Generative neural networks

Sigmund Rolfsjord

Practical

INF5860 - searching for teaching assistants (spring 2019)

https://www.uio.no/studier/emner/matnat/ifi/INF5860/v18/

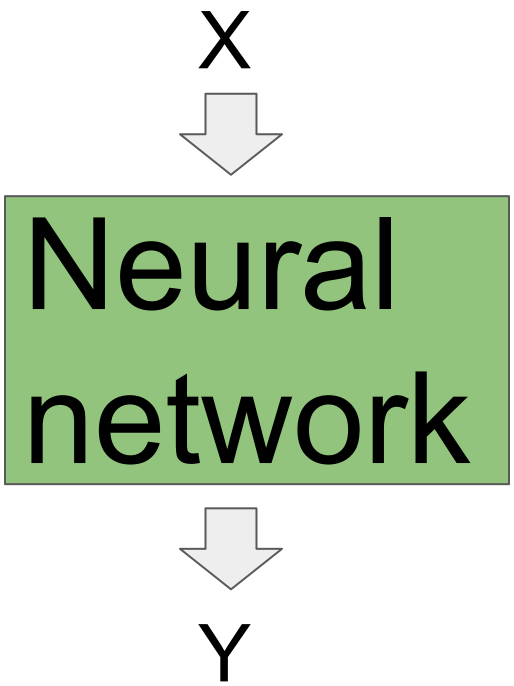
Overview of wasserstein GAN:

https://medium.com/@jonathan_hui/gan-wasserstein-gan-wgan-gp-6a1a2aa1b490

Generating data with deep networks

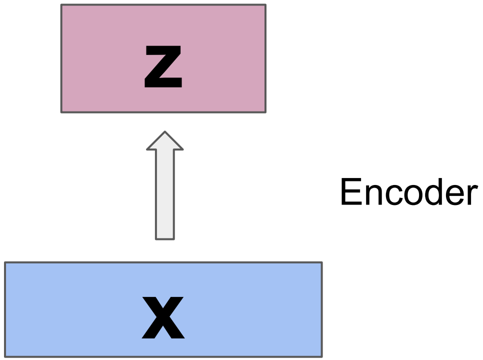
We are already doing it.

- How to make it “look” realistic
- What loss function can we optimize



Autoencoders

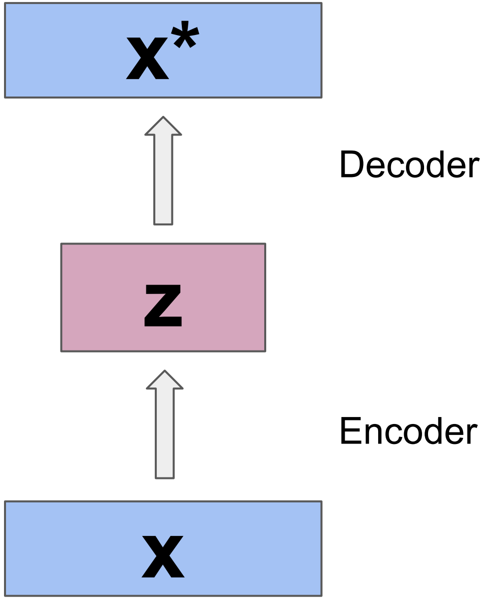
- A neural network transforming the input
- Often into a smaller dimension



Autoencoders

- A neural network transforming the input
- Often into a smaller dimension
- Then a decoder network reconstructs the input

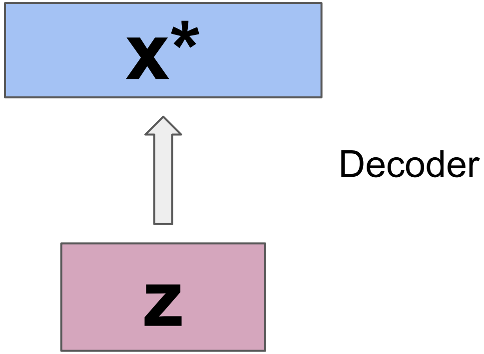
Old idea [Modular Learning in Neural Networks 1987, Ballard]



Autoencoders - Generating images

- A neural network transforming the input
- Often into a smaller dimension
- Then a decoder network reconstructs the input

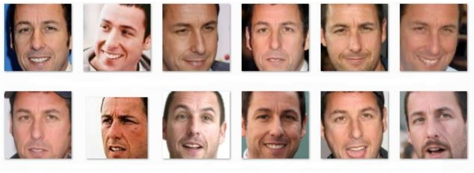
- With different values of Z, you can generate new images

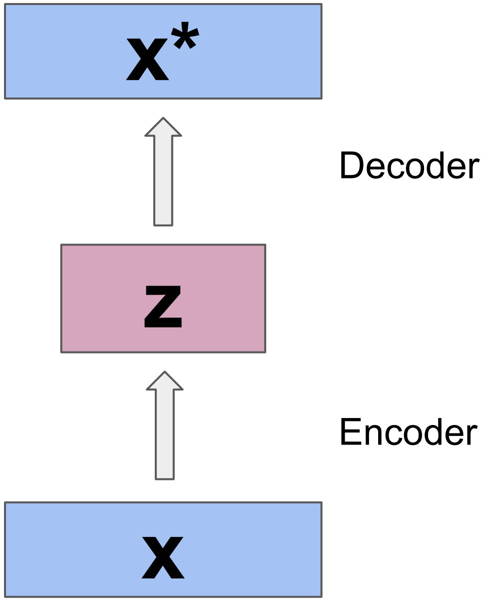


Autoencoders

- A neural network transforming the input
- Often into a smaller dimension
- Then a decoder network reconstructs the input
- Restrictions are put on **z** either through loss functions, or **size**

- Often used with convolutional architectures for images

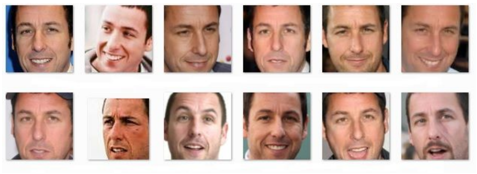


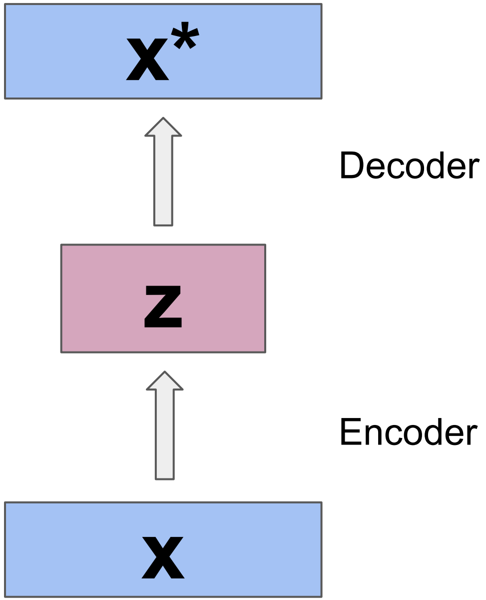


Autoencoders

- Restrictions are put on **z** either through loss functions, or **size**
- Often minimizing l2 loss:

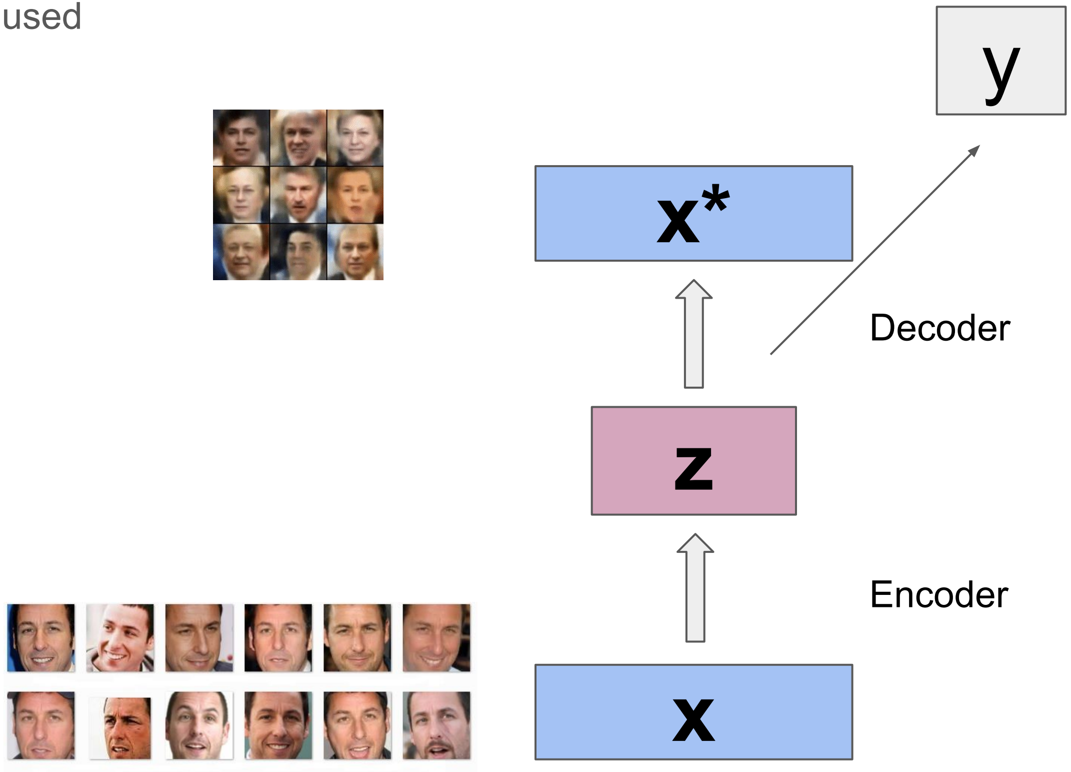
$$L(x) = (x - x*)^2$$



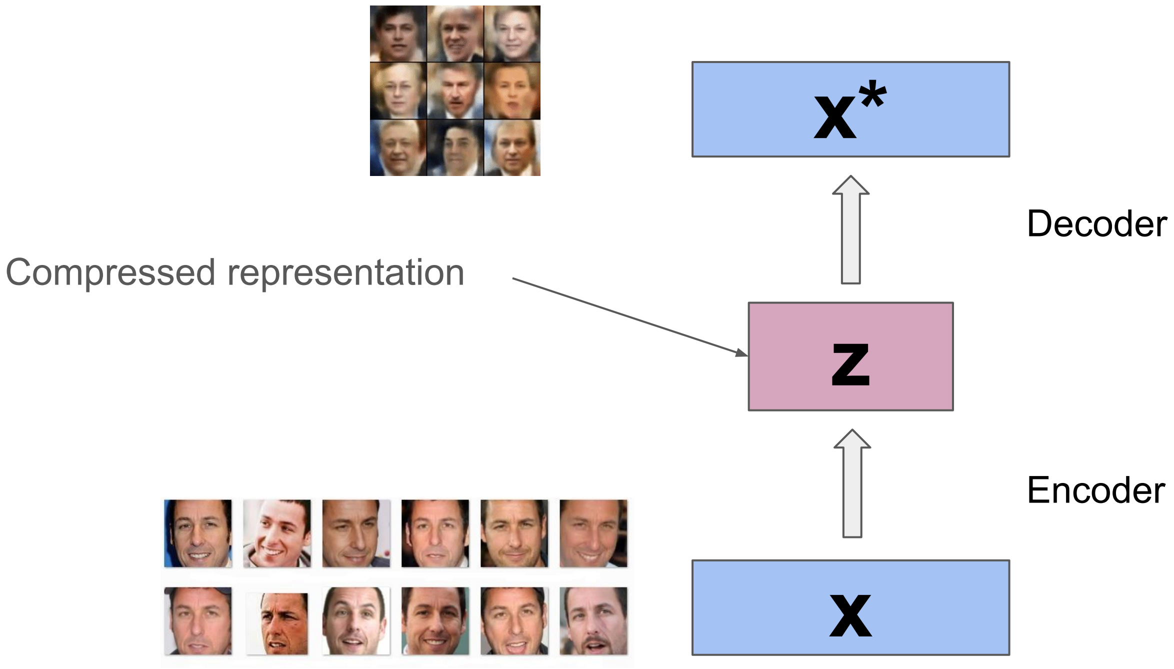


Autoencoders - Semi-supervised learning

- The encoded feature is sometimes used as features for supervised-learning



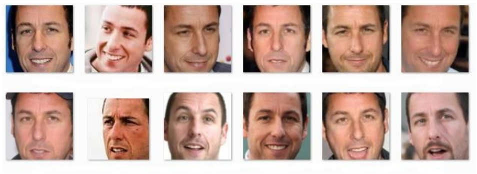
Autoencoders - Compressed representation

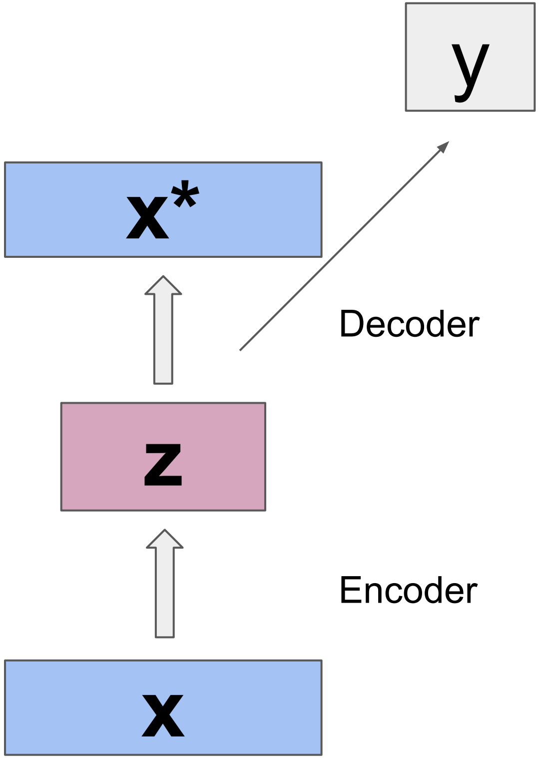


Autoencoders - Some challenges

You don't have control over the features learned:

- Even though the features compress the data, they may not be good for categorization.





