Convolutional Neural Networks for Biomedical Image Analysis

Alex Kalinin, PhD Candidate
DCM&B, University of Michigan
June 1, 2017
@alxndrkalinin

About me

- BSc, MSc in Applied Math and Informatics from Russia
- 2 years of experience in software development
- Fulbright Scholar at UCLA Statistics
- 4th year PhD Candidate in Bioinformatics
- working on
  - 3D microscopic cell image analysis (Athey Lab)
  - web-based visual analytics (SOCR)
- co-organizer of annual Ann Arbor Deep Learning event (a2dlearn)

Contact info: http://alxndrkalinin.github.io

Contents

This talk contains buzz-words

and highly non-convex objective functions

that some attendees might find disturbing.

Contents

1. **Deep Learning Introduction:**
   - Perceptron and MLP intro
   - Convolutional NN intro
   - Deep CNN
   - Tools and methods for Deep CNNs
2. **Applications of CNN to biomedical image analysis:**
   - 2D histology
   - CT scans
   - etc
3. **Example: 3D Sparse CNN for nucleus shape classification**

(Longer)
Deep Learning Intro

- Perceptron and MLP intro
- Convolutional NN intro
- Deep CNN
- Tools and methods for Deep CNNs

Artificial Neural Networks

Artificial Neural Networks — a family of biologically-inspired machine learning algorithms

- ANNs invented in 1950's
- Have been outperformed by SVM and Random Forest

2012 – AlexNet started "deep neural network renaissance"

Why is it working now:

- lots of [labeled] data
- computing power

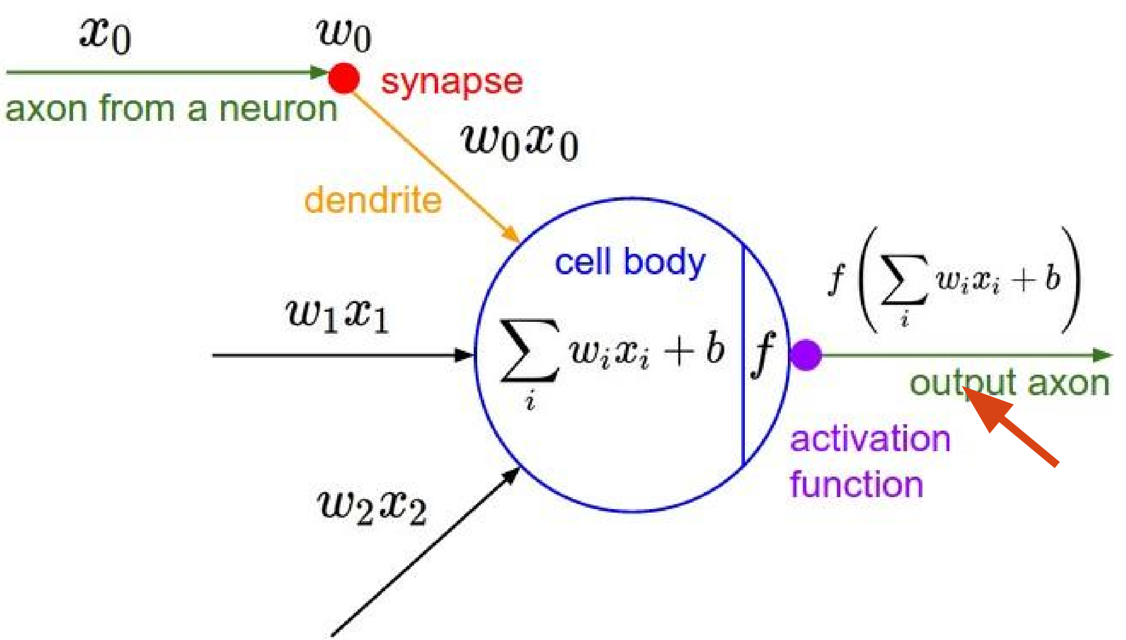
Modeling one neuron

[very] coarse model:

- synaptic strengths (the weights *w*) are learnable and control the strength of influence

- dendrites carry the signal to the cell body where they all get summed

- if the final sum is above a certain threshold, the neuron can fire, sending a spike along its axon

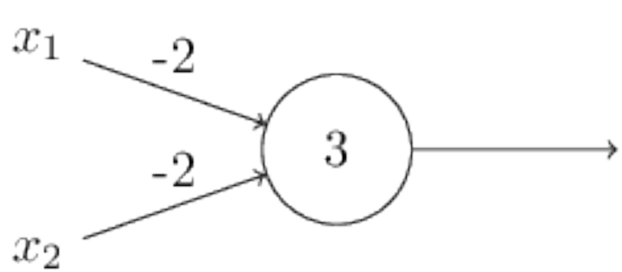


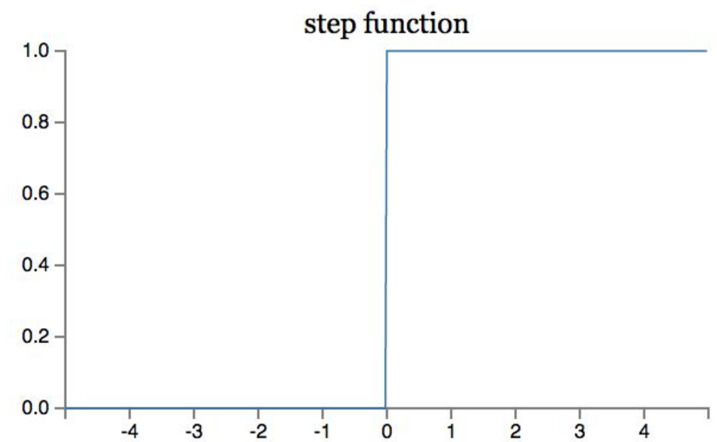
Perceptron: step activation function

Consider perceptron consisting of 1 neuron:

- 2 binary inputs, each with known weight = -2
- bias = 3
- step activation function with binary output

$$f\left(\sum_i w_i x_i + b\right)$$





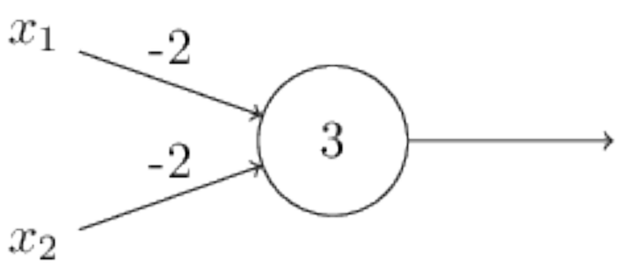
Michael A. Nielsen, "Neural Networks and Deep Learning", Determination Press, 2015

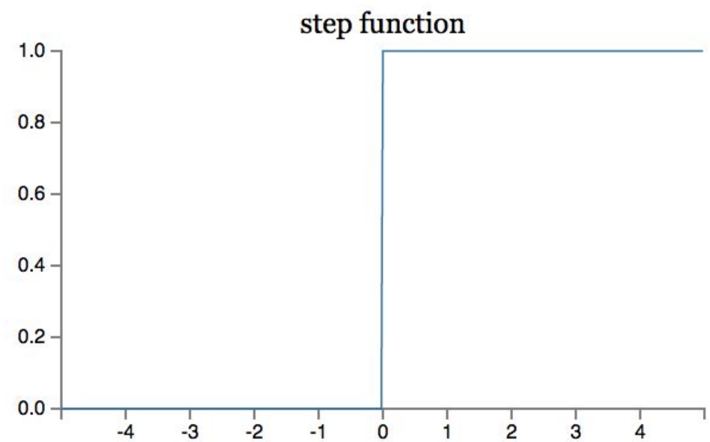
Perceptron: NAND example

Consider perceptron consisting of 1 neuron:

$$f\left(\sum_i w_i x_i + b\right)$$

- 2 binary inputs, each with known weight = -2
- bias = 3
- step activation function with binary output





Consider binary inputs:

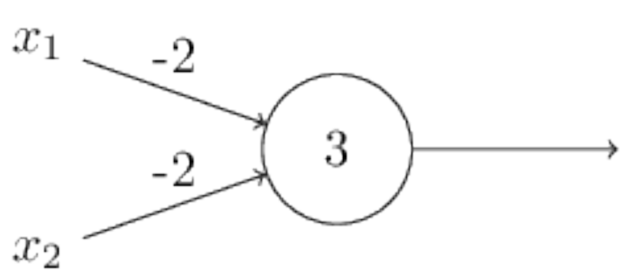
- [0,0]: $(-2) * 0 + (-2) * 0 + 3 = 3$ – outputs 1

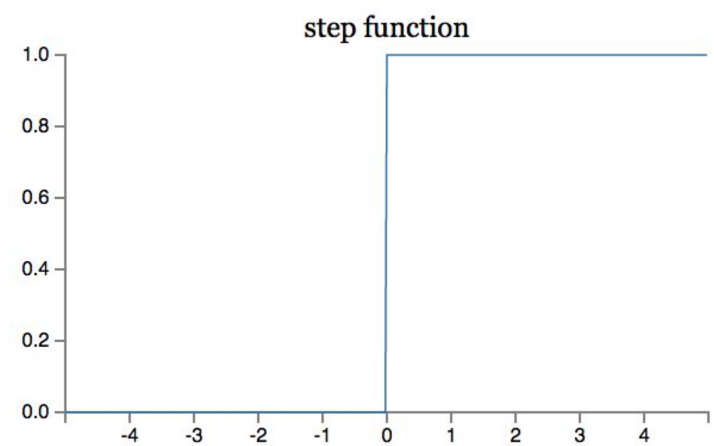
Perceptron: NAND example

Consider perceptron consisting of 1 neuron:

- 2 binary inputs, each with known weight = -2
- bias = 3
- step activation function with binary output

$$f\left(\sum_i w_i x_i + b\right)$$





Consider binary inputs:

- [0,0]: $(-2) * 0 + (-2) * 0 + 3 = 3$ – outputs 1
- [0,1]: $(-2) * 0 + (-2) * 1 + 3 = 1$ – outputs 1

Michael A. Nielsen, "Neural Networks and Deep Learning", Determination Press, 2015

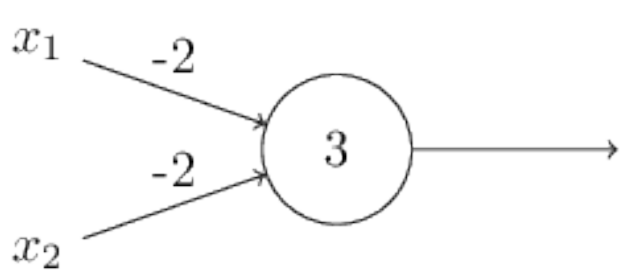
@alxndrkalinin

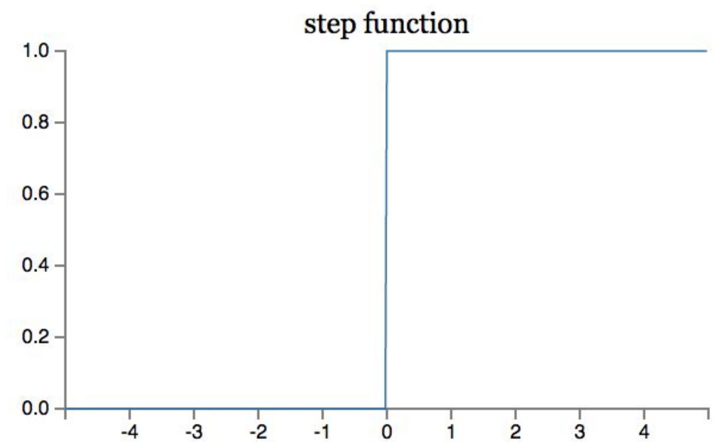
Perceptron: NAND example

Consider perceptron consisting of 1 neuron:

- 2 binary inputs, each with known weight = -2
- bias = 3
- step activation function with binary output

$$f\left(\sum_i w_i x_i + b\right)$$





Consider binary inputs:

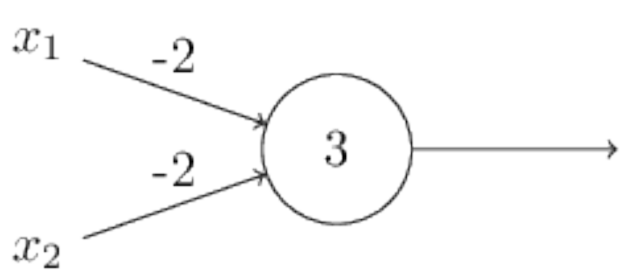
- [0,0]: $(-2) * 0 + (-2) * 0 + 3 = 3$ – outputs 1
- [0,1]: $(-2) * 0 + (-2) * 1 + 3 = 1$ – outputs 1
- [1,0]: $(-2) * 1 + (-2) * 0 + 3 = 1$ – outputs 1

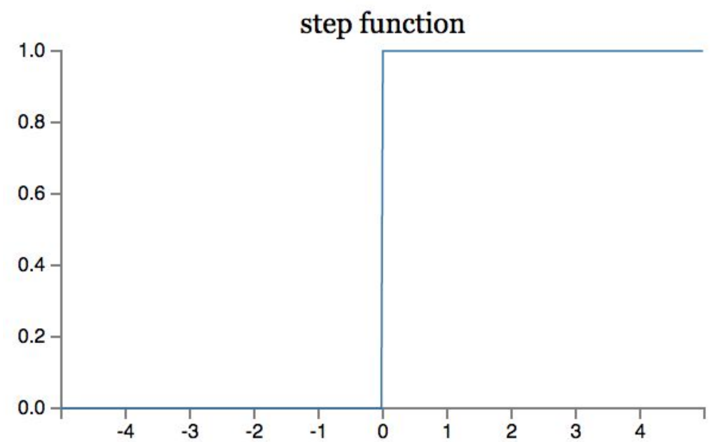
Perceptron: NAND example

Consider perceptron consisting of 1 neuron:

$$f\left(\sum_i w_i x_i + b\right)$$

- 2 binary inputs, each with known weight = -2
- bias = 3
- step activation function with binary output





Consider binary inputs:

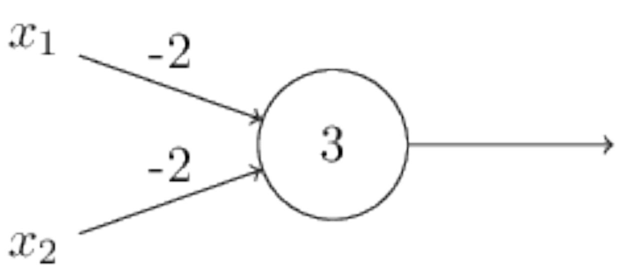
- [0,0]: $(-2) * 0 + (-2) * 0 + 3 = 3$ – outputs 1
- [0,1]: $(-2) * 0 + (-2) * 1 + 3 = 1$ – outputs 1
- [1,0]: $(-2) * 1 + (-2) * 0 + 3 = 1$ – outputs 1
- [1,1]: $(-2) * 1 + (-2) * 1 + 3 = -1$ – outputs 0

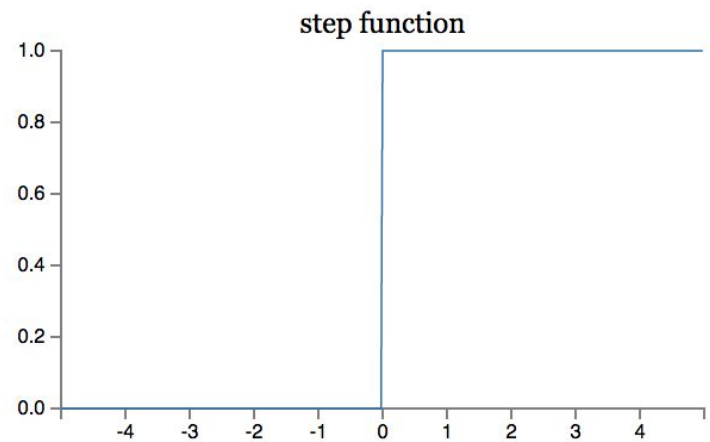
Perceptron: NAND example

Consider perceptron consisting of 1 neuron:

$$f\left(\sum_i w_i x_i + b\right)$$

- 2 binary inputs, each with known weight = -2
- bias = 3
- step activation function with binary output



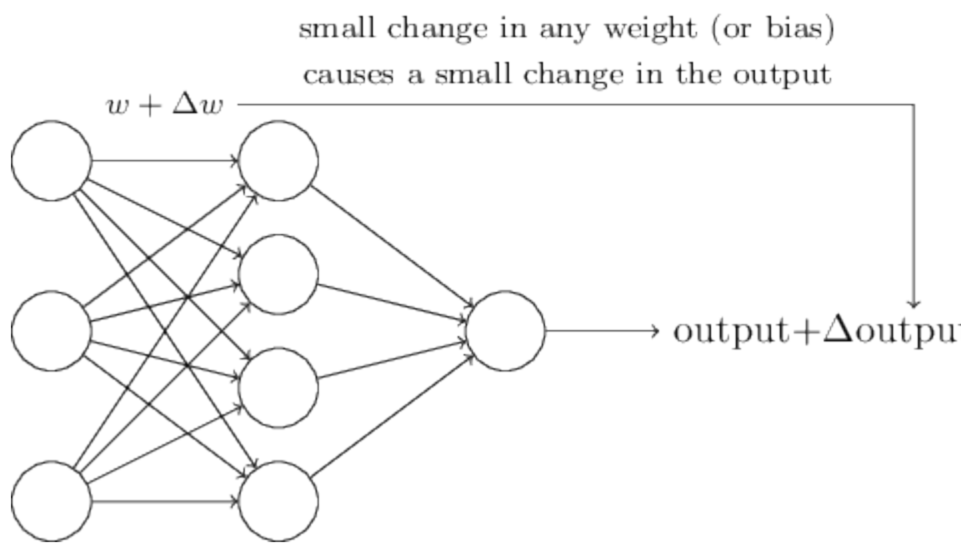


Consider binary inputs:
- [0,0]: $(-2) * 0 + (-2) * 0 + 3 = 3$ – outputs 1
- [0,1]: $(-2) * 0 + (-2) * 1 + 3 = 1$ – outputs 1
- [1,0]: $(-2) * 1 + (-2) * 0 + 3 = 1$ – outputs 1
- [1,1]: $(-2) * 1 + (-2) * 1 + 3 = -1$ – outputs 0

INPUT		OUTPUT
A	B	A NAND B
0	0	1
0	1	1
1	0	1
1	1	0

Learning weights in ANNs

If a small change in a weight => a small change in output,
then we could use this info to iteratively modify the weights
to get our network to fit the data. This is **learning**.



