Machine learning

Yann Coadou

CPPM Marseille

esipap
European School of Instrumentation in Particle & Astroparticle Physics

Online edition, 27 January 2021

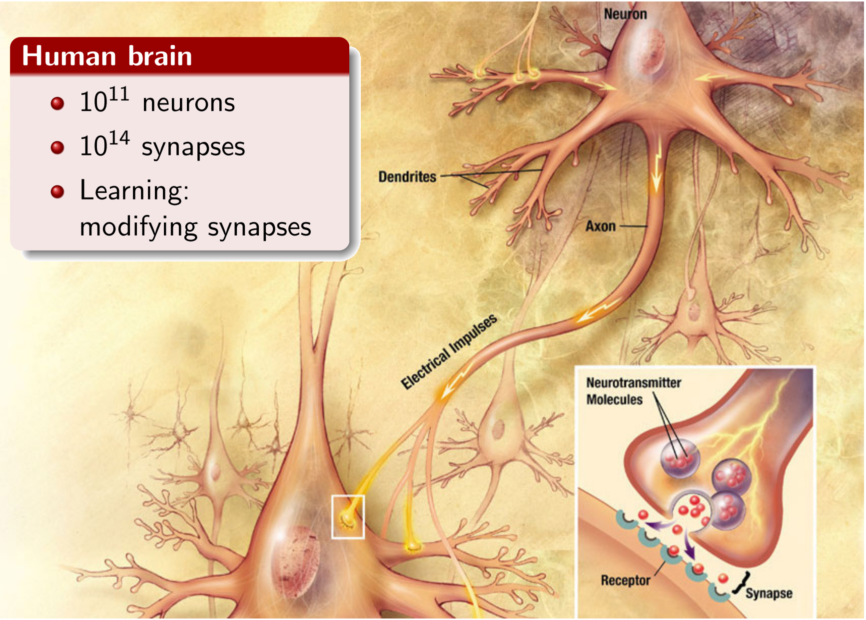
Outline


Apply now!
esipap...
European School of Instrumentation in Particle & Astroparticle Physics
18 JANUARY
12 MARCH
2021
Remote learning
• Course 1
Physics of Particle and Astroparticle Detectors
18 Jan - 12 Feb
• Course 2
Advanced Lectures on Particle Detectors and Applications
15 Feb - 12 Mar
Deadline 27 November 2020
ON-LINE APPLICATIONS
www.esipap.eu

Neural networks

Human brain

- 10^{11} neurons
- 10^{14} synapses
- Learning:
  modifying synapses

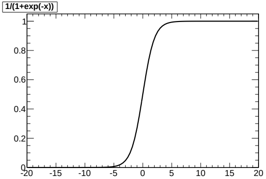


Brief history of artificial neural networks

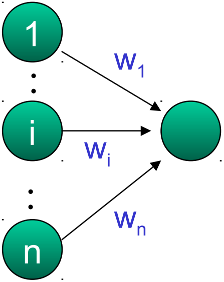
- 1943: W. McCulloch and W. Pitts explore capabilities of networks of simple neurons
- 1958: F. Rosenblatt introduces perceptron (single neuron with adjustable weights and threshold activation function)
- 1969: M. Minsky and S. Papert prove limitations of perceptron (linear separation only) and (wrongly) conjecture that multi-layered perceptrons have same limitations
  $\Rightarrow$ ANN research almost abandoned in 1970s!!!
- 1986: Rumelhart, Hinton and Williams introduce "backward propagation of errors": solves (partially) multi-layered learning
- Next: focus on multilayer perceptron (MLP)

Single neuron

- Remember linear separation (Fisher discriminant):
  $\lambda(x) = w \cdot x = \sum_{i=1}^{n} w_i x_i + w_0$
- Boundary at $\lambda(x) = 0$
- Replace threshold boundary by sigmoid (or tanh):



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



- σ: activation function (neuron activity)
- Neuron behaviour completely controlled by weights $w = \{w_0, \ldots, w_n\}$
- Training: minimisation of error/loss function (quadratic deviations, entropy [maximum likelihood]), via gradient descent or stochastic approximation

Universal approximation theorem

Let $\sigma(.)$ be a non-constant, bounded, and monotone-increasing continuous function. Let $\mathcal{C}(I_n)$ denote the space of continuous functions on the n-dimensional hypercube. Then, for any given function $f \in \mathcal{C}(I_n)$ and $\varepsilon > 0$ there exists an integer M and sets of real constants w_j, w_{ij} where $i = 1, \ldots, n$ and $j = 1, \ldots, M$ such that

$$y(x, w) = \sum_{j=1}^{M} w_j \sigma \left(\sum_{i=1}^{n} w_{ij} x_i + w_{0j} \right)$$

is an approximation of $f(.)$, that is $|y(x) - f(x)| < \varepsilon$.

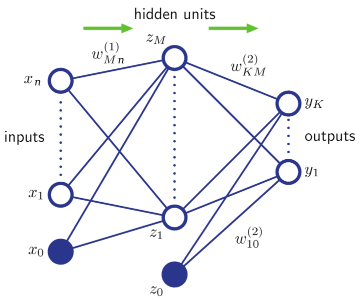
Interpretation

- You can approximate any continuous function to arbitrary precision with a linear combination of sigmoids
- Corollary 1: can approximate any continuous function with neurons!
- Corollary 2: a single hidden layer is enough
- Corollary 3: a linear output neuron is enough

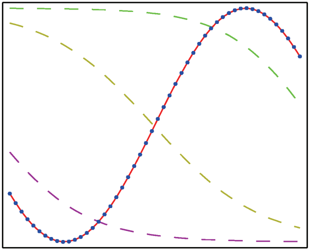
Multilayer perceptron: feedforward network

- Neurons organised in layers
- Output of one layer becomes input to next layer

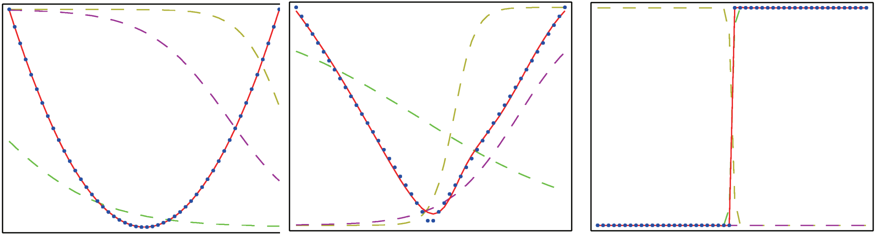
$$y_k(x, w) = \sum_{j=0}^{M} w_{kj}^{(2)} \underbrace{\sigma\left(\sum_{i=0}^{n} w_{ji}^{(1)} x_i\right)}_{z_j}$$



- 1 input (training data), 1 output
- 3 hidden neurons on one hidden layer



z_1 — *output*
z_2 — *training data*
z_3



Neural network training

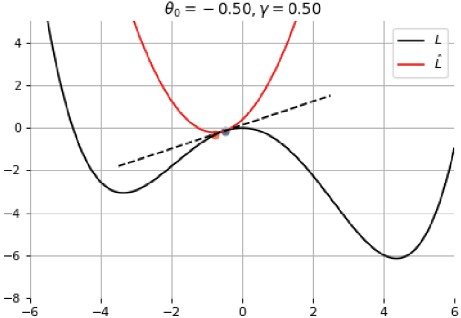
- Training means minimising error function $E(w)$
- $\frac{\partial E}{\partial w_j} = \sum_{n=1}^{N} -(t^{(n)} - y^{(n)})x_j^{(n)}$ with target $t^{(n)}$ (0 or 1), so $t^{(n)} - y^{(n)}$ is the error on event n
- All events at once (batch learning):
  - weights updated all at once after processing the entire training sample
  - finds the actual steepest descent
  - takes more time
  - usually: mini-batches (send events by batches)
  - new training events: need to restart training from scratch
- or one-by-one (online learning):
  - incremental learning: new training events included as they come
  - speeds up learning
  - may avoid local minima with stochastic component in minimisation
  - careful: depends on the order of training events
- One epoch: going through the entire training data once

Neural network training

- Minimise error function $E(w)$
- Gradient descent: $w^{(k+1)} = w^{(k)} - \eta \nabla_w E^{(k)}$
  with learning rate η

Neural network training

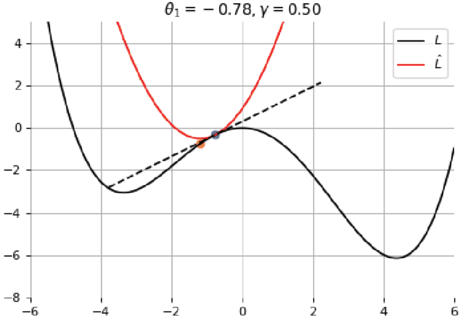
- Minimise error function $E(w)$
- Gradient descent: $w^{(k+1)} = w^{(k)} - \eta \nabla_w E^{(k)}$
  with learning rate η



©G. Louppe

local minimum

- Minimise error function $E(w)$
- Gradient descent: $w^{(k+1)} = w^{(k)} - \eta \nabla_w E^{(k)}$
  with learning rate η

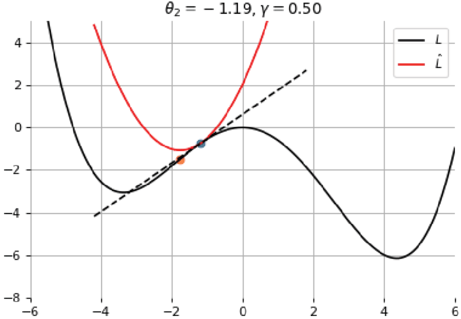


©G. Louppe

local minimum

Neural network training

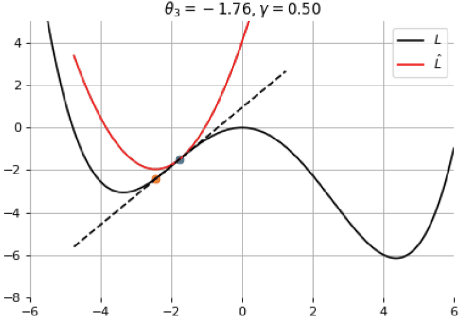
- Minimise error function $E(w)$
- Gradient descent: $w^{(k+1)} = w^{(k)} - \eta \nabla_w E^{(k)}$
  with learning rate η



©G. Louppe

local minimum

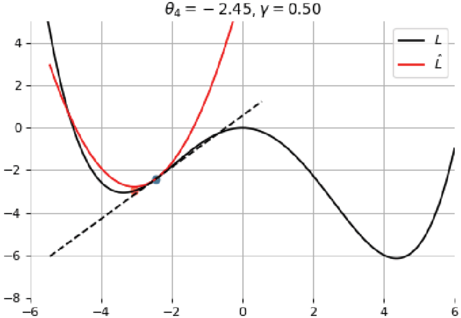
- Minimise error function $E(w)$
- Gradient descent: $w^{(k+1)} = w^{(k)} - \eta \nabla_w E^{(k)}$
  with learning rate η



local minimum

Neural network training

- Minimise error function $E(w)$
- Gradient descent: $w^{(k+1)} = w^{(k)} - \eta \nabla_w E^{(k)}$ with learning rate η

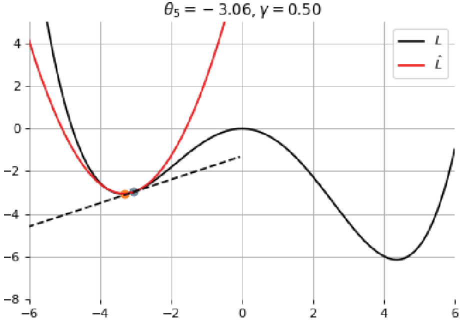


ⒸG. Louppe

local minimum

Neural network training

- Minimise error function $E(w)$
- Gradient descent: $w^{(k+1)} = w^{(k)} - \eta \nabla_w E^{(k)}$
  with learning rate η

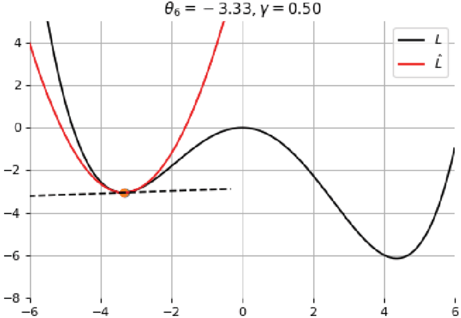


©G. Louppe

local minimum

Neural network training

- Minimise error function $E(w)$
- Gradient descent: $w^{(k+1)} = w^{(k)} - \eta \nabla_w E^{(k)}$ with learning rate η

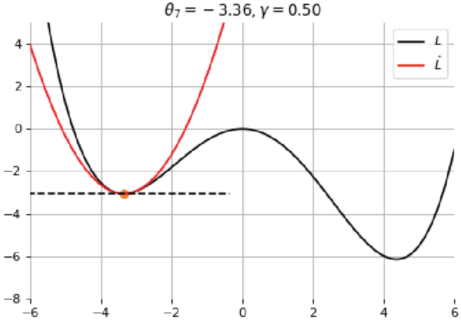


©G. Louppe

local minimum

Neural network training

- Minimise error function $E(w)$
- Gradient descent: $w^{(k+1)} = w^{(k)} - \eta \nabla_w E^{(k)}$
  with learning rate η

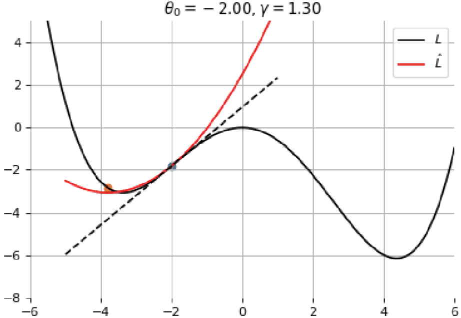


©G. Louppe

local minimum

Neural network training

- Minimise error function $E(w)$
- Gradient descent: $w^{(k+1)} = w^{(k)} - \eta \nabla_w E^{(k)}$
  with learning rate η

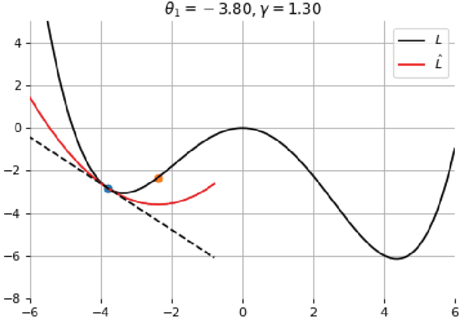


©G. Louppe

too large learning rate

Neural network training

- Minimise error function $E(w)$
- Gradient descent: $w^{(k+1)} = w^{(k)} - \eta \nabla_w E^{(k)}$
  with learning rate η

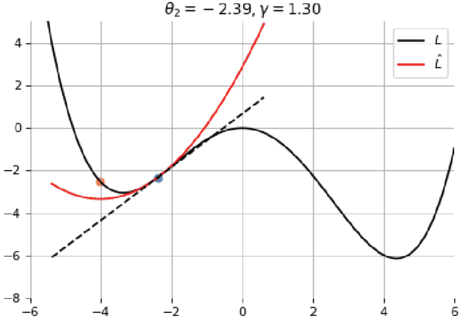


©G. Louppe

too large learning rate

Neural network training

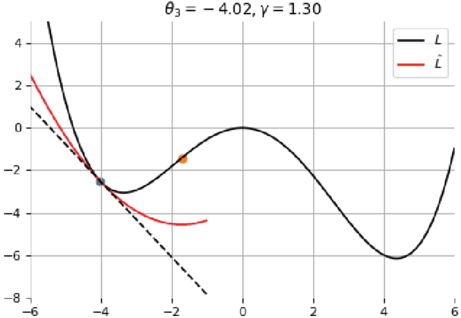
- Minimise error function $E(w)$
- Gradient descent: $w^{(k+1)} = w^{(k)} - \eta \nabla_w E^{(k)}$
  with learning rate η



ⒸG. Louppe

too large learning rate

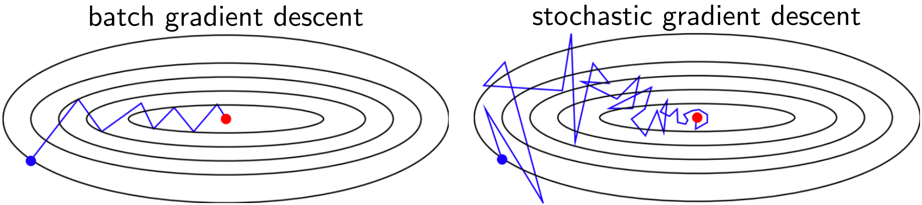
- Minimise error function $E(w)$
- Gradient descent: $w^{(k+1)} = w^{(k)} - \eta \nabla_w E^{(k)}$
  with learning rate η



too large learning rate

Neural network training: SGD

- Solution: stochastic gradient descent (SGD)

