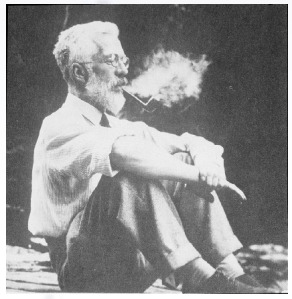
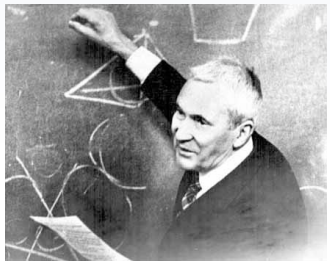
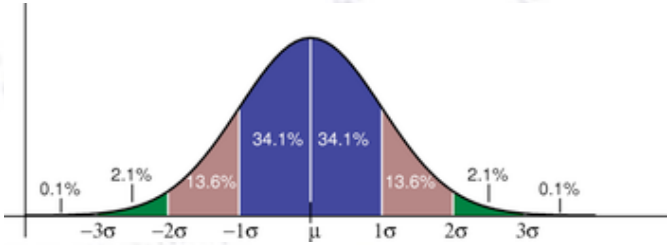
Applied ML

AutoEncoders



Troels C. Petersen (NBI)

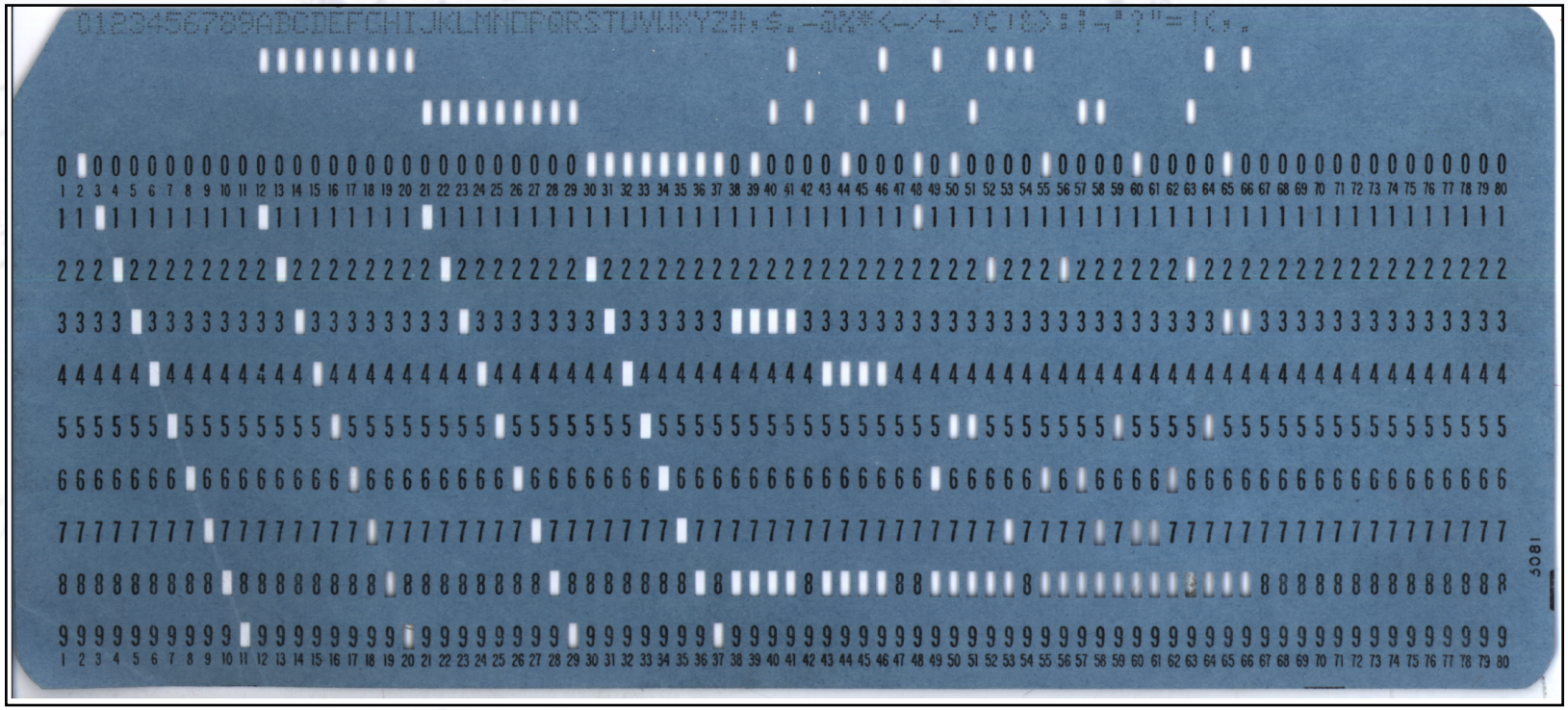


"Statistics is merely a quantisation of common sense - Machine Learning is a sharpening of it!"

Encoders

An **encoder** is a network that can **take a signal and produce a code**!

Typically, the code is a description of the signal, which could be images, sound, etc. The code is usually "smaller" than the signal, not unlike a summary. An image classifier is an example of an encoder.

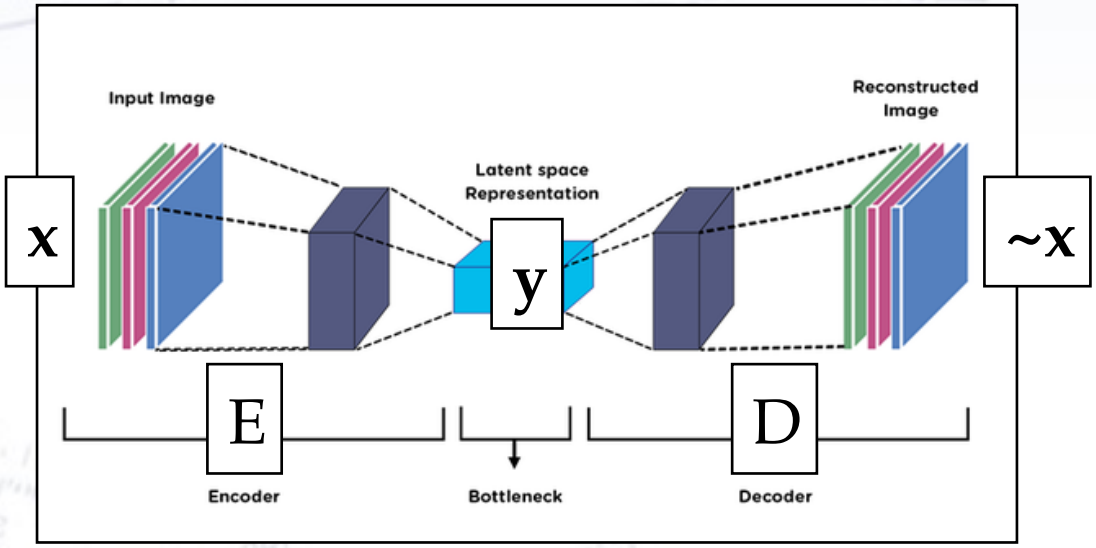


Hollerith 80-column punch card (1950s)

AutoEncoders

An **AutoEncoder** (AE) is a **coupled pair of encoder and decoder**. The encoder maps signals into code, and the decoder reconstructs the original signal from those codes.

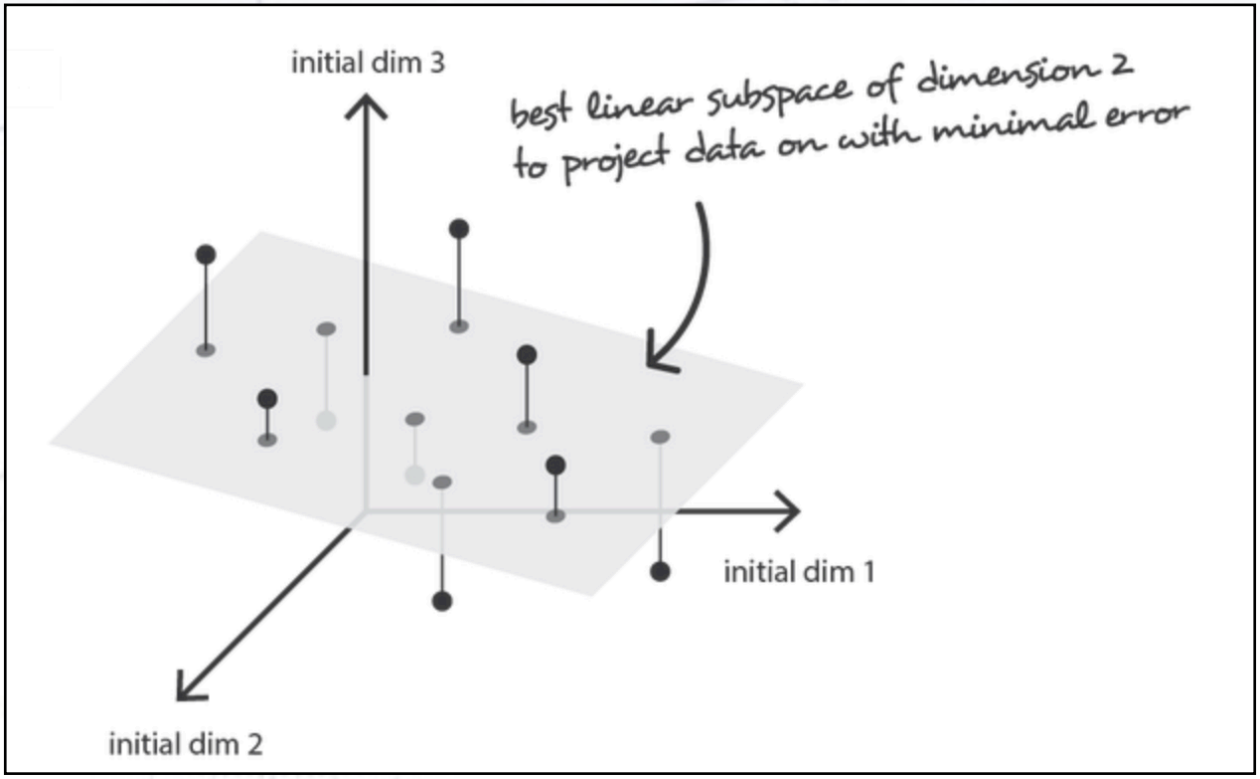
The pair is trained to have the most accurate reconstruction: If you give a signal **x** to an encoder E to get **y** = E(**x**), then the decoder D should ensure that D(**y**) is close to **x**.



One application is unsupervised feature learning, where it tries to construct a useful feature set from a set of unlabelled images. We could use the code produced as a source of features.

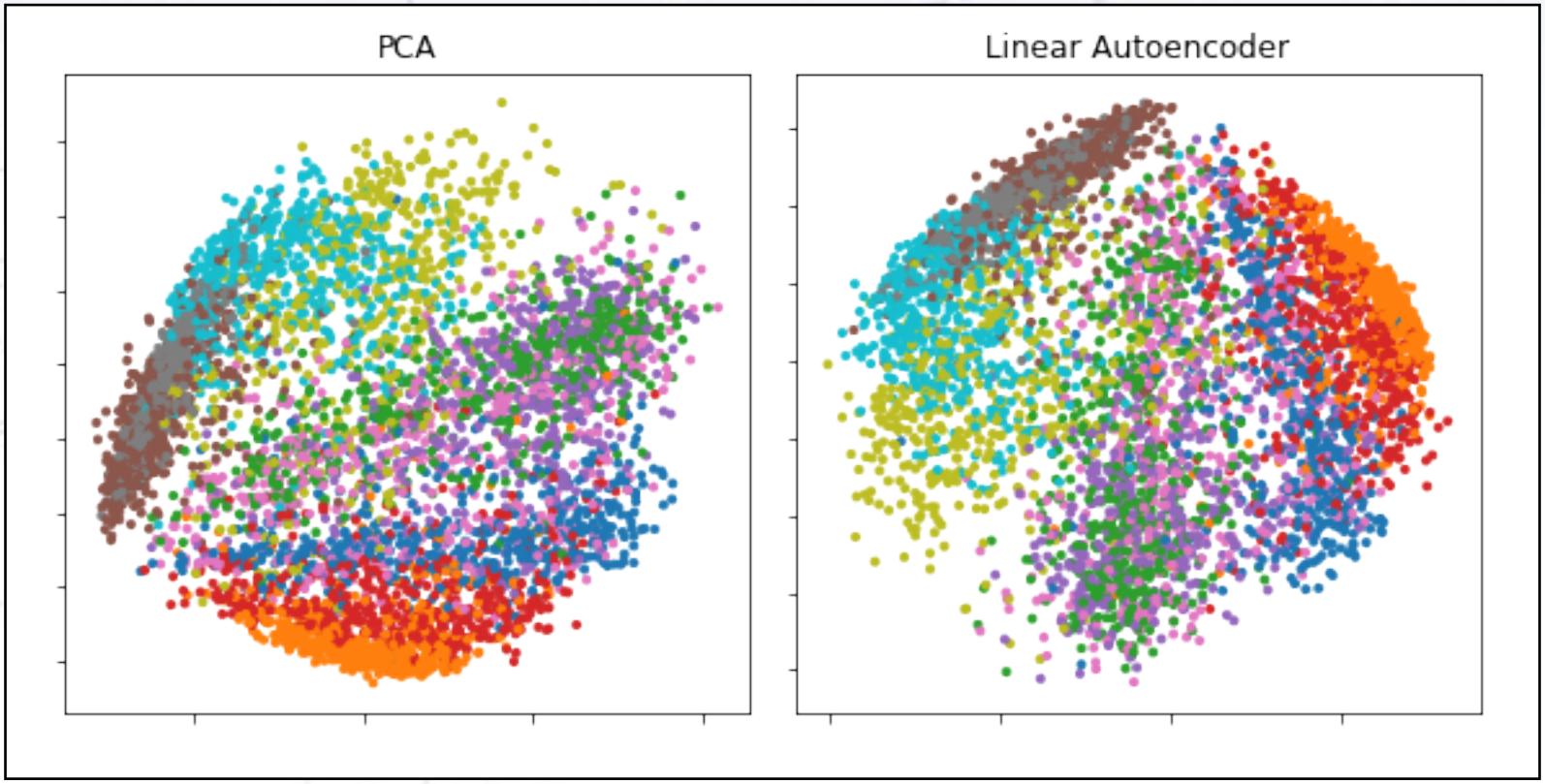
PCA as an autoencoder

A PCA is a linear AE. One can project a higher dimension down on (fewer) principle components (encoding) and then "reconstruct" the original data from the latent space, by choosing the low dimensional points in the original dimensionality.



