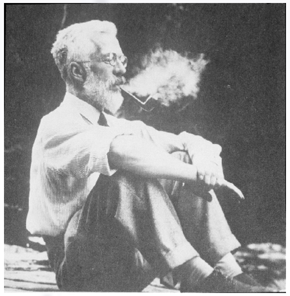
Applied ML

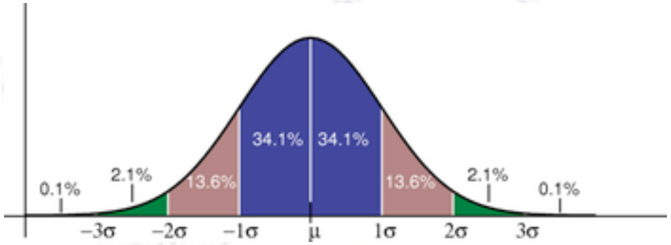
Generative Adversarial Networks








Troels C. Petersen (NBI)

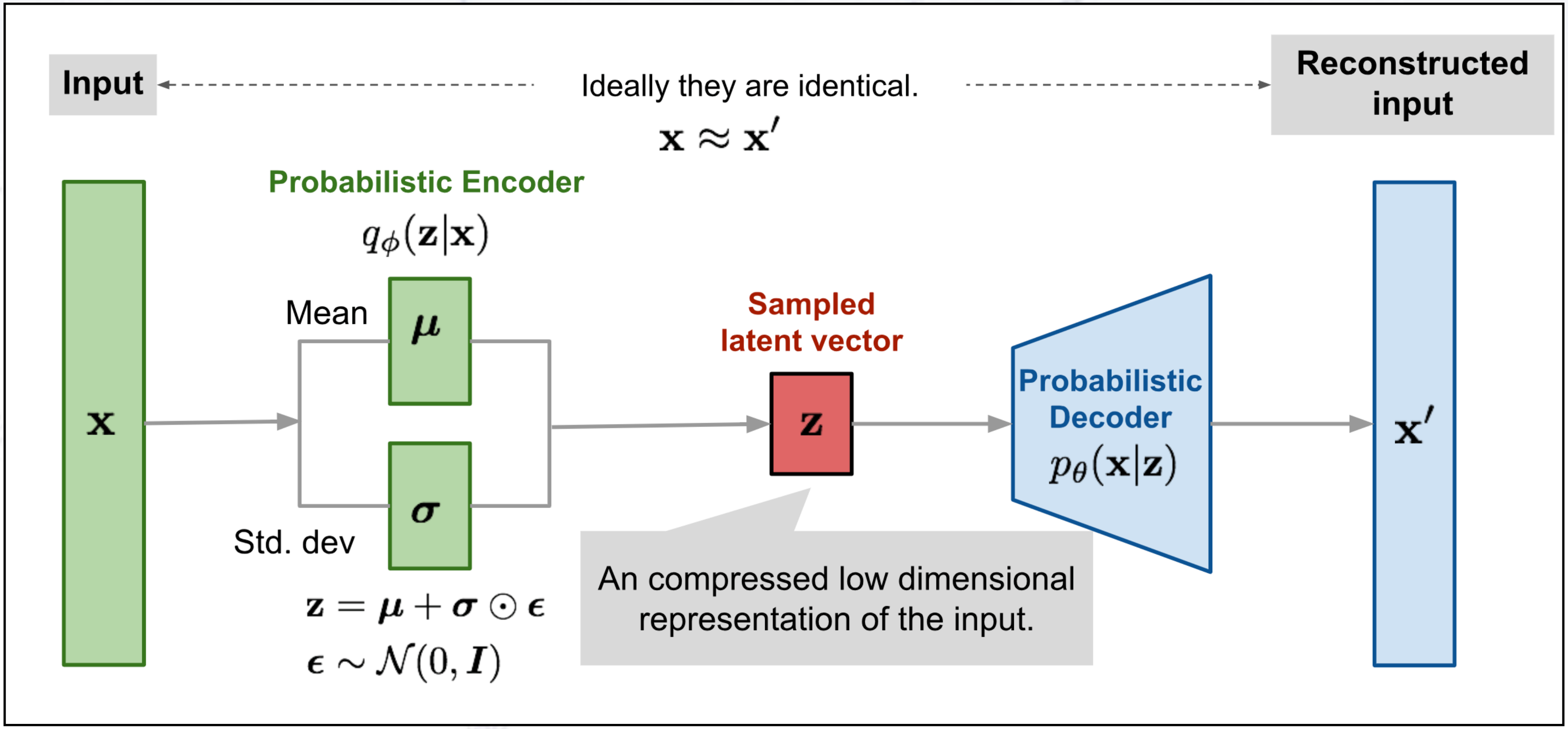


"Statistics is merely a quantisation of common sense - Machine Learning is a sharpening of it!"

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").
This dimensionality reduction is schematically shown below. The idea is "old" (80ies) and closely related to (the basis of) Generative Networks.

