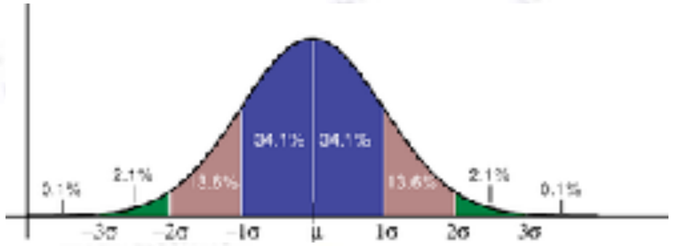
Applied ML

Generative Adversarial Networks

Troels C. Petersen (NBI)



"Statistics is merely a quantisation of common sense - Machine Learning is a sharpening of it!"

Variational Auto-Encoders

Variational AutoEncoders

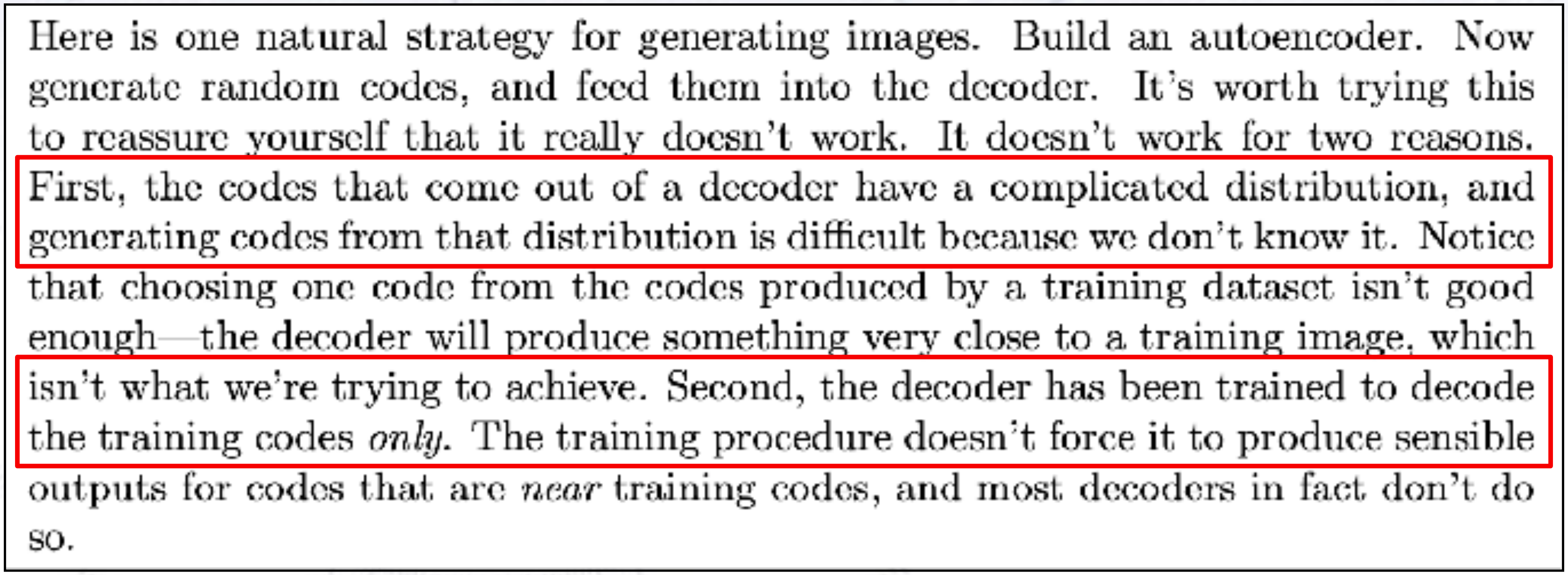
An auto-encoder (AE) is a method (typically based on neural networks) to learn efficient data codings in an unsupervised manner (hence the “auto”). The idea is “old” (80ies) and closely related to (the basis of) Generative Networks.

However, the latent space of AutoEncoders are “complex”, which means that you can not simply choose a “random number” from this space, and expect it to represent something “realistic” when decoded:

Thus, due to non-regularized latent space AE, the decoder can not be used to generate valid input data from vectors sampled from the latent space.

Variational AutoEncoders

An auto-encoder (AE) is a method (typically based on neural networks) to learn efficient data codings in an unsupervised manner (hence the "auto"). The idea is "old" (80ies) and closely related to (the basis of) Generative Networks.



Here is one natural strategy for generating images. Build an autoencoder. Now generate random codes, and feed them into the decoder. It's worth trying this to reassure yourself that it really doesn't work. It doesn't work for two reasons. First, the codes that come out of a decoder have a complicated distribution, and generating codes from that distribution is difficult because we don't know it. Notice that choosing one code from the codes produced by a training dataset isn't good enough—the decoder will produce something very close to a training image, which isn't what we're trying to achieve. Second, the decoder has been trained to decode the training codes *only*. The training procedure doesn't force it to produce sensible outputs for codes that are *near* training codes, and most decoders in fact don't do so.

[David Forsyth, 19.3.1, why AEs are not VAEs]

Variational AutoEncoders

An auto-encoder (AE) is a method (typically based on neural networks) to learn efficient data codings in an unsupervised manner (hence the "auto"). The idea is "old" (80ies) and closely related to (the basis of) Generative Networks.

