Deep Learning Taxonomy

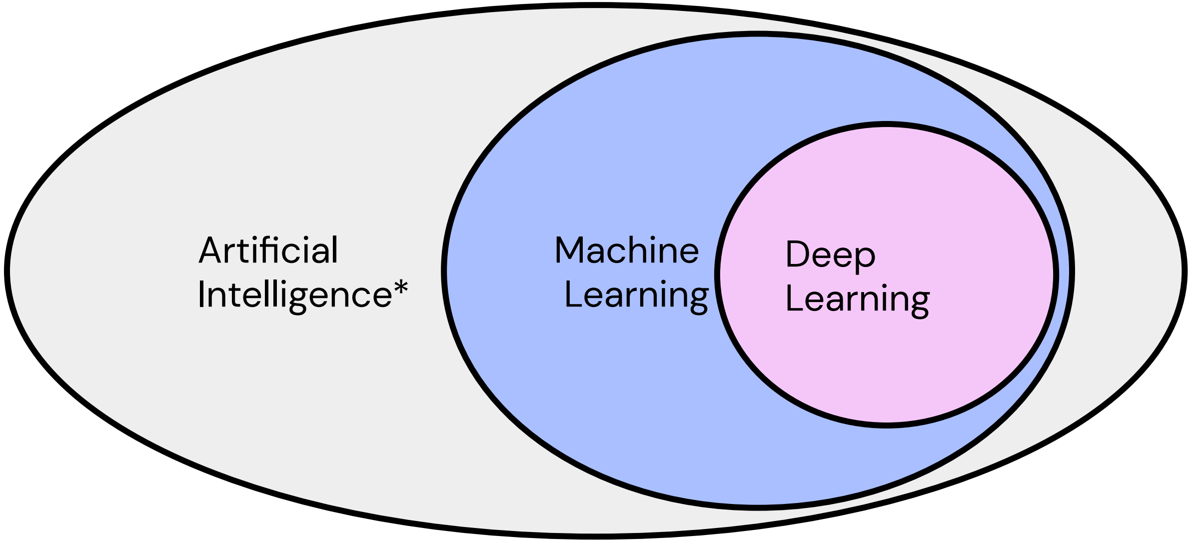
Razvan Pascanu
Research Scientist @ DeepMind
razp@google.com

https://xkcd.com/1838/

What is Deep Learning about?

Intuitively, machine learning (and by extension deep learning) studies how to teach machines (agents) to *solve tasks* with little to no human intervention by learning from interaction with the particular task (i.e. from data). The solution is not prescribed by the practitioner, just the learning strategy (and encoding of the solution).

Deep Learning focuses on a particular family of *parametric* approaches that has been extremely successful recently. This usually indicates a particular family of learning algorithms (typically gradient-based ones) and architectures (neural networks).



Starting point: Supervised Learning

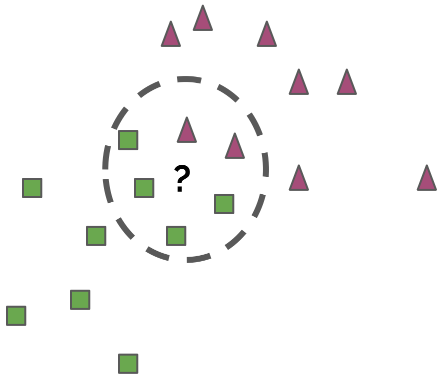
- Let $\mathcal{X}$ denote the space of input values
- Let $\mathcal{Y}$ denote the space of output values
- Given a data set $D \subset \mathcal{X} \times \mathcal{Y}$, find a function:

$$h : \mathcal{X} \to \mathcal{Y}$$

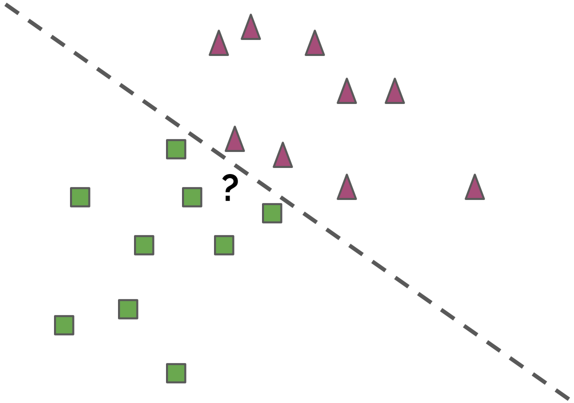
  such that $h(\mathbf{x})$ is a *"good predictor"* for the value of y.
- h is called a *hypothesis*
- Problems are categorized by the type of output domain
  - If $\mathcal{Y} = \mathbb{R}$, this problem is called *regression*
  - If $\mathcal{Y}$ is a categorical variable (i.e., part of a finite discrete set), the problem is called *classification*
  - In general, $\mathcal{Y}$ could be a lot more complex (graph, tree, etc), which is called *structured prediction*

$$h : \mathcal{X} \to \mathcal{Y}$$

non-parametric
k nearest neighbour classifier



parametric
Linear models
Neural Networks



A parameterized function is a function:

$$h : \theta \times \mathcal{X} \rightarrow \mathcal{Y}$$

for example a linear function of the form

$$h(w, x) = wx$$

Learning then boils down to finding *the best* θ to minimize the distance between prediction and targets

$$\arg\min_{\theta} L(\theta) = \arg\min_{\theta} \mathbb{E}\left[dist(h(\theta, x_i), y_i)\right]$$

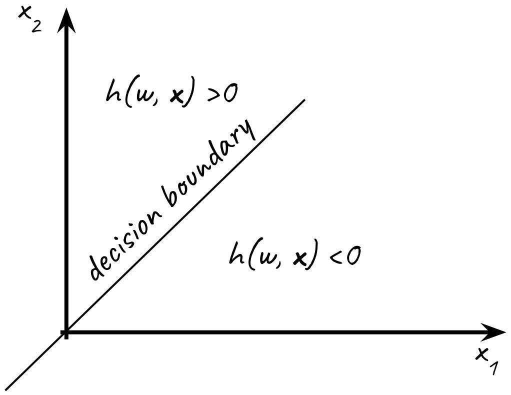
We will focus on two questions:

What is a good parametrization ?

How do we find these optimal parameters ?

What is a good parametrization ?

The linear model $h({\color{red}w}, x) = {\color{red}w}x$

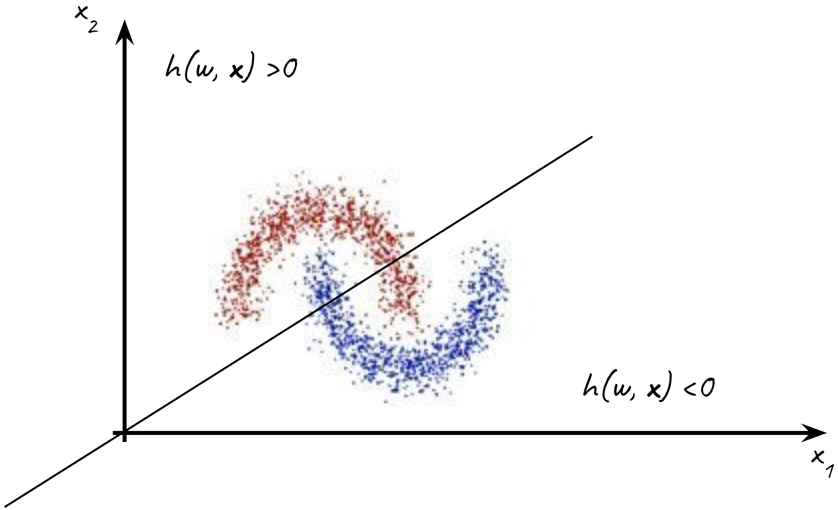


- Can straightforwardly be used for classification (and regression)
- Defines linear decision boundaries (and represents linear functions)

This (intuitively) is not enough

What is a good parametrization ?

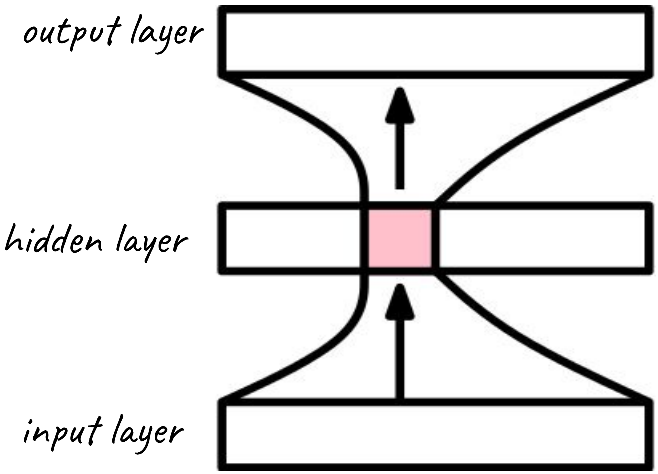
Most problems are not linearly separable.



Neural networks to the rescue!

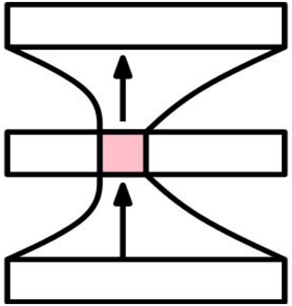
$$h(\mathbf{x}, \theta) = \mathbf{W}_{out} ReLU(\mathbf{W}_{in}\mathbf{x} + \mathbf{b}_{in}) + \mathbf{b}_{out}$$

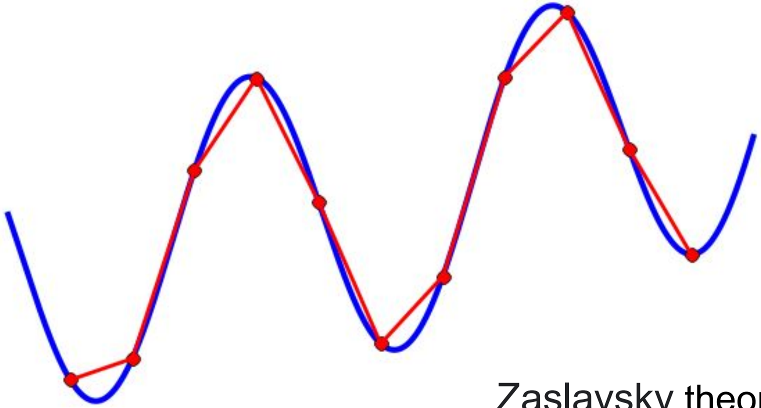
$$ReLU(x) = \begin{cases} x & x > 0 \\ 0 & \text{otherwise} \end{cases}$$

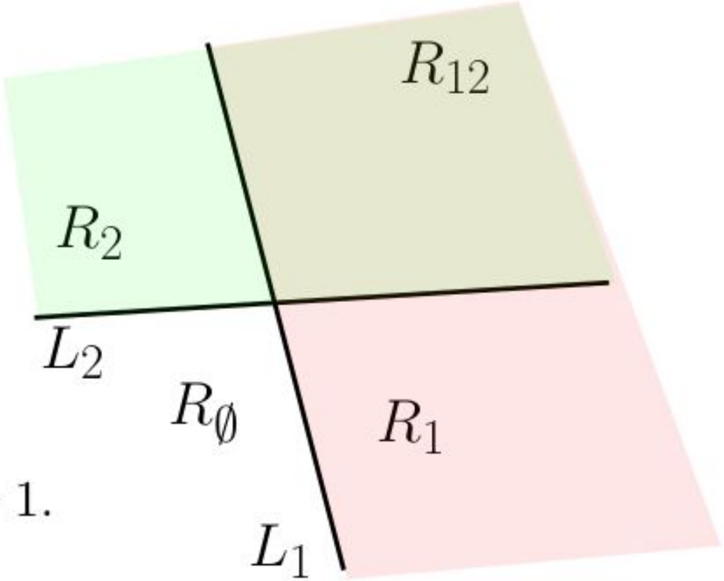


- Other non-linearities are possible
- **Why have a non-linearity?**

$$h(\mathbf{x}, \theta) = \mathbf{W}_{out} ReLU(\mathbf{W}_{in}\mathbf{x} + \mathbf{b}_{in}) + \mathbf{b}_{out}$$







Zaslavsky theorem (1975):

$$r(\mathcal{A}_m) = \binom{m}{2} + m + 1.$$

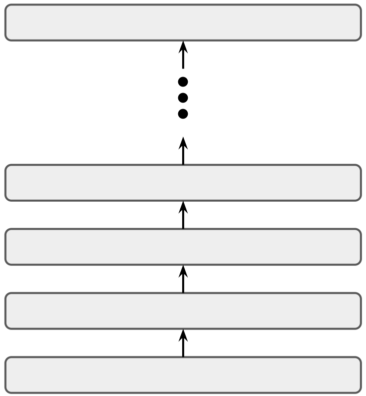
Note: proof is very nice and visual :)

Deep in deep learning:

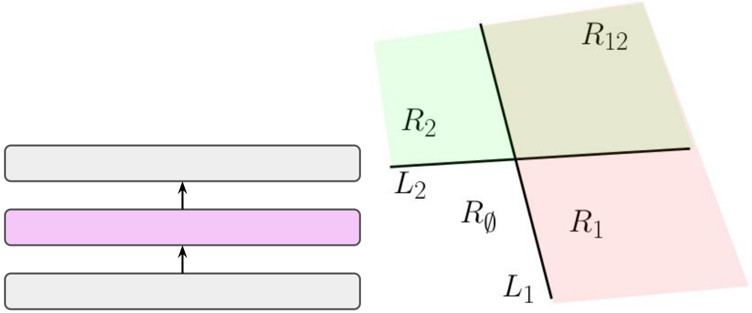
$$h(\mathbf{x},\theta) = \mathbf{W}_k ReLU(\mathbf{W}_{k-1} ReLU(\mathbf{W}_{k-2} \ldots ReLU(\mathbf{W}_1\mathbf{x}+\mathbf{b}_1)+\ldots)+\mathbf{b}_{k-1})+\mathbf{b}_k$$

The term ***deep learning*** or ***deep neural networks*** was introduce to highlight the importance of ***depth****.*

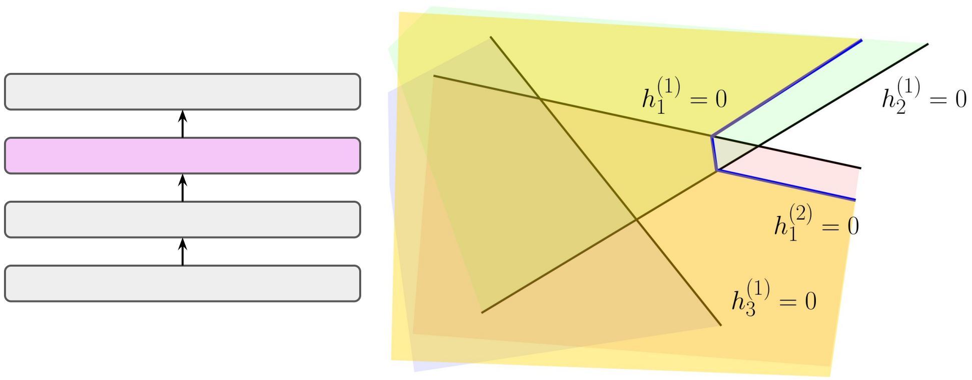
- What does depth provide? Why is it the secret sauce?



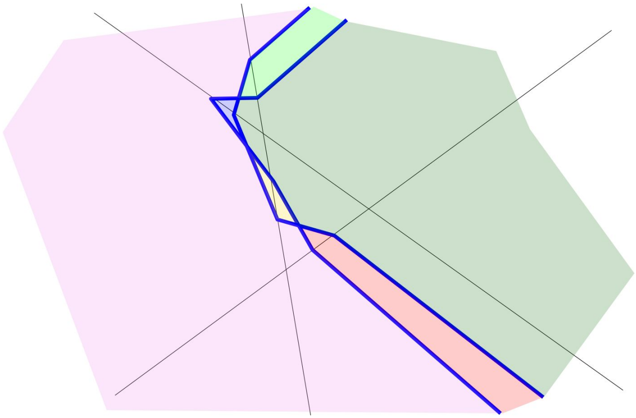
$$\mathbf{W}_{1,[i:]}\mathbf{x} + \mathbf{b}_{1,[i]} = 0$$

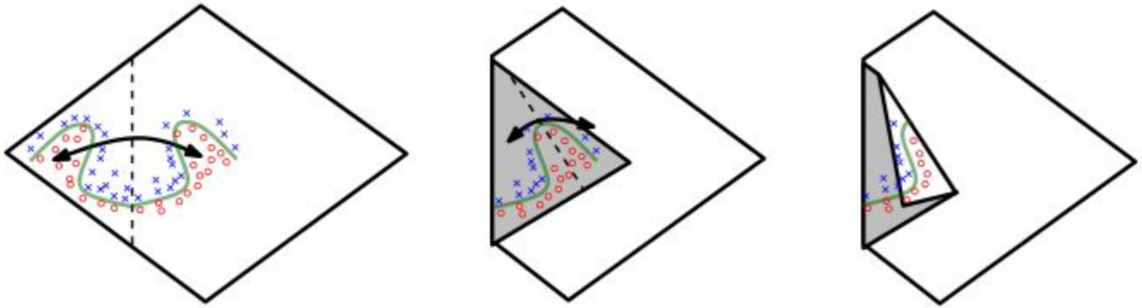


$$\mathbf{W}_{2,[i:]}ReLU(\mathbf{W_1}\mathbf{x} + \mathbf{b}_1) + \mathbf{b}_{2,[i]} = 0$$



Partitioning into regions by second hidden layer

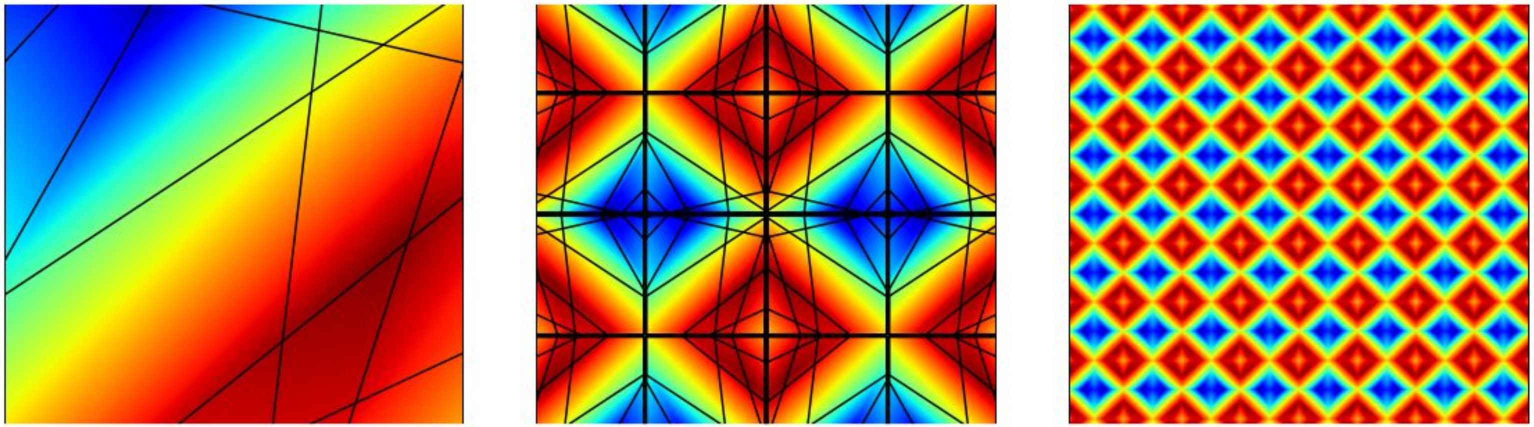




Guido Montufar et al 2014, On the number of linear regions of Deep Neural Networks

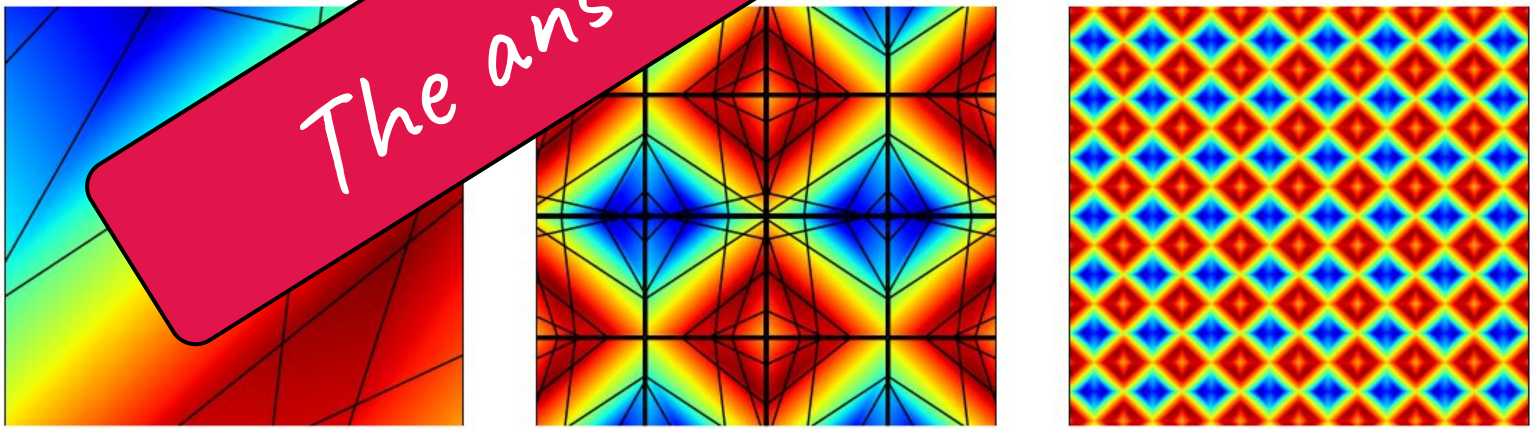
By folding the space, you gain expressivity!
(exponentially more linear regions)
without increasing the number of parameters!

