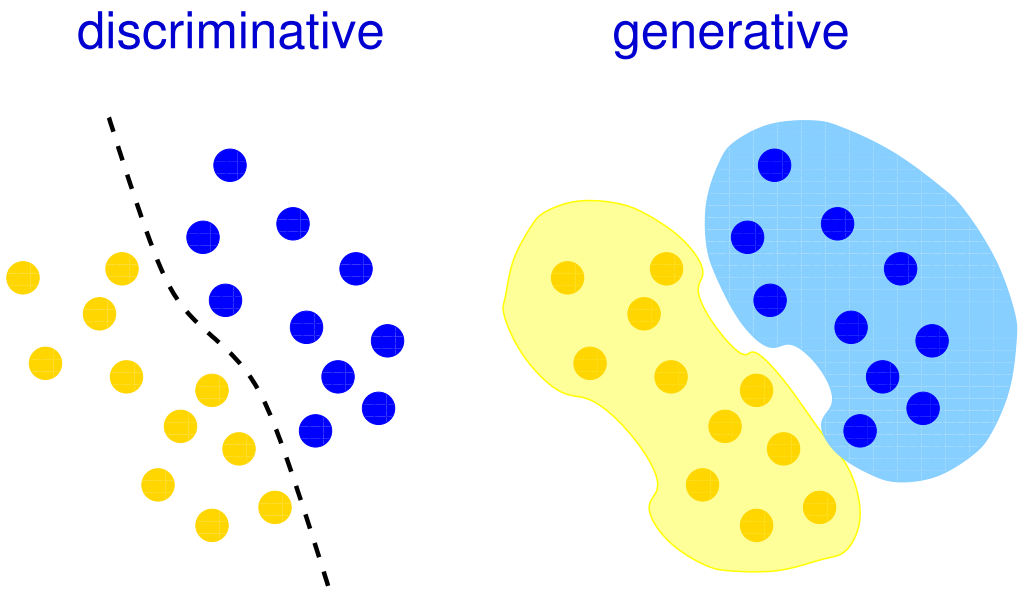
Generative Adversarial Networks

Suggested reading:
NIPS 2016 Tutorial: Generative Adversarial Networks

Generative Models



Generative Models

A Generative Adversarial Network (GAN) is a generative model (similarly to VAE).

Generative Model: a model that tries to learn the actual distribution p_{data} of real data from available samples (training set).

Goal: build probaility distribution p_{model} close to p_{data}.

We can either try to

- explicitly estimate the distribution
- build a generator able to sample according to p_{model}

Generative modles mostly focus on sample generation.

Why studying Generative Models?

- improve our knowledge on the latent representation of data and the encoding of complex high-dimensional distributions
- generative models, and GANs in particular, allow us to work with multi-modal outputs, forcing a choice instead of averaging
- find a way to produce realistic samples from a given probability distribution
- generative models can be incorporated into reinforcement learning, e.g. to predict possible futures
- ...

Multi-modal output

The output of generative models is intrinsically multimodal: e.g. generate "a car"

Even if suitably constrained (model, orientation, color, ...), there are a lot of possible outputs.

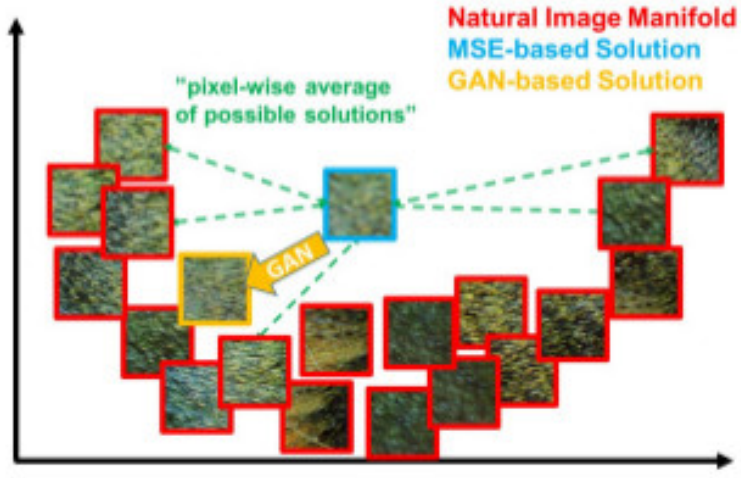
If we base learning on minimizing an average distance with all elements in the training set with the "expected features", we could end up with exceedingly blurred images.

Regions of high probability

Patches from the natural image manifold (red) and super-resolved patches obtained with MSE (blue) and GAN (orange).

Pixel-wise average of possible solutions could produce images outside the actual data manifold.

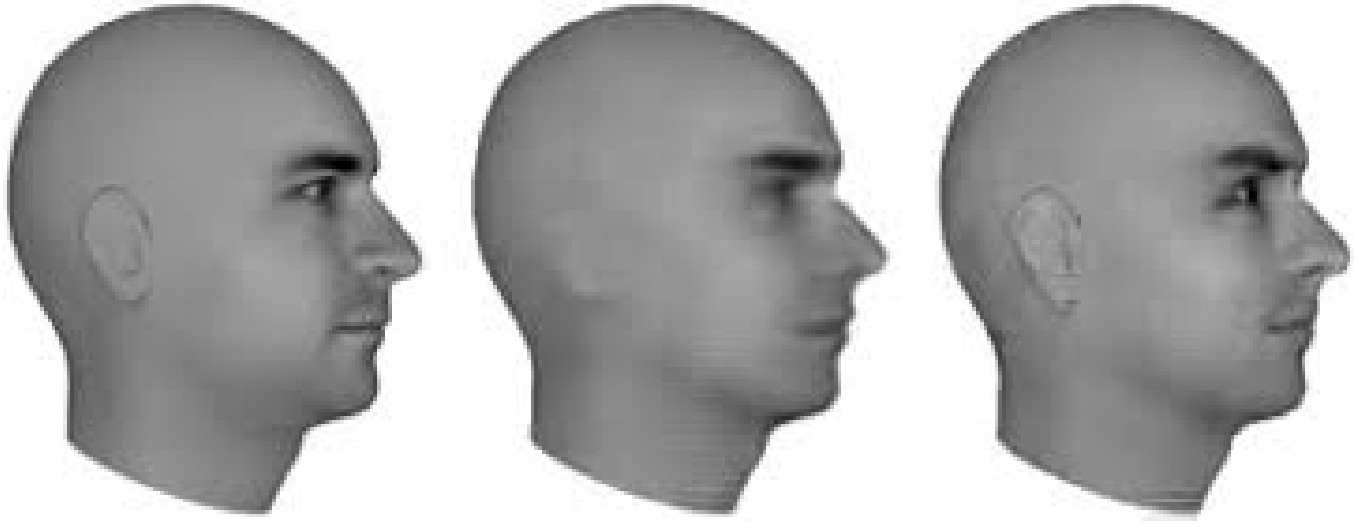
GAN drives the reconstruction towards the natural image manifold producing perceptually more convincing solutions.



