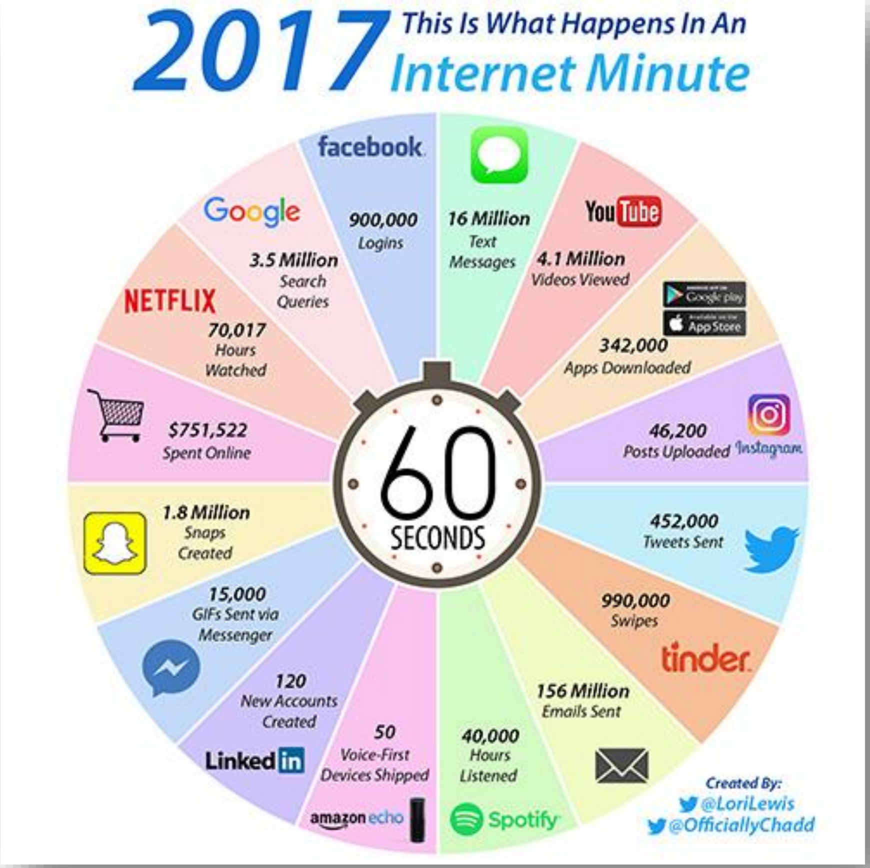
Fundamentals of Deep (Artificial) Neural Networks (DNN)

Gregory Tsagkatakis

CSD – UOC & ICS – FORTH

http://users.ics.forth.gr/~greg/

The Big Data era

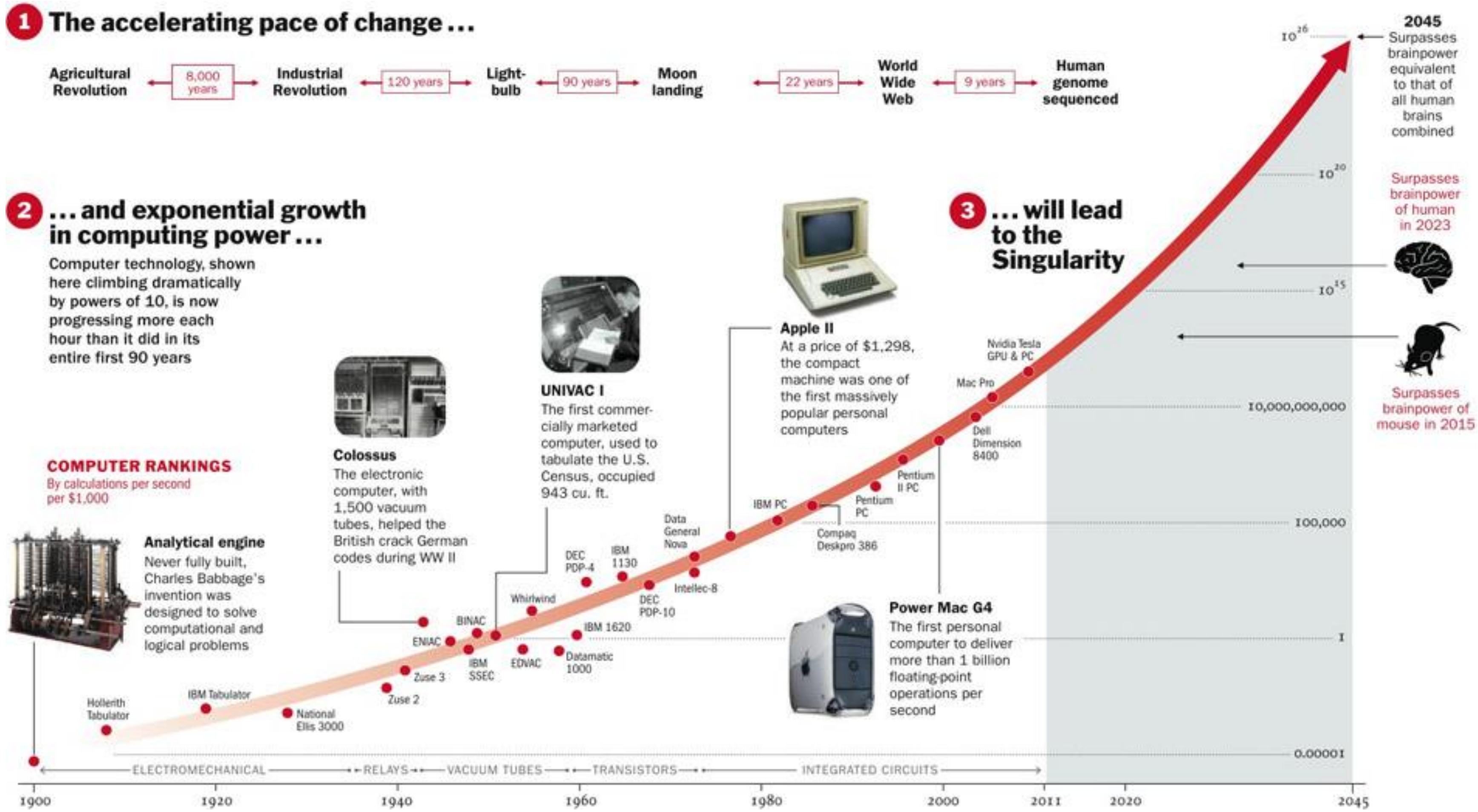


The Big Data era in Astronomy

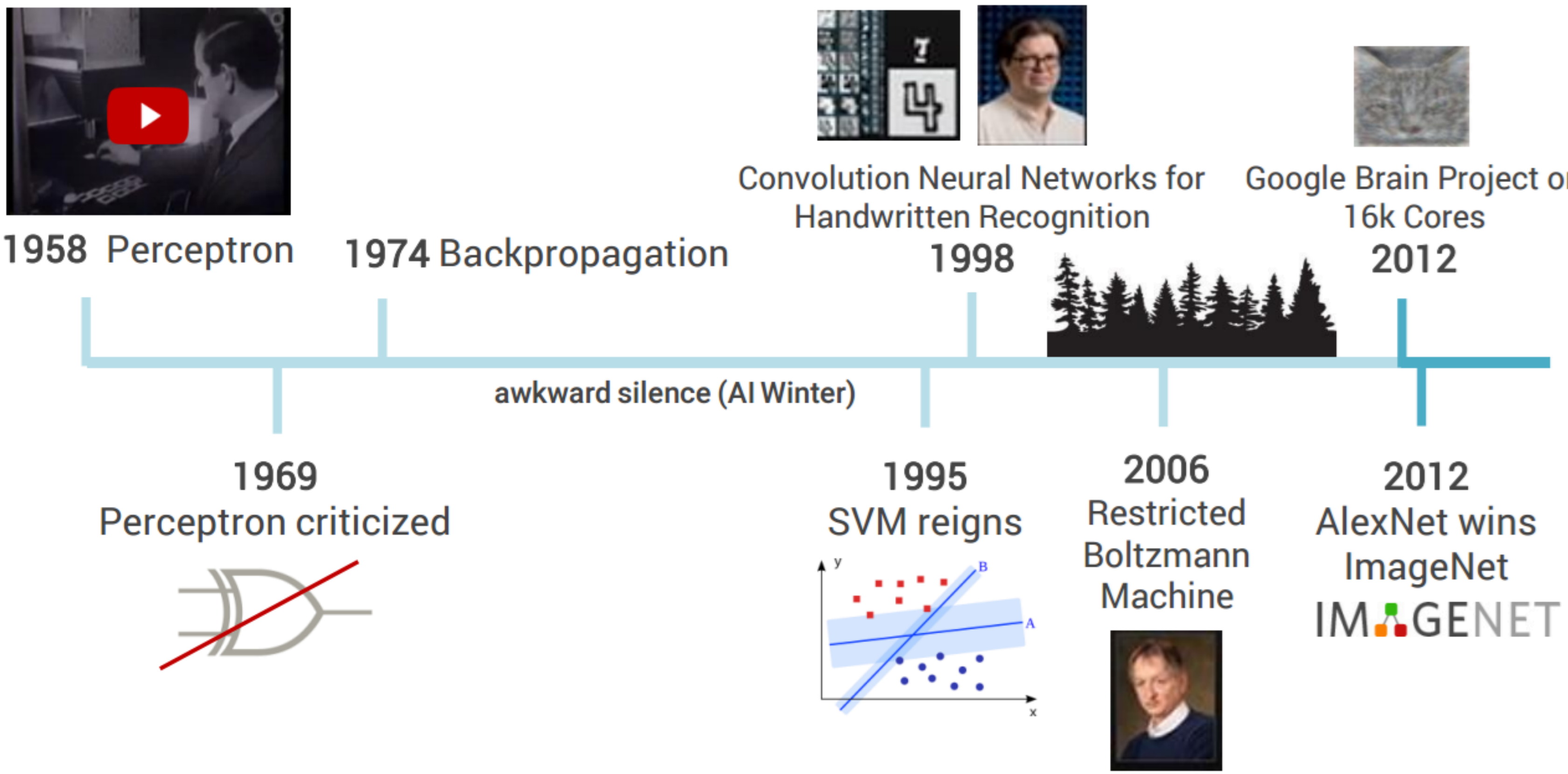
Sky Survey Project	Volume	Velocity	Variety
Sloan Digital Sky Survey (SDSS)	50 TB	200 GB per day	Images, redshifts
Large Synoptic Survey Telescope (LSST)	~ 200 PB	30 TB per day	Images, catalogs
Square Kilometer Array (SKA)	~ 4.6 EB	150 TB per day	Images, redshifts

Garofalo, Mauro, Alessio Botta, and Giorgio Ventre. "Astrophysics and Big Data: Challenges, Methods, and Tools." Proceedings of the International Astronomical Union 12.S325 (2016)

Accelerated growth

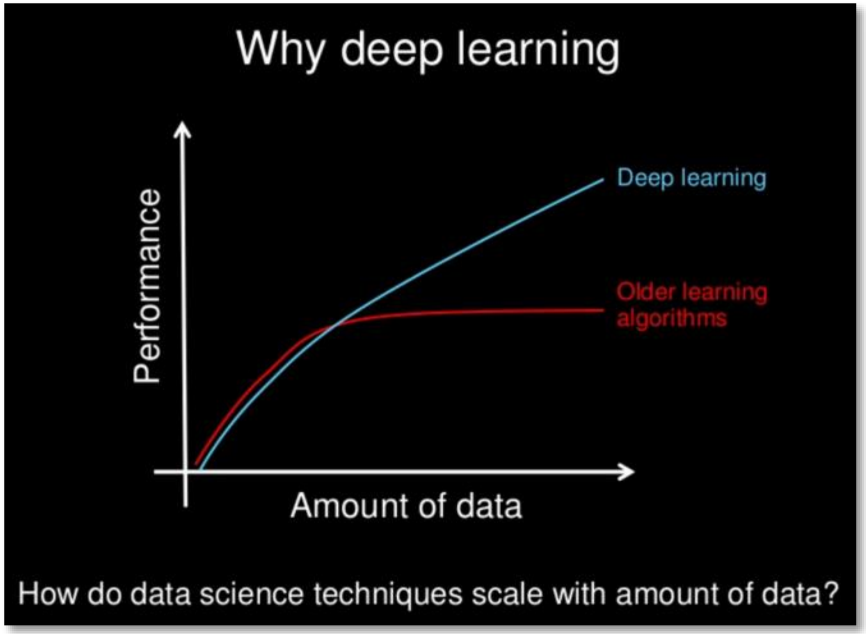


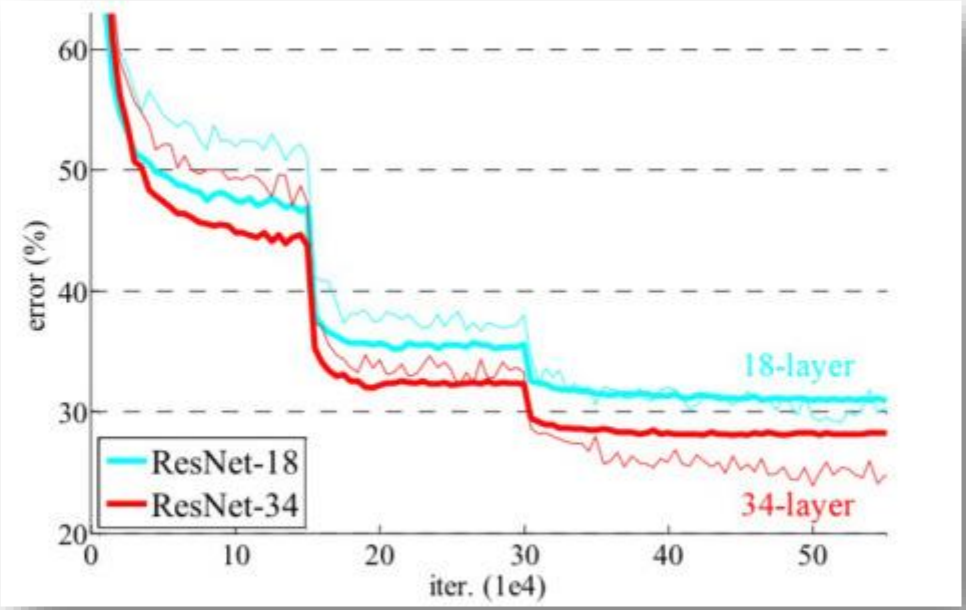
Brief history of DL



Why Today?

Lots of Data

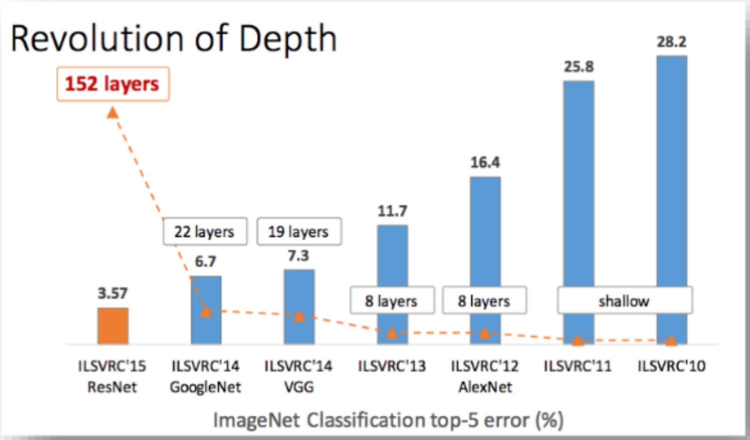


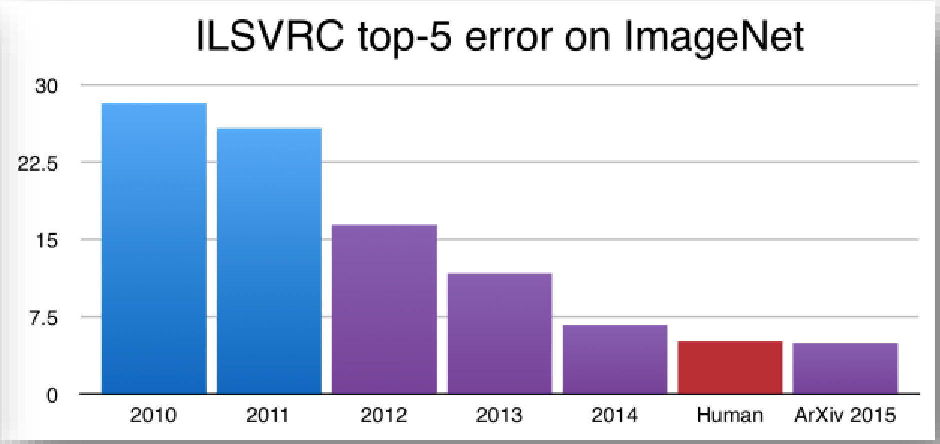


Why Today?

Lots of Data

Deeper Learning



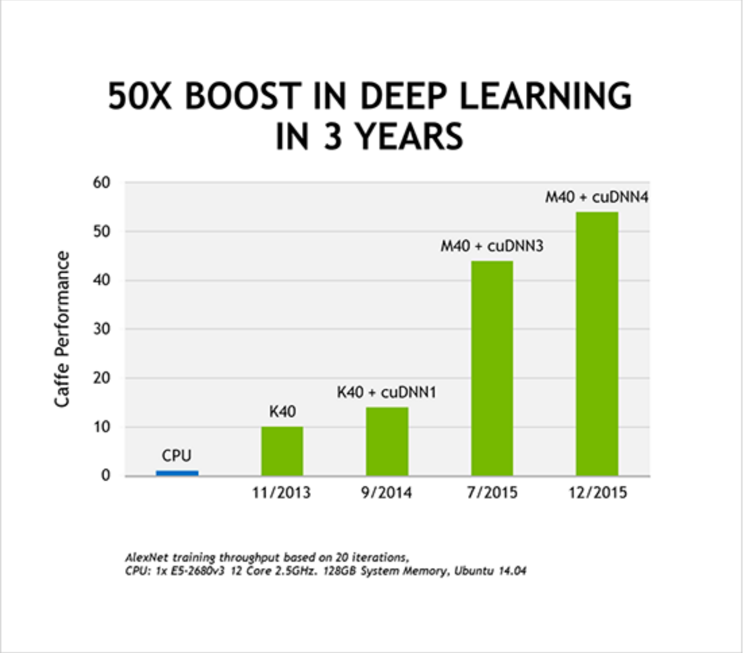


Why Today?

