Generative Adversarial Learning

Badri N Subudhi

GAN

- Generative Adversarial Networks (GANs) were developed in 2014 by Ian Goodfellow and his teammates.
- GAN is basically an approach to generative modeling that generates a new set of data based on training data that look like training data.
- GANs have two main blocks(two neural networks) which compete with each other and are able to capture, copy, and analyze the variations in a dataset.
- The two models are usually called Generator and Discriminator which we will cover in Components on GANs.

Generator and Discriminator

- The generator network takes random input (typically noise) and generates samples resembling the training data, such as images, text, or audio.
- Its goal is to produce samples indistinguishable from real data.
- The discriminator network, on the other hand, distinguishes between real and generated samples, classifying real data as real and generated data as fake.

General Terms

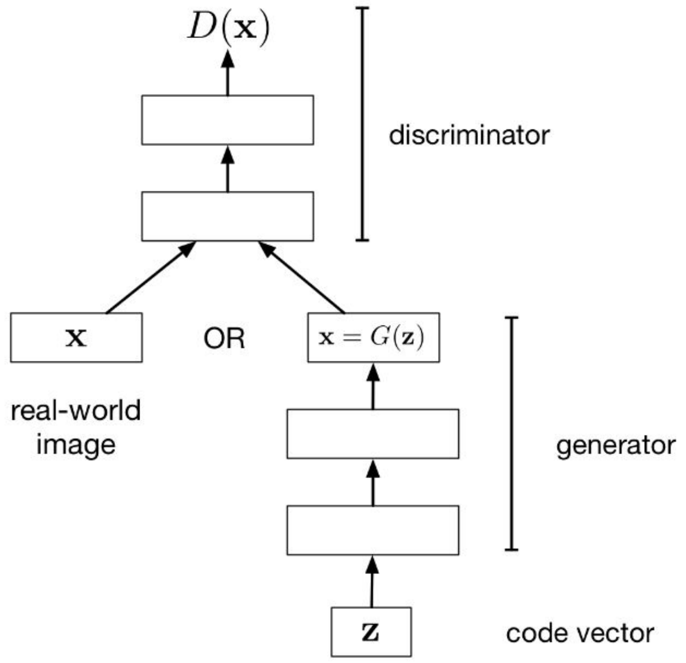
- Generative – To learn a generative model, which describes how data is generated in terms of a probabilistic model. In simple words, it explains how data is generated visually.
- Adversarial – The training of the model is done in an adversarial setting.
- Networks – use deep neural networks for training purposes.

- The training process involves an adversarial game where the generator tries to fool the discriminator, and the discriminator aims to improve its ability to distinguish real from generated data.
- Over time, the generator becomes better at creating realistic samples, and the discriminator becomes more skilled at identifying them. This process ideally leads to the generator producing high-quality samples that are hard to distinguish from real data.
- GAN Techniques have shown impressive results in various domains like image synthesis, text generation, and video generation, enhancing the field of generative modeling and enabling new creative applications in artificial intelligence.

- Generator Network: Takes random input to generate samples resembling training data.
- Discriminator Network: Distinguishes between real and generated samples.
- Adversarial Training: Generator tries to fool the discriminator, and the discriminator improves its distinguishing skills.
- Progression: Generator produces more realistic samples; discriminator becomes better at identifying them.
- Applications: Image synthesis, text generation, video generation, creating deepfakes, enhancing low-resolution images.