PCA as an autoencoder

A PCA is a linear AE. One can project a higher dimension down on (fewer) principle components (encoding) and then "reconstruct" the original data from the latent space, by choosing the low dimensional points in the original dimensionality.



Usage of AE

AEs are used for many things, such as:

- Unsupervised learning (e.g. clustering) on images, sound, graphs, etc.
- Compression (with loss!) of e.g images
- De-noising and inpainting images
- Anomaly detection
- Training on large dataset with few labels

Most AEs are CNN based and produced for images. However, the AE concept is more general, and applies to anything, that can be passed through an NN.

Usage of AE

AEs are used for many things, such as:

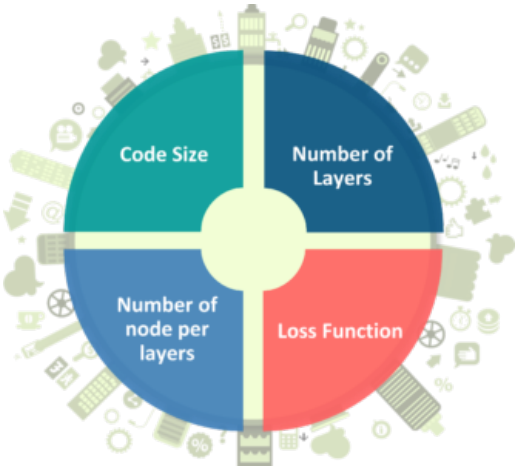
- Unsupervised learning (e.g. clustering) on images, sound, graphs, etc.
- Compression (with loss!) of e.g images
- De-noising and inpainting images
- Anomaly detection
- Training on large dataset with few labels

Most AEs are CNN based and produced for images. However, the AE concept is more general, and applies to anything, that can be passed through an NN.

The most central hyperparameters to consider are:

- Size of the latent space (code)
- Architecture of NN (layers and nodes)
- Loss function

As we shall see, these HPs to some extend determine what type of AE you're making.



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 spaces 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 spaces 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:

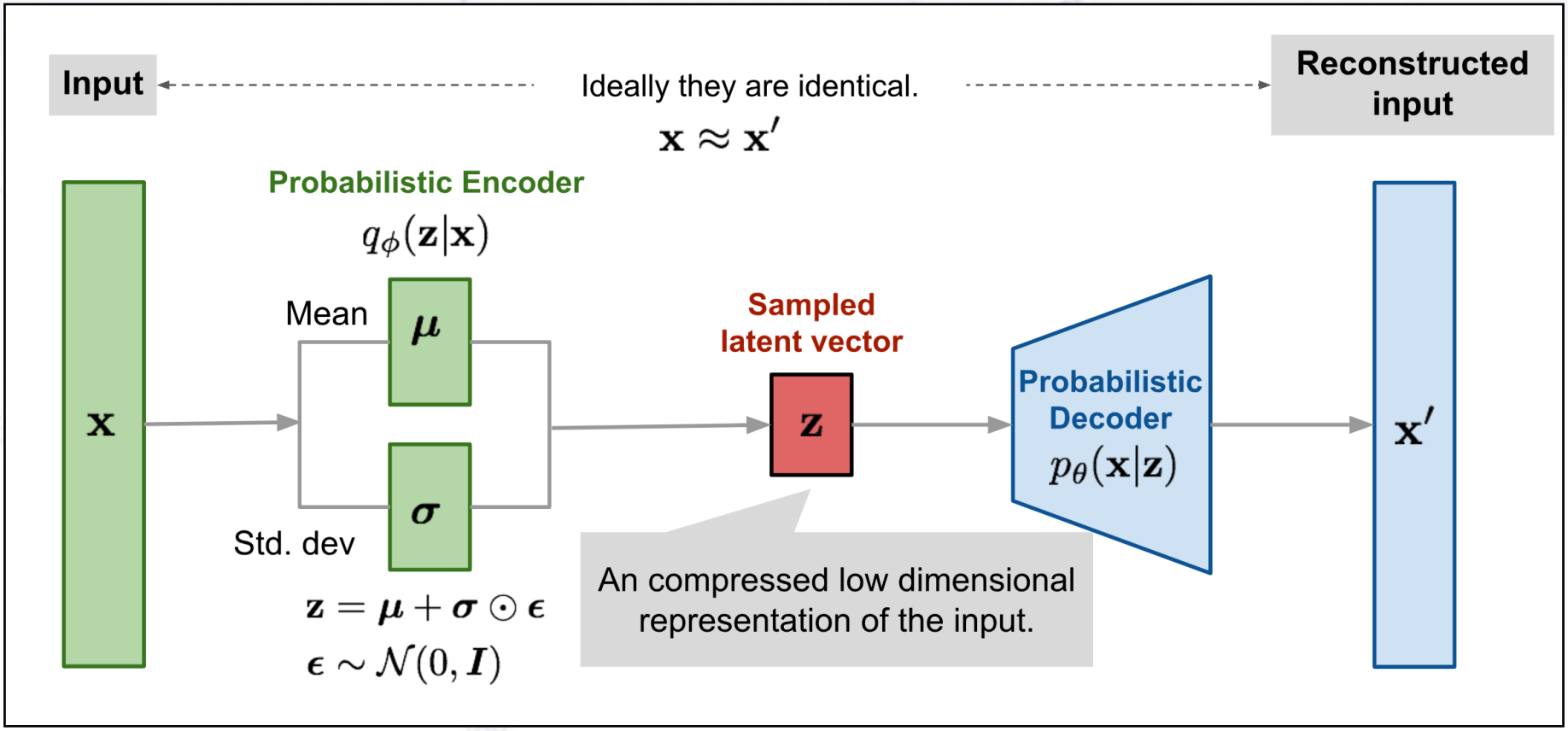
Thus, due to non-regularized latent space AE, the decoder can not be used to generate valid input data from vectors sampled from the latent space.

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.

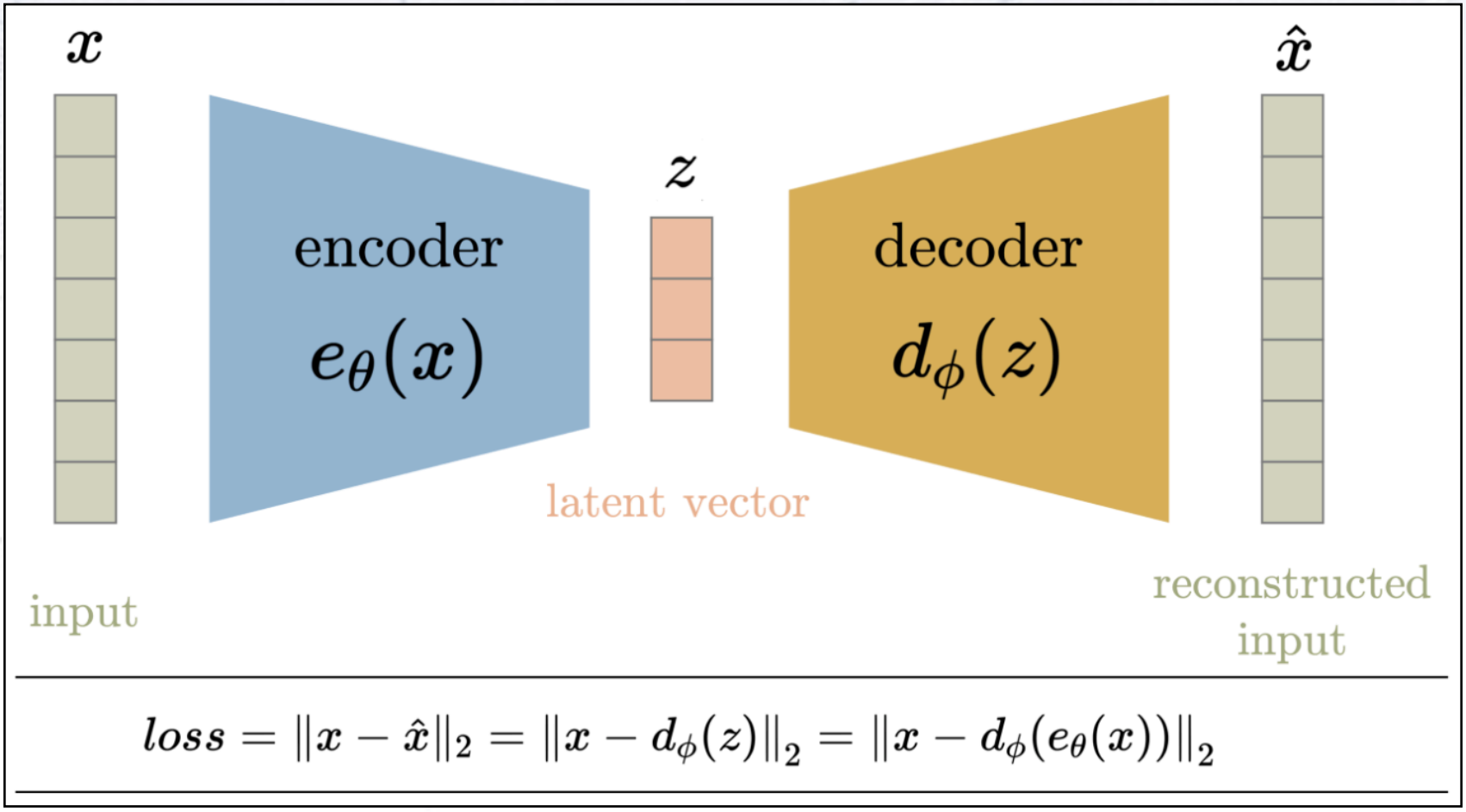
Variational AutoEncoders

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.



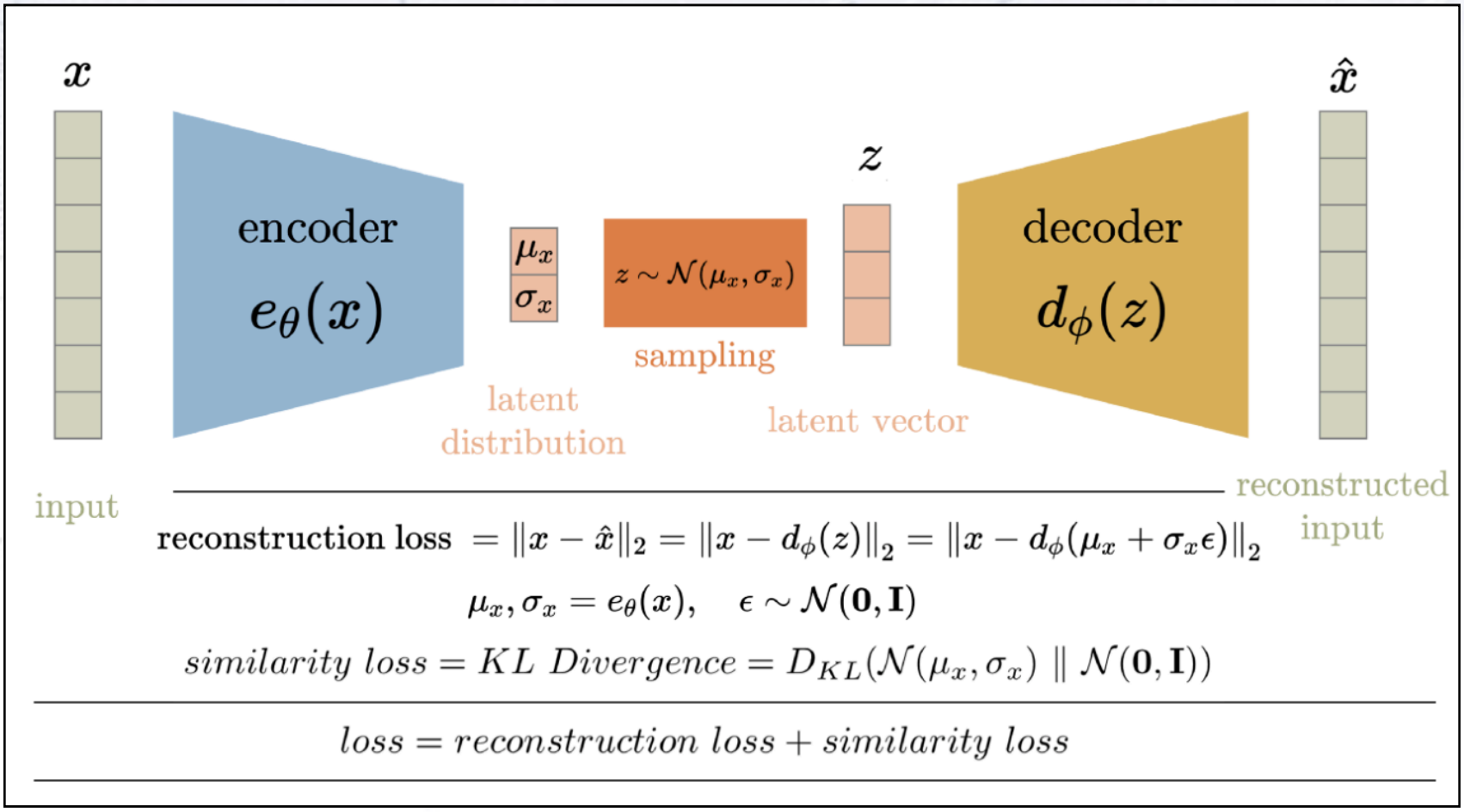
Variational AutoEncoders

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.



