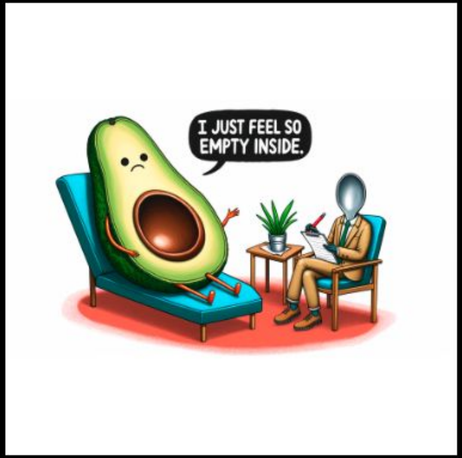
Lecture 7
Generative modeling

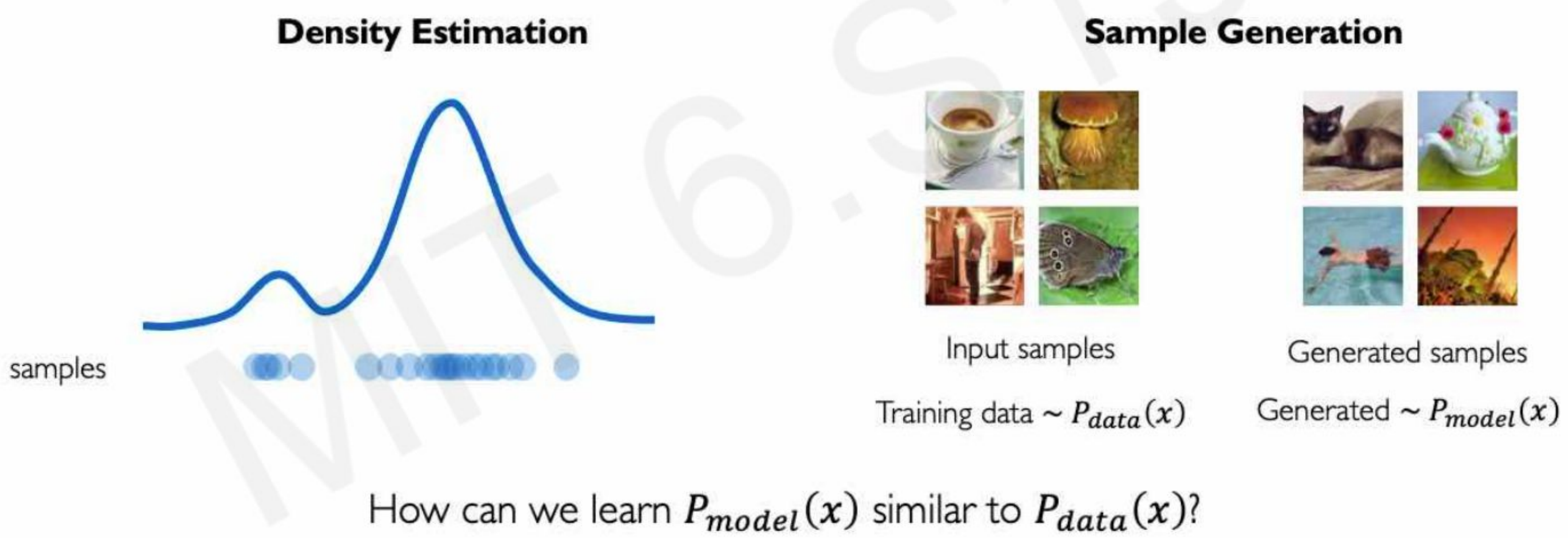
Image taken from https://generated.photos/faces





dall-e-3

Generative models

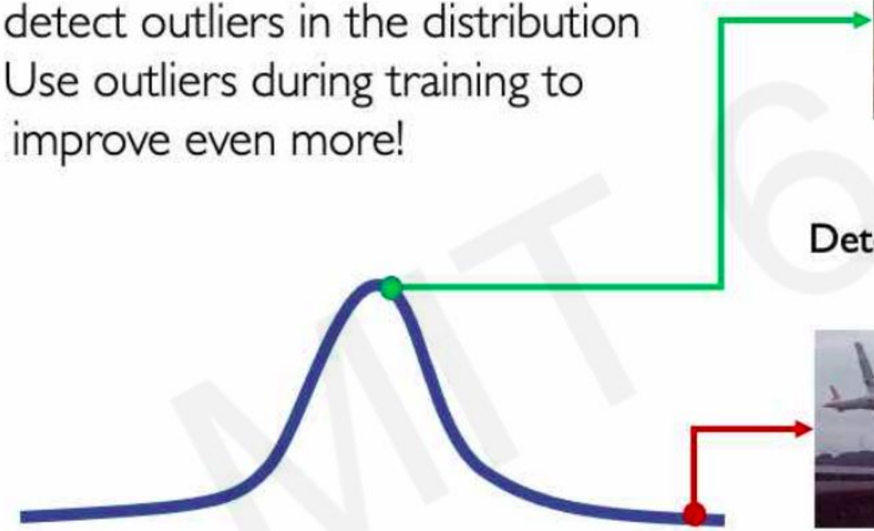


source

Why generative models? Outlier detection

- **Problem:** How can we detect when we encounter something new or rare?
- **Strategy:** Leverage generative models, detect outliers in the distribution
- Use outliers during training to improve even more!

95% of Driving Data:
(1) sunny, (2) highway, (3) straight road



Detect outliers to avoid unpredictable behavior when training

Edge Cases

Harsh Weather

Pedestrians

source

But why?

- better understand the data (model structure, parameters, etc.) → outlier detection,
- impute missing data,
- facilitate supervised learning,
- creative applications.

Plan

- Probabilistic nature of datasets: latent variable models.
- GAN-s.
- Generative models: Autoencoders & Variational Autoencoders (VAE)
-
- Lecture by Mateusz Wyszyński (1 or 2 weeks from now):
    - Diffusion models.
    - Flow matching.

Slides credits

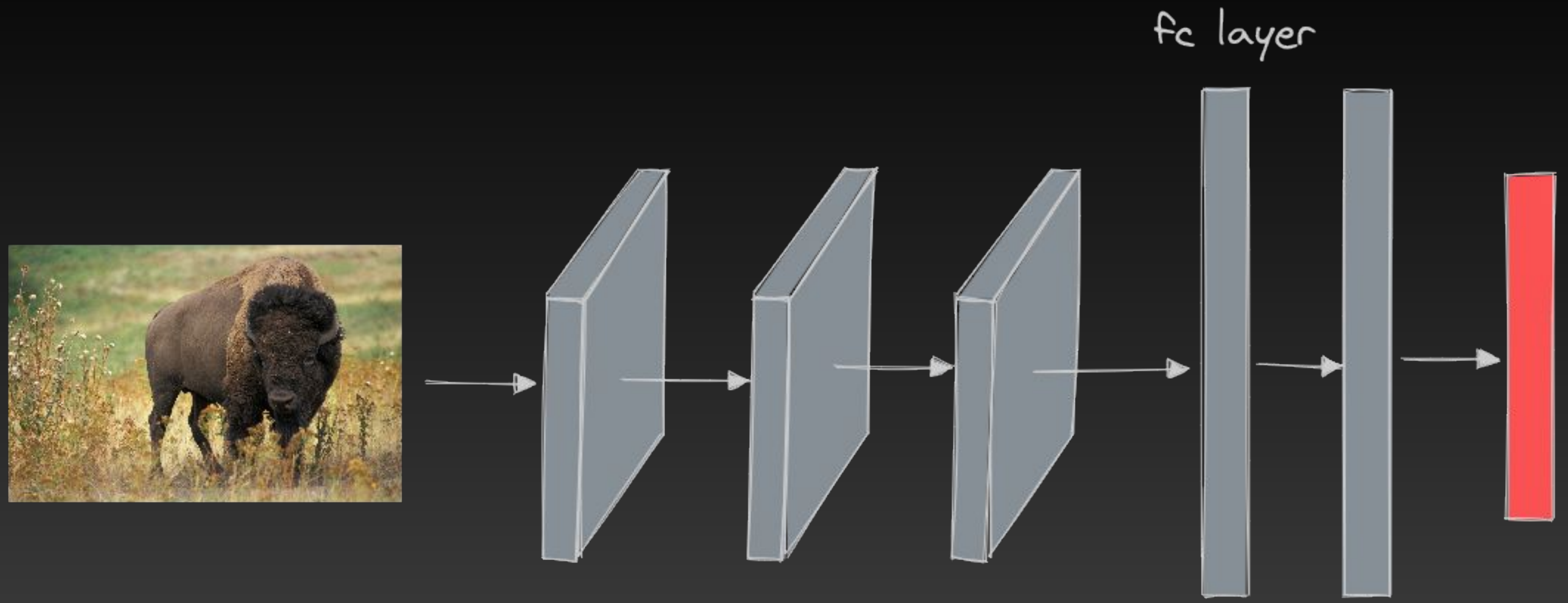
Some slides taken from:

- CS 182 - Berkeley course by Sergey Levine (lecture 17, 18).
- MIT Introduction to Deep Learning

Some images from:

- Understanding Variational Autoencoders (VAEs)

