Karlsruher Institut für Technologie

Machine Learning 1 – Fundamentals

Convolutional Neural Networks

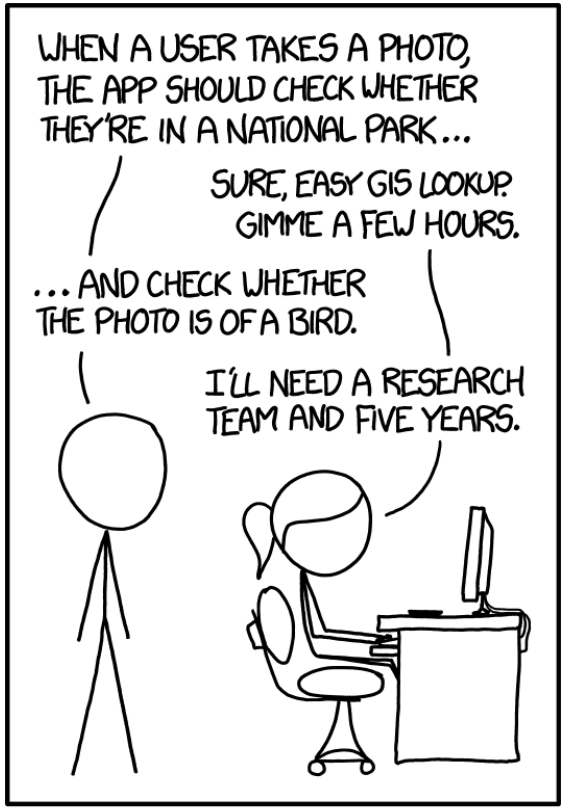
Prof. Dr. J. M. Zöllner, M.Sc. Nikolai Polley, M.Sc. Marcus Fechner

Overview

- Motivation
- Basics
  - Convolution
- Convolutional Neural Networks
  - Layers and Parameters
  - Learning of Filters
  - Important Architectures
- Outlook and Extensions

Image Classification

- XKCD-Comic from 2014
- „Virtually impossible to classify if an image contains a bird“



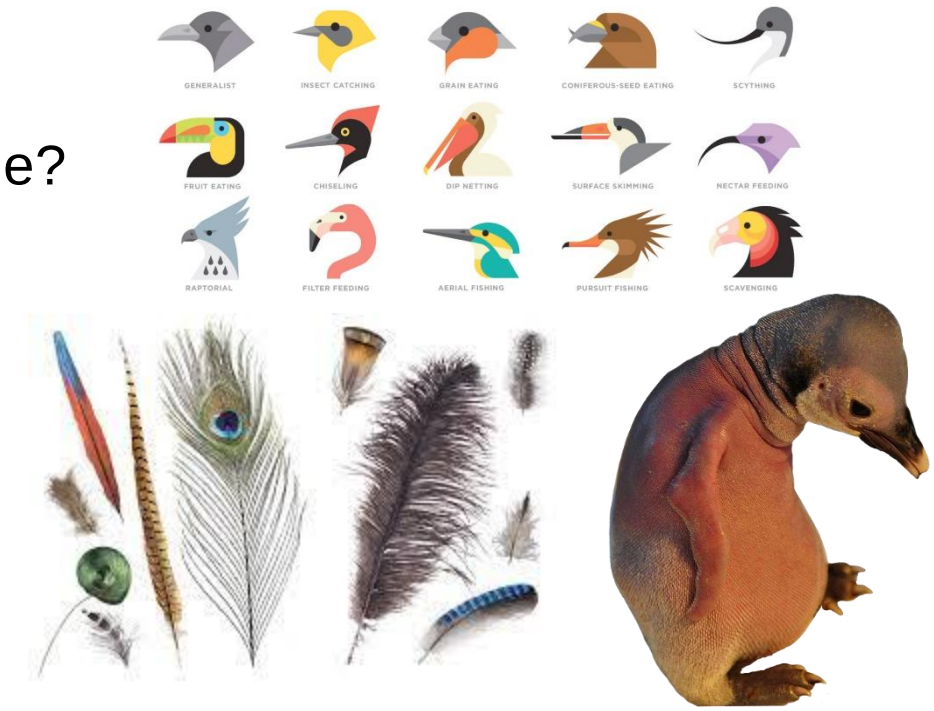
Digital Image

- Pixels are a numeric representation of color intensity
- Does this image contain a bird?

```
67 78 32 95 103 56 73 81 44 65 78 45 96 107 49 96 105 39 101 111 48 50 64 37 106 96 84 65 77 41
30 45 21 22 27 21 100 90 71 92 90 70 46 53 42 140 133 87 76 88 47 75 67 50 166 161 109 160 146 111
144 117 107 50 44 48 61 56 58 146 132 113 161 143 116 148 150 93 155 140 118 107 101 70 172 162 114 171 161 117
155 151 87 111 111 65 131 118 124 78 65 66 78 67 67 165 153 112 150 139 104 107 95 61 167 169 88 157 157 87
132 142 64 115 112 58 91 80 61 105 79 76 120 95 82 78 66 62 115 98 79 102 94 69 173 163 114 150 151 74
142 139 65 108 105 50 130 128 62 121 115 61 68 43 41 108 83 74 69 55 44 106 103 69 133 140 53 144 144 52
147 151 77 122 133 52 142 149 67 145 153 52 126 122 56 37 22 13 88 82 45 98 86 54 121 113 58 117 118 39
139 143 61 142 140 64 129 140 58 146 139 55 156 149 59 163 160 72 152 141 60 118 129 42 120 127 51 107 103 51
133 128 56 134 147 57 135 132 54 138 126 48 147 138 62 157 135 63 145 137 73 126 127 45 125 111 46 142 125 61
110 107 41 136 135 56 129 124 44 124 116 41 161 143 78 150 139 67 139 137 55 133 131 49 170 157 69 161 163 61
```

Image Classification

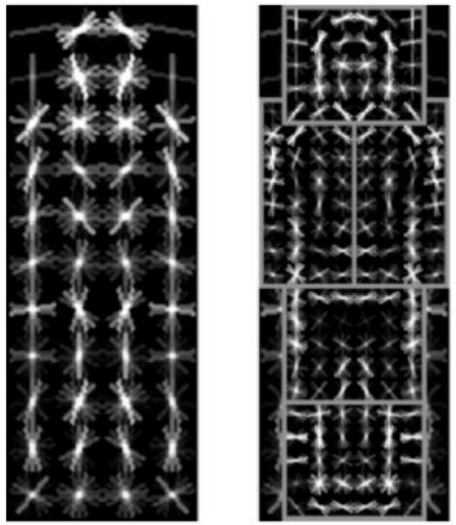
- Thought Experiment: Classify images with regular programming
  - Wikipedia: Birds are […] characterized by feathers and beak
    - Contains beak:
      - How does a beak look like?
      - What if the beak is not visible in image?
    - Contains feathers:
      - Young Birds?

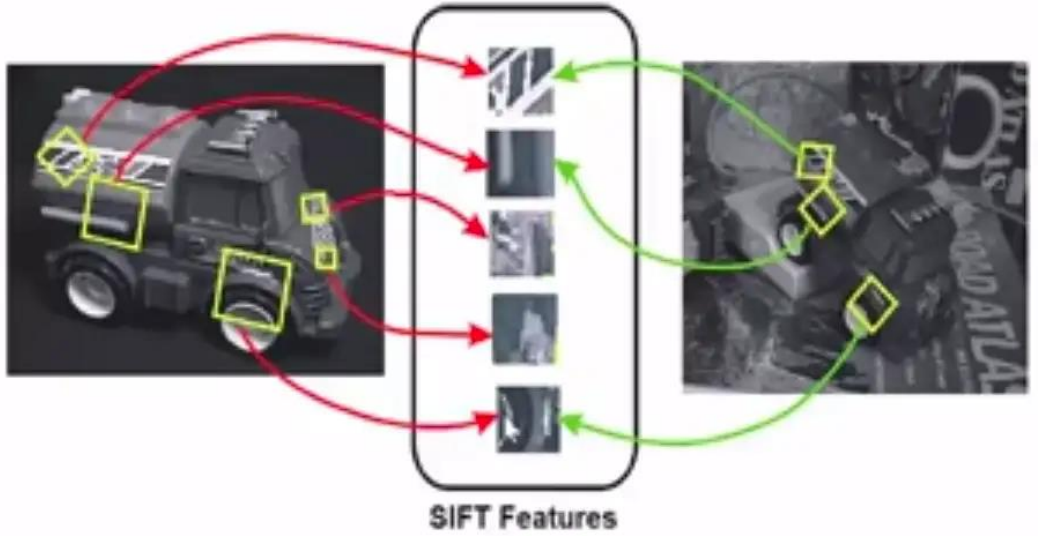


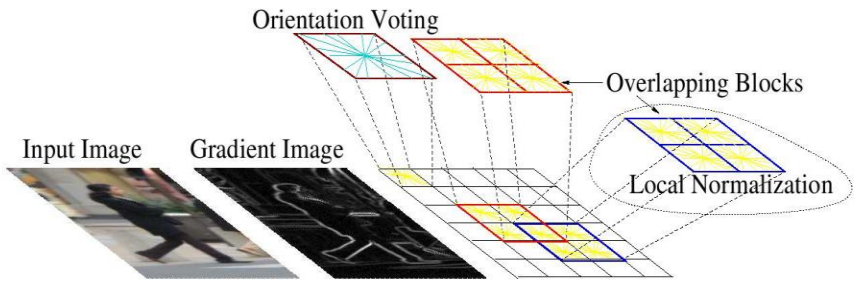
- → We can‘t define it
- → Machine should learn it itself

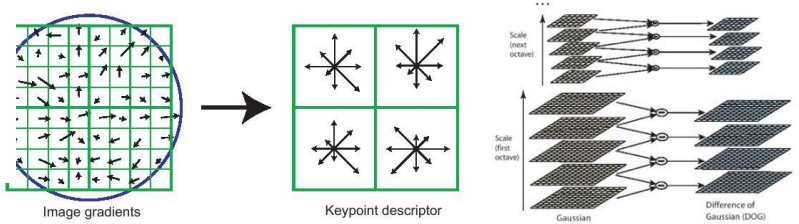
Features

- Until ~2012
  - Manually created features
    - SIFT, HOG



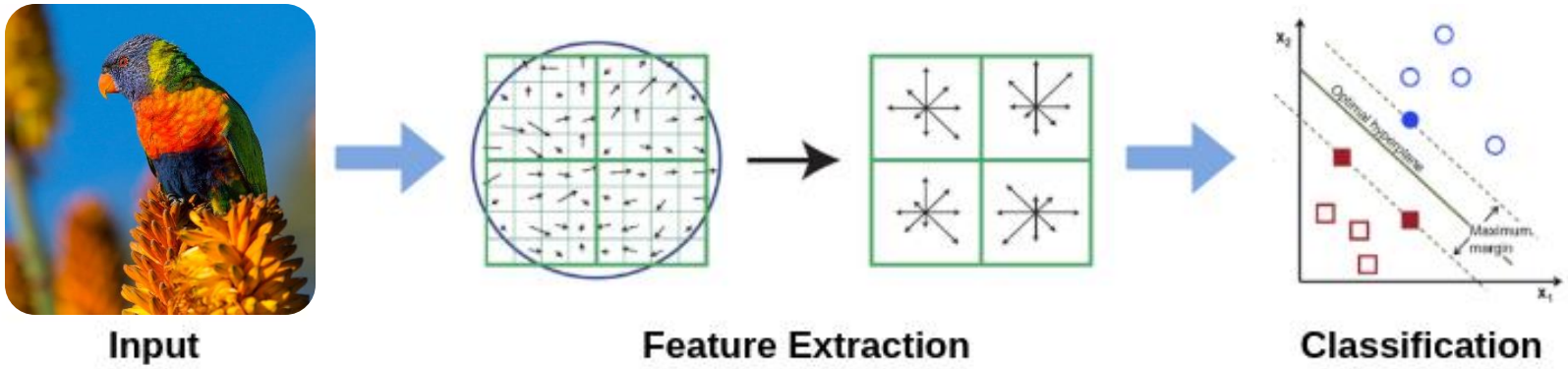




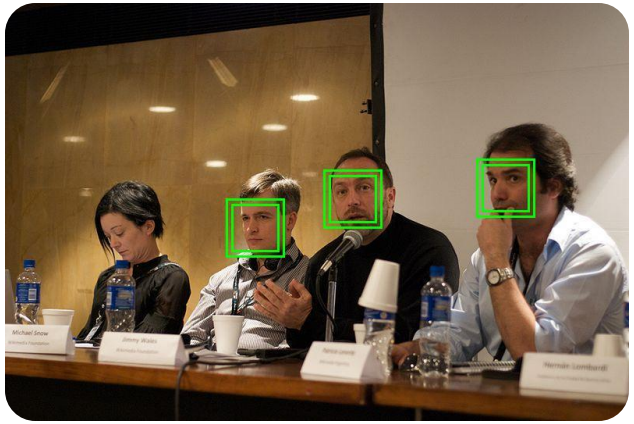


Features

- Classical Pipeline

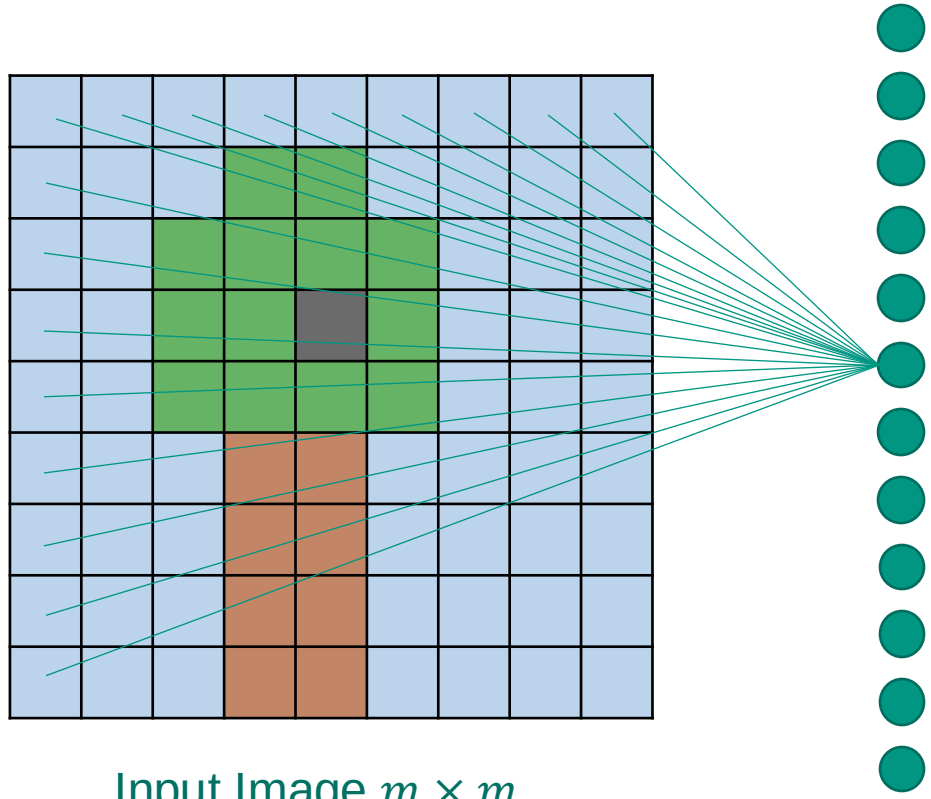


- How to get features
  - Expert crafts features
  - Features only as good as the expert
- Is it possible to learn how to create these features?



Standard Neural Network

- Take pixel values as input
- Each neuron learns weights for each pixel
- Multiple layers of neurons
- Final layer classifies image into possible classes



Input Image $m \times m$

$$y_j = \sum_{i=1}^{m*m*c} w_{ji} x_i + b$$

Layer 1

But there‘s a catch

- Let‘s calculate the size of this model
  - E.g. input is a 512×512 pixel RGB image
    - Each neuron contains ~785.000 parameter
      - $512 \cdot 512 \cdot 3 + 1$
  - 500 neurons in first layer would contain ~400 million parameter

Input Image

Layer 1

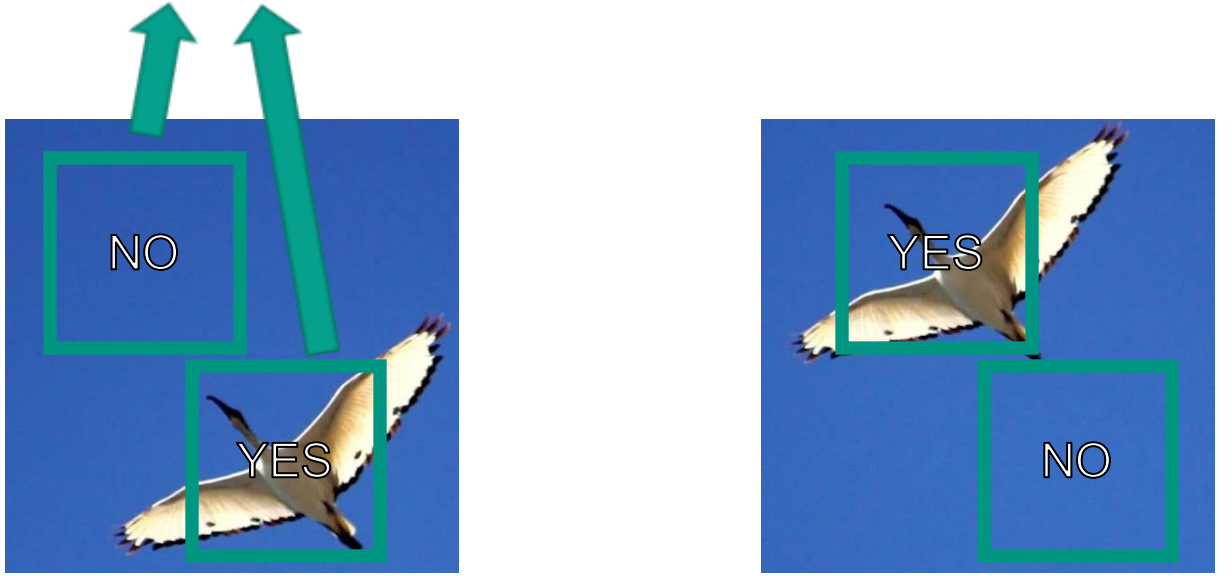
Resolution 512 x 512 Pixel RGB

- Too many parameters!