GANs

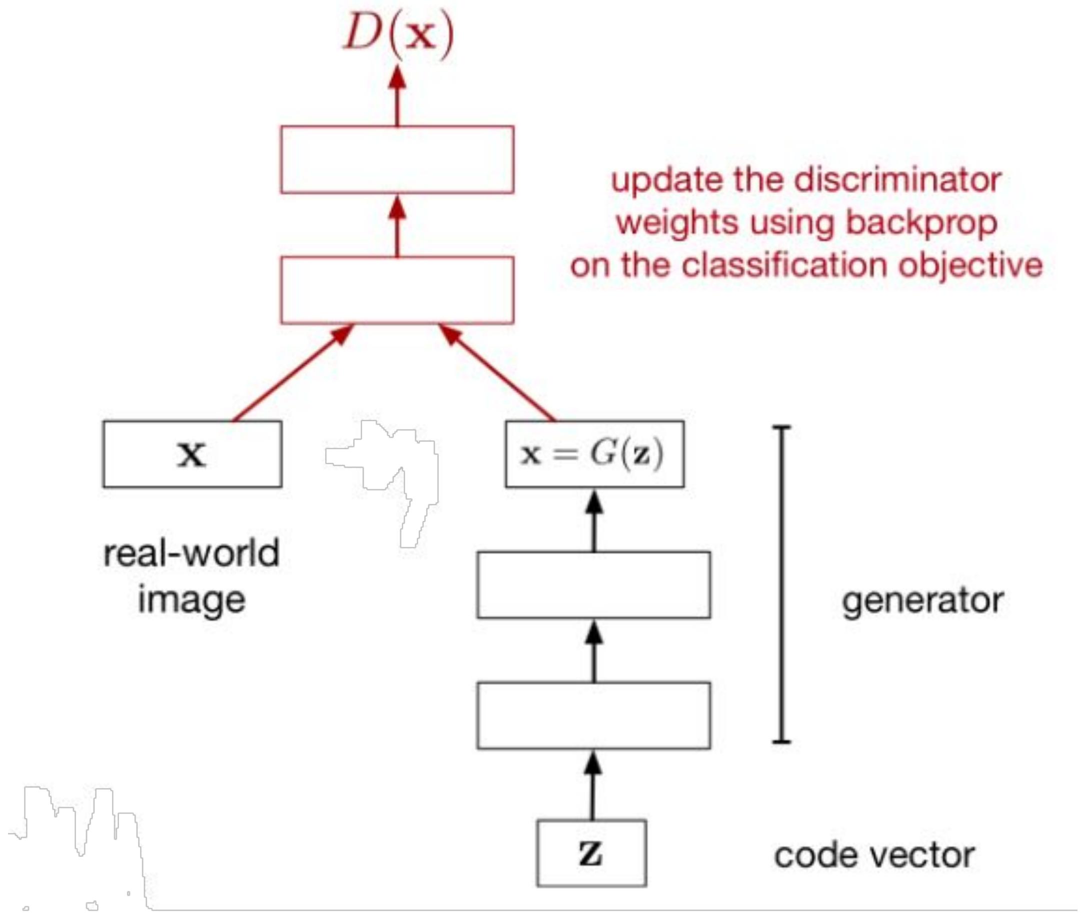
- Z is some random noise sampled from gaussian or uniform distribution
- Z can also be thought as the latent state representation of image



Discriminator

- The discriminator network's purpose is to distinguish between real data and data generated by the generator.
- It receives both real data from the training set and fake data from the generator as input.
- The output is a probability indicating whether the input data is real or fake.
- The primary objective of the discriminator is to correctly identify real versus generated data.