Internal representations


fc layer

Visualizing the representation

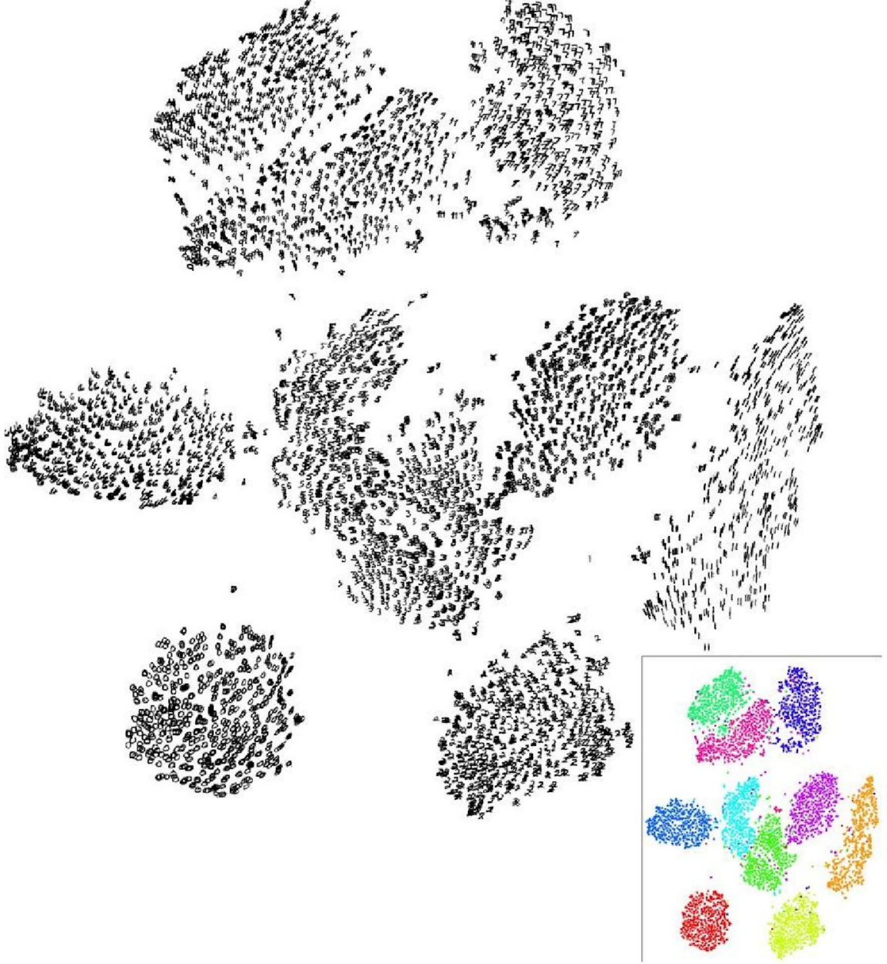
t-SNE visualization

[van der Maaten & Hinton]

Embed high-dimensional points so that locally, pairwise distances are conserved

i.e. similar things end up in similar places. dissimilar things end up wherever

Right: Example embedding of MNIST digits (0-9) in 2D



Latent space

Dataset generation

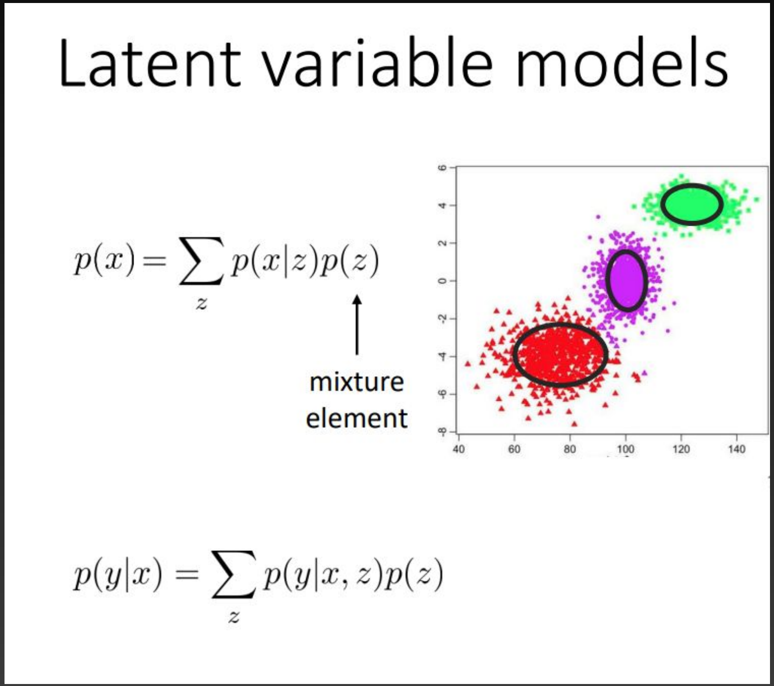
- Image dataset creation. Repeat n times:
  - sample a random image x_i from the internet,
  - a human assigns a label y_i with prob.: $p(y_i|x_i)$
- The probability of obtaining $\mathcal{D} = \{(x_i, y_i) : i = 1, \ldots, n\}$

  Is $p(\mathcal{D}) = \Pi_{i=1}^{n} p(x_i) p(y_i|x_i)$

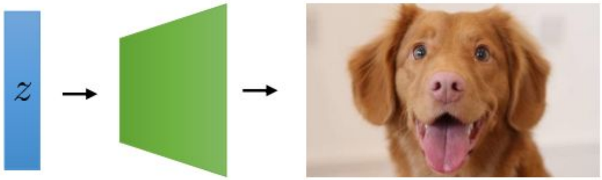
Intuition:

- Imagine a dataset of random dog images with prob p(x).

- Consider a latent space with dimensions representing dog's:
  - size,
  - fur,
  - eye color,
  - pose,
  - lighting.
- Above "type" has probability p(z).



Latent variable models in deep learning



$p(z)$ $\quad p_\theta(x|z)$

$\mathcal{N}(0, \mathbf{I})$

A latent variable deep generative model is (usually) just a model that turns random numbers into valid samples (e.g., images)

Please don't tell anyone I said this, it destroys the mystique

There are many types of such models: VAEs, GANs, normalizing flows, etc.

Using the model for **generation**:

1. sample $z \sim p(z)$ "generate a vector of random numbers"
2. sample $x \sim p(x|z)$ "turn that vector of random numbers into an image"

Generative
Adversarial
Networks

GANs - motivation

Setting:
Currency with notes that do not change over time.
Obvious target for counterfeiters.
Need to establish anti-counterfeit agency.
What happens in the long run?

Counterfeiters and agents improve simultaneously.

Eventually, counterfeiters produce perfect fakes and agents are helpless (unless it’s not possible ofc).

