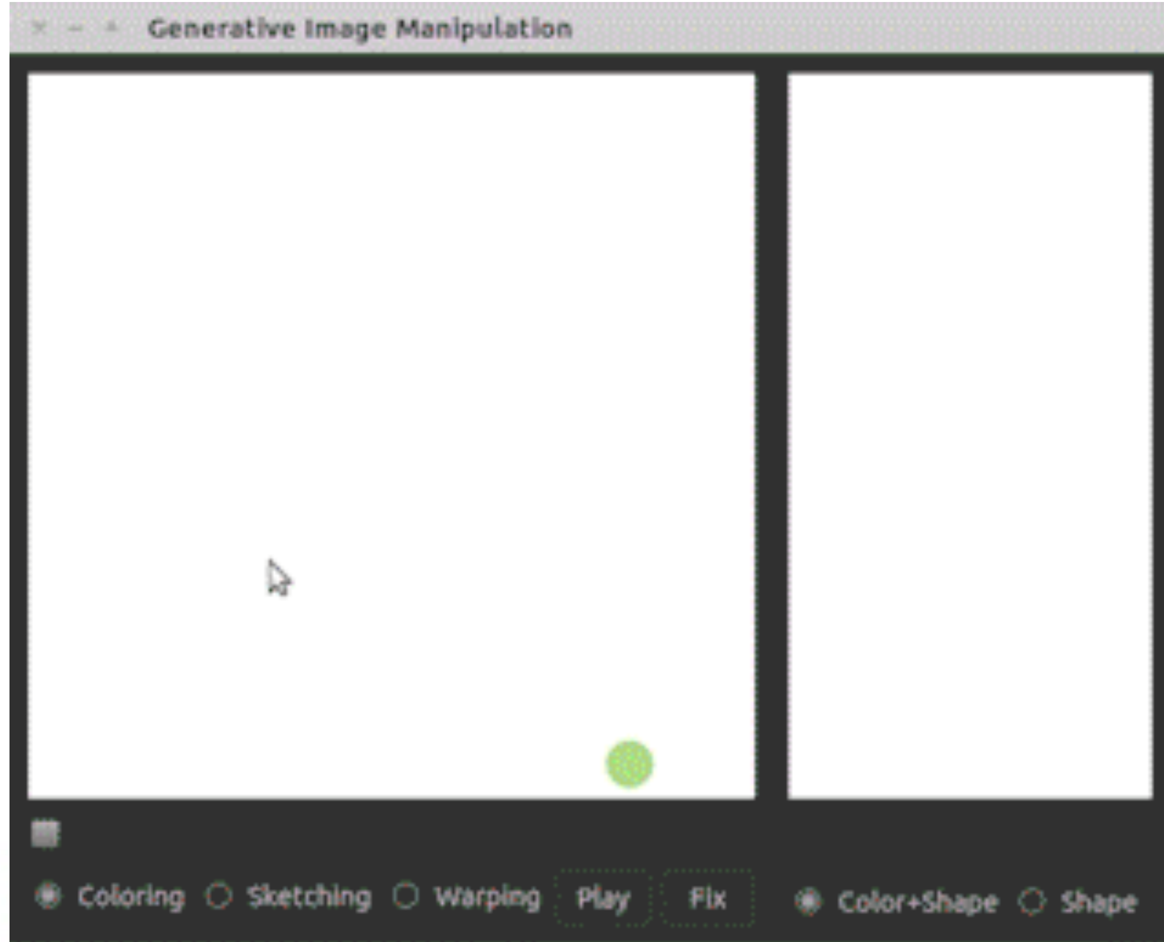
Head First Generative Adversarial Networks From Theoretic View

Yanran Li

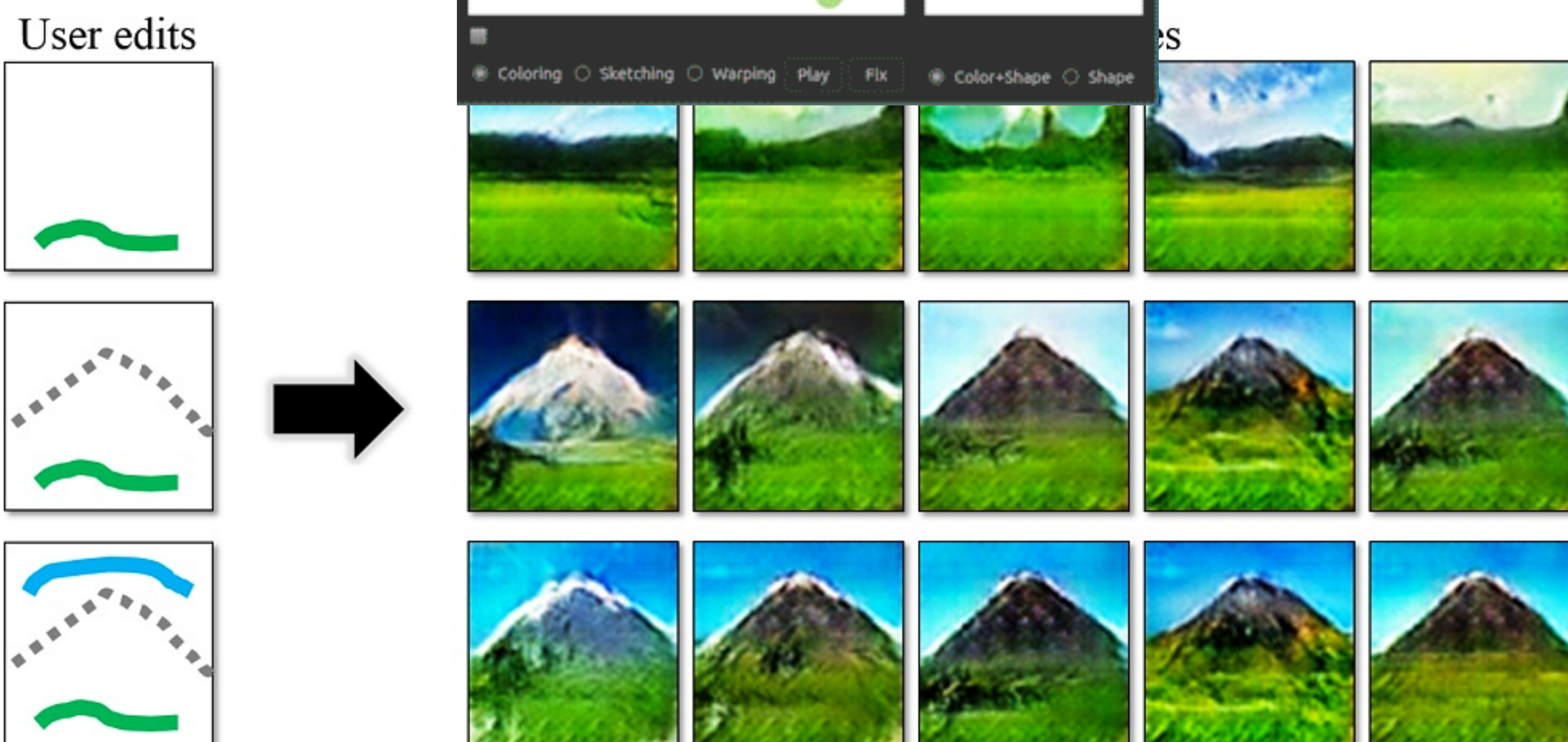
The Hong Kong Polytechnic University

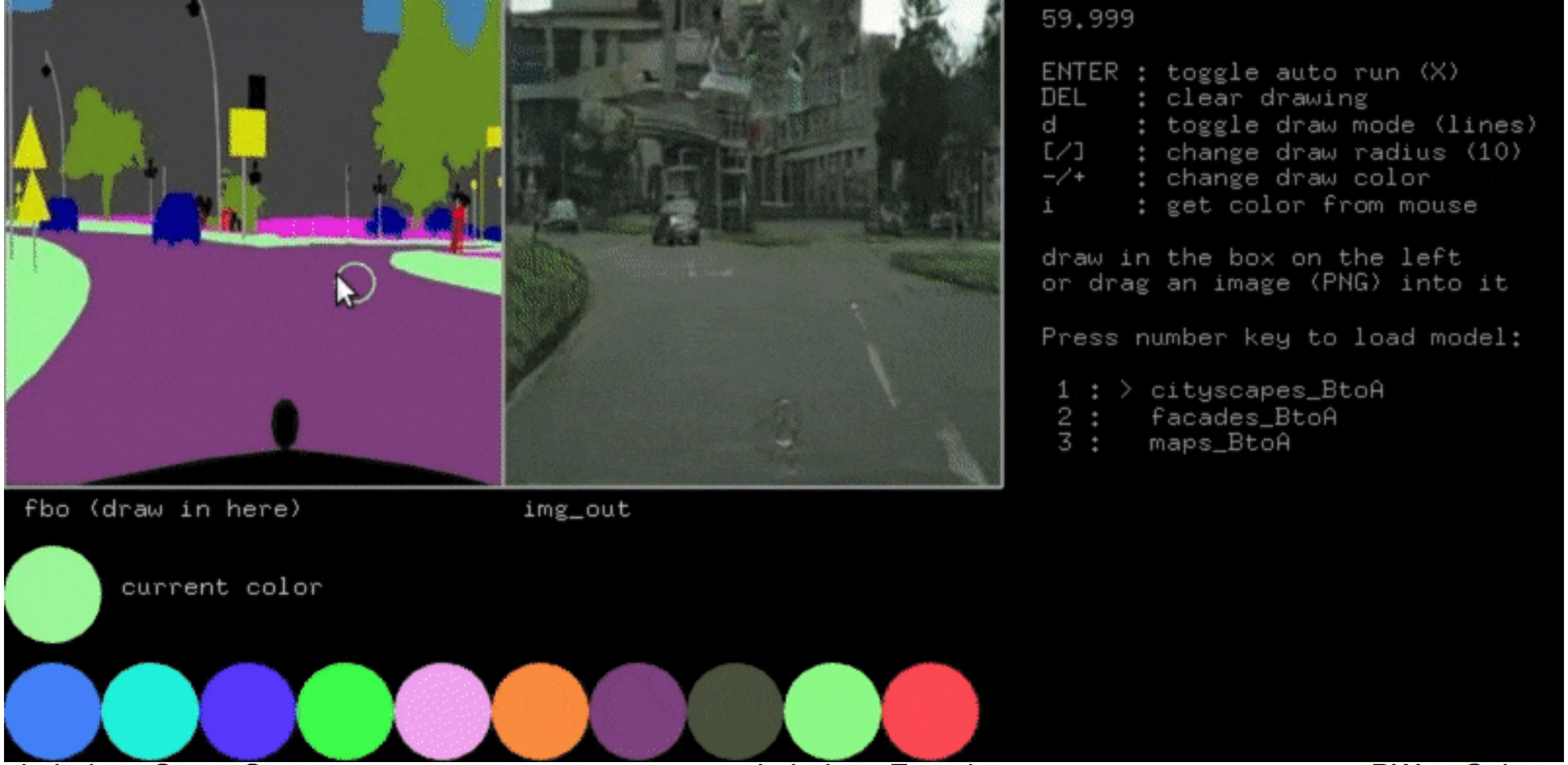
MedGAN ID-CGAN CoGAN LR-GAN CGAN IcGAN b-GAN LS-GAN AffGAN DiscoGAN AdaGAN LSGAN CatGAN LAPGAN iGAN IAN InfoGAN AMGAN MPM-GAN MIX+GAN McGAN C-RNN-GAN DR-GAN MGAN GoGAN BS-GAN C-VAE-GAN FF-GAN DCGAN 3D-GAN CCGAN AC-GAN BiGAN GAWWN DualGAN CycleGAN GP-GAN Bayesian GAN AnoGAN EBGAN Context-RNN-GAN MAGAN MAD-GAN DTN ALI f-GAN MARTA-GAN ArtGAN BEGAN MalGAN AL-CGAN

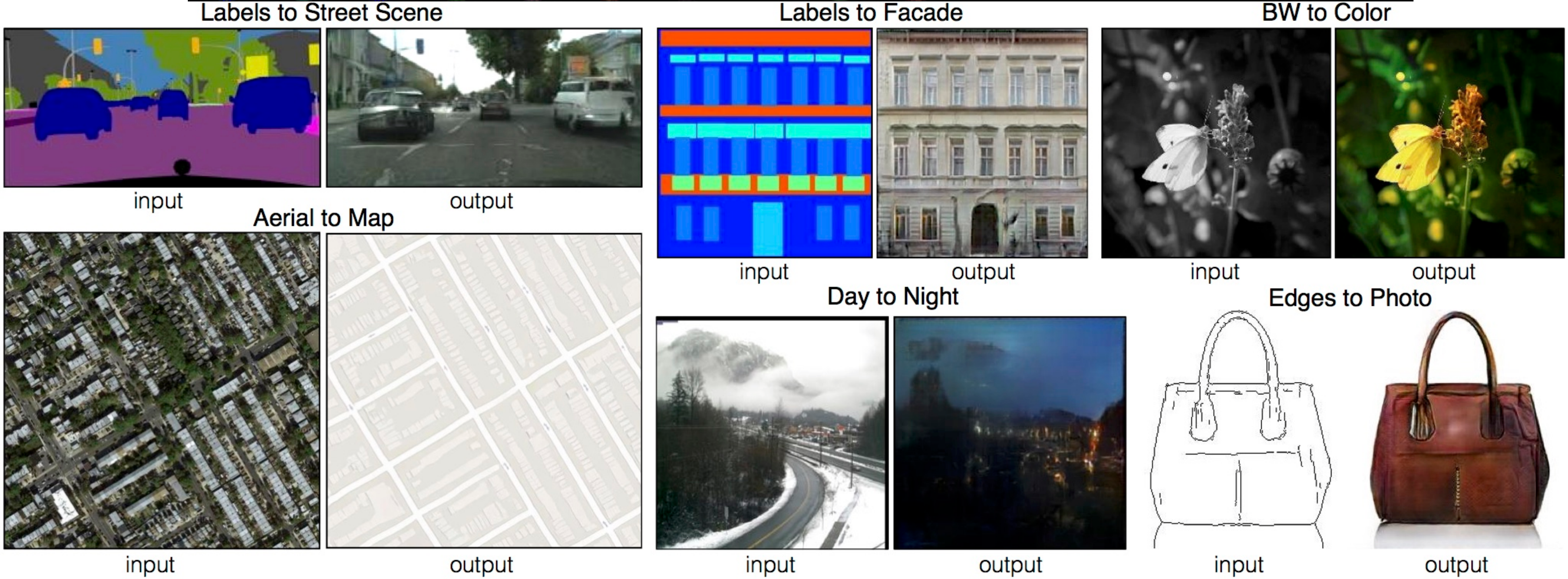
(https://github.com/kaonashi-tyc/zi2zi)

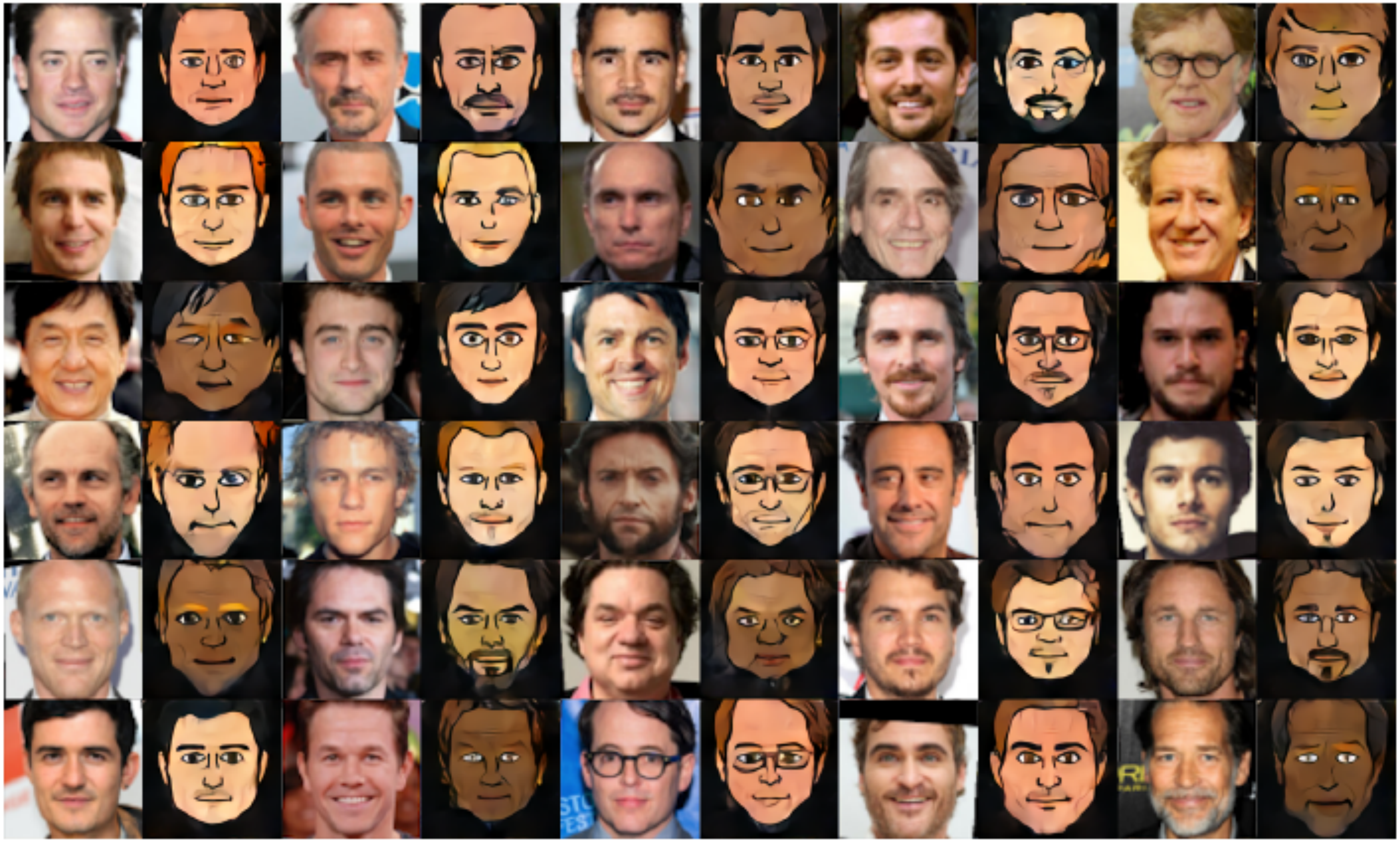


(https://github.com/junyanz/iGAN)



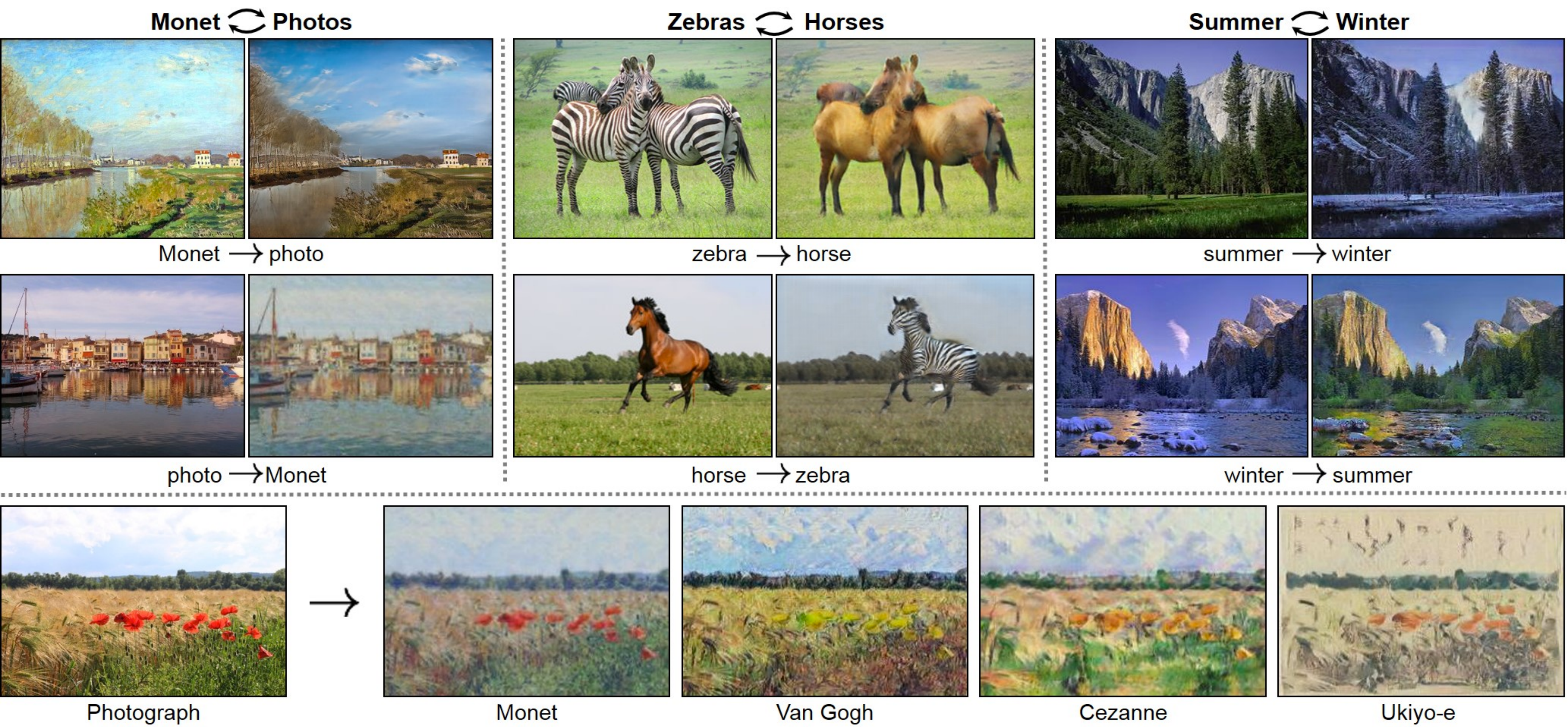

59.999
ENTER : toggle auto run (X)
DEL : clear drawing
d : toggle draw mode (lines)
[/] : change draw radius (10)
-/+ : change draw color
i : get color from mouse
draw in the box on the left
or drag an image (PNG) into it
Press number key to load model:
1 : > cityscapes_BtoA
2 : facades_BtoA
3 : maps_BtoA
fbo (draw in here)
img_out
current color


Labels to Street Scene
input
output
Labels to Facade
input
output
BW to Color
input
output
Aerial to Map
input
output
Day to Night
input
output
Edges to Photo
input
output



(Taigman et al., 2017)

(https://junyanz.github.io/CycleGAN/)



(genekogan@Twitter)

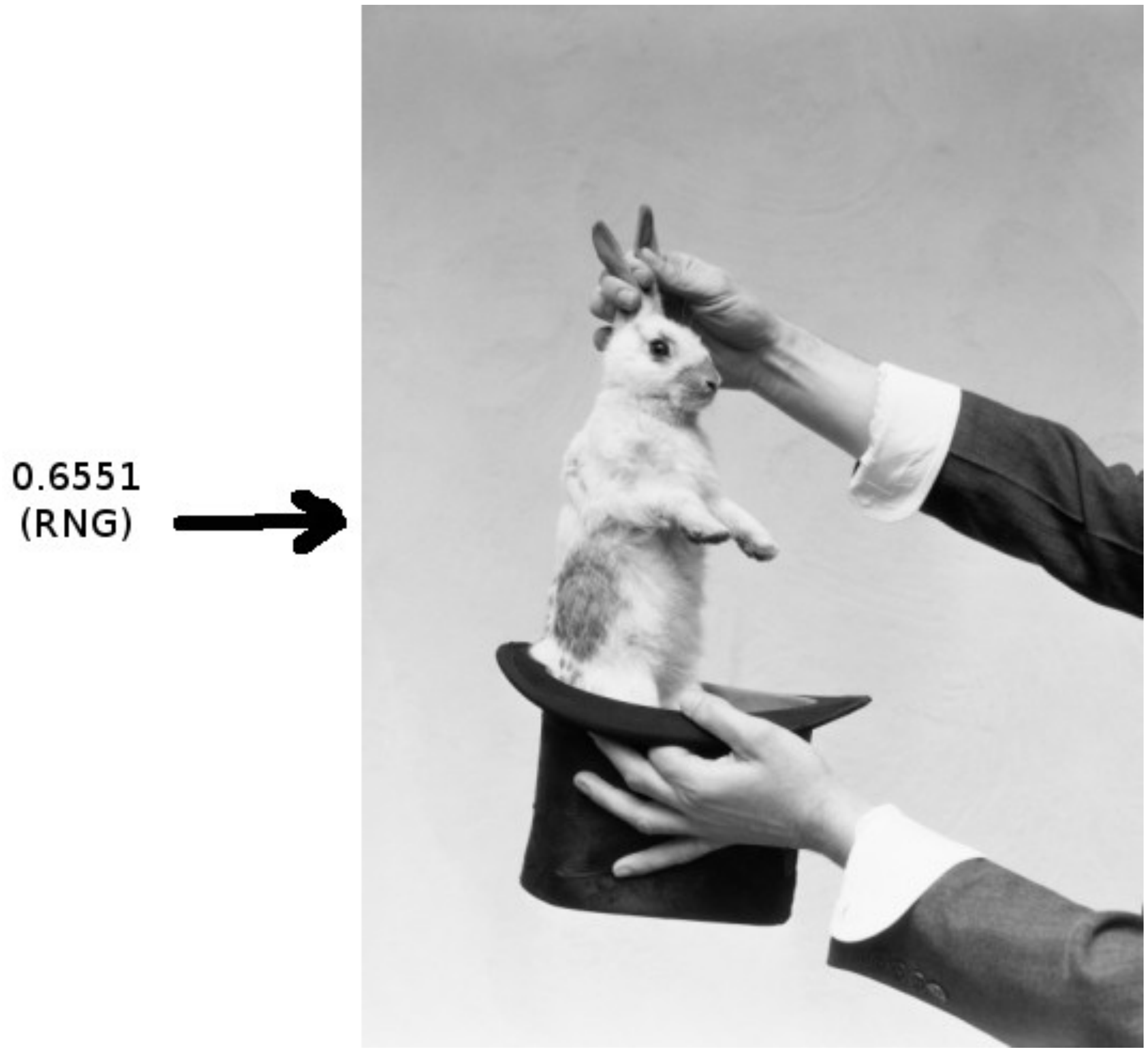
Content

- Generative Adversarial Networks
    - Basics and Attractiveness
    - Difficulties
- Solution 1: Partial and Fine-grained Guidance
- Solution 2: Encoder-incorporated
- Solution 3: Wasserstein Distance

Content

- Generative Adversarial Networks
  - Basics and Attractiveness
  - Difficulties
- Solution 1: Partial and Fine-grained Guidance
- Solution 2: Encoder-incorporated
- Solution 3: Wasserstein Distance

Generative Adversarial Networks



(Eric Jang’s blog)

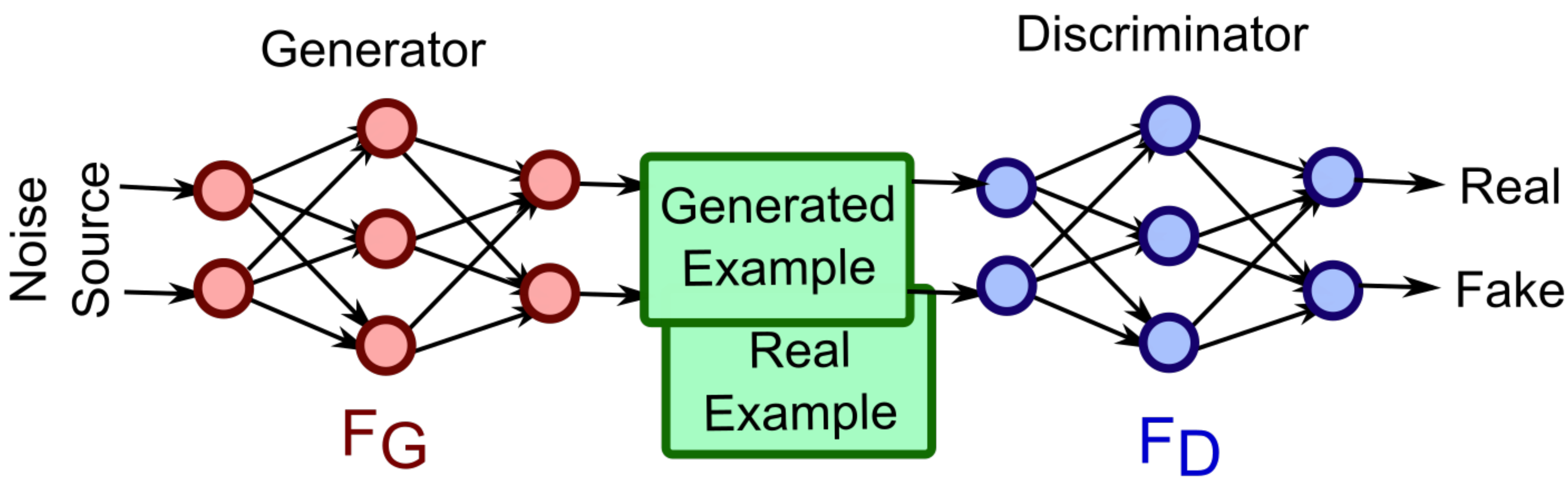
Generative Adversarial Networks

- A counterfeiter-police game between two components: a generator ***G*** and a discriminator ***D***
- ***G:*** counterfeiter, trying to fool police with fake currency
- ***D:*** policy, trying to detect the counterfeit currency
- Competition drives both to improve, until counterfeits are *indistinguishable* from genuine currency

(Nicholas Gutenberg's blog)

Generative Adversarial Networks

- A min-max game between two components: a generator ***G*** and a discriminator ***D***



(Nicholas Gutenberg's blog)

Generative Adversarial Networks

- A min-max game between two components: a generator ***G*** and a discriminator ***D***

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

D predicting that real data is genuine

D predicting that G's generated data is fake

(Goodfellow et al., 2014)

Generative Adversarial Networks

- A min-max game between two components: a generator ***G*** and a discriminator ***D***

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

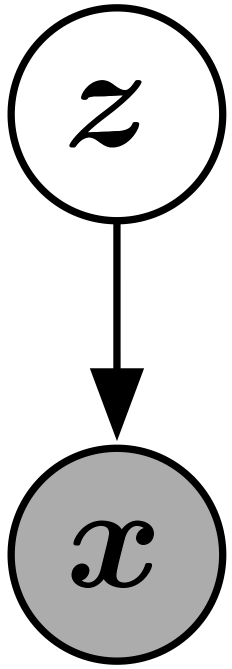
D predicting that real data is genuine

D predicting that G's generated data is fake

- ***D***'s goal: maximize *V(D,G)*
  G's goal: minimize max *V(D,G)*

(Goodfellow et al., 2014)

Attractiveness



- Generator Networks $\boldsymbol{x} = G(\boldsymbol{z}; \boldsymbol{\theta}^{(G)})$
- It is only required that, G is differentiable.
- So, having training data $x \sim p_{data}(x)$
what we want is a model that can draw samples $x \sim p_{model}(x)$, where $p_{model} \approx p_{data}$
- Don’t write a formula for $p_{data}(x)$, just learn to draw sample directly.

