Machine learning (part II)

with a high energy physicist bias

Yann Coadou

CPPM Marseille

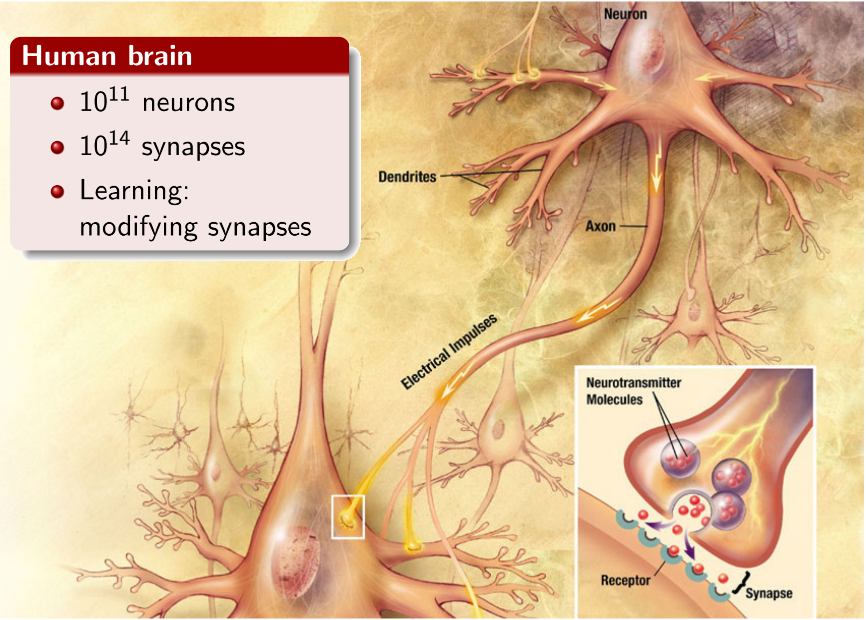
Online edition, 8 September 2021

Outline

Neural networks

Human brain

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

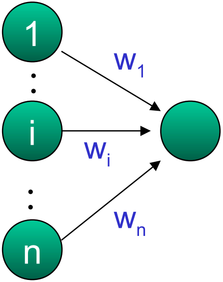


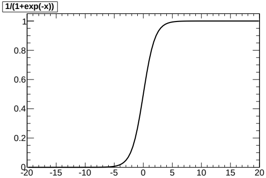
Brief history of artificial neural networks

- 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

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

Neural networks

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 $|y(x) - f(x)| < \varepsilon$.

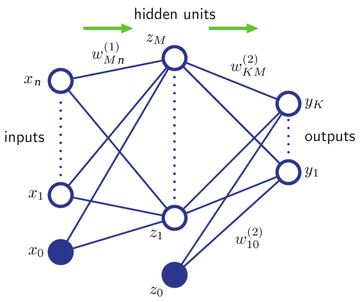
Interpretation

- 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

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



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

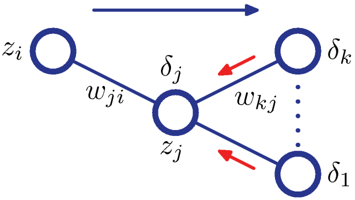
Backpropagation

- 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:

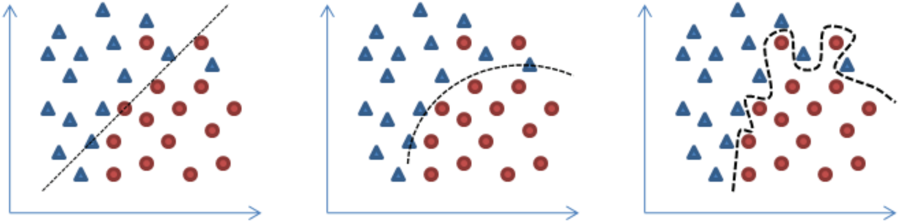
$$\frac{dE}{dw_{ji}} = \delta_j z_i, \text{ where}$$

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

- Hence errors are propagated backwards

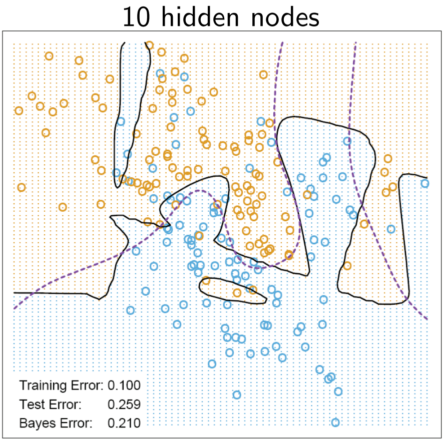


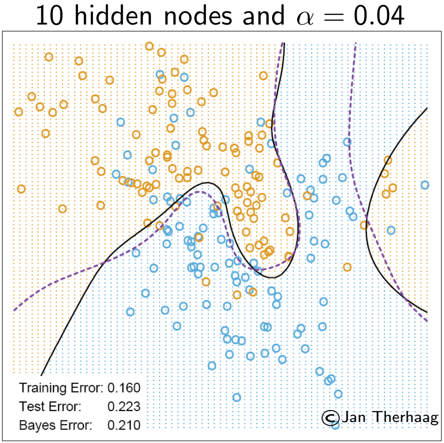
Neural network overtraining



- 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. For instance L2-regularisation:

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

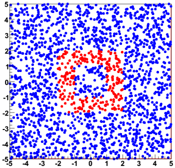


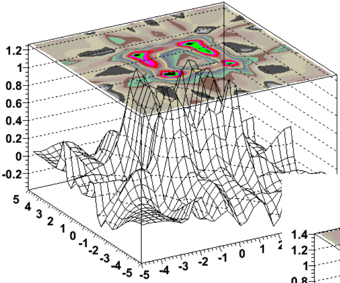


- Much less overfitting, better generalisation properties

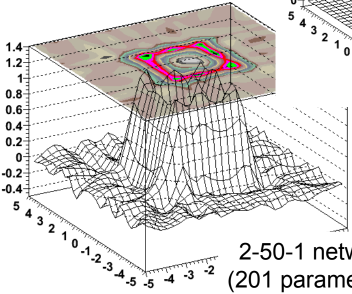
- Preprocess data:
  - if relevant, provide e.g. x/y instead of x and y
  - subtract the mean because the sigmoid derivative becomes negligible very fast (so, input mean close to 0)
  - normalise variances (close to 1)
  - shuffle training sample (order matters in online training)
- Initial random weights should be small to avoid saturation
- 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

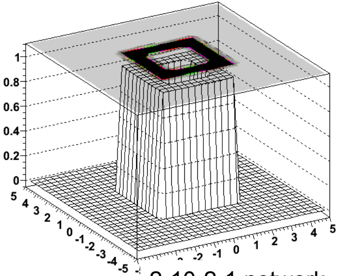




2-20-1 network
(81 parameters)



2-50-1 network
(201 parameters)



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

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 (esp. visual cortex)
- 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.

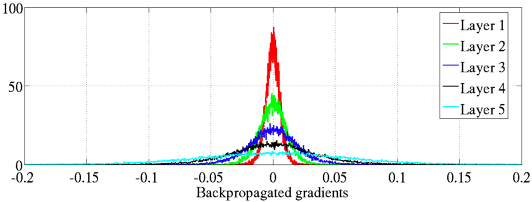
Deep learning revolution

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 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

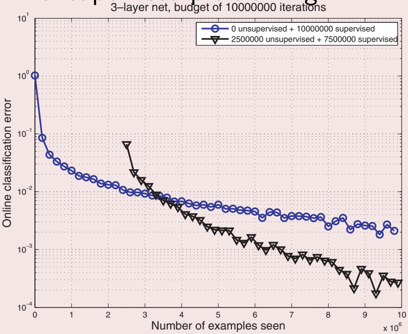
Breakthroughs around 2006 (Bengio, Hinton, LeCun)

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

Why does unsupervised training work?

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

Optimisation hypothesis

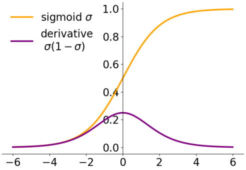
- Training one layer at a time scales well
- Backpropagation from sensible features
- Better local minimum than random initialisation, local search around it

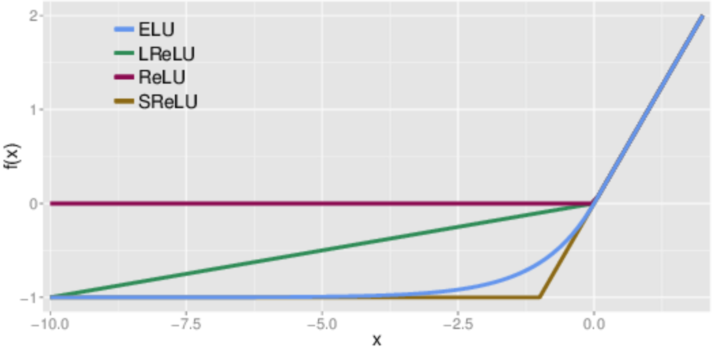
Overfitting/regularisation hypothesis

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

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.

