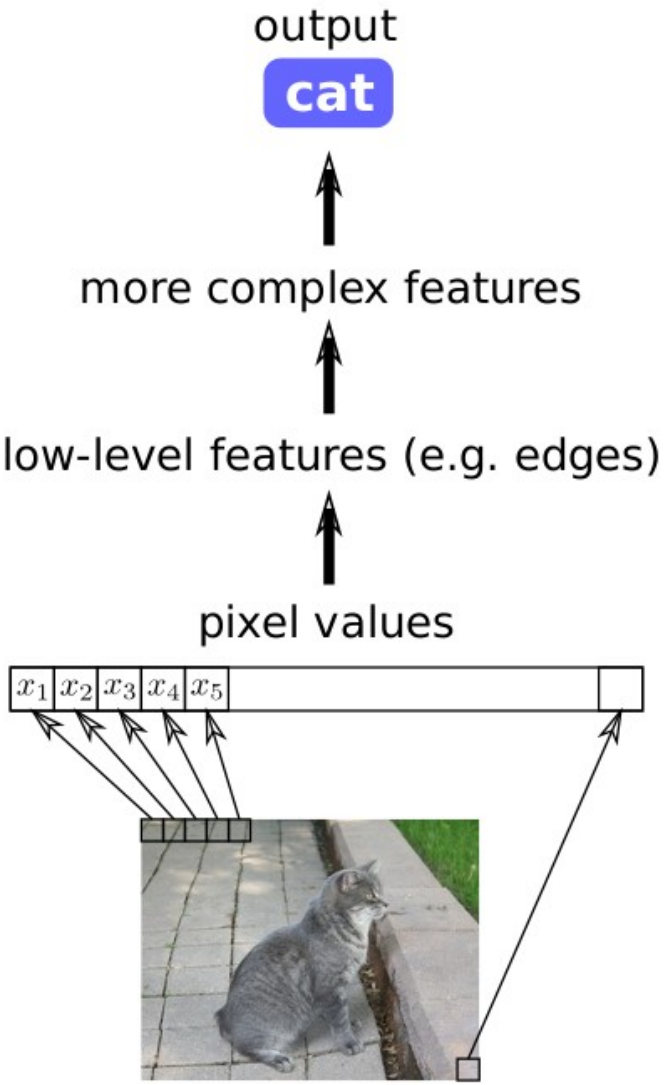
SoSe 2018: Deep Neural Networks

# Lecture 1: Introduction

Machine Learning Group
Technische Universität Berlin

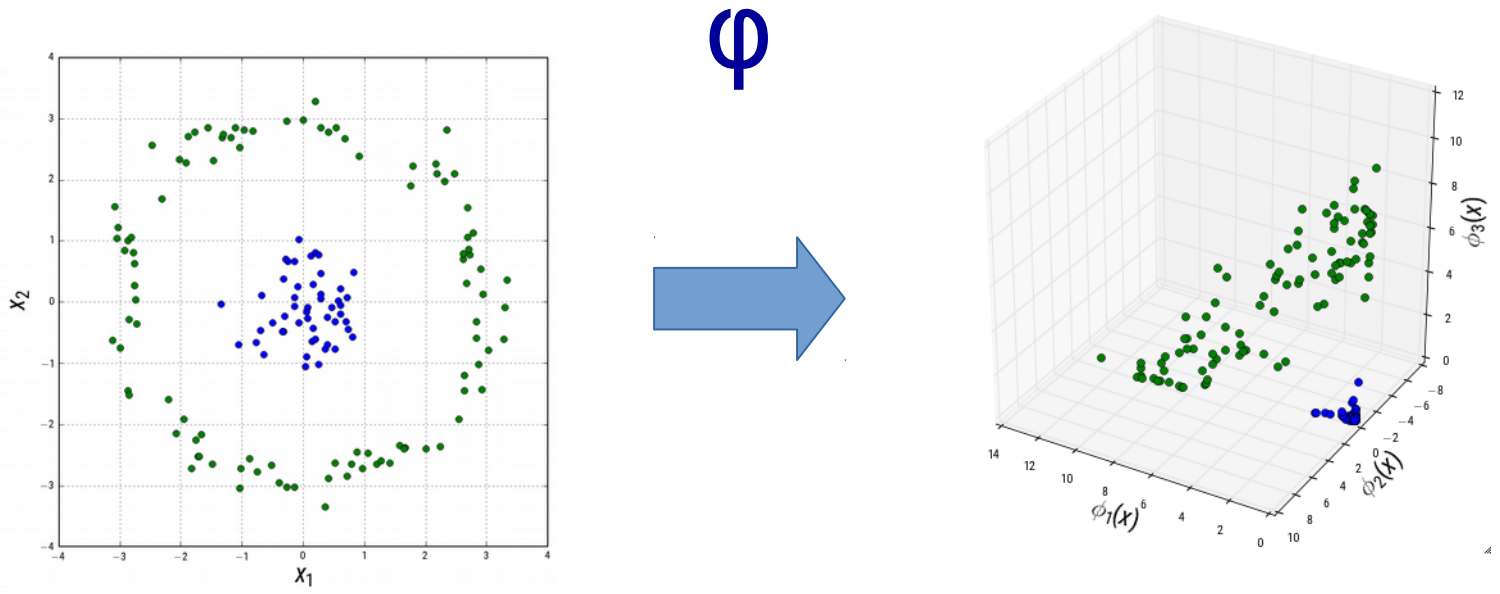
Technische Universität Berlin

# From Input to Abstractions



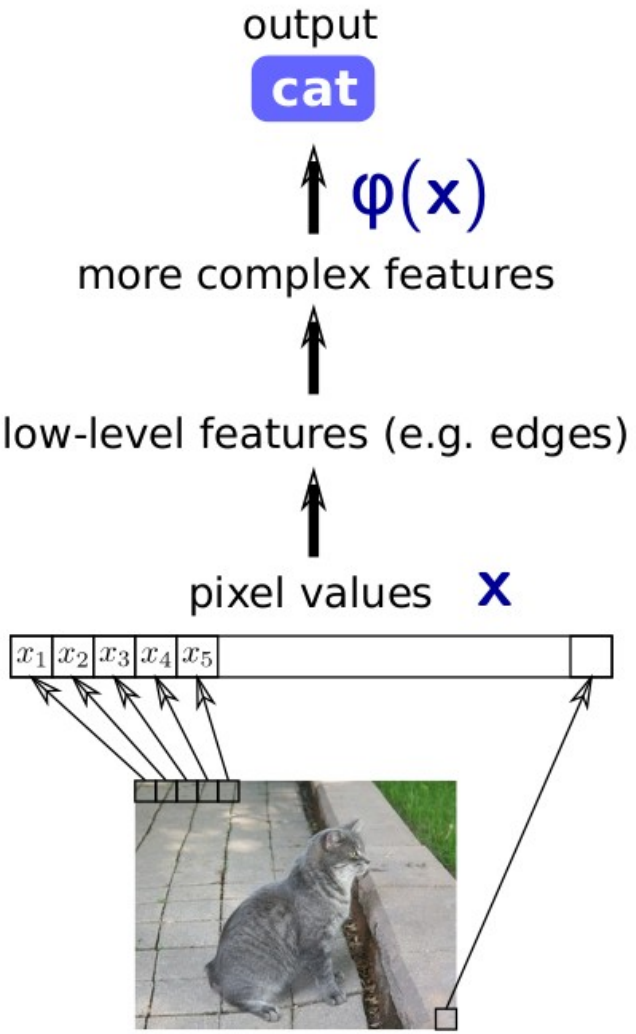
# Nonlinear Models

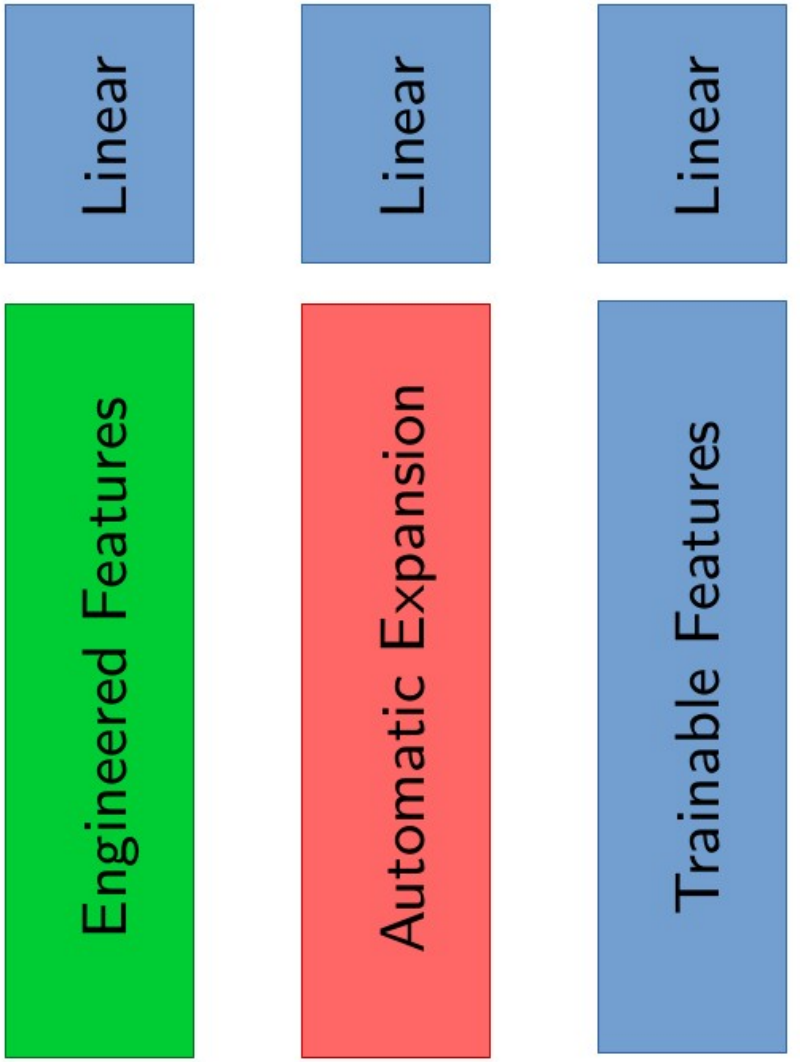
A common way to solve nonlinear problems is to apply a nonlinear mapping to some high-dimensional feature space, followed by linear function. The linear function is easy to fit with the data.




**Question:** What should the nonlinear part be like?

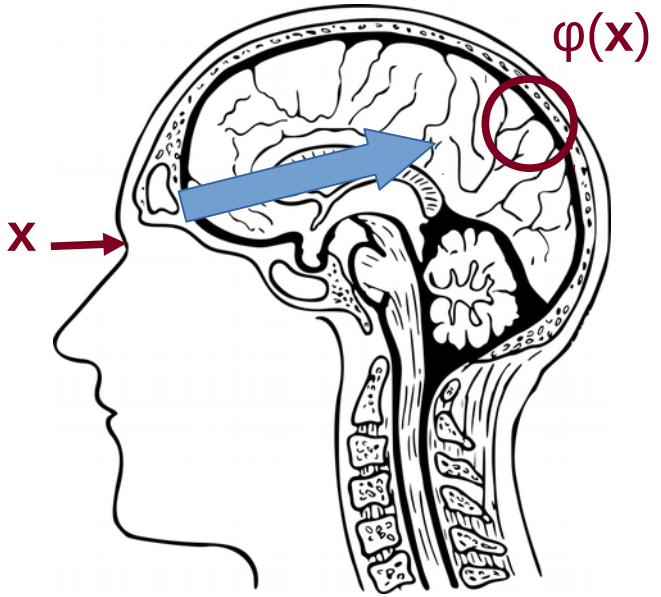
# Representing and Learning Abstractions



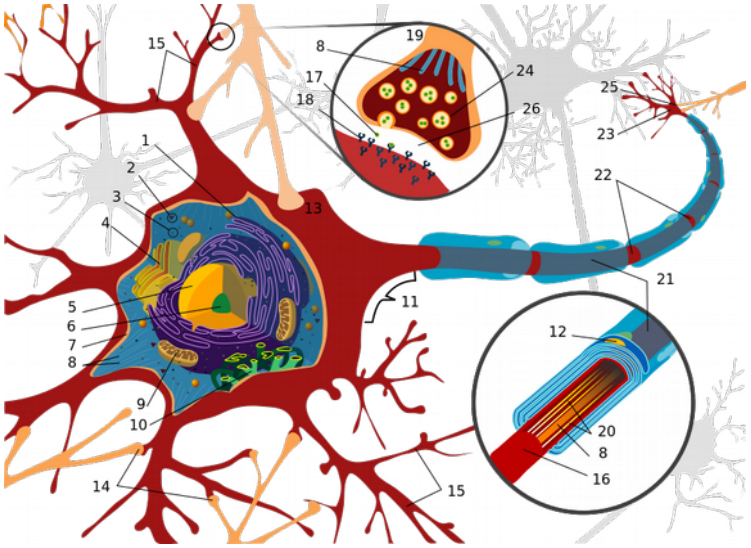
# Representation and Learning in the Brain

The brain has the ability to represent the desired abstractions, and to learn them from repeated observations.

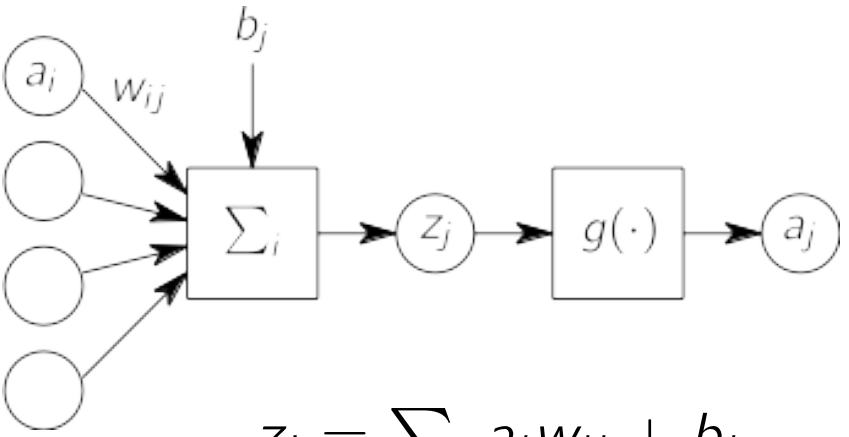



Artificial neural networks aim to mimic the ability of the brain to represent and learn the abstractions.

# Biological vs. Artificial Neuron




Highly sophisticated physical system, with spatio-temporal dynamics.




$$z_j = \sum_i a_i w_{ij} + b_j$$

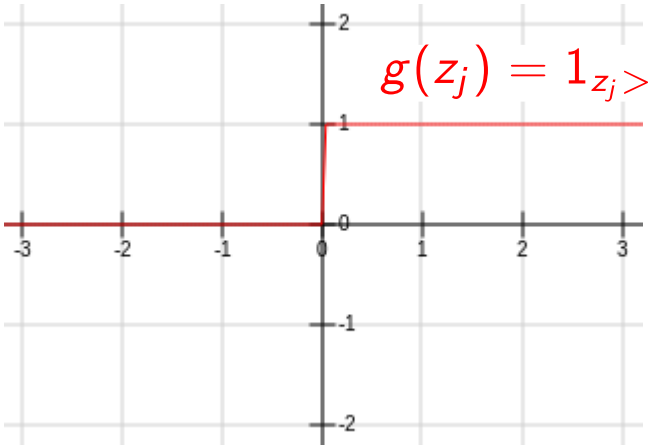
$$a_j = g(z_j)$$

Simple multivariate and nonlinear function.