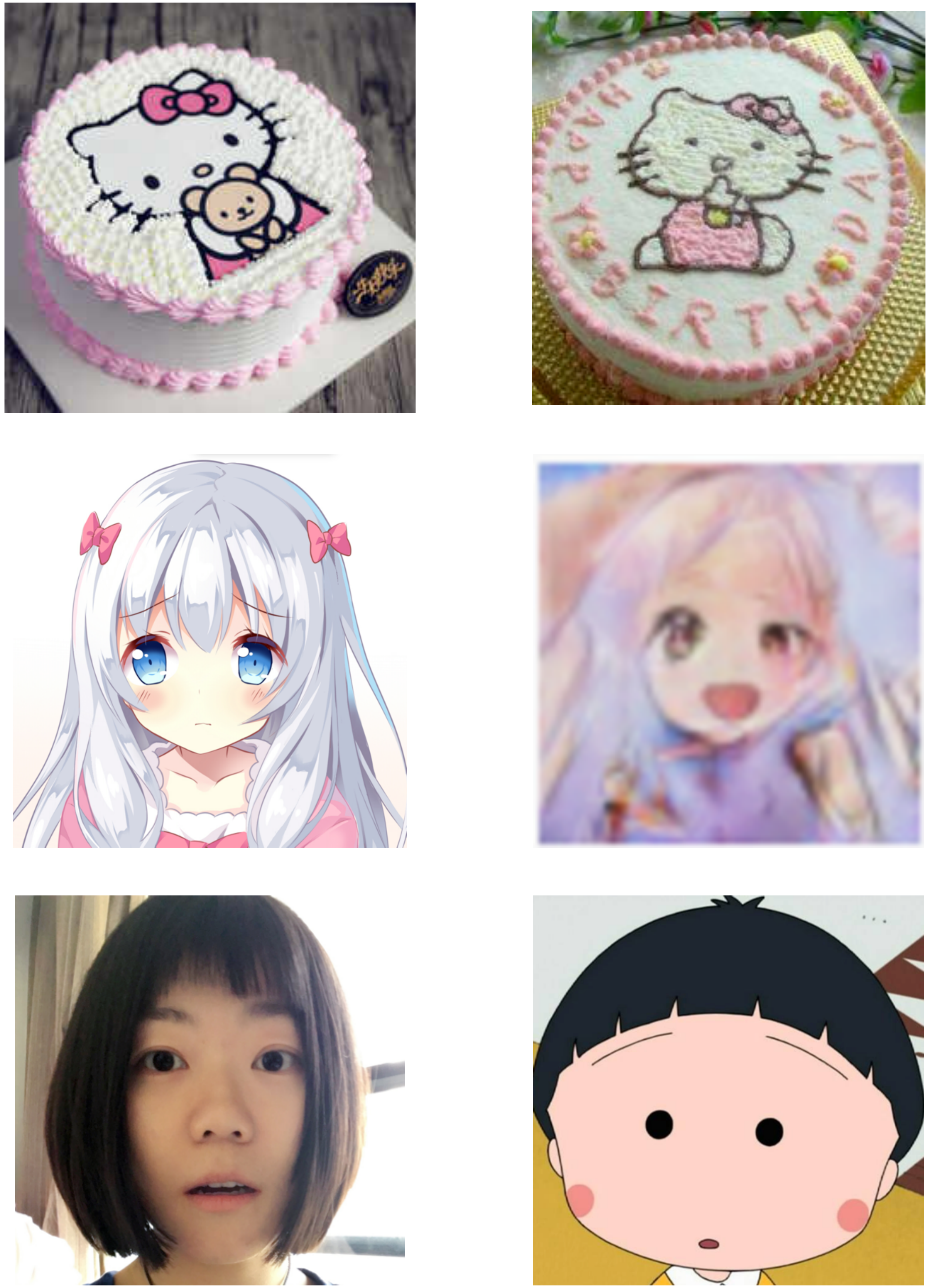
“There’s no free lunch.”

–From Economics

Original

Generated

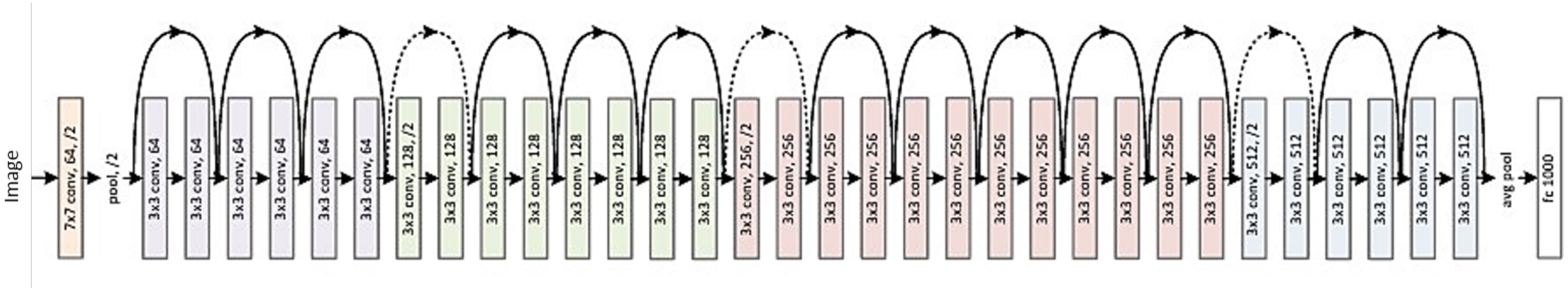
Original



Generated

Difficulty 1

- The gradient issues existed in deep neural networks
- The deeper, the more difficult



Objectives for GAN

- The objective of ***D:***

$$L(D, g_\theta) = \mathbb{E}_{x\sim\mathbb{P}_r}[\log D(x)] + \mathbb{E}_{x\sim\mathbb{P}_g}[\log(1 - D(x))]$$

- The objective of ***G:***
  - the original: $\mathbb{E}_{z\sim p(z)}[\log(1 - D(g_\theta(z)))]$
  - the alternative: $\mathbb{E}_{z\sim p(z)}\left[-\log D(g_\theta(z))\right]$
  - *Why alternative?*

Difficulty 2

- using the original form of the objective of ***G***

$$\mathbb{E}_{z\sim p(z)}[\log(1 - D(g_\theta(z)))]$$

  will result in gradient vanishing issue of ***D*** for ***G*** because *intuitively*, at the very early phase of training, ***D*** is very easy to be confident in detecting ***G***, so ***D*** will output almost always 0

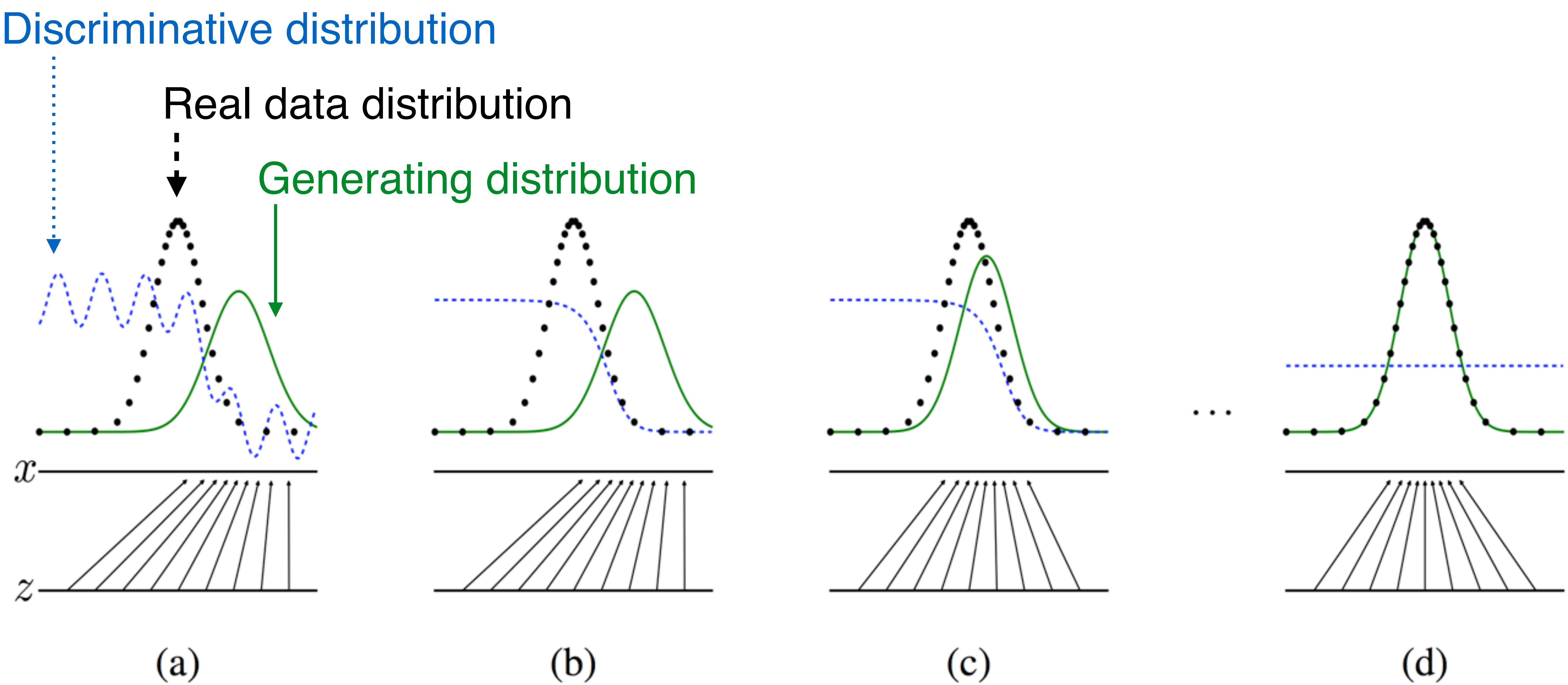
Difficulty 2

- using the original form of the objective of ***G***

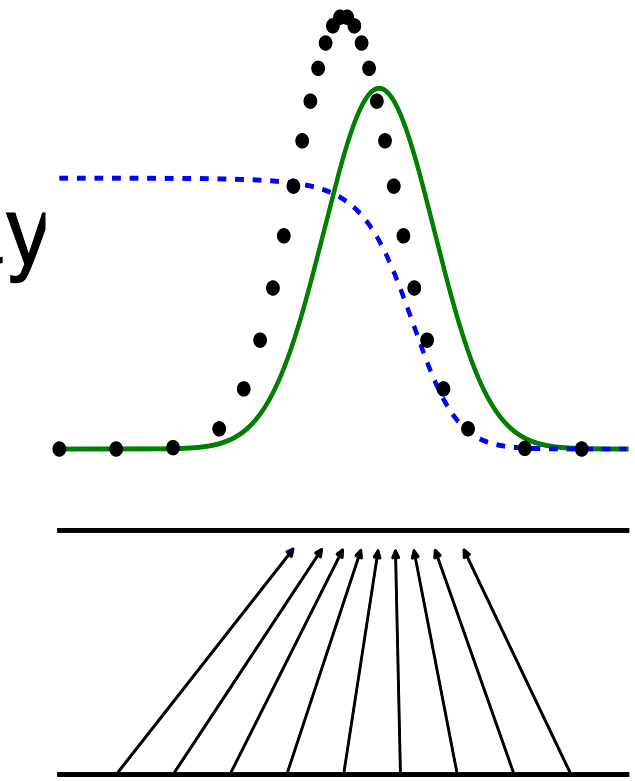
$$\mathbb{E}_{z\sim p(z)}[\log(1 - D(g_\theta(z)))]$$

will result in gradient vanishing issue of ***D*** for ***G*** because *theoretically*, when ***D*** is *optimal*, minimizing the loss is equal to minimizing the *JS divergence* (Arjovsky & Bottou, 2017)

Difficulty 2



Difficulty 2



- The optimal D for any P_r and P_g is alway

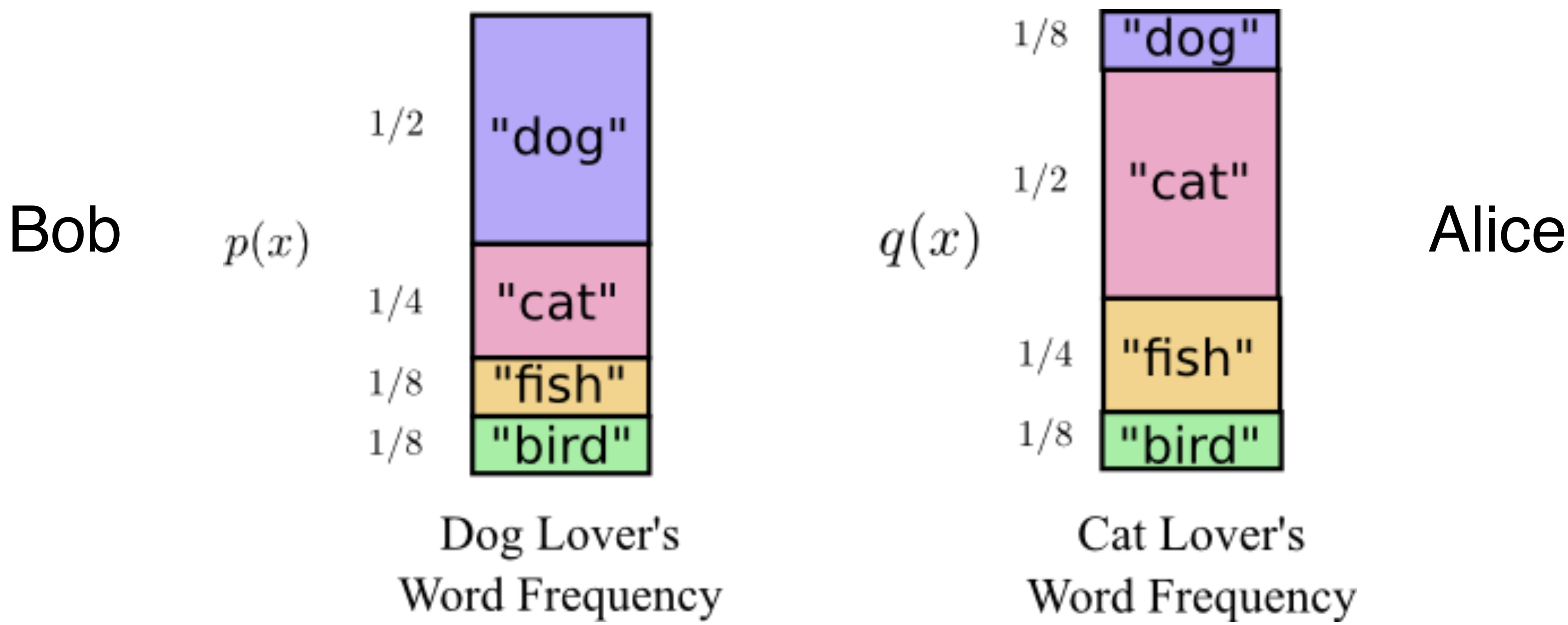
$$D^*(x) = \frac{P_r(x)}{P_r(x) + P_g(x)}$$

and that $$L(D^*, g_\theta) = 2JSD(\mathbb{P}_r \| \mathbb{P}_g) - 2\log 2$$

so, when ***D*** is *optimal*, minimizing the loss is equal to minimizing the *JS divergence* (Arjovsky & Bottou, 2017)

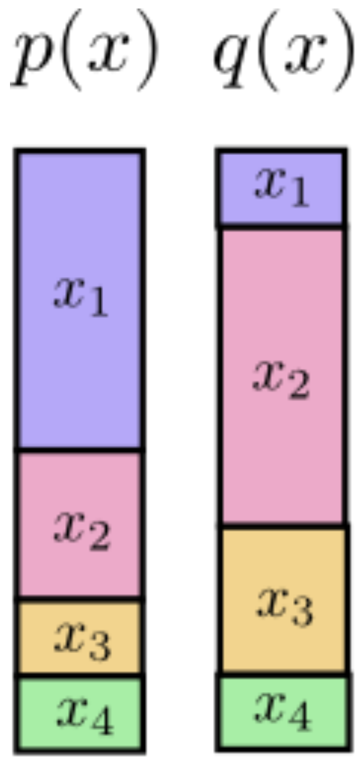
(Goodfellow et al., 2014)

Recall KL and JS Divergence



(Colah's blog)

Recall KL and JS Divergence



Bob Alice

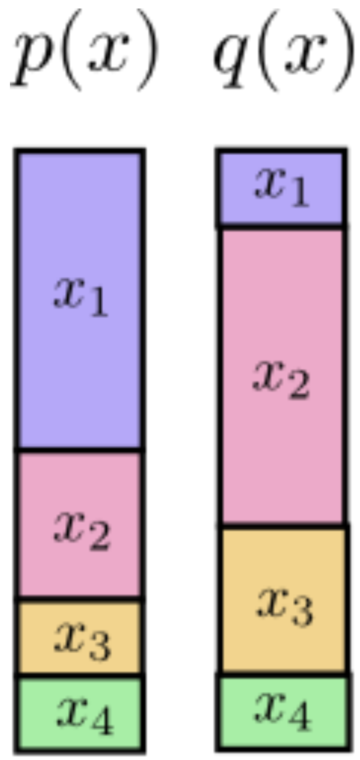
Cross-Entropy: $H_p(q)$

Average Length
of message from q(x)
using code for p(x).

$$H_p(q) = \sum_x q(x) \log_2\left(\frac{1}{p(x)}\right)$$

(Colah's blog)

Recall KL and JS Divergence



Bob Alice

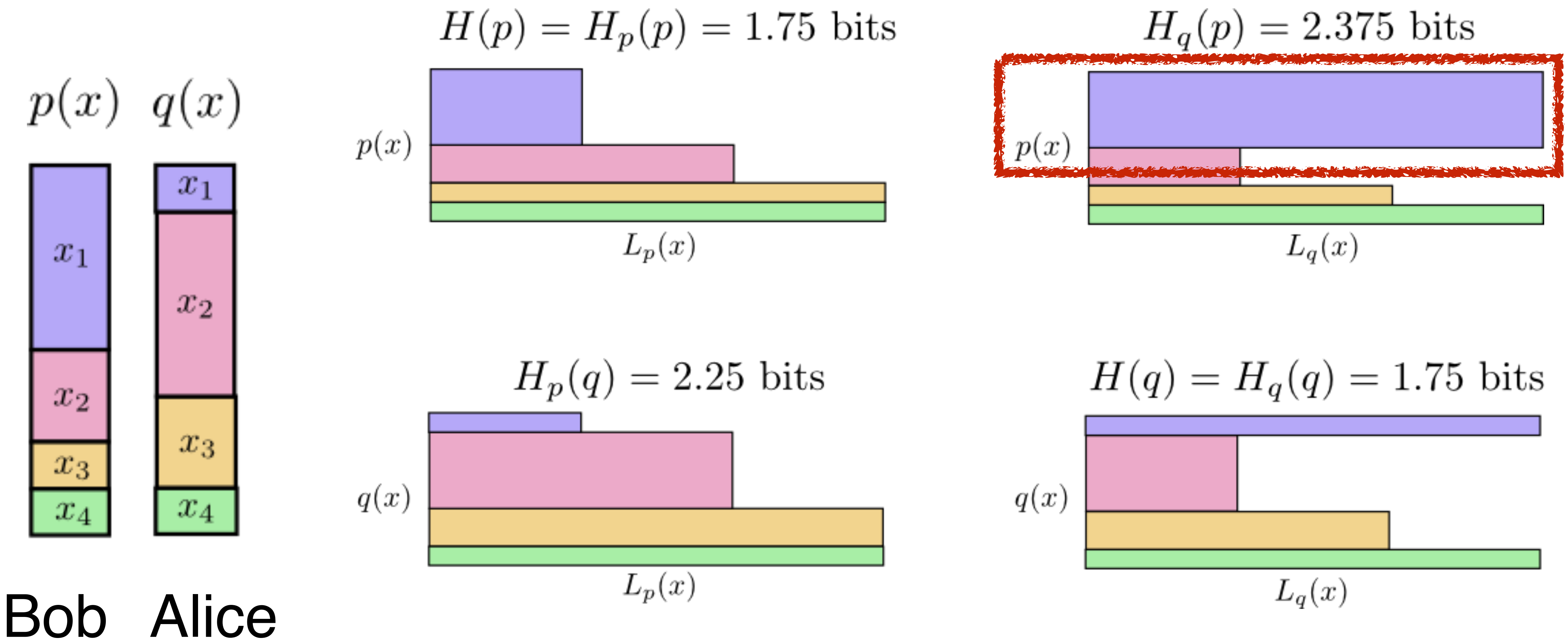
Cross-Entropy: $H_p(q)$

Average Length
of message from q(x)
using code for p(x).

$$H_p(q) = \sum_x q(x) \log_2\left(\frac{1}{p(x)}\right)$$

(Colah's blog)

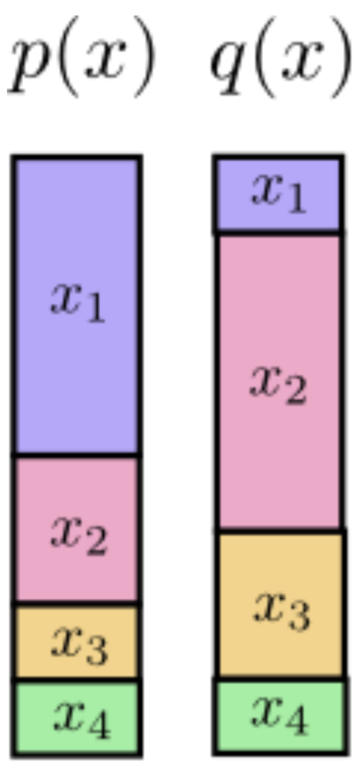
Recall KL and JS Divergence



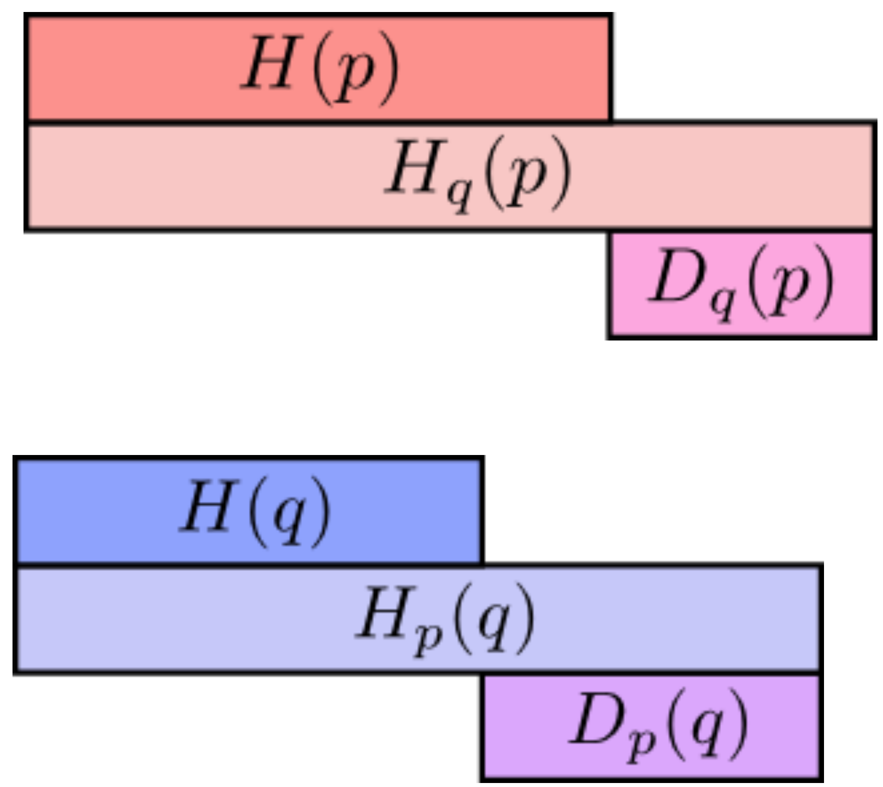
$$H_p(q) \neq H_q(p)$$

(Colah's blog)

Recall KL and JS Divergence



Bob Alice

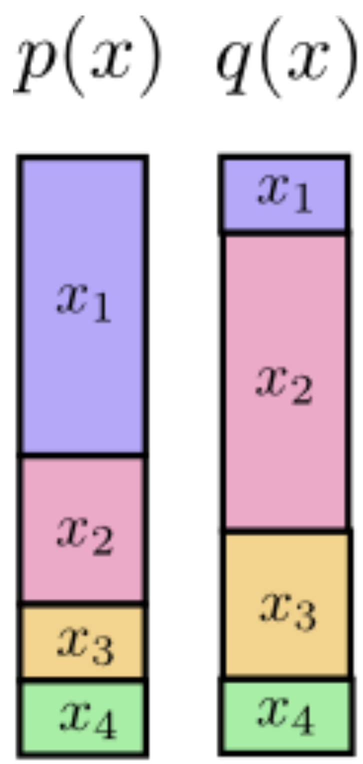


$$H_p(q) \neq H_q(p)$$

KL Divergence $D_q(p) = H_q(p) - H(p)$

(Colah's blog)

Recall KL and JS Divergence



Bob Alice

JS Divergence

$$D_{JS}(p|q) = D_{JS}(q|p) = \frac{1}{2}D_{KL}(p|r) + \frac{1}{2}D_{KL}(q|r)$$

$$r = \tfrac{1}{2}(p+q)$$

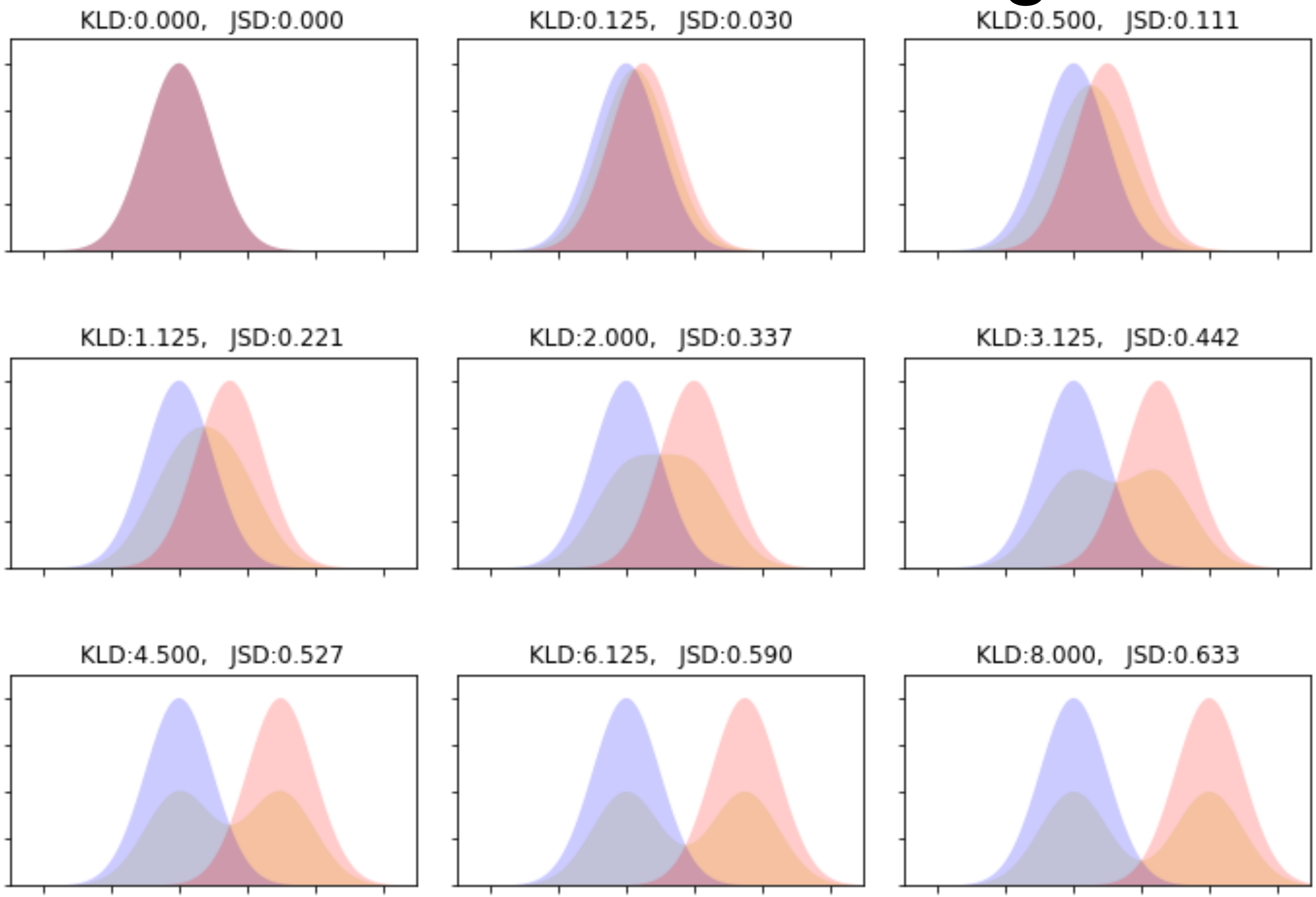
Be symmetric!

$$H_p(q) \neq H_q(p)$$

KL Divergence $D_q(p) = H_q(p) - H(p)$

(Colah's blog)

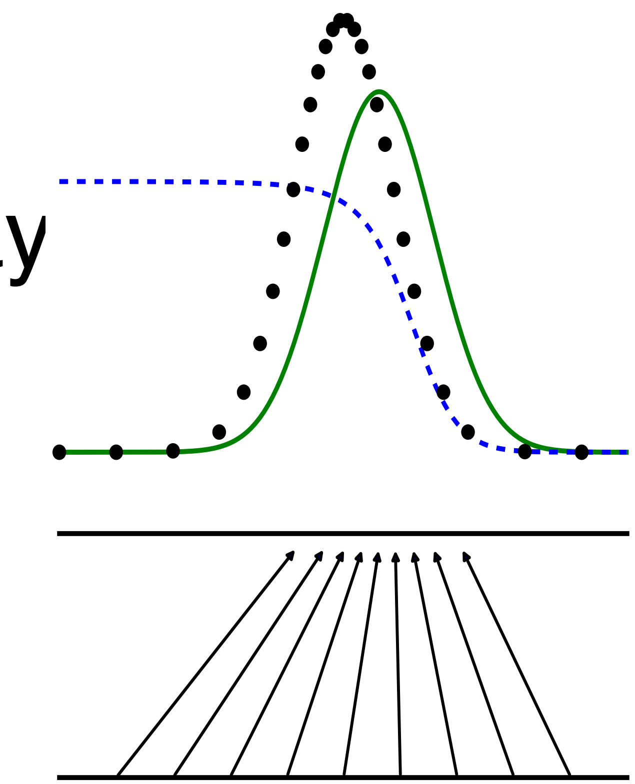
Recall KL and JS Divergence



(Hatena's blog)

Difficulty 2

- The optimal D for any P_r and P_g is alway



$$D^*(x) = \frac{P_r(x)}{P_r(x) + P_g(x)}$$

and that $$L(D^*, g_\theta) = 2JSD(\mathbb{P}_r \| \mathbb{P}_g) - 2\log 2$$

so, when ***D*** is *optimal*, minimizing the loss is equal to minimizing the *JS divergence* (Goodfellow et al., 2014)

(Goodfellow et al., 2014)

Difficulty 2

- when:

$$L(D^*, g_\theta) = 2JSD(\mathbb{P}_r \| \mathbb{P}_g) - 2\log 2$$

- The JS divergence for the two distributions P_r and P_g is (almost) always log2 because P_r and P_g hardly can overlap (Arjovsky & Bottou, 2017, Theorem 2.1~2.3)

- This results in vanishing gradient in theory!