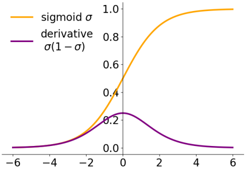




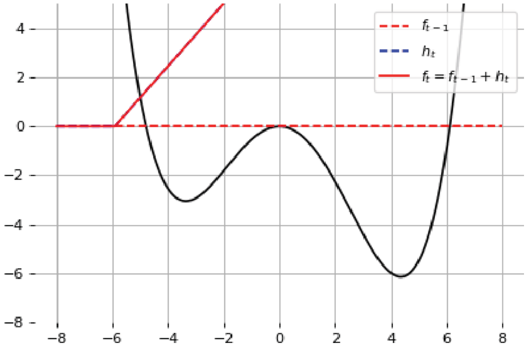
arXiv: 1511.07289

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

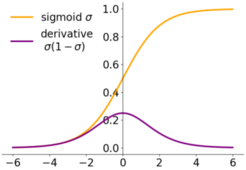


- Other variants: leaky ReLU, shifted ReLU (SReLU), exponential linear unit (ELU), etc.

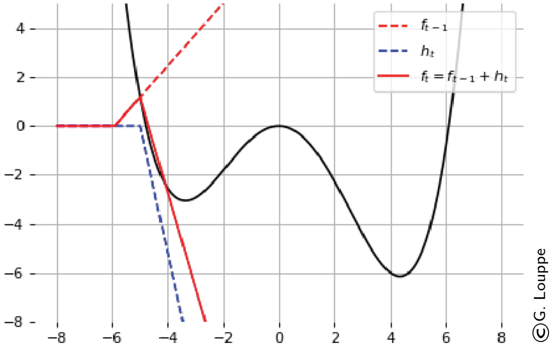


© G. Louppe

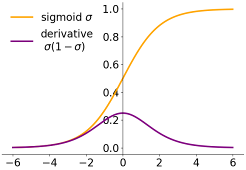
- One of reasons for vanishing gradient: sigmoid activation
  - tiny non-varying derivative away from zero
- Solution: non-saturating function
- Simplest case: rectified linear unit ReLU



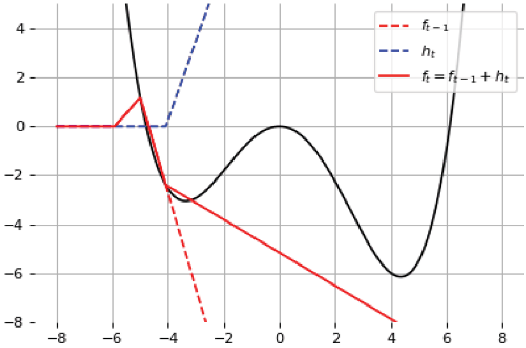
- Other variants: leaky ReLU, shifted ReLU (SReLU), exponential linear unit (ELU), etc.



- 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

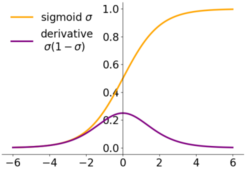


- Other variants: leaky ReLU, shifted ReLU (SReLU), exponential linear unit (ELU), etc.

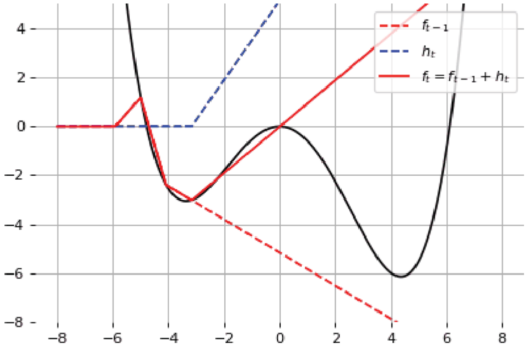


©G. Louppe

Rectified linear unit ReLU

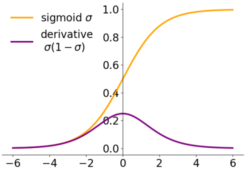


- One of reasons for vanishing gradient: sigmoid activation
  - tiny non-varying derivative away from zero
- Solution: non-saturating function
- Simplest case: rectified linear unit ReLU
- Other variants: leaky ReLU, shifted ReLU (SReLU), exponential linear unit (ELU), etc.

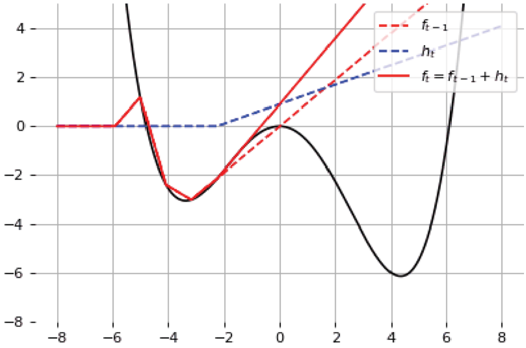


©G. Louppe

- 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



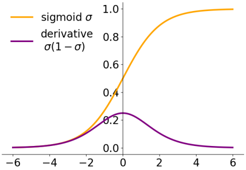
- Other variants: leaky ReLU, shifted ReLU (SReLU), exponential linear unit (ELU), etc.



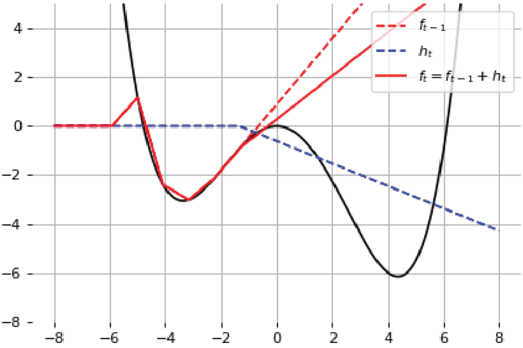
©G. Louppe

Rectified linear unit ReLU

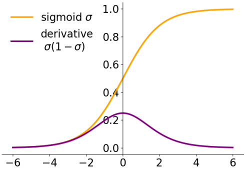
- One of reasons for vanishing gradient: sigmoid activation
  - tiny non-varying derivative away from zero
- Solution: non-saturating function
- Simplest case: rectified linear unit ReLU



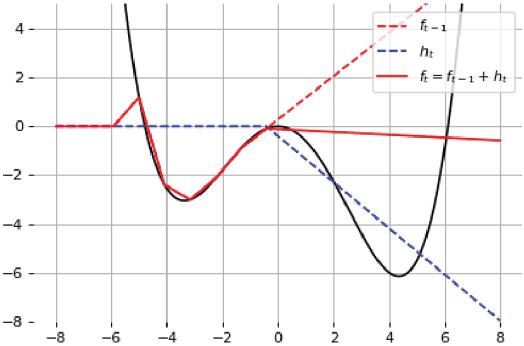
- Other variants: leaky ReLU, shifted ReLU (SReLU), exponential linear unit (ELU), etc.



©G. Louppe



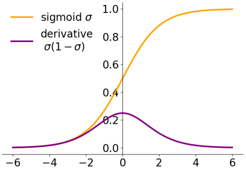
- One of reasons for vanishing gradient: sigmoid activation
  - tiny non-varying derivative away from zero
- Solution: non-saturating function
- Simplest case: rectified linear unit ReLU
- Other variants: leaky ReLU, shifted ReLU (SReLU), exponential linear unit (ELU), etc.



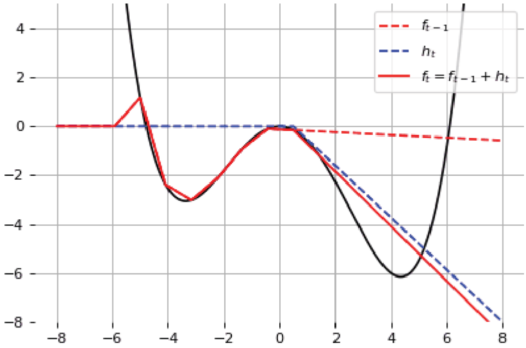
©G. Louppe

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

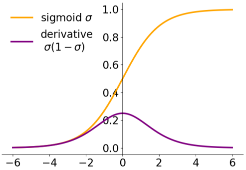


- Other variants: leaky ReLU, shifted ReLU (SReLU), exponential linear unit (ELU), etc.

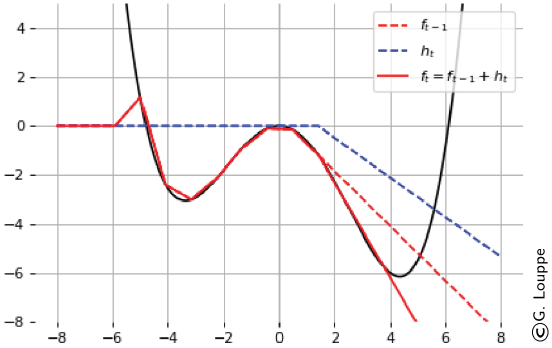


©G. Louppe

Rectified linear unit ReLU

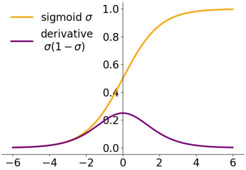


- One of reasons for vanishing gradient: sigmoid activation
  - tiny non-varying derivative away from zero
- Solution: non-saturating function
- Simplest case: rectified linear unit ReLU
- Other variants: leaky ReLU, shifted ReLU (SReLU), exponential linear unit (ELU), etc.