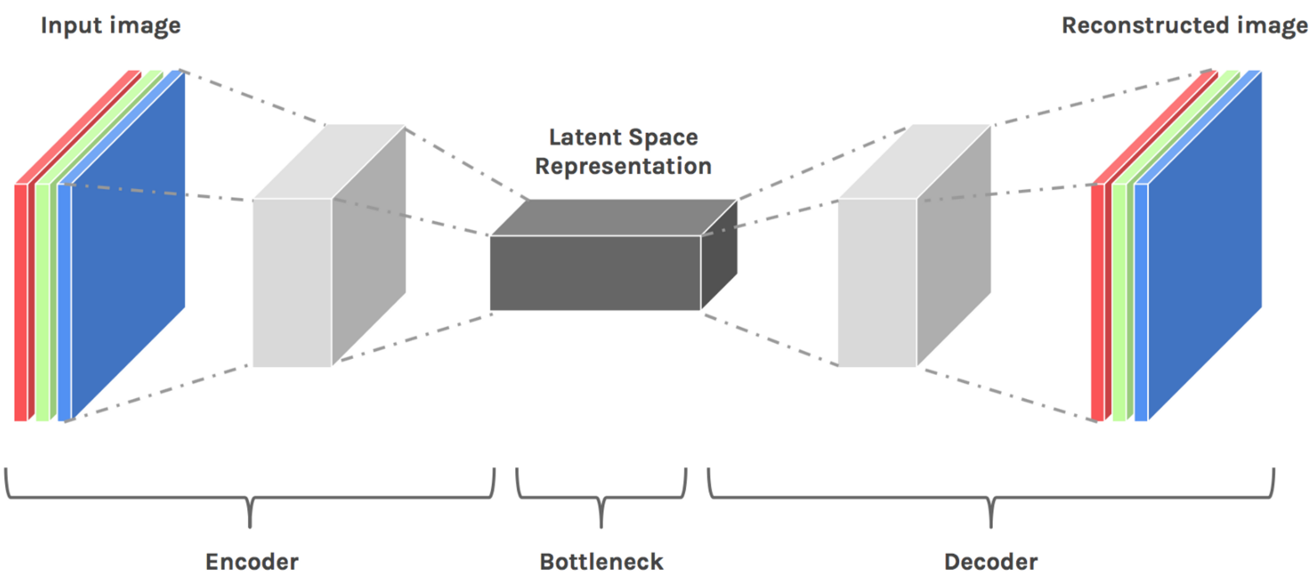


Latent space

Latent variables are variables that are **inferred** instead of directly **observed**. They may correspond to some (combination of) physical reality, e.g. temperature, (then also called hidden variables), but can also correspond to abstract concepts!!!

One advantage of using latent variables is that they can serve to **reduce the dimensionality of data**. Also, latent variables link observable data in the real world to symbolic data in the modelled world.

A **latent space** is one spanned by latent variables, thus containing the main features.

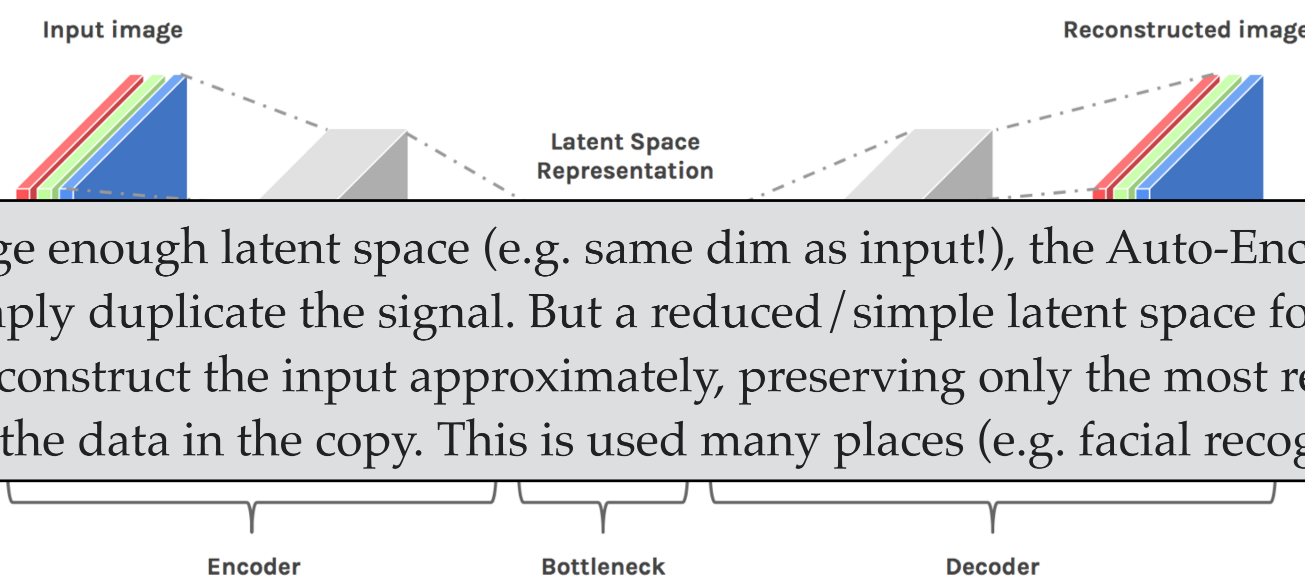


Latent space

Latent variables are variables that are **inferred** instead of directly **observed**. They may correspond to some (combination of) physical reality, e.g. temperature, (then also called hidden variables), but can also correspond to abstract concepts!!!

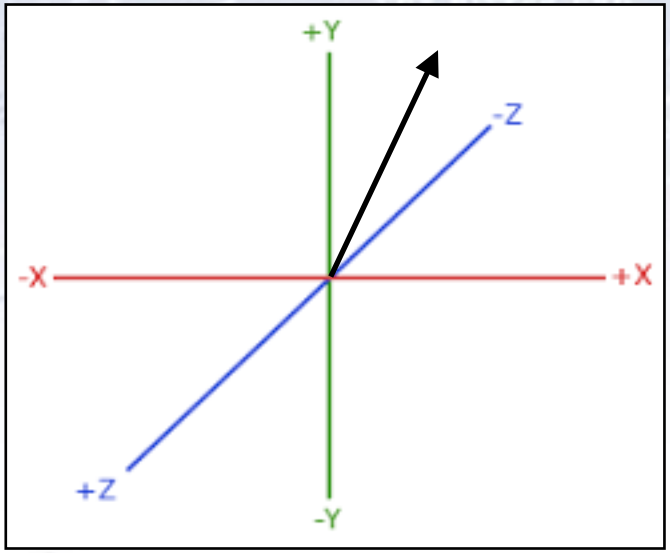
One advantage of using latent variables is that they can serve to **reduce the dimensionality of data**. Also, latent variables link observable data in the real world to symbolic data in the modelled world.

A **latent space** is one spanned by latent variables, thus containing the main features.



With a large enough latent space (e.g. same dim as input!), the Auto-Encoder would simply duplicate the signal. But a reduced/simple latent space forces them to reconstruct the input approximately, preserving only the most relevant aspects of the data in the copy. This is used many places (e.g. facial recognition).

