# 《多模态机器学习》

## 第三章 视觉模态与卷积神经网络

黄文炳
中国人民大学高瓴人工智能学院
hwenbing@126.com

2024年秋季

# 内容提纲

① 图片表示

② 卷积神经网络

③ 卷积神经网络的可视化

④ 3D卷积神经网络

# 内容提纲

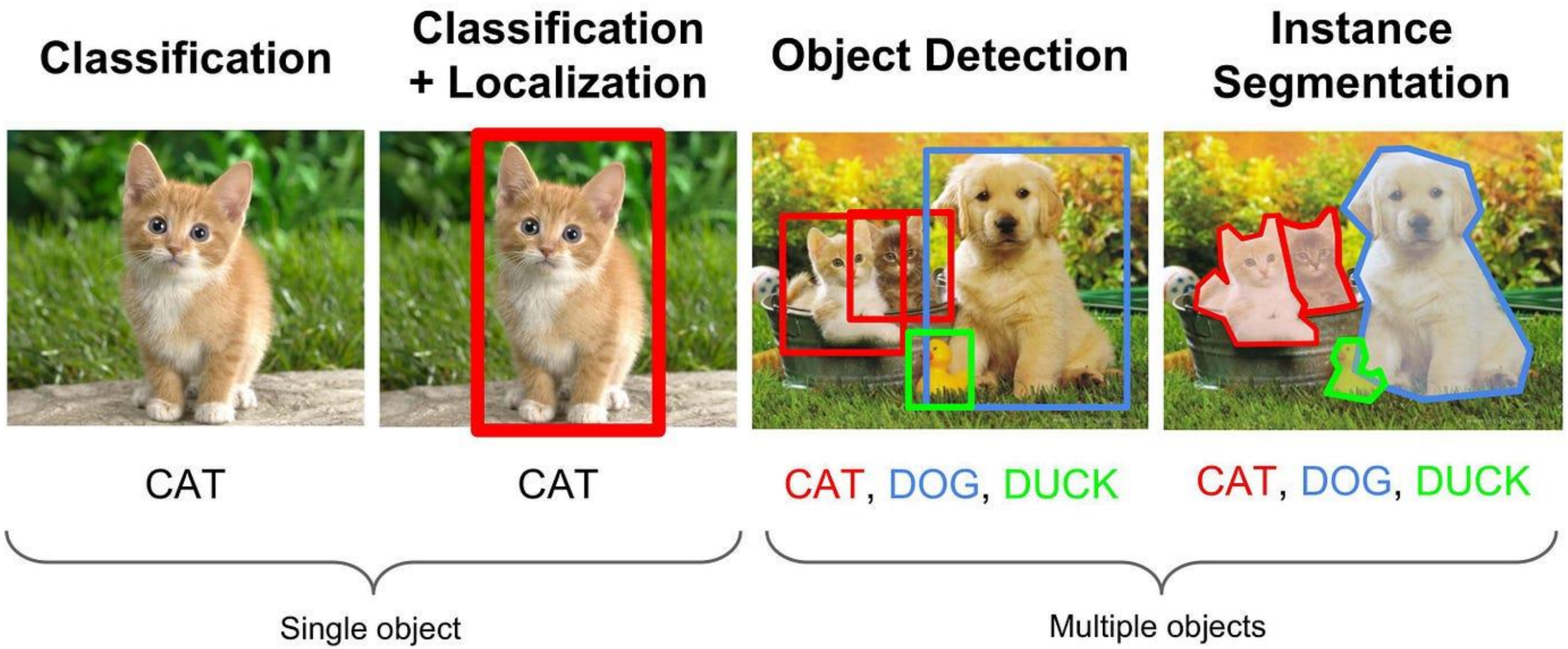
① 图片表示

② 卷积神经网络

③ 卷积神经网络的可视化

④ 3D卷积神经网络

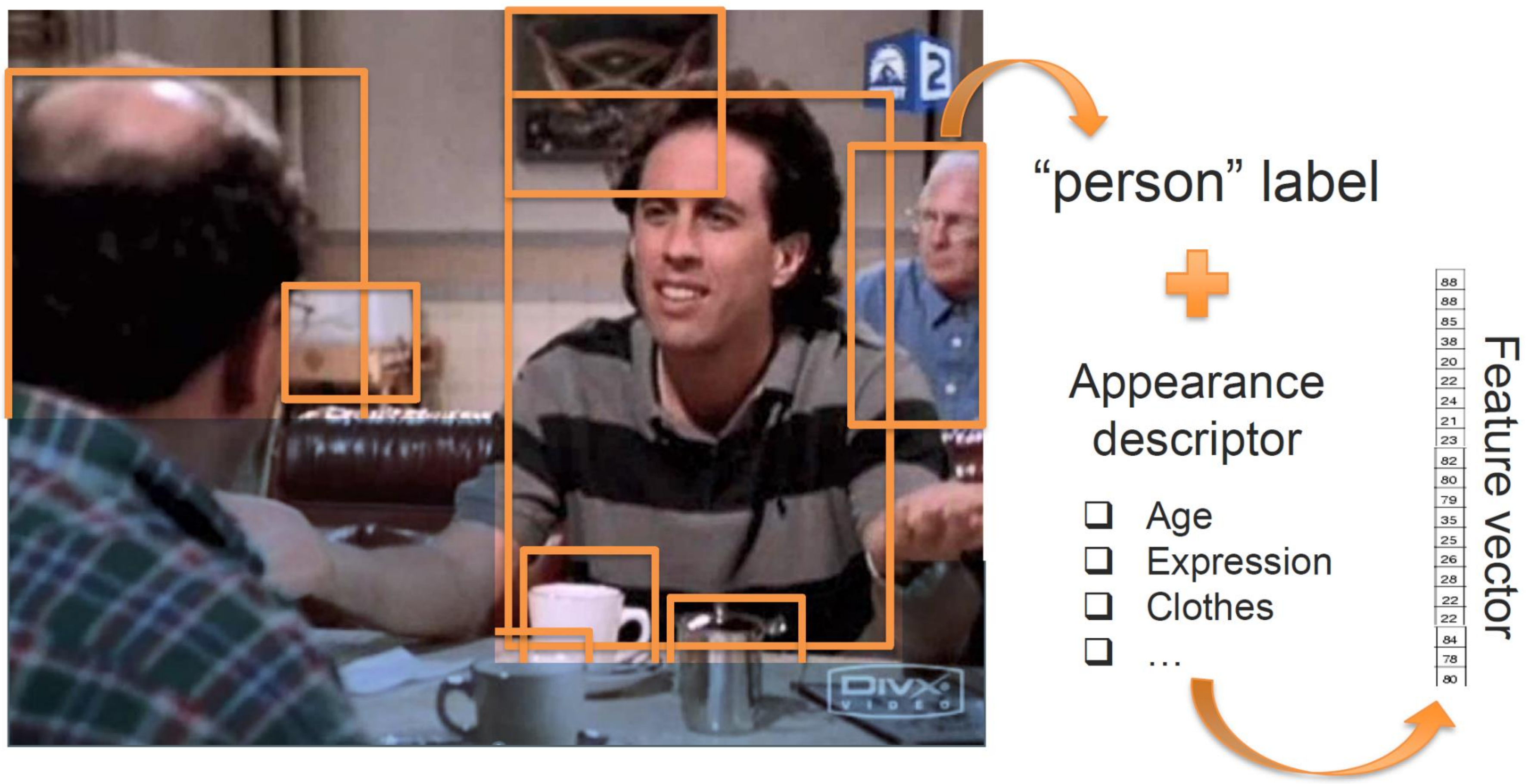
# Computer Vision Tasks



# How Would You Describe This Image?


| |
|---|
| 88 |
| 88 |
| 85 |
| 38 |
| 20 |
| 22 |
| 24 |
| 21 |
| 23 |
| 82 |
| 80 |
| 79 |
| 35 |
| 25 |
| 26 |
| 28 |
| 22 |
| 22 |
| 84 |
| 78 |
| 80 |
| ⋮ |

# How Would You Describe This Image?



# Object Descriptors

How to represent and detect an object?

## Many approaches over the years…

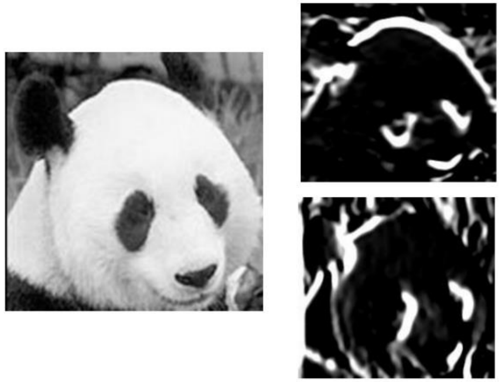
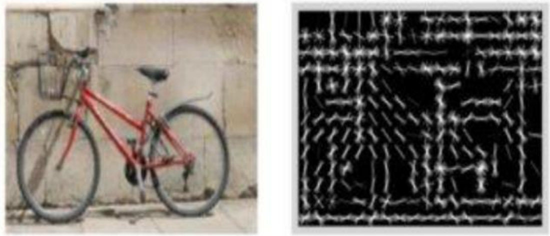