GANs

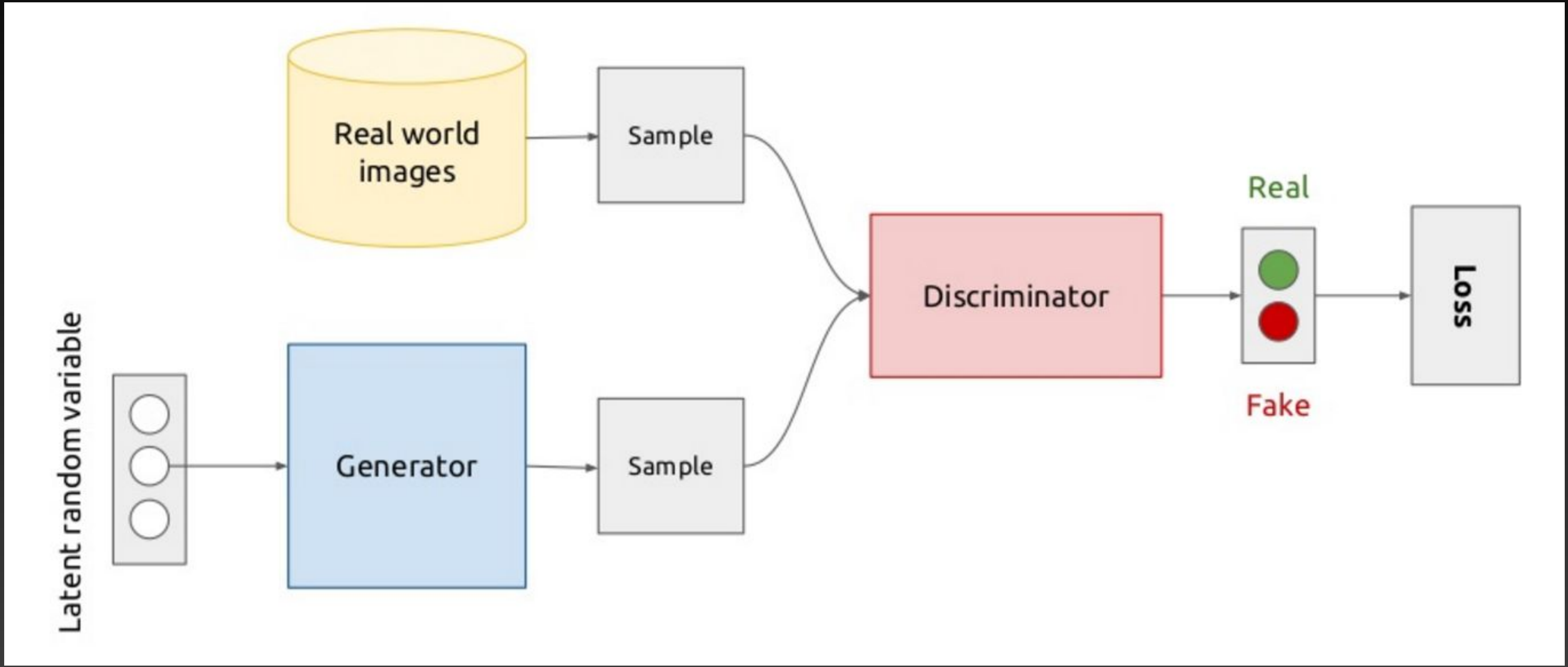


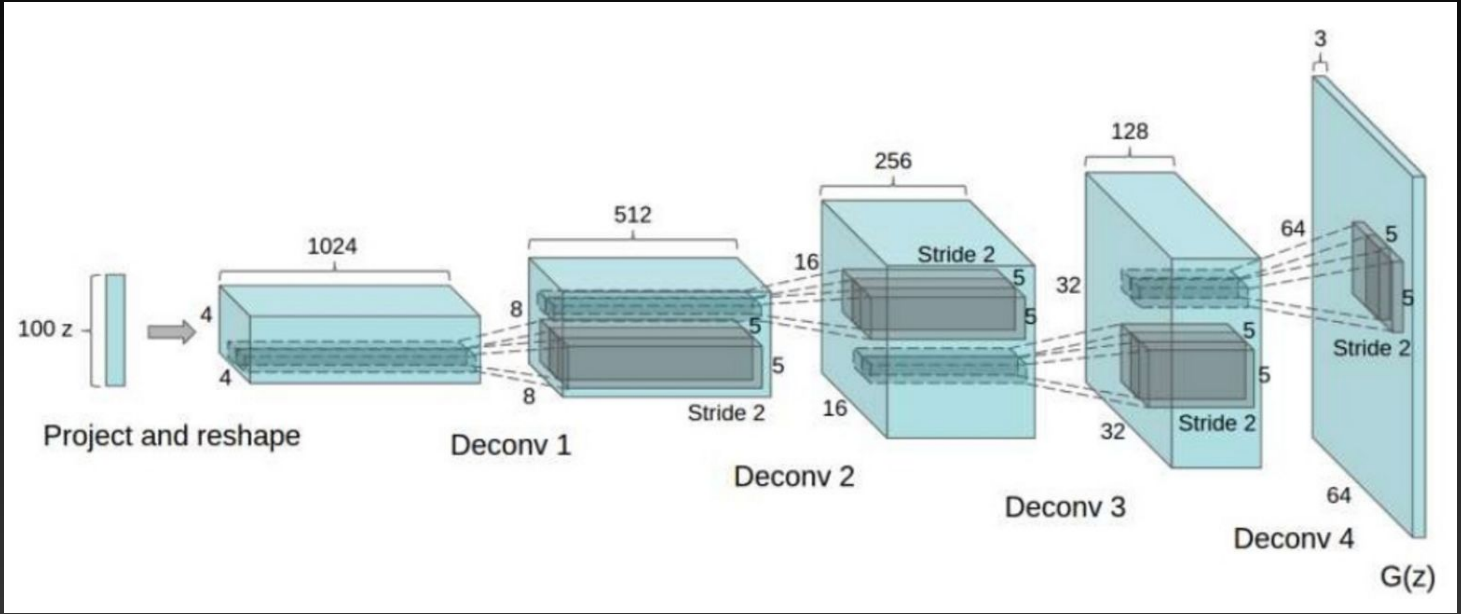
Image taken from http://imatge-upc.github.io/telecombcn-2016-dlcv/

GAN training

- Generate a batch of real images.
- Generate a batch of fake images from a batch of random vectors.
- Make a gradient step for discriminator on the two batches.

- Generate a batch of random vectors.
- Make a gradient step for generator to best fool the discriminator on these vectors.
  - Discriminator kept constant when computing the gradients.

Generator architecture (DCGAN)

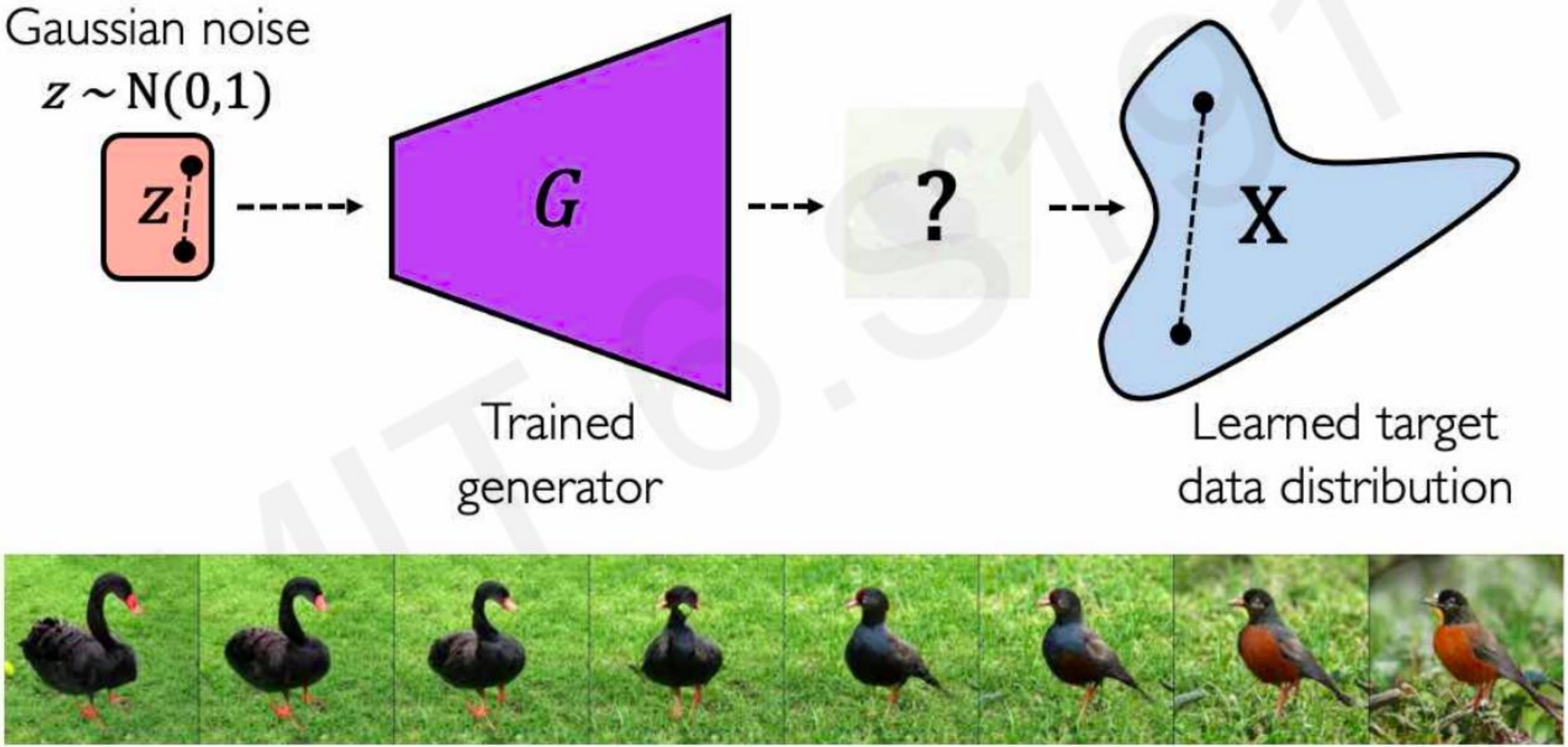


Discriminator would use a mirror of this architecture (think U-Net).

