Artificial Intelligence for Computer Vision

Krzysztof Krawiec

Institute of Computing Science

Summer School on Applied and Interdisciplinary Artificial Intelligence (S2AI2)

Poznań University of Technology

4.09.2024

POLISH NATIONAL AGENCY FOR ACADEMIC EXCHANGE

STER PROGRAMME

Agenda

Parts 1 and 2:

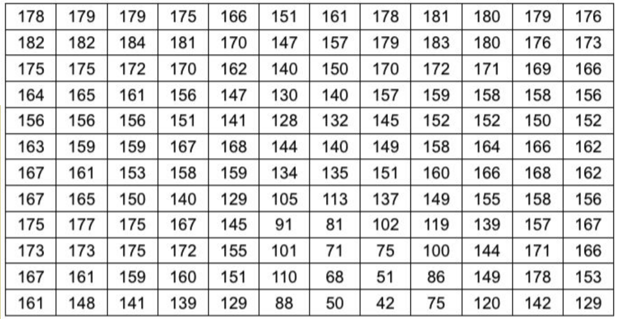
1. **Part 1: “What part”:** Introduction to Computer Vision (CV)
    a. What is special about image data?
    b. Why is it hard?
    c. What can be achieved nowadays?
    d. Conventional approaches to CV.
2. **Part 2: “How part”:** Contemporary AI and Machine Learning for CV
    a. Introduction to neural networks
    b. Convolutional neural networks (ConvNets)
    c. A review of milestone architectures.
    d. Challenges and limitations

Seminar (after lunch):

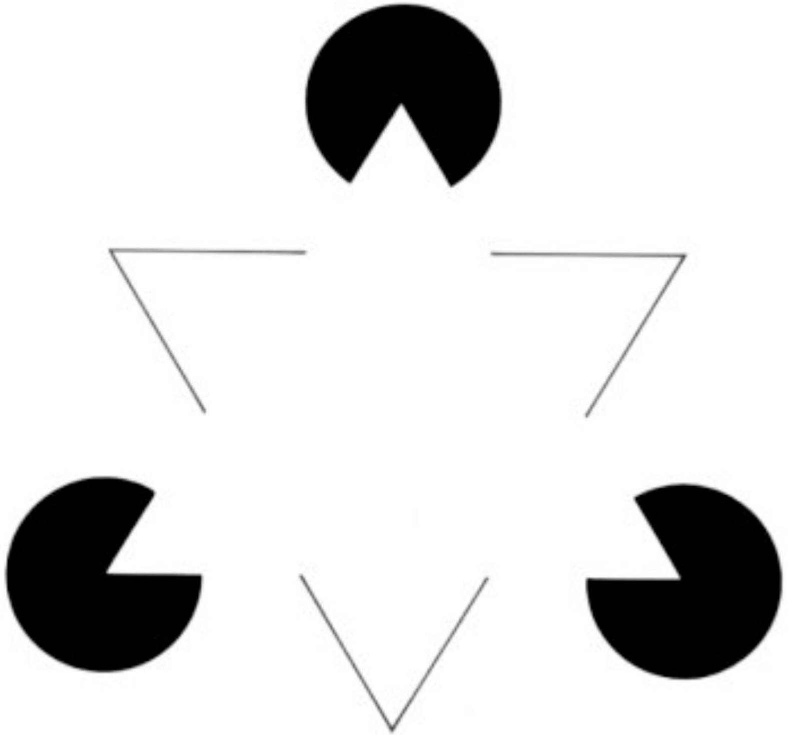
1. Online demos
2. Interactive demos using Jupyter Notebook

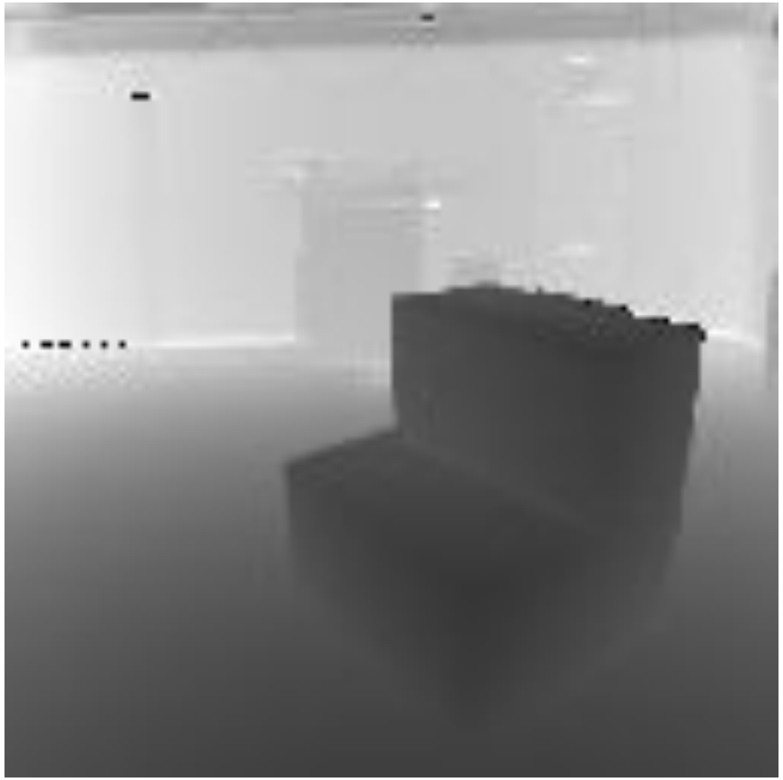
Detailed agenda (with hyperlinks)

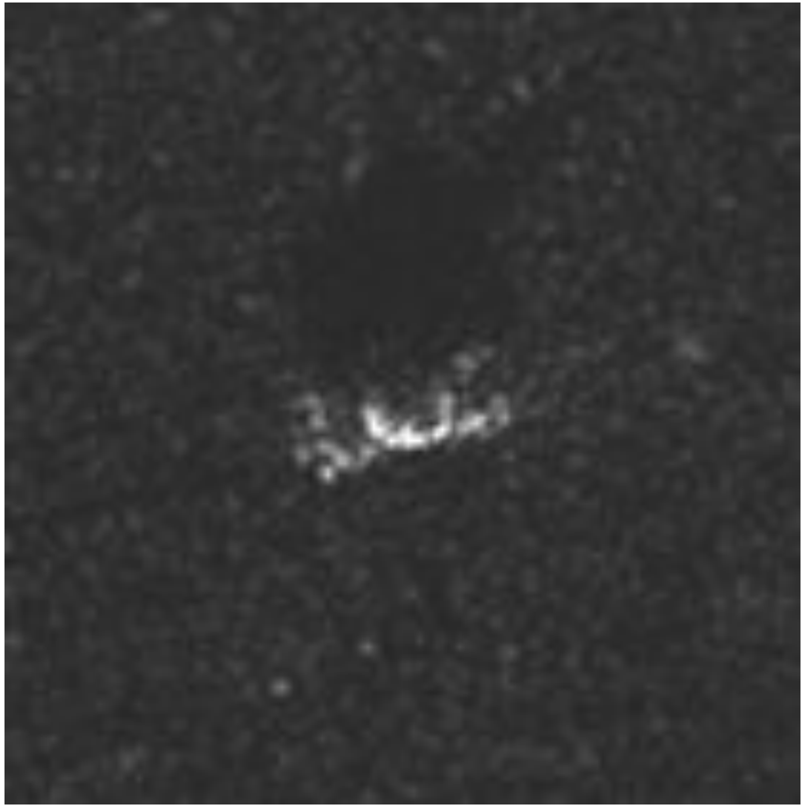
Part 1:
“What part”:
Introduction to Computer Vision

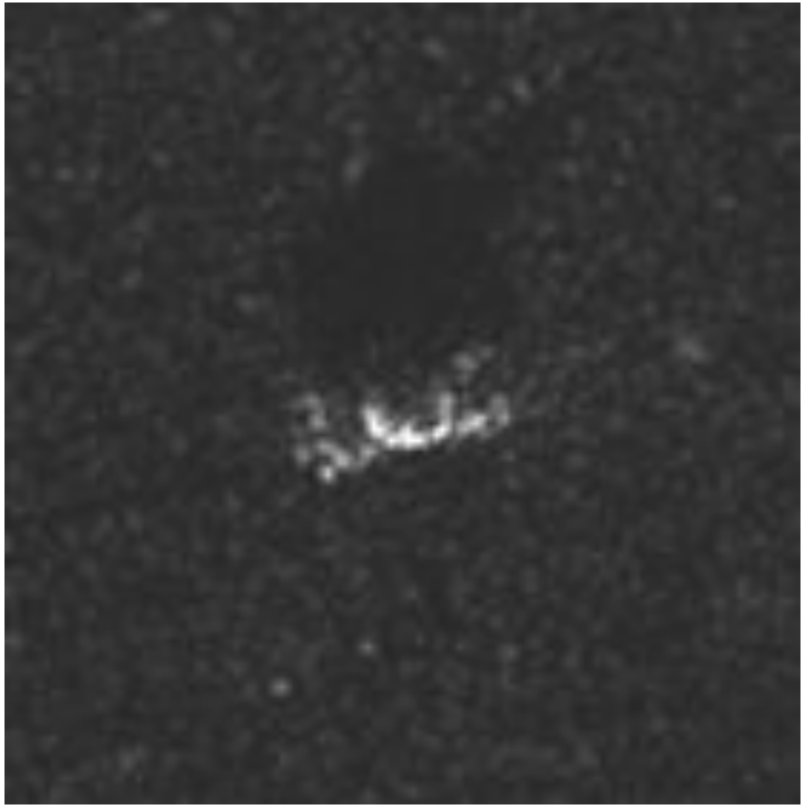


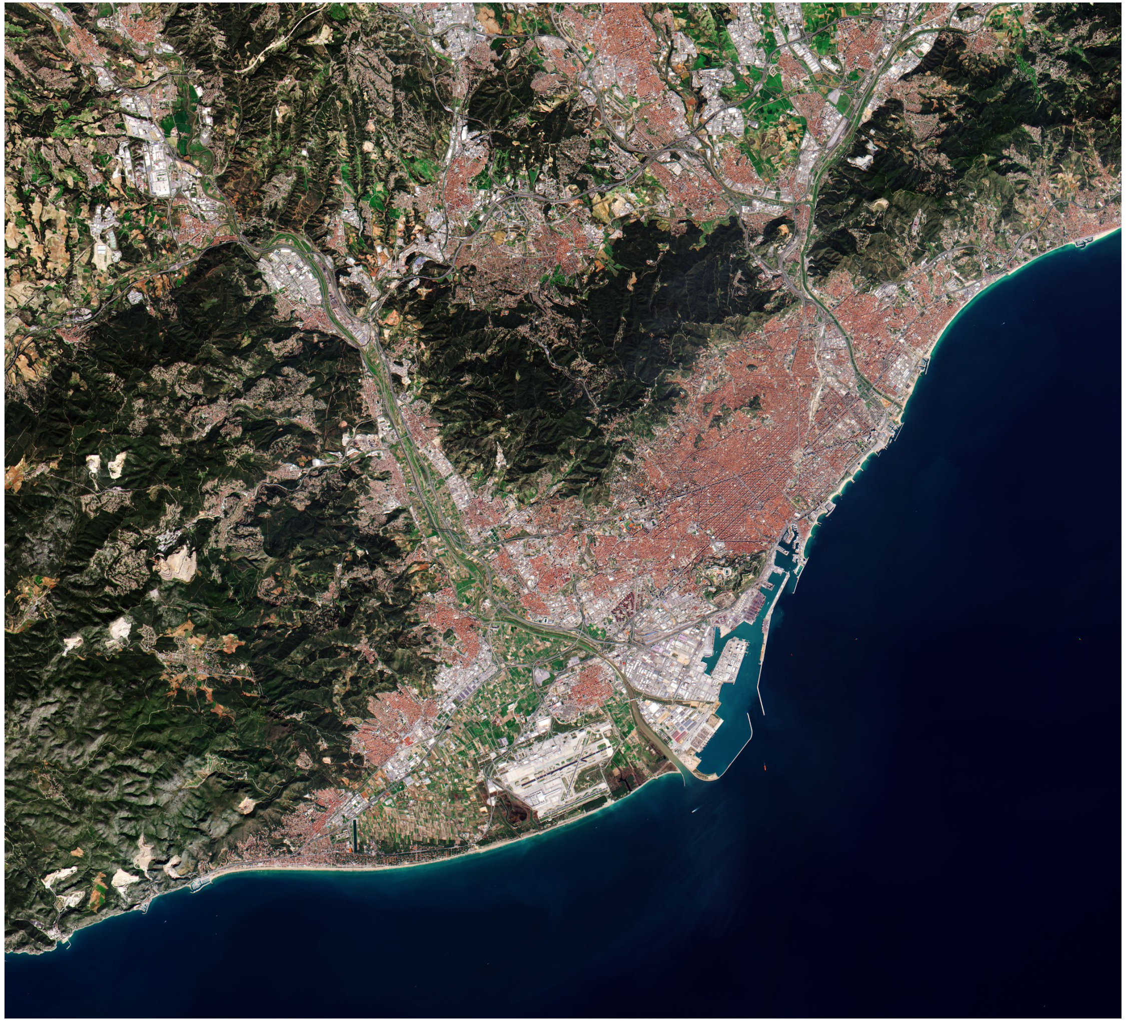
178	179	179	175	166	151	161	178	181	180	179	176
182	182	184	181	170	147	157	179	183	180	176	173
175	175	172	170	162	140	150	170	172	171	169	166
164	165	161	156	147	130	140	157	159	158	158	156
156	156	156	151	141	128	132	145	152	152	150	152
163	159	159	167	168	144	140	149	158	164	166	162
167	161	153	158	159	134	135	151	160	166	168	162
167	165	150	140	129	105	113	137	149	155	158	156
175	177	175	167	145	91	81	102	119	139	157	167
173	173	175	172	155	101	71	75	100	144	171	166
167	161	159	160	151	110	68	51	86	149	178	153
161	148	141	139	129	88	50	42	75	120	142	129



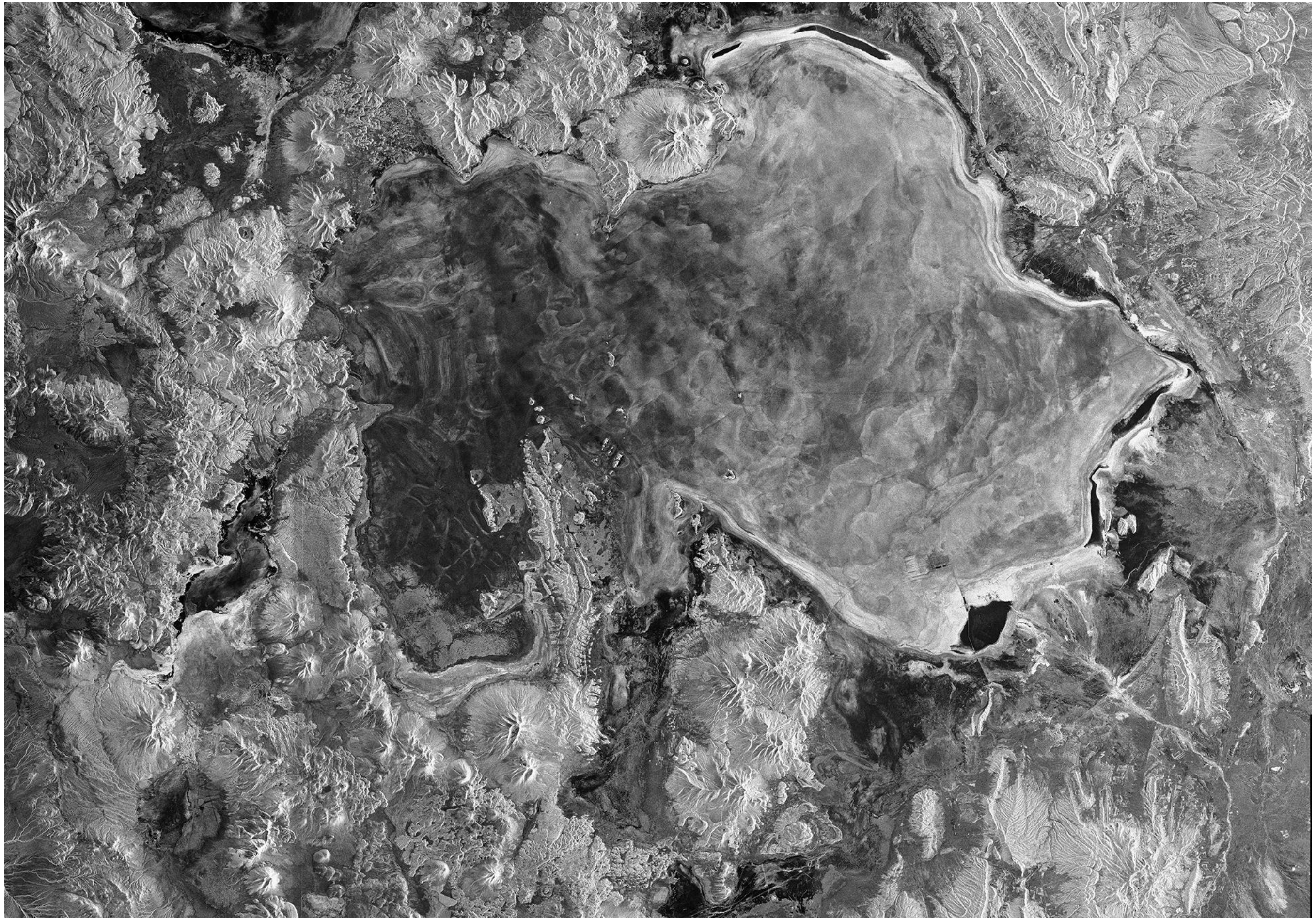




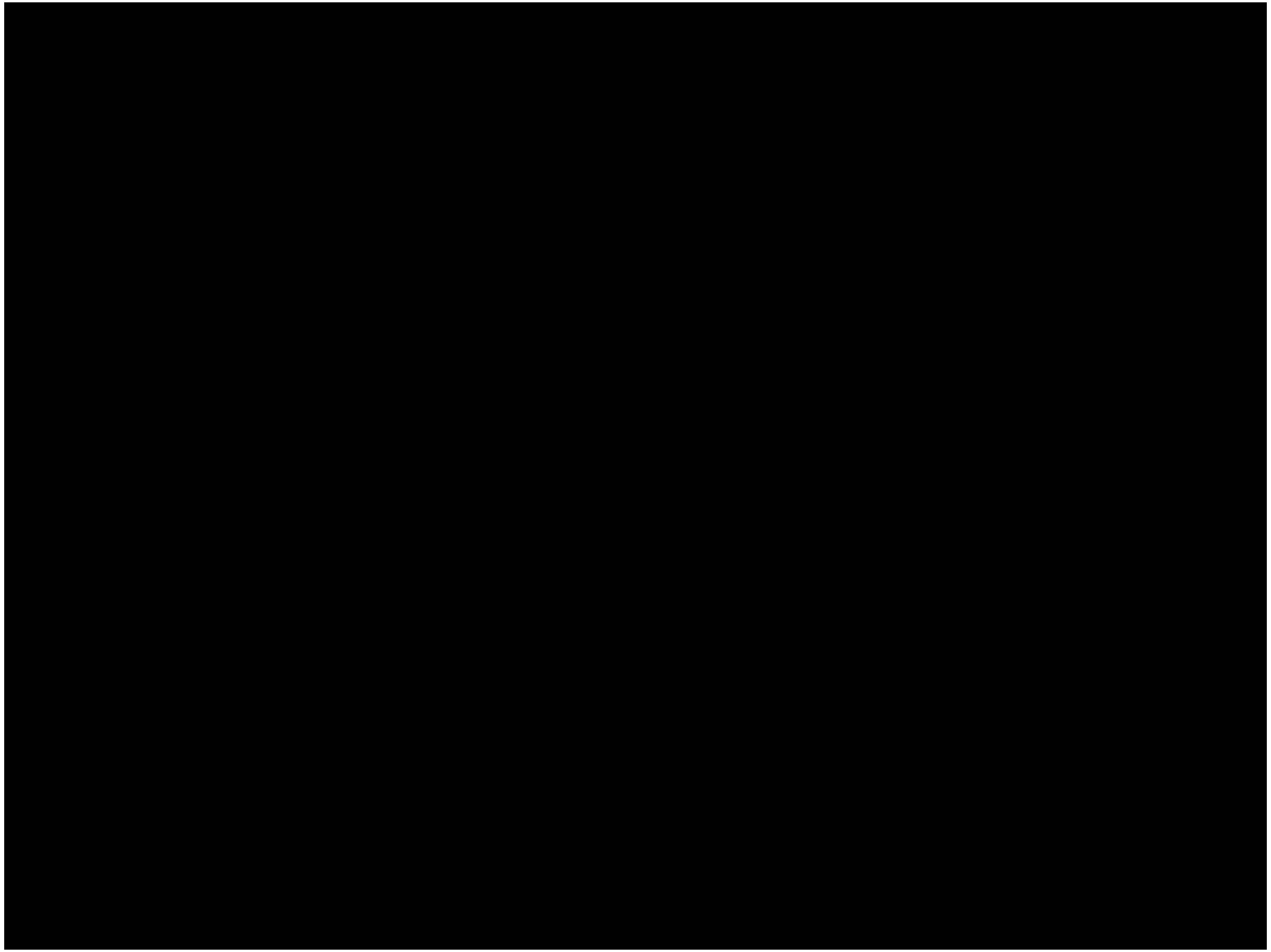




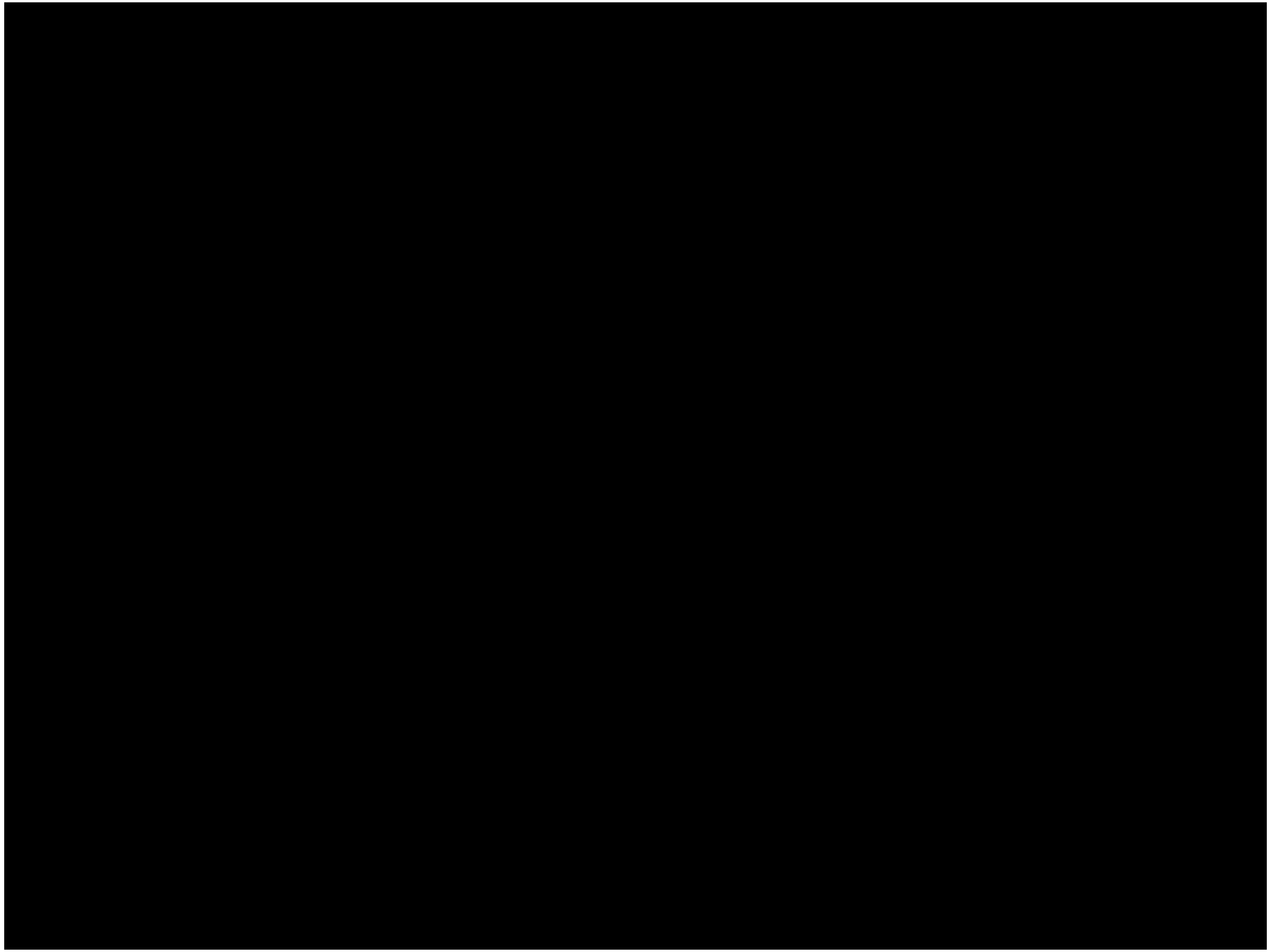
Barcelona. Attribution: Contains modified Copernicus Sentinel data 2019



Sentinel-1A satellite (radar), Bolivia. Attribution: Contains modified Copernicus Sentinel data 2019



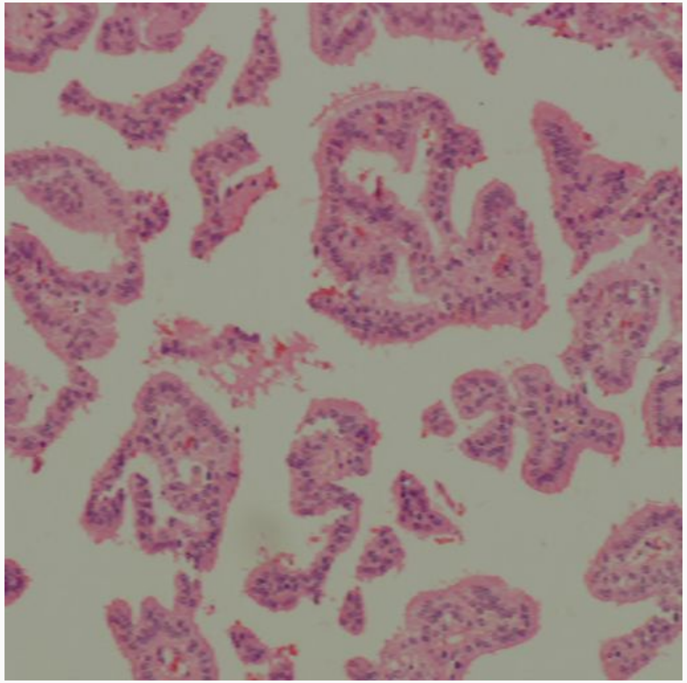
https://www.nature.com/articles/s41598-017-04220-8

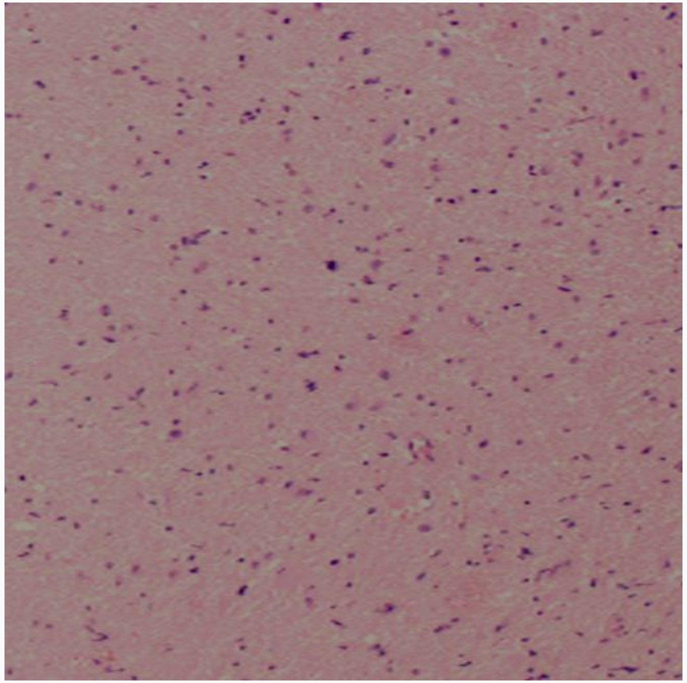


https://www.nature.com/articles/s41598-017-04220-8

Conventional approaches

Diagnosing CNS tumours





J. Jelonek, K. Krawiec, R. Słowiński, and J. Szymaś. “Grizzly – An image processing and analysis system oriented towards medical images”. Journal of Decision Systems 7.3–4 (1998).

