Generative Models in Computer Vision

Meysam Madadi

Discriminative vs. generative models

- Discriminative models learn
  - Conditional probability,
  - Boundaries between classes,
  - Mainly supervisedly.
- Generative models learn
  - Joint probability,
  - Data distribution,
  - Mainly unsupervisedly.

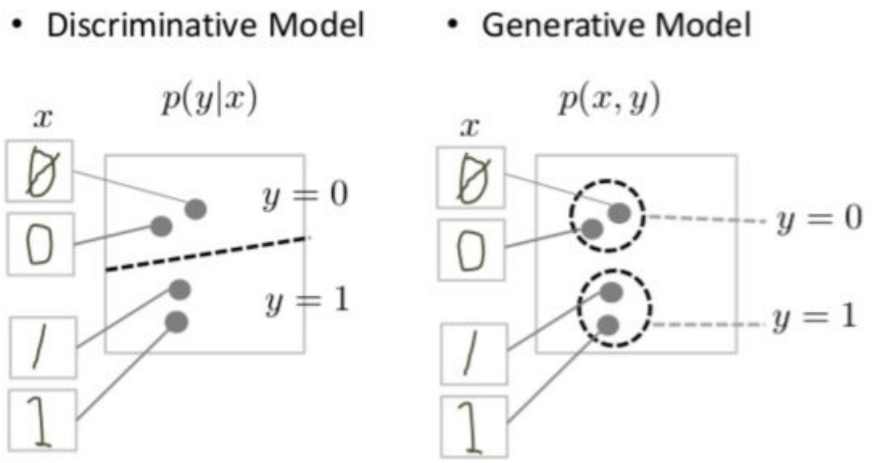


Image credit: https://developers.google.com/machine-learning/gan/generative

Discriminative vs. generative models

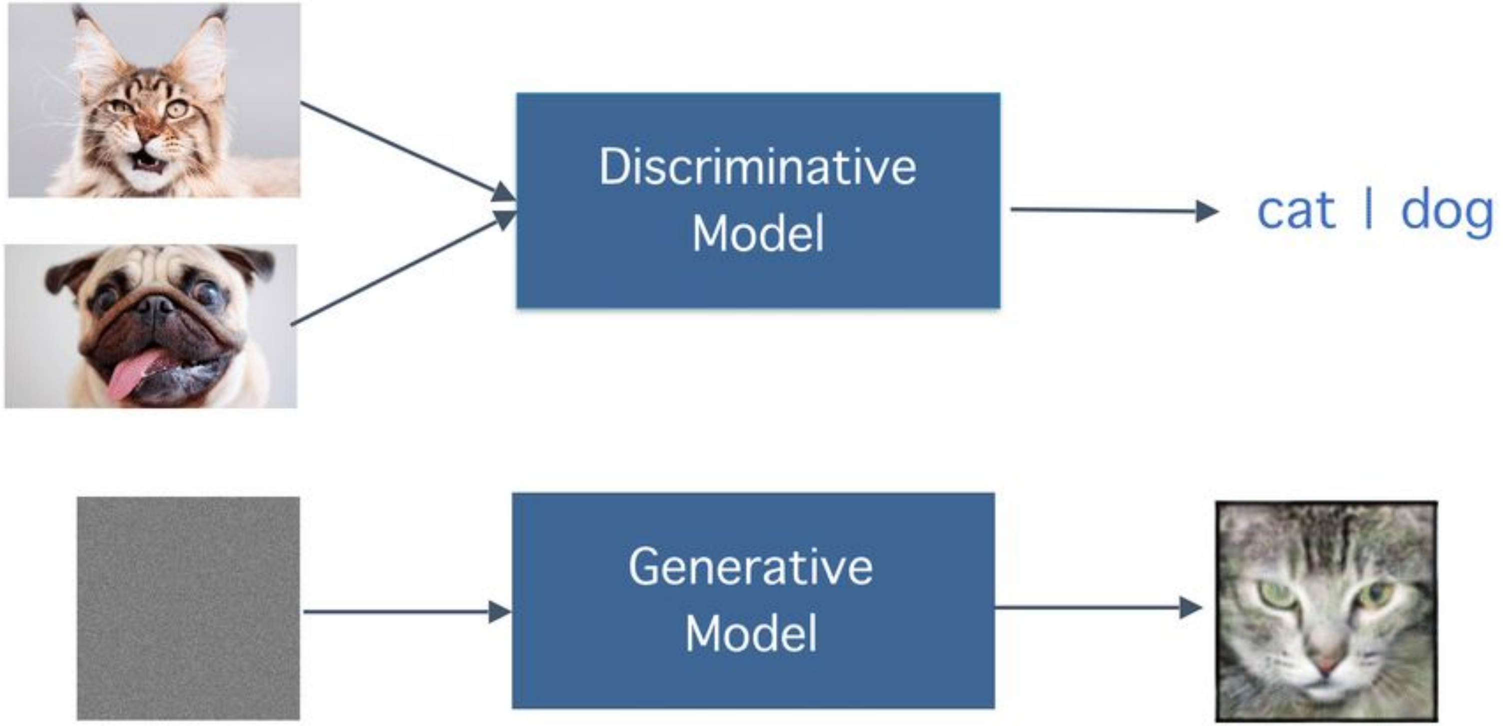
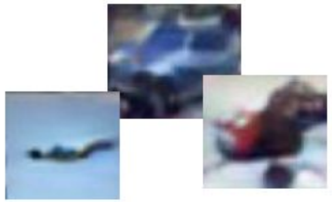


Image credit: https://arxiv.org/pdf/2005.08679.pdf

Generative models

Given training data, generate new samples from same distribution

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

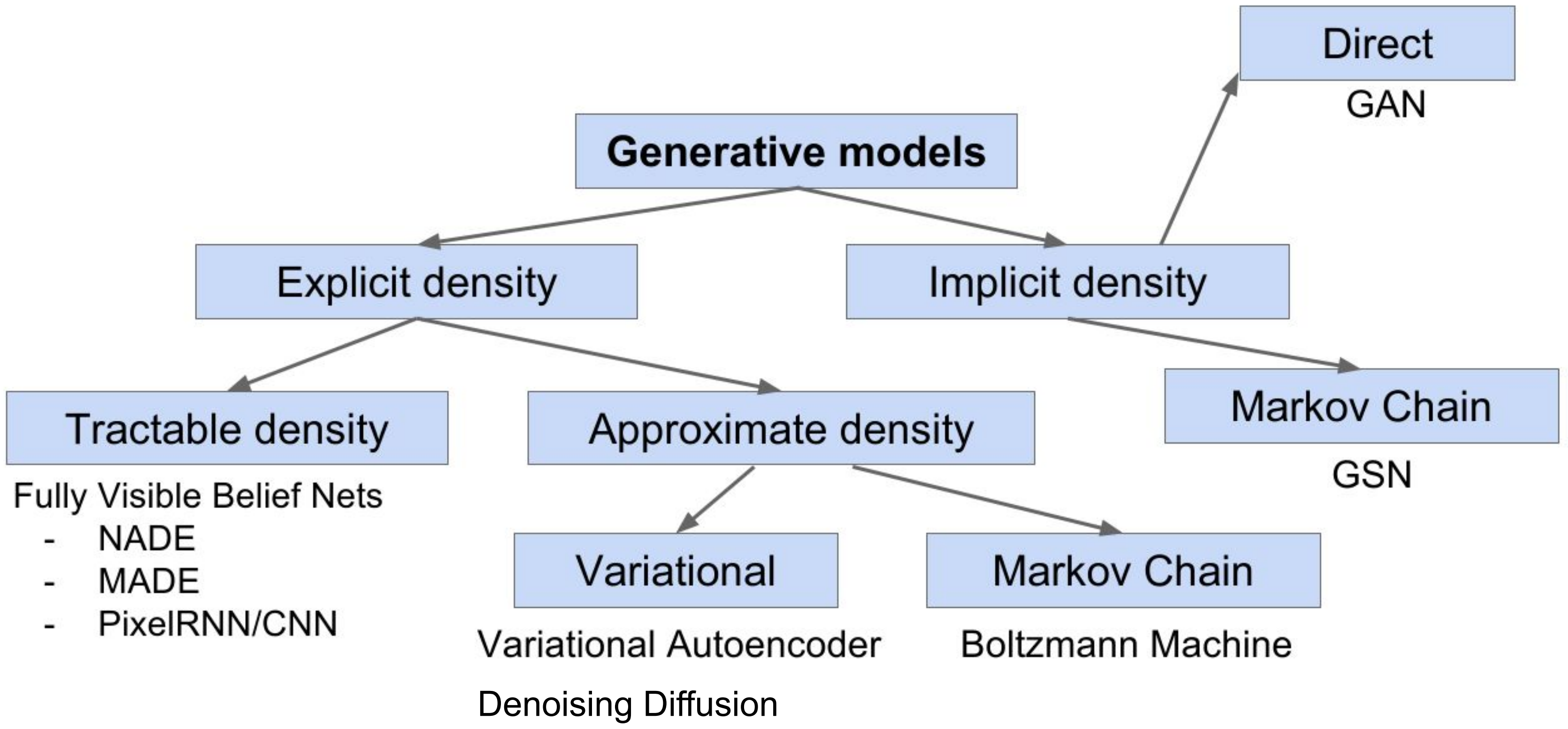


Generated samples ~ $p_{model}(x)$

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

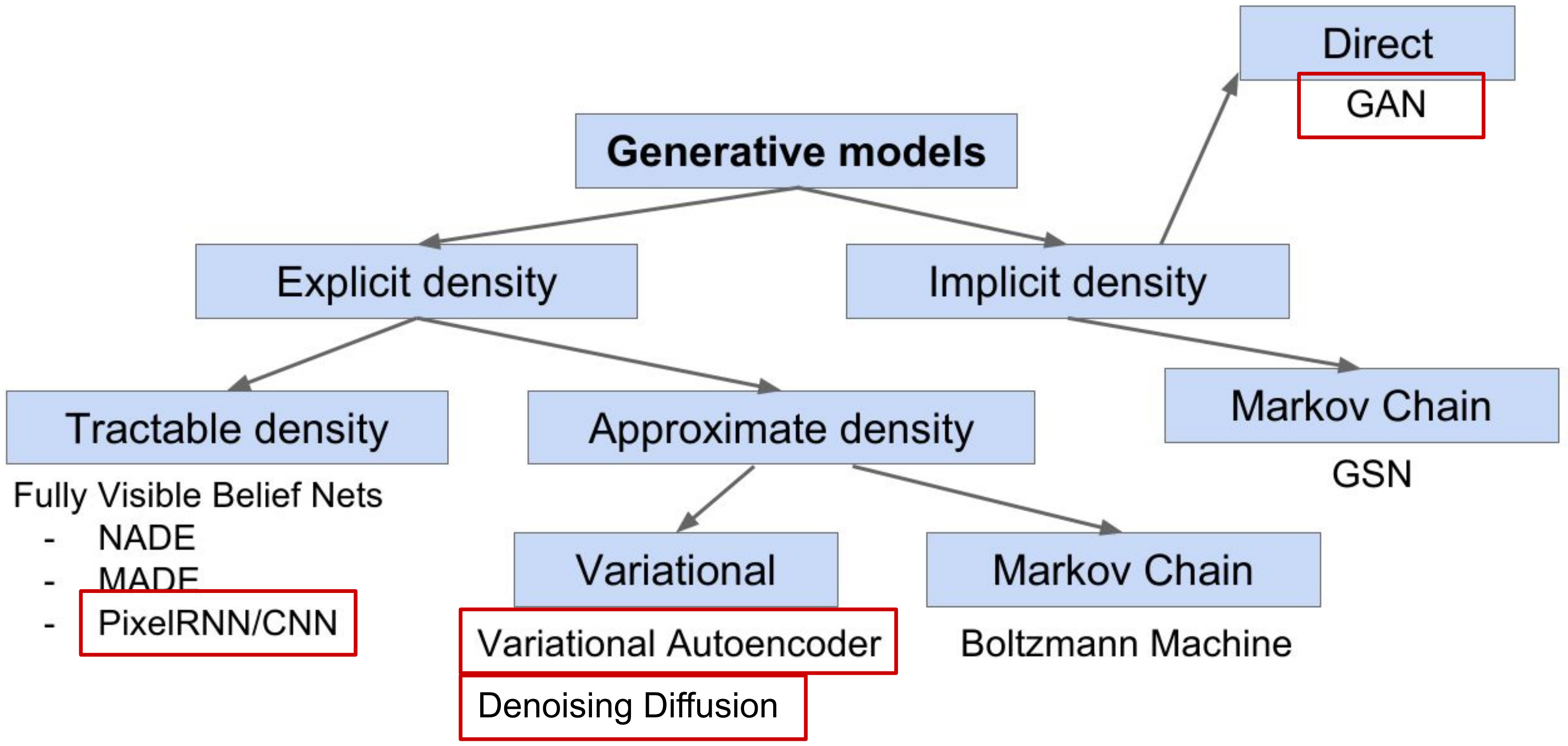
Slide credit: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf

Taxonomy of generative models



Slide credit: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf

Taxonomy of generative models



PixelRNN/CNN

Fully visible belief network

Explicit density model

Use chain rule to decompose likelihood of an image x into product of 1-d distributions:

$$\underset{\substack{\uparrow \\ \text{Likelihood of} \\ \text{image x}}}{p(x)} = \prod_{i=1}^{n} \underset{\substack{\uparrow \\ \text{Probability of i'th pixel value} \\ \text{given all previous pixels}}}{p(x_i|x_1, \ldots, x_{i-1})}$$

Then maximize likelihood of training data

Slide credit: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf

Fully visible belief network

Explicit density model

Use chain rule to decompose likelihood of an image x into product of 1-d distributions:

$$p(x) = \prod_{i=1}^{n} p(x_i | x_1, ..., x_{i-1})$$

Likelihood of image x

Probability of i'th pixel value given all previous pixels

Will need to define ordering of "previous pixels"

Complex distribution over pixel values => Express using a neural network!

Then maximize likelihood of training data

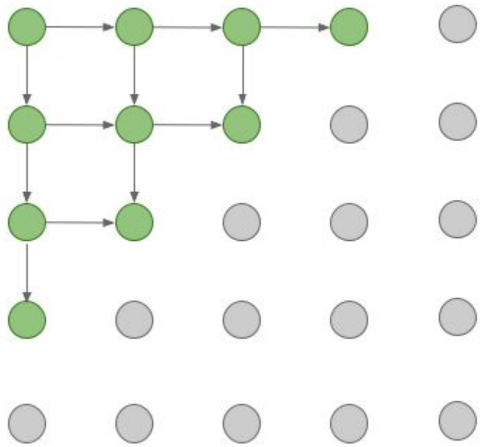
Slide credit: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf

PixelRNN

Generate image pixels starting from corner

Dependency on previous pixels modeled using an RNN (LSTM)

Drawback: sequential generation is slow!



Slide credit: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf

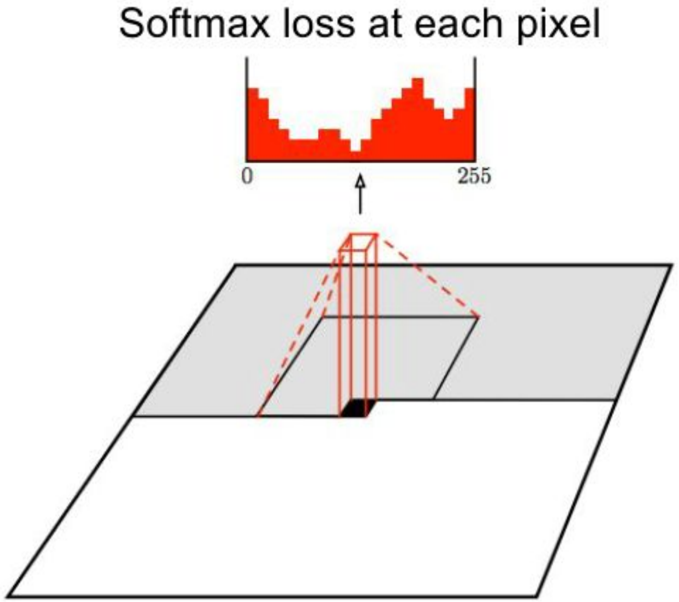
PixelCNN

Still generate image pixels starting from corner

Dependency on previous pixels now modeled using a CNN over context region

Training is faster than PixelRNN
(can parallelize convolutions since context region values known from training images)

Generation must still proceed sequentially
=> still slow

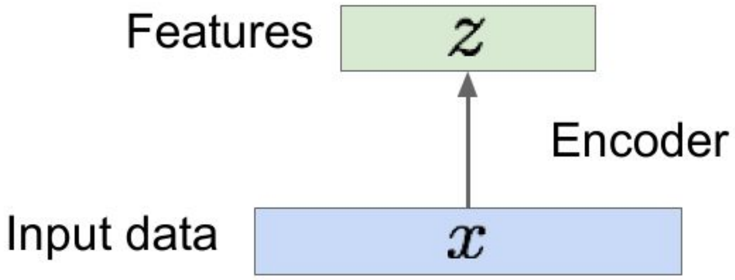


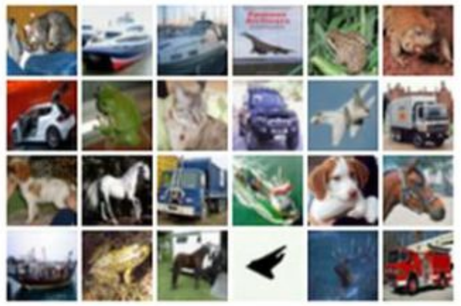
Slide credit: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf

Variational AutoEncoders (VAE)

Some background first: Autoencoders

Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data





Slide credit: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf

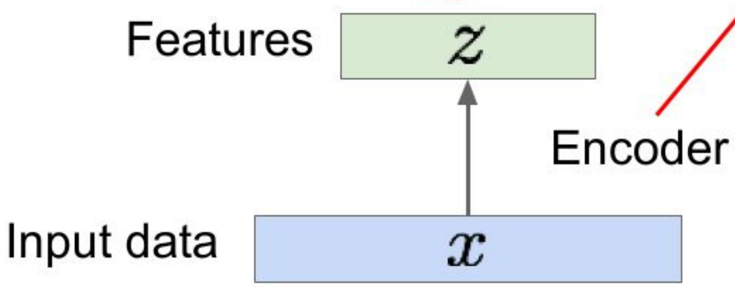
Some background first: Autoencoders

Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data

z usually smaller than **x** (dimensionality reduction)

Q: Why dimensionality reduction?

Originally: Linear + nonlinearity (sigmoid)
Later: Deep, fully-connected
Later: ReLU CNN



Slide credit: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf

Some background first: Autoencoders

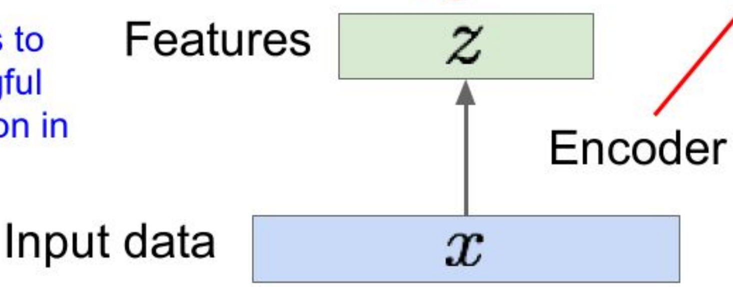
Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data

z usually smaller than **x** (dimensionality reduction)

Q: Why dimensionality reduction?

A: Want features to capture meaningful factors of variation in data

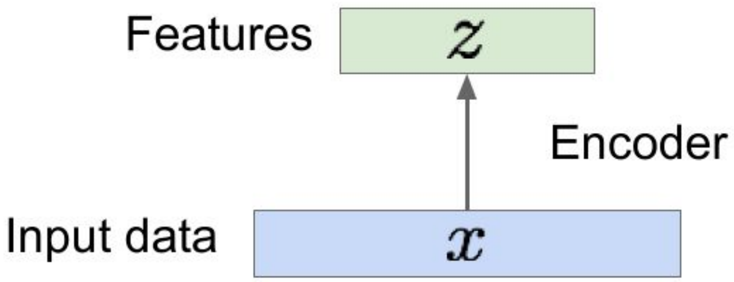
Originally: Linear + nonlinearity (sigmoid)
Later: Deep, fully-connected
Later: ReLU CNN

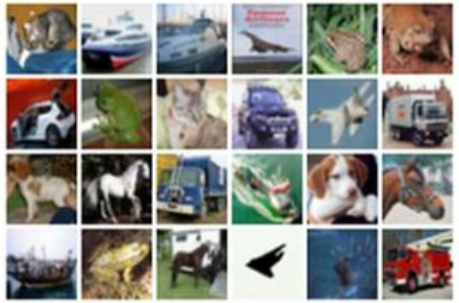


Slide credit: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf

Some background first: Autoencoders

How to learn this feature representation?





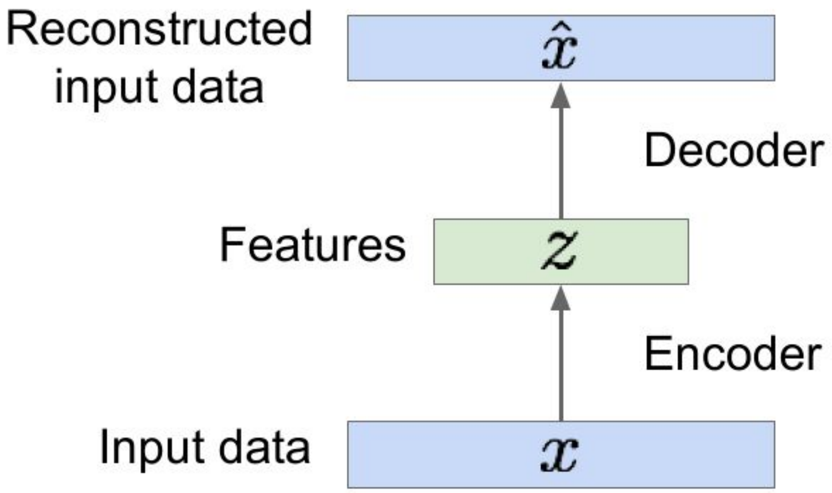
Slide credit: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf

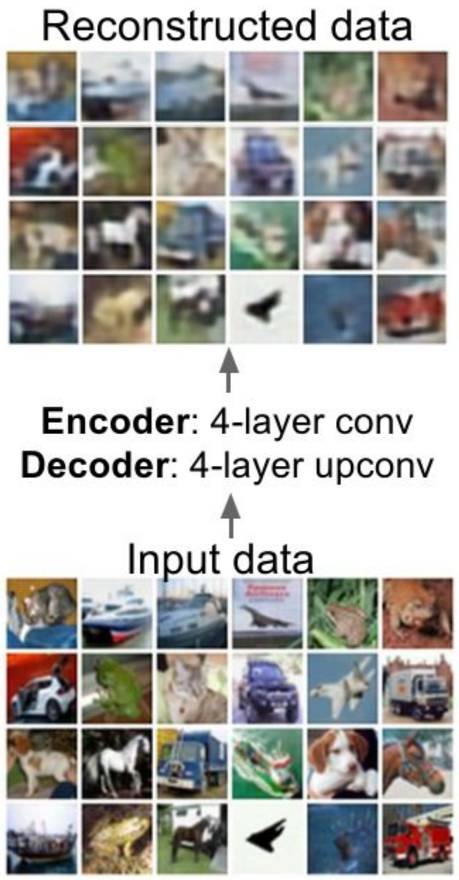
Some background first: Autoencoders

How to learn this feature representation?