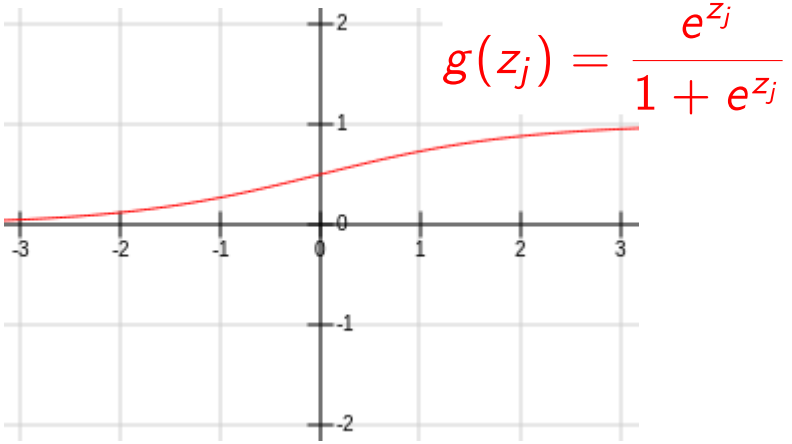
**Common denominator:** (1) They both have an adaptation mechanism. (2) Ability to represent abstractions derives from interconnecting a large number of them.

# Examples of Activation Functions g( · )

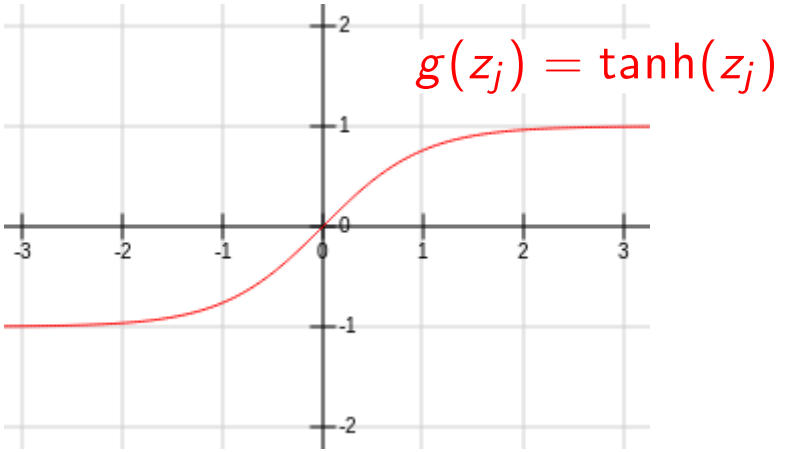
## threshold function



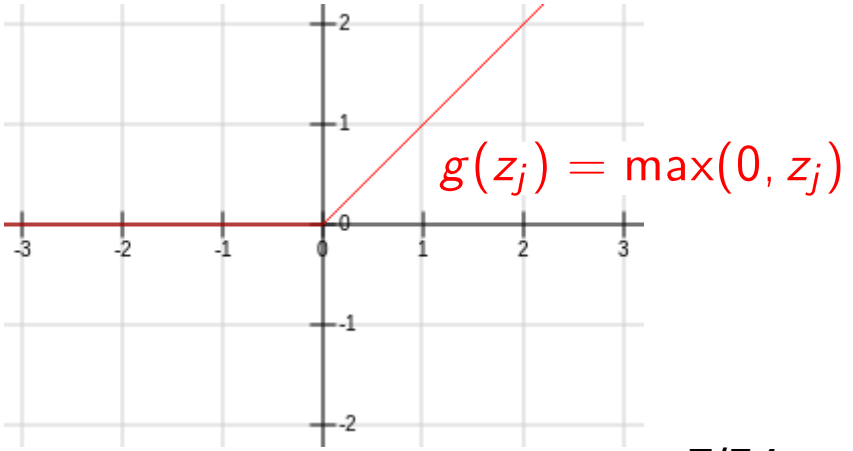

## logistic sigmoid



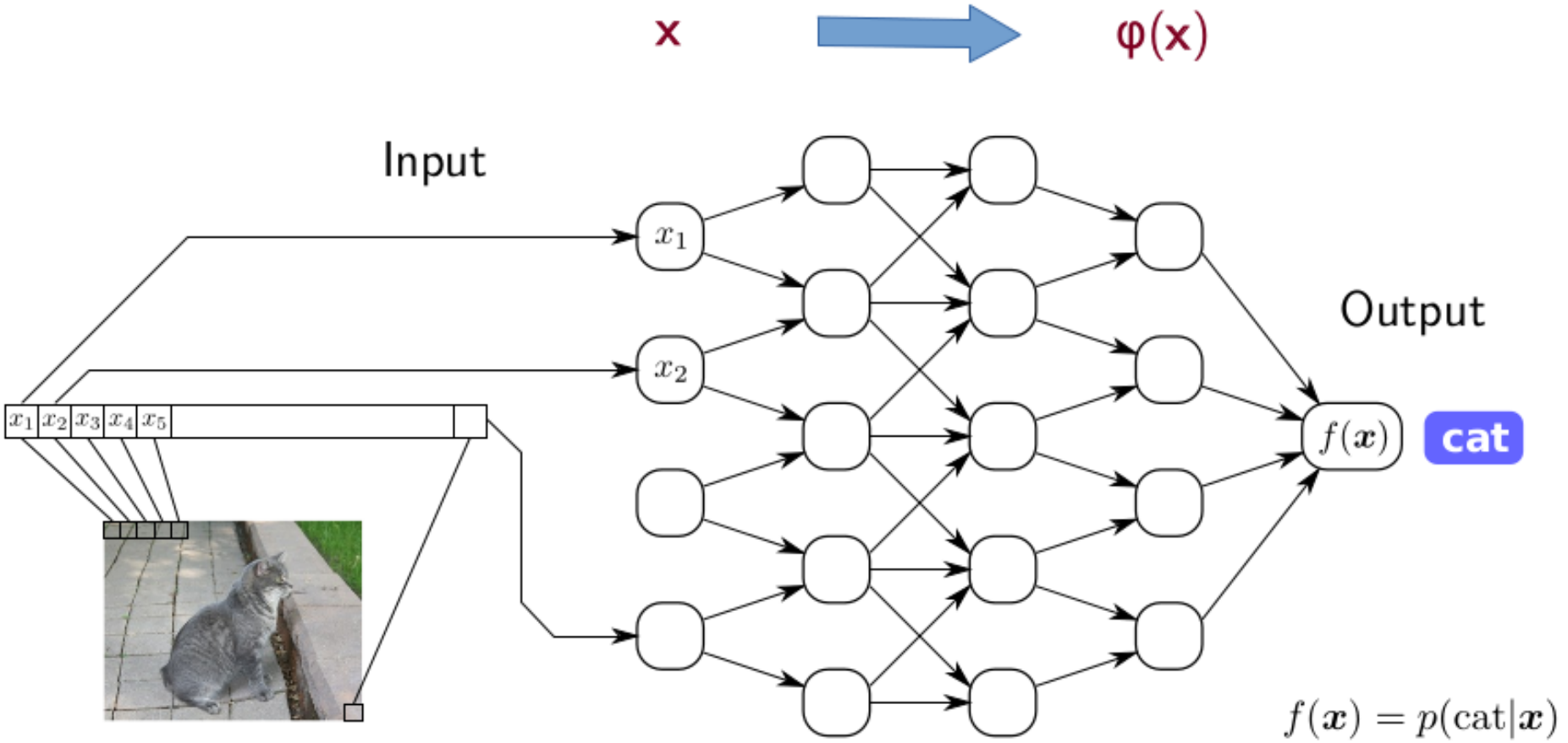

## hyperbolic tangent




## rectified linear unit

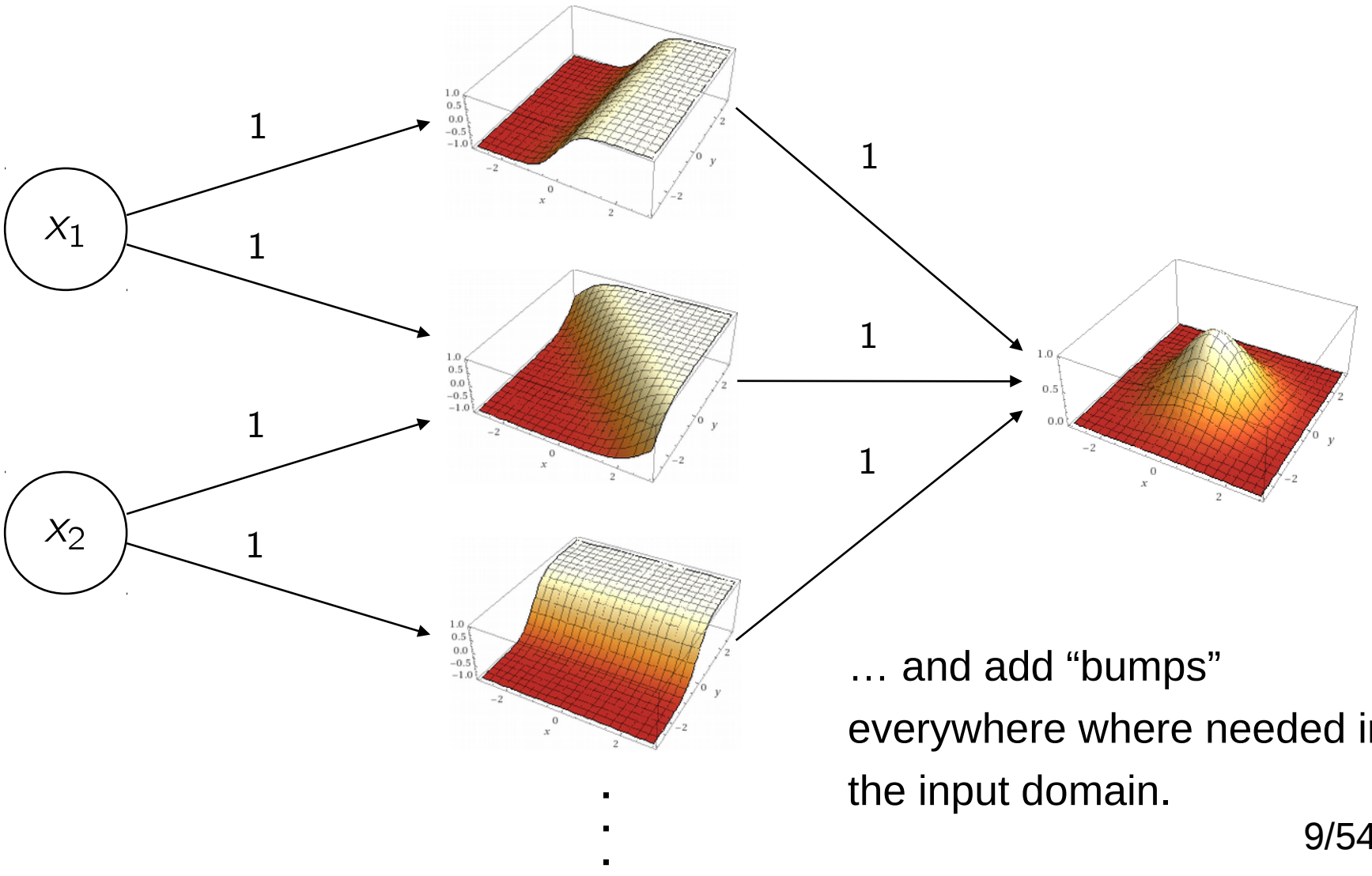


# Graphical View of a Neural Network



# Neural Networks are Universal Approximators

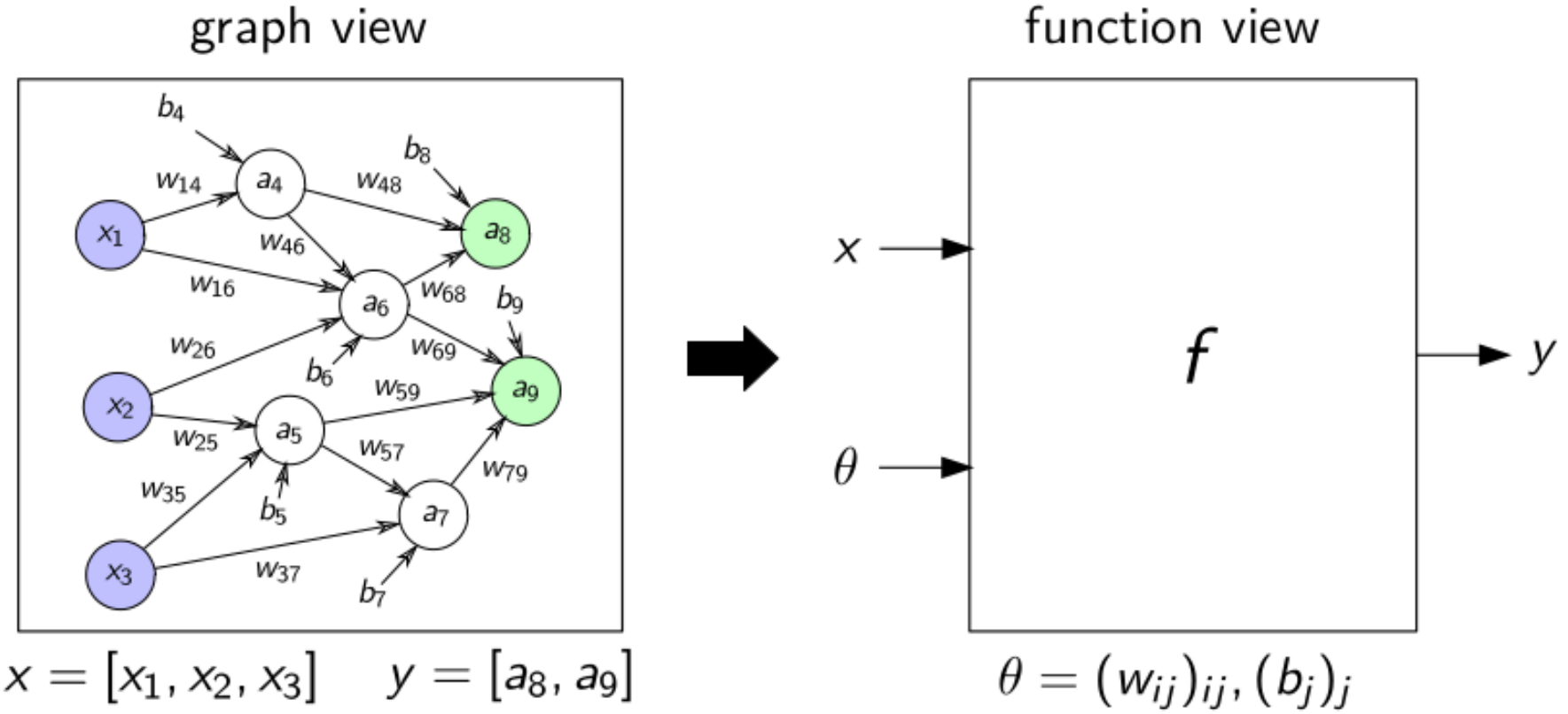
"Proof" by construction:




… and add "bumps" everywhere where needed in the input domain.

