https://loma-aimotion.github.io/ml_visualizations/MiB_schiendorfer.pdf

Siehst du den Fehler? Wie funktionieren eigentlich Sichtprüfungen mit KI in der Produktion?

Prof. Dr. Alexander Schiendorfer

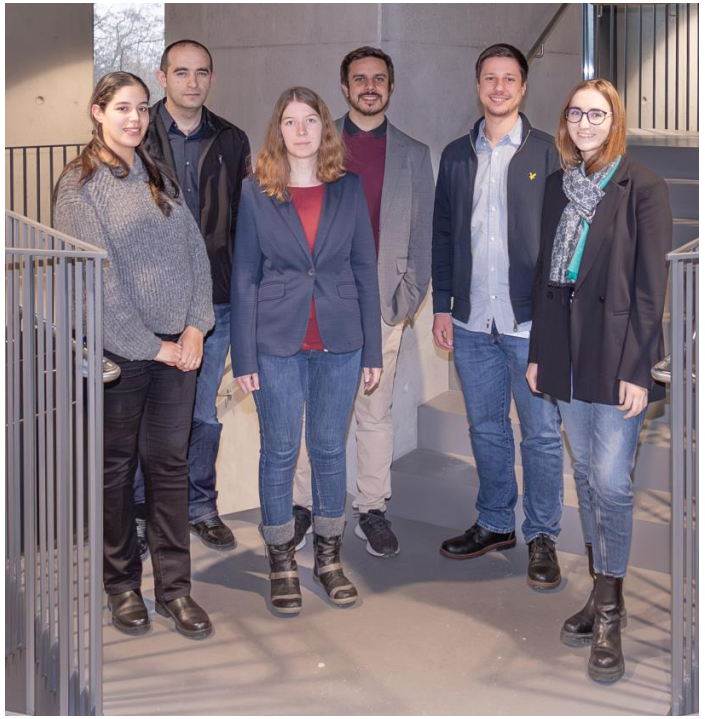
KI-basierte Optimierung in der Automobilproduktion

Technische Hochschule Ingolstadt

Forschungsprofessur Schiendorfer

Doktoranden und PostDocs

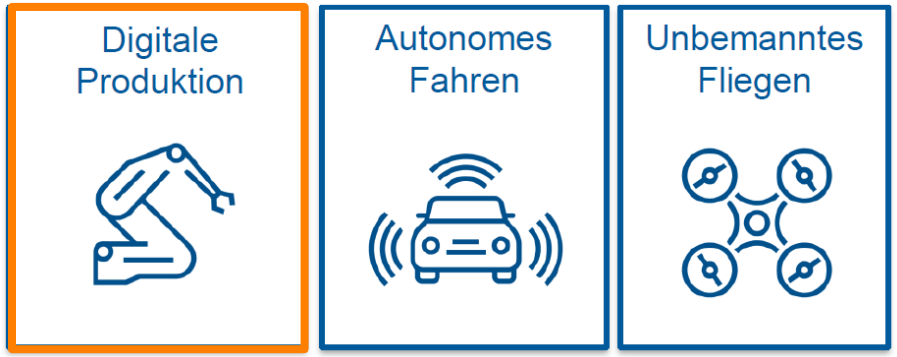
- Lukas Lodes, MSc
- Kristina Dachtler, MEng
- Dr. Karima Outafraout
- Pauline Steffel, MSc
- Gheorghe Lisca, MSc
- Mert Alagözlü, MEng



- F+E Partner
  - Valeo
  - plus10 GmbH
  - Convergent-IT
  - PSI Software
  - Büchl Entsorgungswirtschaft

Key facts „AImotion Bavaria“ aimotion.de

Bayerischer KI-Mobilitätsknoten an der THI



- 20 Forschungsprofessuren
  - *Schwerpunkte*: Autonome Mobilität (Fahren, Fliegen), ***KI-gestützte (Automobil)produktion***
  - *Zielsetzung*: Aufbau von bis zu 120 Forschende durch Drittmittelforschung (dzt. 75)

- Im Gebäude K (Neuer Dalwigk)
  - 4.000 m² Fläche, Seminarräume, KI-Labore
  - Ausbau Informatik, KI im Mobilitätsknoten

AImotion Bavaria

Institut AImotion Bavaria in Zahlen (Stand: 31.12.2023)

Wissenschaftliche Mitarbeitende 60

27 Promovierende

26 Professuren
+ 3 ab 2024/25

86 Forschende

1 Labor für Flugmechanik

5 Labore ab 2026

16 Demonstratoren

Dig. Produktion 1,83 Mio.

Eingeworbene Drittmittel: 6,8 Mio. €

Autonomes Fahren 620 T €

Unbemanntes Fliegen 1,3 Mio.

Querschnitts-cluster 3,06 Mio.

THI: 23 Mio. €. p.a.

AI 2023

Publikationen: 57

2 Spin-offs

Events
- KONVENS
- Hackadon
- AI.BAY
- UAM-Netzwerkkonferenz
- BAI.CON
- BRK Drohnentag

Fahrzeuge
Roboterhund
Forschungsgeräte
Drohnen Turtlebots
Mikroskope HPC

Institute des KI-Mobilitätsknotens:

AImotion Bavaria
KI-Knoten

AININ gGmbH
An-Institut

Technologie-Transfer Zentrum
Manching

KI-Ökosystem:

baiosphere the bavarian ai network

KI HUB BAYERN

KI BUNDESVERBAND

brigk

brigk AIR

Center of Entrepreneurship

EXISTENZGRÜNDERZENTRUM INGOLSTADT

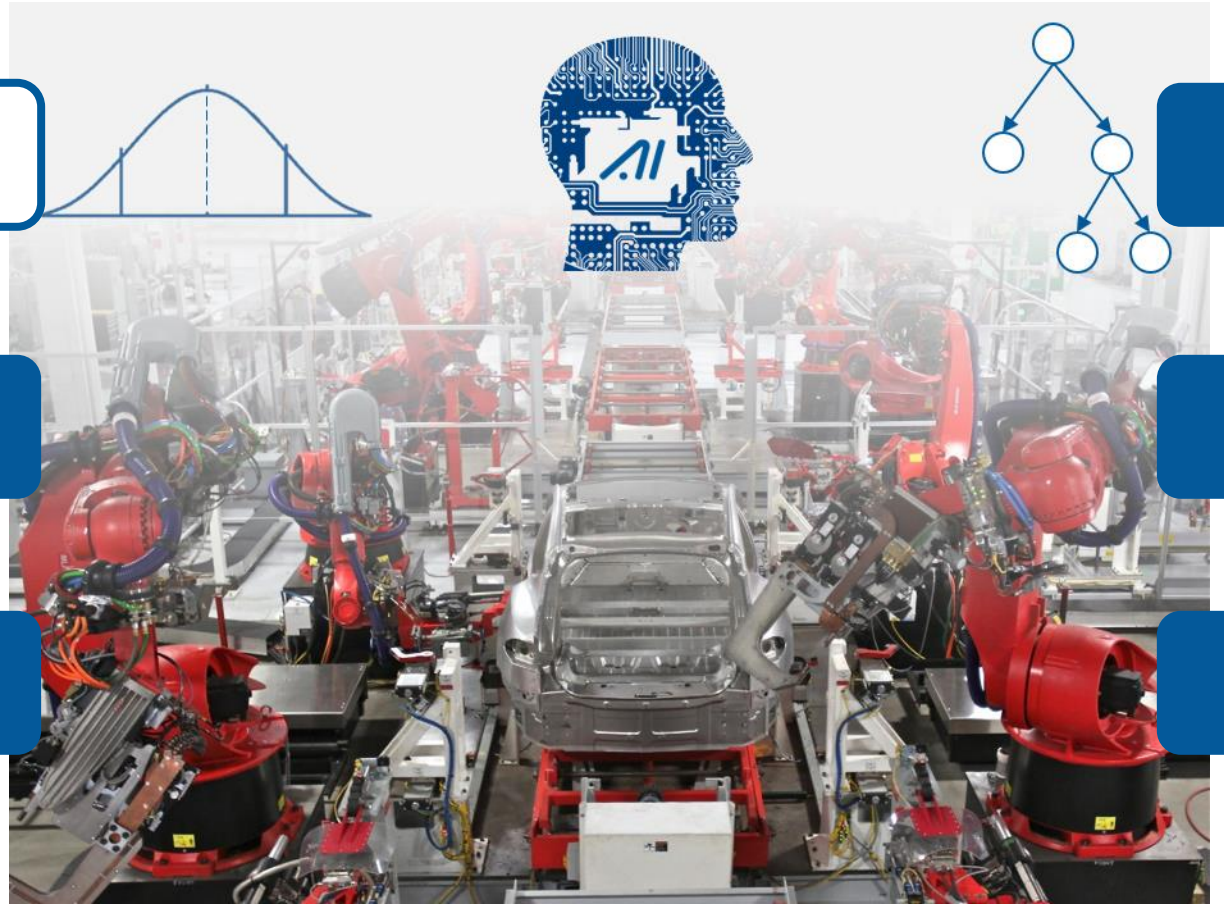
AI in Automotive Production

Mensch in Bewegung

Visual Inspection

Predictive Maintenance

Predictive Quality



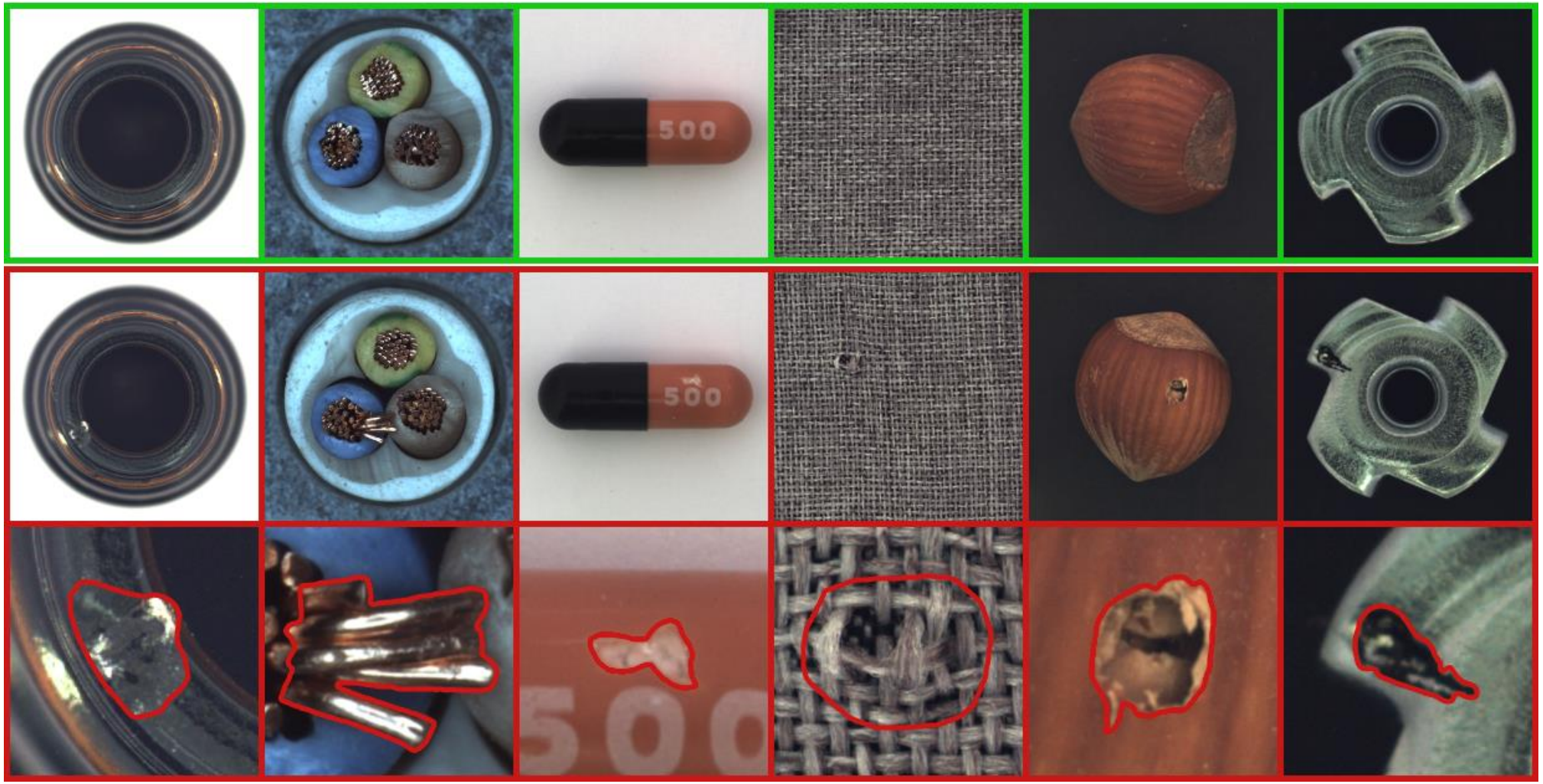
Digital Twins

Scheduling

Process Control

KI-basierte Bildverarbeitung

Visual anomaly detection in industrial use cases

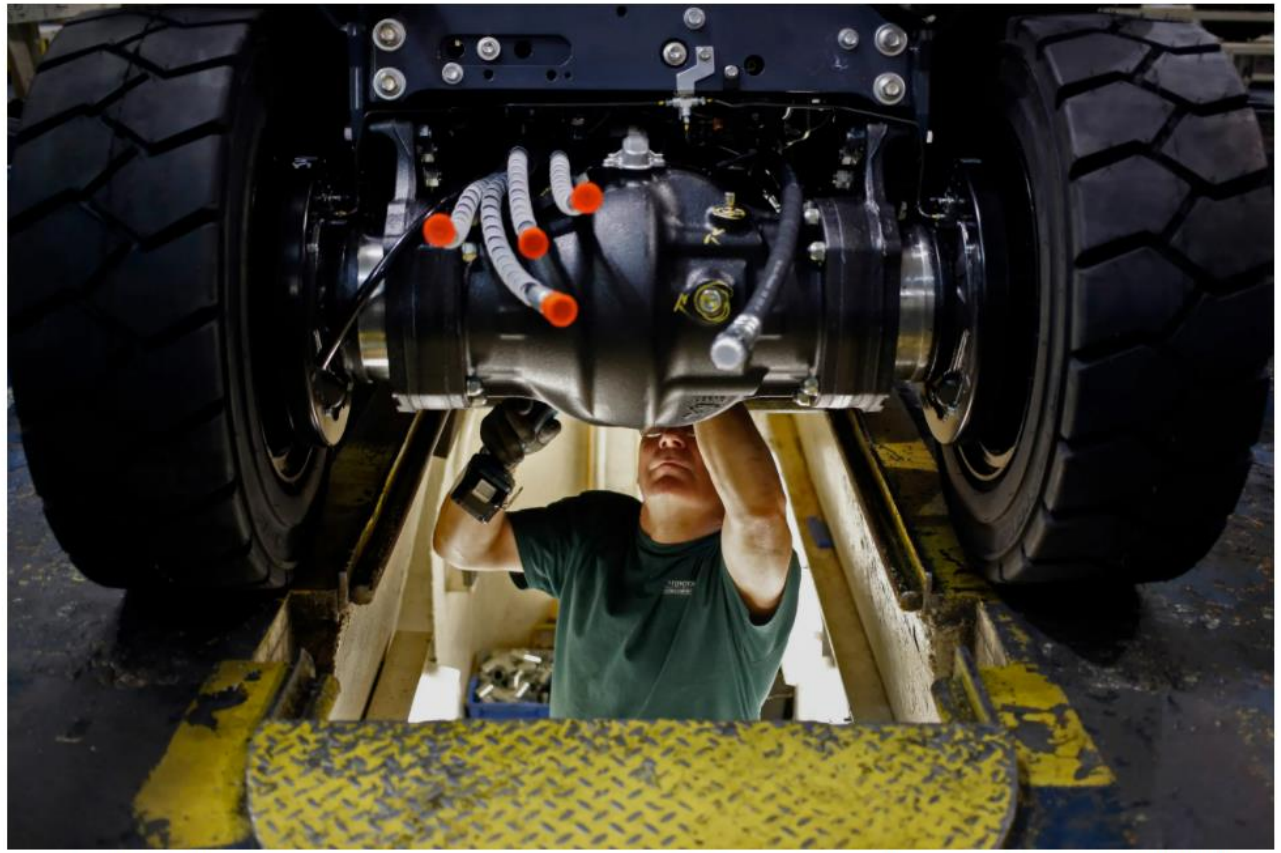


https://www.mvtec.com/company/research/datasets/mvtec-ad

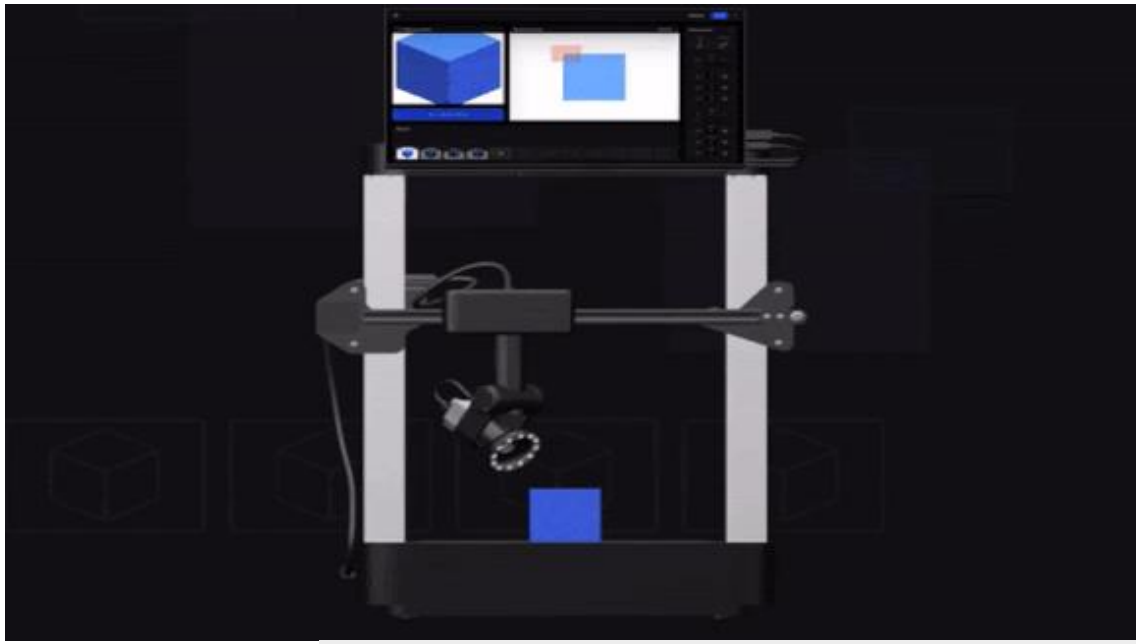
WILL KNIGHT BUSINESS 07.01.2020 07:00 AM

In These Factories, Inspector Robot Will Check Your Work

Artificially intelligent camera systems look for defects and misplaced parts in many industries. The coronavirus pandemic makes them extra useful.