Handcrafted segmentation algorithms

Task characteristics:

- Conventional optical microscopy
- 14 classes of neoepithelial tumors
- Textural features (esp. spatial arrangement of nuclei) essential

Solution:

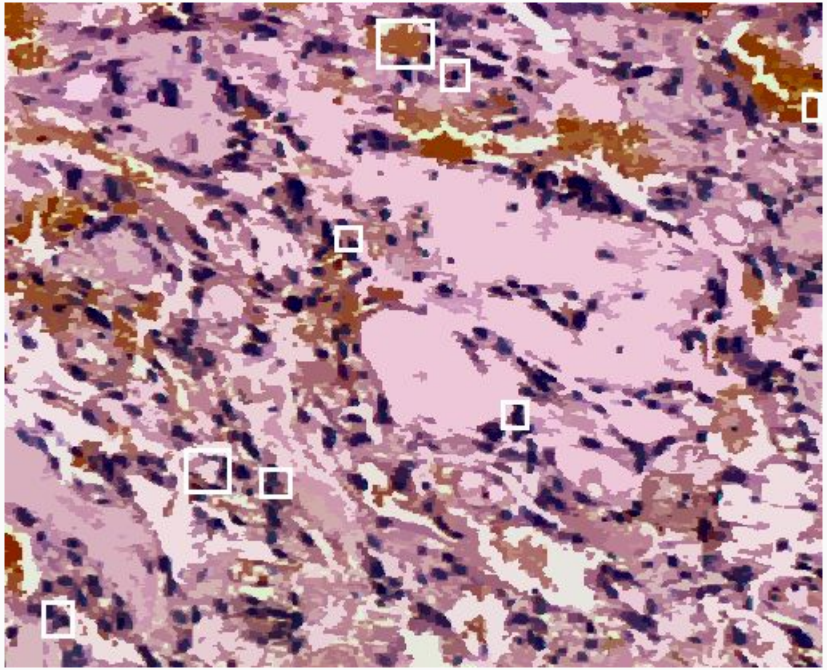
- Handcrafted pipeline of preprocessing, feature extraction, and various classifiers

A (Astrocytic tumours)	B (Glial tumours)
Astrocytoma anaplasticum	Astrocytoma fibrillare
Astrocytoma fibrillare	Oligodendroglioma isomorphum
Astrocytoma gemistocyticum	Ependymoma
Astrocytoma pilocyticum	Choroid plexus papilloma
Astrocytoma protoplasmaticum	Glioblastoma multiforme
Glioblastoma multiforme	Medulloblastoma

J. Jelonek, K. Krawiec, and R. Słowiński. "Rough Set Reduction of Attributes and their Domains for Neural Networks". In: Computational Intelligence 11.2 (1995), pp. 339–347.

Handcrafted segmentation algorithms

A (Astrocytic tumours)	B (Glial tumours)
Astrocytoma anaplasticum	Astrocytoma fibrillare
Astrocytoma fibrillare	Oligodendroglioma isomorphum
Astrocytoma gemistocyticum	Ependymoma
Astrocytoma pilocyticum	Choroid plexus papilloma
Astrocytoma protoplasmaticum	Glioblastoma multiforme
Glioblastoma multiforme	Medulloblastoma



J. Jelonek, K. Krawiec, and R. Słowiński. "Rough Set Reduction of Attributes and their Domains for Neural Networks". In: Computational Intelligence 11.2 (1995), pp. 339–347.

Automated feature construction

Automated feature construction

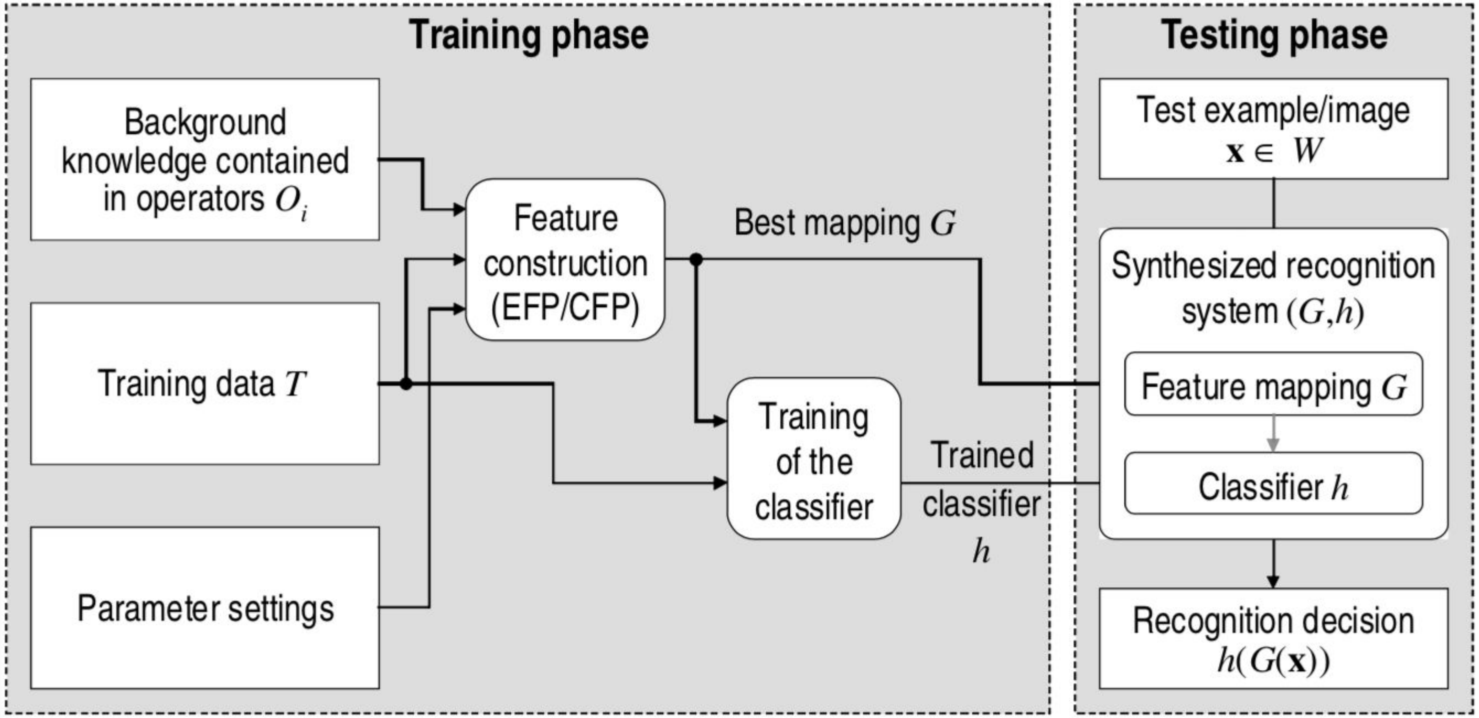
Manual feature construction is:

- Time consuming
- Subjective
- Expensive

Can we automate it?

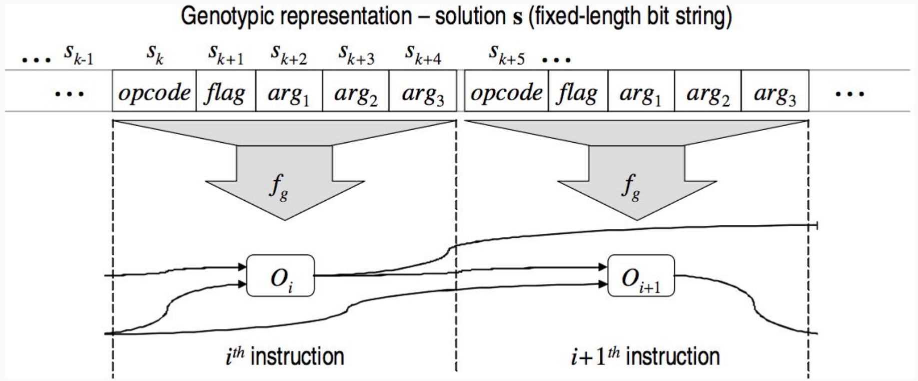
- In particular, can we make it a part of the training/design process?

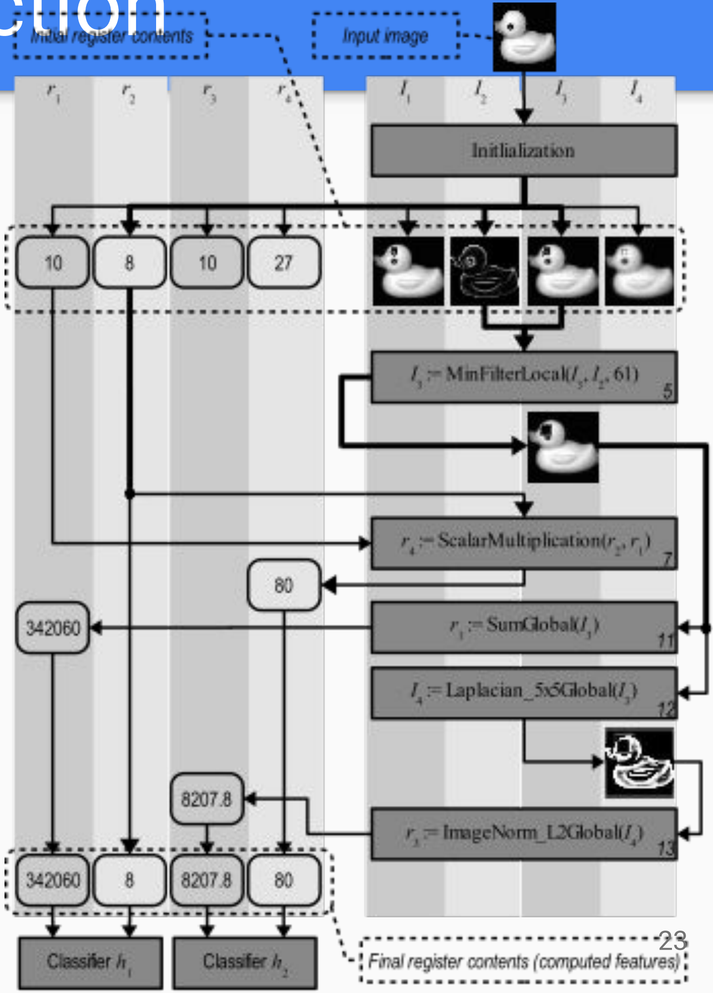
Evolutionary feature construction



Training phase
Testing phase
Background knowledge contained in operators O_i
Training data T
Parameter settings
Feature construction (EFP/CFP)
Best mapping G
Training of the classifier
Trained classifier h
Test example/image $\mathbf{x} \in W$
Synthesized recognition system (G,h)
Feature mapping G
Classifier h
Recognition decision $h(G(\mathbf{x}))$

Evolutionary feature construction

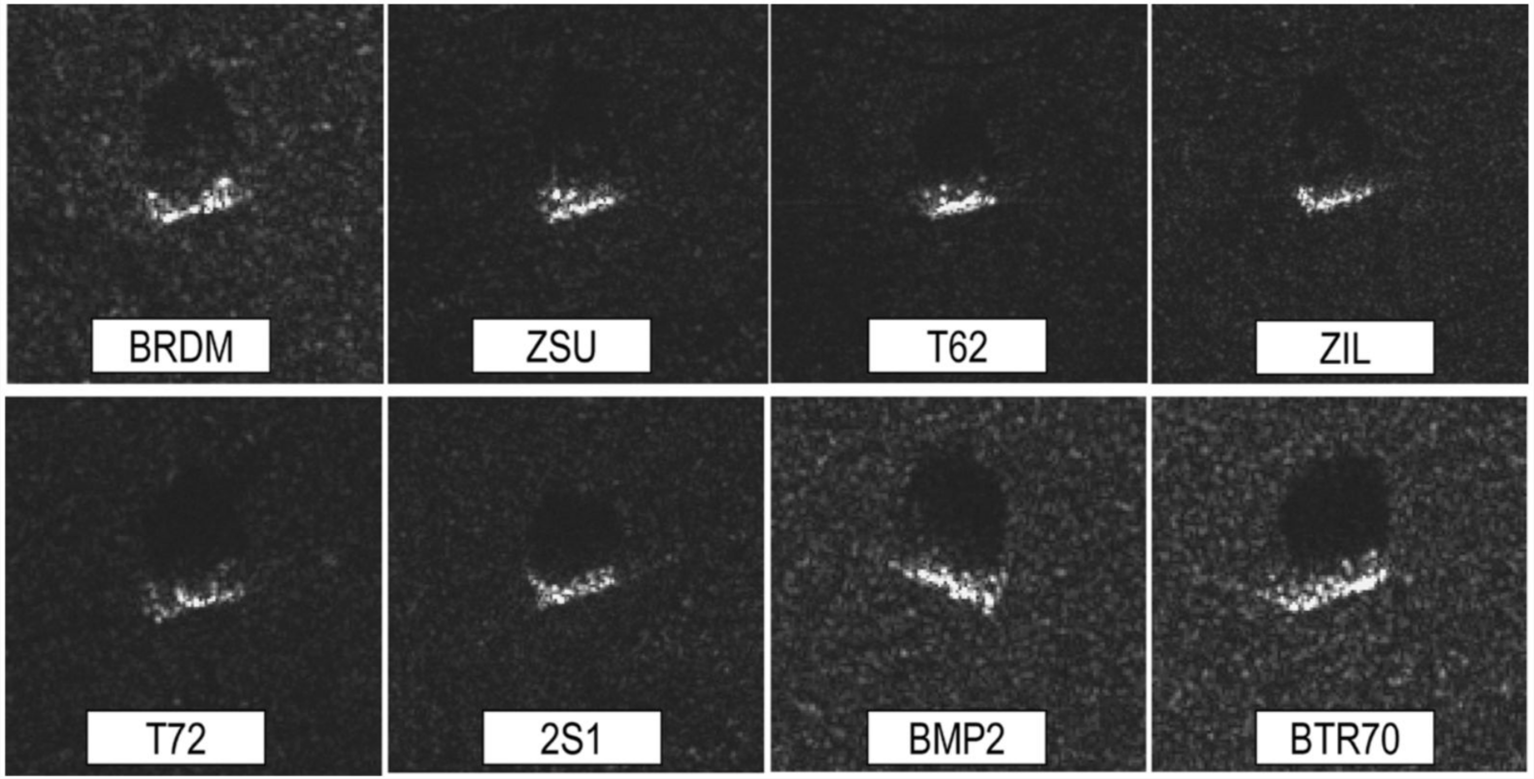




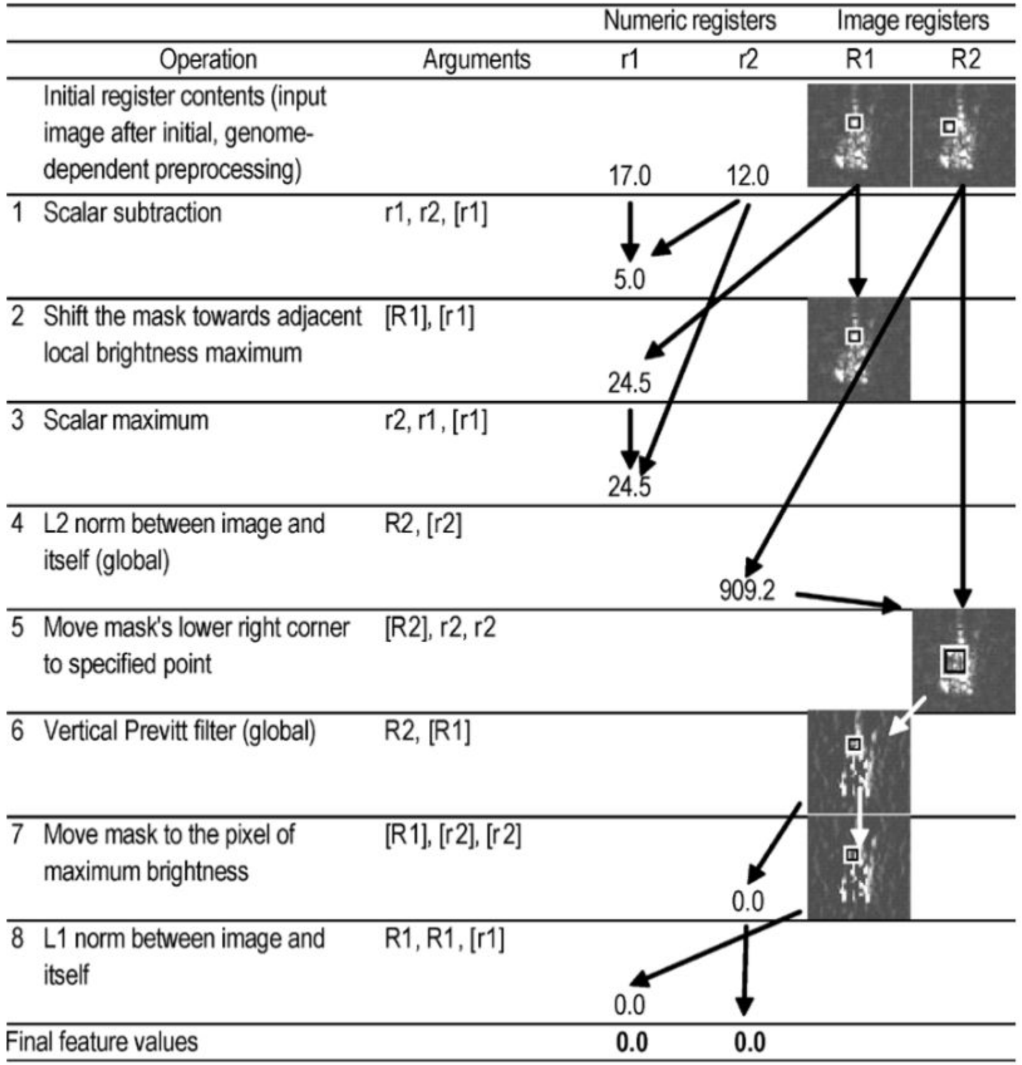
Initial register contents
Input image
r_1
r_2
r_3
r_4
I_1
I_2
I_3
I_4
Initialization
10
8
10
27
$I_3 := \text{MinFilterLocal}(I_1, I_2, 61)$
5
$r_4 := \text{ScalarMultiplication}(r_2, r_1)$
7
80
342060
$r_1 := \text{SumGlobal}(I_3)$
11
$I_4 := \text{Laplacian\_5x5Global}(I_3)$
12
8207.8
$r_3 := \text{ImageNorm\_L2Global}(I_4)$
13
342060
8
8207.8
80
Classifier h_1
Classifier h_2
Final register contents (computed features)

Krzysztof Krawiec and Bir Bhanu. “Visual Learning by Evolutionary and Coevolutionary Feature Synthesis”. IEEE Transactions on Evolutionary Computation 11 (5 Oct. 2007).

Object detection in SAR imagery

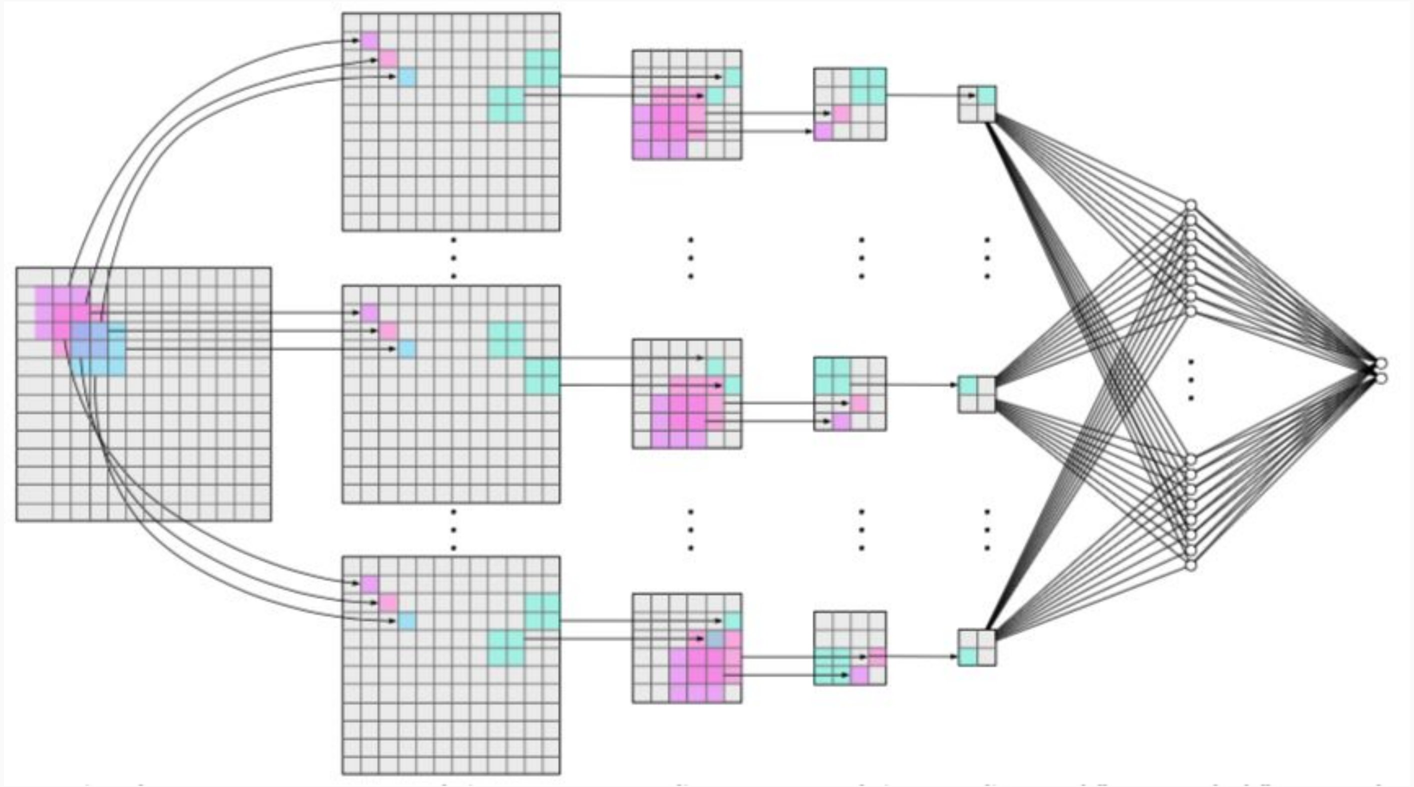


Krzysztof Krawiec and Bir Bhanu. “Visual Learning by Coevolutionary Feature Synthesis”.
IEEE Transactions on System, Man, and Cybernetics – Part B 35.3 (June 2005), pp. 409–425.

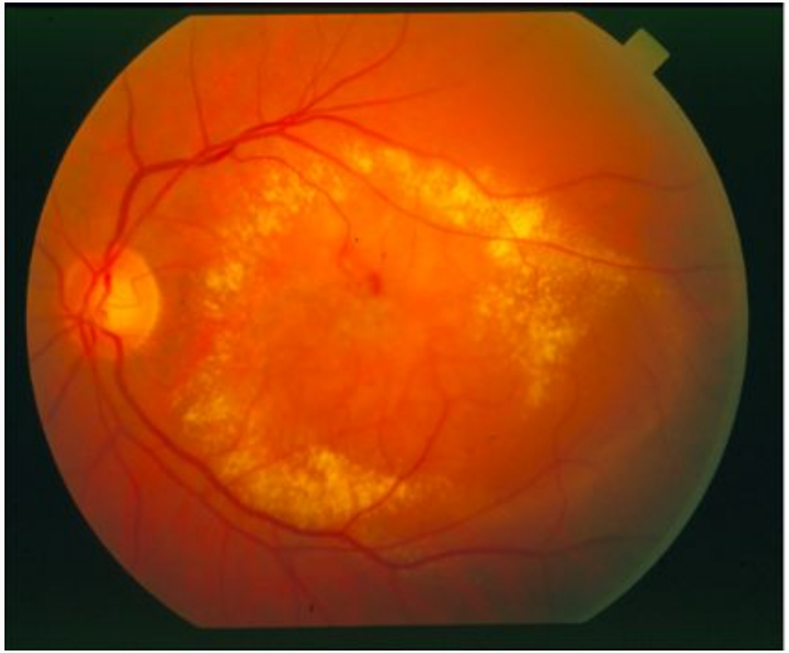


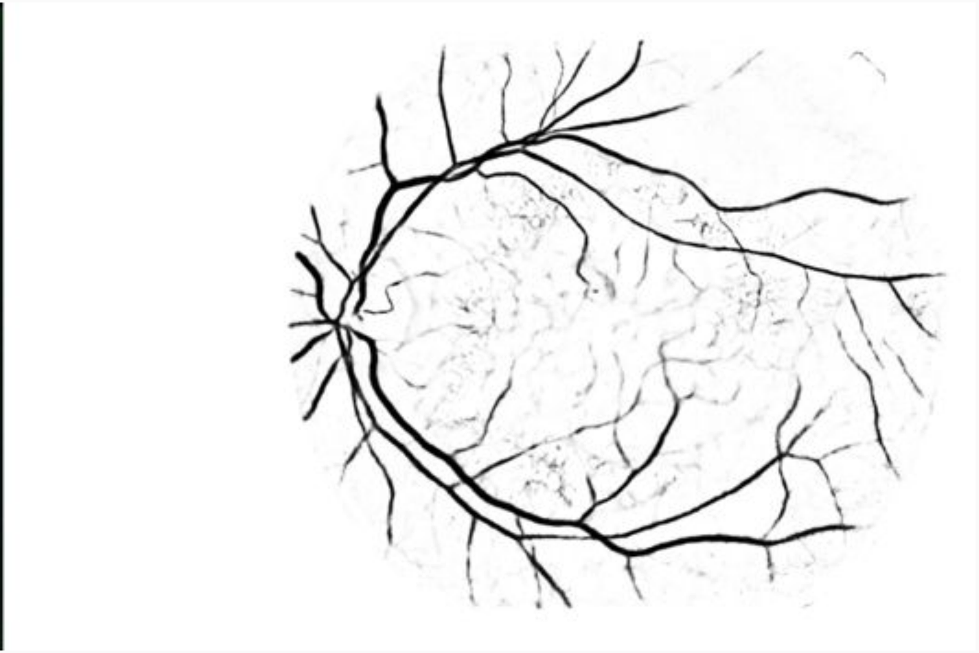
		Numeric registers		Image registers	
Operation	Arguments	r1	r2	R1	R2
Initial register contents (input image after initial, genome-dependent preprocessing)		17.0	12.0		
1 Scalar subtraction	r1, r2, [r1]	5.0			
2 Shift the mask towards adjacent local brightness maximum	[R1], [r1]	24.5			
3 Scalar maximum	r2, r1, [r1]	24.5			
4 L2 norm between image and itself (global)	R2, [r2]		909.2		
5 Move mask's lower right corner to specified point	[R2], r2, r2				
6 Vertical Previtt filter (global)	R2, [R1]				
7 Move mask to the pixel of maximum brightness	[R1], [r2], [r2]		0.0		
8 L1 norm between image and itself	R1, R1, [r1]	0.0			
Final feature values		**0.0**	**0.0**		

Deep neural networks for image analysis

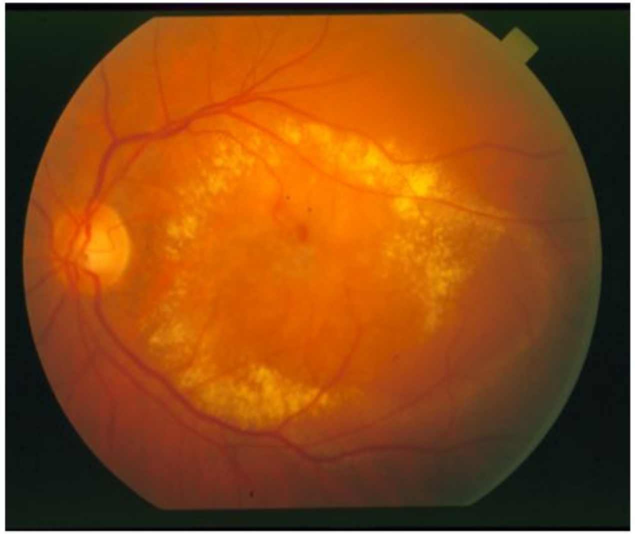
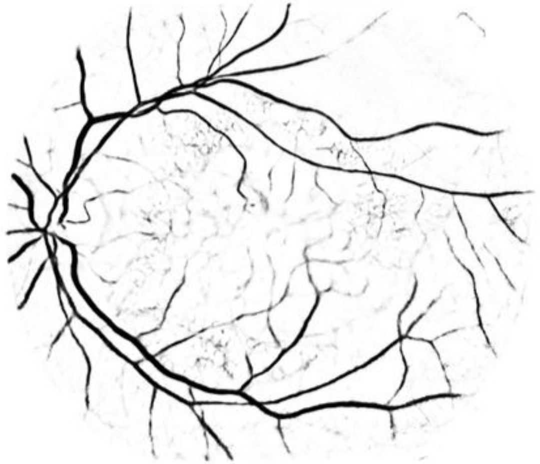
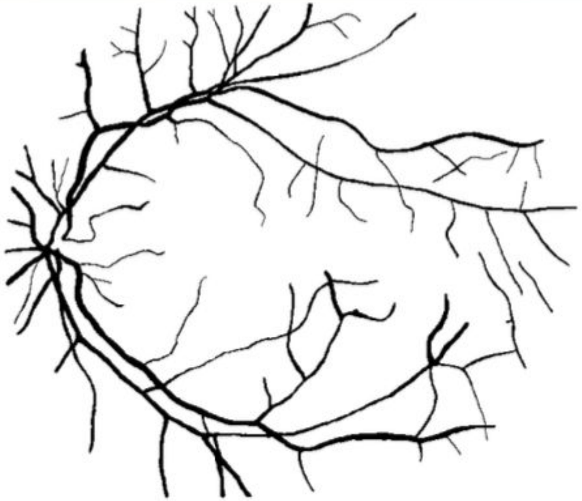
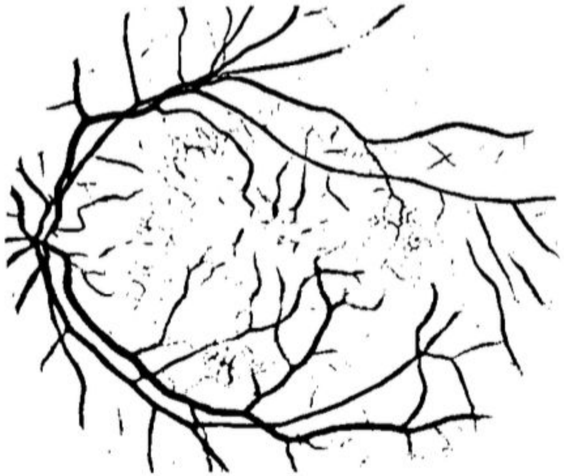


Detector’s response



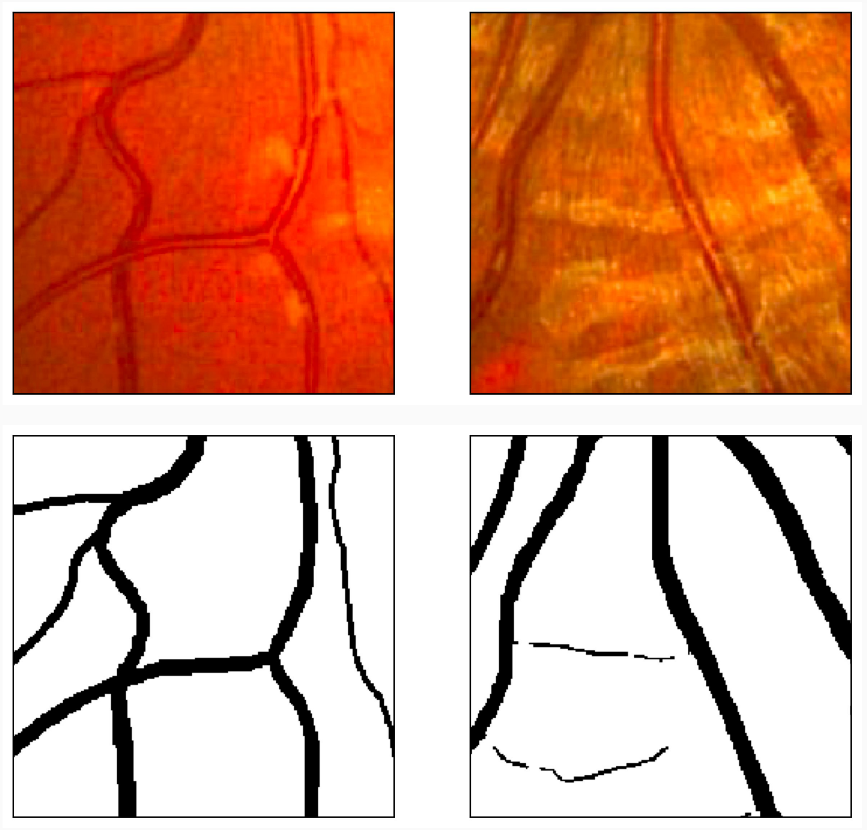


Liskowski, Paweł, and Krzysztof Krawiec. “Segmenting Retinal Blood Vessels With Deep Neural Networks.”
IEEE Transactions on Medical Imaging 35, no. 11 (November 2016): 2369–80. https://doi.org/10.1109/TMI.2016.2546227.