Image gradient

Edge detection



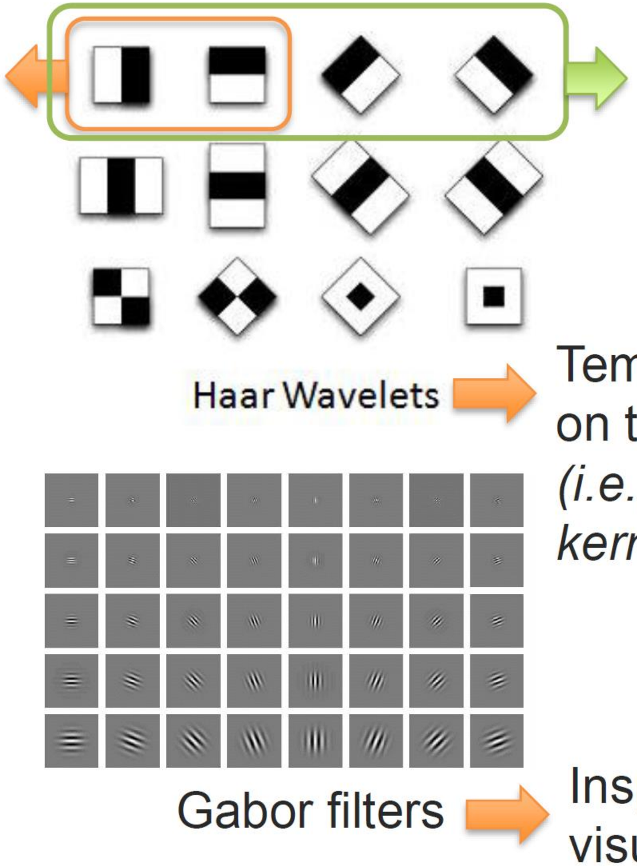
Histograms of Oriented Gradients

Optical Flow

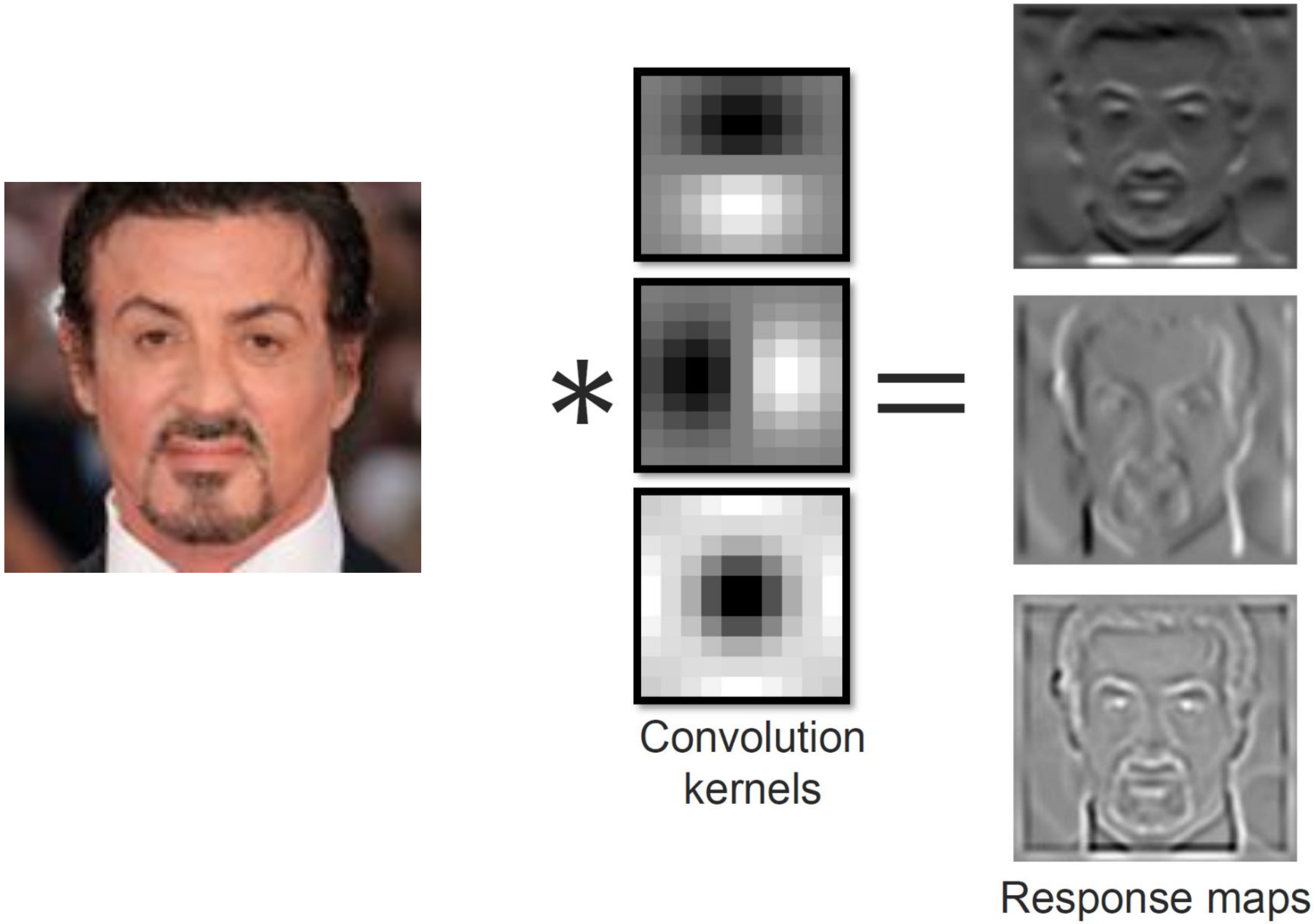
# Object Descriptors

How to represent and detect an object?

## Many approaches over the years…



# Convolution Kernels

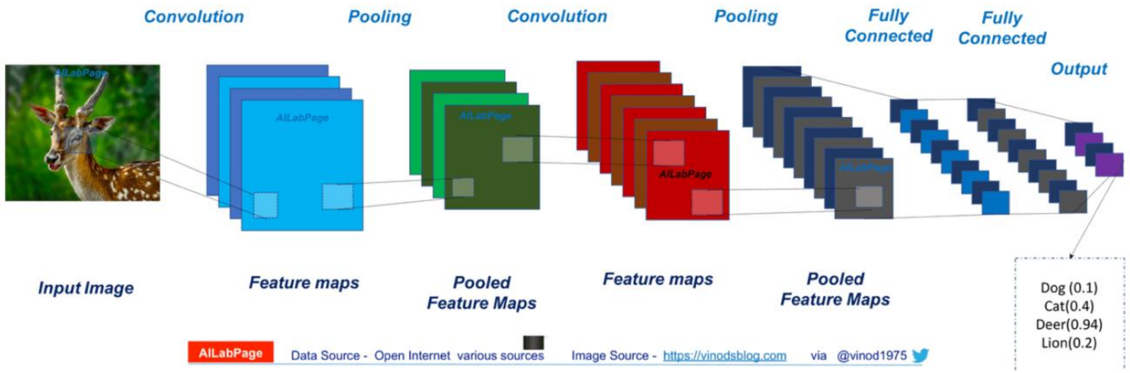


# Object Descriptors

How to represent and detect an object?

## Many approaches over the years…

Convolutional Neural Network (CNN)




More details about CNNs is coming…

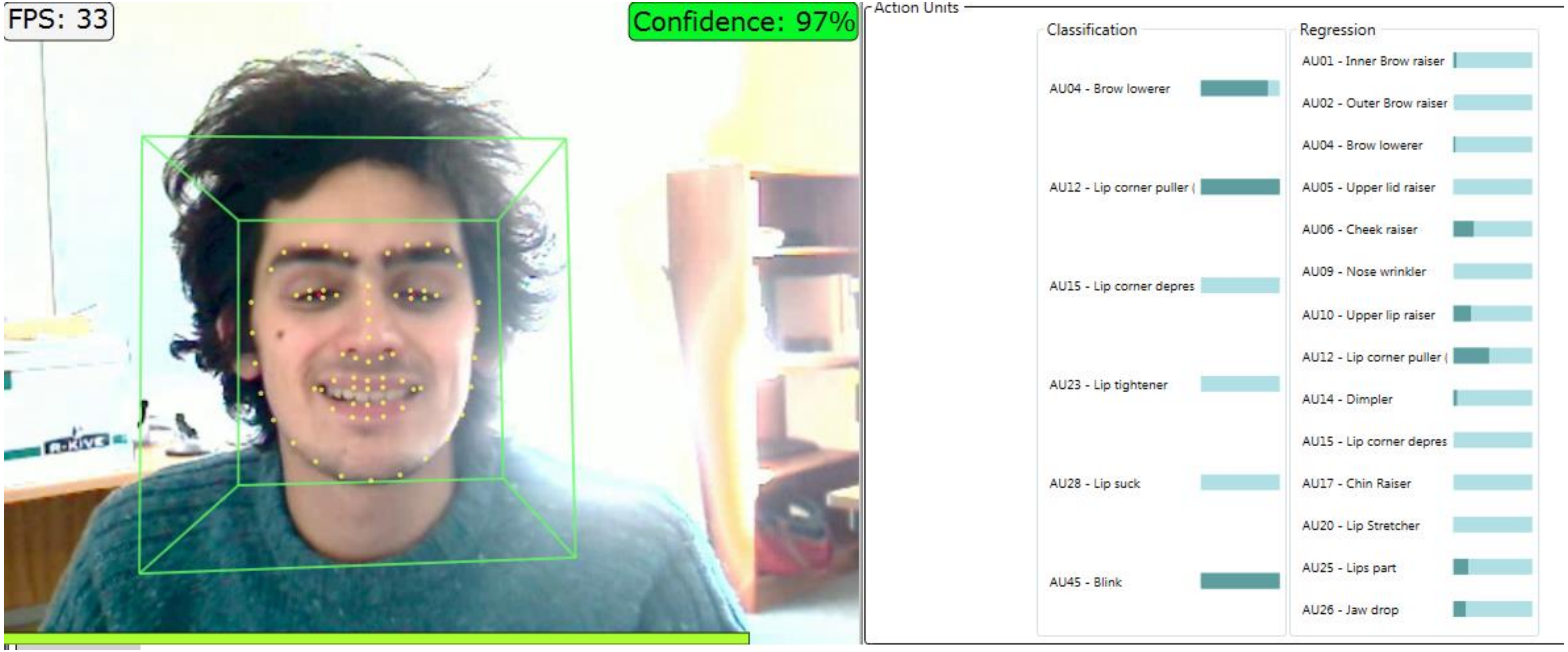
… and we will also talk about visual transformers in coming weeks…

And images are more than a list of objects!

# One representation, lots of tasks


https://github.com/facebookresearch/detectron2

# Facial expression analysis




[OpenFace: an open sourcefacial behavior analysis toolkit, T. Baltrušaitiset al., 2016]

# Articulated Body Tracking: OpenPose


https://github.com/CMU-Perceptual-Computing-Lab/openpose

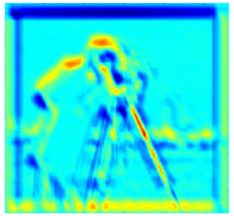
# 内容提纲

# The issue of MLP

**Input**

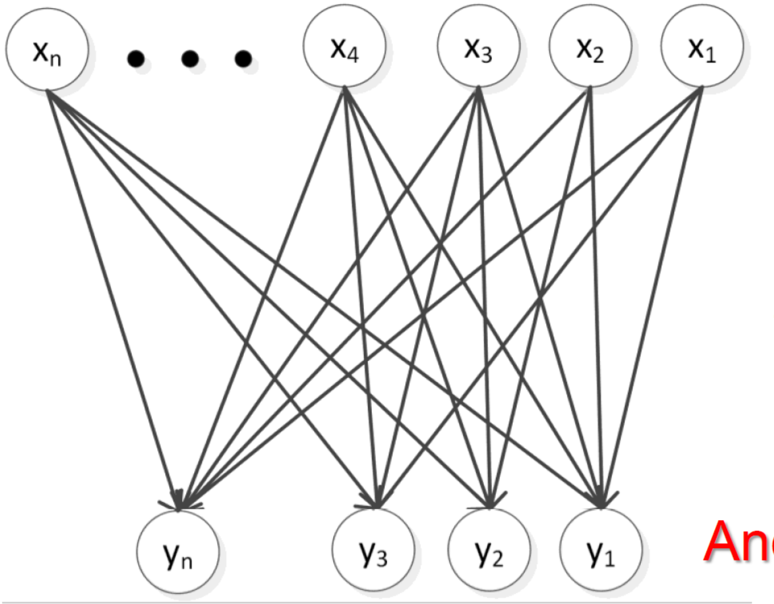
(image)

**Output**



(response map)

**Input:** all pixels




**Output:** kernel responses

**Not efficient!**

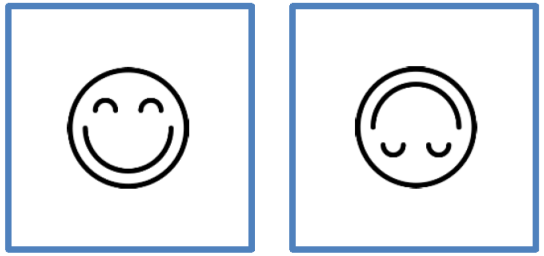
$200 \times 200$ image
requires
$40{,}000 \times n$ parameters

(where $n$ is size of kernel)

And it may learn different kernels for different pixel positions

➡ Not translation invariant

# The issue of MLP



2 Data Points – Which one is up?

- MLP can easily learn this task (possibly with only 1 neuron!)

What happens if the face is slightly translated?

- The model should still be able to classify it

Conventional MLP models are not translation invariant!

