Applied Deep Learning

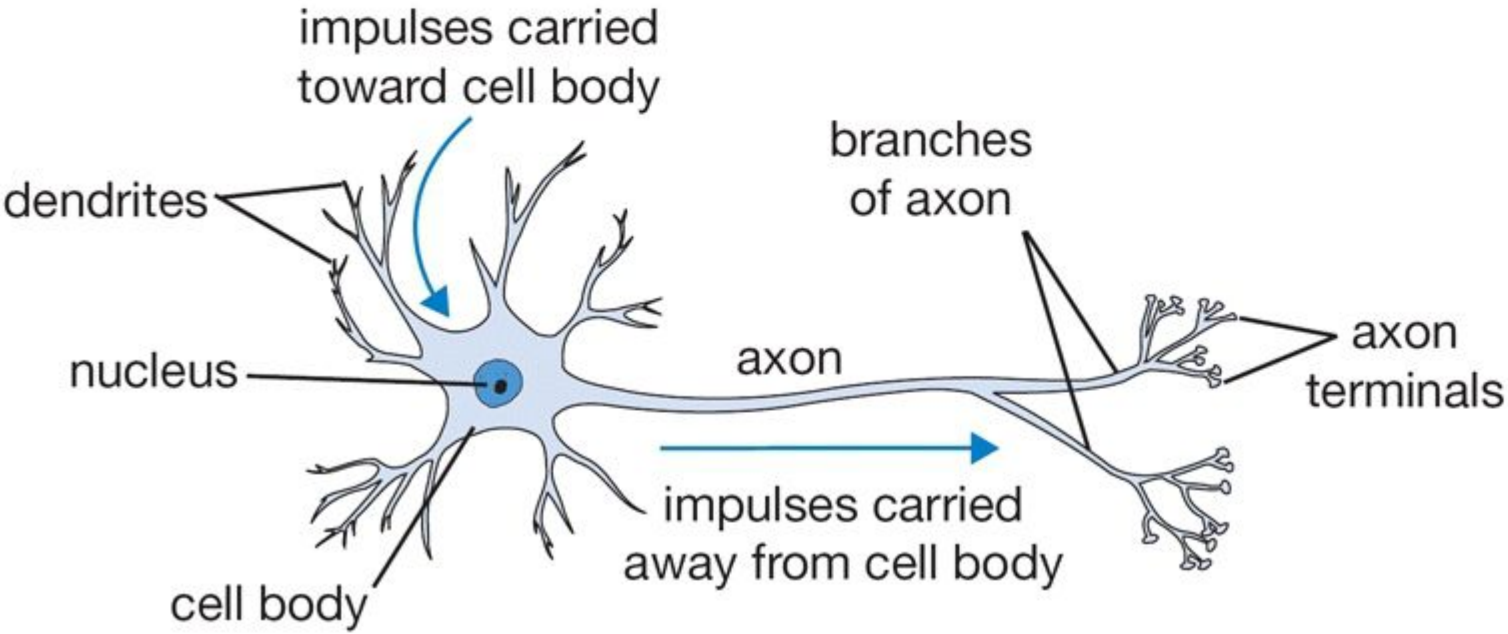
Convolutional Neural Networks and Visual Computing
17.10.2019

Alexander Pacha

Recap

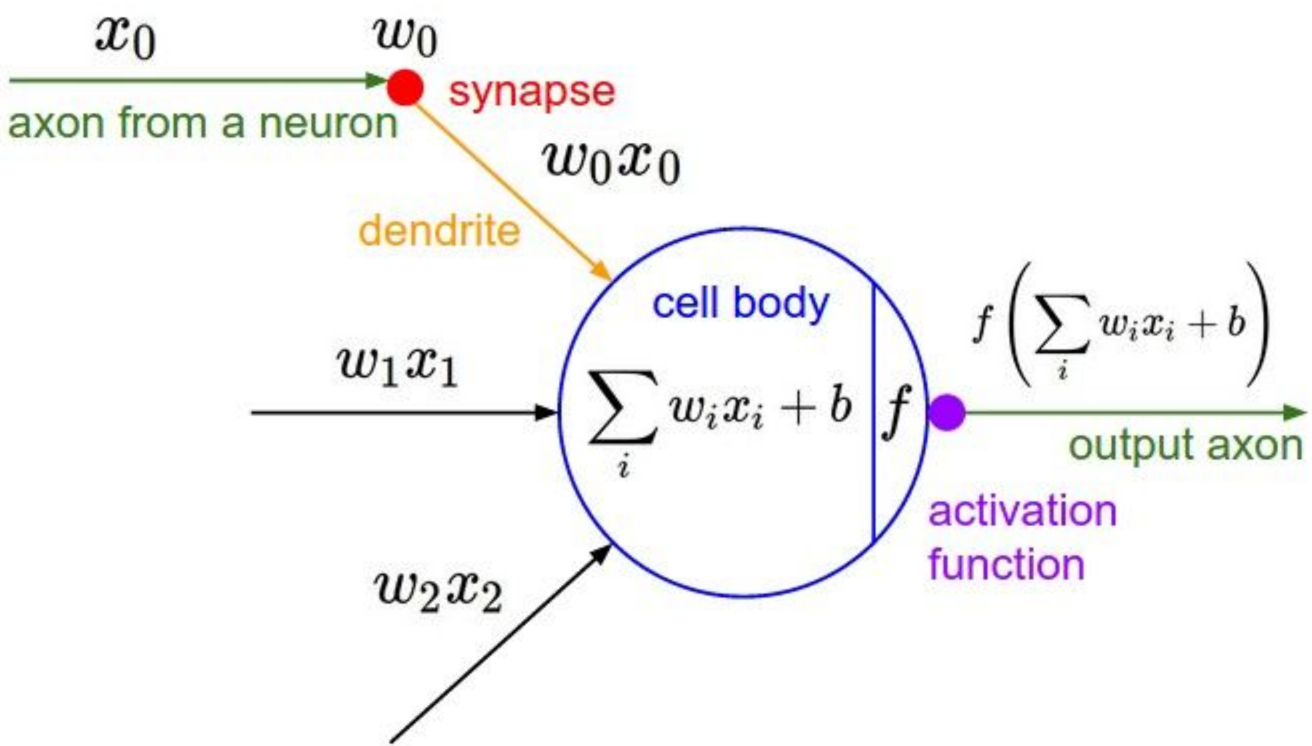
- 3 Parts required for Machine Learning?
- What is a Gradient?
- What is Gradient Descent and what does Backpropagation have to do with it?

Recap - Biological Neuron



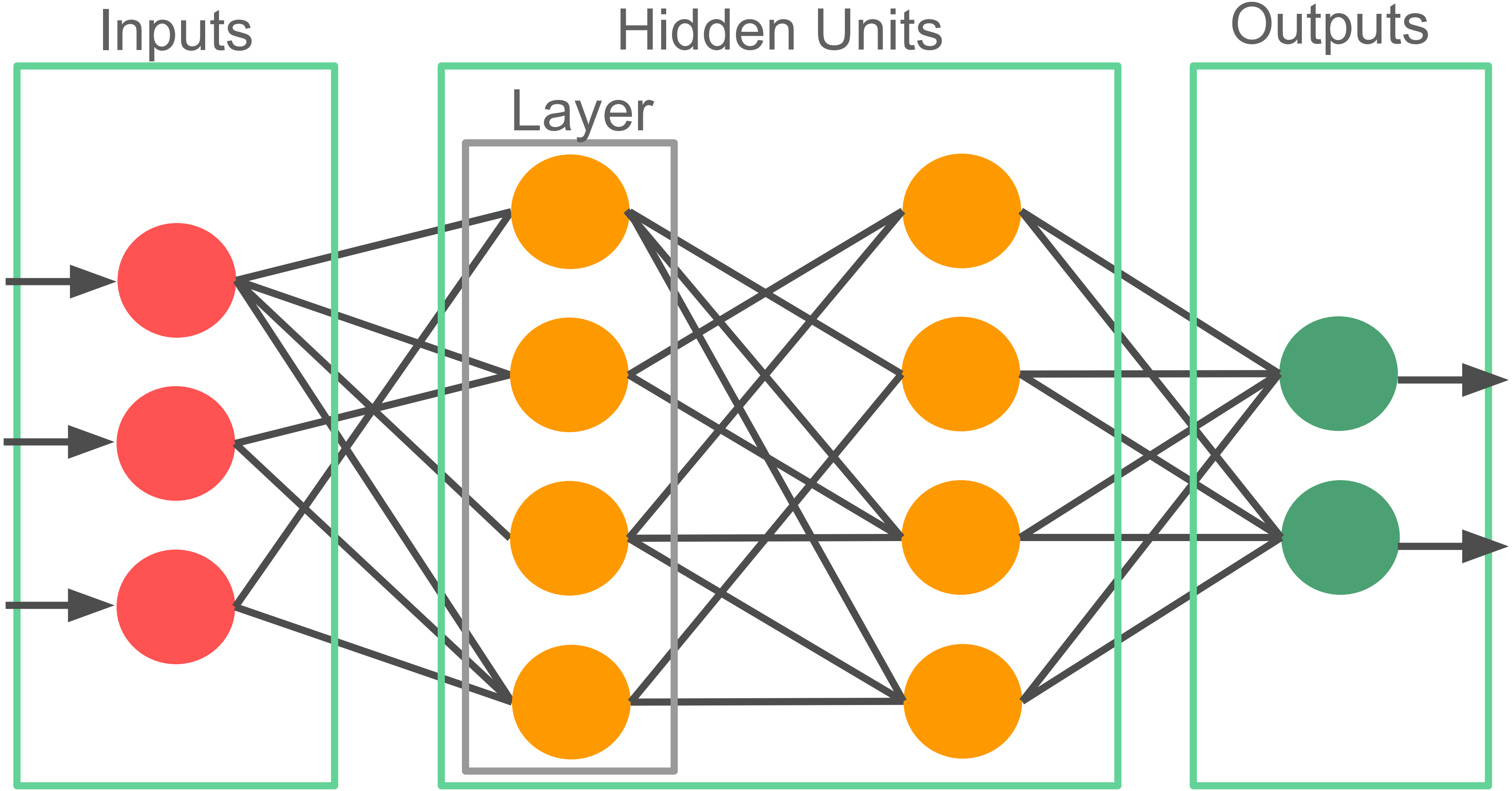
Source: http://cs231n.github.io/neural-networks-1/

