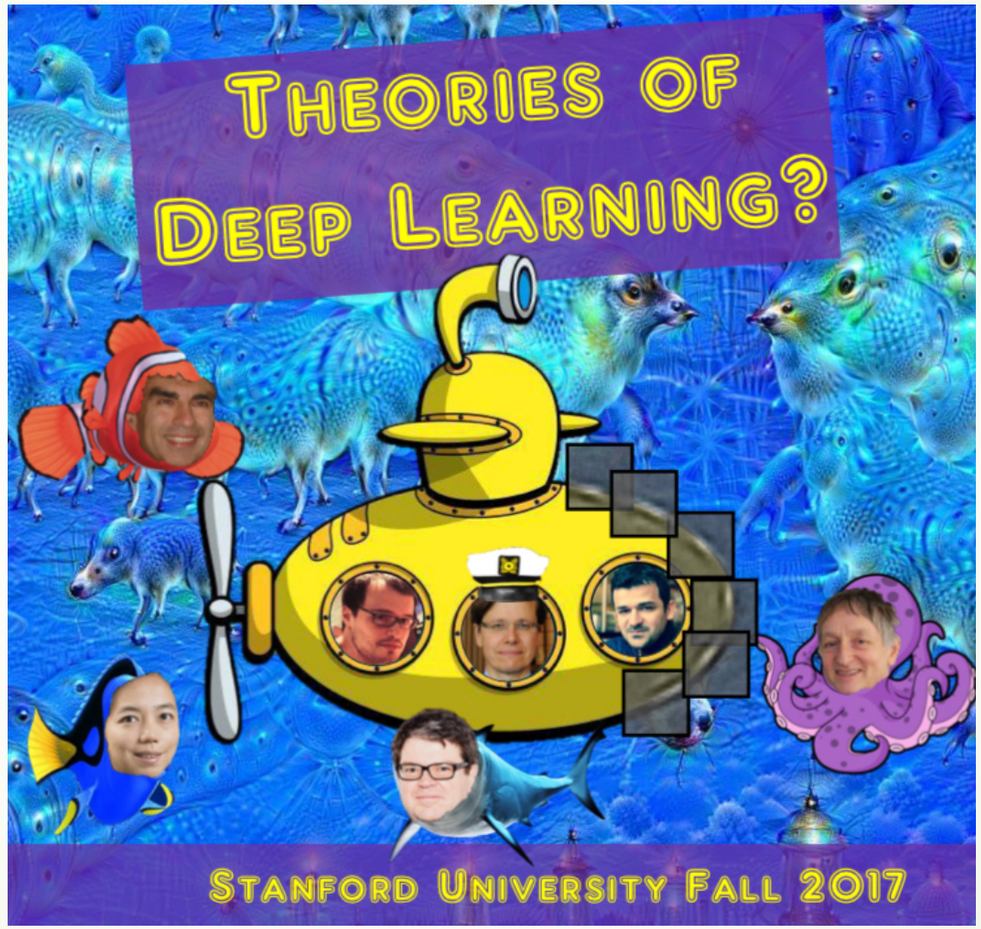
On Mathematical Theories of Deep Learning

Yuan YAO

HKUST

Acknowledgement



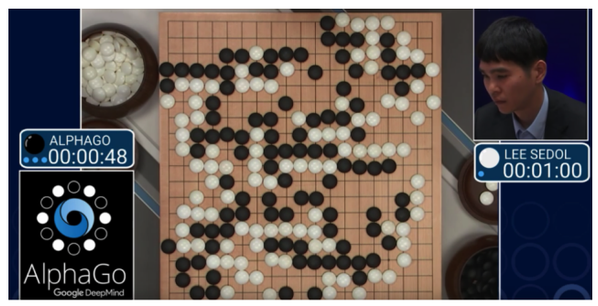
A following-up course at HKUST: https://deeplearning-math.github.io/

Outline

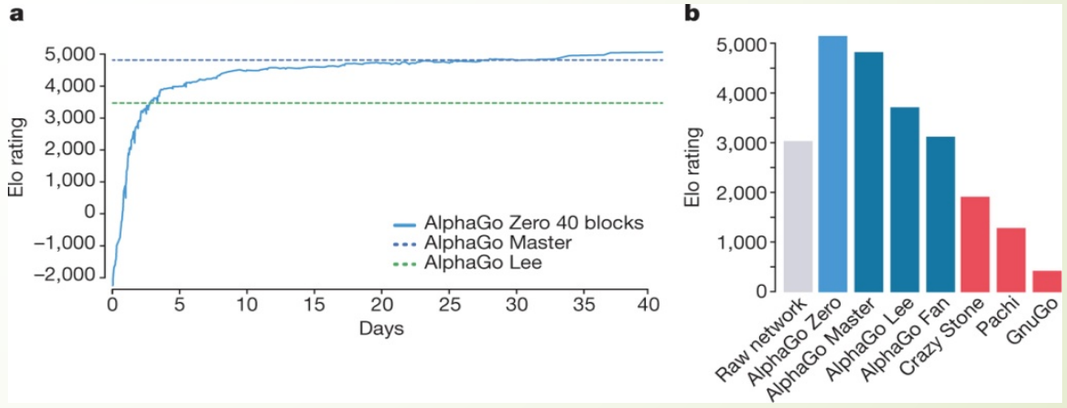
- Why mathematical theories of Deep Learning?
  - The tsunami of deep learning in recent years…
- What Theories Do We Have or Need?
  - Harmonic Analysis: what are optimal representation of functions?
  - Approximation Theory: when deep networks are better than shallow ones?
  - Optimization: what are the landscapes of risk and how to efficiently find a good optimum?
  - Statistics: how deep net models can generalize well?

Reaching Human Performance Level

Deep Blue in 1997



AlphaGo "LEE" 2016

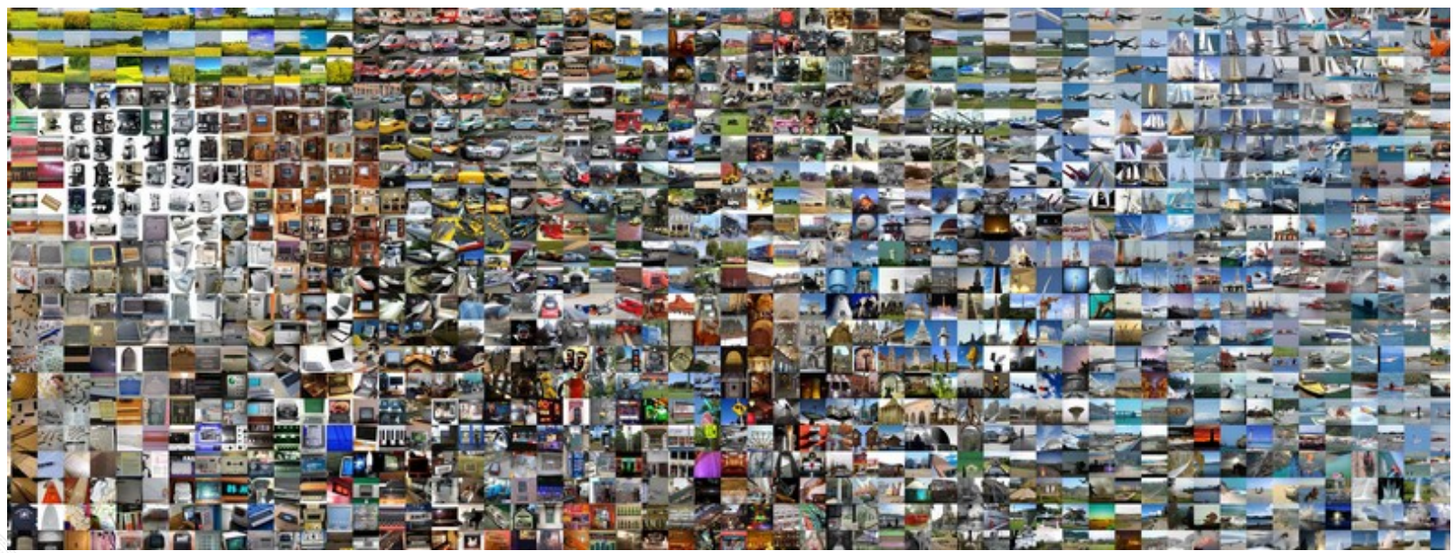


AlphaGo "ZERO" D Silver *et al. Nature* **550,** 354–359 (2017) doi:10.1038/nature24270

ImageNet Dataset

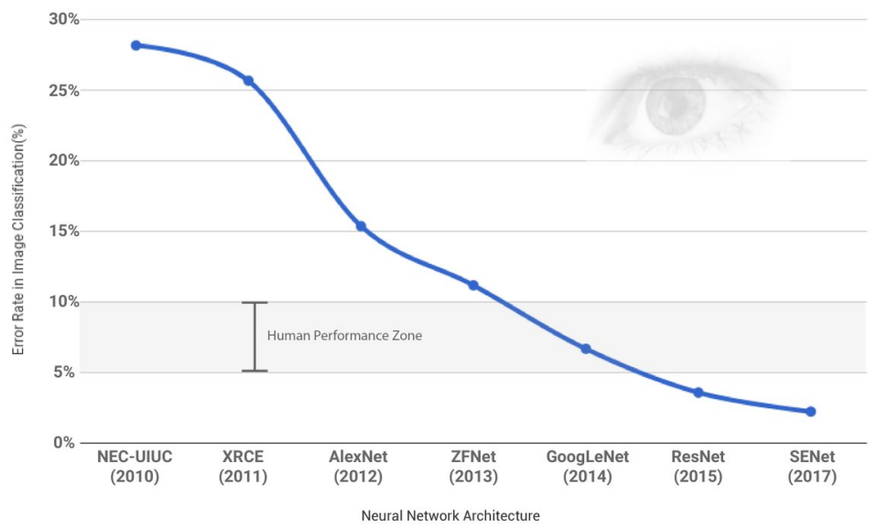
- 14,197,122 labeled images
- 21,841 classes
- Labeling required more than a year of human effort via Amazon Mechanical Turk

IMAGENET



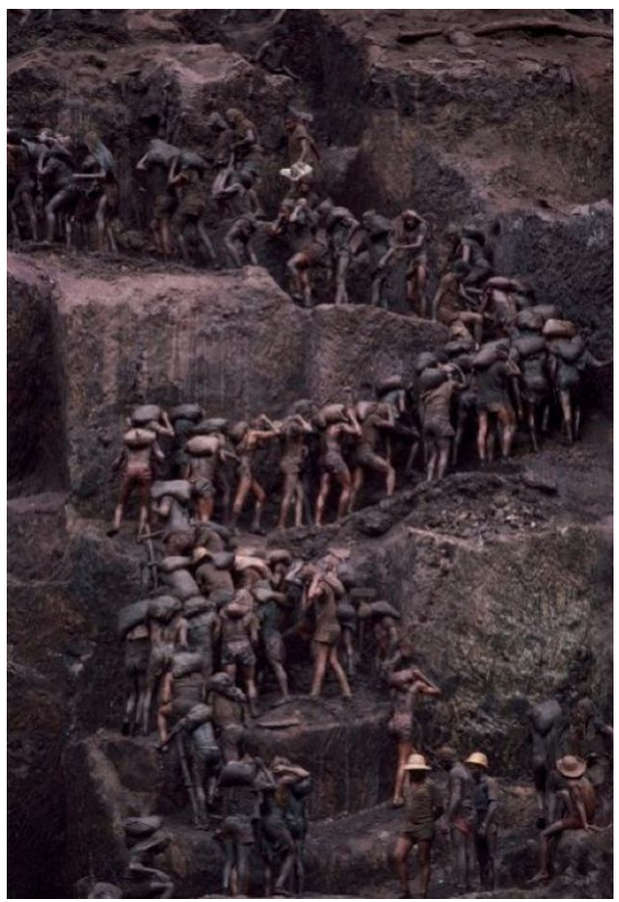
ImageNet Top 5 classification error

- ImageNet (subset):
  - 1.2 million training images
  - 100,000 test images
  - 1000 classes
- ImageNet large-scale visual recognition Challenge