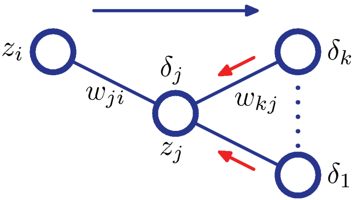


©G. Louppe

Backpropagation

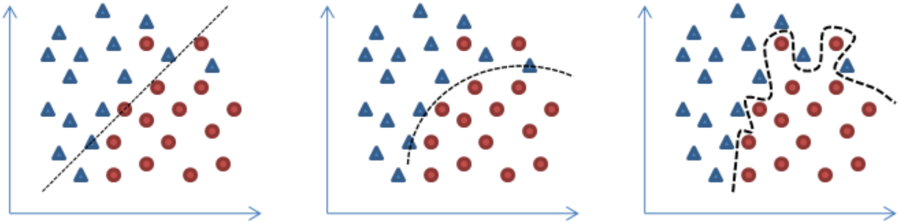
- Training means minimising error function $E(w)$
- For single neuron: $\frac{dE}{dw_k} = (y-t)x_k$
- One can show that for a network:

$$\frac{dE}{dw_{ji}} = \delta_j z_i, \text{ where}$$



$$\begin{aligned} \delta_k &= (y_k - t_k) \text{ for output neurons} \\ \delta_j &\propto \sum_k w_{kj}\delta_k \text{ otherwise} \end{aligned}$$

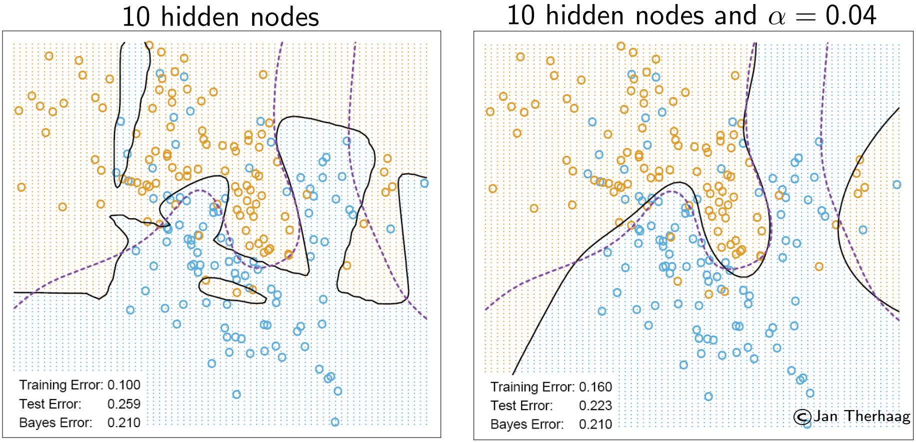
- Hence errors are propagated backwards



- Diverging weights can cause overfitting
- Mitigate by:
  - early stopping (after a fixed number of epochs)
  - monitoring error on test sample
  - regularisation, introducing a "weight decay" term to penalise large weights, preventing overfitting:

$$\tilde{E}(w) = E(w) + \frac{\alpha}{2}\sum_i w_i^2$$

Regularisation

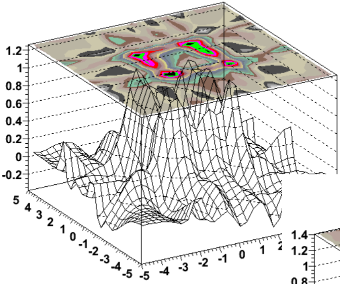


- Much less overfitting, better generalisation properties

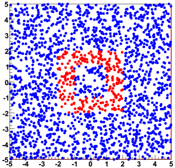
Neural networks: Tricks of the trade

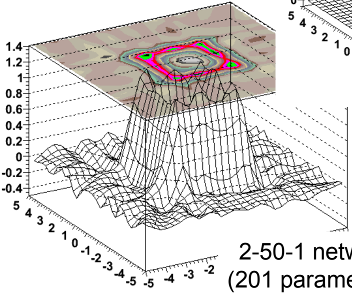
- Preprocess data:
  - if relevant, provide e.g. x/y instead of x and y
  - subtract the mean because the sigmoid derivative becomes negligible very fast (so, input mean close to 0)
  - normalise variances (close to 1)
  - shuffle training sample (order matters in online training)
- Initial random weights should be small to avoid saturation
- Batch/online training: depends on the problem
- Regularise weights to minimise overtraining.
- Make sure the training sample covers the full parameter space
- No rule (not even guestimates) about the number of hidden nodes (unless using constructive algorithm, adding resources as needed)
- A single hidden layer is enough for all purposes, but multiple hidden layers may allow for a solution with fewer parameters

Adding a hidden layer

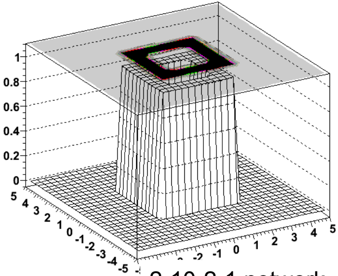


2-20-1 network
(81 parameters)





2-50-1 network
(201 parameters)



2-10-2-1 network
(55 parameters)

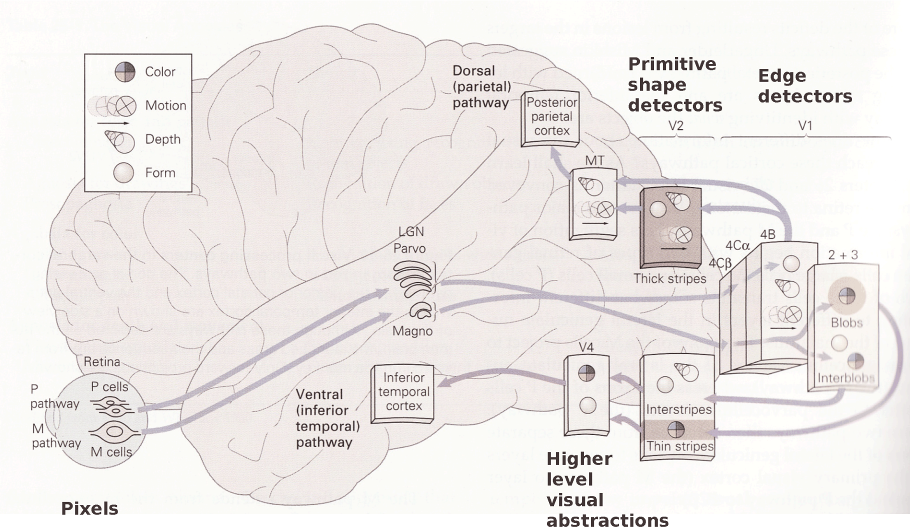
Deep learning

What is learning?

- Ability to learn underlying and previously unknown structure from examples
  $\Rightarrow$ capture variations
- Deep learning: have several hidden layers (> 2) in a neural network

Motivation for deep learning

- Inspired by the brain!
- Humans organise ideas hierarchically, through composition of simpler ideas
- Heavily unsupervised training, learning simpler tasks first, then combining into more abstract ones
- Learn first order features from raw inputs, then patterns in first order features, then etc.



Color
Motion
Depth
Form
Dorsal (parietal) pathway
Posterior parietal cortex
Primitive shape detectors
Edge detectors
V2
V1
MT
LGN
Parvo
Magno
4B
4Cα
4Cβ
2 + 3
Thick stripes
Blobs
Interblobs
Retina
P pathway
P cells
M pathway
M cells
Ventral (inferior temporal) pathway
Inferior temporal cortex
V4
Interstripes
Thin stripes
Higher level visual abstractions
Pixels

Deep learning in artificial intelligence

Mimicking the brain

- About 1% of neurons active simultaneously in the brain: distributed representation
  - activation of small subset of features, not mutually exclusive
  - more efficient than local representation
  - distributed representations necessary to achieve non-local generalization, exponentially more efficient than 1-of-N enumeration
  - example: integers in 1..N
    - local representation: vector of N bits with single 1 and N-1 zeros
    - distributed representation: vector of $\log_2 N$ bits (binary notation), exponentially more compact
- Meaning: information not localised in particular neuron but distributed across them

Deep architecture

- Insufficient depth can hurt
- Learn basic features first, then higher level ones
- Learn good intermediate representations, shared across tasks

Deep learning revolution

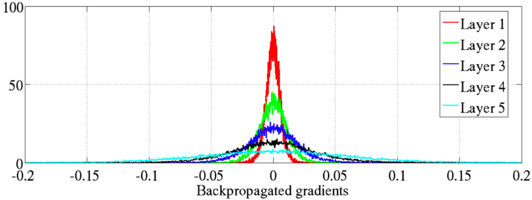
Deep networks were unattractive

- One layer theoretically enough for everything
- Used to perform worse than shallow networks with 1 or 2 hidden layers
- Apparently difficult/impossible to train (using random initial weights and supervised learning with backpropagation)
- Backpropagation issues:
  - requires labelled data (usually scarce and expensive)
  - does not scale well, getting stuck in local minima
  - "vanishing gradient": gradients getting very small further away from output $\Rightarrow$ early layers do not learn much, can even penalise overall performance

Deep learning revolution

Deep networks were unattractive

- One layer theoretically enough for everything
- Used to perform worse than shallow networks with 1 or 2 hidden layers
- Apparently difficult/impossible to train (using random initial weights and supervised learning with backpropagation)
- Backpropagation issues:
  - requires labelled data (usually scarce and expensive)
  - does not scale well, getting stuck in local minima
  - "vanishing gradient": gradients getting very small further away from output ⇒ early layers do not learn much, can even penalise overall performance



Deep learning revolution

Deep networks were unattractive

- One layer theoretically enough for everything
- Used to perform worse than shallow networks with 1 or 2 hidden layers
- Apparently difficult/impossible to train (using random initial weights and supervised learning with backpropagation)
- Backpropagation issues:
  - requires labelled data (usually scarce and expensive)
  - does not scale well, getting stuck in local minima
  - "vanishing gradient": gradients getting very small further away from output $\Rightarrow$ early layers do not learn much, can even penalise overall performance

Breakthroughs around 2006 (Bengio, Hinton, LeCun)

- Train each layer independently
- Can use unlabelled data (a lot of it)
- New activation functions
- Possible thanks to algorithmic innovations, computing resources, data!

Greedy layer-wise pre-training

Algorithm

- Take input information
- Train feature extractor
- Use output as input to training another feature extractor
- Keep adding layers, train each layer separately
- Finalise with a supervised classifier, taking last feature extractor output as input
- All steps above: pre-training
- Fine-tune the whole thing with supervised training (backpropagation)
  - initial weights are those from pre-training

Feature extractors

- Restricted Boltzmann machine (RBM), auto-encoder, sparse auto-encoder, denoising auto-encoder, etc.
- Note: important to not use linear activation functions in hidden layers. Combination of linear functions still linear, so equivalent to single hidden layer

Why does unsupervised training work?

Optimisation hypothesis

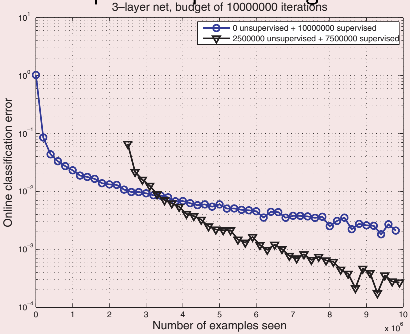
- Training one layer at a time scales well
- Backpropagation from sensible features
- Better local minimum than random initialisation, local search around it

Overfitting/regularisation hypothesis

- More info in inputs than labels
- No need for final discriminant to discover features
- Fine-tuning only at category boundaries

Example

- Stacked denoising auto-encoders
- 10 million handwritten digits
- First 2.5 million used for unsupervised pre-training



- Worse with supervision: eliminates projections of data not useful for local cost but helpful for deep model cost

An example from Google research team

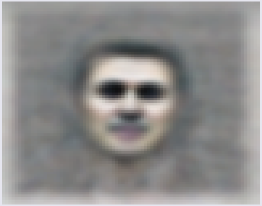
▸ 2011 paper

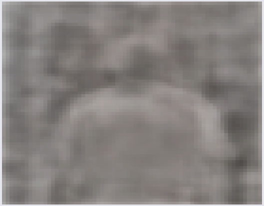
A "giant" neural network

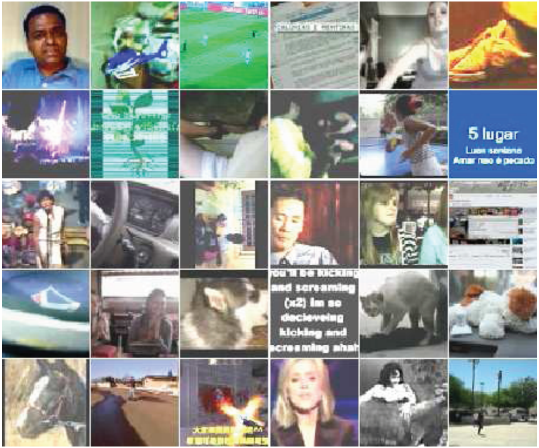
- At Google they trained a 9-layered NN with 1 billion connections
  - trained on 10 million 200×200 pixel images from YouTube videos
  - on 1000 machines (16000 cores) for 3 days, unsupervised learning
- Sounds big? The human brain has 100 billion (10^{11}) neurons and 100 trillion (10^{14}) connections...

What it did

- It learned to recognise faces, one of the original goals
- . . . but also cat faces (among the most popular things in YouTube videos) and body shapes



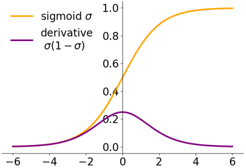




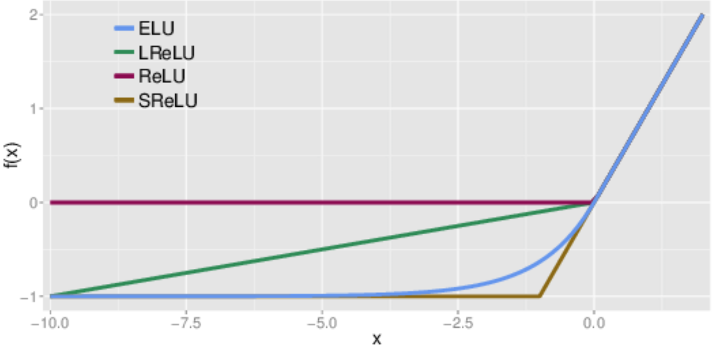
- Features extracted from such images
- Results shown to be robust to
  - colour
  - translation
  - scaling
  - out-of-plane rotation

Rectified linear unit ReLU

- One of reasons for vanishing gradient: sigmoid activation
  - tiny non-varying derivative away from zero
- Solution: non-saturating function
- Simplest case: rectified linear unit ReLU

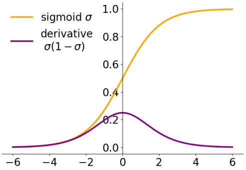


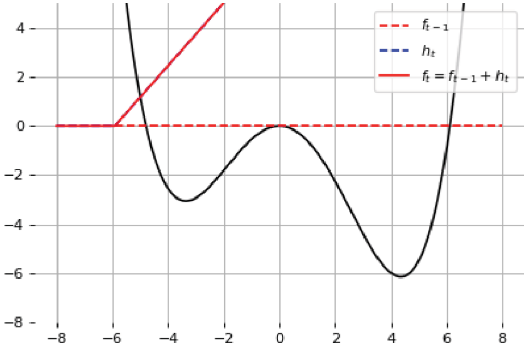
- Other variants: leaky ReLU, shifted ReLU (SReLU), exponential linear unit (ELU), etc.



arXiv: 1511.07289

- One of reasons for vanishing gradient: sigmoid activation
  - tiny non-varying derivative away from zero
- Solution: non-saturating function
- Simplest case: rectified linear unit ReLU
- Other variants: leaky ReLU, shifted ReLU (SReLU), exponential linear unit (ELU), etc.

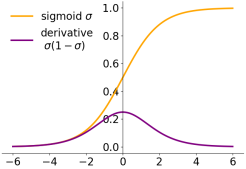


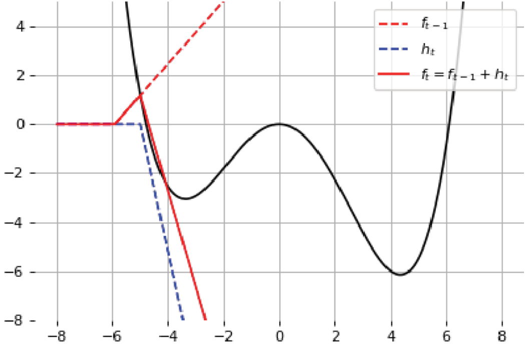


©G. Louppe

Rectified linear unit ReLU

- One of reasons for vanishing gradient: sigmoid activation
  - tiny non-varying derivative away from zero
- Solution: non-saturating function
- Simplest case: rectified linear unit ReLU
- Other variants: leaky ReLU, shifted ReLU (SReLU), exponential linear unit (ELU), etc.

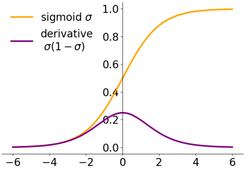




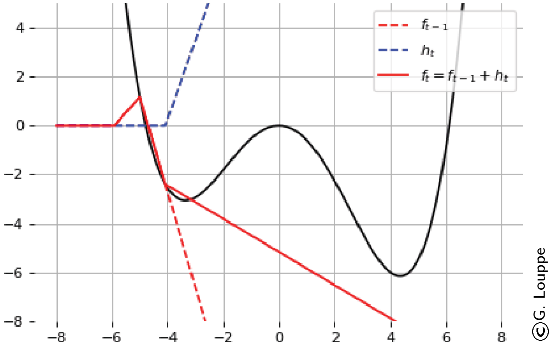
©G. Louppe

Rectified linear unit ReLU

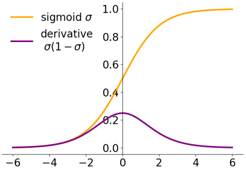
- One of reasons for vanishing gradient: sigmoid activation
  - tiny non-varying derivative away from zero
- Solution: non-saturating function
- Simplest case: rectified linear unit ReLU



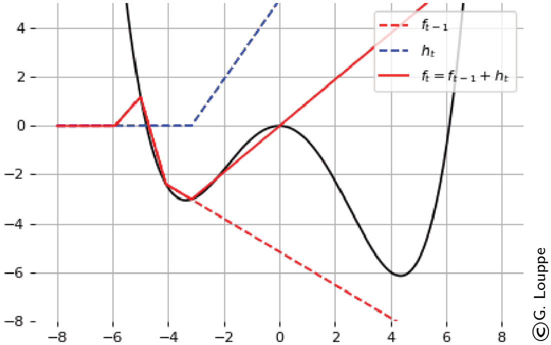
- Other variants: leaky ReLU, shifted ReLU (SReLU), exponential linear unit (ELU), etc.



