

CAS Machine Learning

Deep Learning: From Core Foundations to Applications

Javier Montoya, Dr. sc. ETH
javier.montoya@hslu.ch

Core Foundation

Artificial Intelligence

- History
- Perceptron
- Neurons and Layers
- Activation Functions

Neural Networks

- Architectures
- Optimization:
Forward/Backprop.
- Loss Functions

Training in Practice

- Regularization
- Train/Validation/Test

Deep Learning meets Computer Vision

Foundations

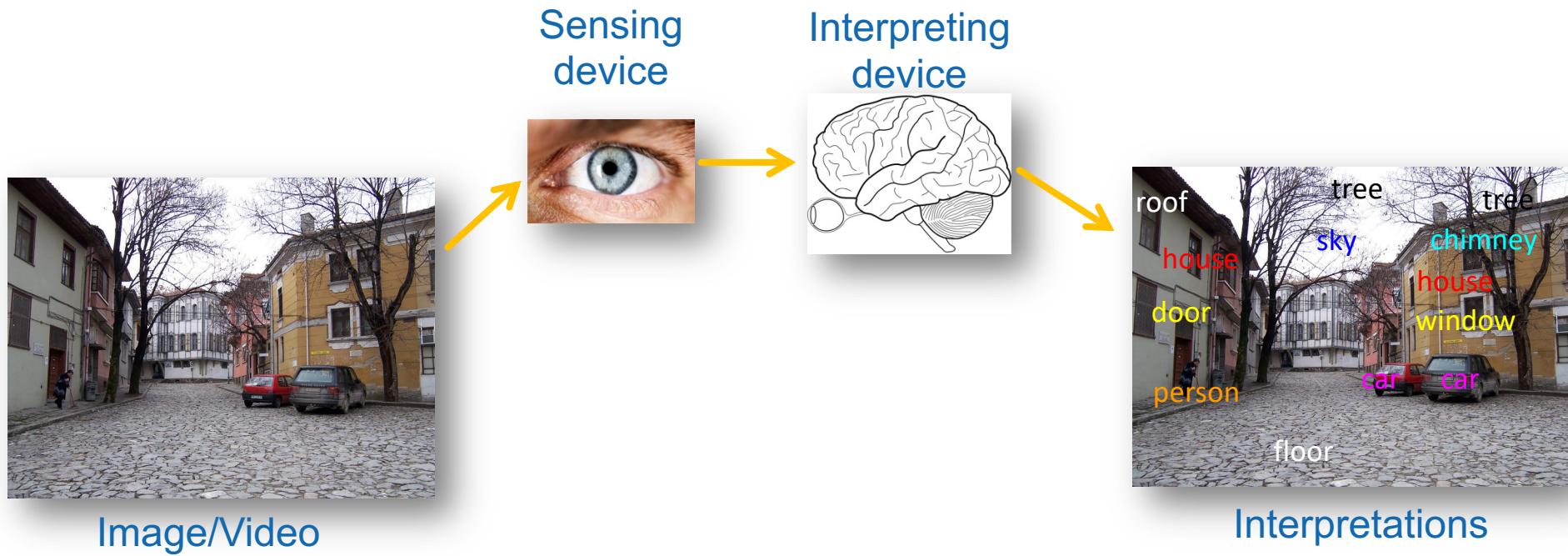
- What's Computer Vision?
- Challenges

Convolutional Neural Networks

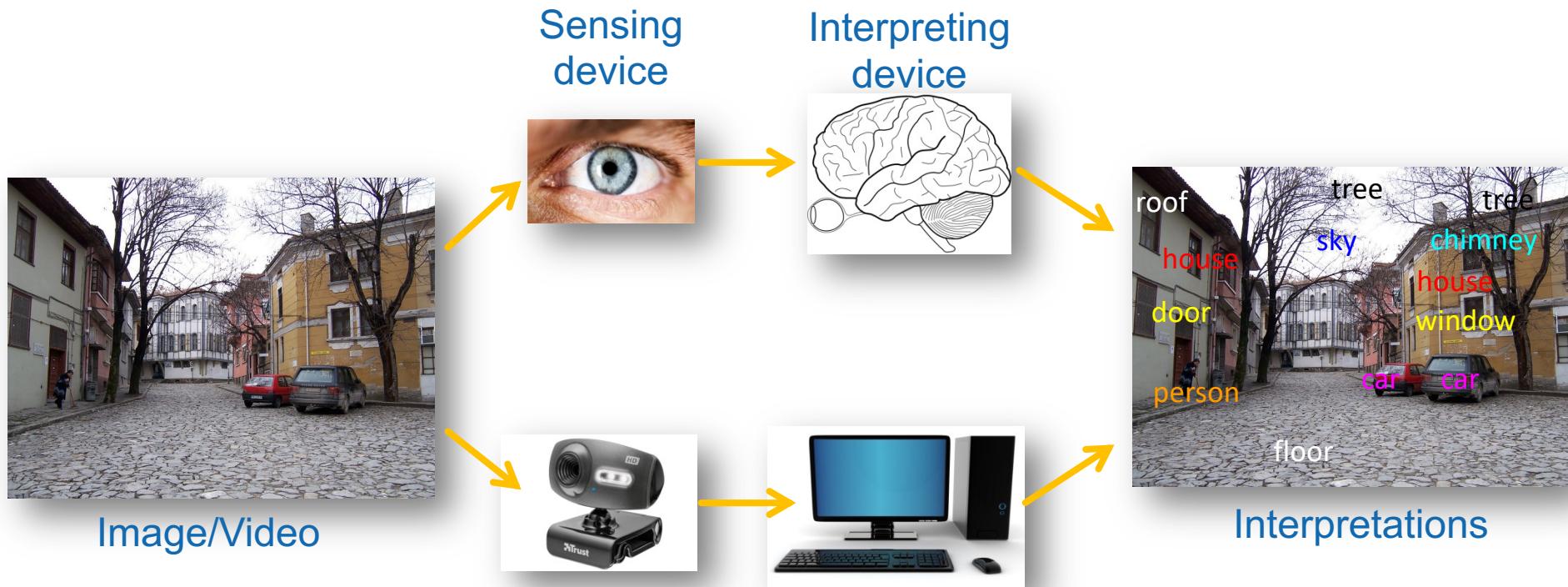
- Architectures
- Layers
- Loss Functions
- Common Tasks

Deep Learning meets Computer Vision

Computer Vision



Computer Vision



Digital Representation of Images



What humans see

Digital Representation of Images



What humans see

62 62 63 64 65 66 67 67 69 70 71 72 72 73 73 73 73 72 72 71 70 69 67 66 66 66 65 63 62 61 60 6
61 62 63 64 66 66 67 68 68 69 70 71 71 72 72 73 72 72 71 71 70 69 68 66 66 65 65 63 62 61 60 6
61 62 63 64 66 66 68 68 69 70 70 71 72 73 73 72 72 71 71 69 68 67 66 66 65 65 64 63 62 61 6
61 63 64 64 66 67 68 68 69 70 71 71 72 55 53 69 72 72 71 71 70 69 68 67 66 65 64 63 62 61 61 6
61 63 64 65 67 68 68 69 69 70 70 71 72 42 4 5 11 48 72 71 71 69 68 67 66 65 64 62 62 60 59 5
63 64 65 66 67 68 68 69 70 71 71 71 72 18 4 4 7 8 66 71 70 69 68 68 67 66 65 64 63 61 59 59 5
63 65 67 67 68 68 69 69 70 71 71 72 64 4 27 24 54 33 20 52 64 68 68 67 66 65 64 63 62 61 59 58 5
64 65 66 66 68 68 69 70 71 41 24 24 12 17 24 48 60 37 43 36 52 66 68 67 66 65 64 63 61 60 59 58 5
65 66 67 67 68 68 69 71 49 6 6 5 6 34 36 12 47 34 17 29 54 43 63 67 66 65 64 63 62 60 59 58 5
64 65 66 66 68 69 39 6 6 5 5 7 16 19 4 47 44 27 24 40 67 66 66 65 65 64 63 61 60 59 58 5
63 64 65 65 67 50 6 6 5 5 6 8 9 20 27 51 78 41 44 66 65 65 65 64 63 62 60 59 58 5
63 64 65 65 34 5 5 5 5 5 5 4 19 6 7 54 64 20 59 65 65 64 64 63 62 61 60 59 57 5
63 64 64 65 14 5 6 5 5 4 5 4 18 7 5 4 19 10 11 65 64 64 63 61 66 62 61 60 59 58 5
63 64 64 65 53 7 4 5 6 6 7 10 6 5 5 4 21 24 18 64 64 64 63 62 64 65 62 62 60 59 58 5
64 64 64 65 50 4 4 5 11 16 6 4 6 35 16 26 66 64 64 63 61 72 67 63 62 61 59 58 5
64 64 64 65 46 4 4 4 5 6 9 8 5 29 10 43 56 29 57 64 64 63 61 70 67 62 64 65 59 59 5
64 64 65 66 27 5 4 4 5 6 6 6 18 66 20 57 60 46 36 75 70 62 61 70 67 62 61 60 59 58 5
49 50 62 65 57 5 5 6 5 6 6 6 41 59 22 60 58 44 22 63 71 72 60 69 68 61 60 58 59 59 5
42 52 57 52 26 5 5 5 5 5 5 5 70 50 43 61 62 64 39 42 64 60 62 56 63 65 65 67 61 53 5
32 32 32 33 6 5 5 5 5 6 6 11 39 21 33 51 50 45 46 18 32 36 33 23 44 70 71 51 42 27 3
50 50 51 39 5 5 5 5 6 6 42 69 28 34 42 39 43 37 26 29 40 26 29 26 35 42 35 33 18 1
52 53 51 22 5 5 5 6 5 44 56 17 51 54 53 54 56 51 22 54 54 55 55 54 53 53 53 52 5
54 54 53 8 5 5 5 6 5 6 13 52 42 21 51 54 51 49 49 50 22 41 45 42 42 41 40 41 44 43 4
52 52 54 36 8 5 5 6 6 5 6 28 55 32 32 54 53 51 51 51 51 44 25 51 51 49 49 50 49 48 46 4
54 54 52 53 30 7 5 6 6 5 6 40 54 29 52 51 53 56 55 52 52 51 38 52 52 50 49 46 46 45 46 4
51 52 51 53 27 14 5 4 5 4 7 47 51 21 39 49 47 49 52 52 52 49 35 31 48 46 47 47 47 46 46 4
48 50 51 53 25 14 17 8 4 4 17 46 40 18 43 47 46 49 52 54 53 53 54 18 50 49 46 47 47 47 47 4
49 49 49 49 22 12 20 24 6 14 35 51 39 48 48 50 51 51 49 51 51 52 50 41 58 48 47 47 47 45 45 4
51 49 50 50 22 13 19 36 13 12 42 50 40 73 50 50 50 49 48 49 49 48 49 45 51 46 44 44 44 42 45 4
47 49 49 47 20 16 26 39 21 15 36 48 42 61 47 48 51 47 50 51 51 51 49 47 47 52 47 47 44 43 45 4

