Machine Learning
– an introduction –

Summer School on Emergent Phenomena in Quantum Materials
Cornell, May 2017

Simon Trebst
University of Cologne

Machine learning

In computer science, machine learning is concerned with algorithms that allow for **data analytics**, most prominently **dimensional reduction** and **feature extraction**.

Many computer programs/apps have machine learning algorithms built-in already.

spam filters

face recognition

voice recognition

Machine learning

Applications of machine learning techniques are booming and poised to enter our daily lives.

digital assistants

self-driving cars

Machines at play

Machine learning techniques can make **computers play**.

A computer at play is probably one of the most striking realization of **artificial intelligence**.

1996: G. Kasparow vs. IBM's deep blue

2016: L. Sedol vs. Google's AlphaGo

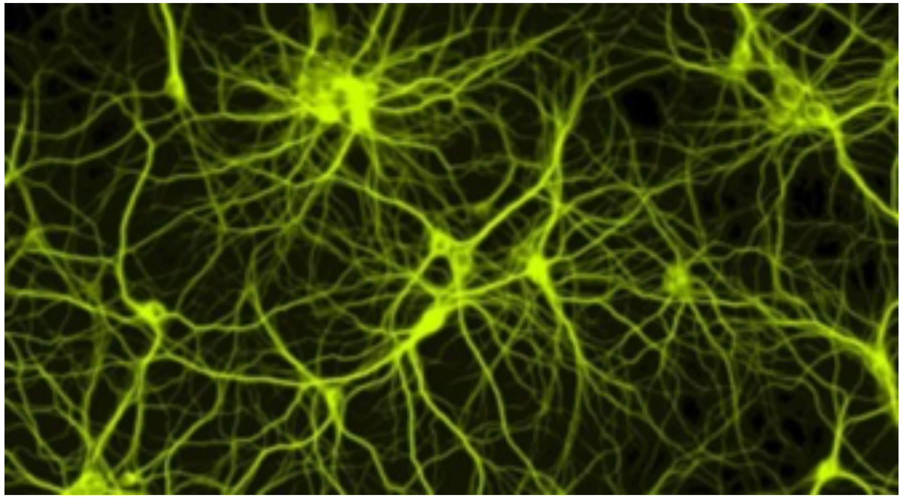
How do machines learn?

How do machines learn?

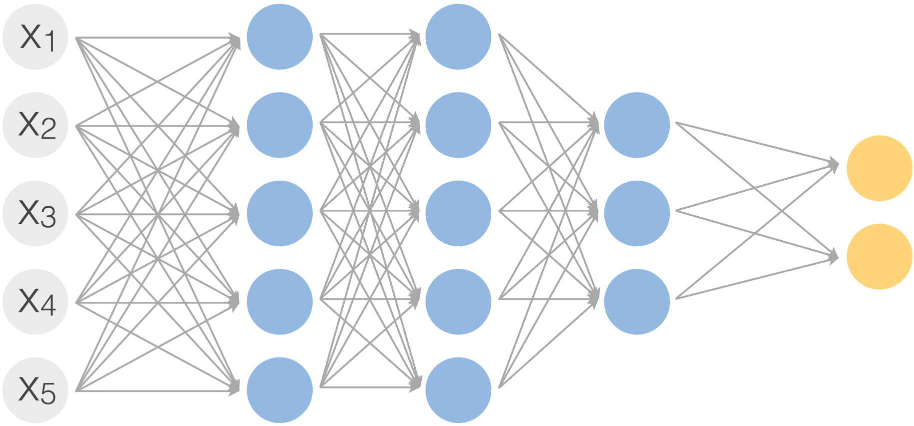
What is it that they can learn?

Can we **control** what they learn?

How can we **benefit** from machine learning?



biological neural network

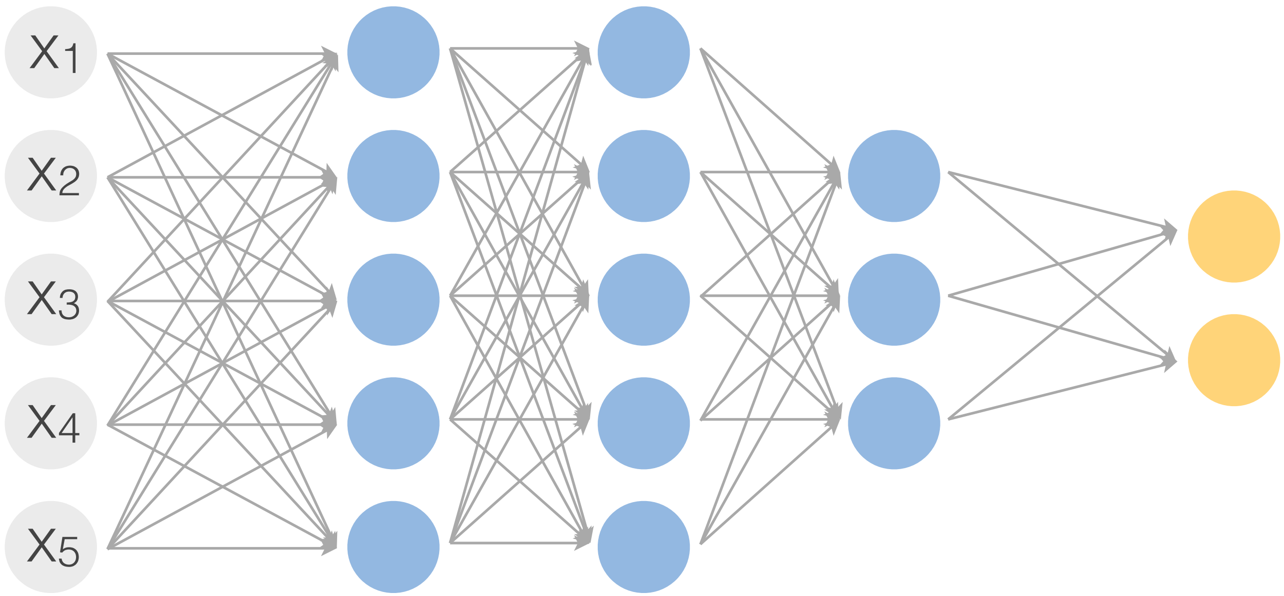


artificial neural network

Artificial neural networks and deep learning

Recommended introduction: http://neuralnetworksanddeeplearning.com by Michael Nielsen

artificial neural networks

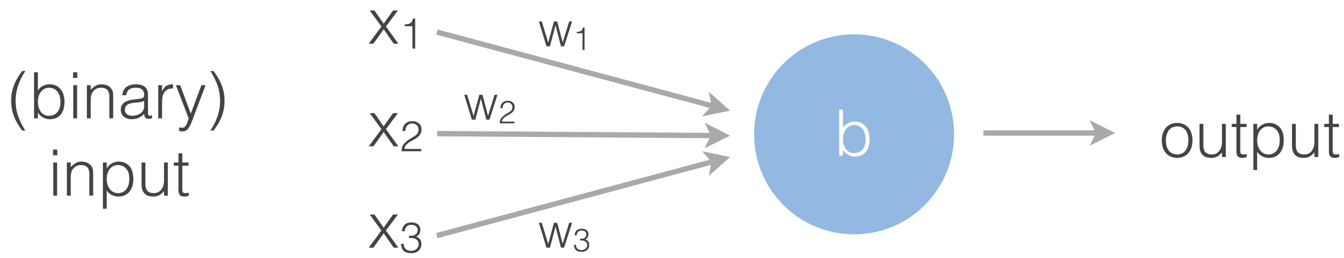


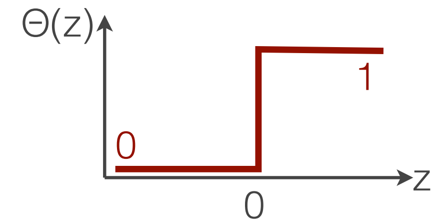
Artificial neural networks **mimic biological neural networks** (albeit at a much smaller scale).

