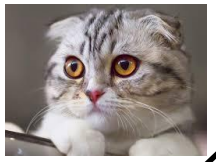
Generative Adversarial Networks (GANs)

Jenq-Neng Hwang, **Professor**
Department of Electrical & Computer Engineering
University of Washington, Seattle WA
hwang@uw.edu

A Generative Model

Discriminative model (e.g., CART, SVM, CNN)

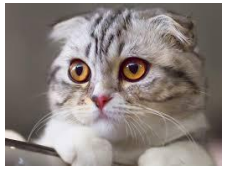


decision boundary

Posterior prob P(Y= cat| X= ) = 0.9

Generative model (e.g., mixture of Gaussians, HMM)

Y= cat, X= 

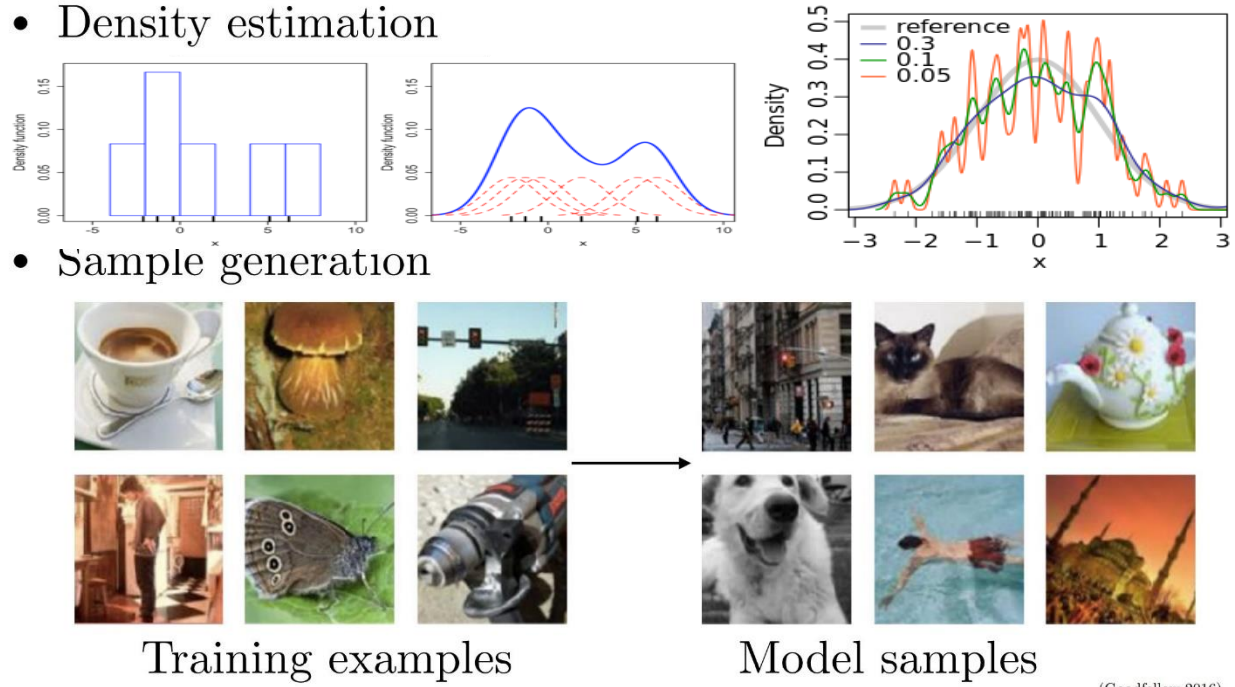
Y= cat, X= 

Y= dog, X= 

likelihood P(X=  | Y=dog)

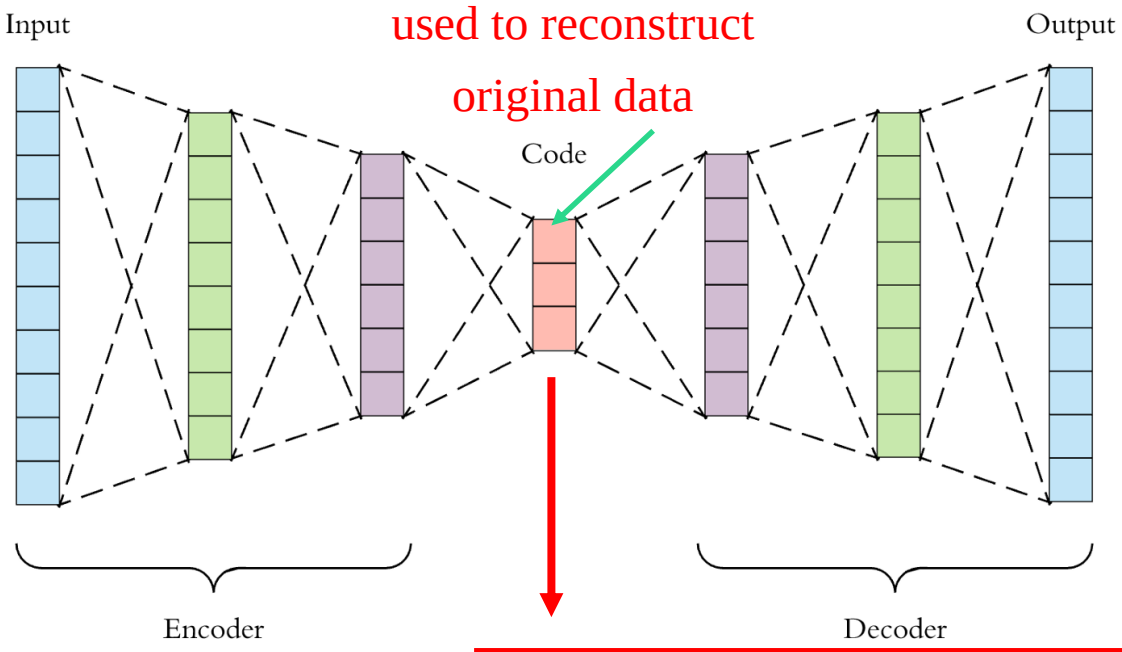
A Generative Model

- Can we learn *P(x/y)* from a set of (image) data sampled from class *y*? So that we can generate some representative data (image) ***x*** from *P(**x**/y)*?

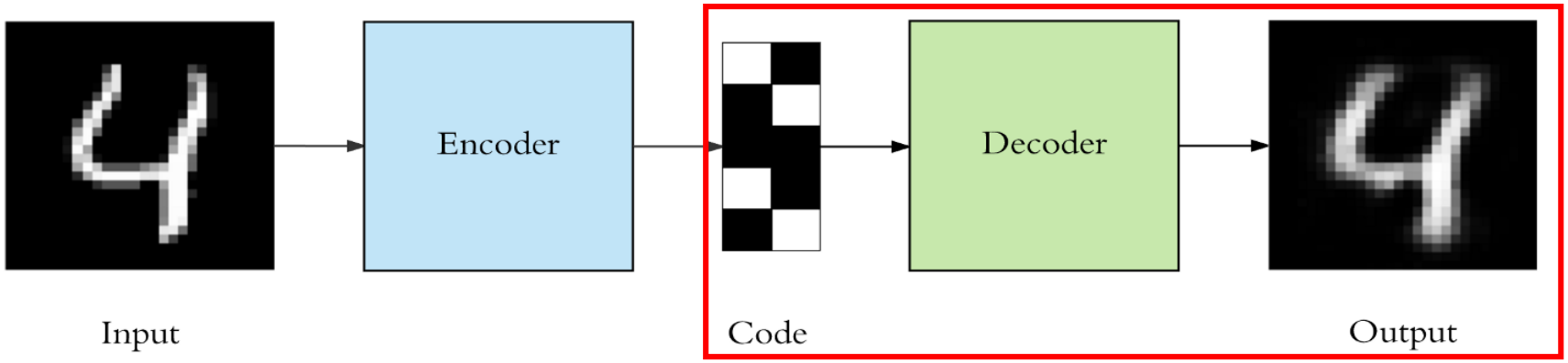


Auto-Encoder (AE)

output is the same as input (regression)

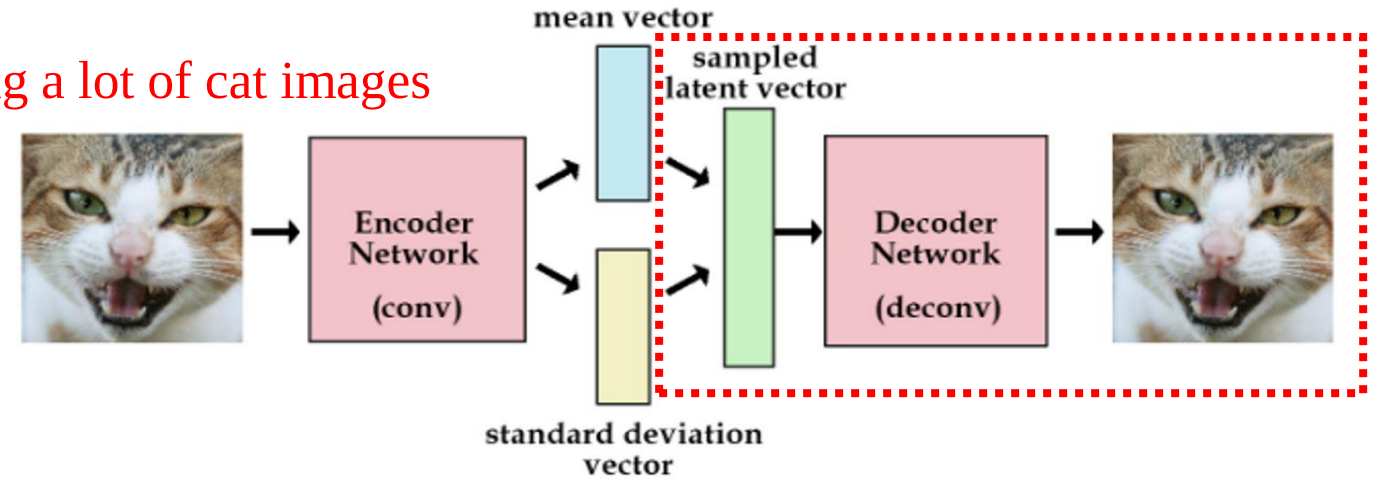


Training loss: mean squared error or binary cross-entropy

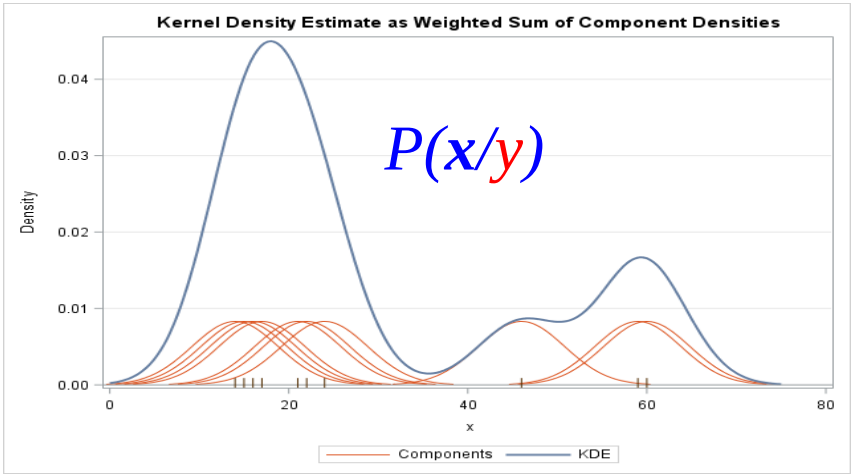


Variational Auto-Encoder (VAE)

Training a lot of cat images

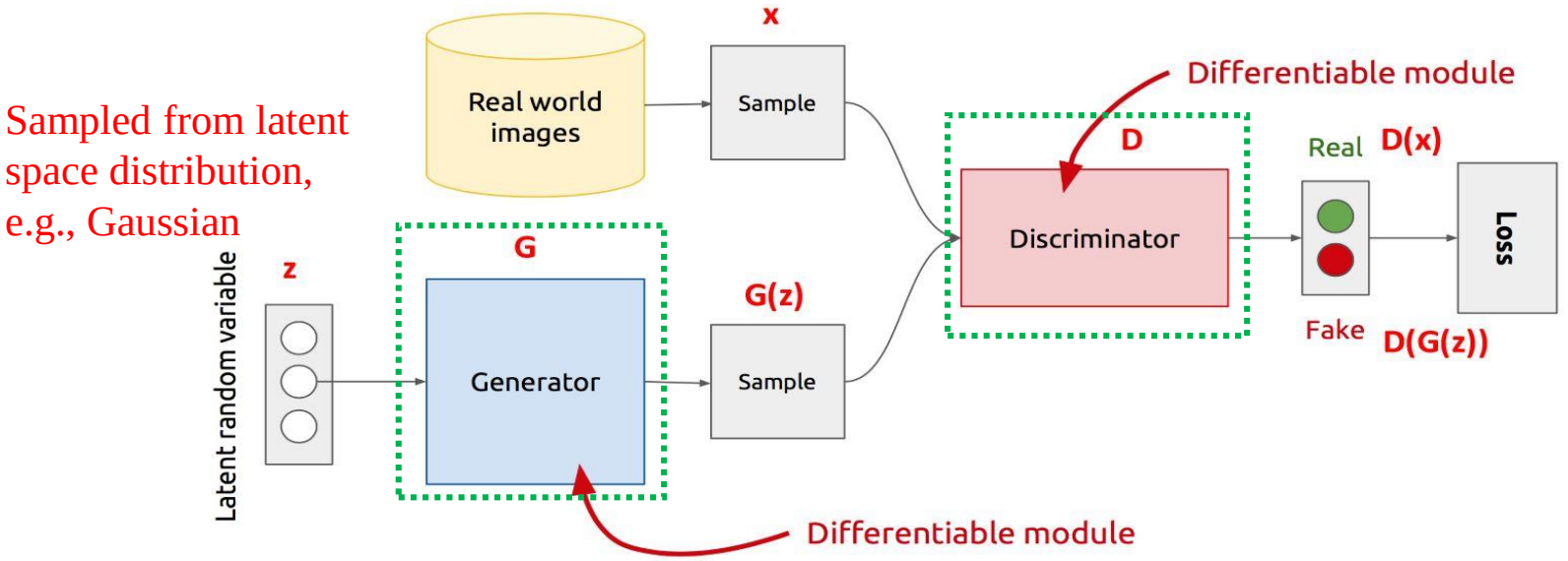


- An unsupervised learning mechanism to extract useful latent vector (embedding code) for data regeneration



A Generative Adversarial Network (GAN)

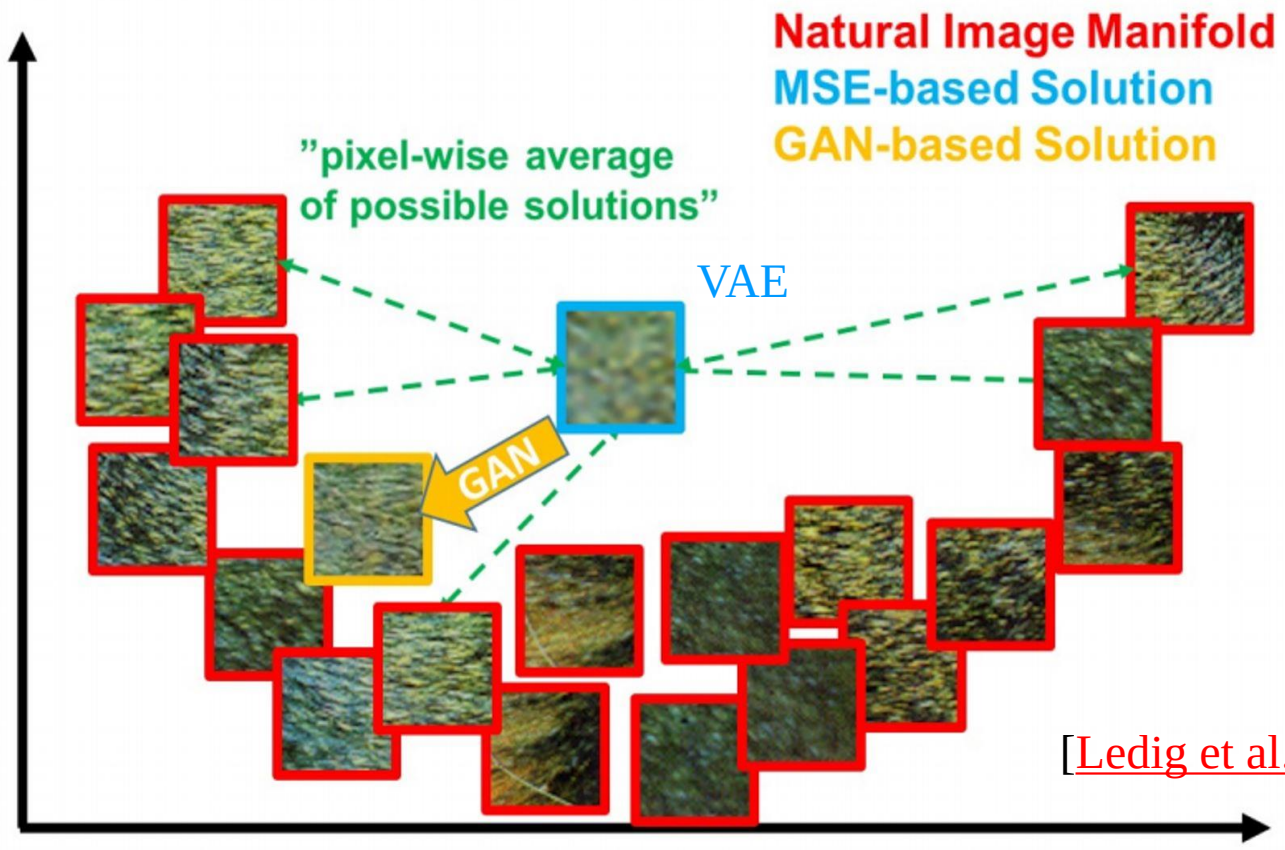
- **Discriminator network**: try to distinguish between real and fake images
- **Generator network**: try to fool the discriminator by generating real-looking images



Ian Goodfellow, et al. "Generative adversarial networks," NIPS'2014 .