Example: Latent space for PCA



Consider a 3 dimensional space on which we apply a PCA analysis.

Then the principle component will fall in some direction spanned by the three dimensions.

If we choose only to use this component, then this 1D direction forms the latent space:

- All 3D points can be boiled down to this line,
- this line can give an approximation to all 3D points.

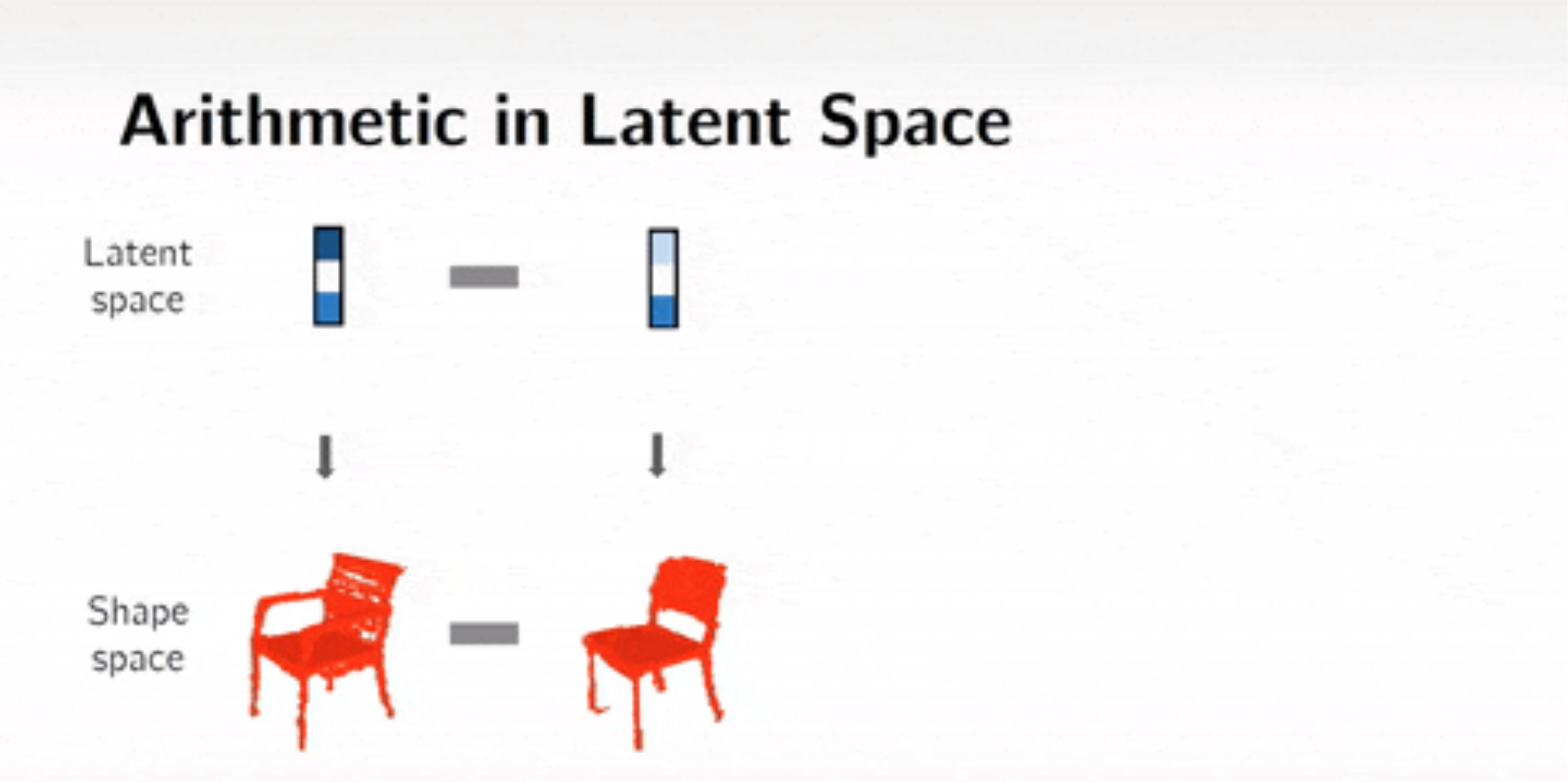
It works best, if most data points are along this line. But in high dimensional real data, this is almost always the case.

This is a linear example in low dimensionality.
Typically, ML-problems are non-linear and in high dimensionality. Therefore, the latent spaces can also have significant dimensionality, though it should of course always have a (much) lower dimensionality than the problem itself.

Latent space illustration

The below animation shows how 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 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:

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.