Train such that features can be used to reconstruct original data

“Autoencoding” - encoding itself



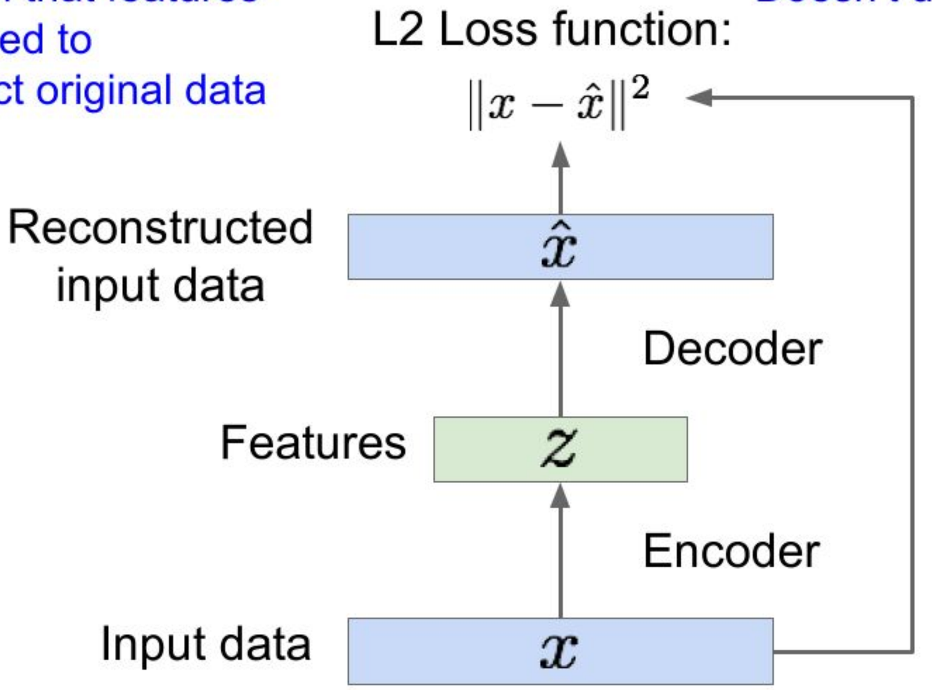


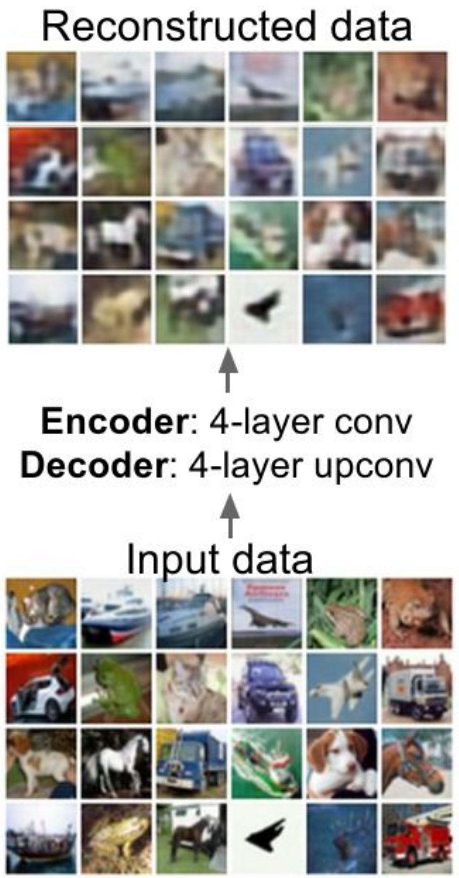
Slide credit: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf

Some background first: Autoencoders

Train such that features can be used to reconstruct original data

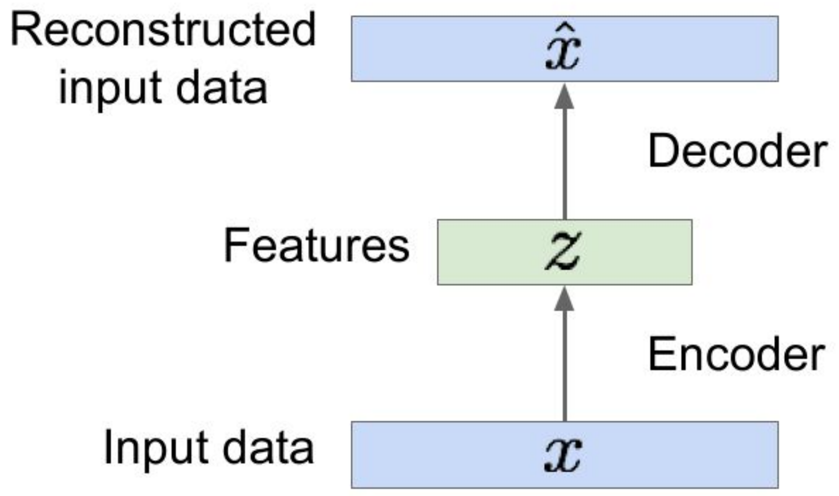
Doesn't use labels!





Slide credit: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf

Some background first: Autoencoders



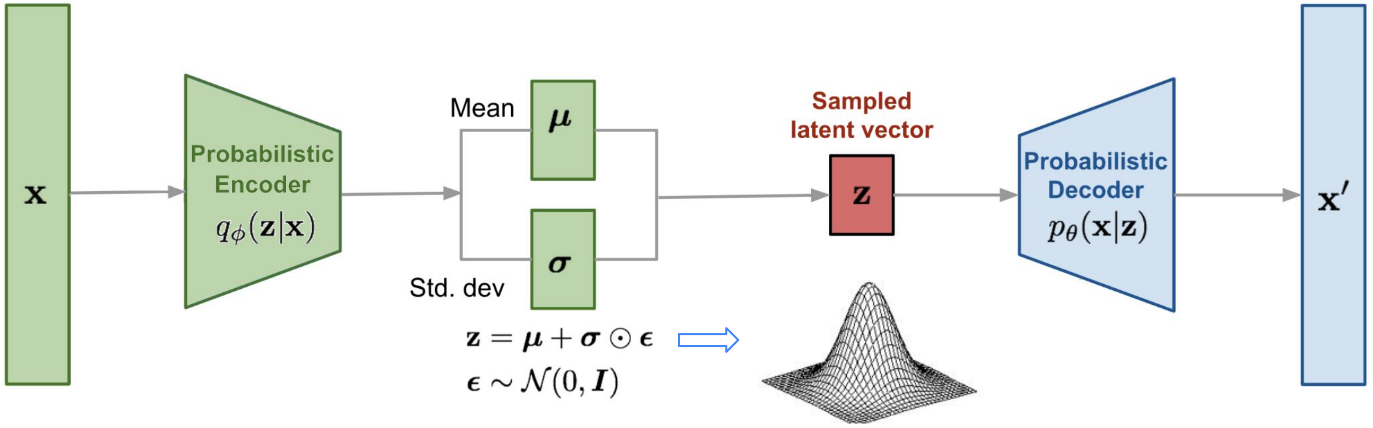
Autoencoders can reconstruct data, and can learn features to initialize a supervised model

Features capture factors of variation in training data. Can we generate new images from an autoencoder?

Slide credit: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf

Variational Autoencoders

- The latent space in autoencoders is deterministic and not continuous, and can not be sampled from,
- Solution: make autoencoders probabilistic,
- How:
  a. By defining a prior distribution for the latent space: a unit Gaussian,
  b. Reparameterization trick: include a random noise in the sampling to make autoencoder stochastic while trainable


$\mathbf{x}$
Probabilistic Encoder
$q_\phi(\mathbf{z}|\mathbf{x})$
Mean
$\boldsymbol{\mu}$
Std. dev
$\boldsymbol{\sigma}$
$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$
$\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I})$
Sampled latent vector
$\mathbf{z}$
Probabilistic Decoder
$p_\theta(\mathbf{x}|\mathbf{z})$
$\mathbf{x}'$

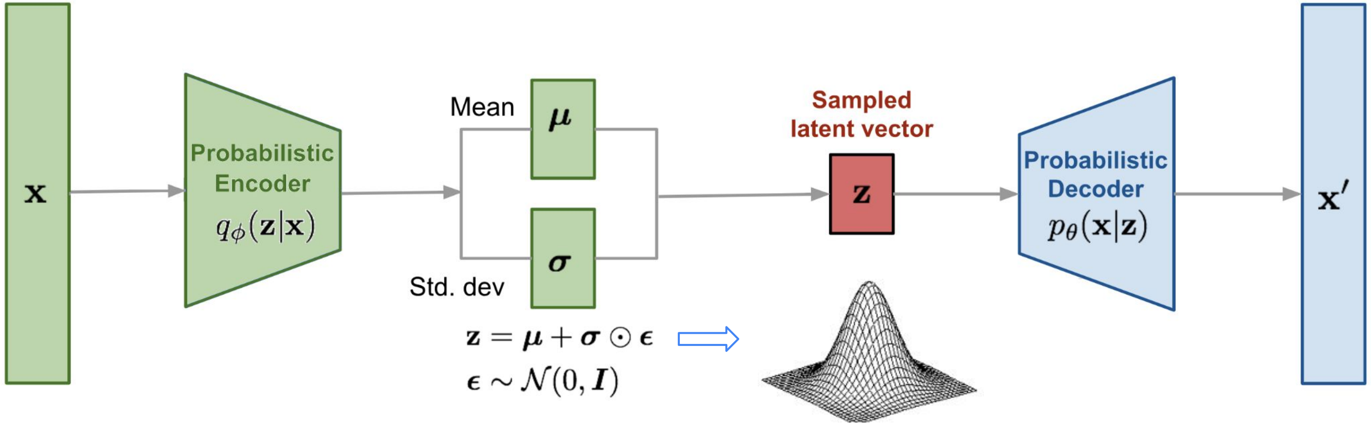
Variational Autoencoders: How to train

- Minimize the negative log likelihood,
- Estimate the true parameters of the posterior ($p_\theta(z|x)$) by optimizing a lower bound

$$-\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) + D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}))$$

Reconstruction loss
Usually L1 or L2

Kullback-Lieber divergence
Force the latent code to be a unit Gaussian



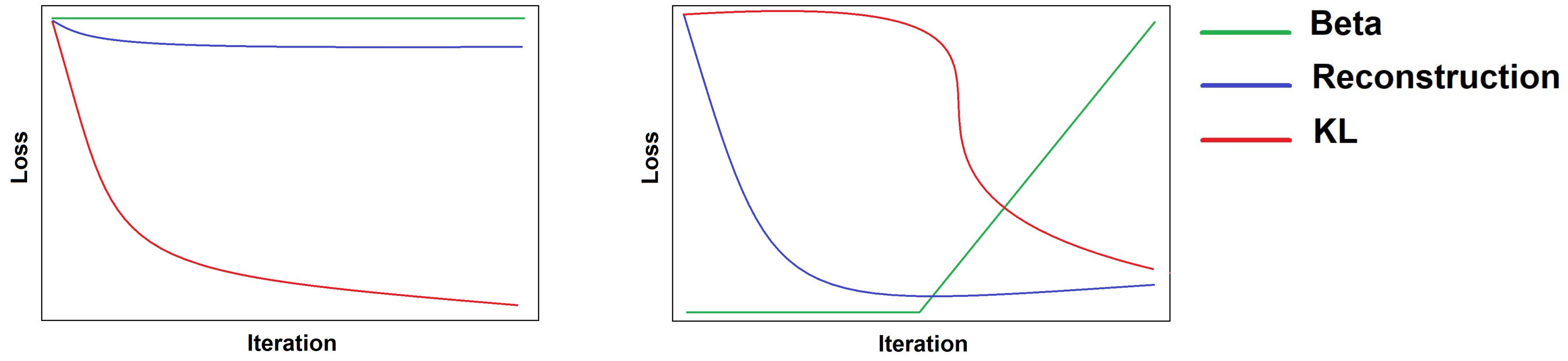
$$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \epsilon$$
$$\epsilon \sim \mathcal{N}(0, \boldsymbol{I})$$

Variational Autoencoders: How to train (β-VAE)

- Reconstruction and KL losses compete against each other,
- Solution: introduce a hyperparameter to balance them:

$$-\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) + \beta D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}))$$

