Generative adversarial networks

Biplab Banerjee

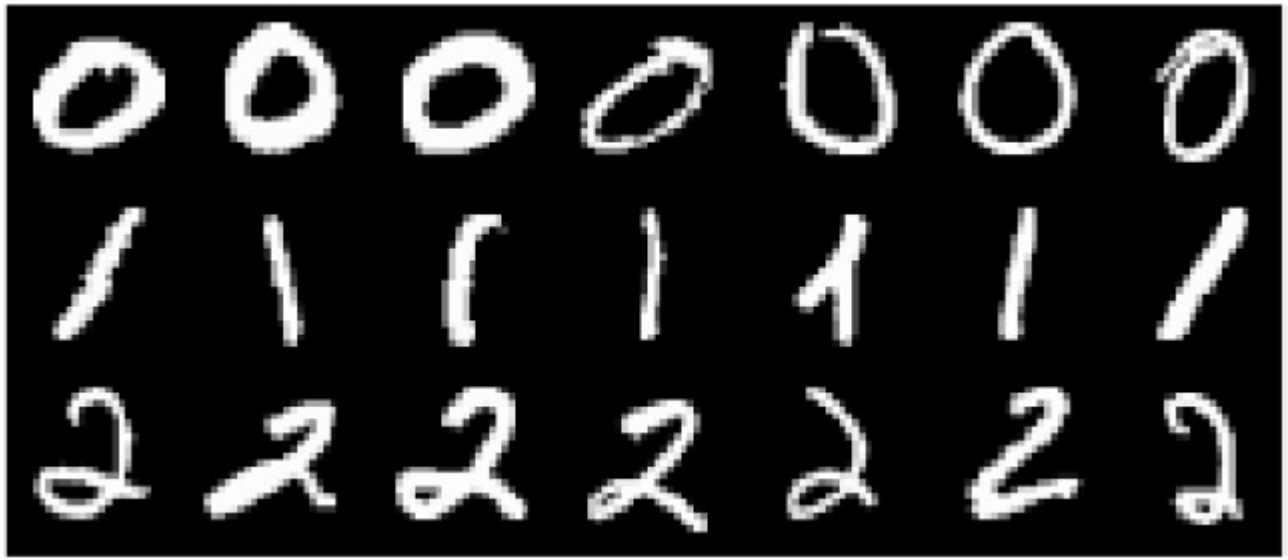
- **We've only seen discriminative models so far**
  - Given an image **X**, predict a label **Y**
  - Estimates **P(Y|X)**

- **Discriminative models have several key limitations**
  - Can't model **P(X)**, i.e. the probability of seeing a certain image
  - Thus, can't sample from **P(X)**, i.e. **can't generate new images**

- **Generative models (in general) cope with all of above**
  - Can model **P(X)**
  - Can generate new images

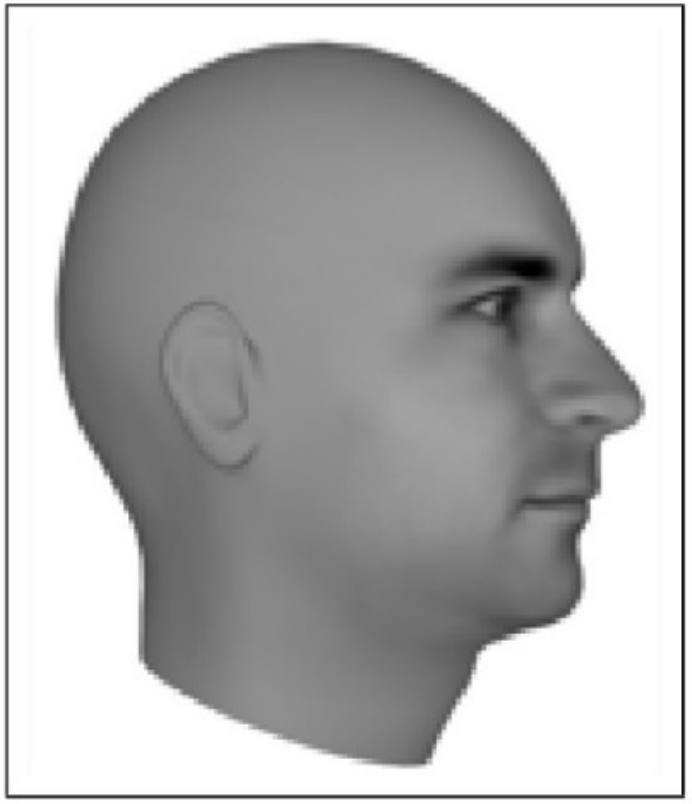
Implicit density estimation problem

- What if we are only interested to sample from the complex high dimensional intractable data distribution without caring to know *P(data)* at all?

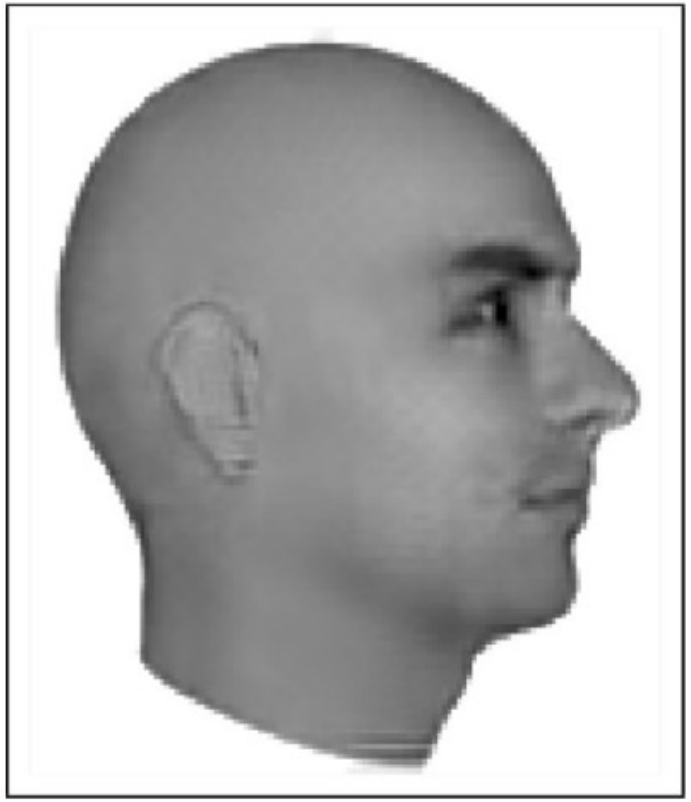


Magic of GAN

Ground Truth



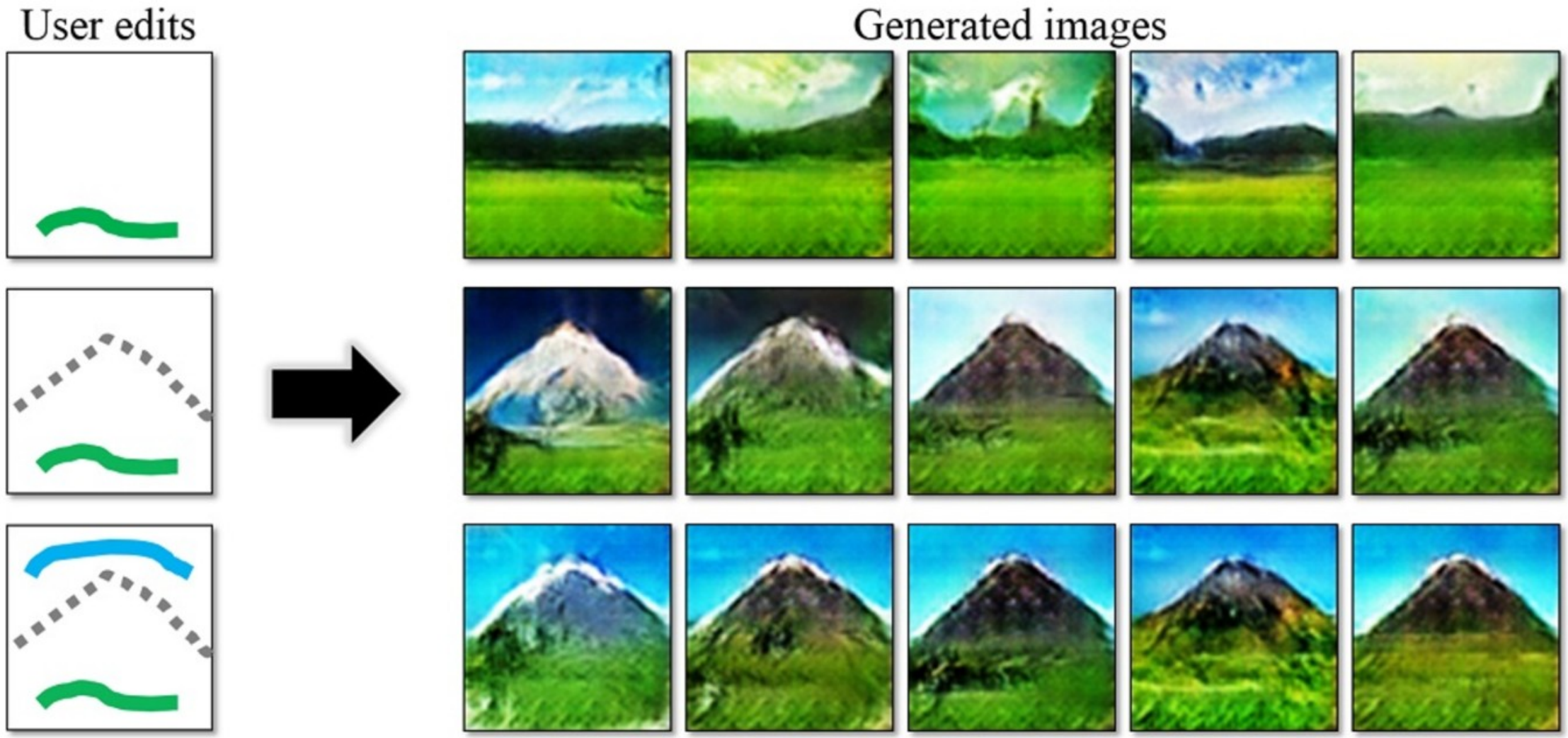
Adversarial



Lotter, William, Gabriel Kreiman, and David Cox. "Unsupervised learning of visual structure using predictive generative networks." *arXiv preprint arXiv:1511.06380* (2015).

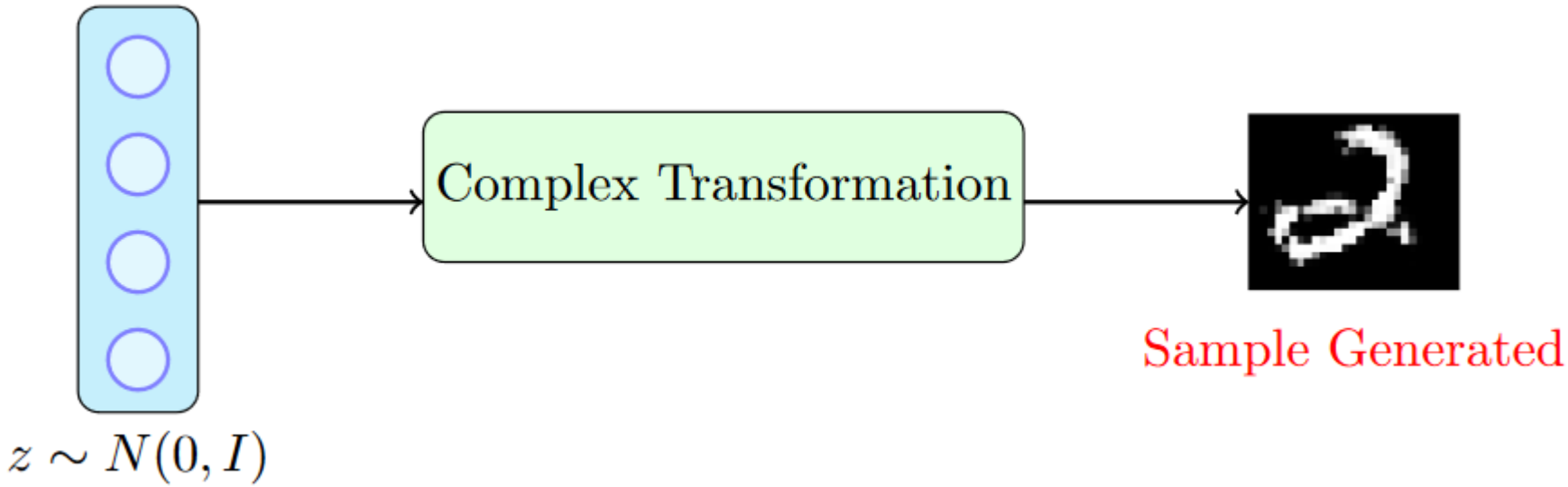
Magic of GAN

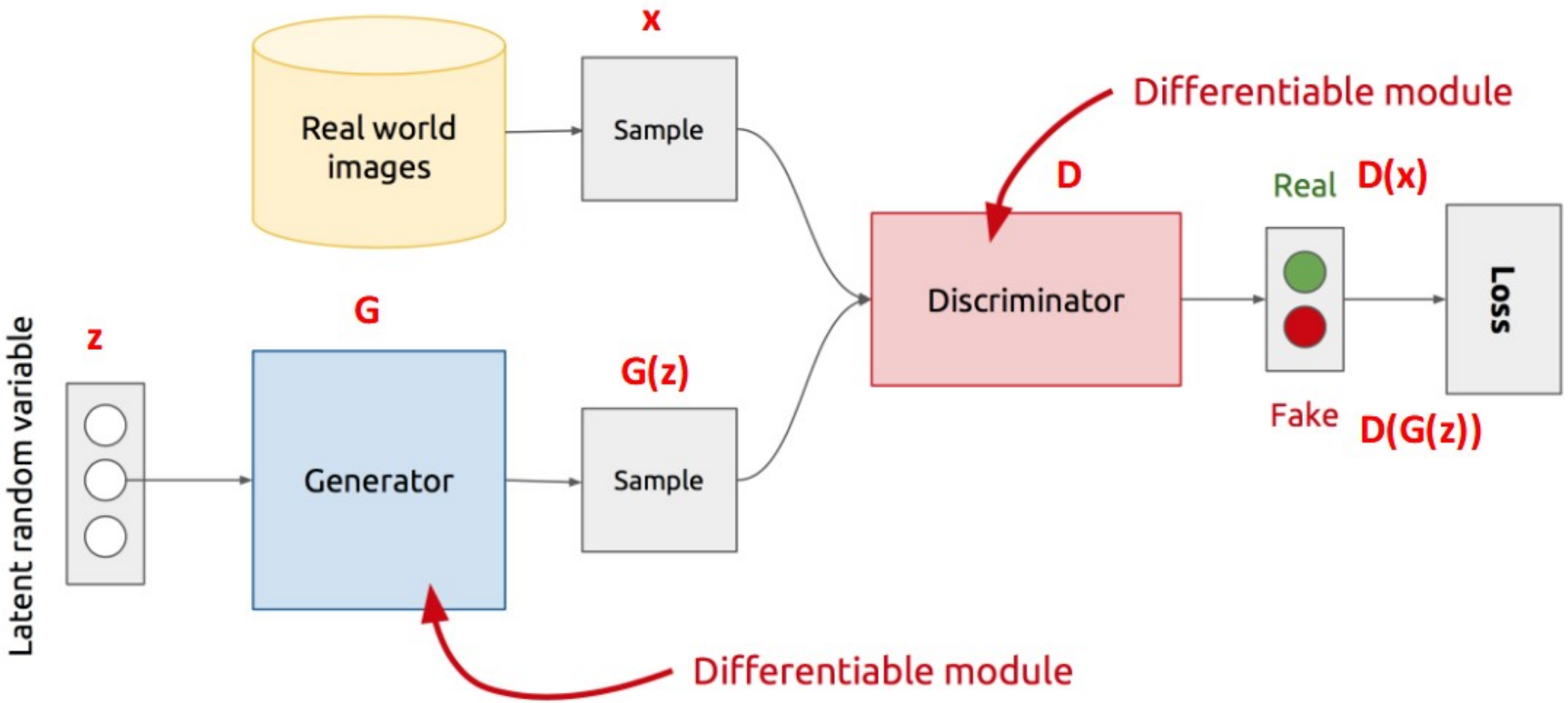
Which one is Computer generated?