The alternative objective

- The alternative objective of ***G:***

$$\mathbb{E}_{z \sim p(z)} \left[-\log D(g_\theta(z)) \right]$$

- Instead of minimizing, let ***G*** maximize the log-probability of the discriminator being mistaken

- It is heuristically motivated that generator can still learn even when discriminator successfully rejects all generator samples, but not theoretically guaranteed

(Goodfellow's tutorial)

Difficulty 3

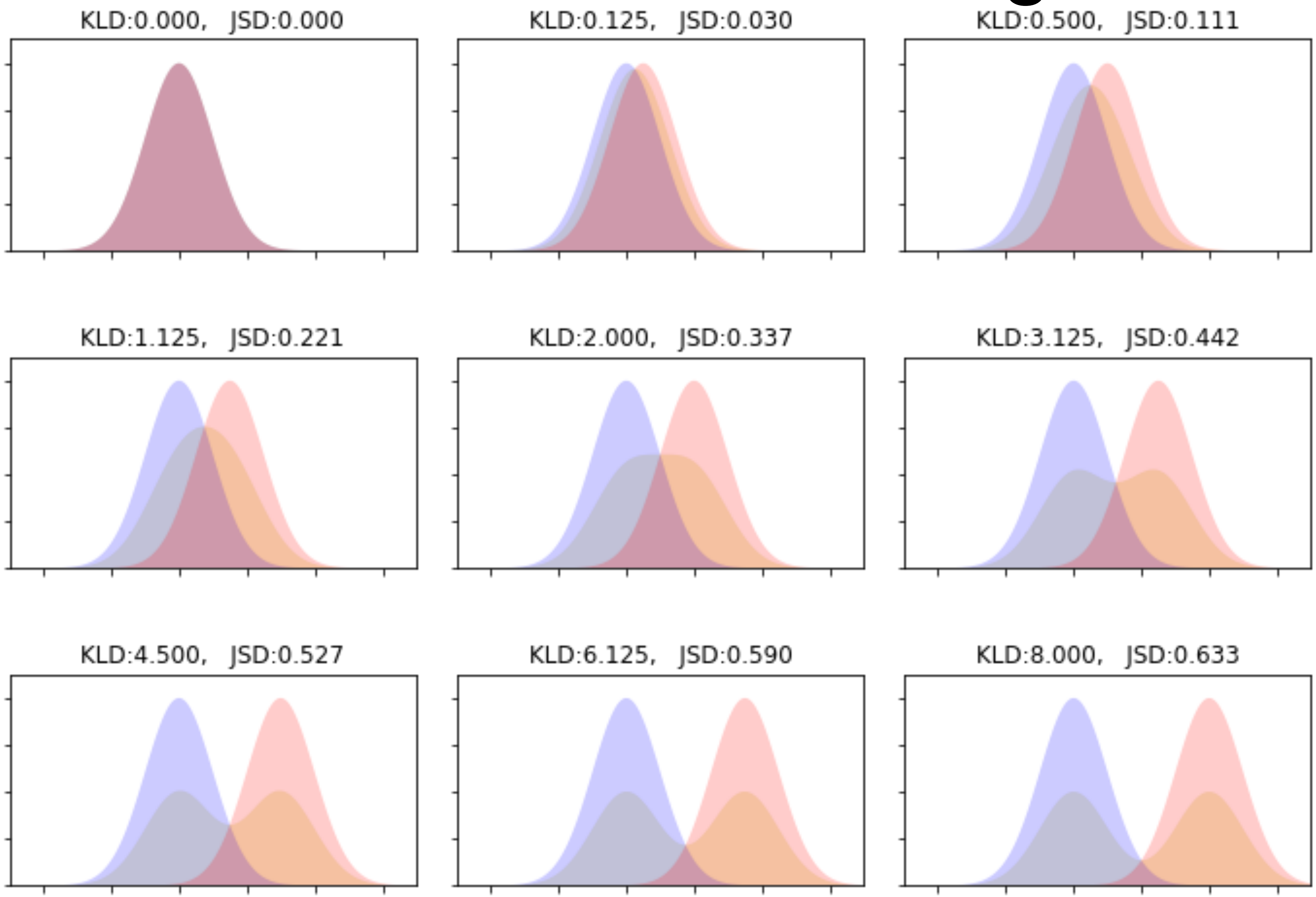
- using the alternative form of the objective of ***G***

$$\mathbb{E}_{z\sim p(z)}\left[-\log D(g_\theta(z))\right]$$

will result in gradient unstable issue and mode missing problem because *theoretically*, when ***D*** is *optimal*, minimizing the loss is equal to minimizing the *KL divergence* meanwhile maximizing the *JS divergence* (Arjovsky & Bottou, 2017, Theorem 2.5):

$$KL(\mathbb{P}_{g_\theta}||\mathbb{P}_r) - 2JSD(\mathbb{P}_{g_\theta}||\mathbb{P}_r)]$$

Recall KL and JS Divergence



(Hatena's blog)

Difficulty 3

- minimizing the *KL divergence* meanwhile maximizing the *JS divergence* is crazy:

$$KL(\mathbb{P}_{g_\theta}||\mathbb{P}_r) - 2JSD(\mathbb{P}_{g_\theta}||\mathbb{P}_r)]$$

- which results in gradient unstable issue

Difficulty 3

- minimizing the *KL divergence* only is biased:

$$KL(\mathbb{P}_{g_\theta}||\mathbb{P}_r) - 2JSD(\mathbb{P}_{g_\theta}||\mathbb{P}_r)]$$

- because *KL divergence* is asymmetric, and thus it is not equally treated when ***G*** generates an unreal sample and when ***G*** fails to generate real sample

- Therefore, ***G*** will generate too many few-mode (less diverse) but real samples , a safer strategy

Content

- Generative Adversarial Networks
  - Basics and Attractiveness
  - Difficulties
- Solution 1: Partial and Fine-grained Guidance
- Solution 2: Encoder-incorporated
- Solution 3: Wasserstein Distance

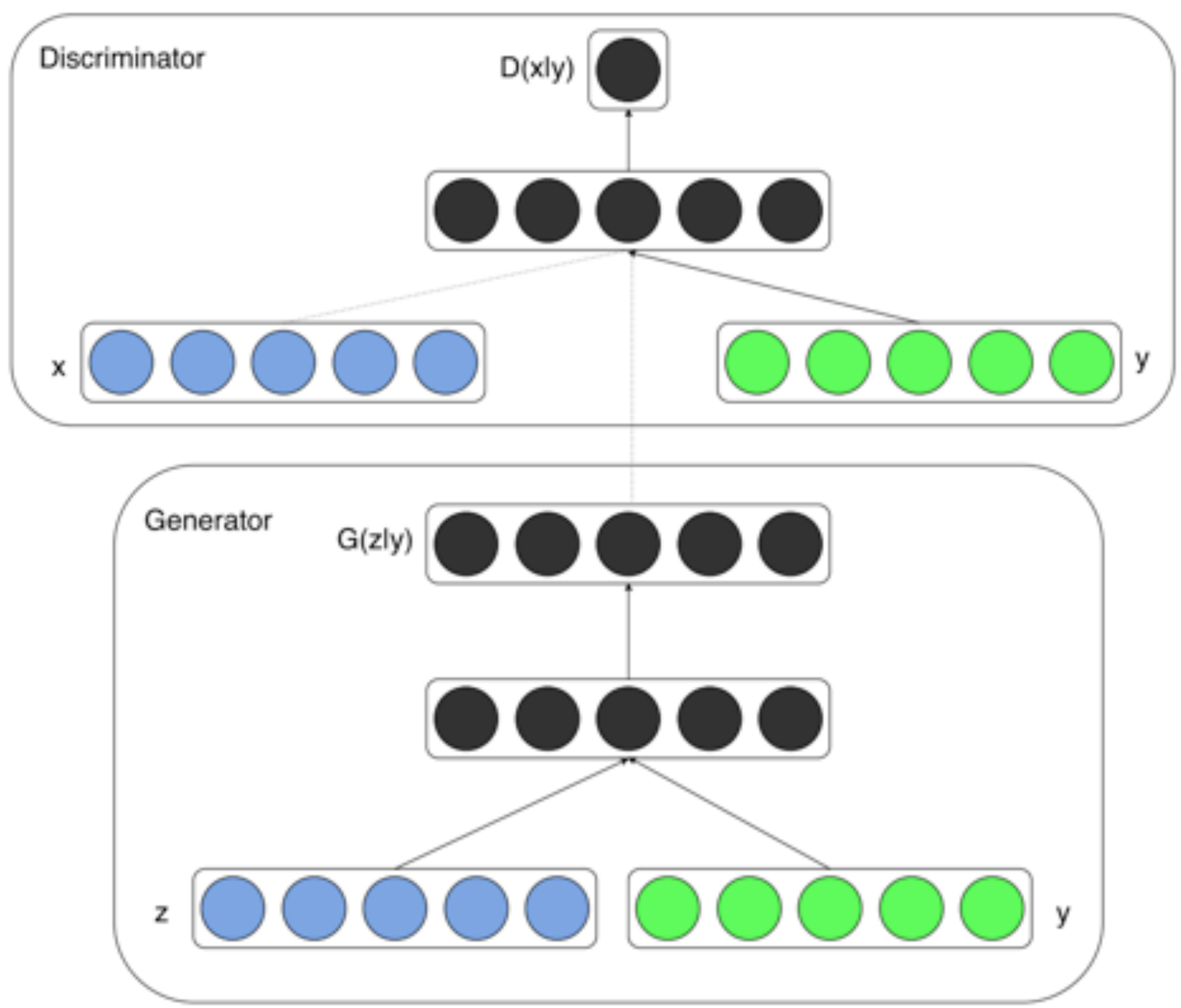
Solution 1.1: Partial Guidance

- Conditional GANs (Mirza & Osindero, 2014)
- Improved GAN (Salimans et al., 2016)
- iGAN/GVM (Zhu et al., 2016)
- pix2pix (Isola et al., 2017)
- GP-GAN (Wu et al., 2017)

Solution 1.1: Partial Guidance

- Conditional GANs (Mirza & Osindero, 2014)

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



(Mirza & Osindero, 2014)

Solution 1.1: Partial Guidance

- Improved GAN (Salimans et al., 2016)
  - feature matching

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

  - minibatch discrimination

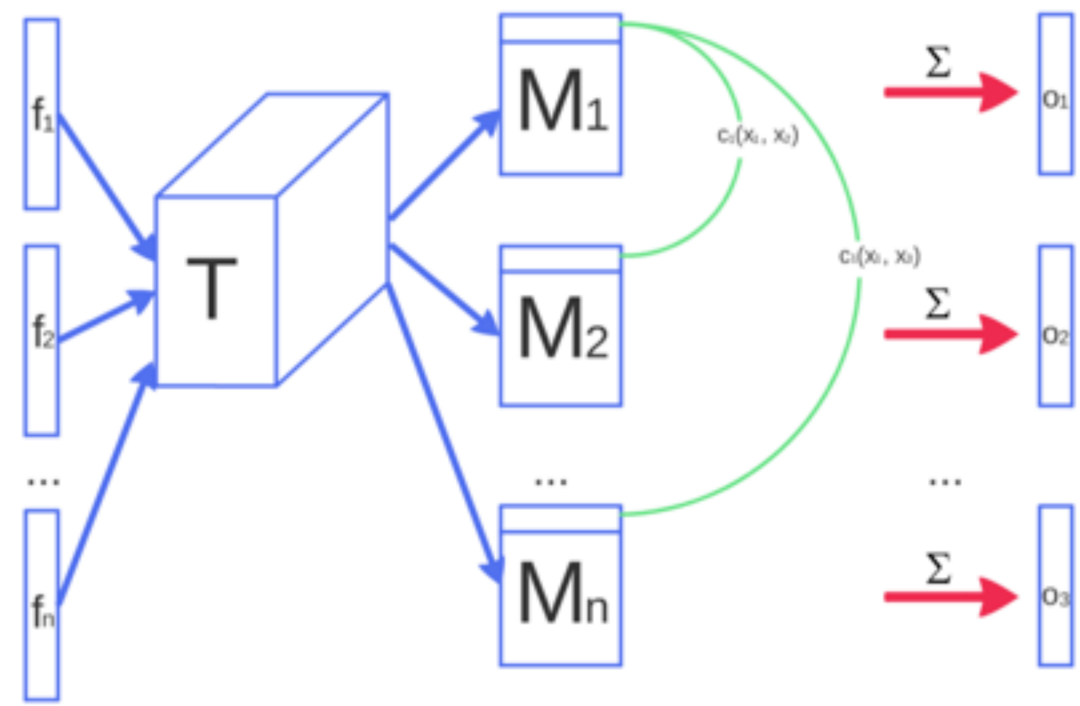
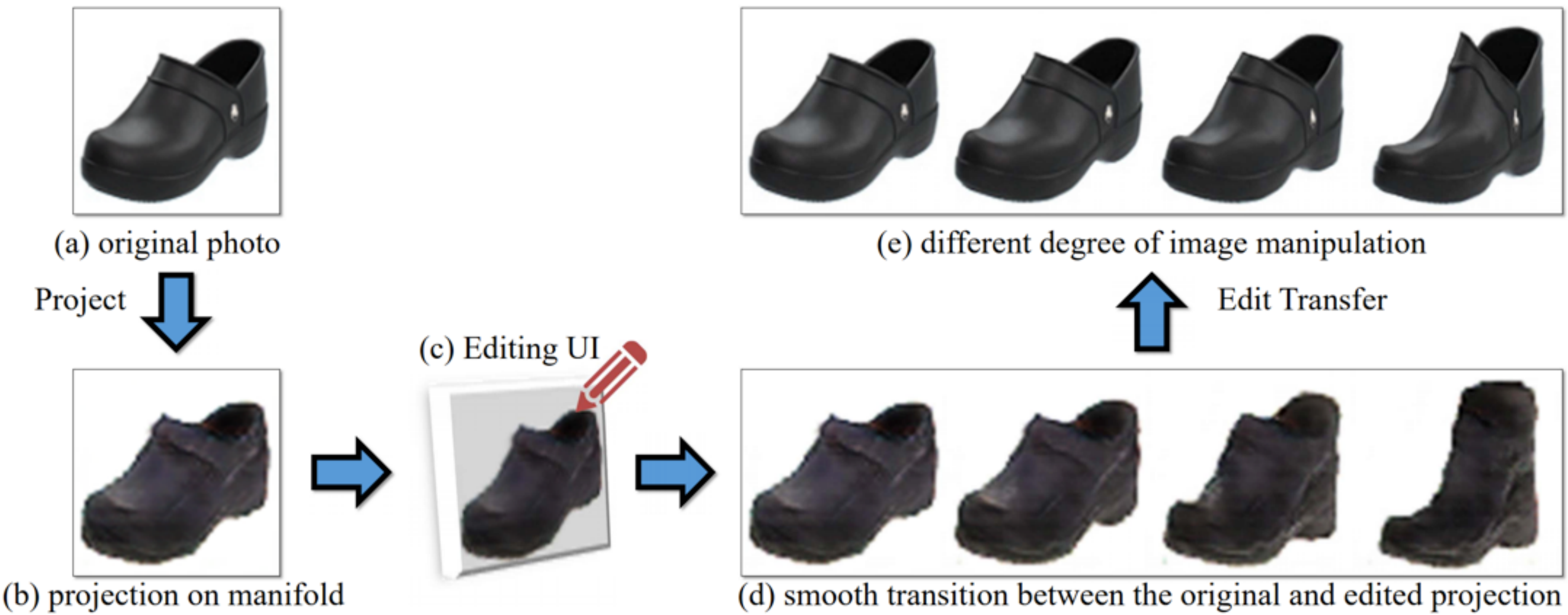


Figure 1: Figure sketches how minibatch discrimination works. Features $\mathbf{f}(\boldsymbol{x}_i)$ from sample $\boldsymbol{x}_i$ are multiplied through a tensor T, and cross-sample distance is computed.

(Salimans et al., 2016)

Solution 1.1: Partial Guidance

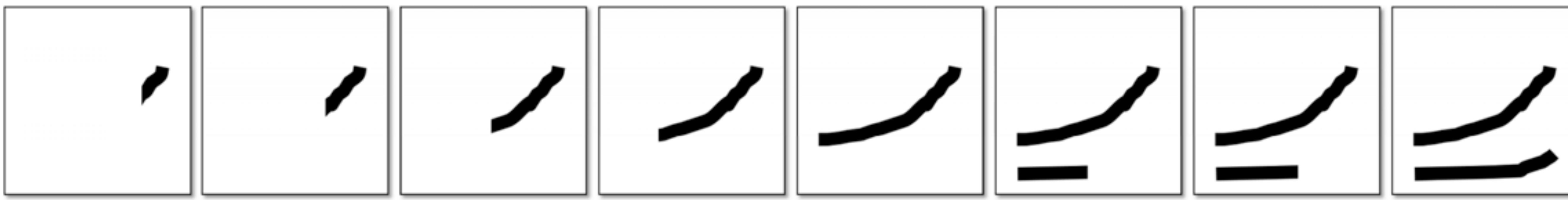
- iGAN/GVM (Zhu et al., 2016)



(Zhu et al., 2016)

Solution 1.1: Partial Guidance

- iGAN/GVM (Zhu et al., 2016)



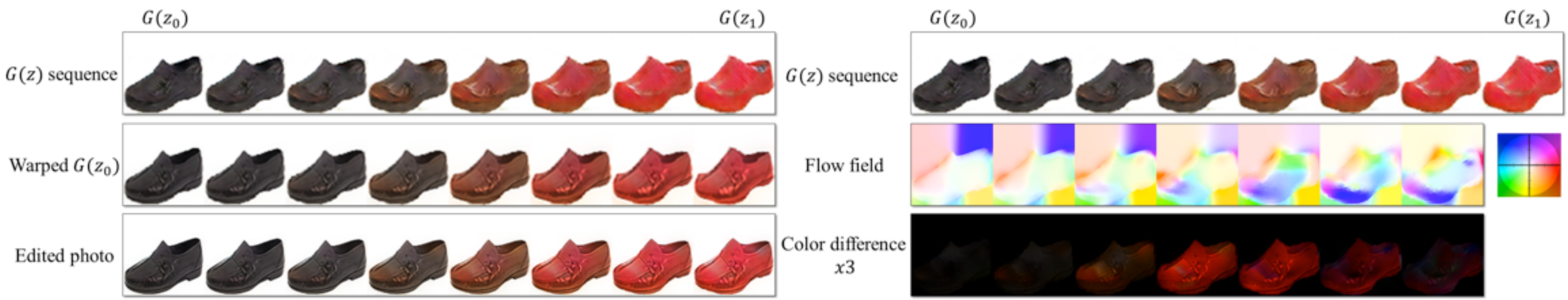
(a) User constraints v_g at different update steps

$G(z_0)$ (b) Updated images according to user edits $G(z_1)$

(c) Linear interpolation between $G(z_0)$ and $G(z_1)$

Solution 1.1: Partial Guidance

- iGAN/GVM (Zhu et al., 2016)



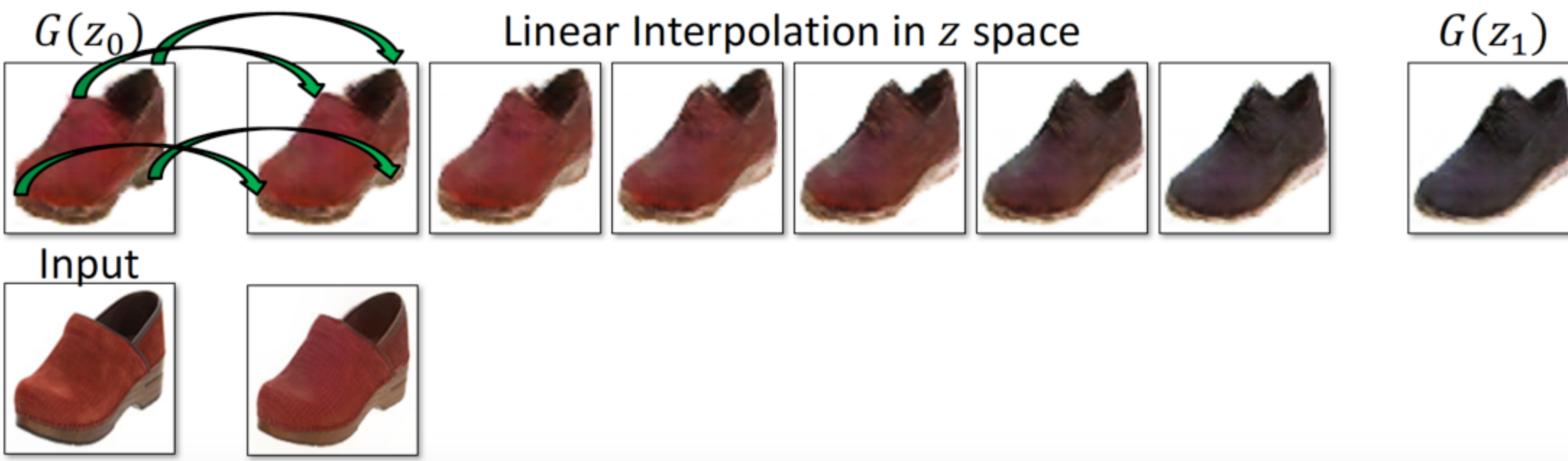
(Zhu et al., 2016)

Solution 1.1: Partial Guidance

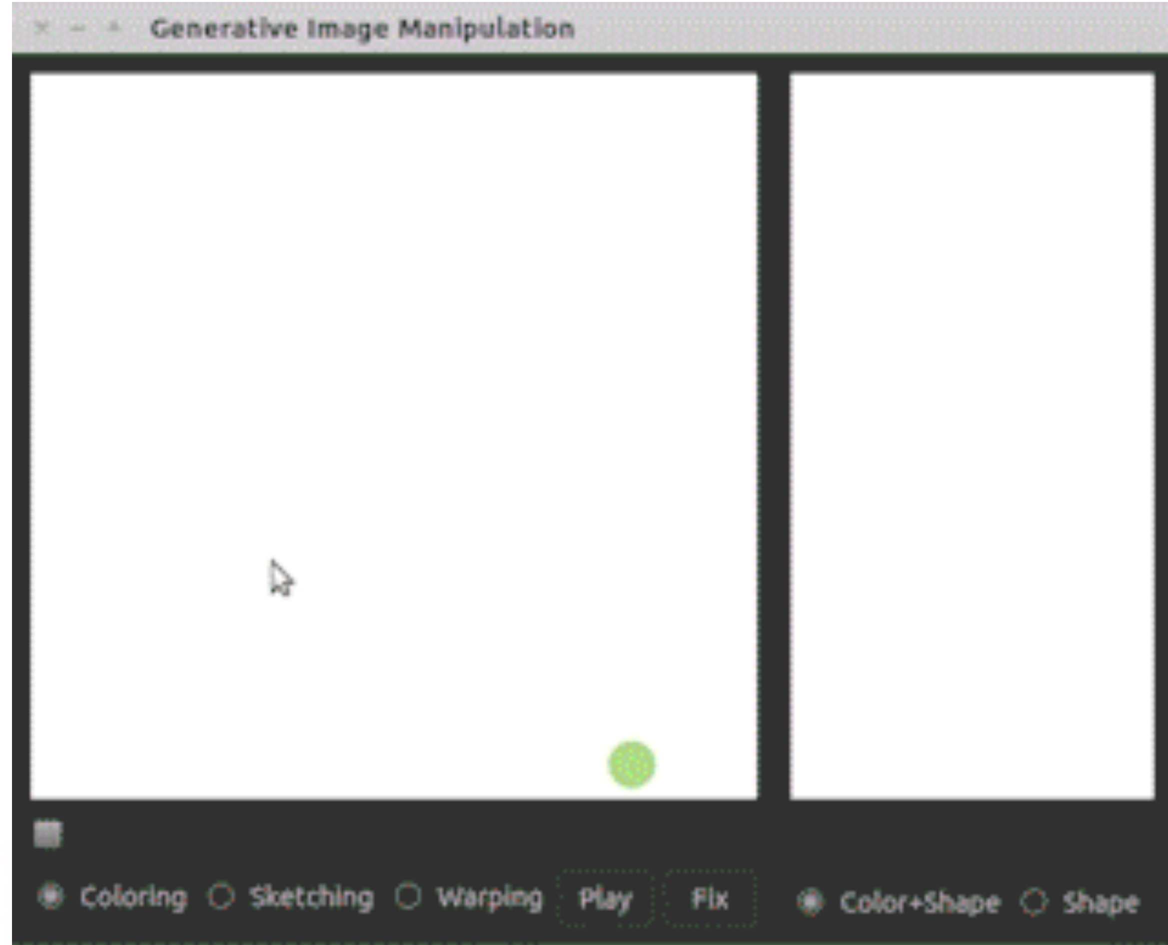
- iGAN/GVM (Zhu et al., 2016)

Motion (u, v)+ Color ($A_{3\times4}$): estimate per-pixel geometric and color variation

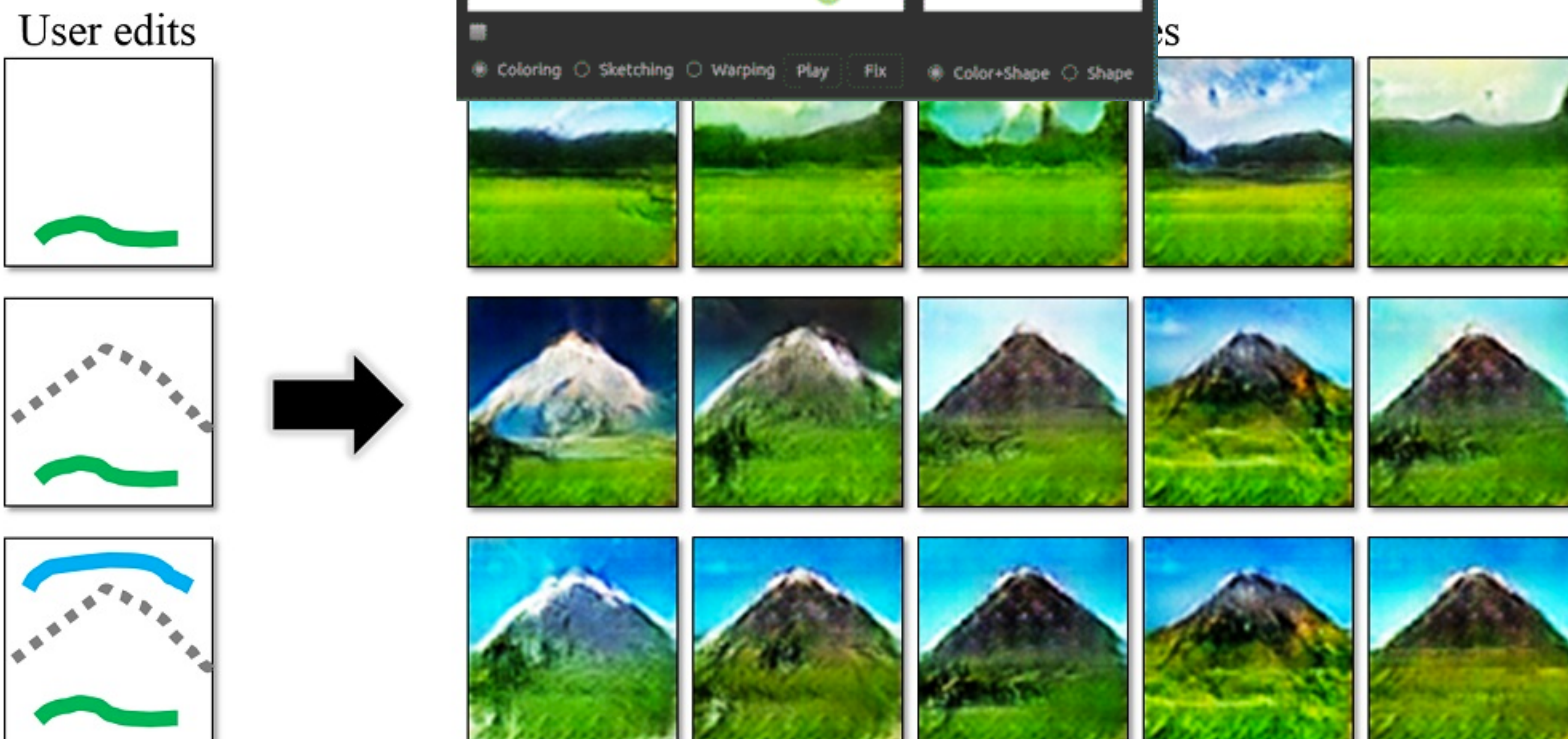
$$\iint \underbrace{\|I(x,y,t) - \boxed{A}\cdot I(x+\boxed{u}, y+\boxed{v}, t+1)\|^2}_{\text{data term}} + \underbrace{\sigma_s(\|\nabla u\|^2+\|\nabla v\|^2)}_{\text{spatial reg}} + \underbrace{\sigma_c\|\nabla A\|^2}_{\text{color reg}} dxdy$$



(Zhu's slides, 2017)

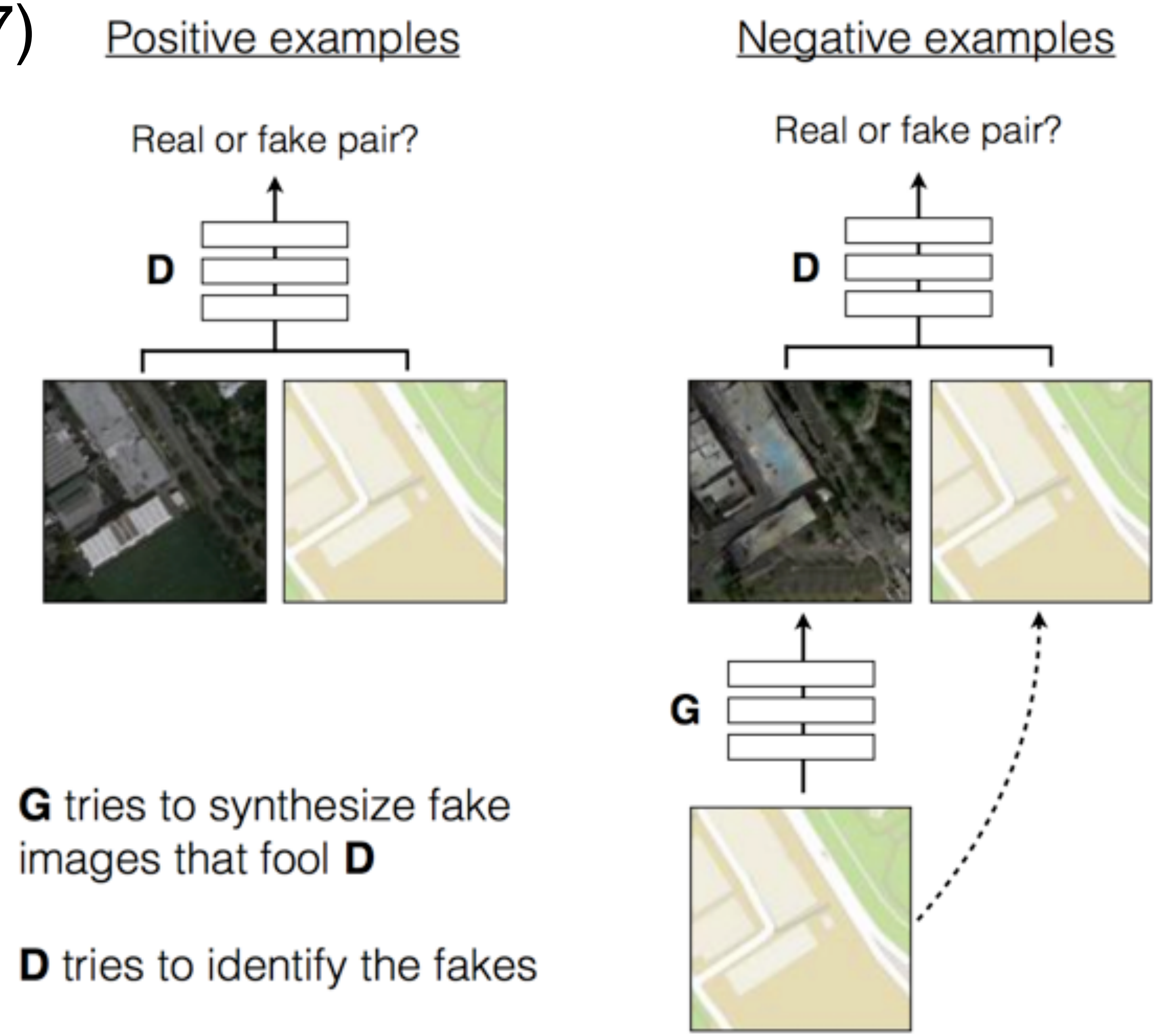


(https://github.com/junyanz/iGAN)