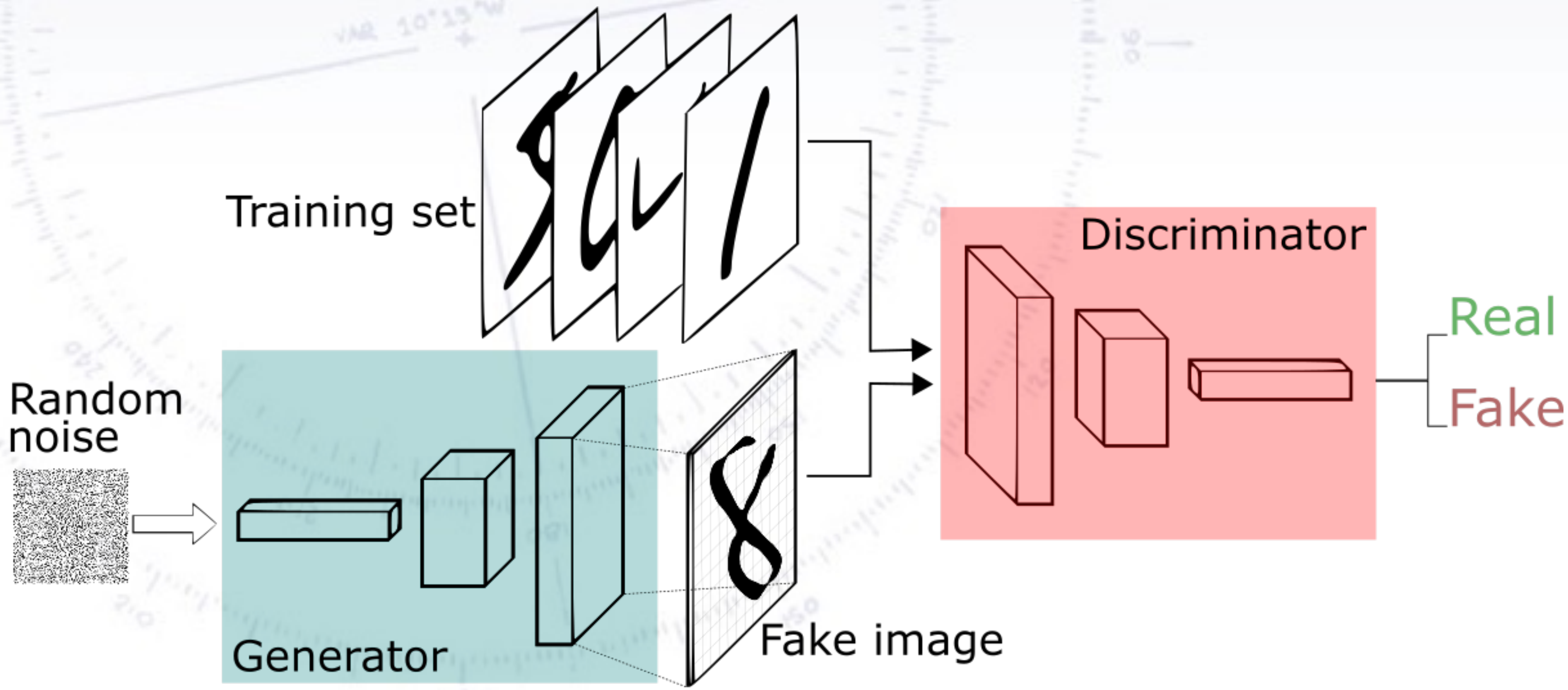
"The coolest idea in machine learning in the last twenty years"

[Yann LeCun, French computer scientist]

GAN drawing

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

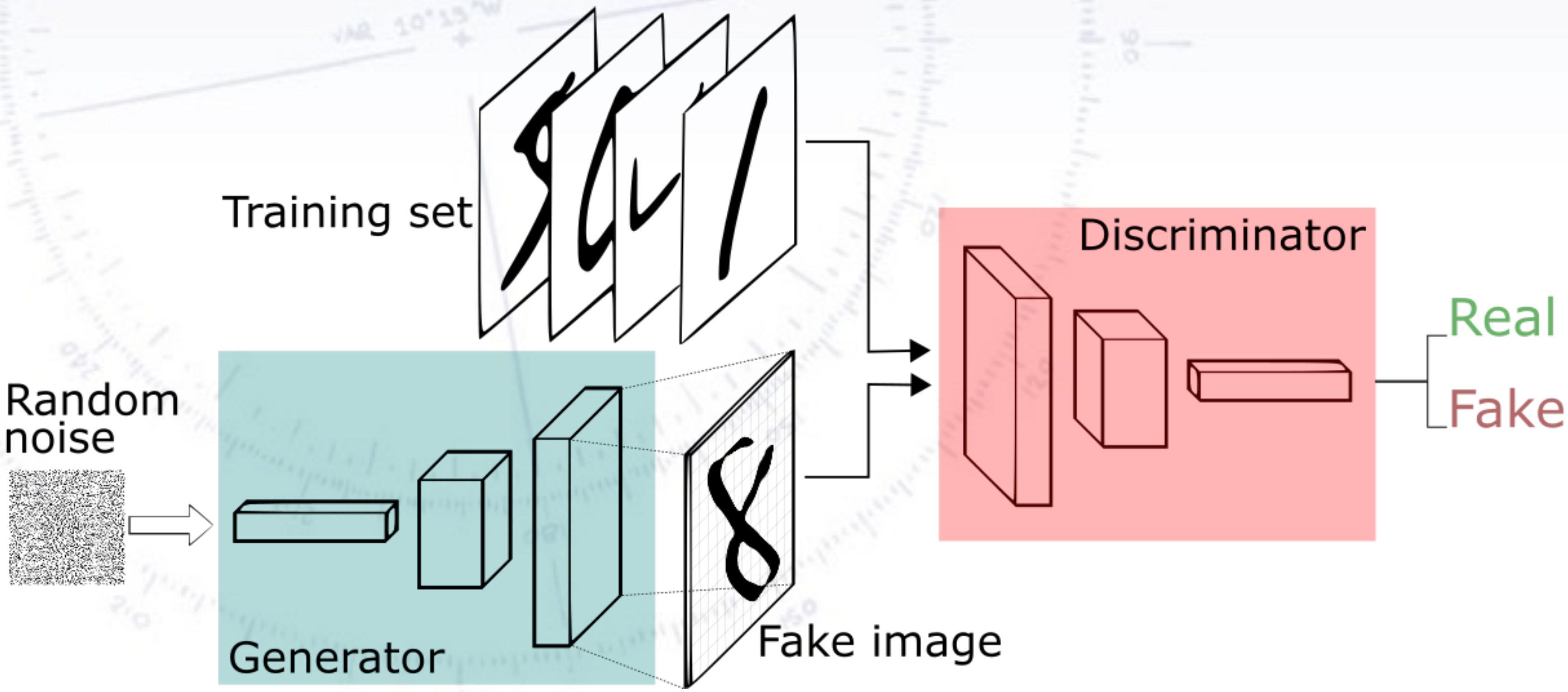
Given a large training set, you can ask you GAN to produce numbers. At first it will do poorly, but as it is "punished" by the discriminator, it improves, and at the end it might be able to produce numbers of **equal quality to real data**:



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:

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



GANs producing face images

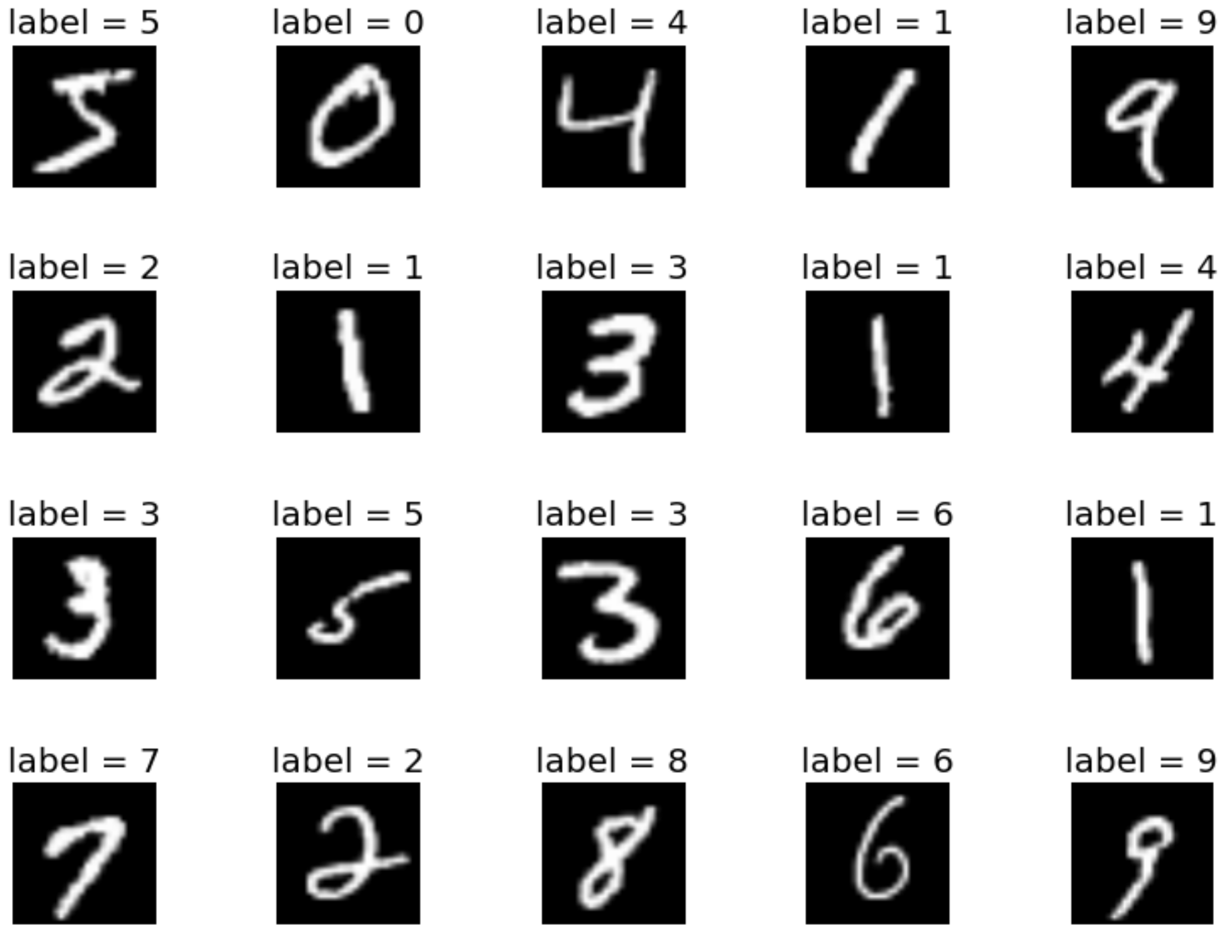
In 2017, Nvidia published the result of their "AI" GANs for producing celebrity faces. There is of course a lot of training data… here are the results:

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

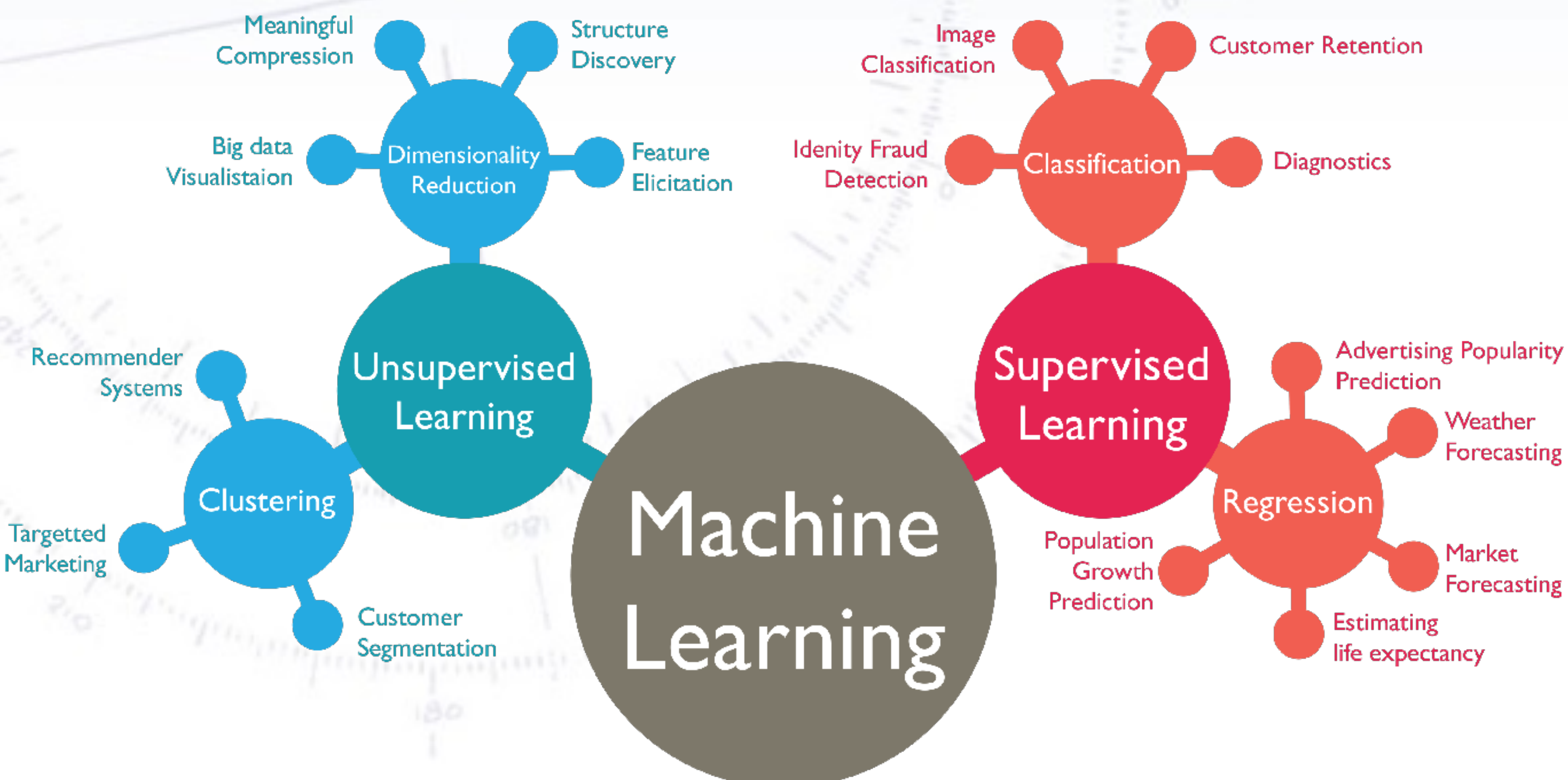
A "famous case" has been hand written numbers. The data c… 8x28 gray scale images of numbers. While that spans a large s… is probably (surely!) much smaller, as far fro… d intensities are present.

