An introduction to deep learning

The perspective via inductive biases

Blake Richards

Mila/McGill/CIFAR

What this talk won't be

This talk is not going to be a tutorial on how to build deep neural networks for analyzing your data, nor an explanation of the mathematics involved. There are *thousands* of resources for that, and ultimately, you must consider your specific application.

For a basic intro to coding them up, I recommend the **PyTorch tutorials**.

For a more in-depth overview of deep learning I recommend Goodfellow, Bengio and Courville's book titled **Deep Learning**.

What this talk will be

The goal of this talk is to give you a sense of what deep learning is actually good for, and what it's not good for.

My hope is that this will give you the wherewithal to know when deep learning is the right strategy for you to take in your research, and when it is not.

Outline

1. No free lunch in AI
2. Inductive biases and the AI set
3. Deep architectures as a good inductive bias
4. Making learning work in deep architectures
5. When to use deep learning

No free lunch in AI

Can we develop general purpose learning?

The early dream of machine learning researchers was that we could develop general purpose learning algorithms that could learn well in any scenario on any dataset.

Duct tape for AI as it were... some neuroscientists still think of Hebbian plasticity this way.

Crushing our hopes and dreams

It turns out this dream is an impossibility

There exists no learning algorithm that can perform better than all other learning algorithms on all tasks.

Crushing our hopes and dreams

We know this harsh reality due to the
No Free Lunch Theorem for Optimization, which was given by
Wolpert & MacReady (1997).

See: Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. IEEE transact. on evolutionary comput., 1(1), 67-82.

The No Free Lunch Theorem

Consider the class of all possible loss functions, $f \in \mathcal{F}$. We can view these as functions from data points, $x \in \mathcal{X}$, to scalar values, $c \in \mathcal{C}$: $f : \mathcal{X} \mapsto \mathcal{C}$.

For any given learning algorithm, $a \in A$, and dataset of size m visited in sequence, $\{x_1, \ldots, x_m\}$, we get a series of loss function values from the algorithm: $c_m = \{f(x_1), \ldots, f(x_m)\}$, which is obviously going to be a stochastic variable depending on our initialization and data samples.

The No Free Lunch Theorem

We can then ask, what is the expected value of obtaining a particular set of loss function values for a given algorithm, $a_1 \in \mathcal{A}$, with the expectation over loss functions?

$$E_f[P(c_m|f, m, a_1)] = \sum_f P(f)P(c_m|f, m, a_1)$$

If we have two algorithms we are considering, a_1 and a_2, then obviously, we want to select the algorithm that makes this expected value as low as possible.

The No Free Lunch Theorem

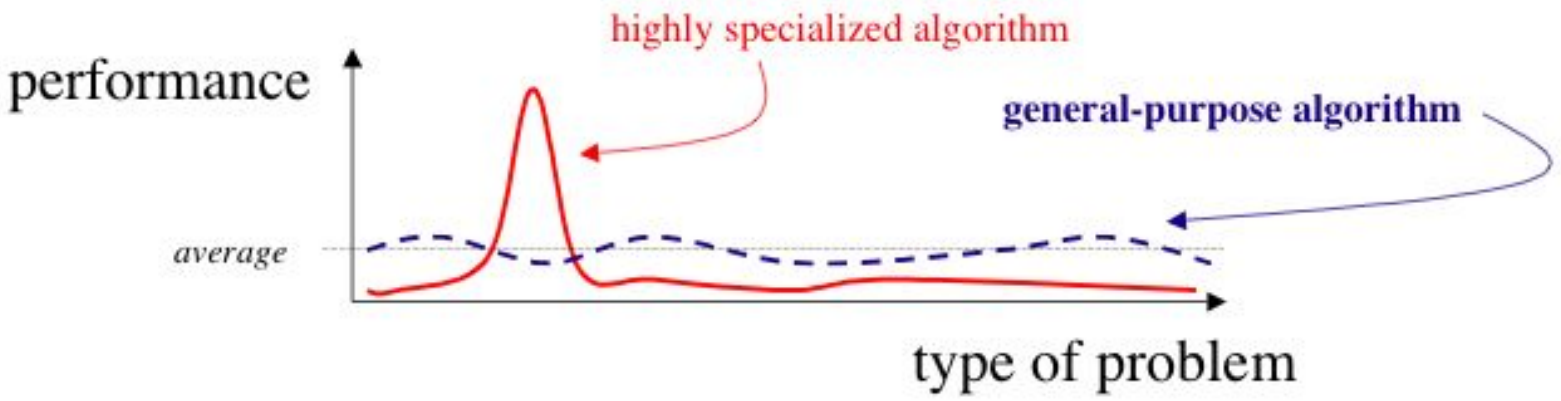
Problem is, Wolpert & MacReady (1997) proved (in an algorithm and loss function agnostic manner) that for any two algorithms:

$$\sum_{f} P(c_m|f, m, a_1) = \sum_{f} P(c_m|f, m, a_2)$$

As such, if we have a flat prior over loss functions, $P(f_1) = P(f_2)$, $\forall f_1, f_2 \in \mathcal{F}$, then no algorithm can perform better than any other algorithm on average!

The take home message

Essentially, this tells us that there can be no general purpose algorithm that outperforms specialized algorithms on all tasks. You have to assume a non-flat prior over loss functions and select an algorithm that is well suited to learning that specific task if you want best performance.



Inductive biases and the AI set

How to pay for our lunch?

If we cannot have a general purpose learning algorithm that outperforms specialist algorithms, how do we select the right algorithms for AI? Bengio & LeCun (2007) identified three typical response categories to this question.

See: Bengio, Y., and LeCun, Y. "Scaling learning algorithms towards AI." Large-scale kernel machines 34, no. 5 (2007): 1-41.

Defeatism

It’s impossible, we’re screwed. We just have to hand-craft specific solutions to every specific task we ever want to accomplish in AI.

Denial

La la la la la... We can't hear you Wolpert & MacReady (1997). We have proofs showing that things like kernel machines can approximate any function, and we have bounds on their ability to generalize with regularization, so it's all good, go away now please.

Optimism

The No Free Lunch Theorem just tells us that we can't have a **completely** general purpose algorithm. But, we don't want to hand craft everything, that will be unusable in most real-world applications. So, maybe we can define the set of things we actually want to do with AI and
design systems that are general purpose within that restricted set.

This requires **inductive biases**.

Inductive biases

Inductive biases are assumptions that we bake into our algorithms about the sort of tasks we will be performing. They are a means of embedding prior knowledge into an optimization system.

Example

The world is organized into objects, which are spatiotemporally constant. We can build this into our learning systems and assume that we will always see consistent movement in sensory space.