Autoencoders - Some challenges

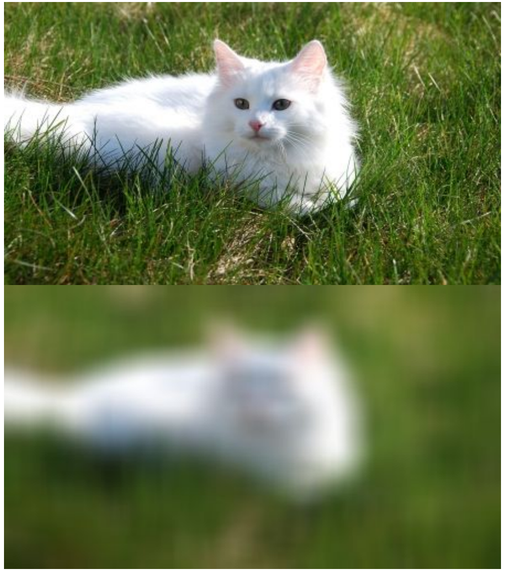
Pixel wise difference may not be relevant.

- Pixel wise a black cat on a red carpet, can be opposite from a white cat on green grass

Autoencoders - Some challenges

Pixel wise difference may not be relevant.

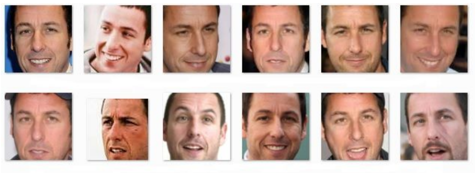
- Pixel wise a black cat on a red carpet, can be opposite from a white cat on green grass
- The image is compressed through blurring, not concept abstraction

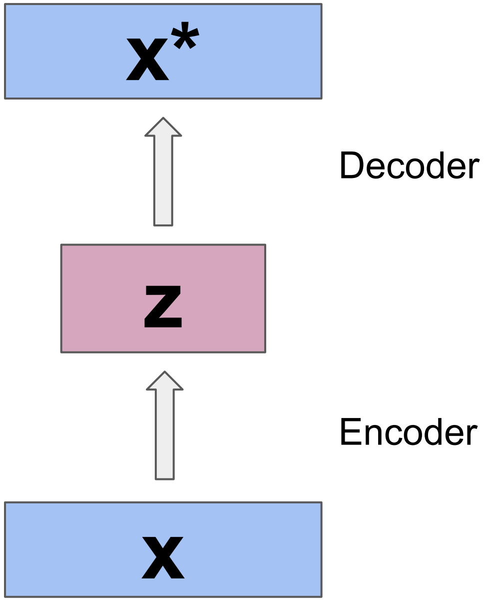


Autoencoders - Some challenges

You don’t have control over the features learned:

- Even though the features compress the data, they may not be good for categorization.
- Where should you sample Z?
    - Values of Z may only give reasonable results in some locations

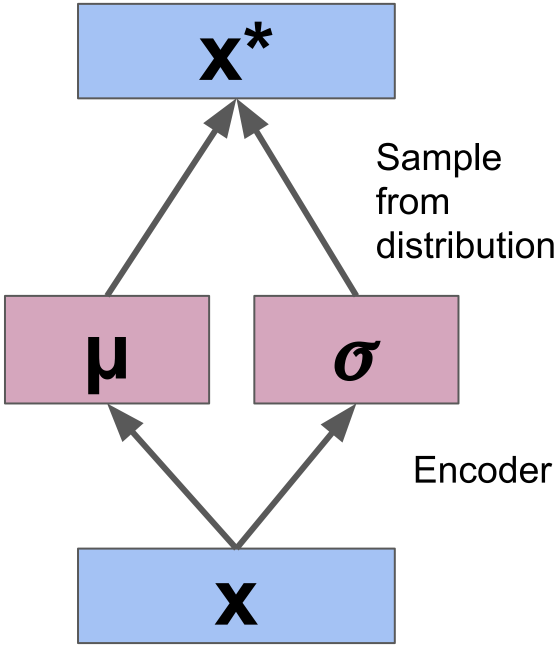




Variational Autoencoder

Find the data distribution instead of reconstructing simple images

- Assume some prior distribution
- Use the encoder to estimate distribution parameters
- Sample a **z** from the distribution and try to reconstruct

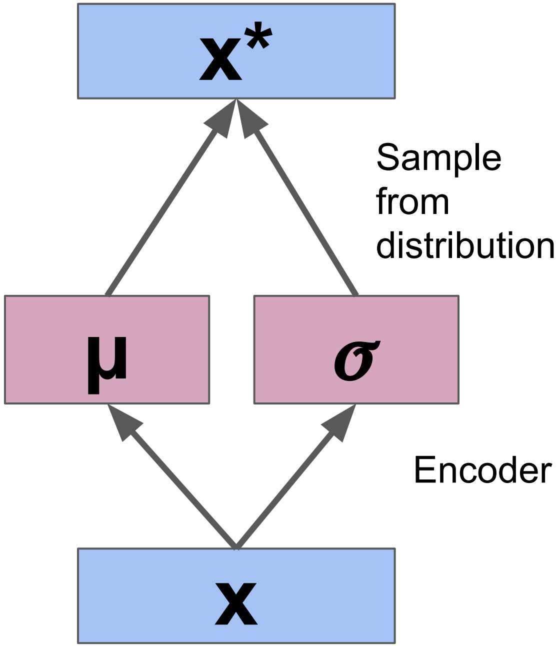


Variational Autoencoder - loss function

Find the data distribution instead of reconstructing simple images

Often

- L2 loss between images
- KL-divergence between estimated distribution and prior distribution
    - Typically unit gaussian



$$\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))$$

Variational Autoencoder - loss function

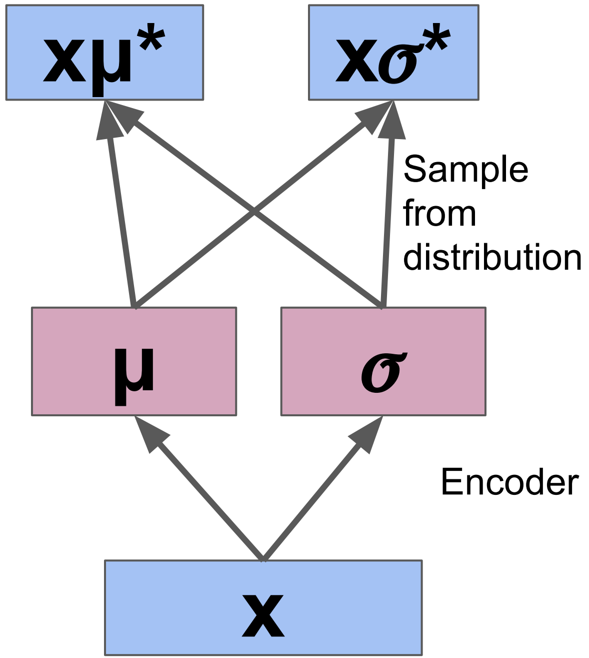
Find the data distribution instead of reconstructing simple images

Often

- L2 loss between images
- KL-divergence between estimated distribution and prior distribution
    - Typically unit gaussian

Alternatively:

- Decode image distribution
- Loss is then the log likelyhood of the inputed image, given the outputted distribution.

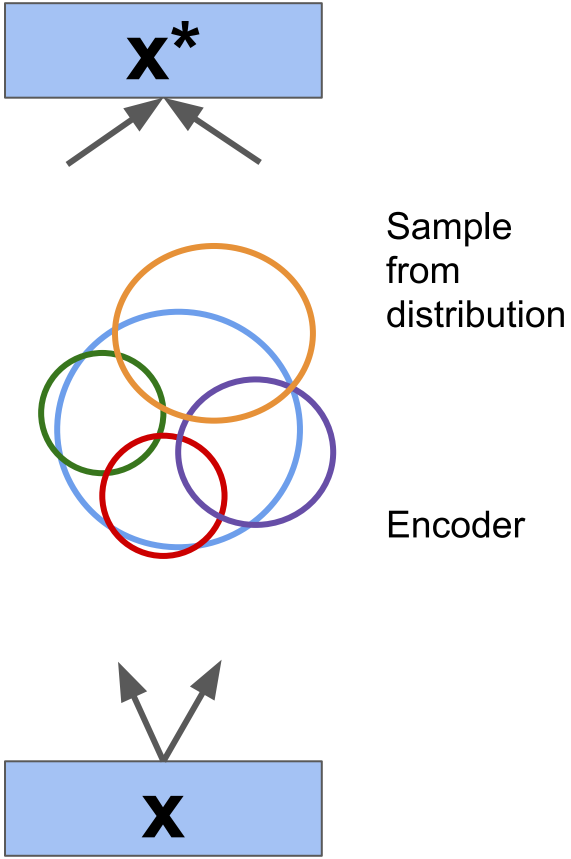


$$\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))$$

Variational Autoencoder - loss function

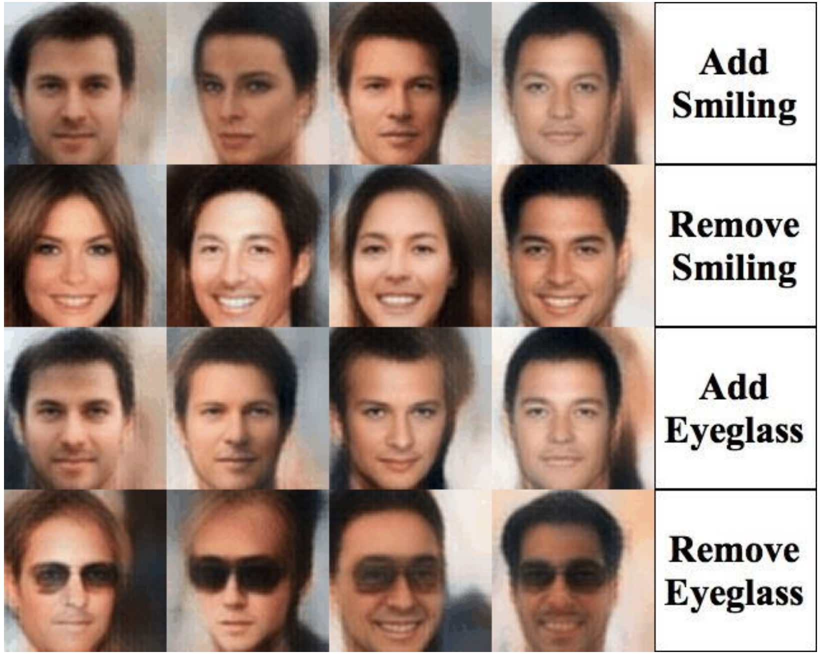
Find the data distribution instead of reconstructing simple images

- Force similar data into overlapping distribution
- To really separate some data, you need small variance
    - You pay a cost for lowering variance
    - Have to be weighted by gain in reconstruction
- You train the network to reconstruct “any” input
- Interpolating between samples should give viable results



Variational Autoencoder

Interpolating between samples should give viable results

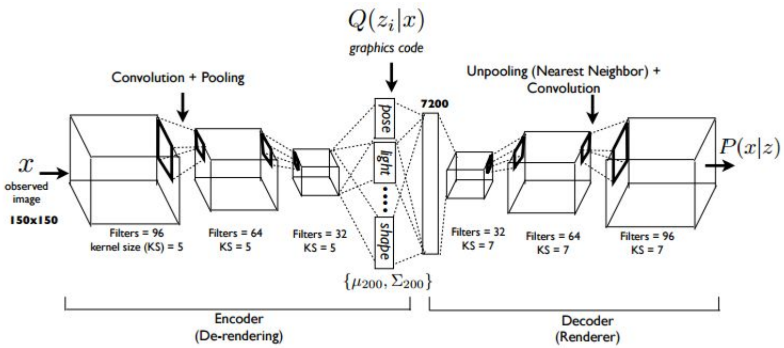


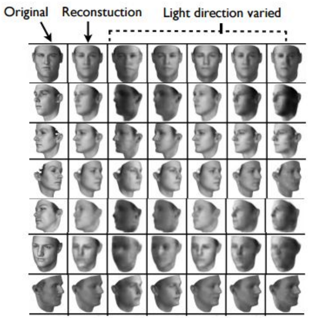
[Deep Feature Consistent Variational Autoencoder]

Variational Autoencoder - forcing sematics

Interpolating between samples should give viable results

We can insert specific information to do semi-supervised learning, and force the embedding to be what we want.

