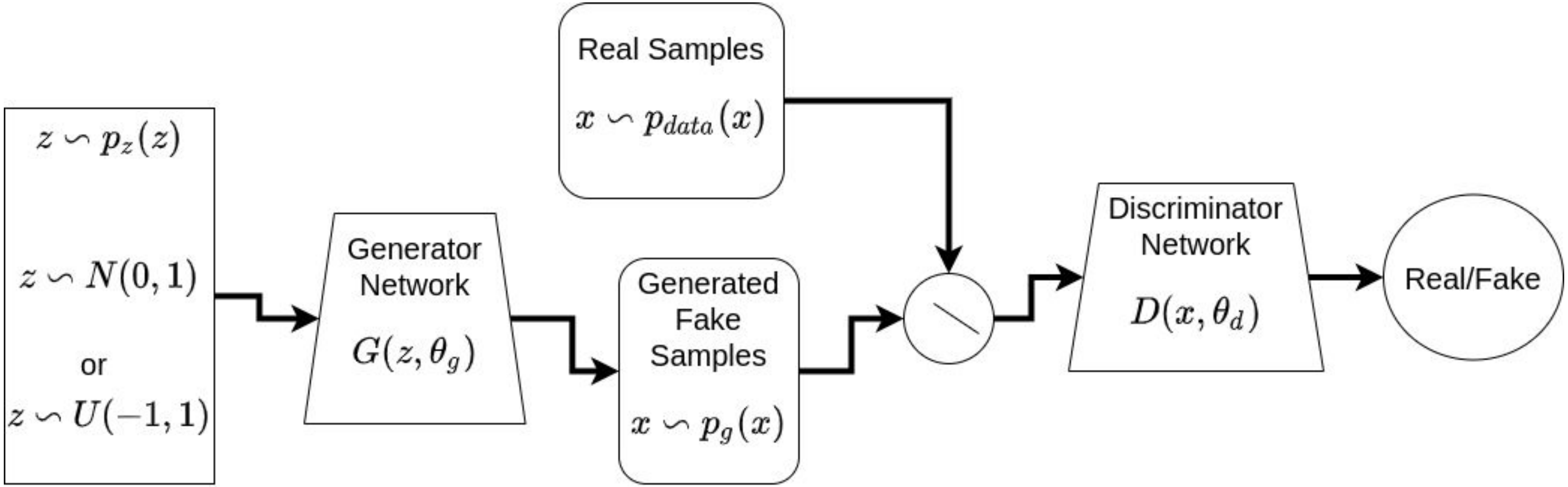
Why Image Generation is challenging?

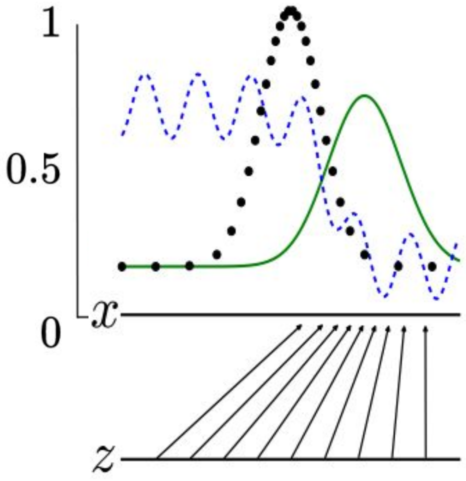
Training Challenges In GANs

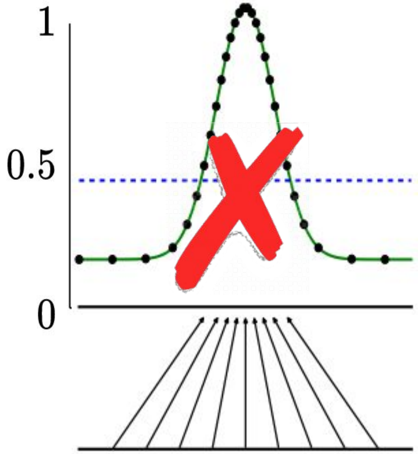
Training Objective

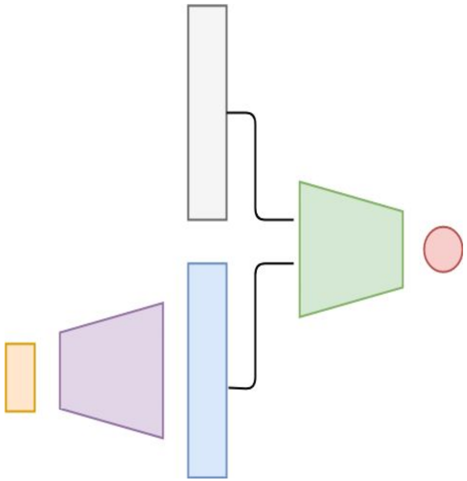
$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$



Training Instability Example (Powerful Discriminator)







Note that the generator do not have the access to the original samples.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., & Bengio, Y. (2014). Generative adversarial nets. Advances in neural information processing systems.

Uninformative Loss

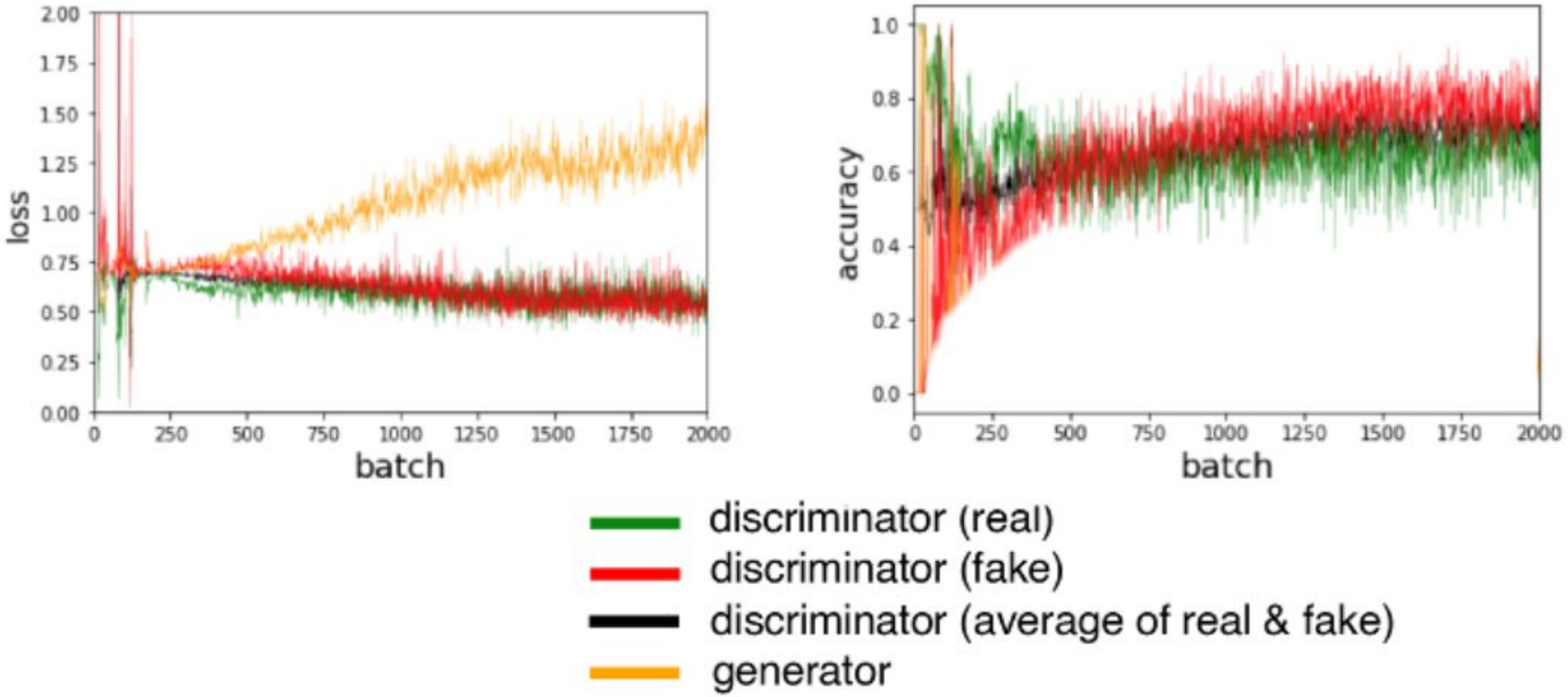


Figure 4-8. Loss and accuracy of the discriminator and generator during training

Oscillating loss in an unstable GAN

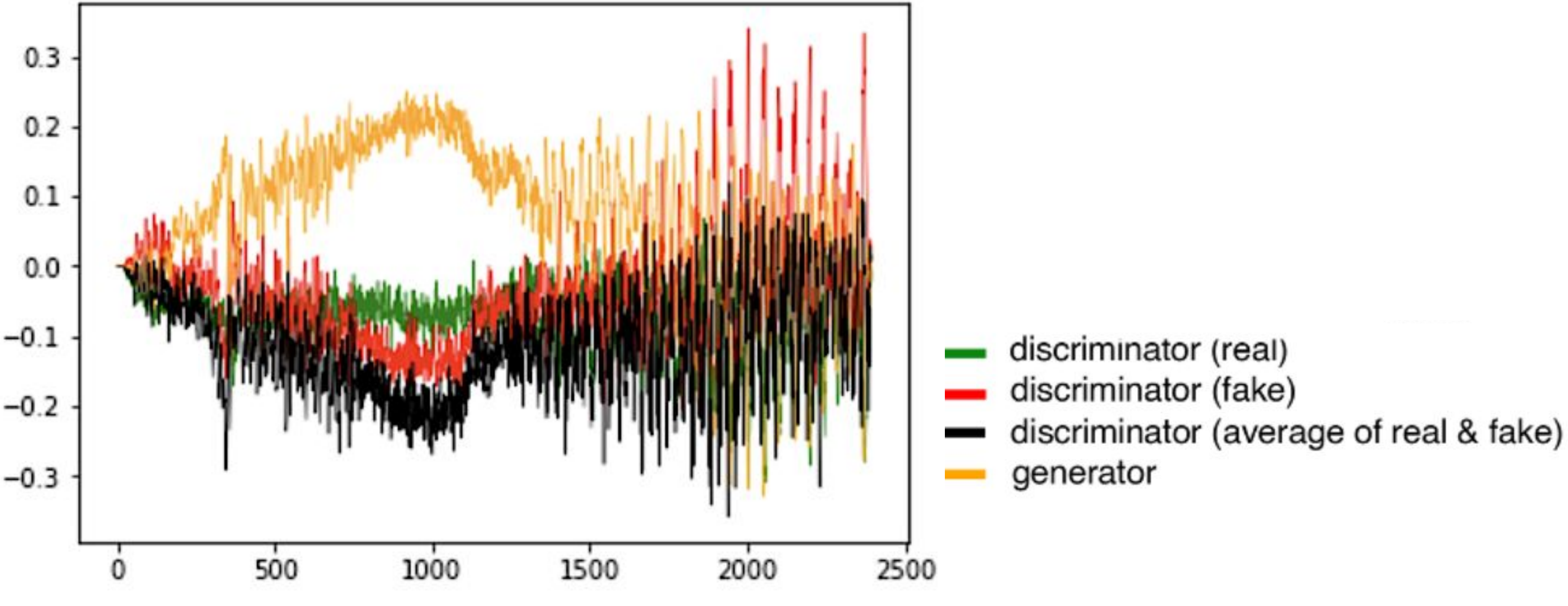
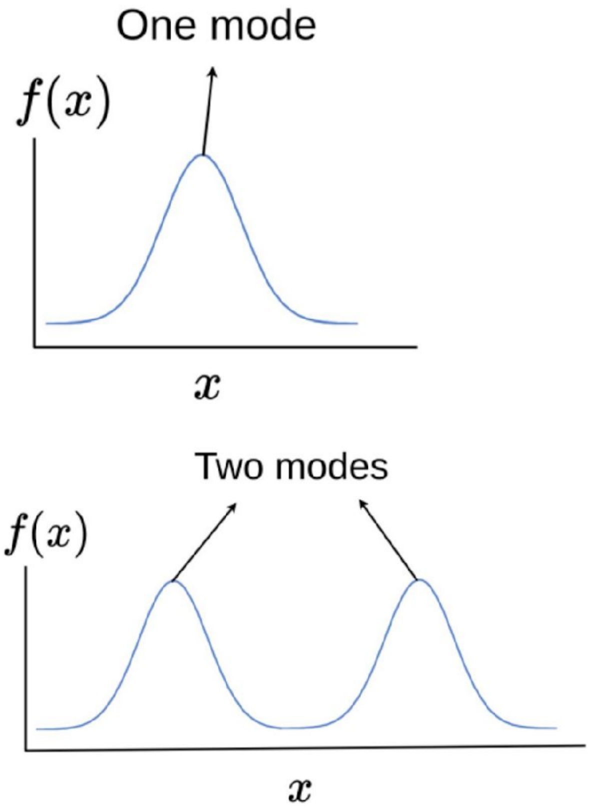
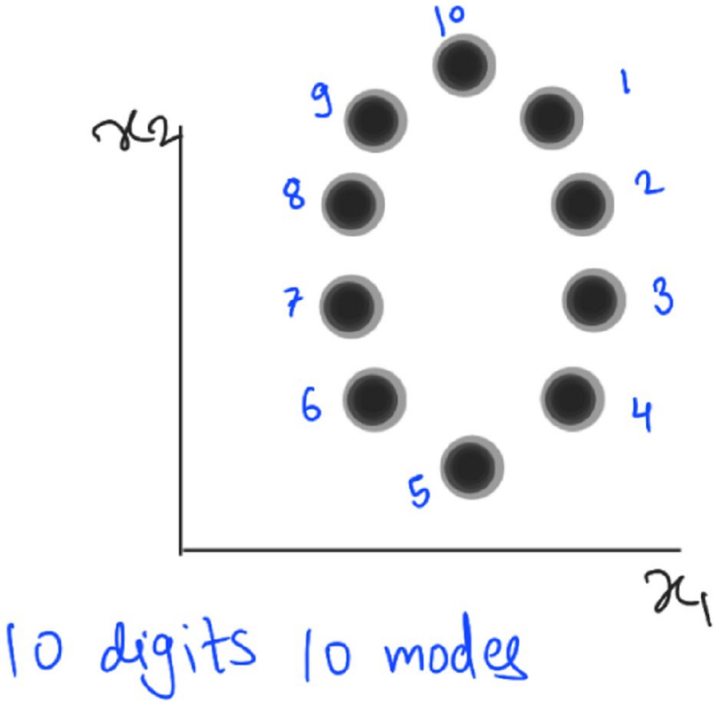