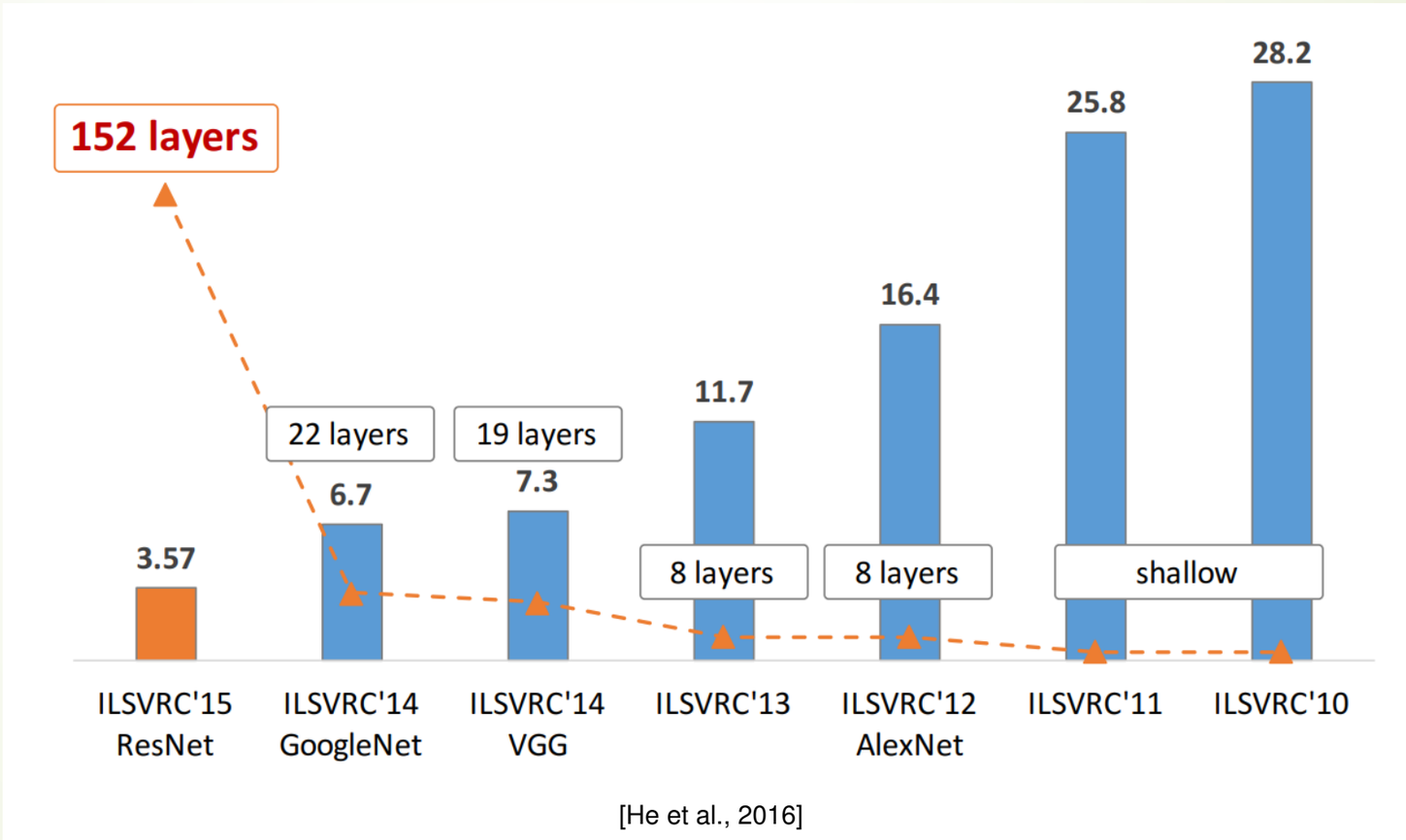


source: https://www.linkedin.com/pulse/must-read-path-breaking-papers-image-classification-muktabh-mayank

Crowdcomputing:
researchers raising the competition record

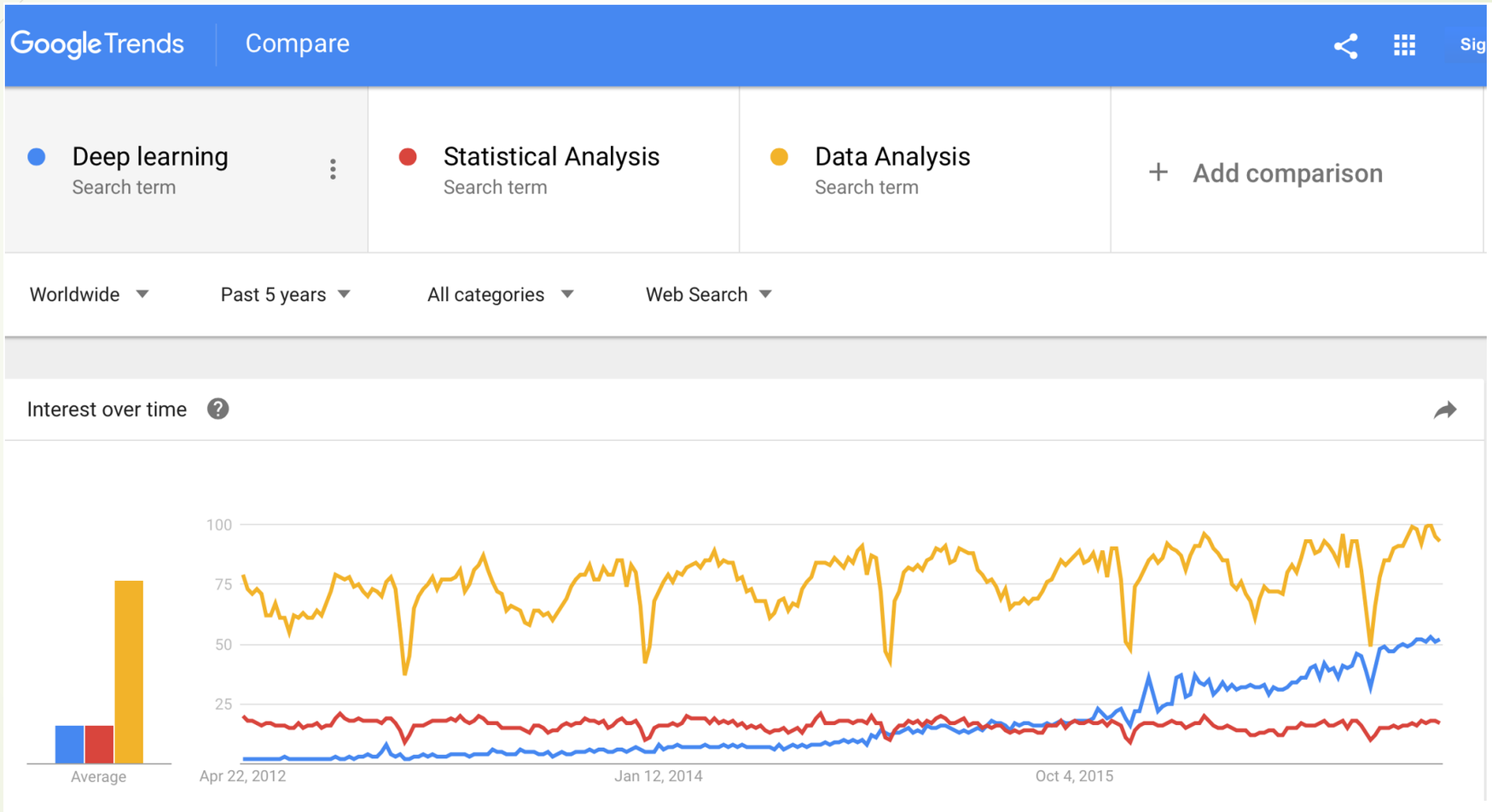


Depth as function of year



[He et al., 2016]

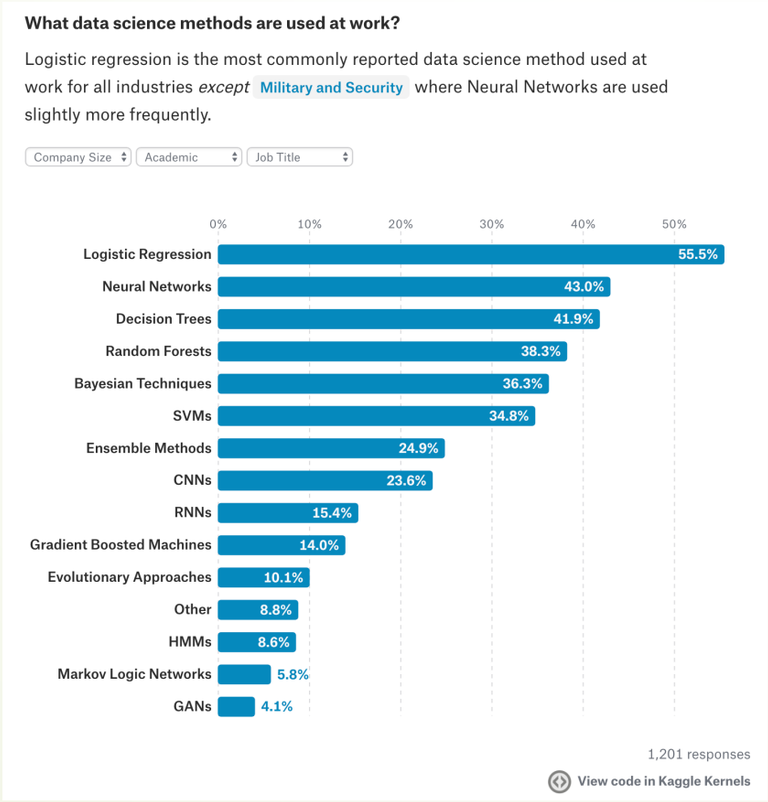
Growth of Deep Learning



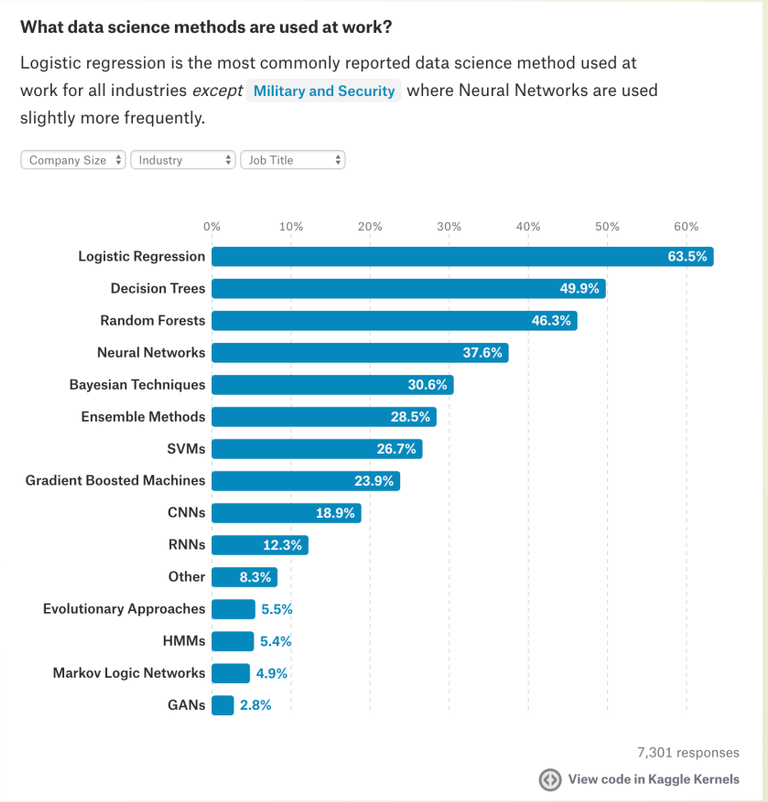
Kaggle survey: Top Data Science Methods

https://www.kaggle.com/surveys/2017

Academic



Industry



What's wrong with deep learning?

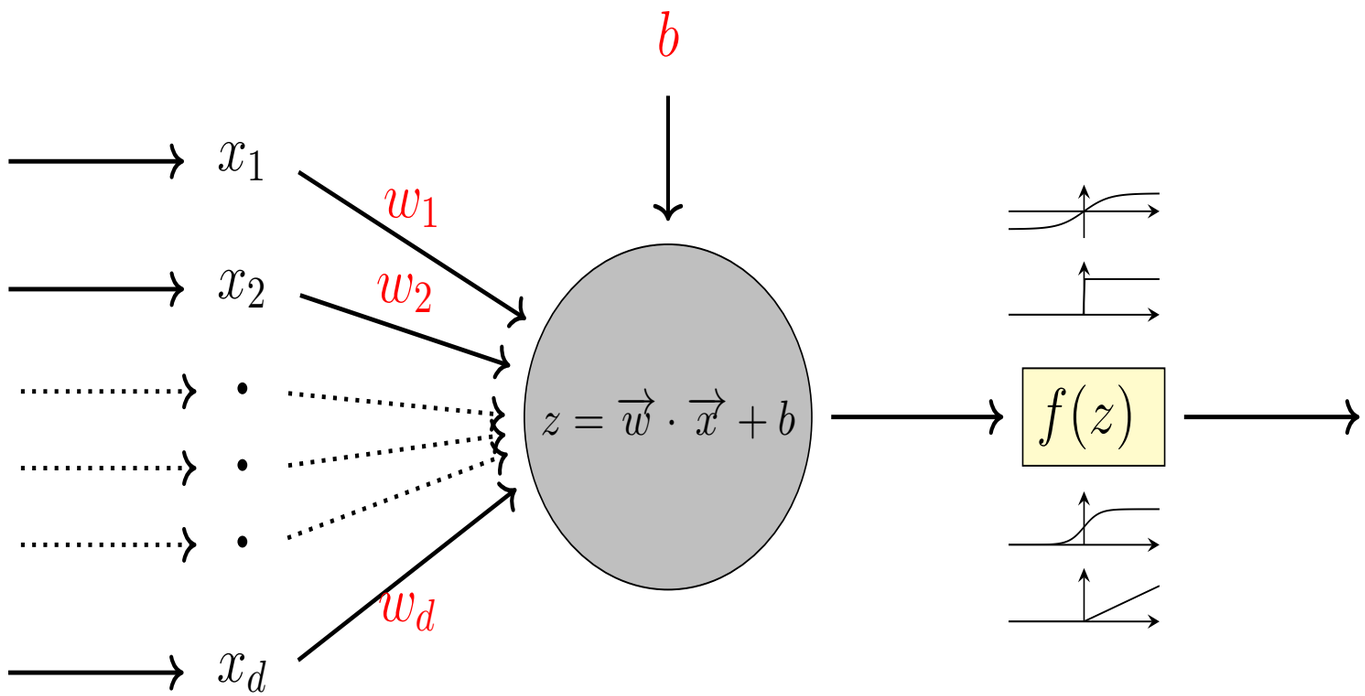
Ali Rahimi NIPS'17: Machine (deep) Learning has become **alchemy**.
https://www.youtube.com/watch?v=ORHFOnaEzPc

