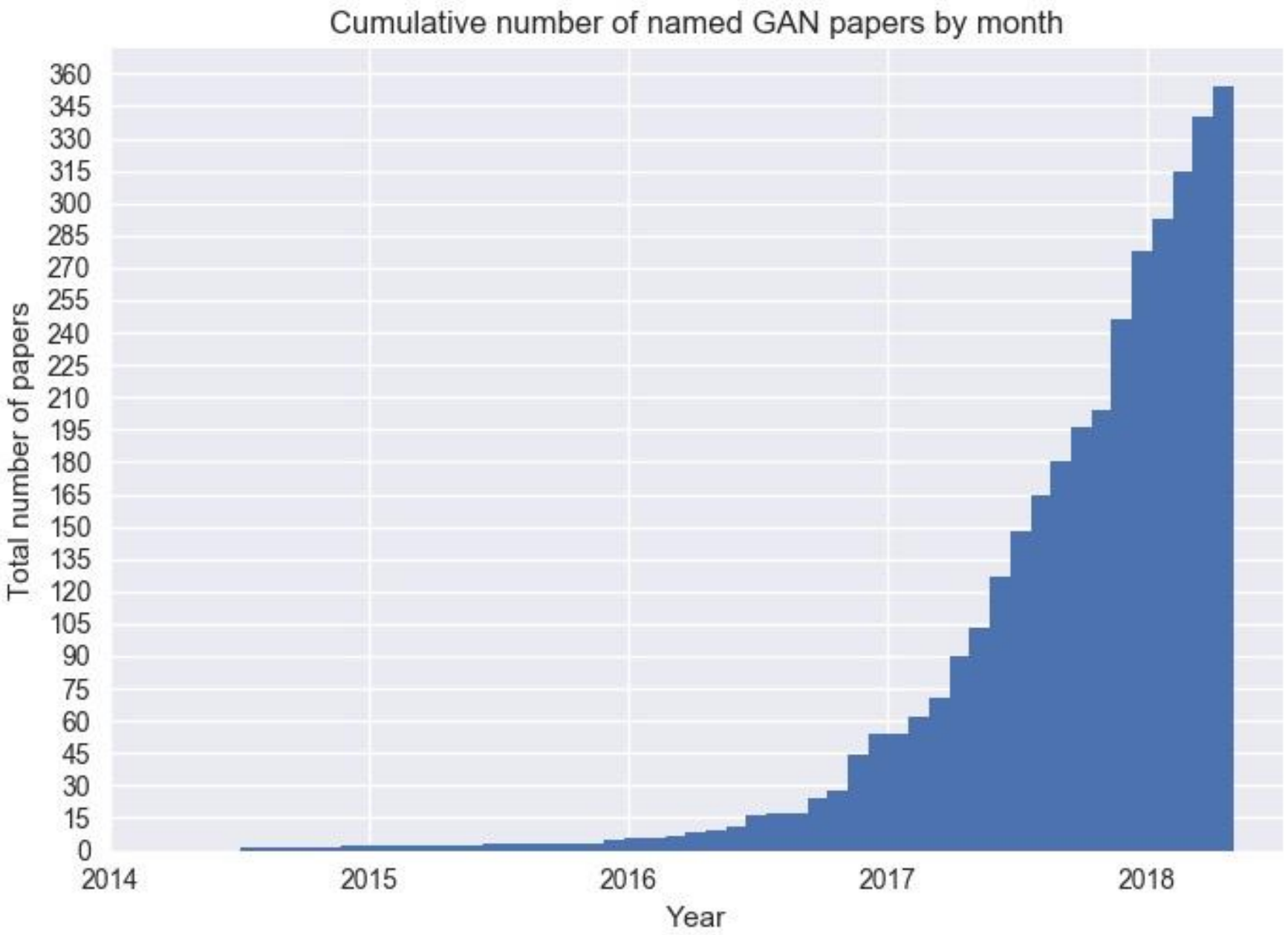
Fundamentals of Media Processing

Lecturer:
池畑　諭（Prof. IKEHATA　Satoshi）
児玉　和也（Prof. KODAMA Kazuya）

Support:
佐藤　真一（Prof. SATO　Shinichi）
孟　洋（Prof. MO　Hiroshi）

Generative Adversarial Networks


Cumulative number of named GAN papers by month
Total number of papers
360
345
330
315
300
285
270
255
240
225
210
195
180
165
150
135
120
105
90
75
60
45
30
15
0
2014
2015
2016
2017
2018
Year

Some Materials from Tutorials at CVPR2017

Tutorial on Theory and Application of Generative Adversarial Networks

Event: CVPR 2017 at Honolulu

Date: Wednesday, 7/26/2017

Organizers: Ming-Yu Liu, Julie Bernauer, Jan Kautz

Time: PM 13:30 --- 14:30

Description

Generative adversarial network (GAN) has recently emerged as a promising generative modeling approach. It consists of a generative network and a discriminative network. Through the competition between the two networks, it learns to model the data distribution. In addition to modeling the image/video distribution in computer vision problems, the framework finds use in defining visual concept using examples. To a large extent, it eliminates the need of hand-crafting objective functions for various computer vision problems. In this tutorial, we will present an overview of generative adversarial network research. We will cover several recent theoretical studies as well as training techniques and will also cover several vision applications of generative adversarial networks.

Outlines

https://github.com/mingyuliutw/cvpr2017_gan_tutorial

slides

Differentiable Generator Networks

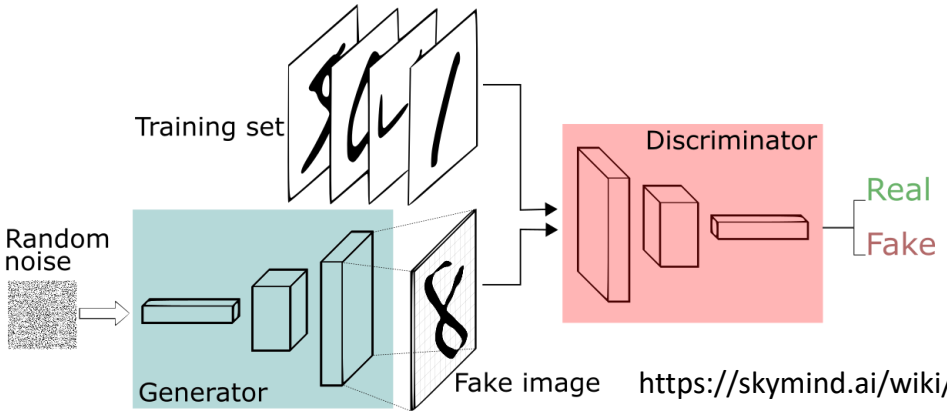
- ***Generative Adversarial Networks***(GAN; Goodfellow2014) are a generative modeling approach based on ***differentiable generator networks*** which is paired with ***discriminator networks***

- The model transforms samples of latent variables $\boldsymbol{z}$ to sample $\boldsymbol{x}$ or to distributions over samples $\boldsymbol{x}$ using a differentiable function $g(\boldsymbol{z}; \theta^{(g)})$, which is typically represented by a neural network

- This model class also includes ***variational autoencoders*** (VAE; Kingma2014), which constrains the encoder to output a conditional Gaussian distribution, and the decoder reconstructs inputs from samples from conditional distribution (generate blurry images due to the gaussian modeling)