Image taken from https://arxiv.org/abs/1511.06434

Some applications

GANs are distribution transformers



source

Interpolating faces

Image taken from https://arxiv.org/pdf/1707.05776.pdf

Interpolating bedrooms

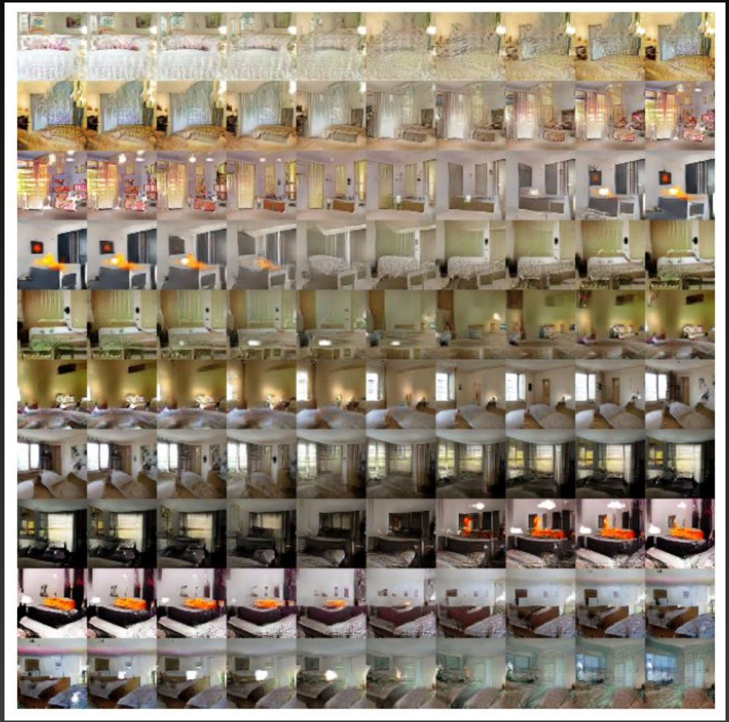


Image taken from https://arxiv.org/abs/1511.06434

Code arithmetic on faces

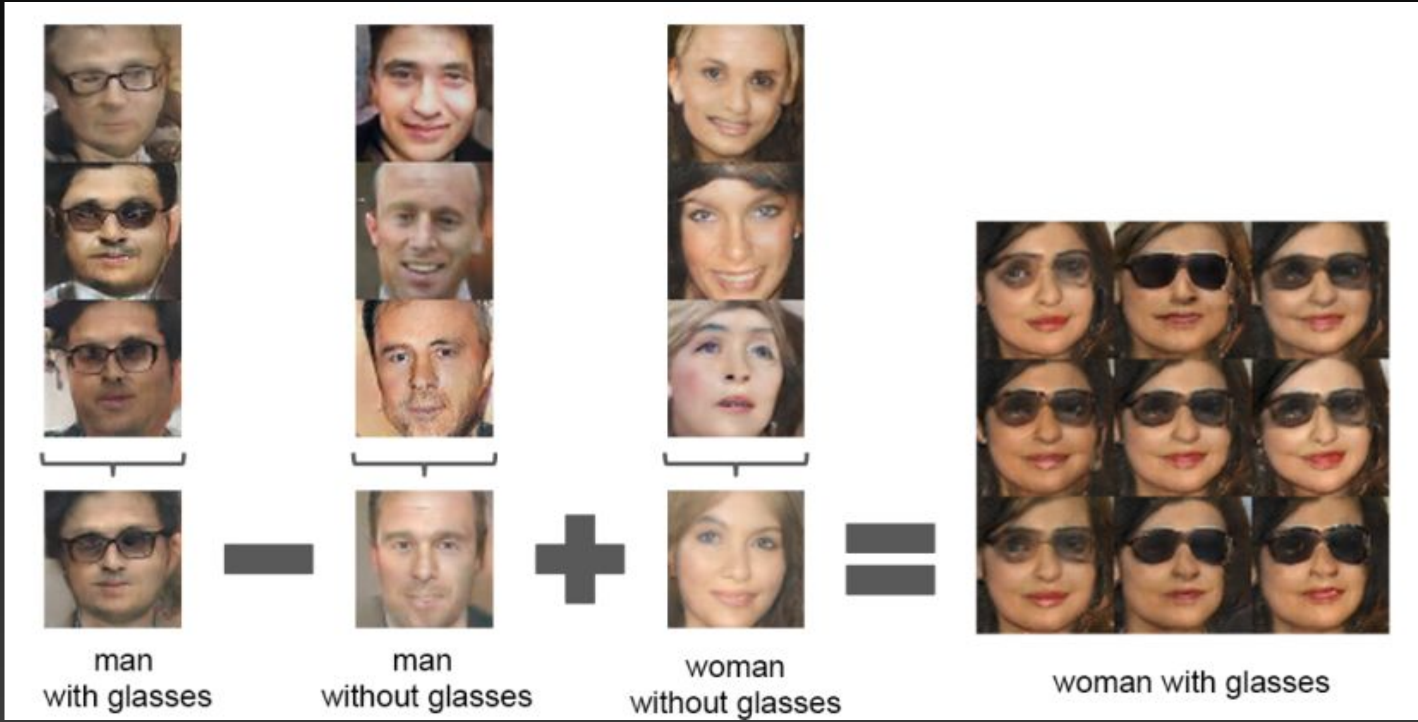
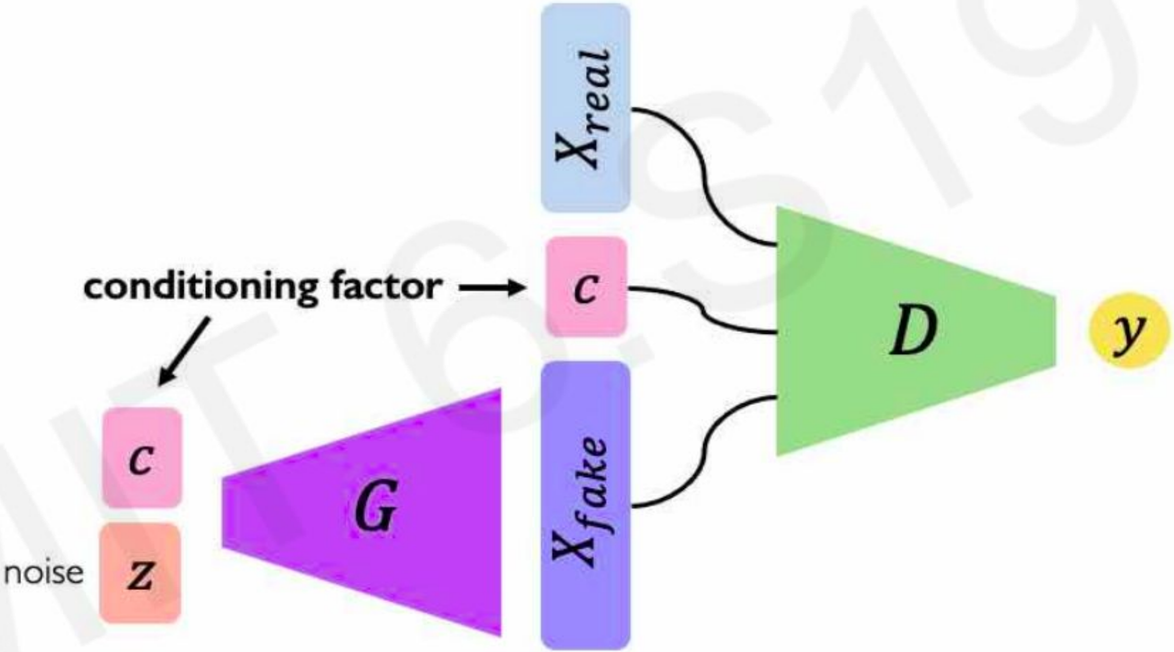


Image taken from https://arxiv.org/abs/1511.06434

Conditional GANs

What if we want to control the nature of the output, by **conditioning** on a label?



[source](#)

Conditional GAN (CGAN)

One can condition the generator's output on an arbitrary random variable y:

- append y to generator's random input z.
- append y to discriminator's input.
- the disciminator decides: does the image come from $p_{data}(\ |y)$.

Examples:

- sample a given digit from MNIST,
- sample a given class from ImageNet.

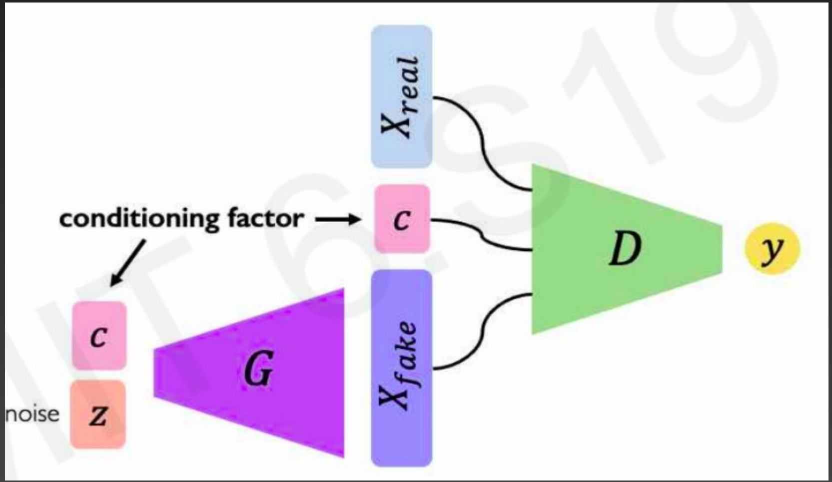
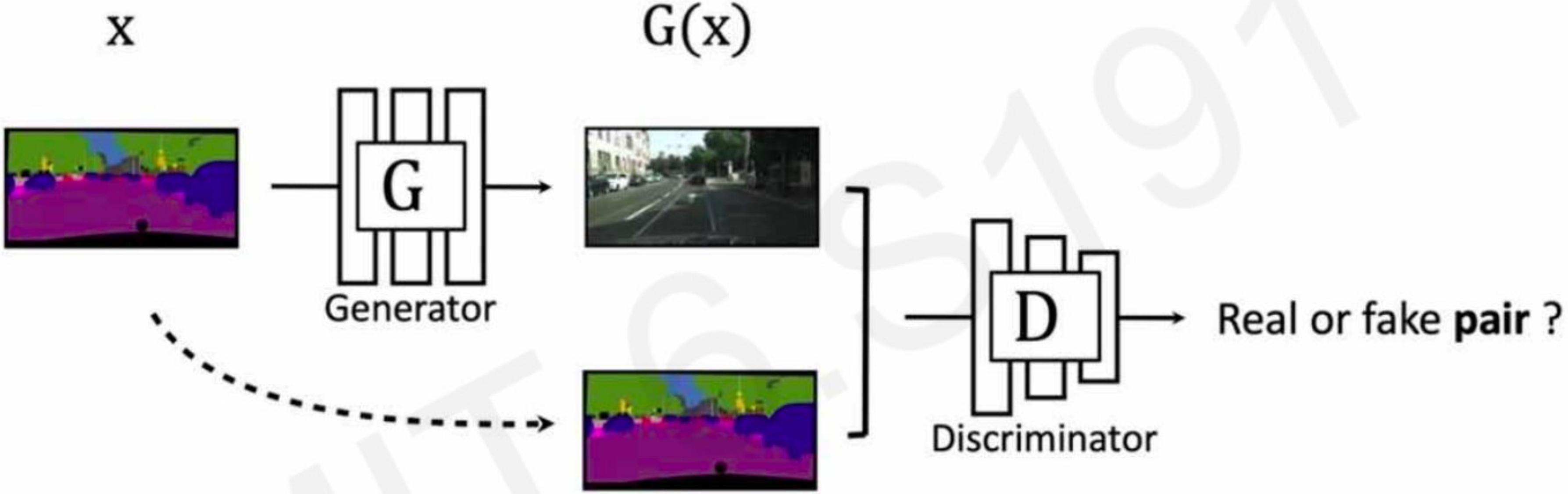


Image to image translation



The discriminator, D, classifies between fake and real **pairs**.
The generator, G, learns to fool the discriminator.