Recap - Artificial Neuron



Source: http://cs231n.github.io/neural-networks-1/

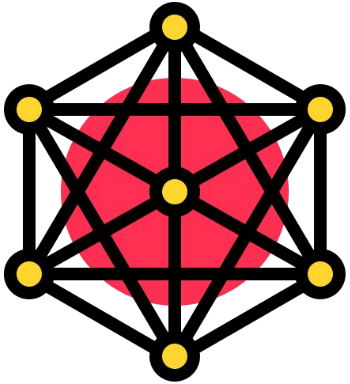
Recap - Neural Network



How to wire the Neural Network?

Multilayer Perceptrons

- One input layer, one output layer, a number of hidden layers
- Neurons in layer `L` are connected to all neurons in layer `L-1`
- Such layers are called fully-connected or dense layers



Can MLPs be used for images?

Starting with simple two-layer MLP:

- One stage learns to extract features
- Final stage trains a linear classifier on these features

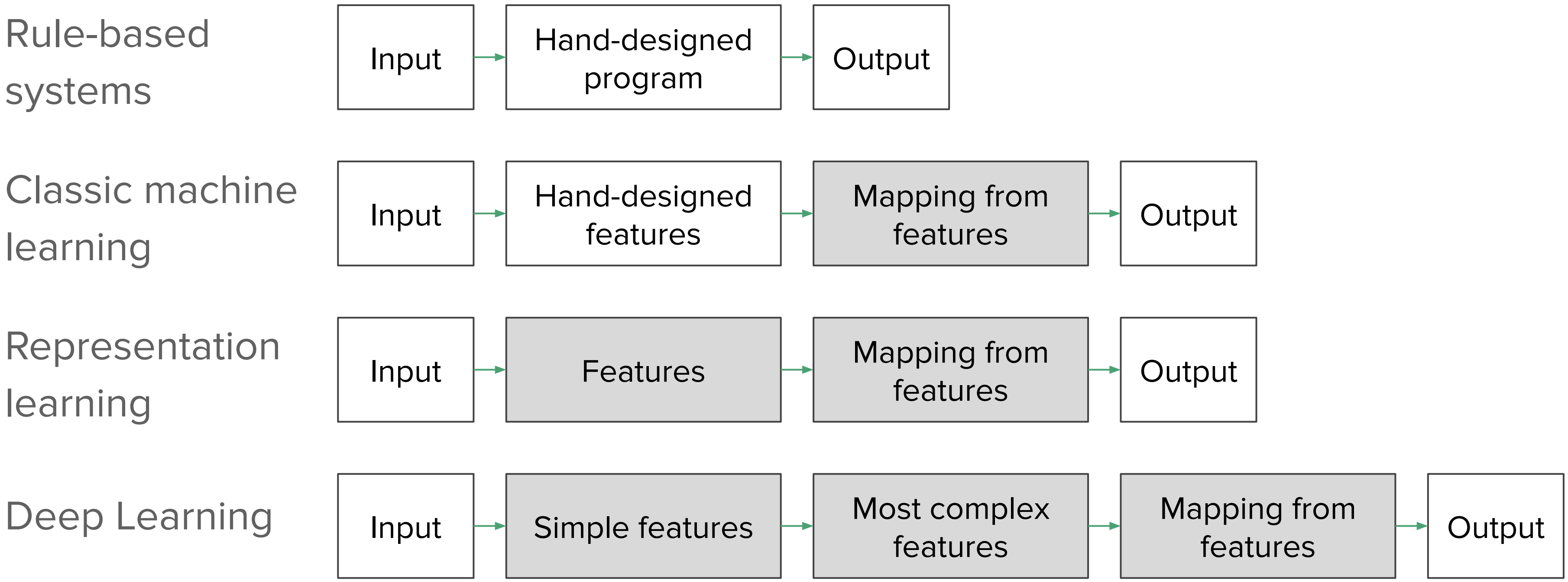
→ Way too hard!

How about adding more layers?

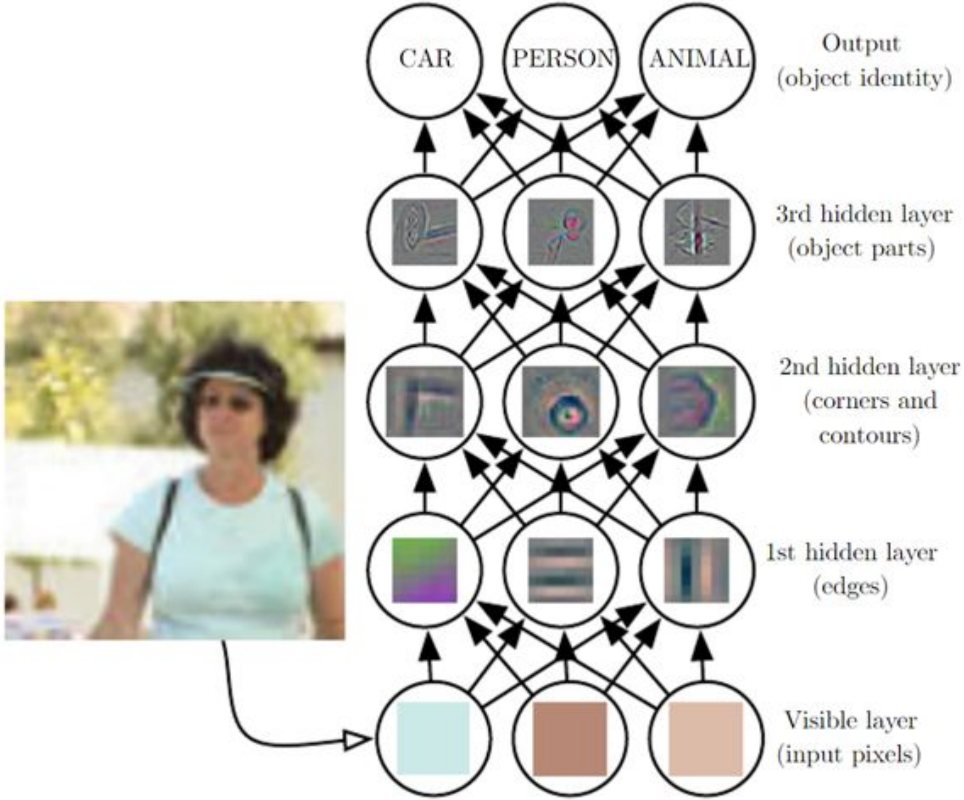
- Number of parameters increase quickly with number of layers and image size:
  - 128x128 RGB image, 500 hidden neurons = 25 million parameters
  - 224x224 RGB image, 500 hidden neurons = 75 million parameters
- Usually no improvements as depth exceeds 3
- No understanding of images

We need
an understanding of images
(a proper representation)

How to get a good representation?