Figure 4-11. Oscillating loss in an unstable GAN

Mode

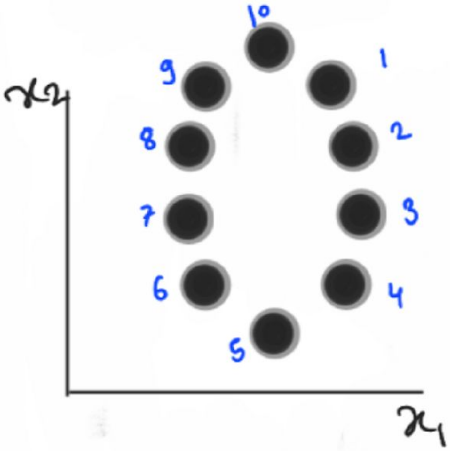

One mode
$f(x)$
x
Two modes
$f(x)$
x

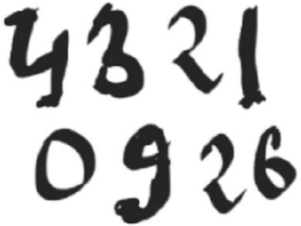

10
1
9
2
8
7
3
6
4
5
x_2
x_1
10 digits 10 modes

Mode collapse

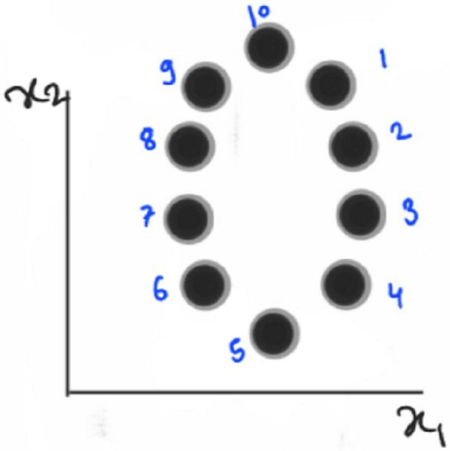
- Mode collapse is a problem where the generator can generate a limited number of images (or even just one), regardless of the latent input vector value.
- Generator tries to minimize $\mathbb{E}_{\boldsymbol{z} \sim p_z(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$ while the weights of the discriminator are fixed.
- In other words, the generator tries to generate a fake image, x*, so that $\mathbf{x}^* = \arg\max_{\mathbf{x}} D(\mathbf{x})$
- The above objective function does not force the generator to create a unique image, x*, for different values of the input latent vector, therefore model collapse.

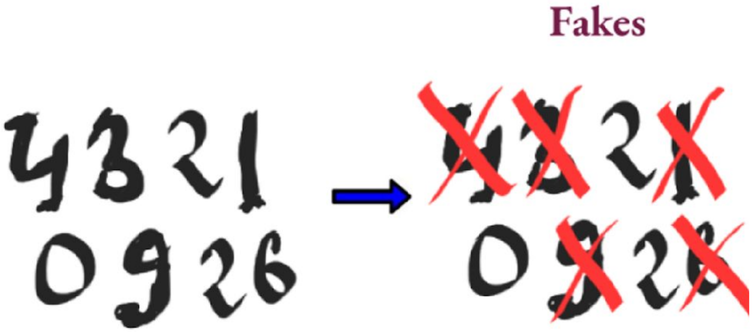
Mode collapse



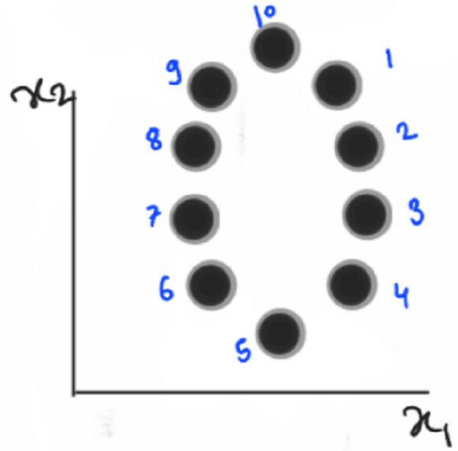


Mode collapse

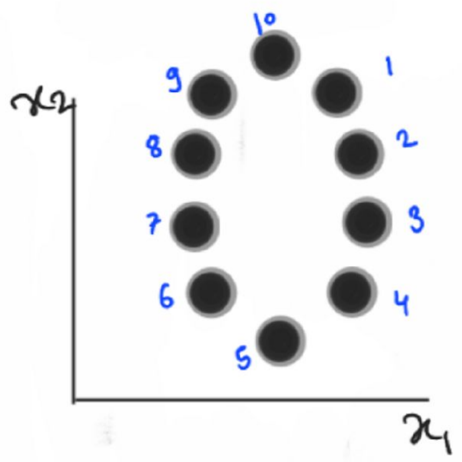


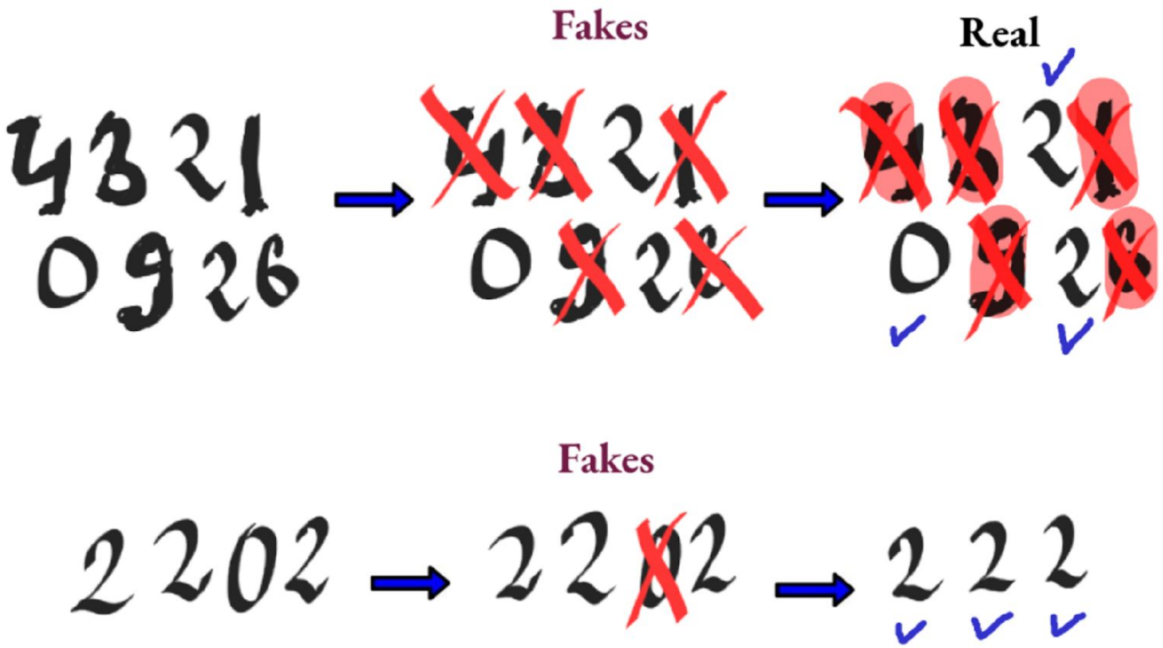


Mode collapse

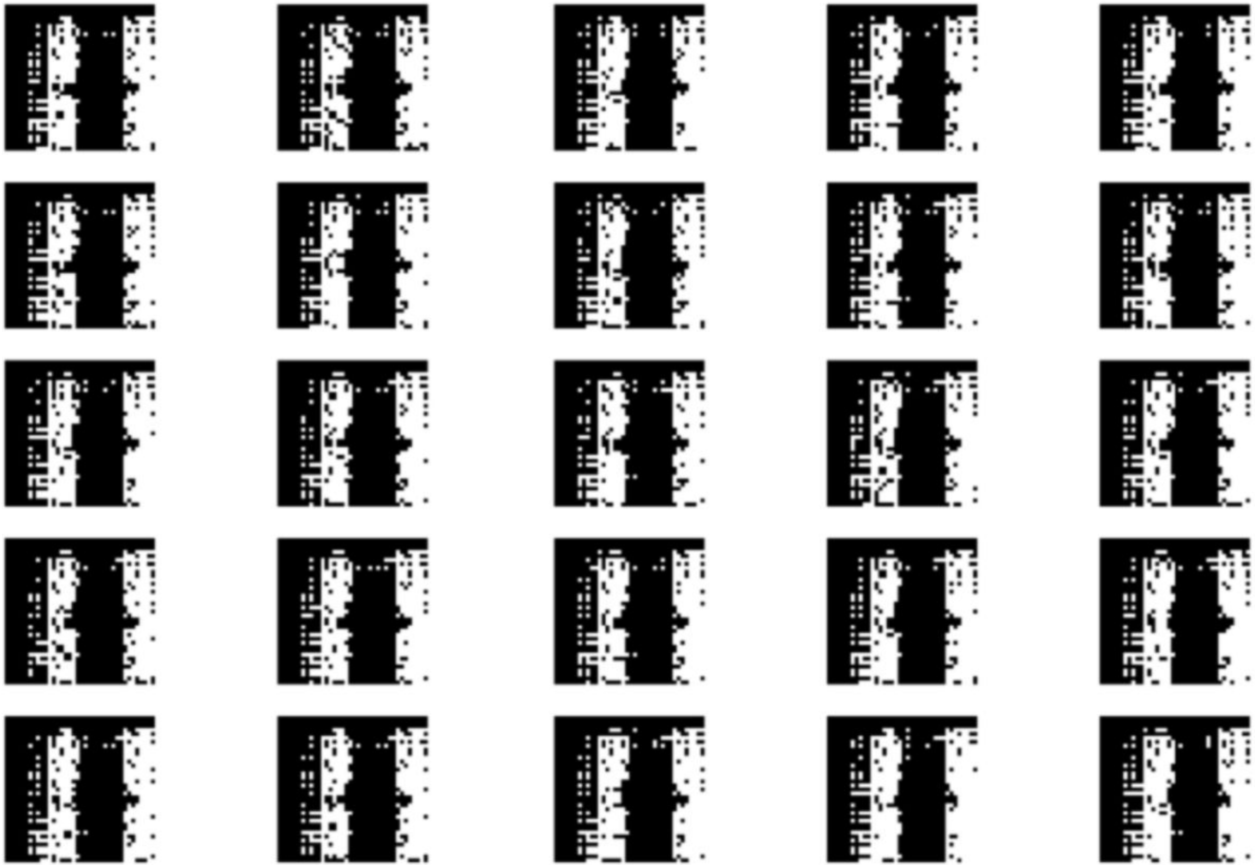


Mode collapse





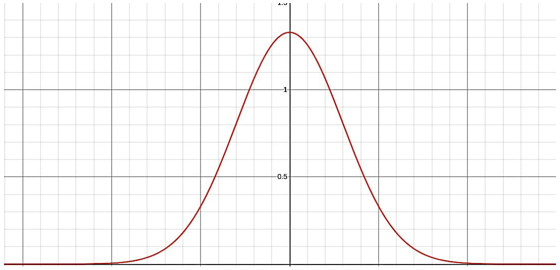
Mode Collapse (Ganimal)



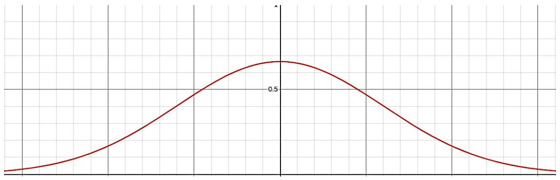
Mode Collapse (How to approach)

- Use an objective function that is different from the original GAN objective, which compares distributions from real and fake data with a different distance. For example, Wasserstein GAN uses the Wasserstein distance.
- Adding Noise: Injects noise into the inputs or labels of the discriminator to destabilize its confidence and encourage generator diversity.