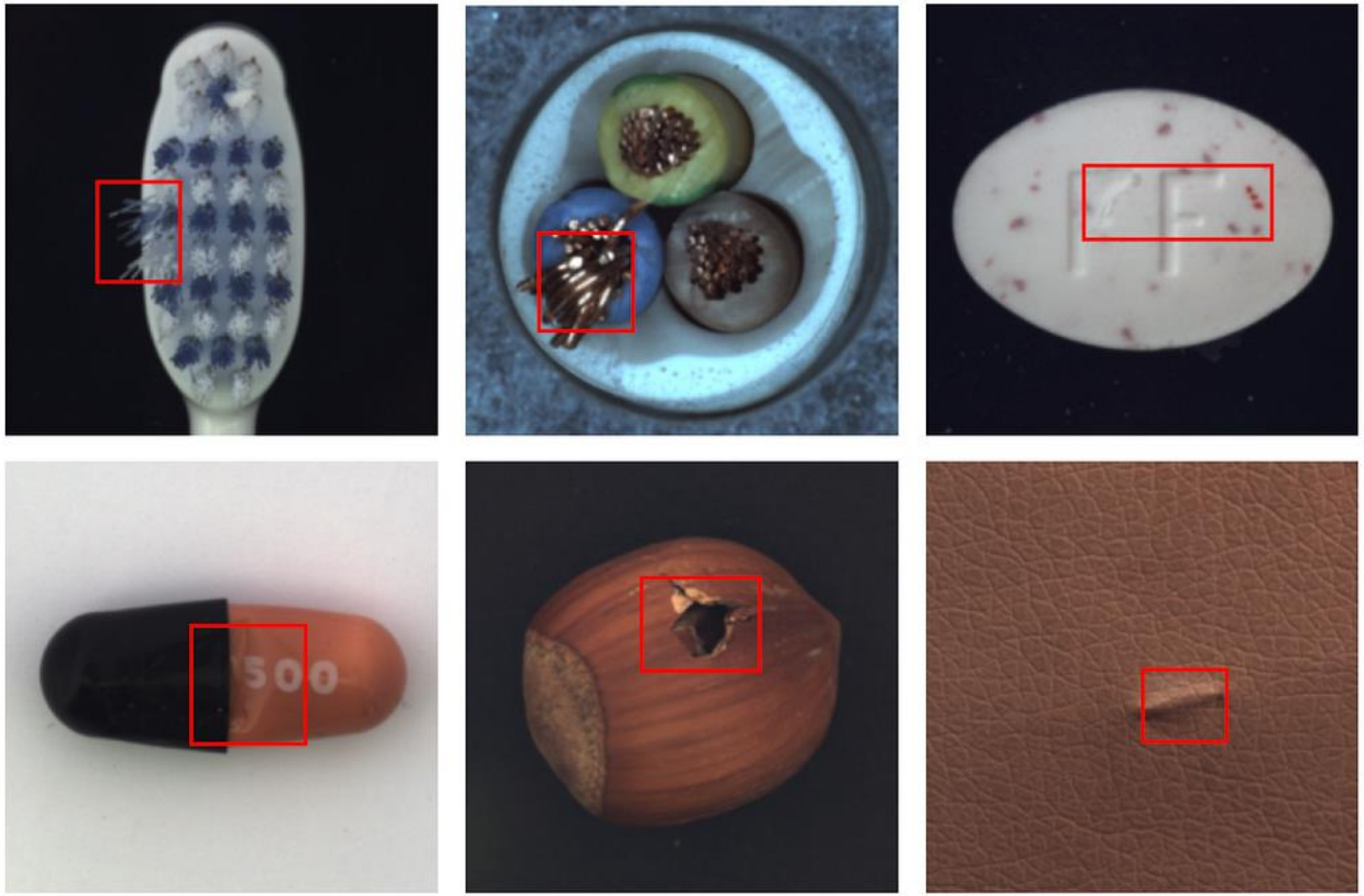
A Toyota assembly plant in Indiana is preparing to deploy a robotic system that moves a camera around an object, to ensure employees are using the correct parts. PHOTOGRAPH: LUKE SHARRETT/BLOOMBERG/GETTY IMAGES



INSTRUMENTAL

https://www.wired.com/story/factories-inspector-robot-check-work/

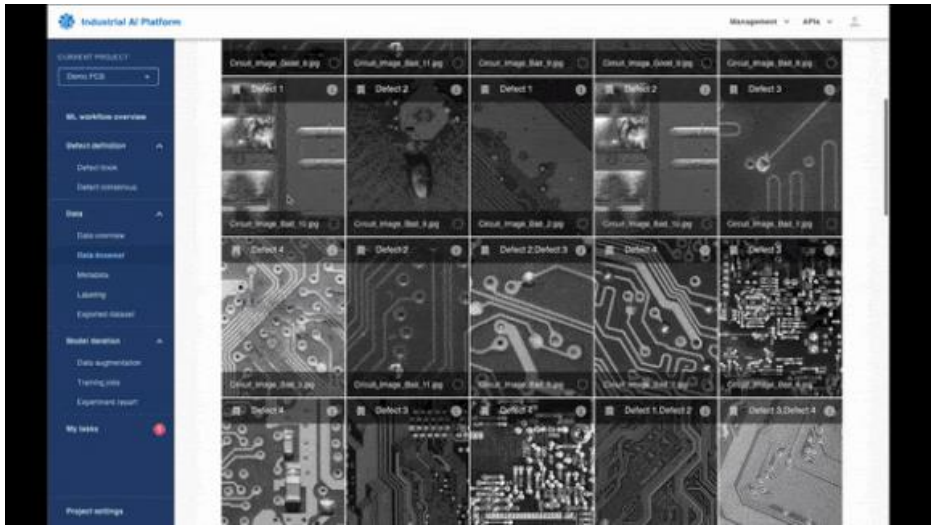
Defect Detection



Tracking the state of a jobshop in production

https://sordi.ai/

You‘ll work with AI systems!



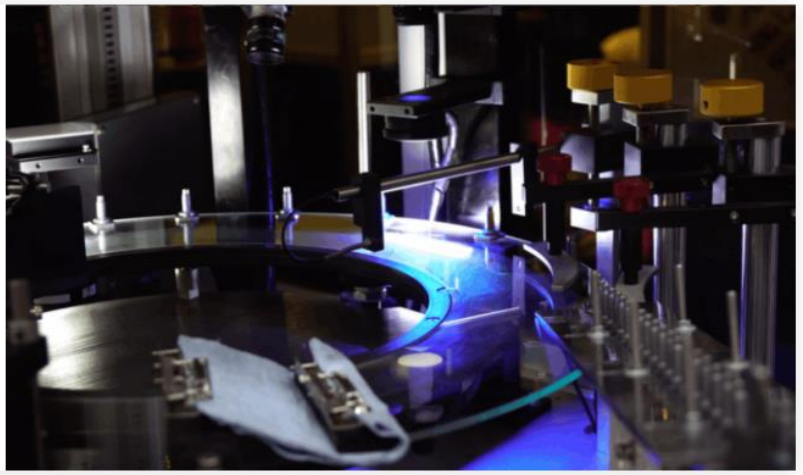
Unambiguous defect (y=1) All labelers will agree

Ambiguous whether to classify as defect Even expert labelers disagree

Unambiguous no defect (y=0) All labelers will agree

Manufacturing

Every scratch, chip, or defective part matters to the bottom line. Identifying defects correctly and consistently is critical to production success. Inspectors are constantly challenged with defining defects randomly. Landing AI provides an integrated, end-to-end visual inspection platform that effectively manages your data and builds solutions you can rely on.



Rather than holding the training set fixed and trying to improve the model, we hold the model fixed and help manufacturers improve the training set. We've found that this approach leads to faster progress in production settings.

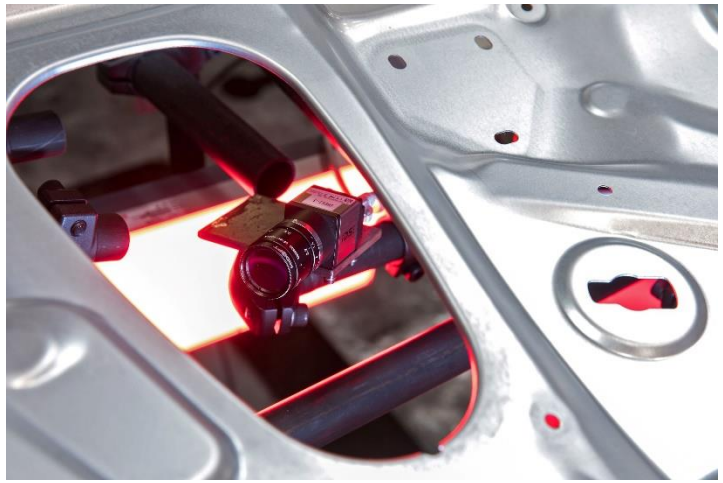
https://landing.ai/industries/#automotive

Other application: Visual quality inspections at Audi

10/15/18 | Ingolstadt | Company

Audi optimizes quality inspections in the press shop with artificial intelligence

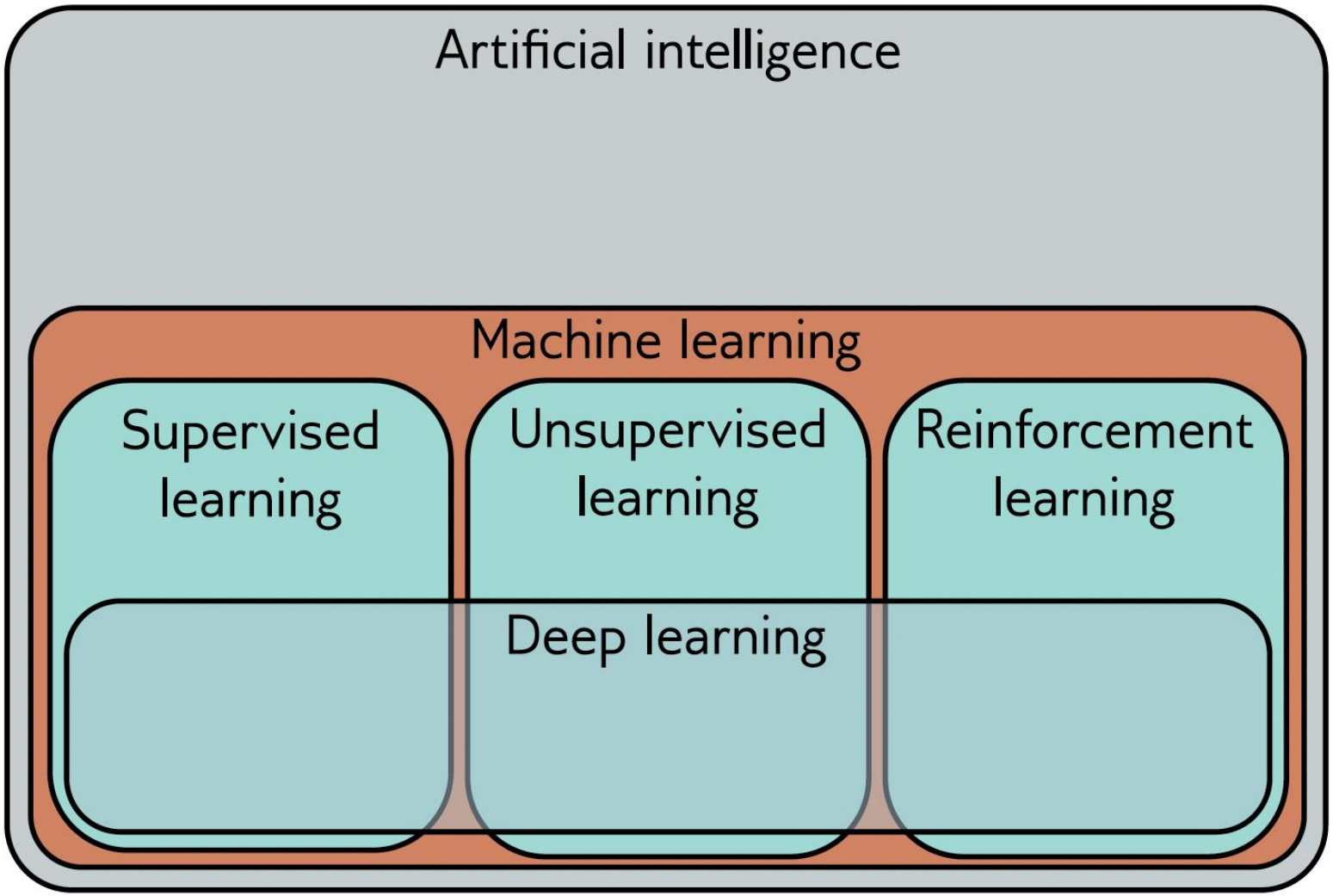
- First application case goes into series development
- Crack detection in the press shop to be automated with machine learning



https://www.audi-mediacenter.com/en/press-releases/audi-optimizes-quality-inspections-in-the-press-shop-with-artificial-intelligence-10847

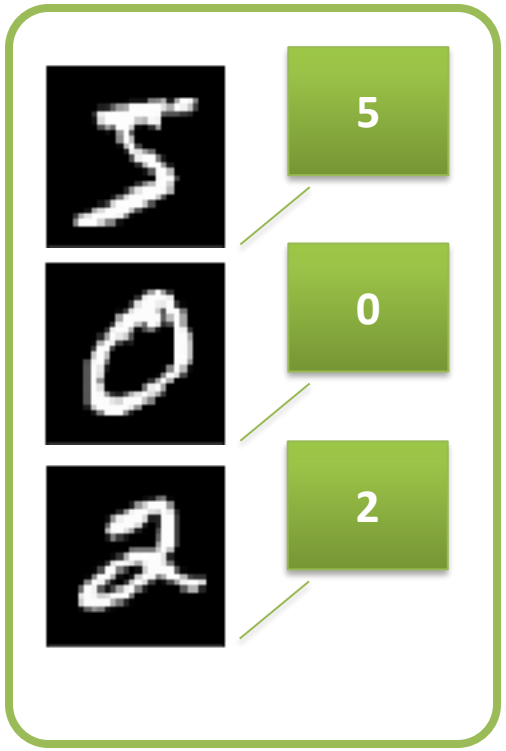
What is Deep Learning?