They allow for an **implicit knowledge representation**, which is infused in **supervised** or **unsupervised learning** settings.

artificial neural networks

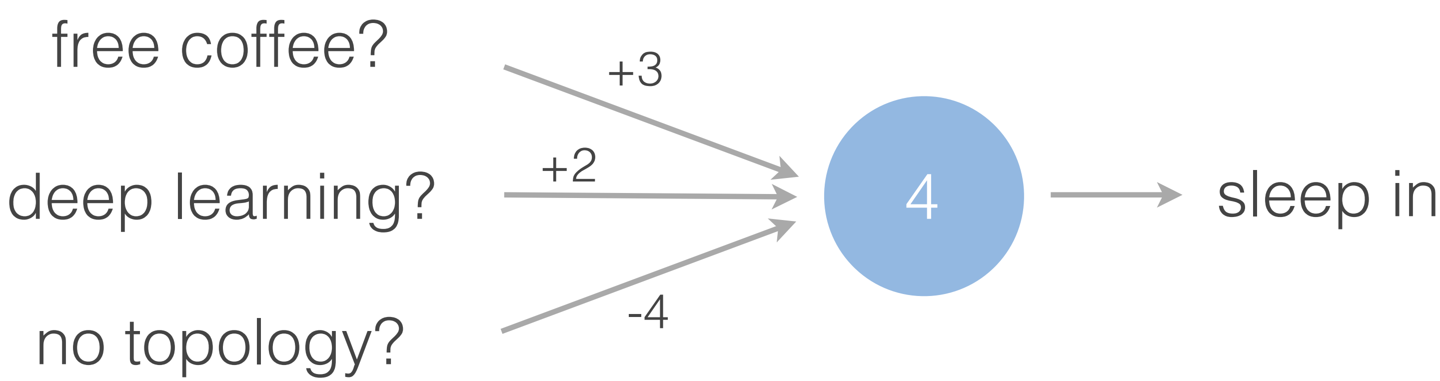
artificial neurons





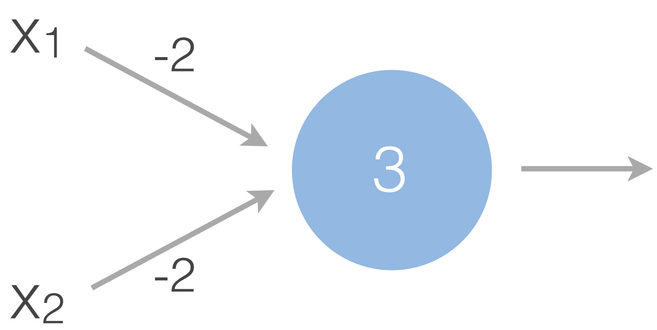
$$\Theta\left(\vec{w}\cdot\vec{x}-b\right)$$

example – Should I skip the first talk?



artificial neural networks

Artificial neural networks are pretty powerful.



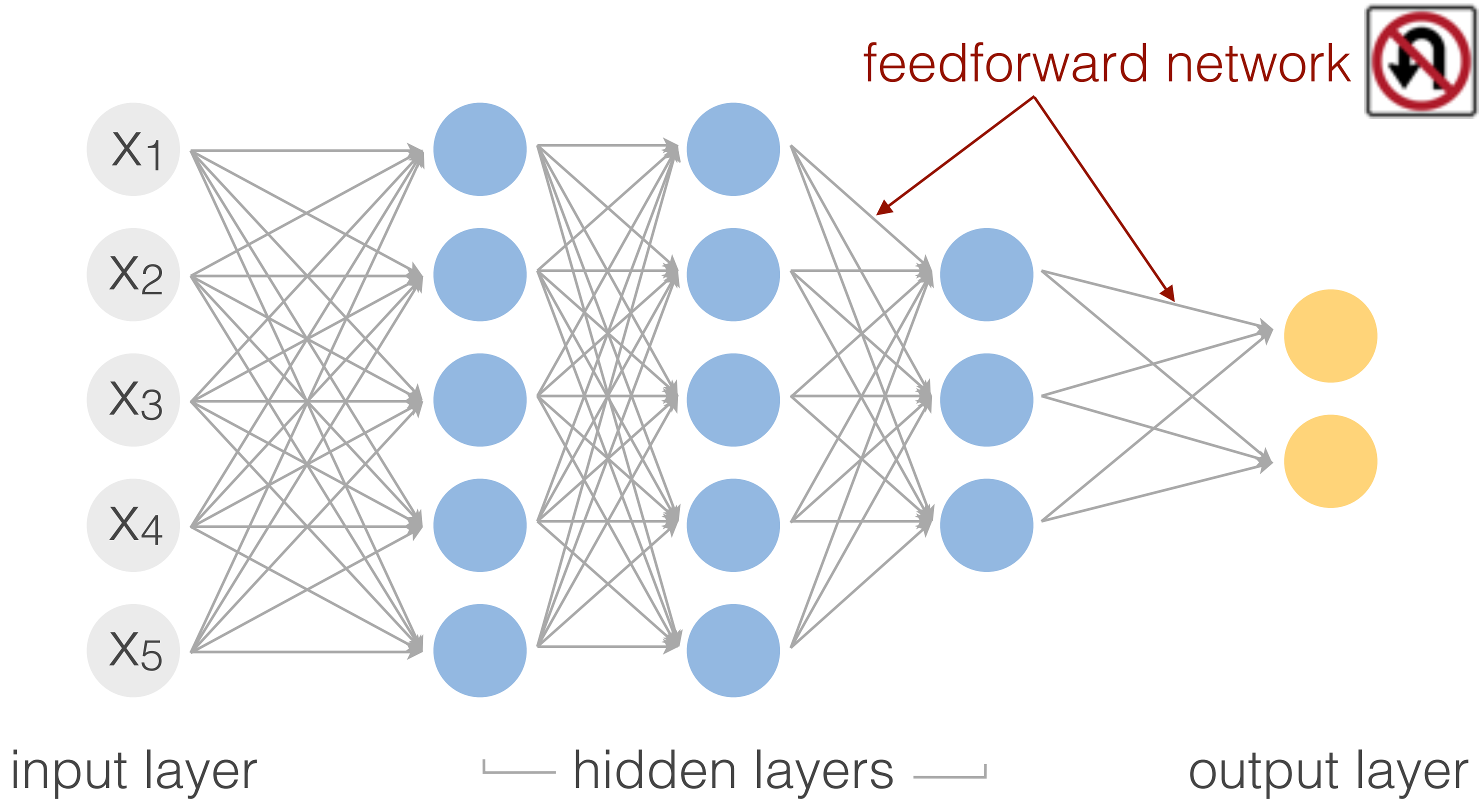
x_1	x_2	output
0	0	1
0	1	1
1	0	1
1	1	0

NAND-gate

Like circuits of NAND gates artificial neural networks can encode arbitrarily complex logic functions, thus allowing for **universal computation**.