Difference between VAE and GAN

- Learning of $P(x/y)$



[Ledig et al. CVPR 2017]

Progresses of GANs

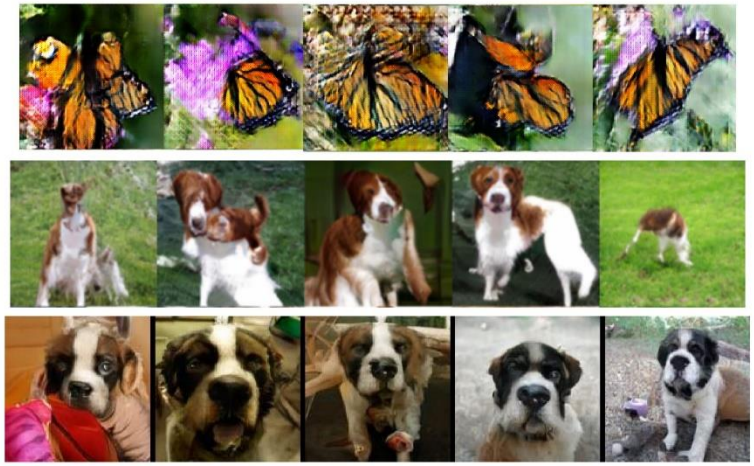
Given training data, generate new samples from same distribution

Training data ~ $p_{data}(x)$

Odena et al 2016

Miyato et al 2017

Zhang et al 2018

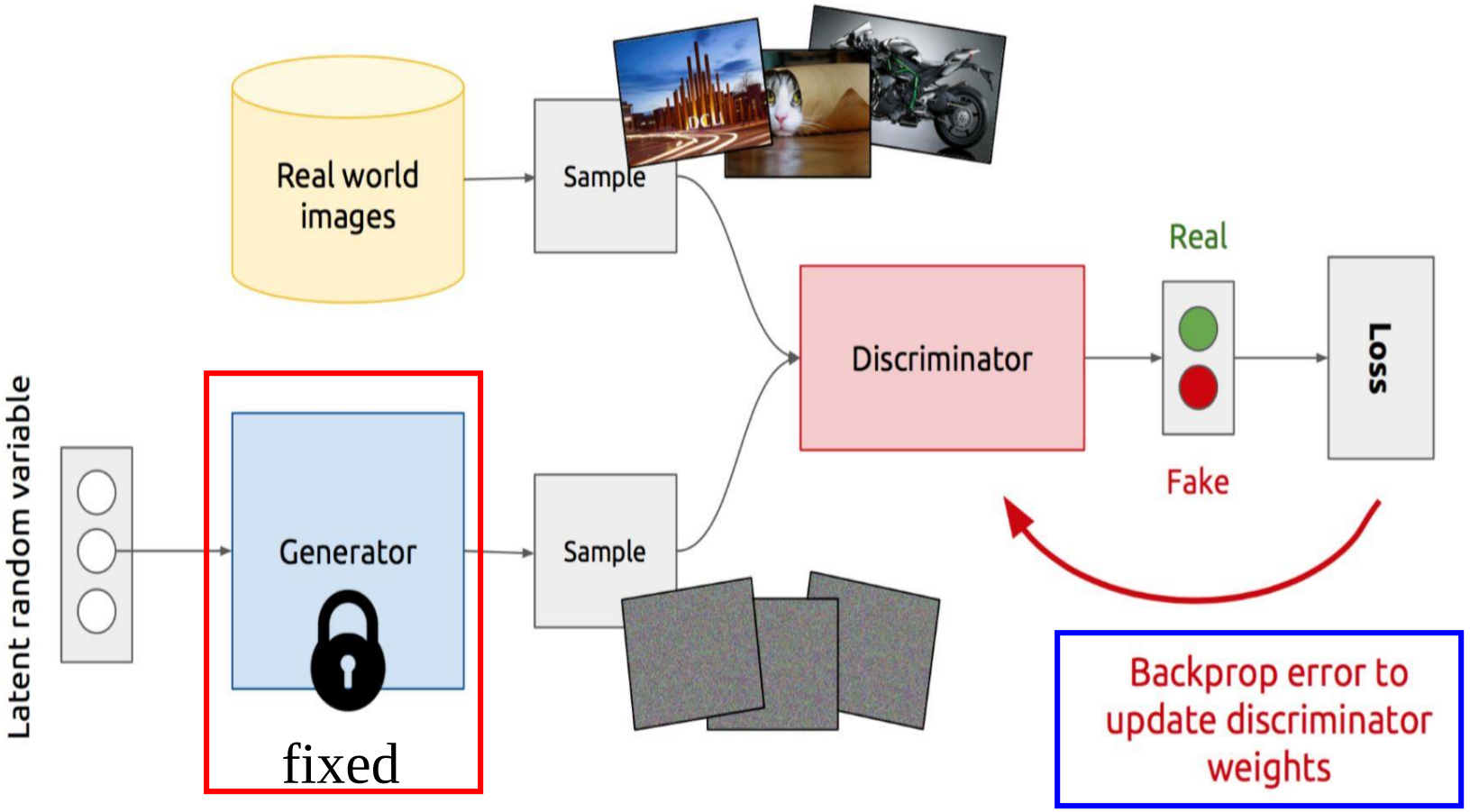


(Goodfellow 2018)

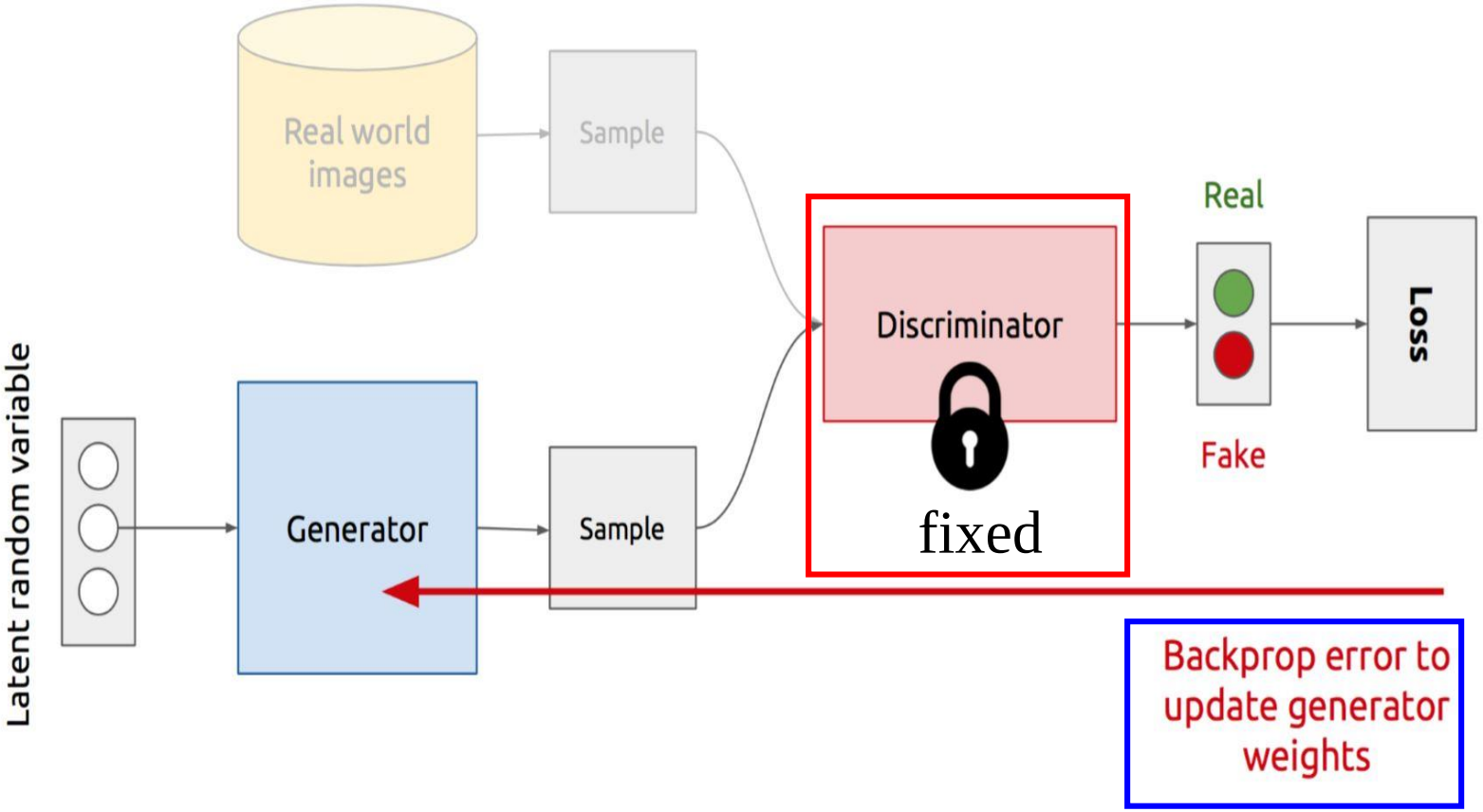
Generated samples ~ $p_{model}(x)$

Want to learn $p_{model}(x)$ similar to $p_{data}(x)$

Training A Discriminator



Training A Generator



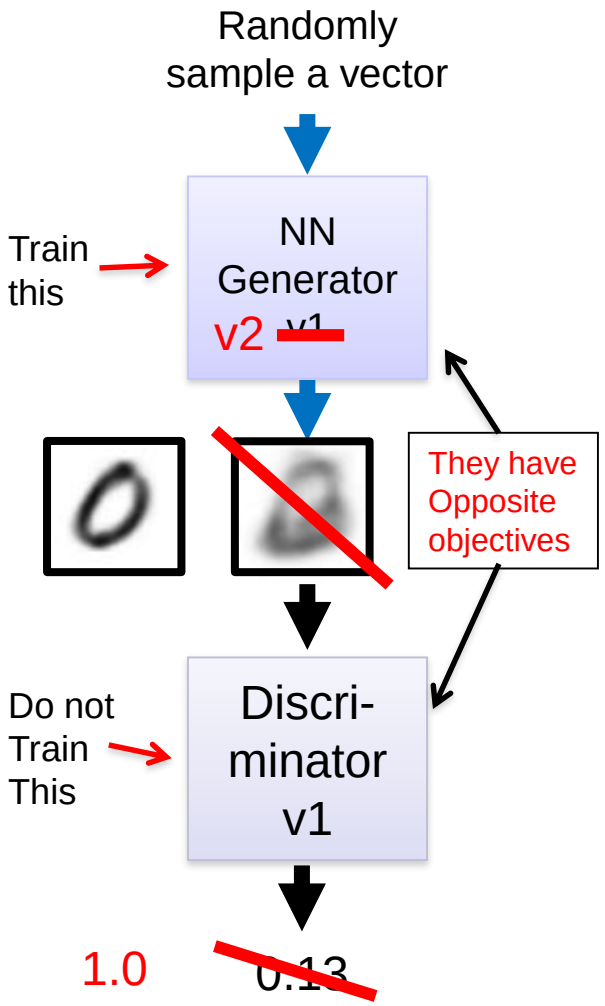
GAN: Training A Generator

Updating the parameters of generator

The output be classified as "real" (as close to 1 as possible)

Generator + Discriminator = a network

Using gradient descent (backpropagation) to update the parameters in the Generator, but fix the Discriminator



Alternate Training A GAN

cross-entropy loss

$$E = -\sum_{n=1}^{N} t^{(n)} \log o^{(n)} + (1 - t^{(n)}) \log(1 - o^{(n)}) \qquad t^{(n)}=[0,1]$$

$$o^{(n)} = (1 + \exp(-z^{(n)})^{-1}$$

Alternate between:

1. **Gradient ascent** on discriminator (the generator is fixed)

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

2. **Gradient descent** on generator (the discriminator is fixed)

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

Generator minimizes the log-probability of the discriminator being correct

- Discriminator (θd) wants to **maximize objective** such that *D(x)* is close to 1 (real) and *D(G(z))* is close to 0 (fake)
- Generator (θg) wants to **minimize objective** such that *D(G(z))* is close to 1 (discriminator is fooled into thinking generated G(z) is real)

Simple Illustration of GAN

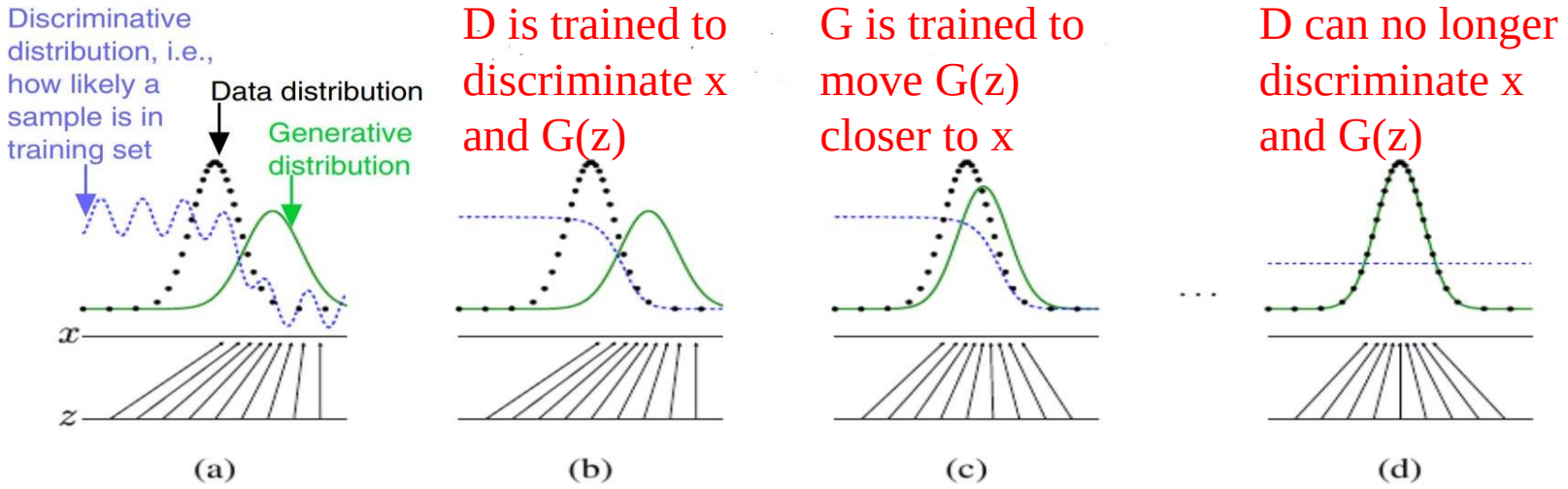


Figure 1: Generative adversarial nets are trained by simultaneously updating the **d**iscriminative distribution (D, blue, dashed line) so that it discriminates between samples from the data generating distribution (black, dotted line) $p_{\boldsymbol{x}}$ from those of the **g**enerative distribution p_g (G) (green, solid line). The lower horizontal line is the domain from which $\boldsymbol{z}$ is sampled, in this case uniformly. The horizontal line above is part of the domain of $\boldsymbol{x}$. The upward arrows show how the mapping $\boldsymbol{x} = G(\boldsymbol{z})$ imposes the non-uniform distribution p_g on transformed samples. G contracts in regions of high density and expands in regions of low density of p_g. (a) Consider an adversarial pair near convergence: p_g is similar to p_{data} and D is a partially accurate classifier. (b) In the inner loop of the algorithm D is trained to discriminate samples from data, converging to $D^*(\boldsymbol{x}) = \frac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}$. (c) After an update to G, gradient of D has guided $G(\boldsymbol{z})$ to flow to regions that are more likely to be classified as data. (d) After several steps of training, if G and D have enough capacity, they will reach a point at which both cannot improve because $p_g = p_{\text{data}}$. The discriminator is unable to differentiate between the two distributions, i.e. $D(\boldsymbol{x}) = \frac{1}{2}$.

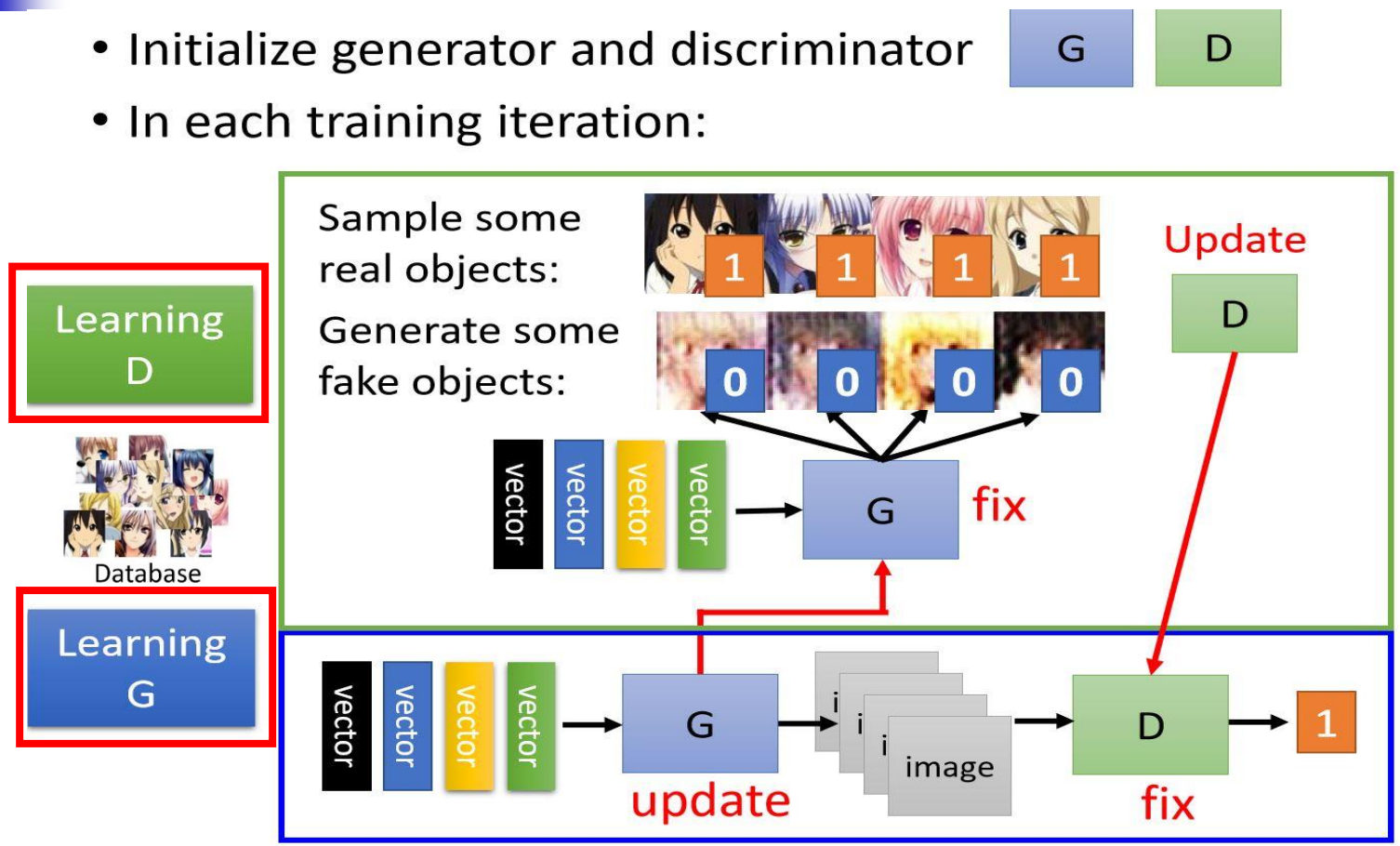
Anime Face Dataset

- A dataset of 21,551 anime faces scraped from www.getchu.com, which are then cropped and resized to 64 * 64 (https://www.kaggle.com/soumikrakshit/anime-faces)