- Is having exponentially more linear regions a good thing? Can this explain the success of DL?
- Is the limiting factor of previous methods expressivity?
- Do shallow model underperform because of lack of capacity?

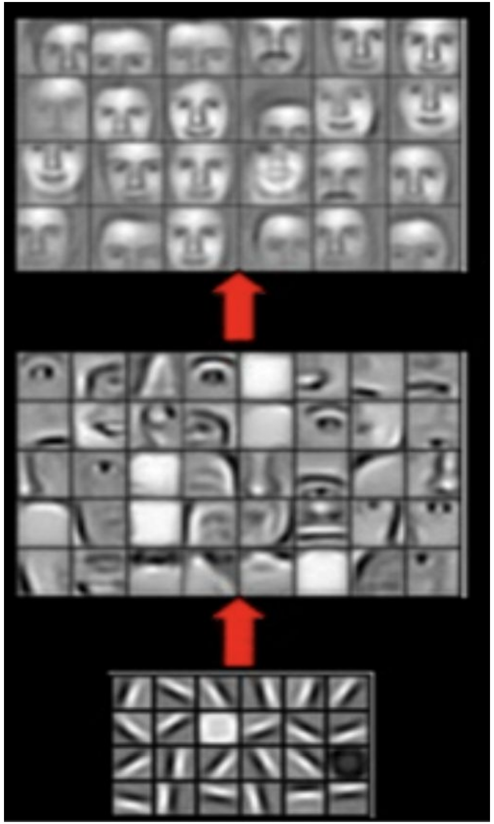


- Is having exponentially more linear regions a good thing? Can this explain the success of DL?
- Is the limiting factor of previous metho expressivity?
- Do shallow model underpe lack of capacity?

The answer is probably no

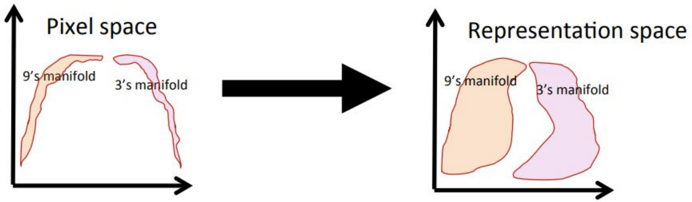


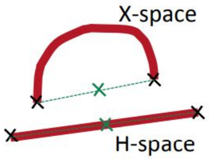
- Depth acts as an inductive bias (constraints the solution).
- Do we understand what this inductive bias is?



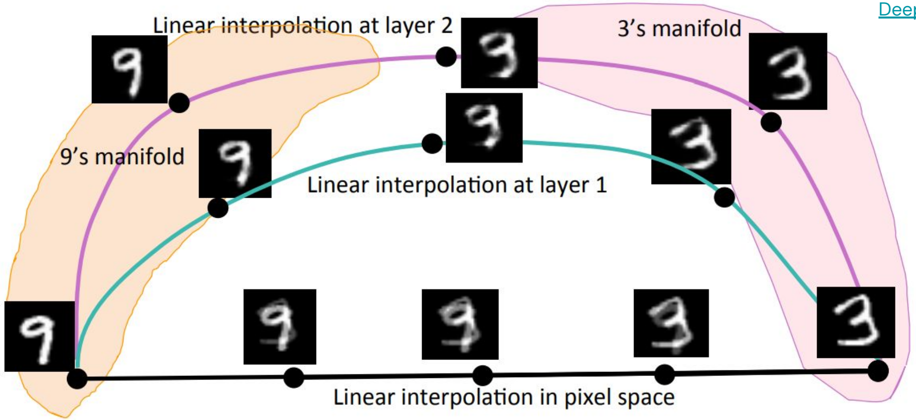
Matt Zeiler & Rob Fergus 2013, Visualizing and Understanding Convolutional Networks

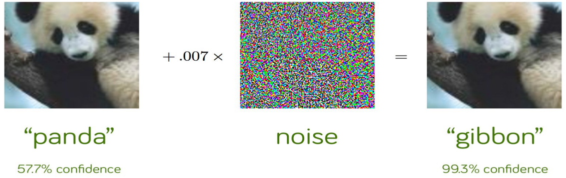
Honglak Lee et al. 2009, Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations





Sherjil Ozair & Yoshua Bengio 2014, Deep Directed Generative Autoencoders





Ian J. Goodfellow et al 2014 Explaining and Harnessing Adversarial Examples

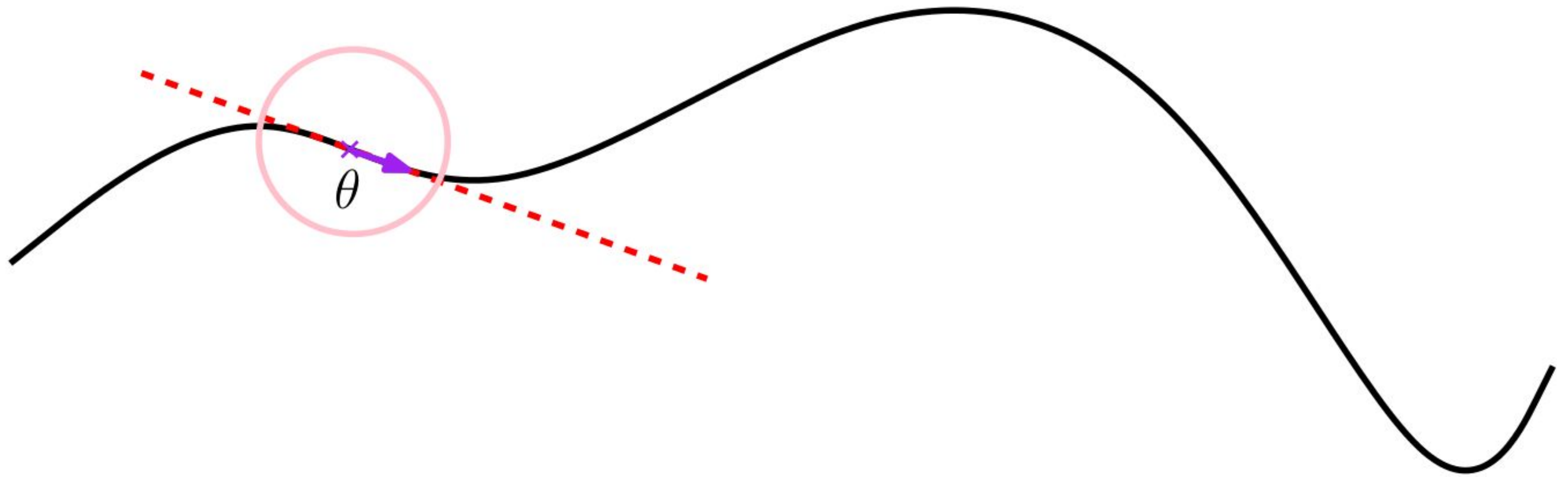
How do we find the optimal parameters ?

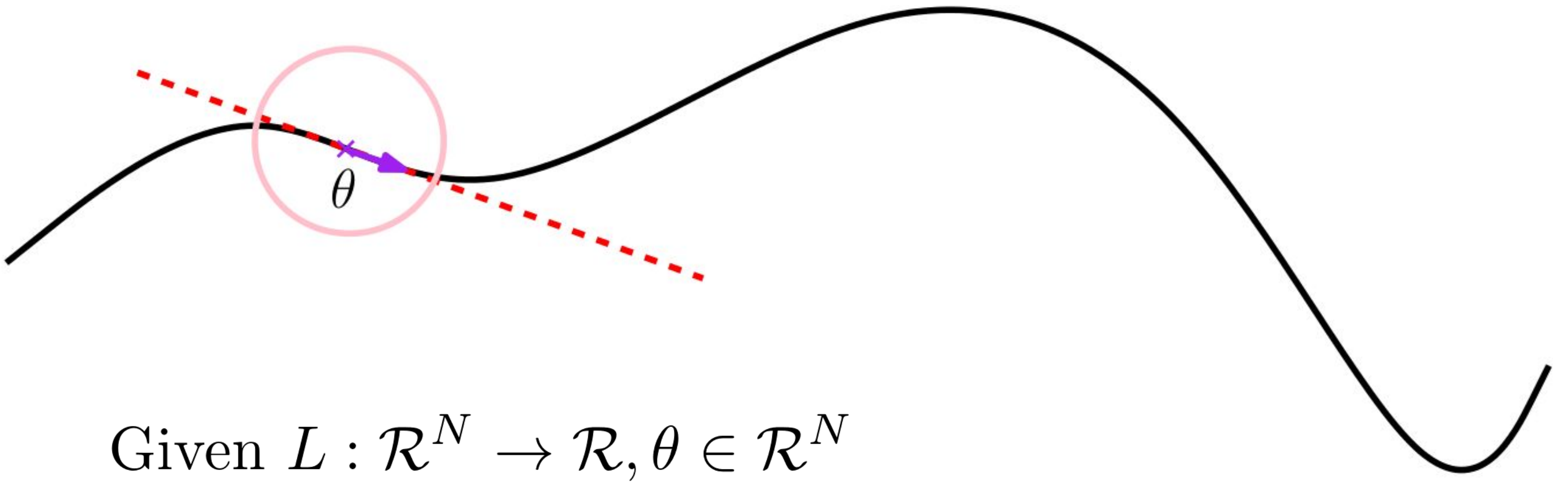
$$\arg\min_{\theta} L(\theta) = \arg\min_{\theta} \boxed{\mathbb{E}} \left[dist(h(\theta, x_i), y_i) \right]$$

Optimization

Learning/Generalization

Optimization: Gradient Descent





$$\text{Given } L : \mathcal{R}^N \to \mathcal{R}, \theta \in \mathcal{R}^N$$

$$\arg\min_{\Delta\theta} L(\theta + \Delta\theta) \approx \arg\min_{\Delta\theta} \left[L(\theta) + \Delta\theta \frac{\partial L}{\partial \theta} \right]$$

$$s.t. \|\Delta\theta\| \leq \epsilon$$

$$\arg\min_{\Delta\theta}\left[L(\theta)+\Delta\theta\frac{\partial L}{\partial\theta}\right]$$

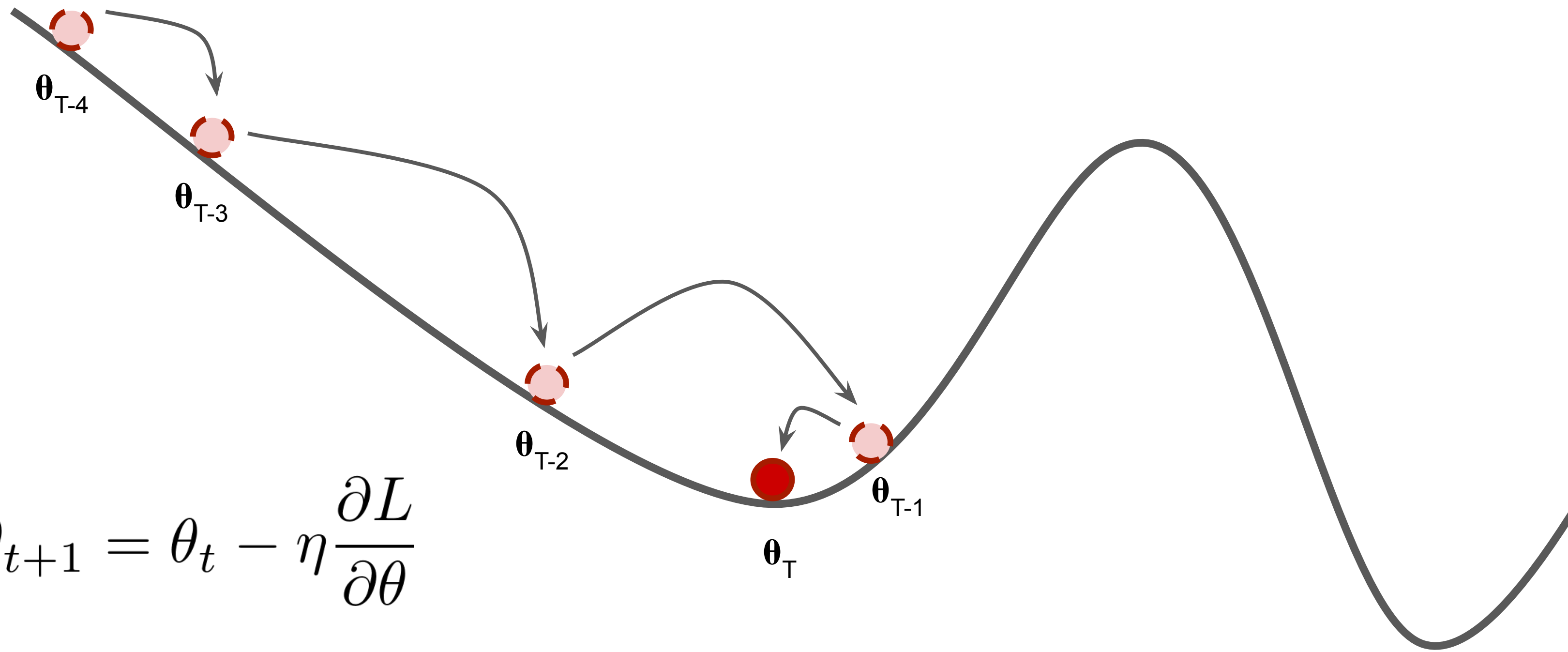
$$s.t.\|\Delta\theta\| \leq \epsilon$$

$$\big\downarrow \text{ Lagrange multipliers}$$

$$\arg\min_{\Delta\theta}\left[L(\theta)+\Delta\theta\frac{\partial L}{\partial\theta}+\eta\boldsymbol{\Delta\theta}\mathbf{I}\boldsymbol{\Delta\theta}^{T}\right]$$

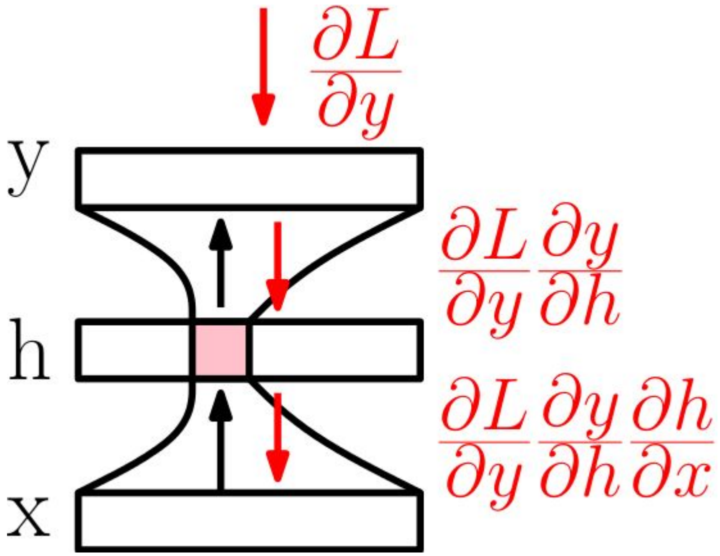
$$\big\downarrow$$

$$\theta_{t+1}=\theta_t-\eta\frac{\partial L}{\partial\theta}$$



$$\theta_{t+1} = \theta_t - \eta \frac{\partial L}{\partial \theta}$$

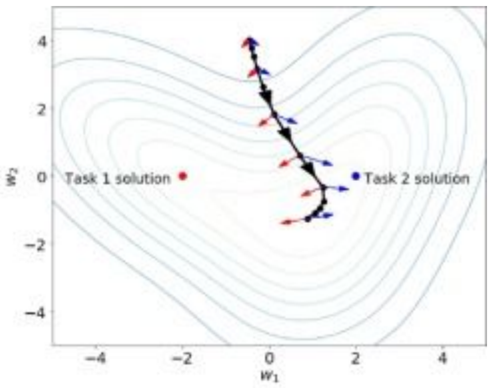
The celebrated ***Backpropagation Algorithm***



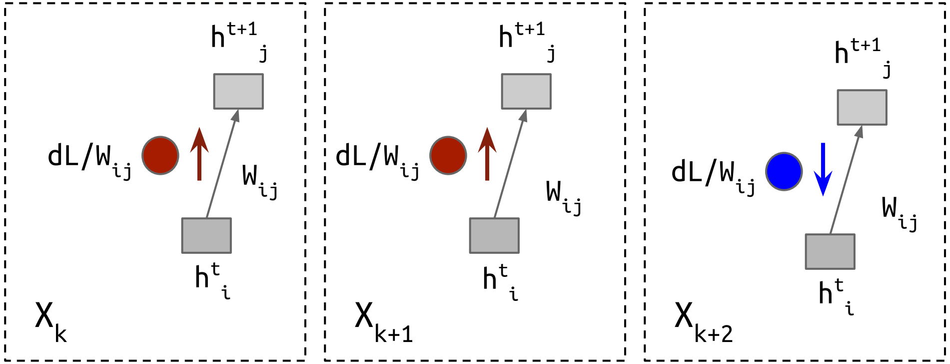
$$\theta_{t+1} = \theta_t - \eta \frac{\partial L}{\partial \theta}$$

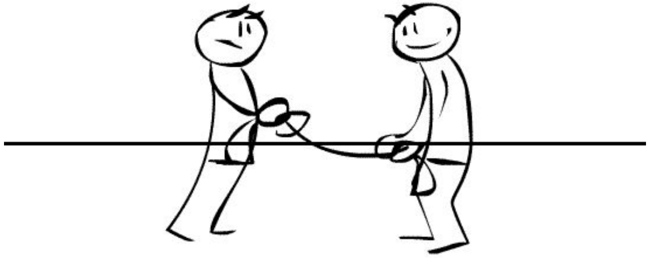
- Why is it important to start from the output towards the input?
- How is it different from the chain rule?