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

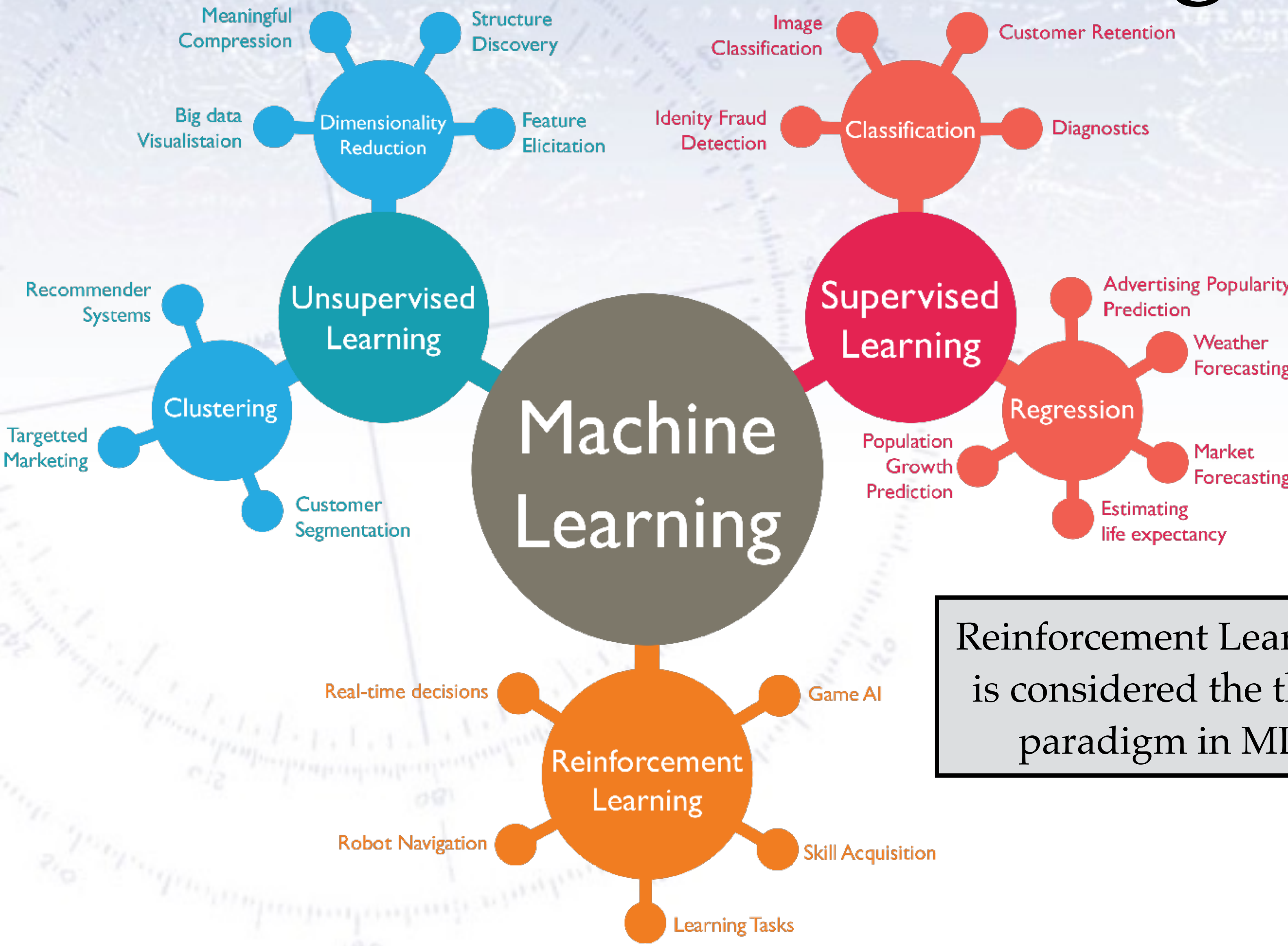
Reinforcement Learning

Classification vs. Regression
Unsupervised learning vs. supervised

Machine Learning can be supervised (you have correctly labelled examples) or unsupervised (you don't)… [or reinforced]. Following this, one can be using ML to either classify (is it A or B?) or for regression (estimate of X).



Reinforcement Learning



Reinforcement Learning is considered the third paradigm in ML.

