max planck institut informatik

SIC Saarland Informatics Campus

Elements of Data Science and Artificial Intelligence

Introduction to Artificial Neural Networks and Deep Learning

Bernt Schiele - schiele@mpi-inf.mpg.de

Classification Problem

- E.g.: Image classification:

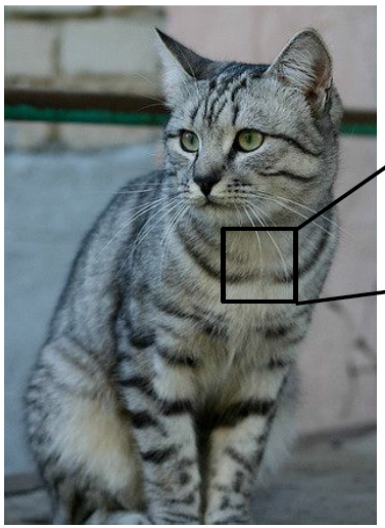
This image by Nikita is licensed under CC-BY 2.0

(assume given set of discrete labels)
{dog, cat, truck, plane, ...}

→ cat

Image Classification

The Problem: Semantic Gap




```
[[105 112 108 111 104  99 106  99  96 103 112 119 104  97  93  87]
 [ 91  98 102 106 104  79  98 103  99 105 123 136 110 105  94  85]
 [ 76  85  90 105 128 105  87  96  95  99 115 112 106 103  99  85]
 [ 99  81  81  93 120 131 127 100  95  98 102  99  96  93 101  94]
 [106  91  61  64  69  91  88  85 101 107 109  98  75  84  96  95]
 [114 108  85  55  55  69  64  54  64  87 112 129  98  74  84  91]
 [133 137 147 103  65  81  80  65  52  54  74  84 102  93  85  82]
 [128 137 144 140 109  95  86  70  62  65  63  63  60  73  86 101]
 [125 133 148 137 110 121 117  94  65  79  80  65  54  64  72  98]
 [127 125 131 147 133 127 126 131 111  96  89  75  61  64  72  84]
 [115 114 109 123 150 148 131 118 113 109 100  92  74  65  72  78]
 [ 89  93  90  97 108 147 131 118 113 114 113 109 106  95  77  80]
 [ 63  77  86  81  77  79 102 123 117 115 117 125 125 130 115  87]
 [ 62  65  82  89  78  71  80 101 124 126 119 101 107 114 131 119]
 [ 63  65  75  88  89  71  62  81 120 138 135 105  81  98 110 118]
 [ 87  65  71  87 106  95  69  45  76 130 126 107  92  94 105 112]
 [118  97  82  86 117 123 116  66  41  51  95  93  89  95 102 107]
 [164 146 112  80  82 120 124 104  76  40  45  66  80 101 102 109]
 [157 170 157 120  93  86 114 132 112  97  69  55  70  82  69  94]
 [130 128 134 161 139 100 109 118 121 134 114  87  65  53  69  86]
 [128 112  96 117 150 144 120 115 104 107 102  93  87  81  72  79]
 [123 107  96  86  83 112 153 149 122 109 104  75  80 107 112  99]
 [122 121 102  80  82  86  94 117 145 148 153 102  58  78  92 107]
 [122 164 148 103  71  56  78  83  93 103 119 139 102  61  69  84]]
```

What the computer sees

An image is just a big grid of numbers between [0, 255]:

e.g. 800 x 600 x 3
(3 channels RGB)