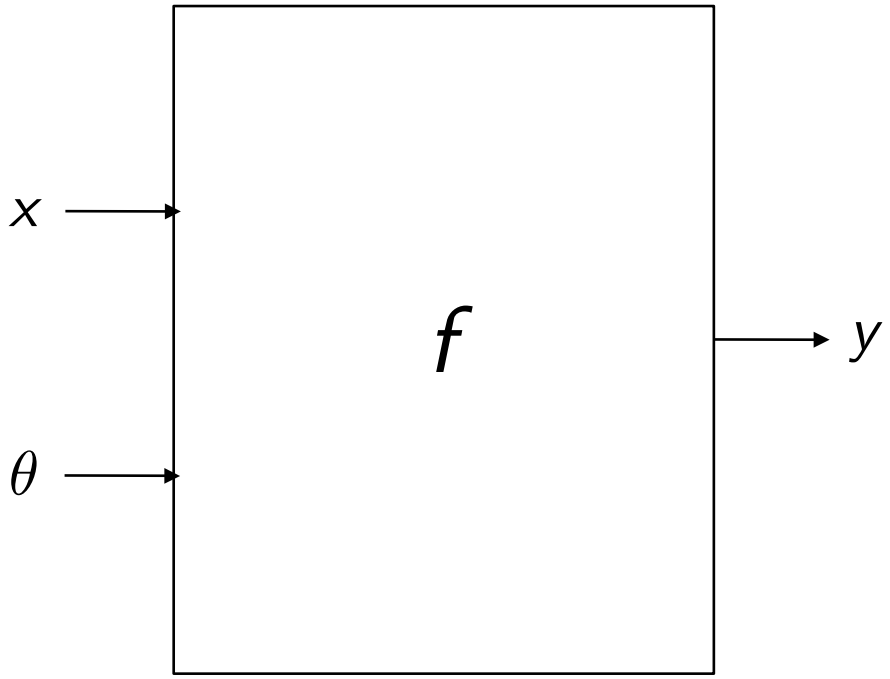
# How to Learn in a Neural Network

**Observation:** A neural network is a function of both its inputs and parameters.



# How to Learn in a Neural Network

function view

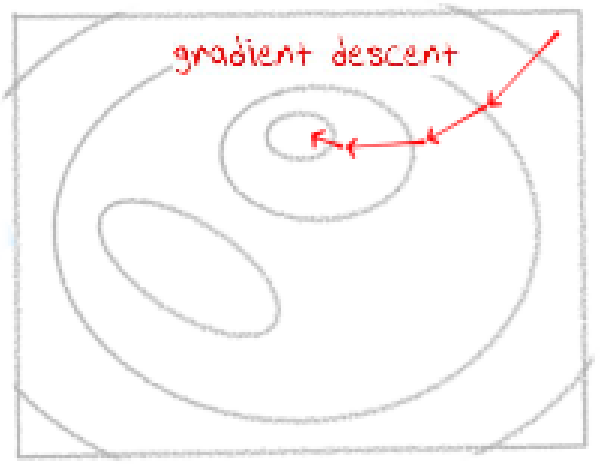



Define an error function

$$E(\theta) = \sum_n (f(\boldsymbol{x}_n; \theta) - t_n)^2$$

and minimize by gradient descent

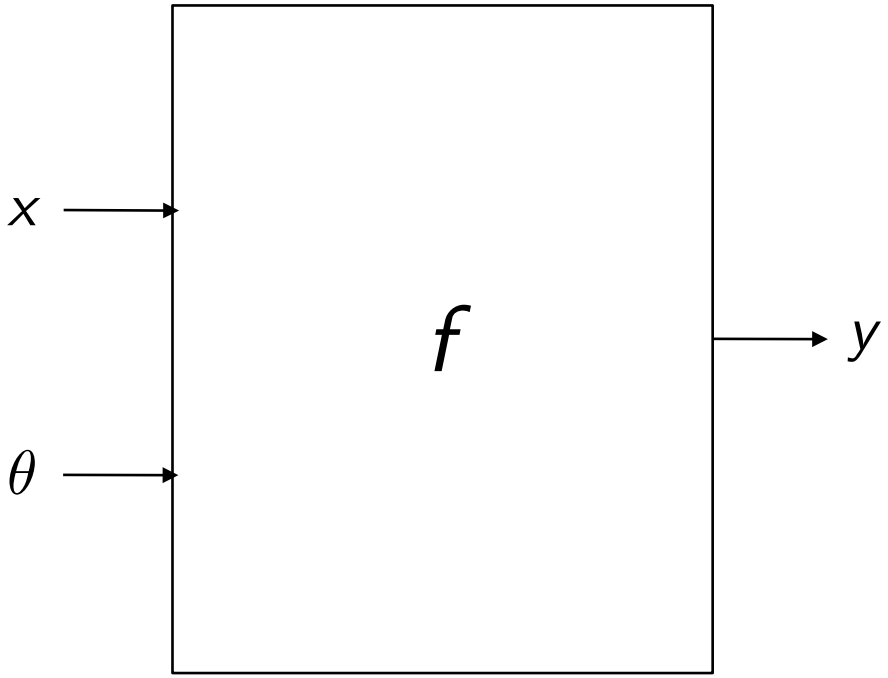
$$\theta \leftarrow \theta - \gamma \cdot \nabla_\theta E(\theta)$$



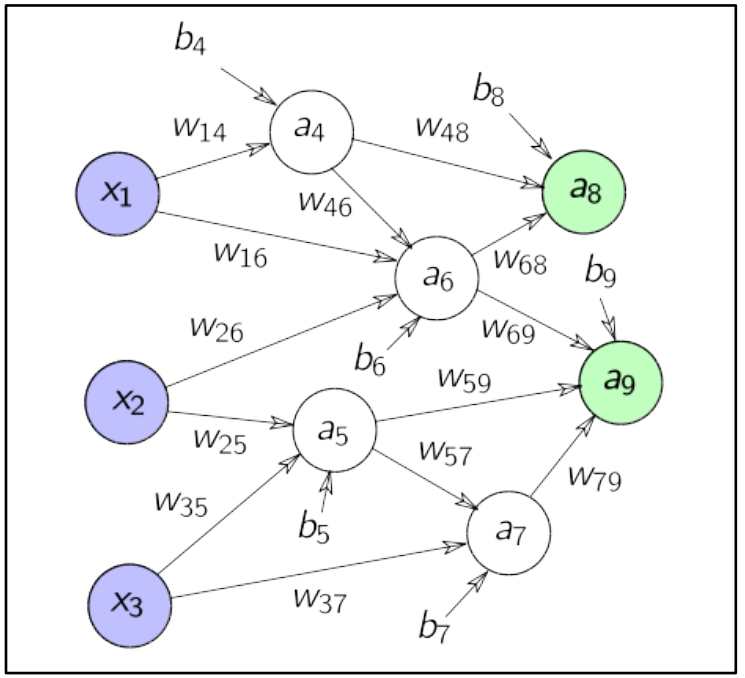
# How to Compute the Gradient?

**Idea:** go back to the graph view...

function view



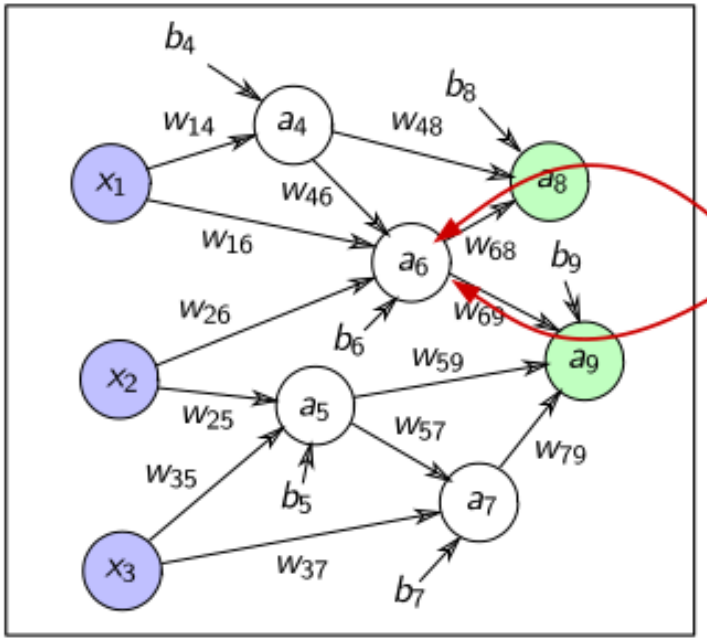

graph view




and propagate derivatives using the chain rule.

# Error Backpropagation

graph view




$E^{(n)}$

Multivariate chain rule:

$$\frac{\partial E^{(n)}}{\partial a_6} = \frac{\partial E^{(n)}}{\partial a_8}\frac{\partial a_8}{\partial a_6} + \frac{\partial E^{(n)}}{\partial a_9}\frac{\partial a_9}{\partial a_6}$$

received from higher layers

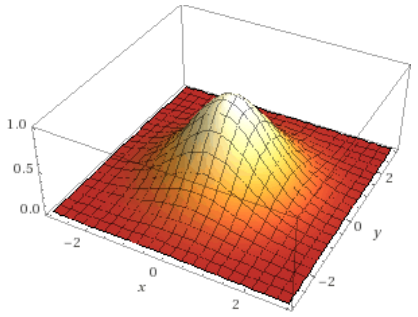
easy to derive (local computation)

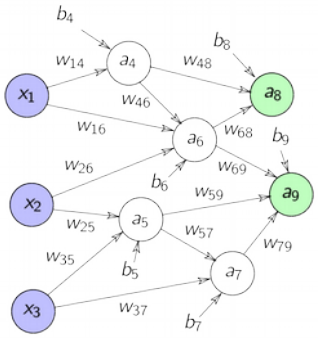
cost of backprop: $O(\#\text{edges})$ (graph traversal)

Deep Neural Networks

# Week # 2

## Universal Approximation Theorems, Error Backpropagation



# How to Optimize?

Objective to Minimize:

$$E(\theta) = \frac{1}{N} \sum_{n=1}^{N} E^{(n)}(\theta)$$

Batch GD:

**while** True:

$$\theta \leftarrow \theta - \gamma \frac{\partial}{\partial \theta} \frac{1}{N} \sum_{n=1}^{N} E^{(n)}$$

$O(N)$

Stochastic GD (SGD):

**while** True:

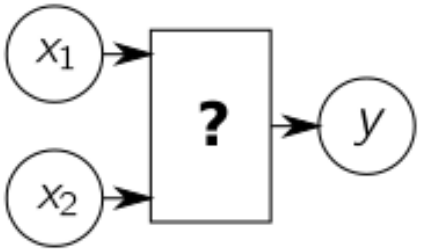
$$n \leftarrow \text{random}(1, N)$$

$$\theta \leftarrow \theta - \gamma \frac{\partial E^{(n)}}{\partial \theta}$$

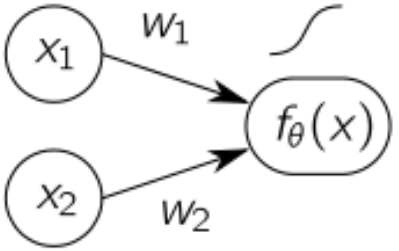
$O(1)$

# GD vs. SGD

ground truth

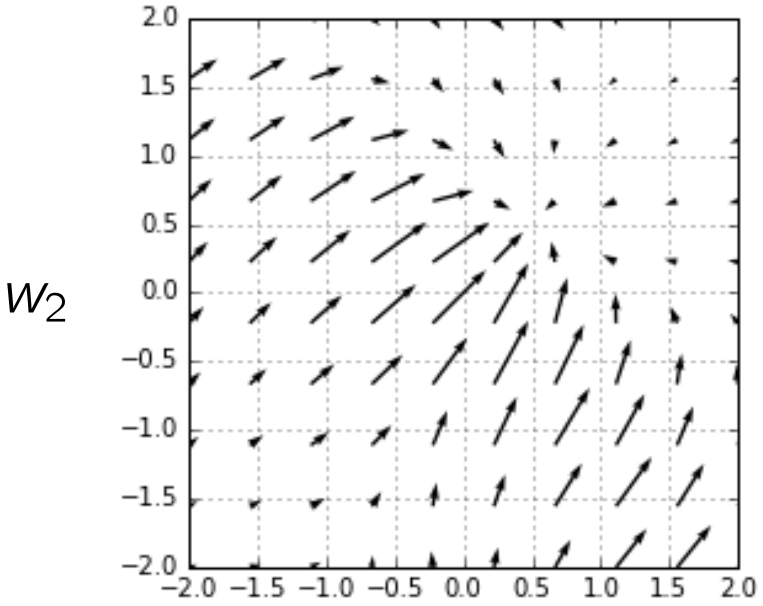


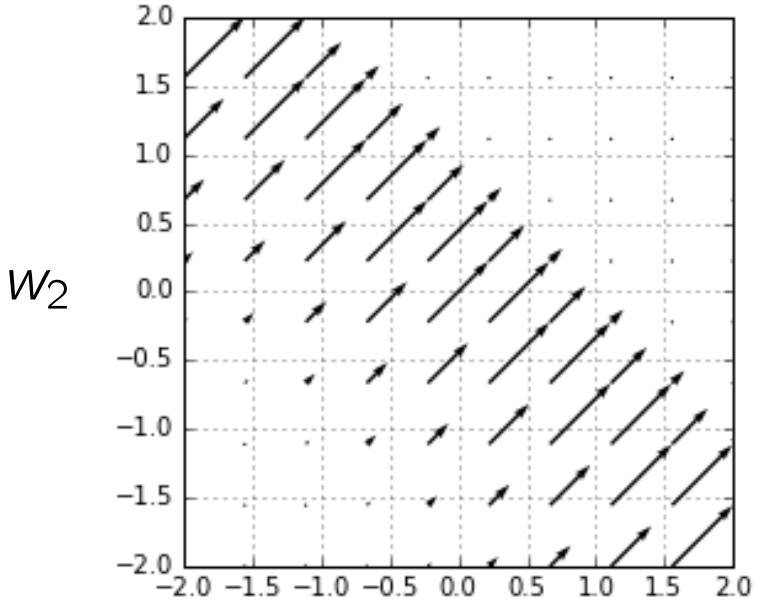

model




GD objective

$$\frac{1}{N}\sum_{i=1}^{N}(y^{(n)} - f_\theta(x^{(n)}))^p$$

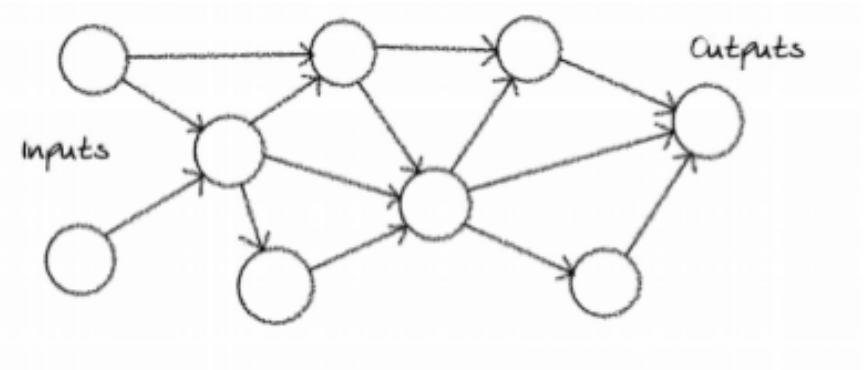


# Neural Network Training Time

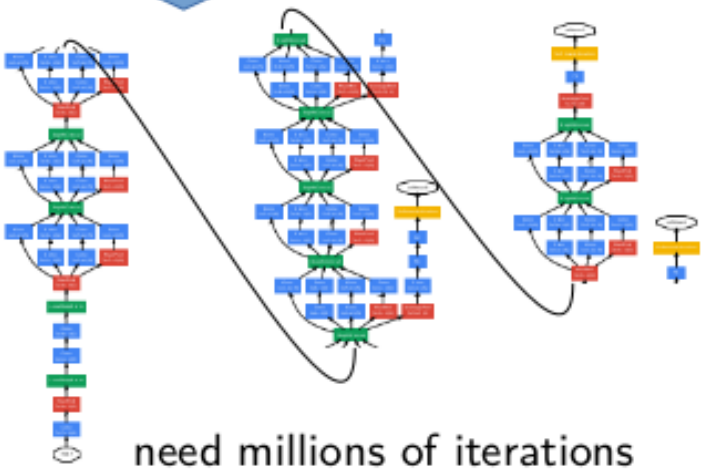
**SGD** → $O(\#\text{iterations} \times \#\text{edges})$ ← **Backprop**

This network: 13 connections



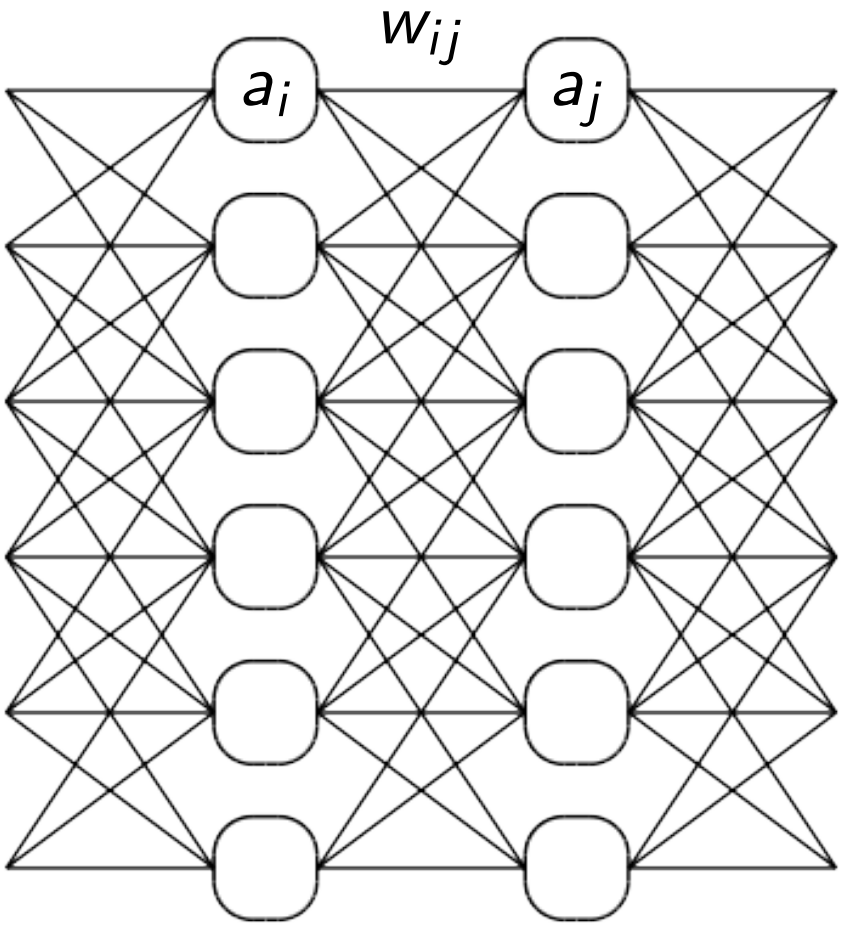

can probably be trained with a few hundreds iterations.