- But CNNs are kernel-based, which helps with translation invariance and reduce number of parameters

# Convolution Neural Layer

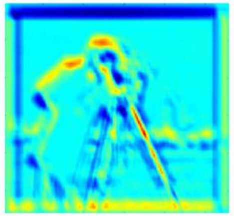
**Input**

(image)

Weighted sum

$Wx$

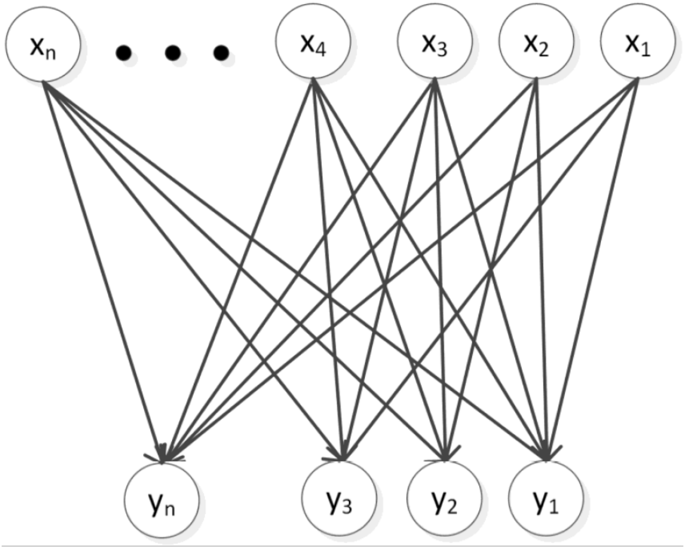
**Output**



(response map)

$$\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x}$$

**Input:** all pixels




**Output:** kernel responses

Example with 1D kernel:

| | | |
|---|---|---|
| $w_1$ | $w_2$ | $w_3$ |

Convolution kernel

# Convolution Neural Layer

**Modification 1:** Sliding window – Only apply the kernel to a small region

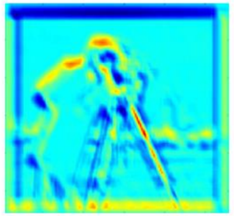
**Input**

(image)

Weighted sum

$Wx$

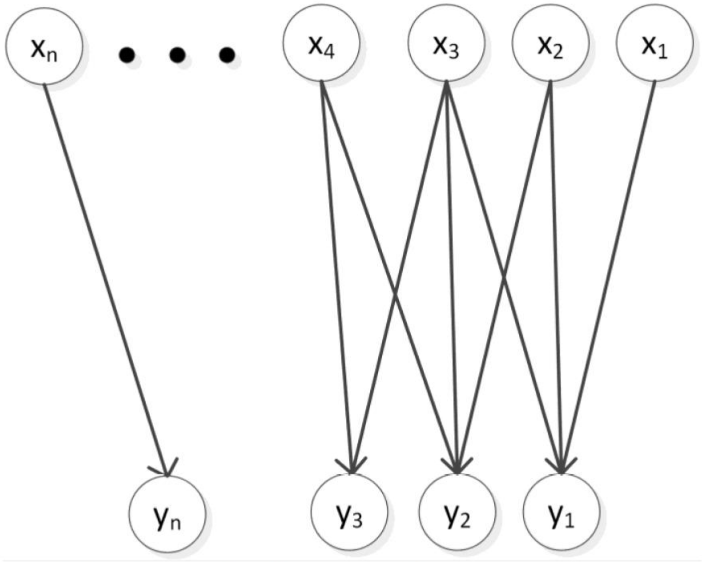
**Output**



(response map)

$$\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x}$$

**Input:** all pixels




**Output:** kernel responses

Example with 1D kernel:

| | | |
|---|---|---|
| $w_1$ | $w_2$ | $w_3$ |

Convolution kernel

# Convolution Neural Layer

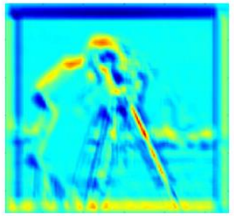
**Modification 2:** Same kernel applied to all sliding windows

**Input**

(image)

Weighted sum
$Wx$

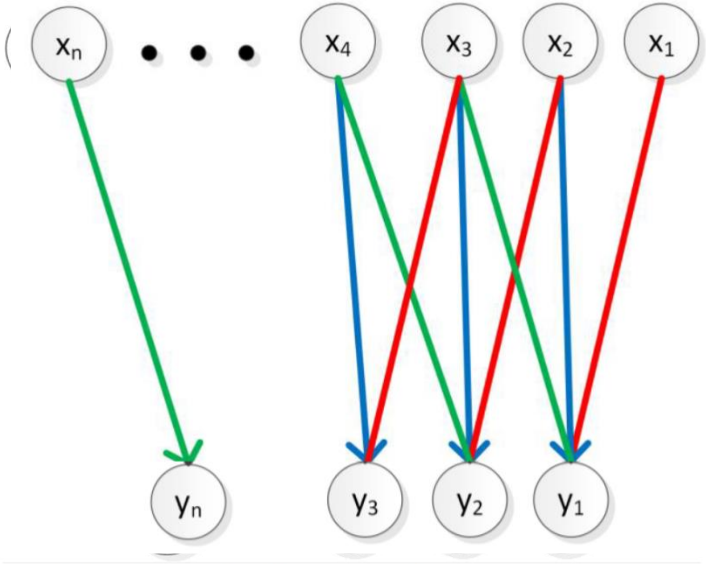
**Output**



(response map)

$$y = Wx$$

**Input:** all pixels




**Output:** kernel responses

Example with 1D kernel:

| $w_1$ | $w_2$ | $w_3$ |
|---|---|---|

Convolution kernel

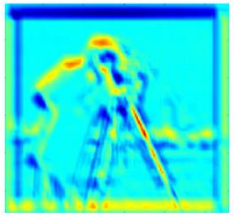
# Convolution Neural Layer

**Modification 2:** Same kernel applied to all sliding windows

Input

(image)

Output



(response map)

$$y = Wx$$

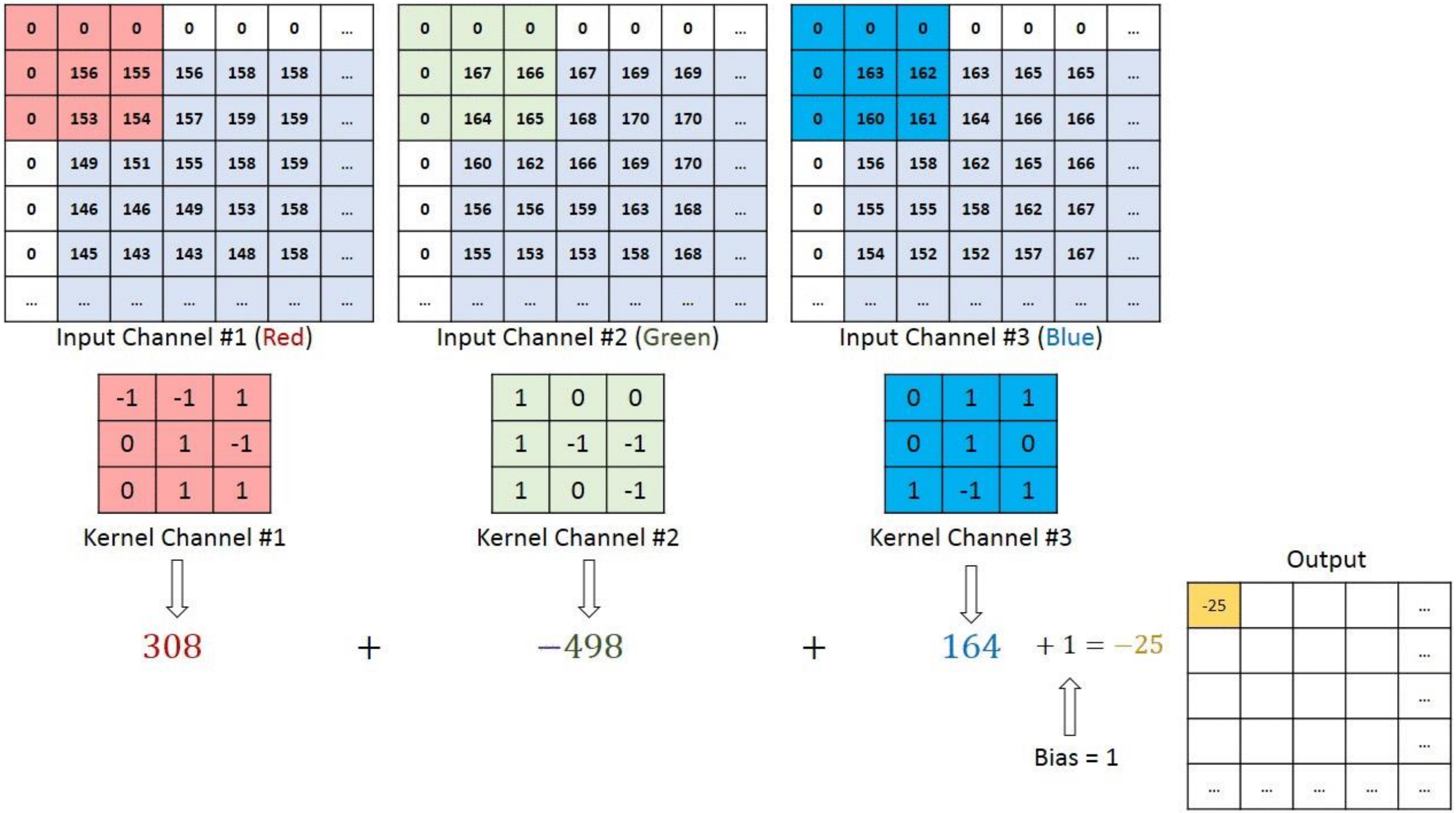
$$W = \begin{pmatrix} w_1 & w_2 & w_3 & & 0 & 0 & 0 \\ 0 & w_1 & w_2 & \cdots & 0 & 0 & 0 \\ 0 & 0 & w_1 & & 0 & 0 & 0 \\ & \vdots & & \ddots & & \vdots & \\ 0 & 0 & 0 & & w_3 & 0 & 0 \\ 0 & 0 & 0 & \cdots & w_2 & w_3 & 0 \\ 0 & 0 & 0 & & w_1 & w_2 & w_3 \end{pmatrix}$$

Example with 1D kernel:

| | | |
|---|---|---|
| $w_1$ | $w_2$ | $w_3$ |

- Can be implemented efficiently on GPUs
- *W* will be 3D: 3rd dimension allows for multiple kernels