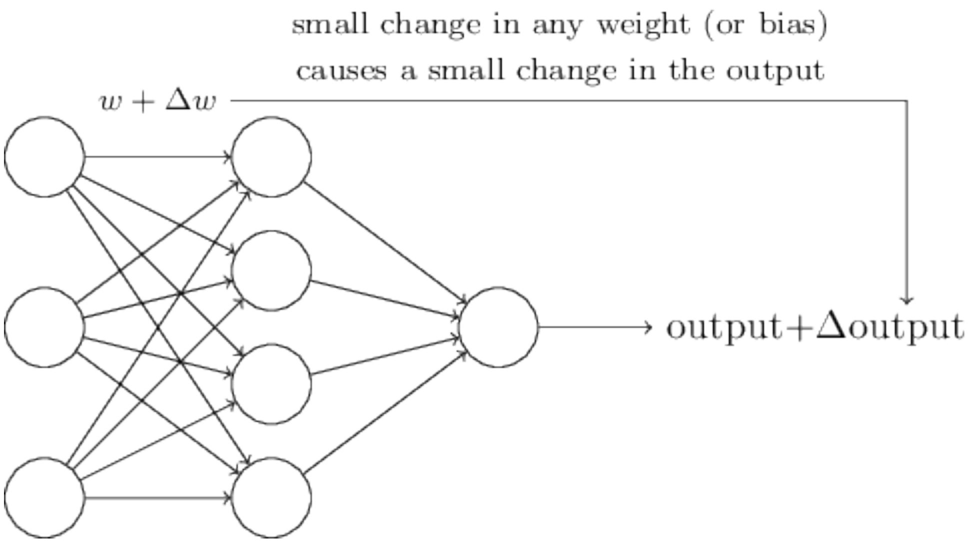
Michael A. Nielsen, "Neural Networks and Deep Learning",
Determination Press, 2015

@alxndrkalinin

Learning weights in ANNs

If a small change in a weight => a small change in output,
then we could use this info to iteratively modify the weights
to get our network to fit the data. This is **learning**.

Cannot learn in perceptrons since output is binary – impossible to know how much small change of weights changes output.

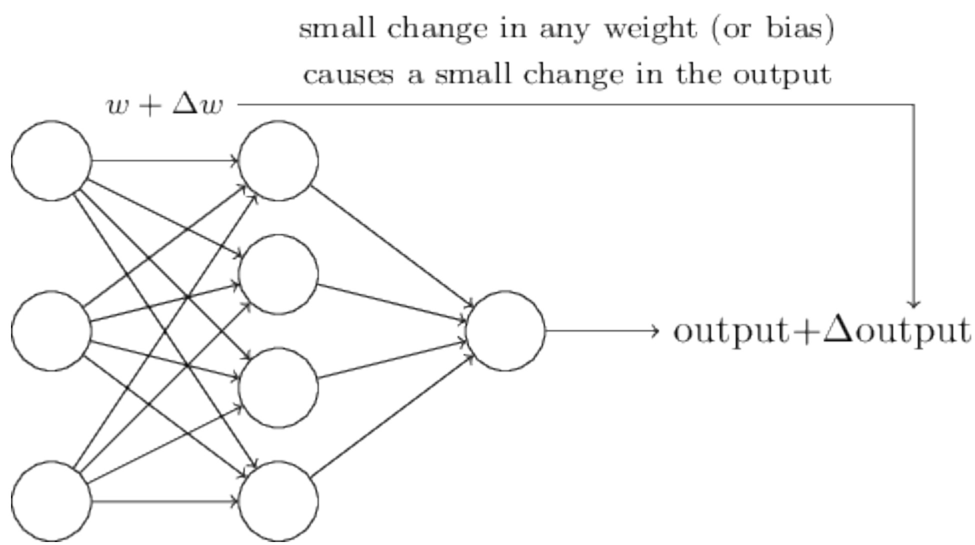


Michael A. Nielsen, "Neural Networks and Deep Learning", Determination Press, 2015

Learning weights in ANNs

If a small change in a weight => a small change in output, then we could use this info to iteratively modify the weights to get our network to fit the data. This is **learning**.

Cannot learn in perceptrons since output is binary – impossible to know how much small change of weights changes output.



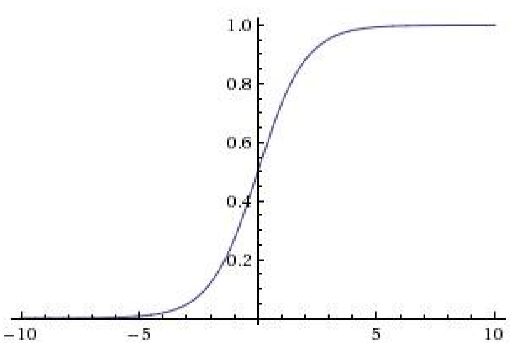
Michael A. Nielsen, "Neural Networks and Deep Learning", Determination Press, 2015

Common activation functions

Sigmoid:

squashes real numbers to range between [0,1]

not 0-centered



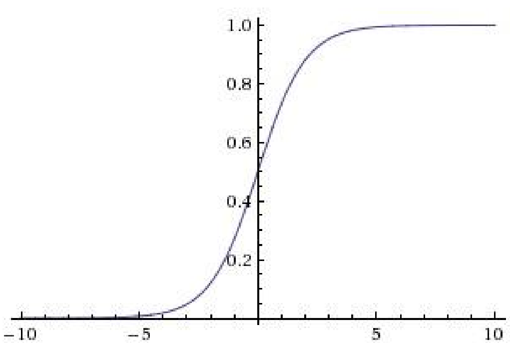
$$\sigma(x) = 1/(1 + e^{-x})$$

Common activation functions

Sigmoid:

squashes real numbers to range between [0,1]

not 0-centered



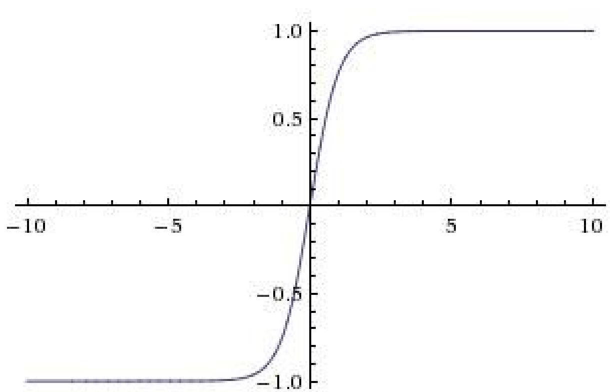
$$\sigma(x) = 1/(1 + e^{-x})$$

Tanh:

squashes real numbers to range between [-1,1]

0-centered

$$\tanh(x) = 2\sigma(2x) - 1$$



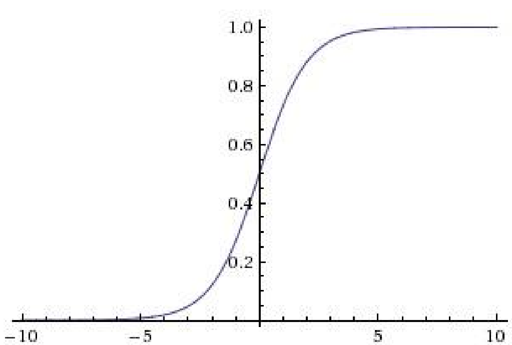
https://cs231n.github.io/neural-networks-1/

Common activation functions

Sigmoid:

squashes real numbers to range between [0,1]

not 0-centered



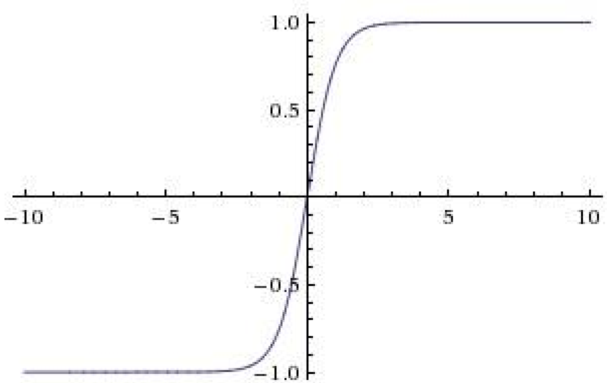
$$\sigma(x) = 1/(1 + e^{-x})$$

Tanh:

squashes real numbers to range between [-1,1]

0-centered

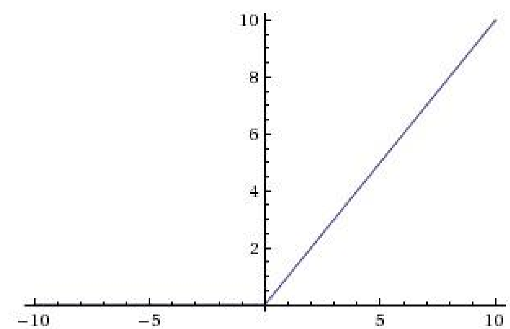
$$\tanh(x) = 2\sigma(2x) - 1$$



ReLU (Rectified Linear Unit):

zero when *x < 0* and then linear with slope 1 when *x > 0*

simple – 6x improvement in convergence

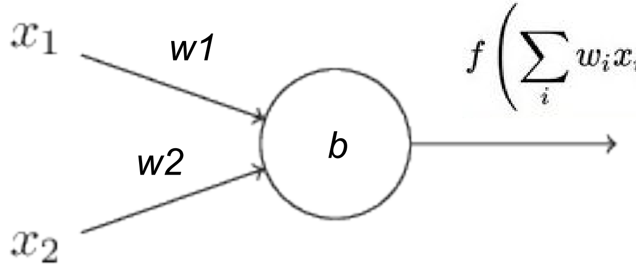


$$f(x) = \max(0, x)$$

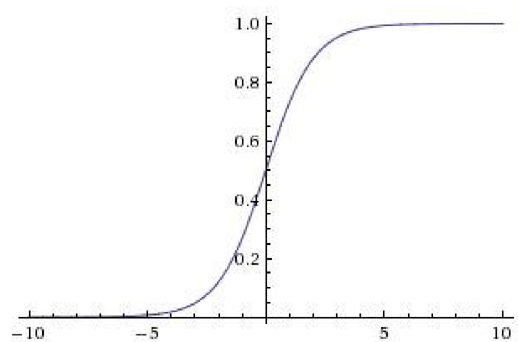
https://cs231n.github.io/neural-networks-1/

Single neuron sigmoid binary classifier

Binary output – binary logistic classification



$$f\left(\sum_i w_i x_i + b\right)$$



$$\frac{1}{1 + \exp(-\sum_j w_j x_j - b)}$$

Probabilistic interpretation:

$P(y_i = 1 \mid x_i; w)$ - output of neuron, probability for class 1

$P(y_i = 0 \mid x_i; w) = 1 - P(y_i = 1 \mid x_i; w)$ - probability for class 0

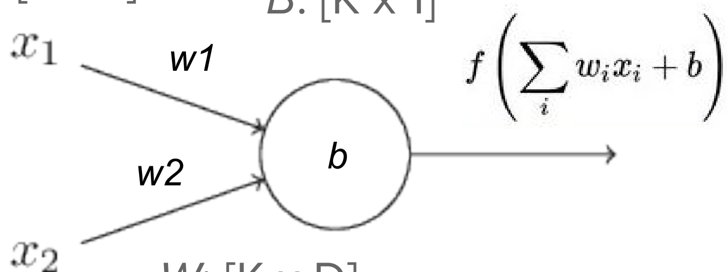
Parameters by iteratively minimizing cost function, e.g. the negative log likelihood of the correct class (MLE) with gradient-based methods

Single neuron sigmoid classifier

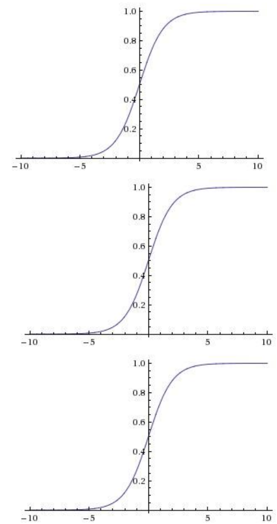
K classes – generalization of binary logistic classification to multinomial

X: [D x 1]

B: [K x 1]



W: [K x D]



outputs [K x 1] vector of probabilities

Probabilistic interpretation:

$$P(y_i \mid x_i; W) = \frac{e^{f_{y_i}}}{\sum_j e^{f_j}}$$

\- normalized probability of each class

Estimation by iteratively minimizing the negative log likelihood of the correct class (MLE) with gradient-based methods

(Shorter)
Deep Learning Intro

- single neuron classifier
- multilayer neural network
- convolutional neural network

Artificial Neural Networks

Artificial Neural Networks — a family of biologically-inspired machine learning algorithms

- ANNs invented in 1950's
- Have been outperformed by SVM and Random Forest

2012 – AlexNet started "deep neural network renaissance"

Why is it working now:

- lots of [labeled] data
- computing power

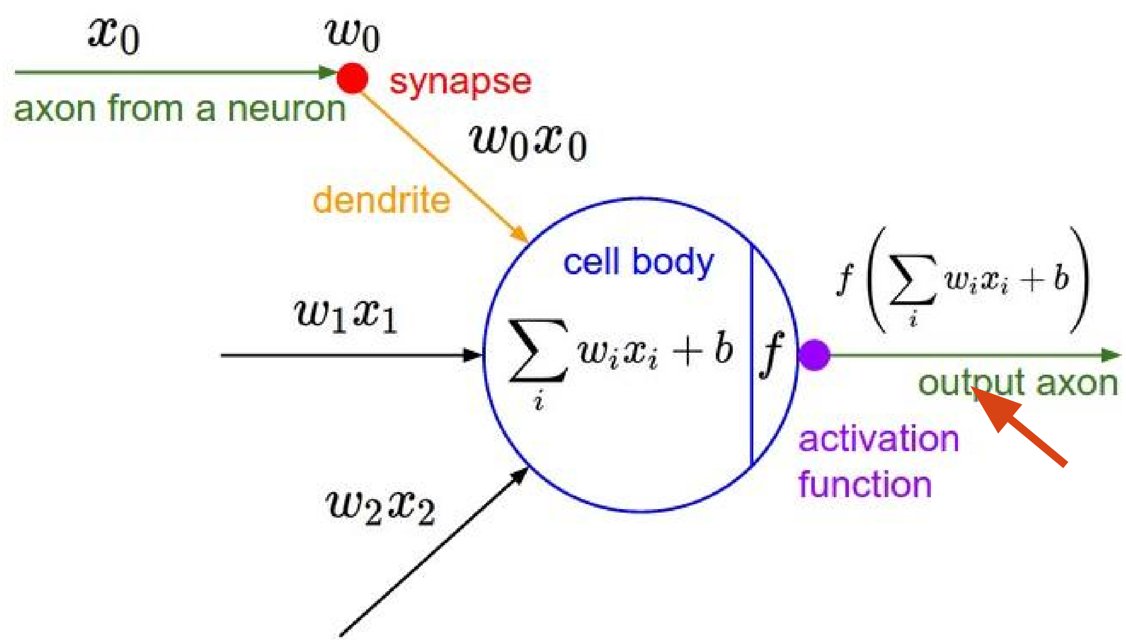
Modeling one neuron

[very] coarse model:

- synaptic strengths (the weights *w*) are learnable and control the strength of influence

- dendrites carry the signal to the cell body where they all get summed

- if the final sum is above a certain threshold, the neuron can fire, sending a spike along its axon

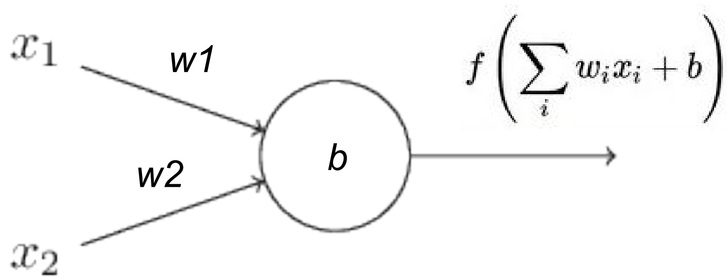


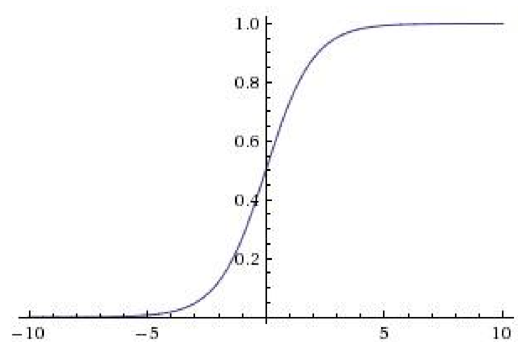
https://cs231n.github.io/neural-networks-1/

@alxndrkalinin

Single neuron sigmoid binary classifier

Binary output – binary logistic classification





$$\frac{1}{1 + \exp(-\sum_j w_j x_j - b)}$$

Probabilistic interpretation:

$P(y_i = 1 \mid x_i; w)$ - output of neuron, probability for class 1

$P(y_i = 0 \mid x_i; w) = 1 - P(y_i = 1 \mid x_i; w)$ - probability for class 0

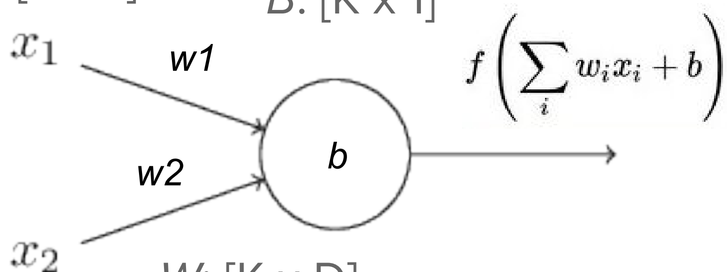
Parameters by iteratively minimizing cost function, e.g. the negative log likelihood of the correct class (MLE) with gradient-based methods

Single neuron sigmoid classifier

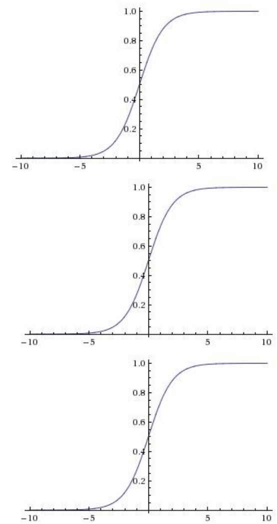
K classes – generalization of binary logistic classification to multinomial

X: [D x 1]

B: [K x 1]



W: [K x D]



outputs [K x 1] vector of probabilities

Probabilistic interpretation:

$$P(y_i \mid x_i; W) = \frac{e^{f_{y_i}}}{\sum_j e^{f_j}}$$

\- normalized probability of each class

Estimation by iteratively minimizing the negative log likelihood of the correct class (MLE) with gradient-based methods

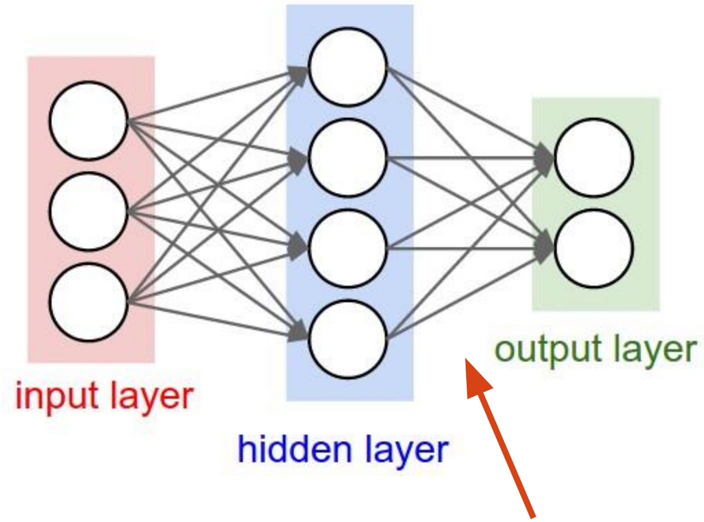
@alxndrkalinin

Multilayer Neural Networks

Fully-connected layer:

- full pairwise connection of all units (neurons) in adjacent layers

2-layer network:



combinations of “neuron” outputs

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep learning. MIT Press, 2016.

Multilayer Neural Networks

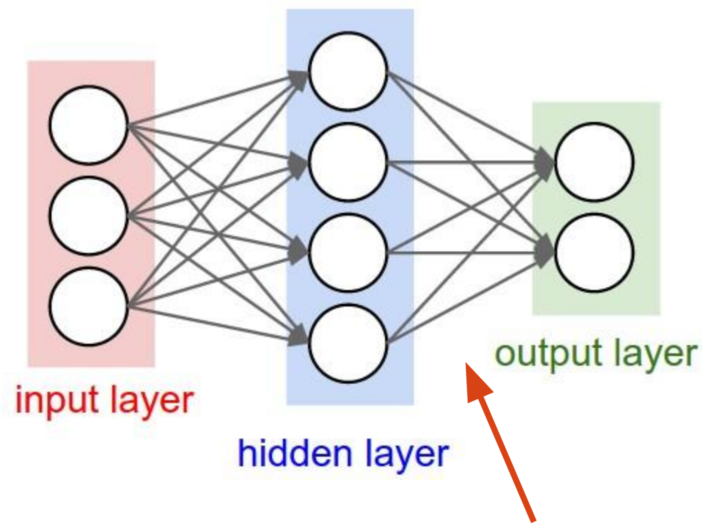
Fully-connected layer:

- full pairwise connection of all units (neurons) in adjacent layers

Width vs depth:

- NNs with at least one hidden layer are **universal approximators**, i.e. given enough hidden units 2-layer network can approximate **any** continuous function (good for complex problem such as image classification).

2-layer network:



combinations of “neuron” outputs

Hornik, Kurt, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators." Neural networks 2.5 (1989): 359-366. – 13700 citations

Multilayer Neural Networks

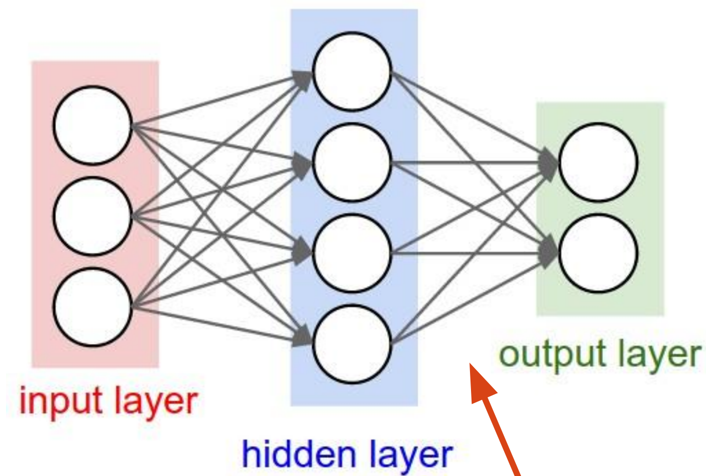
Fully-connected layer:

- full pairwise connection of all units (neurons) in adjacent layers

Width vs depth:

- NNs with at least one hidden layer are **universal approximators**, i.e. given enough hidden units 2-layer network can approximate **any** continuous function.

2-layer network:



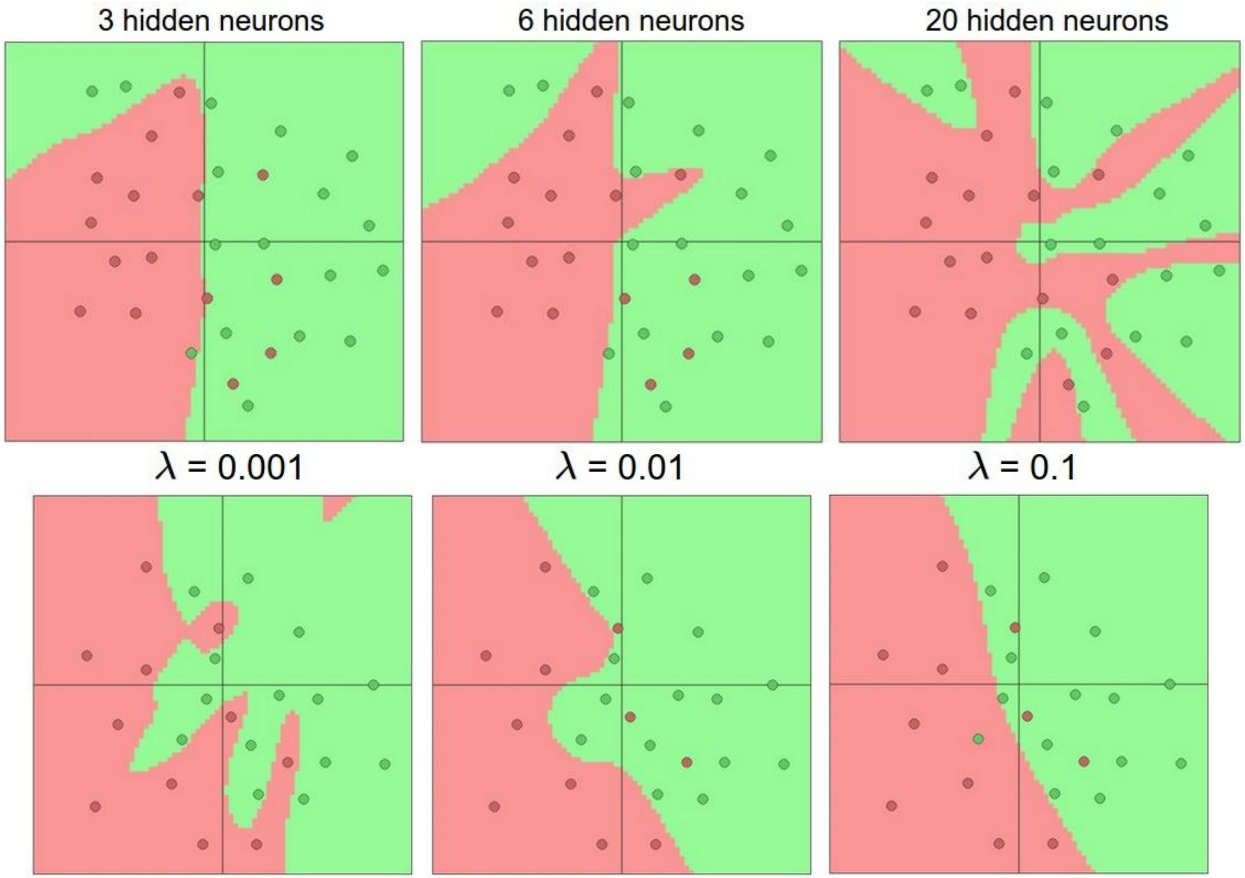
combinations of “neuron” outputs

However,
- not guaranteed that the training algorithm will be able to learn that function
- the layer may be unfeasibly large and may fail to generalize correctly

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep learning. MIT Press, 2016.

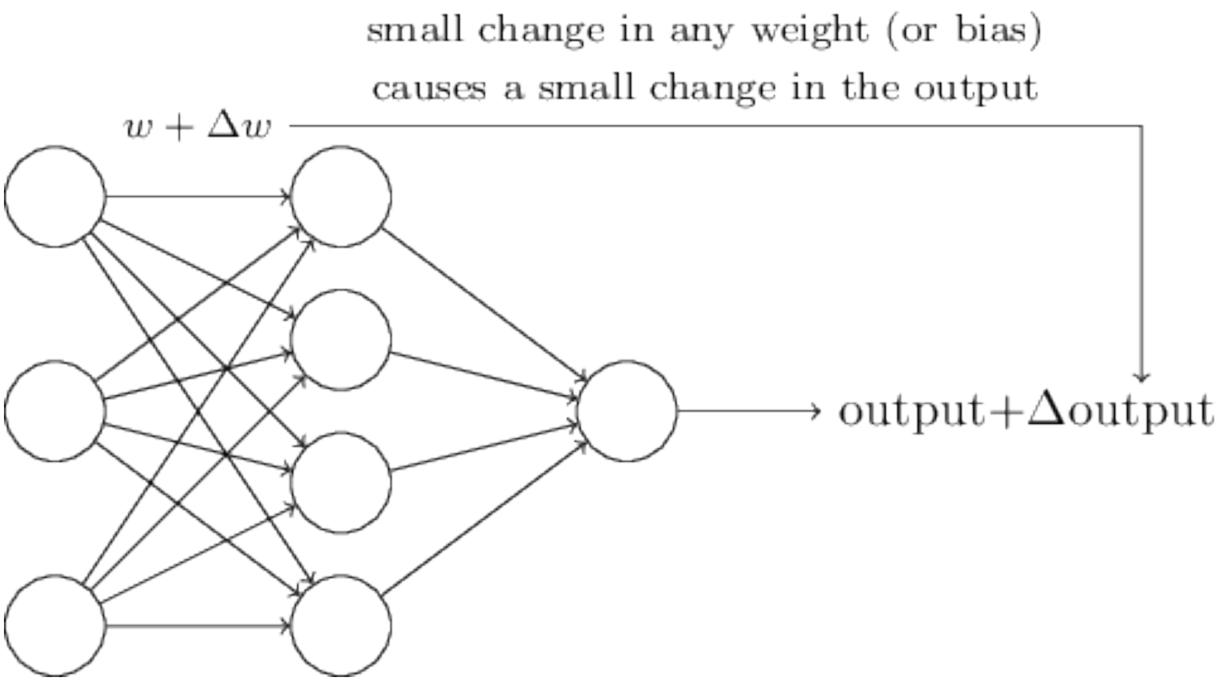
Deep Fully-Connected Networks

Depth also increases the capacity of the network, i.e. representation power
However, it can lead to overfitting - requires regularization (*L2*, dropout, etc.)



Learning weights in ANNs

If a small change in a weight => a small change in output, then we could use this info to iteratively modify the weights to get our network to fit the data. This is **learning**.