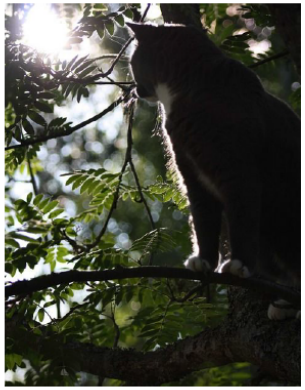
Image Classification

Challenges: Illumination

This image is CC0 1.0 public domain

This image is CC0 1.0 public domain

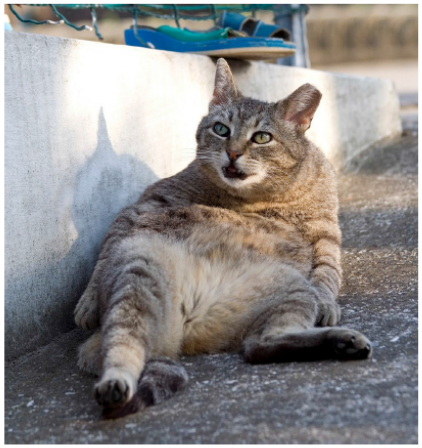
This image is CC0 1.0 public domain



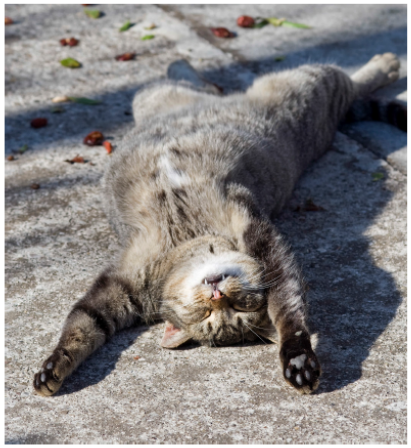
This image is CC0 1.0 public domain

Image Classification

Challenges: Deformation

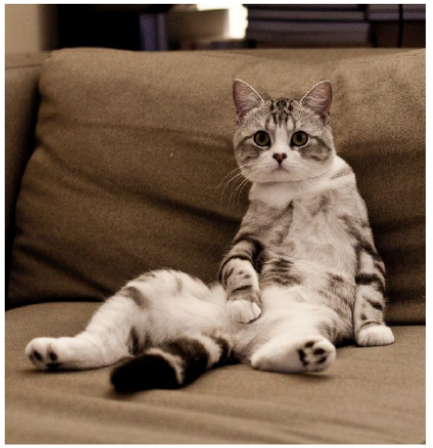


This image by Umberto Salvagnin is licensed under CC-BY 2.0



This image by Umberto Salvagnin is licensed under CC-BY 2.0

This image by sare bear is licensed under CC-BY 2.0



This image by Tom Thai is licensed under CC-BY 2.0

Image Classification

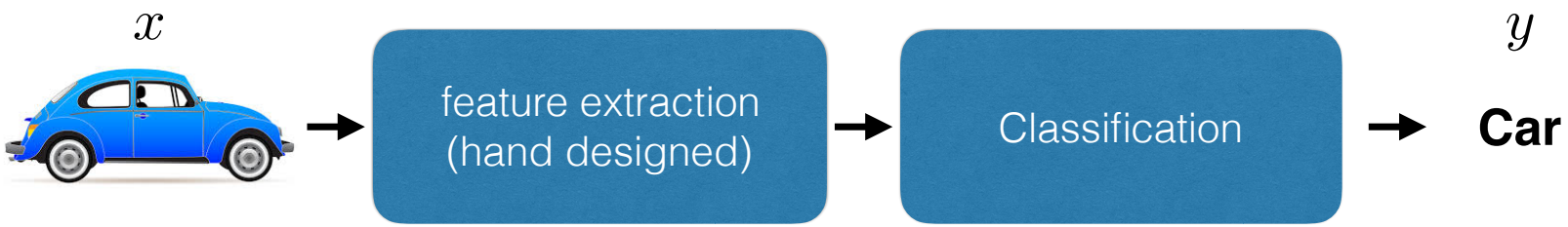
Challenges: Occlusion

This image is CC0 1.0 public domain

This image is CC0 1.0 public domain

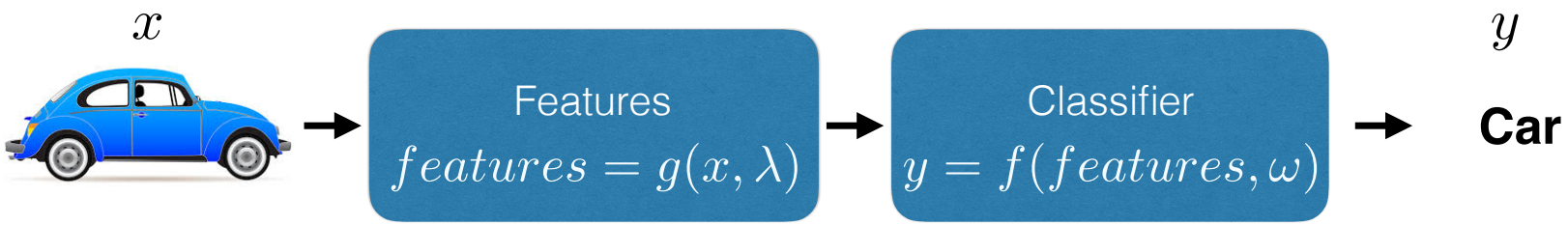
This image by jonsson is licensed under CC-BY 2.0

Traditional Approach to Classification



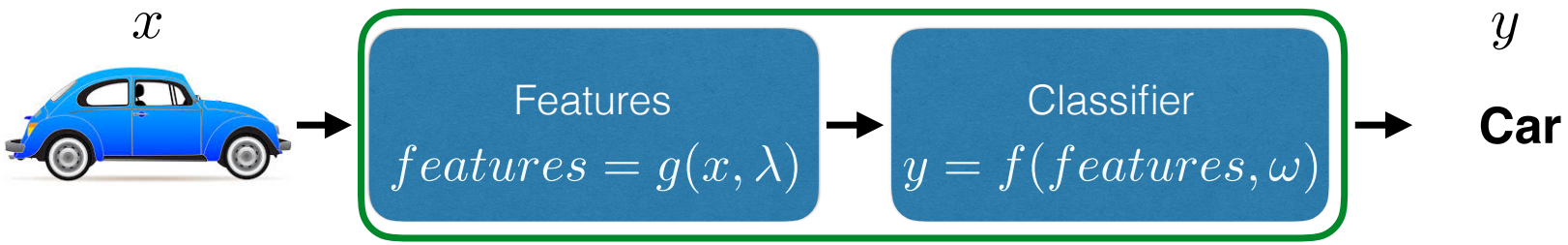
- Feature extraction
  - features should be (ideally) invariant to illumination, deformation, occlusion, …
  - often hand designed and fixed
  - features
    - might be too general (not task-specific enough)
    - might be too specific (does not generalize to other tasks)
- How to achieve best classification performance
  - more complex classifier (e.g. multi-feature, non-linear)?
  - how specialized for the task?

Deep Learning Approach for Classification: Trainable features



- Parameterized feature extraction
- Features should be
  - efficient to compute
  - efficient to train (differentiable)

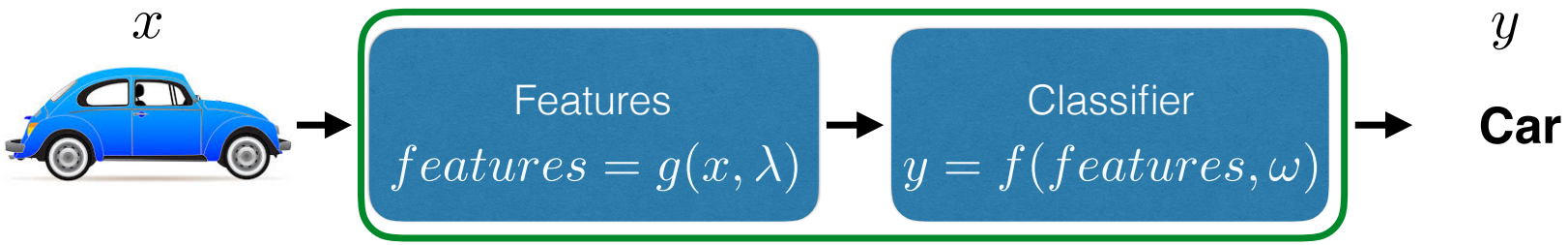
Deep Learning: Joint Training of all Parameters



"End-to-End" System

- Parameterized feature extraction
- Features should be
  - efficient to compute
  - efficient to train (differentiable)
- **Joint** training of **feature** extraction and **classification**
  - Feature extraction and classification merge into one pipeline

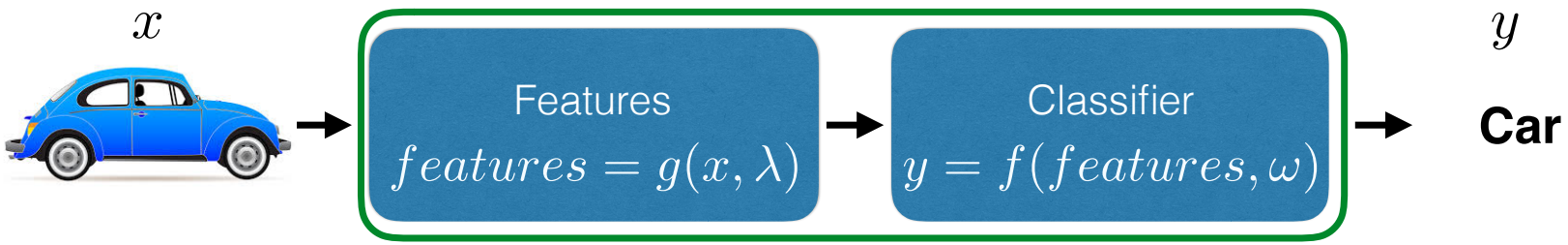
Deep Learning: Joint Training of all Parameters



“End-to-End” System

- All parts are adaptive
- No distinction between feature extraction and classification
- Non linear transformation from input to desired output

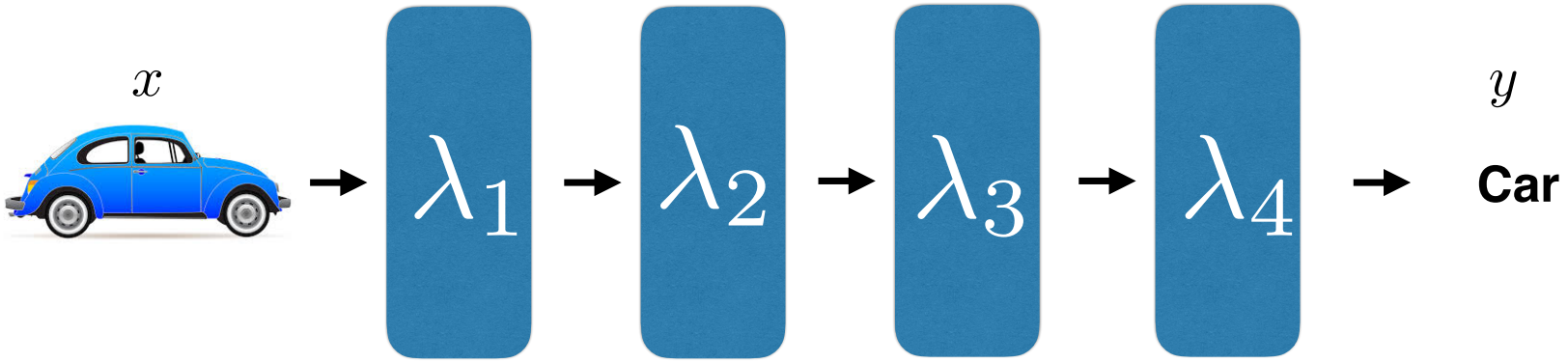
Deep Learning: Complex Functions by Composition