picture from Photo-Realistic Single Image Super-Resolution. C.Ledig et al., 2016.

Average of possible outputs

Example: generating the next frame in a video.



Averaging on all possible outputs could produce a blurred image.

Unsupervised Learning of Visual Structure using Predictive Generative Networks. W.Lotter, G.Kreiman and D.Cox, 2015.

Another application: super-resolution

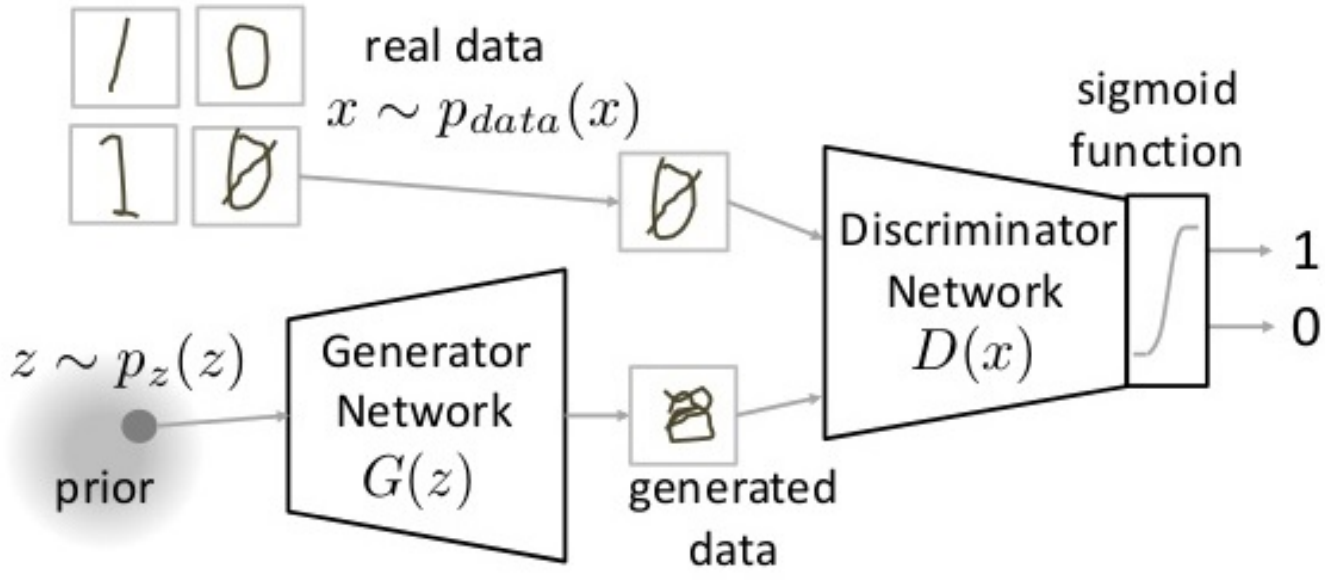
Figure 2: From left to right: bicubic interpolation, deep residual network optimized for MSE, deep residual generative adversarial network optimized for a loss more sensitive to human perception, original HR image. Corresponding PSNR and SSIM are shown in brackets. [4× upscaling]

Forcing to operate a choice - instead of mediating - could result in sharper images

Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. C.Ledig et al., 2016.

The GAN approach: a two player game

A game between the generator and the discriminator



Generative Adversarial Networks I.J.Goodfellow et al., 2014

A Min Max game

$$Min_G Max_D V(D, G)$$

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

- $\mathbb{E}_{x \sim p_{data}(x)}[log\, D(x)]$ = negative cross entropy of the discriminator w.r.t the true data distribution
- $\mathbb{E}_{z \sim p_z(z)}[log\,(1 - D(G(z)))]$ = negative cross entropy of the "false" discriminator w.r.t the fake generator

An example

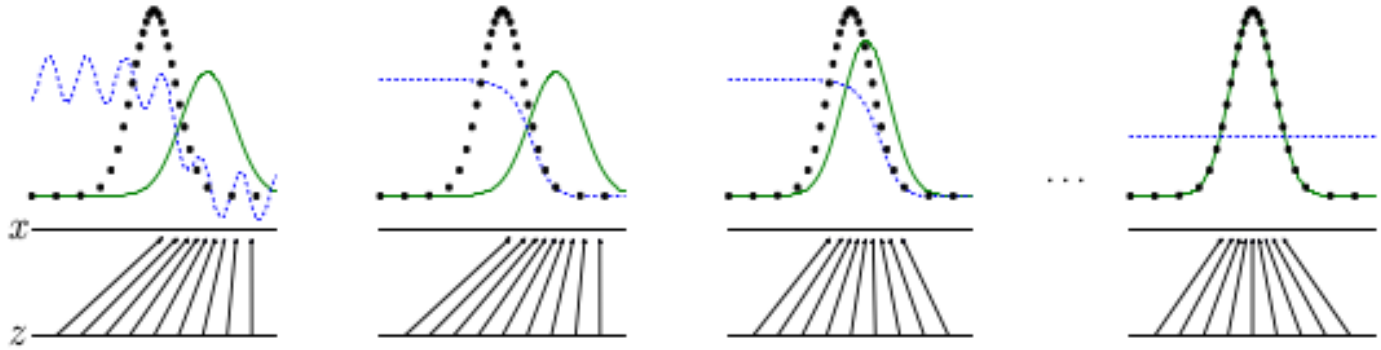


Figure 1: Generative adversarial nets are trained by simultaneously updating the **d**iscriminative distribution (D, blue, dashed line) so that it discriminates between samples from the data generating distribution (black, dotted line) $p_{\boldsymbol{x}}$ from those of the **g**enerative distribution 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. $D(\boldsymbol{x}) = \frac{1}{2}$.