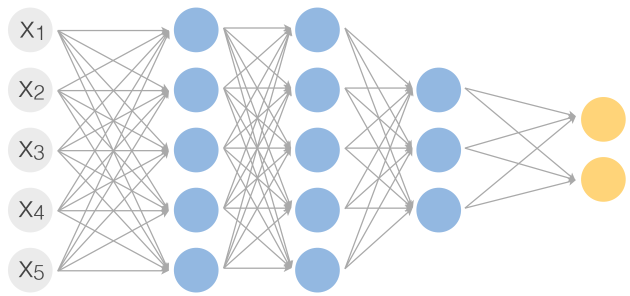
But the power of neural networks really comes about by **varying the weights** such that one obtains some desired **functionality**.

neural network architectures



Neural networks with **multiple hidden layers**
have been popularized as **“deep learning”** networks.

How to train a neural network?

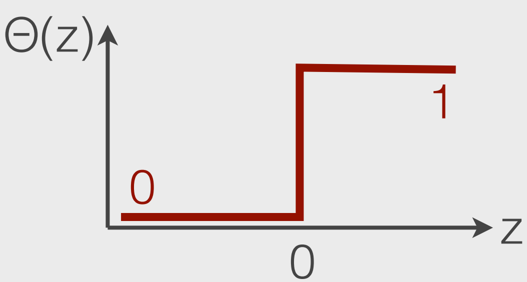


- quadratic **cost function**

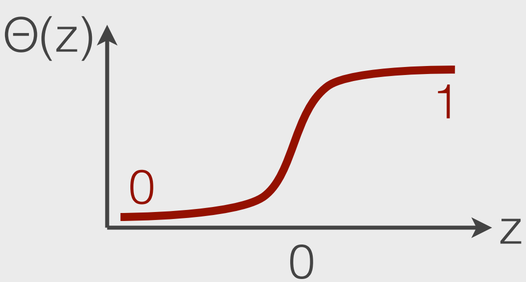
$$C(\vec{w},\vec{b}) = \frac{1}{2n}\sum_x ||\underbrace{y(x)}_{\text{desired output}} - \underbrace{a(x)}_{\text{actual output}}||^2$$

Small adjustments on the level of a single neuron should result in small changes of the cost function.

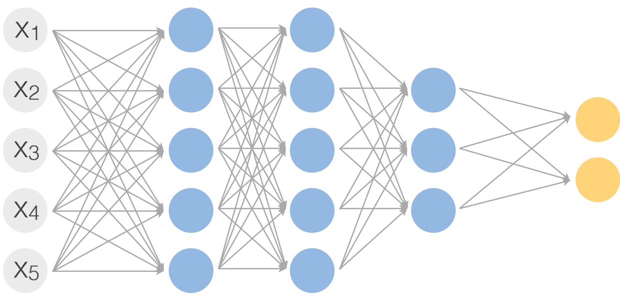
perceptrons



sigmoid neurons



How to train a neural network?

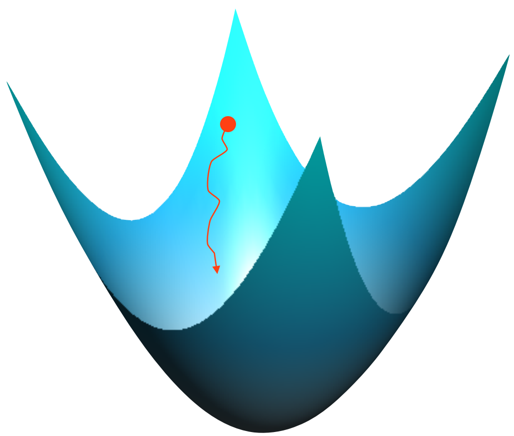


- quadratic **cost function**

$$C(\vec{w},\vec{b}) = \frac{1}{2n}\sum_x ||y(x) - a(x)||^2$$

desired output

actual output



gradient descent

- **back propagation** algorithm

Rumelhart, Hinton & Williams, Nature (1986)

extremely efficient way to calculate *all* partial derivatives

$$\frac{\partial C}{\partial w} \qquad \frac{\partial C}{\partial b}$$

needed for a **gradient descent** optimization.

Three flavors of machine learning

* thanks to Giuseppe Carleo (ETH Zurich) for some of the slides

Supervised Learning

- training with **labeled data**

$$\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$$

data point

expected label

$$C(\mathbf{w}, \mathbf{b}) = \frac{1}{2n} \sum_i C[y_i, F(x_i, \mathbf{w}, \mathbf{b})]$$

cost function

output
neural network

- stochastic **gradient descent**

learning
rate

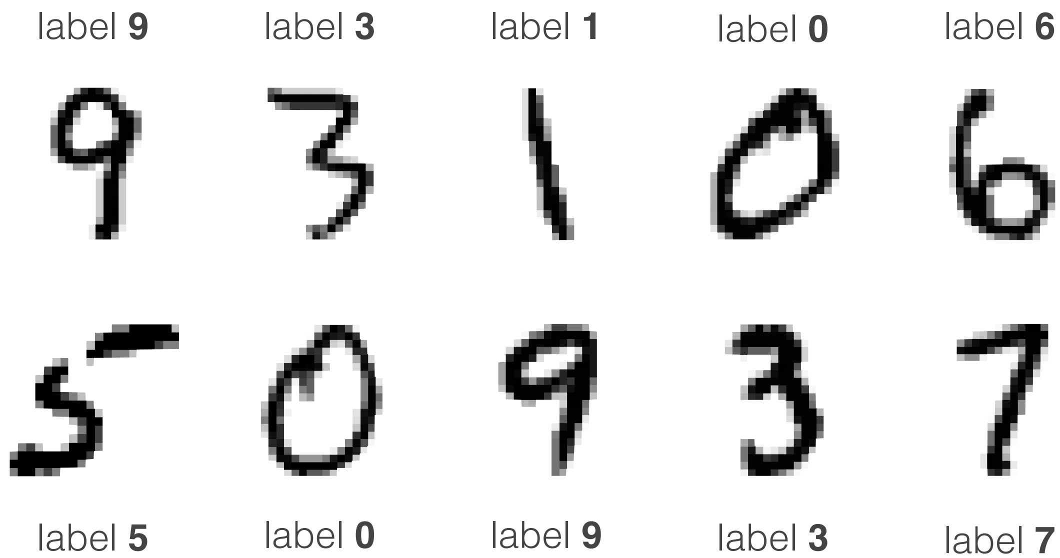
back propagation

$$(\mathbf{w}, \mathbf{b})' = (\mathbf{w}, \mathbf{b}) - \eta \cdot \nabla C[y_i, F(x_i, \mathbf{w}, \mathbf{b})]$$