- One of reasons for vanishing gradient: sigmoid activation
  - tiny non-varying derivative away from zero
- Solution: non-saturating function
- Simplest case: rectified linear unit ReLU

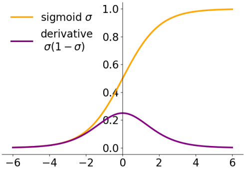


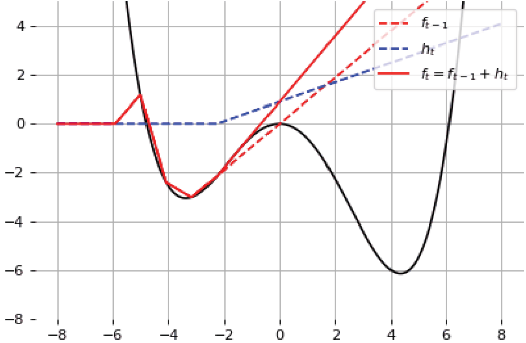
- Other variants: leaky ReLU, shifted ReLU (SReLU), exponential linear unit (ELU), etc.



Rectified linear unit ReLU

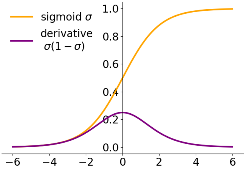
- One of reasons for vanishing gradient: sigmoid activation
  - tiny non-varying derivative away from zero
- Solution: non-saturating function
- Simplest case: rectified linear unit ReLU
- Other variants: leaky ReLU, shifted ReLU (SReLU), exponential linear unit (ELU), etc.



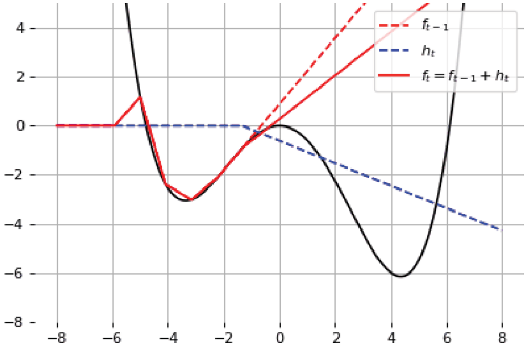


©G. Louppe

- One of reasons for vanishing gradient: sigmoid activation
  - tiny non-varying derivative away from zero
- Solution: non-saturating function
- Simplest case: rectified linear unit ReLU



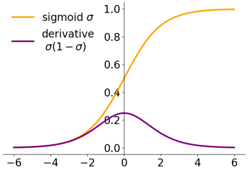
- Other variants: leaky ReLU, shifted ReLU (SReLU), exponential linear unit (ELU), etc.



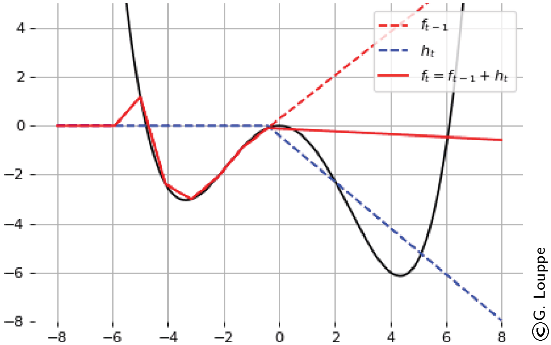
©G. Louppe

Rectified linear unit ReLU

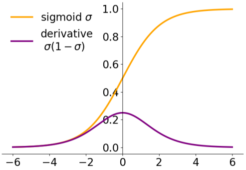
- One of reasons for vanishing gradient: sigmoid activation
  - tiny non-varying derivative away from zero
- Solution: non-saturating function
- Simplest case: rectified linear unit ReLU



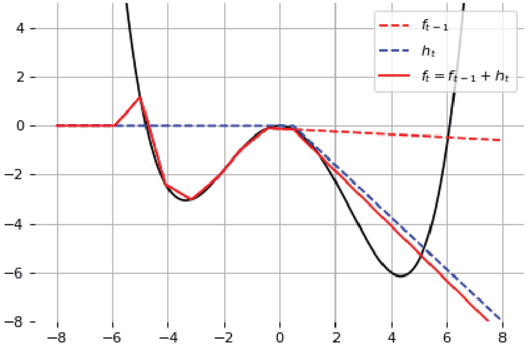
- Other variants: leaky ReLU, shifted ReLU (SReLU), exponential linear unit (ELU), etc.



- One of reasons for vanishing gradient: sigmoid activation
    - tiny non-varying derivative away from zero
- Solution: non-saturating function
- Simplest case: rectified linear unit ReLU



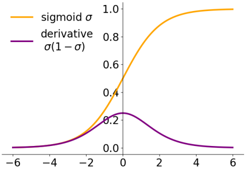
- Other variants: leaky ReLU, shifted ReLU (SReLU), exponential linear unit (ELU), etc.



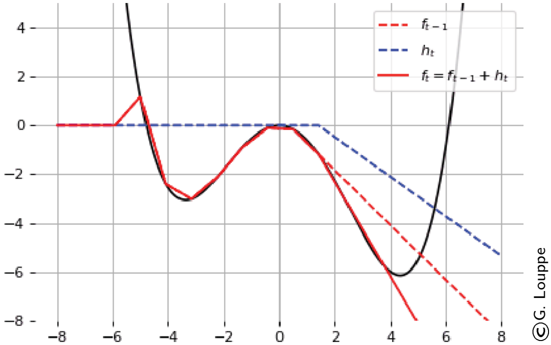
©G. Louppe

Rectified linear unit ReLU

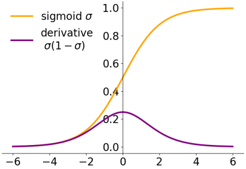
- One of reasons for vanishing gradient: sigmoid activation
  - tiny non-varying derivative away from zero
- Solution: non-saturating function
- Simplest case: rectified linear unit ReLU



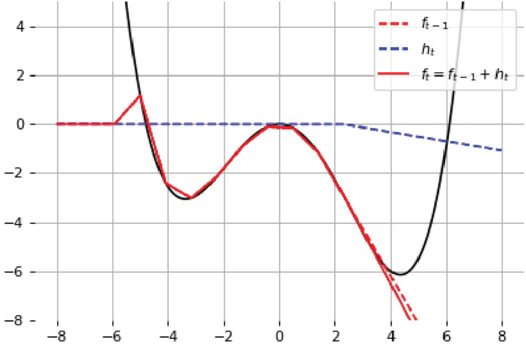
- Other variants: leaky ReLU, shifted ReLU (SReLU), exponential linear unit (ELU), etc.



- One of reasons for vanishing gradient: sigmoid activation
  - tiny non-varying derivative away from zero
- Solution: non-saturating function
- Simplest case: rectified linear unit ReLU

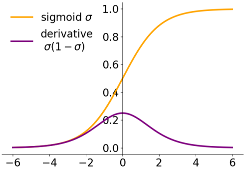


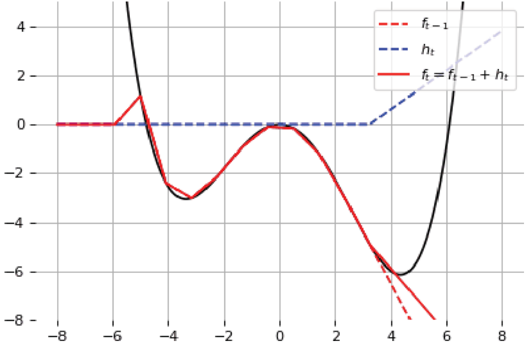
- Other variants: leaky ReLU, shifted ReLU (SReLU), exponential linear unit (ELU), etc.



© G. Louppe

- One of reasons for vanishing gradient: sigmoid activation
  - tiny non-varying derivative away from zero
- Solution: non-saturating function
- Simplest case: rectified linear unit ReLU
- Other variants: leaky ReLU, shifted ReLU (SReLU), exponential linear unit (ELU), etc.

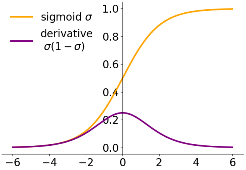


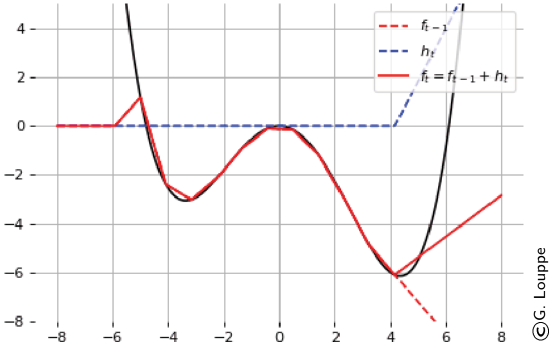


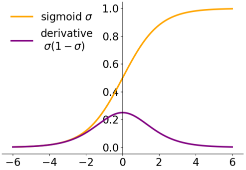
© G. Louppe

Rectified linear unit ReLU

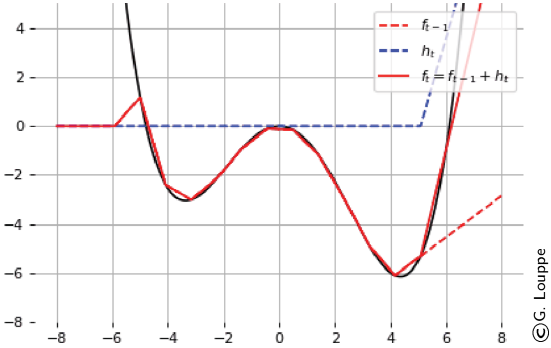
- One of reasons for vanishing gradient: sigmoid activation
  - tiny non-varying derivative away from zero
- Solution: non-saturating function
- Simplest case: rectified linear unit ReLU
- Other variants: leaky ReLU, shifted ReLU (SReLU), exponential linear unit (ELU), etc.



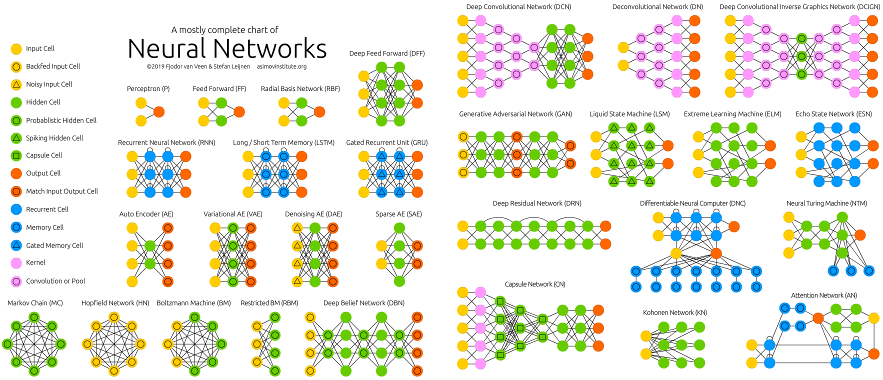




- One of reasons for vanishing gradient: sigmoid activation
  - tiny non-varying derivative away from zero
- Solution: non-saturating function
- Simplest case: rectified linear unit ReLU
- Other variants: leaky ReLU, shifted ReLU (SReLU), exponential linear unit (ELU), etc.



Neural network zoo

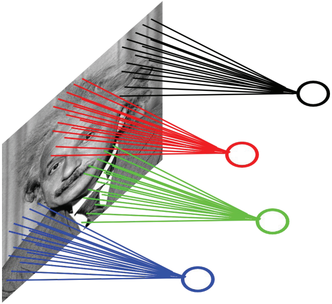


▸ https://www.asimovinstitute.org/

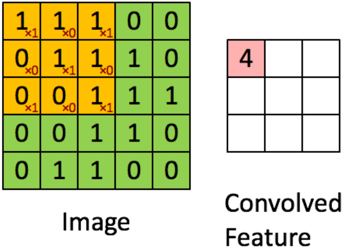
- Many possible network structures
- Moving away from feature engineering to model design

Convolutional networks

- Images are stationary: can learn feature in one part and apply it in another

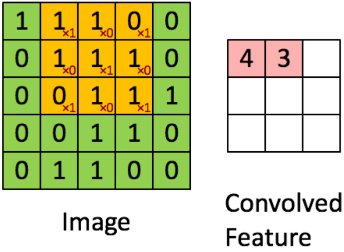


- Images are stationary: can learn feature in one part and apply it in another
- Use e.g. small patch sampled randomly, learn feature, convolve with full image



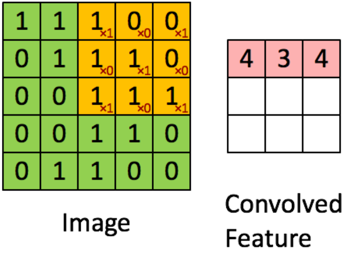
Convolutional networks

- Images are stationary: can learn feature in one part and apply it in another
- Use e.g. small patch sampled randomly, learn feature, convolve with full image



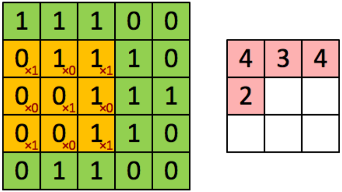
Convolutional networks

- Images are stationary: can learn feature in one part and apply it in another
- Use e.g. small patch sampled randomly, learn feature, convolve with full image



Convolutional networks

- Images are stationary: can learn feature in one part and apply it in another
- Use e.g. small patch sampled randomly, learn feature, convolve with full image

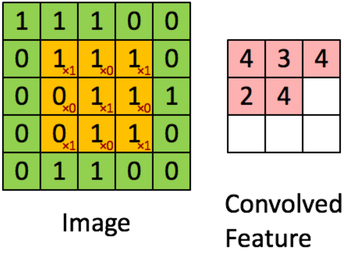


Image

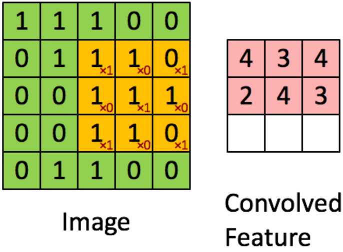
Convolved Feature

Convolutional networks

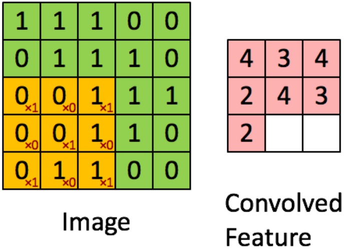
- Images are stationary: can learn feature in one part and apply it in another
- Use e.g. small patch sampled randomly, learn feature, convolve with full image



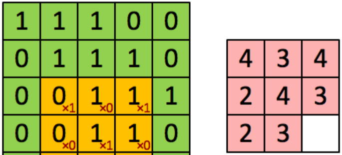
- Images are stationary: can learn feature in one part and apply it in another
- Use e.g. small patch sampled randomly, learn feature, convolve with full image



- Images are stationary: can learn feature in one part and apply it in another
- Use e.g. small patch sampled randomly, learn feature, convolve with full image



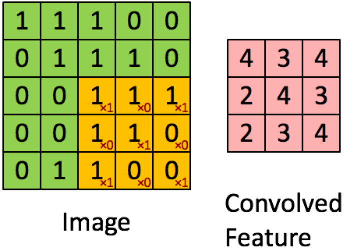
- Images are stationary: can learn feature in one part and apply it in another
- Use e.g. small patch sampled randomly, learn feature, convolve with full image



Image

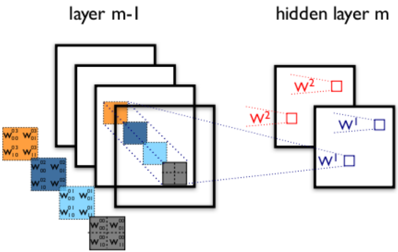
Convolved Feature

- Images are stationary: can learn feature in one part and apply it in another
- Use e.g. small patch sampled randomly, learn feature, convolve with full image

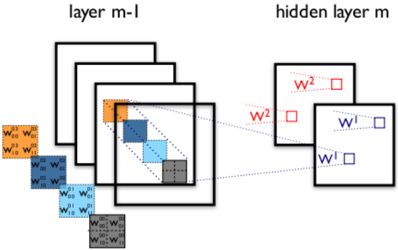


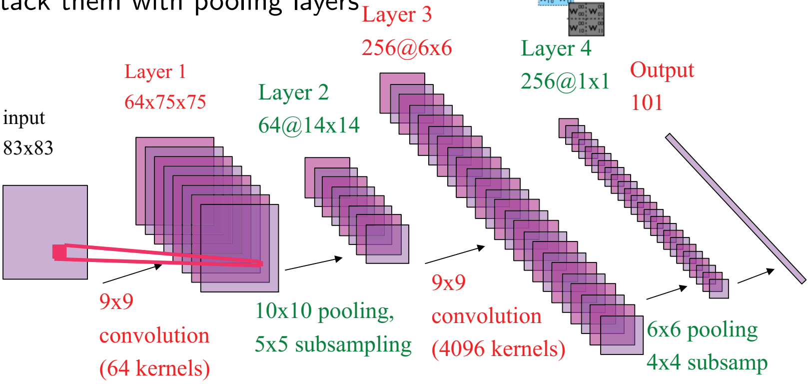
Convolutional networks

- Images are stationary: can learn feature in one part and apply it in another
- Use e.g. small patch sampled randomly, learn feature, convolve with full image
- Build several "feature maps"