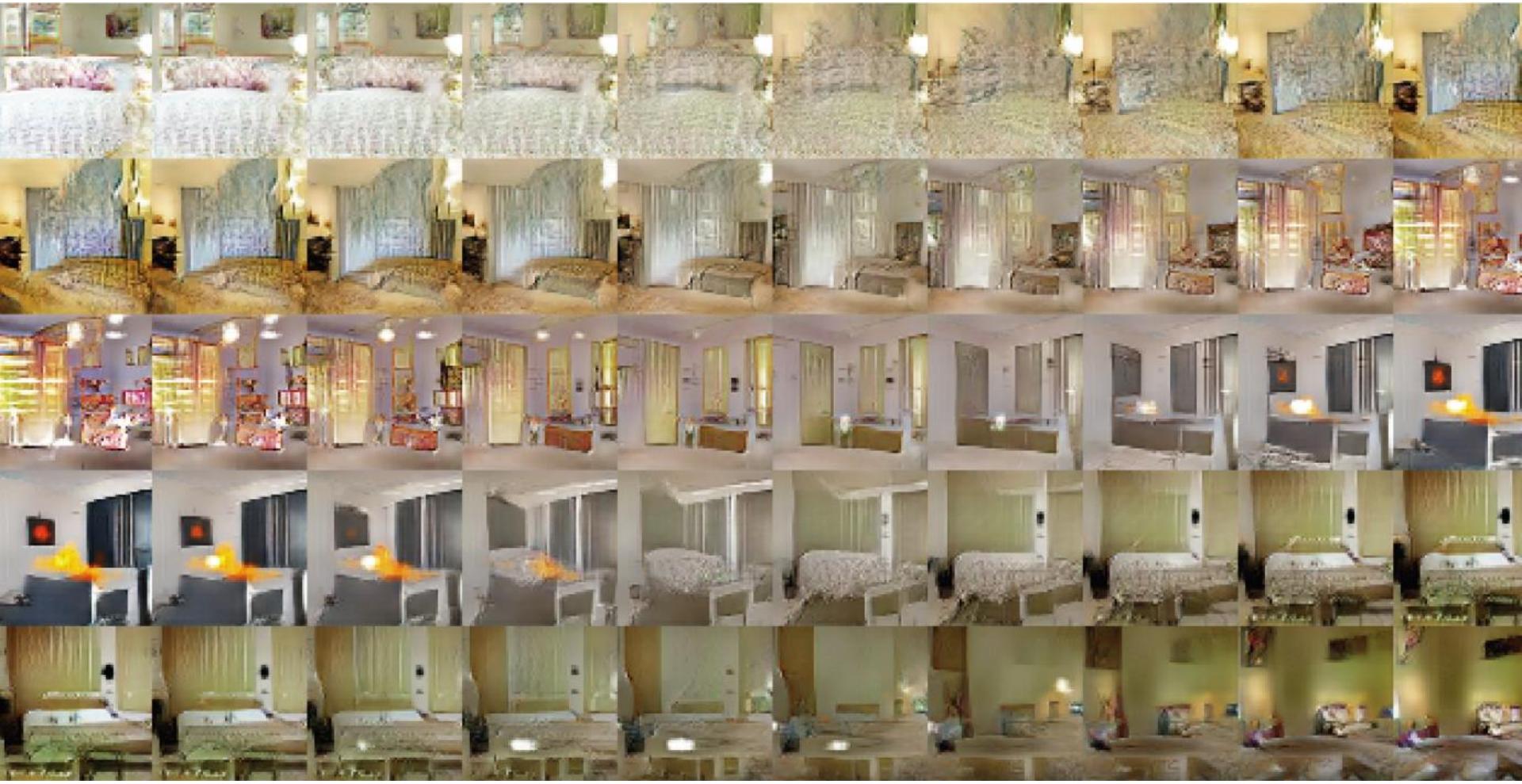
Generative Adversarial Networks

- GAN (or Deep Convolutional GAN; DCGANN, Redford2016) are based on a game theoretic scenario in which the generator network must compete against an adversary
  - The ***generator network*** directly produces samples $\boldsymbol{x} = g\left(\boldsymbol{z}; \theta^{(g)}\right)$
  - The ***discriminator network*** attempts to distinguish between samples drawn from the training data and samples drawn from the generator
  - The discriminator network emits a probability value given by $d\left(\boldsymbol{x}; \theta^{(g)}\right)$, indicating the probability that $\boldsymbol{x}$ is a real training example rather than a fake sample drawn from the model



https://skymind.ai/wiki/generative-adversarial-network-gan

Generative Adversarial Networks



Generative Adversarial Networks

- The simplest way to formulate learning in generative adversarial networks is as a zero-sum game, in which a function $v(\boldsymbol{\theta}^{(g)}, \boldsymbol{\theta}^{(d)})$ determines the payoff of the discriminator. The generator receives $-v(\boldsymbol{\theta}^{(g)}, \boldsymbol{\theta}^{(d)})$ as its own payoff. During learning, each player attempts to maximize its own payoff, so that at convergence

$$g^* = \arg\min_g \max_d \underbrace{\mathbb{E}_{x \sim p_{x(data)}} \log d(\boldsymbol{x}) + \mathbb{E}_{x \sim p_{z(model)}} \log\big(1 - d(g(\boldsymbol{z}))\big)}_{v(\boldsymbol{\theta}^{(g)}, \boldsymbol{\theta}^{(d)})}$$

- This drives *the discriminator to attempt to learn to correctly classify samples as real or fake*. Simultaneously *the generator attempts to fool the classifier into believing its samples are real*. At convergence, the generator's samples are indistinguishable from real data, and the discriminator outputs ½ everywhere

Alternating Gradient Updates

- Alternating gradient algorithm is generally applied to train the GANs

(1) Fix g and perform a gradient step to

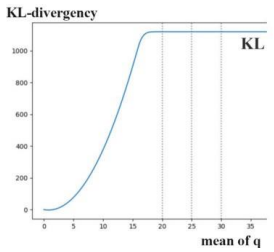
$$\max_d \mathbb{E}_{x\sim p_{x(data)}} \log d(\boldsymbol{x}) + \mathbb{E}_{x\sim p_{z(model)}} \log\big(1 - d(g(\boldsymbol{z}))\big)$$

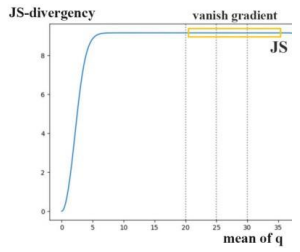
(2) Fix d and perform a gradient step to

- $\min_g \mathbb{E}_{x\sim p_{z(model)}} \log\big(1 - d(g(\boldsymbol{z}))\big)$ (in theory)
- $\max_g \mathbb{E}_{x\sim p_{z(model)}} \log d(g(\boldsymbol{z}))$ (in practice)

- Under an ideal discriminator, the generator minimizes the Jensen-Shannon divergence between p_x and $p_{g(z)}$

Proof:https://medium.com/@jonathan_hui/proof-gan-optimal-point-658116a236fb





Non-convergence in GANs

- GAN training is difficult in practice because
  - g and d are represented by neural networks and $\max_d v(g, d)$ is not convex
  - In practice, the generator is harder to learn than discriminator. If the generator is not doing a good job yet, the gradient for the generator diminishes and the generator learns nothing (***Model collapse***)

- The challenge is how to train the generator successfully by avoiding the vanishing gradient problem
  - LSGAN, WGAN, WGAN-GP, EBGAN, BEGAN, UnrolledGAN, DRAGAN

LSGAN

- Least-squared GANS (MAO2016): is designed to help the generator better by minimizing *Peason chi-square distance* between $p_{x(data)} + p_{z(model)}$ and $2p_{z(model)}$ (e.g,. $a, b, c = 0, 1, 1$)

$$\min_D \mathcal{J}(D) = \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[(D(\boldsymbol{x}) - b)^2\right] + \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z})) - a)^2\right]$$

$$\min_G \mathcal{J}(G) = \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z})) - c)^2\right]$$

- When $p_{x(data)}$ and $p_{z(model)}$ are different, the gradient does not vanish, and the solution converges as p_z approaches p_x
- Intuitively, LSGAN wants the target discriminator label for real images to be 1 and generated images to be 0. And for the generator, it wants the target label for generated images to be 1.

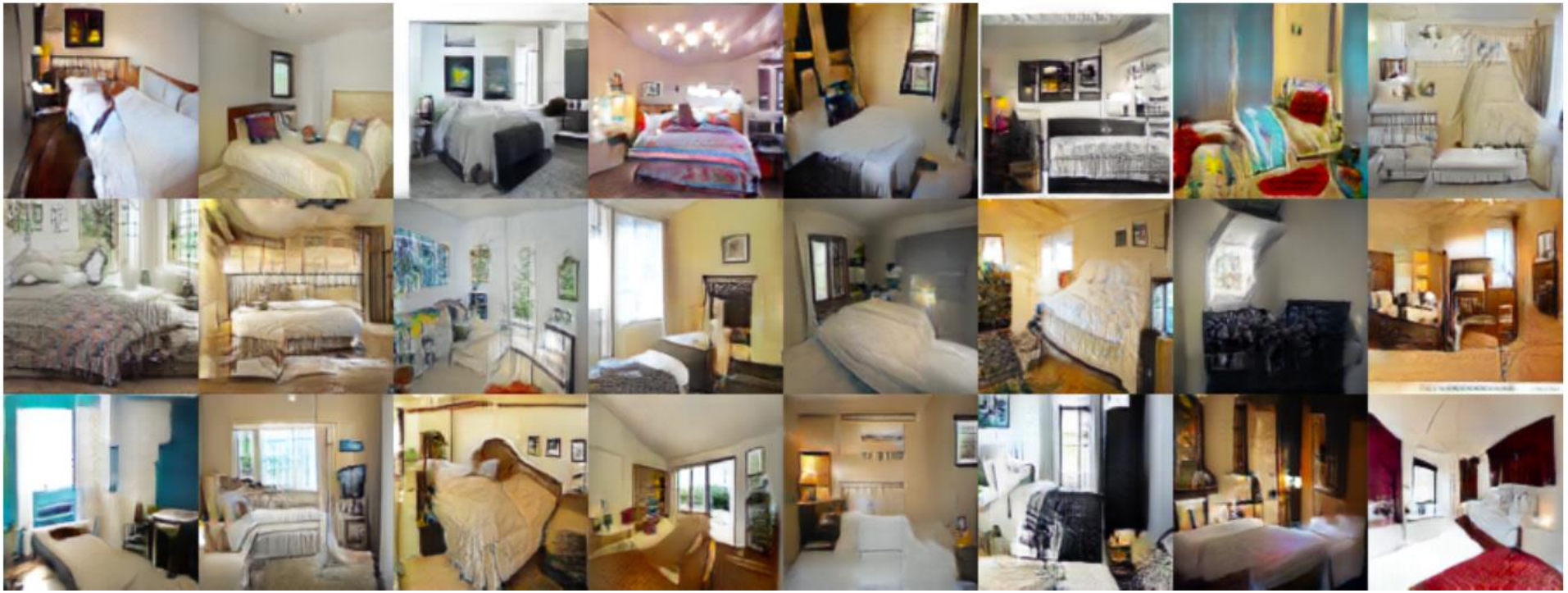
$$\min_D V_{\text{LSGAN}}(D) = \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[(D(\boldsymbol{x}) - 1)^2\right] + \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z})))^2\right]$$

$$\min_G V_{\text{LSGAN}}(G) = \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z})) - 1)^2\right].$$

LSGAN

GAN

LSGAN



Wasserstein GAN

- Wasserstein GAN (Arjovsky2016) minimizes the *earth mover distance* between $p_{x(data)}$ and $p_{z(model)}$
- A distribution mapping function (critic) that is continuous with respect to its parameters and ***locally lipschitz*** has a continuous and almost everywhere differentiable Wasserstein distance.

$$EM(p_X, p_{G(Z)}) = \inf_{\gamma \in \prod(p_X, p_{G(Z)})} E_{(x,y)\sim\gamma}[||x - y||]$$

