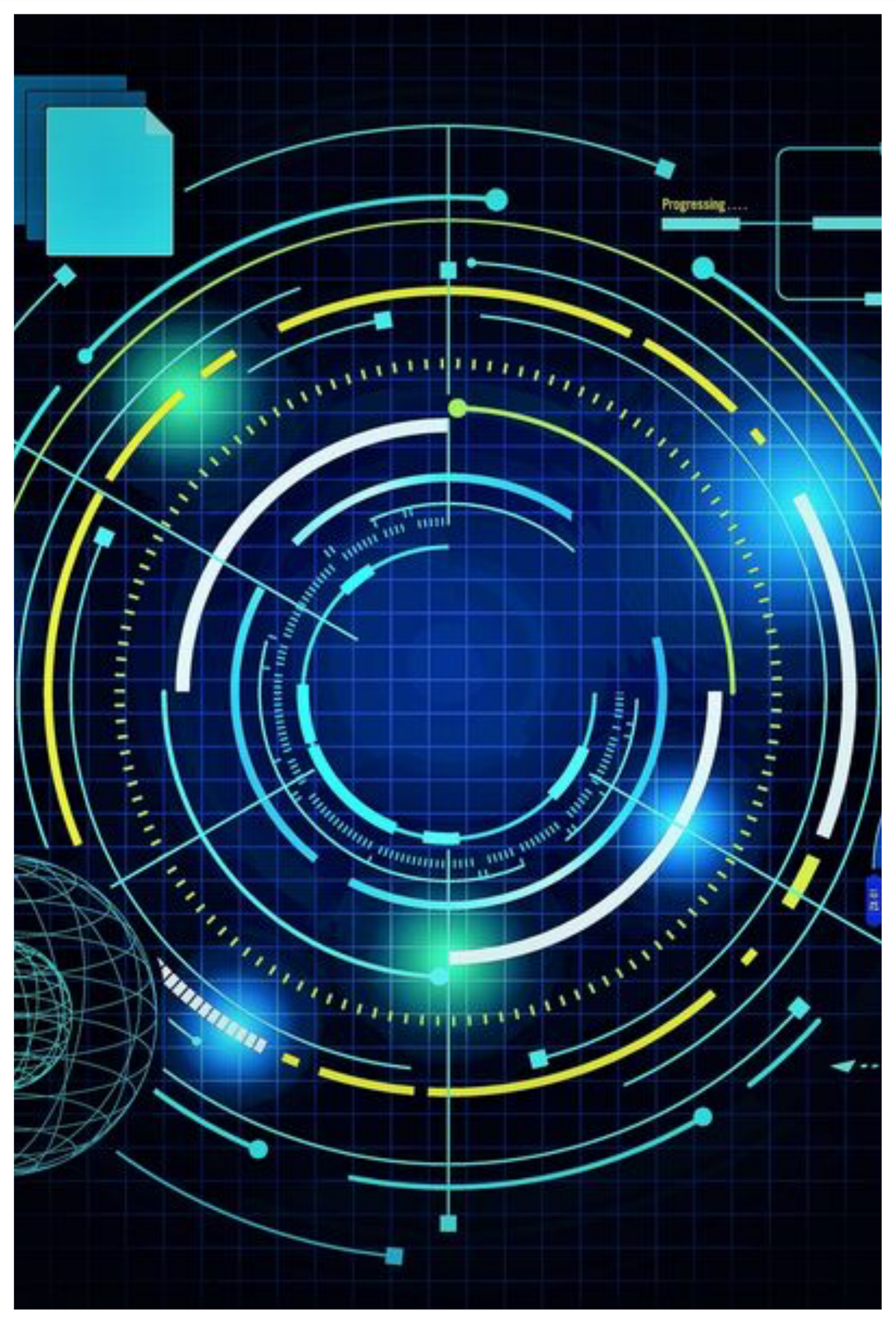
Introduction to Deep Learning

Veronica Sanz
Universitat de Valencia - IFIC (Spain)
& Sussex University (UK)

@SoS '22



Today, we will talk about

Machine Learning techniques:

Supervised

Unsupervised

Reinforcement

Transfer

Time-evolution

Generative

My aim today is
to introduce a range of uses of ML in Physics & beyond
and some concepts and keywords to help you start looking

Tomorrow: hands-on!

Human vs Machine Learning

Human learning

repeat and improve on a task

Previous experience

Human learning

repeat and improve on a task

predict the evolution of a situation

Previous experience

Human learning

Previous experience

repeat and improve on a task

predict the evolution of a situation

discover unknown relations

Human learning

Previous experience

repeat and improve on a task

predict the evolution of a situation

discover unknown relations

choose the option that maximises return

Human learning

Previous experience

repeat and improve on a task

predict the evolution of a situation

discover unknown relations

choose the option that maximises return

imagine new possibilities

VERY IMPRESSIVE, YET

human learning is limited by

our personal viewpoint,

our collective intelligence (*newspeak?)*

& our inherent capacity to process information

(amount , speed, level of detail)

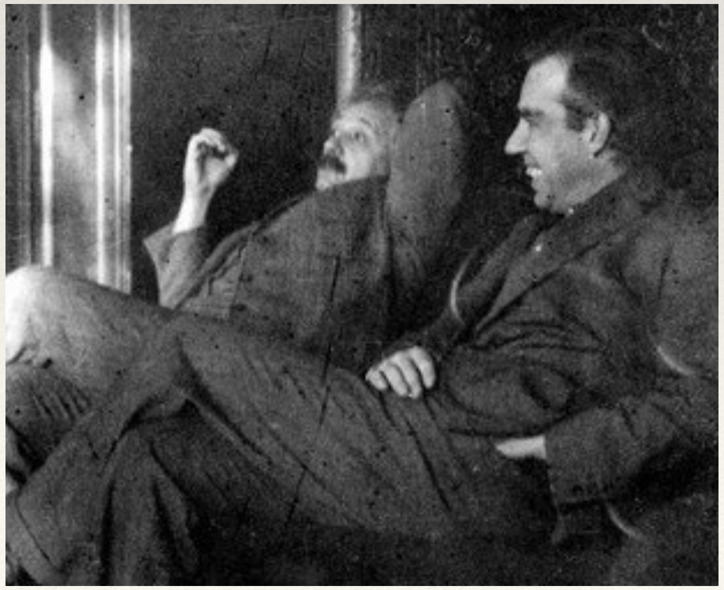
ON THE OTHER HAND

the ultimate limitations of machine learning

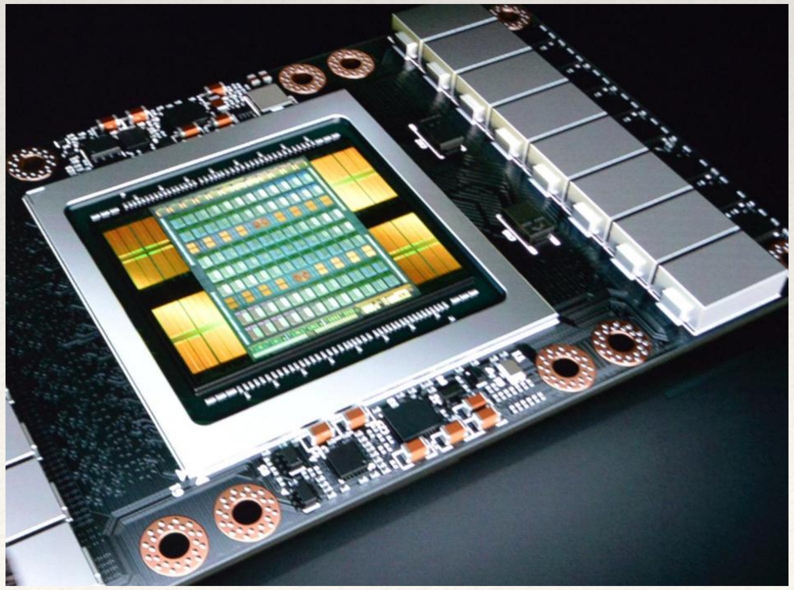
are unknown (if they do exist)

CPU-> GPU, TPU, FPGA, IPU -> …

Quantum Computing, Neurophotonics…



VS



Machine learning

Previous experience

repeat and improve on a task

SUPERVISED MACHINE LEARNING

predict the evolution of a situation

TIME-SERIES LEARNING

discover unknown relations

CLUSTERING/UNSUPERVISED

choose the option that maximises return

REINFORCEMENT LEARNING

imagine new possibilities

GENERATIVE AI

Machine learning

TOMORROW's HANDS-ON

Previous experience

repeat and improve on a task

SUPERVISED MACHINE LEARNING

predict the evolution of a situation

TIME-SERIES LEARNING

discover unknown relations

CLUSTERING/UNSUPERVISED

choose the option that maximises return

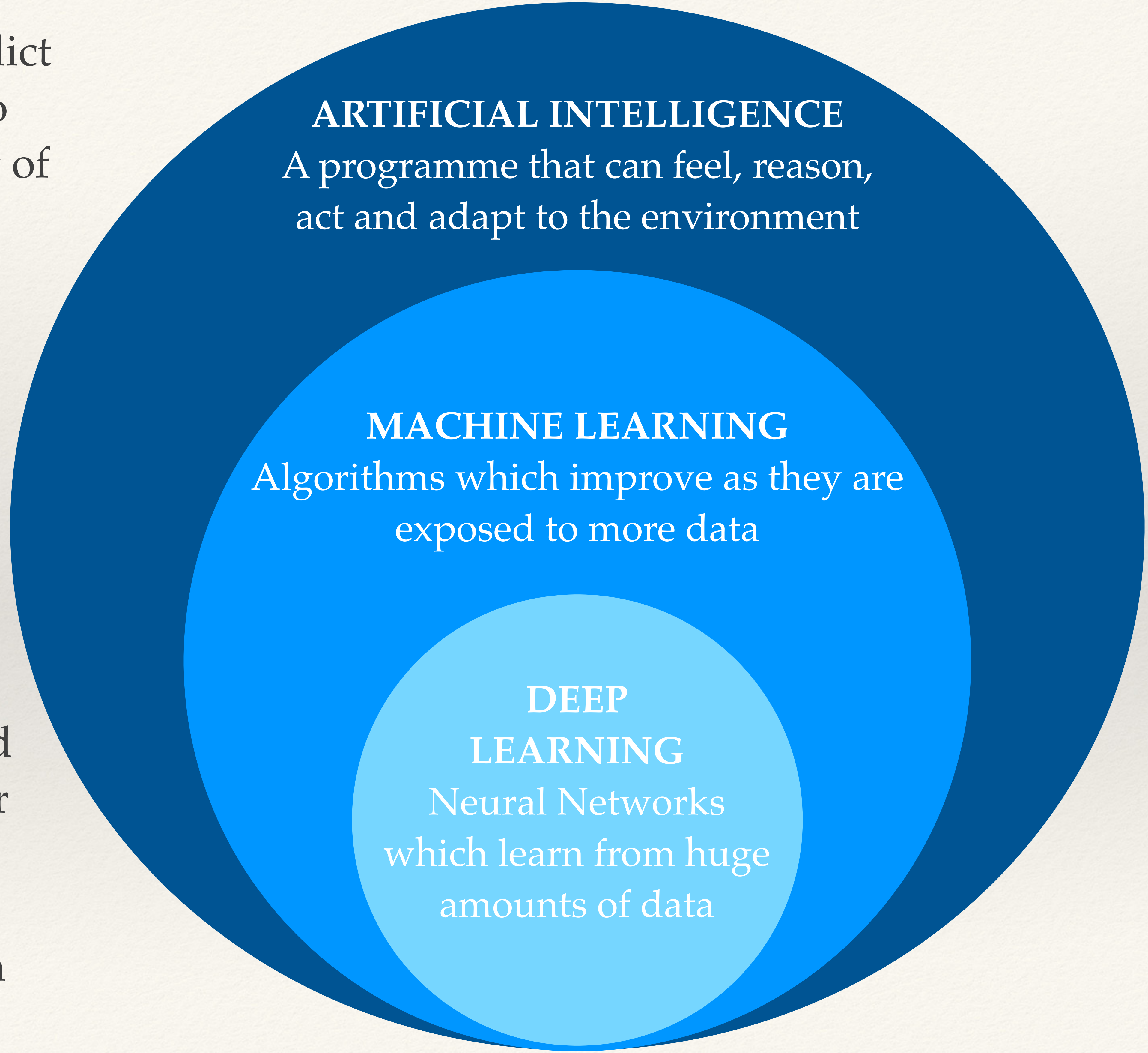
REINFORCEMENT LEARNING

imagine new possibilities

GENERATIVE AI

This technology is truly *disruptive*

we are unable to predict how fast is going to evolve and the extent of its applications



new algorithms and applications appear every day, and this tendency does not seem to slow down

Learning by example: Supervised ML

repeat and improve on a task

A basic task: good or bad?

Is it a crocodile?

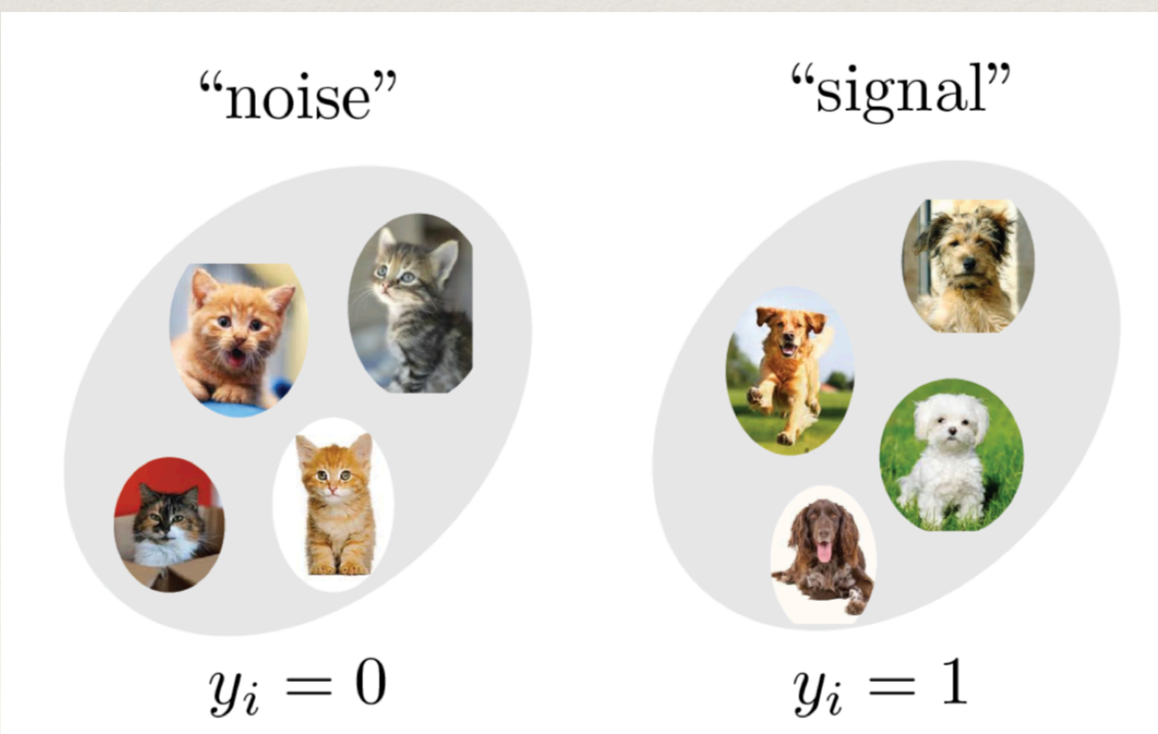
Yes / No answer

A basic task: good or bad?

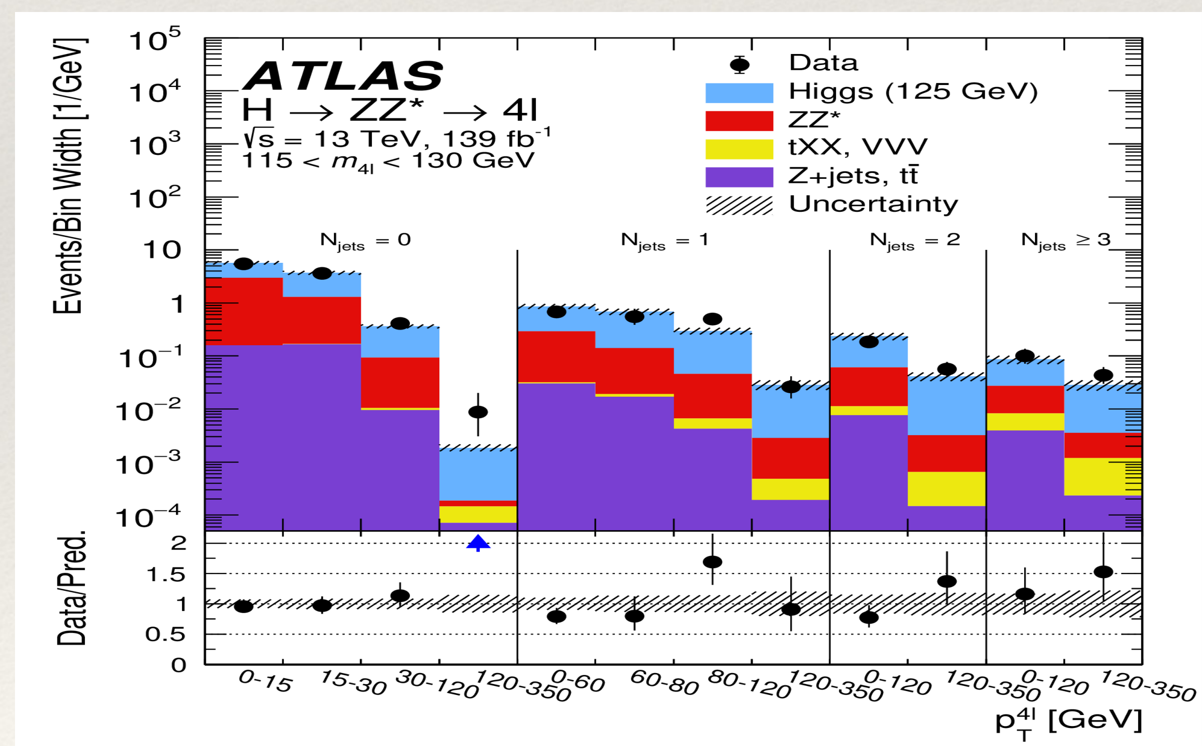
Is it a crocodile?

Yes/No answer

To learn, dataset $\mathcal{D}(x_i, y_i)$ $y \in \{0, 1\}$ with labels



Cat or dog?



Is this New Physics?

The simplest classification problem

Interpret the output of this transformation as a binomial probability

$$P(y_i = 1) = f(\mathbf{x}_i^T \mathbf{w}) = 1 - P(y_i = 0).$$

e.g. event b-tagged or not,
event new physics or not

logistic regression: probability datapoint *x_i* as true or false

We define a cost function for this problem using
Maximum Likelihood Estimation (MLE)

$$P(\mathcal{D}|\mathbf{w}) = \prod_{i=1}^{n} \left[f(\mathbf{x}_i^T \mathbf{w})\right]^{y_i} \left[1 - f(\mathbf{x}_i^T \mathbf{w})\right]^{1-y_i}$$

prob dataset $\mathcal{D}$ explained by
our *model w*

Binary cost function: Cross-entropy

then log-likelihood is

$$l(\mathbf{w}) = \sum_{i=1}^{n} y_i \log f(\mathbf{x}_i^T \mathbf{w}) + (1 - y_i) \log \left[1 - f(\mathbf{x}_i^T \mathbf{w})\right]$$

best description: parameters *w maximize* the log-likelihood

$$\hat{\mathbf{w}} = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} y_i \log f(\mathbf{x}_i^T \mathbf{w}) + (1 - y_i) \log \left[1 - f(\mathbf{x}_i^T \mathbf{w})\right]$$

Cost function is then chosen to be
CROSS-ENTROPY

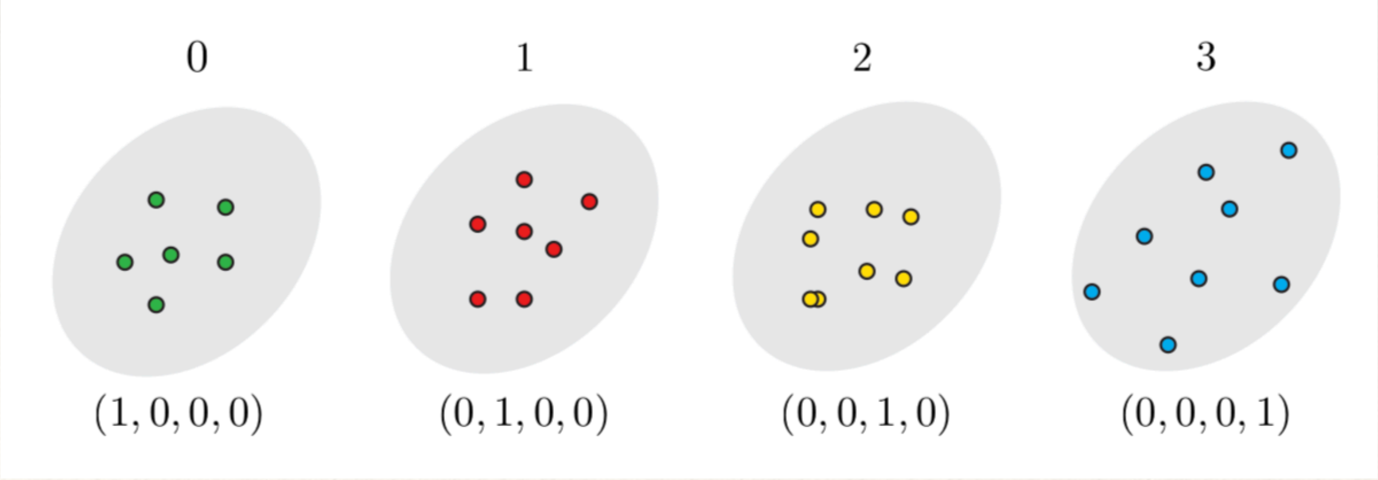
$$\mathcal{C}(\mathbf{w}) = -l(\mathbf{w})$$ +regularization

Logistic regression

Take a dataset (**X**,y) where y is binary
build a *cost function* = *cross-entropy*
*m*inimise it and find the parameters **w**

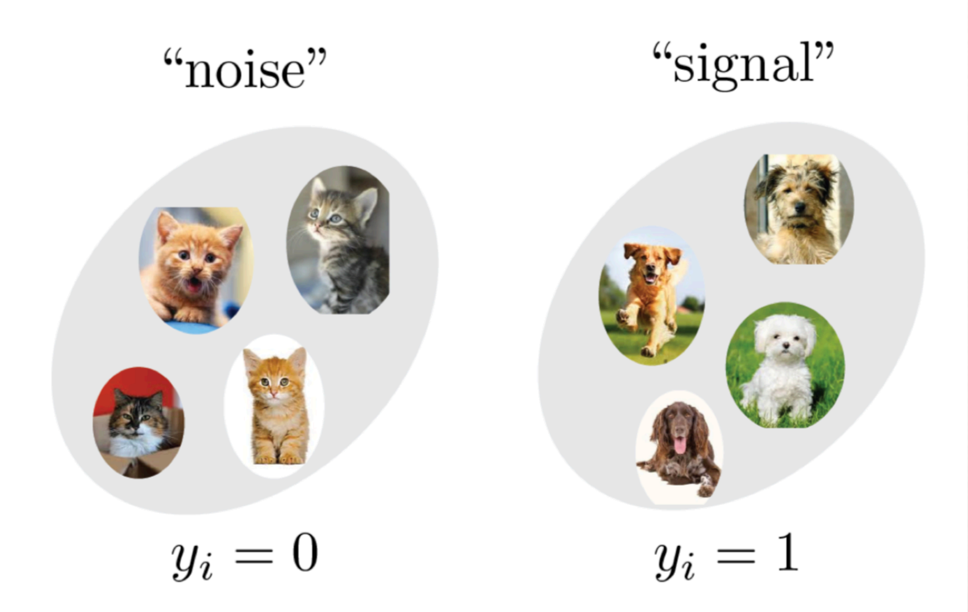
We can use L1 or L2 regularisation

Binary is an example of *categorical* outputs
When we have more than 2 categories, we use a cost function called
SOFTMAX REGRESSION (a generalization of cross-entropy)



often convenient to
describe the categories with
ONE-HOT vectors

Logistic regression: measures of performance



How do we measure performance in a classification task?