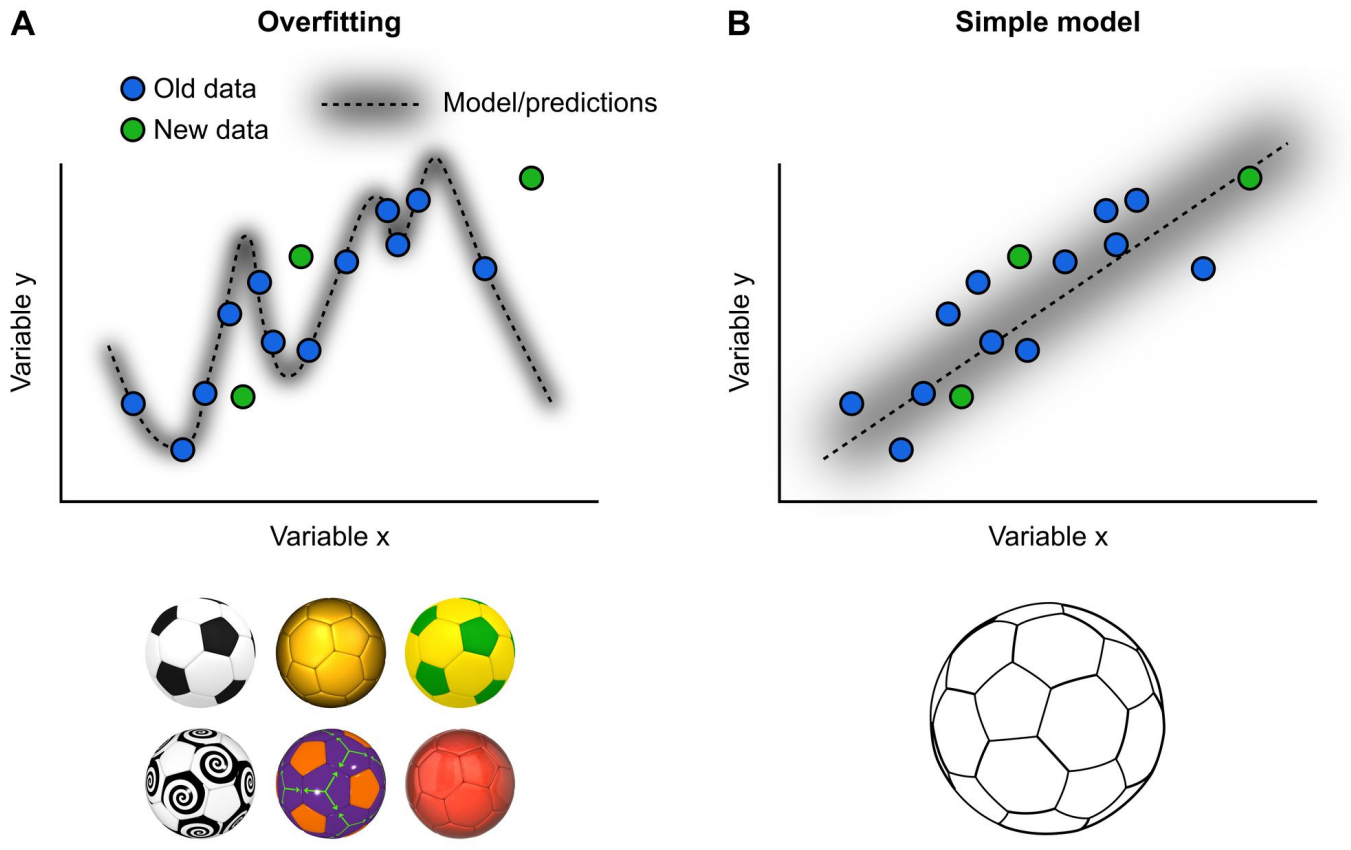
Inductive biases

Broadly, we can build inductive biases into machine learning algorithms using the following three components:

1. Using hand-wired pre-processing of data (e.g. extracting pre-determined features).
2. Using specific architectures for our learning machines.
3. Using specific loss functions and regularizers for our learning machines.

Note: a regularizer is just a way of preventing over-fitting to data.

Over-fitting



The AI Set

Okay, let's say we want to be optimists and identify good inductive biases for AI. What should those be?

First, we need to define the set of problems we're interested in for AI.

The AI Set

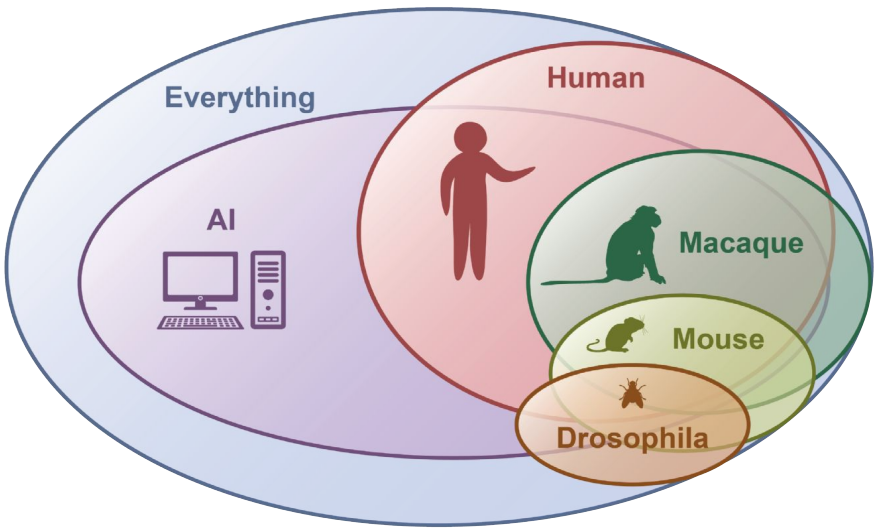
In-line with the original ideas of Turing and others about what AI is all about, Bengio & LeCun (2007) argued that AI should take inspiration from brains!

Our brains seem general purpose in their learning, but they are actually pretty restricted, and learn certain things far more easily than others (e.g. learning to read natural language is easier than learning to read barcodes).

The AI Set

So, maybe AI should be concerned with the broad set of tasks that animals and people are good at, and maybe some related tasks that animals and people aren't good at only due to physical/speed limitations.

They called this the **AI set**.



Refocusing AI

If we accept this logic, then AI research should be about defining good inductive biases that are as minimal as possible in order to do well on the AI set.

Using the three items identified above, we should build learning systems that can do human-like tasks. But, we want to minimize our reliance on hard-wired features, and only embed as much of an inductive bias as is necessary for the AI set.

Interim summary

- The **No Free Lunch Theorem** demonstrated that no general-purpose learning algorithm can outperform specialist algorithms on average.
- We therefore have to build **inductive biases** into our systems that make them good at specific sets of tasks.
- Bengio & Lecun (2007) argued that we don't want to make this too restrictive, and should focus on the **AI set**.
- The AI set is the set of tasks that humans and animals are good at, plus some additional related tasks (and maybe minus a few, like mating, etc.)

Deep architectures as a good inductive bias

Before deep learning...

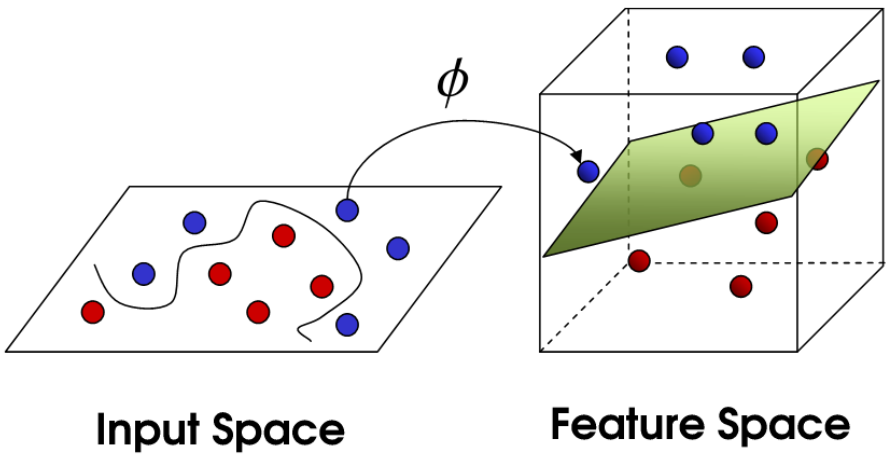
In the early 2000s, the most popular approach in machine learning was a set of approaches known as **kernel machines**.

Say we want to classify some data, x, into two classes. Ideally, we would have a situation where we can deal with various non-linear class boundaries, but still have the nice properties of Perceptrons, namely convex loss functions and learning algorithms that are guaranteed to converge.

Kernel machines are essentially a means of trying to get this combo.

Getting good features

The basic idea: project the data into a high-dimensional space where it becomes linearly separable.



The most obvious way to do this is to try to define a set of pre-processing stages that will accomplish this. But then, we are headed down the defeatist path...

Kernel machines

Kernel machines try to avoid hand-crafting the pre-processing stages by instead using the data itself to construct the pre-processing. We define a **kernel function**, $K(x, x_i)$ with which we pre-process data using the training samples x_i, such that our function used for classification is:

$$f(x) = \sum_{i=1}^{n} \alpha_i K(x, x_i)$$

Kernel machines

If $K(x, y)$ satisfies some specific mathematical properties (see Mercer's condition), then we know that $K(x, y) = \phi(x) \cdot \phi(y)$ for some function ϕ. This can hold even if ϕ is a non-linear function.

Thus, our classification problem becomes linear by applying ϕ to our data points and taking:

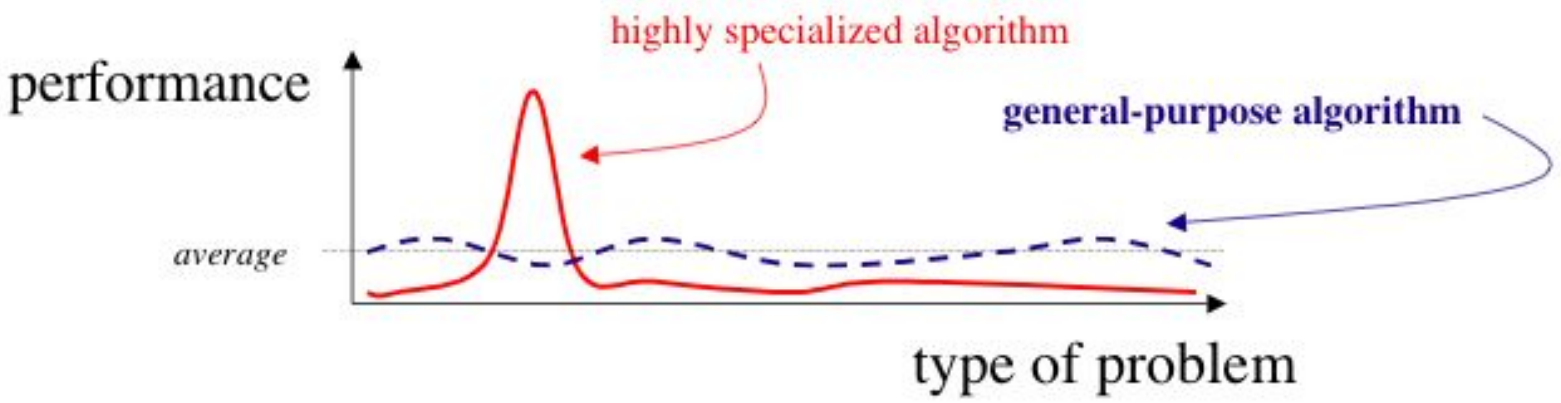
$$f(x) = \sum_{i=1}^{n} \alpha_i \phi(x) \cdot \phi(x_i)$$

So, we will now a get convex optimization on α_i, with guarantees on convergence and generalization, even though we will project our data into a new space using a non-linear function, ϕ.