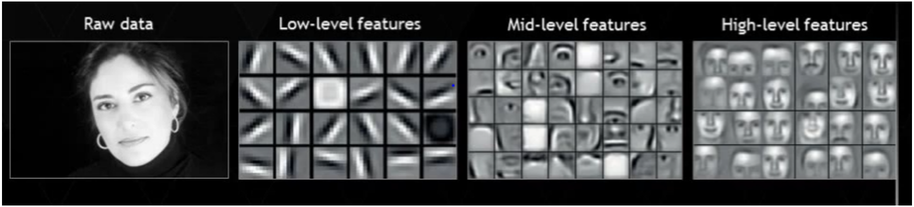


- Images are stationary: can learn feature in one part and apply it in another
- Use e.g. small patch sampled randomly, learn feature, convolve with full image
- Build several “feature maps”
- Stack them with pooling layers



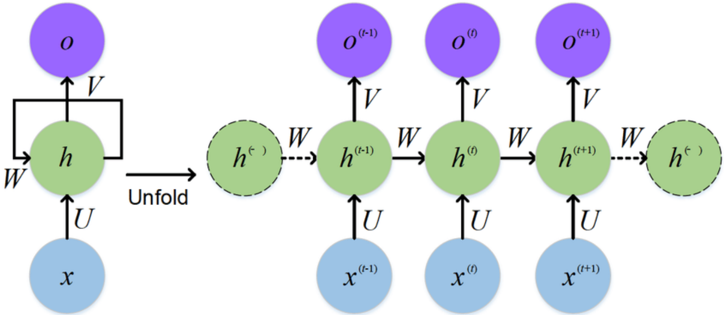


Learning feature hierarchy



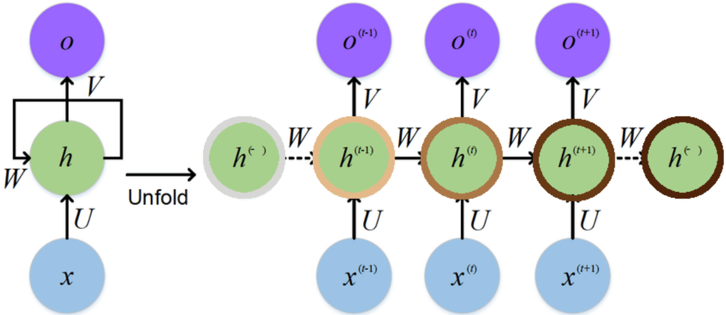
- Many problems require processing a sequence
  - sequence classification
    - text analysis ("sentiment analysis")
    - DNA sequencing
    - action selection
  - sequence synthesis
    - text synthesis
    - music/video
  - sequence translation
    - speech recognition
    - translation
- Usually variable length sequences (number of words/ notes/ frames/ etc.)
- Use a recurrent model, maintaining a recurrent state updated after each step

Recurrent neural networks

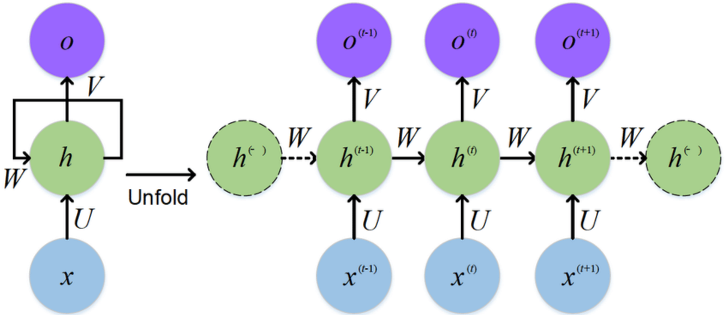


- Keeps information from earlier frames while processing (variable-size) sequence
- Could also be bi-directional, consuming sequence in both directions

Recurrent neural networks

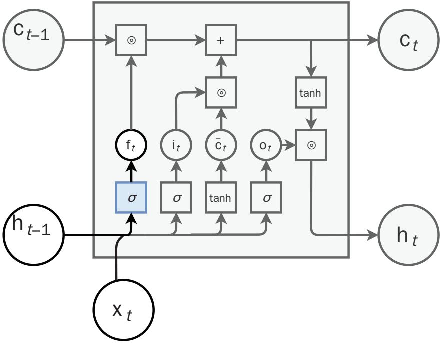


- Keeps information from earlier frames while processing (variable-size) sequence
- Could also be bi-directional, consuming sequence in both directions
- Issue: early frames diluted over sequence $\Rightarrow$ memory loss



- Keeps information from earlier frames while processing (variable-size) sequence
- Could also be bi-directional, consuming sequence in both directions
- Issue: early frames diluted over sequence ⇒ memory loss
- Introducing long short-term memory (LSTM) networks
  - using forget gate to regulate information flow
  - also possible with gated recurrent units (GRU)

Long short-term memory (LSTM)

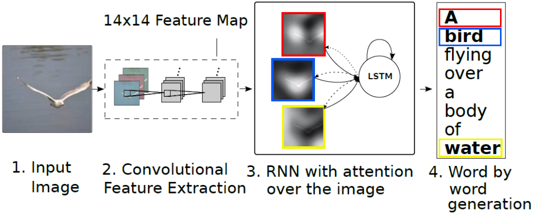


- Recurrent state split in two parts
  - cell state c_t
  - output state h_t
- Forget gate f_t to erase cell state info
- Input gate i_t to update cell state info
- Output gate o_t to select output state

Recurrent neural networks examples

Labelling images

arXiv:1502.03044

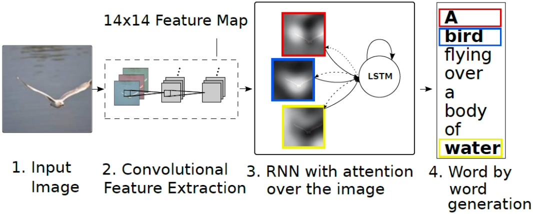


A woman is throwing a frisbee in a park.

Recurrent neural networks examples

Labelling images

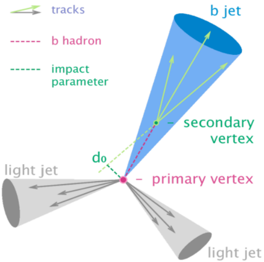
arXiv:1502.03044



A woman is throwing a frisbee in a park.

b-jet tagging in ATLAS experiment

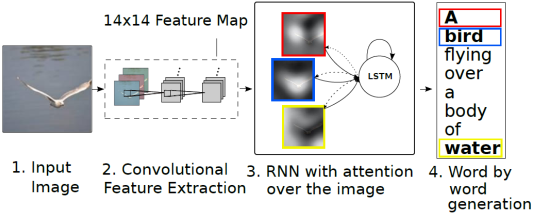
ATL-PHYS-PUB-2017-003



Recurrent neural networks examples

Labelling images

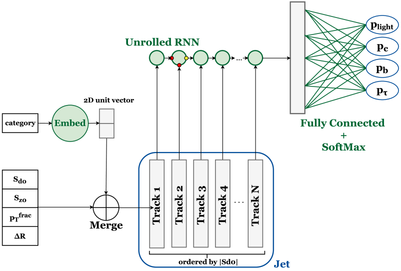
arXiv:1502.03044

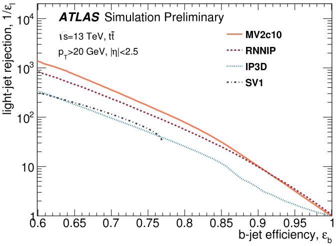


A woman is throwing a frisbee in a park.

b-jet tagging in ATLAS experiment

ATL-PHYS-PUB-2017-003





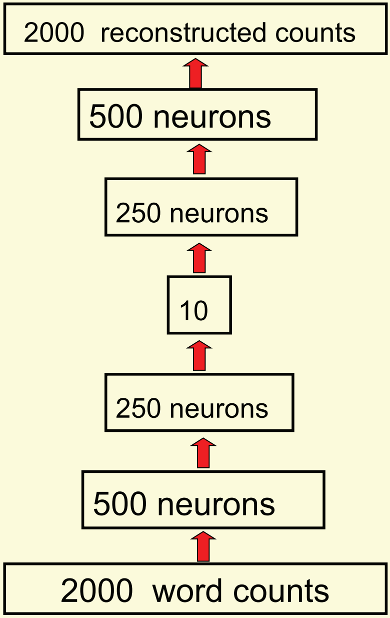
Auto-encoders

Approximate the identity function

- Build a network whose output is similar to its input
- Sounds trivial? Except if imposing constraints on network (e.g., # of neurons, locally connected network) to discover interesting structures
- Can be viewed as lossy compression of input

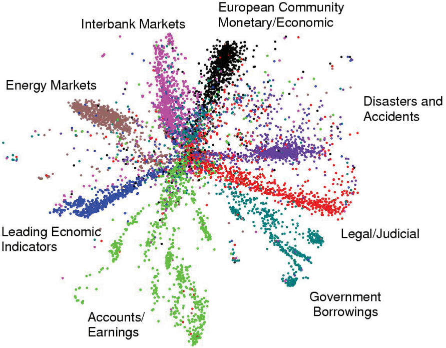
Finding similar books

- Get count of 2000 most common words per book
- "Compress" to 10 numbers



With principle component analysis (PCA)

With autoencoder



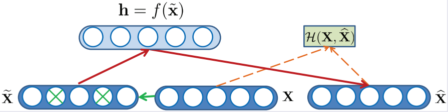
Other auto-encoders

Sparse auto-encoder

- Sparsity: try to have low activation of neurons (like in the brain)
- Compute average activation of each hidden unit over training set
- Add constraint to cost function to make average lower than some value close to 0

Denoising auto-encoder

- Stochastically corrupt inputs
- Train to reconstruct uncorrupted input



Locally connected auto-encoder

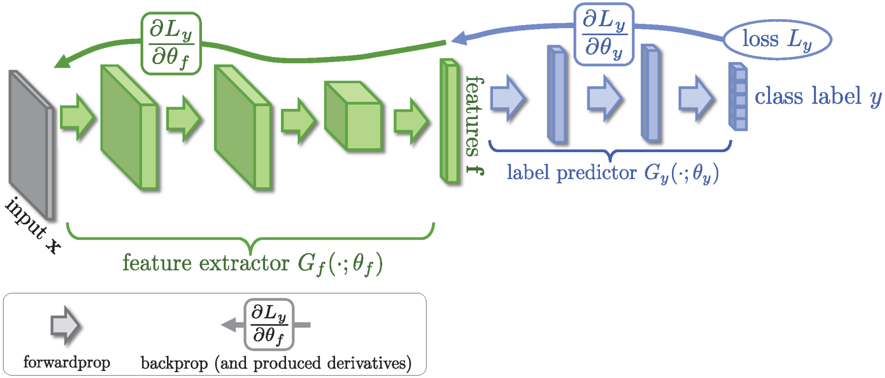
- Allow hidden units to connect only to small subset of input units
- Useful with increasing number of input features (e.g., bigger image)
- Inspired by biology: visual system has localised receptive fields

Domain adaptation and adversarial training

- Typical training
  - signal and background from simulation
  - results compared to real data to make measurement
- Requires good data–simulation agreement

arXiv:1409.7495

arXiv:1505.07818

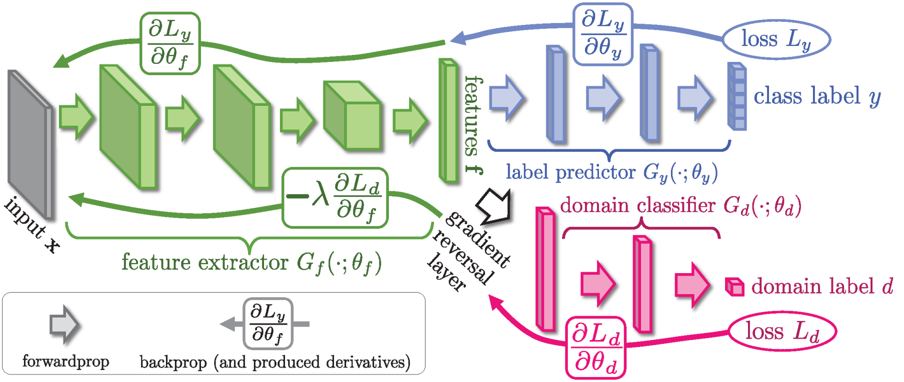


Domain adaptation and adversarial training

- Typical training
  - signal and background from simulation
  - results compared to real data to make measurement
- Requires good data–simulation agreement
- Possibility to use adversarial training and domain adaptation to account for discrepancies/systematic uncertainties

arXiv:1409.7495

arXiv:1505.07818

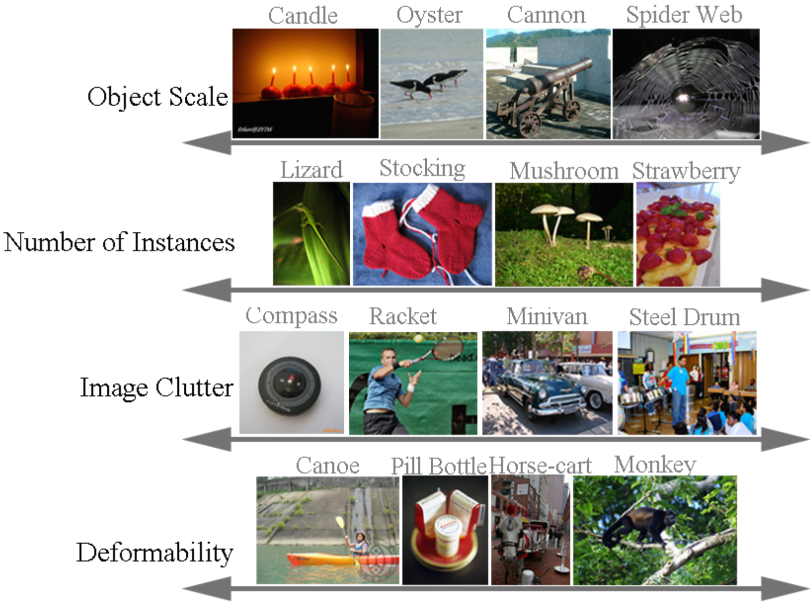


Summary paper

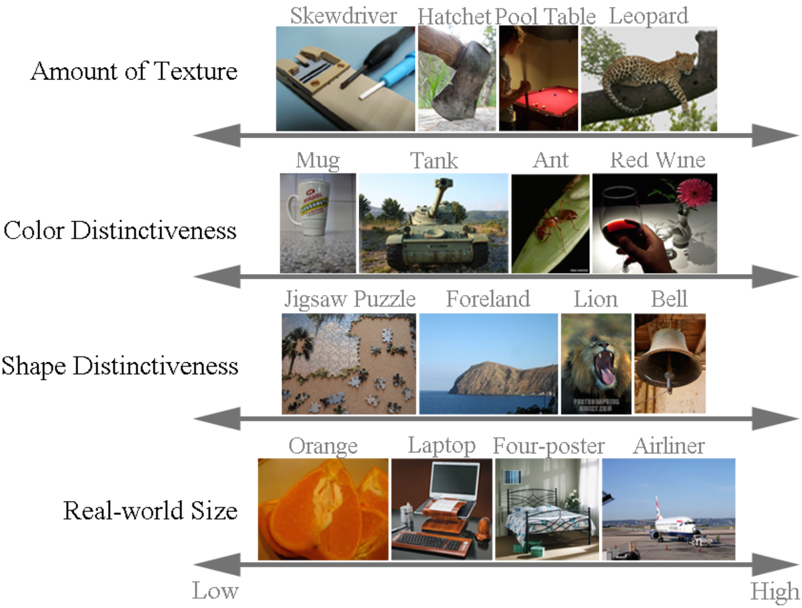
ImageNet Large Scale Visual Recognition Challenge

- ImageNet: database with 14 million images and 20k categories
- Used 1000 categories and about 1.3 million manually annotated images