noisy approximation
of the true gradient

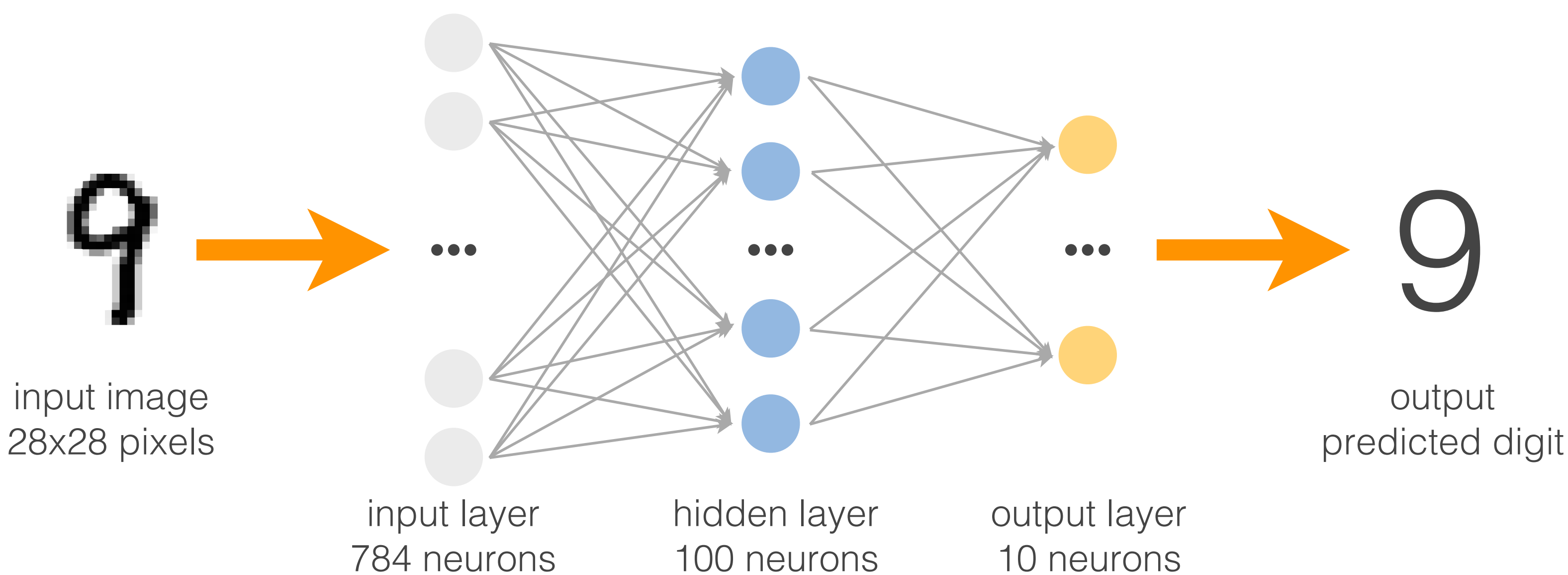
converges to global minimum
(~Langevin dynamics)

Example: digit recognition



labeled data for training

Some **60 lines of code** (Python/Julia) will do this for you with **>95% accuracy**.

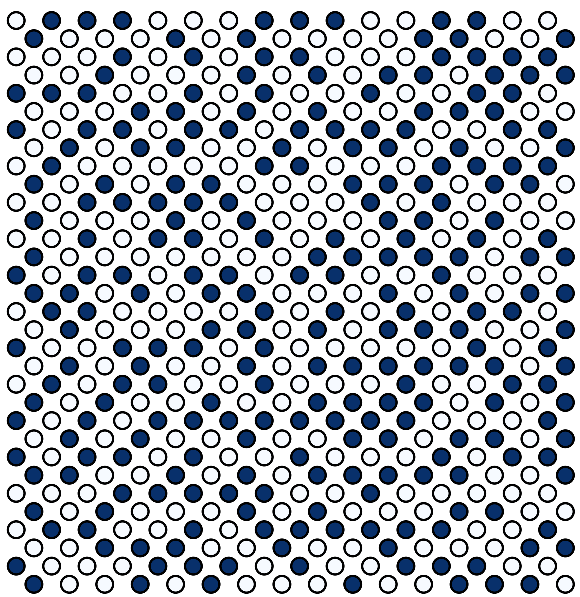


example of neural network architecture

Physics: phase classification

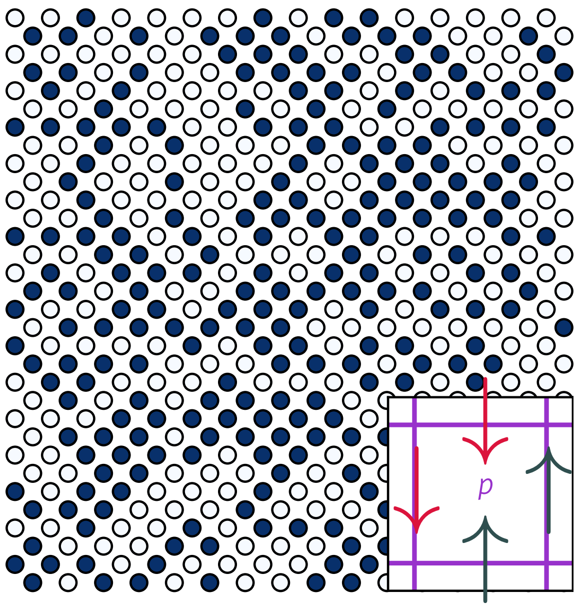
Carrasquilla and Melko, Nat. Phys. (2017)

high-temperature Ising model



$$\mathcal{H} = -J \sum_{\langle i,j \rangle} \sigma_i^z \sigma_j^z$$

low-temperature Ising gauge



$$\mathcal{H} = -J \sum_{p} \prod_{i \in p} \sigma_i^z$$

After training, 100% of configurations shown are correctly identified.
Try to do it by eye instead ...

Unsupervised Learning

- training with **unlabeled data**

$$\{x_1, x_2, \ldots, x_N\}$$

data point

uniformly drawn from (unknown) $P(x)$

$$F(x, \mathbf{w}, \mathbf{b}) \simeq P(x)$$

Goal: Find an approximation for the data distribution (to find correlations etc.)

- typical **cost function**

$$D_{\mathrm{KL}}(P||F) = \sum_i P(x_i) \log \frac{P(x_i)}{\bar{F}(x_i)}$$

Kullback-Leibler divergence

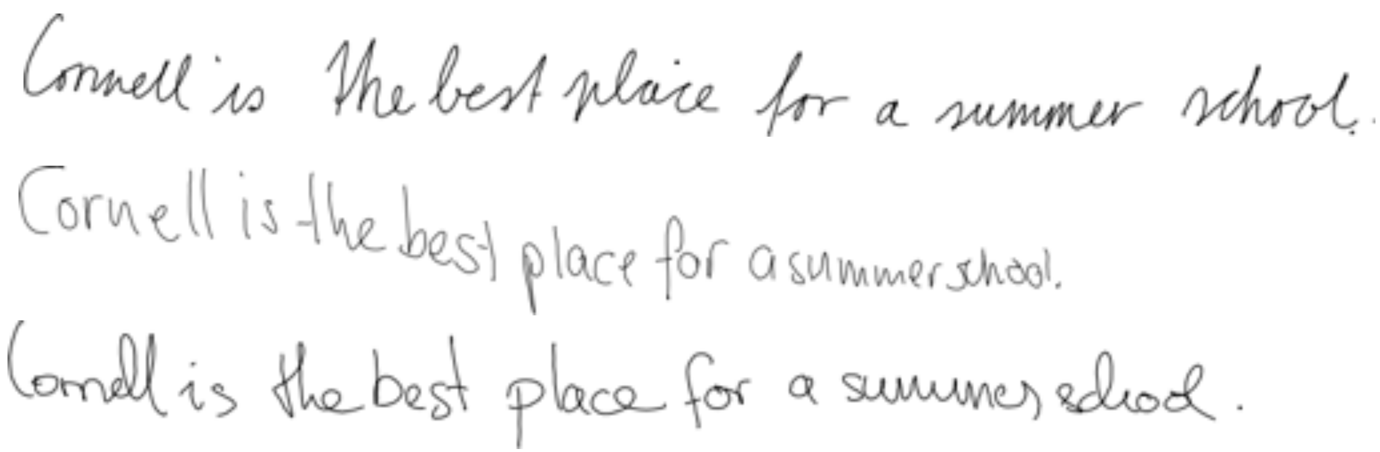
normalized probability (intractable)

$$\nabla D_{\mathrm{KL}}(P||F) = \langle G(x) \rangle_P - \langle G(x) \rangle_F$$