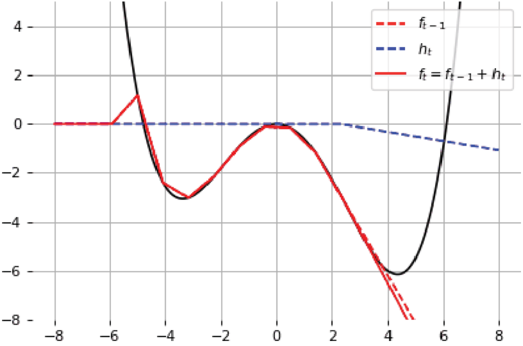


Rectified linear unit ReLU

- 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

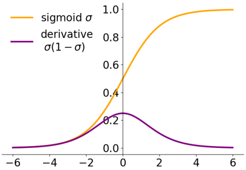


- Other variants: leaky ReLU, shifted ReLU (SReLU), exponential linear unit (ELU), etc.

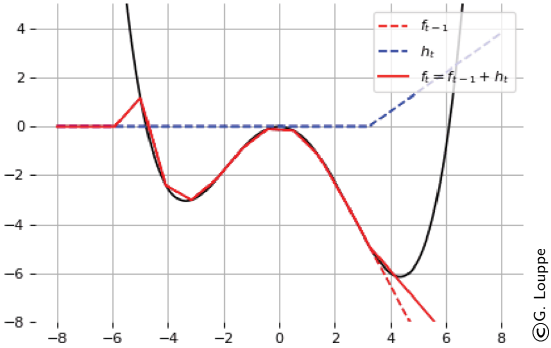


©G. Louppe

Rectified linear unit ReLU

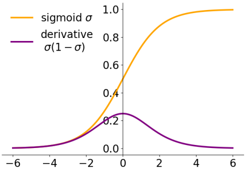


- One of reasons for vanishing gradient: sigmoid activation
  - tiny non-varying derivative away from zero
- Solution: non-saturating function
- Simplest case: rectified linear unit ReLU
- Other variants: leaky ReLU, shifted ReLU (SReLU), exponential linear unit (ELU), etc.

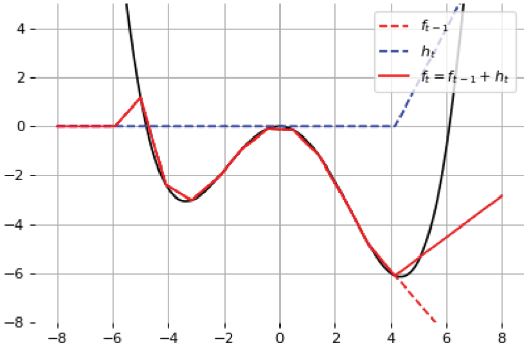


Rectified linear unit ReLU

- 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



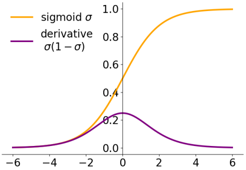
- Other variants: leaky ReLU, shifted ReLU (SReLU), exponential linear unit (ELU), etc.



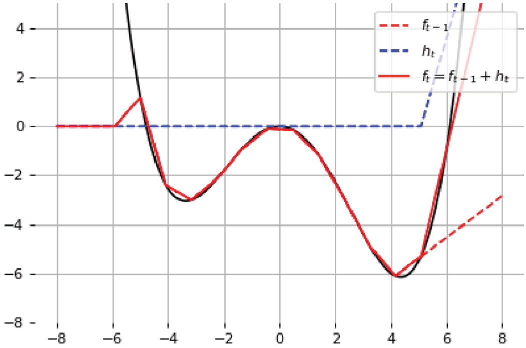
© G. Louppe

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

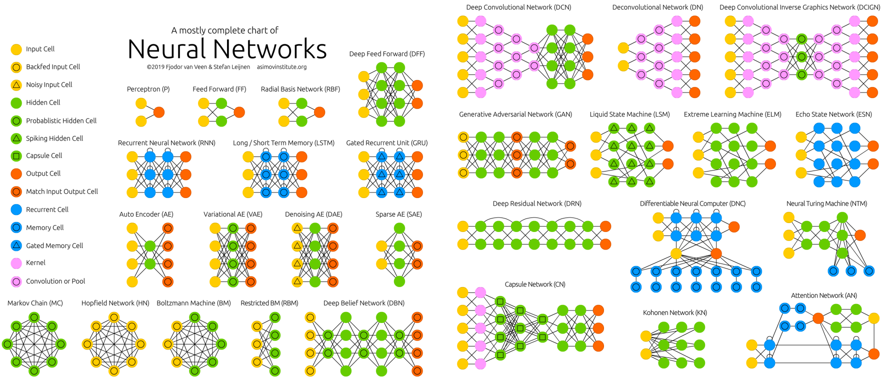


- Other variants: leaky ReLU, shifted ReLU (SReLU), exponential linear unit (ELU), etc.



©G. Louppe

Neural network zoo

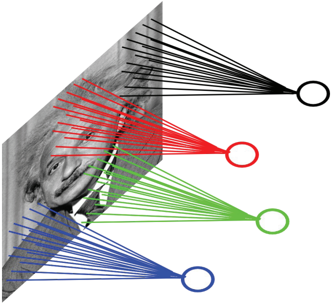


https://www.asimovinstitute.org/

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

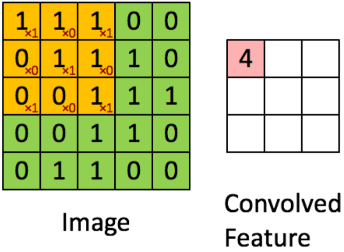
Convolutional networks (CNN)

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

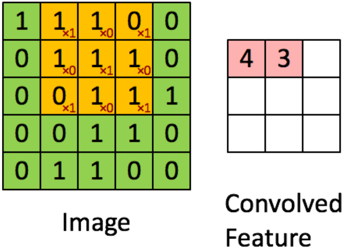


Convolutional networks (CNN)

- 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

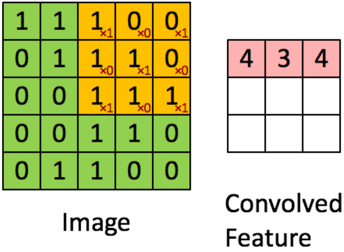


- Images are stationary: can learn feature in one part and apply it in another
- Use e.g. small patch sampled randomly, learn feature, convolve with full image



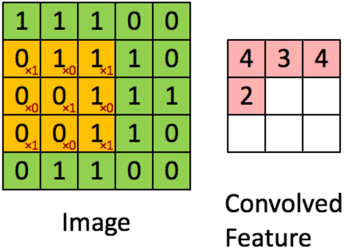
Convolutional networks (CNN)

- 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



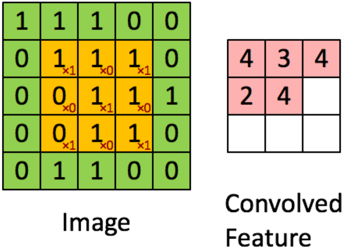
Convolutional networks (CNN)

- 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

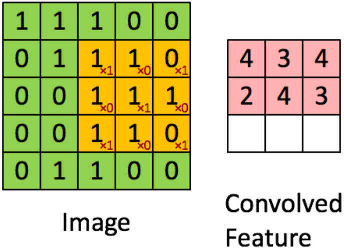


Convolutional networks (CNN)

- 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

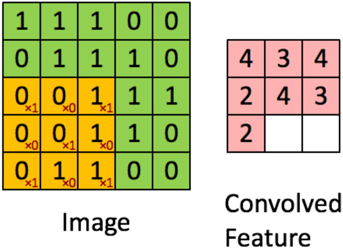


- Images are stationary: can learn feature in one part and apply it in another
- Use e.g. small patch sampled randomly, learn feature, convolve with full image

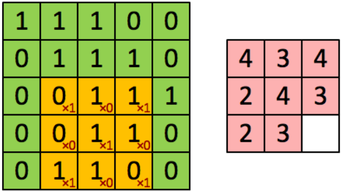


Convolutional networks (CNN)

- 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

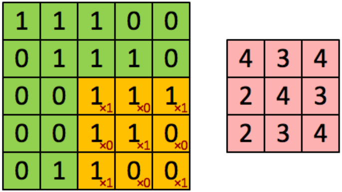


Image

Convolved Feature

Convolutional networks (CNN)