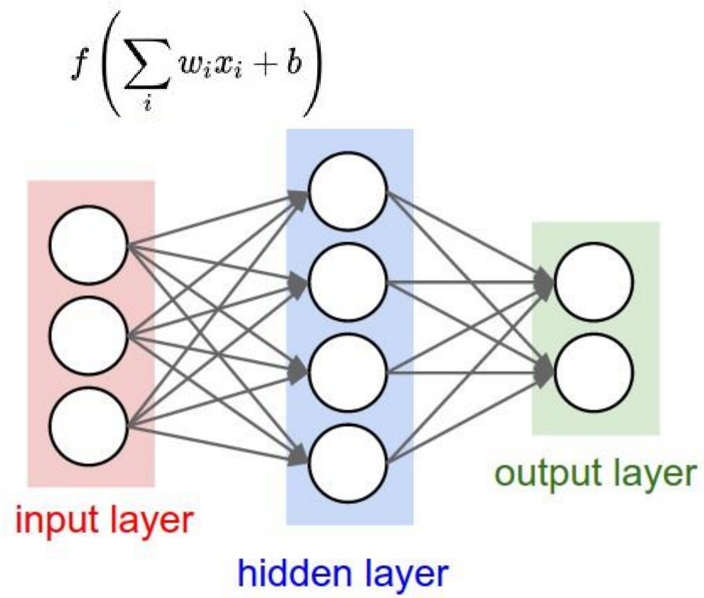


Michael A. Nielsen, "Neural Networks and Deep Learning", Determination Press, 2015

Learning in Multilayer Neural Networks

How it learns:

- weights are randomly initialized
- first, forward pass is performed with fixed weight values (w, b)
- gradient of the loss function is computed given input
- gradients are "back-propagated" through network with backprop algorithm
- weights are updated using optimization technique (e.g. SGD)



https://cs231n.github.io/neural-networks-3/

Convolutional Neural Network (CNN)

Motivation:

- fully-connected (FC) multilayer networks don't scale for images
    - e.g., for 256x256x3 RGB image 1 fully-connected neuron in the first hidden layer has 196608 parameters (* # of neurons * # of layers)
    - simple idea: restrict connections between neurons, such that each hidden unit to connect to only a small subset of the input units

- in images we also want to take advantage of structures within local region, i.e. detect 2D/3D features directly vs. "unrolling" images into "flat" feature vectors

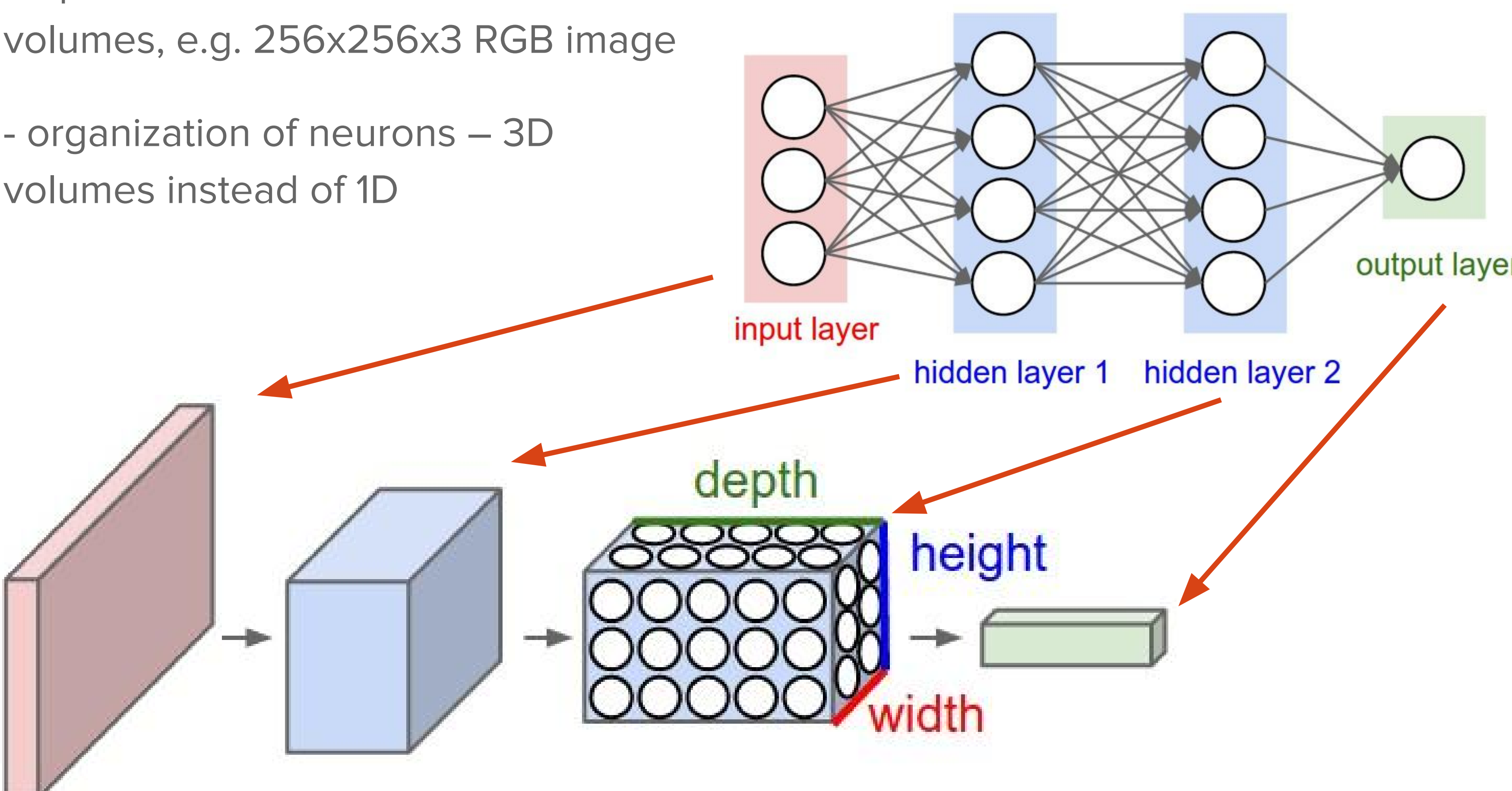
- images are "stationary" i.e. features that we learn at one part of the image can also be applied to other parts of the image (e.g. edges, etc)

CNN: 3D Structure

- inputs are served as 2D or 3D volumes, e.g. 256x256x3 RGB image

- organization of neurons – 3D volumes instead of 1D

3-layer FC network:



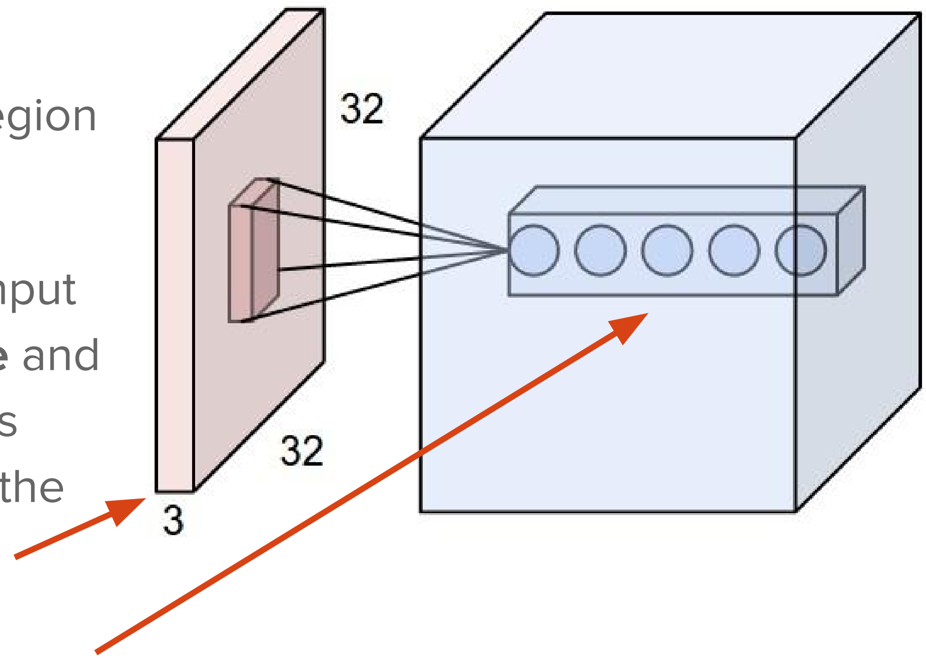
CNN: Local Connectivity

It is impractical to connect neurons to all neurons in the previous layer.

Trick #1:

connect each neuron to only a local region of the input volume

- how much the neuron “sees” of input called **receptive field** or **filter size** and equals to number of weights it has
- in depth axis it is always equal to the depth of the input volume (e.g. 3)
- depth of the output volume is a hyperparameter: it corresponds to the number of filters we want to learn



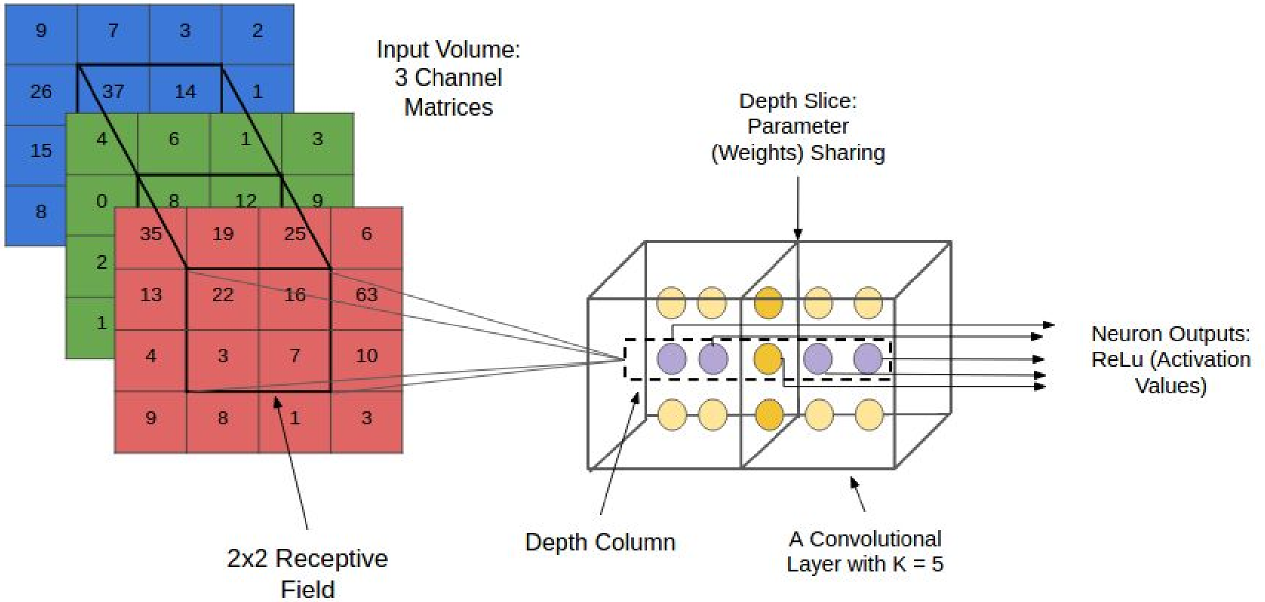
@alxndrkalinin

CNN: Parameter Sharing

- local connectivity reduces the number of connections from each neuron
- there are still many parameters overall, eps. when learning many filters (depth)
- images are “stationary”: if feature is useful to compute at some spatial position (x1,y2), then it should also be useful to compute at a different position (x2,y2)

Trick #2:

constrain the neurons
in each depth slice
to use the same weights



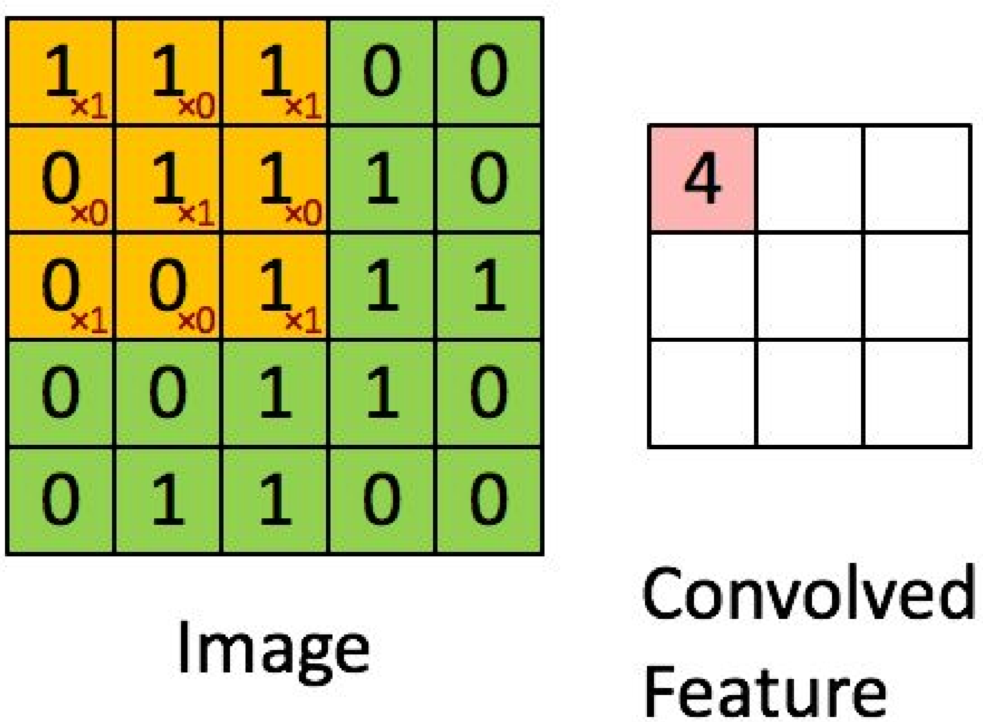
Output can be computed as convolution of the neuron’s weights with the input.

http://xrds.acm.org/blog/2016/06/convolutional-neural-networks-cnns-illustrated-explanation/