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

'We now can distinguish cats and dogs with 95% accuracy'
but maybe identifying cats is easier than dogs
you nearly always get cats right (98%),
but sometimes confuse a little dog for a cat (3%)

Moreover, in your learning you may want to specialise in id'ing dogs because want to make sure you don't miss any. You raise your threshold for them, paying a price cat performance

Logistic regression: measures of performance

So, let's say your focus is on dogs (you were bitten or had a bad experience)

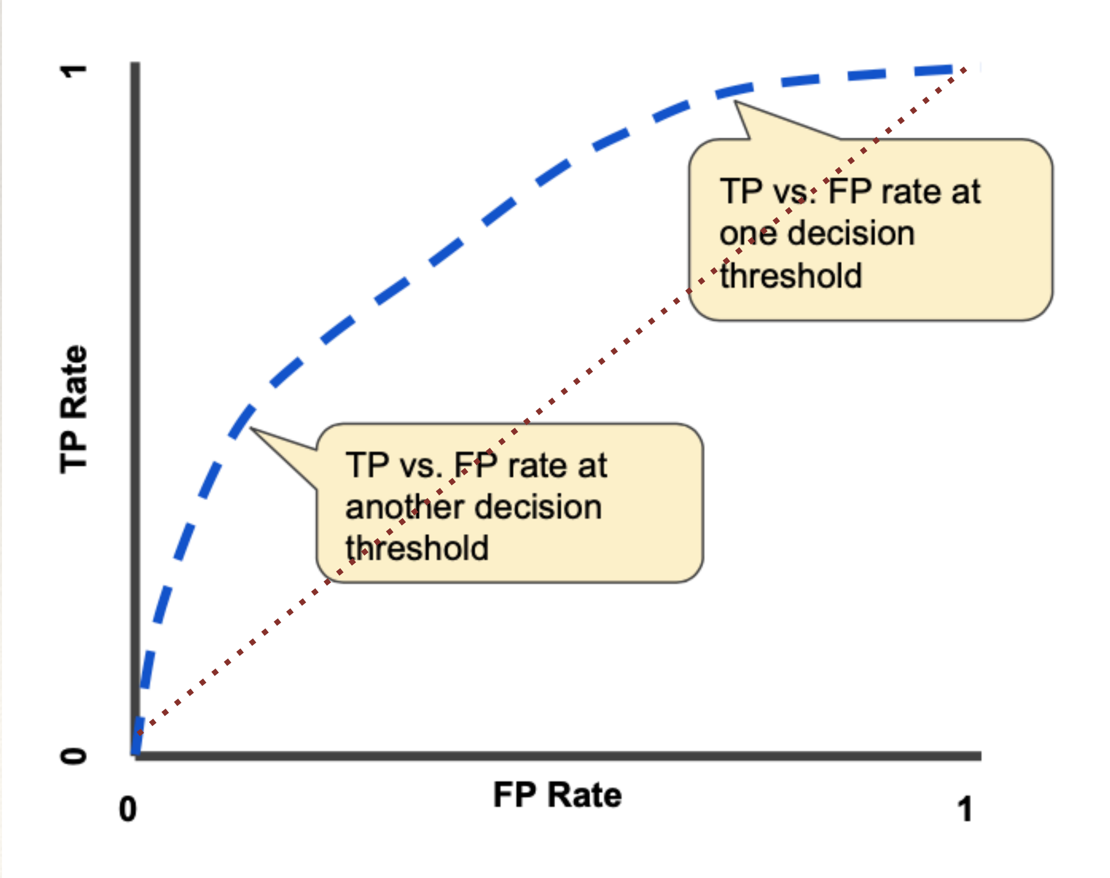
You would care about

True positives (**TP**) : how many dogs you id

False negatives (**FN**): how many you miss

False positives (**FP**): how many cats you confuse with dogs

True negatives (**TN**): cats you id



$$TPR = \frac{TP}{TP + FN}$$

$$FPR = \frac{FP}{FP + TN}$$

AUC = indicates goodness of learning

$$\text{Precision} = \frac{TP}{TP + FP}$$

how often when i say 'dog' is dog?

$$\text{Recall} = \frac{TP}{TP + FN}$$

proportion of dogs I id correctly

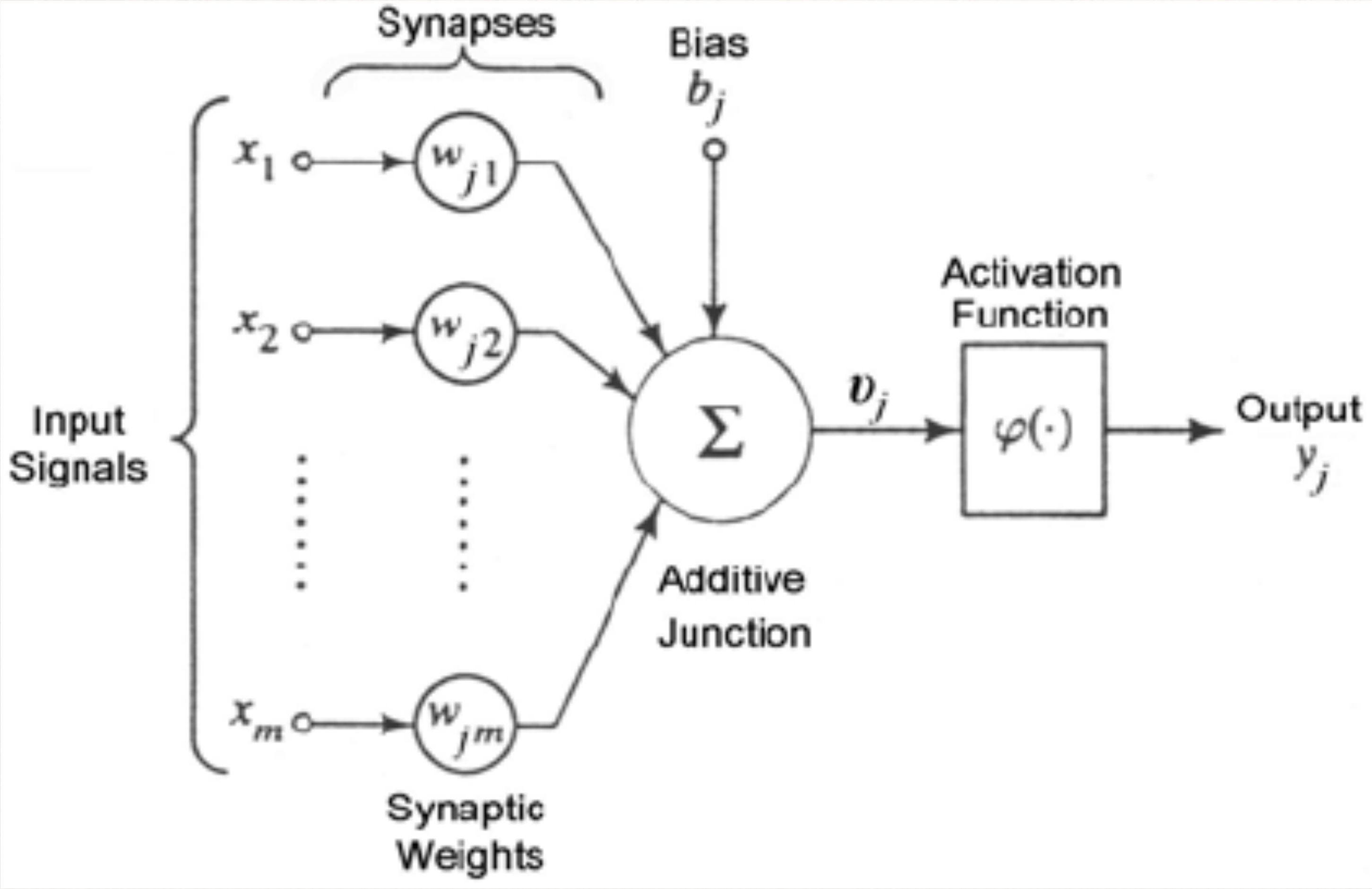
Neural Networks

Learning inspired by biology

Neural Networks (NNs)

A framework to develop AI, based on an architecture of *neurons*

ONE NEURON = BUILDING BLOCKS OF NNs



First, a linear transformation

$$z = w.x + b$$

Second, a non-linear function

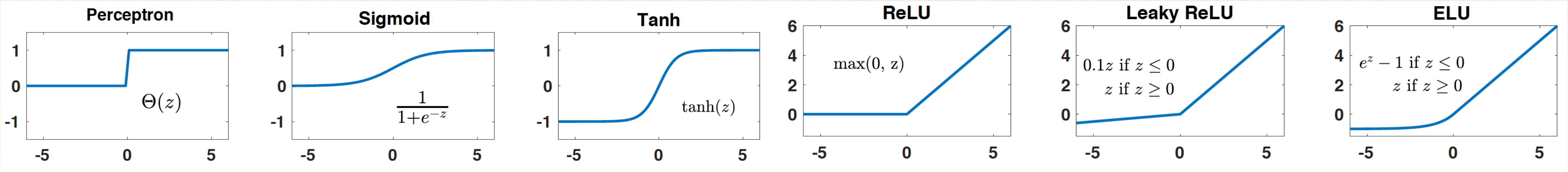
$$y = f(z)$$

y: **output, scalar**

(passes information, or not)

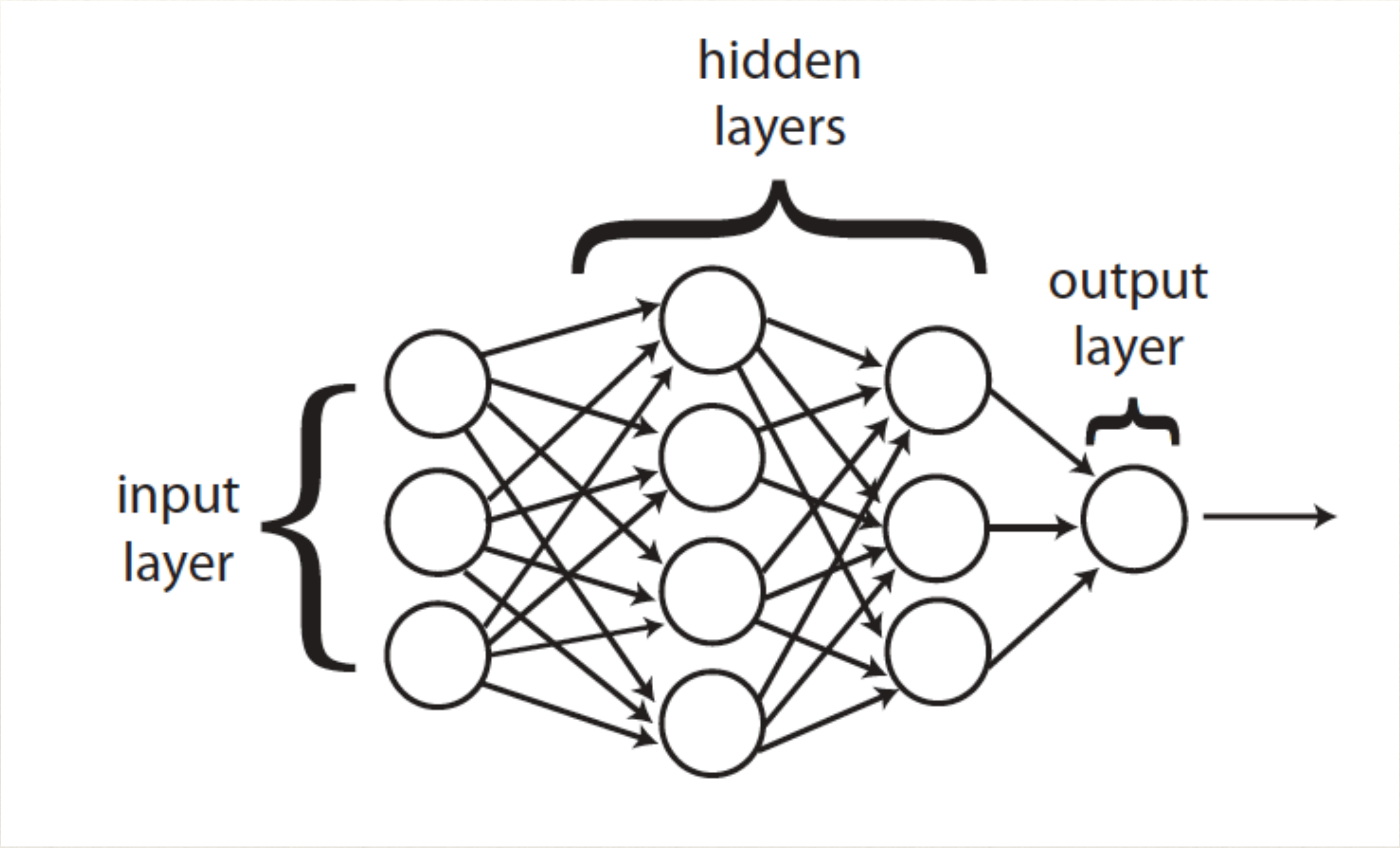
f: **activation function**

Examples of activation functions



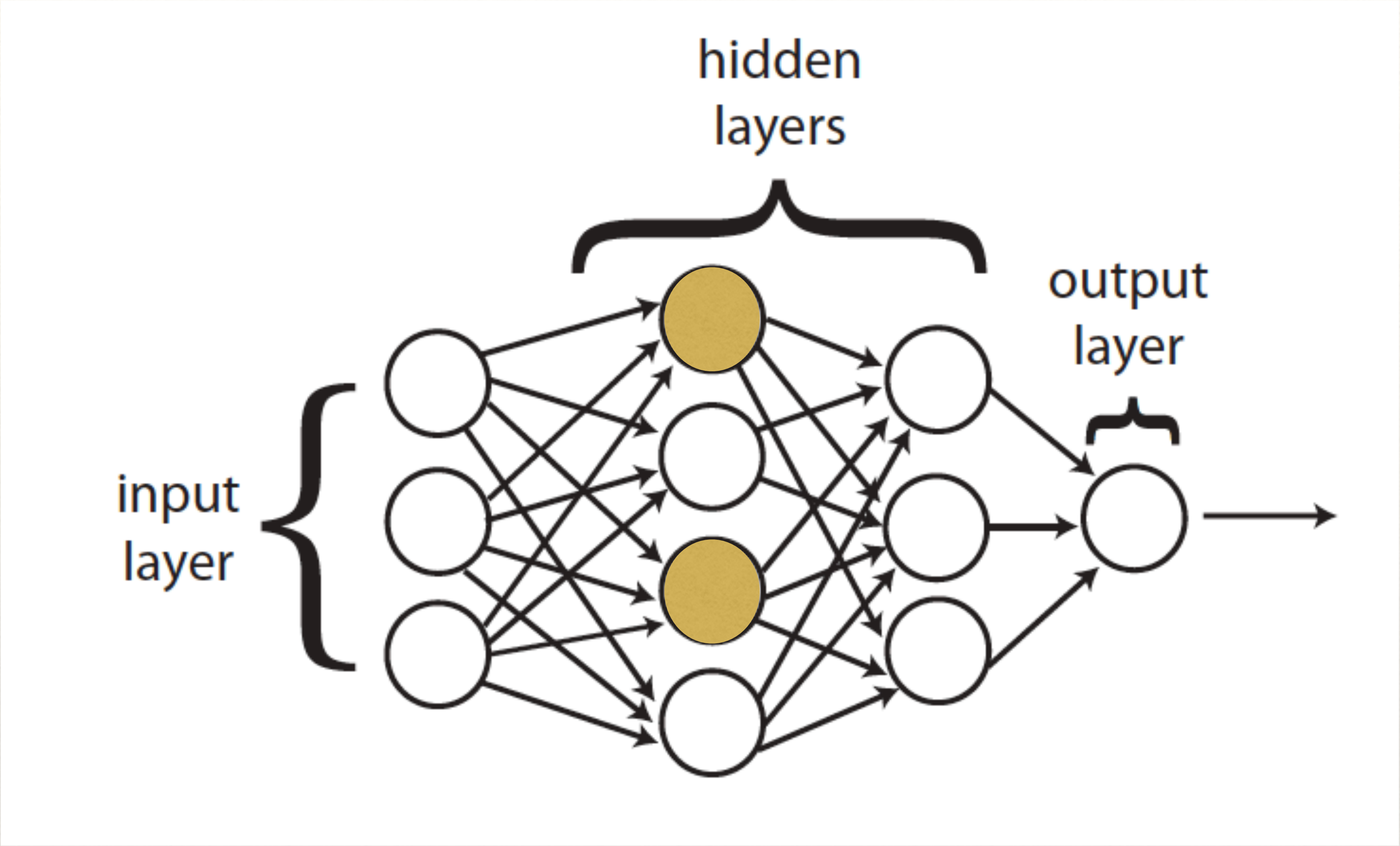
NN Architecture

Taking many neurons together, we can build an *architecture*



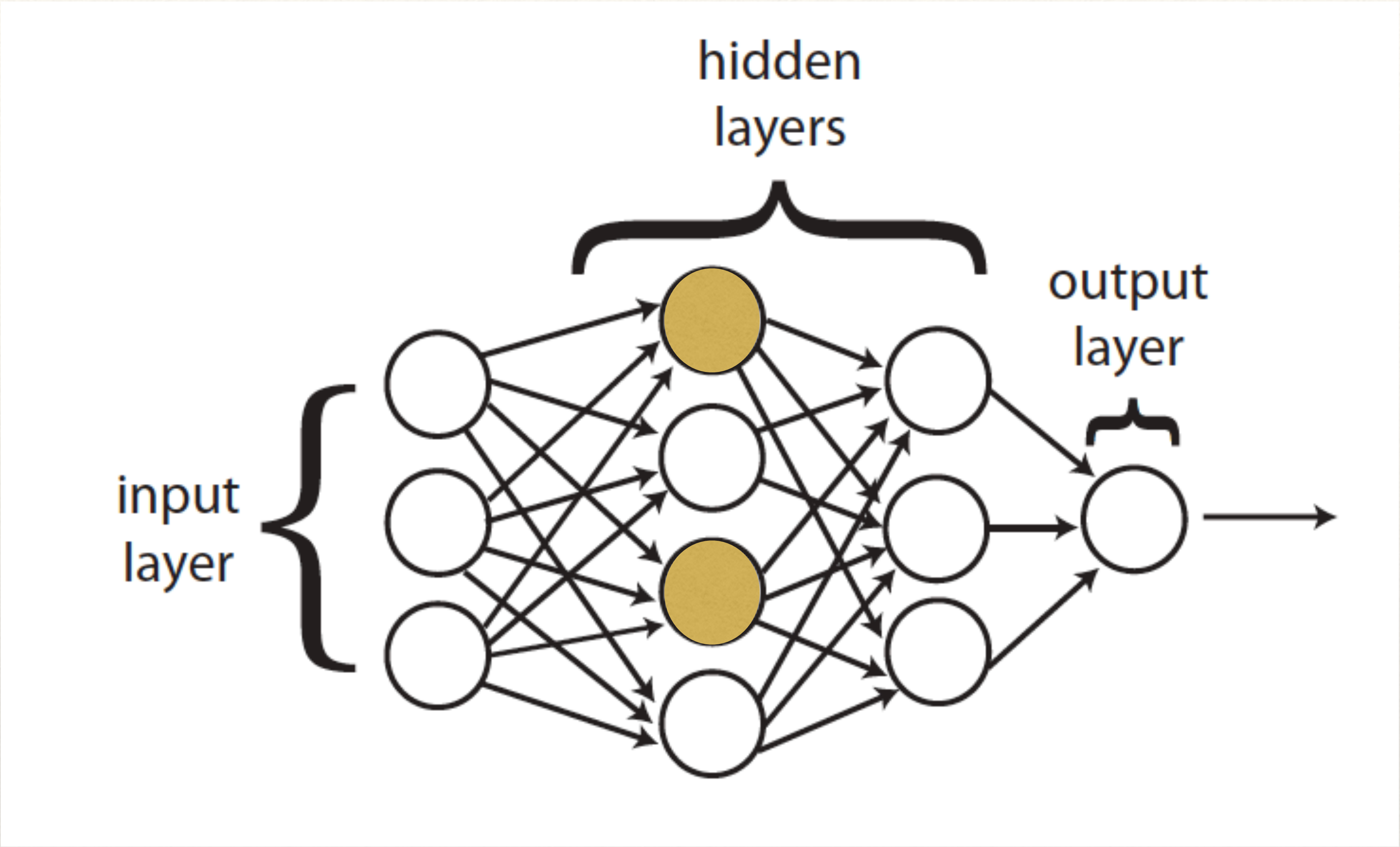
each circle is a neuron,
where the inputs (in-arrows) are transformed into output (out-arrows)
the outputs of each layer serve as input for the next

Why are we doing this?



This NN transforms
inputs (at the input layer) into an output (output layer)
by passing via the hidden layers
non-linear transformations of many non-linear transformations=
highly non-linear transformation of input into output

Why are we doing this?

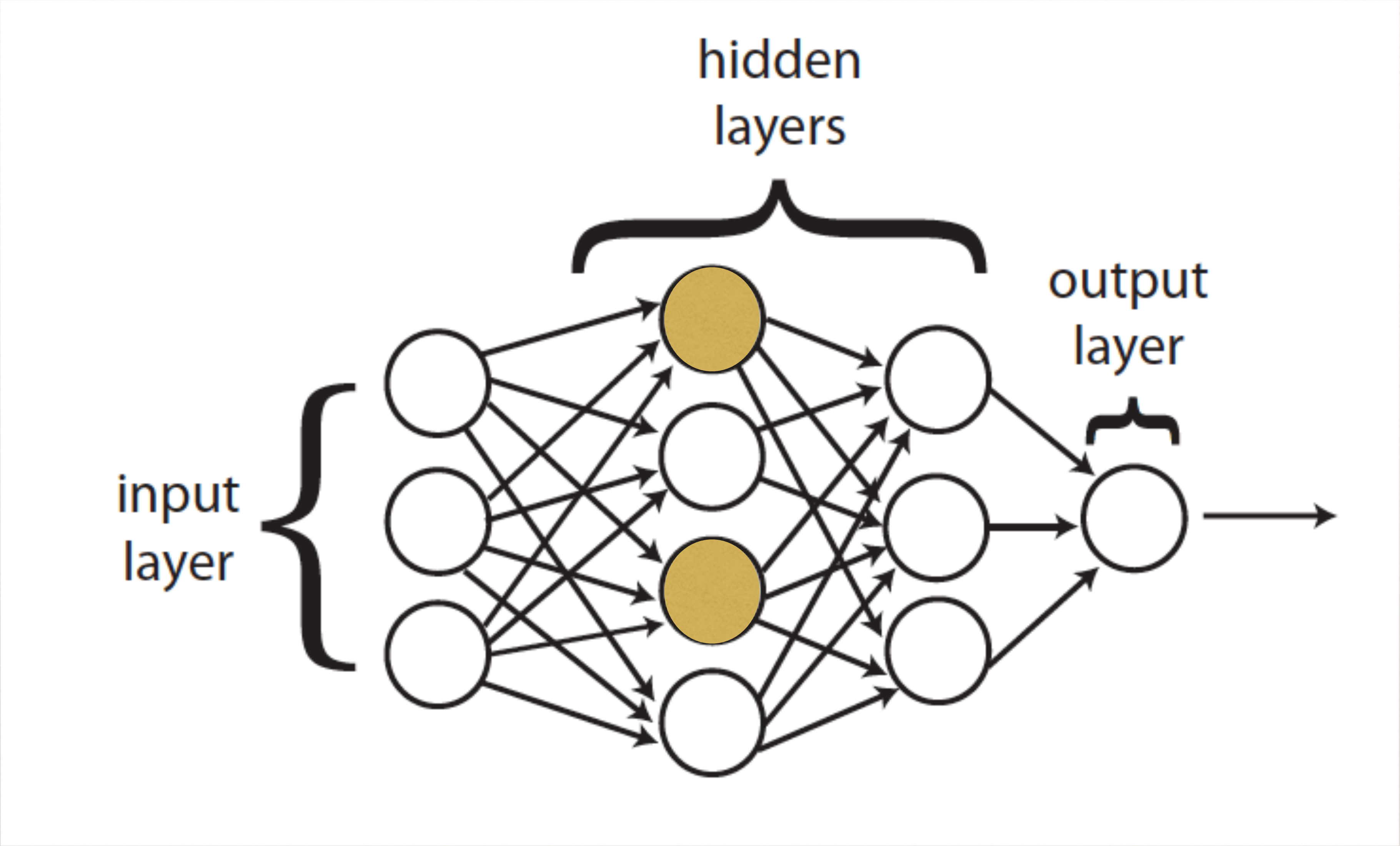


This NN transforms
inputs (at the input layer) into an output (output layer)

$$y(x)$$

which couldn't be captured by simple functional forms

Why are we doing this?