Discriminator

GAN $\max_D E_{x\sim p_X}[\log D(x)] + E_{z\sim p_Z}[\log(1 - D(G(z))]$

WGAN $\max_D E_{x\sim p_X}[D(x)] - E_{z\sim p_Z}[D(G(z))]$

Generator

GAN $\max_G E_{z\sim p_Z}[\log D(G(z))]$

WGAN $\max_G E_{z\sim p_Z}[D(G(z))]$

64

D and G should be locally lipschitz (e.g., w is small enough): clip w between –c and c (e.g., 0.001)

$$\min_G \max_D E_{x\sim p_X}[D(x)] - E_{z\sim p_Z}[D(G(Z))] + \lambda E_{y\sim p_Y}[(||\nabla_y D(y)||_2 - 1)^2]$$

$$y = ux + (1 - u)G(z)$$

- *y:* imaginary samples

WGAN-GP (Gulrajani2017) uses gradient penalty for satisfying Lipschitz constraint (1-lipschitz) without clipping

DCGAN **LSGAN**

Baseline (*G*: DCGAN, *D*: DCGAN)

WGAN (clipping) **WGAN-GP (ours)**

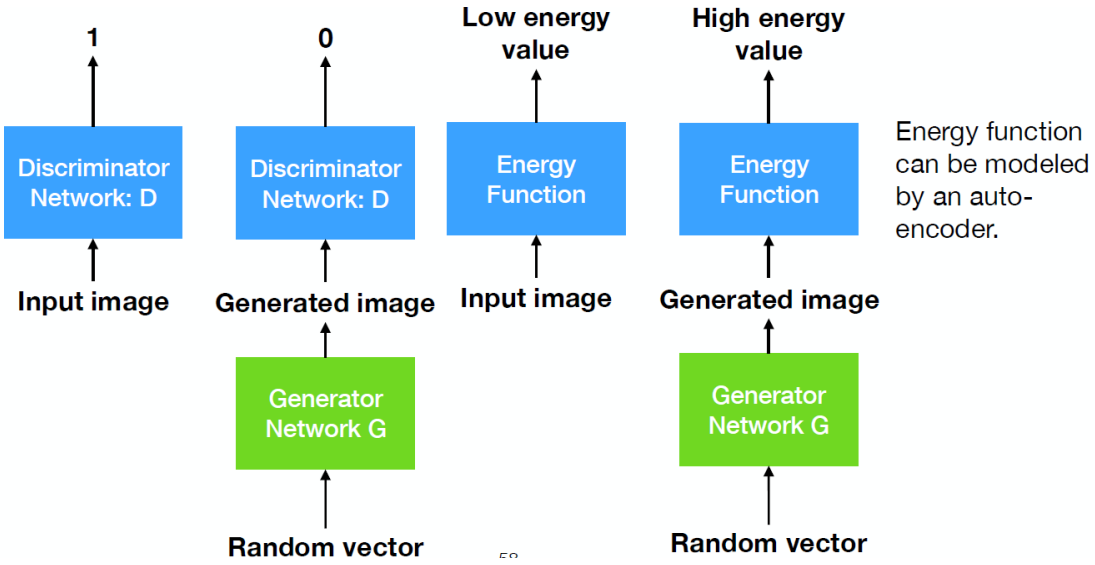
EBGAN

- Energy-Based GAN (Zhao2017) minimizes the energy function based on the autoencoder which helps to stabilize the discriminator training in the early stage (pretraining is available)

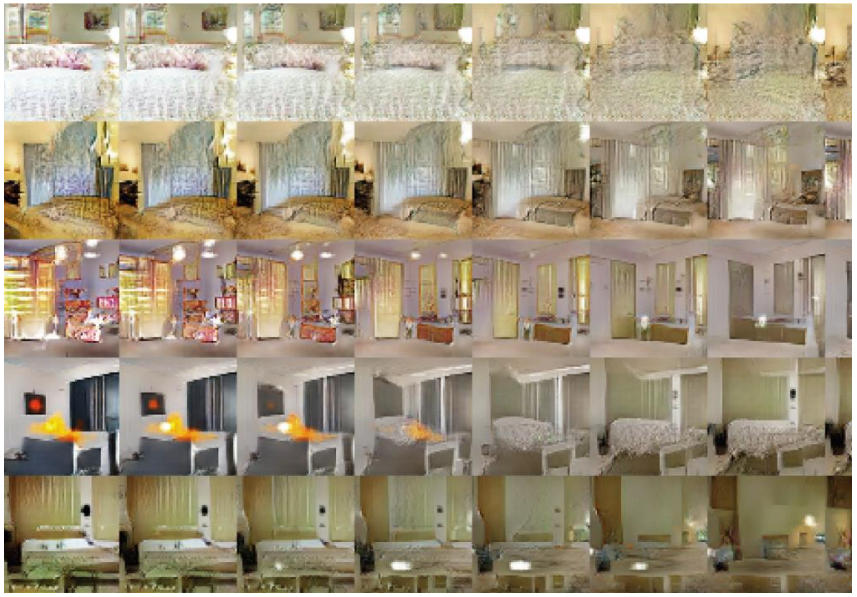
$$En(\boldsymbol{x}) = \left\| Decoder\left(Encoder(\boldsymbol{x})\right) - \boldsymbol{x} \right\| \qquad (= \max(0, m - En(G(\boldsymbol{z}))$$

- Discriminator: $\min_E \mathbb{E}_{x \sim p_{x(data)}}[En(\boldsymbol{x})] + \mathbb{E}_{x \sim p_{z(model)}}\left[m - En(G(\boldsymbol{z}))\right]^+$

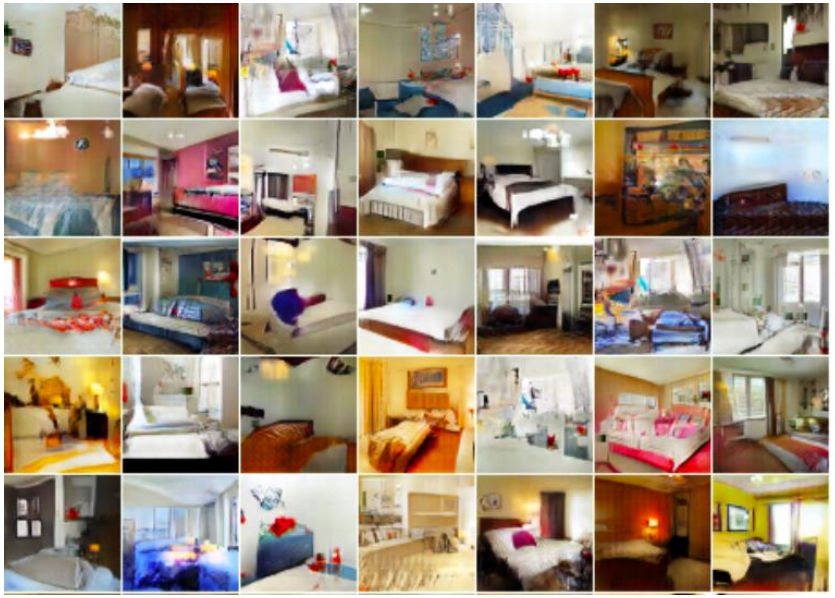
- Generator: $\min_G \mathbb{E}_{x \sim p_{z(model)}}\left[En(G(\boldsymbol{z}))\right]$



DCGAN vs EBGAN



DCGAN



EBIGAN

BEGAN

- Boundary-Equilibrium GAN (Barthelot2017) also uses the auto-encoder-based EBGAN for discriminator, but introduces the control parameter k_t to control the learning process

$$\text{Discriminator objective:} \quad \min_{En} En(x) - k_t En(G(z))$$

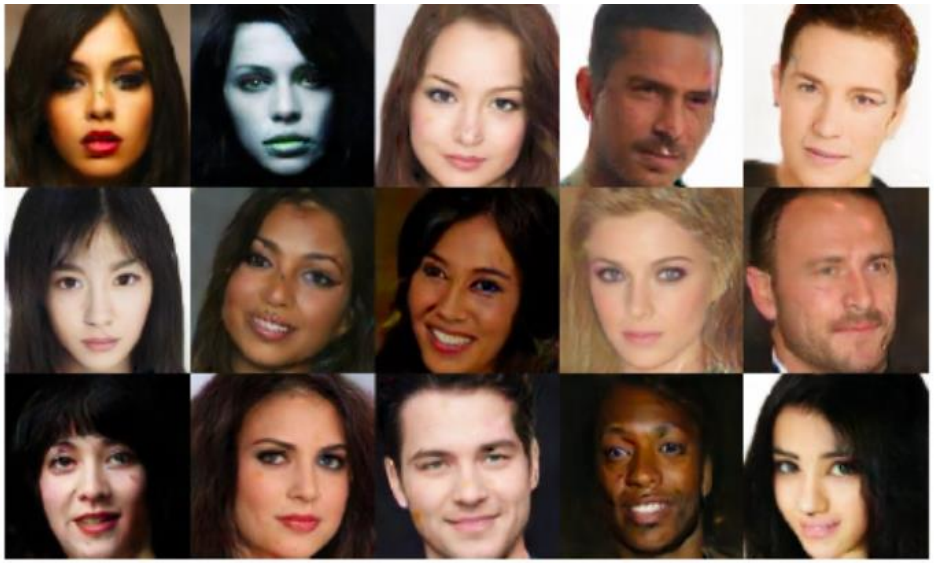
$$\text{Generator objective:} \quad \min_{G} En(G(z))$$

$$\text{Equilibrium objective:} \quad k_{t+1} = k_t + \lambda_k(\gamma En(x) - En(G(z)))$$

- γ is diversity ratio [0,1] and k is a tradeoff parameter which starts from zero, then gradually increases

EBGAN vs BEGAN

(a) EBGAN (64x64)

