Deep Learning Workshop

January 16th & 17th 2019

CompCancer

Agenda

1. Provide **theoretical** and **practical** hands-on knowledge on Deep Learning
2. Provide domain expert knowledge on **Deep-Learning interpretability**, **image-classification** and **'omics data analysis**

Aim

1. Improve/ establish programming skills
2. Gain theoretical insights into the keynote topics
3. Motivate your Deep-Learning-based publications

Context

CompCancer

- Today CompCancer PhD Graduate School
- First workshop offshoot Berlin Social Bioinformatics Meetings
- Next meeting January 29th

Bioinformatics Social Meetings Berlin

2018

Workshop Organizers

Manuela Bernary

- CompCancer organizer
- Blüthgen Lab
- *manuela.benary@cms.hu-berlin.de*

Teresa Domaszewska

- Bioinformatician
- Robert Koch Institute
- domaszewskat@rki.de

Raik Otto

- Bioinformatician
- AG Leser
- *raik.otto@hu-berlin.de*

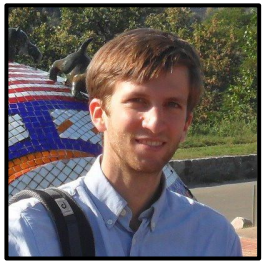
Schedule

Today

1. Session 1 - Introduction
   a. Programming basics
   b. Theoretical basics
2. Session 2- Interpretability of DL

Tomorrow

3. Session 3 - Image Analysis
4. Session 4 - Transcriptomics & Drug response



Keynote 1
Grégoire Montavon

Keynote 2
Dagmar Kainmüller

Keynote 3
Jonathan Ronen

Contributors & Supporters

Lisa Mais

Maryam Habibi

Timon Hick

Sven Giese

Peter Hirsch

Photography

We would like to take pictures.

Let us know if you would like to opt-out from being shown in pictures.

2018

CompCancer

Session 1
Programming Basics & Theory of Deep Learning

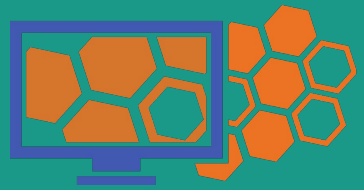
Aim Session 1

Programming

- Hardware & software requirements
- How to connect to and monitor GPU servers
- How to train Deep Learning models
- Empower participants to train models independently
- Essential terminology

Agenda of Programming session

1. Connecting to & monitoring of server
2. First DL model - Diabetes prediction
    a. Guided
    b. General theory of DL
    c. Explanation source code
3. Second model - Number classification
    a. Apply skills from 1 independently
    b. Basic concepts
4. Mini-Hackathon - Melanoma classification
    a. Real-world challenge
    b. Advanced concepts

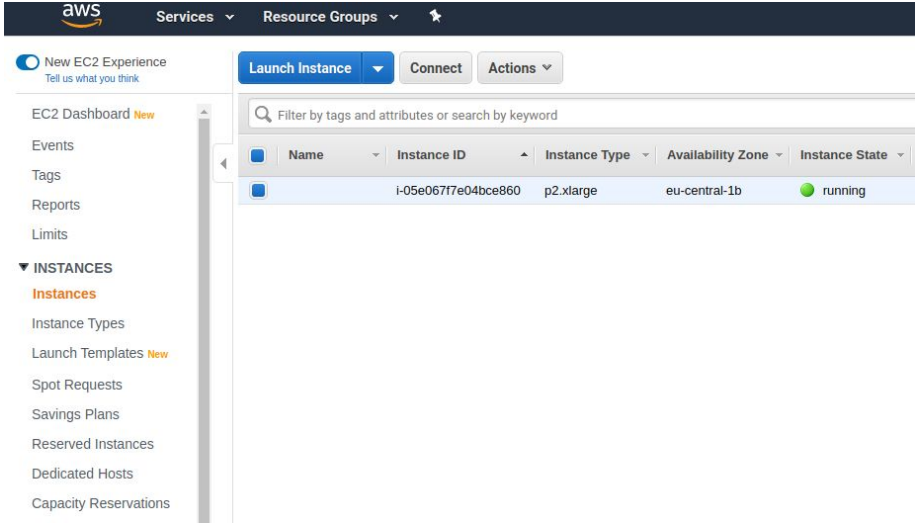


CompCancer

Connecting & Monitoring

Amazon Web Service GPU-Server Cloud

- Model creating (training) takes place on a GPU server
- Server has to have access to data
- Sudo rights!
- Amazon Web Services (AWS) sponsors the GPU servers for this workshop
- We connect to virtual cloud instances

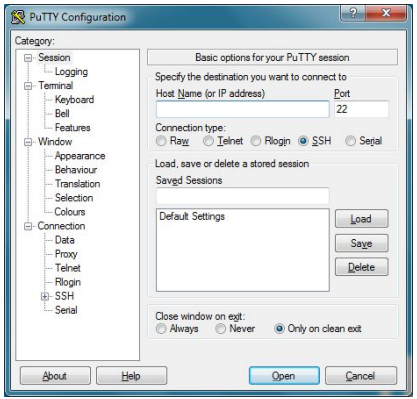


SSH and Port Forwarding

Secure SHell (SSH)

- Encrypted communication protocol
- Authenticates remote users and remote systems
- Commonly consists of public and private key

PuTTY Client



Port Forwarding

Ports are ‘gateways’ where software can communicate with another system’s software

- Avoids collisions of software communication
- Fixed ports for important software exists

Aim: Redirect the port of the DL programming software from the AWS Server to your laptop

Connecting to your server from POSIX systems

Updated!

1. Go to github.com/RaikOtto/CompCancer
2. Look up your IP **XYZ** -> number on post-it
3. Download key **.pem** file
4. Open terminal and *cd* to download folder
5. `> chmod 400 Workshop_SSH_Key.pem`
6. `> ssh -i "Workshop_SSH_Key.pem"  -Y -L 8889:localhost:8889 -L 6014:localhost:6014 XYZ -l ubuntu`

↓ Updated!

Connecting to your server from MS Windows

Updated!

1. Go to github.com/RaikOtto/CompCancer
2. Look up your IP **XYZ** -> number on post-it
3. Download key **.ppk** file
4. Start PuTTY
5. Hostname: **XYZ**
6. Auth - autologin username: ubuntu
7. Auth - private key file -> **.ppk** file
8. Tunnels
   a. Source port: 8889
   b. Destination: localhost:8889 (click '**Add**')
   c. Source port: 6014
   d. Destination: localhost:6014 (click '**Add**') ← Updated!
9. 'Open', accept button

Session control via Servlet

```
Screen version 4.00.03 (FAU) 23-Oct-06

Copyright (c) 1993-2002 Juergen Weigert, Michael Schroeder
Copyright (c) 1987 Oliver Laumann
```

- Setup server-sided application

- Facilitates simultaneously working in different screens

```
> screen + ‘space bar’ # new session
> Ctrl + a
> Ctrl + c # create new screen
> Ctrl + 1 # move back to screen 1
```

- Prevents scripts from terminating due to disconnects

```
> screen -rd # pickup disconnected session
```

NVIDIA-smi

Monitoring of the GPU

- Monitor GPU status

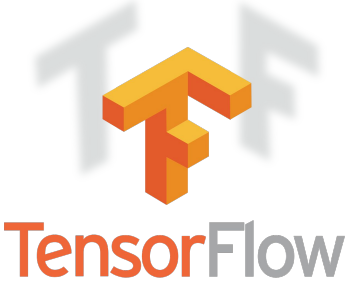
> `Ctrl + a`
> `2`
> `Watch -n 1 nvidia smi`

```
Every 1.0s: nvidia-smi

Sat Jan 11 15:08:04 2020
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 418.87.01    Driver Version: 418.87.01    CUDA Version: 10.1     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla K80           On   | 00000000:00:1E.0 Off |                    0 |
| N/A   54C    P0    74W / 149W |  11032MiB / 11441MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|    0     22440      C   ...aconda3/envs/tensorflow2_p36/bin/python 11019MiB |
+-----------------------------------------------------------------------------+
```

Tensorflow & CUDA

Tensorflow

- Differential programming library
- Trains Deep-Learning models
- Google open source software
- Based on Google-internal *DistBelieve*
- Developed by Geoffrey Hinton and Jeffrey Dean



CUDA

- Compute Unified Device Architecture (CUDA)
- Application programming interface (API)
- Requires correct CUDA Toolkit version

```
> nvcc --version
```

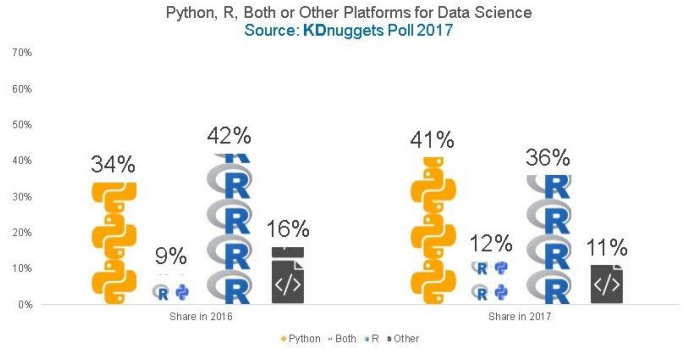
CUDA Toolkit	Linux x86_64 Driver Version
CUDA 10.2 (10.2.89)	>= 440.33
CUDA 10.1 (10.1.105)	>= 418.39
CUDA 10.0 (10.0.130)	>= 410.48
CUDA 9.2 (9.2.88)	>= 396.26

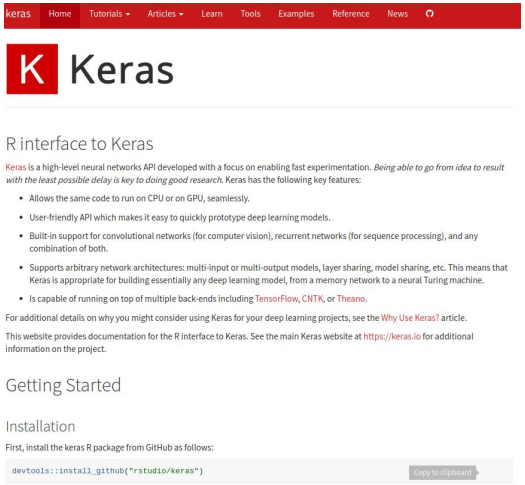
Keras

- High-level neural network API
- Focus on eager execution
- Supports arbitrary network architectures
- Built-in support for most common architectures
    - Convolutional networks
    - Recurrent networks etc.
- Supports different backends
    - TensorFlow
    - CNTK
    - Theano (deprecated)

Python versus R

- Python dominant over R for DL modeling
    - Other languages exist
- DL model three will cover DL in R





Preparation Tensorflow Python models

Testing the installed software

```
> python

> import tensorflow
```

```
ubuntu@ip-172-31-41-43:~$ python
Python 3.6.6 |Anaconda, Inc.| (default, Jun 28 2018, 17:14:51)
[GCC 7.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'tensorflow'
>>>
```

Tensorflow software required

Install software: BioConda & PIP

Large bioinformatics repository

```
> conda install -c anaconda tensorflow-gpu
```

Generic Python repository

```
> pip install tensorflow-gpu --user
```

Selecting the correct software environment

Software environment

- Different versions of the same software exist on the same system
- Activation of environment required