- 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

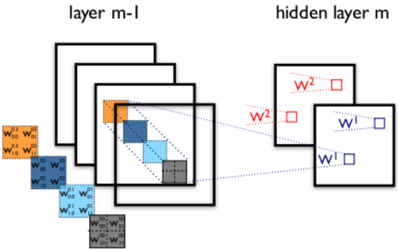


Image

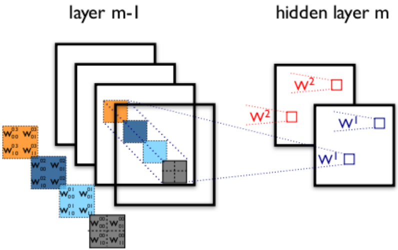
Convolved Feature

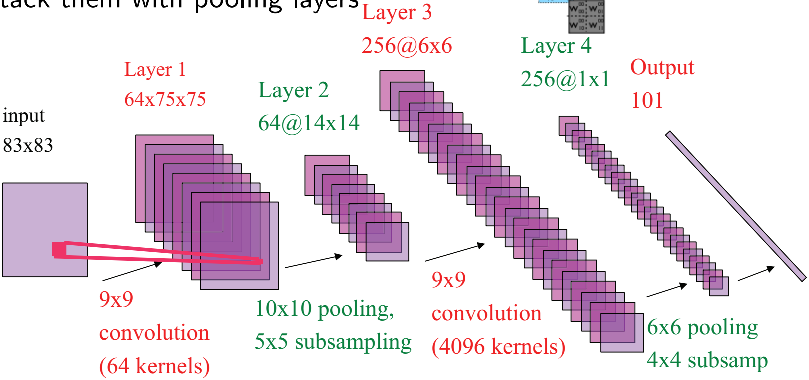
Convolutional networks (CNN)

- 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"

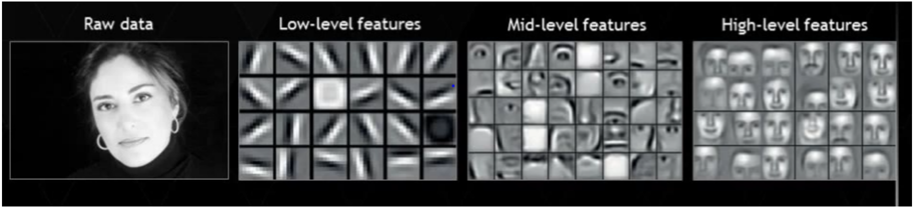


- Images are stationary: can learn feature in one part and apply it in another
- Use e.g. small patch sampled randomly, learn feature, convolve with full image
- Build several "feature maps"
- Stack them with pooling layers





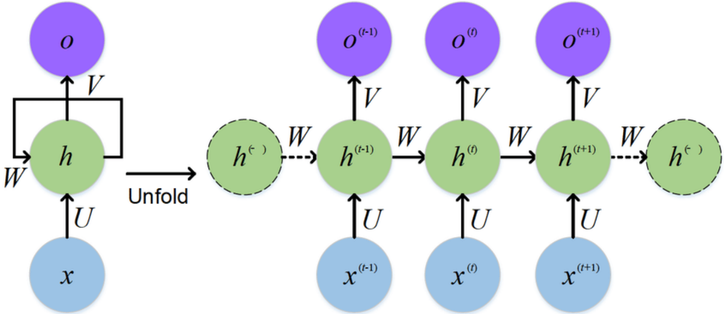
Learning feature hierarchy



Recurrent neural networks (RNN)

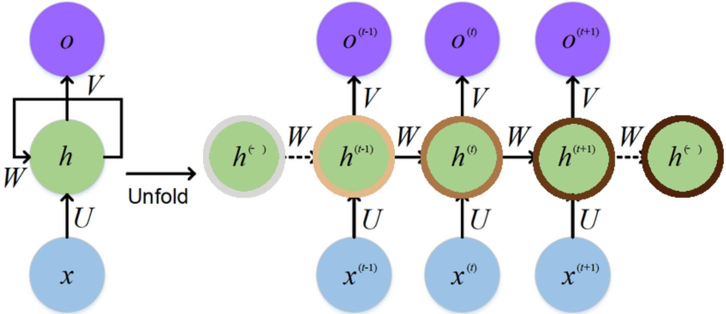
- Many problems require processing a sequence
  - sequence classification
    - text analysis ("sentiment analysis")
    - DNA sequencing
    - action selection
  - sequence synthesis
    - text synthesis
    - music/video
  - sequence translation
    - speech recognition
    - translation
- Usually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

Recurrent neural networks

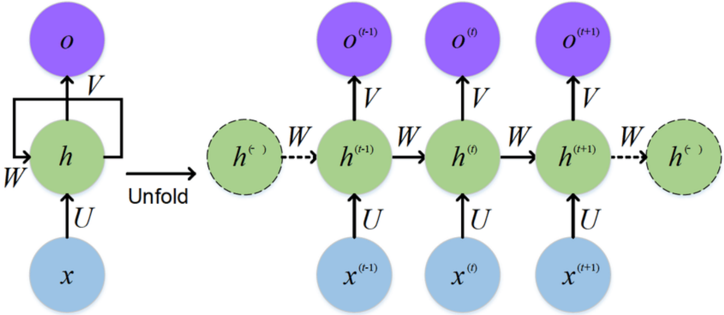


- Keeps information from earlier frames while processing (variable-size) sequence
- Could also be bi-directional, consuming sequence in both directions

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)

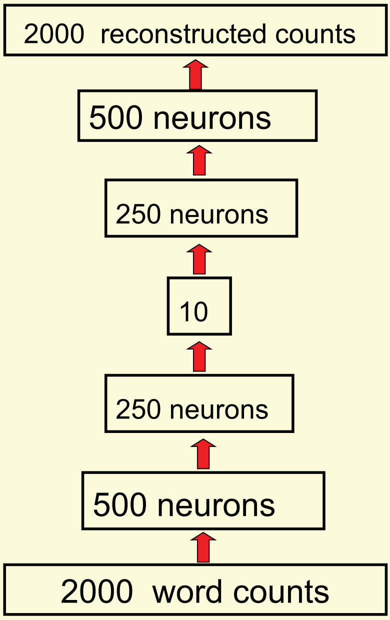
▸ more in backup

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

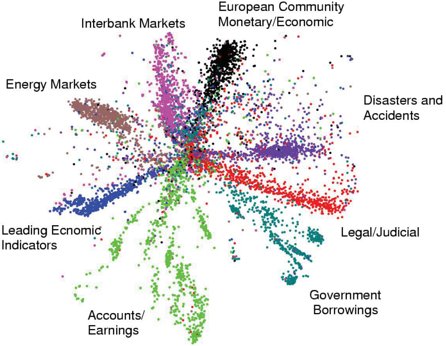
Finding similar books

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



With principle component analysis (PCA)

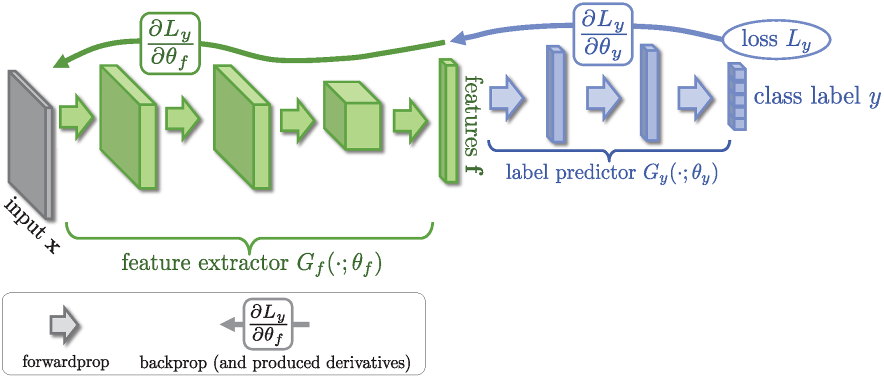
With autoencoder



more in backup

- 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

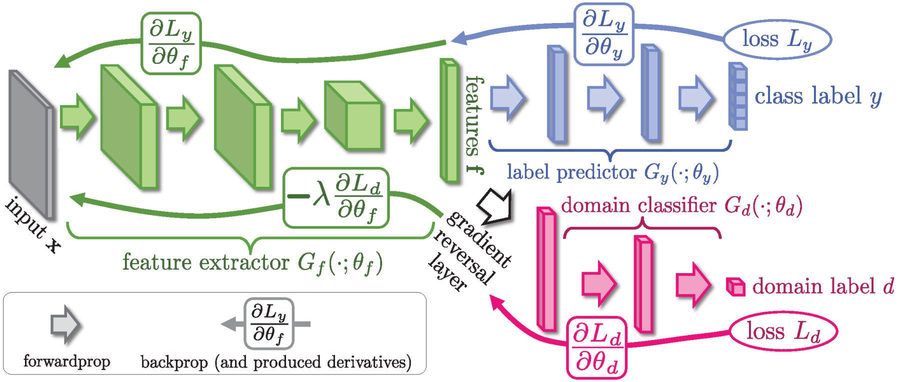


Domain adaptation and adversarial training

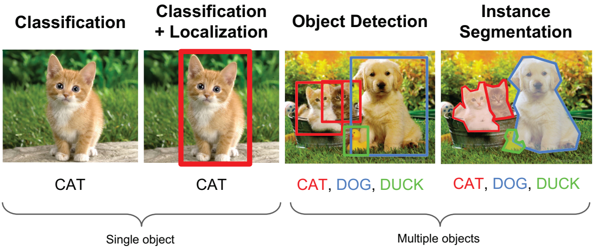
- Typical training
  - signal and background from simulation
  - results compared to real data to make measurement
- Requires good data–simulation agreement
- Possibility to use adversarial training and domain adaptation to account for discrepancies/systematic uncertainties