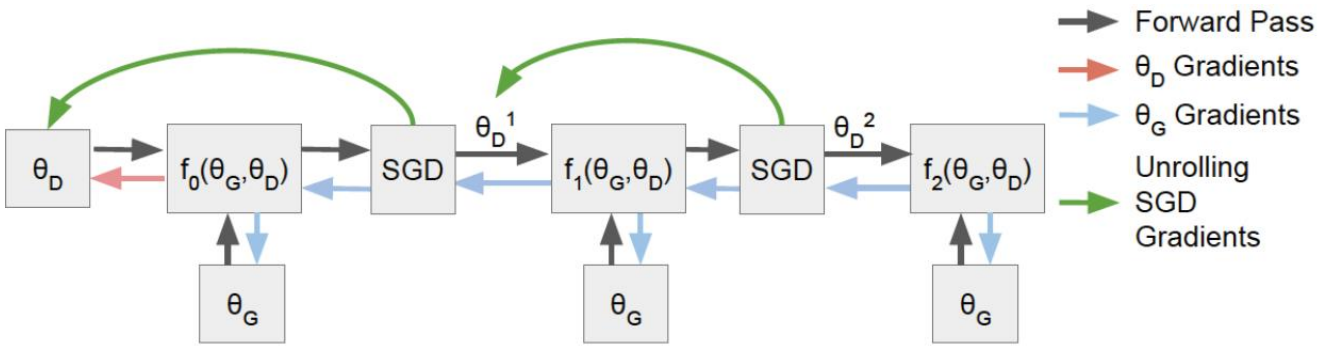


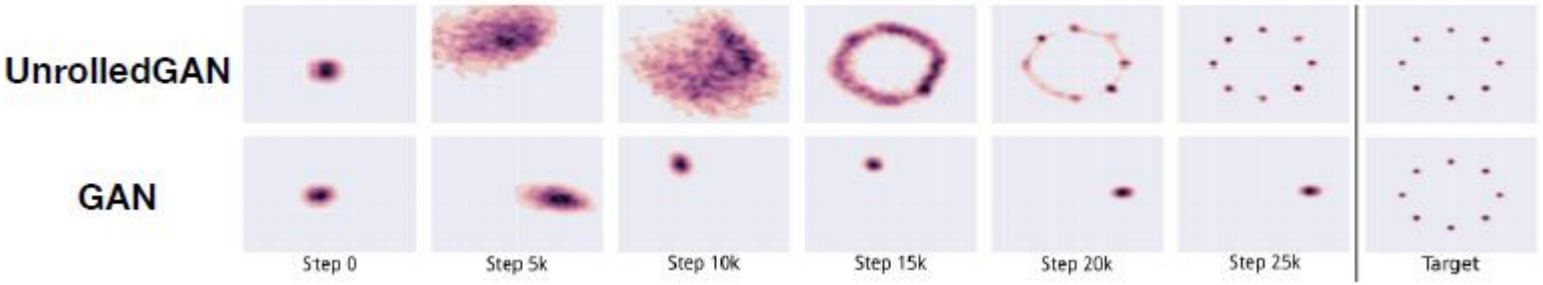
(b) Our results (128x128)

Unrolled GAN

- Unrolled GAN (Metz2017) considers several future updates of the discriminator as updating the generator
- Update the discriminator in the same way
- Avoid that the generator generates samples from few modes

$$f_K(\theta_G, \theta_D^{(K)}) = \mathbb{E}_{x \sim p_{data}} \left[\log(D(x; \theta_D^{(K)})) \right] + \mathbb{E}_{z \sim \mathcal{N}(0,I)} \left[\log(1 - D(G(z; \theta_G); \theta_D^{(K)})) \right]$$





DRAGAN

- DRAGAN (Kodali2017) is similar with WGAN-GP. The difference is where the discriminator function calculated by the critic network is gradient-constrained as $|f(\theta) = 1|$

WGAN-GP

$$\min_G \max_D E_{x\sim p_X}[D(x)] - E_{z\sim p_Z}[D(G(Z))] + \lambda E_{y\sim p_Y}[(||\nabla_y D(y)||_2 - 1)^2]$$

$$y = ux + (1-u)G(z)$$

- *y:* imaginary samples

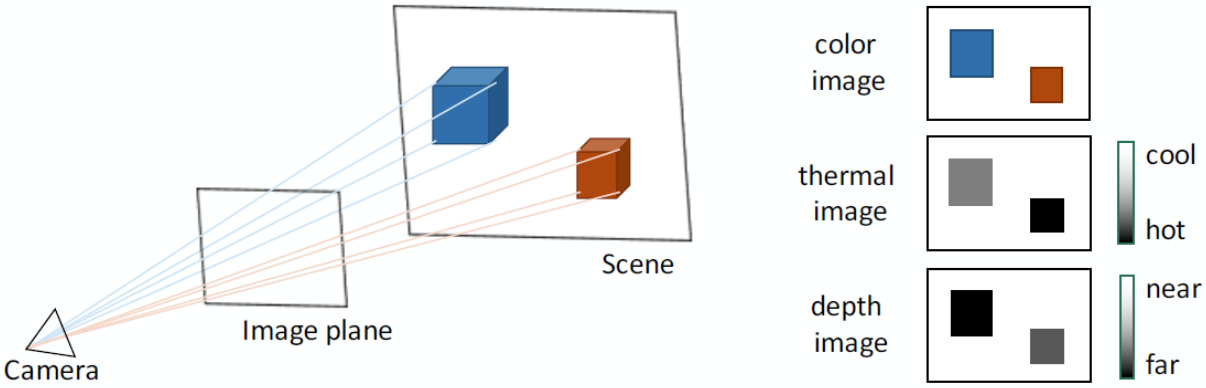
DRAGAN

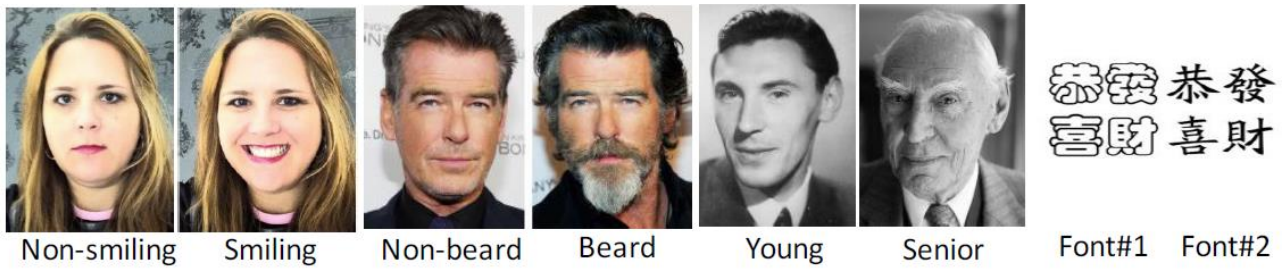
$$\min_G \max_D E_{x\sim p_X}[\log D(x)] - E_{z\sim p_Z}[\log(1 - D(G(Z)))] + \lambda E_{y\sim p_Y}[(||\nabla_y D(y)||_2 - 1)^2]$$

$$y = \alpha x + (1-\alpha)(x+\delta)$$ *y:* imaginary samples around true sample

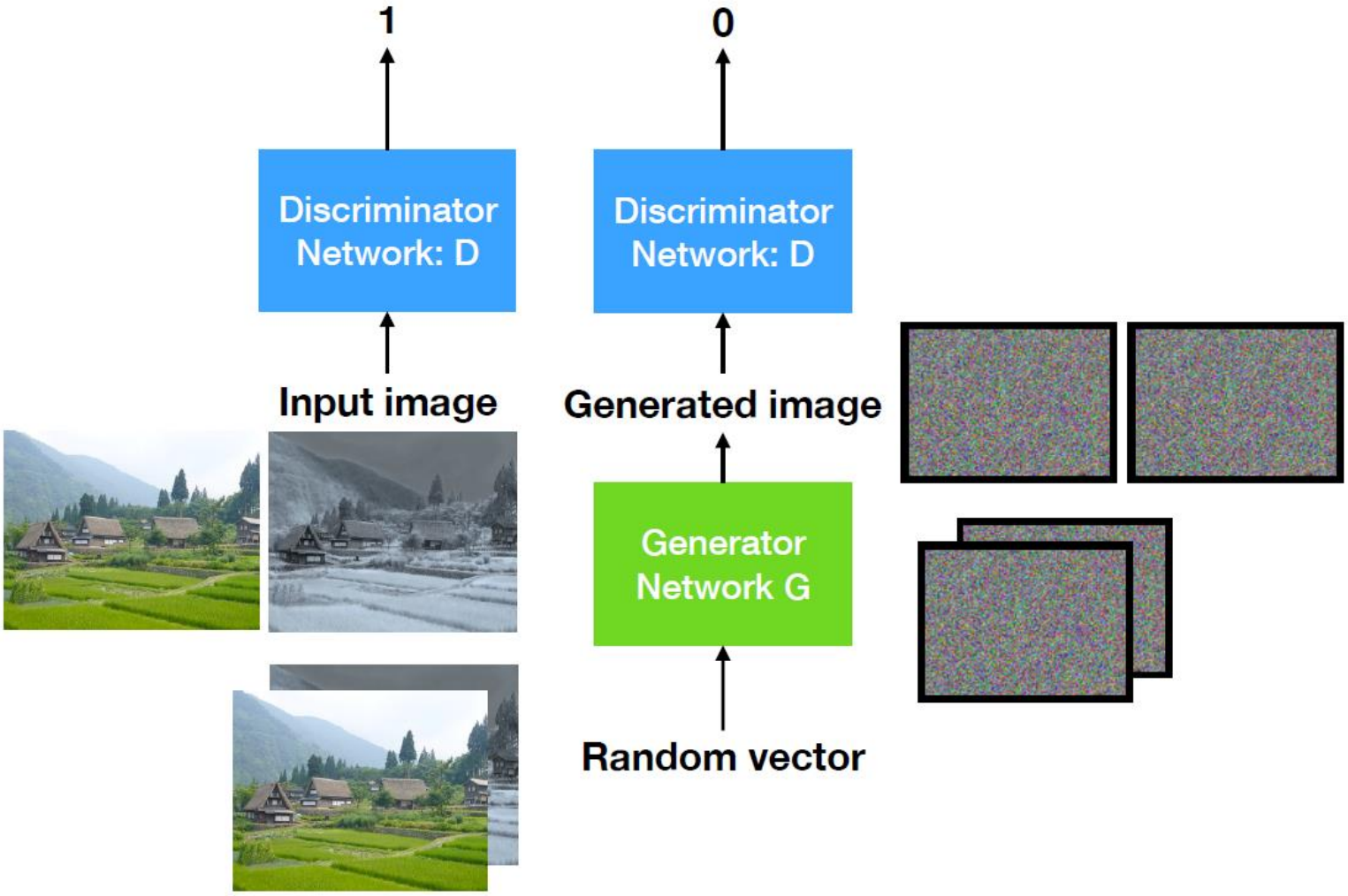
Joint-Distribution of Multi-domain of Images