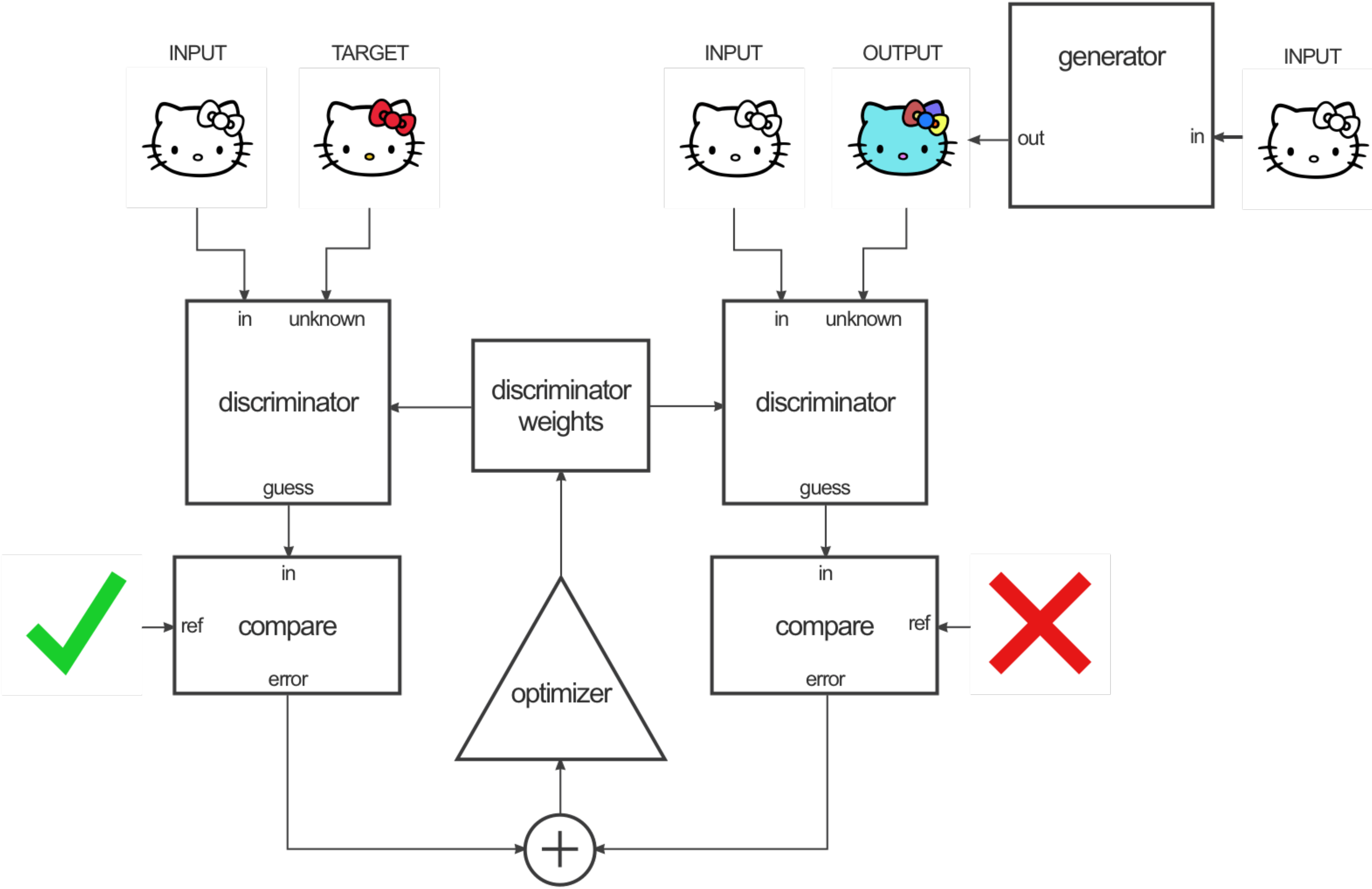


Solution 1.1: Partial Guidance

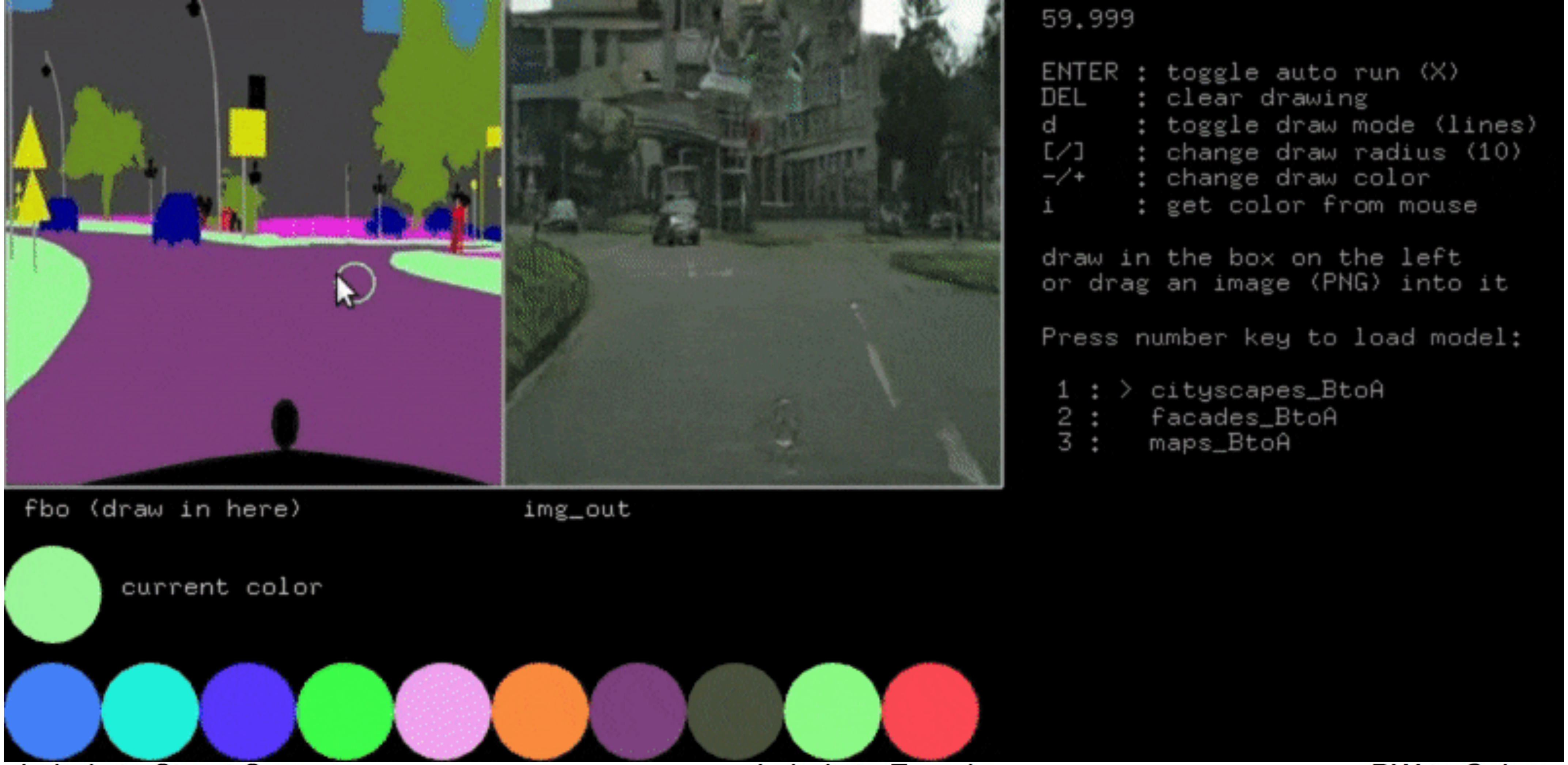
- pix2pix (Isola et al., 2017)

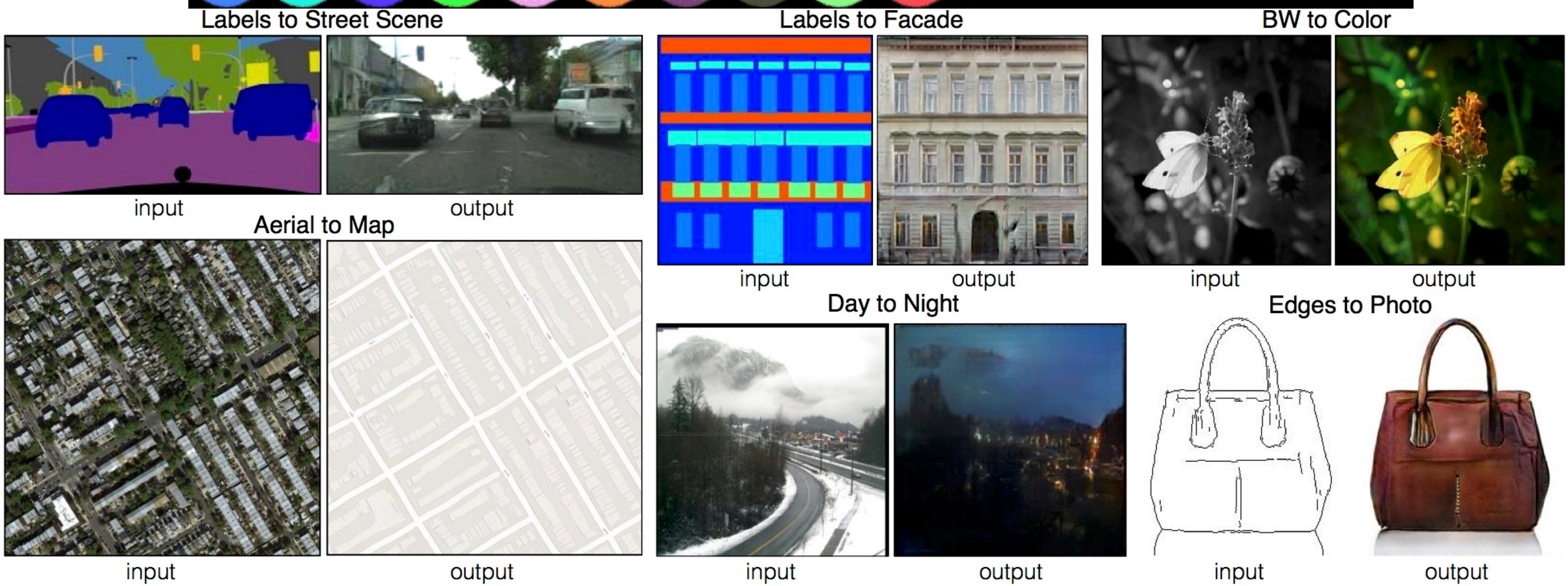


(Isola et al., 2017)



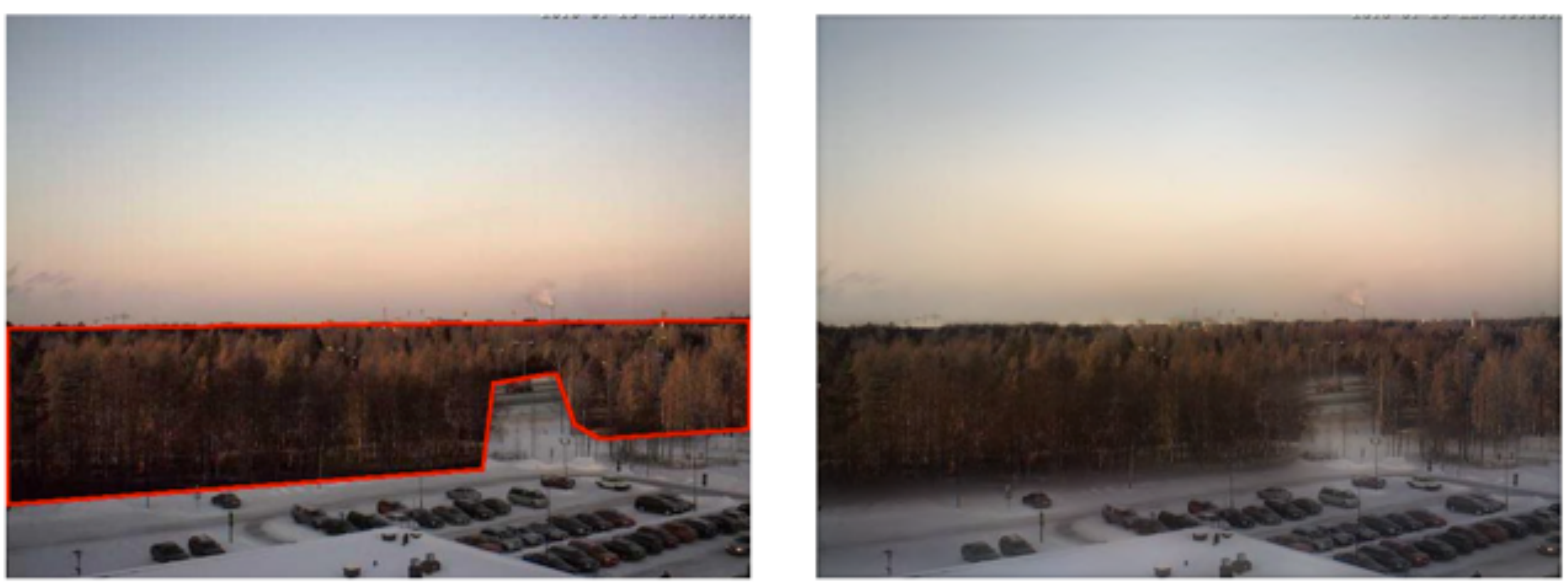
(Christopher Hesse’s blog)


59.999
ENTER : toggle auto run (X)
DEL : clear drawing
d : toggle draw mode (lines)
[/] : change draw radius (10)
-/+ : change draw color
i : get color from mouse
draw in the box on the left
or drag an image (PNG) into it
Press number key to load model:
1 : > cityscapes_BtoA
2 : facades_BtoA
3 : maps_BtoA
fbo (draw in here)
img_out
current color


Labels to Street Scene
input
output
Aerial to Map
input
output
Labels to Facade
input
output
BW to Color
input
output
Day to Night
input
output
Edges to Photo
input
output

Solution 1.1: Partial Guidance

- GP-GAN (Wu et al., 2017)

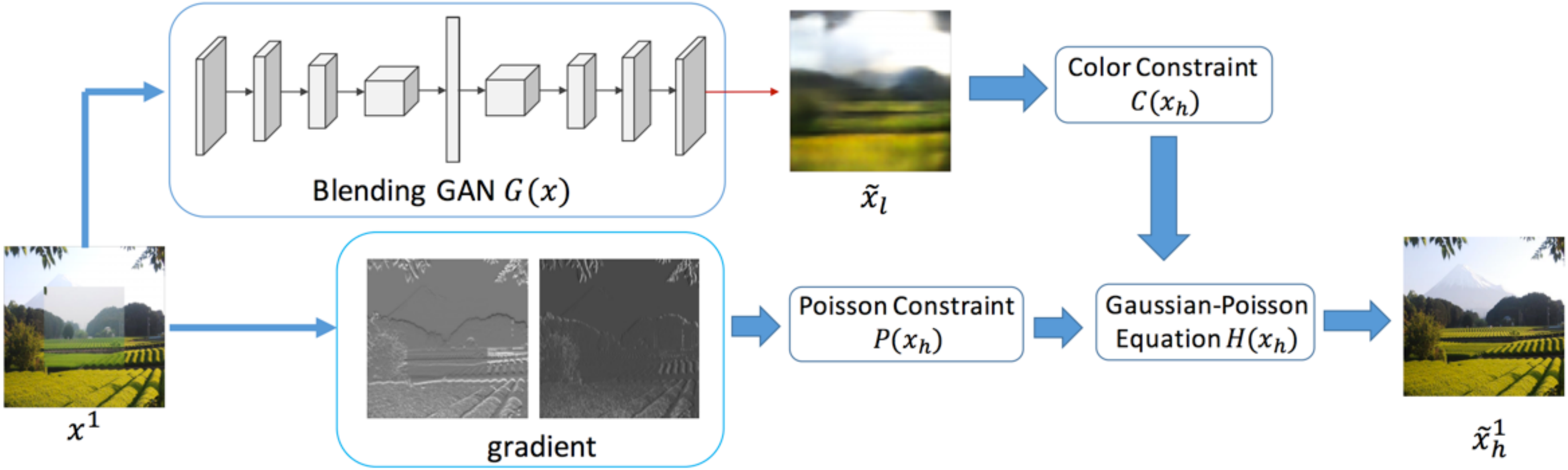


(a) (d)

(https://github.com/wuhuikai/GP-GAN)

Solution 1.1: Partial Guidance

- GP-GAN (Wu et al., 2017)



(https://github.com/wuhuikai/GP-GAN)

Solution 1.2: Fine-grained Guidance

- LAPGAN (Denton et al., 2015)
- Matching-aware Discriminator (Reed et al., 2016)
- StackGAN (Zhang et al., 2016)
- PPGN (Nguyen et al., 2017)

Solution 1.2: Fine-grained Guidance

- LAPGAN (Denton et al., 2015)

$$\min_G \max_D \mathbb{E}_{h,l\sim p_{\text{Data}}(\mathbf{h},\mathbf{l})}[\log D(h,l)] + \mathbb{E}_{z\sim p_{\text{Noise}}(\mathbf{z}),l\sim p_l(\mathbf{l})}[\log(1-D(G(z,l),l))]$$

(Denton et al., 2015)

Solution 1.2: Fine-grained Guidance

- Matching-aware Discriminator (Reed et al., 2016)
    - implicitly separate two sources of error: unrealistic images (for any text), and realistic images of the wrong class that mismatch the conditioning

$$
\begin{aligned}
&\hat{x} \leftarrow G(z, h) \ \{\text{Forward through generator}\} \\
&s_r \leftarrow D(x, h) \ \{\text{real image, right text}\} \\
&s_w \leftarrow D(x, \hat{h}) \ \{\text{real image, wrong text}\} \\
&s_f \leftarrow D(\hat{x}, h) \ \{\text{fake image, right text}\} \\
&\mathcal{L}_D \leftarrow \log(s_r) + (\log(1 - s_w) + \log(1 - s_f))/2
\end{aligned}
$$

(Reed et al., 2016)

Text descriptions (content)

Images (style)

The bird has a **yellow breast** with **grey** features and a small beak.

This is a large **white** bird with **black wings** and a **red head**.

A small bird with a **black head and wings** and features grey wings.

This bird has a **white breast**, brown and white coloring on its head and wings, and a thin pointy beak.

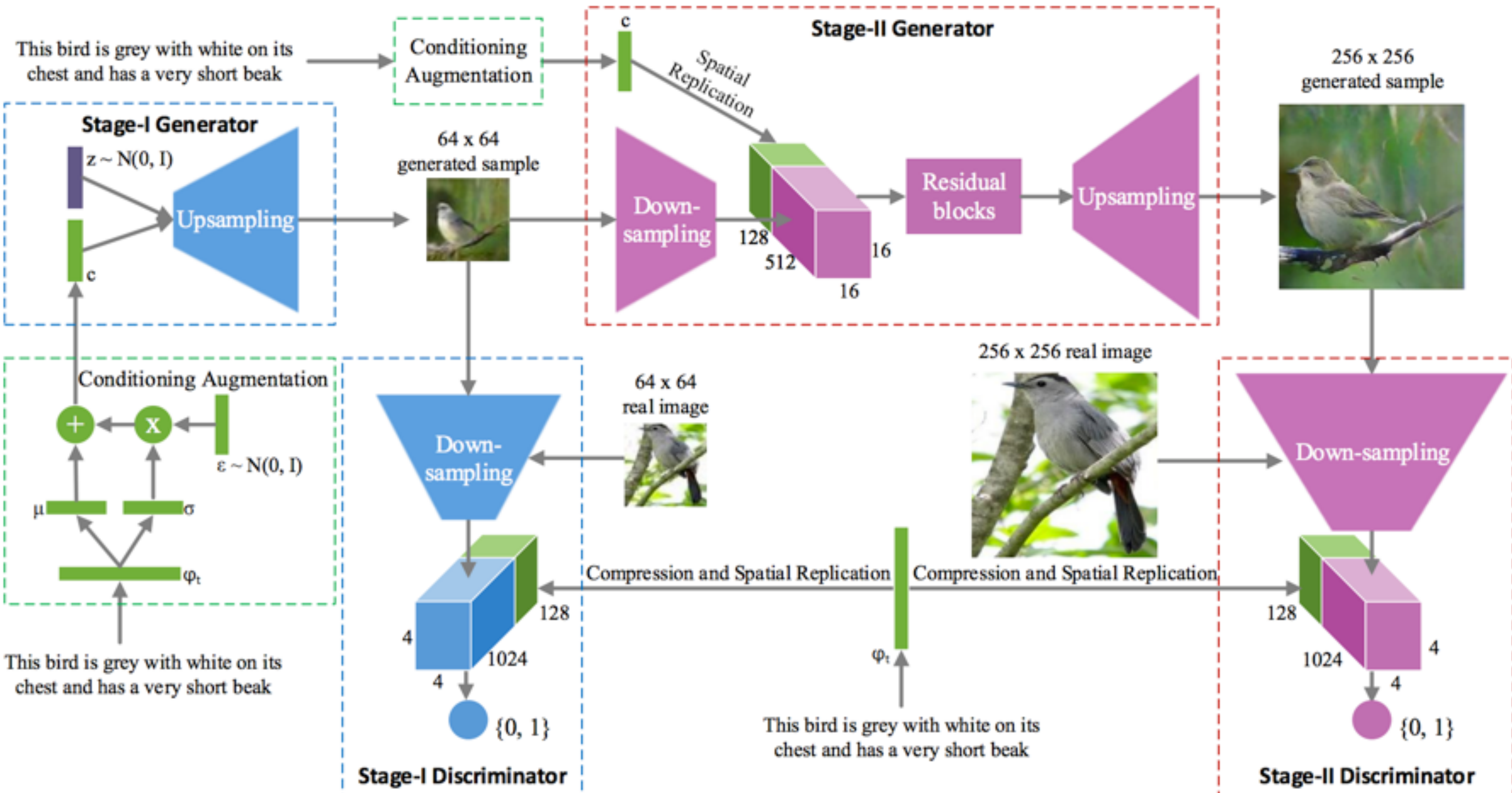
A small bird with **white base** and **black stripes** throughout its belly, head, and feathers.

A small sized bird that has a cream belly and a short pointed bill.

(Reed et al., 2016)

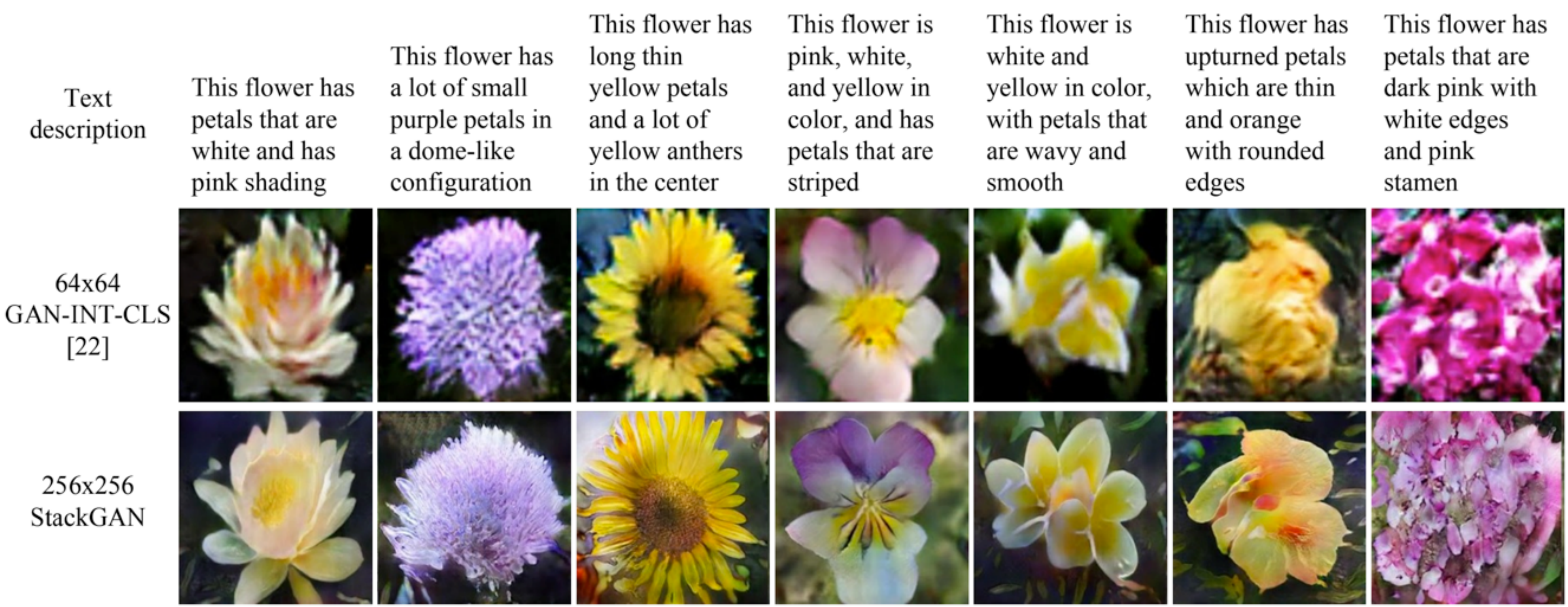
Solution 1.2: Fine-grained Guidance

- StackGAN (Zhang et al., 2016)



Solution 1.2: Fine-grained Guidance

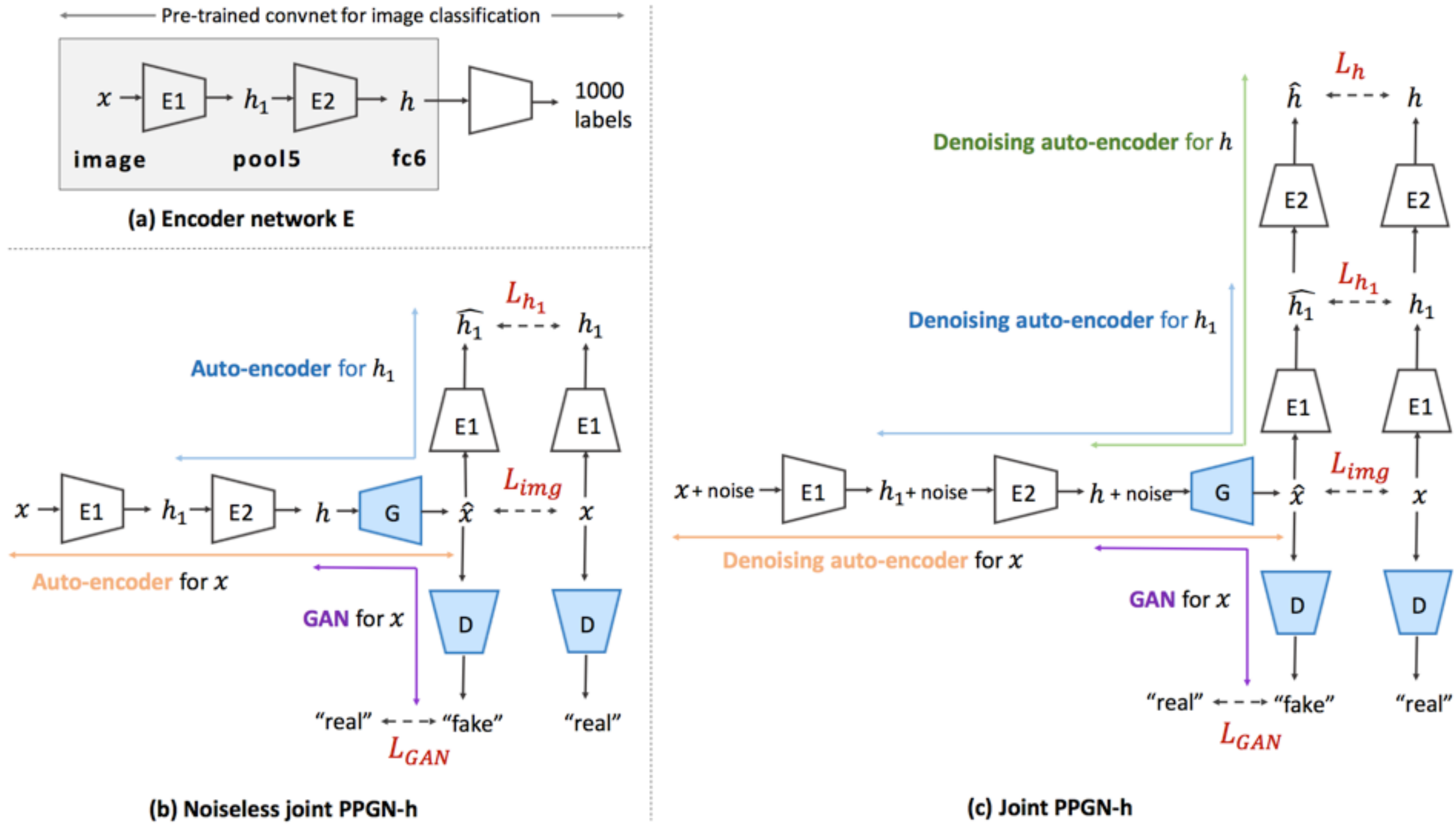
- StackGAN (Zhang et al., 2016)



Solution 1.2: Fine-grained Guidance

- PPGN (Nguyen et al., 2017)

(Nguyen et al., 2017)



(Nguyen et al., 2017)