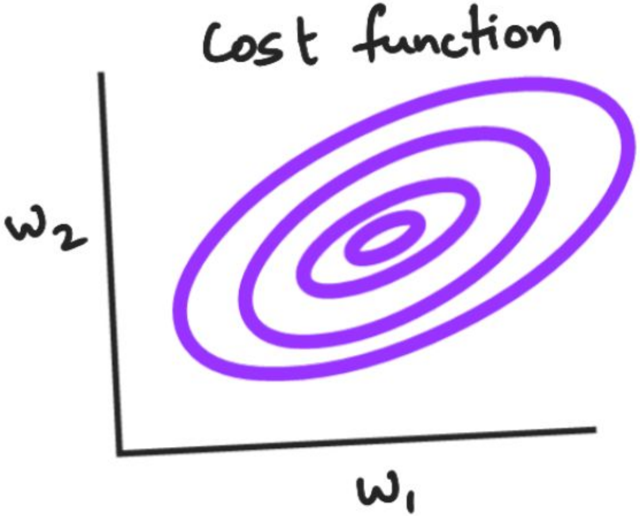
Internal Covariate Shift



x_1



x_2



Batch normalization helps to reduce internal covariate shift.

Ioffe, S., & Szegedy, C. (2015, June). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning* (pp. 448-456). PMLR.

Internal Covariate Shift

- An internal covariate shift occurs when the distribution of each layer's inputs changes during training, as the parameters of the previous layers change.
- When the input distribution changes, hidden layers try to learn to adapt to the new distribution. This slows down the training process.

Ioffe, S., & Szegedy, C. (2015, June). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning* (pp. 448-456). PMLR.

Sensitivity to hyperparameter initialization

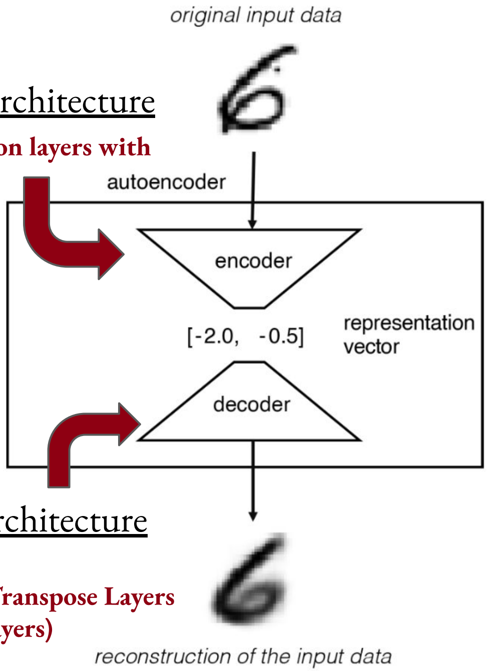
- Are GANs created equal? The study suggests that many models can reach similar FID scores with enough hyperparameter optimization and random restarts (having high computational budget)
- However, most models are very sensitive to hyperparameter initialization.

Lucic, Mario, et al. "Are gans created equal? a large-scale study." Advances in neural information processing systems 31 (2018).

Artifacts due to implementation

<u>Encoder Architecture</u>

Convolution layers with Stride=2



<u>Decoder Architecture</u>

Convolution Transpose Layers (upsampling layers)

Figure 4-6. Artifacts when using convolutional transpose layers[3]

Artifacts when using Convolutional Transpose Layers.

Slow convergence

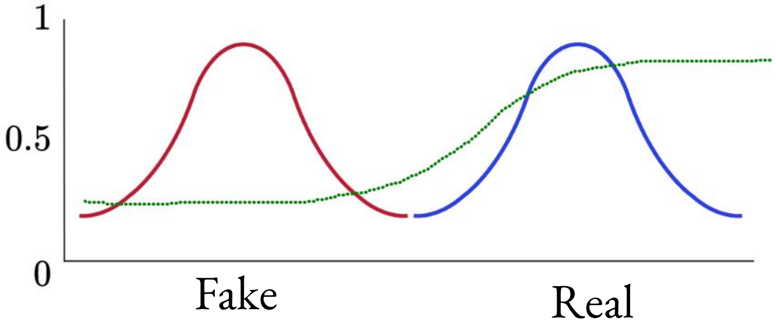
This is a big problem with GANs and unsupervised settings. Generally the speed of convergence is slow (e.g., InGAN, and SinGAN).

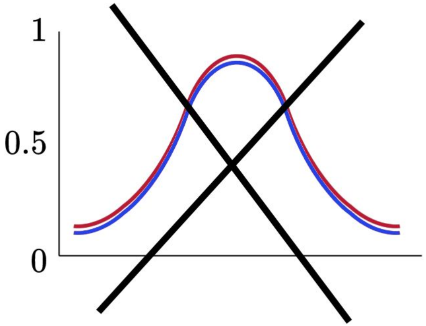
Overgeneralization

- GAN start producing outputs samples (i.e., modes) that semantically should not exist.
- For example, a cow with multiple bodies but only one head, or vice versa. It happens when the GAN overgeneralizes and learns things that should not exist based on the real data.

Other Challenges

- Lack of a Proper Evaluation Metric





- Disjoint support between fake images and real images: A possibly effective solution for disjoint distributions is to add limited noise to real images (Instance Noise).

GAN Frameworks and Architectures

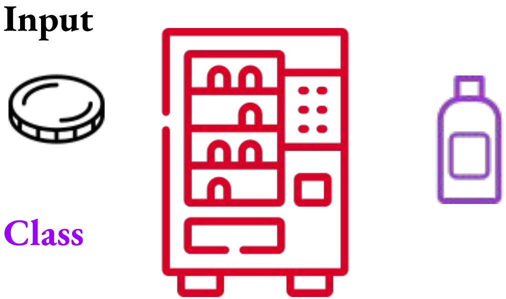
GANs Frameworks

Unconditional output

You get item from a random class

Unconditional GANs

Conditional output



You select the item you want

Conditional GANs

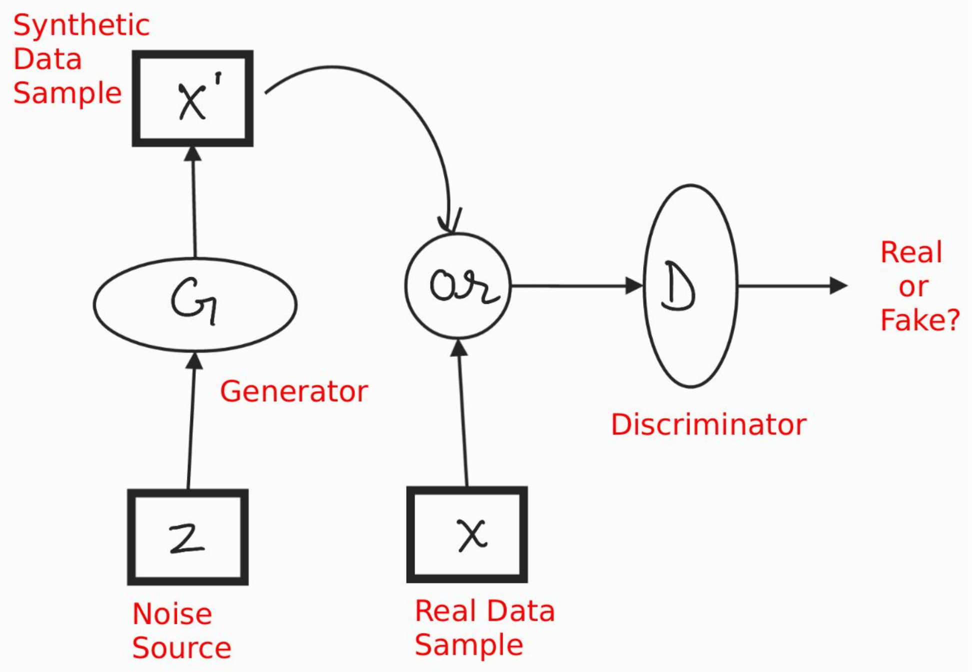
GANs Frameworks

Conditional GANs	Unconditional GANs
Examples from the classes you want	Examples from the random classes
Training dataset needs to be labeled	Training dataset does not need to be labeled

Unconditional GANs

Unconditional GANs (Recall)

- **Generator** is trained to map a ***noise*** sample ***to*** a ***synthetic*** data sample that can "fool" the discriminator.
- **Discriminator** is trained to distinguish real data samples from synthesized samples.

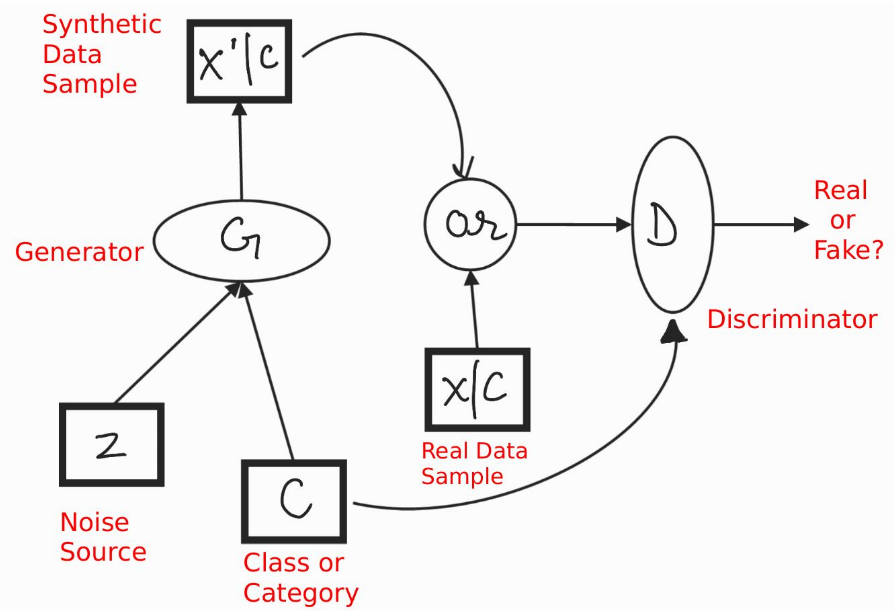


Conditional GANs

Conditional GANs

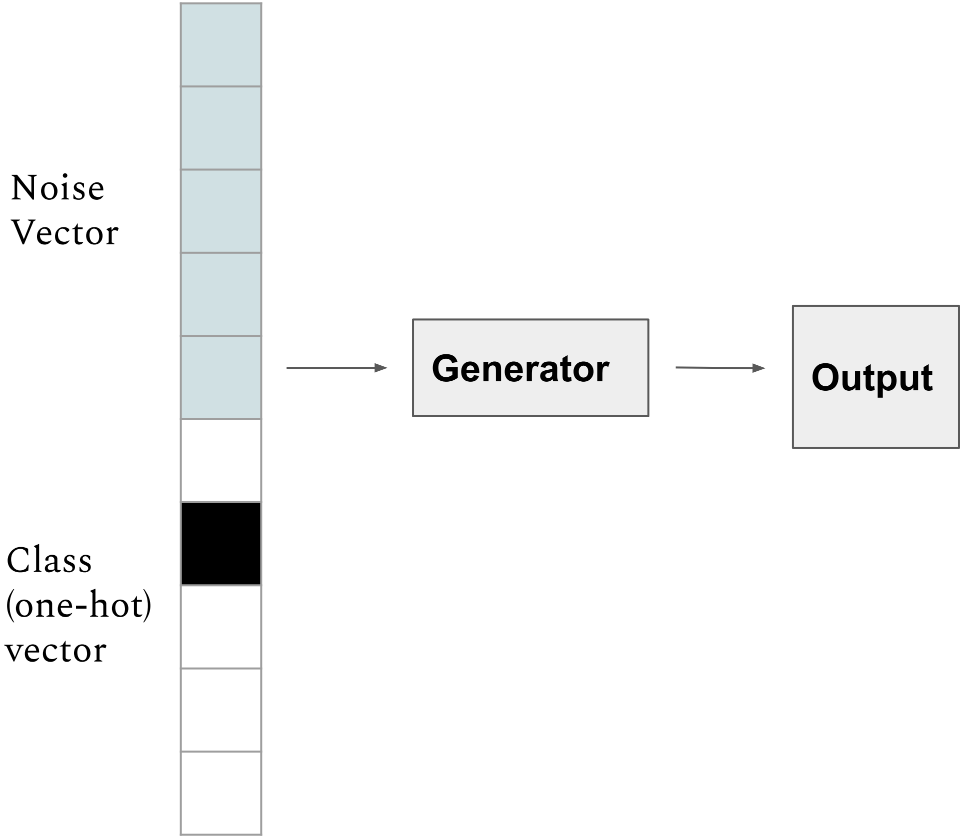
- **Generator** Must learn to create class conditional image samples. So Generator takes Noise and Class Label as input and map to Synthetic Sample.

- **Discriminator** is trained to distinguish real data from synthesized samples, conditional on class or category.