Training

Alternately train the discriminator, freezing the generator, and the generator freezing the discriminator:

---

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  end for

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

A simple demo

A short video!

Demo!

Another demo!

Training problems

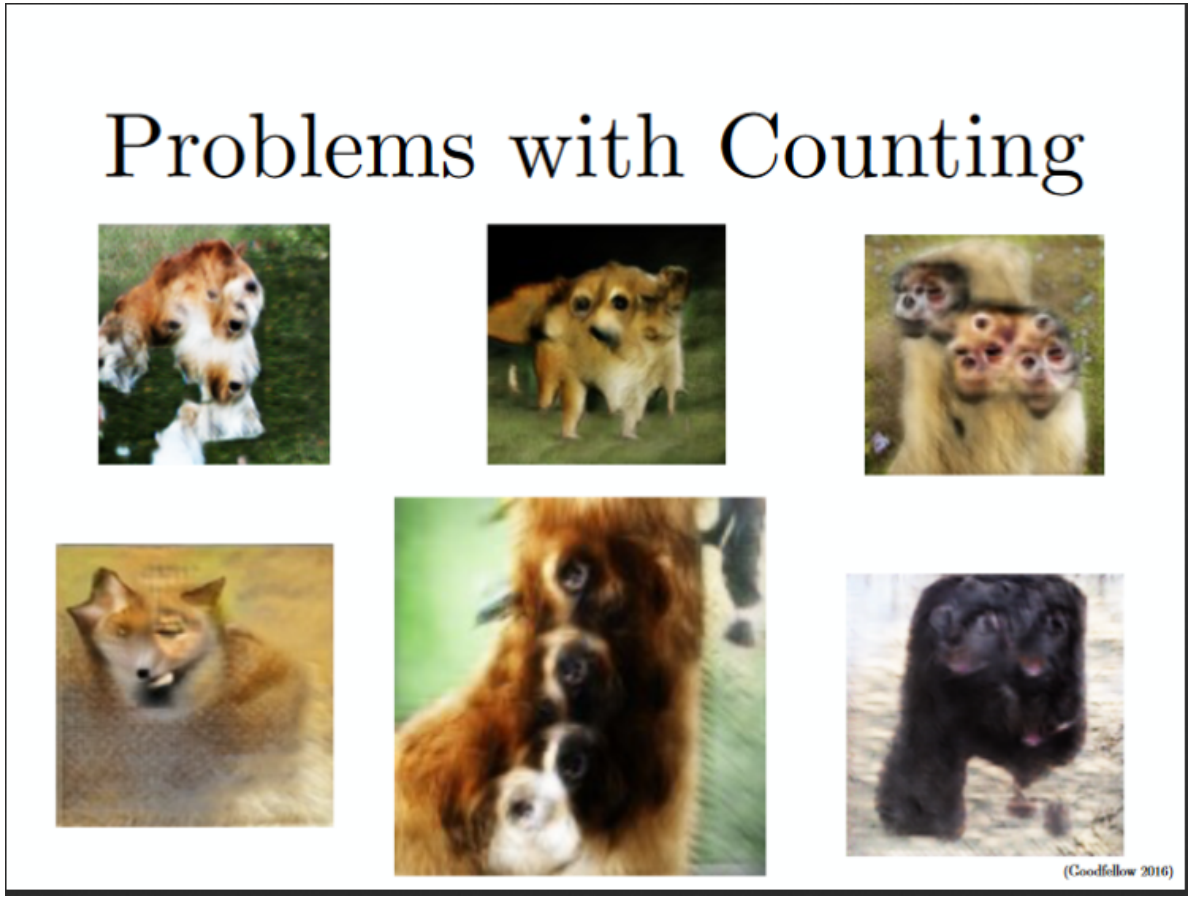
Other Problems with gans

- the fact that the discriminator get fooled does not mean the fake is good (neural networks are easily fooled)
- problems with counting, perspective, global structure, ...
- mode collapse: generative specialization on a good, fixed example