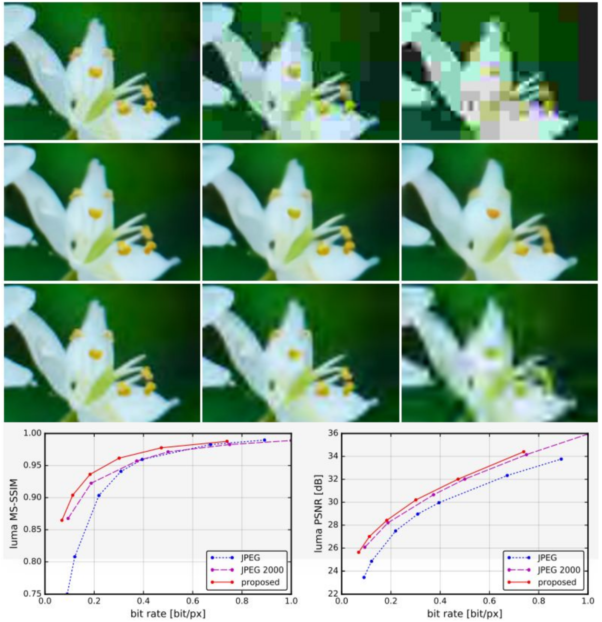




[Deep Convolutional Inverse Graphics Network]()

Variational Autoencoder - compression

Perhaps not surprisingly, autoencoders work well for image compression.

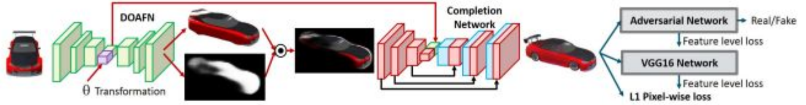


[End-to-end Optimized Image Compression]()

Variational Autoencoder - forcing sematics

Interpolating between samples should give viable results

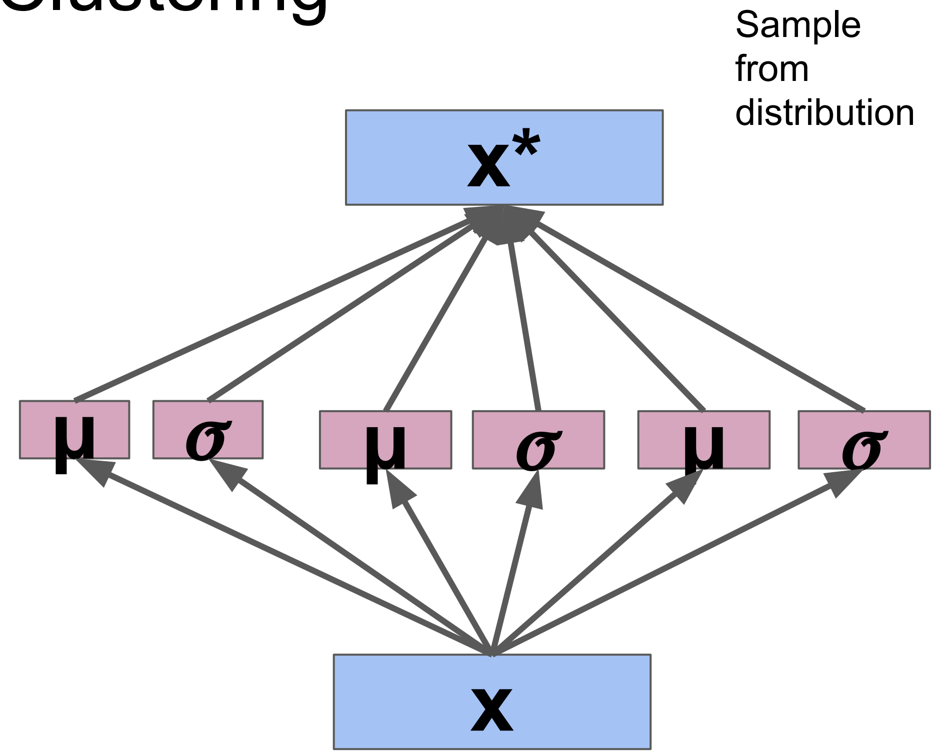
We can insert specific information to do semi-supervised learning, and force the embedding to be what we want.



[Transformation-Grounded Image Generation Network for Novel 3D View Synthesis]()

Variational Autoencoder - Clustering

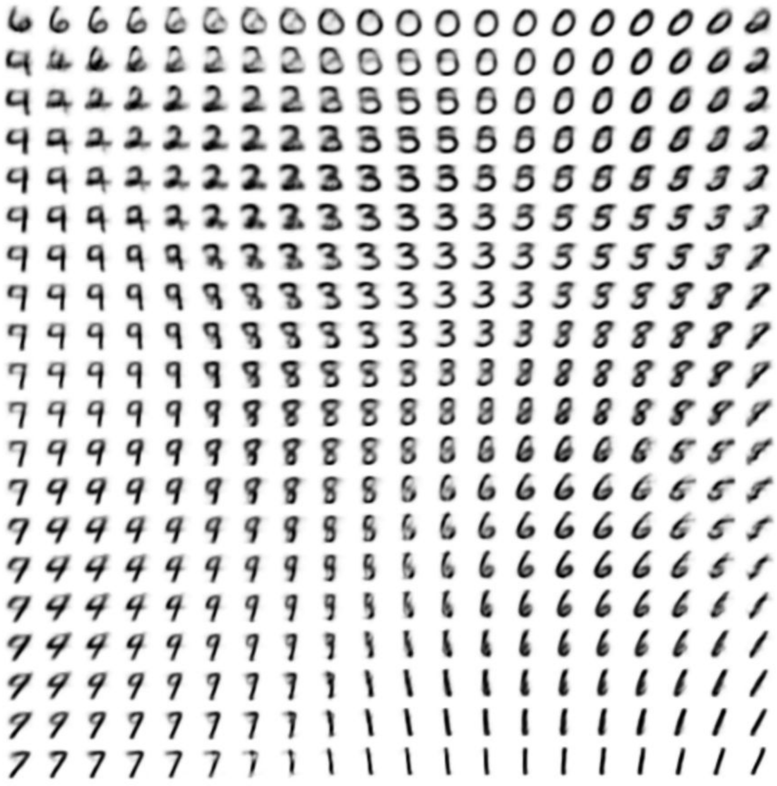
- One option is to use k-means clustering on the reduced dimension
- An alternative is to make your prior distribution multimodal
- So your encoder has to put the encoding close to one of the K predefined modes.



[DEEP UNSUPERVISED CLUSTERING WITH GAUSSIAN MIXTURE VARIATIONAL AUTOENCODERS]()

Variational Autoencoder - modelling the data

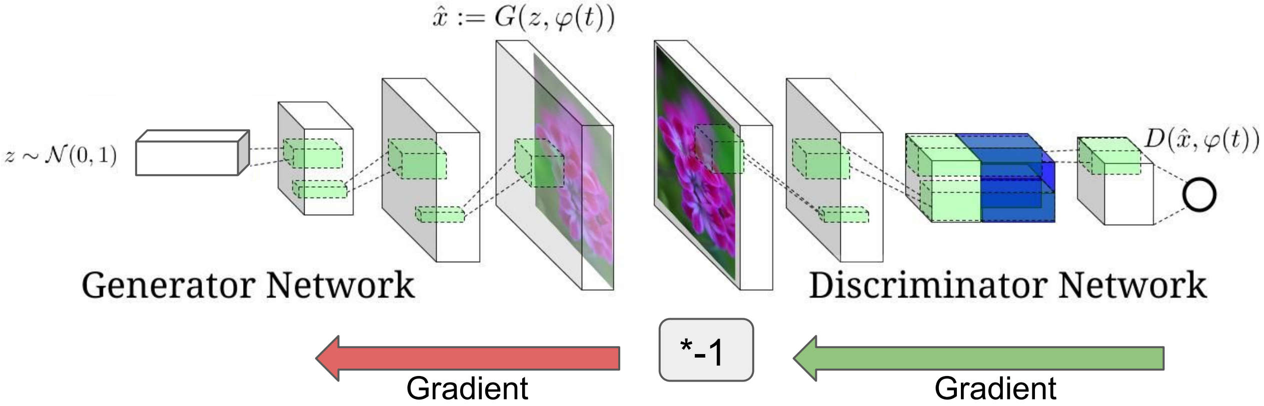
- Can be good at modelling how the data varies
- Generated results are often some sort of averaged images
    - Works well if averinging photos works



Generative adversarial networks (GAN)

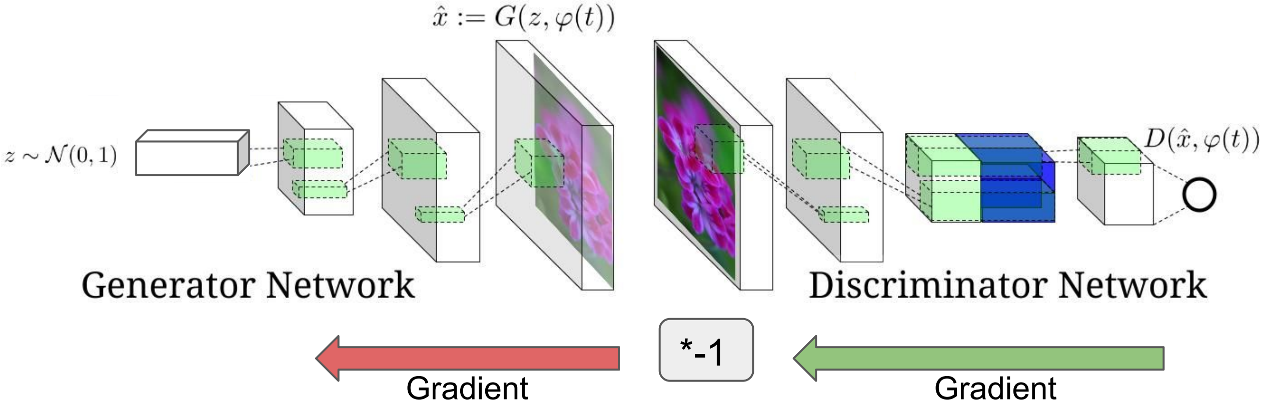
Generating images

- Two competing networks in one
- One Generator (G)
- One Discriminator (D)
- Generator knows how to change in order to better fool the discriminator



Generating images

- Input of generator network is a random vector
- Sampled with some strategy



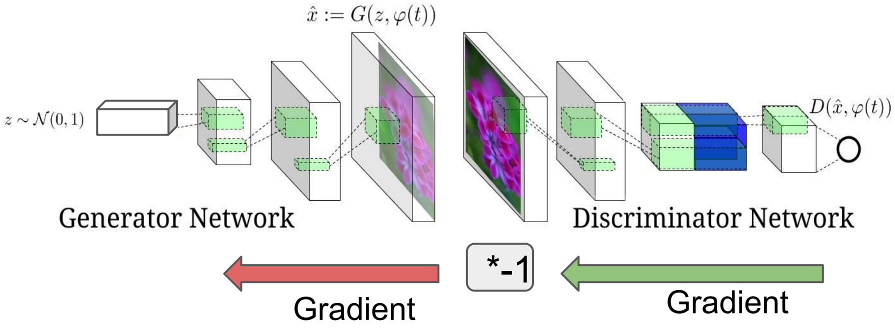
Generating images

Discriminator maximizes:

$$\frac{1}{m}\sum_{i=1}^{m}\log D(x^{(i)}) + \frac{1}{m}\sum_{i=1}^{m}\log(1 - D(g_\theta(z^{(i)})))$$

Generator minimizes:

$$\frac{1}{m}\sum_{i=1}^{m}\log(1 - D(g_\theta(z^{(i)})))$$



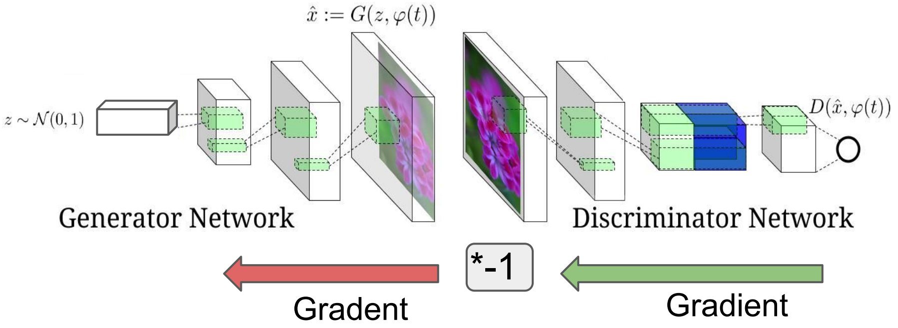
Generating images

Discriminator maximizes:

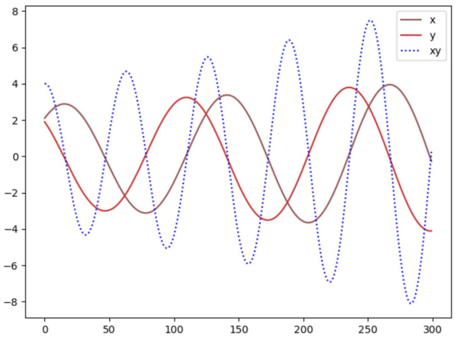
$$\frac{1}{m}\sum_{i=1}^{m}\log D(x^{(i)}) + \frac{1}{m}\sum_{i=1}^{m}\log(1 - D(g_\theta(z^{(i)})))$$

Generator minimizes:

$$\frac{1}{m}\sum_{i=1}^{m}\log(1 - D(g_\theta(z^{(i)})))$$



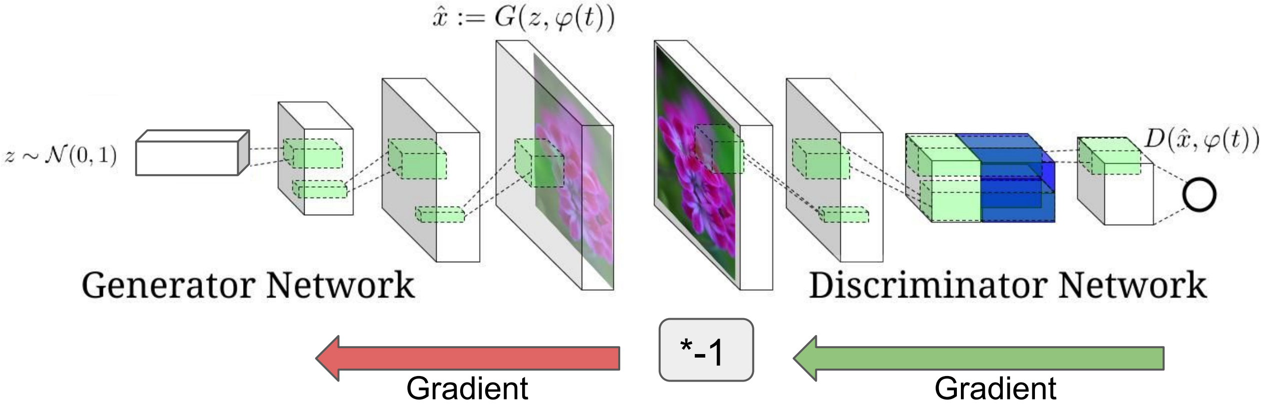
How do you know that you are improving?