Neural Networks can model **complexity**

They have a high degree of expressivity

/ exhibit high representational power

More hidden layers=> more complex features

Deep learning, deep NN

Nowadays, Machine Learning is in the middle of a revolution: processing speed and storing capacity have increased enormously but **more importantly** the *way* machines learn has changed

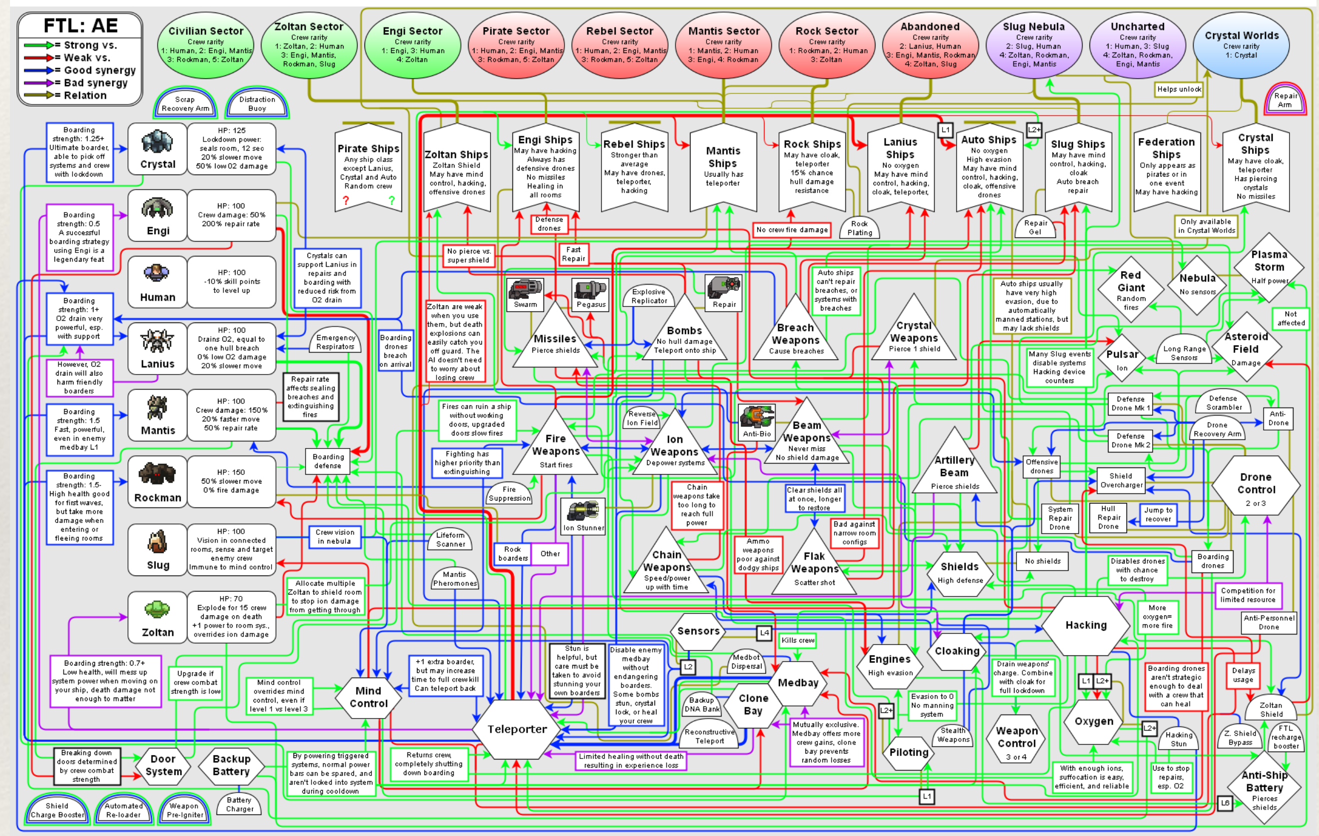
TRADITIONALLY

learning was limited to lines of code we (humans) were writing

```
if something_is_in_the_way is True:
    stop_moving()
else:
    continue_moving()
```

we can write *extremely complex* codes and the machine can improve in performing tasks

but the structure of *thought* behind decision making is human



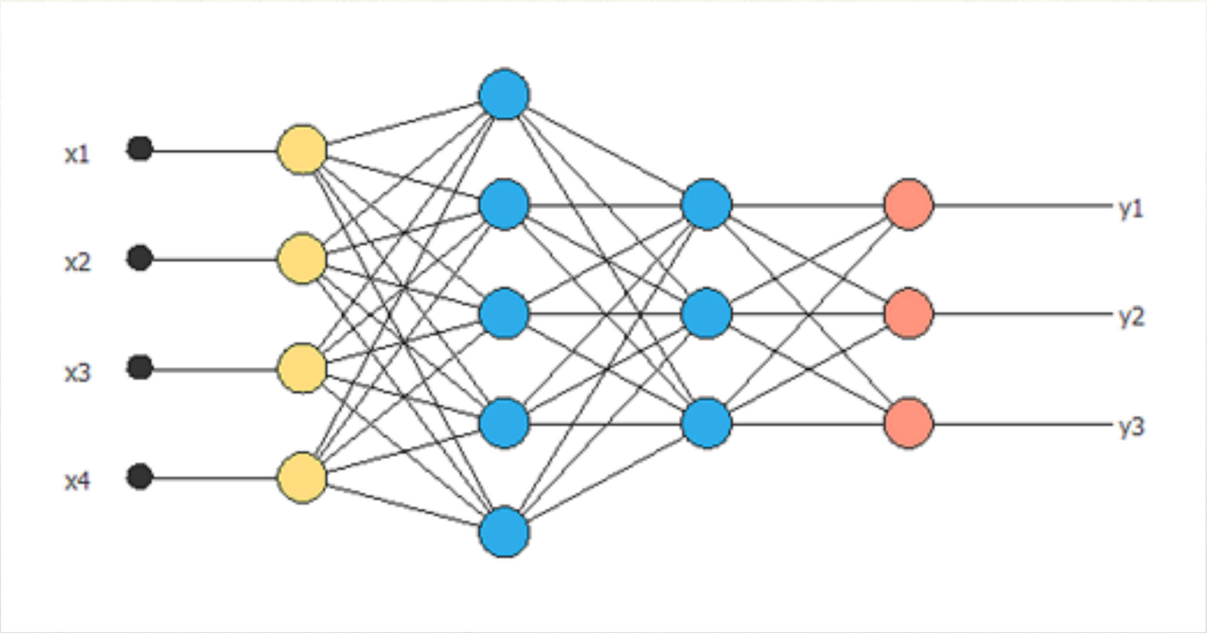
The Machine can't describe relations we haven't coded in

like a born-blind person who is asked to think of ***blue***

A new way of *thinking*: Neural Networks

Structures made of units called *neurons*
and organised by *layers*

inputs



outputs

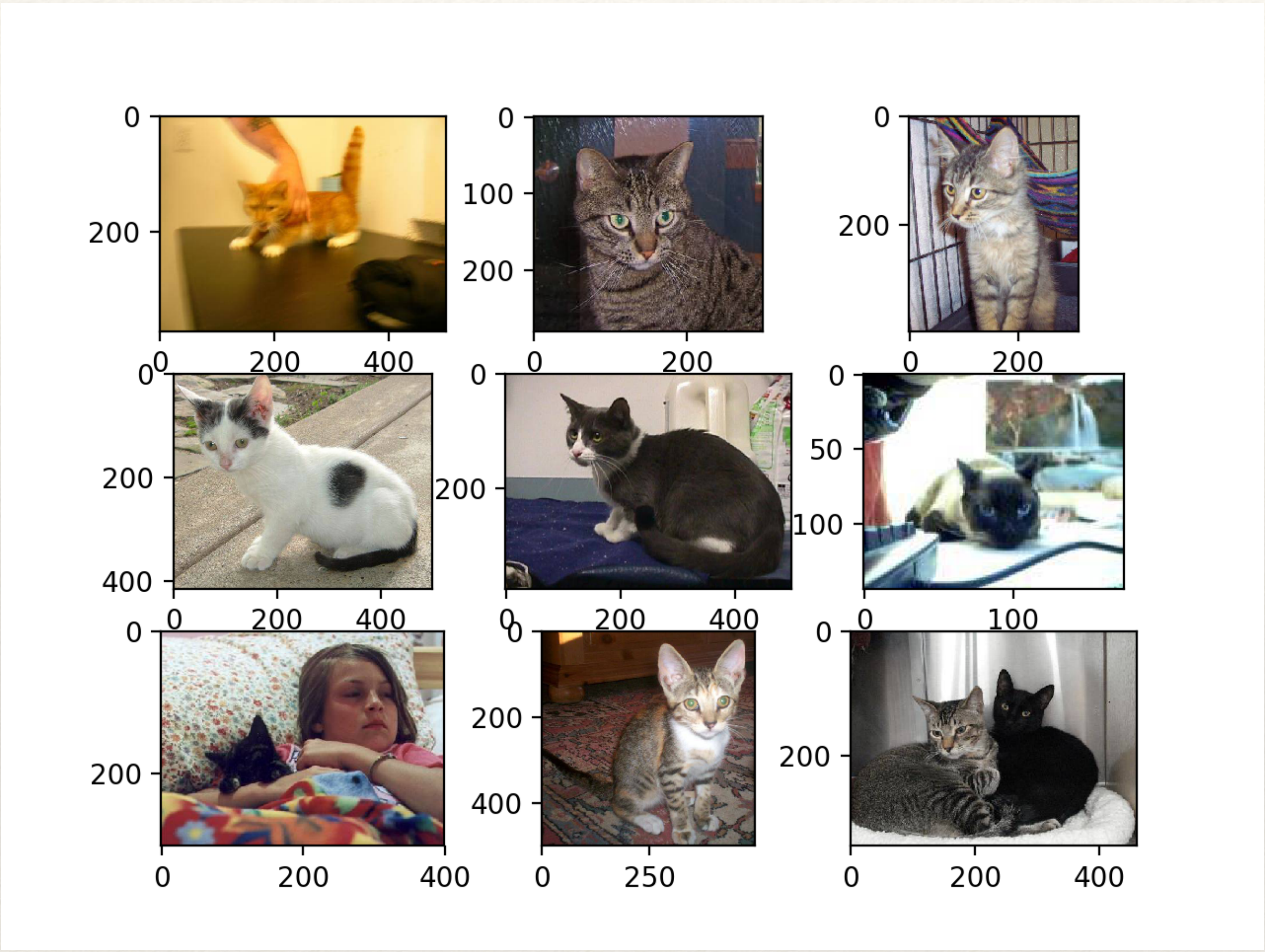
The network learns from data with no structured instructions

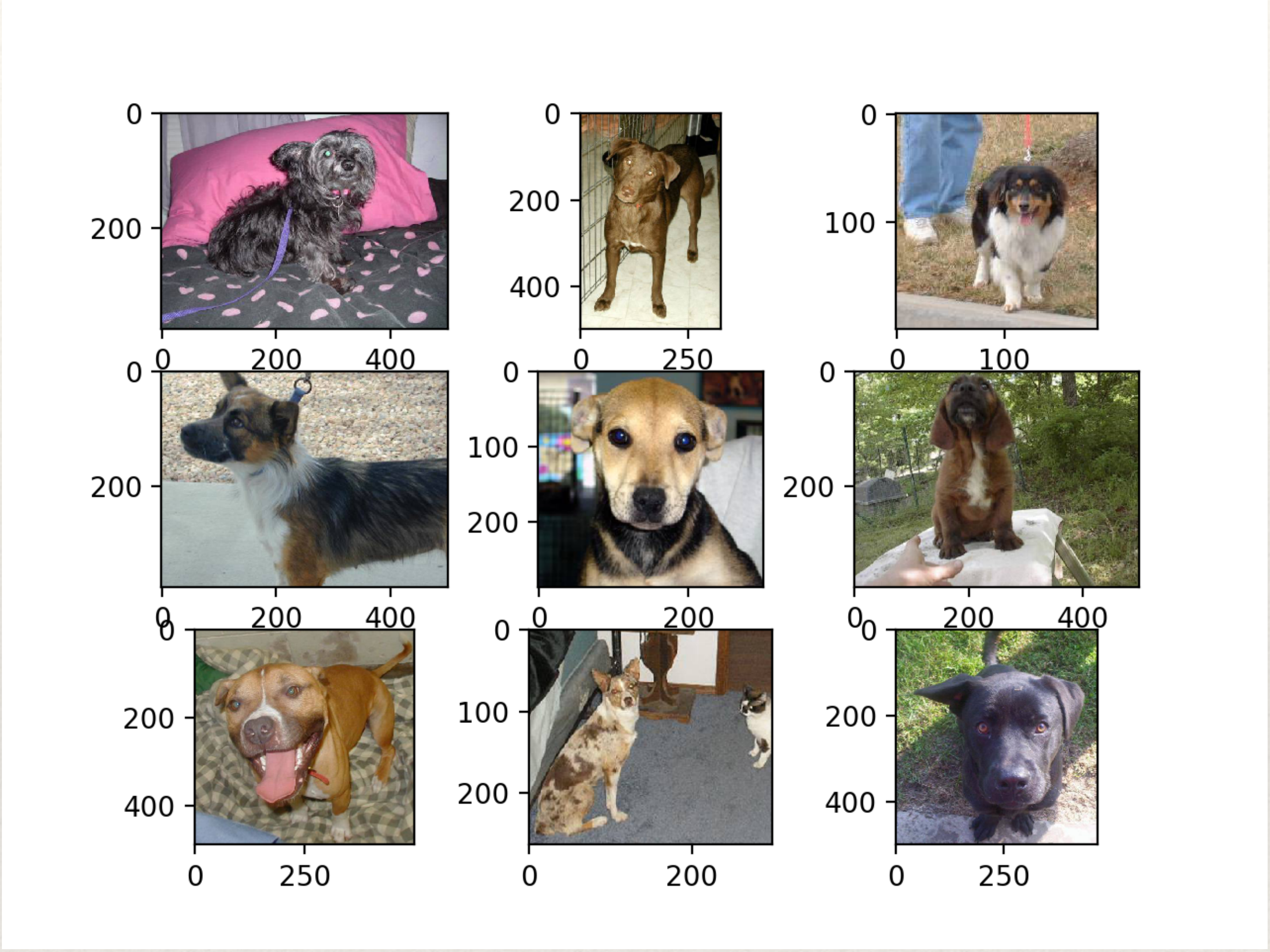
Neural networks are able to explore relations between inputs and
outputs which cannot be contained in lines of codes
their degree of expressivity is immense
and it is extremely fast
built from simple units and in a layered architecture

Complex features

images, speech : are complex

For example: cats / dogs

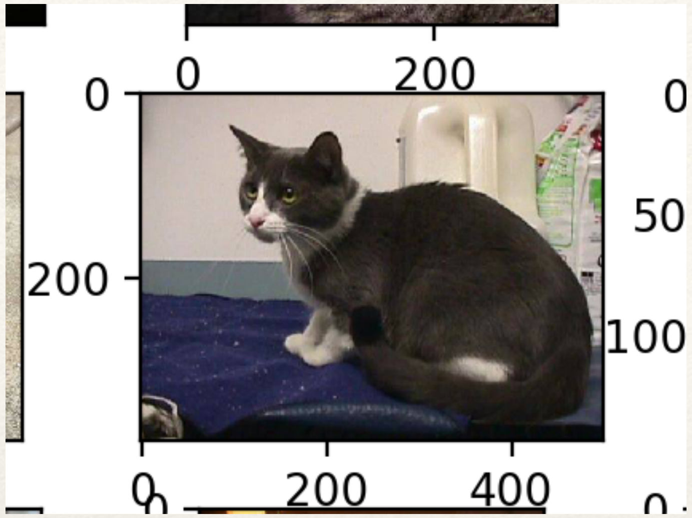


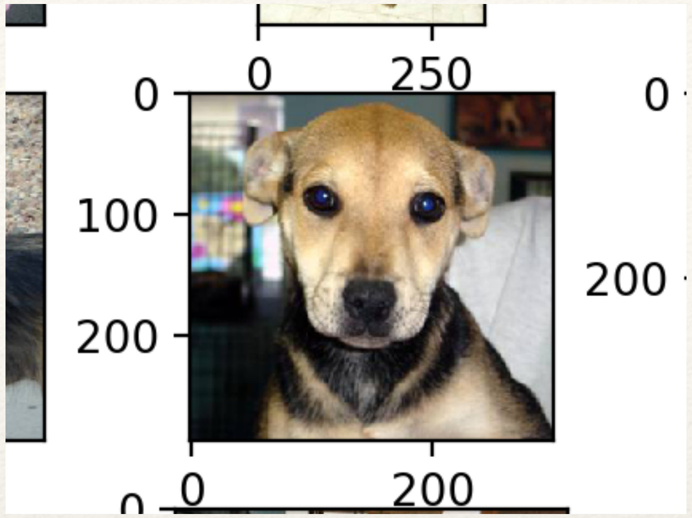


you can distinguish these cats and dogs, right? but how?
would you be able to write a code which classifies them with ~ 100% accuracy? well, a NN can learn to do this!

Convolutional Neural Networks
CNNs

Complex features are often *local*



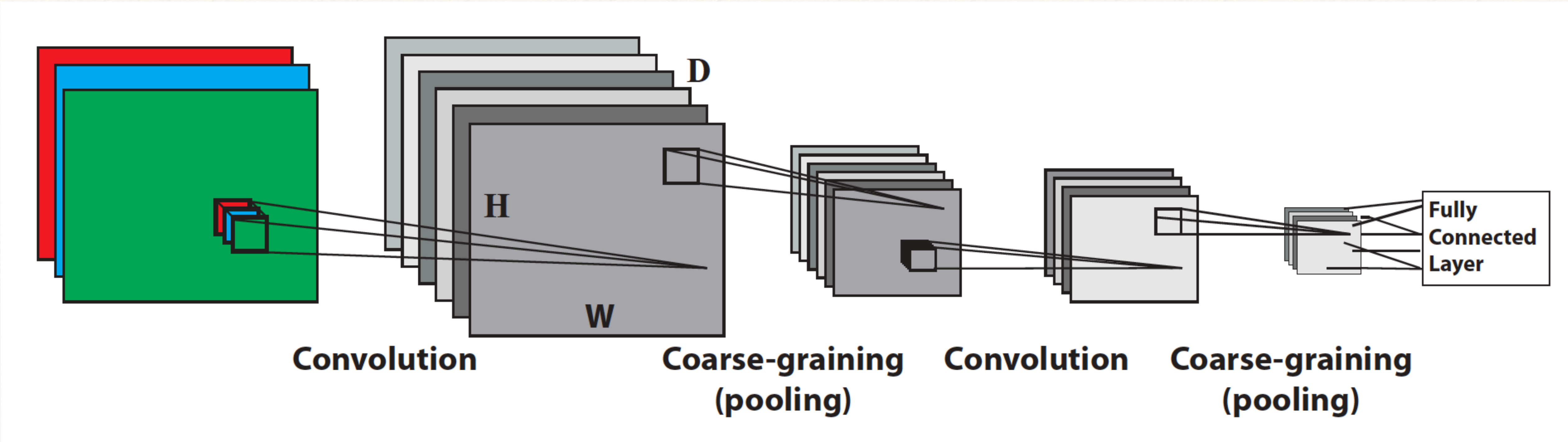


Apart from shape and color,
we know a cat is a cat because there are relations among their features, e.g. the position of the eyes/ ears respect to the head centre, independently of where in the image the cat is

Locality and **translational invariance** must end up playing a role in the identification task

Convolutional Neural Network (CNN)
a type of NN architecture designed to exploit these two characteristics

CNNs

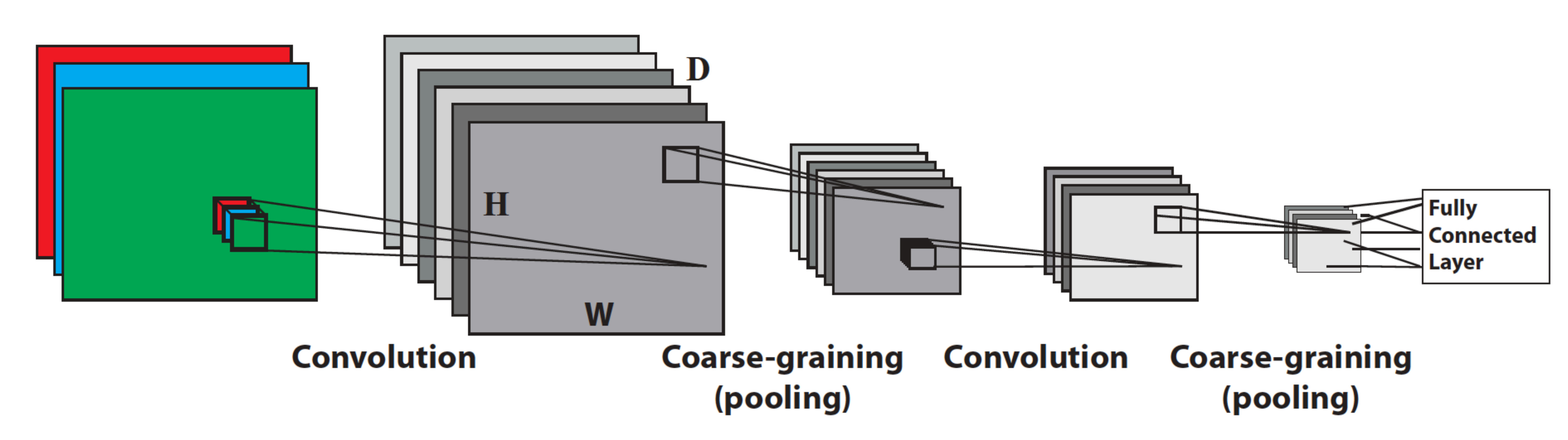


Two types of basic layers

Convolution layer: Height, Width and Depth (e.g. RGB channels) *Convolution*= operation to reduce information while maintaining spatial relations (locality and translation properties)

Pooling: Take areas of the image and reduce them. Example, max-pooling would take 2X2 neurons and replace by a single neuron with input the max of the 4

CNNs



Why do we do this?

Too much superfluous information in an image

Need to transform the image and capture the essentials while maintaining spatial relations

As we advance in the layers, the CNN is transforming the original image into something more and more abstract

In physics, translationally invariant systems can be parametrised by wave number and functional form (sin, cos) whereas an arbitrary system would be *much more complex*

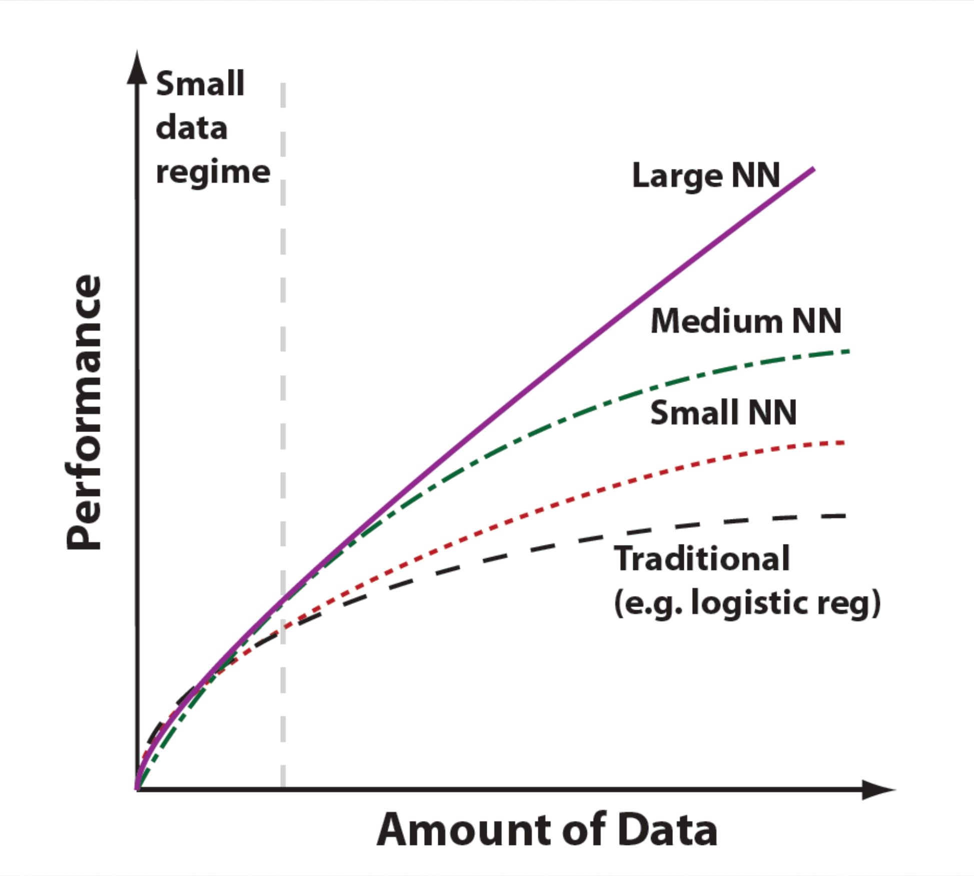
Why are NNs so good at learning?