See Ian Goodfellows' slides

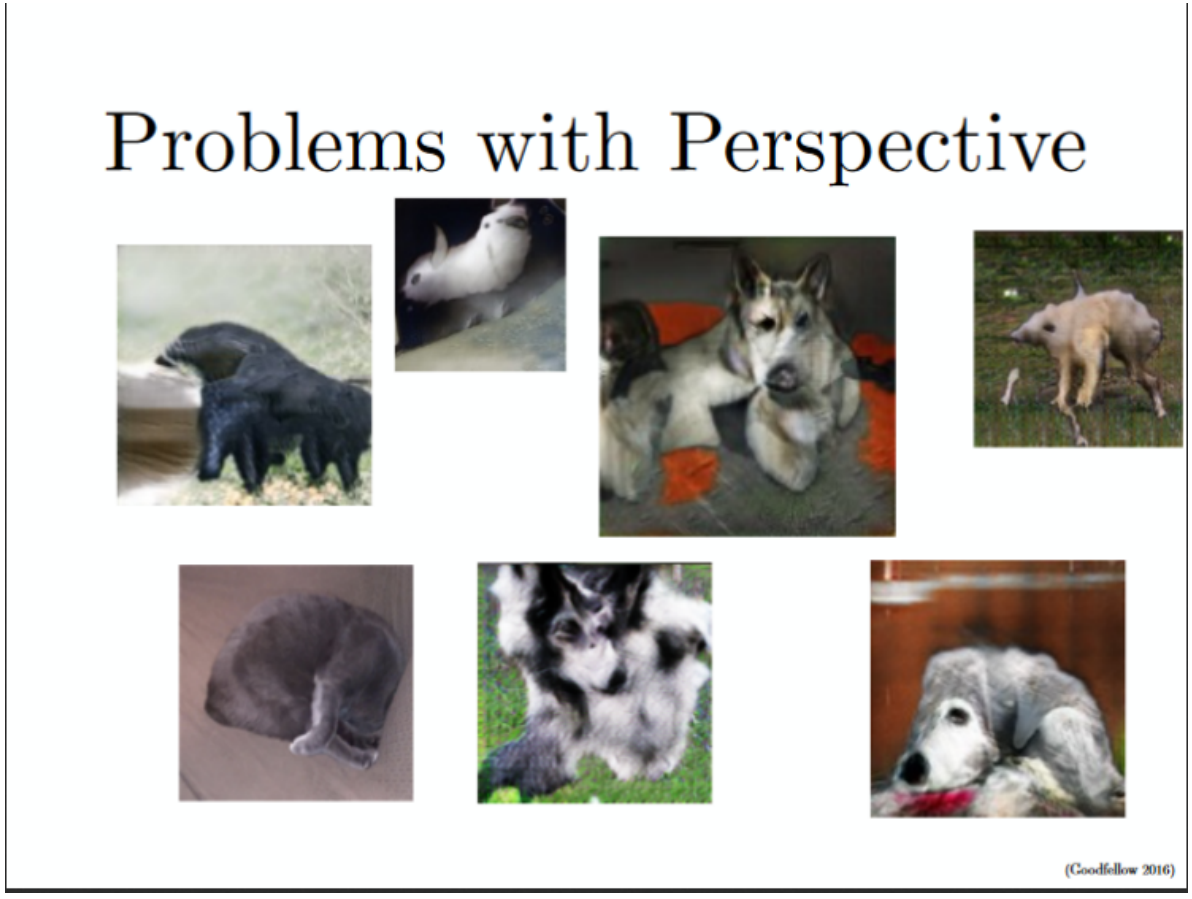
Problems with counting

GANs fail to differentiate how many of a particular object should occur at a location.



Problems with perspective

Problems to adapt to 3D objects. It doesn't understand perspective, i.e. difference between frontview and backview.



Problems with global structure

Problems to understand a holistic structure.

Integrating Gans with VAEs

Suggested reading:
VAEs and GANs

Combining VAEs and GANs

problems with VAE :

the similarity metric is crucial. Pixel-wise metrics like squared error are too sensible to local perturbations like small translations or rotations.

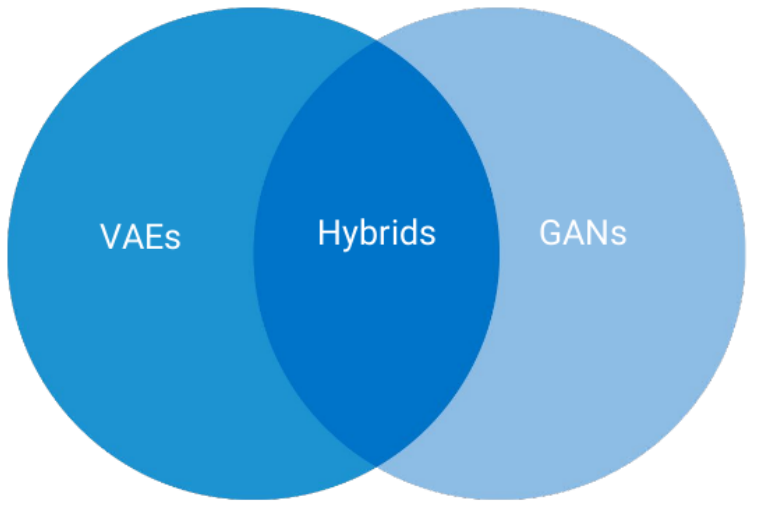
problems with GAN :

the generative approach has problems to capture the real data distribution; learning is difficult and unstable