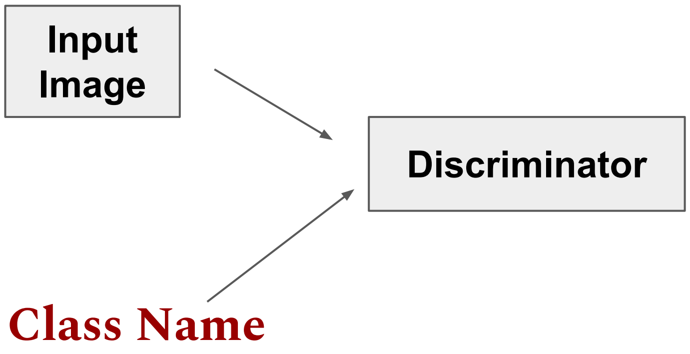
Conditional GAN Generator Input

- The class is passed to the generator as one-hot vectors, where the size of the vector represents the number of classes.



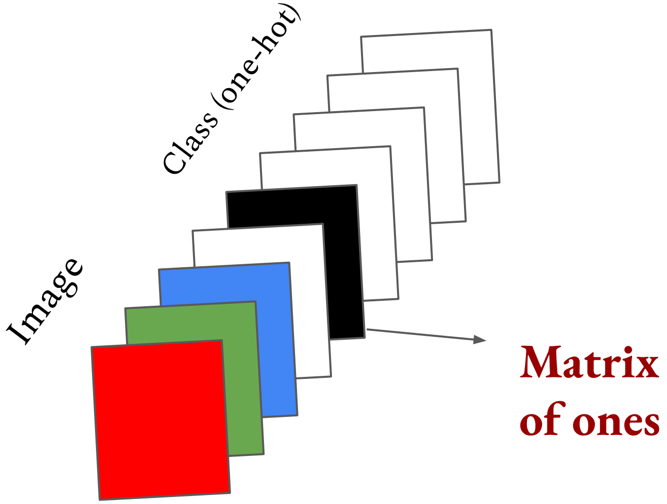
Conditional GAN Discriminator Input

- The class is passed to the discriminator as one-hot matrices, where the number of matrices represents the number of classes.



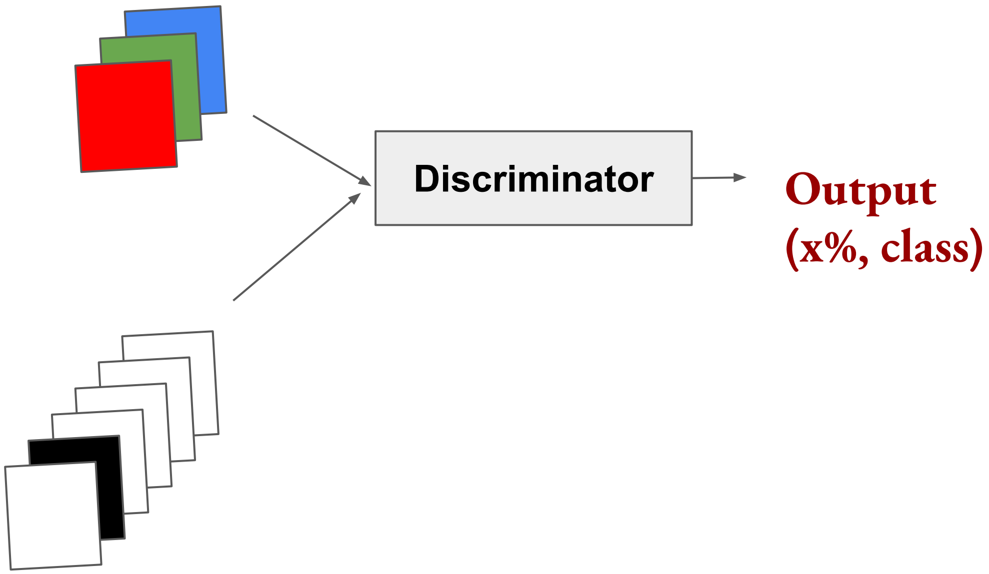
Conditional GAN Discriminator Input

- The class is passed to the discriminator as one-hot matrices, where the number of matrices represents the number of classes.

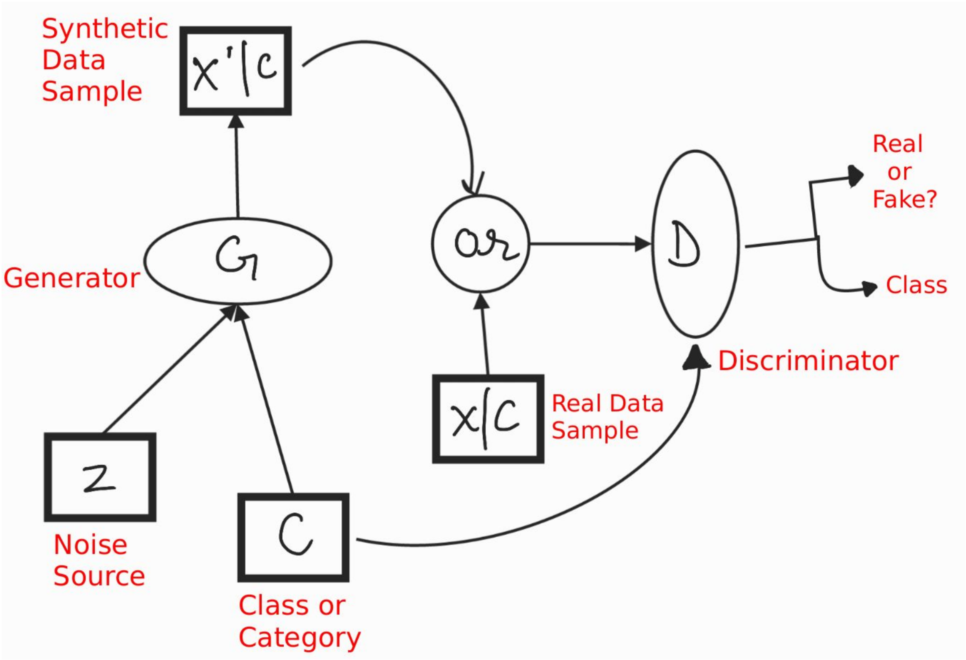


Conditional GAN Discriminator Input

- The class is passed to the discriminator as one-hot matrices, where the number of matrices represents the number of classes.



Conditional GAN



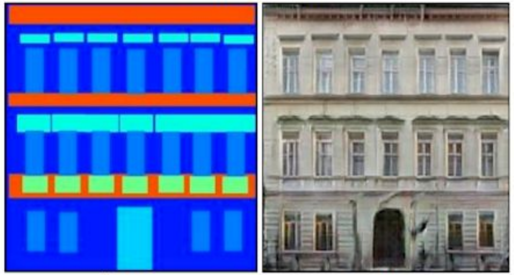
$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x}|\boldsymbol{y})] + \mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}|\boldsymbol{y})))]$$

Creswell, Antonia, et al. "Generative adversarial networks: An overview." IEEE Signal Processing Magazine 35.1 (2018): 53-65.

Image-to-Image Translation (Pix2Pix)

Image-to-image translation

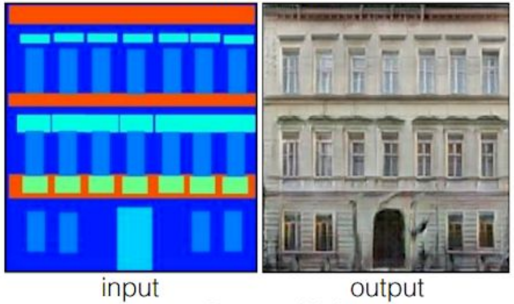
- Image-to-image translation transforms images from one domain to domain.

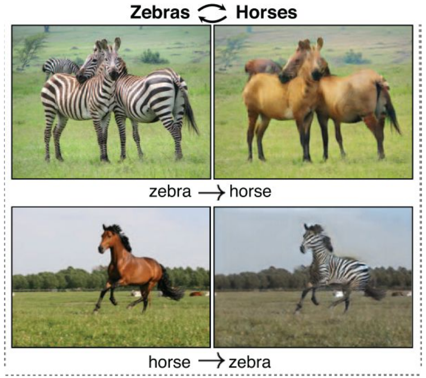


Isola, Phillip, et al. "Image-to-image translation with conditional adversarial networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

Image-to-image translation

- Image-to-image translation transforms images from one domain to domain.
- GANs provides realistic image transformation.



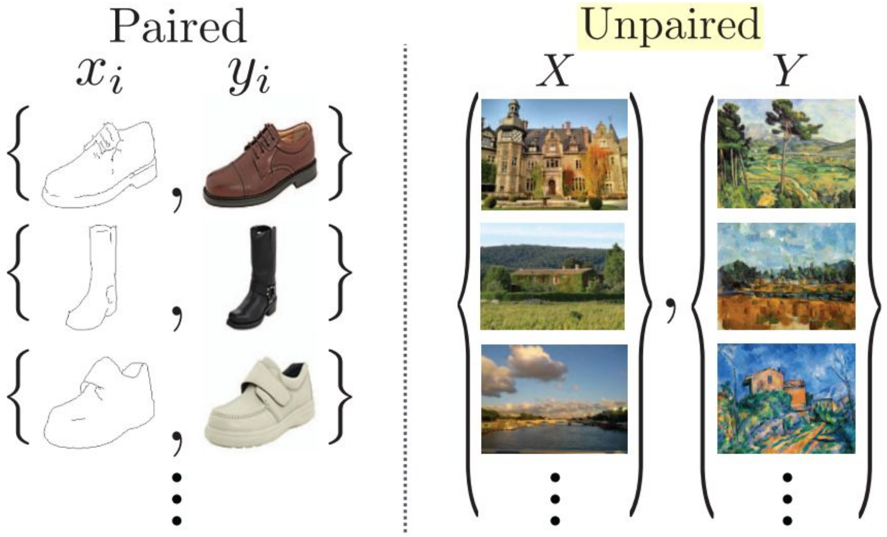


Isola, Phillip, et al. "Image-to-image translation with conditional adversarial networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

Image-to-image translation

Paired

Unpaired



Zhu, Jun-Yan, et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks." Proceedings of the IEEE international conference on computer vision. 2017.

Image-to-image translation

Paired

Pix2Pix

Unpaired

CycleGAN

Image-to-image translation

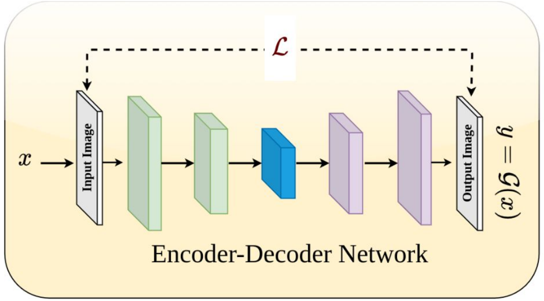
Paired

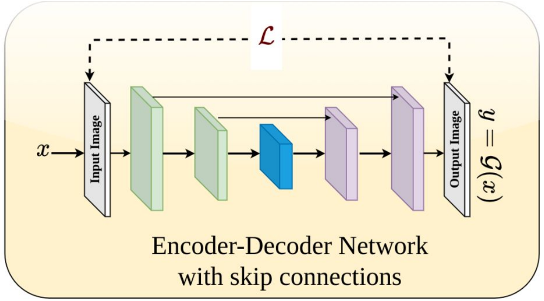
<u>Pix2Pix</u>

- Pix2Pix Generator
- Pix2Pix Discriminator
- Objective Function (Conditional GAN objective and Pixel Loss)
- Pix2Pix Results
- Pix2Pix Future Work

Pix2Pix Tools

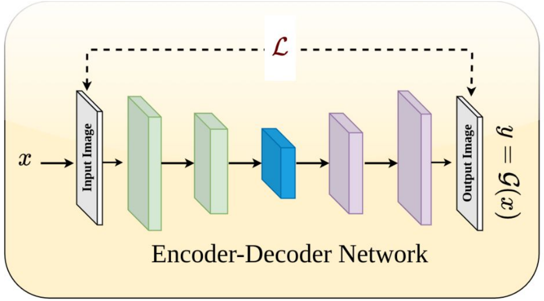
- Pix2Pix **Generator** is U-Net
- U-Net is an Encoder-Decoder network, with same-size inputs and outputs

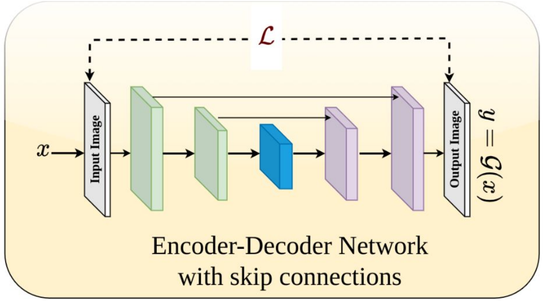




Pix2Pix Tools

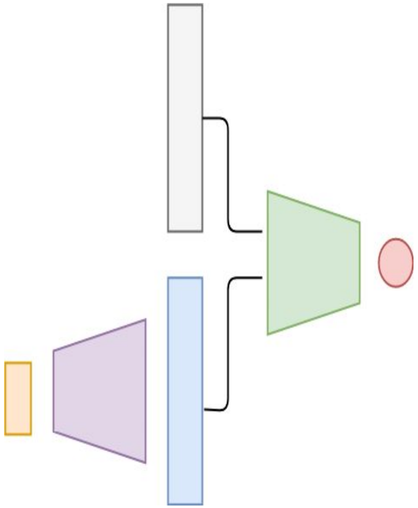
- Pix2Pix **Generator** is U-Net
- U-Net is an Encoder-Decoder network, with same-size inputs and outputs



