Intro to Neural Networks

1st COMCHA School, U Ramon Llull, La Salle

Veronica Sanz (Sussex & Alan Turing Institute & Valencia)

University of Sussex

The Alan Turing Institute

Vniversitat de València

What are we doing today?

- Learn basics of ML, including the binary problem because it is behind the Neural Network architecture
- Move onto Neural Networks, which are able to learn complex features with moderate computational demands, and best suited to Big Data problems
- Talk about Convolutional Neural Networks (CNNs) because their powerful performance is related to symmetries, hence applicable to Physics problems
- Finish the theory part with an eagle's view of NNs, to debunk a bit the myth of *black boxes*
- Move onto a **practical session** using a *Jupyter notebook* (interface to run python code) to build and train NNs

Machine Learning in a nutshell

Machine Learning

Data sample
(**in**-sample)
$\mathcal{D}(x_i, y_i)$
$i = 1 \dots n$
x_i inputs
y_i outputs

Understand / Learn
relations in/out
predict in->out
(**out**-sample)

Machine Learning

Data sample
(**in**-sample)
$\mathcal{D}(x_i, y_i)$
$i = 1 \dots n$
x_i inputs
y_i outputs

Understand / Learn
relations in/out
predict in->out
(**out**-sample)

( , diag1) ( , diag4)

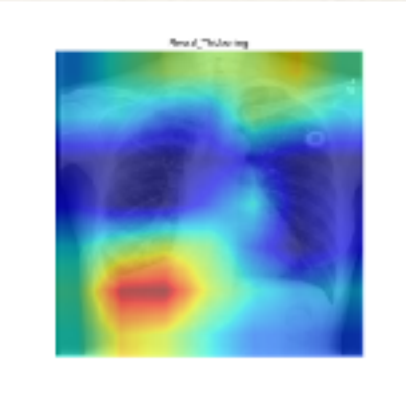
( , diag2) ( , diag5)

( , diag3) ( ,diag6)

...

diagnosis?

features?



Machine Learning

Learning depends on

DATA: amount, quality (*stats&syst*)

VARIANCE

MODEL COMPLEXITY: how we interpret the data

all inputs and outputs?

assumptions $y(x)$ *e.g.* $y(x) = \sum_p a_p x^p$

BIAS

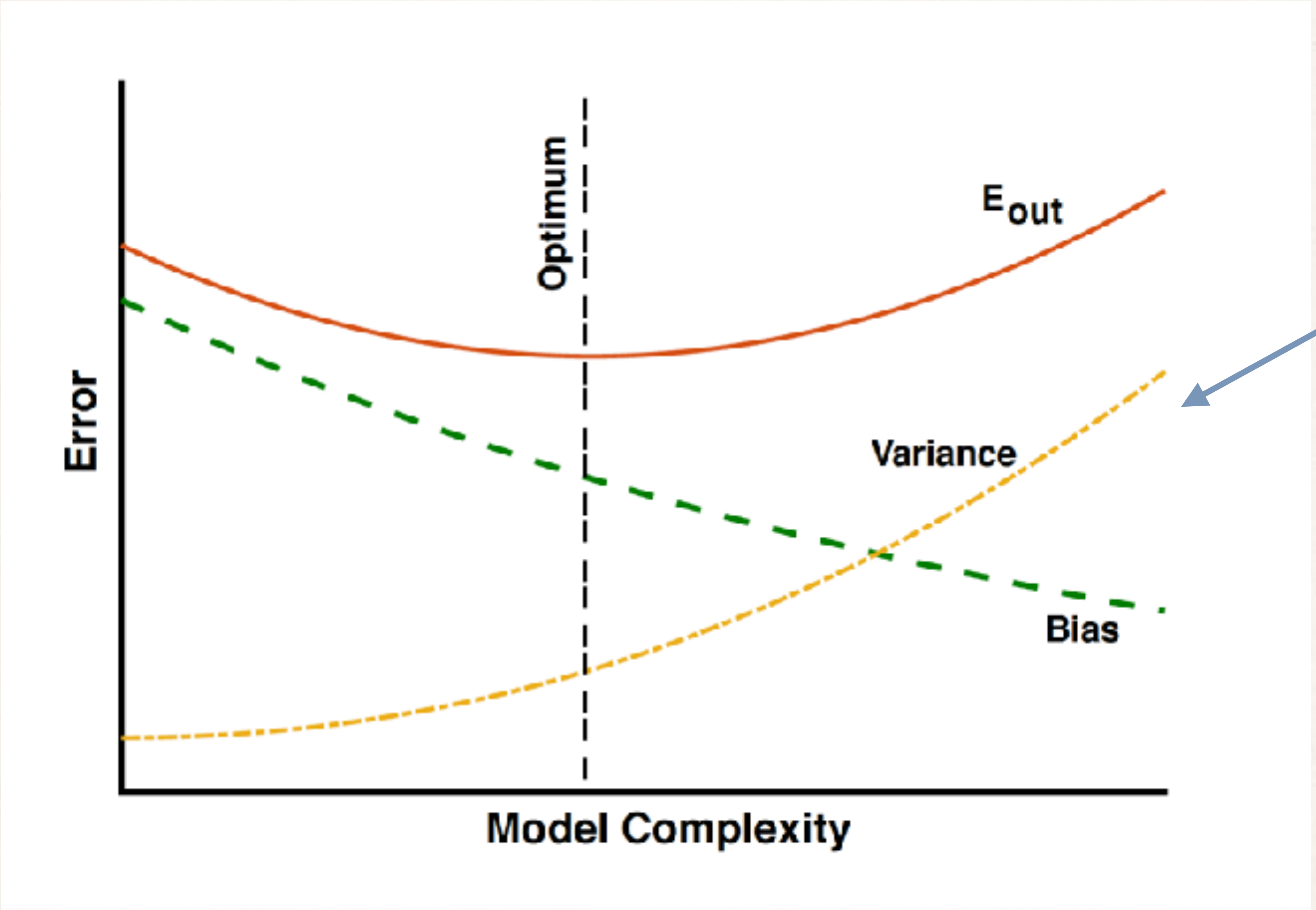
COMPUTATIONAL LIMITATIONS:

limited resources

architectural bias (von Neumann)

Variance-bias trade-off

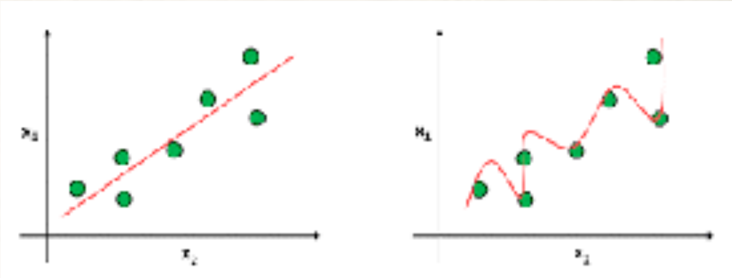
for a fixed dataset



few features

many features

at some point,
not enough data to
reliably account for
features
OVERFITTING



when comparing with
out-sample, bad model

Understand data?

diagnosis?