Yann LeCun CVPR'15, invited talk: **What's wrong with deep learning?**
One important piece: **missing some theory**!

http://techtalks.tv/talks/whats-wrong-with-deep-learning/61639/

Perceptron: single-layer

- Invented by Frank Rosenblatt (1957)



Locality or Sparsity of Computation

Minsky and Papert, 1969

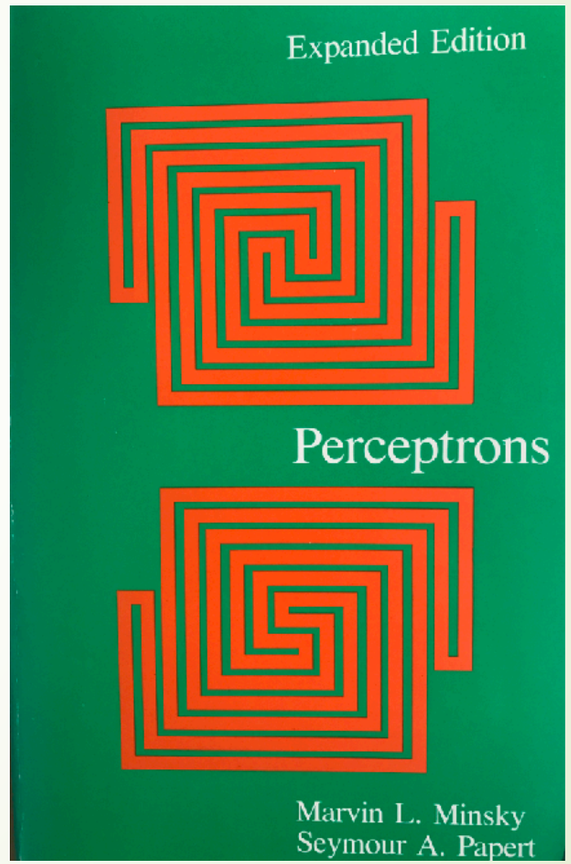
Perceptron can't do **XOR** classification

Perceptron needs infinite global information to compute **connectivity**

Locality or **Sparsity** is important:

Locality in time?

Locality in space?

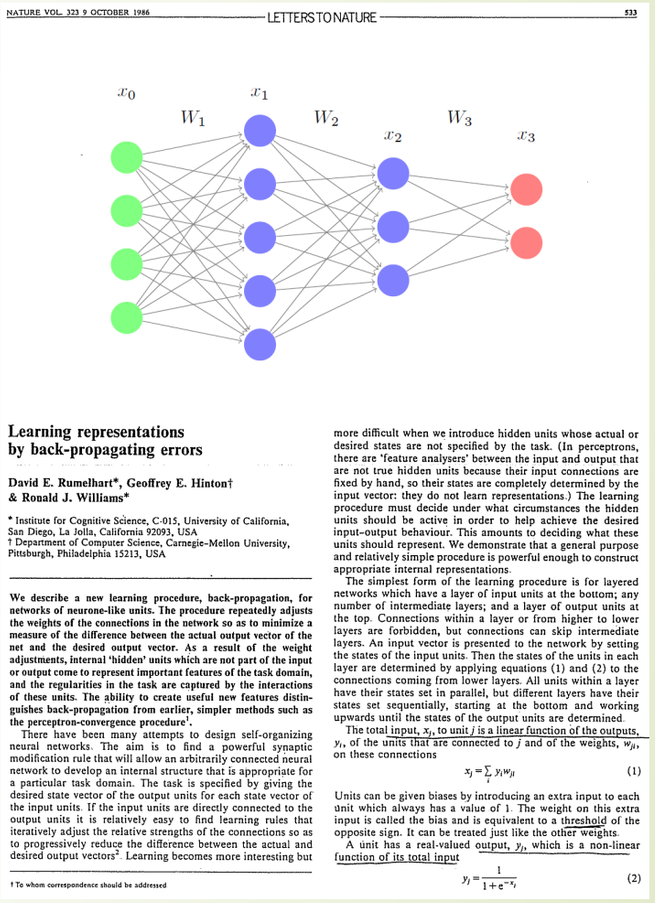


Multilayer Perceptrons (MLP) and Back-Propagation (BP) Algorithms

Rumelhart, Hinton, Williams (1986)
Learning representations by back-propagating errors, Nature, 323(9): 533-536

BP algorithms as **stochastic gradient descent** algorithms (**Robbins–Monro 1950; Kiefer-Wolfowitz 1951**) with Chain rules of Gradient maps

MLP classifies XOR, but global hurdle on topology (connectivity) computation still exists



Learning representations by back-propagating errors

David E. Rumelhart*, Geoffrey E. Hinton†
& Ronald J. Williams*

* Institute for Cognitive Science, C-015, University of California, San Diego, La Jolla, California 92093, USA
† Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Philadelphia 15213, USA

We describe a new learning procedure, back-propagation, for networks of neurone-like units. The procedure repeatedly adjusts the weights of the connections in the network so as to minimize a measure of the difference between the actual output vector of the net and the desired output vector. As a result of the weight adjustments, internal 'hidden' units which are not part of the input or output come to represent important features of the task domain, and the regularities in the task are captured by the interactions of these units. The ability to create useful new features distinguishes back-propagation from earlier, simpler methods such as the perceptron-convergence procedure[1].

There have been many attempts to design self-organizing neural networks. The aim is to find a powerful synaptic modification rule that will allow an arbitrarily connected neural network to develop an internal structure that is appropriate for a particular task domain. The task is specified by giving the desired state vector of the output units for each state vector of the input units. If the input units are directly connected to the output units it is relatively easy to find learning rules that iteratively adjust the relative strengths of the connections so as to progressively reduce the difference between the actual and desired output vectors[2]. Learning becomes more interesting but more difficult when we introduce hidden units whose actual or desired states are not specified by the task. (In perceptrons, there are 'feature analysers' between the input and output that are not true hidden units because their input connections are fixed by hand, so their states are completely determined by the input vector: they do not learn representations.) The learning procedure must decide under what circumstances the hidden units should be active in order to help achieve the desired input–output behaviour. This amounts to deciding what these units should represent. We demonstrate that a general purpose and relatively simple procedure is powerful enough to construct appropriate internal representations.

The simplest form of the learning procedure is for layered networks which have a layer of input units at the bottom; any number of intermediate layers; and a layer of output units at the top. Connections within a layer or from higher to lower layers are forbidden, but connections can skip intermediate layers. An input vector is presented to the network by setting the states of the input units. Then the states of the units in each layer are determined by applying equations (1) and (2) to the connections coming from lower layers. All units within a layer have their states set in parallel, but different layers have their states set sequentially, starting at the bottom and working upwards until the states of the output units are determined.

The total input, x_j, to unit j is a linear function of the outputs, y_i, of the units that are connected to j and of the weights, w_{ji}, on these connections

$$x_j = \sum_i y_i w_{ji} \quad (1)$$

Units can be given biases by introducing an extra input to each unit which always has a value of 1. The weight on this extra input is called the bias and is equivalent to a threshold of the opposite sign. It can be treated just like the other weights.

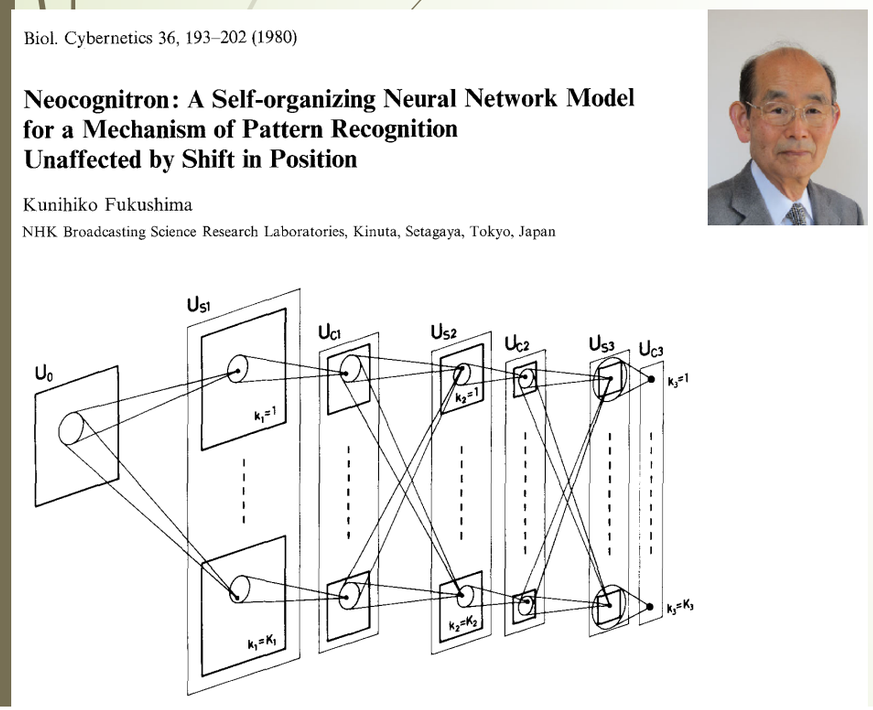
A unit has a real-valued output, y_j, which is a non-linear function of its total input

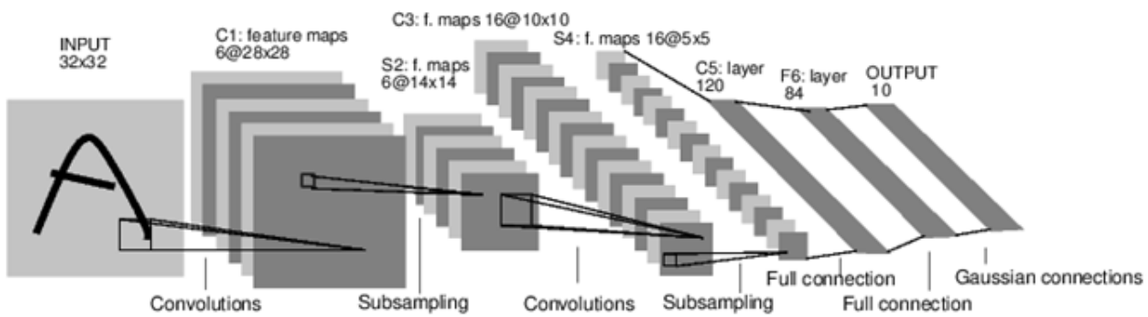
$$y_j = \frac{1}{1+e^{-x_j}} \quad (2)$$

† To whom correspondence should be addressed

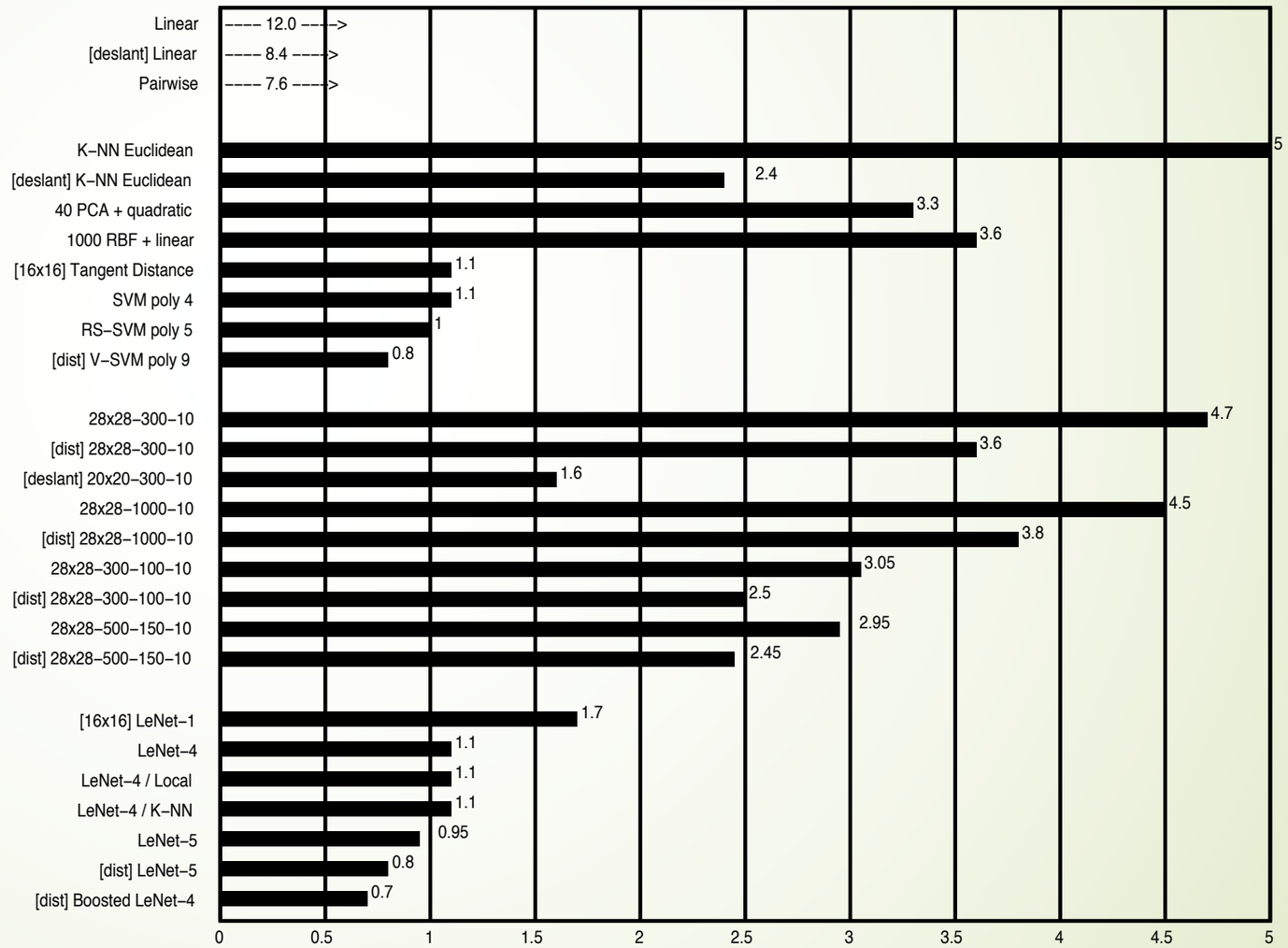
Convolutional Neural Networks: shift invariances and locality