- Multi-domain images are views of an object with different attributes



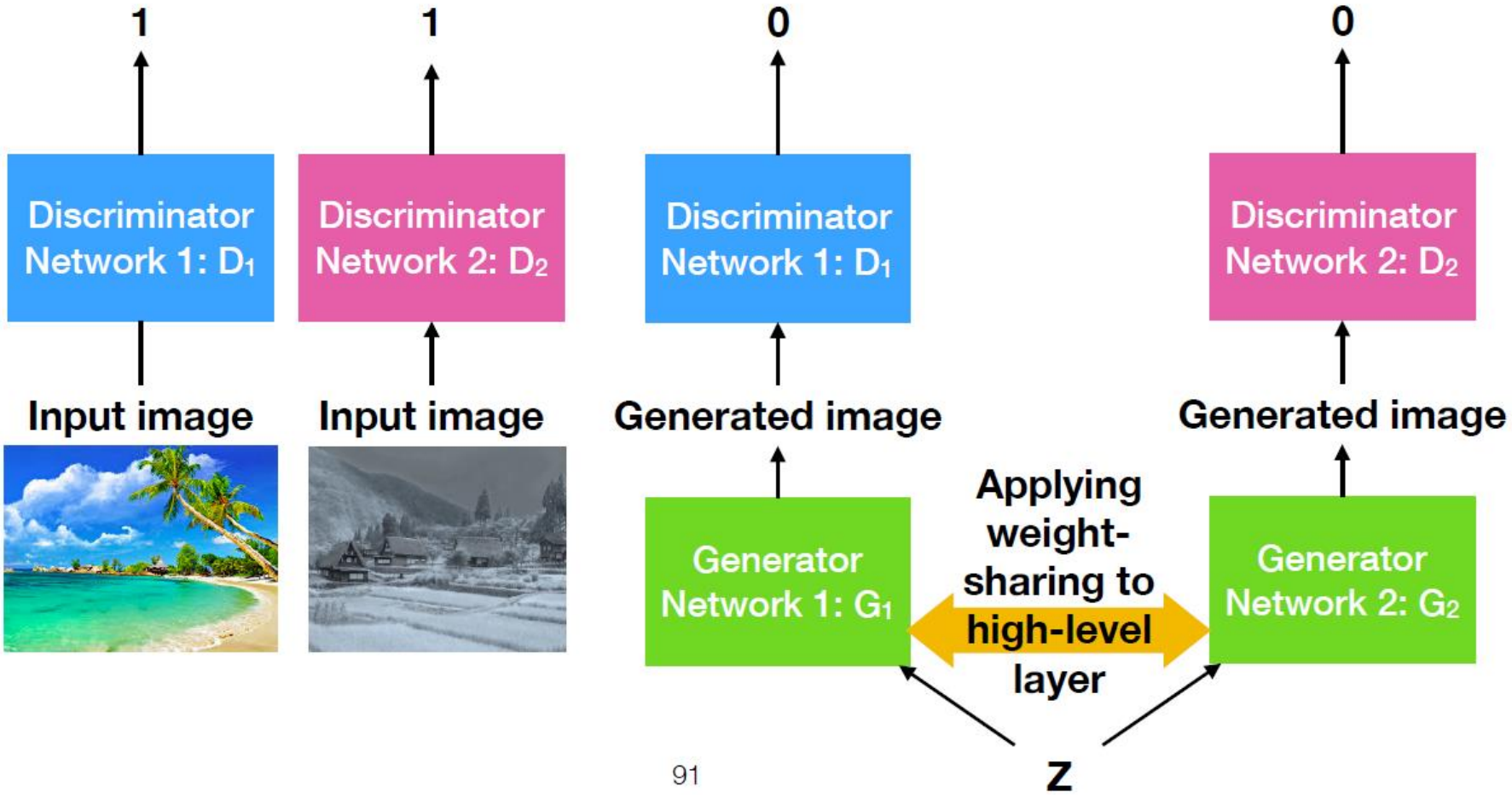


Extending GAN for Joint Image Distribution Learning



CoGAN

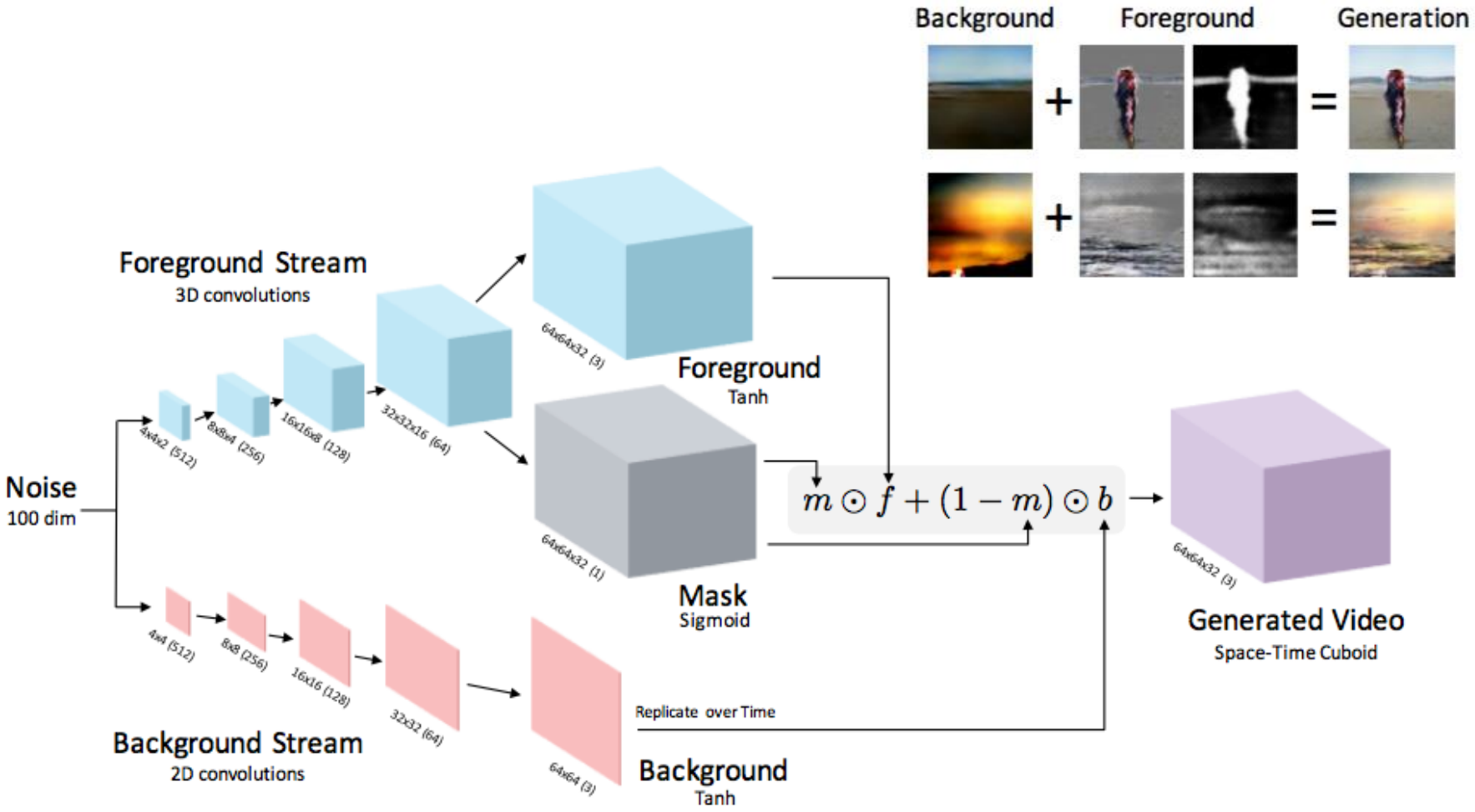
- Coupled Generative Adversarial Networks (CoGAN, Liu2016) inputs the *unpaired* images to train GAN for joint image distribution learning



CoGAN

VideoGAN

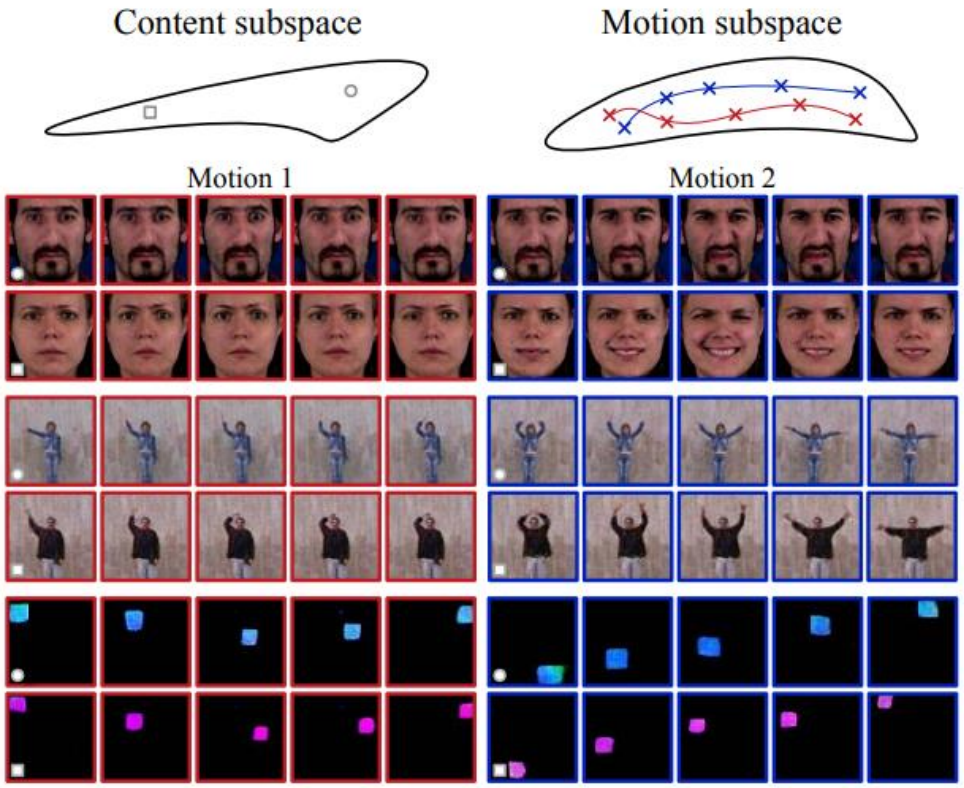
- Video GAN (VGAN; Vondorick2016) generates a video whose inter-motion is reasonably natural