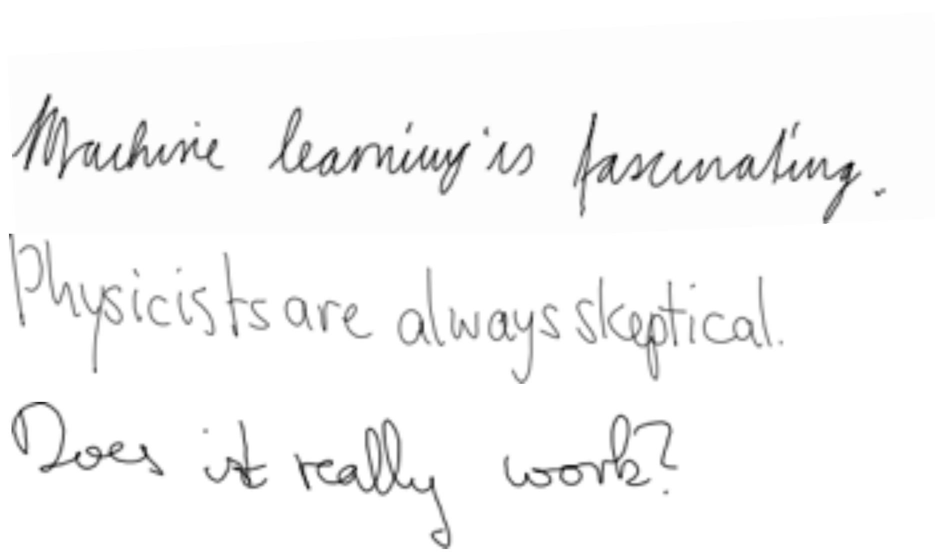
gradient is difference between two expectation values (tractable with sampling, no need to know P)

Example: forging hand writing

http://www.cs.toronto.edu/~graves/handwriting.cgi



unsupervised learning on different hand-writing styles



arbitrary sentences using different hand-writing styles

Physics: improve Monte Carlo moves

Huang, and Wang, PRB 95, 035105 (2017)
Liu, Qi, and Fu, PRB 95, 041101 (2017)

- unsupervised training

$$P(x)$$

We want to sample efficiently from this probability distribution.

$$F(x, \mathbf{w}, \mathbf{b}) \simeq P(x)$$

We can learn *P*, and perform standard cluster updates on *F*.

- transition probabilities

Use samples from machine as proposed configurations

$$A(x \to x') = \min\left(1, \frac{P(x')}{P(x)} \cdot \frac{F(x)}{F(x')}\right)$$

For perfectly learned *F*=*P*, one *always* accepts move.

Reinforcement Learning

- generate data, obtain feedback, come up with strategy

$$S[F]$$

"Scoring function" is a functional of the network

Network generates/harvests data by some "strategy"

$$\min_F S[F]$$

The best "strategy" obtains the best score

- learning

Produce meaningful input/output with *F*

Feedback from *S* (reinforcement stimulus)

Adapt the network accordingly

Example: game playing

Silver et al., Nature 529, 484 (2016)
(AlphaGo)