What computers see

Computer Vision: Challenges



Koala

Computer Vision: Challenges



Koala



Illumination



Object Pose



Clutter



Occlusions



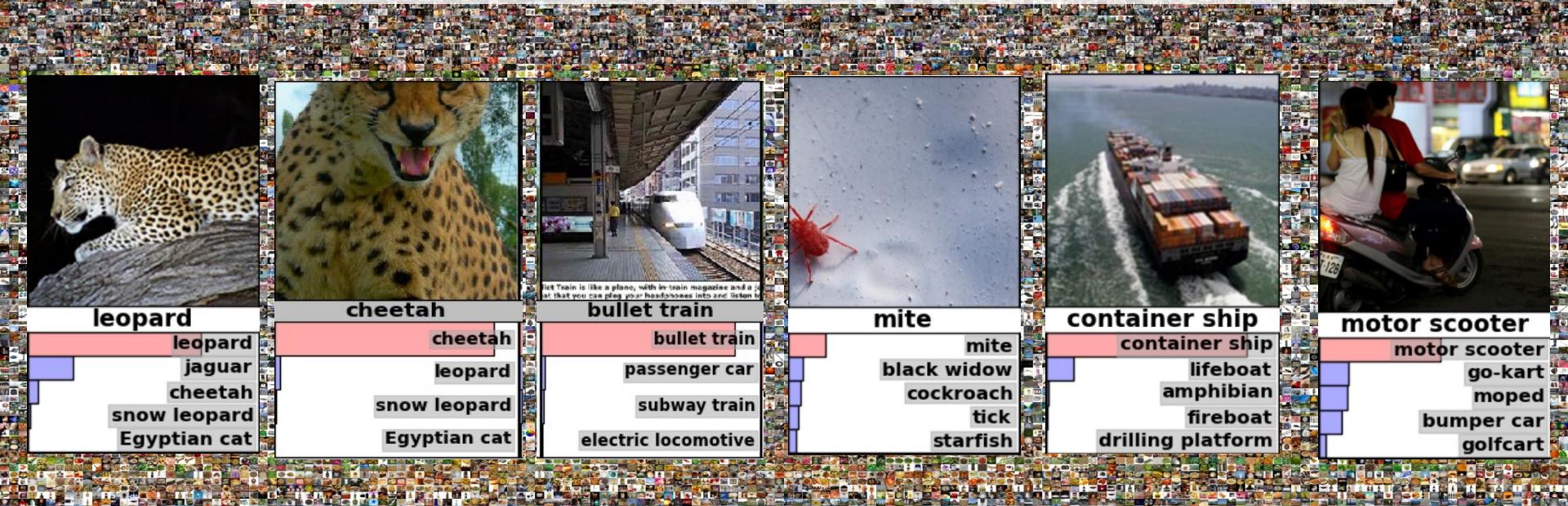
Intra-class
appearance



Viewpoint

Imagenet Challenge

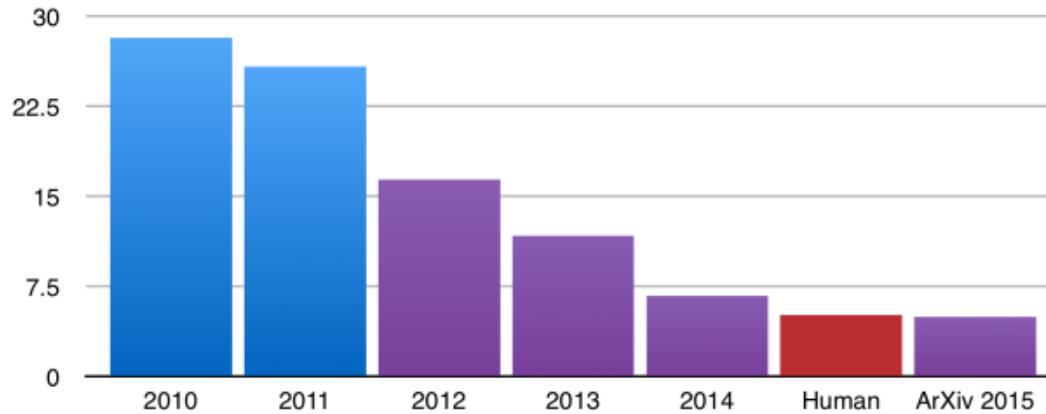
The Image Classification Challenge:
1 000 Object Classes
1 431 167 Images