So what's the problem?

This would seem to be exactly the sort of general-purpose learning device that No Free Lunch tells us we can't have. What gives?

It turns out, kernel machines don't actually perform that well. They are something like our blue line here:



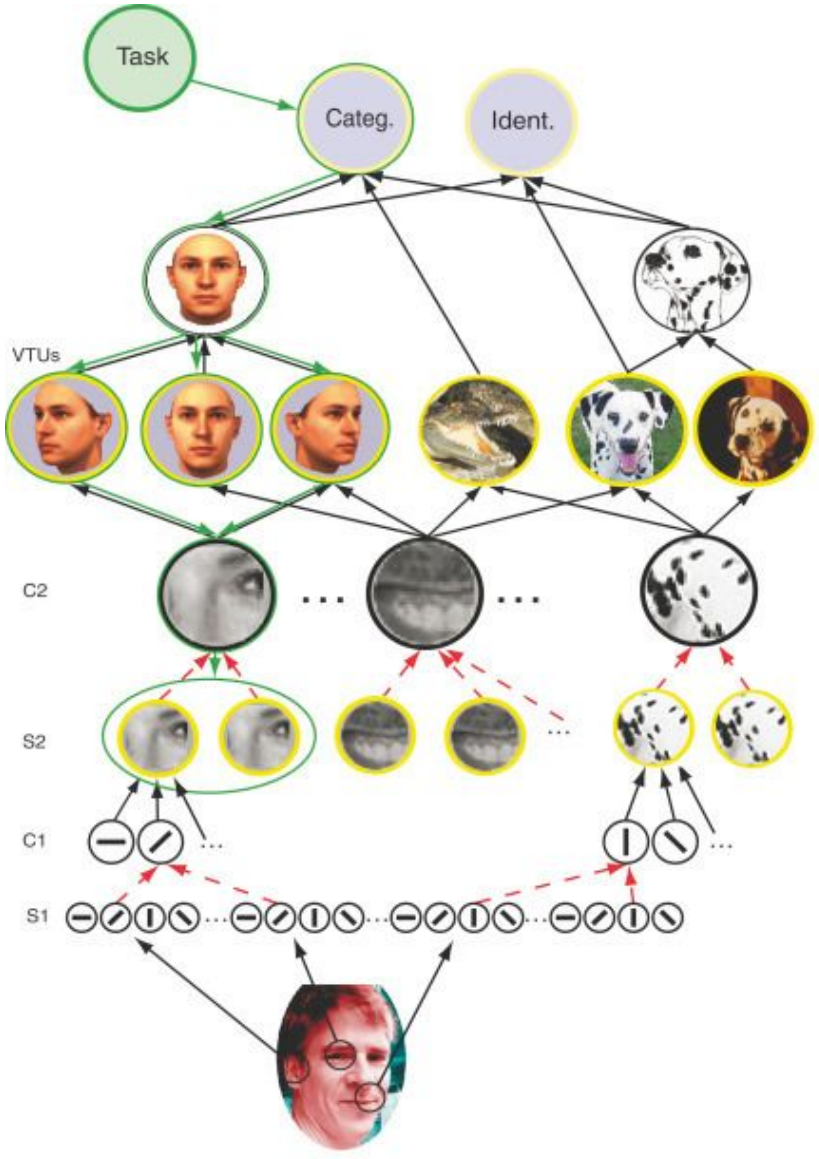
We need more inductive biases

If we want to do better on the AI set, we need to select some more inductive biases then. What would a good inductive bias be for the AI set?

The world is hierarchical

The world around us is *compositional*. This means that when we experience something, it is usually composed of smaller pieces, that are themselves composed of smaller pieces, etc.

The world is hierarchical



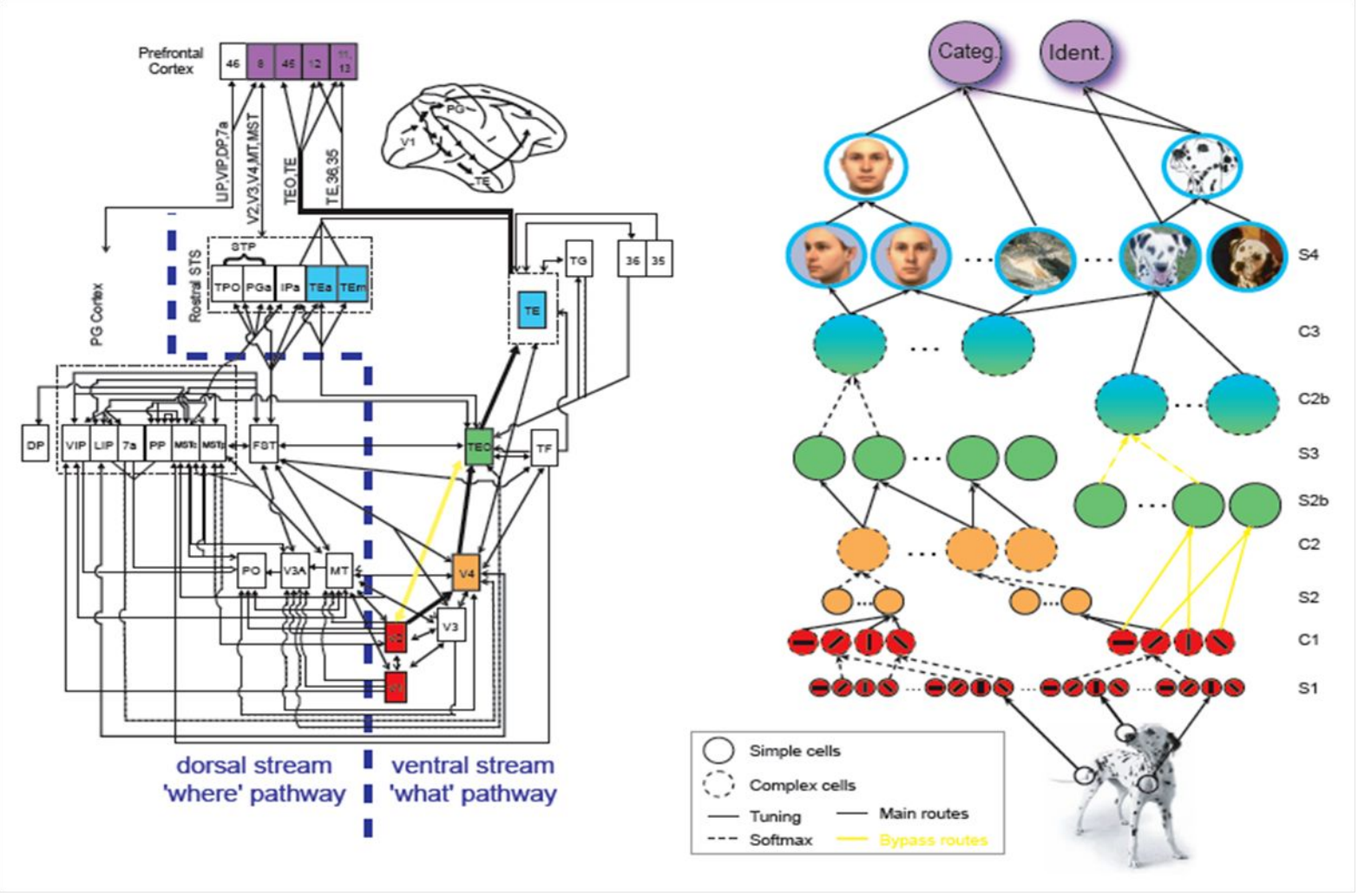
AI should be hierarchical

If the world is best understood in terms of compositions of different pieces, then what we want is not a single non-linear projection into a new space (per kernel machines).