http://people.eecs.berkeley.edu/~junyanz/projects/gvm/

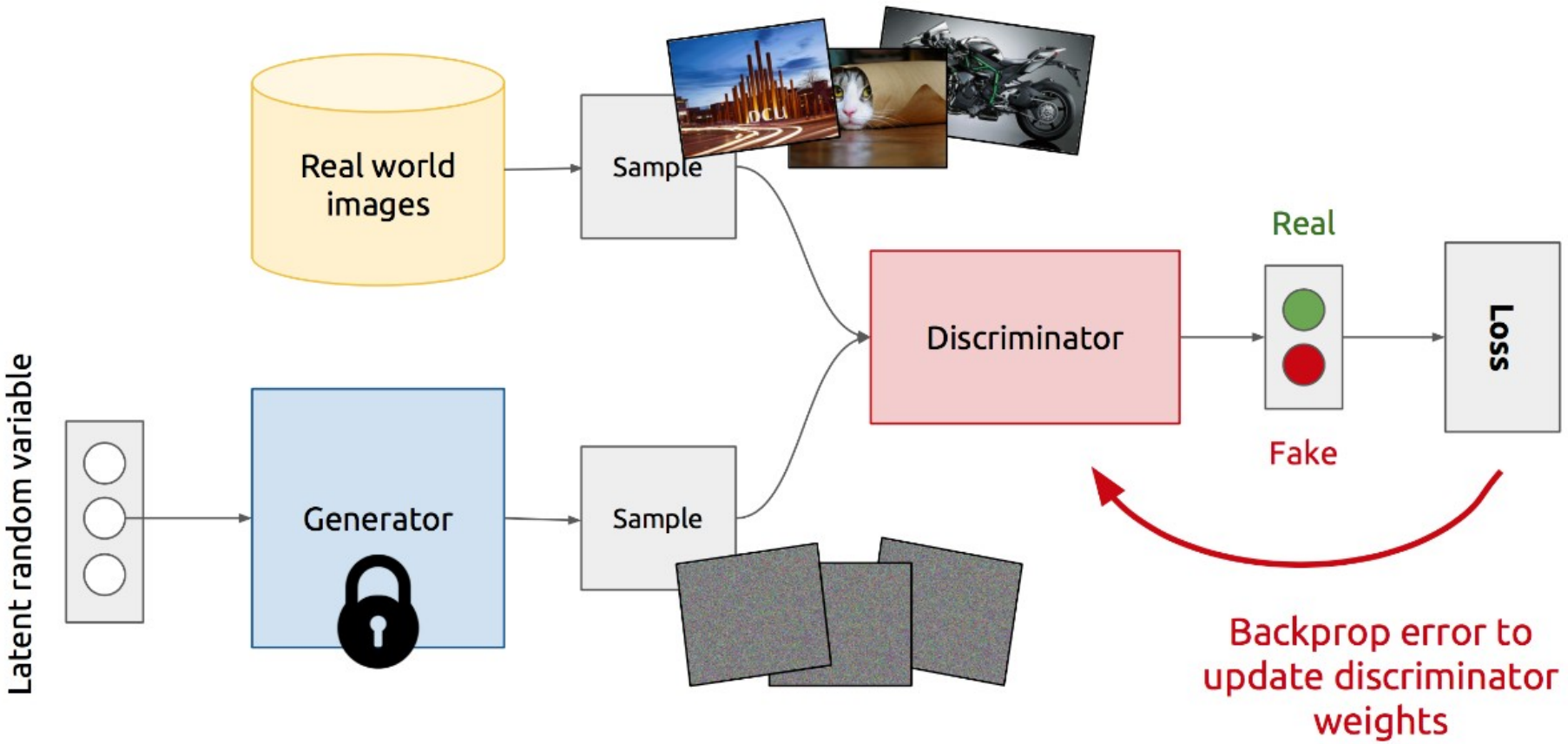
What about



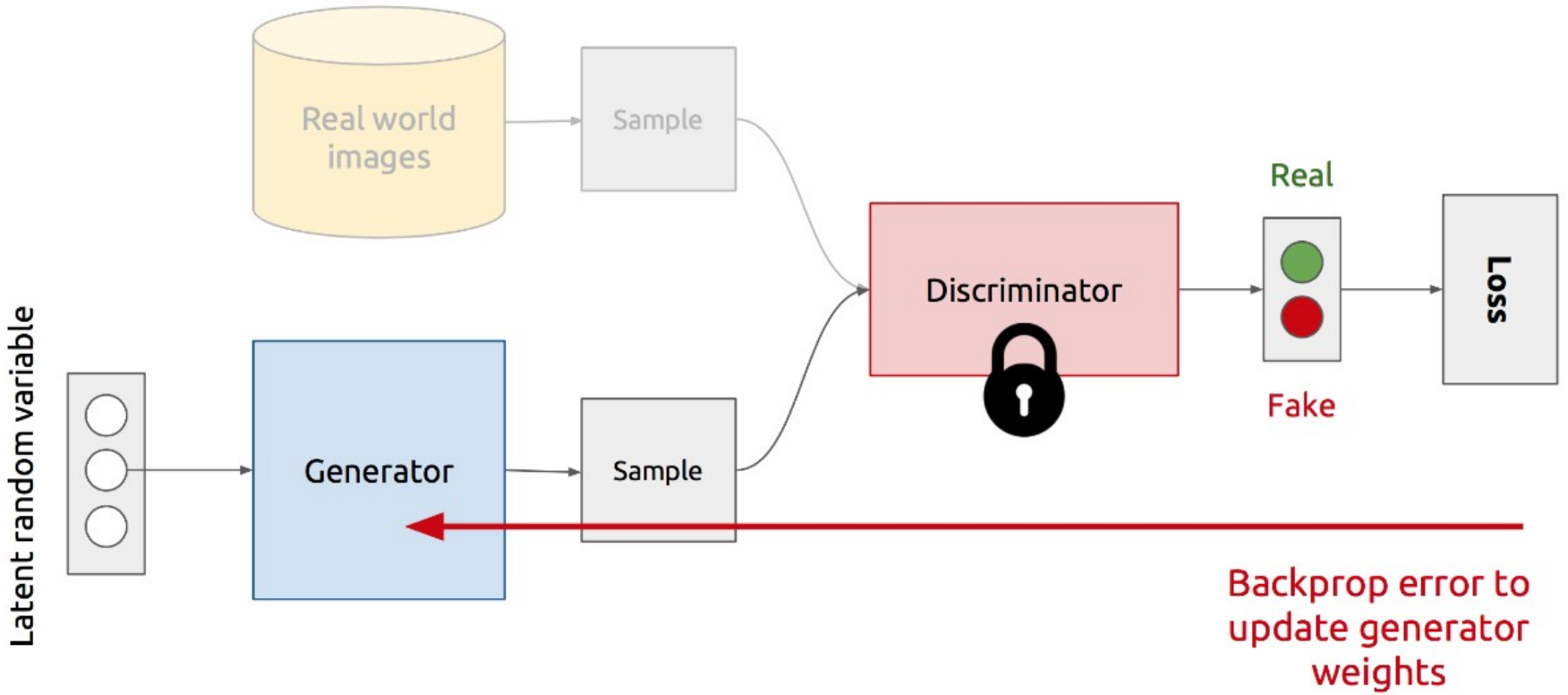


- **Z** is some random noise (Gaussian/Uniform).
- **Z** can be thought as the latent representation of the image.

Training discriminator



Training generator



Let G_ϕ be the generator and D_θ be the discriminator (ϕ and θ are the parameters of G and D, respectively)

We have a neural network based generator which takes as input a noise vector $z \sim N(0, I)$ and produces $G_\phi(z) = X$

We have a neural network based discriminator which could take as input a real X or a generated $X = G_\phi(z)$ and classify the input as real/fake

Given an image generated by the generator as $G_\phi(z)$ the discriminator assigns a score $D_\theta(G_\phi(z))$ to it

This score will be between 0 and 1 and will tell us the probability of the image being real or fake

For a given z, the generator would want to maximize $\log D_\theta(G_\phi(z))$ (log likelihood) or minimize $\log(1 - D_\theta(G_\phi(z)))$

The task of the discriminator is to assign a high score to real images and a low score to fake images

And it should do this for all possible real images and all possible fake images

Loss functions

- For generator

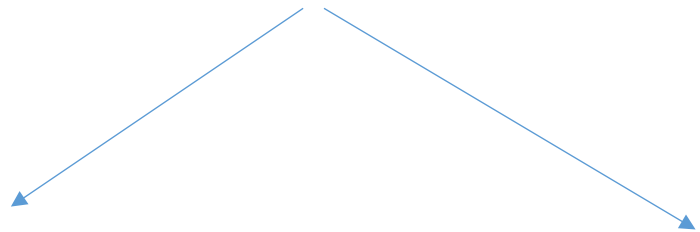
$$\min_{\phi} E_{z \sim p(z)}[\log(1 - D_\theta(G_\phi(z)))]$$

- For discriminator

$$\max_{\theta} E_{x \sim p_{data}}[\log D_\theta(x)] + E_{z \sim p(z)}[\log(1 - D_\theta(G_\phi(z)))]$$

Loss function

$$\min_{\phi} \max_{\theta} \quad [\mathbb{E}_{x \sim p_{data}} \log D_\theta(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_\theta(G_\phi(z)))]$$



Generator minimizes it

Discriminator maximizes it

Training a vanilla GAN

Step 1: Gradient Ascent on Discriminator

$$\max_{\theta} \; [\mathbb{E}_{x \sim p_{data}} \log D_\theta(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_\theta(G_\phi(z)))]$$

Step 2: Gradient Descent on Generator

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

procedure GAN TRAINING

  for number of training iterations **do**

    for k steps **do**

      • Sample minibatch of m noise samples $\{\mathbf{z}^{(1)}, .., \mathbf{z}^{(m)}\}$ from noise prior $p_g(\mathbf{z})$

      • Sample minibatch of m examples $\{\mathbf{x}^{(1)}, .., \mathbf{x}^{(m)}\}$ from data generating distribution $p_{data}(\mathbf{x})$

      • Update the discriminator by ascending its stochastic gradient:

$$\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} \left[\log D_\theta \left(x^{(i)} \right) + \log \left(1 - D_\theta \left(G_\phi \left(z^{(i)} \right) \right) \right) \right]$$

    end for

    • Sample minibatch of m noise samples $\{\mathbf{z}^{(1)}, .., \mathbf{z}^{(m)}\}$ from noise prior $p_g(\mathbf{z})$

    • Update the generator by ascending its stochastic gradient

$$\nabla_\phi \frac{1}{m} \sum_{i=1}^{m} \left[\log \left(D_\theta \left(G_\phi \left(z^{(i)} \right) \right) \right) \right]$$

  end for

end procedure

Tackling vanishing gradients

$$\min_G \max_D V(D,G)$$

$$V(D,G) = \mathbb{E}_{x\sim p(x)}[\log D(x)] + \boxed{\mathbb{E}_{z\sim q(z)}\left[\log\left(1 - D(G(z))\right)\right]}$$

$$\nabla_{\theta_G} V(D,G) = \nabla_{\theta_G} \mathbb{E}_{z\sim q(z)}\left[\log\left(1 - D\left(G(z)\right)\right)\right]$$

- $\nabla_a \log\left(1-\sigma(a)\right) = \frac{-\nabla_a \sigma(a)}{1-\sigma(a)} = \frac{-\sigma(a)\left(1-\sigma(a)\right)}{1-\sigma(a)} = -\sigma(a) = -D\left(G(z)\right)$
- Gradient goes to 0 if D is confident, i.e. $D\left(G(z)\right) \to 0$

- Minimize $\boxed{-\mathbb{E}_{z\sim q(z)}\left[\log D\left(G(z)\right)\right]}$ for **Generator** instead (keep Discriminator as it is)

- **Discriminator Loss:**
  $\max_D \ \mathbb{E}_{x \sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))]$
- **Generator Loss (Non-Saturating):**
  $\min_G \ -\mathbb{E}_{z \sim p_z}[\log(D(G(z)))]$

- For instance, if $D(G(z)) = 0.01$, then $\log(0.01) \approx -4.605$.
- The generator's objective $-\log(D(G(z)))$ is therefore $\approx +4.605$, which is **high**; pushing it down yields a large gradient.
- In contrast, in the saturating version, $\log(1 - 0.01) = \log(0.99) \approx -0.01005$. That's nearly zero and doesn't push as hard.

Optimal D

For a fixed generator G, the distribution p_g is fixed. We want:

$$\max_D \int \Big[p_r(x)\, \log D(x) \;+\; p_g(x)\, \log\big(1 - D(x)\big) \Big]\, dx.$$

Inside the integral, each x is independent in terms of $D(x)$. Define:

$$f(D(x)) \;=\; p_r(x)\, \log\big[D(x)\big] \;+\; p_g(x)\, \log\big[1 - D(x)\big].$$

Take the derivative w.r.t. $D(x)$ and set to zero:

$$\frac{\partial f}{\partial D(x)} = \frac{p_r(x)}{D(x)} \;-\; \frac{p_g(x)}{1 - D(x)} = 0 \;\;\Longrightarrow\;\; p_r(x)\,\big[1 - D(x)\big] = p_g(x)\,\big[D(x)\big].$$

Solving for $D(x)$ gives the **optimal** (pointwise) discriminator:

$$D^*(x) \;=\; \frac{p_r(x)}{p_r(x) + p_g(x)}.$$

Plug $D^*(x)$ back into $V(D, G)$:

$$\max_D V(D,G) = \int \left[p_r(x)\, \log\!\Big(\tfrac{p_r(x)}{p_r(x)+p_g(x)}\Big) \;+\; p_g(x)\, \log\!\Big(\tfrac{p_g(x)}{p_r(x)+p_g(x)}\Big) \right] dx.$$

Global optima

$$\mathrm{JS}(P \parallel Q) \;=\; \tfrac{1}{2}\,\mathrm{KL}\Big(P \parallel \tfrac{P+Q}{2}\Big) \;+\; \tfrac{1}{2}\,\mathrm{KL}\Big(Q \parallel \tfrac{P+Q}{2}\Big),$$

where KL denotes the Kullback–Leibler divergence.