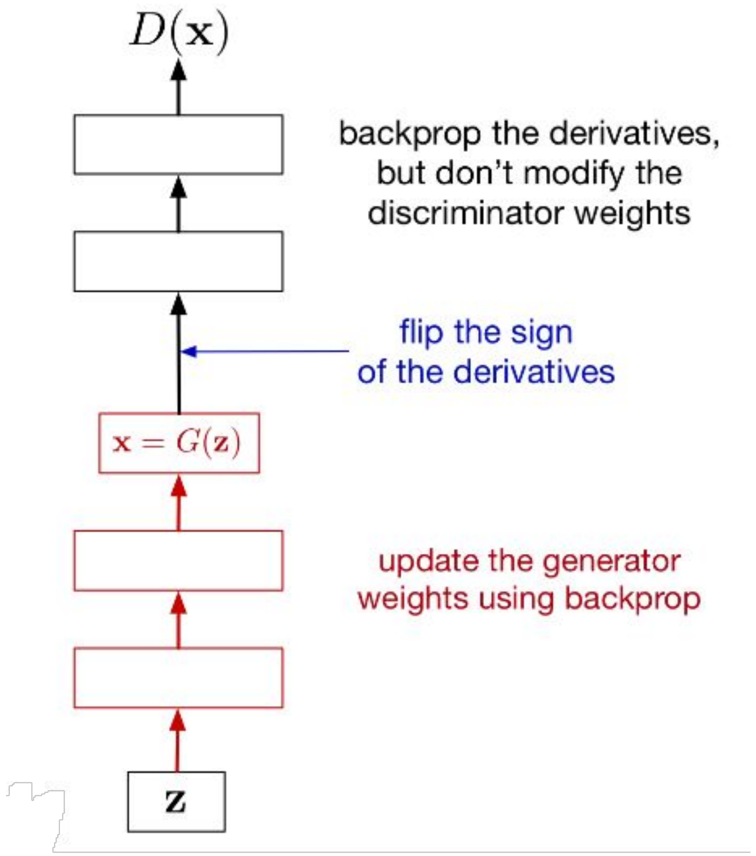
Training the discriminator


$D(\mathbf{x})$
update the discriminator weights using backprop on the classification objective
$\mathbf{x}$
real-world image
$\mathbf{x} = G(\mathbf{z})$
generator
$\mathbf{z}$
code vector

Generator

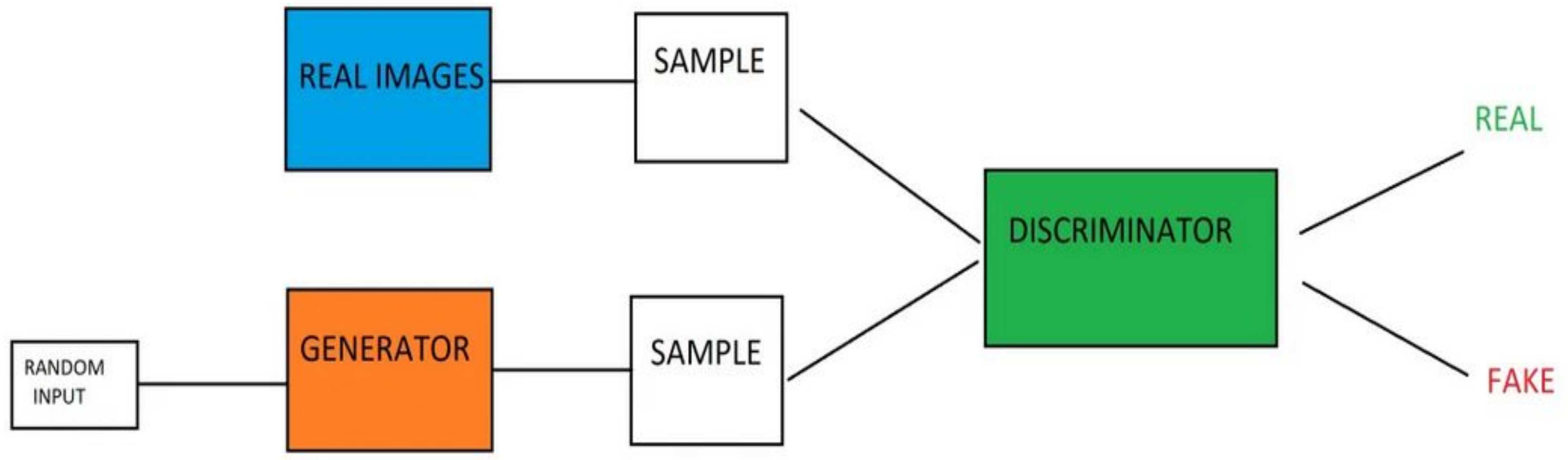
- The generator network takes random input (typically noise) and generates samples resembling the training data, such as images, text, or audio. Its goal is to produce samples indistinguishable from real data.