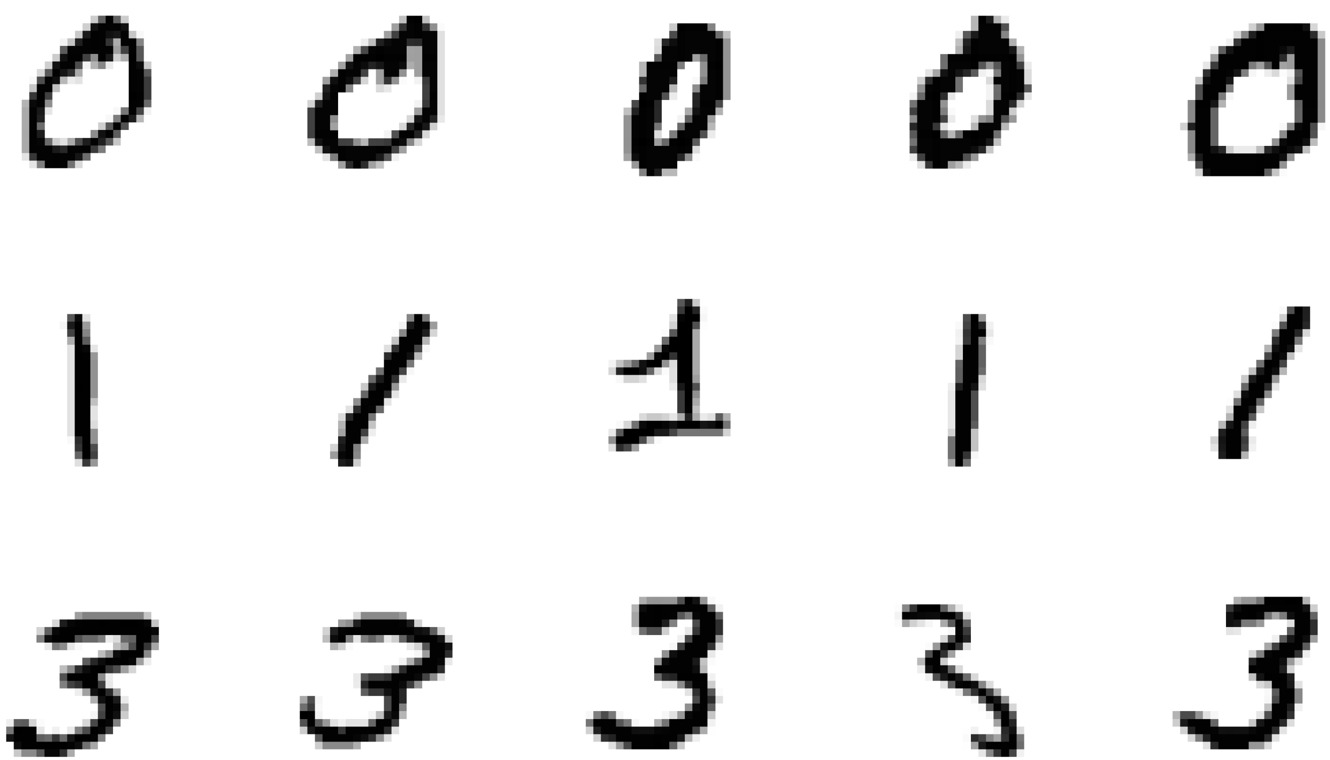
after a short training period

after a few hours of training

Preprocessing

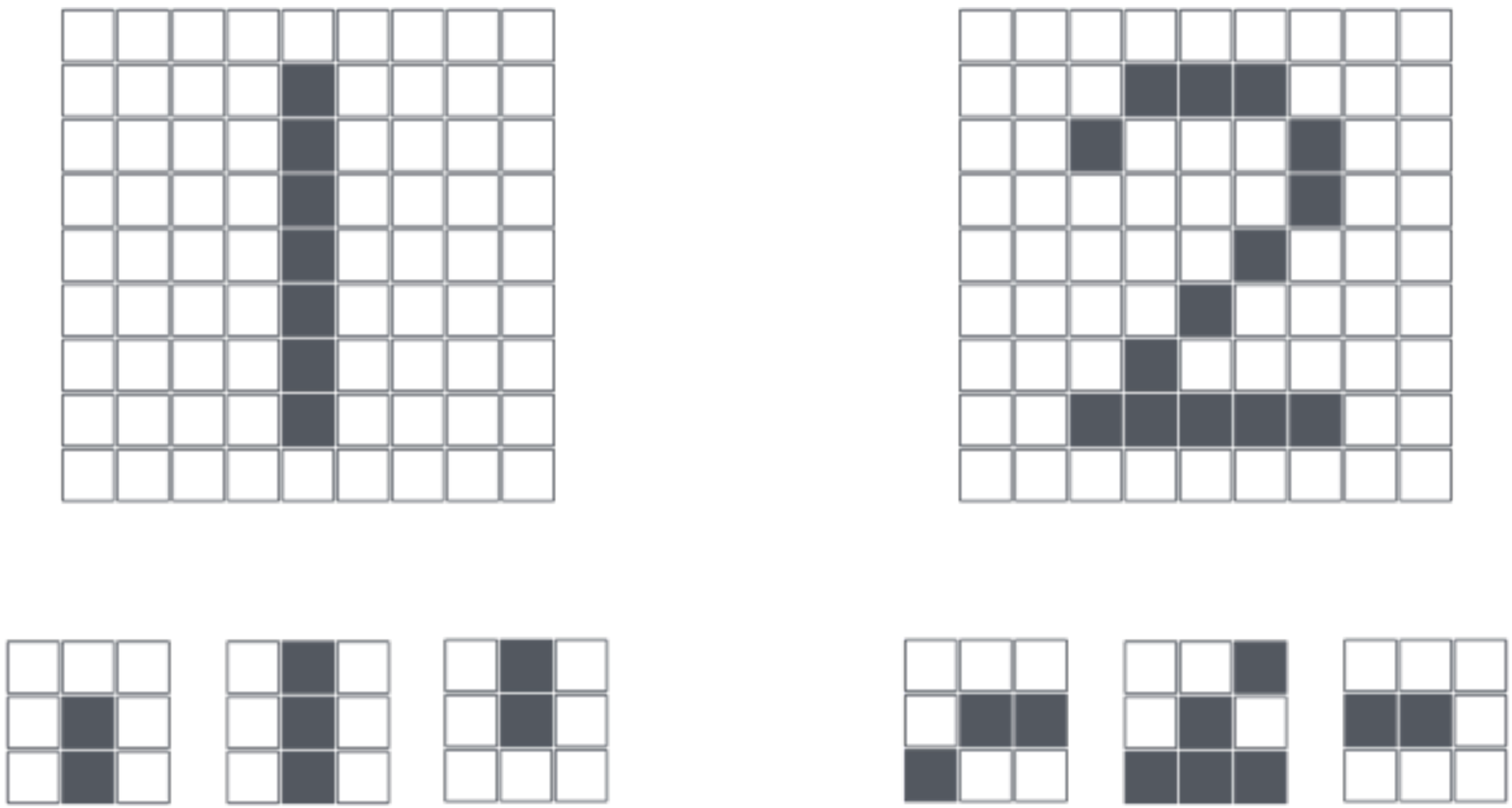
convolutional neural networks

Convolutional neural networks preprocess data by first looking for **recurring patterns** using small filters (and then sending it into a neural network).



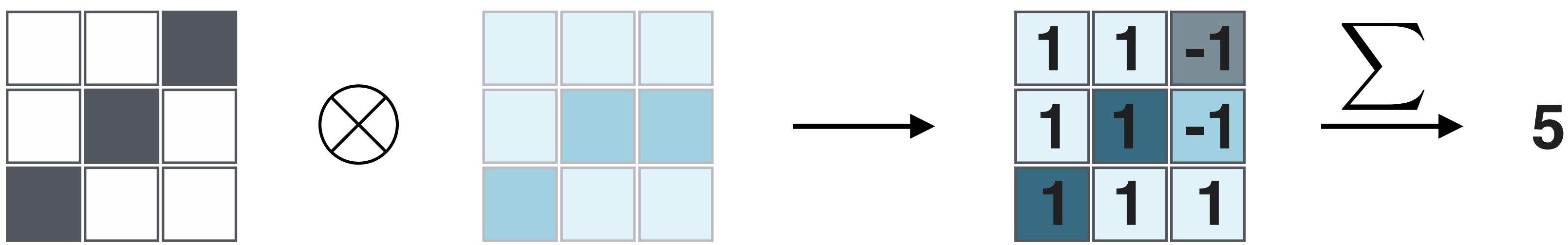
convolutional neural networks

Convolutional neural networks look for **recurring patterns** using small filters.



convolutional neural networks

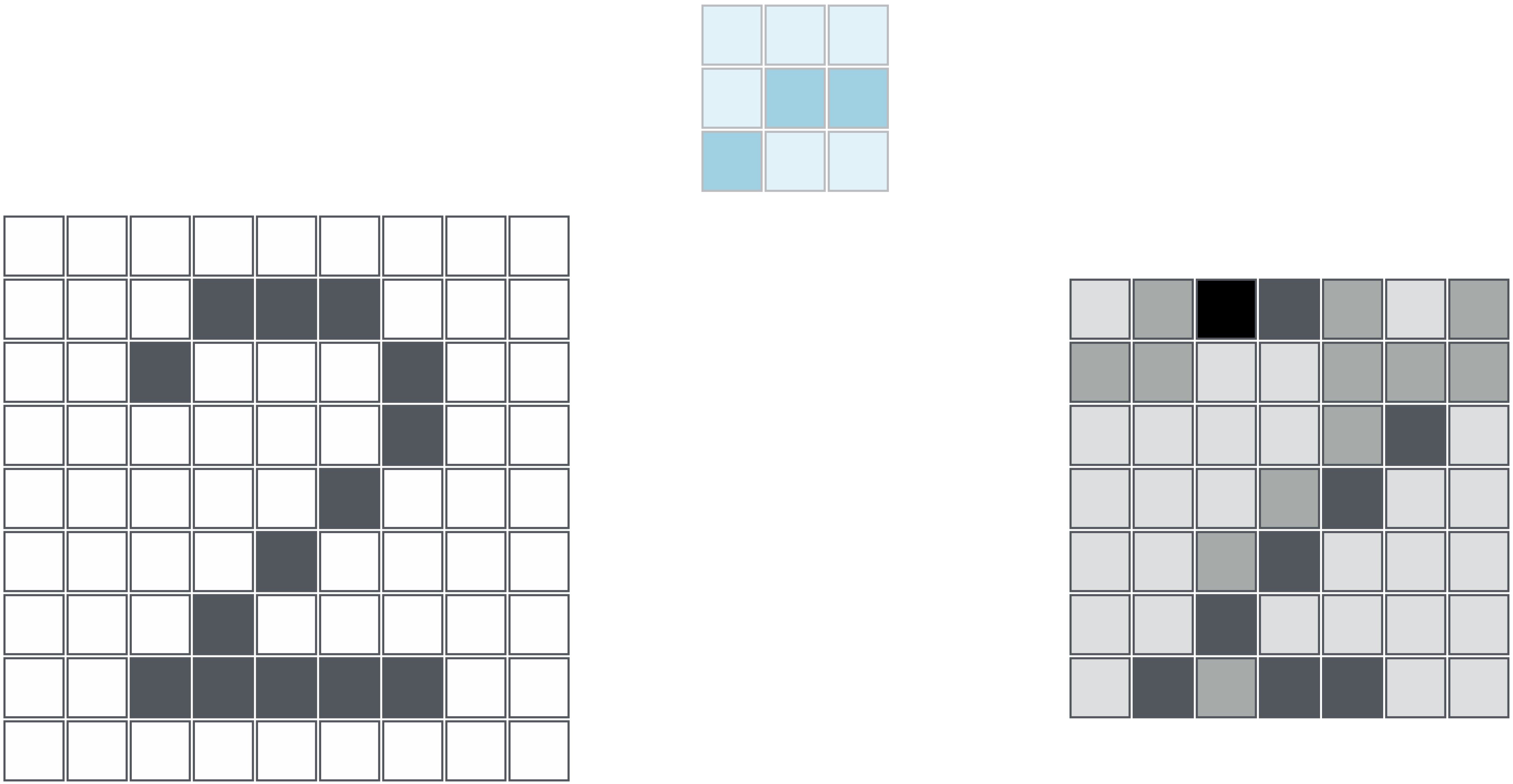
Convolutional neural networks look for **recurring patterns** using small filters.