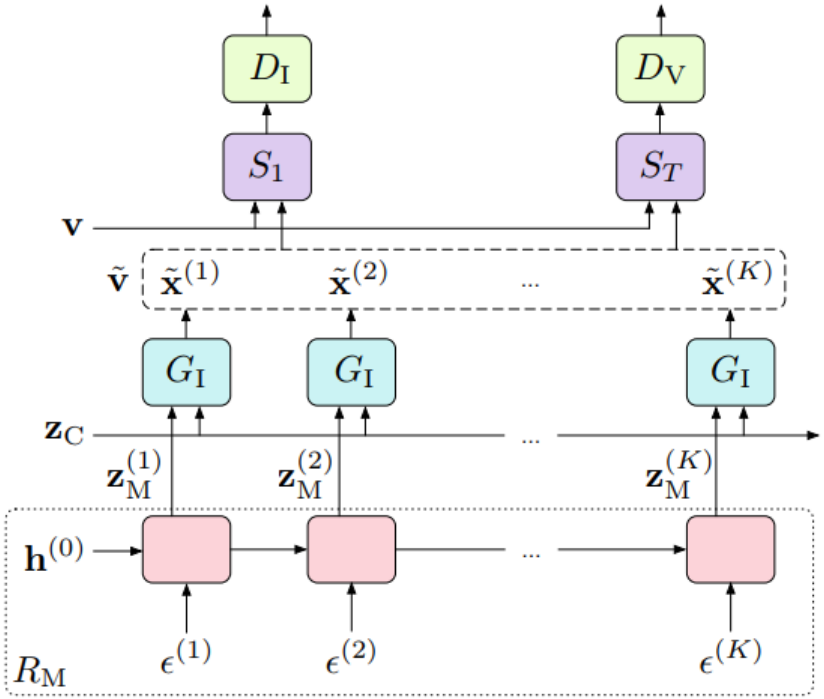


VideoGAN

MoCoGAN

- Motion and Content GAN (MoCoGAN; Tulyakov2017) treats a video with a combination of motion and content and generates frames one by one which are temporally consistent instead of a video (high quality + variable length of a video)





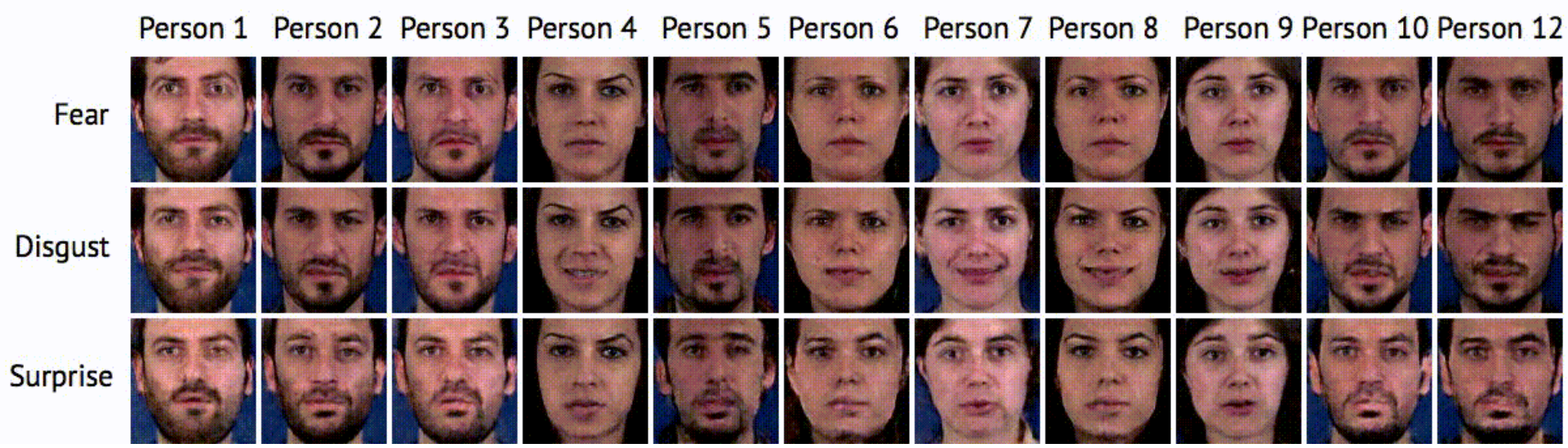
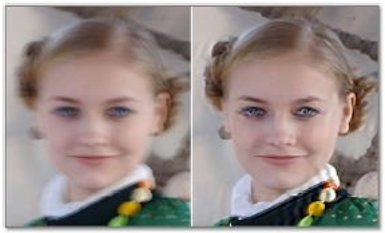

Person 1
Person 2
Person 3
Person 4
Person 5
Person 6
Person 7
Person 8
Person 9
Person 10
Person 12
Fear
Disgust
Surprise

Image-to-Image Translation

Low-res to high-res



Blurry to sharp

Thermal to color

Synthetic to real

LDR to HDR

Noisy to clean

Image to painting

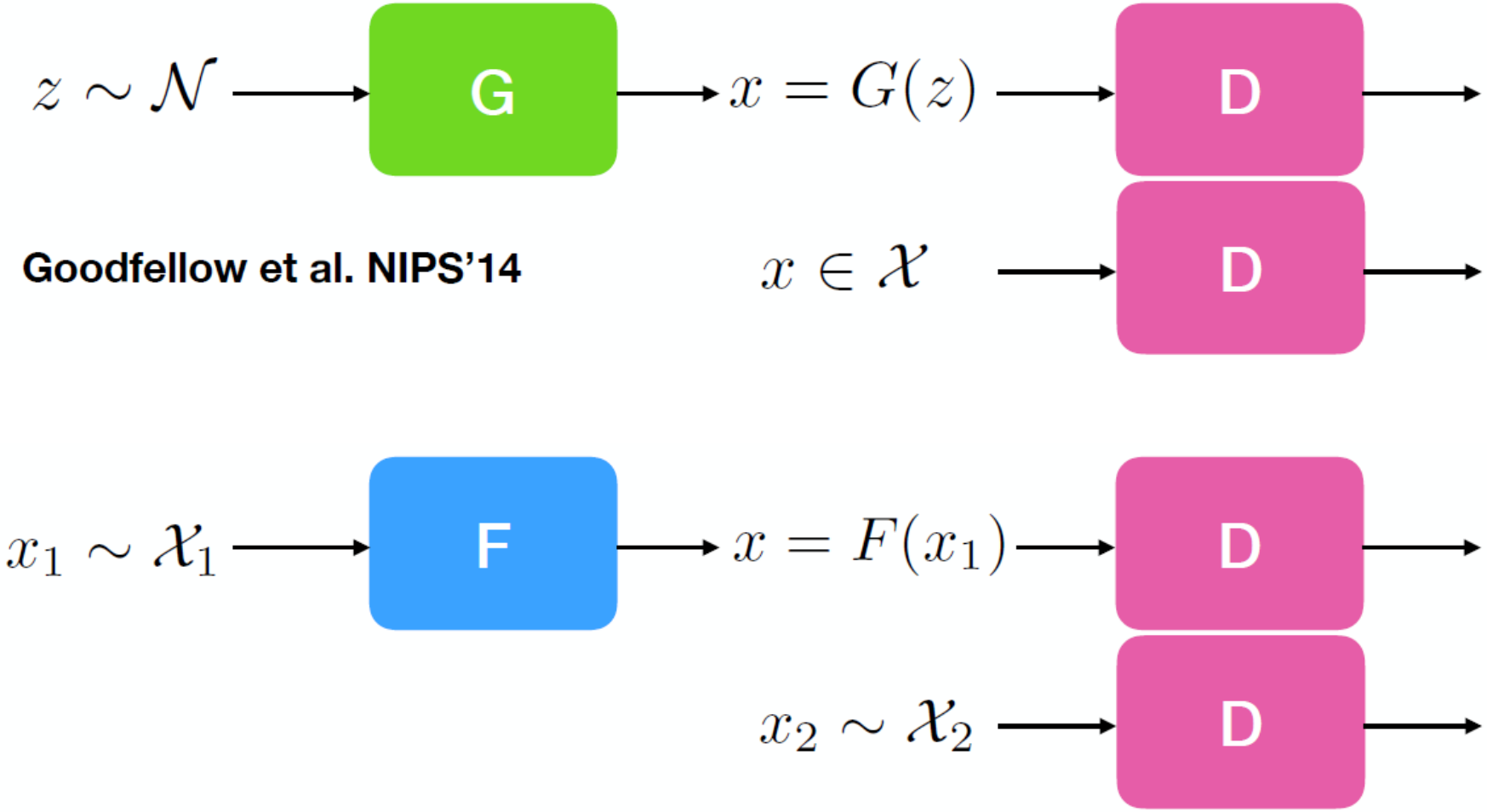
Day to night

Summer to winter

- Bad weather to good weather
- Greyscale to color
- …

Conditional GAN

- Conditional GAN inputs specific values instead of a random value to generate the preferable pairs of input/output (firstly proposed in super-resolution work by Ledig2017)