Categorization of AI



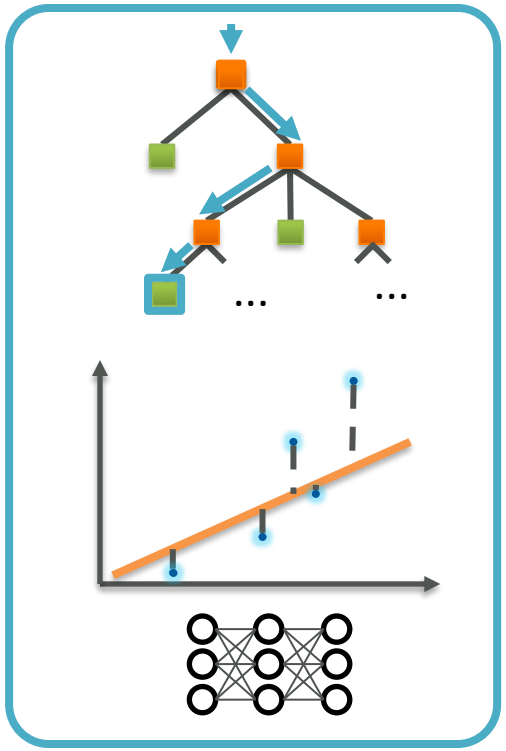
The three pillars of all machine learning systems

$X \in \mathbb{R}^{m \times n}$
$y \in \mathbb{R}^{m \times 1}$



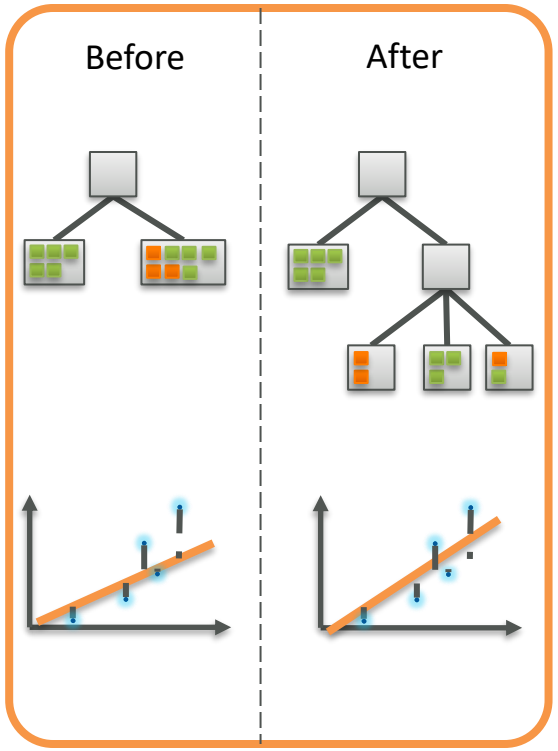
Data

$f_\theta : \mathbb{R}^m \mapsto \mathbb{R}$
$\theta \in \mathbb{R}^{k \times 1}$



Models

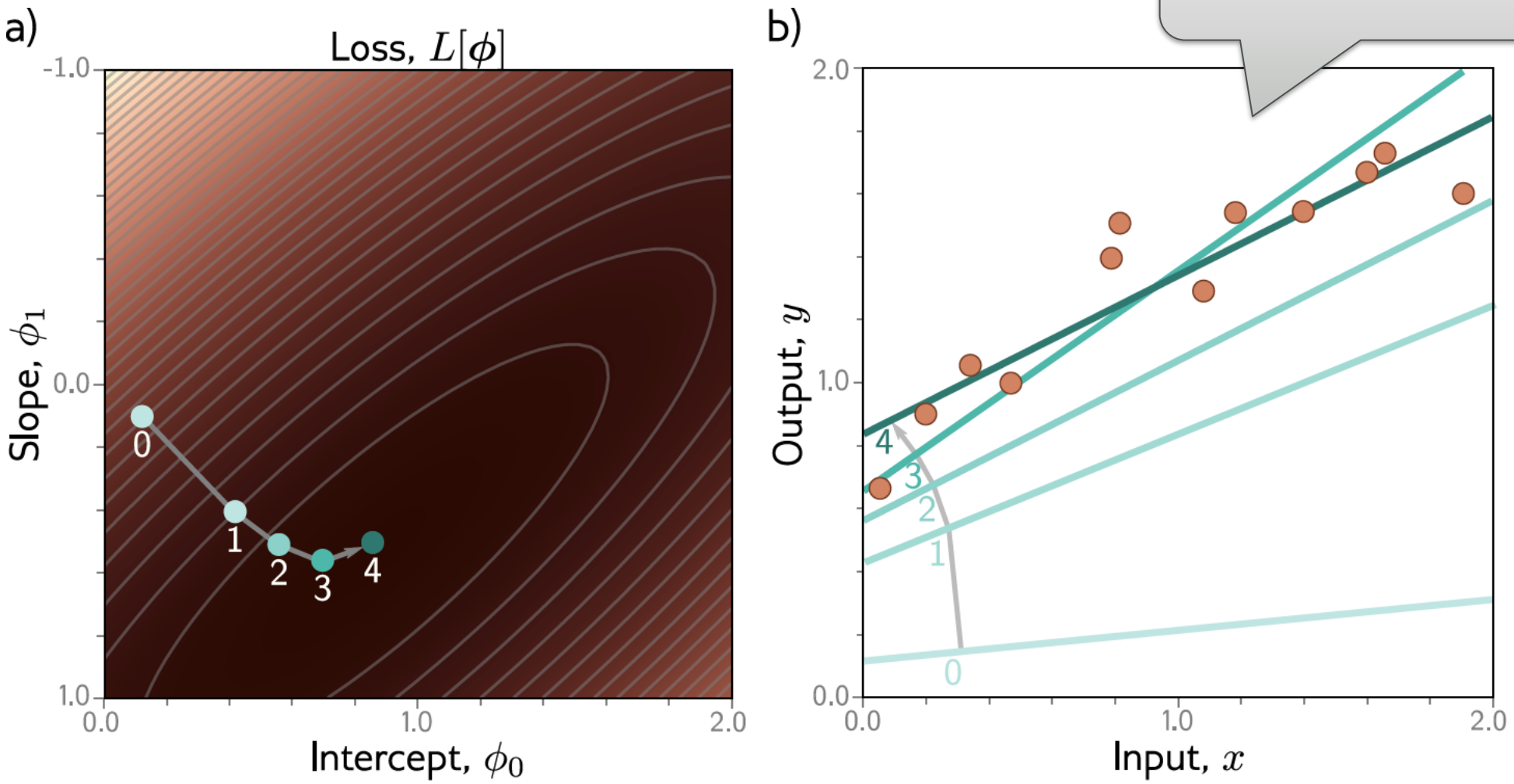
$\theta = \text{maximize}\, P$
$f_\theta(x^{(i)}) \approx y^{(i)}, \forall i \in 1 \ldots m$



Learning

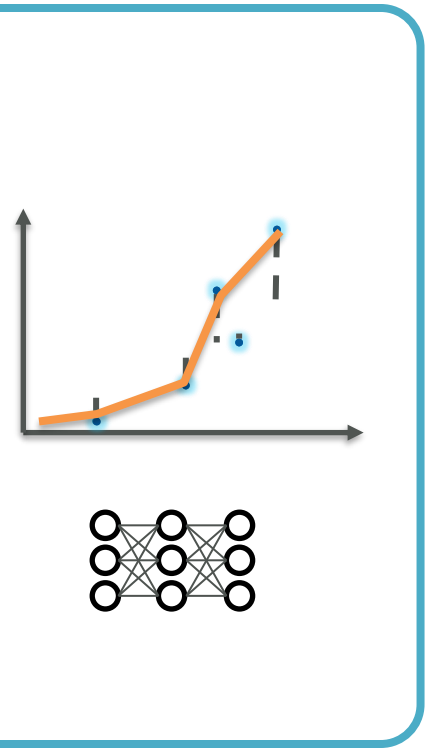
Example of Non-Deep ML: Linear Regression

Model contains all linear relations between x and y.



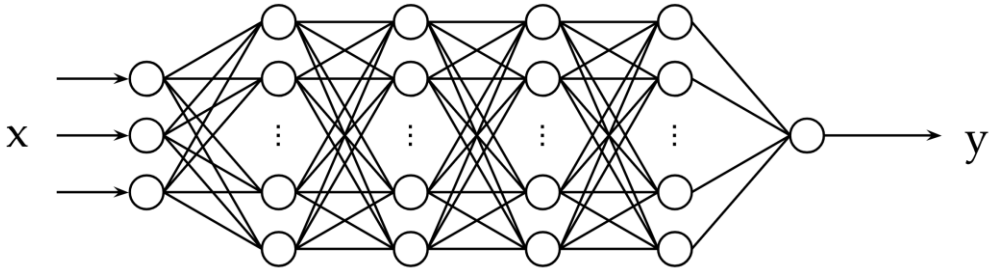
In deep learning:

$$f_\theta: \mathbb{R}^m \mapsto \mathbb{R}$$
$$\theta \in \mathbb{R}^{k \times 1}$$



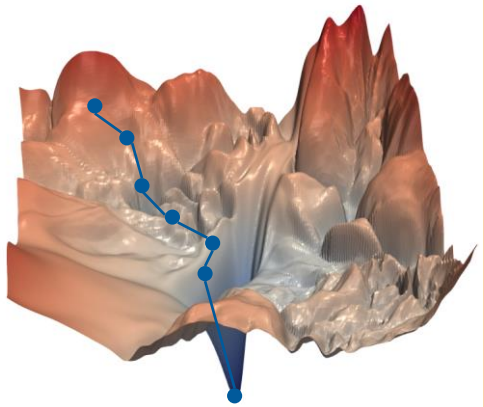
Models

Models are "neural networks" with many parameters, stacked in many layers -> **universal function approximators**



Learning is some form of **gradient descent**, where a loss function between current and desired target values is minimized.

$$\theta = \text{maximize}\, P$$
$$f_\theta\left(x^{(i)}\right) \approx y^{(i)}, \forall i \in 1 \ldots m$$

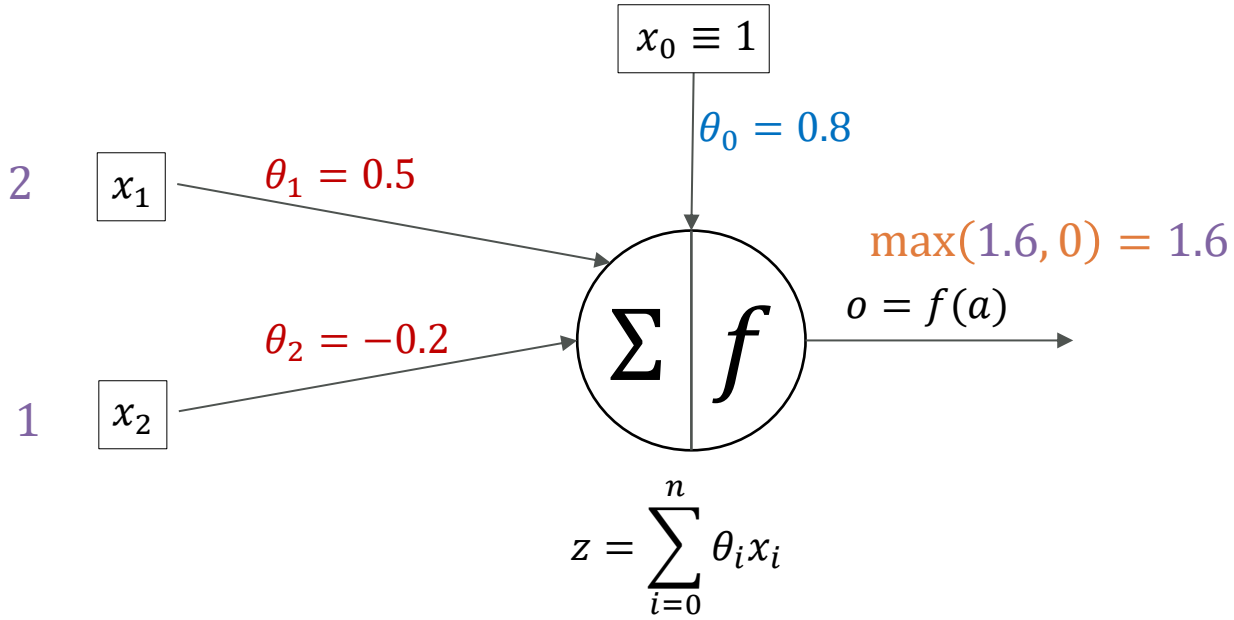


Learning

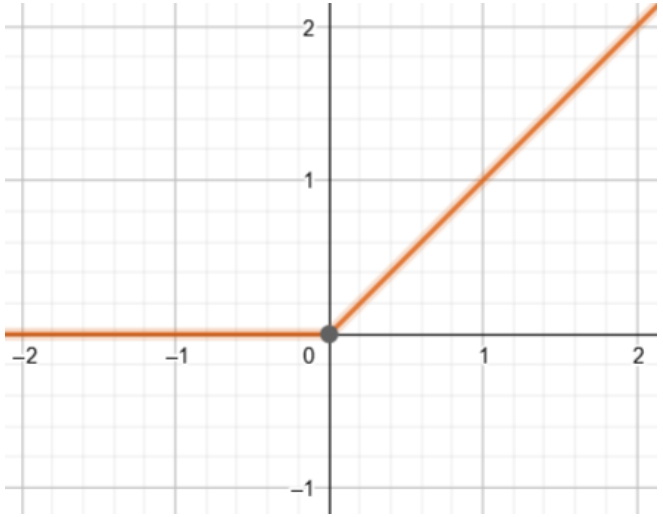
https://arxiv.org/pdf/1712.09913.pdf

Inside a Neural Network

Example (single) Artificial Neuron



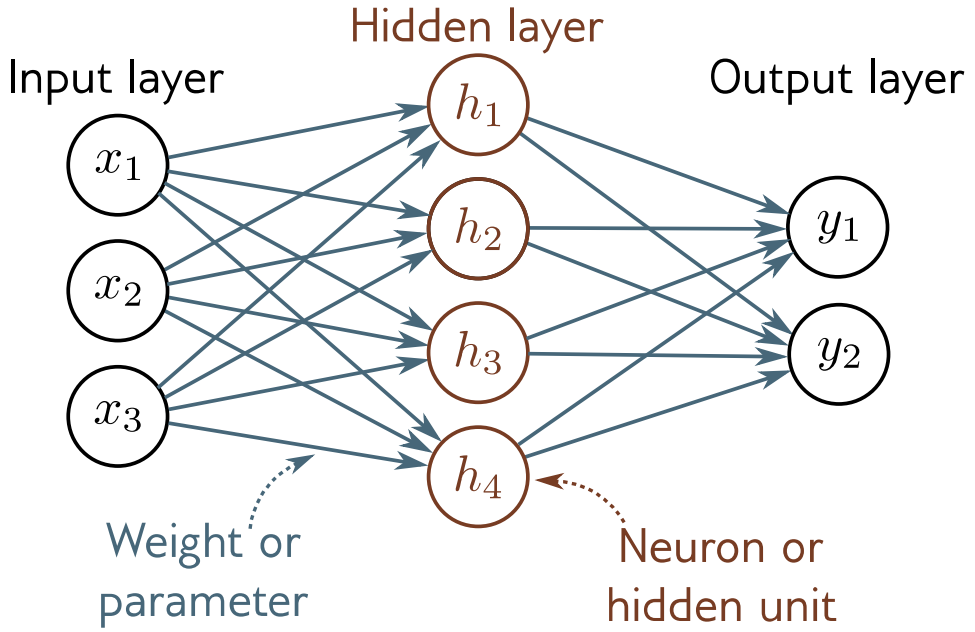
$2 \cdot 0.5 + 1 \cdot (-0.2) + 0.8 = 1.6$



$f(z) = \max(z, 0)$

x_i:	inputs	θ_i:	weights
θ_0:	bias weight	$a(z)$:	activation function
z:	pre-activation	o:	output

Nomenclature



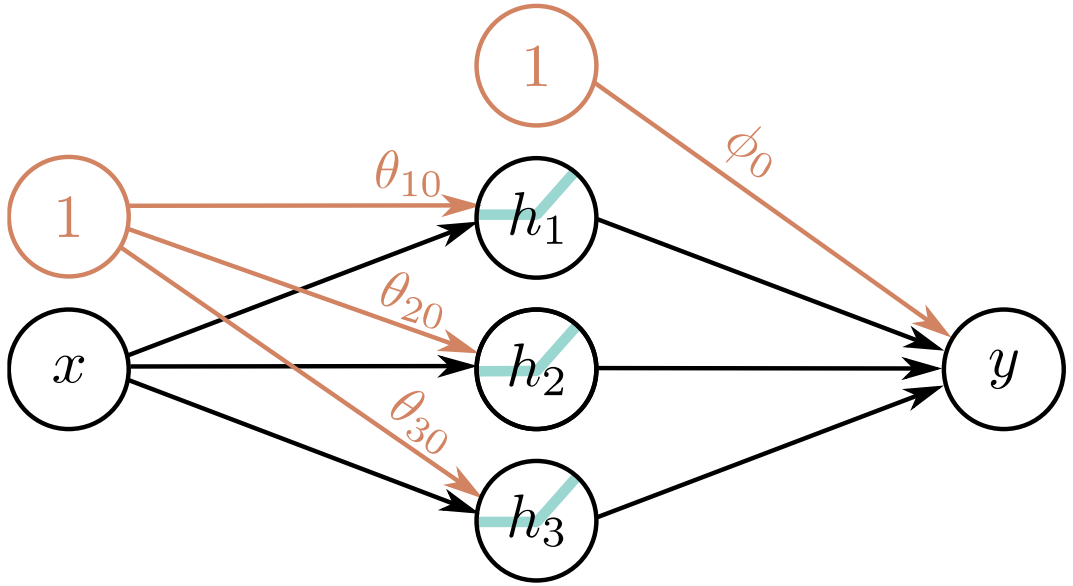
- Y-offsets = biases
- Slopes = weights
- Everything in one layer connected to everything in the next = fully connected network
- No loops = feedforward network
- Values after ReLU (activation functions) = activations
- Values before ReLU = pre-activations
- One hidden layer = shallow neural network
- More than one hidden layer = deep neural network
- Number of hidden units ≈ capacity

https://loma-aimotion.github.io/ml_visualizations/shallow_neural_network.html

Depicting neural networks

$$h_1 = \mathrm{a}[\theta_{10} + \theta_{11}x]$$
$$h_2 = \mathrm{a}[\theta_{20} + \theta_{21}x]$$
$$h_3 = \mathrm{a}[\theta_{30} + \theta_{31}x]$$