Understand could mean getting better at *predicting* behavior

Assuming some functional dependence $y(x)$

& getting better at obtaining this function

(e.g. with a polynomial fit, a better value for the coefficients)

so that given a new set of inputs, we predict the outputs to a high precision

minimize difference between **(what we observe - what we predicted)**

COST FUNCTION = some functional dependence on this difference

Understand data?

diagnosis?

e.g. SQUARED ERROR over a dataset $\mathcal{D}(x_i, y_i)$

minimize $\mathcal{C}(\mathcal{D}) = \sum_{i=1}^{n} (y_i - \hat{y}(x_i))^2$

to obtain $y(x) = f(w.x + b)$

weight bias

Understand data?

diagnosis?

e.g. SQUARED ERROR over a dataset $\mathcal{D}(x_i, y_i)$

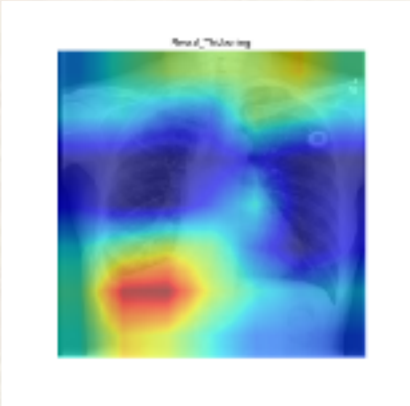
minimize $\mathcal{C}(\mathcal{D}) = \sum_{i=1}^{n} (y_i - \hat{y}(x_i))^2$

to obtain $y(x) = f(w.x + b)$

weight bias

INPUTS *in the X-Ray example* **OUTPUTS**

x image information, levels of grey with some binning 2D images: 28X28, 124X124 etc	y diagnostic 0=healthy, 1=ill 1=pneumothorax, 2=actealasis, 3=…



weights: signal how important is each region in the image
bias: just a shift

Understand data?

diagnosis?

e.g. SQUARED ERROR over a dataset $\mathcal{D}(x_i, y_i)$

minimize $\mathcal{C}(\mathcal{D}) = \sum_{i=1}^{n} (y_i - \hat{y}(x_i))^2$

to obtain $y(x) = f(w.x + b)$

weight bias

Minimization done numerically

the cost function can be *very complex*, have many minima

often use ***regularization***: term in the cost function which diminishes importance of features, so they don't dominate the fit

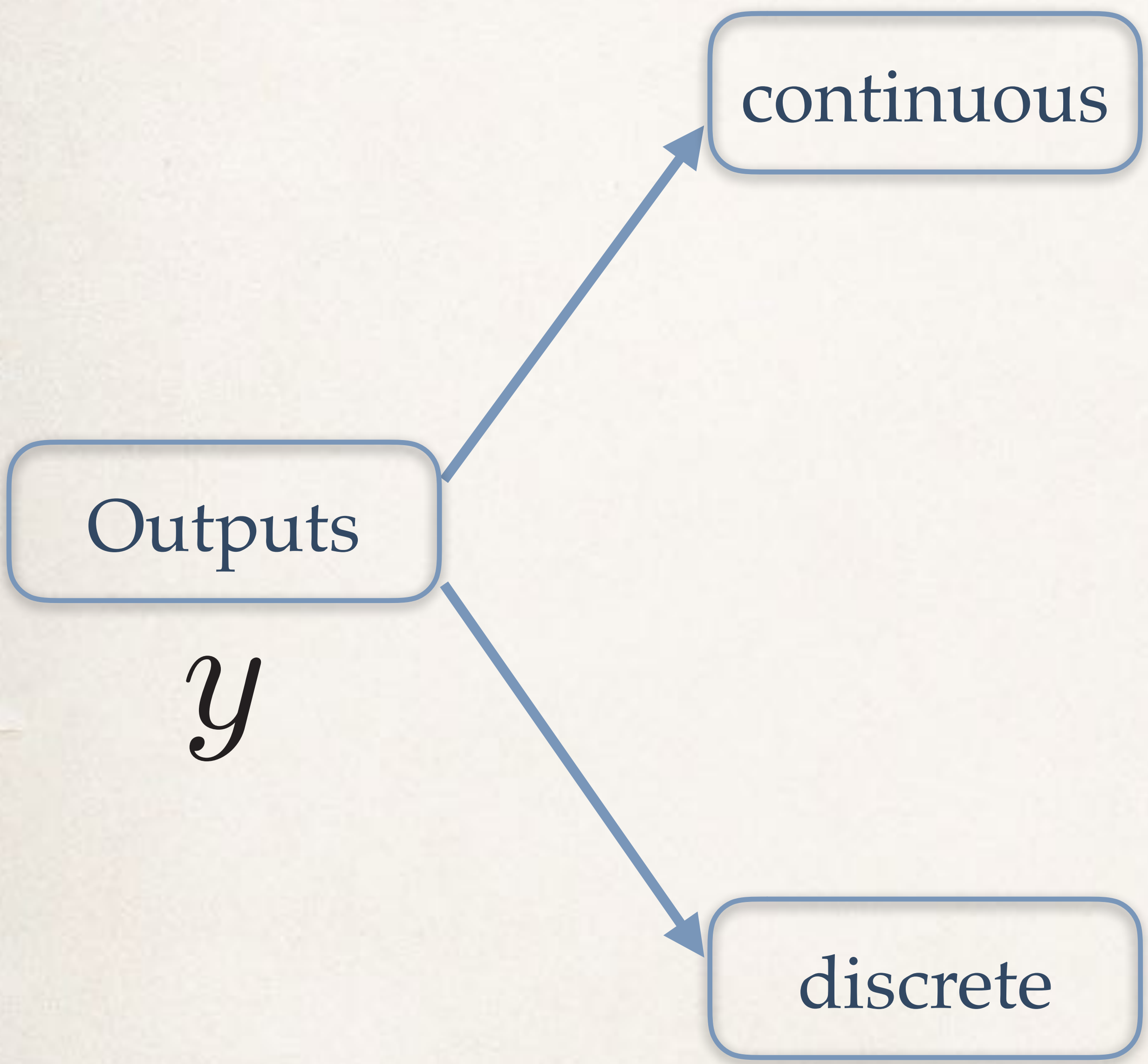
Apart from numerics, which soon become a **BLACK BOX**

what are we (humans) introducing as a bias?