Meaningful Representation



Source: http://www.deeplearningbook.org/

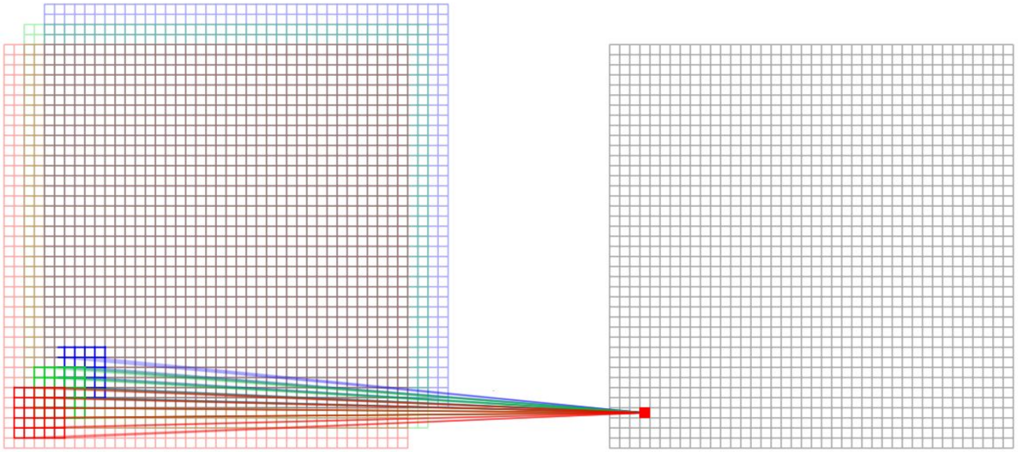
Convolutional Layer

- Using fully connected layers for images not efficient
- Spatially close pixels are highly correlated, others are not

Solution:

- Arrange hidden layer neurons in a grid with only sparse connectivity

Layer l - 1

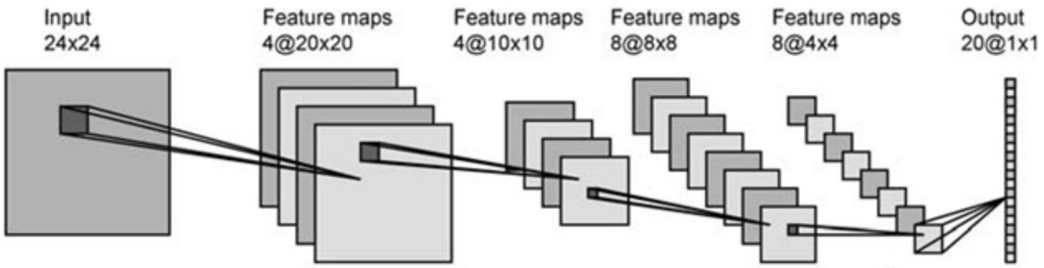


Layer l

Source: https://github.com/cpra/dlvc2016

Convolutional Layer

- Weights are now matrices
- Every neuron learns to extract features in local neighborhood
- Every neuron learns different features
    - Usually a feature is useful anywhere in an image
    - Enforced by parameter sharing between neurons: Same parameters for all neurons in layer
- With parameter sharing, every layer can learn only single feature
    - So we replicate neurons D times (new hyperparameter)
    - Each depth slice is called feature map
    - Only neurons in the same feature map share weights

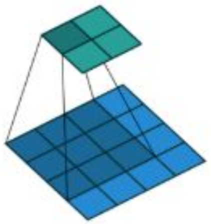
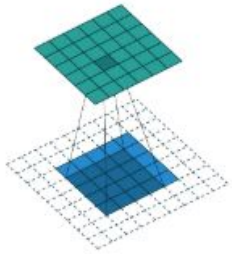
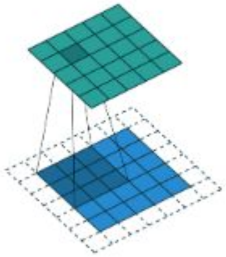
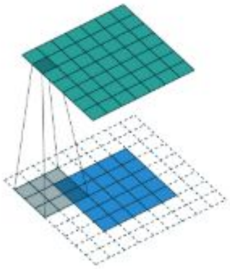
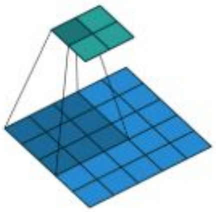
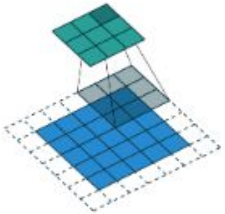
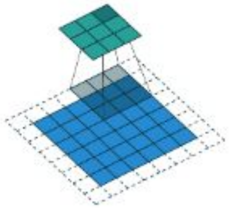


Convolutions in Code

https://colab.research.google.com/drive/1tPUopl0KUsd12ikGKzIOeAgd46r-uTkD

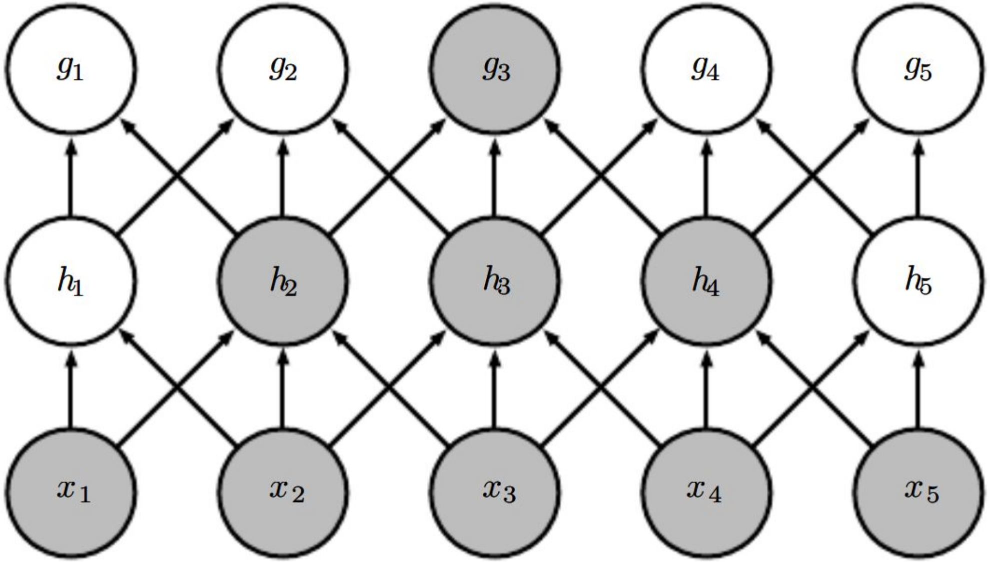
Convolutional Parameters

https://github.com/vdumoulin/conv_arithmetic

			
No padding, no strides	Arbitrary padding, no strides	Half padding, no strides	Full padding, no strides
			
No padding, strides	Padding, strides	Padding, strides (odd)	

Receptive field

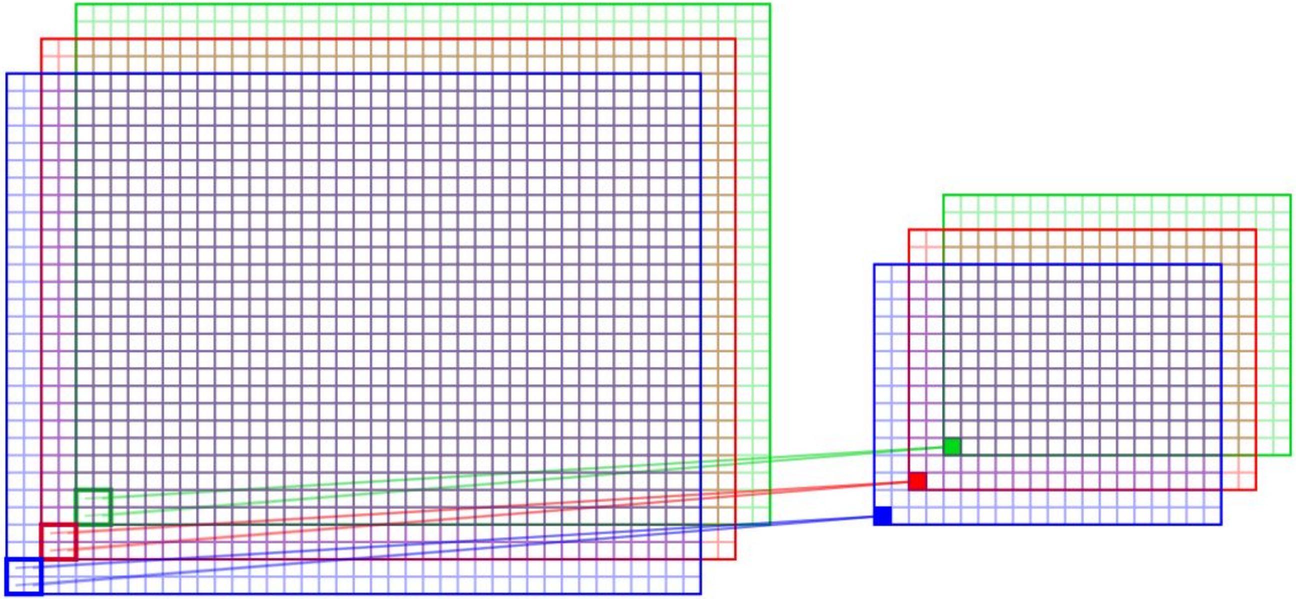
- Although neurons are only connected to a local neighborhood, higher layers can “see” the entire image.
- Early layers learn more local features, later layers learn more global features