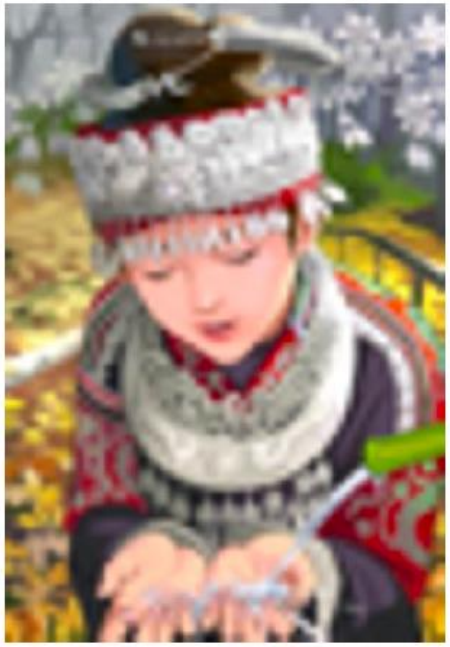


SRGAN

- SRGAN (Super-resolution GAN, Ledig2017) upsamples a low-res image without blurry artifacts appeared in images recovered by previous model-based approaches. The methods relates input-output by a content loss

$$||x - x_2^{(i)}||_2 + ||\mathrm{VGG}(x) - \mathrm{VGG}(x_2^{(i)})||_2$$ Content loss

bicubic (21.59dB/0.6423)



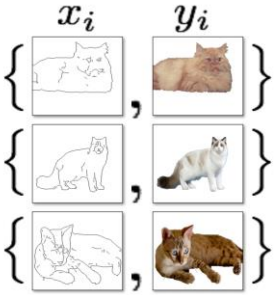
SRResNet (23.53dB/0.7832)

SRGAN (21.15dB/0.6868)

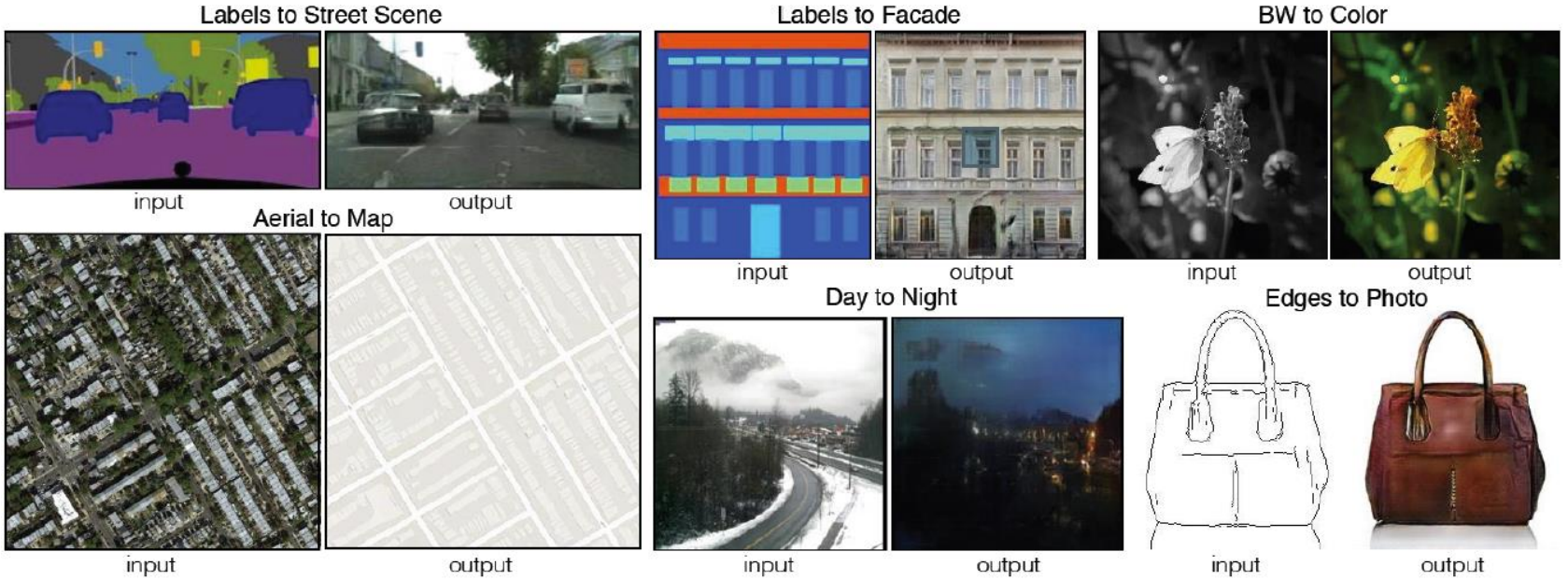
original

Pix-to-Pix

- Pix2pix (Isola2017) extends the conditional GAN to general tasks by learning the joint distribution of input and generated image
- The network is trained based on ground truth pairs of the input and output



$$\max_F E_{p_{\mathcal{X}_1}} [\log(D(x_1, F(x_1)))]$$

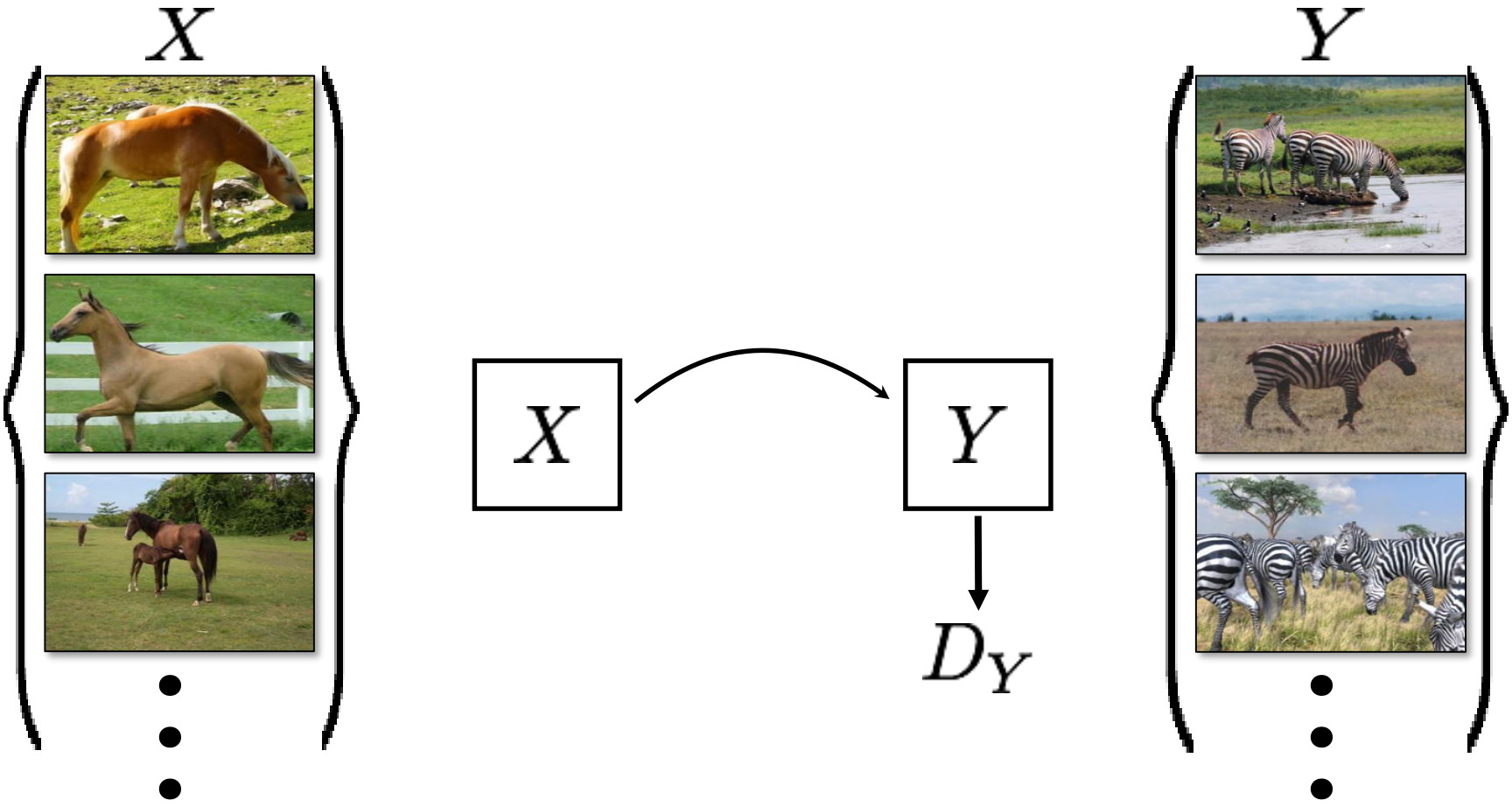


Interactive Demo

https://affinelayer.com/pixsrv/

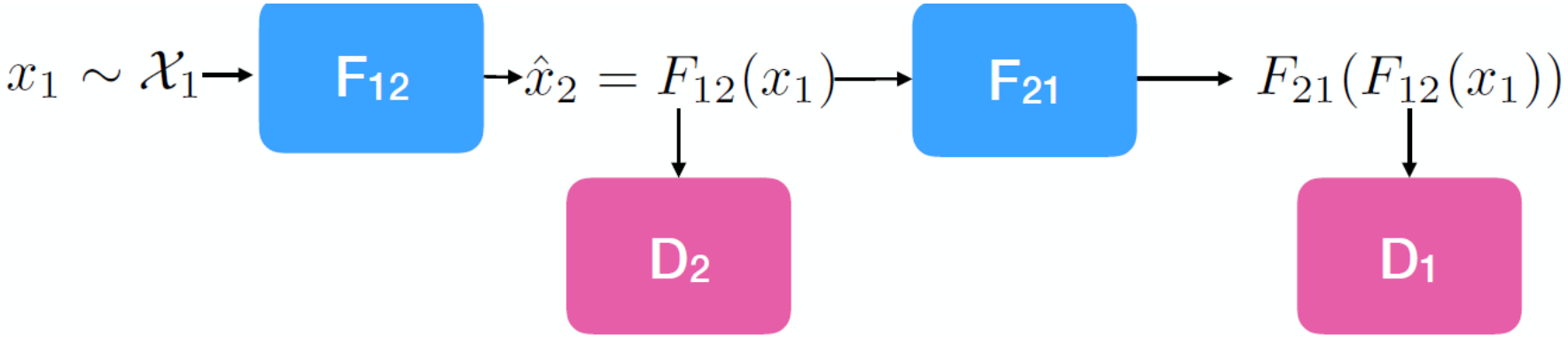
CycleGAN

- CycleGAN (Zhu2017) or DiscoGAN (Kim2017) train the conditional GAN without pair of input/output images