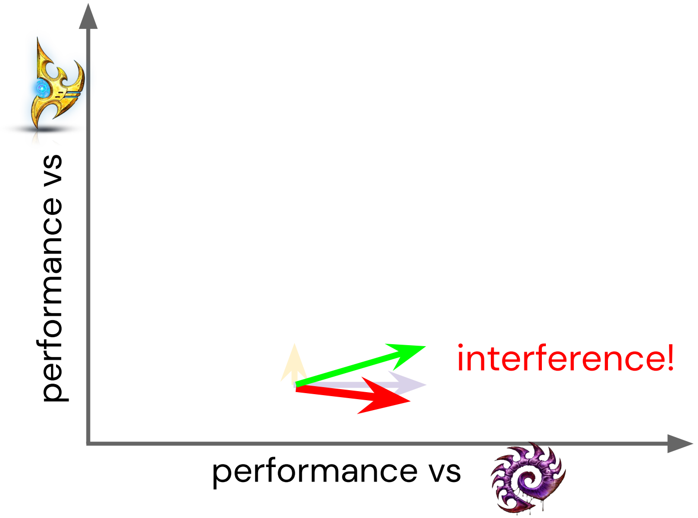
The IID assumption in Gradient Descent

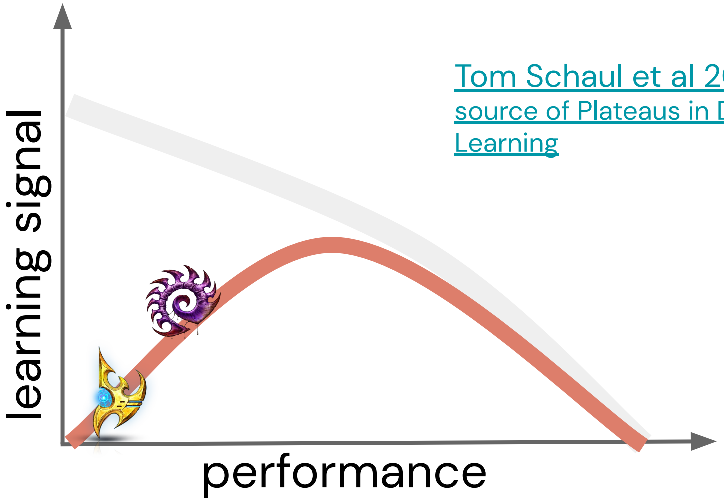


$$\frac{\partial L}{\partial W_{ij}} = \lim_{\epsilon \to 0} \frac{L(W_{ij}) - L(W_{ij} + \epsilon)}{\epsilon}$$

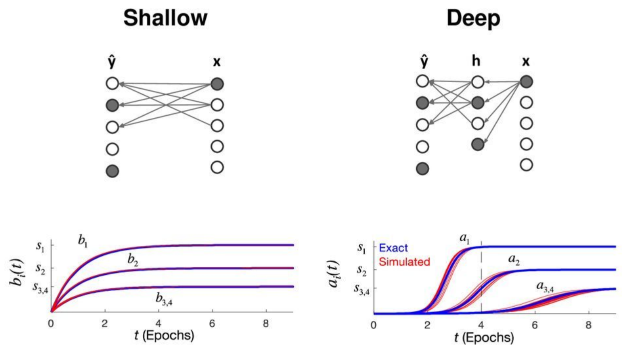








[Tom Schaul et al 2019, Ray interference: a source of Plateaus in Deep Reinforcement Learning]



[Andrew Saxe et al 2013, Exact solutions to the nonlinear dynamics in deep linear models]

Learning has **tug-of-war** dynamics to resolve credit assignment

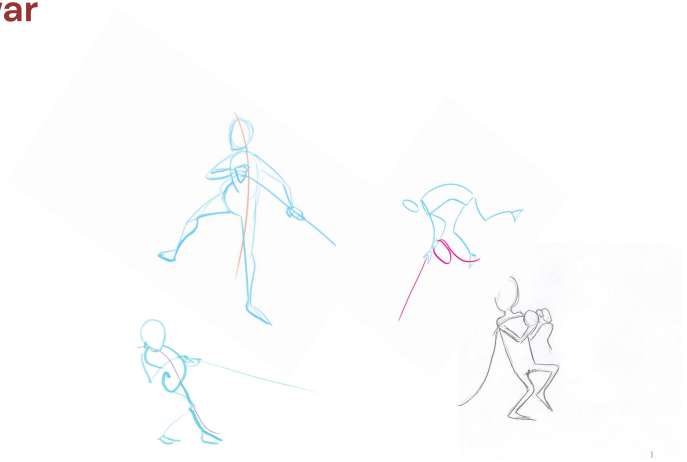
- This requires data to be **I.I.D.**
- **No explicit knowledge composition**

- Failure in credit assignment leads to catastrophic forgetting

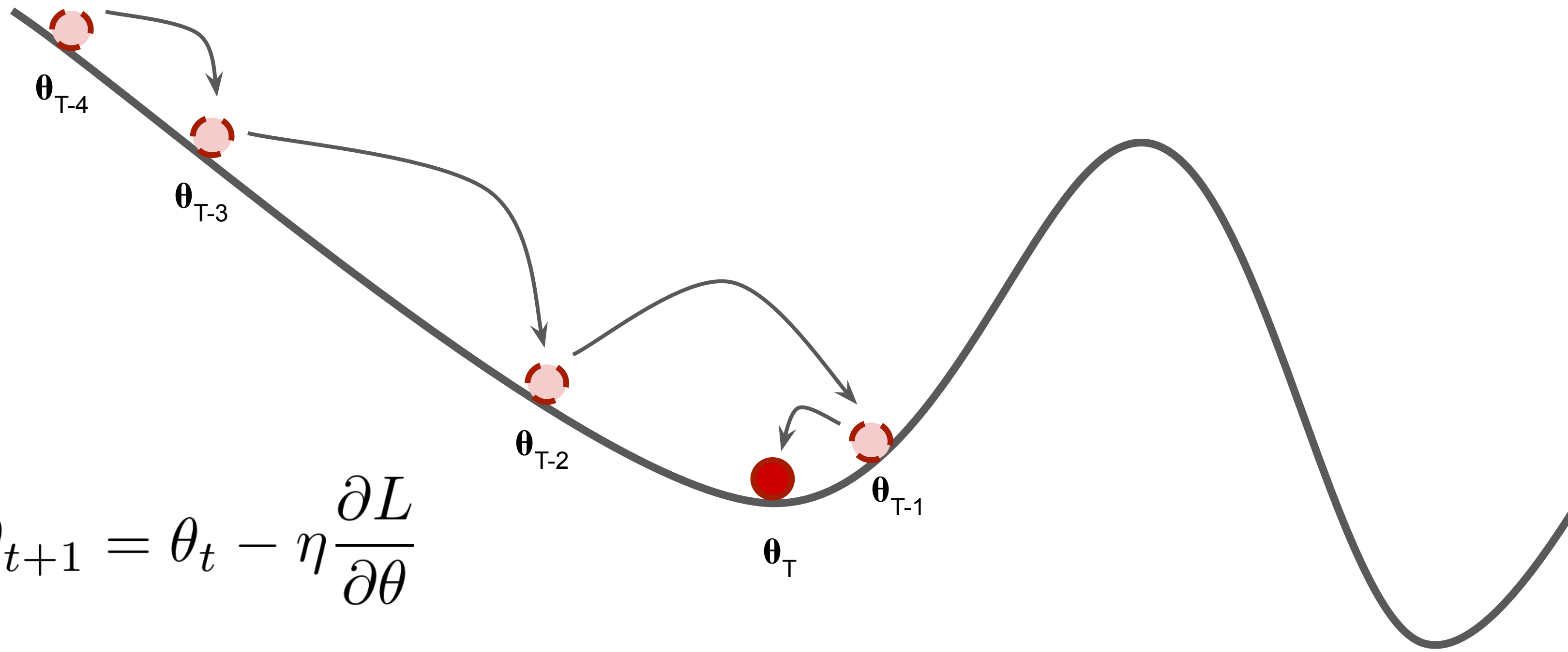
Continual Learning studies credit assignment in learning, and in particular the effects of this tug-of-war dynamics.

Resolving continual learning means finding better/different mechanism of doing credit assignments. CL is about optimization/learning.

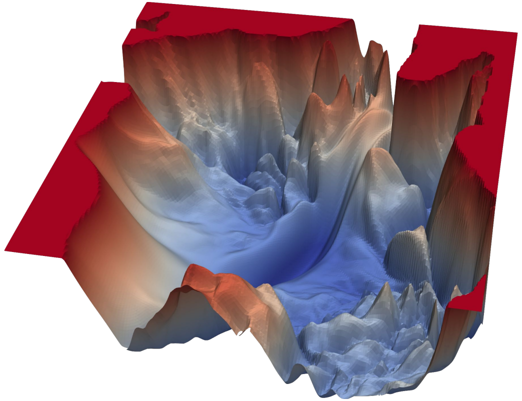
It has big implications for our understanding of the learning process for neural networks.



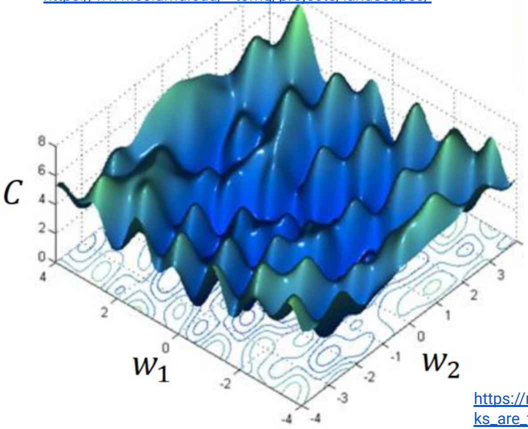
Converging to an optimal !?



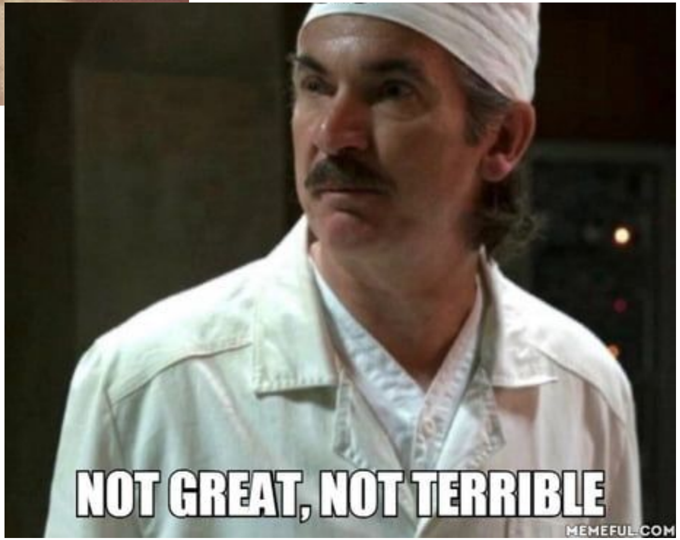
$$\theta_{t+1} = \theta_t - \eta \frac{\partial L}{\partial \theta}$$



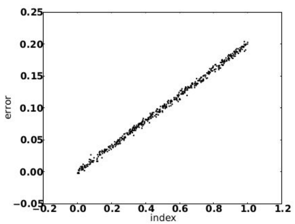
https://www.cs.umd.edu/~tomg/projects/landscapes/

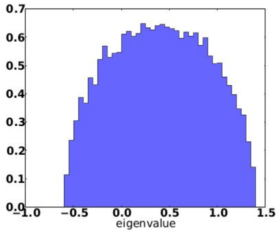


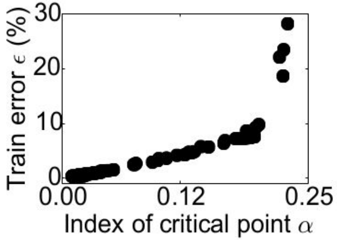
https://ml4a.github.io/ml4a/how_neural_networks_are_trained/



- Statistical physics (on random gaussian fields) [Bray and Dean, 2007, Fyodorov and Williams, 2007]

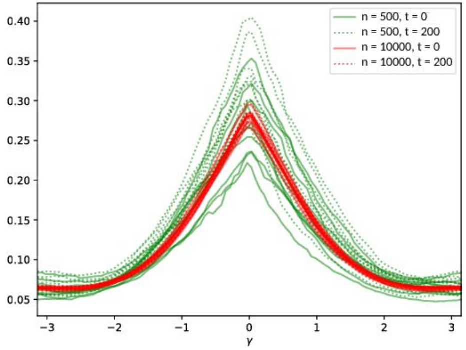






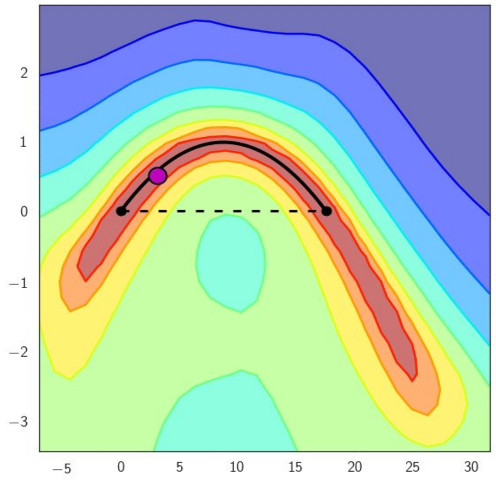
[Dauphin et al, *Identifying and attacking the saddle point problem in high-dimensional nonconvex optimization, NIPS 2014*]

[Jacot et al, Neural Tangent Kernel: Convergence and Generalization in Neural Networks]

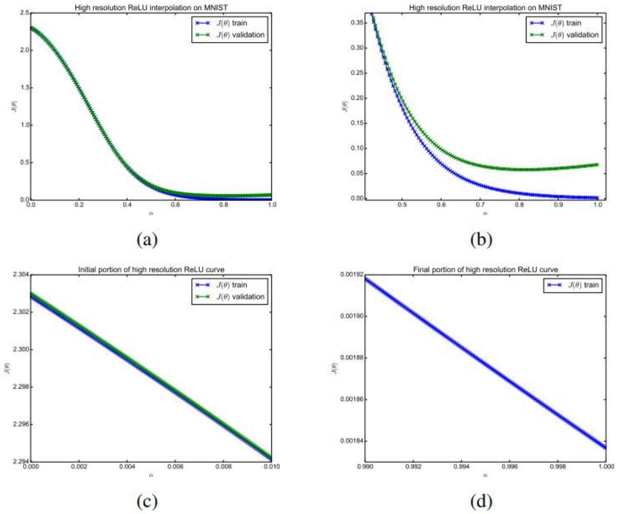


[Loss Surfaces, Mode Connectivity, and Fast Ensembling of DNNs, Garipov et al]

+ Related works



[Goodfellow, Vinyals & Saxe Qualitatively Characterizing Neural Network Optimization Problems, ICLR 2015]

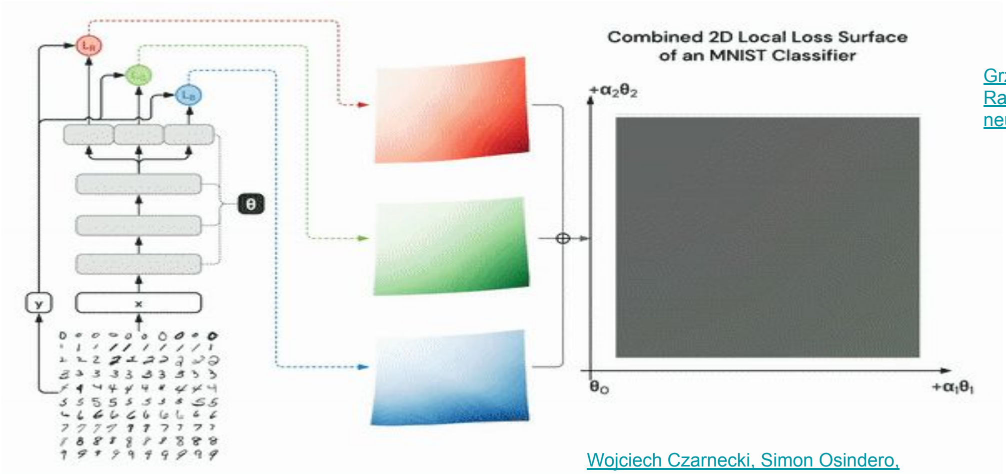


(a) (b) (c) (d)

The Deep Learning Myth:

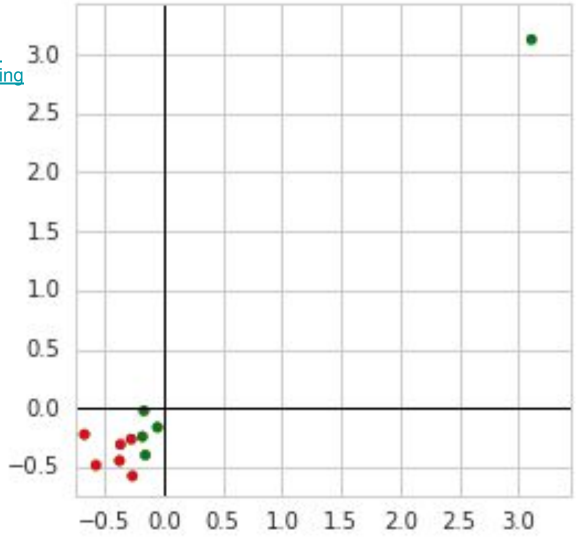
Deep Networks can be inserted almost anywhere and will just work

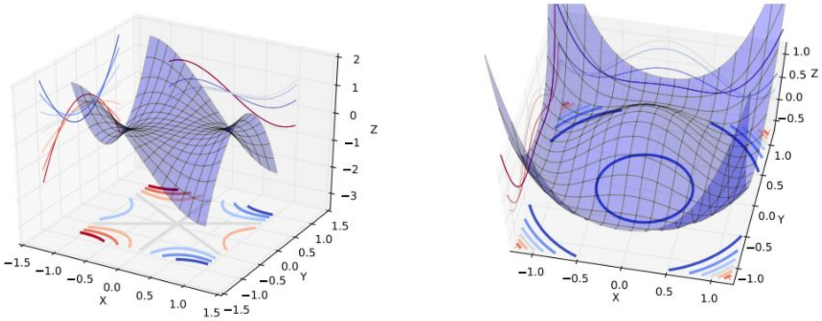
Optimization is as if the loss was convex.



Wojciech Czarnecki, Simon Osindero, Razvan Pascanu, Max Jaderberg,A neural network's loss surface contains every low dimensional pattern

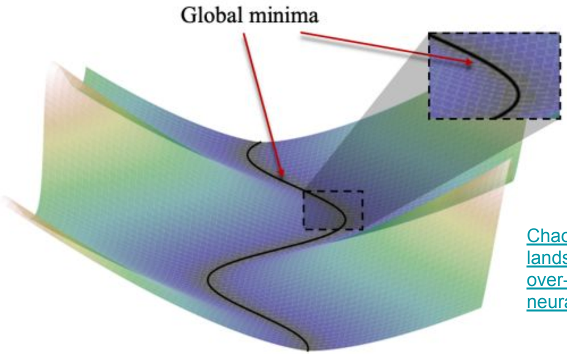
Grzegorz Swirzcz, Wojciech Czarnecki, Razvan Pascanu, Local minima in training neural networks, 2017



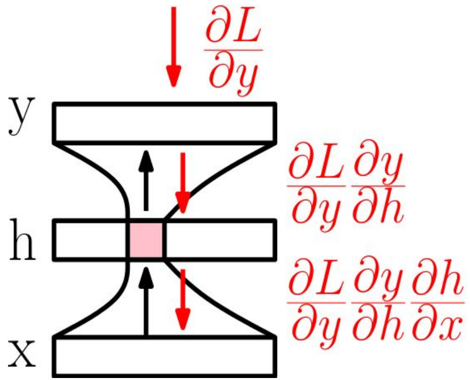


(c) (d)

https://arxiv.org/pdf/1406.2572.pdf



Chaoyue Liu, Libin Zhu, Mikhail Belkin, Loss landscapes and optimization in over-parameterized non-linear systems and neural networks

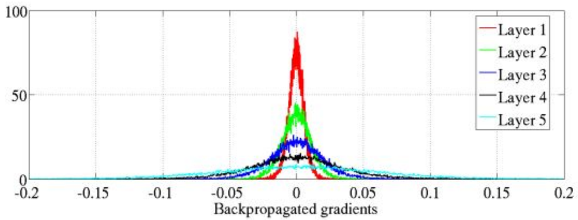


- It is always useful to step back, and don't think of your neural network as a function. Look at its structure, and at how gradients propagate.

Looking back at activations

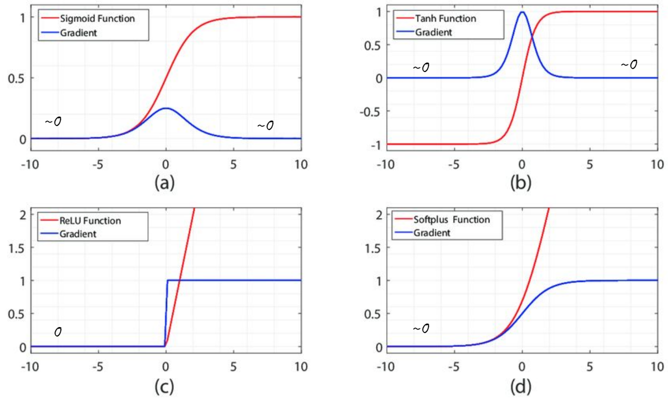
- One can reason about gradient propagation (saturated regimes)
- Still an active area of research (e.g. [Gu et al. 2020](#))

Initialization (and gradient propagation)



[Glorot and Bengio AISTATS 2010](#)

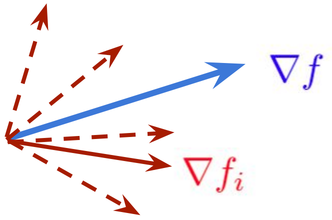
[He et al. ICCV 2015](#) adds a correction for ReLU.