GAN generated anime faces

Anime Faces Generation

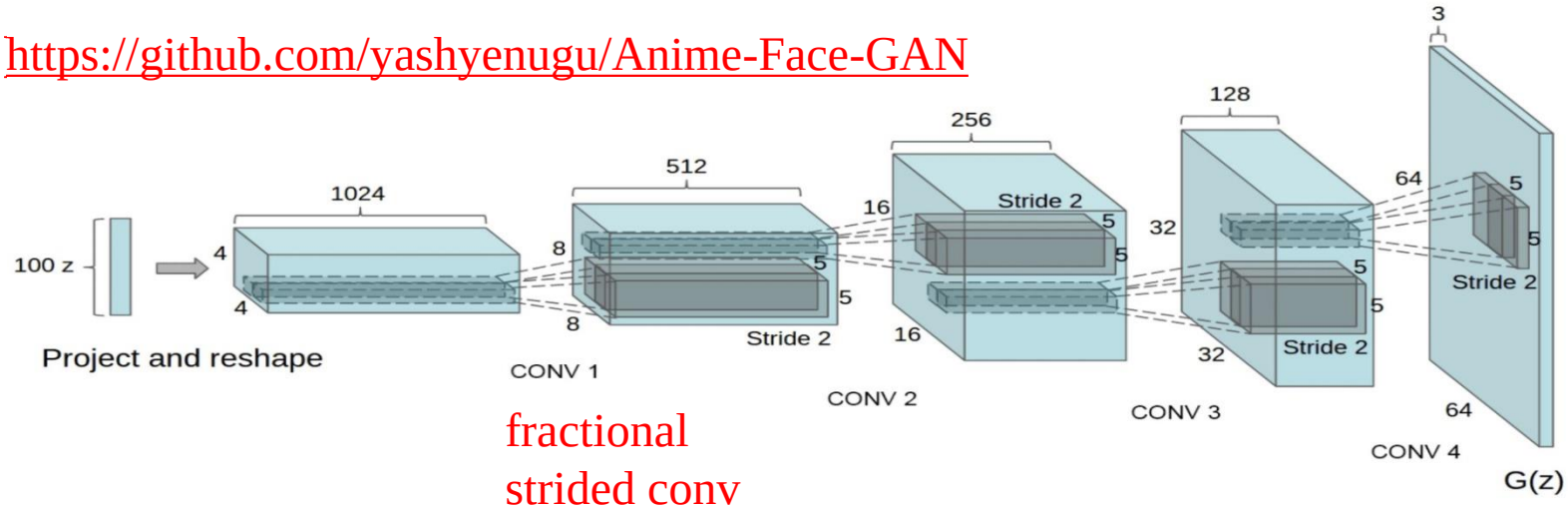
- Initialize generator and discriminator
- In each training iteration:



https://github.com/yashyenugu/Anime-Face-GAN

CNN based GAN Generator

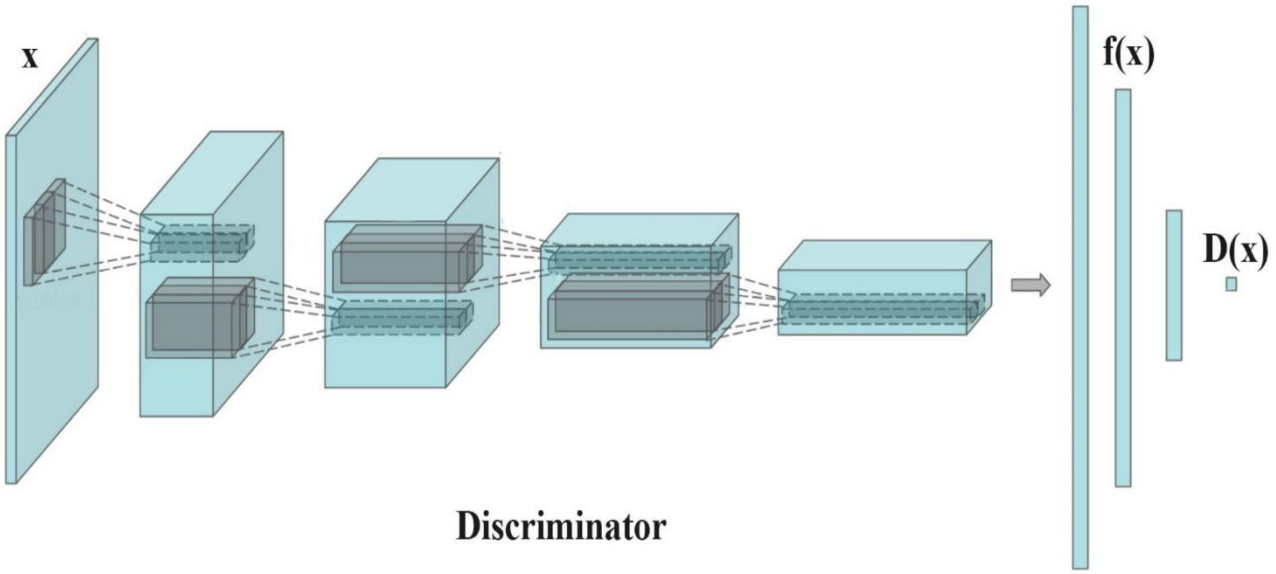
- Generator: a 100-dim (or 128-dim) noise vector, normal distribution *N(0,1)*. It is then followed by a Dense layer (fully connected) and reshaped to (4,4,1024).
- A few (fractional strided) convolutional layers are followed which then results in an image of size (64,64,3) with pixel values of the range [-1,1] due to a tanh activation.
- https://github.com/yashyenugu/Anime-Face-GAN

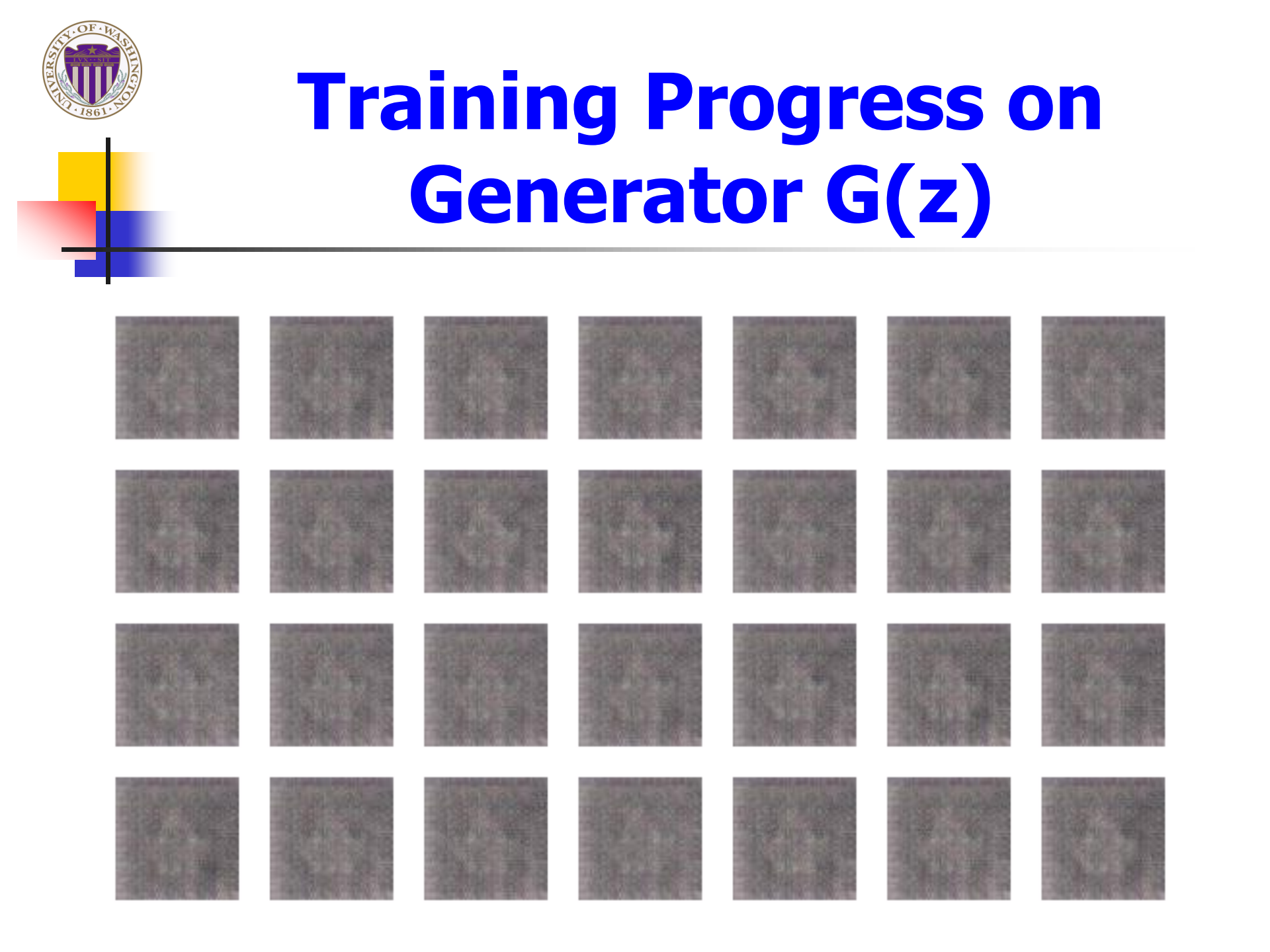


fractional strided conv

CNN based Discriminator

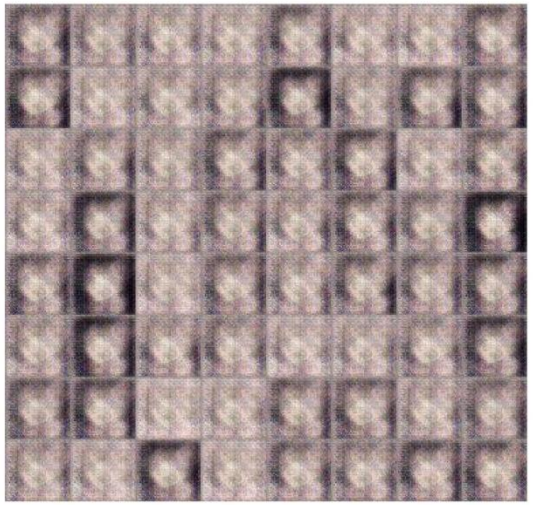
- Dsicriminator can be any image classifier architecture with one output
- *f(x)* is the feature vector extracted in an immediate layer by the discriminator.



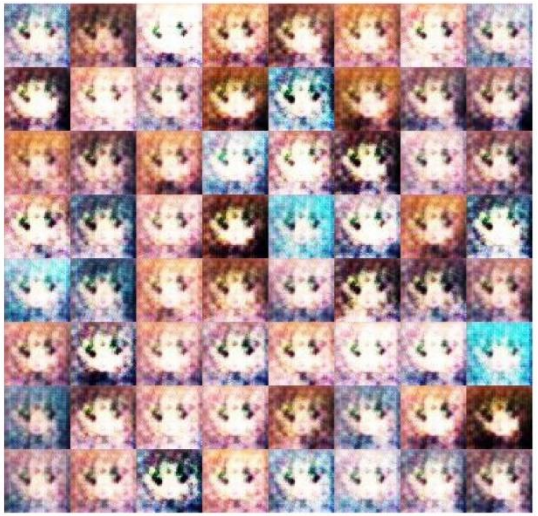

Training Progress on
Generator G(z)

Generated Anime Images

100 updates



1000 updates



5000 updates

50,000 updates

Improving Convergence of GAN Training

- Feature Matching: train the generator to minimize L2 distance of the expected features on an intermediate layer of the discriminator

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

- Virtual Batch Normalization: each example *x* is normalized based on the statistics collected on a reference batch of examples that are chosen once and fixed at the start of training, and on *x* itself.
- One Sided Label Smoothing: replaces the positive targets for a classifier with smoothed values, like .9, shown to reduce the vulnerability of neural networks to adversarial examples, discourages the discriminator from being overconfident about its classification

Tim Salimans, et al., "Improved Techniques for Training GANs," NIPS 2016

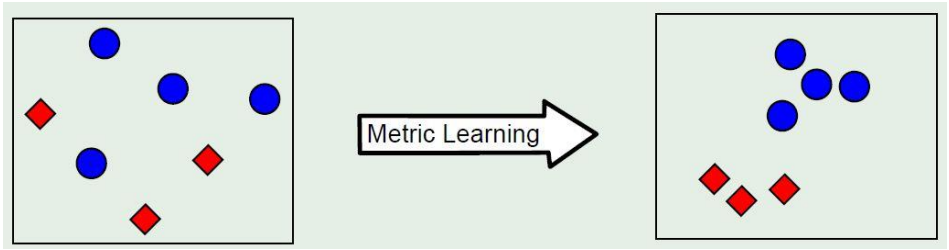
Distance Metric Learning (DML)

- We wish to find a linear (matrix $W \in R^{n \times m}$) transformation $y = W^T x$, so that the simple Euclidean distance in new space of y's can be used as a good distance measurement

$$D^2(x_i, x_j) = \| W^T x_i - W^T x_j \|^2 = \| W^T (x_i - x_j) \|^2$$
$$= (x_i - x_j)^T W W^T (x_i - x_j) = (x_i - x_j)^T A (x_i - x_j)$$

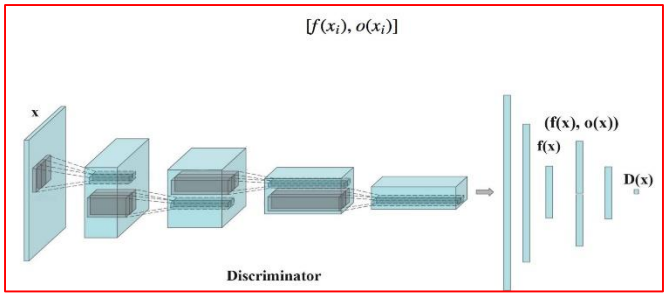
A: distance metric -- a positive semi-definite matrix

- Goal : keep ***all*** the data points within the same classes close, while separating ***all*** the data points from different classes.



Improving GAN Training Convergence by DML

- Mini-batch discrimination: Similarity ***o(x)*** in one of the dense layers in the discriminator is concatenated with $f(x_i)$, that were its input to tensor T, to classify whether this image is real or generated.



$c_b(x_i, x_j) = exp(-\|M_{i,b} - M_{j,b}\|_1) \in \mathbb{R}$

$o(x_i)_b = \sum_{j=1}^{n} c_b(x_i, x_j) \in \mathbb{R}$

$f(x_i) \in \mathbb{R}^A$

$f(x_j)$

$T \in \mathbb{R}^{A \times B \times C}$

$M_{i,b} \in \mathbb{R}^C$

$M_i = f(x_i)T \in \mathbb{R}^{B \times C}$

M_j

Σ

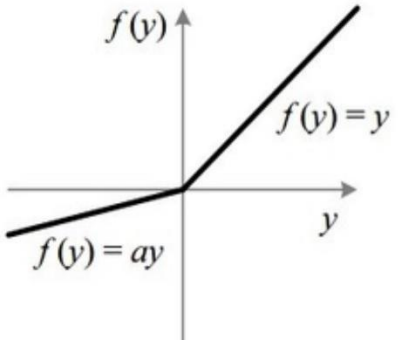
$o(x_i) = [o(x_i)_1, o(x_i)_2, \ldots, o(x_i)_B] \in \mathbb{R}^B$

Transformation tensor to be learned

allows to generate visually appealing samples very quickly, it is shown superior to feature matching

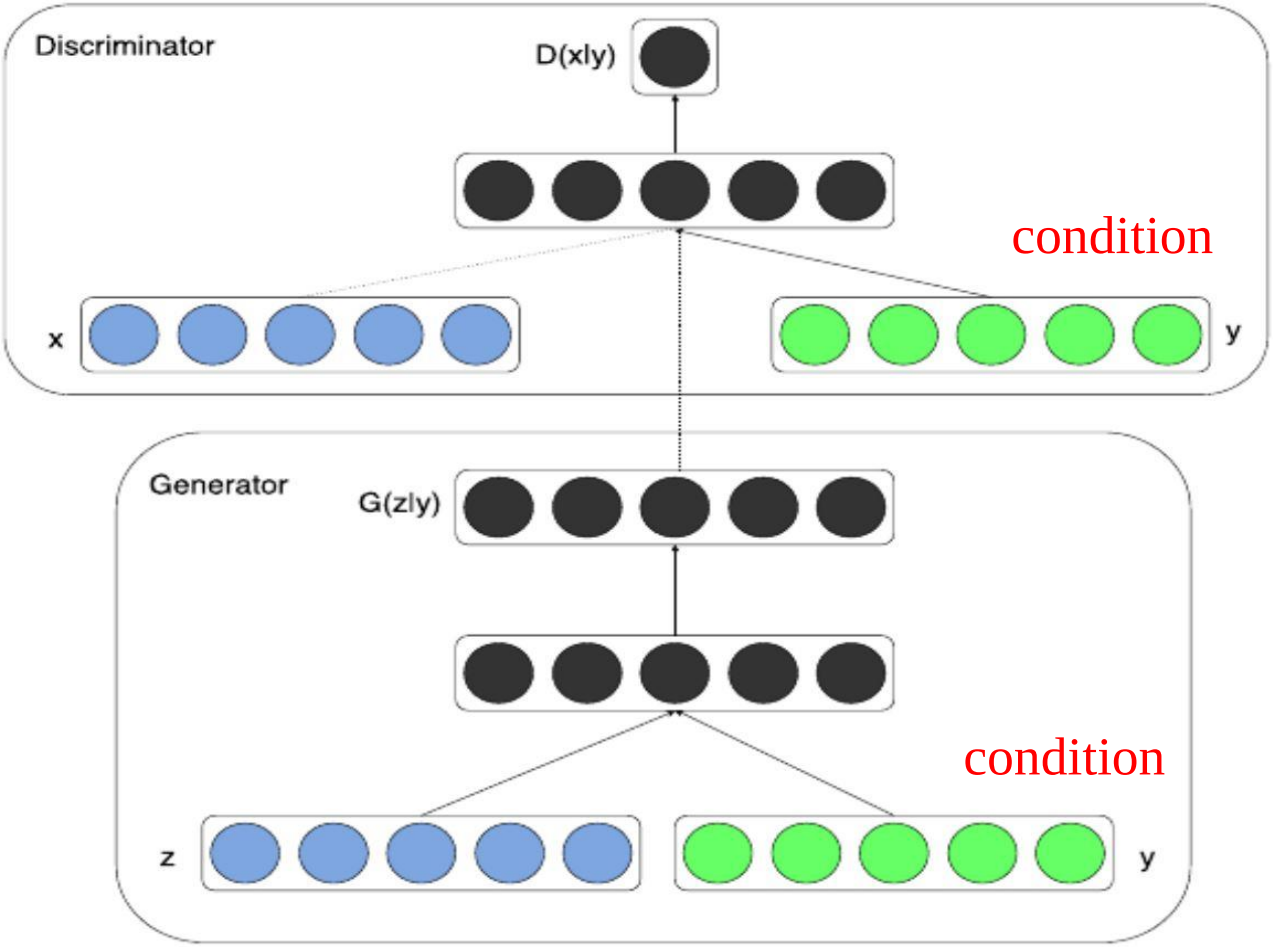
Guidelines for Stable Deep Convolutional GANs

- 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 for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