Variational AutoEncoders

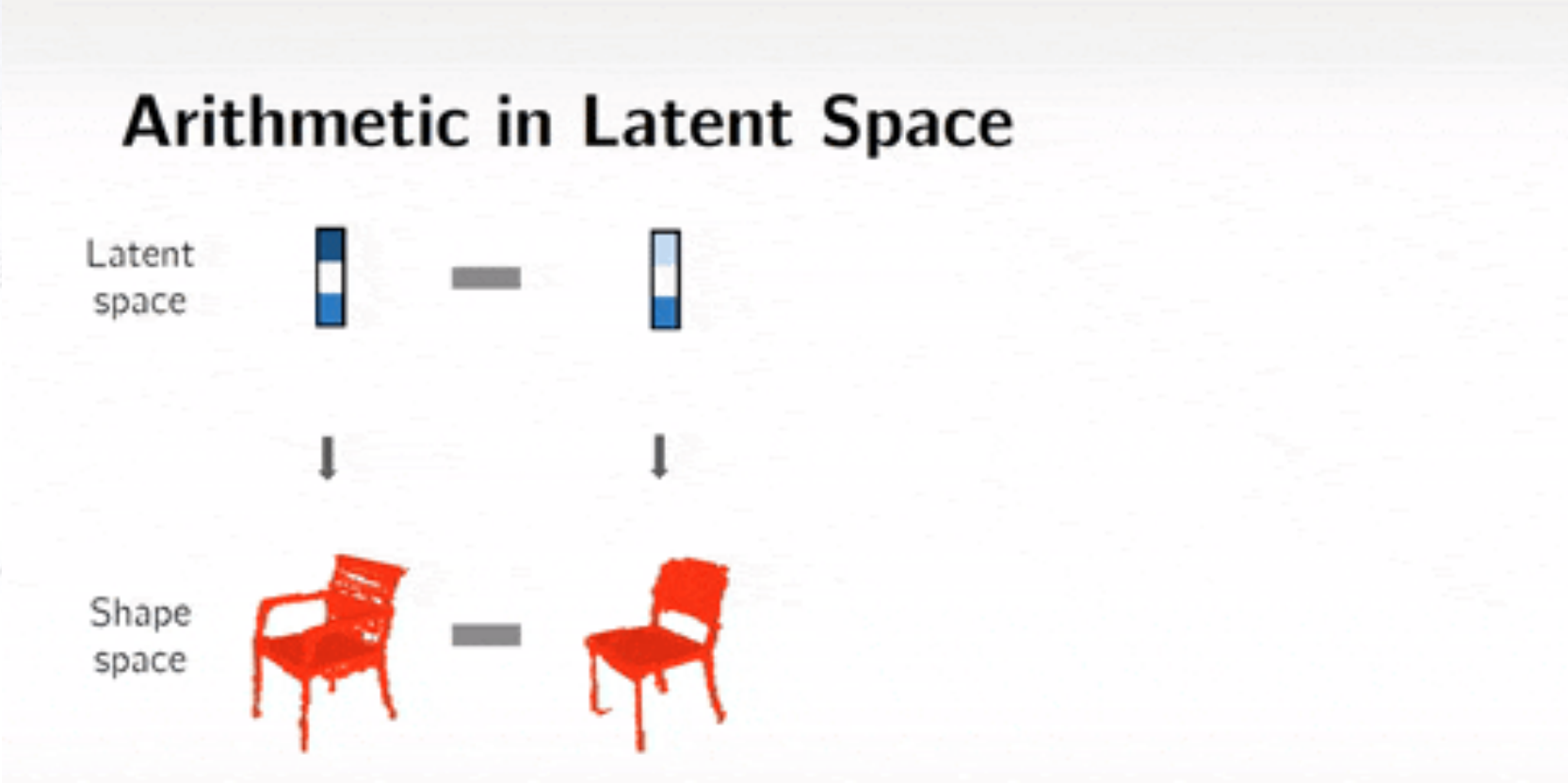
A VAE thus uses a Gaussian-like latent space distribution. It is probabilistic in nature - it produces random cases close (i.e. ε away from) to the original. This is achieved by a "smart" loss function with the Kullback–Leibler (KL) divergence.



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.

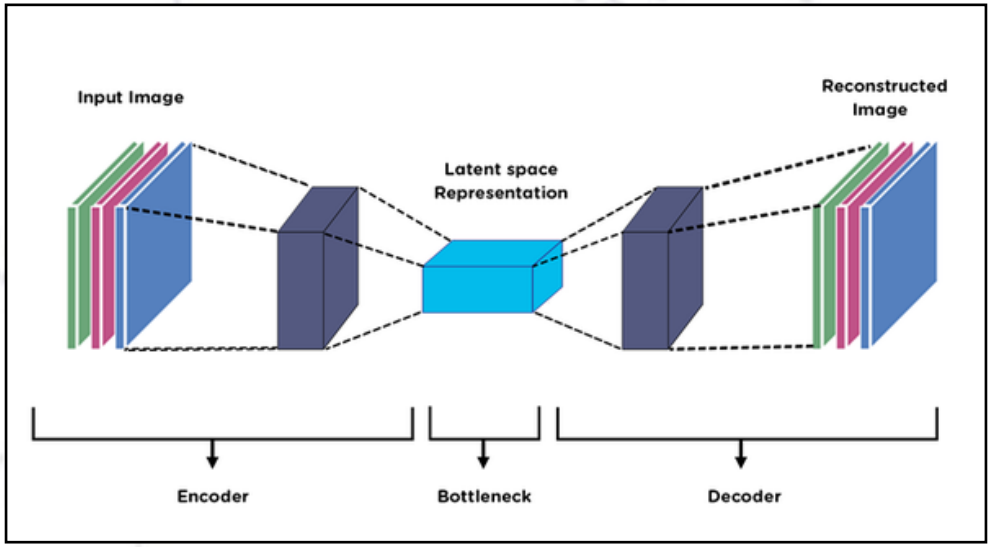
For this reason, one can do arithmetics (typically interpolate) between the inputs:

AE Unsupervised

Since one might have no knowledge of the content of images (or sound, etc.), training an AE is inherently unsupervised. The result is simply a latent space that is a good **representation** of the images.

However, this can be used to cluster "less simple data", as one can apply both dimensionality reduction and/or clustering to the latent space.

This enables one to analyse very complex data in an unsupervised manner.
(e.g. "How many zebra calls exists?")



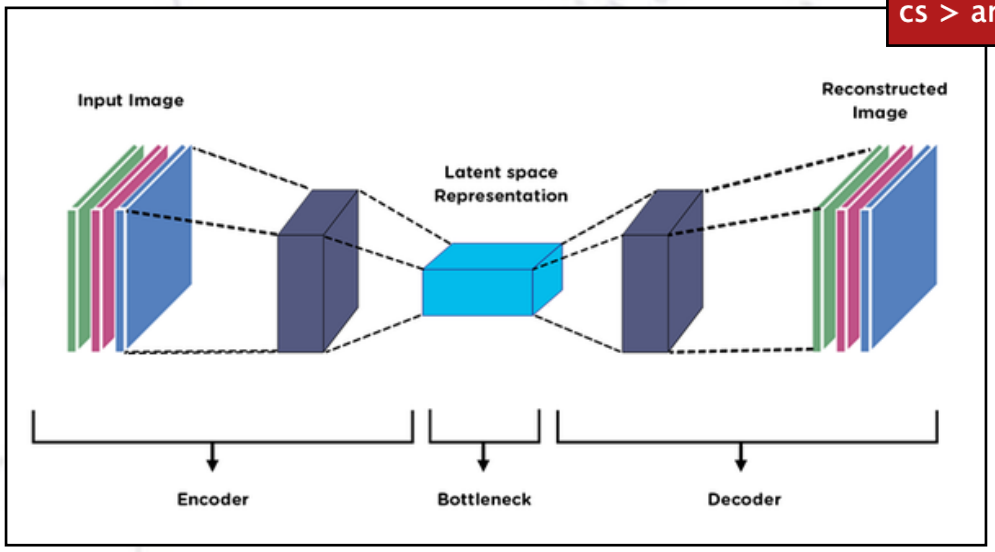
AE Compression

The latent space is also called the “code”, and is a good (but not perfect) representation of the image.
This very “code” is an efficient way of compressing images - or anything else that can be fed into an NN - down to a fixed size.
The approach has proven to be rather performant, competing with JPEG2000.

An Autoencoder-based Learned Image Compressor:

We propose a lossy image compression system using the deep-learning autoencoder structure… []. Our aim is to produce reconstructed images with good subjective quality subject to the 0.15 bits-per-pixel constraint.

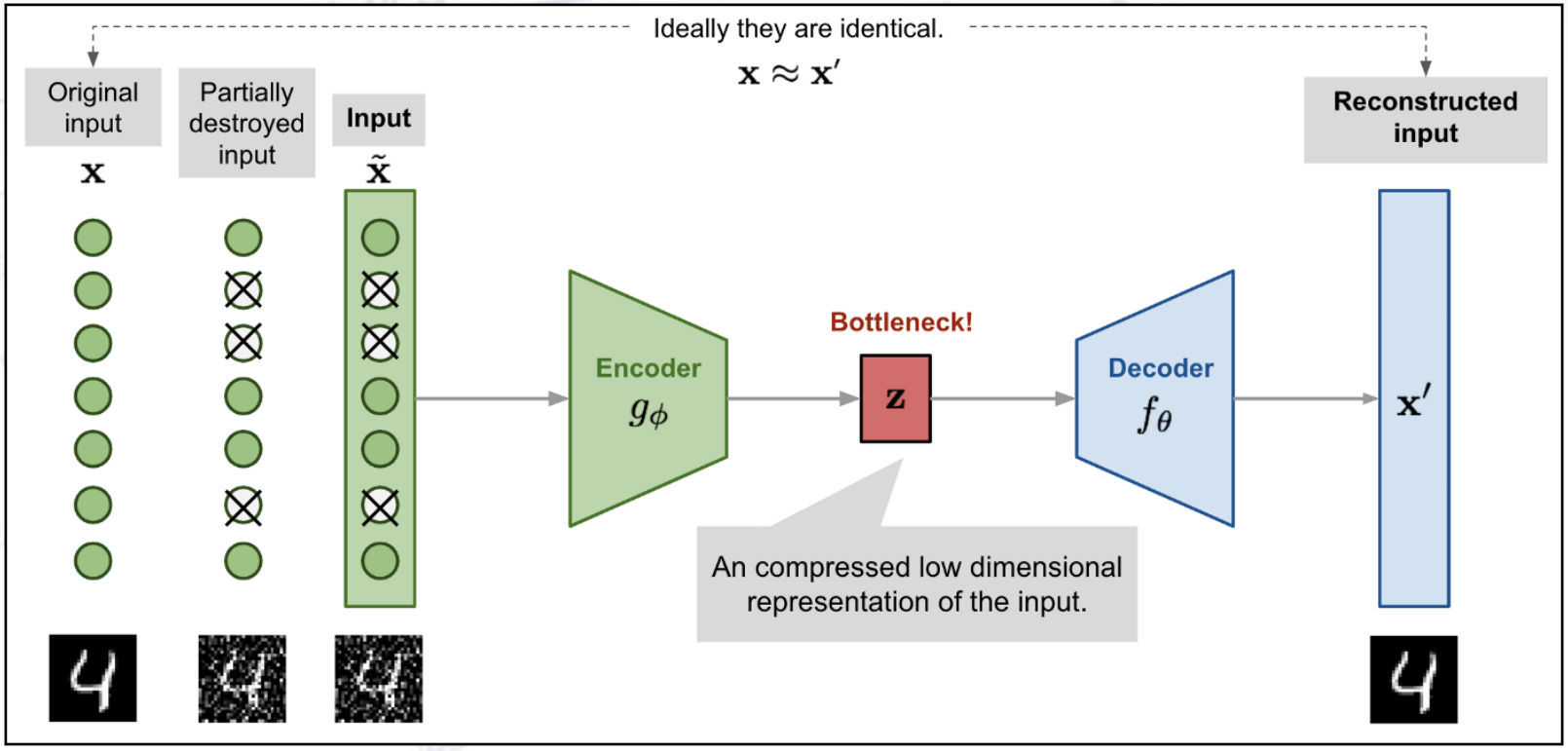
cs > arXiv:1902.07385



De-noising AutoEncoders

AutoEncoders can be used for de-noising for example images.