Image to image translation

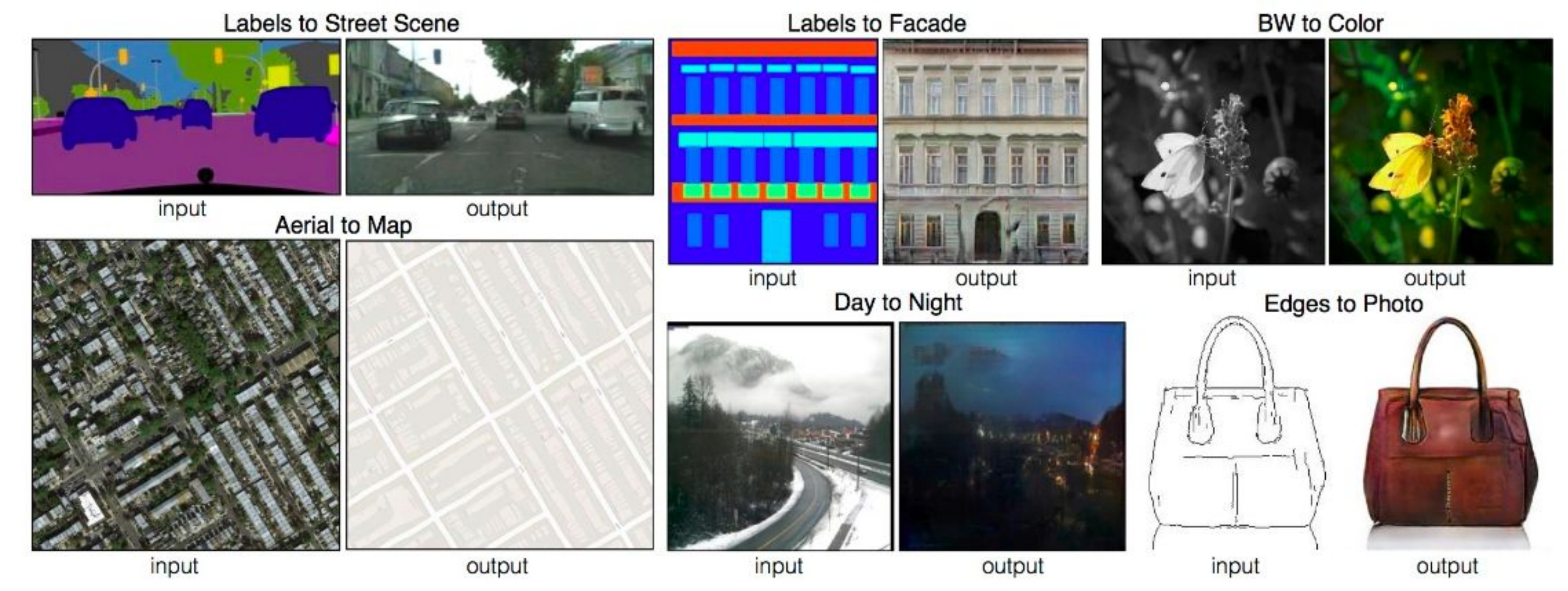


Image taken from https://github.com/phillipi/pix2pix

Super-resolution

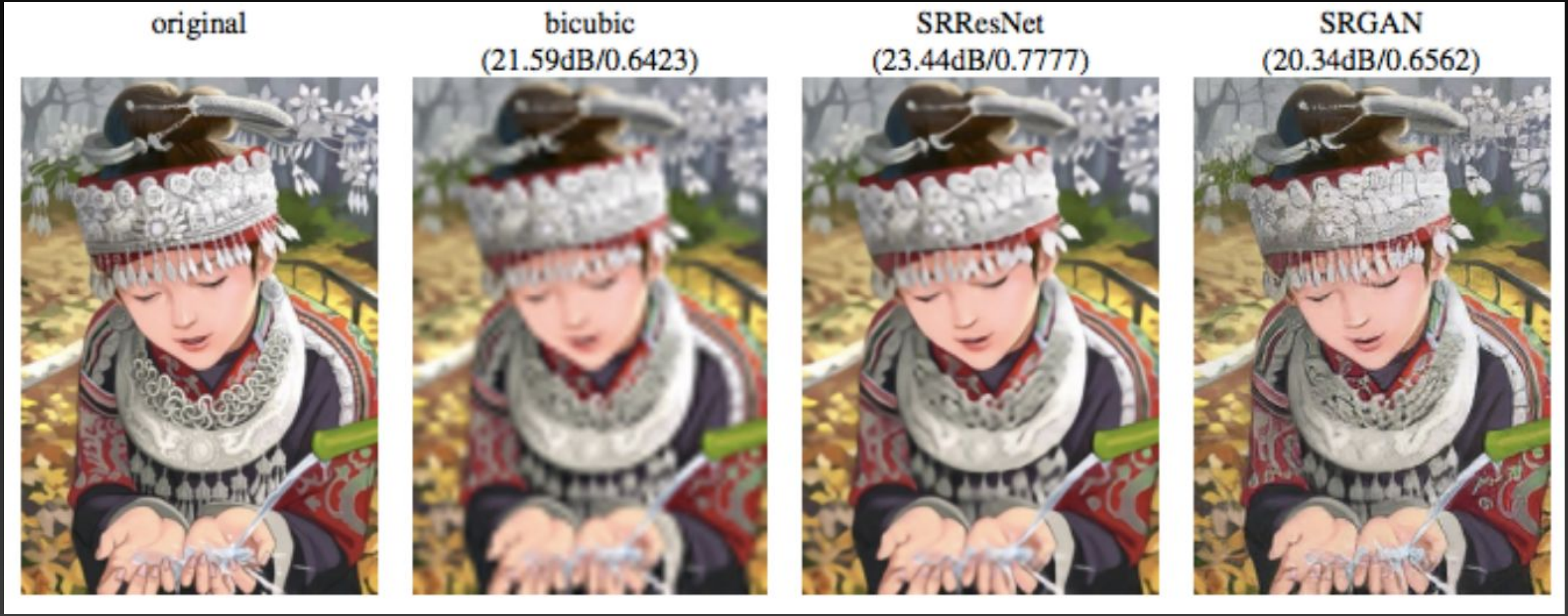
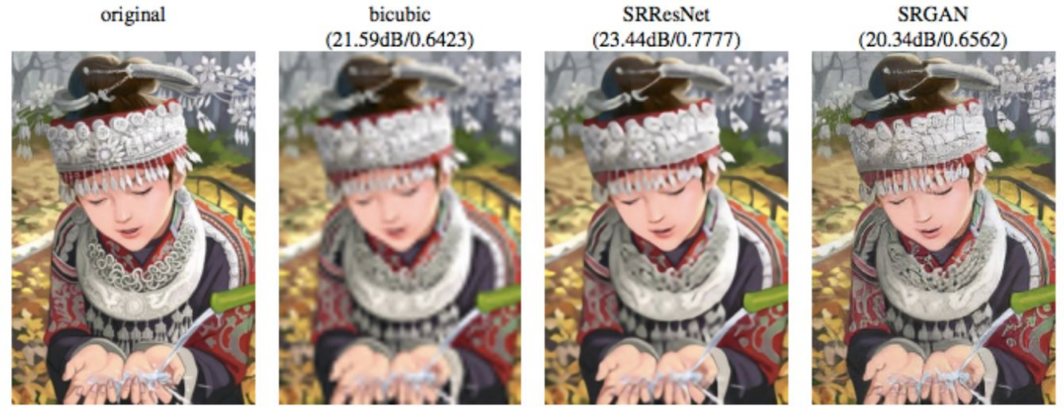


Image taken from https://arxiv.org/abs/1609.04802

Super resolution
- how it works



Conditional GAN.

- img=scaled down IMG,
- z=random noise
- generator maps (img,z) -> G(img) .

Discriminator distinguishes between (img,G(img)) and (img,IMG).

This is the basic trick and it is used in many other settings.

How to train a GAN?

GAN training

- Generate a batch of real images.
- Generate a batch of fake images from a batch of random vectors.
- Make a gradient step for discriminator on the two batches.
- Generate a batch of random vectors.
- Make a gradient step for generator to best fool the discriminator on these vectors.

Note: Gradient based approach so usual caveats apply, e.g. no direct text generation

Loss function

$$L_D = -E_x \log(D(x)) - E_z \log(1-D(G(z)))$$

$$L_G = -L_D$$

Generator and discriminator are “doing the same thing”.

Problem: gets stuck when generator is very bad.

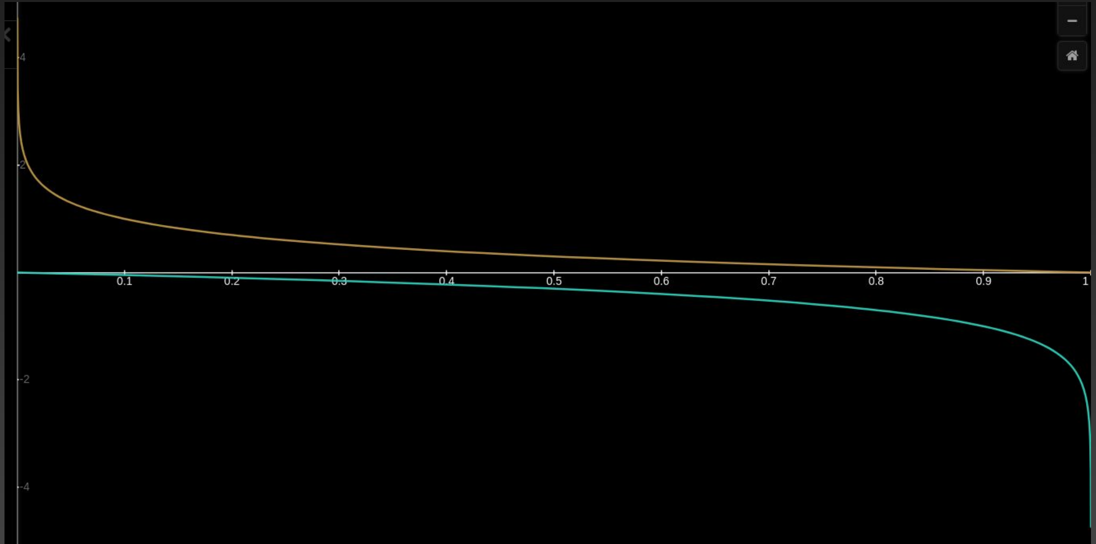
Instead...