"End-to-End" System

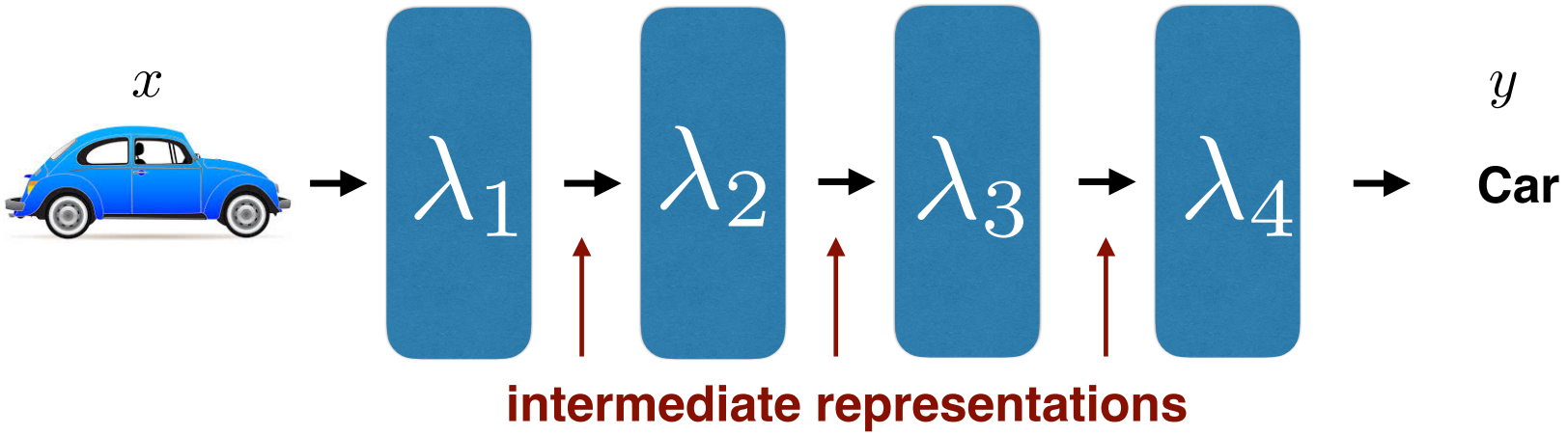
- How can we build such systems?
- What is the parameterization (hypothesis)?
- Composition of simple building blocks can lead to complex systems (e.g. neurons - brain)

Deep Learning: Complex Functions by Composition



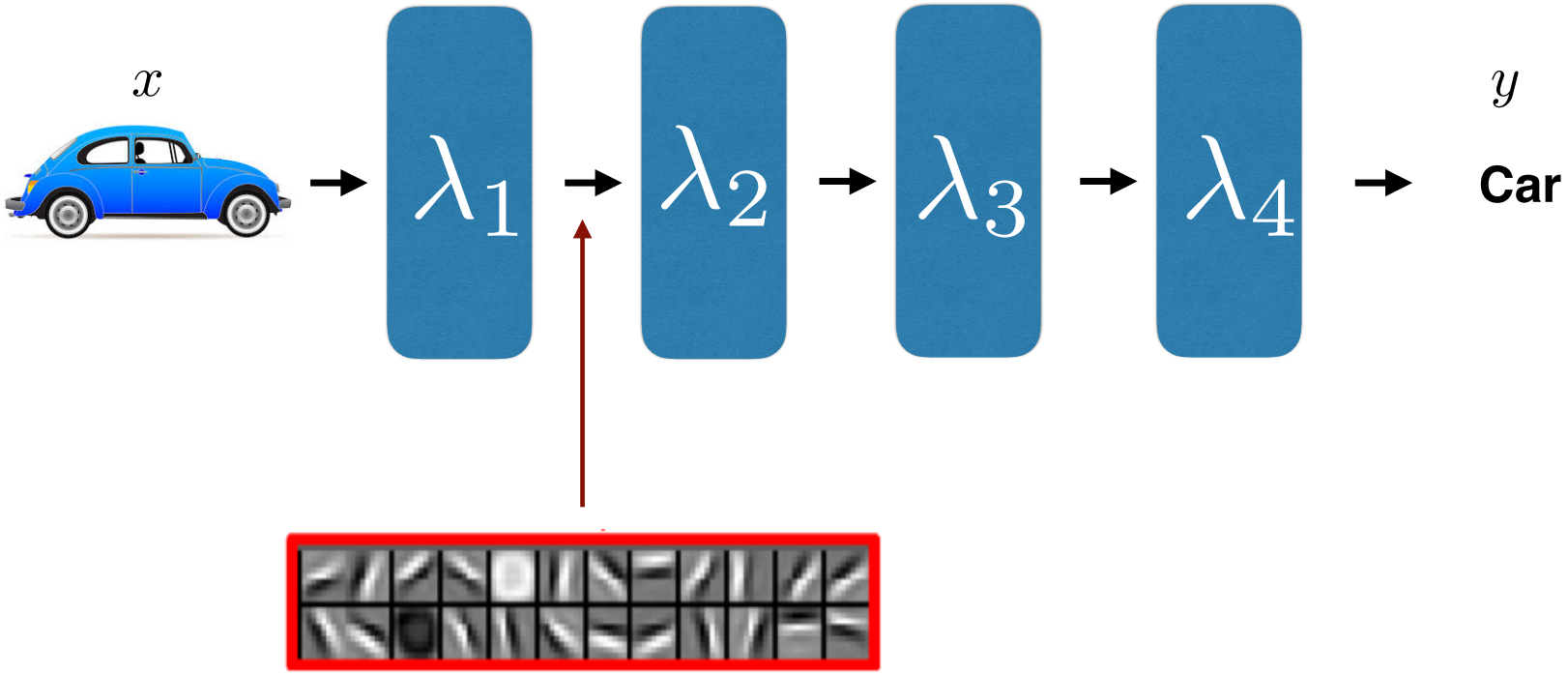
- How can we build such systems?
- What is the parameterization (hypothesis)?
- Composition of simple building blocks can lead to complex systems (e.g. neurons - brain)
- Each block has trainable parameters λ_i