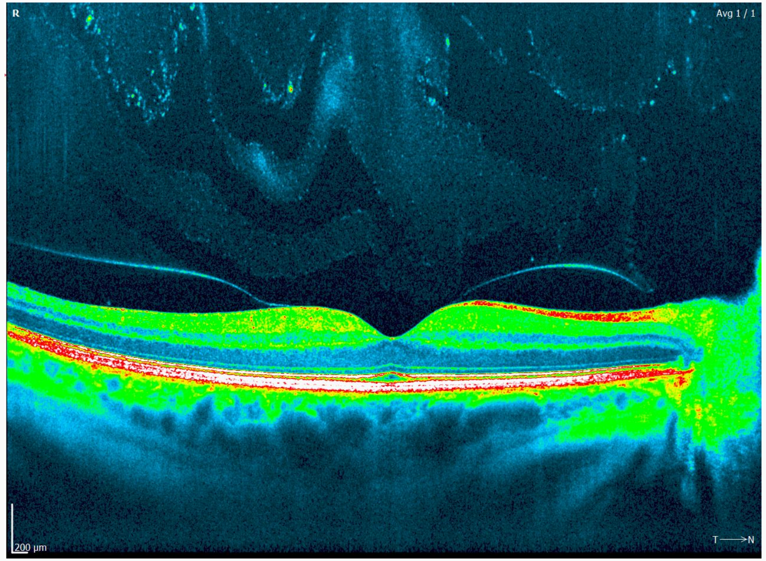





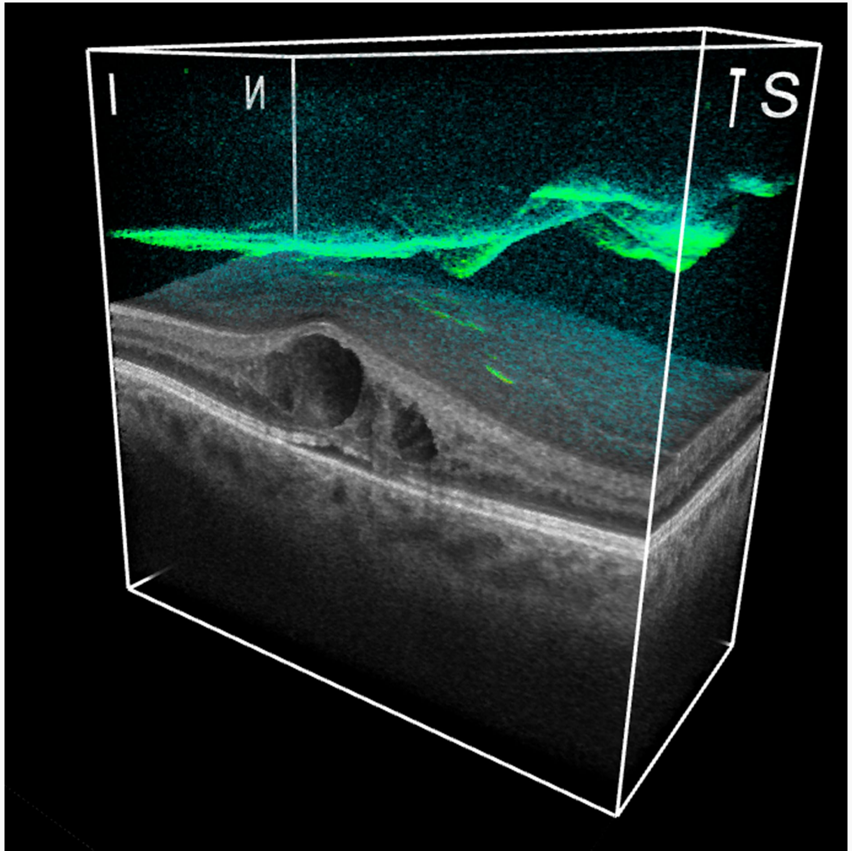
Central vessel reflex

- Light bounces off top surface of vessels particularly strongly
- Source of common errors



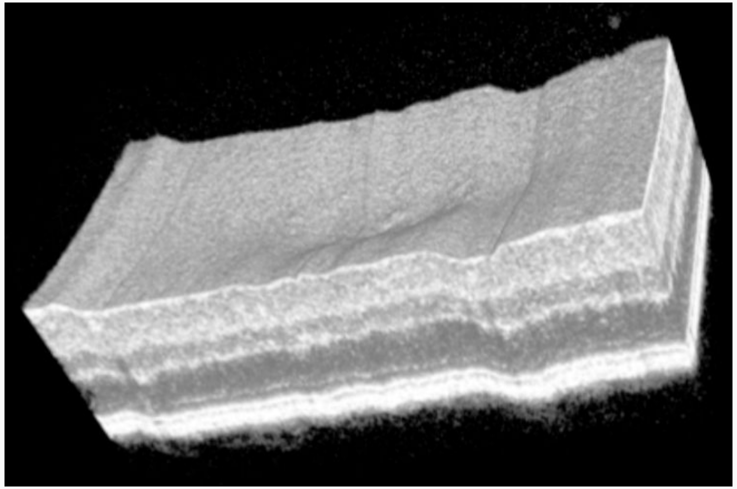
OCT images

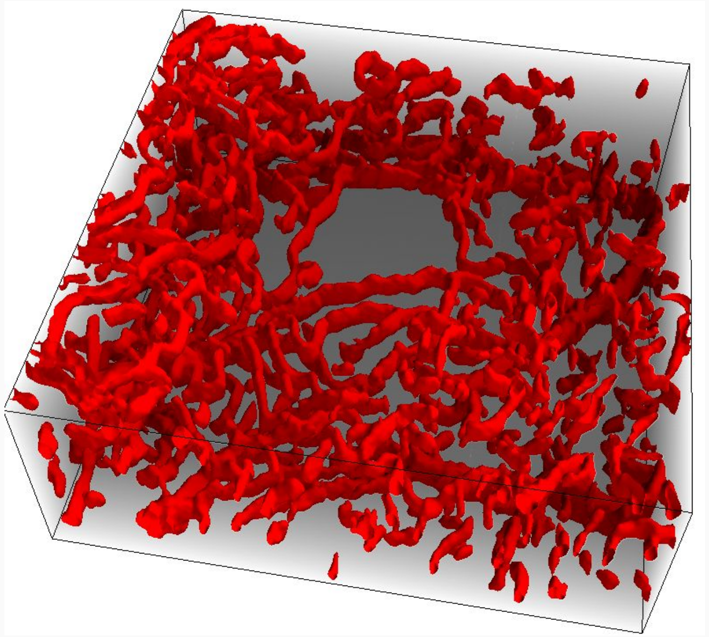




Detection of blood vessels in OCT

Optical coherence tomography



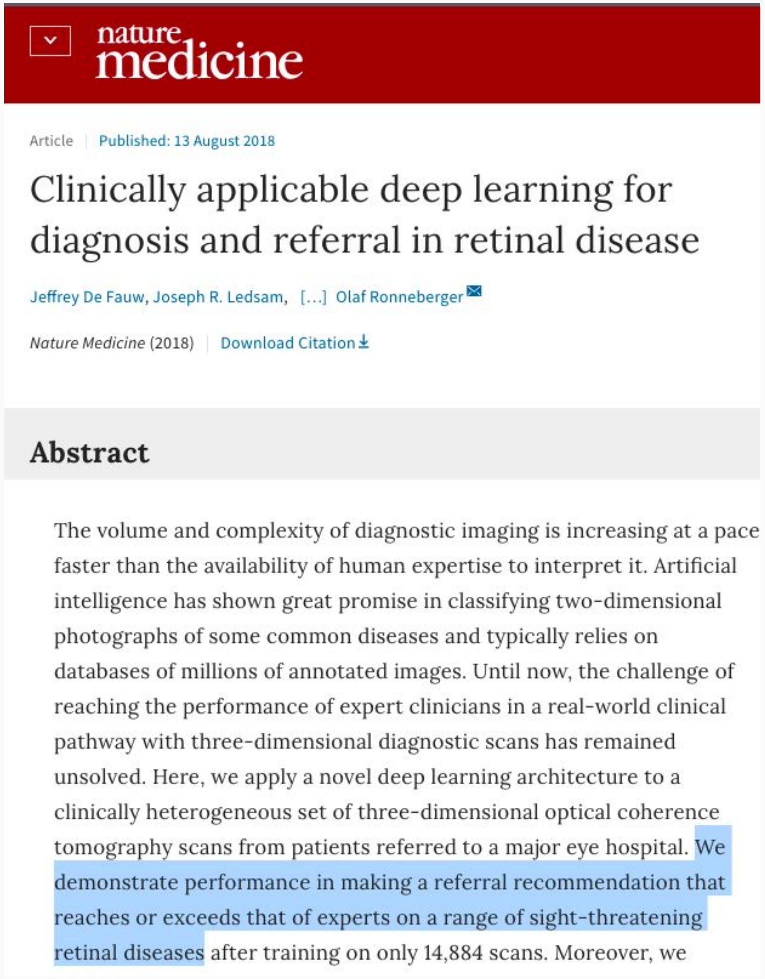


Detection of blood vessels in OCT

Karol Karnowski, Anna Ajduk, Bartosz Wieloch, Szymon Tamborski, Krzysztof Krawiec, Maciej Wojtkowski, and Maciej Szkulmowski. “Optical coherence microscopy as a novel, non-invasive method for the 4D live imaging of early mammalian embryos”. In: Scientific Reports 7.1 (2017), p. 4165.

End-to-end diagnosing support

- Ophthalmology



nature medicine

Article | Published: 13 August 2018

Clinically applicable deep learning for diagnosis and referral in retinal disease

Jeffrey De Fauw, Joseph R. Ledsam, [...] Olaf Ronneberger

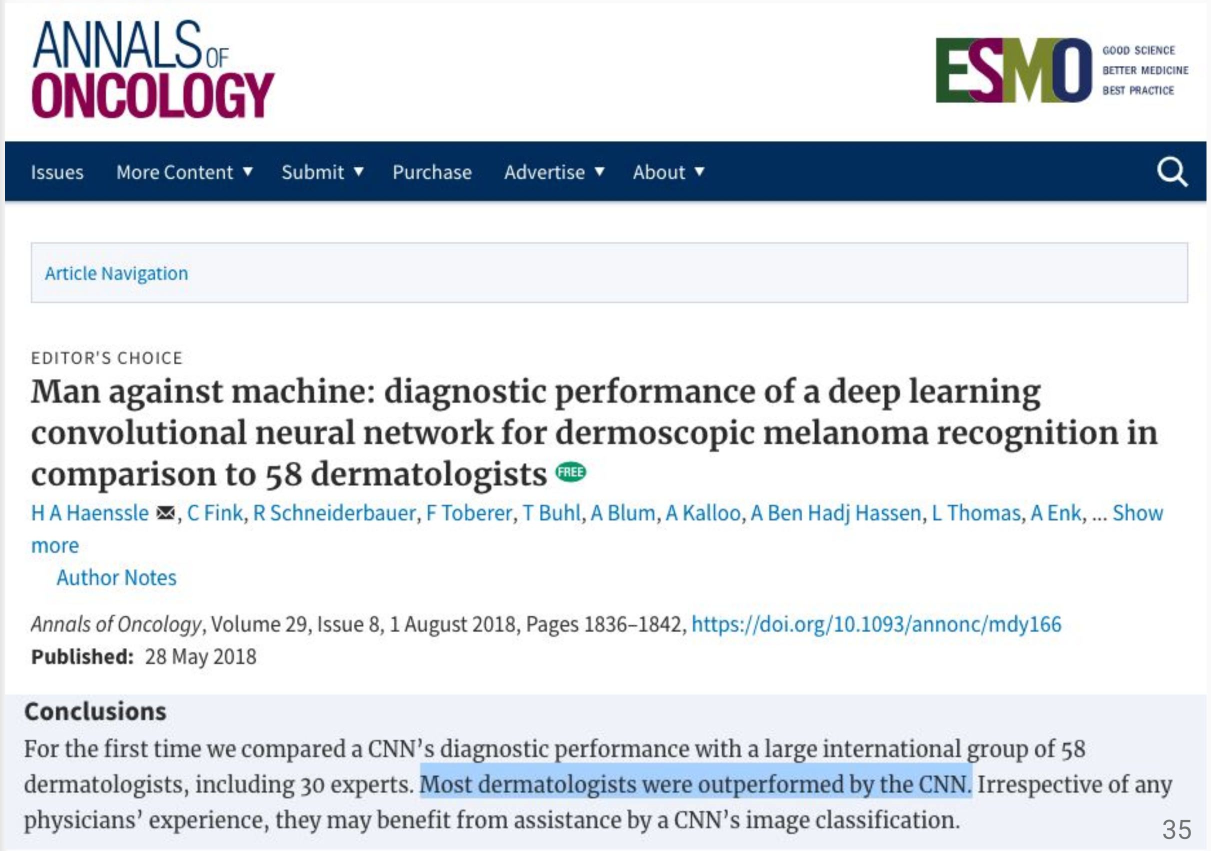
Nature Medicine (2018) | Download Citation

Abstract

The volume and complexity of diagnostic imaging is increasing at a pace faster than the availability of human expertise to interpret it. Artificial intelligence has shown great promise in classifying two-dimensional photographs of some common diseases and typically relies on databases of millions of annotated images. Until now, the challenge of reaching the performance of expert clinicians in a real-world clinical pathway with three-dimensional diagnostic scans has remained unsolved. Here, we apply a novel deep learning architecture to a clinically heterogeneous set of three-dimensional optical coherence tomography scans from patients referred to a major eye hospital. We demonstrate performance in making a referral recommendation that reaches or exceeds that of experts on a range of sight-threatening retinal diseases after training on only 14,884 scans. Moreover, we

End-to-end diagnosing support

- Ophthalmology
- Dermatology



EDITOR'S CHOICE

Man against machine: diagnostic performance of a deep learning convolutional neural network for dermoscopic melanoma recognition in comparison to 58 dermatologists FREE

H A Haenssle, C Fink, R Schneiderbauer, F Toberer, T Buhl, A Blum, A Kalloo, A Ben Hadj Hassen, L Thomas, A Enk, ... Show more

Author Notes

Annals of Oncology, Volume 29, Issue 8, 1 August 2018, Pages 1836–1842, https://doi.org/10.1093/annonc/mdy166

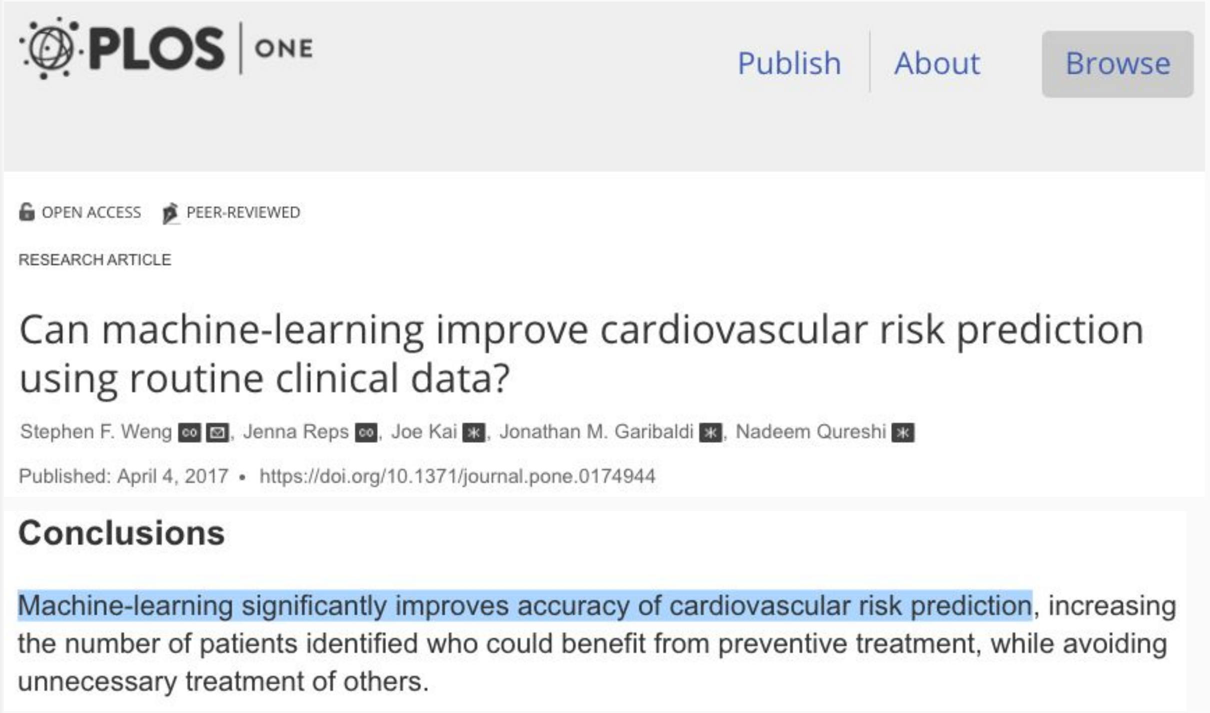
Published: 28 May 2018

Conclusions

For the first time we compared a CNN's diagnostic performance with a large international group of 58 dermatologists, including 30 experts. Most dermatologists were outperformed by the CNN. Irrespective of any physicians' experience, they may benefit from assistance by a CNN's image classification.

End-to-end diagnosing support

- Ophthalmology
- Dermatology
- Cardiology



RESEARCH ARTICLE

Can machine-learning improve cardiovascular risk prediction using routine clinical data?

Stephen F. Weng, Jenna Reps, Joe Kai, Jonathan M. Garibaldi, Nadeem Qureshi

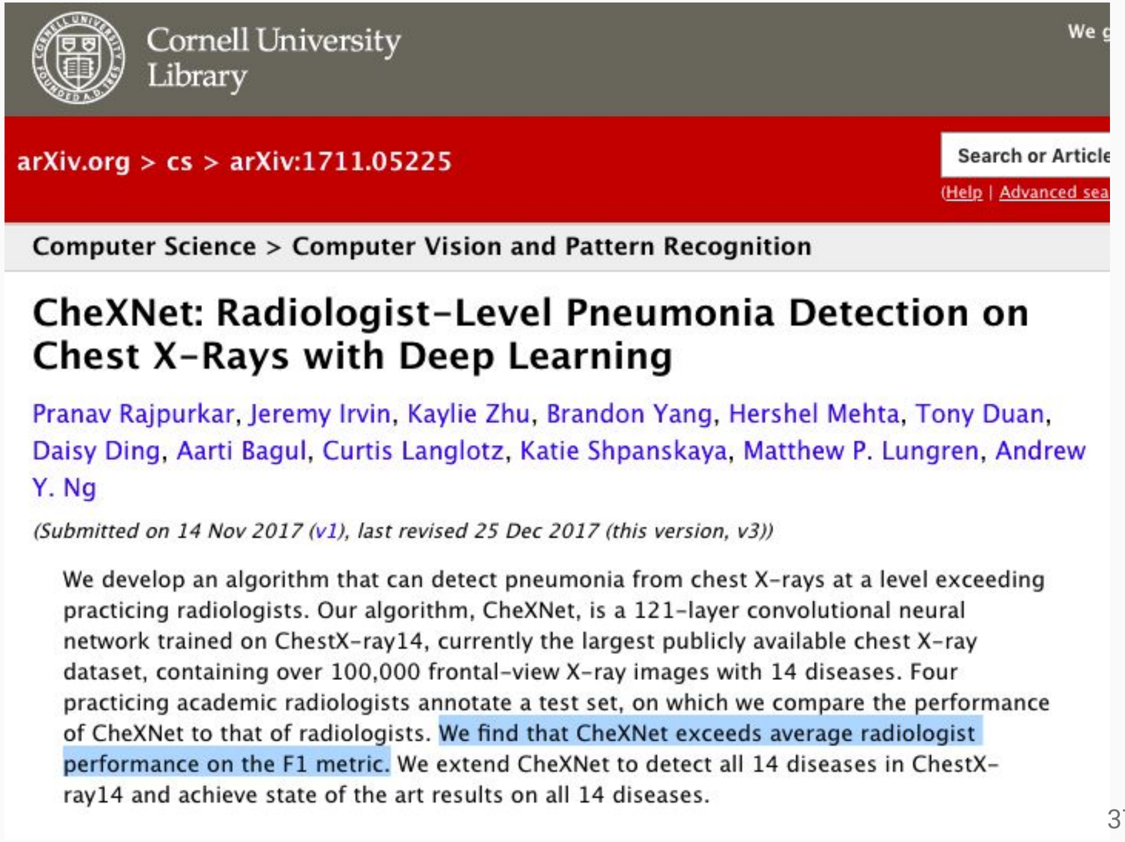
Published: April 4, 2017 • https://doi.org/10.1371/journal.pone.0174944

Conclusions

Machine-learning significantly improves accuracy of cardiovascular risk prediction, increasing the number of patients identified who could benefit from preventive treatment, while avoiding unnecessary treatment of others.

End-to-end diagnosing support

- Ophthalmology
- Dermatology
- Cardiology
- Radiology
- ...



Cornell University Library

We g

arXiv.org > cs > arXiv:1711.05225

Search or Article

(Help | Advanced sea

Computer Science > Computer Vision and Pattern Recognition

CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning

Pranav Rajpurkar, Jeremy Irvin, Kaylie Zhu, Brandon Yang, Hershel Mehta, Tony Duan, Daisy Ding, Aarti Bagul, Curtis Langlotz, Katie Shpanskaya, Matthew P. Lungren, Andrew Y. Ng

(Submitted on 14 Nov 2017 (v1), last revised 25 Dec 2017 (this version, v3))

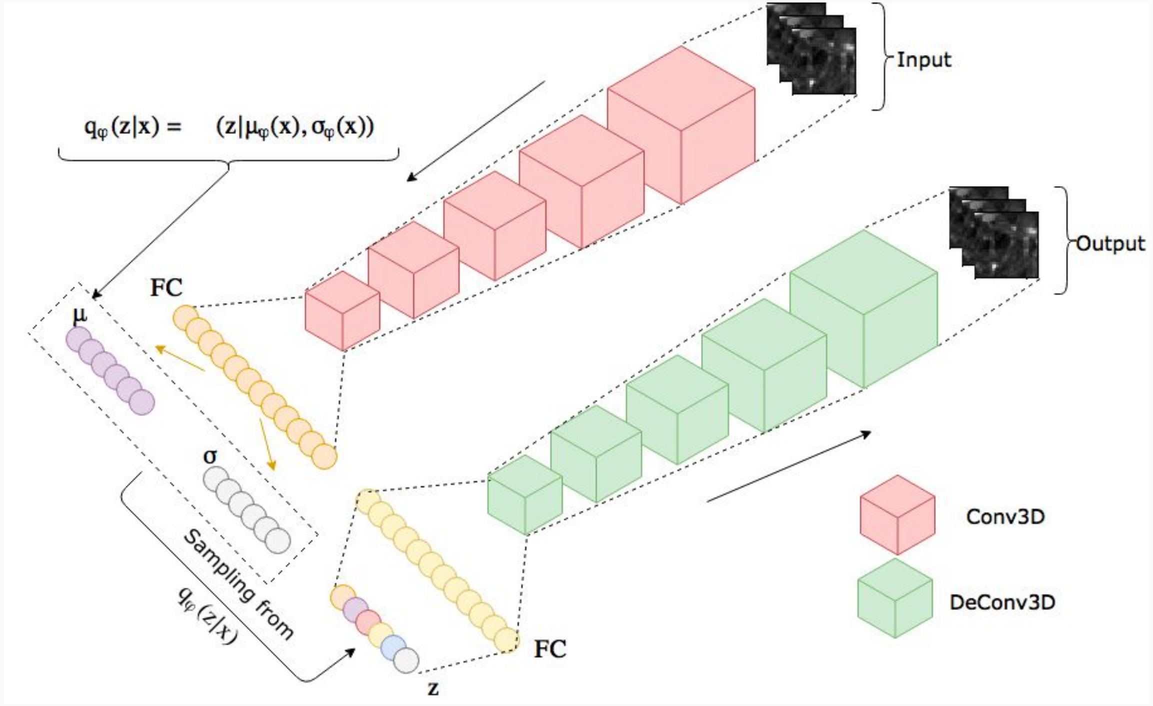
We develop an algorithm that can detect pneumonia from chest X-rays at a level exceeding practicing radiologists. Our algorithm, CheXNet, is a 121-layer convolutional neural network trained on ChestX-ray14, currently the largest publicly available chest X-ray dataset, containing over 100,000 frontal-view X-ray images with 14 diseases. Four practicing academic radiologists annotate a test set, on which we compare the performance of CheXNet to that of radiologists. We find that CheXNet exceeds average radiologist performance on the F1 metric. We extend CheXNet to detect all 14 diseases in ChestX-ray14 and achieve state of the art results on all 14 diseases.

Can we learn without labels?

Can we reason about images without supervision?