Good at learning: ability to learn with little *domain knowledge*

That's something physicists (as humans) are good at

(Physics -> other things)

DNNs are good at this too, they are able to take large streams of data and learn features with little guidance, work like *black boxes*



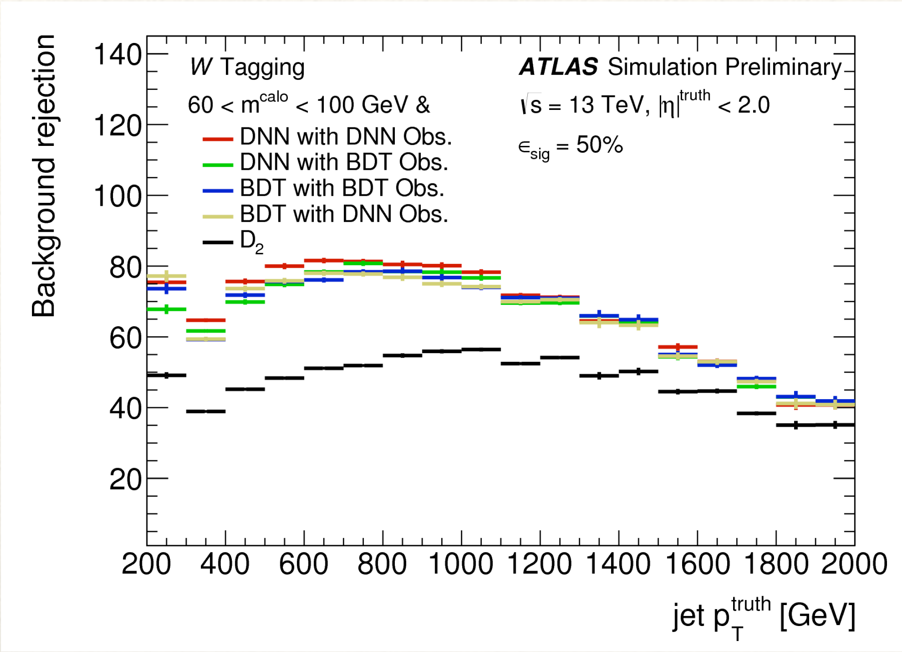
Good at handling large amounts of data: needle in a haystack

The NN structure (layers, 0/1 gates) allows a high representation power with moderate computational demands, e.g. allows parallelisation, use of GPUs…

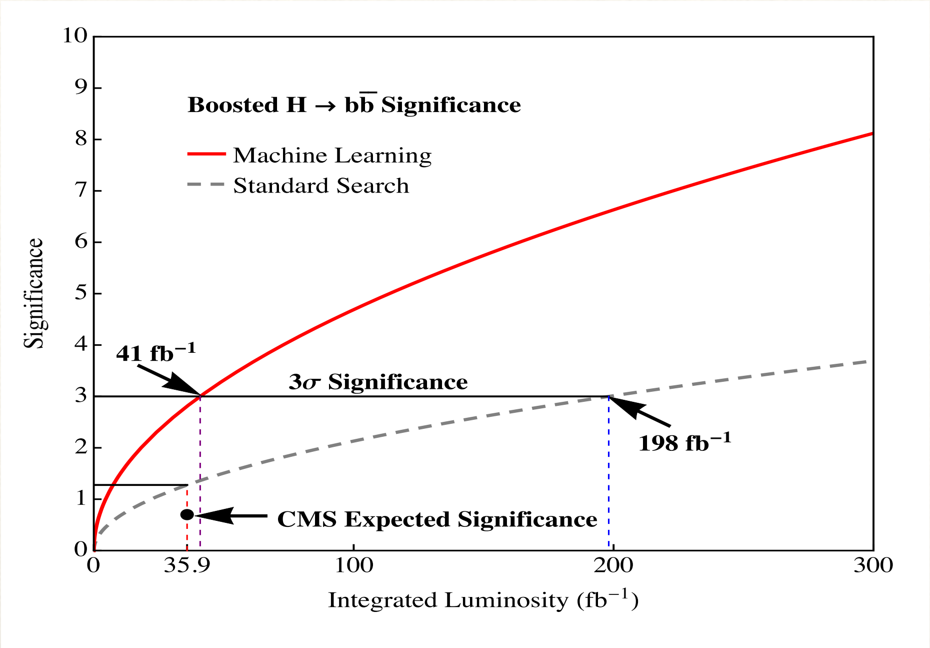
It scales better than other learning methods (like SVMs)

A lot of ML in Particle Physics is answering YES/NO questions

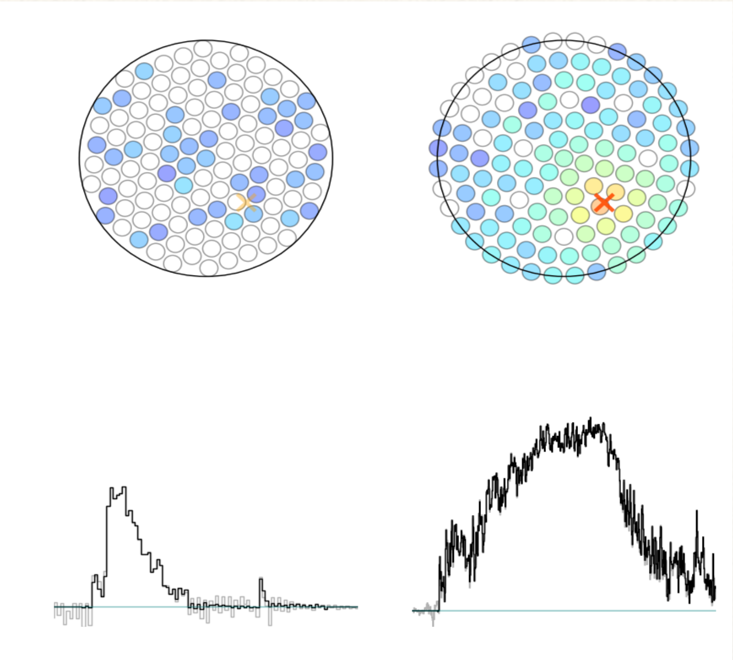
Is it a W?



Is it a Higgs?



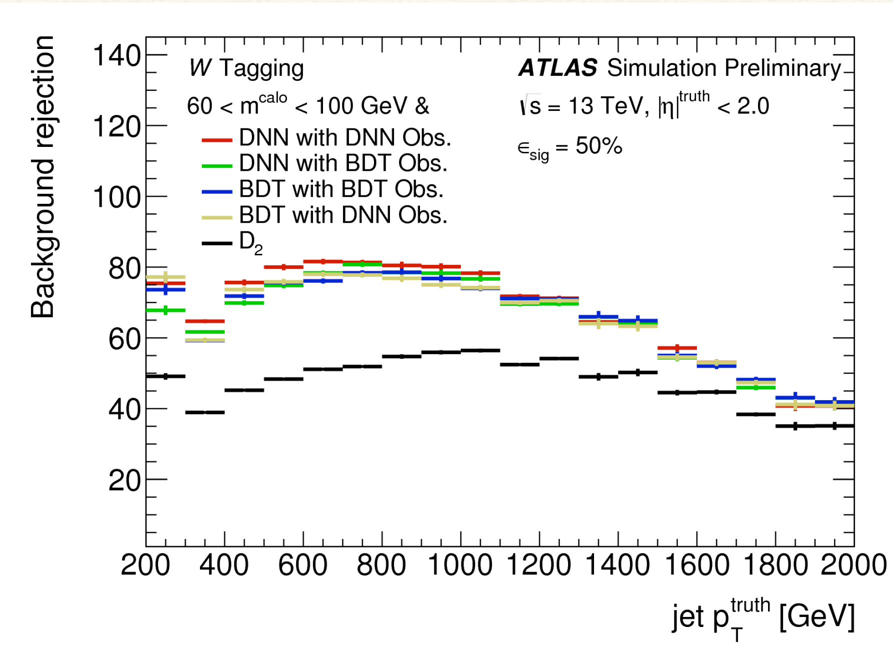
Is it DM?



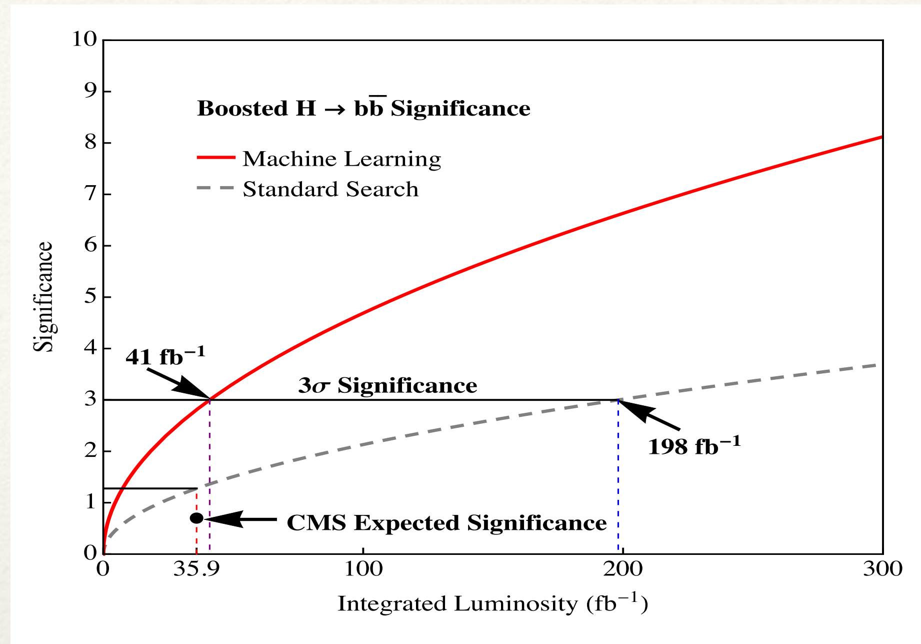
mostly using Neural Networks to deal with images (CNNs)

A lot of ML in Particle Physics is answering YES/NO questions

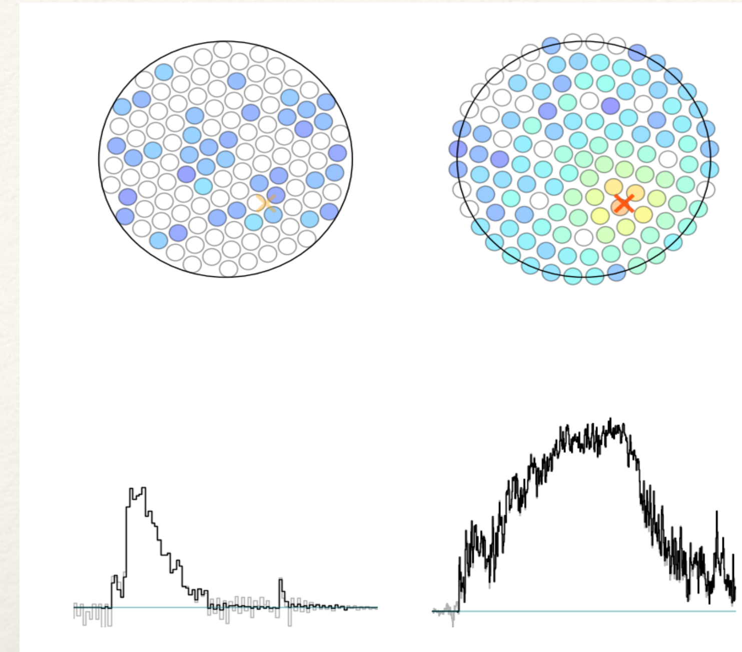
Is it a W?



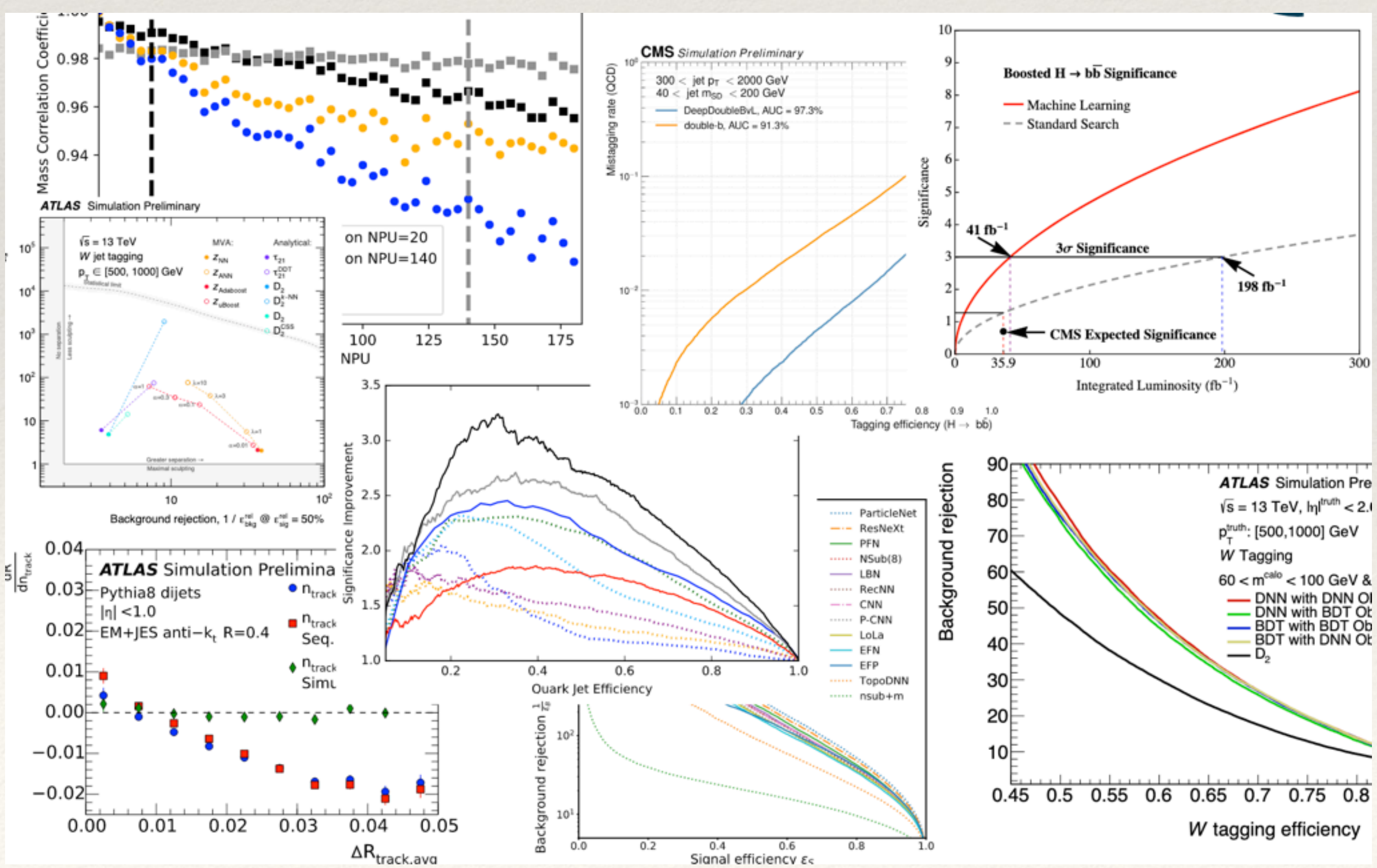
Is it a Higgs?



Is it DM?



mostly using Neural Networks to deal with images (CNNs)



The gains in ID-ing phenomena are typically in the range of 5%-30%

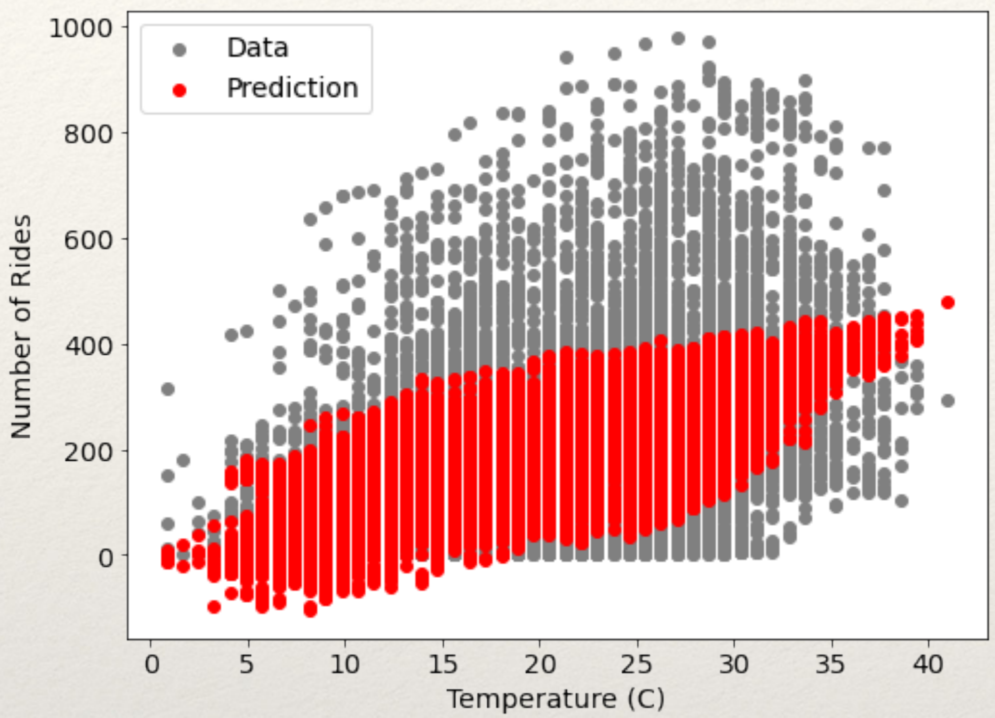
for tricky environments: difference between discovery or not

Diving into the unknown

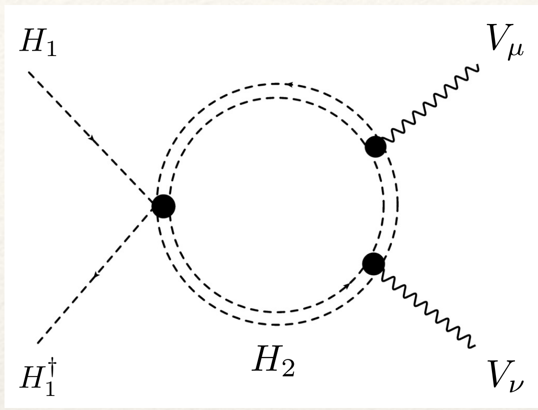
what if we did not know what we were looking for?

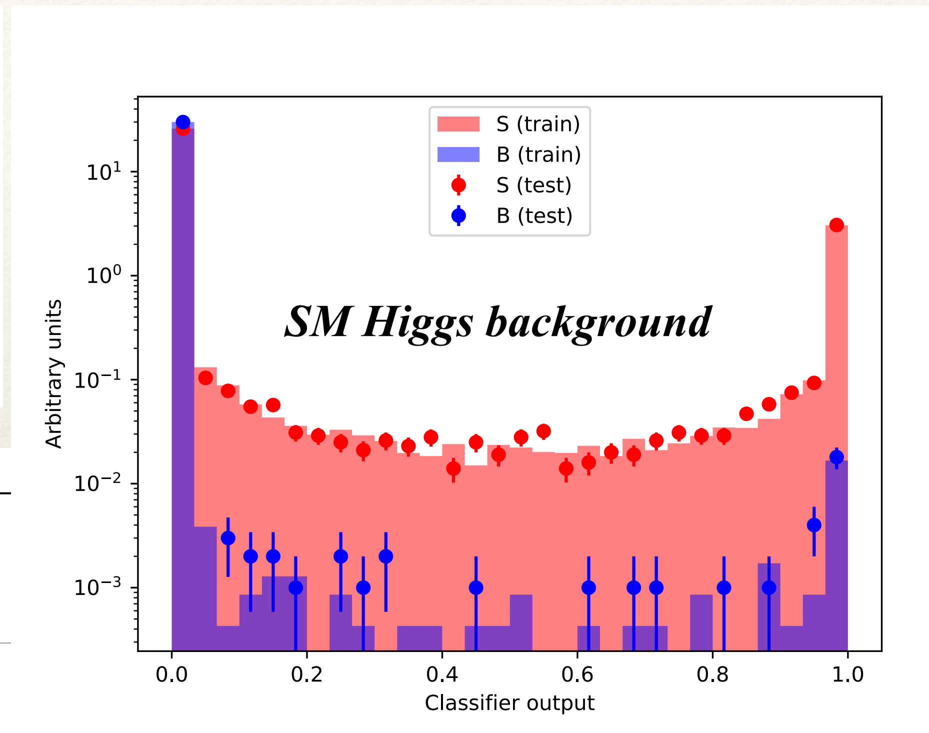
So far getting better at predicting outputs

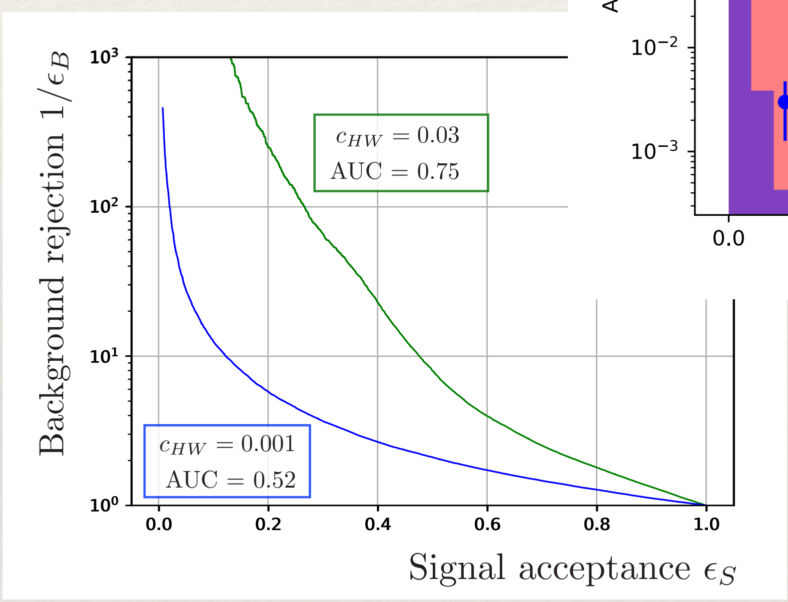
Multivariate linear regression



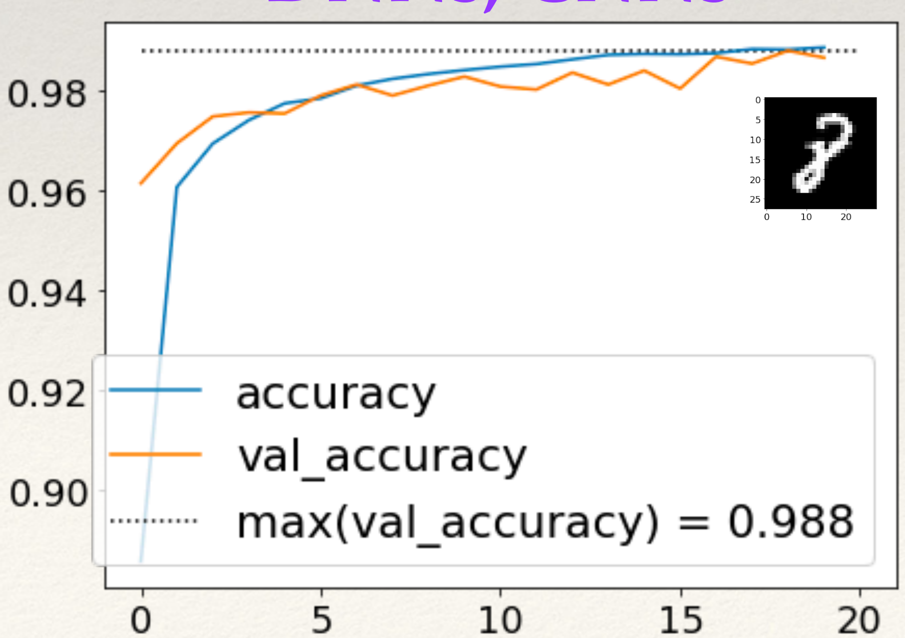
Binary classification in particle physics







Classification task with images DNNs, CNNs



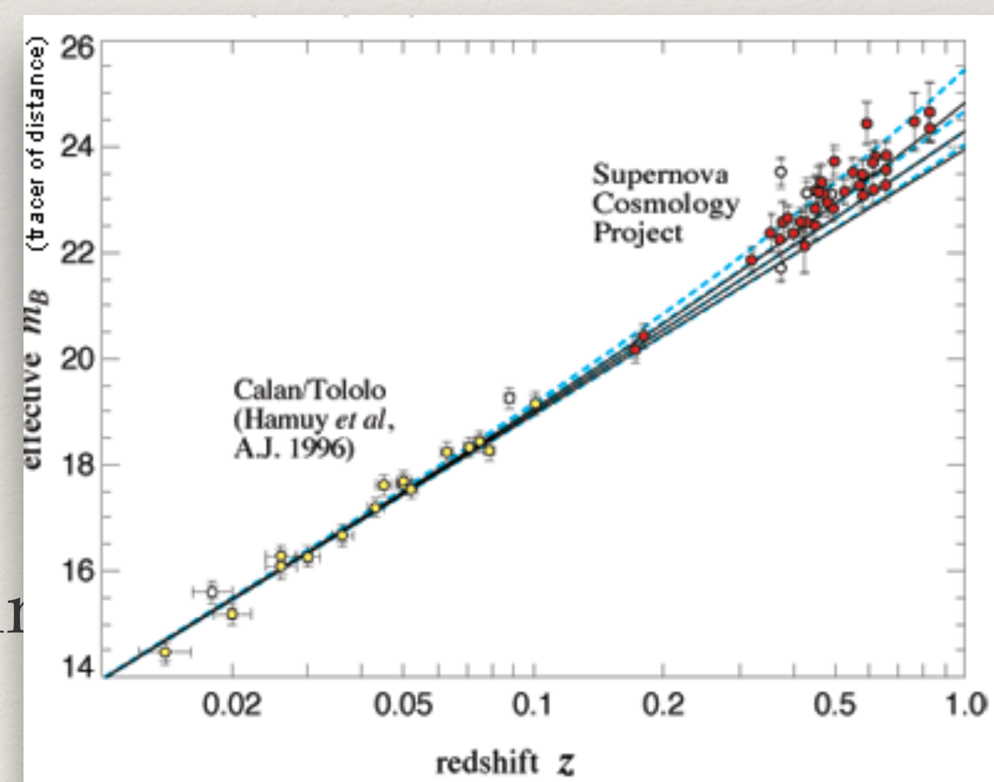
in all these cases we knew the labels,
the output of each input
we knew what we were looking for
our learning was *guided, supervised*

What if we didn't know what we were looking for?

what if the labels were not there?
because they are unknown or too costly to be obtained
or you wanted to learn something beyond these labels?