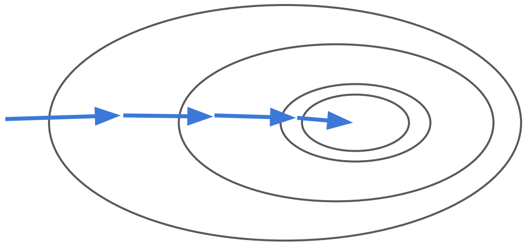
Stochastic GD vs Gradient descent: estimate true gradient by using a small subset of datapoints.

$$\nabla f \stackrel{\text{def}}{=} \sum_i \nabla f_i$$

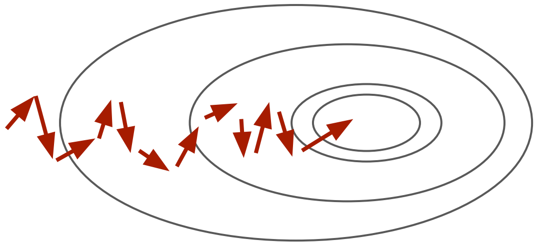
$$i \sim \mathcal{U}[1, N] \rightarrow \nabla f \approx \nabla f_i$$

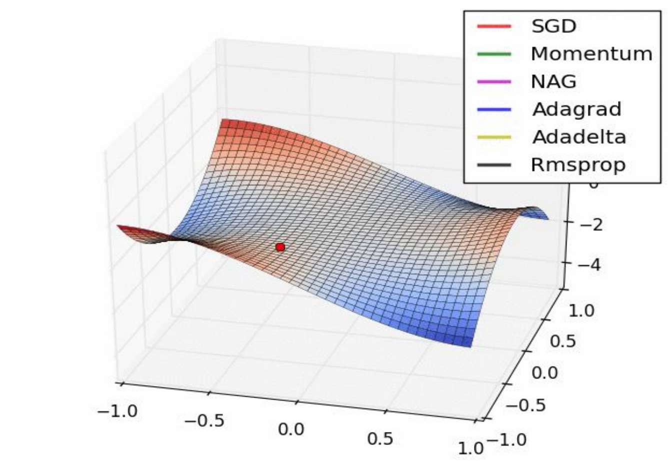


Gradient Descent



Stochastic Gradient Descent



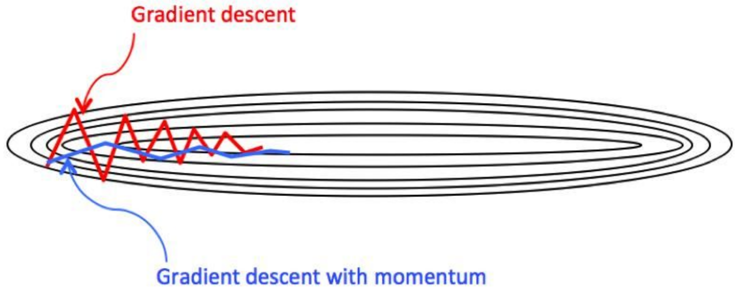


https://medium.com/@ramrajchandradevan/the-evolution-of-gradient-descend-optimization-algorithm-4106a6702d39

Momentum

$$V_t = \alpha V_{t-1} + (1-\alpha)\nabla f_i$$

$$\theta_t = \theta_{t-1} - \gamma V_t$$

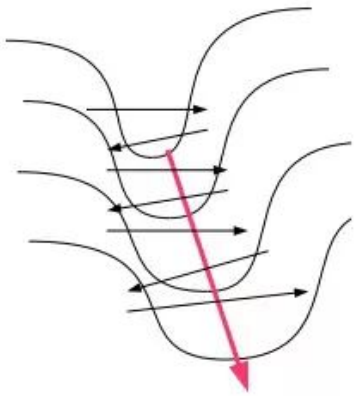


Rmsprop / ~Adam

$$V_t = \alpha V_{t-1} + (1-\alpha)\nabla f_i$$

$$S_t = \beta S_{t-1} + (1-\beta)(\nabla f_i)^2$$

$$\theta_t = \theta_{t-1} - \gamma \frac{V_t}{\sqrt{S_t + \epsilon}}$$

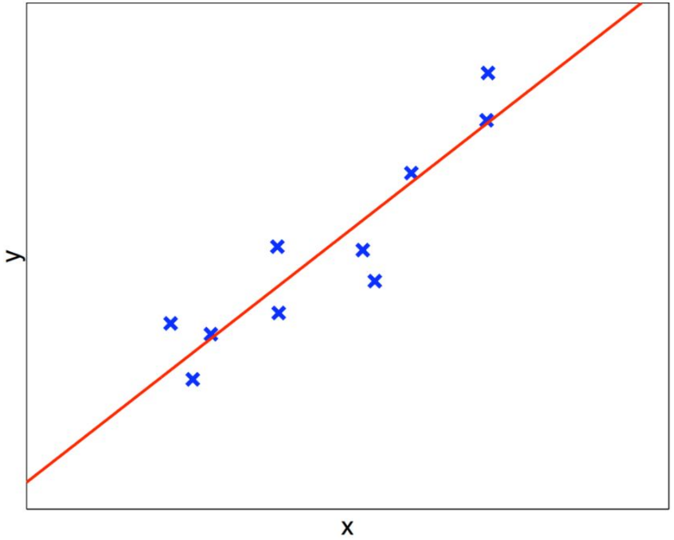


Main idea:
Approximate curvature (cheaply).

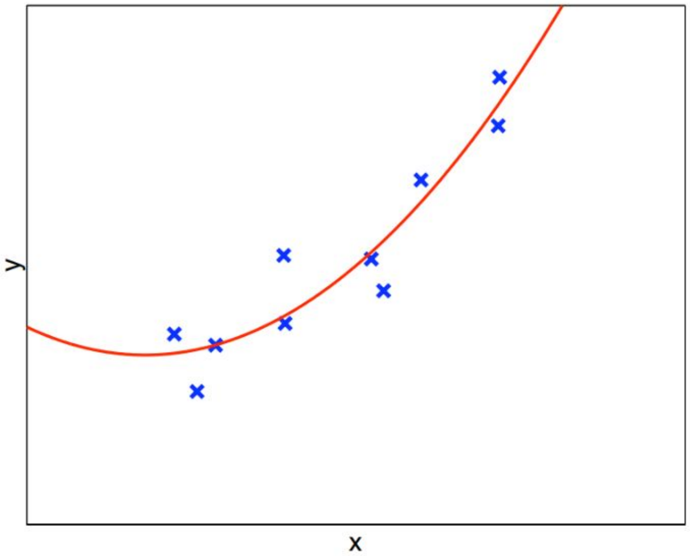
Learning / Generalization

Example: Data and best linear hypothesis

$$y = 1.60x + 1.05$$

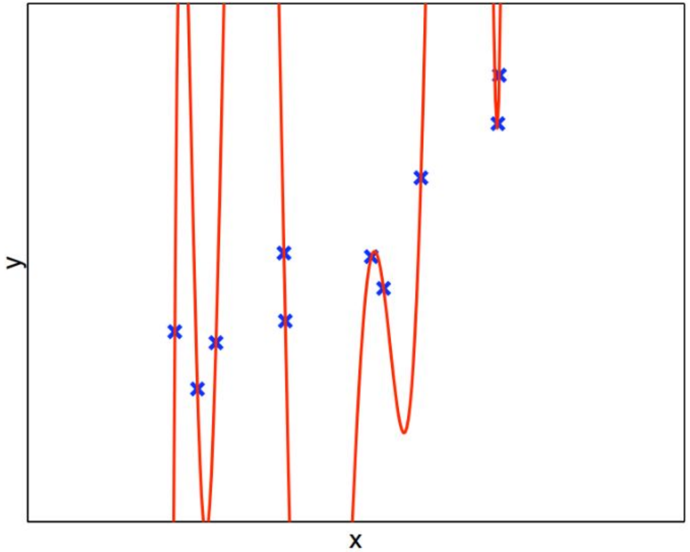


Order-2 fit

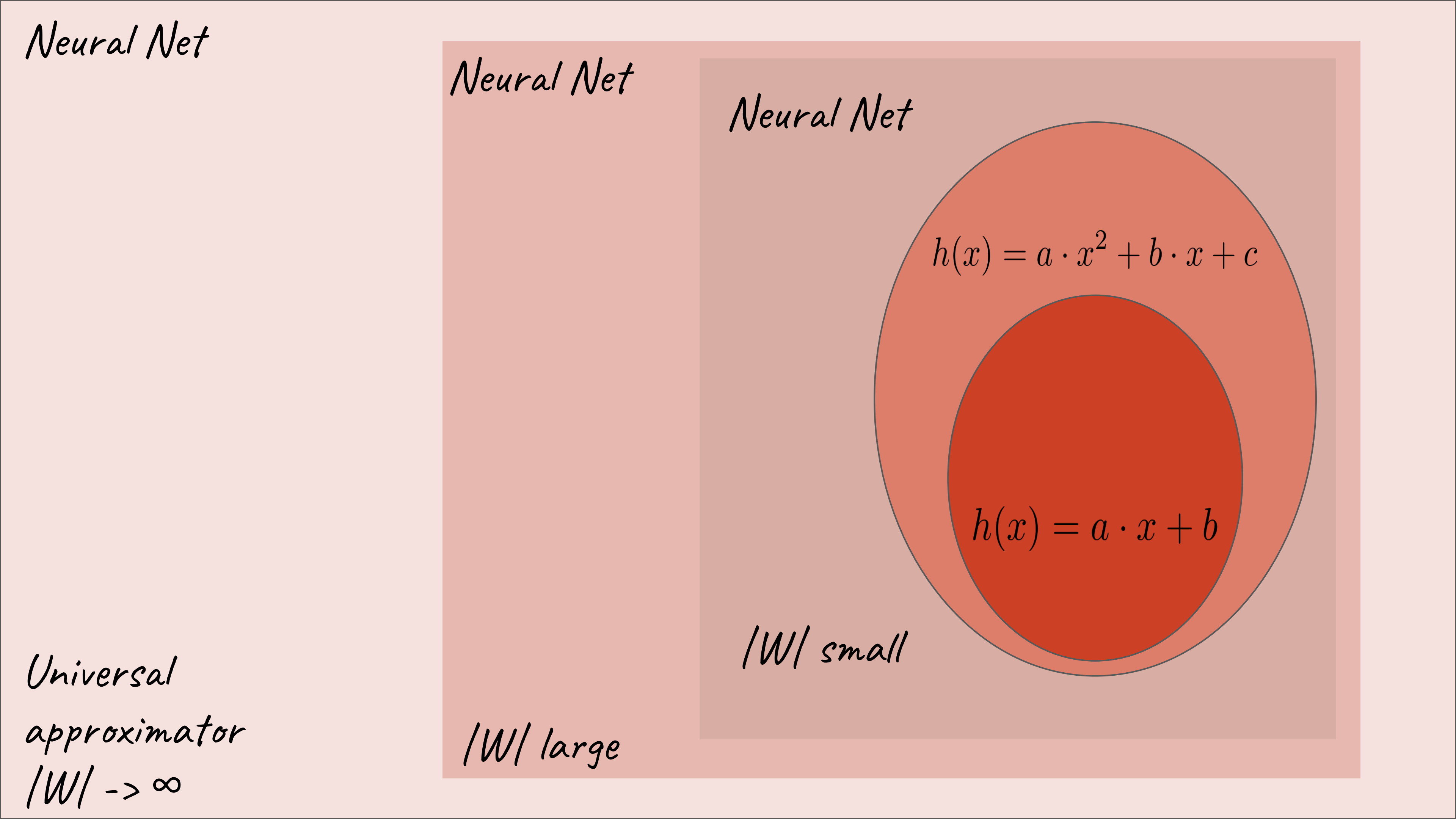


Is this a better fit to the data?

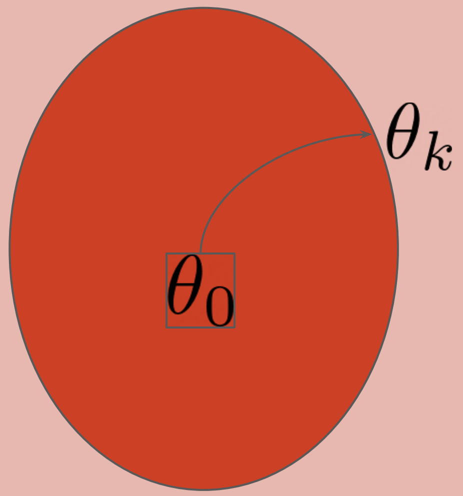
Order-9 fit



Is this a better fit to the data?



Neural Net
Universal approximator
|W| -> ∞
Neural Net
|W| large
Neural Net
|W| small
$h(x) = a \cdot x^2 + b \cdot x + c$
$h(x) = a \cdot x + b$



Universal approximator

UNDERSTANDING DEEP LEARNING REQUIRES RETHINKING GENERALIZATION

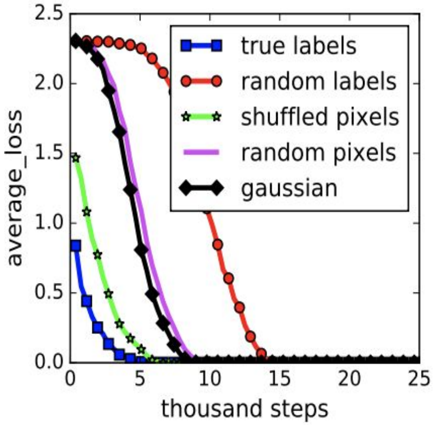
Chiyuan Zhang[*]
Massachusetts Institute of Technology
chiyuan@mit.edu

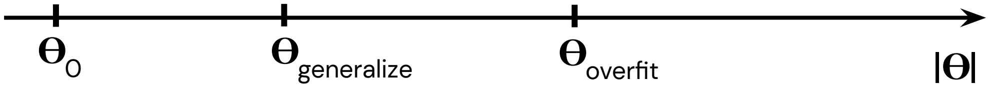
Samy Bengio
Google Brain
bengio@google.com

Moritz Hardt
Google Brain
mrtz@google.com

Benjamin Recht[†]
University of California, Berkeley
brecht@berkeley.edu

Oriol Vinyals
Google DeepMind
vinyals@google.com



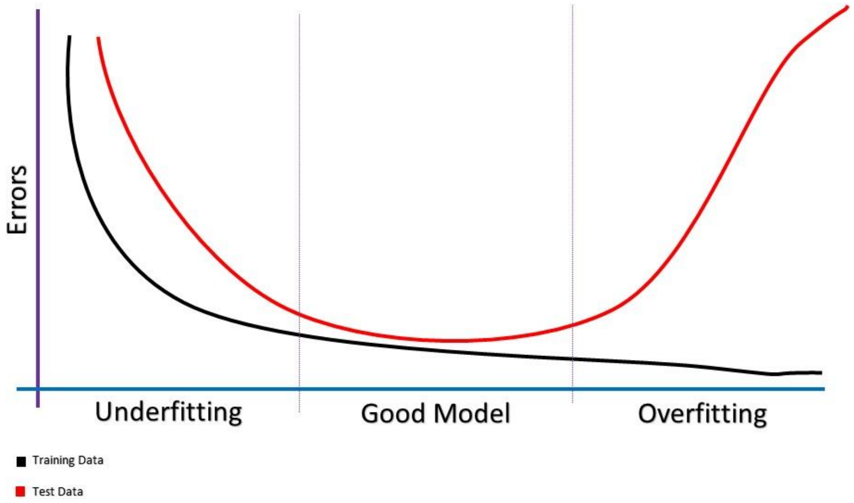


- *We want to be able* ***to generalize***

Use

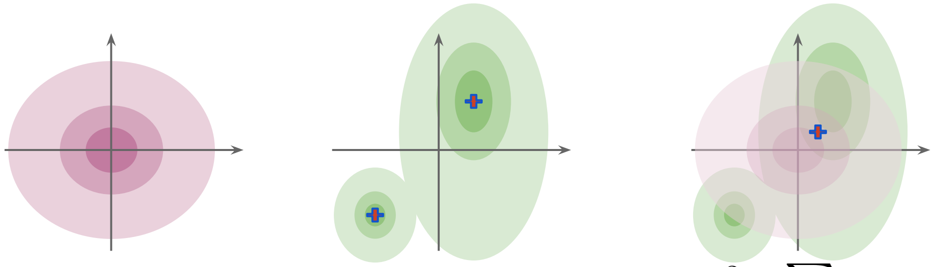
Training set: data used for finding the right parameters

Validation set: data used to estimate true loss on unseen data

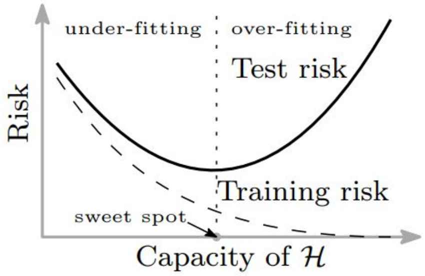


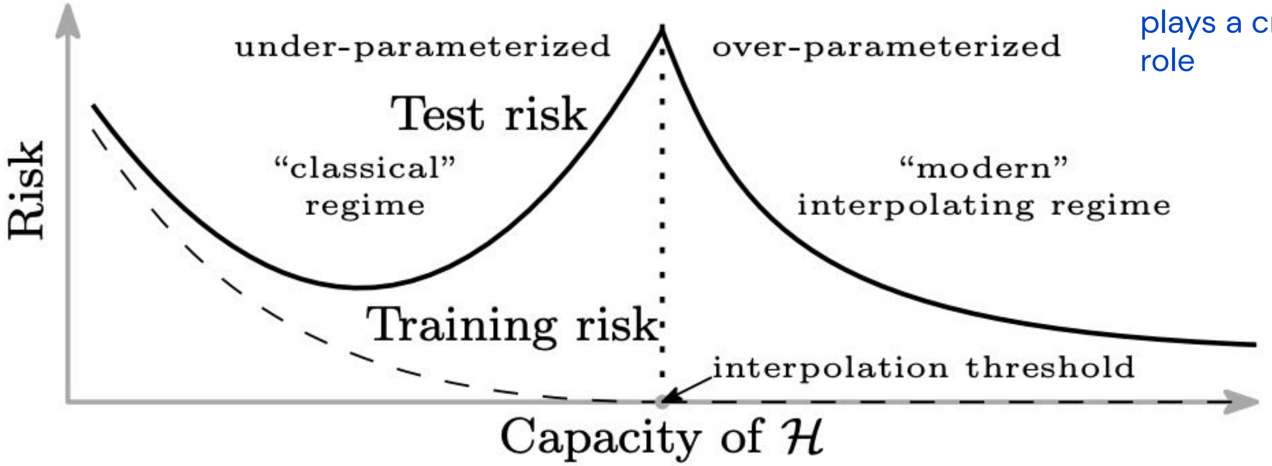
Learning is about minimizing an intractable function via optimizing a tractable approximation of it

- Priors/regularization terms and inductive biases provide mechanisms to introduce knowledge in the learning problem
- It restricts the search space for the parameters of the model
- They can take various forms:
  - Parametrization restricts the search space, making certain solutions unrepresentable
  - Or it can make the loss surface such that certain solutions are easier to find given initial conditions (e.g. what is the role of depth in deep learning)



$\|\theta\|^2$ $\qquad$ $\sum_i [h(x_i) - y_i]^2$ $\qquad$ $\gamma\|\theta\|^2 + \sum_i [h(x_i) - y_i]^2$





Initialization plays a crucial role

Model cannot achieve zero training loss

Multiple solutions with zero training loss

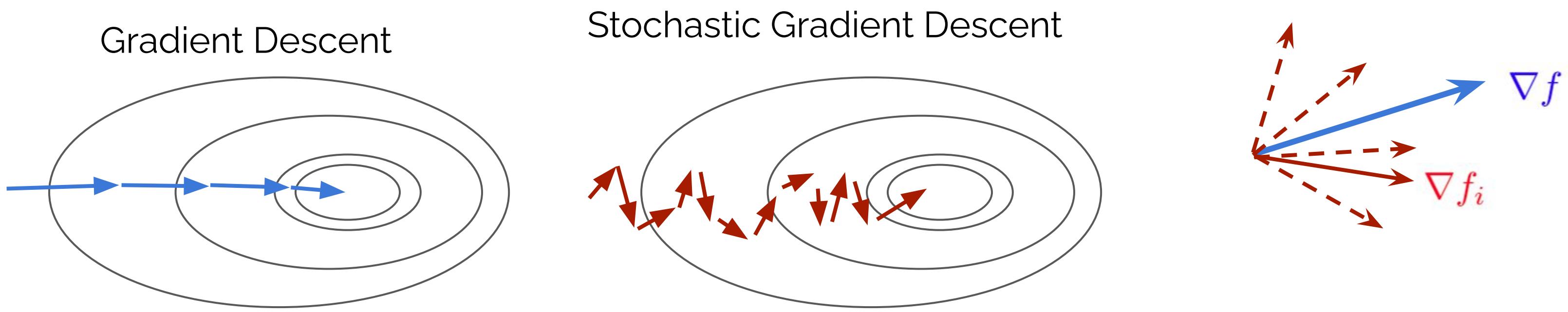
As capacity increases, the <u>best available solution is increasingly complex</u>

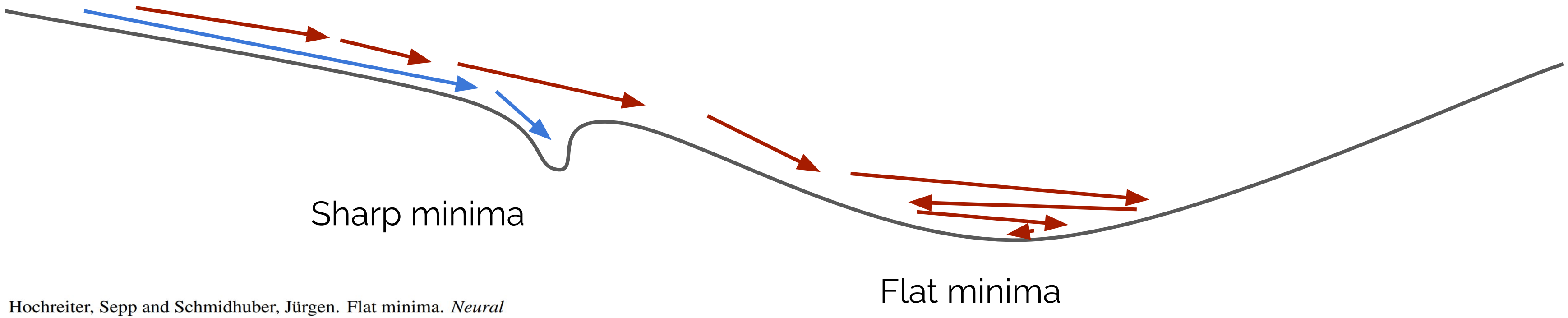
Already many solutions available, <u>optimization selects the first solution encountered</u>

Exactly one solution with zero training loss

Double descent curve from *Reconciling modern machine learning practice and the bias-variance trade-off* Belkin et al.
https://arxiv.org/abs/1812.11118

Implicit regularization of GD & SGD (helpful noise)

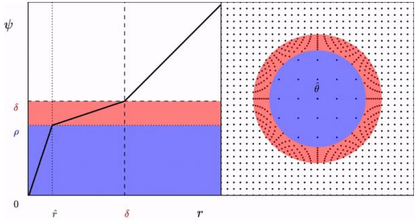




Hochreiter, Sepp and Schmidhuber, Jürgen. Flat minima. *Neural Computation*, 9(1):1–42, 1997.