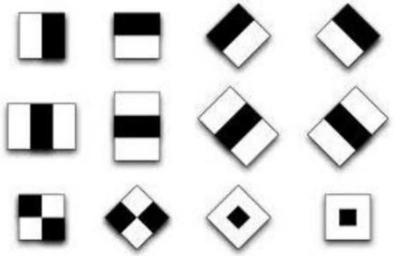
# Convolution Neural Layer



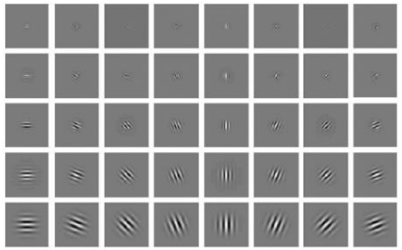

https://medium.datadriveninvestor.com/convolutional-neural-networks-3b241a5da51e

# Predefined vs Learned Kernels

## Predefined kernels



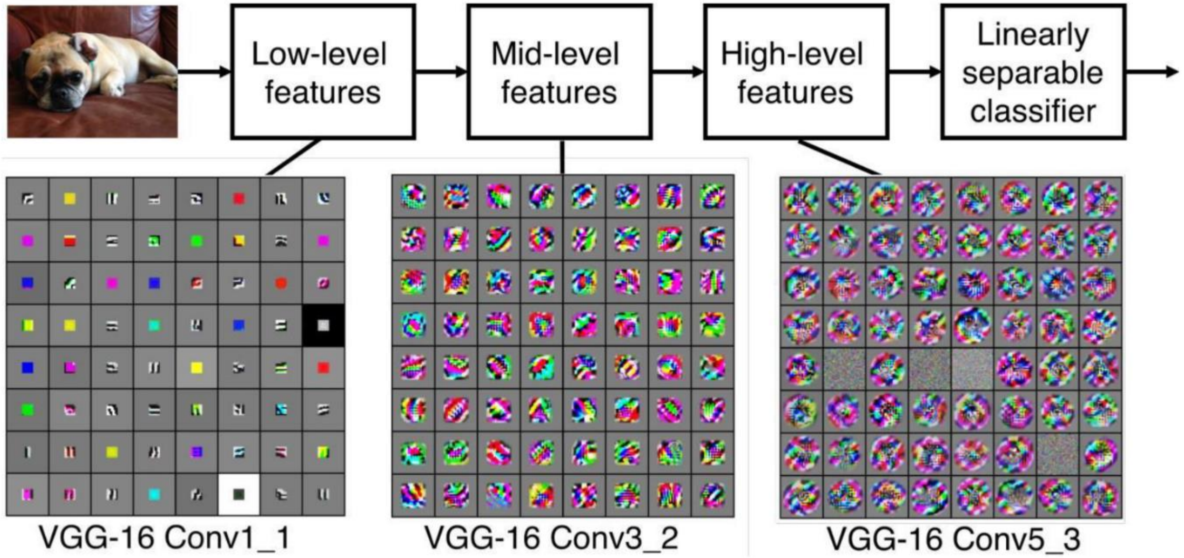
Haar Wavelets



Gabor filters

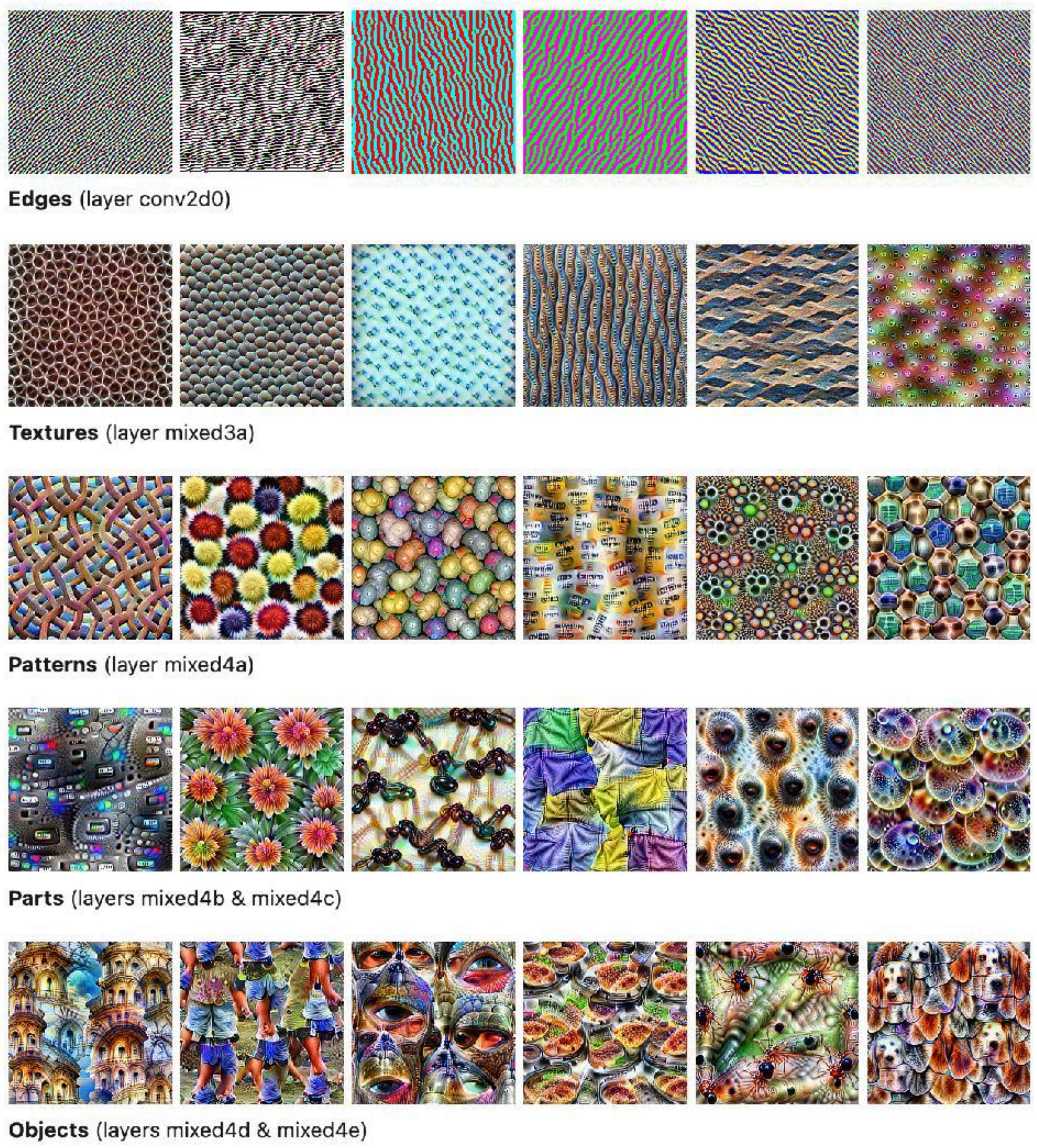
## Learned kernels

Convolutional Neural Network (CNN)




With CNNs, the kernel values are learned as model parameters

# Learned Filters (a.k.a. Convolution Kernels)

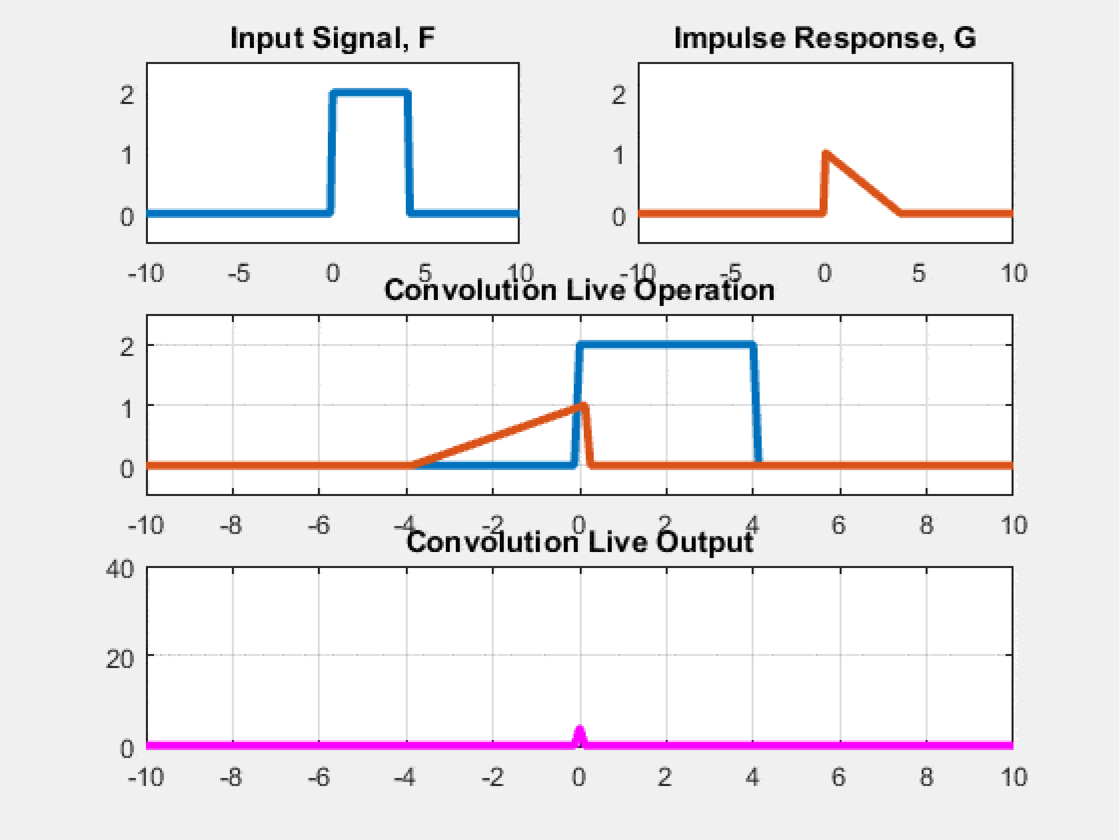



https://distill.pub/2017/feature-visualization/

# Convolution in Digital Signal Processing

卷积的运算过程：翻转 —> 平移 —> 相乘 —> 求和

$$y[n] = (\boldsymbol{x} * \boldsymbol{h})[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

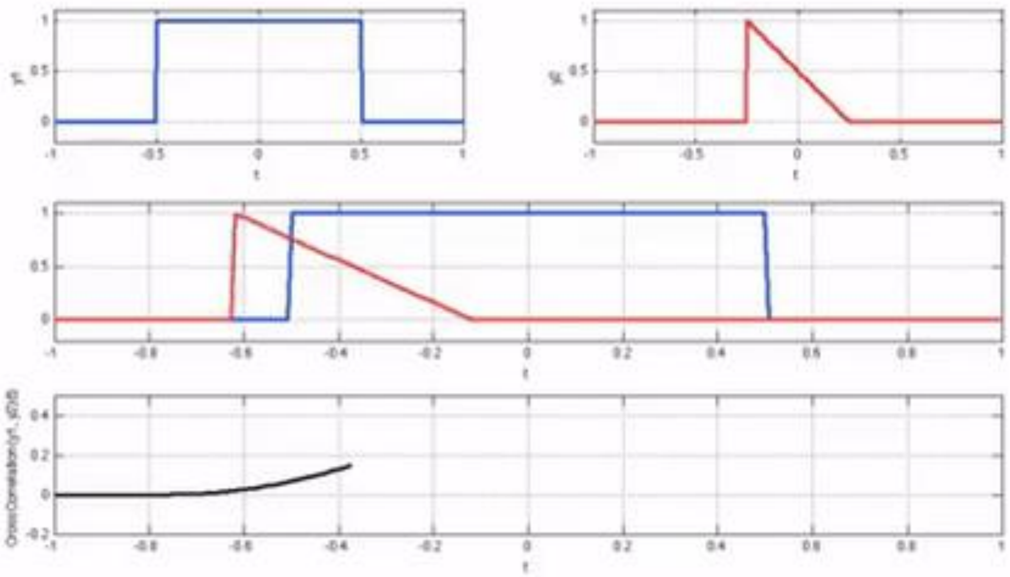



https://quincyaflint.weebly.com/academic-material/discrete-convolution

# Cross-Correlation in Digital Signal Processing

互相关的运算过程：共轭 —> 平移 —> 相乘 —> 求和

$$r_{xy}[n] = (\boldsymbol{x} \star \boldsymbol{y})\,[n]$$
$$= \sum_{k=-\infty}^{\infty} x[k]y^*[k-n]$$