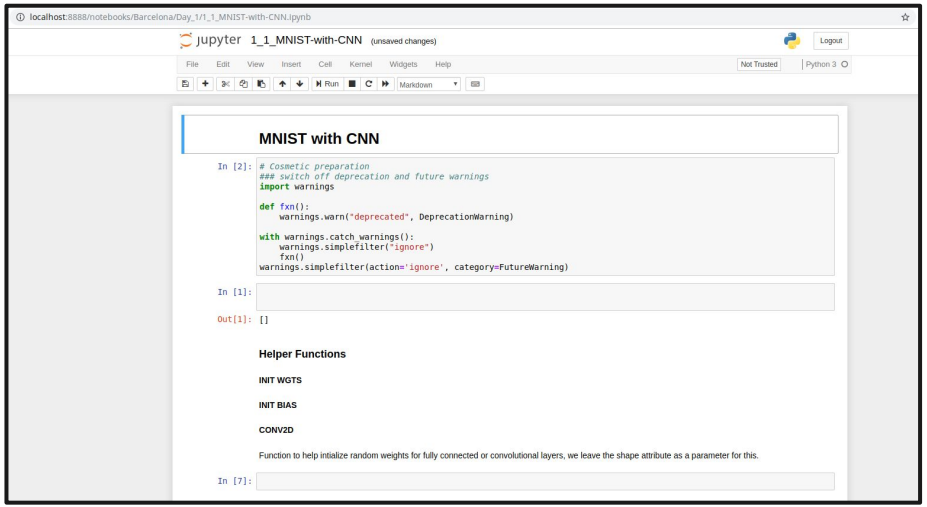
```
> conda env list
```

```
> conda activate tensorflow2_p36
```

```
ubuntu@ip-172-31-41-43:~$ conda activate tensorflow2_p36
(tensorflow2_p36) ubuntu@ip-172-31-41-43:~$ python
Python 3.6.5 |Anaconda, Inc.| (default, Apr 29 2018, 16:14:56)
[GCC 7.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow
>>>
```

Jupyter Server

- Programming Graphical User Interface (GUI)
- MS Word for Machine-Learner
- Based on IPython
- Uses ‘Notebooks’
    - Integrates source code and comments



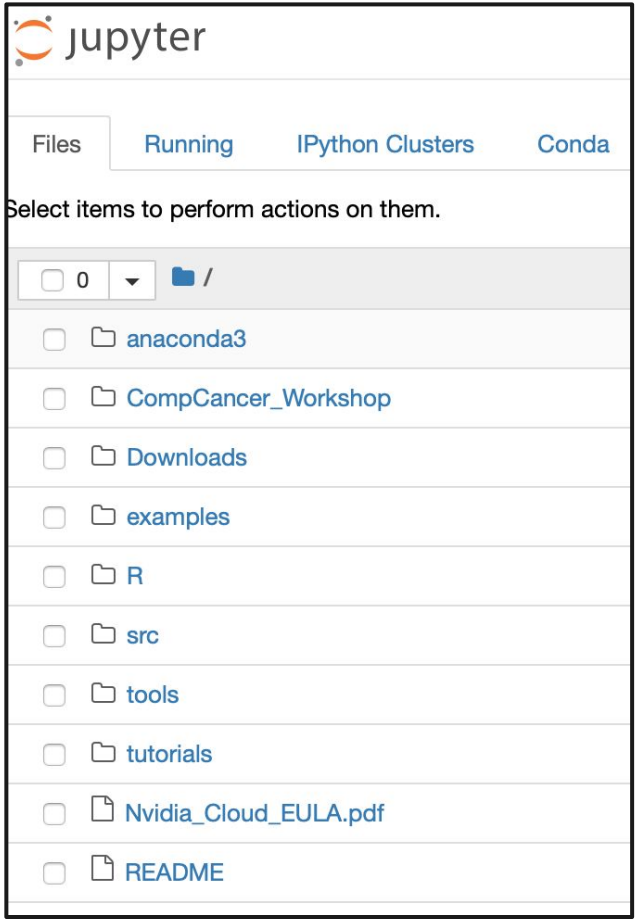
Jupyter notebook

- Text files with .ipynb ending
- JSON format

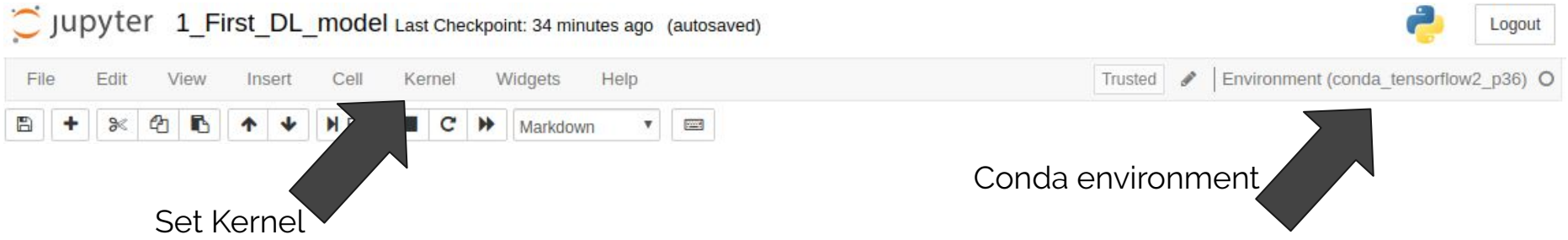
Start Jupyter server

```
> Ctrl + a
> 1
> jupyter-notebook --port=8889
```

- **Password: 'CompCancer'**
- Start server in servlet session to avoid stopping the server on logout



Interaction & Keyboard Shortcuts



- Kernels governs communication with GPU and software
- Cells either contain source code or Markdown
- Markdown ~= Website CSS

- Enter = enter cell
- Esc = leave cell
- Shift + enter = execute cell & move
- Ctrl + enter= execute cell & stay
- A insert cell above
- B insert cell above below
- M transform cell Markdown
- D + D delete

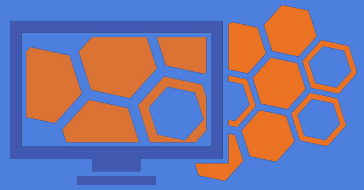
Mini-Hackathon

Hackathon Challenge

- Apply your learned Deep-Learning skills
- Real-world Stanford-created challenge
- Melanoma-classification
- Employs advanced concepts
- Announcement of winner at the end of Session 1

Melanoma-classification Hackathon

Credit Image: Nature Journal



CompCancer

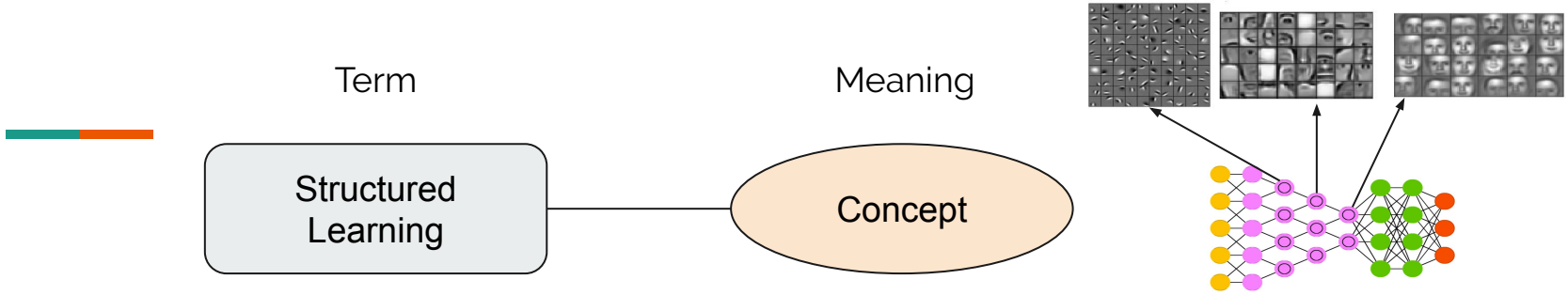
General Theory Deep-Learning

Actual name of Deep-Learning

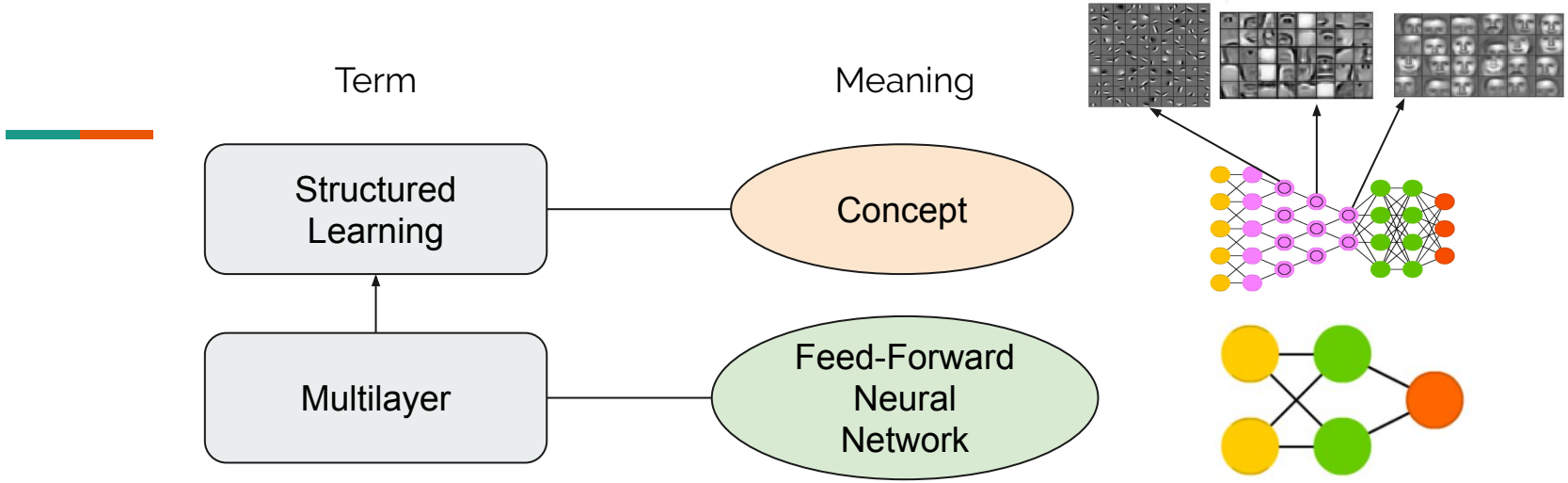
Structured Learning Feed-Forward Multilayer Perceptron Artificial Neural Network

(in general parlance)

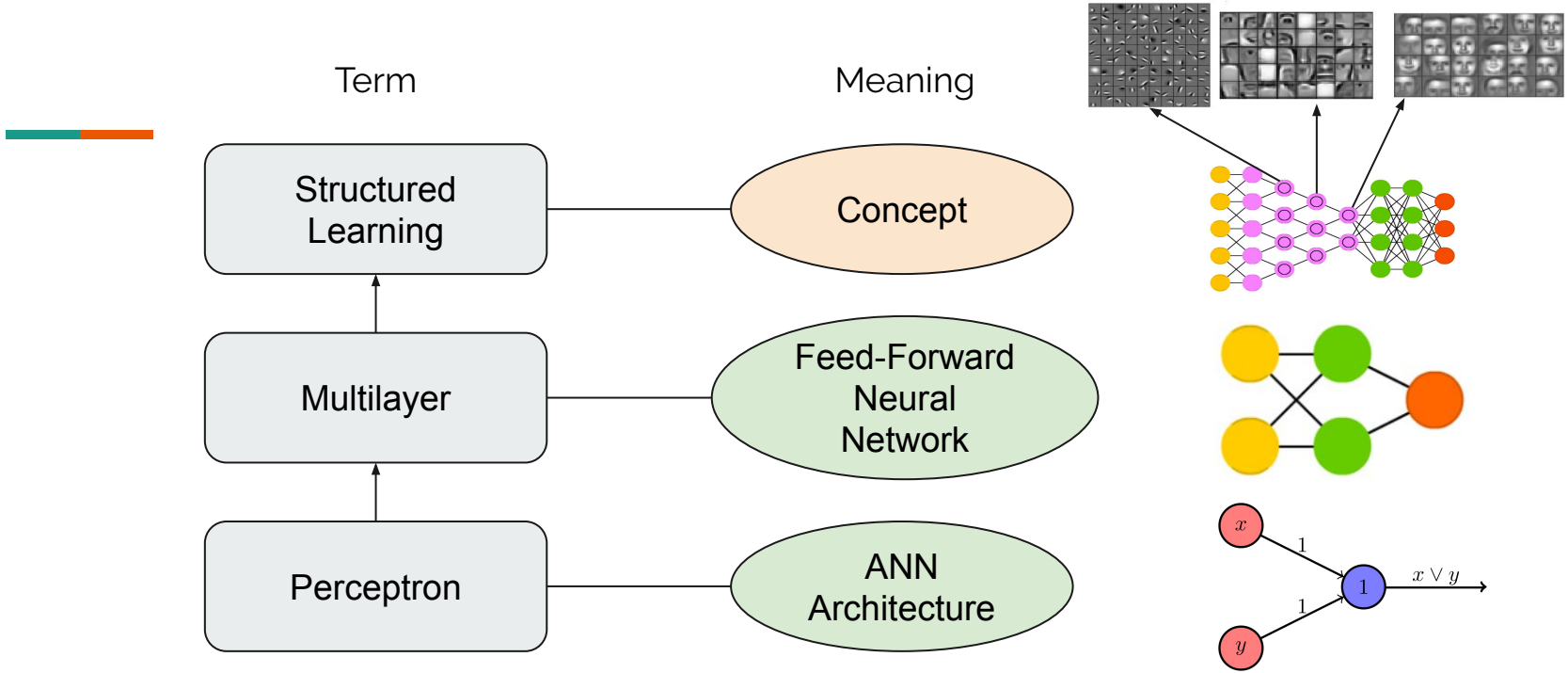
Structured Learning Multilayer Perceptron Network