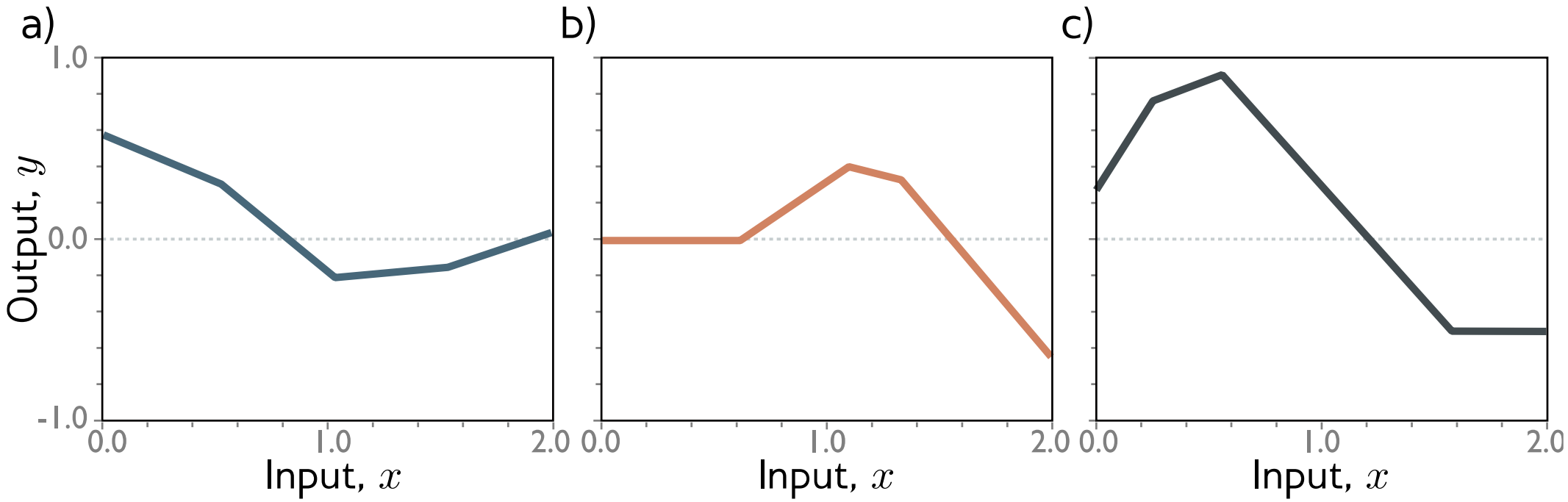
$$y = \phi_0 + \phi_1 h_1 + \phi_2 h_2 + \phi_3 h_3$$



Each parameter multiplies its source and adds to its target

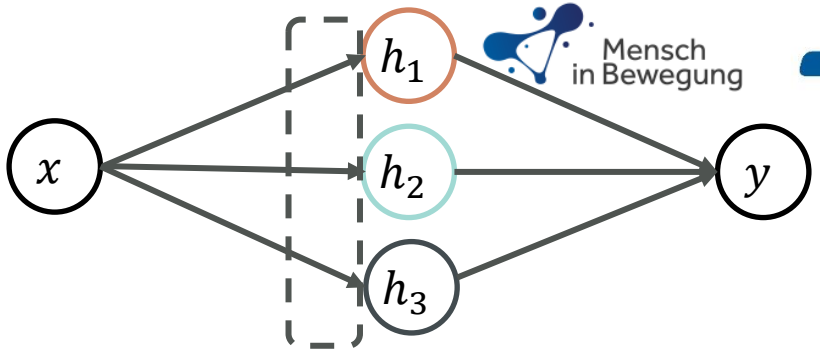
Example shallow network

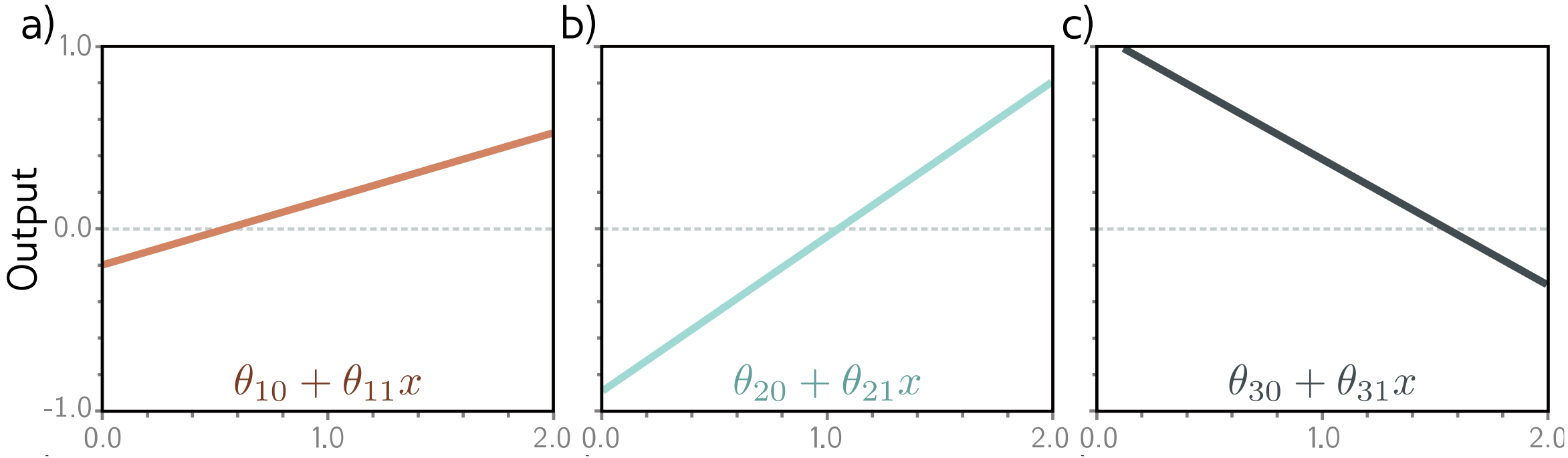
$$y = \phi_0 + \phi_1 \mathrm{a}[\theta_{10} + \theta_{11}x] + \phi_2 \mathrm{a}[\theta_{20} + \theta_{21}x] + \phi_3 \mathrm{a}[\theta_{30} + \theta_{31}x].$$



Piecewise linear functions with three joints

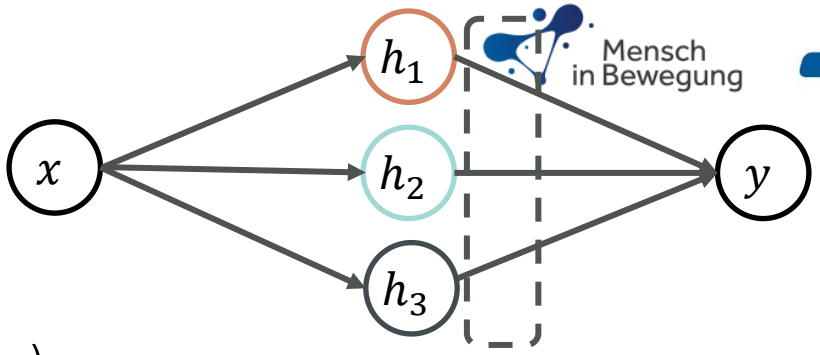
1. compute three linear functions

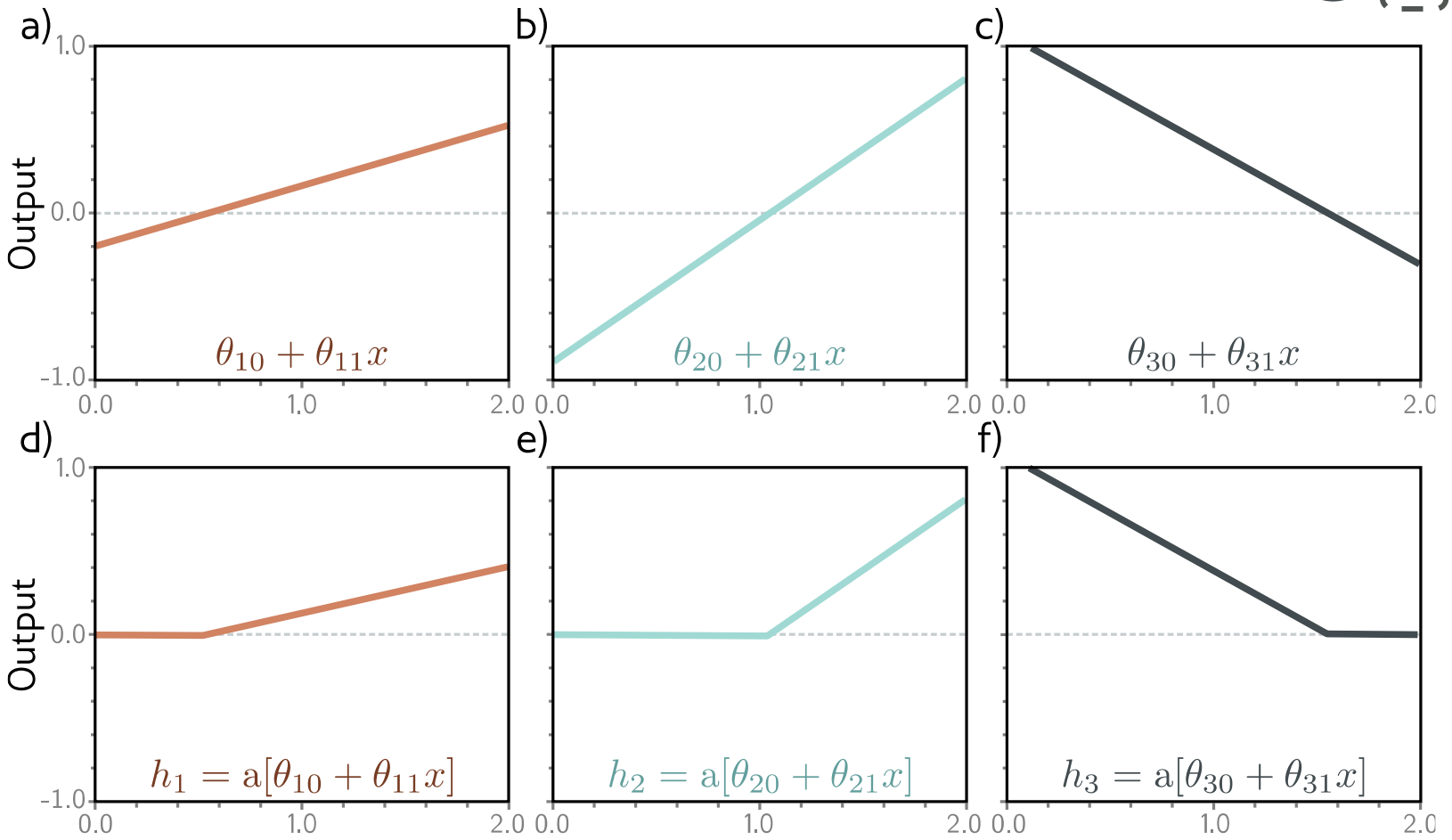




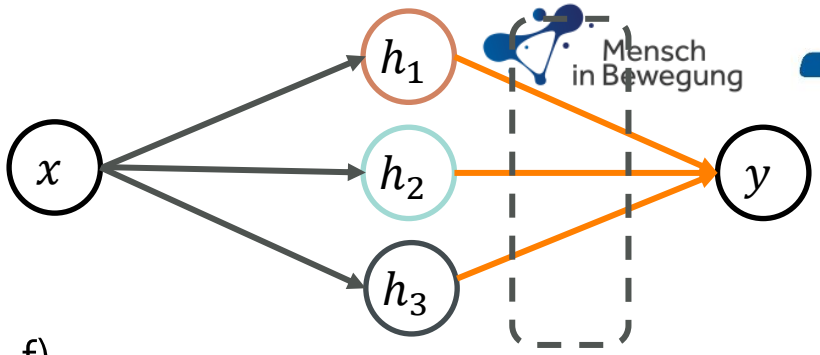
2. Pass through ReLU functions

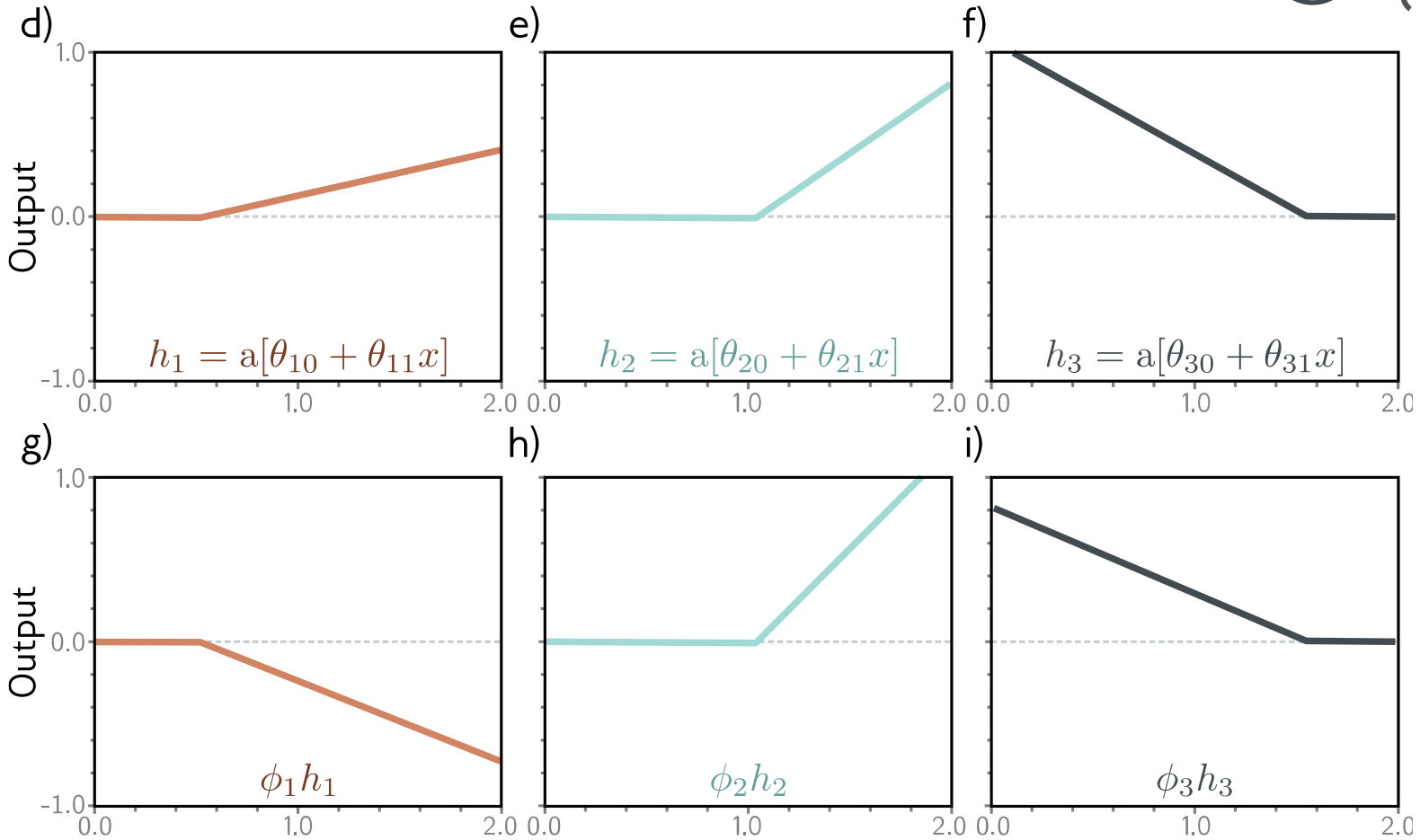
$$h_1 = \mathrm{a}[\theta_{10} + \theta_{11}x]$$
$$h_2 = \mathrm{a}[\theta_{20} + \theta_{21}x]$$
$$h_3 = \mathrm{a}[\theta_{30} + \theta_{31}x],$$



