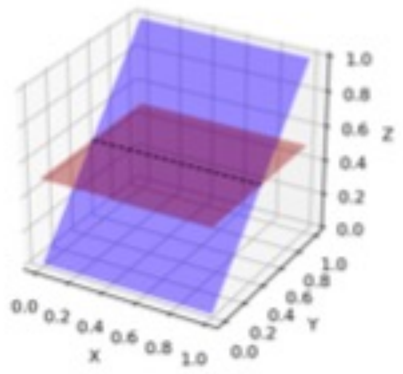
- Pros:
  - Can utilize power of back-prop.
  - No explicit intractable integral.
  - No MCMC needed.
- Cons:
  - Unclear stopping criteria.
  - No explicit representation of $g_\theta(x)$.
  - Hard to train.
  - No evaluation metric so hard to compare with other models.
  - Easy to get trapped in local optima that memorize training data.
  - Hard to invert generative model to get back latent z from generated x.

GAN ways out

Diminishing Gradients

- We have seen already that signal data is located on manifold.
- GAN case is in fact more complicated, as we need a discriminator that distinguishes two supports.
- This is way too easy, if supports are disjoint.



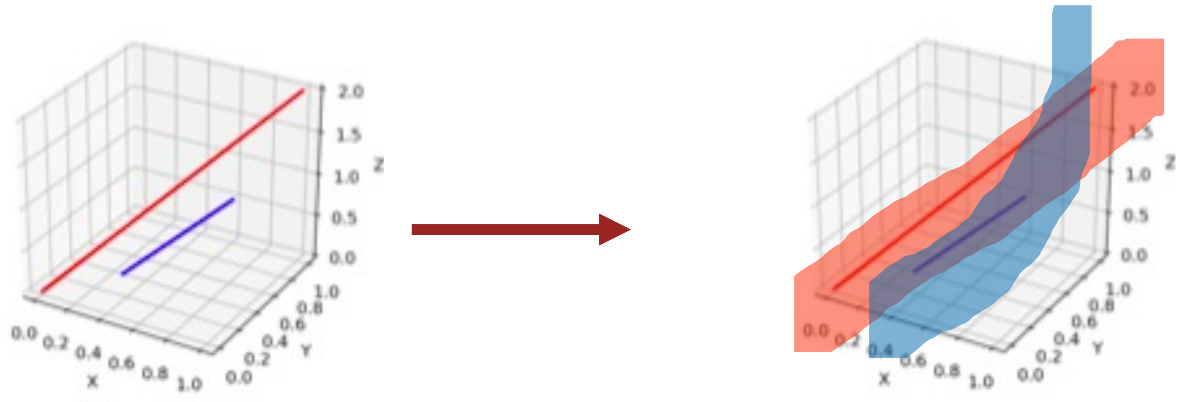


Diminishing Gradients: Noisy Supports

- Let’s make the problem harder: introduce random noise $\varepsilon \sim N(0; \sigma^2 I)$:

$$\mathbb{P}_{x+\varepsilon(x)} = \mathbb{E}_{y\sim P(x)}\mathbb{P}_{\varepsilon}(x - y).$$

- This will make noisy supports, that makes it difficult for discriminator.



Martin Arjovsky, Towards Principled Methods for Training Generative Adversarial Networks , ICLR17