Translation Invariance

- Neural networks **do not work on variable size input**
- Neural networks are **not translation invariant**
- **Idea**: Use smaller neural network and let it glide over the image

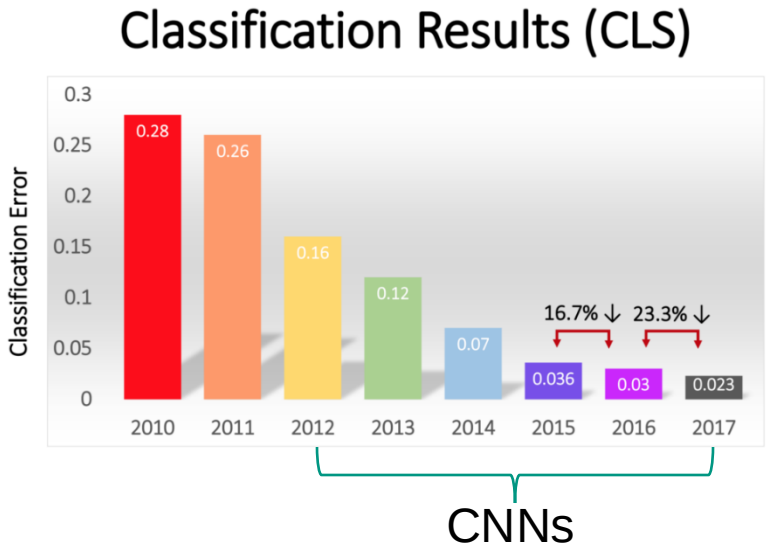
Same neural network with different input

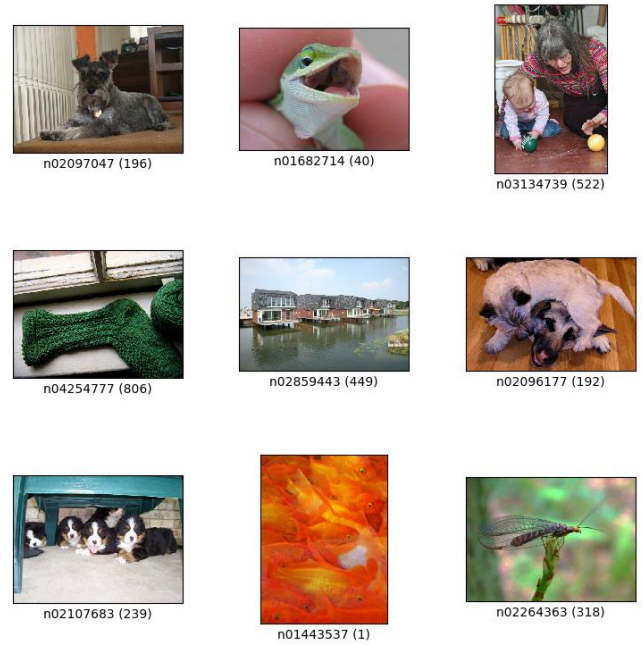


We add inductive biases in neural networks

Rise of CNNs

- ImageNet dataset
  - ~14 million images
  - Each labeled into one of 1000 categories
  - Create algorithm that determines category of image
- After 2012, CNNs beat all other approaches



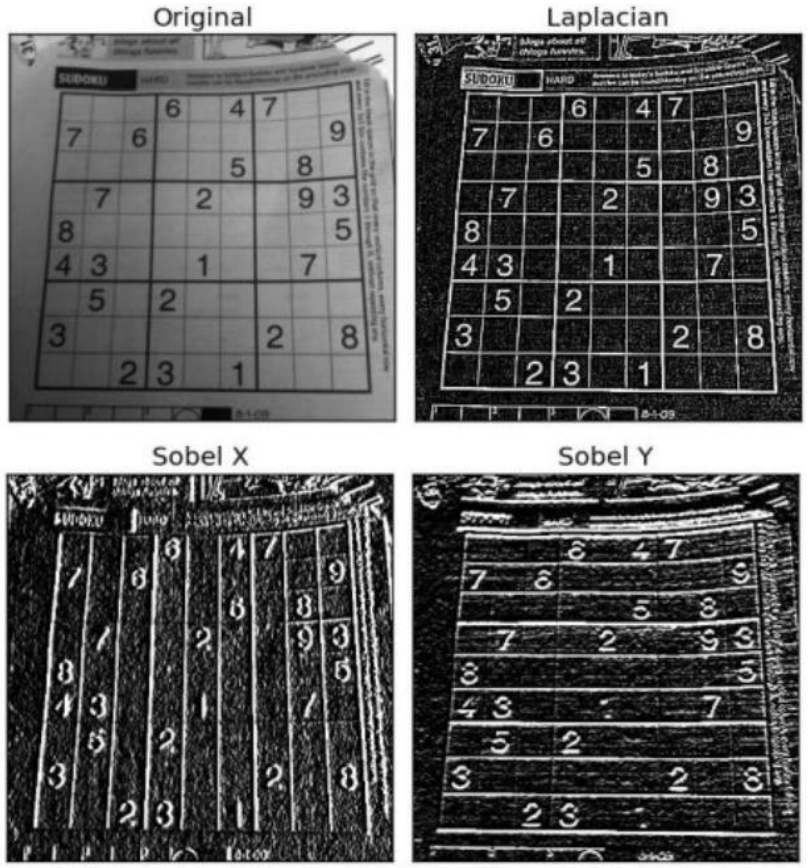


Overview

- Motivation
- Basics
  - Convolution
- Convolutional Neural Networks
  - Layers and Parameters
  - Learning of Filters
  - Important Architectures
- Outlook and Extensions

Convolution: General

- CNNs employ convolutions
- More specific: Discrete Convolutions
- Known from image processing
  - Edge detection
  - Image smoothing
  - Image sharpening

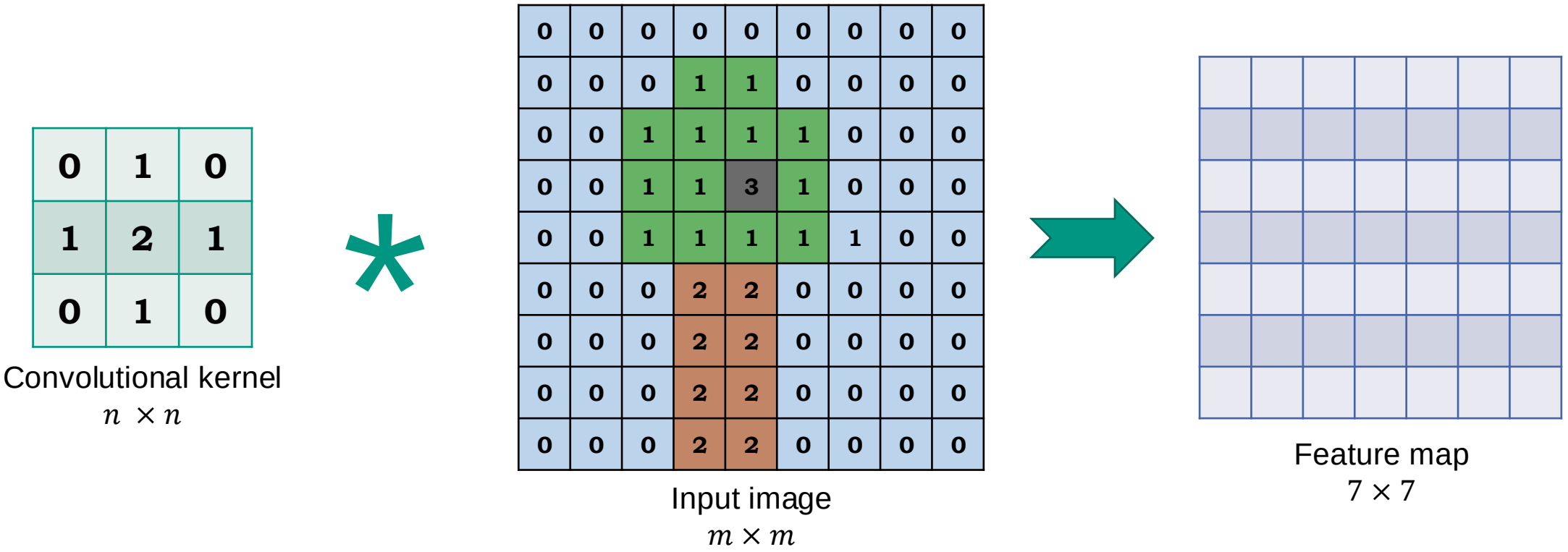


Convolution: Theoretical

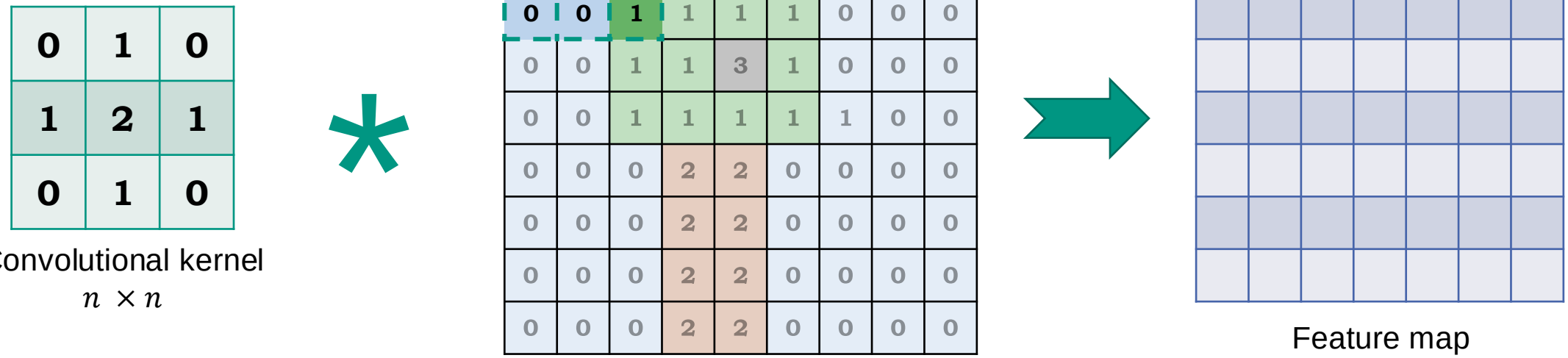
- Image $I(x, y) \in \mathbb{N}^{m_1 \times m_2}$
- Kernel $K(x, y) \in \mathbb{R}^{n \times n}$
- Convolution: $S(x, y) = (K * I)(x, y) = \sum_u \sum_v I(x - u, y - v) K(u, v)$
- Cross-correlation: $S(x, y) = (K * I)(x, y) = \sum_u \sum_v I(x + u, y + v) K(u, v)$
- Convolution = Cross-correlation with flipped kernel

Convolutional Kernel $n = 3$

Convolution: Practical

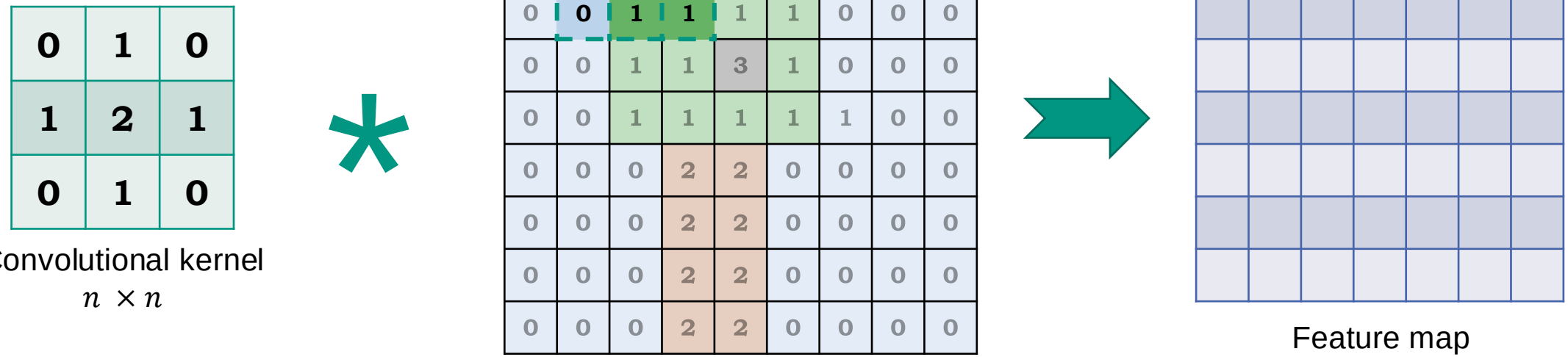


Convolution: Practical



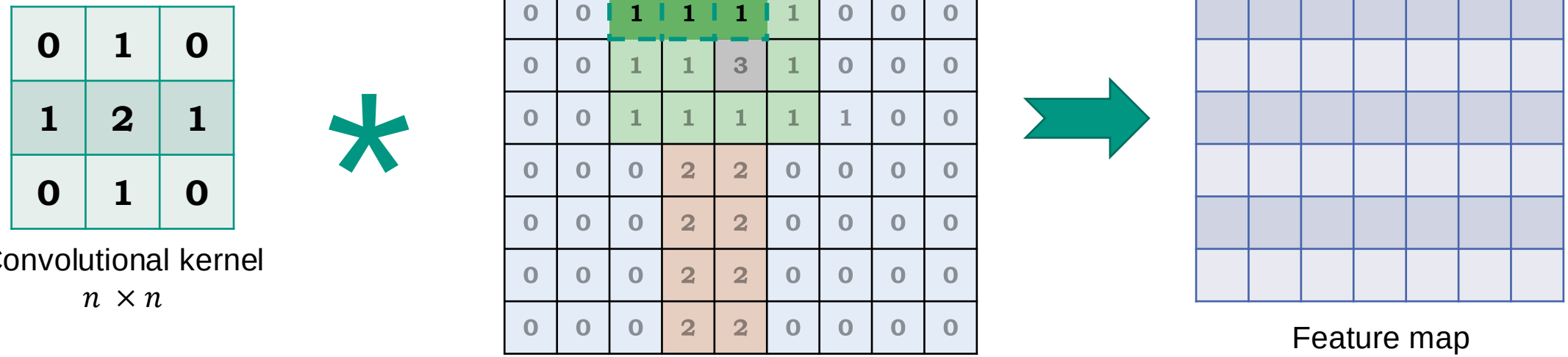
$$0 \cdot 0 + 1 \cdot 0 + 0 \cdot 0 + 1 \cdot 0 + 2 \cdot 0 + 1 \cdot 0 + 0 \cdot 0 + 1 \cdot 0 + 0 \cdot 1 = 0$$

Convolution: Practical



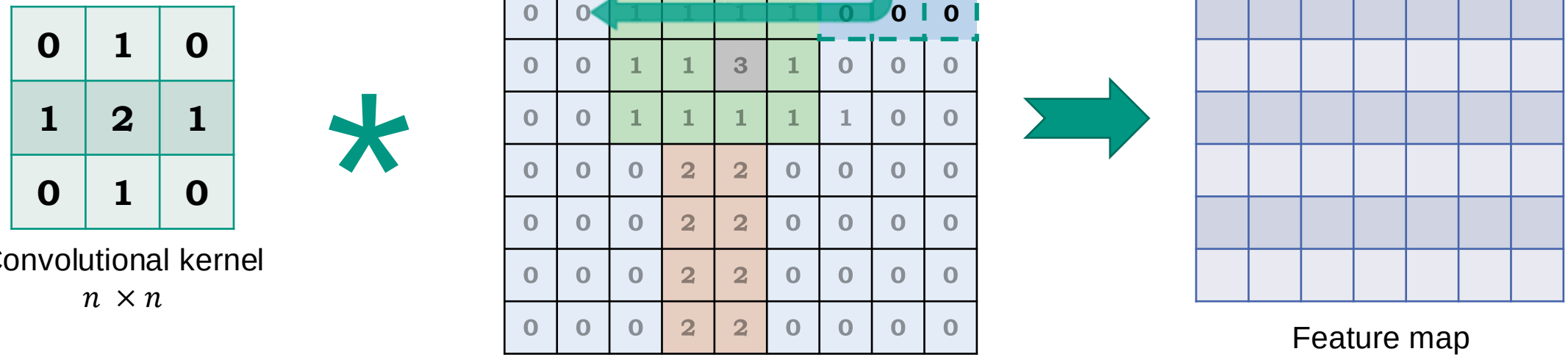
$$0 \cdot 0 + 1 \cdot 0 + 0 \cdot 0 + 1 \cdot 0 + 2 \cdot 0 + 1 \cdot 1 + 0 \cdot 0 + 1 \cdot 1 + 0 \cdot 1 = 2$$

Convolution: Practical



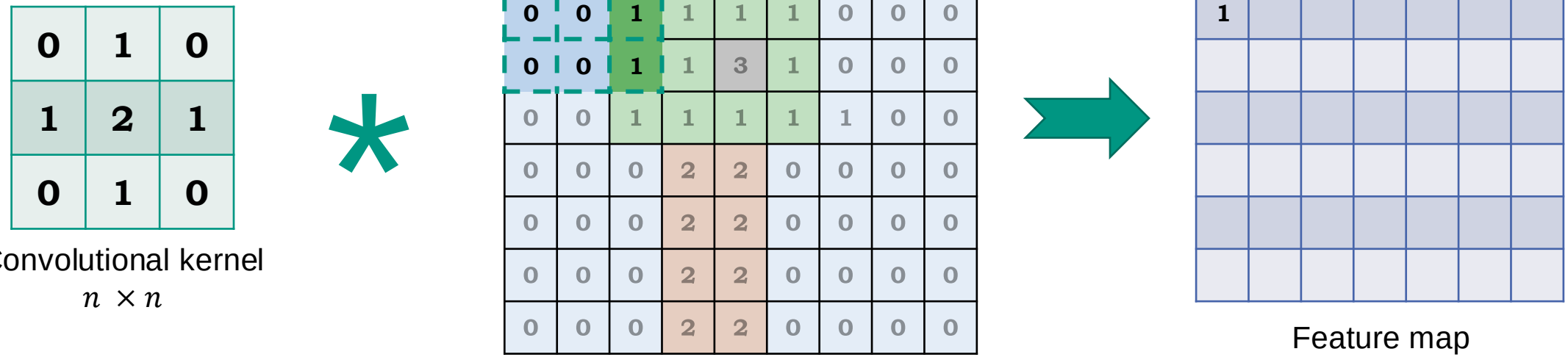
$$0 \cdot 0 + 1 \cdot 0 + 0 \cdot 0 + 1 \cdot 0 + 2 \cdot 1 + 1 \cdot 1 + 0 \cdot 1 + 1 \cdot 1 + 0 \cdot 1 = 4$$