1. What we want to minimize : the cost function

2. The way we view the data: functional dependence f

Types of problems



x: characteristics of proton-proton collisions

y: total number of events, kinematic distributions…

CLASSIFICATION

x: characteristics of an event

y: b-tag or no b-tag, quark or gluon jet, EM/HAD…

Let's talk about the binary problem

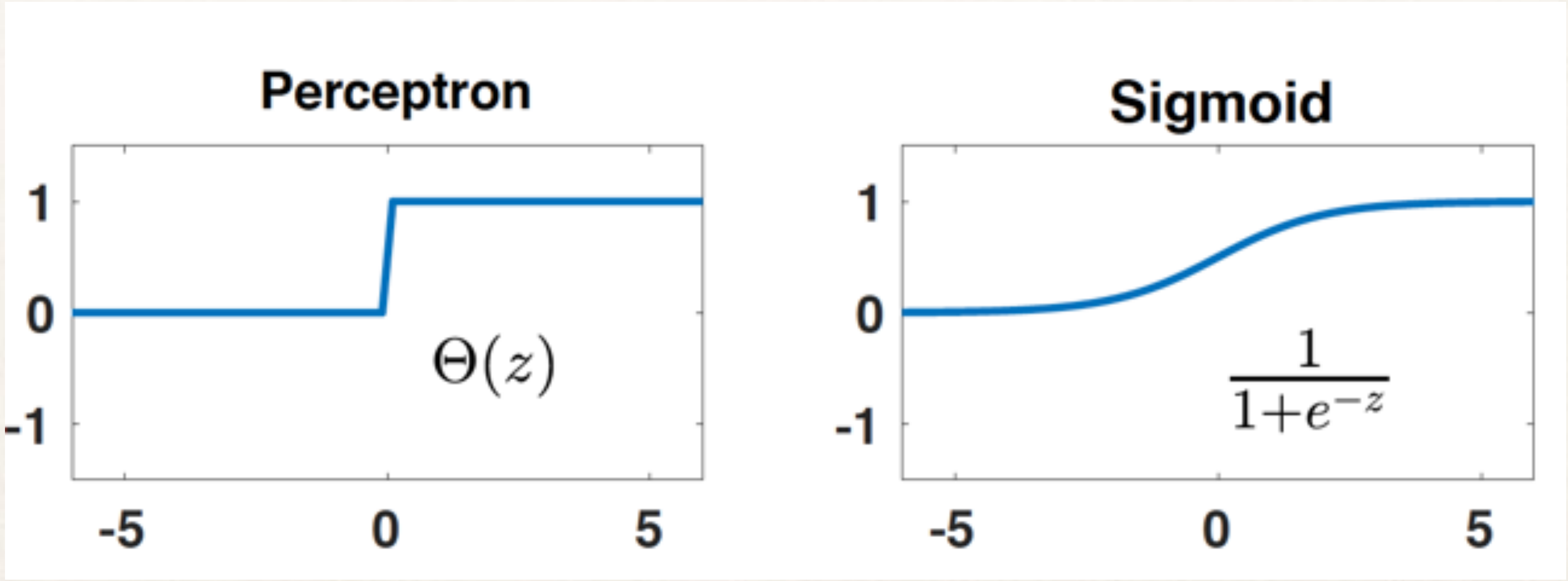
dataset $\mathcal{D}(x_i, y_i)$ with $y \in \{0, 1\}$ {no, yes}

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

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

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

f itself can be a function within 0 and 1



Example of cost function: cross-entropy

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*

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

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

supplemented with regularization terms

This is *just* a minimization problem

The best model (*w*) is the one that satisfies

$$\mathbf{0} = \nabla \mathcal{C}(\mathbf{w}) = \sum_{i=1}^{n} \left[f(\mathbf{x}_i^T \mathbf{w}) - y_i \right] \mathbf{x}_i,$$

Nice equations, but what does this mean in practice?

we need to devise a way to obtain the values *w* which minimize the cost function, and this in general is a very complex procedure which involves taking **derivatives** of the cost function respect to the parameters

derivatives should not blow up
you should not get stuck on a local minimum
you should not hop too far and miss minima
and your model should work well on new data

all this, in a high-dimensional parameter space…

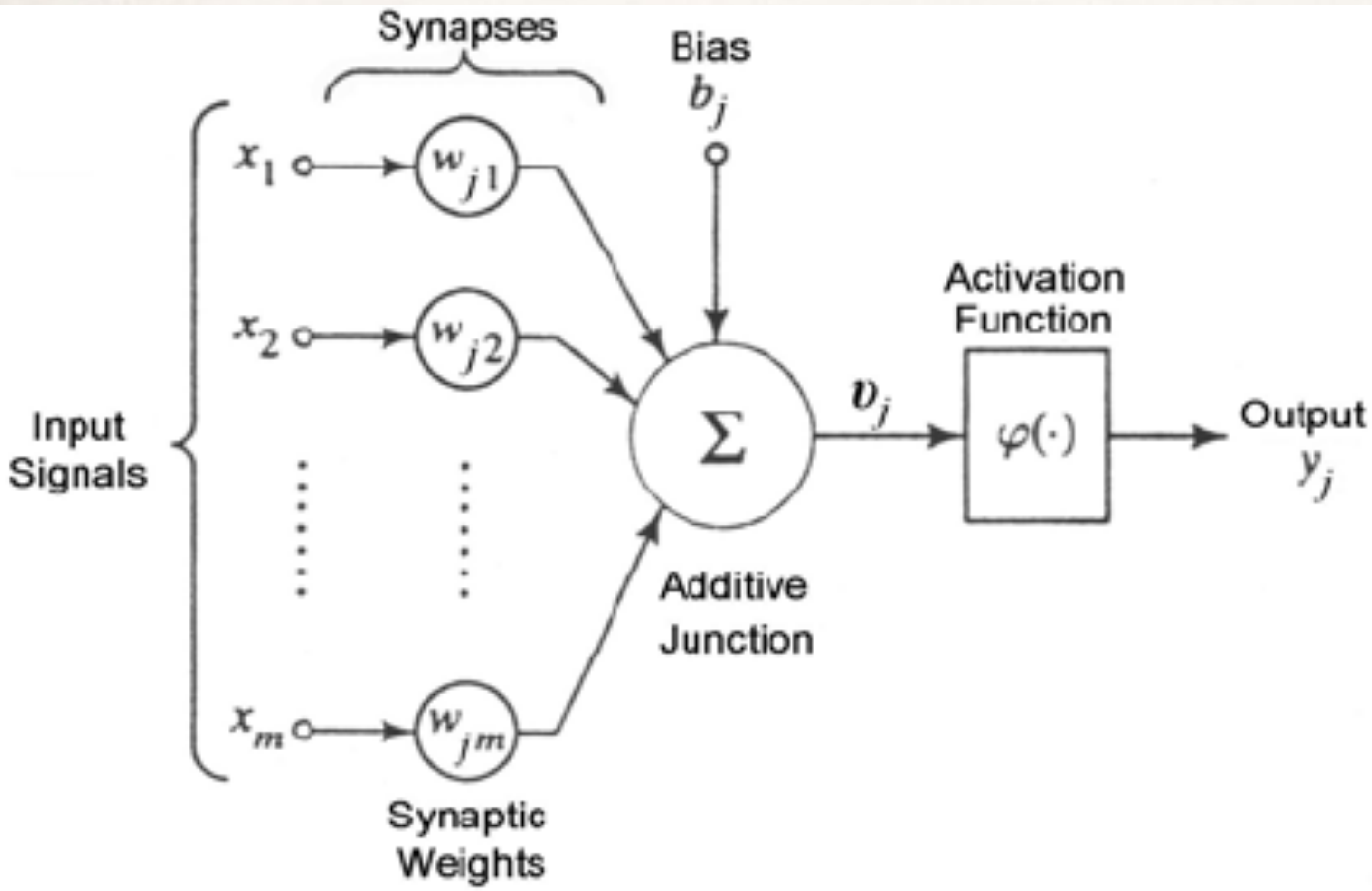
Neural Networks

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

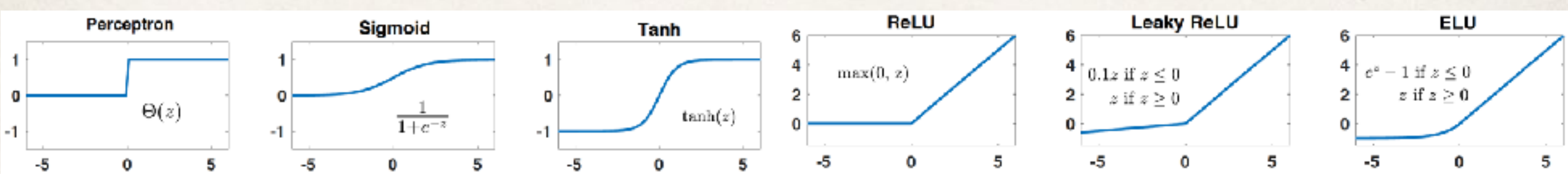
$$y = f(z)$$

y: **output, scalar**

(passes information, or not)