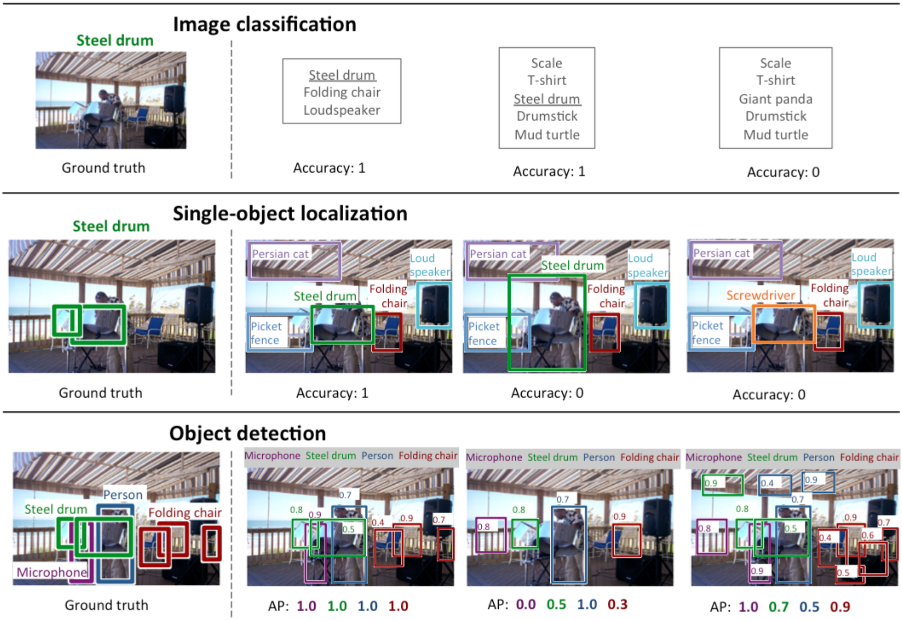
ILSVRC 2014 images



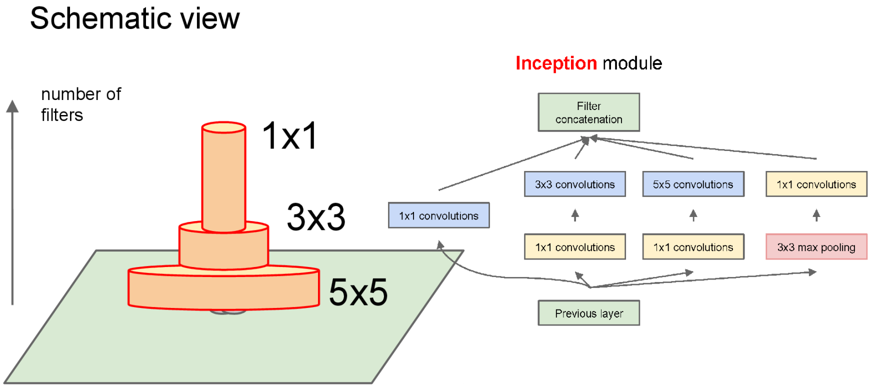
ILSVRC 2014 images



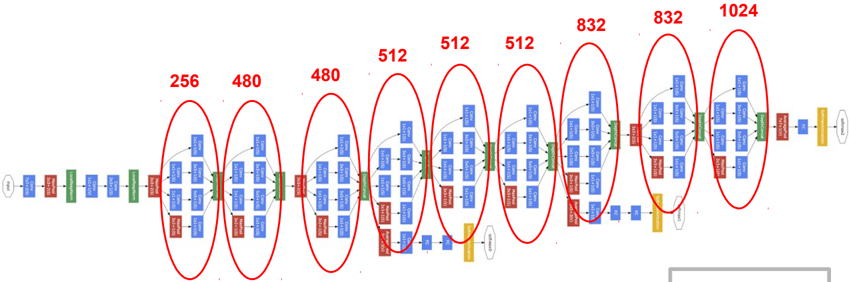
ILSVRC 2014 tasks



- Google of course! (first time)
- GoogLeNet:



- Google of course! (first time)
- GoogLeNet:



9 **Inception** modules

Network in a network in a network...

Convolution
Pooling
Softmax
Other

Classification failure cases

Groundtruth: **Police car**

GoogLeNet:

- **laptop**
- **hair drier**
- **binocular**
- **ATM machine**
- **seat belt**

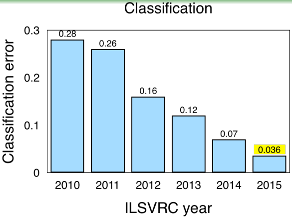
Classification failure cases

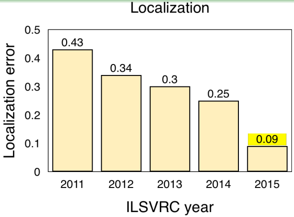
Groundtruth: **hay**

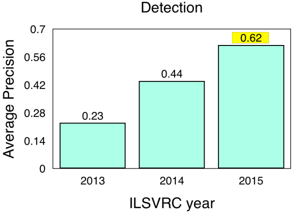
GoogLeNet:

- **sorrel (horse)**
- **hartebeest**
- **Arabian camel**
- **warthog**
- **gaselle**

ILSVRC 2010–2016





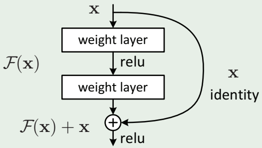


2010–14: 4.2x reduction   1.7x reduction   1.9x increase

ILSVRC 2015 (same dataset as 2014)

arXiv:1512.03385

- Winner: MSRA (Microsoft Research in Beijing)
- Deep residual networks with > 150 layers
- Classification error: 6.7% → 3.6% (1.9x)
- Localisation error: 26.7% → 9.0% (2.8x)
- Object detection: 43.9% → 62.1% (1.4x)

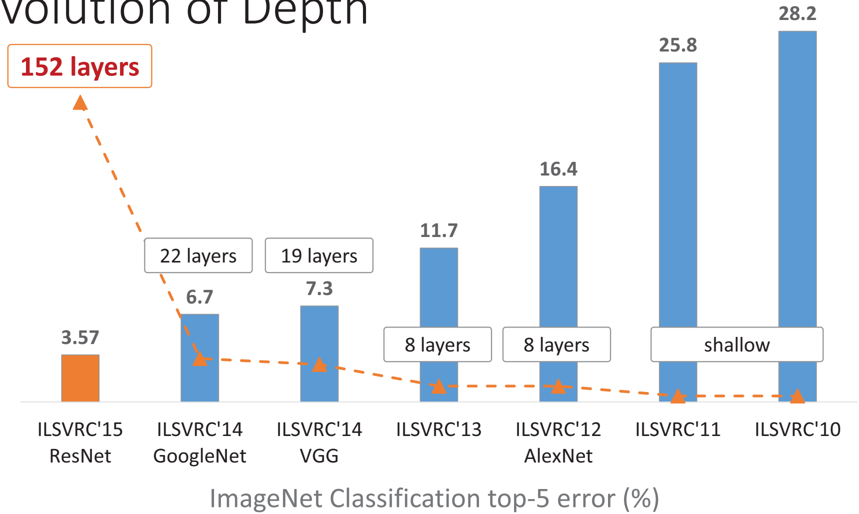


ILSVRC 2016

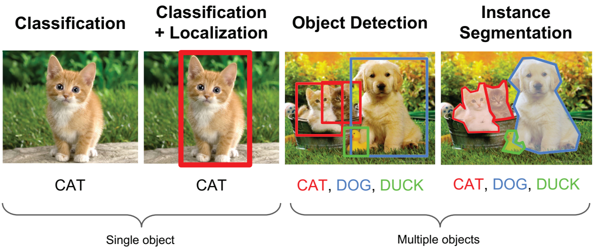
http://image-net.org/challenges/LSVRC/2016

- Mostly ResNets. Classification: 0.030; localisation: 0.08; detection: 0.66

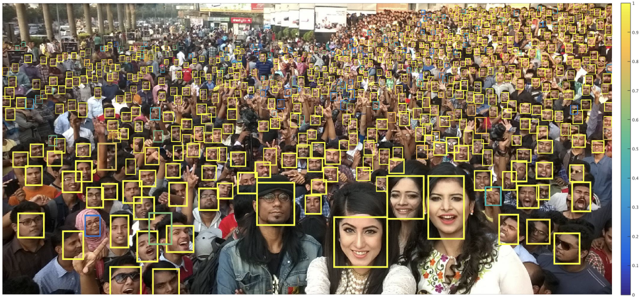
Revolution of Depth



Going further



- More and more refinement (segmentation)
- More objects, in real time on video1/video2/video3



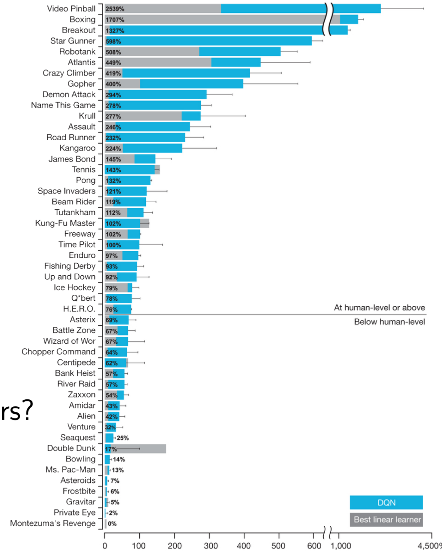
Nature 518, 529 (2015)

- Learning to play 49 different Atari 2600 games
- No knowledge of the goals/rules, just 84x84 pixel frames
- 60 frames per second, 50 million frames (38 days of game experience)
- Deep convolutional network with reinforcement: DQN (deep Q-network)
  - action-value function $Q^*(s,a) = \max_\pi \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots | s_t = s, a_t = a, \pi]$
  - maximum sum of rewards r_t discounted by γ at each timestep t, achievable by a behaviour policy $\pi = P(a|s)$, after making observation s and taking action a
- Tricks for scalability and performance:
  - experience replay (use past frames)
  - separate network to generate learning targets (iterative update of Q)
- Outperforms all previous algorithms, and professional human player on most games

Algorithm 1: deep Q-learning with experience replay.

Initialize replay memory D to capacity N
Initialize action-value function Q with random weights θ
Initialize target action-value function $\hat{Q}$ with weights $\theta^- = \theta$
For episode $= 1, M$ **do**
  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  For $t = 1$,T **do**
    With probability ε select a random action a_t
    otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D
    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \hat{Q}\left(\phi_{j+1}, a'; \theta^-\right) & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $\left(y_j - Q\left(\phi_j, a_j; \theta\right)\right)^2$ with respect to the network parameters θ
    Every C steps reset $\hat{Q} = Q$
  End For
End For

- What about Breakout or Space invaders?

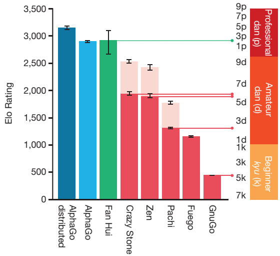


Google DeepMind: mastering Go

- Game of Go considered very challenging for AI
- Board games: can be solved with search tree of b^d possible sequences of moves (b = breadth [number of legal moves], d = depth [length of game])
- Chess: $b \approx 35$, $d \approx 80 \rightarrow$ go: $b \approx 250$, $d \approx 150$
- Reduction:
  - of depth by position evaluation (replace subtree by approximation that predicts outcome)
  - of breadth by sampling actions from probability distribution (policy $p(a|s)$) over possible moves a in position s
- 19×19 image, represented by CNN
- Supervised learning policy network from expert human moves, reinforcement learning policy network on self-play (adjusts policy towards winning the game), value network that predicts winner of games in self-play.

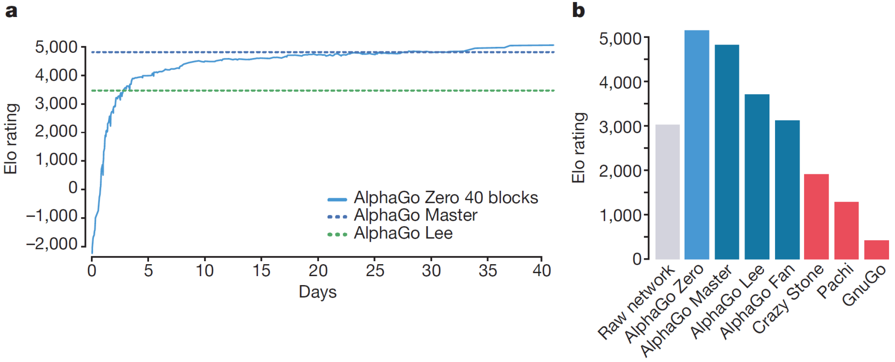
Nature 529, 484 (2016)

- AlphaGo: 40 search threads, simulations on 48 CPUs, policy and value networks on 8 GPUs. Distributed AlphaGo: 1020 CPUs, 176 GPUs
- AlphaGo won 494/495 games against other programs (and still 77% against Crazy Stone with four handicap stones)
  - Fan Hui: 2013/14/15 European champion
  - Distributed AlphaGo won 5–0
  - AlphaGo evaluated thousands of times fewer positions than Deep Blue (first chess computer to bit human world champion) ⇒ better position selection (policy network) and better evaluation (value network)

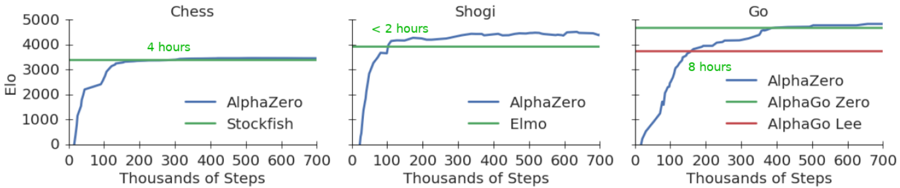


- Then played Lee Sedol (top Go play in the world over last decade) in March 2016 ⇒ won 4–1. AlphaGo given honorary professional ninth dan, considered to have “reach a level ‘close to the territory of divinity’ ”
- Ke Jie (Chinese world #1): “Bring it on!”. May 2017: 3–0 win for AlphaGo. New comment: “I feel like his game is more and more like the ‘Go god’. Really, it is brilliant”

- Learn from scratch, just from the rules and random moves
- Reinforcement learning from self-play, no human data/guidance
- Combined policy and value networks
- 4.9 million self-play games
- Beats AlphaGo Lee (several months of training) after just 36 hours
- Single machine with four TPU



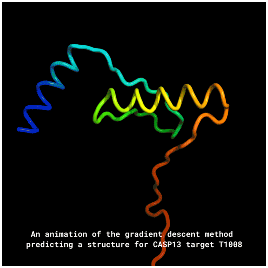
- Same philosophy as AlphaGo Zero, applied to chess, shogi and go
- Changes:
  - not just win/loss, but also draw or other outcomes
  - no additional training data from game symmetries
  - using always the latest network to generate self-play games rather than best one
  - tree search: 80k/70M for chess AlphaZero/Stockfish, 40k/35M for shogi AlphaZero/Elmo

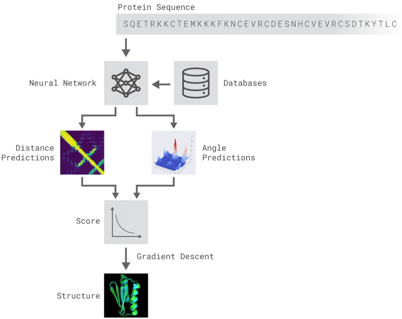


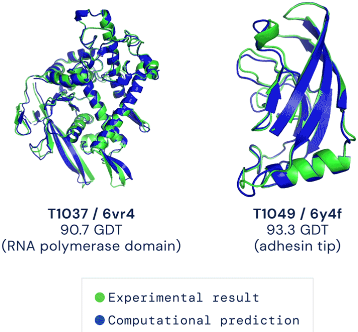
DeepMind AlphaFold

▸ Blog Dec 2018 ▸ AlphaFold2 Nov 2020

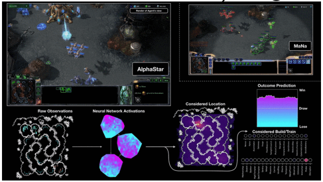
- Trying to tackle scientific problem
- Goal: predict 3D structure of protein based solely on genetic sequence
- Using DNN to predict
  - distances between pairs of amino acids
  - angles between chemical bonds
- Search DB to find matching existing substructures
- Also train a generative NN to invent new fragments
- Achieved best prediction ever



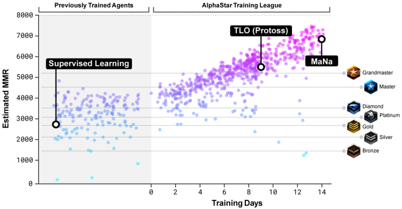


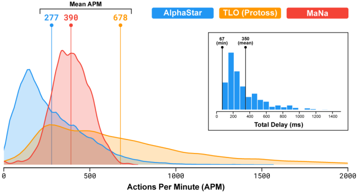


- Mastering real-time strategy game StarCraft II
- Challenges in game theory (no single best strategy), imperfect information (hidden parts of game), long term planning, real time (continuous flow of actions), large action space (many units/buildings)
- Using DNN trained
  - directly on raw data games
  - supervised learning on human games
  - reinforcement learning (continuous league)



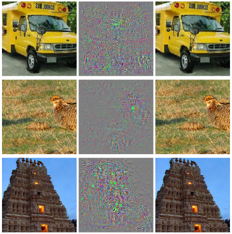
- DNN output: list of actions
- Trained for 14 days; each agent: up to 200 years of real-time play
- Runs on single desktop GPU
- Defeated 5–0 one of best pro-players





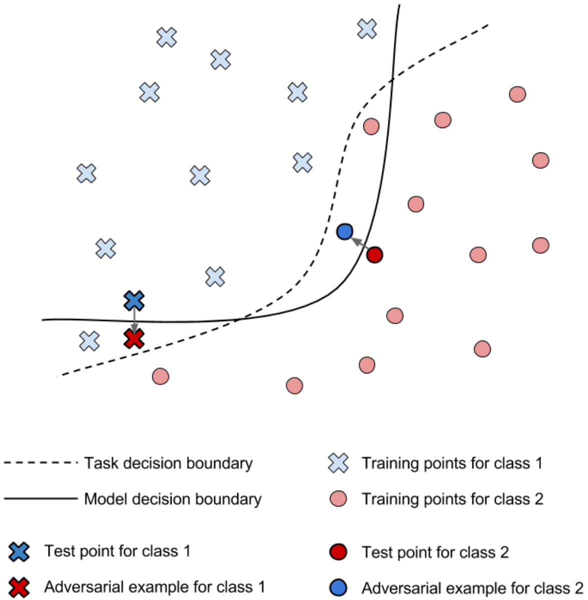
Deep networks: new results all the time

- Playing poker
  - Libratus (AI developed by Carnegie Mellon University) defeated four of the world's best professional poker players (Jan 2017)
  - After 120,000 hands of Heads-up, No-Limit Texas Hold'em, led the pros by a collective $1,766,250 in chips
  - Learned to bluff, and win with incomplete information and opponents' misinformation
- Lip reading ▸ arXiv:1611.05358 [cs.CV]
  - human professional: deciphers less than 25% of spoken words
  - CNN+LSTM trained on television news programs: 50%
- Limitation: adversarial attacks ▸ arXiv:1312.6199 [cs.CV]



- left: correctly classified image
- middle: difference between left image and adversarial image (x10)
- right: adversarial image, classified as ostrich

Adversarial attack: what is happening?