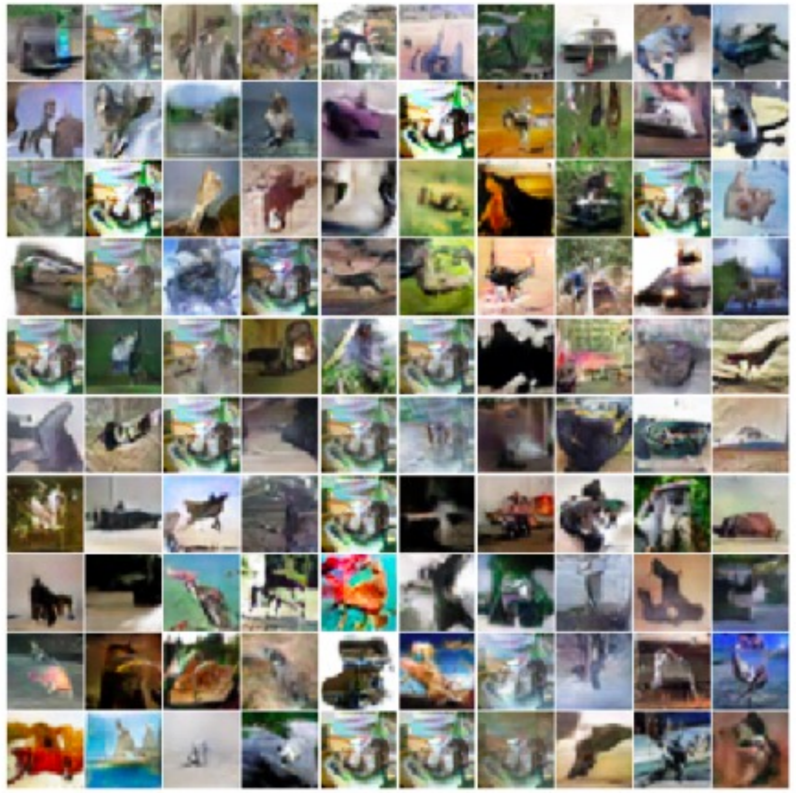
Feature Matching

- Change the objective of the generator:

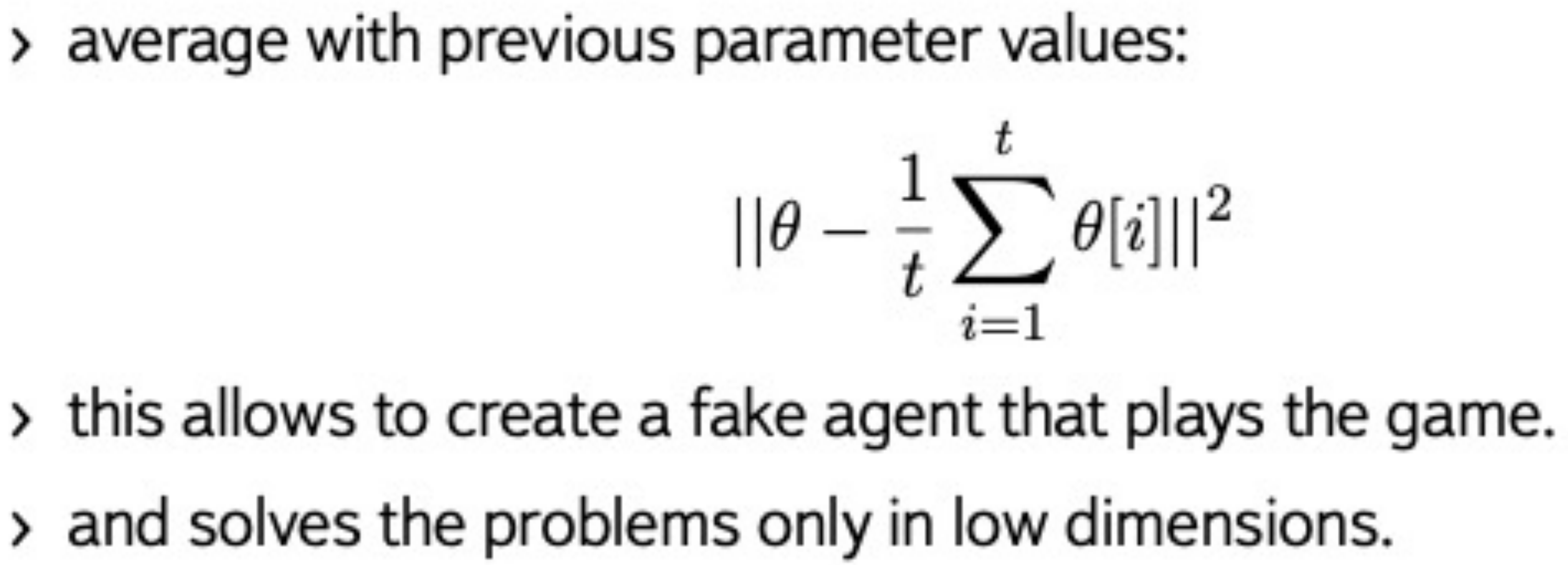
$$||\mathbb{E}_{x\sim p(x)} f(x) - \mathbb{E}_{z\sim p_z(z)} f(G(z))||^2$$

- Here $f(x)$ can be any property we need (including the output of another network.