Lots of Data

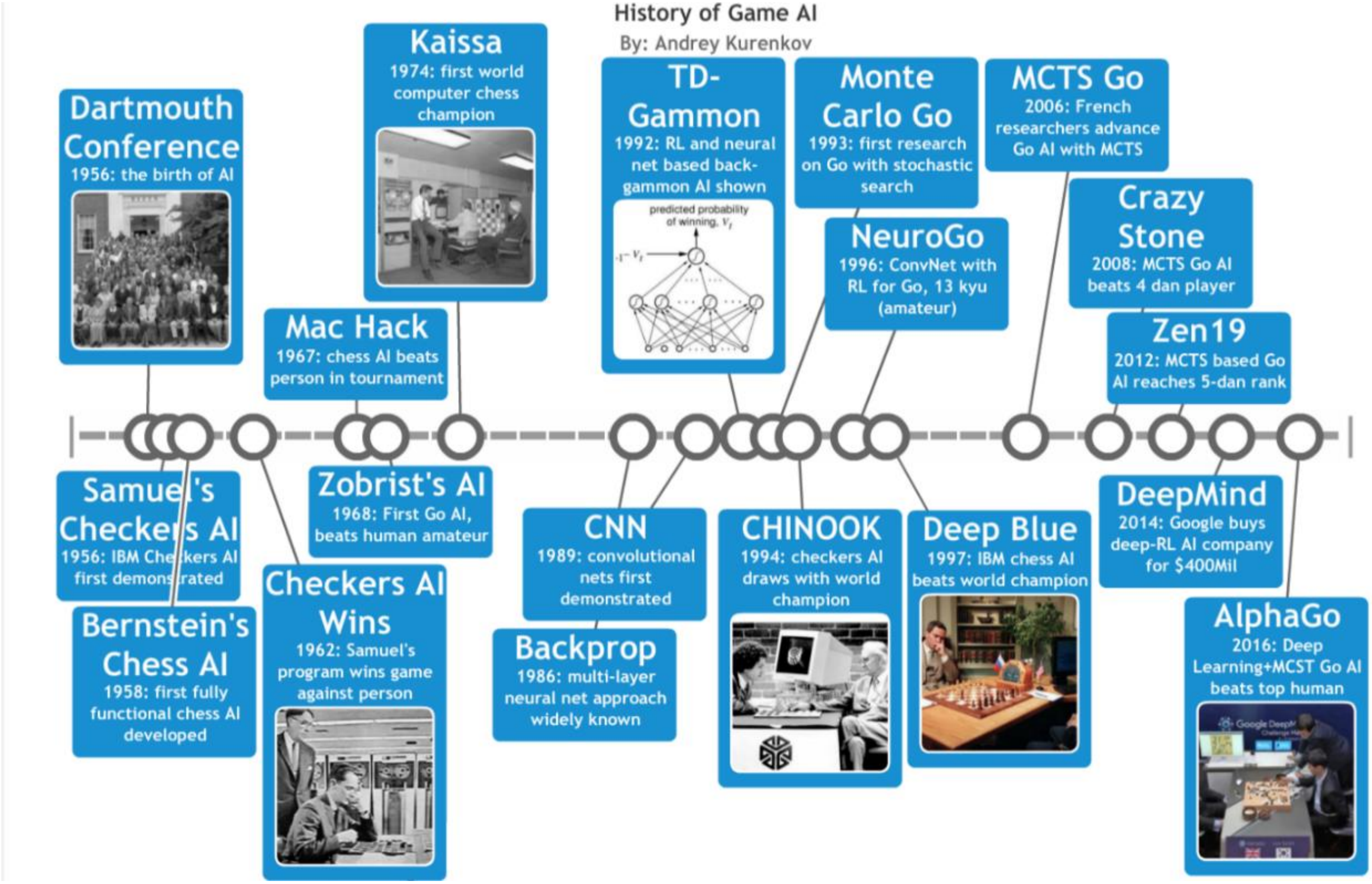
Deep Learning

More Power



https://blogs.nvidia.com/blog/2016/01/12/accelerating-ai-artificial-intelligence-gpus/
https://www.slothparadise.com/what-is-cloud-computing/

Apps: Gaming

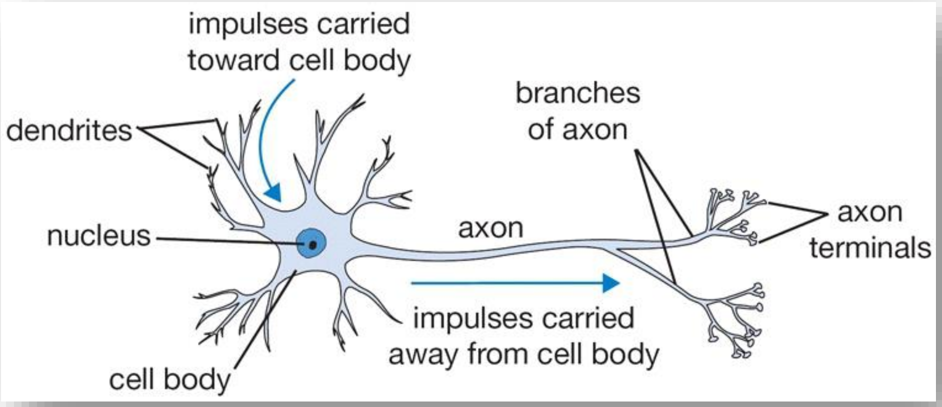

History of Game AI
By: Andrey Kurenkov
Dartmouth Conference
1956: the birth of AI
Kaissa
1974: first world computer chess champion
Mac Hack
1967: chess AI beats person in tournament
TD-Gammon
1992: RL and neural net based back-gammon AI shown
predicted probability of winning, V_t
Monte Carlo Go
1993: first research on Go with stochastic search
NeuroGo
1996: ConvNet with RL for Go, 13 kyu (amateur)
MCTS Go
2006: French researchers advance Go AI with MCTS
Crazy Stone
2008: MCTS Go AI beats 4 dan player
Zen19
2012: MCTS based Go AI reaches 5-dan rank
Samuel's Checkers AI
1956: IBM Checkers AI first demonstrated
Bernstein's Chess AI
1958: first fully functional chess AI developed
Checkers AI Wins
1962: Samuel's program wins game against person
Zobrist's AI
1968: First Go AI, beats human amateur
CNN
1989: convolutional nets first demonstrated
Backprop
1986: multi-layer neural net approach widely known
CHINOOK
1994: checkers AI draws with world champion
Deep Blue
1997: IBM chess AI beats world champion
DeepMind
2014: Google buys deep-RL AI company for $400Mil
AlphaGo
2016: Deep Learning+MCST Go AI beats top human

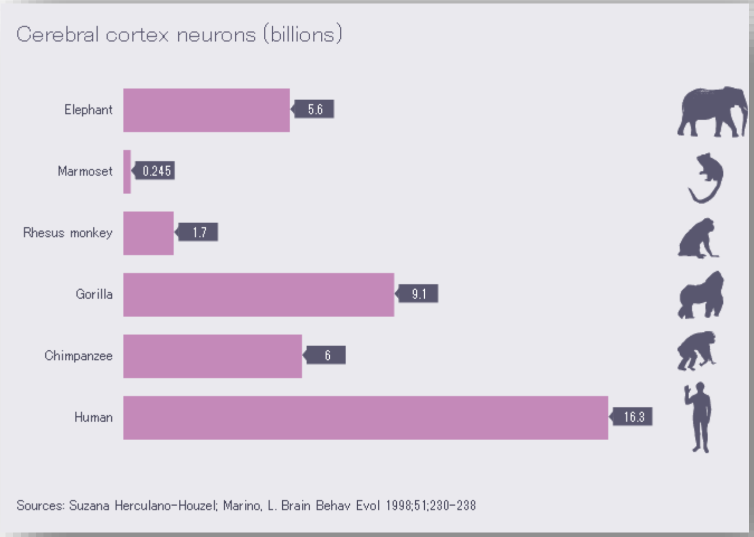
Machine learning is everywhere

Deep Learning

Inspiration from brain

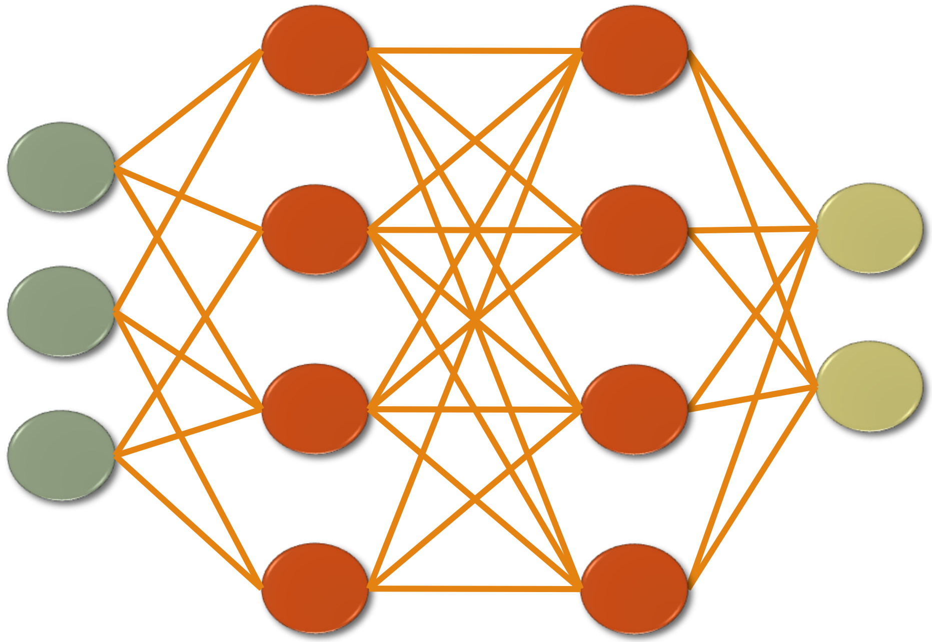
- 86Billion neurons
- 10^{14}-10^{15} synapses





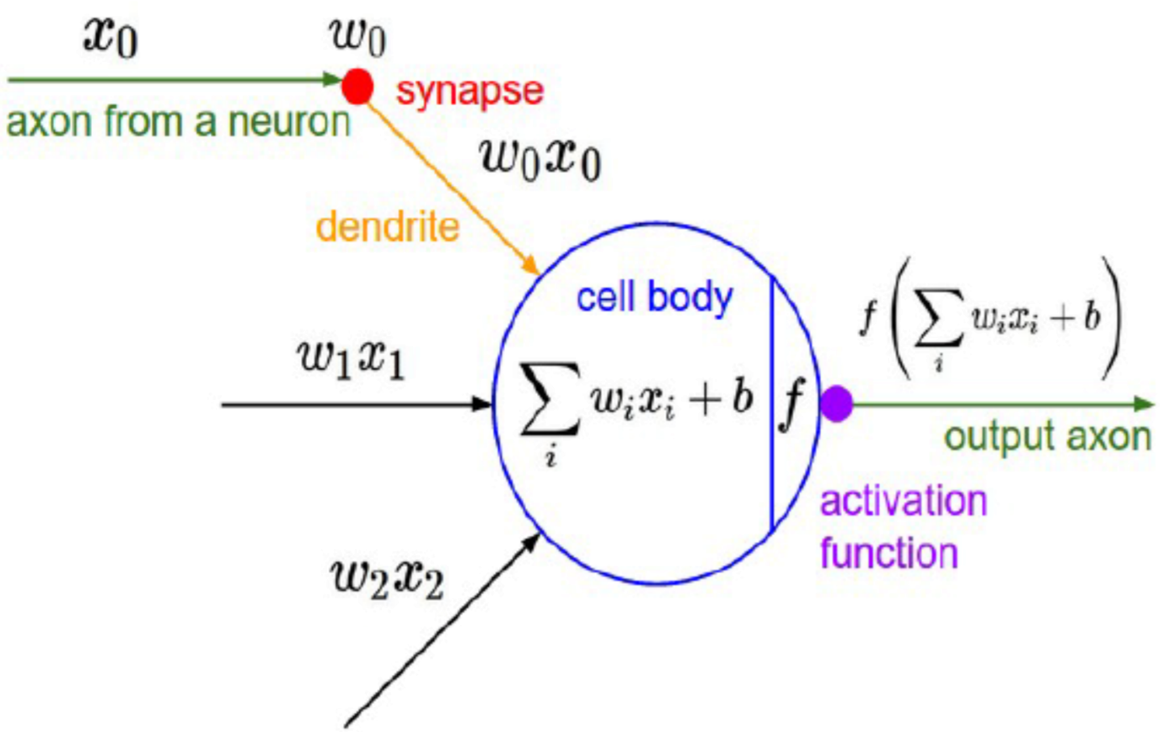
Key components of ANN

- Architecture (input/hidden/output layers)



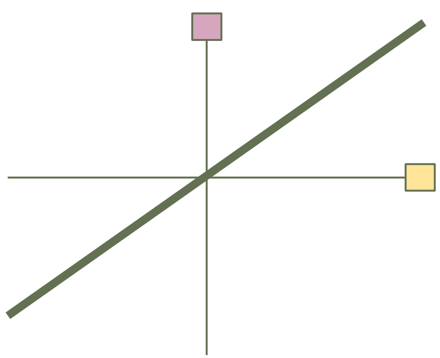
Key components of ANN

- Architecture (input/hidden/output layers)
- Weights

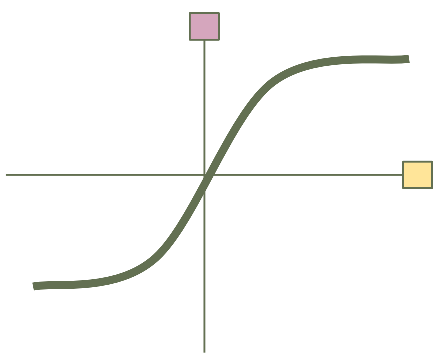


Key components of ANN

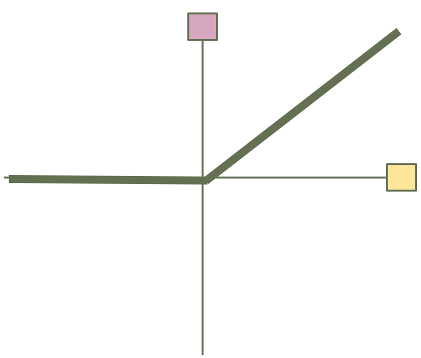
- Architecture (input/hidden/output layers)
- Weights
- Activations



LINEAR



LOGISTIC / SIGMOIDAL / TANH



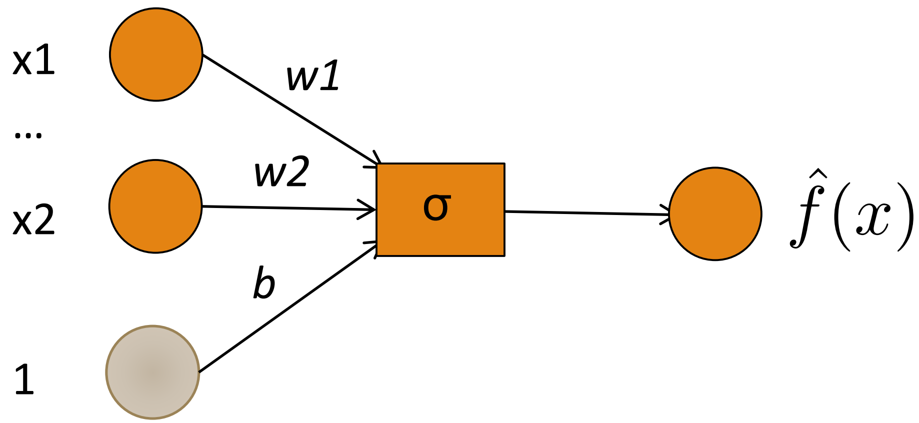
RECTIFIED LINEAR (ReLU)

Perceptron: an early attempt

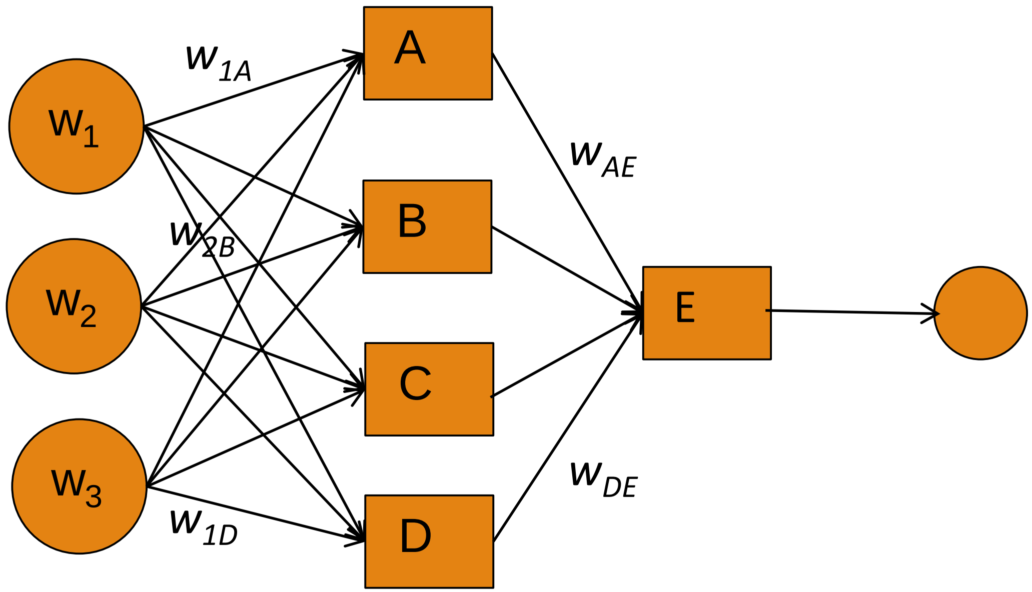
Activation function

$$\hat{f}(x) = \sigma(w \cdot x + b) \qquad \sigma(y) = \begin{cases} 1, & y > 0 \\ 0, & o/w \end{cases}$$

Need to tune w and b



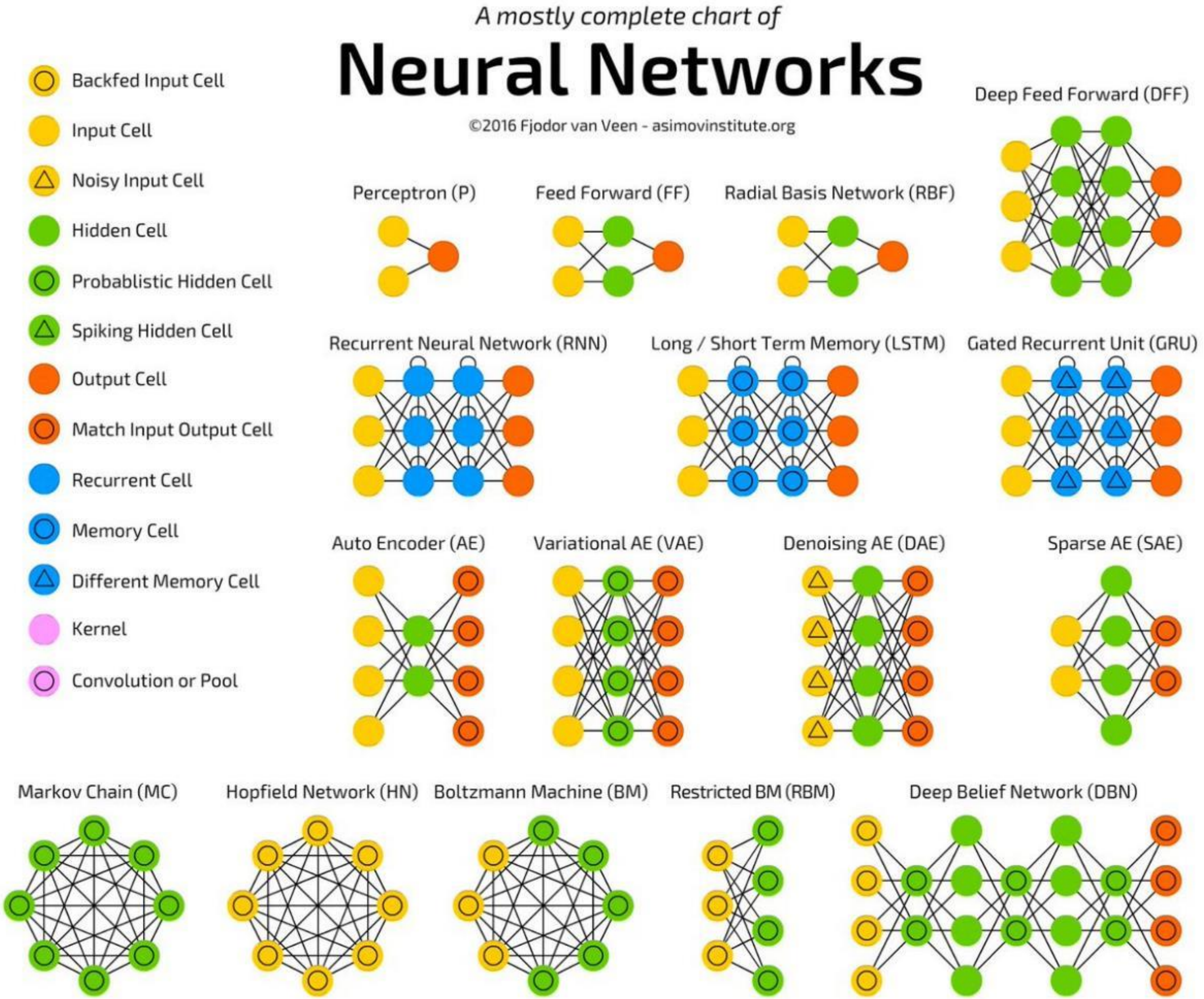
Multilayer perceptron



We just added a neuron layer!

We just introduced non-linearity!