Let $P = p_r$ and $Q = p_g$. Then their **midpoint** is

$$M(x) = \frac{p_r(x) + p_g(x)}{2}.$$

$$\mathrm{KL}\big(p_r \parallel M\big) = \int p_r(x)\,\log\Big[\tfrac{p_r(x)}{M(x)}\Big]\,dx = \int p_r(x)\,\log\Big[\tfrac{p_r(x)}{\frac{p_r(x)+p_g(x)}{2}}\Big]\,dx.$$

$$= \int p_r(x)\,\log\Big[\frac{2\,p_r(x)}{p_r(x)+p_g(x)}\Big]\,dx = \int p_r(x)\,\log\Big[\tfrac{p_r(x)}{p_r(x)+p_g(x)}\Big]\,dx \;+\; \int p_r(x)\,\log(2)\,dx.$$

Since $\int p_r(x)\,dx = 1$,

$$\mathrm{KL}\big(p_r \parallel M\big) = \int p_r(x)\,\log\Big[\tfrac{p_r(x)}{p_r(x)+p_g(x)}\Big]\,dx \;+\; \log(2).$$

$$\mathrm{KL}\big(p_g \parallel M\big) = \int p_g(x)\,\log\Big[\tfrac{p_g(x)}{p_r(x)+p_g(x)}\Big]\,dx \;+\; \log(2).$$

$$\mathrm{KL}(p_r\|M) + \mathrm{KL}(p_g\|M) = \int p_r(x)\,\log\Big[\tfrac{p_r(x)}{p_r(x)+p_g(x)}\Big]\,dx \;+\; \int p_g(x)\,\log\Big[\tfrac{p_g(x)}{p_r(x)+p_g(x)}\Big]\,dx \;+\; 2\,\log(2).$$

Global optima

$$\mathrm{JS}(p_r \| p_g) = \frac{1}{2}\left[\int p_r(x)\,\log\!\left(\frac{p_r(x)}{p_r(x)+p_g(x)}\right)dx \;+\; \int p_g(x)\,\log\!\left(\frac{p_g(x)}{p_r(x)+p_g(x)}\right)dx \;+\; 2\,\log(2)\right].$$

$$\int p_r(x)\,\log\!\left[\frac{p_r(x)}{p_r(x)+p_g(x)}\right] \;+\; \int p_g(x)\,\log\!\left[\frac{p_g(x)}{p_r(x)+p_g(x)}\right] \;=\; 2\left[\mathrm{JS}(p_r \| p_g) - \log(2)\right].$$

Recall:

$$\max_D V(D,G) = \int p_r(x)\,\log\!\left[\frac{p_r(x)}{p_r(x)+p_g(x)}\right]dx \;+\; \int p_g(x)\,\log\!\left[\frac{p_g(x)}{p_r(x)+p_g(x)}\right]dx.$$

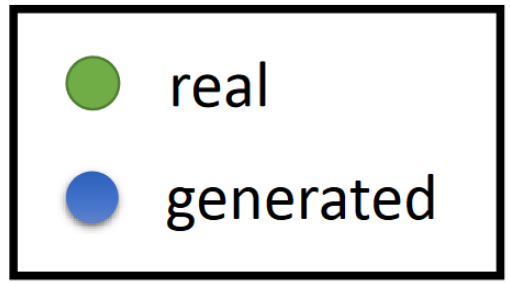
$$\max_D V(D,G) = 2\left[\mathrm{JS}(p_r \| p_g) - \log(2)\right].$$

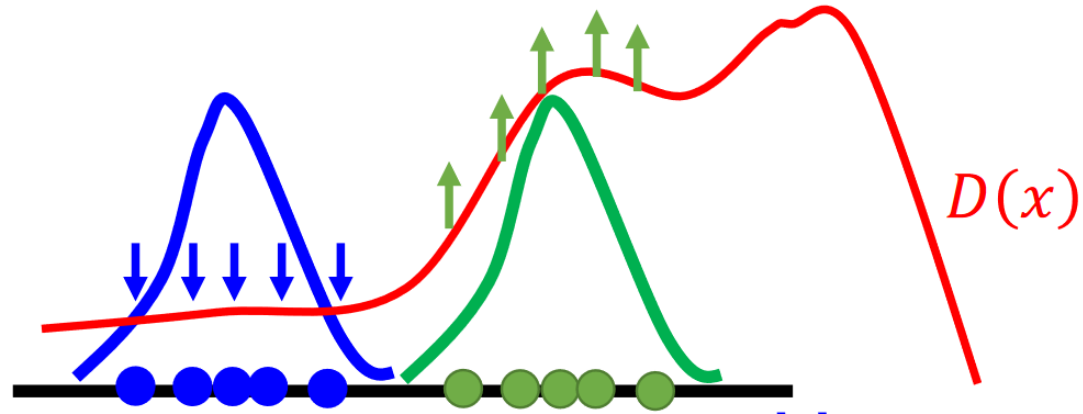
Interpretation

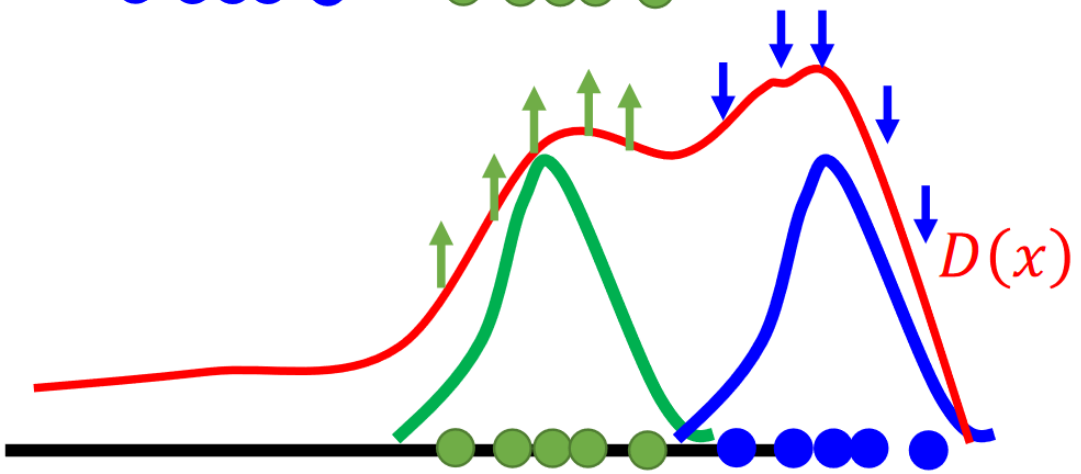
- **When $p_g = p_r$:**
  $\mathrm{JS}(p_r \| p_g) = 0$. Hence $\max_D V(D, G) = -2\log(2)$. Equivalently, $-\log(4)$.
  In this case, the discriminator's best guess is $D^*(x) = \frac{1}{2}$ everywhere (completely random guess).
- **When $p_g \neq p_r$:**
  $\mathrm{JS}(p_r \| p_g) > 0$. The discriminator can achieve a value higher than $-2\log(2)$, up to 0 if the two distributions are perfectly separable.

Hence the **GAN game** ideally pushes p_g to **match** p_r (so $\mathrm{JS} = 0$), achieving the global optimum.

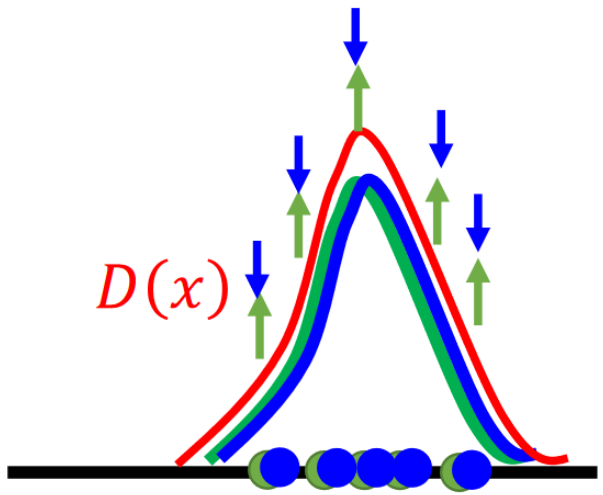
Discriminator - Training

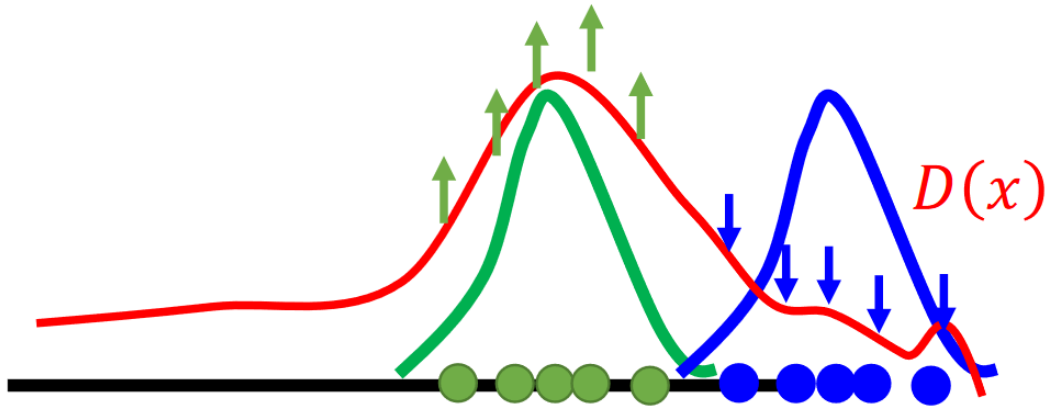




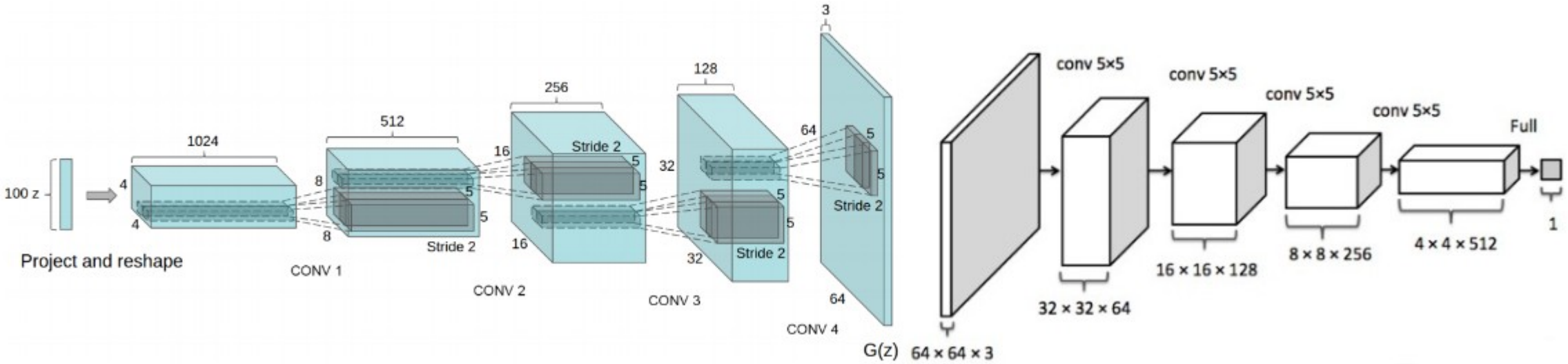


In the end ……





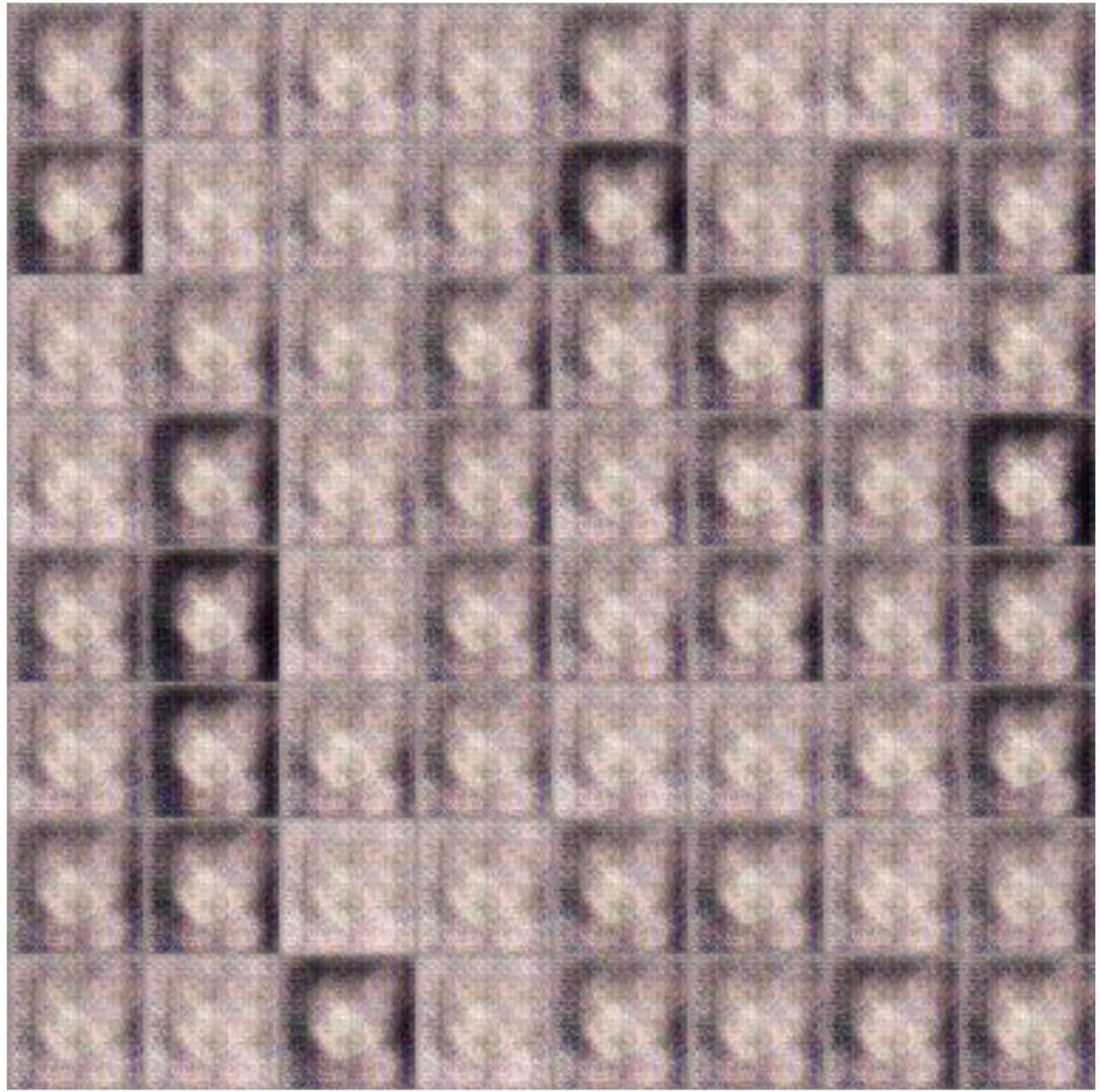
DC GAN


100 z
Project and reshape
1024
512
256
128
3
4
8
16
32
64
5
Stride 2
CONV 1
CONV 2
CONV 3
CONV 4
G(z)
64 × 64 × 3
conv 5×5
conv 5×5
conv 5×5
conv 5×5
Full
32 × 32 × 64
16 × 16 × 128
8 × 8 × 256
4 × 4 × 512
1

Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." arXiv:1511.06434 (2015).

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

100 updates



1000 updates

5000 updates

50,000 updates

Some tips for GAN network design

- Replace pooling by strided convolution and unpolling by fractional strided convolution
- Use batch-norm
- Avoid FC layers in deeper architectures
- Use ReLU in all the generator layers except output where

use tanh

- Use leaky-ReLU in the discriminator layers

** See GAN-hacks (by soumith chintala) for more details

- **Why GANs?**
  - Sampling (or generation) is straightforward.
  - Training doesn't involve Maximum Likelihood estimation.
  - Robust to Overfitting since Generator never sees the training data.
  - Empirically, GANs are good at capturing the modes of the distribution.

Non convergence of GAN training

- **Deep Learning models (in general) involve a single player**
  - The player tries to maximize its reward (minimize its loss).
  - Use SGD (with Backpropagation) to find the optimal parameters.
  - SGD has convergence guarantees (under certain conditions).
  - **Problem:** With non-convexity, we might converge to local optima.

$$\min_G L(G)$$

- **GANs instead involve two (or more) players**
  - Discriminator is trying to maximize its reward.
  - Generator is trying to minimize Discriminator's reward.

$$\min_G \max_D V(D, G)$$

  - SGD was not designed to find the Nash equilibrium of a game.
  - **Problem:** We might not converge to the Nash equilibrium at all.