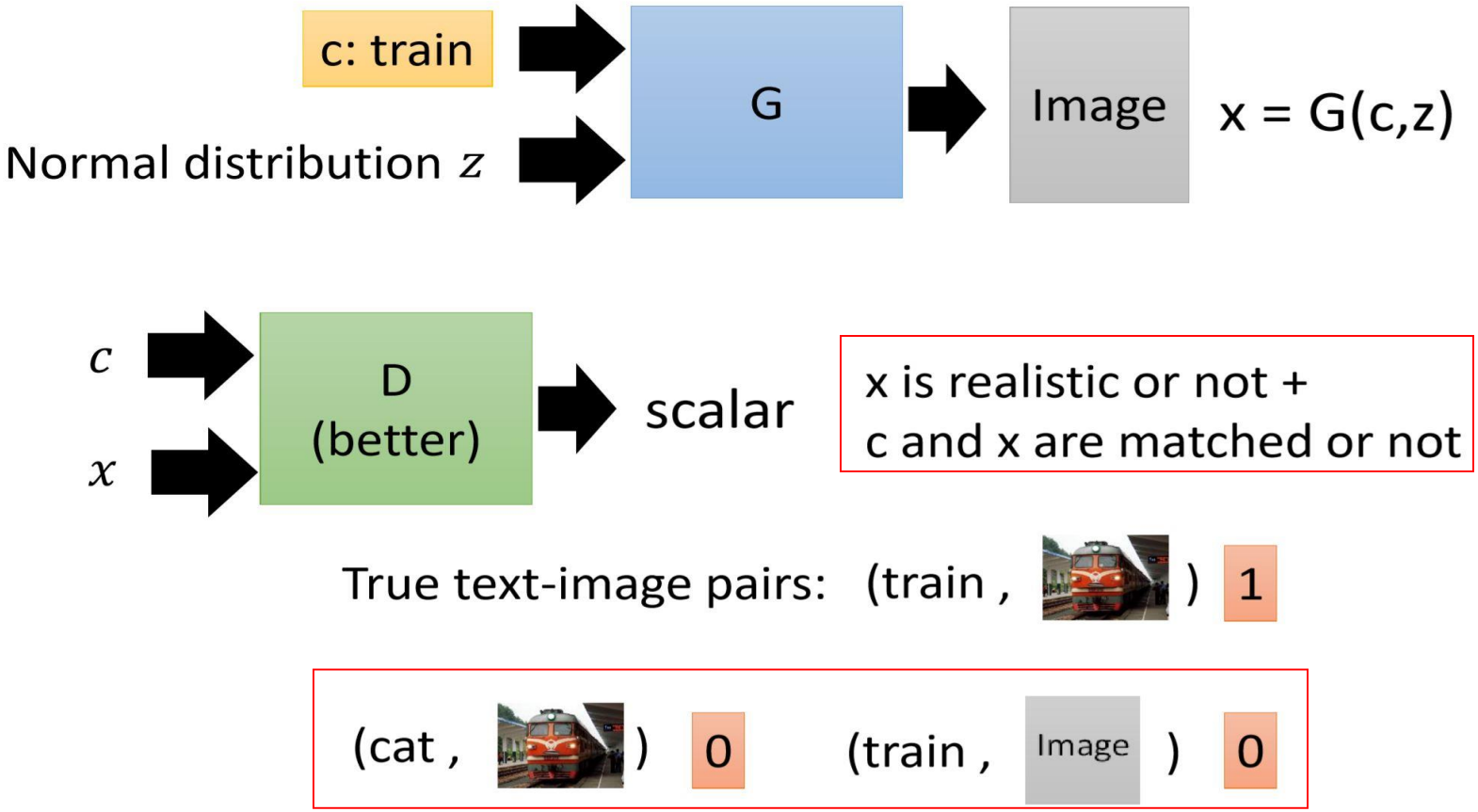


https://machinelearningmastery.com/how-to-train-stable-generative-adversarial-networks/

Conditional GAN (CGAN)



Conditional GAN: Side Text Information Hint



[Scott Reed, et al, "Generative Adversarial Text to Image Synthesis ICML, 2016]

Text2Image

this small bird has a pink breast and crown, and black primaries and secondaries.

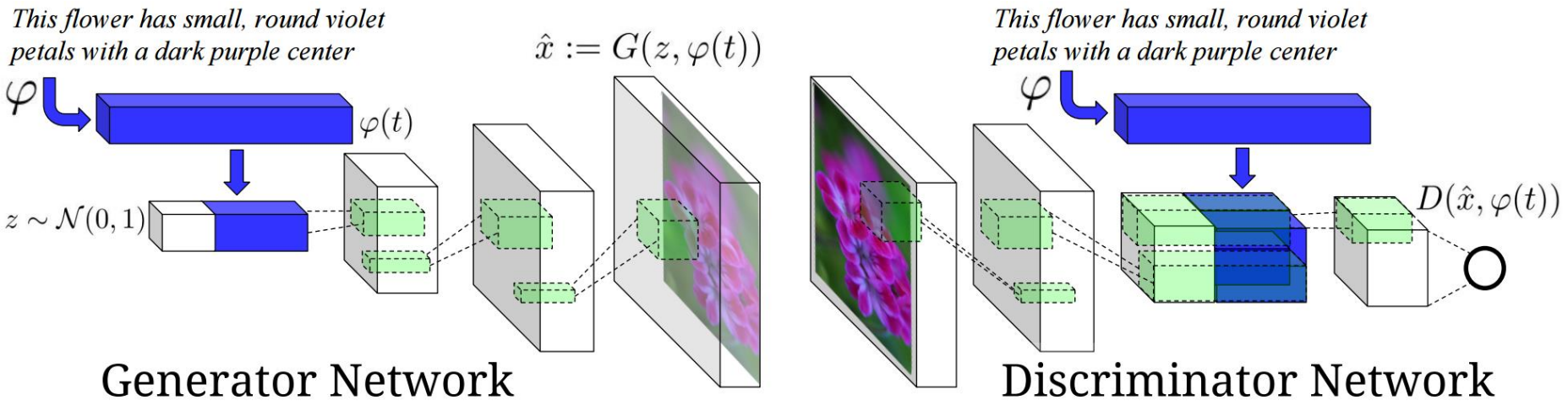
this magnificent fellow is almost all black with a red crest, and white cheek patch.

the flower has petals that are bright pinkish purple with white stigma

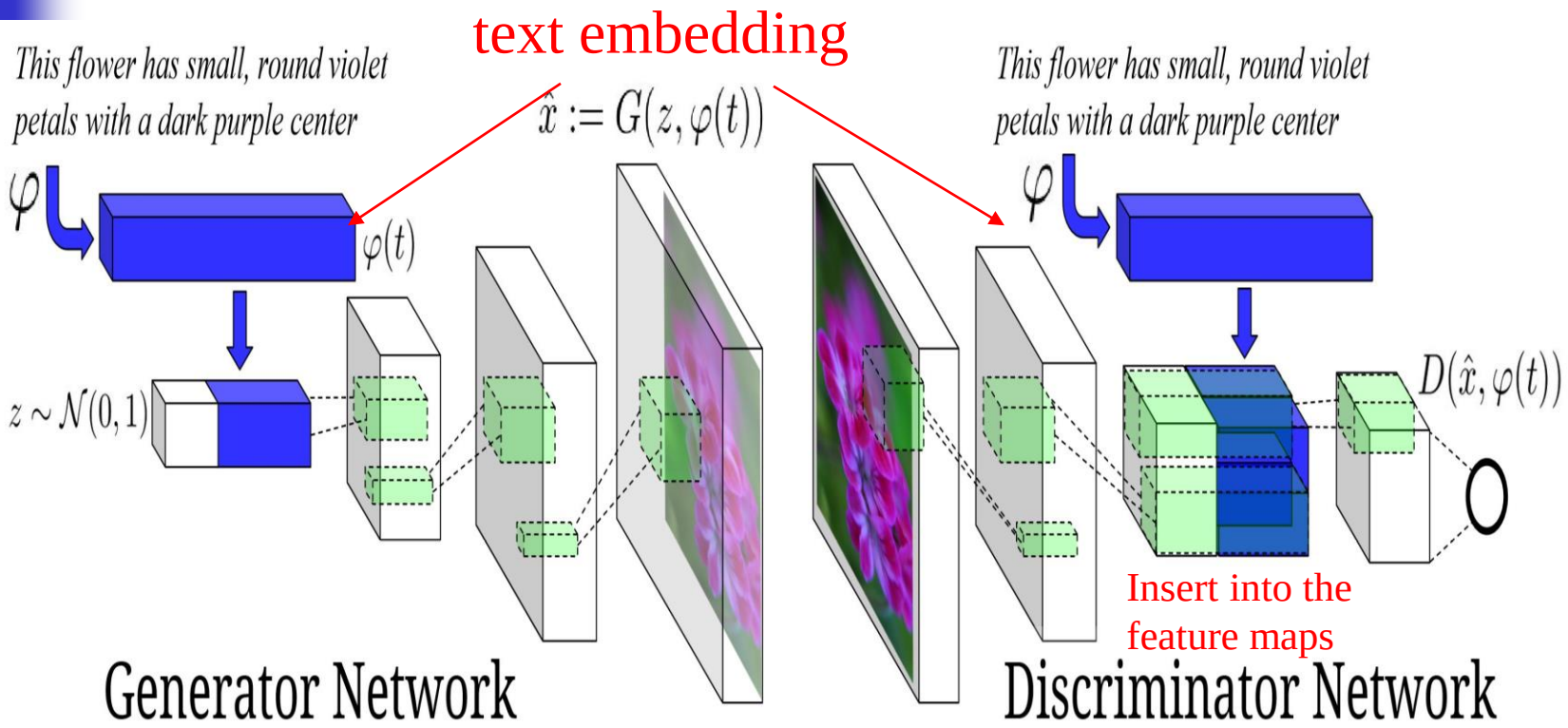
this white and yellow flower have thin white petals and a round yellow stamen

Text2Image

- Positive samples:
  - real image + right texts
- Negative samples:
  - fake image + right texts
  - Real image + wrong texts



Text Conditional GAN



$\mathcal{L}_G \leftarrow \log(s_f)$
$G \leftarrow G - \alpha \partial \mathcal{L}_G / \partial G$ {Update generator}

$s_r \leftarrow D(x, h)$ {real image, right text}
$s_w \leftarrow D(x, \hat{h})$ {real image, wrong text}
$s_f \leftarrow D(\hat{x}, h)$ {fake image, right text}
$\mathcal{L}_D \leftarrow \log(s_r) + (\log(1 - s_w) + \log(1 - s_f))/2$
$D \leftarrow D - \alpha \partial \mathcal{L}_D / \partial D$ {Update discriminator}

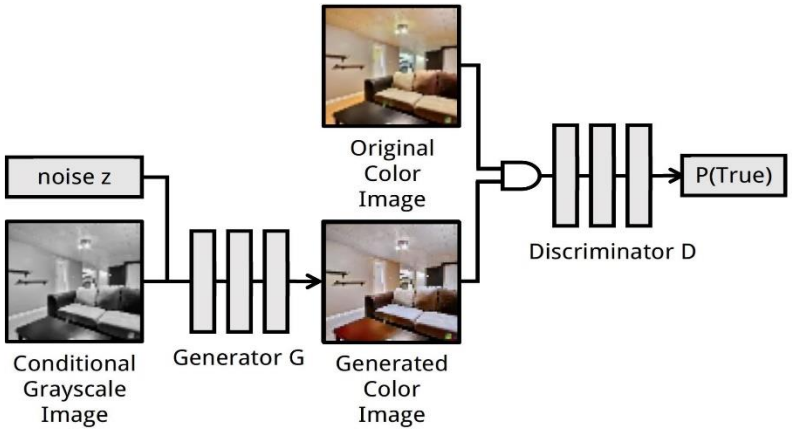
Text Conditional GAN

GAN with matching-aware discriminator (GAN-CLS)

GAN with manifold interpolation (GAN-INT)

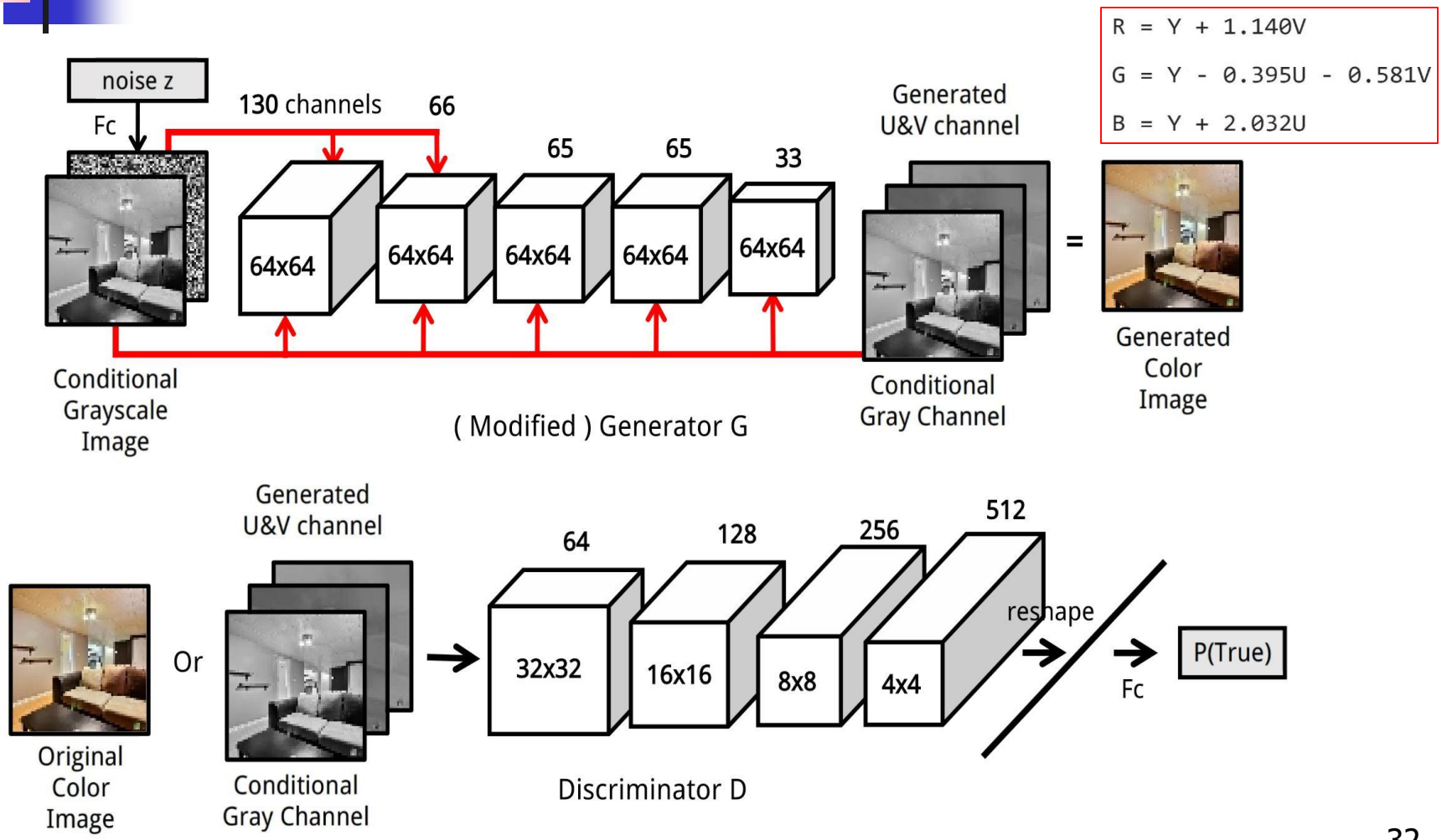
Text Conditional GAN

Conditional GAN: Side Image Information Hint



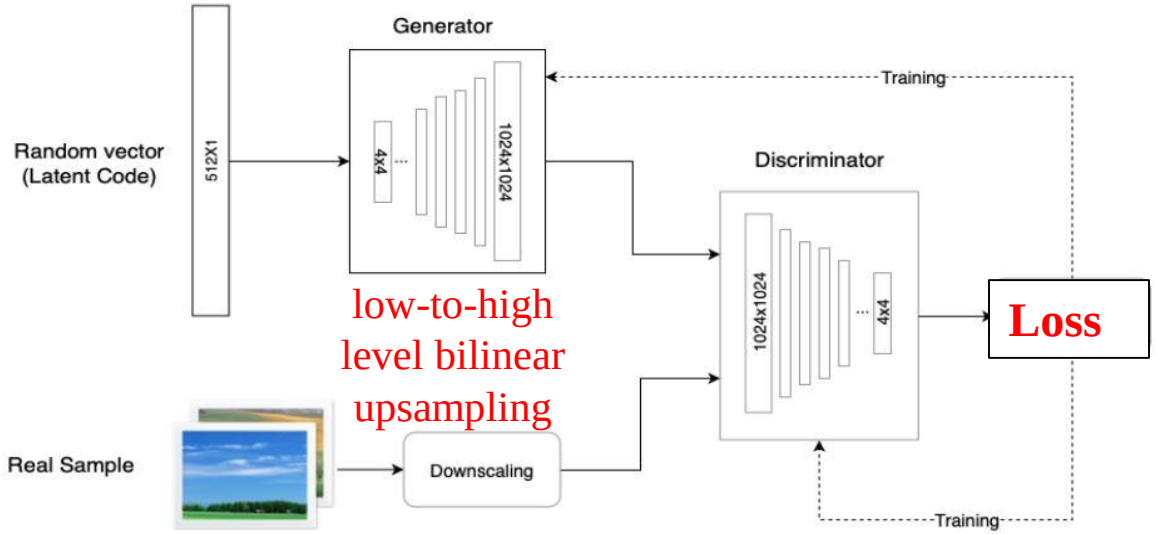
Yun Gao, et al., "Unsupervised Diverse Colorization via Generative Adversarial Networks," ECML PKDD 2017

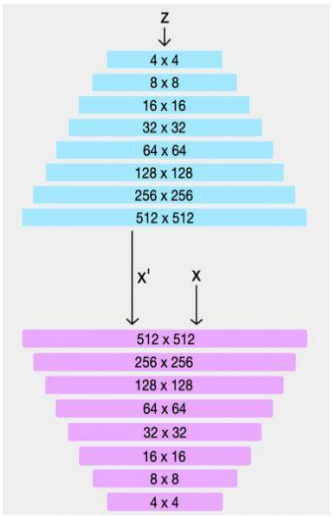
Multi-Layer Noise and Multi-Layer Condition

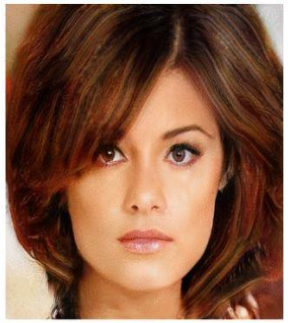


Progressive Growing GAN for Realistic Face Generation

- First creates the foundation of the image by learning the base features which appear even in a low-resolution image
- Learns more and more details over time as the resolution increases.
- Training the low-resolution images first is not only easier and faster, it also helps in training the higher levels, resulting in faster total training.