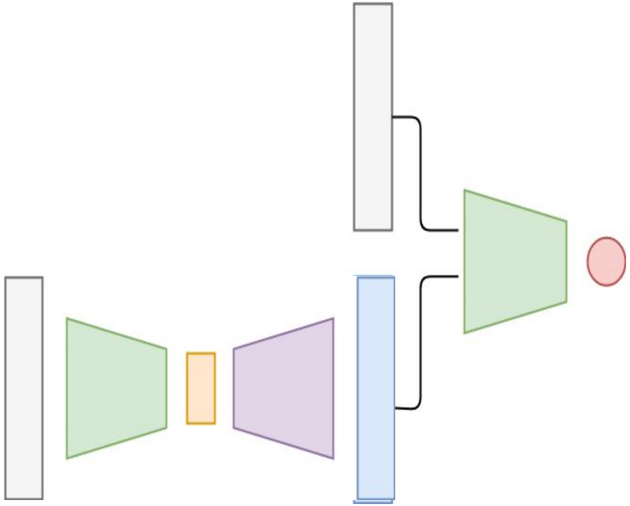


- Pix2Pix **Discriminator** is Patch Discriminator

Generator as Encoder-Decoder Network

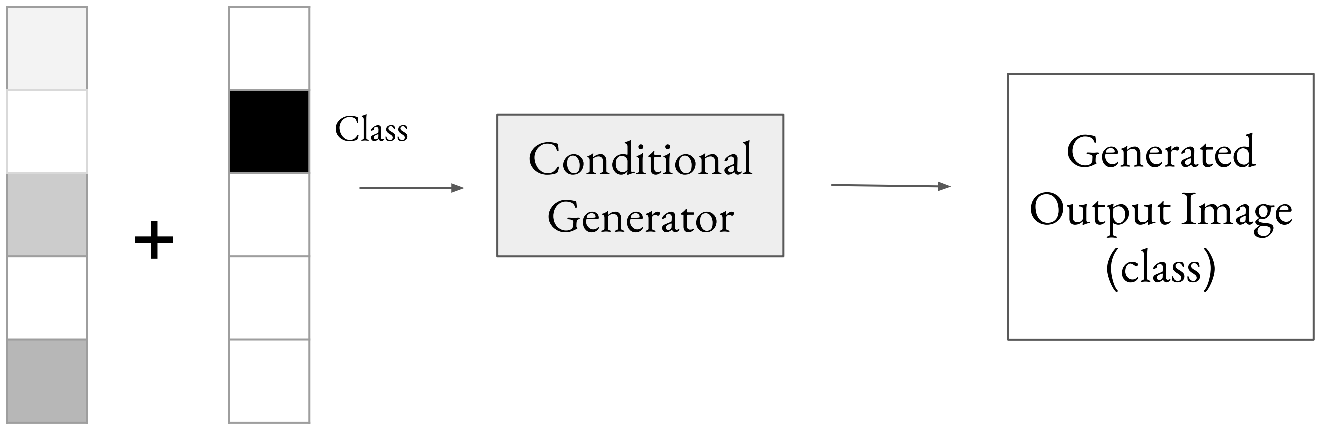


Generator as simple CNN



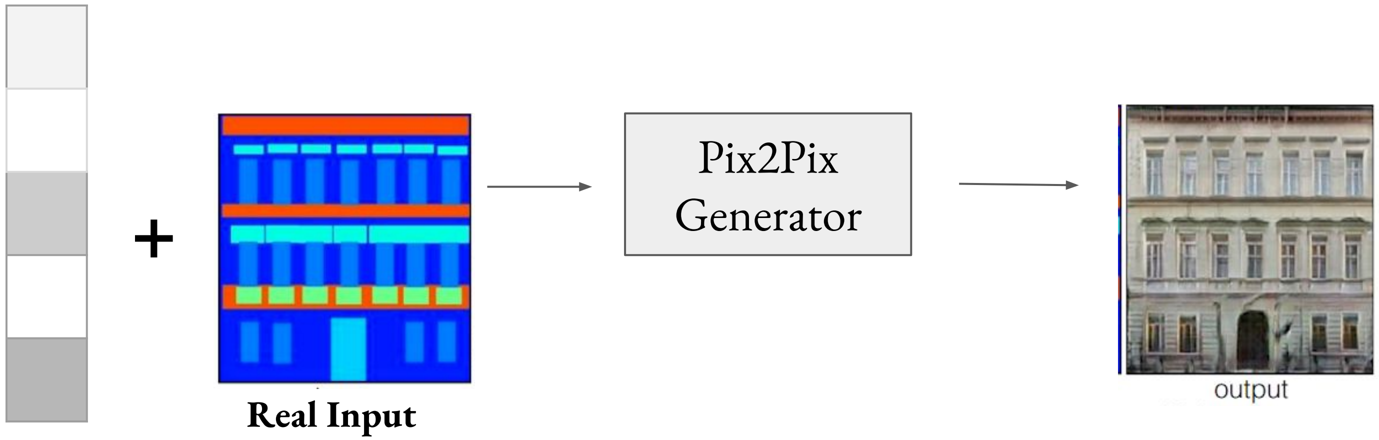
**Generator as
Encoder-Decoder Network**

Conditional Generator



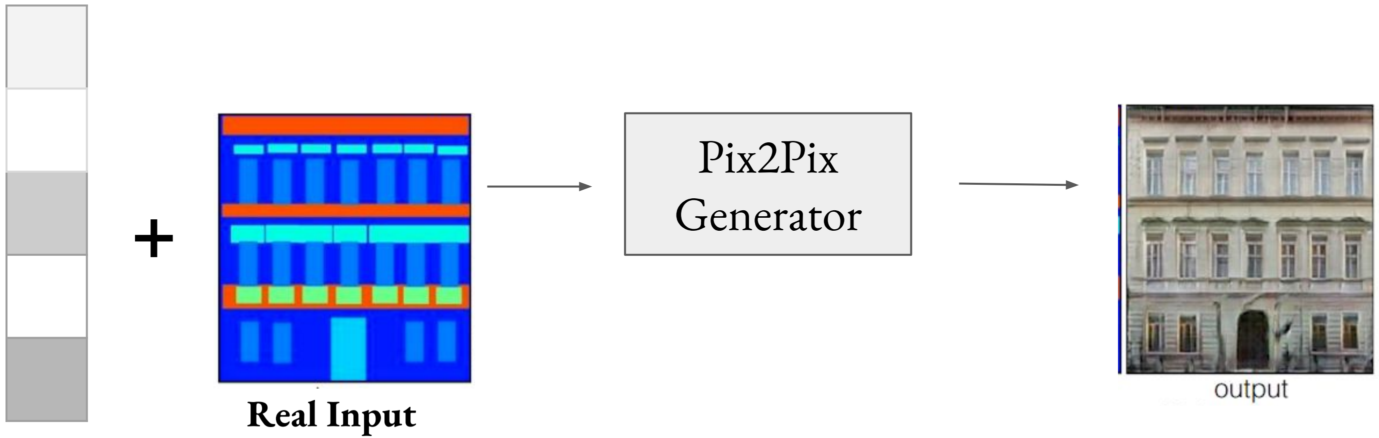
- Conditional GAN takes the class as input and outputs image with the desired class.

Pix2Pix Generator



- Inputs and outputs of Pix2Pix are similar to conditional GAN
  - Take in the original image, instead of the class vector

Pix2Pix Generator



- Inputs and outputs of Pix2Pix are similar to conditional GAN
  - Take in the original image, instead of the class vector
- Noise vector input to generator does not greatly influence performance. Generator is able to learn input to output mapping conditioned on the type (style)

Pix2Pix Generator (Skip Connections)

- U-Net uses skip connections.
- Skip connections help the decoder learn details from the layers of the encoder directly.

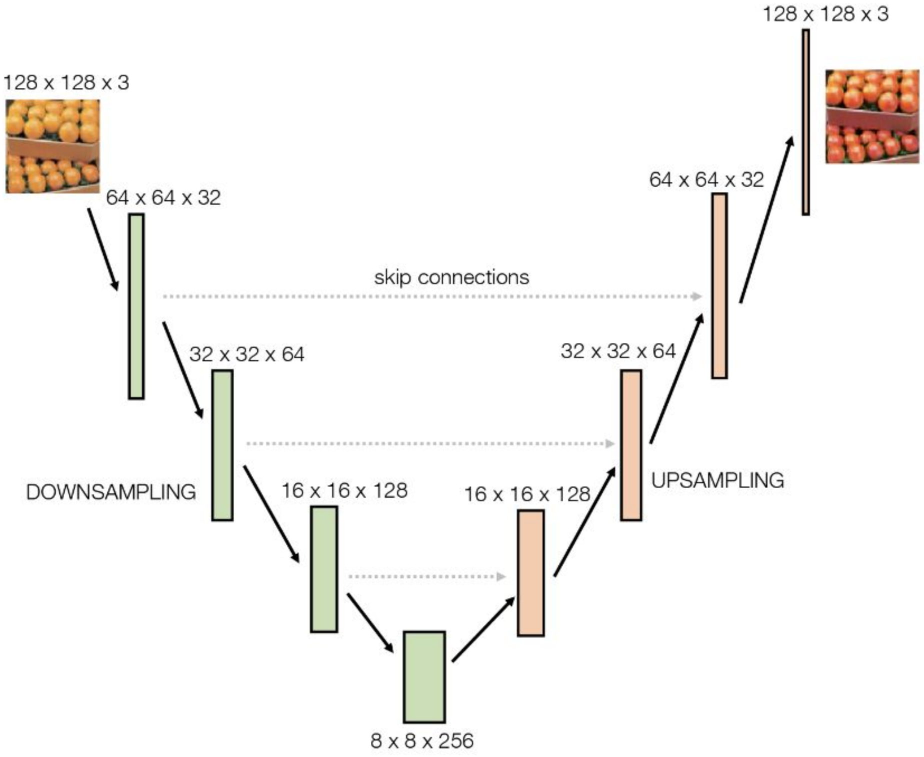
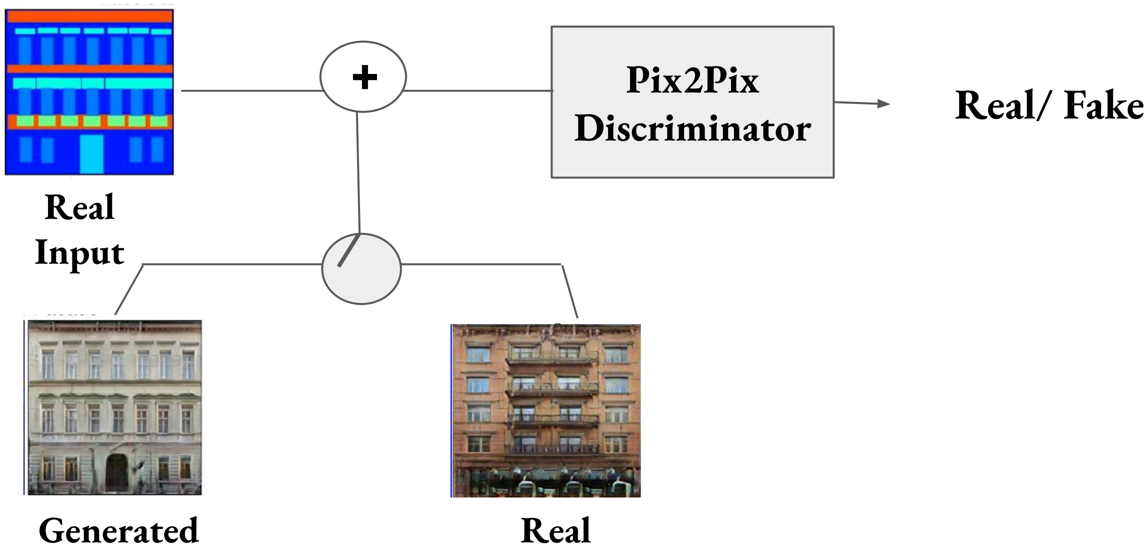