Instead, we want multiple layers of non-linear functions, each one operating with the features identified by the lower layers.

This is an inductive bias. It says, essentially, that we assume tasks where we have to build hierarchies of features.

Brains are hierarchical



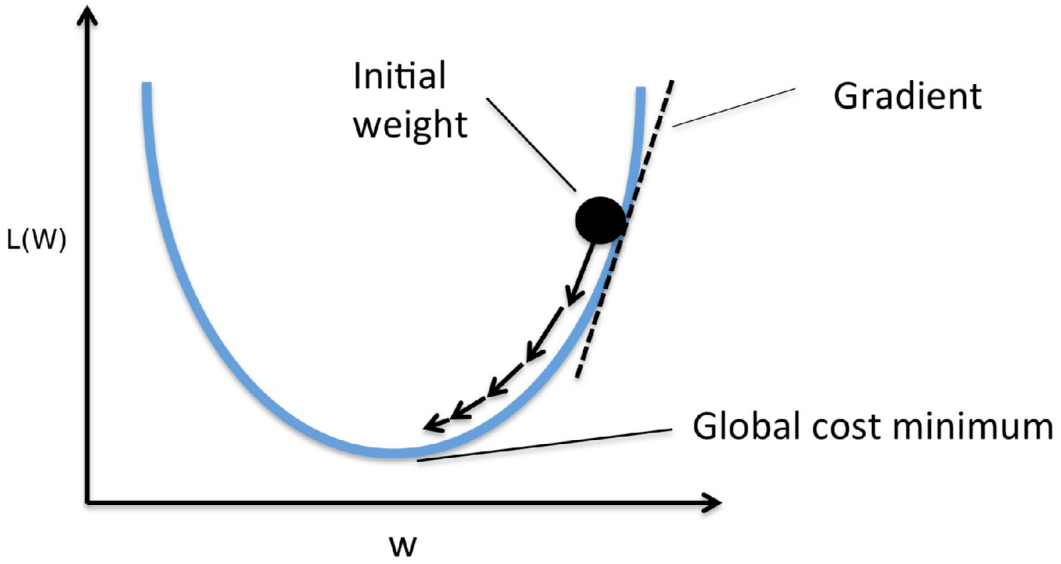
Depth is the goal then...

According to this logic, we need to devise AI architectures that have built-in hierarchy. In other words, we need deep architectures. This line of logic is the origins of “deep learning”.

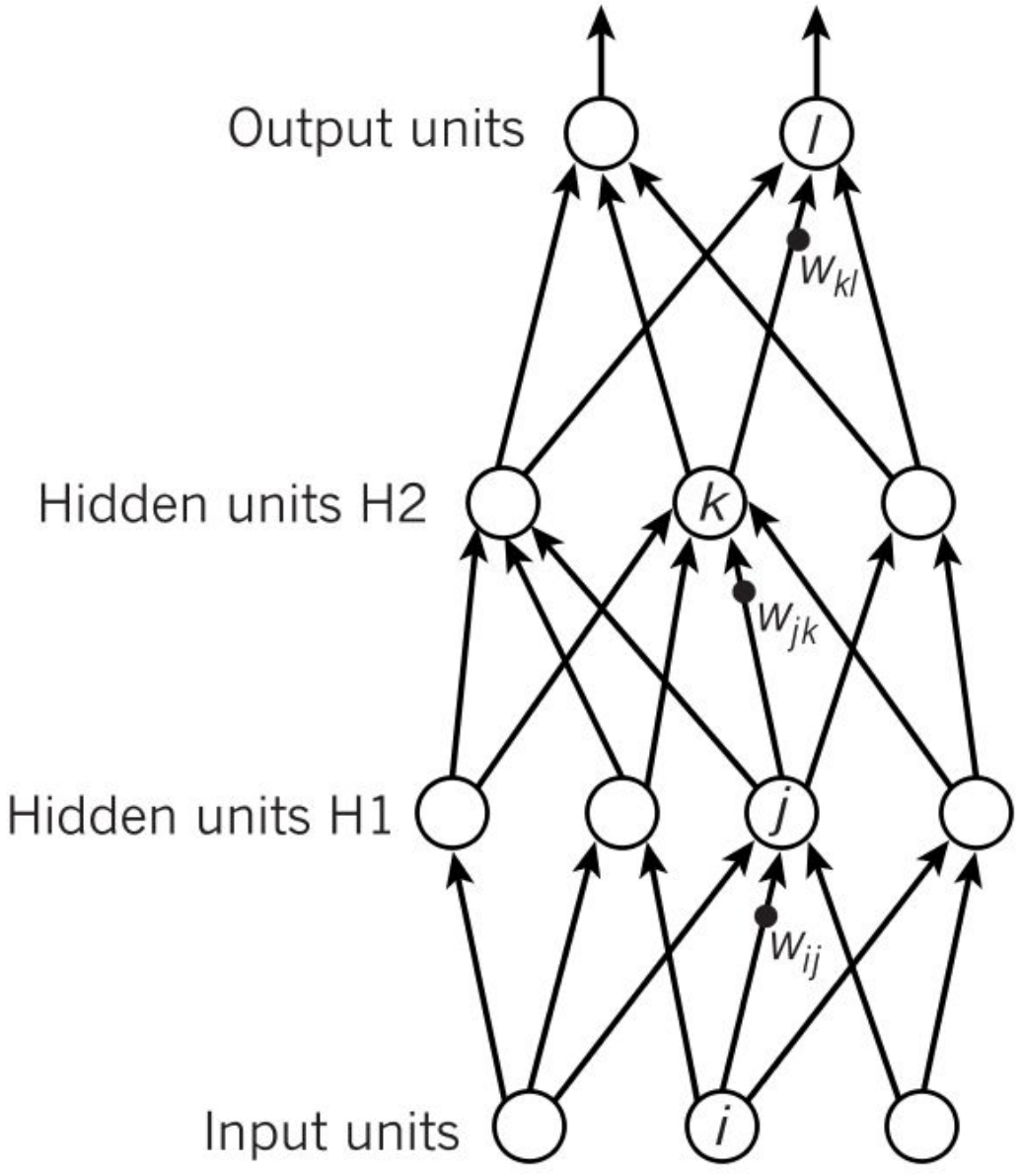
Making learning work in deep architectures

Towards deep learning: gradient descent

You want to learn a hierarchy. How are you going to do it? The most successful approach has been **gradient descent**.



Towards deep learning: gradient descent



$$y_l = f(z_l)$$

$$z_l = \sum_{k \,\varepsilon\, \text{H2}} w_{kl}\, y_k$$

$$y_k = f(z_k)$$

$$z_k = \sum_{j \,\varepsilon\, \text{H1}} w_{jk}\, y_j$$

$$y_j = f(z_j)$$

$$z_j = \sum_{i \,\varepsilon\, \text{Input}} w_{ij}\, x_i$$

Towards deep learning: gradient descent

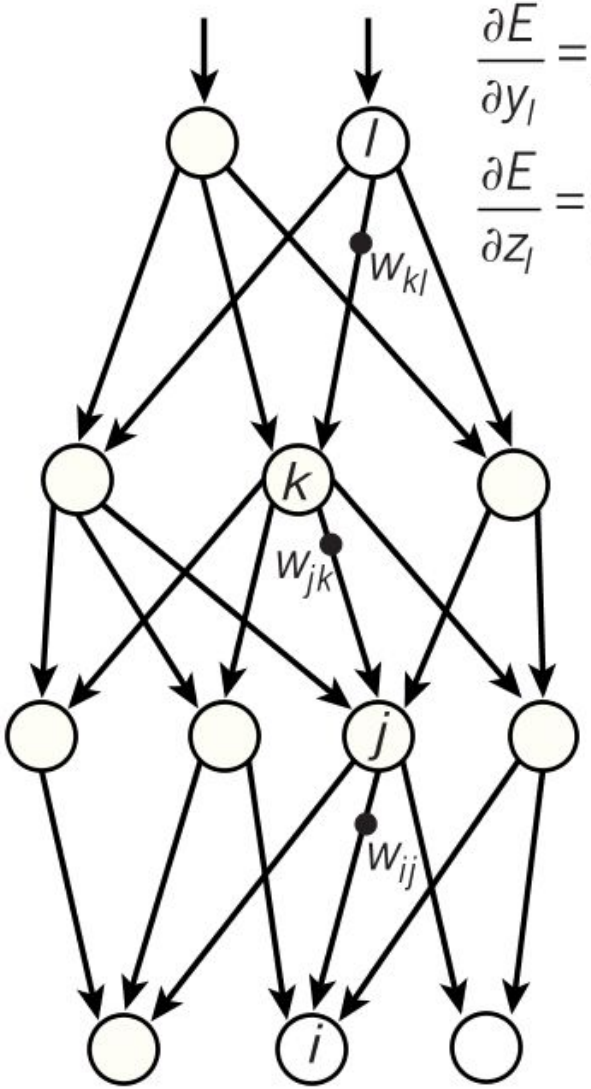
Compare outputs with correct answer to get error derivatives

$$\frac{\partial E}{\partial y_l} = y_l - t_l$$

$$\frac{\partial E}{\partial z_l} = \frac{\partial E}{\partial y_l}\frac{\partial y_l}{\partial z_l}$$

$$\frac{\partial E}{\partial y_k} = \sum_{l \,\varepsilon\, \text{out}} w_{kl}\frac{\partial E}{\partial z_l}$$

$$\frac{\partial E}{\partial z_k} = \frac{\partial E}{\partial y_k}\frac{\partial y_k}{\partial z_k}$$