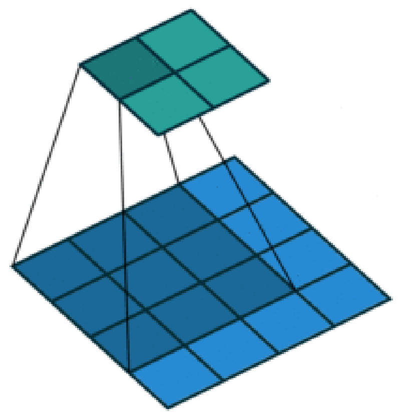


Convolution Neural Networks are indeed

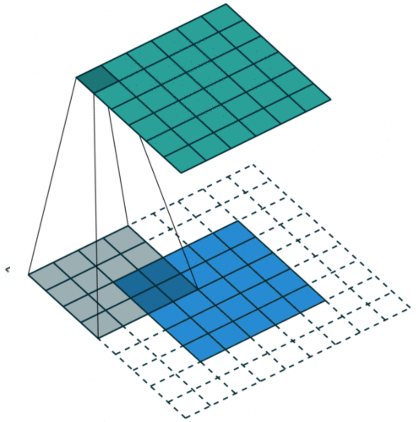
Correlation Neural Networks

https://gfycat.com/brownquickisabellineshrike

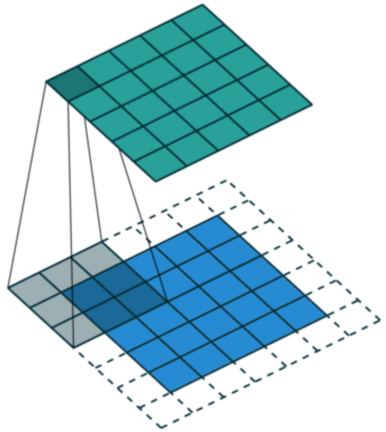
# Paddings, Strides



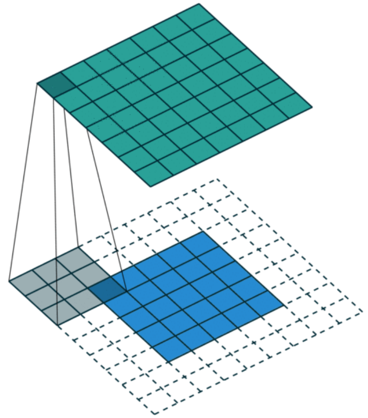
No padding, no strides



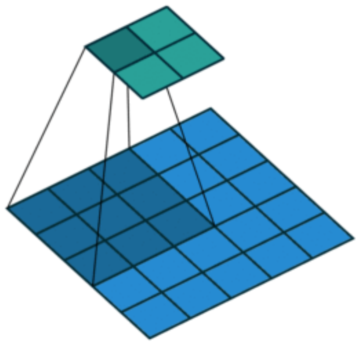
Arbitrary padding, no strides



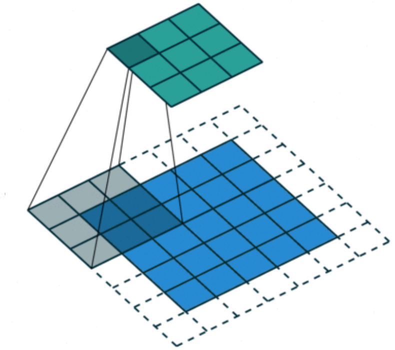
Half padding, no strides



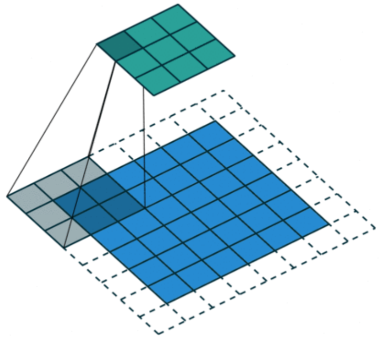
Full padding, no strides



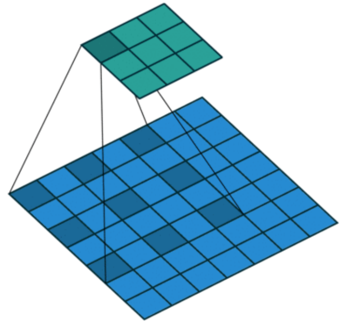
No padding, strides



Padding, strides



Padding, strides (odd)



No padding, no stride, dilation

https://browse.arxiv.org/pdf/1603.07285.pdf

# Paddings, Strides

- 2-D discrete convolutions ($N = 2$),
- square inputs ($i_1 = i_2 = i$),
- square kernel size ($k_1 = k_2 = k$),
- same strides along both axes ($s_1 = s_2 = s$),
- same zero padding along both axes ($p_1 = p_2 = p$).

**Relationship 6.** *For any $i$, $k$, $p$ and $s$,*

$$o = \left\lfloor \frac{i + 2p - k}{s} \right\rfloor + 1.$$

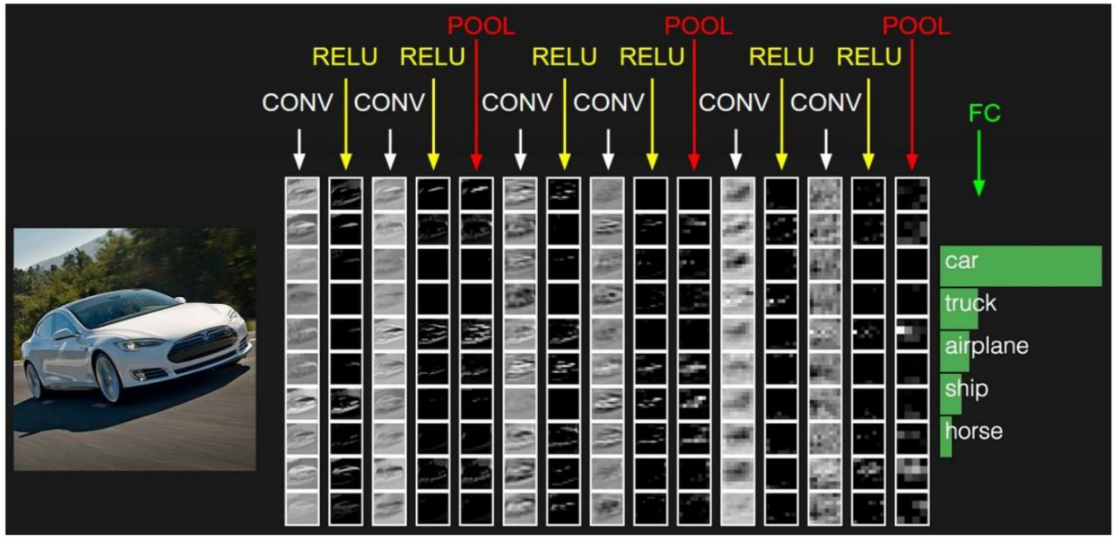
https://browse.arxiv.org/pdf/1603.07285.pdf

# The entire architectures

Repeat several times:

- Start with a convolutional layer
- Followed by non-linear activation and pooling

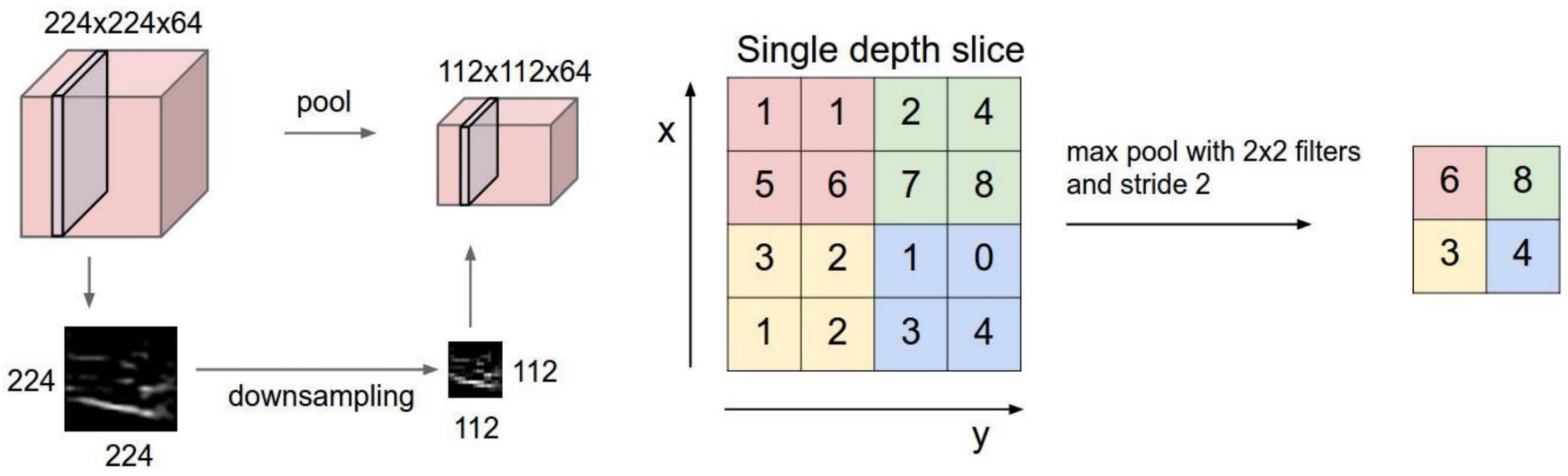
End with a fully connected (MLP) layer



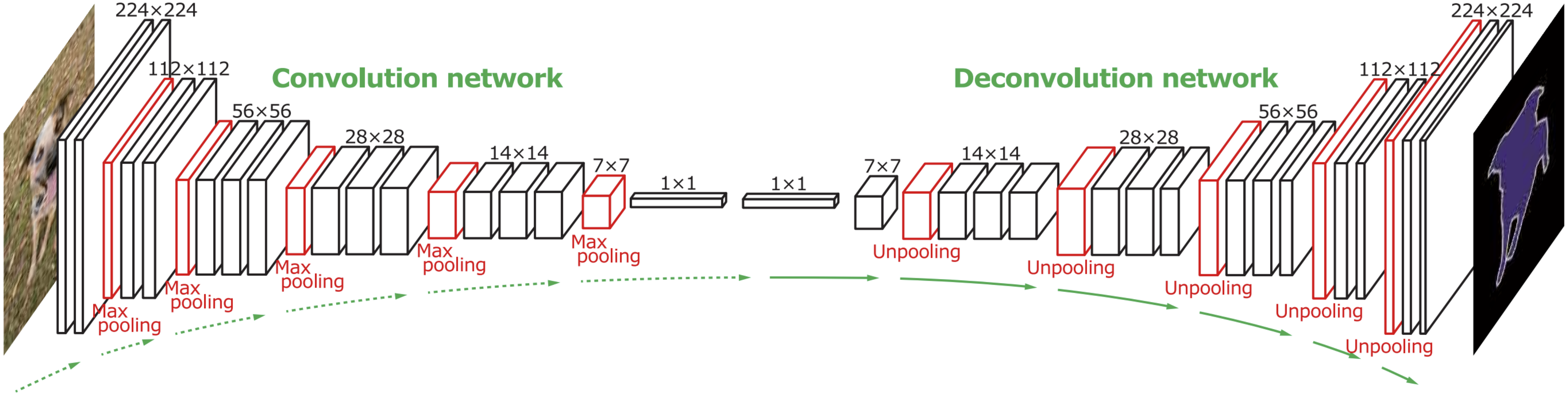
# Pooling Layer

Response map subsampling:

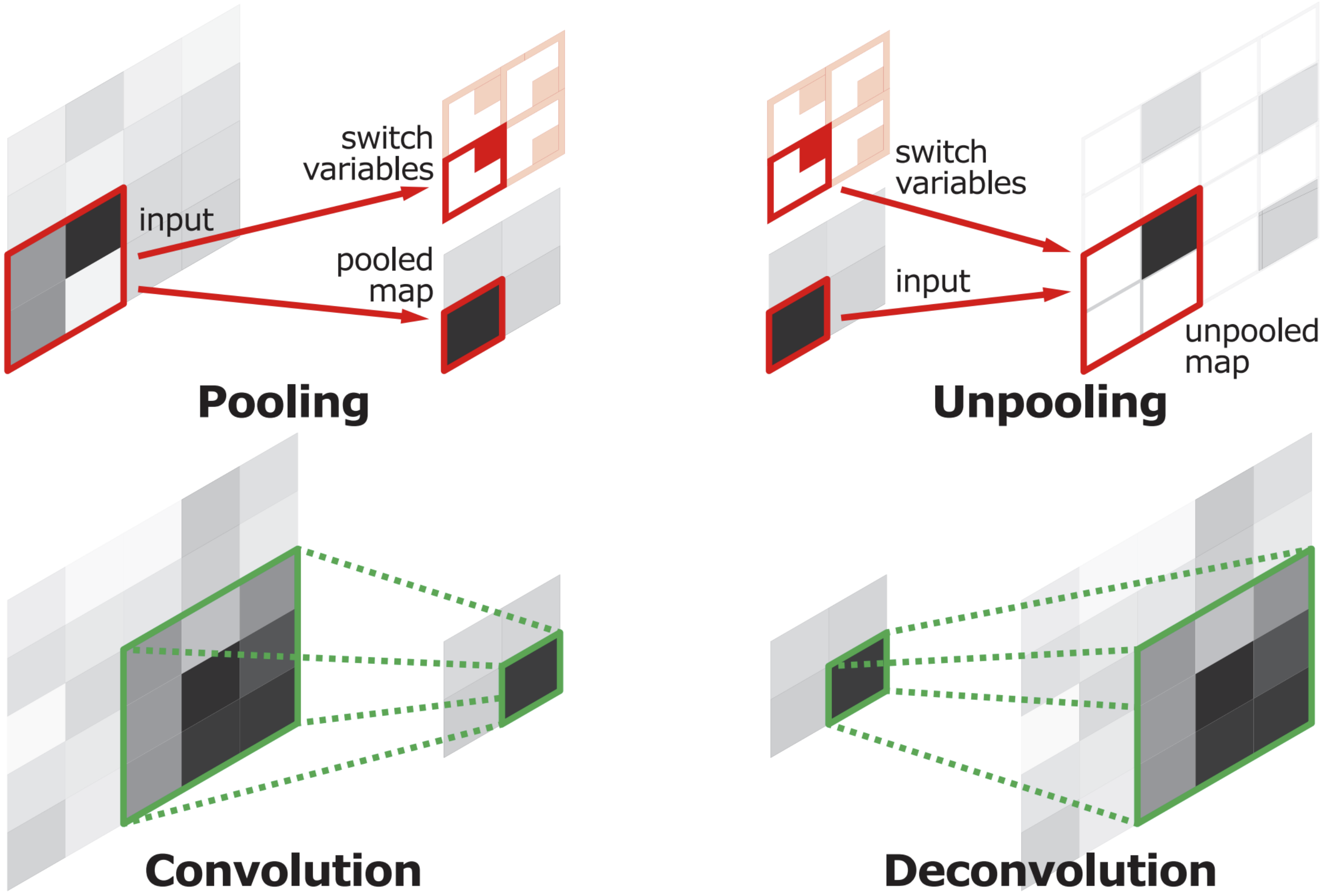
Allows summarization of the responses



# Pixel-wise tasks

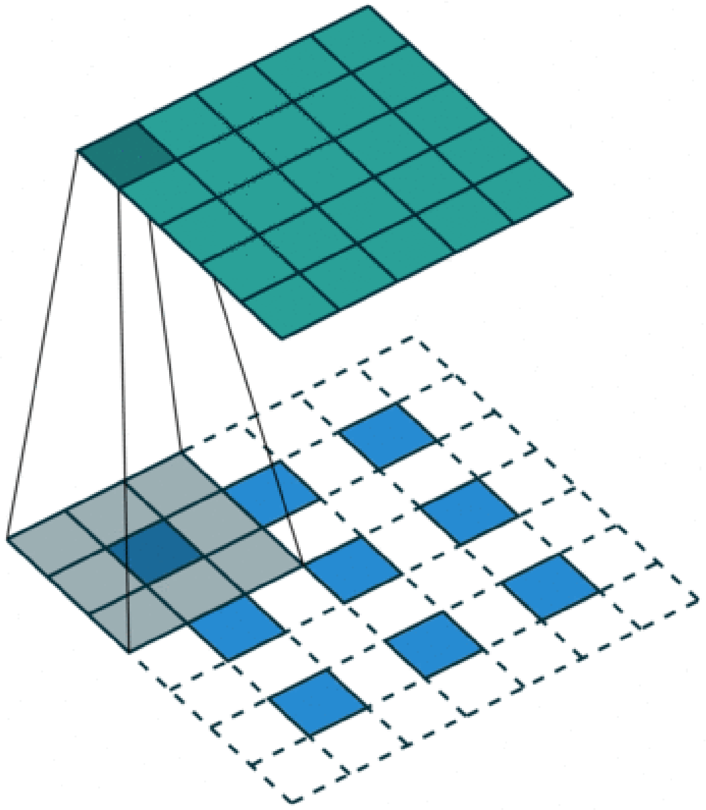


# Unpooling



switch
variables
input
pooled
map
Pooling
switch
variables
input
unpooled
map
Unpooling
Convolution
Deconvolution

# Transposed convolution (fractionally-strided convolution, deconvolution)

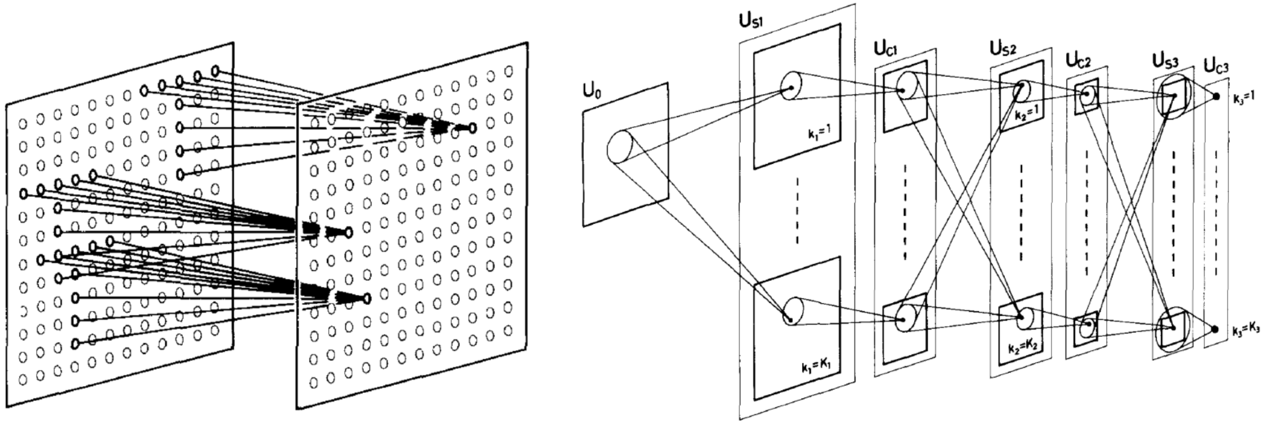


The transpose of convolving a 3×3 kernel over a 5×5 input padded with a 1 × 1 border of zeros using 2 × 2 strides (i.e., $i$= 5, $k$ = 3, $s$ = 2 and $p$ = 1). It is equivalent to convolving a 3 × 3 kernel over a 3 × 3 input (with 1 zero inserted between inputs) padded with a 1 × 1 border of zeros using unit strides (i.e., $i'$ = 5, $k'$ = $k$, $s'$= 1 and $p'$ = 1).

https://browse.arxiv.org/pdf/1603.07285.pdf

# Common architectures

*Neocognitron*




Neocognitron, an early geometric neural network

**K. Fukushima**

1980

Fukushima 1980

# Common architectures

## *Neocognitron*

- Deep neural network (7 layers tested)
- Local connectivity (“receptive fields”)
- Nonlinear filters with shared weights (S-layers)
- Average pooling (C-layers)
- ReLU activation function
- ”Self-organised” (unsupervised) – **no backprop yet!**

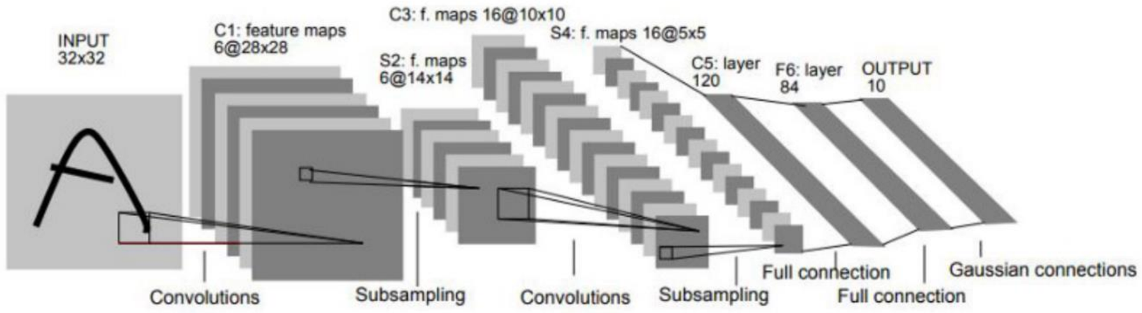
K. Fukushima

1980

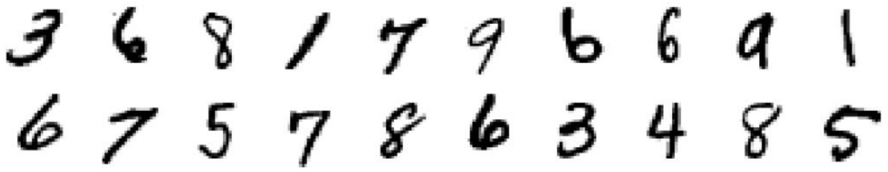
Fukushima 1980 https://www.rctn.org/bruno/public/papers/Fukushima1980.pdf