Training time: 10 days
512x512 resolution

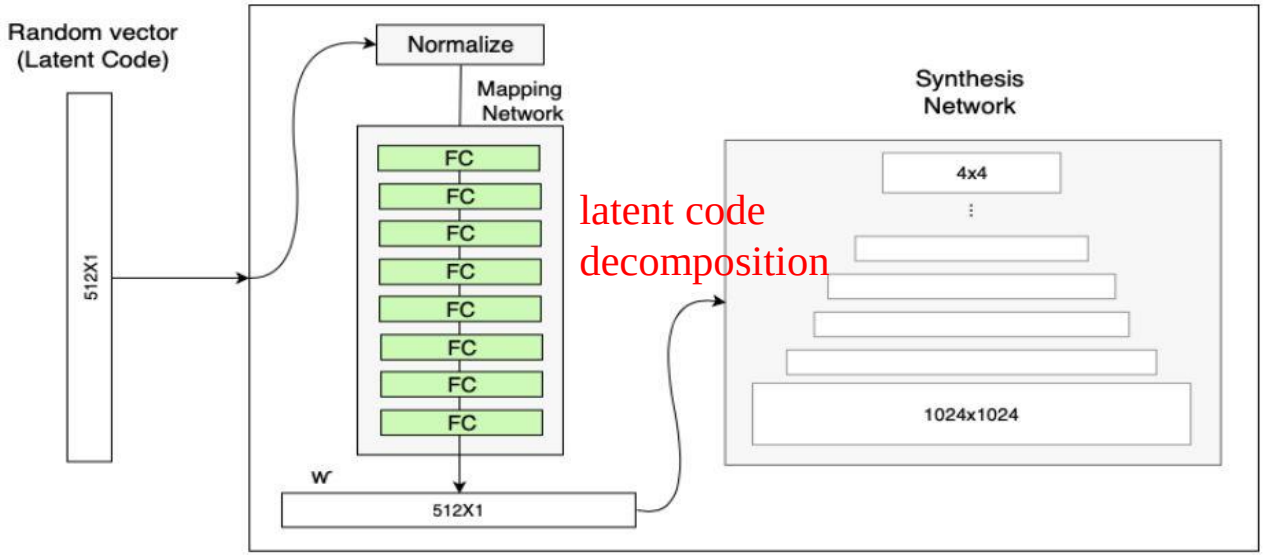
Generator
Discriminator

z = random code
x = real image
x' = generated image

Rero Karras, et al. "Progressive Growing of GANs for Improved Quality, Stability, and Variation," ICLR 2018

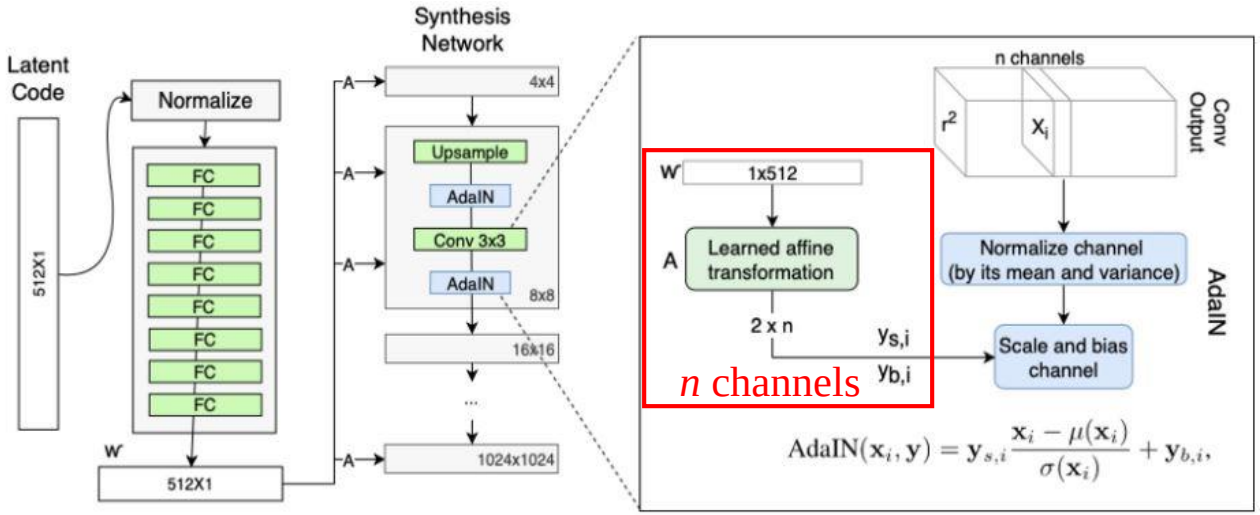
Mapping Network: Undo Feature Entanglement

- Given an input vector (latent code), the GAN generates the output following the probability density of the training data
- A model is not capable of mapping parts of the input (elements in the vector) to features, a phenomenon called features entanglement
- A mapping network encodes the input vector into an intermediate vector whose different elements control different visual features



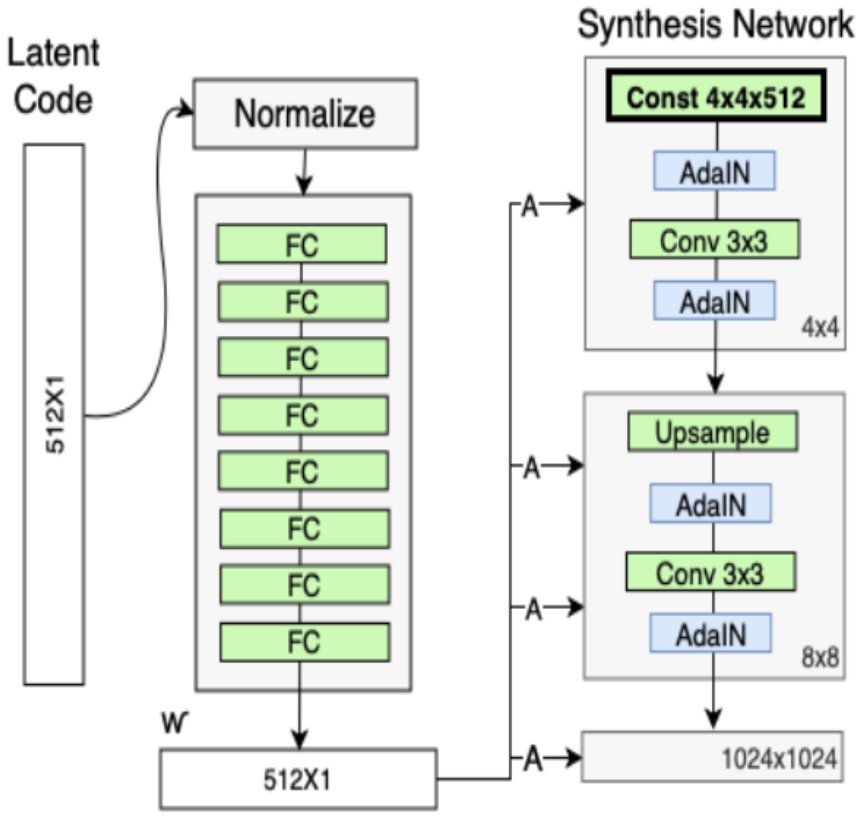
Mapping Network

- Each channel of the convolution layer output is first normalized to make sure the scaling and shifting of growing have the expected effect
- The intermediate vector w is transformed using another 8-layer MLP (marked as A) into a scale and bias (style) for each channel
- The scale and bias vectors shift each channel of the convolution output, thereby defining the importance of each filter in the convolution. This tuning translates the information from w to a visual representation



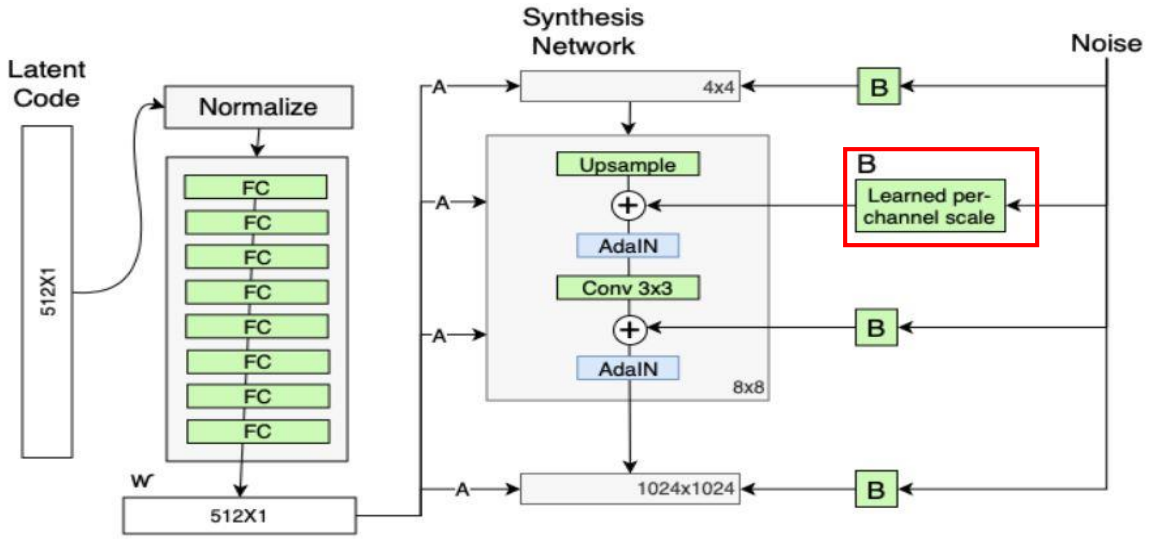
Constant Initial Value Input

- Latent codes become control variables, not inputs
- Instead of using the random input to create the initial image of the generator (i.e., the input of the 4×4 level), the StyleGAN uses constant values (4x4x512).
- It can reduces feature entanglement — it's easier for the network to learn only using w without relying on the entangled input vector.



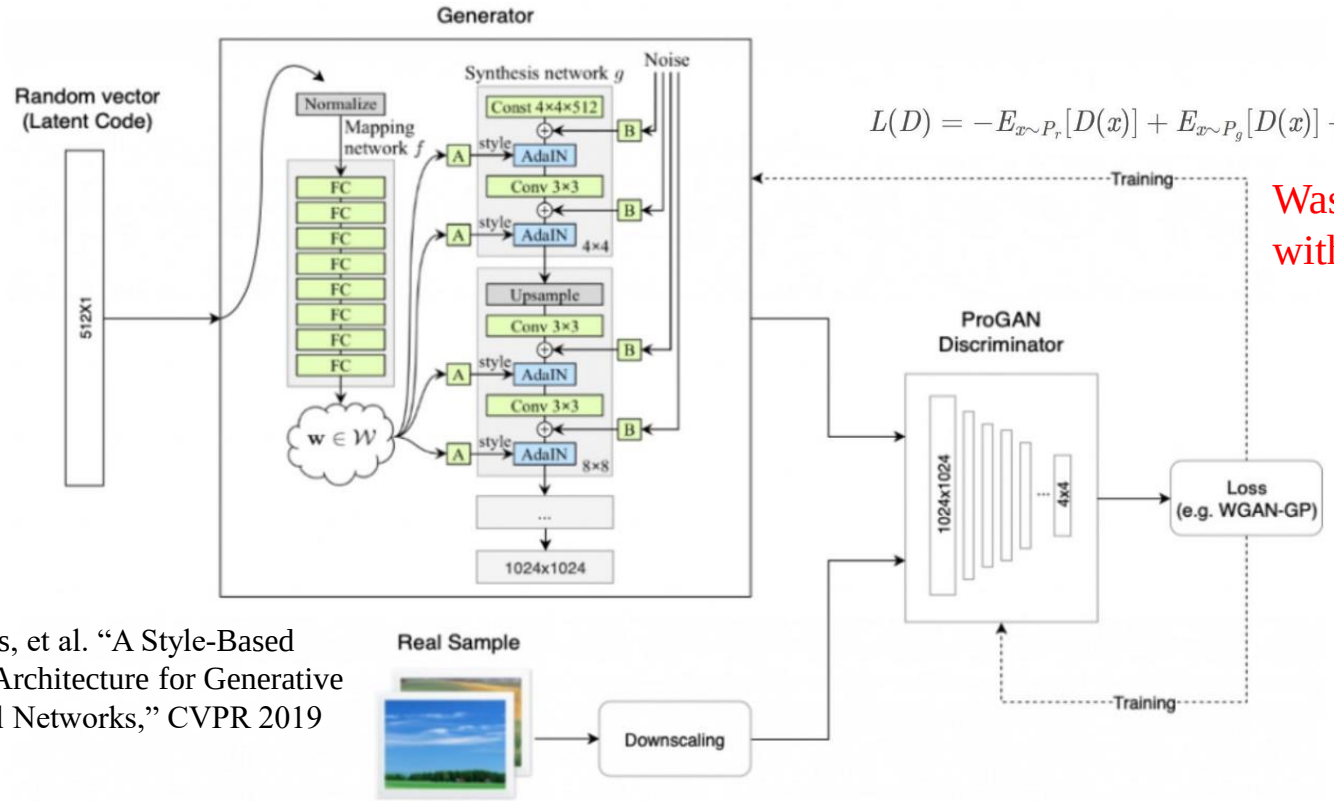
Stochastic Variation

- Many aspects in people's faces that are small and can be seen as stochastic, such as freckles, exact placement of hairs, wrinkles, features which make the image more realistic and increase the variety of outputs.
- The noise in StyleGAN is added in a similar way to the AdaIN mechanism — A learned scaled noise (by B) is added to each channel before the AdaIN module and changes a bit the visual expression of the features of the resolution level it operates on.



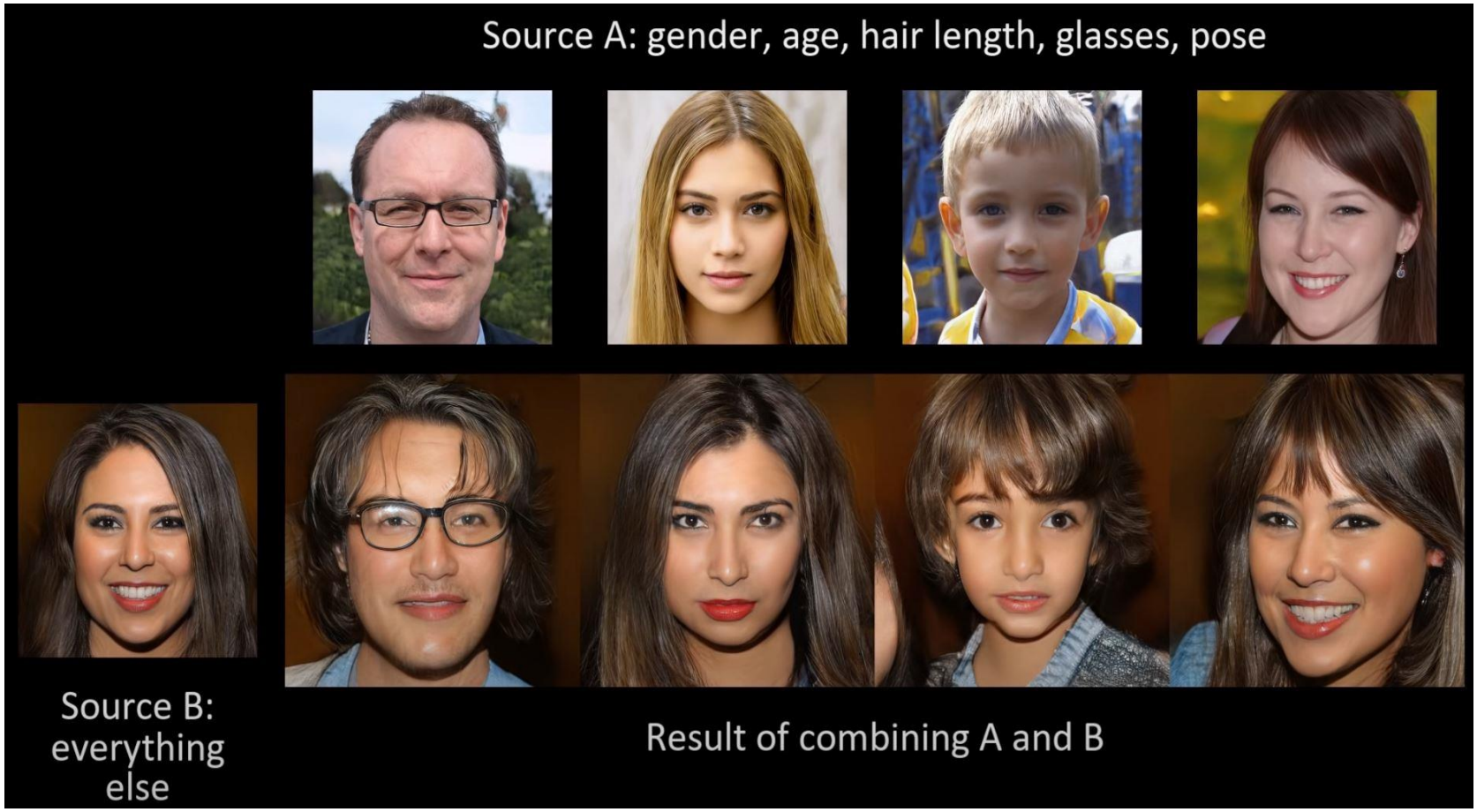
Random Switches of Level Learning

- The generated intermediate vector w is used to train "some of the levels" in random switches. The random switch ensures that the network won't learn and rely on a correlation between levels.



Rero Karras, et al. "A Style-Based Generator Architecture for Generative Adversarial Networks," CVPR 2019