A neuron is of the form **σ(w.x + b)** where **σ** is an *activation* function



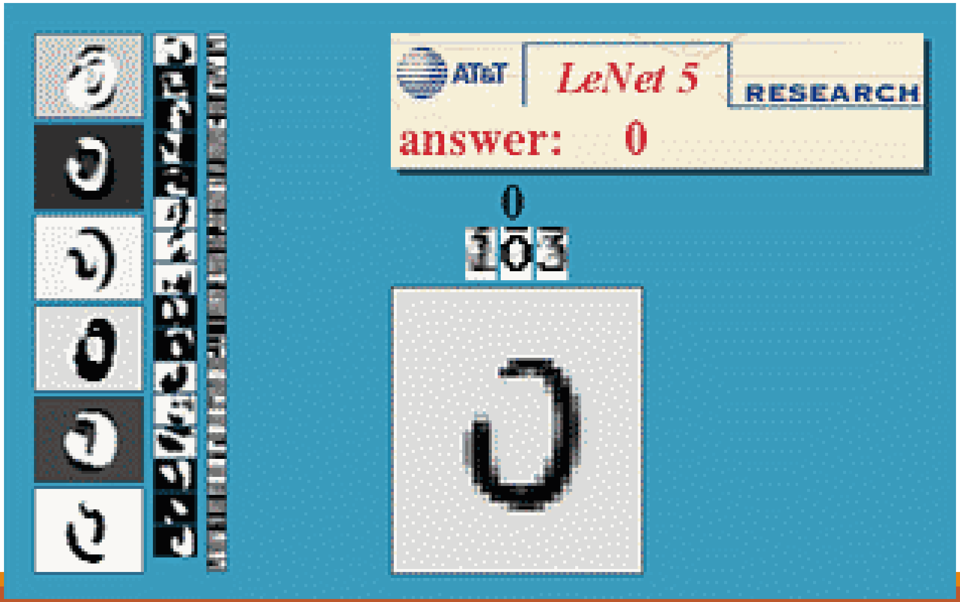
[Sasen Cain (@spectralradius)]

Training & Testing

Training: determine weights

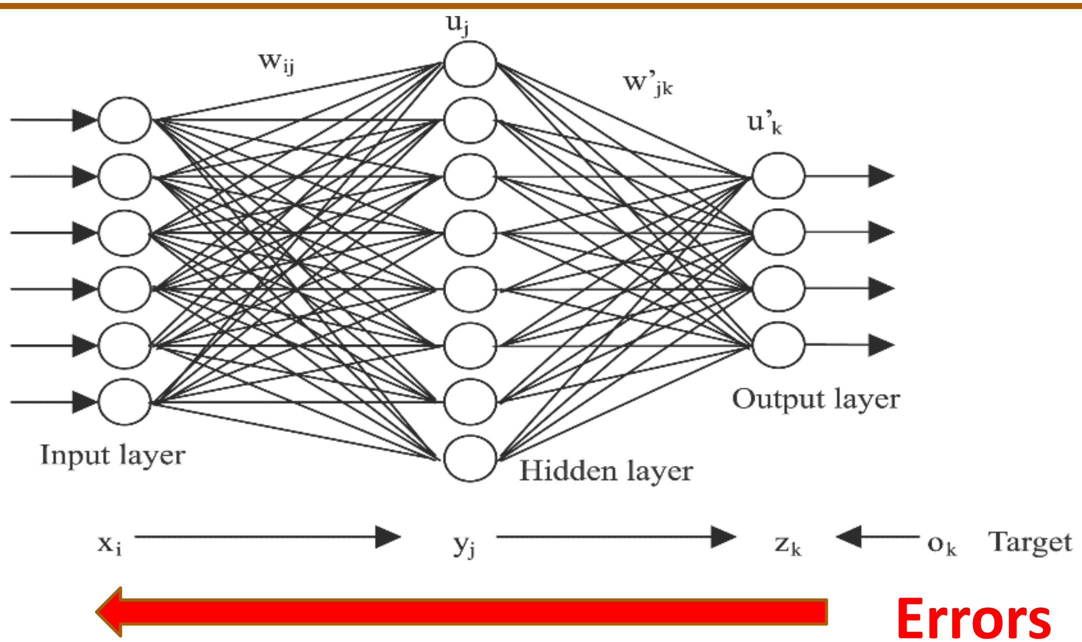
- Supervised: labeled training examples
- Unsupervised: no labels available
- Reinforcement: examples associated with rewards

Testing (Inference): apply weights to new examples



Training DNN

1. Get batch of data
2. Forward through the network -> estimate loss
3. Backpropagate error
4. Update weights based on gradient



Backpropagation

Chain Rule in Gradient Descent: Invented in 1969 by Bryson and Ho

Defining a loss/cost function $$J(x, y; \theta) = \frac{1}{2}\sum(y - f(x;\theta))^2$$

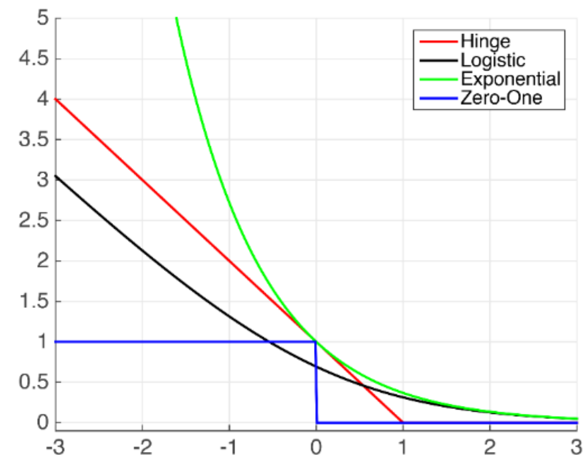
Assume a function

$$f(x;\theta) = w^T x + b \quad , \quad \theta = \{w, b\}$$

and associated predictions $\hat{y} = f(x;\theta)$

Examples of Loss function

- Hinge $J(\hat{y}, y) = max\{0, 1 - \hat{y}y\}$
- Exponential $J(\hat{y}, y) = exp(-\hat{y}y)$
- Logistic $J(\hat{y}, y) = log_2(1 + exp(-\hat{y}y))$



Gradient Descent

- Minimize function J w.r.t. parameters θ

New weights

$$\theta^* = \theta - n * \nabla J(y, x; \theta)$$

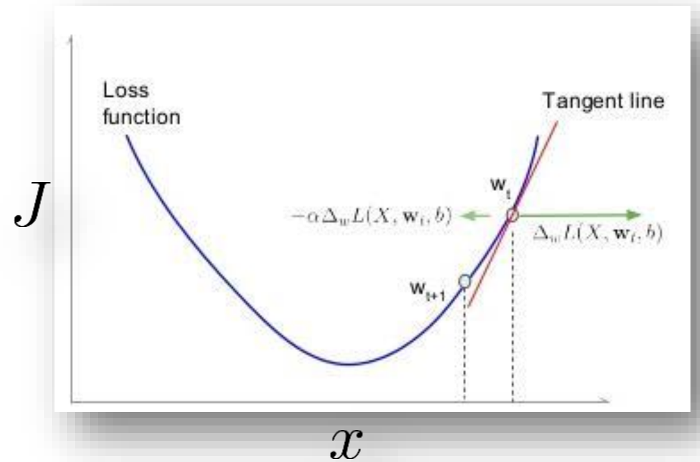
Gradient

Old weights

Learning rate

- Gradient

$$\nabla J(x) = (\frac{\partial J(x)}{\partial x_1}, \frac{\partial J(x)}{\partial x_2}, ..., \frac{\partial J(x)}{\partial x_n})$$



- Stochastic Gradient Descend

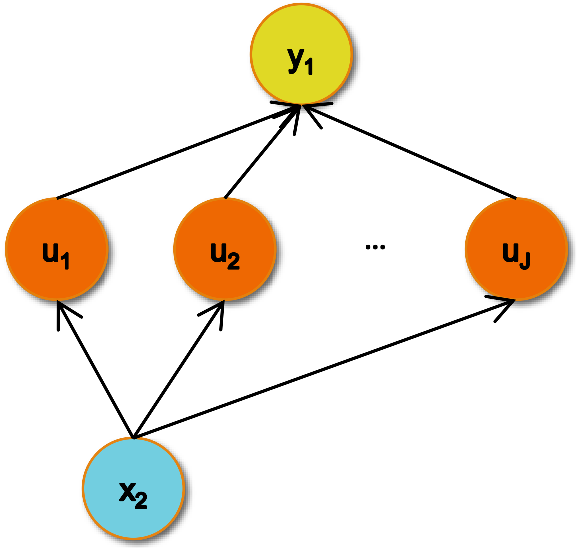
$$\theta^* := \theta - n \sum_{i=1}^{N} \nabla J(y_i, x_i; \theta)$$

Backpropagation

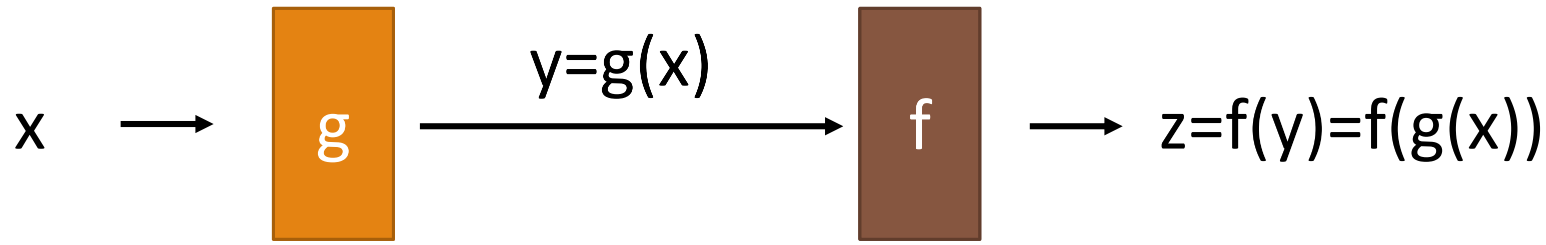
Given: $\boldsymbol{y} = g(\boldsymbol{u})$ and $\boldsymbol{u} = h(\boldsymbol{x})$.

Chain Rule:

$$\frac{dy_i}{dx_k} = \sum_{j=1}^{J} \frac{dy_i}{du_j}\frac{du_j}{dx_k}, \quad \forall i, k$$



Backpropagation



Chain rule:

- **Single variable**

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}.$$

- **Multiple variables**

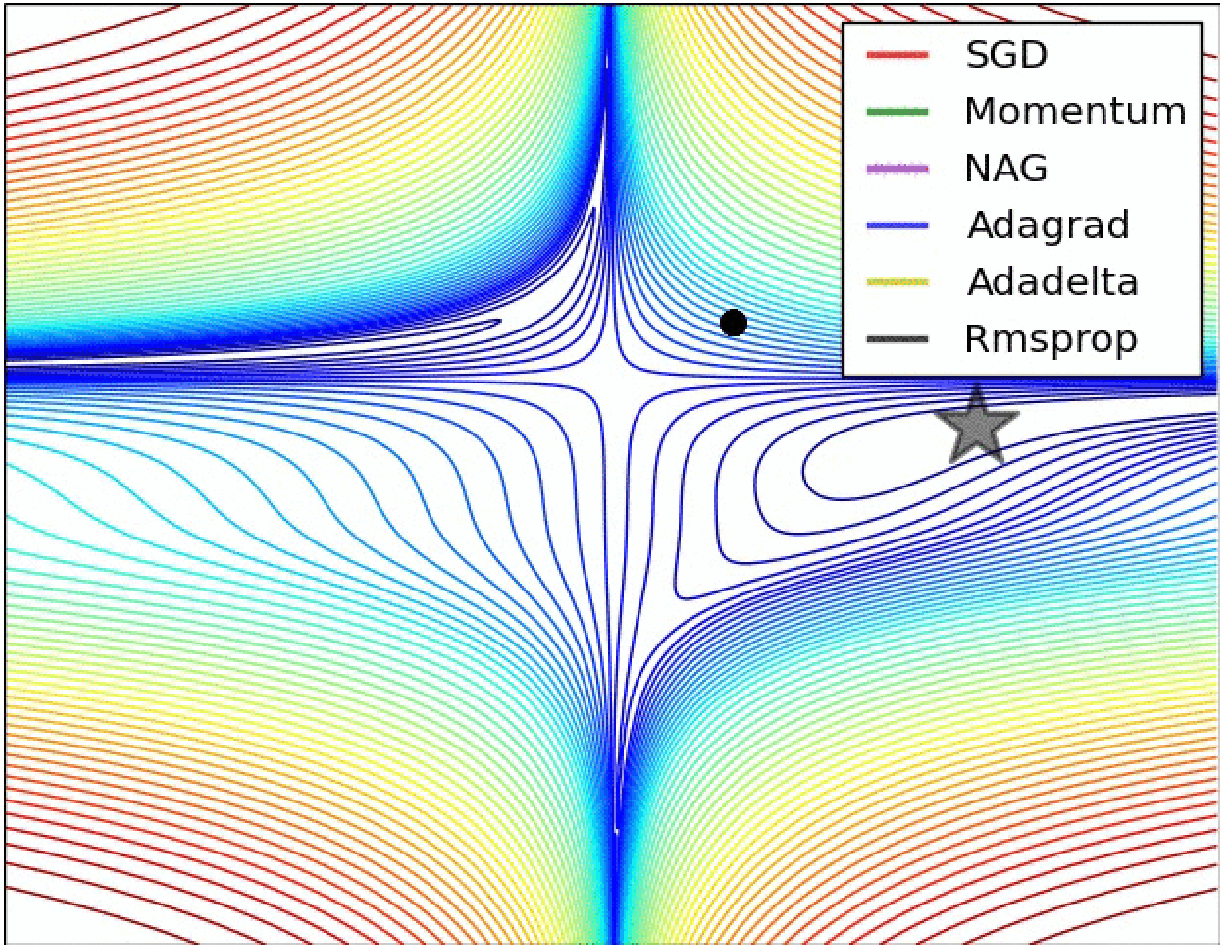
$$\frac{\partial z}{\partial x_i} = \sum_j \frac{\partial z}{\partial y_j}\frac{\partial y_j}{\partial x_i}.$$

https://google-developers.appspot.com/machine-learning/crash-course/backprop-scroll/

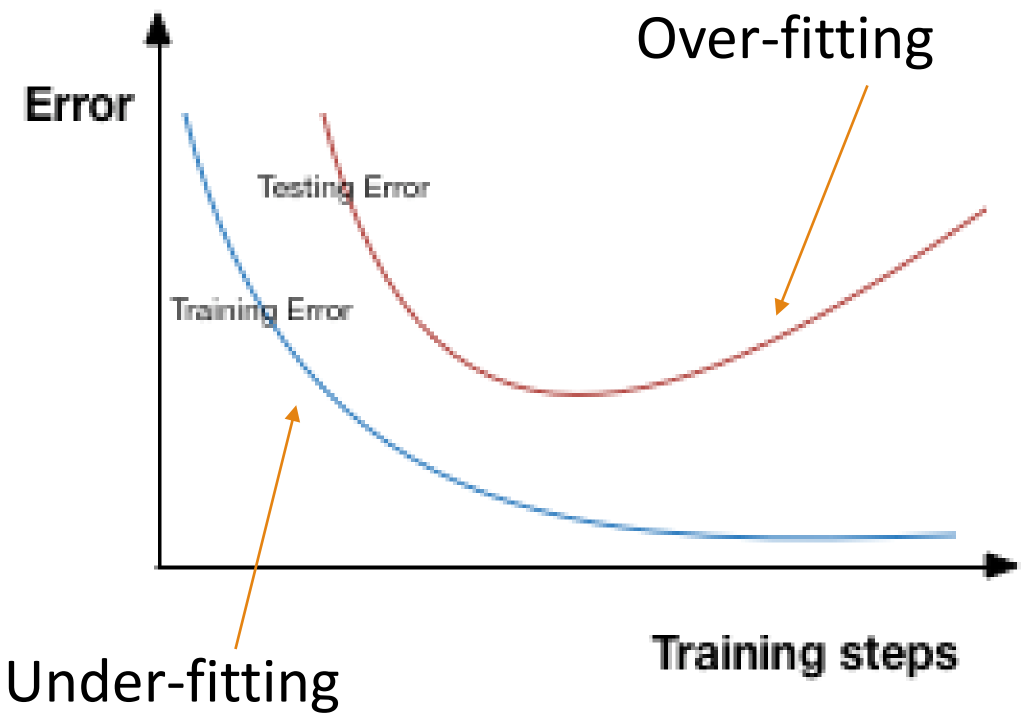
Optimization algorithms

Optimization algorithm	Core idea	Pros	Cons
SGD [140]	Computes the gradient of mini-batches iteratively and updates the parameters	• Easy to implement	• Setting a global learning rate required • Algorithm may get stuck on saddle points or local minima • Slow in terms of convergence • Unstable
Nesterov's momentum [125]	Introduces momentum to maintain the last gradient direction for the next update	• Stable • Faster learning • Can escape local minima	• Setting a learning rate needed
Adagrad [126]	Applies different learning rates to different parameters	• Learning rate tailored to each parameter • Handle sparse gradients well	• Still requires setting a global learning rate • Gradients sensitive to the regularizer • Learning rate becomes very slow in the late stages
Adadelta [141]	Improves Adagrad, by applying a self-adaptive learning rate	• Does not rely on a global learning rate • Faster speed of convergence • Fewer hyper-parameters to adjust	• May get stuck in a local minima at late training
RMSprop [140]	Employs root mean square as a constraint of the learning rate	• Learning rate tailored to each parameter • Learning rate do not decrease dramatically at late training • Works well in RNN training	• Still requires a global learning rate • Not good at handling sparse gradients
Adam [127]	Employs a momentum mechanism to store an exponentially decaying average of past gradients	• Learning rate stailored to each parameter • Good at handling sparse gradients and non-stationary problems • Memory-efficient • Fast convergence	• It may turn unstable during training

Visualization



Training Characteristics



Supervised Learning

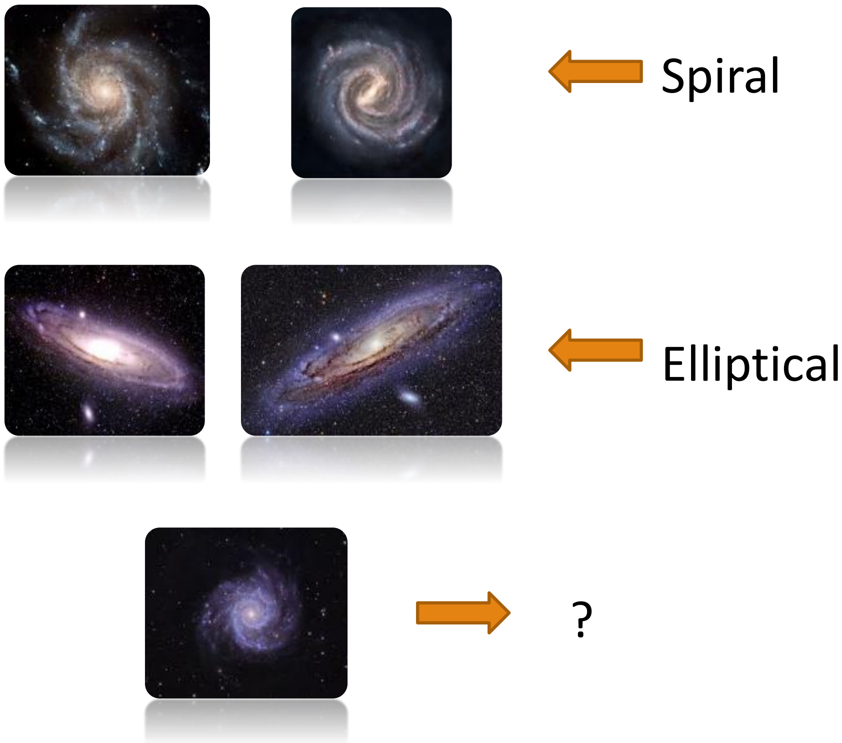
Supervised Learning

Data
Labels

Model
Prediction

Exploiting prior knowledge

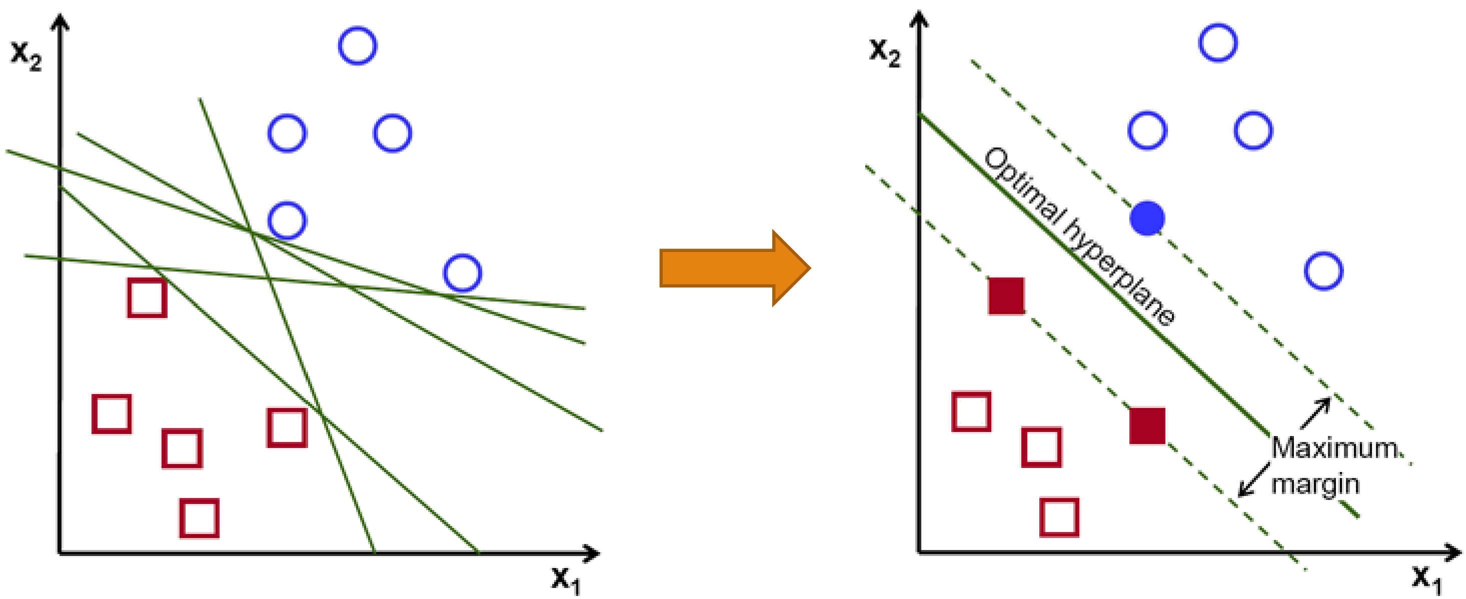
- Expert users
- Crowdsourcing
- Other instruments



State-of-the-art (before Deep Learning)

Support Vector Machines

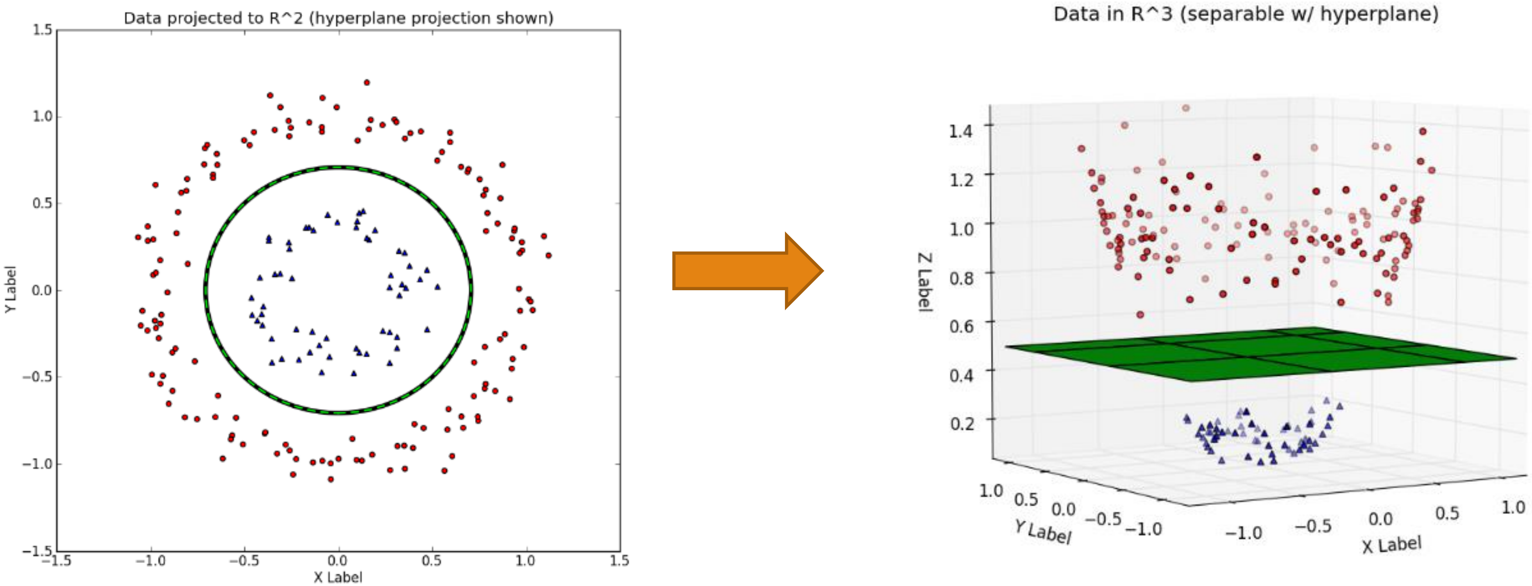
- Binary classification



State-of-the-art (before Deep Learning)