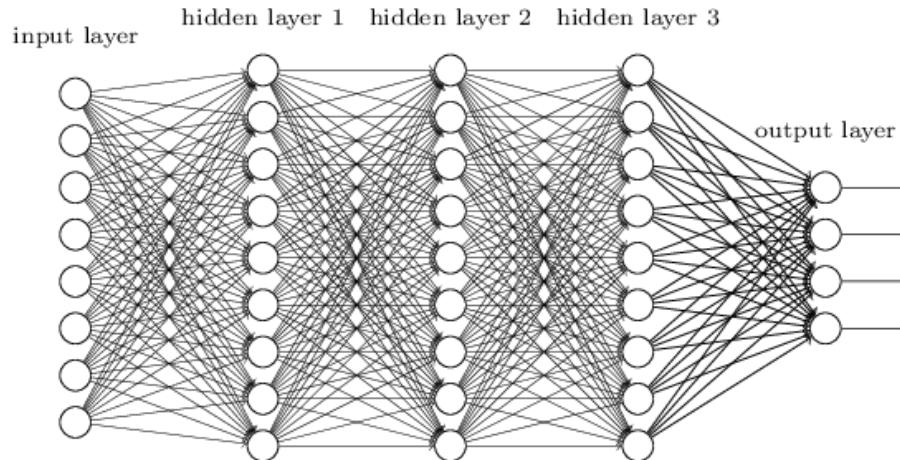
Imagenet Challenge

The Image Classification Challenge:
1 000 Object Classes
1 431 167 Images

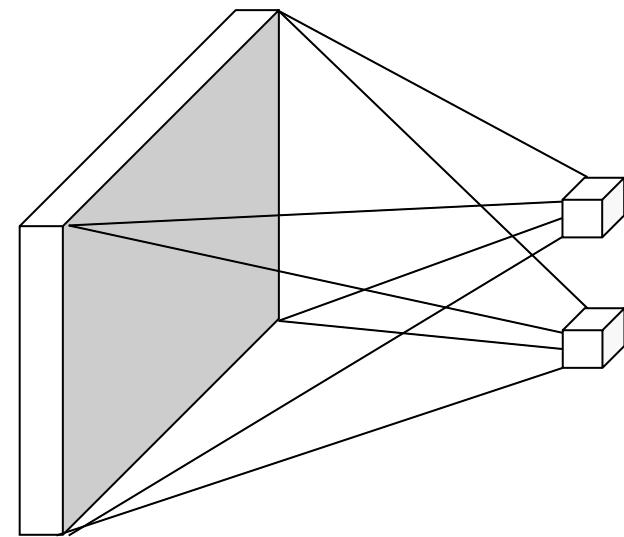
ILSVRC top-5 error on ImageNet



Convolutional Neural Networks



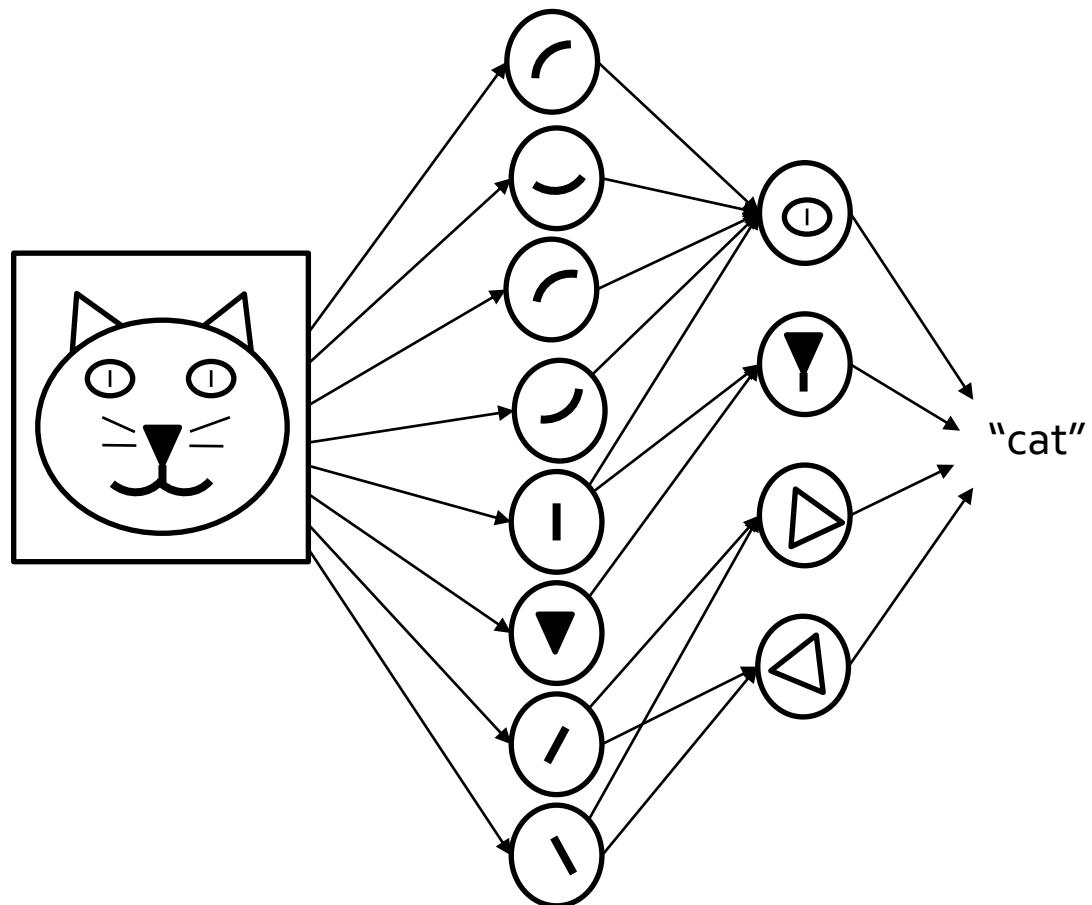
Multilayer Perceptron



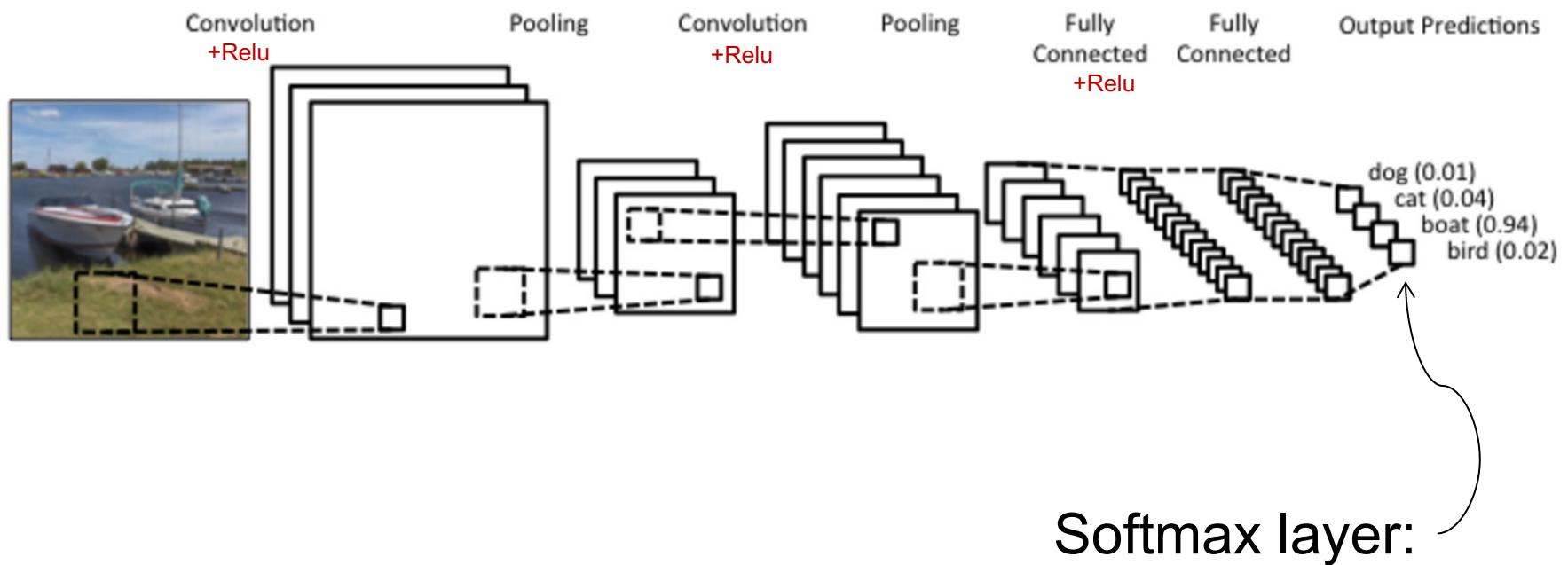
Convolutional Neural Network (CNN)

Convolutional Neural Networks

- We use convolutional layers to capture local spatial information.
 - We can use a sequence of those layers.
 - The stacking of successive layers give us a hierarchical representation.



Convolutional Neural Networks



$$P(c | \mathbf{x}) = \frac{\exp(\mathbf{w}_c \cdot \mathbf{x})}{\sum_{k=1}^C \exp(\mathbf{w}_k \cdot \mathbf{x})}$$

Convolution

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

Image

1	0	1
0	1	0
1	0	1

Kernel

Convolution

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

Image

X

1	0	1
0	1	0
1	0	1

Kernel

=

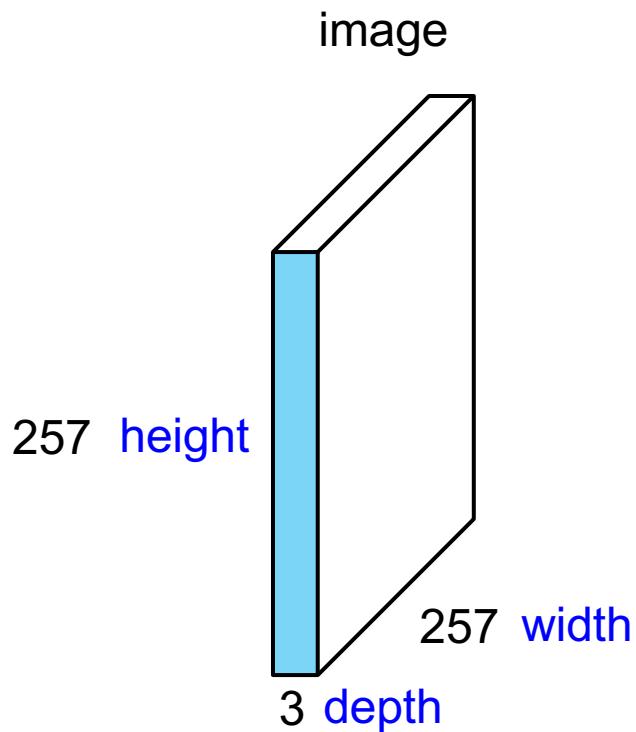
1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Image

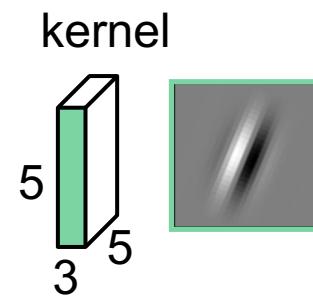
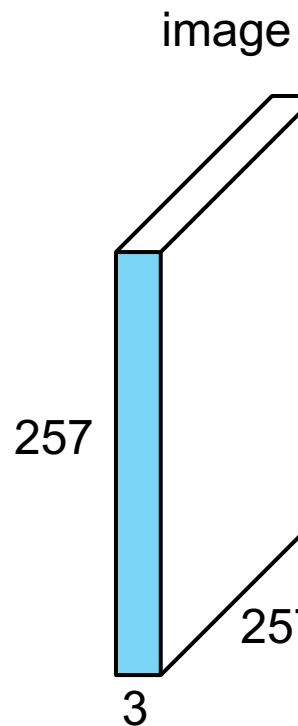
4		