arXiv:1409.7495

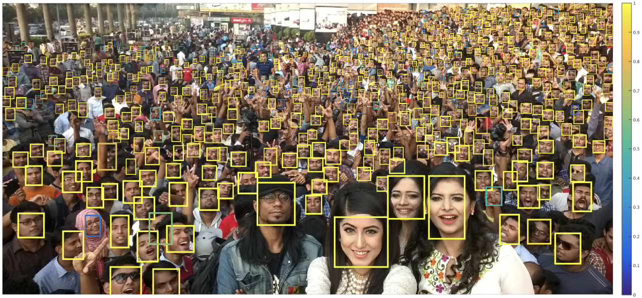
arXiv:1505.07818



Increasing refinement



- More and more granularity
- More objects, in real time on video1/video2/video3



Google DeepMind: arcade games

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

Google DeepMind: training&performance

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 $a_t = \mathrm{argmax}_a Q(\phi(s_t), a; \theta)$

    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 $\left(\phi_t, a_t, r_t, \phi_{t+1}\right)$ in D

    Sample random minibatch of transitions $\left(\phi_j, a_j, r_j, \phi_{j+1}\right)$ from D

    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \hat{Q}\left(\phi_{j+1}, a'; \theta^-\right) & \text{otherwise} \end{cases}$

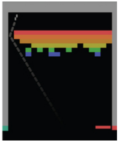
    Perform a gradient descent step on $\left(y_j - Q\left(\phi_j, a_j; \theta\right)\right)^2$ with respect to the network parameters θ

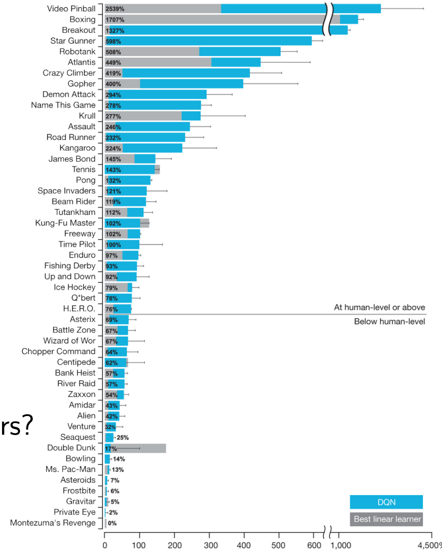
    Every C steps reset $\hat{Q} = Q$

  End For

End For

- What about Breakout or Space invaders?

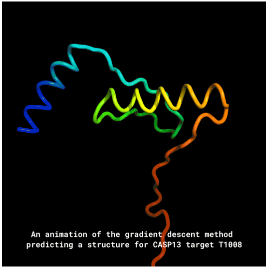


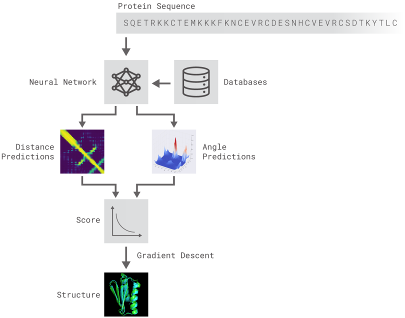


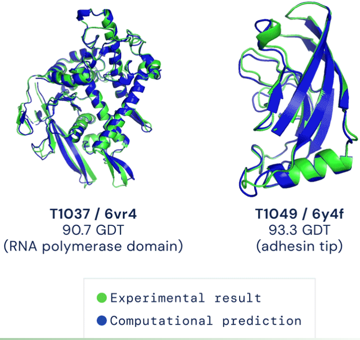
See also AlphaGo/AlphaZero in backup

DeepMind AlphaFold

- 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



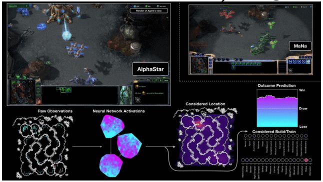




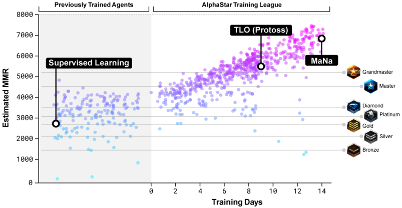
DeepMind AlphaStar

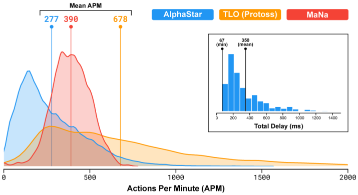
▸ Blog Jan 2019 ▸ Nature 575, 350–354 (2019)

- 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)

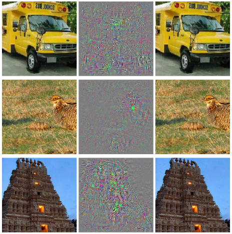


- DNN output: list of actions
- Trained for 14 days; each agent: up to 200 years of real-time play
- Runs on single desktop GPU
- Defeated 5–0 one of best pro-players



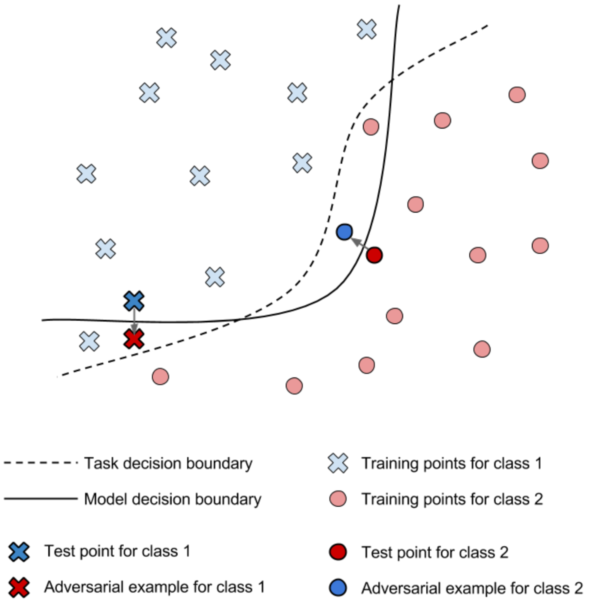


- Playing poker
  - Libratus (AI developed by Carnegie Mellon University) defeated four of the world's best professional poker players (Jan 2017 arXiv:1705.02955)
  - After 120,000 hands of Heads-up, No-Limit Texas Hold'em, led the pros by a collective $1,766,250 in chips
  - Learned to bluff, and win 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

Adversarial attacks

Cisse et al arXiv:1707.05373

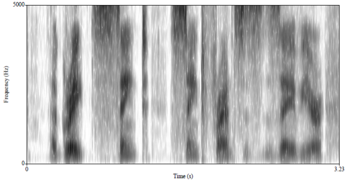
original semantic segmentation framework

adversarial attack

Adversarial attacks

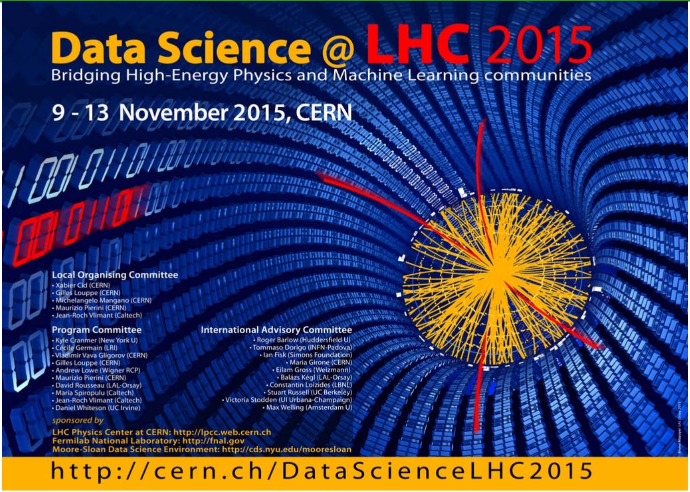
Cisse et al arXiv:1707.05373

original semantic segmentation framework   adversarial attack   compromised semantic segmentation framework