Training the generator


$D(\mathbf{x})$
backprop the derivatives, but don't modify the discriminator weights
flip the sign of the derivatives
$\mathbf{x} = G(\mathbf{z})$
update the generator weights using backprop
$\mathbf{z}$

Adversarial Game

- Generator: learns to generate better images to fool the discriminator
- Discriminator: learns to become better as distinguishing real from generated images



Why Are GANs Are Called Generative Models?

- The generative part comes from the fact that the model "generates“ new data
- Usually, generative models use an approximation to compute the usually intractable distribution; here, the discriminator part does that Approximation

$$\mathcal{J}_D = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}[-\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}}[-\log(1 - D(G(\mathbf{z})))]$$

$$\arg\max_{G}\min_{D}\mathcal{J}_D.$$

Learning

$$\min_{G} \max_{D} L(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\tan}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{x}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

Discriminator gradient

$$\nabla_{\mathbf{W}_D} \frac{1}{n} \sum_{i=1}^{n} \left[\log \overbrace{D\left(\boldsymbol{x}^{(i)}\right)}^{\substack{\text{predict well on real images} \\ \text{=> probability close to 1}}} + \log \left(1 - \overbrace{D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)}^{\substack{\text{predict well on fake images} \\ \text{=> probability close to 0}}}\right) \right]$$

Learning

$$\min_{G} \max_{D} L(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{tan}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{x}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

Generator gradient

$$\nabla_{\mathbf{w}_G} \frac{1}{n} \sum_{i=1}^{n} \log \left(1 - \overbrace{D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)}^{\substack{\text{predict badly on fake images} \\ \text{=> probability close to 1}}}\right)$$

Convergence

- Converges when Nash-equilibrium (Game Theory concept) is reached in the minmax (zero-sum) game
- Nash-Equilibrium in Game Theory is reached when the actions of one player won't change depending on the opponent's actions
- Here, this means that the GAN produces realistic images and the discriminator outputs random predictions (probabilities close to 0.5)

Advantages of GANs

- Sampling is straight forward
- Training involves maximum likelihood estimation
- Robust to overfitting since generator never sees the training data
- Empirically, GANs are good at capturing the modes of the distribution

Challenges of GANs

- Not easy to compute P(X)
- Training is hard due to non convergence
- Mode-Collapse issue

Problems in Training

- Discriminator is too strong, such that the gradient for the generator vanishes and the generator can't keep up

$$\nabla_{\mathbf{w}_G} \frac{1}{n} \sum_{i=1}^{n} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)$$

- It is modified as

$$\nabla_{\mathbf{w}_G} \frac{1}{n} \sum_{i=1}^{n} \log\left(D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)$$

Convergence: discriminator

- Maximize prediction probability of real as real and fake as fake
- Maximizing log likelihood

Convergence: generator

- Minimize likelihood of the discriminator to make correct predictions (predict fake as fake; real as real), which can be achieved by maximizing the cross-entropy
- Issue of zero gradient
- flip labels and minimize cross entrop

$$\begin{aligned}\mathcal{J}_G &= -\mathcal{J}_D \\ &= \text{const} + \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z})))]\end{aligned}$$

Mode collapse

- Mode collapse refers to a scenario where the generator starts producing a limited variety of outputs, often very similar to each other, instead of a diverse range that represents the real data distribution
- Example: Instead of generating pictures of different animals, but the GAN only produces images of cats, ignoring all other species

Mode Collapse

- During training, generator may find that generating certain samples is able to fool discriminator most of times
- Hence, it starts generating only those samples