Mixing Style B into Source A

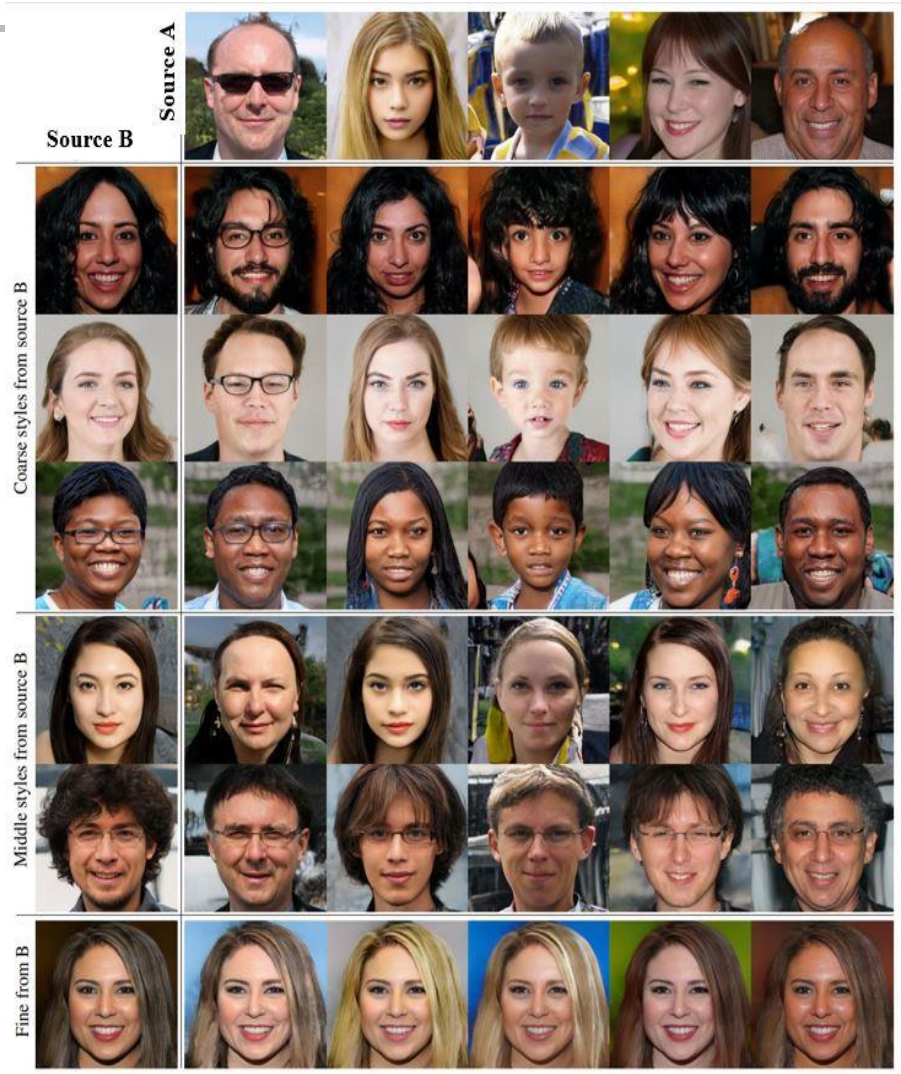


Style Mixing Regularization

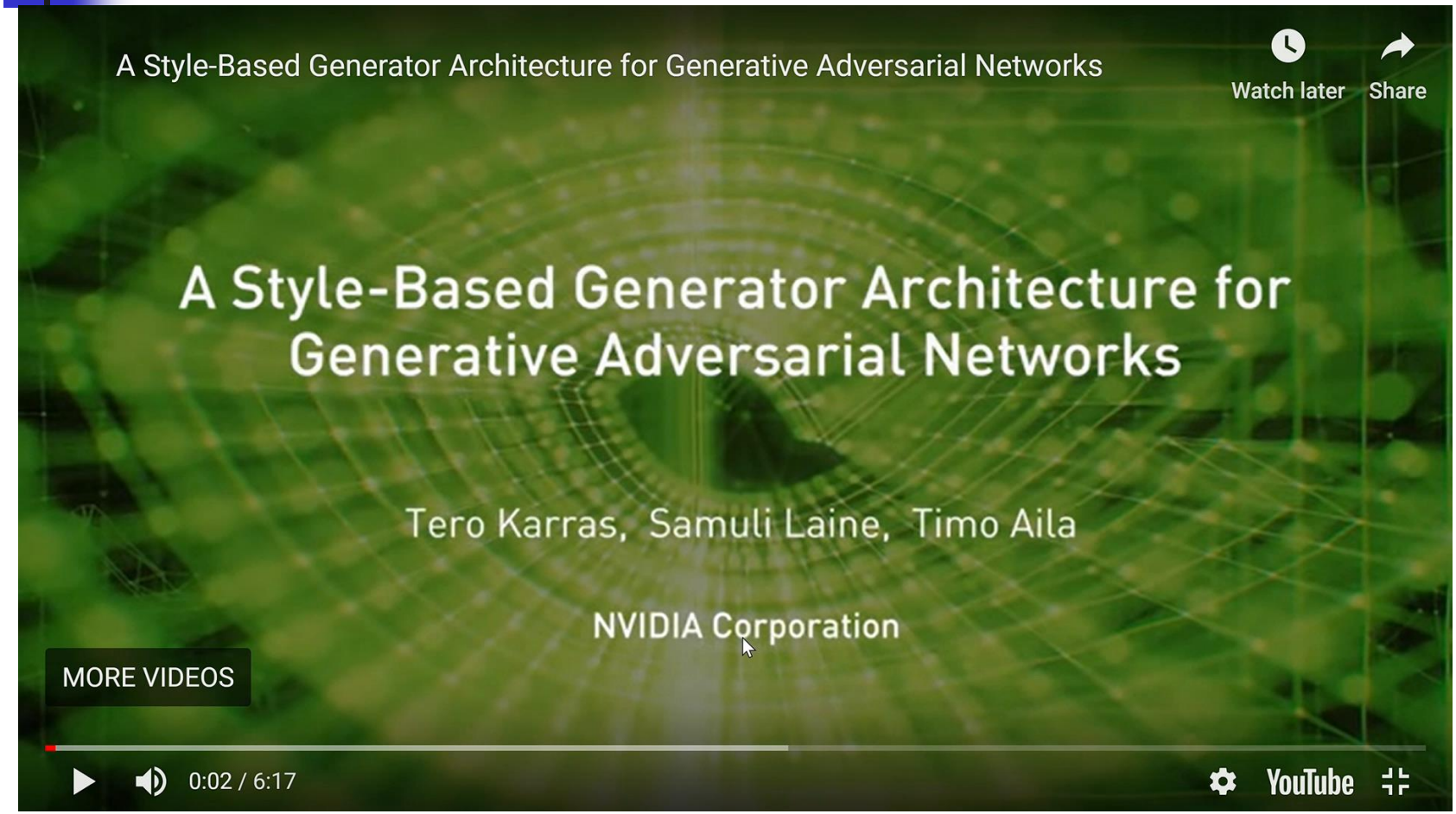
- Mixing regularization: a given percentage of images are generated using two random latent codes instead of one during training.
- When generating such an image, we simply switch from one latent code to another (style mixing) at a randomly selected point in the synthesis network.
- Two latent codes z_1,z_2 through the mapping network, and have the corresponding w_1,w_2 control the styles so that w_1 applies before the crossover point and w_2 after it.

Style Mixing in StyleGAN

- Two sets of images were generated from their respective latent codes (sources A and B)
- Resolutions (4^2 – 8^2) brings coarse-level aspects such as pose, hair style, face shape, from source B
- Middle resolutions (16^2 – 32^2) inherit middle-level aspects, such as facial features, eyes open/closed from B, while the pose, general face shape, and eyeglasses from A are preserved
- Fine styles (64^2 – 1024^2) from B brings mainly the color scheme and microstructure

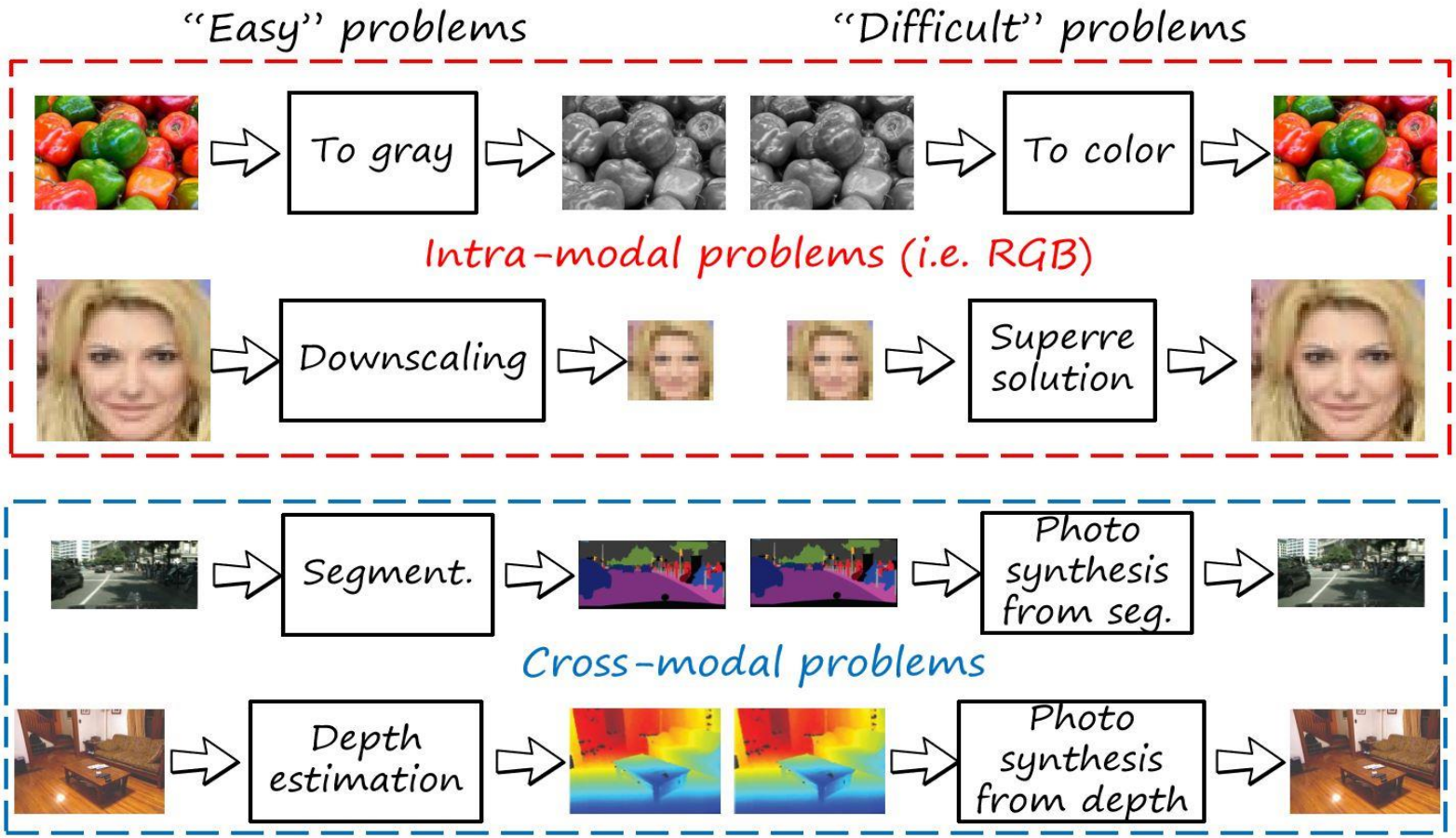


StyleGAN by Nvidia
(IEEE CVPR 2019)



https://www.youtube.com/watch?v=kSLJriaOumA

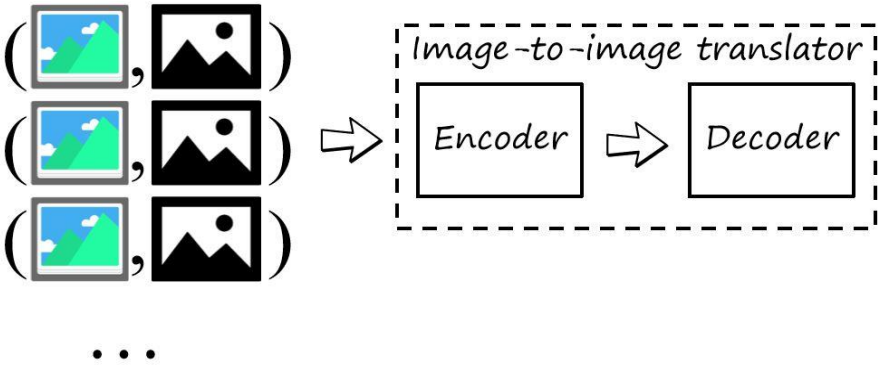
Paired Image-to-Image Translation (pix2pix)

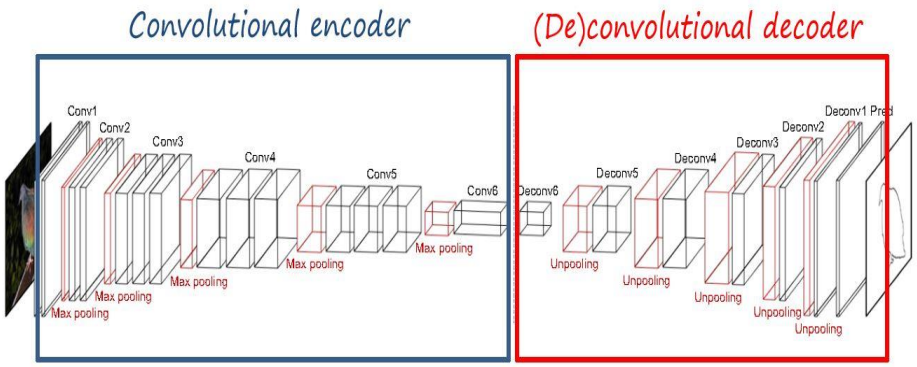


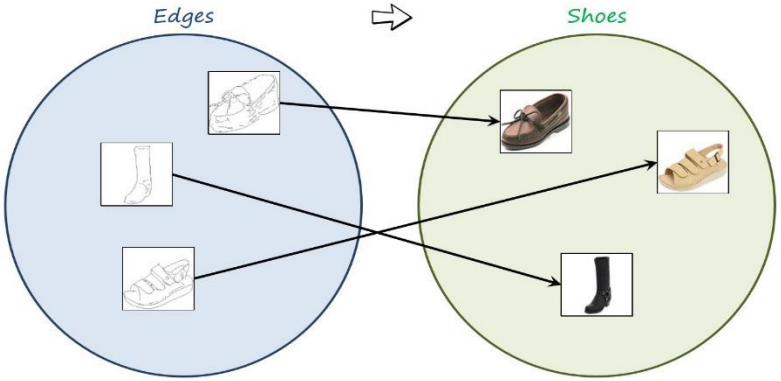
Phillip Isola, et al, "Image-to-Image Translation with Conditional Adversarial Networks," CVPR 2017

Pix2Pix GAN

- Learns from image pairs (input, output)

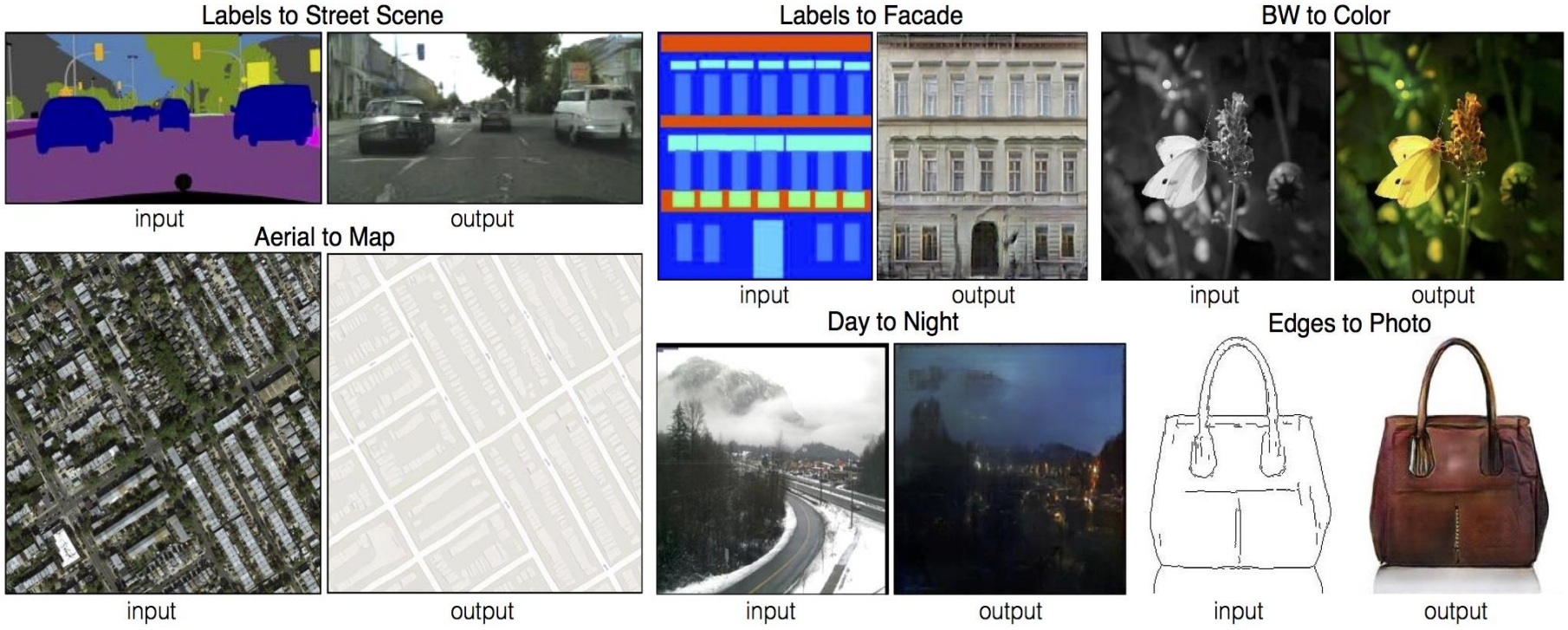






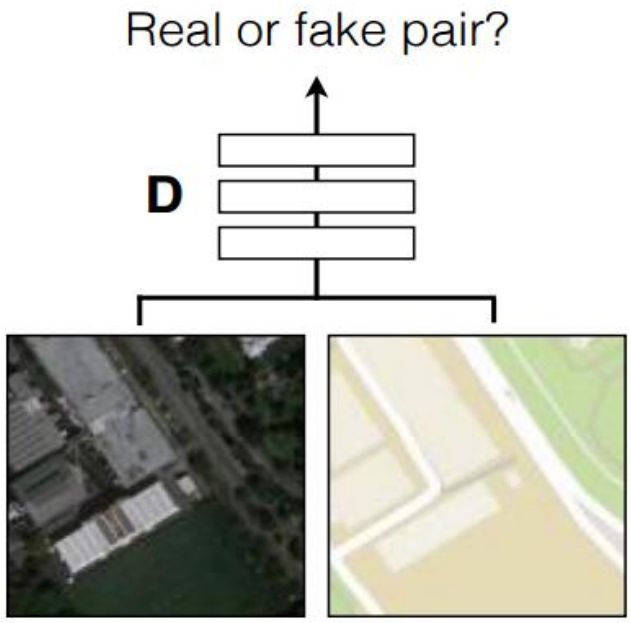
https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

Conditional GAN for Image-to-Image Translation

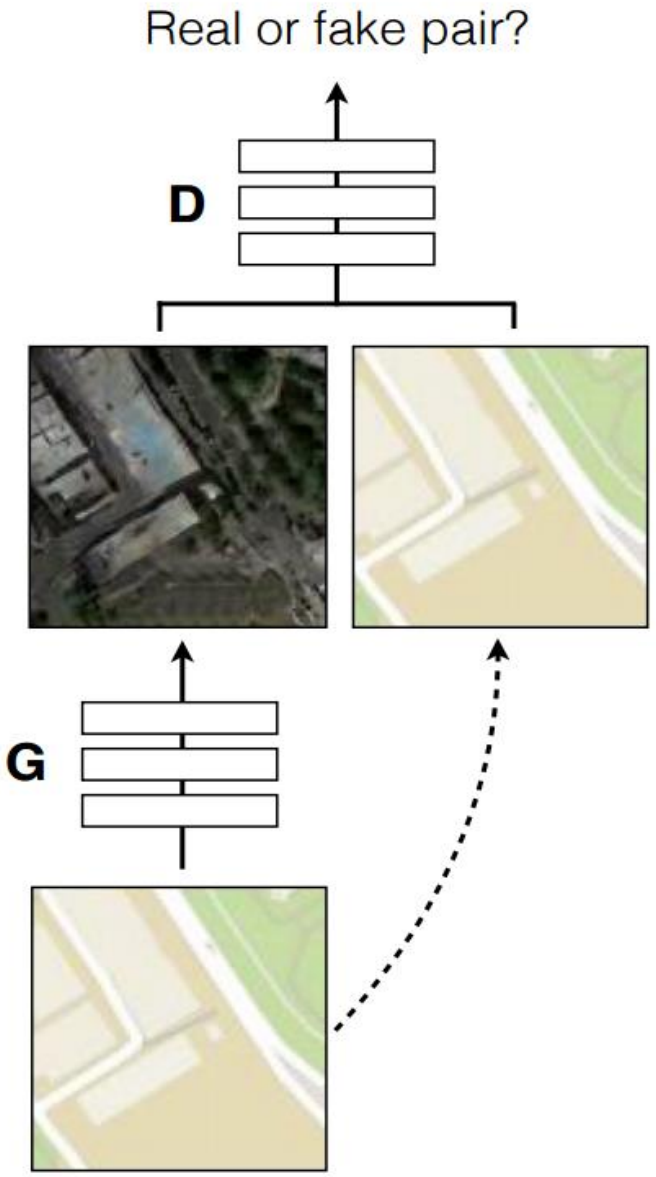


- Conditioned on an image of different modality
- No need to specify the loss function