Source: http://www.deeplearningbook.org/

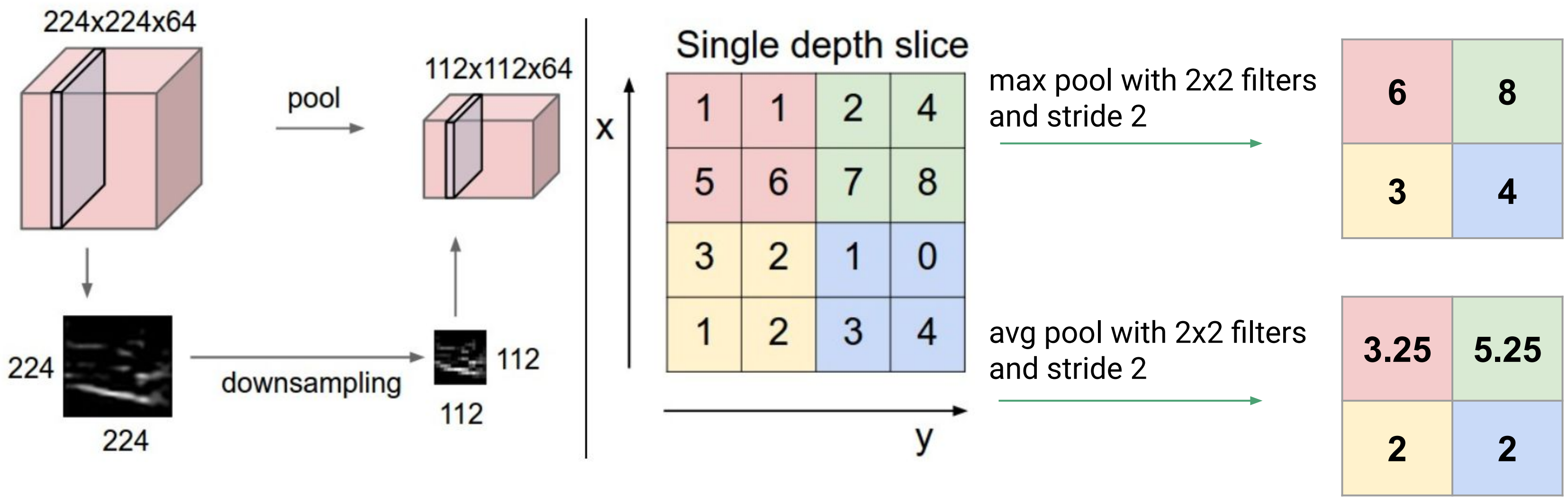
Pooling

- Reduction of spatial resolution (but not depth) of input
- Goal: Reduce number of parameters and computations



Source: https://github.com/cpra/dlvc2016

Max-Pooling, Avg-Pooling

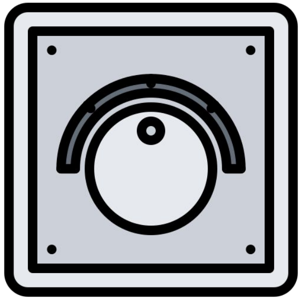


Sidenote: Convolutions with strides can be an alternative to Pooling Layers

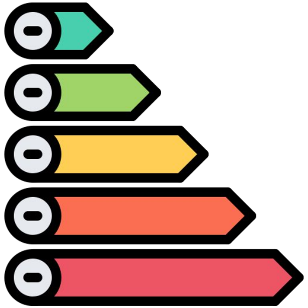
Source: http://cs231n.github.io

Activation Functions for Output Units

Identity for regression



Softmax for classification



Softmax

Cross-entropy is dissimilarity between two probability distributions p and q

$H(p,q) = -\Sigma_x (p(x) \log q(x))$ with
$p \sim \mathbf{w}_s$, encoding the true class distribution
$q \sim \mathbf{w}$, encoding the predicted class distribution

Using one-hot encoding to obtain $\mathbf{w}_s$ from single label w_s

Number of classes $T = 4$

$$w_s = 3 \longrightarrow \mathbf{w}_s = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \qquad w_s = 1 \longrightarrow \mathbf{w}_s = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Softmax

Predictions w are not valid probability distributions

- Values are unbounded
- Sum of all outputs generally does not sum to 1

Solution:

- Have T output neurons
- Regard their output **w** as unnormalized log probabilities
- Use softmax function for normalization

$$\mathrm{softmax}_k(\mathbf{w}) = \frac{\exp(w_k)}{\Sigma_{t=1}^{T}\exp(w_t)} \qquad \mathrm{softmax}\left(\begin{pmatrix} -1 \\ 3 \\ 1 \end{pmatrix}\right) \approx \left(\begin{pmatrix} 0.016 \\ 0.867 \\ 0.117 \end{pmatrix}\right)$$

Network Architectures

Network Architectures

Main layer types:

- Convolutional (conv)
- Pooling (pool)
- Fully-Connected (fc)

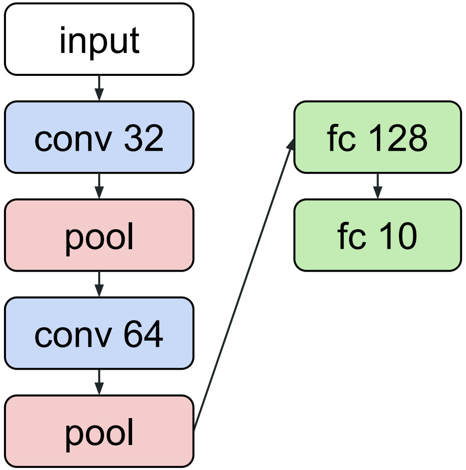
Body / Frontend: conv + pool for feature extraction

Head / Backend: fc for classification/regression

Simple Architecture

Simple MNIST Handwritten Digits Classification:

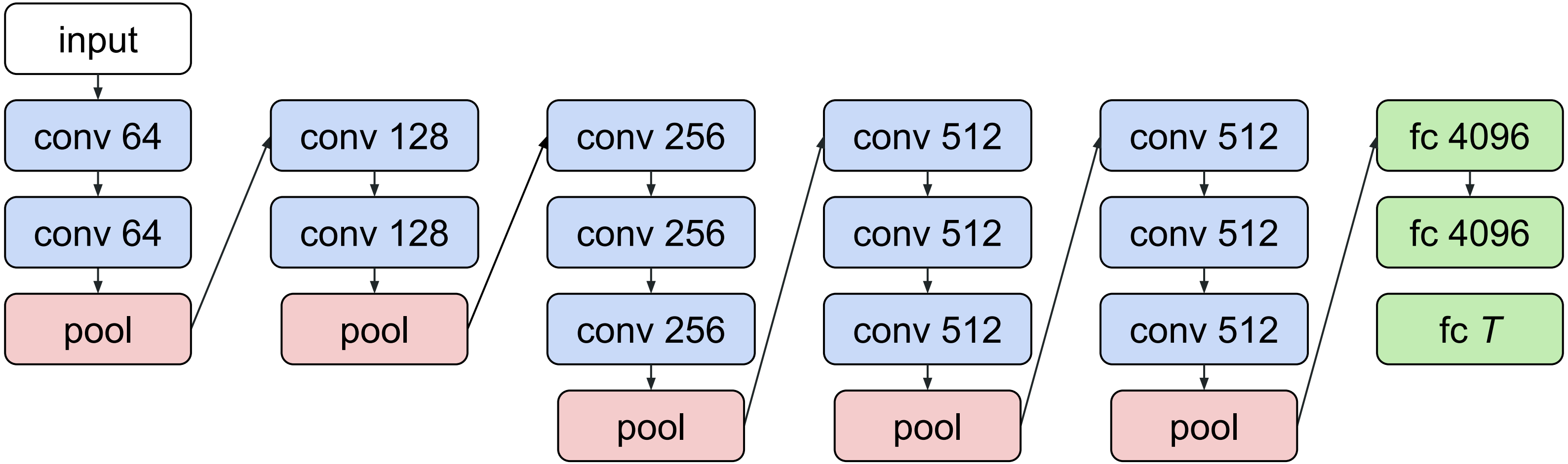
https://colab.research.google.com/drive/1tPUopl0KUsd12ikGKzIOeAgd46r-uTkD