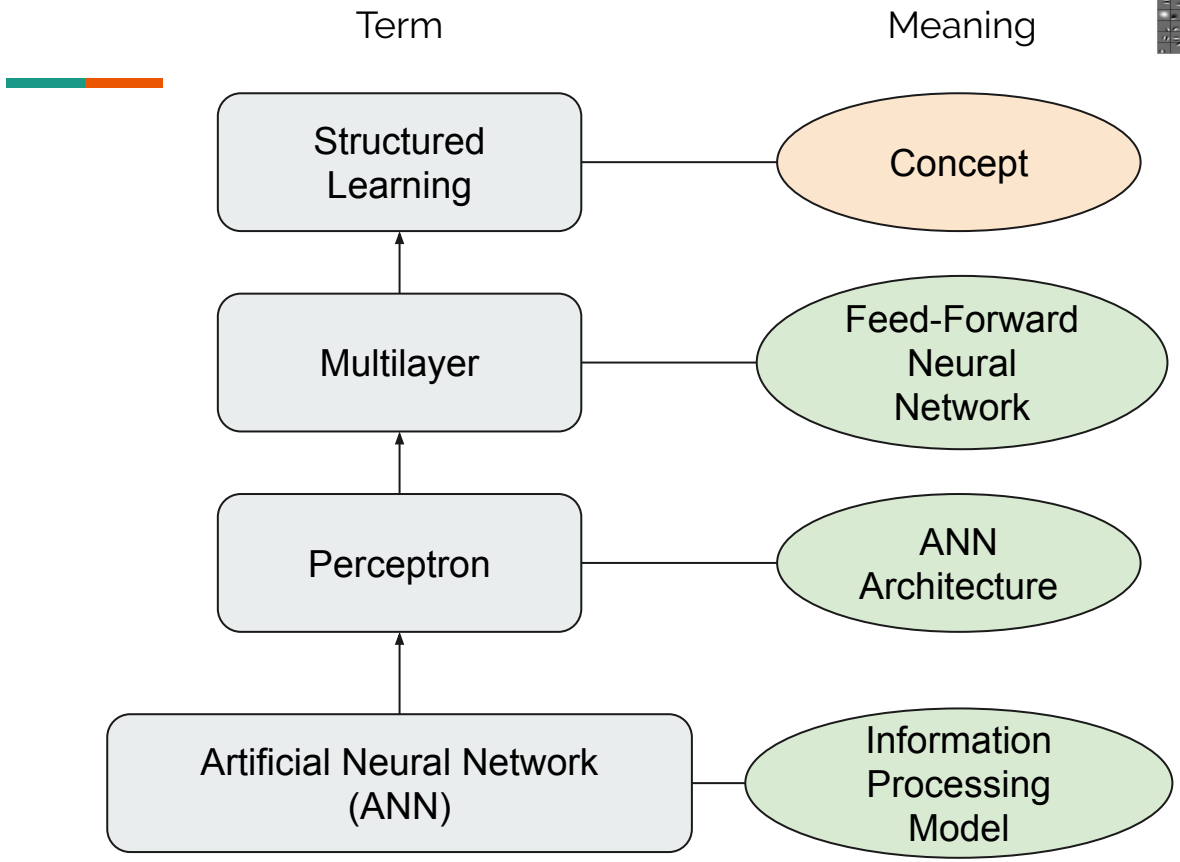
Structured Learning Multilayer Perceptron Network

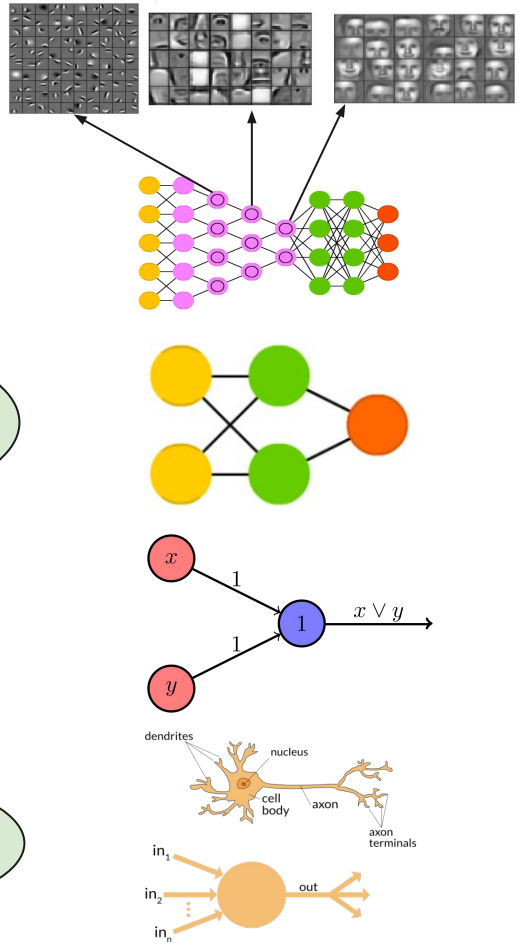


Structured Learning Multilayer Perceptron Network



Structured Learning Multilayer Perceptron Network

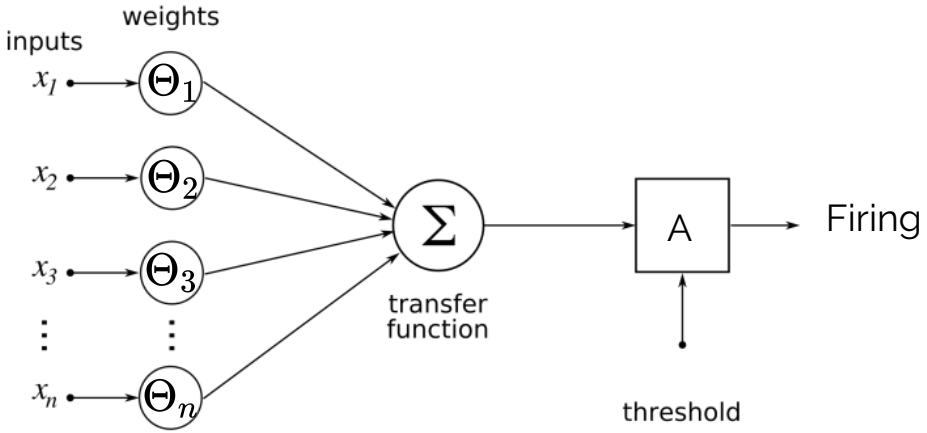




Artificial Neural Networks

Artificial Neural Network (ANN)

- Information processing model
- Set of **nodes** V connected by **edges** E
- Firing of node depends on **activation function** A
- Weights of edges Θ



Artificial Neuron

$$Firing = A(\sum(weights\ \Theta \cdot input\ x) + bias)$$

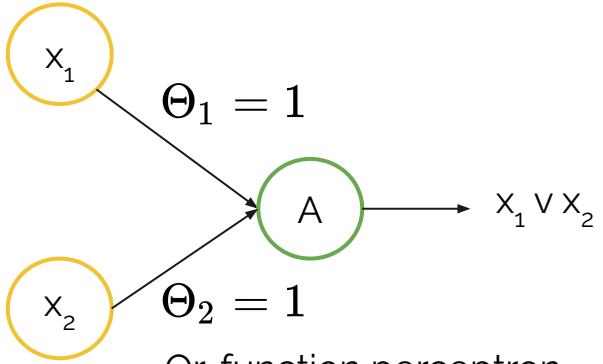
Credit Image: Perceptron. Mitchell, Machine Learning, p87.

Perceptron

Single-layer Perceptron

- Input layer labeled as ***x***
  - Binary
- No hidden layers
- Weight convergence ensured for one layer
  - Delta rule
- Prediction is matrix operation
  - (Historically) binary

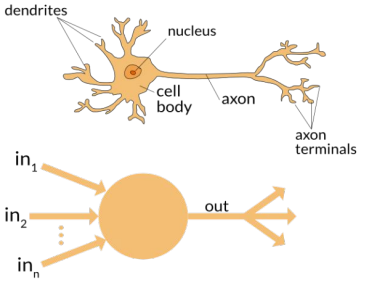
$$h_\Theta(x) = A(\Theta \cdot x)$$



Or-function perceptron

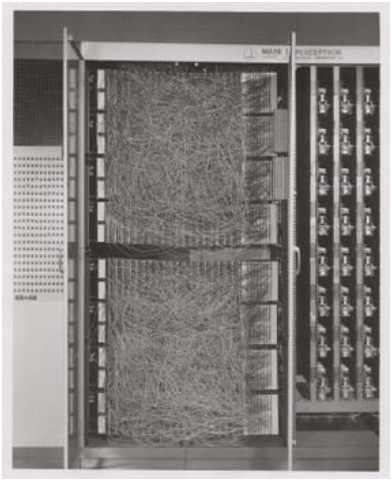
x_1	x_2	$h_\Theta(x)$
0	0	0
1	0	1
0	1	1
1	1	1

History of perceptron / Deep-Learning



- Loosely inspired by biological neural networks
- Developed by Frank Rosenblatt in 1958
  - US Navy
  - Classify images of numbers
- DL Winter
  - 1969, Minsky & Papert
  - XOR-function impossible for one-layer perceptron

Edge weight via potentiometer



'Perceptron Machine'

Motivation Machine-Learning

- Hypothesis

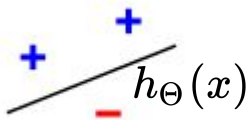
$$h_\Theta(x) = \Theta_0 + \Theta_1 \cdot x$$

- Parameters

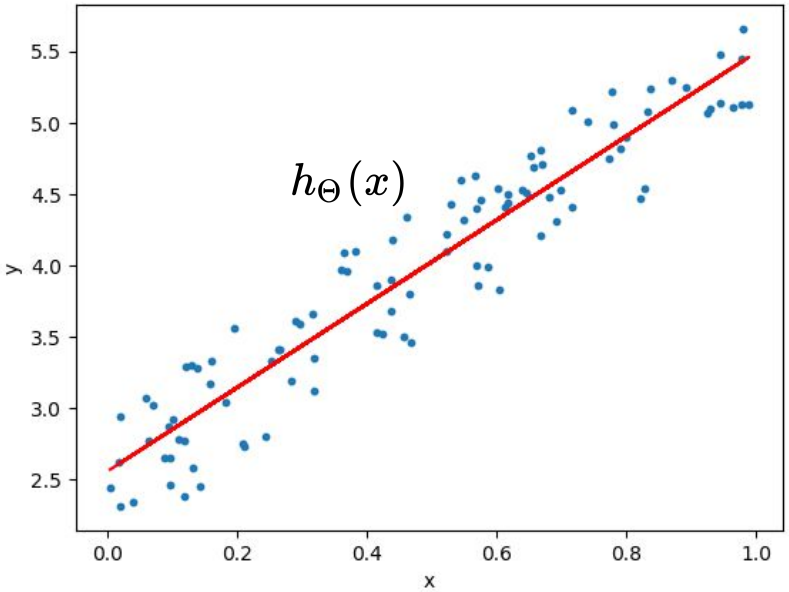
$$\Theta_0, \Theta_1$$

- Cost Function

$$J(\Theta_0, \Theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\Theta(x_i) - y_i)^2$$