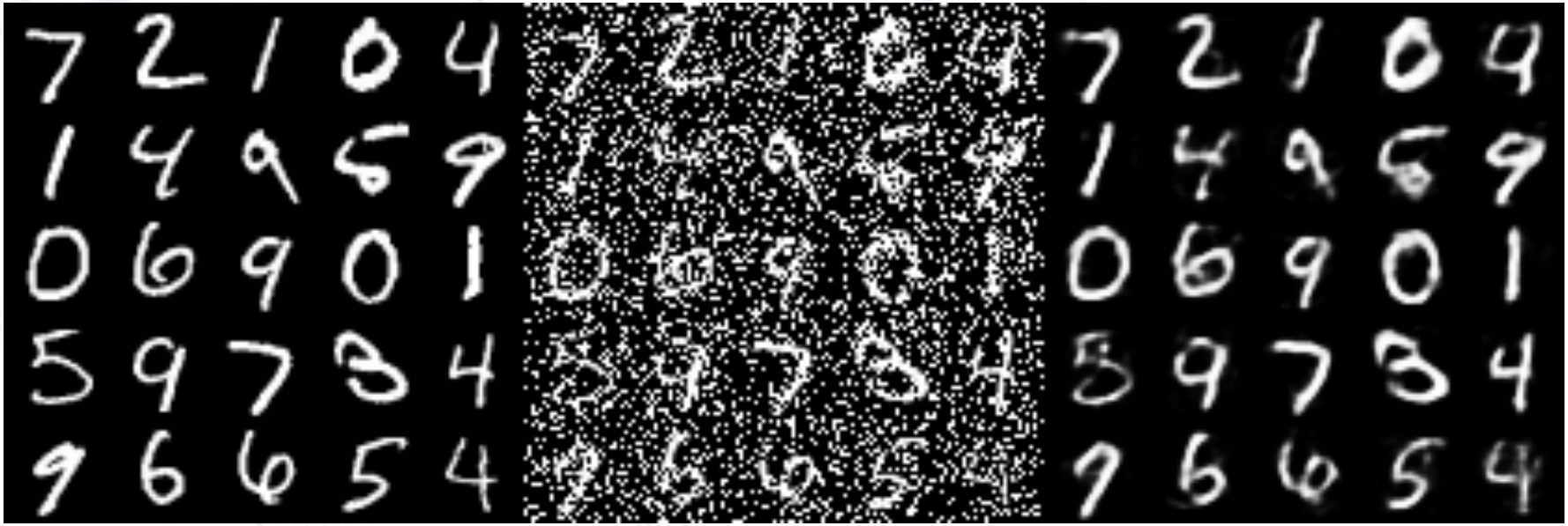
The method consists of first introducing noise (partially ruining the input), and then train an autoencoder to **produce the original image from the noisy input**. Once this is learned, noisy images can be de-noised using this network.



De-noising AutoEncoders

Another way of making this work is perceptual loss. Here one does not (only) compare the input and output images, but also (or only) the subsequent layers in the CNN. These layers learns how images in general works, and thus aren't as affected by noise.

This idea can be taken further to reconstructing larger parts of an image. This is called "inpainting".



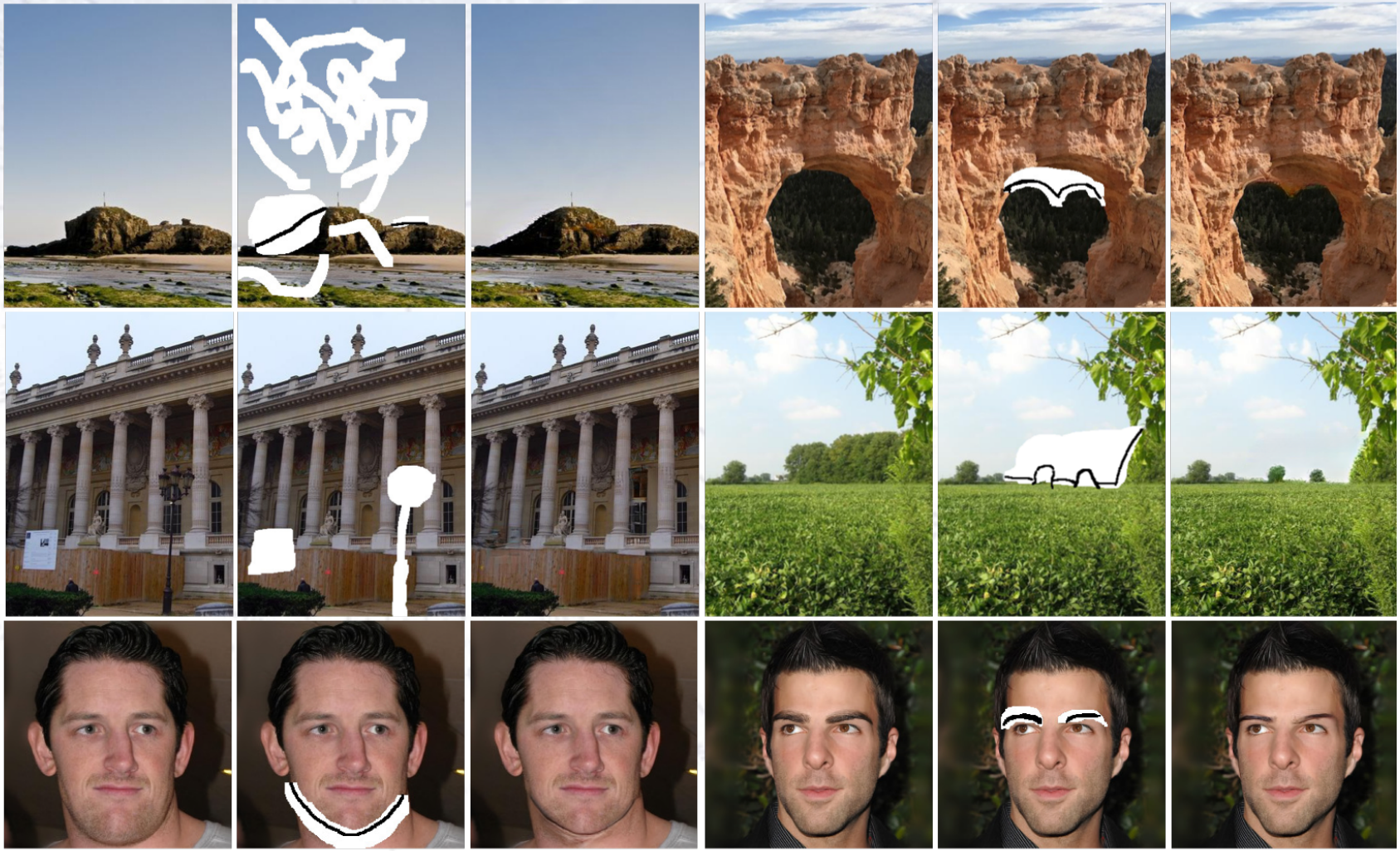
Tutorial on De-noising AutoEncoder

Inpainting with AEs

AEs can also be used for “inpainting”, which means replacing damaged/lost parts of an image.

Inpainting with AEs

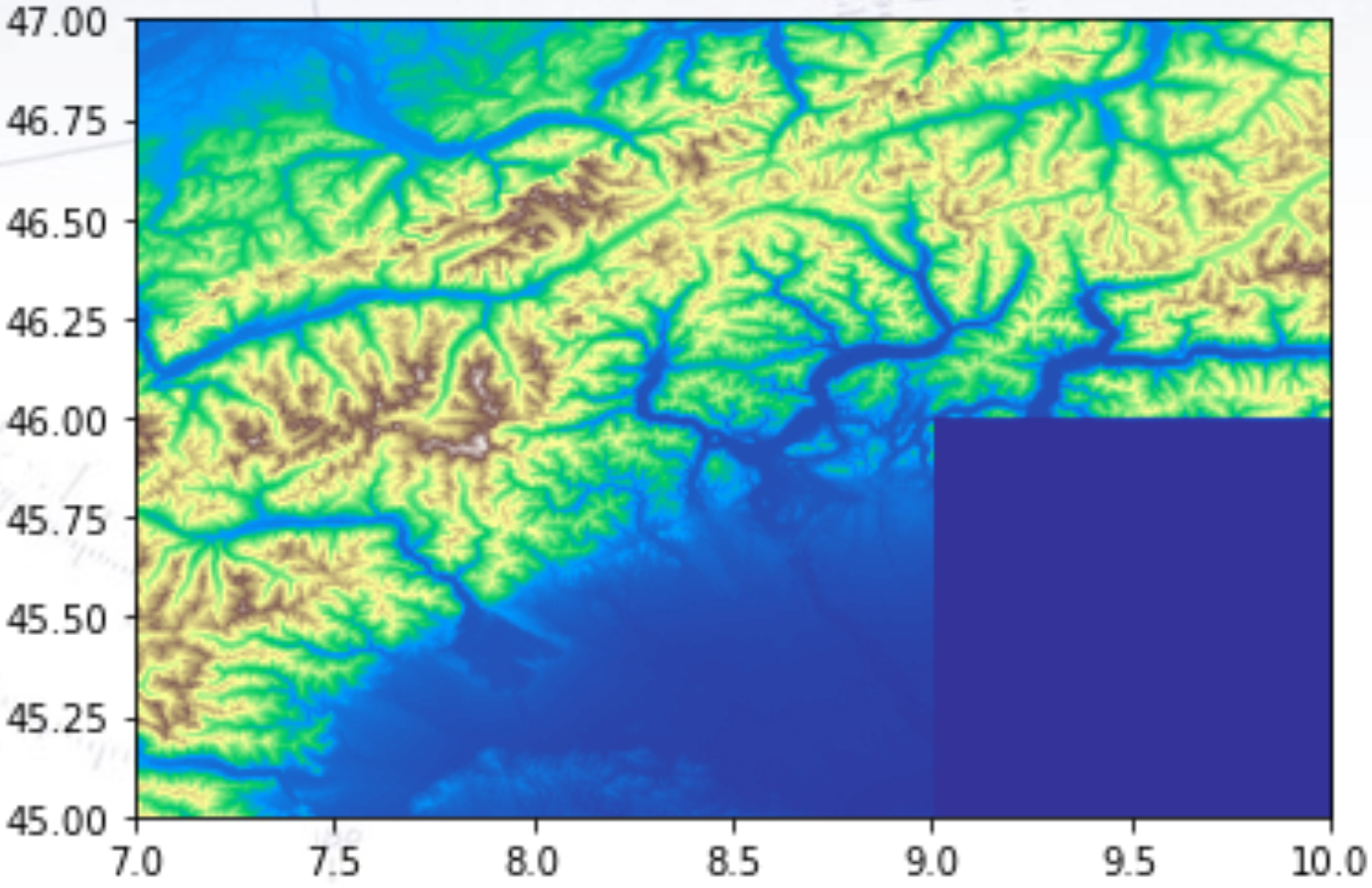
AEs can also be used for “inpainting”, which means replacing damaged/lost parts of an image.



Inpainting with AEs

AEs can also be used for “inpainting”, which means replacing damaged/lost parts of an image.

This can be used for estimating the volume of glaciers, given an altitude map.



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.

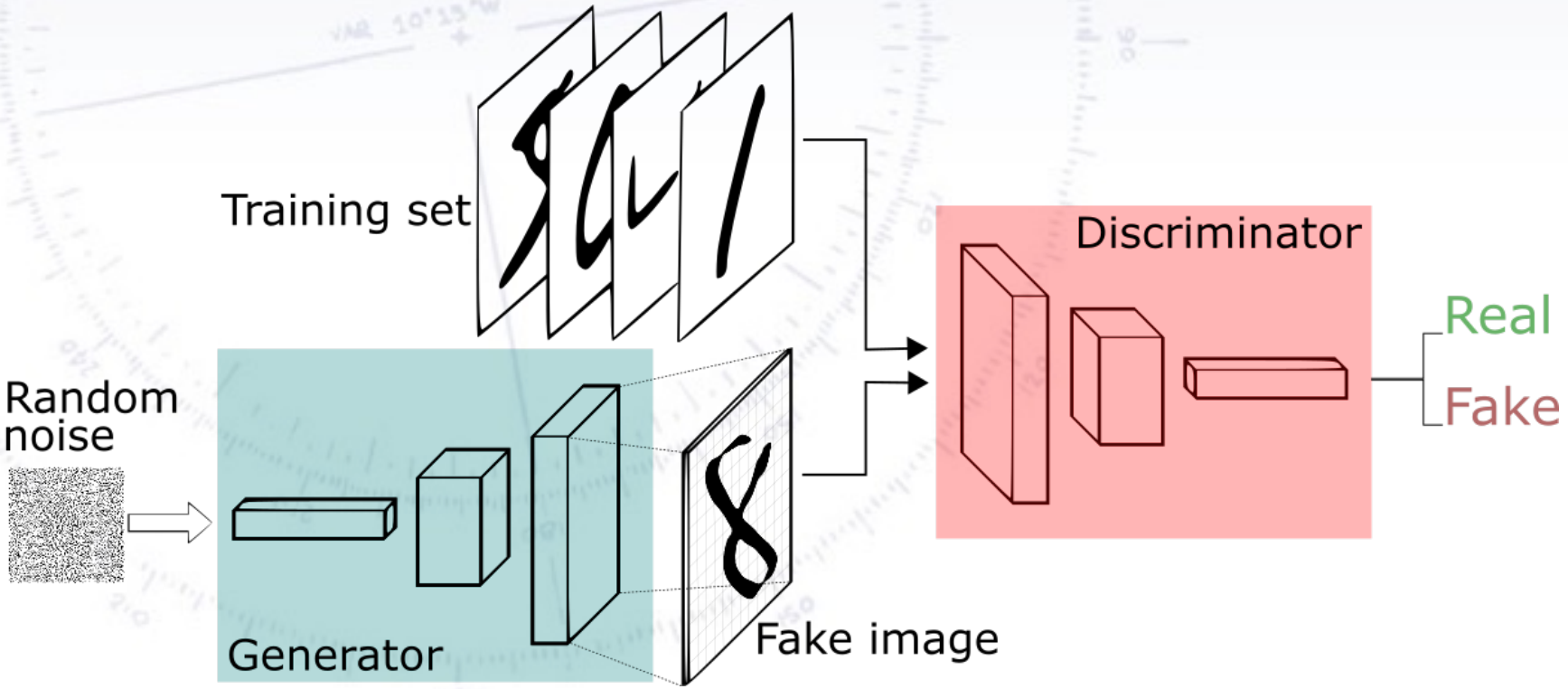
"The coolest idea in machine learning in the last twenty years"