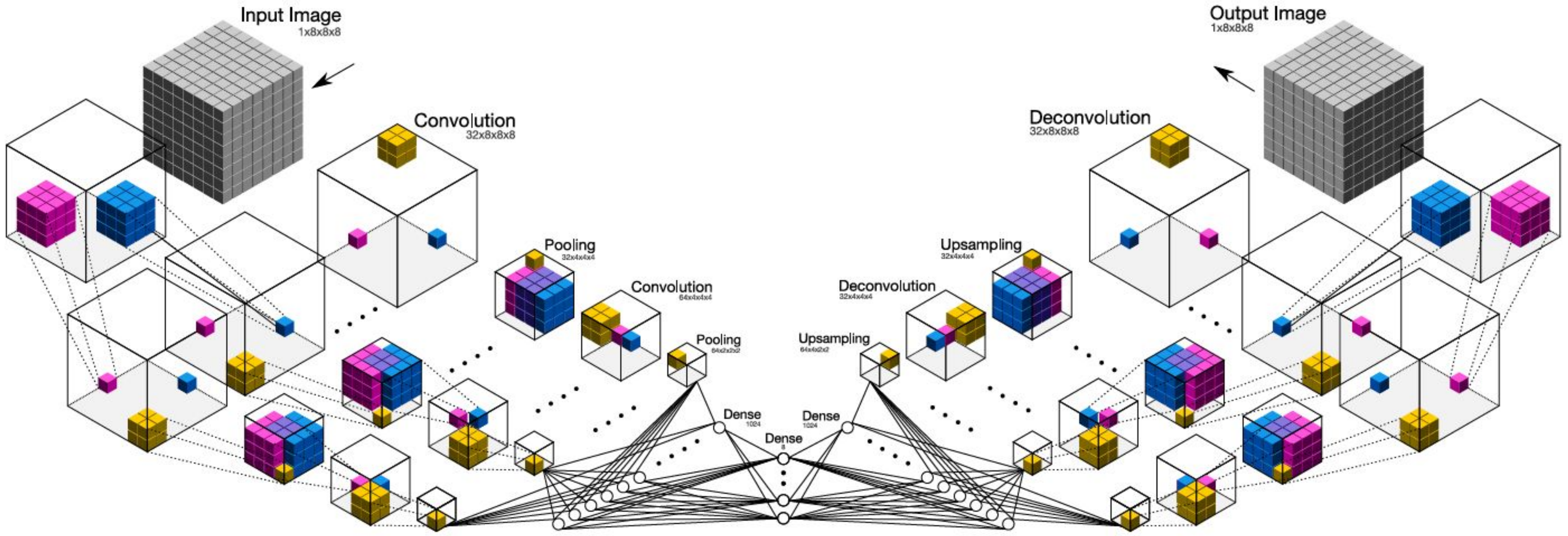
Unsupervised detection of anomalies

Variational Autoencoder



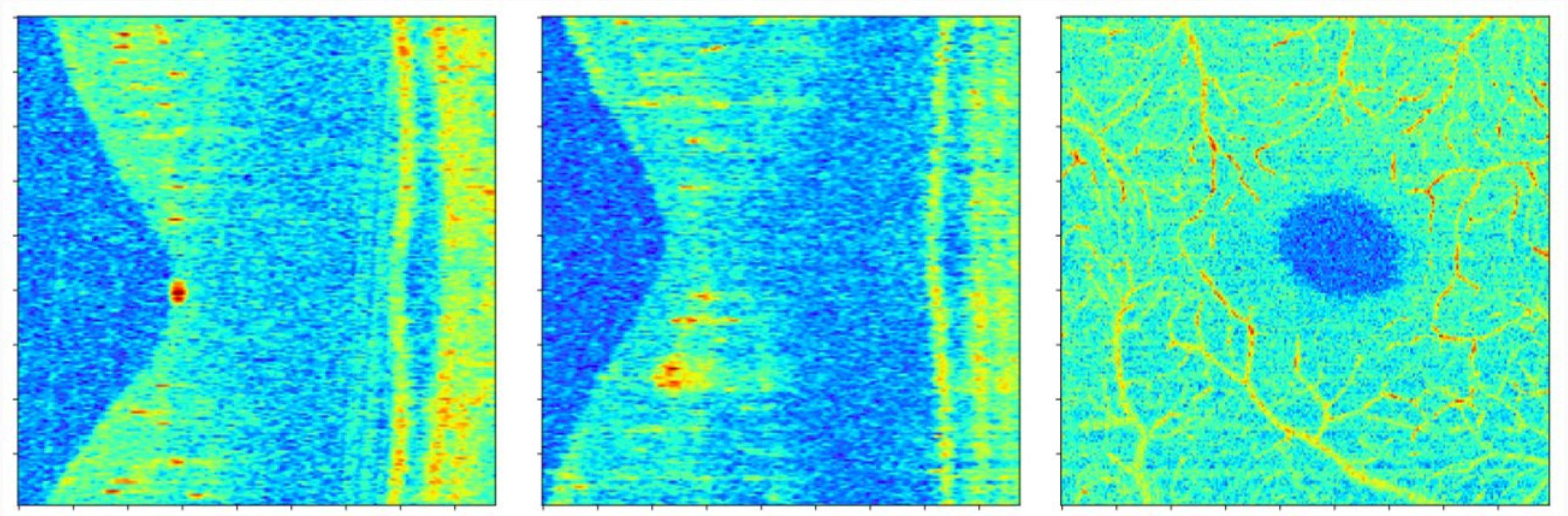
Input
$q_\varphi(z|x) = \quad (z|\mu_\varphi(x), \sigma_\varphi(x))$
FC
μ
σ
Sampling from $q_\varphi(z|x)$
z
FC
Output
Conv3D
DeConv3D

Tuzel, Szymon. “Deep Learning for Anomaly Detection in Volumetric Computer Tomography,” Masters Thesis, Institute of Computing Science, Poznan University of Technology, 2017.



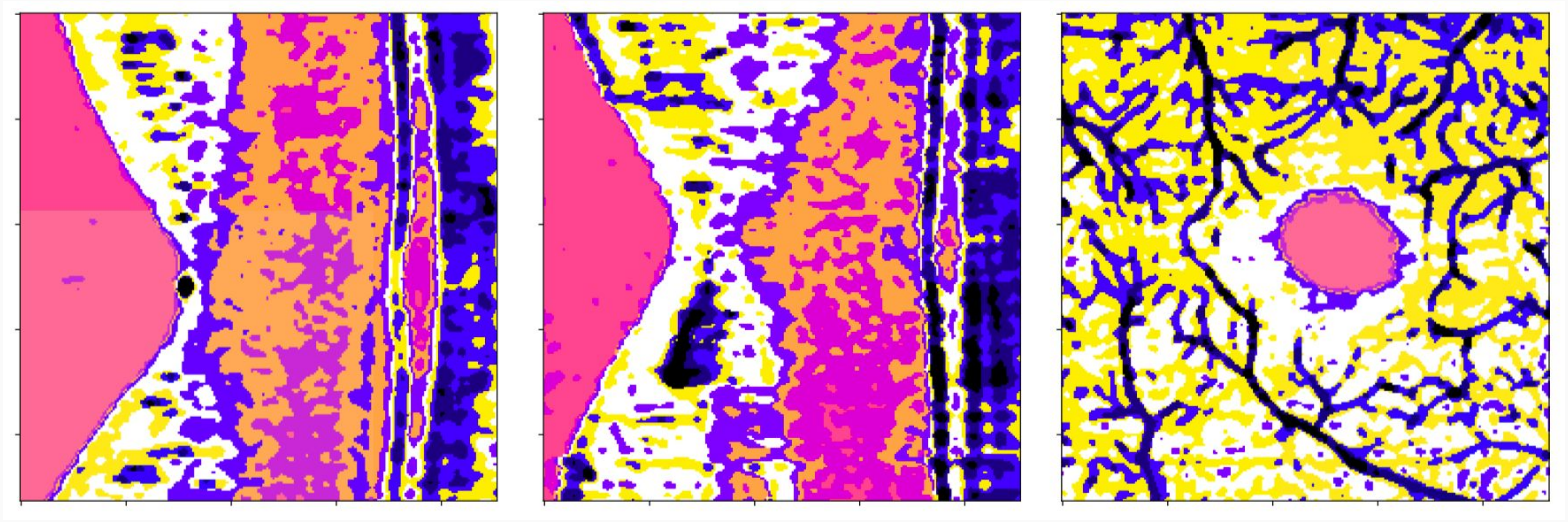
Gliszczyński, Patryk. “Deep Autoencoders for Unsupervised Image Segmentation in Optical Coherence Tomography,” Masters Thesis, Institute of Computing Science, Poznan University of Technology, 2018.

Unsupervised segmentation via autoassociation



Unsupervised segmentation via autoassociation

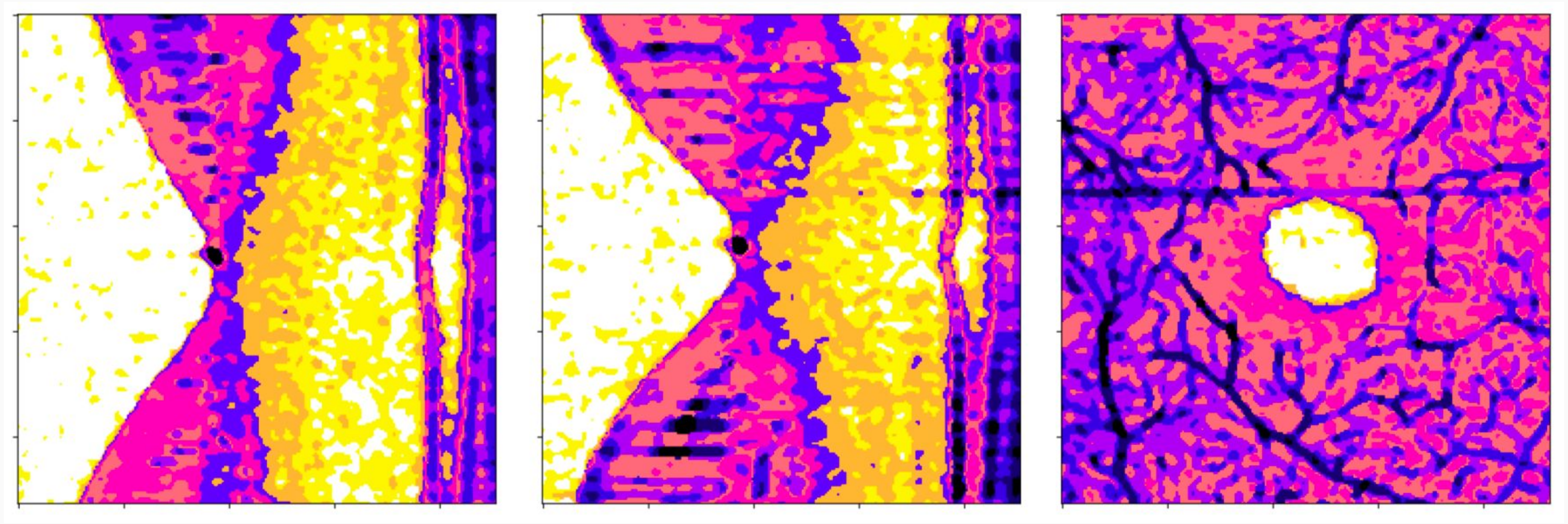
Optical coherence tomography



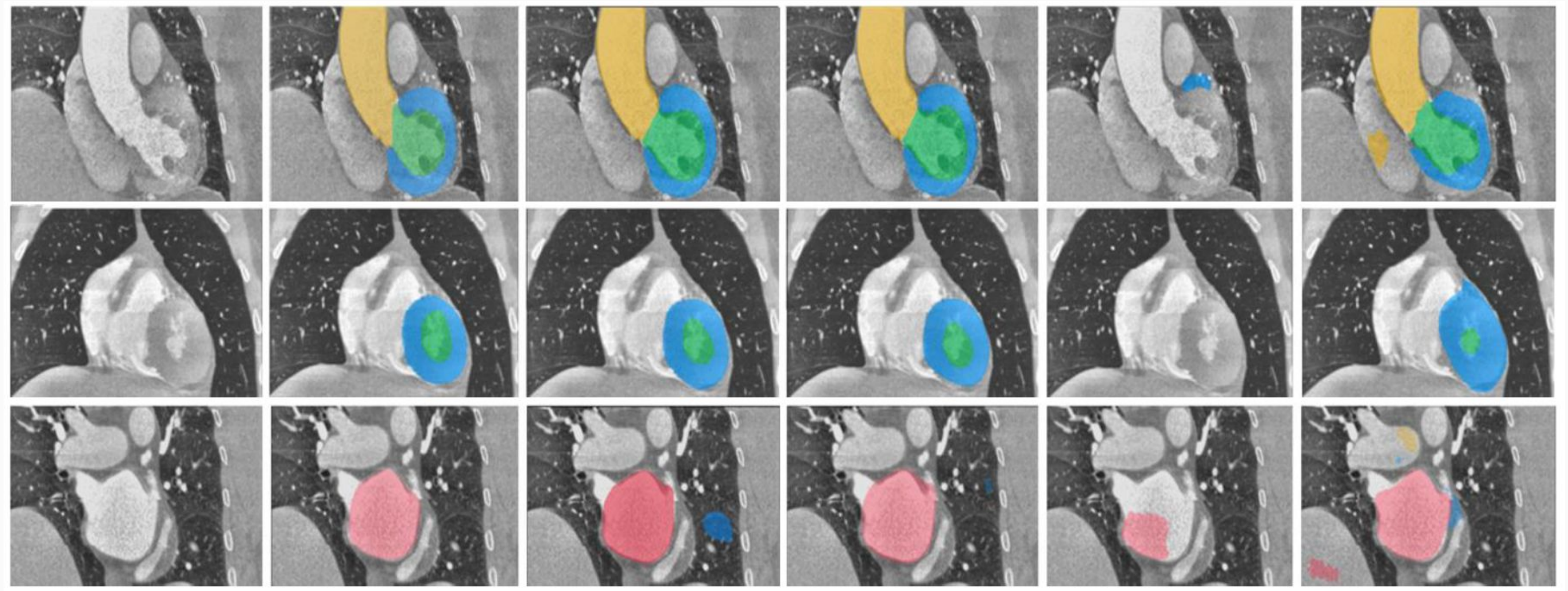
Gliszczyński, Patryk. "Deep Autoencoders for Unsupervised Image Segmentation in Optical Coherence Tomography," Masters Thesis, Institute of Computing Science, Poznan University of Technology, 2018.

Unsupervised segmentation

Optical coherence tomography



Unsupervised segmentation


(a) CT Image (b) CT Label (c) Seg-CT-STL (d) Seg-CT (e) Seg-CT-noDA (f) Seg-CT-UDA

Dou, Qi, Cheng Ouyang, Cheng Chen, Hao Chen, and Pheng-Ann Heng. “Unsupervised Cross-Modality Domain Adaptation of ConvNets for Biomedical Image Segmentations with Adversarial Loss.” ArXiv:1804.10916 [Cs], April 29, 2018. http://arxiv.org/abs/1804.10916.

A sample of capabilities

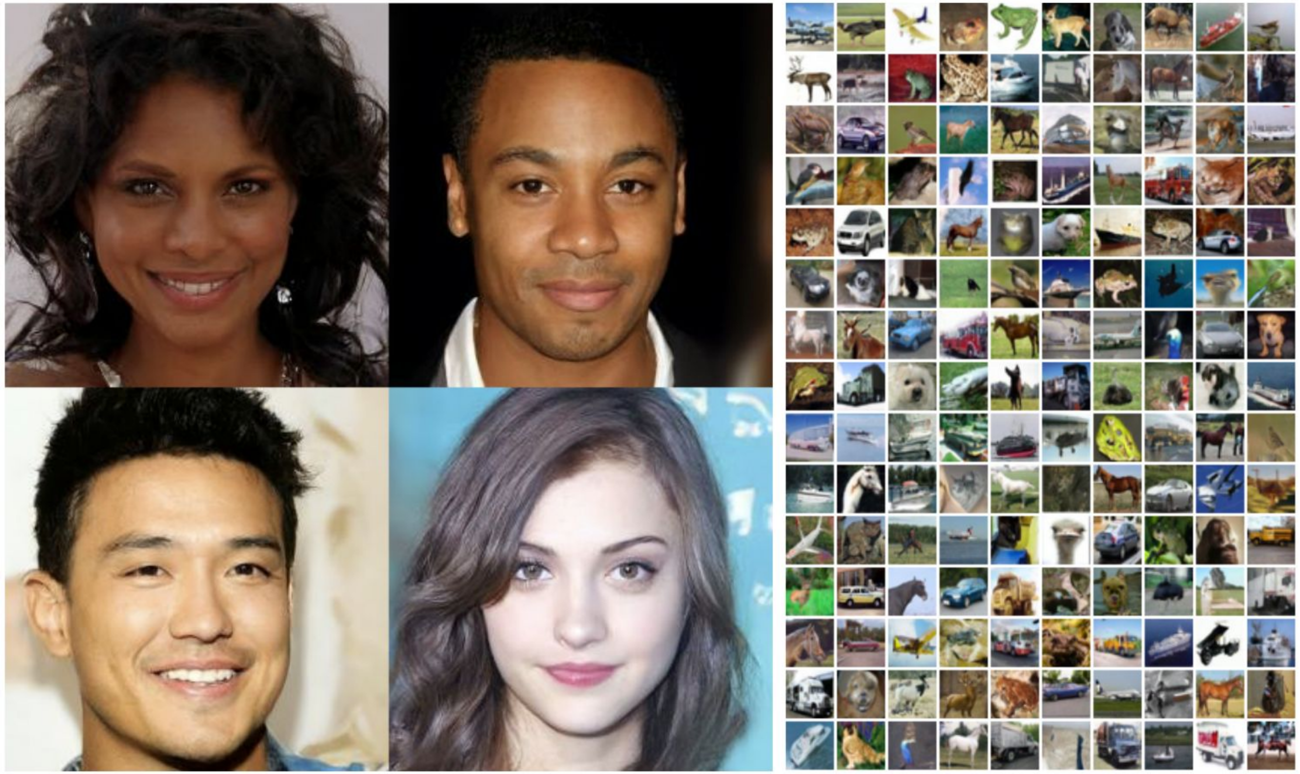


Figure 1: Generated samples on CelebA-HQ 256 × 256 (left) and unconditional CIFAR10 (right)

J. Ho, A. Jain, and P. Abbeel, Denoising Diffusion Probabilistic Models, 2020. http://arxiv.org/abs/2006.11239

16
AMMO
94%
HEALTH
ARMS
0%
ARMOR

Software libraries and tools

Programming language: **Python**

- (though notice the usefulness of C/C++ implementations in some contexts)

Libraries:

- OpenCV, Numpy, Scikit
- Tensorflow+Keras

Other tools:

- ImageJ and Fiji https://imagej.net/Fiji
- ImageMagick
- Jupyter notebooks

Part 2:
“How part”:
AI and Machine Learning for CV

Introduction to neural networks

Module outline

What is a neural network?

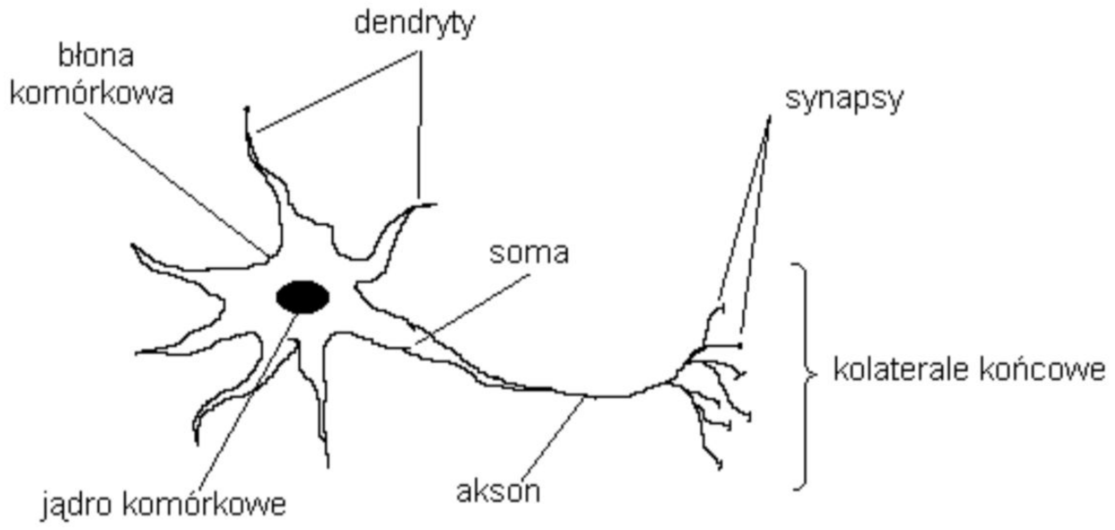
What is [an artificial] neural network?

Definition: A number of interconnected, relatively simple elementary processing units ('neurons').

- The expressive power stems from the architecture and the number of entities, not from their individual capabilities.
  - The 'natural' way of making a NN model more powerful is to equip it with more units.
- An alternative model of computation (bio-inspired, but very loosely).
- A machine learning model, which can be trained and queried.
- Model querying:
  - Input data fed into network (input units).
  - The data ('signals') propagate through the architecture
  - Collecting model's response from the output units.

Biological inspirations for ANNs

- More essential in the early days of the field, now largely forgotten.
  - (or found largely irrelevant).
- ANNs' units are very very crude models of biological neurons.



Krzysztof Krawiec Jerzy Stefanowski. Uczenie maszynowe i sieci neuronowe. Politechnika Poznańska. Wydawnictwo, 2003, 2004

A bit of history

- Beginnings: 1940s (McCulloch and Pitts, 1943)
- **First spring**: Early days of AI
  - Perceptron: (Rosenblatt, 1958)
  - Hopfield networks
- **Second spring**: late 1980s – mid 1990s
  - Parallel Distributed Processing: Explorations in the Microstructure of Cognition, by Rumelhart, McClelland, and PDP Research Group, 1986,
- Then: Neural network winter
  - Saturation of capabilities combined with insufficient performance on many real-world tasks.
  - Growing popularity and performance of other ML methods (support vector machines (SVM), random forests, Bayesian models, etc.)
- **Third spring**: Big come-back: ~2005 and on
  - New wave: Deep Learning
  - Facilitated by conceptual developments, growing affordability of computing power and increasing availability of data.

Key figures

- Geoffrey Hinton
- Yann LeCun
- Juergen Schmidhuber
- Yoshua Bengio
- Michael Jordan
- Andrew Ng
- Ian Goodfellow
- Daphne Koller
- Andrej Karpathy
- Fei-Fei Li
- Sebastian Thrun
- Sepp Hochreiter
- …

Fathers of the Deep Learning Revolution Receive ACM A.M. Turing Award

Bengio, Hinton and LeCun Ushered in Major Breakthroughs in Artificial Intelligence

https://awards.acm.org/about/2018-turing

Current state of the domain

- Currently: one of the most successful paradigms of AI and machine learning (ML), particularly in:
  - image analysis, pattern recognition, computer vision,
  - natural language processing,
  - generative models.
- Brought the capability to handle (simulate, train, query) large, deep (many-layers) networks.
- Rich branch of research on the verge of AI, ML, and other disciplines (e.g., cognitive sciences)
- A broad range of approaches and architectures: probabilistic, evolutionary, spiking, counterpropagation, discrete, bidirectional, ...
- Handles surprisingly well different data representations, both:
  - fixed-size: vectors, matrices, images, tensors,
  - variable-size: sequences, trees, graphs, text, …

Units and layers

[Artificial] neuron (unit)

Linear unit: an aggregating function, typically a weighted sum of inputs:

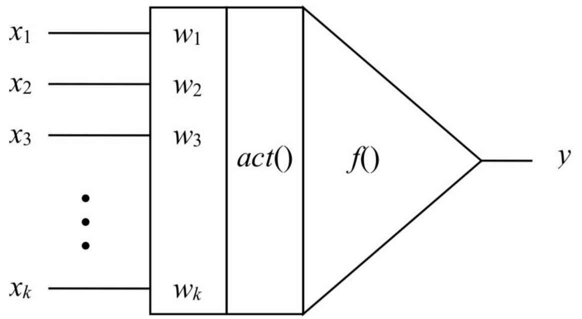
$$y = \sum_{i=1}^{n} w_i x_i + b$$

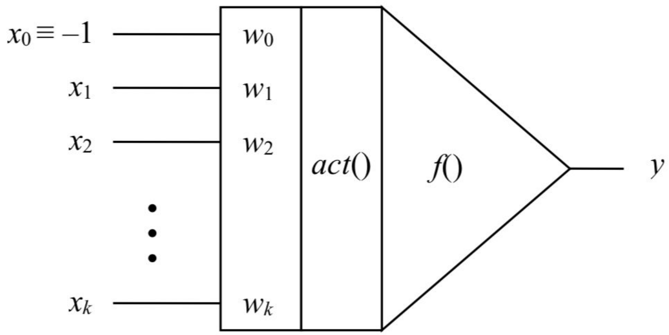
where w_i – weights, b – bias

Convenient reformulation:

$$y = \sum_{i=0}^{n} w_i x_i$$

where x_0 is assumed to be constant (typically 1 or -1).





Units: comments

- Weighted sum of inputs sometimes referred to as *excitation* or *activation*.
    - Warning: the latter is often conflated with the output of a unit (after applying activation function).
- Weights = parameters.
- Number of inputs = a hyperparameter.
- Typically implemented in a stateless (memory-less) manner.
- Implements a *dychotomizer*: divides the space of inputs into the positive and the negative half-space.

Implication of linearity

- There is no point in composing linear units: their composition is a linear unit (weighted sum of weighted sums).
- Nevertheless, occasionally used in contemporary architectures to reduce computational costs or for other purposes (to be shown later).

Interlude: other types of excitations

More precisely: other ways of defining unit's excitation.

Radial unit: activation determined by a radial function, e.g.

$$||x - w||_2$$

- Excitation captures the similarity of the input vector *x* to the weight vector *w*.
- Used in Radial Basis Function models (RBFs),
  - An architecture designed for supervised or unsupervised clustering.
- Relatively rare nowadays.

Nonlinear unit

The weighted sum of inputs (‘activation’) is passed through a form of nonlinearity:

$$y = \sigma(\sum_{i=0}^{n} w_i x_i)$$

where σ is an activation function.

Composition of nonlinear units is not redundant anymore (compared to linear units): it increases expressibility.

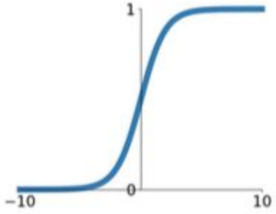
Types of nonlinearities (activation functions) used in ANNs:

- Bounded codomain: typically S-shaped, squeezing functions :
  - Bipolar: tanh, codomain [-1; 1]
  - Unipolar: sigmoid function, codomain [0; 1]
- Unbounded codomain: e.g., Rectified Linear Unit (ReLU): relu(x) = max(x; 0)

Activation functions

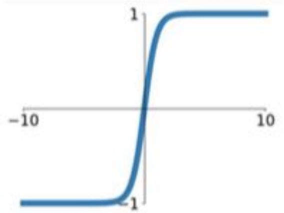
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



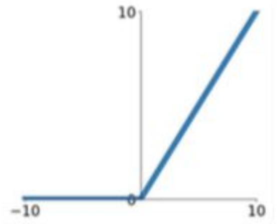
tanh

$\tanh(x)$



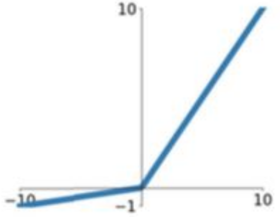
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

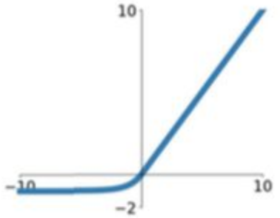


Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



More about that later.

Layers

- In the ‘second spring’ of neural networks (late 1990s), the discourse was (still) conducted mostly on the level of individual <u>units</u> ([artificial] neurons).
- However:
  - NN and DL are paradigms for highly distributed, parallel information processing, so individual units are rarely critical.
  - An individual unit can be considered a degenerate layer (layer of size 1).
- Therefore, today we tend to talk in terms of <u>layers</u>.
  - Units in a layer typically operate in parallel and perform the same computation.
  - They may be differently parameterized, but don’t have to (see: weight sharing). Examples:
    - Dense layer: units have the same arity, but separate vectors of parameters.
    - Convolutional layer: units have the same arity (and also ‘shape’) and share parameters.

Layers

A natural abstraction of the concept of unit: a layer.

- Neurons typically combined into layers (‘vectors’ of units), because:
  - A single unit can produce only scalar output.
  - Vector outputs are often necessary.
- The simplest layers are one-dimensional, but sometimes we endow them with some higher-dimensional topology:
  - matrix (2D)
  - tensor (nD, $n > 2$)
- In practice, layer is the smallest unit of discourse in contemporary deep learning.
  - Note that a single unit can be considered a degenerate layer.

Layers vs. units

Convenient correspondence with linear algebra; for instance, a dense linear layer with n inputs (x) and m outputs (y) implements matrix multiplication:

$$y = Wx$$

The elements w_{ij} of the matrix is the jth weight of the ith unit in the layer.