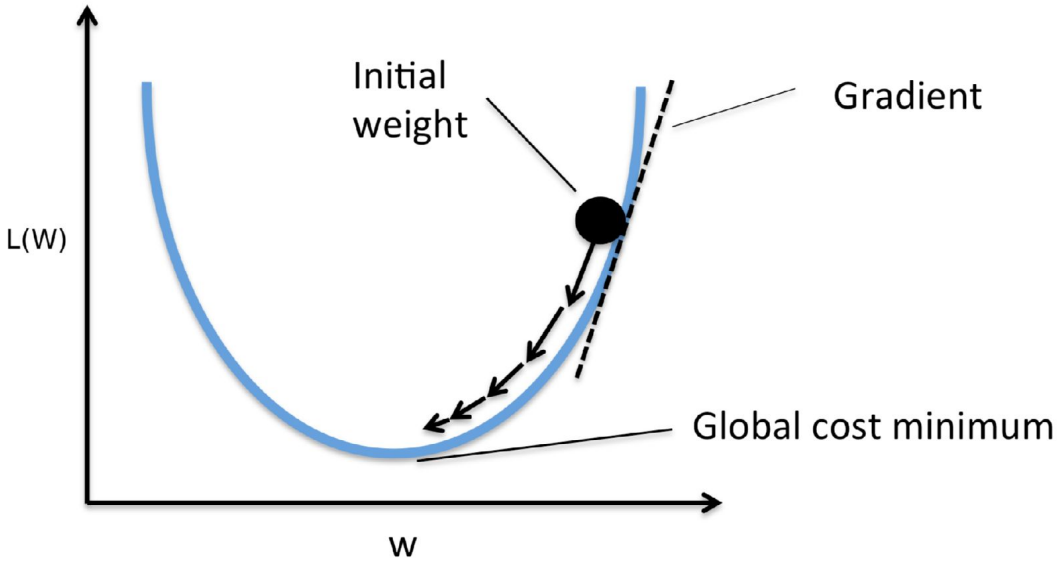


$$\frac{\partial E}{\partial y_j} = \sum_{k \,\varepsilon\, \text{H2}} w_{jk}\frac{\partial E}{\partial z_k}$$

$$\frac{\partial E}{\partial z_j} = \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial z_j}$$

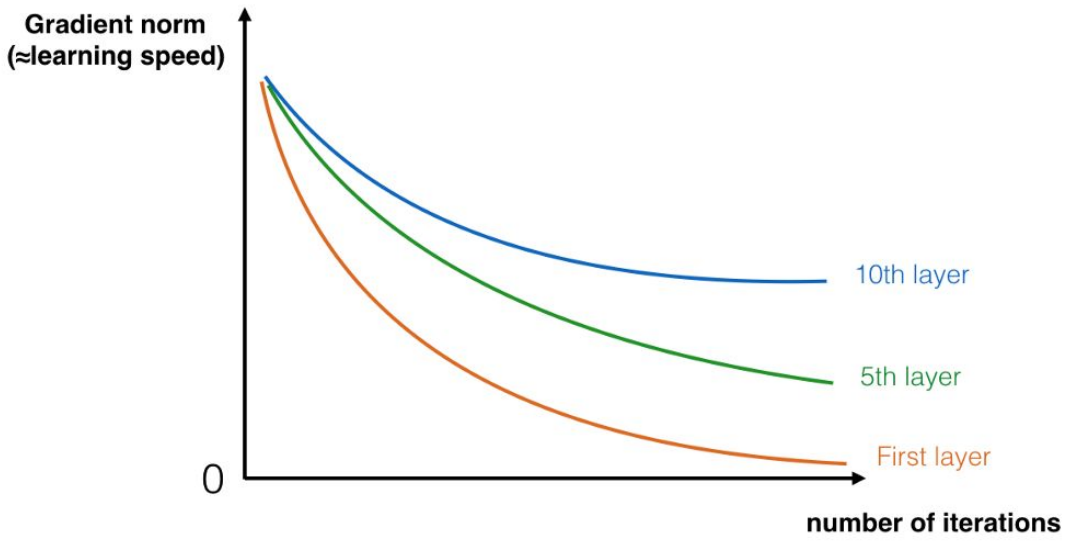
Towards deep learning: gradient descent

But, back in the early 2000s, it was thought that gradient descent would be unworkable for deep nets.