- Can be traced to *Neocognitron* of Kunihiko Fukushima (1979)
- Yann LeCun combined convolutional neural networks with back propagation (1989)
- Imposes **shift invariance** and **locality** on the weights
- Forward pass remains similar
- Backpropagation slightly changes – need to sum over the gradients from all spatial positions





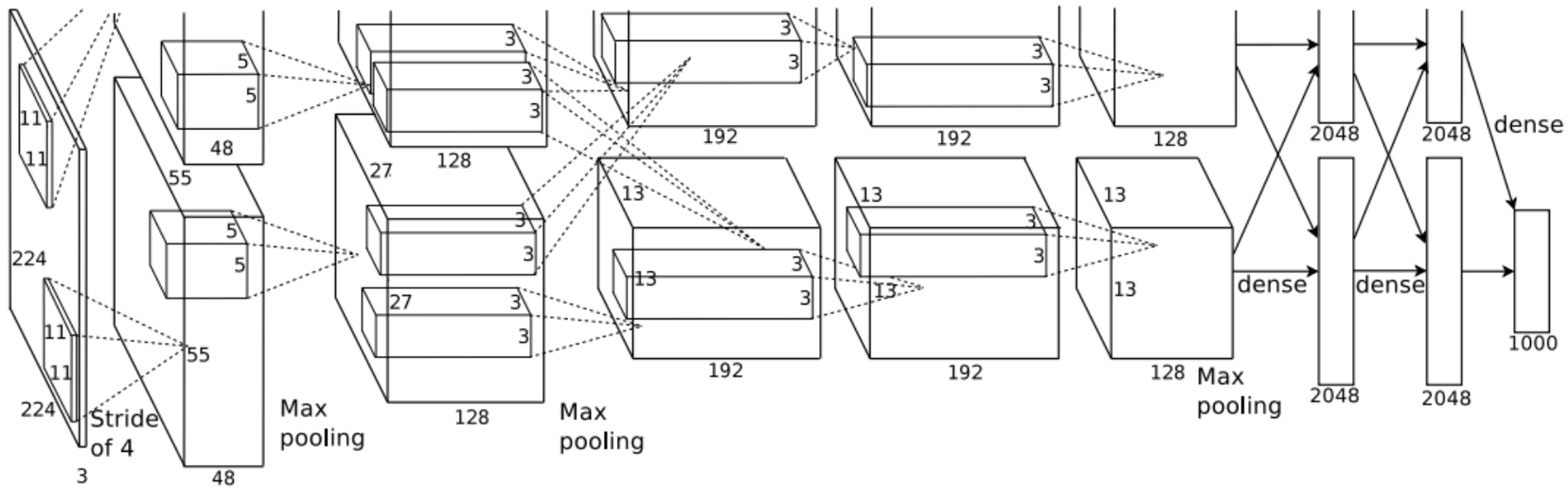
MNIST Dataset Test Error
LeCun et al. 1998



Simple SVM performs as well as Multilayer Convolutional Neural Networks which need careful tuning (LeNets)

AlexNet (2012)

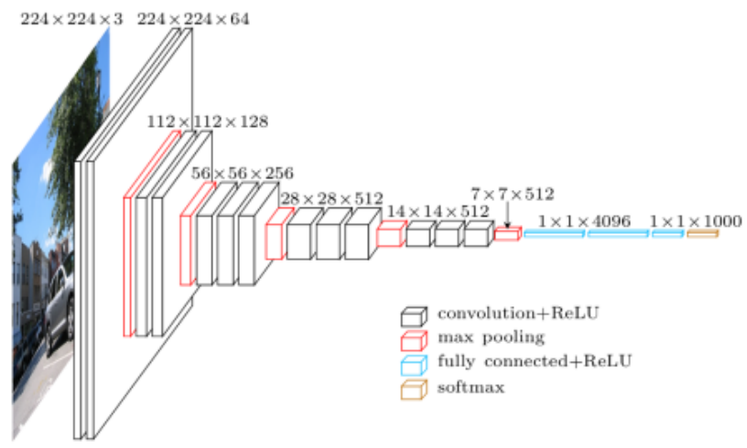
- 8 layers: first 5 convolutional, rest fully connected
- ReLU nonlinearity
- Local response normalization
- Max-pooling
- Dropout



Source: [Krizhevsky et al., 2012]

VGG (2014) [Simonyan-Zisserman'14]

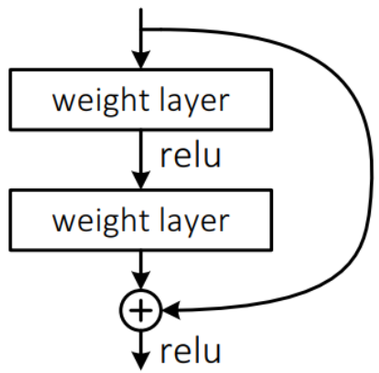
- Deeper than AlexNet: 11-19 layers versus 8
- No local response normalization
- Number of filters multiplied by two every few layers
- Spatial extent of filters 3×3 in all layers
- Instead of 7×7 filters, use three layers of 3×3 filters
  - Gain intermediate nonlinearity
  - Impose a regularization on the 7×7 filters

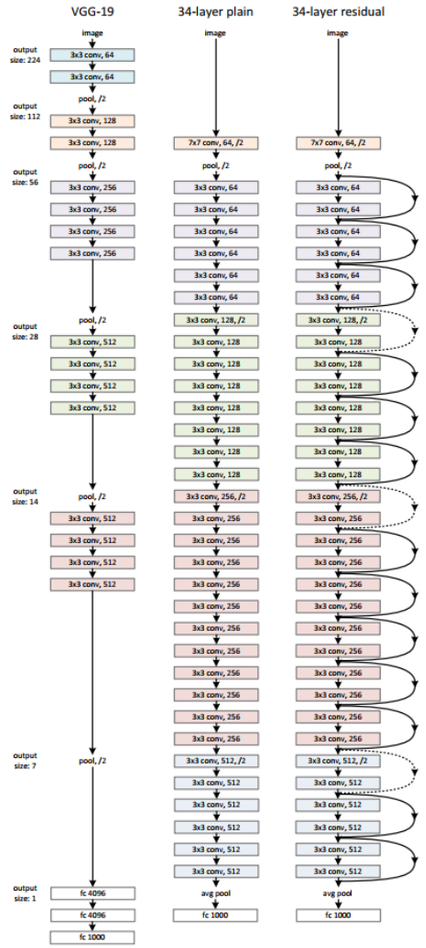


Source: https://blog.heuritech.com/2016/02/29/

ResNet (2015) [HGRS-15]

- Solves problem by adding skip connections
- Very deep: 152 layers
- No dropout
- Stride
- Batch normalization





Source: Deep Residual Learning for Image Recognition

Visualizing Deep Neural Networks

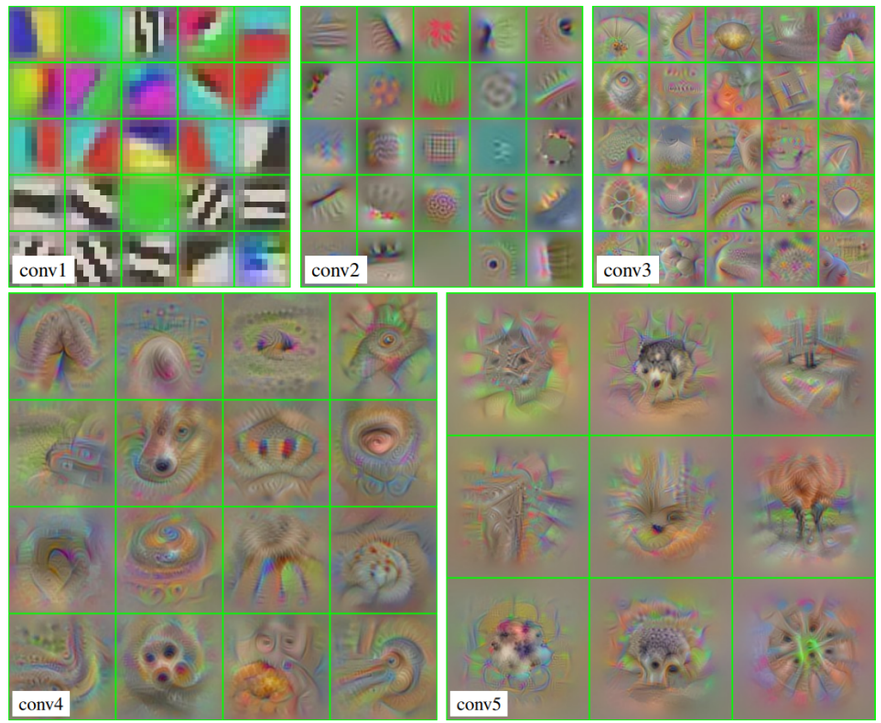
- Filters in first layer of CNN are easy to visualize, while deeper ones are harder
- *Activation maximization* seeks input image maximizing output of the i-th neuron in the network
- Objective

$$x^* = \arg\min_x \mathcal{R}(x) - \langle \Phi(x), e_i \rangle$$

- e_i is indicator vector
- $\mathcal{R}(x)$ is simple natural image prior

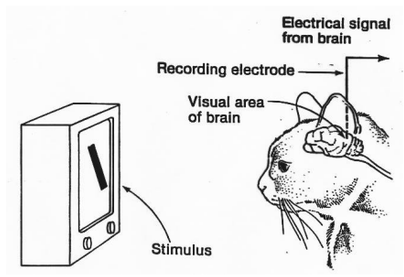
Visualizing VGG

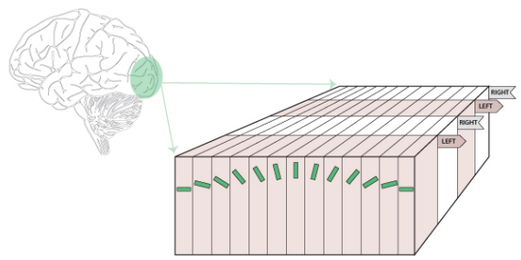
- Gabor-like images in first layer
- More sophisticated structures in the rest

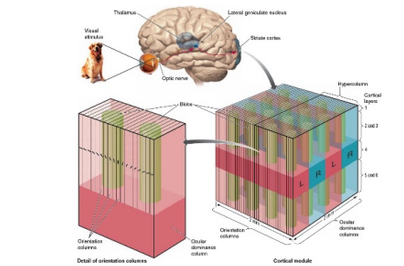


[Mahendran and Vedaldi, 2016]

Visual Neuroscience: Hubel/Wiesel, …

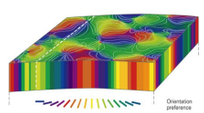




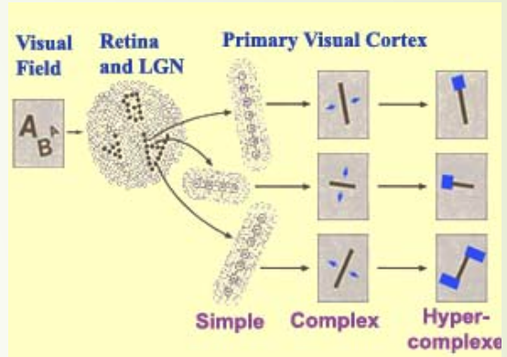


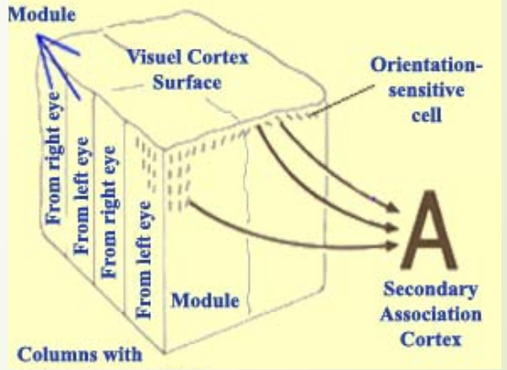
V1 orientation maps

- Continuous maps of orientation preference: "pinwheels"
- Consistent preference though the depth of cortex: columnar architecture



Kandel et. al., *Princ. of Neural Sci.*





Olshausen and Field 1996

Experimental Neuroscience uncovered the

- ... neural architecture of Retina/LGN/V1/V2/V3/ etc
- ... existence of neurons with weights and activation functions (simple cells)
- ... pooling neurons (complex cells)