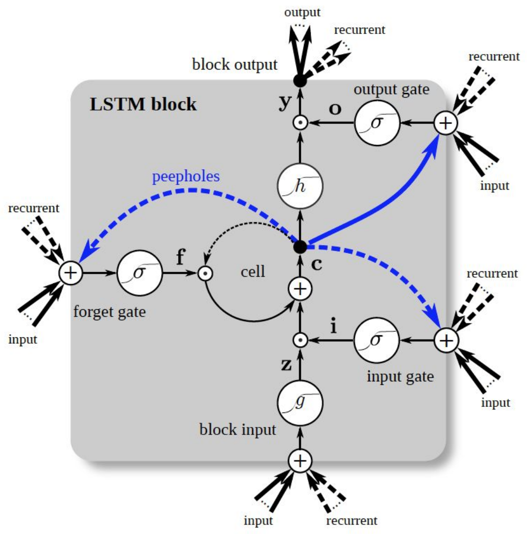
Layers vs. units

In most cases, a layer is just an ‘arrangement’ of units:

- Dense layer: a 1D ‘vector’ of units.
- Convolutional layer: a 2D ‘array’ of units.

However, some layers have more complex internal architecture:

- They encapsulate a smaller computation graph, and so hide complexity.
- A form of modularization.
- Example: Recurrent architectures, e.g. LSTM (Long Short-Term Memory unit/cell)



Computational capabilities of neural networks

Important features of the neural computing paradigm

- Continuous
  - Opens the door to the use of whole lot of maths (algebra, calculus, …)
  - Does not preclude handling discrete variables (via, e.g., output thresholding)
  - Certain properties are formally provable (see next slide, for instance).
- Distributed, and hence:
  - Easily parallelizable (even in training phase, see federated learning).
  - Robust to local failures (kind of, mostly when implemented in hardware)
- Non-symbolic (sometimes referred to as subsymbolic)
  - However, neurosymbolic systems start becoming popular in recent years.

ANNs are universal approximators (Cybenko, 1989)

Let φ be a nonconstant, bounded, and monotonically-increasing continuous function. Let I_m denote the m-dimensional unit hypercube $[0,1]^m$, and $C(I_m)$ denote the space of continuous functions on I_m.
Then, given any ε > 0 and any function $f \in C(I_m)$, there exists an integer N, real constants $v_i, b_i \in \mathbb{R}$ and real vectors $w_i \in \mathbb{R}^m$, where i = 1, …, N, such that we may define

$$F(x) = \sum_{i=1}^{N} v_i \phi(w_i^T x + b_i)$$

as an approximate realization of the function f, where f is independent of φ; that is,

$$|F(x) - f(x)| < \epsilon$$

for all $x \in I_m$.

Cybenko, G. (1989). "Approximation by superpositions of a sigmoidal function". Mathematics of Control, Signals, and Systems. 2 (4): 303–314.

ANNs are universal approximators (Cybenko, 1989)

Summary: a linear combination of monotonic nonlinearities (e.g., an ANN with one sigmoid layer followed by a linear layer) can approximate any bounded continuous function f arbitrarily well.

Practical implication:

- In principle, any function meeting the assumptions of the theorem can be approximated by a 2-layer network (linear combination of N nonlinear units).
- Approximation error is only function of N, and can be arbitrarily reduced by increasing the number of units.

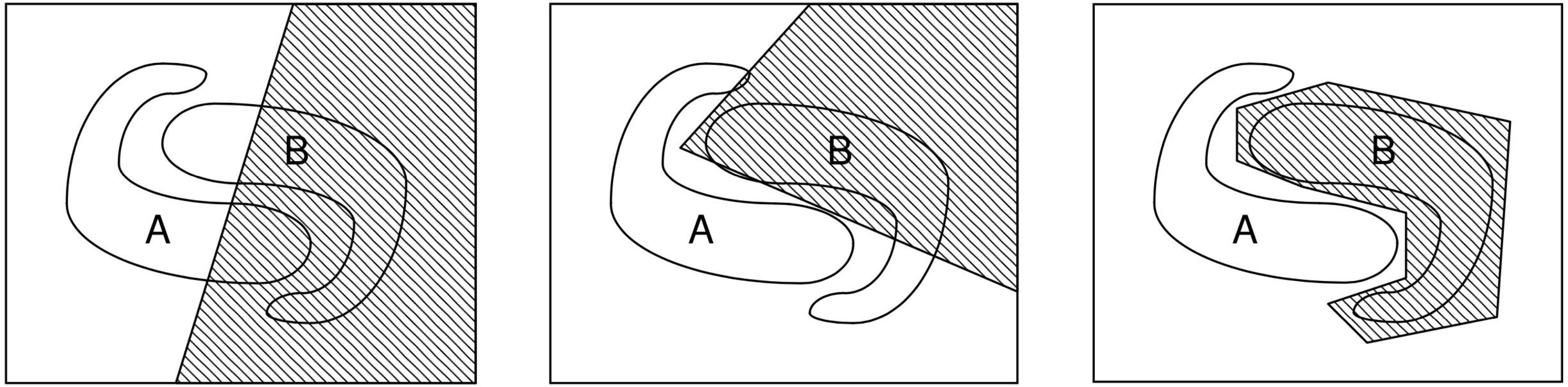
However:

- Cybenko's theorem confirms the *existence* of such approximation. *Finding* N and w_is is an open problem (intractable in general, see later).
- In practice, f is not given – we know only a *sample* of its behavior, i.e. the training set.

A note on expressibility

Universal approximation with Cybenko's model may require unpractical number of units N. In practice, we boost the expressive capabilities of ANNs by increasing the number of layers. One of the possible/popular interpretations:

- A single unit partitions the input space into two half-spaces.
- Two layers can 'carve out' an arbitrary convex polyhedron in the input space.
- A third layer can combine multiple convex regions into an arbitrary shape.
- Single units may approximate logical operators: and, or, nor, ...



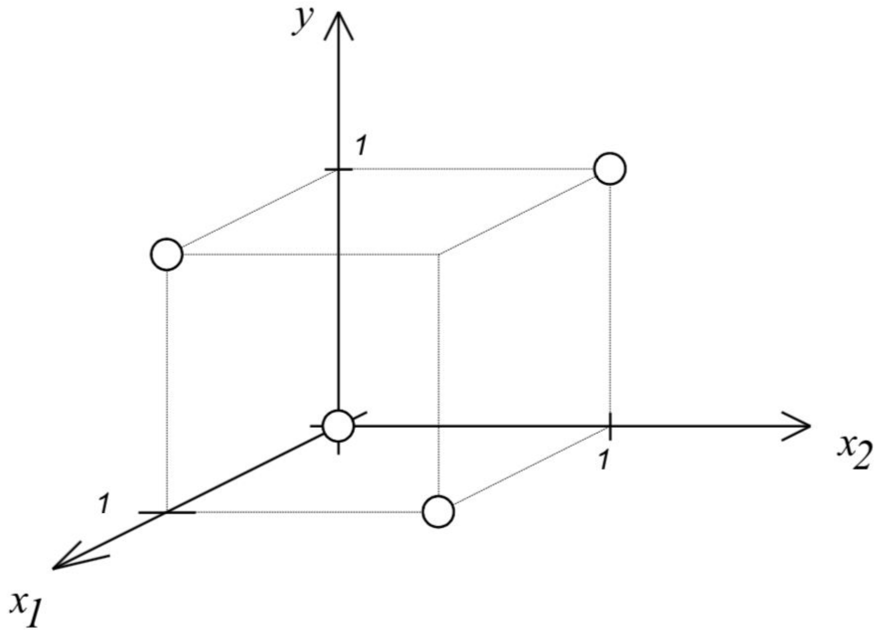
The need for composition of nonlinearities

Even apparently simple problems are not linearly separable.

Example: the XOR problem.

Truth table (exhaustive ‘training set’):

x1	x2	y=XOR(x1,x2)
0	0	0
0	1	1
1	0	1
1	1	0



Desired behavior of the model (in the classification sense, i.e. producing the output with the correct sign) cannot be realized with a single layer.

Layers can be arranged in arbitrary architectures

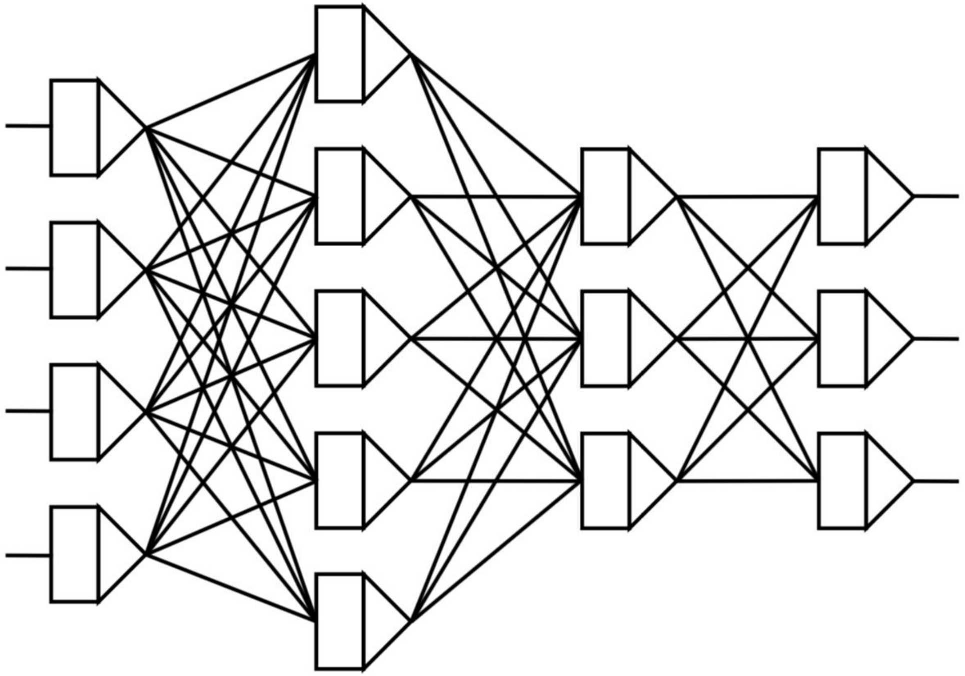
- Main line of division:
  - Feed-forwarded
  - Feed-back (e.g. recurrent, but not only those)
- Implications:
  - The former are stateless, the latter stateful.
  - Both types used in contemporary models.
- Note: Some architectures do not feature explicitly defined inputs/outputs
  - E.g., the Hopfield network is a clique of units, each collecting signals from all units, including itself).

Next slides: A few glimpses on some of the architectures presented in this course.

Fully connected (FC), feed-forward architecture

Each unit from a given layer receives signals from all units in the previous layer.

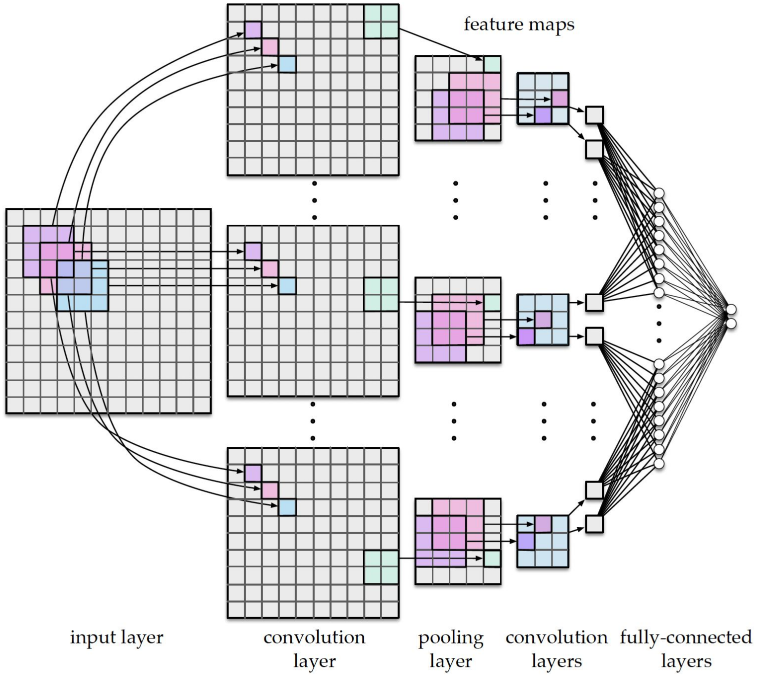
- No connections to preceding layers.



Mixed convolutional+FC architecture

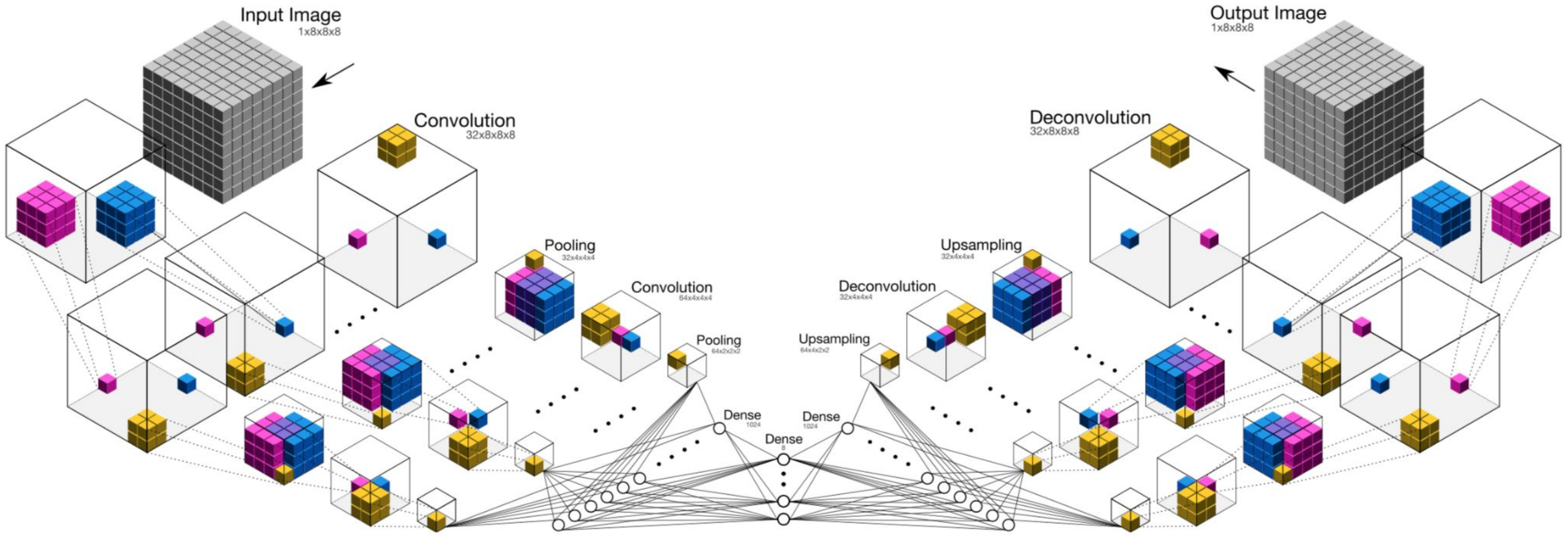
Fully connected, a.k.a. dense.

The first architecture of choice for problems of object/image recognition.



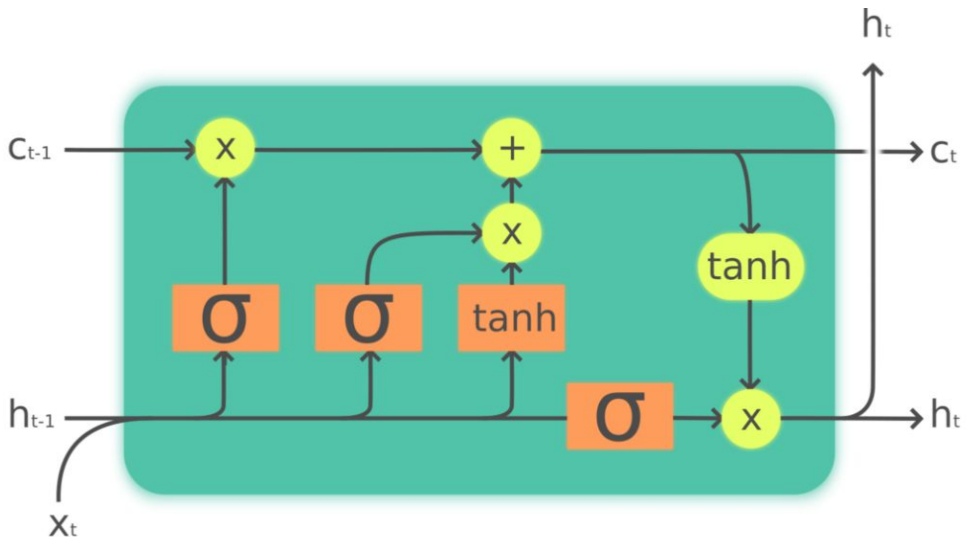
Convolutional autoencoder (3-dimensional)

- Formally a special case of ConvNet (convolutions + dense layers).
- Trained via autoassociation, i.e. reproduce the input.



Recurrent cell

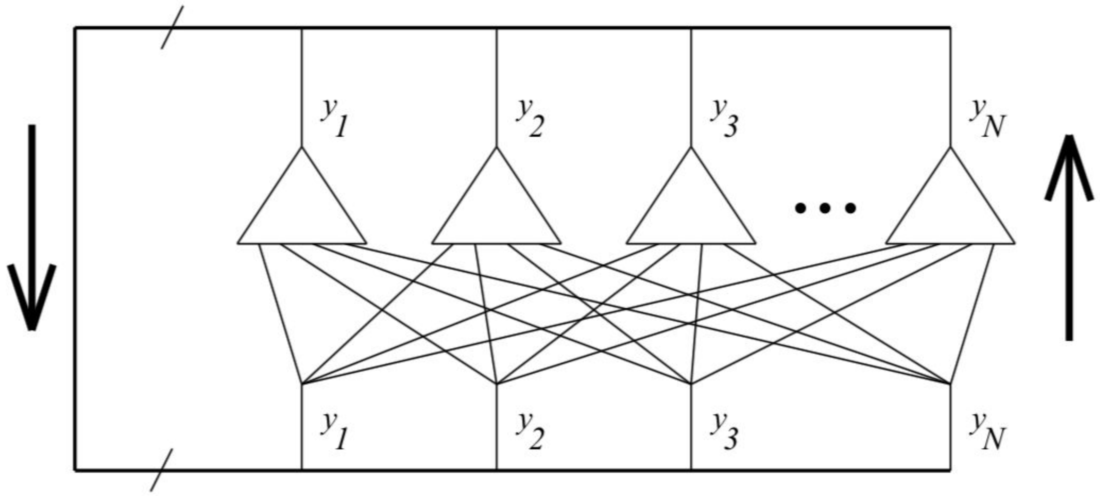
- Long short-term memory model (LSTM)
- Note: recurrence not explicitly visible in the diagram.
- The diagram shows only processing for a single time step.



Fully connected graph (clique) architecture

Each unit receives inputs from the outputs of all other units.

- Feedbacks possible
- Technically: a recurrent model (though mainly of theoretical importance)
- Notable example: Hopfield network



Hopfield, J. J. (1982). "Neural networks and physical systems with emergent collective computational abilities". Proceedings of the National Academy of Sciences. 79 (8): 2554–2558.

Introduction to Convolutional Neural Networks

NN training in a nutshell

Model training/fitting = optimization of network parameters.

Neural networks are a special case of a machine learning model - hence the term 'model' appearing in the following slides.

Inputs:

1. A training set in the form of pairs (x,y), where
   a. The x is the input to the model,
   b. y is the desired (expected) output corresponding to x,
2. A network architecture f, compatible with types of x and y
   a. Parameters initialized by some method, usually small signed values (e.g., in [-0.1,0.1]).
   b. Note: the distribution of initial weights can have a critical impact on the efficiency of the learning process (e.g., different activation functions 'prefer' different distributions).
3. A loss function L that quantifies how the model diverges from the desired behavior exemplified by the training set.

NN training in a nutshell

Typical training loop:

1. Query the model on an example x, obtaining the output of the model y'=f(x)
2. Calculate the loss function L(y,y') and its gradient w.r.t. model parameters.
3. Correct the model parameters guided by the gradient descent (*gradient descent*)

   $$w \leftarrow w - \eta \frac{\partial L}{\partial w}$$

   where w is the vector of model parameters and η is the learning rate hyperparameter.
4. If no stopping conditions are met, jump to 1.

- Known as error backpropagation algorithm and stochastic gradient descent.
- Gradient calculated analytically with specialized automatic differentiation (autodiff) algorithms backed by libraries (Pytorch, TensorFlow, JAX, …).

NN training in a nutshell: Technical comments

- Gradient descent is a heuristic – the gradient vector does not necessarily point toward the global minimum of the loss function.
    - Training may get stuck in a local optimum.
    - However, local optima become less likely when the number of parameters is large.
- Gradient descent is deterministic.
    - However, the starting point (initial parameters) is drawn at random.
    - The order of presenting examples is often randomized too.
    - Advanced low-level implementations can dynamically reschedule operations, leading to indeterminism.
        - Recall that floating-point addition and multiplication is not associative: $(a+b)+c \neq a+(b+c)$
- Examples often given in *batches* (*minibatches*).
- Signals represented as *vectors*.
- Model parameters stored as *matrices*.
- Both vectors and matrices implemented technically as *tensors*.

The concept of tensor

In modern deep learning, operations (as well as their implementations) are often defined for an arbitrary number of dimensions:

- 1-dimensional (1D, e.g. time)
- 2-dimensional (2D, height x width)
- 3-dimensional (3D, height x width x depth)
- 4-dimensional (4D, height x width x depth x time)
- etc.

Hence, the prevailing assumption is that the data processed by the individual components are represented as *tensors*, or multidimensional arrays.

The concept of tensor

- A 2D raster image is a 2- or 3-dimensional tensor (3-dimensional if we assume a separate dimension for the channels, which is the norm).
    - The input images and tensors transmitted between layers in network are technically no different from each other, which is convenient.
- For 2D images, convolutional layers (or entire models) work on batches represented as 4-dimensional arrays (tensors) of dimensions:
    - N: batch (example number in the package)
    - H: height (height, Y coordinate)
    - W: width (width, X coordinate)
    - C: channel (channel, depth).
- The two most common technical conventions: NHWC (channels last) and NCHW (channels first)
    - NHWC often required, and more efficient in hardware implementations (e.g., Tensor Cores)

Examples vs. receptive fields

- In Computer Vision, an *example* is usually an *image* (example = image)
- However, from the unit's point of view, a single example is a fragment of an image visible in its receptive field.
    - Thus, even within a single image, convolutional units learn simultaneously from multiple receptive fields.
    - Each image conveys multiple examples.
    - This makes learning of convolutional layers more efficient (in the sense of 'data-efficient'*) and less prone to overfitting.

*Data efficiency refers to the ability of a machine learning method to produce an effective model with a limited volume of learning data (usually measured by the number of examples).

- More technically: the number of examples required to construct a well-generalizing model.

Why neural networks for computer vision?

Motivations (1)

Motivations for using NNs/DL in image analysis and computer vision:

1. Many of the operations used in image analysis are differentiable (e.g., convolution), and therefore can be parameterized with gradient algorithms typically used for NNs.
2. Many of the operations used in image analysis are local (based on a local window of the image), and thus naturally parallelizable, which coincides with the parallel/distributed information processing model typical for NNs (multiple independent neurons/units).
3. We represent raster images as vectors, matrices and tensors, the processing of which in NNs is very natural.

Motivations (2)

Motivations for using NNs/DL in image analysis and computer vision:

1. Opens the door to the rich repertoire of resources of modern machine learning and deep learning, including architectures, training algorithms, etc.
2. Neural networks (and machine learning more broadly) address one of the fundamental challenges of CV: the difficulty of designing and parameterizing efficient algorithms/systems that solve specific tasks.
    a. DL facilitates semi-automation of that process.
3. Ease of merging subnetworks/components with others, within a single paradigm and technology, facilitating production of end-to-end solutions.
    a. Also: bridging with other data modalities, e.g. automated *image captioning* (see next slide).

Types of architectures

Two main categories (from the point of view of image analysis):

1. Fully convolutional neural networks (CNNs, ConvNets)

- Consist only of convolutional layers and other layers that preserve the locality of processing (such as maxpooling; more later).
- [It would be more appropriate to call them *spatially local* architectures]

2. Mixed architectures

- On top of layers that process image locally, they feature other components, most often dense (fully connected) layers or other components.

See next 2 slides for details.

Fully convolutional architectures (CNNs, ConvNets)

Typical applications:

- *Image de-noising* (*image denoising*)
- Feature detection
  - E.g. feature detectors learned using generative adversarial networks (GAN) models
- Style transfer: transformation of an original image into a preset aesthetic/artistic style
  - E.g., the well-known Pix2pix model
- Image segmentation, including
  - Semantic *segmentation*
  - [*object*] *instance segmentation*
- Describing images in natural language (*image captioning*)

The list of applications and usage scenarios grows longer every year.

Mixed architectures

Common applications:

- Classification
  - Usually a stack of *convolutional* layers (*convolutional stack*), followed by a dense subnet, aggregating information across the image.
- Regression
  - E.g., assessment of image quality (e.g. in medical applications).
- Generating images (generative models, e.g., generative adversarial networks, GANs)
- Combining with deep natural language processing (NLP) models, e.g., in terms of describing images (*image captioning*)
  - Often involves recurrent subnetworks, such as LSTM (Long Short-Term Memory) or GRU (Gated Recurrent Unit) models.
  - Or transformer-type models (e.g., BERT).

The convolution operation

Recall the definition of one-dimensional convolution:

$$g(x) = (f * M)(x) = \sum_{h=-\left\lfloor \frac{k}{2} \right\rfloor}^{\left\lfloor \frac{k}{2} \right\rfloor} M(h) f(x-h)$$

where f is the image, M a mask (filter, kernel) of dimensions kxk.

In traditional image analysis, it is assumed that M is a given, i.e. it was designed to achieve a specific goal/effect, e.g.

- Gaussian mask (normal distribution) purpose of de-noising,
- Sobel or Schar filter for edge detection,
- Laplasian to detect transitions through zero of the second derivative.

In neural approaches, M becomes part of the model and is subject to automatic optimization (learning).

Convolution versus convolution: Differences

In the following slides, we summarize the most significant differences between the convolution operation used in traditional image processing and the convolutional layers used in neural networks.

- Illustrating these differences will help us better understand the 'added value' offered by neural models.

Differences (1)

As in typical layers in neural networks, the outputs of units in convolution layers are passed through a nonlinear activation function:

$$g(x) = act\left(\sum_{h=-\left\lfloor \frac{k}{2} \right\rfloor}^{\left\lfloor \frac{k}{2} \right\rfloor} M(h)f(x-h)\right)$$

This is necessary because the output of a convolutional layer is usually passed to the inputs of the next layer (convolutional or not), and the units in that layer realize the scalar product (dot product, weighted sum) of the inputs. A direct connection to the next layer would be a composite of two linear operations, and this would miss the point, since it could be replaced by a single convolution.

- Typical activation functions are s-shaped (squeezing functions), or the popular Rectified Linear Unit (ReLU), i.e. act(v) = max(v, 0).
- With new advances in deep learning, we can construct and effectively train models with dozens of convolutional layers.

Differences (2)

Like typical units in neural networks, convolutional units in convolutional layers are equipped with a threshold (bias, offset, free parameter) b, which is also subject to learning. That is, a convolution in DL is *de facto* defined as:

$$g(x) = \sum_{h=-\left\lfloor \frac{k}{2} \right\rfloor}^{\left\lfloor \frac{k}{2} \right\rfloor} M(h) f(x-h) + b$$

This is essential because:

- We do not know in advance whether the input signals are zero-centered (i.e. whether the average f=0), especially when the inputs to the layer come from earlier layers in the architecture
    - In the presence of a bias (lack of zero-centering), the algorithm can tune b to compensate for that.

Differences (3)

In traditional image analysis, we usually use single convolutions. In neural networks, almost always:

1. We apply convolution to multiple input channels, summing the result of the convolutions:

$$g(x) = \sum_c \sum_{h=-\left\lfloor \frac{m}{2} \right\rfloor}^{\left\lfloor \frac{m}{2} \right\rfloor} M_c(h) f_c(x-h)$$

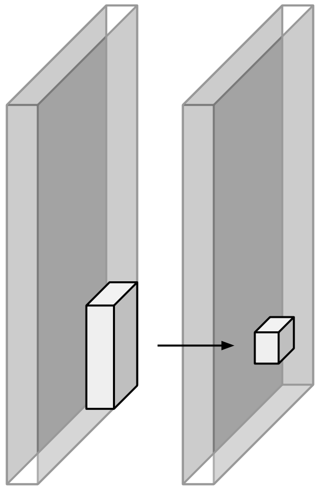
   where M_c is the mask for channel c and f_c is the channel c of the input image.
2. We conduct many such multi-channel convolutions in parallel.

So a convolutional layer accepts an n-channel image at the input and produces an m-channel image at the output.

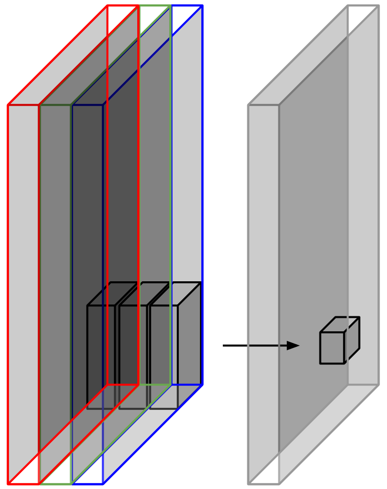
- Each output channel is a linear combination (before the activation function) of the input channels ⇒ a convolutional layer can 'mix' channels.
- The raster associated with one output channel is known as a feature map.
- The number of channels: depth [tensor] (depth).