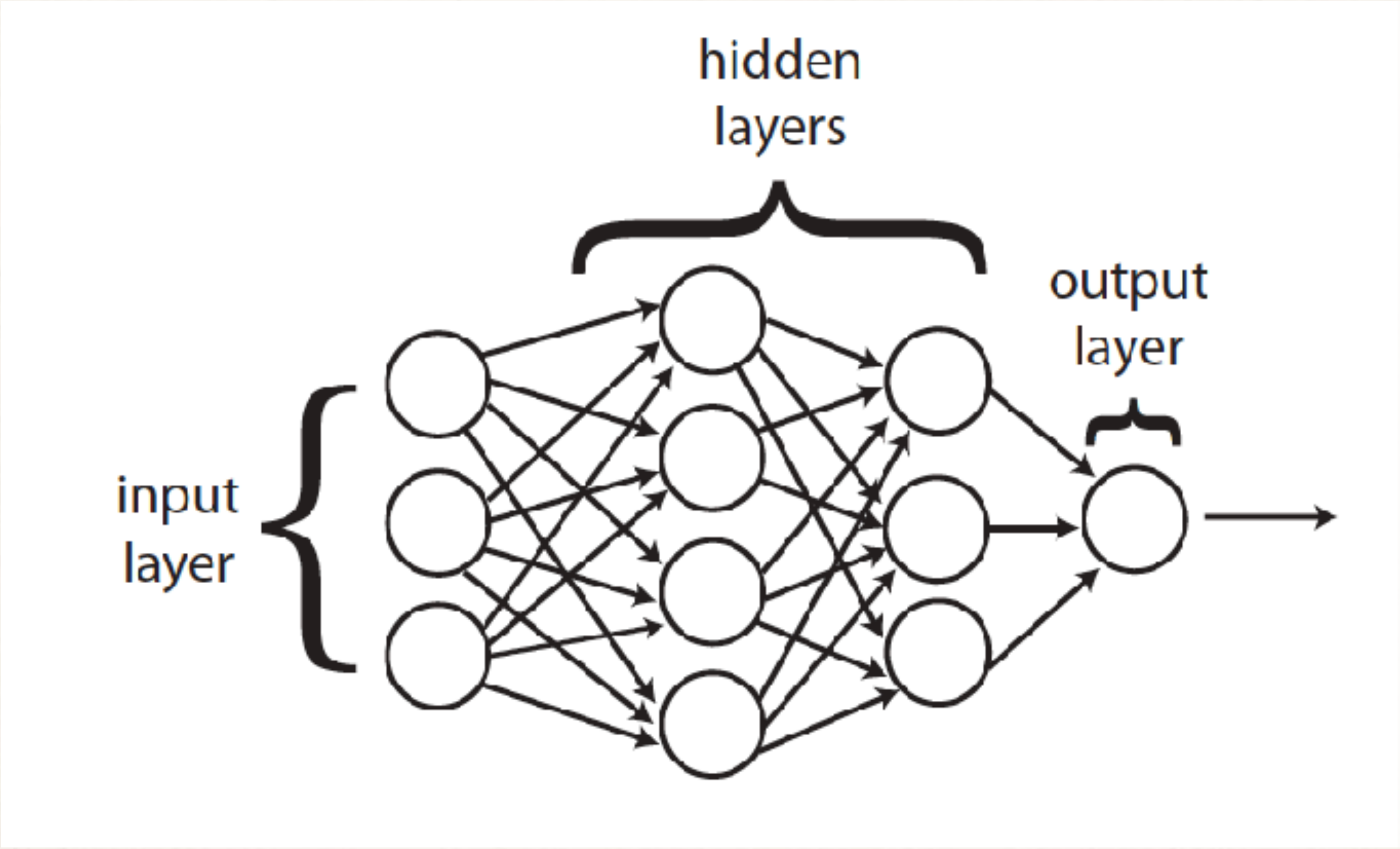
f: **activation function**

Examples of activation functions



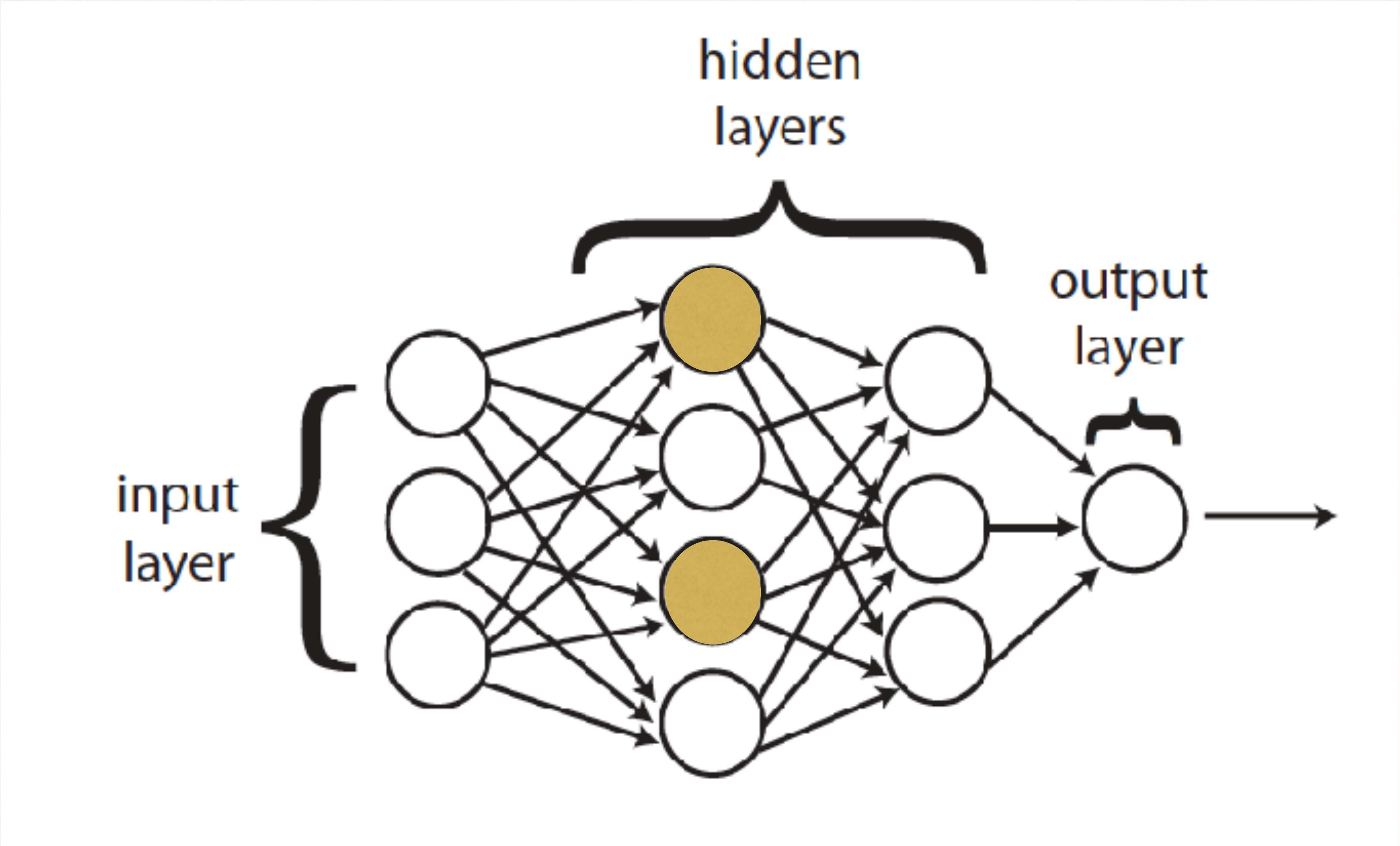
NN Architecture

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



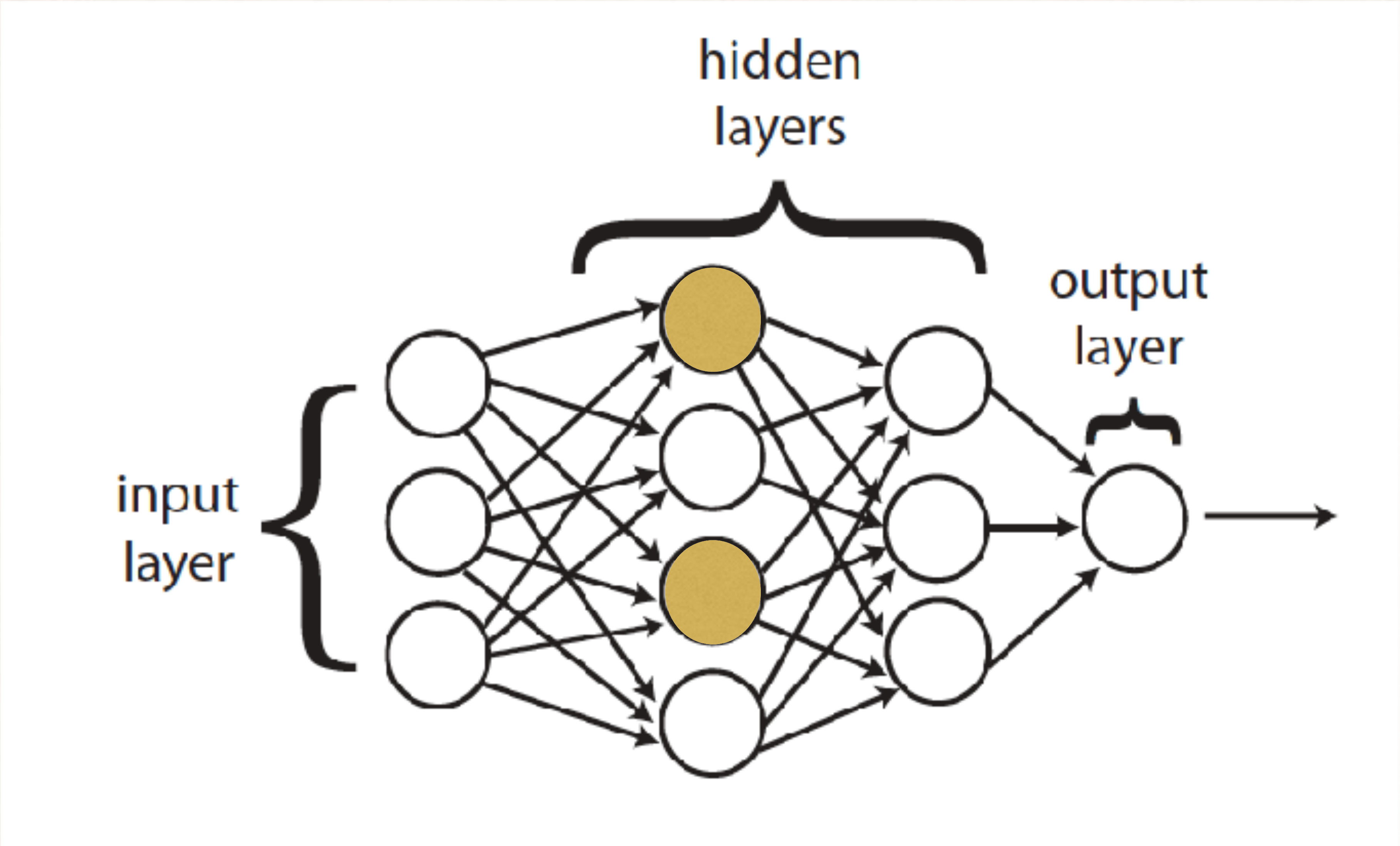
each circle is a neuron,
where the inputs (in-arrows) are transformed into output (out-arrows)
the outputs of each layer serve as input for the next

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?