What does *z* mean, if anything

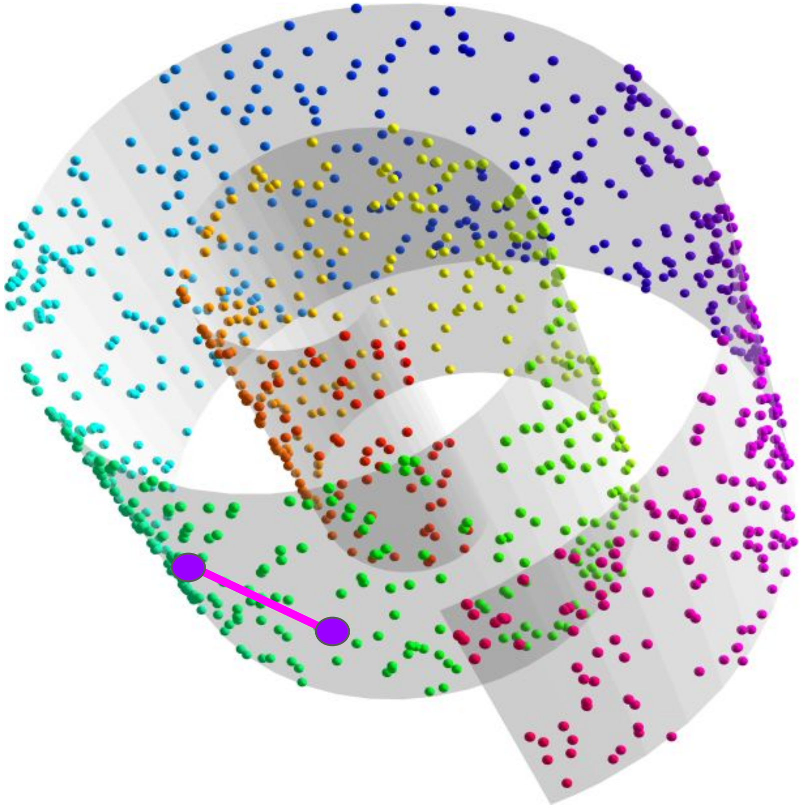
The network is trained to:

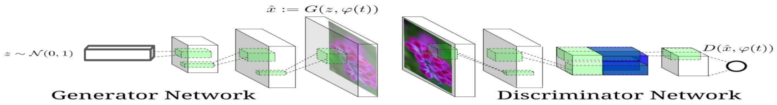
- Generate a feasible image for all possible values of *z*



A manifold representation view

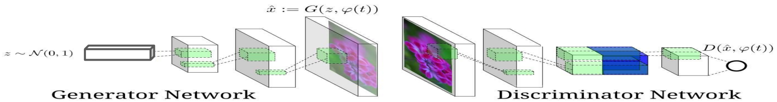
- Since all *z* are “valid” images, it means we have found a transformation from the image manifold to pixel space

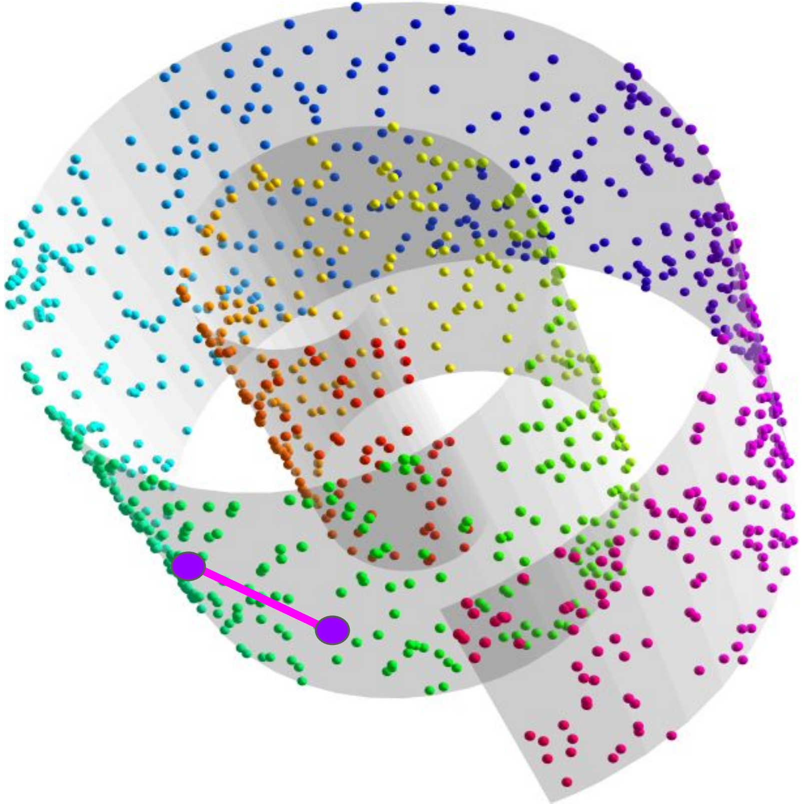




A manifold representation view

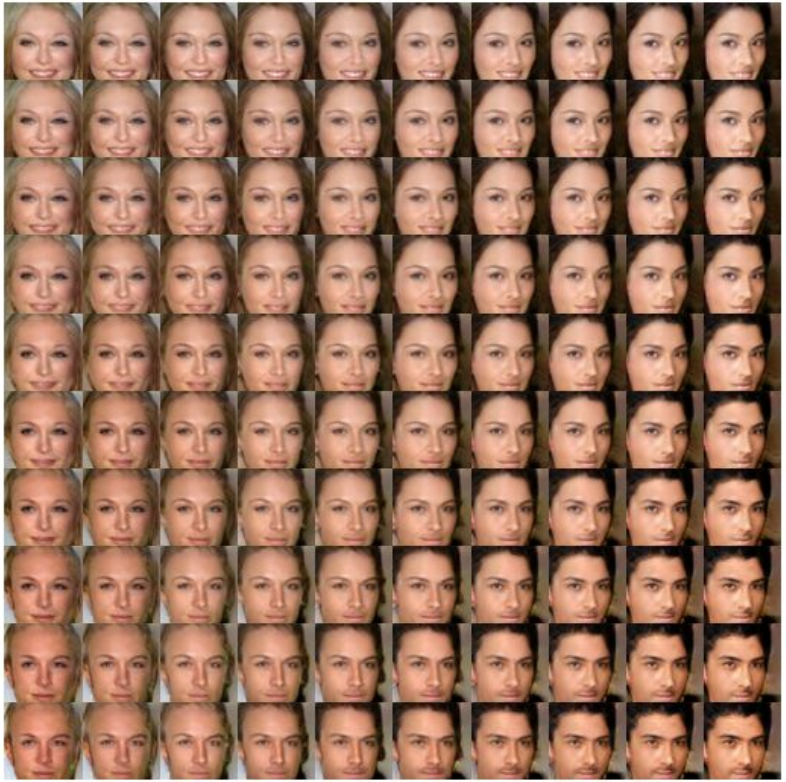
- Since all *z* are “valid” images, it means we have found a transformation from the image manifold to pixel space
- Or at least an approximation…





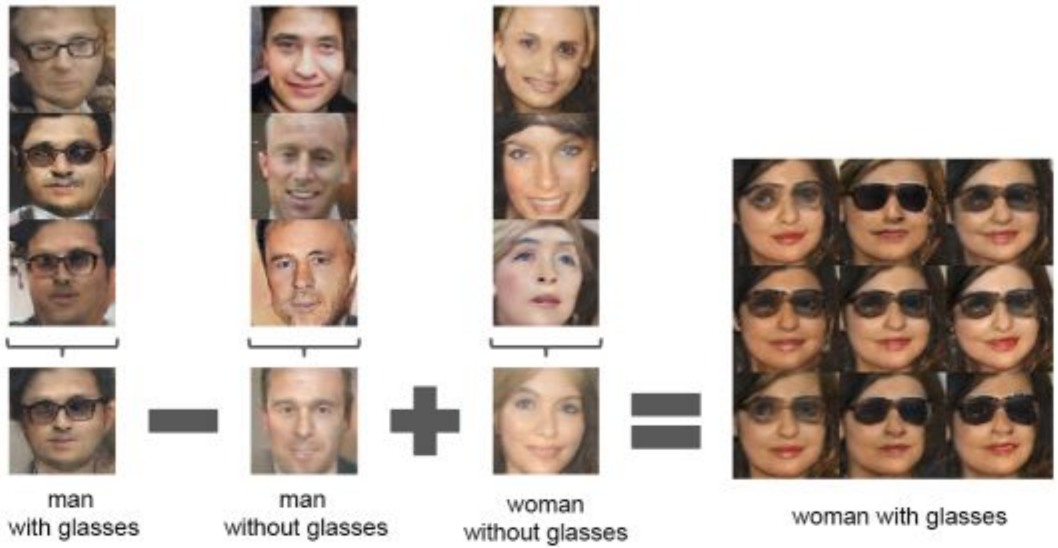
Moving along the manifold

- Small changes in input generally generally give small changes in output
- This means that you can interpolate between *z* vectors and get gradual changes in images



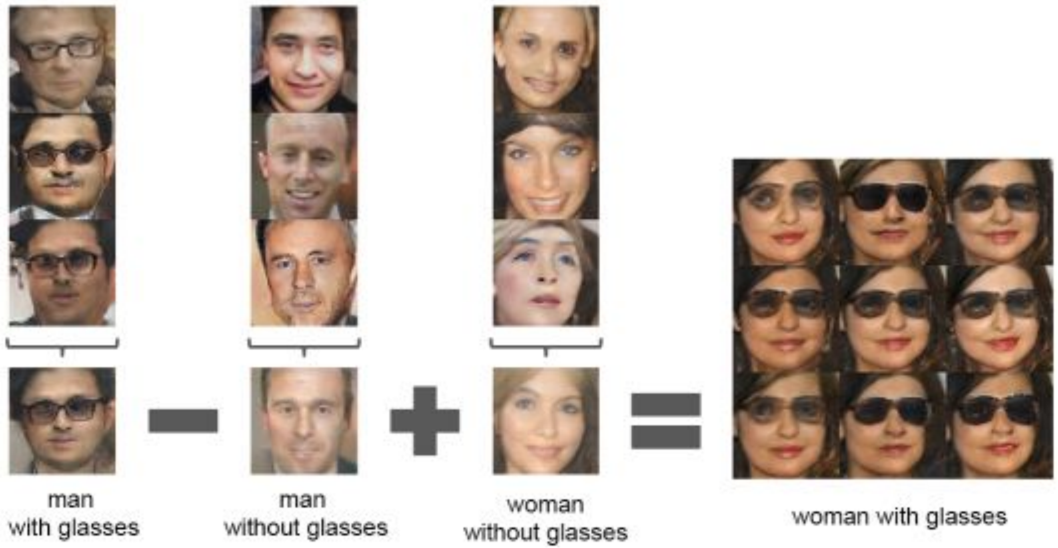
Moving along the manifold

- Similar results as variational autoencoder
- Interesting arithmetic effects
- May be an effect of the way networks effectively stores representations… *shared*
- Still some work to find representational vectors