VGGNet

Homogeneous architecture

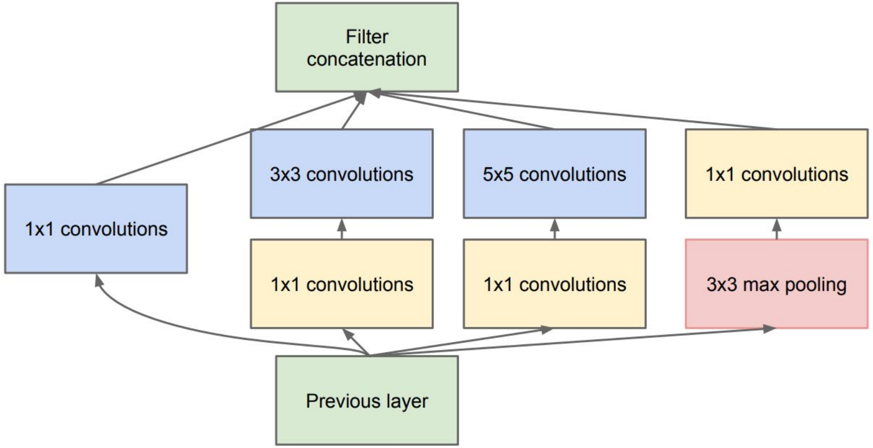
- 3x3 convolutions
- 2x2 max-pooling between blocks
- Depth usually doubled after pooling



Alternative Building Blocks

Inception Module

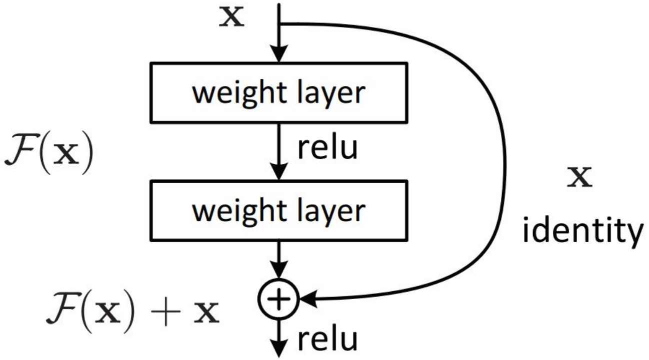
- Learns feature at multiple scales



Source [2]

Residual Module

- Shortcut (skip) connection ease learning by providing guidance



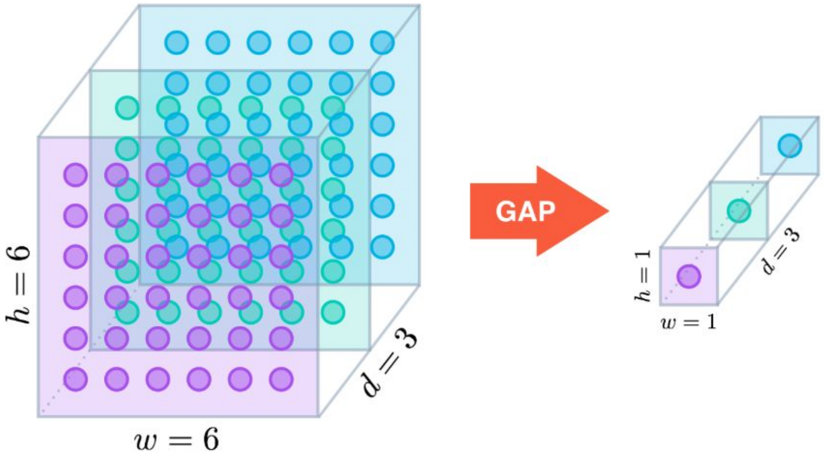
Source [4]

Flattening and Global Average Pooling

How to get from 2D features to 1D output, e.g., for classification?

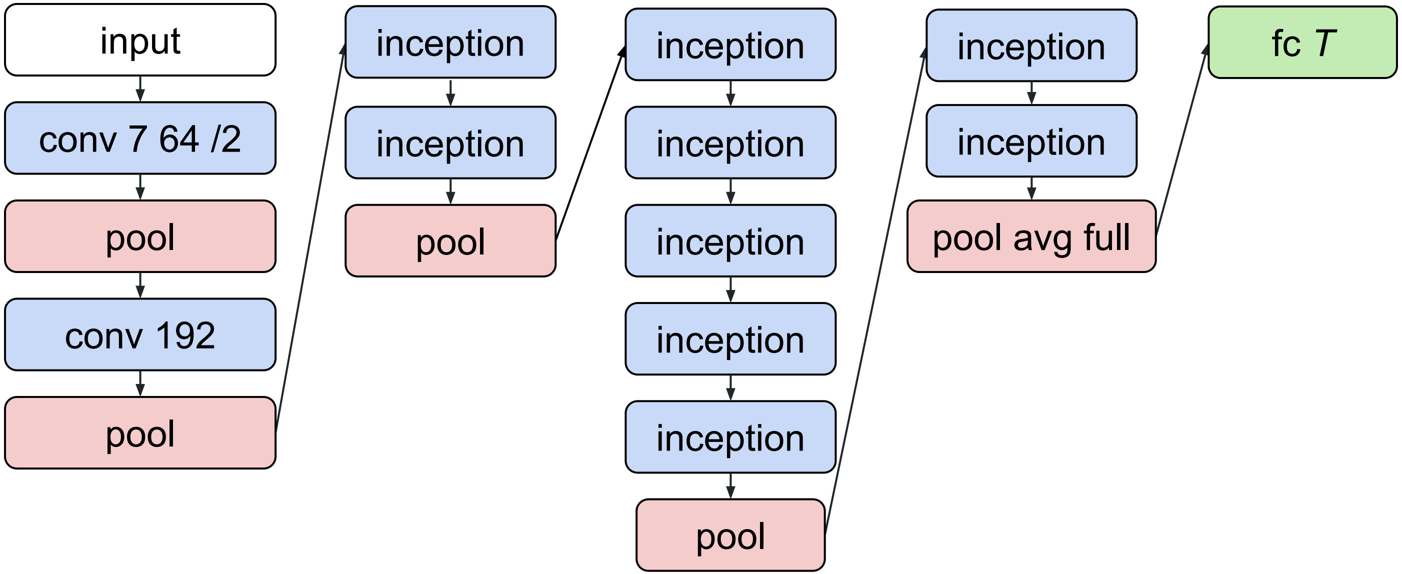
- Flatten
  - Simple and effective way
  - Drops spatial relationship

- Global Average Pooling
  - Reducing h*w*d vector to 1*1*d by taking average of all h*w
  - Preserves spatial relationship
  - Last layer before pooling with T feature maps