8.65126013755798E-01	-1.10967397689819E+00	1.37884104251862E+00	1.2258266210556E+00	-1.59931445121765E+00	-3.35545927286148E-01	4.52621191740036E-01	1.396528840065E+00	1.43482126295567E-02	7.61019229888916E-01	8.73991847038269E-01	6.46077632904053E-01	6.55984878540039E-01	2.75867860764265E-02
6.79041624069214E-01	-1.16113260388374E-01	-1.44924676418304E+00	7.25930333137512E-01	8.32636177539825E-01	-1.06863963603973E+00	7.34275758266449E-01	6.16540551185608E-01	1.10223841667175E+00	-3.8318920135498E-01	6.74581229686737E-01	9.89126741886139E-01	1.30116701126099E+00	2.11356925964355E+00
6.96710646152496E-01	1.53248345851898E+00	4.51736569404602E-01	9.53612387180328E-01	1.08532404899597E+00	-1.26148509979248E+00	2.19569746404886E-02	7.69306182861328E-01	1.76076497882605E-02	3.32875728607178E-01	6.81880474090576E-01	1.27228811383247E-01	1.65573984384537E-01	4.10077683627605E-02
1.00609767436981E+00	-2.18872010707855E-01	1.27333605289459E+00	1.62148654460907E+00	7.60176777839661E-02	5.86132228374481E-01	5.61284124851227E-01	1.70752331614494E-01	5.829758644104E-01	4.111288189888E-01	1.04620361328125E+00	7.77394592761993E-01	6.59388244152069E-01	1.17781555652618E+00
8.37866902351379E-01	1.13452970981598E+00	-9.11251723766327E-01	6.16625964641571E-01	1.00015830993652E+00	1.2927371263504E+00	1.27245831489563E+00	-6.76616489887238E-01	7.64919698238373E-01	1.16551697254181E+00	6.31192624568939E-01	7.86714017391205E-01	1.10604000091553E+00	8.15494537353516E-01
1.64174771308899E+00	-1.82980680465698E+00	1.50152146816254E-01	2.70264196395874E+00	-1.2972204387188E-01	-1.38081395626068E+00	1.94475173950195E-01	-1.47864866256714E+00	5.51068745553493E-02	3.0361208319664E-01	2.63883185386658E+00	6.36116921901703E-01	2.13914230465889E-01	2.300715893507E-01
1.46664869785309E+00	1.15517094731331E-01	-1.03616142272949E+00	8.77246916294098E-01	6.90861403942108E-01	1.13006901741028E+00	8.9072197675705E-01	4.26515430212021E-01	1.28321182727814E+00	-5.63879549503326E-01	1.0900456905365E+00	1.32757031917572E+00	1.08076226711273E+00	1.58807563781738E+00
1.64159500598907E+00	-1.04459571838379E+00	5.43058097362518E-01	7.44841694831848E-01	-3.0812594294548E-01	-9.56824660301208E-01	1.24903869628906E+00	-1.47708666324615E+00	7.31511473655701E-01	-8.47167015075684E-01	1.16093373298645E+00	1.60183656215668E+00	1.2244086265564E+00	1.25674676895142E+00
1.36144161224365E+00	-5.06246566772461E-01	-6.46163880825043E-01	1.04786920547485E+00	1.13672995567322E+00	6.98664963245392E-01	1.65792429447174E+00	-6.15012310445309E-02	2.15440988540649E+00	1.67285251617432E+00	1.52531015872955E+00	1.11749887466431E+00	6.50135576725006E-01	0E+00
1.29748213291168E+00	-2.54911869764328E-01	1.11683440208435E+00	9.79169011116028E-01	-1.35346204042435E-01	-6.9715404510498E-01	1.02749526500702E+00	-1.66322216391563E-01	1.2542005777359E+00	-4.79648977518082E-01	9.85065758228302E-01	1.2597473859787E+00	1.13483691215515E+00	7.60779619216919E-01
3.7850496172905E-01	-1.08213400840759E+00	1.00809907913208E+00	6.11425042152405E-01	-1.28556573390961E+00	-6.55805096030235E-02	9.16272640228271E-01	-8.86185646057129E-01	1.371866106987E+00	-4.21904683113098E-01	3.91696661710739E-01	7.90428757667542E-01	1.79072606563568E+00	1.38843953609467E+00
6.75932705402374E-01	1.26536428928375E-01	9.56136286258698E-01	6.86048328876495E-01	-1.99499741196632E-01	-1.70811021327972E+00	9.24258589744568E-01	-6.93052709102631E-01	1.38742446899414E+00	-5.49809157848358E-01	5.77164828777313E-01	1.04205667972565E+00	1.60216736793518E+00	1.92126834392548E+00
4.63986992835999E-01	1.13328350707889E-02	3.69053810834885E-01	5.53186655044556E-01	3.89279514551163E-01	3.0157333612442E-01	1.65791296958923E+00	1.52030777931213E+00	2.4887318611145E+00	-5.59999048709869E-01	4.2525789141655E-01	1.12646055221558E+00	2.35060715675354E+00	2.46035838127136E+00
1.16491532325745E+00	-1.21687364578247E+00	1.38738358020782E+00	1.28936243057251E+00	8.53503286838531E-01	-8.64189207553864E-01	5.52336990833282E-01	3.71582925319672E-01	8.29125940799713E-01	-1.4921934902668E-01	1.86399924755096E+00	9.7541731595993E-01	4.64367836713791E-01	1.68607497215271E+00
8.00468981266022E-01	-9.84456762671471E-02	-7.95078039169312E-01	4.77954834699631E-01	-4.29955363273621E-01	1.08130276203156E+00	8.88447284698486E-01	-3.40310782194138E-01	9.66846108436584E-01	8.3782559633255E-01	5.76379001140594E-01	6.60021543502808E-01	1.01617026329041E+00	6.41459822654724E-01
3.41047215461731E+00	5.90151190757751E-01	1.50781488418579E+00	2.19785332679749E+00	9.15379881858826E-01	-2.43708327412605E-01	1.00378489494324E+00	9.47994440793991E-02	8.52464973926544E-01	-5.42668044567108E-01	2.5086042881012E+00	2.03399634361267E+00	7.19505548477173E-01	4.56694543361664E-01

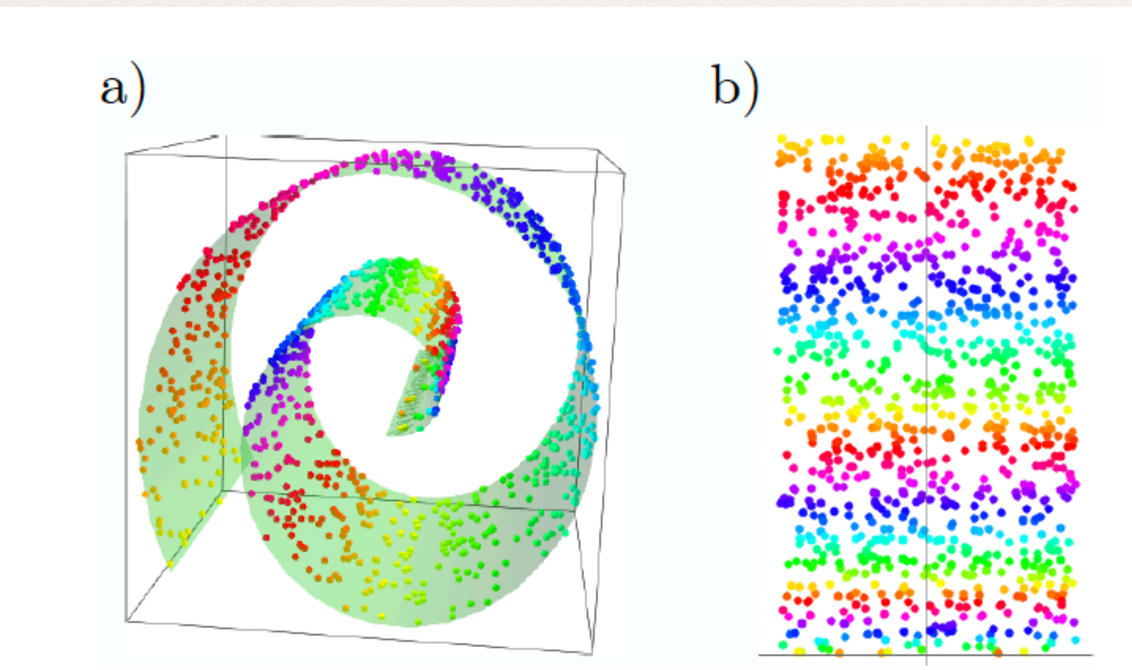
what would you do?
as a physicist, you would start thinking on
possible physical relations, plotting things,
trying to obtain the **best data representation**
the representation which **manifests** a behaviou



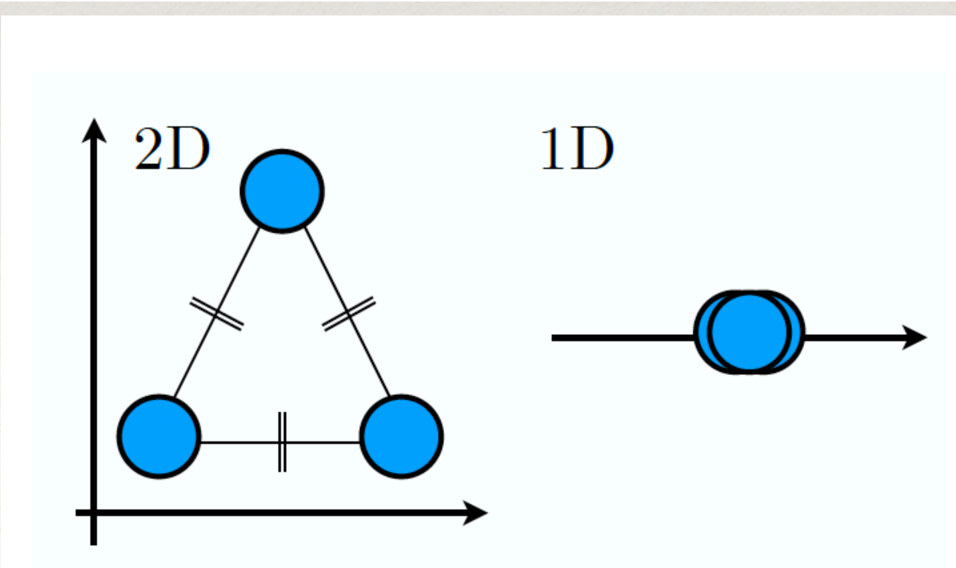
So everything starts with data visualization

But we can’t visualise things in more than 3D
when most data we want to mine is high-dimensional…

So you need to do DIMENSIONAL REDUCTION
from original space to **latent space**



Reduction n-D to few-D isn’t simply
projecting in a lower dimensional space
one dimension at a time
Choice: *direction* to project
to keep as much info as possible



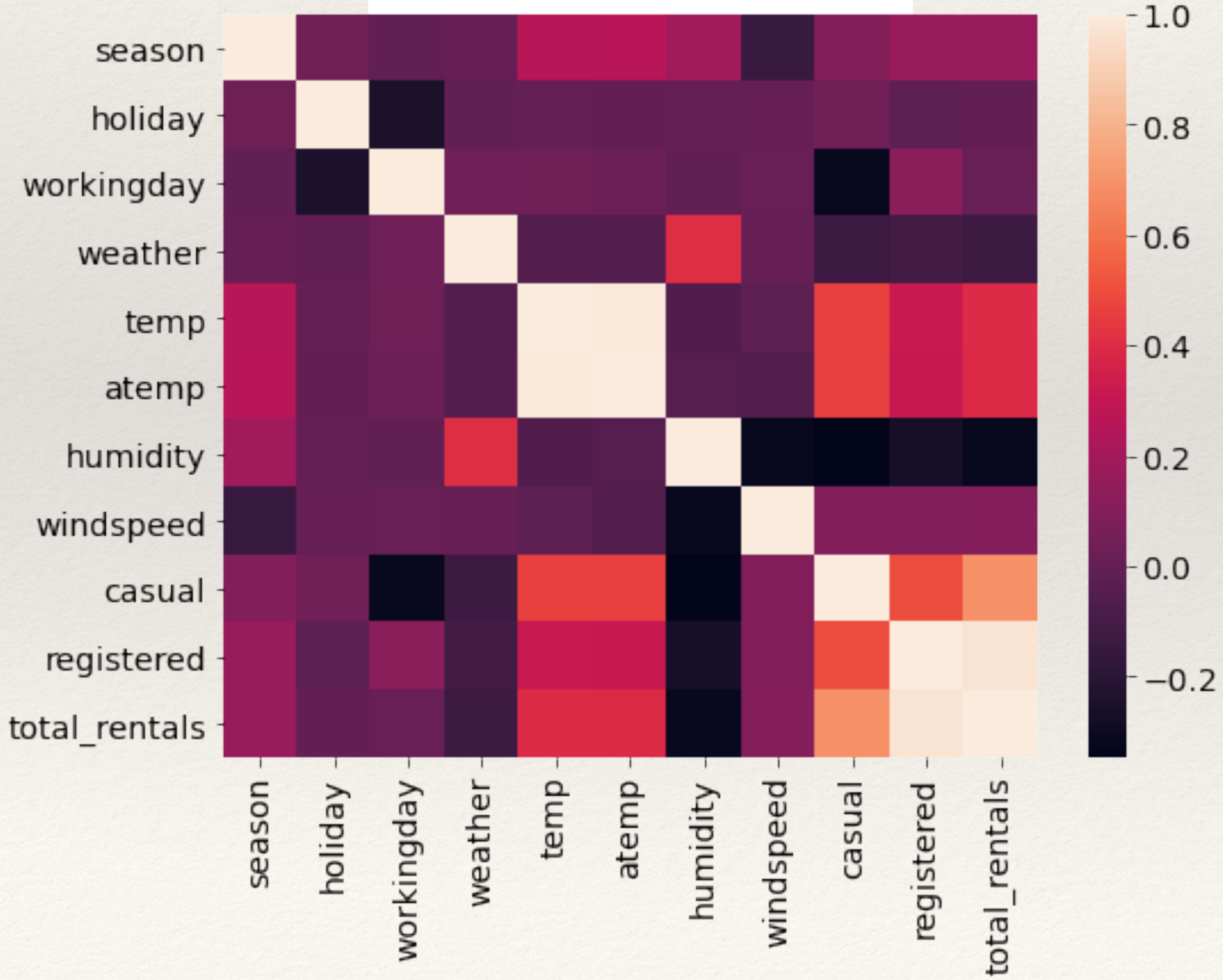
Bad choice
end up with a *crowding problem*
the best choice to represent this data is 2D,
going to 1D does limit your ability to learn
There’s hope, stat dynamics shows
micro->macro can work

Being smart at dimensional reduction

The direction to project out dimensions is important
We need a *criteria*

Principal Component Analysis (PCA)

$$\mathbf{\Sigma}(\boldsymbol{X}) = \frac{1}{N-1}\boldsymbol{X}^T\boldsymbol{X}.$$



In our representation of the data there are clearly some *redundancies* there could be one or more LINEAR COMBINATIONS of some of these variables which capture most of the information

PCA

The procedure is simple enough, take the correlation matrix

$$\boldsymbol{\Sigma}(\boldsymbol{X}) = \frac{1}{N-1}\boldsymbol{X}^T\boldsymbol{X}.$$

and diagonalize it

$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T$ with $\boldsymbol{S}$ diagonal

$$\begin{aligned}\boldsymbol{\Sigma}(\boldsymbol{X}) &= \frac{1}{N-1}\boldsymbol{V}\boldsymbol{S}\boldsymbol{U}^T\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T \\ &= \boldsymbol{V}\left(\frac{\boldsymbol{S}^2}{N-1}\right)\boldsymbol{V}^T \\ &\equiv \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T.\end{aligned}$$

$\boldsymbol{\Lambda}$

is a diagonal matrix with ordered eigenvalues

$\boldsymbol{V}$

contains the eigenvectors the directions of decreasing eigenvalue

We can then dimensionally reduce, but removing the directions in $\boldsymbol{V}$ with the smallest eigenvalues, the ones which carry less information in correlations

eqs. from this excellent review

t-SNE

PCA is good as a first try at visualisation but is limited by its linearity
Often we would like to preserve **local** structures in higher-dimensions, and PCA won't do that
A good example of non-linear techniques is
t-SNE (t-stochastic neighbour embedding)

In a nutshell,
t-SNE compares local distributions in the original and latent space

ORIGINAL

$$p_{i|j} = \frac{\exp(-||x_i - x_j||^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-||x_i - x_k||^2/2\sigma_i^2)}$$

LATENT

$$q_{ij} = \frac{(1+||y_i - y_j||^2)^{-1}}{\sum_{k\neq i}(1+||y_i - y_k||^2)^{-1}}$$

with sigma_i some parameter
and Y the projected latent space

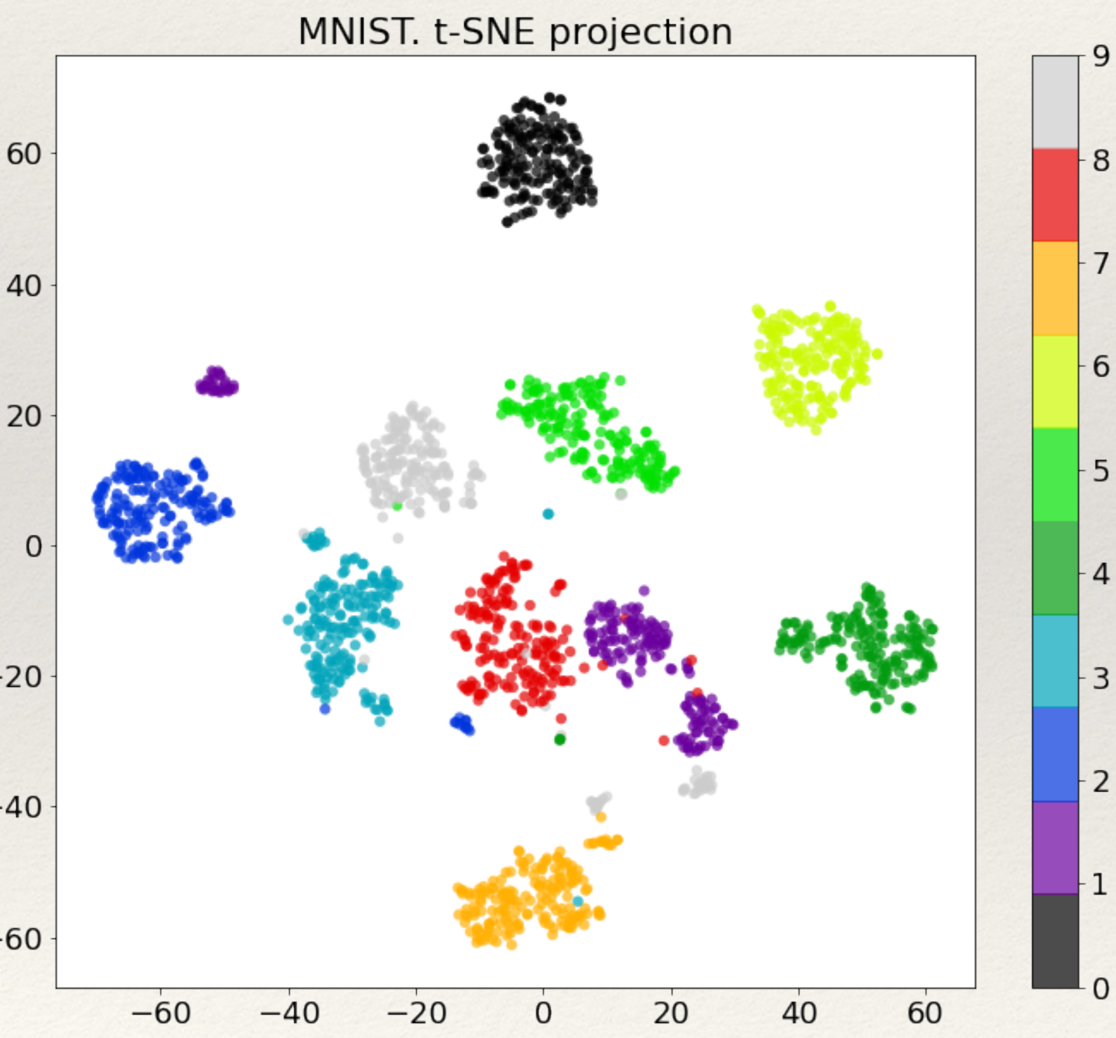
$$\tilde{\boldsymbol{Y}} = \boldsymbol{X}\tilde{\boldsymbol{V}}_{p'}$$

and provides a criteria for minimisation

the latent space choice which achieves the minimum is then chosen as latent space