More recently (ICLR'17) arXiv:1609.04836

We can change flatness (largest eigenvalue) without changing the function !
We need a more robust measure of flatness
https://arxiv.org/pdf/1703.04933.pdf



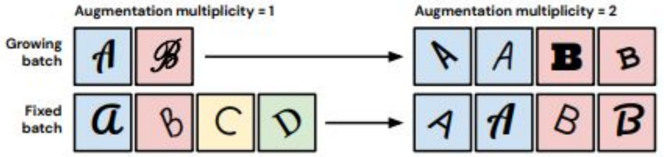
Implicit bias of SGD / GD

https://arxiv.org/abs/2009.11162
https://openreview.net/forum?id=rq_Qr0c1Hyo

$$\tilde{C}_{SGD}(\omega) = C(\omega) + \frac{\epsilon}{4m} \sum_{k=0}^{m-1} ||\nabla \hat{C}_k(\omega)||^2.$$

Punch line: SGD optimizes a different objective than GD!

Not all noise is equal !
https://arxiv.org/abs/2105.13343
(Noise from data augmentation hurts generalization, while noise from data sampling helps)

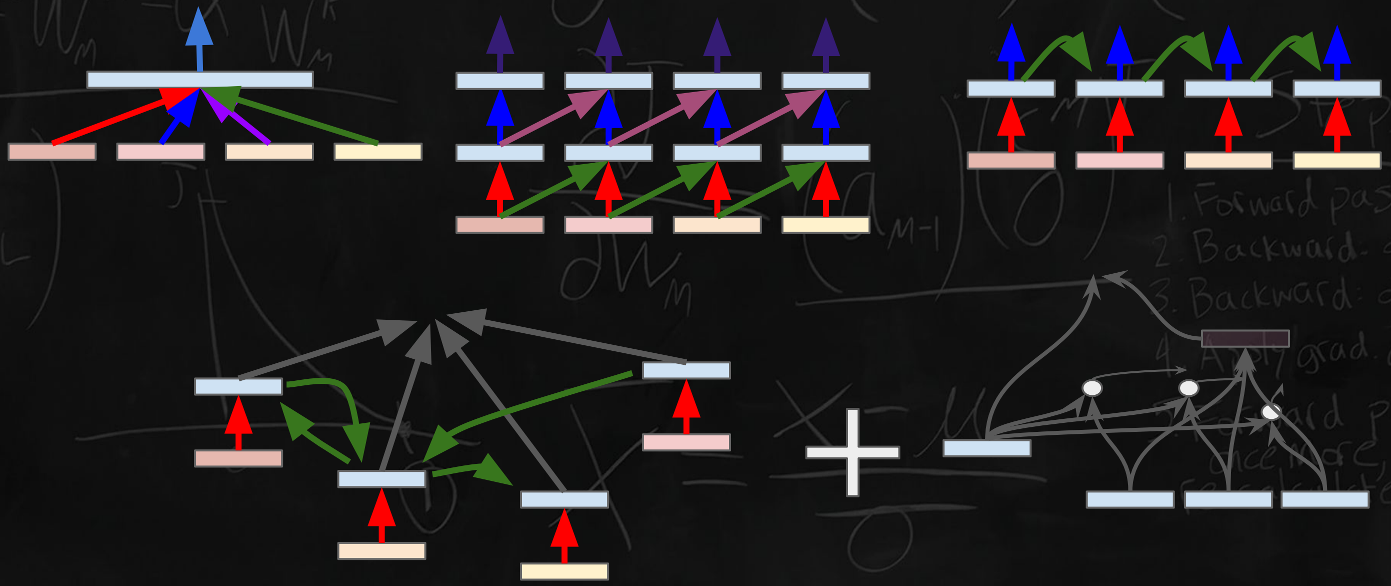


- We need Out-of-Distribution Generalization in order to solve tasks (not fit datasets). However IMHO we need to define properly what we want as I believe there is a lot of confusion (and misunderstanding) in this space. But is at the core of understanding what learning can do!

Summary

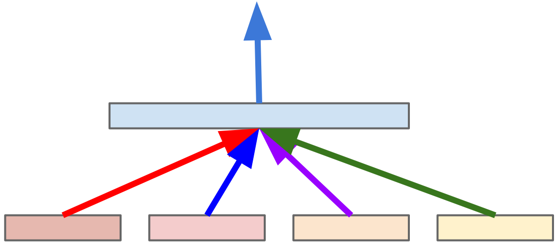
- *DL* corresponds to a family of parametric models (neural networks) optimized by gradient descent techniques
- NN are typically layers of linear projects (sometimes with additional constraints) followed by non-linearities
- They operate in an over-parametrized regime more often than note
- Magic sauce of DL is restricting the search in the learning process to get to solutions that generalize. This is done by architecture design, choice of loss function and alterations of the optimizer.

Taxonomy of Deep Learning Architectures



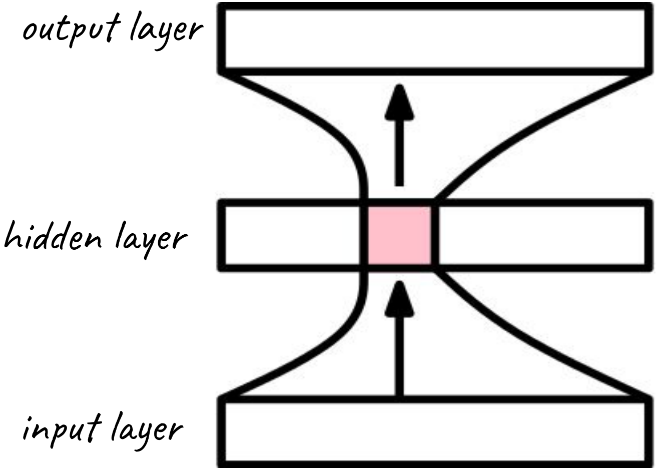
Resources

- Abundance of resources online showing how to instantiate typical architectures using pytorch / jax / tensorflow
- Some examples:
  - EEML practical sessions:
    https://github.com/eemlcommunity/PracticalSessions2021
    https://github.com/eemlcommunity/PracticalSessions2020
    https://github.com/eemlcommunity/PracticalSessions2019
  - https://github.com/deepmind/educational
  - https://roberttlange.github.io/posts/2020/03/blog-post-10/
  -

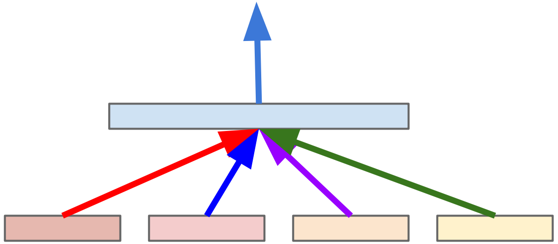


MLP

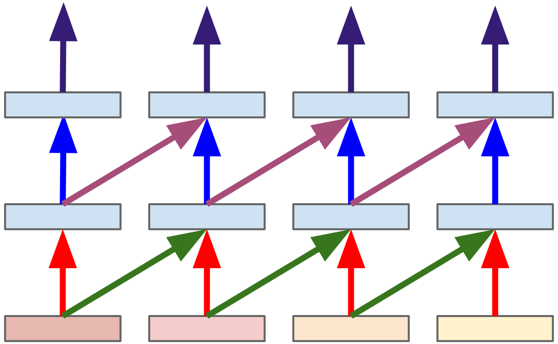
$$h(\mathbf{x}, \theta) = \mathbf{W}_{out} ReLU(\mathbf{W}_{in}\mathbf{x} + \mathbf{b}_{in}) + \mathbf{b}_{out}$$



- Versatile and important/heavily used architecture
- No structure for the data
- Relies on the bias introduced by depth



MLP

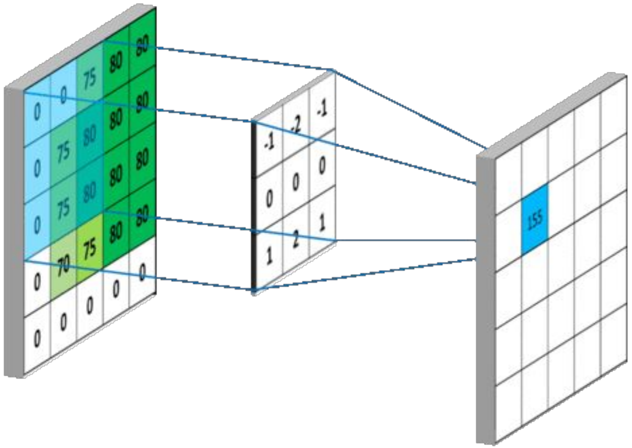


Convolutional Network

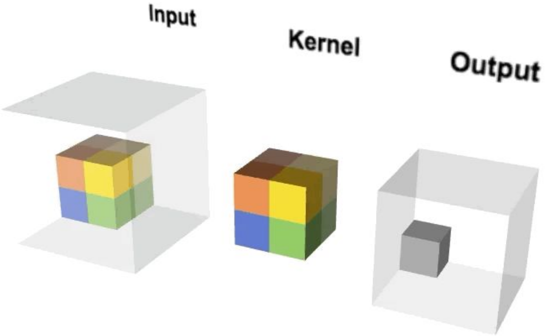
Convolutional Neural Networks

Convolutional Neural Networks

- **Structural prior:** spatial neighbourhood defines the role of a pixel
- Apply **same function** at all position
- Induces translation invariance as features are computed independent of position



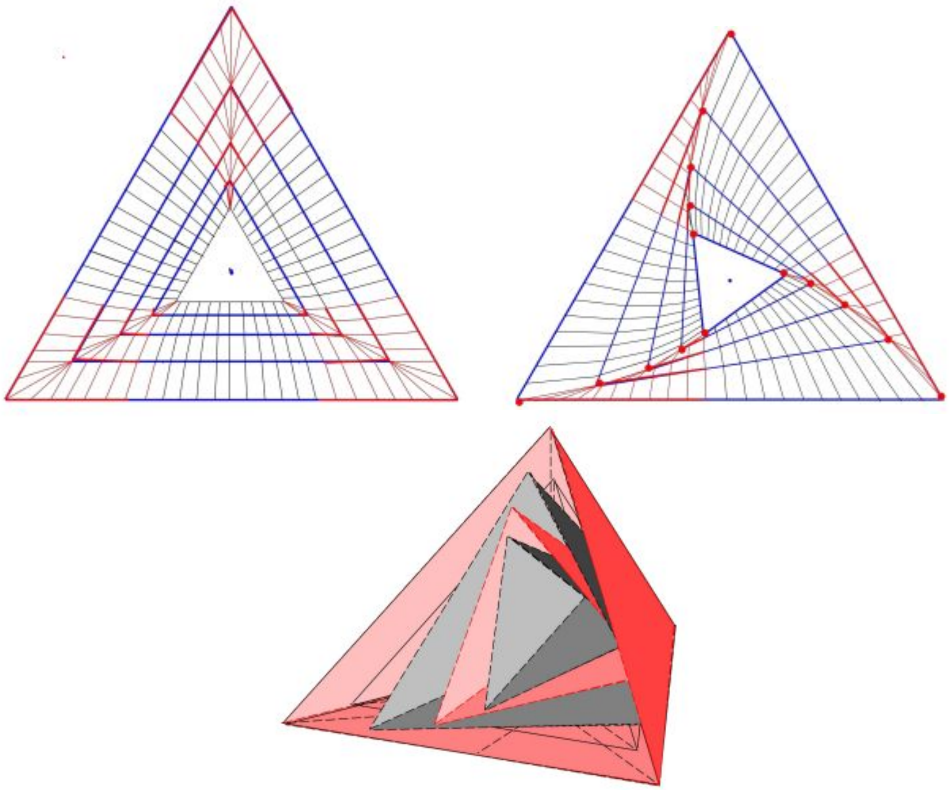
https://www.analyticsindiamag.com/convolutional-neural-network-image-classification-overview/



https://medium.com/apache-mxnet/1d-3d-convolutions-explained-with-ms-excel-5f88c0f35941

Can an MLP reproduce a ConvNet?

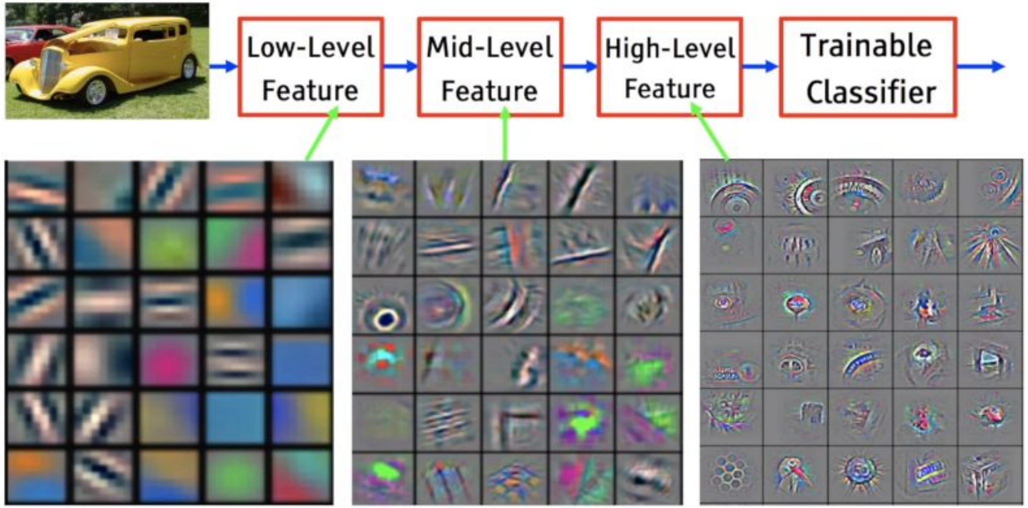
Stefan Carlsson (KTH): https://arxiv.org/abs/1905.08922



Can an MLP reproduce a ConvNet?

- Yes, and you end up with circular matrices

$$
\begin{pmatrix}
w_1 & w_2 & \dots & w_k & 0 & \dots & & 0 \\
0 & w_1 & w_2 & \dots & w_k & 0 & \dots & 0 \\
0 & 0 & w_1 & w_2 & \dots & w_k & 0 \ \dots & 0 \\
 & & & \vdots & & & & \\
0 & \dots & 0 & w_1 & w_2 & \dots & & w_k \\
\hline
w_k & 0 & \dots & 0 & w_1 & w_2 & \dots & w_{k-1} \\
w_{k-1} & w_k & 0 & \dots & 0 & w_1 & w_2 \ \dots & w_{k-2} \\
 & & \vdots & & & & & \\
w_2 & \dots & w_k & 0 & & \dots & 0 & w_1
\end{pmatrix}
$$

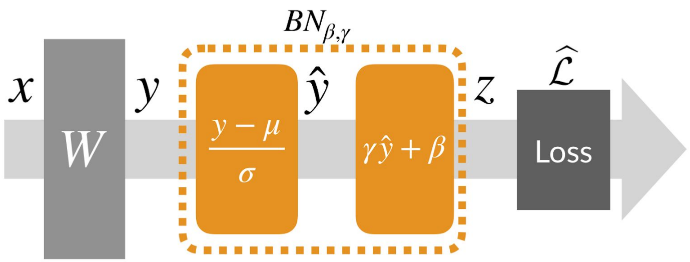


- Pooling is not typically used, but is a great tool to reduce computation
- Large optimization space, usually tuned on ImageNet and relying on intuition
- Bigger is better (though one hits diminishing returns)

- One typically stays close to an existing architecture:
    - AlexNet
    - VGG
    - Inception
    - **Resnet**
    - NFNet
    - MobileNet
    - DenseNet
    - etc.