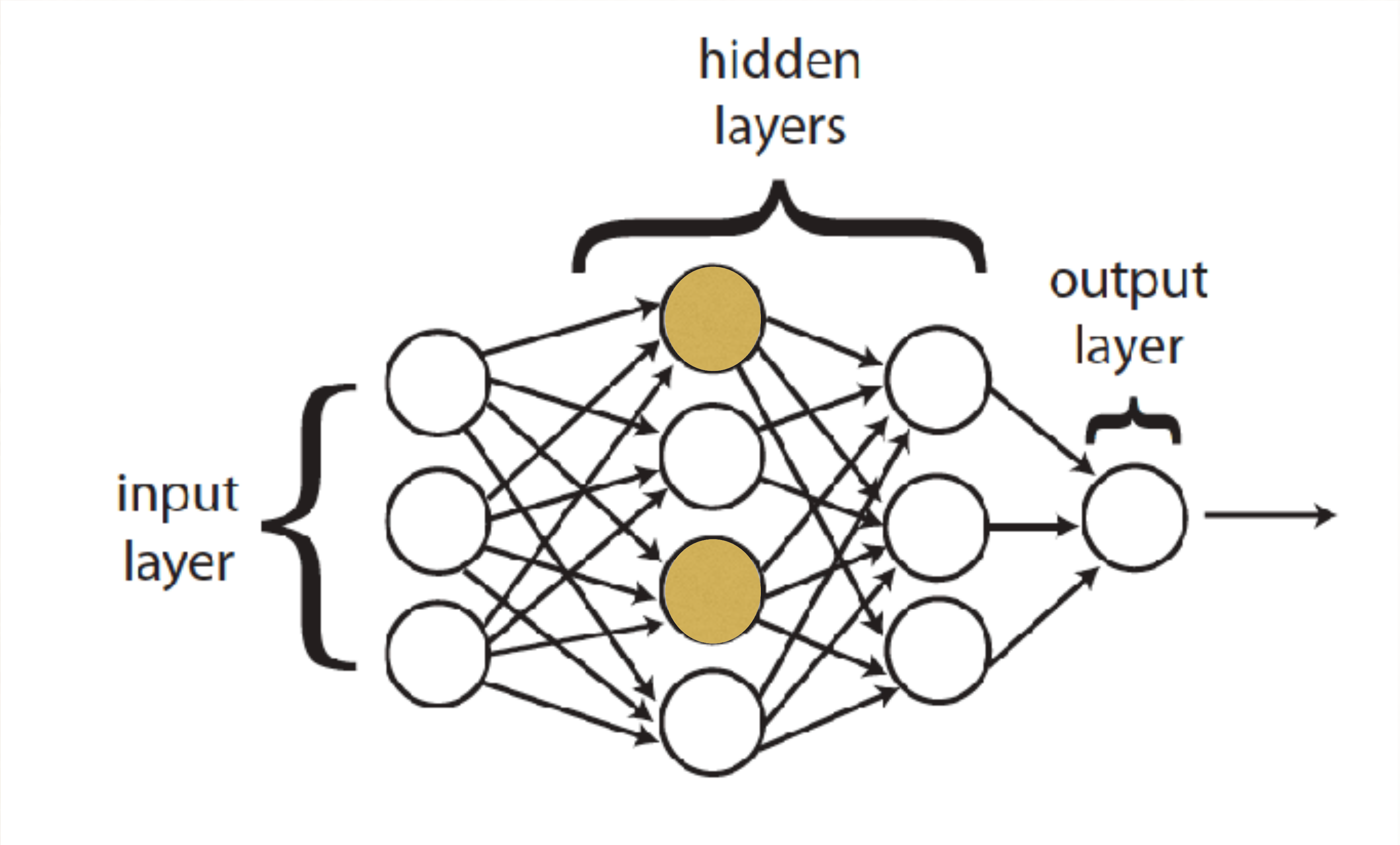


This NN transforms
inputs (at the input layer) into an output (output layer)

$$y(x)$$

which couldn't be captured by simple functional forms

Why are we doing this?