@alxndrkalinin

Convolution

Simple example of 2D convolution

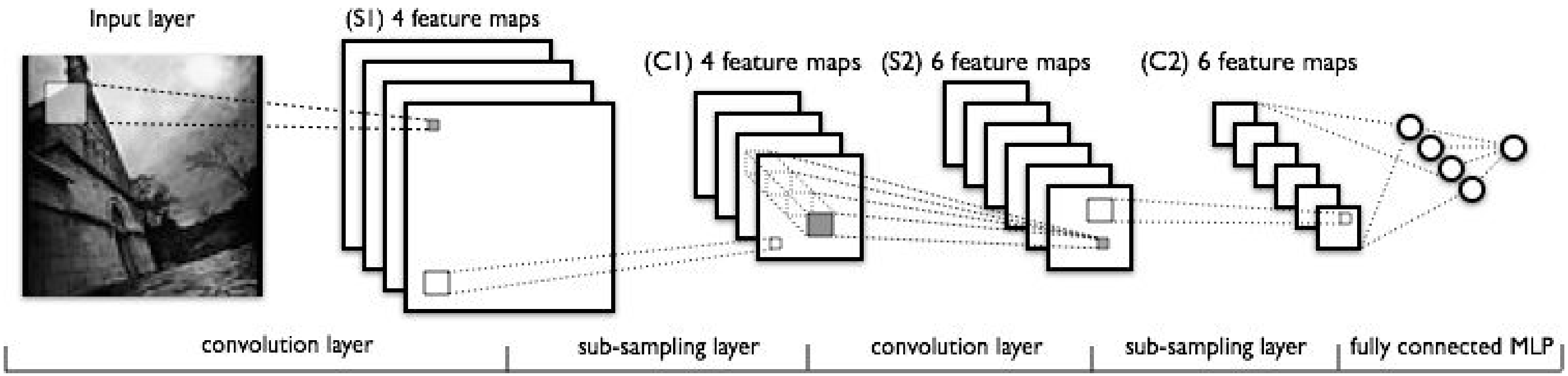


http://ufldl.stanford.edu/tutorial/supervised/FeatureExtractionUsingConvolution/

@alxndrkalinin

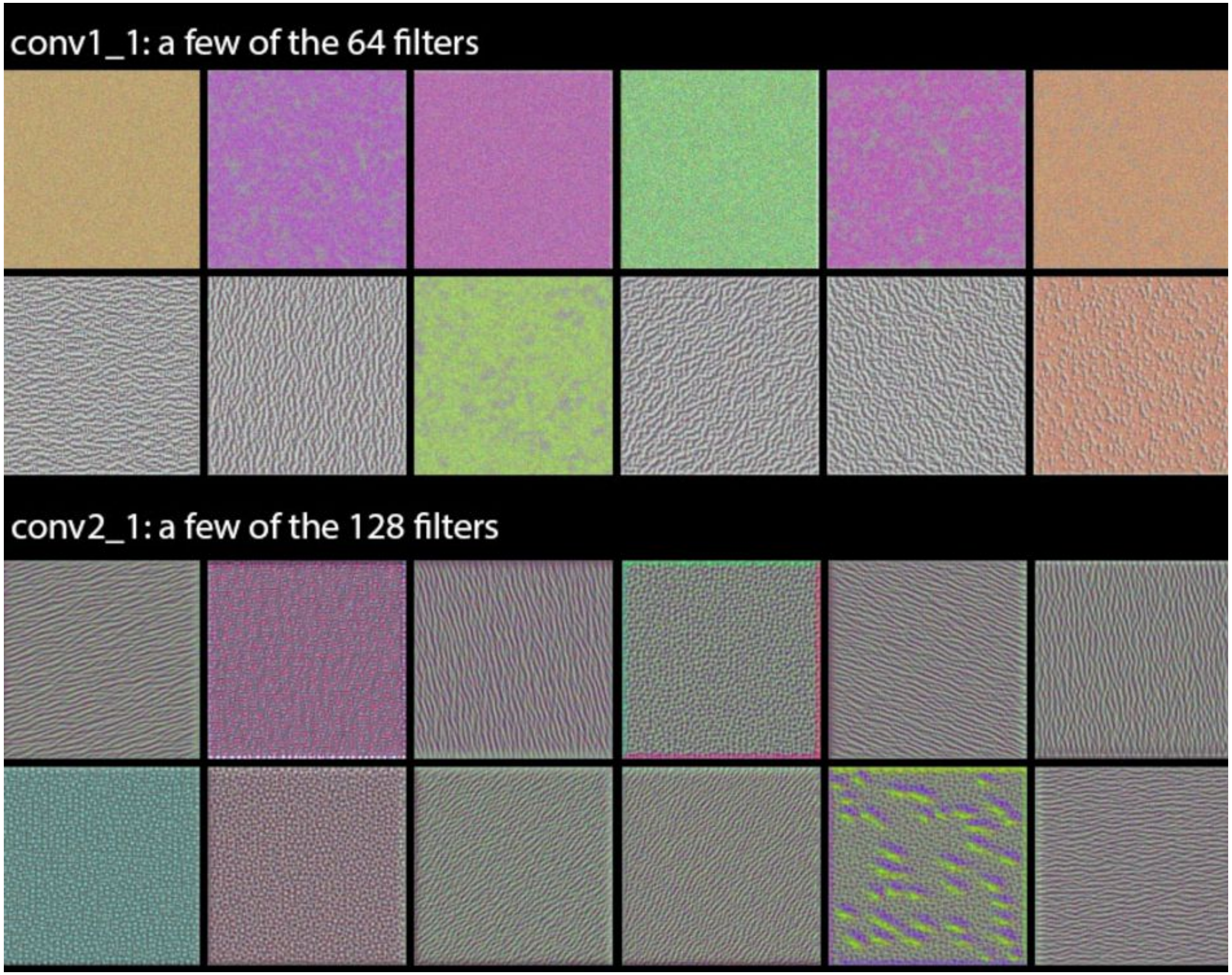
Simple CNN architecture

LeNet, 1998 – 5 layers



LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998d). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278–2324.

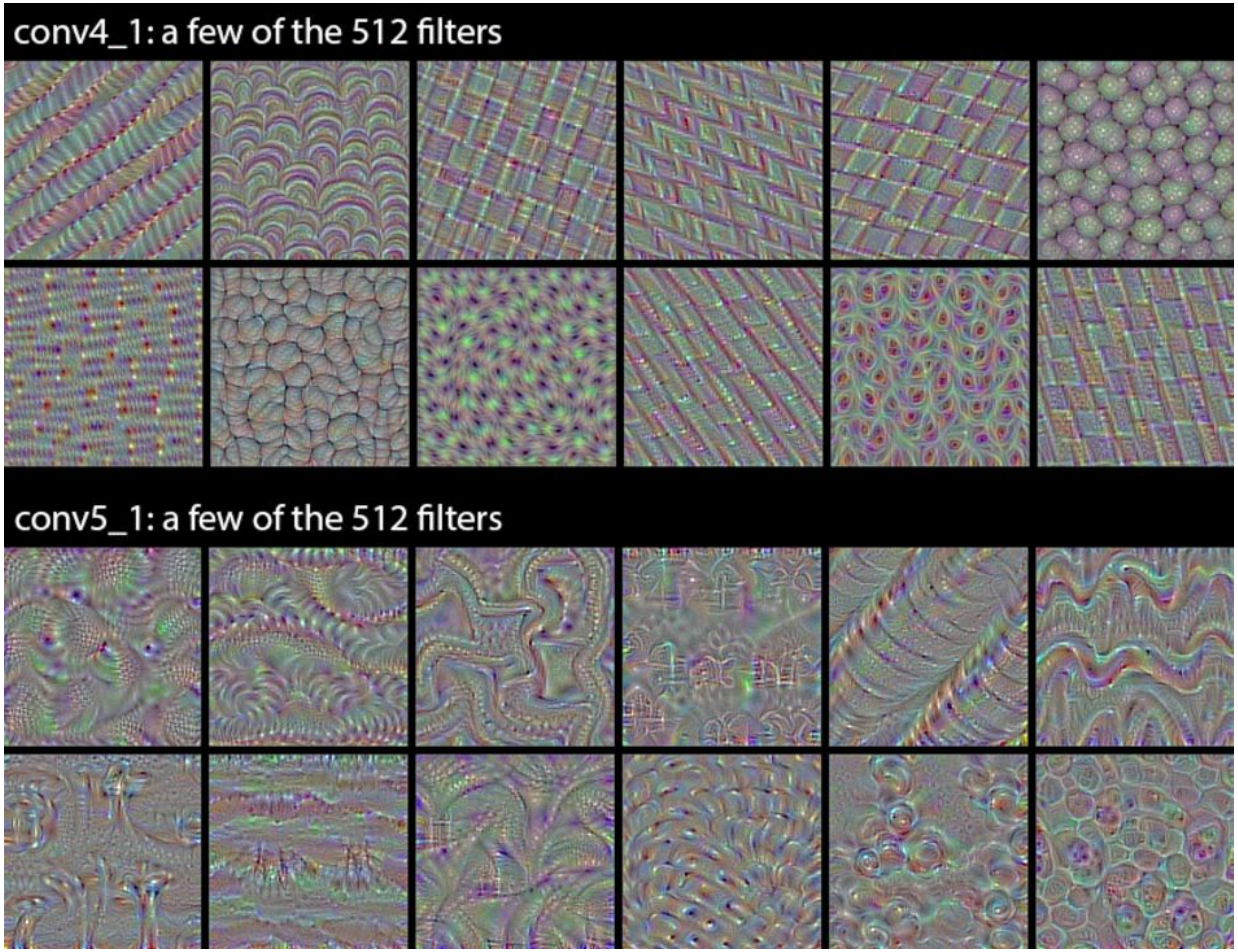
What does it learn? Visualizing filters



https://blog.keras.io/how-convolutional-neural-networks-see-the-world.html

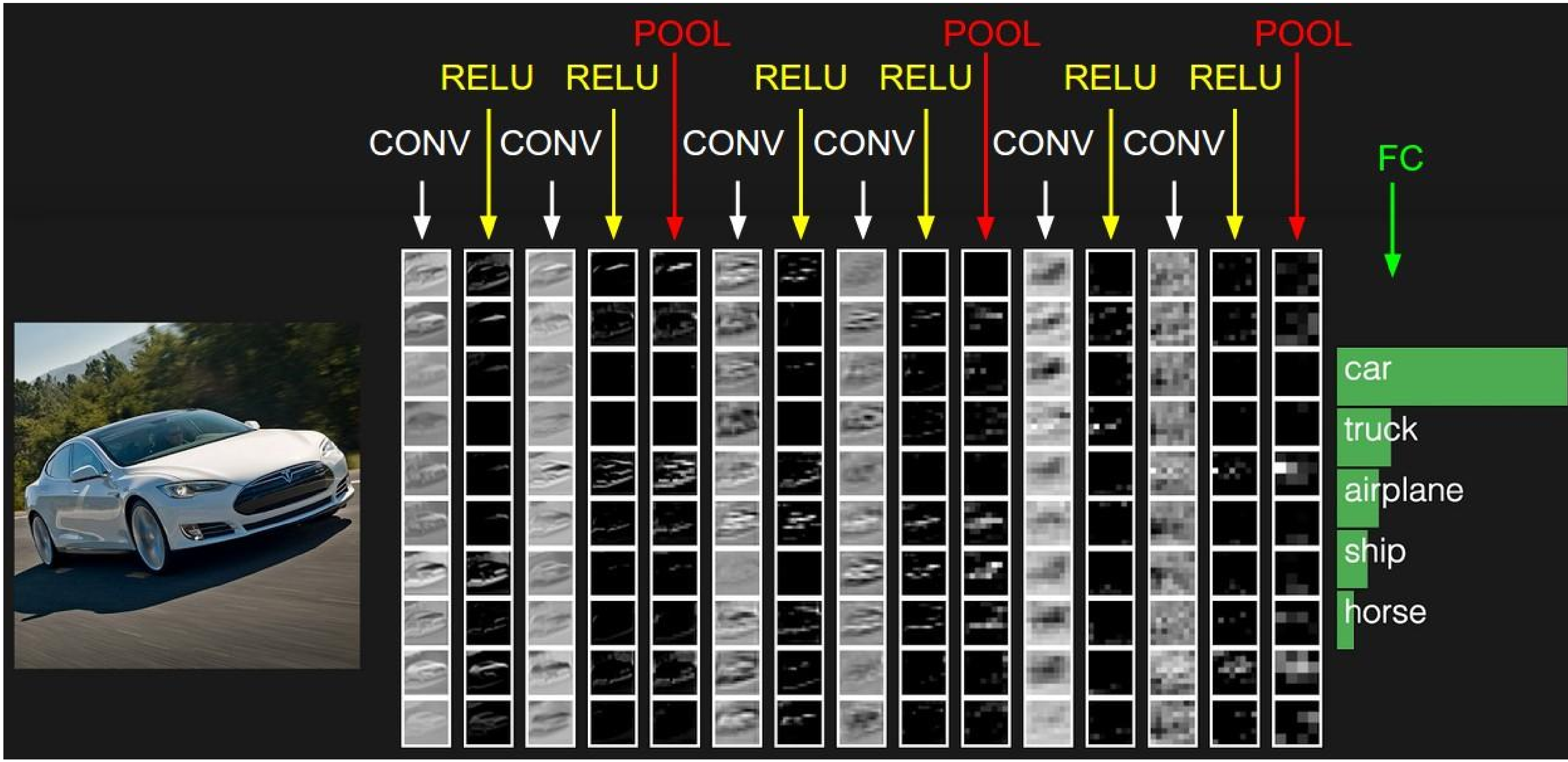
@alxndrkalinin

What does it learn? Visualizing filters



https://blog.keras.io/how-convolutional-neural-networks-see-the-world.html

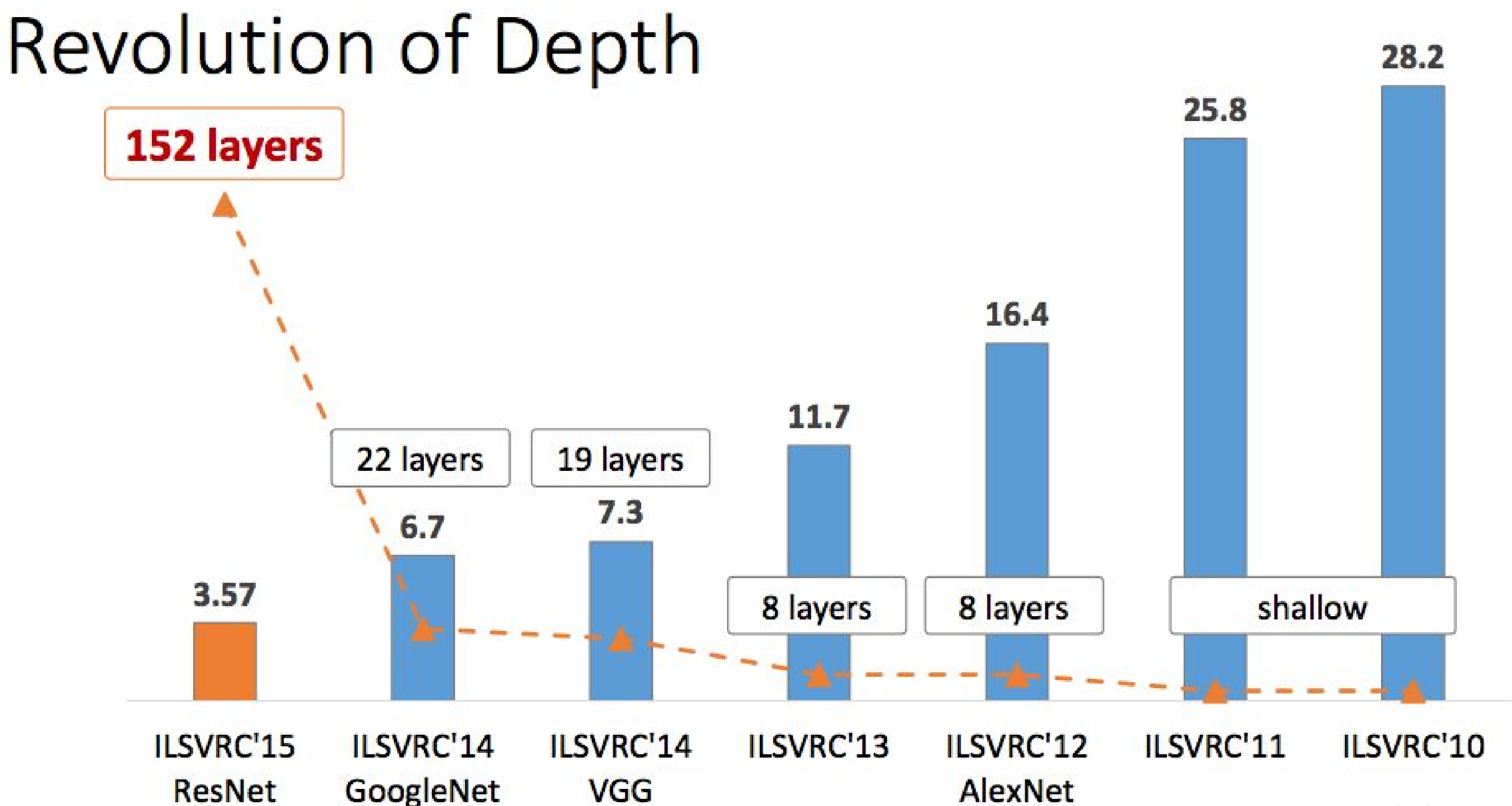
Visualizing activations



https://cs231n.github.io/convolutional-networks/

How deep is deep?