Clustering

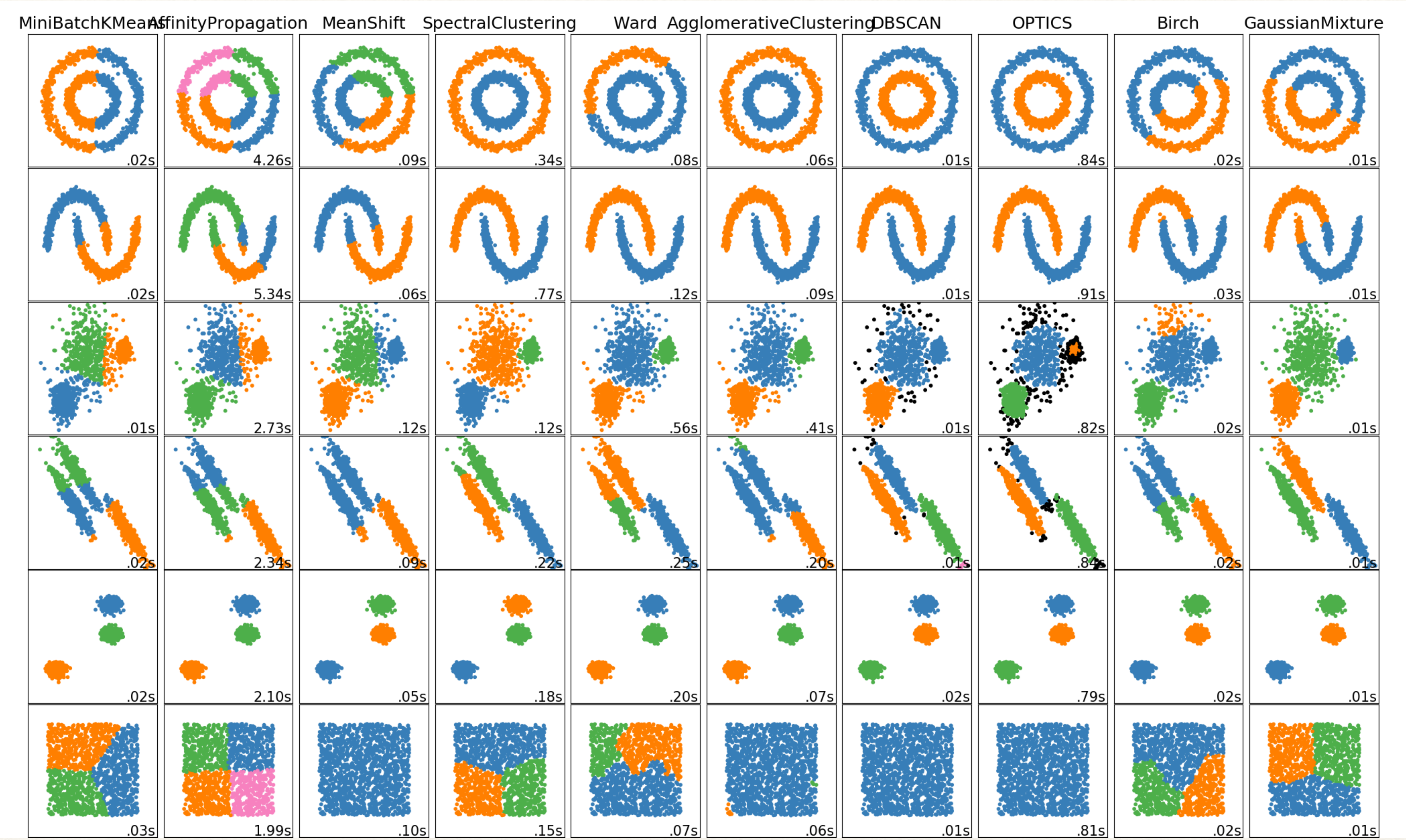
Now that we have reduced dimensionality
by PCA or t-SNE or another method
we can start thinking on finding patterns in it



Clustering is the most intuitive way to
find patterns
Finding clusters of common behaviour
using some distance criteria
in the latent space

Clustering is an iterative procedure
start with some parameters like N clusters,
cluster size etc
and try clustering the data using these criteria
The mathematical expressions are cumbersome
(many definitions of running parameters)
but the intuitive meaning is clear

Different clustering methods



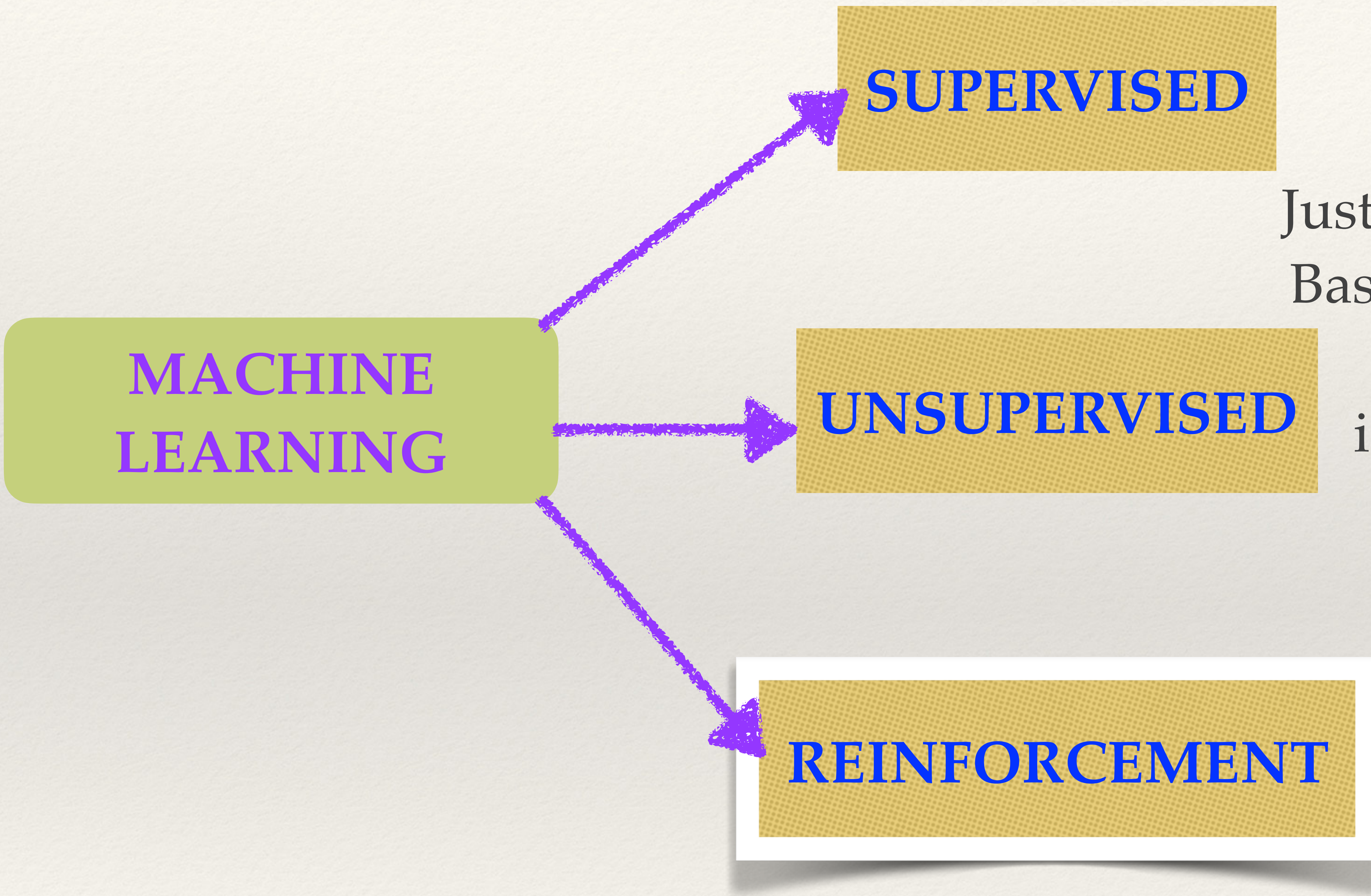
From SCIKIT webpage on clustering methods

Can I have a cookie?

Learning by reward

Additional

Types of learning



Just touched the surface
Basis to explore further
and incorporate
it in your research

Supervised to Reinforced Learning

Cool ways to accelerate learning, capture important aspects of the data, incorporate different types of data

Learn **from** humans to do what humans **already do**, but better and faster, and in more difficult situations

But, what if we wanted a machine to become **better** than a human at completing a high-level task?

* See these lectures

Let's find a DIFFICULT task

A truly human-difficult task
not just a task that a machine can do faster or with lower resolution

Supervised/unsupervised learning identifies *patterns* in data
But this isn't the same as learning to develop a ***strategy***
and to do it better than a human

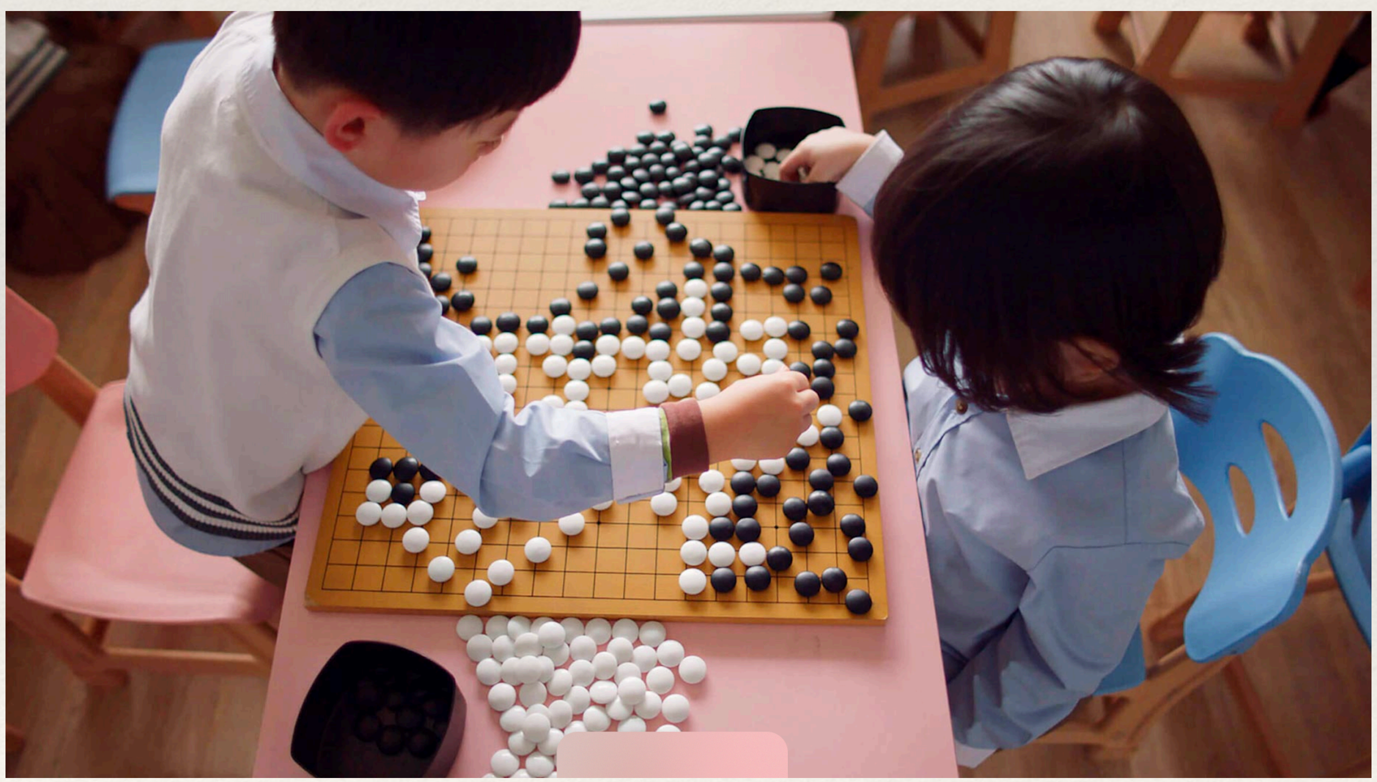
Chess is a high-level activity
different players develop different strategies
the goal is ***long-term***
important pieces can be sacrificed to achieve
checkmate some moves along the way
and you have an adversary which will oblige
you to ***reassess*** your strategy at each step
combinatorics is ginormous

Human vs Machine

February 1996

Deep Blue (IBM) beat Garry Kasparov (World Champion) and did it again many times after brute-force computing power analysing many hundreds of millions positions / second

Human vs Machine

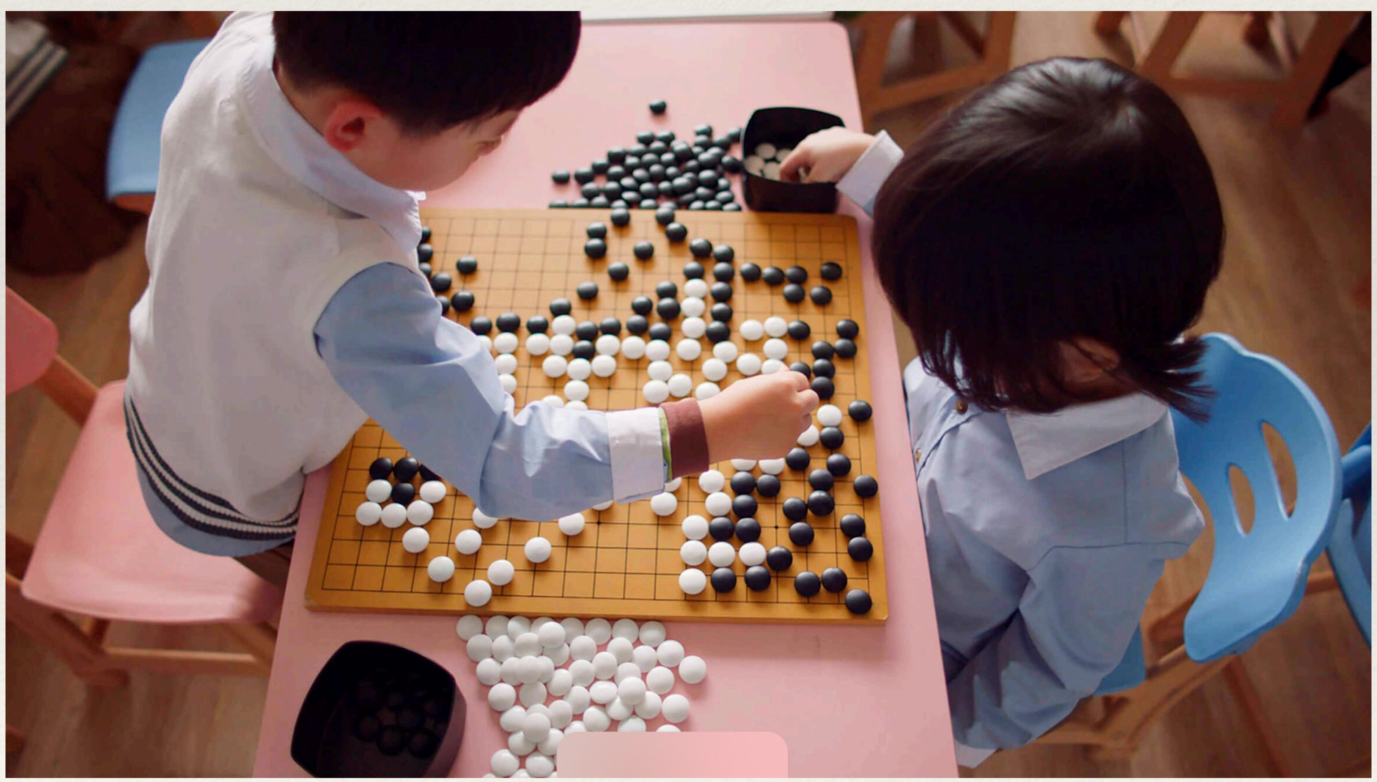


February 1996
Deep Blue (IBM) beat Garry Kasparov (World Champion) and did it again many times after brute-force computing power analysing many hundreds of millions positions / second

October 2015
AlphaGo Zero beats a professional Go player learned from playing against *itself*

November 2017
AlphaZero builds on DNNs to beat world champions in Go, chess and shogi

Human vs Machine



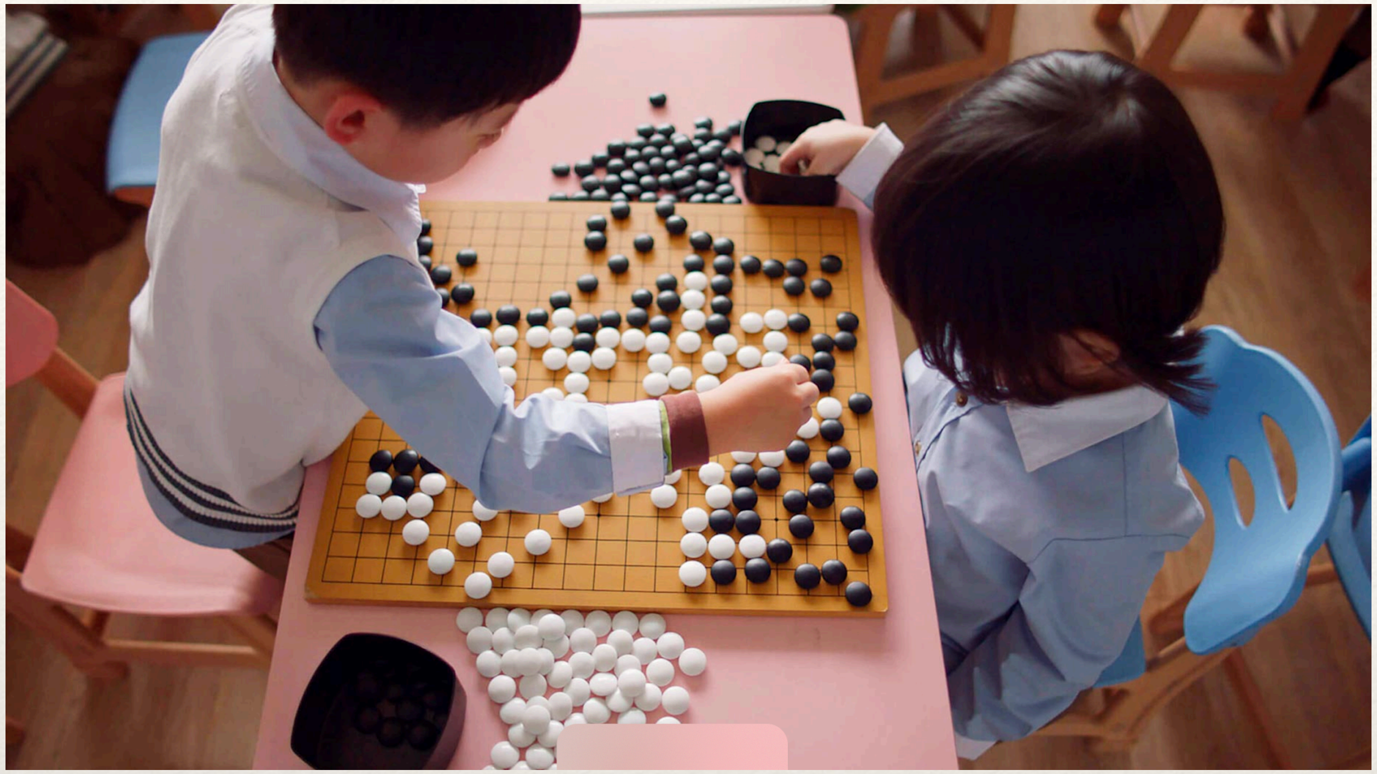
February 1996
Deep Blue (IBM) beat Garry Kasparov (World Champion) and did it again many times after brute-force computing power analysing many hundreds of millions positions / second

October 2015
AlphaGo Zero beats a professional Go player learned from playing against *itself*

November 2017
AlphaZero builds on DNNs to beat world champions in Go, chess and shogi

A new paradigm of learning: **REINFORCEMENT**

Go game

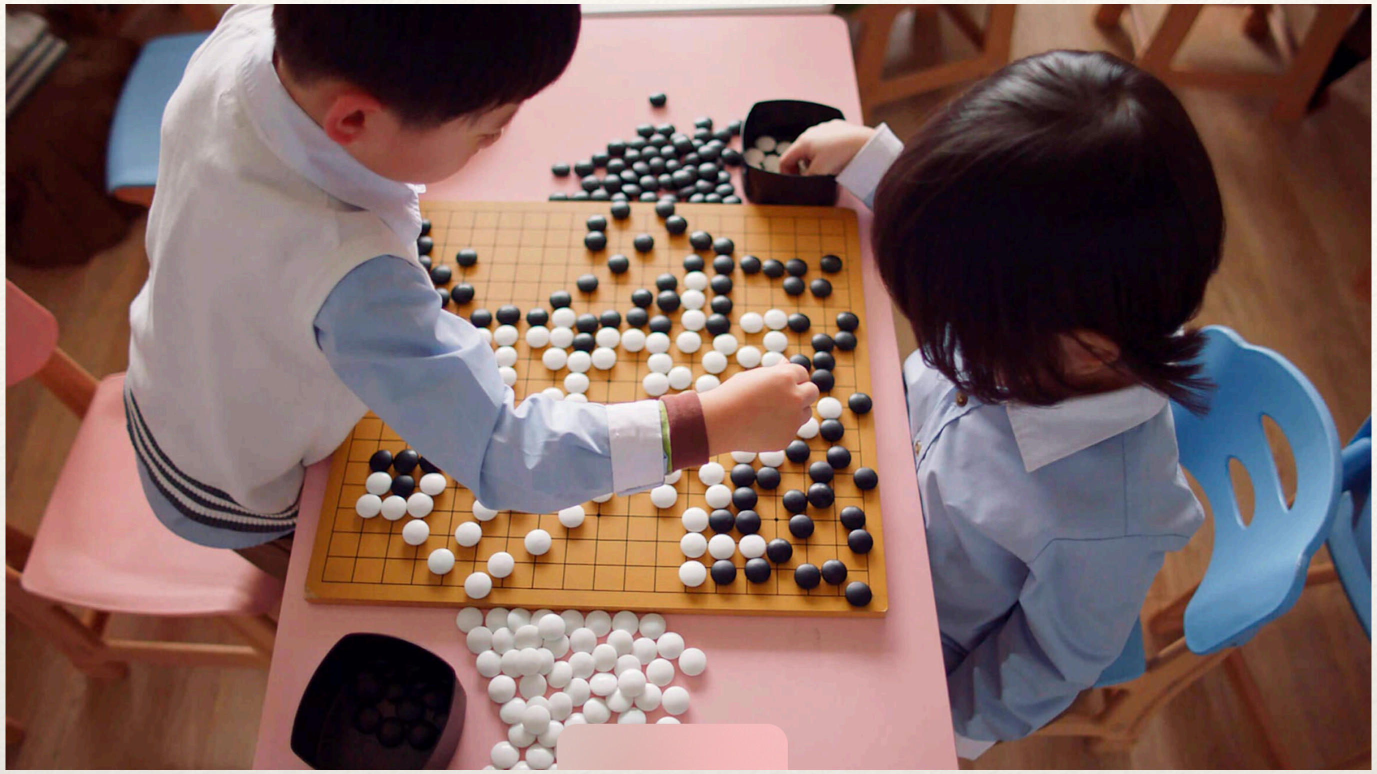


Simple game: moves are simple
no hierarchy like chess
king/queen/bishop/pawn…
goal: surround and capture
opponents' pieces

Simple rules, extreme levels of complexity when building strategies
no machine could beat a Go-master until 2015
Why is it so difficult?
how would you teach a machine to learn this game?