Convolution: Practical



$$0 \cdot 0 + 1 \cdot 0 + 0 \cdot 0 + 1 \cdot 0 + 2 \cdot 0 + 1 \cdot 0 + 0 \cdot 0 + 1 \cdot 0 + 0 \cdot 0 = 0$$

Convolution: Practical

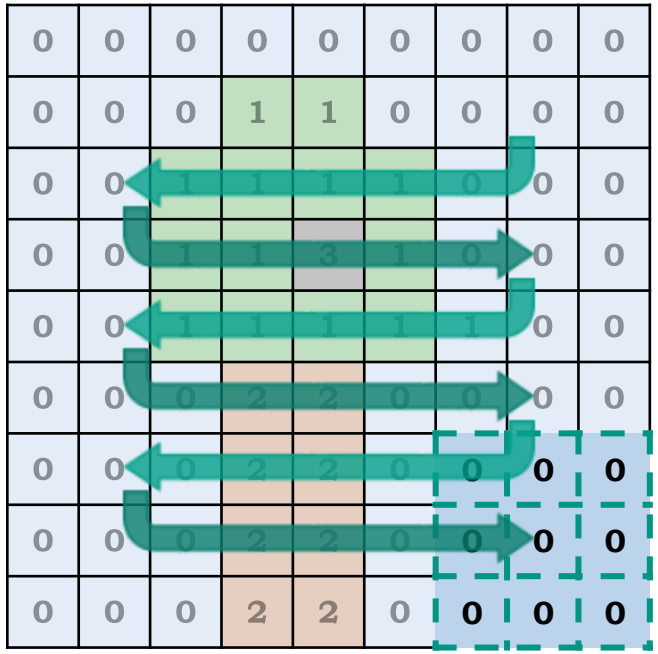


$$0 \cdot 0 + 1 \cdot 0 + 0 \cdot 0 + 1 \cdot 0 + 2 \cdot 0 + 1 \cdot 1 + 0 \cdot 0 + 1 \cdot 0 + 0 \cdot 1 = 4$$

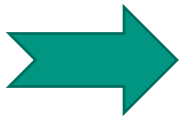
Convolution: Practical

0	1	0
1	2	1
0	1	0

Convolutional kernel
$n \times n$



Input image
$m \times m$

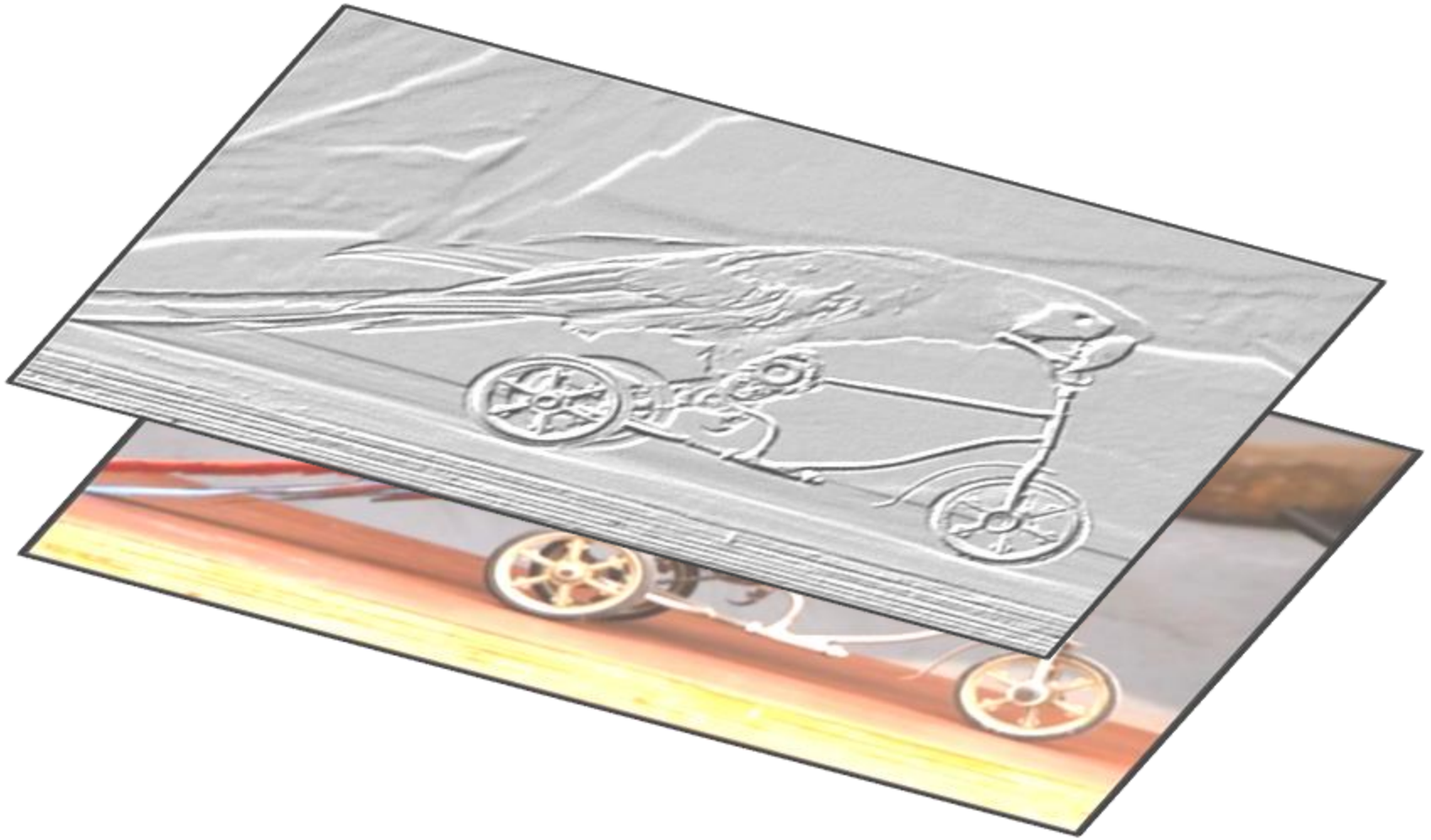


0	2	4	4	2	0	0
1	4	6	8	4	1	0
1	5	8	10	7	1	0
1	4	7	9	4	1	0
0	3	9	9	3	0	0
0	2	10	10	2	0	0
0	2	10	10	2	0	0

Feature map
7×7

$$0 \cdot 0 + 1 \cdot 0 + 0 \cdot 0 + 1 \cdot 0 + 2 \cdot 0 + 1 \cdot 0 + 0 \cdot 0 + 1 \cdot 0 + 0 \cdot 0 = 0$$

Convolution Visualization

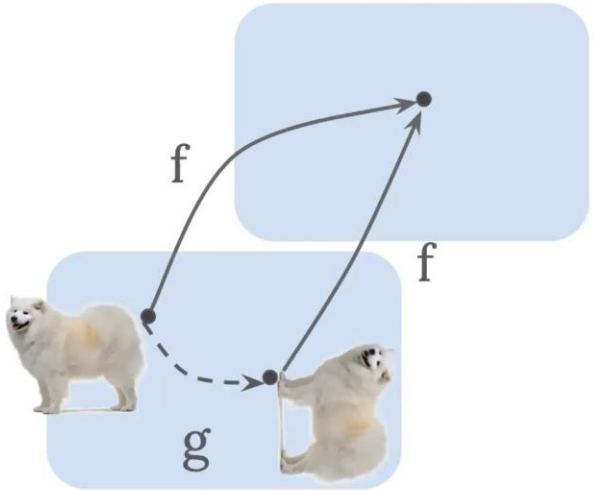


Convolution

Image

Invariance vs Equivariance

- **Invariant** 不变性
  - A function or model is **invariant** to a transformation if its output remains unchanged when the input undergoes that transformation.
  - $f(g(x)) = f(x)$



- **Equivariant** 等变性
  - A function or model is **equivariant** if its output transforms in a predictable, corresponding way when the input is transformed.
  - $f(g(x)) = g'(f(x))$

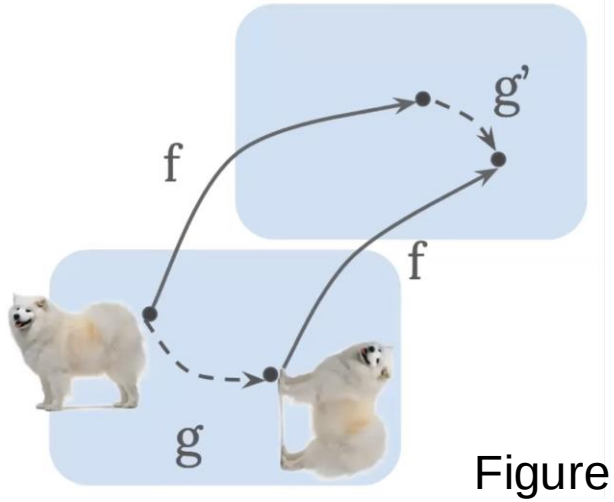
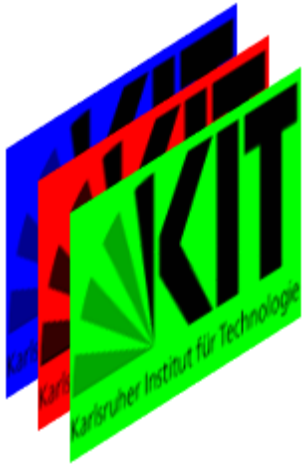


Figure from Elise van der Pol

Convolution: Practical

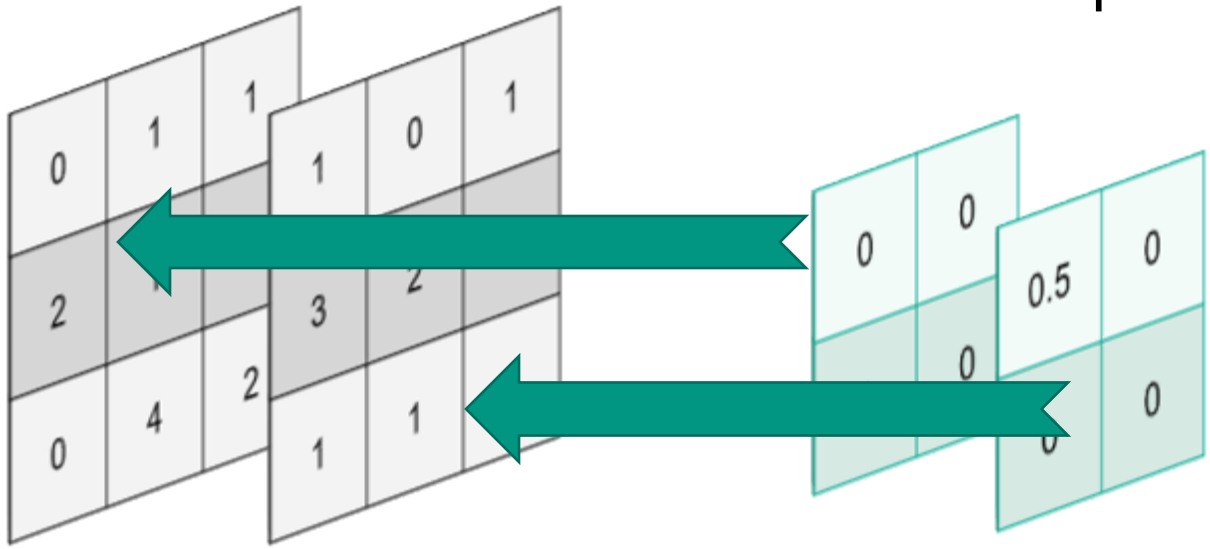
- Until now, we only considered two-dimensional images
  - width × height
- RGB-images are three-dimensional
  - width × height × channels
  - Each color is represented in one channel
    → 3 channels



- Use different convolutional kernels for each channel
  - → Convolutional Filter

Convolution: Practical

- Convolutional filter contains bias and multiple kernels
  - A conv. filter creates a feature map with one channel

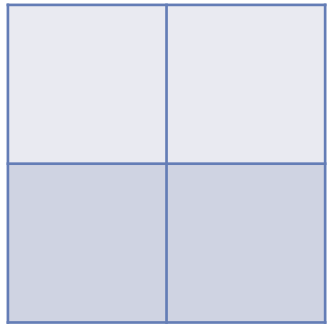


Input image

$m \times m \times \boldsymbol{c}$

One convolutional filter contains
$\boldsymbol{c}$ convolutional kernel

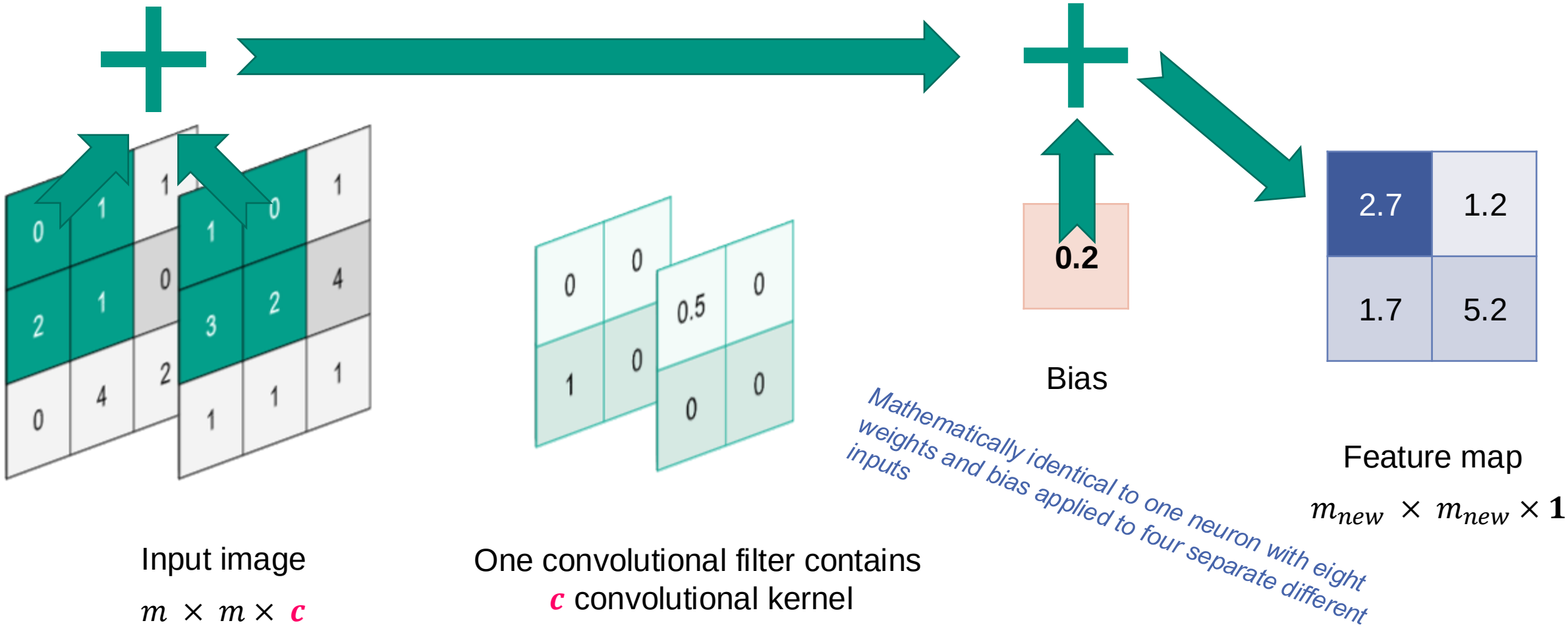
Bias



Feature map

$m_{new} \times m_{new} \times \mathbf{1}$

Convolution: Practical



Input image

$m \times m \times c$

One convolutional filter contains
c convolutional kernel

Bias

Mathematically identical to one neuron with eight weights and bias applied to four separate different inputs

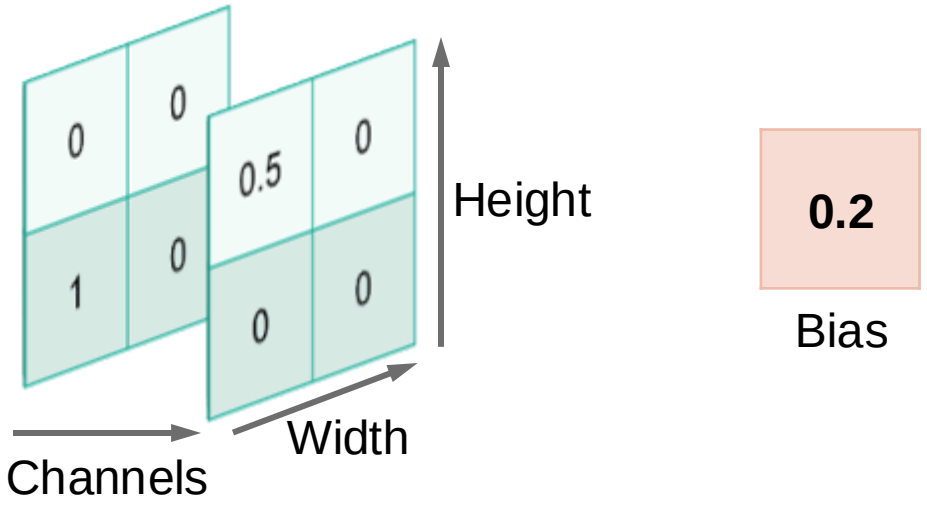
Feature map

$m_{new} \times m_{new} \times 1$

Trainable Parameters in Convolutional Filter

- Learn the weights and bias for these convolutional filters with backpropagation
  - Width w of kernel
  - Height h of kernel
  - Number kernels = Number of channels c of input
  - One bias

$$No.params = w \times h \times c + 1$$



Overview

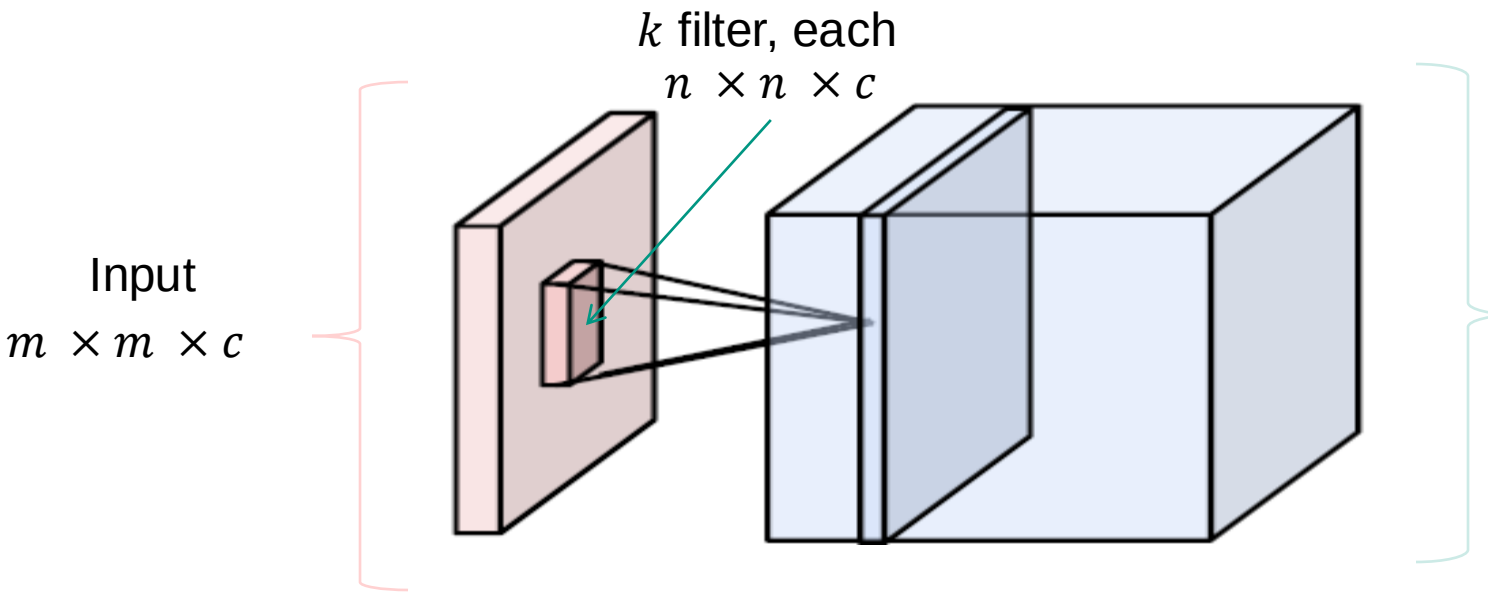
- Motivation
- Basics
  - Convolution
- Convolutional Neural Networks
  - Layers and Parameters
  - Learning of Filters
  - Important Architectures
- Outlook and Extensions

Convolutional Layer

- Consists of **multiple** convolutional filters

- Input: Previous feature maps
- Output: Newly created feature maps

In the following, we assume square feature maps $m \times m$. If width and height differ, the formulas must be calculated separately for each dimension.