Lets consider an example

$$\min_x \max_y \; xy$$

Gradient descent would update:

- $x \leftarrow x - \eta y$
- $y \leftarrow y + \eta x$

Even though the **true saddle point is at (0, 0)**, these updates keep rotating around the origin forever — they **never converge.**

$$\mathbf{z}_t = \begin{bmatrix} x_t \\ y_t \end{bmatrix}, \quad \mathbf{z}_{t+1} = \underbrace{\begin{bmatrix} 1 & -\eta \\ \eta & 1 \end{bmatrix}}_{A} \mathbf{z}_t$$

The update matrix A has **complex eigenvalues**:

$$\lambda = 1 \pm i\eta$$

- If $|\lambda| > 1$, the updates will **spiral outward**, causing **divergence.**
- If $|\lambda| \approx 1$, the updates **oscillate in circles.**

- $x_0 = 1$
- $y_0 = 0$

Step 1:

$$x_1 = 1 - \eta \cdot 0 = 1$$

$$y_1 = 0 + \eta \cdot 1 = \eta$$

Step 2:

$$x_2 = 1 - \eta \cdot \eta \approx 1 - \eta^2$$

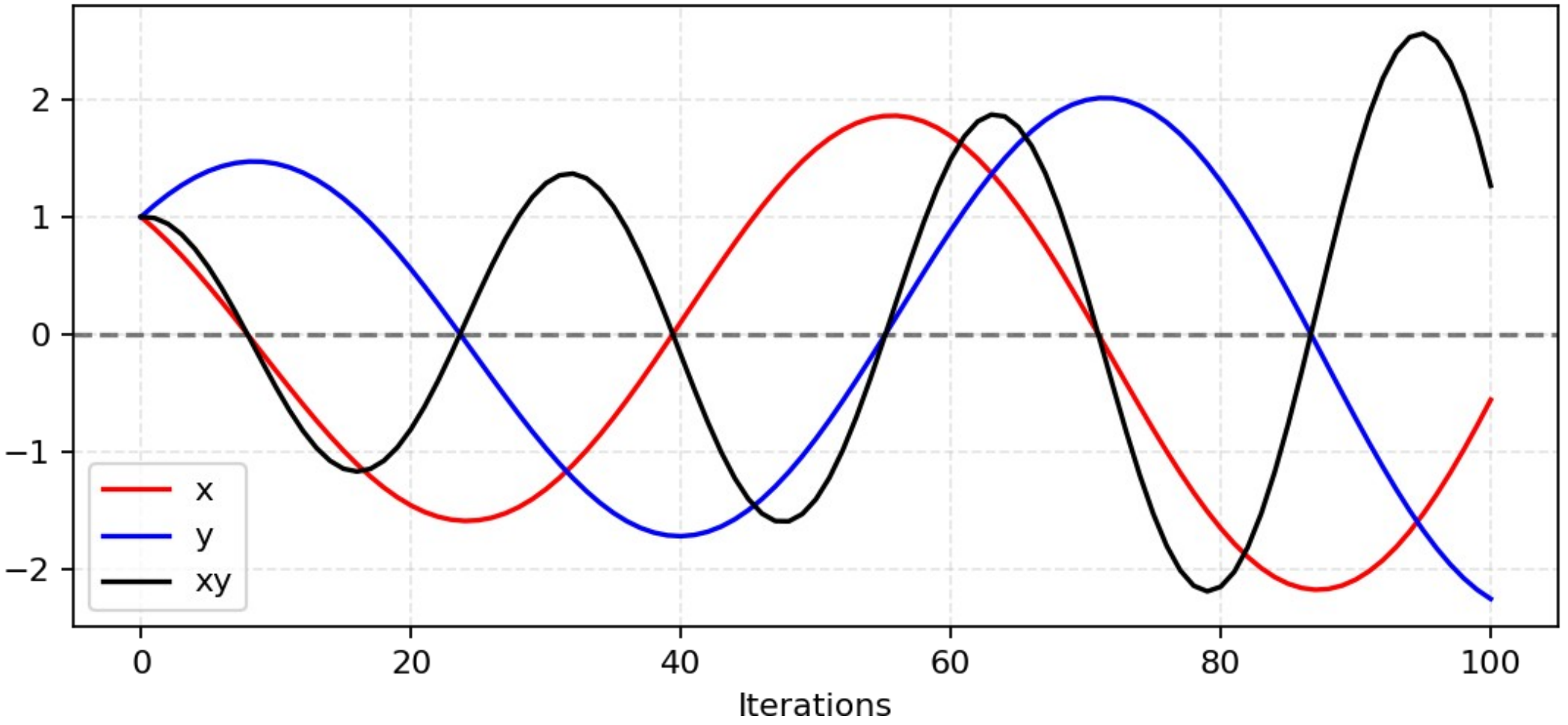
$$y_2 = \eta + \eta \cdot 1 = \eta(1 + \eta)$$

Step 3:

$$x_3 = 1 - \eta^2 - \eta(1 + \eta) \approx 1 - \eta^2 - \eta - \eta^2$$

$$y_3 = \eta(1 + \eta) + \eta(1 - \eta^2) \approx 2\eta + \eta^3$$

Problem in obtaining the equilibrium



What are the solutions

- Reduce learning rate
- Use different learning rates for the generator and discriminator
- Predict the next step’s gradient and use it to stabilize updates.

1. **Predictive Step (Lookahead):**
   - First, take a partial gradient step to **predict where the opponent will move.**
   - Update to a temporary point $\tilde{x}, \tilde{y}$:

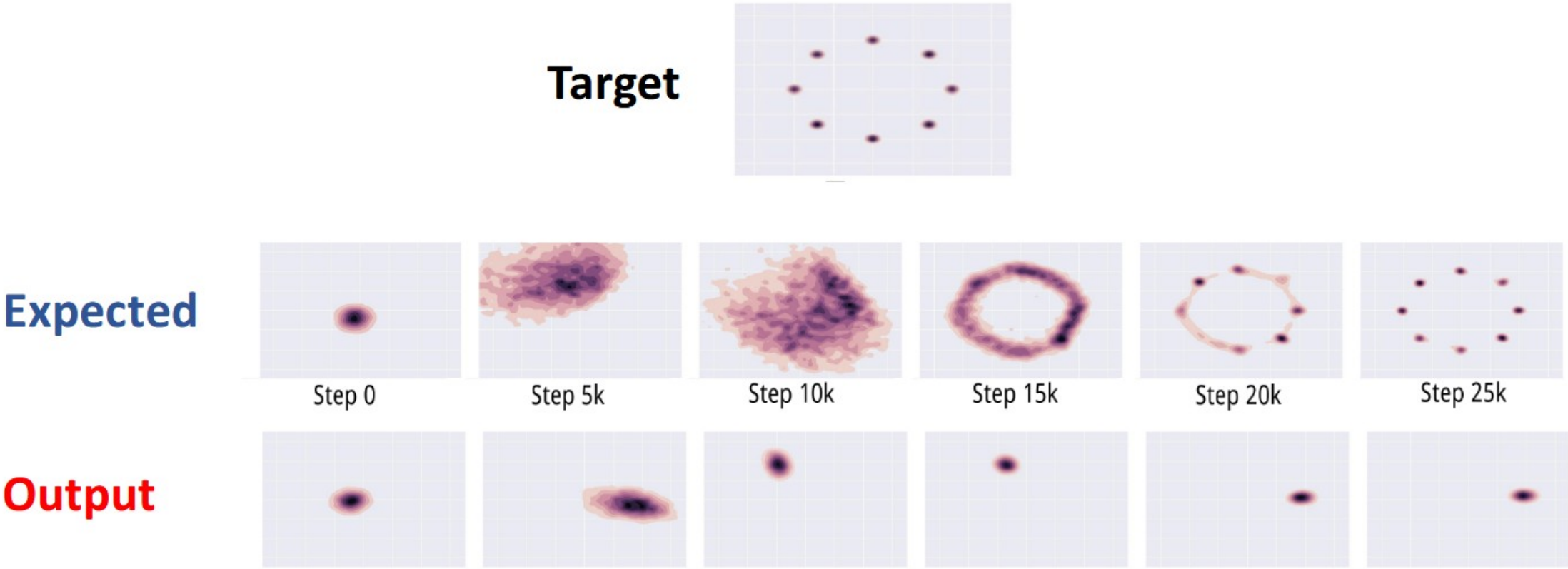
$$\tilde{x} = x_t - \eta \frac{\partial f}{\partial x}(x_t, y_t)$$
$$\tilde{y} = y_t + \eta \frac{\partial f}{\partial y}(x_t, y_t)$$

2. **Corrective Step (Using Lookahead Gradient):**
   - Now, use the gradients at this new predicted point $(\tilde{x}, \tilde{y})$ to **correct the original update:**

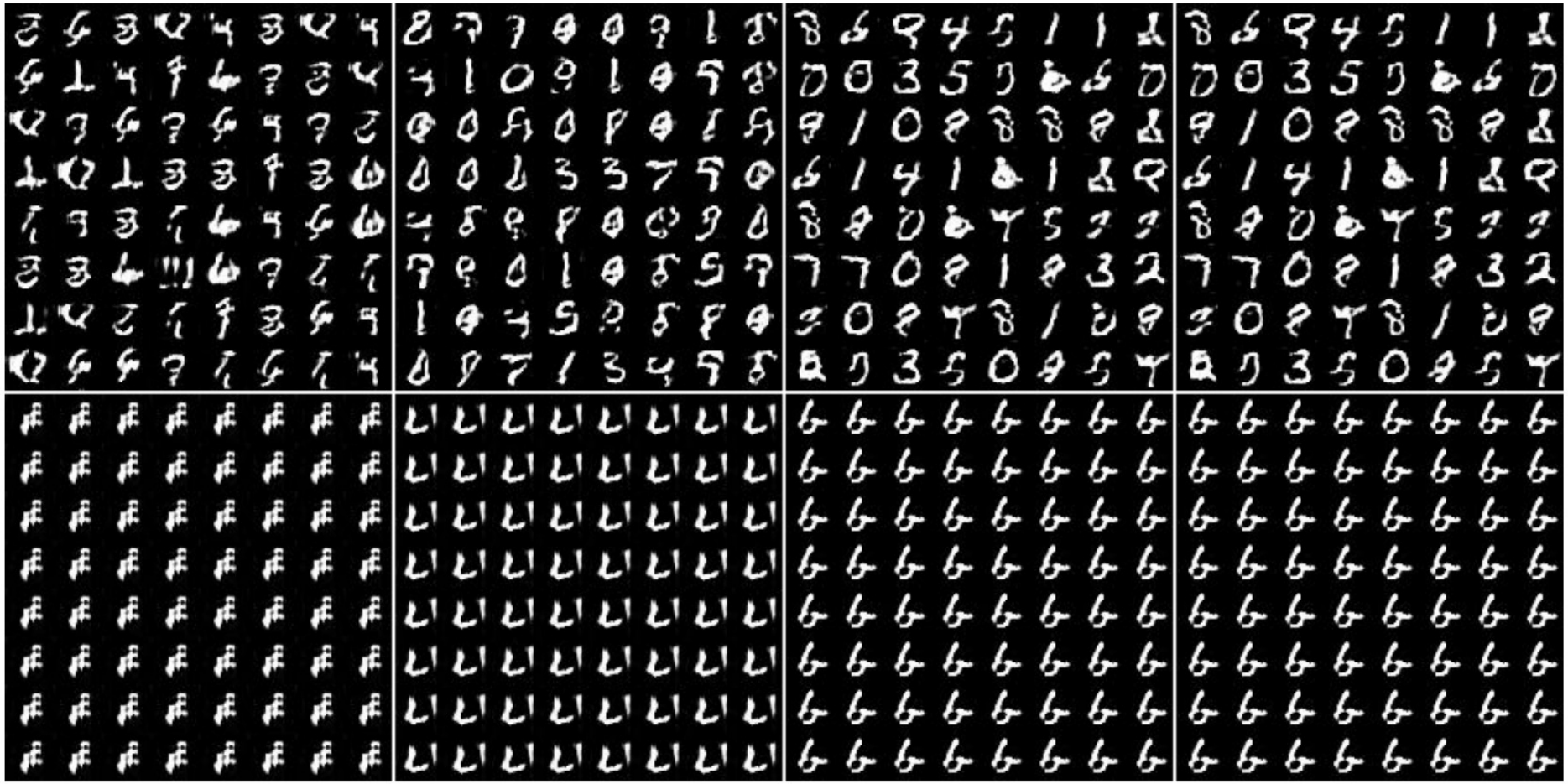
$$x_{t+1} = x_t - \eta \frac{\partial f}{\partial x}(\tilde{x}, \tilde{y})$$
$$y_{t+1} = y_t + \eta \frac{\partial f}{\partial y}(\tilde{x}, \tilde{y})$$

Mode collapse

- **Generator fails to output diverse samples**



Mode collapse in MNIST

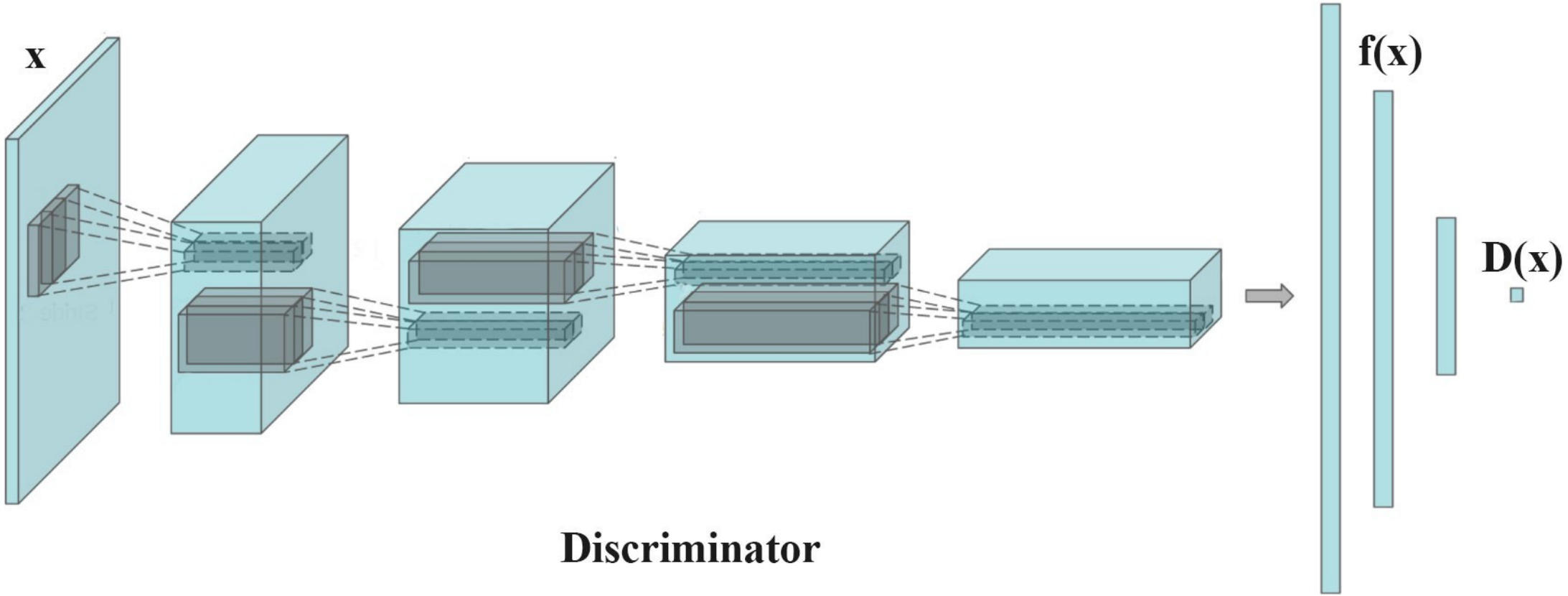


Feature matching

In **Feature Matching:**

- Instead of directly minimizing the divergence between real and generated distributions in pixel space, we:
  - **Extract feature representations** of real and generated samples using an intermediate layer of the discriminator.
  - **Match the statistics** (e.g., mean, covariance) of these feature representations to ensure that generated samples cover the same modes as the real samples.