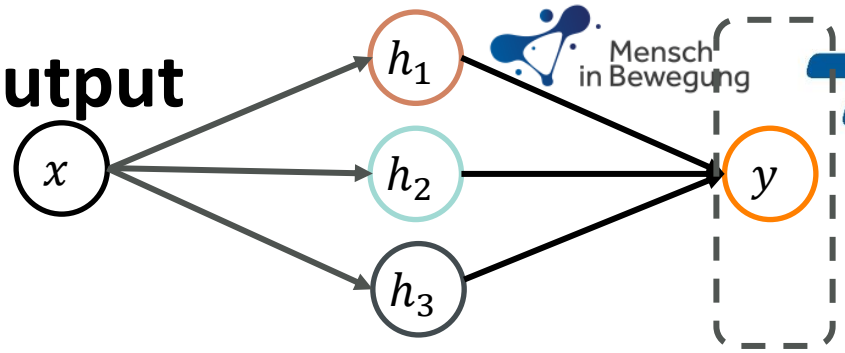


3. Weight the hidden units

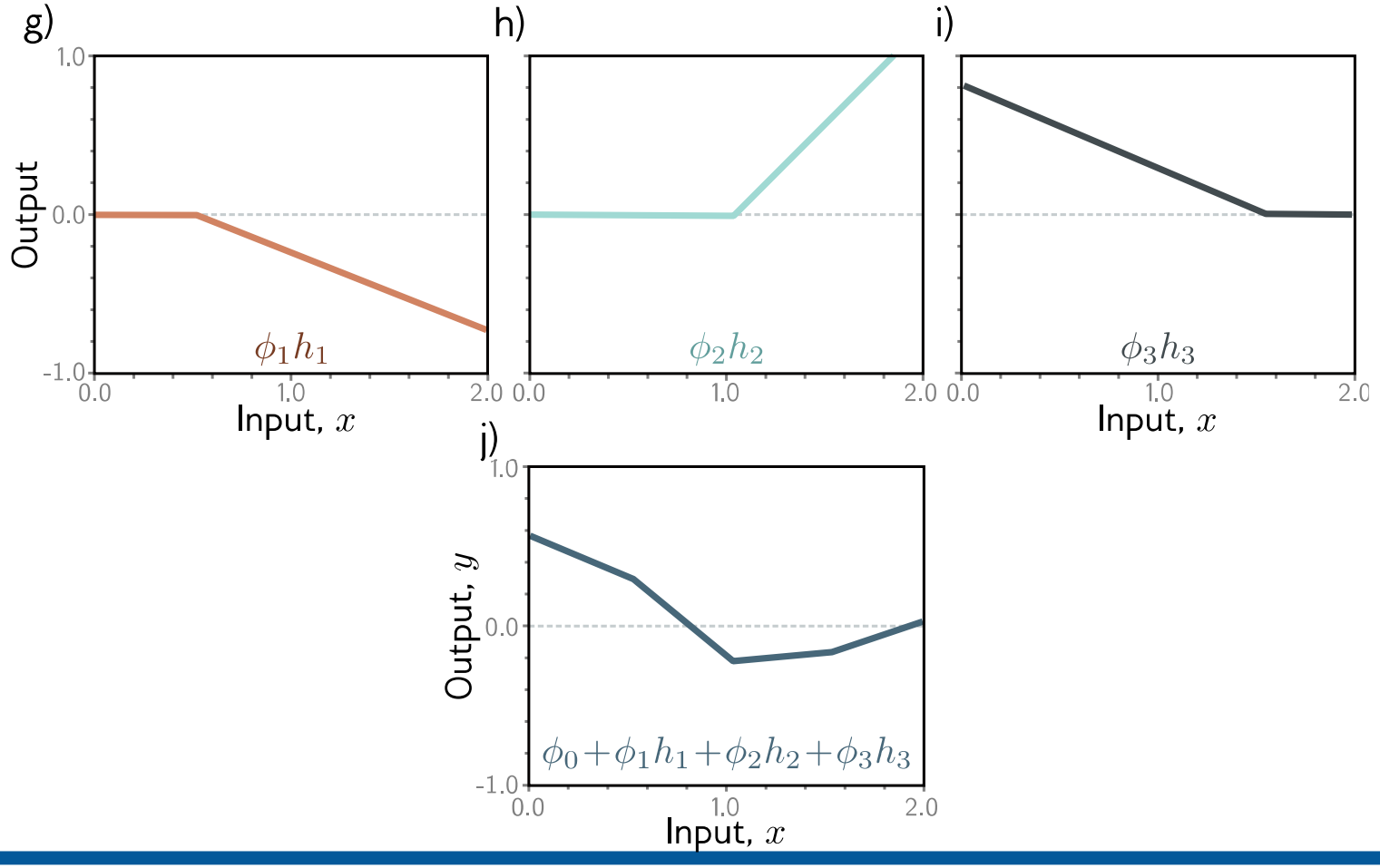




4. Sum the weighted hidden units to create output

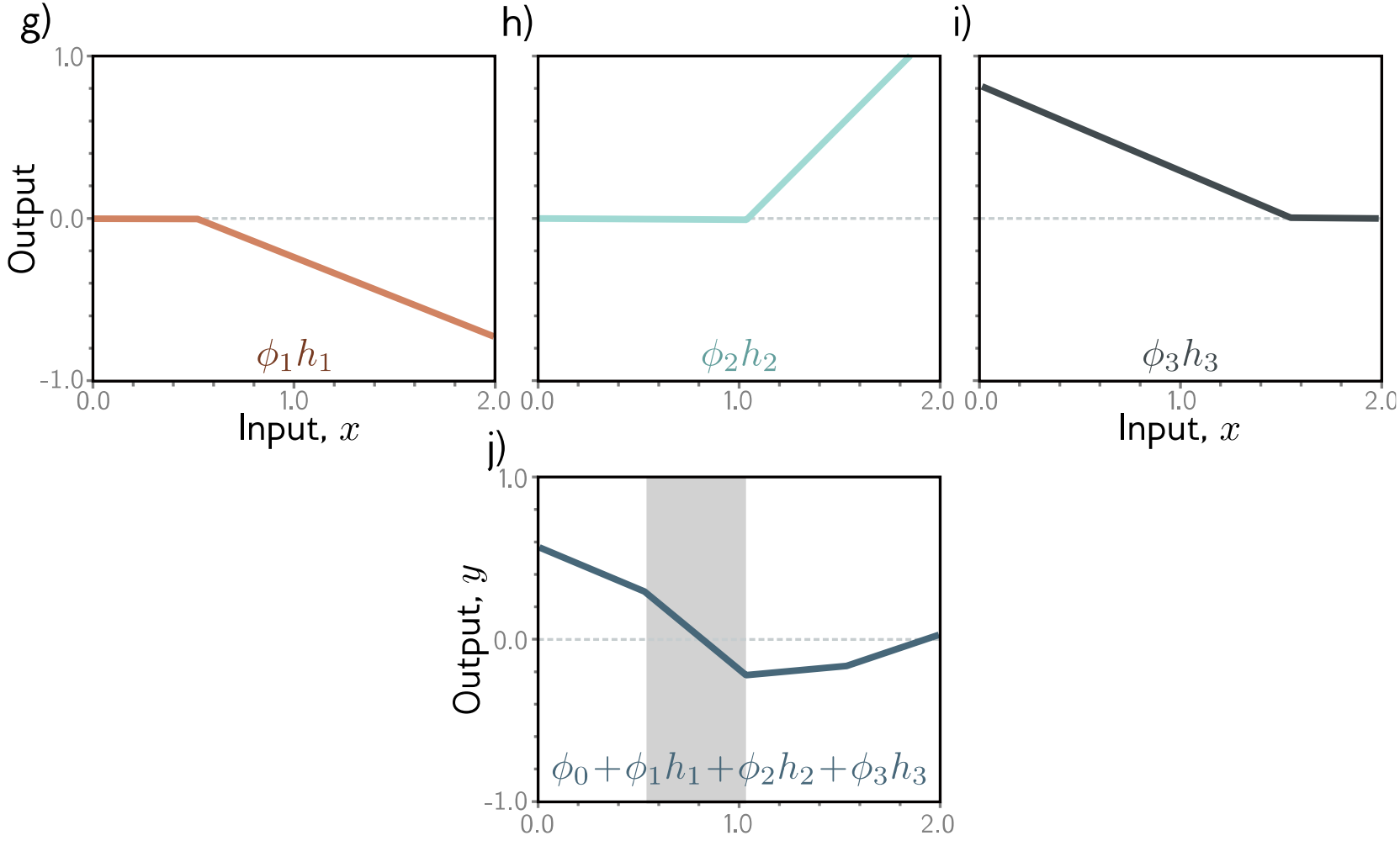


$$y = \phi_0 + \phi_1 h_1 + \phi_2 h_2 + \phi_3 h_3$$



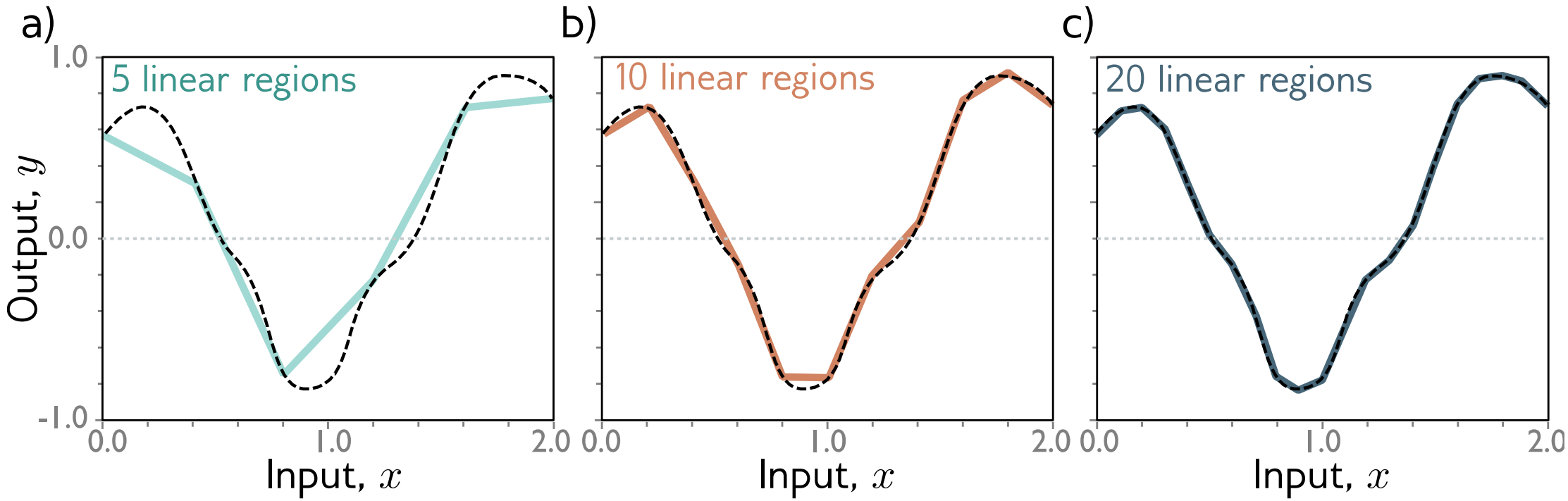
Activation pattern = which hidden units are activated

- Shaded region:
- Unit 1 active
- Unit 2 inactive
- Unit 3 active



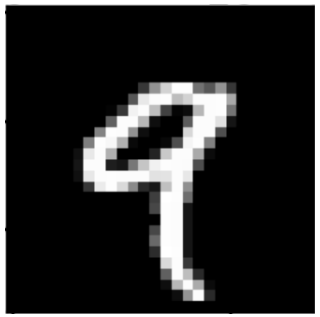
With enough hidden units…

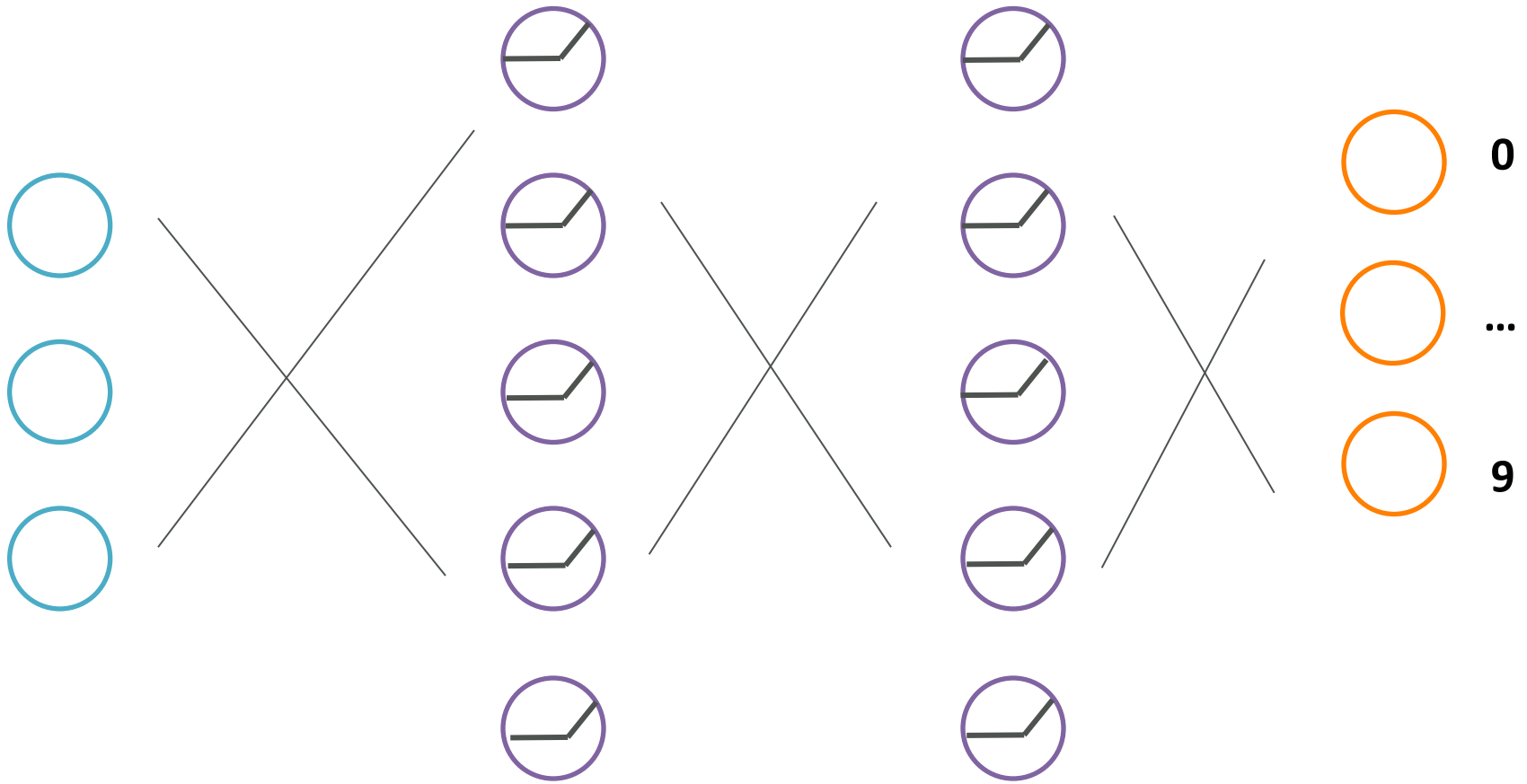
… we can describe any 1D function to arbitrary accuracy