[Yann LeCun, French computer scientist]

GAN drawing

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

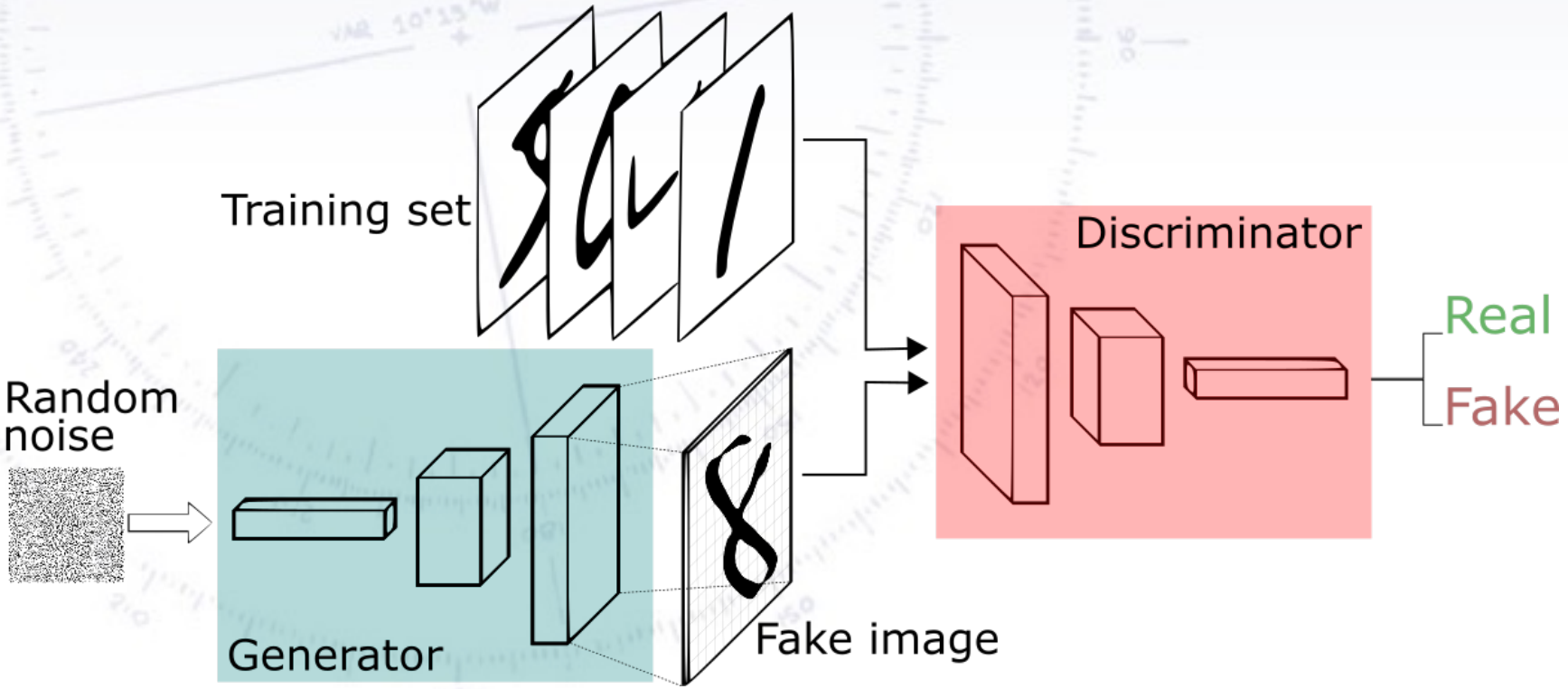
Given a large training set, you can ask you GAN to produce numbers. At first it will do poorly, but as it is "punished" by the discriminator, it improves, and at the end it might be able to produce numbers of **equal quality to real data**:



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

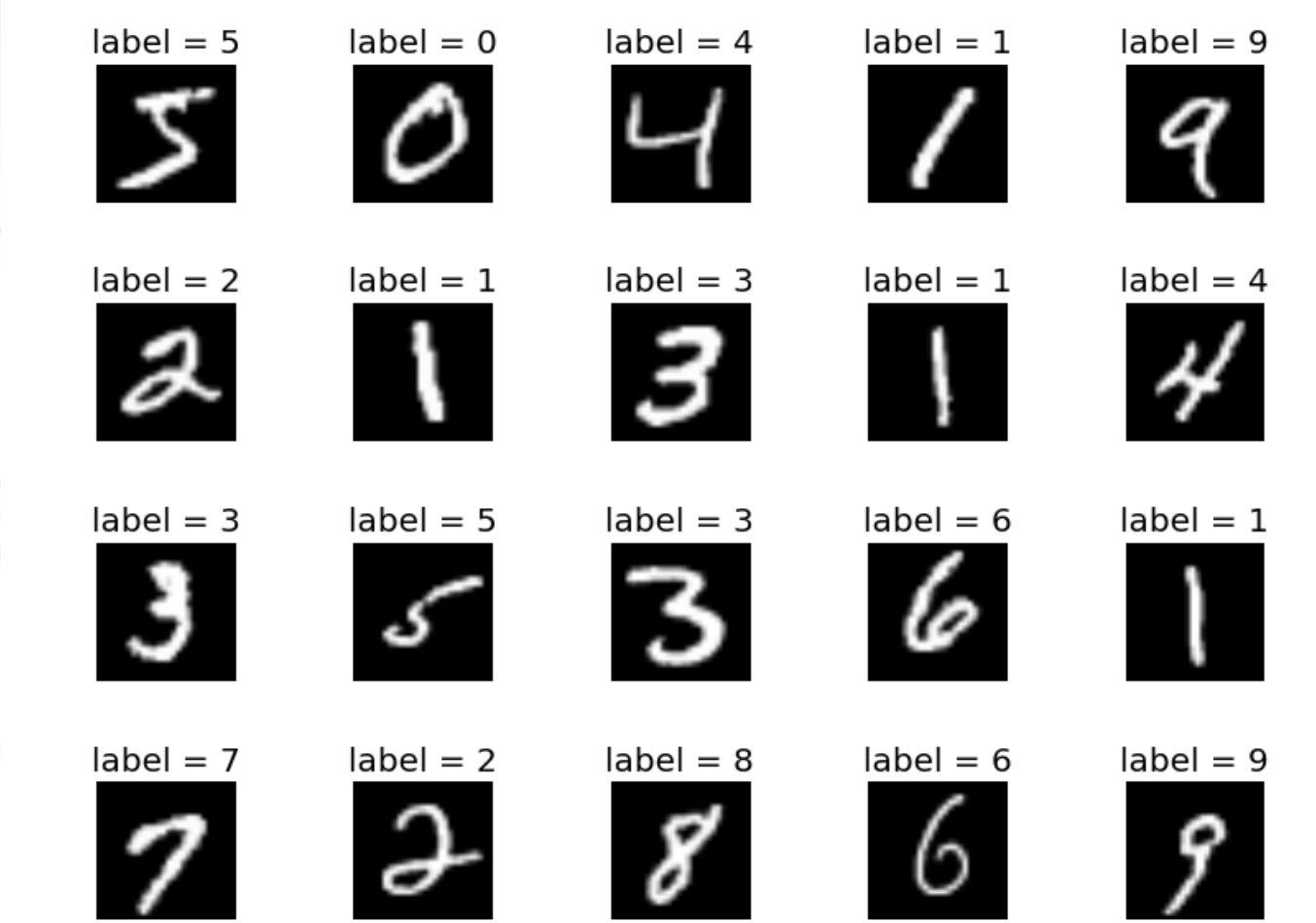
In 2017, Nvidia published the result of their "AI" GANs for producing celebrity faces. There is of course a lot of training data… here are the results:

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… 28x28 gray scale images of numbers. While that spans a large sp… is probably (surely!) much smaller, as far fro… d intensities are present.

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

Anomaly Detection

AE Anomaly Detection

By learning to replicate the most salient features in the training data under some of the constraints described previously, the model is encouraged to learn to precisely reproduce the most frequently observed characteristics.

When facing **anomalies**,
the model should worsen its reconstruction performance.

AE Anomaly Detection

By learning to replicate the most salient features in the training data under some of the constraints described previously, the model is encouraged to learn to precisely reproduce the most frequently observed characteristics.

When facing **anomalies**,
the model should worsen its reconstruction performance.

In most cases, **only data with normal instances are used to train the autoencoder**. In others, the frequency of anomalies is small compared to the observation set, so that its contribution to the learned representation could be ignored. After training, the autoencoder will accurately reconstruct "normal" data, while failing to do so with unfamiliar anomalous data.

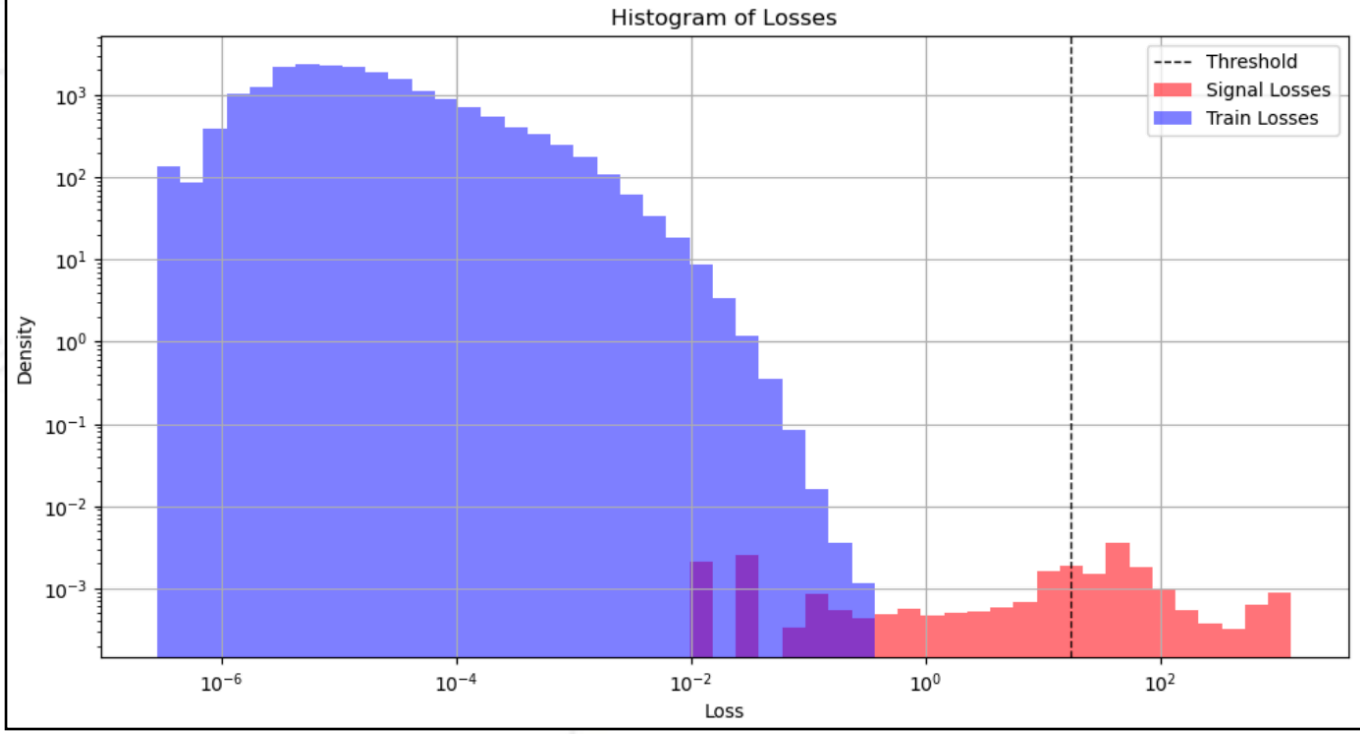
Reconstruction error (the error between the original data and its low dimensional reconstruction) is used as an anomaly score to detect anomalies.

Note: *It has been observed that sometimes the autoencoder "generalizes" so well that it can also reconstruct anomalies well, leading to the miss detection of anomalies.*

AE Anomaly Detection

By learning to replicate the most salient features in the training data under some of the constraints described previously, the model is encouraged to learn to precisely reproduce the most frequently observed characteristics.

When facing **anomalies**,
the model should worsen its reconstruction performance.