Reinforcement Learning

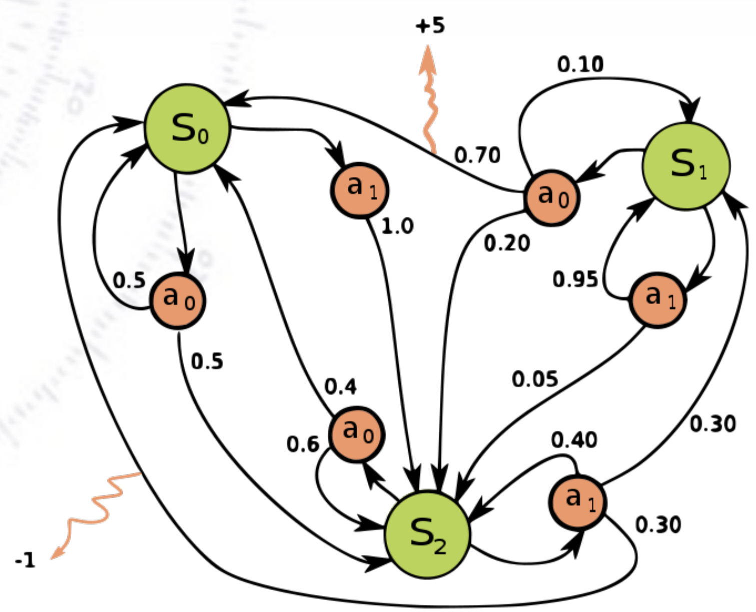
Reinforcement Learning (RL) does not need data per se, but rather an environment/set of rules in which it needs to optimise it's actions/behaviour.

In doing so, the RL needs to find a balance between exploration (of uncharted territory) and exploitation (of current knowledge).

The environment can be formulated as a Markov Decision Process (MDP), as shown below.

Reinforcement Learning does not assume knowledge of the MDP (i.e. it doesn't know what environment it is in - all it needs is a score).

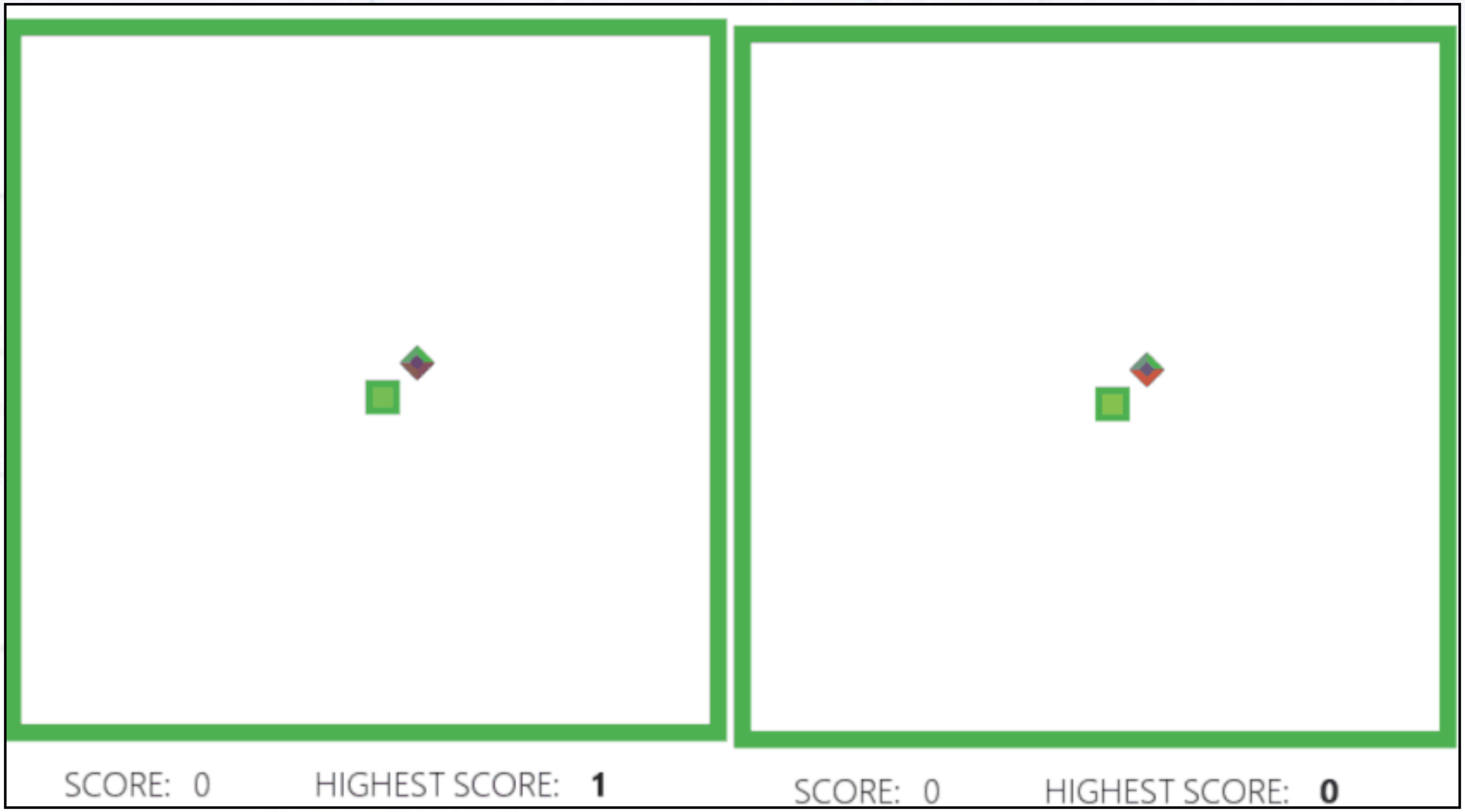
And typically RF has great success in (potentially very) large environments, such as "real life".



The Snake/Worm Game

A classic "old school" video game is Snake (or Worm) Game, which due to its simplicity has been made in 100s of versions since the first inception in 1976.

Below is the untrained AI (left) and the same AI after training for 150 games.