CycleGAN

- CycleGAN minimizes the cycle consistency loss



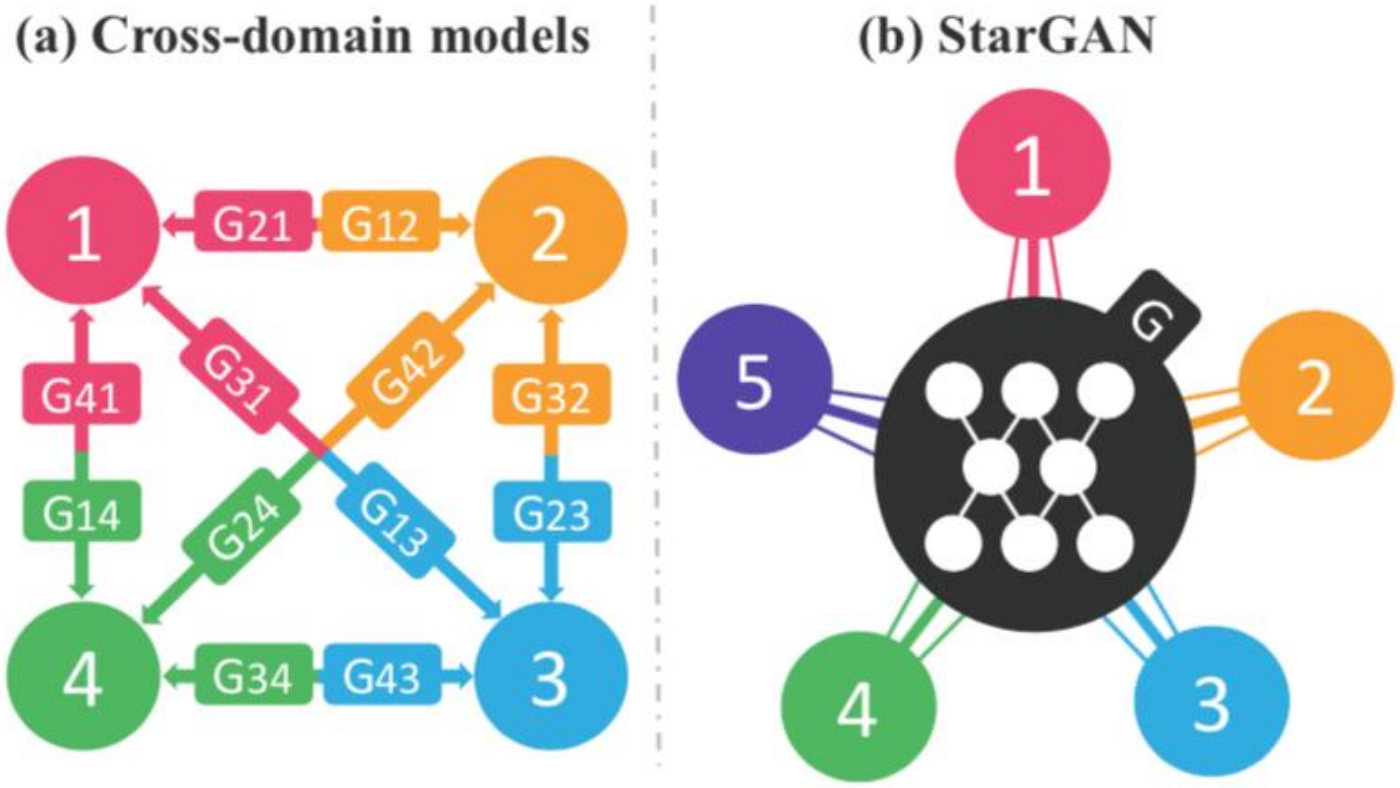
$$\max_{F_{12},F_{21}} E_{p\mathcal{X}_1}[\log(D_2(F_{12}(x_1)) - \lambda||F_{21}(F_{12}(x_1)) - x_1||_p^p] +$$
$$E_{p\mathcal{X}_2}[\log(D_1(F_{21}(x_2)) - \lambda||F_{12}(F_{21}(x_2)) - x_2||_p^p]$$

Results

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

StarGAN

- StarGAN (Choi2018) learns the domain transfer network with a *single* generator/discriminator by paring the domain/image for generating samples



StarGAN

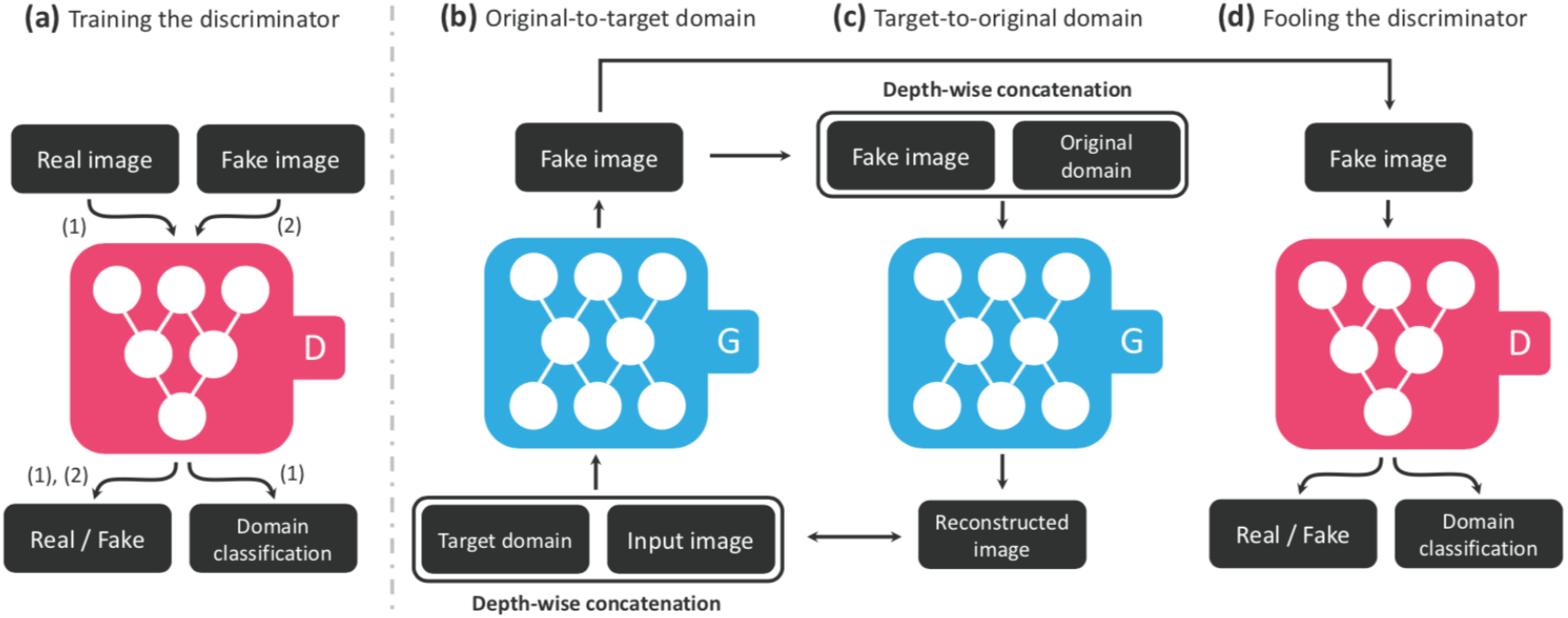


Figure 3. Overview of StarGAN, consisting of two modules, a discriminator D and a generator G. **(a)** D learns to distinguish between real and fake images and classify the real images to its corresponding domain. **(b)** G takes in as input both the image and target domain label and generates an fake image. The target domain label is spatially replicated and concatenated with the input image. **(c)** G tries to reconstruct the original image from the fake image given the original domain label. **(d)** G tries to generate images indistinguishable from real images and classifiable as target domain by D.

Results

Endnotes

- Uncovered topics: machine learning with text (e.g., captioning, visual QA, Deep Reinforcement Learning (e.g., AlphaGO)), recurrent neural net, 3-D CNN and so on

- **Signal Processing part (by Prof. Kodama) starts at 1/15 (Tue)**

- **Final report (Machine Learning part):**

  Summarize and discuss a machine learning paper which was not mentioned in the lecture, was published in 2017 or 2018, and was cited by more than 100 papers **(A4 2pages maximum, deadline will be announced by Prof. Kodama)**

  - E.g.) Why do you think the paper was cited by many papers? What is the importance?
  - E.g.) Any suggestions to improve the result?
  - *Please freely describe your idea rather than just summarize it*!