Danger of overtrain to match known tests!



Historical averaging



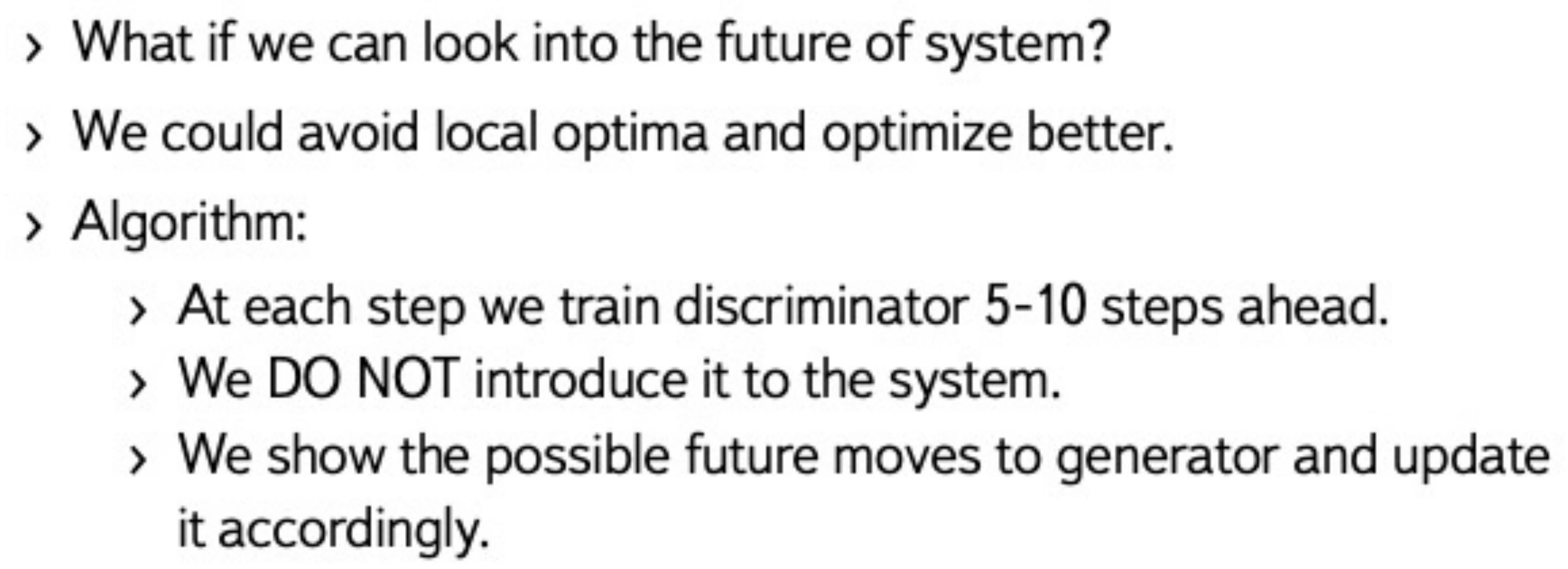
- average with previous parameter values:

$$||\theta - \frac{1}{t}\sum_{i=1}^{t}\theta[i]||^2$$

- this allows to create a fake agent that plays the game.
- and solves the problems only in low dimensions.

Salimans et al. Improved Techniques for Training GANs, NIPS16

Look into the Future: unrolled GANs



- What if we can look into the future of system?
- We could avoid local optima and optimize better.
- Algorithm:
  - At each step we train discriminator 5-10 steps ahead.
  - We DO NOT introduce it to the system.
  - We show the possible future moves to generator and update it accordingly.