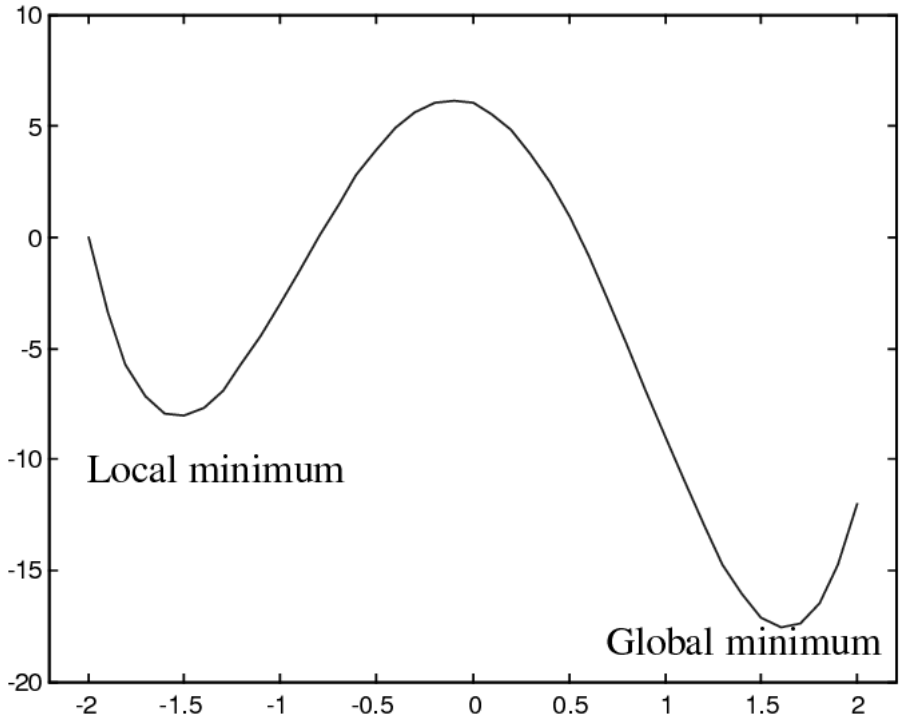
Towards deep learning

First, there is that vanishing gradients problem…



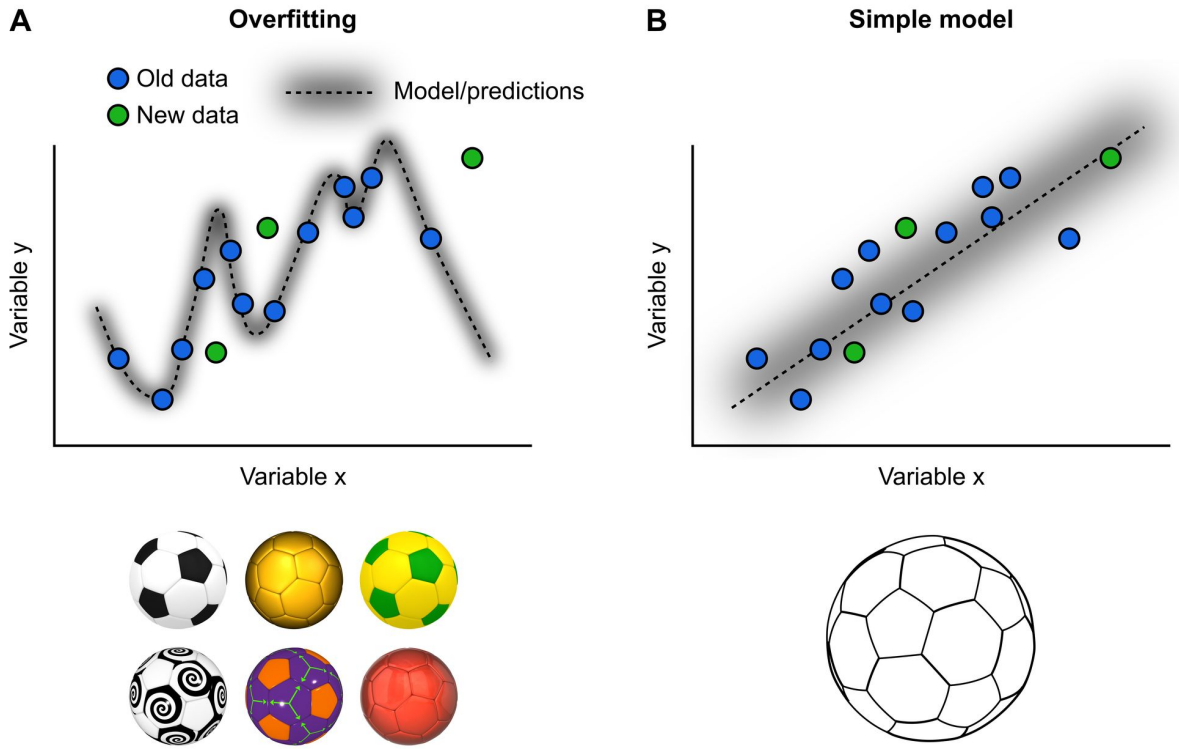
Towards deep learning

Second, there is the local minima problem...



Towards deep learning

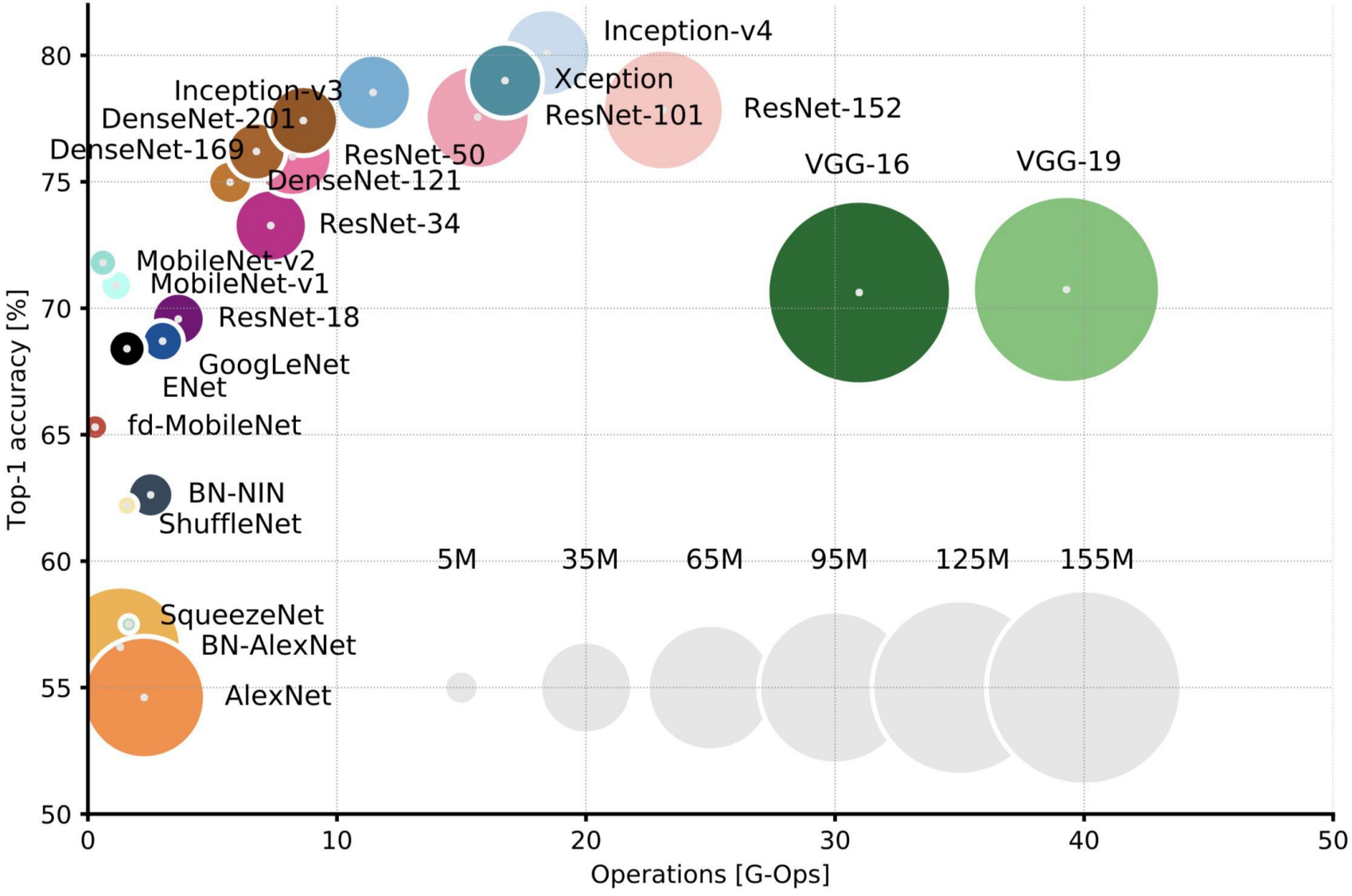
Finally, there's over-fitting...



Deep learning was dead in the water... until...

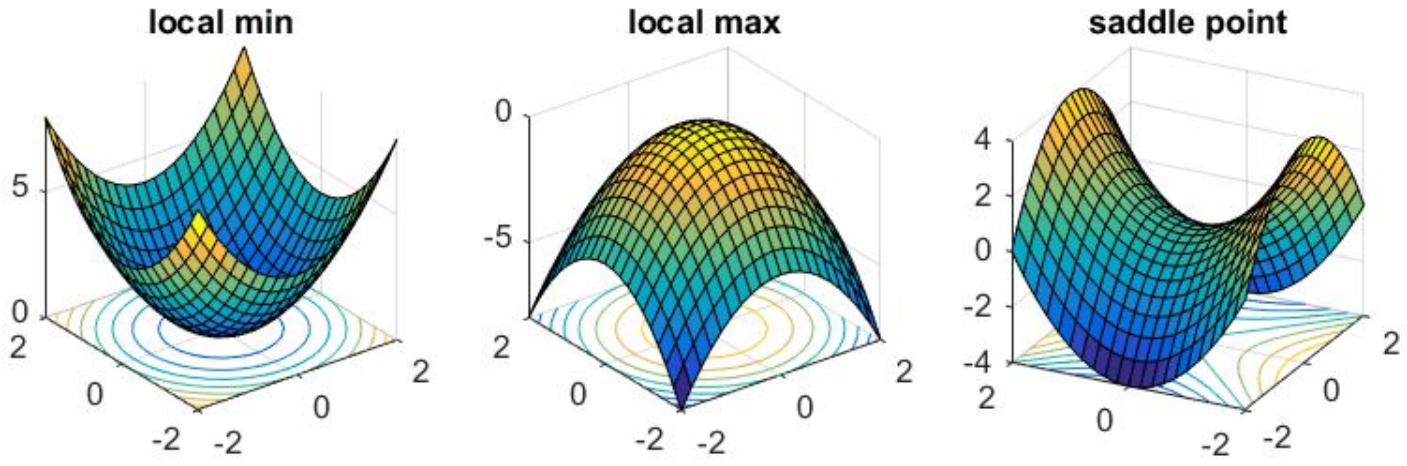
In the late 90s and early 2000s, it was thought that deep architectures were just unworkable due to these issues. But, in the last decade we've realized that in the right circumstances, gradient descent in deep networks can work really well.

Size matters...



