Image-to-Image Translation

Gurdit Singh Siyan
gurdit@cmi.ac.in

Chennai Mathematical Institute

May 7, 2022

Introduction

What is Image-to-Image Translation?

- Many problems in image processing, graphics, and vision involve transforming an input image into a corresponding output image.

What is Image-to-Image Translation?

- Many problems in image processing, graphics, and vision involve transforming an input image into a corresponding output image.
- There exist a wide variety of domain specific solutions for many problems, even though the goal is always the same: mapping pixels to pixels.

Image-to-Image Translation

Examples

Figure: Colorization: Black & White $\rightarrow$ Color

Image-to-Image Translation

Examples

Figure: Semantic Segmentation: Cityscape $\rightarrow$ Segmented Image

Image-to-Image Translation

Examples

Figure: Style Transfer: Portrait $\rightarrow$ Stylized Image

Image-to-Image Translation

Examples

Figure: Background Removal: Image $\rightarrow$ Image without Background

Towards a General Purpose Solution

- The goal is always to find a map from an input domain X to an output domain Y.

Towards a General Purpose Solution

- The goal is always to find a map from an input domain X to an output domain Y.
- That is, given X and Y, we want to find a function $G : X \to Y$.

Towards a General Purpose Solution

- The goal is always to find a map from an input domain X to an output domain Y.
- That is, given X and Y, we want to find a function $G : X \to Y$.
- Can we build a general solution to this problem?

Towards a General Purpose Solution

- The goal is always to find a map from an input domain X to an output domain Y.
- That is, given X and Y, we want to find a function $G : X \to Y$.
- Can we build a general solution to this problem?
- Yes! We can utilize a class of machine learning models known as GANs.

Background: Generative Adversarial Networks

Generative Adversarial Networks

Definition

- Generative Adversarial Networks (Goodfellow et al., 2014) — also known as GANs—are a form of generative machine learning model which learn via an adversarial process.

Generative Adversarial Networks

Definition

- Generative Adversarial Networks (Goodfellow et al., 2014) — also known as GANs—are a form of generative machine learning model which learn via an adversarial process.
- GANs consist of two sub models: a generator G and a discriminator D.

Generative Adversarial Networks

Definition

- Generative Adversarial Networks (Goodfellow et al., 2014) — also known as GANs—are a form of generative machine learning model which learn via an adversarial process.
- GANs consist of two sub models: a generator G and a discriminator D.

Generative Adversarial Networks

Definition

- Generative Adversarial Networks (Goodfellow et al., 2014) — also known as GANs—are a form of generative machine learning model which learn via an adversarial process.
- GANs consist of two sub models: a generator G and a discriminator D.
  - The goal for G is to generate some target distribution. from a noise vector z.

$$G : Z \to X$$

Generative Adversarial Networks

Definition

- Generative Adversarial Networks (Goodfellow et al., 2014) — also known as GANs—are a form of generative machine learning model which learn via an adversarial process.
- GANs consist of two sub models: a generator G and a discriminator D.
  - The goal for G is to generate some target distribution. from a noise vector z.

$$G : Z \to X$$

  - And the goal for D is to estimate the probability that a given sample was generated by the training distribution rather than the generator G.