Breaking things easy, Papernot and Goodfellow, 2016

original semantic segmentation framework

Cisse et al arXiv:1707.05373

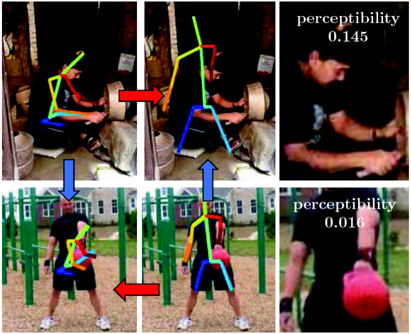
original semantic segmentation framework

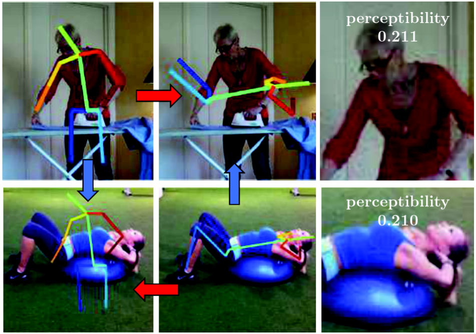
adversarial attack

Adversarial attacks

Cisse et al arXiv:1707.05373

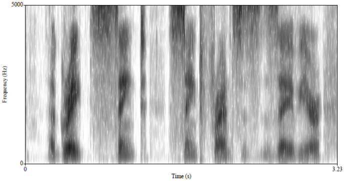
original semantic segmentation framework — adversarial attack — compromised semantic segmentation framework


perceptibility
0.145
perceptibility
0.016

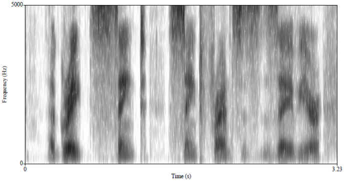

perceptibility
0.211
perceptibility
0.210

Adversarial attacks

Cisse et al arXiv:1707.05373



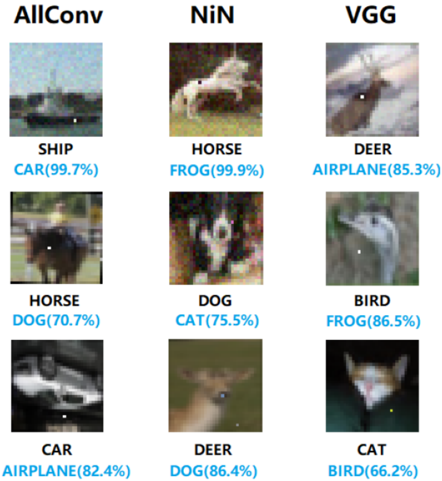
(a) a great saint saint francis zaviour

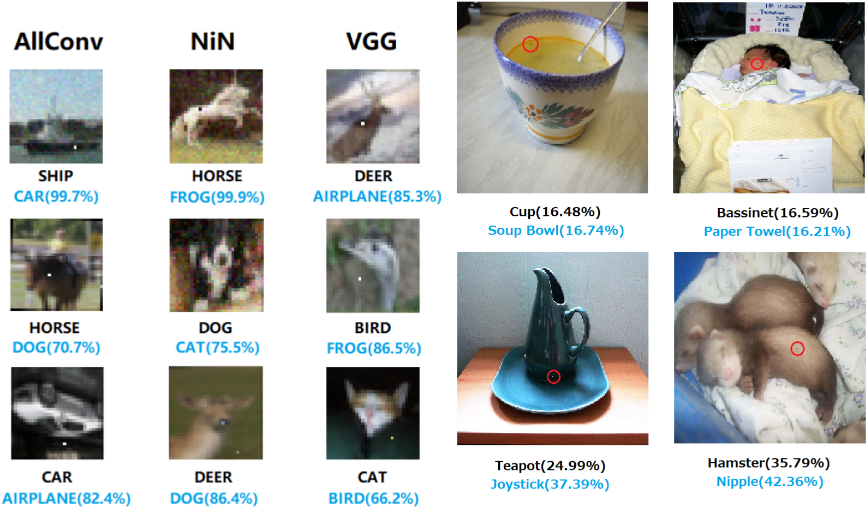


(b) i great sinkt shink t frimsuss avir

Figure 7: The model models' output for each of the spectrograms is located at the bottom of each spectrogram. The target transcription is: `A Great Saint Saint Francis Xavier.`

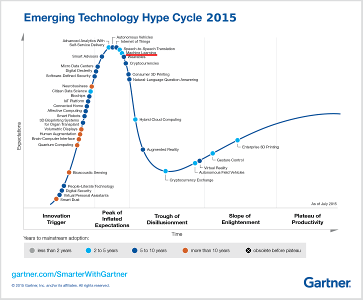
One-pixel attack



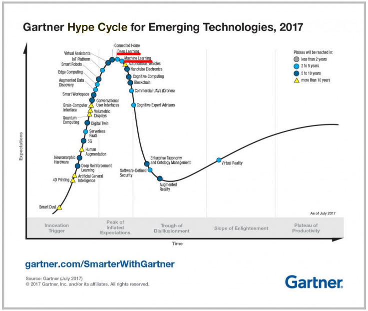


AllConv
NiN
VGG
SHIP
CAR(99.7%)
HORSE
FROG(99.9%)
DEER
AIRPLANE(85.3%)
HORSE
DOG(70.7%)
DOG
CAT(75.5%)
BIRD
FROG(86.5%)
CAR
AIRPLANE(82.4%)
DEER
DOG(86.4%)
CAT
BIRD(66.2%)
Cup(16.48%)
Soup Bowl(16.74%)
Bassinet(16.59%)
Paper Towel(16.21%)
Teapot(24.99%)
Joystick(37.39%)
Hamster(35.79%)
Nipple(42.36%)

Hype cycle

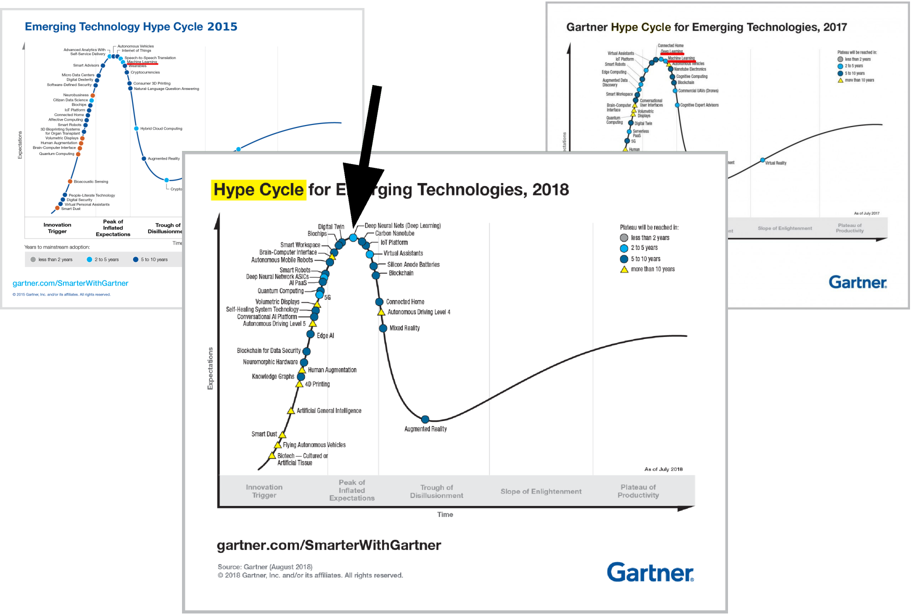


Emerging Technology Hype Cycle 2015
Expectations
Advanced Analytics With Self-Service Delivery
Autonomous Vehicles
Internet of Things
Speech-to-Speech Translation
Machine Learning
Wearables
Smart Advisors
Cryptocurrencies
Micro Data Centers
Digital Dexterity
Software-Defined Security
Consumer 3D Printing
Natural-Language Question Answering
Neurobusiness
Citizen Data Science
Biochips
IoT Platform
Connected Home
Affective Computing
Smart Robots
3D Bioprinting Systems for Organ Transplant
Volumetric Displays
Human Augmentation
Brain-Computer Interface
Quantum Computing
Hybrid Cloud Computing
Augmented Reality
Enterprise 3D Printing
Gesture Control
Virtual Reality
Autonomous Field Vehicles
Cryptocurrency Exchange
Bioacoustic Sensing
People-Literate Technology
Digital Security
Virtual Personal Assistants
Smart Dust
As of July 2015
Innovation Trigger
Peak of Inflated Expectations
Trough of Disillusionment
Slope of Enlightenment
Plateau of Productivity
Time
Years to mainstream adoption:
less than 2 years
2 to 5 years
5 to 10 years
more than 10 years
obsolete before plateau
gartner.com/SmarterWithGartner
© 2015 Gartner, Inc. and/or its affiliates. All rights reserved.
Gartner



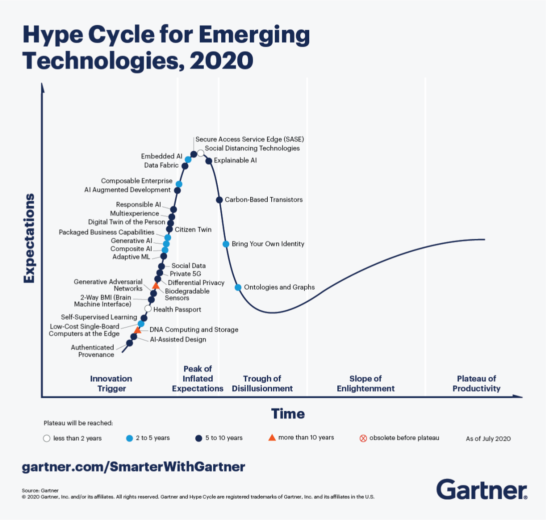
Gartner Hype Cycle for Emerging Technologies, 2017
Expectations
Connected Home
Deep Learning
Machine Learning
Autonomous Vehicles
Nanotube Electronics
Cognitive Computing
Blockchain
Commercial UAVs (Drones)
Cognitive Expert Advisors
Virtual Assistants
IoT Platform
Smart Robots
Edge Computing
Augmented Data Discovery
Smart Workspace
Conversational User Interfaces
Brain-Computer Interface
Volumetric Displays
Quantum Computing
Digital Twin
Serverless PaaS
5G
Human Augmentation
Neuromorphic Hardware
Deep Reinforcement Learning
Artificial General Intelligence
4D Printing
Smart Dust
Enterprise Taxonomy and Ontology Management
Software-Defined Security
Augmented Reality
Virtual Reality
Plateau will be reached in:
less than 2 years
2 to 5 years
5 to 10 years
more than 10 years
As of July 2017
Innovation Trigger
Peak of Inflated Expectations
Trough of Disillusionment
Slope of Enlightenment
Plateau of Productivity
Time
gartner.com/SmarterWithGartner
Source: Gartner (July 2017)
© 2017 Gartner, Inc. and/or its affiliates. All rights reserved.
Gartner

Hype cycle



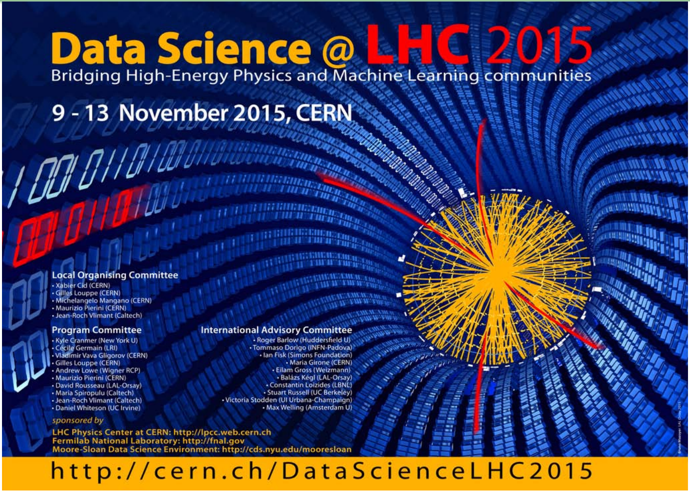
Emerging Technology Hype Cycle 2015
Innovation Trigger
Peak of Inflated Expectations
Trough of Disillusionment
Expectations
Years to mainstream adoption:
less than 2 years
2 to 5 years
5 to 10 years
Machine Learning
gartner.com/SmarterWithGartner
© 2015 Gartner, Inc. and/or its affiliates. All rights reserved.
Gartner Hype Cycle for Emerging Technologies, 2017
Deep Learning
Machine Learning
Plateau will be reached in:
less than 2 years
2 to 5 years
5 to 10 years
more than 10 years
Virtual Reality
As of July 2017
Slope of Enlightenment
Plateau of Productivity
Gartner
Hype Cycle for Emerging Technologies, 2018
Digital Twin
Biochips
Deep Neural Nets (Deep Learning)
Carbon Nanotube
IoT Platform
Smart Workspace
Brain-Computer Interface
Autonomous Mobile Robots
Virtual Assistants
Silicon Anode Batteries
Blockchain
Smart Robots
Deep Neural Network ASICs
AI PaaS
Quantum Computing
5G
Volumetric Displays
Self-Healing System Technology
Conversational AI Platform
Autonomous Driving Level 5
Connected Home
Autonomous Driving Level 4
Mixed Reality
Edge AI
Blockchain for Data Security
Neuromorphic Hardware
Knowledge Graphs
Human Augmentation
4D Printing
Artificial General Intelligence
Smart Dust
Flying Autonomous Vehicles
Biotech — Cultured or Artificial Tissue
Augmented Reality
Expectations
Plateau will be reached in:
less than 2 years
2 to 5 years
5 to 10 years
more than 10 years
As of July 2018
Innovation Trigger
Peak of Inflated Expectations
Trough of Disillusionment
Slope of Enlightenment
Plateau of Productivity
Time
gartner.com/SmarterWithGartner
Source: Gartner (August 2018)
© 2018 Gartner, Inc. and/or its affiliates. All rights reserved.
Gartner

Hype cycle

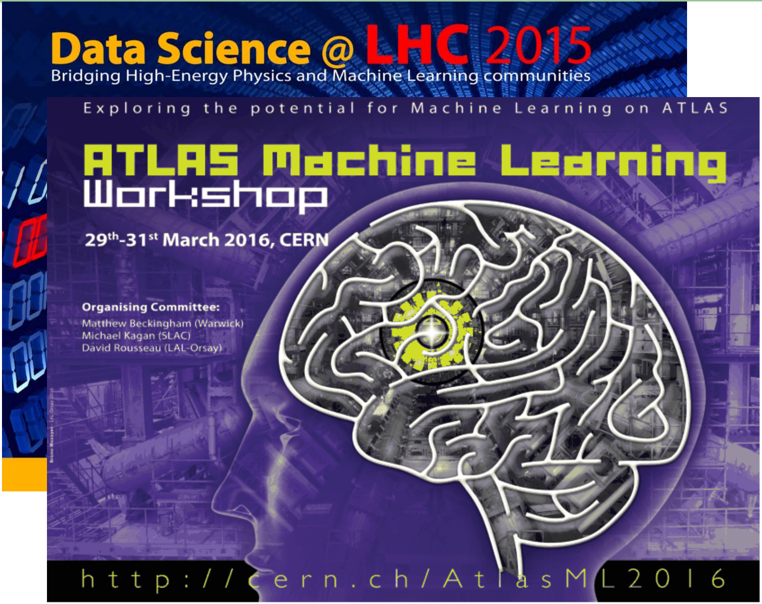


- No deep learning/ML anymore (since 2019)
- Instead, all sorts of "AI" and ML-driven systems
  - explainable AI
  - embedded AI
  - generative AI / generative adversarial networks
  - adaptive ML
  - self-supervised learning
  - AI-assisted design