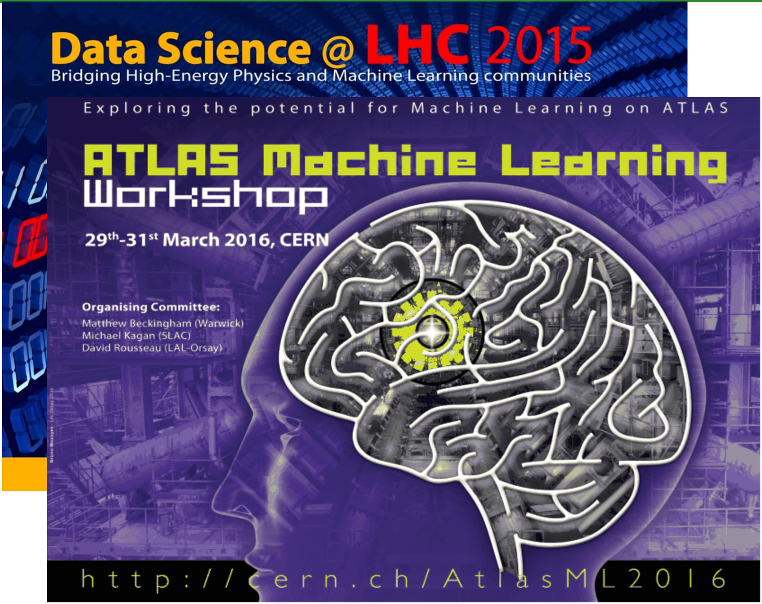


(a) a great saint saint francis zaviour (b) i great sinkt shink t frimsuss avir

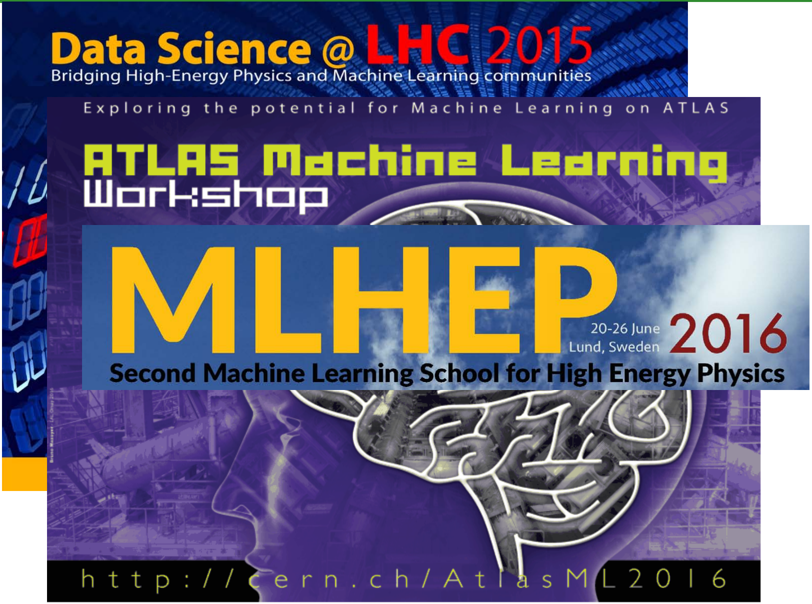
Figure 7: The model models' output for each of the spectrograms is located at the bottom of each spectrogram. The target transcription is: `A Great Saint Saint Francis Xavier.`



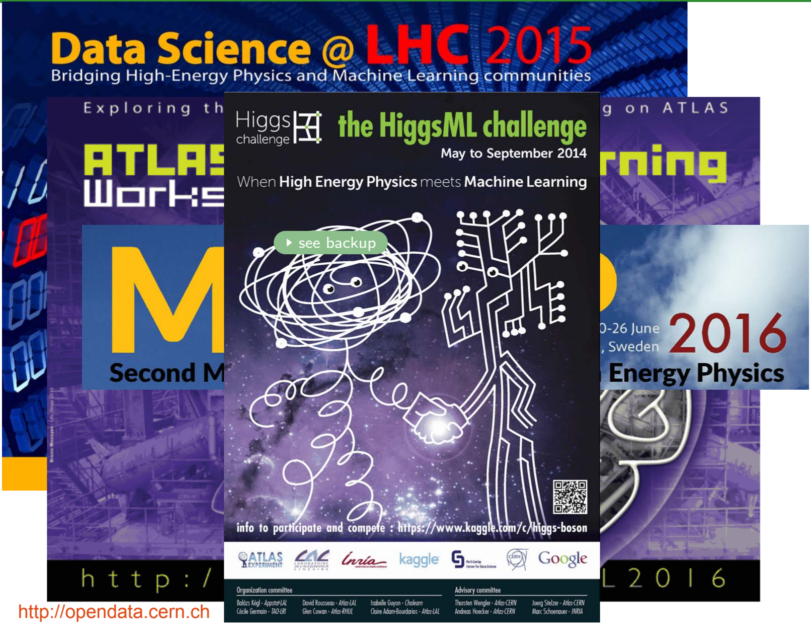
http://opendata.cern.ch



http://opendata.cern.ch

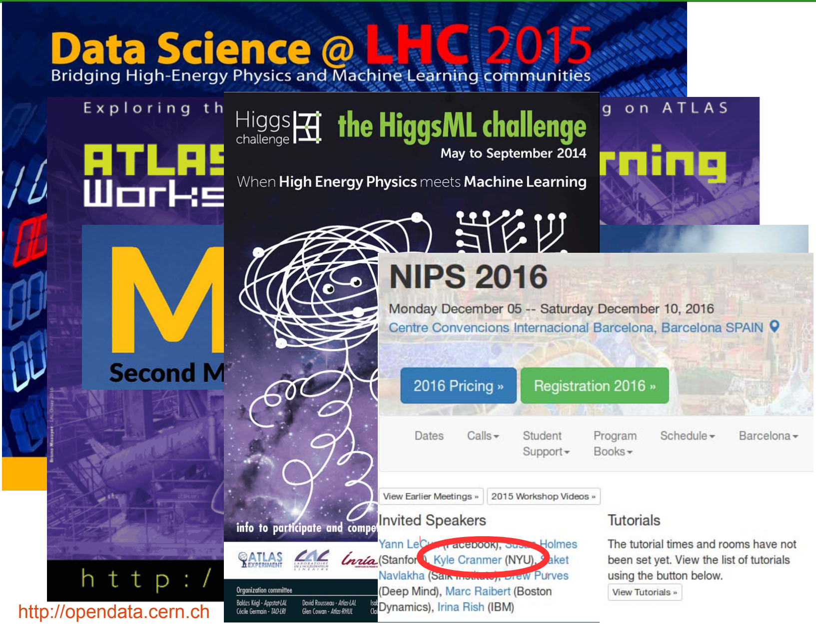


http://opendata.cern.ch



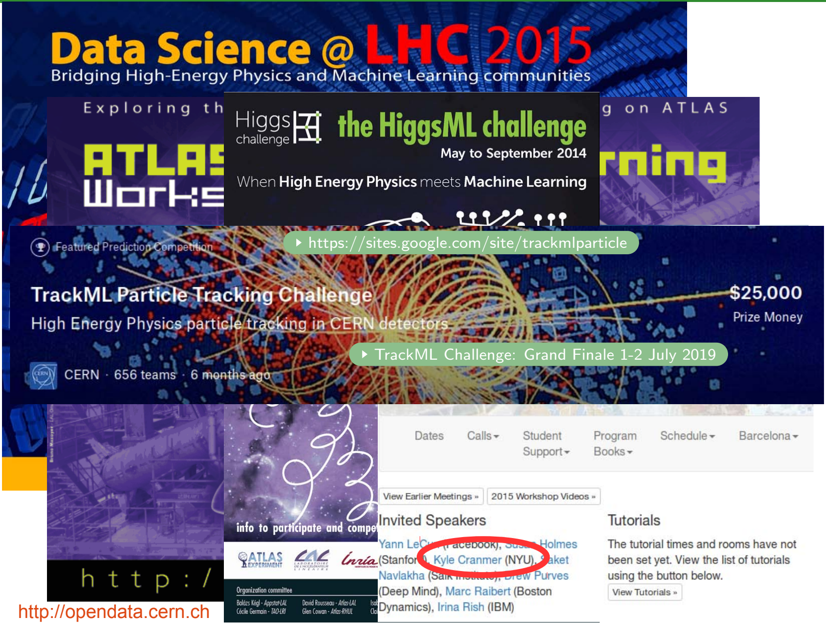
see backup

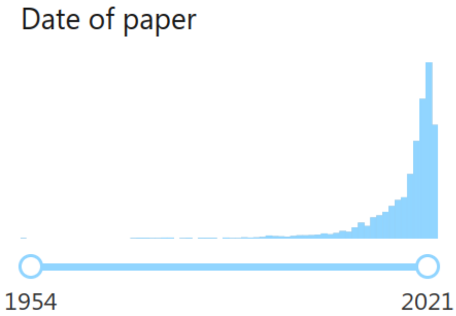
http://opendata.cern.ch



http://opendata.cern.ch

Machine learning and particle physics





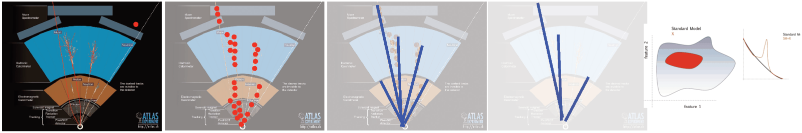
https://inspirehep.net/literature?q=machine learning or deep learning or multivariate