Illustration

Traditional convolution of a single-channel (scalar) signal (image)

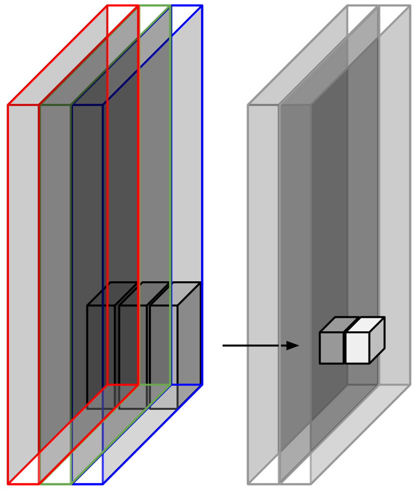


A convolutional layer applied to a 3-channel image: the unit aggregates the input channels into a single output channel.

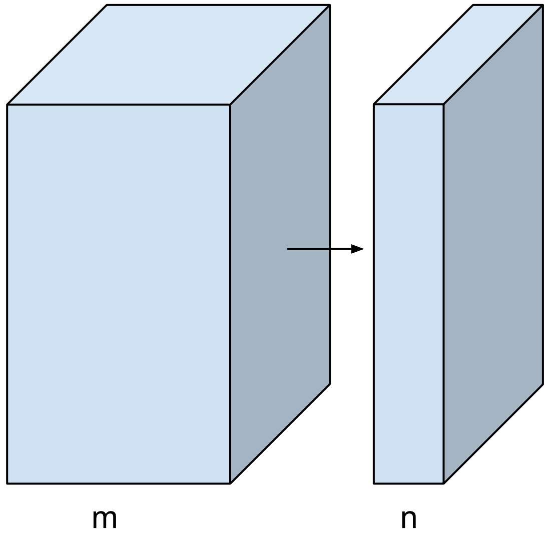


Illustration

General operator: applying a convolution to an m-channel image (tensor) results in an n-channel tensor (here m=3, n=2)



Any m and n



Note that this case should not be equated with a three-dimensional convolution: m and n are constant.

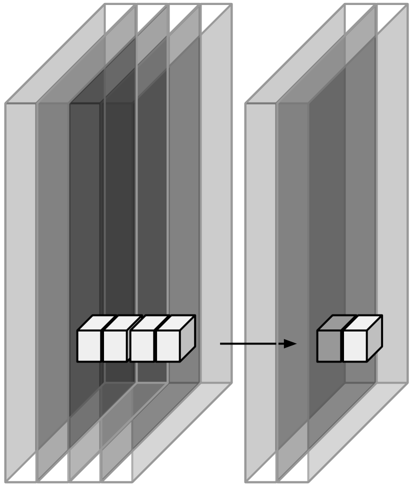
Differences (4)

Implication of multichannel processing (previous slide): it becomes reasonable to use convolution with a 1x1 mask/kernel.

- Such a mask does not realize spatial aggregation of the signal.
- Nevertheless, it aggregates the signal across channels
  - (and transforms the result of this aggregation nonlinearly).
- Useful for reducing the number of channels, which in turn helps reduce the number of parameters (weights) in subsequent network layers.
  - The number of channels used in contemporary DL is often in orders of tens or even hundreds.
- Known in the literature as "1 by 1 convolutions"
  - Of course, there is no point in applying 1x1 convolutions to single-channel images.

Illustration: 1x1 convolution

Channels in the output tensor aggregate the channel values in the input tensor. No spatial aggregation.



Observation:

- A 1x1 convolution is equivalent to applying a dense layer (dense, fully connected) to each pixel independently.
- For this diagram, it would be a dense layer consisting of 2 units, each of which is equipped with 4 inputs.

The number of parameters of a convolution layer

For a mask (kernel) of size kxk pixels, a convolutional layer with m input channels and n output channels has a number of parameters equal to:

$$(k^2 m + 1)n$$

For example, $(3^2 * 16+1)*32 = 4640$

- Not much by today's DL standards; dense layers often have many more parameters.
- Moreover, this does not depend on the size of the input image.
  - Technical implication: we can apply software components implementing convolutional layers (e.g., Conv2D objects in TensorFlow) to images of any size.
  - Image sizes can be determined 'on the fly'.
- Many studies have shown that k=3 is often sufficient, hence in practice, the main determinants of the number of parameters are m and n.
  - ... which indicates the usefulness of 1x1 splices (see previous slides).

The concept of an effective receptive field

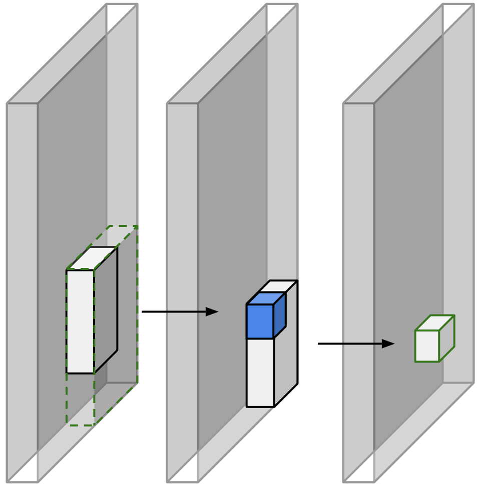
Example: Consider composing two convolutional layers A and B, both with mask sizes of 3x3.

- Each unit in B depends on a 5x5 area in the image fed to A.
- We say that the *effective receptive field* of the unit in B is 5x5.

In models with many convolutional layers, the ERF can be very large.

- ... and often covers the entire input image.
- This is often desirable, allowing the model to develop global features of the image.

The green unit in Layer 3 aggregates values within a mask in Layer 2. Each unit in Layer 2 aggregates signals in a mask from Layer 1 (among others, the highlighted blue unit). The highlighted green unit in Layer 3 has an ERF in Layer 1 marked with a dashed green line.



Hyperparameters of convolutional layers (1)

Kernel size, k:

- The dominant setting in current practice: k=3
- Provides minimal spatial aggregation (for odd k; even k is not likely to be used),
- Reduces the total number of required parameters.

  Example: a 5x5 ERF can achieved with:
  - One layer with a 5x5 kernel ⇒ $5^2 +1 = 26$ parameters
  - Two layers with 3x3 kernels ⇒ $2(3^2 +1) = 20$ parameters

    The two-layer solution has fewer parameters, while involving additional nonlinearity between the layers, which may allow more sophisticated features to be modeled.

In most implementations, k can be set independently for each spatial dimension (height, width, ...), i.e. the masks do not have to be square.

Hyperparameters of convolutional layers (2)

Padding: How to handle pixels for which the mask 'extends' beyond the input tensor? Several variants:

- *valid*: we do not allow mask to extend beyond the tensor; consequently, the output tensor is k-1 pixels smaller on each dimension
- *same*: we allow the extension: in such cases, the mask also aggregates 'virtual' pixels from beyond the image.
  The values of such pixels have to be set somehow; the default solution: complementing with zeros.
  - May cause a sudden change in brightness at the edge of the image, resulting, for example, in the appearance of apparent edges.
  - Better* output: simulating image continuation outside the raster range by
    - Mirroring the edge portion of the image (odd or even mirroring), or
    - Repeating edge pixels.
  - *From experience: Not necessarily always better.

Hyperparameters of convolutional layers (3)

Stride (step) of shifting the mask relative to the input tensor:

- stride = 1:
  Default setting: as in traditional image processing.
- stride > 1:
  The output tensor becomes *stride* times smaller.
  - Note: when stride > k, certain pixels in the input tensor will be completely ignored.
- 0 < stride < 1:
  *Transposed/fractional convolution**.
  - Often implemented independently, with a separate parameter (dilation)

In most implementations, stride can be set independently for each spatial dimension (height, width, ...).

*Sometimes called (incorrectly) deconvolution - this term translates to 'unbundling', and it means a completely different operation in signal processing.

Hyperparameters of convolutional layers (4)

Other parameters:

- Activation function
  - It is often embedded in a programming function (or object) representing the conv layer, allowing a more efficient low-level implementation.
    - For example, process profiling in TensorFlow: specialized low-level functions that combine implementation of scalar product and thresholding (for ReLU activation)
- Channel grouping
  - It forces aggregation only in groups of channels (unlike a typical layer, where each output channel aggregates all input channels).
  - A special case: *channel separable convolution*.

Parameters related to the learning algorithm:

- Initialization of weights
  - The bias is sometimes treated differently.
- Normalizations (batch normalization, BN; layer normalization, LN)

Visualizations of convolutions

https://github.com/vdumoulin/conv_arithmetic

Presents also the *dilated convolutions*.

Other local operations

Similarly to convolution, they rely on spatial aggregation of signals.

- The result is still a spatial tensor (although it may have a different size).
- The number of channels usually preserved (processing takes place in each channel independently).

Frequently used operations:

- Grouping/aggregation (pooling):
  - Particularly popular: max pooling: output is the maximum from the values in the kxk window
  - Similarly: min pooling, average pooling, ...
  - The 'max' operator used particularly often: strong response signals the presence of a feature at a given image location.
    - Particularly true for activation functions like ReLU.
- Image/tensor resampling
- Normalizations: per batch, per layer, per channel, etc.

Max pooling: Examples

For a window size of 2x2, stride=2:

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

$\longrightarrow$

4	8
5	9

For a window size of 2x2, stride=1:

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

$\longrightarrow$

4	8	8
5	9	9
5	9	9

For a window size of 3x3, stride=1:

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

$\longrightarrow$

9	9
9	9

Max pooling is usually used with the stride equal to the size of the mask (like the example in the upper left corner of this slide).

Pooling and gradient propagation

Signal pooling can impede backward gradient propagation.

The line of argument: consider the composition of the maximum operator with three other functions f, g, and h:

$$h(\max(f(x_1), g(x_2))$$

When $f(x_1) < g(x_2)$, we have:

$$\frac{\partial \max}{\partial f} = 0$$

The chain of partial derivatives defining the partial derivative of h is zeroed:

$$\frac{\partial h}{\partial \max} \frac{\partial \max}{\partial f} \frac{\partial f}{\partial x_1} = 0$$

As a result, the parameters of f are not updated (in a given training step).
In max pooling, this means that the gradient is propagated to only one pixel/element (location) in the input mask.

Covariance of convolutional layers

Convolution, pooling and other operations typical for fully convolutional architectures have in common the algebraic property of *covariance*.

An operation f is *covariant* with respect to some transformation operator T when it *commutes* with it, i.e.

$$f \circ T = T \circ f$$

For ConvNet architectures, T is image translation.

- As a result, it does not matter whether we translate the input image and then apply f to it, or the other way round (f being an arbitrarily complex ConvNet architecture)

Technical realizations

Currently dominant technologies in the market:

PyTorch

- Facebook's AI Research lab (FAIR).
- License: BSD
- https://pytorch.org/

TensorFlow

- Developed initially by Google Brain for Google's internal use.
- In the public domain since 2015.
- License: Apache
- https://www.tensorflow.org/

Technical realizations

Dominant opinions:

- PyTorch is more research-based, while TensorFlow is more technologically mature on the 'production' side.
- New architectures (proposed in scientific articles and at conferences) are prototyped slightly more frequently and quicker in PyTorch.

However, the balance of advantages and disadvantages depends on the context and application.

- Both environments currently offer very similar capabilities, and require similar efforts in preparing models.
- Attempts to integrate and communicate across platforms: Open Neural Network Exchange (ONNX) https://onnx.ai/

Relation with programming languages

Deep learning coincides with new programming paradigms:

- Differentiable programming
  - The program is treated as a computational graph, with nodes implementing differentiable operations and edges reflecting the flow of data.
  - The graph may be
    - static (created/compiled once), or
    - dynamic, built in the course of calculations, when data comes in (so-called eager mode).
- Probabilistic programming

  Program variables become random variables.

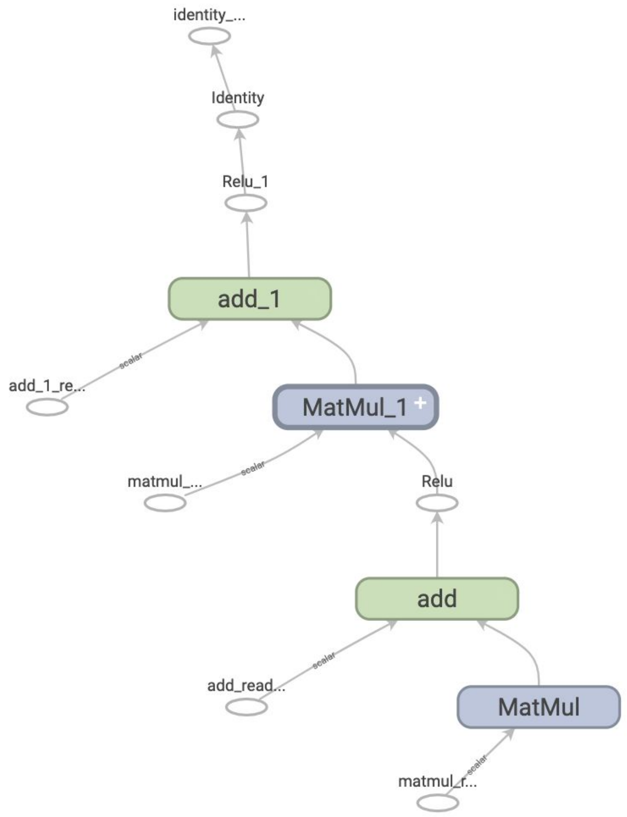
  Known representatives:
  - Church (based on LISP)
  - Figaro (based on Scala)
  - PRISM (based on Prolog)
  - Anglican (based on Clojure)

Example

An example of a computational graph (TensorFlow), involving

- a matrix multiplication,
- adding,
- the ReLU activation function.

Contemporary environments and libraries offer tools for analyzing, visualizing, as well as optimizing such graphs (e.g. Grappler in TensorFlow).



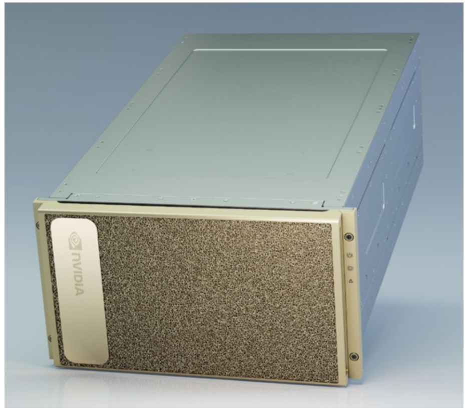
https://www.tensorflow.org/guide/intro_to_graphs

Increasing capabilities of hardware

Hardware manufacturers compete at developing platforms oriented to 'tensor-like' parallel processing.

Architectures are increasingly specialized; examples:

- TensorCores in contemporary GPUs (16b FP precision)
- NVidia's A100 processors: 9.7 TFLOPS
  (used in the DGX A100, among others, pictured right)
- Tensor Processing Unit, TPU (Google): 45 TFLOPS
  (mainly used in Google's cloud solutions).

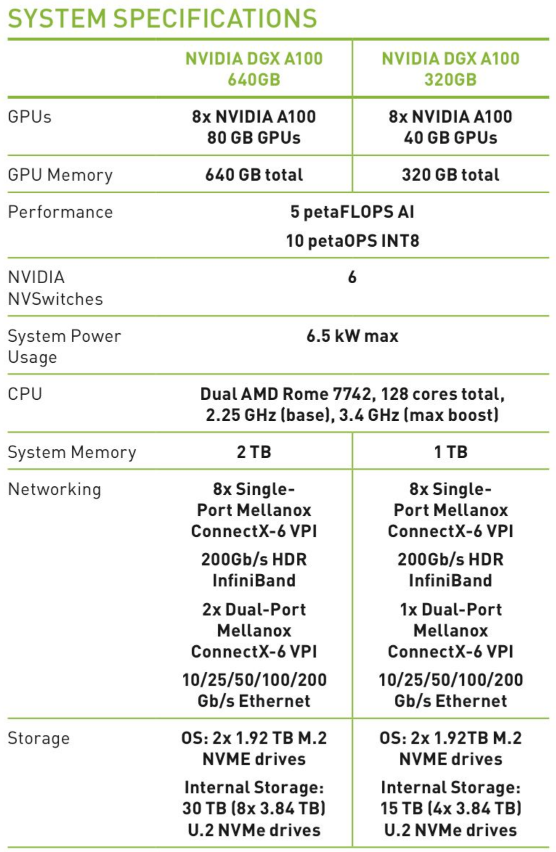


Increasing capabilities of hardware

The growing capabilities of processors bring with them greater requirements for data, network, and operational and storage infrastructure.

Example: characteristics of the DGX A100 server (right).

These requirements often drive up prices more than the computing units themselves (processors, cores).



SYSTEM SPECIFICATIONS

	NVIDIA DGX A100 640GB	NVIDIA DGX A100 320GB
GPUs	8x NVIDIA A100 80 GB GPUs	8x NVIDIA A100 40 GB GPUs
GPU Memory	640 GB total	320 GB total
Performance	5 petaFLOPS AI 10 petaOPS INT8	
NVIDIA NVSwitches	6	
System Power Usage	6.5 kW max	
CPU	Dual AMD Rome 7742, 128 cores total, 2.25 GHz (base), 3.4 GHz (max boost)	
System Memory	2 TB	1 TB
Networking	8x Single-Port Mellanox ConnectX-6 VPI 200Gb/s HDR InfiniBand 2x Dual-Port Mellanox ConnectX-6 VPI 10/25/50/100/200 Gb/s Ethernet	8x Single-Port Mellanox ConnectX-6 VPI 200Gb/s HDR InfiniBand 1x Dual-Port Mellanox ConnectX-6 VPI 10/25/50/100/200 Gb/s Ethernet
Storage	OS: 2x 1.92 TB M.2 NVME drives Internal Storage: 30 TB (8x 3.84 TB) U.2 NVMe drives	OS: 2x 1.92TB M.2 NVME drives Internal Storage: 15 TB (4x 3.84 TB) U.2 NVMe drives

Selected milestones of ConvNet architectures

AlexNet

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton

http://arxiv.org/pdf/1409.0575

https://dl.acm.org/doi/10.5555/2999134.2999257

(NIPS 2012) (currently NeurIPS)

Contributions

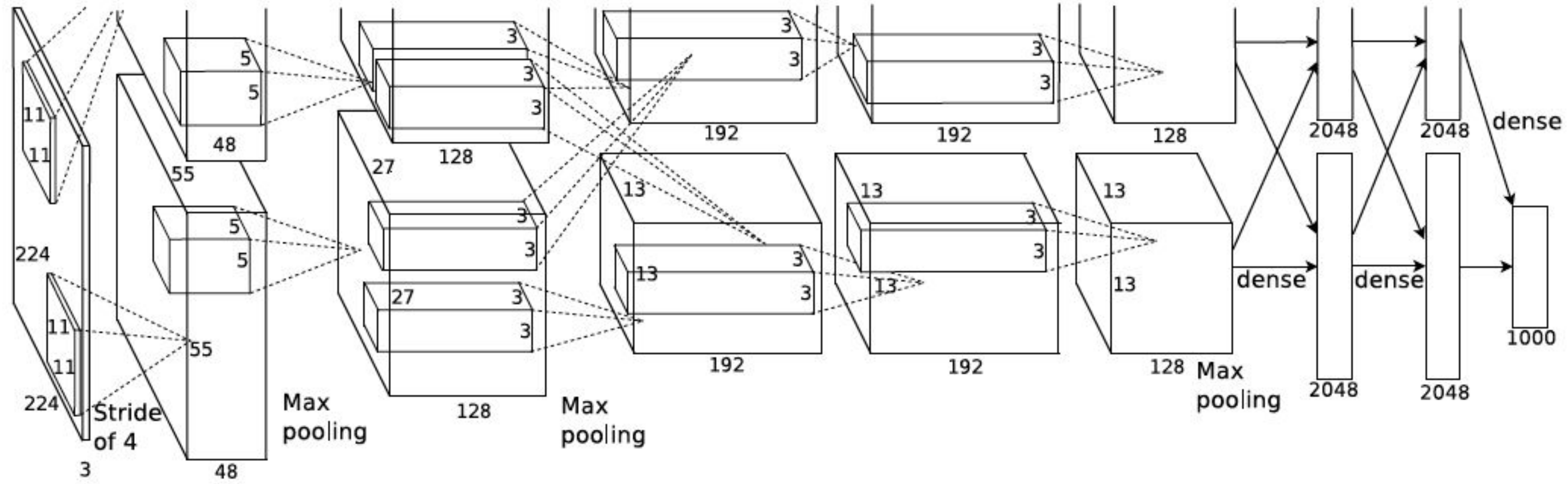
- Large and deep (by 2012 standards)
- Trained on 1.2M images
- Outperformed SoTA methods of the time.
- Second place in the ILSVRC-2012 competition

Architecture: convolutional-dense image classification model

- Five convolutional layers
- Max pooling layers
- Dropout (recently introduced at that time)
- One of the first networks to use ReLU activation functions
- ~60M parameters
- 1000 classes (ImageNet LSVRC contest)

Architecture

Deployed and trained on 2 GPUs (GTX 580, 3GB)



Results

Effectiveness of ReLUs ————>

Summary of results:

Model	Top-1 (val)	Top-5 (val)	Top-5 (test)
SIFT + FVs [7]	—	—	*26.2%*
1 CNN	40.7%	18.2%	—
5 CNNs	38.1%	16.4%	**16.4%**
1 CNN*	39.0%	16.6%	—
7 CNNs*	36.7%	15.4%	**15.3%**

1 CNN - single CNN

5 CNN - averaging the predictions of 5 CNNs

Another important conclusion: depth is essential.

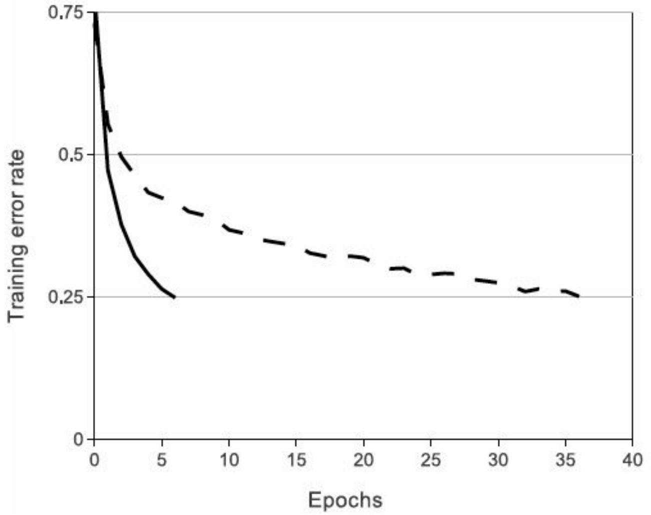
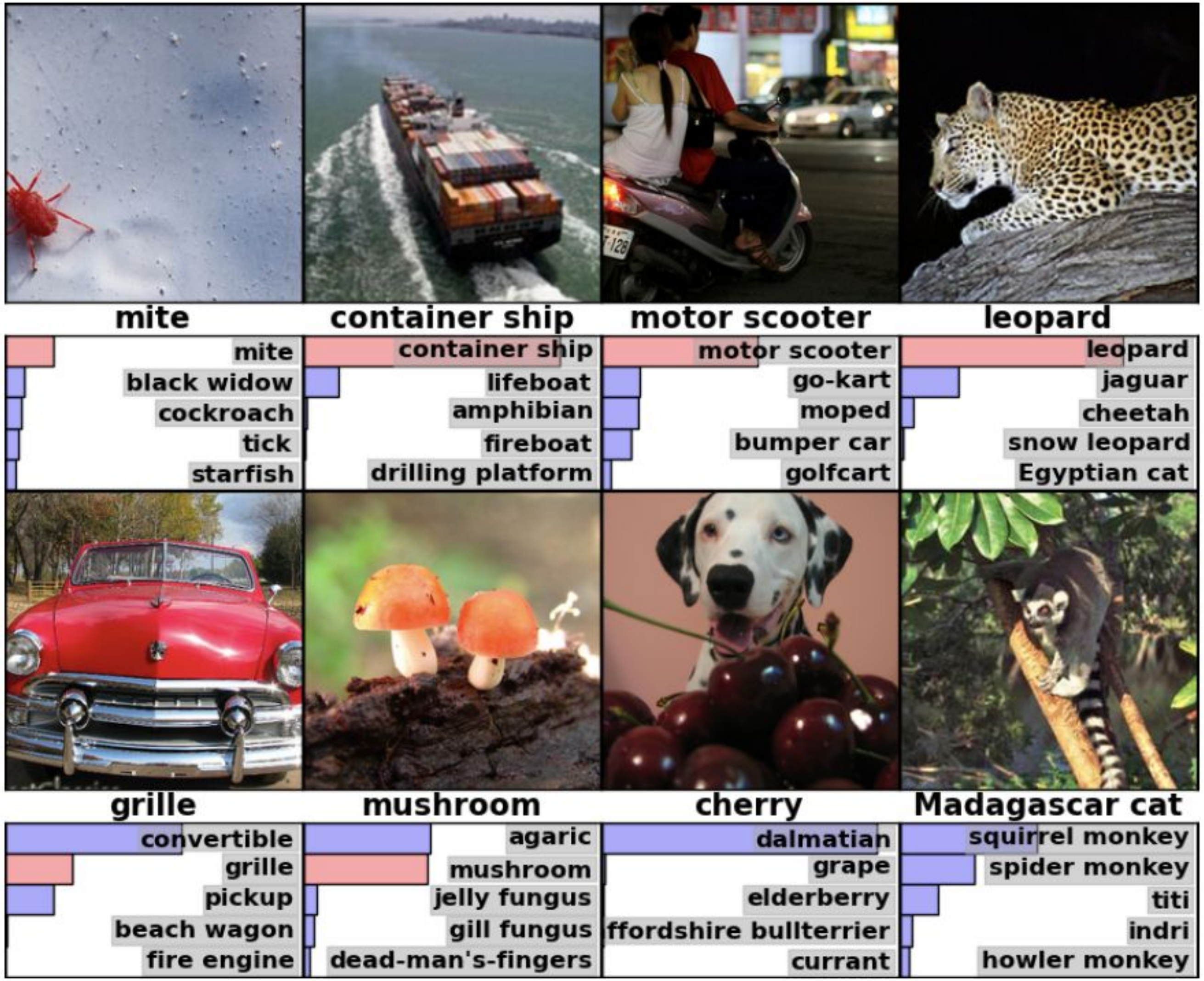


Figure 1: A four-layer convolutional neural network with ReLUs **(solid line)** reaches a 25% training error rate on CIFAR-10 six times faster than an equivalent network with tanh neurons **(dashed line)**. The learning rates for each network were chosen independently to make training as fast as possible. No regularization of any kind was employed. The magnitude of the effect demonstrated here varies with network architecture, but networks with ReLUs consistently learn several times faster than equivalents with saturating neurons.