Googlenet: $> 10^9$ connections



need millions of iterations to be trained.

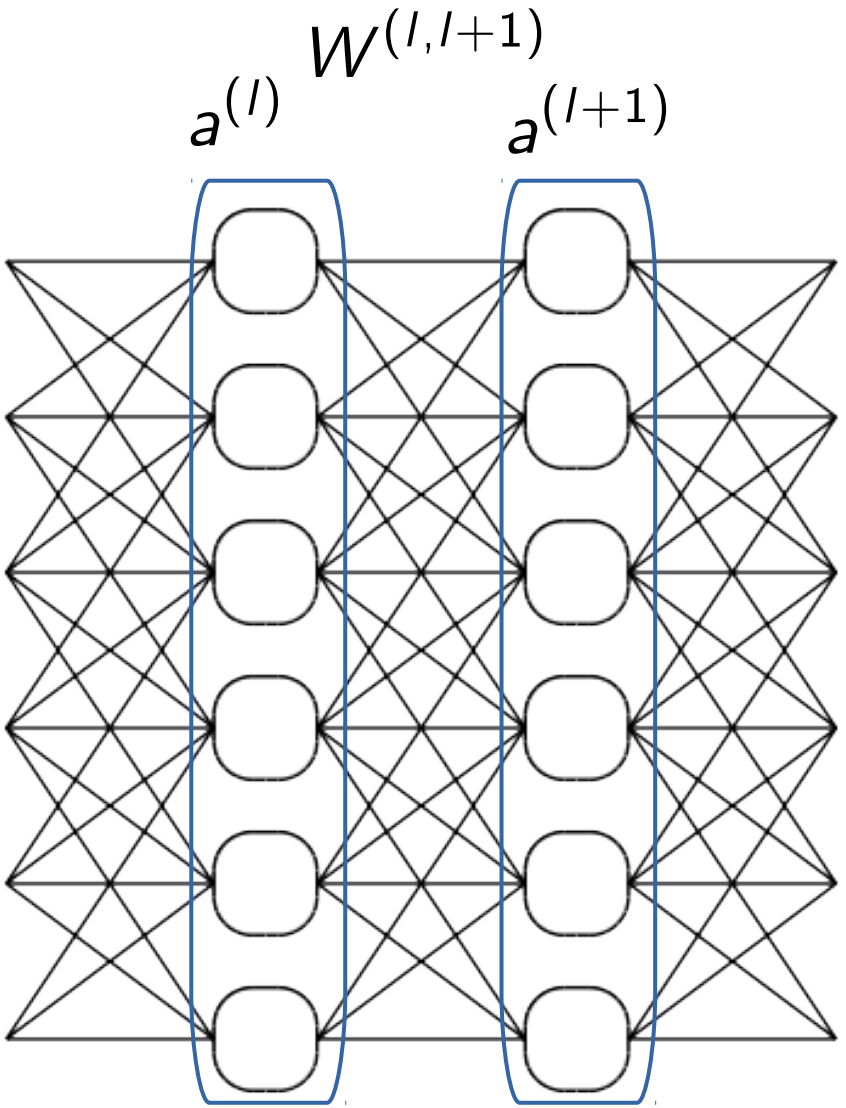
# Structuring DNNs for Parallel Computation




Per-neuron forward computations

$$\forall_j : a_j = g\left(\sum_i a_i w_{ij} + b_j\right)$$

# Structuring DNNs for Parallel Computation




Per-neuron forward computations

$$\forall_j : a_j = g\left(\sum_i a_i w_{ij} + b_j\right)$$

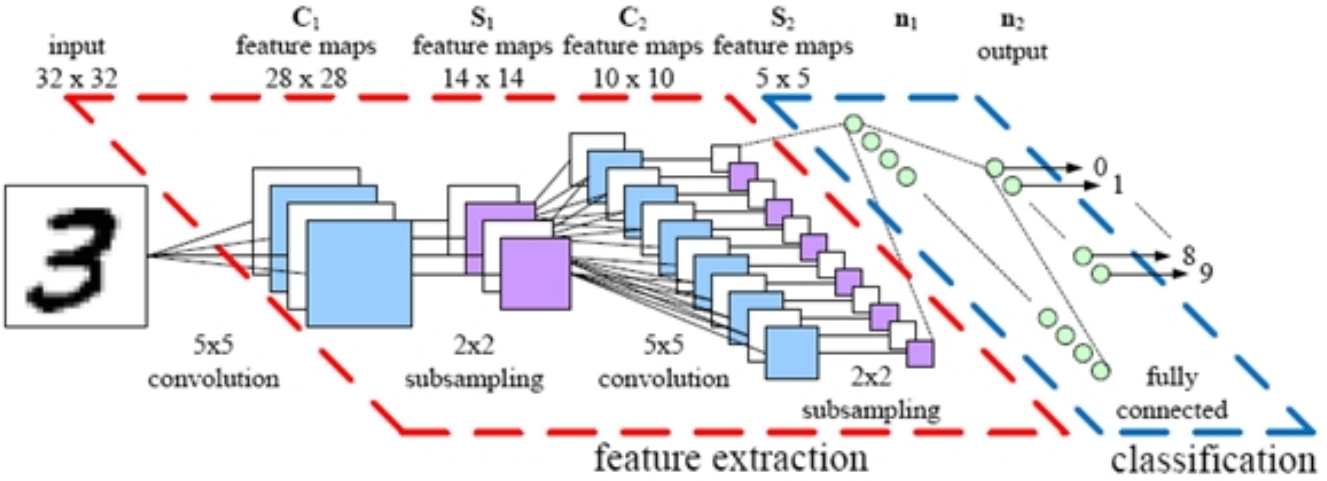
**Whole-layer computation**

$$a^{(l+1)} = g\left(W^{(l,l+1)} \cdot a^{(l)} + b^{(l+1)}\right)$$

matrix-vector products (e.g. numpy.dot)

element-wise application of nonlinearity

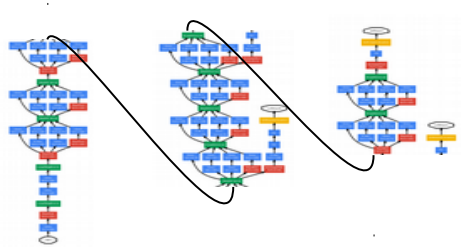
# ConvNets: Avoiding Dimensionality Bottlenecks




(source: Peemen et al. 2011: Speed sign detection and recognition by convolutional neural networks).

- Lower layers detect simple features at exact locations.
- Higher layers detect complex features at approximate locations.
- Layers progressively replace spatial information with semantic information → keep dimensionality and number of connections low at each layer.

# Example: GoogleNet

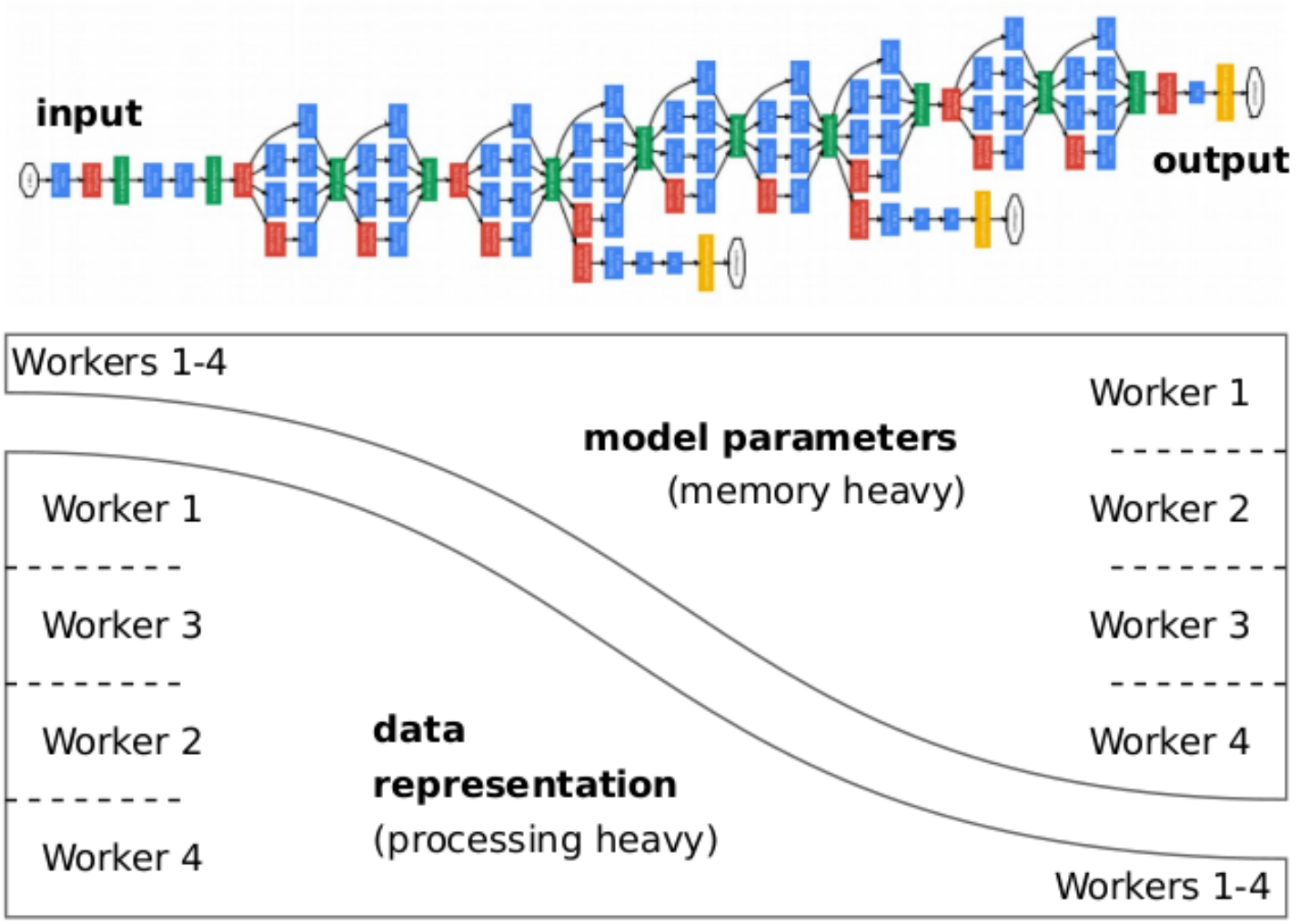


| type | patch size/ stride | output size | depth | #1×1 | #3×3 reduce | #3×3 | #5×5 reduce | #5×5 | pool proj | params | ops |
|---|---|---|---|---|---|---|---|---|---|---|---|
| convolution | 7×7/2 | 112×112×64 | 1 | | | | | | | 2.7K | 34M |
| max pool | 3×3/2 | 56×56×64 | 0 | | | | | | | | |
| convolution | 3×3/1 | 56×56×192 | 2 | | 64 | 192 | | | | 112K | 360M |
| max pool | 3×3/2 | 28×28×192 | 0 | | | | | | | | |
| inception (3a) | | 28×28×256 | 2 | 64 | 96 | 128 | 16 | 32 | 32 | 159K | 128M |
| inception (3b) | | 28×28×480 | 2 | 128 | 128 | 192 | 32 | 96 | 64 | 380K | 304M |
| max pool | 3×3/2 | 14×14×480 | 0 | | | | | | | | |
| inception (4a) | | 14×14×512 | 2 | 192 | 96 | 208 | 16 | 48 | 64 | 364K | 73M |
| inception (4b) | | 14×14×512 | 2 | 160 | 112 | 224 | 24 | 64 | 64 | 437K | 88M |
| inception (4c) | | 14×14×512 | 2 | 128 | 128 | 256 | 24 | 64 | 64 | 463K | 100M |
| inception (4d) | | 14×14×528 | 2 | 112 | 144 | 288 | 32 | 64 | 64 | 580K | 119M |
| inception (4e) | | 14×14×832 | 2 | 256 | 160 | 320 | 32 | 128 | 128 | 840K | 170M |
| max pool | 3×3/2 | 7×7×832 | 0 | | | | | | | | |
| inception (5a) | | 7×7×832 | 2 | 256 | 160 | 320 | 32 | 128 | 128 | 1072K | 54M |
| inception (5b) | | 7×7×1024 | 2 | 384 | 192 | 384 | 48 | 128 | 128 | 1388K | 71M |
| avg pool | 7×7/1 | 1×1×1024 | 0 | | | | | | | | |
| dropout (40%) | | 1×1×1024 | 0 | | | | | | | | |
| linear | | 1×1×1000 | 1 | | | | | | | 1000K | 1M |
| softmax | | 1×1×1000 | 0 | | | | | | | | |

Table 1: GoogLeNet incarnation of the Inception architecture

Szegedy'14

# Distributing Computations Across Machines

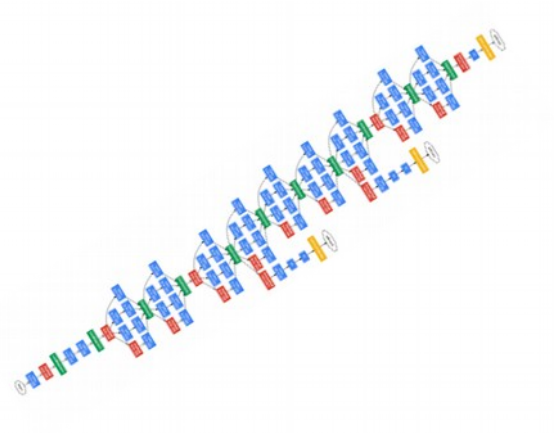



see also Krizhevsky'14: One weird trick for parallelizing convolutional neural networks