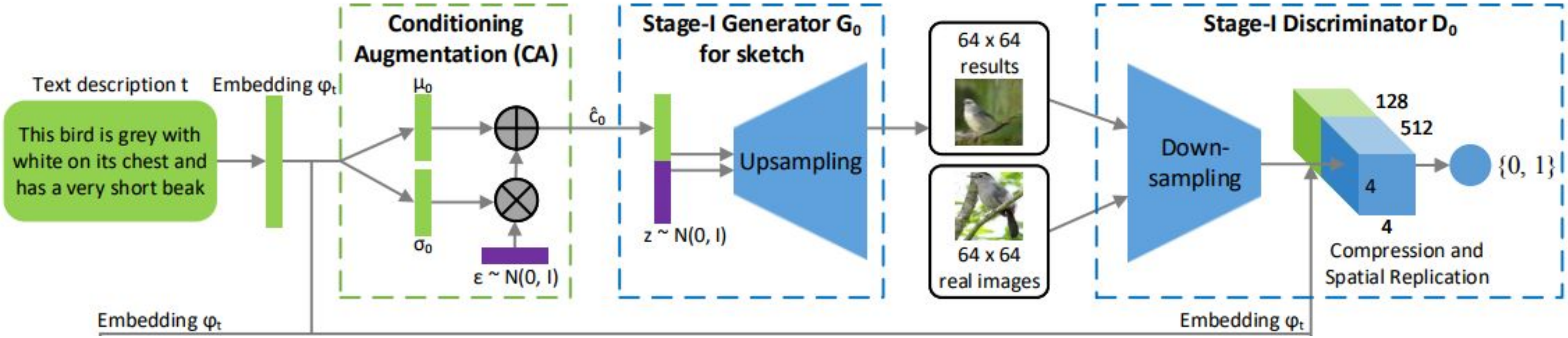


Looking into the Z-vector

- Manual work to find “glasses” representation etc.
- Need multiple examples

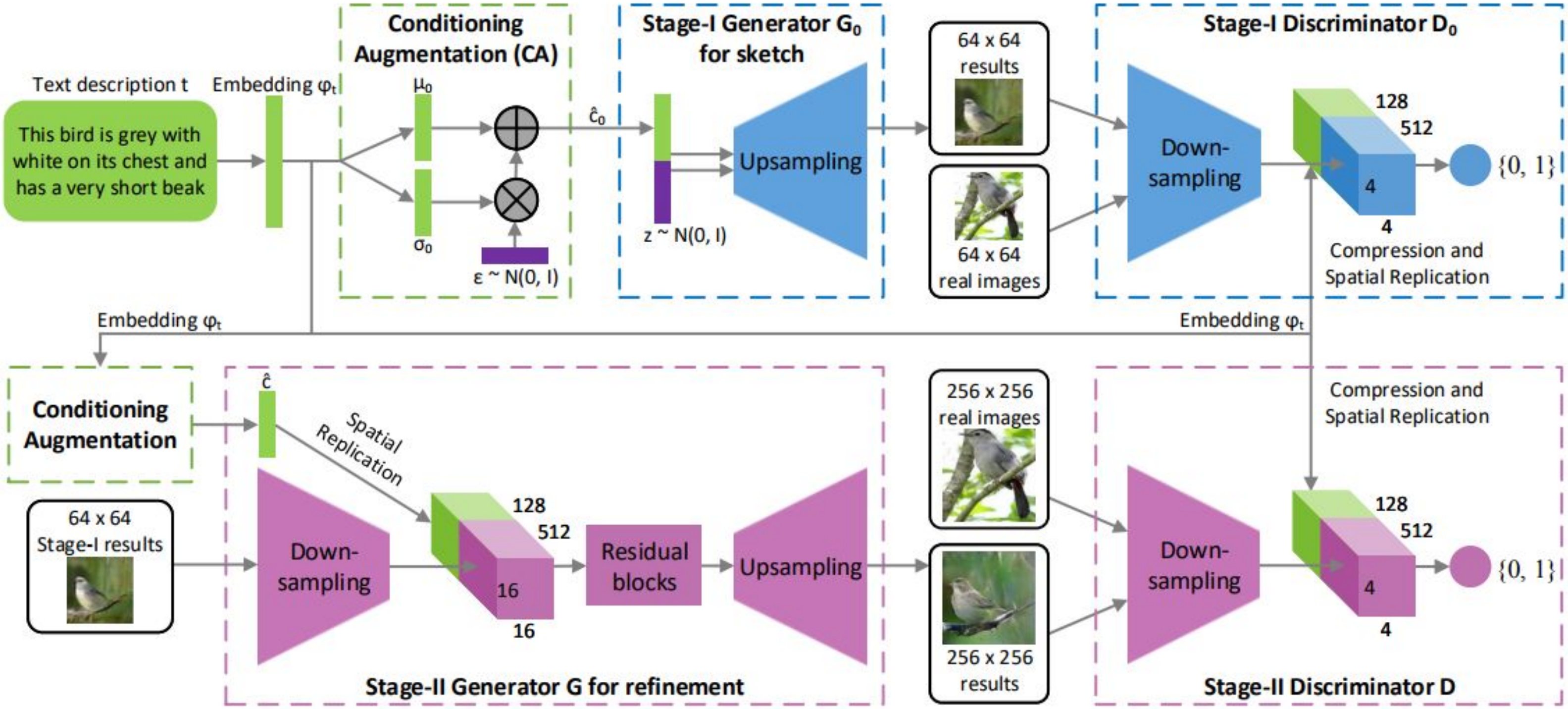


Conditional image generation



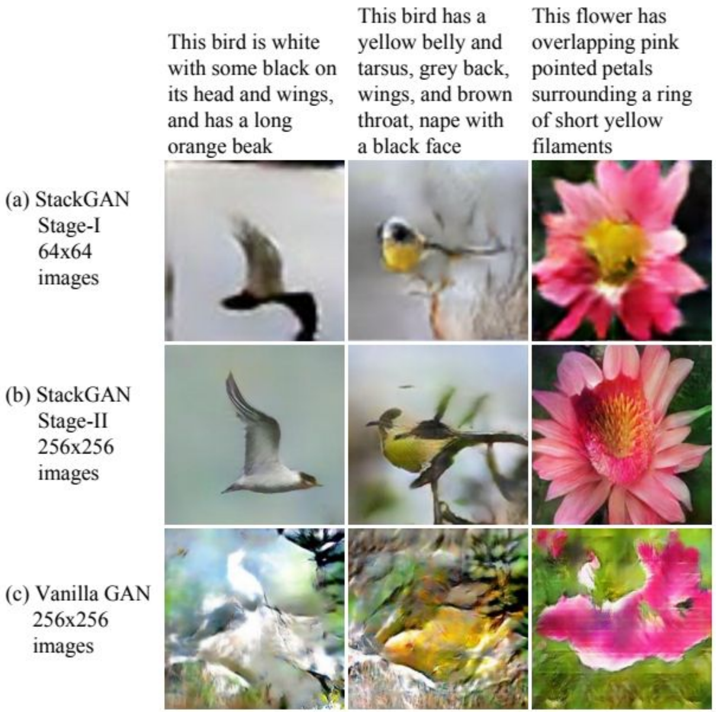
StackGAN

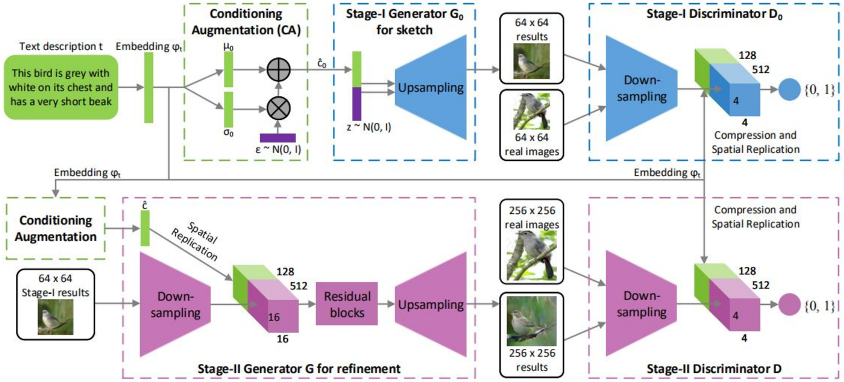
Generated images



StackGAN

Generated images





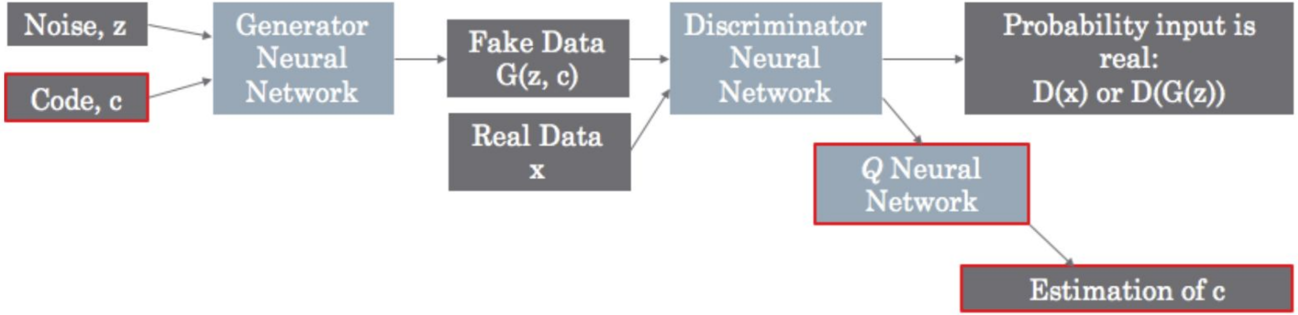
[StackGAN](#)

Generated images

StackGAN

InfoGAN - Unsupervised

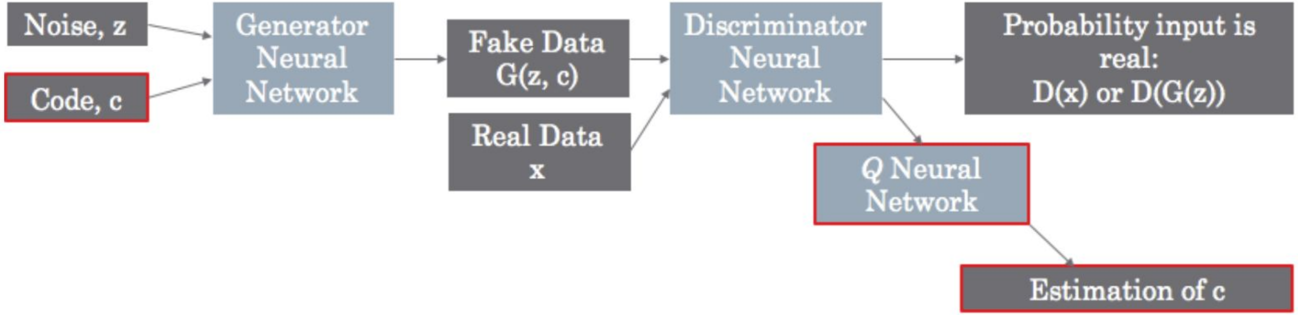
1. Add code: Input a code in addition to the random noise



[InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets]

InfoGAN - Unsupervised

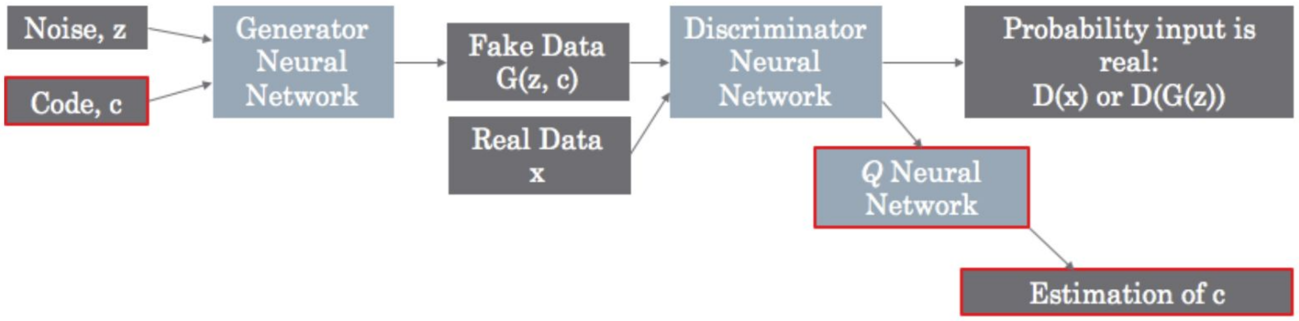
1. Add code
2. Guess c: Let the discriminator network also estimated a probability distribution of the code (given G(x,c))



[InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets]()

InfoGAN - Unsupervised

1. Add code
2. Guess c
3. Favors generated images that clearly show it's code



Adding a regularization loss, basically guessing code:

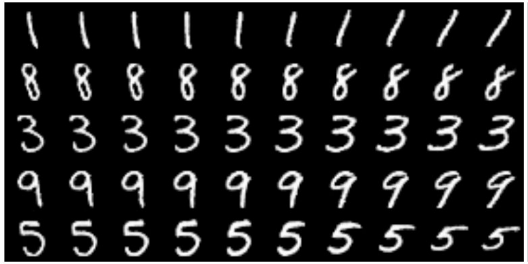
$$\lambda L_I(G, Q)$$

[InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets]

InfoGAN - Results

(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)

[InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets]

InfoGAN - Results

(a) Azimuth (pose) (b) Elevation

(c) Lighting (d) Wide or Narrow

[InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets]

InfoGAN - Results

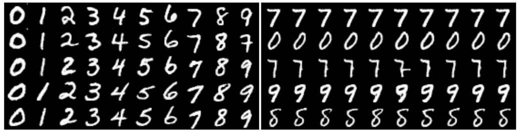
(a) Azimuth (pose) (b) Presence or absence of glasses

(c) Hair style (d) Emotion

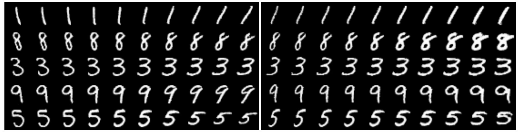
[InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets]

InfoGAN - Results

At least seems to work for data with clear modes of variance.