$$D : X \to [0, 1]$$

Generative Adversarial Networks

Example

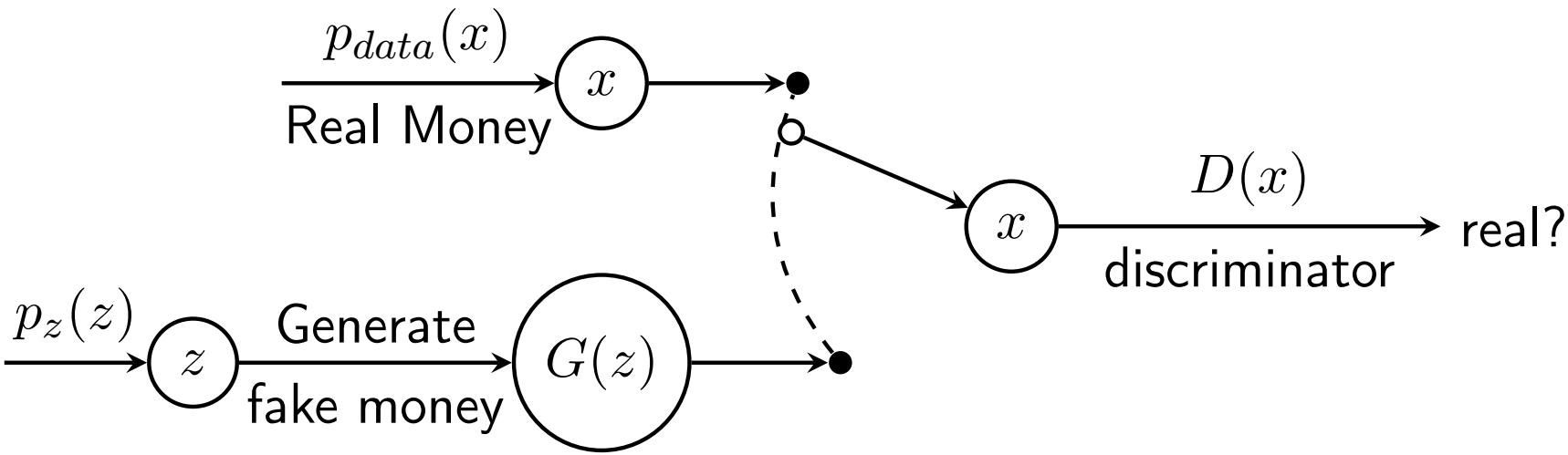


Figure: A visual representation of Generative Adversarial Network

- Canonical Example

Generative Adversarial Networks

Example

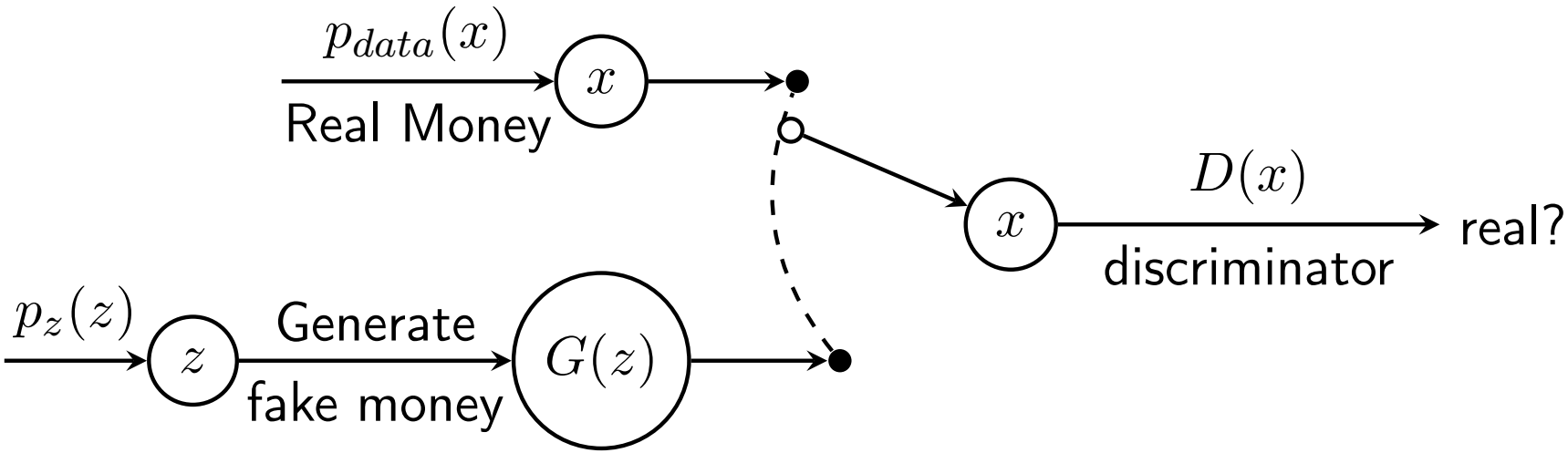


Figure: A visual representation of Generative Adversarial Network

- Canonical Example
  - Think of G as a fraud trying to create fake currency and use it without detection.

Generative Adversarial Networks

Example

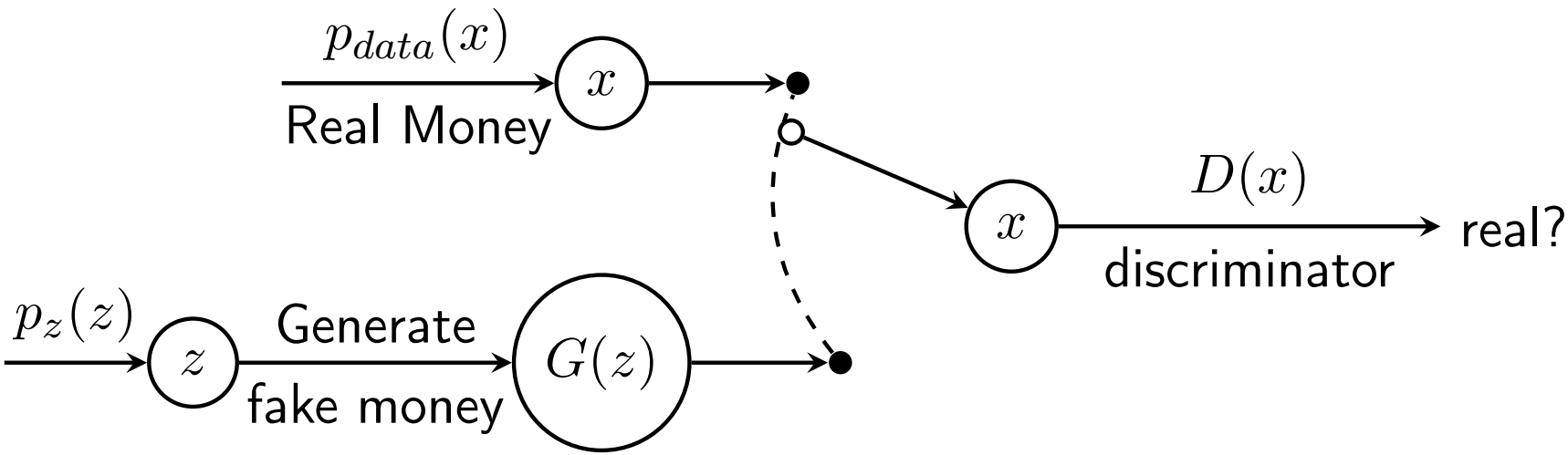


Figure: A visual representation of Generative Adversarial Network

- Canonical Example
  - Think of G as a fraud trying to create fake currency and use it without detection.
  - Think of D is working for the police who is trying to detect counterfeit money.

Generative Adversarial Networks

Objective

- Objective for Discriminator D :

Generative Adversarial Networks

Objective

- Objective for Discriminator D :
  - If a sample x came from the training set, then D needs to predict 1. Which is the same as maximizing:

$$\mathbb{E}_{x \sim p_{data}(x)} \log D(x)$$

Generative Adversarial Networks

Objective

- Objective for Discriminator D :
  - If a sample x came from the training set, then D needs to predict 1. Which is the same as maximizing:

$$\mathbb{E}_{x \sim p_{data}(x)} \log D(x)$$

  - If a sample $G(z)$ was generated by G then D needs to predict 0. Which is the same as maximizing:

$$\mathbb{E}_{z \sim p_G(z)} \log(1 - D(G(z)))$$

Generative Adversarial Networks

Objective

- Objective for Discriminator D :
  - If a sample x came from the training set, then D needs to predict 1. Which is the same as maximizing:

$$\mathbb{E}_{x \sim p_{data}(x)} \log D(x)$$

  - If a sample $G(z)$ was generated by G then D needs to predict 0. Which is the same as maximizing:

$$\mathbb{E}_{z \sim p_G(z)} \log(1 - D(G(z)))$$

- Objective for Generator G.

Generative Adversarial Networks

Objective

- Objective for Discriminator D :
  - If a sample x came from the training set, then D needs to predict 1. Which is the same as maximizing:

$$\mathbb{E}_{x \sim p_{data}(x)} \log D(x)$$

  - If a sample $G(z)$ was generated by G then D needs to predict 0. Which is the same as maximizing:

$$\mathbb{E}_{z \sim p_G(z)} \log(1 - D(G(z)))$$

- Objective for Generator G.
  - The goal for G is to make D predict 1 on its generated samples $G(z)$. Which is the same as minimizing:

$$\mathbb{E}_{z \sim p_G(z)} \log(1 - D(G(z)))$$

Generative Adversarial Networks

Objective

- The total objective for a GAN is:

$$V(G, D) = \min_G \max_D \mathbb{E}_x \log D(x) + \mathbb{E}_z \log(1 - D(G(z)))$$

Generative Adversarial Networks

Objective

- The total objective for a GAN is:

$$V(G, D) = \min_G \max_D \mathbb{E}_x \log D(x) + \mathbb{E}_z \log(1 - D(G(z)))$$

- This is a two-player minimax game.