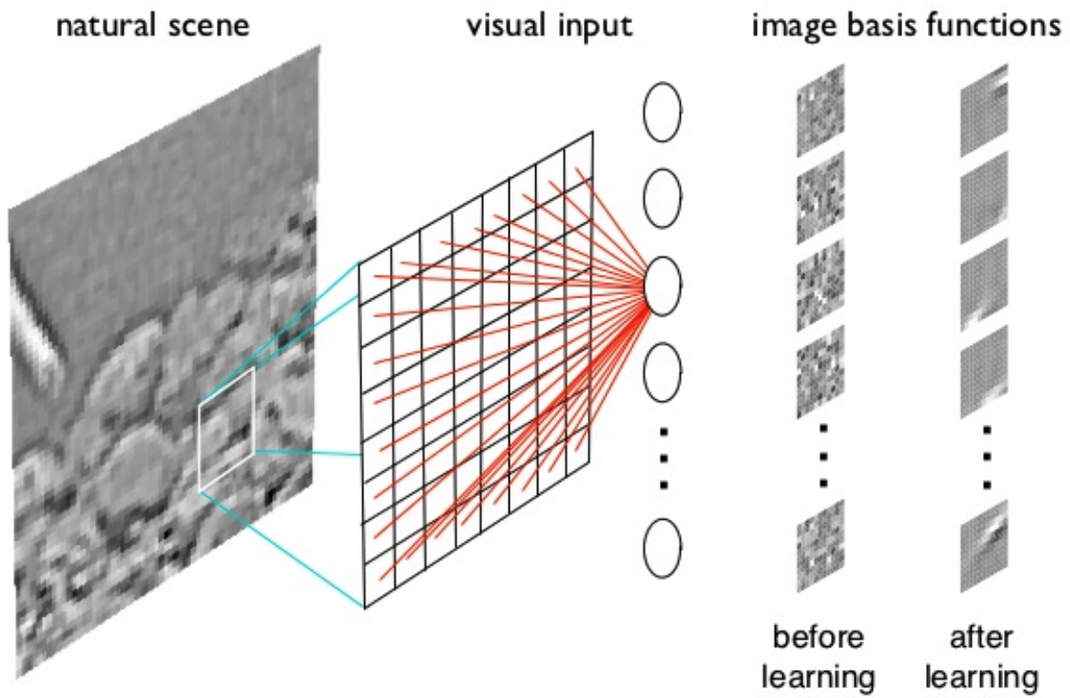
All these features are somehow present in today's sucessful Deep Learning systems

Neuroscience	Deep Network
Simple cells	First layer
Complex celle	Pooling Layer
Grandmother cells	Last layer

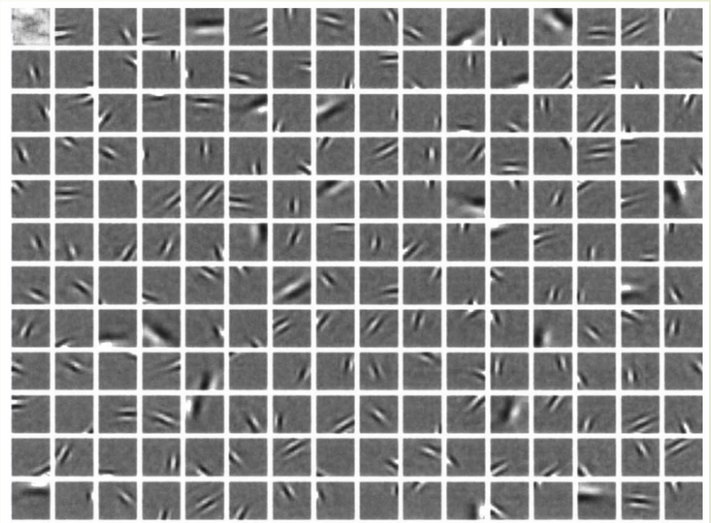
Theorists Olshausen and Field (Nature, 1996) demonstrated that receptive fields learned from image patches

First layers learned …

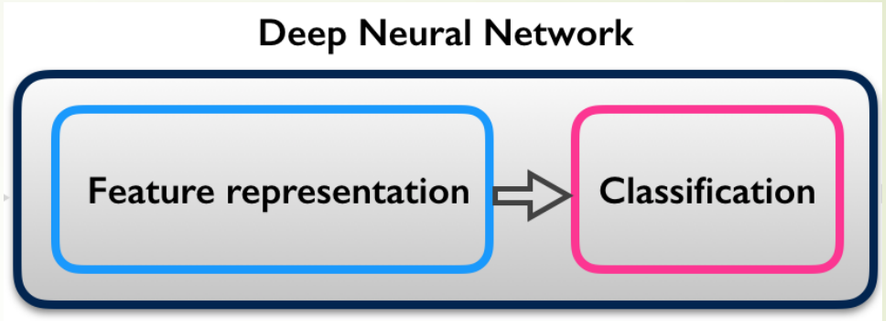
Efficient coding of natural images: Olshausen and Field, 1996



Network weights are adapted to maximize coding efficiency:
minimizes redundancy and maximizes the independence of the outputs



Transfer Learning



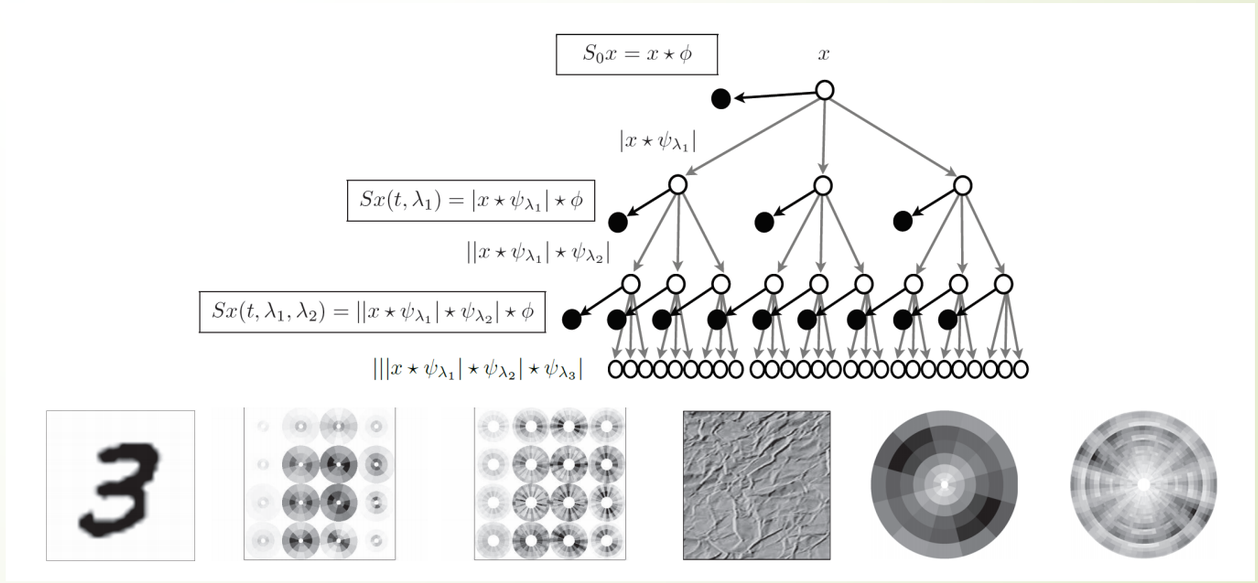
- Filters learned in first layers of a network are transferable from one task to another
- When solving another problem, no need to retrain the lower layers, just fine tune upper ones
- Is this simply due to the large amount of images in ImageNet?
- Does solving many classification problems simultaneously result in features that are more easily transferable?
- Does this imply filters can be learned in unsupervised manner?
- Can we characterize filters mathematically?

Some Open Theoretical Problems

- *Harmonic Analysis:* What are the optimal (transferrable) representations of functions as input signals (sounds, images, …)?
- *Approximation Theory:* When and why are deep networks better than shallow networks?
- *Optimization:* What is the landscape of the empirical risk and how to minimize it efficiently?
- *Statistics:* How can deep learning generalize well without overfitting the noise?

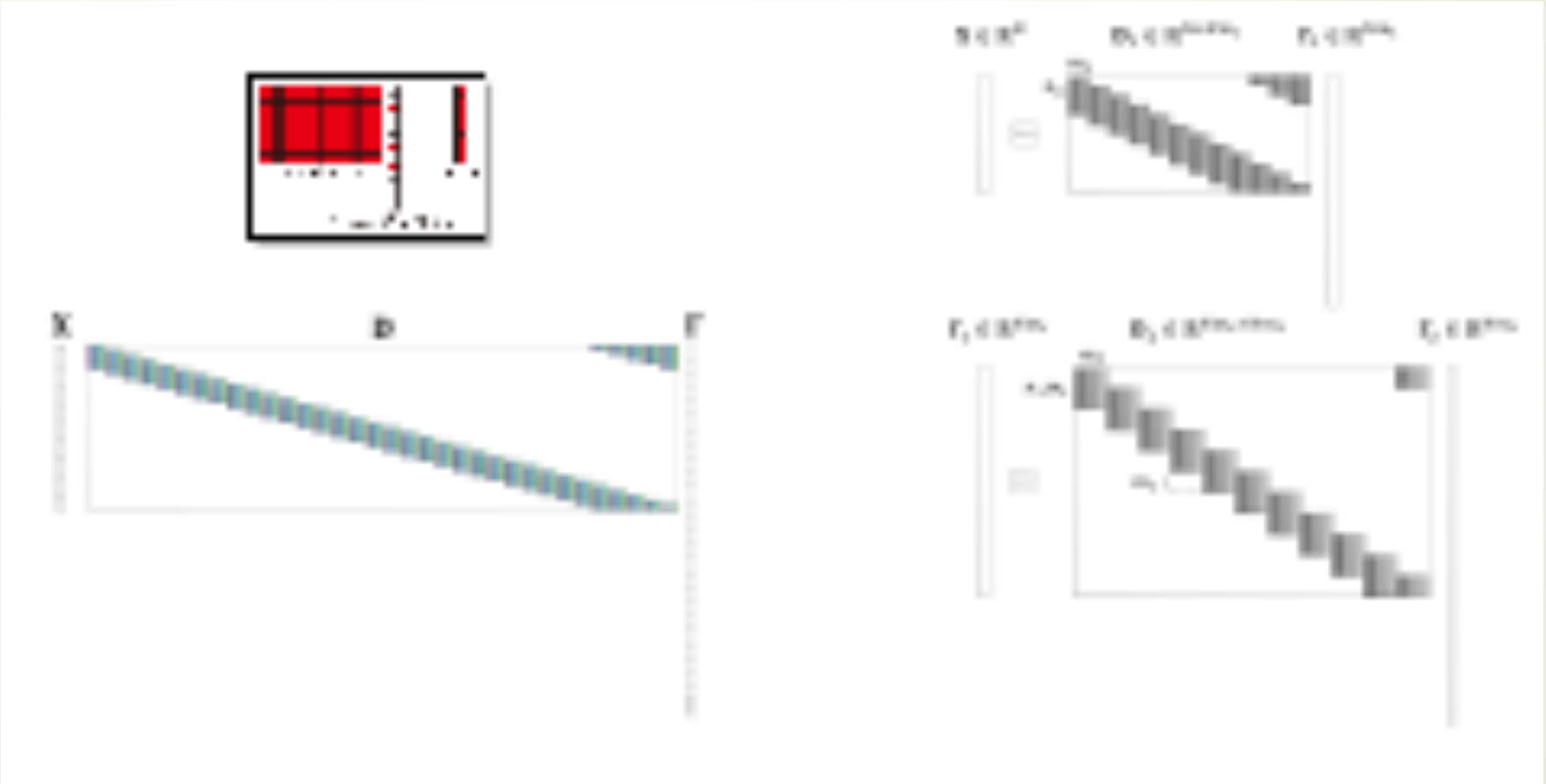
Harmonic Analysis

- Harmonic analysis: optimal representation of input signals
- Wavelets are optimal sparse representations for certain class of images
- **Stephane Mallat:** Deep Scattering Transform -- translational, rotational, scaling, and small deformation invariances, the deeper is the network, the larger are the invariances
- **Mathew Hirn** @IAS-HKUST talked about scattering net for energy functions on 3-D densities (images)



Scattering Transform: Mallat'12

Sparse Representations



Compressed Sensing

Given a signal, we would like to find its sparse representation

Convexify

$$\min_{\mathbf{\Gamma}} \|\mathbf{\Gamma}\|_0 \quad \text{s.t.} \quad \mathbf{X} = \mathbf{D}\mathbf{\Gamma}$$

$$\min_{\mathbf{\Gamma}} \|\mathbf{\Gamma}\|_1 \quad \text{s.t.} \quad \mathbf{X} = \mathbf{D}\mathbf{\Gamma}$$

Crude approximation

$$\mathcal{S}_\beta\{\mathbf{D}^T\mathbf{X}\}$$

Compressed Sensing

Given a signal, we would like to find its sparse representation

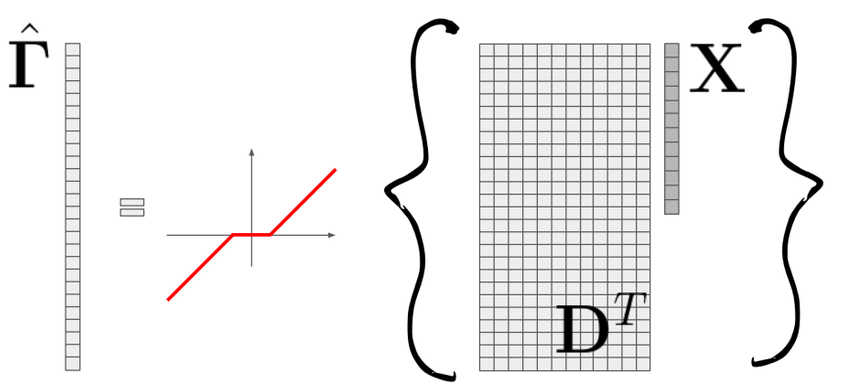
Convexify

Crude approximation

$$\min_{\mathbf{\Gamma}} \|\mathbf{\Gamma}\|_0 \quad \text{s.t.} \quad \mathbf{X} = \mathbf{D}\mathbf{\Gamma}$$