Classification

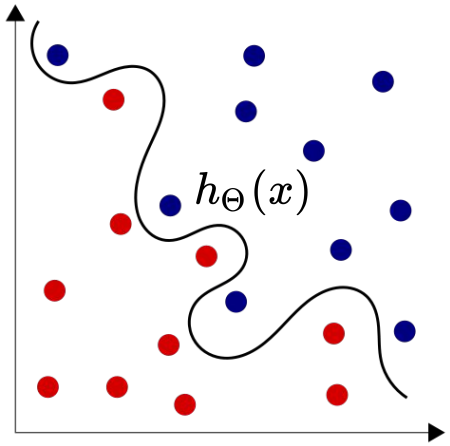


Regression

Credit Image: Gilbert Tanner

Non-linearity

- Non-linearity defies linear hypothesis
- Addition of non-linear features x^2
- (Generally) require complex hypothesis functions
- Utilization of 'Kernel-trick'
  - Projection into higher-dimension
  - Requires fine-tuning of model and/or domain knowledge



Non linear separability

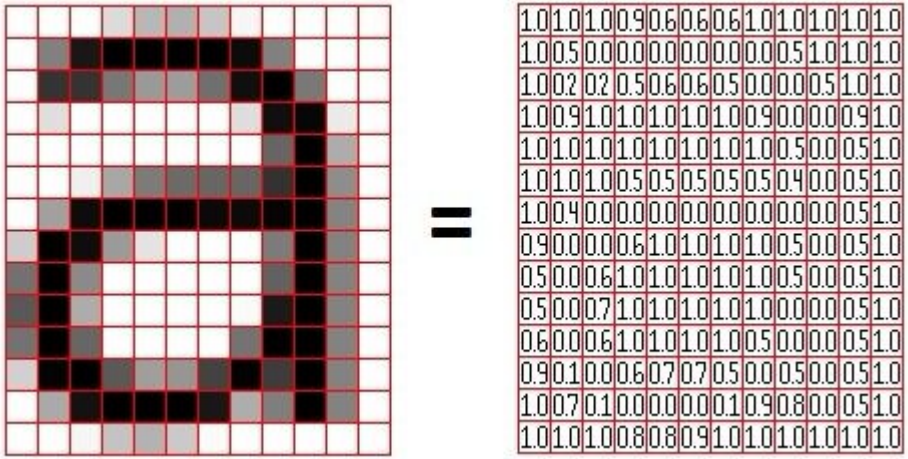
$$h_\Theta(x_1, x_2) = (x_1, x_2, x_1^2 + x_2^2)$$

Non-linear hypothesis function $h_\Theta(x)$

Credit Image: Wikipedia, Mithrandir Mage

Complex hypothesis: Combinatorial Explosion

- Complex problem require complex pattern recognition
  - Inefficient or infeasible
- ‘Classical’ ML models challenged by input dimension



Example: 50 x 50 pixel image

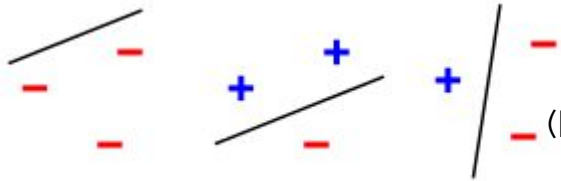
$$h_\Theta(x_1, x_2) = (x_1, x_2, x_1 \cdot x_2)$$

```
> choose(2500,2)
[1] 3123750
```

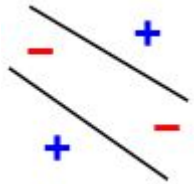
Vapnik–Chervonenkis Dimension

Vapnik–Chervonenkis Dimension (VC)

- ‘Measure of complexity’
- Upper error bound calculation possible
- **Cardinality** of the largest shatterable set of points



VC 3 - Straight line
(Linear classification)

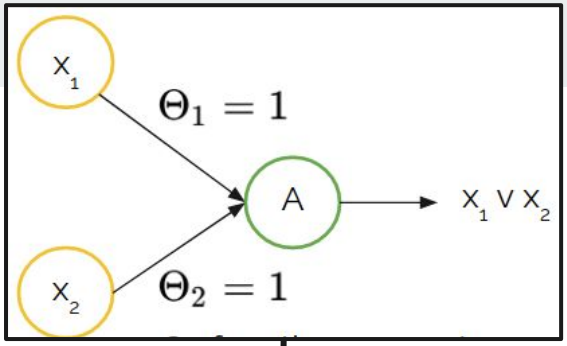


VC 4 - Two
Straight lines

#Variables	#Boolean Function	#BF Linearly separable
2	16	14
3	256	104
4	65536	1882

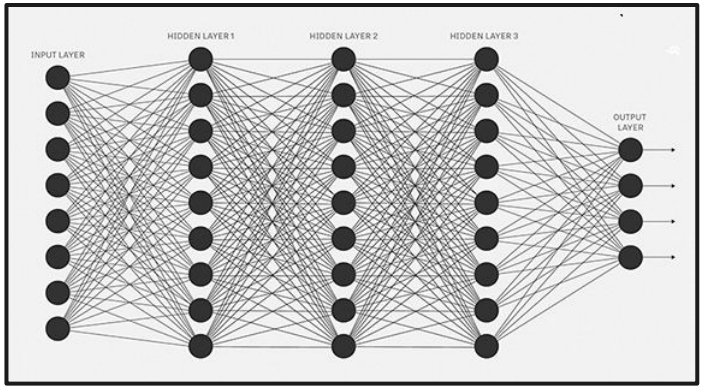
Credit Image: Wikipedia, Mithrandir Mage

Multilayer Perceptron



$$h_\Theta(x) = A(\Theta \cdot x)$$

- VC dimension dynamic
- Deep-Learning architecture parameters
  - |E| number edges
  - |V| number nodes
- Computationally expensive



Increased number of layers

VC Sign-Activation Function $O(|E| \cdot log(|E|))$

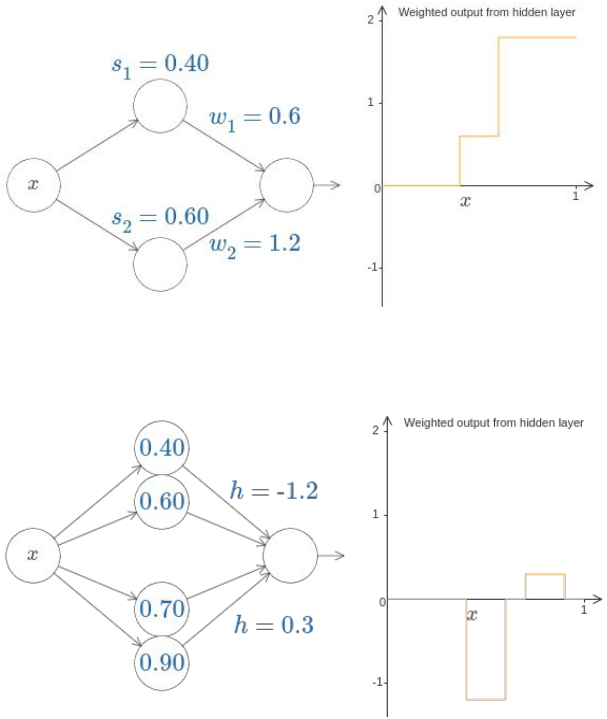
VC Sigmoid-Activation Function $O(|E|^2 \cdot |V|^2)$

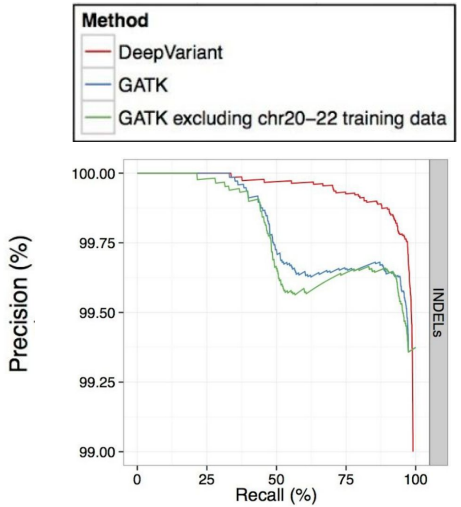
$$h_\Theta(x) = A_k(\Theta_{k-1}\cdot, \ldots, A_2(\Theta_1 \cdot x))$$

Credit Image: Perceptron. Mitchell, Machine Learning, p87.

Universal Function Approximation

- Approximation of any Lebesgue-integrable function
- $h_\Theta(x)$ can be latent
- Google's DeepMind beat GATK by learning a latent error function
- Error function (almost) intractable
  - Approximated by GATK with mixed gaussian models



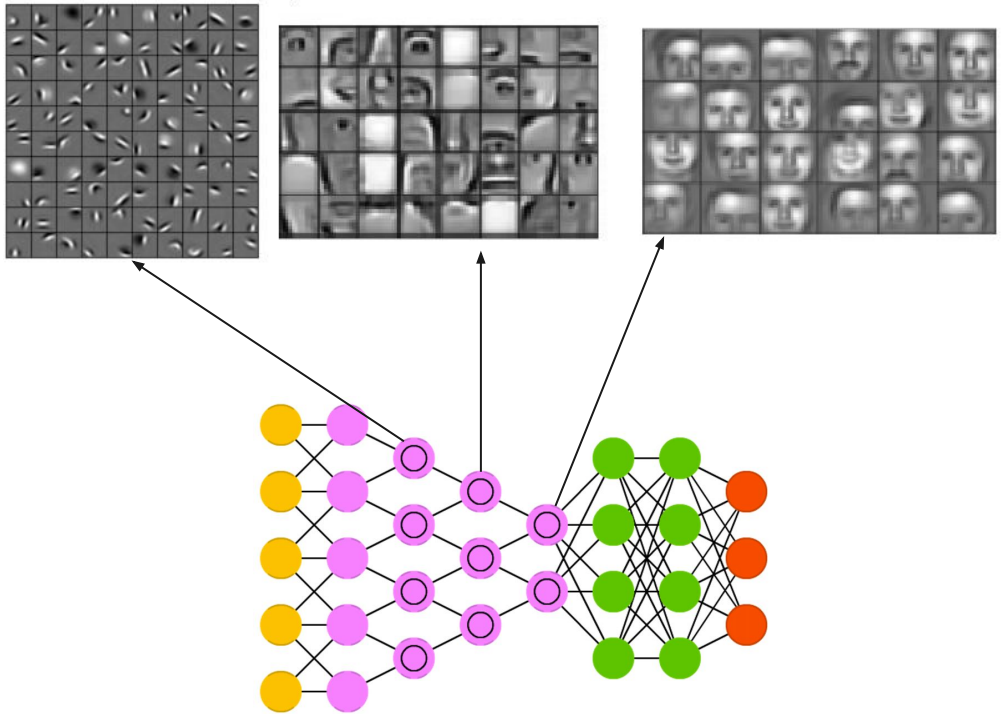


DL beats GATK gold-standard

Credit Images: Google 2018, Michael Nielsen

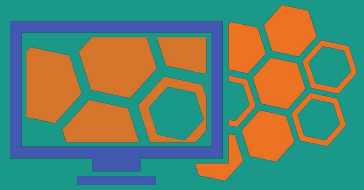
Definition Structured Learning

- Abstract concept of how to learn
- Each layer learns different level of abstraction
- Independent from ANNs
- Automatic feature selection
- Analogy to PCA: layers are an analogy to eigenvectors of matrix