# Common architectures

*LeNet-5*




LeNet-5 classical CNN architecture



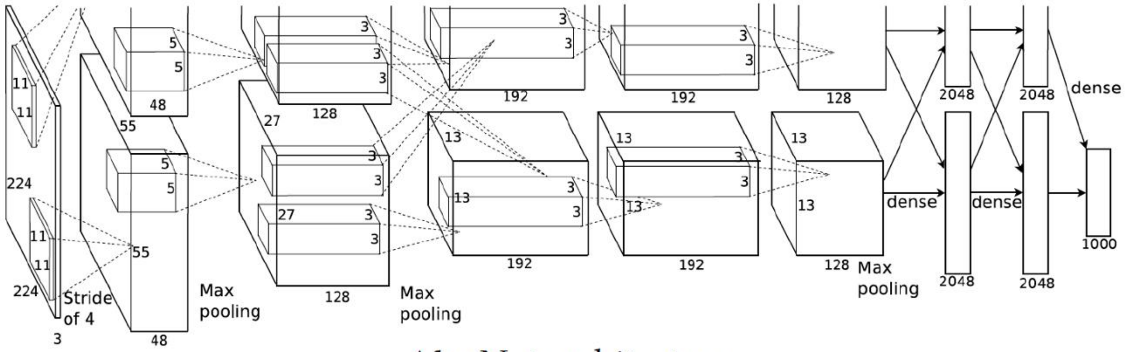
MNIST digits dataset

**Y. LeCun**

LeCun et al. 1989

# Common architectures

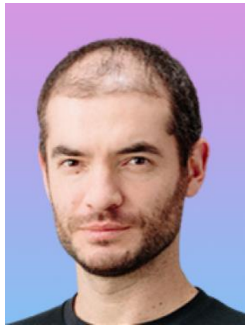
## *AlexNet*



AlexNet architecture


Nvdia GTX 580 GPU capable of ~200G FLOP/sec


**Alex Krizhevsky**


**Ilya Sutskever**

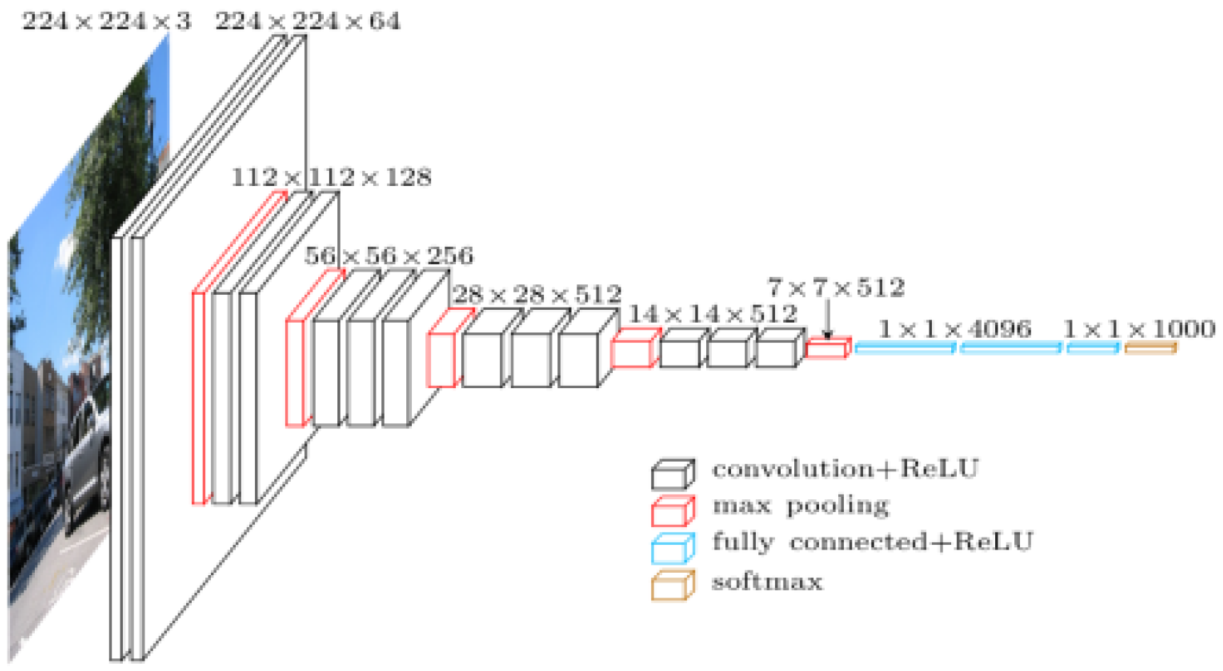

**Geoffrey E. Hinton**

Krizhevsky et al. 2012

# VGGNet

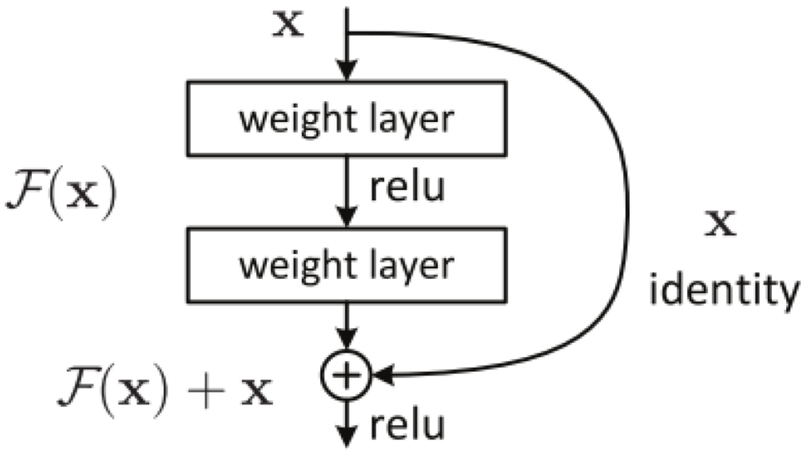
Used for object classification task

- 1000-way classification task
- 138 million parameters



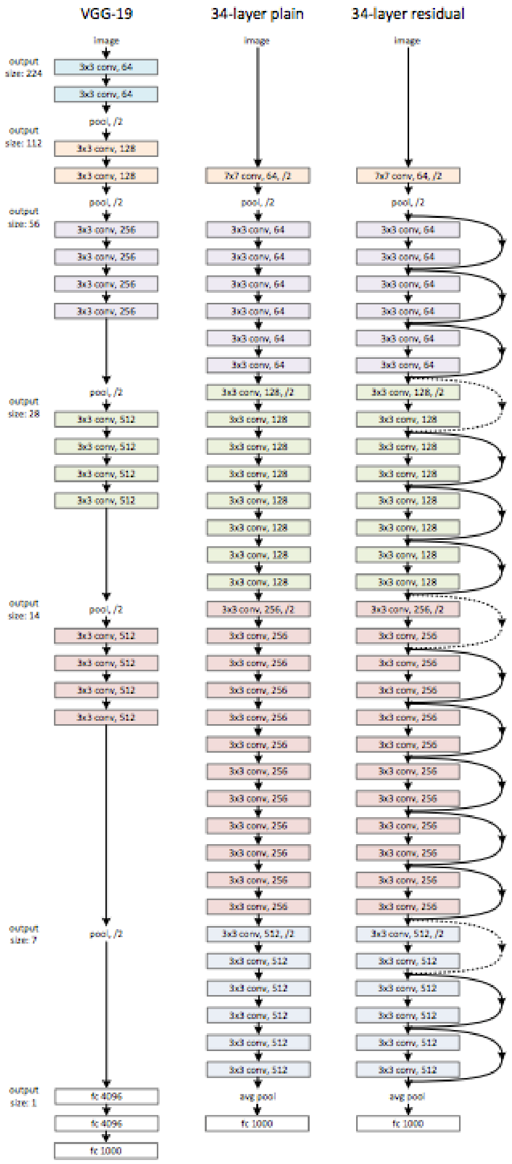
# Residual Networks (ResNet)

## Adding residual connections




ResNet (He et al., 2015)

- Up to 152 layers!



# 内容提纲

① 图片表示

② 卷积神经网络

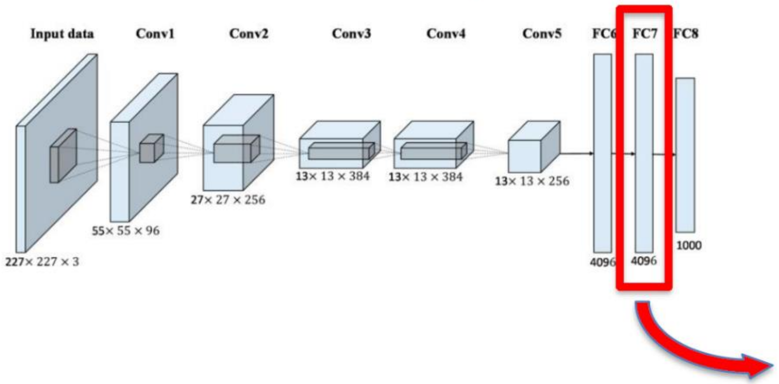
③ 卷积神经网络的可视化

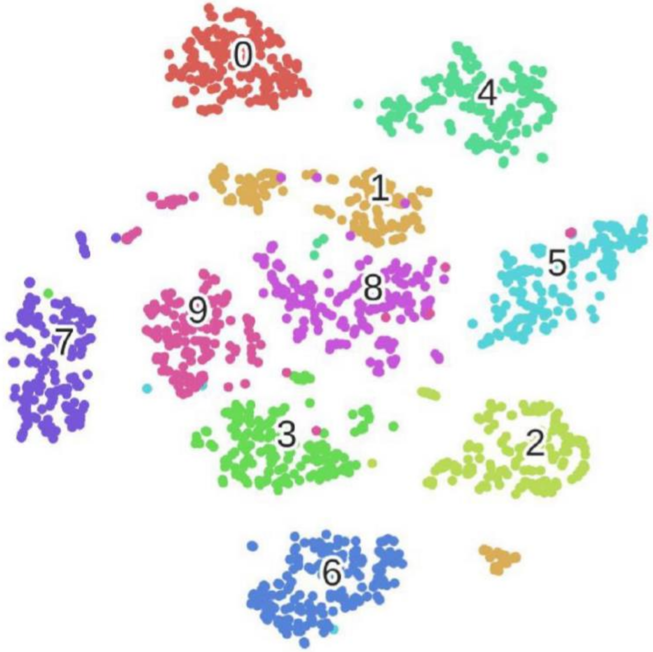
④ 3D卷积神经网络

# Visualizing the Last CNN Layer

PCA, tsne, etc

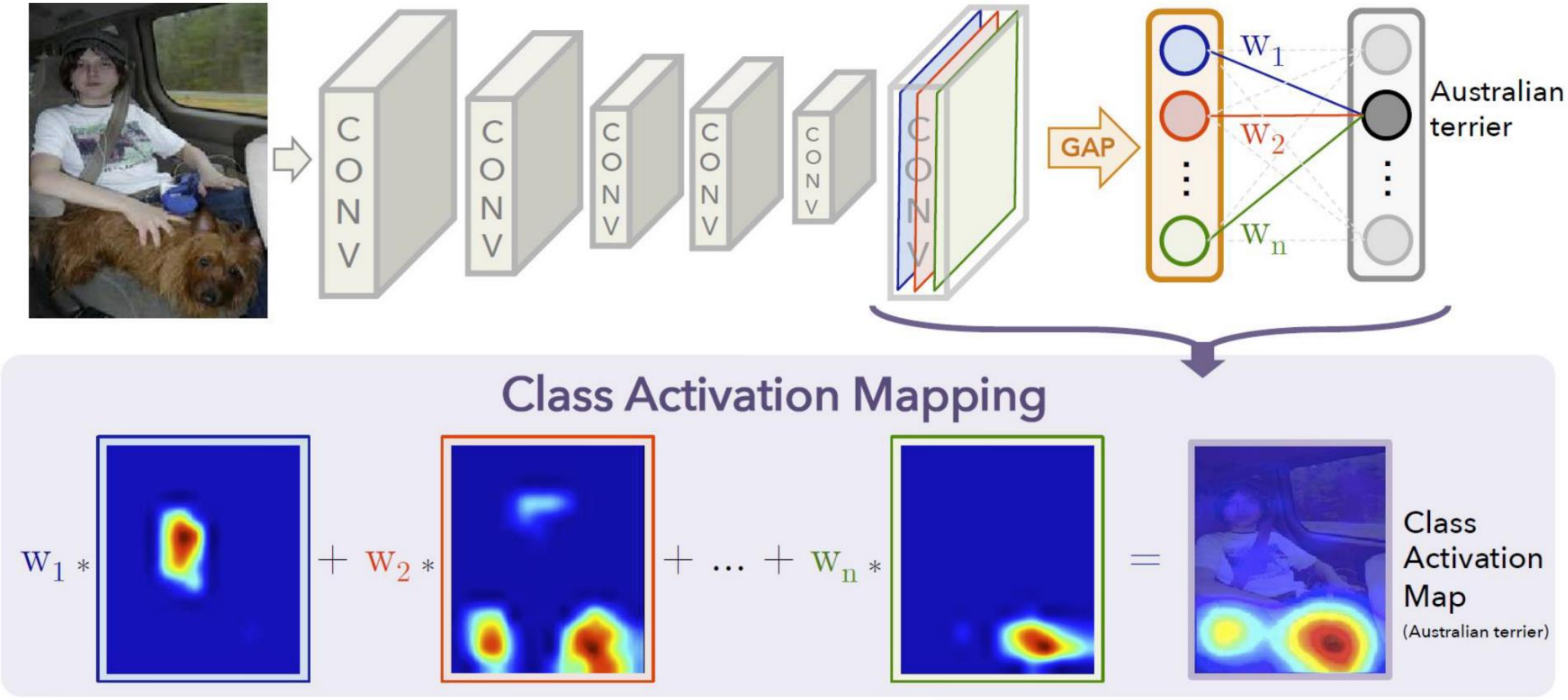
Alex Net




Embed high dimensional data points (i.e. feature codes) so that pairwise distances are conserved in local neighborhoods.