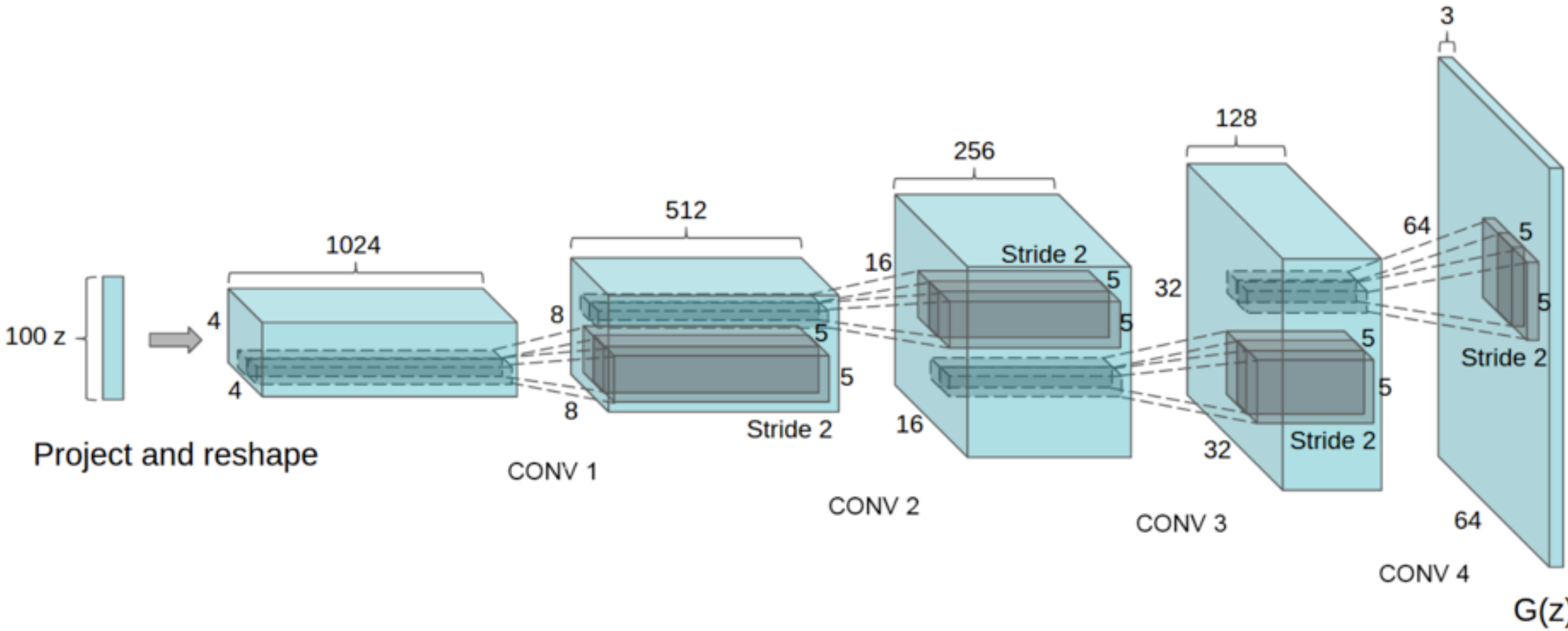
(Nguyen et al., 2017)

Solution 1.3: Special Architecture

- DCGAN (Radford et al., 2016)
- pix2pix (Isola et al., 2017)
- GP-GAN (Wu et al., 2017)

Solution 1.3: Special Architecture

- DCGAN (Radford et al., 2016)

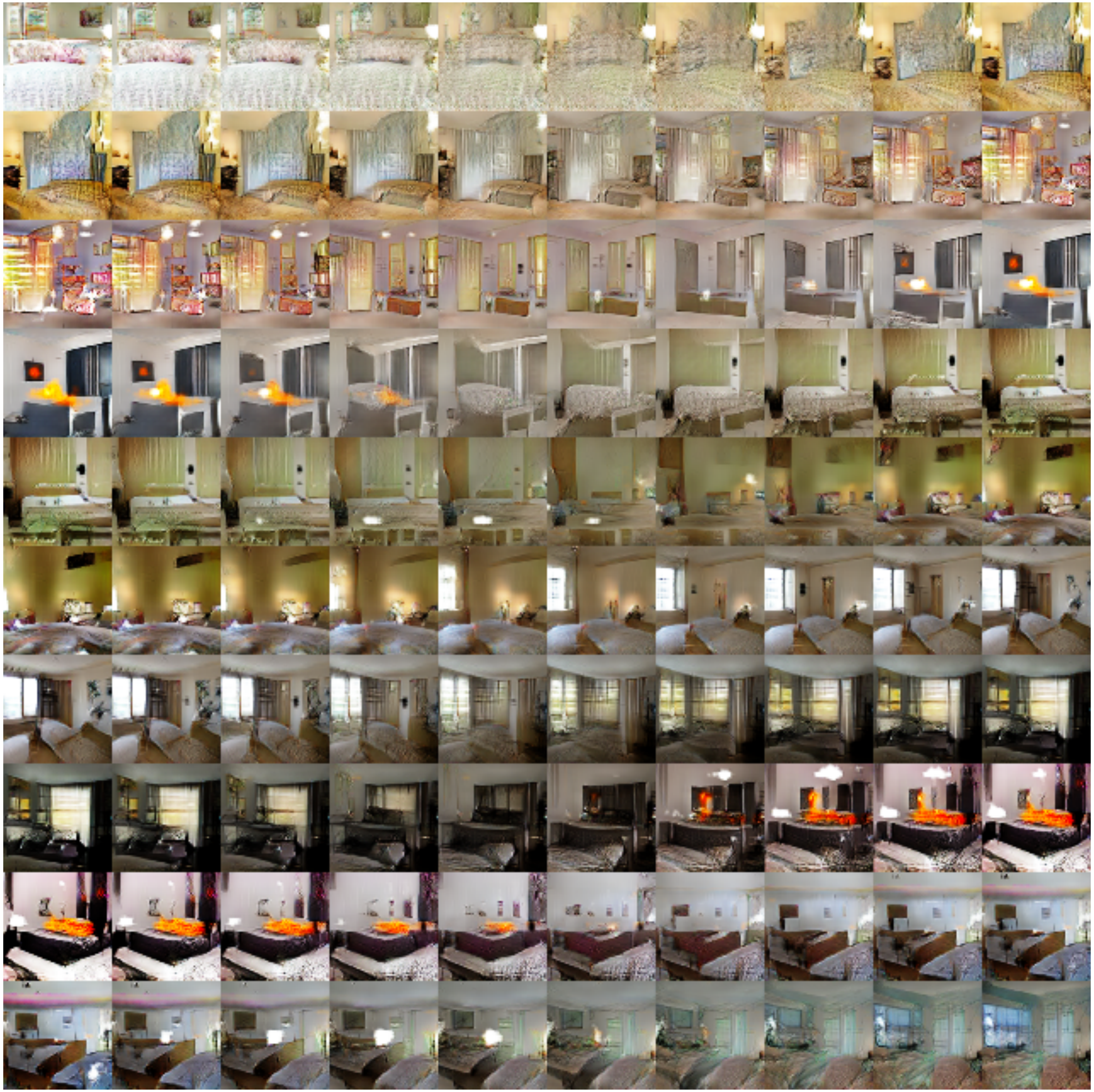


(Radford et al., 2016)

Solution 1.3: Special Architecture

- DCGAN (Radford et al., 2016)
  - Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator)
  - Use batchnorm in both the generator and the discriminator
  - Remove fully connected hidden layers
  - Use ReLU activation in generator for all layers except for the output, which uses Tanh; Use LeakyReLU activation in the discriminator for all layers

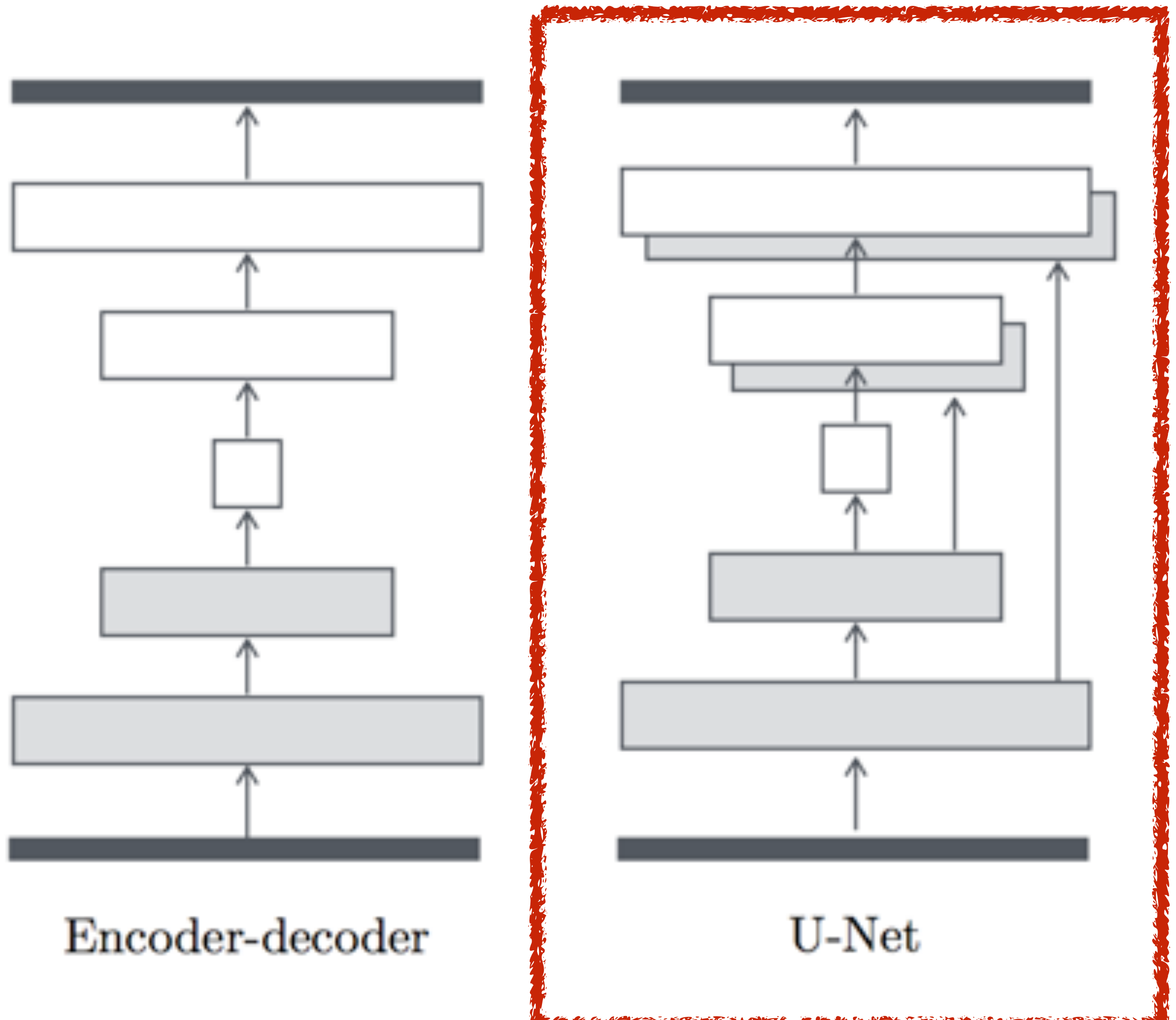
(Radford et al., 2016)



(Radford et al., 2016)

Solution 1.3: Special Architecture

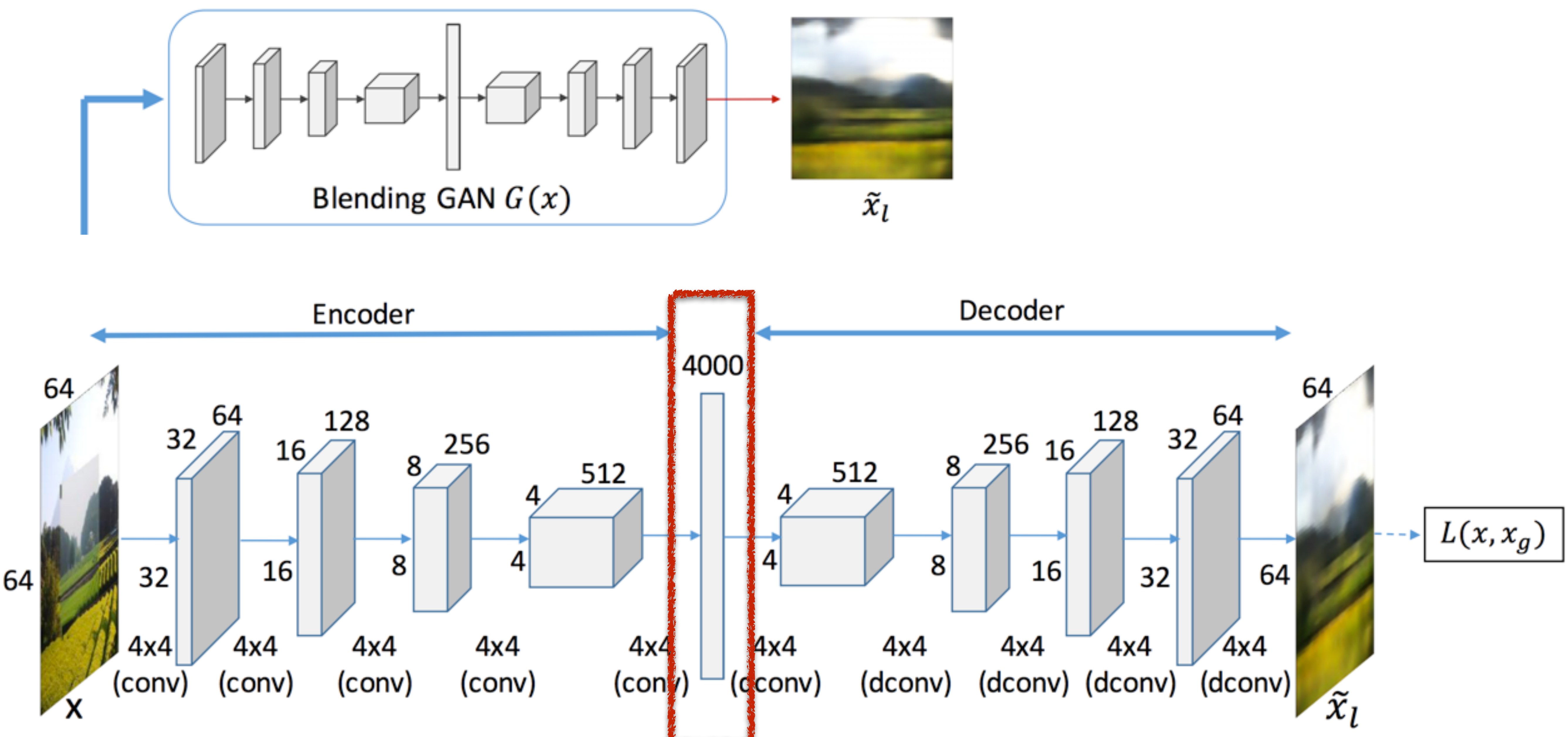
- pix2pix (Isola et al., 2017)



(Isola et al., 2017)

Solution 1.3: Special Architecture

- GP-GAN (Wu et al., 2017)

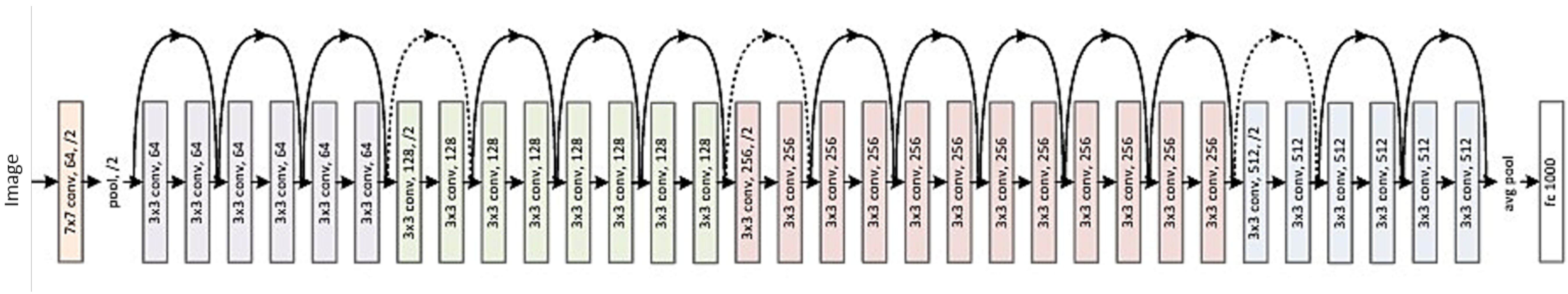


Difficulty 1

Tackled!

- The gradient issues existed in deep neural networks

- The deeper, the more difficult



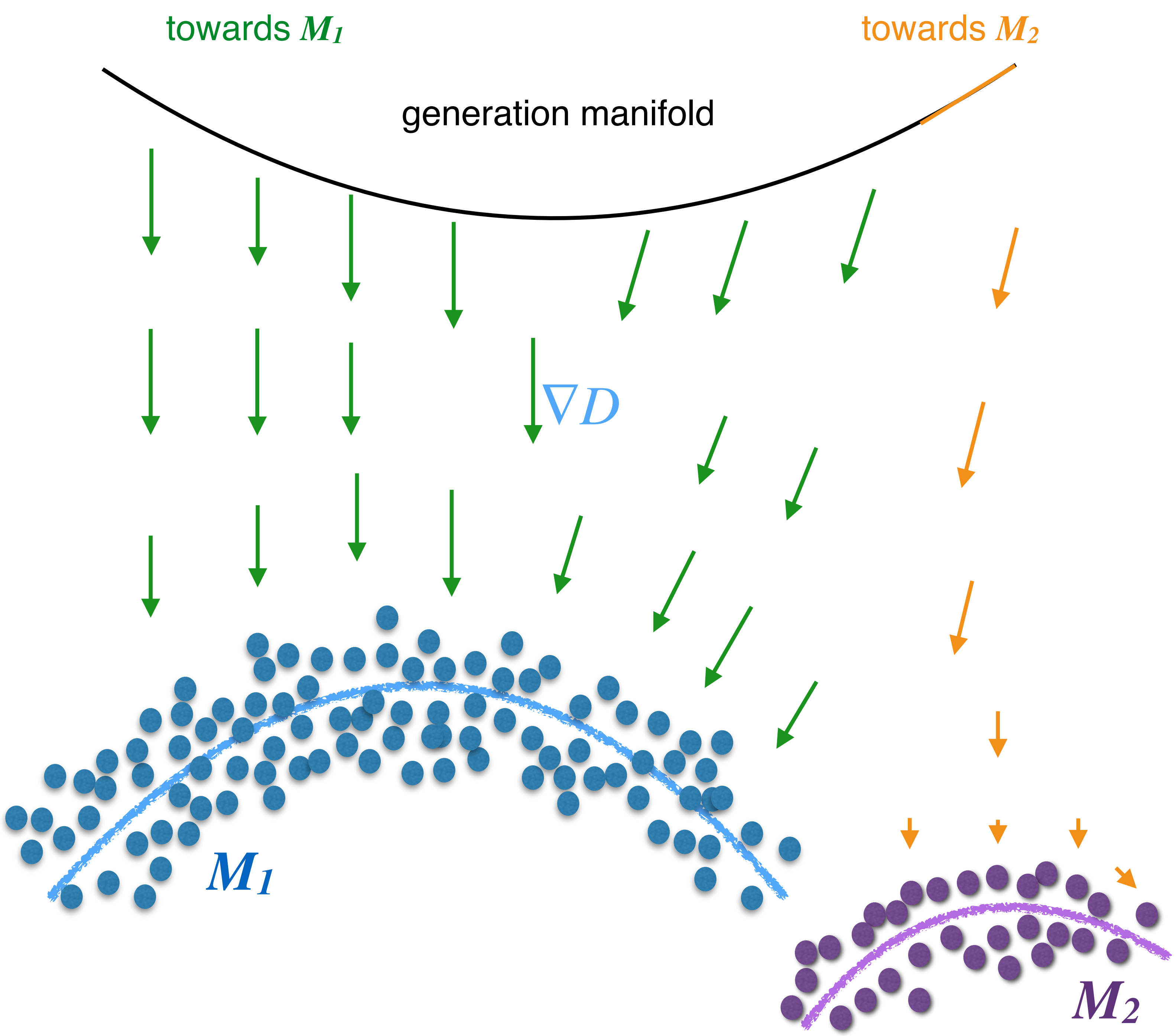
Content

- Generative Adversarial Networks
  - Basics and Attractiveness
  - Difficulties
- Solution 1: Partial and Fine-grained Guidance
- Solution 2: Encoder-incorporated
- Solution 3: Wasserstein Distance

Solution 2: Encoder-incorporated

- Mode Regularized GANs (Che et al., 2017)
- Tackling the gradient vanishing issue and mode missing problem by incorporating an additional encoder ***E*** to:
  - (1) “enforce” P_r and P_g overlap
  - (2) “build a bridge” between *fake data* and *real data*

Mode Missing Problem

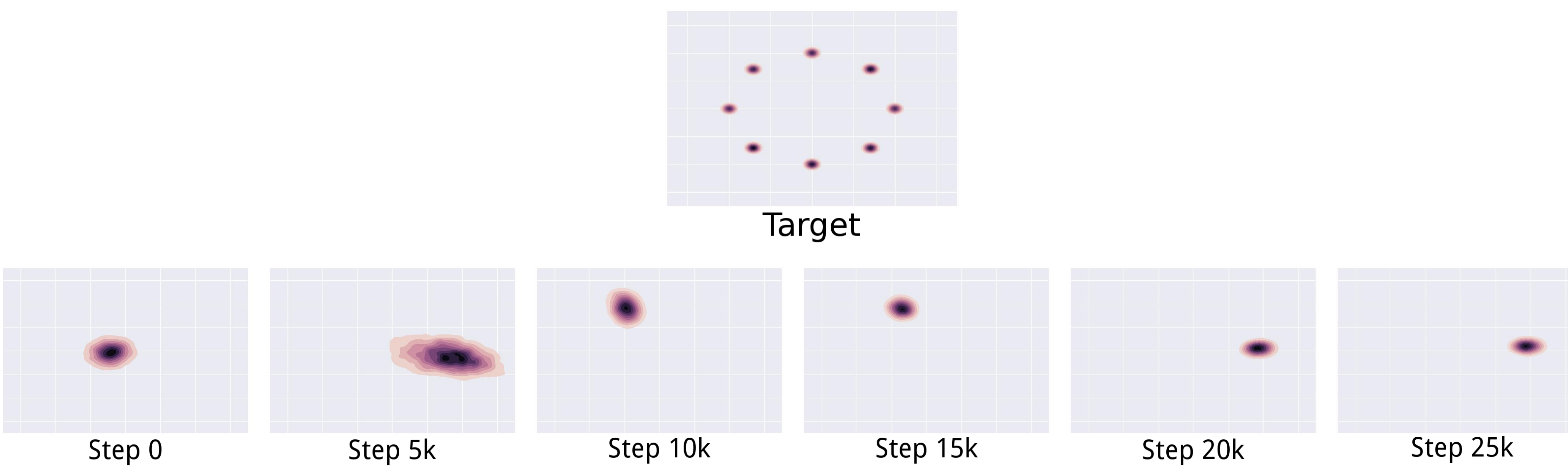


(Che et al., 2017)

Mode Missing Problem

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

- ***D*** in inner loop: convergence to correct distribution
- ***G*** in inner loop: place all mass on most likely point



(Goodfellow's tutorial)

(Metz et al., 2016)

Mode Regularized GANs

- Regularized GANs
  - for encoder ***E***: $\mathbb{E}_{x\sim p_d}[\lambda_1 d(x, G\circ E(x)) + \lambda_2 \log D(G\circ E(x))]$
  - for generator ***G***:

$$-\mathbb{E}_z[\log D(G(z))] + \mathbb{E}_{x\sim p_d}[\lambda_1 d(x, G\circ E(x)) + \lambda_2 \log D(G\circ E(x))]$$

  - for discriminator ***D***: same as vanilla GAN

Mode Regularized GANs

- Regularized GANs
  - for encoder ***E***: $\mathbb{E}_{x\sim p_d}[\lambda_1 d(x, G \circ E(x)) + \lambda_2 \log D(G \circ E(x))]$
  - for generator ***G***:

$$-\mathbb{E}_z[\log D(G(z))] + \mathbb{E}_{x\sim p_d}[\lambda_1 d(x, G \circ E(x)) + \lambda_2 \log D(G \circ E(x))]$$

  - for discriminator ***D***: same as vanilla GAN

- But it still suffers from gradient vanishing!
- because ***D*** is still comparing between real data and fake data

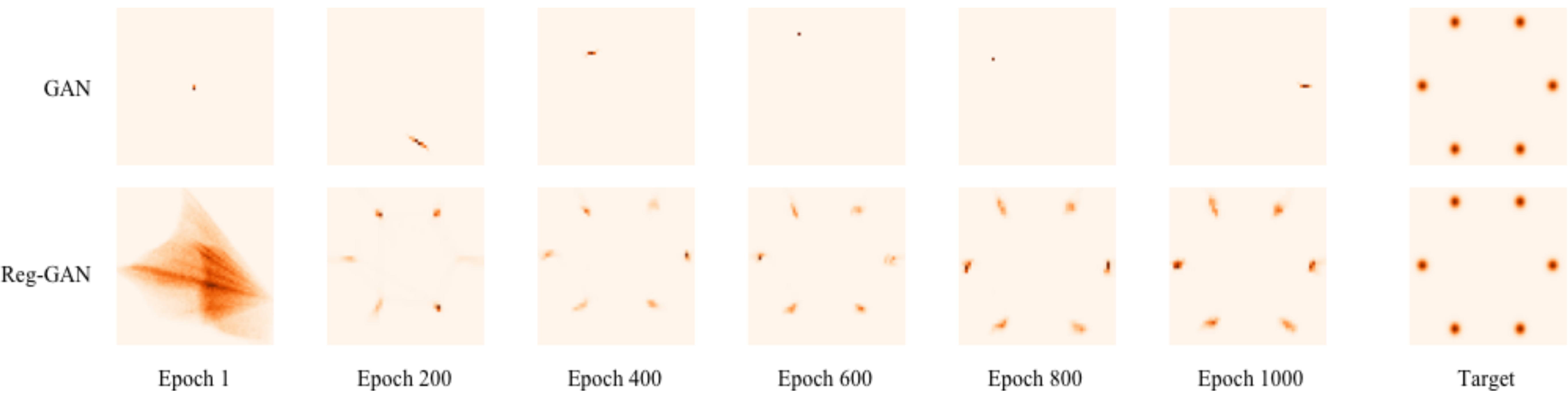
Mode Regularized GANs

- Manifold-Diffusion GANs (MDGAN):
  - Manifold-step:
    - Try to match the generation manifold and the real data manifold
  - Diffusion-step:
    - Try to distribute the probability mass on the generation manifold fairly according to the real data distribution

Mode Regularized GANs

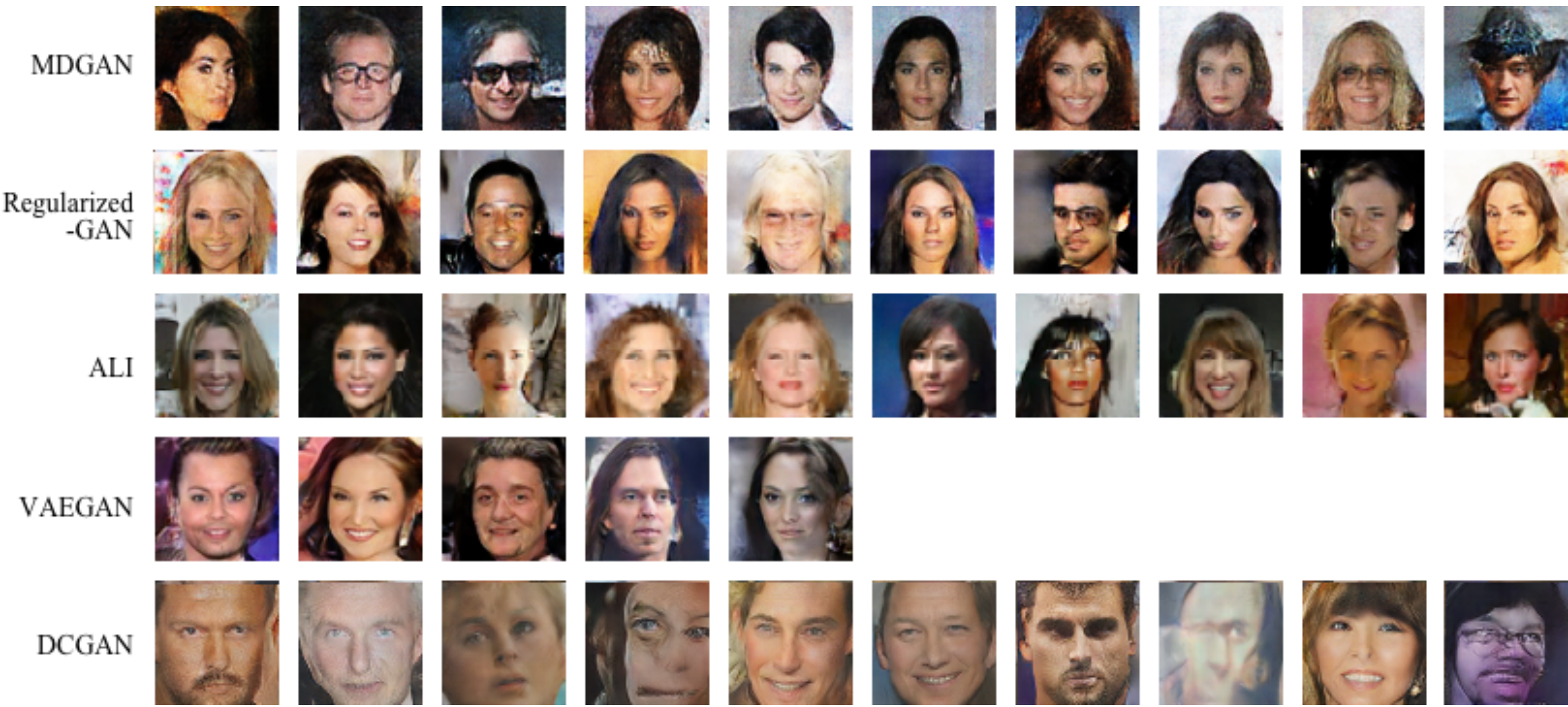
- Manifold-Diffusion GANs (MDGAN):
  - Manifold-step:
    - Try to match the generation manifold and the real data manifold
  - Diffusion-step:
    - Try to distribute the probability mass on the generation manifold fairly according to the real data distribution
- ***D*** is firstly comparing between real data and the encoded data — much harder!