Unrolled GAN: results

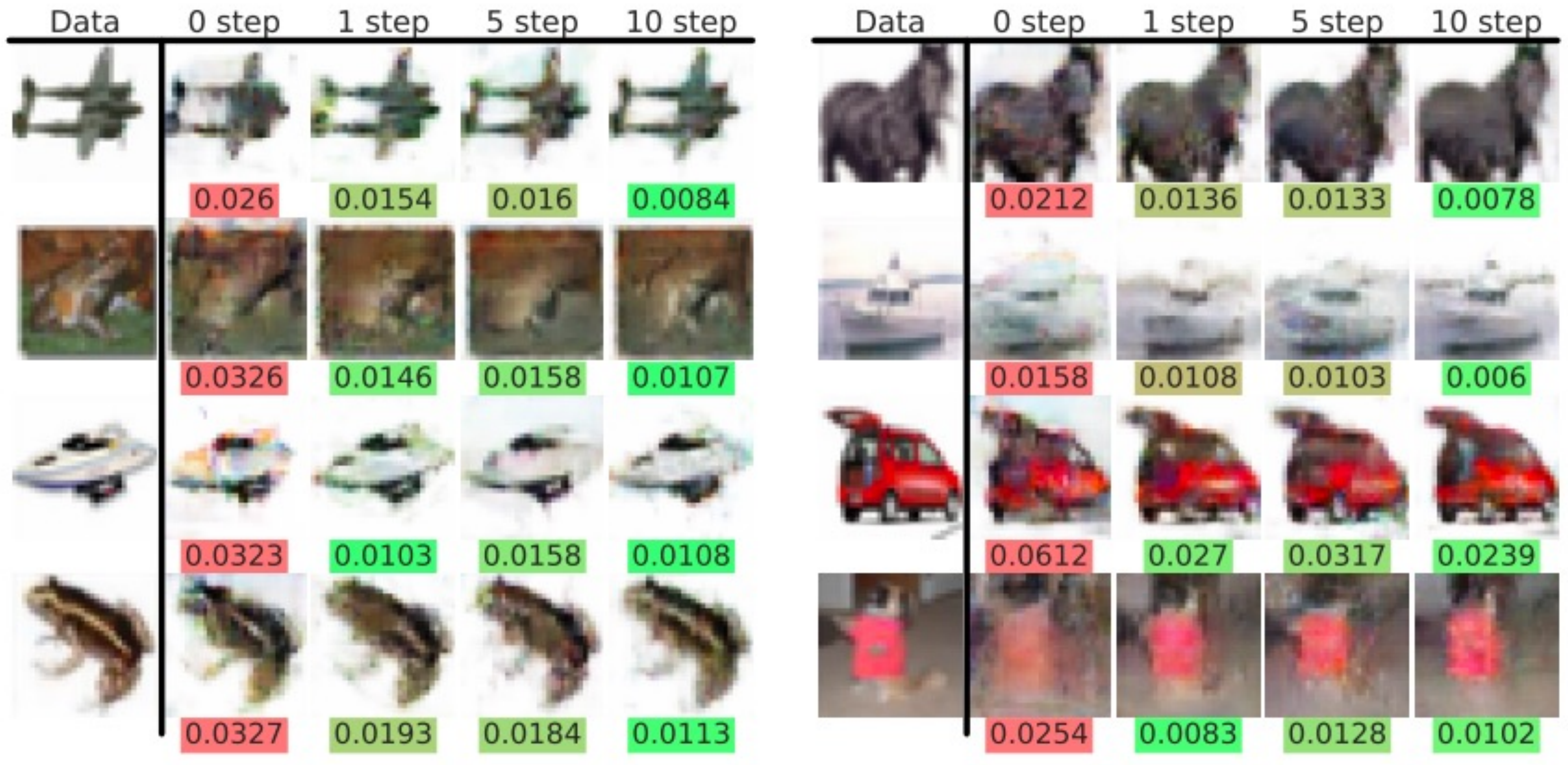


Figure 5: Training set images are more accurately reconstructed using GANs trained with unrolling than by a standard (0 step) GAN, likely due to mode dropping by the standard GAN. Raw data is on the left, and the optimized images to reach this target follow for 0, 1, 5, and 10 unrolling steps. The reconstruction MSE is listed below each sample. A random 1280 images where selected from the training set, and corresponding best reconstructions for each model were found via optimization. Shown here are the eight images with the largest absolute fractional difference between GANs trained with 0 and 10 unrolling steps.

Luke Metz et al Unrolled Generative Adversarial Networks ICLR 2017

Conclusions: GANs

- use Generator-Discriminator game to estimate the distance from generated distribution to the true one.
- produce sharp images.
- reconstruct implicit model of target PDF.