Solutions to mode collapse

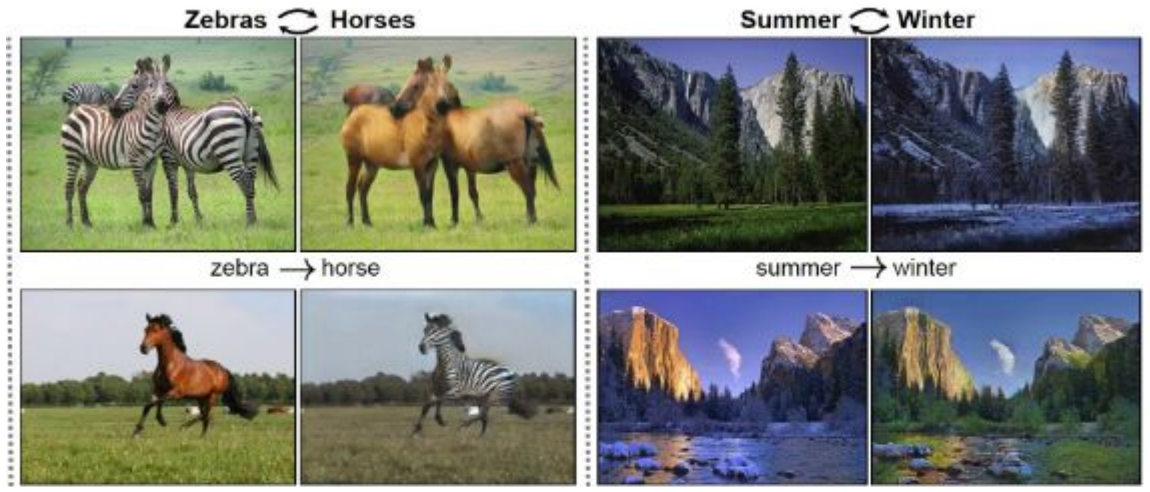
1. **Modified Loss Functions**: Techniques like WGAN (Wasserstein GAN) introduce a new loss function that provides smoother gradients, reducing the chance of mode collapse.
2. **Mini-batch Discrimination**: This involves feeding the discriminator batches of samples and allowing it to use the statistics of the batch to make decisions. This discourages the generator from producing identical samples.
3. **Unrolled GANs**: Here, the generator's update is based on a few steps ahead of the discriminator's update, giving it a broader view of the game and preventing it from getting stuck in mode collapse.

Wasserstein GAN

- Wasserstein distance or “earth moving distance”, which intuitively is the minimum mass displacement to transform one distribution into the other.

Cycle GAN

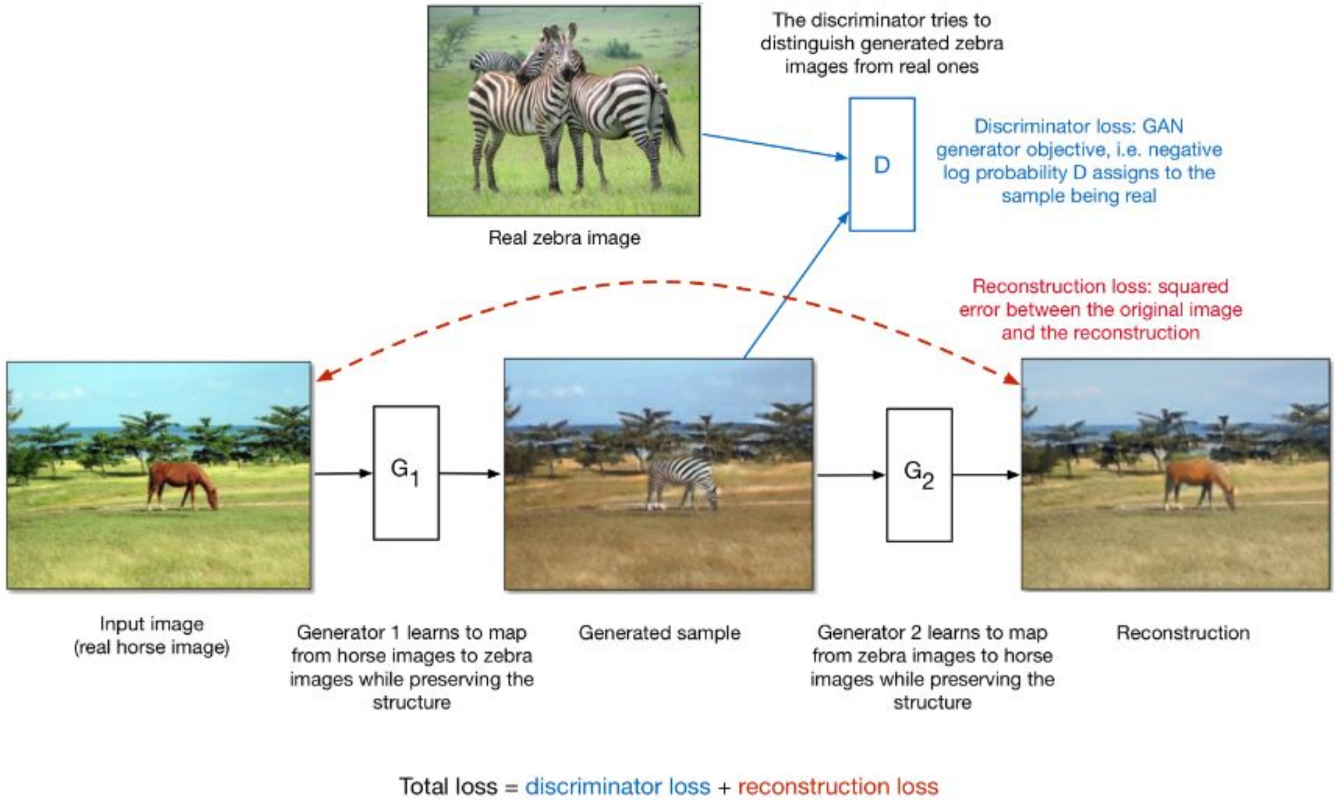
- Style can be transferred using cycle GAN
- Style transfer problem: change the style of an image while preserving the content.