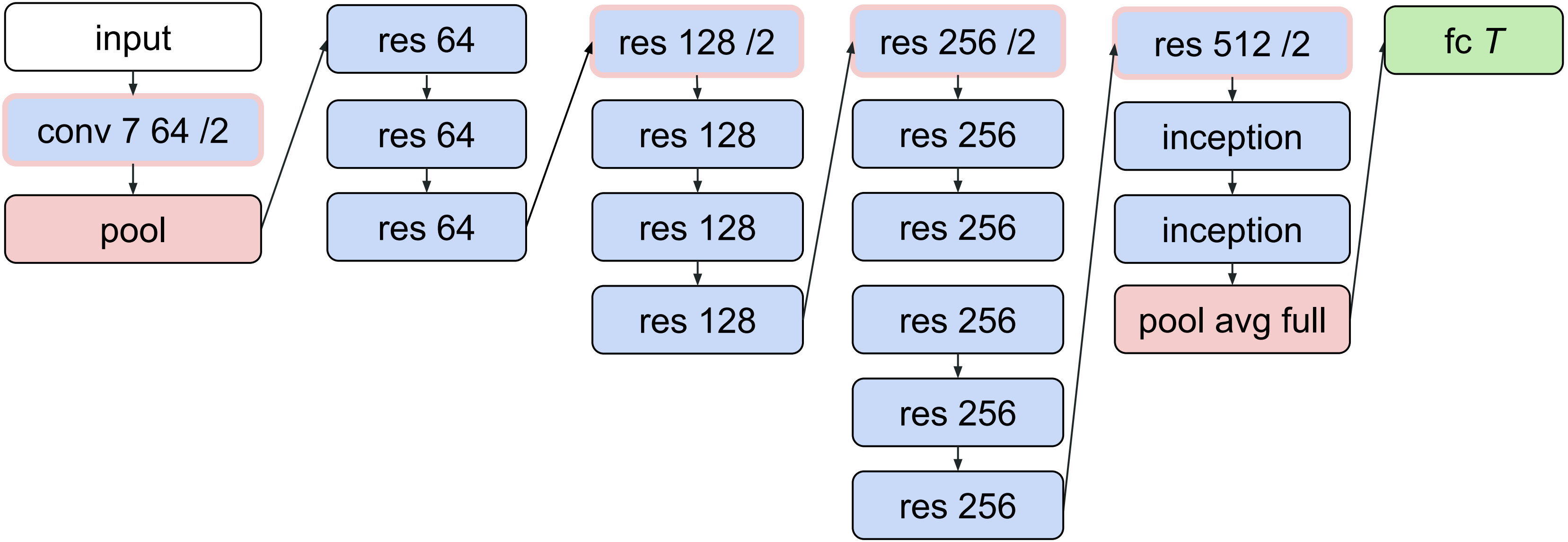
Inception

- Aggressive size reduction in the first few layers
- Global Average Pooling instead of flattening



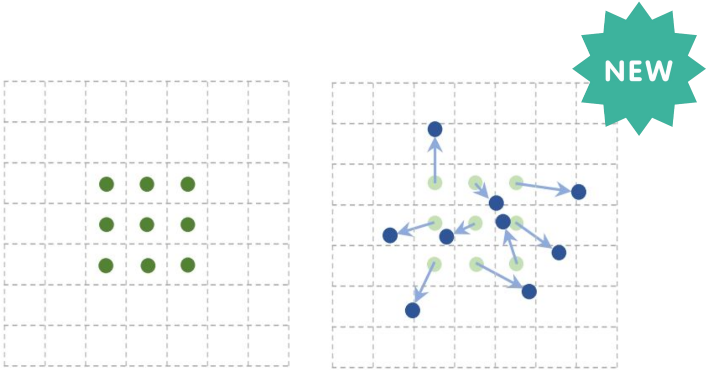
ResNet

- Aggressive size reduction in the first few layers
- Size reduction with strided convolutions instead of pooling

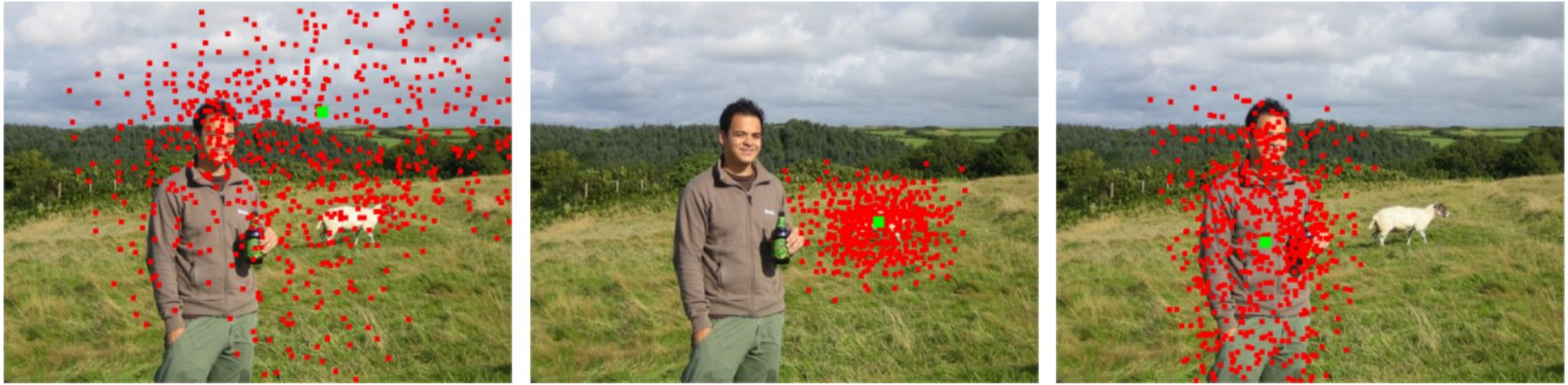


Special forms of convolutions

- Dilated Convolutions
- Deformable Convolutions
- Many more...



Source [3]



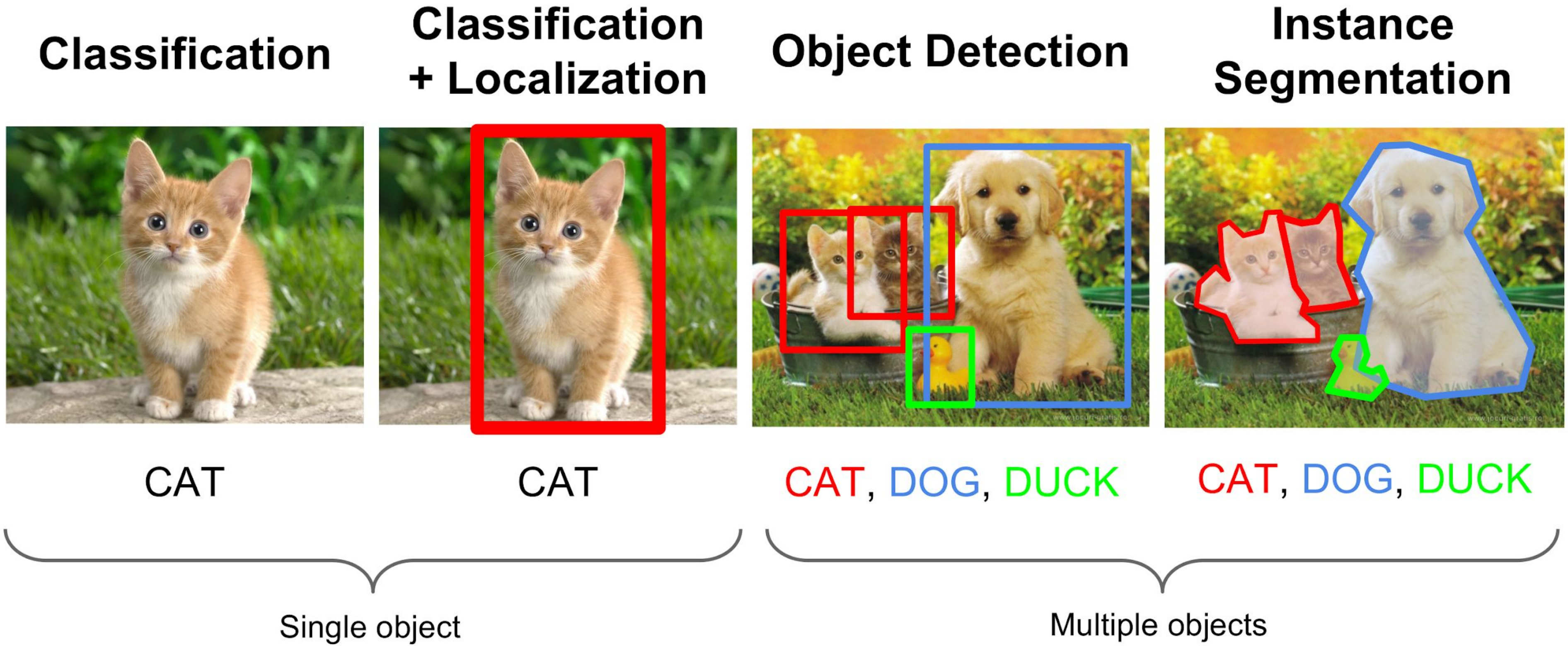
Considerations

- How many layers should I use?
  - It depends
  - Adding more layers might not bring any benefit
- What hyperparameters should I use?
  - Grid search / Random Search / Evolutionary Search
- What image resolution?
  - Start small and see if increasing size helps
- Which architecture is the best?
  - Playground of current research ⟶ Read papers
  - Neural Architecture Search ⟶ Automatic selection of architecture [6, 7]
    - Try it: https://colab.research.google.com/drive/1MWUnkkIG4QYpzQLFf8vjBizt0mR-YBJE