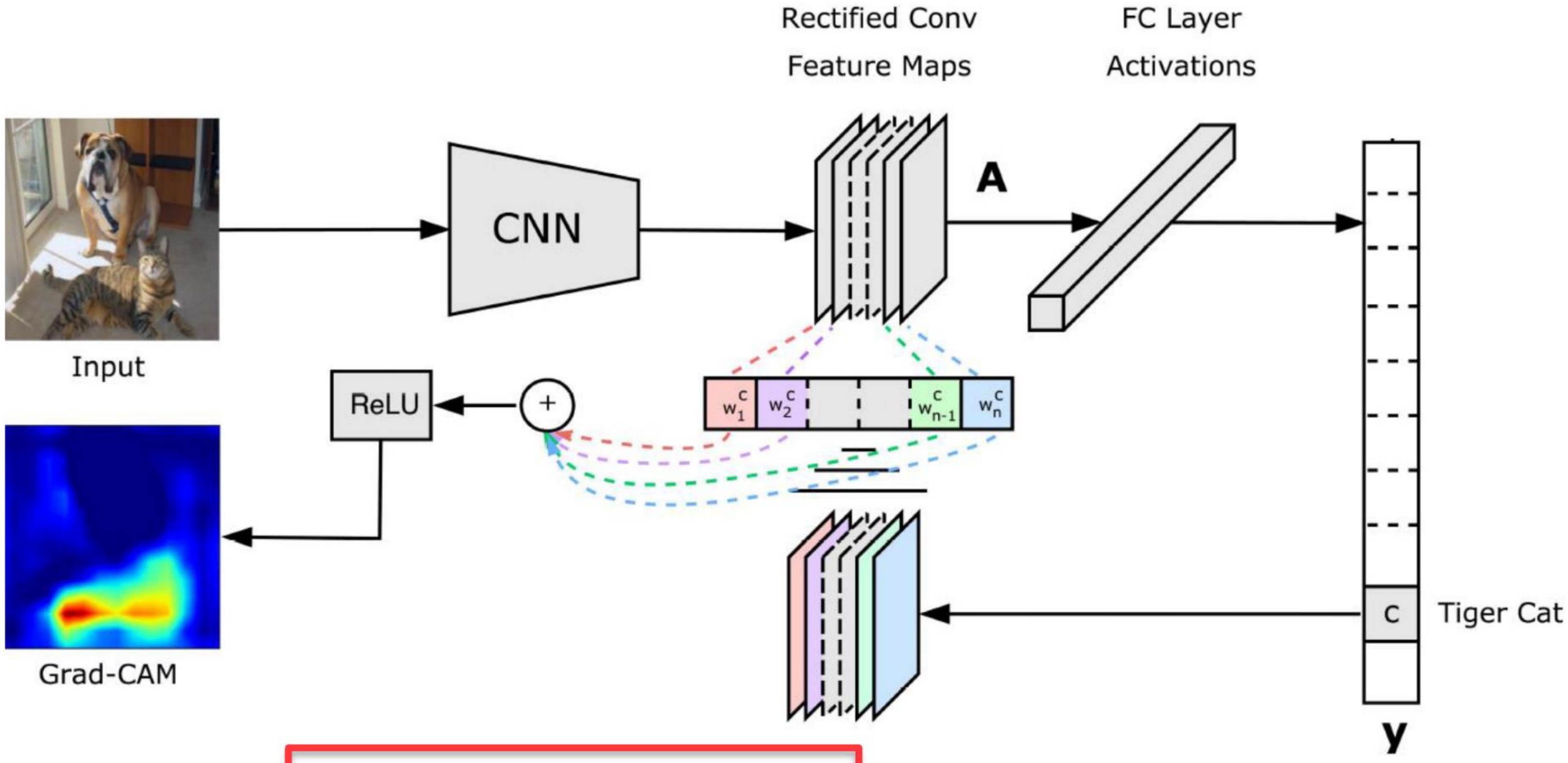
# CAM: Class Activation Mapping [CVPR 2016]




$$L^c_{\mathrm{CAM}} = \underbrace{\sum_k w^c_k A^k}_{\text{linear combination}}$$

http://cnnlocalization.csail.mit.edu/Zhou_Learning_Deep_Features_CVPR_2016_paper.pdf

# Grad-CAM [ICCV 2017]




$$\alpha_k^c = \overbrace{\frac{1}{Z}\sum_i \sum_j}^{\text{global average pooling}} \underbrace{\frac{\partial y^c}{\partial A_{ij}^k}}_{\text{gradients via backprop}}$$

$$L_{\text{Grad-CAM}}^c = ReLU \underbrace{\left(\sum_k \alpha_k^c A^k\right)}_{\text{linear combination}}$$

https://browse.arxiv.org/pdf/1610.02391.pdf

# 内容提纲

① 图片表示

② 卷积神经网络

③ 卷积神经网络的可视化

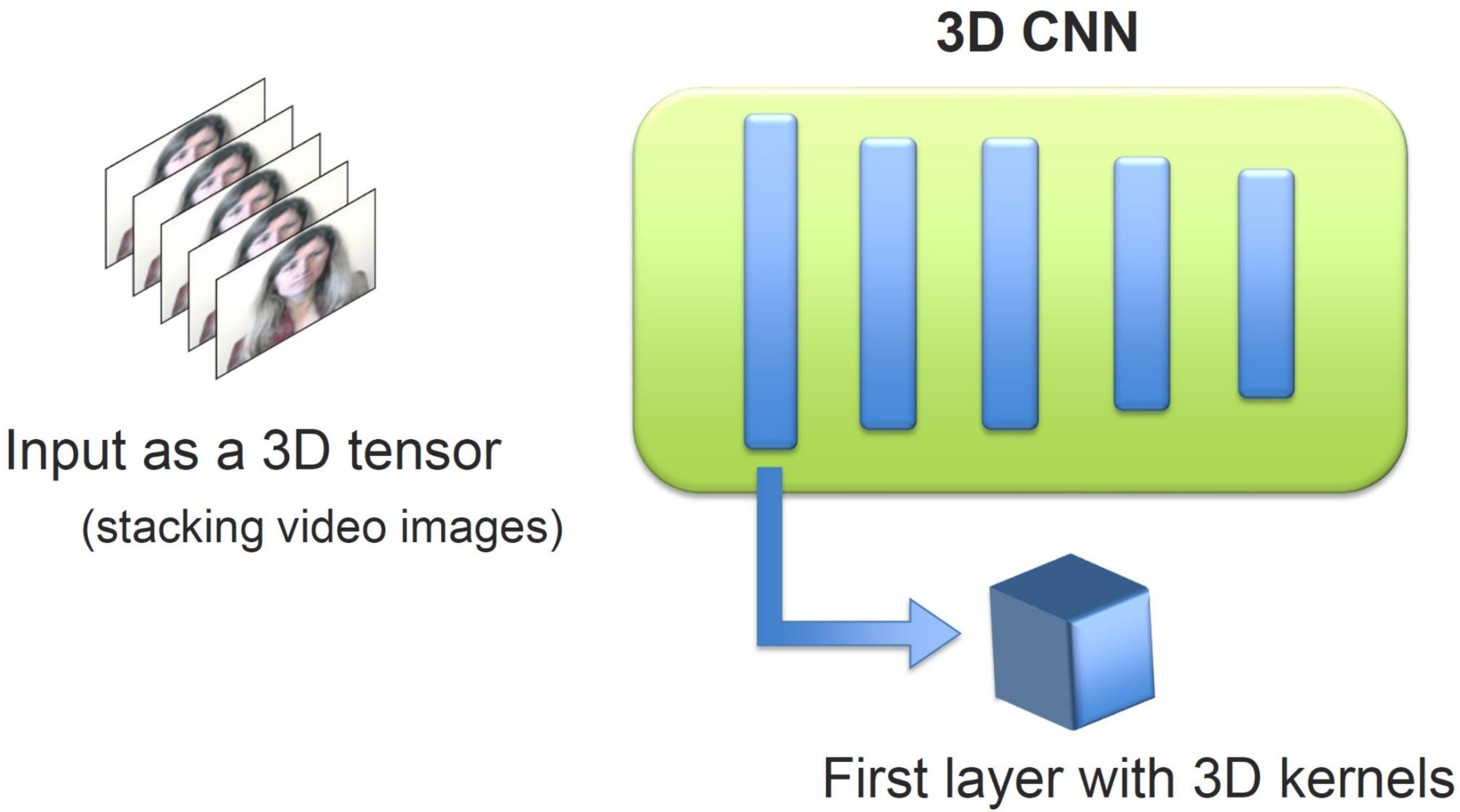
④ 3D卷积神经网络

# Modeling Temporal and Sequential Data

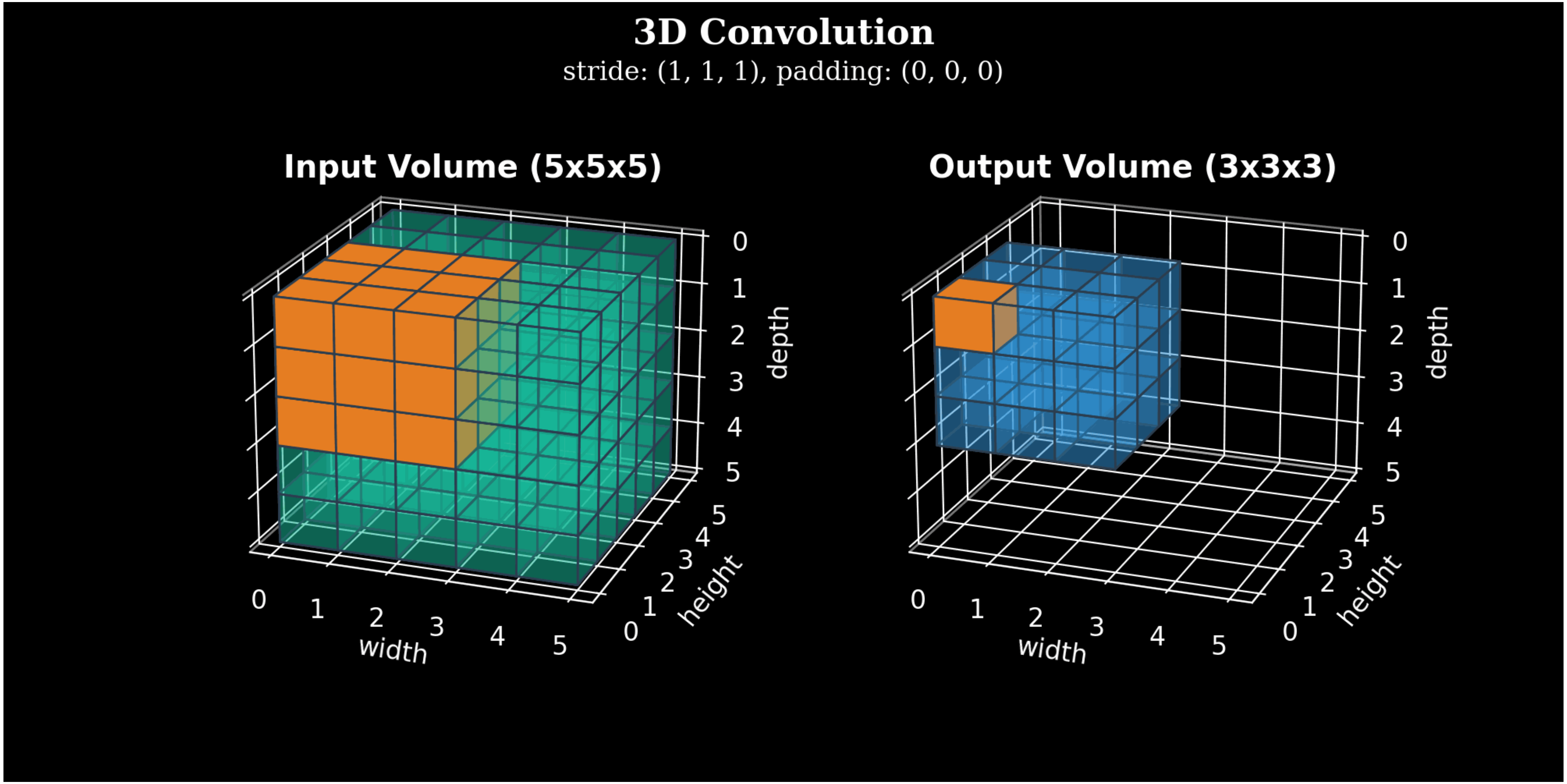
How to represent a video sequence?

One option: Recurrent Neural Networks

# 3D CNN


3D CNN
Input as a 3D tensor
(stacking video images)
First layer with 3D kernels

# 3D CNN




https://www.kaggle.com/code/mariuszwisniewski/3d-cnn-2-for-alzheimer-disease-poc

# 总结

- 了解计算机视觉领域的典型任务
- 掌握卷积神经网络的设计及其特点
- 了解3D卷积神经网络