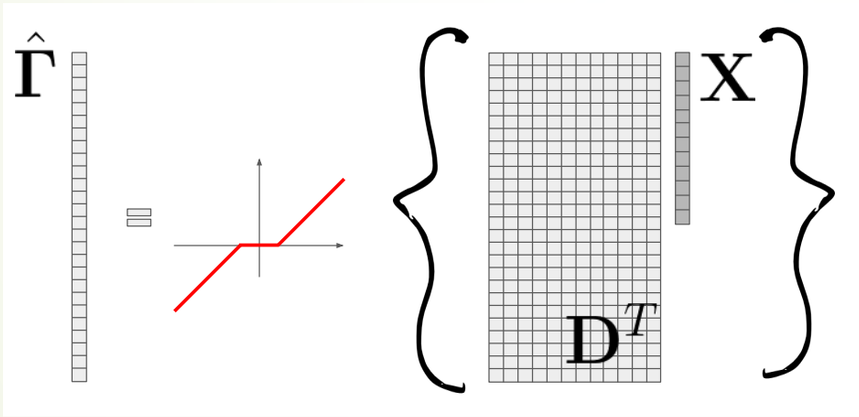
$$\min_{\mathbf{\Gamma}} \|\mathbf{\Gamma}\|_1 \quad \text{s.t.} \quad \mathbf{X} = \mathbf{D}\mathbf{\Gamma}$$

$$\mathcal{S}_\beta\{\mathbf{D}^T\mathbf{X}\}$$

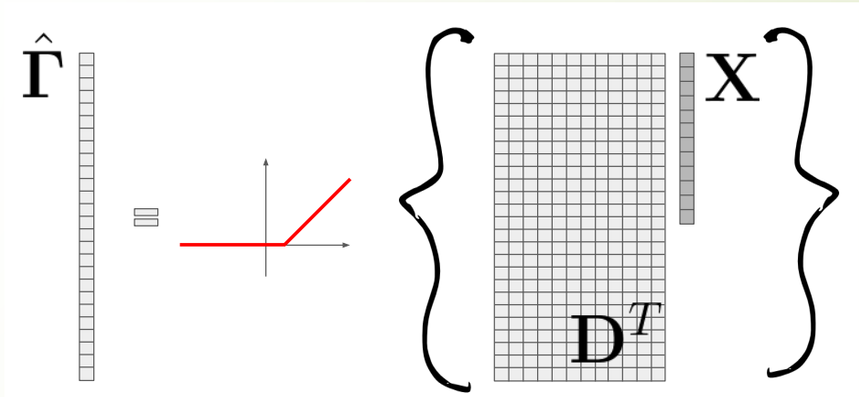


From Soft Thresholding to ReLU

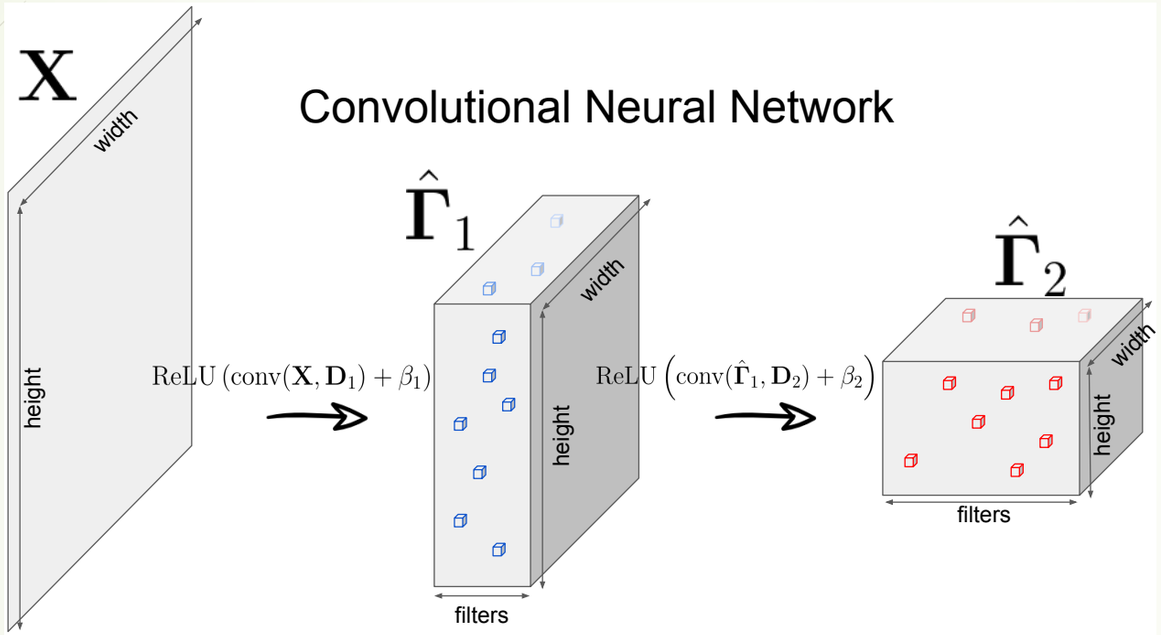
Soft Thresholding



ReLU: Soft Nonnegative Threshoding

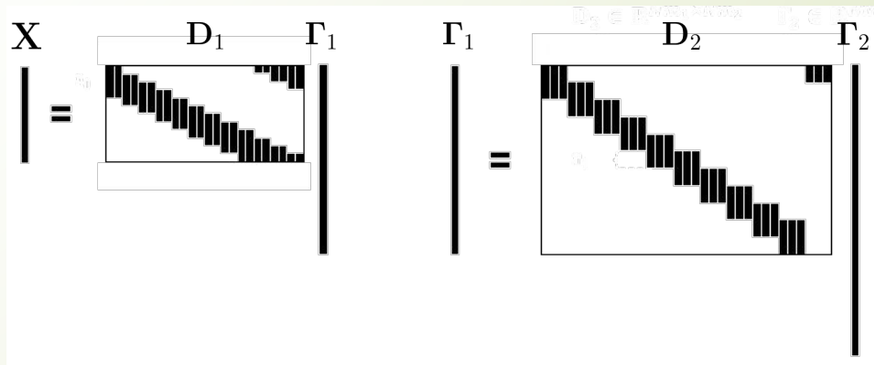


Convolutional Neural Network



Can we simultaneously learn dictionaries $\mathbf{D}$s and $\mathbf{\Gamma}$s?

Incoherence…
Papyan, Sulam, and Elad 2016



Approximation Theory

- Class prediction rule can be viewed as function $f(x)$ of high-dimensional argument
- *Curse of Dimensionality*
  - Traditional theoretical obstacle to high-dimensional approximation
  - "*Functions of high dimensional x can wiggle in too many dimensions to be learned from finite datasets*"

Approximation Theory

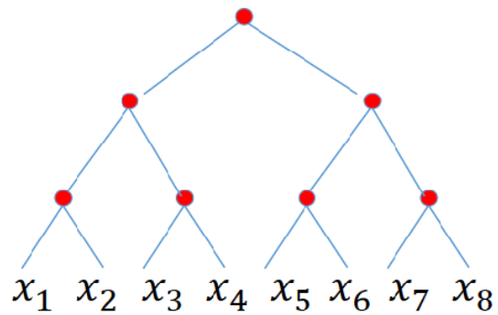
- Ridge Functions $\rho(u'x)$ mathematically same as deep learning first layer outputs.
- Sums of Ridge Functions mathematically same as input to second layer.
- Approximation by Sums of Ridge Functions $f \approx \sum_i \rho_i(u_i'x)$ studied for decades
- Theorists (1990's-Today): certain functions $f(x)$ approximated by ridge sums with no curse of dimensionalty

(Sparse) Compositional Functions

- Compositional functions $f(x) = h(g_1(x_{i_{1,1}}, ..., x_{i_{1,k}}), g_2(x_{i_{2,1}}, \ldots, x_{i_{2,k}}), \ldots, g_\ell(x_{i_{\ell,1}}, \ldots, x_{i_{\ell,k}}))$ are functions of small number of functions; $\ell, k \ll d$.
- VGG Nets are deep compositions
- Approximation by Compositional Functions studied for decades
- Theorists (1990's-Today): certain functions $f(x)$ avoid curse of dimensionalty using multilayer compositions
- T. Poggio (MIT) and Hrushikesh Mhaskar (Caltech) have several papers analyzing deepnets as deep compositions.

Mhaskar-Poggio-Liao'16

$$f(x_1, x_2, ..., x_8) = g_3(g_{21}(g_{11}(x_1, x_2), g_{12}(x_3, x_4))g_{22}(g_{11}(x_5, x_6), g_{12}(x_7, x_8)))$$



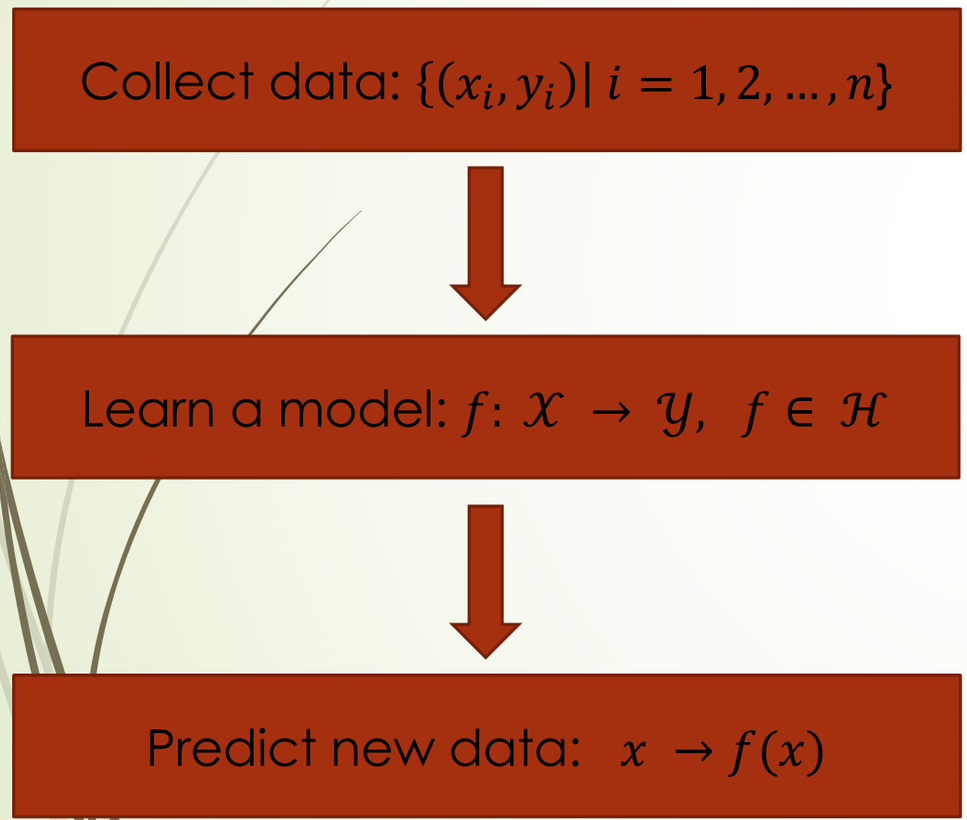
Theorem (informal statement)

Suppose that a function of d variables is hierarchically, locally, compositional . Both shallow and deep network can approximate f equally well. The number of parameters of the shallow network depends exponentially on d as $O(\varepsilon^{-d})$ with the dimension whereas for the deep network dance is $O(d\varepsilon^{-2})$

IAS-HKUST workshop talks

- *9 Jan 2018, Tuesday*:
  - **Ding-Xuan ZHOU** *Approximation Analysis of Distributed Learning and Deep CNNs*
- *10 Jan 2018, Wednesday:*
  - **Philipp Grohs** *Approximation Results for Deep Neural Networks*
- *11 Jan 2018, Thursday*:
  - **Gitta Kutyniok** *Optimal Approximation with Sparsely Connected Deep Neural Networks*
  - **Philipp Petersen** *Optimal Approximation of Classifier Functions by Deep ReLU Networks*

Generalization of Supervised Learning



All $(x, y) \sim \mathcal{D}$, where $\mathcal{D}$ is unknown

A common approach to learn:

ERM (Empirical Risk Minimization)

$$\min R_n(w) := \frac{1}{n} \sum_i l(w; x_i, y_i)$$

w: model parameters
$l(w; x, y)$: loss function w.r.t. data

Population Risk: $R(w) := \mathrm{E}[l(w; x, y)]$

Generalization Error

- We consider the standard ML setup:

$$\hat{E}(\Theta) = \mathbb{E}_{(X,Y)\sim\hat{P}}\ell(\Phi(X;\Theta),Y) + \mathcal{R}(\Theta)$$
$$E(\Theta) = \mathbb{E}_{(X,Y)\sim P}\ \ell(\Phi(X;\Theta),Y)\ .$$