Different degrees of abstractions in different layers

Credit Image: NVIDIA



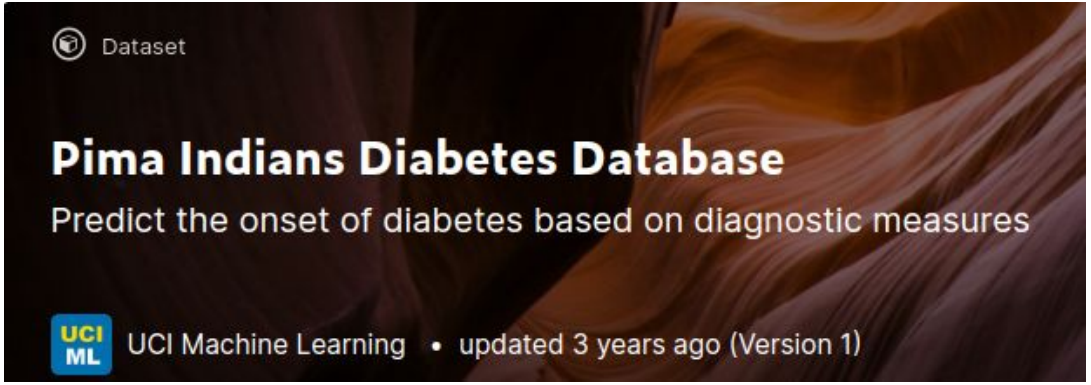
CompCancer

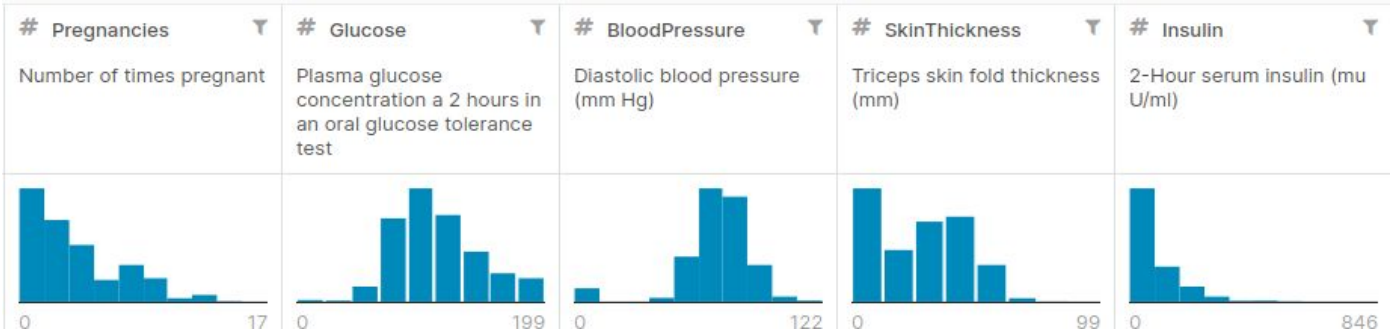
First Deep Learning model

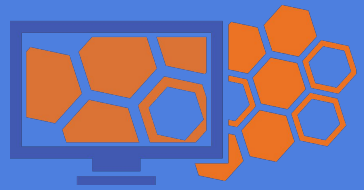
Diabetes prediction

Deep-Learning model 1

- Predict Diabetes status of Pima Indians
- Numeric matrix
  - Columns 1-8: data
  - Column 9: Phenotype



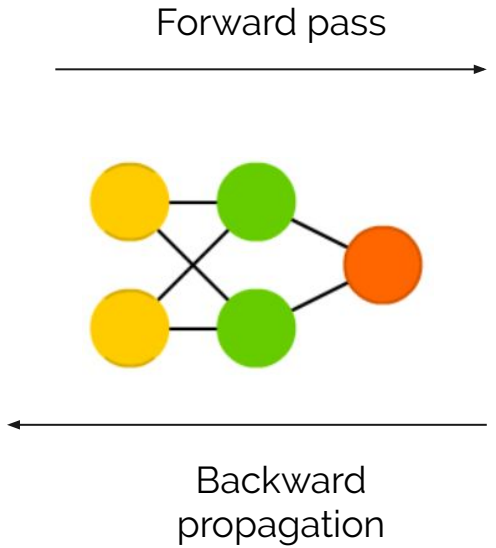




CompCancer

Theory Model 1

Feed Forward Artificial Neural Network



Feed Forward Artificial Neural Network (FFNN)

- Propagate an input signal through network
  - From **input** to **output** layer with intermediate **hidden** layers
- Prediction by forward pass
- Learn edges during backward propagation

Output network = $f_{act}(\Sigma\ (a_i \text{ final layer}))$

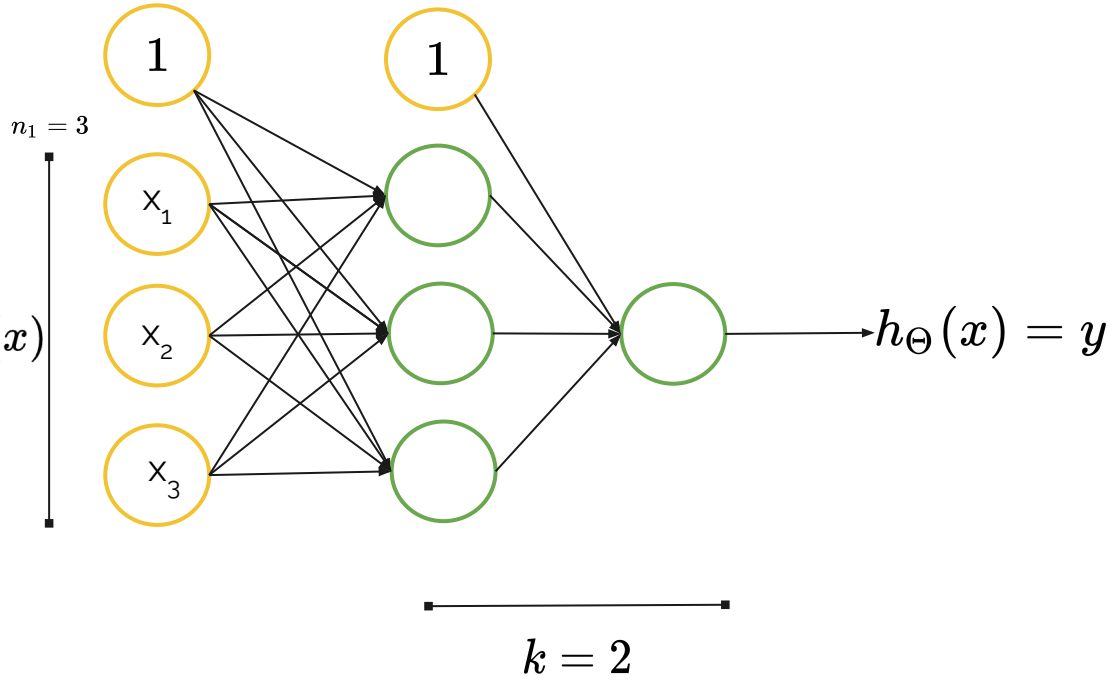
Input Cell Output Cell

Hidden Cell

Fjodor van Veen & Stefan Leijnen

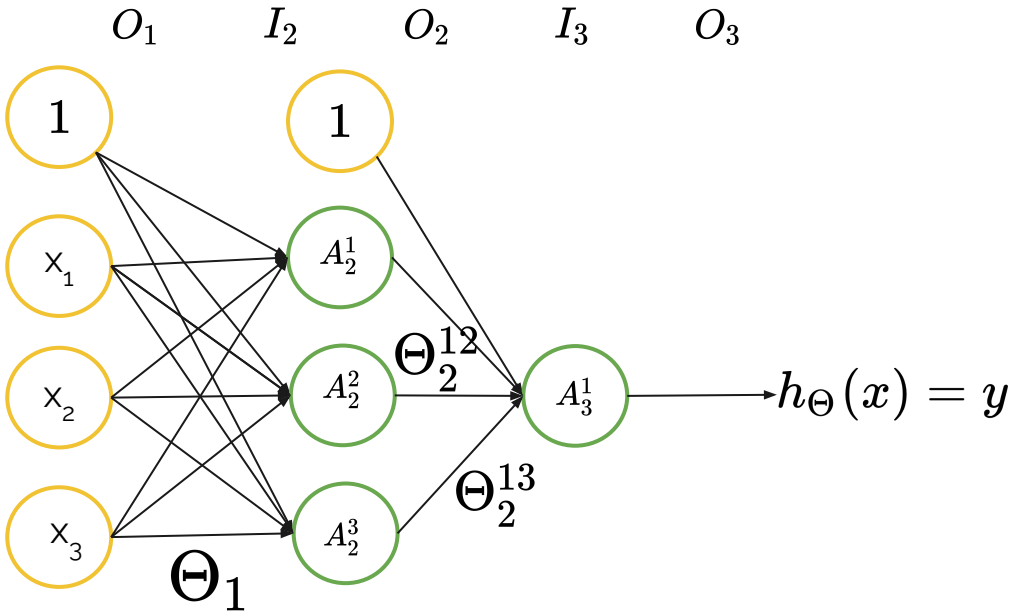
Parameters FFNN Network

- Input $x = [x_1, x_2, \ldots, x_n]$
- Output $y = [y_1, y_2, \ldots, y_n] = h_\Theta(x)$
- Number layers (depths) k
- Width n



Parameters FFNN Layer

- Input layer I_i
- Output layer O_i
- Activation function layer A_i
- Weights edges $\Theta_{layer}^{to\ from}$

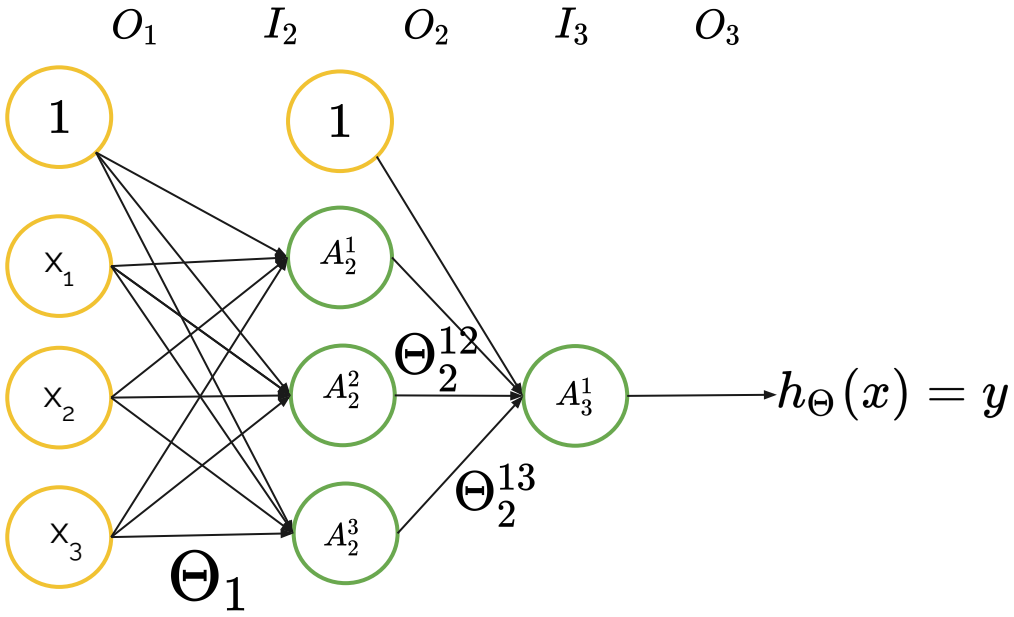


#weights fully connected layer: $a_n \cdot (a_{n-1} + 1)$

Feed Forward Pass

- Pass input through network
- From **input** to **output** layer
- *Forward pass == Forward propagation*

$$
\begin{aligned}
O_1 &= x \\
I_2 &= \Theta_1 \cdot O_1 \\
O_2 &= A(I_2)\ (add\ O_0^1 / x_0) \\
I_3 &= \Theta_2 \cdot O_2 \\
O_3 &= A(I_3)\ (add\ O_0^2) \\
h_\Theta(x) &= O_3
\end{aligned}
$$