Support Vector Machines

- Binary classification
- Kernels <-> non-linearities



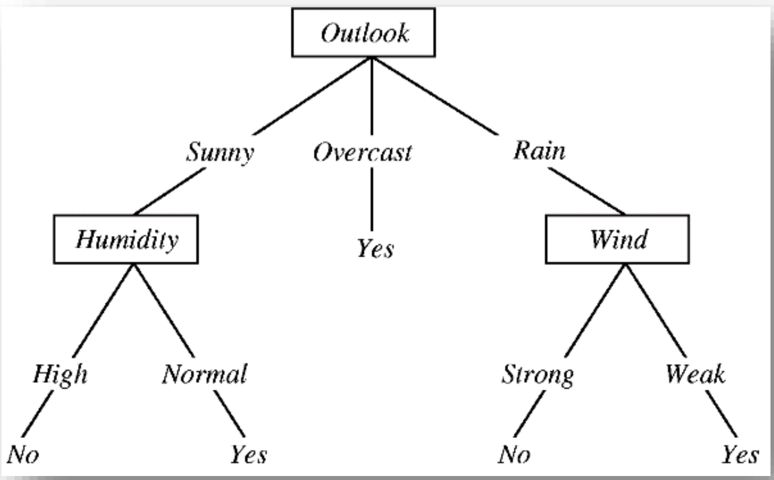
State-of-the-art (before Deep Learning)

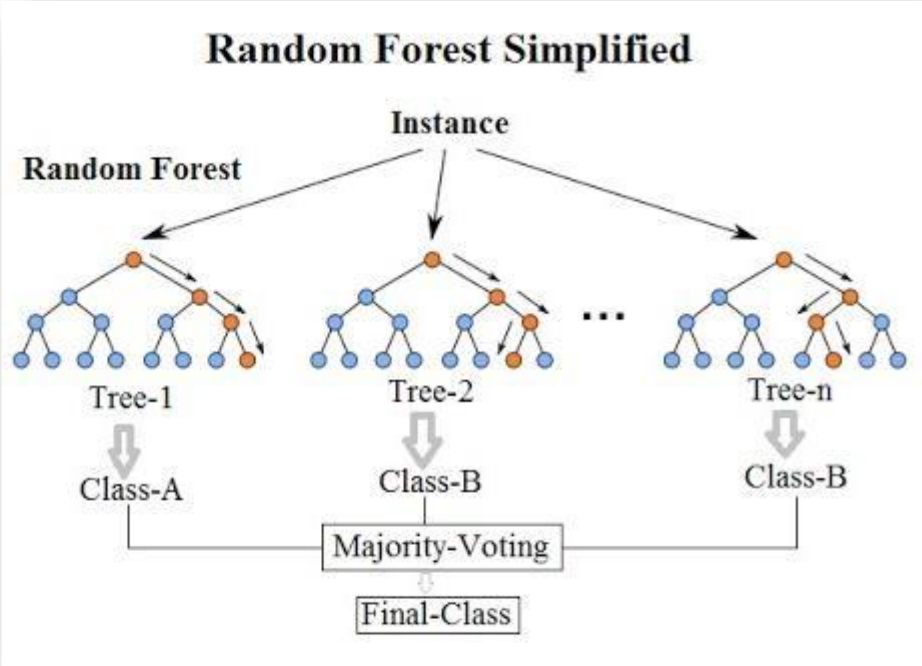
Support Vector Machines

- Binary classification
- Kernels <-> non-linearities

Random Forests

- Multi-class classification





State-of-the-art (before Deep Learning)

Support Vector Machines

- Binary classification
- Kernels <-> non-linearities

Random Forests

- Multi-class classification

Markov Chains/Fields

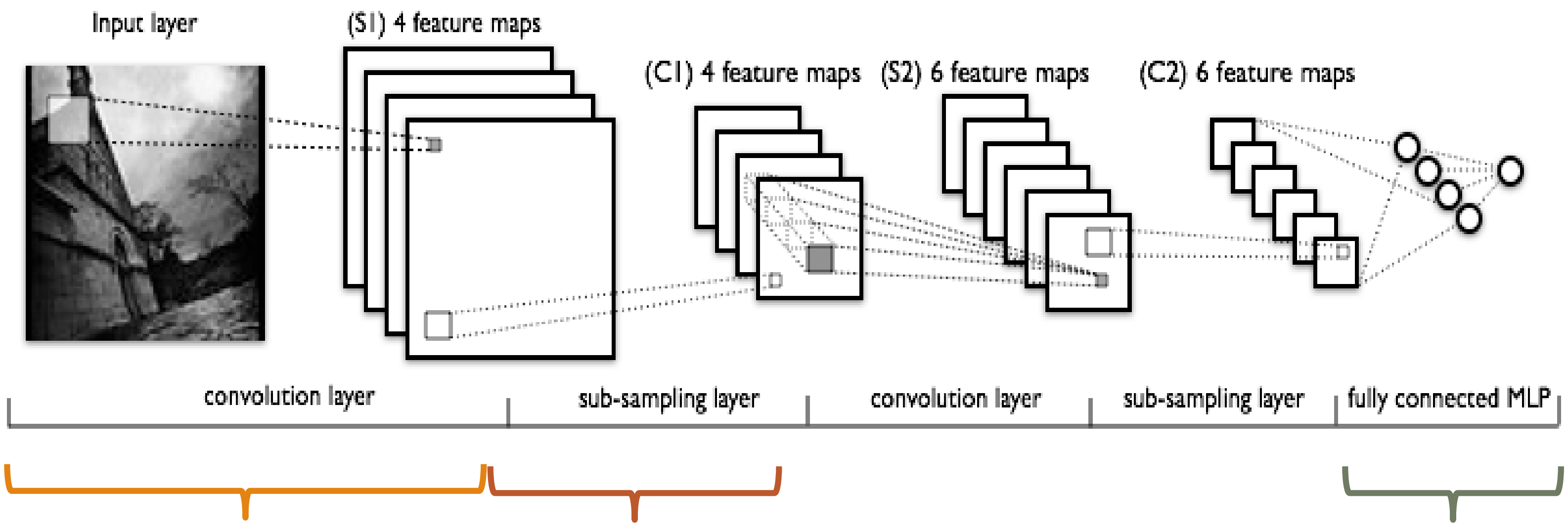
- Temporal data

State-of-the-art (since 2015)

Deep Learning (DL)

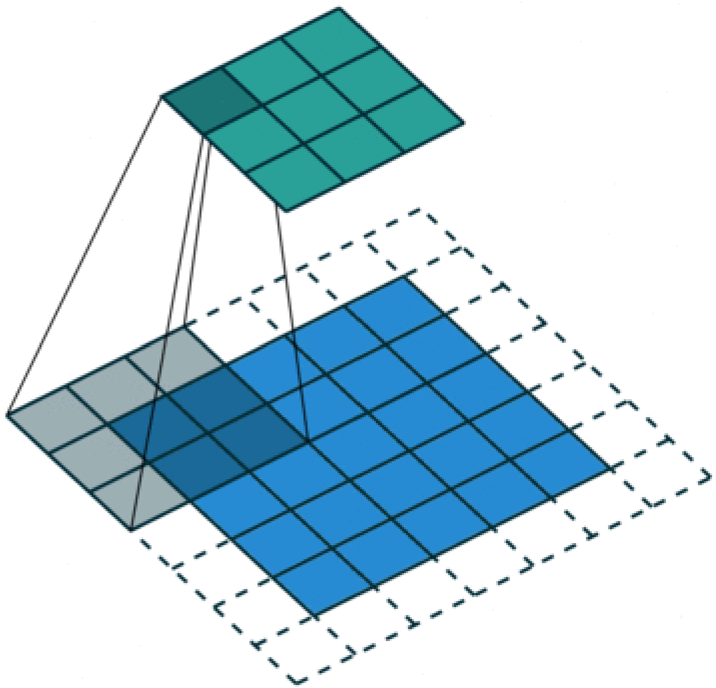
- Convolutional Neural Networks (CNN) <-> Images
- Recurrent Neural Networks (RNN) <-> Audio

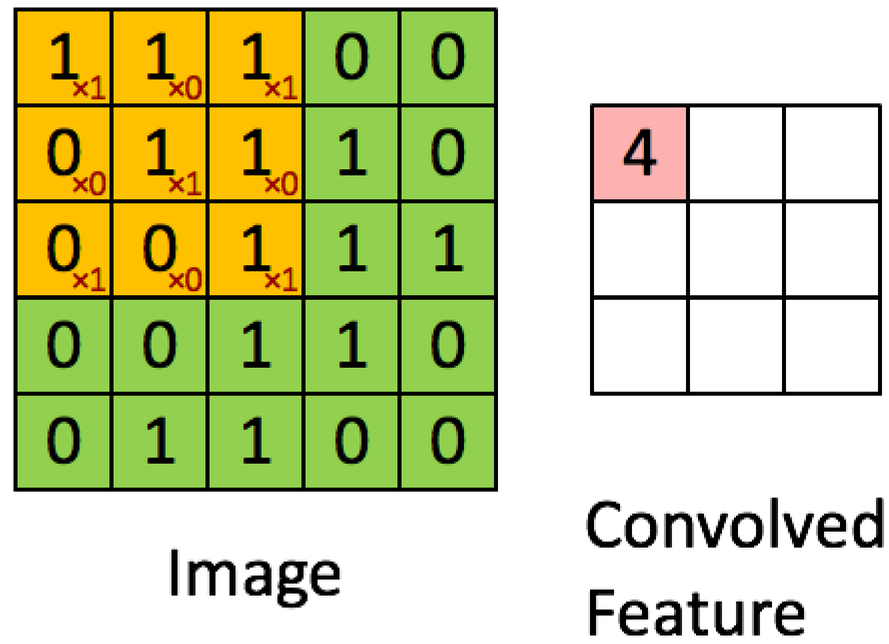
Convolutional Neural Networks



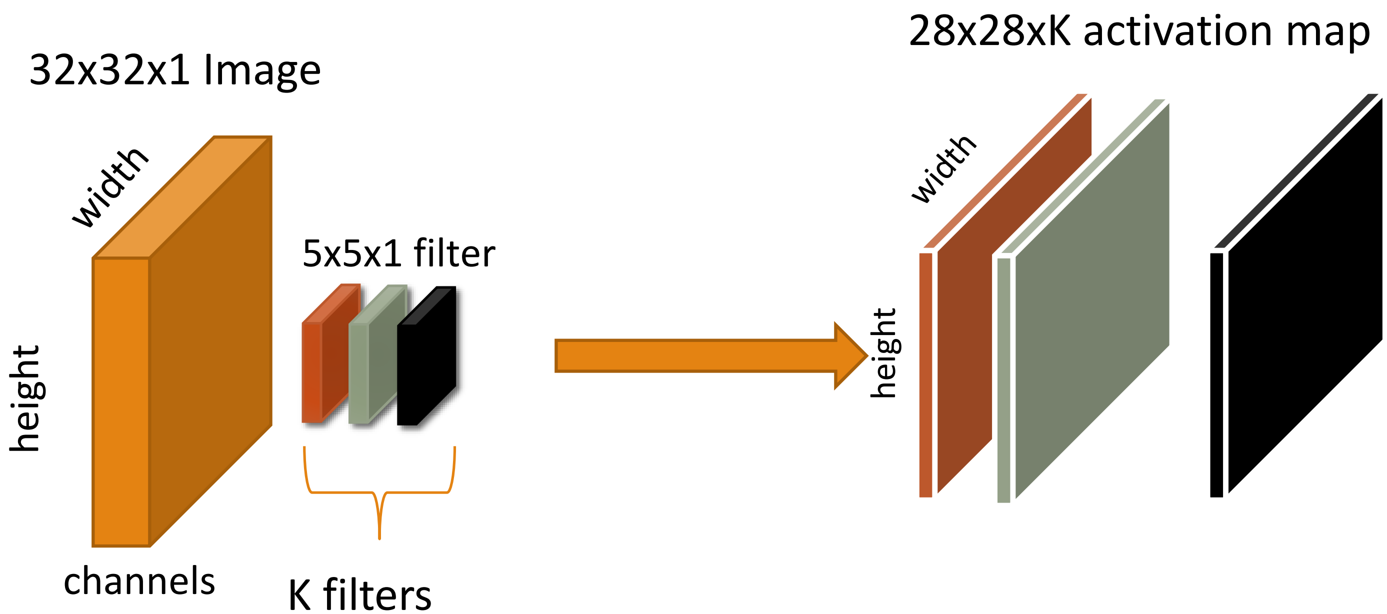
(Convolution + Subsampling) + () ... + Fully Connected

Convolution operator





Convolutional Layers



$$(I * K)_{ij} = \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} I(i-m, j-n) K(m,n)$$

$$= \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} I(i+m, j+n) K(-m,-n)$$

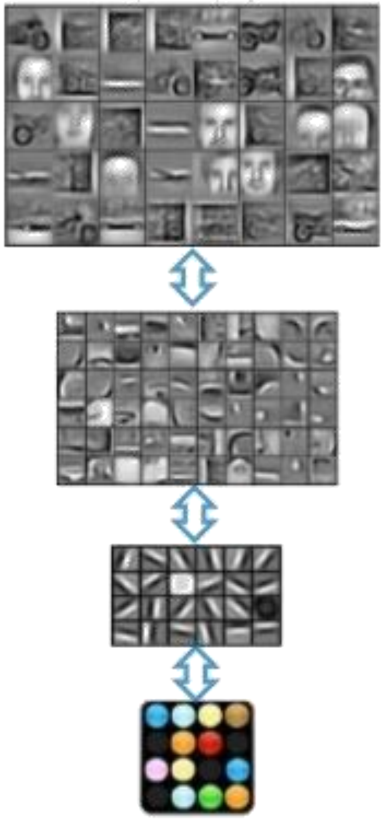
Convolutional Layers

Characteristics

- Hierarchical features
- Location invariance

Parameters

- Number of filters (32,64…)
- Filter size (3x3, 5x5)
- Stride (1)
- Padding (2,4)



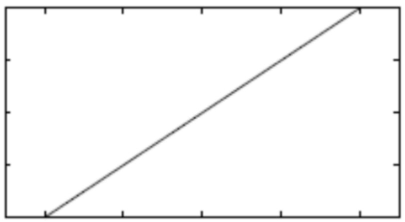
“Machine Learning and AI for Brain Simulations” – Andrew Ng Talk, UCLA, 2012

Activation Layer

Introduction of non-linearity

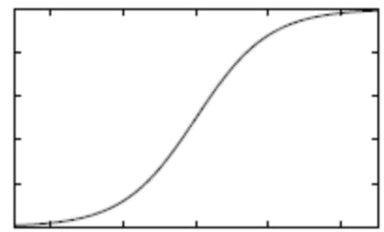
- Brain: thresholding -> spike trains

Identity (Linear)



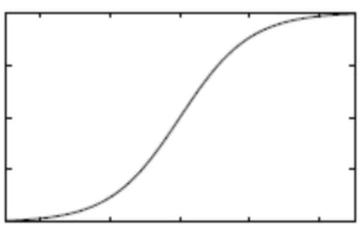
$$identity(x) = x$$

Sigmoid



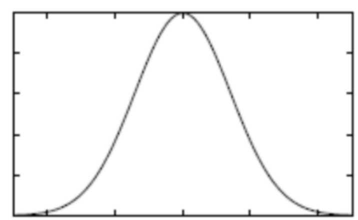
$$sigmoid(x) = \frac{1}{1 + e^{-x}}$$

Tanh (Hypertangent)



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

Gaussian



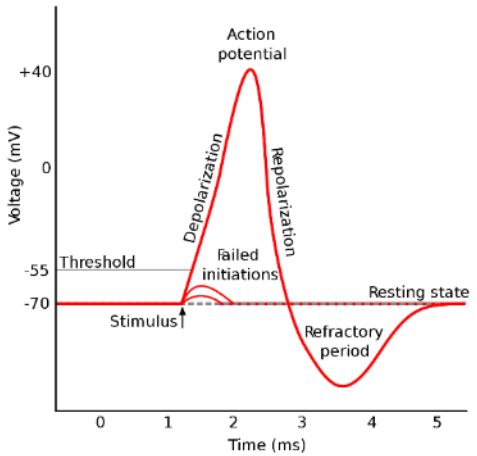
$$gaussian(x) = e^{-x^2/\sigma^2}$$

Activation Layer

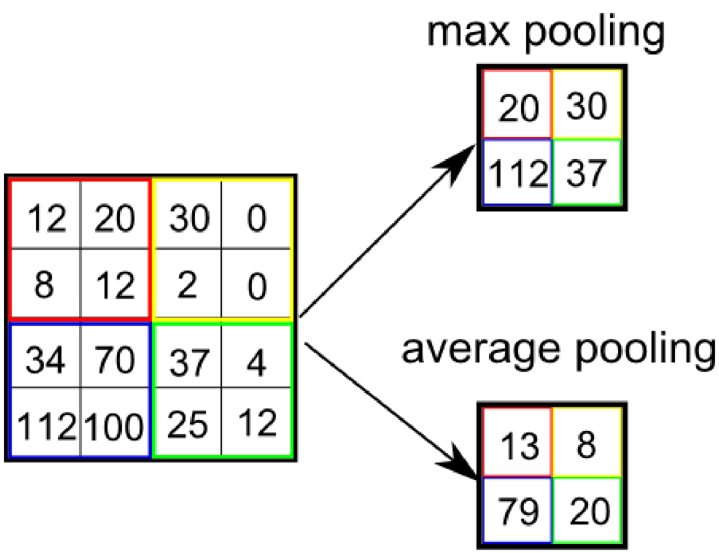
ReLU: x=max(0,x)

- ✓ Simplifies backprop
- ✓ Makes learning faster
- ✓ Avoids saturation issues
- ✓ ~ non-negativity constraint

(Note: The brain)



Subsampling (pooling) Layers



<-> downsampling

➢ Scale invariance

Parameters

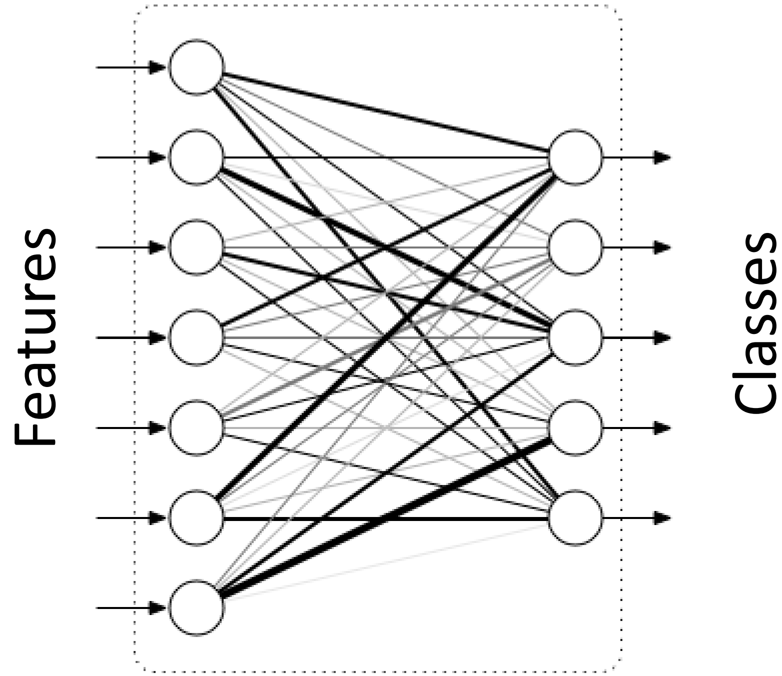
- Type
- Filter Size
- Stride

Fully Connected Layers

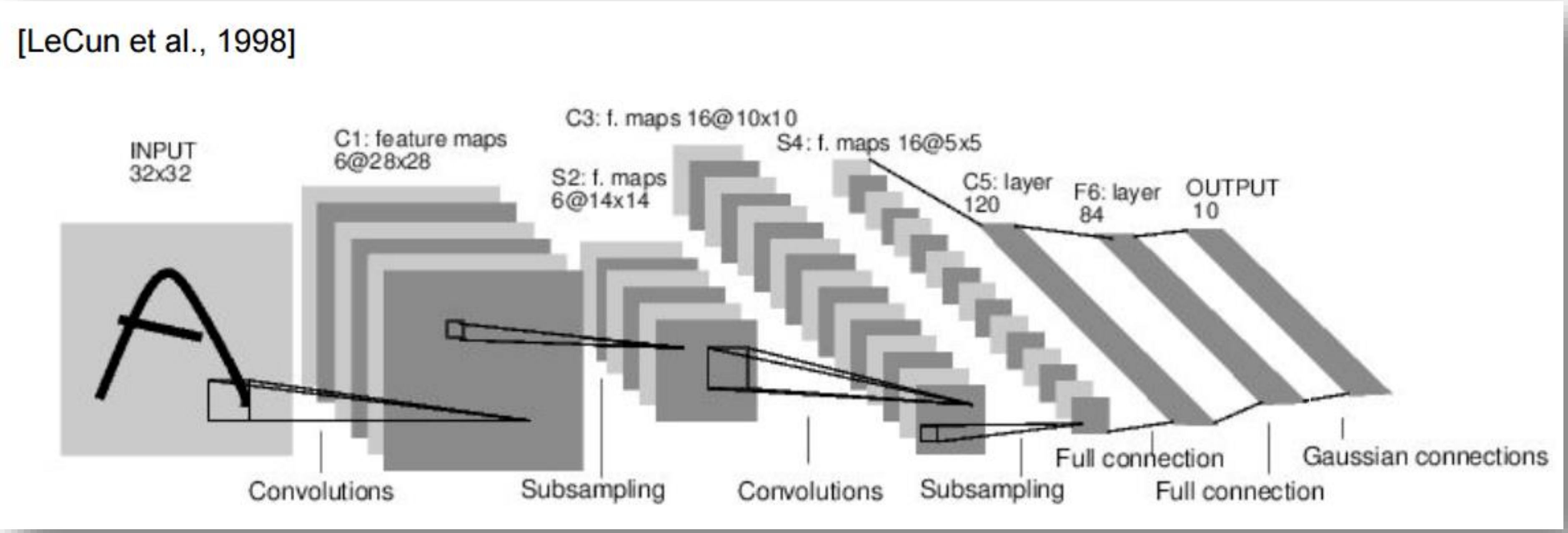
Full connections to all activations in previous layer