Output size

$$\boldsymbol{m_{new} \times m_{new} \times k}$$

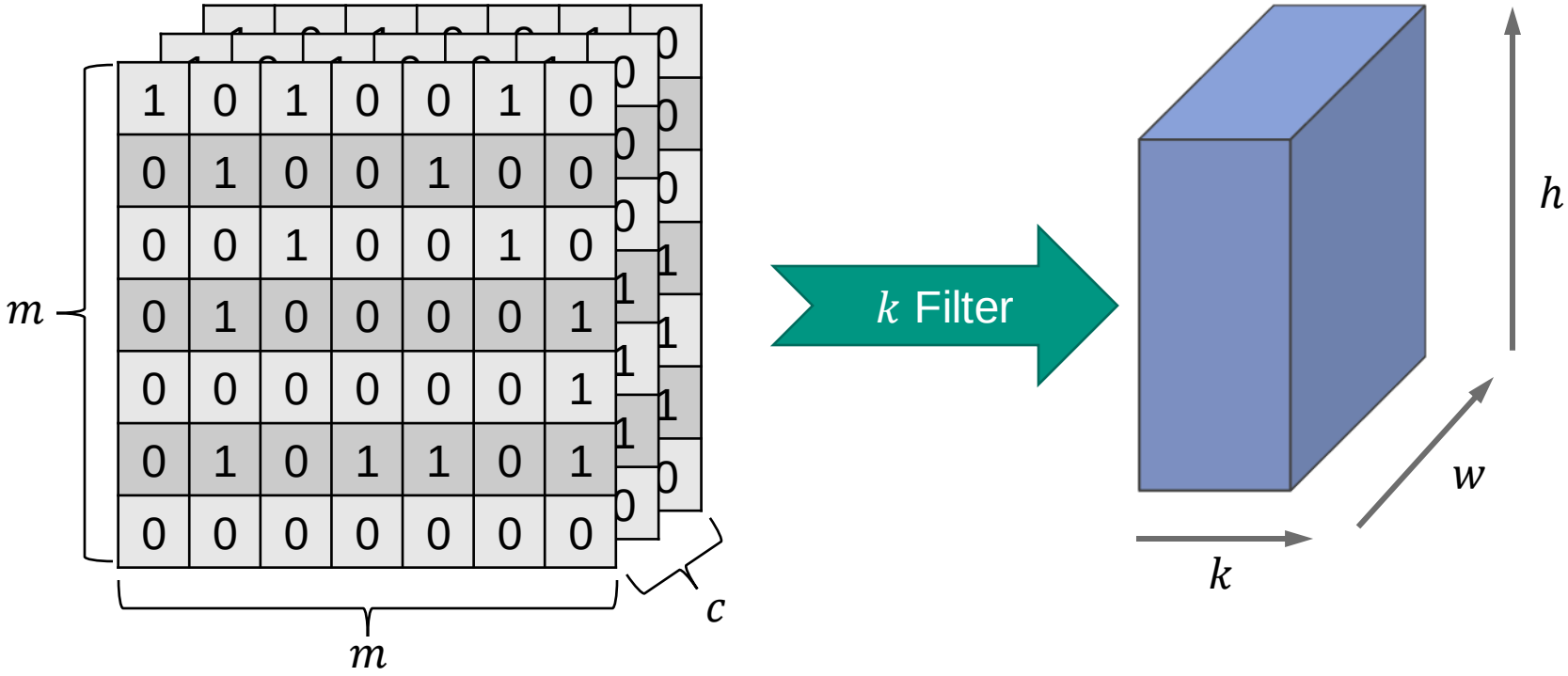
whereas

$$\boldsymbol{m_{new} = \frac{(m - n)}{s} + 1}$$

s = stride

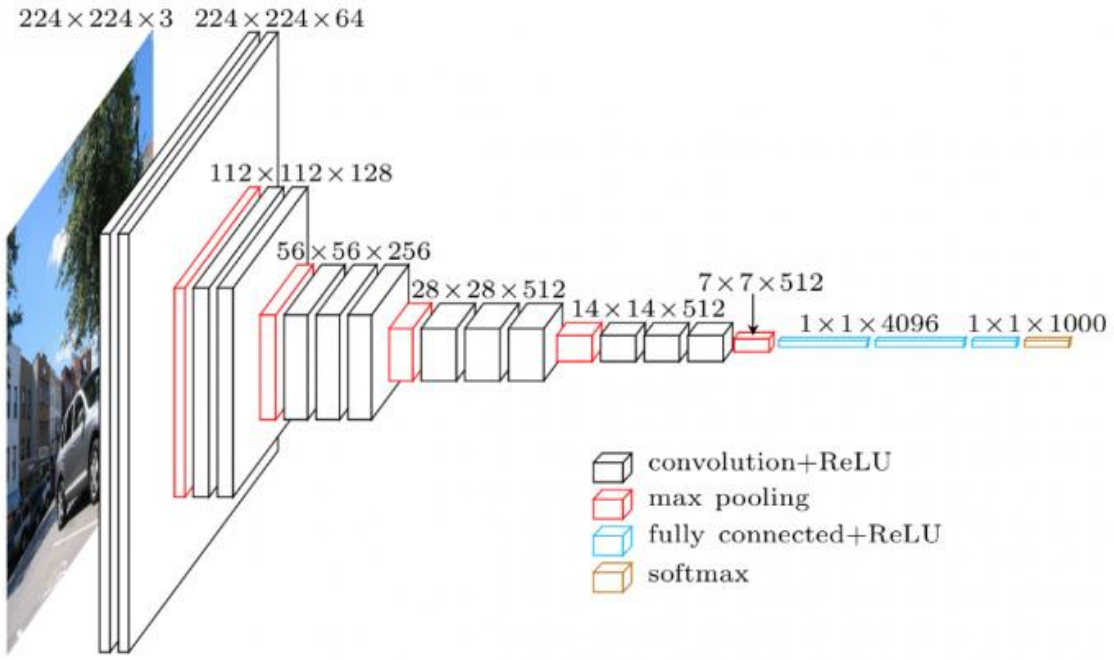
Dataflow in CNN

- The output of convolutional filters are called feature maps
  - k convolutional filter create k feature maps $w \times h \times k$
  - RGB-input image ≡ feature map with 3 channels



Standard CNN Architecture

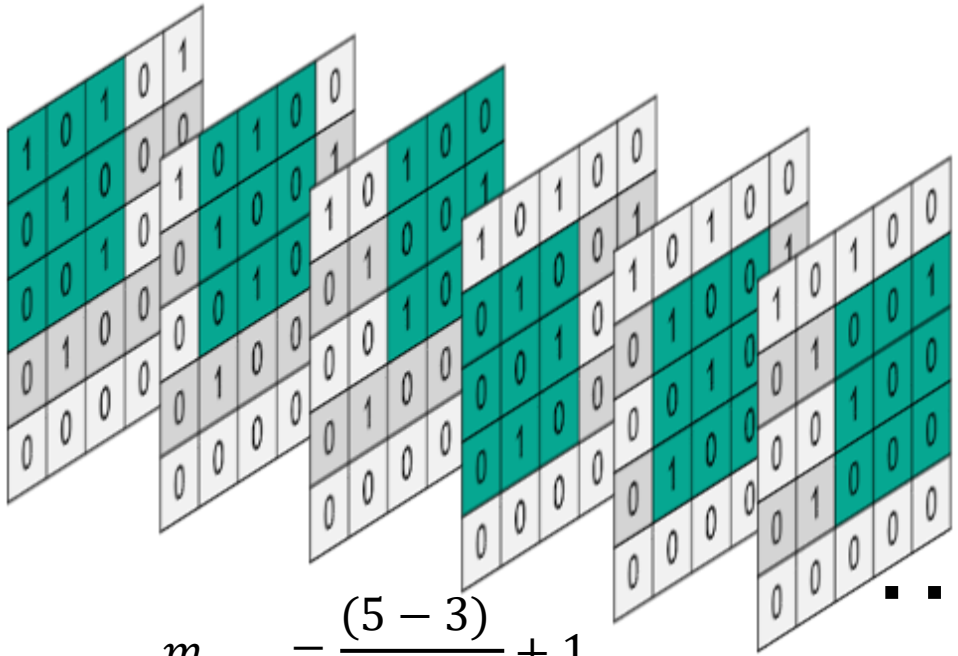
- Architecture usually divided into 5 stages
  - Each contains multiple conv. layers
    - After each layer → activation function
  - Spatial feature map resolution stays constant in each stage
  - Quartering spatial resolution after each stage
  - Doubling filters after each stage 每个阶段后加倍过滤



[M. Loukadakis]

Strided Kernels

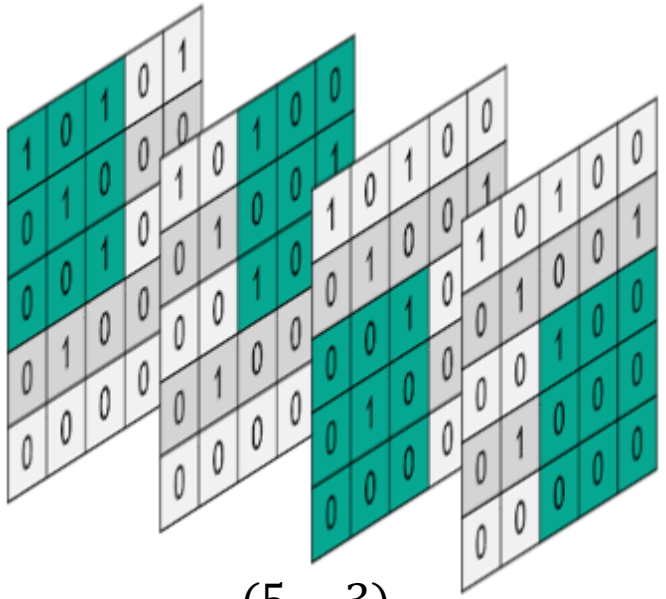
- Stepping distance of kernel
  - In both dimensions
  - Decreases feature map resolution



$$m_{new} = \frac{(5-3)}{1} + 1$$

. . .

$$\boldsymbol{m_{new} = \frac{(m-n)}{s} + 1}$$



$$m_{new} = \frac{(5-3)}{2} + 1$$

Convolutional Filter: Padding

- Conv. filter using kernels larger than 1×1 reduce resolution
- Pixels on edges never seen with middle value of the kernel
- Staying on same resolution oftentimes preferable
- Strided conv. filter might ignore values

1	0	1	0
0	1	0	0
0	0	1	0
0	1	0	0

3x3 Kernel with stride 2 ignores right column and lower row

Convolutional Filter: Padding

- Increase resolution of input feature map
  - **Zero Padding**
    - Add zeros at the edges
  - Reflect Padding
    - Mirrors inner feature map
  - Circular Padding
    - Take values from the opposite side
- Creates false data at the edges of feature maps
- Reminder: $m_{new} = \frac{(m-n)}{s} + 1$
  - $m = \frac{(\boldsymbol{m}+\boldsymbol{2}-3)}{1} + 1$ keeps resolution constant for 3×3 Kernel

0	0	0	0	0	0	0	0	0
0	1	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0	0
0	0	0	1	0	0	1	0	0
0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	0	1	0
0	0	1	0	1	1	0	1	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

Zero Padding

2	9	2	0	4	1	9	0	9
8	4	8	6	4	6	0	7	0
2	9	2	0	4	1	9	0	9
6	4	6	0	7	7	4	6	4
2	6	2	9	6	5	5	6	5
3	3	3	2	3	5	4	8	4
9	6	9	6	2	1	5	3	5
1	3	1	9	6	2	3	0	3
9	6	9	6	2	1	5	3	5

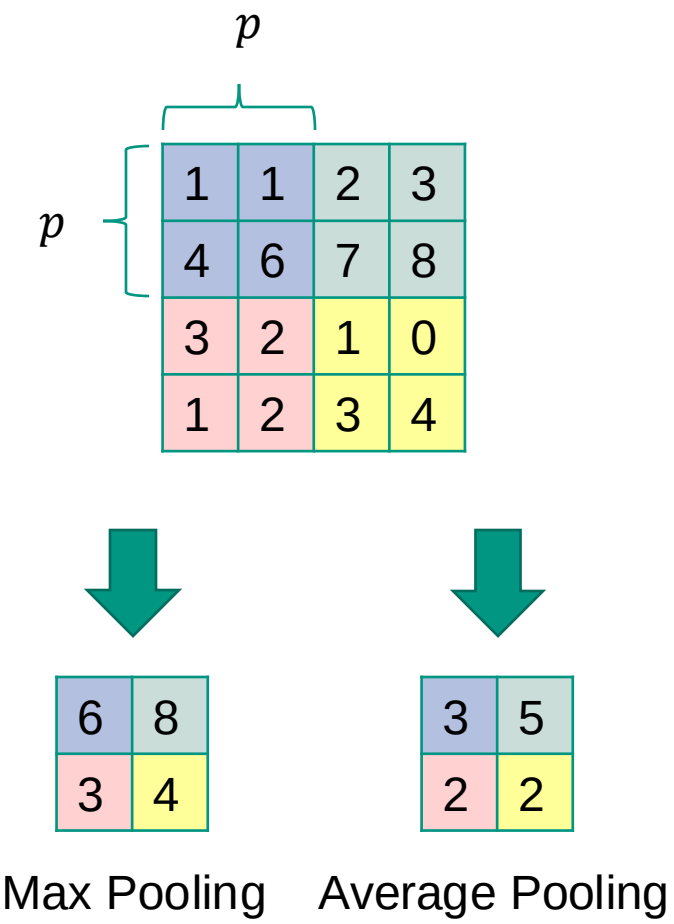
Reflect Padding

7	2	3	5	3	6	7	7	2
7	5	1	8	8	3	0	7	5
0	7	7	2	9	9	1	0	7
1	3	5	2	2	8	0	1	3
1	6	2	6	7	1	1	1	6
0	3	8	5	5	6	1	0	3
2	0	3	8	1	5	4	2	0
7	2	3	5	3	6	7	7	2
7	5	1	8	8	3	0	7	5

Circular Padding

Pooling Layer

- Aggregation of small image areas in feature maps
  - $p \times p$ area A
  - Usually $p = 2$
  - Usually stride $s = p$
- Different Strategies
  - Max Pooling $\max\{a \in A\}$
  - Average Pooling $\frac{1}{|A|} \sum_{a \in A} a$
- Advantages
  - Local translation invariance
  - Data reduction
  - No learnable parameters



Reduce Resolution: Strided Conv. vs Pooling

- Strided conv. filters can reduce resolution
- Conv. filters contains learnable parameters
  - → slower training and inference
  - Nowadays
    - First reductions usually uses conv. layer with stride = 2 and a large kernel
      - Pooling loses too much information
    - Pooling operations sometimes used for later resolution reductions but in the last years only rarely seen.