Positive examples



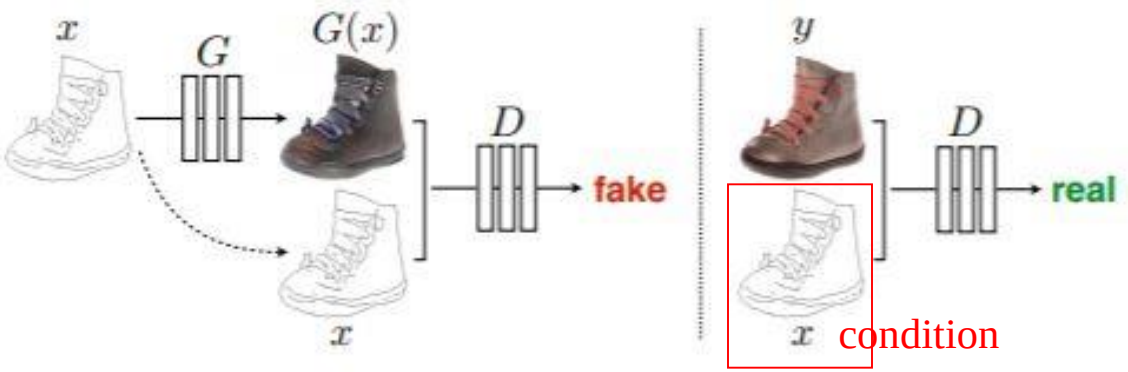
Negative examples



G tries to synthesize fake images that fool **D**

D tries to identify the fakes

From Pix2Pix to A Conditional GAN (CGAN)

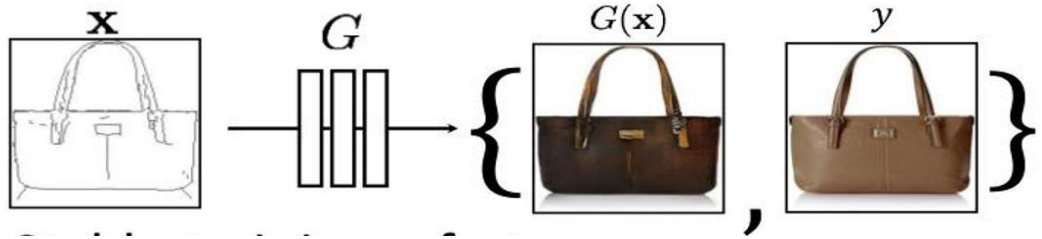


Philip Isola, et al, "Image-to-Image Translation with Conditional Adversarial Networks," CVPR 2017

$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G).$$

$\lambda = 100$

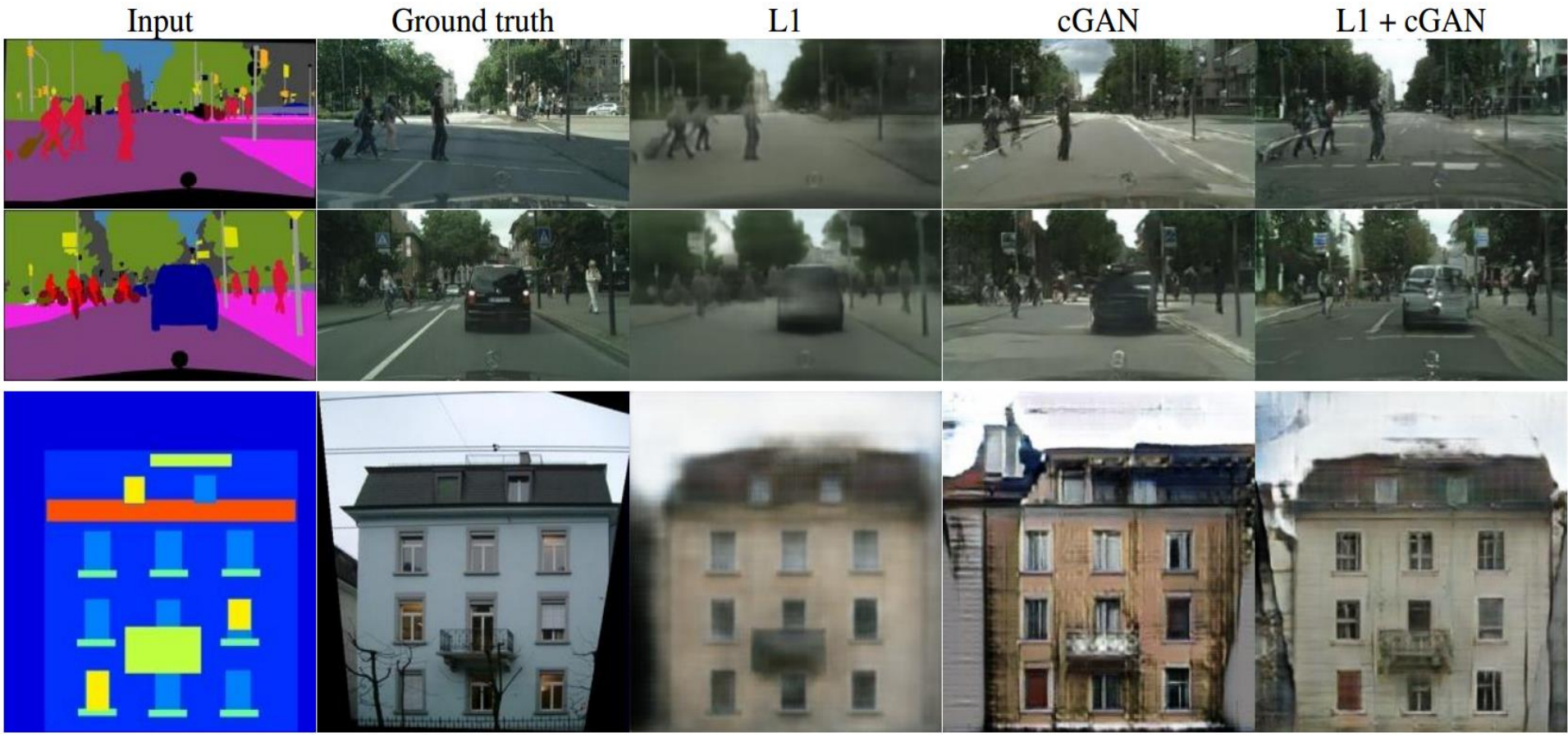
$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x,z)\|_1].$$



L1 encourages sharper outputs, while average, grayish colors

https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

Label2Image Performance



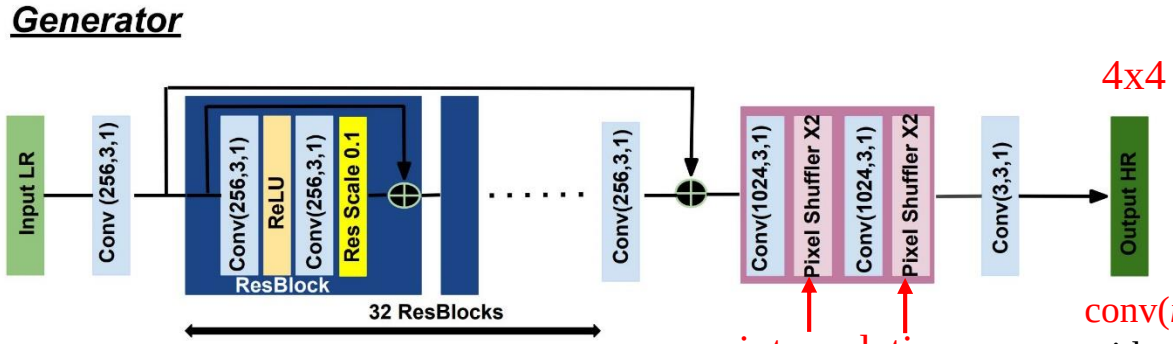
Phillip Isola, et al, "Image-to-Image Translation with Conditional Adversarial Networks," CVPR 2017

Edges2Image Performance

Phillip Isola, et al, "Image-to-Image Translation with Conditional Adversarial Networks," CVPR 2017

Application to Image Super-Resolution

- Enhanced Perceptual Super-Resolution (EPSR) Network



Subeesh Vasu, et al., "Analyzing Perception-Distortion Tradeoff using Enhanced Perceptual Super-resolution Network," ECCV 2018

conv(n,k,s) refers to a convolution layer with n number of $k \times k$ filters performing convolution by a stride factor of s.

Pretrained VGG19 (4th conv) as the feature extraction layer

$f(x) = 1(x < 0)(\alpha x) + 1(x >= 0)(x)$ where α is a small constant.

$$\mathcal{L} = \lambda_1 \mathcal{L}_{\mathcal{VGG}} + \lambda_2 \mathcal{L}_E + \lambda_3 \mathcal{L}_{adv}$$

$$\mathcal{L}_{\mathcal{VGG}} = ||\phi(I_{\text{est}}) - \phi(I_{\text{HR}})||_2^2$$

Feature Maps errors

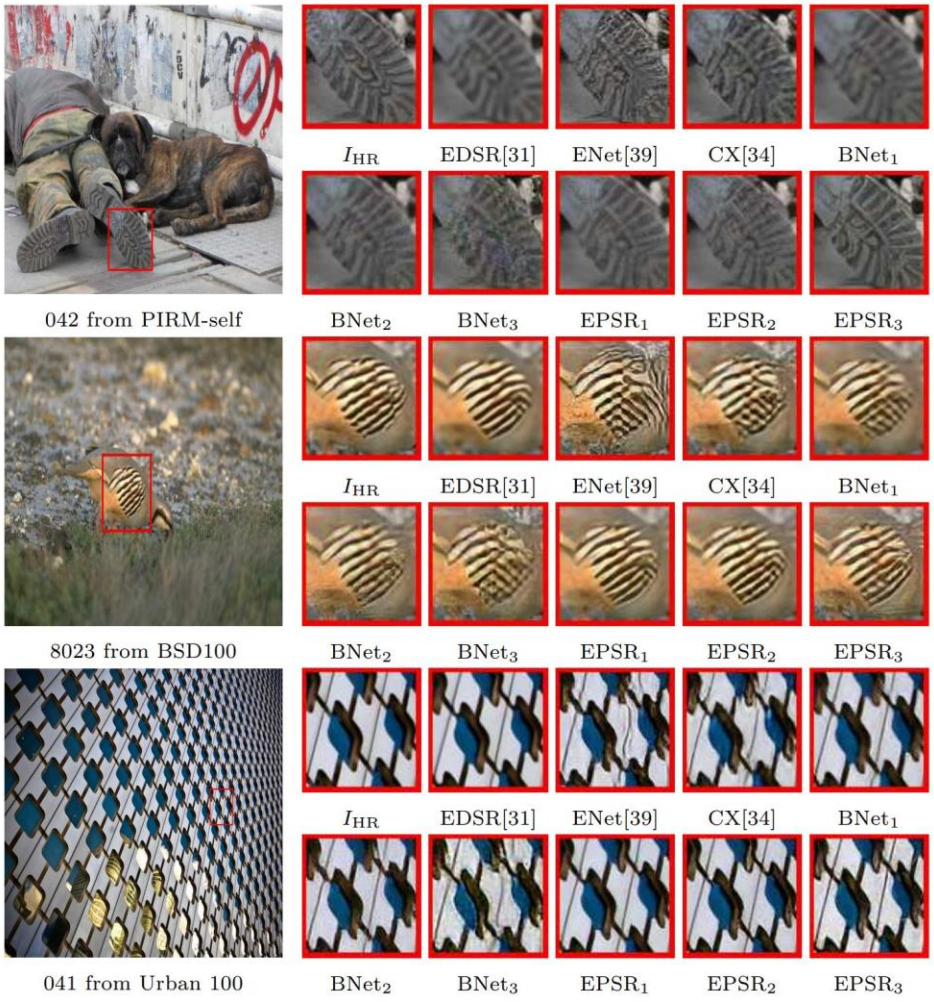
$$\mathcal{L}_E = ||I_{\text{est}} - I_{\text{HR}}||_2^2$$

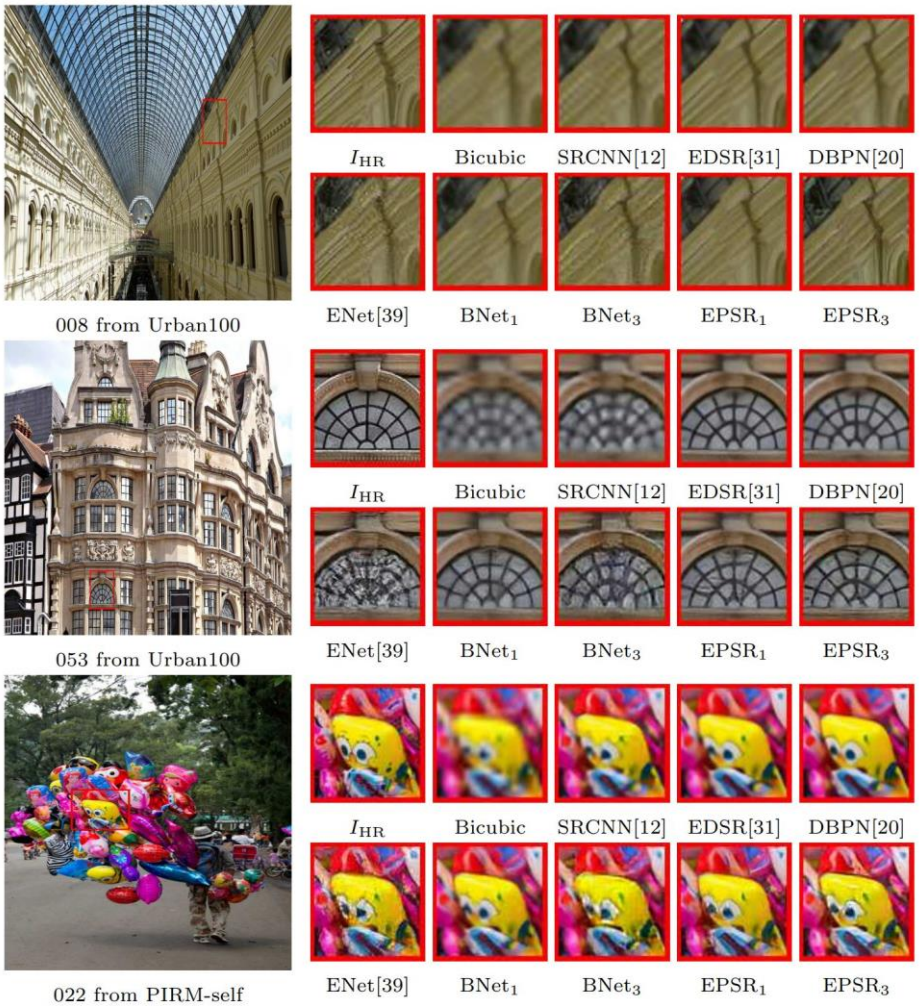
Image appearance errors

$$\mathcal{L}_{adv} = -\log D(G(I_{\text{LR}}))$$

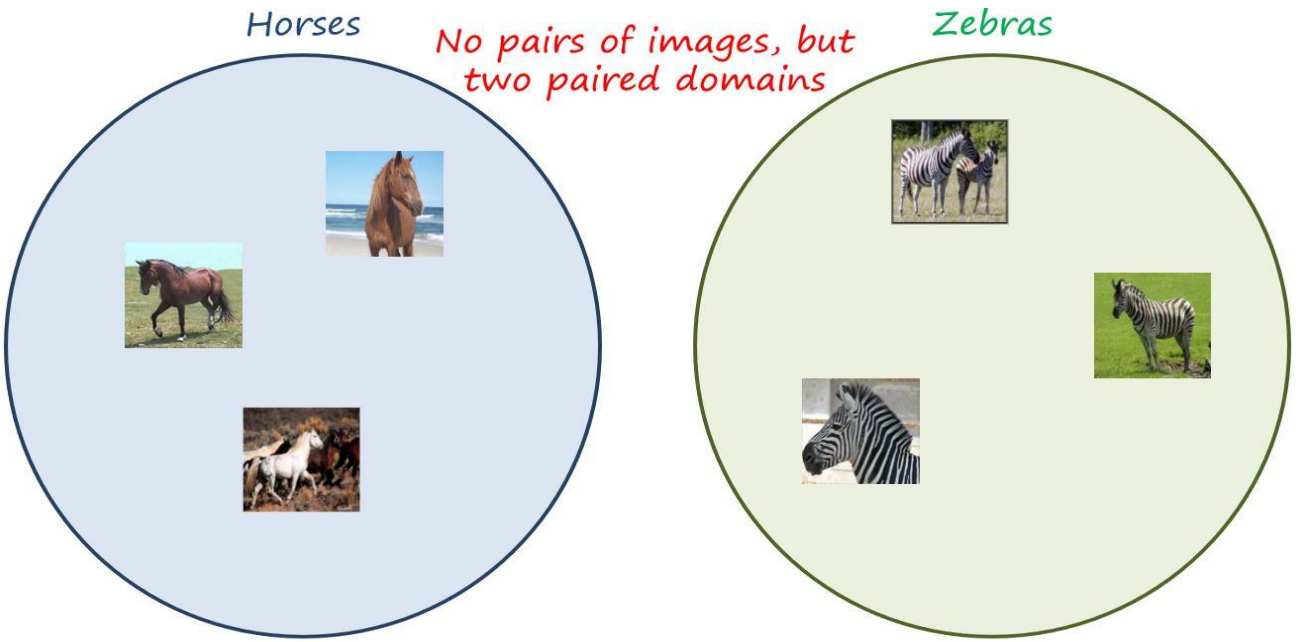
GAN loss

SR CGAN Performance





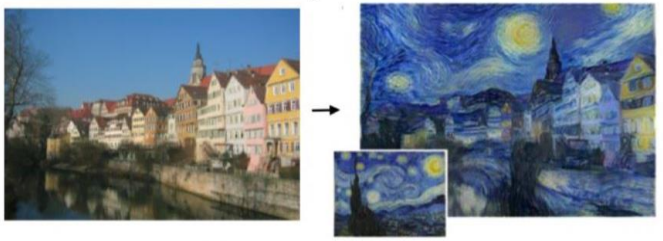
Unpaird Image-to-Image Translation



Season change

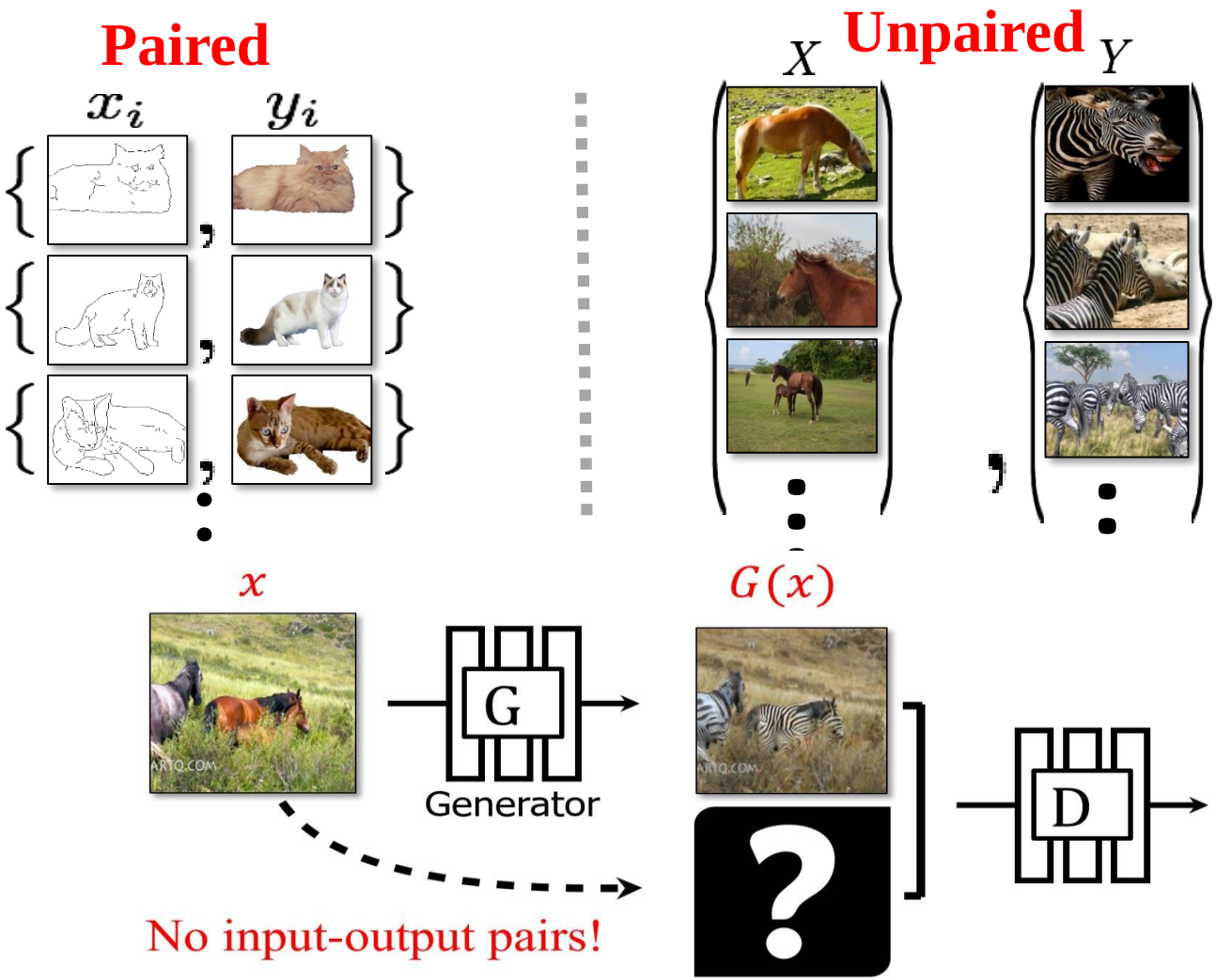
[Laffont et al. 2014]