Convolved
Feature

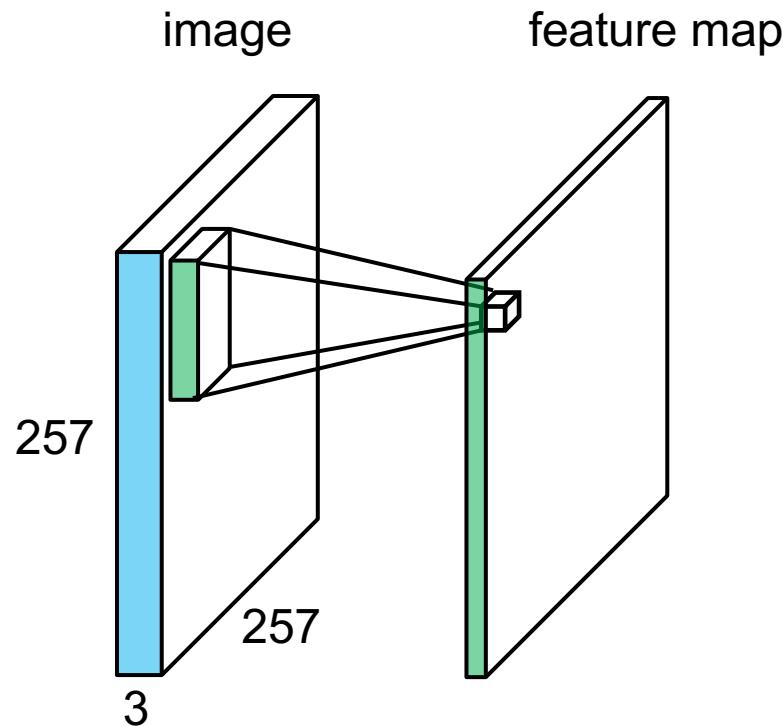
Convolutional Neural Networks



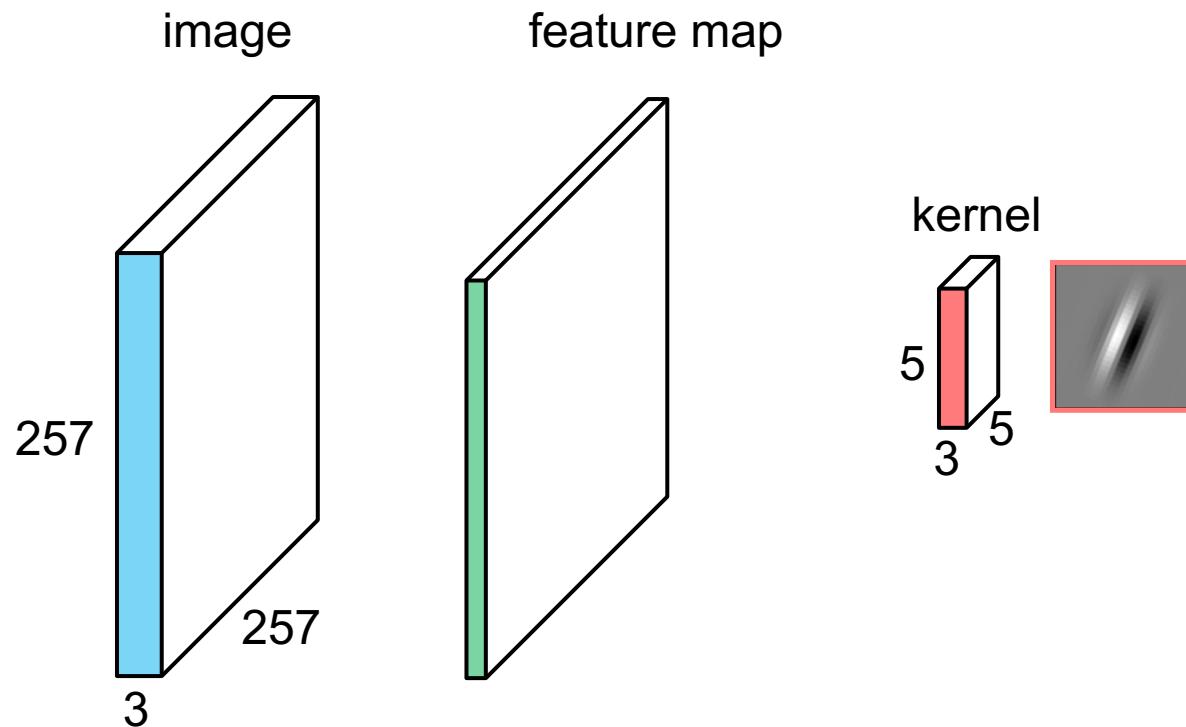
Convolutional Neural Networks



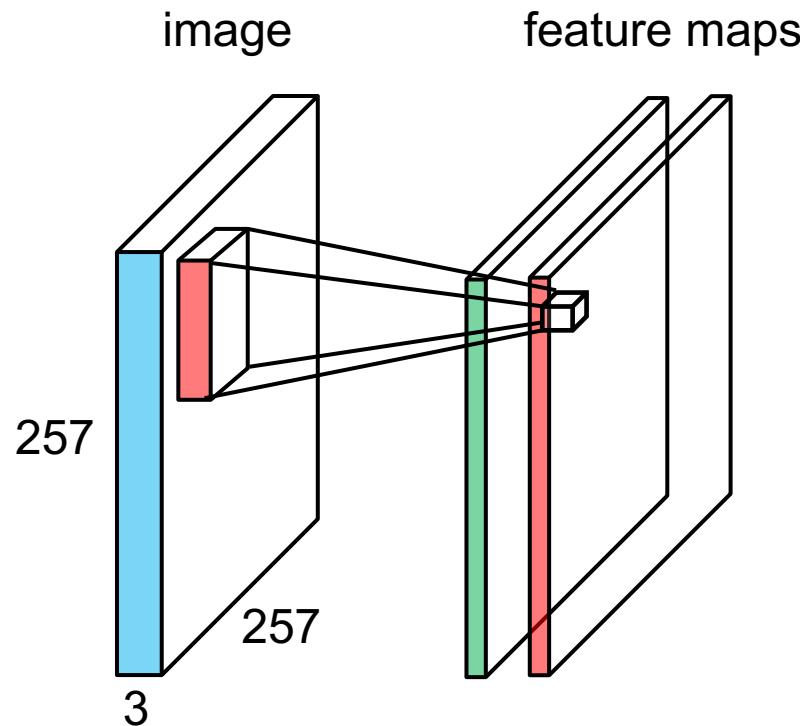
Convolutional Neural Networks



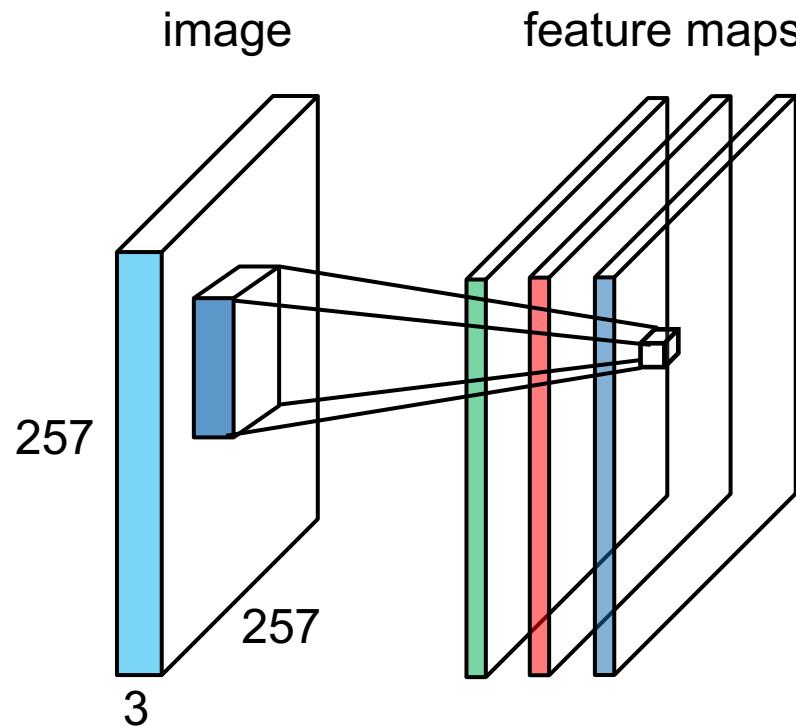
Convolutional Neural Networks



Convolutional Neural Networks



Convolutional Neural Networks



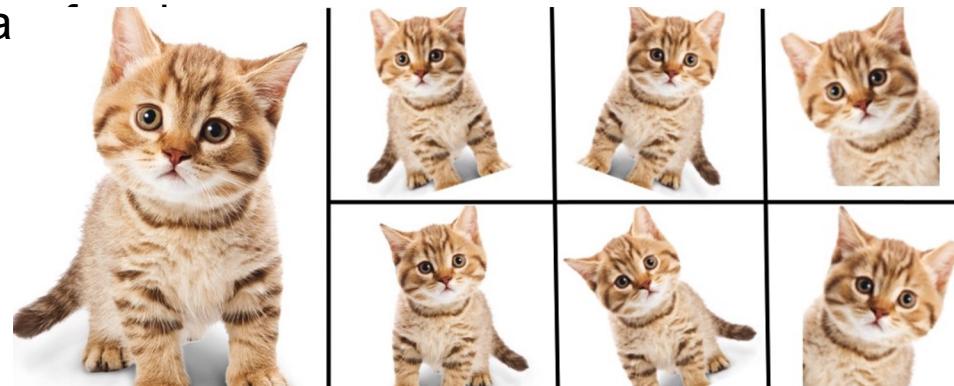
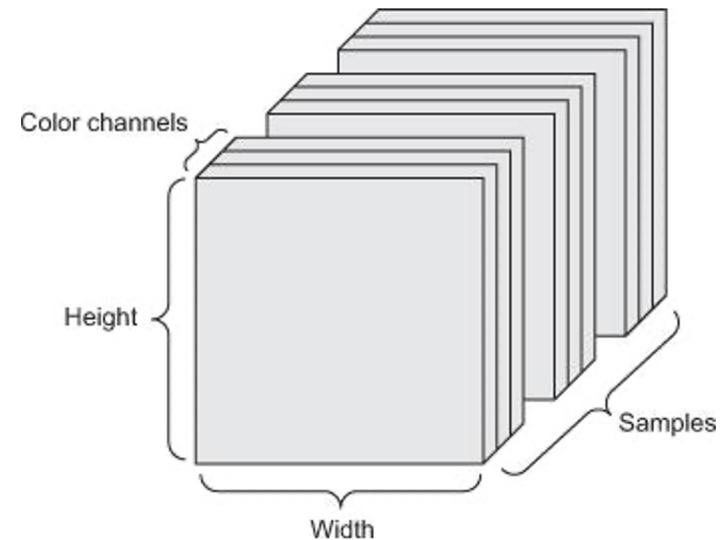
Convolutional Neural Networks