Applications

Real-Time Object Detection with YOLO

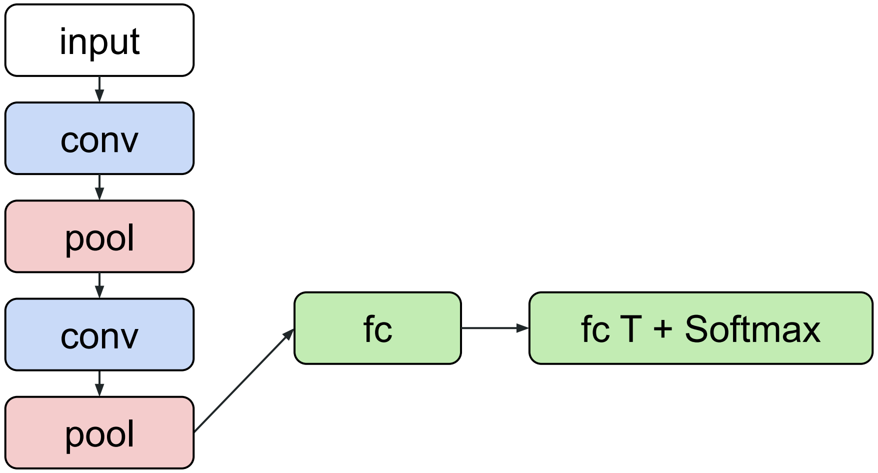
Typical Computer Vision Tasks



Source: http://cs231n.github.io

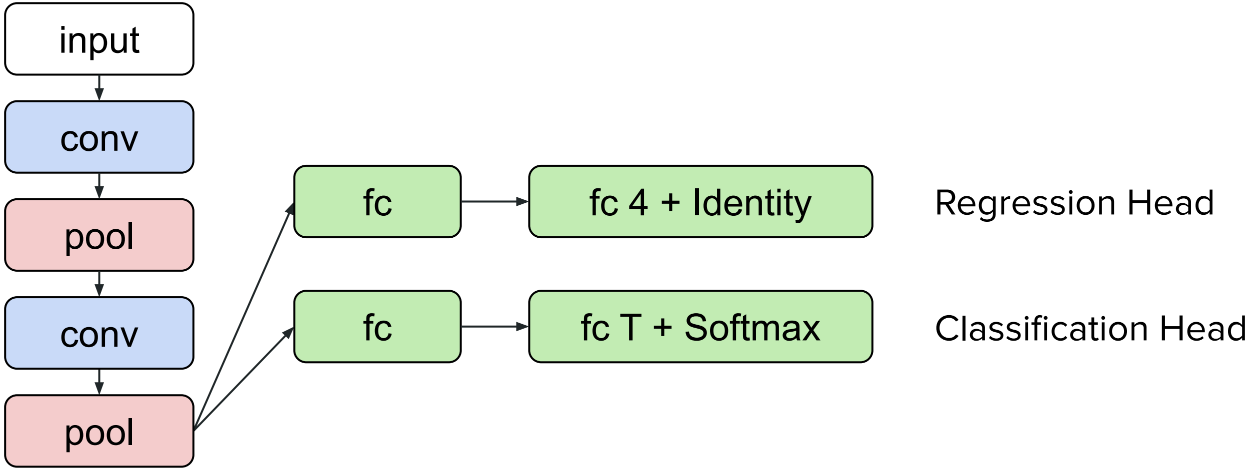
CNNs for Image Classification

- Backend: Conv + Pool
- Frontend: FC or Global Avg Pool + Softmax



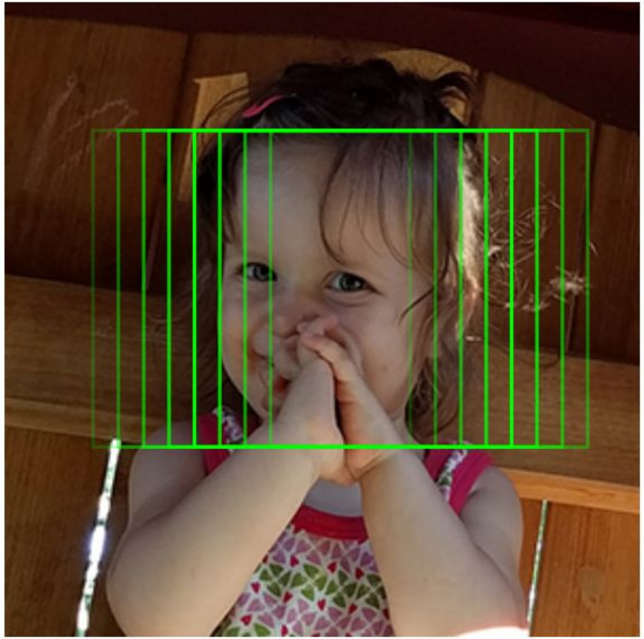
CNNs for Image Localization

- Backend: Conv + Pool
- Frontend:
  - One head for classification
  - One head for bounding box regression



CNNs for Object Detection

Naive: Sliding Window



Source: https://github.com/cpra/dlvc2016

CNNs for Object Detection: Faster R-CNN

Two stages:

- Region proposal
- Proposal classification + refinement

R-CNN

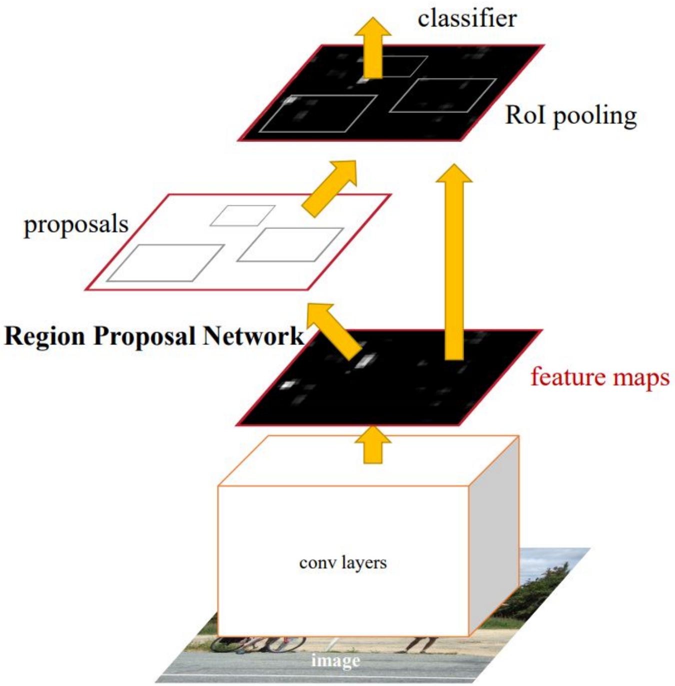
- External mechanism for proposal generation

Fast R-CNN

- Learnt proposals, two networks

Faster R-CNN

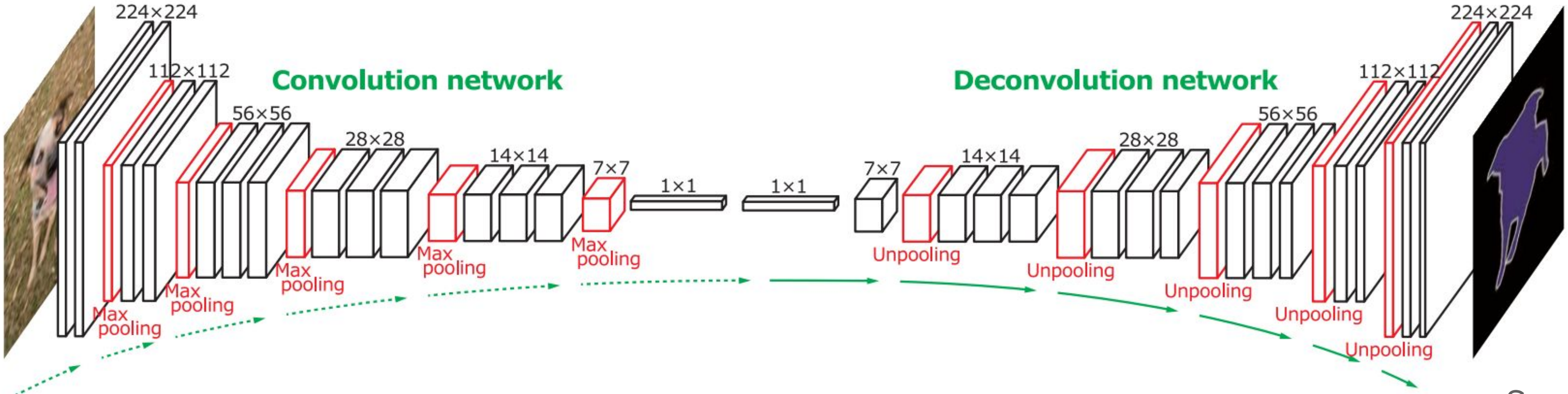
- Shared backbone for both stages



Source: [8]