Deep Learning: Complex Functions by Composition



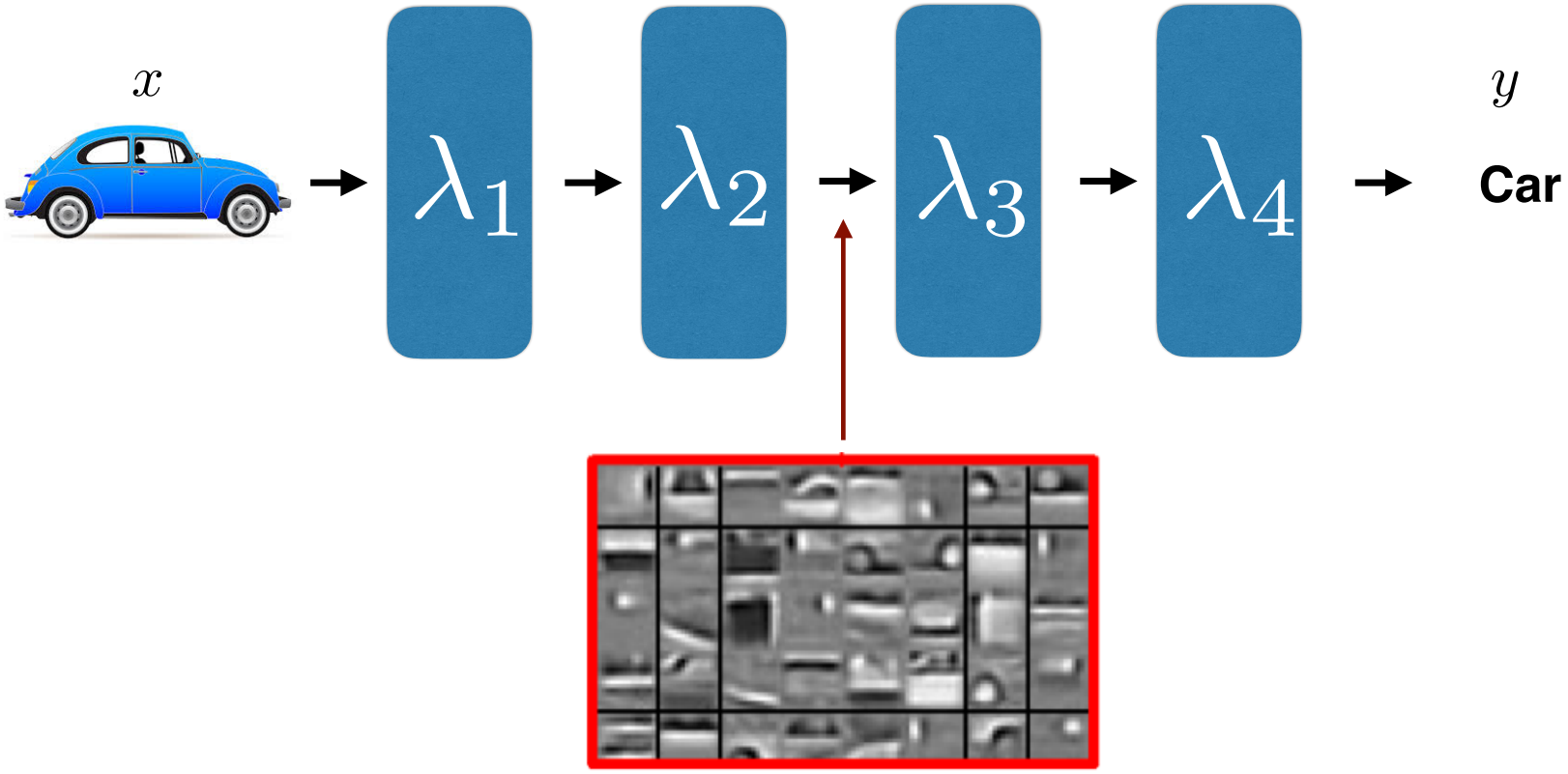
- How can we build such systems?
- What is the parameterization (hypothesis)?
- Composition of simple building blocks can lead to complex systems (e.g. neurons - brain)
- Each block has trainable parameters λ_i

Deep Learning: Complex Functions by Composition



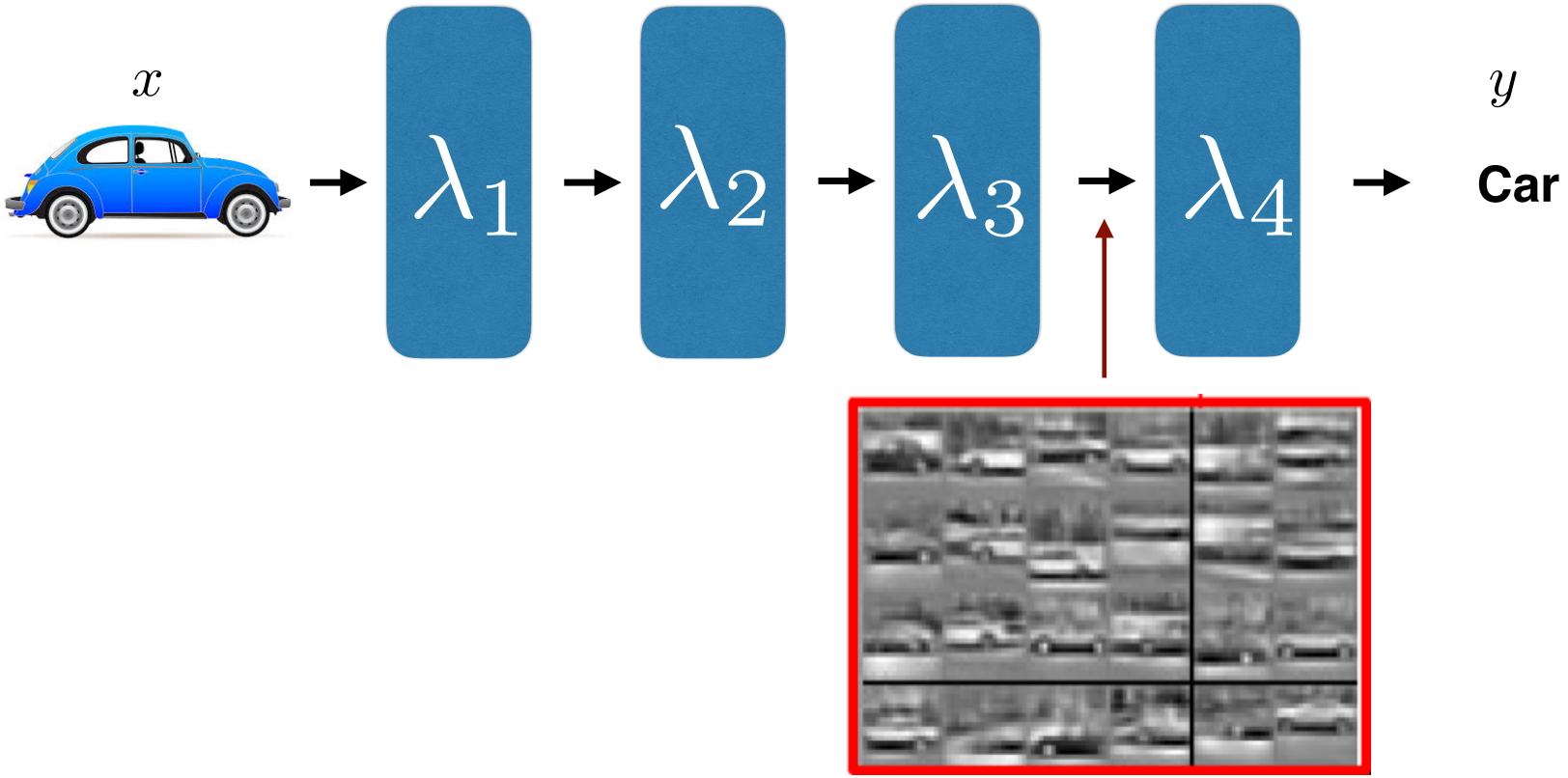
Lee et al. “Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations”