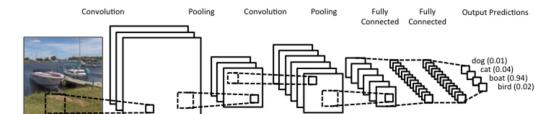
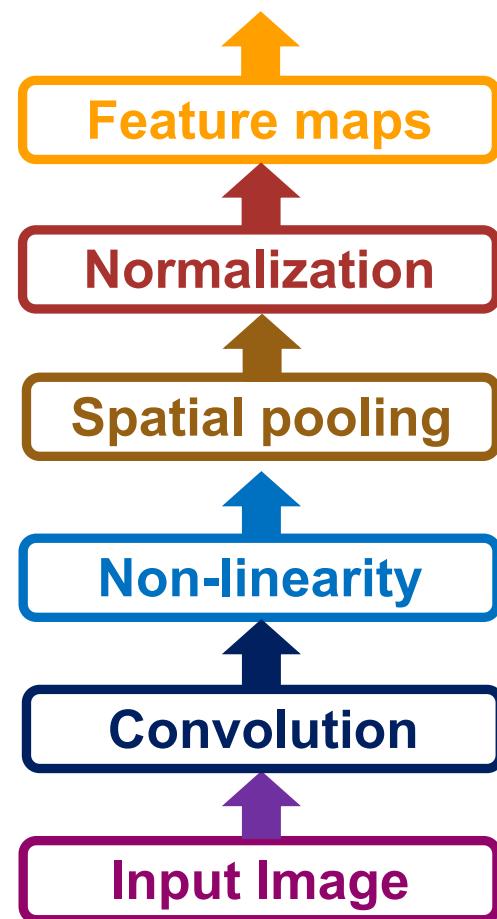
Examples of learned kernels/weights

Convolutional Neural Networks

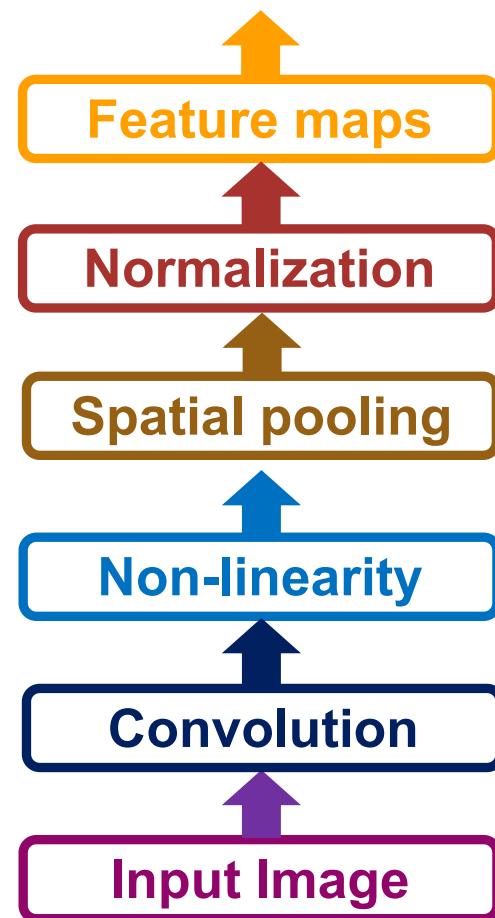
- Color image mini-batches are 4D tensors:
width × height × color channels × samples
- Plenty of big datasets for training exist, e.g., ImageNet with 1,2 million images in 1000 classes
- Data augmentation for small datasets:
generate more training data by transforming existing data
- E.g., shifting, rotation, cropping, Scaling, adding noise, etc ...



Convolutional Neural Networks

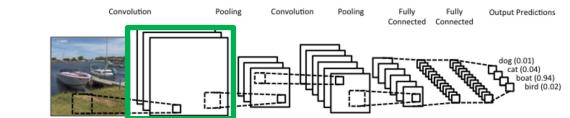


Convolutional Neural Networks



Convolution

- Apply learned filter weights.
- One feature map per filter.

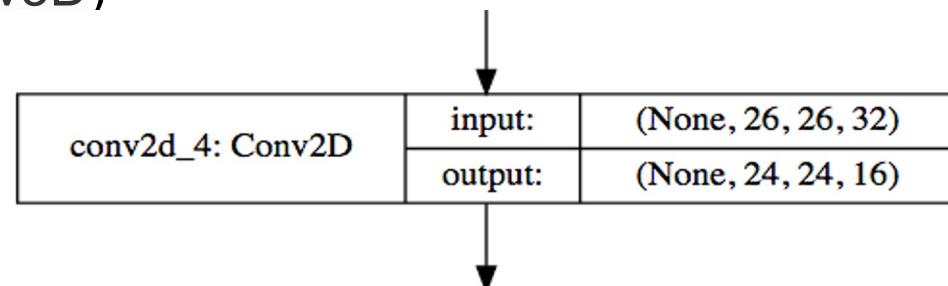


Convolutional Neural Networks

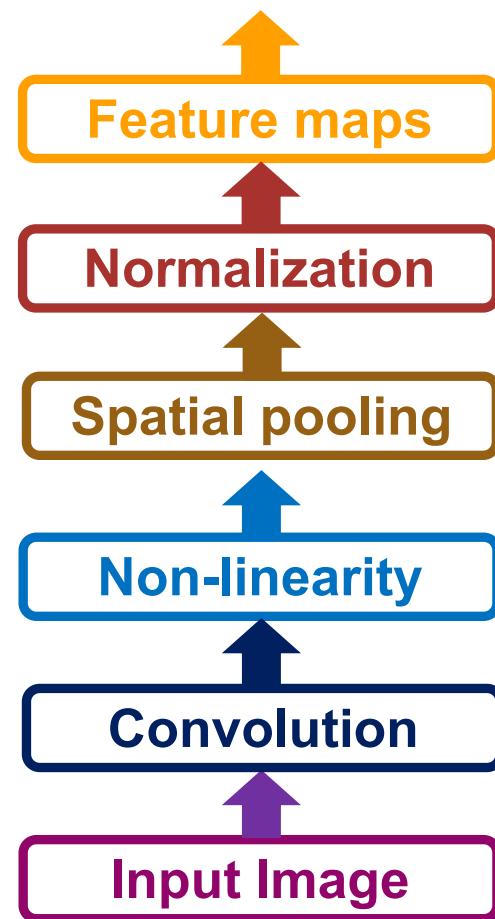
- Input: tensor of size $N \times W_i \times H_i \times C_i$
- Hyperparameters:
 - K : number of filters
 - w, h : kernel size
 - padding: how to handle image borders
 - activation function
- Output: tensor of size $N \times W_o \times H_o \times K$
- In tf.keras:

```
layers.Conv2D(filters, kernel_size,
               padding, activation)
```

(there is also Conv1D and Conv3D)

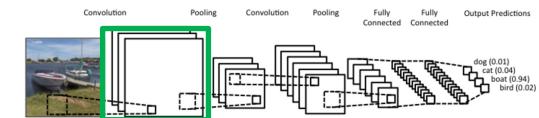
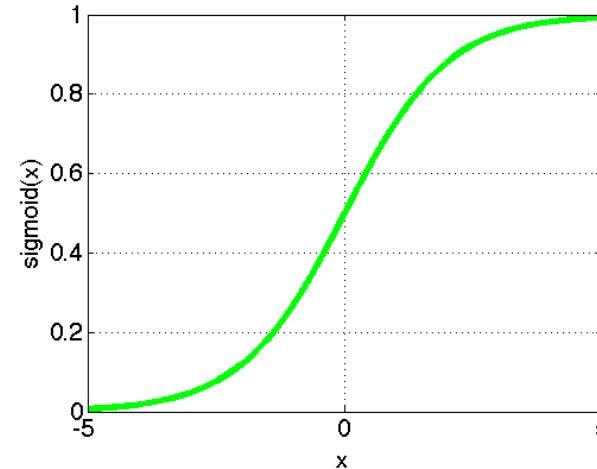