Mode Regularized GANs



(Che et al., 2017)

Mode Regularized GANs



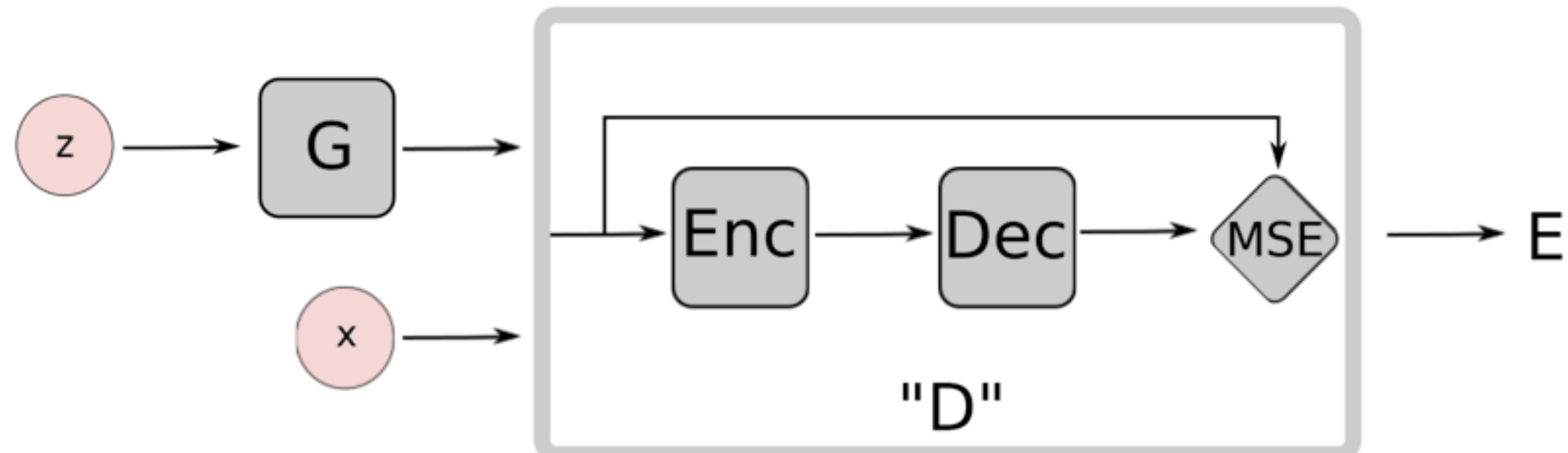
(Che et al., 2017)

Solution 2: Encoder-incorporated

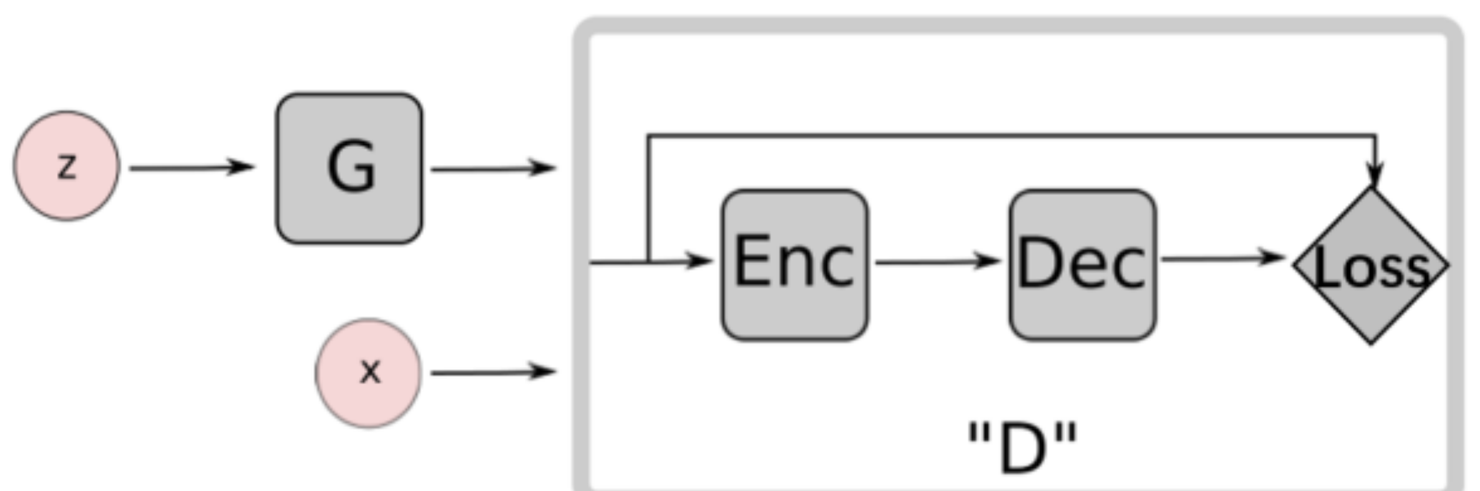
- Mode Regularized GANs (Che et al., 2017)
- Energy-based GANs (Zhao et al., 2017)
- Boundary Equilibrium GANs (Berthelot et al., 2017)
- etc.

Solution 2: Encoder-incorporated

- Energy-based GANs (Zhao et al., 2017)



- Boundary Equilibrium GANs (Berthelot et al., 2017)



(Zhao et al., 2017)
(Berthelot et al., 2017)

Solution 2: **Noisy Input*

- Add noise to input (both real data and fake data) before passing into D (Arjovsky & Bottou, 2017, *Theorem 3.2*)

- Add noise to layers in D and G (Zhao et al., 2017)

- Instance Noise (Sønderby et al., 2017)

- All these are indeed "enforcing" P_r and P_g to overlap

Review Mode Missing Problem

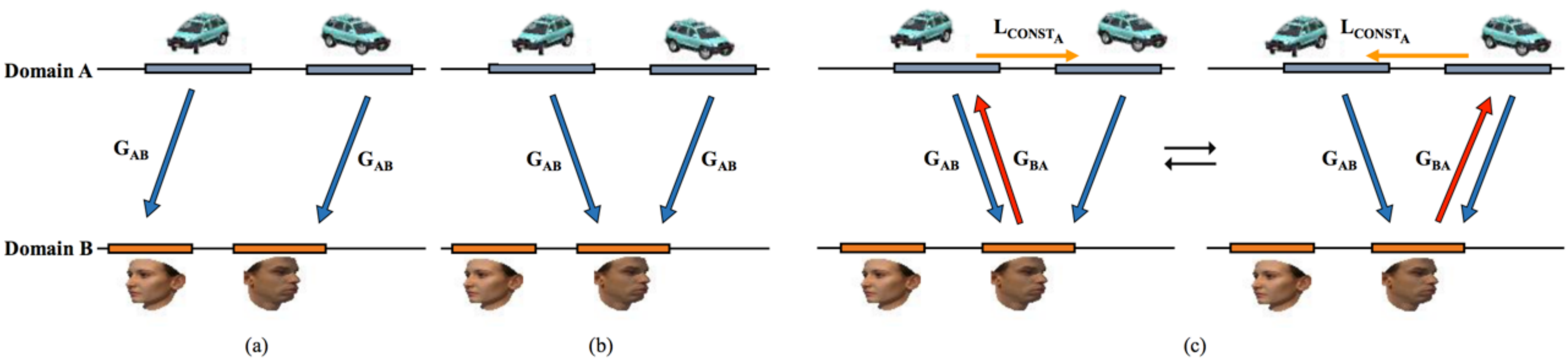
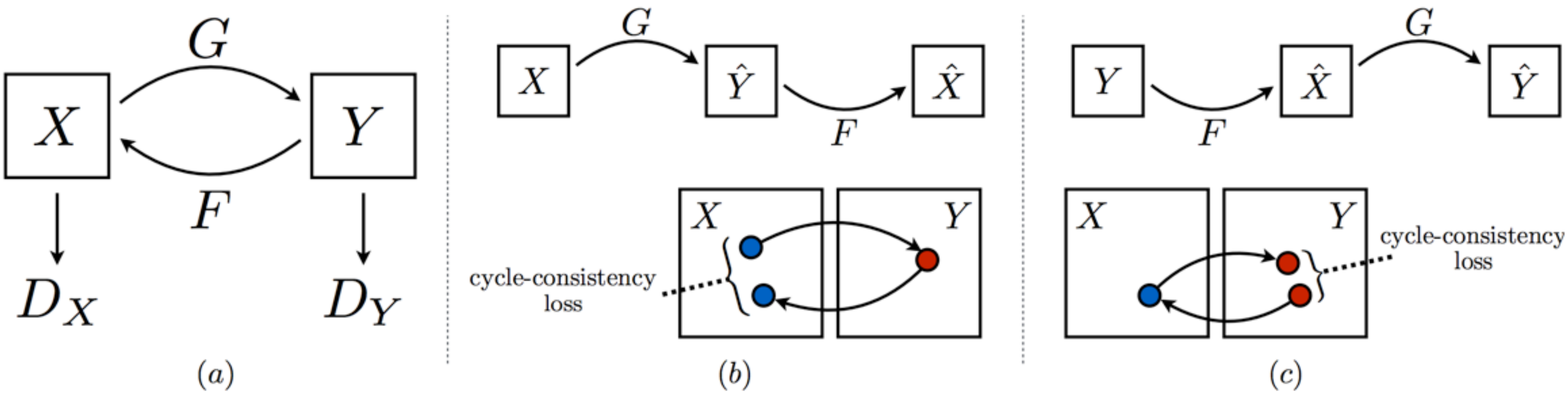


Figure 3. Illustration of our models on simplified one dimensional domains. (a) ideal mapping from domain A to domain B in which the two domain A modes map to two different domain B modes, (b) GAN model failure case, (c) GAN with reconstruction model failure case.

(Kim et al., 2017)

Solution 2: Encoders-incorporated

- CycleGAN (Zhu et al., 2017)



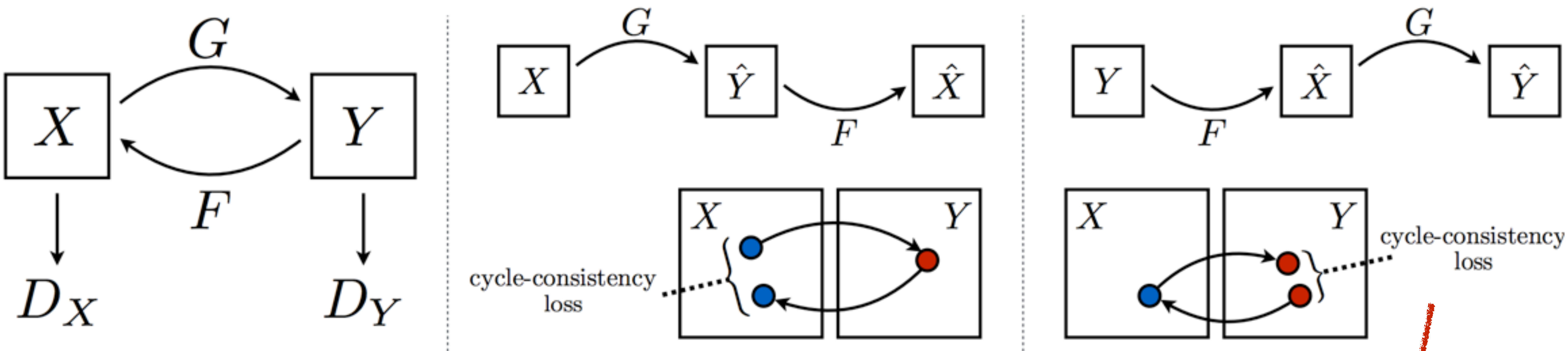
- DiscoGAN (Kim et al., 2017)

- DualGAN (Yi et al., 2017)

(Zhu et al., 2017)

Solution 2: Encoders-incorporated

- CycleGAN (Zhu et al., 2017)



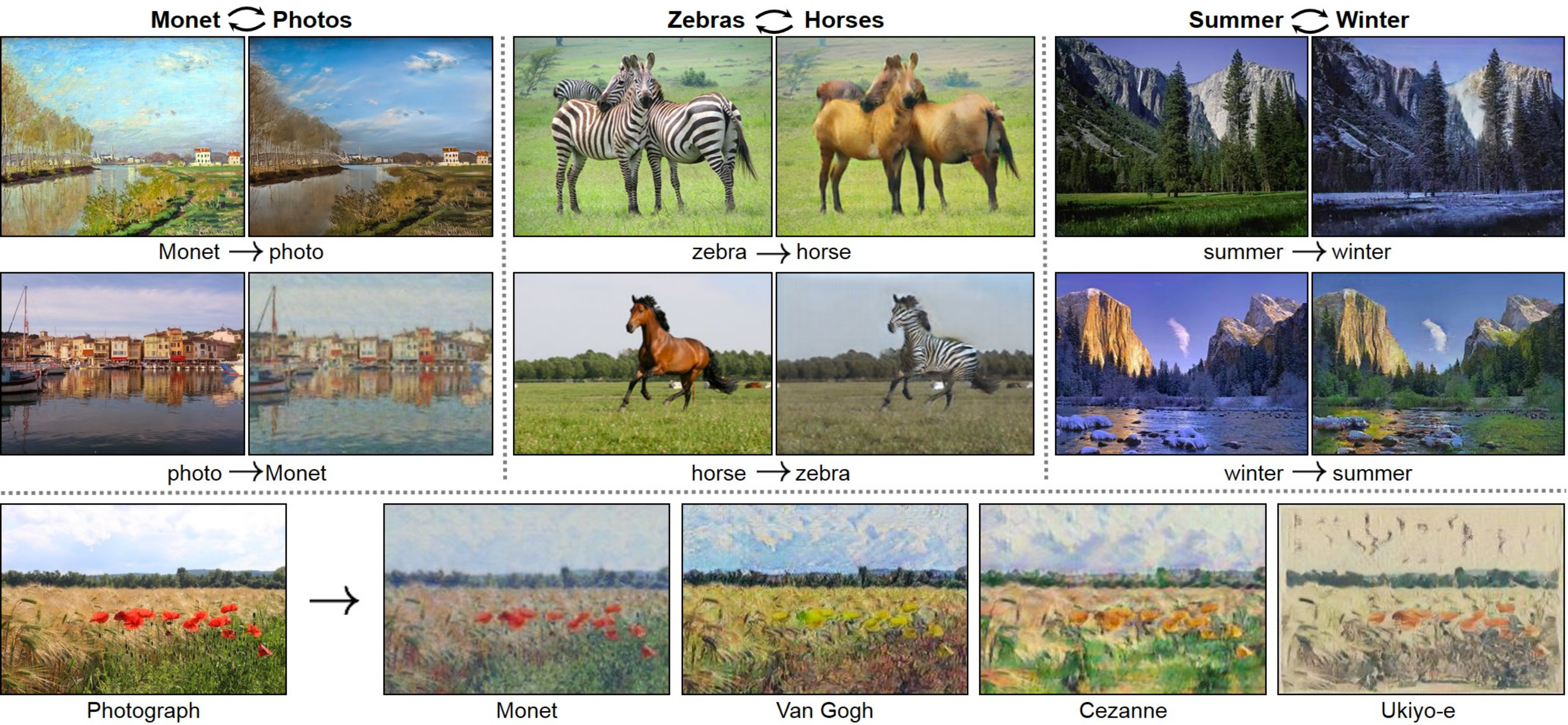
(a) (b) (c)

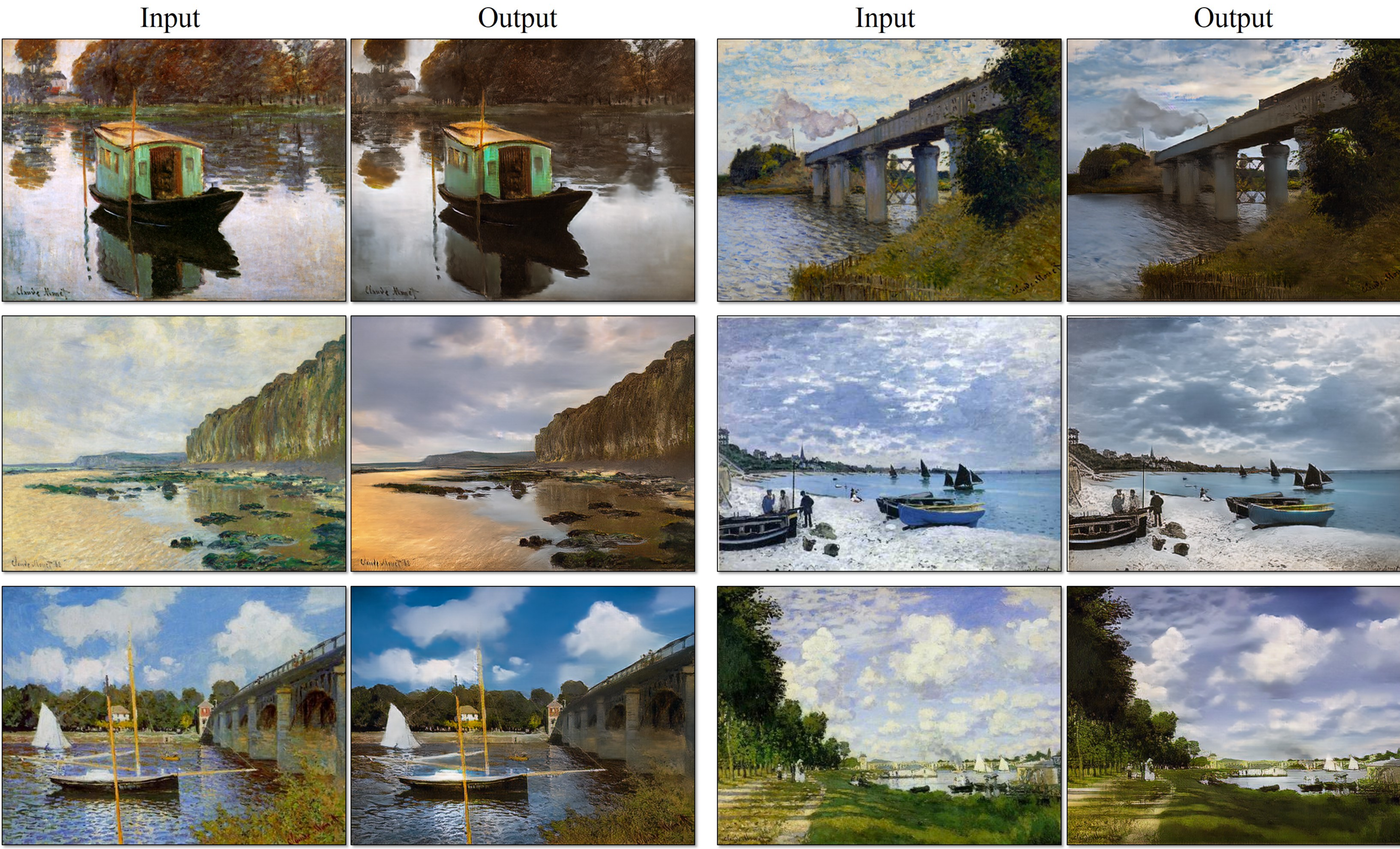
$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) + \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) + \lambda \mathcal{L}_{\text{cyc}}(G, F),$$

$$\mathcal{L}_{\text{cyc}}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1].$$

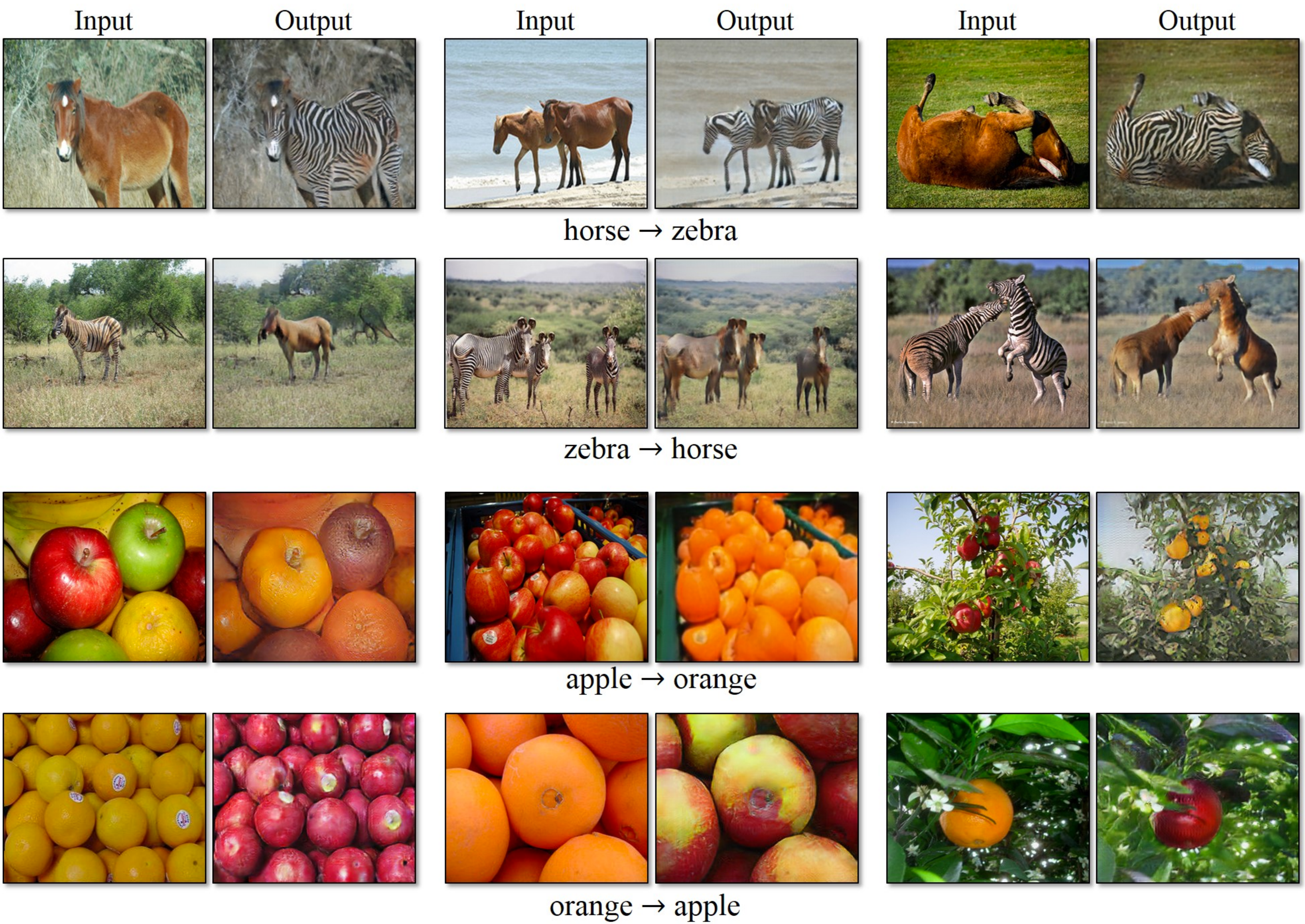
(Zhu et al., 2017)

(https://junyanz.github.io/CycleGAN/)




Input
Output
Input
Output


Input
Monet
Van Gogh
Cezanne
Ukiyo-e


Input
Output
Input
Output
Input
Output
horse → zebra
zebra → horse
apple → orange
orange → apple

Solution 2: Encoders-incorporated

- CycleGAN (Zhu et al., 2017)

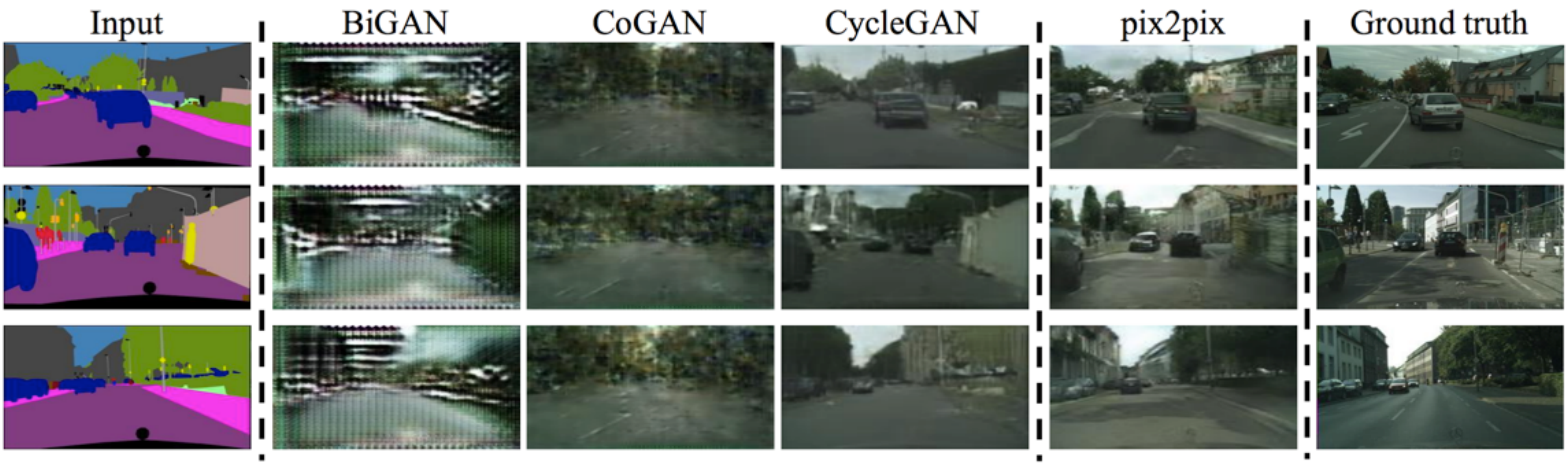
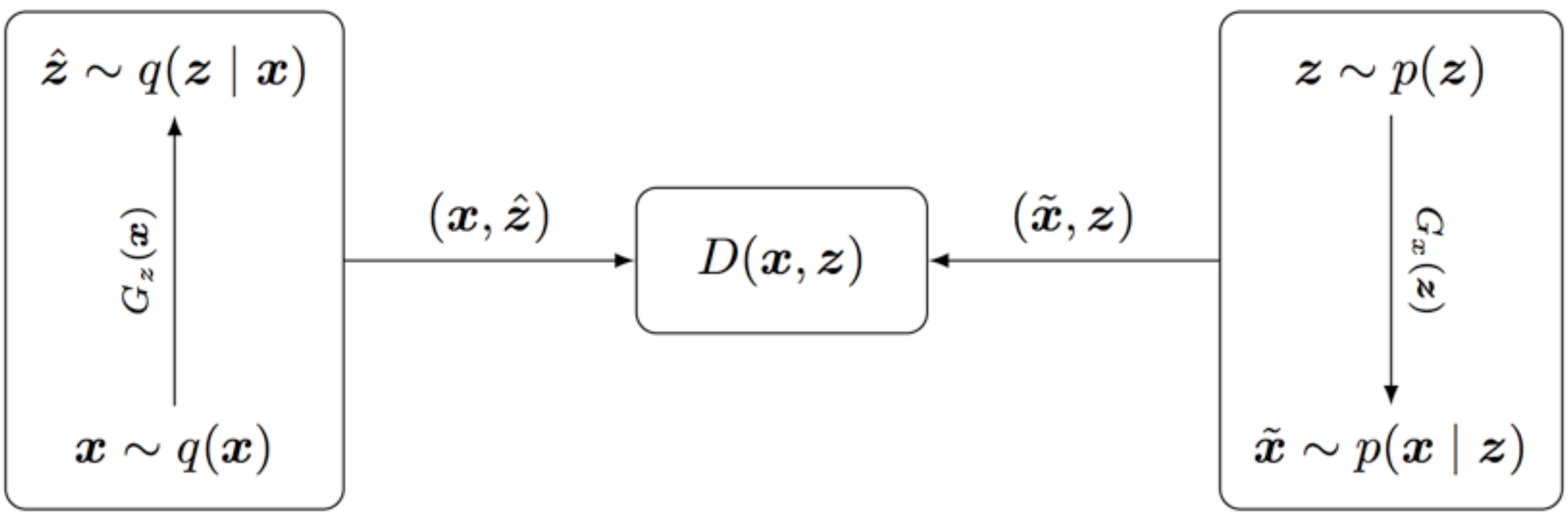


Figure 5: Different methods for mapping labels↔photos trained on cityscapes. From left to right: input, BiGAN [5, 6] CoupledGAN [27], CycleGAN (ours), pix2pix [18] trained on paired data, and ground truth.

(Zhu et al., 2017)

Solution 2: Encoders-incorporated

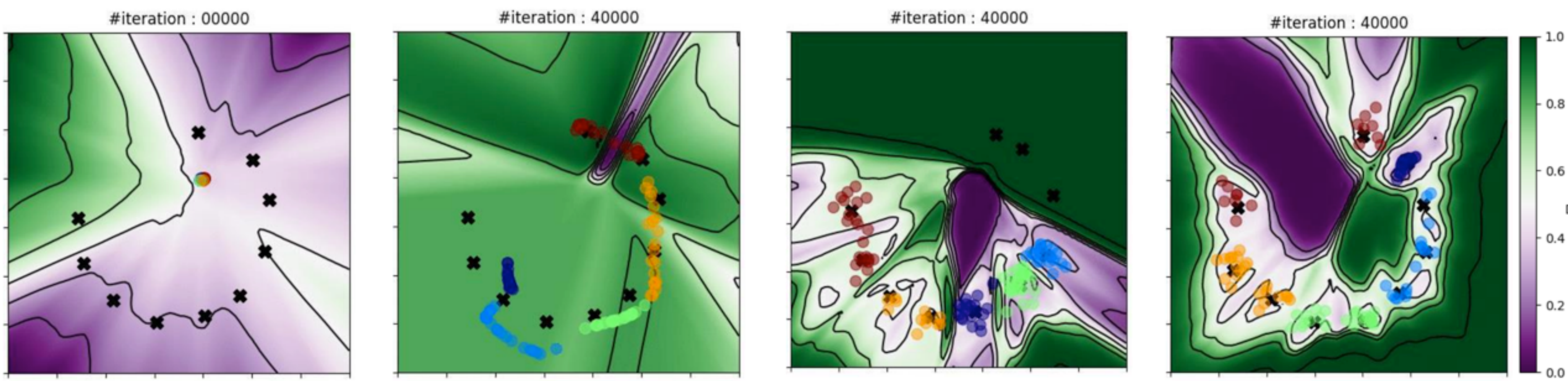
- CycleGAN (Zhu et al., 2017)
- BiGAN (Donahue et al., 2017; Dumoulin et al., 2017)
    - G: Z—> X + F: X—> Z



(Dumoulin et al., 2017)

Solution 2: Encoders-incorporated

- DiscoGAN (Kim et al., 2017)

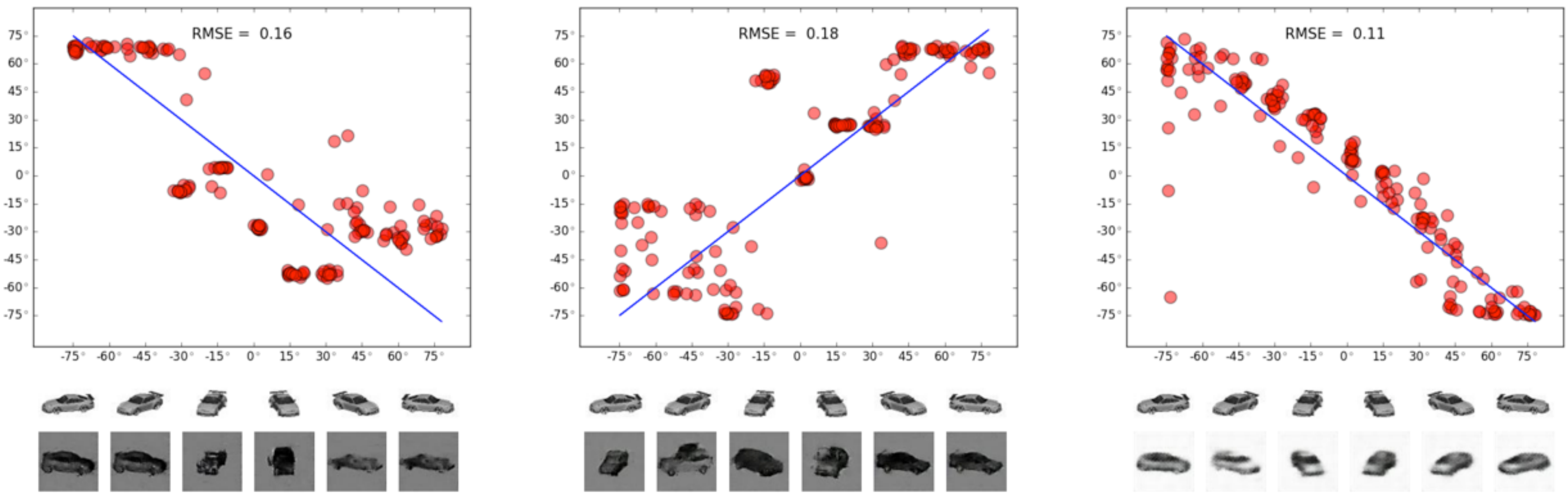


(a) (b) (c) (d)

(Zhu et al., 2017)

Solution 2: Encoders-incorporated

- DiscoGAN (Kim et al., 2017)



(a) (b) (c)

(Zhu et al., 2017)

Difficulty 3

Tackled!