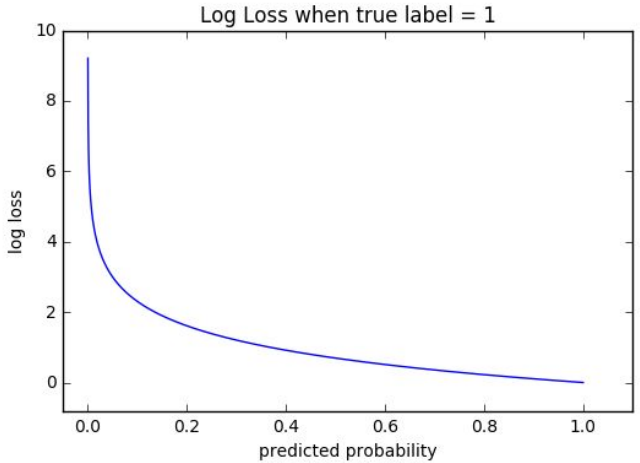


$$h_\Theta(x) = A_3(\Theta_2 \cdot A_2(\Theta_1 \cdot x))$$

Recursive formulation
Biases nodes omitted

Cost / loss function

- Quantification of difference prediction to truth y
- Partial derivatives required
- Cross-entropy function for classification
- RMSE for regression



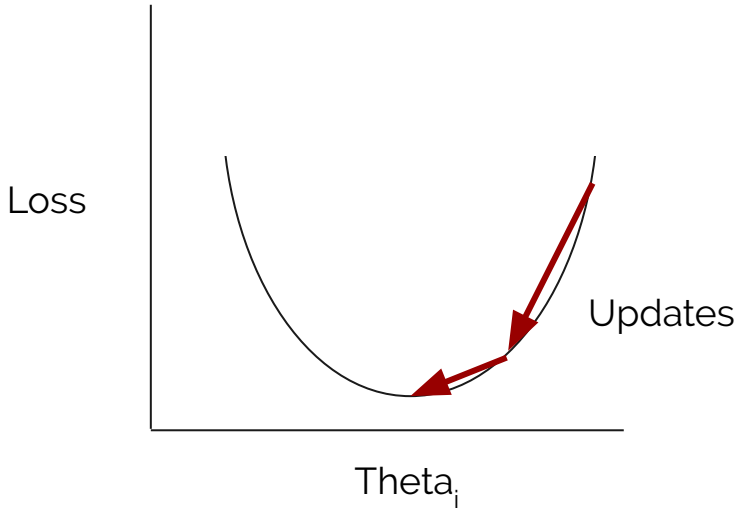
Error Quantification

$$J_\Theta(x,y) = -\frac{1}{m}\left[\sum_{i=1}^{m} y_i \cdot log(h_\Theta(x_i)) + (1-y_i) \cdot log(1-(h_\Theta(x_i))\right]$$

Credit Image: ml-cheatsheet

Gradient descent

- Calculate minimum of J
- Not necessarily global optimum
- Optimizers
  - Adam, Adadelta, RMS Prop
  - Global optimum not ensured



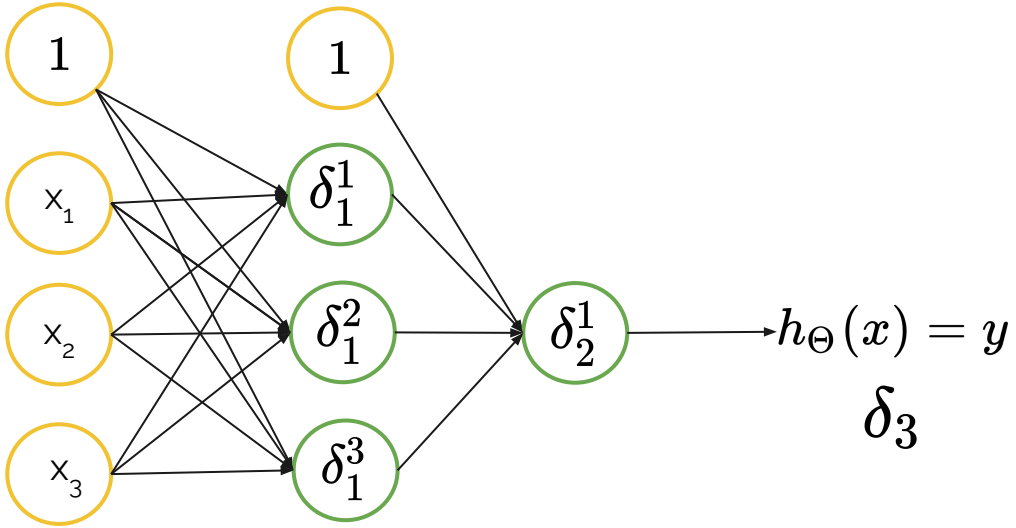
$$J_\Theta(A_3(\Theta_2 \cdot A_2(\Theta_1 \cdot x)), y) = J(x, y, \Theta) = J(x, y, \Theta_k, \Theta_{k-1}, \ldots, \Theta_1)$$

Credit Image: ml-cheatsheet

Intuition Backpropagation

- Error for node *j* in layer *i* δ_i^j
- Quantify error for every node
- Calculate partial derivatives — $\frac{\partial}{\partial \Theta} J_\Theta(x)$
- Update weights by error gradient

  with learning rate



$$\delta_3 = J_\Theta(x) = y_{truth} - A_3(I_3)$$
$$\delta_2 = \Theta_2 \cdot \delta_3 \cdot A_2'(I_2)$$
$$\delta_1 = \Theta_1 \cdot \delta_2 \cdot x$$

Total derivative shown

Outline Mathematics Backpropagation

(1) $J_\Theta(A_3(\Theta_2 \cdot A_2(\Theta_1 \cdot x)), y)$

(2) $\frac{\delta J_\Theta}{\delta x} = \frac{\delta J_\Theta}{\delta O_3} \cdot \frac{\delta O_3}{\delta I_2} \cdot \frac{\delta I_2}{\delta O_1} \cdot \frac{\partial O_1}{\partial x}$

(3) $\frac{\delta J_\Theta}{\delta A_3} \cdot (A_3)' \cdot \Theta_2 \cdot (A_2)' \cdot \Theta_1$

$\dots$

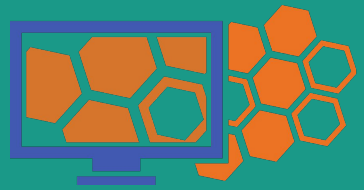
Chain rule

$$\frac{\delta a}{\delta c} = \frac{\delta a}{\delta b} \cdot \frac{\delta b}{\delta c}$$

$$(f(g(x)))' = f'(g(x)) \cdot g'(x)$$

(4) $\delta_i := (A_i)' \cdot (\Theta_{i+1})^T \cdot, \dots, \cdot (\Theta_{k-1})^T \cdot (A_{k-1})' \cdot (\Theta_k) \cdot (A_k)' \nabla_{O_k} J_\Theta(x)$

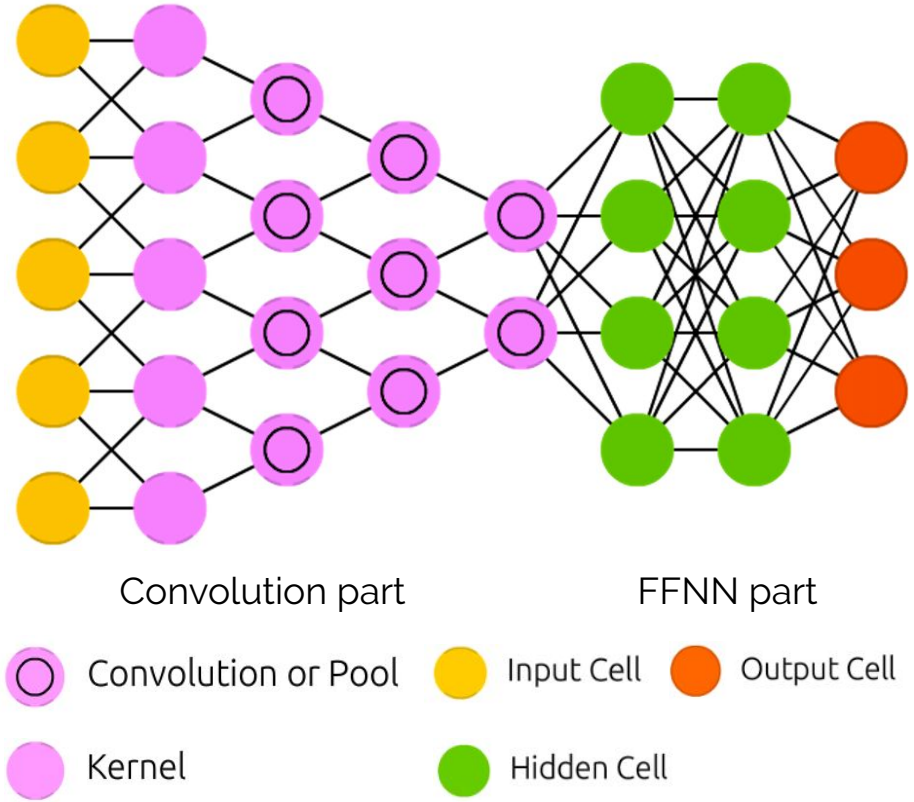
(5) $\nabla_{\Theta_i} J_\Theta(x) = \delta_i (O_{i-1})^T$



Second model
Convolutional Neural Network
Number recognition

Convolutional Neural Networks

- Image classification
- **Convolve** a pixels into decision units
  - 200 x 200 pixels -> 40 000 input units -> e.g. 20 decision units
- Project abstract patterns into higher and smaller layers
- Optional combination with FFNN for non-linear behavior

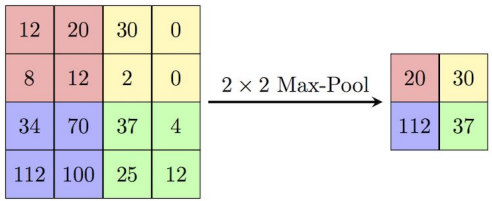


Fjodor van Veen & Stefan Leijnen

Max Pooling & Striding & Padding

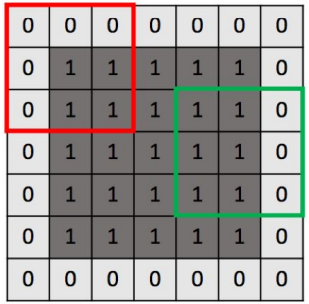
(Max)-Pooling

- Kernel filter function
- Reduce features
- Induces abstract pattern recognition