Convolutional Neural Networks

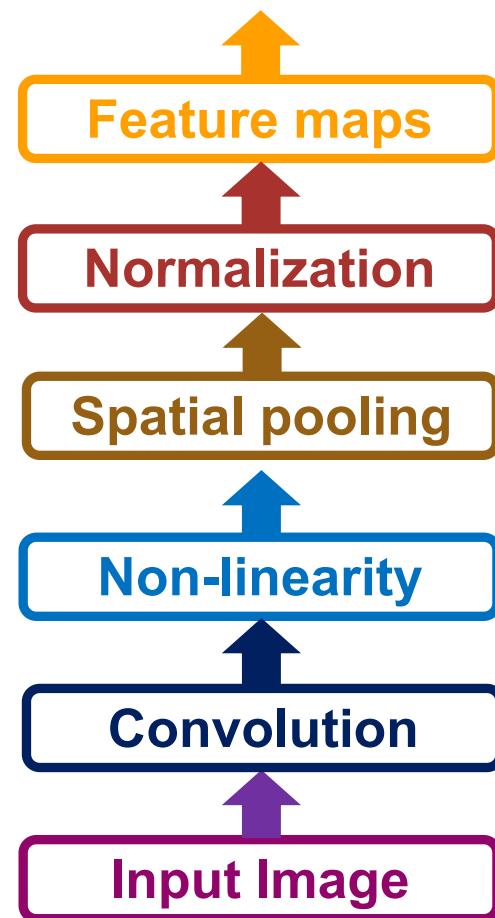


Activation Functions

- Per-feature independent.
- **Sigmoid:** $1/(1+\exp(-x))$

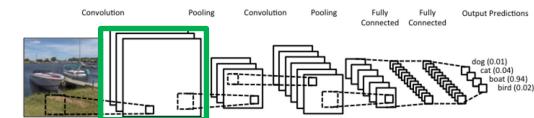
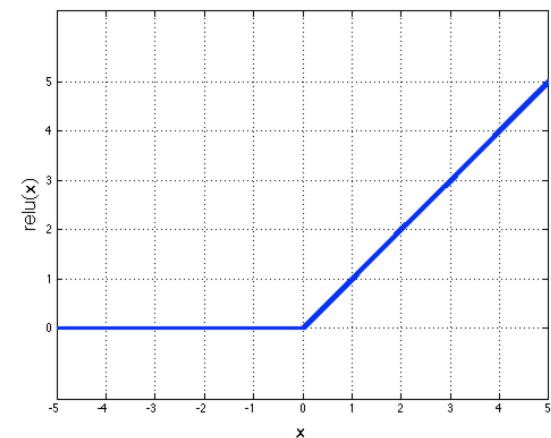
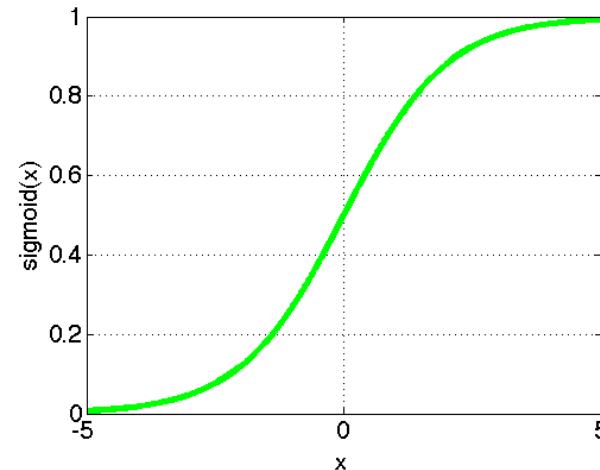


Convolutional Neural Networks

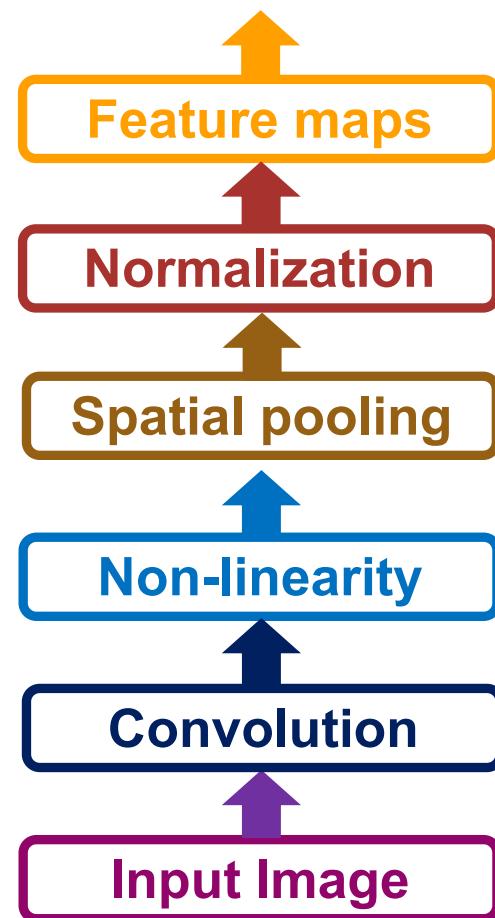


Activation Functions

- Per-feature independent.
- Sigmoid: $1/(1+\exp(-x))$
- Relu: $\max(0,x)$

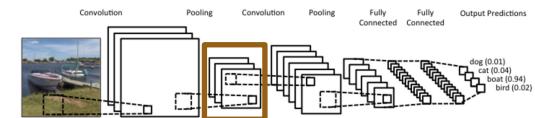
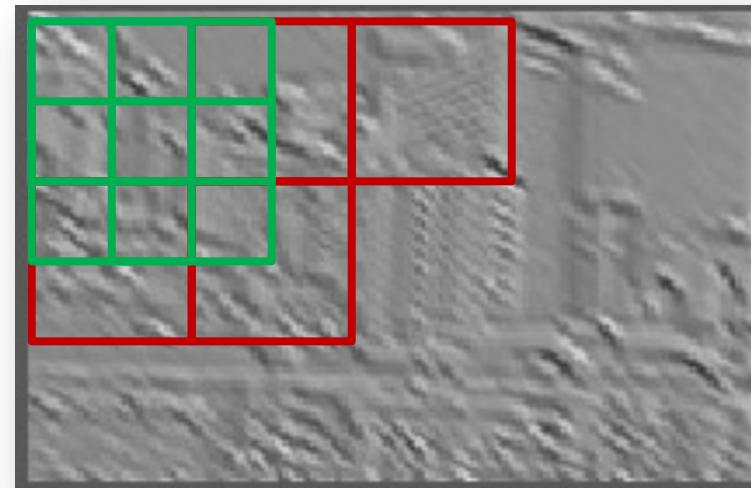


Convolutional Neural Networks

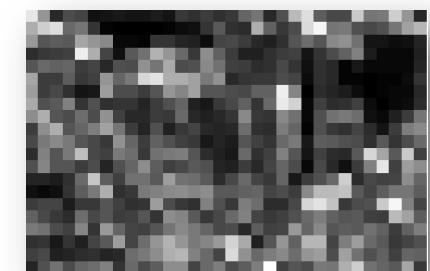


Spatial Pooling

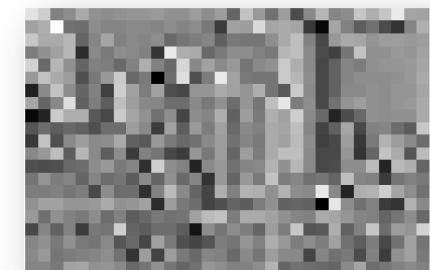
- Provides invariance to small transformations.
- Larger receptive fields (see more of the input).



Max

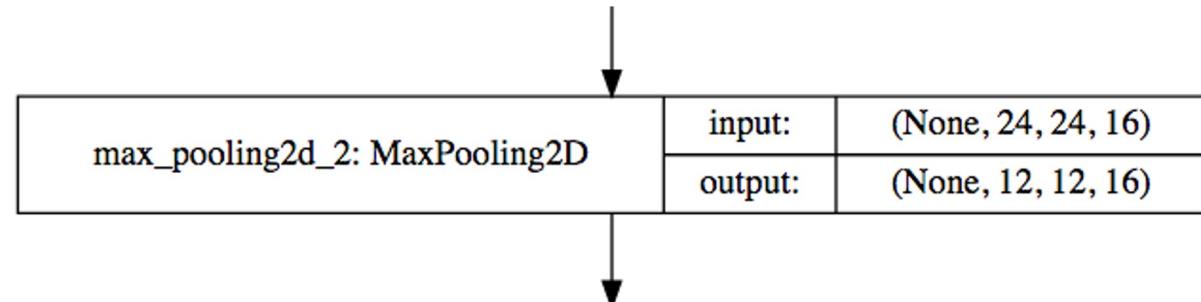
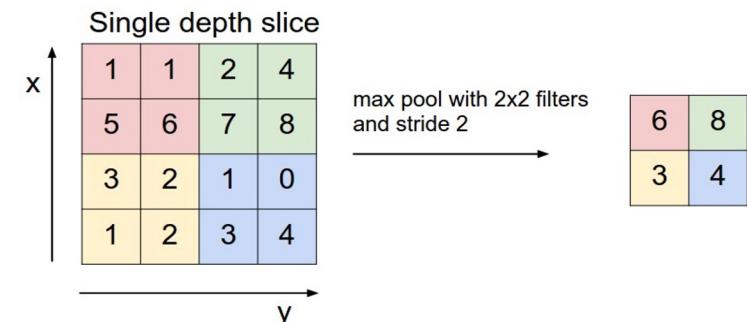