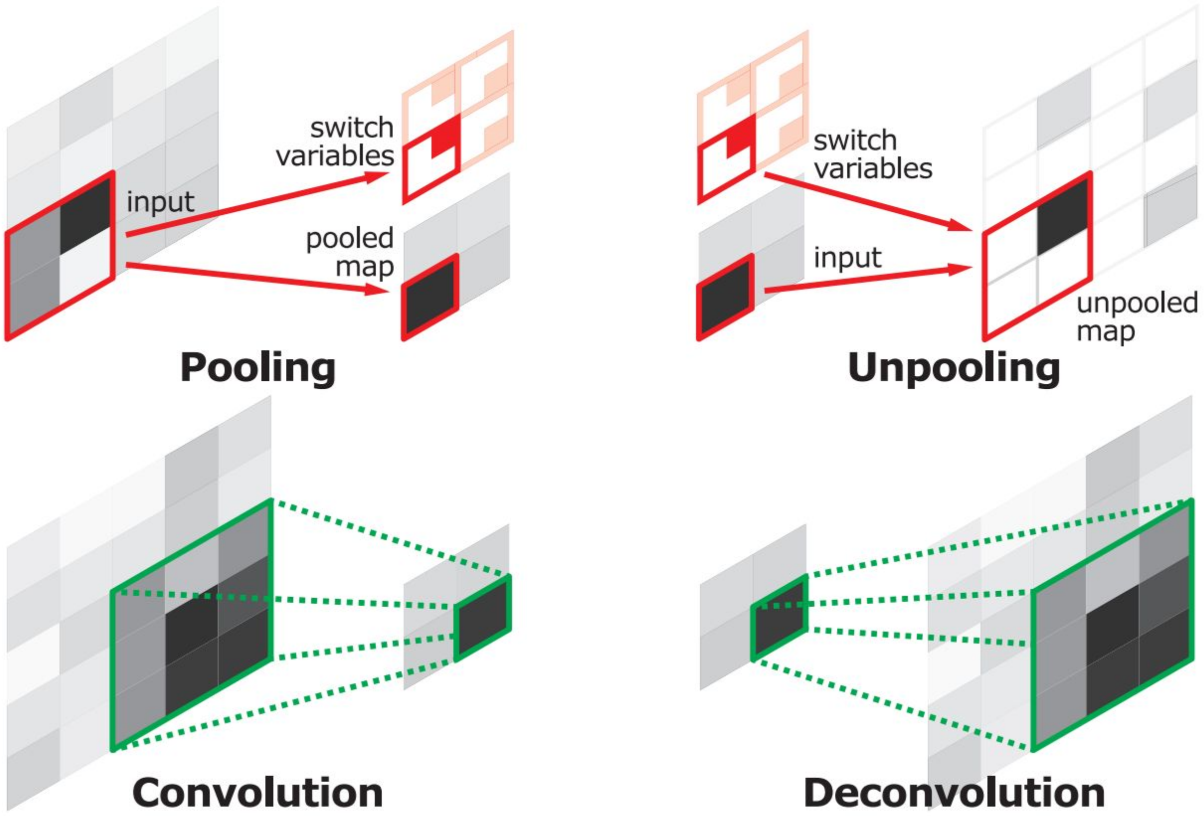
CNNs for Semantic Segmentation

- Fully convolutional network
- Classify each pixel into T classes
  - Downsample part
  - Upsample part



Source: [9]

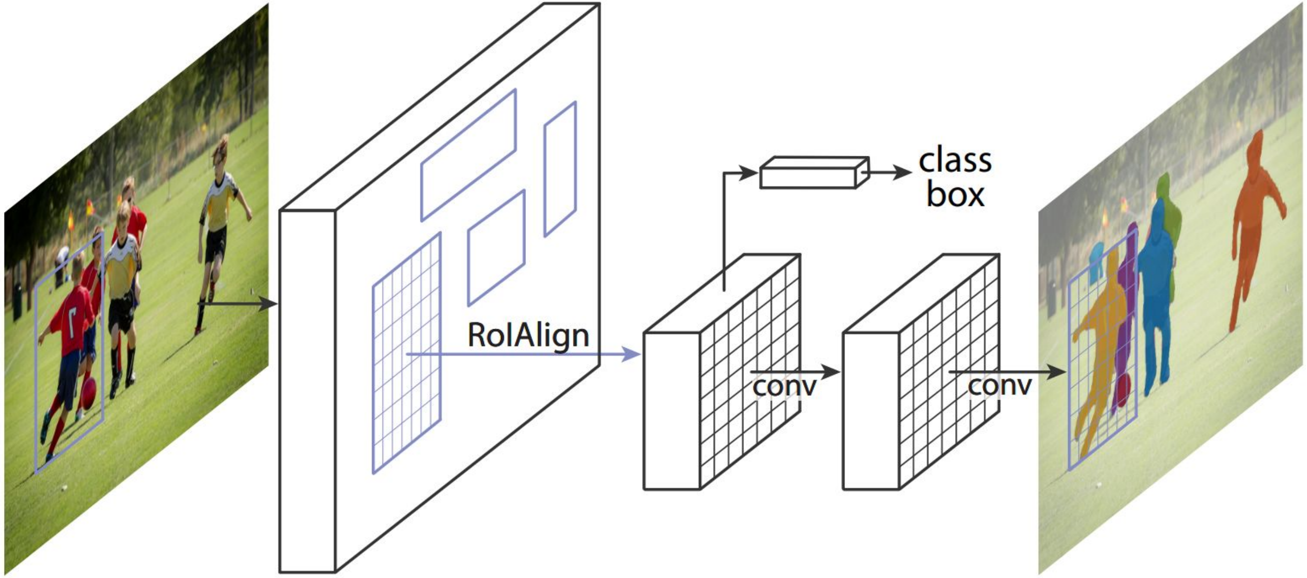
Unpooling and Deconvolution



Source: [9]

CNNs for Instance Segmentation: Mask R-CNN

Faster R-CNN + Additional head for instance segmentation



Source: [10]

Many recent advances

https://github.com/open-mmlab/mmdetection

	ResNet	ResNeXt	SENet	VGG	HRNet
RPN	✓	✓	☐	✗	✓
Fast R-CNN	✓	✓	☐	✗	✓
Faster R-CNN	✓	✓	☐	✗	✓
Mask R-CNN	✓	✓	☐	✗	✓
Cascade R-CNN	✓	✓	☐	✗	✓
Cascade Mask R-CNN	✓	✓	☐	✗	✓
SSD	✗	✗	✗	✓	✗
RetinaNet	✓	✓	☐	✗	✓
GHM	✓	✓	☐	✗	✓
Mask Scoring R-CNN	✓	✓	☐	✗	✓
FCOS	✓	✓	☐	✗	✓
Double-Head R-CNN	✓	✓	☐	✗	✓
Grid R-CNN (Plus)	✓	✓	☐	✗	✓
Hybrid Task Cascade	✓	✓	☐	✗	✓
Libra R-CNN	✓	✓	☐	✗	✓
Guided Anchoring	✓	✓	☐	✗	✓

Summary

- We need good representations and abstractions
  - Ideally learnt from the data
- CNNs connect local neighborhood
  - Nearby pixels are highly correlated
- Pooling can reduce spatial resolution
- Many ideas how to construct better networks
- Different applications achieved by constructing alternative head and clever wiring of the individual layers
- Field moves very quickly
  - Hard to assess the quality of new proposals
  - Time will probably tell

Literature

1. A guide to convolutional arithmetic for deep learning: https://arxiv.org/abs/1603.07285
2. Going deeper with convolutions: https://arxiv.org/abs/1409.4842
3. Deformable Convolutions: https://arxiv.org/abs/1703.06211
4. Deep Residual Networks: https://arxiv.org/abs/1512.03385
5. Global Average Pooling: https://alexisbcook.github.io/2017/global-average-pooling-layers-for-object-localization/
6. Neural Architecture Search: https://arxiv.org/abs/1707.07012
7. AutoKeras: https://github.com/keras-team/autokeras
8. Faster R-CNN: https://arxiv.org/abs/1506.01497
9. Semantic Segmentation using Fully Convolutional Neural Networks: https://arxiv.org/abs/1505.04366
10. Mask R-CNN: https://arxiv.org/abs/1703.06870

Icon credits

Free icons from Flaticon:

- https://www.flaticon.com/free-icon/asking_900415
- https://www.flaticon.com/free-icon/big-data_1554222
- https://www.flaticon.com/free-icon/dimmer_1833516
- https://www.flaticon.com/free-icon/energy-class_1833531
- https://www.flaticon.com/free-icon/crane_222566
- https://www.flaticon.com/free-icon/new_179452
- https://www.flaticon.com/free-icon/code_1383431