Resnets - powering computer vision

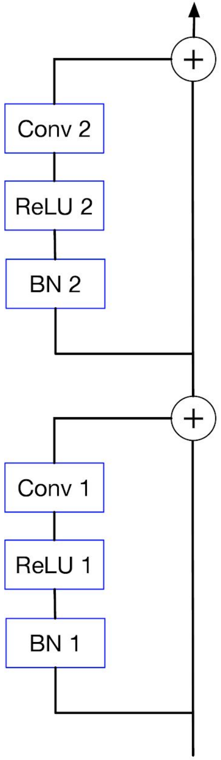
Batch normalized network

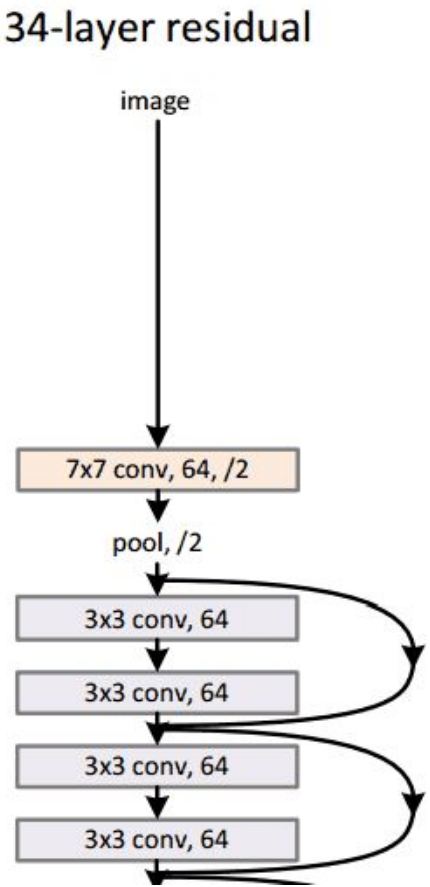


http://gradientscience.org/batchnorm/

Ioffe & Szegedy
https://arxiv.org/abs/1502.03167

He et al. 2015

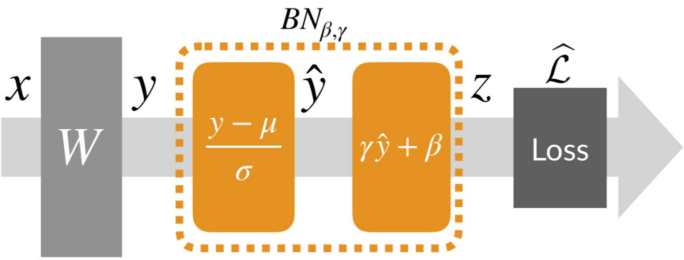




Initialization in Deep Learning

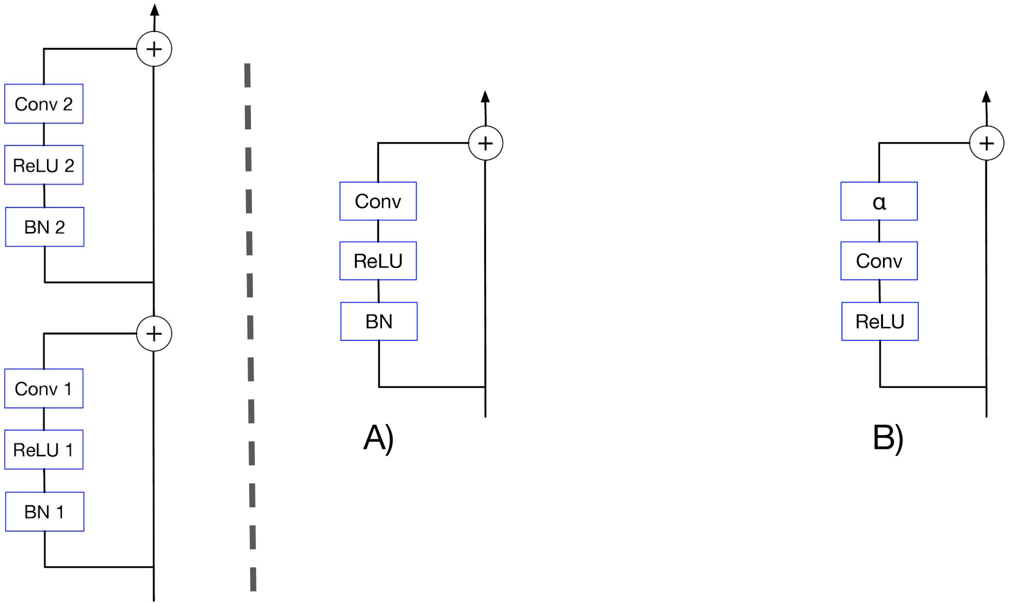
De & Smith: Batch Normalization Biases Deep Residual Networks Towards shallow paths https://arxiv.org/pdf/2002.10444.pdf
Stabilizing Transformer for RL: Parisotto et al. 2020

Batch normalized network

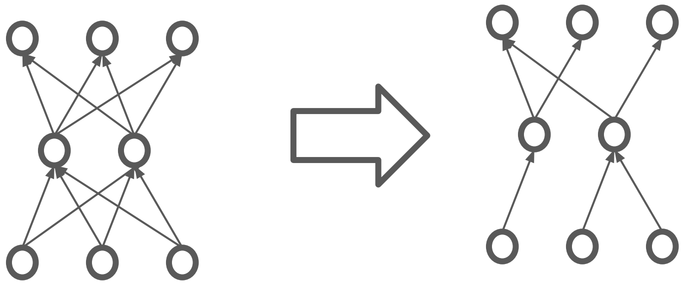


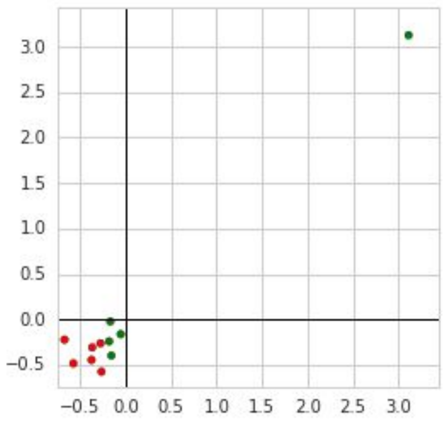
http://gradientscience.org/batchnorm/

Ioffe & Szegedy
https://arxiv.org/abs/1502.03167

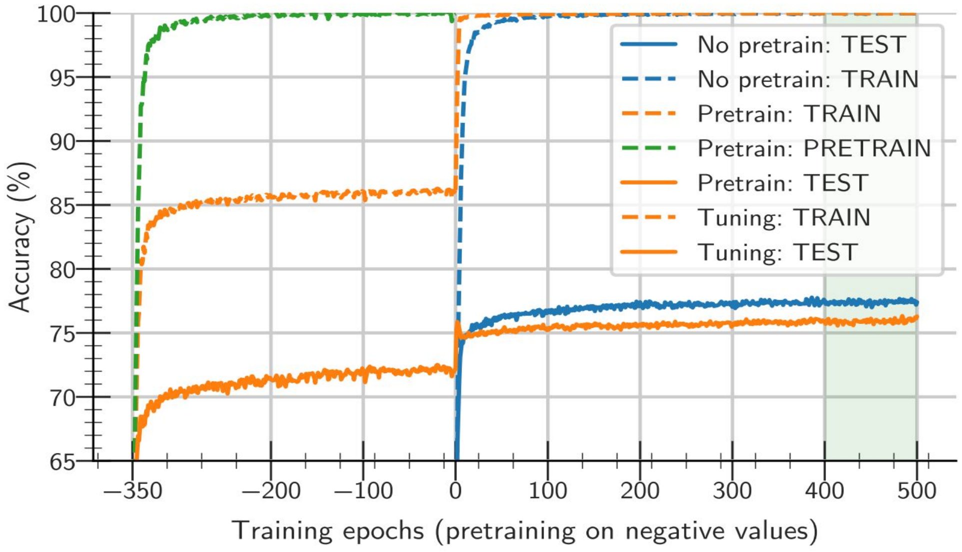


Reconciling modern machine learning practice and the bias-variance trade-off Belkin et al.
https://arxiv.org/abs/1812.11118
Jonathan Frankle, Michael Carbin 2018, The lottery ticket hypothesis



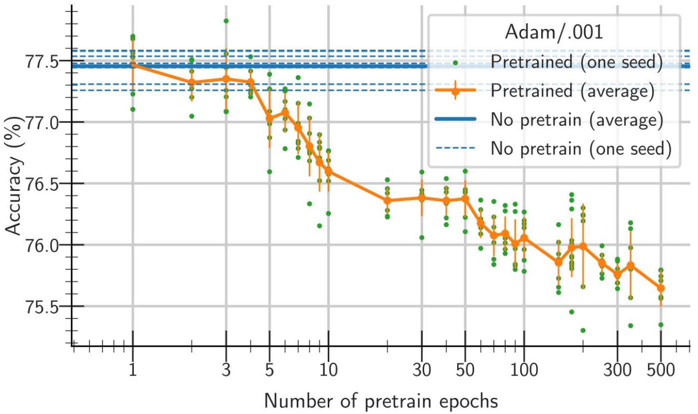


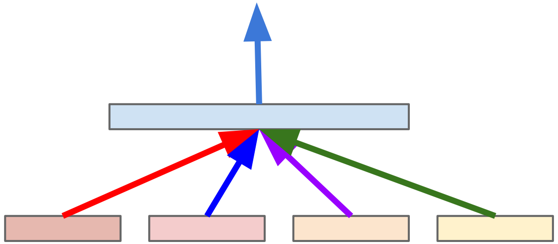
Grzegorz Swirzcz, Wojciech Czarnecki, Razvan Pascanu, Local minima in training neural networks, 2017



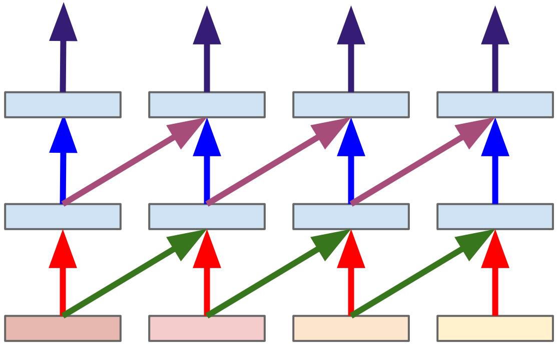
Berariu et al. 2020

Ash et al, 2019
On the difficulty of warm-starting neural network training

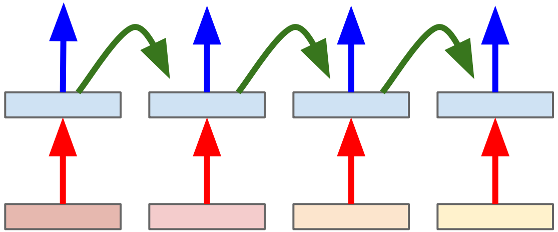




MLP

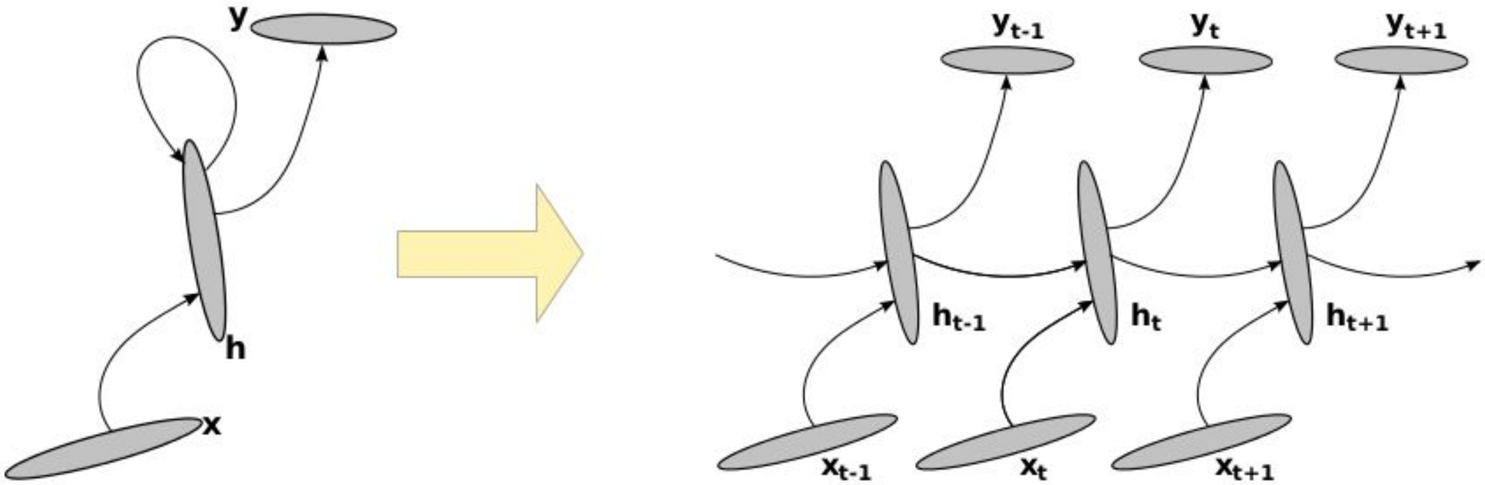


Convolutional Network



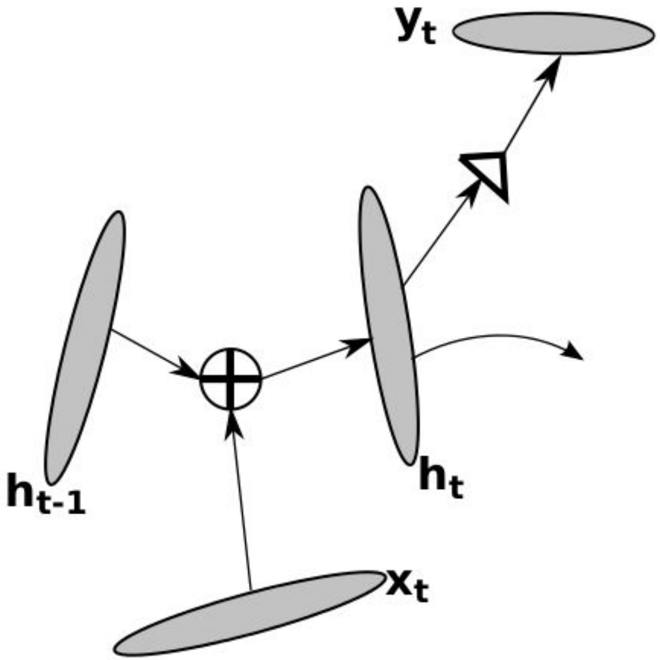
Recurrent Network

Recurrent Neural Networks



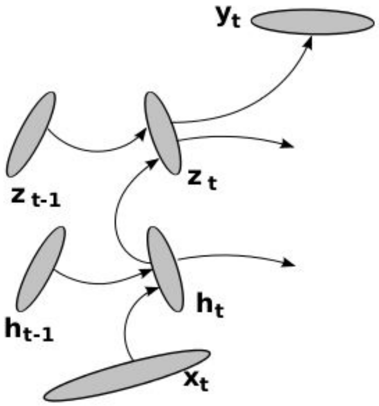
[Pascanu et al. 2014]

$$\mathbf{h}_t = f_h(\mathbf{x}_t, \mathbf{h}_{t-1})$$
$$\mathbf{y}_t = f_o(\mathbf{h}_t),$$

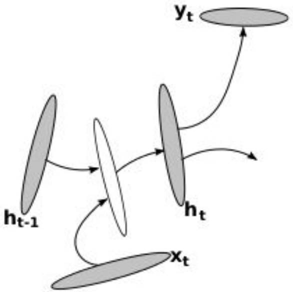


Pascanu et al. 2014

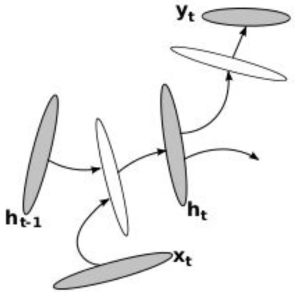
Deep RNNs..



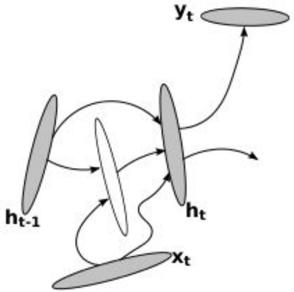
Stacked RNNs



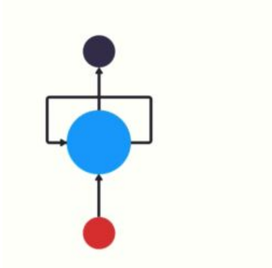
DT-RNN



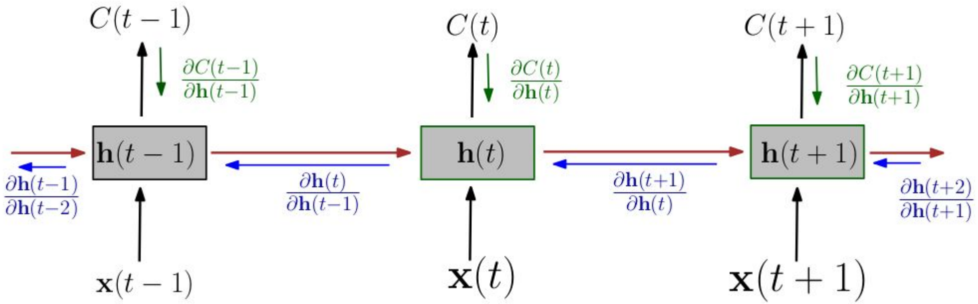
DOT-RNN



DOT(s)-RNN



Exploding Gradients

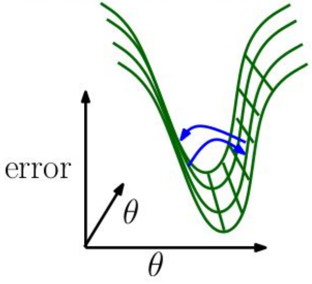


$$\frac{\partial C}{\partial \mathbf{W}} = \sum_t \frac{\partial C(t)}{\partial \mathbf{W}} = \sum_t \sum_{k=0}^{t} \frac{\partial C(t)}{\partial \mathbf{h}(t)} \frac{\partial \mathbf{h}(t)}{\partial \mathbf{h}(t-k)} \frac{\partial \mathbf{h}(t-k)}{\partial \mathbf{W}}$$

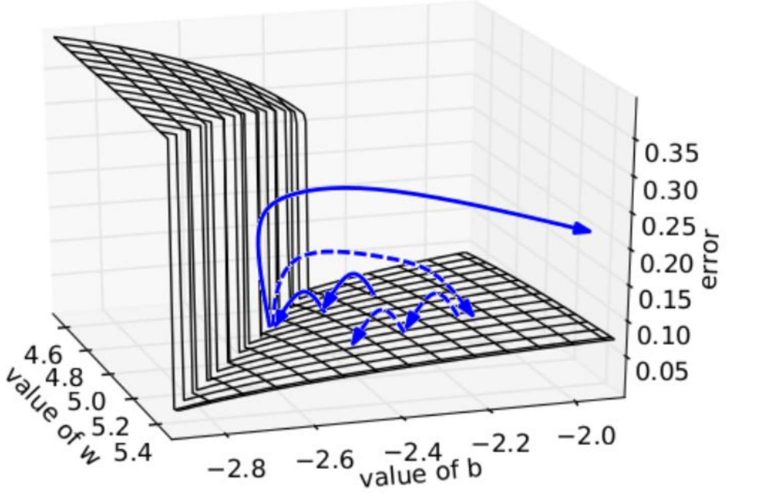
$$\frac{\partial \mathbf{h}(t)}{\partial \mathbf{h}(t-k)} = \prod_{j=k+1}^{t} \frac{\partial \mathbf{h}(j)}{\partial \mathbf{h}(j-1)}$$

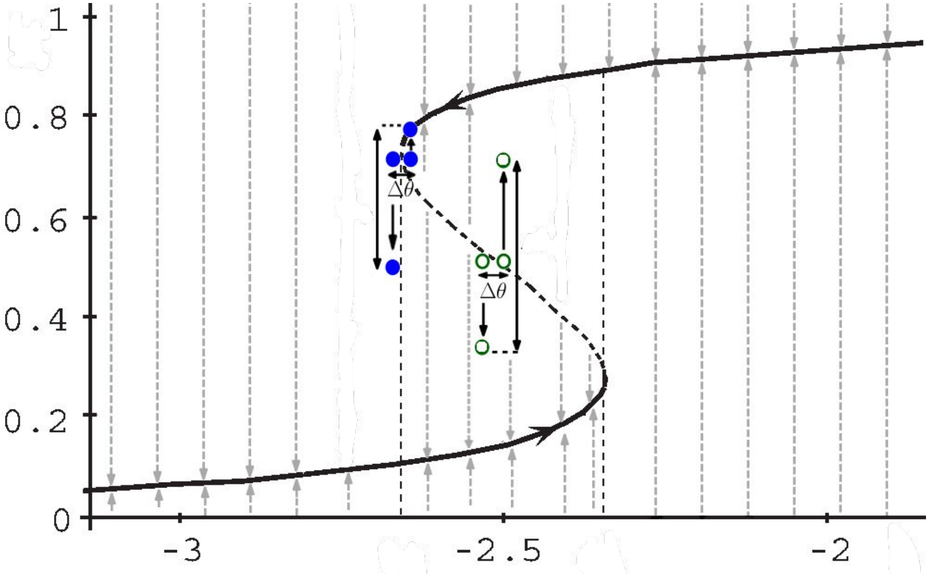
$\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$
if $\|\hat{\mathbf{g}}\| \geq threshold$ **then**
$\quad \hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|} \hat{\mathbf{g}}$
end if

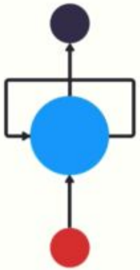
Classical view for:



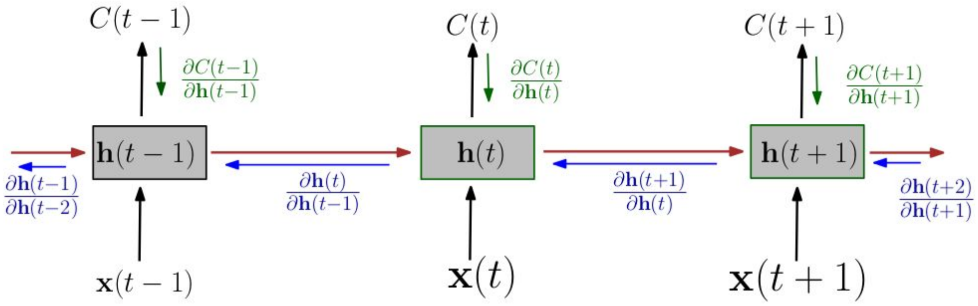
The error is $(h(50) - 0.7)^2$ for $h(t) = w\sigma(h(t-1)) + b$ with $h(0) = 0.5$