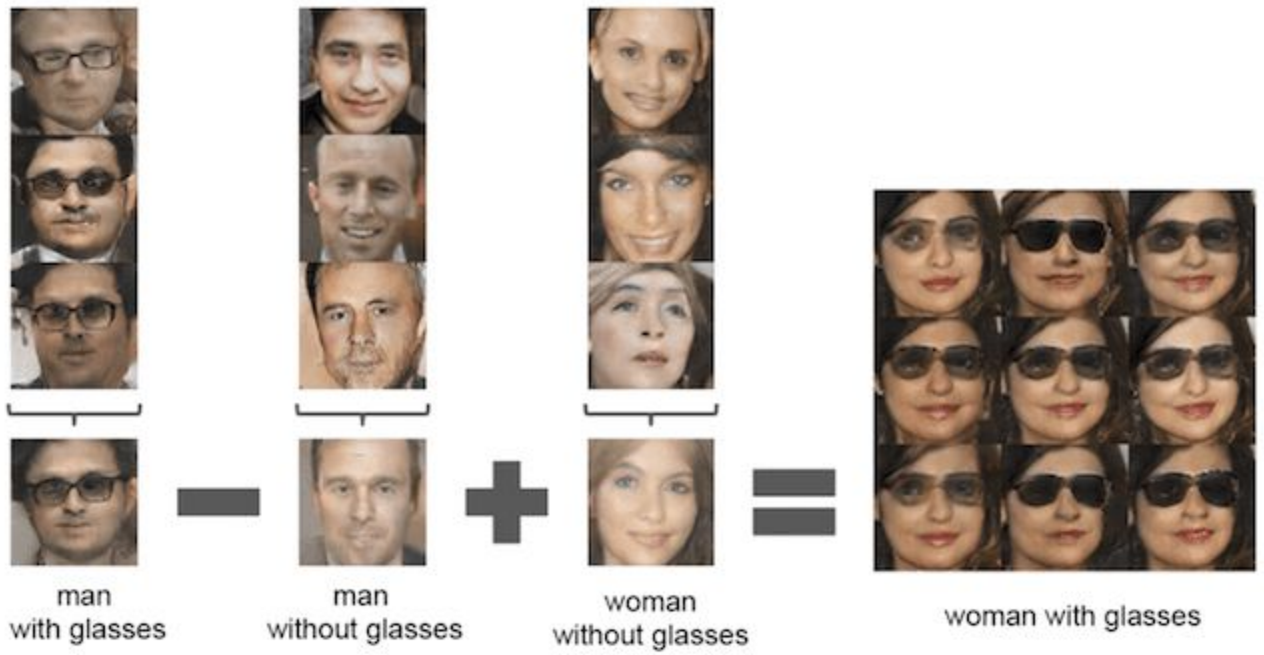
Cycle GAN

- If we had paired data (same content in both styles), this would be a supervised learning problem. But this is hard to find.
- The CycleGAN architecture learns to do it from unpaired data
    - Train two different generator nets to go from style 1 to style 2, and vice versa
    - Make sure the generated samples of style 2 are indistinguishable from real images by a discriminator net
    - Make sure the generators are cycle-consistent: mapping from style 1 to style 2 and back again should give you almost the original image

Cycle GAN


Real zebra image
The discriminator tries to distinguish generated zebra images from real ones
D