VGG-19 (2014) – 19 layers, “very deep convolutional neural network”
Deep Residual Networks with Stochastic Depth (2016) – more than **1200 layers**



@alxndrkalinin

CNN Structure Summary

- consist of multiple layers of nonlinear processing units organized in volumes
- each neuron is connected to only a local region of the input volume
- neurons in each depth level/slice share same weights
- supervised learning of feature representations in each layer, with the layers forming a hierarchy from low-level to high-level features

Achievements and Challenges

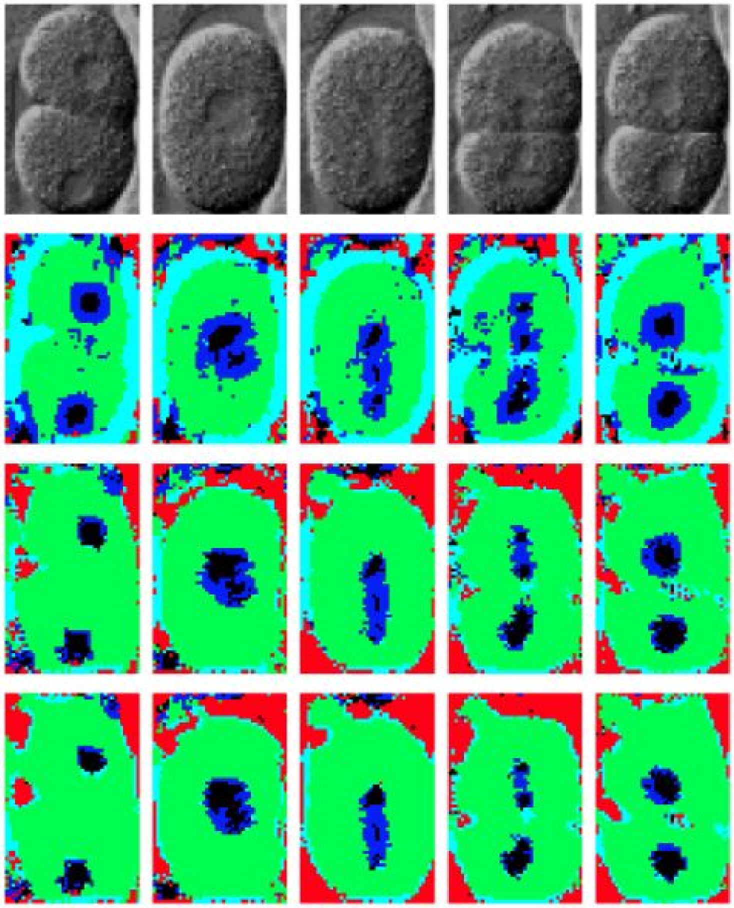
\+ SOTA in many hard CV problems on natural images and videos

\+ Human-level or better in problems like classification, segmentation, tracking

- Requires lots of labeled data
  - data augmentation, regularization, and transfer learning help with smaller data
- Unsupervised learning still works poorly
  - this is one of the hottest areas of research right now
- Not a silver bullet, fairly complicated type of models
  - if nothing else works, CNN quite possible be better, e.g. with images
- Requires special hardware (GPU)
  - Intel is working on improving CPU architectures/instructions for DL
- Hard to interpret
  - there is a large body of recent work in theory, visualization, and interpretation of CNNs (VGG-CAM, LIME, etc)

2. Applications bioimage analysis

Automatic phenotyping of embryos, 2005



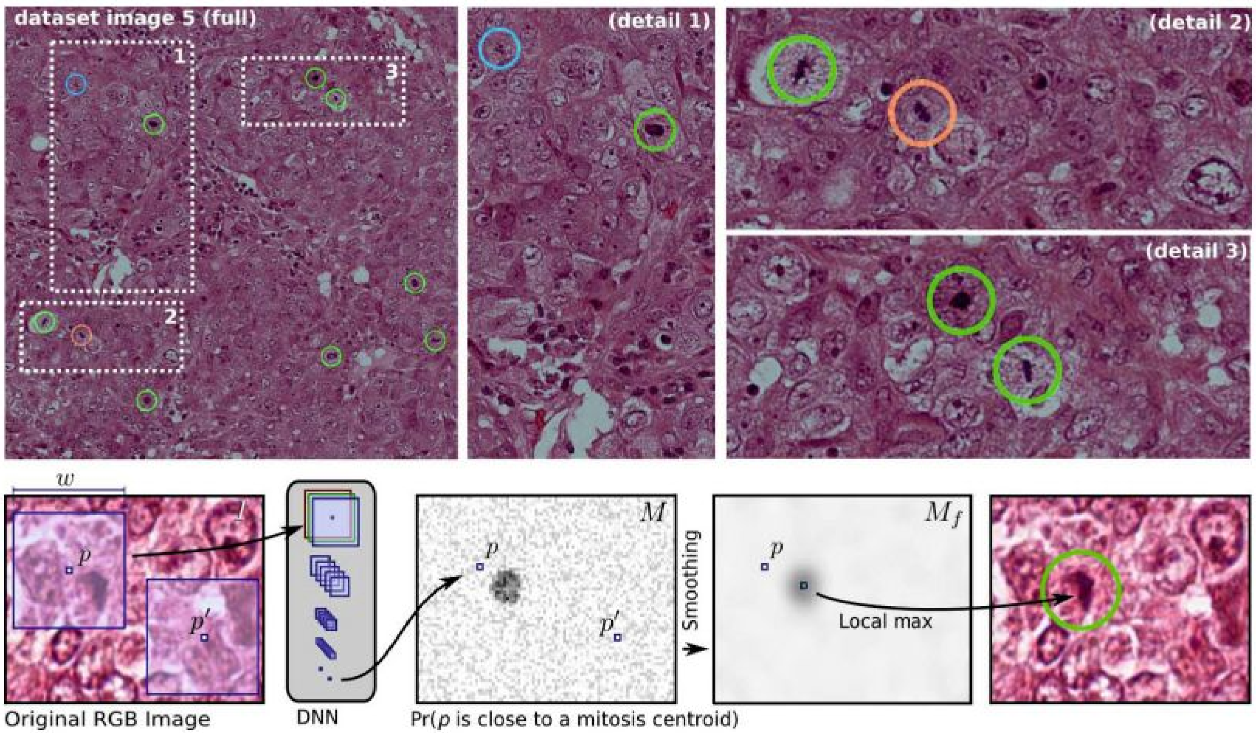
End-to-end system to automatically detect, segments, and locates cells and nuclei in microscopic images.

Segmentation with CNN (LeNet-like):

- pixel labeling with large context using a CNN
- CNN takes a window of pixels and produces a label for the central pixel
- cleanup using a kind of conditional random field (CRF)

Ning, Feng, et al. "Toward automatic phenotyping of developing embryos from videos." IEEE Transactions on Image Processing 14.9 (2005): 1360-1371.

Mitosis Detection in Breast Cancer Histology, 2013



12-layer CNN trained on samples from 50 2084 × 2084 RGB images manually annotated by experts

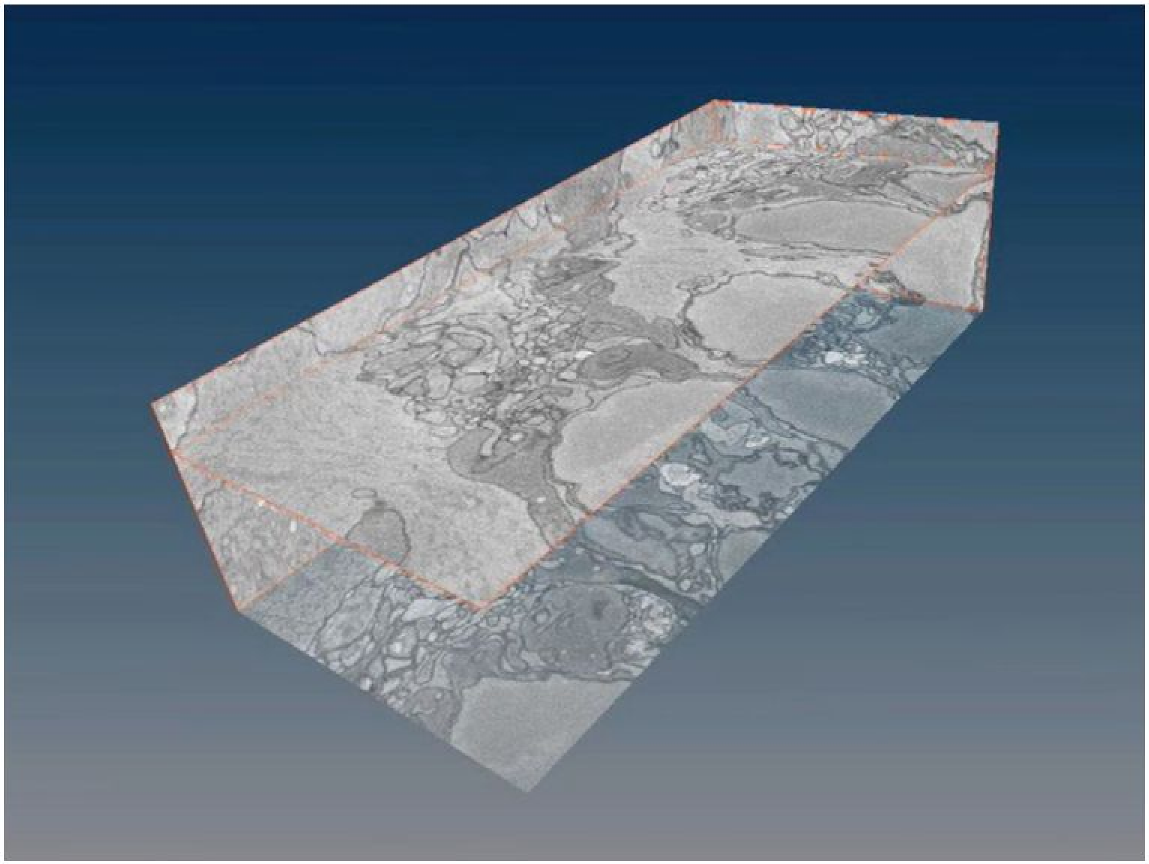
66000 mitosis pixels and 151 million non-mitosis pixels

MATLAB code with 1 day of training on GPU

outperformed other 12 teams

Cireşan, Dan C., et al. "Mitosis detection in breast cancer histology images with deep neural networks." International Conference on Medical Image Computing and Computer-assisted Intervention. Springer Berlin Heidelberg, 2013.

Segmentation in Connectomics



Helmstaedter, Moritz, et al. "Connectomic reconstruction of the inner plexiform layer in the mouse retina." Nature 500.7461 (2013): 168-174.

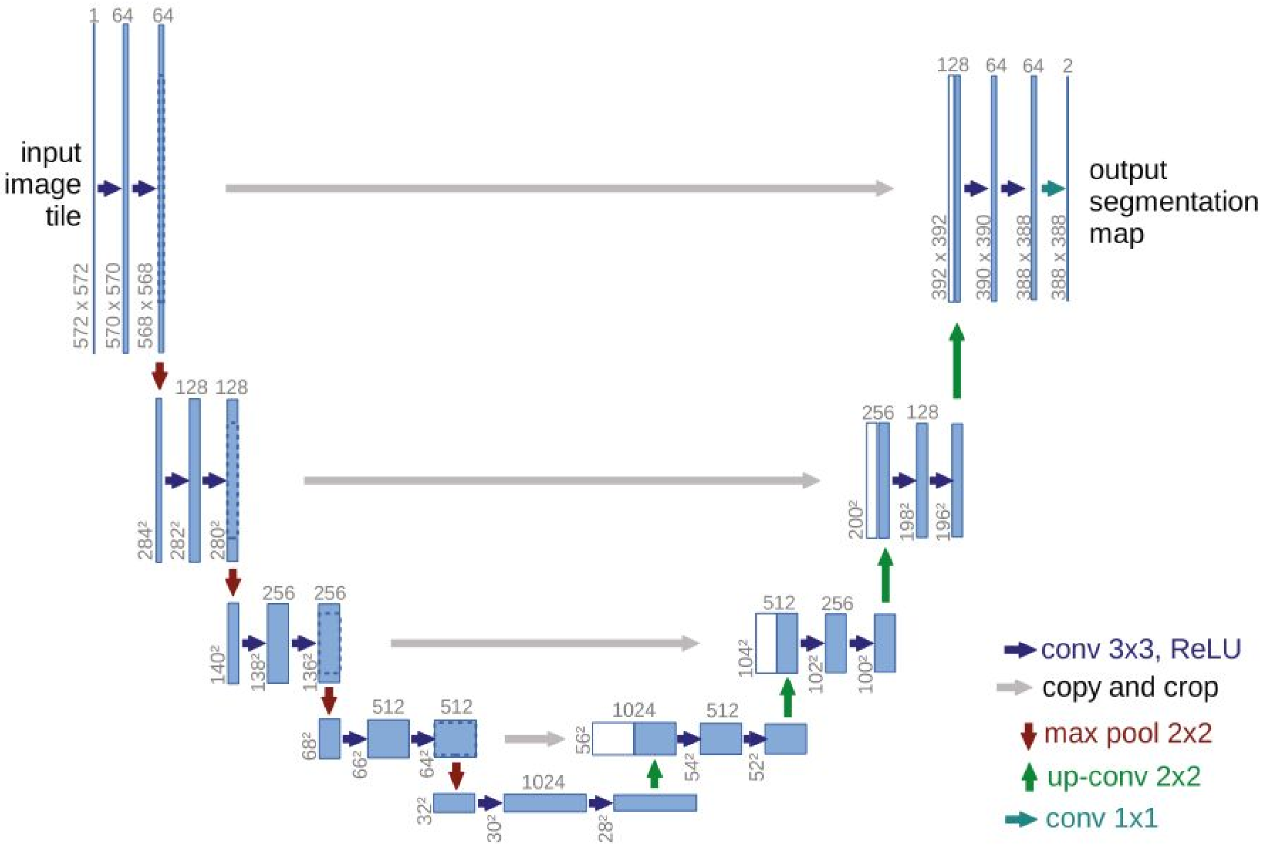
6-layer 3D CNN was trained on manually annotated 100×100×100 voxels sub-volumes

all filters were 7×7×7 voxels in size and used a logistic sigmoid nonlinearity

first application of 3D CNN in bioimage analysis

~ 300 citations

U-Net: CNN for Biomedical Segmentation



Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." International Conference on Medical Image Computing and Computer-Assisted Intervention. Springer International Publishing, 2015.

Mirrored CNN with contraction and expansion parts and shortcut connections

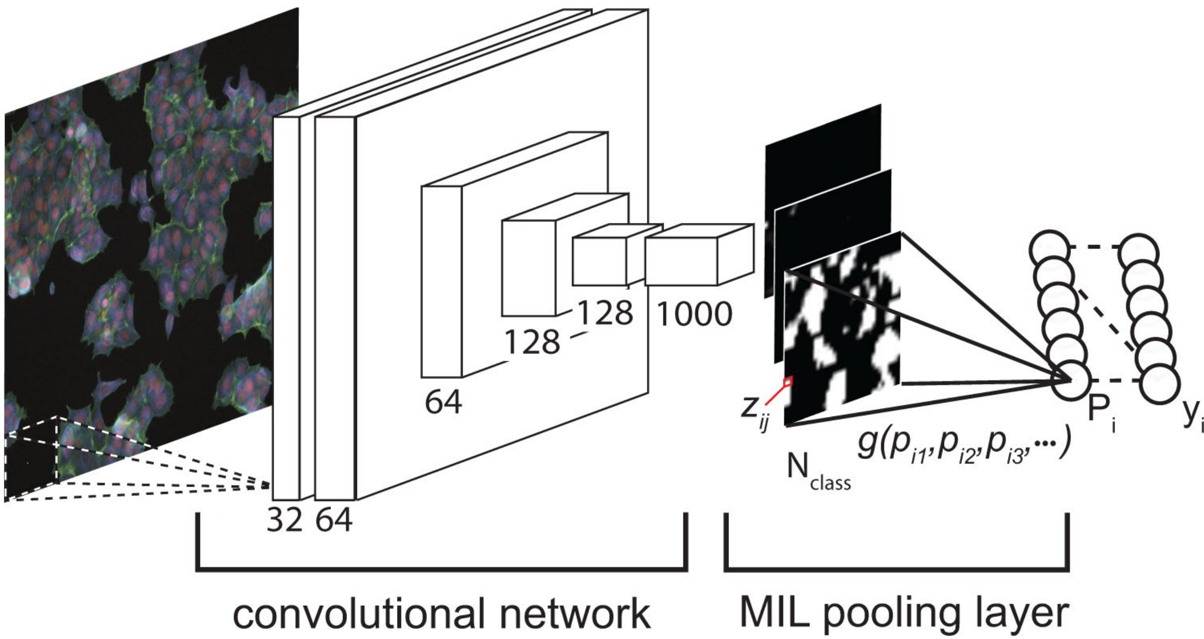
outperformed other 9 teams in segmentation of neurons in EM stacks

used to segment cells, nuclei, organs, blood vessels

> 150 citations since Oct 2015

@alxndrkalinin

Classifying and segmenting microscopy images with deep multiple instance learning

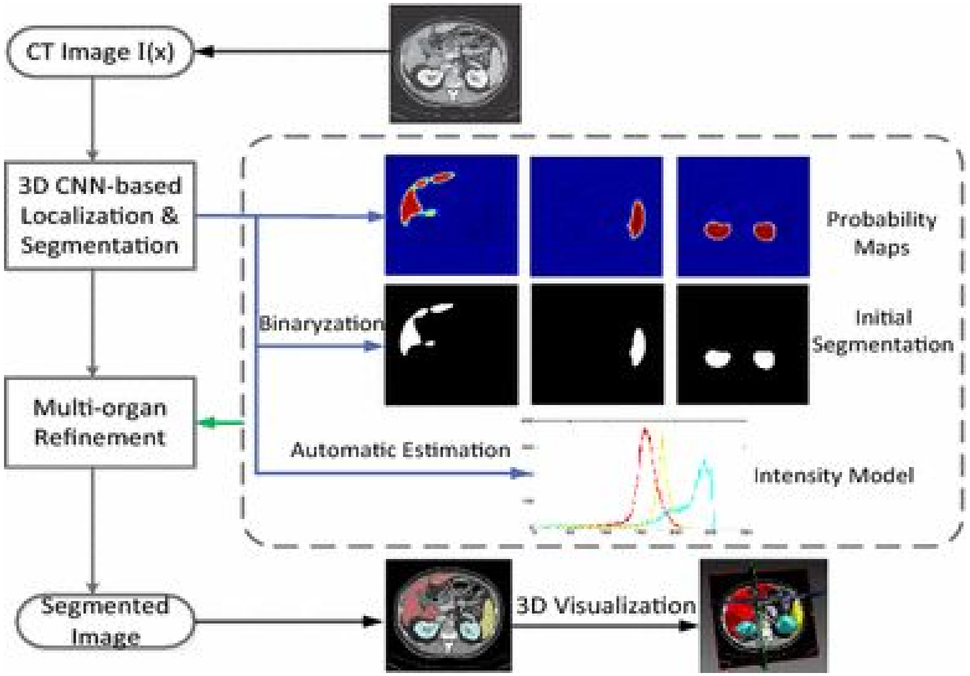


combined 12 layer CNN with new Multiple Instance Learning layer trained on 350 1000 × 1200 images for the breast cancer dataset and 2500 1000 × 1300 images for the yeast dataset with image level labels

outperforms ensemble of 60 SVMs

Kraus, Oren Z., Jimmy Lei Ba, and Brendan J. Frey. "Classifying and segmenting microscopy images with deep multiple instance learning." Bioinformatics 32.12 (2016): i52-i59.