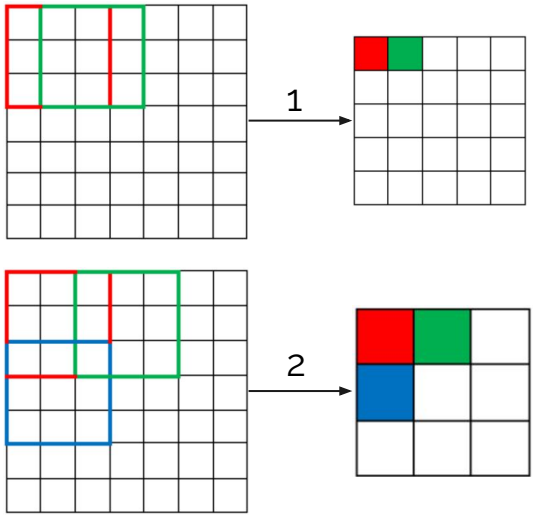
Padding

- Addition to outer pixels
- Ensure desired kernel coverage of pixels



Striding

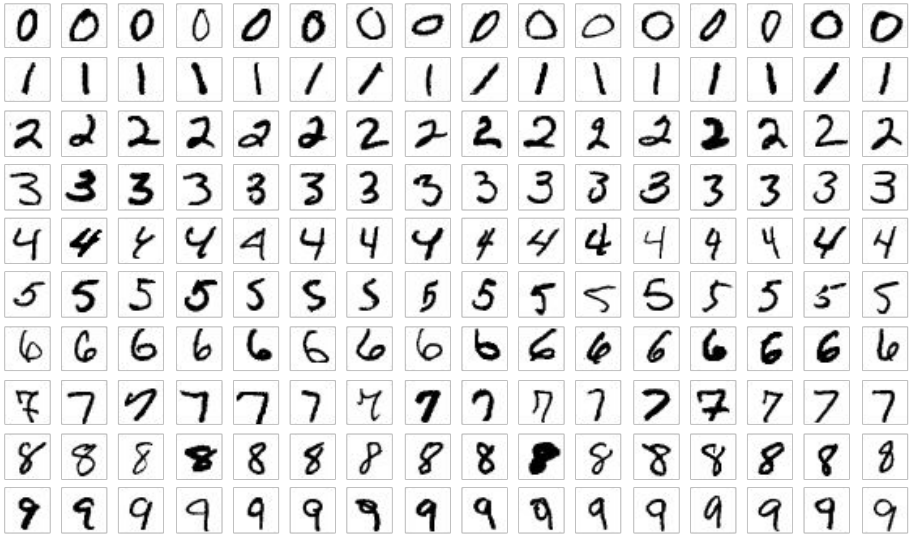
- Movement control of filter kernel
- Amount of skipped pixels



Credit Image: rMachineLearning reddit, Adit Deshpande

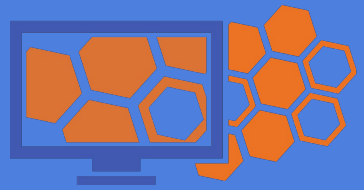
MNIST dataset

- Very well known tutorial dataset
- Used to be real-world challenge before being ‘solved’
- 60,000 samples
- 28x28 pixels
- You will solve MNIST via DL convolutional network



Hand-written number classification

Credit Image: Josef Steppan

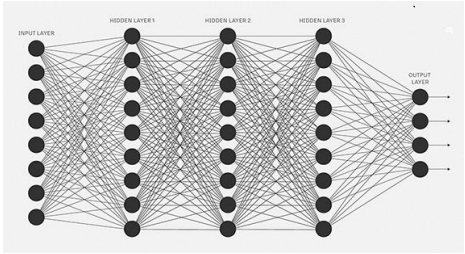


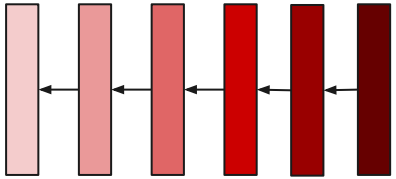
CompCancer

Theory model 2

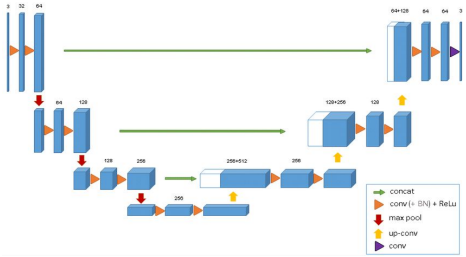
Vanishing gradient problem

- Partial derivatives decay through layers
- Slows down or stops model training
  - Contributed to AI winter
- Remedies:
  - More computational power
  - ANN architecture adaptation
  - Residual ANNs





Decay of partial derivative $-\frac{\partial}{\partial\Theta}J_\Theta(x)$



Note high-level edges

Credit Image: Google, Perceptron. Mitchell, Machine Learning, p87.

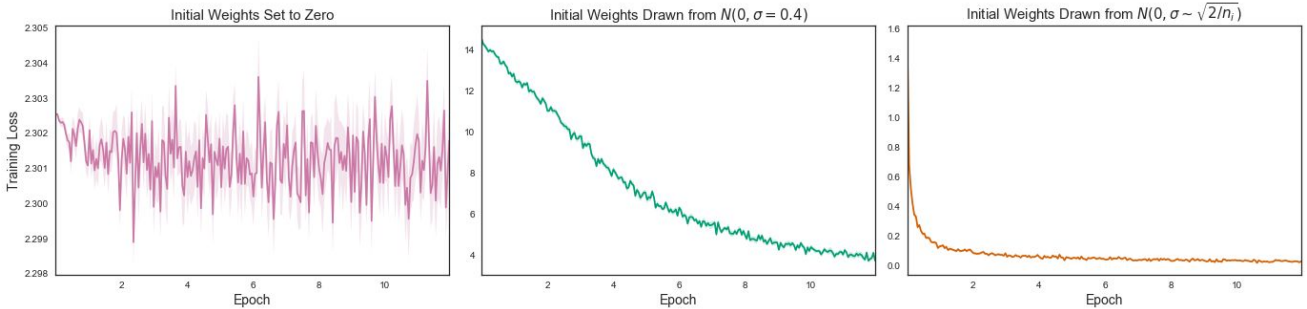
Weight initialization

Properties

- Critical for effective model training
- Each edge has weight parameter
- Avoiding symmetries necessary

Initialization types

- [0,..,1], $\mathbb{R}$, uniform
- N(μ,σ)
- Orthogonal
  - Weights drawn from orthogonal matrix
- All zeros
- All ones

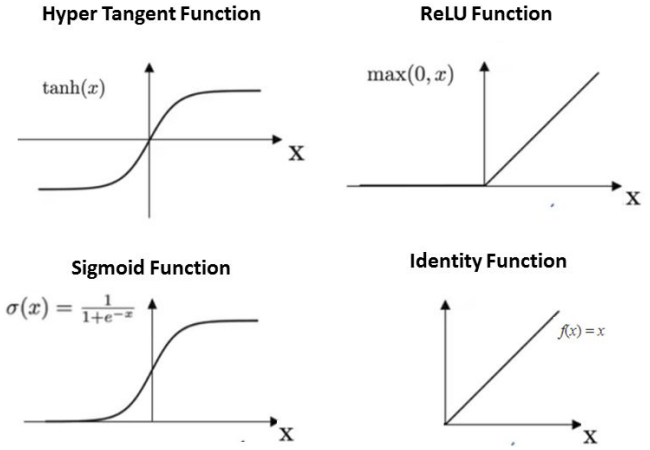


Impact weight initialization on loss

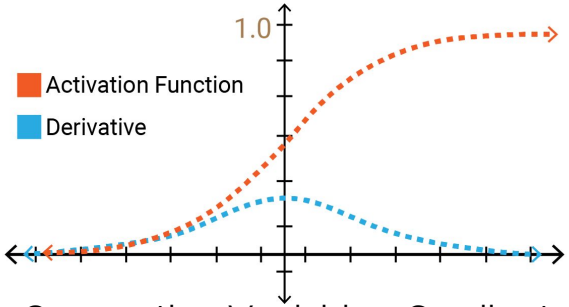
Credit Image: Perunicic et al. 2018

Activation functions

- Key for non-linearity
- Influences multiple behaviors
  - Vanishing gradient
  - Prediction performance
- Frequently utilized functions
  - Hyper Tangent -> [-1,..,1]
  - Sigmoid -> [0,...,1]
- Last layer
  - Binary - Logistic
  - Multinomial - Softmax



Types of activation functions

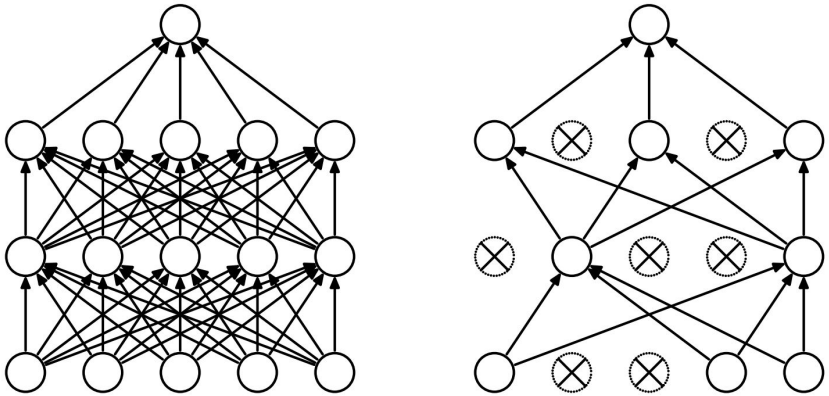


Connection Vanishing Gradient

Credit Image: Animesh Agarwal

Dropout layer

- Avoids overfitting
- Enforce redundancy
- Increases training time
- Can be problematic for Convolutional Neural Networks (CCN) - Model 2



(a) Standard Neural Net (b) After applying dropout.

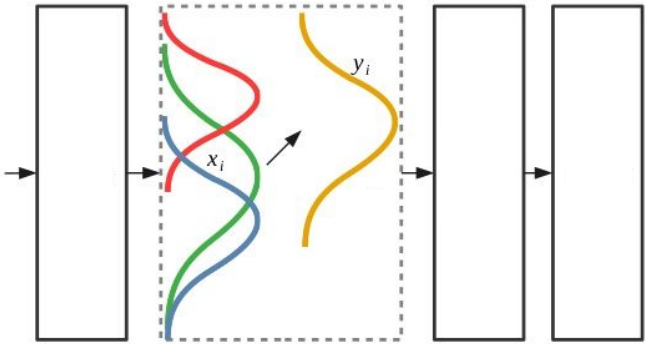
Dropout application on architecture

Credit Image: Srivastava et al., 2014

Definition ‘Batch size’ and ‘Epoch’

Epoch & Batch

- **Batch** partition of dataset
  - Batch size # of samples per batch
- **Epoch** complete iteration over batches
- **Batch normalization** Solves ‘*internal covariate shift*’
  - Distribution of input features differs for different batch
- **Assumption** batches independently and identically distributed

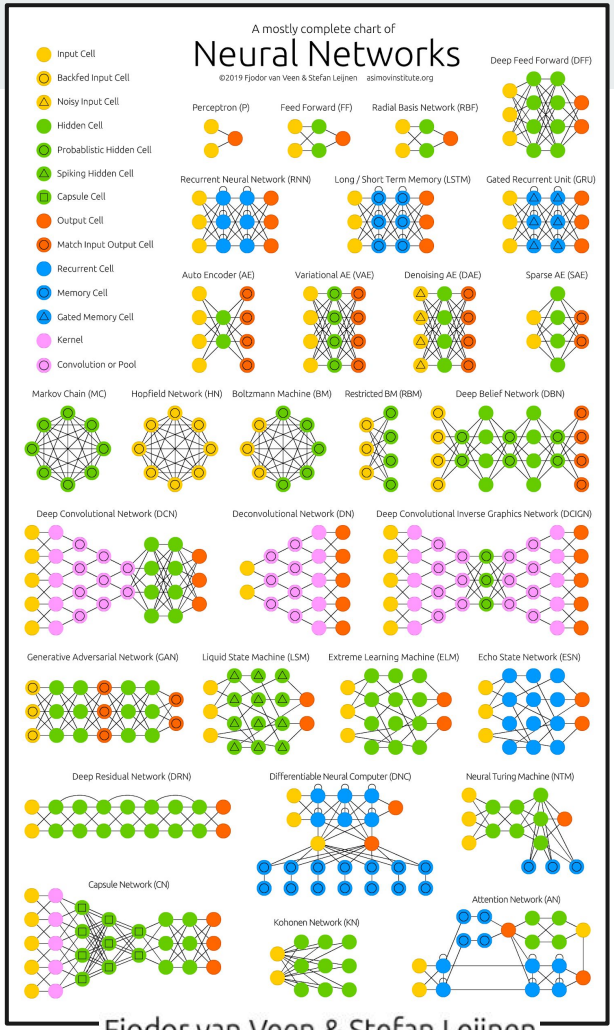


Ensure same mean and variance for x and y

Credit Image: Taiwan Center for Disease Control

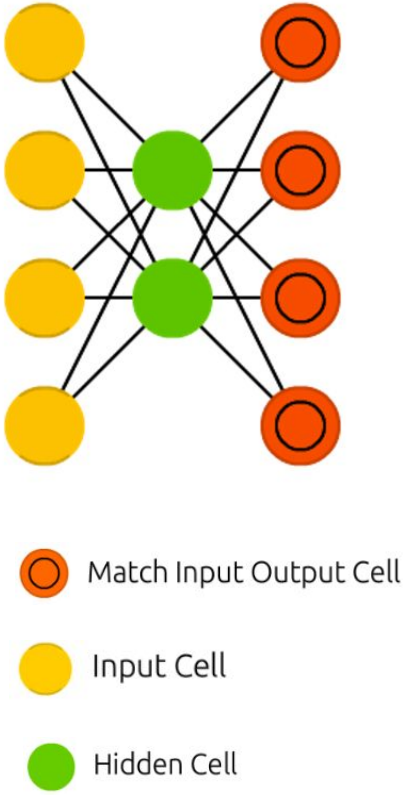
Network Architectures

1. Feed forward neural networks **FFNN**
2. Convolutional Neural Network **CNN**
3. Recurrent neural networks **RNN**
4. Long / short term memory **LSTM**
5. Gated recurrent units **GRU**
6. (Variational) Autoencoders (**V**)**AE**
7. Deep belief networks **DBN**
8. Capsule Networks **CapsNet**
9. ...