Deep Neural Networks

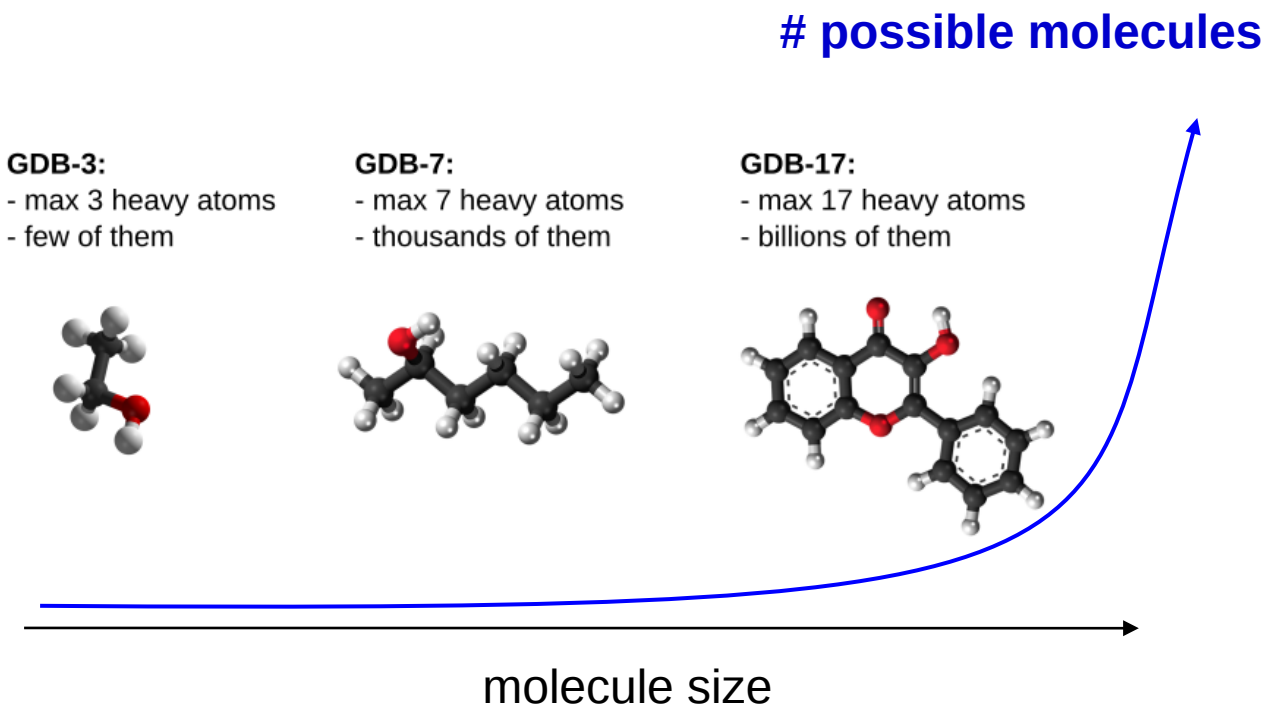
# Week # 3

## Scaling and Optimizing Deep Nets

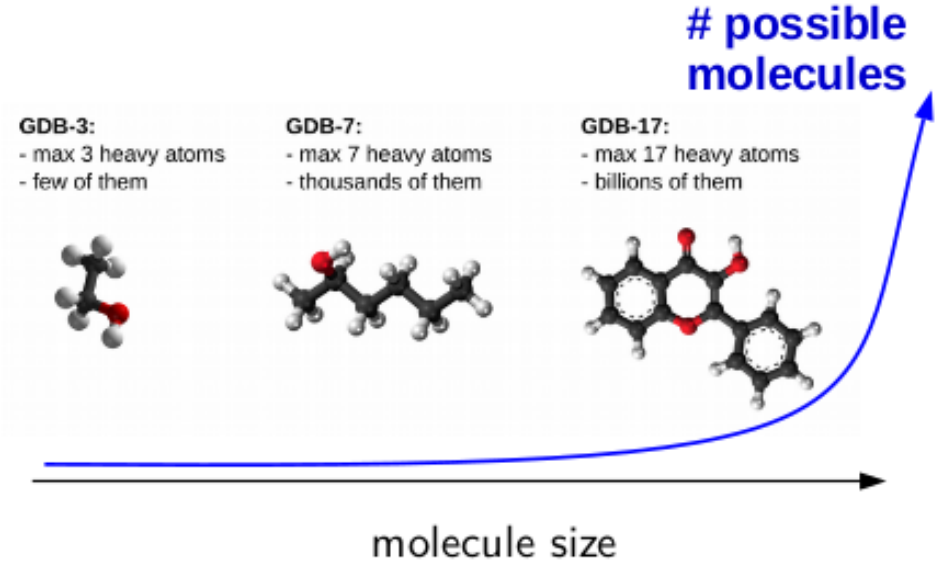
# Deep Nets for Quantum Chemical Predictions

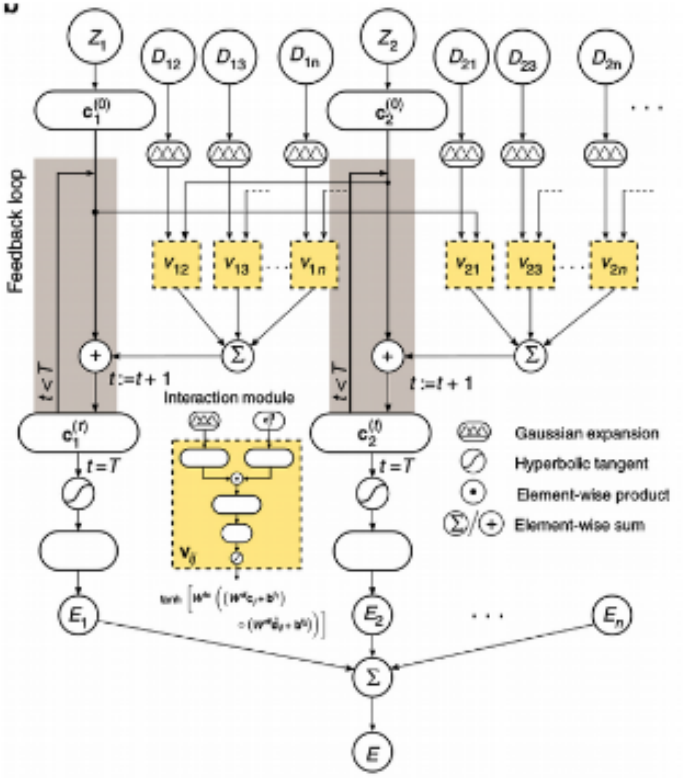
**Challenge:** how to predict an exponentially large number of molecules from finite examples?



# Deep Nets for Quantum Chemical Predictions

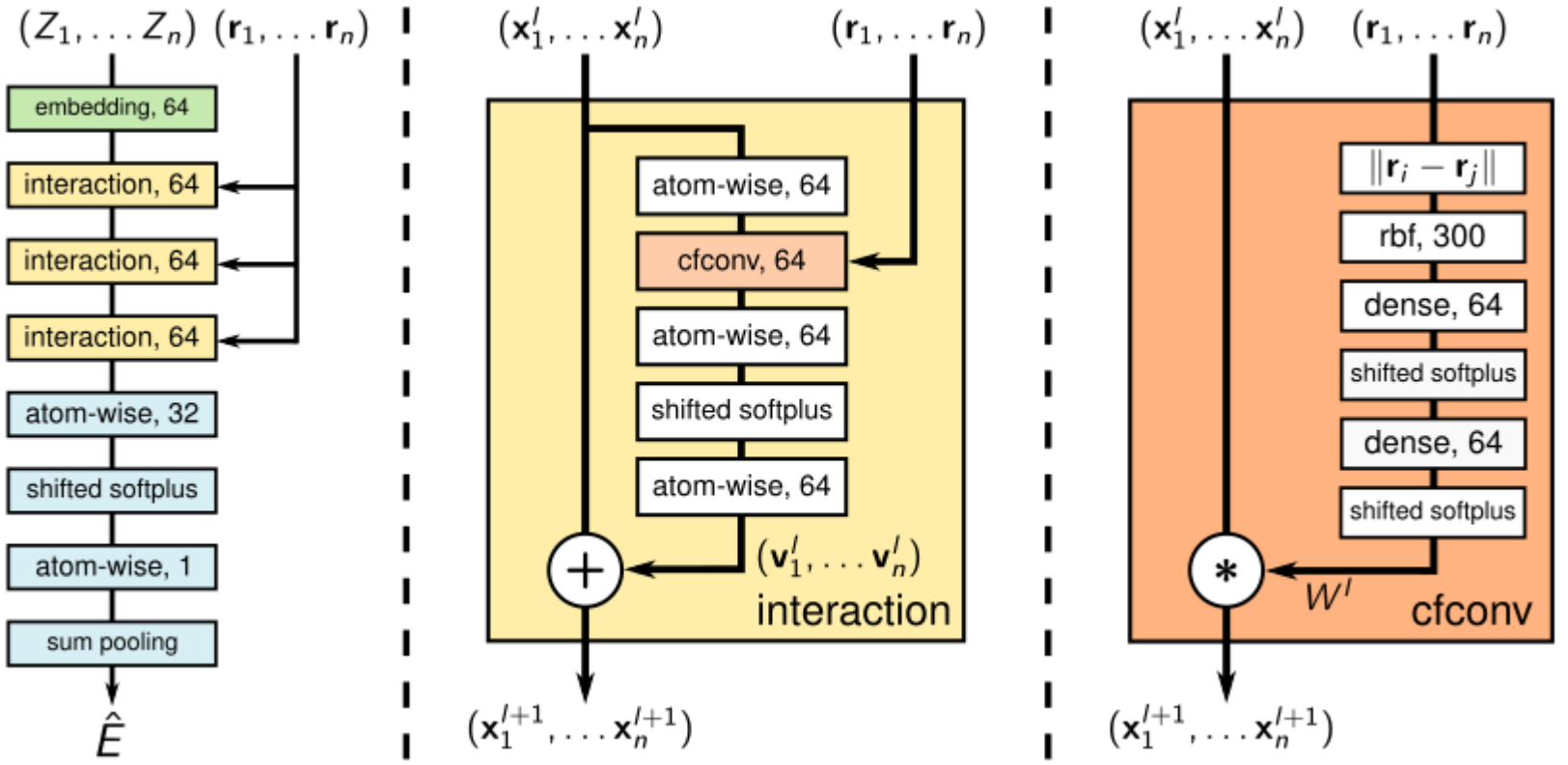
**Challenge:** how to predict an exponentially large number of molecules from finite examples?



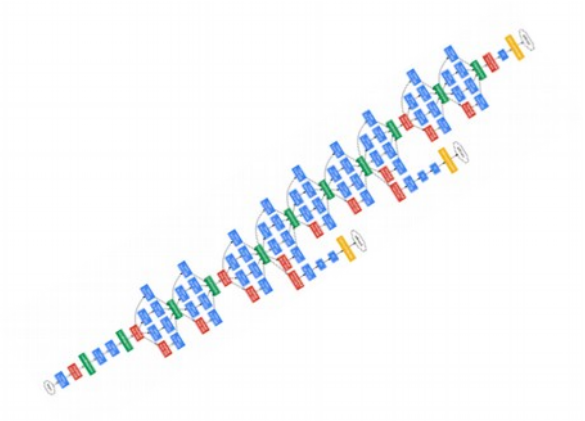

**[Schütt'17]:** DTNN neural network models relations between local chemical environments, and compute the energy progressively.

# Deep Nets for Quantum Chemical Predictions




**[Schütt'17]**

# Deep Neural Networks: Wrap-Up

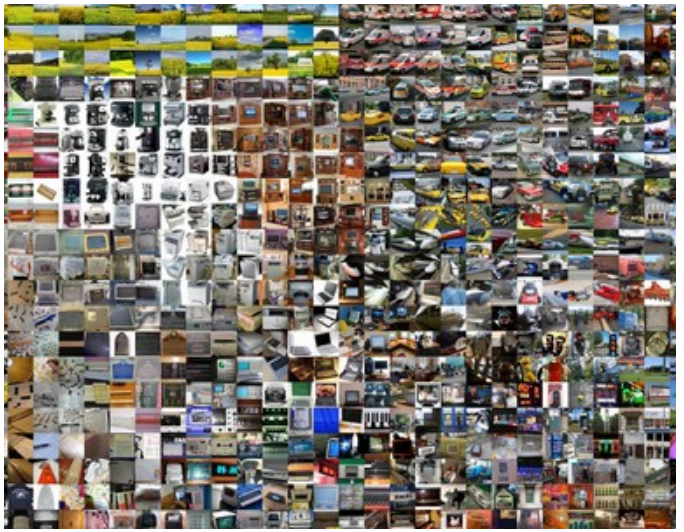


neural nets enable

**big models**


GPUs + fast algorithms enable

**training** these models



**What about the training data?**

IMAGENET

14,197,122 images, 21841 synsets indexed

Explore Download Challenges Publications CoolStuff About

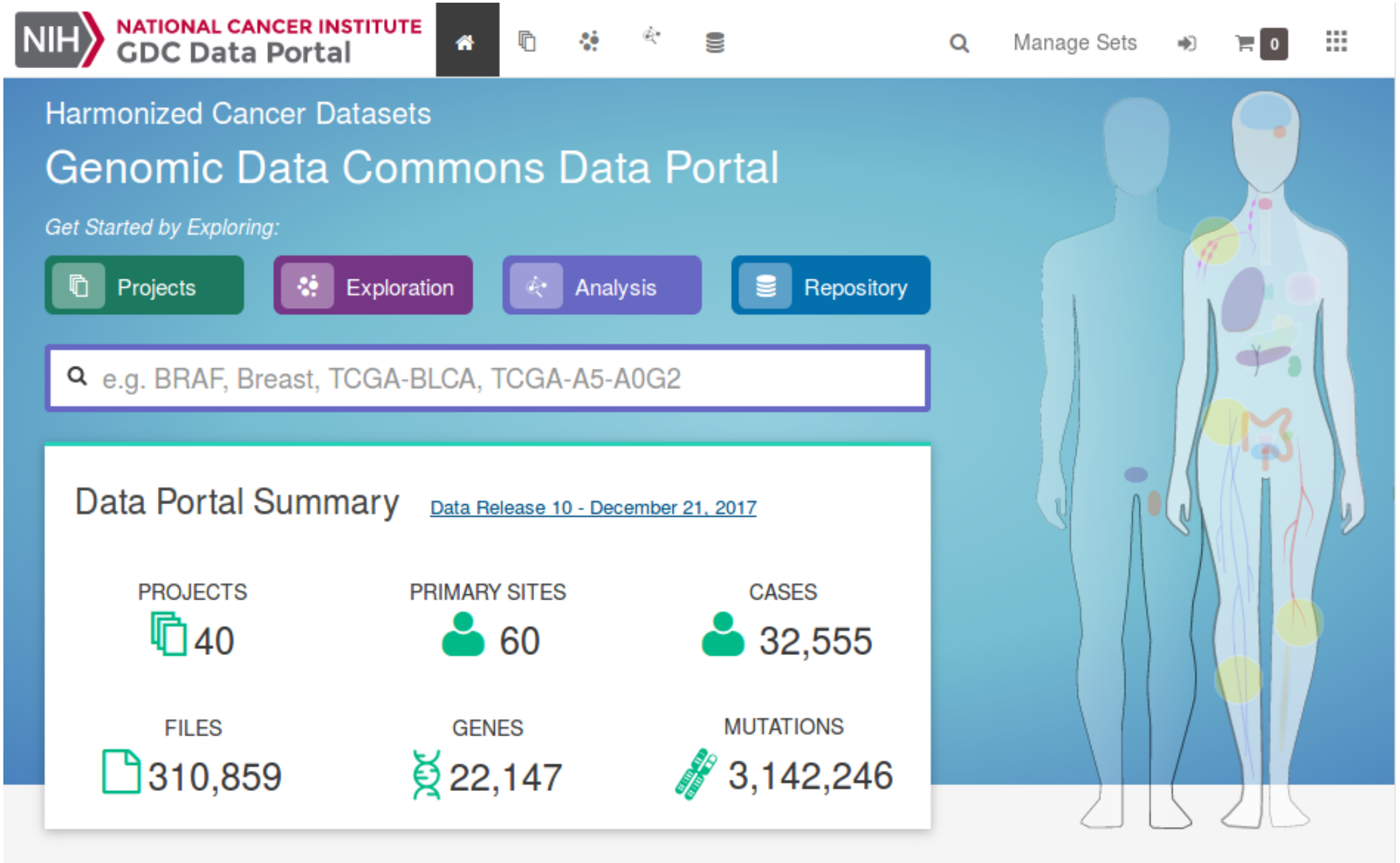
Not logged in. Login | Signup

**ImageNet** is an image database organized according to the WordNet hierarchy (currently only the nouns), in which each node of the hierarchy is depicted by hundreds and thousands of images. Currently we have an average of over five hundred images per node. We hope ImageNet will become a useful resource for researchers, educators, students and all of you who share our passion for pictures.