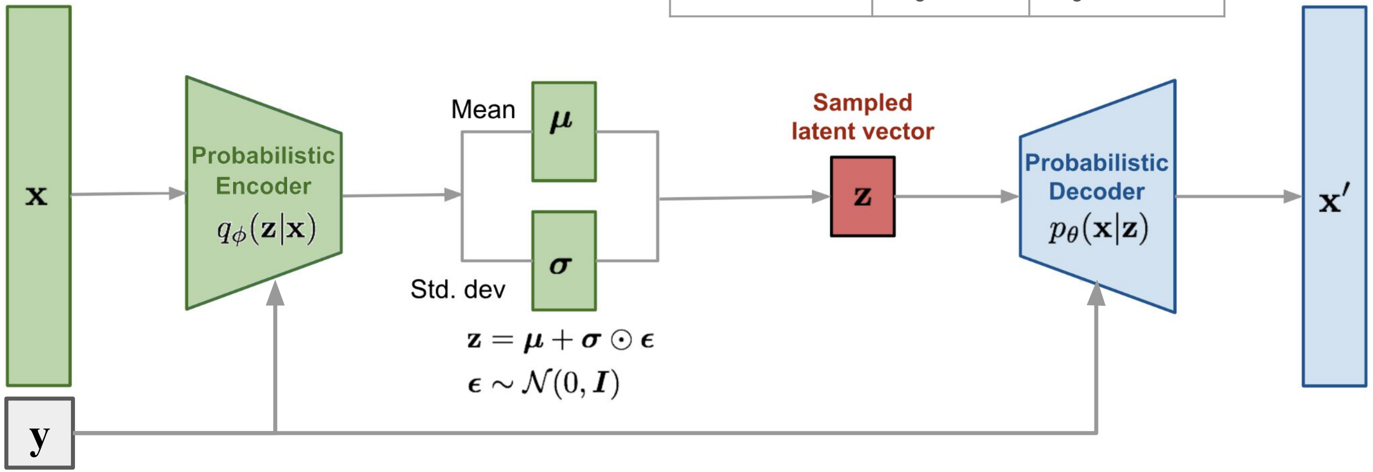
A balancing term with scheduling



Conditional Variational Autoencoders

- Requires labeled data,
- Provides more control over the generation,

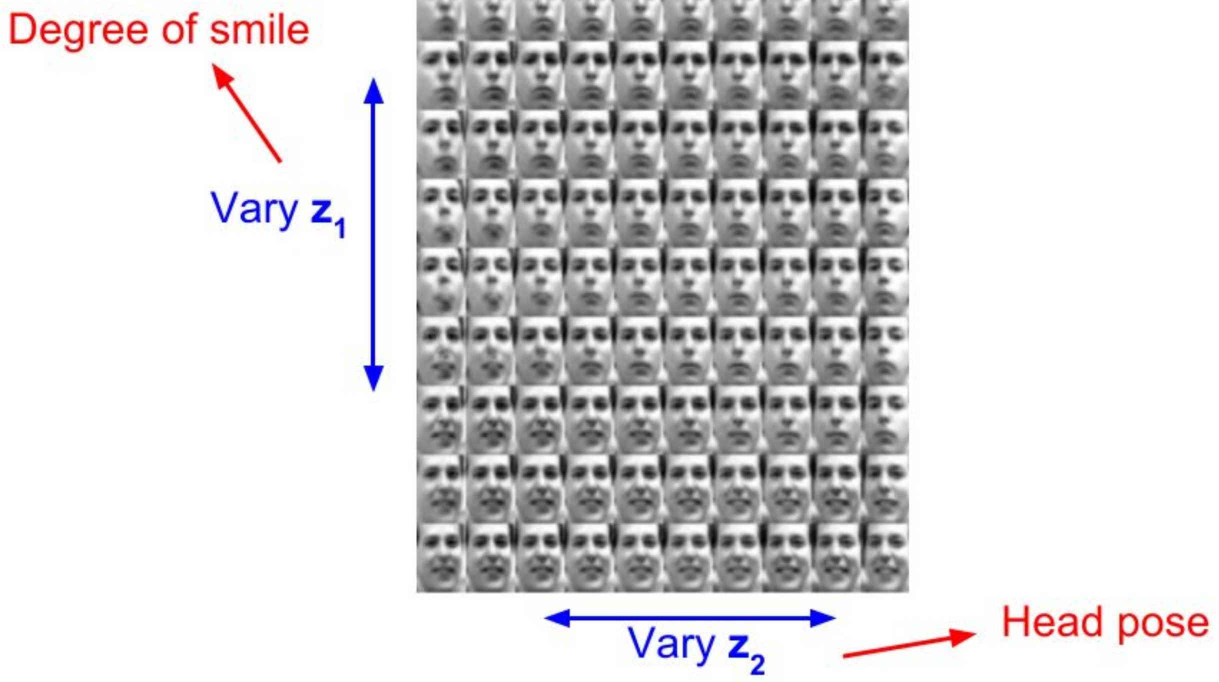
	VAE	CVAE
Prior	$p_\theta(z)$	$p_\theta(z\|y)$
Likelihood	$p_\theta(x\|z)$	$p_\theta(x\|z,y)$
Posterior	$p_\theta(z\|x)$	$p_\theta(z\|x,y)$


$\mathbf{x}$
Probabilistic Encoder
$q_\phi(\mathbf{z}|\mathbf{x})$
Mean
$\boldsymbol{\mu}$
$\boldsymbol{\sigma}$
Std. dev
$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$
$\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I})$
Sampled latent vector
$\mathbf{z}$
Probabilistic Decoder
$p_\theta(\mathbf{x}|\mathbf{z})$
$\mathbf{x}'$
$\mathbf{y}$

Variational Autoencoders: Generating Data

Diagonal prior on **z**
=> independent
latent variables

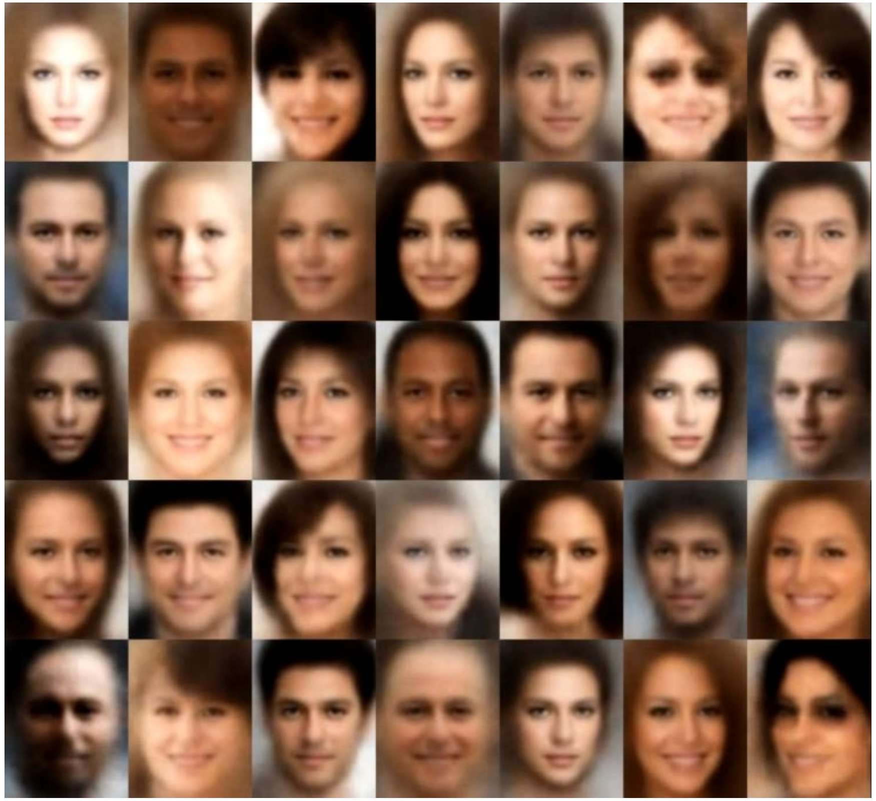
Different
dimensions of **z**
encode
interpretable factors
of variation



Slide credit: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf

Variational Autoencoders: Generating Data

- The generated images are blurry.
- How to improve:
  - More complex architectures and losses,
  - Stack of VAEs,
  - pyramid VAE



Generative Adversarial Network (GAN)

GAN framework

1. Train Discriminator:
   a. Freeze generator,
   b. Create a batch of mixed real and generated data,
   c. Train discriminator for K steps using cross entropy.
2. Train generator:
   a. Freeze discriminator,
   b. Create a batch of generated data and assign a **TRUE/REAL** label to them,
   c. Optimize generator with discriminator gradients computed by cross-entropy loss

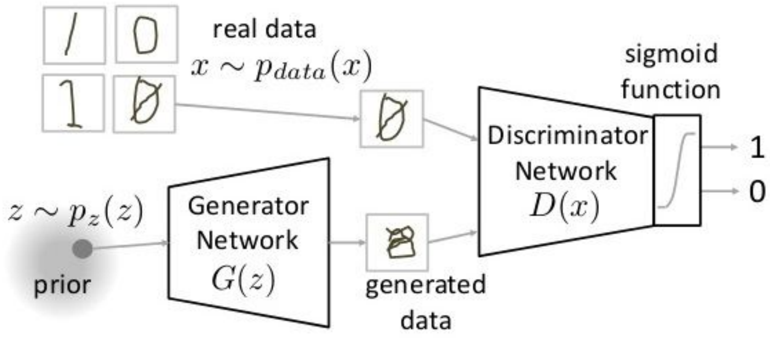


Image credit: https://www.analyticsvidhya.com/blog/2017/06/introductory-generative-adversarial-networks-gans/

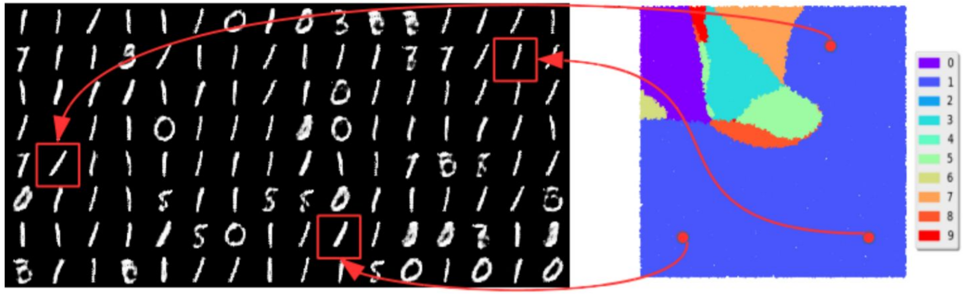
Basic GAN pros and cons

- Cons:
  - Designing an stable GAN is not trivial,
    - Generator and discriminator must have equal power,
    - You must take care how many steps to train each discriminator and generator,
    - You must fetch balanced data into the batch,
    - You must take care of gradient propagation,
    - Batch norm in generator can cause strong intra-batch correlation.

Image credit: http://www.iangoodfellow.com/slides/2016-12-04-NIPS.pdf

Basic GAN pros and cons

- Cons:
    - Not correctly designing the GAN can lead to mode collapse,

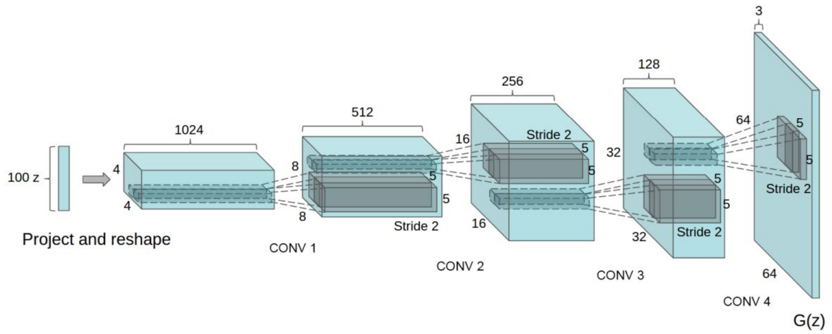


    - GAN is not aware of the context, e.g. it can generate a dog with three eyes.