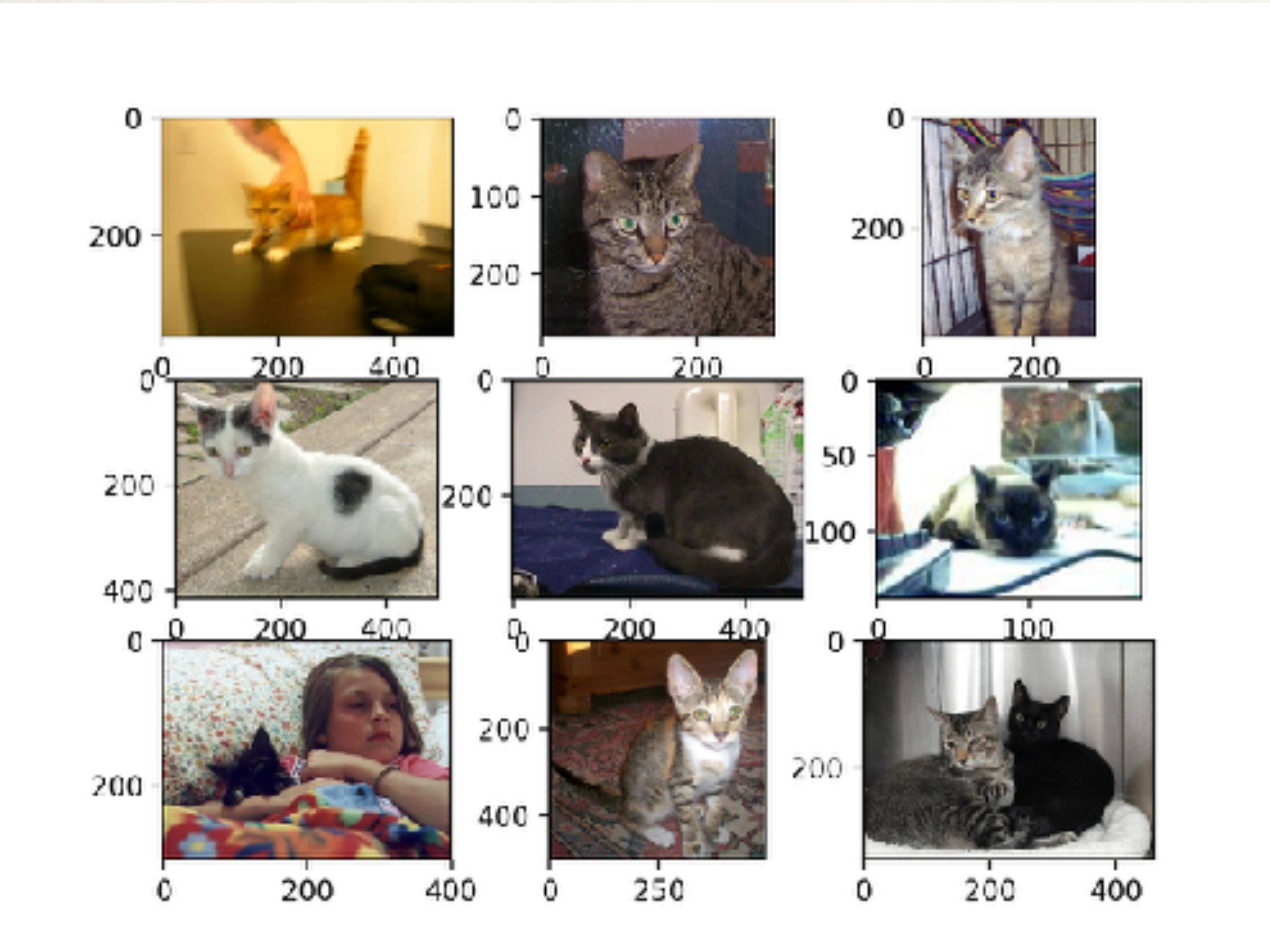


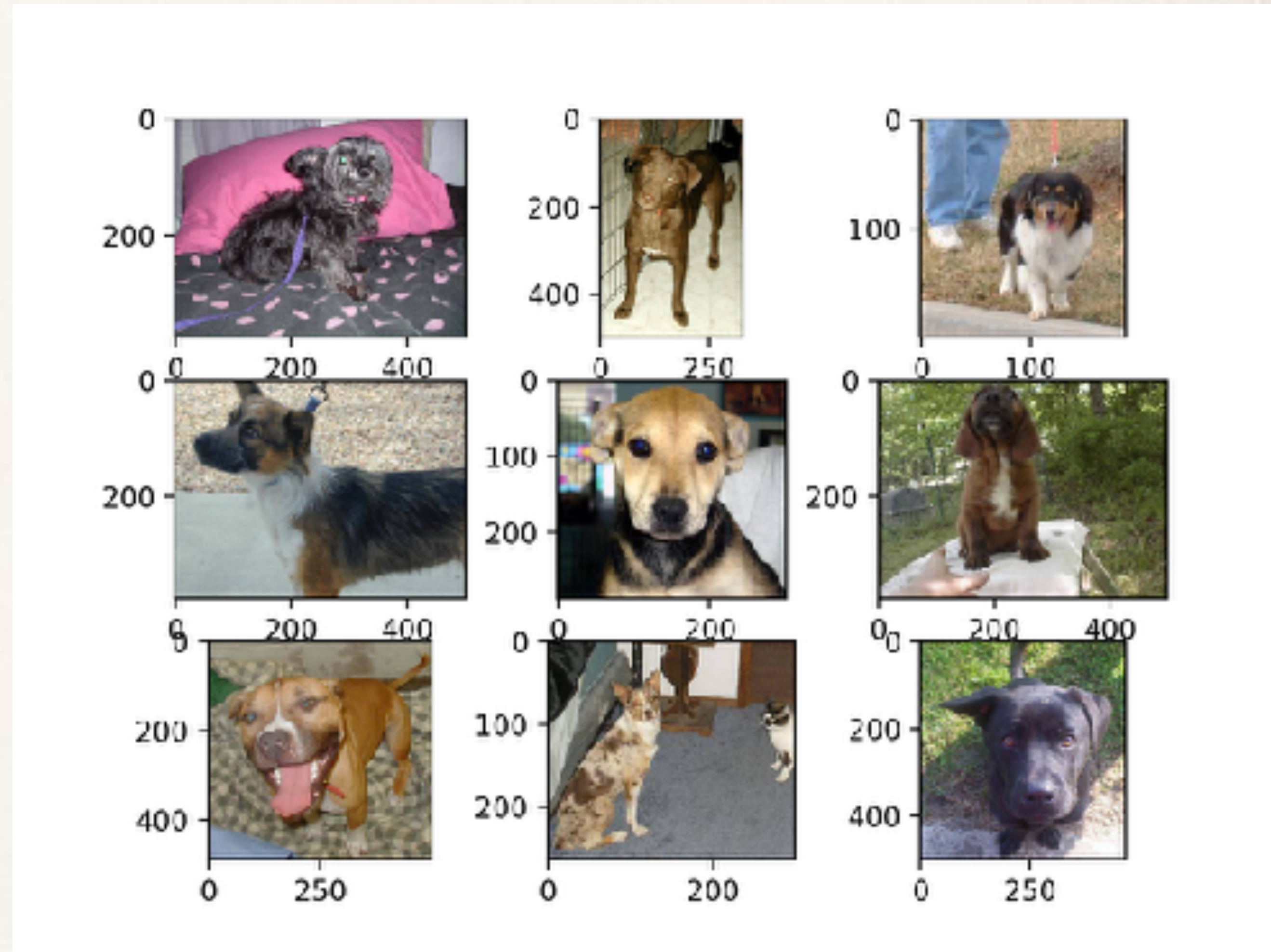
Neural Networks can model **complexity**
They have a high degree of expressivity
/exhibit high representational power
More hidden layers=> more complex features
Deep learning, deep NN

Complex features

images, speech : are complex

For example: cats/dogs



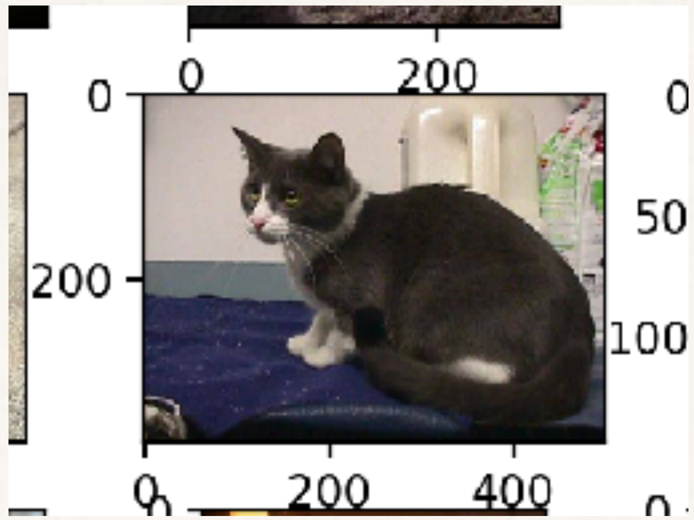