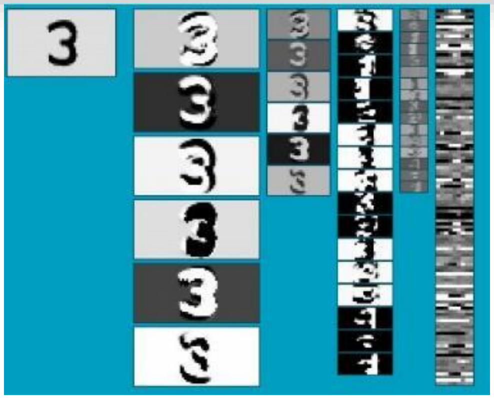
Typically at the end

Can be replaced by conv

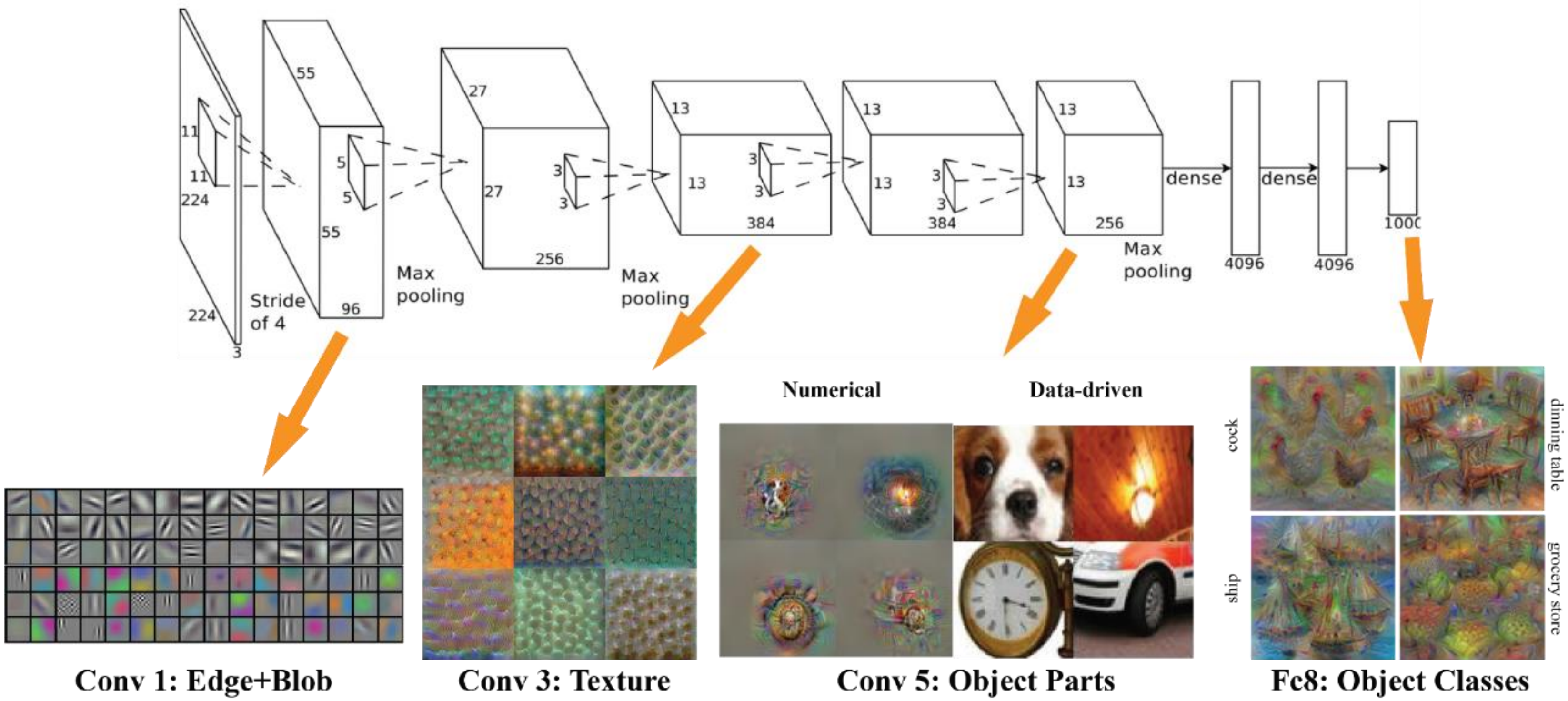


LeNet [1998]


[LeCun et al., 1998]
INPUT
32x32
C1: feature maps
6@28x28
S2: f. maps
6@14x14
C3: f. maps 16@10x10
S4: f. maps 16@5x5
C5: layer
120
F6: layer
84
OUTPUT
10
Convolutions
Subsampling
Convolutions
Subsampling
Full connection
Full connection
Gaussian connections

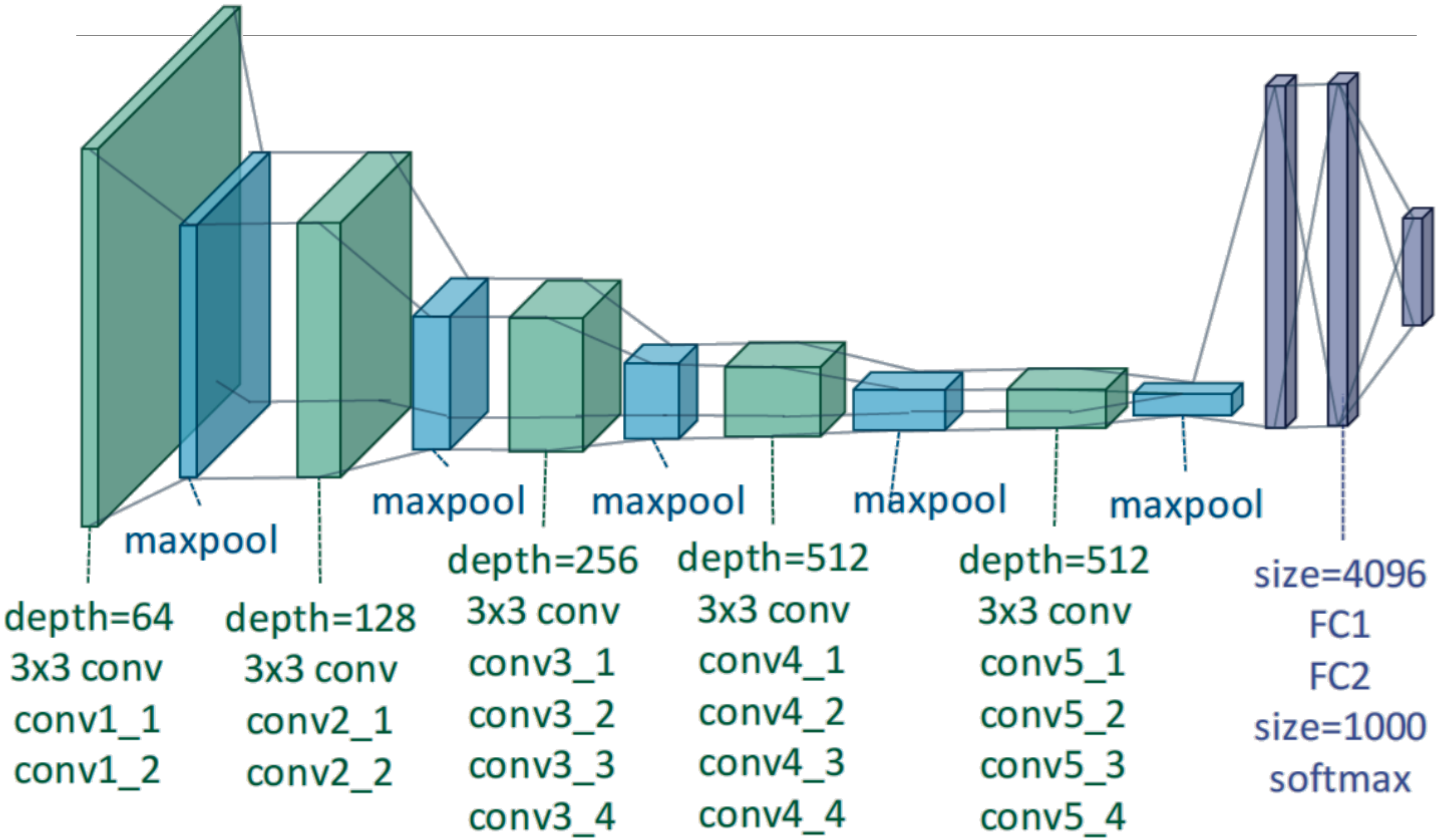


AlexNet [2012]



Alex Krizhevsky, Ilya Sutskever and Geoff Hinton, [ImageNet ILSVRC challenge]() in 2012
http://vision03.csail.mit.edu/cnn_art/data/single_layer.png

VGGnet [2014]



K. Simonyan, A. Zisserman Very Deep Convolutional Networks for Large-Scale Image Recognition, arXiv technical report, 2014

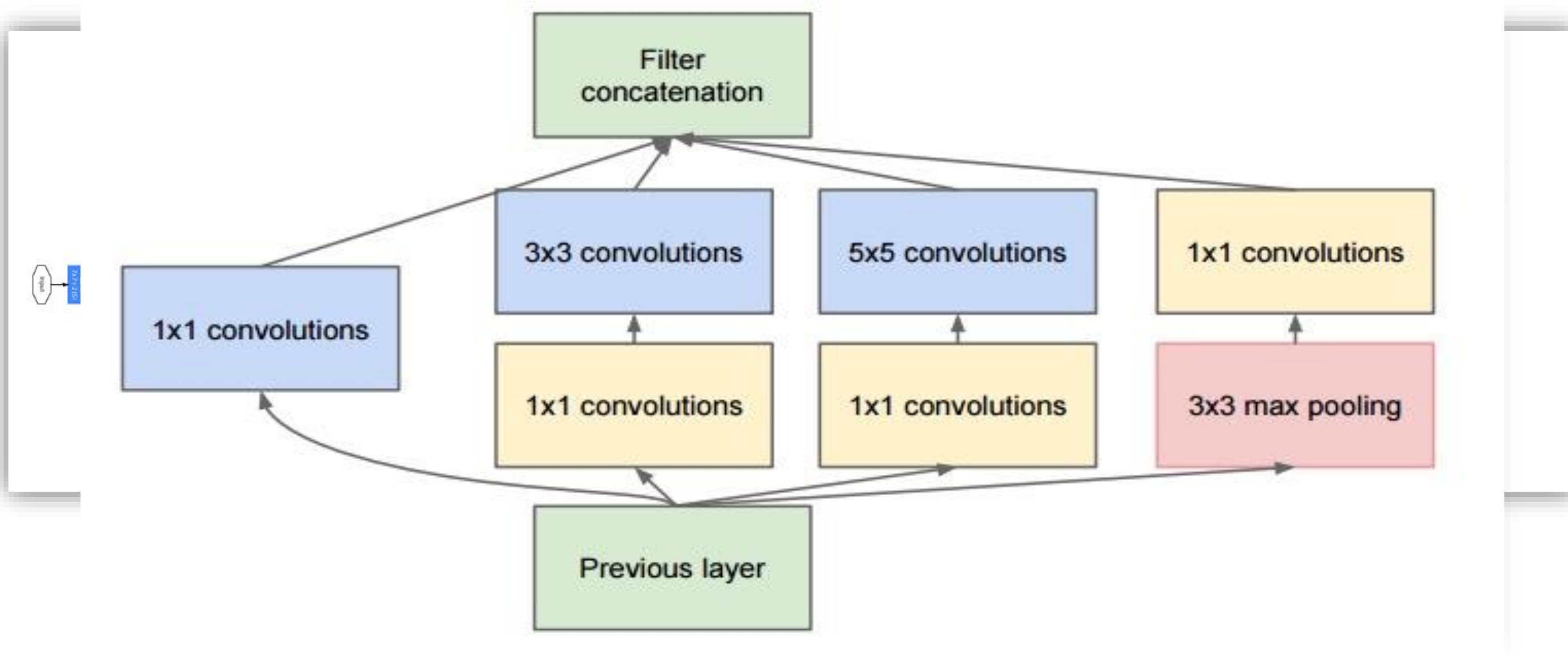
VGGnet

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

D: VGG16
E: VGG19
All filters are 3x3

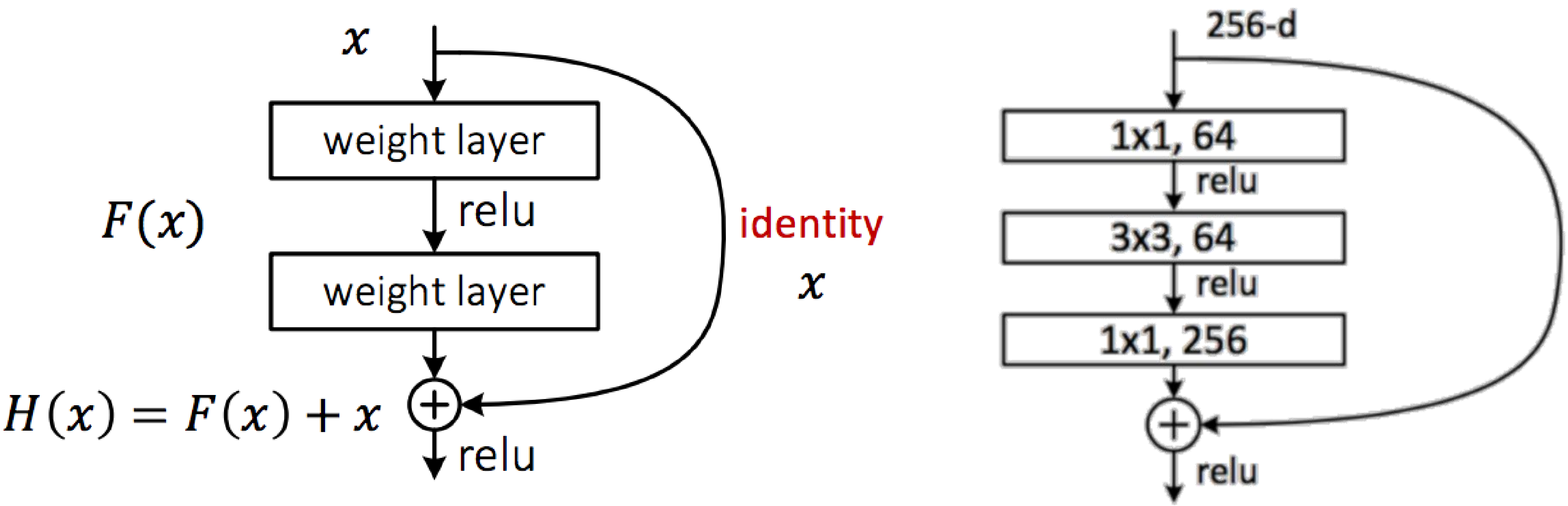
More layers
smaller filters

Inception (GoogLeNet, 2014)



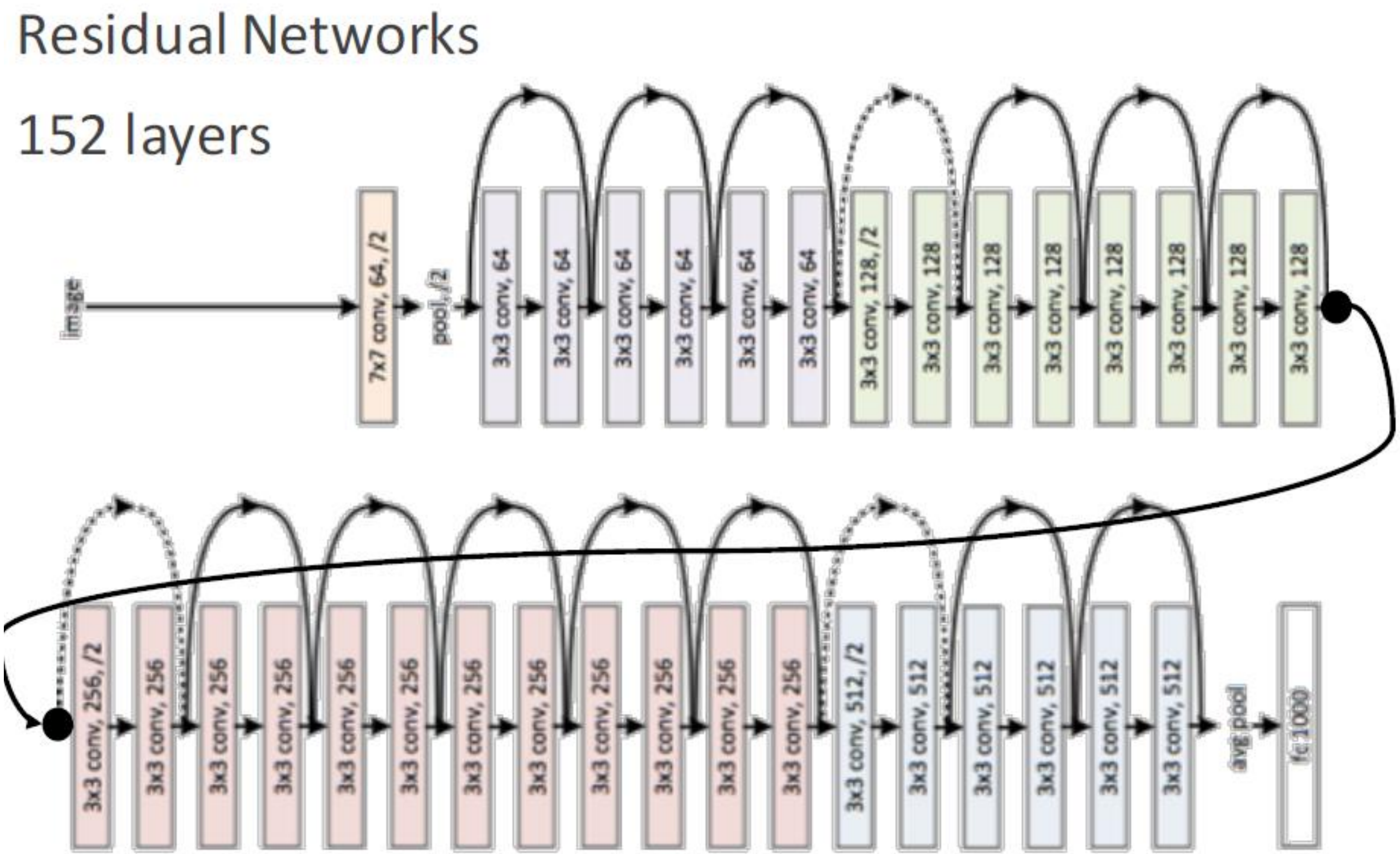
Inception module with dimensionality reduction

Residuals



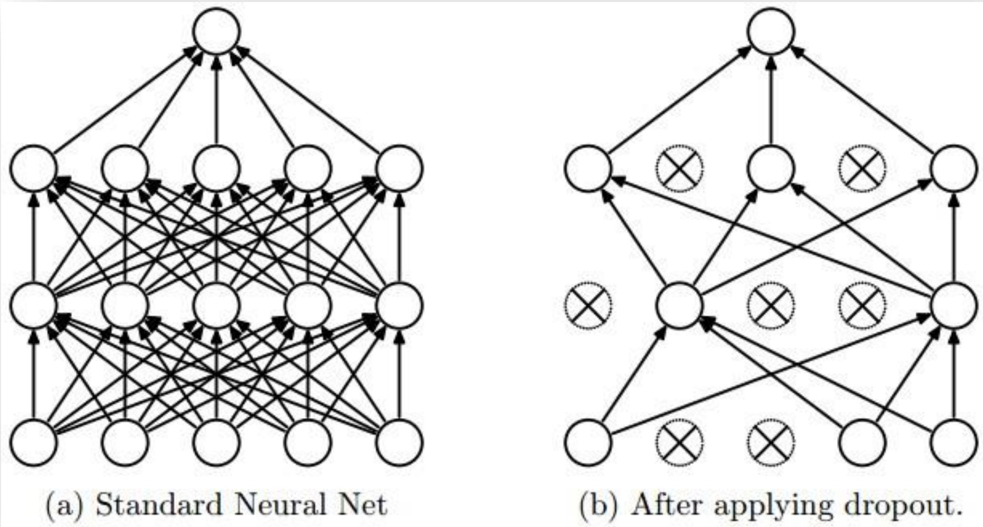
x
weight layer
relu
weight layer
$F(x)$
identity
x
$H(x) = F(x) + x$
relu
256-d
1x1, 64
relu
3x3, 64
relu
1x1, 256
relu

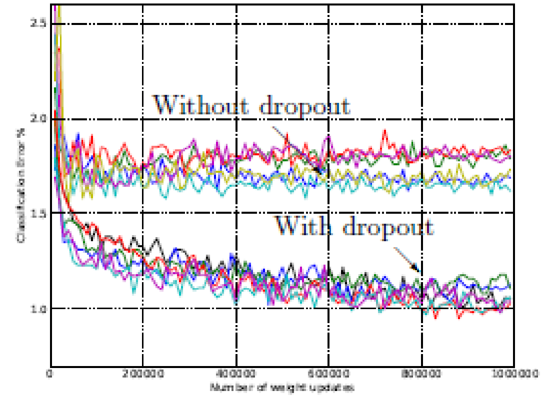
ResNet, 2015

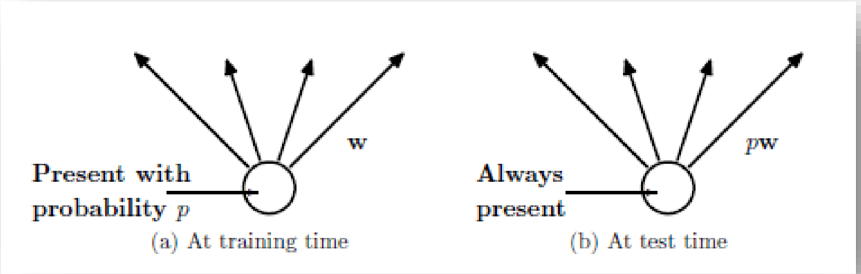


He, Kaiming, et al. "Deep residual learning for image recognition." *IEEE CVPR*. 2016.