The promise of VAEs-GANs Hybdris

improve
sample quality

improve
stability



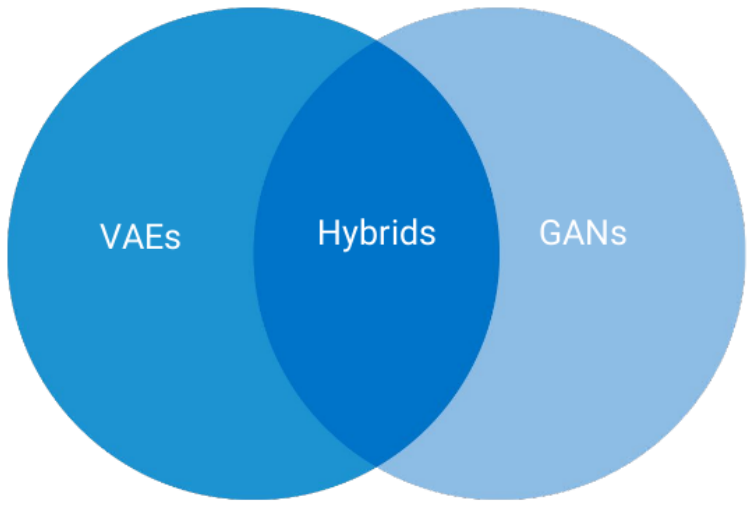
improve
representation learning

improve
diversity

The promise of VAEs-GANs Hybdris

improve
sample quality

improve
representation learning



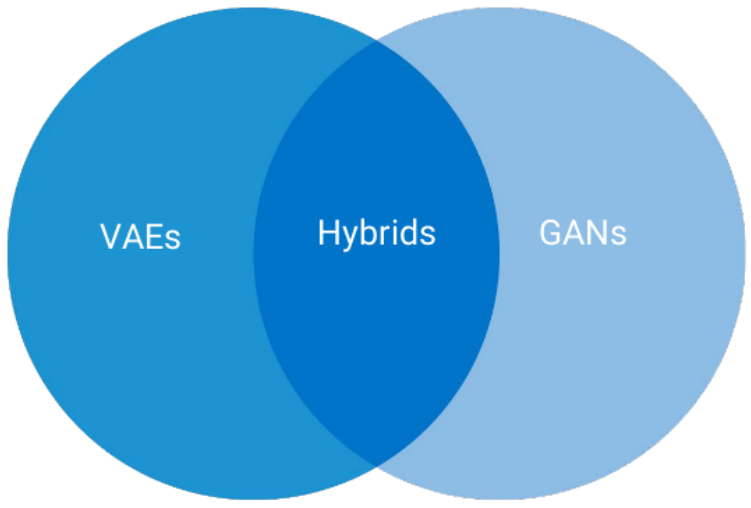
improve
stability

improve
diversity

The promise of VAEs-GANs Hybdris

improve
sample quality

improve
representation learning



improve
stability

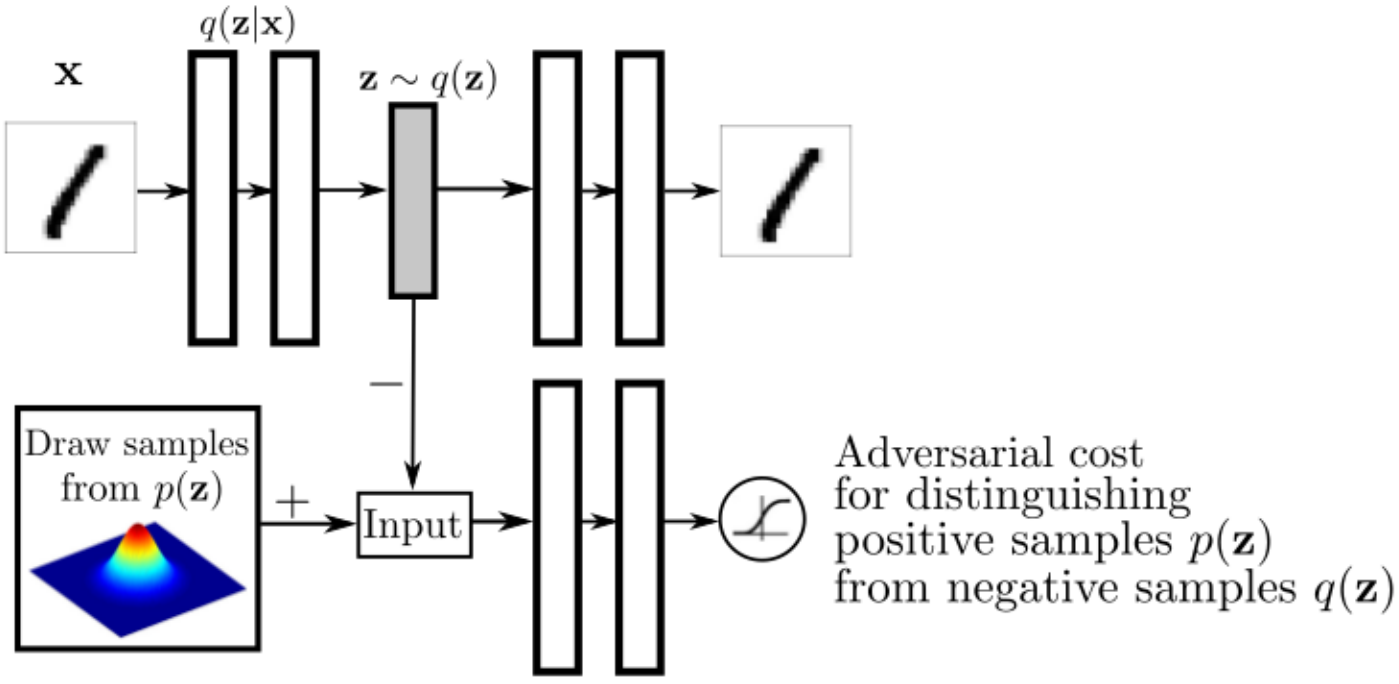
improve
diversity

Main approaches

Two main approaches:

- Acting in the **latent space**
  replace KL-divergence with a Discriminator, matching the aggregated posterior $Q(z)$ with the expected (arbitrary) prior distribution $P(z)$.
  $\Rightarrow$ Adversarial Autoencoders
- Acting in the **visible space**
  replacing the reconstruction loss with a Discriminator trying to distinguish original from reconstructed images
  $\Rightarrow$ VAE-GAN

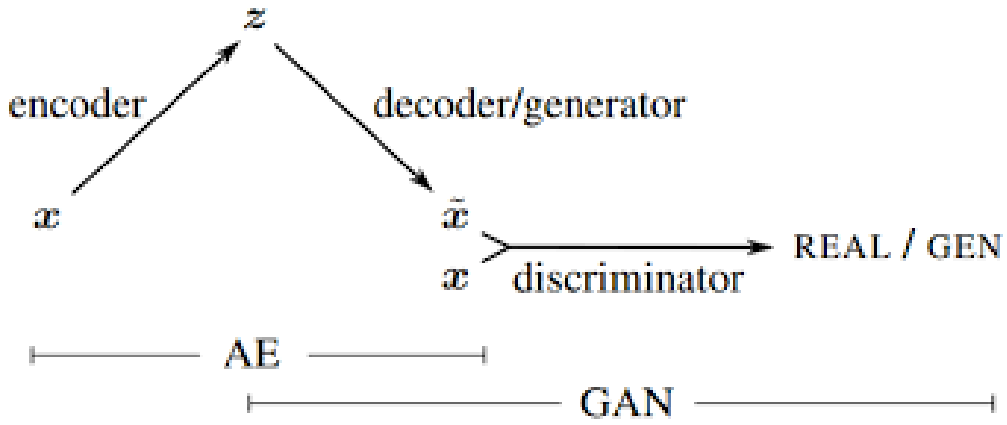
Adversarial regularization



VAE-GAN Network overview

Idea:

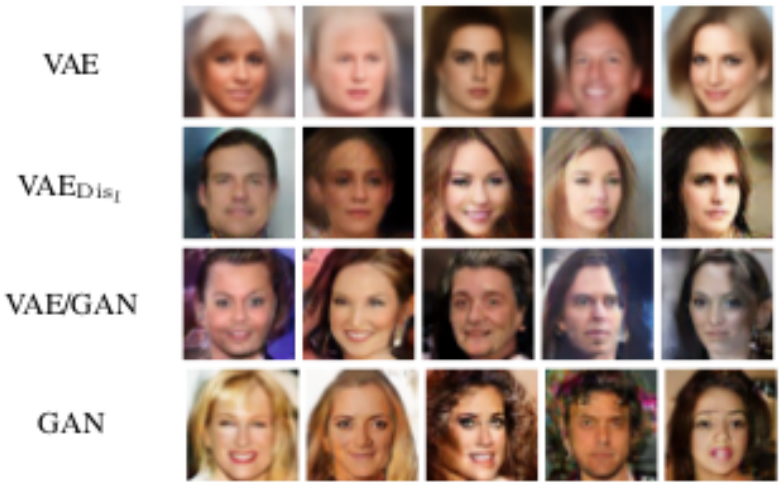
- replace the element-wise reconstruction metric with the feature-wise metric learned by the discriminator.
- improve the generative part of GAN with a variational autoencoder



Autoencoding beyond pixels using a learned similarity metric, A.B.L.Larsen, S.K.Sonderby, H.Larochelle, O.Winther, 2016

Results on the CelebA dataset

CelebA (Liu et al. 2015): dataset of 202,599 face images annotated with 40 binary attributes such as eye-glasses, bangs, pale skin etc.



generation

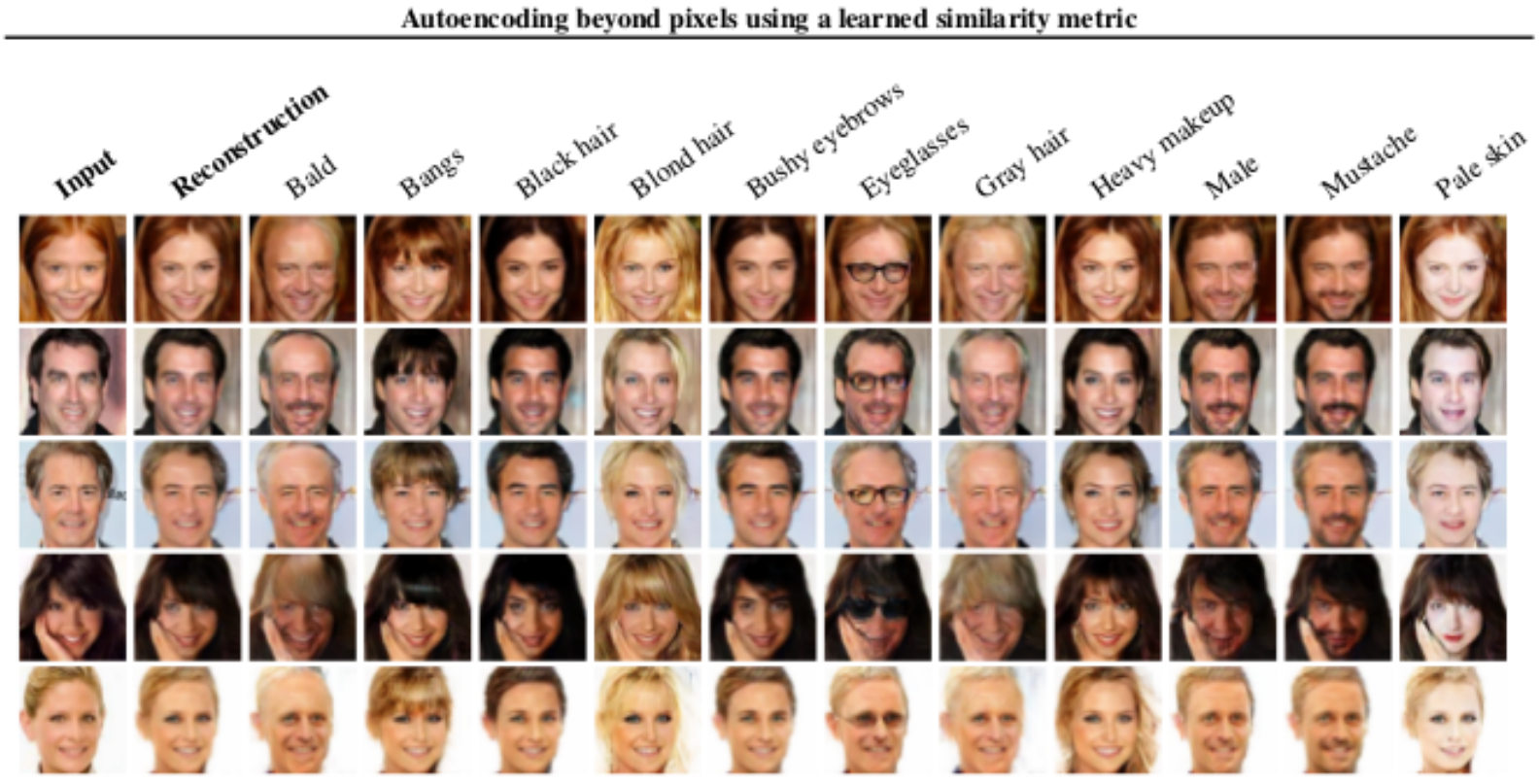
reconstruction

Attribute vectors

Exploit the rich labeling of CelebA to find directions in the latent space corresponding to specific visual features in the image space