Top-5 characteristics



VGG models

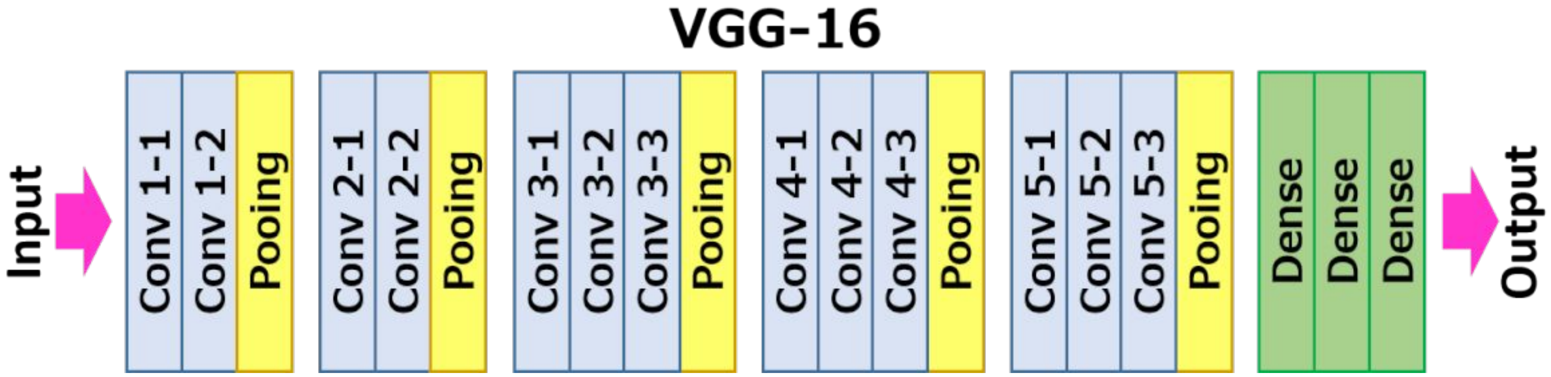
Very Deep Convolutional Networks for Large-Scale Image Recognition
Karen Simonyan, Andrew Zisserman
https://arxiv.org/abs/1409.1556

[The model named after the name of authors’ team that won the ILSVRC-2014 contest]

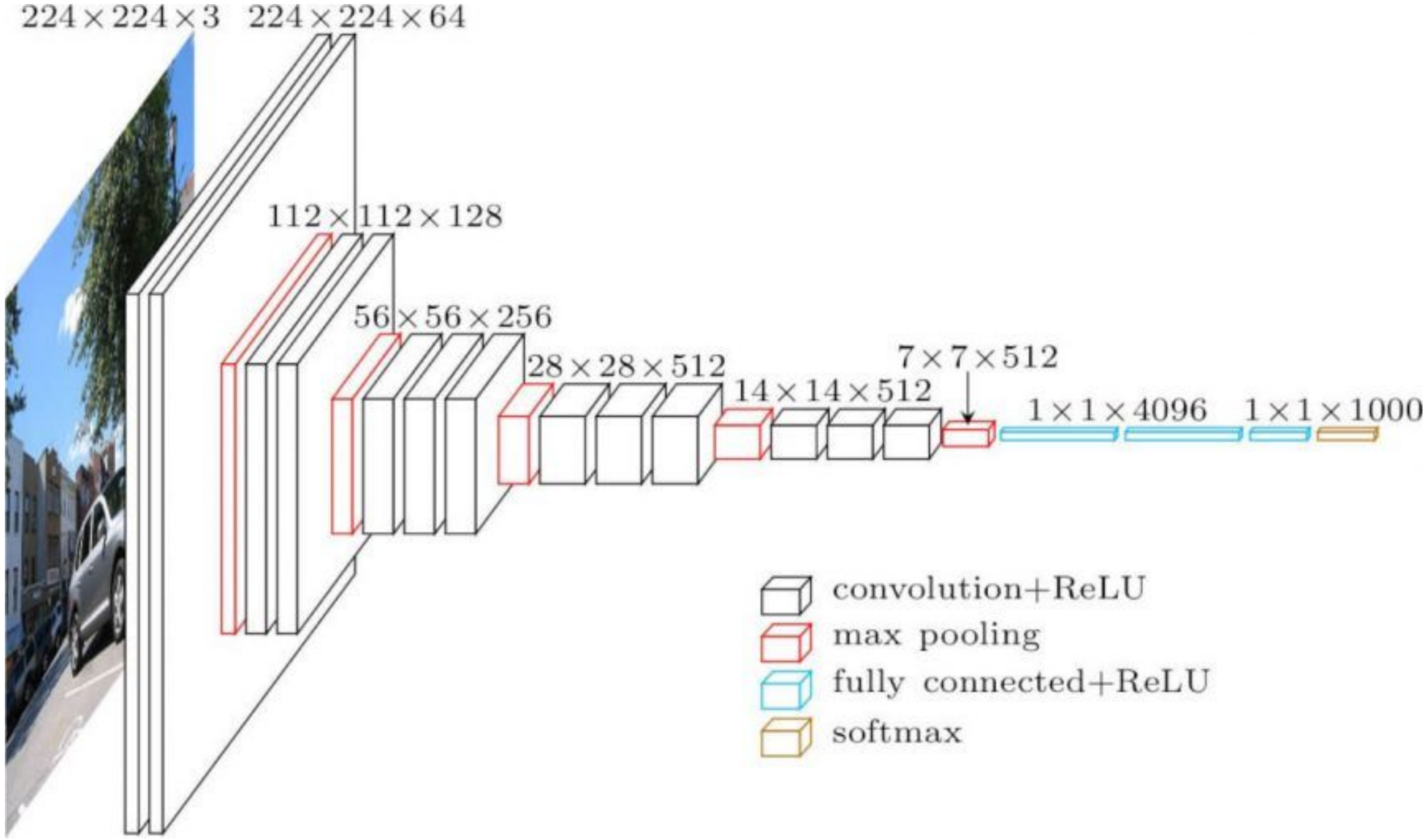
Original contribution

Our main contribution is a thorough evaluation of networks of increasing depth using an architecture with very small (3×3) convolution filters, which shows that a significant improvement on the prior-art configurations can be achieved by pushing the depth to 16–19 weight layers.

Implications: lower number of parameters, more efficient training, less overfitting.



https://neurohive.io/en/popular-networks/vgg16/


224 × 224 × 3
224 × 224 × 64
112 × 112 × 128
56 × 56 × 256
28 × 28 × 512
14 × 14 × 512
7 × 7 × 512
1 × 1 × 4096
1 × 1 × 1000
convolution+ReLU
max pooling
fully connected+ReLU
softmax

Motivation

Compare architectures:

1. One convolutional layer with 5x5 receptive field
2. Two convolutional layers, with 3x3 receptive fields

(assume single-channel layers, for simplicity).

Both have the same *effective* receptive field: 5x5, so the same ‘spatial scope’.
However, in terms of the number of parameters:

1. $5^2+1 = 26$
2. $2(3^2+1) = 20$

The difference becomes only more prominent for greater receptive fields.

Conclusion: Small receptive fields can provide the same ERF at a lower number of parameters. The network becomes deeper, but today we have algorithms that can train deep architectures efficiently.

The Inception Network

Going Deeper with Convolutions
Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich
https://arxiv.org/abs/1409.4842

- Further increase of depth, compared to earlier models.
- Modular architecture.
- Multiple output modules.

The origin of the name: the phrase
“we need to go deeper” from the *Inception* movie.

Main features

- Modularity: most of the architecture is made of multiple instances of the same module (Inception module)
    - Same architecture of the module, different parameterization.
- Increased the depth and width of the network while keeping the computational budget constant
    - More precisely: computational cost increases ~proportionally to the depth of the network.
- Architectural decisions based on the Hebbian principle and the intuition of multi-scale processing.
- Specific variant/instance: GoogLeNet, a 22-layer Inception network

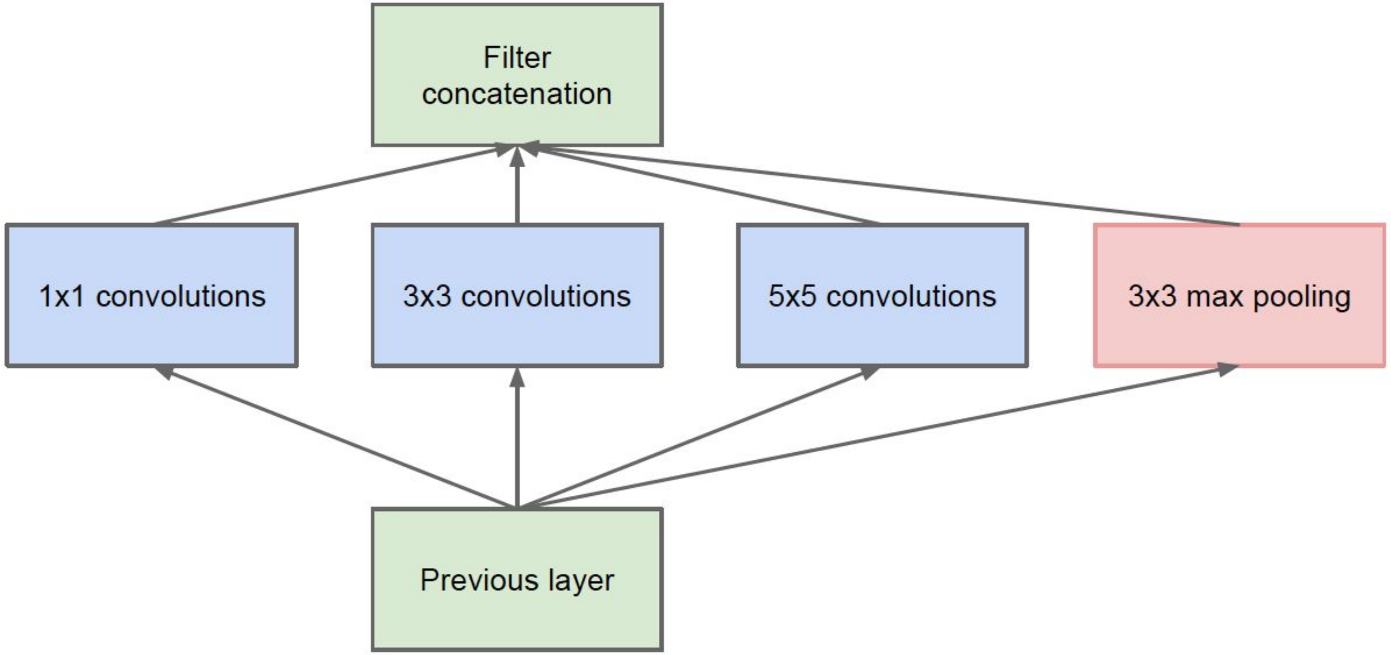
Motivations

- Arora et al. [2]:
  If the probability distribution of the dataset is representable by a large, very sparse deep neural network, then the optimal network topology can be constructed layer after layer <u>by analyzing the correlation statistics of the preceding layer activations and clustering neurons with highly correlated outputs</u>.
  - Related to Hebbian principle – *Neurons that fire together, wire together* [Donald Hebb, 1949]
- If two convolutional layers are chained, an increase in the number of their filters results in a quadratic increase of the number of weights.
  - If the added capacity (weights) is used inefficiently (for example, if <u>most weights end up to be close to zero</u>), then much of the computation is wasted.
  - This could be addressed with sparse data structures, but today's computing architectures are inefficient for such structures.

Motivations

- How an optimal local sparse structure of a convolutional vision network can be approximated and covered by readily available dense components?
  - Assuming translation invariance => use convolutional building blocks.
  - All we need is to find the optimal local construction and to repeat it spatially.
- In the lower layers (the ones close to the input) correlated units would concentrate in local regions.
  - Clusters concentrated in a single region can be covered by a layer of 1x1 convolutions.
- Smaller number of more spatially spread out clusters that can be covered by convolutions over larger patches
  - Hence also 3x3 and 5x5 convolutions.
- All those layers concatenated into a ‘filter bank’
- Additional pooling operations performed in parallel to the above.

Inception module: naive version



- The ratio of 3x3 and 5x5 convolutions to 1x1 should increase with consecutive layers, as features become sparser and more spatially distributed.
- The challenge: large depth of concatenated filters.

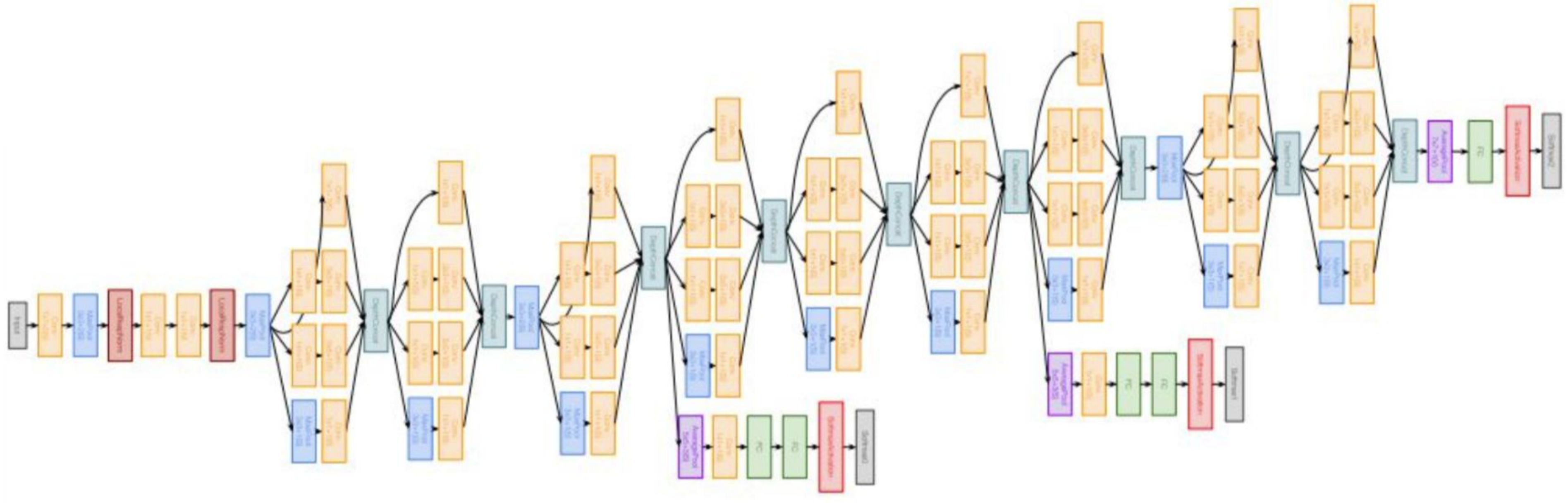
GoogLeNet

- 22 layers deep (if counting only layers with parameters)
  - 27 layers if also counting pooling
- Uses average pooling before the classifier (with additional linear layer)

Addressing the vanishing gradient problem:

- *The strong performance of shallower networks on this task suggests that the features produced by the layers in the middle of the network should be very discriminative*.
- By adding (to the ‘stem network’) auxiliary classifiers connected to these intermediate layers, discrimination in the lower stages in the classifier was expected.
  - Additional source of training signal (gradient).

GoogLeNet: Bird's-eye view



Question

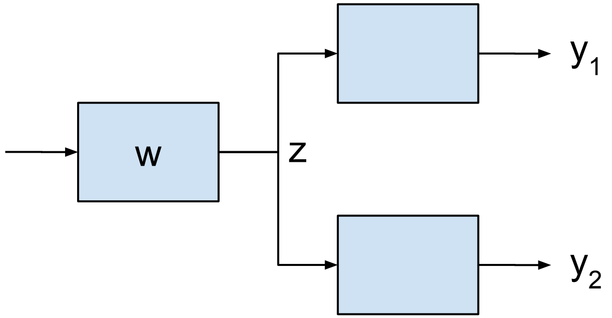
What happens to gradient at the branching points?

Assuming that the total loss is a sum of losses at the ends of branches, where the model returns values y_1 and y_2:

$$L = L_1(y_1) + L_2(y_2)$$

The total derivative of loss function w.r.t. the vector w of weights/parameters of the common part, where z is the output of the common part:

$$\frac{\partial L}{\partial w} = \frac{\partial z}{\partial w}\frac{\partial y_1}{\partial z}\frac{\partial L_1}{\partial y_1} + \frac{\partial z}{\partial w}\frac{\partial y_2}{\partial z}\frac{\partial L_2}{\partial y_2}$$



Residual networks

Deep Residual Learning for Image Recognition

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun

https://arxiv.org/abs/1512.03385

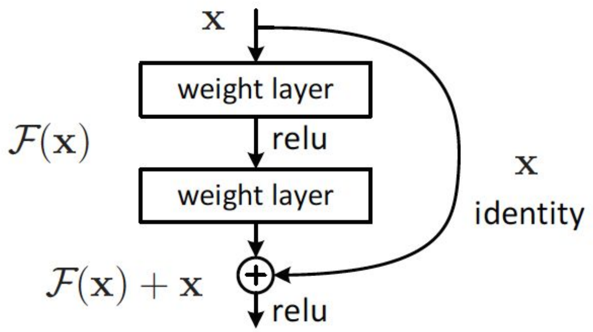
- First well-known study introducing the concept of residual connections.

Residual networks

- Reformulate the training task for a layer as learning a residual function with reference to the layer's inputs, instead of learning the original function.
- The authors use residual nets with a depth of up to 152 layers
  - 8 layers deeper than VGG nets.
- 1st place in the ILSVRC 2015 classification contest.
- 1st places in the following contests:
  - ImageNet detection, ImageNet localization, COCO detection, COCO segmentation.

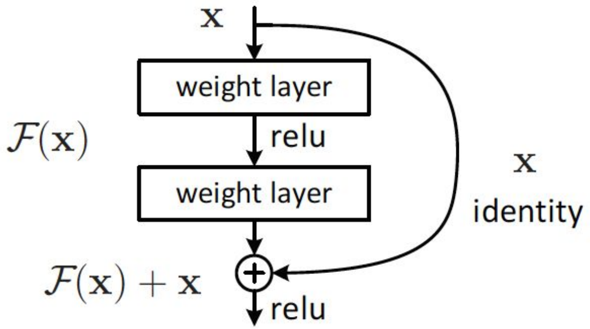
https://www.image-net.org/ https://cocodataset.org/

Motivations



- Let
  - H(x) be the mapping to be fit by a subnetwork/submodel (a few stacked layers, not necessarily the entire net), with x the input to that submodel.
  - Assume that x and H(x) have the same dimensions (‘shapes’).
- If multiple nonlinear layers can asymptotically approximate H(x), then it is equivalent to hypothesize that they can asymptotically approximate the residual function, i.e., H(x) - x
  - Let’s **explicitly let these layers approximate a residual function** F(x) := H(x)-x.
- In other words: after successful training, the subnetwork will implement the function F(x)+x.
- Although both submodels (direct and residual) should be able to asymptotically approximate the desired function (as hypothesized),
  - the difficulty of learning might be different (due to ‘shortcuts’ in gradient flow).
  - This is particularly true when chaining residual modules.

Basic Resnet block



Mapping realized by a Resnet building block:

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x}$$

where $\{W_i\}$ are the parameters of F.

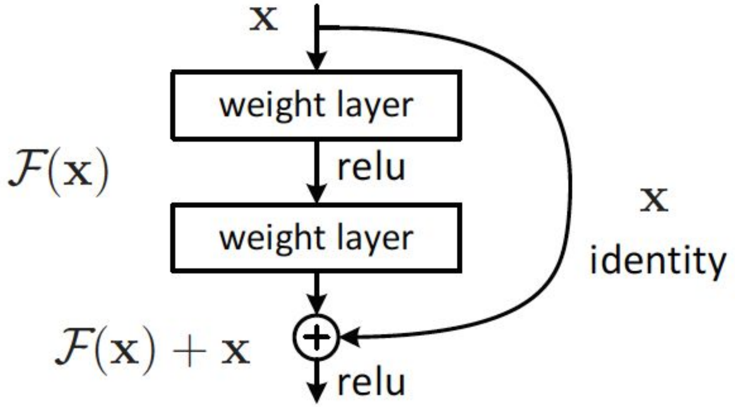
Q: What if, for some reason, the shapes of F and x are different?
A: We can still realize a 'pseudo-shortcut': a linear mapping that uses a (trainable) matrix of parameters W:

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + W_s\mathbf{x}$$

Also: Notice the presence of ReLU activation *after* the summation.

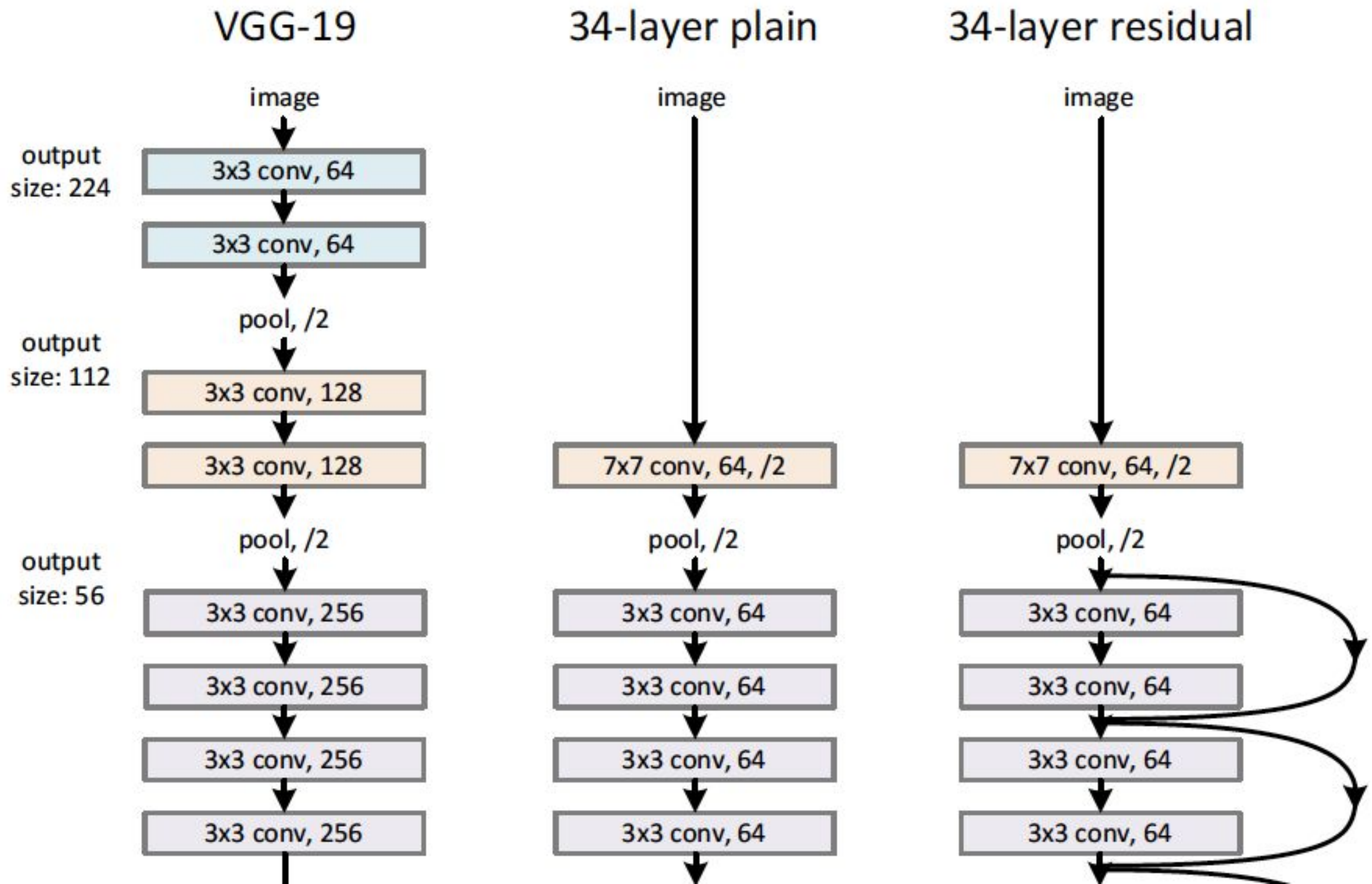
Basic building block

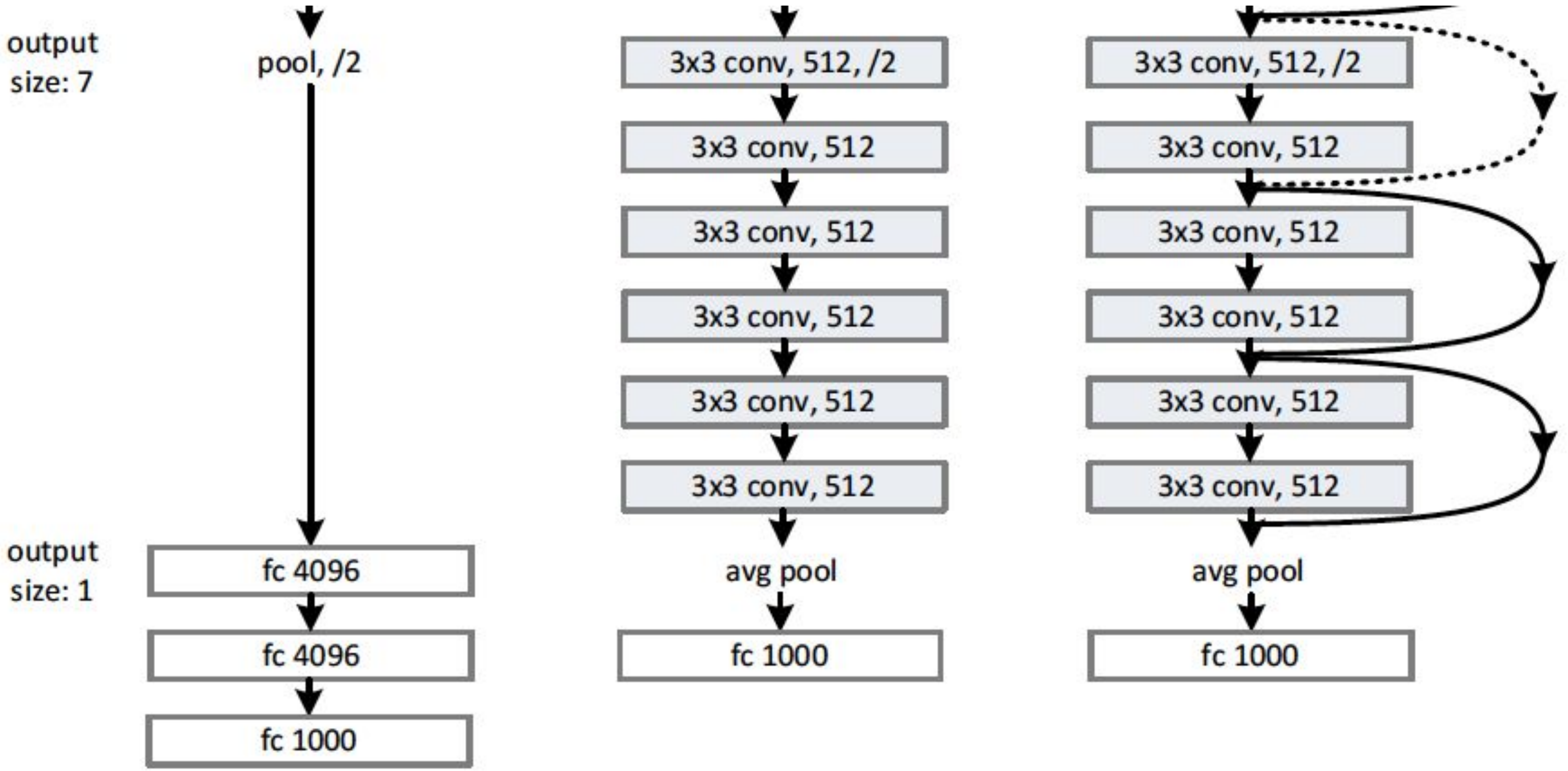
The specific building block used by the authors:



Similarly to GoogLeNet, Resnets are thus *modular networks.*

- However, this is 'just' structural modularity:
    - Each instance of a Resnet block has the same architecture, but has separate weights and thus implements a different function (there's no weight sharing between individual building blocks/modules).


VGG-19
34-layer plain
34-layer residual
image
image
image
output size: 224
3x3 conv, 64
3x3 conv, 64
pool, /2
output size: 112
3x3 conv, 128
3x3 conv, 128
7x7 conv, 64, /2
7x7 conv, 64, /2
pool, /2
pool, /2
pool, /2
output size: 56
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64


output size: 7
pool, /2
output size: 1
fc 4096
fc 4096
fc 1000
3x3 conv, 512, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
avg pool
fc 1000
3x3 conv, 512, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
avg pool
fc 1000

Other elements

Batch normalization after each convolution (before activation)

Two options for the dotted connections:

- A: Parameter-free: The shortcut still performs identity mapping, with extra zero entries padded for increasing dimensions (no extra parameters)
- B: Parametric: Each output channel is a linear combination of input channels. The projection shortcut is used to match dimensions, using 1x1 convolutions:

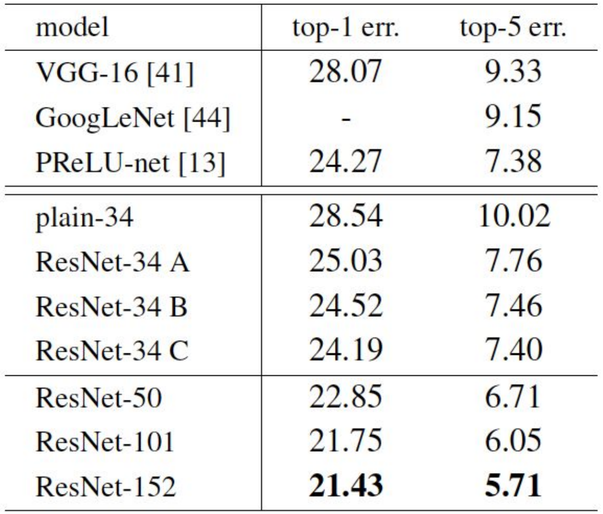
$$\mathbf{y} = \mathcal{F}\left(\mathbf{x}, \{W_i\}\right) + W_s \mathbf{x}$$

Some results

(A) zero-padding shortcuts are used for increasing dimensions, and all shortcuts ar parameter free
(B) projection shortcuts are used for increasing dimensions, and other shortcuts are identity;
(C) all shortcuts are projections

Trained with ordinary SGD with momentum!

Experimented also with ensambles; networks up to 1000 layers (found it to be worse than 110-layer net)



model	top-1 err.	top-5 err.
VGG-16 [41]	28.07	9.33
GoogLeNet [44]	-	9.15
PReLU-net [13]	24.27	7.38
plain-34	28.54	10.02
ResNet-34 A	25.03	7.76
ResNet-34 B	24.52	7.46
ResNet-34 C	24.19	7.40
ResNet-50	22.85	6.71
ResNet-101	21.75	6.05
ResNet-152	**21.43**	**5.71**

Table 3. Error rates (%, **10-crop** testing) on ImageNet validation. VGG-16 is based on our test. ResNet-50/101/152 are of option B that only uses projections for increasing dimensions.

Why do ResNets work?