Sum

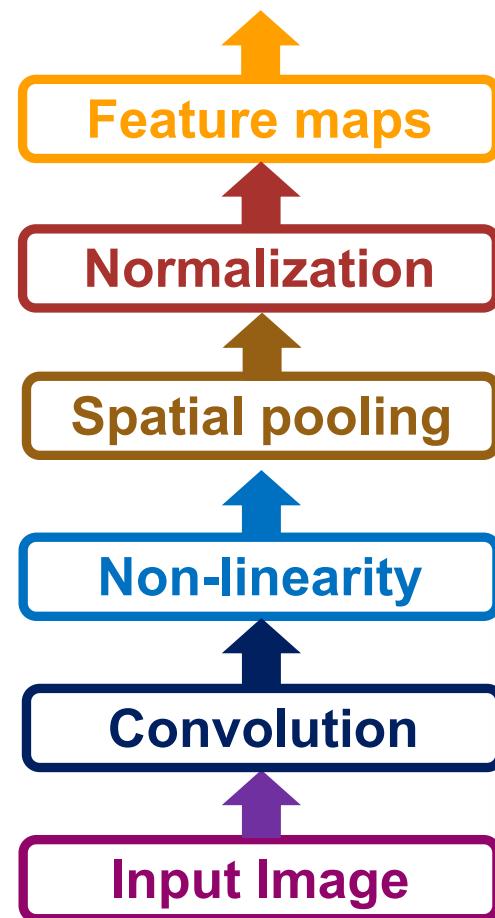


Convolutional Neural Networks

- Used to reduce the spatial resolution
 - independently on each channel
 - reduce complexity and number of parameters
- MAX operator most common
 - sometimes also AVERAGE
- In tf.keras:
`layers.MaxPooling2D(pool_size)`
`layers.AveragePooling2D(pool_size)`

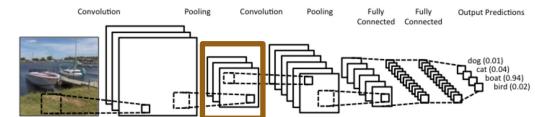
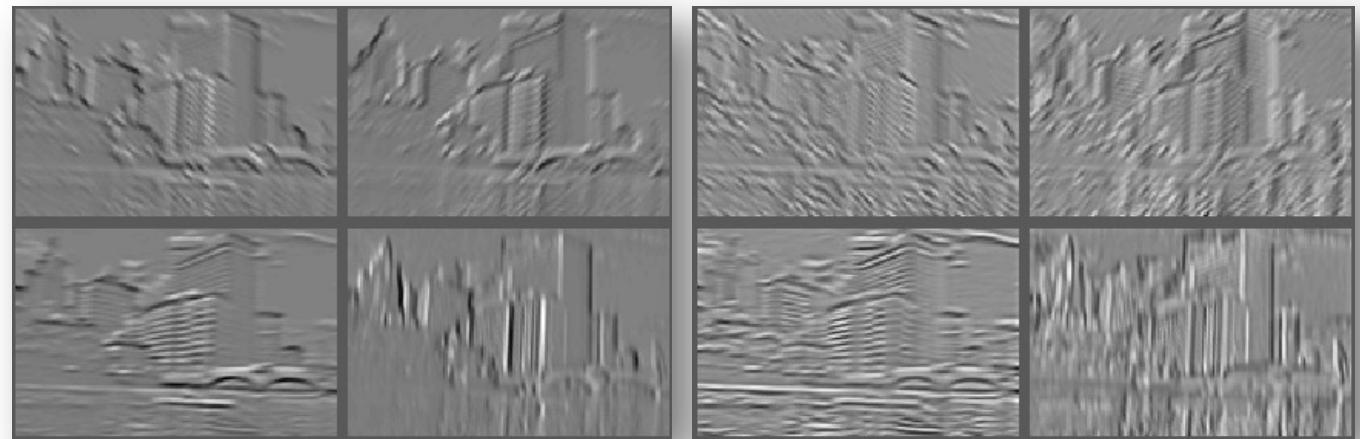


Convolutional Neural Networks



Normalization:

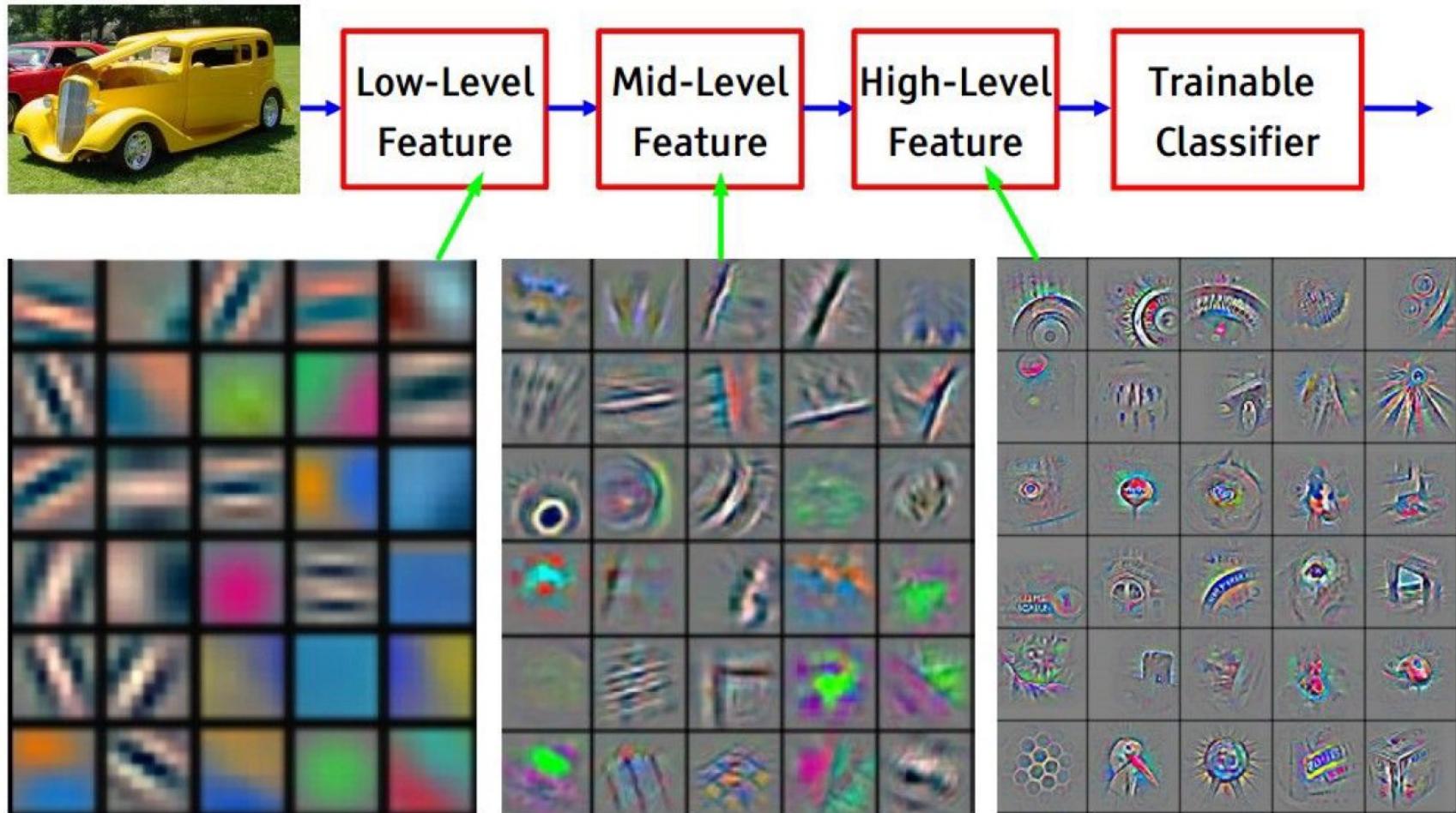
- Equalizes/normalizes feature maps.
- Suppresses low-frequencies/Highlight peaks.



Convolutional Neural Networks

- Flatten
 - flattens the input into a vector (typically before dense layers)
- Dropout
 - similar as with dense layers
- In `tf.keras`:
`layers.Flatten()`
`layers.Dropout(rate)`