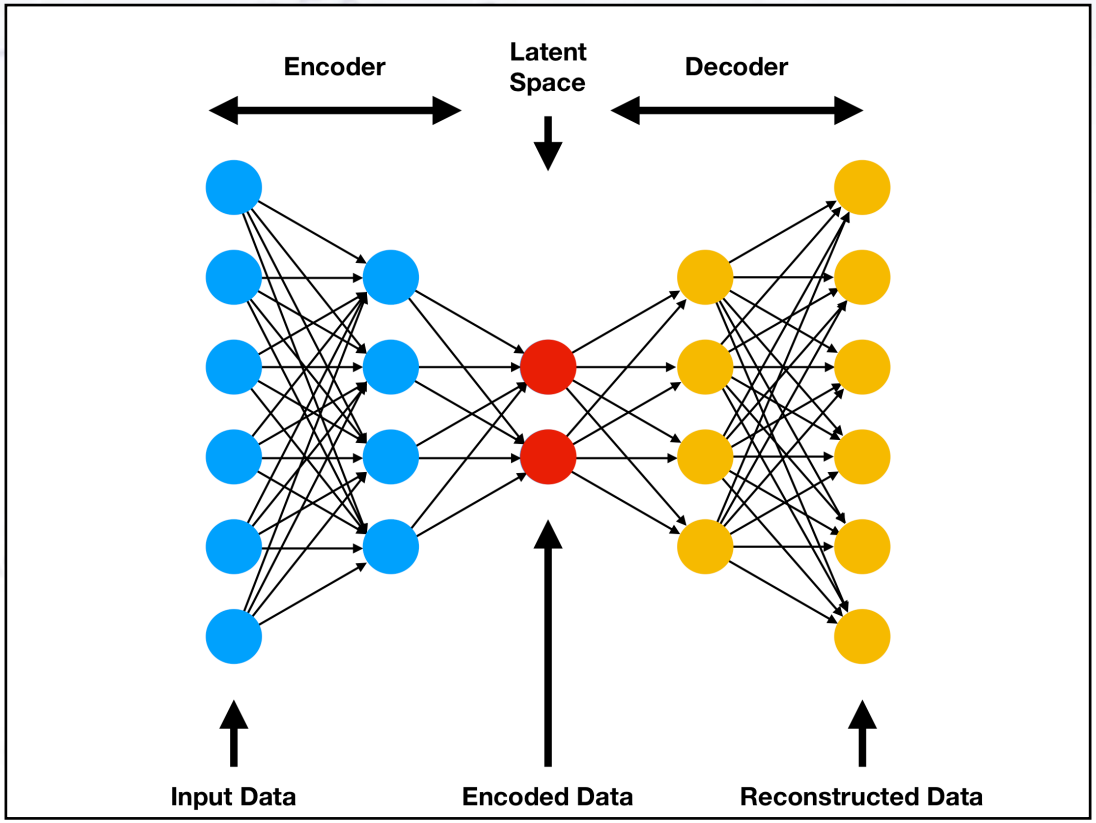
Loss:

$$\mathscr{L}_s = \frac{1}{2}\|y(x_s) - x_s\|^2$$

Example from particle physics: quark jets in Future Circular Collided. 59 input features, 16 dim latent space

AE Training Assistance

Another AE use is "training assistance". Image that you have a large data sample, but only few labelled cases (i.e. few cases where you know the result). The few labelled cases might not be enough to train a full CNN (which can have millions of parameters!).

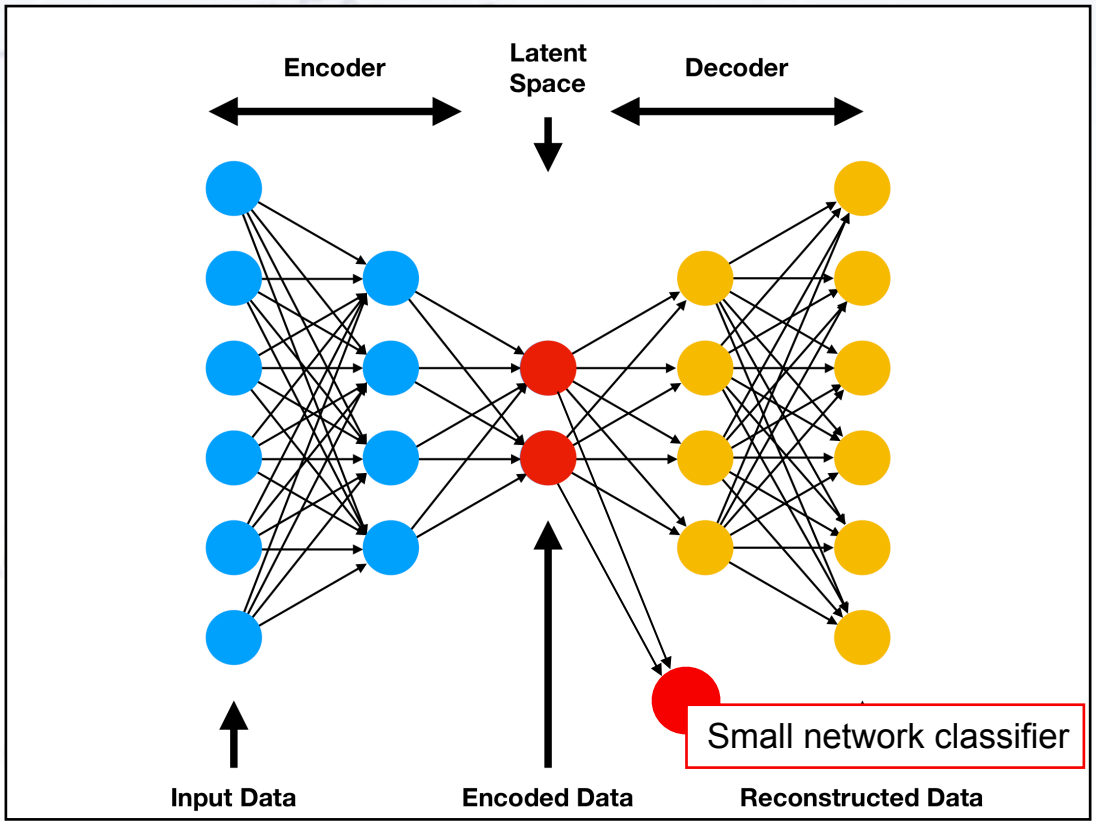


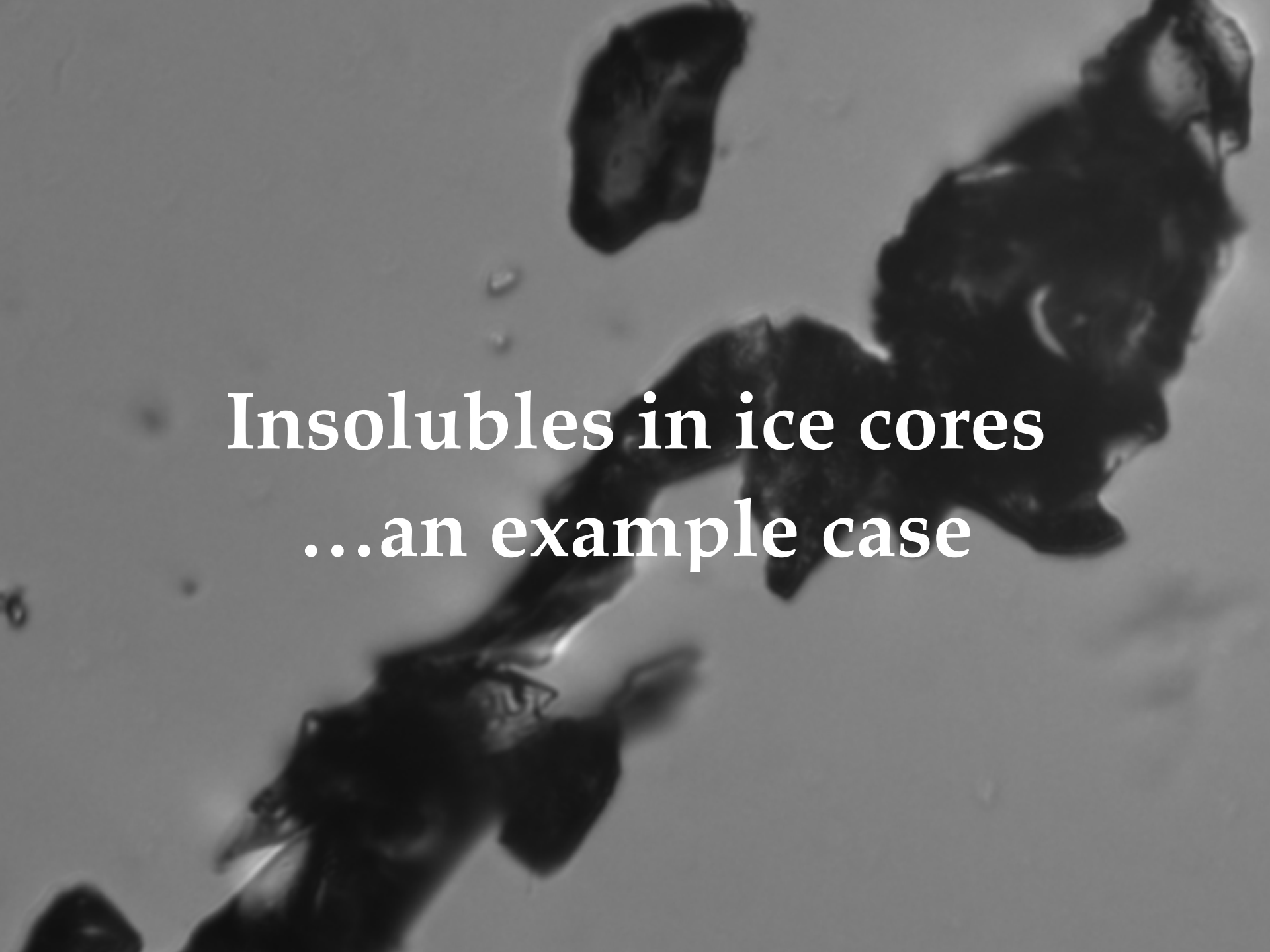
AE Training Assistance

Another AE use is "training assistance". Image that you have a large data sample, but only few labelled cases (i.e. few cases where you know the result). The few labelled cases might not be enough to train a full CNN (which can have millions of parameters!).

But if you train an AE on the large un-labelled dataset, then once this network is in place, you can continue from the latent space.

Now using the small labelled data set, you train a classifier (or regressor).





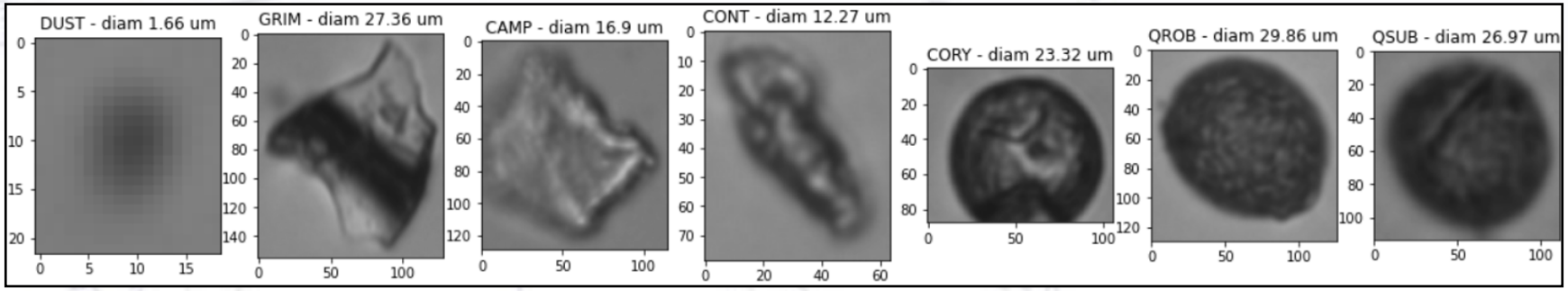
Insolubles in ice cores
…an example case

The data

The data consists of two parts - a training set (control) and a test set (ice core):

- Training: Obtained from (controlled) samples, that was carefully selected.
- Testing: Obtained from melted ice core filtering process.

All the images are gray scale and scaled to be the same size: 128 x 128
In addition, we have 34 numerical features from the microscope imaging.

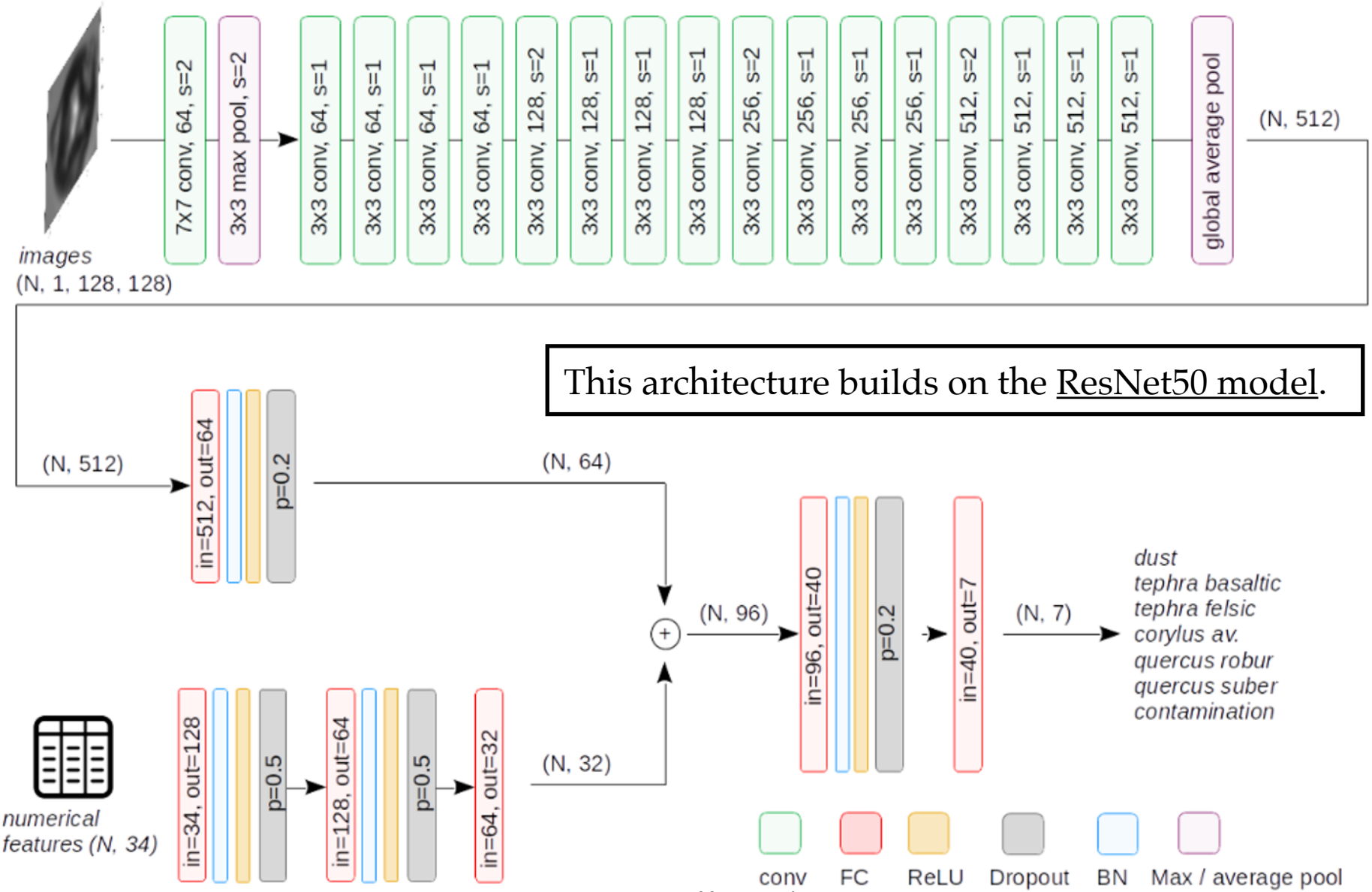


The initial goals are:

- Be able to classify the training images (supervised learning).
- Apply the classification to the testing images (predictions).