- They address the vanishing/exploding gradient problem.
    - Each model component with partial derivatives close to zero increases the risk of vanishing gradient.
    - Squeezing activation functions are particularly notorious in that respect (gradient close to zero almost everywhere).
- This implicitly allows maintaining relatively low depth (number of channels/dimensions) along the network
    - Projections used only thrice in the architecture (cf. figure).
- Does modularity help?
    - An open question.
    - Definitely eases automation of architecture optimization: some design choices cease to be available (part of architectural hiperparameters are fixed).

Related: Wide Residual Networks (Wide-Nets) http://arxiv.org/abs/1605.07146: emphasize Resnet width rather than depth. See the module Auxiliary topics.

Fully convolutional architectures

Fully convolutional networks (ConvNets, FCN)

Models that comprise exclusively local operations, i.e. mostly:

- Convolutions and transposed convolutions
- Pooling
- Re-sampling (down- and upsampling, interpolation).

A fully convolutional architecture F:

- Can handle images of arbitrary size.
- Is covariant with respect to translation
  - F and translation *commute*.

Motivations

Fully convolutional architectures are applicable in any usage scenario that requires image-to-image mapping, i.e. any *image processing*.

Prominent examples:

- Segmentation
  - Including semantic segmentation
- Denoising
  - The network serves as a denoising filter.
  - Can adapt in training to the characteristics (distributions) of a given class of images/domain.
- Superresolution
- Transfer/translation of local characteristics, e.g. style transfer
  - Also: pseudocoloring, texture filling, and more
  - Semantic transformations, e.g. synthesizing an aerial image based on a map.
    - See, e.g., so-called Pix2Pix models

Applying FCNs to image processing tasks

Advantages:

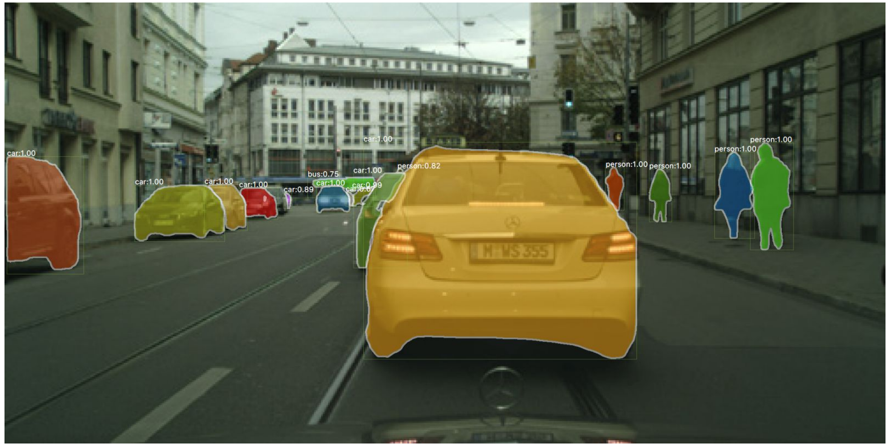
- Each instance of effective receptive field is a separate example.
  - E.g., a single convolutional layer with 3x3 filters, when applied to a 100x100 pixel image, processes 98^2=9604 examples simultaneously
    - (albeit those examples are not independent).
- Each parameter of the model receives gradient from all locations in its ‘impact field’ (the set of locations in the output image that depend on that parameter)
  - Implication: stronger training signal than when the loss is defined only on the level of entire images (e.g. image classification or regression).

Challenges:

- Border effects: the larger ERF, the more prominent.
- Input-output image pairs have to be precisely spatially coregistered.

Image segmentation has many faces ...

- 'Ordinary' segmentation
  - The algorithm is only required to delineate regions that are perceptually distinct.
- Semantic segmentation
  - The algorithm is expected to work with semantics-rich labels, like sidewalk, car, person, road sign, ...
- Object instance segmentation
  - Multiple instances of the same type of entity should be delineated.
- ...



DNN-based approaches to segmentation

Main categories of representatives:

- Patch-based - a ‘naive’ approach
  - Replaces the segmentation task with classification of individual pixels based of image patches
- Fully convolutional (ConvNet, FCNN)
  - Performs segmentation of all image patches in parallel
- Recurrent
  - Uses recurrent layers/cell/networks to ‘sweep’ the input image.
- Attention-based models
  - Have a separate module that ‘actively seeks’ objects in the image.

U-Net

U-Net: Convolutional Networks for Biomedical Image Segmentation

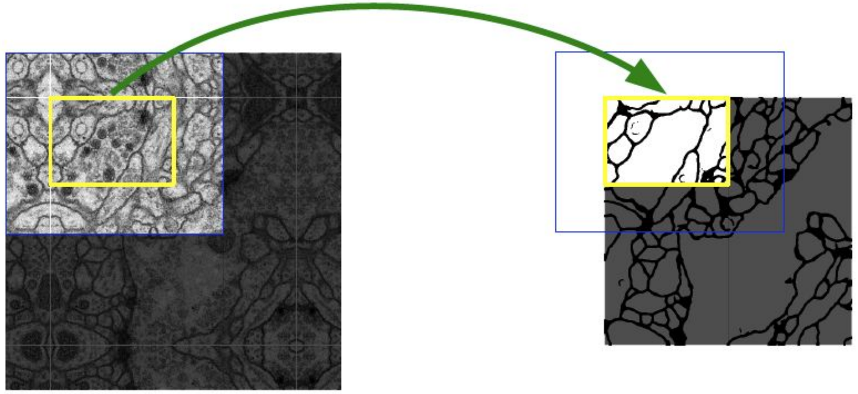
Olaf Ronneberger, Philipp Fischer, Thomas Brox

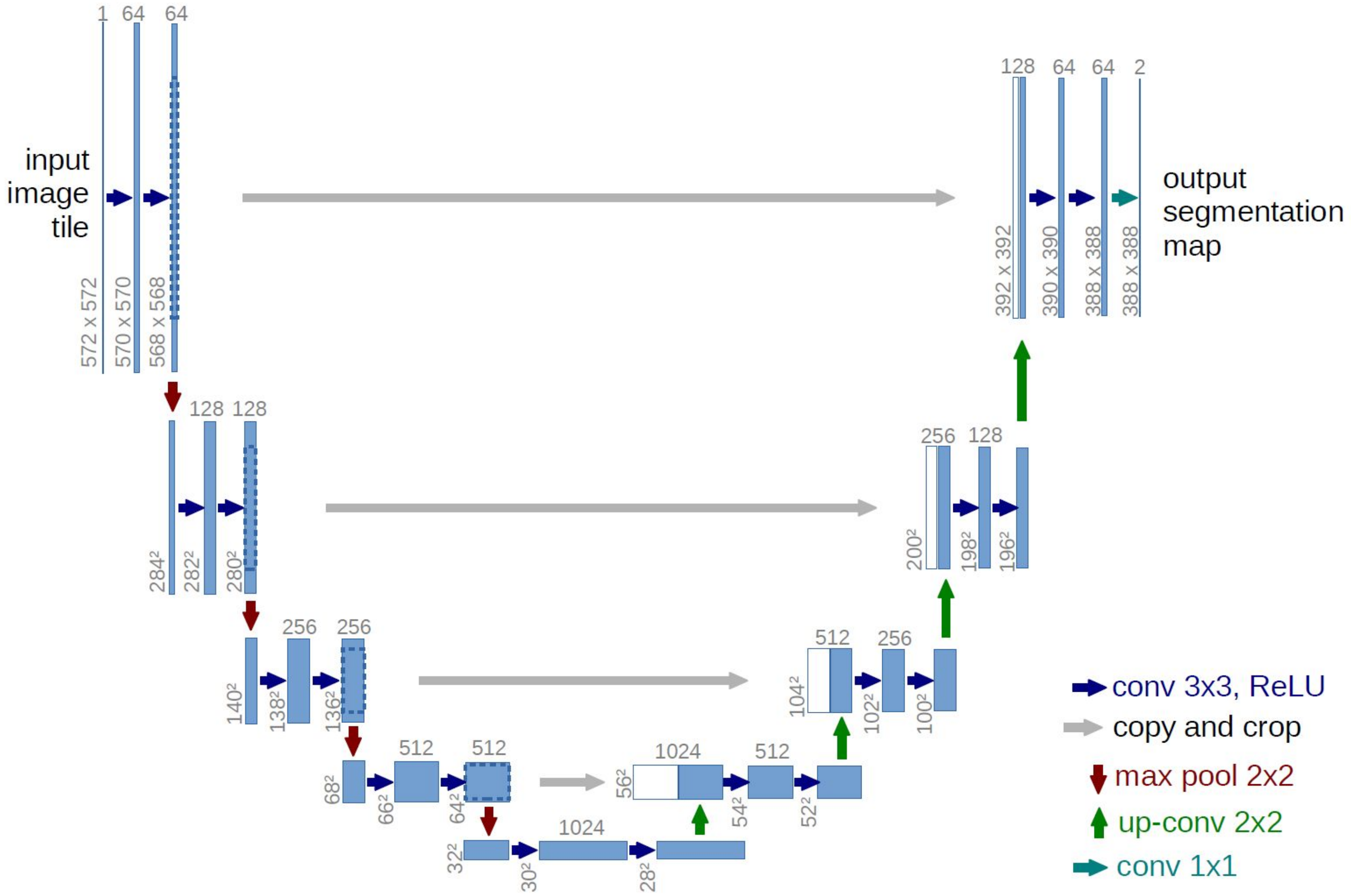
https://arxiv.org/abs/1505.04597

- One of the most popular ‘design patterns’ for fully convolutional architectures.
- Proved very effective in a large number of applications.
- 10k citations on arXiv (as of Dec 2021)

The architecture

- Uses ConvNet to supplement a usual contracting network by successive expanding layers, where pooling operators are replaced by upsampling operators.
- Main building blocks:
  - a contracting stack to capture context and construct higher-level features,
  - a symmetric expanding stack that enables precise localization,
  - ‘copy&crop’ connections bridging the contracting path with the expanding path.
- The network does not have any fully connected layers and only uses the valid part of each convolution.
  - Padding via mirroring at the edge of the image.




1
64
64
input image tile
572 x 572
570 x 570
568 x 568
128
128
284²
282²
280²
256
256
140²
138²
136²
512
512
68²
66²
64²
1024
32²
30²
28²
1024
512
56²
54²
52²
512
256
104²
102²
100²
256
128
200²
198²
196²
128
64
64
2
392 x 392
390 x 390
388 x 388
388 x 388
output segmentation map
conv 3x3, ReLU
copy and crop
max pool 2x2
up-conv 2x2
conv 1x1

Stride and fractional convolutions

Stride controls the speed of shifting the mask (receptive field) over the input tensor comp *relative to the speed of moving over output tensor*.

size(output) = stride*size(input)

- stride=1 ⇒ size(output) = size(input)
- stride>1 ⇒ size(output) < size(input)
  - Example: stride=2 implies skipping every second receptive field in the *input*
- stride<1 ⇒ size(output) > size(input)
  - Example: stride=1/2 implies skipping every second receptive field in the *output*

Naming convention:

- stride≥1 : *convolution*
- stride<1 : *fractional convolution* (because in this case stride=1/k, k∈N)

Transposed convolutions

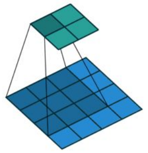
The distinction between the ‘regular’ convolution and the transposed convolution (*up-conv*) concerns the ‘fan-in’ and ‘fan-out’, i.e. the flow of signals.

- Regular convolutions are *contracting*: a k×k receptive field in input is mapped/projected to a single element (unit, location) in output.
- Transposed convolutions are *expanding*: a single input (unit, pixel) is mapped/projected onto a rectangular region in the output tensor.
  - The contributions from individual inputs are subsequently summed.

Transposed convolutions are sometimes (erroneously!) called deconvolutions.

Unet uses transposed fractional convolutions in the expanding path.

Recommended: helpful animated illustrations of various types of convolutions
https://github.com/vdumoulin/conv_arithmetic



Some comments on U-net

- Advantages: Trains effectively from small samples.
  - (like most pure ConvNets)
- Can be trained to label:
  - Regions (interiors)
  - Region borders
- Q: Is this an autoencoder?
- Important: The ‘copy and crop’ connections are <u>not</u> equivalent to residual connections.
  - However, they definitely allow part of gradient flow along a shorter path.
- This is a ‘fully ConvNet’ model:
  - All operations (convolutions, max pooling, up-conv) rely on <u>local receptive fields</u>.
  - Therefore, U-nets are scalable/applicable to images of arbitrary dimensions.

U-Net’s weighted target on HeLa cells image

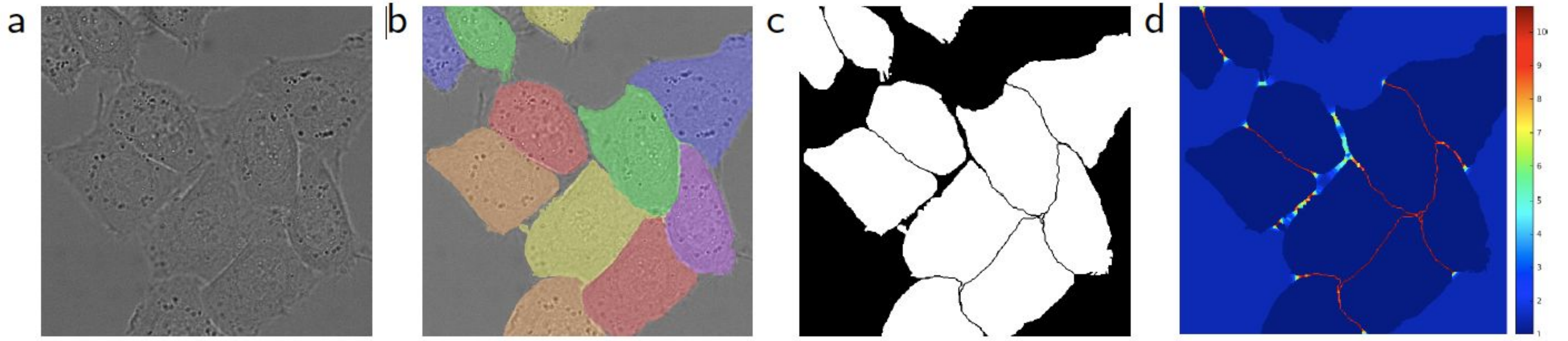


Fig. 3. HeLa cells on glass recorded with DIC (differential interference contrast) microscopy. (**a**) raw image. (**b**) overlay with ground truth segmentation. Different colors indicate different instances of the HeLa cells. (**c**) generated segmentation mask (white: foreground, black: background). (**d**) map with a pixel-wise loss weight to force the network to learn the border pixels.

Some results

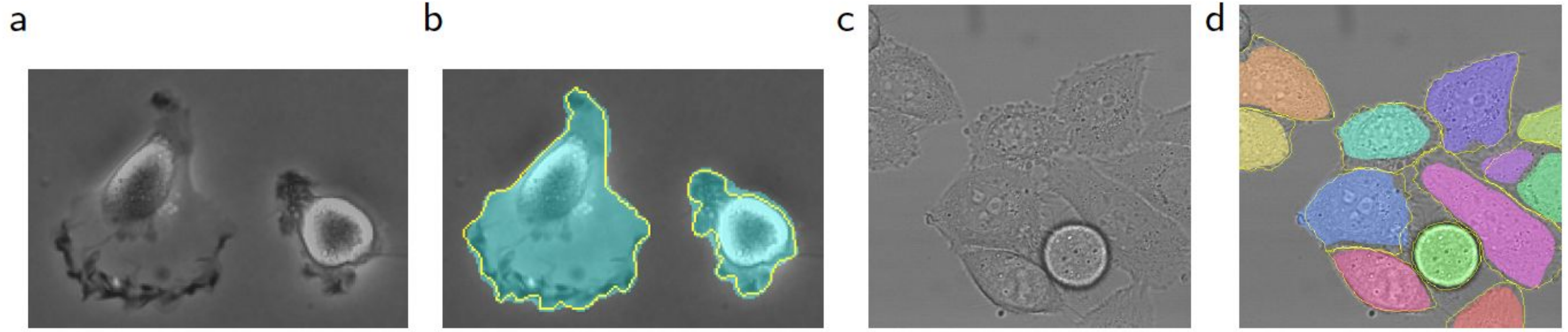
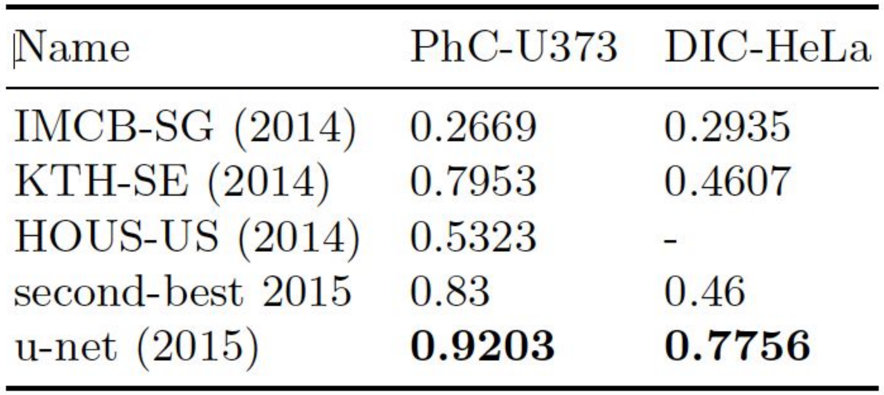


Fig. 4. Result on the ISBI cell tracking challenge. (**a**) part of an input image of the "PhC-U373" data set. (**b**) Segmentation result (cyan mask) with manual ground truth (yellow border) (**c**) input image of the "DIC-HeLa" data set. (**d**) Segmentation result (random colored masks) with manual ground truth (yellow border).

Some results



Name	PhC-U373	DIC-HeLa
IMCB-SG (2014)	0.2669	0.2935
KTH-SE (2014)	0.7953	0.4607
HOUS-US (2014)	0.5323	-
second-best 2015	0.83	0.46
u-net (2015)	**0.9203**	**0.7756**

The metric: IOU = Intersection over Union

- A: detected region
- B: target region

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

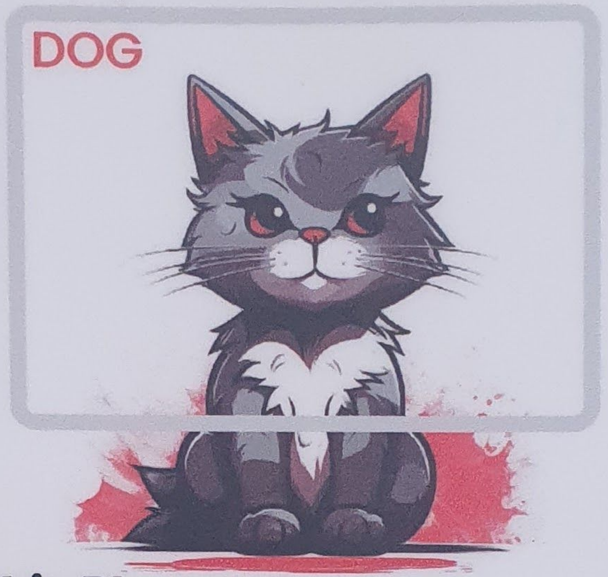
Final comment:

- The original paper assumed the model has to label *regions*.
- UNet works also quite well for labeling *borders*.

Challenges

People saying AI will take over the world

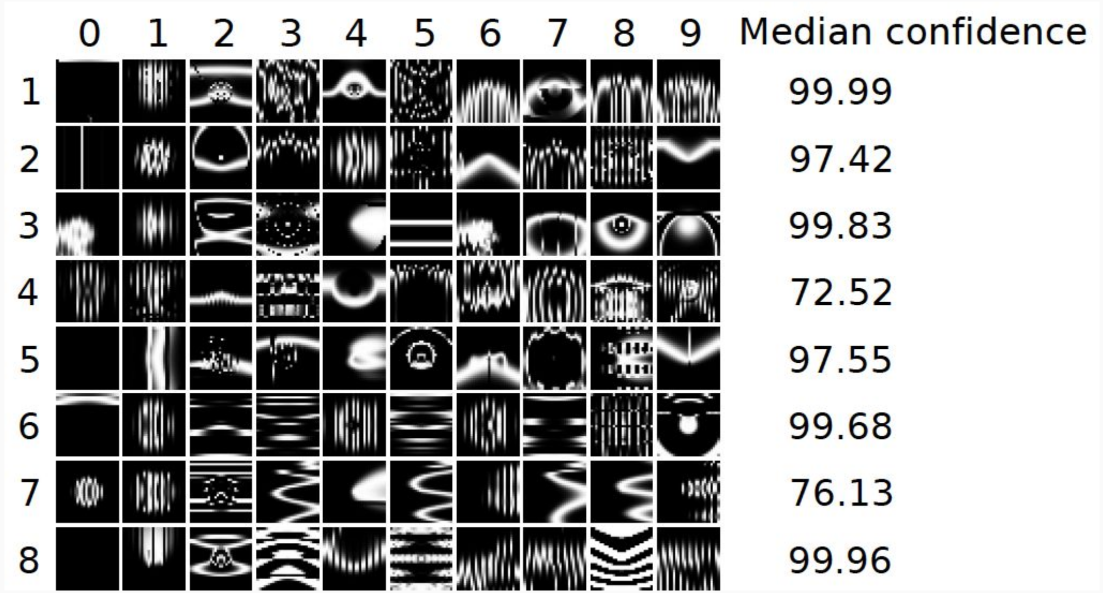
My neural network:



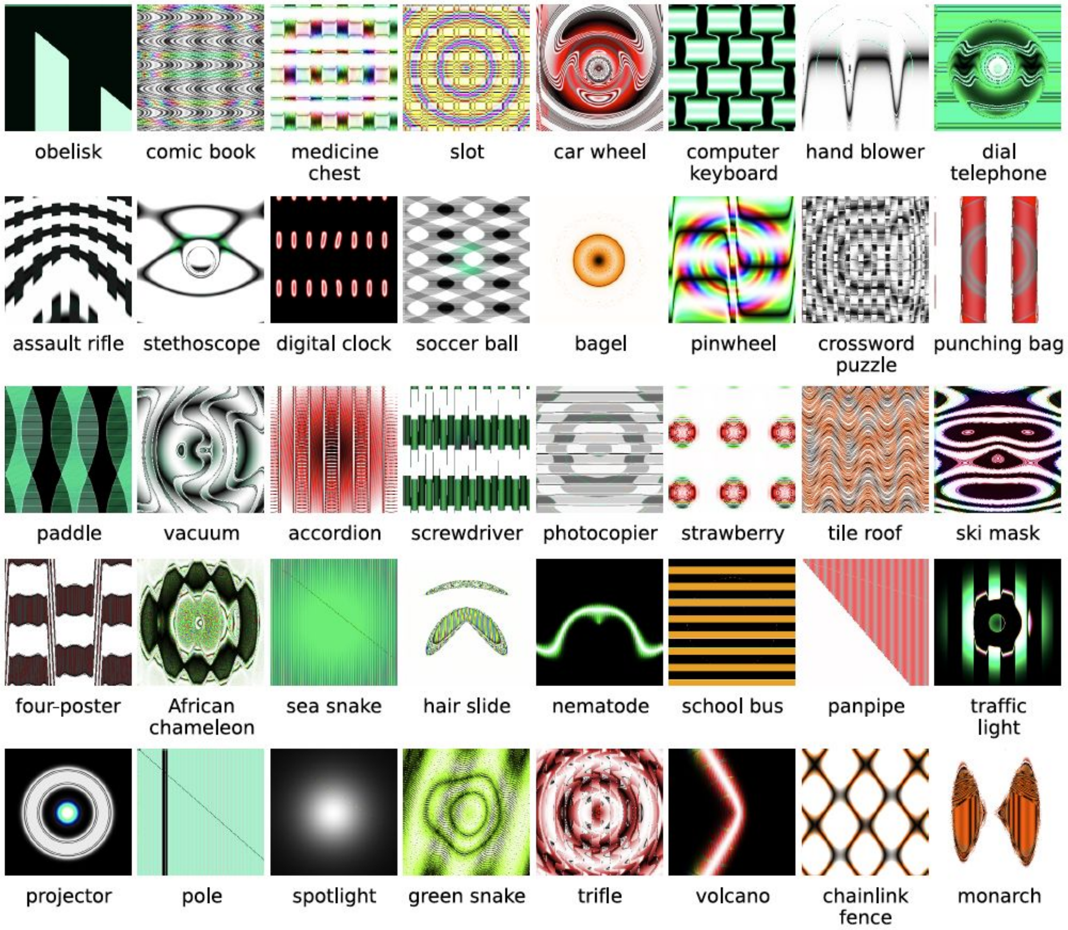
ML in PL ASSOCIATION

Challenges: Overfitting

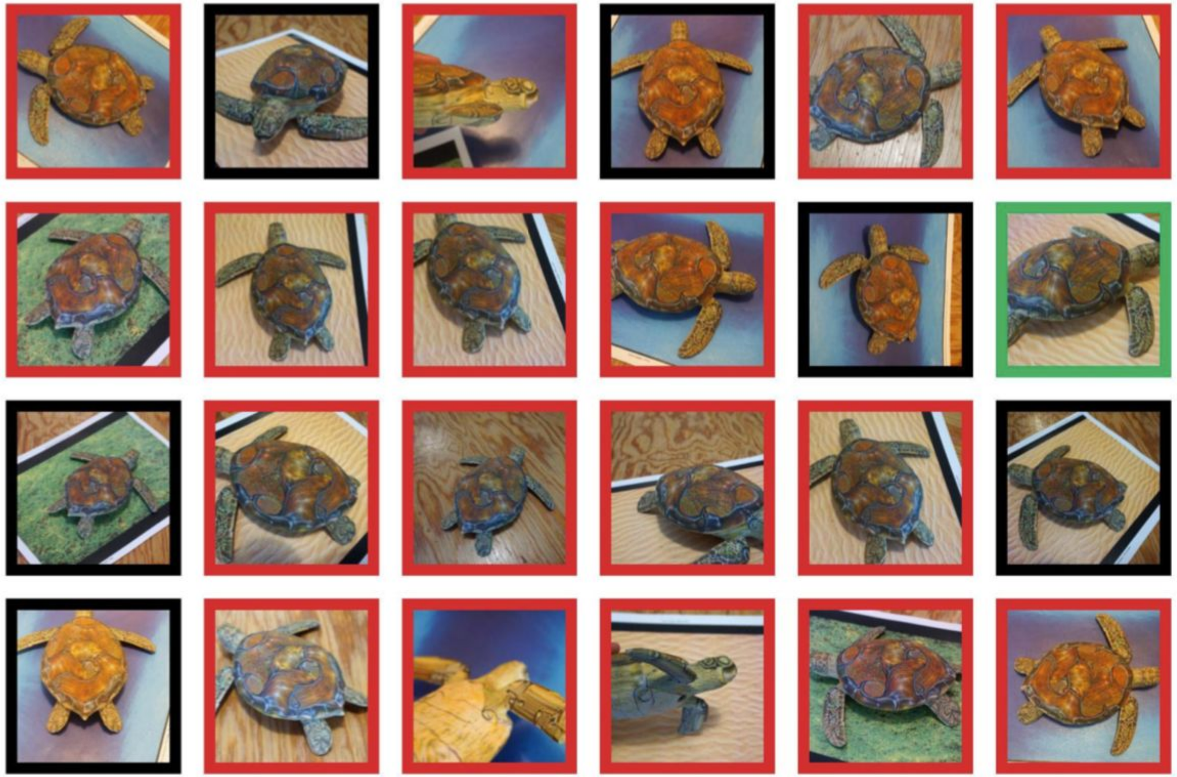
Models are not guaranteed to generalize well.



Nguyen A, Yosinski J, Clune J. Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images. In Computer Vision and Pattern Recognition (CVPR ’15), IEEE, 2015.



Nguyen A, Yosinski J, Clune J. Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images. In Computer Vision and Pattern Recognition (CVPR '15), IEEE, 2015.



🟩 classified as turtle 🟥 classified as rifle
⬛ classified as other

Anish Athalye, Logan Engstrom, Andrew Ilyas, Kevin Kwok, Synthesizing Robust Adversarial Examples.
Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

Challenges: Explainability

Decision-making is not readily explainable:

- Decision-criteria are distributed across many (tens or hundreds of thousands of) units (‘artificial neurons’)
- Eliciting specific reasons why some decision has been taken is a time-consuming challenge even for highly experienced data analysts.

Closing remarks

Take home messages

- AI has a strong foothold in Computer Vision
- Deep Learning architectures excel at achieving state-of-the-art performance in many challenging CV tasks
- Yet, there are certain risks and limitations that one needs to be aware of:
  - Limited explainability – models are largely opaque.
  - Unpredictable behavior in corner/edge cases and long tails of distributions – guardrails are often inevitable.
- Nevertheless, continuous progress is being made, and interesting new avenues keep emerging:
  - Physics-informed machine learning
  - Neurosymbolic architectures
- Exciting times (for AI in CV) to live in!

Artificial Intelligence for Images

Krzysztof Krawiec

Institute of Computing Science

Summer School on Applied and Interdisciplinary Artificial Intelligence (S2AI2)

Poznań University of Technology

4.09.2024

POLITECHNIKA POZNAŃSKA POZNAN UNIVERSITY OF TECHNOLOGY

NAWA POLISH NATIONAL AGENCY FOR ACADEMIC EXCHANGE

STER PROGRAMME