Machine learning and particle physics

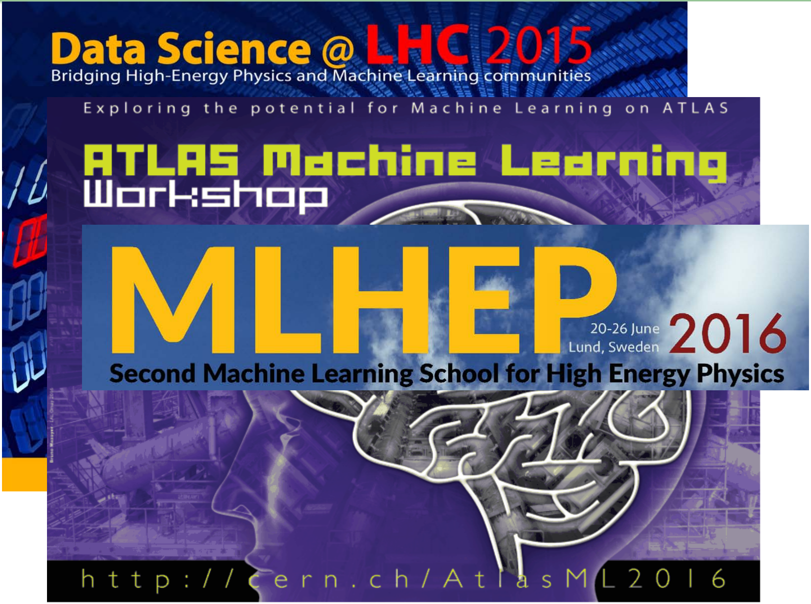


http://opendata.cern.ch

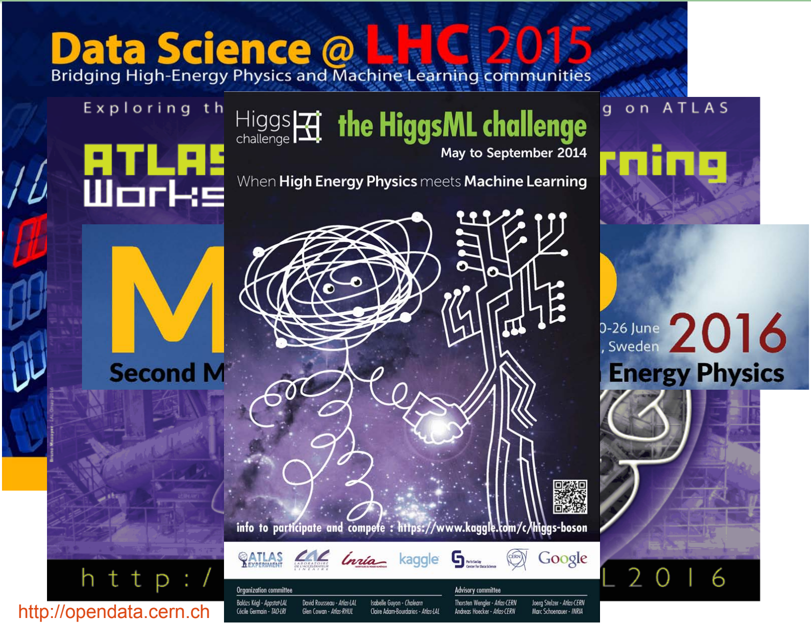


http://opendata.cern.ch

Machine learning and particle physics

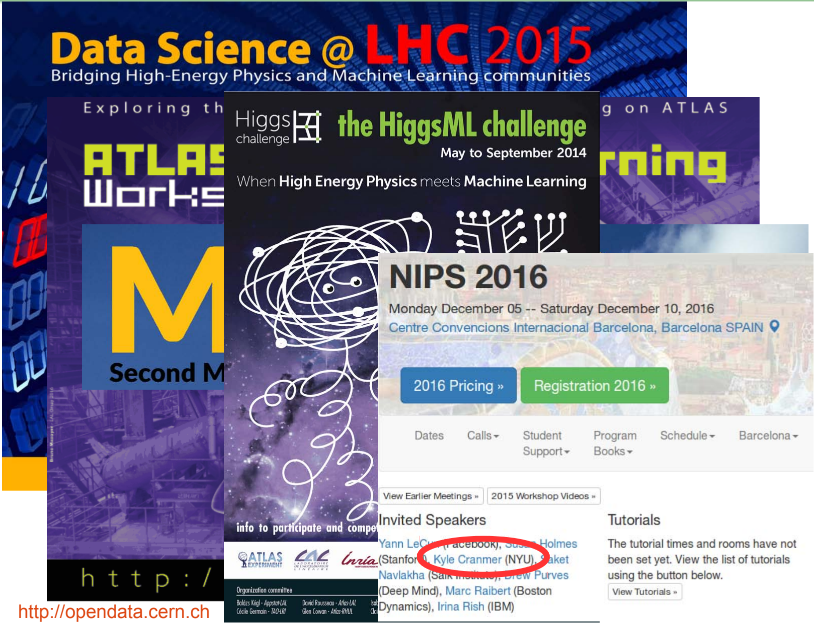


http://opendata.cern.ch



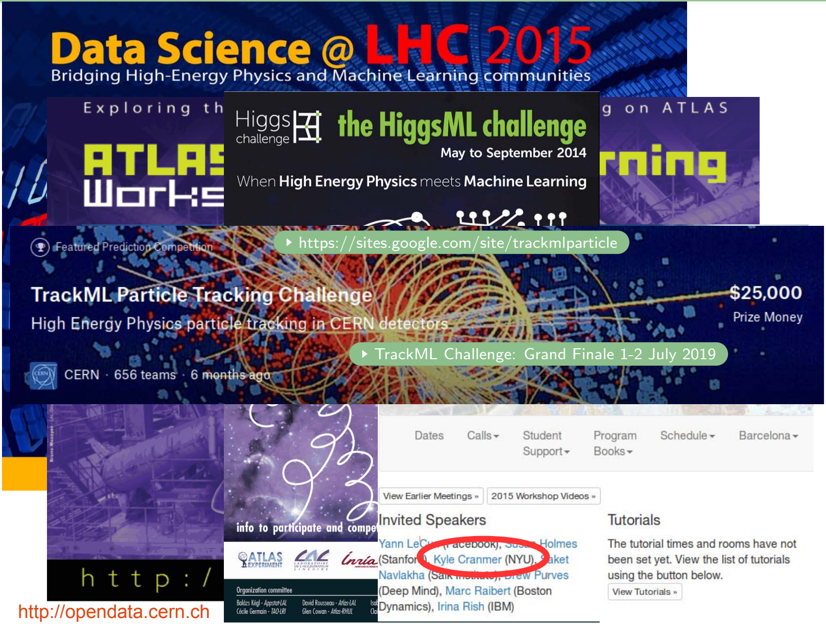
http://opendata.cern.ch

Machine learning and particle physics



http://opendata.cern.ch

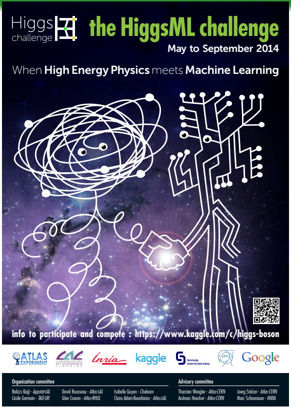
Machine learning and particle physics

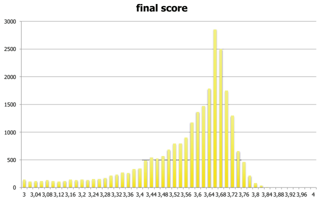


https://sites.google.com/site/trackmlparticle

TrackML Challenge: Grand Finale 1-2 July 2019

http://opendata.cern.ch



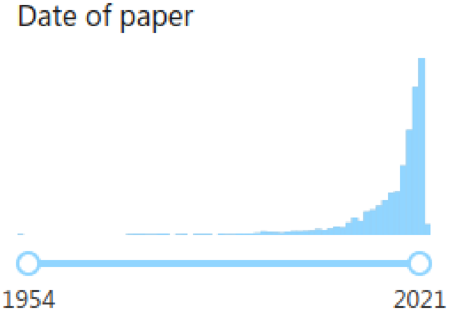


HiggsML challenge

- Put ATLAS Monte Carlo samples on the web ($H \to \tau\tau$ analysis)
- Compete for best signal–bkg separation
- 1785 teams (most popular challenge ever)
- 35772 uploaded solutions
- See ▸ Kaggle web site and ▸ more information

#	Δrank	Team Name ‡ model uploaded * in the money		Score	Entries	Last Submission UTC (Best – Last Submission)
1	↑1	Gábor Melis ‡ *	7000$	3.80581	110	Sun, 14 Sep 2014 09:10:04 (-0h)
2	↑1	Tim Salimans ‡ *	4000$	3.78913	57	Mon, 15 Sep 2014 23:49:02 (-40.6d)
3	↑1	nhlx5haze ‡ *	2000$	3.78682	254	Mon, 15 Sep 2014 16:50:01 (-76.3d)
4	↑38	ChoKo Team		3.77526	216	Mon, 15 Sep 2014 15:21:36 (-42.1h)
5	↑35	cheng chen		3.77384	21	Mon, 15 Sep 2014 23:29:29 (-0h)
6	↑16	quantify		3.77086	8	Mon, 15 Sep 2014 16:12:48 (-7.3h)
7	↑1	Stanislav Semenov & Co (HSE Yandex)		3.76211	68	Mon, 15 Sep 2014 20:19:03
8	↓7	Luboš Motl's team		3.76050	589	Mon, 15 Sep 2014 08:38:49 (-1.6h)
9	↑8	Roberto-UCIIIM		3.75864	292	Mon, 15 Sep 2014 23:44:42 (-44d)
10	↑2	Davut & Josef		3.75838	161	Mon, 15 Sep 2014 23:24:32 (-4.5d)
45	↑5	crowwork ‡	HEP meets ML award Free trip to CERN	3.71885	94	Mon, 15 Sep 2014 23:45:00 (-5.1d)
782	↓149	Eckhard	TMVA expert, with TMVA improvements	3.49945	29	Mon, 15 Sep 2014 07:26:13 (-46.1h)
991	↑4	Rem.		3.20423	2	Mon, 16 Jun 2014 21:53:43 (-30.4h)
		simple TMVA boosted trees		3.19956		

Machine learning and particle physics



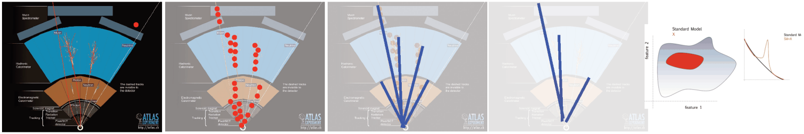
https://inspirehep.net/literature?q=machine learning or deep learning or multivariate

Up-to-date comprehensive review of papers

https://github.com/iml-wg/HEPML-LivingReview

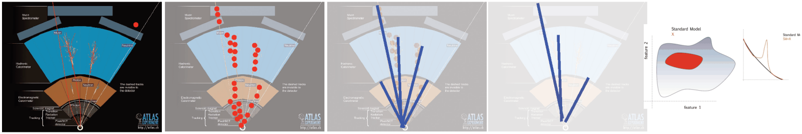
Machine learning and particle physics

Raw	Sparsified	Reco	Select	Physics	Ana
1e7	1e4	100-ish*	50	10	1



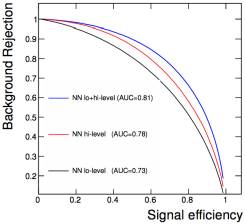
- Reduce data dimensionality to allow analysis

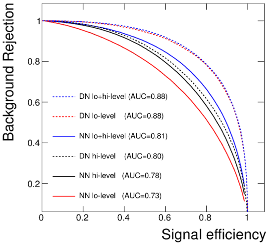
Raw	Sparsified	Reco	Select	Physics	Ana
1e7	1e4	100-ish*	50	10	1



- Reduce data dimensionality to allow analysis
- Going to lower level features

arXiv:1402.4735

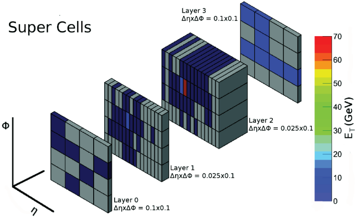


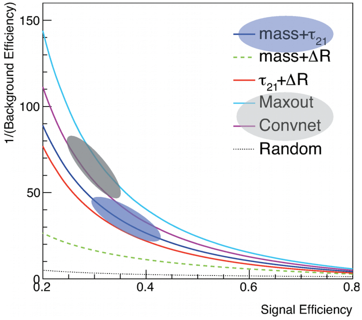


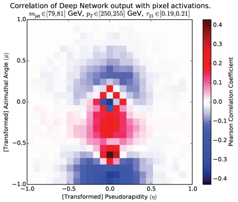
- Transforming inputs into images

arXiv:1511.05190

Super Cells

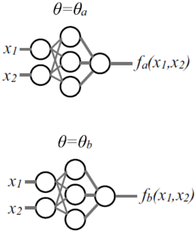


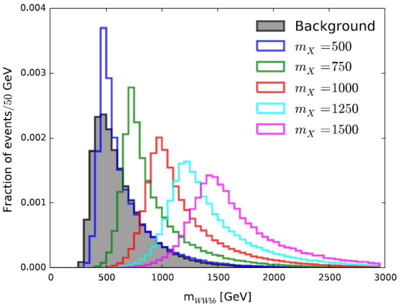




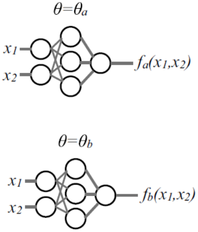
Machine learning and particle physics

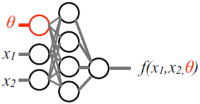
- Looking for new physics scenario with unknown mass
  $\Rightarrow$ one NN for each mass point

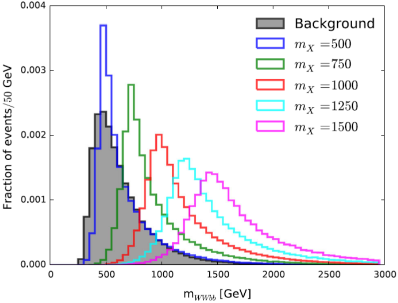




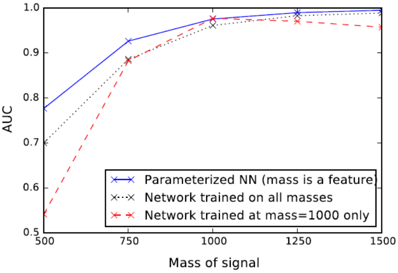
- Looking for new physics scenario with unknown mass
  $\Rightarrow$ one NN for each mass point







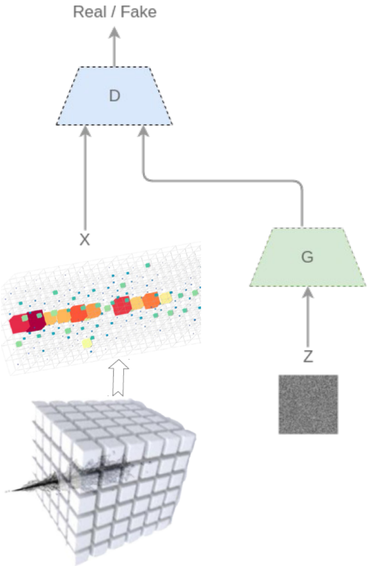
- Parameterised NN ‣ EPJC (2016) 76:235 ‣ arXiv:1601.07913
  - mass as training parameter
  - as good as dedicated training
  - generalises better



Machine learning and particle physics

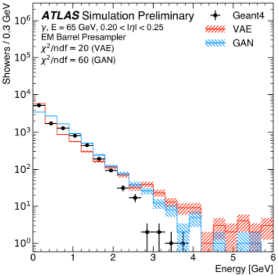
Fast simulation with generative models

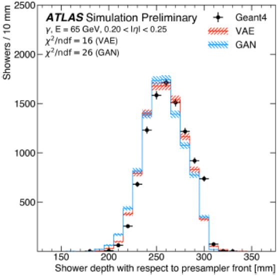
- Heavy CPU cost of simulation ($>$ 50% of grid resources)
  - MC stats becoming limiting factor in analyses
- Replace "full simulation" with approximation
  - already routinely done, using parameterisation of showers or library of pre-simulated objects

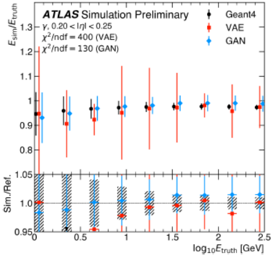


Fast simulation with generative models

ATL-SOFT-PUB-2018-001 ATLAS-SIM-2019-004

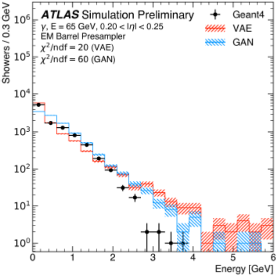


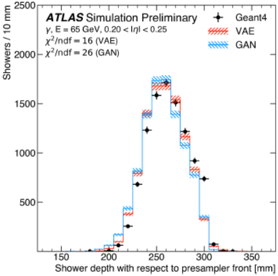


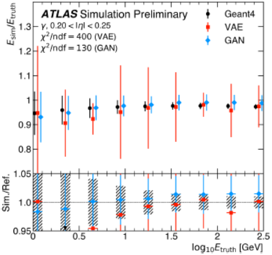


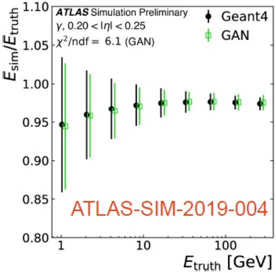
Fast simulation with generative models

ATL-SOFT-PUB-2018-001 ATLAS-SIM-2019-004

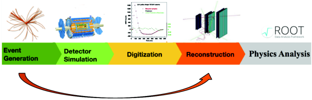








- $<$ 1 ms instead of 10 s per object!
- $\times$100 gain on complete event
- Still some work to be done