Figure 5-6. The U-Net architecture diagram

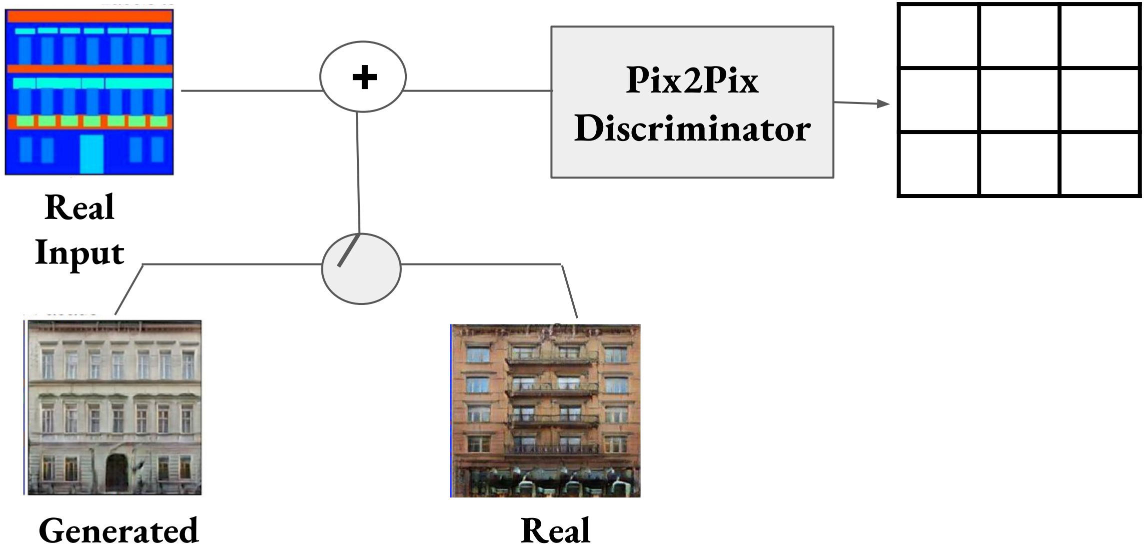
Pix2Pix Discriminator

Discriminator goal is to distinguish between Real image and Generated Fake image



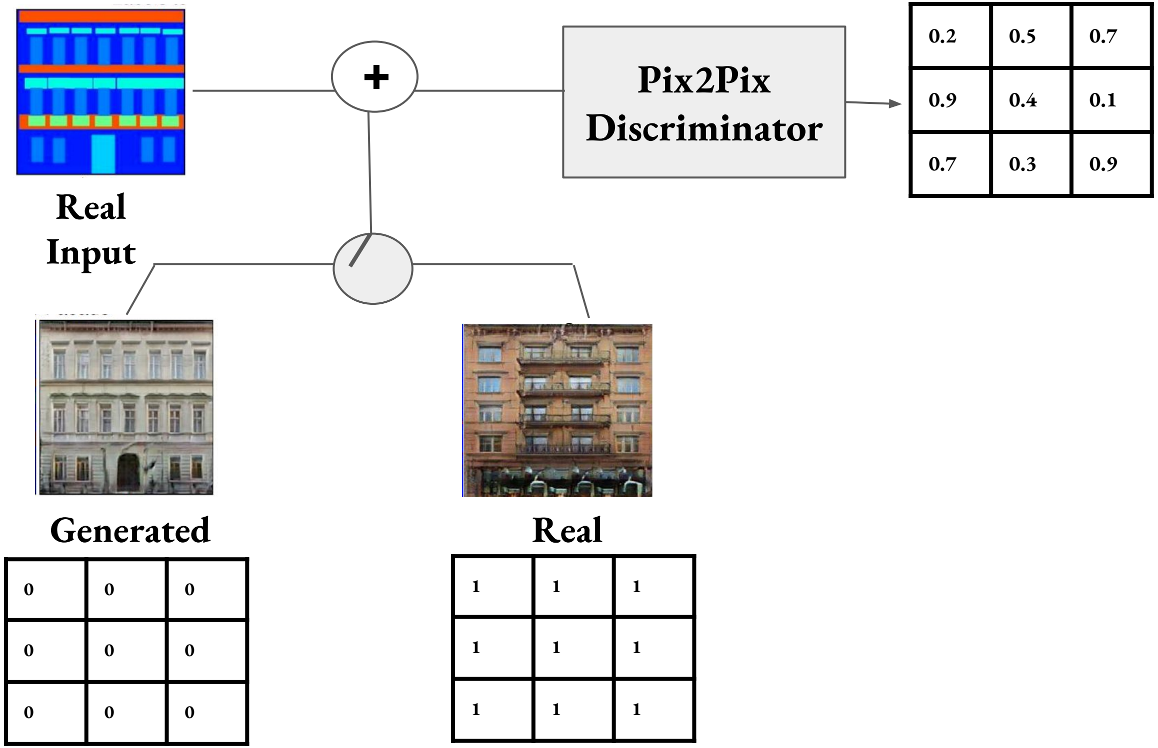
Pix2Pix Discriminator

- Patch discriminator outputs a matrix of values, each between 0 and 1.



Pix2Pix Discriminator

- Patch discriminator outputs a matrix of values, each between 0 and 1.
- Label matrices: 0's = Fake and 1's = Real.



Pix2Pix Objective Function

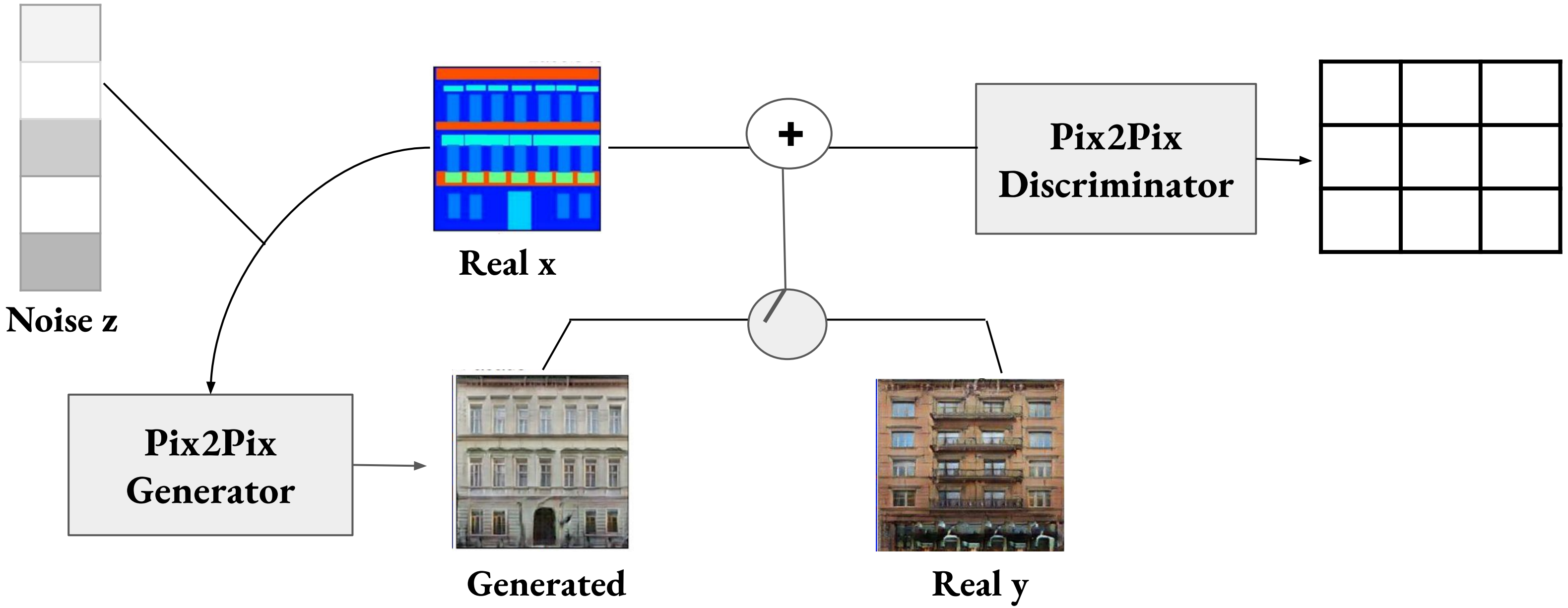
$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G).$$

Conditional GAN objective

Pixel Loss

Conditional GAN objective

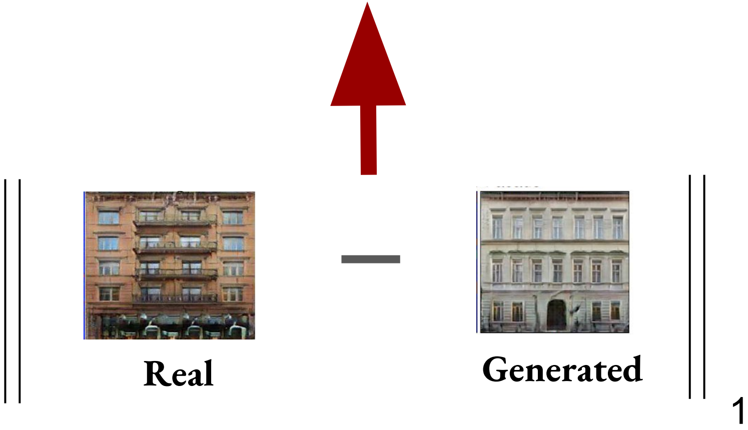
$$\mathcal{L}_{cGAN}(G,D) = \mathbb{E}_{x,y}[\log D(x,y)] + \mathbb{E}_{x,z}[\log(1 - D(x, G(x,z)))]$$



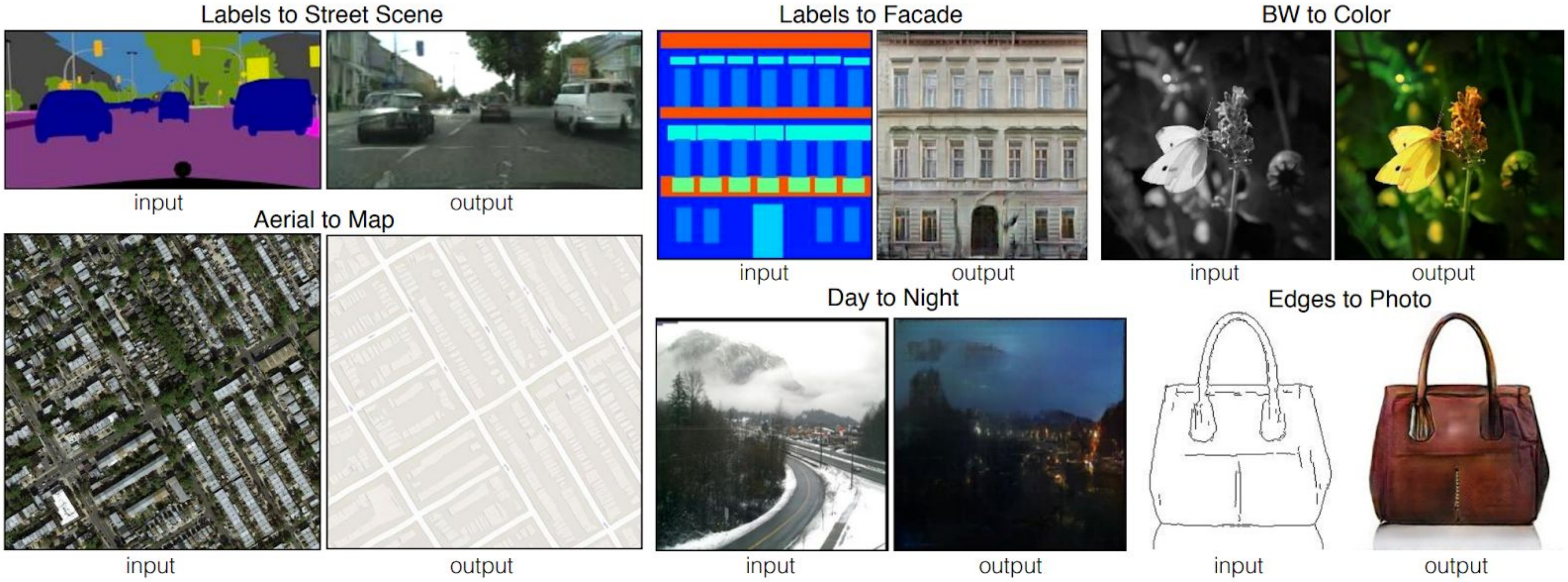
Pixel Loss

- Pix2Pix adds a Pixel Distance Loss term to the generator function.
- This loss term calculates difference between the fake and the real target outputs.
- Pixel Loss allows more sharpe image and less blurring.