3D Multi-organ Segmentation



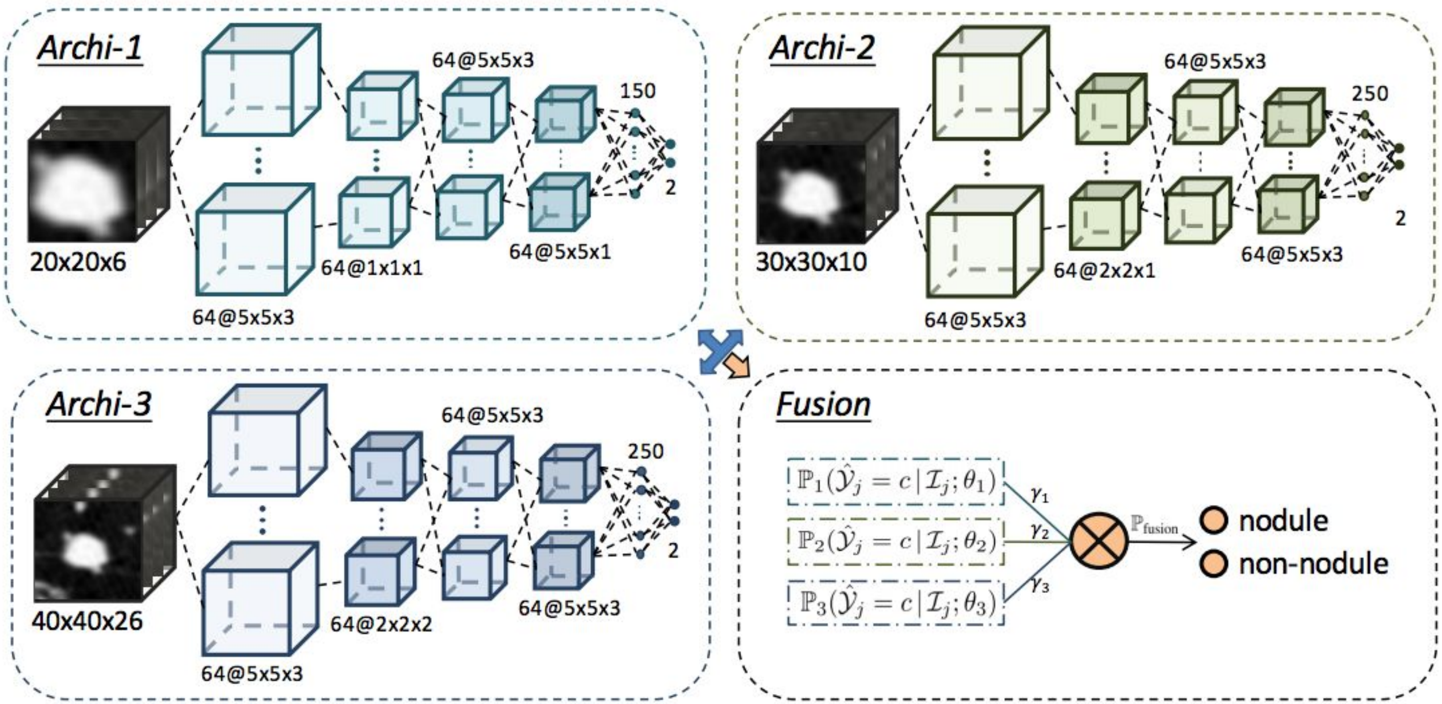
10-layer 3D CNN trained on 140 abdominal CT scans

followed by energy-based refinement model

SOTA on 4 organ segmentation: liver, spleen and both kidneys 96.0, 94.2 and 95.4% respectively

Hu, Peijun, et al. "Automatic abdominal multi-organ segmentation using deep convolutional neural network and time-implicit level sets." International Journal of Computer Assisted Radiology and Surgery (2016): 1-13.

Multi-level Contextual 3D CNNs for False Positive Reduction in Pulmonary Nodule Detection

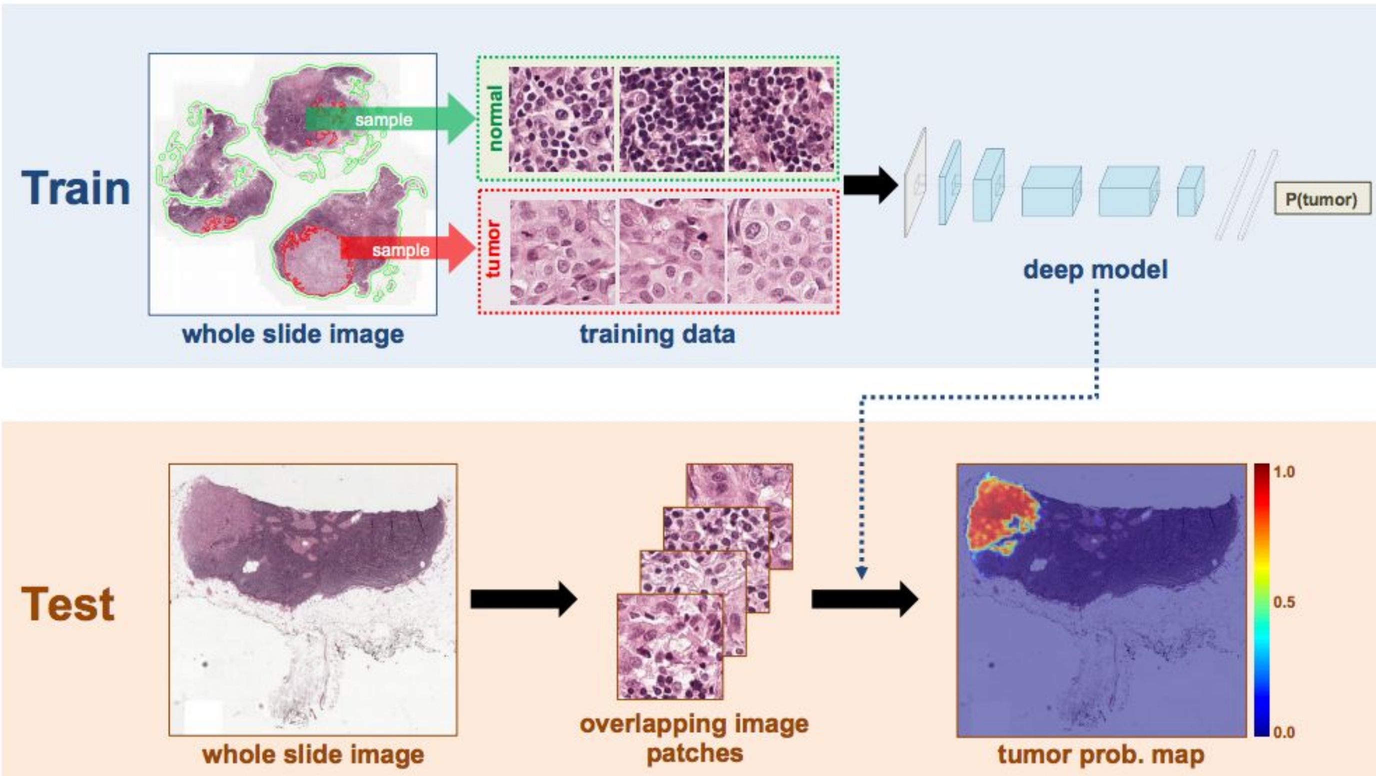


Ensemble of 3D CNNs trained on 3D patches of 888 CT scans

outperformed other 6 teams

Dou, Qi, et al. "Multi-level contextual 3D CNNs for false positive reduction in pulmonary nodule detection." IEEE Transactions on Biomedical Engineering (2016).

Identifying Metastatic Breast Cancer



27-layer CNN trained on patches extracted from WSI

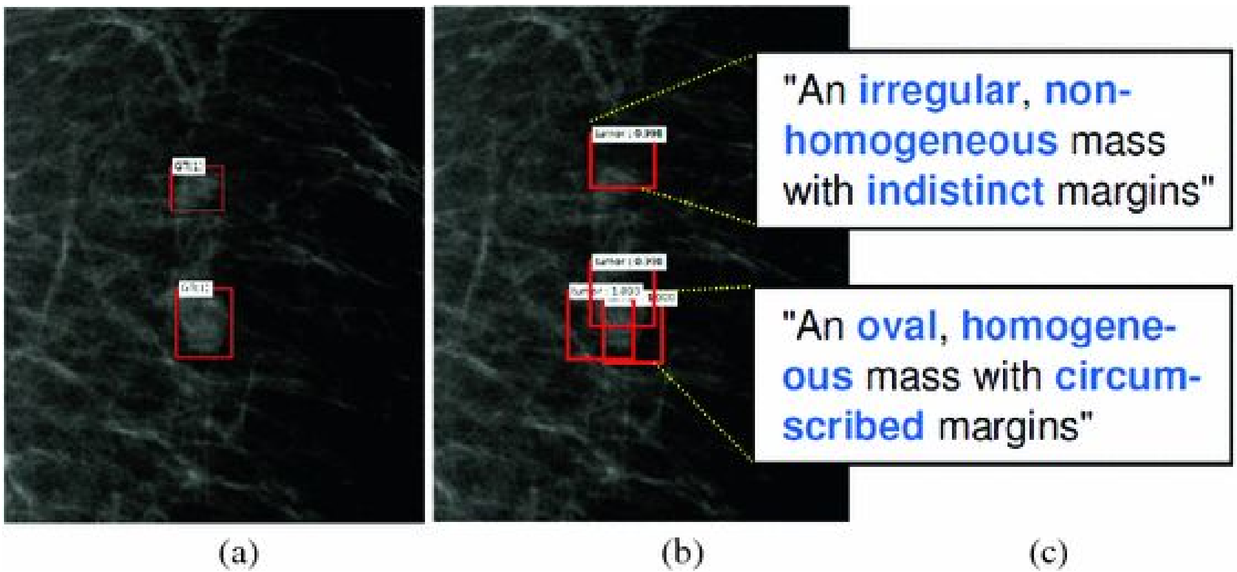
AUC 0.925
Pathologist 0.966
Combined 0.995

~85% reduction in human error rate

outperformed other 5 teams

Wang, Dayong, et al. "Deep learning for identifying metastatic breast cancer." arXiv preprint arXiv:1606.05718 (2016).

Medical Image Description Using Multi-task-loss CNN



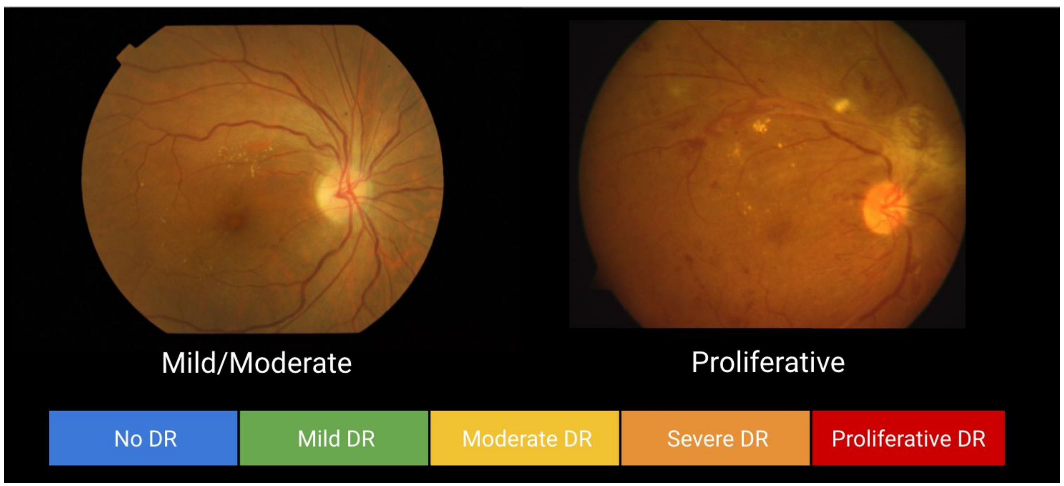
7-layer CNN with multi-task loss function

generates ROIs and then generates semantic description of lesions inside

similar to automated image captioning

Kisilev, Pavel, et al. "Medical Image Description Using Multi-task-loss CNN." International Workshop on Large-Scale Annotation of Biomedical Data and Expert Label Synthesis. Springer International Publishing, 2016.

Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs



Google Brain paper:

48-layer CNN **pretrained on natural images** (ImageNet) and fine-tuned on 128 175 retinal images

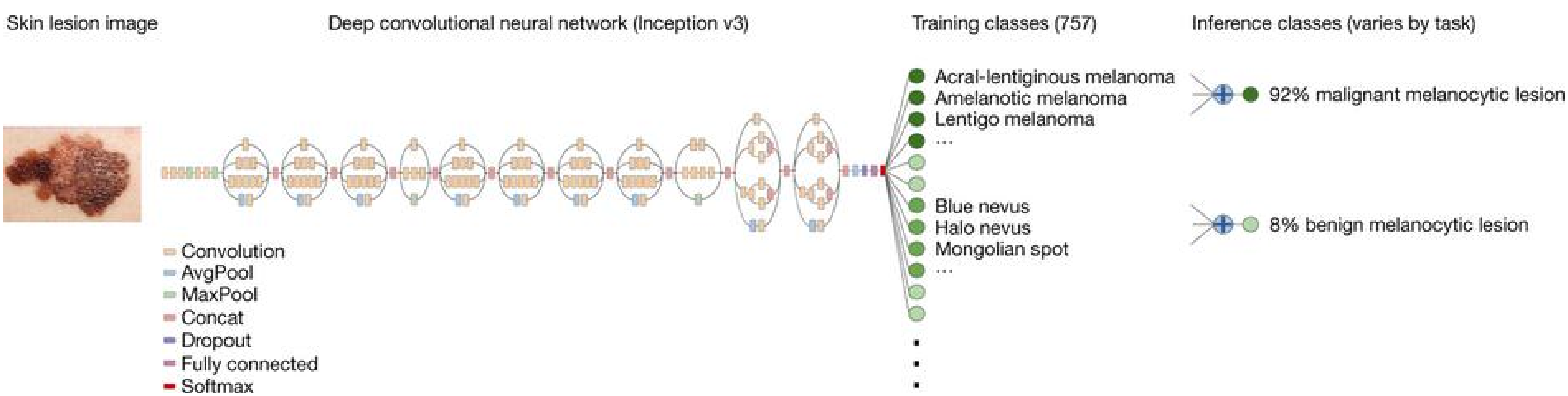
detecting referable diabetic retinopathy AUC 0.99

continue to improve for early detection

Gulshan, Varun, et al. "Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs." JAMA 316.22 (2016): 2402-2410.

Dermatologist-level classification of skin cancer with deep neural networks

48-layer CNN pretrained on natural images (ImageNet) and fine-tuned on 129,450 medical images. Performance tested against 21 board-certified dermatologists on biopsy-proven clinical images of 2 types of skin cancer.