However, the latent space of AutoEncoders are "complex", which means that you can not simply choose a "random number" from this space, and expect it to represent something "realistic" when decoded:

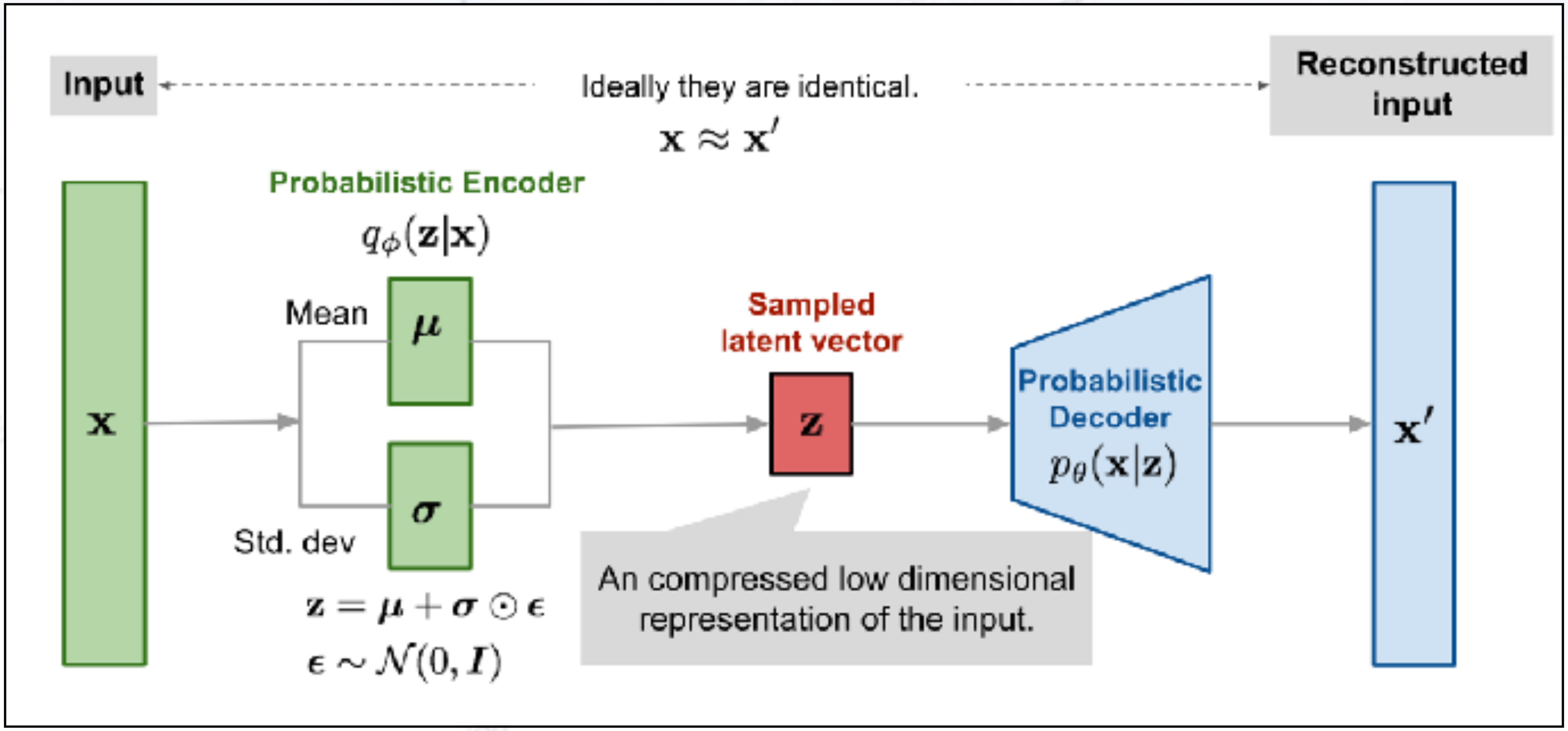
Thus, due to non-regularized latent space AE, the decoder can not be used to generate valid input data from vectors sampled from the latent space.

This requires a special type of AE, **so-called variational autoencoder (VAE)**. Here, the encoder outputs parameters of a pre-defined distribution (multi-dim Gaussian) in the latent space for every input.

The constraint imposed by the VAE ensures that the latent space is **regularised**. This in turn allows one to take a value from the latent space and produce a realistic output.