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



Pix2Pix Results

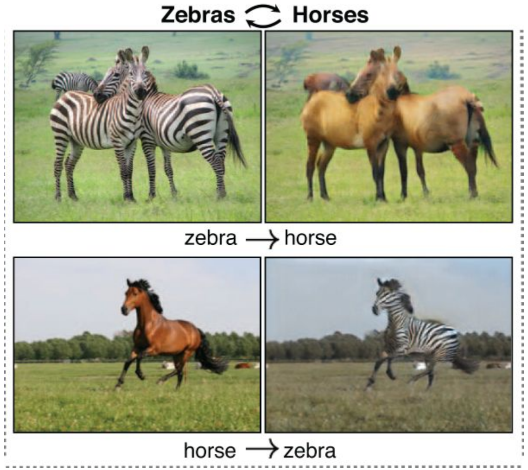


Isola, Phillip, et al. "Image-to-image translation with conditional adversarial networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

Cycle GAN

Outline

- Pix2Pix and Cycle GAN
- CycleGAN Overview
- Objective Function (GAN Loss and CycLoss)
- CycleGAN Applications



Pix2Pix and Cycle GAN

- Unpaired image-to-image translation
  - Learns mapping between two piles of images
  - Examines common elements of the two piles (content) and unique elements of each pile (style)

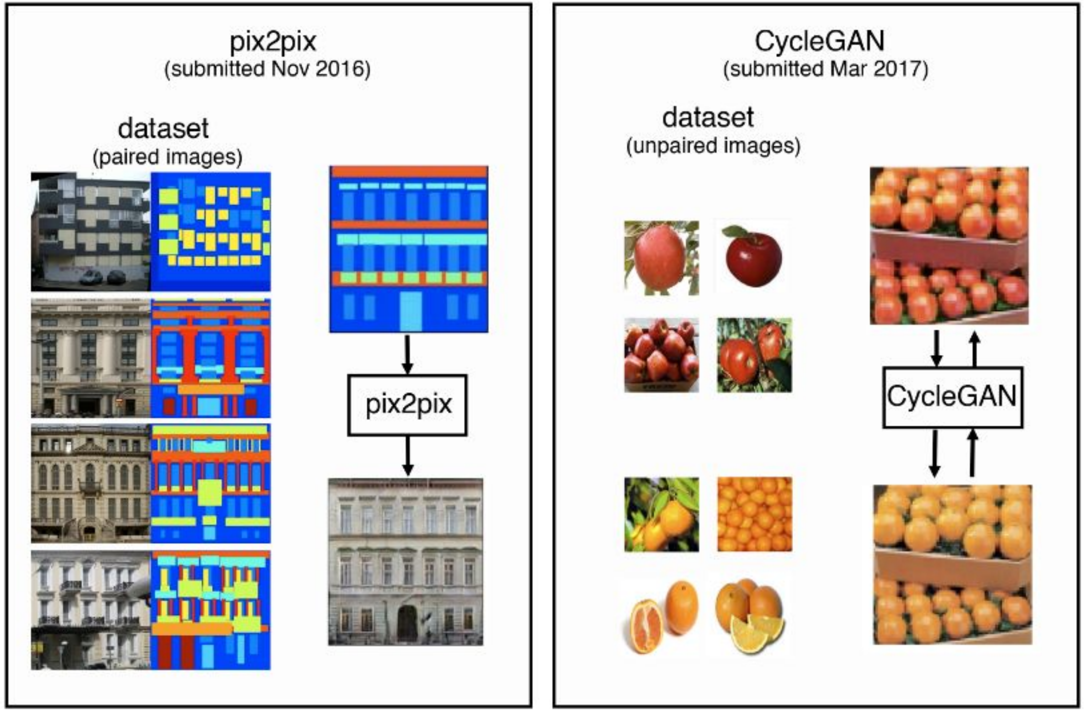
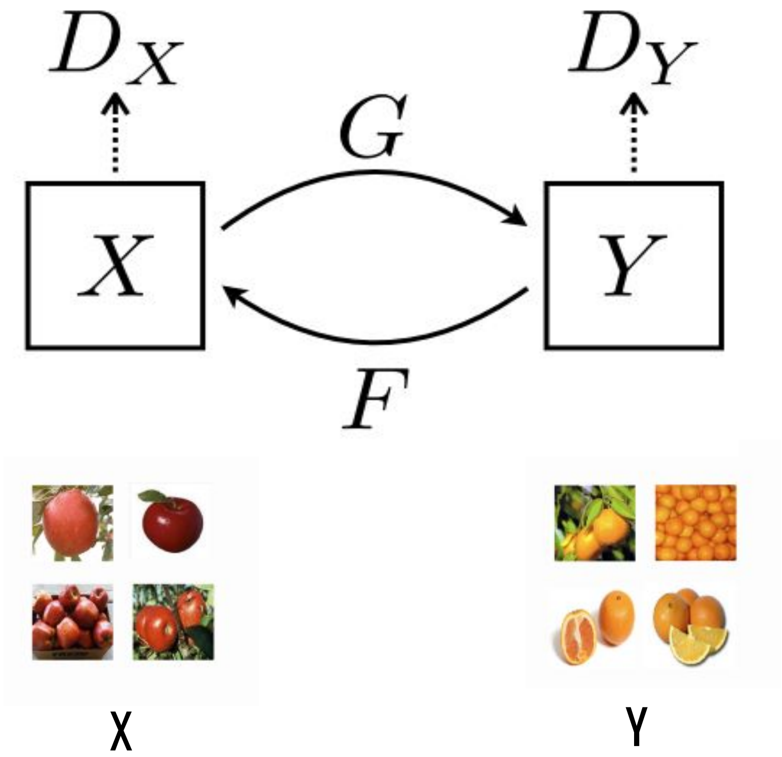


Figure 5-4. pix2pix dataset and domain mapping example

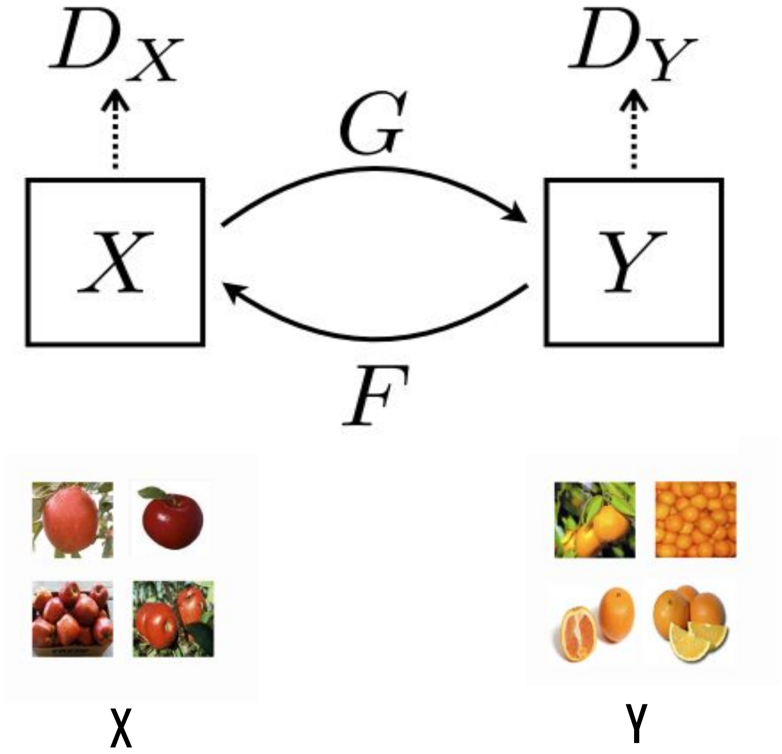
Cycle GAN Overview

CycleGAN uses **Two GANs** for unpaired image-to-image translation.



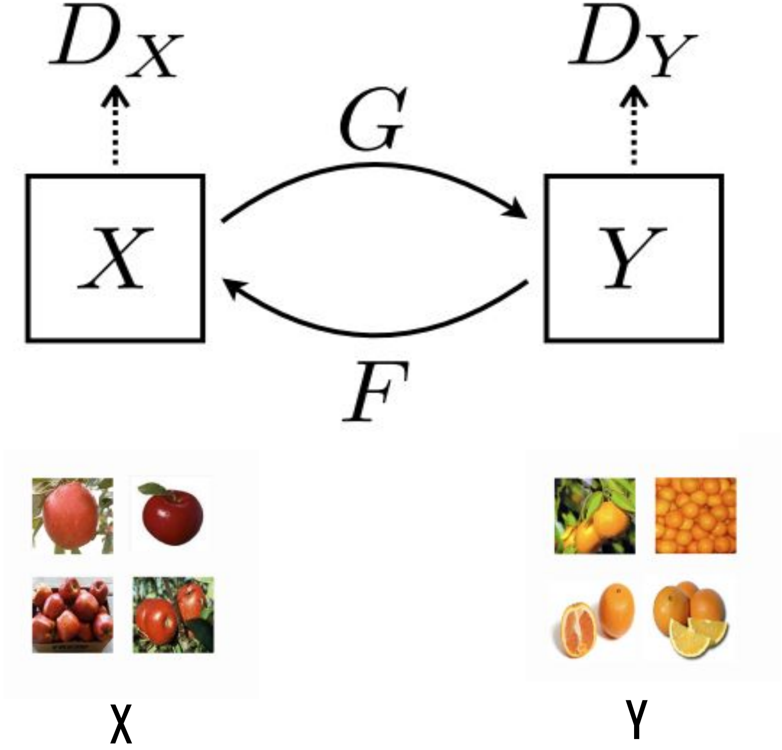
Cycle GAN Overview

- CycleGAN has four components:
    - **Two Generators**: The generators are similar to a U-Net
    - **Two Discriminators**: The discriminators are PatchGAN's



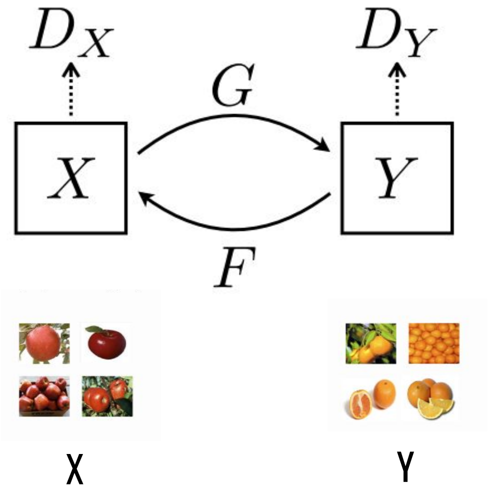
Cycle GAN Overview

- The inputs to the generators and discriminators are similar to Pix2Pix, except each discriminator is in charge of one pile of images.



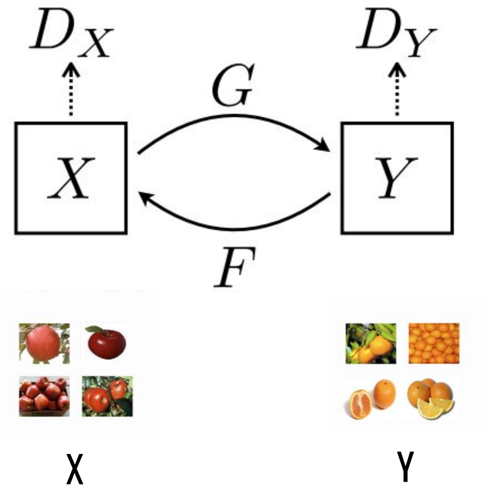
Formal Description

- CycleGAN allows us to do unsupervised image-to-image translation, from two domains X <-> Y
- Specifically, we learn **Two GANs,** G: X → Y and F: Y→ X, where G and F are two Generators.



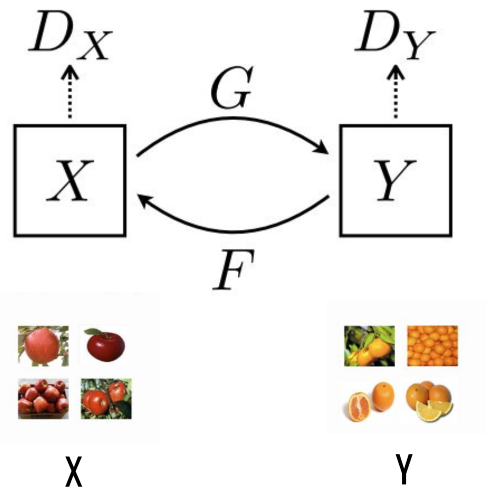
Formal Description

- CycleGAN allows us to do unsupervised image-to-image translation, from two domains X <-> Y
- Specifically, we learn **Two GANs,** G: X → Y and F: Y→ X, where G and F are two Generators.
- There are **Two Discriminators** D_X and D_Y associated with generators G and F.
    - D_X is associated with F, and compares true X and generated sample F(Y).
    - D_Y is associated with G, and compares true Y and generated sample G(X).

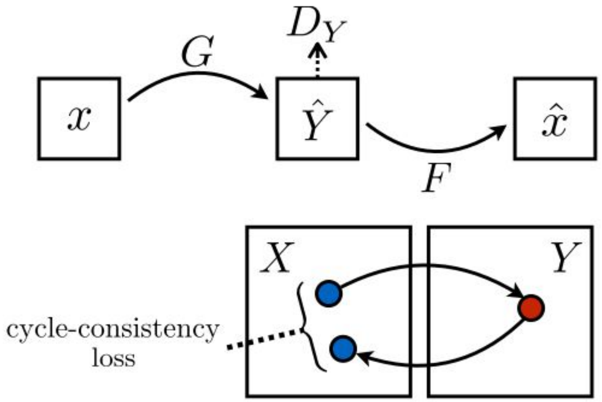


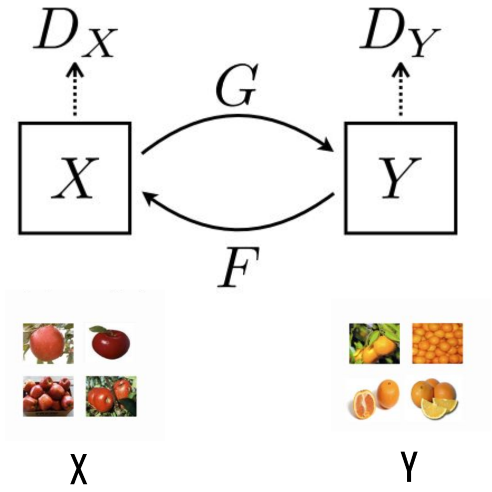
Cycle Consistency

- Cycle Consistency state that if we can go from X to Ŷ via G, then we should be able to go from Ŷ to X via F.
- Cycle consistency is used in both directions.
- Cycle consistency helps transfer uncommon style elements between the two GANs, while maintaining common content.