Generative Adversarial Networks

Objective

- The total objective for a GAN is:

$$V(G, D) = \min_G \max_D \mathbb{E}_x \log D(x) + \mathbb{E}_z \log(1 - D(G(z)))$$

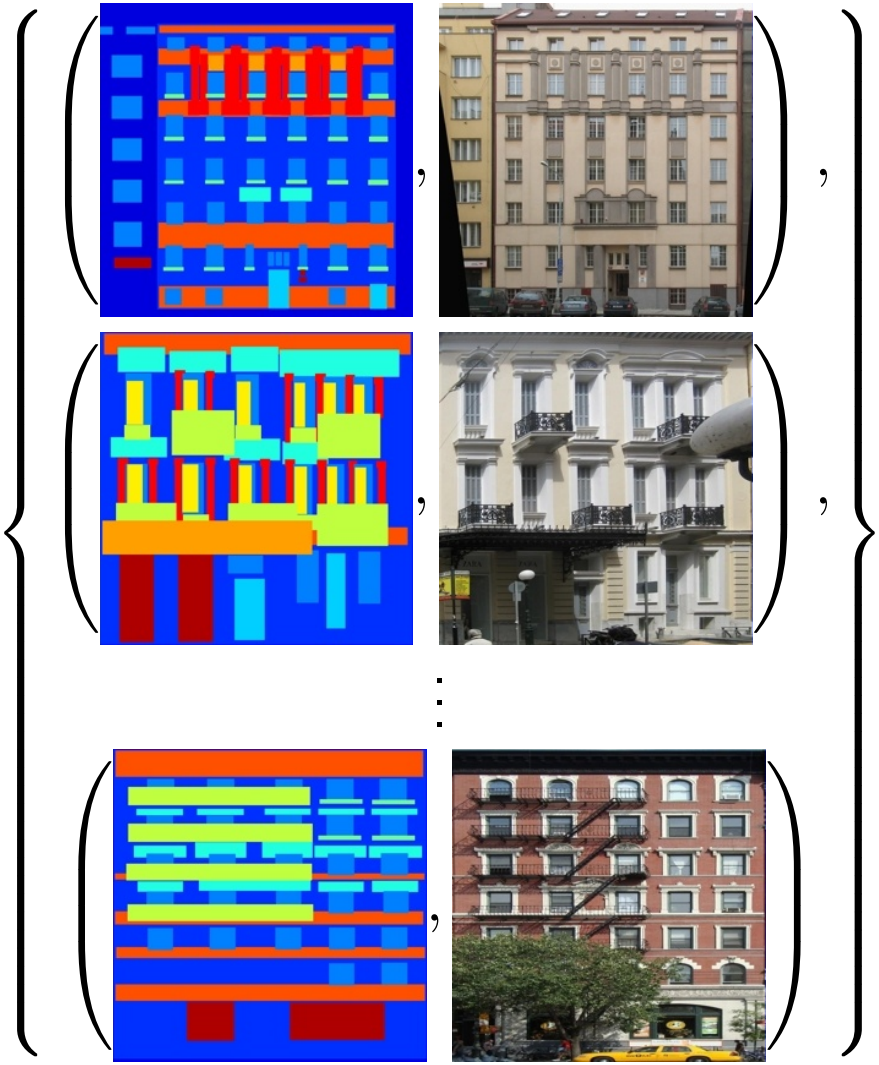
- This is a two-player minimax game.
- There exists a straightforward extension to GANs, called conditional GAN (Mirza & Osindero, 2014), where we condition both G and D on some c:

$$V(G, D) = \min_G \max_D \mathbb{E}_{c,x} \log D(c, x) + \mathbb{E}_{c,z} \log(1 - D(c, G(c, z)))$$

Supervised Image-to-Image Translation

Supervised Image-to-Image Translation

In supervised Image-to-Image translation there exist "paired" examples of input-ouput images for the problem we are solving.



Can we use a simple CNN?

- There exist several generative CNN models — for example Encoder-Decoder, and U-Net — so why don't we use them?

Can we use a simple CNN?

- There exist several generative CNN models — for example Encoder-Decoder, and U-Net — so why don't we use them?
- CNNs learn to minimize a loss function and although the learning process is automatic, we still need to carefully design the loss function. That is, we need to tell the CNN what we want to minimize.

Can we use a simple CNN?

- There exist several generative CNN models — for example Encoder-Decoder, and U-Net — so why don't we use them?
- CNNs learn to minimize a loss function and although the learning process is automatic, we still need to carefully design the loss function. That is, we need to tell the CNN what we want to minimize.
- But if we take a naive approach and tell the CNN to minimize the Euclidean distance between the input image and output image, then we will get blurry results.

Can we use a simple CNN?

- There exist several generative CNN models — for example Encoder-Decoder, and U-Net — so why don't we use them?
- CNNs learn to minimize a loss function and although the learning process is automatic, we still need to carefully design the loss function. That is, we need to tell the CNN what we want to minimize.
- But if we take a naive approach and tell the CNN to minimize the Euclidean distance between the input image and output image, then we will get blurry results.
- This happens because it is much "safer" for the L2 loss to predict the mean of the distribution, because this minimizes the mean pixel-wise error, but results in a blurry averaged image. (Zhang et al., 2016)

Pix2Pix

- Rather than specifying a carefully crafted model and loss for each Image-to-Image problem. It would be better if we could specify a goal like: *"make the output indistinguishable from reality"*.

Pix2Pix

- Rather than specifying a carefully crafted model and loss for each Image-to-Image problem. It would be better if we could specify a goal like: *"make the output indistinguishable from reality"*.
- While this may seem like a vague goal, this is exactly what Generative Adversarial Models claim to do!

Pix2Pix

- Rather than specifying a carefully crafted model and loss for each Image-to-Image problem. It would be better if we could specify a goal like: *"make the output indistinguishable from reality"*.
- While this may seem like a vague goal, this is exactly what Generative Adversarial Models claim to do!
- **Pix2Pix** was one of the first general-purpose Image-to-Image translation network which was introduced in the seminal paper *"Image-to-Image Translation with Conditional Adversarial Networks"* (Isola et al., 2017).

Pix2Pix

Objective

- The goal is to generate a target image based on some source image and we don't want the GAN to generate random images, so we use a conditional GAN.

Pix2Pix

Objective

- The goal is to generate a target image based on some source image and we don't want the GAN to generate random images, so we use a conditional GAN.
- In their experiments they found using both the adversarial loss and L1 loss generated the best looking results.
- The loss function for D remains unchanged but G now not only has to fool the D but also minimize the L1 loss between the predicted and target

Objective

- Adversarial loss for cGAN:

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

- L1 loss for the generator G:

$$\mathcal{L}_{\text{L1}}(G) = \mathbb{E}_{y,c,z} \|y - G(c, z)\|$$

- Final objective is:

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

Discriminator Architecture

PatchGAN

- Since we are already adding a traditional loss such as L1(or L2) which — although produce blurry results when used alone — are very useful for enforcing correctness at the low frequency.
- This is the main idea behind the **PatchGAN** discriminator.

Discriminator Architecture

PatchGAN

- Since we are already adding a traditional loss such as L1(or L2) which — although produce blurry results when used alone — are very useful for enforcing correctness at the low frequency.
- This is the main idea behind the **PatchGAN** discriminator.
- **PatchGAN** architecture restricts the discriminator to model only high frequency structure in the generated image.
- To model this, it is sufficient to restrict the attention to local image patches.

Discriminator Architecture

PatchGAN

- Since we are already adding a traditional loss such as L1(or L2) which — although produce blurry results when used alone — are very useful for enforcing correctness at the low frequency.
- This is the main idea behind the **PatchGAN** discriminator.
- **PatchGAN** architecture restricts the discriminator to model only high frequency structure in the generated image.
- To model this, it is sufficient to restrict the attention to local image patches.
- Rather than generating a single probability, D generates a $k \times k$ matrix of values in range $[0, 1]$ where each element in the matrix corresponds to probability of a local "patch" of the image.