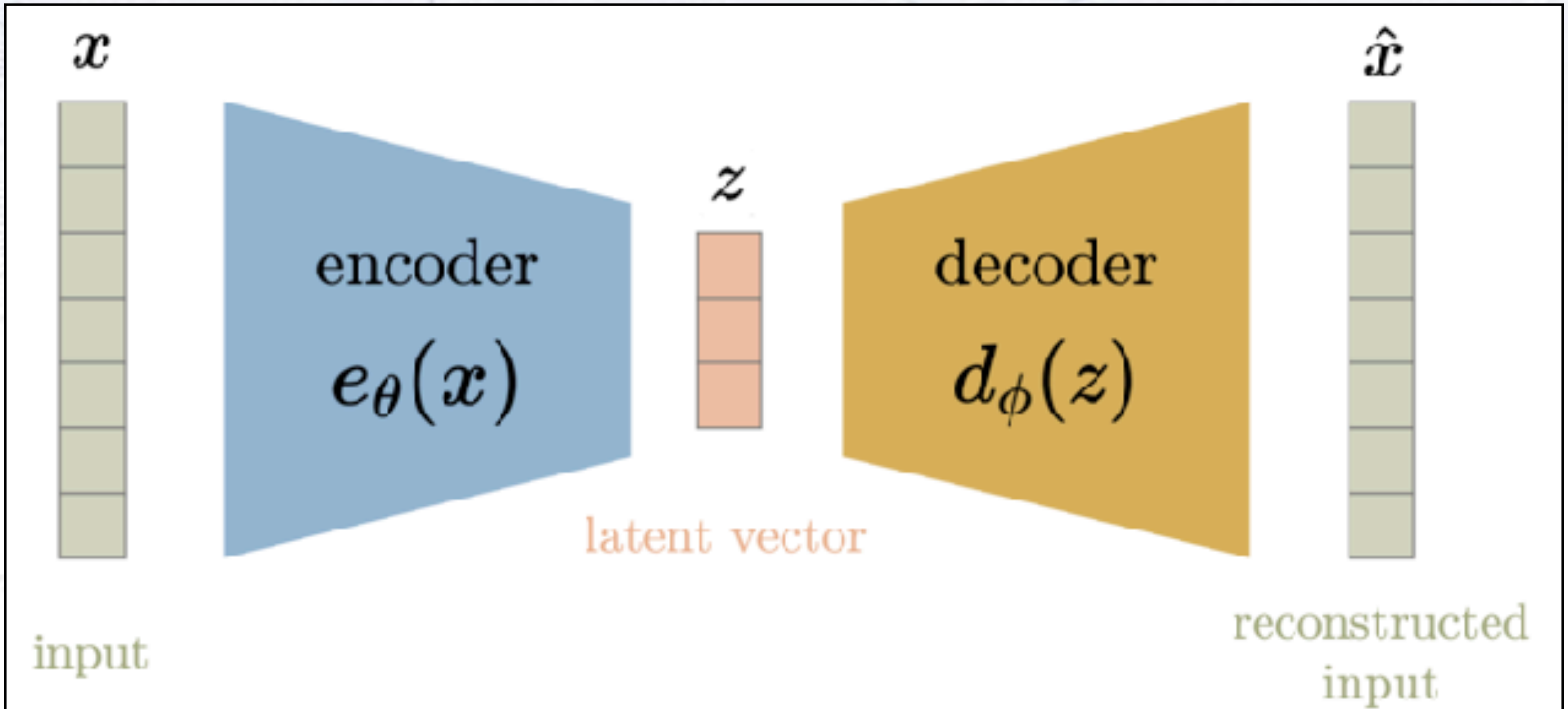
Variational AutoEncoders

A variational autoencoder thus uses a Gaussian-like latent space distribution. It is probabilistic in nature - it produces random cases close (i.e. ε away from) to the original.


Input
Ideally they are identical.
$\mathbf{x} \approx \mathbf{x}'$
Reconstructed input
Probabilistic Encoder
$q_\phi(\mathbf{z}|\mathbf{x})$
Mean
$\boldsymbol{\mu}$
Std. dev
$\boldsymbol{\sigma}$
$\mathbf{x}$
Sampled latent vector
$\mathbf{z}$
Probabilistic Decoder
$p_\theta(\mathbf{x}|\mathbf{z})$
$\mathbf{x}'$
$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$
$\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I})$
An compressed low dimensional representation of the input.

Variational AutoEncoders

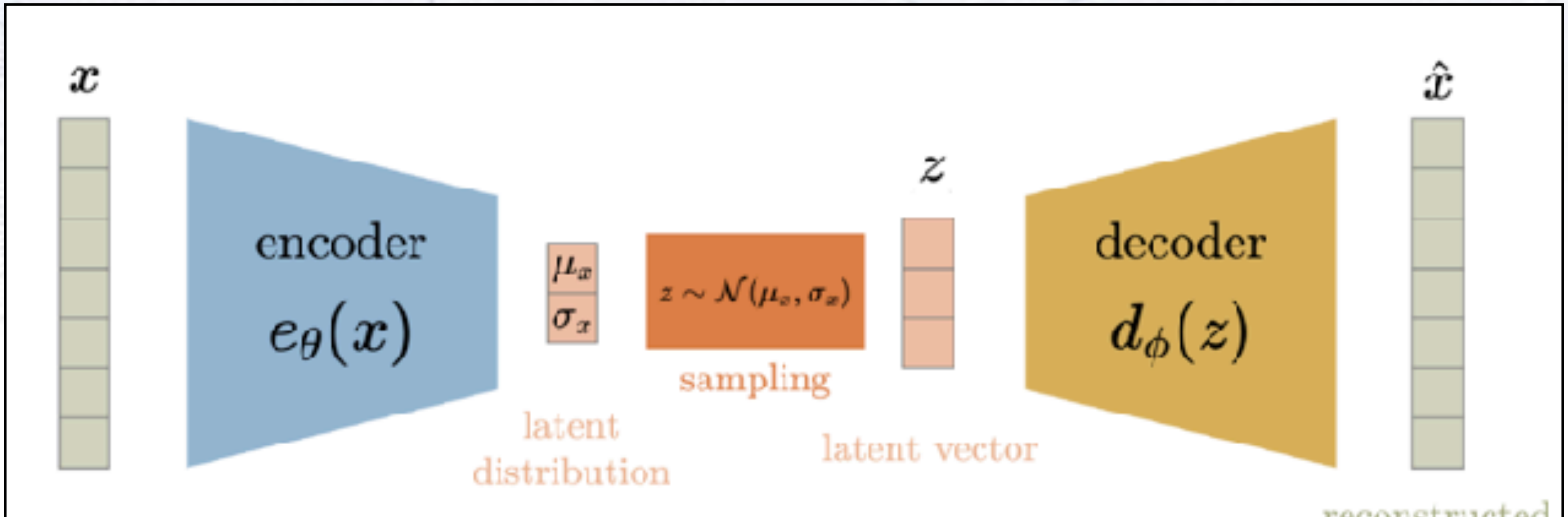
A VAE thus uses a Gaussian-like latent space distribution. It is probabilistic in nature - it produces random cases close (i.e. ε away from) to the original. This is achieved by a "smart" loss function with the Kullback–Leibler (KL) divergence.