Loss function: -log D trick

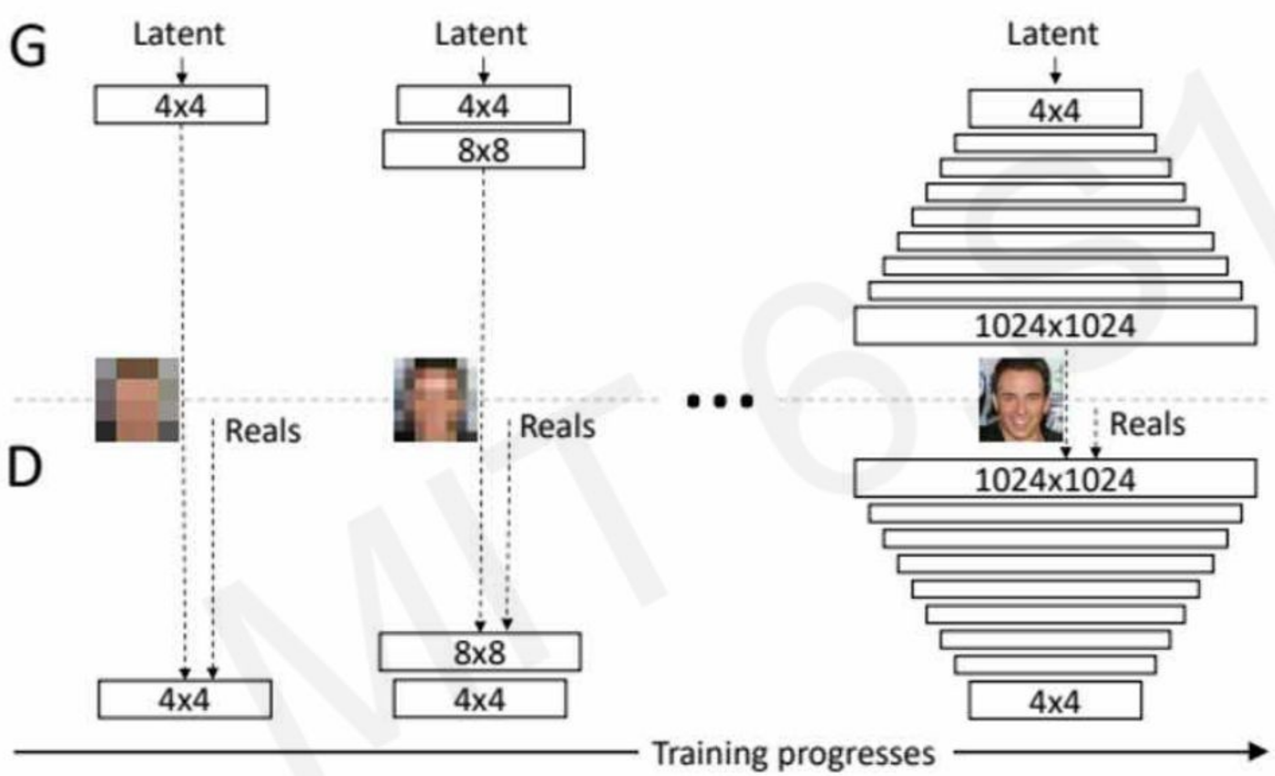
$$L_D = -E_x \log(D(x)) - E_z \log(1-D(G(z)))$$

$$L_G = -E_z \log(D(G(z)))$$

Here, generator should make progress even if very bad.



Progressive growing of GANs



source

Progressive growing of GANs: results

source

Mode collapse

Question: Suppose that the discriminator is fixed. What should the generator do?

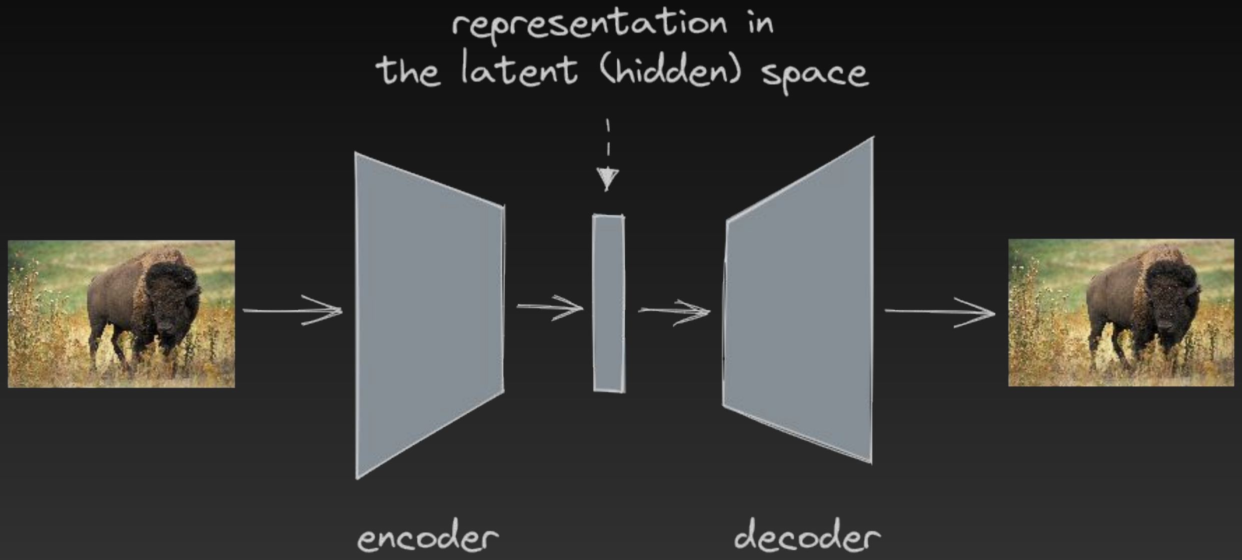
Answer: Output the same answer maximizing the score assigned by discriminator.

This actually happens in practice - mode collapse. Serious problem.

Discriminator eventually learns that this is a bad example, and generator moves to a different one. Generator entropy is lost.

Autoencoders

Autoencoders



- The network is forced to compress information.
- Loss function = reconstruction error.

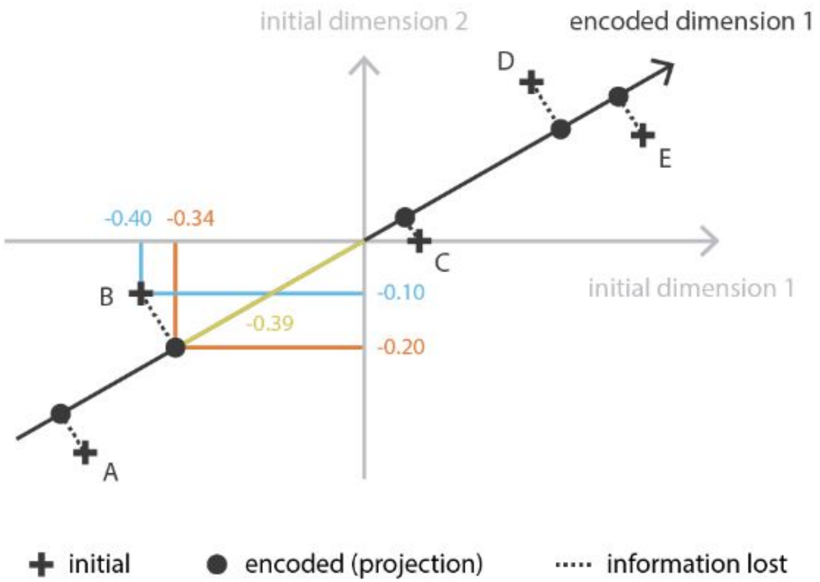
Dimensionality of latent space → reconstruction quality

Autoencoding is a form of compression!
Smaller latent space will force a larger training bottleneck