X,y

Go game



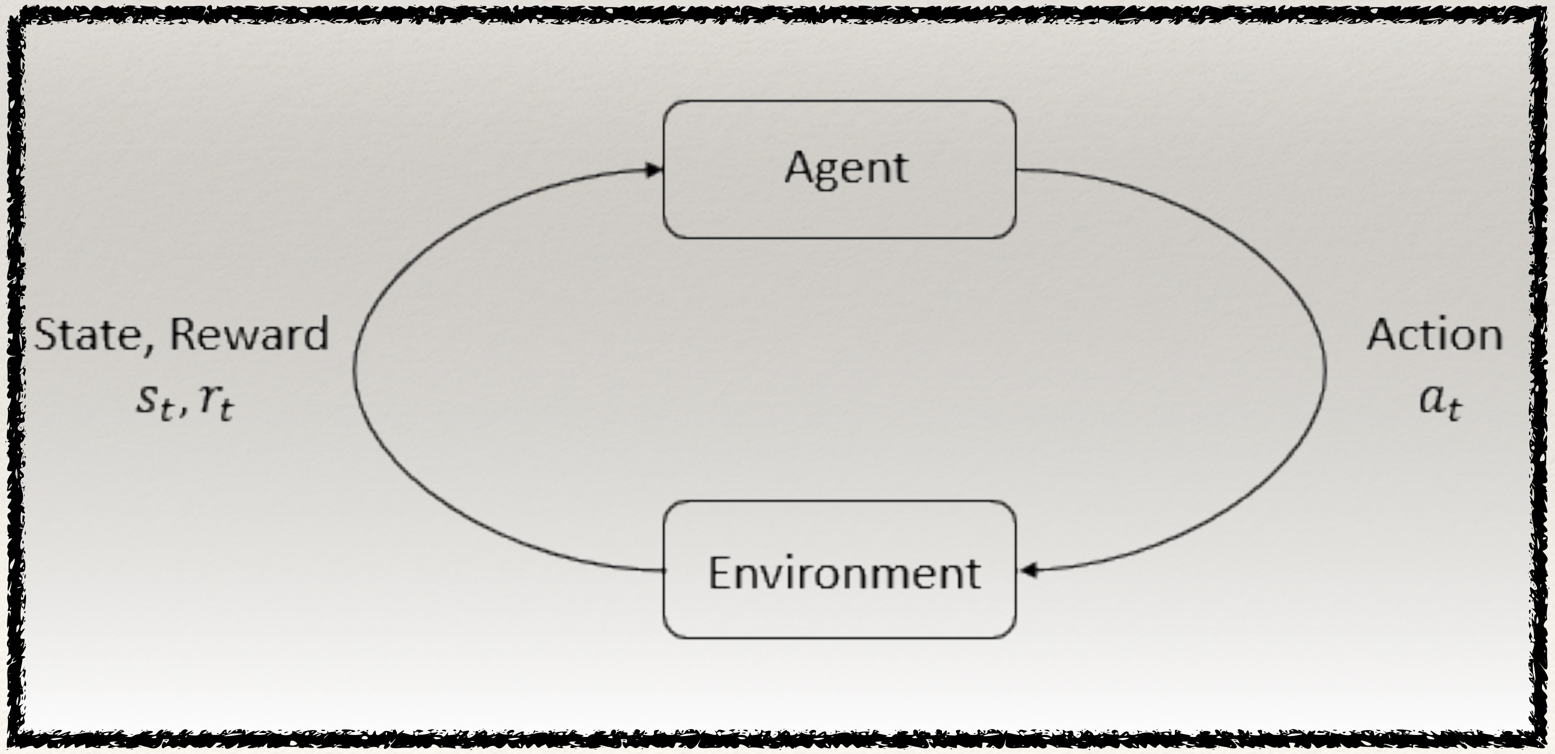
Simple game: moves are simple
no hierarchy like chess
king/queen/bishop/pawn…
goal: surround and capture
opponents' pieces

develop a strategy for long-term winning:
3^(19*19)~10^172 configurations at one step
decision in this one step guided by possible future gains
but opponent's actions change every subsequent move

Reinforcement learning

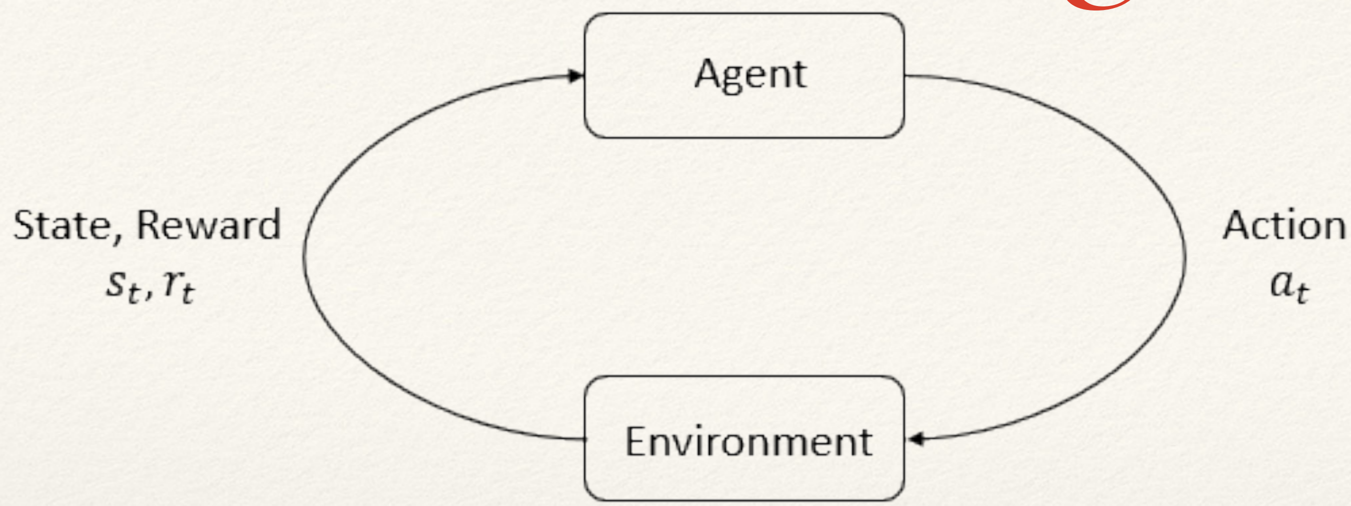
The task of getting better at Go was too difficult
too many possibilities, no human could teach from example
To beat humans we had to allow machines to learn in a different way

Machine needs to learn to make good sequences of decisions
dealing with delayed labels and developing a long-term strategy
Some form of iterative way of improving strategy
which can examine many steps ahead



agent interacts with
the **environment** in **state** *st*
takes **actions** based on **reward** *rt*
which tells about good current state is
GOAL: maximise total about of
rewards (**return**)
RL help the agent to achieve goal

Reinforcement learning: concepts



State/Observation: some kind of tensor (e.g. *an image*)

Action: possible transformation of the state (e.g. *move pawn*)

Policy: rule used by the agent to decide what action to take

$$a_t = \pi(s_t)$$

can be deterministic or stochastic and often parametrised

$$a_t = \pi_\theta(s_t)$$

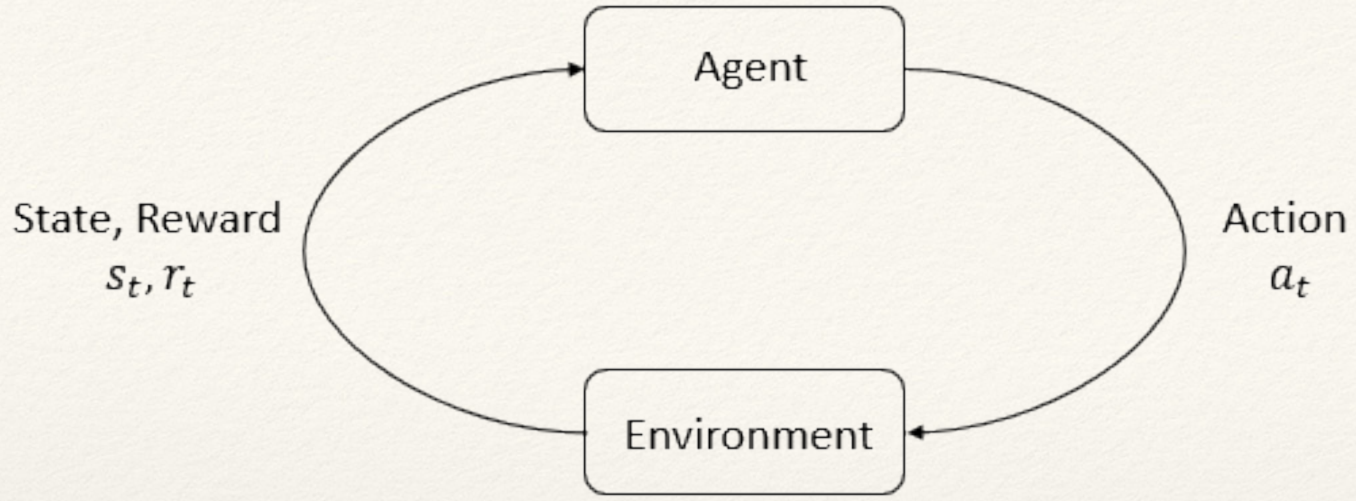
by some form of modelling of possible new situations

this sampling of possible trajectories

$$\tau(s_0,\, a_0,\, s_1,\, a_1 \ldots)$$

often done with DNNs (**Deep Reinforcement**)

Reinforcement learning: concepts



Reward: some function of current state and action taken, and next state

Return: total reward in a full sequence, a trajectory

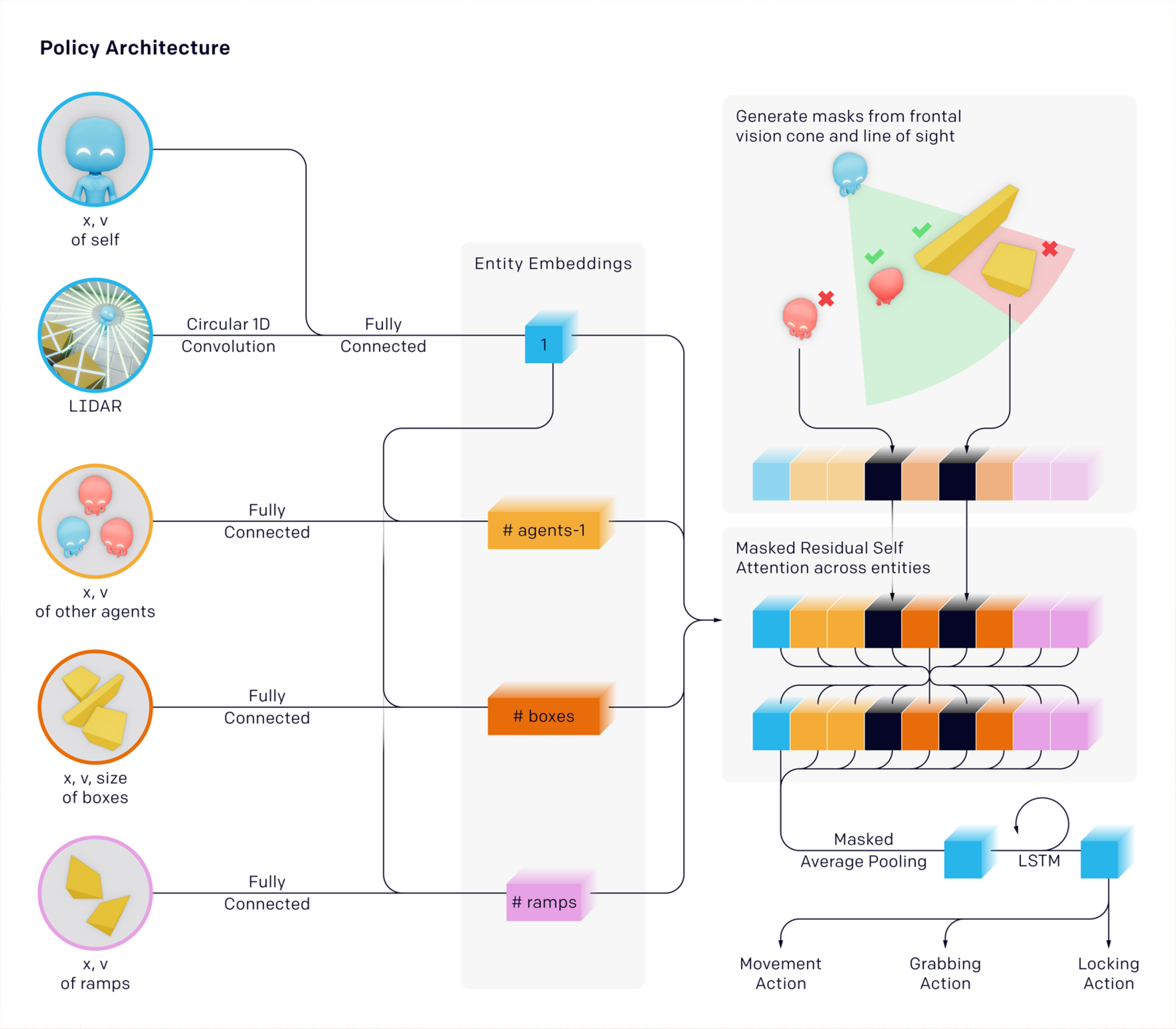
$$R(\tau) = \sum_{t=0}^{T} r_t$$

Real problems have limitations, so not all trajectories can be taken at no cost (e.g. *time limit, loss at each step…*) and we introduce a **discount**

$$R(\tau) = \sum_{t=0}^{T} \gamma^t r_t$$

cash now / cash in few years

Reinforcement learning: fun video



And a super fun blog and video

Reinforcement learning: overview

Clearly, this is a complex set-up

states could contain lots of information

an agent could choose among many actions

the number of possible steps could be very large

rewards are set to help the algorithm to increase final return

(**policy optimisation**)

but their efficiency depends on ability to explore

how the state changes by itself (opponent) or by the action

RL helps learning an *environment*

There are many, many possibilities

most successful are based on Deep Learning

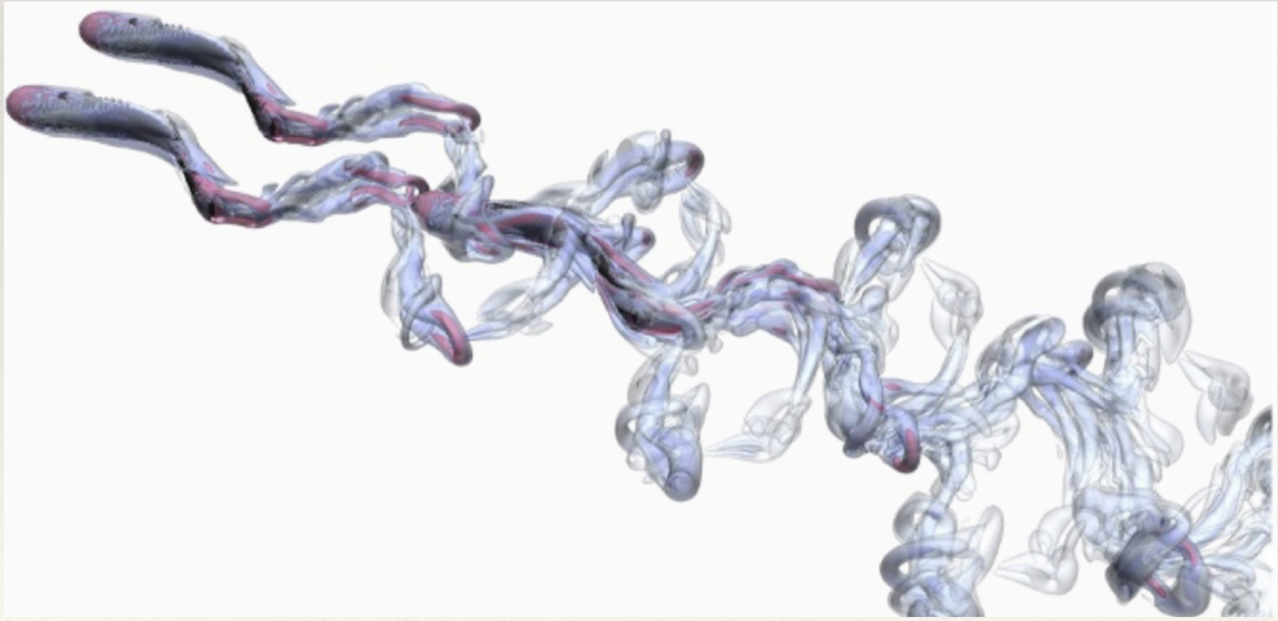
& good adaptation to environment changes

e.g. ability to dynamically drop and add terms in policy

Going further?

You could follow a *Tensorflow* tutorial
on training agents using different policies
Environment: Cartpole (start reading this post)

In Physics, mostly unexplored
complex numerical simulations: many body, fluid dynamics…



situations with many agents
and interactions
see e.g. this nice example fish
coordinated swimming for
energy saving

Transfer Learning

Additional

Transfer Learning

So far, our Machine was like a newborn baby
looking at a dataset/environment
and learning from it
with or without guidance, using rewards

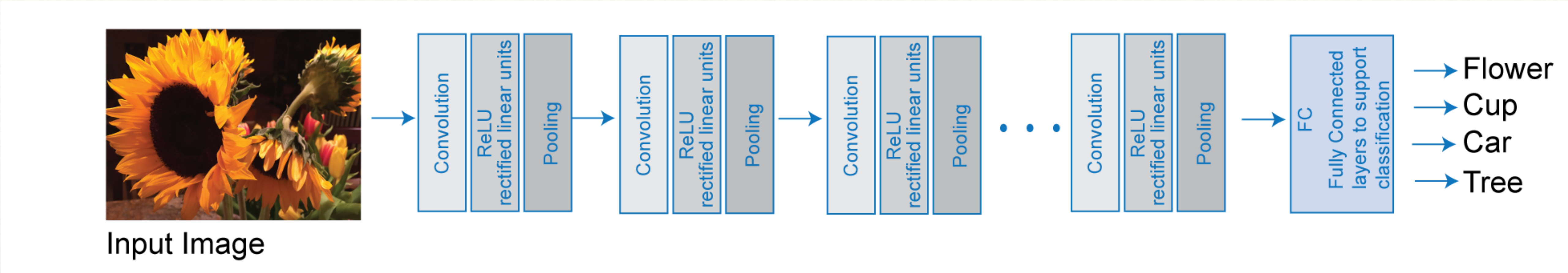
Complex datasets require Deep Learning
long time to run and optimise
and *specific* to the dataset

BUT babies do not learn every task from scratch

example: **language**
learn generic concepts
actions->verbs
objects/people->names
characteristics->adjectives

Babies are able to
TRANSFER LEARNING
ENGLISH -> SPANISH, CHINESE etc
and our Machine should do too

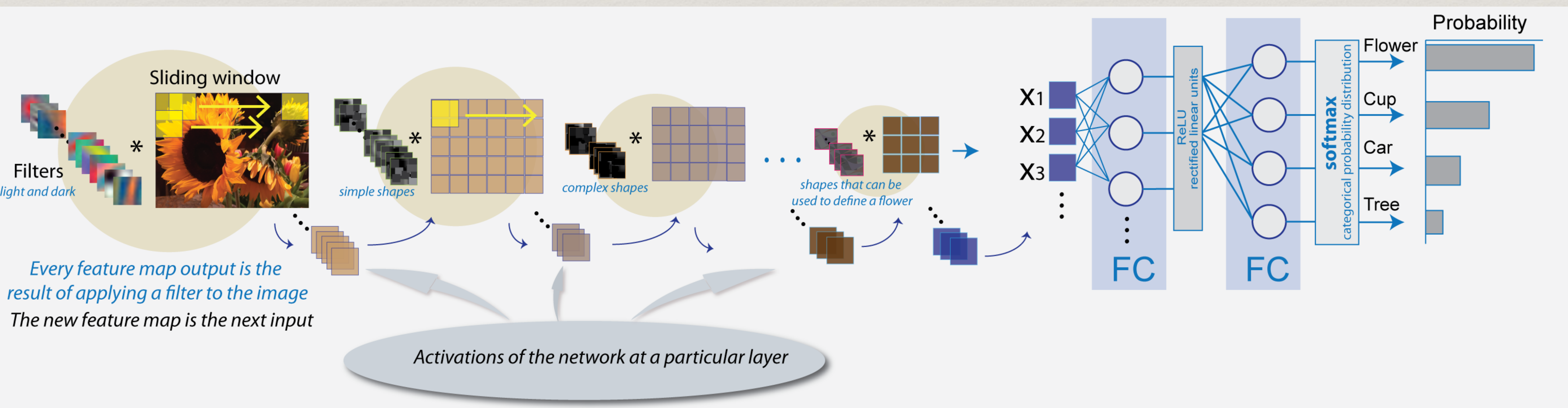
Supervised learning Image classification problem



Hidden layers are transforming the initial image into something more abstract (simple/complex shapes->shapes specific to a flower) and at the end this is transformed into a set of vectors which are then fed to a set of FC NNs

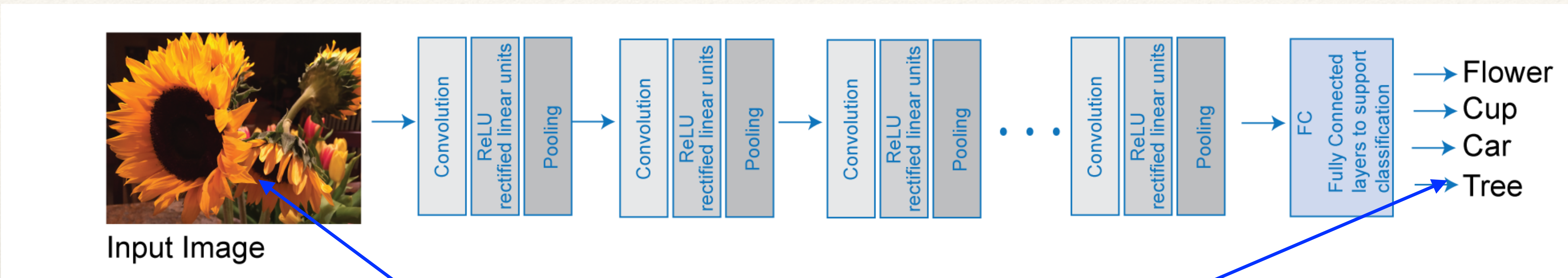
initial layers are more problem-independent

later layers are more specific to the problem at hand



Changing the target

What if now I want to classify types of dogs or leaves or cars?
I would start by changing…



Dataset

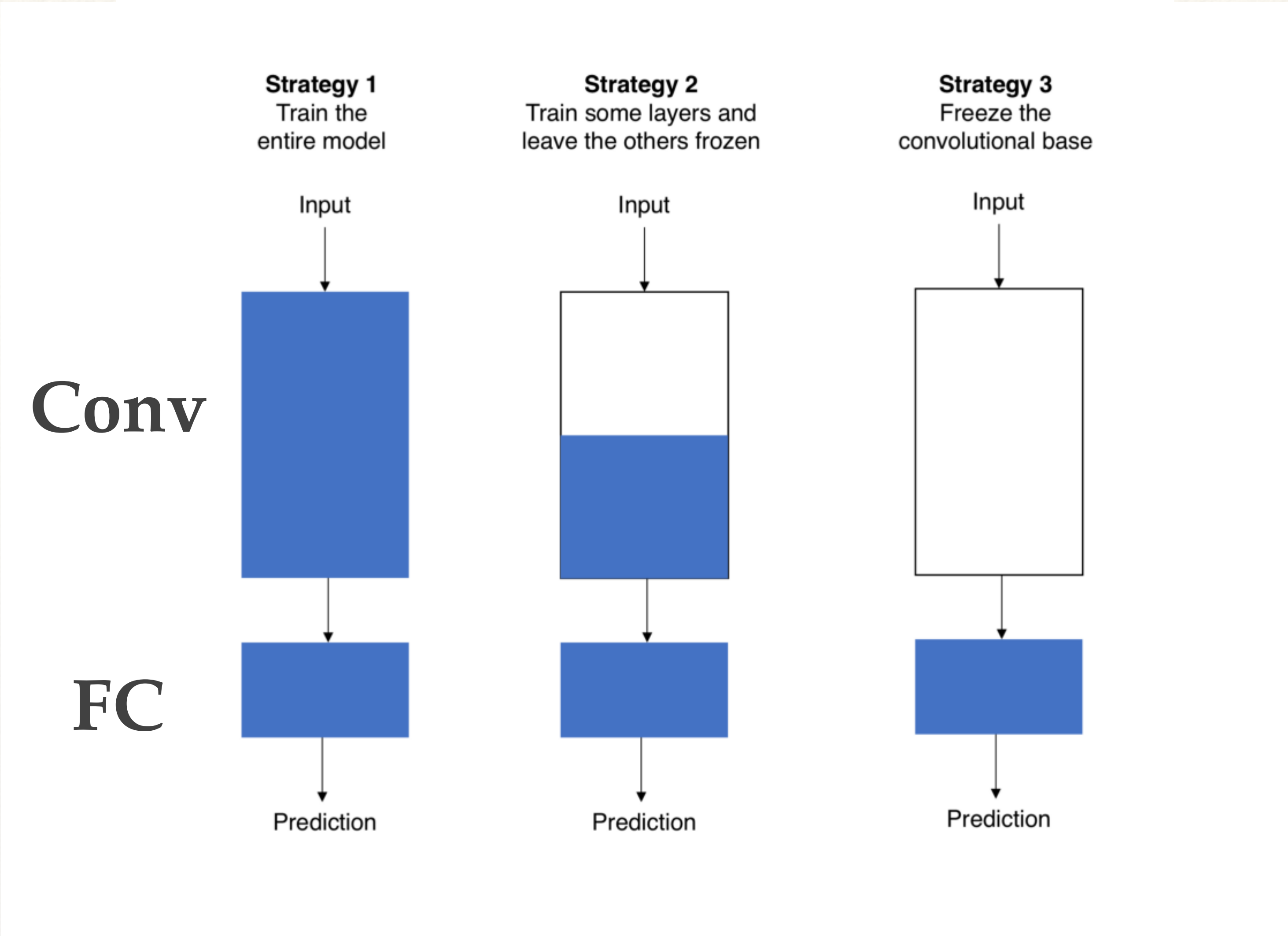
Output structure

But probably build a very similar NN architecture

Transfer learning: allows me to keep some of the architecture and initial computations (some weights in the hidden layers) and just re-run part of the network to specialise on this new problem

Increase AI's speed, reusability and generalisation

Repurposing pre-trained networks



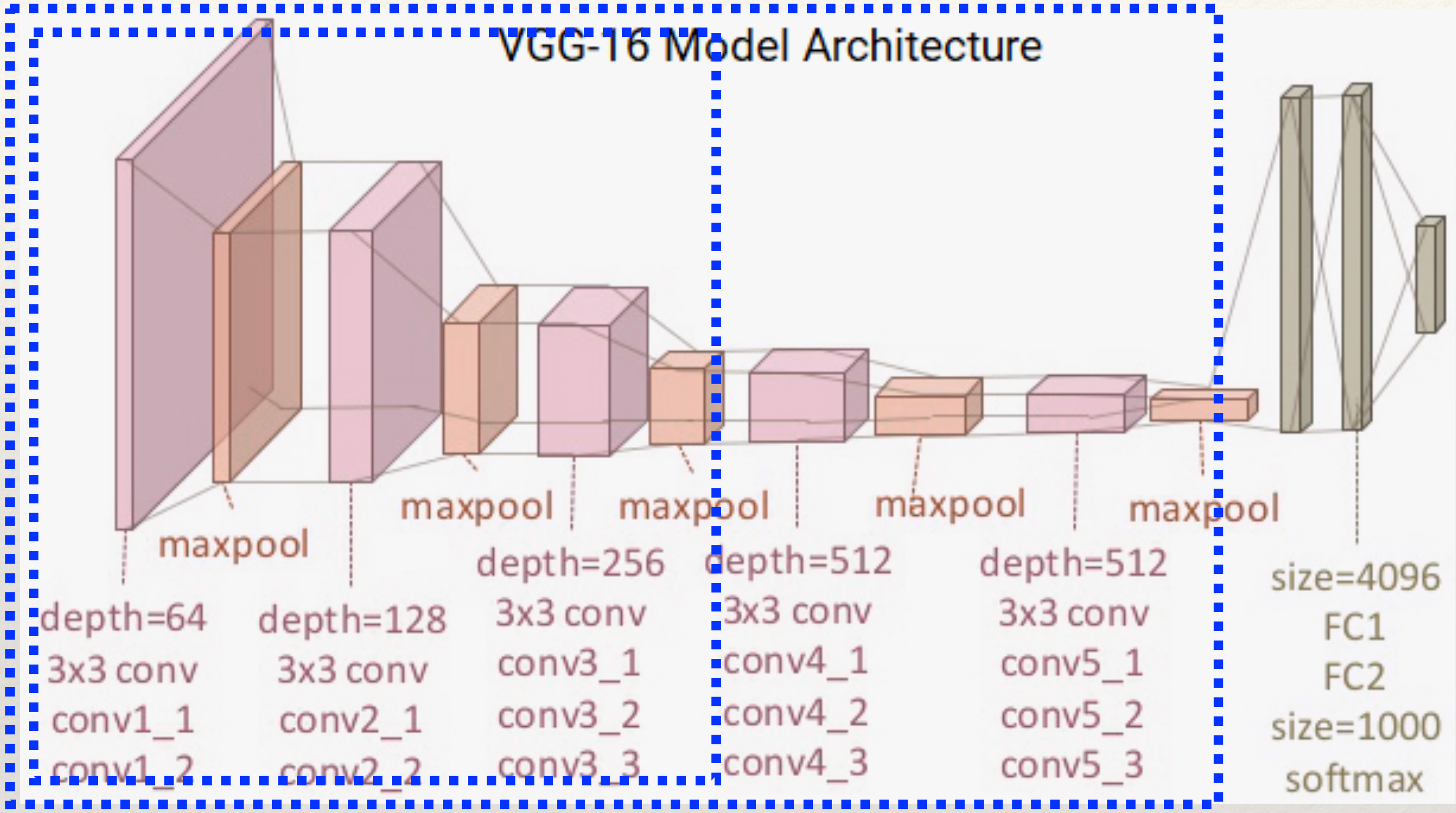
as we run the network, we update the weights to improve the accuracy
this weight updating is costly
we can **freeze** some of the weights that are generic for problem = image
and just adjust the later layers

All ML frameworks contained pre-trained models

Images==> VGG/ResNet/DenseNet/Inception/Xception…
Language (NLP) ==> Word2Vec, GloVe, FastText…

An example of pre-trained model: VGG16

Freeze initial layers and tune the rest



This strategy reduces a lot the computing time and helps generalising tasks

Data augmentation is typically used to make the procedure more robust

Going further?