Automatically Training a Neural Network

Training neural nets with backpropagation

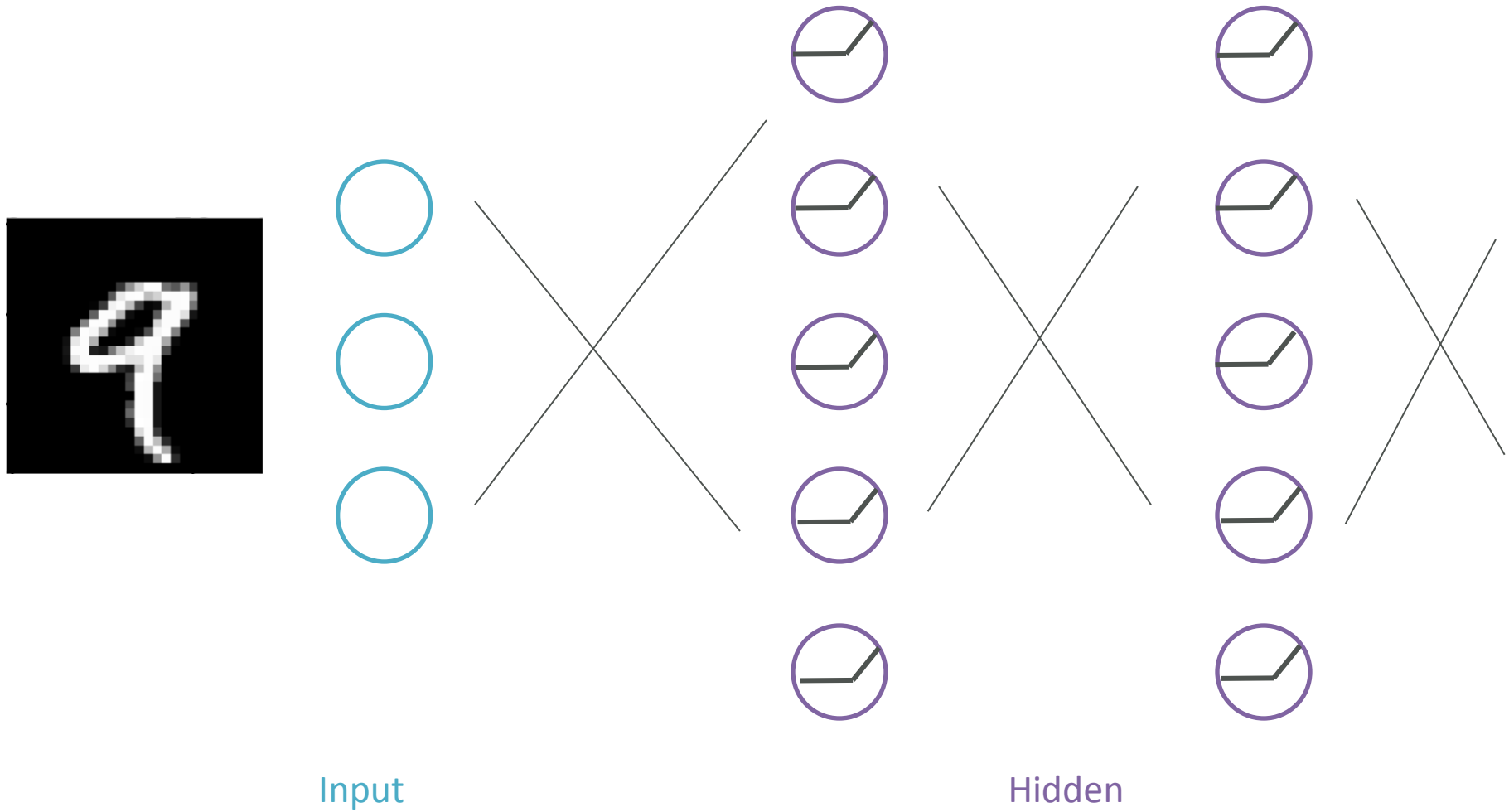




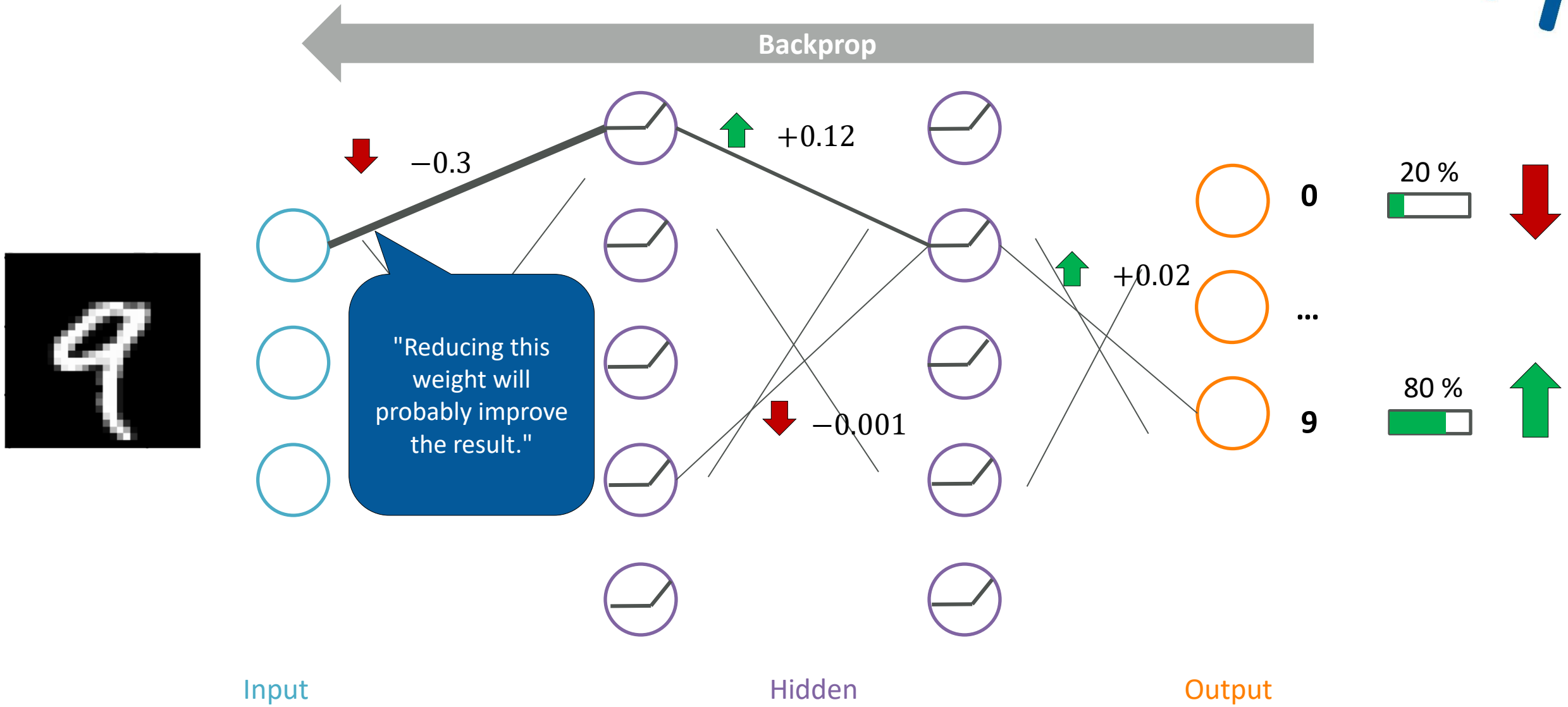
Input Hidden Output

$y = F(x; w)$

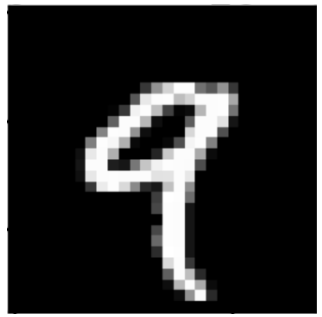
Forward Pass

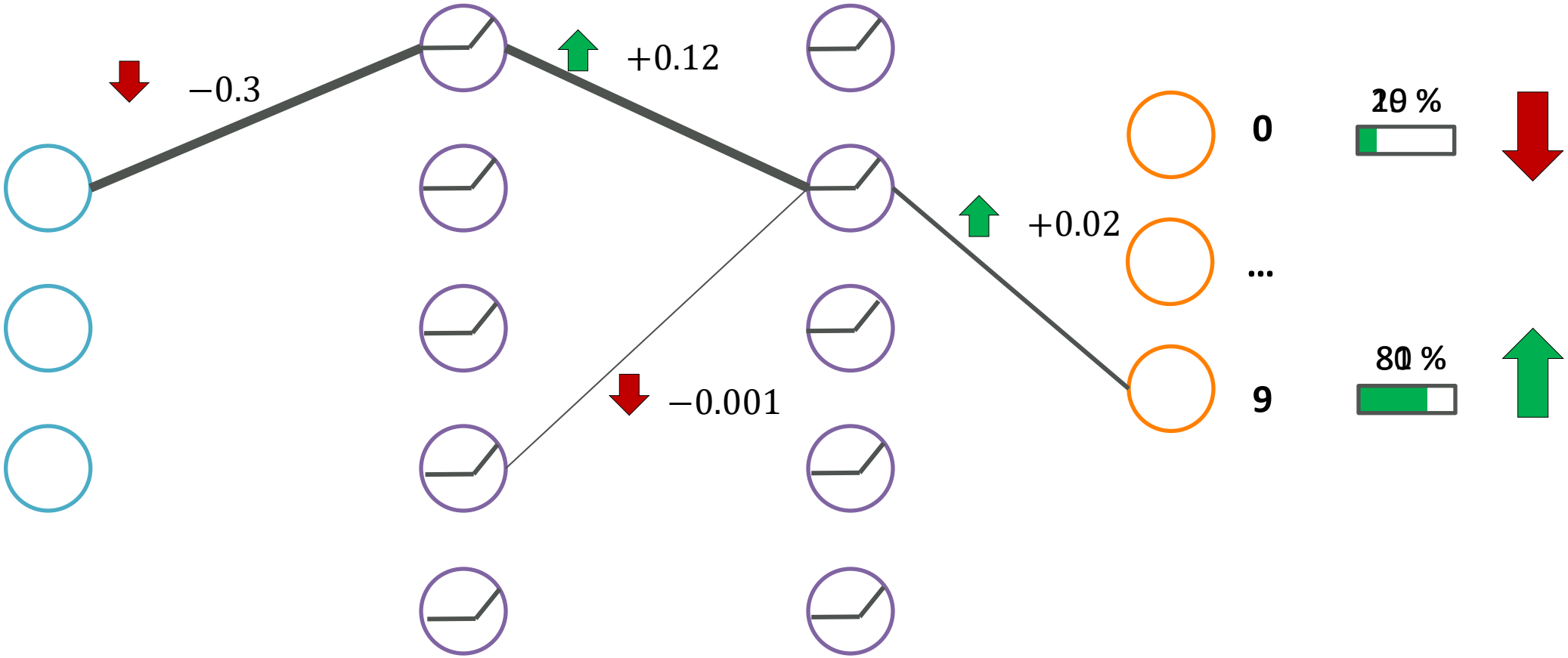


Backpropagation – Gradient $\nabla_w L$



Backpropagation – Gradient update





Input Hidden Output

Convolutional Neural Networks

Guiding Neuroscientific Principles

Hubel & Wiesel

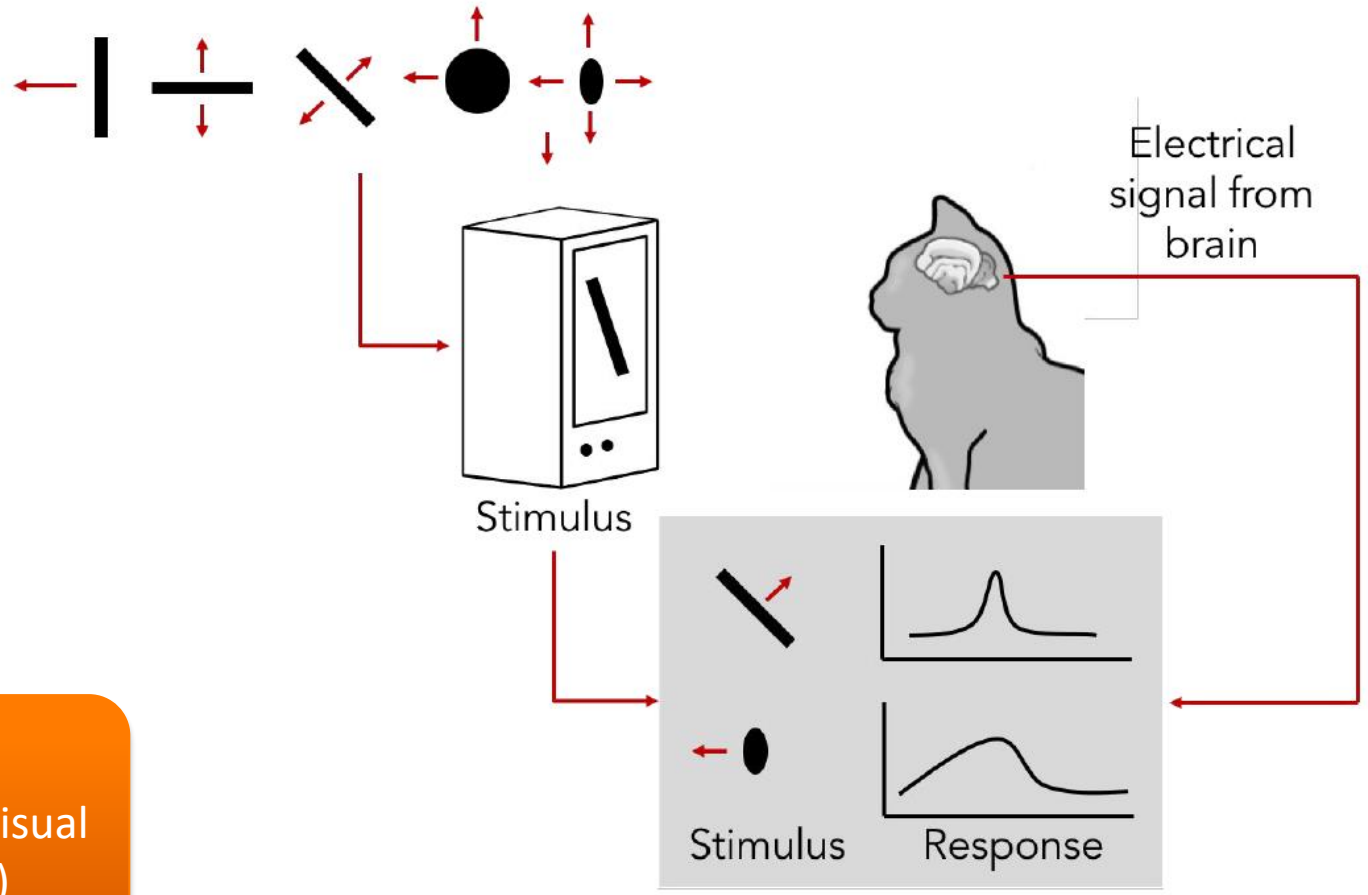
Nobel Prize in Physiology or Medicine in 1981

1959

RECEPTIVE FIELDS OF SINGLE NEURONES IN THE CAT'S STRIATE CORTEX

1962

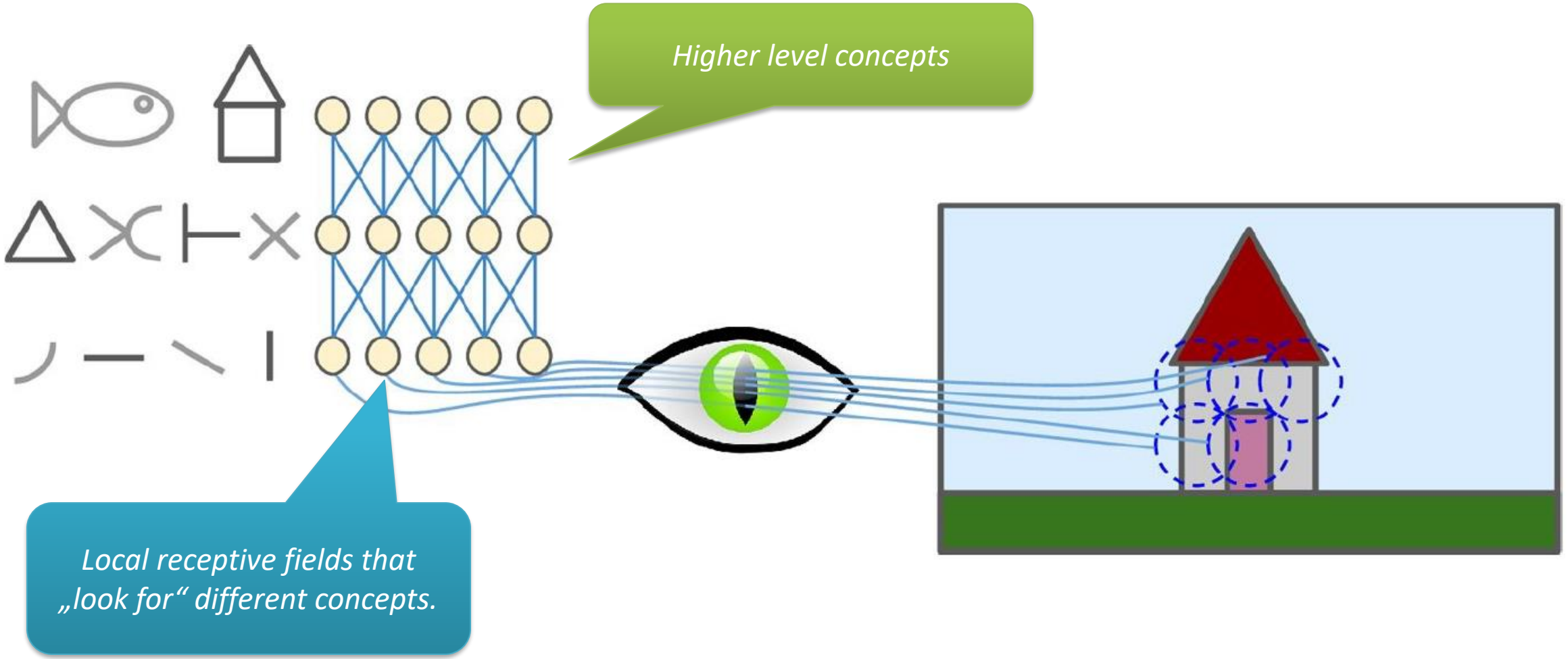
RECEPTIVE FIELDS, BINOCULAR INTERACTION AND FUNCTIONAL ARCHITECTURE IN THE CAT'S VISUAL CORTEX



Concept neurons:

Some neurons only react to certain patterns in the visual field (e.g., „horizontal lines“, „vertical lines“, etc.)

A Cartoon Impression Of Our Visual Cortex



[Hands-On Machine Learning with Scikit-Learn and Tensorflow, Géron, 2017]

Convolutions

Input Image In Channels

Original image (RGB)

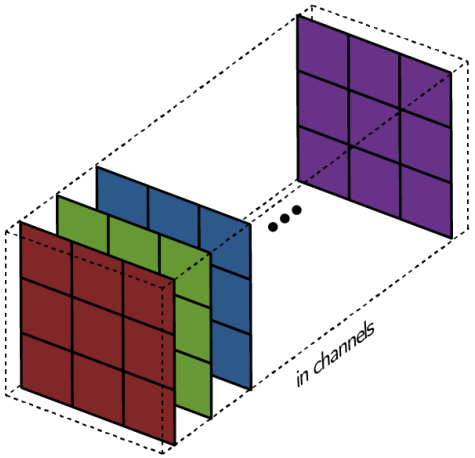
R channel

G channel

B channel

An image is a tensor with dimensions: $\mathbb{R}^{w \times h \times d}$

w(idth)
h(eight)
d(epth)



Discrete Convolutions as feature extractors

Filter kernel

-1	1

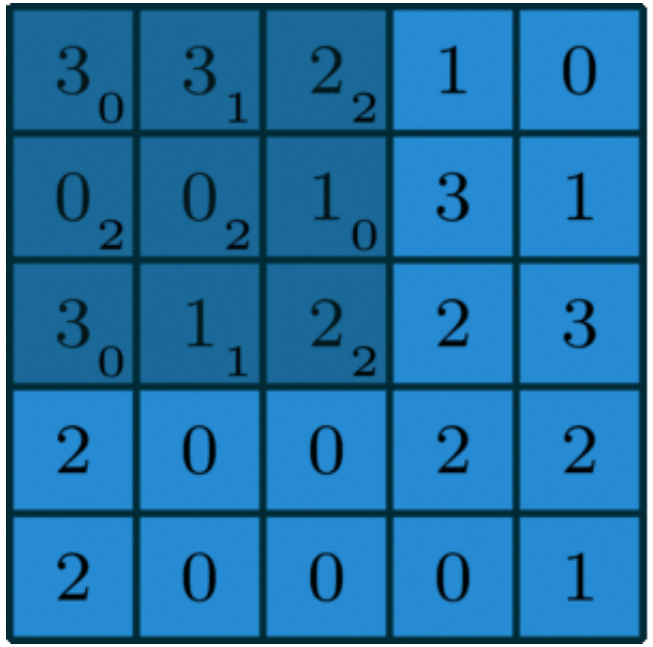
-1 1

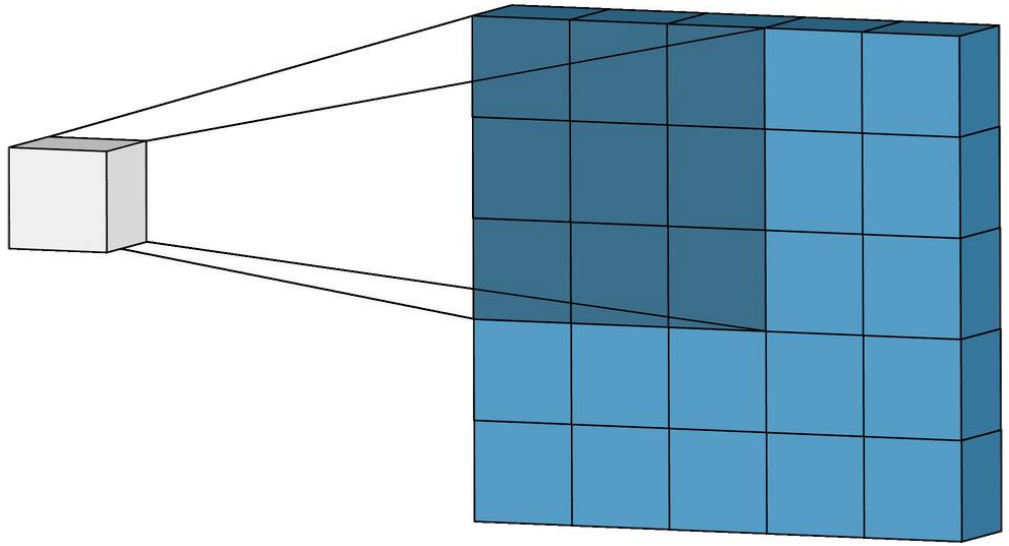
4	7	11	3	12	5

3	4	-8	9	-7

Convolutions: Core Idea

- Exploit **grid-like spatial structure** (1D time series, 2D images, 3D videos, etc.)
- Restrict input locations to neighborhoods (receptive fields)