$$loss = \|x - \hat{x}\|_2 = \|x - d_\phi(z)\|_2 = \|x - d_\phi(e_\theta(x))\|_2$$

Variational AutoEncoders

A VAE thus uses a Gaussian-like latent space distribution. It is probabilistic in nature - it produces random cases close (i.e. ε away from) to the original. This is achieved by a "smart" loss function with the Kullback–Leibler (KL) divergence.



$$\text{reconstruction loss} = \|x - \hat{x}\|_2 = \|x - d_\phi(z)\|_2 = \|x - d_\phi(\mu_x + \sigma_x \epsilon)\|_2$$

$$\mu_x, \sigma_x = e_\theta(x), \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

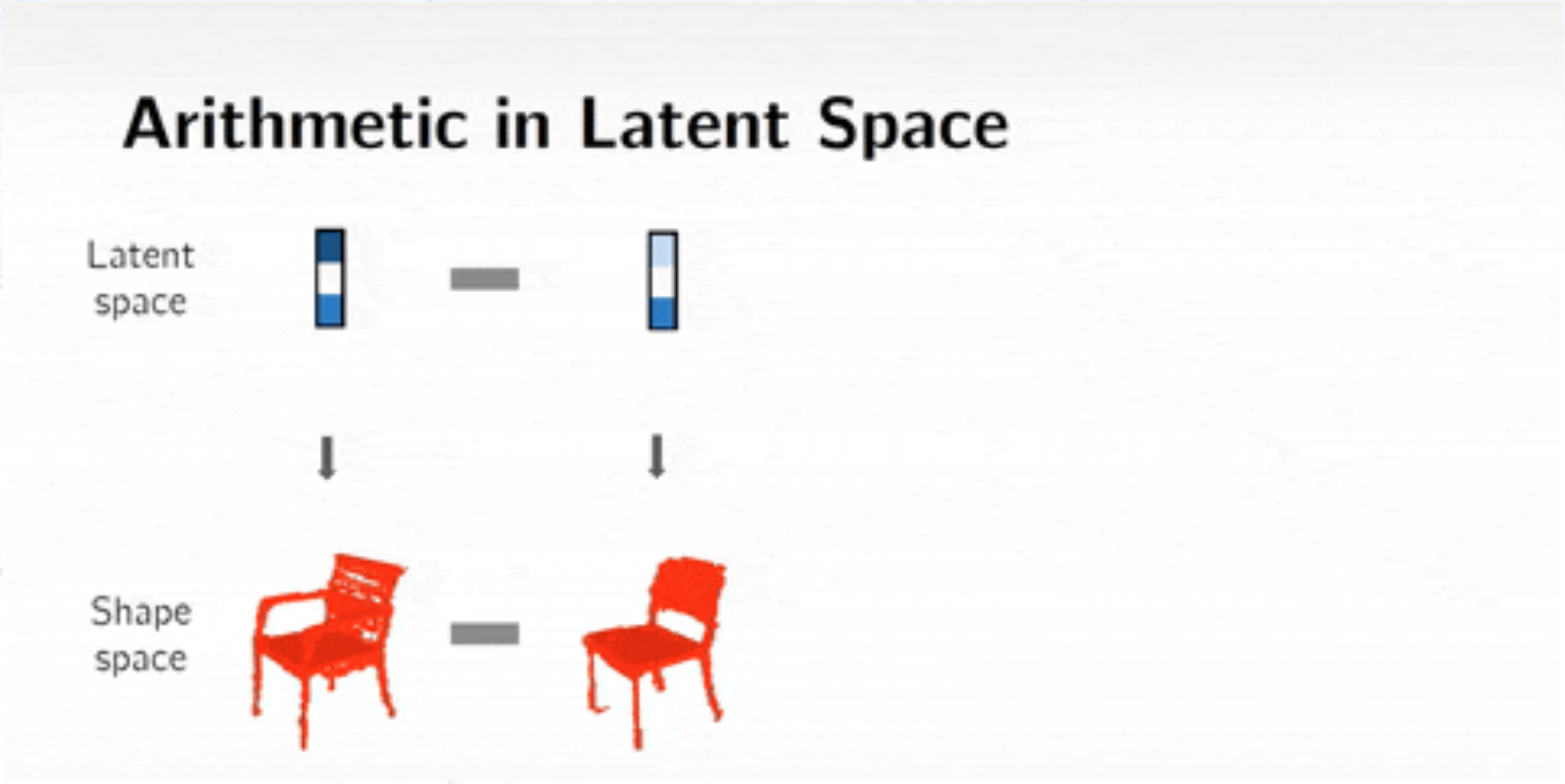
$$\textit{similarity loss} = \textit{KL Divergence} = D_{KL}(\mathcal{N}(\mu_x, \sigma_x) \parallel \mathcal{N}(\mathbf{0}, \mathbf{I}))$$

$$\textit{loss} = \textit{reconstruction loss} + \textit{similarity loss}$$

Latent space illustration

The below animation shows how VAE latent spaces are a simplified representation of the more complex objects, containing the main features of these.