Esteva, Andre, et al. "Dermatologist-level classification of skin cancer with deep neural networks." Nature 542.7639 (2017): 115-118.

@alxndrkalinin

Kaggle competitions won by CNNs

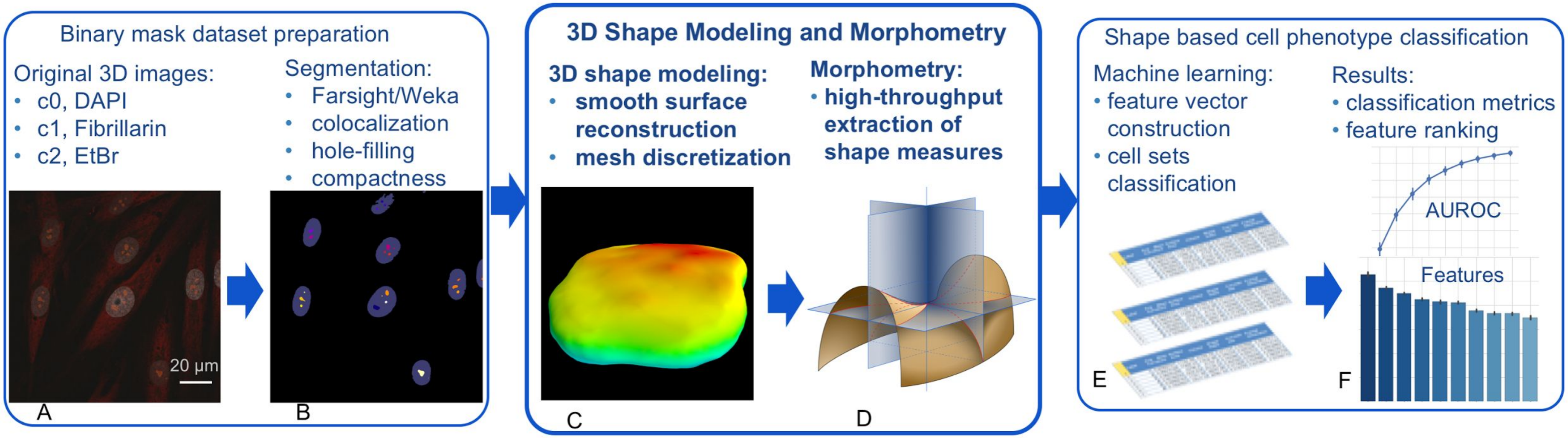
- CIFAR-10
- Diabetic Retinopaty Detection
- National Data Science Bowls 1, 2, 3
- Distracted Driver Detection
- Satellite Imagery Feature Detection
- etc.

3. CNNs in my research

Sparse 3D CNN for nuclear shape classification

(unpublished)

Traditional workflow

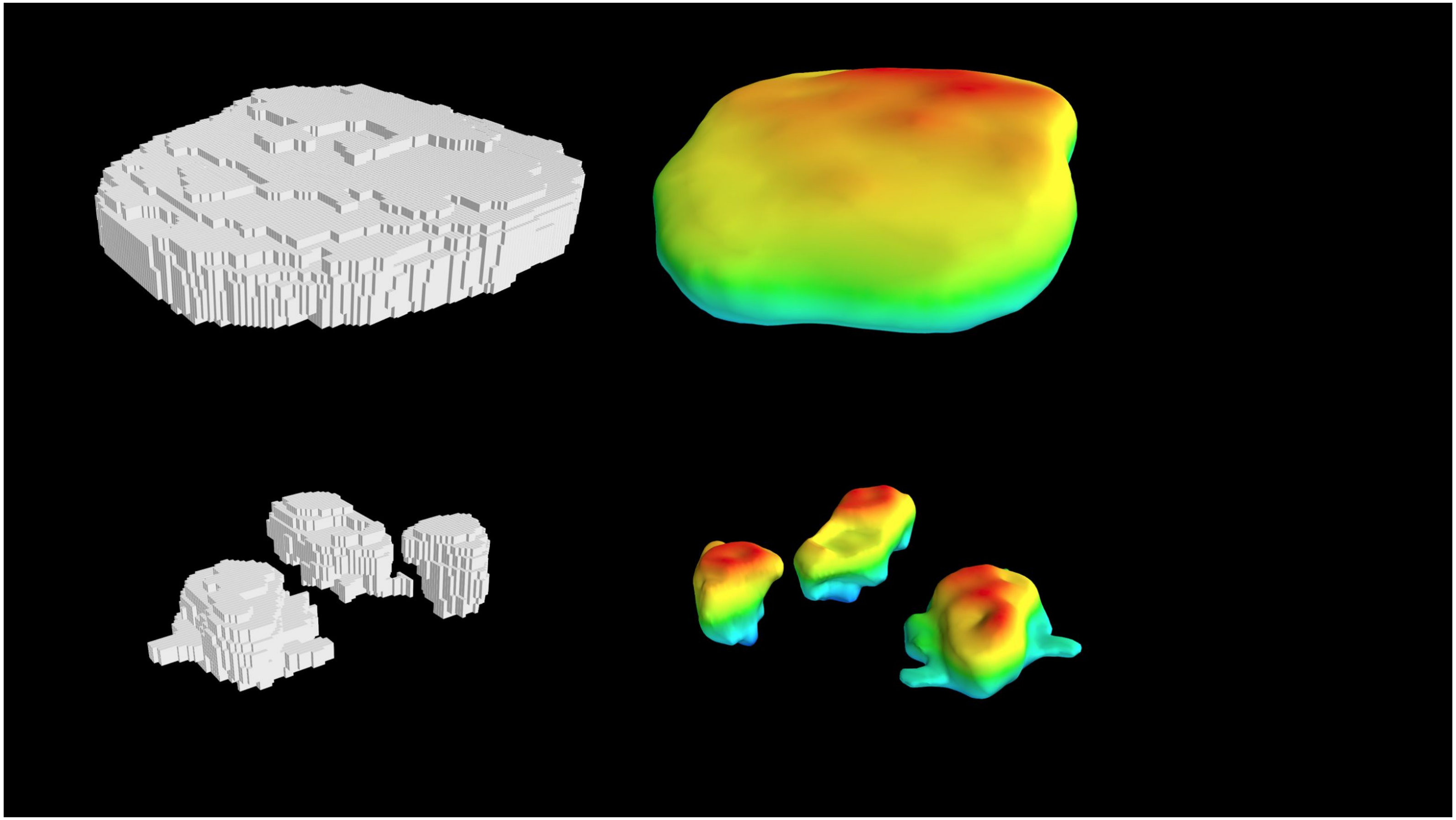


- pre-processing, curation, and segmentation
- 3D shape modeling
- manually defined 6 morphometric measures extraction
- shape classification

Kalinin et al. High-Throughput Pipeline Workflow for 3D Cell Nuclear Morphological Modeling and Classification. Submitted to BMC Bioinformatics.

Traditional workflow

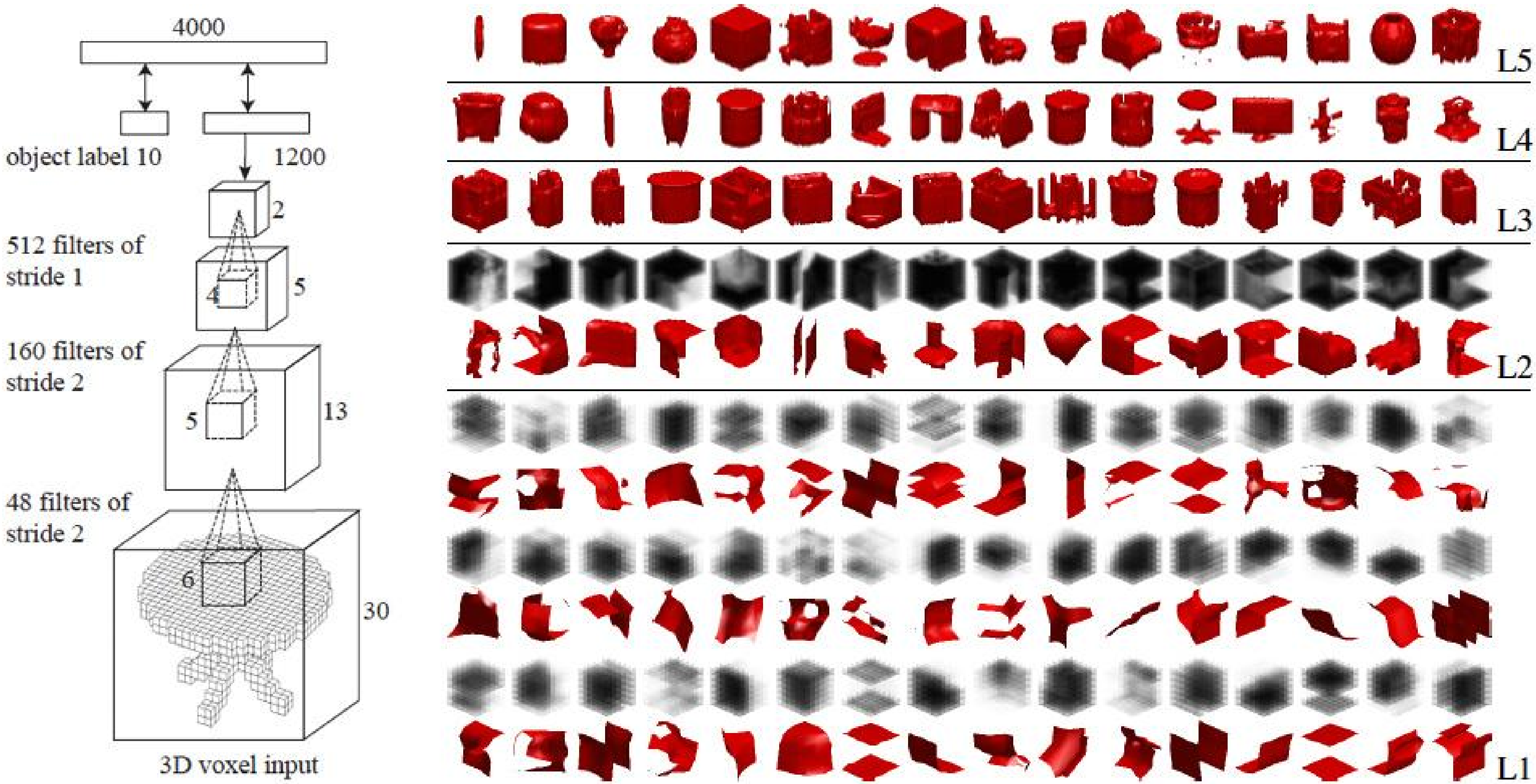
AUC ~75%. Topology-limited to genus zero 2-manifolds. Can we do better?



Kalinin et al. High-Throughput Pipeline Workflow for 3D Cell Nuclear Morphological Modeling and Classification. Submitted to BMC Bioinformatics.

3D CNNs for shape recognition

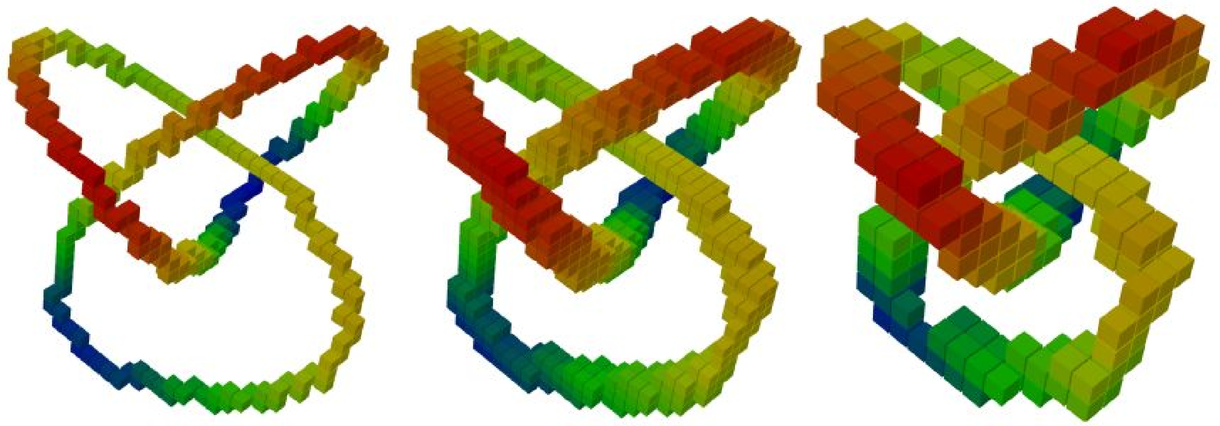
3D ShapeNets, CVPR 2015



Kalinin et al. High-Throughput Pipeline Workflow for 3D Cell Nuclear Morphological Modeling and Classification. Submitted to BMC Bioinformatics.

@alxndrkalinin

Sparse CNN

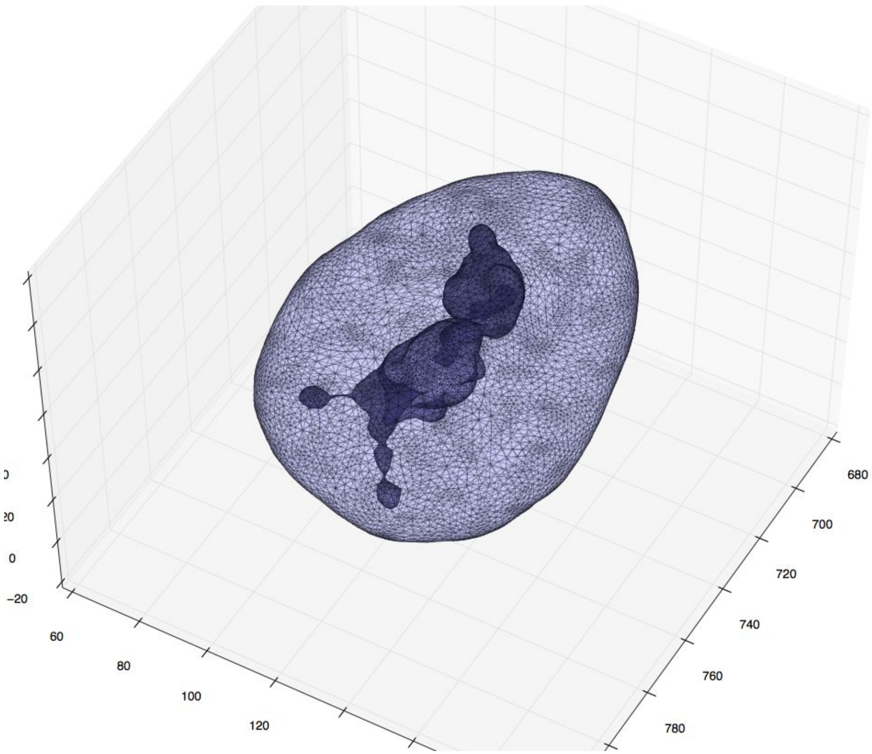


Speeds up volumetric calculations by convolving only with non-zero elements. Topology-agnostic. CUDA/C++ implementation on GitHub with 3D support.

1st place in :
- Kaggle CIFAR-10 competition, 2014
- Kaggle Diabetic Retinopathy Detection competition, 2014.

Graham, Ben. "Sparse 3D convolutional neural networks." arXiv preprint arXiv:1505.02890 (2015).

Trying it on our data



Using 3D Sparse ConvNet:

- 12 layers
- ~ 10 hours to train
- heavy augmentation by 3D rotation
- dropout augmentation

Preliminary results:

Morphometric measures:
75% AUC

3D Sparse CNN: **85% AUC**

Still lots of room for improvement.

Literature

Review of biomedical applications:

Ching et al. Opportunities And Obstacles For Deep Learning In Biology And Medicine. bioRxiv, May 28, 2017. https://doi.org/10.1101/142760

Deep Learning:

1. Stanford's Unsupervised Feature Learning and Deep Learning Tutorial
2. Michael A. Nielsen, "Neural Networks and Deep Learning", Determination Press, 2015.
3. Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep learning. MIT Press, 2016.
4. Stanford's CS231n Convolutional Neural Networks for Visual Recognition
5. Convolutional Neural Networks (CNNs): An Illustrated Explanation. XRDS.
6. How convolutional neural networks see the world. Keras Blog.

Questions?

Thanks for your attention!

Join Ann Arbor Data Science Slack group:

https://a2mads.herokuapp.com/