Dropout







Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." *Journal of machine learning research*15.1 (2014): 1929-1958.

Batch Normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

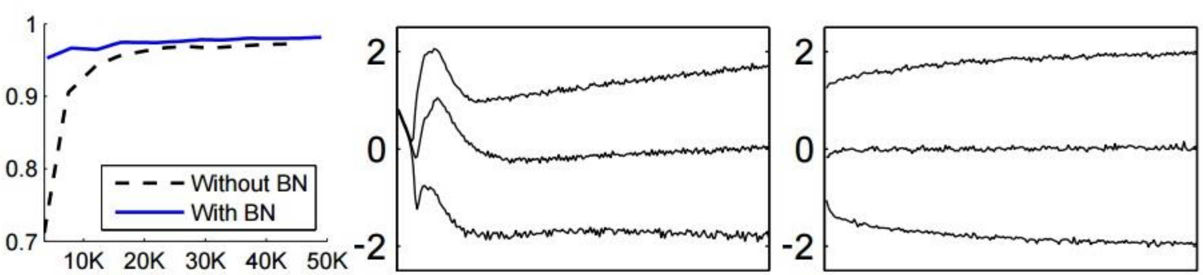
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

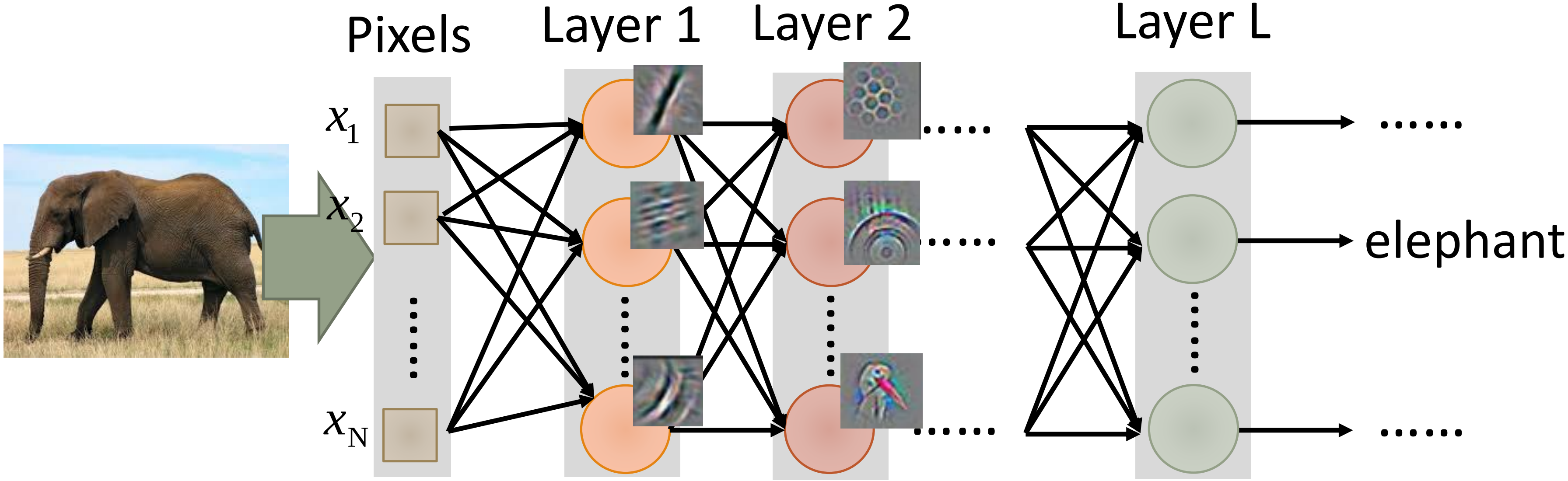
$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$



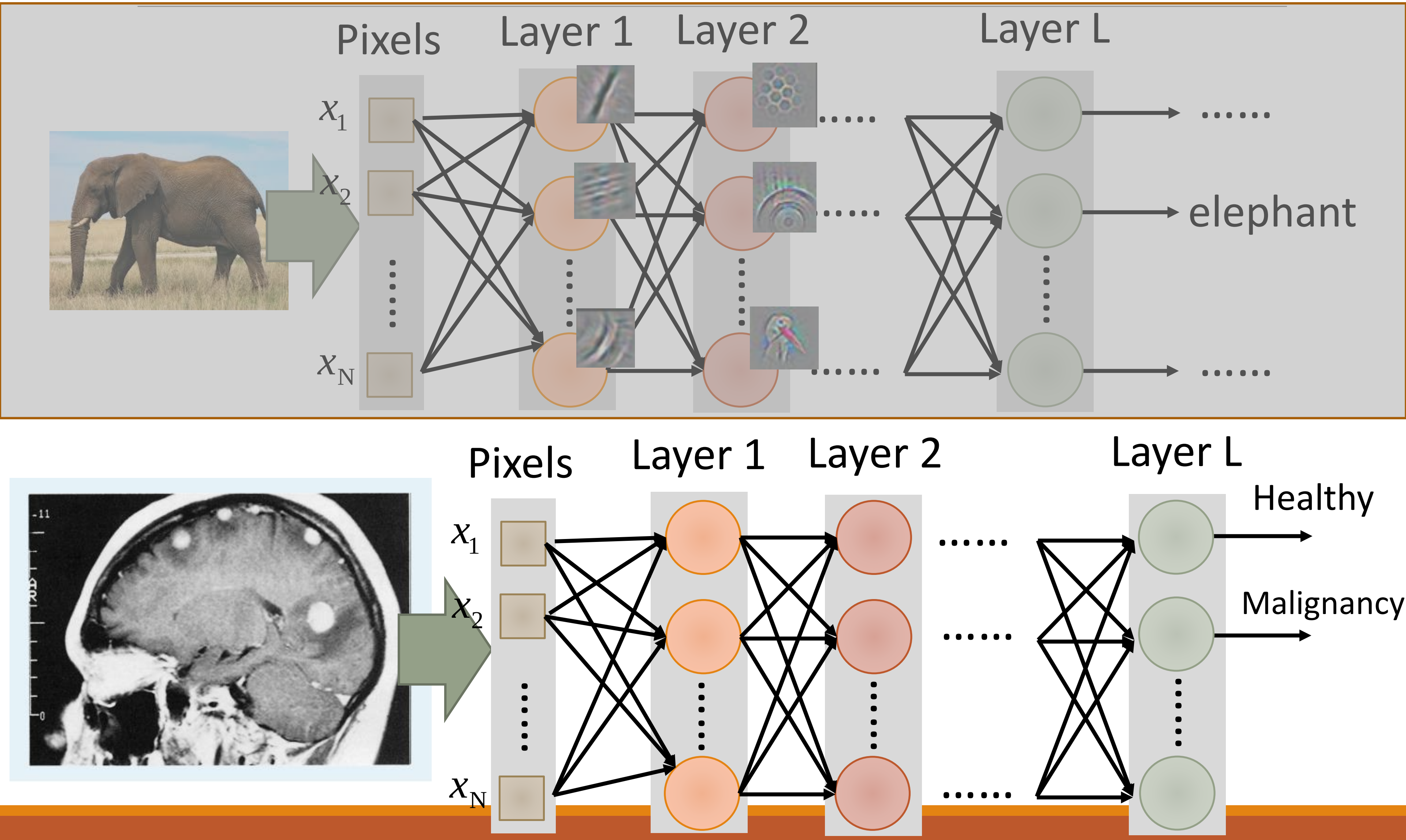
(a) (b) Without BN (c) With BN

Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift [Ioffe and Szegedy 2015]

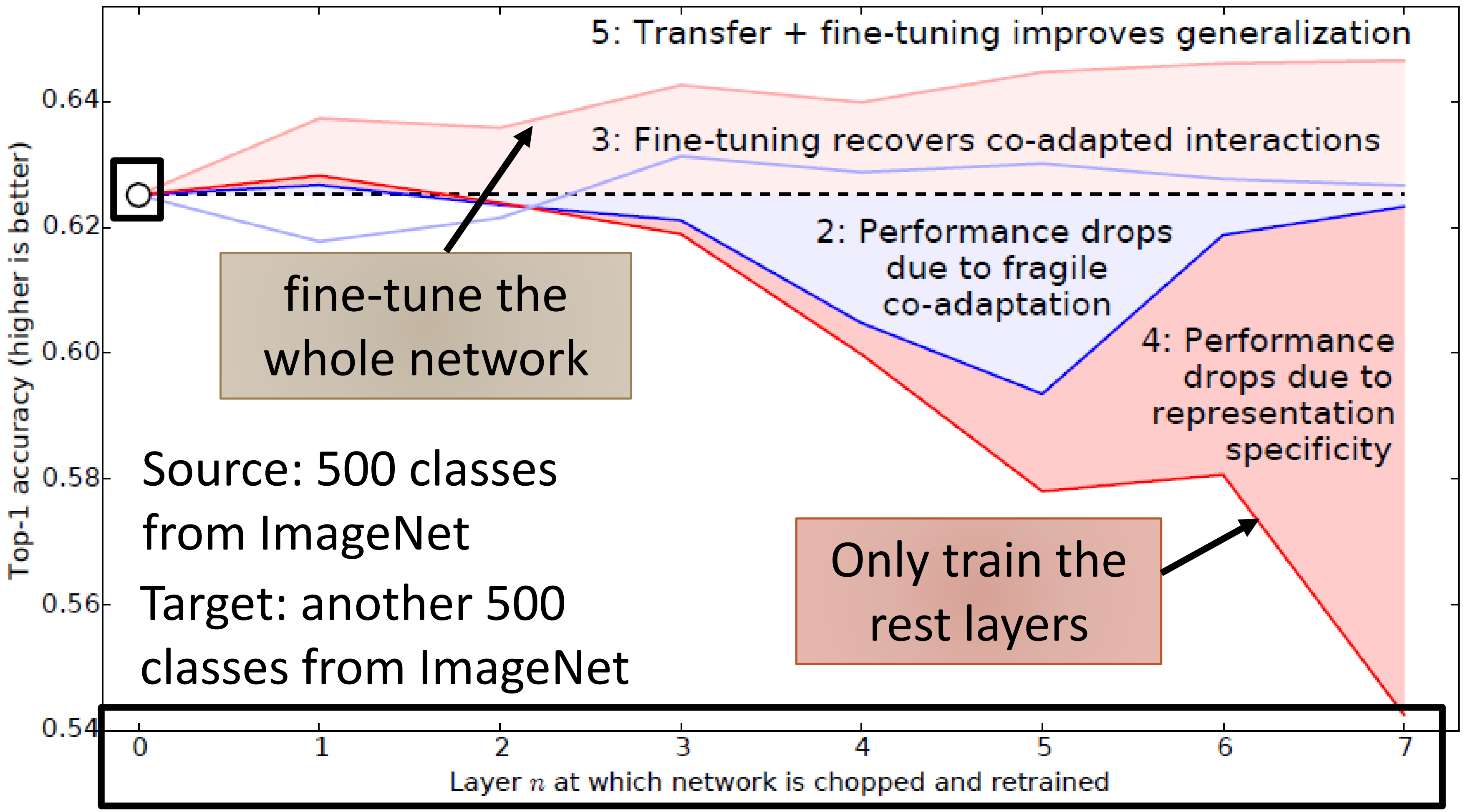
Transfer Learning



Transfer Learning



Layer Transfer - Image



Jason Yosinski, Jeff Clune, Yoshua Bengio, Hod Lipson, “How transferable are features in deep neural networks?”, NIPS, 2014

Deep Learning Applications in Astronomy

Gregory Tsagkatakis

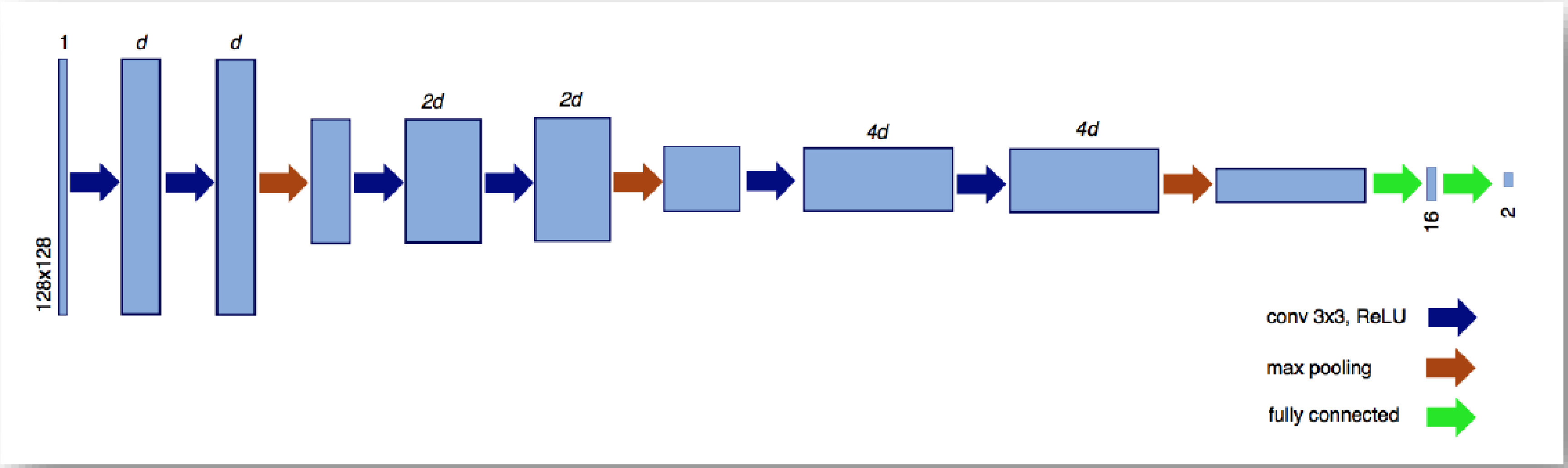
CSD – UOC & ICS – FORTH

http://users.ics.forth.gr/~greg/

DL for galaxy morphology

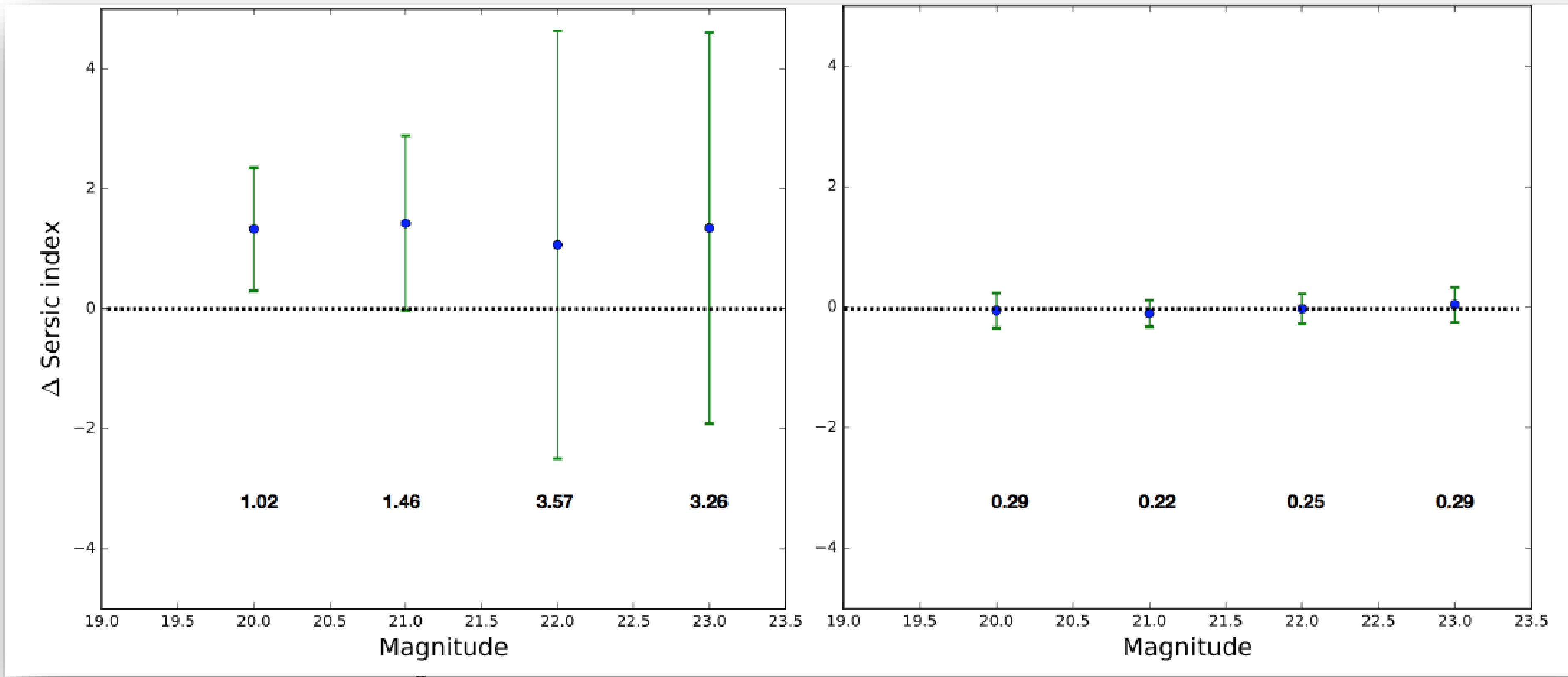
Recovery of galaxy parameters for HST images

Simulation of 31K galaxies (24K training), H band PSF, CANDELS survey noise



Tuccillo, D., Etienne Decencìère, and Santiago Velasco-Forero. "Deep learning for studies of galaxy morphology." *Proceedings of the International Astronomical Union* 12.S325 (2016): 191-196.