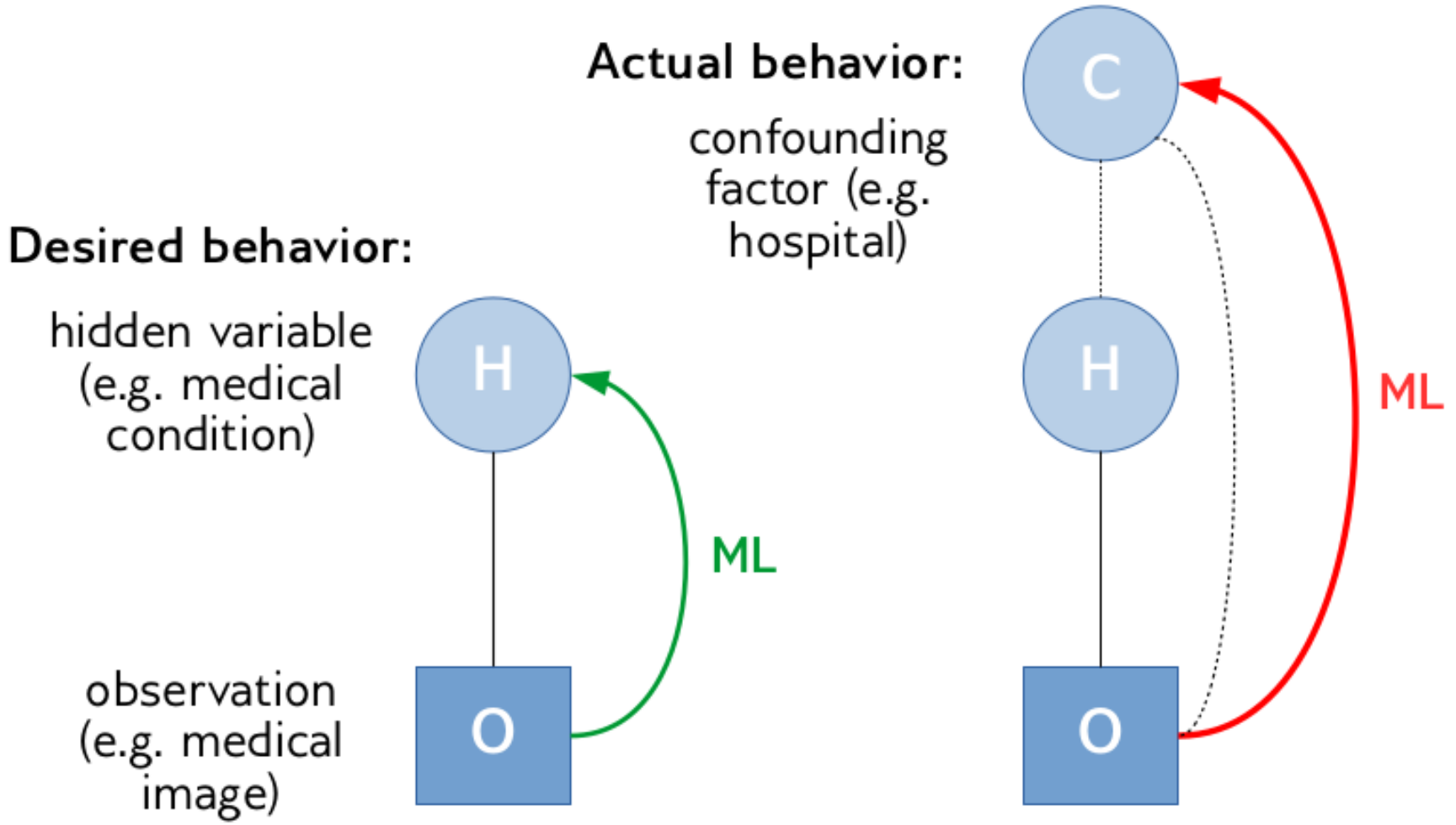
Click here to learn more about ImageNet, Click here to join the ImageNet mailing list.

What do these images have in common? *Find out!*

Check out the ImageNet Challenge on Kaggle!

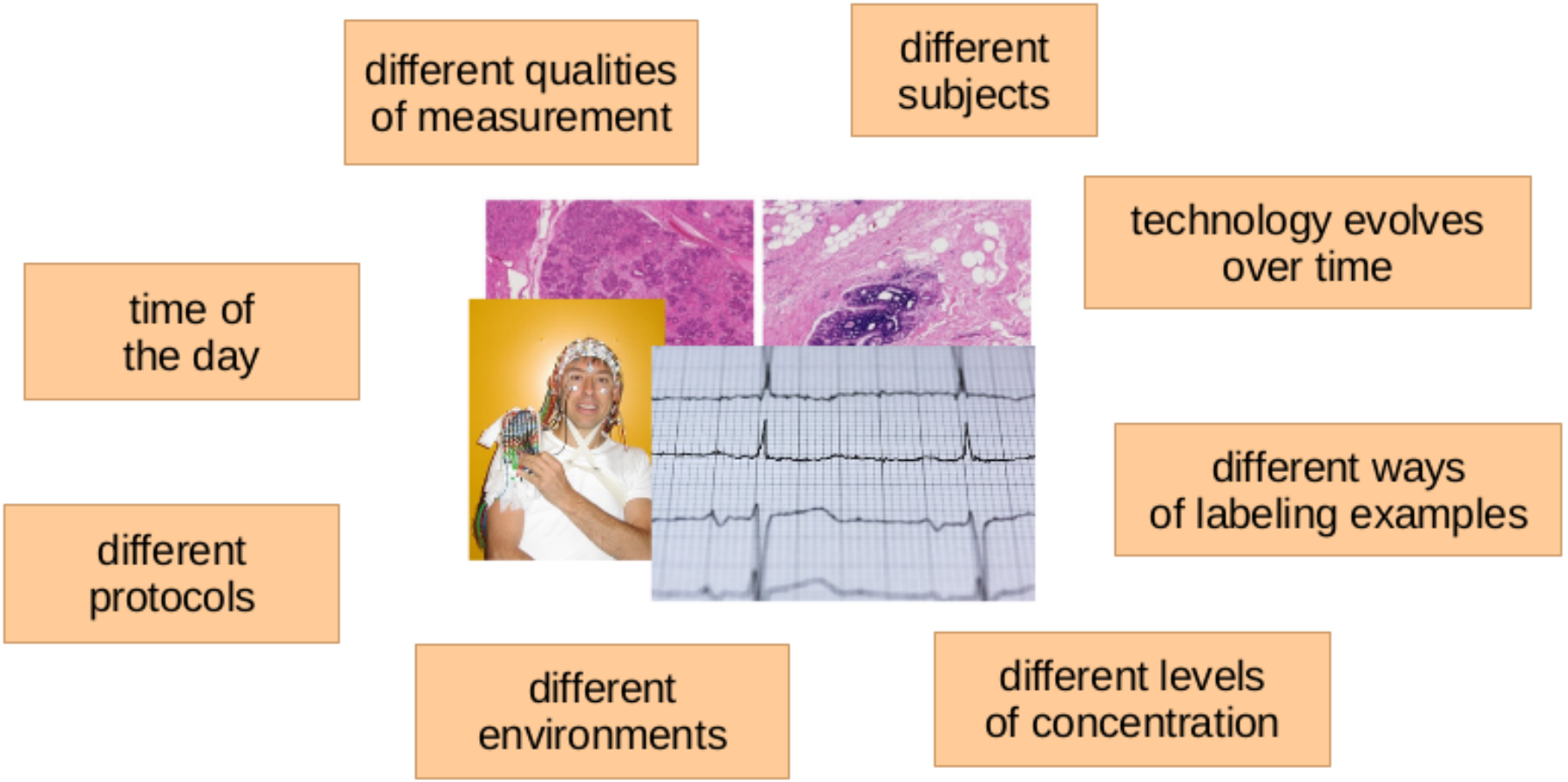

NIH
NATIONAL CANCER INSTITUTE
GDC Data Portal
Manage Sets
0
Harmonized Cancer Datasets
Genomic Data Commons Data Portal
Get Started by Exploring:
Projects
Exploration
Analysis
Repository
e.g. BRAF, Breast, TCGA-BLCA, TCGA-A5-A0G2
Data Portal Summary
Data Release 10 - December 21, 2017
PROJECTS
40
PRIMARY SITES
60
CASES
32,555
FILES
310,859
GENES
22,147
MUTATIONS
3,142,246

# A Common Problem: Confounding Factors

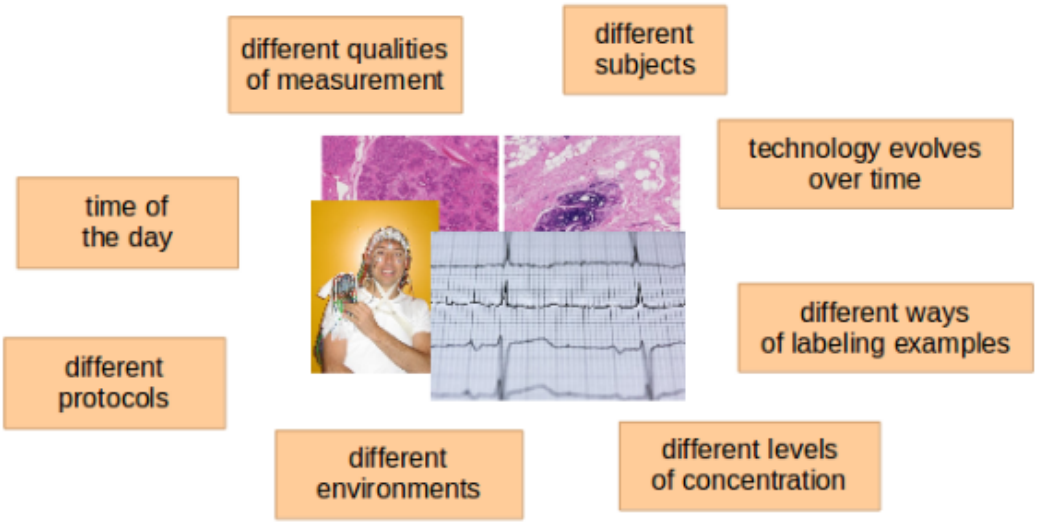


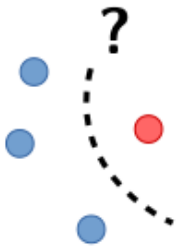
# “Big Data” and “Big Confusion”

**Observation:** Many confounding factors in practice.



# Avoiding “Big Confusion”





Give up on big data, use a controlled setting, and make careful use of “little” data.

Inspect what the big data model has learned.

# Big Data ML “Ansatz”

$$N \to \infty \quad \Rightarrow \quad E_{\text{train}}(f) \approx E(f)$$

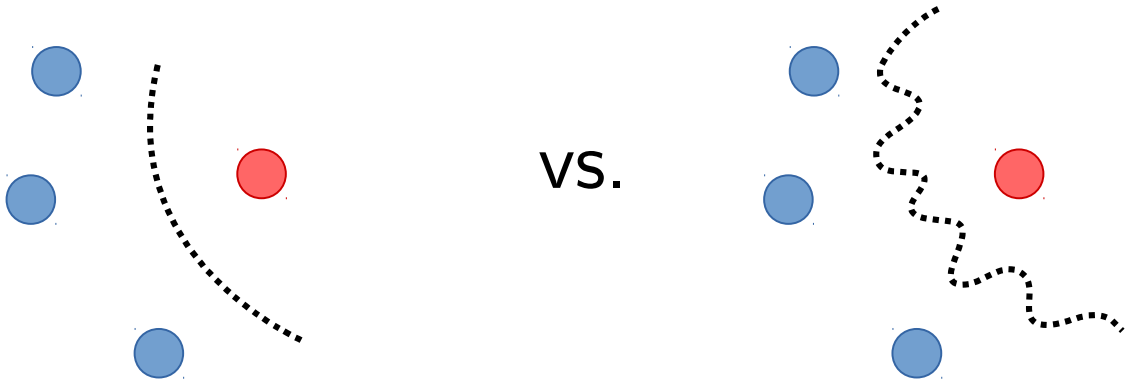
All the data we need is there. All we need to do is to optimize the model *f* quickly and to make sure that *f* is big enough.

# Deep Learning with “Little” Data

Let’s minimize:

$$E_{\text{train}}(f) = \frac{1}{N} \sum_{i=1}^{N} \| f(x_i) - y_i \|^2 \qquad N \neq \infty$$

**Question:** If several functions $f$ reach this goal, which one to select?



# Occam's Razor

*"Among competing hypotheses, the one with the fewest assumptions should be selected."*

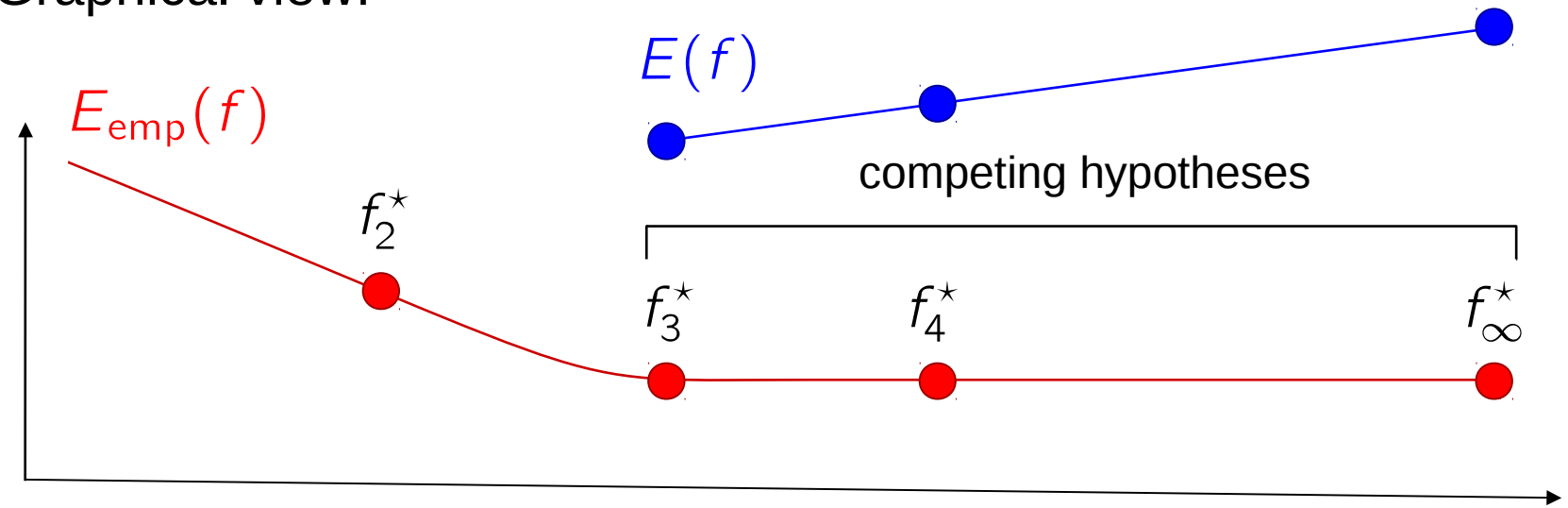
Domingos (1998): "Occam's two Razors: The Sharp and the Blunt":

*Given two models with the same training-set error the simpler one should be preferred because it is likely to have lower generalization error."*

# Occam's Razor

*"Among competing hypotheses, the one with the fewest assumptions should be selected."*

Graphical view:



# Complexity and Generalization Error

**Generalization bound [Vapnik'95]:**

Let $h$ denote the VC-dimension of $\mathcal{F}$. The true error $E[f]$ (with $f \in \mathcal{F}$) is upper-bounded as:

$$E[f] \leq E_{\text{emp}}[f] + \sqrt{\frac{h\left(\log \frac{2N}{h} + 1\right) - \log(\eta/4)}{N}}$$

with probability $1 - \eta$.

**Question:**

- Can we compute the VC-dimension $h$ ?