The strange double descent phenomenon

Saddle points



Other things that help with deep learning

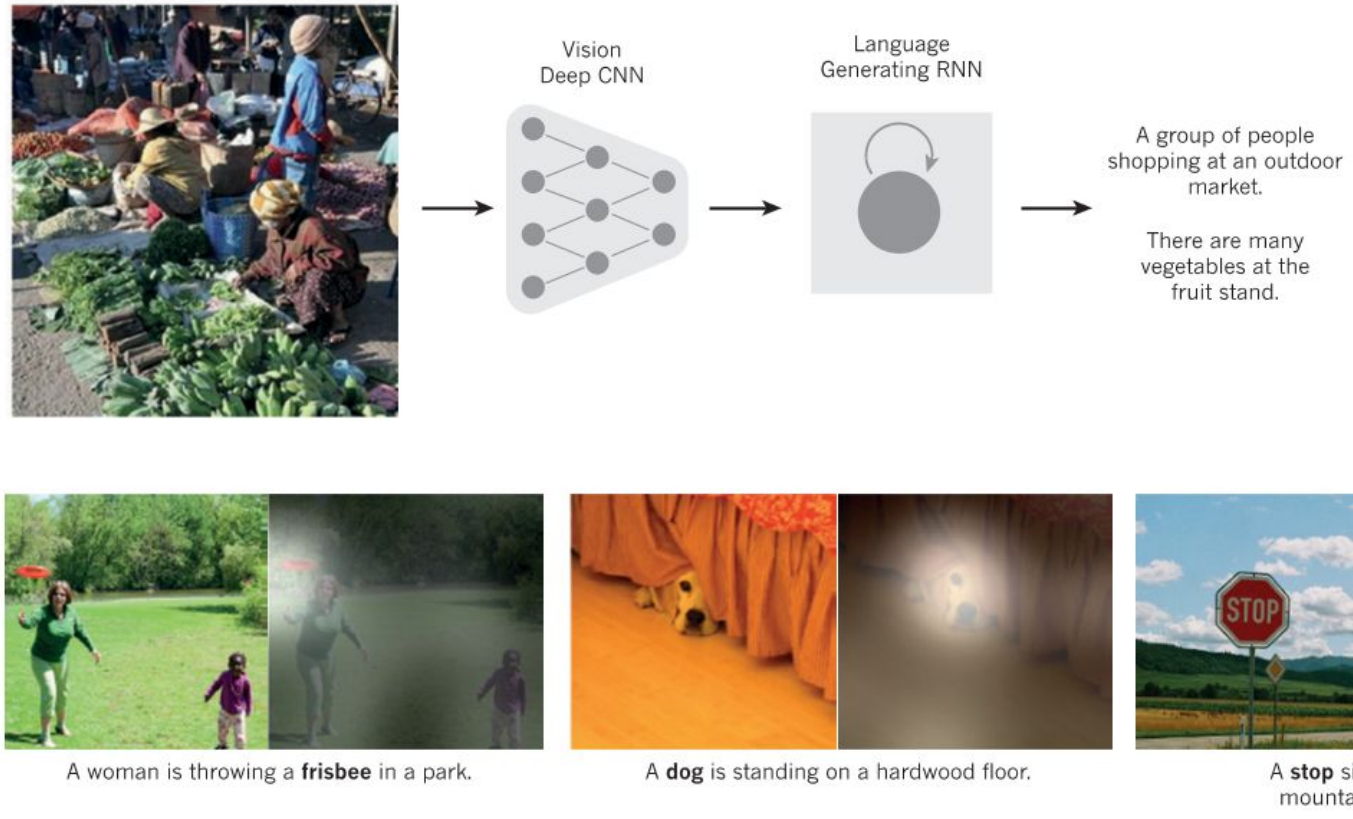
- Modifications to gradient descent to escape saddles (e.g. ADAM)
- Activation functions that mitigate vanishing gradients (e.g. ReLU)
- Good regularization schemes (e.g. Dropout)

Where we stand today...

After decades of people pushing ever deeper architectures, and refining the tools more and more, we can now say that Bengio & LeCun (2007) were basically correct. Deep architectures help with a variety of tasks, most notably the sort of tasks that people are good at.

Perception, communication, motor control... things that require multiple constraint satisfaction in a hierarchical manner. Deep architectures seem to help a lot on such tasks, especially when you have big models and large amounts of data.

Where we stand today...



(Image from LeCun et al., 2015, Nature, 521:436-444)

When to use deep learning

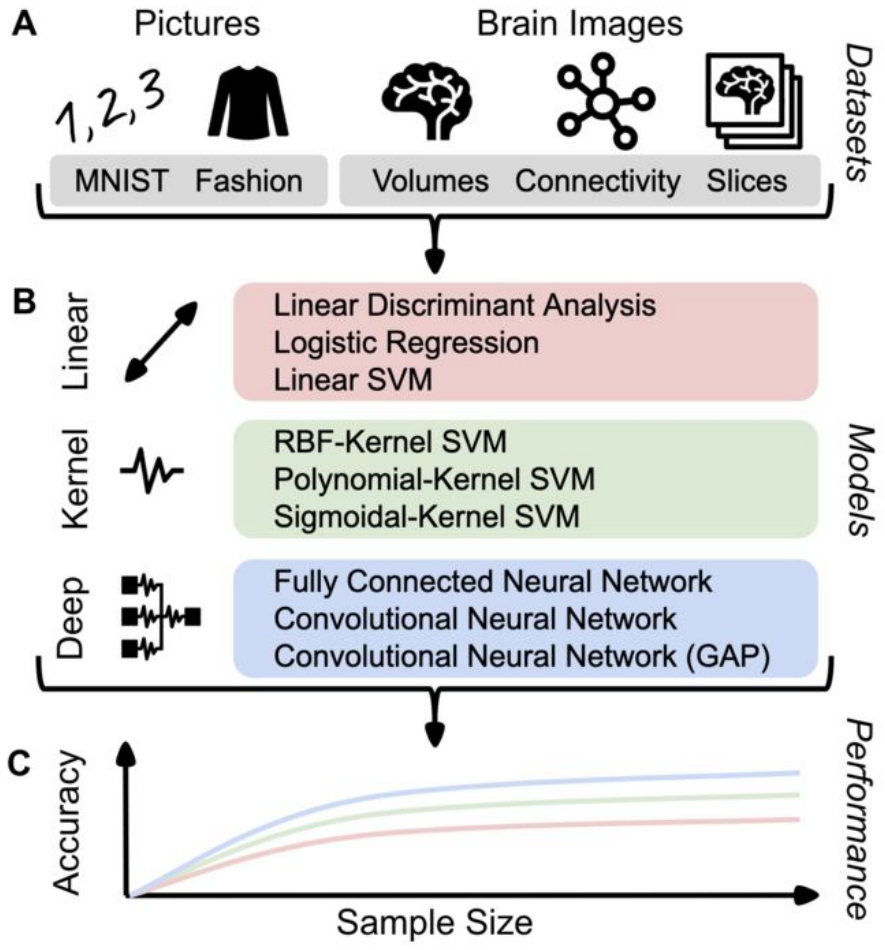
Should you use deep learning?

Should you use deep learning?

Before you try using deep learning in your research, ask yourself:

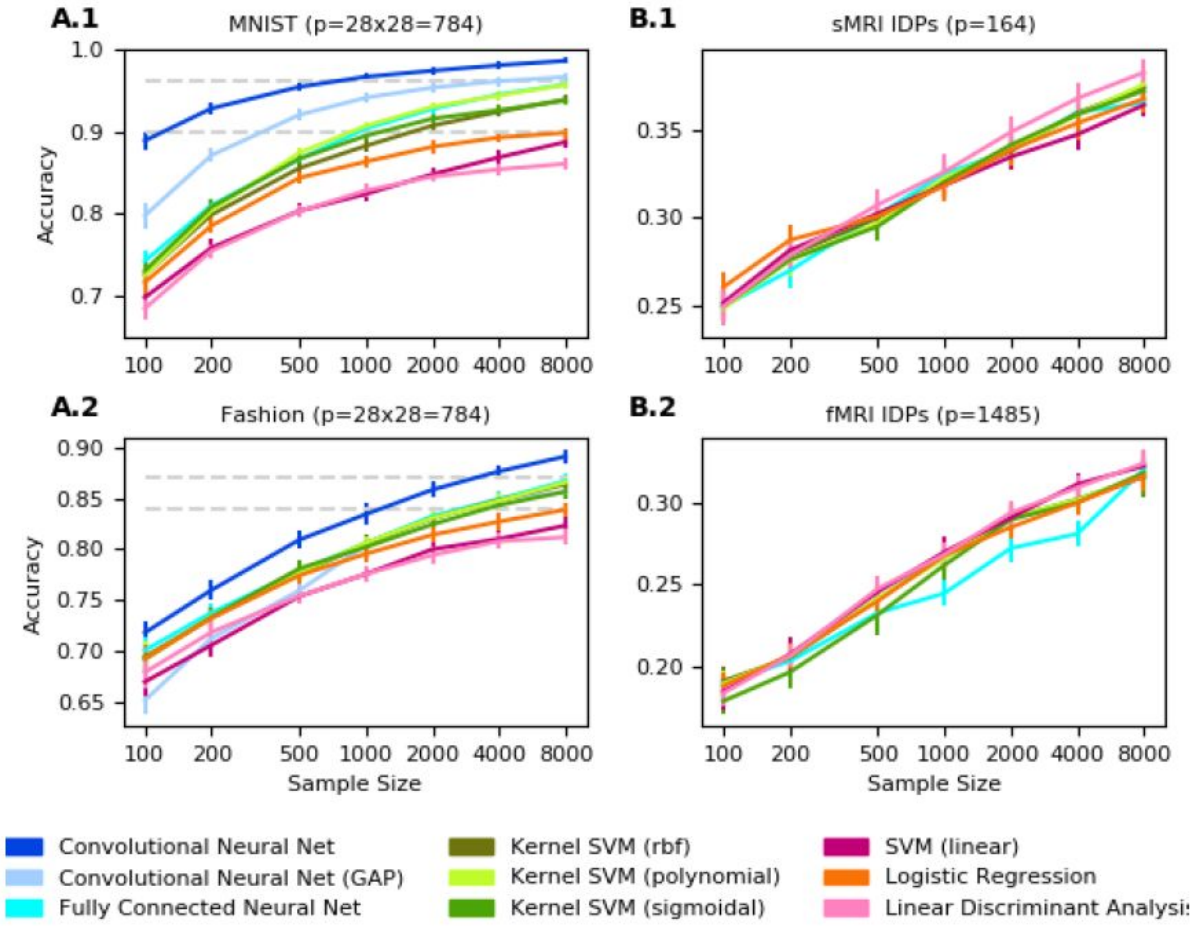
- Is this a task that involves hierarchy?
- What computational resources do I have?
- How much data do I have?
- Are there other datasets I could use to pre-train my model?