DL for of galaxy morphology (con't)



GALFIT

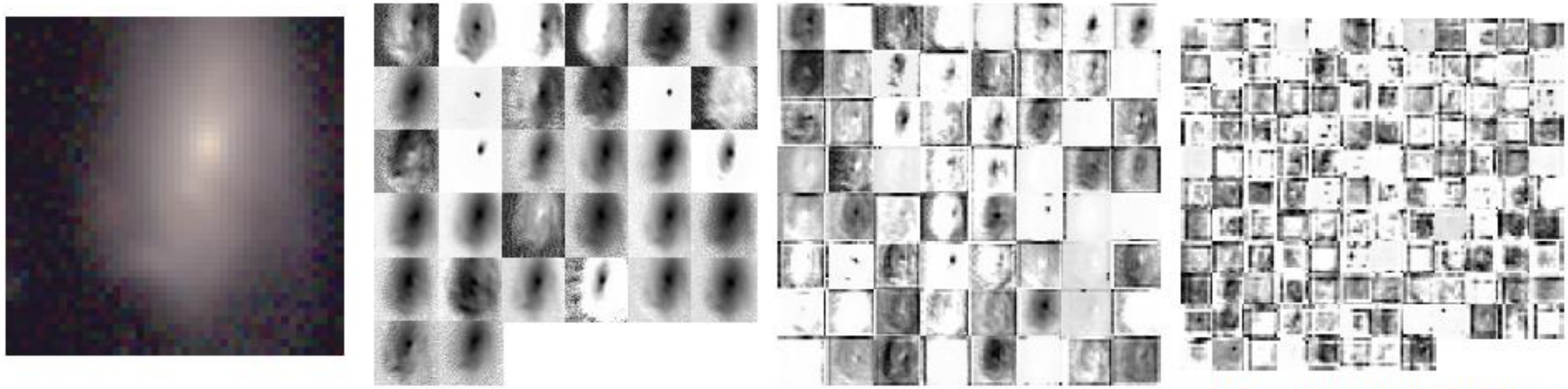
CNN

CNN: Star-galaxy Classification

type	filters	filter size	padding	non-linearity	initial weights	initial biases
convolutional	32	5×5	-	leaky ReLU	orthogonal	0.1
convolutional	32	3×3	1	leaky ReLU	orthogonal	0.1
pooling	-	2×2	-	-	-	-
convolutional	64	3×3	1	leaky ReLU	orthogonal	0.1
convolutional	64	3×3	1	leaky ReLU	orthogonal	0.1
convolutional	64	3×3	1	leaky ReLU	orthogonal	0.1
pooling	-	2×2	-	-	-	-
convolutional	128	3×3	1	leaky ReLU	orthogonal	0.1
convolutional	128	3×3	1	leaky ReLU	orthogonal	0.1
convolutional	128	3×3	1	leaky ReLU	orthogonal	0.1
pooling	-	2×2	-	-	-	-
fully-connected	2048	-	-	leaky ReLU	orthogonal	0.01
fully-connected	2048	-	-	leaky ReLU	orthogonal	0.01
fully-connected	2	-	-	softmax	orthogonal	0.01

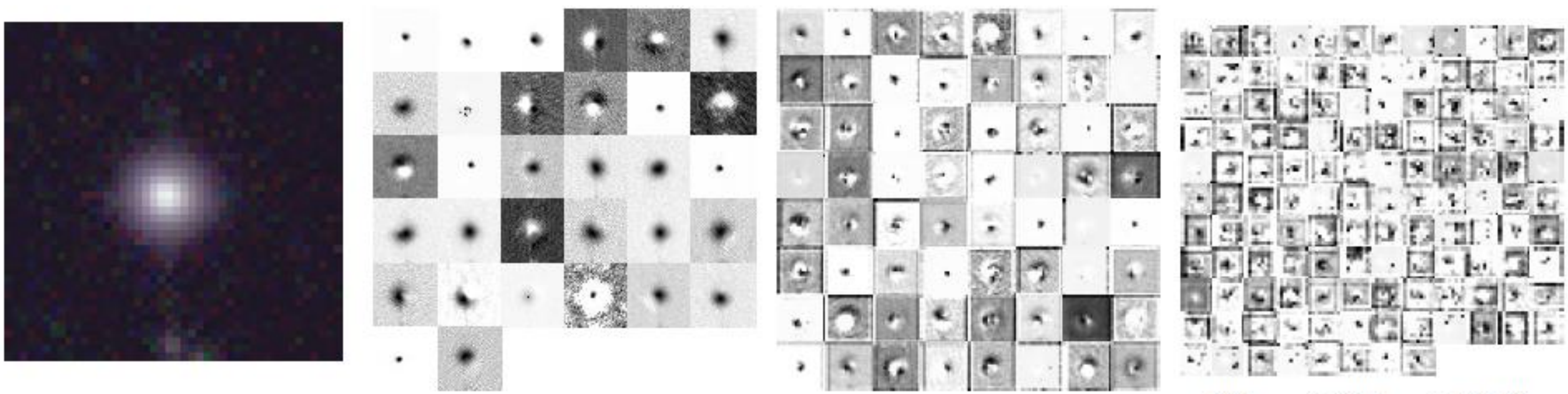
Kim, Edward J., and Robert J. Brunner. "Star-galaxy classification using deep convolutional neural networks." *Monthly Notices of the Royal Astronomical Society* (2016): stw2672.

Galaxy



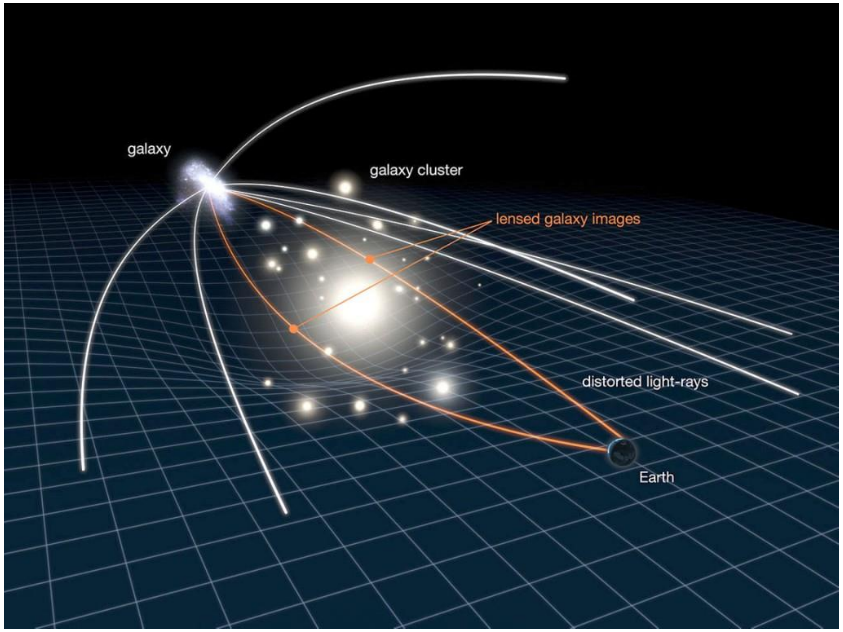
(a) Input (5 bands×44×44) (b) Layer 1 (32 maps×40×40) (c) Layer 3 (64 maps×20×20) (d) Layer 6 (128 maps×10×10)

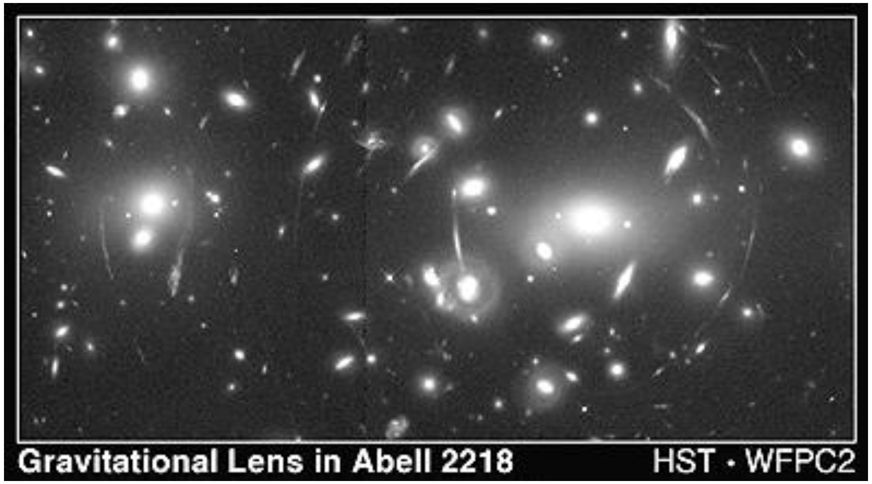
Star



(a) Input (5 bands×44×44) (b) Layer 1 (32 maps×40×40) (c) Layer 3 (64 maps×20×20) (d) Layer 6 (128 maps×10×10)

Gravitational Lensing

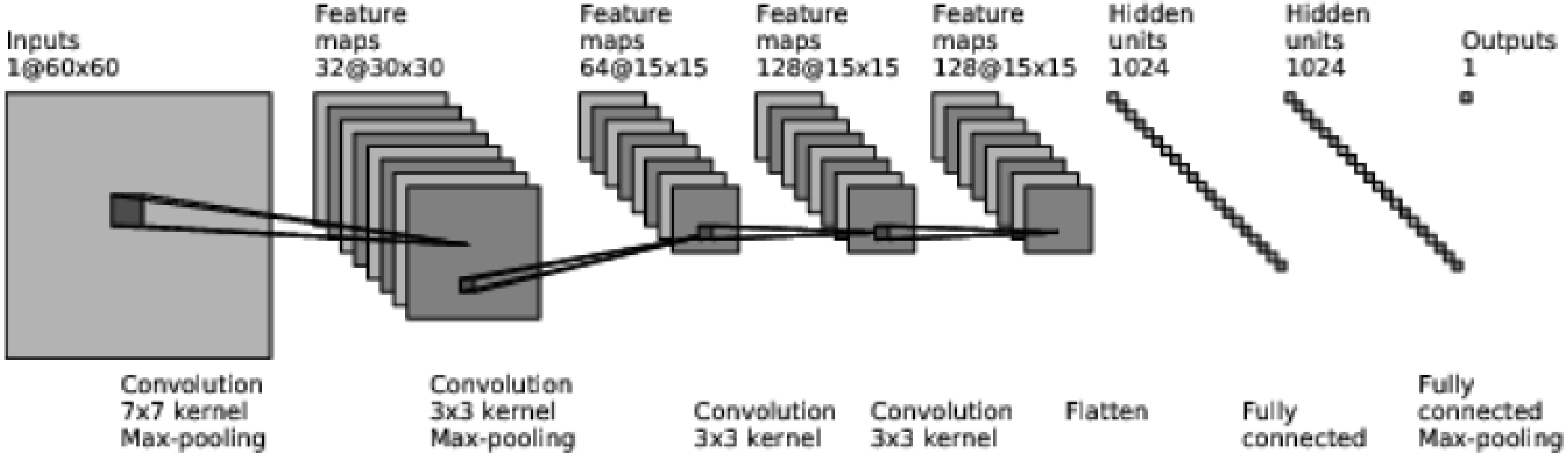




CNN for lensing

- CNNs in Kilo Degree Survey

21789 colour-magnitude selected Luminous Red Galaxies, of which 3 are known lenses, the CNN retrieves 761 strong-lens candidates and correctly classifies 2/3 of known lenses.



Petrillo, C. E., C. Tortora, S. Chatterjee, G. Vernardos, et al. "Finding strong gravitational lenses in the Kilo Degree Survey with convolutional neural networks." Monthly Notices of the Royal Astronomical Society 472, no. 1 (2017)

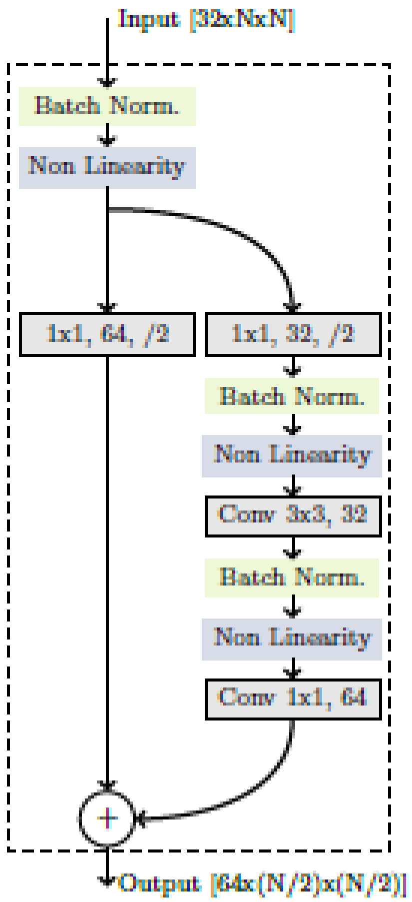
DeepLens

- Training

  20,000 LSST-like observations

- Testing

  for a rejection rate of non-lenses of 99%, a completeness of 90% can be achieved for lenses with Einstein radii larger than 1.400 and S/N larger than 20 on individual g-band LSST exposures.



Lanusse, Francois, et al. "CMU DeepLens: Deep Learning For Automatic Image-based Galaxy-Galaxy Strong Lens Finding." arXiv preprint arXiv:1703.02642 (2017).

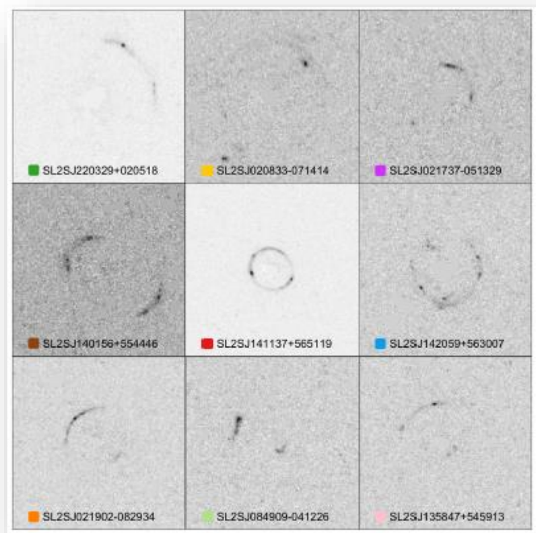
Detecting strong lensing

- Strong galaxy-galaxy lensing systems
- CA-FR-HA Telescope Legacy Survey (CFHTLS)
- Ensemble of trained DL networks
- Search of 1.4 million early type galaxies selected from the survey catalog as potential deflectors,
- Identified 2,465 candidates (117 previously known lens candidates, 29 confirmed lenses, 266 novel probable or potential lenses and 2097 false positives.

Jacobs, Colin, et al. "Finding strong lenses in CFHTLS using convolutional neural networks." Monthly Notices of the Royal Astronomical Society 471.1 (2017)

Fast Strong Gravitational Lenses analysis

- Typical ML approaches: single lens → few weeks & experts
- Estimation of lensing parameters via CNN
  - Singular Isothermal Ellipsoid density profile
  - Parameters: Einstein radius, complex ellipticity, the coordinates of the lens center
- Lens removal through ICA

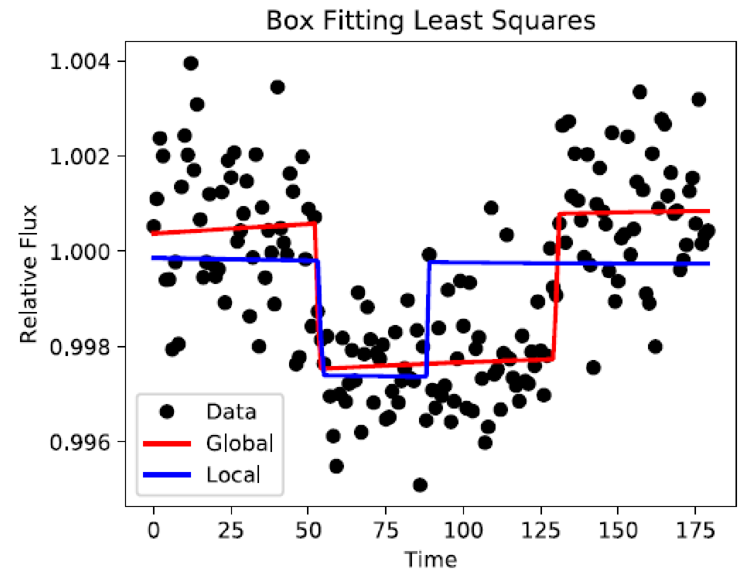


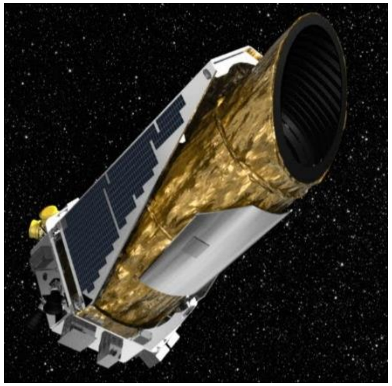
Hezaveh, Yashar D., Laurence Perreault Levasseur, and Philip J. Marshall. "Fast automated analysis of strong gravitational lenses with convolutional neural networks." *Nature* 548.7669 (2017)

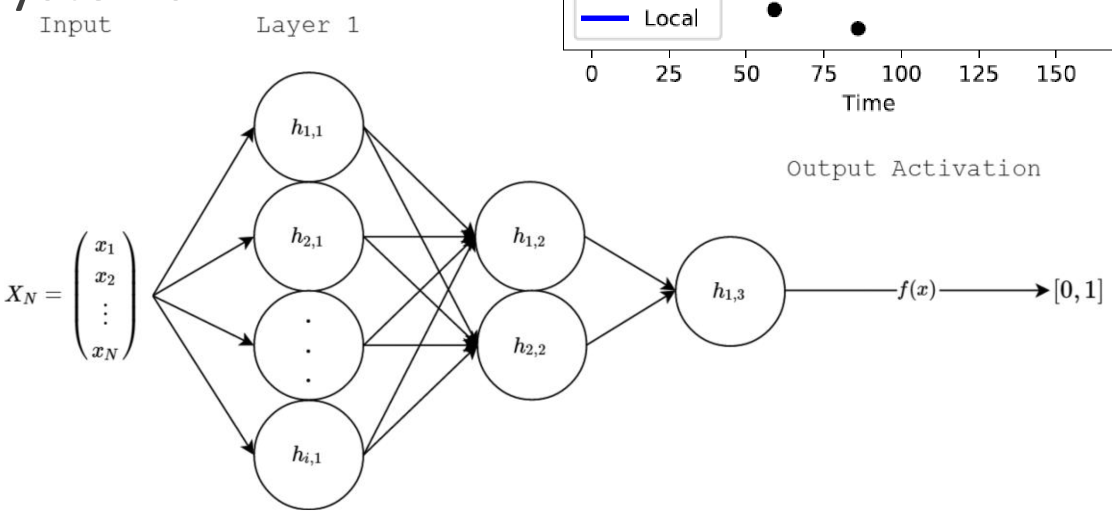
Exoplanet detection

The ideal algorithm should be

- fast
- robust to noise
- capable of learning and abstracti highly non-linear systems.

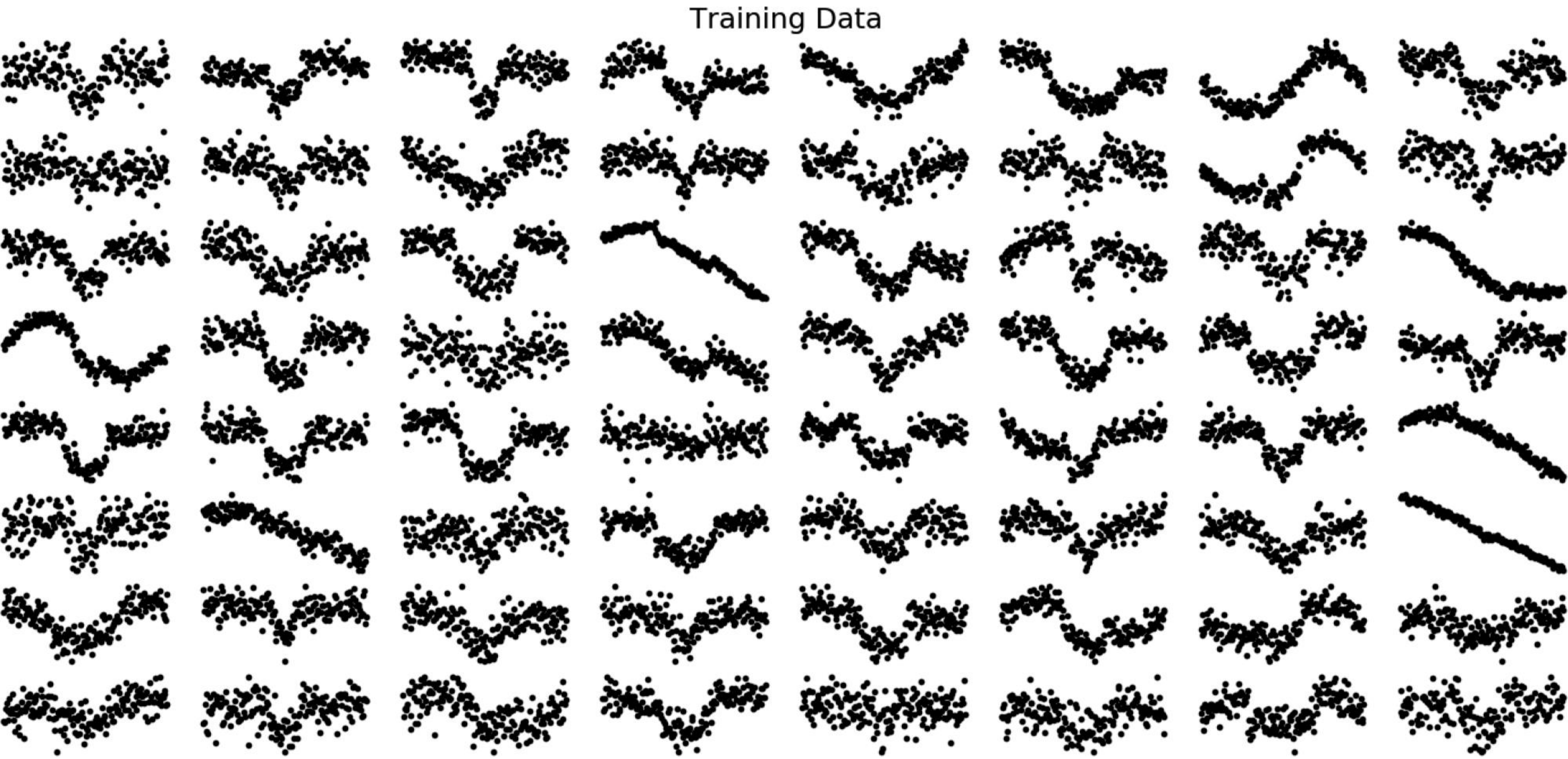




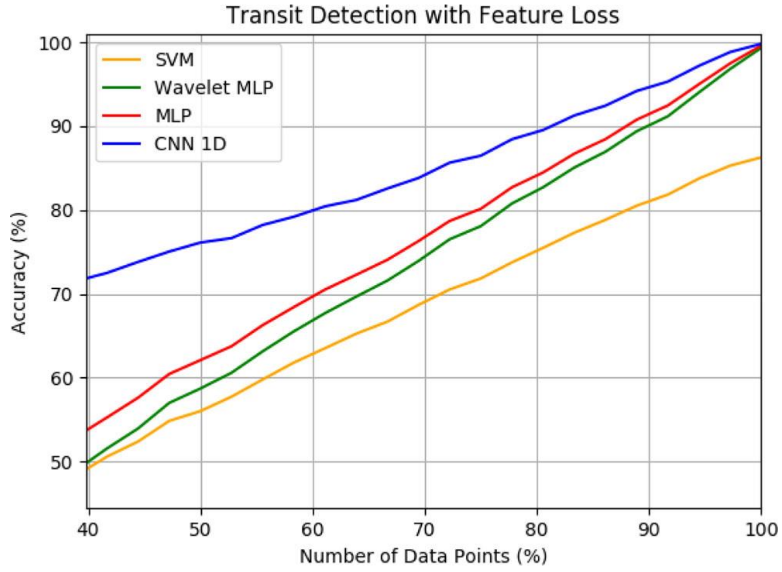


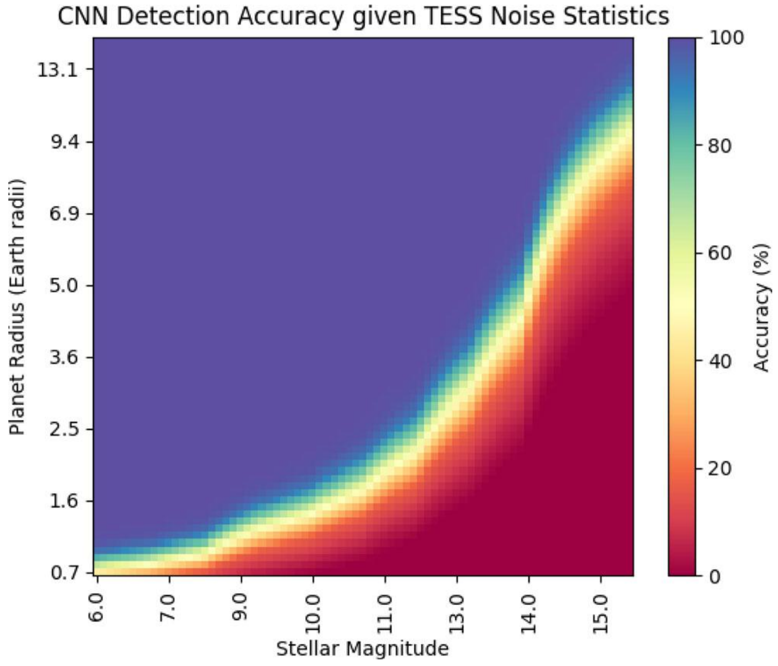
Pearson, Kyle A., Leon Palafox, and Caitlin A. Griffith. "Searching for exoplanets using artificial intelligence." Monthly Notices of the Royal Astronomical Society 474.1 (2017): 478-491.