In this notebook you can find some brief examples of transfer learning for MNIST and for a PP example
Use PYTORCH and FASTAI, syntax is compact
PP example: LHC Olympics 2020
images of boosted jets produced by SM and by a new heavy particle
small dataset, use augmentation

If you got time, check-out some of the newest applications, like *Feynman AI*
or Transfer Learning for music

“Experience is a lantern that you carry on your back and that only lights up the path you have traveled.” *Confucius*

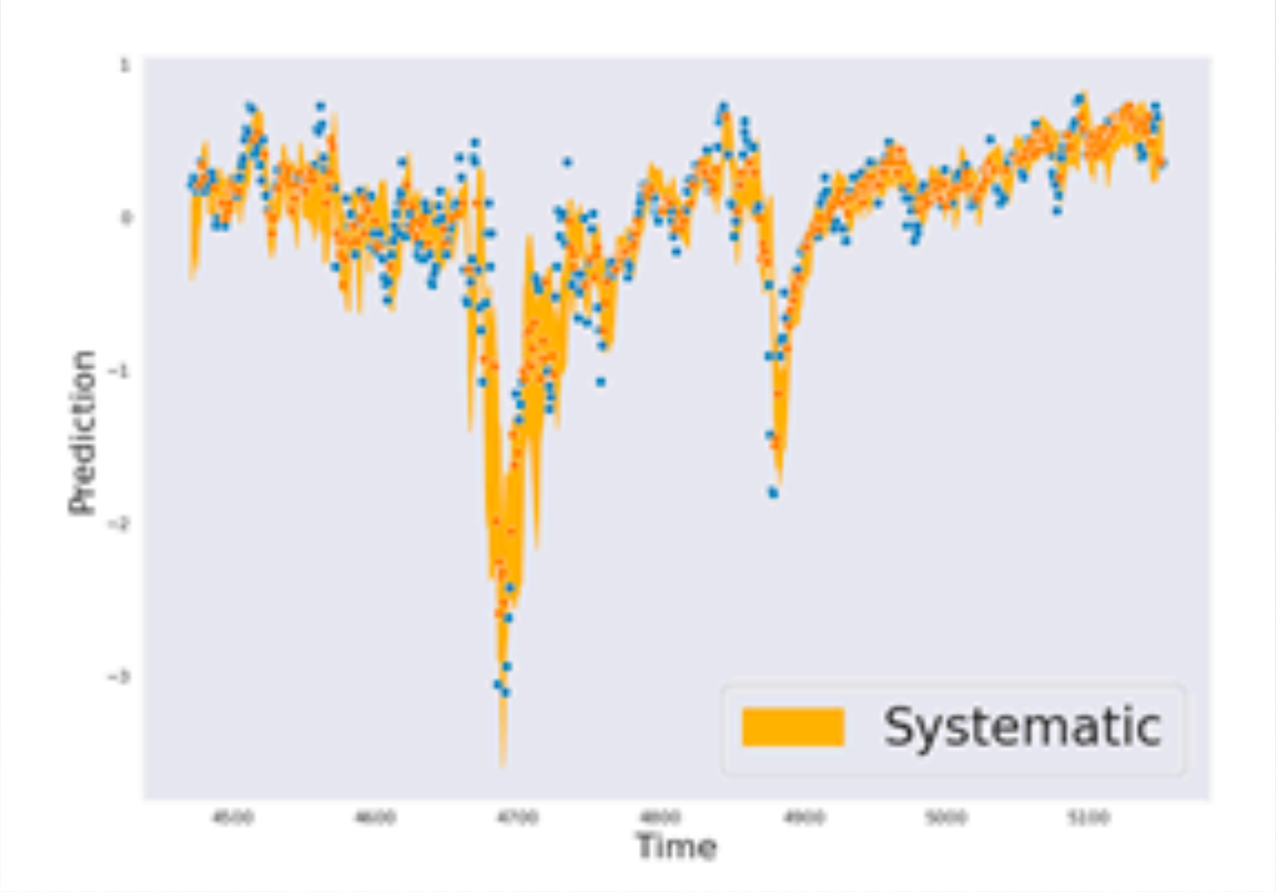
Knowing the past, predicting the future

predict the evolution of a situation

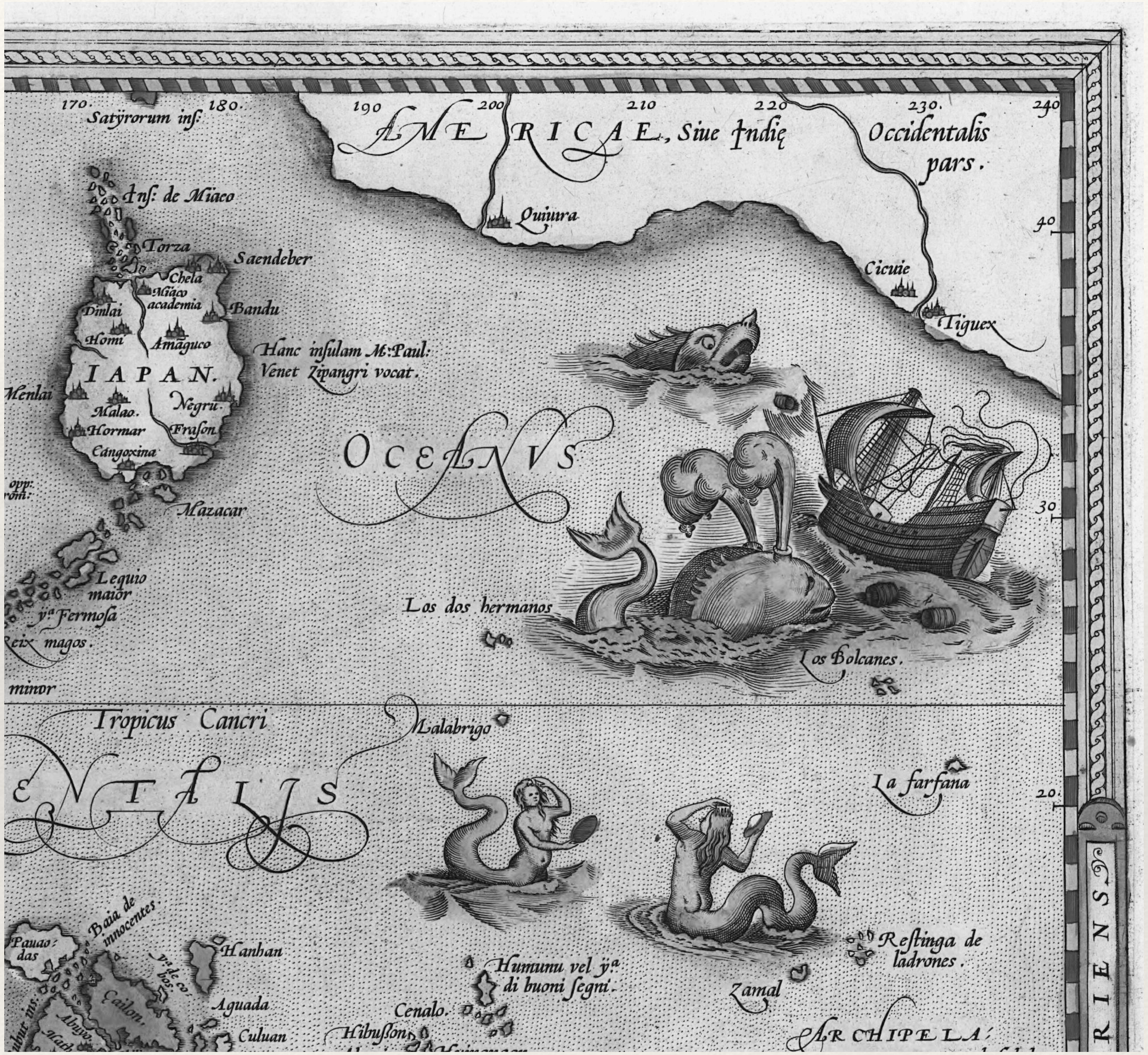
Additional

Never mind, Confucius!
ML *can* predict the future

By learning from examples of time series
(snapshots of past->future sequences)
and
using **RNNs (recurrent NNs)**
in particular **LSTMs (long short term memory)**



Time evolution of
the solar activity
blue-> reality
orange-> prediction



Here be dragons!

Going further

imagine new possibilities

Additional

What if we didn’t ask for an outcome?

Supervised learning input-> predict output

what if we just asked ‘look at this!’ with no determined output?

GANs (Generative Adversarial Networks)

and **VAEs (Variational AutoEncoders)**

In CNNs, benchmarks were cats/dogs and hand-written digits (MNIST)

Here, human faces

What if we didn't ask for an outcome?

Supervised learning input-> predict output

what if we just asked 'look at this!' with no determined output?

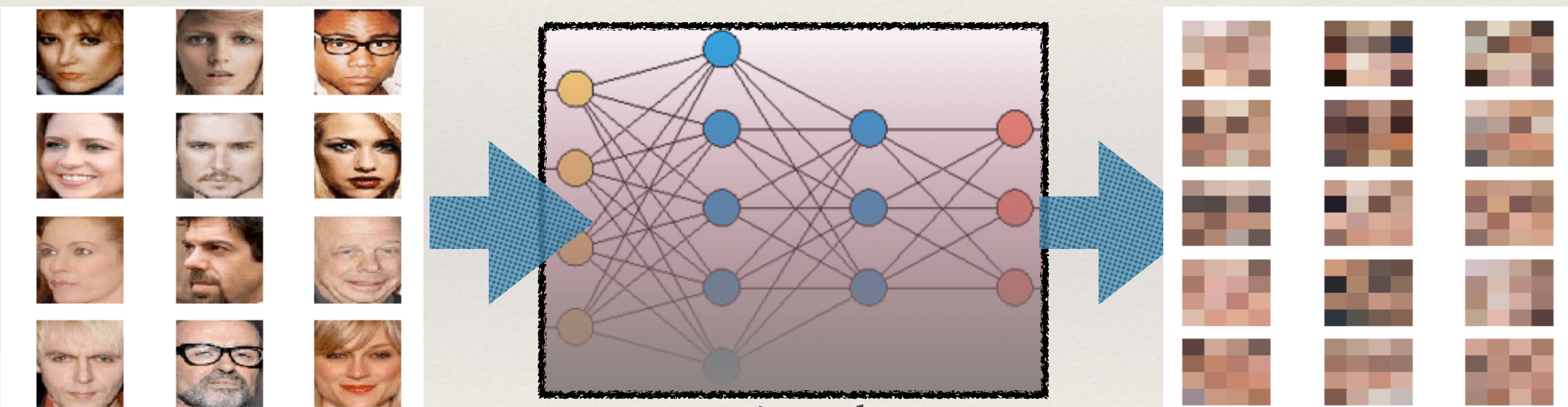
GANs (Generative Adversarial Networks)

and **VAEs (Variational AutoEncoders)**

In CNNs, benchmarks were cats/dogs and hand-written digits (MNIST)

Here, human faces

STEP 1 - 'LEARN' what is a human face



Take face images: x

Transform them in complicated ways

Create an avatar: x'

Doing this many times, while the DISCRIMINATOR says:

'You are going in the right direction', 'You are completely lost!'

What if we didn't ask for an outcome?

Supervised learning input-> predict output

what if we just asked 'look at this!' with no determined output?

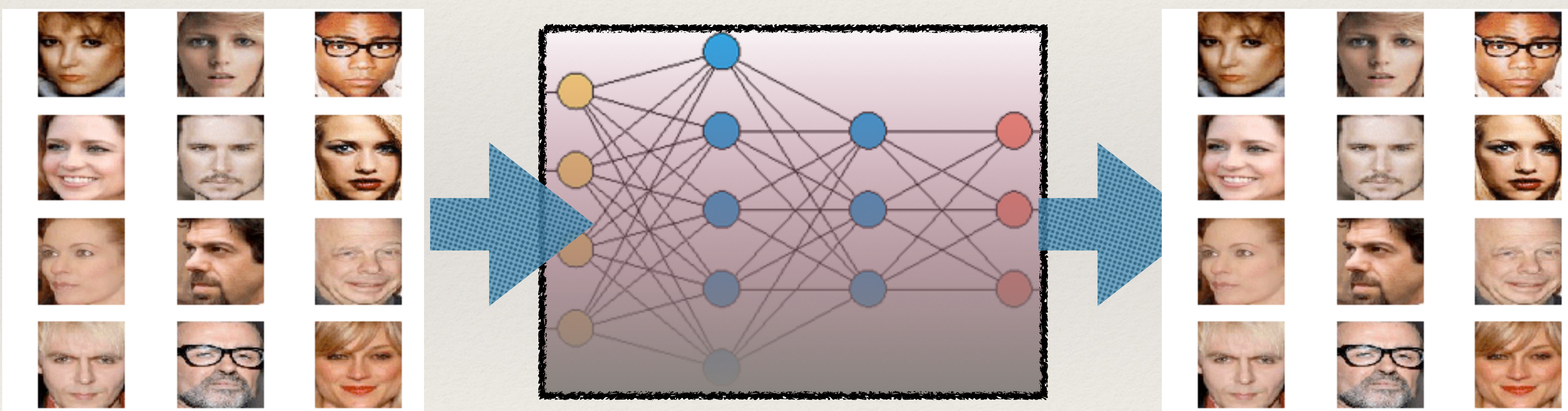
GANs (Generative Adversarial Networks)

and **VAEs (Variational AutoEncoders)**

In CNNs, benchmarks were cats/dogs and hand-written digits (MNIST)

Here, human faces

STEP 2- AFTER MANY ITERATIONS…



When the avatars are indistinguishable to the DISCRIMINATOR, **game is over**

What if we didn't ask for an outcome?

Supervised learning input-> predict output

what if we just asked 'look at this!' with no determined output?

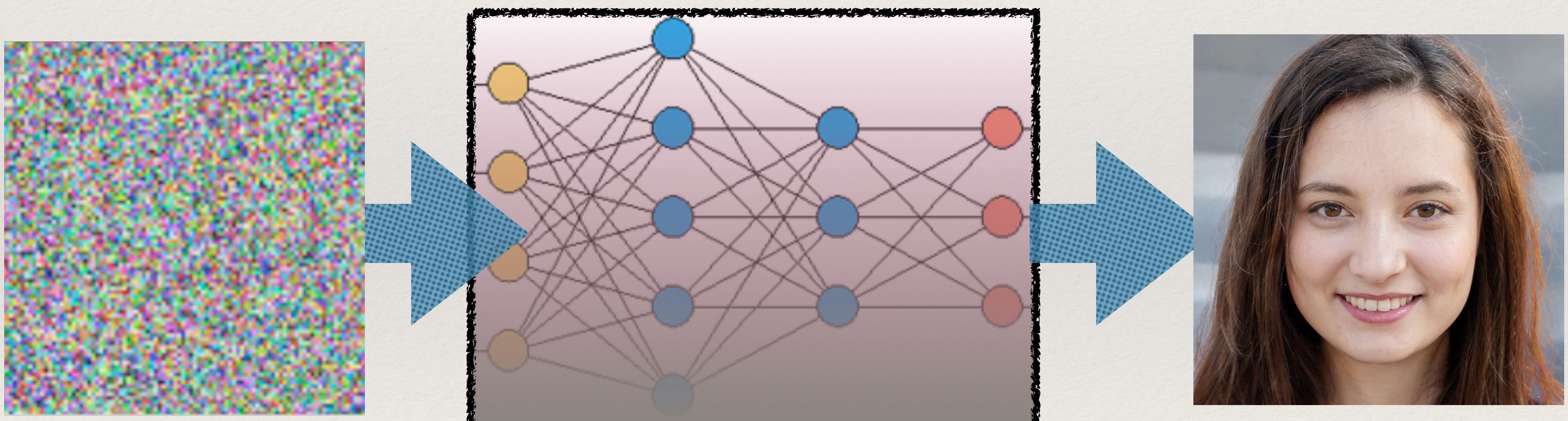
GANs (Generative Adversarial Networks)

and **VAEs (Variational AutoEncoders)**

In CNNs, benchmarks were cats/dogs and hand-written digits (MNIST)

Here, human faces

STEP 3- CREATE NEW POSSIBILITIES



This woman does not exist. It has been generated from noise.
The NN has learnt the concept of 'human face' and now can create human faces from noise

What if we didn't ask for an outcome?

Supervised learning input-> predict output

what if we just asked 'look at this!' with no determined output?

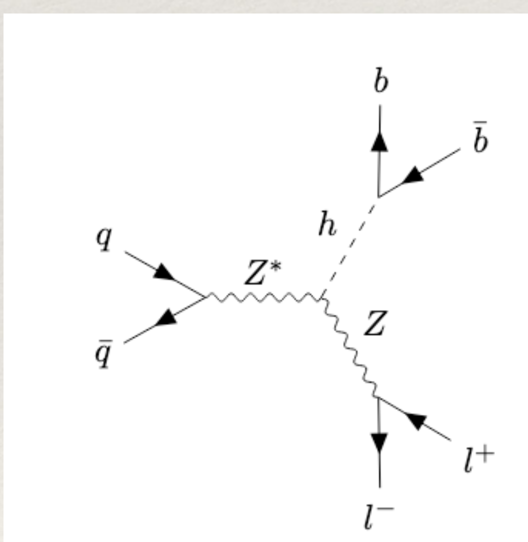
GANs (Generative Adversarial Networks)

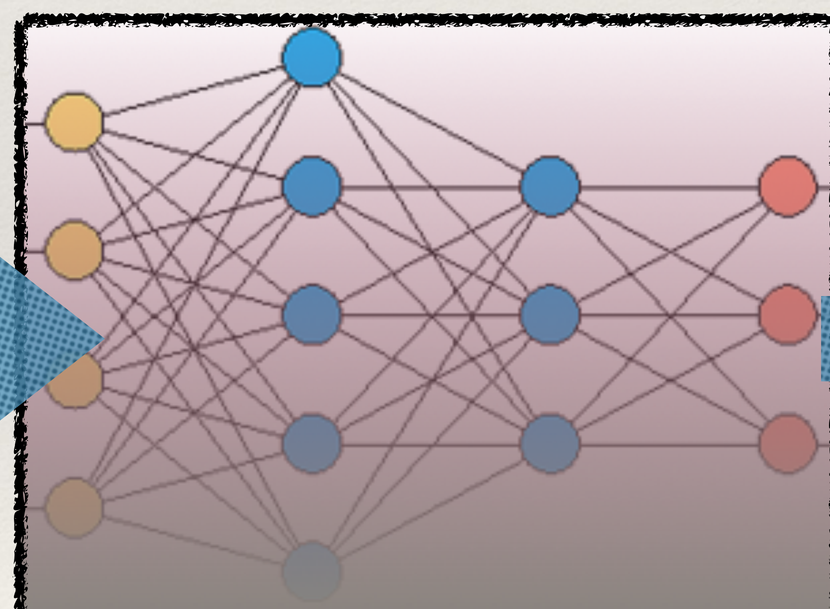
and **VAEs (Variational AutoEncoders)**

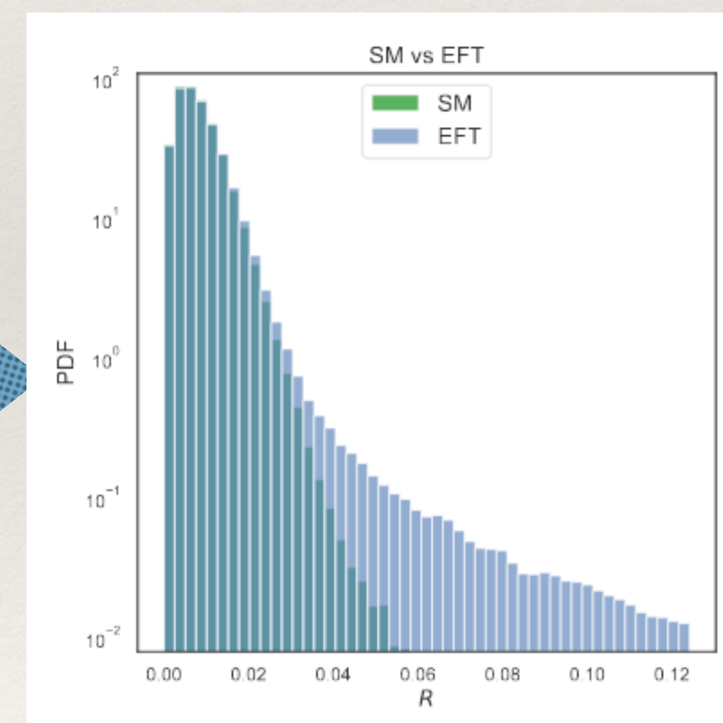
In CNNs, benchmarks were cats/dogs and hand-written digits (MNIST)

Here, human faces

Anomaly Detection







Ask to look only to Standard Model ('normal') events

Learns to ID outliers ('New Physics')

Summing up…

In this document: lots of links to learn more

Tomorrow: a hands-on starting from the dataset David gave you yesterday

We are just starting to understand the applications of ML in Physics

So far, dominated by the low-hanging fruit: supervised classification

ML brings added value, shortening data-taking times

They go beyond a mere iteration of our traditional statistical methods: unsupervised methods, generative AI, reinforcement learning…

Remember that through AI methods we could get interesting cross-pollination between our area (PP) and others

Opportunity to learn from other areas in Science

Keep exploring!