$\frac{1}{4}$	$\frac{1}{4}$
$\frac{1}{4}$	$\frac{1}{4}$

Conv. kernel with stride = 2
→ average pooling

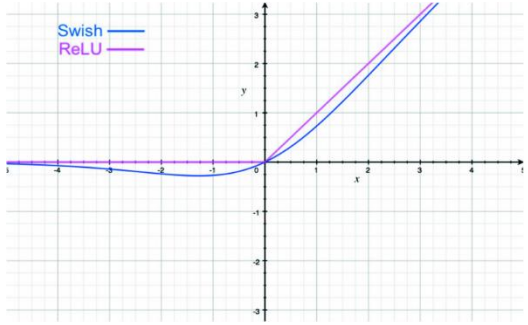
- "*The pooling operation used in convolutional neural networks is a big mistake and the fact that it works so well is a disaster*"
(Geoffry Hinton, Turing Award & Nobel Price winner, developer of backpropagation)

Activation Functions

- Create nonlinearity between convolutional layers
- ReLU is used most often
  - $f(x) = \max(0, x)$
  - Very fast to calculate
- More recently Swish/SiLU
  - $f(x) = x \cdot \sigma(x)$
  - Or in its gated version: SwiGLU

-1	1	2	1
1	1	1	1
1	1	**-3**	0
-1	0	0	0

Output of previous conv. layer



ReLU

0	1	2	1
1	1	1	1
1	1	**0**	0
0	0	0	0

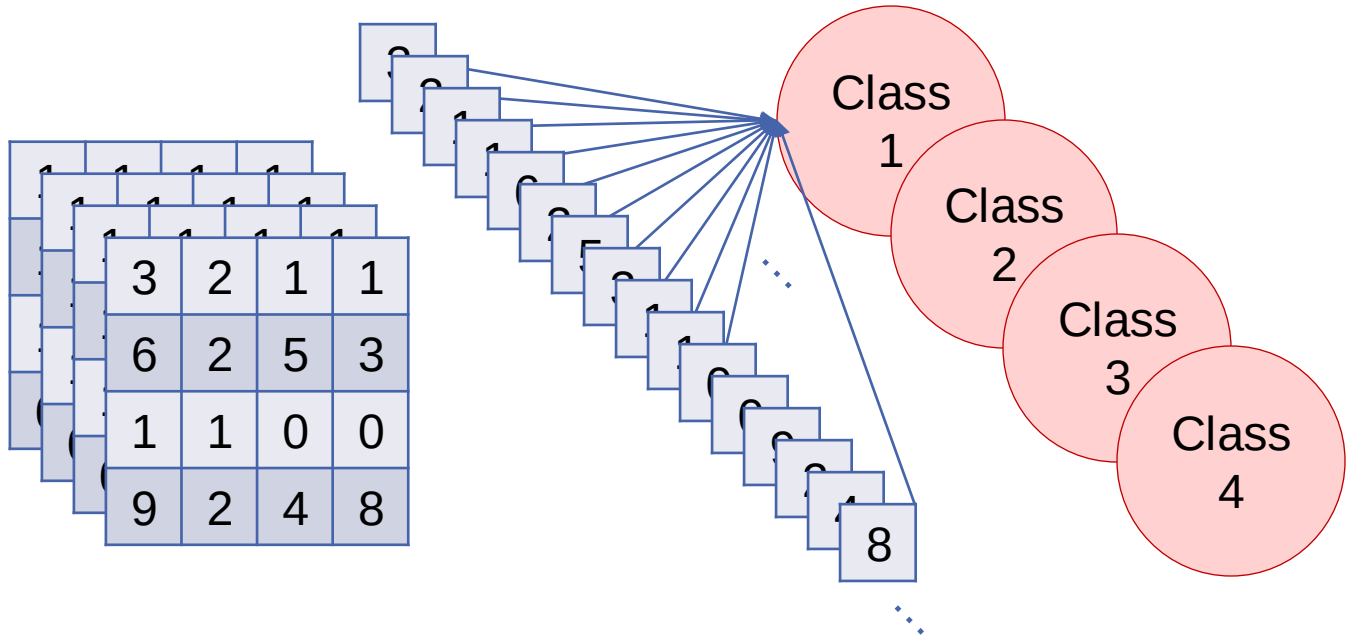
Input next conv. layer

$$\mathrm{ReGLU}(x, W, V, b, c) = \max(0, xW + b) \otimes (xV + c)$$
$$\mathrm{GEGLU}(x, W, V, b, c) = \mathrm{GELU}(xW + b) \otimes (xV + c)$$
$$\mathrm{SwiGLU}(x, W, V, b, c, \beta) = \mathrm{Swish}_\beta(xW + b) \otimes (xV + c)$$

as well as better results on many downstream language-understanding tasks. These architectures are simple to implement, and have no apparent computational drawbacks. We offer no explanation as to why these architectures seem to work; we attribute their success, as all else, to divine benevolence. [SwiGLU]

Classification

- Earlier CNNs had multiple fully connected layers
  - Lots of weights

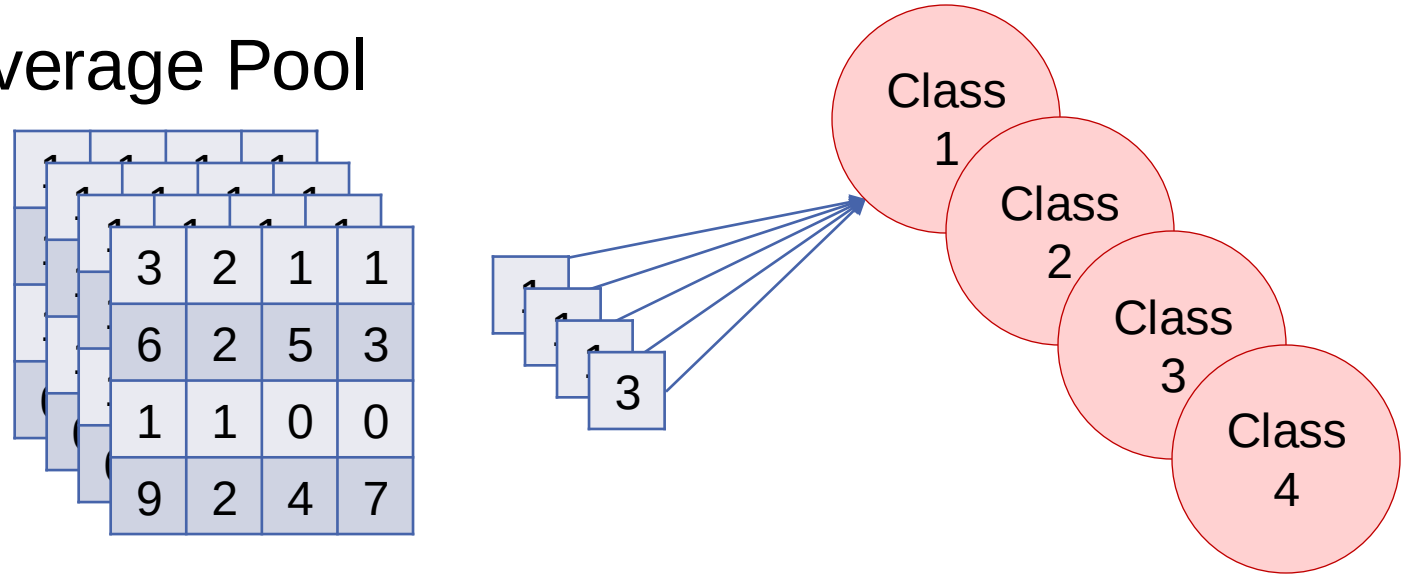


- Recently only one FC layer
  - Number classes = number neurons

Resolution Independent Classification

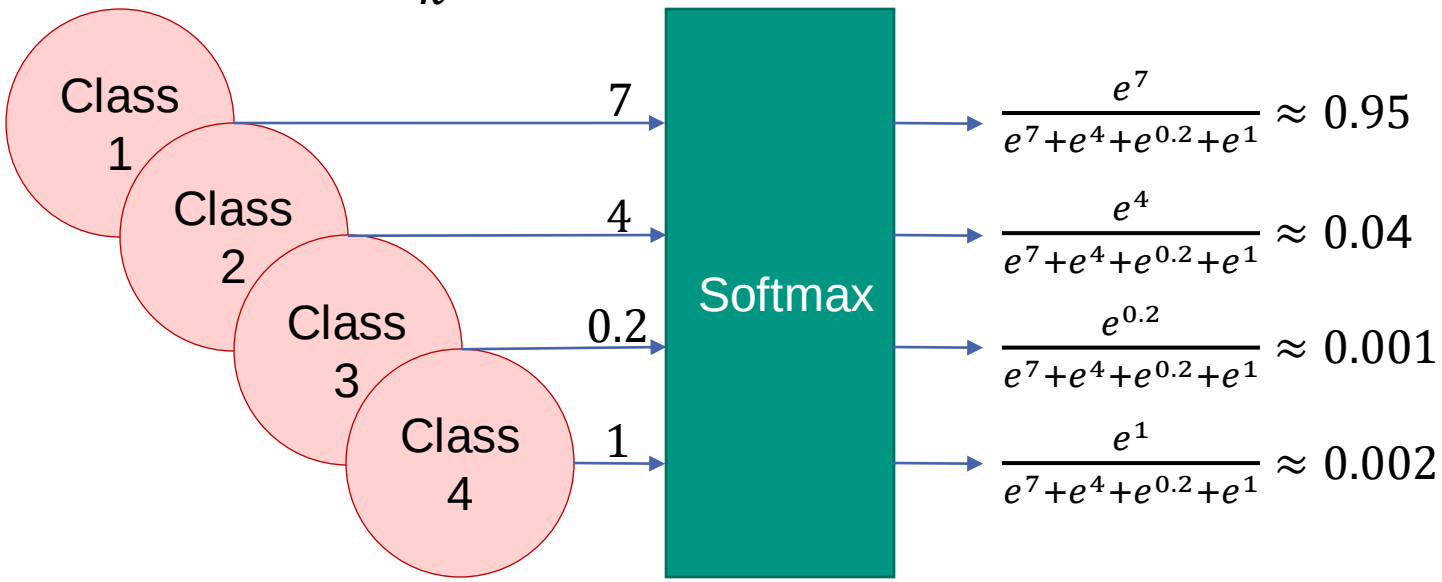
- Convolutional filter can be used for different input resolutions
  - Fully connected layers need fixed resolution
  - CNNs with fully connected layers can‘t be used for different resolutions

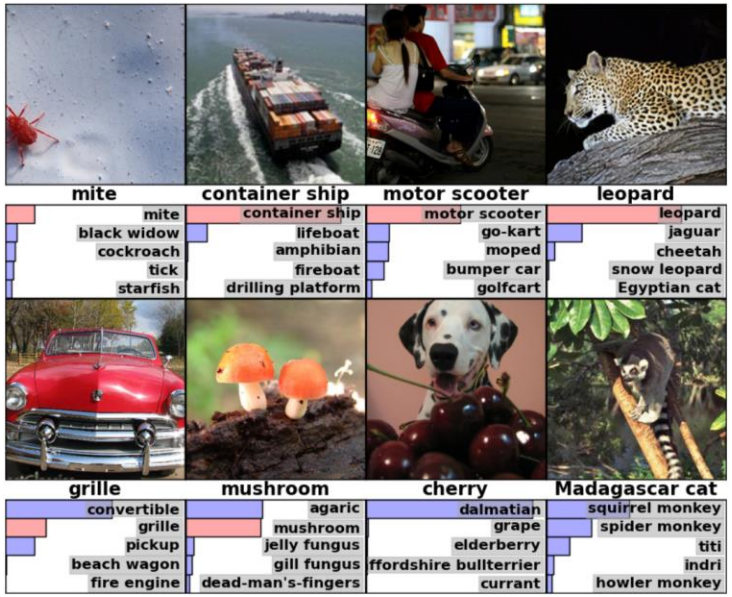
- → Channelwise Global Average Pool



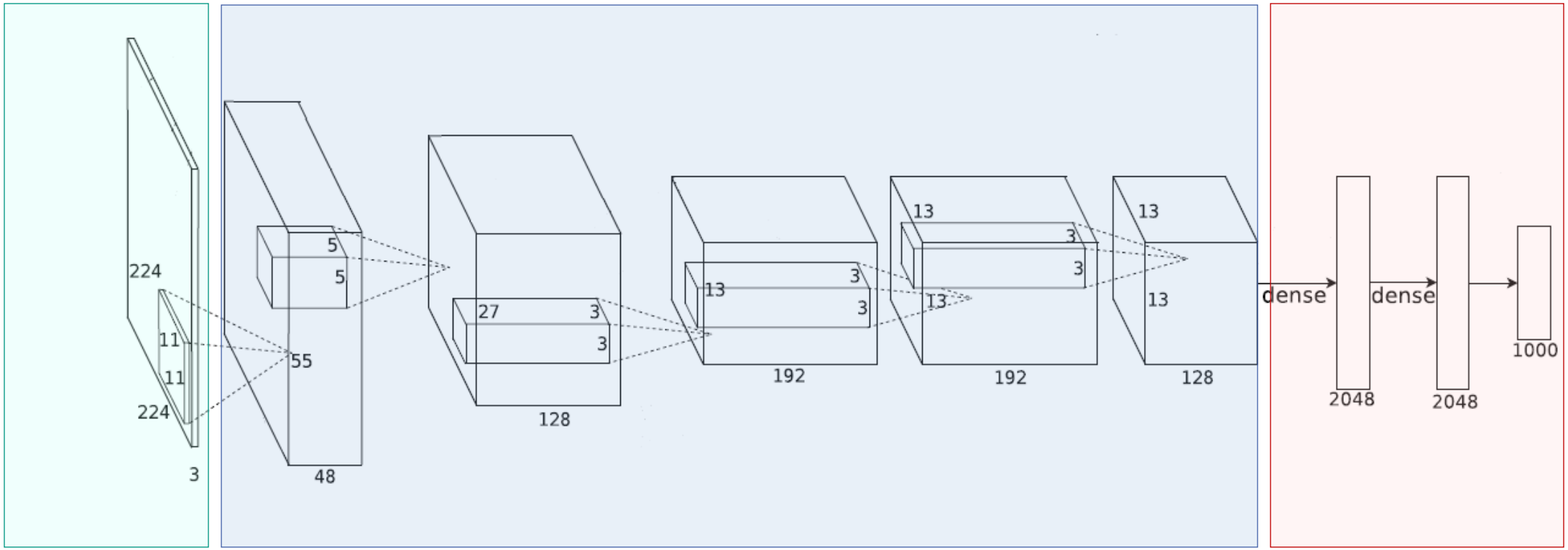
Classification: Softmax

- Last „activation“ function of fully connected layer
- Get a normalized probability distribution
- $softmax(z_i) = \frac{e^{z_i}}{\sum_k e^{z_k}}$





Where are we now



Input Image

Feature Extraction

Classification

Initialization of weights

- Weight initialization of kernel values can drastically impact the training

- Different approaches
  - Start training from scratch
    - Usually, Kaiming He initialization

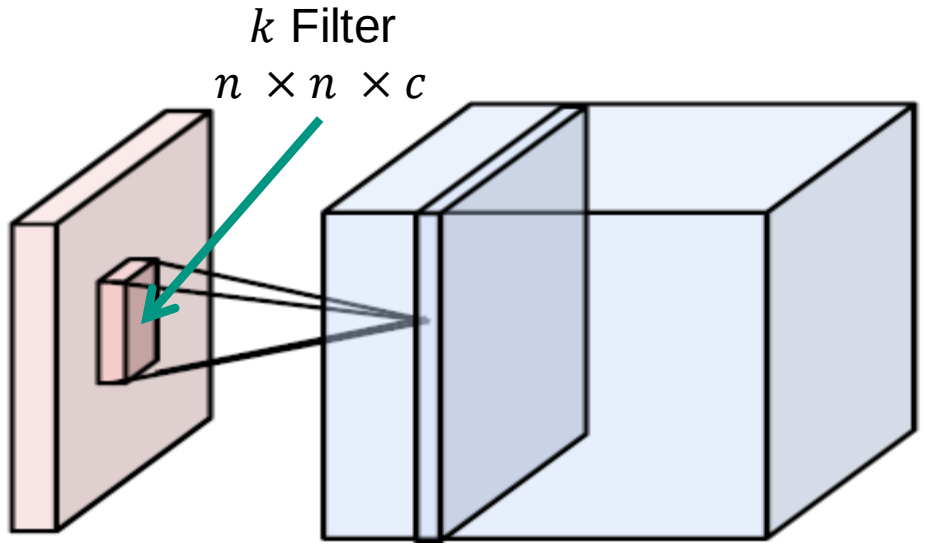
  - Use weights from an already trained CNN (**Transfer Learning**)
    - Only applicable using identical architectures
    - Two alternatives in application:
      - Only train classifier, keep weights of previous convolutional layers fixed
      - Train all weights, but use small learning rates

Overview

- Motivation
- Basics
  - Convolution
- Convolutional Neural Networks
  - Layers and Parameters
  - Learning of Features
  - Important Architectures
- Outlook and Extensions

Features in CNNs

- Convolutional filters are feature detectors
  - Outputs are called feature maps



- **Learn (weights of) convolutional filters with backpropagation and gradient descent!**

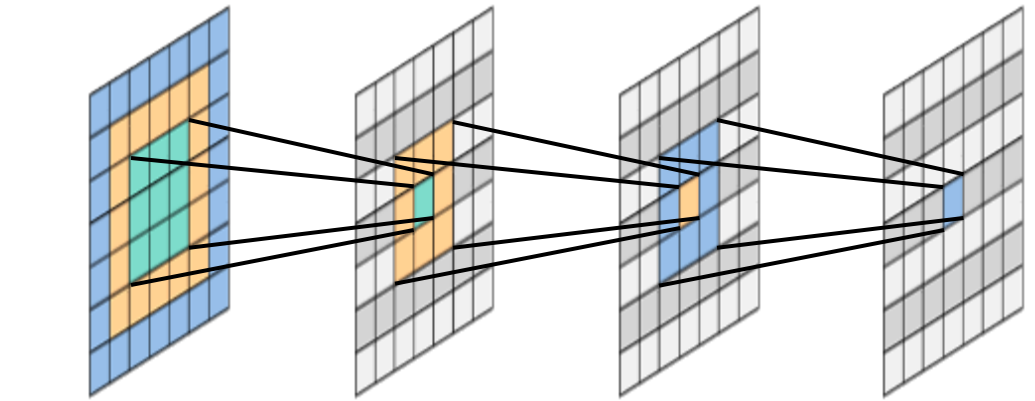
Receptive Field of Convolutions

感受野

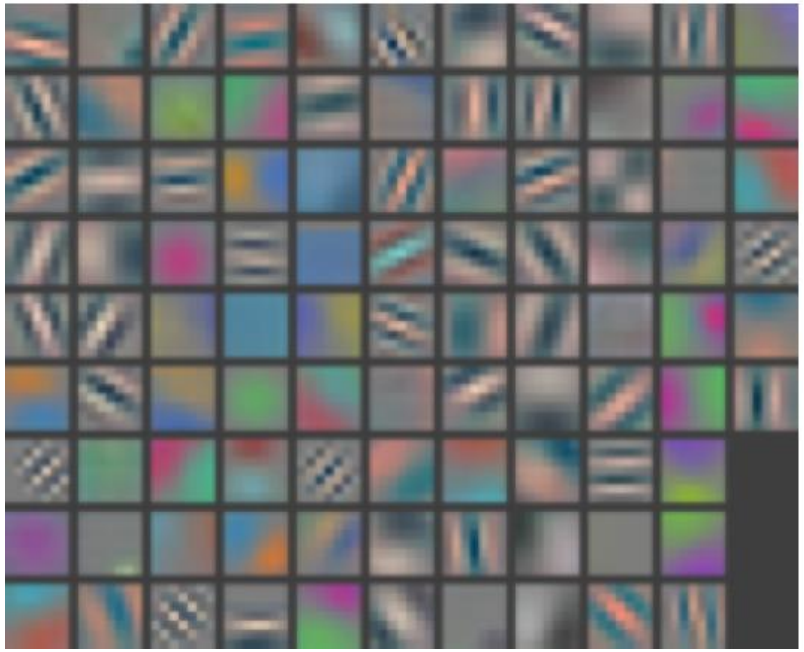
在原始输入中，对某个特征图像素产生影响的区域大小

- Definition: region size in the original input that produced a feature
  - How much information of the original image is stored in a feature map pixel?