Discriminator Architecture

PatchGAN

- Since we are already adding a traditional loss such as L1(or L2) which — although produce blurry results when used alone — are very useful for enforcing correctness at the low frequency.
- This is the main idea behind the **PatchGAN** discriminator.
- **PatchGAN** architecture restricts the discriminator to model only high frequency structure in the generated image.
- To model this, it is sufficient to restrict the attention to local image patches.
- Rather than generating a single probability, D generates a $k \times k$ matrix of values in range $[0, 1]$ where each element in the matrix corresponds to probability of a local "patch" of the image.
- One advantage of **PatchGAN** is that the same architecture can extend to images of higher resolution with same number of parameters, and it would generate a similar, but larger, $k' \times k'$ matrix.

Discriminator Architecture

PatchGAN

Inspired by **DCGAN** (Radford, Metz, & Chintala, 2015), the discriminator is implemented as a Deep CNN mostly build up of repetitions of modules of form: Convolution - BatchNorm - ReLU, where we down-sample until we reach the required $k \times k$ matrix.

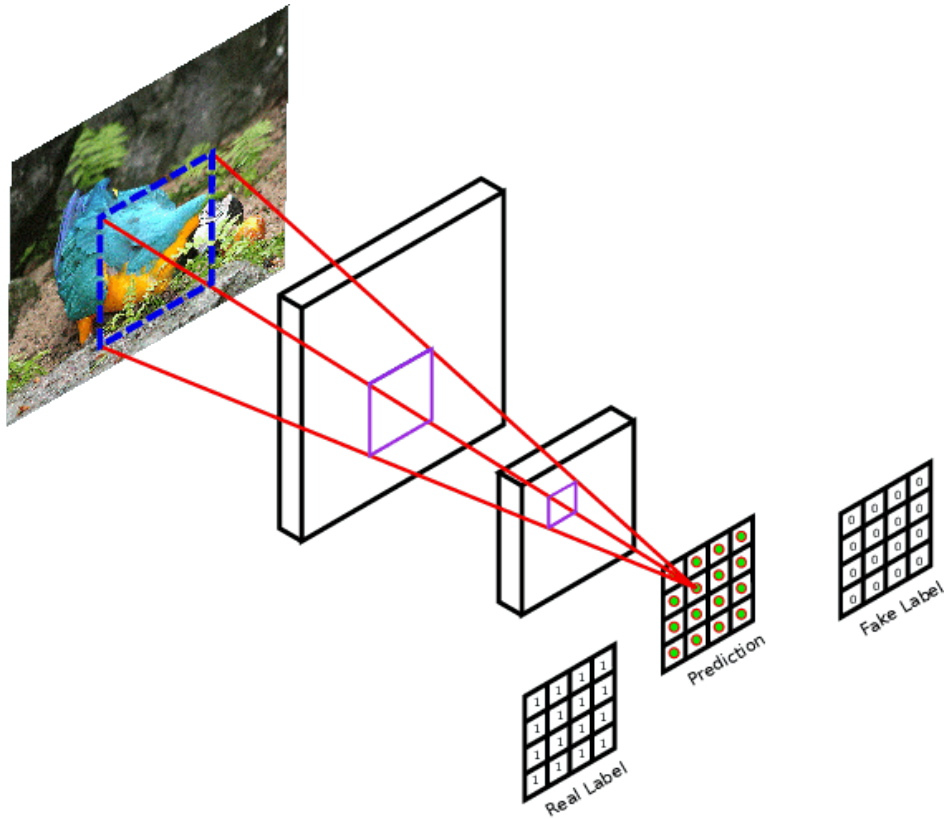


Figure: Visualization of PatchGAN

Generator Architecture

Encoder-Decoder

- One of the obvious choices for a generator architecture is the encoder-decoder network.
- In this network, the input image is passed through a series of layers, called **encoder**, which decreases the resolution but increases the number of channels until we reach an image of size 1×1, called the "bottleneck".
- After that, the process is reversed until we reach the resolution of the original image. This reverse network is called a **decoder**.

Generator Architecture

Encoder-Decoder

- One of the obvious choices for a generator architecture is the encoder-decoder network.
- In this network, the input image is passed through a series of layers, called **encoder**, which decreases the resolution but increases the number of channels until we reach an image of size 1×1, called the "bottleneck".
- After that, the process is reversed until we reach the resolution of the original image. This reverse network is called a **decoder**.

Note: We don't pass an extra noise z vector to G, so noise is introduced in the model via dropout.

Generator Architecture

Encoder-Decoder

Here also we use modules of the form: Convolution - BatchNorm - LeakyReLU for encoder, for decoder we use: TransposeConvolution - BatchNorm - ReLU.

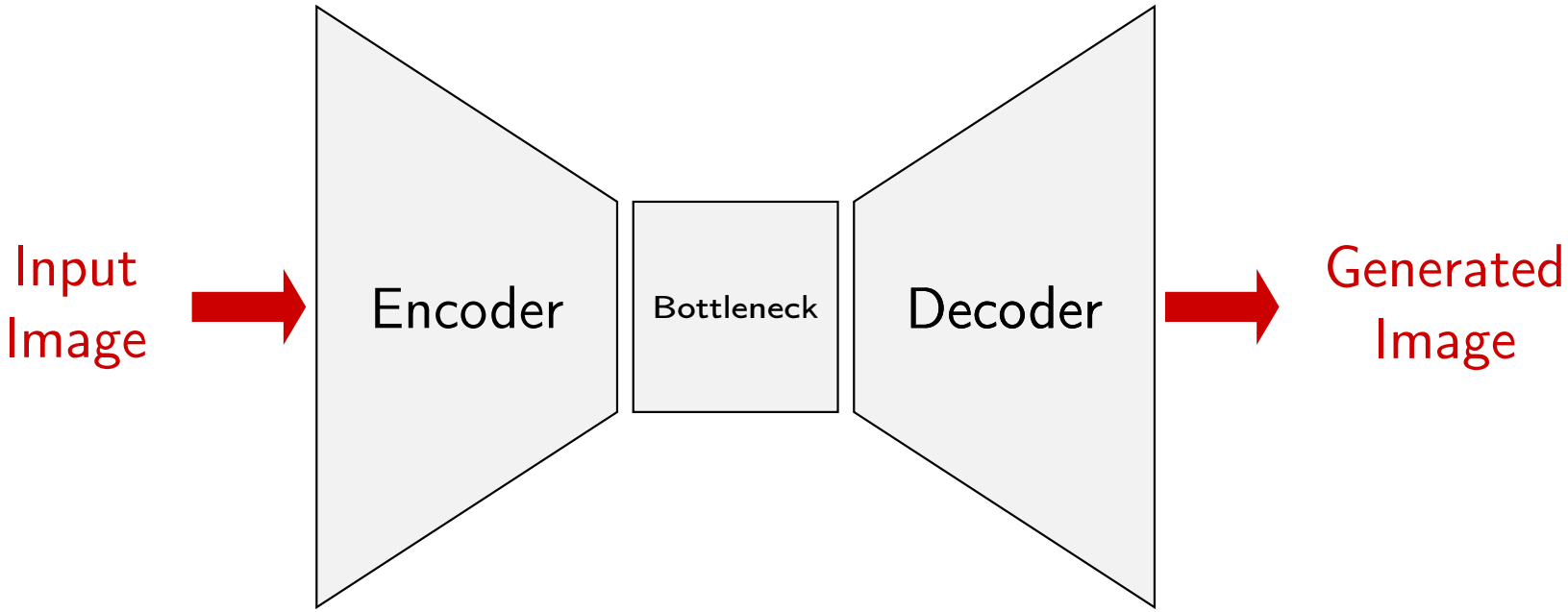


Figure: Encoder-Decoder Architecture

Generator Architecture

U-Net

- For many image-translation problems, there is a lot of low-level information shared between the input and output image.
- For example, in the case of colorization, the input and output share the location of edges.
- To preserve this information, we add skip connections to create a "U-net" like architecture.