Discriminator loss: GAN generator objective, i.e. negative log probability D assigns to the sample being real
Reconstruction loss: squared error between the original image and the reconstruction
G_1
G_2
Input image (real horse image)
Generator 1 learns to map from horse images to zebra images while preserving the structure
Generated sample
Generator 2 learns to map from zebra images to horse images while preserving the structure
Reconstruction
Total loss = discriminator loss + reconstruction loss

Applications

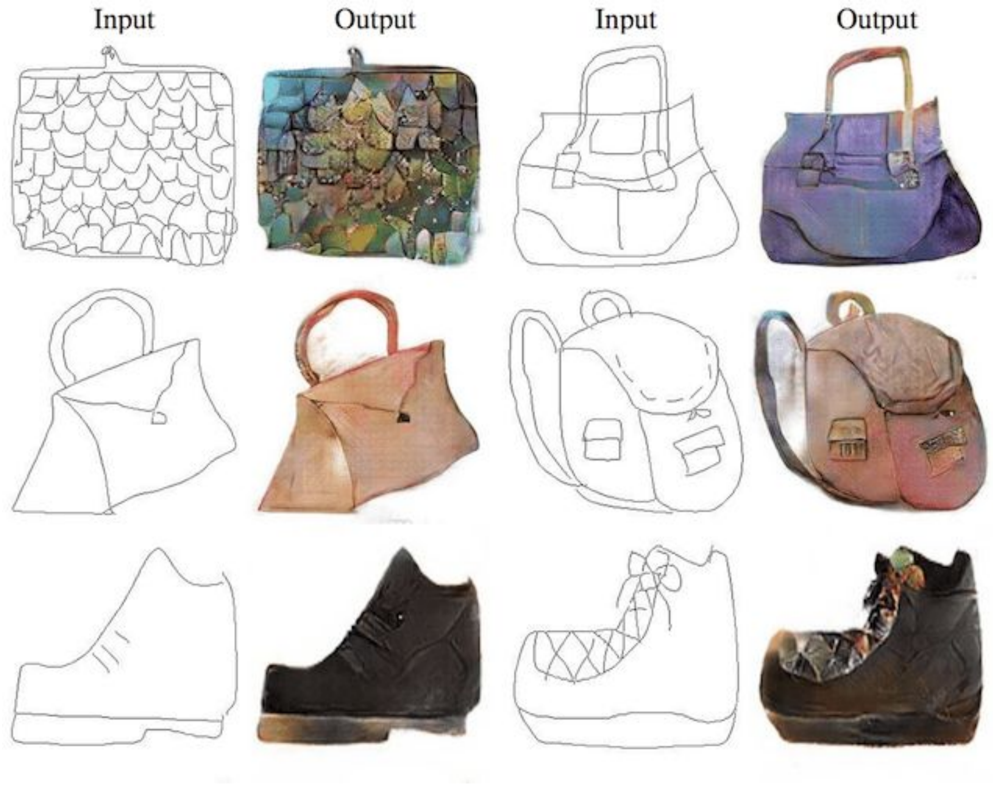


Applications

- Generate images which doesn’t exist
- Translation of

Applications

- Translation of sketches to color photographs.


Input
Output
Input
Output