Slide filters across image and create new image based on how well they fit.

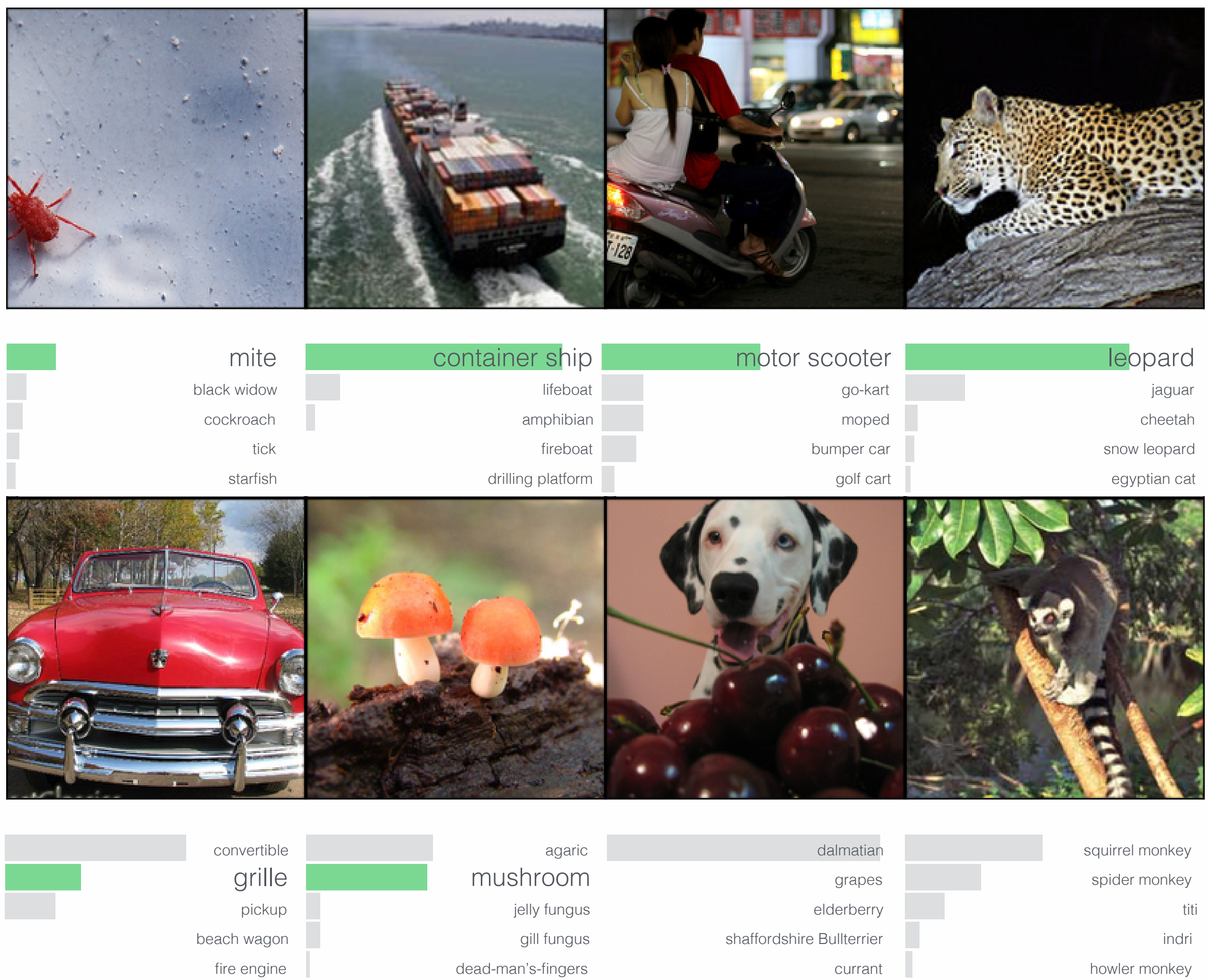
convolutional neural networks

Convolutional neural networks look for **recurring patterns** using small filters.



convolutional neural networks

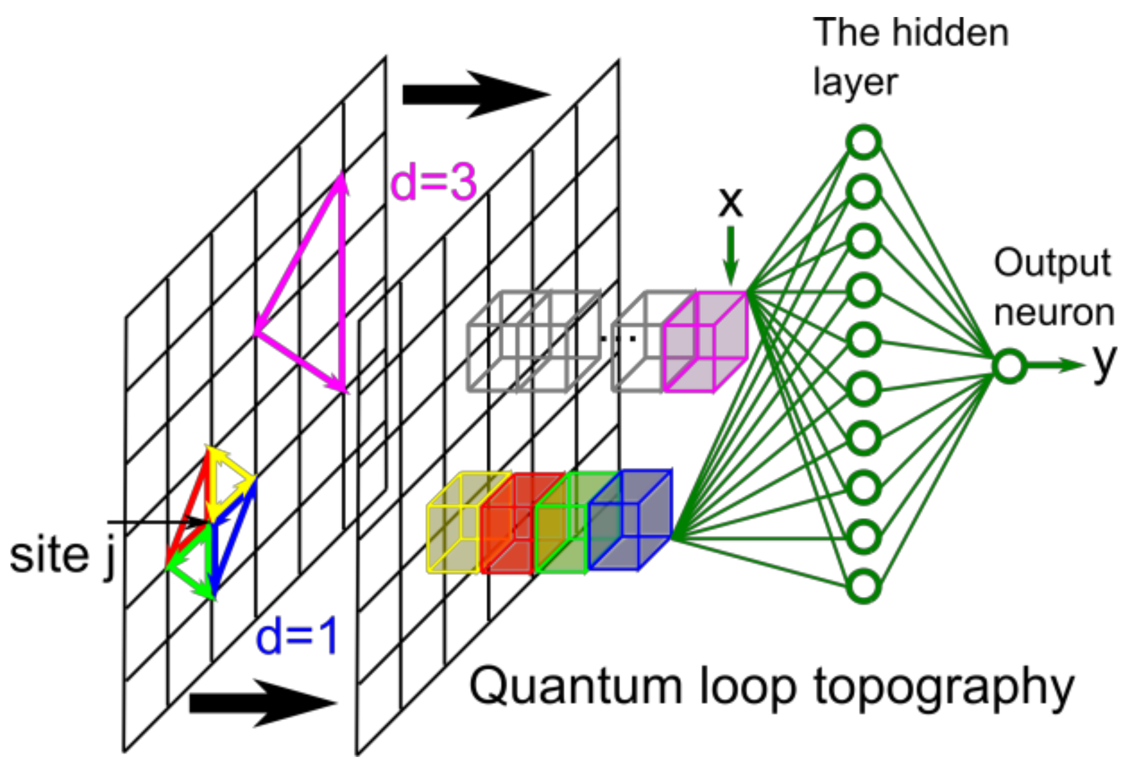
Convolutional neural networks have proved to be some of the most powerful ingredients for **pattern recognition**/machine learning.



Physics: topological preprocessing

Yi (Frank) Zhang and Eun-Ah Kim, PRL (2017)

Quantum loop topography is a physics preprocessor allowing to identify features associated with topological order in quantum many-body systems.



Quantum loop = sample of two-point operators that form loops.