2D latent space | 5D latent space | Ground Truth

source

PCA as linear autoencoders

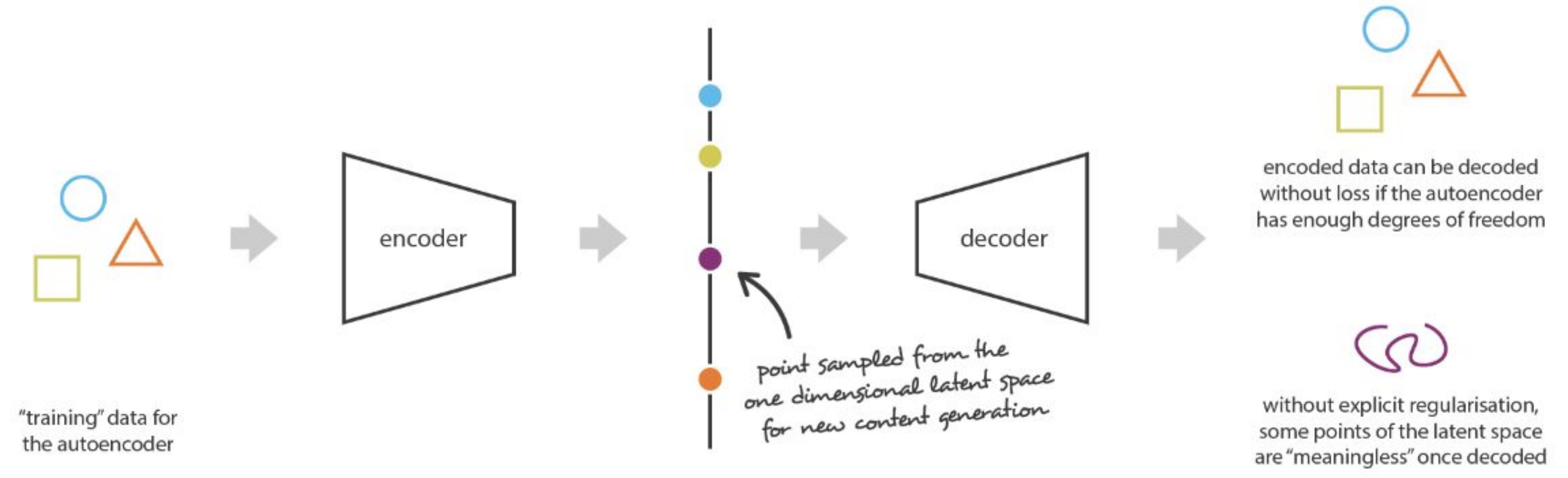


Point	Initial	Encoded	Decoded
A	(-0.50, -0.40)	-0.63	(-0.54, -0.33)
B	(-0.40, -0.10)	-0.39	(-0.34, -0.20)
C	(0.10, 0.00)	0.09	(0.07 0.04)
D	(0.30, 0.30)	0.41	(0.35, 0.21)
E	(0.50, 0.20)	0.53	(0.46, 0.27)

Principal Component Analysis (PCA) is looking for the best linear subspace using linear algebra.

source

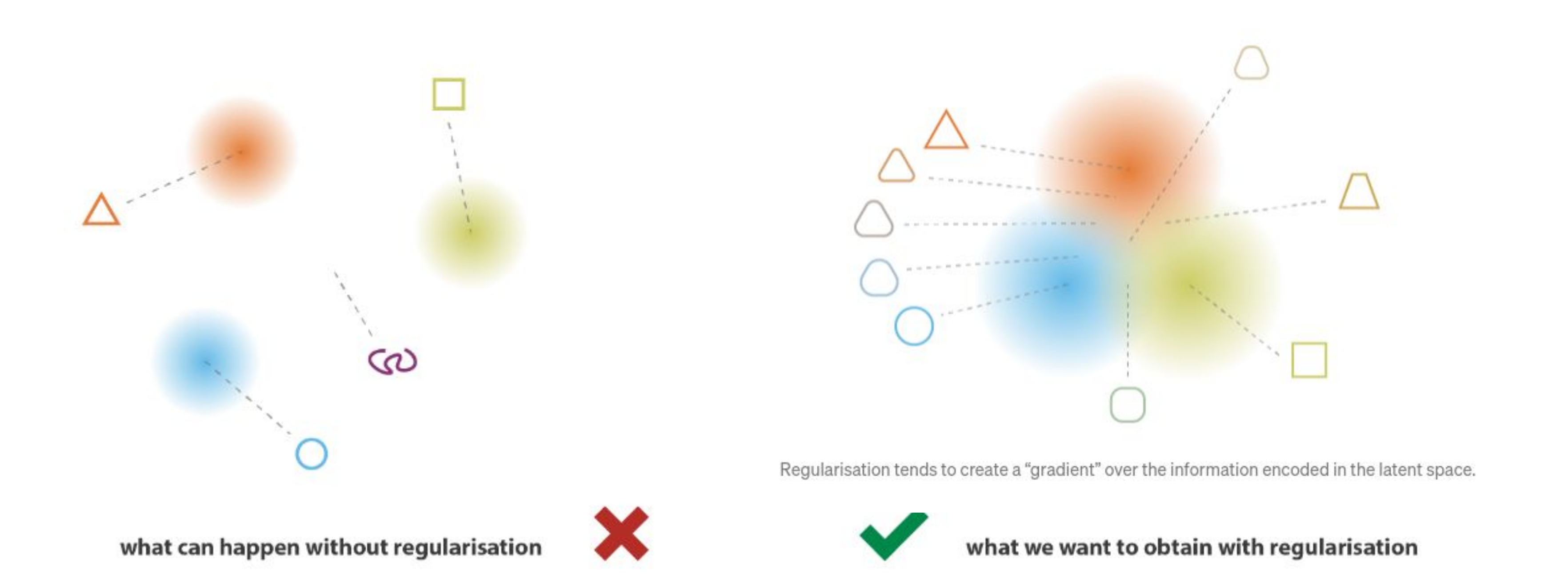
Generating data by sampling from the latent space



Irregular latent space prevent us from using autoencoder for new content generation.

source

Generating data by sampling from the latent space

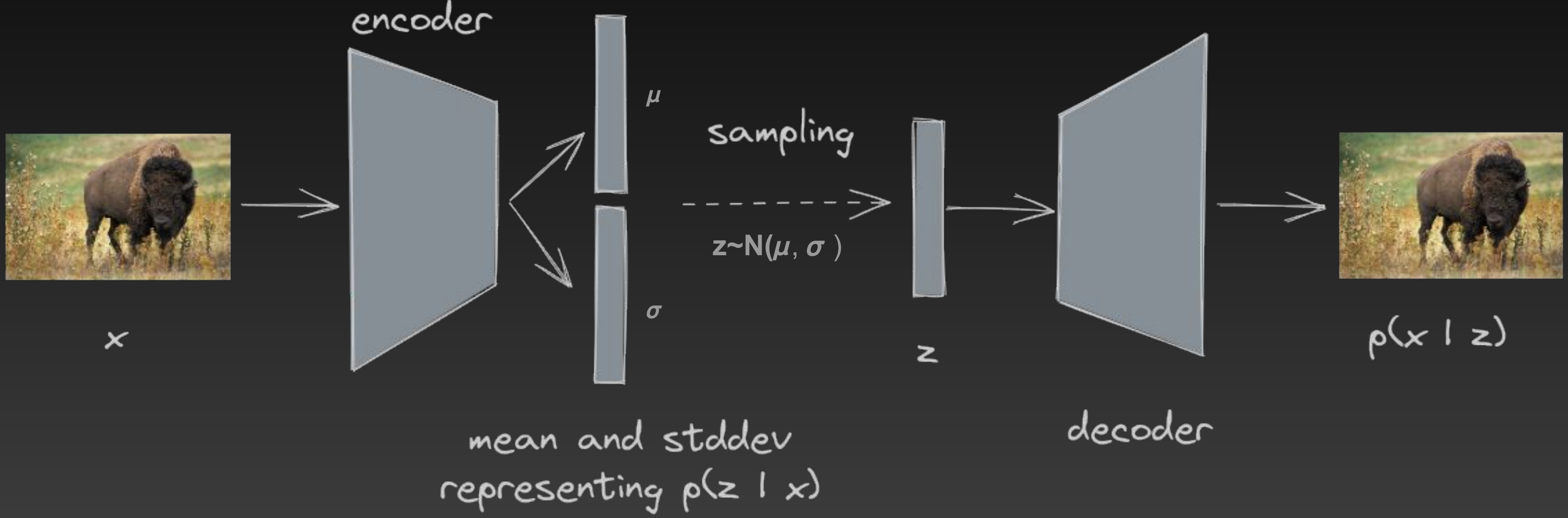


The returned distributions of VAEs have to be regularised to obtain a latent space with good properties.

source

Variational autoencoders

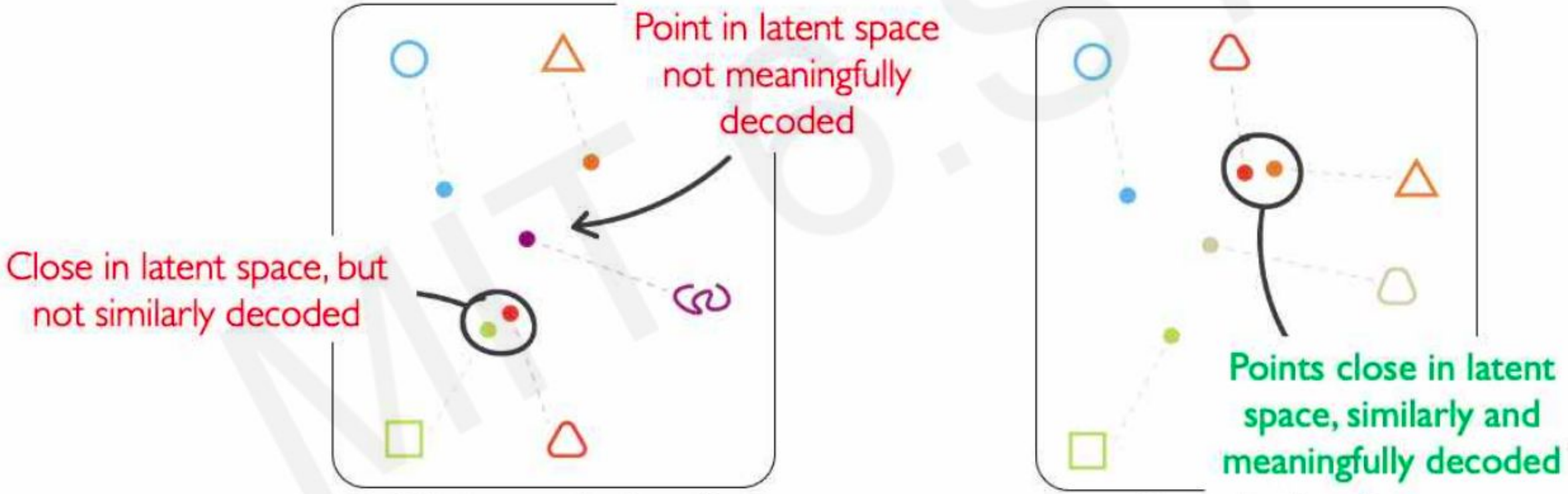
$$\mathcal{L}(\phi, \theta, x) = \text{(reconstruction loss)} + \boxed{\text{(regularization term)}}$$