(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)

(a) Azimuth (pose) (b) Elevation

(c) Lighting (d) Wide or Narrow

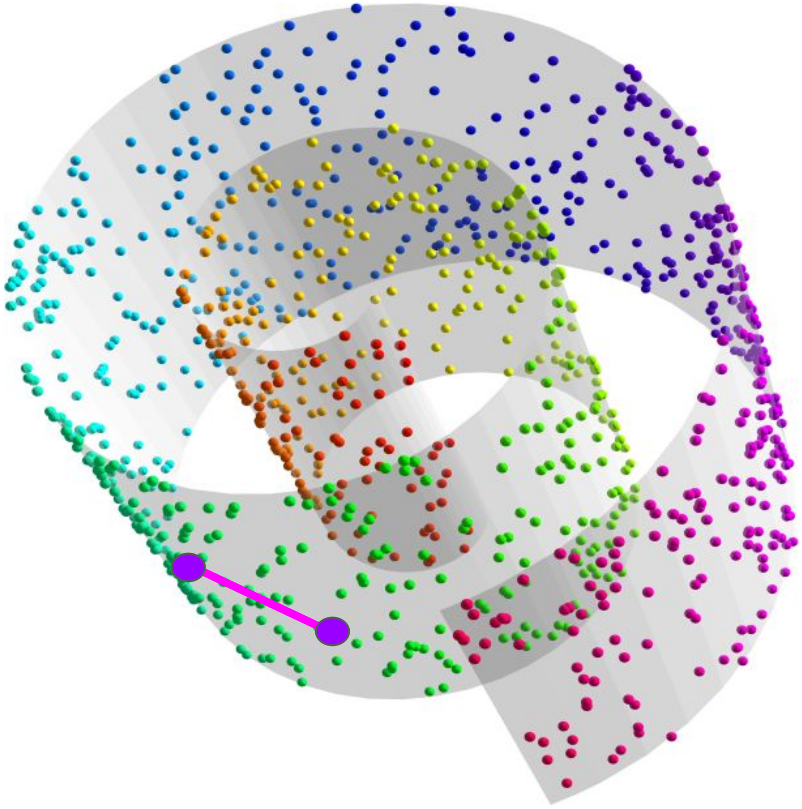
(a) Azimuth (pose) (b) Presence or absence of glasses

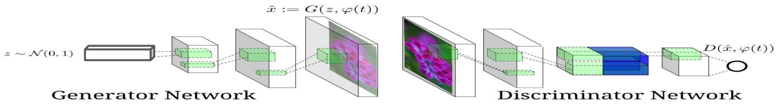
(c) Hair style (d) Emotion

[InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets]

A manifold representation view

- Unfortunately it is not representing the whole manifold
- Not even your dataset



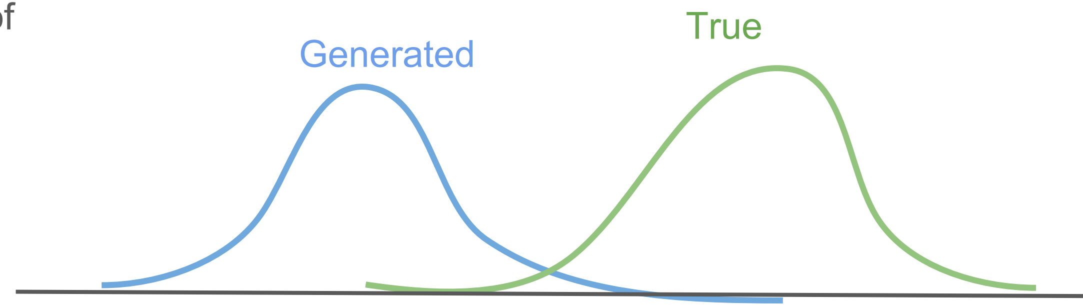


Generative adversarial networks (GAN)

Problems and improvements

A problem with standard GAN approach

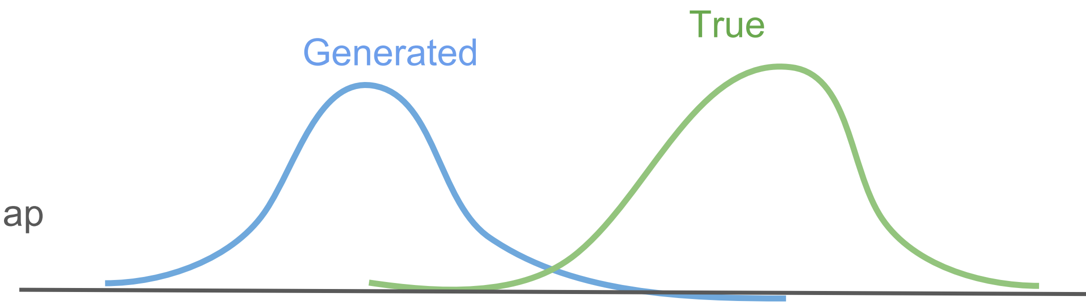
- Imagine that the distribution in the eye of the **Discriminator** is overlapping
- So *green* is the true population
- Then the **Discriminator** know that it should *enhance* features moving the generated to the left
- The **Generator** know it should enhance features moving the distribution to the right



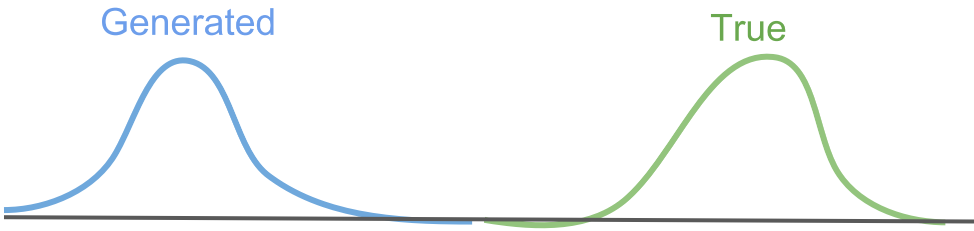
A problem with standard GAN approach

We can view adversarial learning as trying to move the output distribution of the **discriminator**.

The **generator** moves the distribution to overlap with the real images.

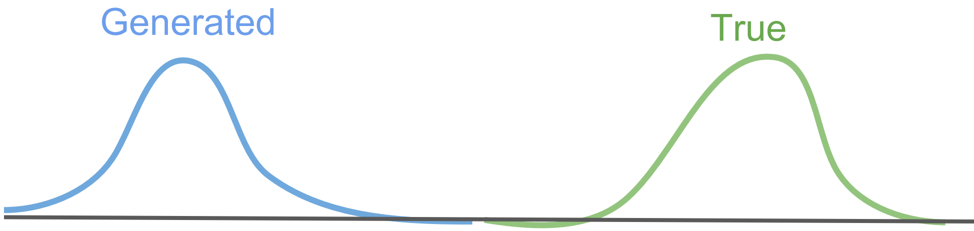


But what about this scenario?



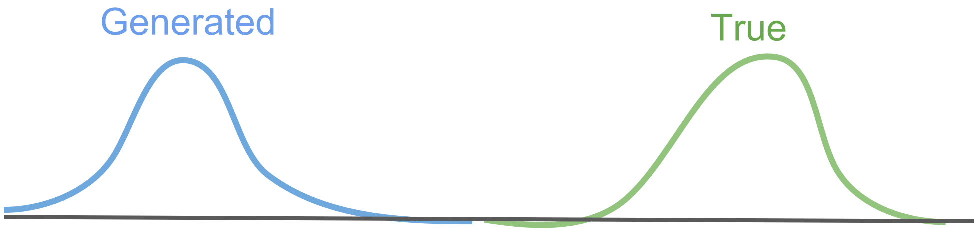
But what about this scenario?

- Overlap is less than noise level



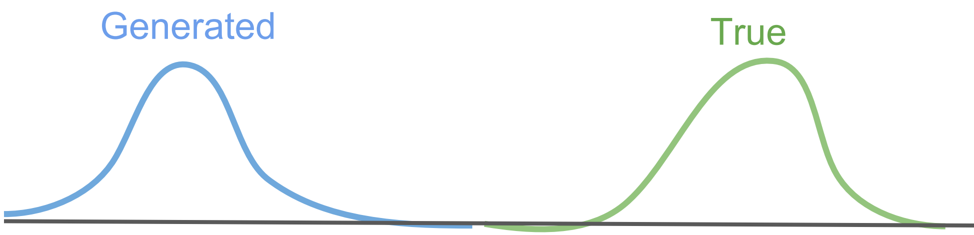
But what about this scenario?

- The discriminator cannot improve because it is already “perfect” - 0 loss
- There are no “small-step” that can improve the generator
    - Of course we know it should move to the right…
    - But *gradient descent* can only see in very small steps (short sighted)



An improved loss function (Wasserstein GAN)

1. Don’t use a standard classification loss (softmax cross-entropy)

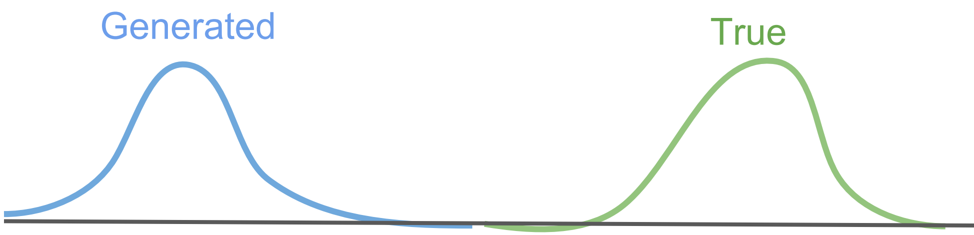


Wasserstein GAN

A "simpified article"

An improved loss function (Wasserstein GAN)

1. Don’t use a standard classification loss (softmax cross-entropy)
2. Simply let the generator maximize the distance from the mean of the generated examples for each real sample

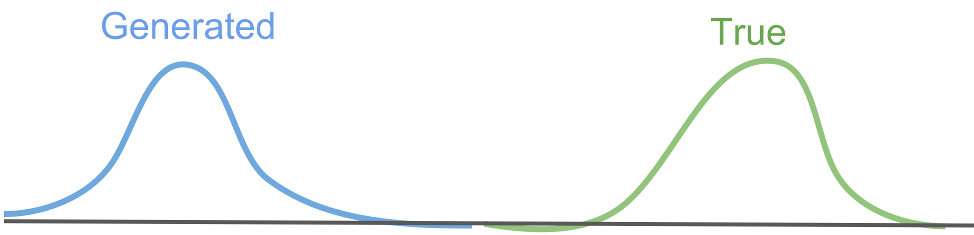


[Wasserstein GAN]

[A "simpified article"]

An improved loss function (Wasserstein GAN)

1. Don’t use a standard classification loss (softmax cross-entropy)
2. Simply let the generator maximize the distance from the mean of the generated examples for each real sample
3. Without constraints this would favour to just to spread everything out (large weights)

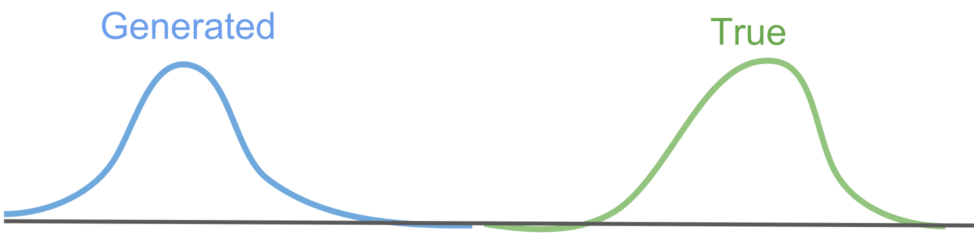


[Wasserstein GAN]

[A "simpified article"]

An improved loss function (Wasserstein GAN)

1. Don’t use a standard classification loss (softmax cross-entropy)
2. Simply let the generator maximize the distance from the mean of the generated examples for each real sample
3. Without constraints this would favour to just to spread everything out (large weights)
4. Clip the weights with a constant to avoid this.



[Wasserstein GAN]

[A "simpified article"]

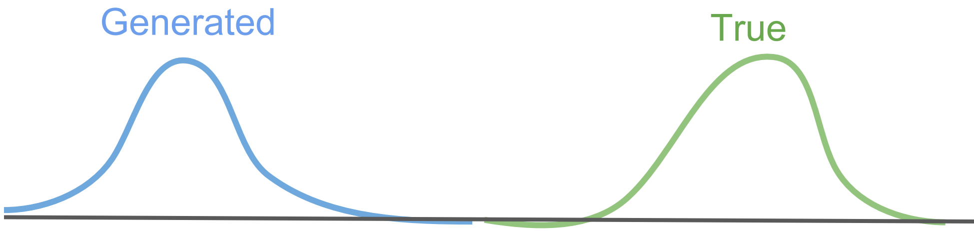
An improved loss function (Wasserstein GAN)

Discriminator loss:

- Simply making output from *true images* give high values and from *false images* low values

Generator loss:

- *False images* should give high values
- Putting the examples where the *true images* are.