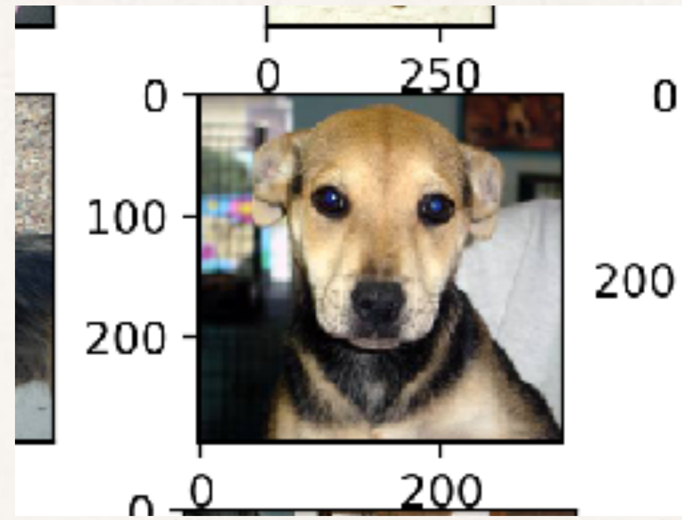
you can distinguish these cats and dogs, right? but how?

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*



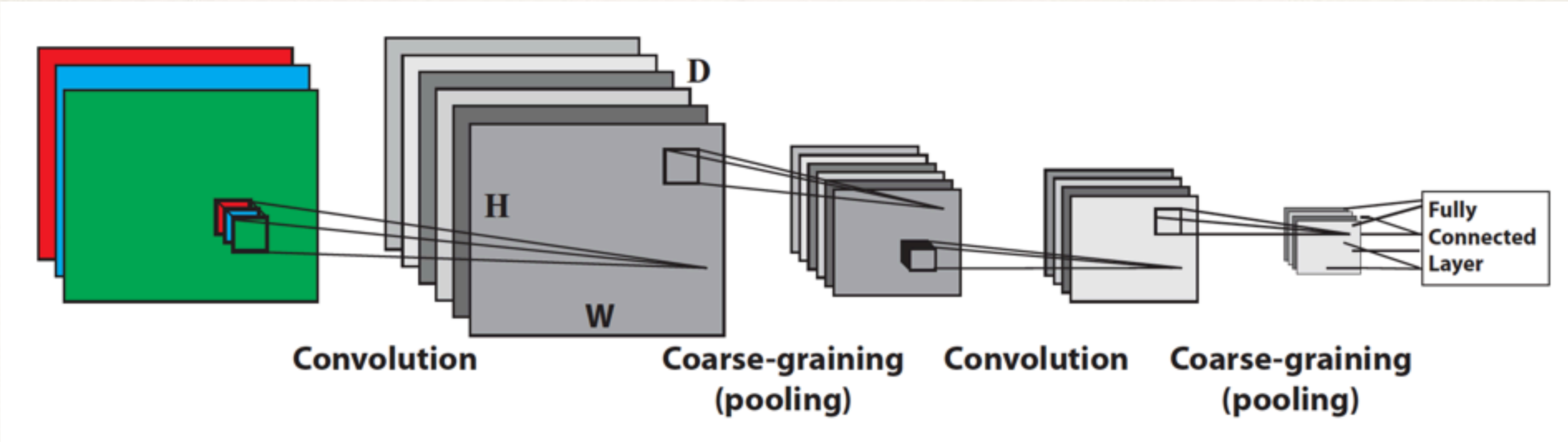


Apart from shape and color,
we know a cat is a cat because there are relations among their features, e.g. the position of the eyes/ears respect to the head centre, independently of where in the image the cat is