- Consider two cases
  - One 7×7 conv. filter
    - 7×7 receptive field
    - 50 params
  - Three 3×3 conv. filter
    - 7×7 receptive field
    - 30 params
- 3×3 conv. filter de facto standard

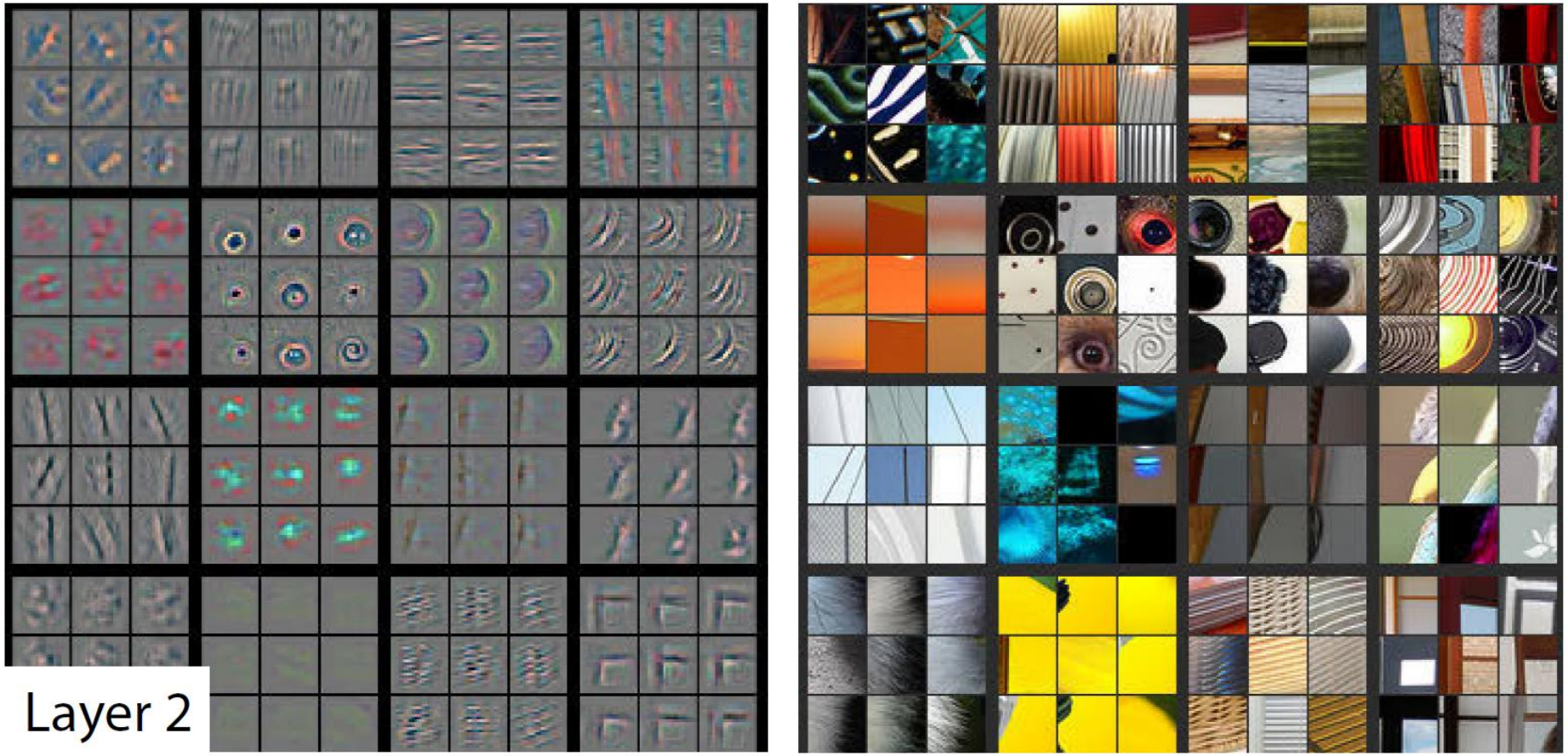


Filters in Stage 1



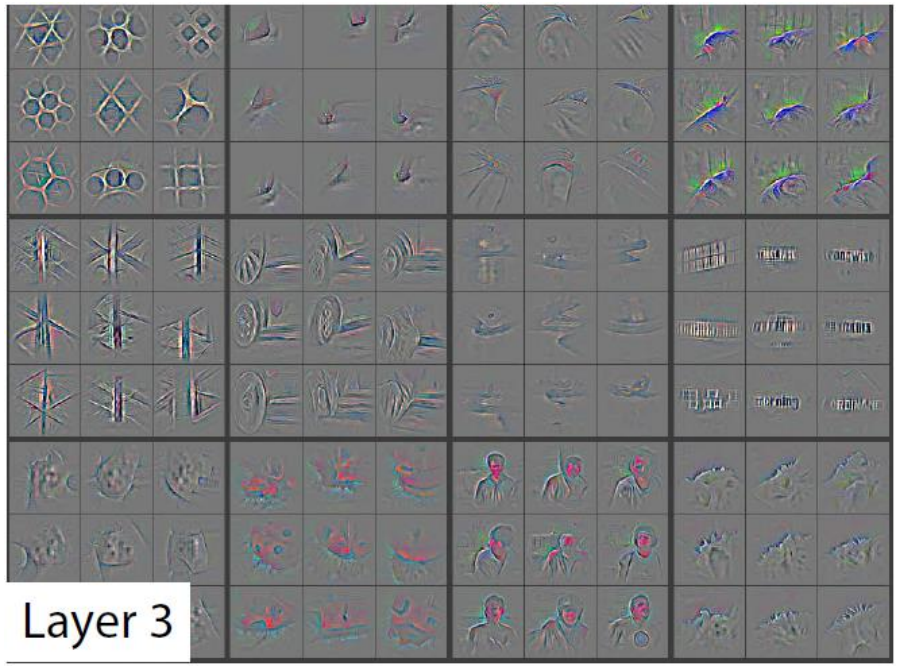
Zeiler et al. – *Visualizing and Understanding Convolutional Neural Networks*

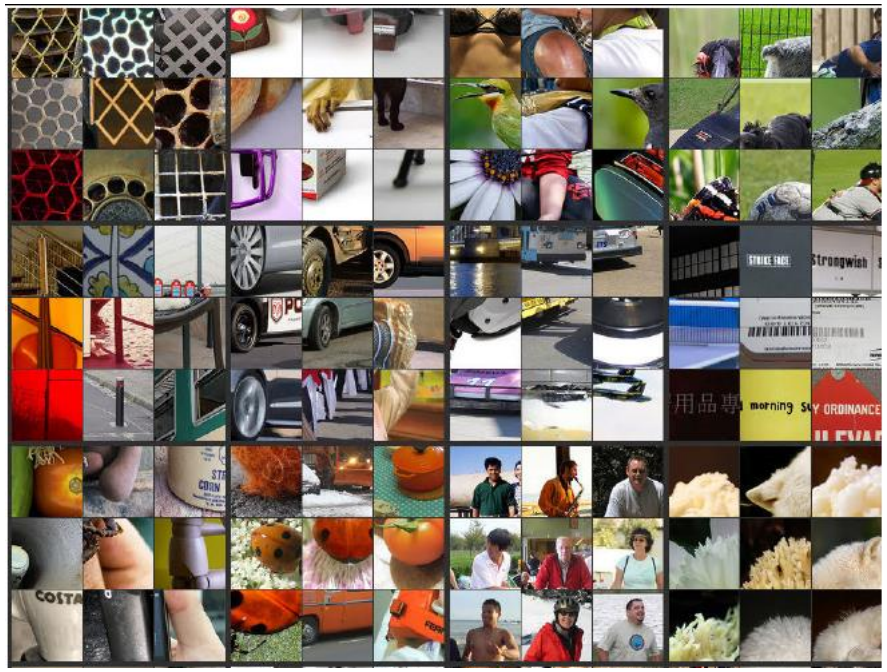
Filters in Stage 2



Zeiler et al. – *Visualizing and Understanding Convolutional Neural Networks*

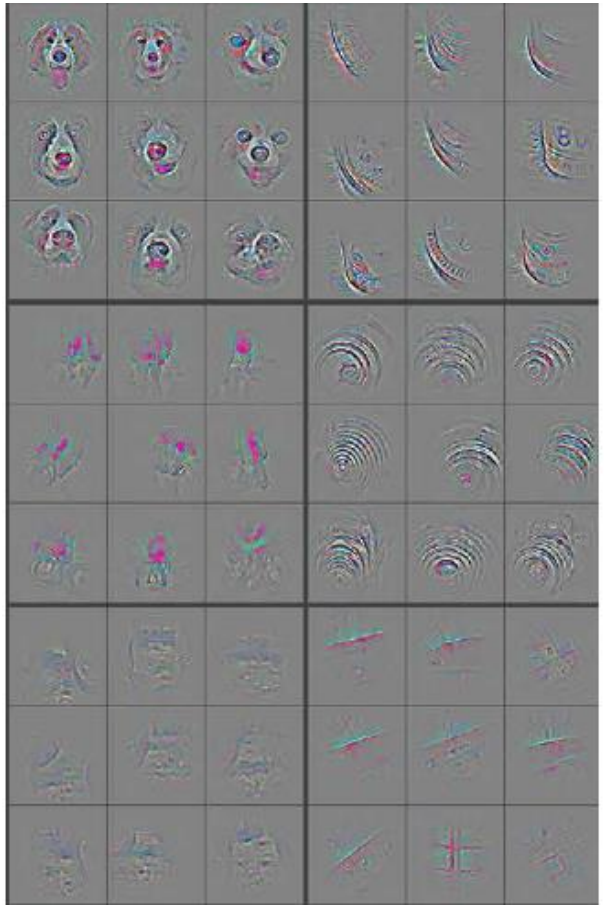
Filters in Stage 3





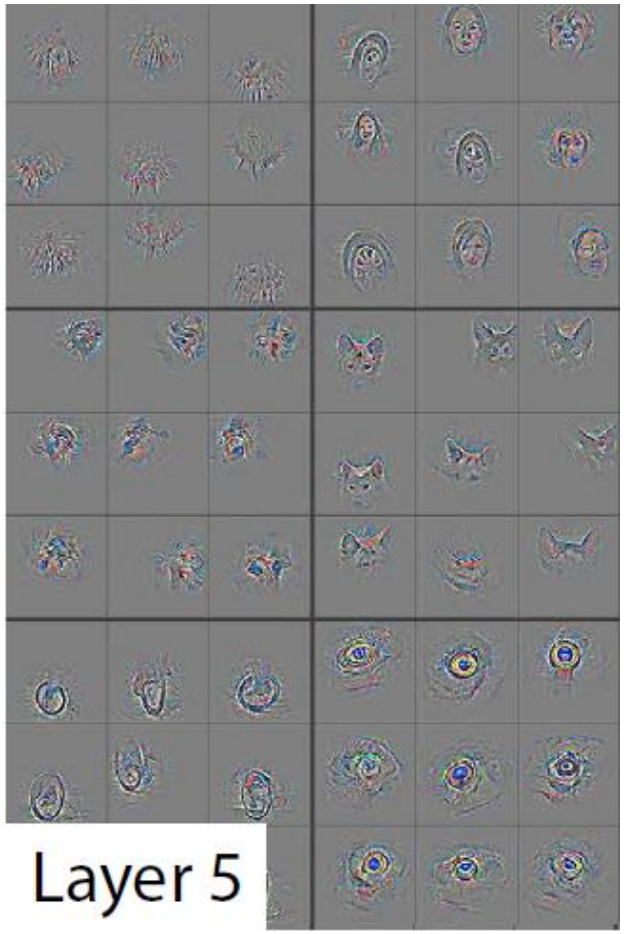
Zeiler et al. – *Visualizing and Understanding Convolutional Neural Networks*

Filters in Stage 4



Zeiler et al. – *Visualizing and Understanding Convolutional Neural Networks*

Filters in Stage 5



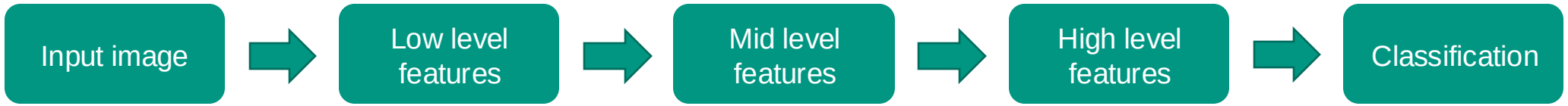
Zeiler et al. – *Visualizing and Understanding Convolutional Neural Networks*

Hierarchical Features

层次化特征

- Stacking convolutional filters increases the receptive field
  - Creates feature hierarchy
    - Features from low receptive fields are mostly edge detectors
    - Features from high receptive fields contain semantic meaning

- Features of one layer are based on the features of previous layer

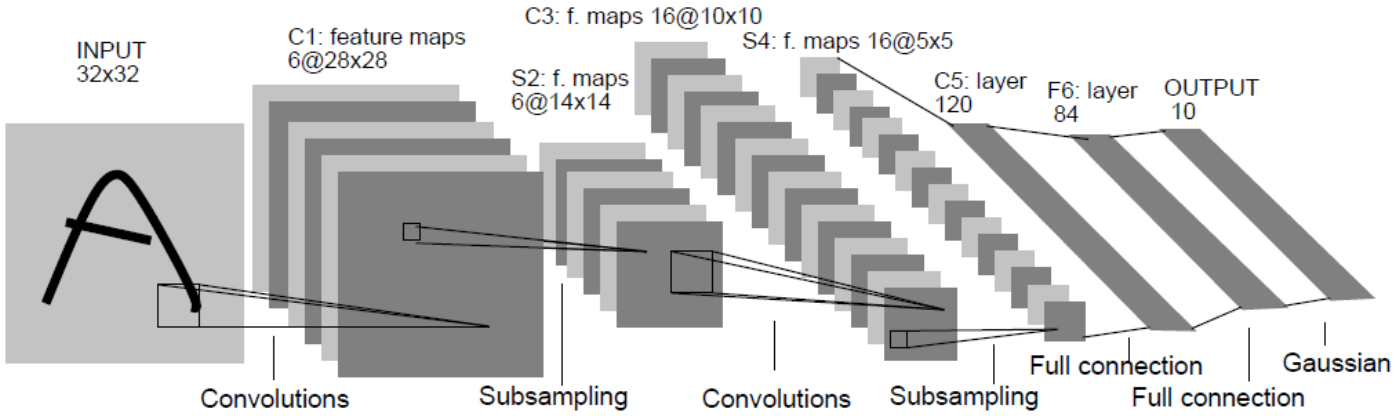


Training CNNs

- Nearly **identical to neural nets**
- **Weights and biases** of the convolutional filters are learned
  - **Shared Weights:** One weight of convolutional kernel is used multiple times for different inputs → Addition of gradients for different inputs
  - Also learn weights and biases of fully connected layers if they exist

- Image as input, e.g. classification as output
- Loss function takes output of CNN and correct label
- Calculate gradients of all weights with regards to the loss function with backpropagation
- Update parameters with Optimizer (e.g. Adam/SGD)

Architecture Analysis: LeNet (1989)



#	Type	Filters @ Patch size / stride	Parameters
	Input		0
1	Convolution	$6 @ 5 \times 5 \times 1 / 1$	156
2	Scaled average pooling	$2 \times 2 / 2$	2
3	Convolution	$16 @ 5 \times 5 \times 6 / 1$	2 416
4	Scaled average pooling	$2 \times 2 / 2$	2
5	Fully Connected	120 neurons	**48 120**
6	Fully Connected	84 neurons	10 164
7	Fully Connected (output)	10 neurons	850
$\sum$			61 710

[Martin Thoma]

LeCun – *Gradient-Based Learning Applied to Document Recognition*

Neural Networks / CNNs are from the eighties. What changed?