Deep learning is not always the right answer



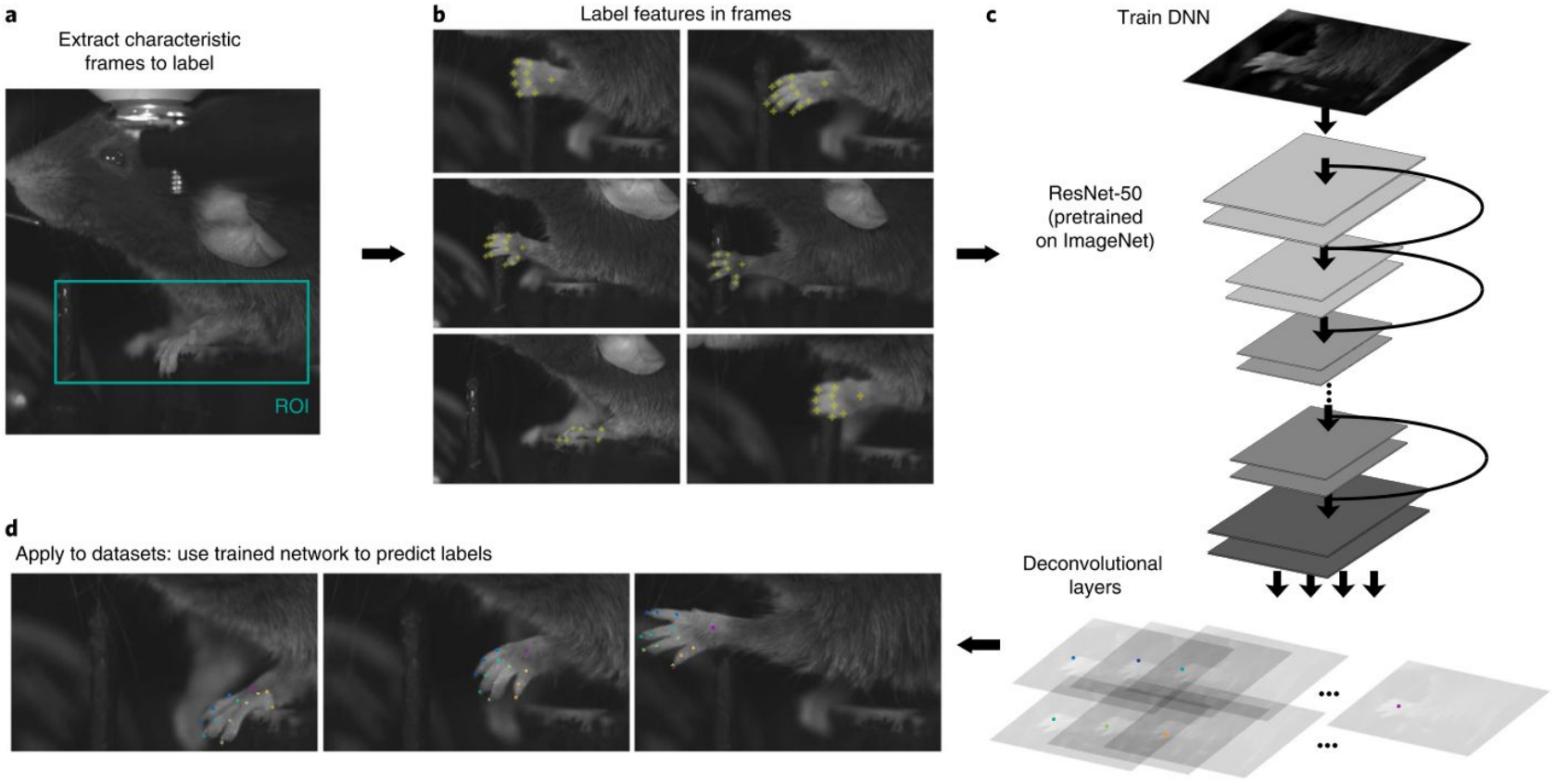
(Image from Schulzet al., 2019, bioRxiv: 757054)

Deep learning is not always the right answer



(Image from Schulzet al., 2019, bioRxiv: 757054)

But sometimes it is!



(Image from Mathis et al., 2018, Nature Neuroscience, 21:1281-1289)

The moral of the story

When to use deep learning:

- Deep learning is a very powerful tool, **but it is designed to operate on the AI Set**, i.e. the set of tasks that humans/animals are good at.
- In particular, deep learning uses an **inductive bias of hierarchy**, which is sometimes really useful, and sometimes not at all.
- Deep learning is particularly effective when you have a huge model and a lot of data (or you can pre-train a huge model on a related dataset).

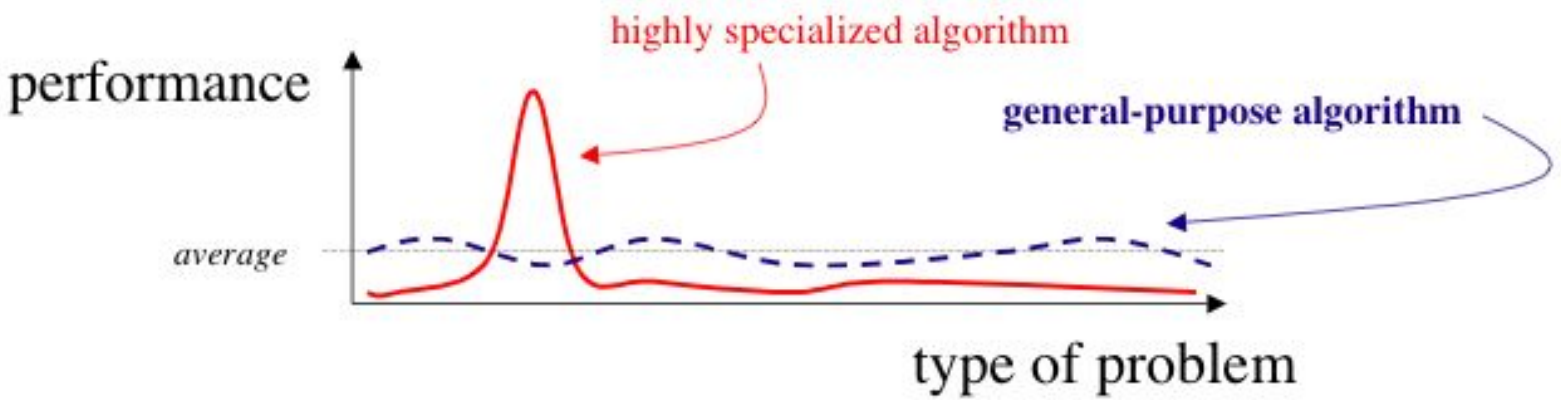
The moral of the story

When **not** to use deep learning:

- If you don't have reason to believe that the problem contains a hierarchical solution, deep learning is not the right answer.
- If you have limited computational resources, deep learning is not the right answer.
- If you have a small amount of data and no related dataset on which to pretrain, deep learning is not the right answer.

The moral of the story

If you're really unsure about the nature of your problem, and you have limited data, your best bet is to use other techniques (linear models, kernel methods, etc.), because it's better to stay general purpose when you don't really know the lay of the land.



Summary

- Deep learning is a “Goldilocks” approach to machine learning which seeks to minimize the use of hand-tuned features but adopt inductive biases suited to the AI Set.
- Deep learning works really well on problems with hierarchical structure and tons of data (especially if you can build really big networks too).
- Deep learning is not a magic technique for analyzing data, it has specific applications that for which is is great.