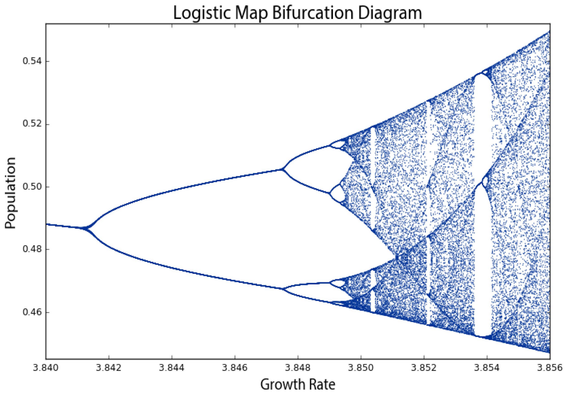


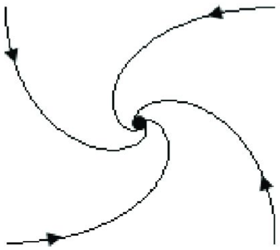
Vanishing Gradients



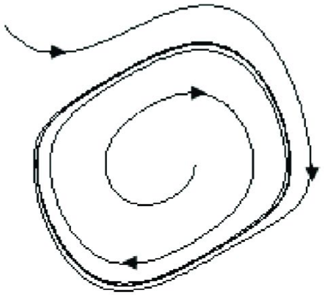
$$\frac{\partial C}{\partial \mathbf{W}} = \sum_t \frac{\partial C(t)}{\partial \mathbf{W}} = \sum_t \sum_{k=0}^{t} \frac{\partial C(t)}{\partial \mathbf{h}(t)} \frac{\partial \mathbf{h}(t)}{\partial \mathbf{h}(t-k)} \frac{\partial \mathbf{h}(t-k)}{\partial \mathbf{W}}$$

$$\frac{\partial \mathbf{h}(t)}{\partial \mathbf{h}(t-k)} = \prod_{j=k+1}^{t} \frac{\partial \mathbf{h}(j)}{\partial \mathbf{h}(j-1)}$$

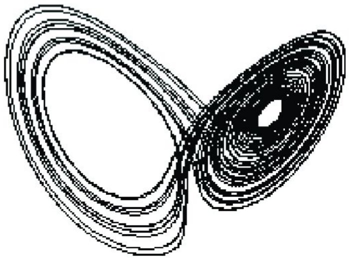




Fixed point attractor



Limit-cycle attractor



Strange attractor

- Weights largest eigenvalue < 1, damping regime [fixed point attractor]
- Weights close or 1, information travels through the system
- Weights largest eigenvalue > 1, potentially in a chaotic regime

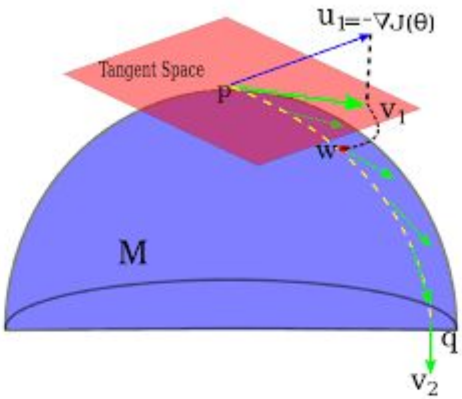
Echo State Network literature, e.g. : http://www.scholarpedia.org/article/Echo_state_network
Ilya Sutskever et al 2013, On the importance of initialization and momentum in deep learning

Saxe et al. 2014 :

- Orthogonal weights as solution for deep linear models

Henaff et al. 2016; Arjovski et al. 2016

- Reparametrize RNN so recurrent weights stays orthogonal



$$\Omega = \sum_k \Omega_k = \sum_k \left(\frac{\left\| \frac{\partial C}{\partial \mathbf{h}_{k+1}} \frac{\partial \mathbf{h}_{k+1}}{\partial \mathbf{h}_k} \right\|}{\left\| \frac{\partial C}{\partial \mathbf{h}_{k+1}} \right\|} - 1 \right)^2$$

Pascanu et al. 2012

But we do need to forget !?

$$\frac{\partial(x_1 + x_2 + x_3)}{\partial x_3} = 1$$

$$x_1 + x_2 + x_3 = 10$$

$$x_3 = ?$$

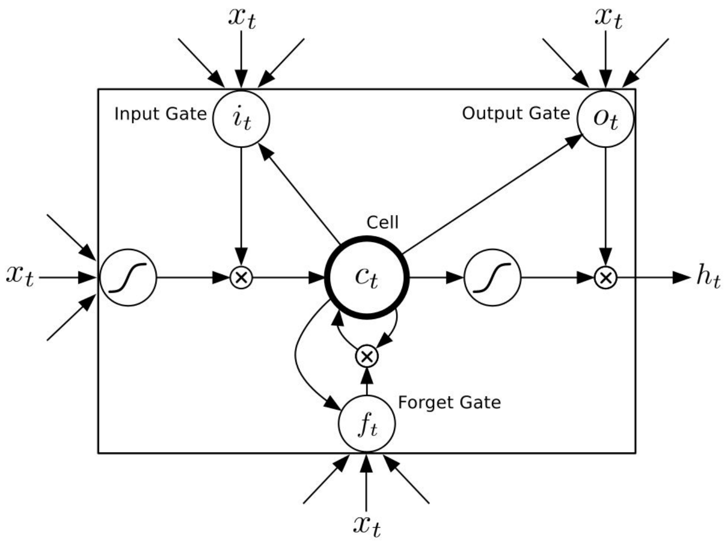
$$i_t = \sigma\left(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i\right)$$

$$f_t = \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f\right)$$

$$c_t = f_t c_{t-1} + i_t \tanh\left(W_{xc}x_t + W_{hc}h_{t-1} + b_c\right)$$

$$o_t = \sigma\left(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o\right)$$

$$h_t = o_t \tanh(c_t)$$



[Hochreiter et al. 1997](#)
[Graves 2013](#)

The gates dilate and contracts time, similar to a low-pass filter in typical signal processing.

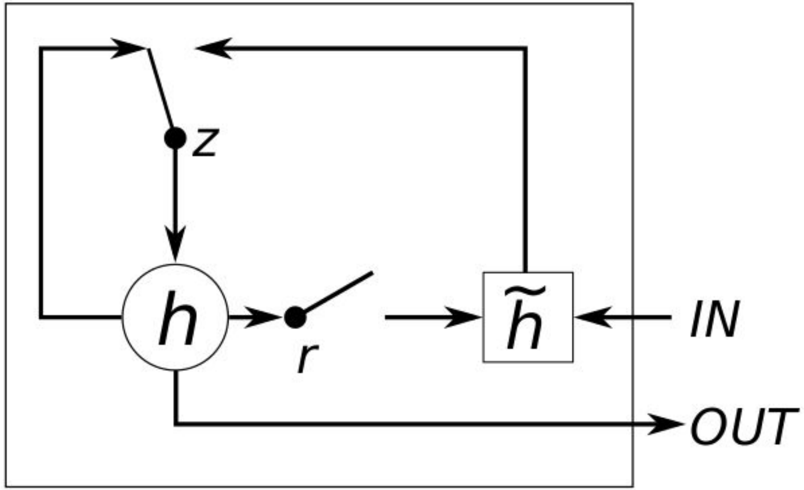
[Chung et al. 2015]

$$z = \sigma(W_z x_t + U_z h_{t-1})$$

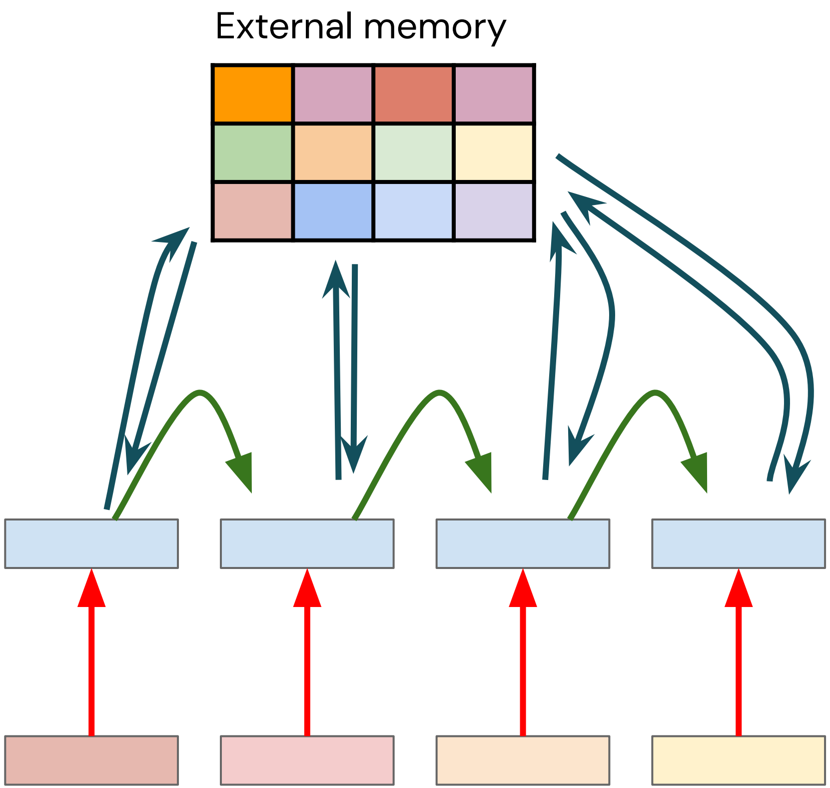
$$r = \sigma(W_r x_t + U_r h_{t-1})$$

$$\tilde{h} = tanh(W_h x_t + U_h(r \circ h_{t-1}))$$

$$h_t = (1 - z) \circ h_{t-1} + z \circ \tilde{h}$$



(b) Gated Recurrent Unit

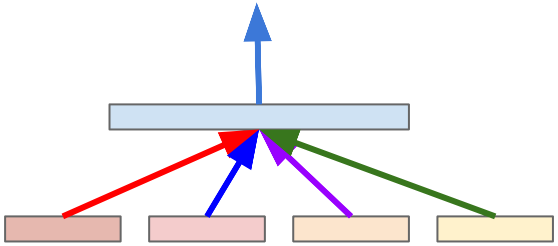


E.g.:

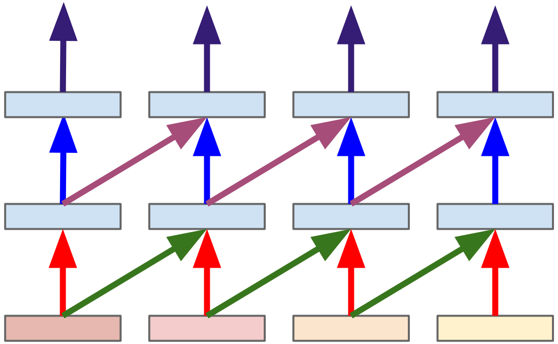
Memory networks:
https://arxiv.org/abs/1410.3916

Turing machines:
https://arxiv.org/abs/1410.5401

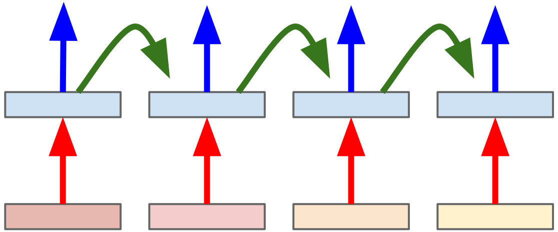
Can be extended towards Complementary Learning Systems (mix of non-parametric & parametric models)



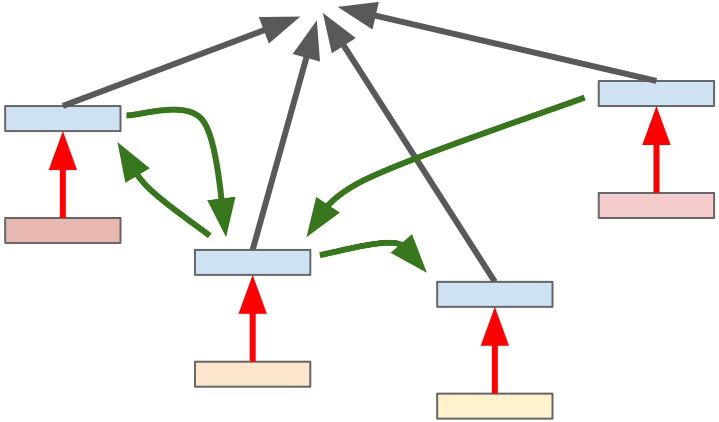
MLP



Convolutional Network



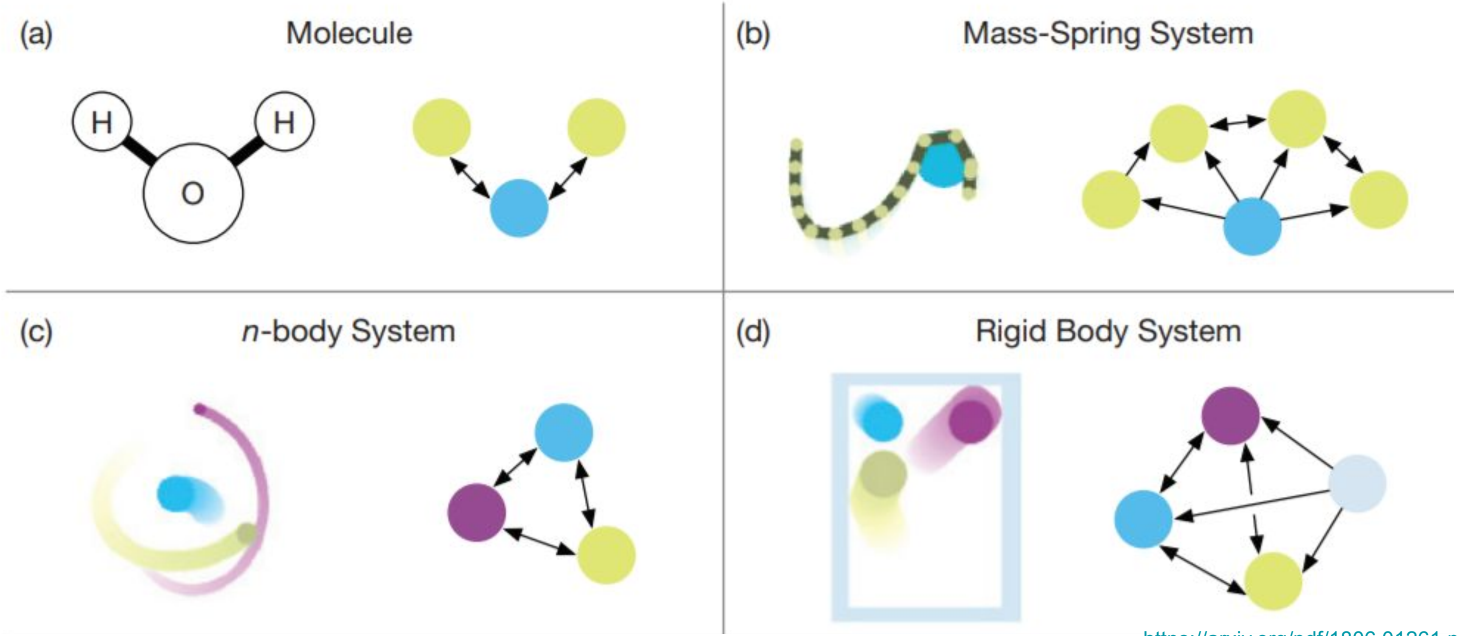
Recurrent Network



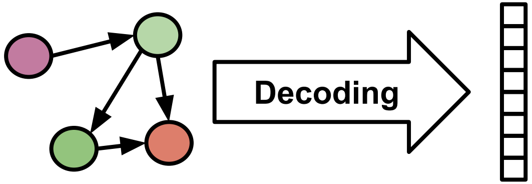
Graph Neural Networks

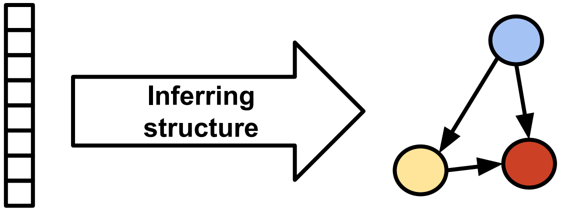
GraphNetworks

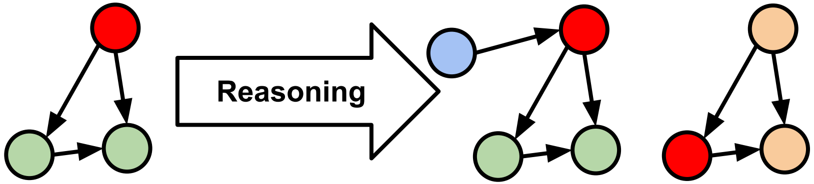
"... infinite use of finite means"



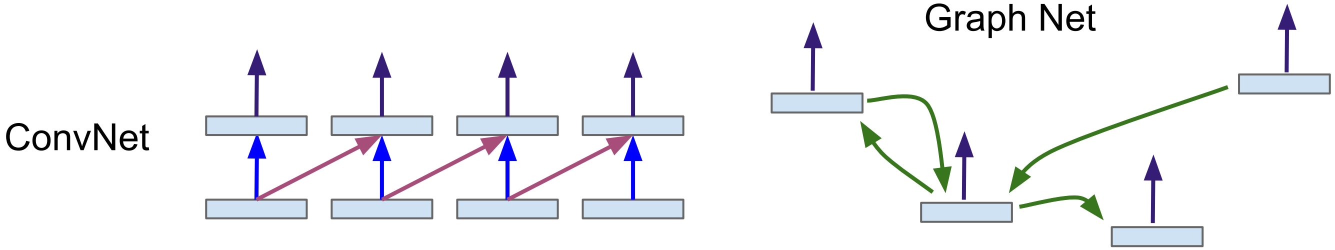
https://arxiv.org/pdf/1806.01261.pdf



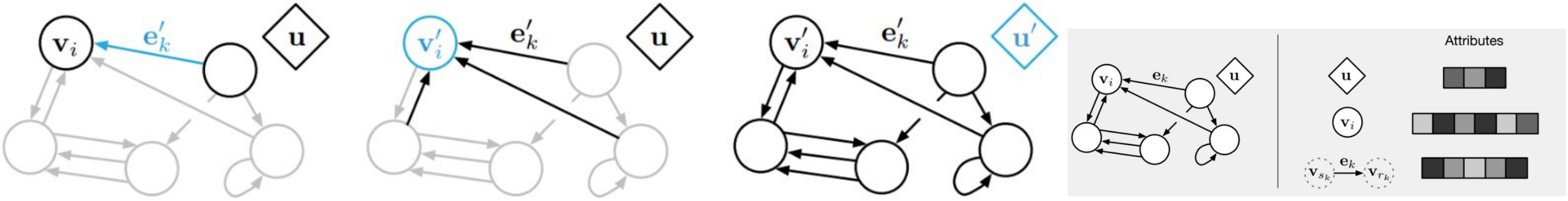




- Allows for adding inductive bias in the process through the structure of the graph
- Fast growing field of research with still many open questions
- Might provide a reliable connection to classical AI (e.g. logic) and to algorithms