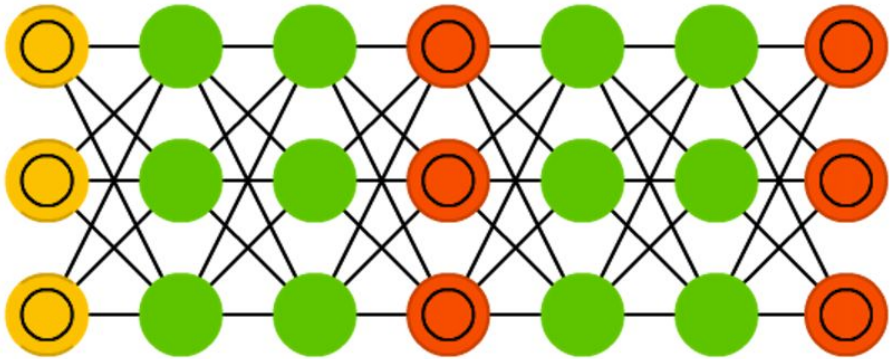
Autoencoders

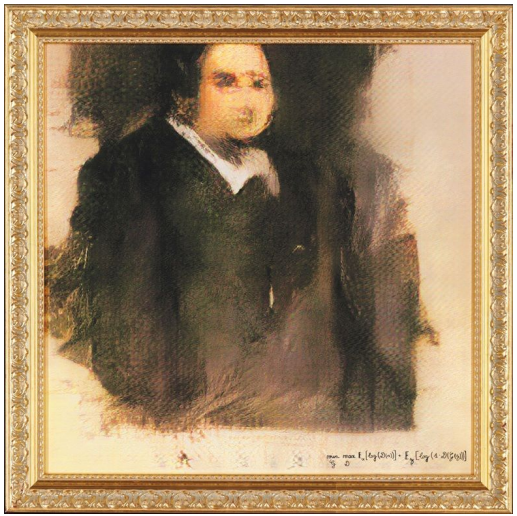
- Encode/ compress information
- Smallest layer (almost) always in the center
  - Layers before: **encoding** layers
  - Central layer: **coding** layer
  - Layers after: **decoding** layers
- Enforce ‘meaningful’ central units that cover the abstract representation of the input to output transformation
- Can conduct a PCA by SVD on coding layers



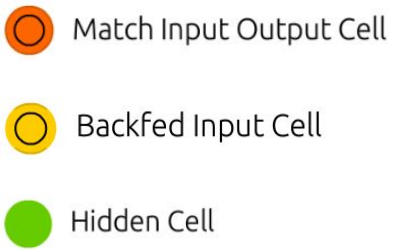
Generative Adversarial Networks

- Combination of two networks
- One generates content
- Other classifies content
  - Generated or real content?
- Competitive approach, train both networks simultaneously





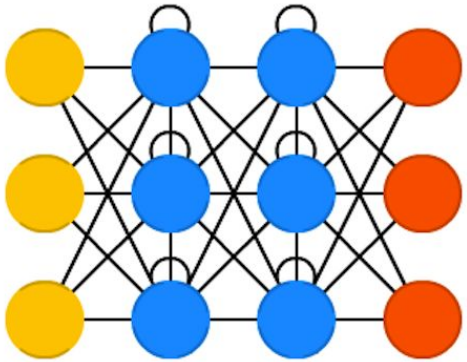
Portrait of Edmond Belamy
432,500$



Fjodor van Veen & Stefan Leijnen

Recurrent Neural Network

- Introduce nodes states as memory
- Nodes process information from **previous layer** and **themselves** from previous pass
- Order of input changes output
- Susceptible for vanishing gradient problem
- State information lost over time
- Utilized for data with a sequence
    - Speech understanding
    - Image processing
    - Autocompletion

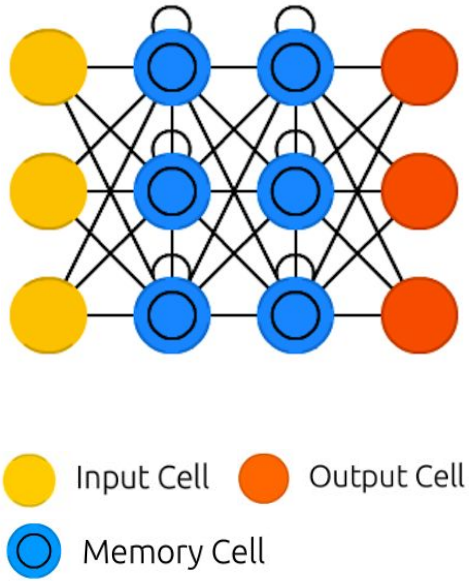


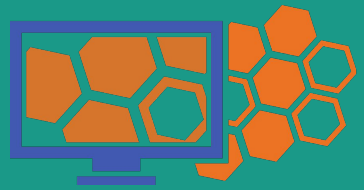
Central memory layer

Input Cell Recurrent Cell Output Cell

Long-Short Term memory

- Combat the vanishing gradient problem with **gates** and **memory cells**
- Three types of gates
  - Input, output and forget
- Gates safeguard information by stopping or allowing information flow
- Able to learn complex sequences:
  - Shakespeare
  - Composing primitive music
- Resources intensive





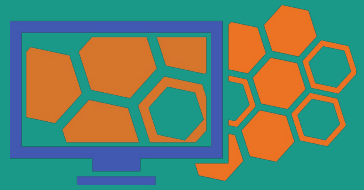
Third model

Keras and R

Keras and R

Requires

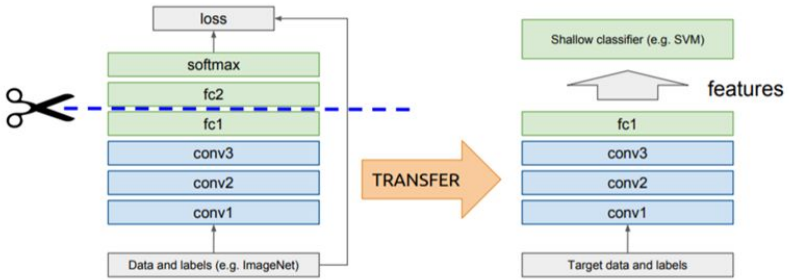
- R-studio Server
- Jupyter notebook
  - Kernel installation
- According to Keras documentation same performance



Third model
Mini-Hackathon
Melanoma Classification

Transfer learning

- Pre-trained model
- Adapt model to new task
  - Add new or retrain last layer(s)
- Reduces data usage and training time
- Powerful models available
  - E.g. BioBert

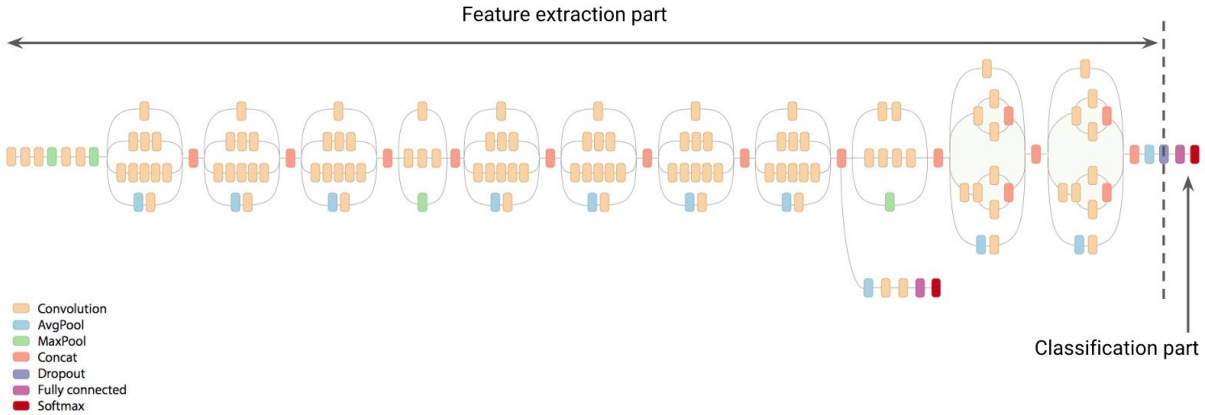


Retrain last layers

Credit Image: Dipjan Sakan

Inception v3 pre-trained model

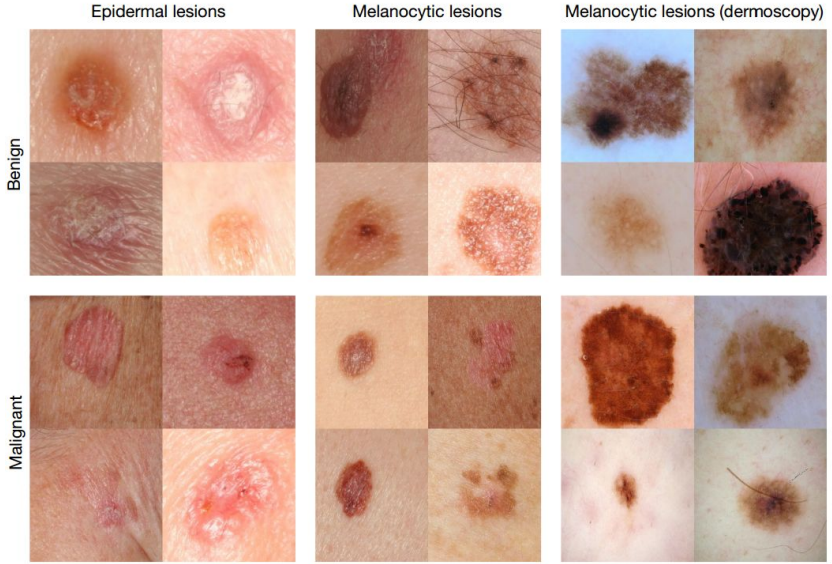
- Trained on various types of images
- Comprises of two parts
- Skip feature part and train classification part
- Won ImageNet classification
    - 14 million images
    - 1,000 classes



Inception v3 architecture

Melanoma classification challenge

- Real-world Kaggle challenge
- Designed by Stanford researchers
- Task is to classify lesions



Credit Image: Esteva, Kuprel et. al, 2017

Summary Deep Learning models

Positive 

- Neural networks flexible
- Transfer-learning powerful
- Non-linearity
- Large dimension input possible
- Dynamically adaptable architecture
- Parallelization possible

Negative ✗

- Complex
- Slow
- Black-box
- Hardware requirements high
- (Generally) requires significant amounts of data

However

"It's not who has the best algorithm that wins. It's who has the best data."