Image credit: https://arxiv.org/pdf/1803.08887.pdf
http://www.iangoodfellow.com/slides/2016-12-04-NIPS.pdf

Basic GAN pros and cons

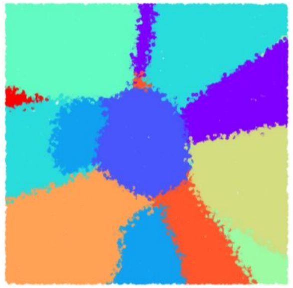
- Pros: After dealing with previous issues, GAN can generate very sharp images,
- DCGAN* is the first stable GAN architecture in image generation:
  - Strided convolution instead of pooling,
  - Using batch normalization,
  - Removing fully connected hidden layers,
  - Using ReLU activation in generator except last layer which using Tanh. Using LeakyReLU in discriminator.



* https://arxiv.org/pdf/1511.06434.pdf

Basic GAN - arithmetic operations

- You do not have control over the distribution or which dimension corresponds to an specific attribute.
- Still arithmetic operations can be applied on the learned space.



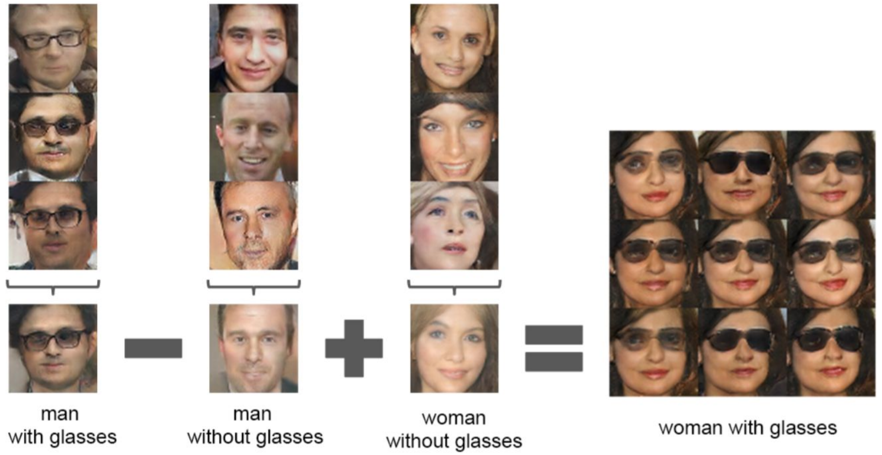
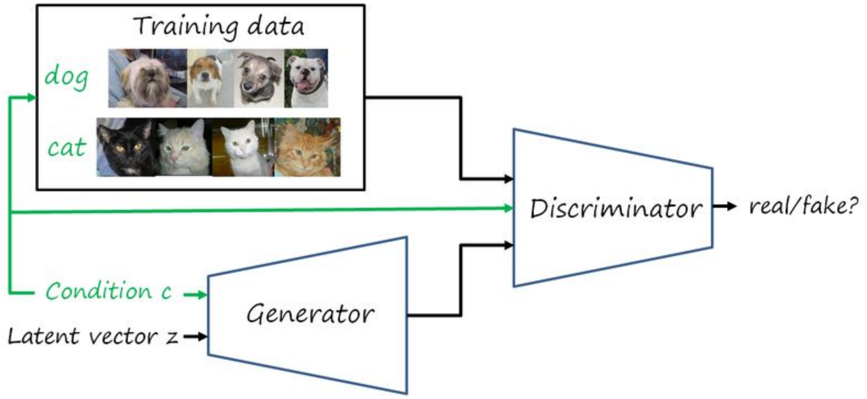


Image credit: https://arxiv.org/pdf/1803.08887.pdf
https://arxiv.org/pdf/1511.06434.pdf

Conditional GAN

- Provides control over the generation,
- Requires annotated data,
- Can be conditioned on more than one variable, including continuous values like color or pose,
- Can be designed by auxiliary classifier as well (ACGAN).
    - Discriminator and classifier can be two separate networks.



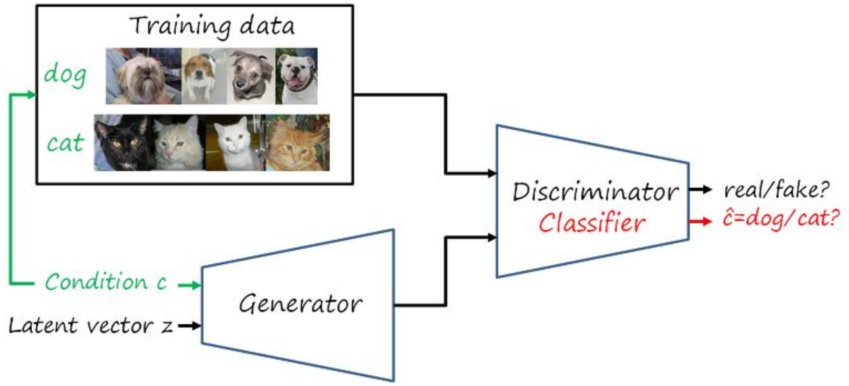
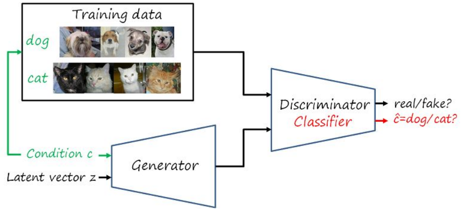


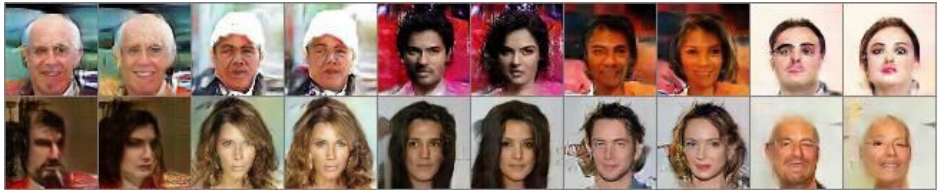
Image credit: http://www.lherranz.org/2018/08/07/imagetranslation/

GAN conditioning on gender - an example

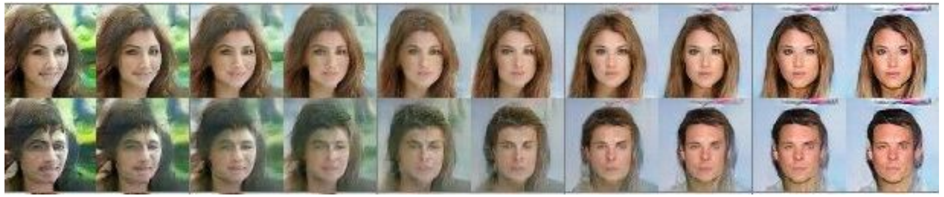
- ACGAN generates better results than CGAN and is more stable,
- You can interpolate between latent codes.



male vs female



identity transformation



identity and gender transformation

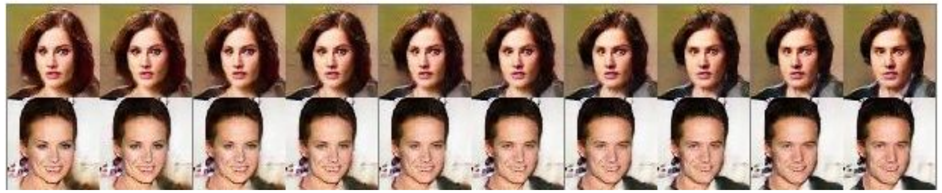


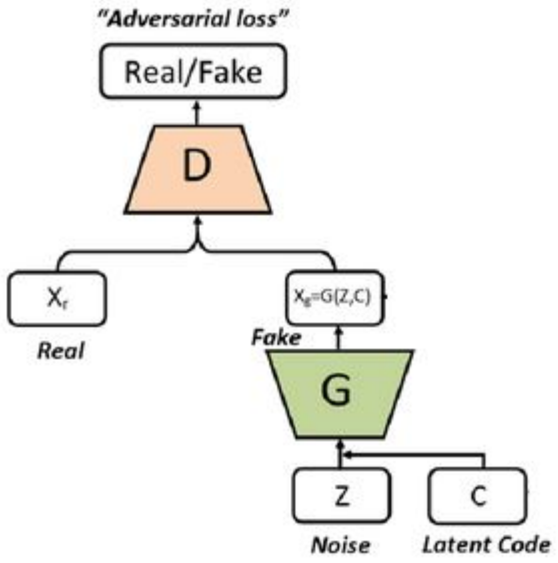
Image credit: https://zhuanlan.zhihu.com/p/44177576

Interpretable latent code

- We want to bind each dimension in the latent code with one attribute/feature in the generated image. This is called disentanglement.
- Although conditional GAN provides control over generation, still latent code is not interpretable.
    - Solution: increase conditional variables as much as possible,
    - Problem:
        - Gathering and annotating data is not trivial specially for continuous values,
        - More variables mean more challenging architectural design and perhaps less stability.
- How to design disentangled GAN?
    - Close latent codes must generate similar images.

InfoGAN*: first unsupervised disentangled GAN

- We define an additional latent input to generator,
- We want to maximize the mutual information *I(c;G(z,c))* between the latent code and generated image,
- This is hard because the posterior *P(c|x)* is unknown,



* https://papers.nips.cc/paper/2016/file/7c9d0b1f96aebd7b5eca8c3edaa19ebb-Paper.pdf

InfoGAN: first unsupervised disentangled GAN

- We define an additional latent input to generator,
- We want to maximize the mutual information *I(c;G(z,c))* between the latent code and generated image,
- This is hard because the posterior *P(c|x)* is unknown,
- Solution: add an auxiliary network Q to approximate the true distribution.

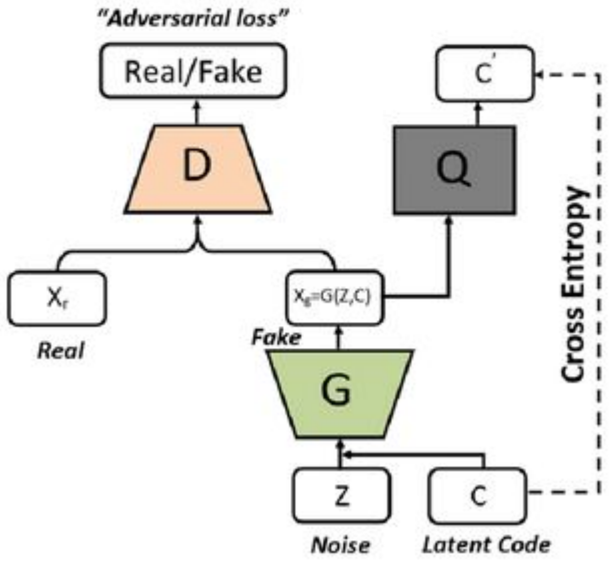
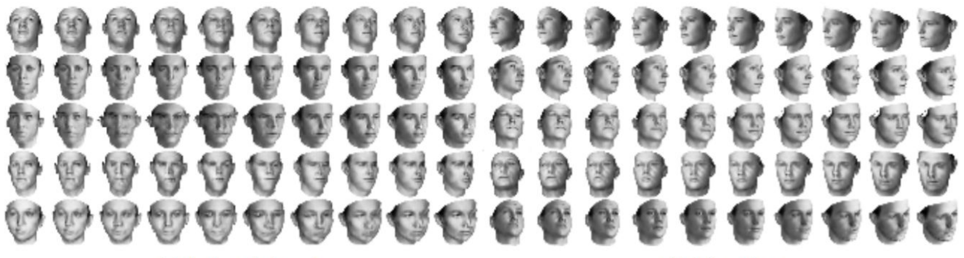


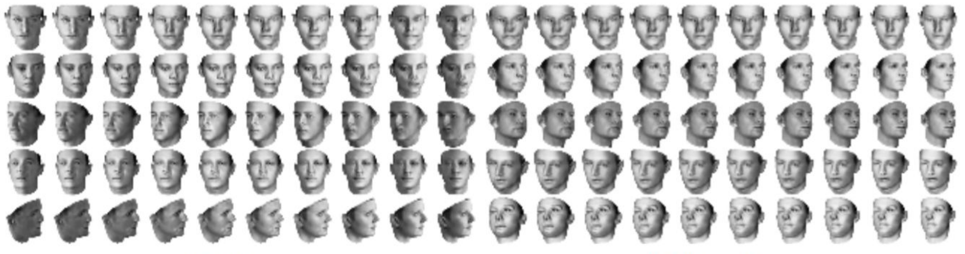
Image credit:
https://www.researchgate.net/publication/343706695_A_Survey_on_Generative_Adversarial_Networks_for_Imbalance_Problems_in_Computer_Vision_Tasks