# VC-Dimension of Various Models

1. Model with a single parameter:

$$f(x;\alpha) = \text{sign}(\sin(\alpha x)) \qquad h = \infty$$

complexity $\neq$ parameters

2. Linear models:

$$f(\boldsymbol{x}, (\boldsymbol{w}, b)) = \text{sign}(\boldsymbol{w}^\top \boldsymbol{x} + b) \qquad h = d + 1$$

# VC-Dimension of Various Models

1. Model with a single parameter:

$$f(x;\alpha) = \text{sign}(\sin(\alpha x)) \qquad h = \infty$$

complexity $\neq$ parameters

2. Linear models:

$$f(\boldsymbol{x}, (\boldsymbol{w}, b)) = \text{sign}(\boldsymbol{w}^\top \boldsymbol{x} + b) \qquad h = d + 1$$

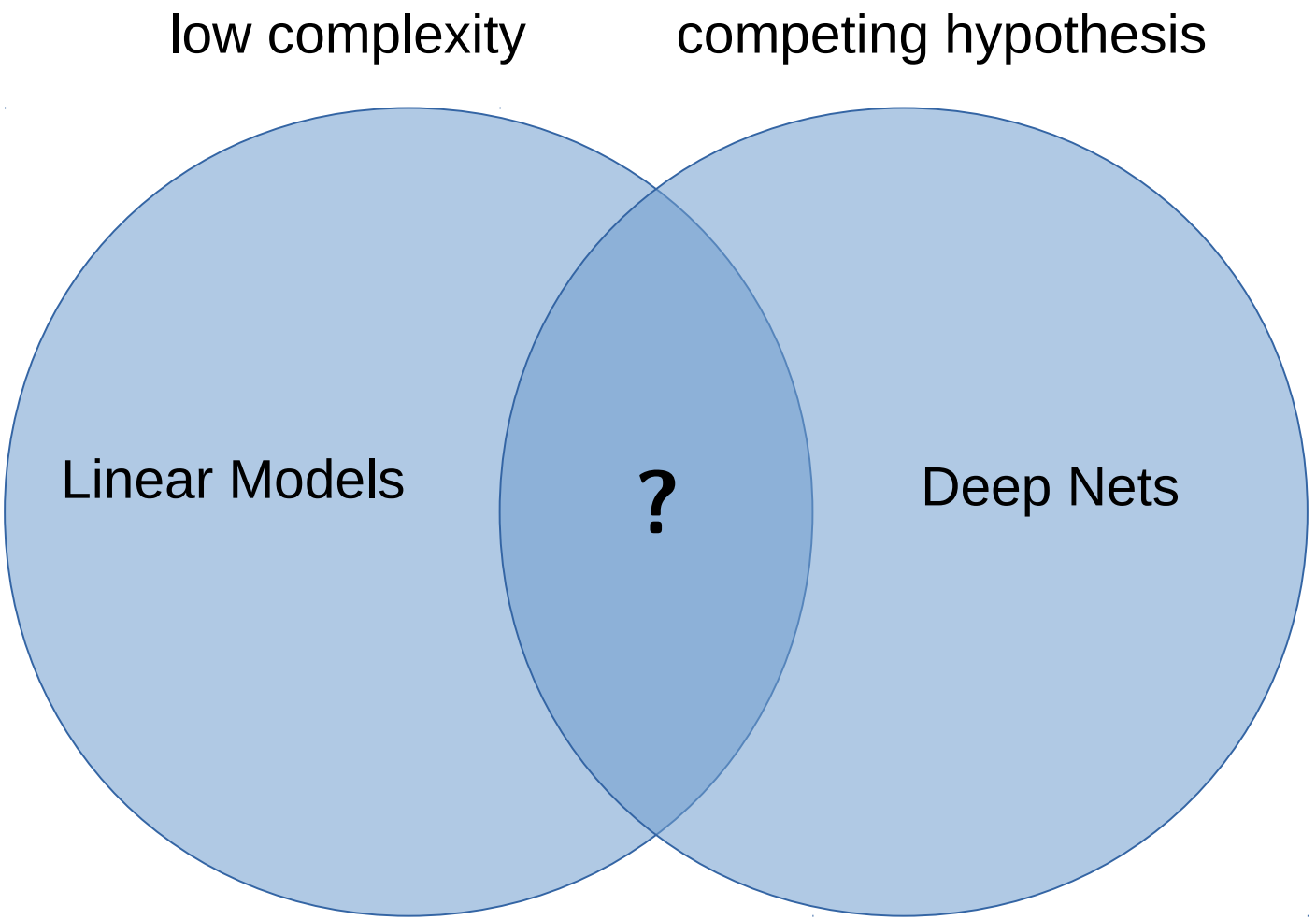
---

*"Among competing hypotheses, the one with the fewest assumptions should be selected."*

---

**Question:** Is the linear model a competing hypothesis?

# Low Complexity and Competing Hypothesis



# VC-Dimension of Various Models

1. Model with a single parameter:

$$f(x;\alpha) = \text{sign}(\sin(\alpha x)) \qquad h = \infty$$

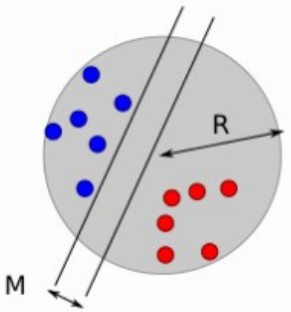
complexity ≠ parameters

2. Linear models:

$$f(\boldsymbol{x}, (\boldsymbol{w}, b)) = \text{sign}(\boldsymbol{w}^\top \boldsymbol{x} + b) \qquad h = d + 1$$

3. Linear models + large margin:

$$f(\boldsymbol{x}, (\boldsymbol{w}, b)) = \text{sign}(\boldsymbol{w}^\top \boldsymbol{x} + b)$$

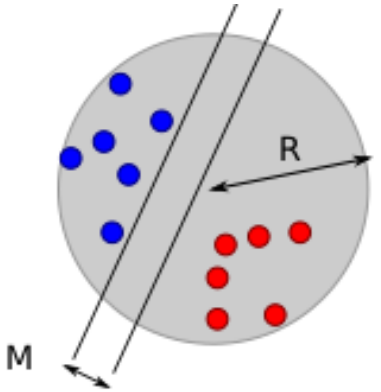



$$h = \min\left\{d + 1\,,\, 4\frac{R^2}{M^2} + 1\right\}$$

margin is key!

# Large Margin → Insensitivity to Perturbation

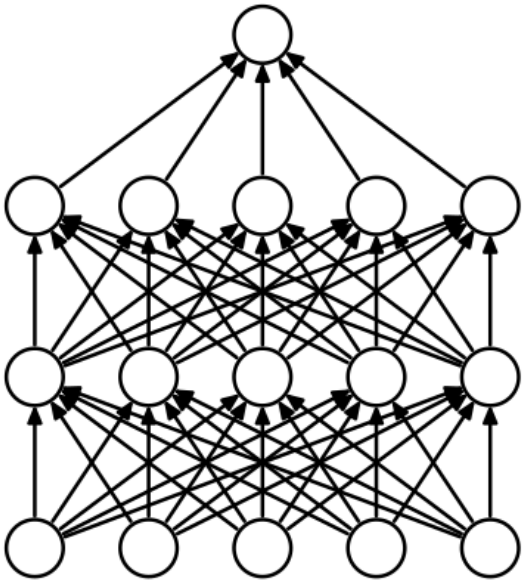
- Linear Models: large-margin or ridge regularization makes the decision insensitive to small perturbation in the input space or feature space.



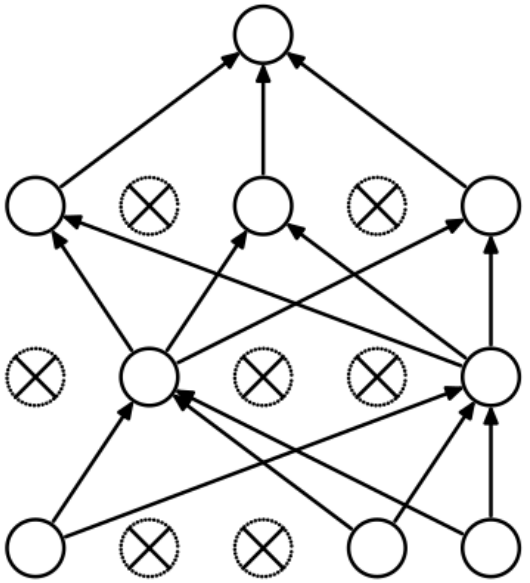

- Neural networks: **Dropout algorithm** (Srivastava'14): replicates this insensitivity to perturbation in the context of neural network, by training the model while randomly turning on and off some neurons and input variables.

# Example: The Dropout Method (Srivastava'14)

- Neural networks: **Dropout algorithm** (Srivastava'14): replicates this insensitivity to perturbation in the context of neural network, by training the model while randomly turning on and off some neurons and input variables.



(a) Standard Neural Net

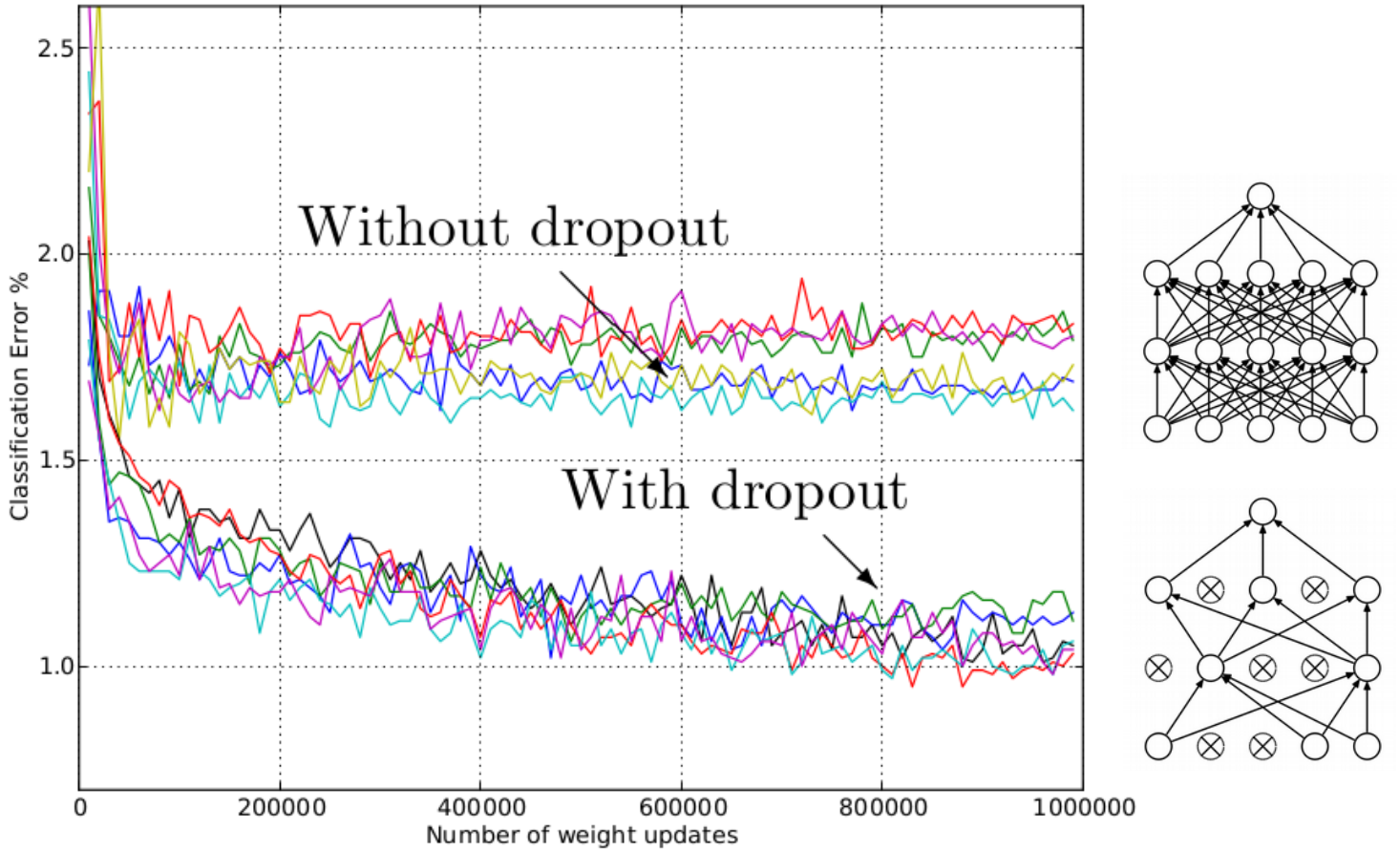


(b) After applying dropout.

(Source: Srivastava et al. 2014: Dropout: A simple way to prevent neural networks from overfitting).

# Example: The Dropout Method (Srivastava'14)

Results on MNIST dataset:

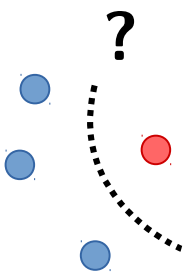



(source: Srivastava et al. 2014: Dropout: A simple way to prevent neural networks from overfitting).

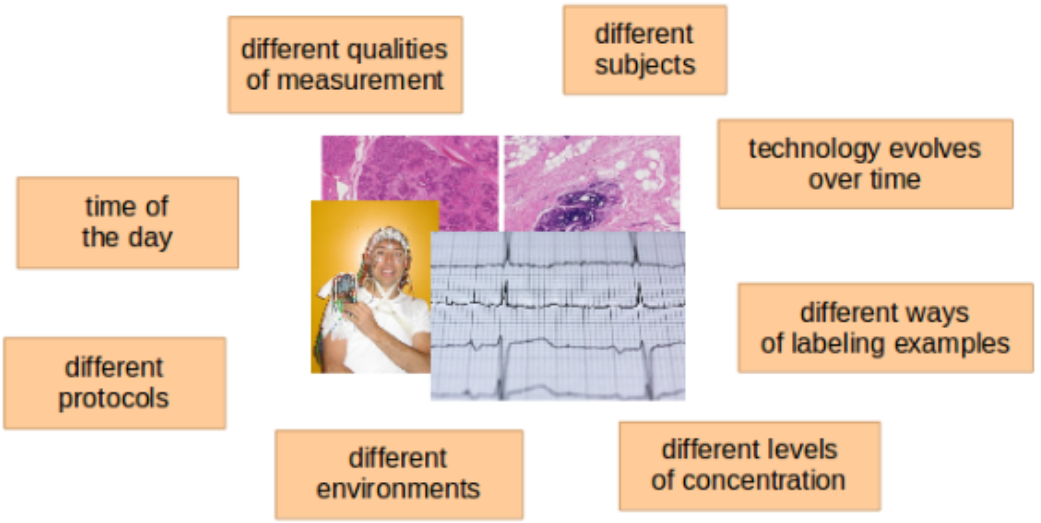
Deep Neural Networks

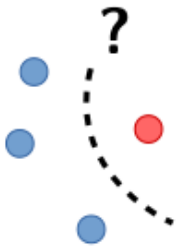
# Week # 4

## Learning with “Little Data”




# Avoiding “Big Confusion”





Give up on big data, use a controlled setting, and make careful use of “little” data.

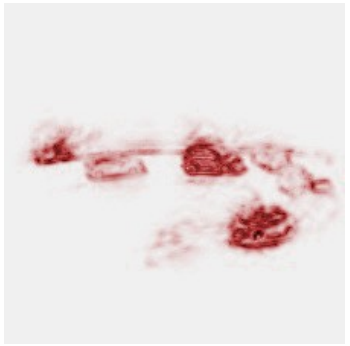
Inspect what the big data model has learned.