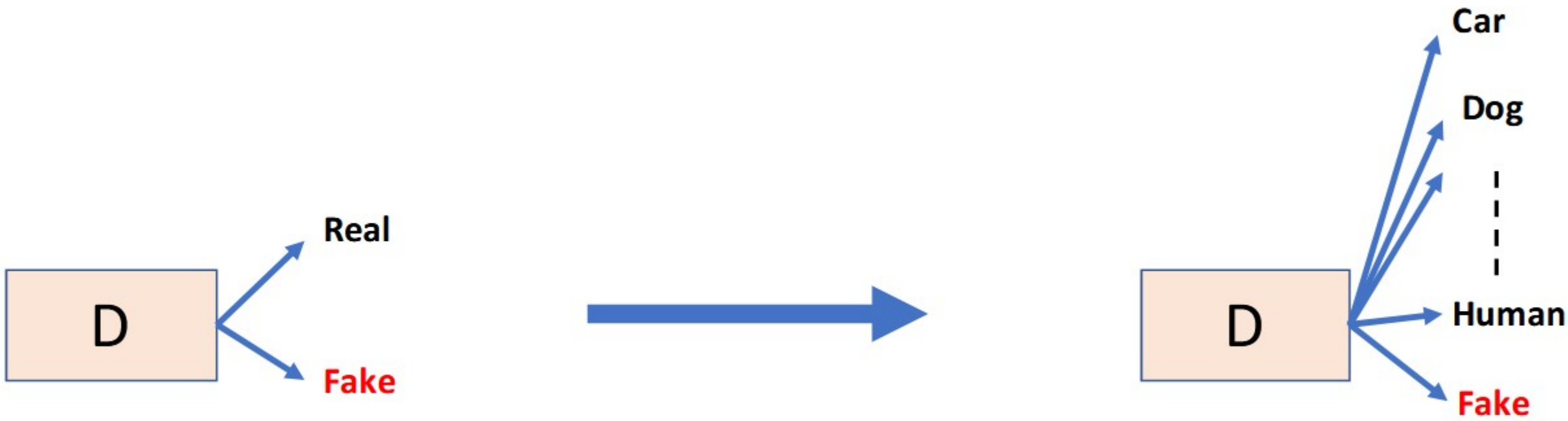
Feature matching

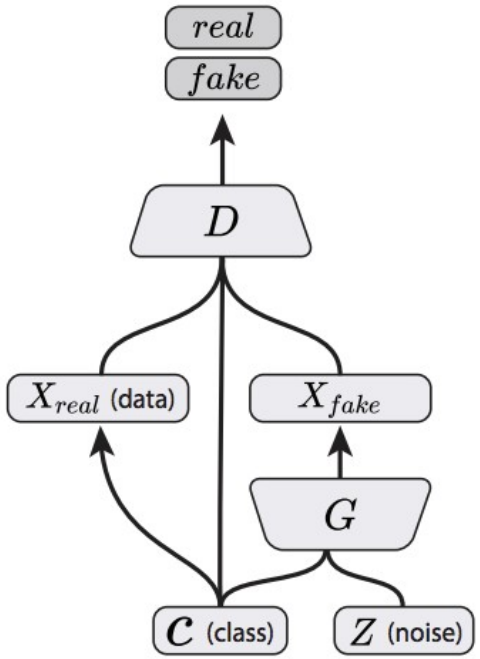


$$||\mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} \mathbf{f}(\boldsymbol{x}) - \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})} \mathbf{f}(G(\boldsymbol{z}))||_2^2$$

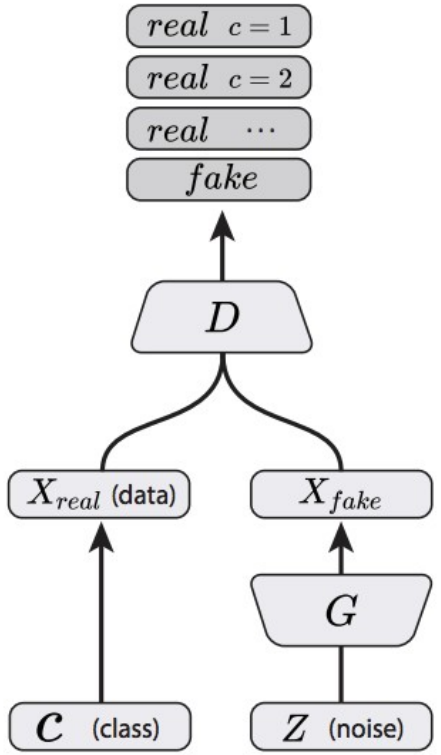
- **Extract features that capture diversity in the mini-batch**
  - For e.g. L2 norm of the difference between all pairs from the batch
- **Feed those features to the discriminator along with the image**
- **Feature values will differ b/w diverse and non-diverse batches**
  - Thus, Discriminator will rely on those features for classification
- **This in turn,**
  - Will force the Generator to match those feature values with the real data
  - Will generate diverse batches

Supervision with labels

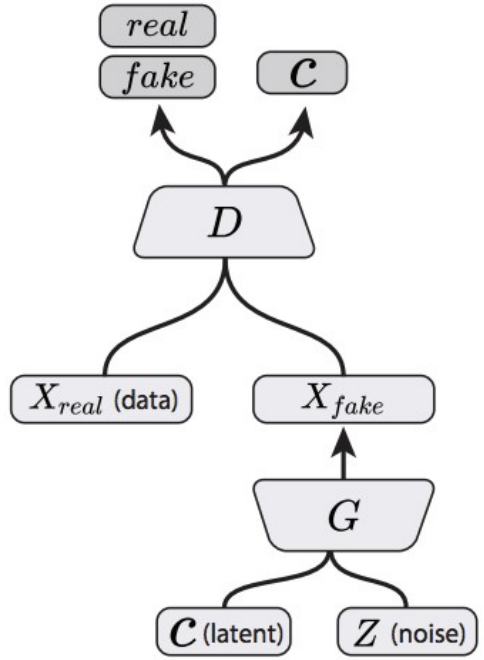




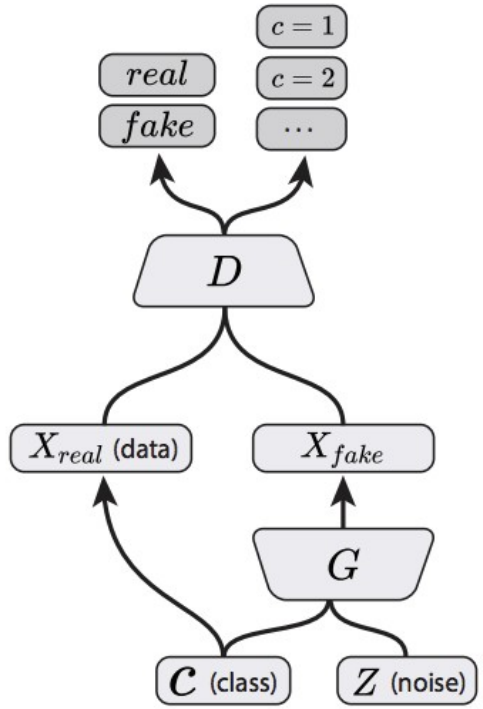
Conditional GAN
(Mirza & Osindero, 2014)



Semi-Supervised GAN
(Odena, 2016; Salimans, et al., 2016)



InfoGAN
(Chen, et al., 2016)



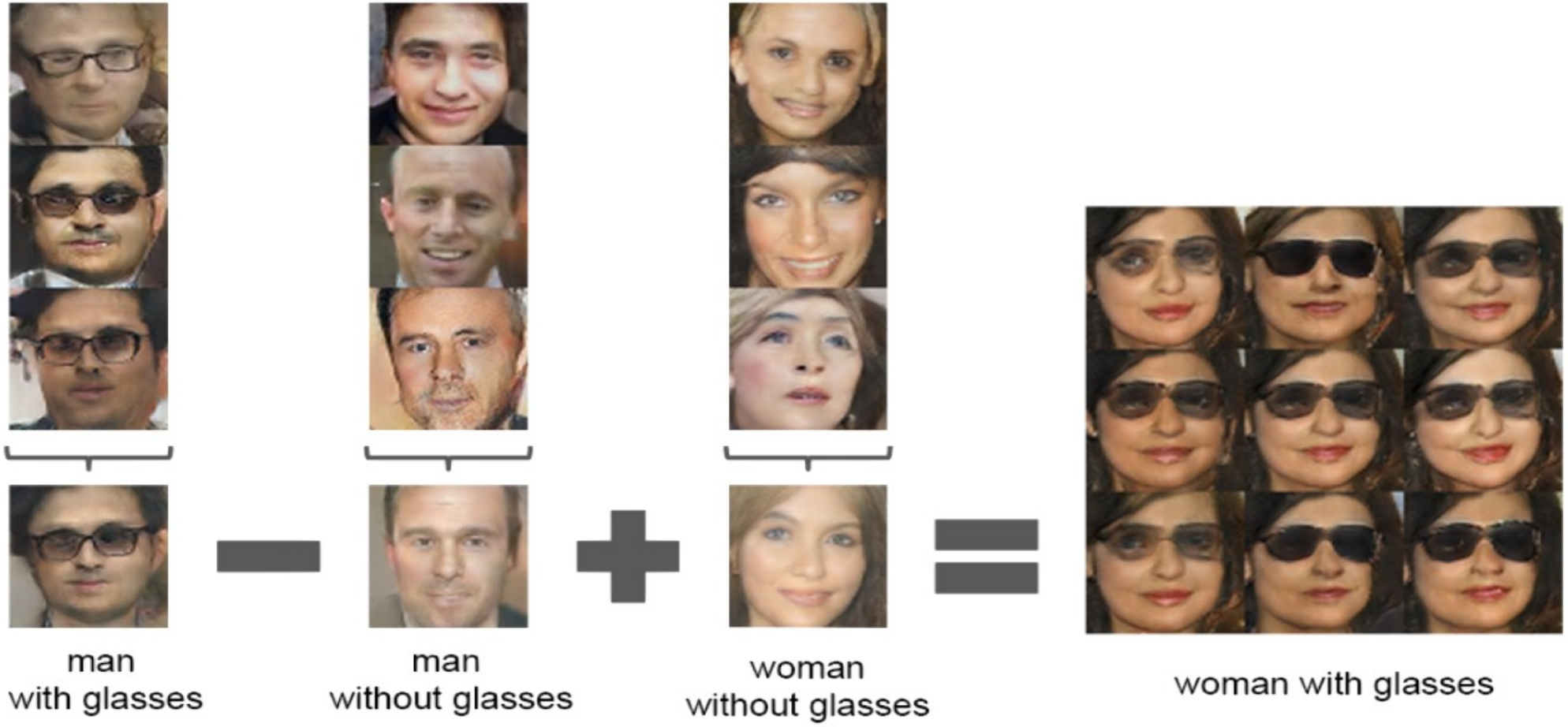
AC-GAN
(Present Work)

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

One issue of vanilla GAN

- The generator maps the noise vector $z \sim p_z$ to the data space.
- However, z lacks **structured information** about the factors of variation in the data.
- As a result:
  - Small changes in z can affect **multiple factors** at the same time.
  - There's **no control** over interpretable features such as rotation, thickness, shape, etc.
  - Latent dimensions are **entangled** — they don't correspond to independent factors of variation.

Latent vectors capture meaningful information



Entangled

Disentangled

Goal of InfoGAN

•Learn a structured and **disentangled latent space** where each latent variable controls a distinct, interpretable factor.

•Encourage the generator to **encode meaningful latent codes** that correspond to different aspects of the generated data.

- **Noise vector** z: Unstructured noise, as in standard GANs.
- **Latent code** c: Structured, interpretable variables that control specific factors.

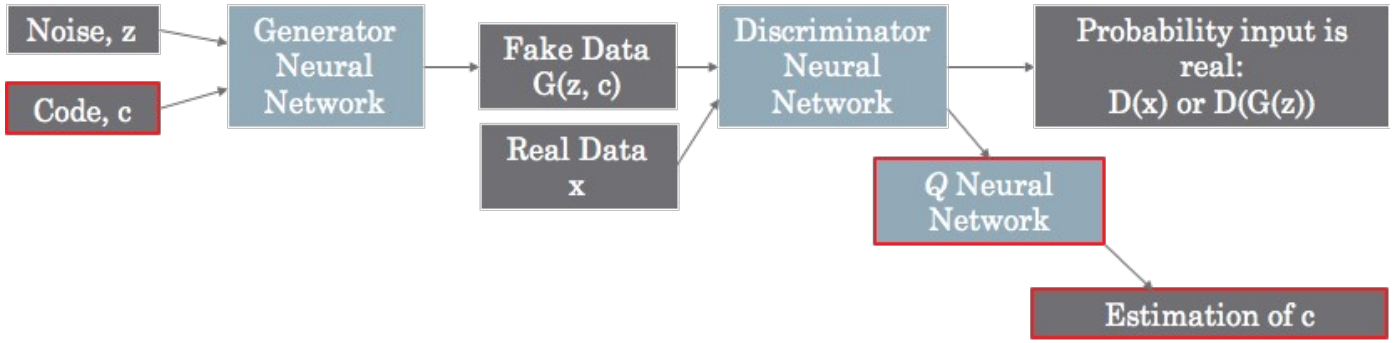
$$G(z, c) \rightarrow x$$

- The generator $G(z, c)$ takes both z and c as input.
- c is explicitly encouraged to **control meaningful factors** by maximizing **mutual information** between c and the generated sample x.

InfoGAN

$$\min_G \max_D V_I(D,G) = V(D,G) - \lambda I(c; G(z,c))$$

1. **Generator** $G(z, c)$:
   - Takes **random noise** $z \sim p_z$ and **structured code** $c \sim p(c)$.
   - Generates a sample $x = G(z, c)$.
   - The goal is to embed meaningful information about c into the generated sample.
2. **Discriminator** $D(x)$:
   - Tries to distinguish between real and fake samples.
   - Trained using the **standard GAN loss.**
3. **Auxiliary Network** $Q(c \mid x)$:
   - Learns to **recover** the latent code c from the generated sample.
   - Ensures that the generator encodes interpretable information into the samples.



Idea of mutual information

Mutual information measures how much **knowing one variable** reduces uncertainty about another.

$$\mathbb{I}(c; x) = H(c) - H(c \mid x)$$

Where:

- $H(c)$ is the entropy of the latent code.
- $H(c \mid x)$ is the conditional entropy of c given the generated sample.

- High mutual information means that knowing the generated sample x tells us a lot about c, ensuring that c is **recoverable** from x.
- To maximize $\mathbb{I}(c; x)$, the generator must **embed the information in c into the generated samples.**

Calculating MI directly is difficult

Since $\mathbb{I}(c; x)$ is hard to compute directly, InfoGAN introduces a **variational lower bound**:

$$\mathbb{I}(c; x) \geq \mathbb{E}_{x \sim G(z,c)}[\log Q(c \mid x)] + H(c)$$

Where:

- $Q(c \mid x)$ is an **auxiliary network** that predicts c from the generated sample.
- **Training Objective:** Maximize $\log Q(c \mid x)$ to ensure that c is **recoverable**.

(a) Azimuth (pose)

(b) Presence or absence of glasses

(c) Hair style

(d) Emotion

An alternate view of GAN

$$\min_G \max_D V(D,G)$$
$$V(D,G) = \mathbb{E}_{x\sim p(x)}[\log D(x)] + \mathbb{E}_{z\sim q(z)}\left[\log\left(1 - D(G(z))\right)\right]$$

$$D^* = \arg\max_D V(D,G)$$

$$G^* = \arg\min_G V(D,G)$$

- In this formulation, Discriminator's strategy was $D(x) \to 1, \quad D\left(G(z)\right) \to 0$

- Alternatively, we can flip the binary classification labels i.e. **Fake = 1**, **Real = 0**

$$V(D,G) = \mathbb{E}_{x\sim p(x)}\left[\log\left(1 - D(x)\right)\right] + \mathbb{E}_{z\sim q(z)}\left[\log\left(D(G(z))\right)\right]$$

- In this new formulation, Discriminator's strategy will be $D(x) \to 0, \quad D\left(G(z)\right) \to 1$

To reframe this as a **margin loss**, we introduce a decision function:

$$f(x) = 2D(x) - 1$$

- $f(x)$ maps $D(x)$ from $[0, 1]$ to $[-1, 1]$.