For this reason, one can do arithmetics (typically interpolate) between the inputs:



Latent space illustration

The below animation shows how VAE latent spaces are a simplified representation of the more complex objects, containing the main features of these.

For this reason, one can do arithmetics (typically interpolate) between the inputs:

Interpolation in Latent Space

Generative Adversarial Networks

Generative Adversarial Networks

Invented (partly) by Ian Goodfellow in 2014, Generative Adversarial Networks (GANs) is a method for learning how to produce new (simulated) datasets from existing data.

The basic idea is, that **two networks “compete” against each other**:
- **Generative Network**: Produces new data trying to make it match the original.
- **Adversarial (Discriminatory) Network**: Tries to classify original and new data.

Typically, the generator is a de-convolutional NN, while the discriminating (adversarial) is convolutional NN.

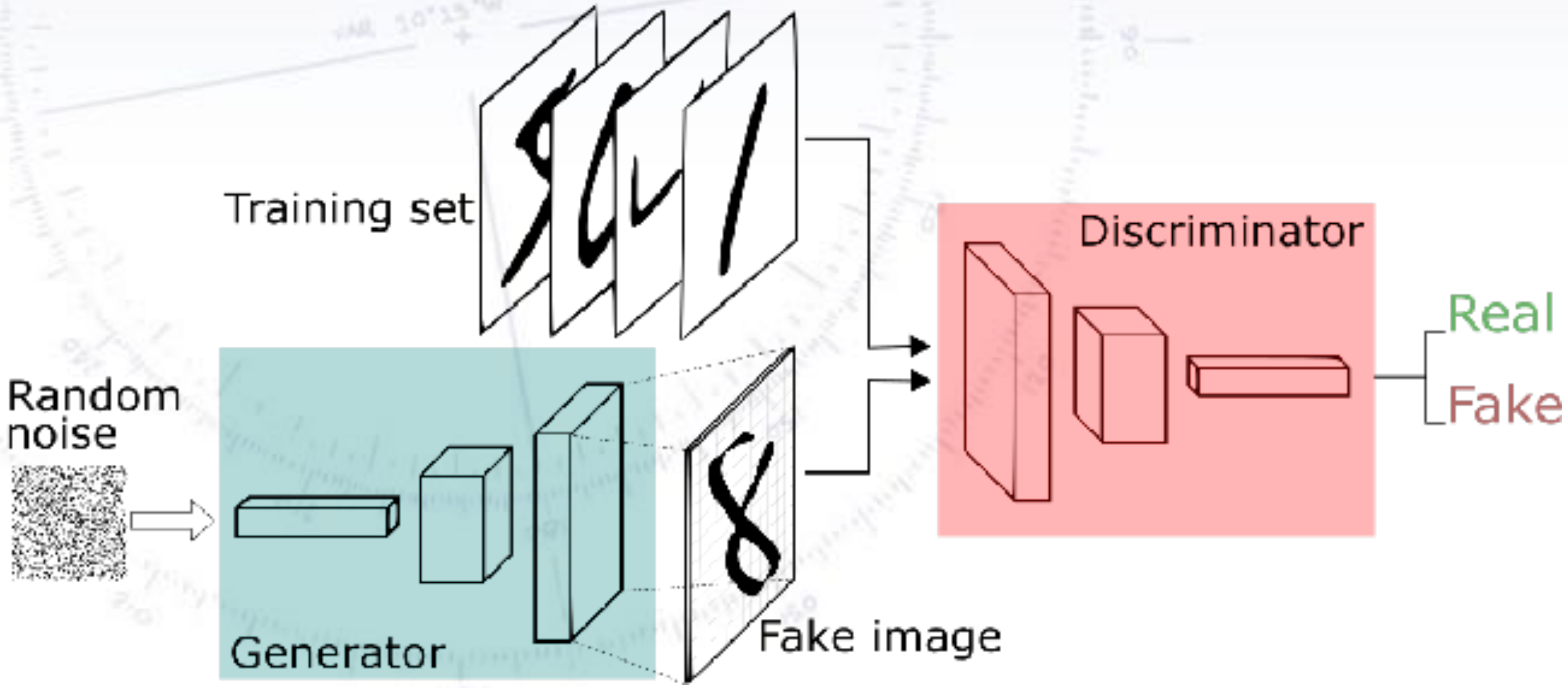
The concept is related to (Variational) Auto-Encoders.

“The coolest idea in machine learning in the last twenty years”
[Yann LeCun, French computer scientist]

GAN drawing

Imagine that you want to write numbers that looks like hand writing.