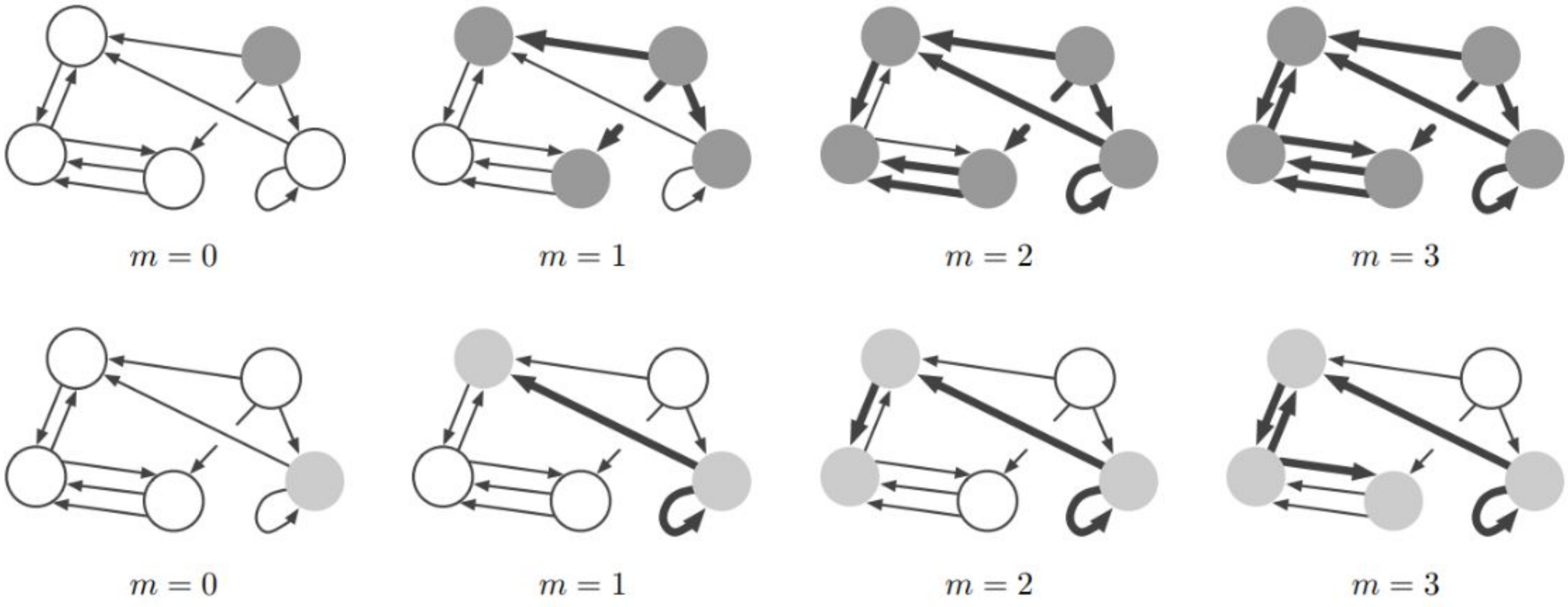


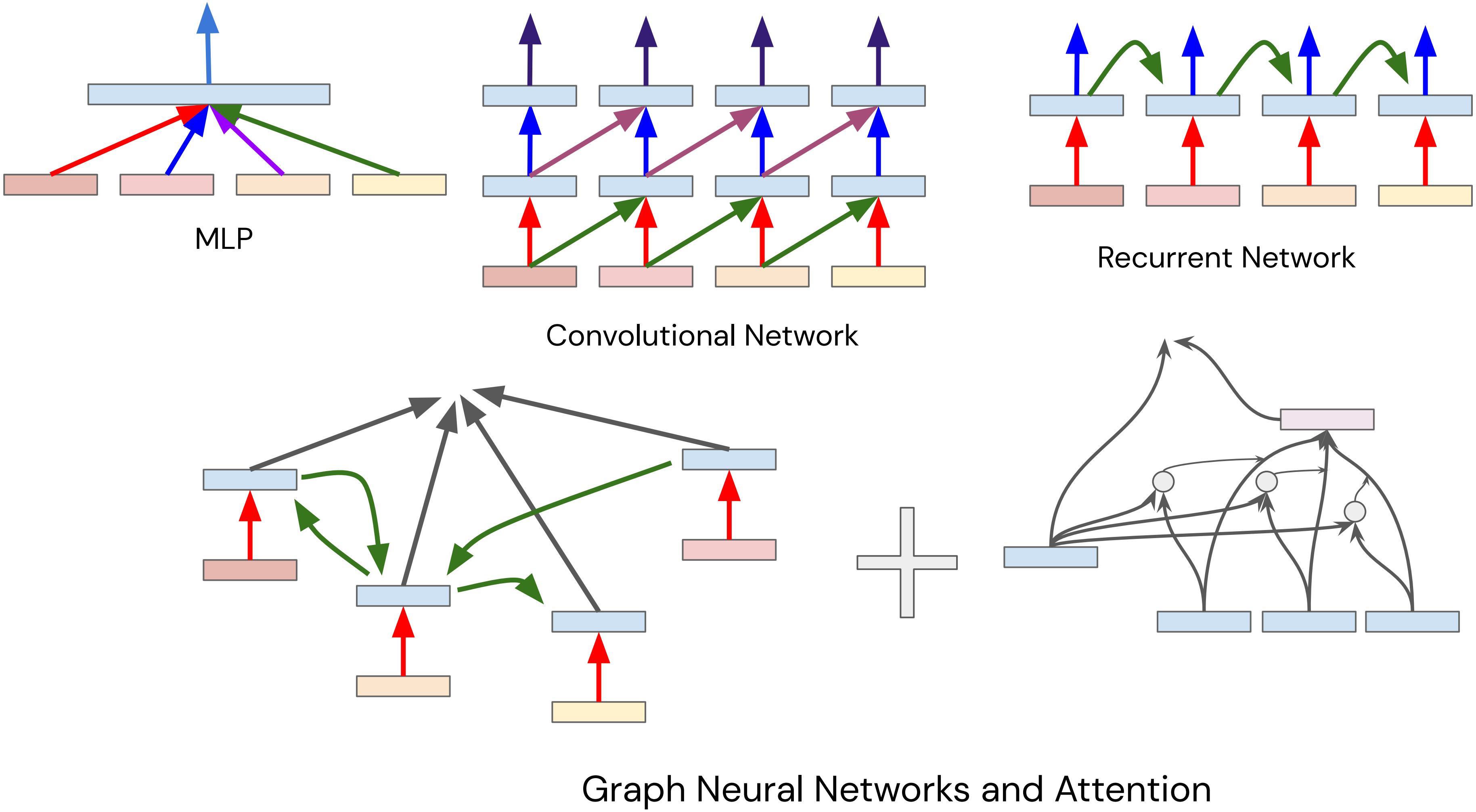
- Similar to convnets consider your input divided into items (objects) -- pixels for CNN
- Define a neighborhood (which nodes are connected to which)
- The (new) state of the node is a function of its neighbourhood, **but** it should be order invariant wrt to the elements in the neighborhood (by using commutative operations like summation)



(a) Edge update (b) Node update (c) Global update

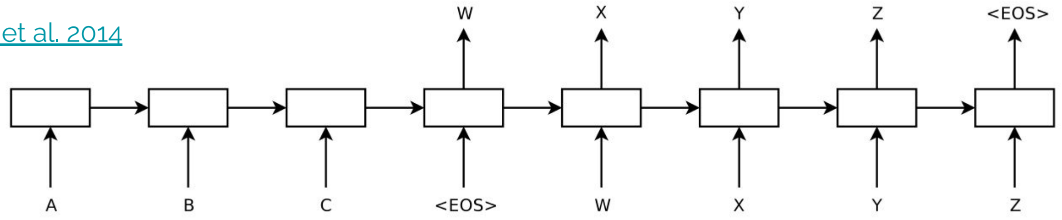
- Graph nets can be thought of in terms of RNN as well (unrolling over time)/




MLP
Convolutional Network
Recurrent Network
Graph Neural Networks and Attention

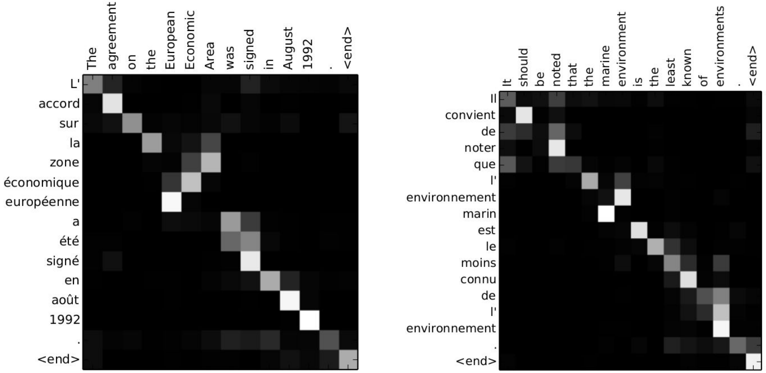
End-to-end translation systems (LSTM based):

Sutskever et al. 2014

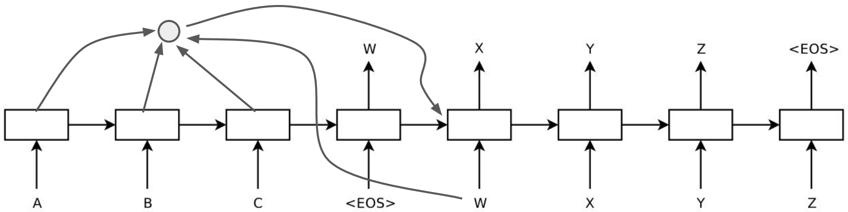


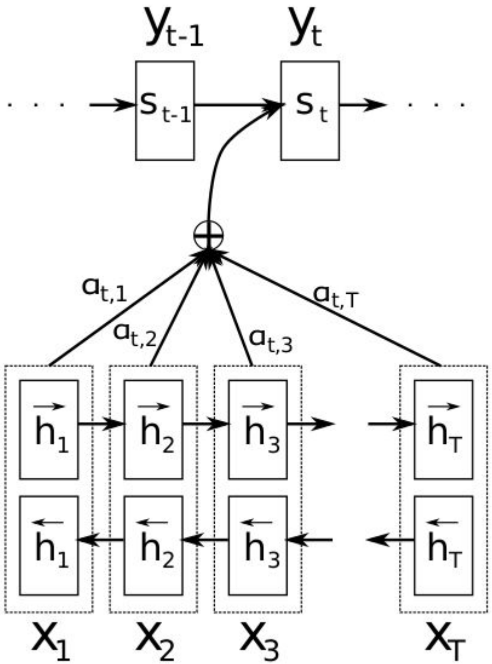
Attention mechanism

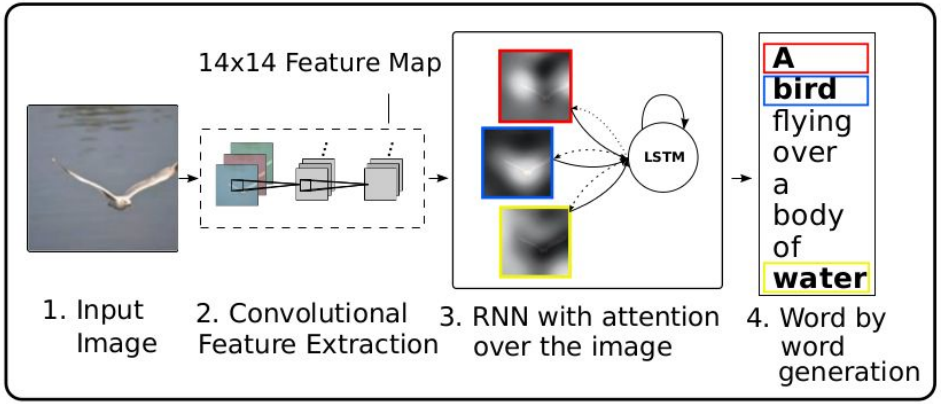
[Bahdanau et al. 2015]



(a) (b)





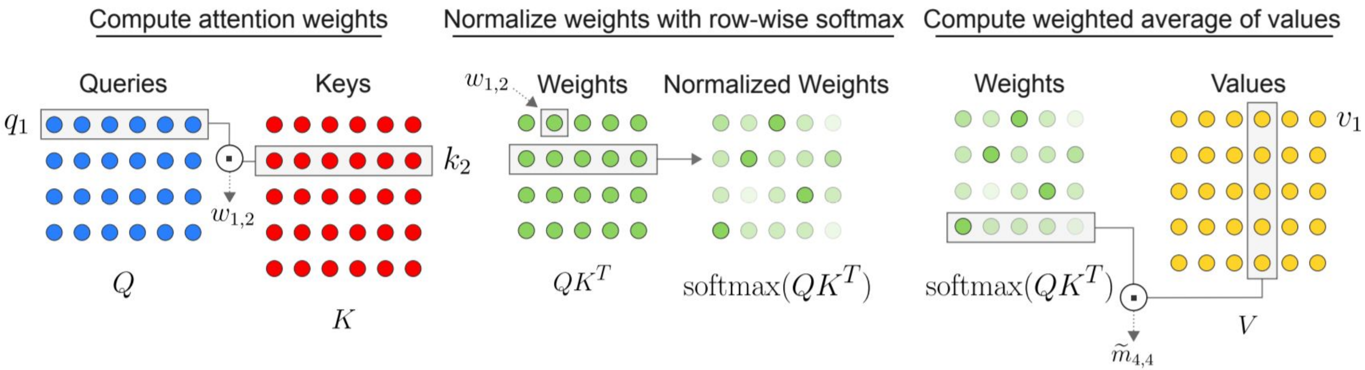


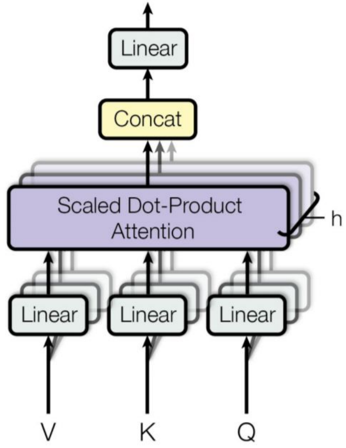
- Baidu/UCLA: Explain Images with Multimodal Recurrent Neural Networks
- Toronto: Unifying Visual-Semantic Embeddings with Multimodal Neural Language Models
- Berkeley: Long-term Recurrent Convolutional Networks for Visual Recognition and Description
- Google: Show and Tell: A Neural Image Caption Generator
- Stanford: Deep Visual-Semantic Alignments for Generating Image Description
- UML/UT: Translating Videos to Natural Language Using Deep Recurrent Neural Networks
- Microsoft/CMU: Learning a Recurrent Visual Representation for Image Caption Generation
- Microsoft: From Captions to Visual Concepts and Back

- Treat inputs as a set
- Everytime you evaluate an item, attend to all other elements

https://arxiv.org/abs/1706.03762

Self-attention (the core component of Transformer network)





https://arxiv.org/pdf/1806.01822.pdf

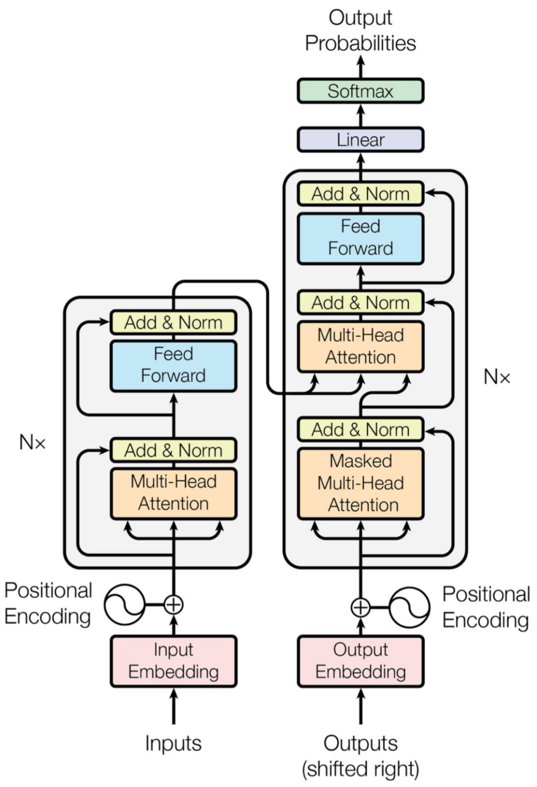
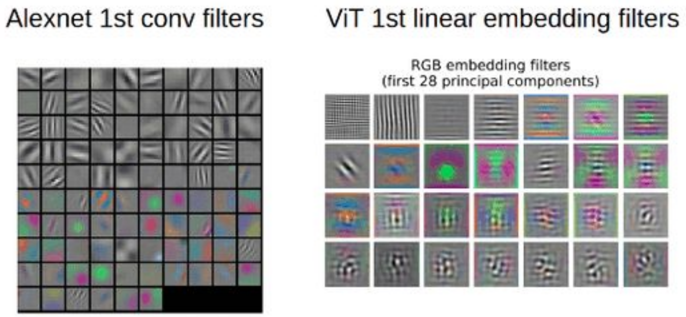
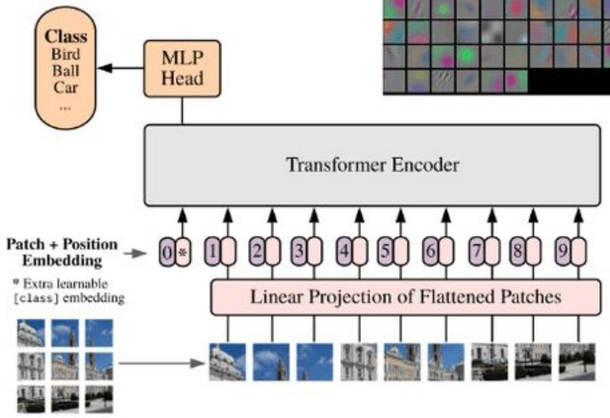
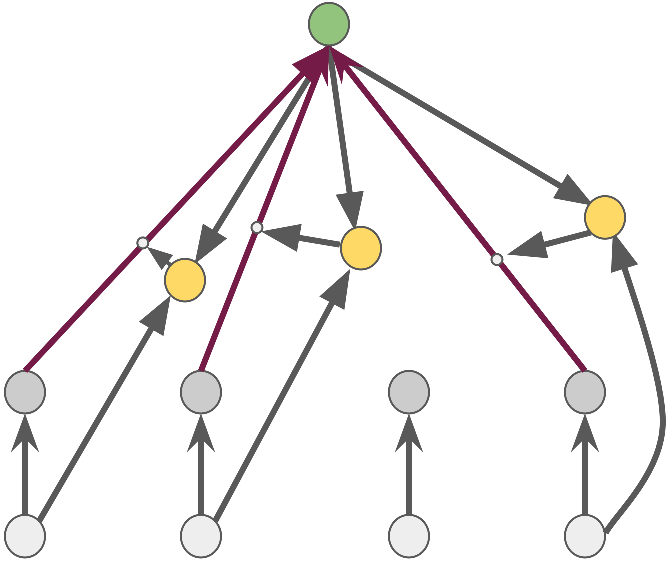


Figure 1: The Transformer - model architecture.







Graph Attention Nets (GATs)

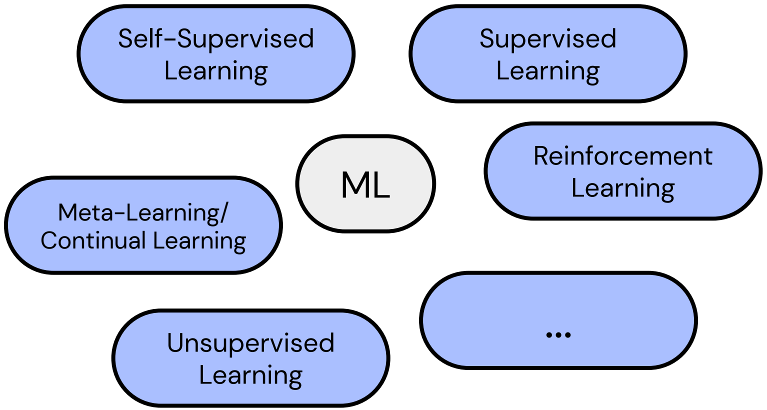
$$h_j^{(t)} = \sigma\left(\sum_i g(h_i^{(t-1)}, h_j^{(t-1)}) f(h_i^{(t-1)})\right)$$

https://arxiv.org/abs/1710.10903

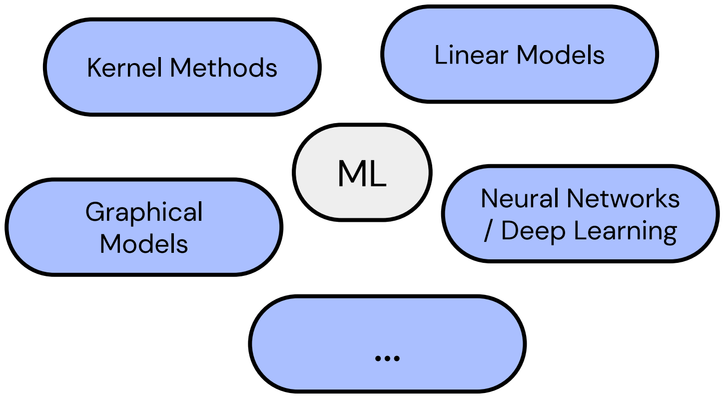
- Even if Transformers were not originally developed as a GraphNet, GraphNets are a natural formalism to describe transformers
- In this space, Transformers are close to GATs

Categorization of Machine Learning

Learning protocols /
Evaluation protocols /
Data



Learning strategy /
Structure of the solution



Splitting Machine Learning and positioning a set of approaches within the field can be messy.

Many more topics I would have liked to bring up:

- Meta / Continual / Transfer / Multi-objective Learning
- Uncertainty
- Different domains: language, vision, etc.
- Different protocols: Unsupervised / Self-supervised
- Optimization (trust region methods, natural gradient, etc.)
- Non-parametric (and interaction with parametric)
- Bayesian DL and probabilistic framework for ML
- Compression/low-precision DL, sparsity
- …

Resources

- Abundance of resources online showing how to instantiate typical architectures using pytorch / jax / tensorflow
- Some examples:
  - EEML practical sessions:
    https://github.com/eemlcommunity/PracticalSessions2021
    https://github.com/eemlcommunity/PracticalSessions2020
    https://github.com/eemlcommunity/PracticalSessions2019
  - https://github.com/deepmind/educational
  - https://roberttlange.github.io/posts/2020/03/blog-post-10/
  -

THANK YOU!