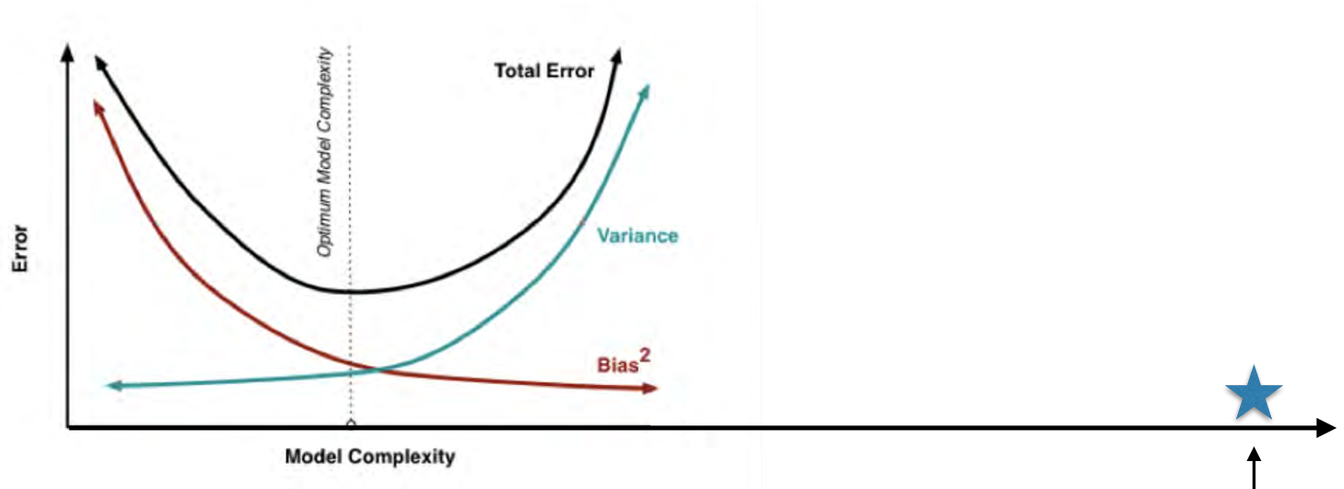
$$\hat{P} = \frac{1}{n}\sum_{i\leq}\delta_{(xi,y_i)}$$
$$\ell(z)\ \text{convex}$$
$$\mathcal{R}(\Theta)\text{: regularization}$$

- Population loss decomposition (*aka* "fundamental theorem of ML"):

$$E(\Theta^*) = \underbrace{\hat{E}(\Theta^*)}_{\text{training error}} + \underbrace{E(\Theta^*) - \hat{E}(\Theta^*)}_{\text{generalization gap}}\ .$$

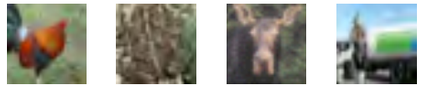
- Long history of techniques to provably control generalization error via appropriate regularization.
- Generalization error and optimization are entangled [Bottou & Bousquet]

Bias-Variance Tradeoff?



Models where p>20n are common

Why big models generalize well?



CIFAR10

n=50,000
d=3,072
k=10

What happens when I turn off the regularizers?

Model	parameters	p/n	Train ***loss***	Test ***error***
CudaConvNet	145,578	2.9	0	23%
CudaConvNet (with regularization)	145,578	2.9	0.34	18%
MicroInception	1,649,402	33	0	14%
ResNet	2,401,440	48	0	13%

Ben Recht FoCM 2017

How to control generalization error?

- However, when $\Phi(X;\Theta)$ is a large, deep network, current best mechanism to control generalization gap has two key ingredients:
  - Stochastic Optimization
    - "During training, it adds the sampling noise that corresponds to empirical-population mismatch" [Léon Bottou].
  - Make the model *as large as possible*.
    - see e.g. "Understanding Deep Learning Requires Rethinking Generalization", [Ch. Zhang *et al*, ICLR'17].

Traditional Learning Theory

Common form of generalization bound (in expectation or high probability)

$$R(w) \leq R_n(w) + \sqrt{\frac{Capacity\ Measure}{n}}$$

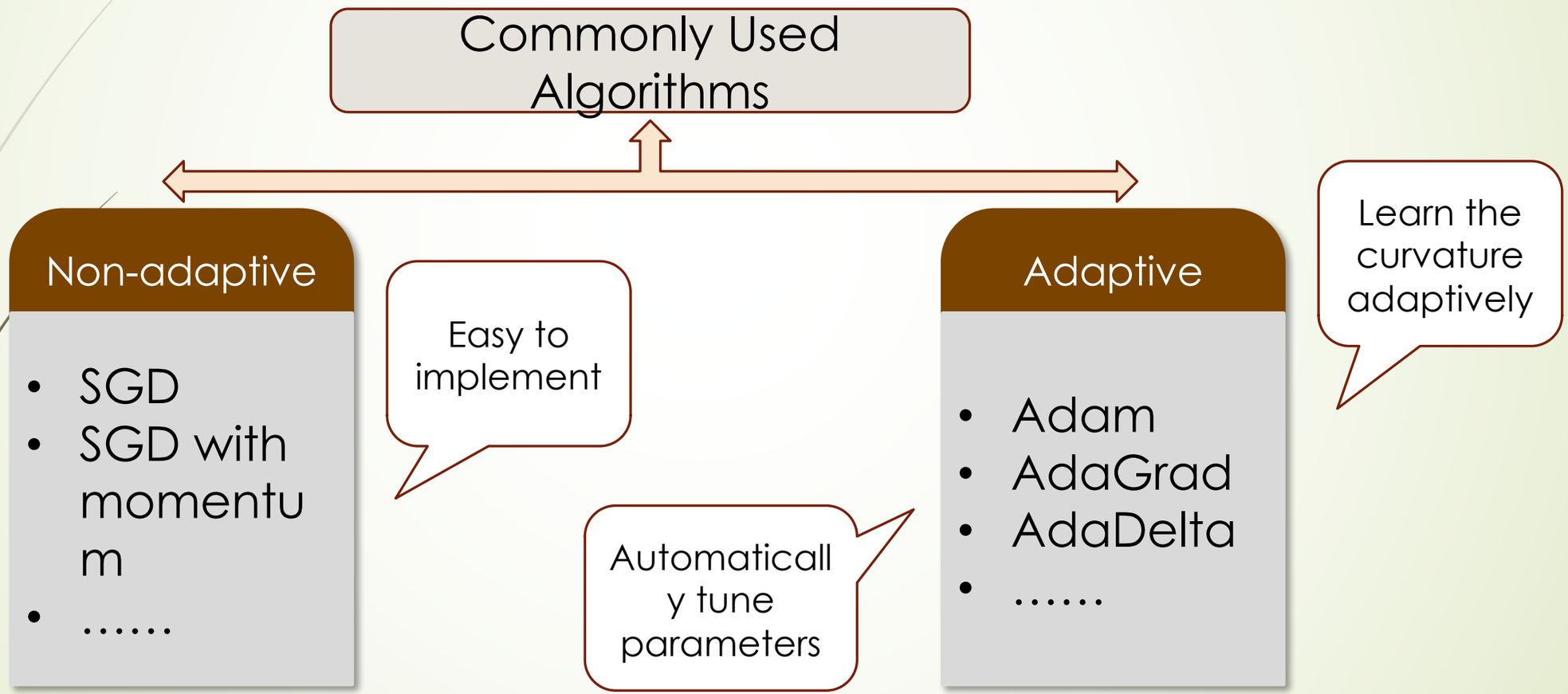
Capacity Measurement	Complexity
VC-dimension	$VC \leq O(\lvert E \rvert \log \lvert E \rvert)$
$\mathcal{E}$-Covering number	$\log_2 N_{l_1}(\mathcal{F}, \epsilon, m) \leq O\left(\frac{\left(AL_\phi\right)^{L(L+1)}}{\epsilon^{2L}}\right)$
Rademacher Average	$R_m(\mathcal{F}) \leq O(\mu^L)$

$|E|$: # of edges

L: # of layers

Big model should fail!

Training Algorithms for Deep Learning



Stochastic Gradient Descent

Objective loss function:

$$\min R_n(w) := \frac{1}{n}\sum_i l(w; x_i, y_i)$$

where $(x_i,\ y_i)$ is the data, w is the parameter vector.

Gradient Descent: $w_{t+1} = w_t - \frac{\eta}{n}\sum \nabla l(w_t; x_i, y_i)$

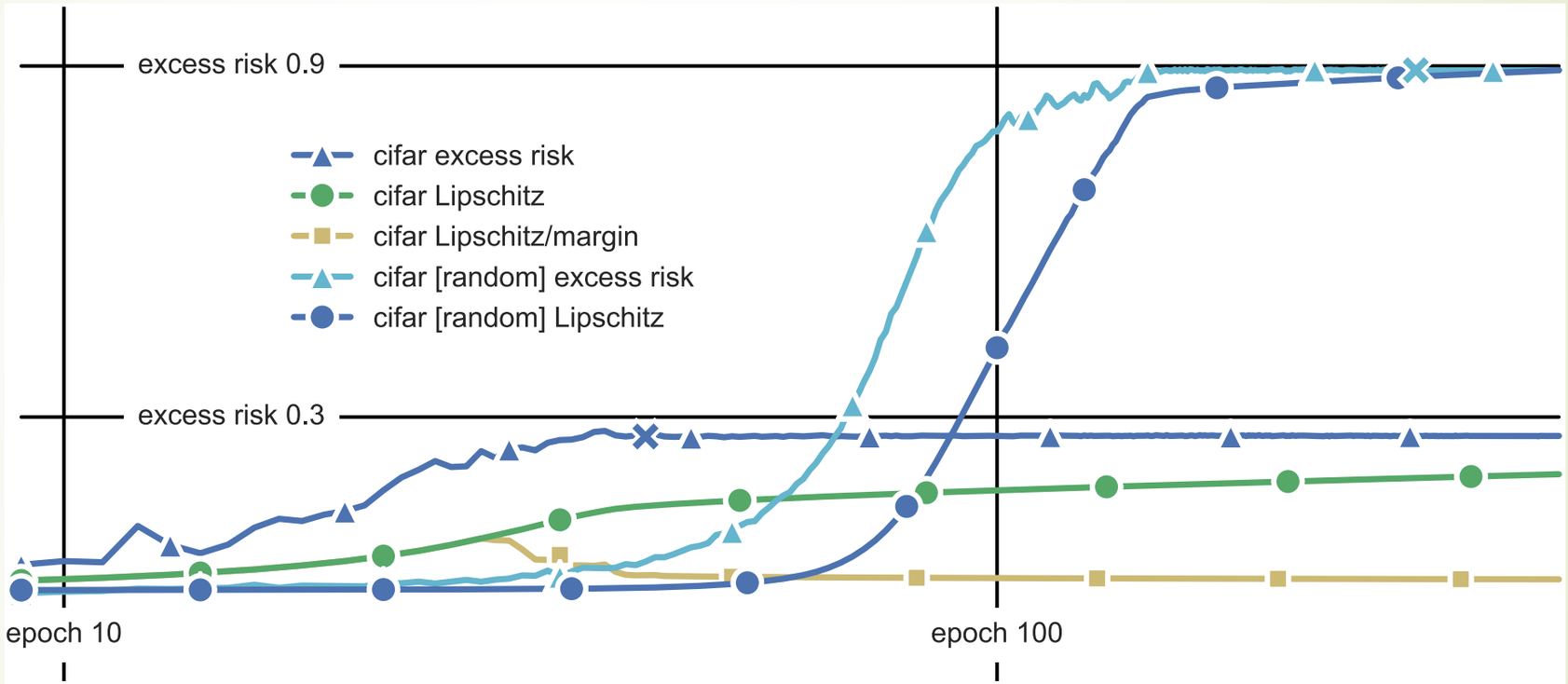
O(n) time complexity

SGD: $w_{t+1} = w_t - \eta \nabla l(w_t; x_{i_t}, y_{i_t})$, where i_t is uniform in {1, …, n}

O(1) time complexity

Extensions: mini-batch SGD

SGD with early stopping regularizes



Bartlett et al. 2017. Generalization error of AlexNet in Cifar10

Margin and Network Lipschitz based Generalization error bound (Bartlett et al. 2017)

Theorem 1.1. *Let nonlinearities $(\sigma_1, \ldots, \sigma_L)$ and reference matrices $(M_1, \ldots, M_L)$ be given as above (i.e., σ_i is ρ_i-Lipschitz and $\sigma_i(0) = 0$). Then for $(x, y), (x_1, y_1), \ldots, (x_n, y_n)$ drawn iid from any probability distribution over $\mathbb{R}^d \times \{1, \ldots, k\}$, with probability at least $1 - \delta$ over $((x_i, y_i))_{i=1}^n$, every margin $\gamma > 0$ and network $F_{\mathcal{A}} : \mathbb{R}^d \to \mathbb{R}^k$ with weight matrices $\mathcal{A} = (A_1, \ldots, A_L)$ satisfy*

$$\Pr\left[\arg\max_j F_{\mathcal{A}}(x)_j \neq y\right] \leq \widehat{\mathcal{R}}_\gamma(F_{\mathcal{A}}) + \widetilde{\mathcal{O}}\left(\frac{\|X\|_2 R_{\mathcal{A}}}{\gamma n}\ln(W) + \sqrt{\frac{\ln(1/\delta)}{n}}\right),$$

where $\widehat{\mathcal{R}}_\gamma(f) \leq n^{-1}\sum_i \mathbb{1}\left[f(x_i)_{y_i} \leq \gamma + \max_{j \neq y_i} f(x_i)_j\right]$ and $\|X\|_2 = \sqrt{\sum_i \|x_i\|_2^2}$.