Artistic style transfer

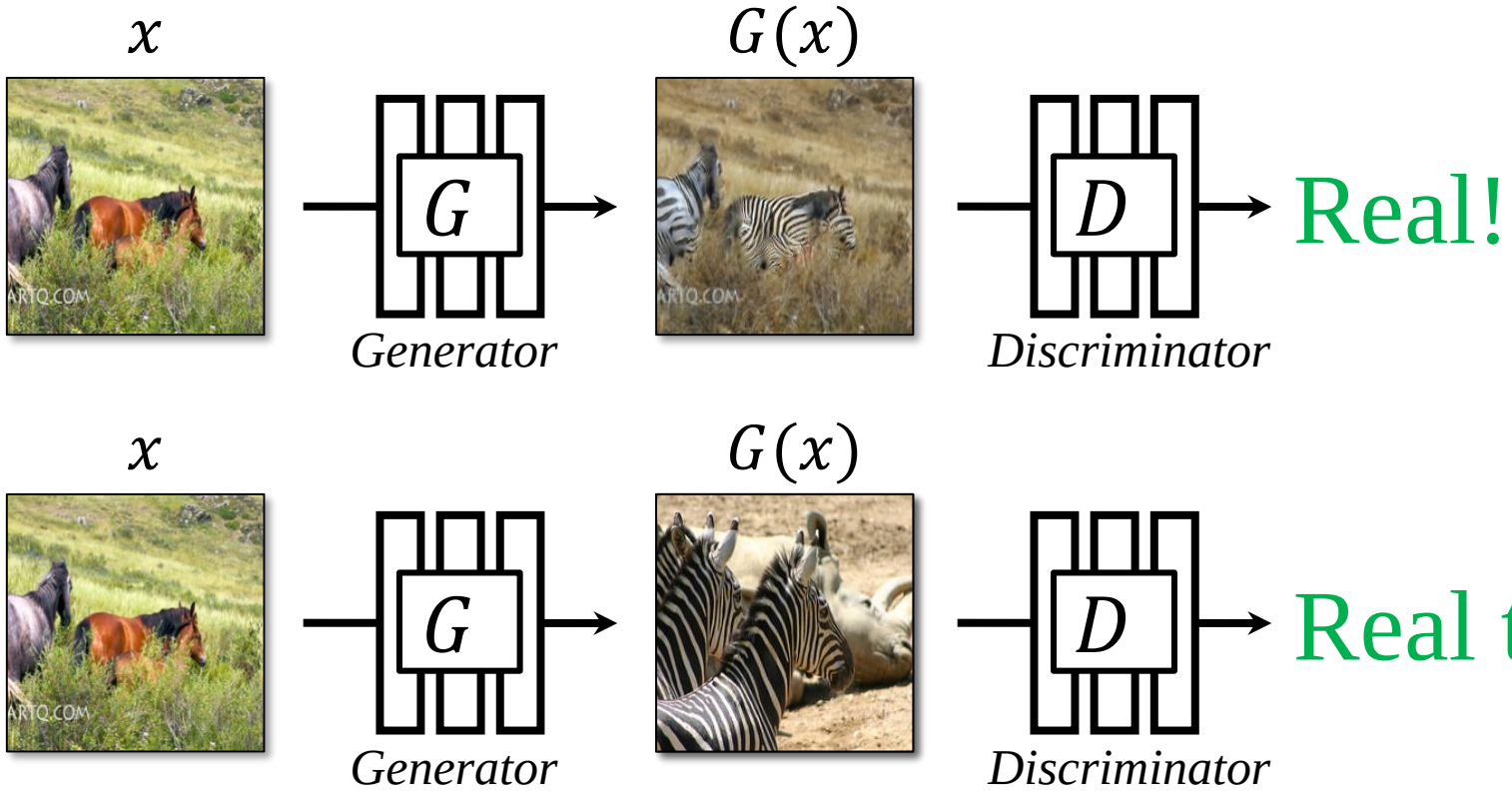


[Gatys et al. 2016]

Unpaired Training Data

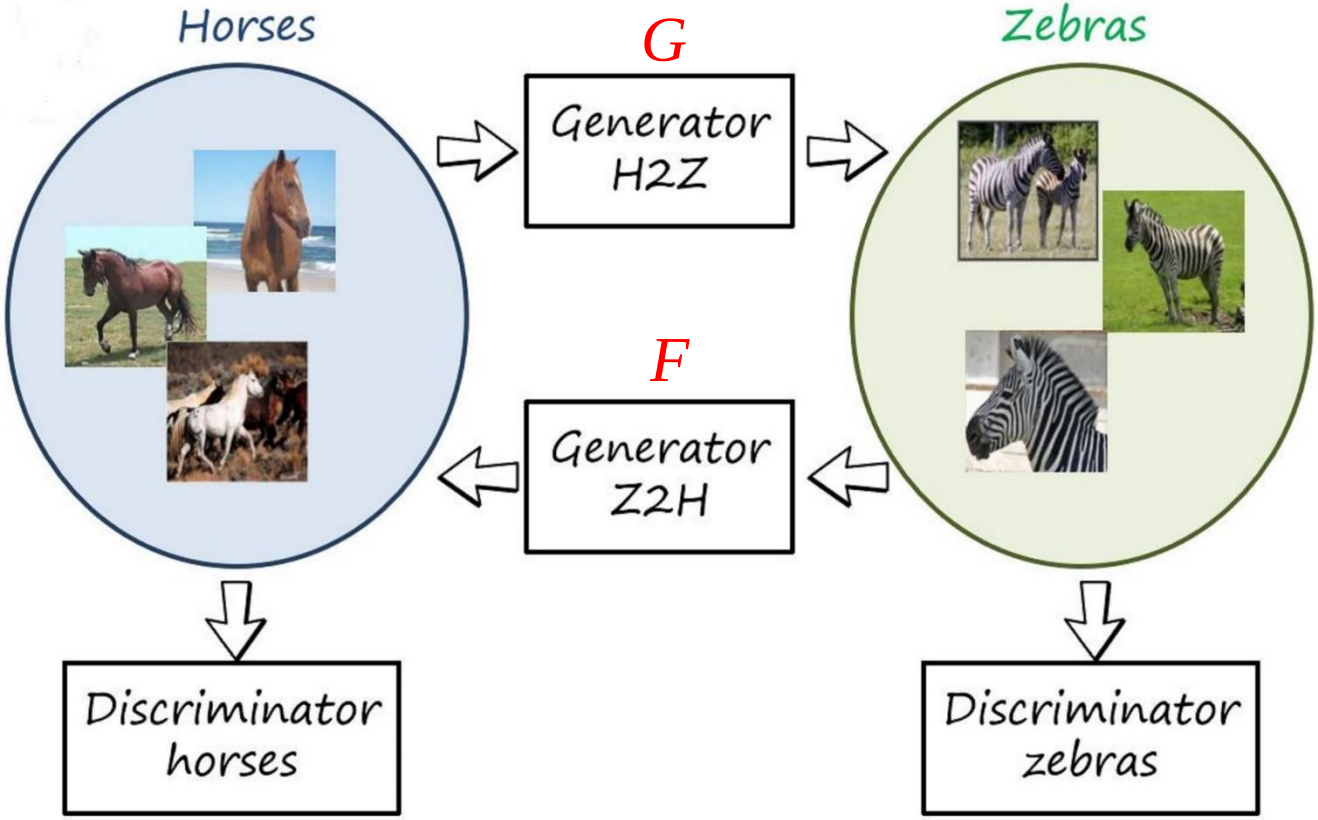


One-to-Many Mapping?



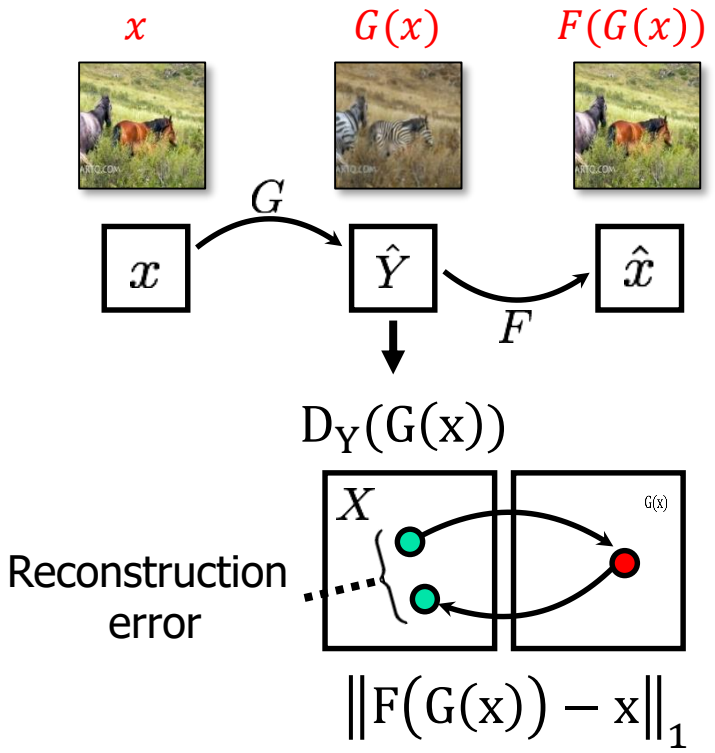
GANs do **not** force output to correspond to input

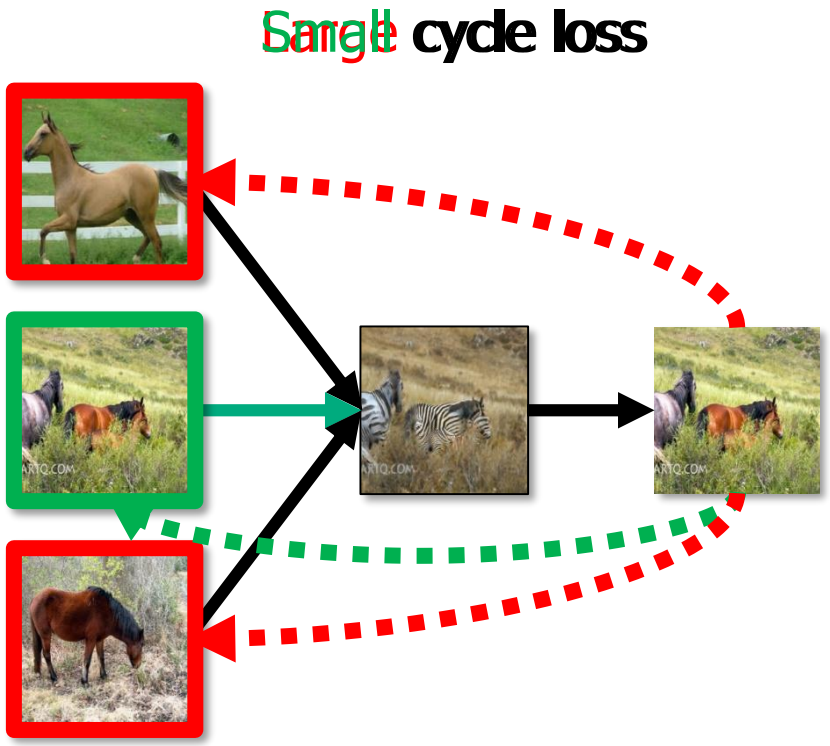
Cycle Consistency GAN (CycleGAN)



Zhu et al, "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks", ICCV 2017

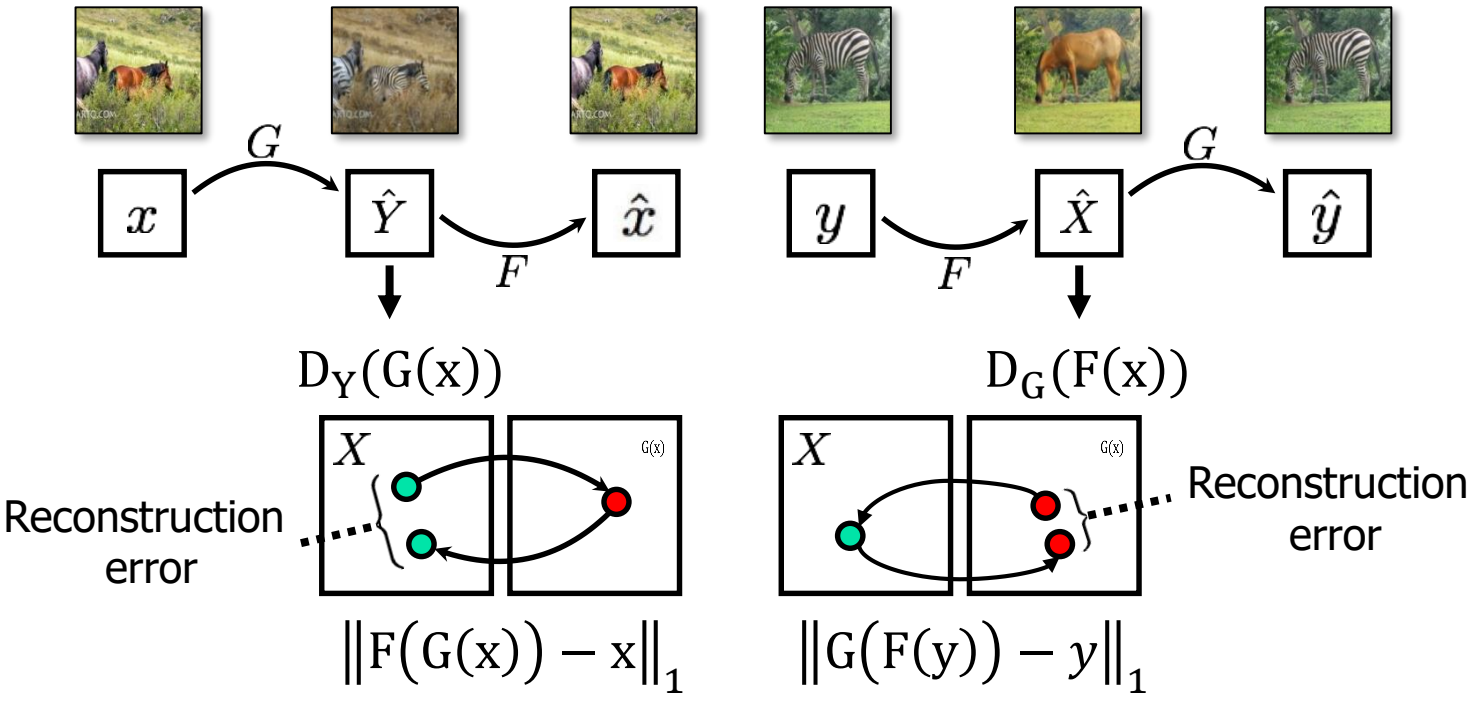
Minimize Cycle Loss





[Zhu*, Park*, Isola, and Efros, ICCV 2017]

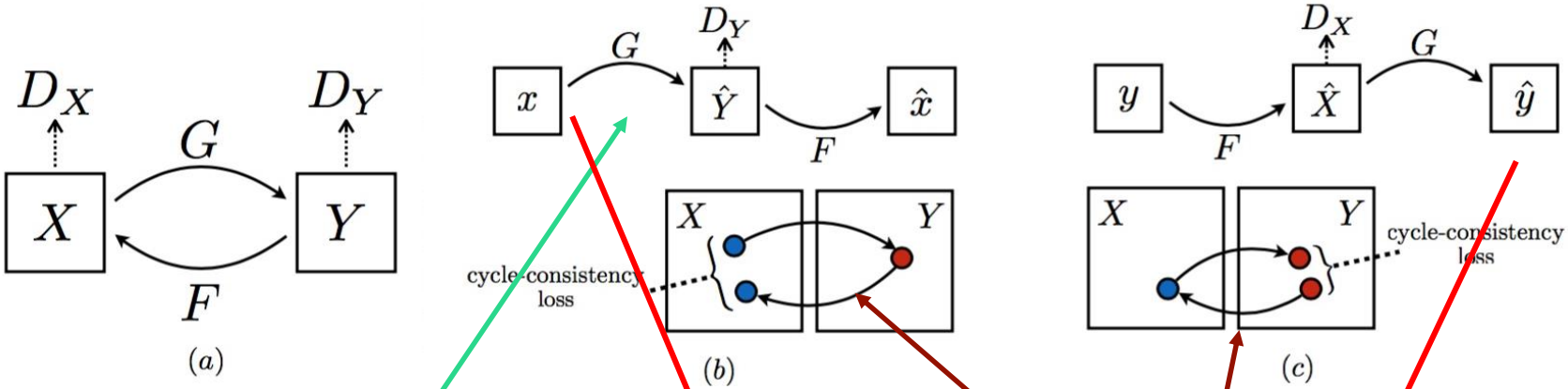
Cycle-Consistent Loss



See similar formulations [Yi et al. 2017], [Kim et al. 2017]

[Zhu*, Park*, Isola, and Efros, ICCV 2017]

Complete CycleGAN Loss



$$\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x))],$$

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

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

CycleGAN Results

Monet Paintings to Photos