Forward cycle-consistency

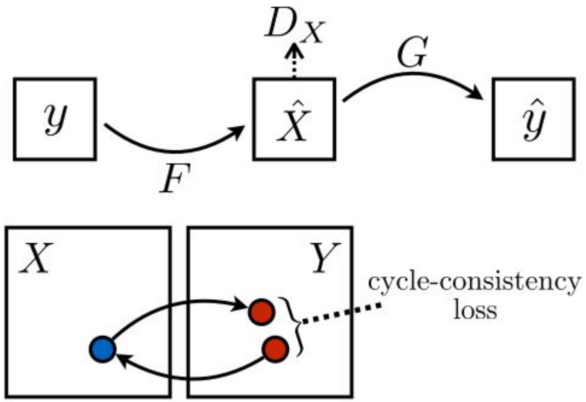


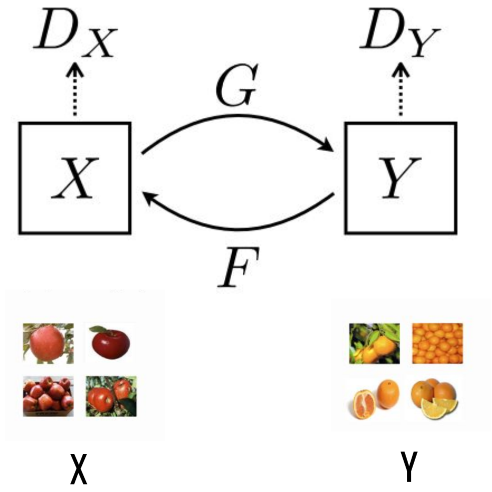


Forward cycle-consistency

x -> G(x) -> F(G(x)) ≈ x

Backward cycle-consistency

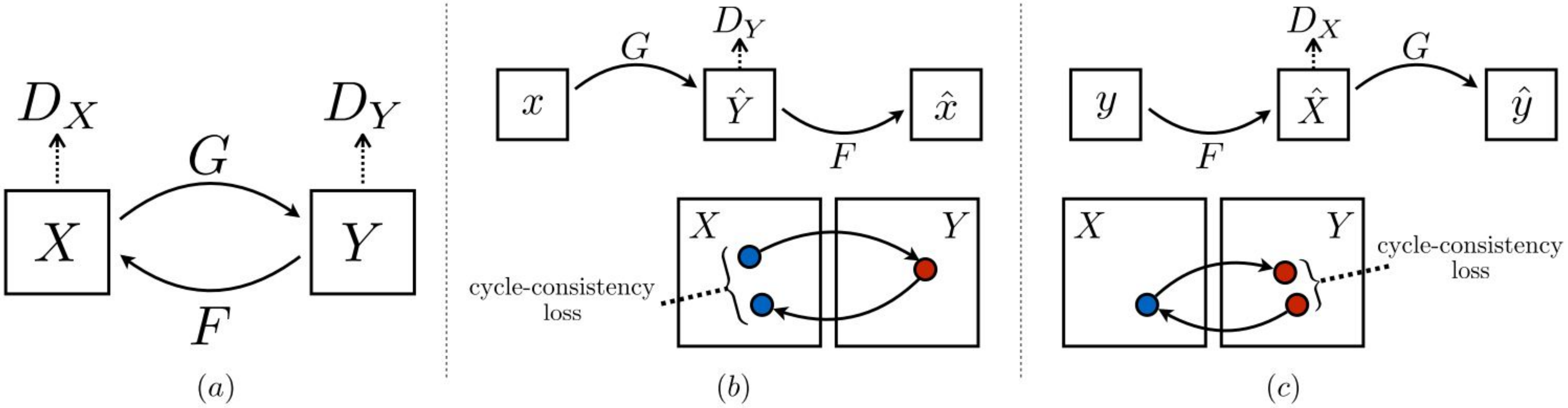




Backward cycle-consistency

y -> F(y) -> G(F(y)) ≈ y

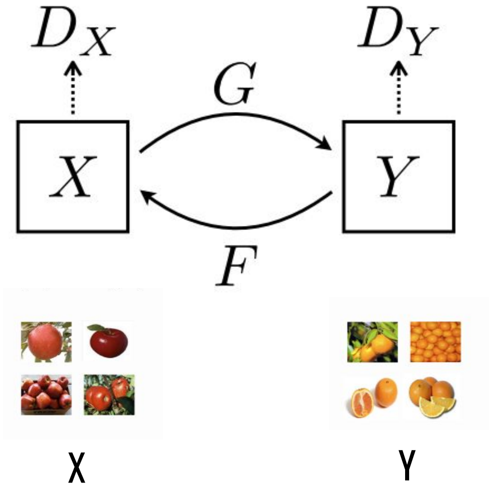
Cycle GAN Key Idea



$$G^*, F^* = \arg\min_{G,F} \max_{D_x, D_Y} \mathcal{L}(G, F, D_X, D_Y)$$

Objective Function

$$G^*, F^* = \arg\min_{G,F}\max_{D_x,D_Y} \mathcal{L}(G, F, D_X, D_Y)$$

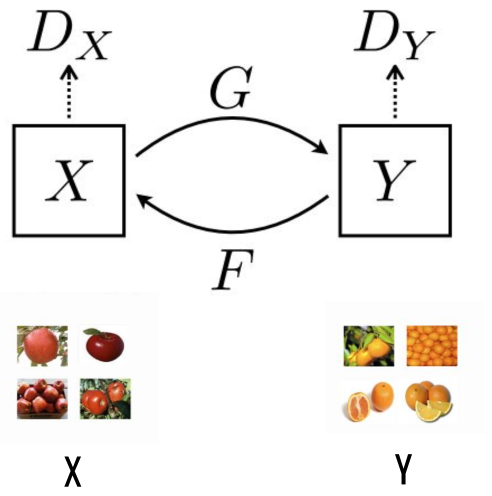


full objective is:

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

Objective Function

$$G^*, F^* = \arg\min_{G,F}\max_{D_x,D_Y}\mathcal{L}(G,F,D_X,D_Y)$$



full objective is:

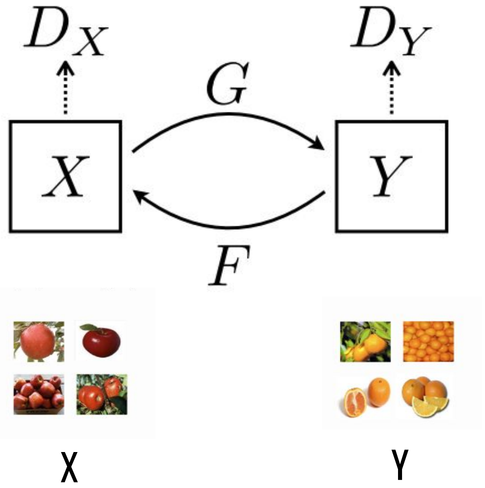
$$\begin{aligned}\mathcal{L}(G,F,D_X,D_Y) =& \mathcal{L}_{\text{GAN}}(G,D_Y,X,Y)\\ &+ \mathcal{L}_{\text{GAN}}(F,D_X,Y,X)\\ &+ \lambda\mathcal{L}_{\text{cyc}}(G,F),\end{aligned}$$

GAN Loss

Objective Function

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

$\mathcal{L}_{\text{GAN}}(F, D_X, Y, X)$?



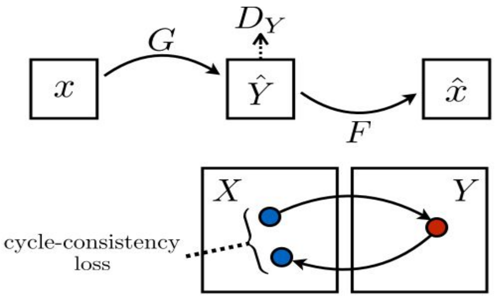
full objective is:

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

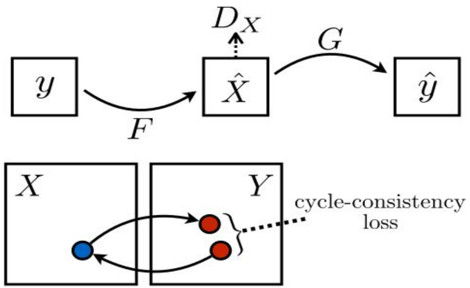
GAN Loss

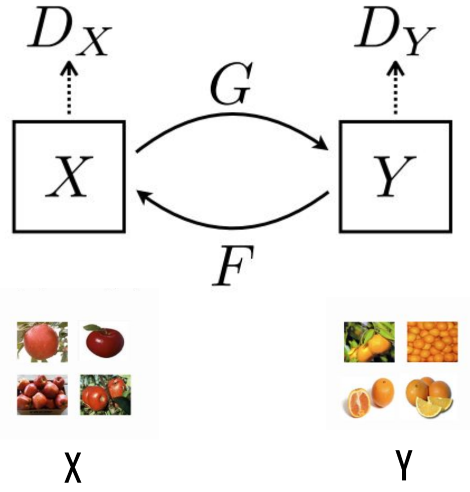
Objective Function

Forward cycle-consistency loss



Backward cycle-consistency loss





full objective is:

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

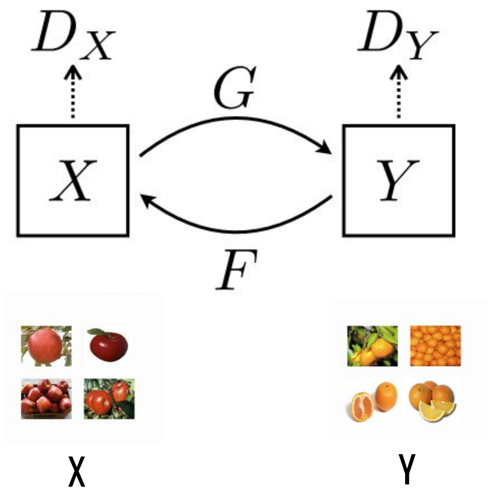
Cycle Consistency Loss

Objective Function

Forward cycle-consistency loss

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

Backward cycle-consistency loss

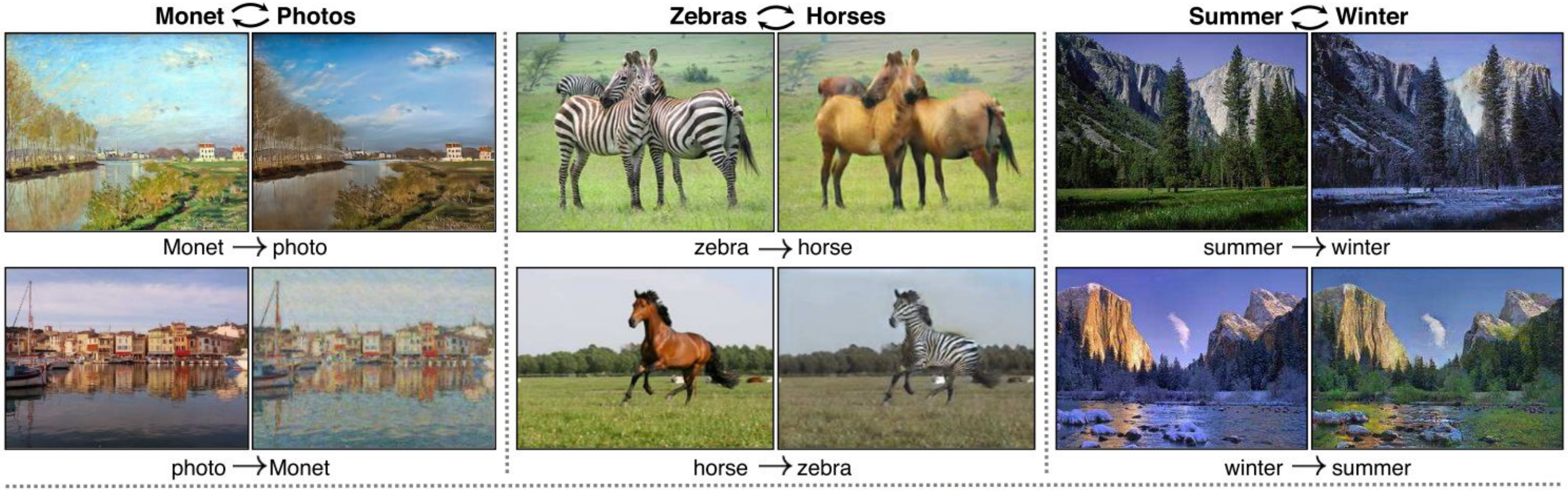


full objective is:

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

Cycle Consistency Loss

CycleGAN Applications



CycleGAN failure Case

horse → zebra

References

- Generative Deep Learning: Teaching Machines to Paint, Write, Compose, and Play. By David Foster. O'Reilly Media.
- Zhu, Jun-Yan, et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks." Proceedings of the IEEE international conference on computer vision. 2017.