Intuition on regularization and the Normal prior

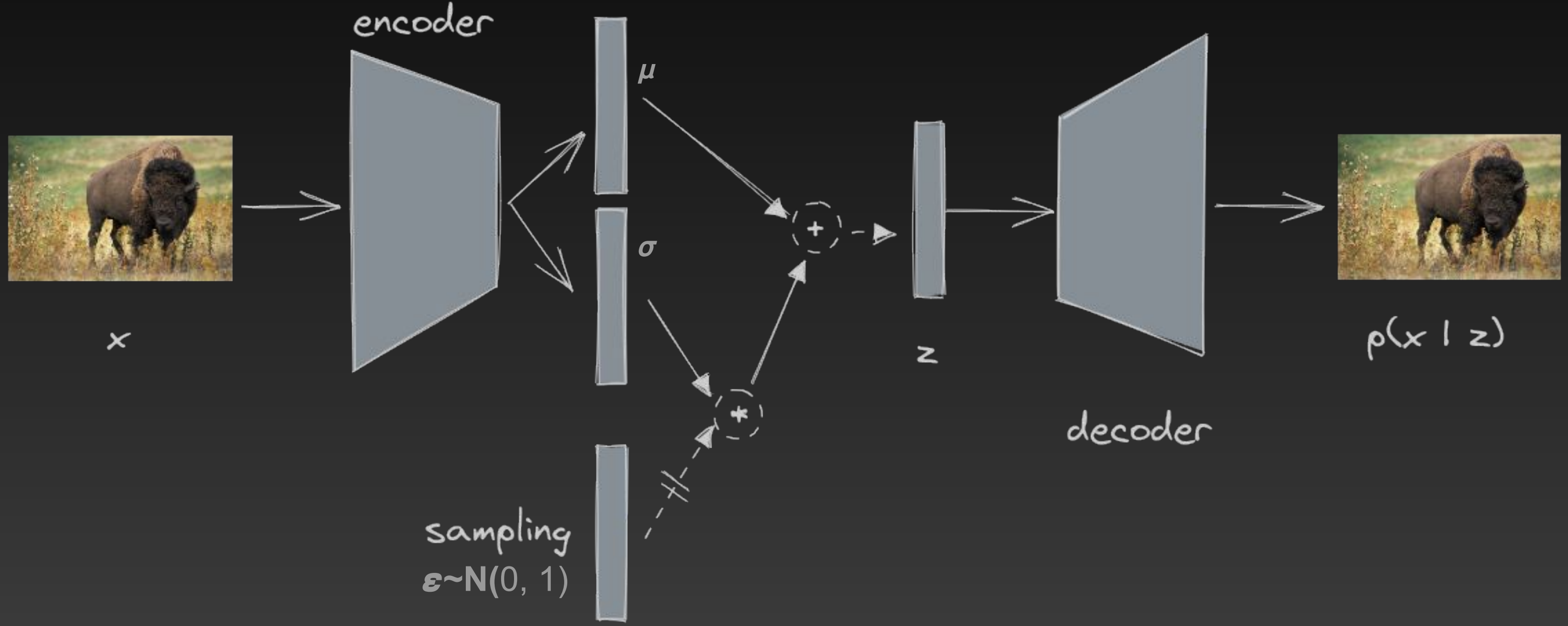
What properties do we want to achieve from regularization?

1. **Continuity:** points that are close in latent space → similar content after decoding
2. **Completeness:** sampling from latent space → "meaningful" content after decoding



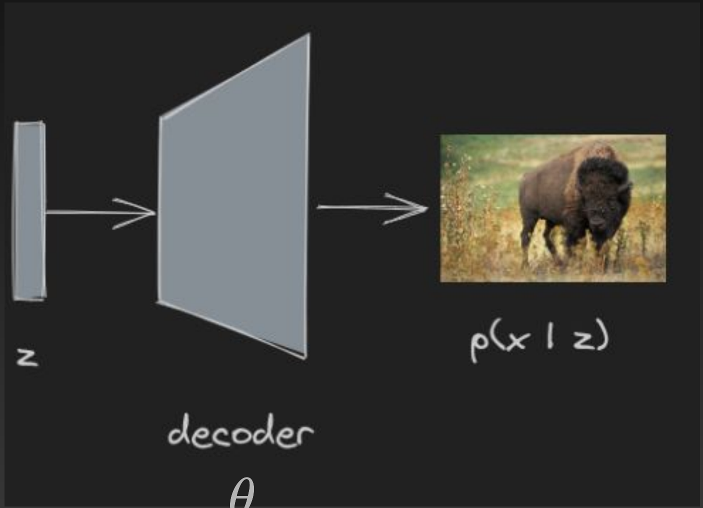
Not regularized **Regularized**

Variational autoencoders - reparametrization trick



VAE - what function to optimize?

- Our generator reads z as input and generates $p_\theta(x|z)$



θ

- Our goal: set parameters θ to maximize the (log) probability of generating the dataset:

$$\sum_i \log p_\theta(x_i) = \sum_i \log \int_z p_\theta(x_i|z)p(z)dz$$

- Integral makes direct optimization intractable.
  - We need a surrogate loss function.

Manipulating objective towards sampling friendly

$$
\begin{aligned}
\log p_\theta(x_i) &= \log \int_z p_\theta(x_i|z)p(z)dz \\
&= \log \int_z \frac{p_\theta(x_i|z)p(z)}{f(z)} f(z)dz \\
&= \log \mathbb{E}_{z\sim f(z)} \left[\frac{p_\theta(x_i|z)p(z)}{f(z)} \right]
\end{aligned}
$$

What f(z) to use?

How to change log expectation to expected log?

Manipulating objective towards sampling friendly

- Take f(z) = $q_\phi(z|x_i)$
- Use Jensen's inequality to get Evidence Lower Bound (ELBO).

$$\begin{aligned}
\log p_\theta(x_i) &= \log \int_z p_\theta(x_i|z)p(z)dz \\
&= \log \int_z \frac{p_\theta(x_i|z)p(z)}{q_\phi(z|x_i)} q_\phi(z|x_i)dz \\
&= \log \mathbb{E}_{z\sim q_\theta(z|x_i)} \left[\frac{p_\theta(x_i|z)p(z)}{q_\phi(z|x_i)} \right] \\
&\geq \mathbb{E}_{z\sim q_\phi(z|x_i)} \left[\log \frac{p_\theta(x_i|z)p(z)}{q_\phi(z|x_i)} \right]
\end{aligned}$$

ELBO - decomposition

$$
\begin{aligned}
\mathrm{ELBO} &= E_{z\sim q_\phi(z|x_i)}\left[\log p_\theta\left(x_i \mid z\right) + \log p(z) - \log q_\phi\left(z \mid x_i\right)\right] \\
&= E_{z\sim q_\phi(z|x_i)}\left[\log p_\theta\left(x_i \mid z\right)\right] + \underbrace{E_{z\sim q_\phi(z|x_i)}\left[\log \frac{p(z)}{q_\phi(z|x_i)}\right]}_{-D_{\mathrm{KL}}\left(q_\phi(z|x_i)\|p(z)\right)} \\
&= E_{z\sim q_\phi(z|x_i)}\left[\log p_\theta\left(x_i \mid z\right)\right] - D_{\mathrm{KL}}\left(q_\phi\left(z \mid x_i\right)\|p(z)\right) \\
&= E_{\epsilon\sim\mathcal{N}(0,1)}\left[\log p_\theta\left(x_i \mid \mu_\phi\left(x_i\right) + \epsilon\sigma_\phi\left(x_i\right)\right)\right] - D_{\mathrm{KL}}\left(q_\phi\left(z \mid x_i\right)\|p(z)\right) \\
&\approx \underbrace{\log p_\theta\left(x_i \mid \mu_\phi\left(x_i\right) + \epsilon\sigma_\phi\left(x_i\right)\right)}_{\text{dec. reconstr. MSE}} - \underbrace{D_{\mathrm{KL}}\left(q_\phi\left(z \mid x_i\right)\|p(z)\right)}_{\text{encoder output reg.}}
\end{aligned}
$$

$$\frac{1}{2}\left[\mu_\phi(x_i)^2 + \sigma_\phi(x_i)^2 - \left(\log\sigma_\phi(x_i)^2 + 1\right)\right]$$

ELBO vs real objective

$$\log p_\theta(x_i) = D_{KL}(q_\phi(z|x_i)||p_\theta(z|x_i)) + \text{ELBO}$$

- The smaller the divergence, the tighter the bound.
- For a fixed p, maximizing ELBO is equivalent to minimizing KL divergence!
- Hence maximizing ELBO makes us at the same time:
  - maximize ELBO (obviously), **and**
  - minimize the difference between ELBO and the real probability.

VAE - additional materials

- Elaborate blogpost Understanding Variational Autoencoders
- Chelsea Finn & Sergey Levine videos
- KL divergence formula derivation: stackexchange

Summary

- Representation, latent space.
  - Probabilistic nature of datasets: latent variable models.
- GAN-s.
- Variational Autoencoders (VAE)

Feedback is a gift

https://tinyurl.com/dnn-2025-11-19