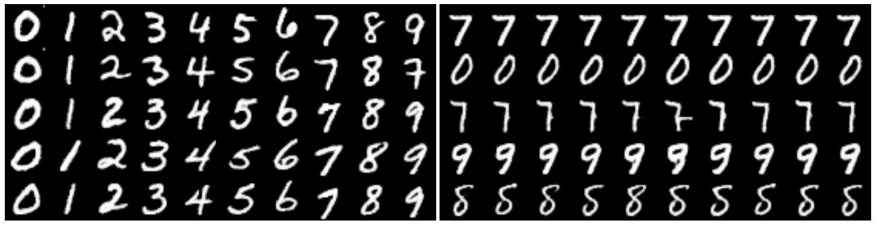
InfoGAN: categorical vs continuous latent code

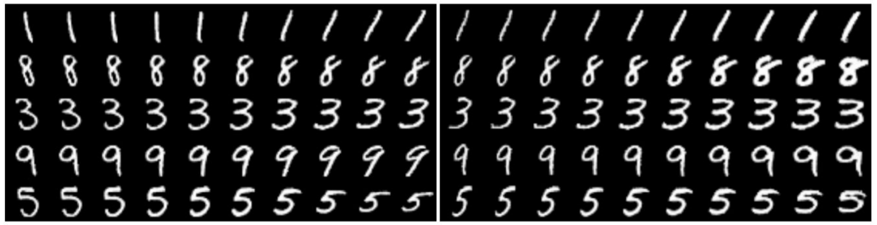
- From information theory: $I(c;G(z,c)) = H(c) - H(c|G(z,c))$
- Categorical:
  - $c'=softmax(Q(x))$,
  - $H(c) = -\Sigma \log(1/|c|) \times c$ where |c| is the c dimensionality,
  - $H(c|G(z,c)) = -\Sigma \log(c'+\varepsilon) \times c$
- Continuous (through Gaussian):
  - $\mu,\sigma = Q(x)$ where μ is the mean and σ is the standard deviation,
  - $g = (c-\mu) / (\sigma+\varepsilon)$,
  - $H(c) = -\Sigma\ 0.5 \log(2\pi) + 0.5\ c^2$,
  - $H(c|G(z,c)) = -\Sigma\ 0.5 \log(2\pi\sigma+\varepsilon) + 0.5\ g^2$

InfoGAN: example results


(a) Azimuth (pose) (b) Elevation


(c) Lighting (d) Wide or Narrow


(a) Varying c_1 on InfoGAN (Digit type) (b) Varying c_1 on regular GAN (No clear meaning)

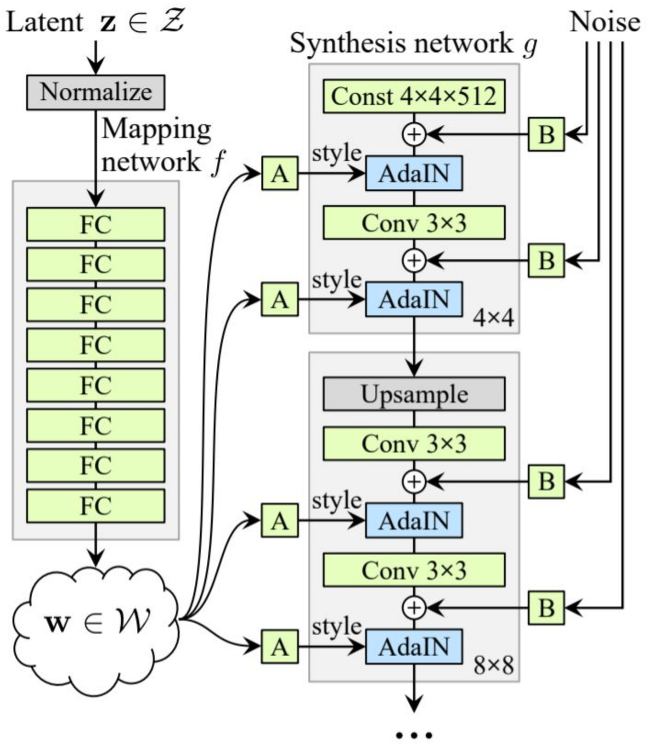

(c) Varying c_2 from -2 to 2 on InfoGAN (Rotation) (d) Varying c_3 from -2 to 2 on InfoGAN (Width)

StyleGAN*: an improved generator

- Styles are defined as **scales** and **biases** per layer,
- Latent code is mapped to the styles through a mapping network *f* and affine layer *A*, ***y = (y_s , y_b) = A(w) = A(f(z))***
- A constant learned input tensor is combined with the noise and transformed through styles,
- Noise is scaled by learned per layer *B,*
- Styles contribute in the generation through adaptive instance normalization layers,

$$\mathrm{AdaIN}(\mathbf{x}_i, \mathbf{y}) = \mathbf{y}_{s,i} \frac{\mathbf{x}_i - \mu(\mathbf{x}_i)}{\sigma(\mathbf{x}_i)} + \mathbf{y}_{b,i},$$

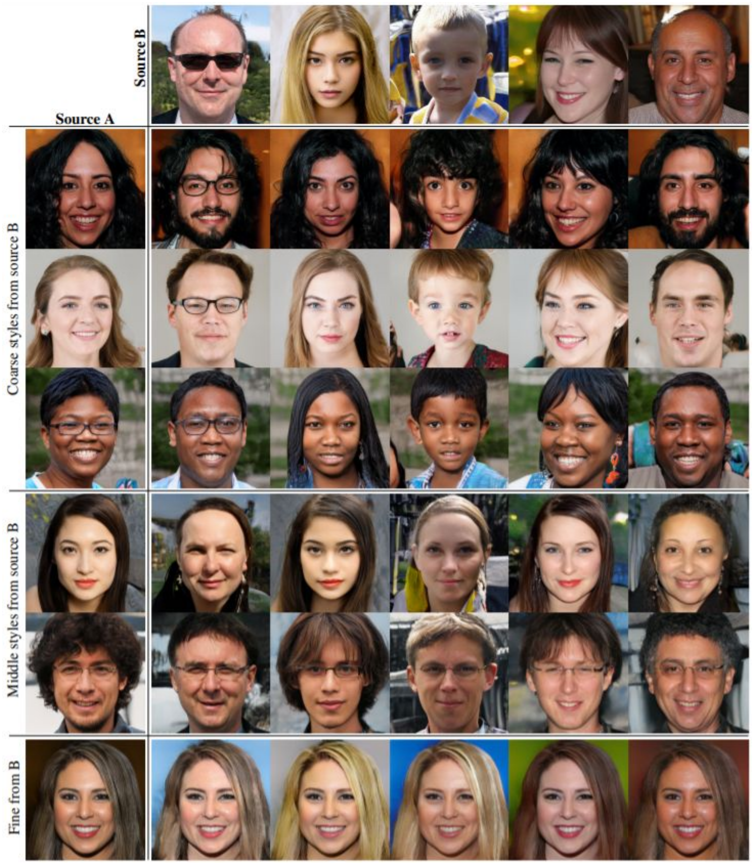
- Style mixing regularization "prevents the network from assuming that adjacent styles are correlated".



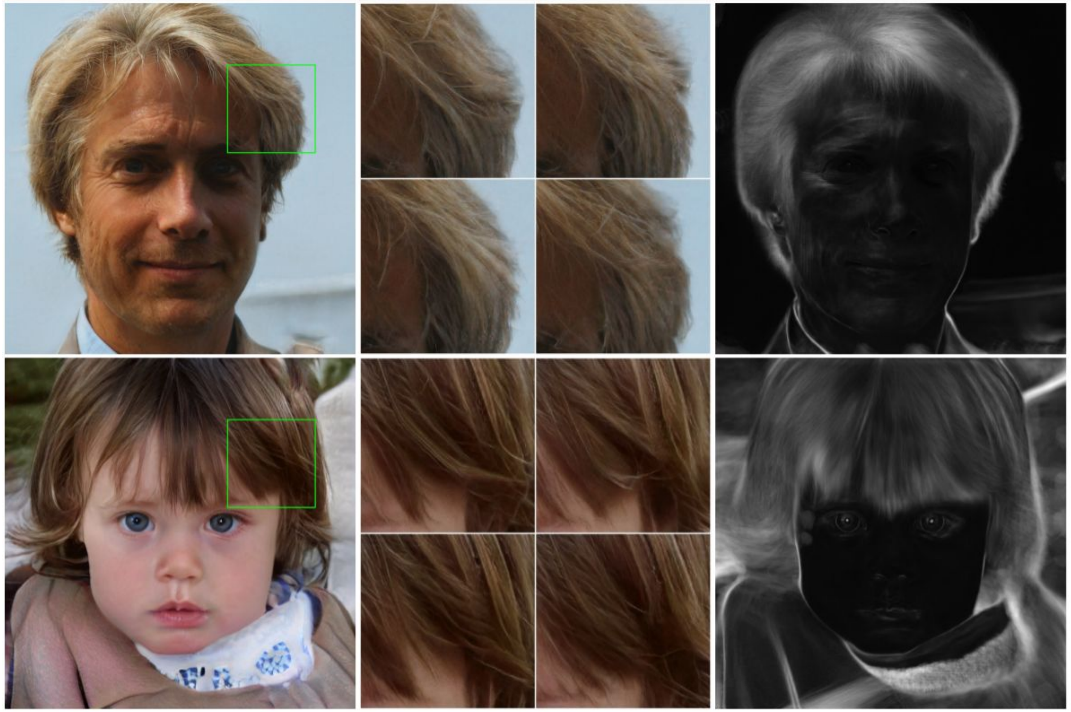
* https://arxiv.org/pdf/1812.04948.pdf

StyleGAN: transferring styles

- First row (source B) and column (source A) are generated images,
- A subset of styles are copied from the source B to source A,
- Global styles like age, gender, pose and glasses appear in the earlier (coarse) layers,
- Local styles like hair or face color appear in the last (fine) layers.