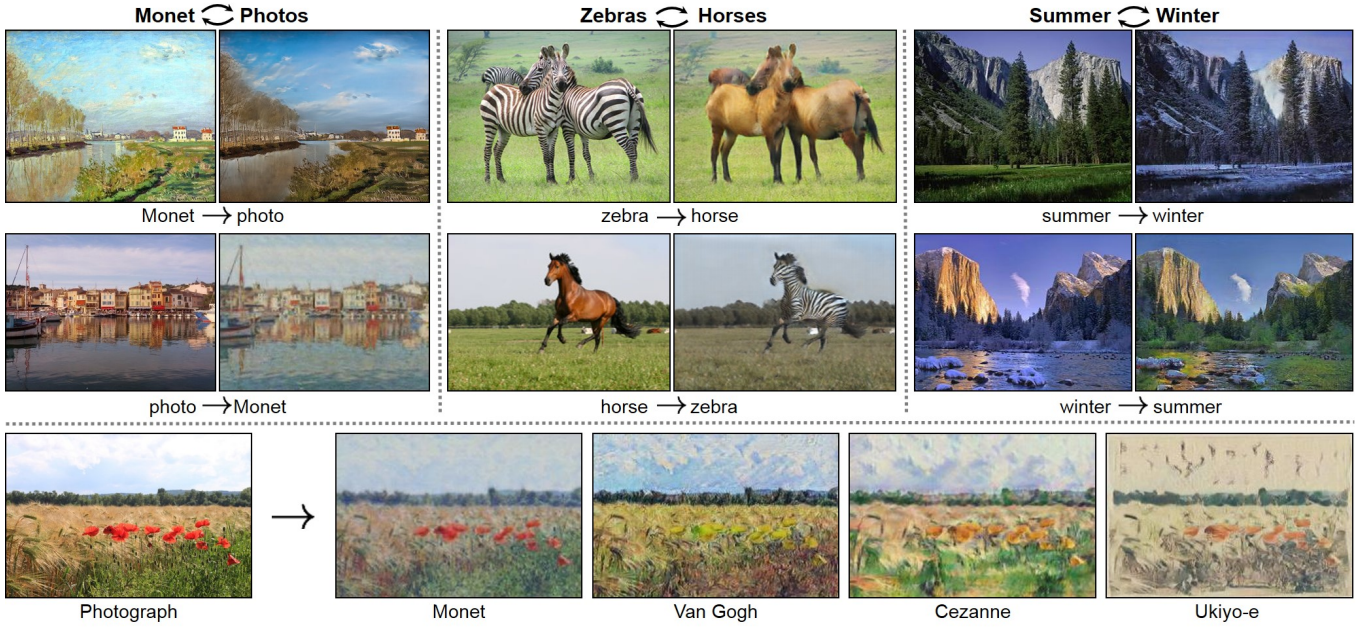
Visual attribute vector:
For each attribute, compute the mean vector for images with the attribute and the mean vector for images without the attribute, and take their difference.
Use attribute vectors to modify the latent space before reconstruction, in order to add a specific feature.