Deep Learning: Complex Functions by Composition



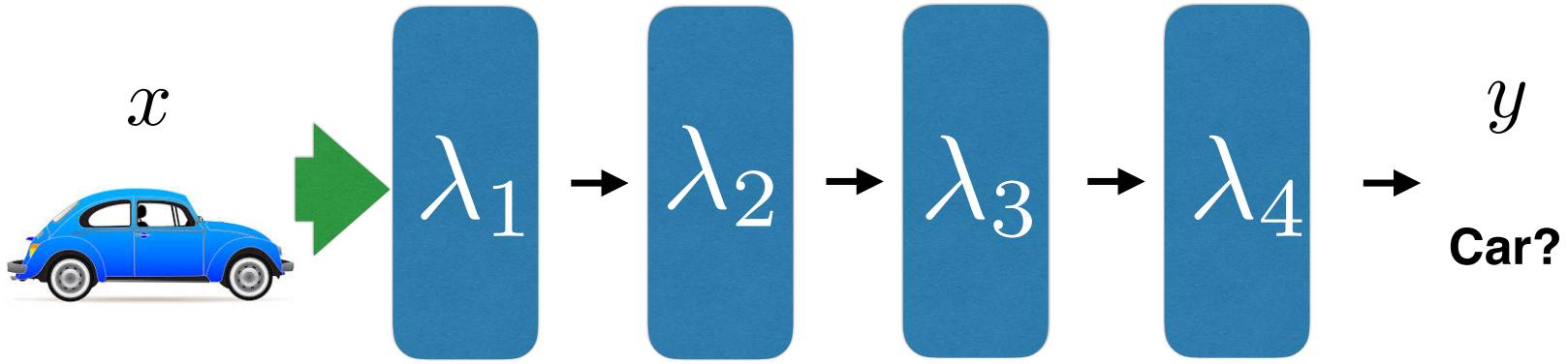
Lee et al. “Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations”

Deep Learning: Complex Functions by Composition



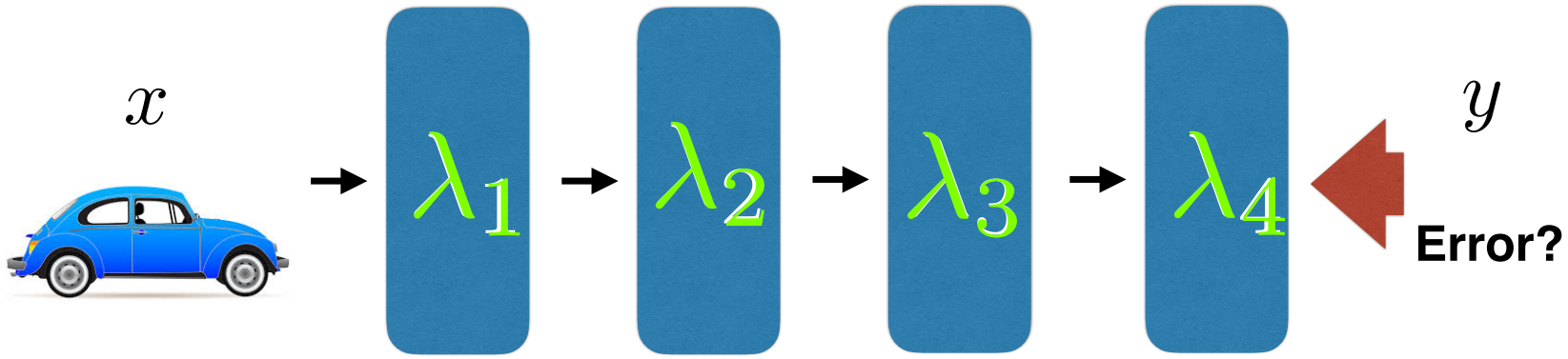
Lee et al. “Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations”

Training: Overview



- Setting
  - generate output y for input x (forward pass)

Training: Overview

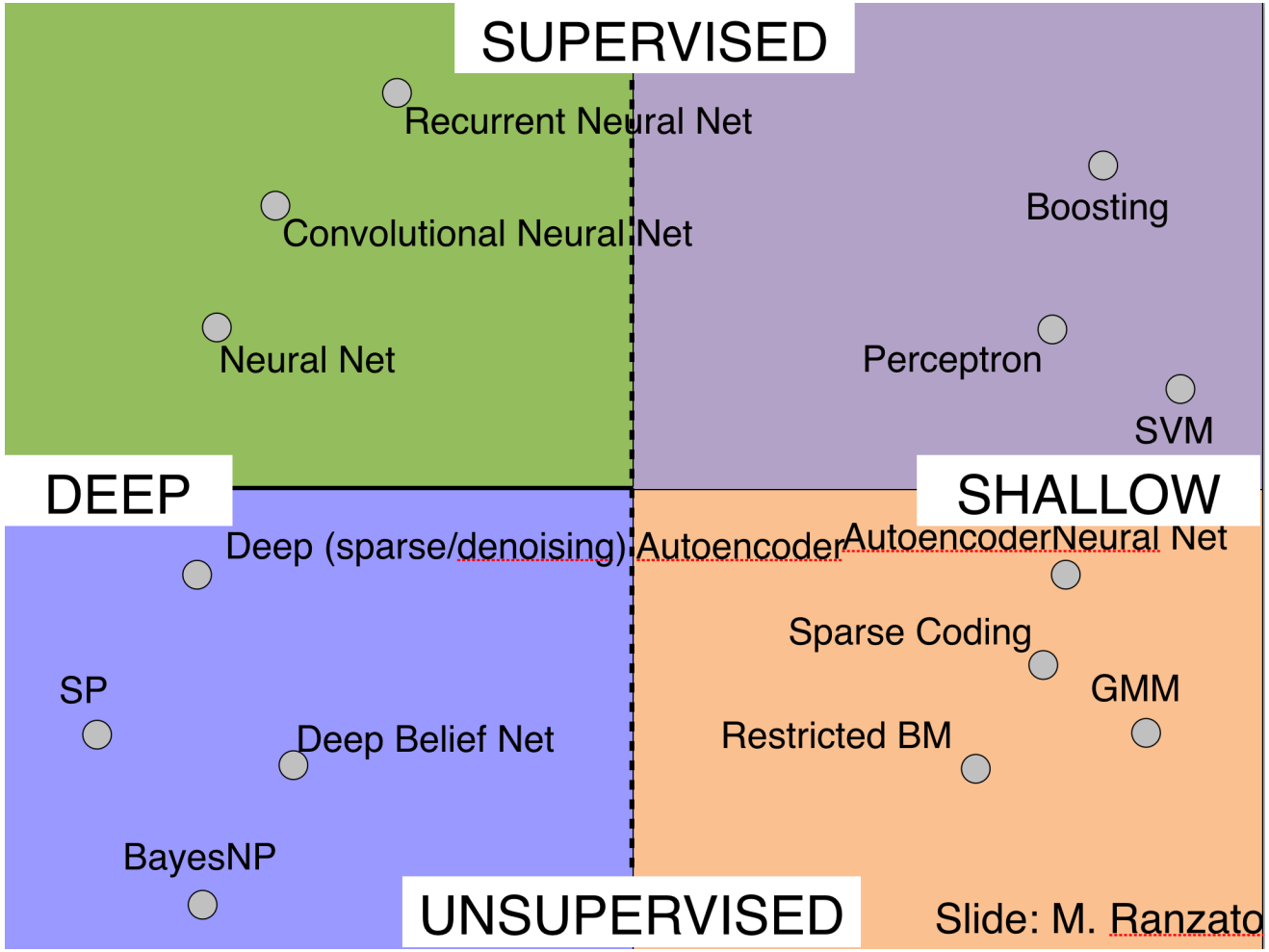


- Setting
  - generate output y for input x (forward pass)
  - when there is an error, propagate error backwards to update weights (error back propagation)