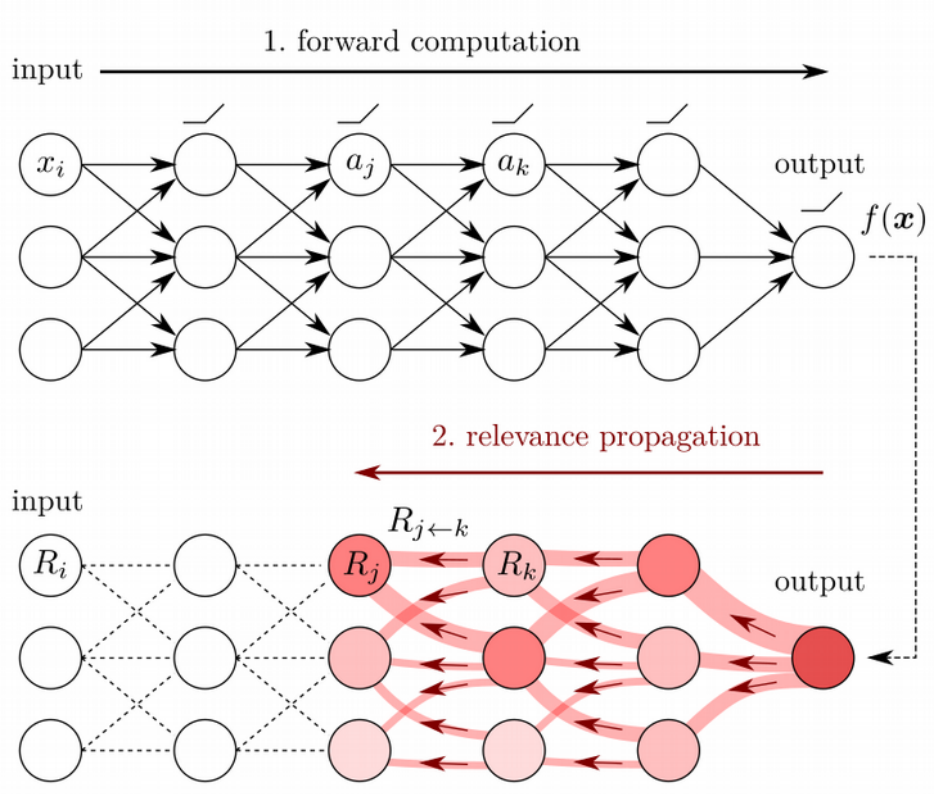
# Interpreting Deep Networks


- Growing research area in machine learning.
- Aims to present the reasoning of the trained ML system to the human, so that he can verify it and understand it better.

# Techniques of Interpretability

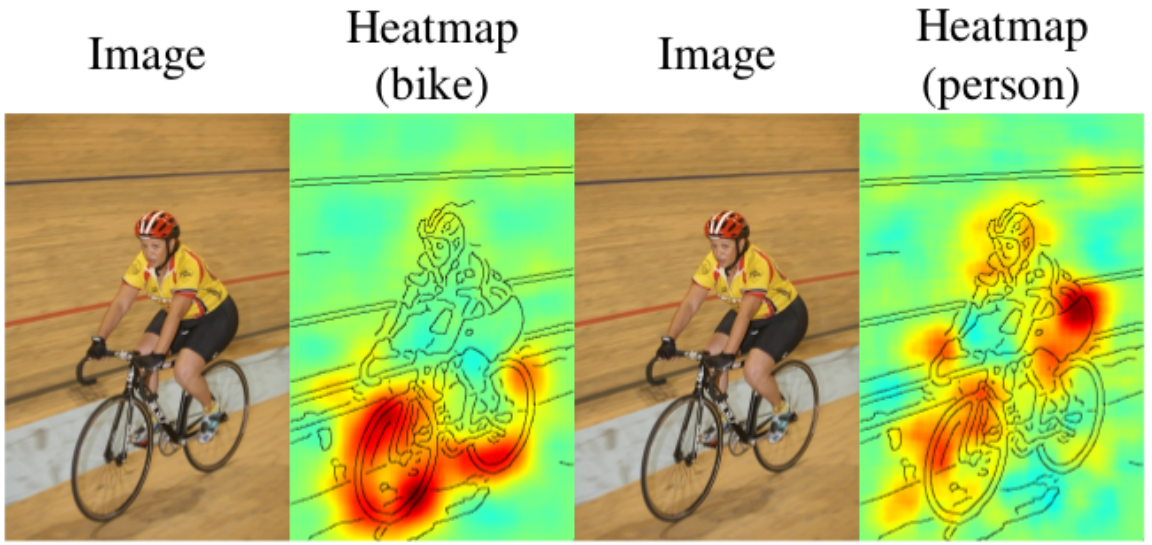
**Example:** Layer-wise relevance propagation (LRP) [Bach'15]

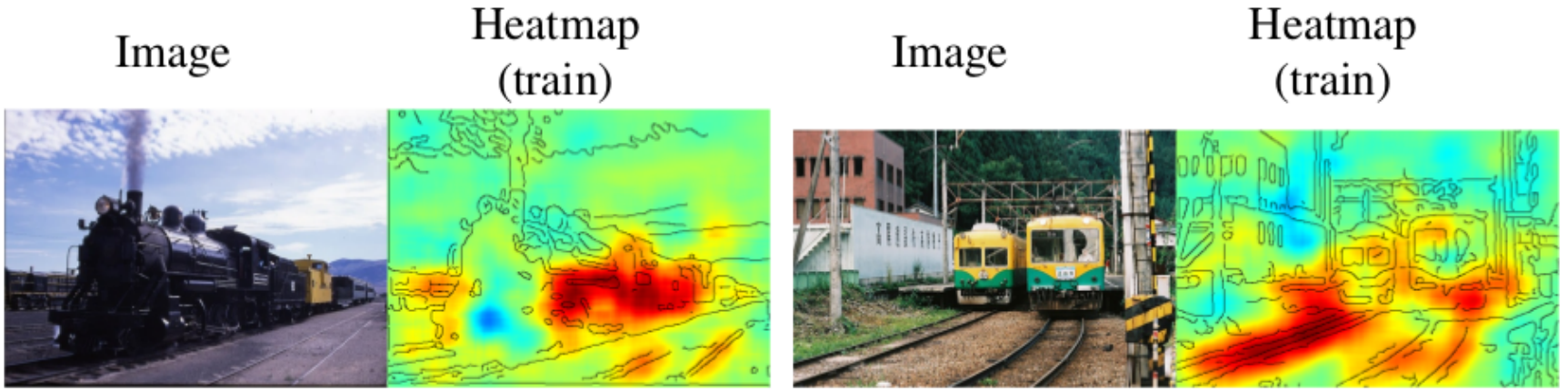





$$R_j = \sum_k \left( \alpha \frac{a_j w_{jk}^+}{\sum_j a_j w_{jk}^+} - \beta \frac{a_j w_{jk}^-}{\sum_j a_j w_{jk}^-} \right) R_k$$

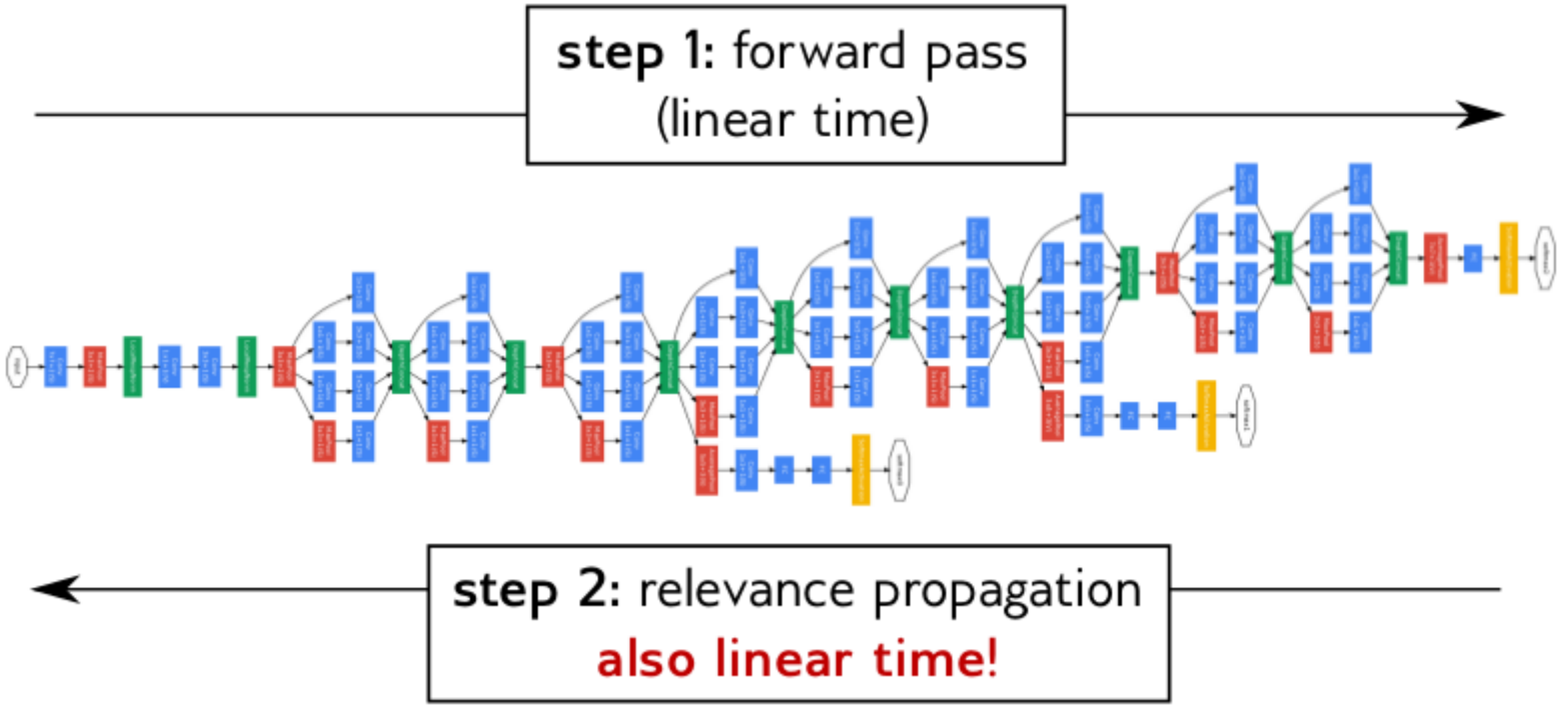
# LRP applied to Fisher Networks (3 layers)




Images from **Lapuschkin'16**

# Cost of Layer-Wise Relevance Propagation




→ Can be used to explain state-of-the-art models

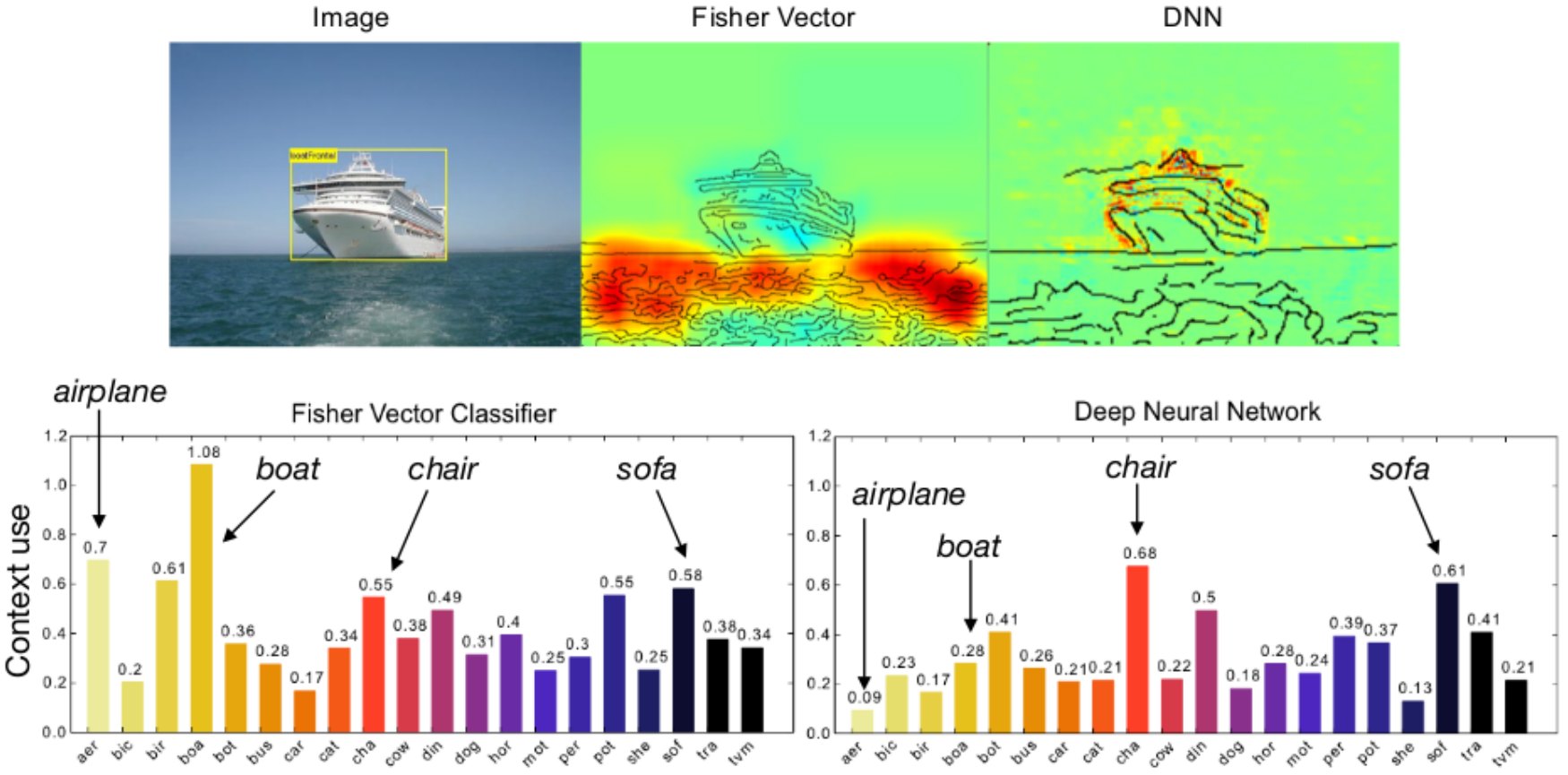
# LRP applied to GoogleNet (22-layers)


Clear focus on the object.

Ignores the background.

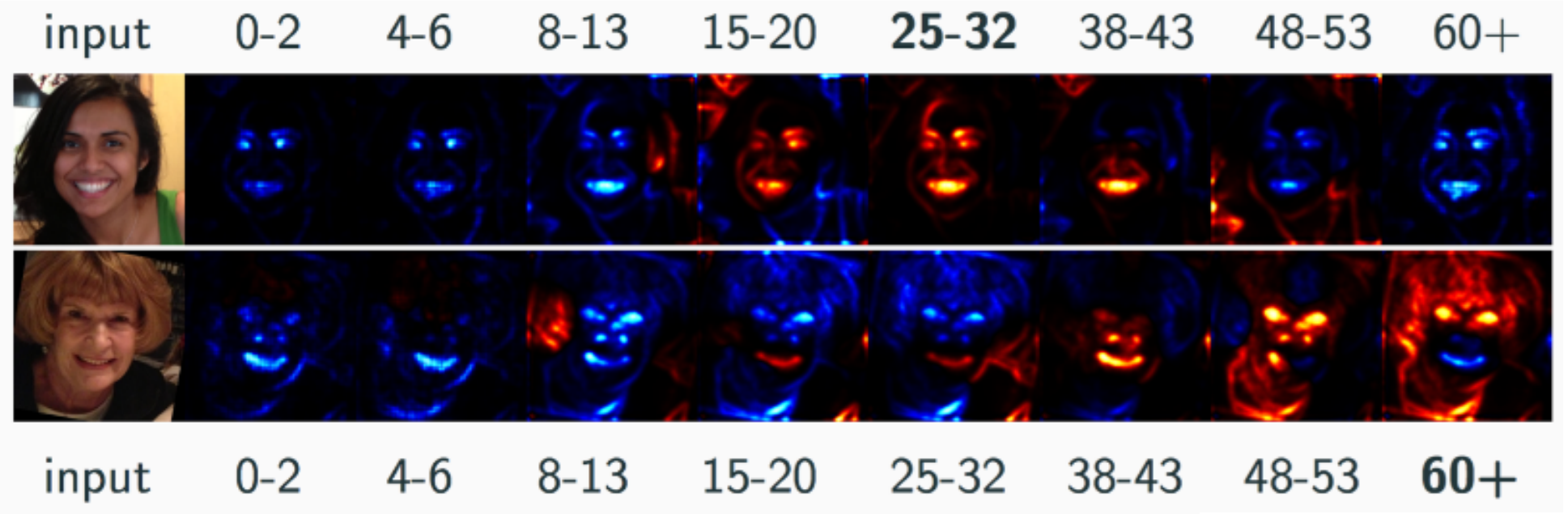
# LRP for Understanding the Model

**Question:** We know where the network should look at. Does the model solves the problem correctly?



# LRP for Understanding the Model Task

**Question:** Assume the model is the ground truth. Can we learn from the model what makes someone be of a certain age?




*(Lapuschkin et al. 2017)*

Deep Neural Networks

# Week # 8

## Interpreting Models and their Decisions