Up-to-date comprehensive review of papers

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

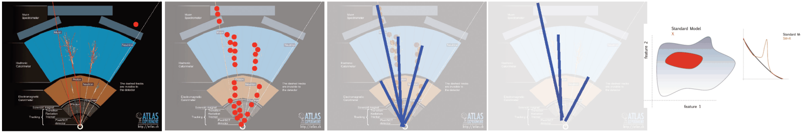
Machine learning and particle physics

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



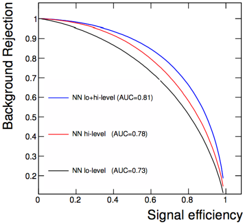
- Reduce data dimensionality to allow analysis

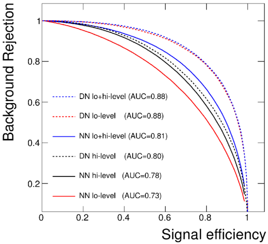
Raw	Sparsified	Reco	Select	Physics	Ana
1e7	1e4	100-ish*	50	10	1



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

arXiv:1402.4735

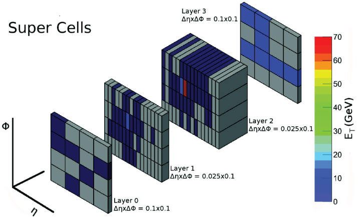


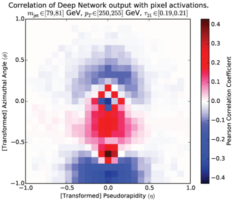


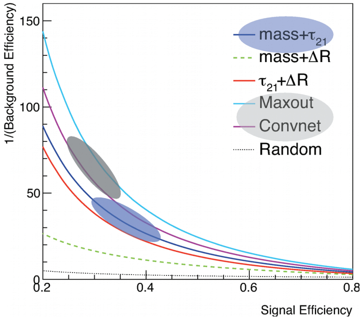
Machine learning and particle physics

- Transforming inputs into images

arXiv:1511.05190

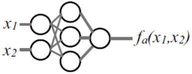




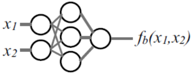


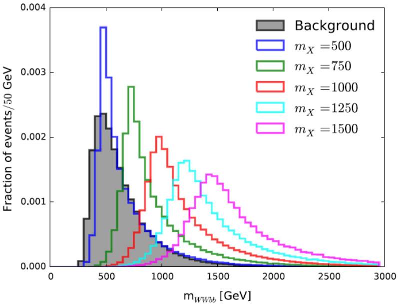
- Looking for new physics scenario with unknown mass
  $\Rightarrow$ one NN for each mass point

$\theta = \theta_a$

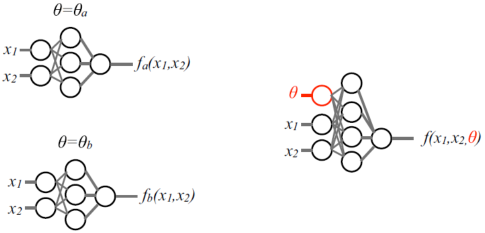


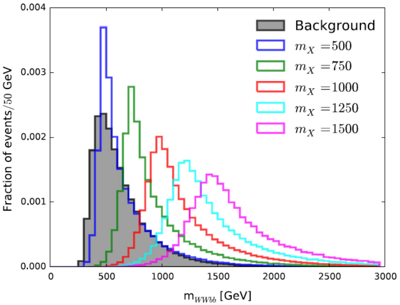
$\theta = \theta_b$



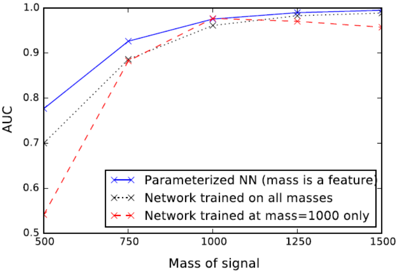


- Looking for new physics scenario with unknown mass
  $\Rightarrow$ one NN for each mass point



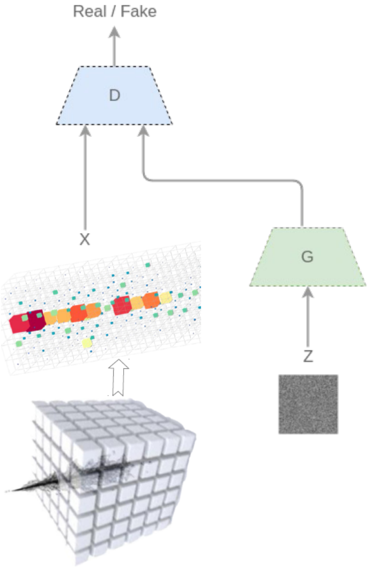


- Parameterised NN EPJC (2016) 76:235 arXiv:1601.07913
  - mass as training parameter
  - as good as dedicated training
  - generalises better



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
  - HOT OFF THE PRESS
    use GAN to simulate medium-range hadrons in ATLAS
    arXiv:2109.02551



Anomaly detection: looking for new physics

thesis

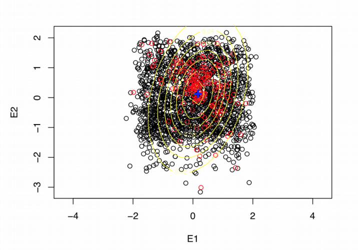
- Learn background (SM) properties
- Flag deviations from prediction without knowing anything about specific new physics scenario

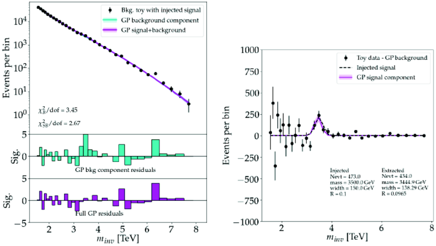
Penalised anomaly detection

- 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"





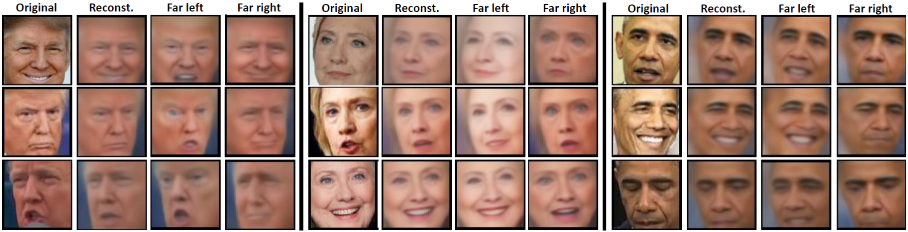
Machine learning and systematics

- No particular rule
- ML algorithm output can be considered as any other cut variable (just more powerful). Evaluate systematics by:
  - varying cut value
  - retraining
  - calibrating, etc.
- Most common (and appropriate): propagate other uncertainties (detector, theory, etc.) up to ML algorithm ouput and check how much the analysis is affected
- More and more common: profiling.
  Watch out:
  - ML algorithm output powerful
  - signal region (high ML algorithm output) probably low statistics
    $\Rightarrow$ potential recipe for disaster if modelling is not good
- May require extra systematics, not so much on technique itself, but because it probes specific corners of phase space and/or wider parameter space (usually loosening pre-ML selection cuts)

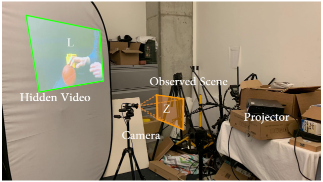
Deep learning: looking forward

- 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
- 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)
- Getting popular: graph networks

- 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

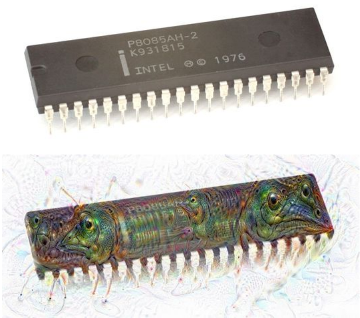
Upcoming reference book (recently read final proofs)

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

Deep networks and art

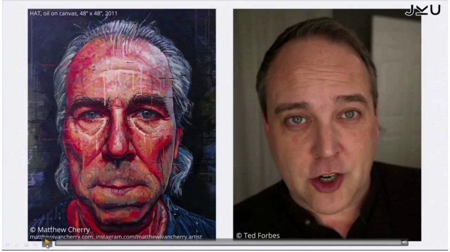
- Learning a style ▸ arXiv:1508.06576 [cs.CV] ▸ Neural-style