Adding attributes



Next arguments

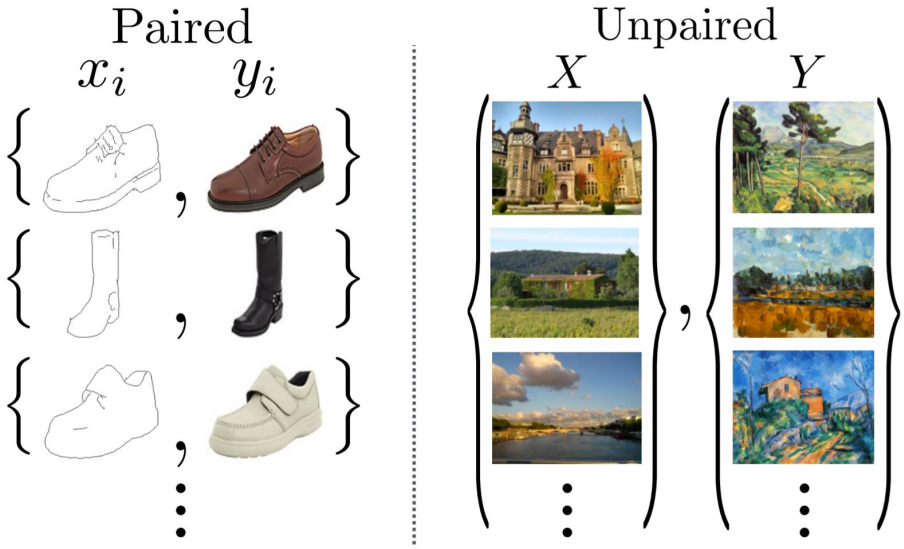
Cycle gans



Unpaired Image-to-image translation with Cycle Generative Adversarial Networks

Unpaired images

For training, we need images in two classes, but **not** paired images



picture from article

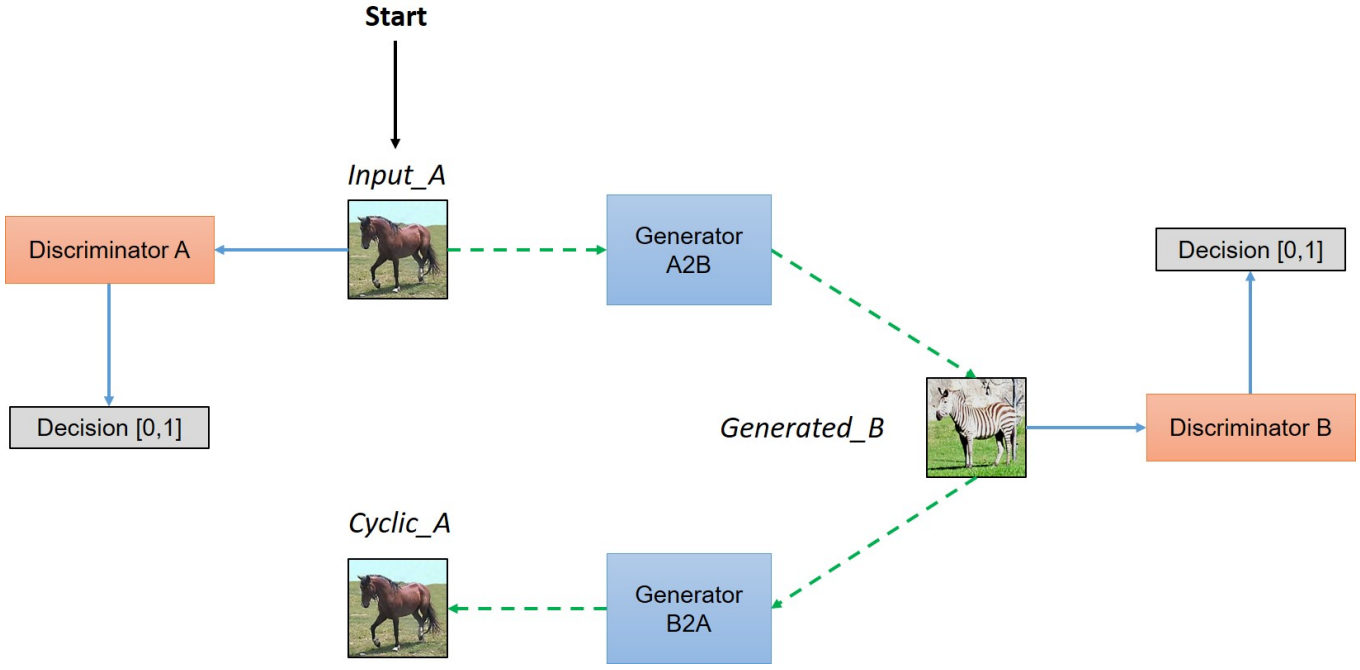
Cycle-consistent adversarial networks

A Generator and a Discriminator network playing against each other. The generator tries to produce samples from the desired distribution, based on an input of the other distibution, and the discriminator tries to predict if the sample belongs to the actual distribution or it was produced by the generator.

This cannot guarantee to generate an image in the target distribution related to the original image.

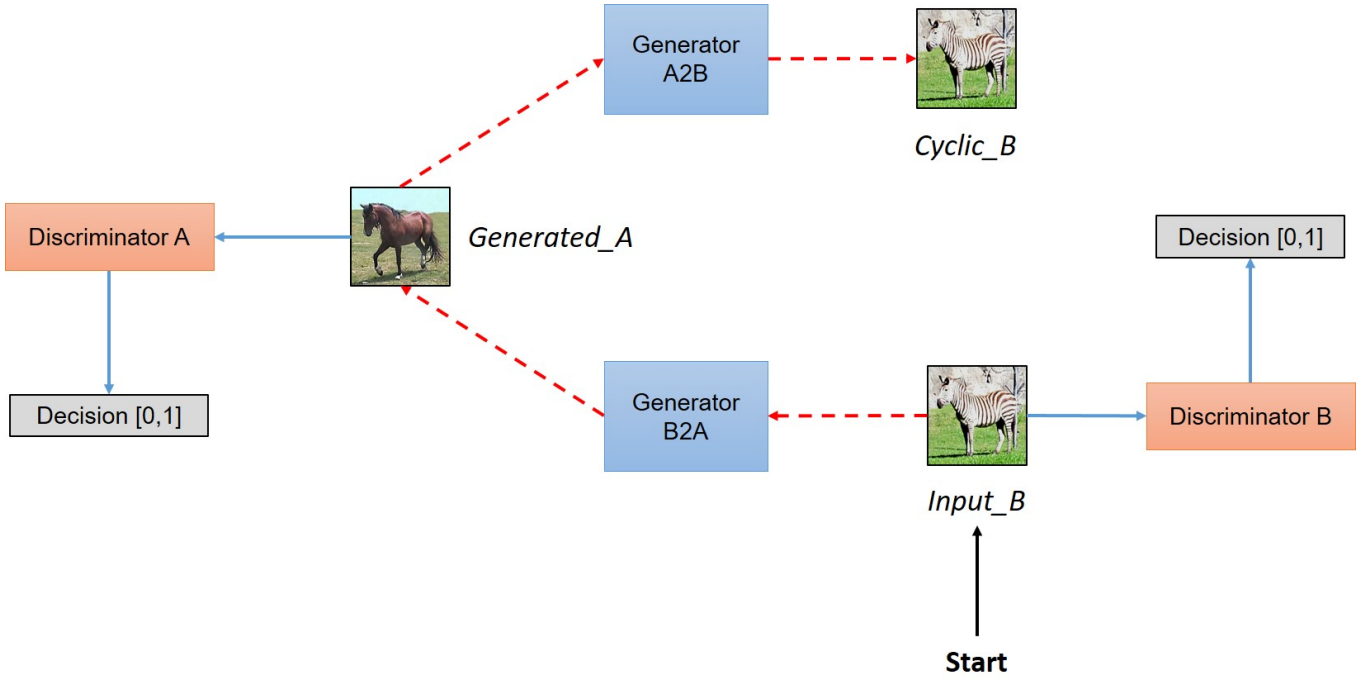
To this aim, the authors introduce the cycle-consistency constraint: if we transform from source distribution to target and then back again to source distribution, we would expect to obtain the original image.

The network architecture: forward ...



Pictures from Understanding and Implementing CycleGAN in TensorFlow

and back



Pictures from Understanding and Implementing CycleGAN in TensorFlow

The loss function

Given mappings $G : X \to Y$, $F : Y \to X$ and discriminators D_Y, D_X we have:

- Adversarial loss

$$\begin{aligned}\mathcal{L}_{GAN}(G; D_Y) &= \mathbb{E}_{y \sim p_{data}(y)}[log(D_Y(y)] \\ &+ \mathbb{E}_{x \sim p_{data}(x)}[log(1 - D_Y(G(x)))]\end{aligned}$$

- Cycle consistency loss

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

The network structure

- each mapping is defined as a sequence of convolutions followed by transposed convolutions
- to improve similarity bewteen input and output, the authors also exploit residuality