Generator Architecture

U-Net

Here we just add skip-connections between layer i in the encoder and layer $n - i$ in the decoder.

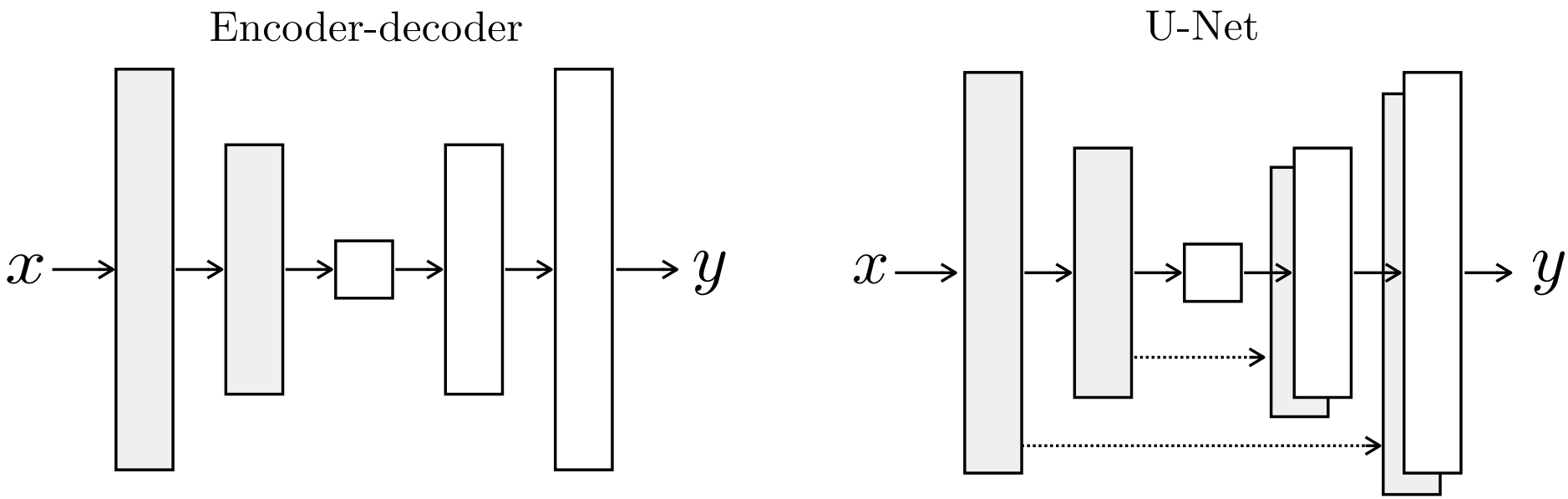


Figure: Difference between Encoder-Decoder and U-Net architectures.

Datasets and Training Details

- Datasets:
  - **Satellite2Map**: 1096 training images from pix2pix datasets (Isola et al., 2017) trained for 150 epochs with batch size 1.
  - **Edges2Shoes**: 50k training images from pix2pix datasets (Isola et al., 2017) trained for 30 epochs with batch size 1.
  - **Anime Sketch Colorization**: 14k training images and 4k validation images from Kaggle (link) trained for 75 epochs with batch size 1.
- Training Details:
  - **Optimizer**: Adam with $\beta_1 = 0.5$, $\beta_2 = 0.999$
  - **Learning Rate**: 0.0002
  - $\boldsymbol{\lambda}$: 100

Note: Instead of the architectures mentioned in this, we use the improved generator and discriminator mentioned in the next section.

Outputs I

Satellite2Map

Figure: Input Image

Figure: Output Image

Figure: Predicted Image

Outputs II

Satellite2Map

Figure: Input Image

Figure: Output Image

Figure: Predicted Image

Outputs III

Satellite2Map

Figure: Input Image

Figure: Output Image

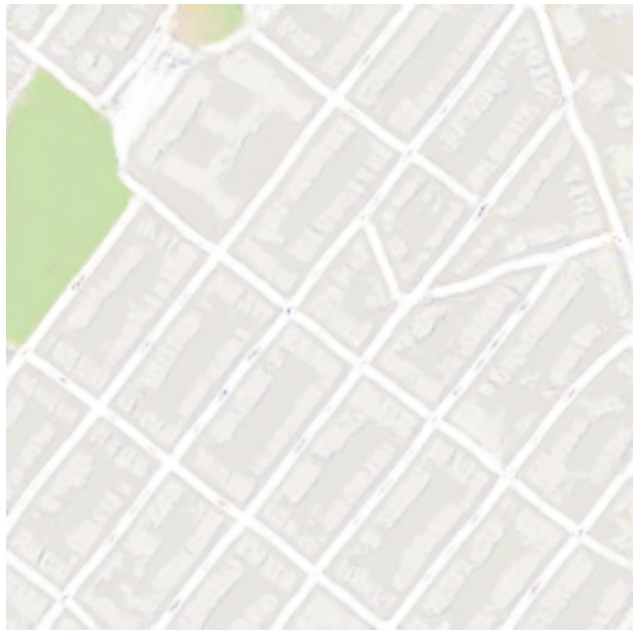


Figure: Predicted Image

Outputs IV

Satellite2Map

Figure: Input Image

Figure: Output Image

Figure: Predicted Image

Outputs V

Satellite2Map

Figure: Input Image

Figure: Output Image

Figure: Predicted Image

Outputs: Bad Cases I

Satellite2Map

Bad Cases:

Figure: Input Image

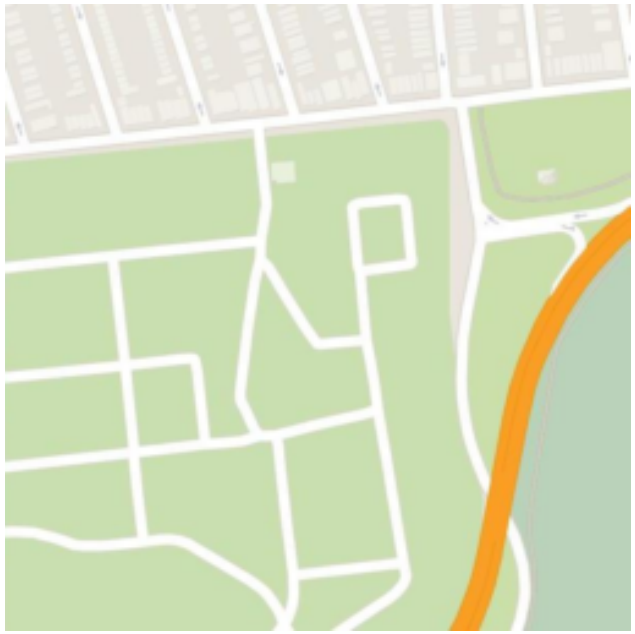


Figure: Output Image

Figure: Predicted Image

Outputs: Bad Cases II

Satellite2Map

Figure: Input Image

Figure: Output Image

Figure: Predicted Image

Outputs I

Edges2Shoes

Figure: Input Image

Figure: Output Image

Figure: Predicted Image

Outputs II

Edges2Shoes

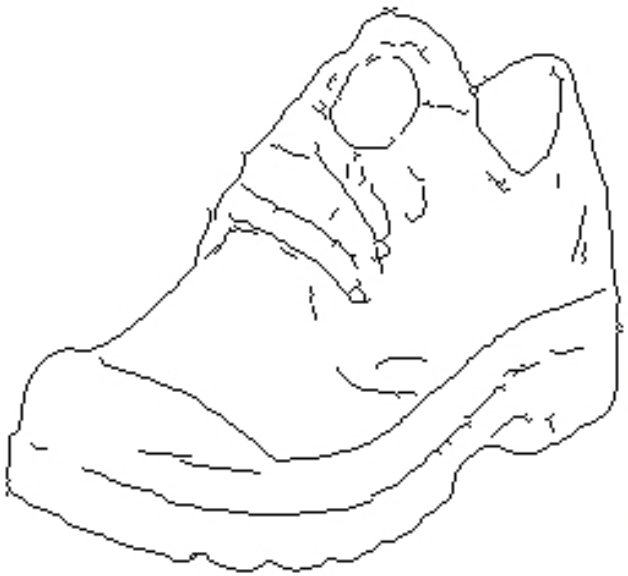


Figure: Input Image

Figure: Output Image

Figure: Predicted Image

Outputs III

Edges2Shoes

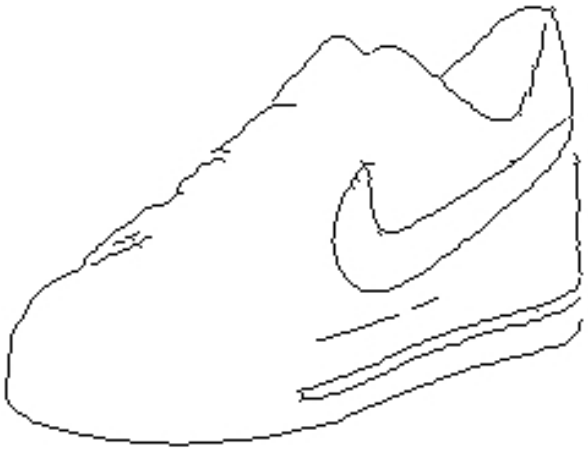


Figure: Input Image

Figure: Output Image

Figure: Predicted Image

Outputs IV

Edges2Shoes

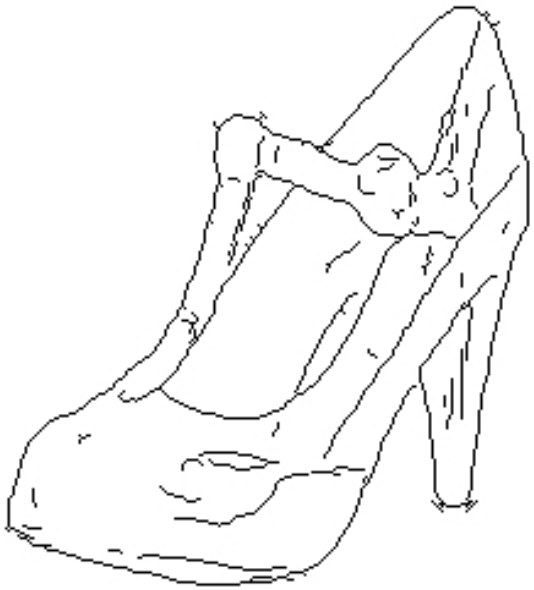


Figure: Input Image

Figure: Output Image

Figure: Predicted Image

Outputs V

Edges2Shoes

Figure: Input Image

Figure: Output Image

Figure: Predicted Image

Outputs I

Anime Sketch Colorization

Figure: Input Image

Figure: Output Image

Figure: Predicted Image

Outputs II

Anime Sketch Colorization

Figure: Input Image

Figure: Output Image

Figure: Predicted Image

Outputs III

Anime Sketch Colorization

Figure: Input Image

Figure: Output Image

Figure: Predicted Image

Outputs IV

Anime Sketch Colorization

Figure: Input Image

Figure: Output Image

Figure: Predicted Image

Outputs: Bad Cases

Anime Sketch Colorization

Figure: Input Image

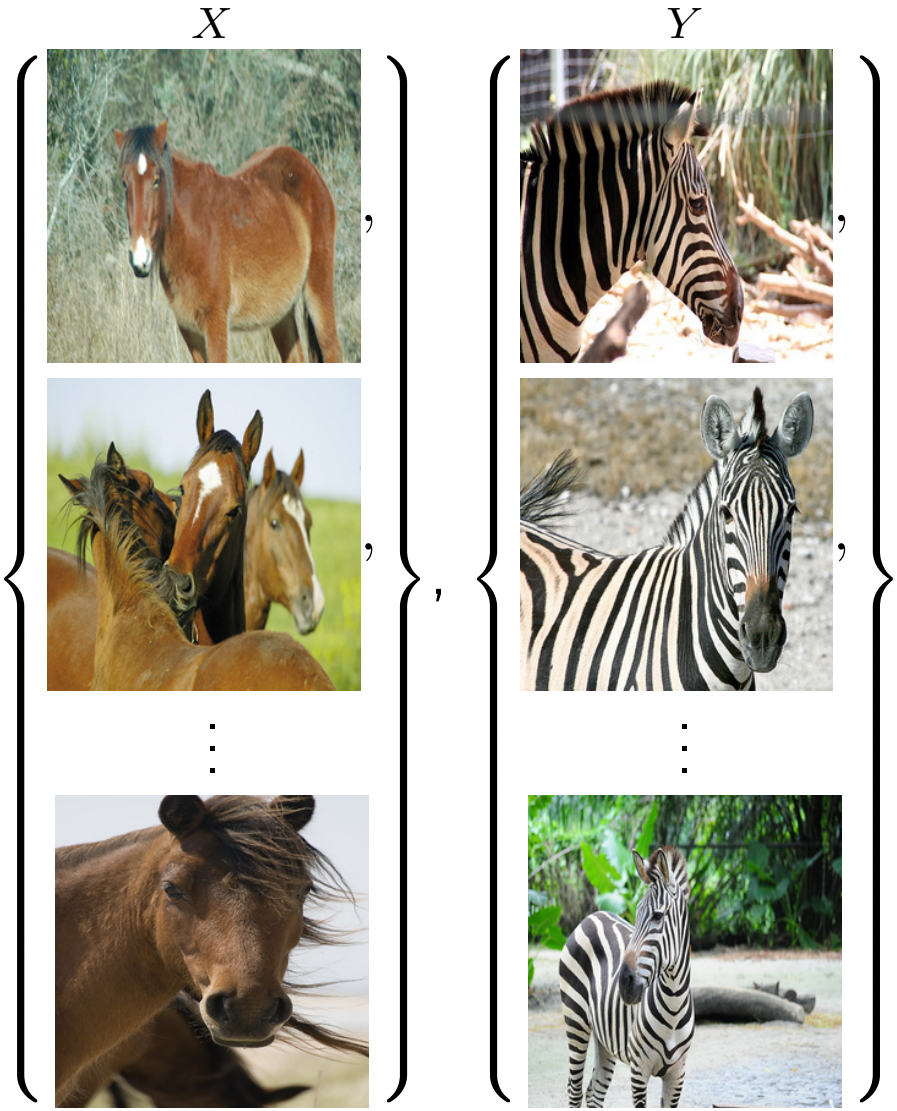
Figure: Output Image

Figure: Predicted Image

Unsupervised Image-to-Image Translation

Unsupervised Image-to-Image Translation

In unsupervised Image-to-Image translation there does not exist "paired" examples of input-ouput images, but only a "domain" (set) of images X and Y.