- minimizing the *KL divergence* only is biased:

$$KL(\mathbb{P}_{g_\theta}||\mathbb{P}_r) - 2JSD(\mathbb{P}_{g_\theta}||\mathbb{P}_r)]$$

- because *KL divergence* is asymmetric, and thus it is not equally treated when ***G*** generates an unreal sample and when ***G*** fails to generate real sample

- Therefore, ***G*** will generate too many few-mode (less diverse) but real samples , a safer strategy

Content

- Generative Adversarial Networks
  - Basics and Attractiveness
  - Difficulties
- Solution 1: Partial and Fine-grained Guidance
- Solution 2: Encoder-incorporated
- Solution 3: Wasserstein Distance

Solution 3: Wasserstein Distance

- Wasserstein GANs (Arjovsky et al., 2017)
- Wasserstein-1 Distance (Earth-Mover Distance):

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}\big[\, \|x - y\| \,\big]$$

Solution 3: Wasserstein Distance

- Wasserstein GANs (Arjovsky et al., 2017)
- Wasserstein-1 Distance (Earth-Mover Distance):

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}\big[\,\|x - y\|\,\big]$$

- *Why is it superior to KL and JS divergence?*

Solution 3: Wasserstein Distance

- Wasserstein-1 Distance (Earth-Mover Distance):

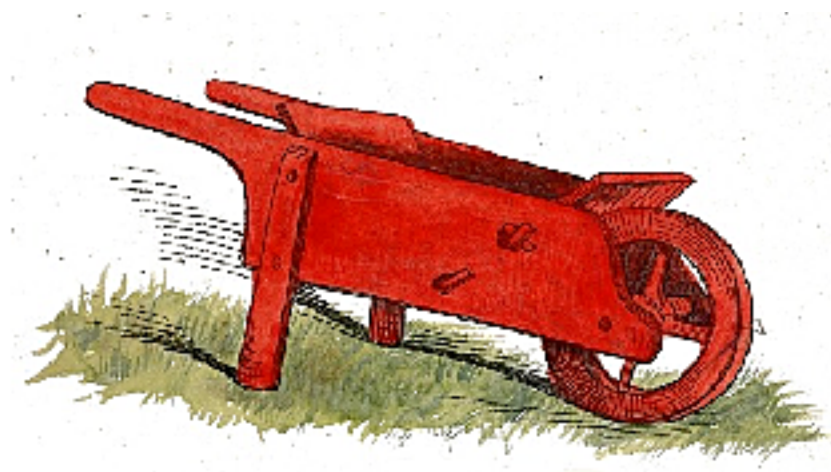
$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma} \big[\|x - y\| \big]$$

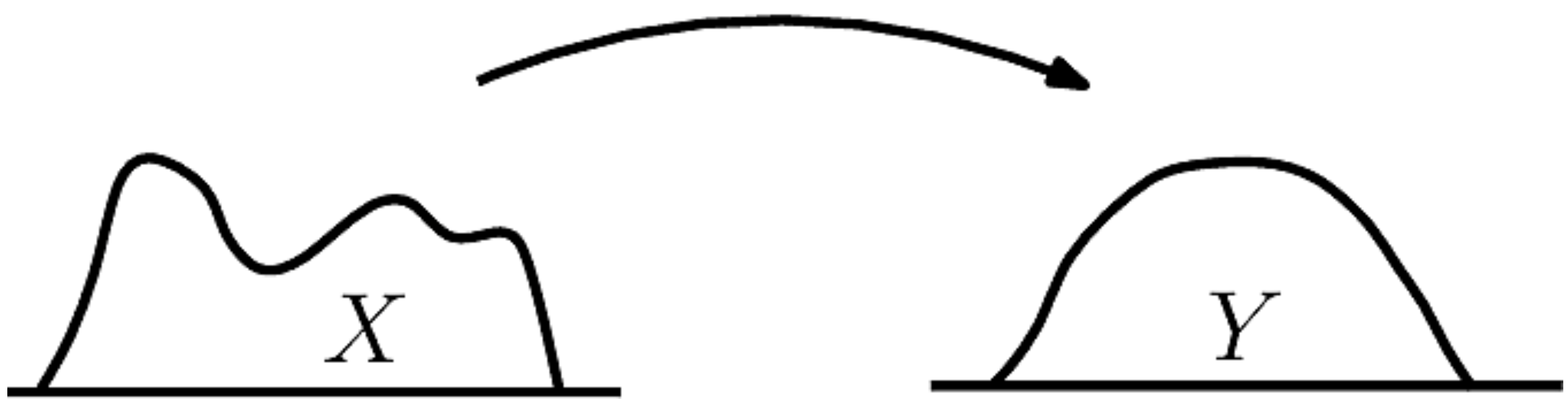
where $\Pi(\mathbb{P}_r, \mathbb{P}_g)$ denotes the set of all joint distributions $\gamma(x, y)$ whose marginals are respectively $\mathbb{P}_r$ and $\mathbb{P}_g$. Intuitively, $\gamma(x, y)$ indicates how much "mass" must be transported from x to y in order to transform the distributions $\mathbb{P}_r$ into the distribution $\mathbb{P}_g$. The EM distance then is the "cost" of the optimal transport plan.

(Arjovsky et al., 2017)

Solution 3: Earth Move Distance

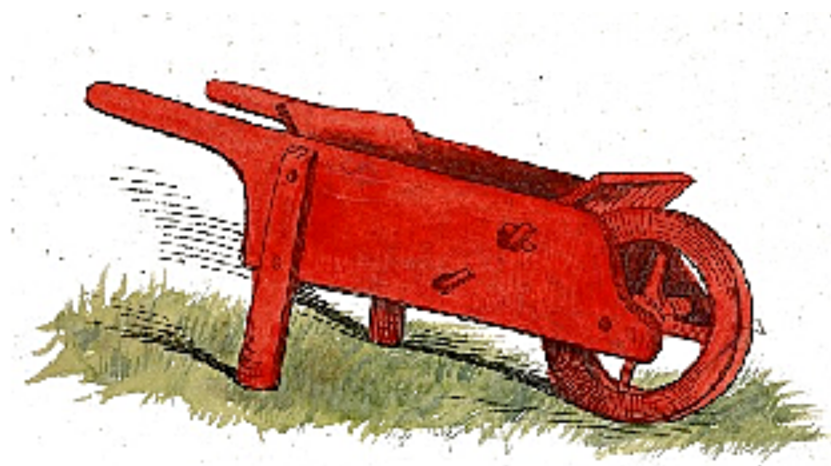
- Wasserstein-1 Distance (Earth-Mover Distance):

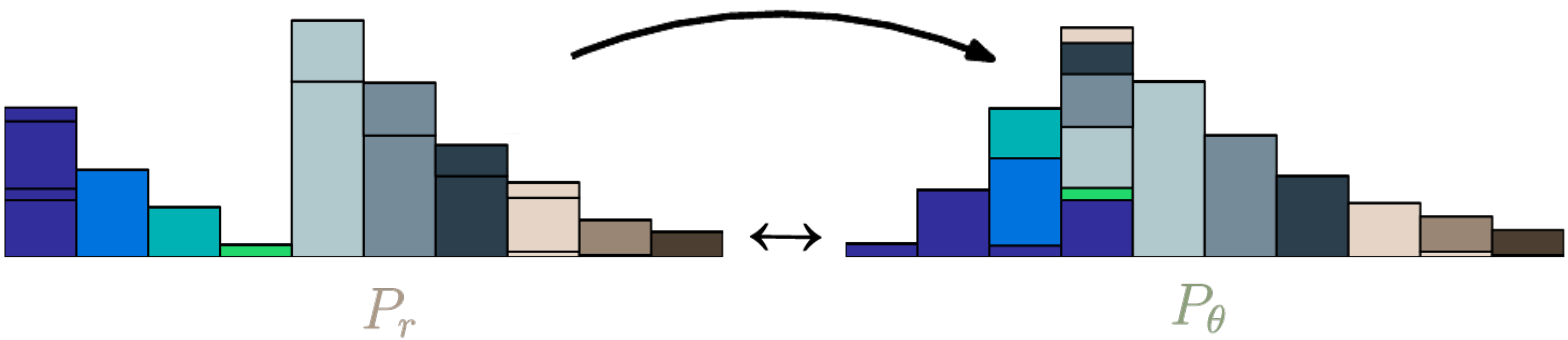




Solution 3: Earth Move Distance

- Wasserstein-1 Distance (Earth-Mover Distance):





(Vincent Herrmann's blog)

Solution 3: Wasserstein Distance

- Wasserstein-1 Distance (Earth-Mover Distance):

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}\big[\, \|x - y\| \,\big]$$

- The distance is shown to have the desirable property that under mild assumptions
  - It is continuous everywhere and
  - differentiable almost everywhere.

(Arjovsky et al., 2017)

Solution 3: Wasserstein Distance

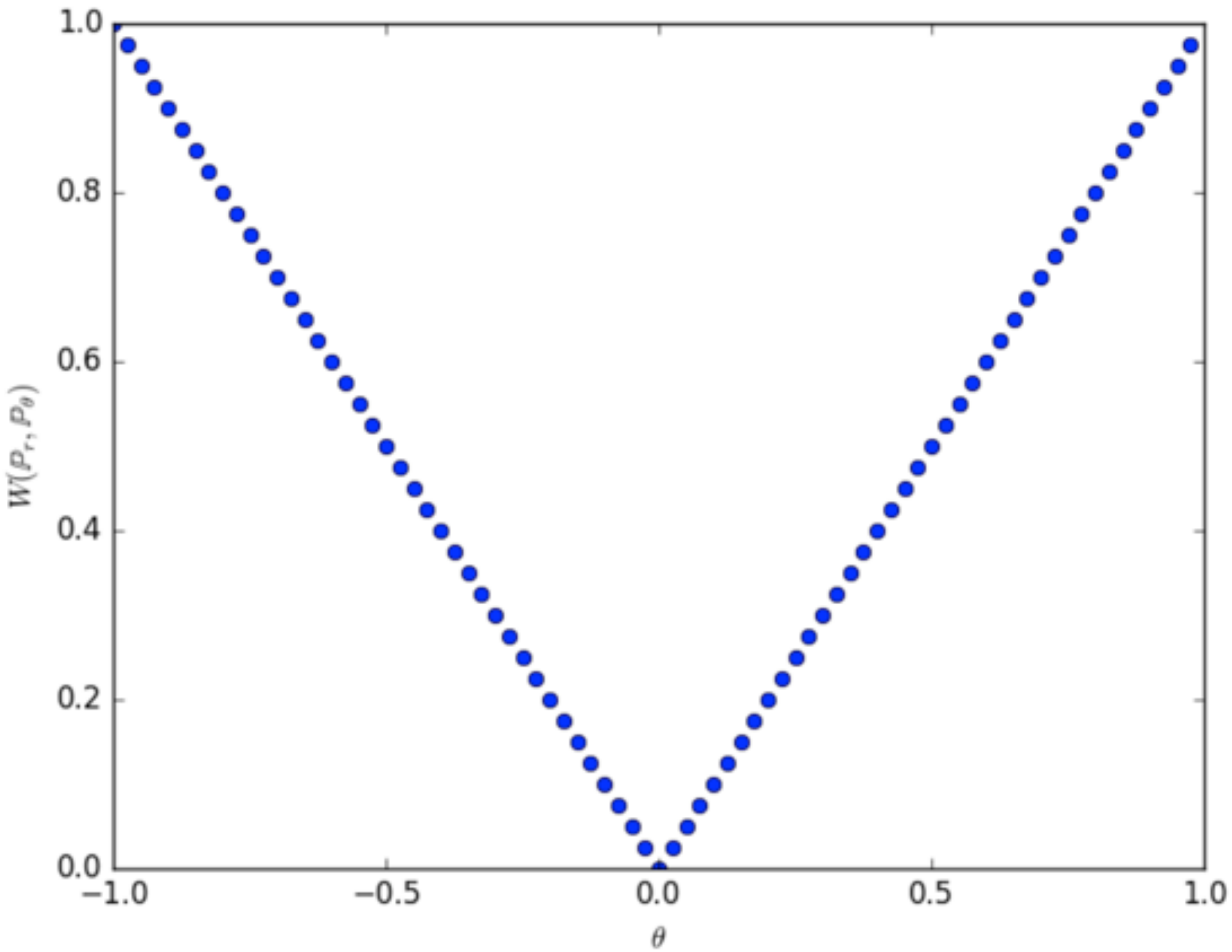
- Wasserstein-1 Distance (Earth-Mover Distance):

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma} \big[\|x - y\| \big]$$

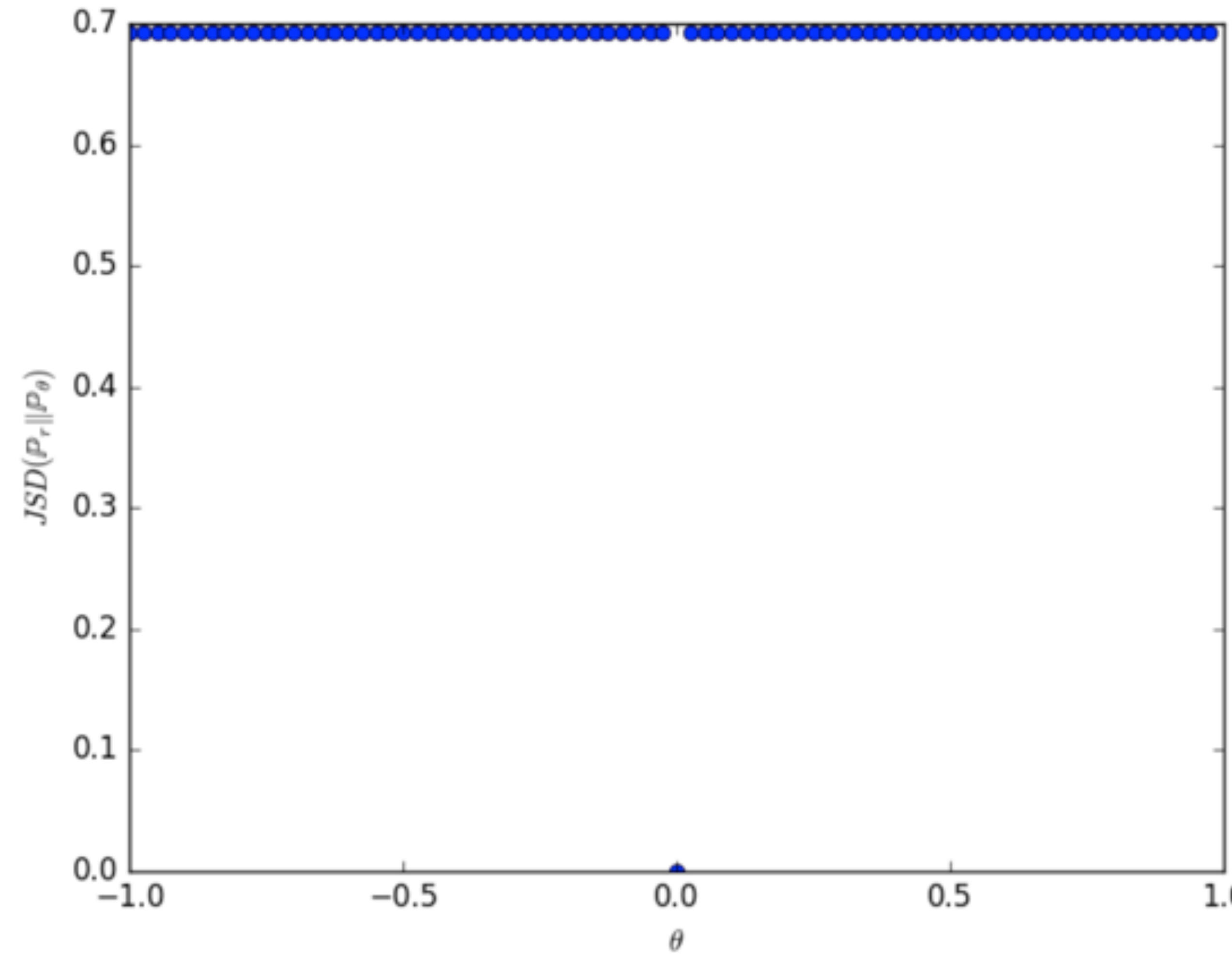
- The distance is shown to have the desirable property that under mild assumptions
  - *And most importantly, it can reflect the distance of two distributions even if they do not overlap, and thus can provide meaningful gradients*

(Arjovsky et al., 2017)

Solution 3: Wasserstein Distance



Wasserstein Distance



JS Divergence

(Arjovsky et al., 2017)

Solution 3: Wasserstein Distance

- Wasserstein-1 Distance (Earth-Mover Distance):

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma} \big[\, \|x - y\| \,\big]$$

- By applying the Kantorovich-Rubinstein duality (Villani, 2008), Wasserstein GANs becomes:

$$\min_{G} \max_{D \in \mathcal{D}} \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathbb{P}_r} \big[D(\boldsymbol{x})\big] - \mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}} \sim \mathbb{P}_g} \big[D(\tilde{\boldsymbol{x}}))\big]$$

(Arjovsky et al., 2017)

Wasserstein GANs

- This new value function of WGAN gives rise to the additional requirement that the discriminator must lie within in the space of 1-Lipschitz functions:

$$\min_{G} \max_{D \in \mathcal{D}} \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathbb{P}_r} \big[D(\boldsymbol{x})\big] - \mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}} \sim \mathbb{P}_g} \big[D(\tilde{\boldsymbol{x}}))\big]$$

- In other words, ***D*** is the set of 1-Lipschitz functions
  - Lipschitz continuity

(Arjovsky et al., 2017)

Lipschitz Continuity

- real-value function: $f: R \to R$
- positive constant: *K*

$$|f(x_1) - f(x_2)| \leq K|x_1 - x_2|$$

- In other words, a Lipschitz continuous function has bounded first derivative. Intuitively, the slope of a KK-Lipschitz function never exceeds KK, for a more general definition of slope.

$$d_Y(f(x_1), f(x_2)) \leq K d_X(x_1, x_2)$$

(Wikipedia)

Wasserstein GANs

- This new value function of WGAN gives rise to the additional requirement that the discriminator must lie within in the space of 1-Lipschitz functions:

$$\min_G \max_{D \in \mathcal{D}} \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathbb{P}_r} \left[D(\boldsymbol{x}) \right] - \mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}} \sim \mathbb{P}_g} \left[D(\tilde{\boldsymbol{x}})) \right]$$

- To satisfy this requirement, WGAN enforces the weights of ***D*** lie within a compact space *[-c, c]* by applying weight clipping

(Arjovsky et al., 2017)

Wasserstein GANs

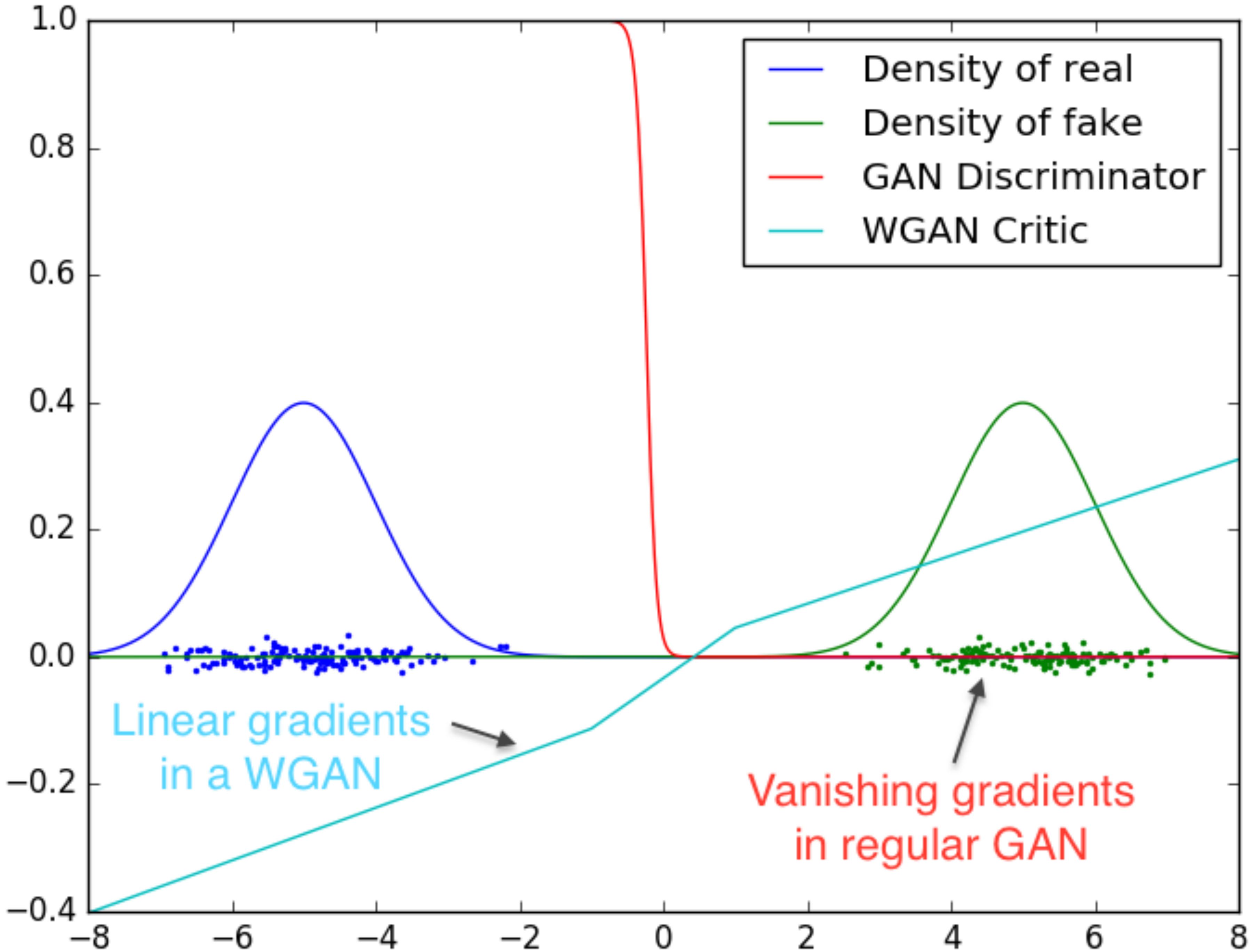
- This new value function of WGAN gives rise to the additional requirement that the discriminator must lie within in the space of 1-Lipschitz functions:

$$\min_{G} \max_{D \in \mathcal{D}} \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathbb{P}_r} \big[D(\boldsymbol{x}) \big] - \mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}} \sim \mathbb{P}_g} \big[D(\tilde{\boldsymbol{x}})) \big]$$

- Also, WGAN removes the sigmoid layer in ***D*** because by using Wasserstein distance, ***D*** in WGAN is doing regression rather than classification

(Arjovsky et al., 2017)

Wasserstein GANs



(Arjovsky et al., 2017)

Difficulty 2

Tackled!

- when:

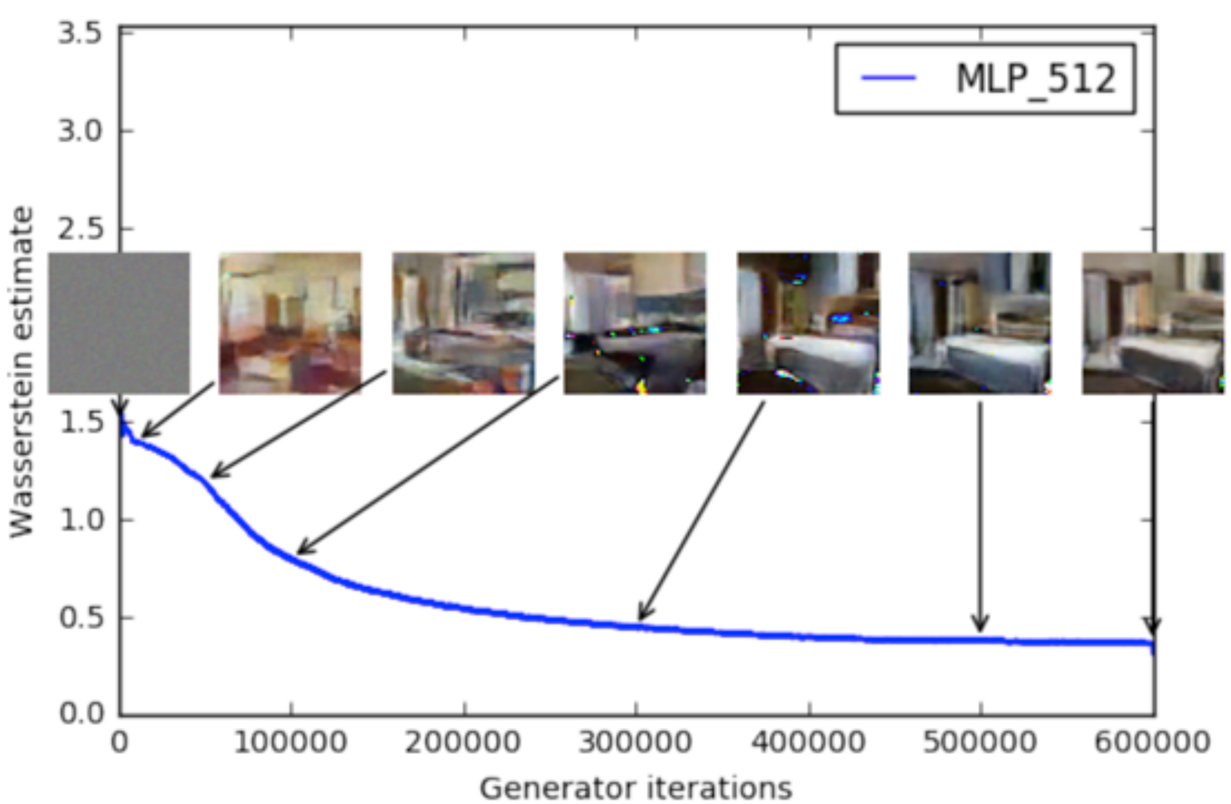
$$L(D^*, g_\theta) = 2JSD(\mathbb{P}_r \| \mathbb{P}_g) - 2\log 2$$

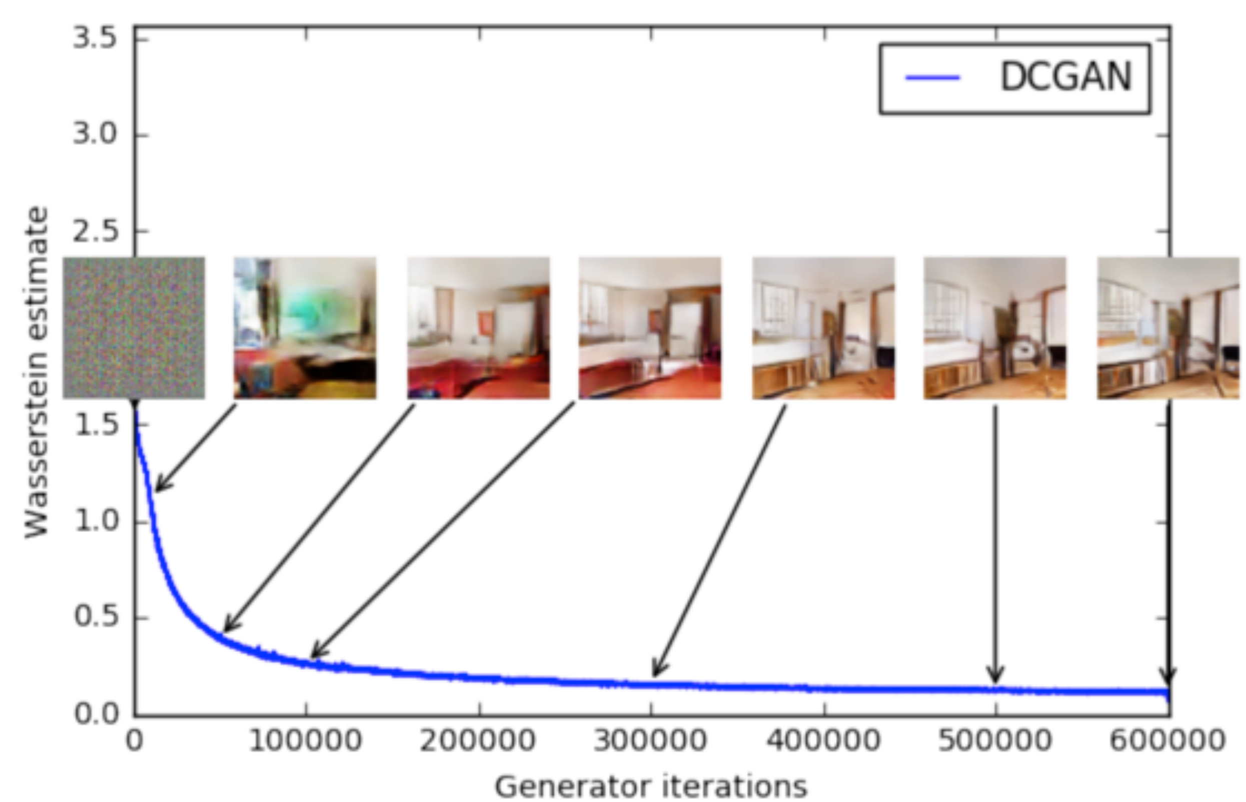
- The JS divergence for the two distributions P_r and P_g is (almost) always log2 because P_r and P_g hardly can overlap (Arjovsky & Bottou, 2017, Theorem 2.1~2.3)

- This results in vanishing gradient in theory!