Exoplanet detection



Exoplanet detection

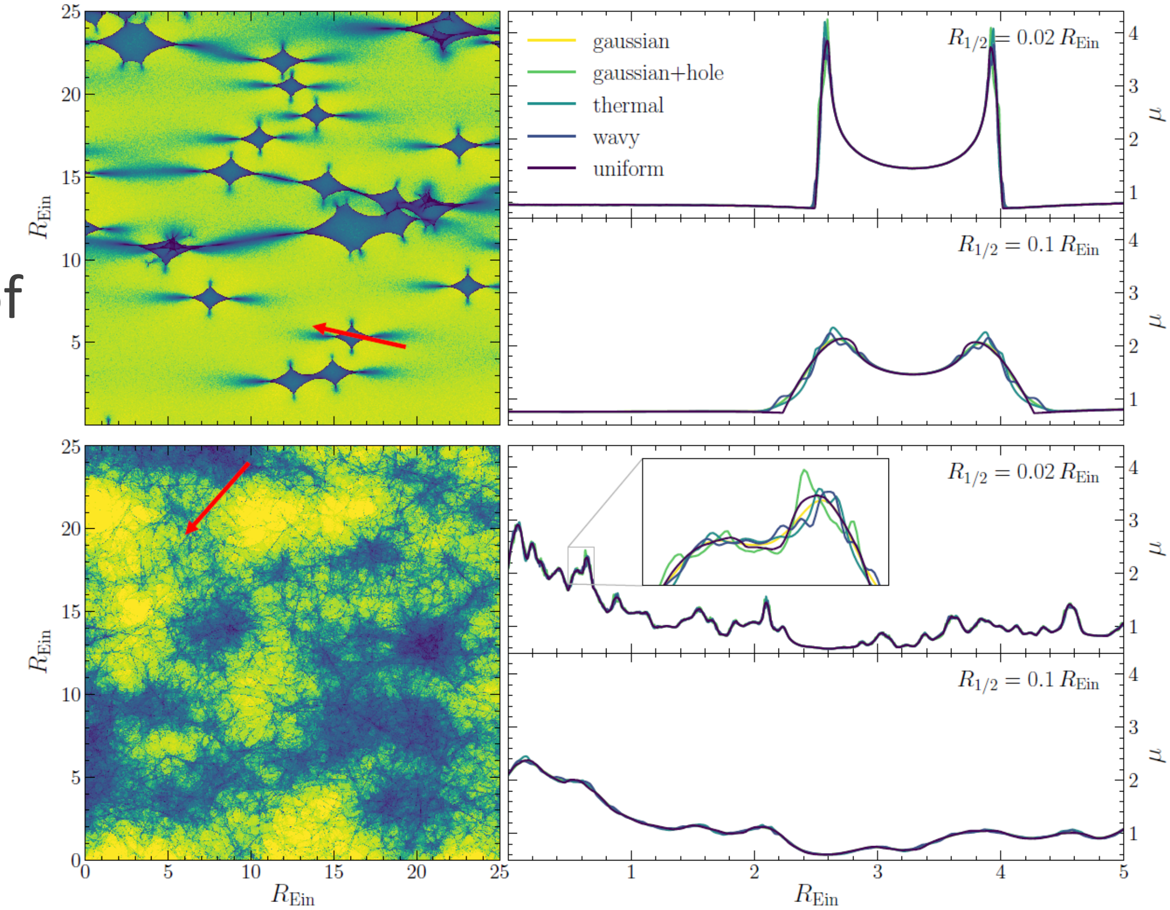




Pearson, Kyle A., Leon Palafox, and Caitlin A. Griffith. "Searching for exoplanets using artificial intelligence." Monthly Notices of the Royal Astronomical Society 474.1 (2017): 478-491.

Quasar micro-lensing light curves

- Use CNN to model quasar microlensing light curves and extract the size and temperature profile of the accretion disc.



G. Vernardos and G. Tsagkatakis "Quasar microlensing light curve analysis using deep machine learning", Monthly Notices of the Royal Astronomical Society, Vol. 486 (2), pp. 1944–1952, June 2019,

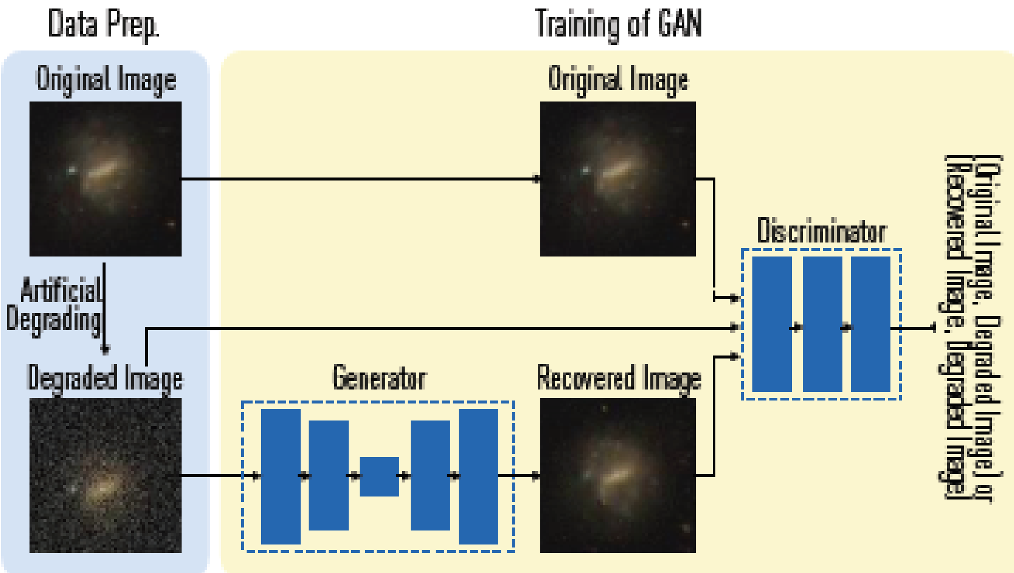
Other applications

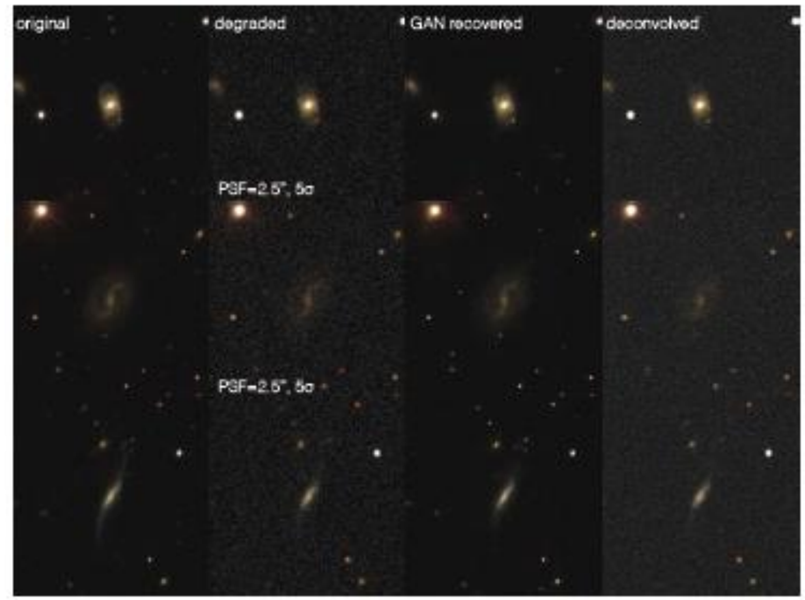
- Classifying Radio Galaxies With Convolutional Neural Network[1]
- Deep-HITS: Rotation Invariant Convolutional Neural Network For Transient Detection[2]
- Galaxy surface brightness profile[3]
- Gravitational wave detection[4]

1. Aniyan AK, Thorat K. Classifying Radio Galaxies with the Convolutional Neural Network. The Astrophysical Journal Supplement Series. 2017 Jun 13
2. Cabrera-Vives G, Reyes I, Förster F, Estévez PA, Maureira JC. Deep-HiTS: Rotation invariant convolutional neural network for transient detection. The Astrophysical Journal. 2017 Feb 10
3. Tuccillo D, Huertas-Company M, Decencière E, Velasco-Forero S, Domínguez Sánchez H, Dimauro P. Deep learning for galaxy surface brightness profile fitting. Monthly Notices of the Royal Astronomical Society. 2017 Dec 11
4. George, Daniel, and E. A. Huerta. "Deep Learning for real-time gravitational wave detection and parameter estimation: Results with Advanced LIGO data." *Physics Letters B* 778 (2018): 64-70.

GANs for deconvolution

4,550 SDSS images of nearby galaxies at 0:01 < z < 0:02





Schawinski, Kevin, et al. "Generative Adversarial Networks recover features in astrophysical images of galaxies beyond the deconvolution limit." *arXiv preprint arXiv:1702.00403* (2017).

TensorFlow

Deep learning library, open-sourced by Google (11/2015)

TensorFlow provides primitives for
- defining functions on tensors
- automatically computing their derivatives

What is a tensor

What is a computational graph

Material from lecture by Bharath Ramsundar, March 2018, Stanford

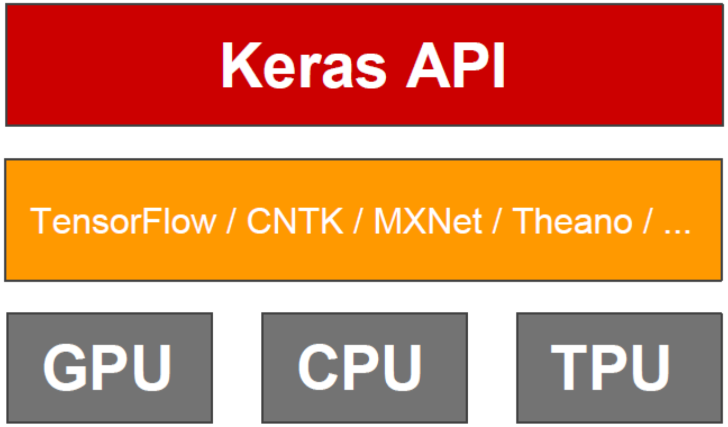
Introduction to Keras

Official high-level API of TensorFlow
- Python
- 250K developers
- Developed by Francois Chollet

Same front-end <-> Different back-ends
- TensorFlow (Google)
- CNTK (Microsoft)
- MXNet (Apache)
- Theano (RIP)

Hardware
- GPU (Nvidia)
- CPU (Intel/AMD)
- TPU (Google)

Companies: AWS, Uber, Google, Nvidia…



Keras models

Installation

- Anaconda -> Tensorflow -> Keras

Build-in

- Conv1D, Conv2D, Conv3D…
- MaxPooling1D, MaxPooling2D, MaxPooling3D…
- Dense, Activation, RNN…

The Sequential Model

- Very simple
- Single-input, Single-output, sequential layer stacks

The functional API

- Mix & Match
- Multi-input, multi-output, arbitrary static graph topologies

Sequential API

```
>> import keras
>> from keras import layers
>> model = Sequential()
>> model.add(layers.Dense(24, activation='relu', input_shape(10,)))
>> model.add(layers.Dense(20, activation=‘relu'))
>> model.add(layers.Dense(10, activation=‘softmax'))
>> model.fit(x_train, y_train, epochs=5, batch_size=32)
```

Functional API

```
>> import keras
>> from keras import layers
>> inputs = keras.Input(shape=(10,))
>> x = layers.Dense(20, activation='relu')(inputs)
>> x = layers.Dense(20, activation='relu')(x)
>> outputs = layers.Dense(10, activation='softmax')(x)
>> model = keras.Model(inputs, outputs)
>> model.fit(x_train, y_train, epochs=5, batch_size=32)
```

Sequential

```
>>from keras.models import Sequential

>>model = Sequential()

>> from keras.layers import Dense

>> model.add(Dense(units=64, activation='relu', input_dim=100))

>> model.add(Dense(units=10, activation='softmax'))

>> model.compile(loss='categorical_crossentropy',
optimizer='sgd',    metrics=['accuracy'])

>> model.fit(x_train, y_train, epochs=5, batch_size=32)

>> loss_and_metrics = model.evaluate(x_test, y_test,
batch_size=128)

>> classes = model.predict(x_test)
```

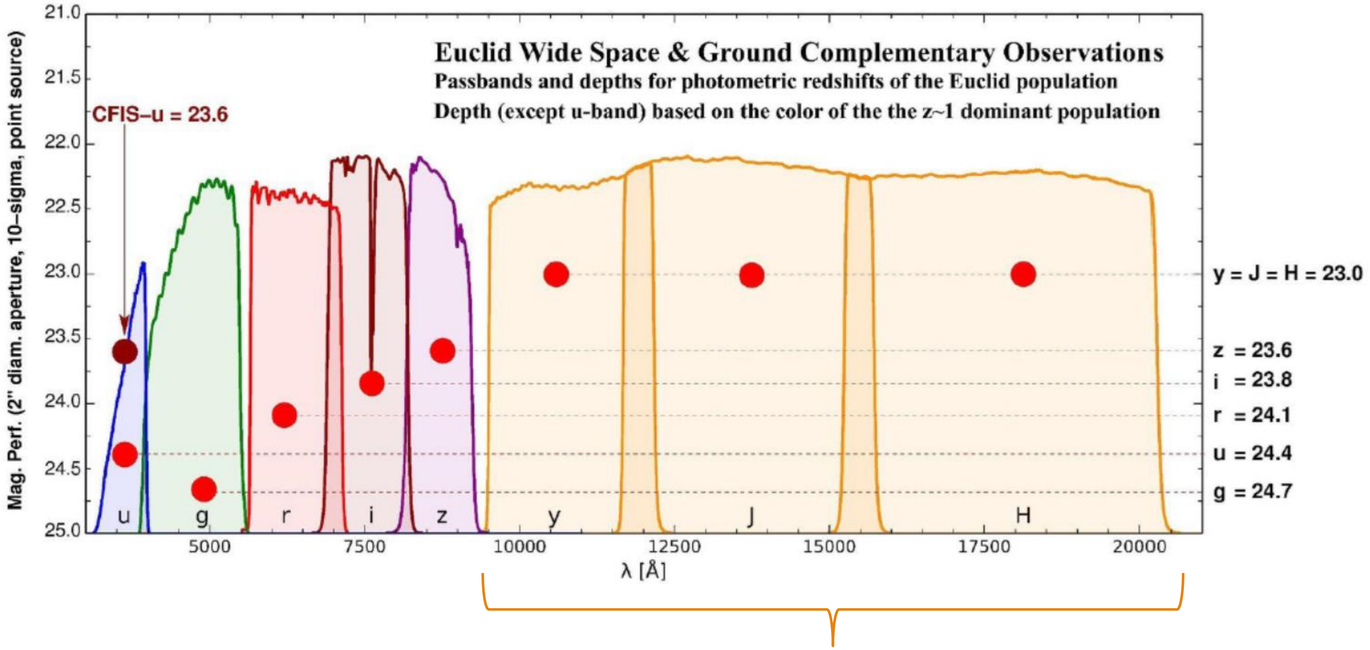
Functional

```
>> from keras.layers import Input, Dense
>> from keras.models import Model
>> inputs = Input(shape=(784,))
>> x = Dense(64, activation='relu')(inputs)
>> x = Dense(64, activation='relu')(x)
>> predictions = Dense(10, activation='softmax')(x)
>> model = Model(inputs=inputs, outputs=predictions)
>> model.compile(optimizer=‘SGD',loss='categorical_crossentropy',
metrics=['accuracy'])
>> model.fit(data, labels)
```

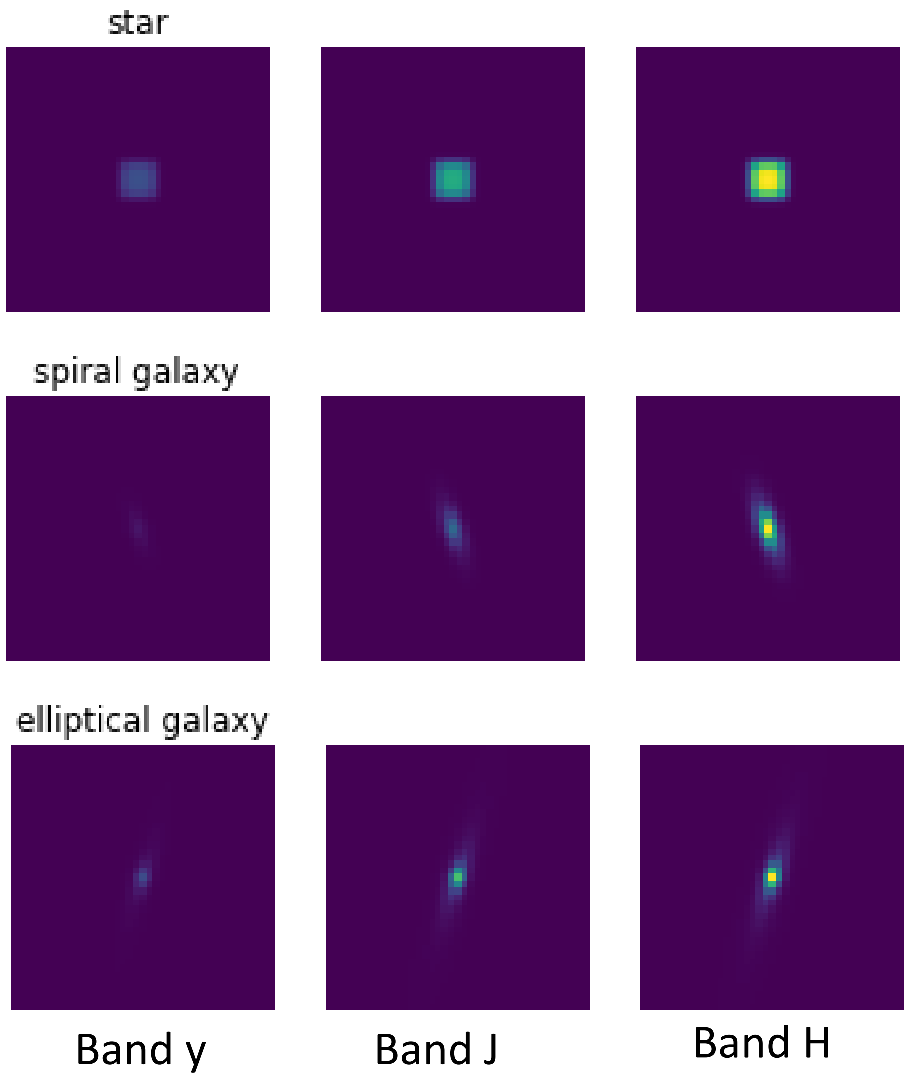
Case study #1: Galaxy morphology

Euclid mission

- Mission: map the dark universe
- Planned launch in 2022



Case study



Kaggle kernels

https://www.kaggle.com/greg1982/astroschool-morphology