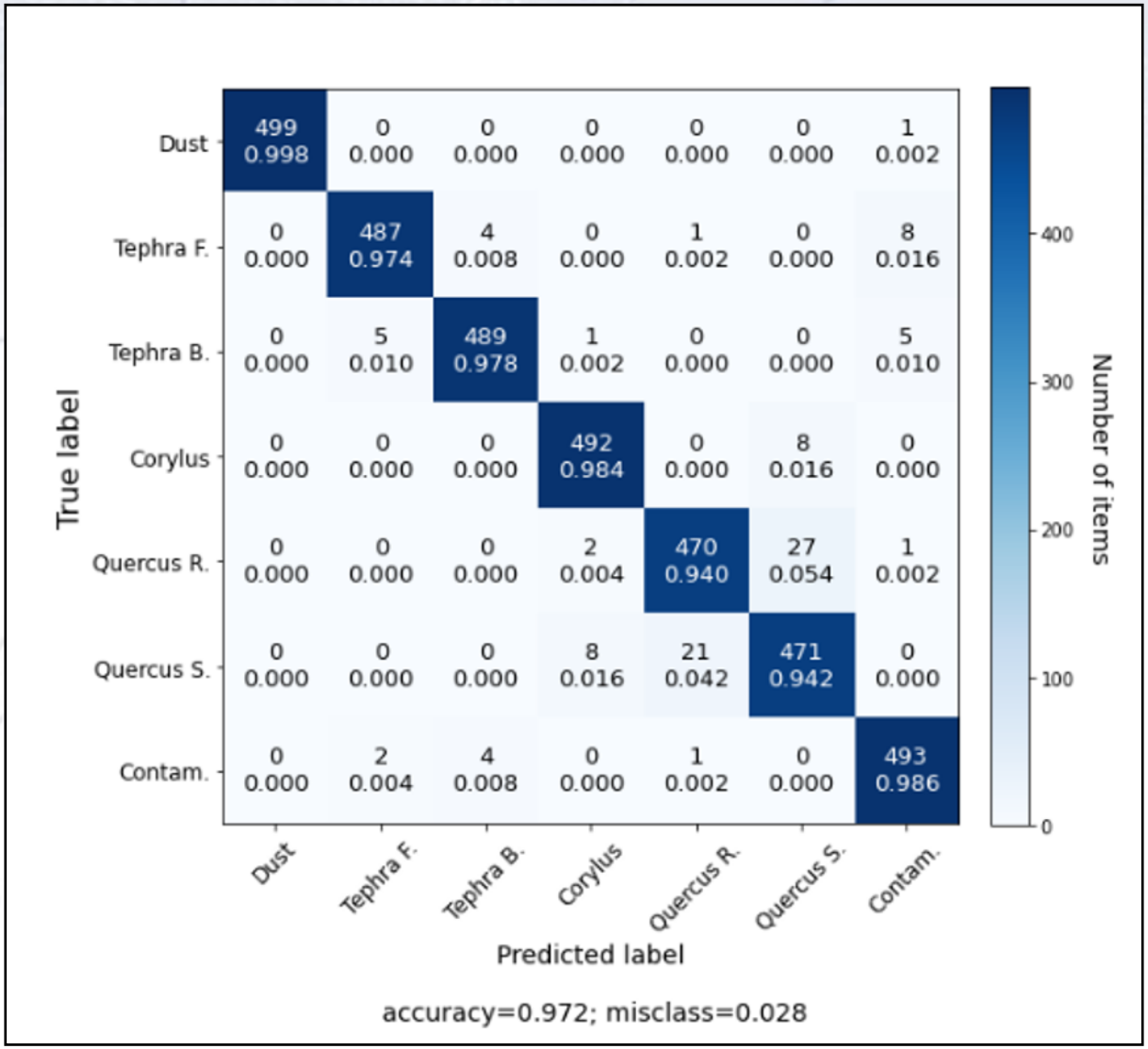
But how can we then know, if there are **other things** in the ice cores?

Classification CNN architecture

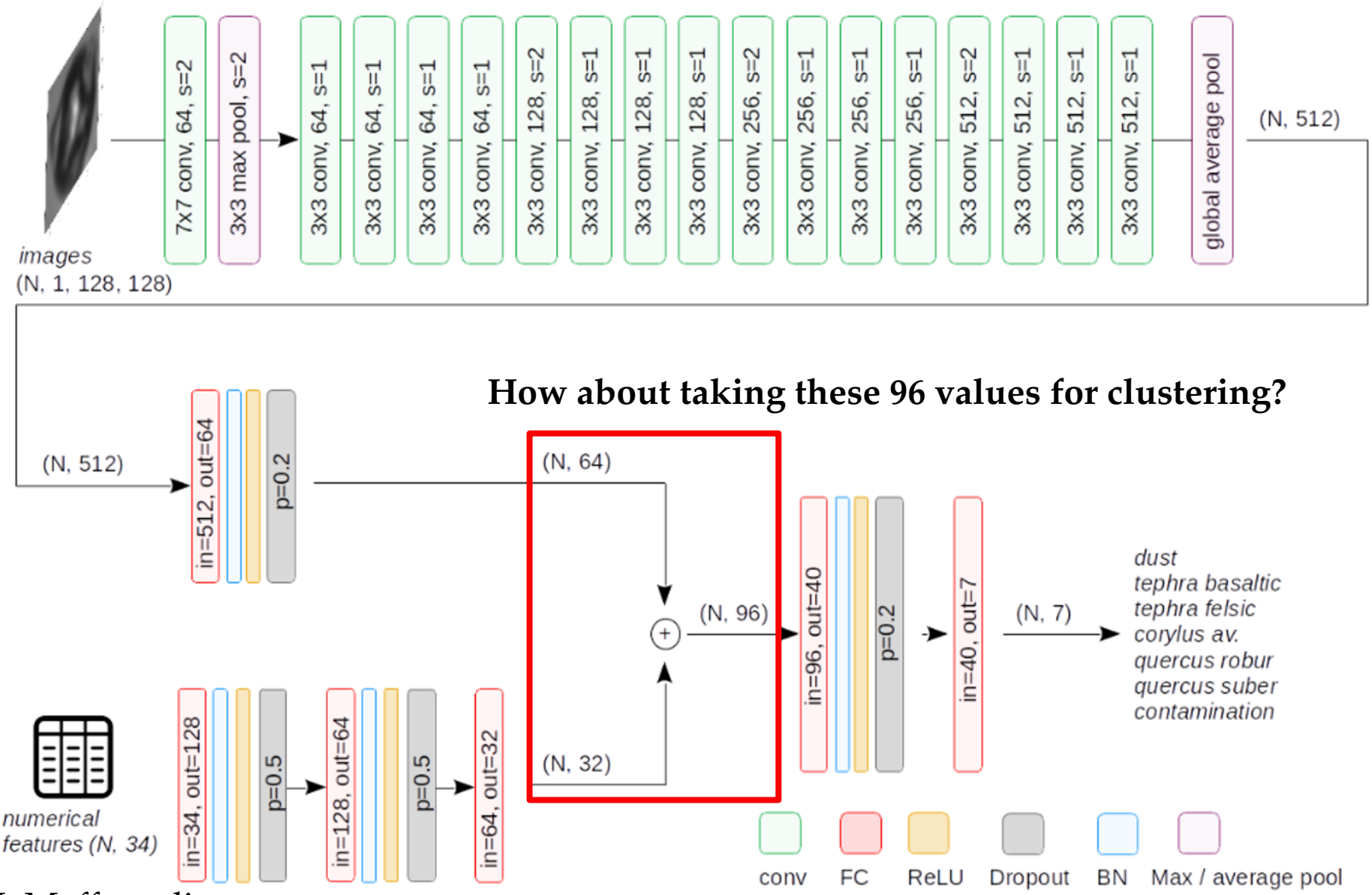


Performance

The CNN performance is rather good:



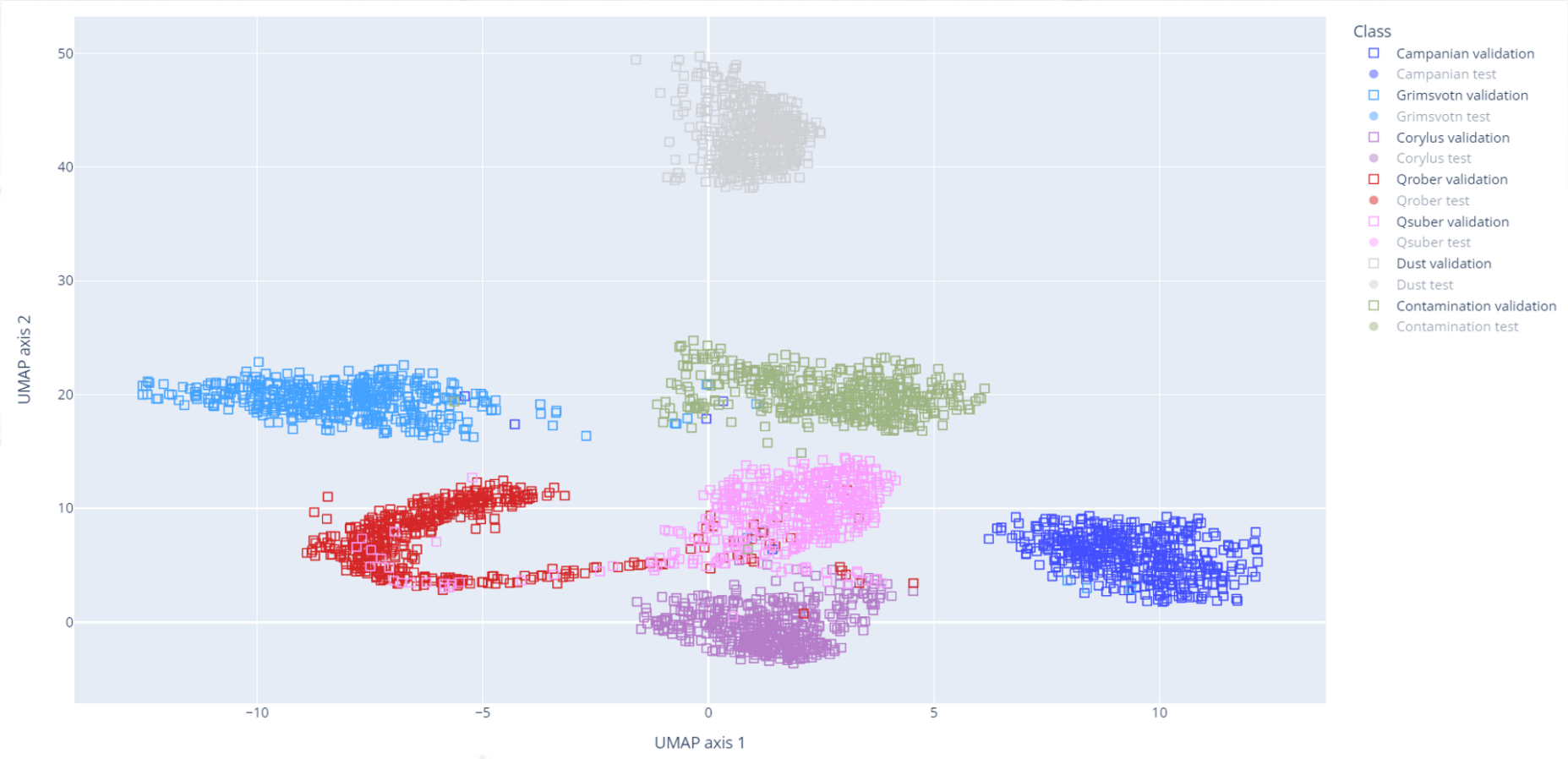
Classification CNN architecture



UMAP clustering

Clearly, this clusters the sample quite well.

As was seen before, the two types of pollen - naturally - have some overlap.



UMAP on testing set…

Applying the UMAP to the testing set… Well, how would you even do that?