Wasserstein GANs

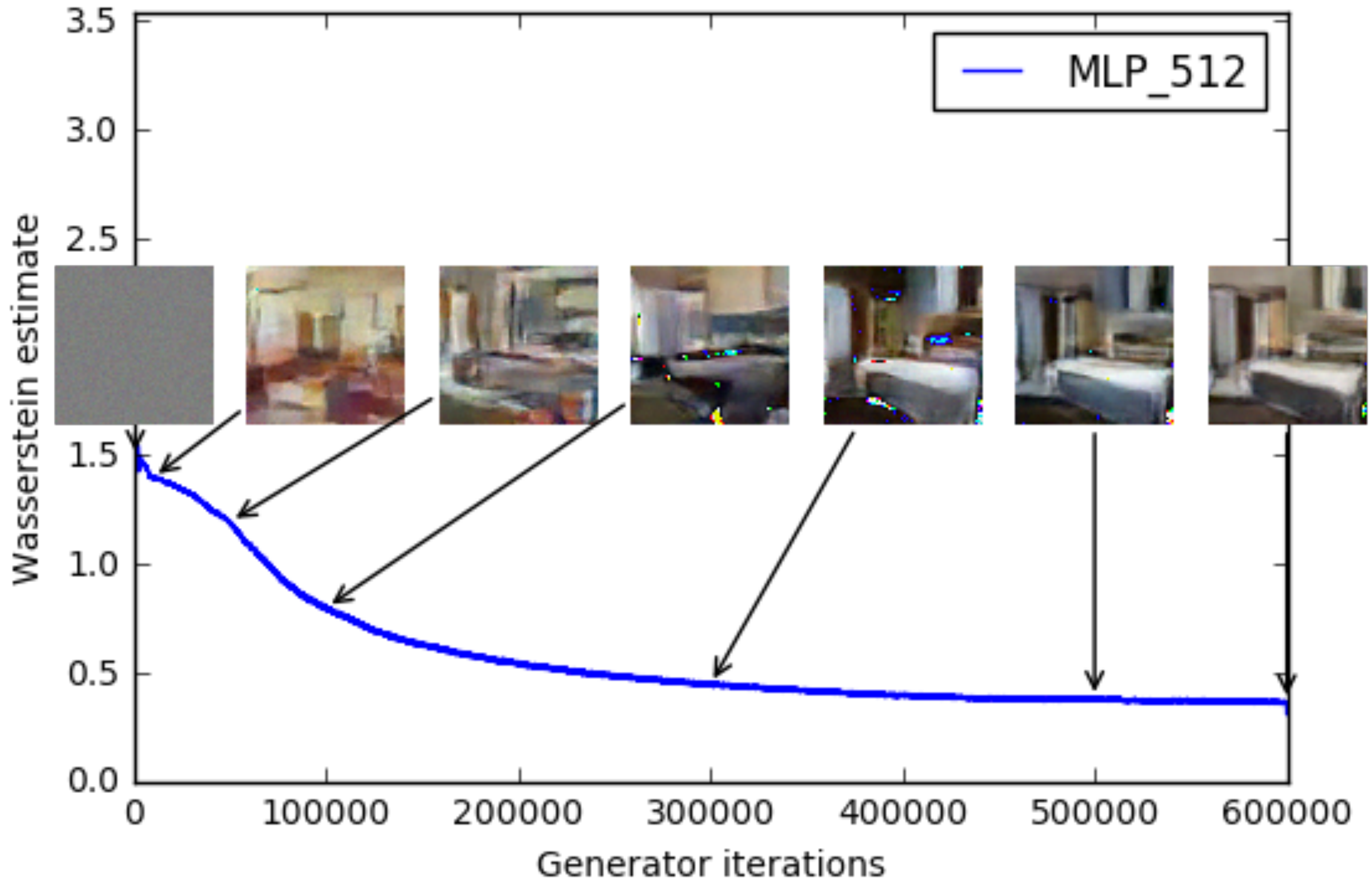
- This new value function of WGAN seems correlate with the quality of the generated samples:





(Arjovsky et al., 2017)

Wasserstein GANs



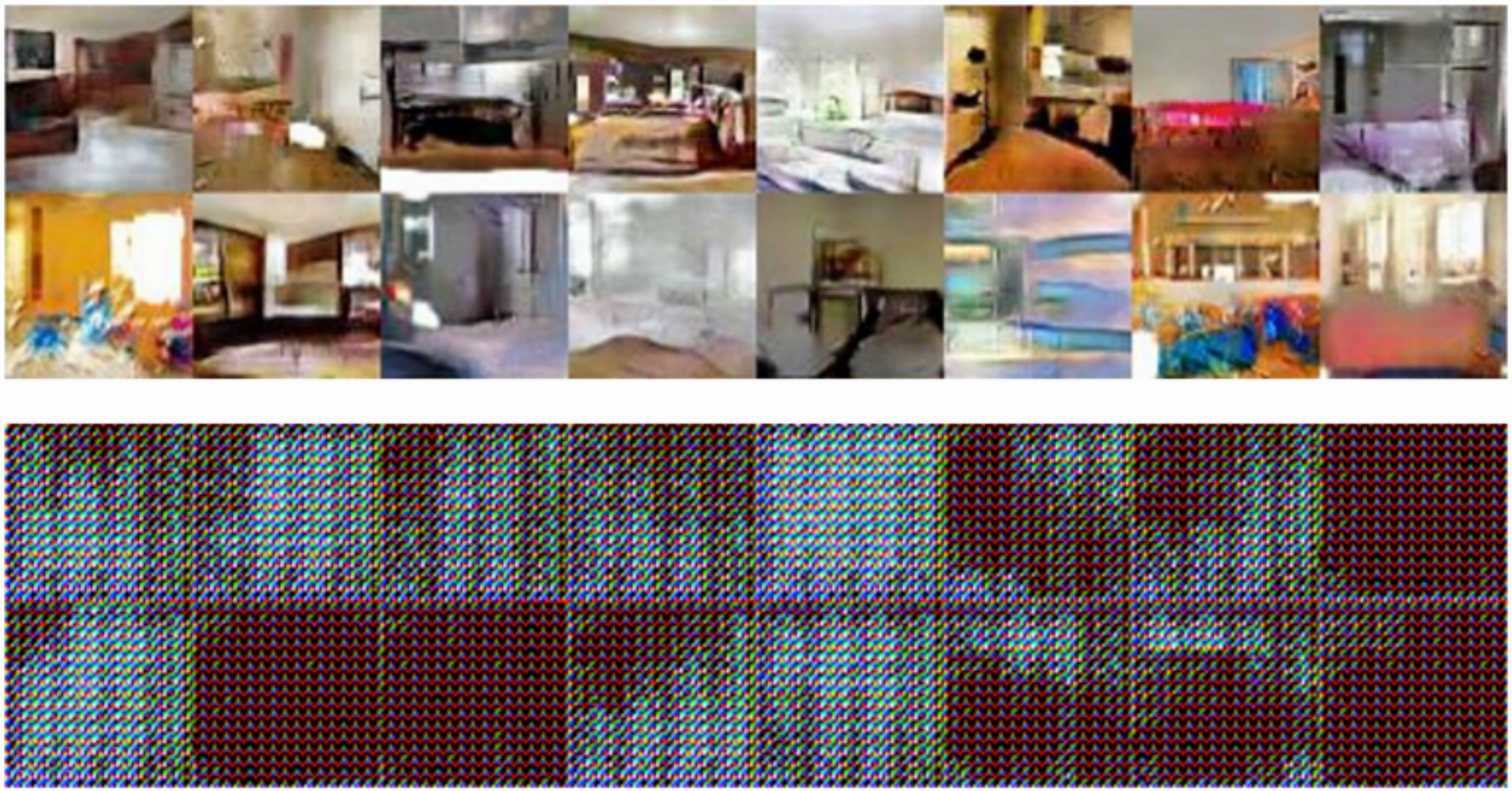
(Arjovsky et al., 2017)

Wasserstein GANs

Top: WGAN with the same DCGAN architecture. *Bottom:* DCGAN

(Arjovsky et al., 2017)

Wasserstein GANs



Top: WGAN with DCGAN architecture, no batch norm. *Bottom:* DCGAN, no batch norm.

(Arjovsky et al., 2017)

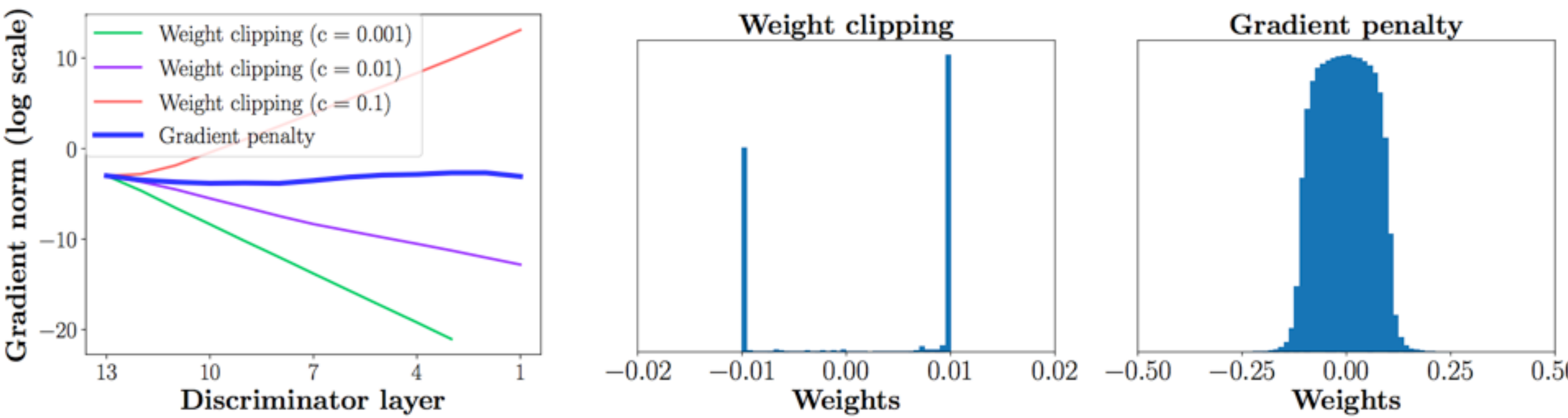
Wasserstein GANs

Top: WGAN with MLP architecture. *Bottom:* Standard GAN, same architecture.

(Arjovsky et al., 2017)

Improved Wasserstein GANs

- The drawbacks of weight clipping



(a) (b)

- bias ***D*** toward much simpler functions

(Gulrajani et al., 2017)

Improved Wasserstein GANs

- The drawbacks of weight clipping

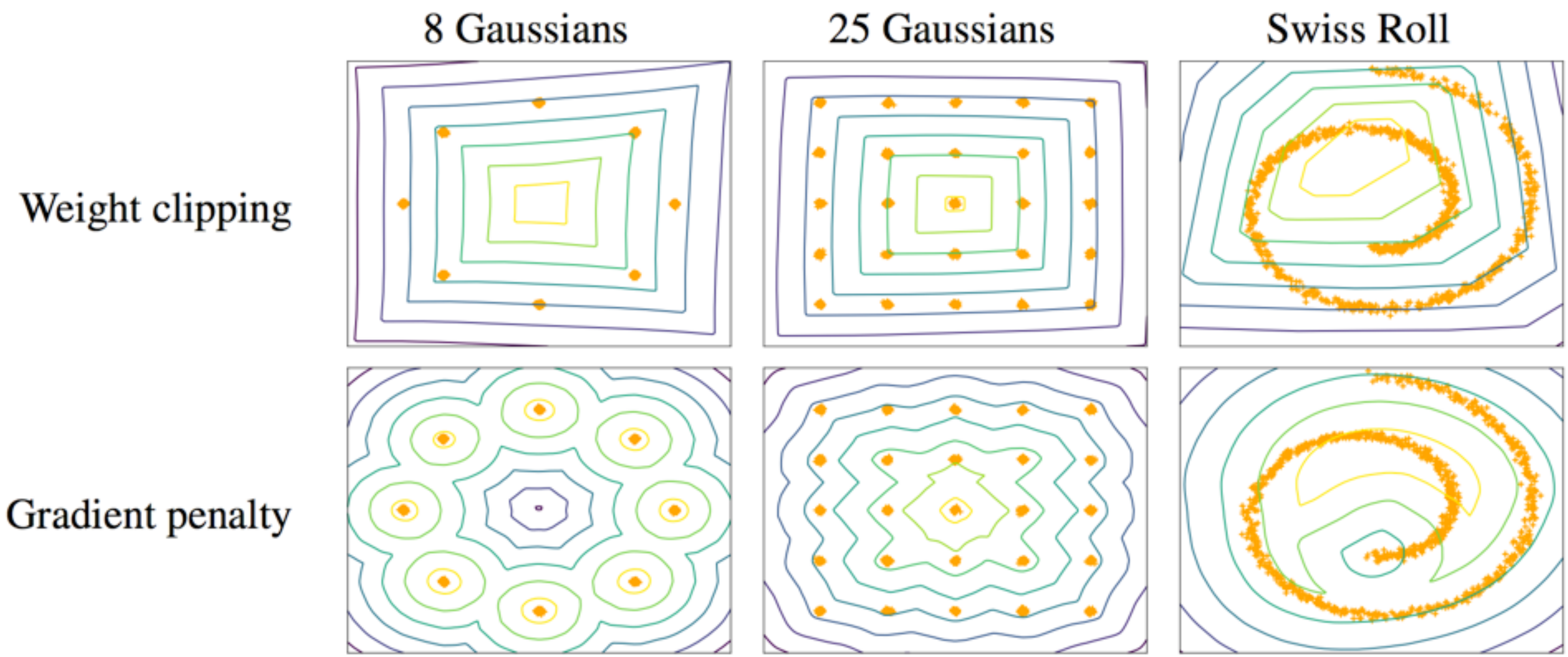
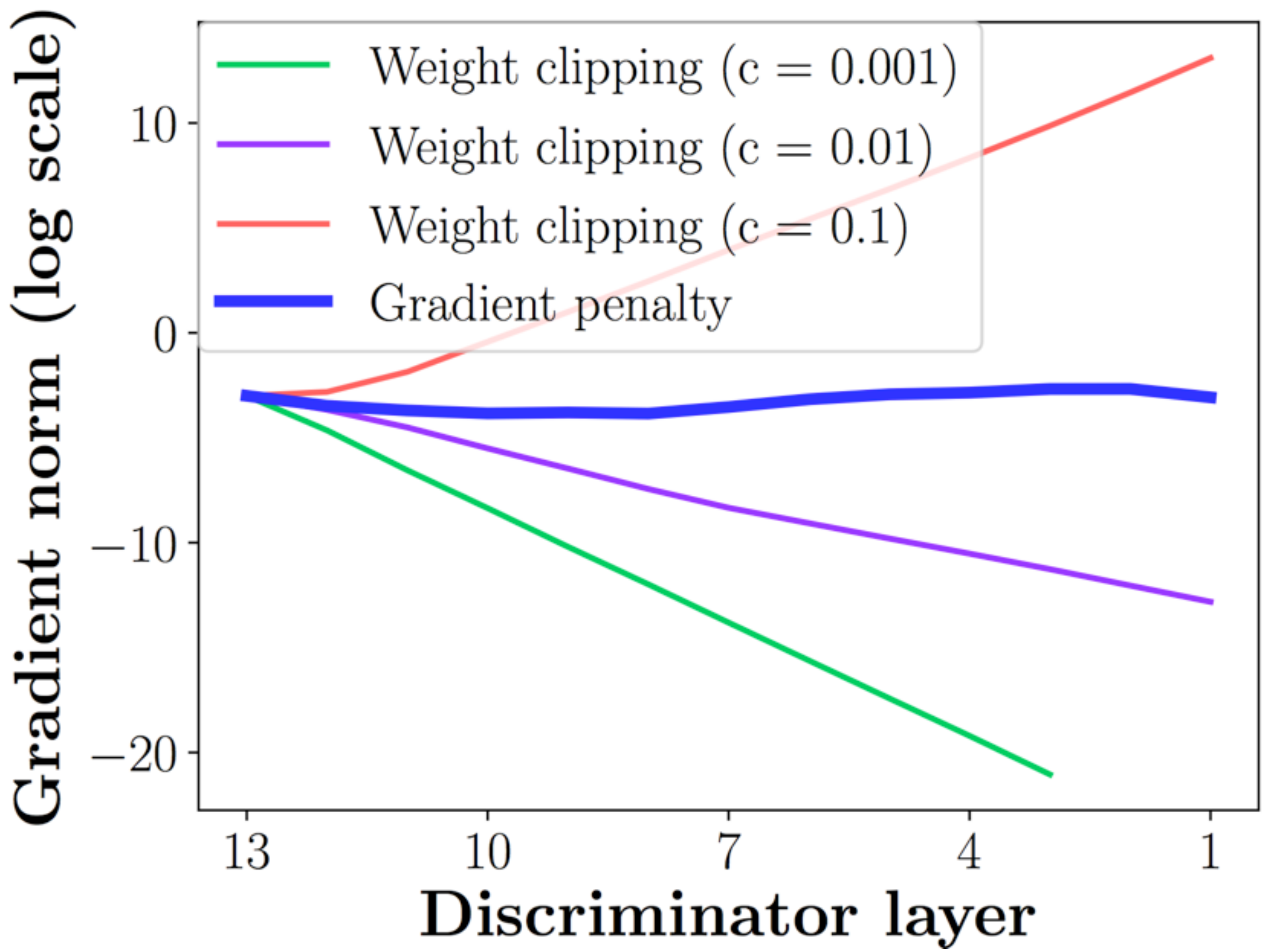


Figure 1: Value surfaces of WGAN critics trained to optimality on toy datasets. Critics trained with weight clipping fail to capture higher moments of the data distribution. The 'generator' is held fixed at the real data plus Gaussian noise.

(Gulrajani et al., 2017)

Improved Wasserstein GANs

- The drawbacks of weight clipping



(Gulrajani et al., 2017)

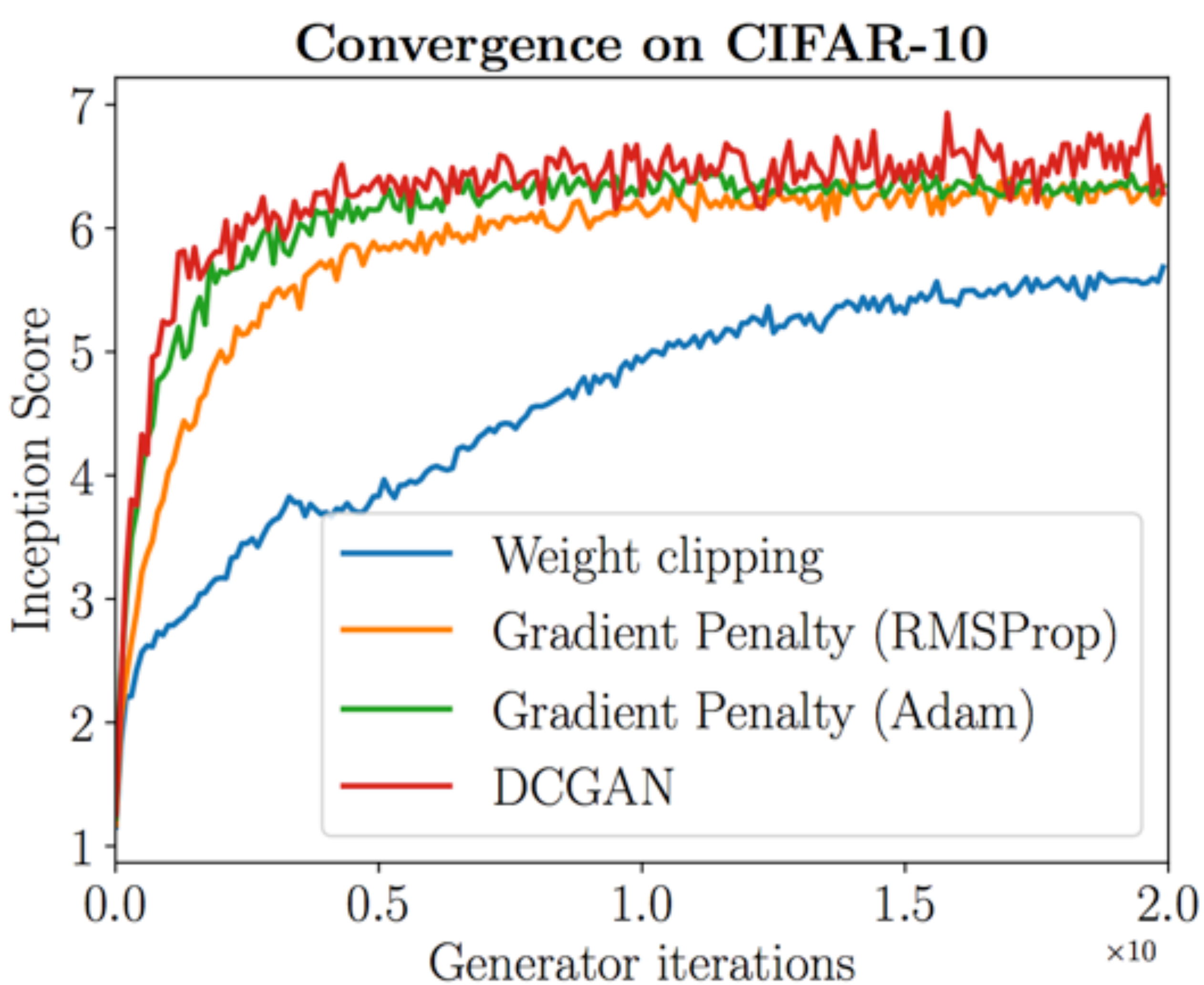
Improved Wasserstein GANs

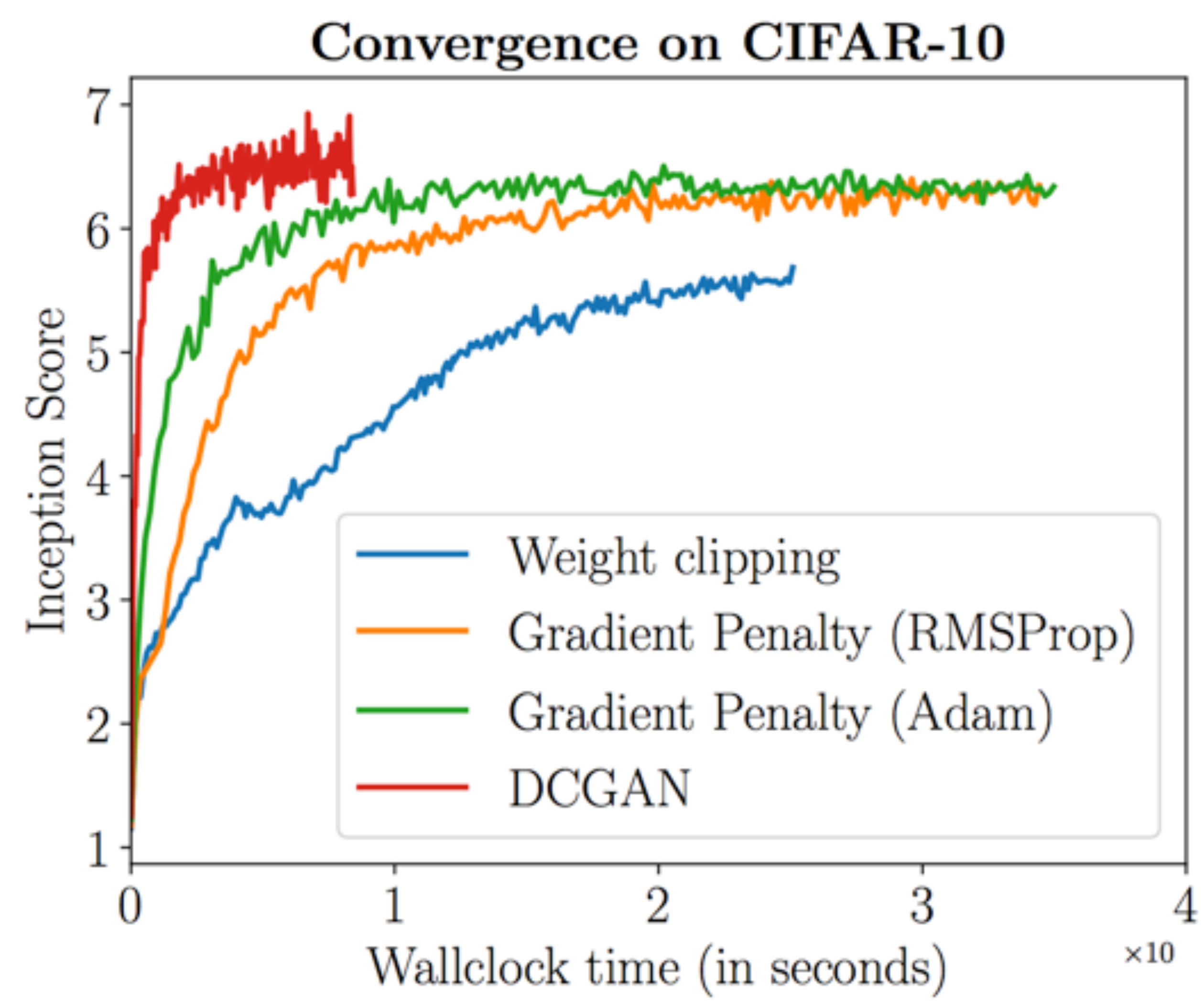
- The optimal ***D*** under WGAN:
  - has gradients with norm 1 almost everywhere under P_r and P_g.
- The objective of improved WGAN-GP:

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

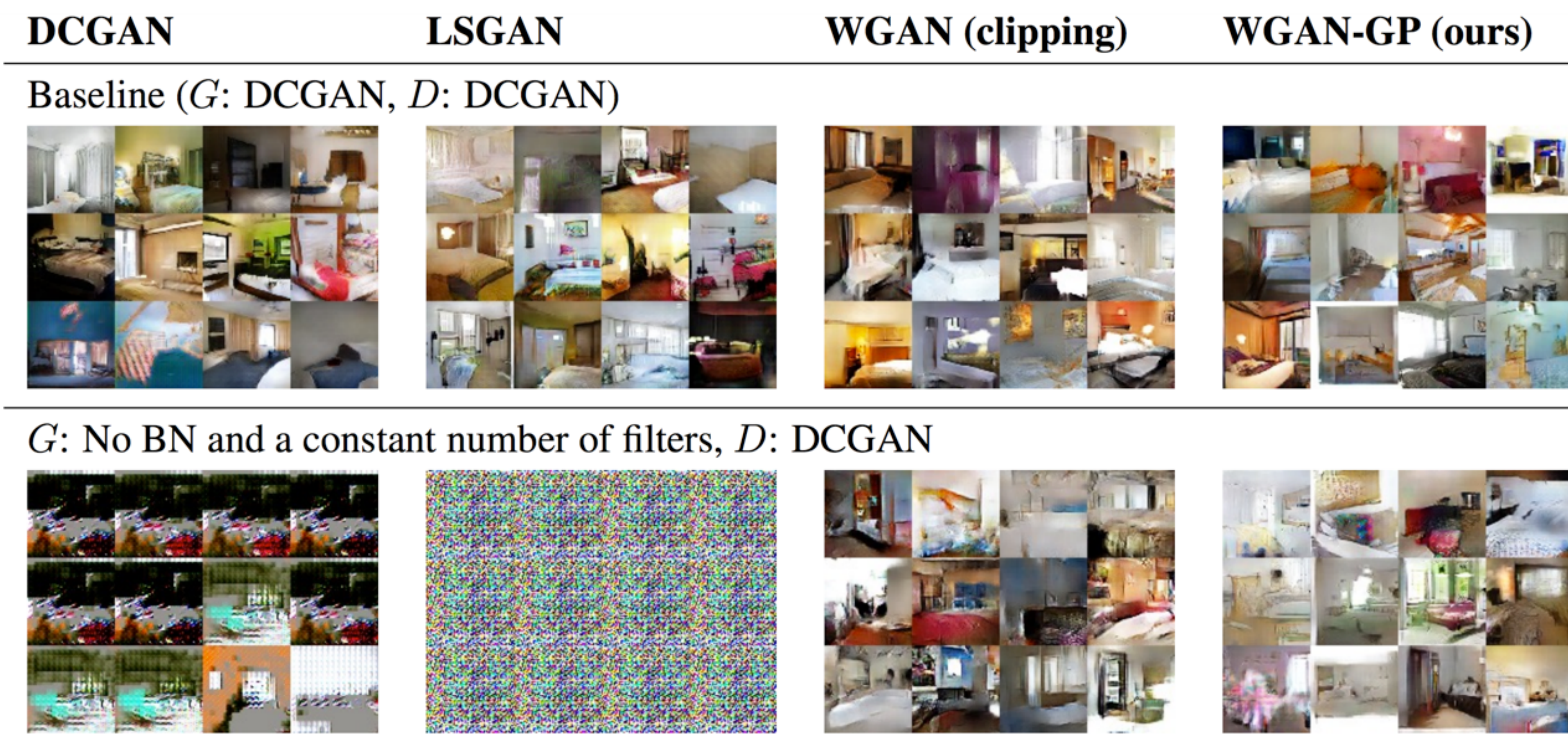
(Gulrajani et al., 2017)

Improved Wasserstein GANs





Improved Wasserstein GANs



Take-home Messages

- Try WGAN-GP
- Try noisy input
- Try specific architecture (with careful analysis of the certain problem)
- Try different type of the noise
- Checklist here: https://github.com/soumith/ganhacks

Thanks for your attention!
Any questions?

图森未来

扫描二维码了解更多

长按二维码关注

References

- Arjovsky and Bottou, “Towards Principled Methods for Training Generative Adversarial Networks”. ICLR 2017.
- Goodfellow et al., “Generative Adversarial Networks”. ICLR 2014.
- Che et al., “Mode Regularized Generative Adversarial Networks”. ICLR 2017.
- Zhao et al., “Energy-based Generative Adversarial Networks”. ICLR 2017.
- Berthelot et al., “BEGAN: Boundary Equilibrium Generative Adversarial Networks”. arXiv preprint 2017.
- Sønderby, et al., “Amortised MAP Inference for Image Super-Resolution”. ICLR 2017.
- Arjovsky et al., “Wasserstein GANs”. ICML 2017.
- Villani, Cedric. “Optimal transport: old and new”, volume 338. Springer Science & Business Media, 2008

References

- Jun-Yan Zhu*, Taesung Park*, Phillip Isola, Alexei A. Efros. “Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks”. arXiv preprint 2017.
- Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, Jiwon Kim. “Learning to Discover Cross-Domain Relations with Generative Adversarial Networks”. ICML 2017.
- Zili Yi, Hao Zhang, Ping Tan, Minglun Gong. “DualGAN: Unsupervised Dual Learning for Image-to-Image Translation”. arXiv preprint 2017.
- Jeff Donahue, Philipp Krähenbühl, Trevor Darrell. “Adversarial Feature Learning”. ICLR 2017.
- Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, Aaron Courville. “Adversarially Learned Inference”. ICLR 2017.
- Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, Aaron Courville. “Improved Training of Wasserstein GANs”. arXiv preprint 2017.

References

- Mehdi Mirza, Simon Osindero. "Conditional Generative Adversarial Nets". arXiv preprint 2014.
- Emily Denton, Soumith Chintala, Arthur Szlam, Rob Fergus. "Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks". arXiv preprint 2015.
- Alec Radford, Luke Metz, Soumith Chintala. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks". ICLR 2016.
- Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, Honglak Lee. "Generative Adversarial Text to Image Synthesis". ICML 2016.
- Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen. "Improved Techniques for Training GANs". arXiv preprint 2016.
- Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaolei Huang, Xiaogang Wang, Dimitris Metaxas. "StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks". arXiv preprint 2016.
- Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros. "Image-to-Image Translation with Conditional Adversarial Networks". CVPR 2017.

References

- Yaniv Taigman, Adam Polyak, Lior Wolf. “Unsupervised Cross-Domain Image Generation”. ICLR 2017.
- Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. “Generative Adversarial Nets”. NIPS 2014.
- Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman and Alexei A. Efros. “Generative Visual Manipulation on the Natural Image Manifold”, ECCV 2016.
- Huikai Wu, Shuai Zheng, Junge Zhang, Kaiqi Huang. “GP-GAN: Towards Realistic High-Resolution Image Blending”. arXiv preprint 2017.
- Anh Nguyen, Jeff Clune, Yoshua Bengio, Alexey Dosovitskiy, Jason Yosinski. “Plug & Play Generative Networks: Conditional Iterative Generation of Images in Latent Space”. CVPR 2017.