- Computer dreams ▸ Google original
  ▸ deepdream



- Face Style ▸ http://facestyle.org

▸ http://dcgi.fel.cvut.cz/home/sykorad/facestyle.html



Summary

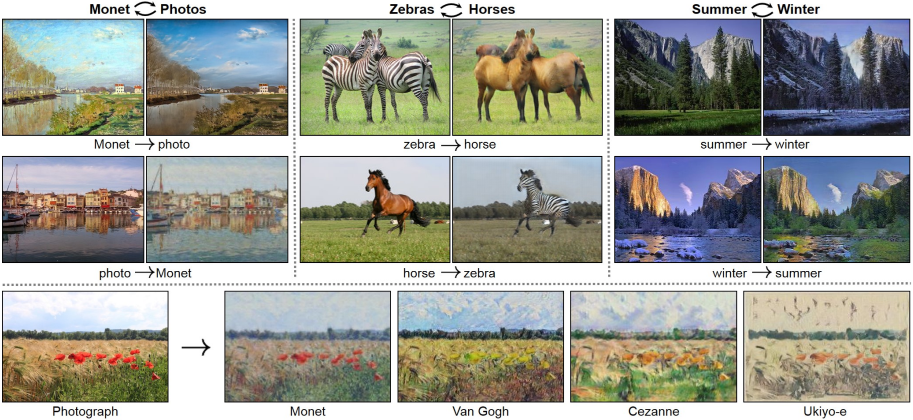

Monet ⟳ Photos
Zebras ⟳ Horses
Summer ⟳ Winter
Monet ⟶ photo
zebra ⟶ horse
summer ⟶ winter
photo ⟶ Monet
horse ⟶ zebra
winter ⟶ summer
Photograph
Monet
Van Gogh
Cezanne
Ukiyo-e

Photo enhancement

Input Output Input Output Input Output

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

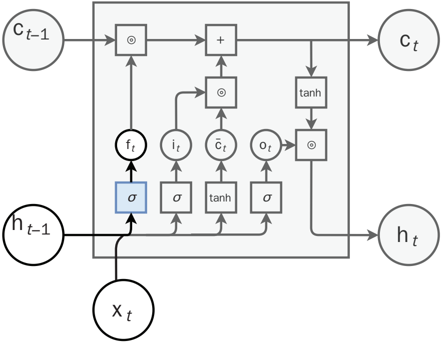
- M.A. Ranzato, C. Poultney, S. Chopra and Y. LeCun, in J. Platt et al., "Efficient Learning of Sparse Representations with an Energy-Based Model", in *Advances in Neural Information Processing Systems 19* (NIPS'06), pages 1137–1144, MIT Press, 2007

- 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

Backup

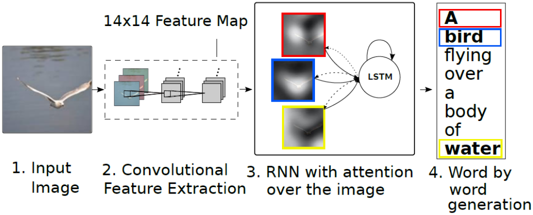


- Recurrent state split in two parts
  - cell state c_t
  - output state h_t
- Forget gate f_t to erase cell state info
- Input gate i_t to update cell state info
- Output gate o_t to select output state

Recurrent neural networks examples

Labelling images

arXiv:1502.03044

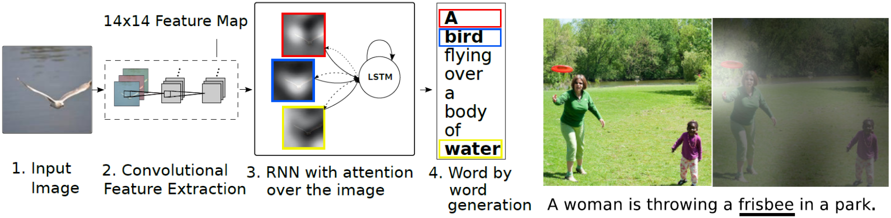


A woman is throwing a frisbee in a park.

Recurrent neural networks examples

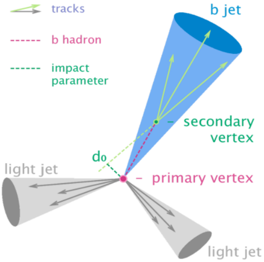
Labelling images

arXiv:1502.03044



b-jet tagging in ATLAS experiment

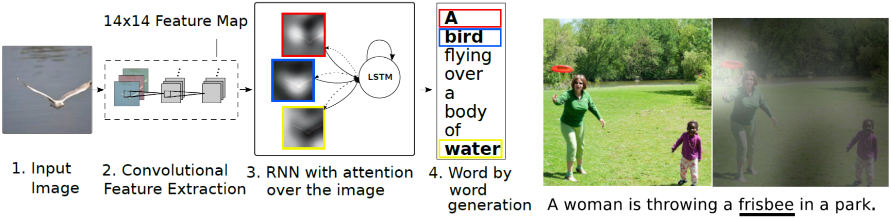
ATL-PHYS-PUB-2017-003



Recurrent neural networks examples

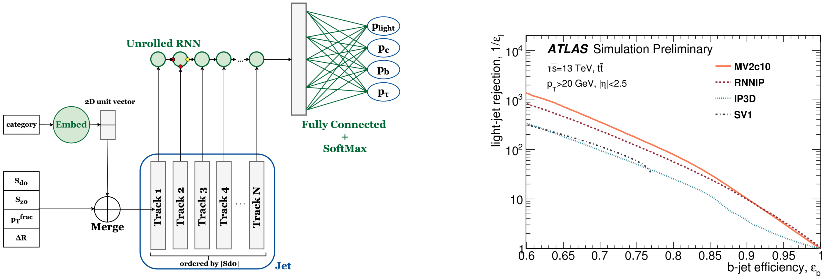
Labelling images

arXiv:1502.03044



b-jet tagging in ATLAS experiment

ATL-PHYS-PUB-2017-003

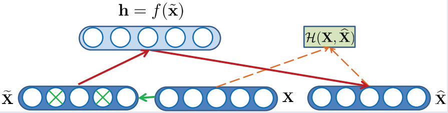


Sparse auto-encoder

- 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

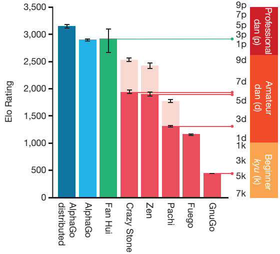
Google DeepMind: mastering Go

- 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.

Google DeepMind: AlphaGo

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) $\Rightarrow$ better position selection (policy network) and better evaluation (value network)

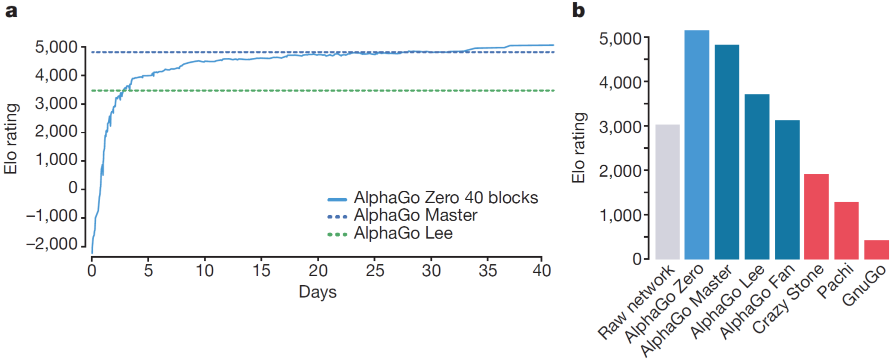


- Then played Lee Sedol (top Go play in the world over last decade) in March 2016 $\Rightarrow$ won 4–1. AlphaGo given honorary professional ninth dan, considered to have "reach a level 'close to the territory of divinity' "
- Ke Jie (Chinese world #1): "Bring it on!". May 2017: 3–0 win for AlphaGo.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

DeepMind AlphaGo Zero

Nature 550, 354 (2017)

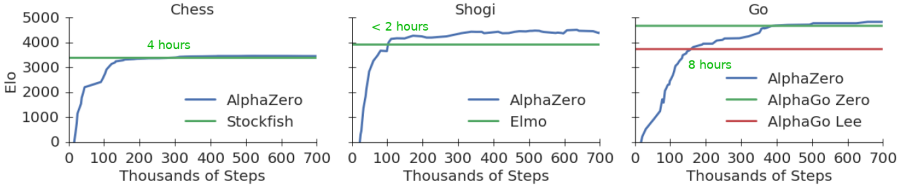
- Learn from scratch, just from the rules and random moves
- Reinforcement learning from self-play, no human data/guidance
- Combined policy and value networks
- 4.9 million self-play games
- Beats AlphaGo Lee (several months of training) after just 36 hours
- Single machine with four TPU



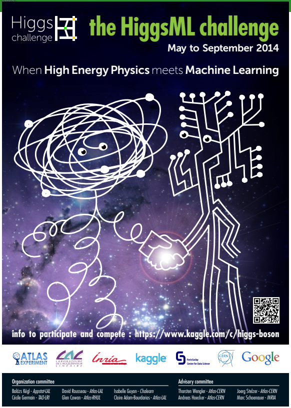
DeepMind AlphaZero

arXiv:1712.01815 [cs.AI]

- 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

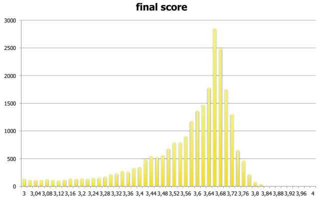


Machine learning and particle physics



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		