[https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1]

Convolution

0.4	0.4	0.6	0	0	0	0	0
0	0	0.5	0	0	0	0	0
0	0	0.6	0	0	0	0	0
0	0	0.6	0	0	0	0	0
0	0	0.4	0	0	0	0	0
0	0	0.2	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Convolve with

1	0	-1
1	0	-1
1	0	-1

-1.3	0.4	1.7	0	0	0
-1.7	0	1.7	0	0	0
-1.6	0	1.6	0	0	0
-1.2	0	1.2	0	0	0
-0.6	0	0.6	0	0	0
-0.2	0	0.2	0	0	0

$$0.4 - 0.6 - 0.5 - 0.6 = -1.3$$

$$0.6 + 0.5 + 0.6 = 1.7$$

Convolution „preserves“ shift operations

0.4	0.4	0.6	0	0	0	0	0
0	0	0.5	0	0	0	0	0
0	0	0.6	0	0	0	0	0
0	0	0.6	0	0	0	0	0
0	0	0.4	0	0	0	0	0
0	0	0.2	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

1	0	-1
1	0	-1
1	0	-1

-1.3	0.4	1.7	0	0	0
-1.7	0	1.7	0	0	0
-1.6	0	1.6	0	0	0
-1.2	0	1.2	0	0	0
-0.6	0	0.6	0	0	0
-0.2	0	0.2	0	0	0

0	0	0.4	0.4	0.6	0	0	0
0	0	0	0	0.5	0	0	0
0	0	0	0	0.6	0	0	0
0	0	0	0	0.6	0	0	0
0	0	0	0	0.4	0	0	0
0	0	0	0	0.2	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

1	0	-1
1	0	-1
1	0	-1

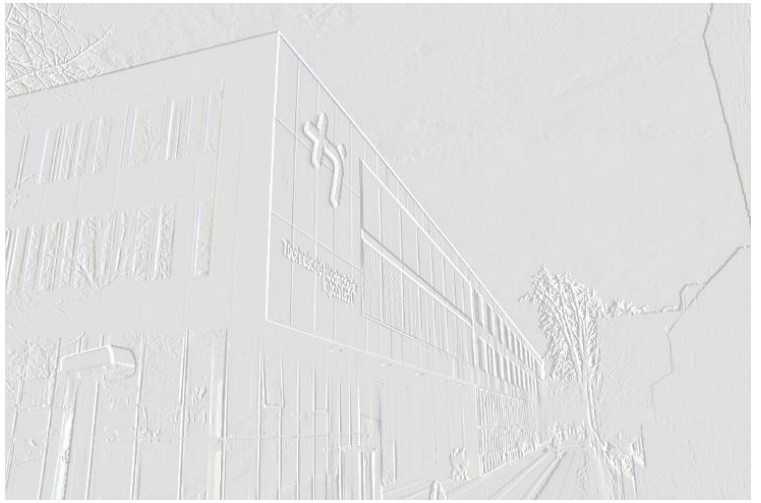
-0.4	-0.4	-1.3	0.4	1.7	0
0	0	-1.7	0	1.7	0
0	0	-1.6	0	1.6	0
0	0	-1.2	0	1.2	0
0	0	-0.6	0	0.6	0
0	0	-0.2	0	0.2	0

Convolution: A Sobel Filter For Edge Detection

I

$$G_x$$

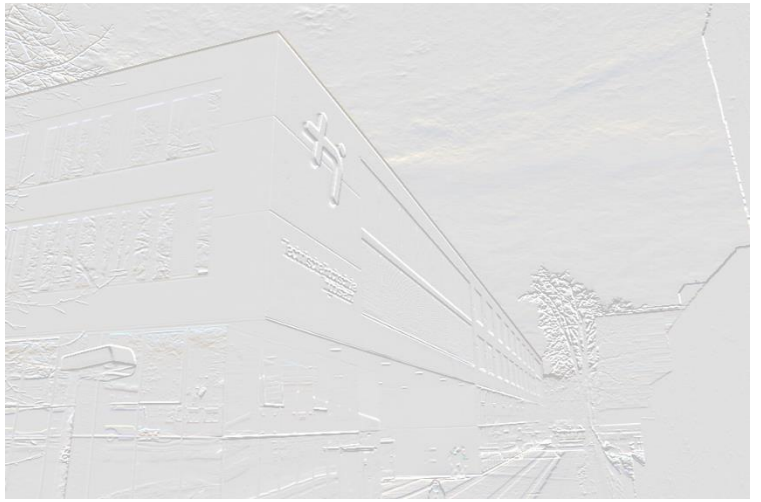
$$* \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$



$I * G_x$

$$G_y$$

$$* \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

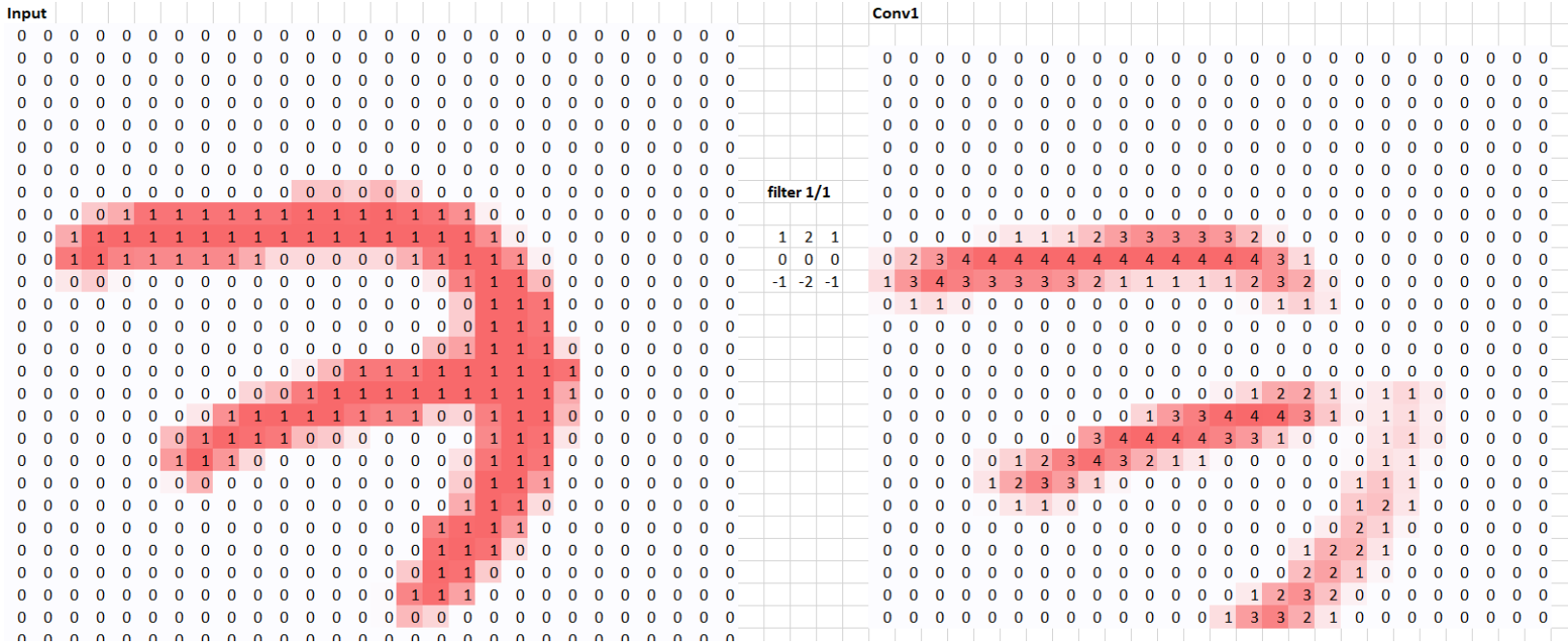


$I * G_y$

Task

Play around with the excel sheet!

https://loma-aimotion.github.io/ml_visualizations/convolution_demo.xlsx

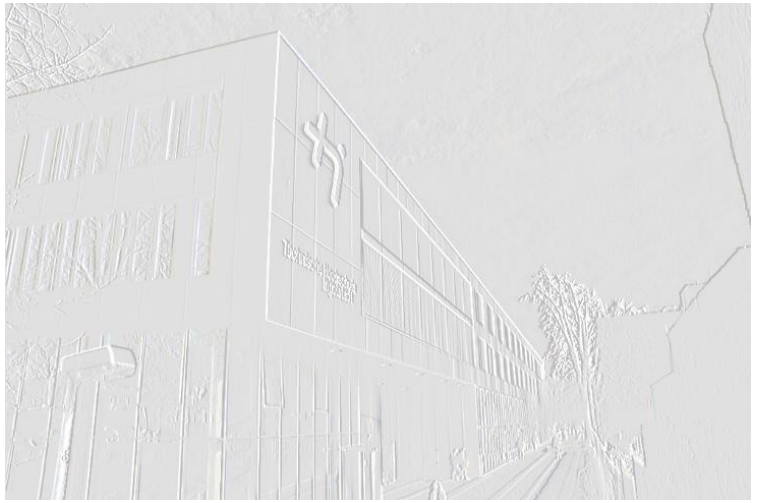


Convolution: A Sobel Filter For Edge Detection

I

$$G_x$$

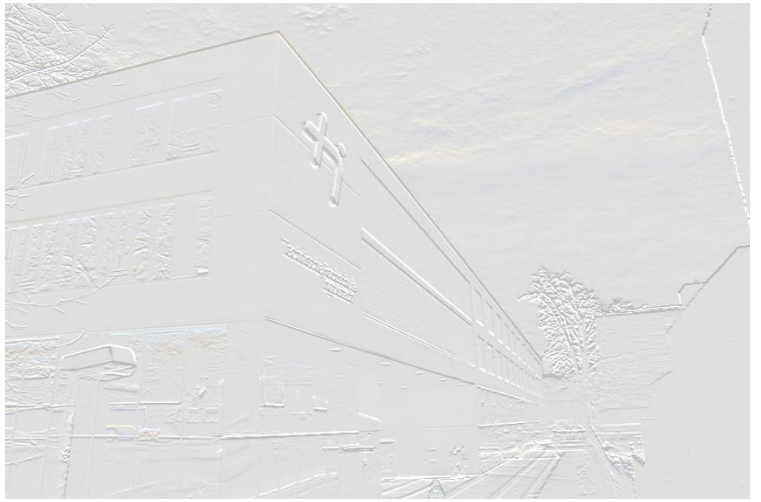
$$* \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$



$I * G_x$

$$G_y$$

$$* \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

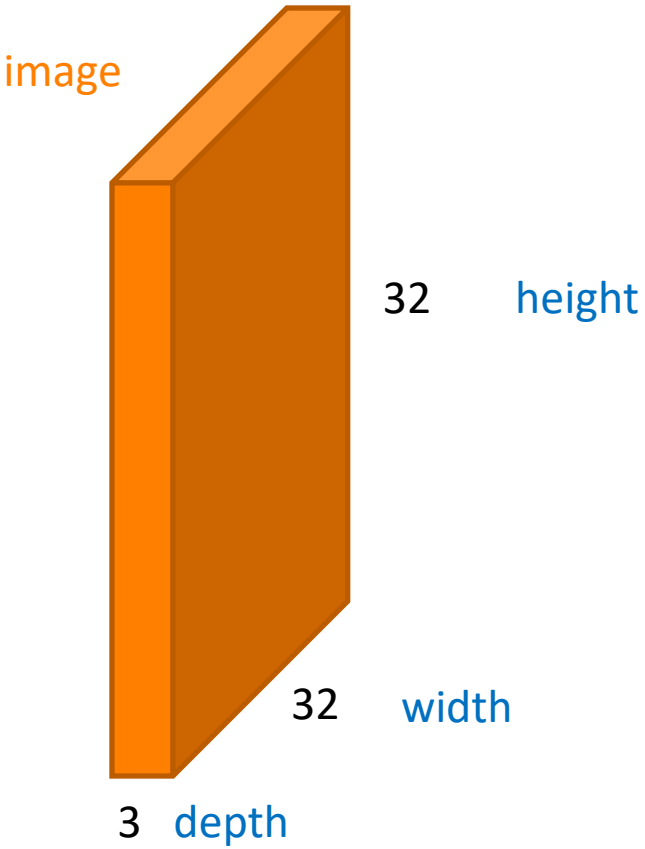


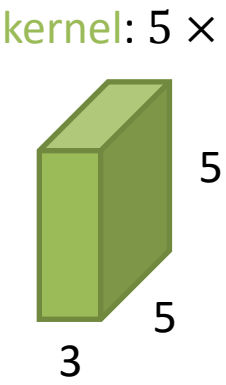
$I * G_y$

Convolutional Layers

Convolutional Layer

Preserve spatial structure of input instead of flattening



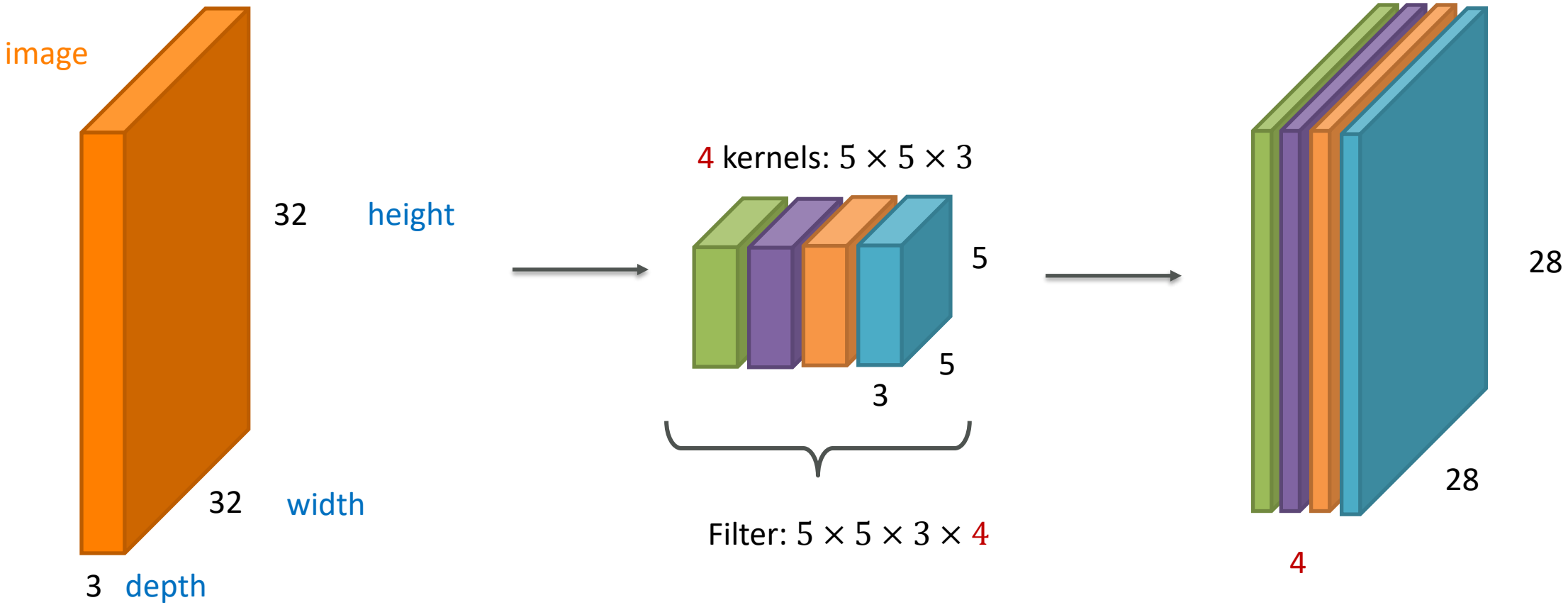


Kernels typically extend through all channels ($d = 3$)!

Convolution means „sliding“ the filter over the image and computing dot products.