- **Huge** datasets

Dataset	Number Classes	Number Images
ImageNet-1k	1000	1.28 mln (~150GB)
ImageNet-22k	21800	14.8 mln (~1.3TB)
JFT-300m (internal Google dataset)	18.000	300 mln
JFT-3B (internal Google dataset)	30.000	3 bln
Laion-5B	Natural language descriptions	5 bln (220 TB)

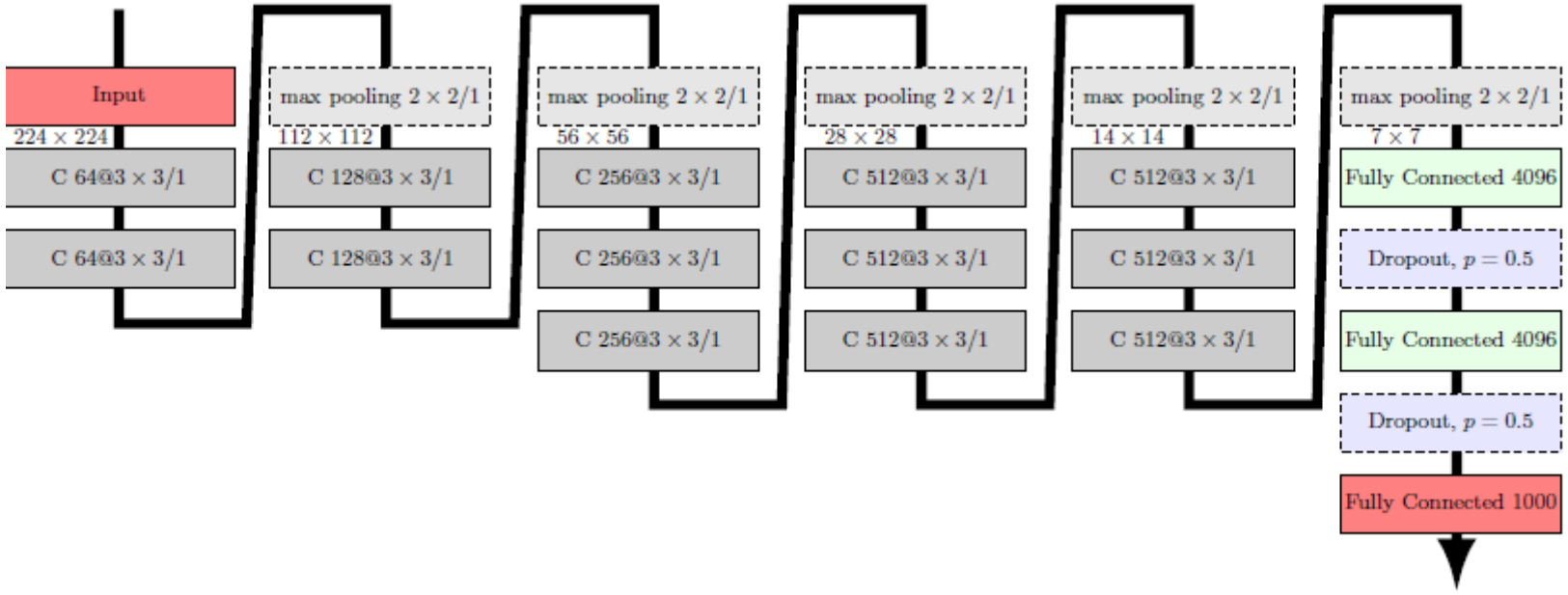
- Parallelization on **GPUs**
  - GPUs are optimized for computer graphics which are mostly matrix operations
  - All mathematical functions of CNNs/NNs can be represented as matrix operations

Zuckerberg's Meta Is Spending Billions to Buy 350,000 Nvidia H100 GPUs

In total, Meta will have the compute power equivalent to 600,000 Nvidia H100 GPUs to help it develop next-generation AI, says CEO Mark Zuckerberg.

By Michael Kan January 18, 2024

Architecture Analysis: VGG 16 (2014)



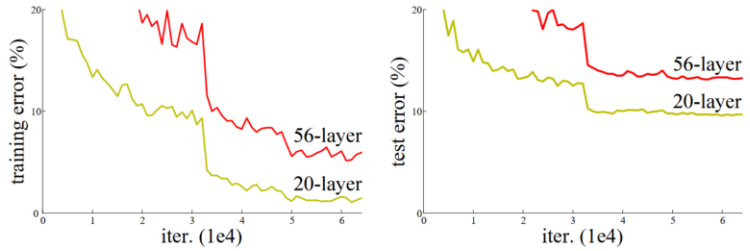
#	Type	Filters @ Patch size / stride	Parameters
	Input		
1	Convolution	64 @ 3 × 3 × 3 / 1	1 792
2	Convolution	64 @ 3 × 3 × 64 / 1	36 928
	Max pooling	2 × 2 / 2	0
3	Convolution	128 @ 3 × 3 × 64 / 1	73 856
4	Convolution	128 @ 3 × 3 × 128 / 1	147 584
	Max pooling	2 × 2 / 2	0
5	Convolution	256 @ 3 × 3 × 128 / 1	295 168
6	Convolution	256 @ 3 × 3 × 256 / 1	590 080
7	Convolution	256 @ 3 × 3 × 256 / 1	590 080
	Max pooling	2 × 2 / 2	0
8	Convolution	512 @ 3 × 3 × 256 / 1	1 180 160
9	Convolution	512 @ 3 × 3 × 512 / 1	2 359 808
10	Convolution	512 @ 3 × 3 × 512 / 1	2 359 808
	Max pooling	2 × 2 / 2	0
11	Convolution	512 @ 3 × 3 × 512 / 1	2 359 808
12	Convolution	512 @ 3 × 3 × 512 / 1	2 359 808
13	Convolution	512 @ 3 × 3 × 512 / 1	2 359 808
	Max pooling	2 × 2 / 2	0
14	FC	4096 neurons	**102 764 544**
	Dropout		0
15	FC	4096 neurons	16 781 312
	Dropout		0
16	FC	1000 neurons	4 097 000
∑			138 357 544

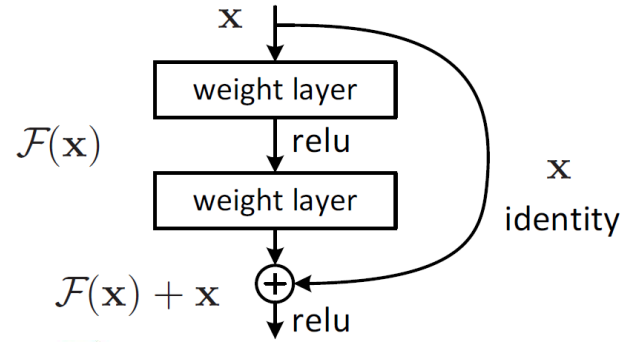
[Martin Thoma]

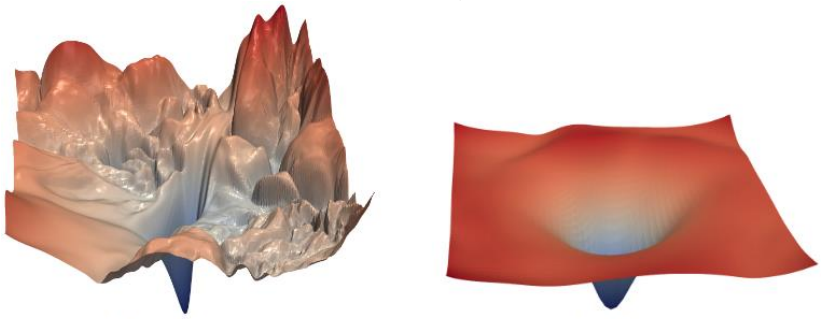
Simonyan – *Very Deep Convolutional Networks For Image Recognition*

Architecture Analysis: ResNet (2015)

- Worse performance with more layers
- Skip Connection
  - Identity parallel to each residual block
  - Add identity to output of block
  - Learn the change of existing feature maps
- Skip connections fix performance problem
  - Probably improvement of the loss landscape





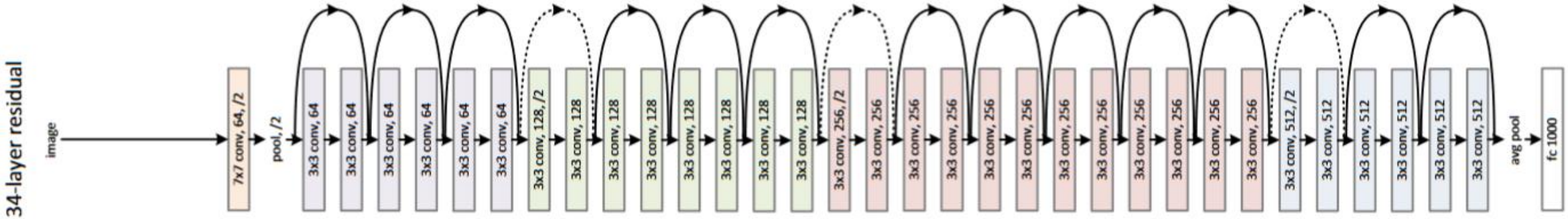


(a) without skip connections (b) with skip connections

Architecture Analysis: ResNet (2015)

- Skip Connections allow training of CNNs with lots of layers
- Rule of thumb: Deeper CNNs → Better performance
  - ResNet with up to 200 conv. Layers

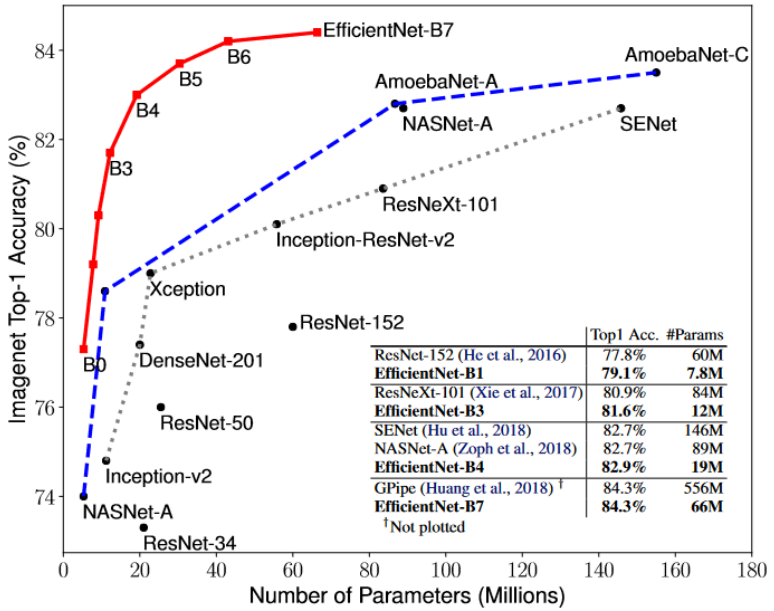
- Skip Connections nowadays in all neural network architecture (LLMs, Diffusion Models …)



He – *Deep Residual Learning for Image Recognition*

Architecture Analysis: EfficientNet (2019)

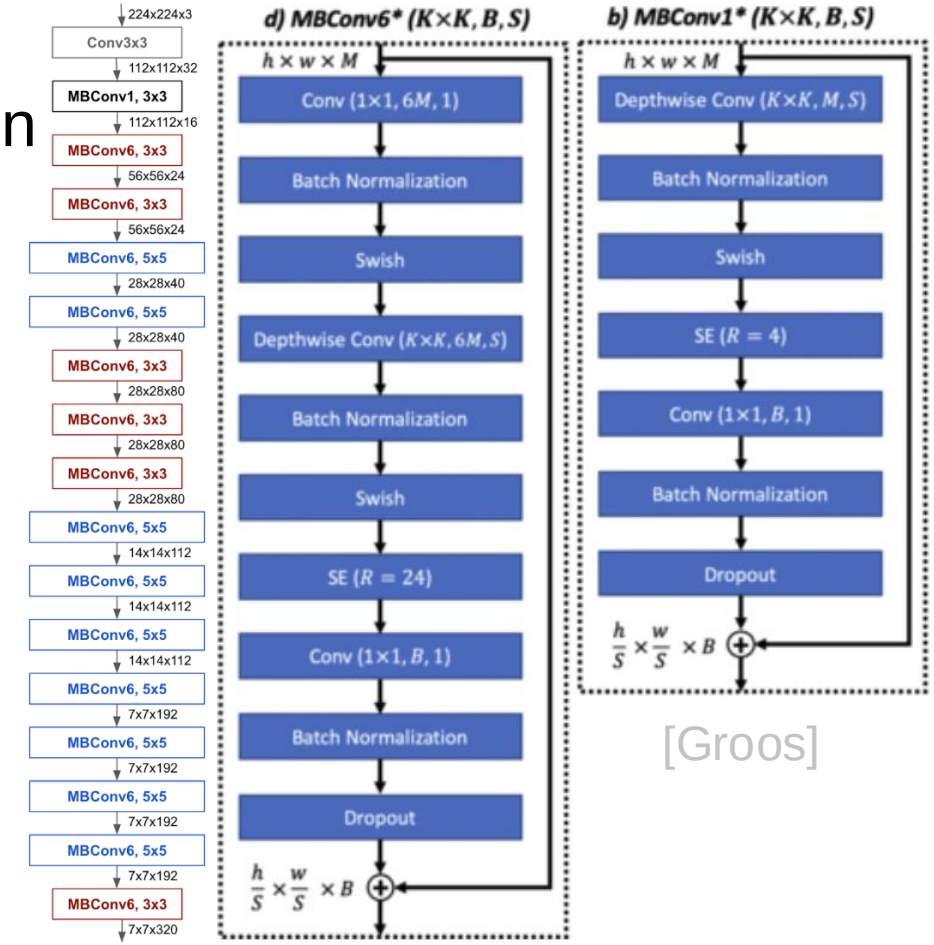
- How to scale CNNs
  - Increase either depth, width or input resolution
  - Search for compound coefficient



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**79.1%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.6%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.9%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.3%**	**66M**

†Not plotted

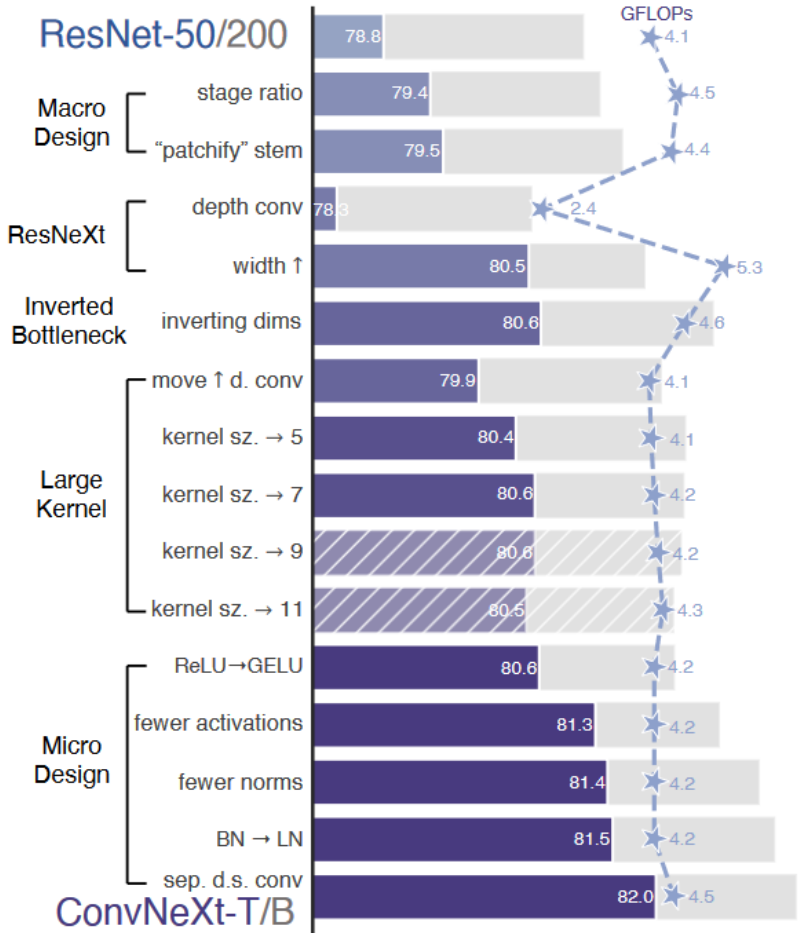
Tan – *EfficientNet: Rethinking Model Scaling for CNNs*



[Groos]

Architecture Analysis: ConvNext (2022)

- Take ResNet and modernize it
  - Idea: CNN resembles modern transformer architectures
    - „Patchify“: Large strides in conv. Filters
    - „Large Kernels“: changes the common 3x3 Kernels to 7x7 kernels
    - Relu -> GELU
    - Fewer activation functions and normalization layers



Overview

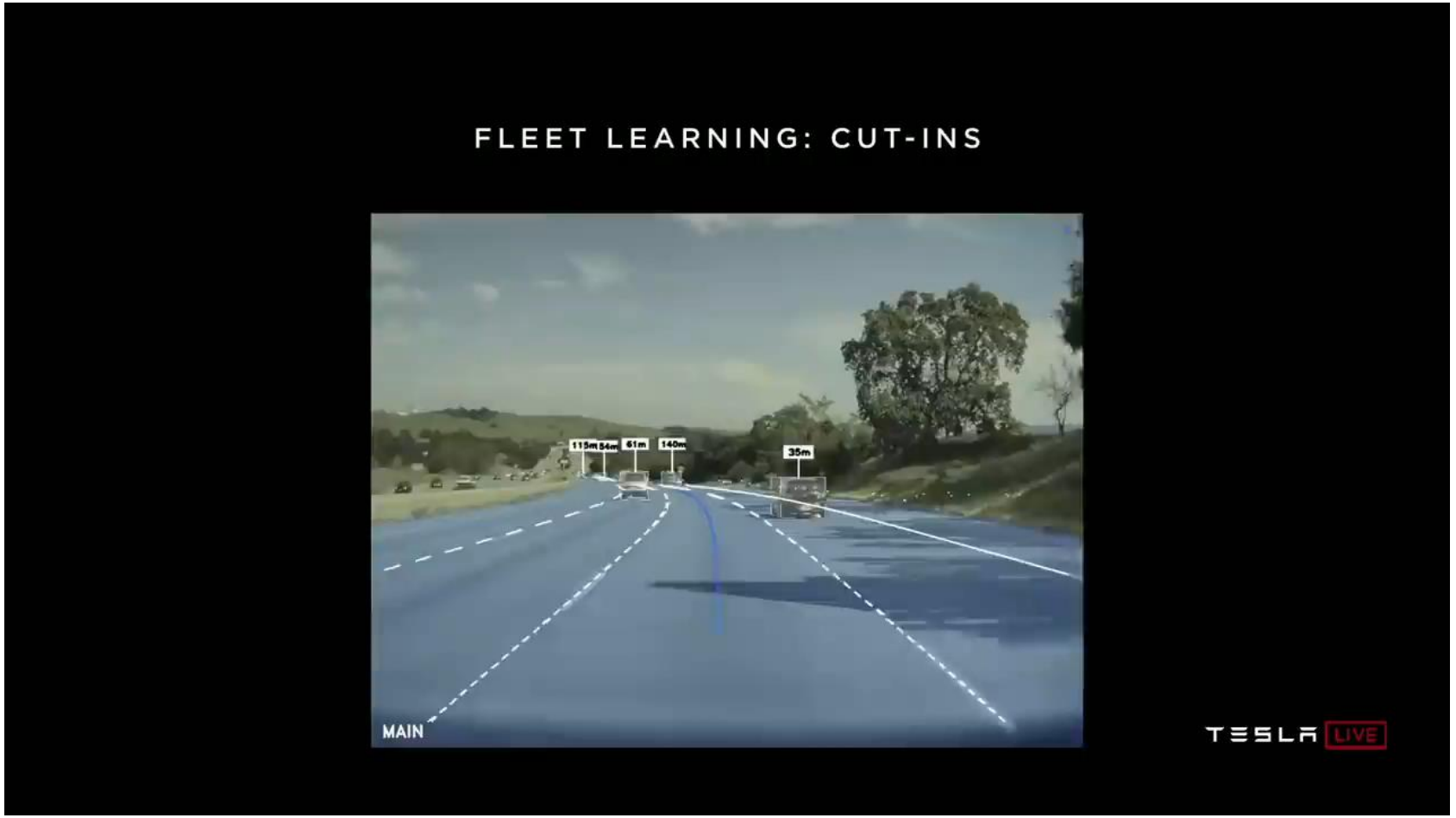
- Motivation
- Basics
  - Convolution
- Convolutional Neural Networks
  - Layers and Parameters
  - Learning of Features
  - Important Architectures
- Outlook and Extensions

ImageNet Benchmark (12.2024)

- ImageNet more or less solved
  - More than one class in image
  - Wrong labels
  - Pure image classification task no longer that interesting…
- Since 2020, large transformer architectures achieve better results than CNNs
  - But for most real-time applications, CNNs still reign supreme. Their accuracy-latency tradeoff is oftentimes better than transformer architectures.