CycleGAN

- In the previous section we saw how we can utilize cGANs to tackle the supervised problem.

CycleGAN

- In the previous section we saw how we can utilize cGANs to tackle the supervised problem.
- Surprisingly, there is a way to extend the previous approach to work in an unsupervised setting!

CycleGAN

- In the previous section we saw how we can utilize cGANs to tackle the supervised problem.
- Surprisingly, there is a way to extend the previous approach to work in an unsupervised setting!
- **CycleGAN** (Zhu et al., 2017) is a model for unsupervised image-translation problem.
- It is build using two GANs, capturing both transformations between domains as $G : X \to Y$ and also $F : Y \to X$.

CycleGAN

- In the previous section we saw how we can utilize cGANs to tackle the supervised problem.
- Surprisingly, there is a way to extend the previous approach to work in an unsupervised setting!
- **CycleGAN** (Zhu et al., 2017) is a model for unsupervised image-translation problem.
- It is build using two GANs, capturing both transformations between domains as $G : X \to Y$ and also $F : Y \to X$.
- One of the main ideas behind this model is to add a cycle-consistency loss.
- That is, if $K \in X$ and $K' \in Y$, then if we apply G and F sequentially on K and vice-versa on K', then we should get the same image back. That is:

$$K \approx F(G(K)) \text{ and } K' \approx G(F(K'))$$

CycleGAN

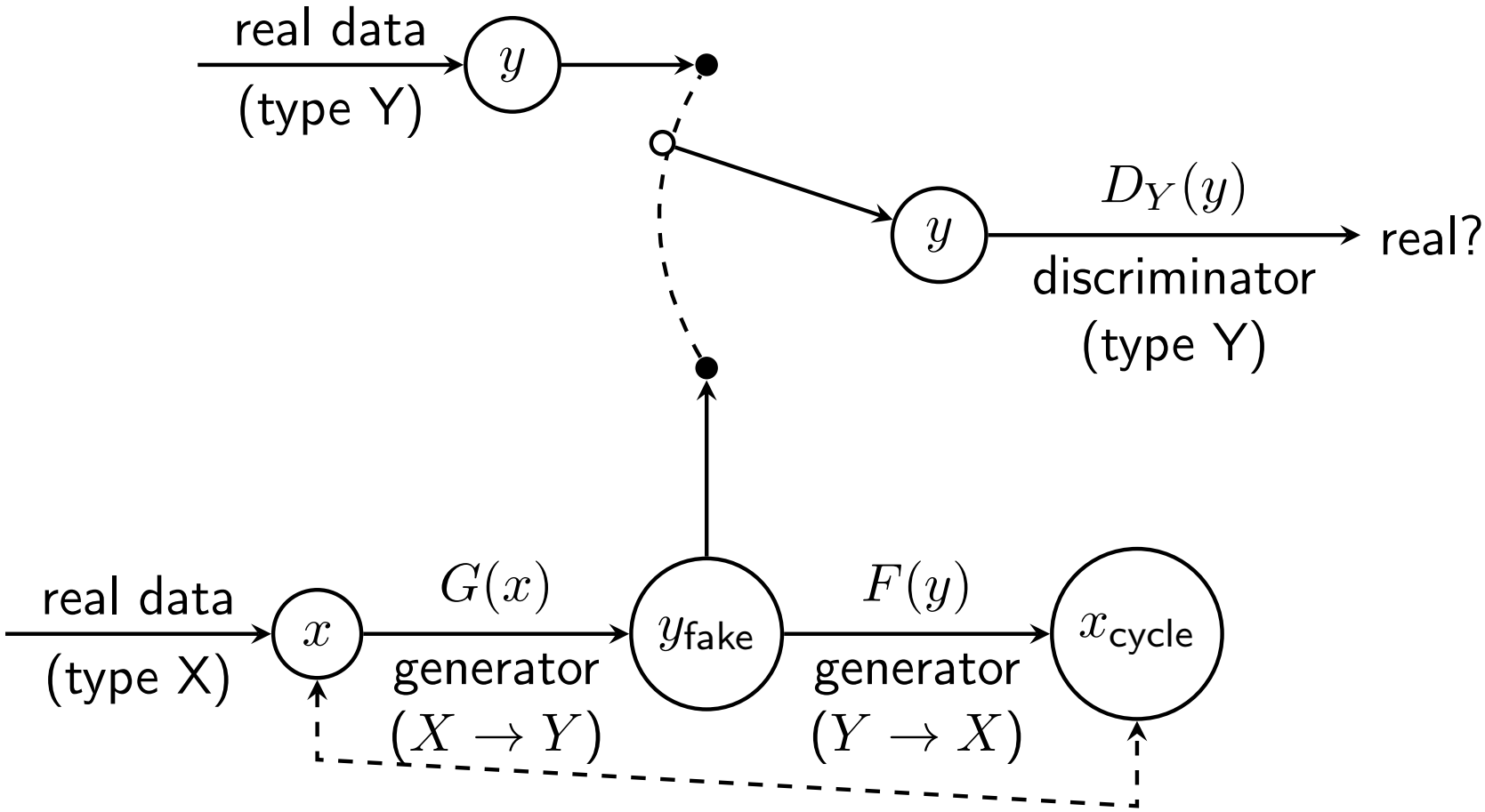


Figure: CycleGAN Visual Representation

CycleGAN

Objective

- Adversarial loss:

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

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

- Cycle Consistency loss:

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

- Full Objective:

$$\begin{aligned}G^*, F^* &= \arg \min_{G,F} \max_{D_X, D_Y} \mathcal{L}(G, F, D_X, D_Y) \\ &= \arg \min_{G,F} \max_{D_X, D_Y} \mathcal{L}_{\text{GAN}}(G, D_Y) + \mathcal{L}_{\text{GAN}}(F, D_X) + \lambda \mathcal{L}_{\text{cyc}}(G, F)\end{aligned}$$

Discriminator Architecture

PatchGAN

- The discriminator architecture is mostly the same as in the **Pix2Pix** model, but there are a few changes.
- We replace BatchNorm with InstanceNorm (Ulyanov, Vedaldi, & Lempitsky, 2016)
- We replace the negative log-likelihood objective by a least-squares loss, as this loss is more stable during training and generate higher quality images (Mao et al., 2017).
  - That is, we train G to minimize:

$$\mathbb{E}_{x \sim p_{\textbf{data}}(x)}[(D_Y(G(x)) - 1)^2]$$

  - And we train D_Y to minimize:

$$\mathbb{E}_{y \sim p_{\textbf{data}}(y)}[(D_Y(y) - 1)^2] + \mathbb{E}_{x \sim p_{\textbf{data}}(x)}[D_Y(G(x))^2]$$

Generator Architecture

- The generator has a completely new architecture inspired networks presented in (Johnson, Alahi, & Fei-Fei, 2016).
- Similarly to the previous generator, we build the model in modules of Convolution - InstanceNorm - ReLU.

Generator Architecture

- The generator has a completely new architecture inspired networks presented in (Johnson et al., 2016).
- Similarly to the previous generator, we build the model in modules of Convolution - InstanceNorm - ReLU.
- The network has:
  - 3 convolution layers.

Generator Architecture

- The generator has a completely new architecture inspired networks presented in (Johnson et al., 2016).
- Similarly to the previous generator, we build the model in modules of Convolution - InstanceNorm - ReLU.
- The network has:
  - 3 convolution layers.
  - several residual blocks,

Generator Architecture

- The generator has a completely new architecture inspired networks presented in (Johnson et al., 2016).
- Similarly to the previous generator, we build the model in modules of Convolution - InstanceNorm - ReLU.
- The network has:
  - 3 convolution layers.
  - several residual blocks,
  - 2 fractionally-strided convolutions to sample up

Generator Architecture

- The generator has a completely new architecture inspired networks presented in (Johnson et al., 2016).
- Similarly to the previous generator, we build the model in modules of Convolution - InstanceNorm - ReLU.
- The network has:
  - 3 convolution layers.
  - several residual blocks,
  - 2 fractionally-strided convolutions to sample up
  - and finally a convolution to map to RGB.

Datasets and Training Details

- Datasets:
  - **Horse2Zebra**: One of the CycleGAN datasets, trained for 10 epochs with batch size 1.
- Training Details:
  - **Optimizer**: Adam with $\beta_1 = 0.5$, $\beta_2 = 0.999$
  - **Learning Rate**: 0.0002
  - $\boldsymbol{\lambda}$: 10

Outputs I

Horse2Zebra

Figure: Input Image

Figure: Predicted Image

Outputs II

Horse2Zebra

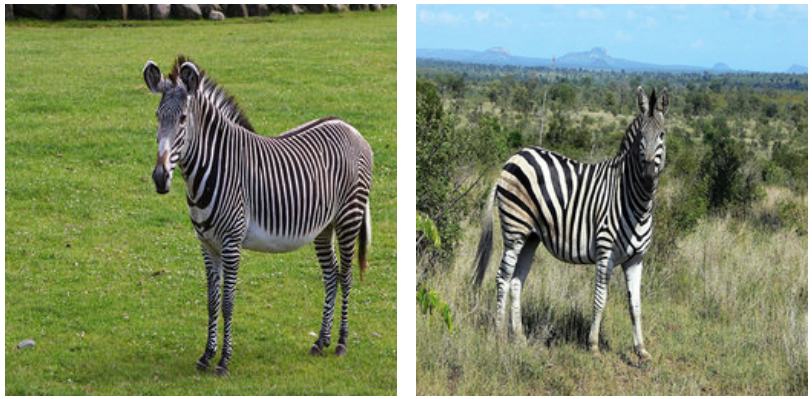


Figure: Input Image

Figure: Predicted Image

Outputs: Bad Cases I

Figure: Input Image

Figure: Predicted Image

Outputs: Bad Cases II

Figure: Input Image

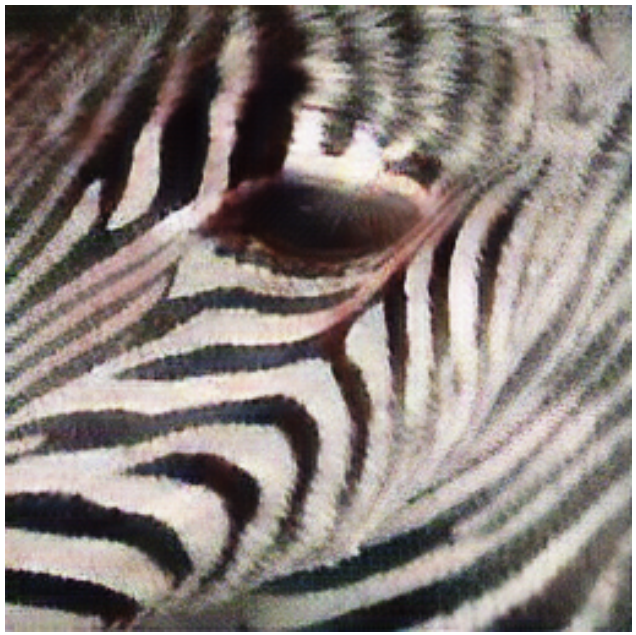


Figure: Predicted Image

Conclusion

Conclusion

- Conditional GANs for general purpose Image-to-Image translation offer powerful and generalizable solution to the problem.
- However, as we saw, they are not without their limitations.
- Source code: www.github.com/16bitmood/ImageToImage

References I

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*, *27*.

Isola, P., Zhu, J.-Y., Zhou, T., & Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 1125–1134).

Johnson, J., Alahi, A., & Fei-Fei, L. (2016). Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision* (pp. 694–711).

Mao, X., Li, Q., Xie, H., Lau, R. Y., Wang, Z., & Paul Smolley, S. (2017). Least squares generative adversarial networks. In *Proceedings of the ieee international conference on computer vision* (pp. 2794–2802).

References II

Mirza, M., & Osindero, S. (2014). Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

Ulyanov, D., Vedaldi, A., & Lempitsky, V. (2016). Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*.

Zhang, R., Isola, P., & Efros, A. A. (2016). Colorful image colorization. In *European conference on computer vision* (pp. 649–666).

Zhu, J.-Y., Park, T., Isola, P., & Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the ieee international conference on computer vision* (pp. 2223–2232).

Thank you!