In this view:

- For real samples, the goal is to **push them below 0**.
- For fake samples, the goal is to **push them above 0**.

$$\mathcal{L}_{\text{margin}} = \mathbb{E}_{x \sim p_{\text{data}}}[\max(0, 1 + f(x))] + \mathbb{E}_{z \sim p_z}[\max(0, 1 - f(G(z)))]$$

Least-Square GAN

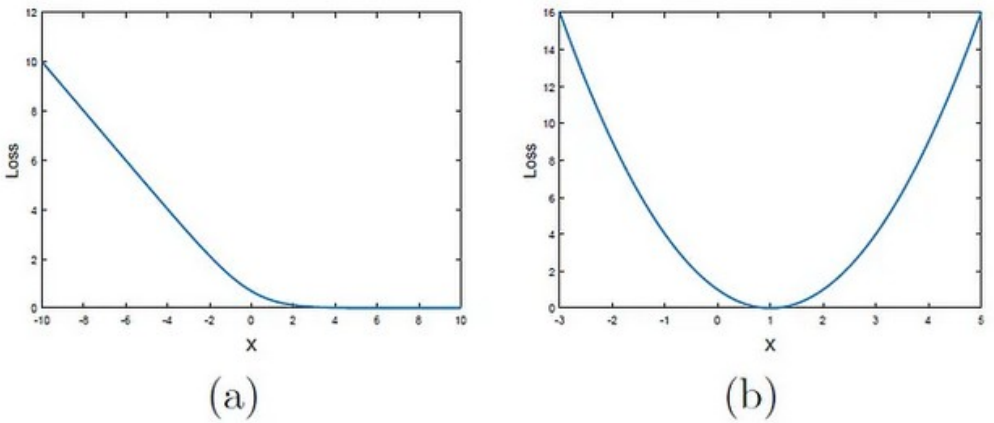
LSGAN minimizes the least squares error between the discriminator's output and the target values.

- **For real samples:**

$$\mathcal{L}_D^{\text{real}} = \frac{1}{2}\mathbb{E}_{x\sim p_{\text{data}}}\left[(D(x)-1)^2\right]$$

- **For fake samples:**

$$\mathcal{L}_D^{\text{fake}} = \frac{1}{2}\mathbb{E}_{z\sim p_z}\left[(D(G(z)))^2\right]$$

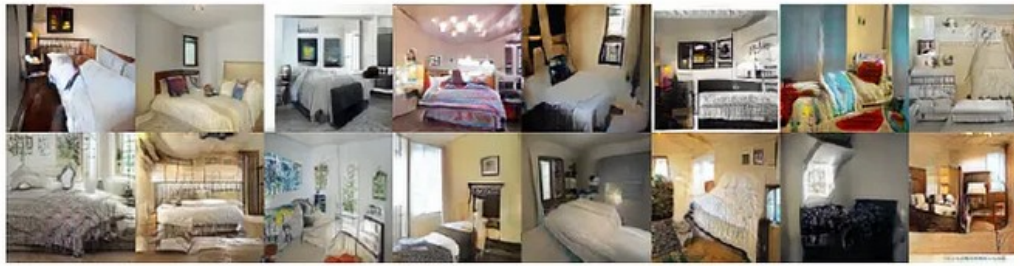


(a): The sigmoid cross entropy loss function. (b): The least squares loss function.

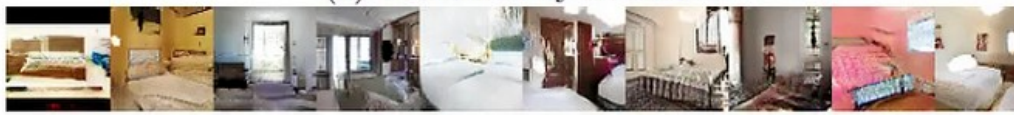
- The generator minimizes the least squares error to **make fake samples look like real samples:**

$$\mathcal{L}_G = \frac{1}{2}\mathbb{E}_{z\sim p_z}\left[(D(G(z))-1)^2\right]$$

Qualitative results



(a) Generated by LSGANs.



(b) Generated by DCGANs (Reported in [11]).

(c) Generated by EBGANs (Reported in [26]).

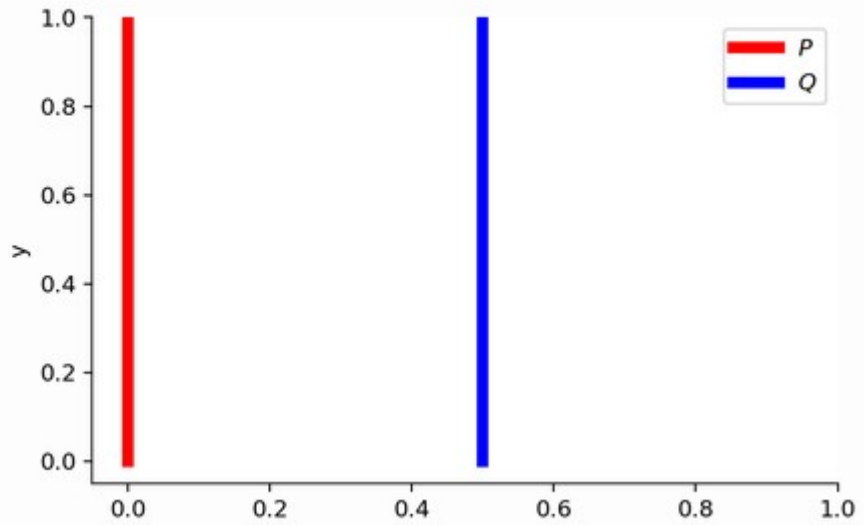
(a) Church outdoor.

(b) Dining room.

(c) Kitchen.

(d) Conference room.

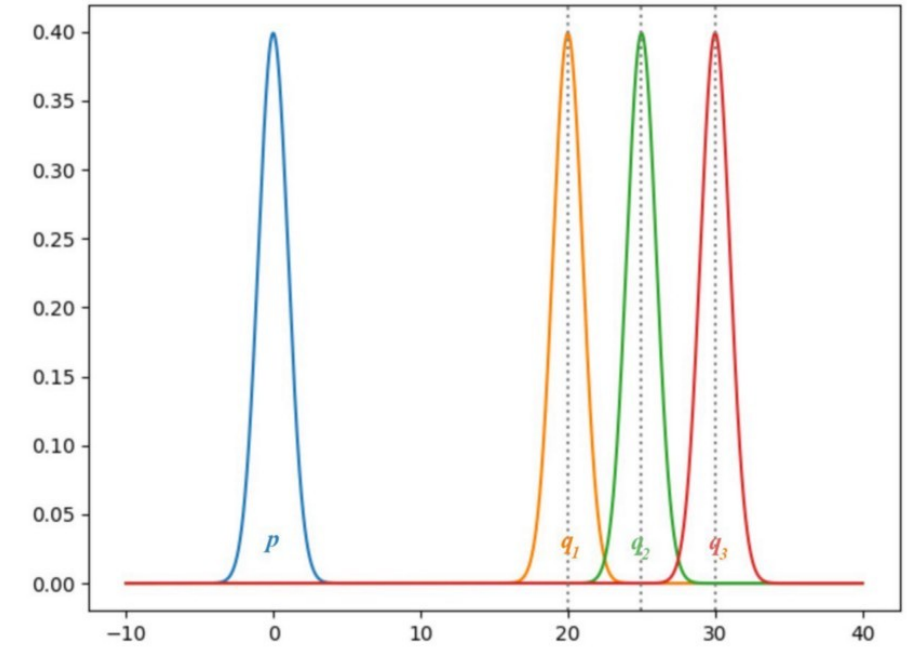
Calculate KL, JS

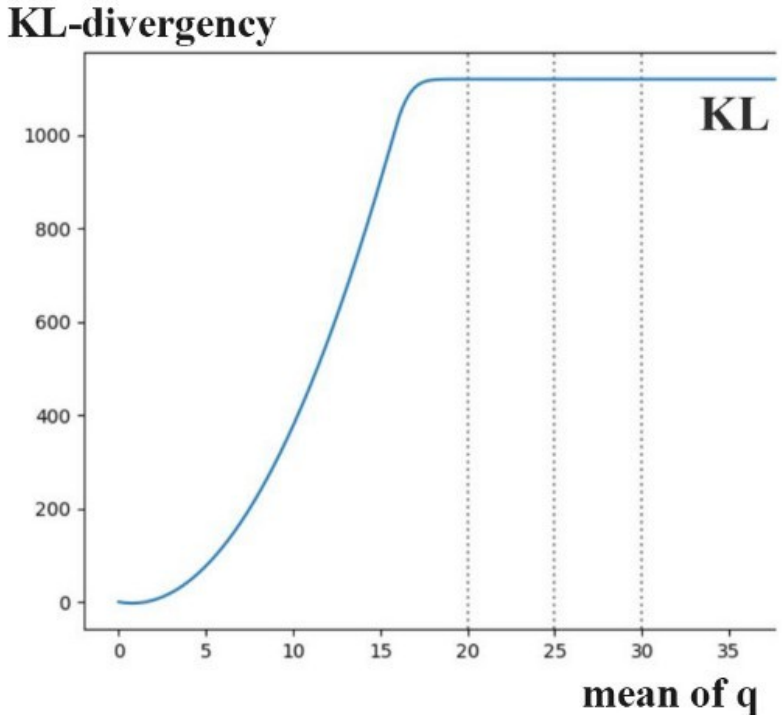


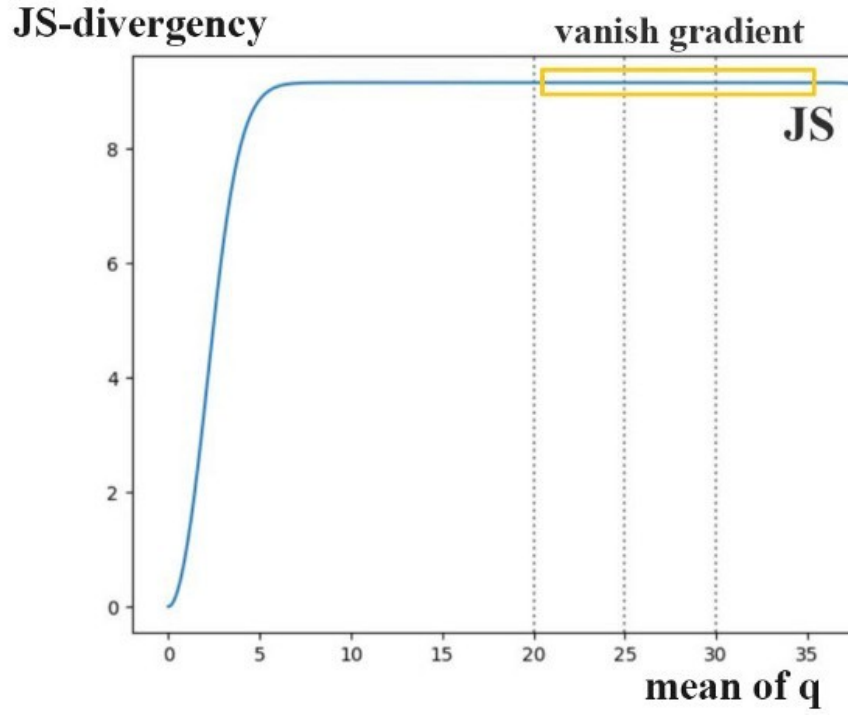
$$D_{KL}(P\|Q) = \sum_{x=0,y\sim U(0,1)} 1 \cdot \log \frac{1}{0} = +\infty$$

$$D_{KL}(Q\|P) = \sum_{x=\theta,y\sim U(0,1)} 1 \cdot \log \frac{1}{0} = +\infty$$

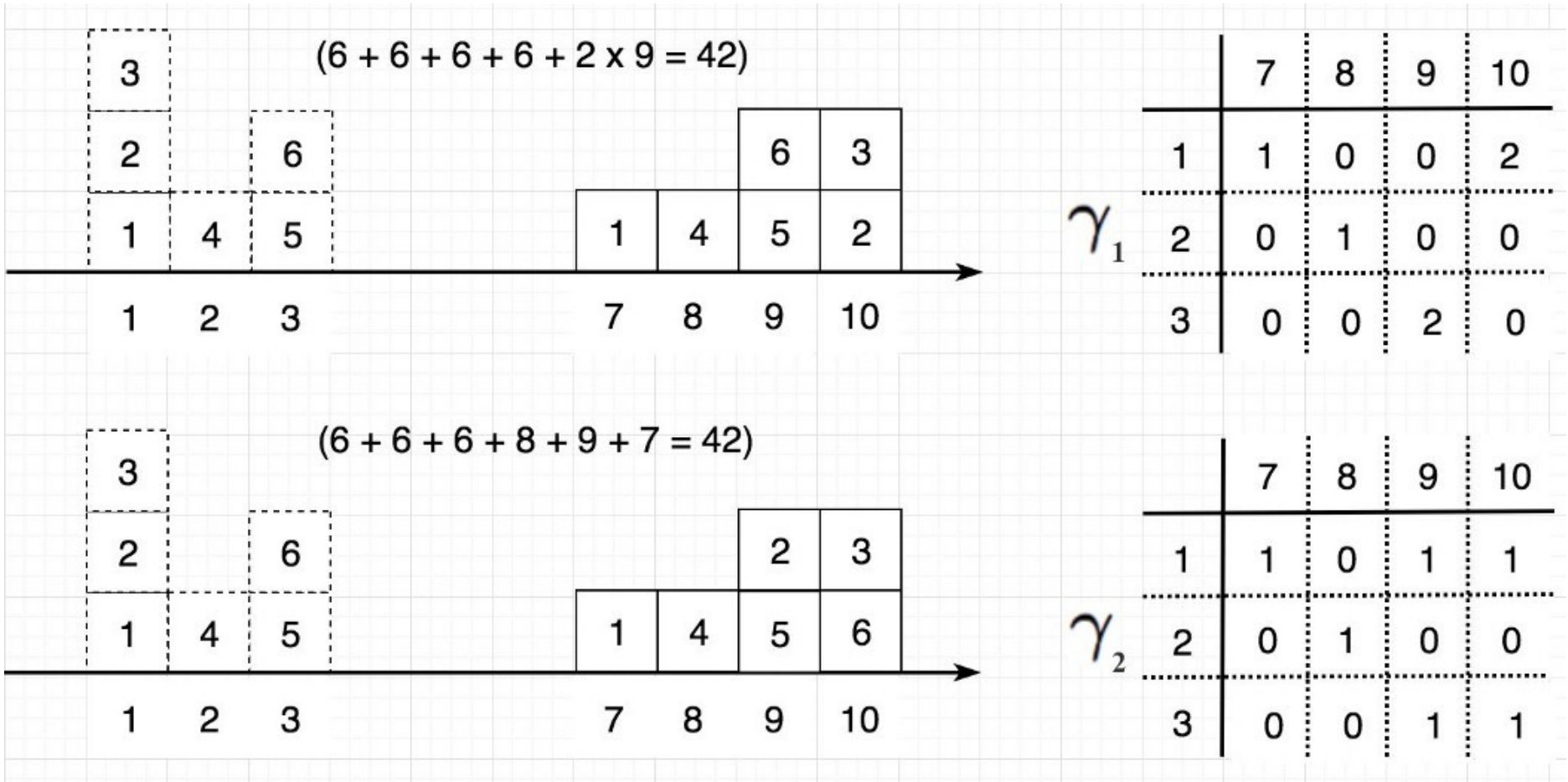
$$D_{JS}(P,Q) = \frac{1}{2}\left(\sum_{x=0,y\sim U(0,1)} 1 \cdot \log \frac{1}{1/2} + \sum_{x=0,y\sim U(0,1)} 1 \cdot \log \frac{1}{1/2}\right) = \log 2$$