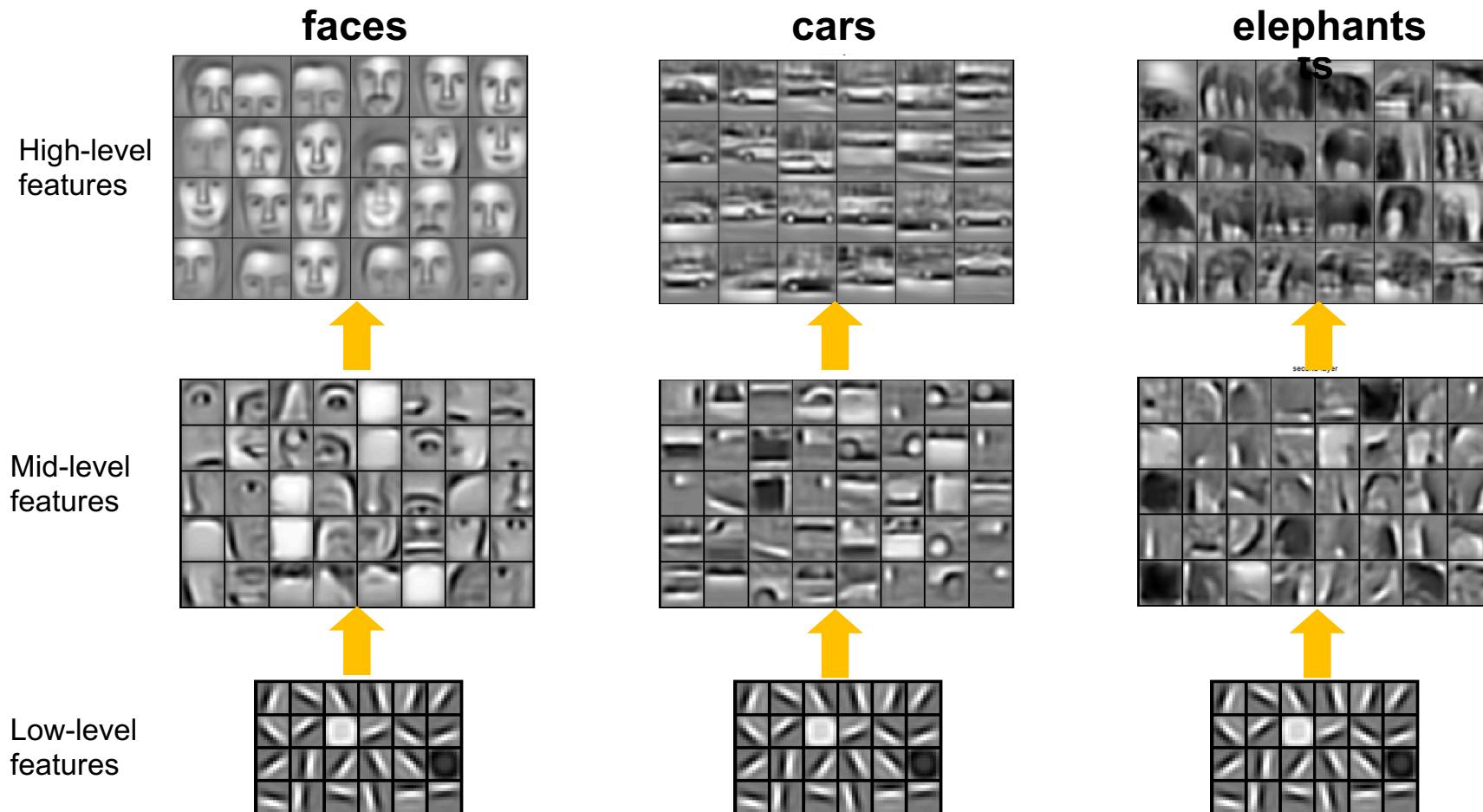
Convolutional Neural Networks



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

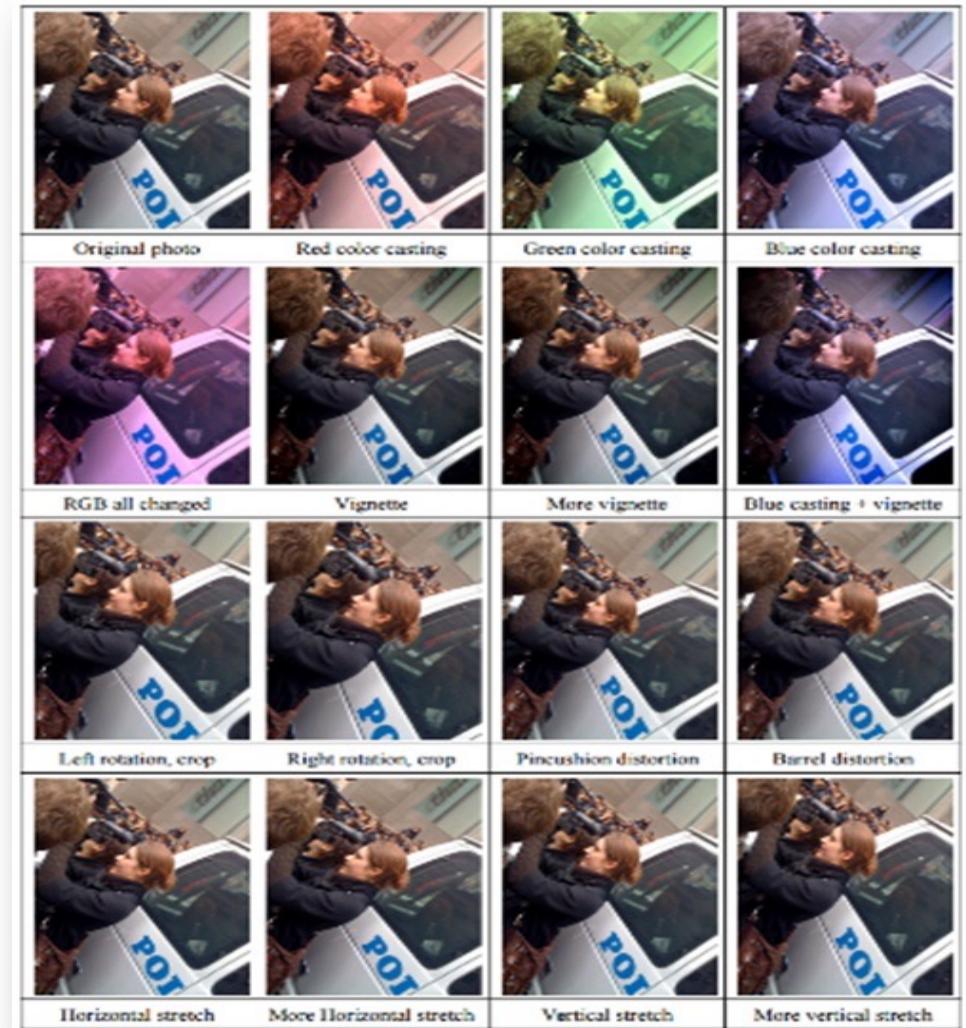
Convolutional Neural Networks

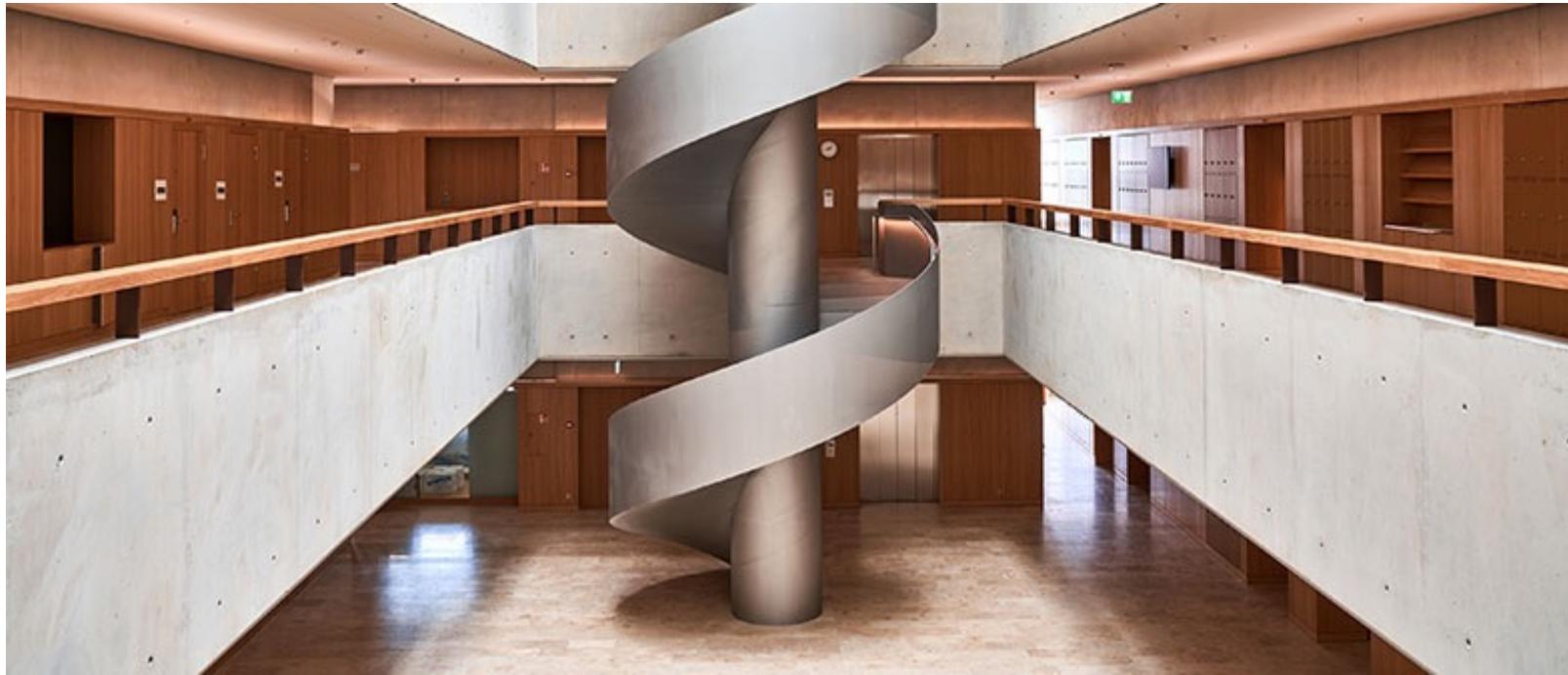
Examples of learned object parts from object categories



Convolutional Neural Networks

- Create *virtual* training samples
 - Horizontal flip
 - Random crop
 - Color casting
 - Geometric distortion





CAS Machine Learning

Deep Learning: From Core Foundations to Applications

Javier Montoya, Dr. sc. ETH
javier.montoya@hslu.ch