Rank	Model	Top 1 Accuracy	Number of params	GFLOPs	Extra Training Data	Paper	Code	Result	Year	Tags
1	**CoCa** (finetuned)	91.0%	2100M		×	CoCa: Contrastive Captioners are Image-Text Foundation Models			2022	ALIGN, Transformer, JFT-3B
2	**Model soups** (BASIC-L)	90.98%	2440M		×	Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time			2022	ALIGN, JFT-3B, Conv+Transformer
3	**Model soups** (ViT-G/14)	90.94%	1843M		×	Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time			2022	JFT-3B, Transformer
4	**ViT-e**	90.9%	3900M		×	PaLI: A Jointly-Scaled Multilingual Language-Image Model			2022	Transformer, JFT-3B
5	**CoAtNet-7**	90.88%	2440M	2586	×	CoAtNet: Marrying Convolution and Attention for All			2021	Conv+Transformer, JFT-3B

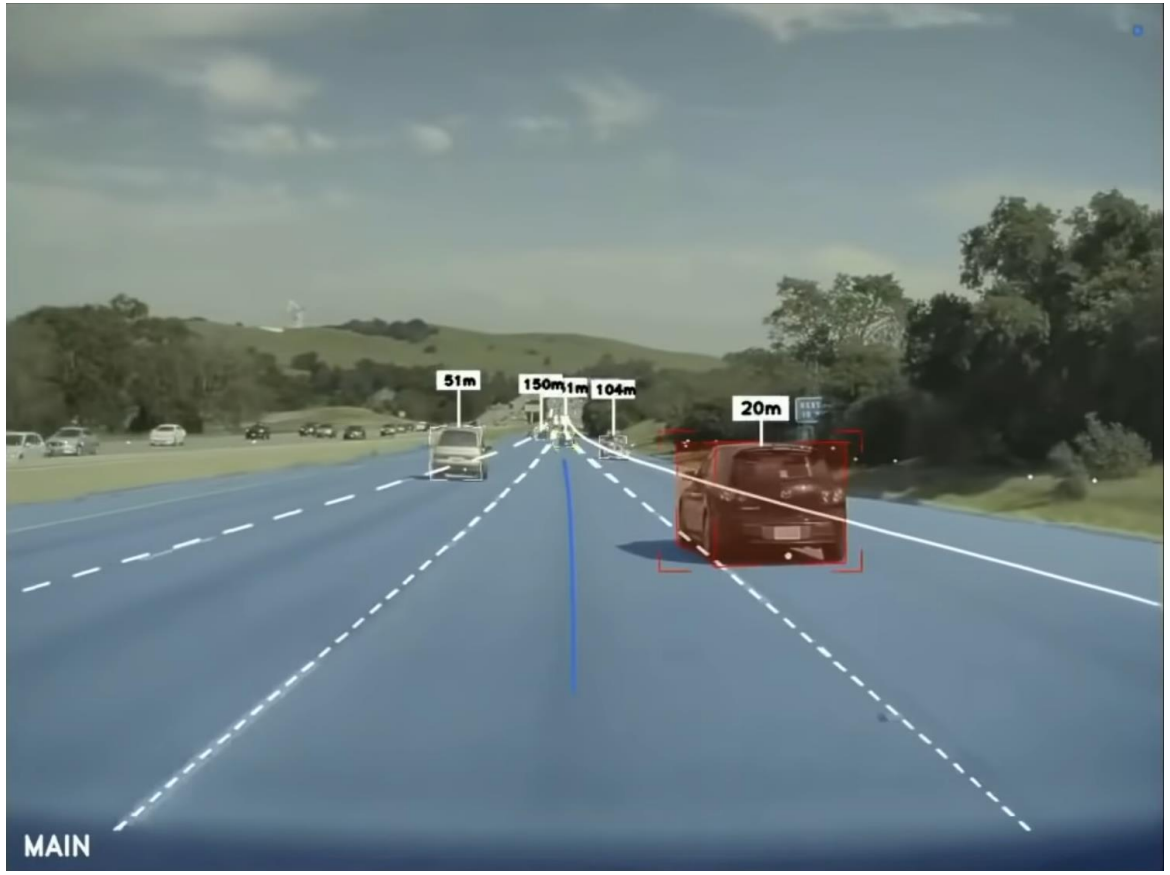
Application of CNNs in Automotive Driving



Tesla, Inc.
https://youtu.be/eZOHA6Uy52k

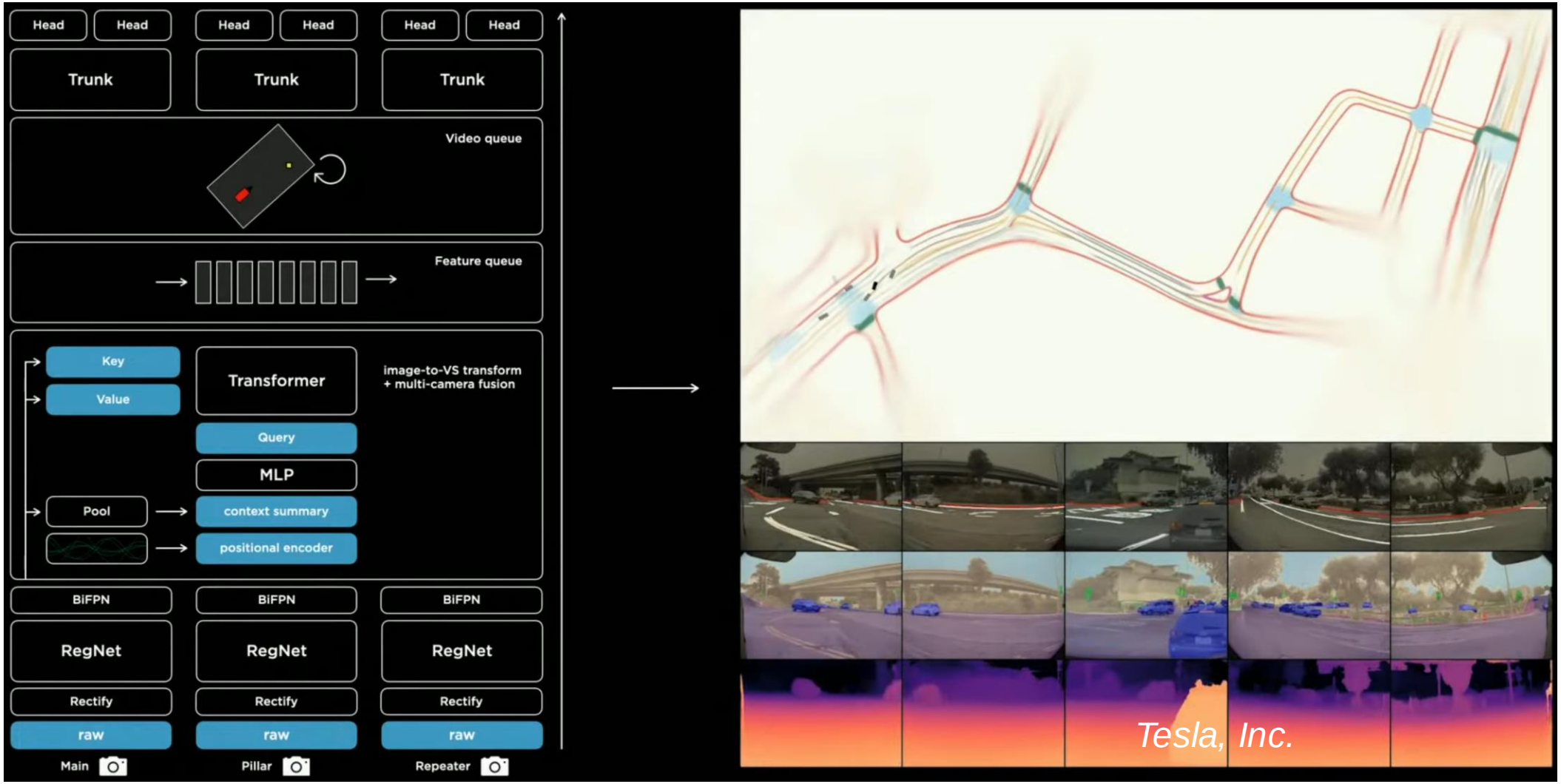
Application of CNNs in Automotive Driving

- Tesla Self Driving
  - Drivable area
  - Lane detection
  - 3D object detection
  - Cut-In detection
  - Path prediction
  - Depth from vision
  - Speed from vision
- Huge dataset of Tesla fleet
  - 1000 person labeling team

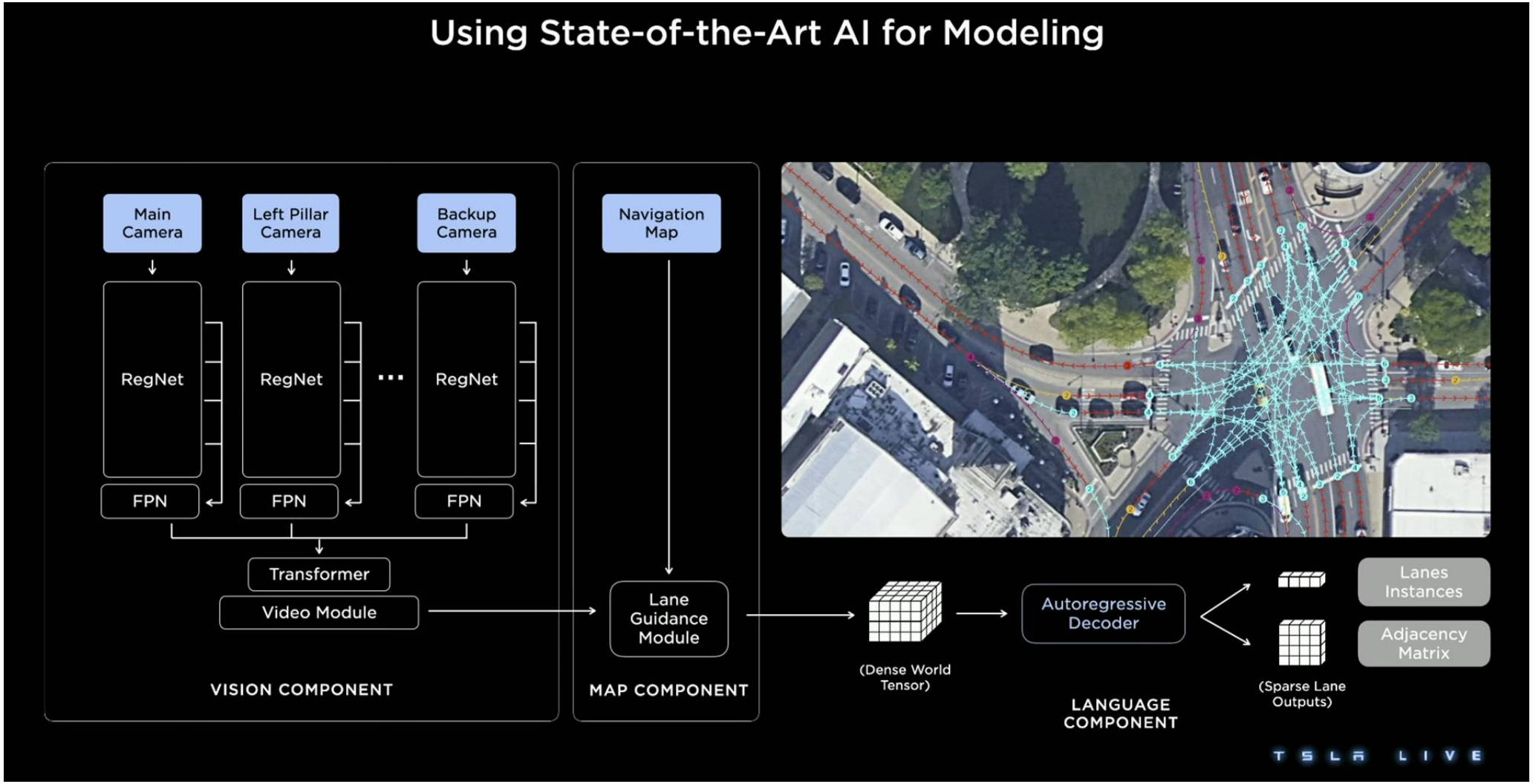


Tesla, Inc.

Tesla Architecture (08.2021)

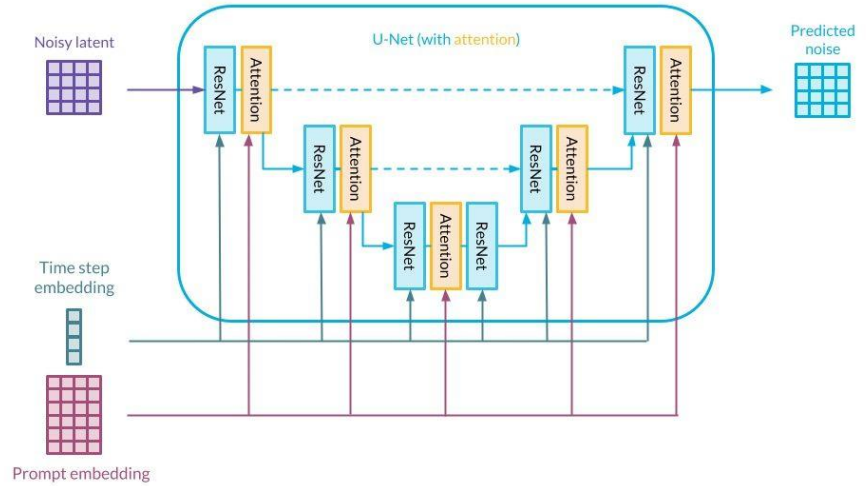


Tesla Architecture (03.2023)



Preview Neural Nets in ML2

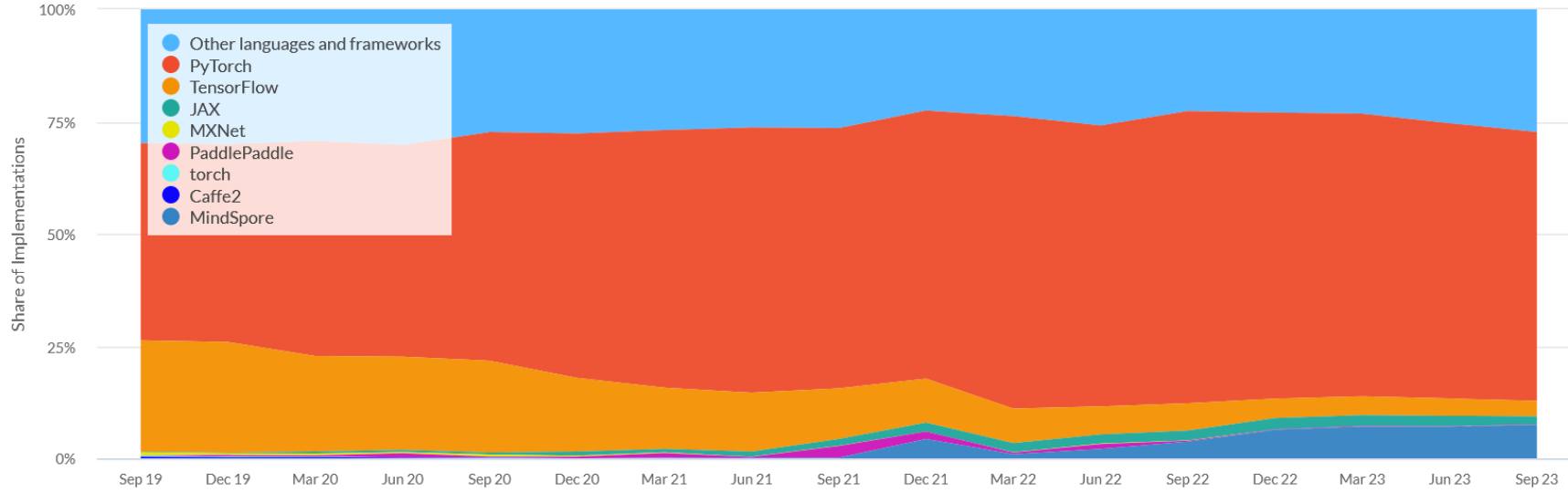
- Object Detection
- Semantic Segmentation
- Diffusion Models
- GANs
- NLP- and Vision Transformer



Prepare for exercise

- For the CNN exercise we will use PyTorch
- https://pytorch.org/get-started/locally/

- If your PC contains an Nvidia GPU → Install with CUDA otherwise CPU
- If you have Linux and an AMD GPU you can try to install ROCm but expect bugs..



Literature

- [He 2015] – Deep Residual Learning for Image Recognition
- [Goodfellow 2016] – Deep Learning (www.deeplearningbook.org)
- [Balduzzi 2017] – The Shattered Gradients Problem: If ResNets are the answer, then what is the question?
- [LeCun 89] – Gradient-Based Learning Applied to Document Recognition
- [Simonyan 2014] – Very Deep Convolutional Networks for Large-Scale Image Recognition
- [Li 2018] – Visualizing the Loss Landscape of Neural Nets
- [Hu 2017] – Squeeze-and-Excitation Networks
- [Tan 2019] – EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks
- [Gang 2021] – Character Recognition of Components Mounted on Printed Circuit Board Using Deep Learning
- [Zeiler 2013] – Visualizing and Understanding Convolutional Neural Networks

Deep Learning Frameworks

- PyTorch: https://pytorch.org/
- TensorFlow: https://www.tensorflow.org/