Summary of Main Ideas in Deep Learning

1. **Learning of feature extraction** (across many layers)
2. **Efficient** and **trainable** systems by **differentiable** building blocks
3. Composition of deep architectures via **non-linear modules**
4. "**End-to-End**" training: no differentiation between feature extraction and classification

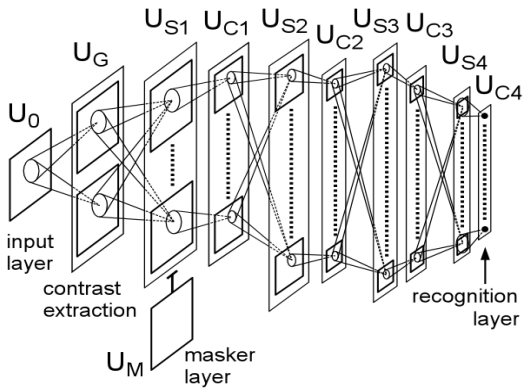
Overview of Deep Learning



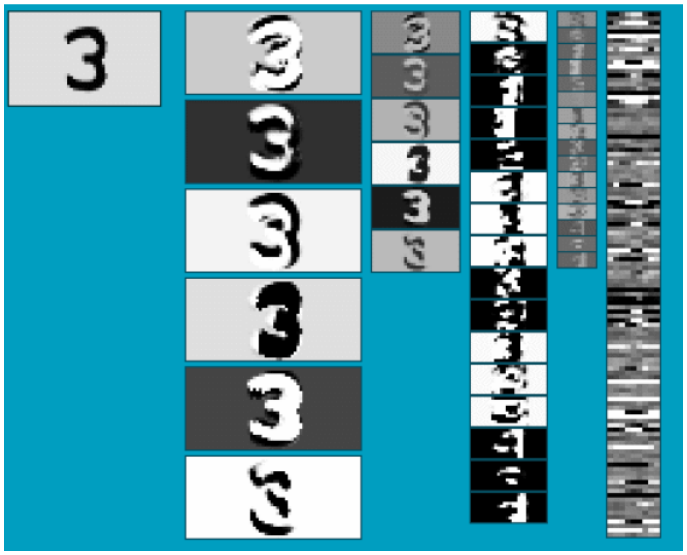
Slide: Y.LeCun

Multistage Hubel & Wiesel Architecture

- [Hubel & Wiesel 1962]
  - simple cells detect local features
  - complex cells “pool” the outputs of simple cells within a retinotopic neighborhood.

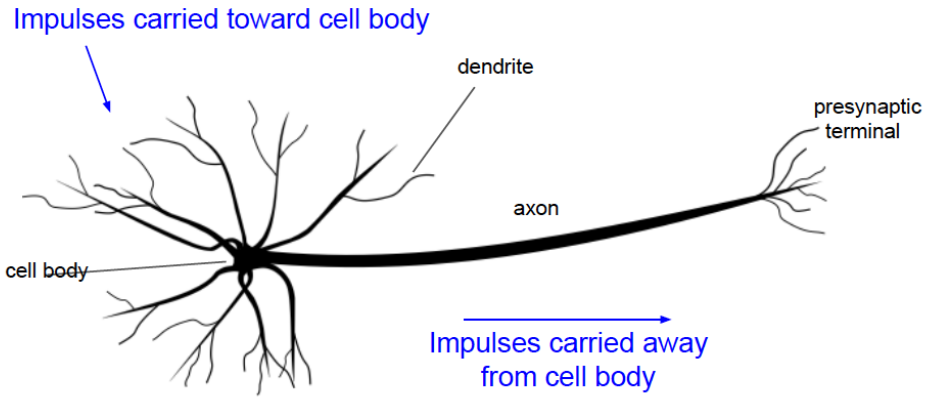


- Cognitron / Neocognitron [Fukushima 1971-1982]
  - Also HMAX [Poggio 2002-2006]



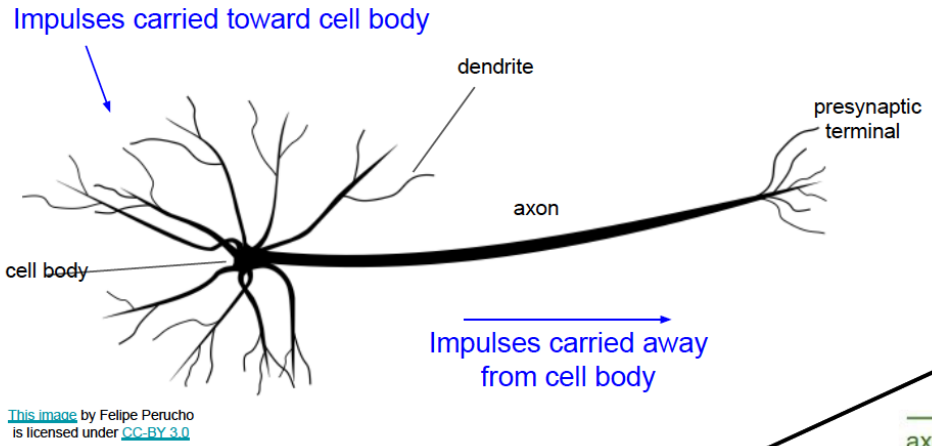
Convolutional Networks
[LeCun 1988-present]

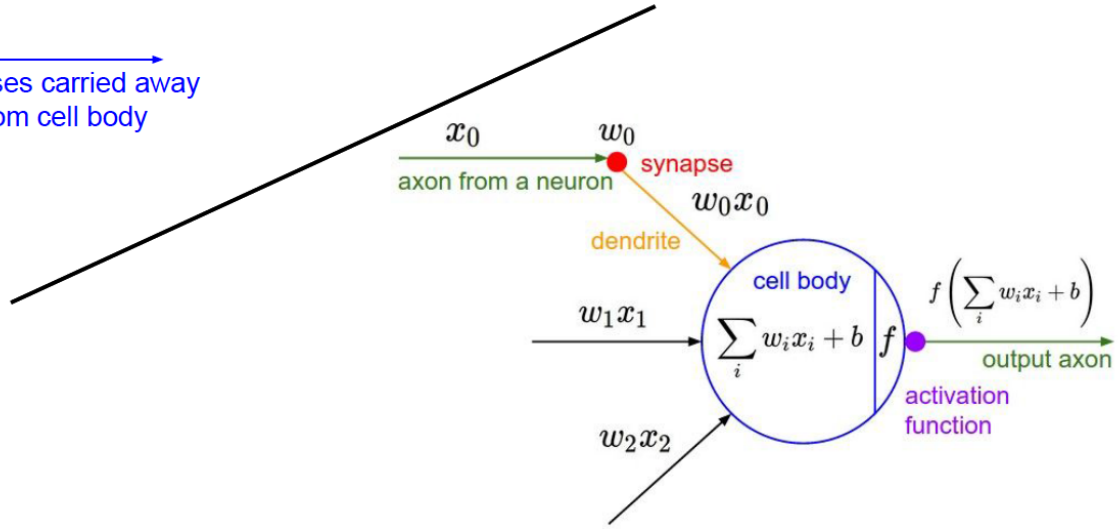
Some Inspiration from the (Human) Brain



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Some Inspiration from the (Human) Brain





slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Some Inspiration from the (Human) Brain