Convolutional Layer

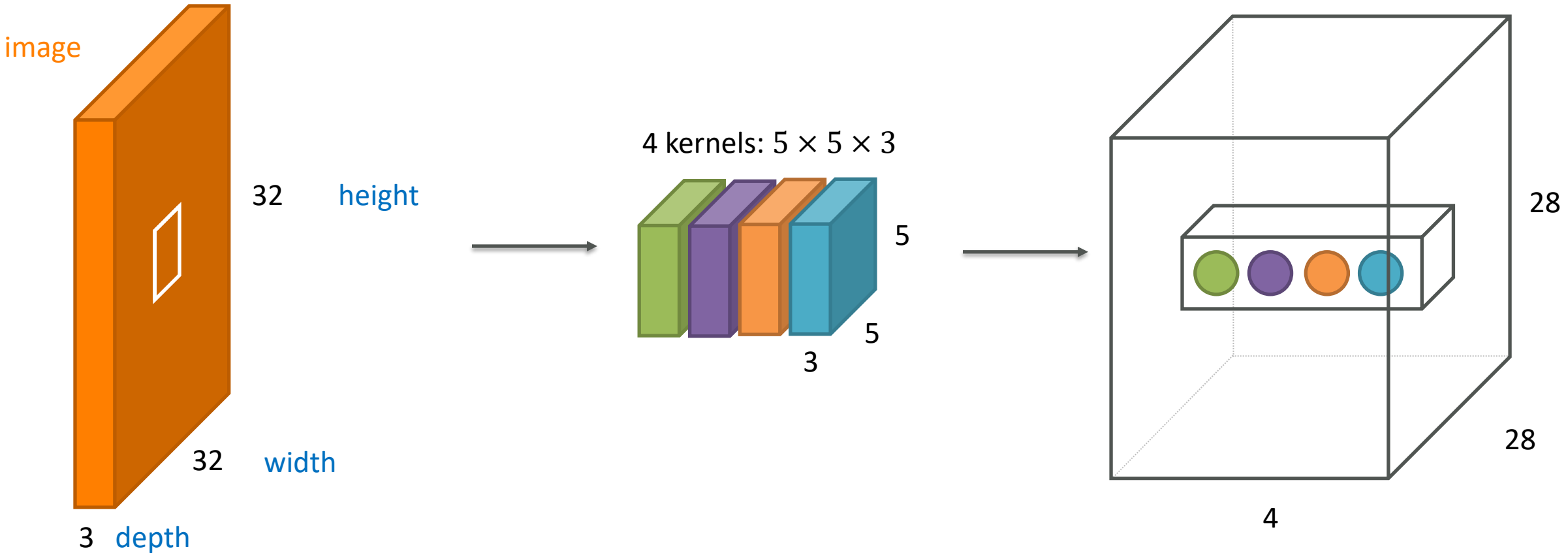
Each kernel results in a different activation map



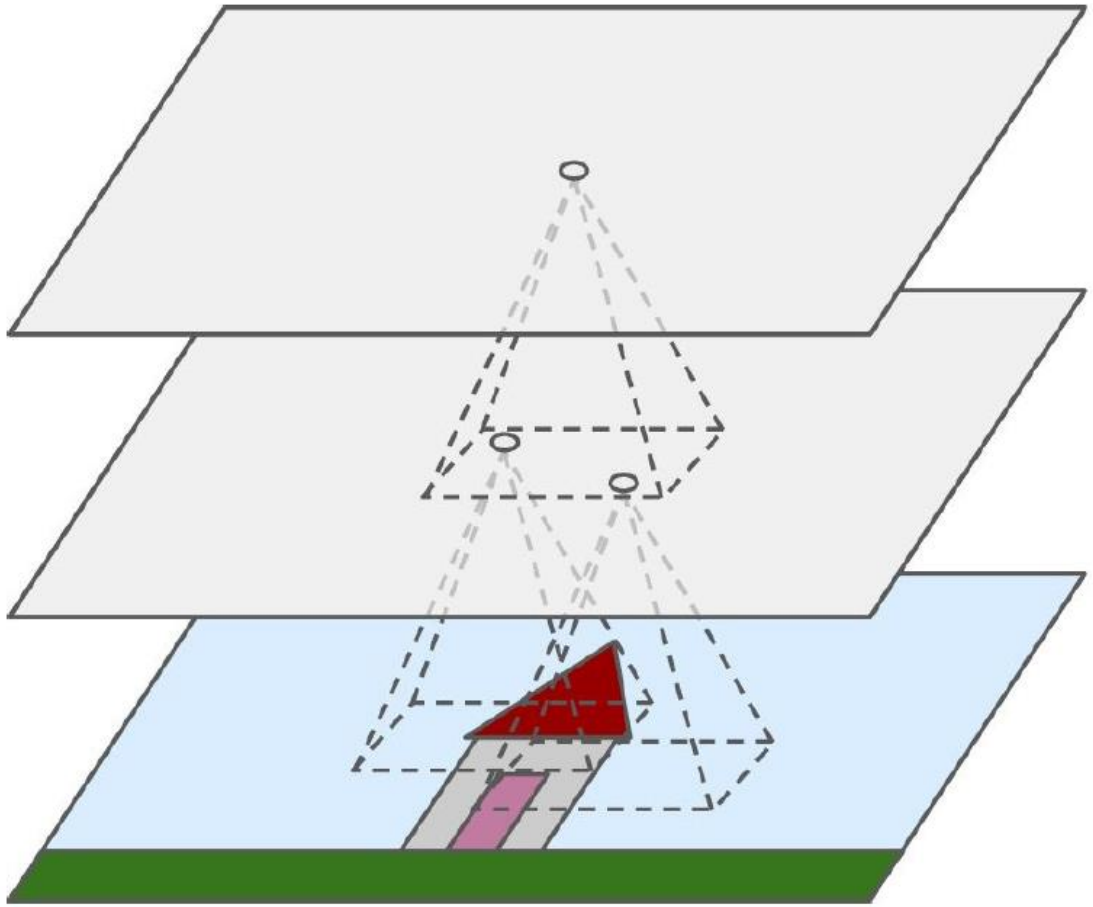
Number of kernels defines number of channels in subsequent layer

Convolutional Layer

Each kernel looks at the same region of the input, but for different things



Interpretation of Convolutional Layers as Receptive Fields



Convolutional layer 2: $w_3 \times h_3 \times d_3$

Convolutional layer 1: $w_2 \times h_2 \times d_2$

Input layer: $w_1 \times h_1 \times d_1$

[Hands-On Machine Learning with Scikit-Learn and Tensorflow, Géron, 2017]

The Computational Graph Of A 1D Convolution

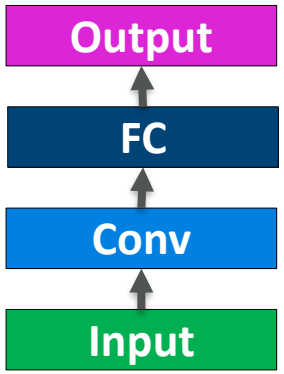
x

t 1.0

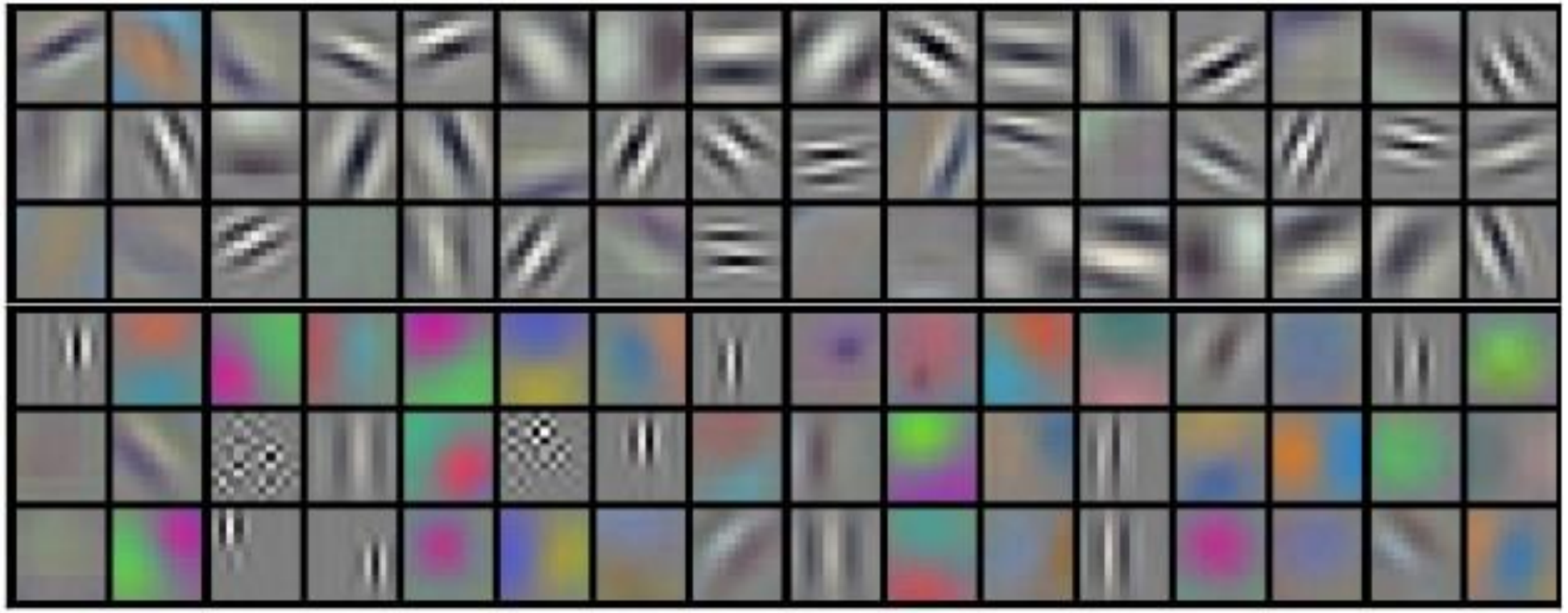
$\boldsymbol{w} = [-1 \quad 1]$

$\boldsymbol{f} = [1 \quad 0.5 \quad 0]$

$y = \boldsymbol{f}^T(\boldsymbol{x} * \boldsymbol{w})$

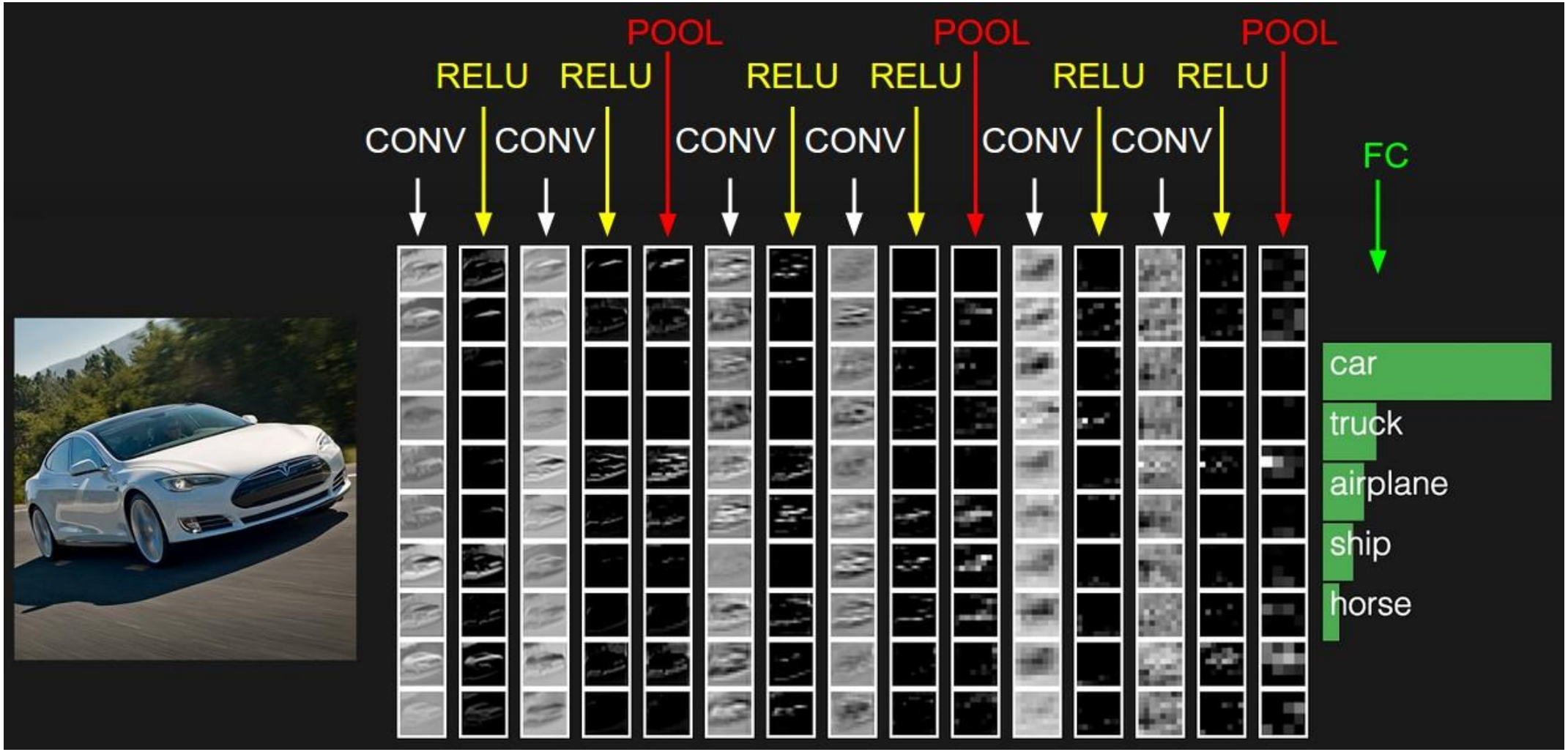


Learned Kernels



[Krizhevsky et al., 2012]

A CNN In Action



Implementation in PyTorch

```python
import torch.nn as nn
import torch.nn.functional as F


class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = x.view(-1, 16 * 5 * 5)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x


net = Net()
```

MaxPool2d

CLASS `torch.nn.MaxPool2d(kernel_size, stride=None, padding=0, dilation=1, return_indices=False, ceil_mode=False)` [SOURCE]

Applies a 2D max pooling over an input signal composed of several input planes.

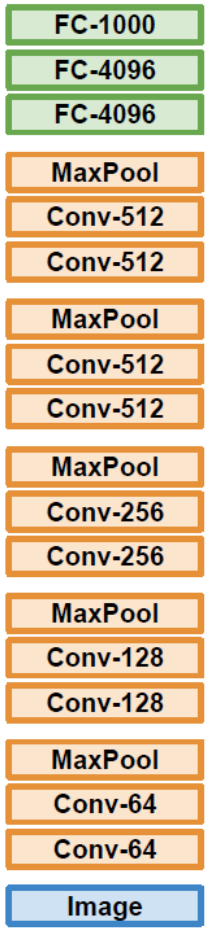
In the simplest case, the output value of the layer with input size (N, C, H, W), output (N, C, H_{out}, W_{out}) and `kernel_size` (kH, kW) can be precisely described as:

$$out(N_i, C_j, h, w) = \max_{m=0,\ldots,kH-1} \max_{n=0,\ldots,kW-1} \text{input}(N_i, C_j, \text{stride}[0] \times h + m, \text{stride}[1] \times w + n)$$

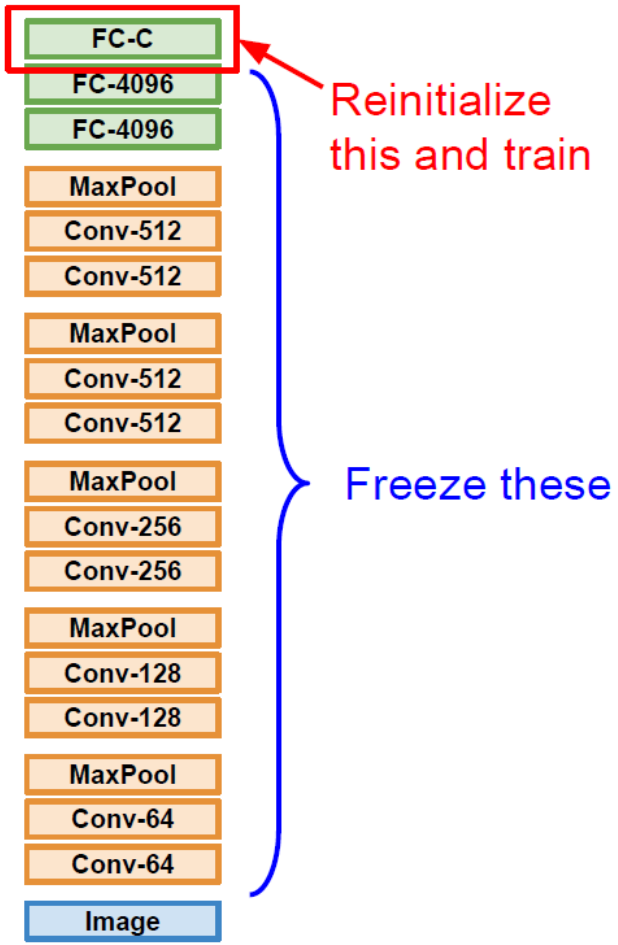
If `padding` is non-zero, then the input is implicitly zero-padded on both sides for `padding` number of points. `dilation` controls the spacing between the kernel points. It is harder to describe, but this link has a nice visualization of what `dilation` does.

Transfer Learning

1. Train on large data set
(or even pre-trained net)



2. Fine-tune: Train the last layer(s) for c custom classes



Later layers (e.g. the FC layers) act as good feature extractors (think very sophisticated basis function expansions).
Freezing means not updating the weights (even if you would get gradients from backprop).

```
import torchvision.models as models
resnet18 = models.resnet18(pretrained=True)
alexnet = models.alexnet(pretrained=True)
squeezenet = models.squeezenet1_0(pretrained=True)
vgg16 = models.vgg16(pretrained=True)
densenet = models.densenet161(pretrained=True)
inception = models.inception_v3(pretrained=True)
googlenet = models.googlenet(pretrained=True)
shufflenet = models.shufflenet_v2_x1_0(pretrained=True)
mobilenet = models.mobilenet_v2(pretrained=True)
resnext50_32x4d = models.resnext50_32x4d(pretrained=True)
wide_resnet50_2 = models.wide_resnet50_2(pretrained=True)
mnasnet = models.mnasnet1_0(pretrained=True)
```

Conclusion

Conclusion

Convolutional layers exploit sparse connectivity and parameter sharing to work with grid-structured data (images, time series)

Convolutional networks consist of at least a single convolutional layer

There is biological inspiration for that kind of local activation

Pooling layers perform subsampling