StyleGAN: examples of stochastic variation



(a) Generated image (b) Stochastic variation (c) Standard deviation

Paired image-to-image translation

- There is pixel to pixel correspondences in the paired input-output,
- Do we need GAN to solve this problem?
  - Easy problems: supervised deep learning can handle it nicely.
  - Difficult problems: supervised approaches will fail:
    - There are multiple valid values for each pixel in the output ===> this means converging to average and blurry images

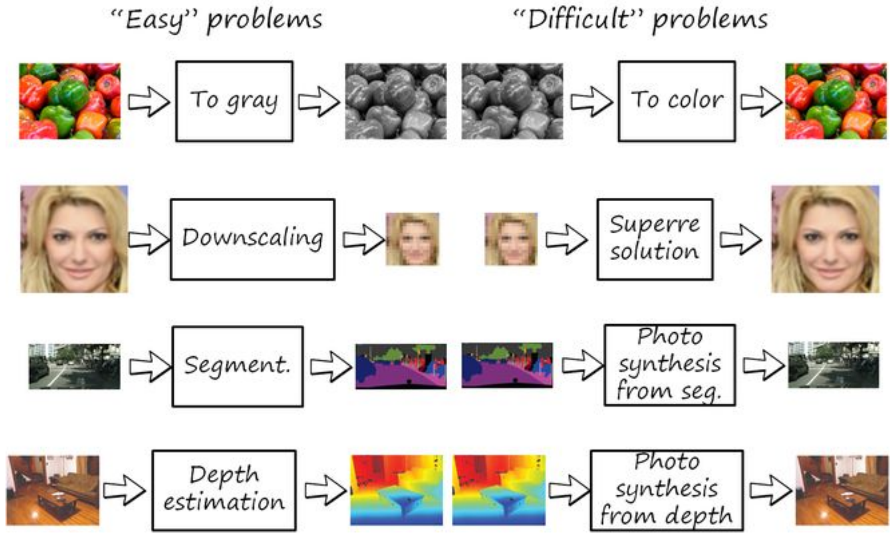
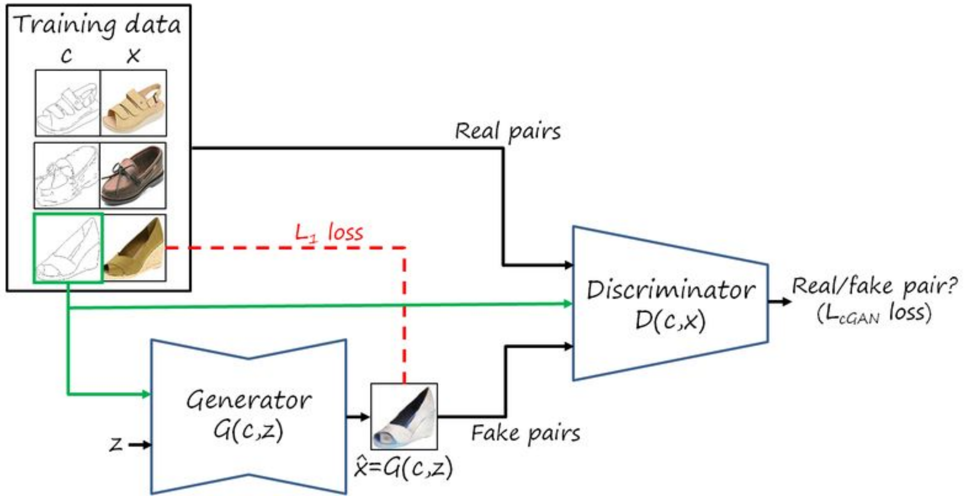


Image credit: http://www.lherranz.org/2018/08/07/imagetranslation/

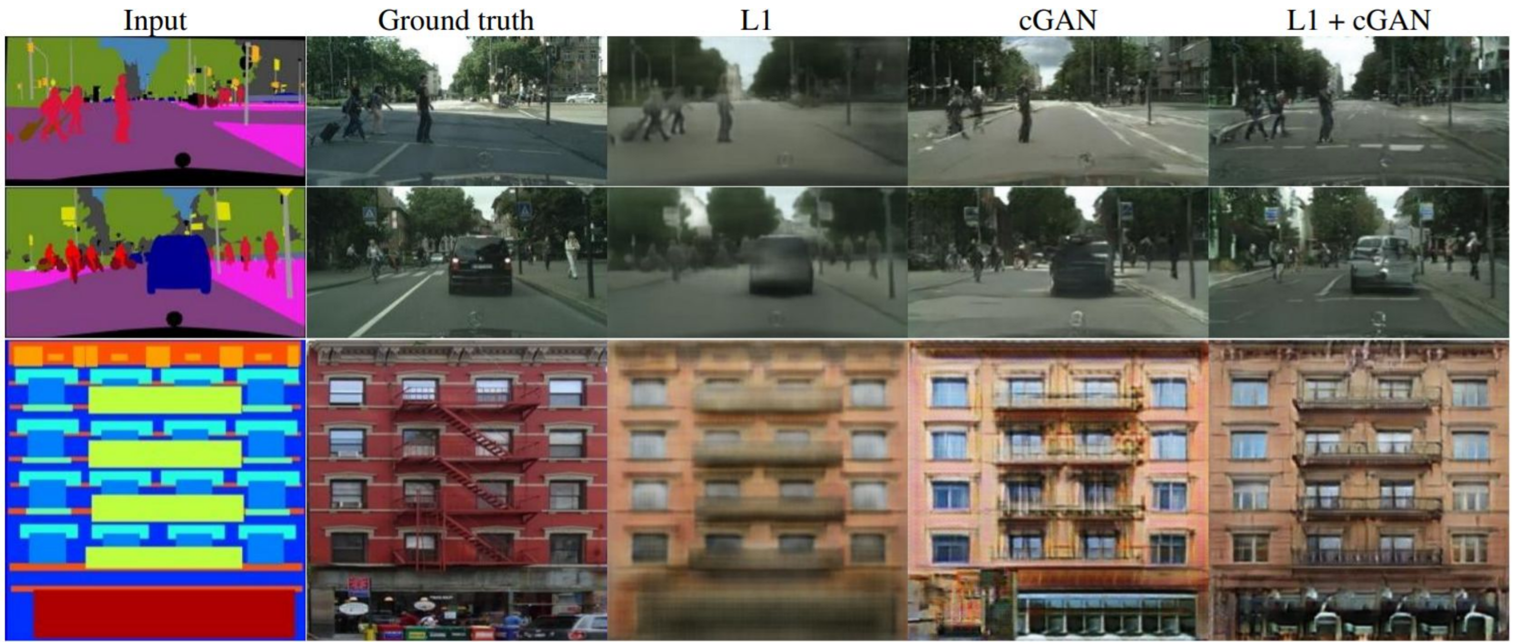
Paired image-to-image translation - Pix2Pix* architecture

- Observations:
  a. Generator has a U-NET structure,
  b. Generator is trained with an additional L1 loss which captures low frequency details,
  c. Discriminator is applied on the NxN image patches (PatchGAN) to enforce learning high frequency details,
  d. Without z, the net could still learn a mapping from x to y, but would produce deterministic outputs. Including z was not effective and the generator simply learned to ignore the noise. For the final models, the noise was only provided in the form of dropout, applied on several layers of the generator at both training and test time.

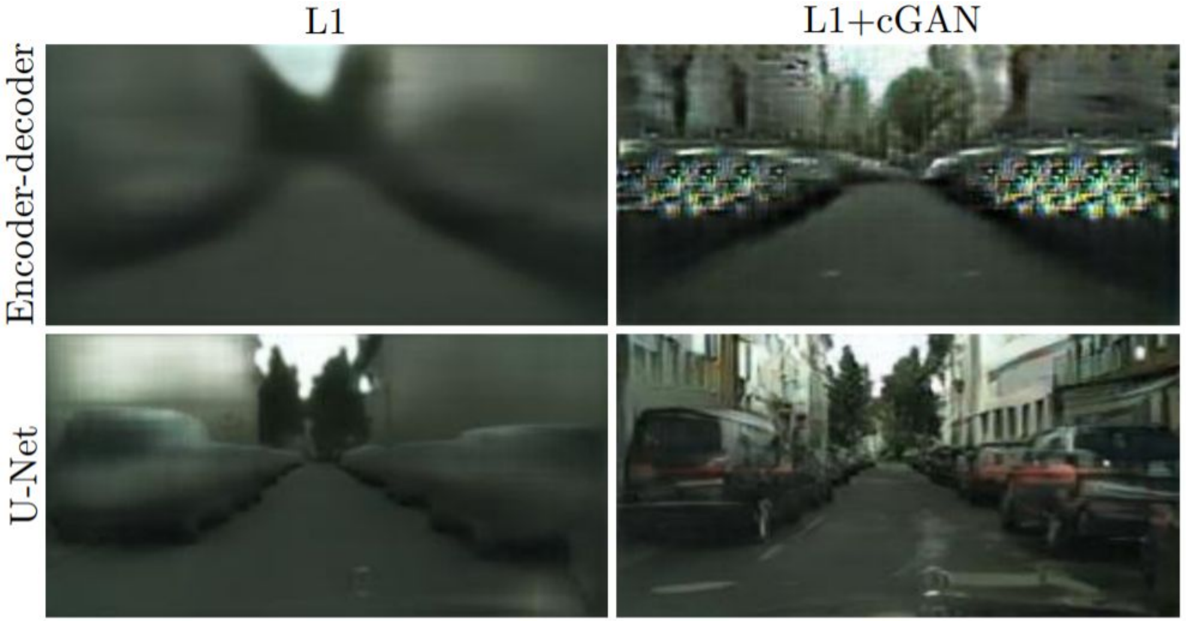


* https://arxiv.org/pdf/1611.07004.pdf
Image credit: http://www.lherranz.org/2018/08/07/imagetranslation/

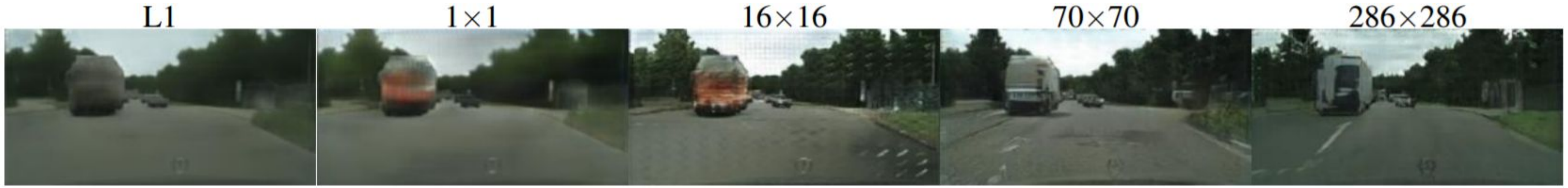
Pix2Pix - some results: the effect of loss



Pix2Pix - some results: U-NET vs encoder-decoder



Pix2Pix - some results: PatchGAN



Unpaired image-to-image translation

- There are two sets of data, that is, two different domains, e.g. horses and zebras,
- There is no instant matching between source and target domains,
- Source and target domains may have conceptual overlapping, e.g. real images and paintings,

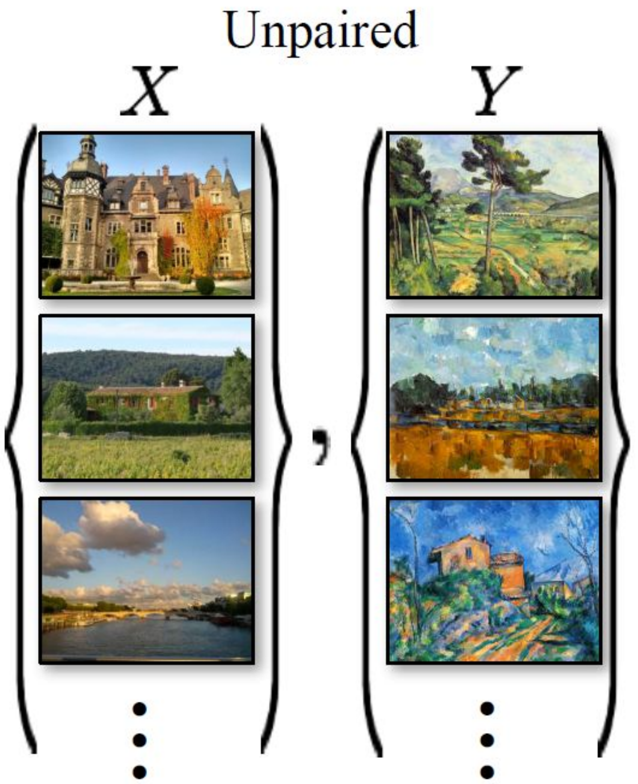
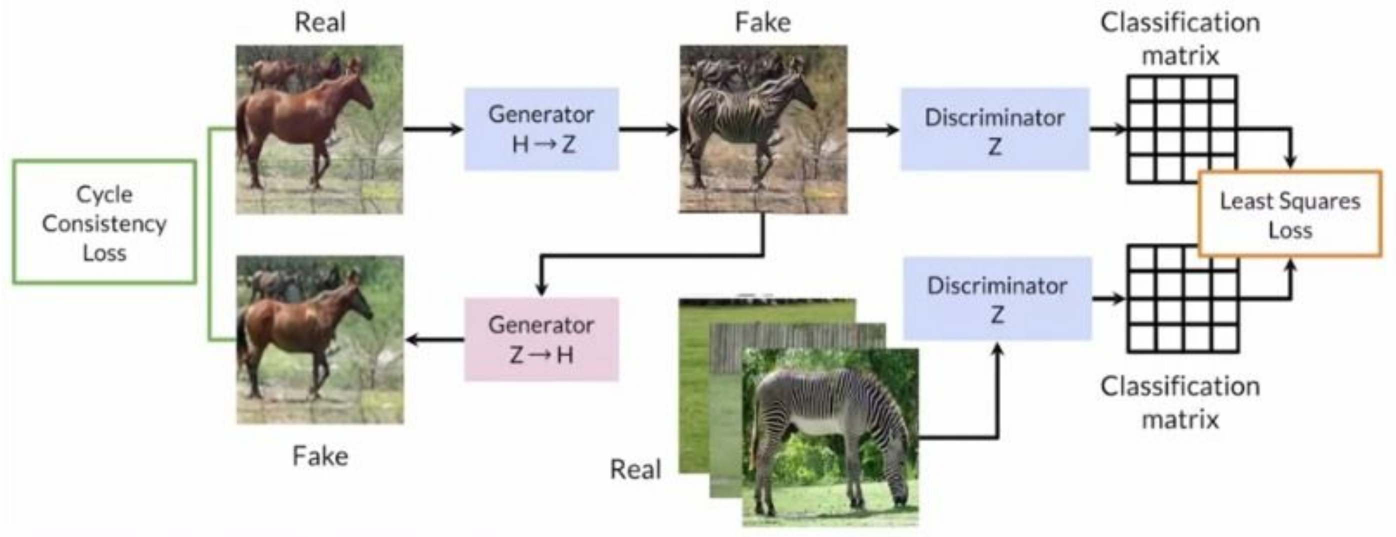
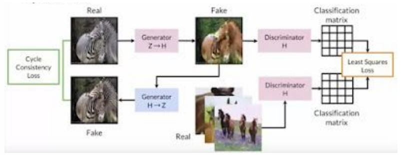