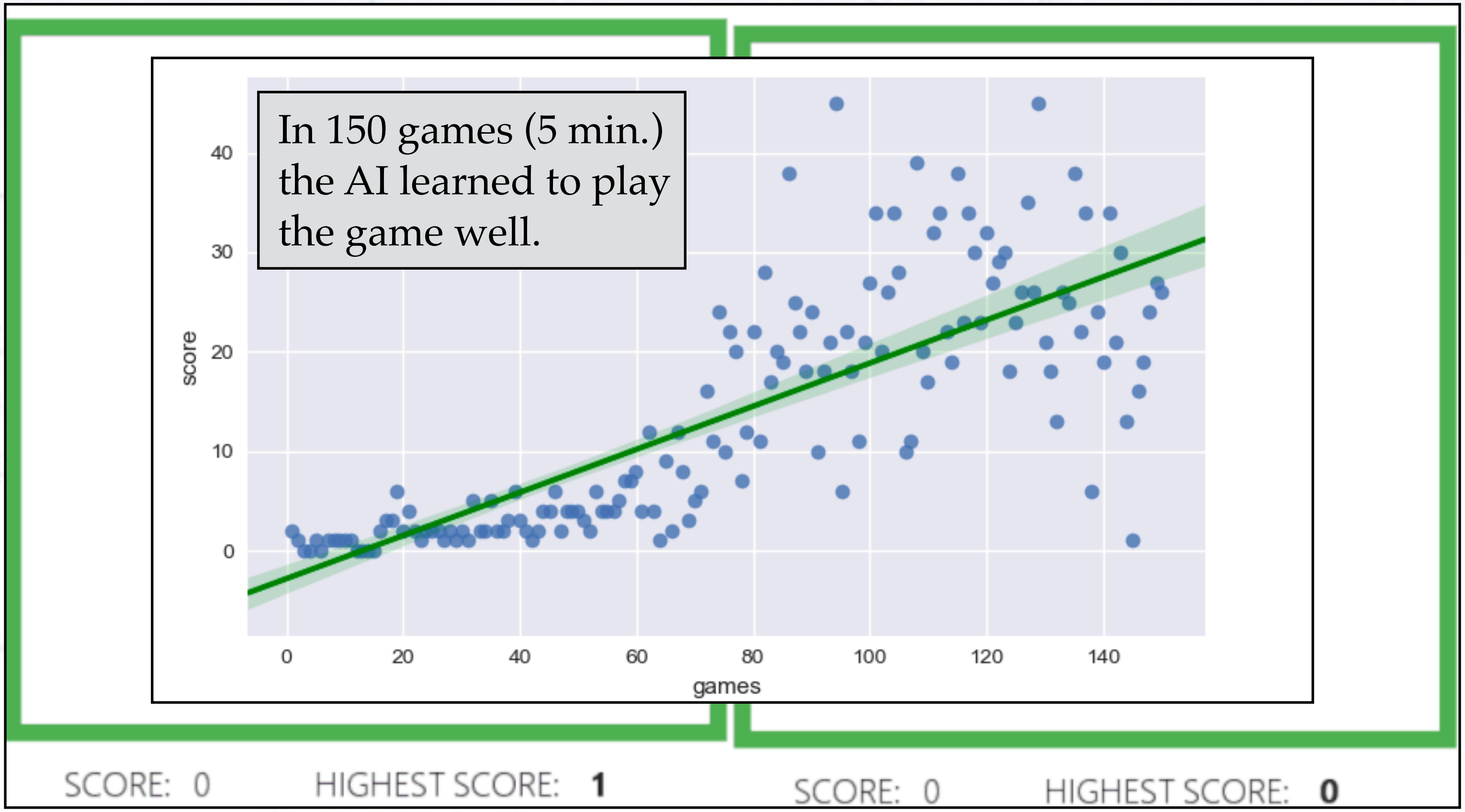


Towards Data Science: How to teach AI to play Games

The Snake/Worm Game

A classic "old school" video game is Snake (or Worm) Game, which due to its simplicity has been made in 100s of versions since the first inception in 1976.

Below is the untrained AI (left) and the same AI after training for 150 games.



Towards Data Science: How to teach AI to play Games

One program to rule them all

In December 2018, AlphaZero was introduced to play three classic strategy board games…

A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play

David Silver[1,2,*,†], Thomas Hubert[1,*], Julian Schrittwieser[1,*], Ioannis Antonoglou[1], Matthew Lai[1], Arthur Guez[1], Marc Lanctot[1], Laurent Sifre[1], Dharshan Kumaran[1], Thore Graepel[1], Timothy Lillicrap[1], Karen Simonyan[1], Demis Hassabis[1,†]

[1]*DeepMind, 6 Pancras Square, London N1C 4AG, UK.*

[2]*University College London, Gower Street, London WC1E 6BT, UK.*

↵[†]Corresponding author. Email: davidsilver@google.com (D.S.); dhcontact@google.com (D.H.)

↵* These authors contributed equally to this work.

− Hide authors and affiliations

Science 07 Dec 2018:
Vol. 362, Issue 6419, pp. 1140-1144
DOI: 10.1126/science.aar6404

One program to rule them all

In December 2018, AlphaZero was introduced to play three classic strategy board games…

A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play

David Silver[1,2,*,†], Thomas Hubert[1,*], Julian Schrittwieser[1,*], Ioannis Antonoglou[1], Matthew Lai[1], Arthur Guez[1], Marc Lanctot[1], Laurent Sifre[1], Dharshan Kumaran[1], Thore Graepel[1], Timothy Lillicrap[1], Karen Simonyan[1], Demis Hassabis[1,†]

[1]*DeepMind, 6 Pancras Square, London N1C 4AG, UK.*

[2]*University College London, Gower Street, London WC1E 6BT, UK.*

↵[†]Corresponding author. Email: davidsilver@google.com (D.S.); dhcontact@google.com (D.H.)

↵* These authors contributed equally to this work.

– Hide authors and affiliations

Science 07 Dec 2018:
Vol. 362, Issue 6419, pp. 1140-1144
DOI: 10.1126/science.aar6404

After four hours of training it beat the best chess program in the world at the time: 72 draws, 28 wins, and… 0 losses.

Within 24 hours AlphaZero achieved a superhuman level of play in ALL three games by defeating world-champion programs…. using only the rules!