0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−10
0
10
20
30
40
p
q_1
q_2
q_3


KL-divergency
KL
1000
800
600
400
200
0
0
5
10
15
20
25
30
35
mean of q


JS-divergency
vanish gradient
JS
8
6
4
2
0
0
5
10
15
20
25
30
35
mean of q

EM distance

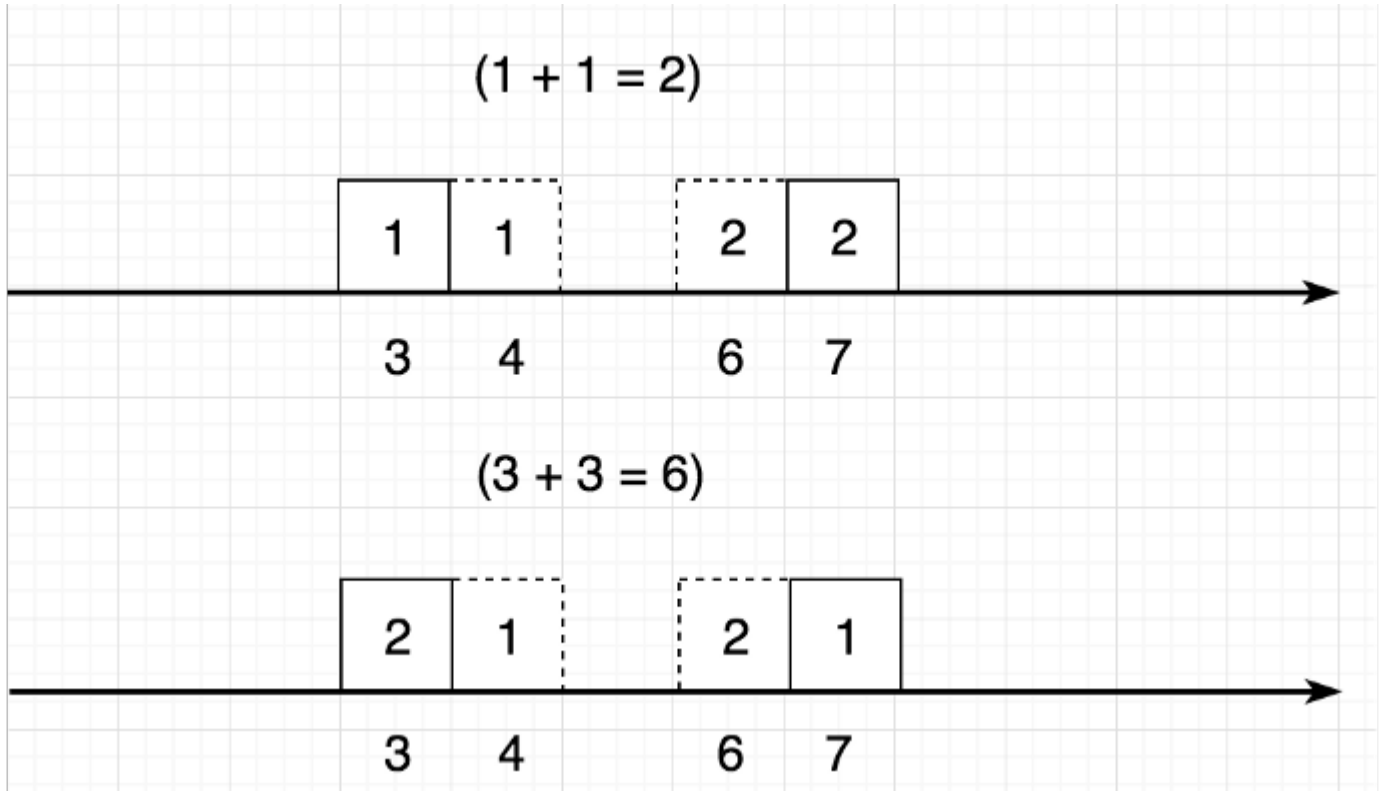


γ_1

	7	8	9	10
1	1	0	0	2
2	0	1	0	0
3	0	0	2	0

γ_2

	7	8	9	10
1	1	0	1	1
2	0	1	0	0
3	0	0	1	1


(1 + 1 = 2)
1
1
2
2
3
4
6
7
(3 + 3 = 6)
2
1
2
1
3
4
6
7

Wasserstein distance

Given two probability distributions P_r and P_g over a metric space Ω, the EMD or Wasserstein-1 distance is defined as:

$$W(P_r, P_g) = \inf_{\gamma \in \Pi(P_r, P_g)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|]$$

Where:

- $\Pi(P_r, P_g)$ is the set of all **joint distributions** (also called transport plans) $\gamma(x, y)$, such that:

  $$\sum_y \gamma(x, y) = P_r(x) \quad \text{and} \quad \sum_x \gamma(x, y) = P_g(y)$$

  This ensures that γ transports mass from P_r to P_g.
- $\|x - y\|$ is the distance between points x and y in the space.
- $\mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|]$ is the expected cost (amount of work) needed to transport the mass.

In discrete space

For discrete distributions:

- $P_r = \{(x_i, p_i)\}$ where x_i is the location and p_i is the probability at that location.
- $P_g = \{(y_j, q_j)\}$ similarly.

The discrete form of EMD becomes:

$$W(P_r, P_g) = \min_{\gamma \in \Pi(P_r, P_g)} \sum_{i,j} \gamma(x_i, y_j) d(x_i, y_j)$$

Where:

- $\gamma(x_i, y_j)$ is the amount of mass transported from x_i to y_j.
- $d(x_i, y_j)$ is the distance (e.g., Euclidean distance) between the points.

Kantorvich-Rubinson dualty

Instead of:

> Finding a transport plan,

we say:

> "Let's find a **function** $f(x)$ that scores how 'real' each point feels, such that it changes smoothly (1-Lipschitz)."

Here's what **Kantorovich-Rubinstein Duality** tells us:

$$W(P_r, P_g) = \sup_{f \in \text{Lip}_1} \left(\mathbb{E}_{x \sim P_r}[f(x)] - \mathbb{E}_{x \sim P_g}[f(x)] \right)$$

- Gives **high values to real samples**.
- Gives **low values to fake samples.**
- Doesn't change too fast — it's **1-Lipschitz**, i.e.:

$$|f(x) - f(y)| \leq \|x - y\|$$

- The bigger the gap between the real and fake averages, the more different the distributions are — that's your **Wasserstein distance**!

The WGAN loss

Critic (Discriminator) Loss:

$$\mathcal{L}_D = \mathbb{E}_{x \sim P_g}[f(x)] - \mathbb{E}_{x \sim P_r}[f(x)]$$

We **maximize** this (i.e., perform gradient **ascent**) to make the critic assign:

- **Higher scores** to real samples ($x \sim P_r$)
- **Lower scores** to fake samples ($x \sim P_g$)

> In code, you'd typically **minimize the negative** of this expression.

Generator Loss:

$$\mathcal{L}_G = -\mathbb{E}_{x \sim P_g}[f(x)]$$

The generator tries to produce samples that the critic scores **high**, so it pushes this expectation **up**, i.e., it tries to **minimize** the negative score.

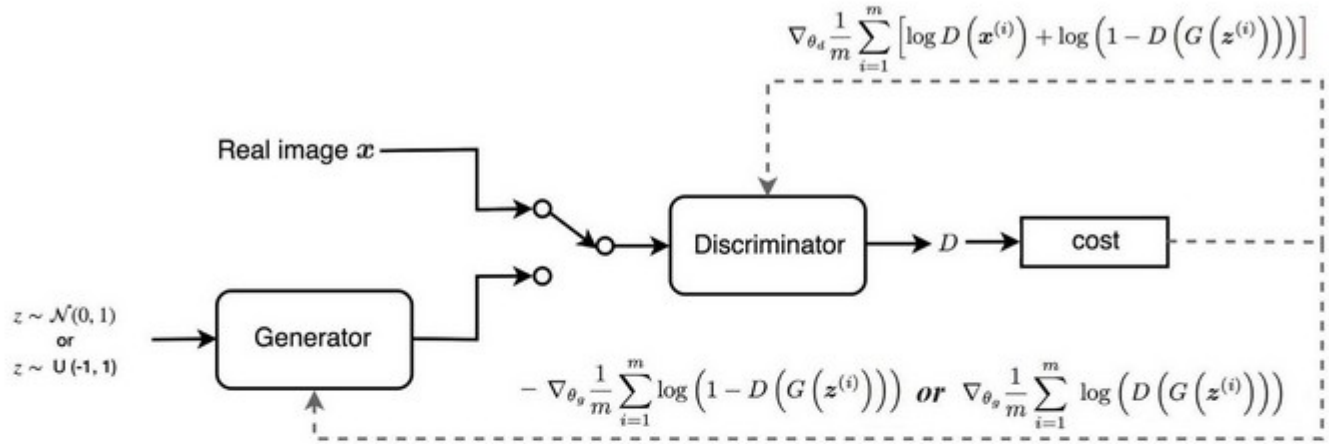
Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.

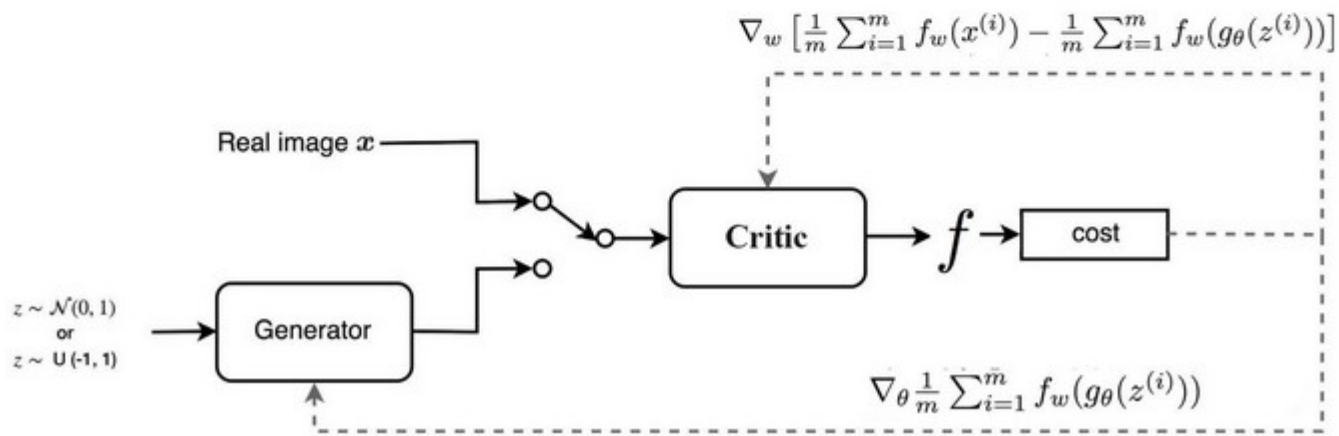
Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

while θ has not converged **do**
  for $t = 0, ..., n_{\text{critic}}$ **do**
    Sample $\{x^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_r$ a batch from the real data.
    Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
    $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$
    $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
    $w \leftarrow \text{clip}(w, -c, c)$
  end for
  Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$
  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
end while

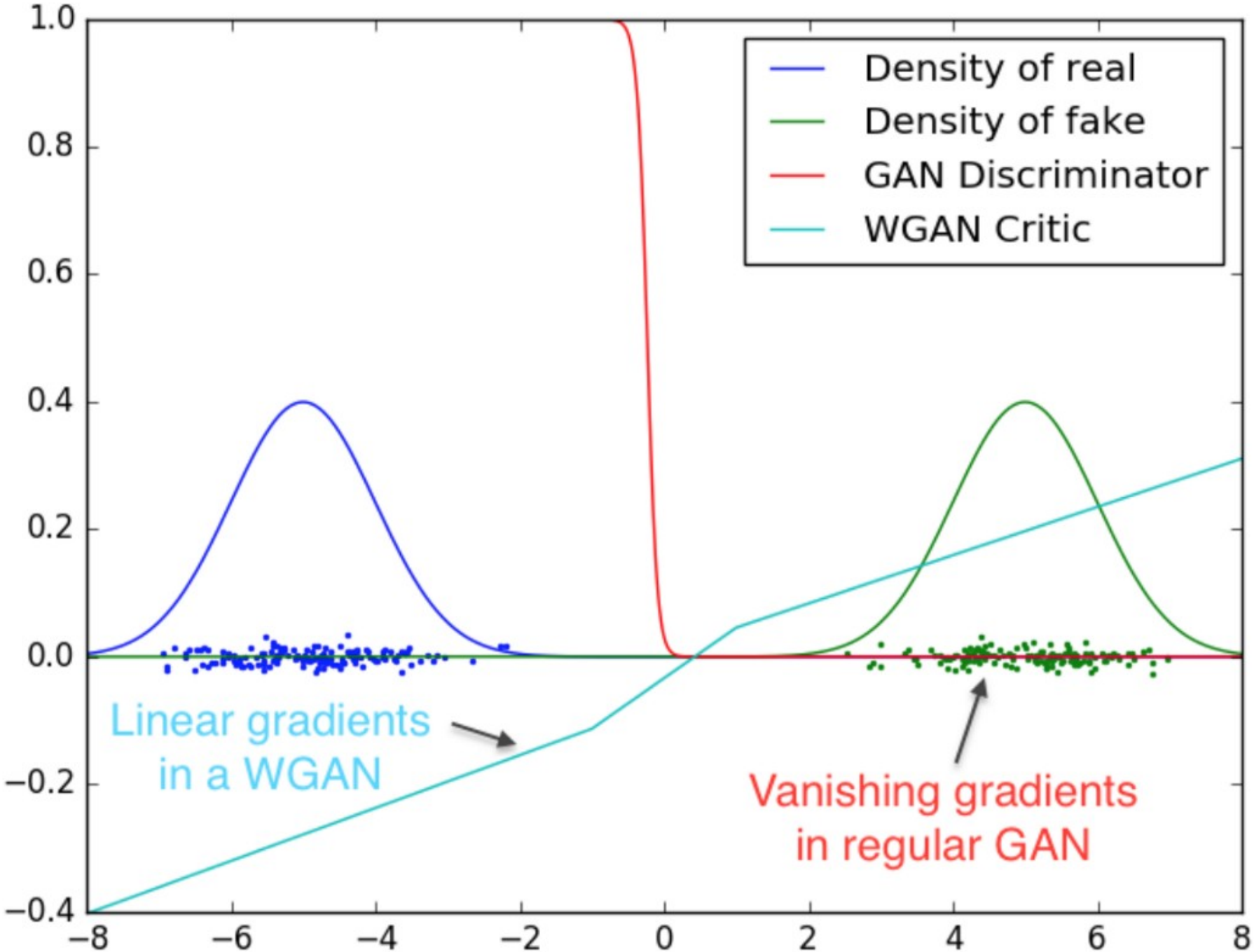
GAN:


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

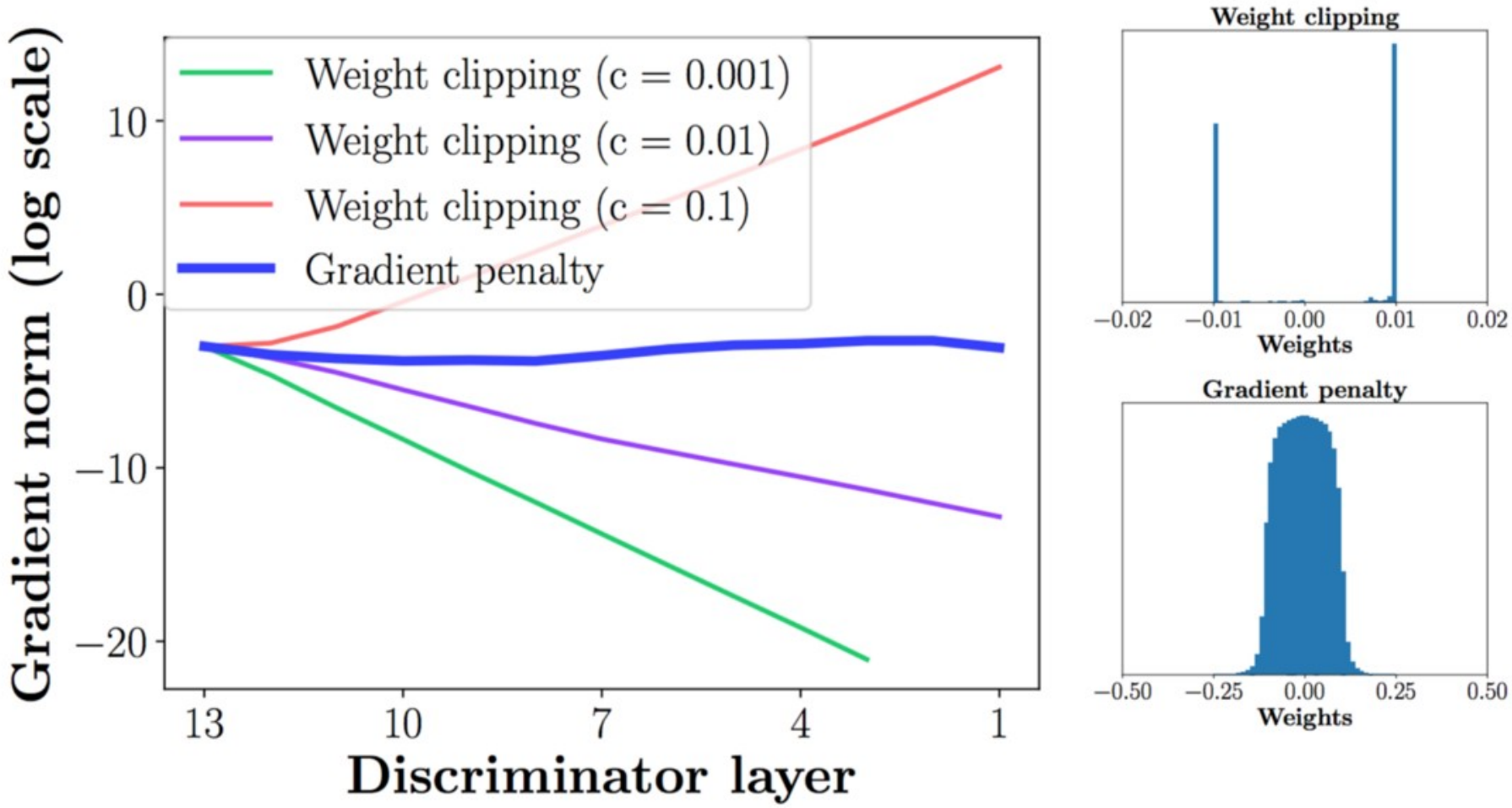
WGAN


$\nabla_w \left[\frac{1}{m} \sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m} \sum_{i=1}^{m} f_w(g_\theta(z^{(i)})) \right]$
Real image $\boldsymbol{x}$
Critic
f
cost
$z \sim \mathcal{N}(0, 1)$
or
$z \sim$ U (-1, 1)
Generator
$\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$

	Discriminator/Critic	Generator
GAN	$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right]$	$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)$
WGAN	$\nabla_w \frac{1}{m} \sum_{i=1}^{m} \left[f\left(x^{(i)}\right) - f\left(G\left(z^{(i)}\right)\right) \right]$	$\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} f\left(G\left(z^{(i)}\right)\right)$


Density of real
Density of fake
GAN Discriminator
WGAN Critic
Linear gradients
in a WGAN
Vanishing gradients
in regular GAN
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−8
−6
−4
−2
0
2
4
6
8

Problem with clipping parameter



W-GAN GP

We want the **critic function** $f(x)$ to satisfy:

$$|f(x_1) - f(x_2)| \leq \|x_1 - x_2\| \quad \text{for all } x_1, x_2$$

This is what it means to be **1-Lipschitz**.

In calculus terms, this means:

$$\|\nabla f(x)\| \leq 1 \quad \text{almost everywhere}$$

So we need to make sure that **the gradient of** f **is not too large anywhere** — otherwise, f can spike or behave non-smoothly.

$$L = \underbrace{\mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}} \sim \mathbb{P}_g} [D(\tilde{\boldsymbol{x}})] - \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathbb{P}_r} [D(\boldsymbol{x})]}_{\text{Original critic loss}} + \underbrace{\lambda \mathop{\mathbb{E}}_{\hat{\boldsymbol{x}} \sim \mathbb{P}_{\hat{\boldsymbol{x}}}} \left[(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2 - 1)^2 \right]}_{\text{Our gradient penalty}}.$$

Why we do not apply the regularizr on real or fake points

- The **optimal critic** in WGAN tends to have **linear behavior between real and fake**

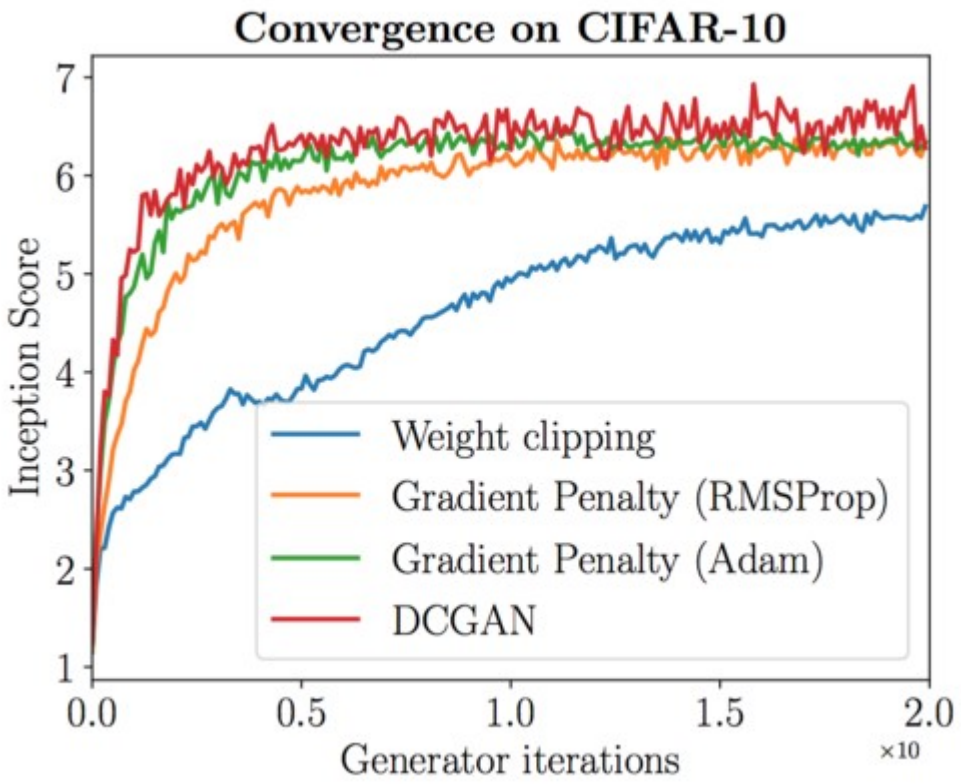
- So the region **between real and fake samples** is where:
  - The gradient might violate the Lipschitz constraint.
  - The critic function is expected to **change the most**.

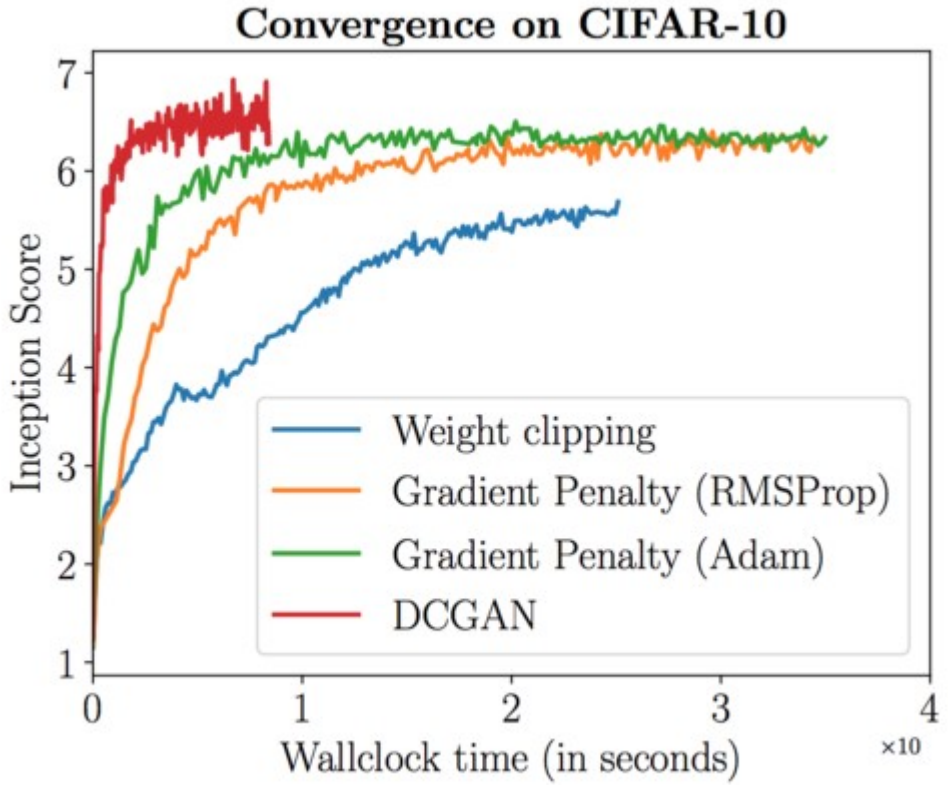
Algorithm 1 WGAN with gradient penalty. We use default values of $\lambda = 10$, $n_{\text{critic}} = 5$, $\alpha = 0.0001$, $\beta_1 = 0$, $\beta_2 = 0.9$.

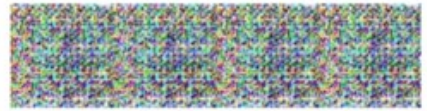
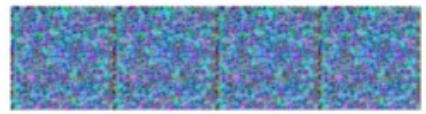
Require: The gradient penalty coefficient λ, the number of critic iterations per generator iteration n_{critic}, the batch size m, Adam hyperparameters α, β_1, β_2.
Require: initial critic parameters w_0, initial generator parameters θ_0.

while θ has not converged **do**
  for $t = 1, ..., n_{\text{critic}}$ **do**
    for $i = 1, ..., m$ **do**
      Sample real data $\boldsymbol{x} \sim \mathbb{P}_r$, latent variable $\boldsymbol{z} \sim p(\boldsymbol{z})$, a random number $\epsilon \sim U[0, 1]$.
      $\tilde{\boldsymbol{x}} \leftarrow G_\theta(\boldsymbol{z})$
      $\hat{\boldsymbol{x}} \leftarrow \epsilon \boldsymbol{x} + (1 - \epsilon)\tilde{\boldsymbol{x}}$
      $L^{(i)} \leftarrow D_w(\tilde{\boldsymbol{x}}) - D_w(\boldsymbol{x}) + \lambda(\|\nabla_{\hat{\boldsymbol{x}}} D_w(\hat{\boldsymbol{x}})\|_2 - 1)^2$
    end for
    $w \leftarrow \text{Adam}(\nabla_w \frac{1}{m} \sum_{i=1}^{m} L^{(i)}, w, \alpha, \beta_1, \beta_2)$
  end for
  Sample a batch of latent variables $\{\boldsymbol{z}^{(i)}\}_{i=1}^{m} \sim p(\boldsymbol{z})$.
  $\theta \leftarrow \text{Adam}(\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} -D_w(G_\theta(\boldsymbol{z})), \theta, \alpha, \beta_1, \beta_2)$
end while

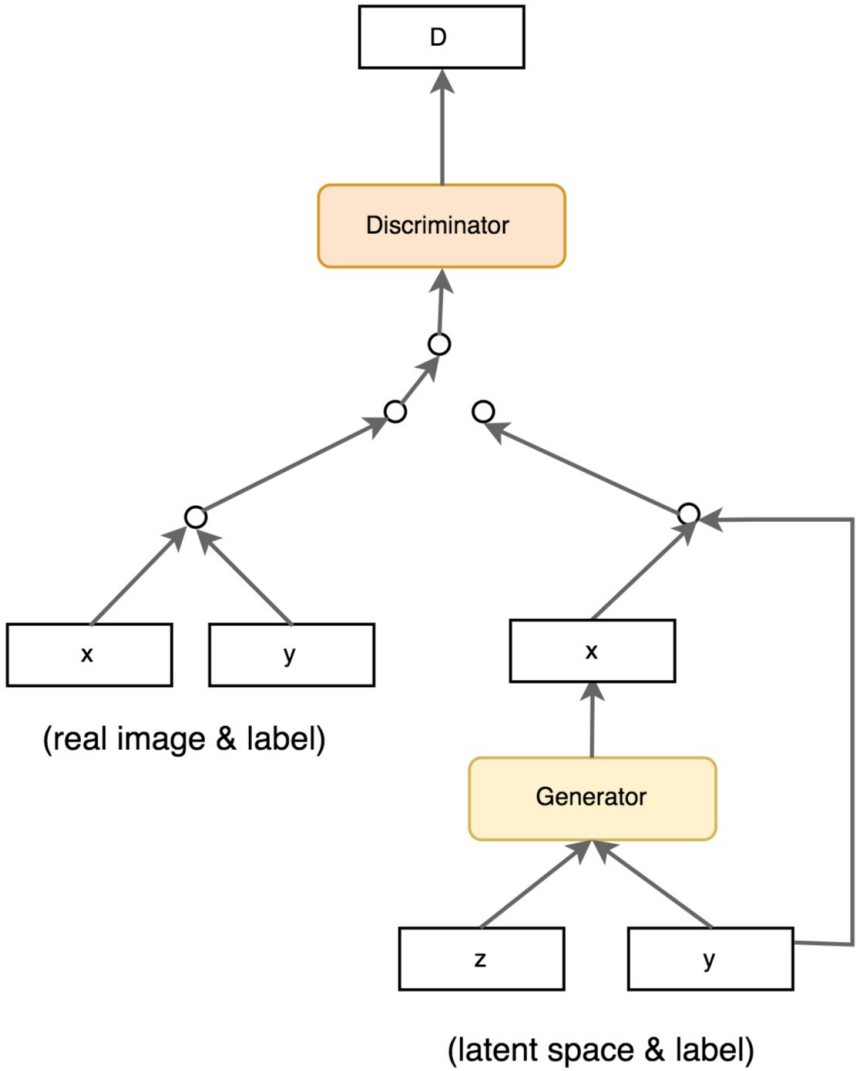
WGAN GP experiments





DCGAN	LSGAN	WGAN (clipping)	WGAN-GP (ours)
Baseline (G: DCGAN, D: DCGAN)				
G: No BN and a constant number of filters, D: DCGAN				
G: 4-layer 512-dim ReLU MLP, D: DCGAN				
No normalization in either G or D				
Gated multiplicative nonlinearities everywhere in G and D				
tanh nonlinearities everywhere in G and D				
101-layer ResNet G and D				

CGAN – using the labels



InfoGAN – labels as latent variables