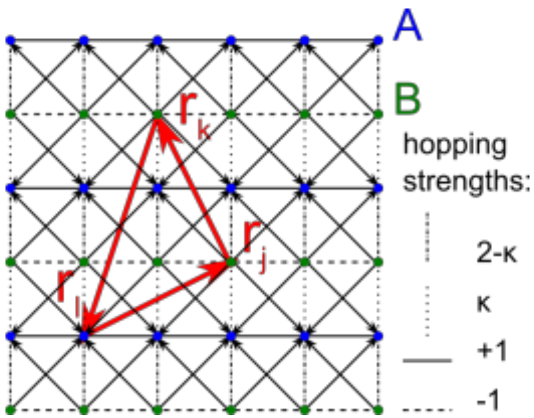
$$\tilde{P}_{jk}\tilde{P}_{kl}\tilde{P}_{lj}$$

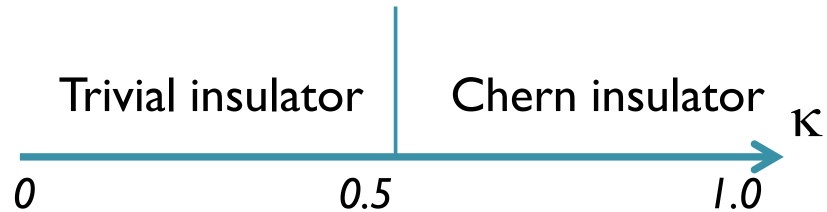
$$\tilde{P}_{jk} \equiv \left\langle c_j^\dagger c_k \right\rangle_\alpha$$

Physics: topological preprocessing

Yi (Frank) Zhang and Eun-Ah Kim, PRL (2017)

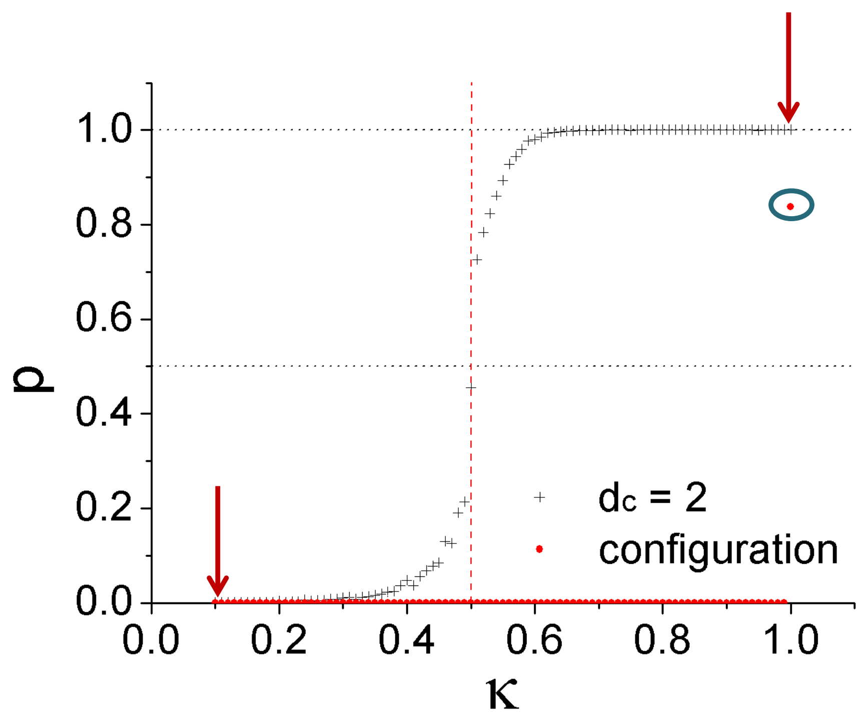
Quantum loop topography is a physics preprocessor allowing to identify features associated with topological order in quantum many-body systems.





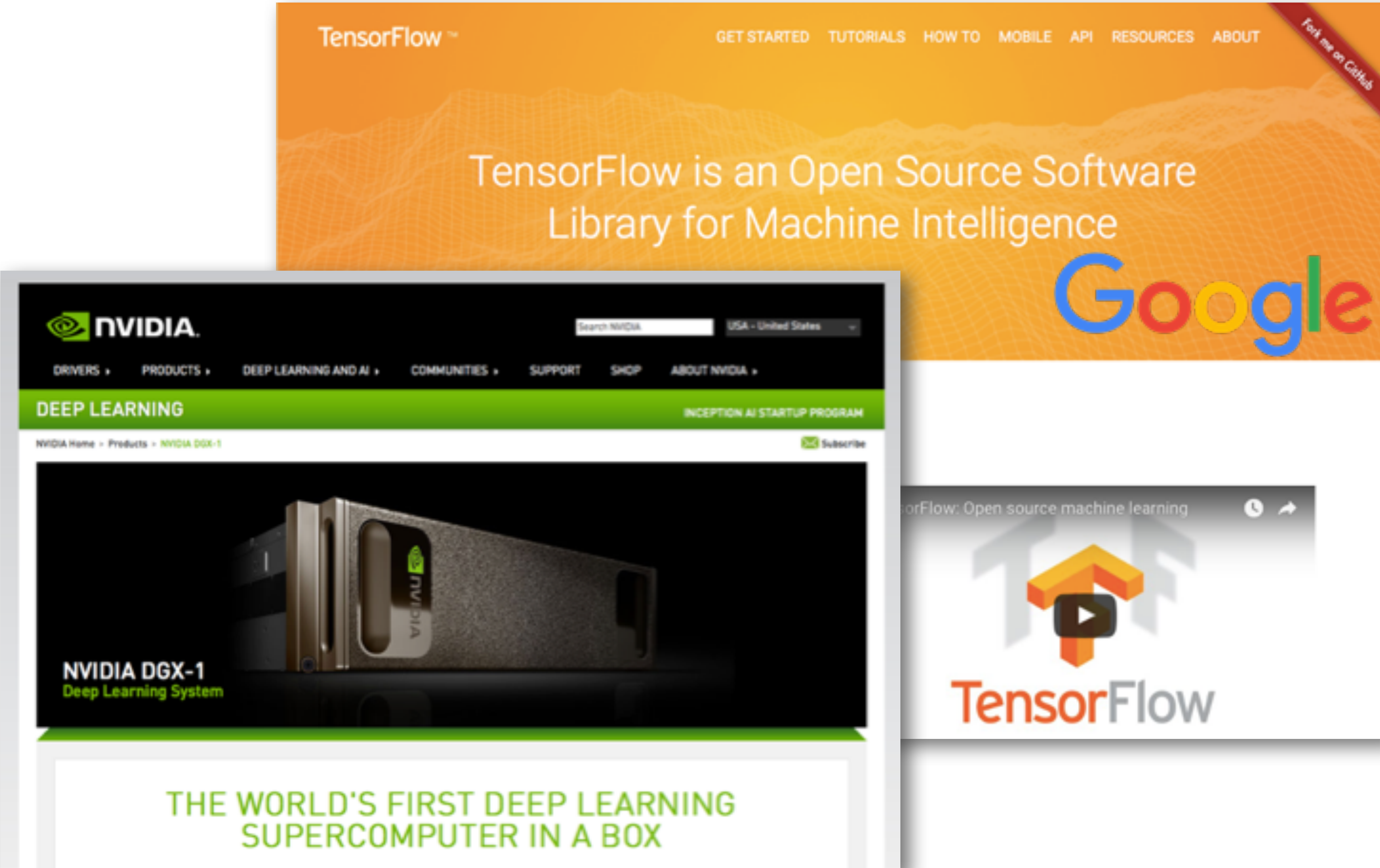
- Training set

Trivial	κ=0.1
Topological	κ=1.0



Need for Speed

GPUs & open-source codes



Thanks!

Let’s take a break.

@SimonTrebst