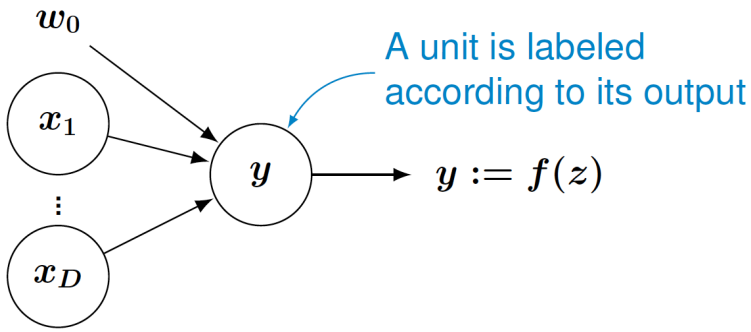
- Be careful with your brain analogies…

- Biological neurons:
  - many different types
  - dendrites can perform complex non-linear computations
  - synapses are n ot a single weight but a complex non-linear dynamical system
  - e.g. [Dendritic Computation, London & Hauser]

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Short Intro: “Standard” Neural Networks

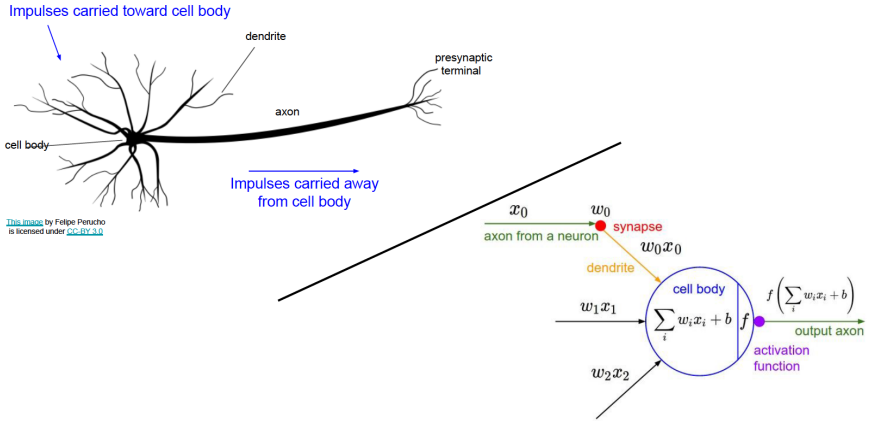
- Core component of a neural network: **processing unit = neuron** “of the human brain”
- The neuron (processing unit) maps multiple input values onto one output value y:



- $x_1, \ldots, x_D$ are **inputs**, e.g. an image or from other processing units within the network
- w_0 is an external input called **bias**
- the **propagation rule** maps all input values onto (intermediate input) z
- the **activation function** is applied to obtain $y = f(z)$

slide credit: David Stutz

Short Intro: Perceptron



- Introduced by Rosenblatt [Rosenblatt 58]

- The (single-layer) **perceptron** consists of *D* input units and *C* output units
  - propagation rule: weighted sum over inputs x_i with weights w_{ij}

$$y_j(\boldsymbol{x}, \boldsymbol{w}) = f(z_j) = f\left(\sum_{k=1}^{D} w_{jk} x_k + w_{j0}\right) \overset{x_0:=1}{=} f\left(\sum_{k=0}^{D} w_{jk} x_k\right).$$

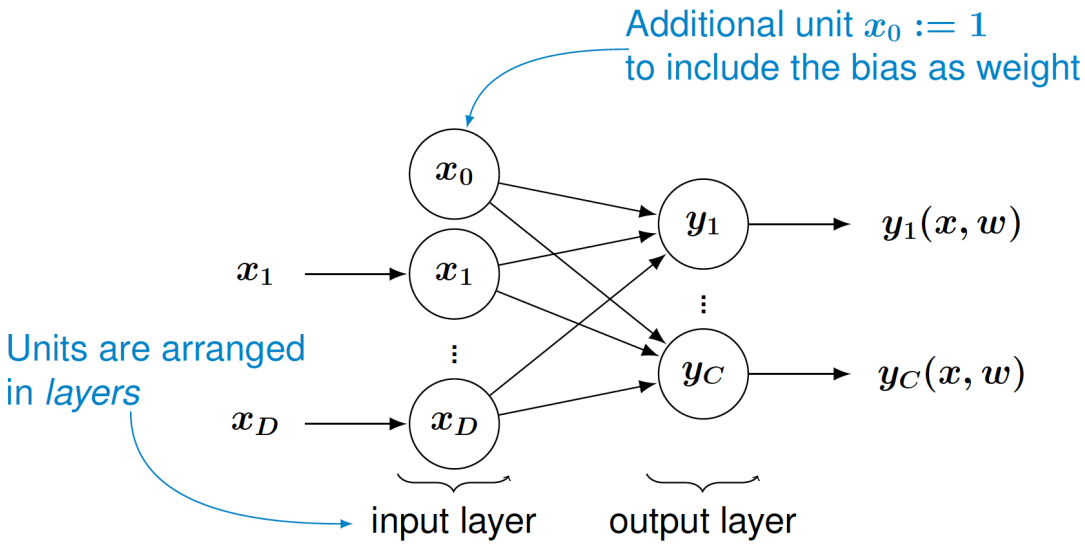
propagation rule with additional bias w_{j0}

slide credit: David Stutz

Short Intro: Perceptron

$$y_j(x, w) = f(z_j) = f\left(\sum_{k=1}^{D} w_{jk}x_k + w_{j0}\right) \overset{x_0:=1}{=} f\left(\sum_{k=0}^{D} w_{jk}x_k\right).$$

propagation rule with additional bias w_{j0}



slide credit: David Stutz

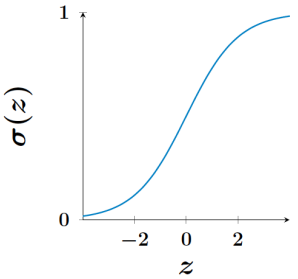
Short Intro: Perceptron - Activation Functions

- How to choose the activation function $f(z)$?
  - Heaviside function $h(z)$ models the electrical impulse of neurons in the human brain:

$$h(z) = \begin{cases} 1 & \text{if z} \geq 0 \\ 0 & \text{if z} < 0 \end{cases}$$

  - we prefer monotonic, differentiable activation functions — e.g. sigmoid $\sigma(z)$ as differentiable version of the Heaviside function:

$$\sigma(z) = \frac{1}{1 + e^{(-z)}} = \frac{1}{1 + \exp(-z)}$$



  - or the extension for multiple output units, the softmax activation function:

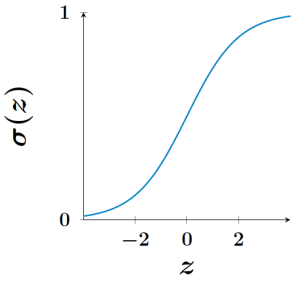
$$\sigma(z_i) = \frac{e^{(-z_i)}}{\sum_{j=1}^{C} e^{(-z_j)}} = \frac{\exp(-z_i)}{\sum_{j=1}^{C} \exp(-z_j)}$$

slide credit: David Stutz

Single Layer Perceptron — what can we represent?

$$y(x, w) = f(z) = f\left(\sum_{k=1}^{D} w_i x_i + w_0\right)$$

- Example: *D = 2:* $y(x, w) = f(z) = f\left(w_1 x_1 + w_2 x_2 + w_0\right)$
  - for all activations functions it is important to understand when $z = 0$
  - e.g. Heaviside function as activation function: $h(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$
  - or Sigmoid function as activation function: $\sigma(z) = \dfrac{1}{1 + \exp(-z)}$



Single Layer Perceptron — what can we represent?

- Example: *D = 2:*

$$y(x, w) = f(z) = f\left(w_1 x_1 + w_2 x_2 + w_0\right)$$

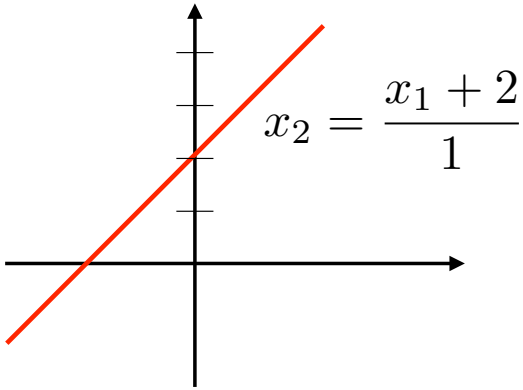
  - let's check:

$$z = 0 = w_1 x_1 + w_2 x_2 + w_0$$

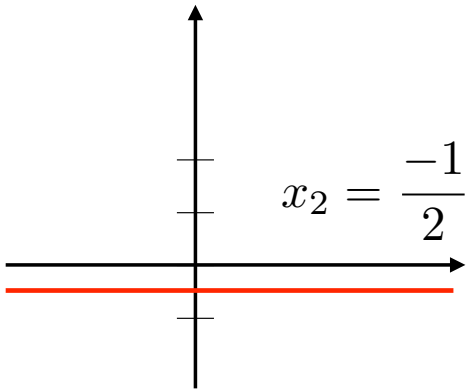
$$x_2 = \frac{-w_1 x_1 - w_0}{w_2}$$

  - examples:

$w_2 = 1, w_1 = -1, w_0 = -2$



$w_2 = 2, w_1 = 0, w_0 = 1$

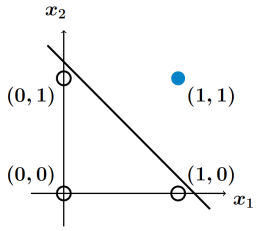


  - **single layer perceptrons can only represent linear decision surface (= hyperplane) !**

Single Layer Perceptron

Which target functions can be modeled using a single-layer perceptron?

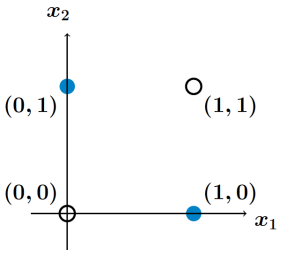
- **A single-layer perceptron represents a hyperplane in multidimensional space.**



Modeling boolean AND with target function $g(x_1, x_2) \in \{0, 1\}$.

Problem: How to model boolean exclusive OR (XOR) using a line in two-dimensional space?

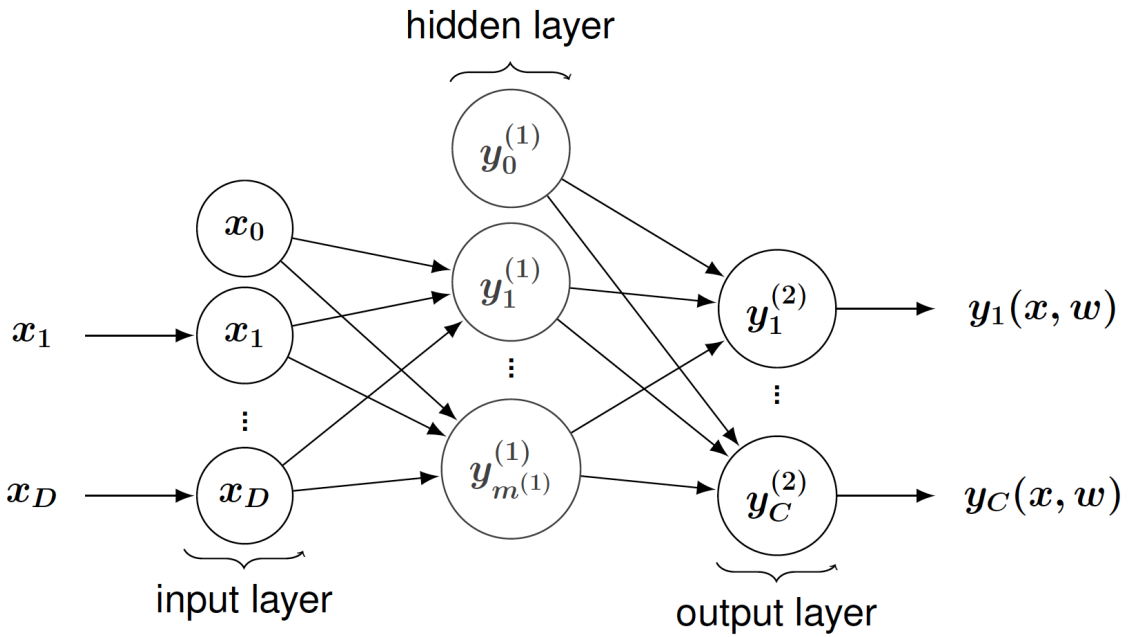
- **Boolean XOR cannot be modeled using a single-layer perceptron.**



Boolean exclusive OR target function.

slide credit: David Stutz

Short Intro: Two-Layer Perceptron



slide credit: David Stutz

Short Intro: Multi-Layer Perceptron (MLP)

Idea: Add additional $L > 0$ *hidden* layers in between the input and output layer.

- **$m^{(l)}$ hidden units in layer (l) with $m^{(0)} := D$ and $m^{(L+1)} := C$.**
- **Hidden unit i in layer l calculates the output**

layer

unit

$$y_i^{(l)} = f\left(\sum_{k=0}^{m^{(l-1)}} w_{ik} y_k^{(l-1)}\right).$$

A *multilayer perceptron* models a function

$$y(\cdot, w) : \mathbb{R}^D \mapsto \mathbb{R}^C, x \mapsto y(x, w) = \begin{pmatrix} y_1(x, w) \\ \vdots \\ y_C(x, w) \end{pmatrix} = \begin{pmatrix} y_1^{(L+1)} \\ \vdots \\ y_C^{(L+1)} \end{pmatrix}$$

where $y_i^{(L+1)}$ is the output of the i-th output unit.

slide credit: David Stutz