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

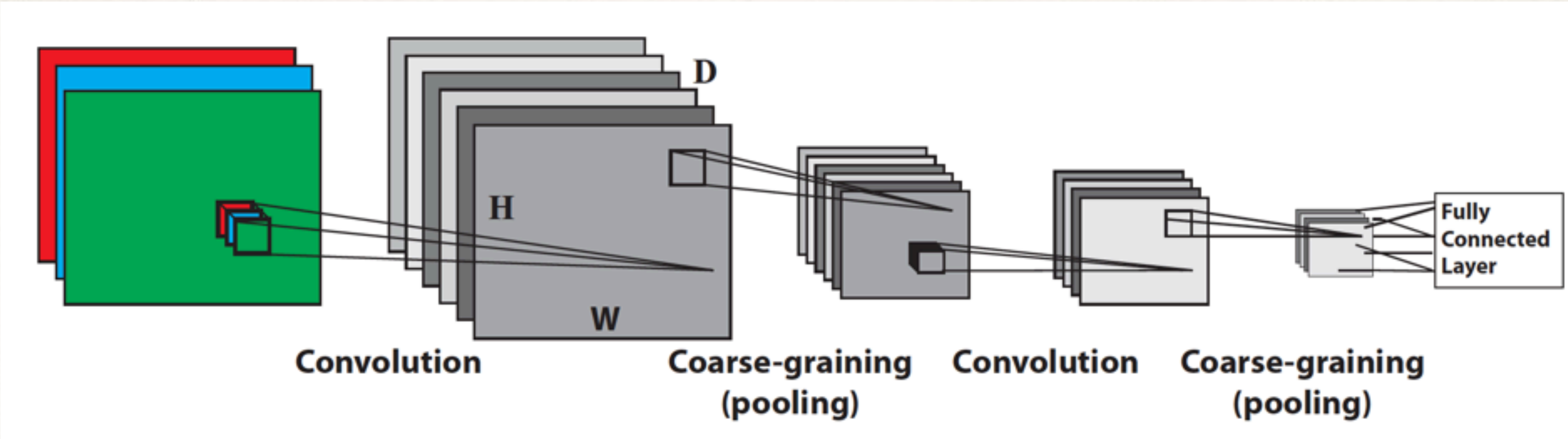


Two types of basic layers

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*

NNs from above

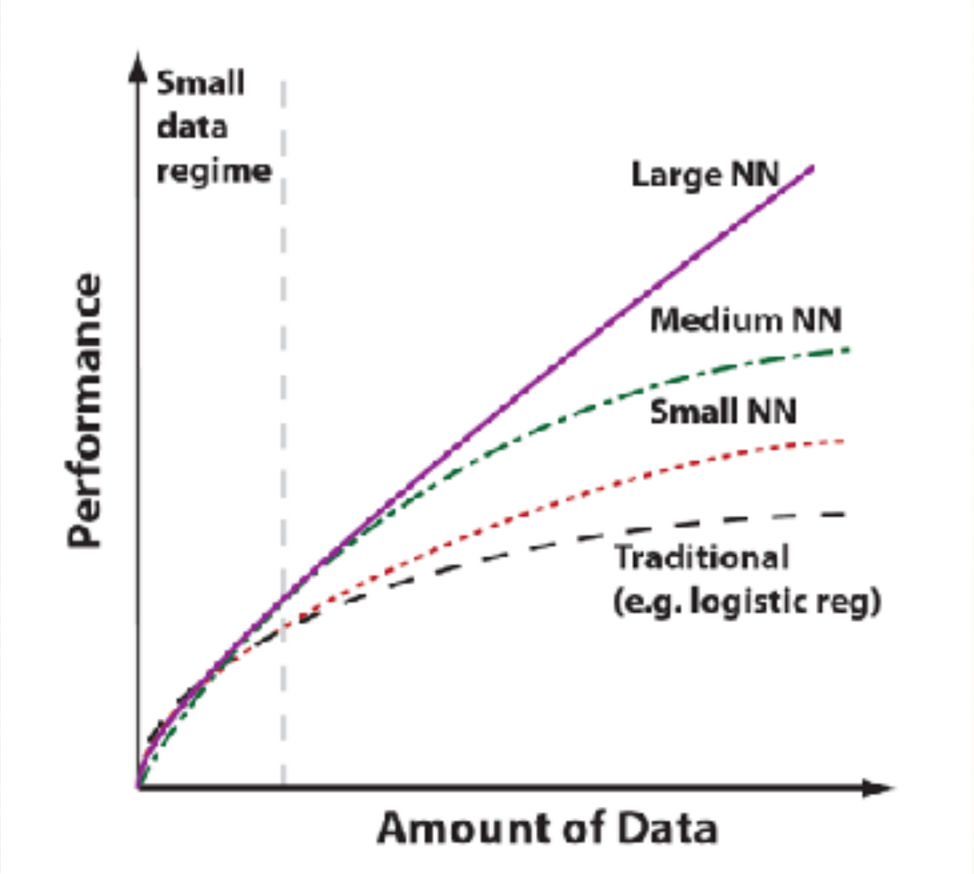
Why are NNs so good at learning

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 parallelization, use of GPUs…

It scales better than other learning methods (like SVMs)

In practice

Isn't coding DNNs super complicated?

Nope!
Thanks to packages where all the building blocks and necessary transformations are just one command away

most popular
keras, pytorch, tensorflow

they are slightly different
to start diving into ML keras is best
(simpler syntax)
but slower when dealing with large datasets
Tensorflow has keras in it

What we will do next

We will take a standard dataset, MNIST

Build and train a NN
to become better at recognising hand-written numbers
This is a *supervised* ML problem
(we know the true labels)
we train on a large sample (60K) images

We will build a fully connected NN,
a Convolutional Neural Network,
and use Data Augmentation
Our precision will go from 96% till 99%
find the link to the notebook here

https://github.com/vsanz/COMCHA_NN

Usual work-flow

1. **Collect and pre-process the data.**
2. **Define the model and its architecture.**
3. **Choose the cost function and the optimizer.**
4. **Train the model.**
5. **Evaluate and *study* the model performance on the validation and test data.**
6. **Adjust the hyperparameters (and, if necessary, network architecture) to optimize performance for the specific dataset.**