UMAP on testing set…

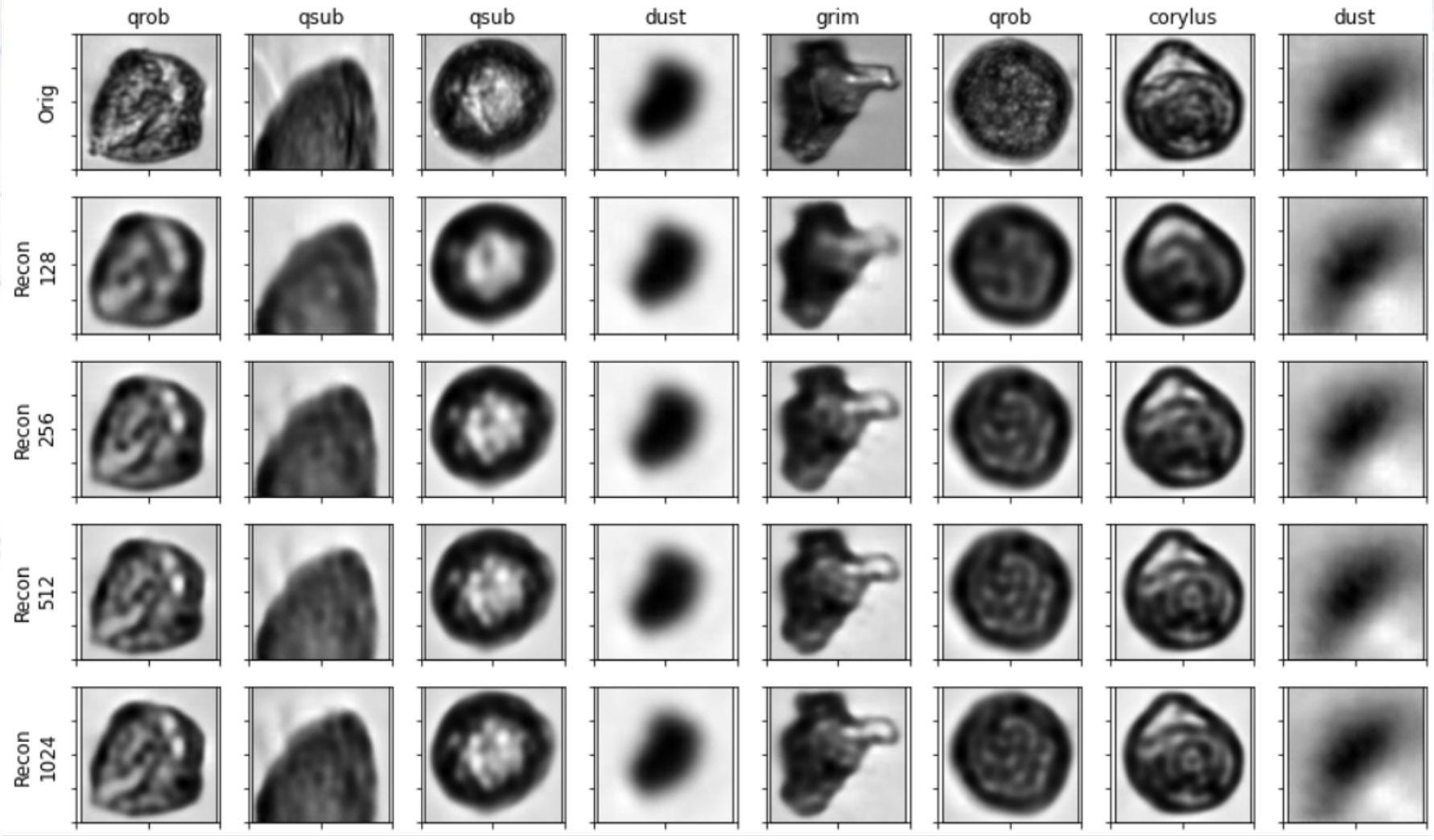
Applying the UMAP to the testing set… Well, how would you even do that?

The result using ParametricUMAP is as follows (dust and cont. dominates).

Autoencoding the images

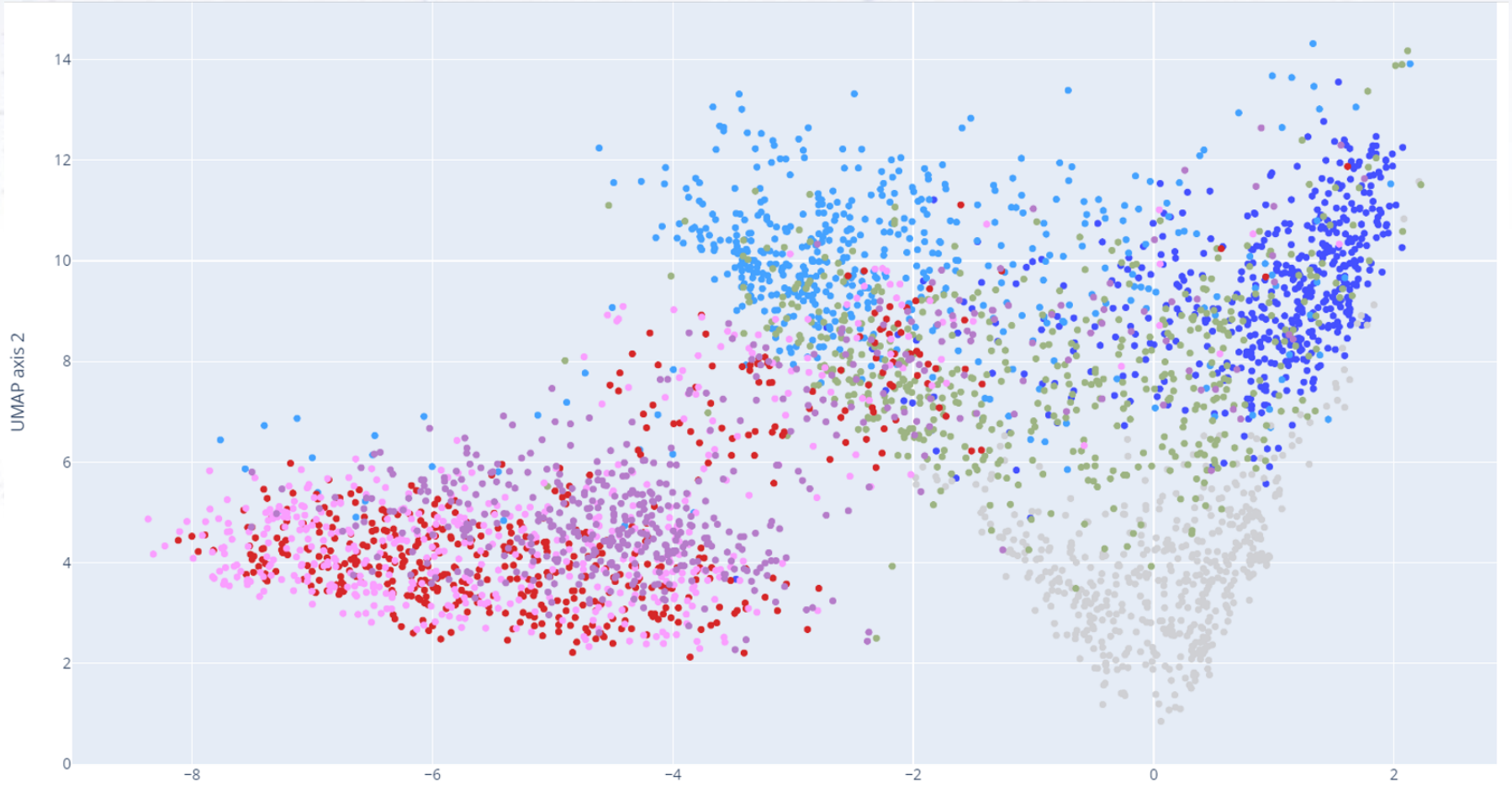
What size should the latent space be?

You should consider what quality of the (output) images you want.



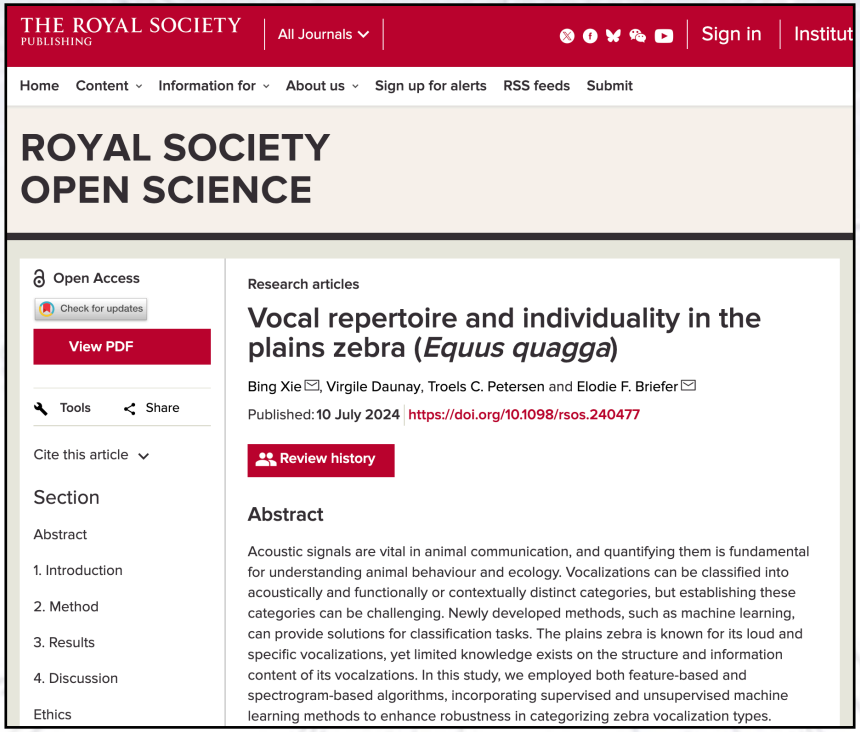
The unsupervised result…

Not using the training sample (which might bias the classification), we have managed to produce the following distribution.
Dust is clearly separated, as is the pollen (though not internally) and the two tephra types. Contaminations overlap mainly with the tephra.



Do you speak Zebra?

From recordings of Zebra sounds, the spectrogram can be clustered:



THE ROYAL SOCIETY PUBLISHING | All Journals | Sign in | Institut

Home | Content | Information for | About us | Sign up for alerts | RSS feeds | Submit

ROYAL SOCIETY OPEN SCIENCE

Open Access

Check for updates

View PDF

Tools | Share

Cite this article

Section

Abstract

1. Introduction

2. Method

3. Results

4. Discussion

Ethics

Research articles

Vocal repertoire and individuality in the plains zebra (*Equus quagga*)

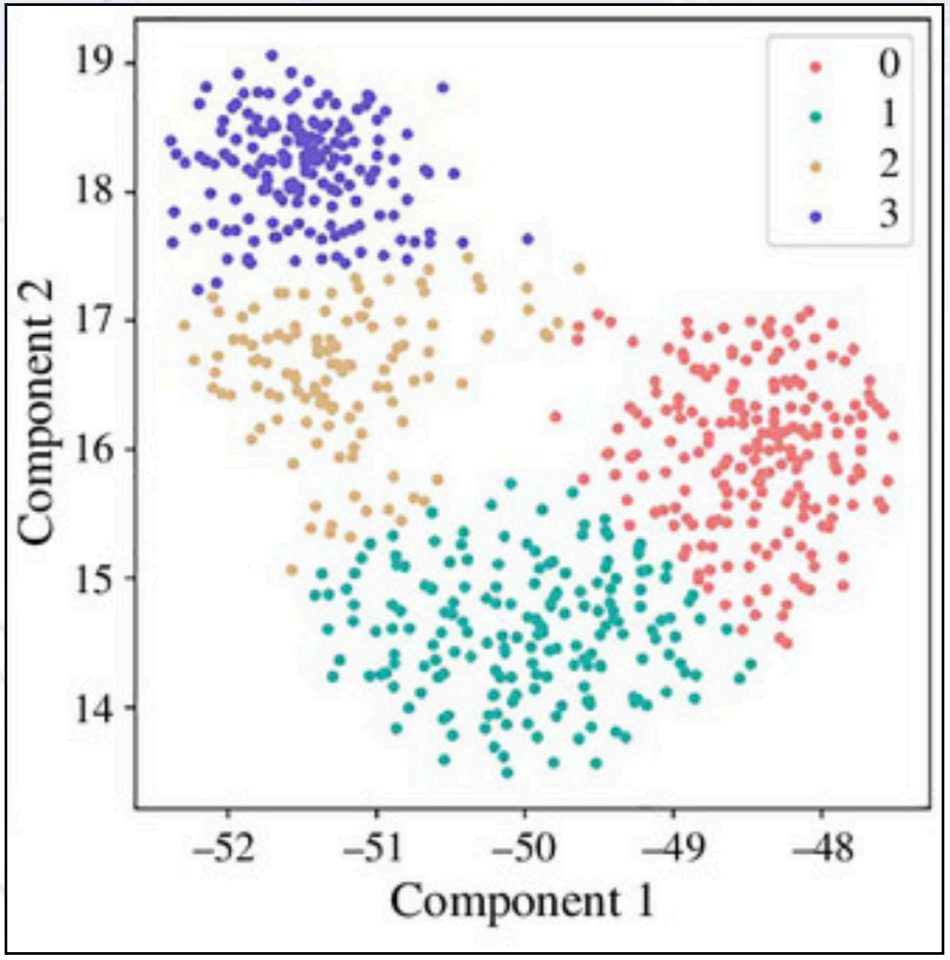
Bing Xie, Virgile Daunay, Troels C. Petersen and Elodie F. Briefer

Published: 10 July 2024 | https://doi.org/10.1098/rsos.240477

Review history

Abstract

Acoustic signals are vital in animal communication, and quantifying them is fundamental for understanding animal behaviour and ecology. Vocalizations can be classified into acoustically and functionally or contextually distinct categories, but establishing these categories can be challenging. Newly developed methods, such as machine learning, can provide solutions for classification tasks. The plains zebra is known for its loud and specific vocalizations, yet limited knowledge exists on the structure and information content of its vocalzations. In this study, we employed both feature-based and spectrogram-based algorithms, incorporating supervised and unsupervised machine learning methods to enhance robustness in categorizing zebra vocalization types.



The Variational AutoEncoder result is then UMAP'ed into 2D, which shows great overlap with the humanly detected sounds.