Anomaly detection: looking for new physics

▸ thesis

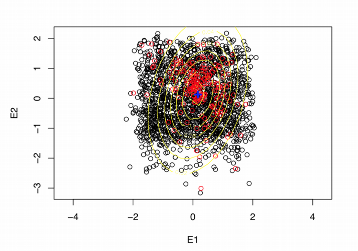
- Learn background (SM) properties
- Flag deviations from prediction without knowing anything about specific new physics scenario

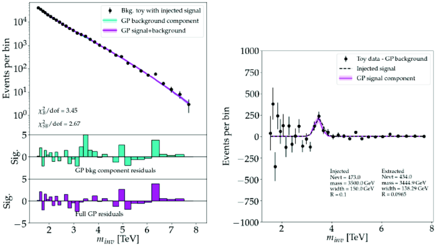
Penalised anomaly detection

- based on Gaussian mixture model
- f_S and f_B: finite sums of Gaussians
- semi-supervised training
- penalty term in LH to select variables

Gaussian processes

- Learn background with GP instead of parametric model
- Compare data to new GP: background model+signal
- Returns parameters of "peak"





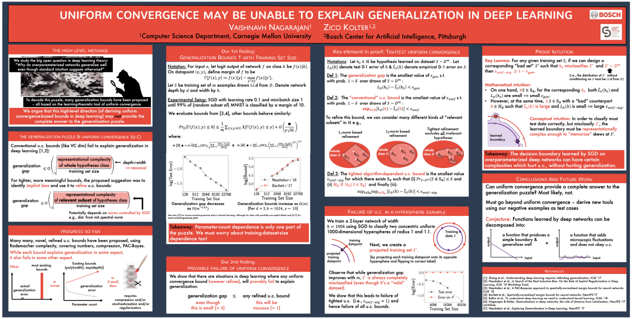
Deep learning: looking forward

- Very active field of research in machine learning and artificial intelligence
  - not just at universities (Google, Facebook, Microsoft, NVIDIA, etc. . .)
- Training with curriculum:
  - what humans do over 20 years, or even a lifetime
  - learn different concepts at different times
  - solve easier or smoothed version first, and gradually consider less smoothing
  - exploit previously learned concepts to ease learning of new abstractions
- Influence learning dynamics can have big impact:
  - order and selection of examples matters
  - choose which examples to present first, to guide training and possibly increase learning speed (called shaping in animal training)
- Combination of deep learning and reinforcement learning
  - still in its infancy, but already impressive results
- Domain adaptation and adversarial training
  - e.g. train in parallel network that produces difficult examples
  - learn discrimination (s vs. b) and difference between training and application samples (e.g. Monte Carlo simulation and real data)

NeurIPS 2019 in Vancouver

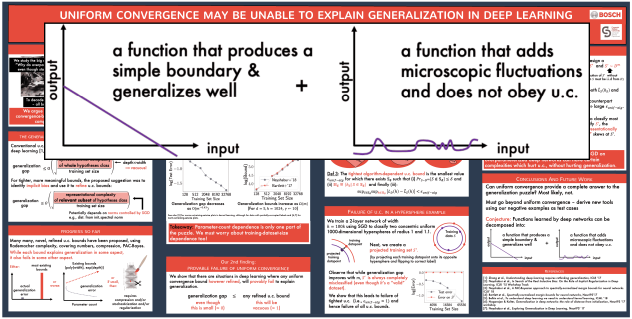
NEURAL INFORMATION
PROCESSING SYSTEMS

NeurIPS2019: generalisation in deep learning



- Actually works surprisingly well
- Over-parameterised DNN should overfit but don't: why?
- Neural tangent kernel (NTK): helps thinking in infinite-width limit. But can do better in reality
- Robustness to adversarial attacks
- Start with large learning rate to learn easy features, then decrease to learn low noise, hard-to-fit patterns

NeurIPS2019: generalisation in deep learning

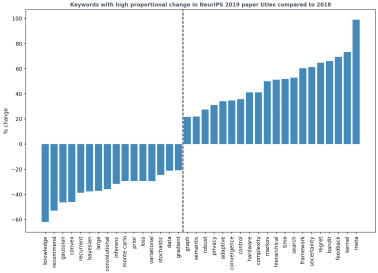


- Actually works surprisingly well
- Over-parameterised DNN should overfit but don't: why?
- Neural tangent kernel (NTK): helps thinking in infinite-width limit. But can do better in reality
- Robustness to adversarial attacks
- Start with large learning rate to learn easy features, then decrease to learn low noise, hard-to-fit patterns

NeurIPS2019: Take away trends

▸ © Chip Huyen

- More people active with neurosciences: ML to understand NS, and NS to understand ML
- Meta learning (learning to learn)
- Reinforcement learning is gaining ground. Other keywords: bandit, feedback, regret, control



- Attributing uncertainty to ML algorithms (often with Bayesian methods in deep learning)
- Generative models still popular
- Hardware keyword on the rise, signaling more hardware-aware algorithms: hardware = bottleneck?
- "Recurrent and convolutional neural networks are literally so last year"
- Growing consciousness of potential impact on society

NeurIPS2019: ML and the Physical Sciences

- 91 short papers accepted for poster presentation (6 selected for talks)
- 70 "digital acceptance" papers (above rejection threshold but beyond capacity)
- 228 referees ▸ web site (incl. videos)
- 5 invited speakers:
  - Alan Aspuru-Guzik: *Recent progress in ML for chemistry: SELFIES, inverse design of drug candidates and materials, and Bayesian algorithms for self-driving laboratories*
  - Yasaman Bahri: *Towards an understanding of wide, deep neural networks*
  - Katie Bouman: Cannot find title, about Event Horizon Telescope imaging technique
  - Bernhard Schölkopf: *Causality and Exoplanets*
  - Maria Schuld: *Innovating machine learning with near-term quantum computing*
  - Lenka Zdeborova: *Understanding machine learning via exactly solvable statistical physics models*

Suggested areas

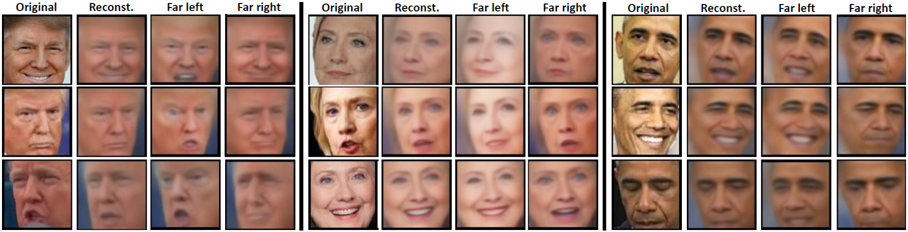
- Application of machine and deep learning to physical sciences
- Generative models
- Likelihood-free inference
- Variational inference
- Simulation-based models
- Implicit models
- Probabilistic models
- Model interpretability
- Approximate Bayesian computation
- Strategies for incorporating prior scientific knowledge into machine learning algorithms
- Experimental design
- Any other area related to the subject of the workshop

NeurIPS2019: AI and society

- ML achieves super-human performance for well-designed problems, or games with score $\Rightarrow$ where one can define a proper loss function or reward
- Scale to "real" problems?
  - explainability
  - causality
  - "moral" stand
  - culture, art
- Many advances in medical imaging, modelling of various phenomena, supernova analysis or LHC physics, but issues with:
  - out-of-distribution generalisation
  - scalability of computing resources, carbon footprint
  - reliability
  - decision bias (gender, race, etc.)
- Workshops/socials on Fairness & ethics, AI for Good, Tackling Climate Change with ML, AI for Humanitarian Assistance and Disaster Response, Safety and Robustness in Decision-making, ...
- Importance of personal decisions

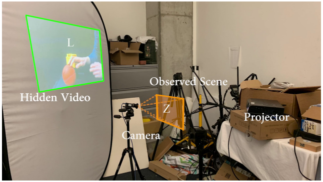
NeurIPS2019: Hidden information

- Predicting the Politics of an Image Using Webly Supervised Data



- Computational Mirrors: Blind Inverse Light Transport by Deep Matrix Factorization

(edited video)



Conclusion

- Many techniques and tools exist to achieve optimal discrimination
- (Un)fortunately, no one method can be shown to outperform the others in all cases
- One should try several and pick the best one for any given problem
- Latest machine learning algorithms (e.g. deep networks) require enormous hyperparameter space optimisation...
- Machine learning and multivariate techniques are at work in your everyday life without your knowning and can easily outsmart you for many tasks
- Try this to convince yourself ▸ http://www.phi-t.de/mousegame/mousegame_en.html

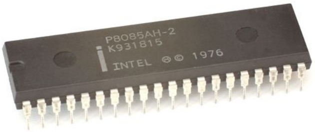
Upcoming reference book (in about six months)

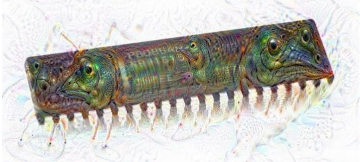
Artificial Intelligence for High Energy Physics ▸ https://doi.org/10.1142/12200

Deep networks and art

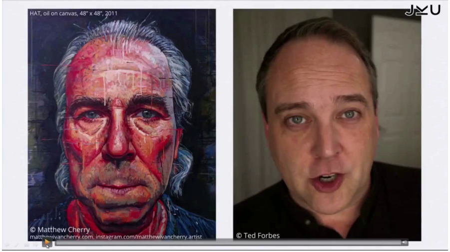
- Learning a style ‣ arXiv:1508.06576 [cs.CV] ‣ Neural-style

- Computer dreams ‣ Google original ‣ deepdream

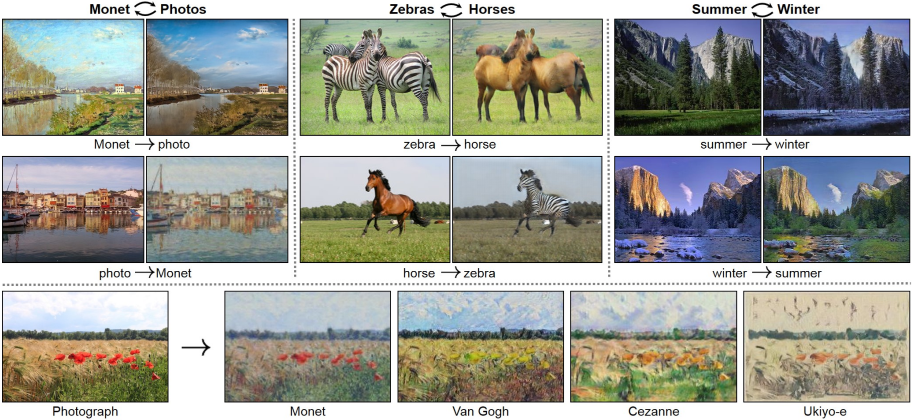




- Face Style ‣ http://facestyle.org ‣ http://dcgi.fel.cvut.cz/home/sykorad/facestyle.html



Summary

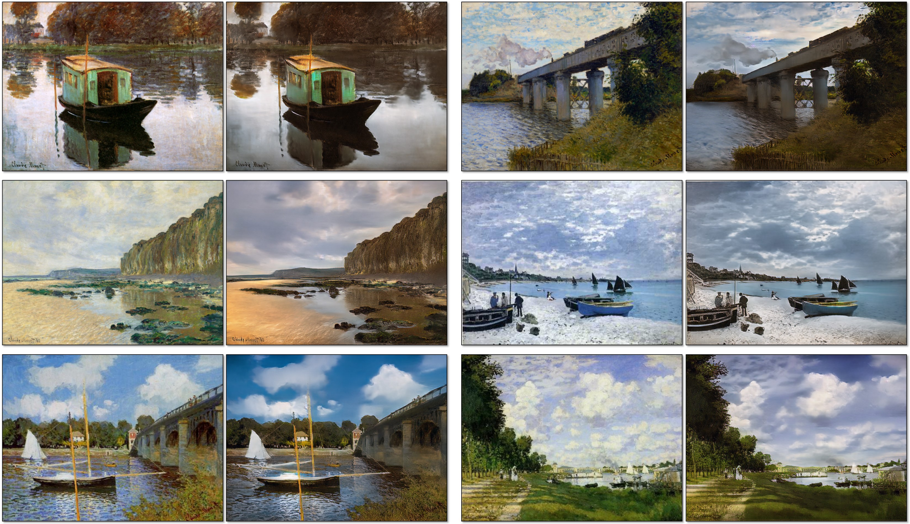

Monet ⇄ Photos
Zebras ⇄ Horses
Summer ⇄ Winter
Monet → photo
zebra → horse
summer → winter
photo → Monet
horse → zebra
winter → summer
Photograph
Monet
Van Gogh
Cezanne
Ukiyo-e

CycleGAN

arXiv:1703.10593 github

From Monnet to photograph

Input | Output | Input | Output



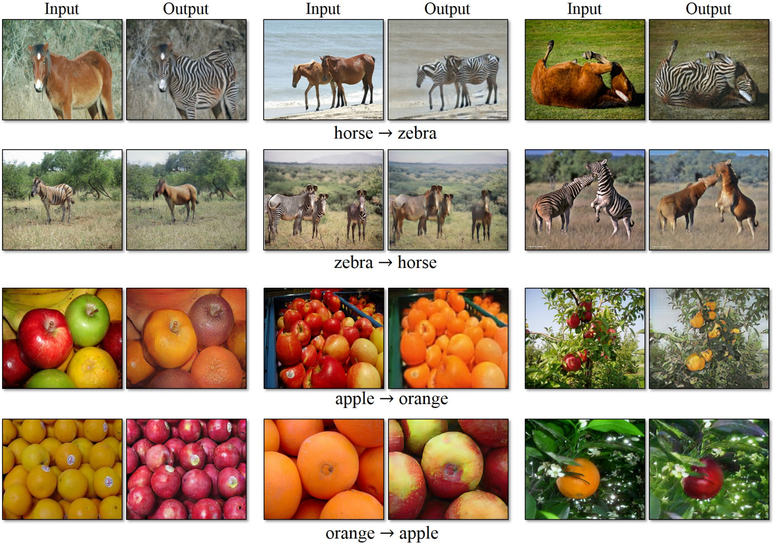
Style transfer

CycleGAN

arXiv:1703.10593 github

CENTRE DE PHYSIQUE DES PARTICULES DE MARSEILLE CPPM

Object transfiguration



Season transfer

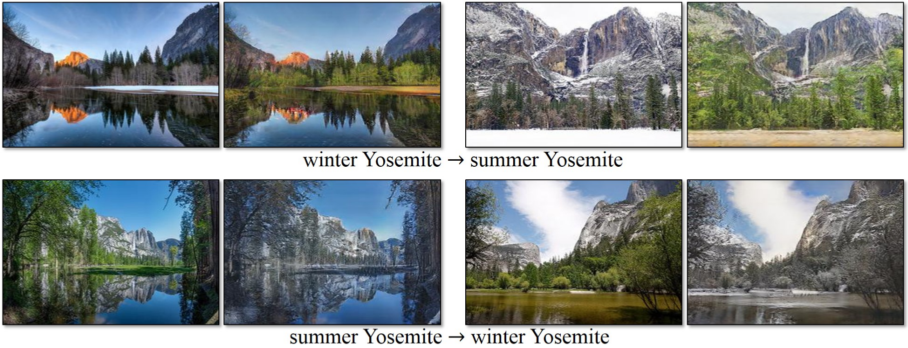


Photo enhancement

Failure 🙂

References III: neural networks

- C.M. Bishop, *Pattern Recognition and Machine Learning*, Springer, New York, 2007
- M. Minsky and S. Papert, *Perceptrons*, MIT Press, Cambridge, 1969
- W.S. McCulloch & W. Pitts, "A logical calculus of the ideas immanent in nervous activity", Bulletin of Mathematical Biophysics, 5, 115-137, 1943
- F. Rosenblatt, "The Perceptron: A Probabilistic Model for Information Storage & Organization in the Brain", Psychological Review, 65, pp. 386-408, 1958
- D.E.Rumelhart et al., "Learning representations by back-propagating errors", Nature vol. 323, p. 533, 1986
- K.Hornik et al., "Multilayer Feedforward Networks are Universal Approximators", Neural Networks, Vol. 2, pp 359-366, 1989
- Y. LeCun, L. Bottou, G. Orr and K. Muller, "Efficient BackProp", in *Neural Networks: Tricks of the trade*, Orr, G. and Muller K. (Eds), Springer, 1998

References IV: deep neural networks

- Q.V. Le et al., "Building High-level Features Using Large Scale Unsupervised Learning", in *Proceedings of the 29th International Conference on Machine Learning*, Edinburgh, Scotland, UK, 2012 ▸ http://research.google.com/pubs/pub38115.html
- Y. Bengio, P. Lamblin, D. Popovici and H. Larochelle, "Greedy Layer-Wise Training of Deep Networks", in *Advances in Neural Information Processing Systems 19* (NIPS'06), pages 153–160, MIT Press 2007
- M.A. Ranzato, C. Poultney, S. Chopra and Y. LeCun, in J. Platt et al., "Efficient Learning of Sparse Representations with an Energy-Based Model", in *Advances in Neural Information Processing Systems 19* (NIPS'06), pages 1137–1144, MIT Press, 2007
- Y. Bengio, "Learning deep architectures for AI", *Foundations and Trends in Machine Learning*, Vol. 2, No. 1 (2009) 1–127. Also book at ▸ Now Publishers ▸ http://www.iro.umontreal.ca/ lisa/publications2/index.php/publications/show/239
- I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*, MIT Press, 2016 ▸ http://www.deeplearningbook.org
- G. Carleo et al., "Machine learning and the physical sciences" ▸ Rev. Mod. Phys. 91, 045002 (2019) ▸ arXiv:1903.10563