Image credit: https://arxiv.org/pdf/1703.10593.pdf

Unpaired image-to-image translation - CycleGAN*

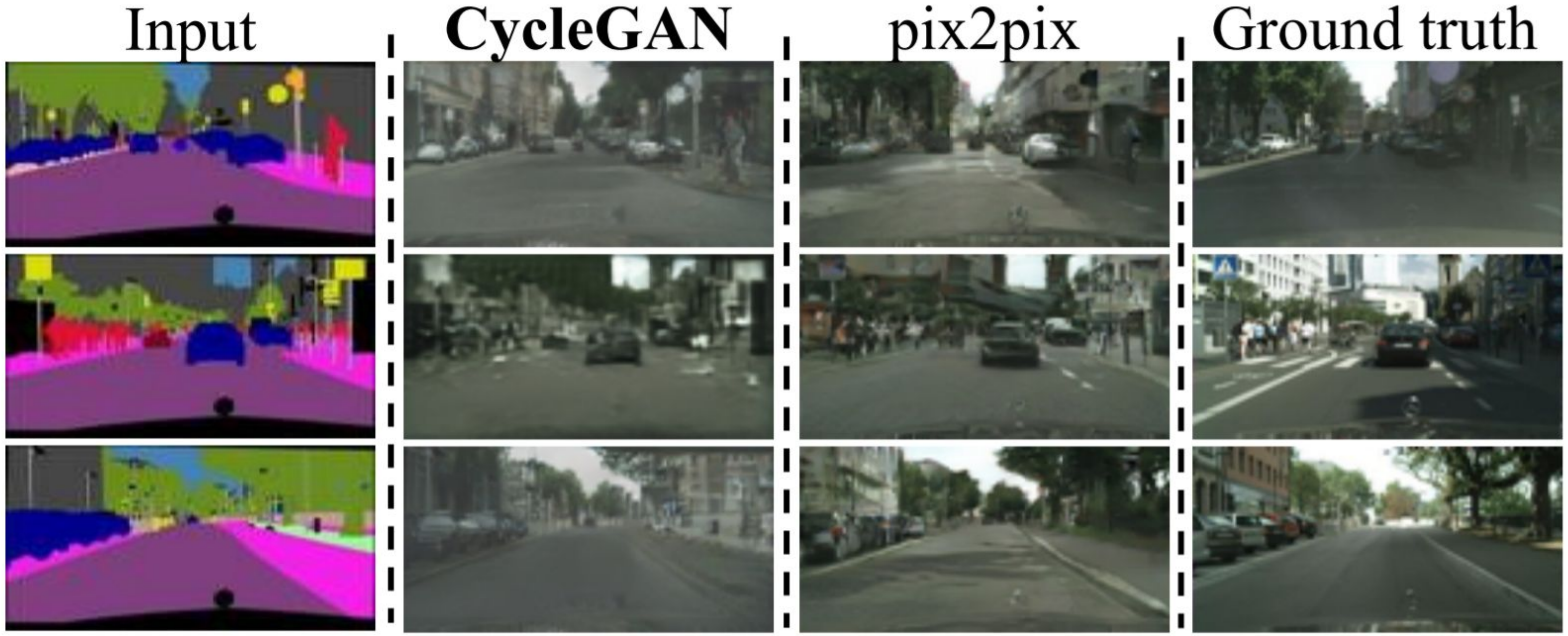
- CycleGAN is a mutual source-to-target and vice-versa generation plus a consistency loss, i.e. regenerated source must look like the input source. Training: 1) S→T→S and 2) T→S→T.
- Generator: three convolutions, several residual blocks, two fractionally-strided convolutions with stride 1/2, and one convolution that maps features to RGB.
- Discriminator: 70x70 PatchGAN



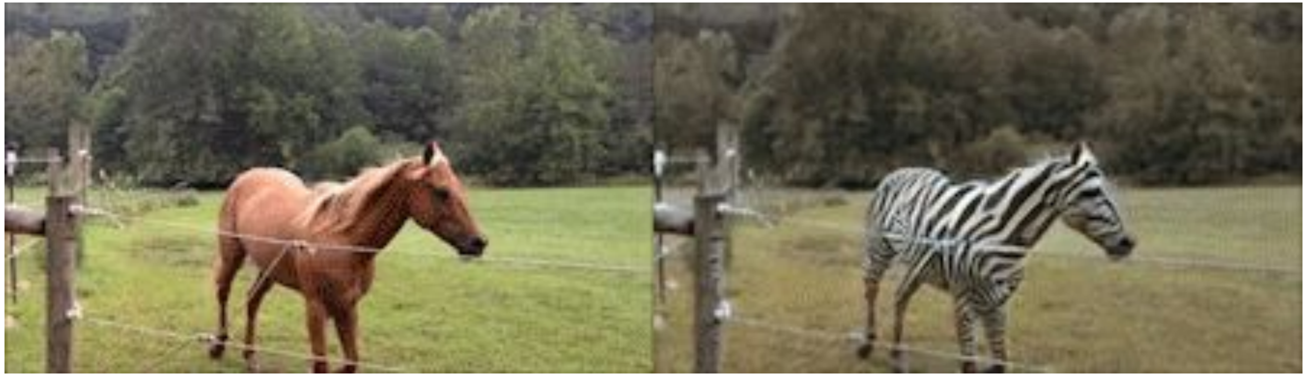


* https://arxiv.org/pdf/1703.10593.pdf
Image credit: https://www.haikutechcenter.com/2020/11/cyclegan-gan-architecture-for-learning.html

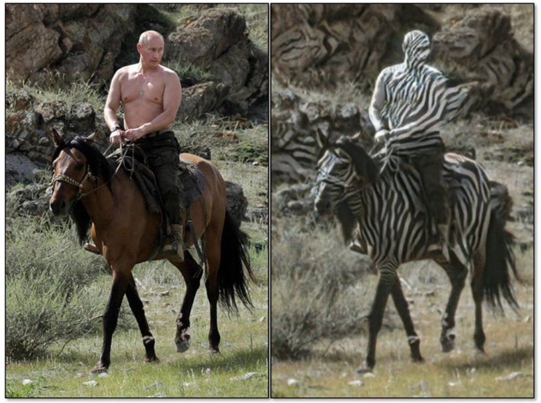
CycleGAN - results on paired data



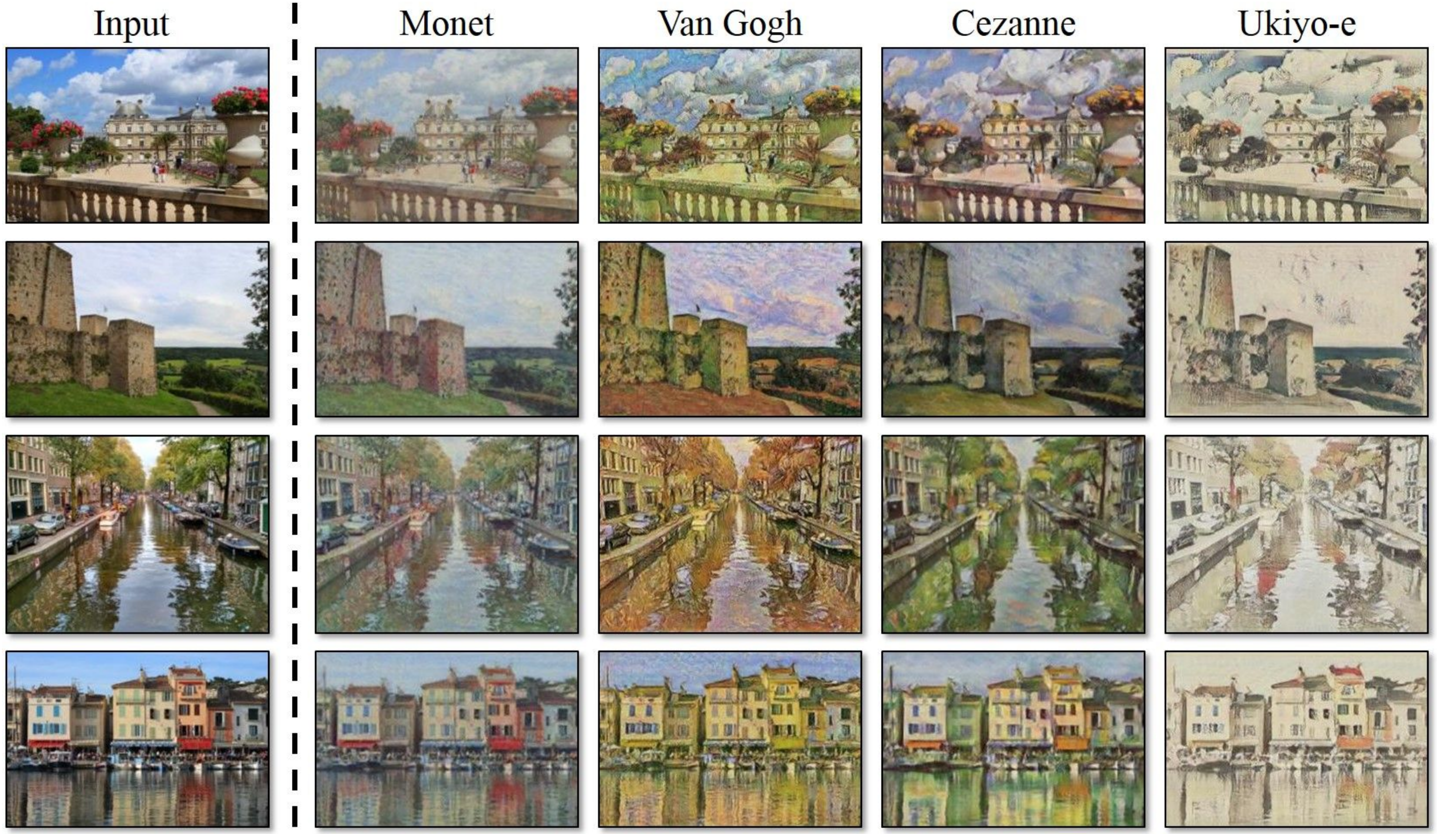
CycleGAN - results on unpaired data



A failure case →



CycleGAN - more results (https://junyanz.github.io/CycleGAN/)



Applications

- Blending and inpainting,
- Style transfer,
    - Face translation: aging, adding attributes, emotion transferring, etc
    - Cloth translation,
    - Sketch painting,
    - Art transferring,
    - etc
- View synthesis: scene and camera geometry, lighting, etc
- Super resolution,