Discriminator loss

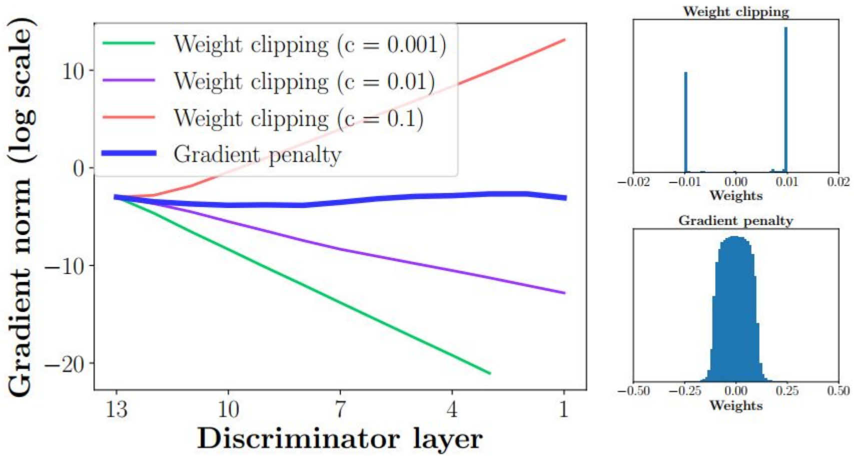
$$g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))\right]$$

Generator loss

$$g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))$$

WGAN - Nasty gradient clipping

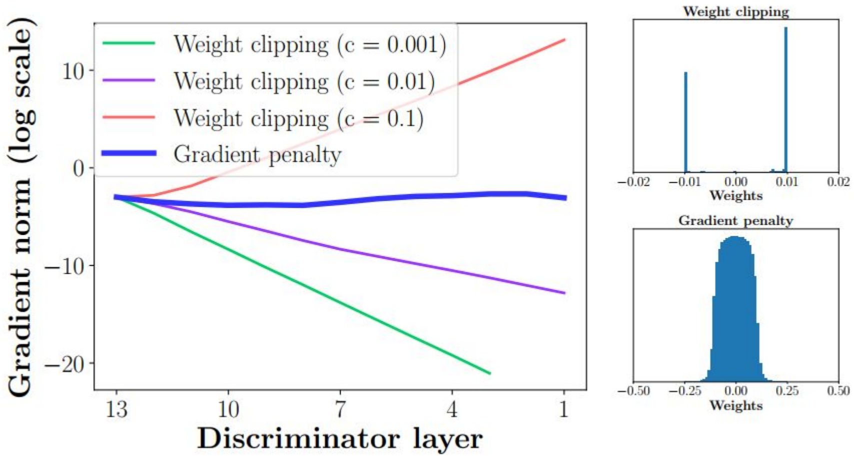
- WGAN performance is very dependent on the clipping constant
- Clipping the weights will drastically increase training time



[Improved Training of Wasserstein GANs]

WGAN - Nasty gradient clipping

- WGAN performance is very dependent on the clipping constant
- Clipping the weights will drastically increase training time
- Adding an additional cost to the gradient size, improves this
- Restricting the “movement” of the discriminator



$$L = \underbrace{\underset{\tilde{\boldsymbol{x}} \sim \mathbb{P}_g}{\mathbb{E}} \left[D(\tilde{\boldsymbol{x}})\right] - \underset{\boldsymbol{x} \sim \mathbb{P}_r}{\mathbb{E}} \left[D(\boldsymbol{x})\right]}_{\text{Original critic loss}} + \underbrace{\lambda \underset{\hat{\boldsymbol{x}} \sim \mathbb{P}_{\hat{\boldsymbol{x}}}}{\mathbb{E}} \left[(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2 - 1)^2\right]}_{\text{Our gradient penalty}}.$$

[Improved WGAN blog post]

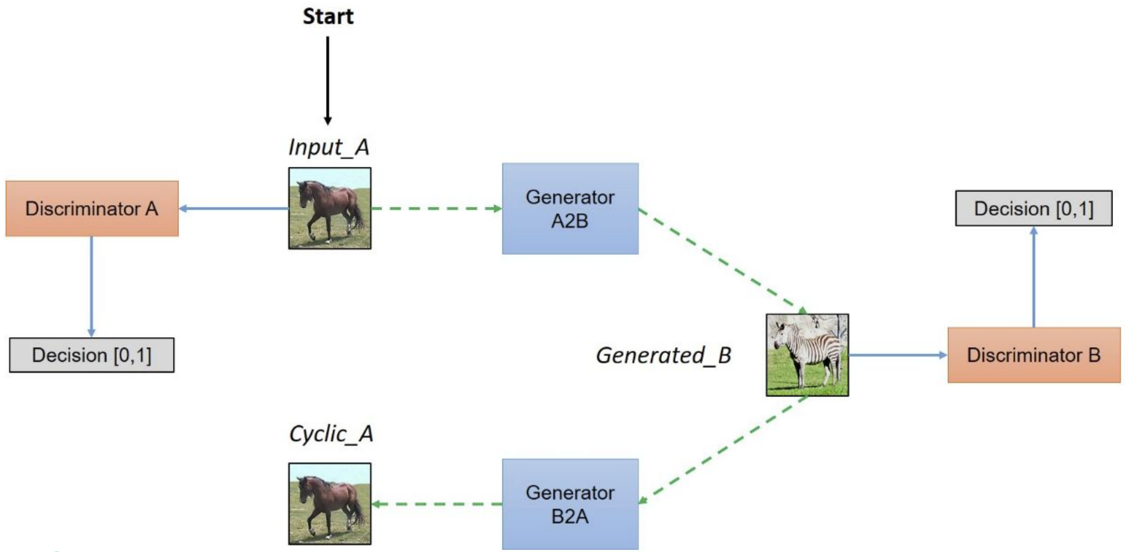
[Improved Training of Wasserstein GANs]

Generative adversarial networks (GAN)

More examples

CycleGAN - unpaired image to image translation

1. Unpaired images from two different domains

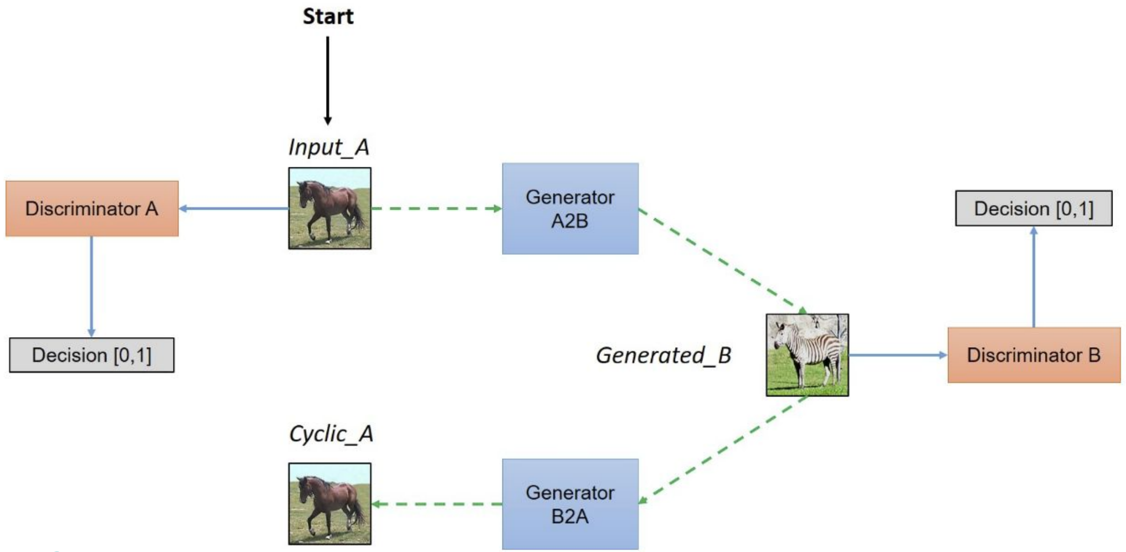


[CycleGAN blog]

[Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks]

CycleGAN - unpaired image to image translation

1. Unpaired images from two different domains
2. Use image from one domain as Z

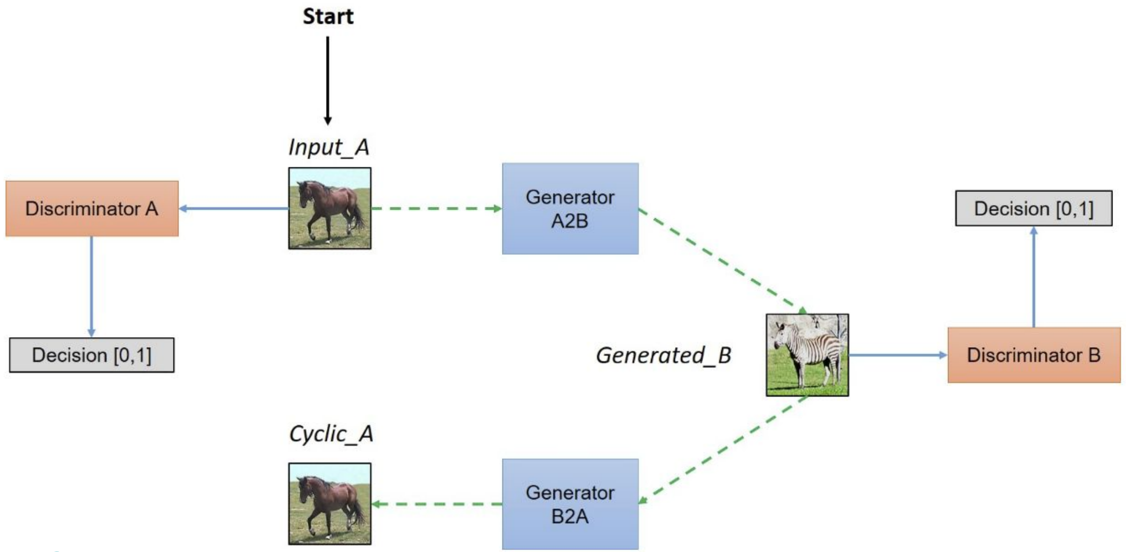


[CycleGAN blog]

[Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks]

CycleGAN - unpaired image to image translation

1. Unpaired images from two different domains
2. Use image from one domain as Z
3. Generate image from the other domain

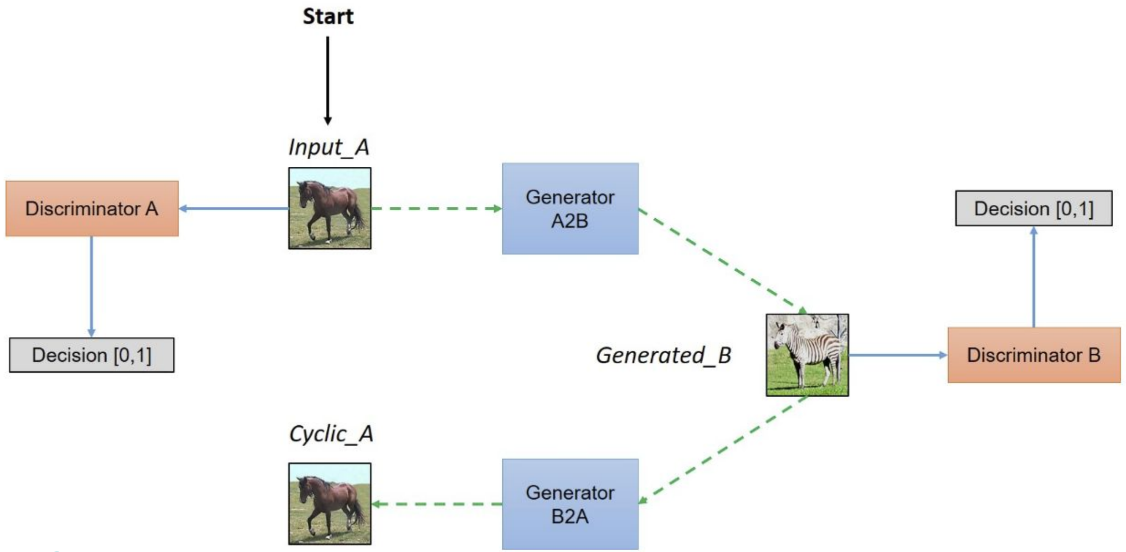


[CycleGAN blog]

[Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks]

CycleGAN - unpaired image to image translation

1. Unpaired images from two different domains
2. Use image from one domain as Z
3. Generate image from the other domain
4. Align images with cycle consistency loss



[Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks]

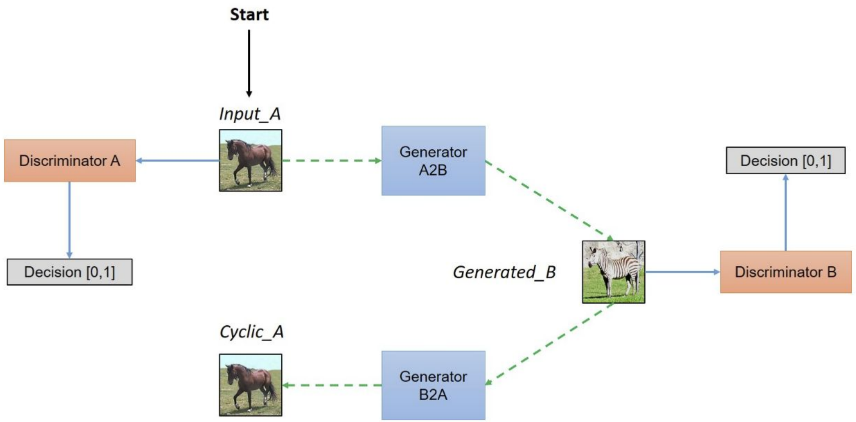
[CycleGAN blog]

CycleGAN - unpaired image to image translation

Cycle consistency loss:

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

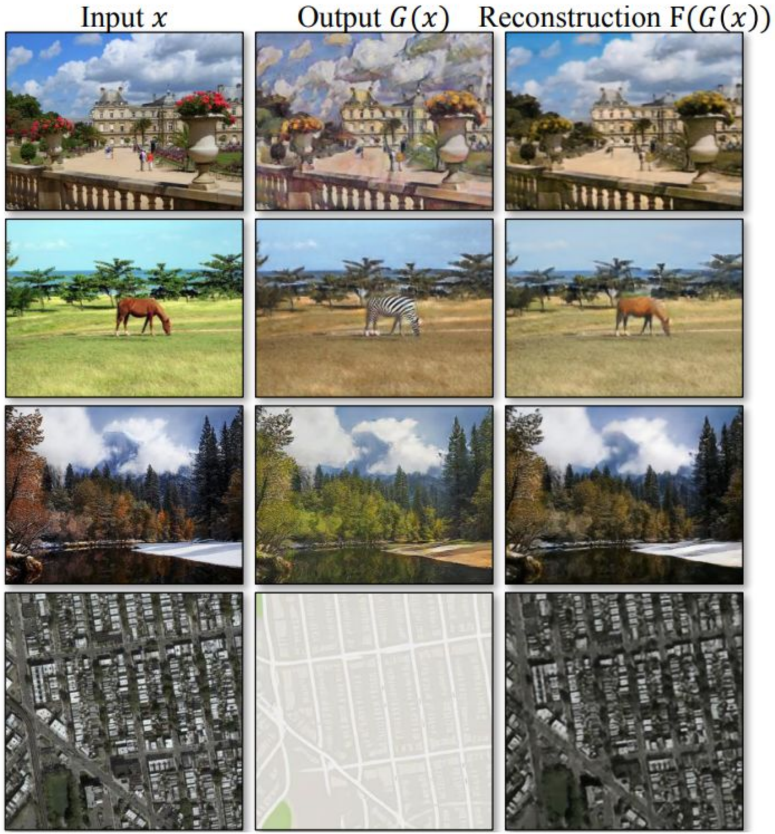
$$\begin{aligned} \mathcal{L}(G, F, D_X, D_Y) =& \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) \\ &+ \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) \\ &+ \lambda \mathcal{L}_{\text{cyc}}(G, F), \end{aligned}$$



[Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks]

[CycleGAN blog]

CycleGAN - unpaired image to image translation



[Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks]()

CycleGAN - unpaired image to image translation

Unpaired Image-to-Image
Translation using
Cycle-Consistent
Adversarial Networks

CycleGAN - unpaired image to image translation

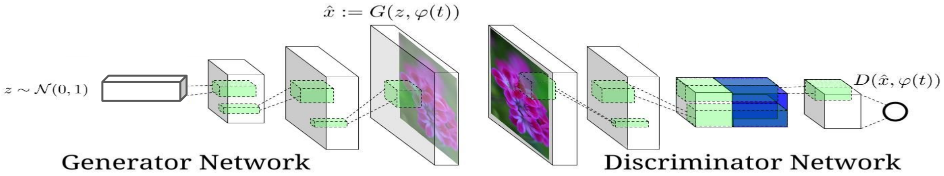
Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks

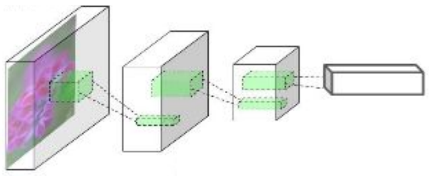
An awesome application - A case study

- Using GAN for photo editing
- A reverse mapping from image space to closest point on manifold

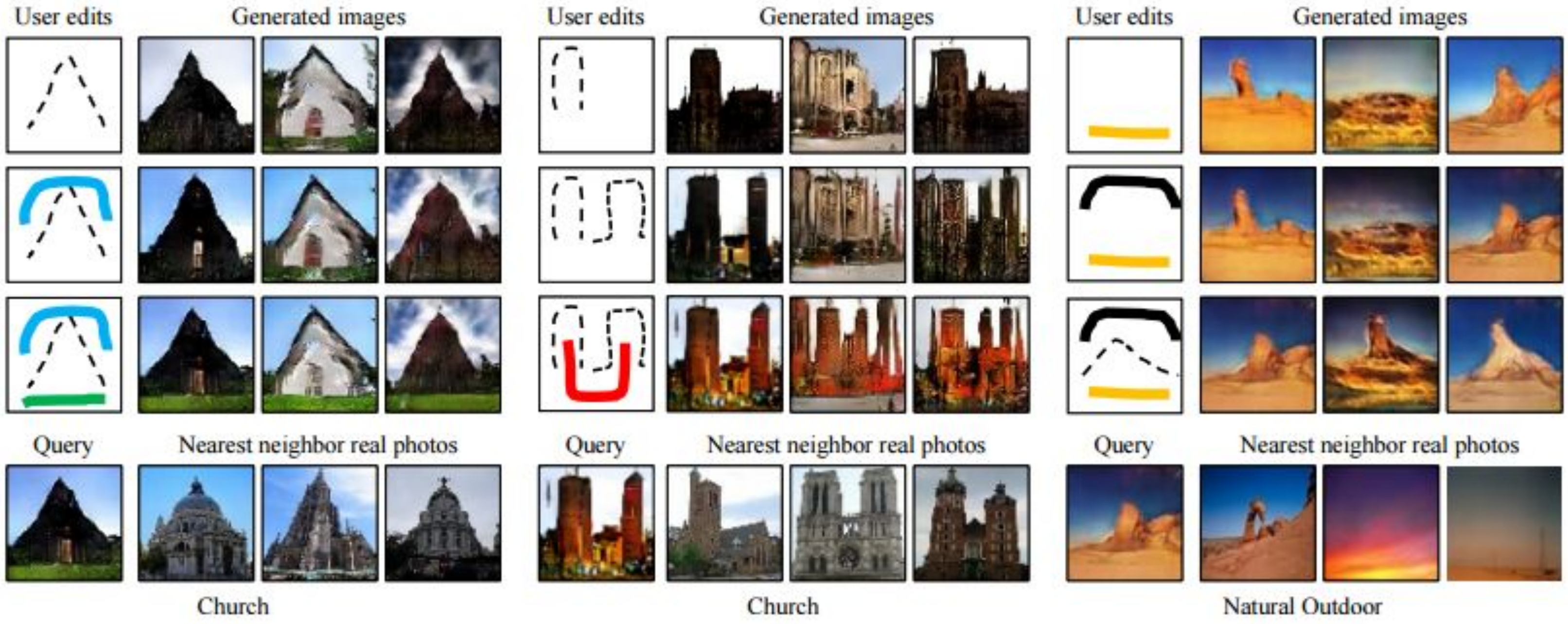
Finding the closest point on the manifold

- Train a network to predict the embedding of a generated image
- Use that network to find an embedding *z*
- Optimize/train that *z* vector to minimize *mean squared error*





Profit!


User edits
Generated images
Query
Nearest neighbor real photos
Church
User edits
Generated images
Query
Nearest neighbor real photos
Church
User edits
Generated images
Query
Nearest neighbor real photos
Natural Outdoor

GANs still have problems with context

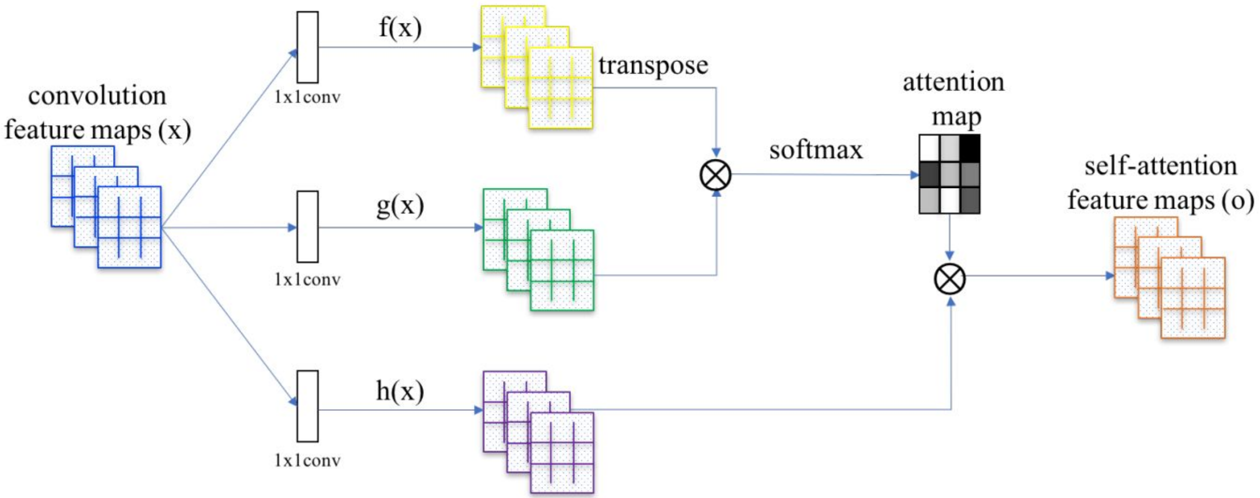
On more complex image domains (ImageNet), GANs often show problems with context. Multiple heads, legs and deformed figures.

Figure 10: ImageNet 256 × 256 generations using an EBGAN-PT.

[Energy-Based Generative Adverserial Networks]

Attention to improve context

1. For each pixel location, compute an attention map
2. Multiply each attention map with input features
3. Use attention in top layer of both generator and discriminator
4. Train GAN as normal

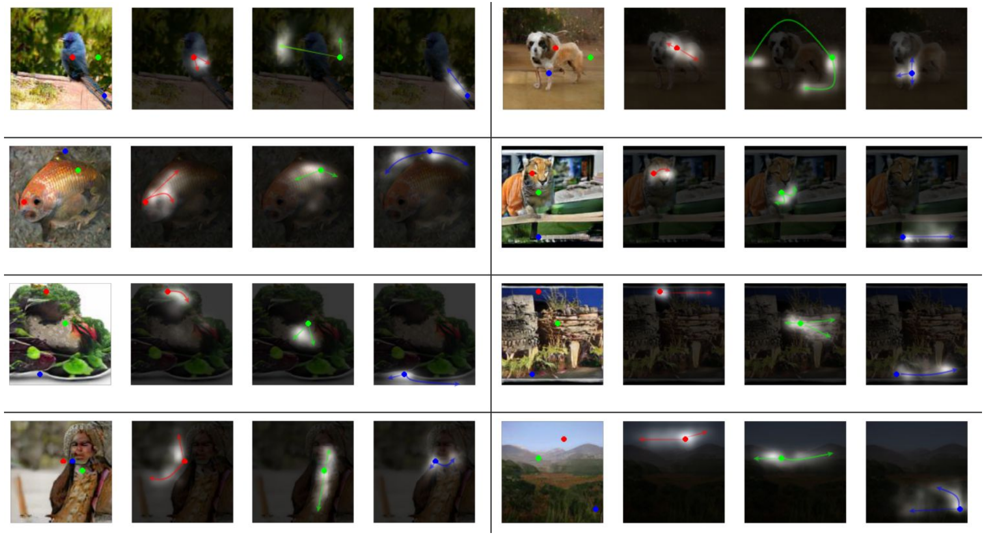


[Self-Attention Generative Adversarial Networks](#) [Non-local Neural Networks](#)

Attention to improve context

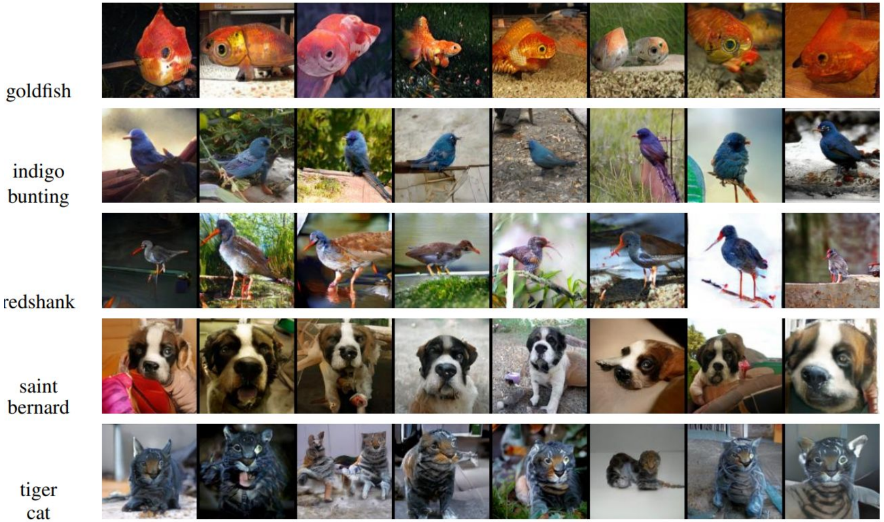
Inspection of attention maps for generator:

- For generating legs, the model looks at both the length and neighbour leg
- Looks at relevant context...



[Self-Attention Generative Adversarial Networks]   [Non-local Neural Networks]

Self-Attention GANs - examples



[Self-Attention Generative Adversarial Networks]

Self-Attention GANs - Tuning and increasing batch size

[Large Scale GAN Training For High Fidelity Natural Image Synthesis]

GANs - Fun, but difficult

Fun:

- Give a lot of opportunities
- Losses that are otherwise impossible or hard

Hard to train

- Discriminator win
- Training longer can make it worse
- Bigger models can be worse than smaller
- More data, does not improve the model