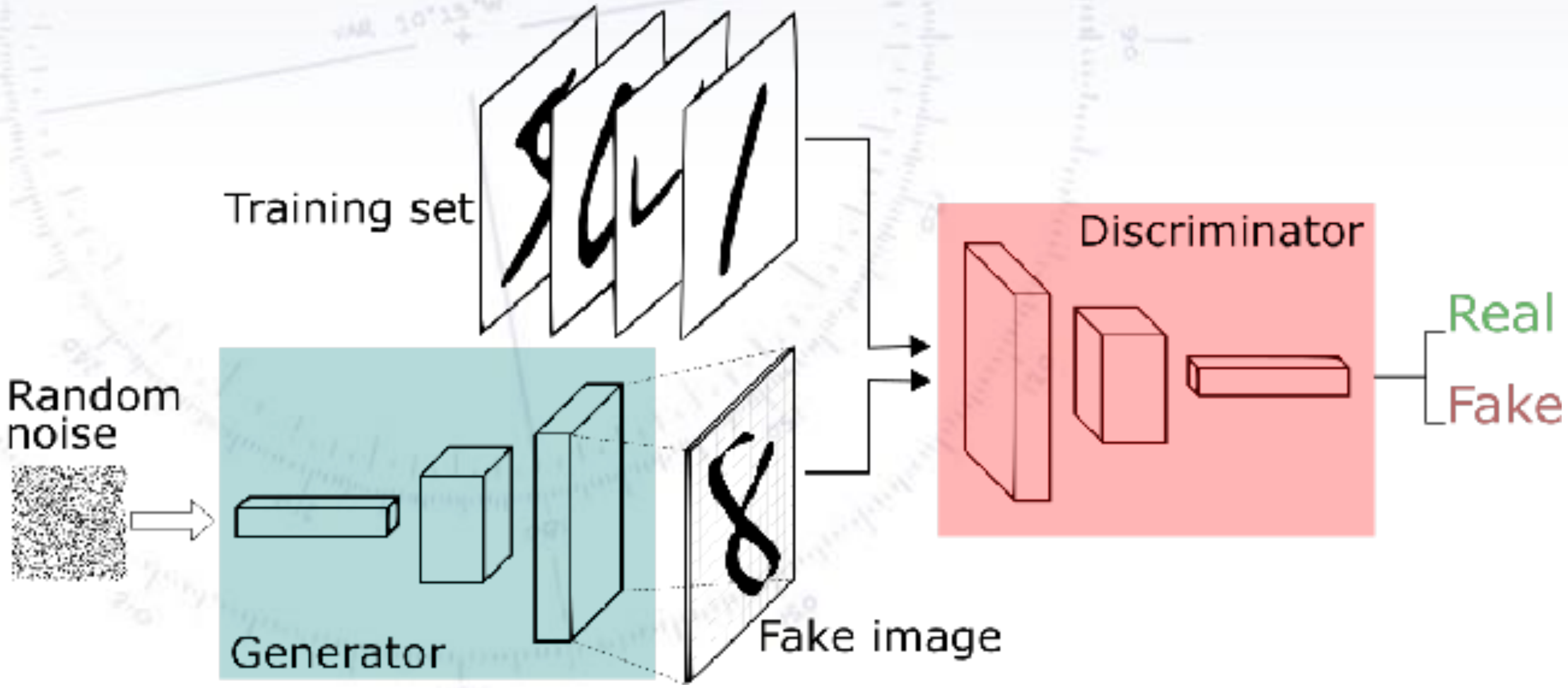
Given a large training set, you can ask you GAN to produce numbers. At first it will do poorly, but as it is "punished" by the discriminator, it improves, and at the end it might be able to produce numbers of **equal quality to real data**:



GAN drawing

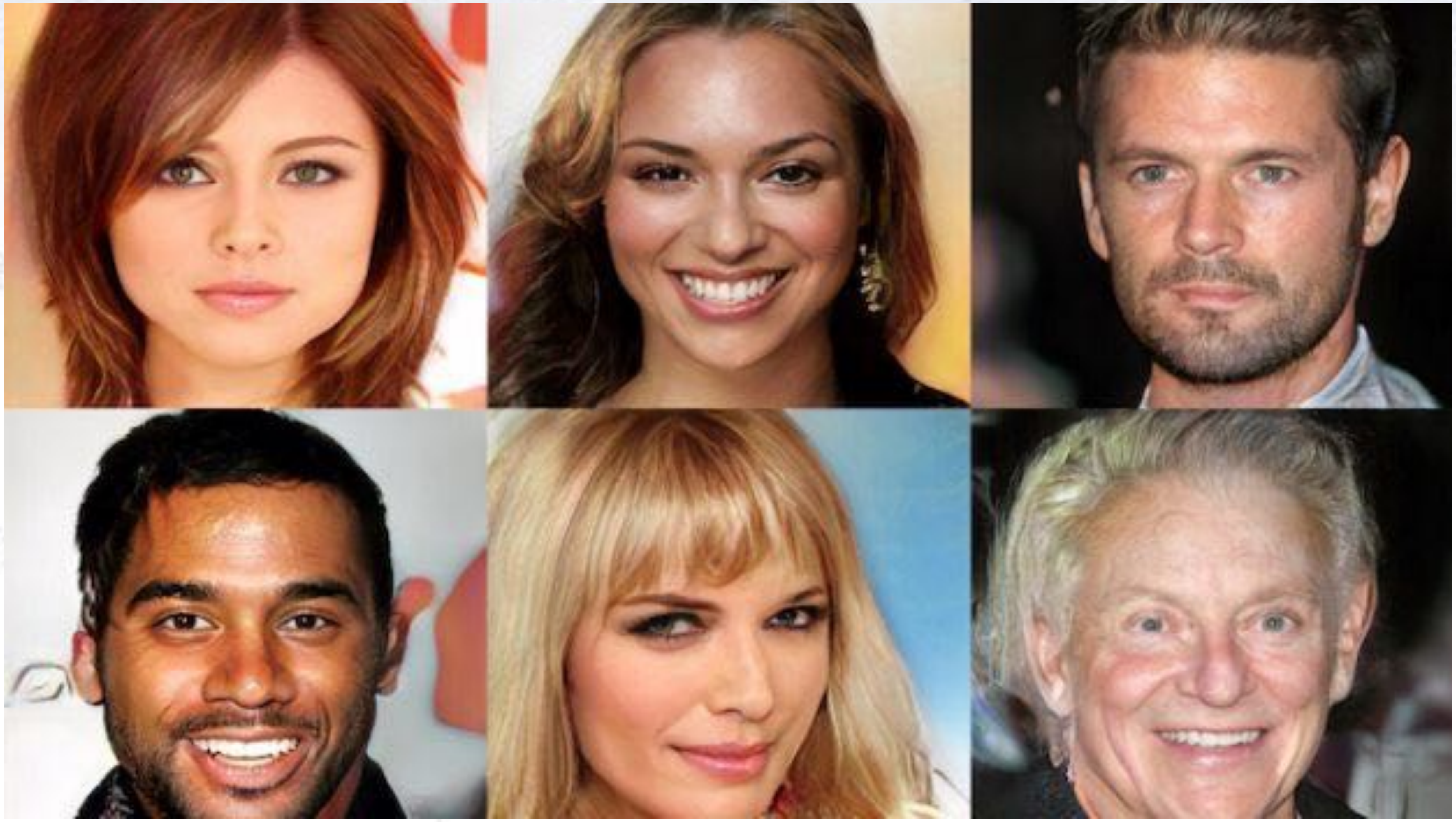
The discriminator/adversarial can also be seen as an addition to loss function, penalising (with λ) an ability to see differences between real and fake:

$$\text{Loss} = \text{Loss} + \lambda \cdot L_{\text{Adversarial}}$$



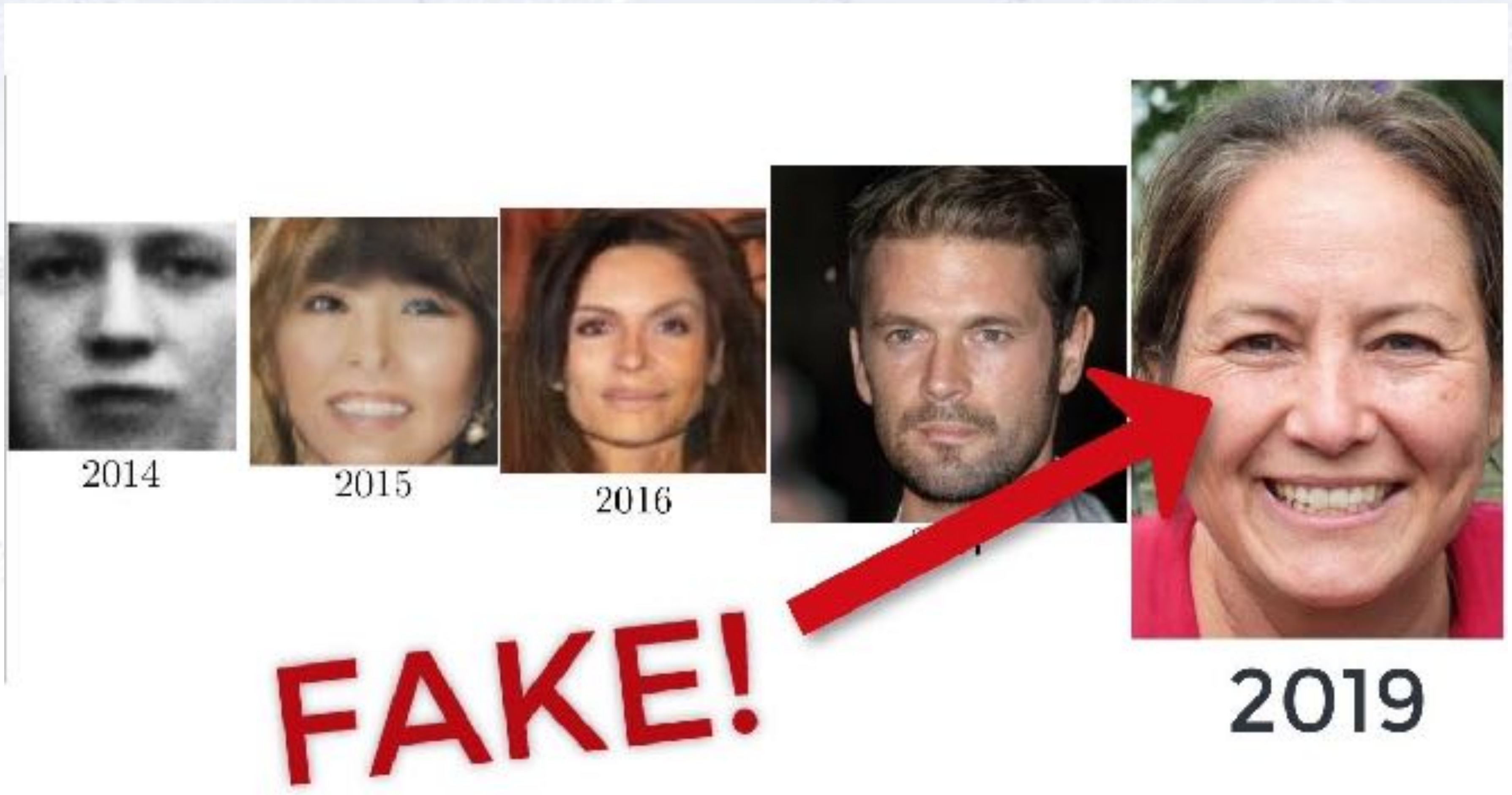
GANs producing face images

In 2017, Nvidia published the result of their “AI” GANs for producing celebrity faces. There is of course a lot of training data… here are the results:



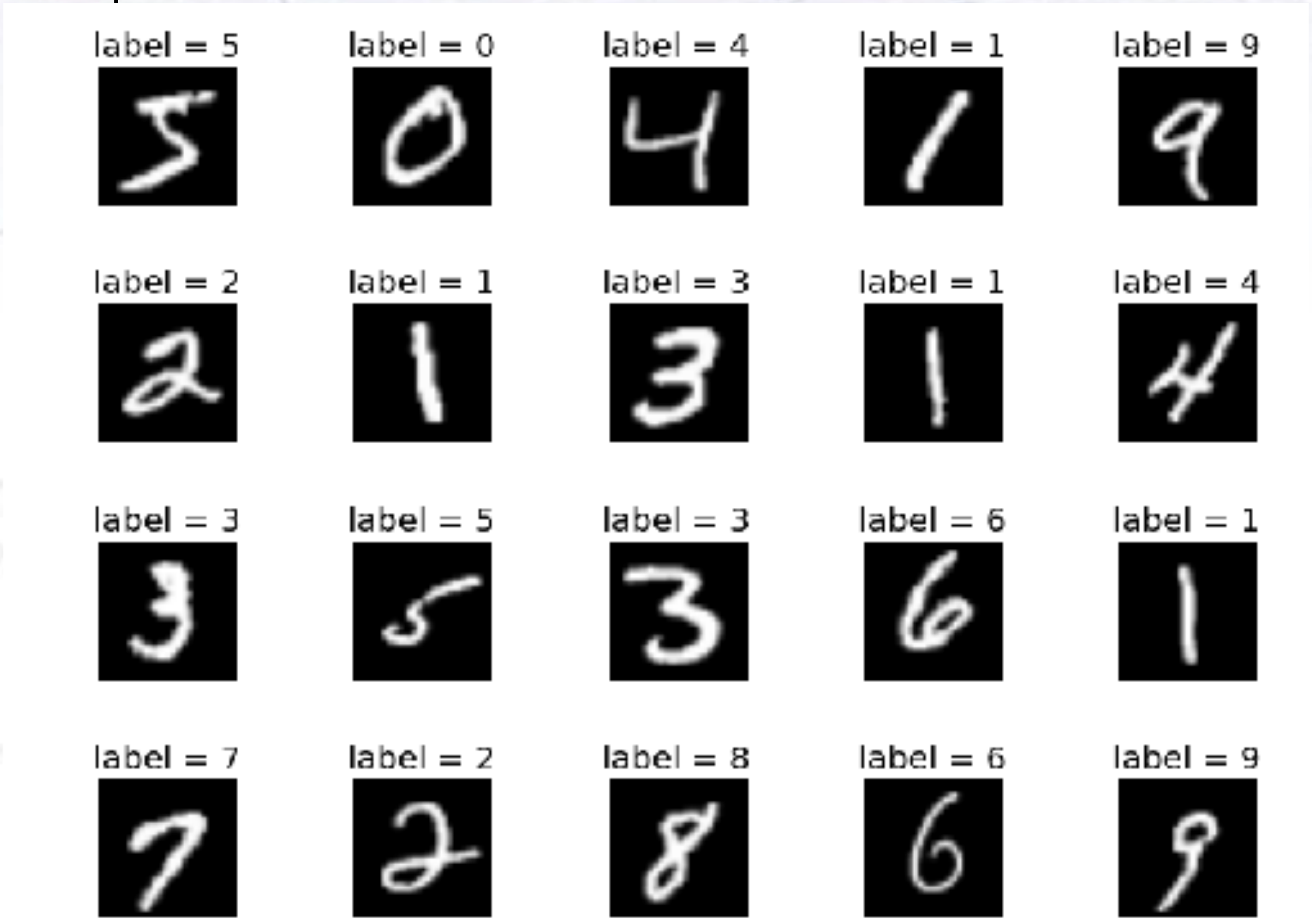
Evolution in facial GANs

There is quiet a fast evolution in GANs, and their ability to produce realistic results….



MNist data: Handwritten numbers

A "famous case" has been hand written numbers. The data consists of 28x28 gray scale images of numbers. While that spans a large space, the latent space is probably (surely!) much smaller, as far from all combinations of pixels and intensities are present.



MNist data: Handwritten numbers

A “famous case” has been hand written numbers. The data co 8x28 gray scale images of numbers. While that spans a large sp is probably (surely!) much smaller, as far fro d intensities are present.

label = 5
label = 6
label = 1
label = 7
label = 2
label = 8
label = 6
label = 9

With GANs, you can produce handwritten letters again - sort of!