The *spectral complexity* $R_{F_{\mathcal{A}}} = R_{\mathcal{A}}$ of a network $F_{\mathcal{A}}$ with weights $\mathcal{A}$ is the defined as

$$R_{\mathcal{A}} := \left(\prod_{i=1}^{L} \rho_i \|A_i\|_\sigma\right)\left(\sum_{i=1}^{L} \frac{\|A_i^\top - M_i^\top\|_{2,1}^{2/3}}{\|A_i\|_\sigma^{2/3}}\right)^{3/2}.$$

Stochastic Gradient/Discrete Langevin Dynamics (SGLD)

SGLD is a variant of SGD:

$$w_{t+1} = w_t - \eta \nabla l\left(w_t; x_{i_t}, y_{i_t}\right) + \sqrt{\frac{2\eta}{\beta}} z_t, \text{ where } z_t \sim \mathcal{N}(0, I_d)$$

Injection of Gaussian noise makes SGLD completely different with SGD

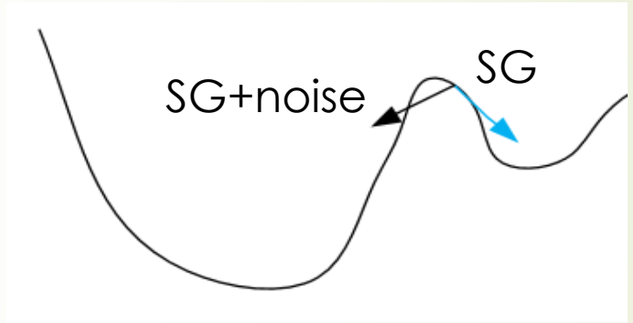
For small enough step size η_t, Gaussian noise will dominate the stochastic gradient.

Distinctions of SGLD

Intuitively, injected isotropic Gaussian noise helps escape saddle points or local minimum

SGLD is the discretization of following SDE

$$dW(t) = -\nabla F(W(t))dt + \sqrt{\frac{2}{\beta}}\,dB(t)$$



where $F(\cdot)$ is the empirical loss function, $B(t)$ is the standard Brownian motion

Its distribution converges to Gibbs distribution $\propto \exp(-\beta F(w))$

Large β will concentrate on the global minimizer of $F(w)$

Liwei Wang et al. 2017

From the view of stability theory:

Under mild conditions of (surrogate) loss function, the generalization error of SGLD at *N-th* round satisfies

$$E[l(w_S, z)] - E_S[l(w_S, z)] \leq O\left(\frac{1}{n}\left(k_0 + L\sqrt{\beta \sum_{k=k_0+1}^{N} \eta_k}\right)\right)$$

where L is the Lipschitz constant, and $k_0 := \min\{k : \eta_k \beta L^2 < 1\}$

If consider high probability form, there is an additional $\tilde{O}\left(\sqrt{1/n}\right)$ term

Lipschitz Bound by Liwei Wang et al. 2017

From the view of PAC-Bayesian theory:

For regularized ERM with $R(w) = \lambda\left|\left|w\right|\right|^2/2$. Under mild conditions, with high probability, the generalization error of SGLD at *N-th* round satisfies

$$E[E[l(w_S, z)]] - E_S[E[l(w_S, z)]] \leq O\left(\sqrt{\frac{\beta}{n}\sum_{k=1}^{N}\eta_k e^{-\lambda(T_N - T_k)/2} E[\left|\left|g_k\right|\right|^2]}\right)$$

where $T_k = \sum_{j=1}^{k}\eta_k$, g_k is the stochastic gradient in each round.

Comparison Two Results

Both bounds suggest "train faster, generalize better", which explain the random label experiments in ICLR17

- In expectation, stability bound has a faster $O\left(\frac{1}{n}\right)$ rate.
- PAC-Bayes bound is data dependent, and doesn't rely on Lipschitz condition.
- Effect of step sizes in PAC-Bayes exponentially decay with time.

The Landscape of Risks

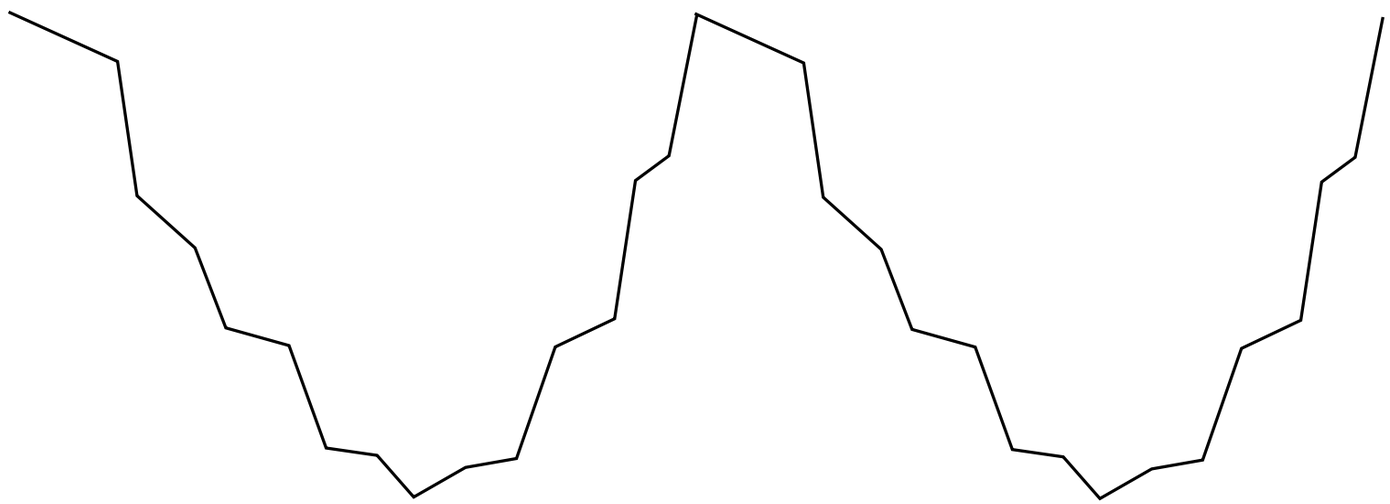
- However, when $\Phi(X;\Theta)$ is a large, deep network, current best mechanism to control generalization gap has two key ingredients:
  - Stochastic Optimization
    - "during training, it adds the sampling noise that corresponds to empirical-population mismatch" [Léon Bottou].
  - Make the model *as large as possible*.
    - see e.g. "Understanding Deep Learning Requires Rethinking Generalization", [Ch. Zhang *et al*, ICLR'17].

- We first address how overparametrization affects the energy landscapes $E(\Theta), \hat{E}(\Theta)$.

A `Deep' Dream:
All Critical Point/local optima = Global Optima?

- Choromanska-LeCun-Ben Arous'15: most of critical values are concentrated in a narrow bind of global optima, using random Morse theory on sphere (spin class models)
- Haeffele et al.'15,16: overparameterized tensor factorization models, every local optima are global optima
- Kawaguchi'16: linear networks have no poor local optima
- Bruna et al.'16,17: simple sublevel set topology of multilinear regression, with group symmetry, and some nonlinear networks
- Chaudhari et al'17: Moreau envelope of empirical risk
- Pennington & Bahri'17: Hessian Analysis using Random Matrix Theory

Non-convex



$$F(\theta) = F(g.\theta) \ , \ g \in G \text{ compact.}$$

- We can perturb any convex function in such a way it is no longer convex, but such that gradient descent still converges.
- E.g. quasi-convex functions.
- In particular, deep models have internal symmetries.

Linear Networks

- Some authors have considered linear "deep" models as a first step towards understanding nonlinear deep models:

$$E(W_1, \ldots, W_K) = \mathbb{E}_{(X,Y)\sim P} \|W_K \ldots W_1 X - Y\|^2 \ .$$

$$X \in \mathbb{R}^n \ , \ Y \in \mathbb{R}^m \ , \ W_k \in \mathbb{R}^{n_k \times n_{k-1}} \ .$$

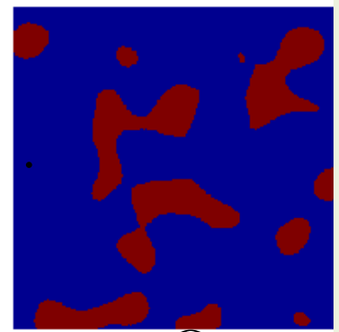
Theorem: [Kawaguchi'16] If $\Sigma = \mathbb{E}(XX^T)$ and $\mathbb{E}(XY^T)$ are full-rank and Σ has distinct eigenvalues, then $E(\Theta)$ has no poor local minima.

- studying critical points.
- later generalized in [Hardt & Ma'16, Lu & Kawaguchi'17]

Toplogy of Nonconvex Landscape

- Given loss $E(\theta)$, $\theta \in \mathbb{R}^d$, we consider its representation in terms of level sets:

$$E(\theta) = \int_0^\infty \mathbf{1}(\theta \in \Omega_u) du \ , \ \Omega_u = \{y \in \mathbb{R}^d \ ; \ E(y) \leq u\}$$



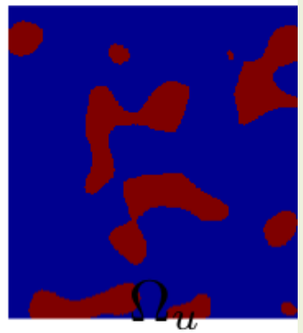
Ω_u

- A first notion we address is about the topology of the level sets .

- In particular, we ask how connected they are, i.e. how many connected components N_u at each energy level u?

Simple Topology

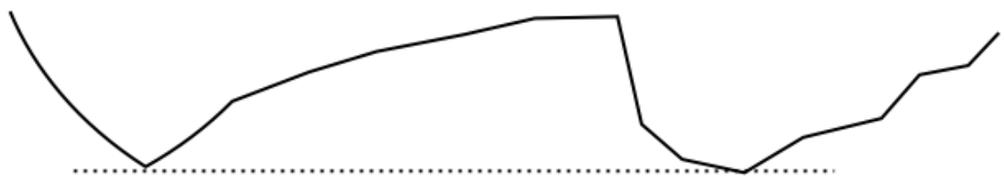
- A first notion we address is about the topology of the level sets .
  - In particular, we ask how connected they are, i.e. how many connected components N_u at each energy level u?
- This is directly related to the question of global minima:



Proposition: If $N_u = 1$ for all u then E has no poor local minima.

$$(\text{i.e. no local minima } y^* \text{ s.t. } E(y^*) > \min_y E(y))$$

- We say *E* is *simple* in that case.
- The converse is clearly not true.



Simple Topology of Linear Networks [Bruna-Freeman'16]

$$E(W_1, \dots, W_K) = \mathbb{E}_{(X,Y)\sim P} \|W_K \dots W_1 X - Y\|^2 \ .$$

Proposition: [BF'16]

1. If $n_k > \min(n, m)$, $0 < k < K$, then $N_u = 1$ for all u.

2. (2-layer case, ridge regression)
   $E(W_1, W_2) = \mathbb{E}_{(X,Y)\sim P} \|W_2 W_1 X - Y\|^2 + \lambda(\|W_1\|^2 + \|W_2\|^2)$
   satisfies $N_u = 1 \ \forall \ u$ if $n_1 > \min(n, m)$.

- We pay extra redundancy price to get simple topology.

Group Symmetries [Bruna-Venturi-Bandiera'17]

- Q: How much extra redundancy are we paying to achieve $N_u = 1$ instead of simply no poor-local minima?

  – In the multilinear case, we don't need $n_k > \min(n, m)$

  - We do the same analysis in the quotient space defined by the equivalence relationship $W \sim \tilde{W} \Leftrightarrow W = \tilde{W}U \ , \ U \in GL(\mathbb{R}^n) \ .$

Corollary [LBB'17]: The Multilinear regression $\mathbb{E}_{(X,Y)\sim P}\|W_1 \dots W_k X - Y\|^2$ has no poor local minima.

- Construct paths on the Grassmanian manifold of subspaces.
- Generalizes best known results for multilinear case (no assumptions on data covariance).

Nonlinear ReLU network

- Good behavior is recovered with nonlinear ReLU networks, provided they are sufficiently overparametrized:
- Setup: two-layer ReLU network:

$$\Phi(X;\Theta) = W_2\rho(W_1X)\ ,\ \rho(z) = \max(0,z).W_1 \in \mathbb{R}^{m\times n}, W_2 \in \mathbb{R}^m$$

Theorem [BF'16]: For any $\Theta^A, \Theta^B \in \mathbb{R}^{m\times n}, \mathbb{R}^m$, with $E(\Theta^{\{A,B\}}) \leq \lambda$, there exists path $\gamma(t)$ from Θ^A and Θ^B such that $\forall\ t\ , E(\gamma(t)) \leq \max(\lambda, \epsilon)$ and $\epsilon \sim m^{-\frac{1}{n}}$.

- Overparametrisation "wipes-out" local minima (and group symmetries).
- The bound is cursed by dimensionality, ie exponential in n .
- *Open question:* polynomial rate using Taylor decomp of $\rho(z)$?

Better Optimization Algorithms?

- Backpropagation Algorithm (made popular by Rumelhart-Hinton-Williams'1986) as stochastic gradient descent is equivalent to Larangian Multiplier method with gradient descent on weights (prox-linear)
  - Used in control theory (dynamic programming) in 1960s
  - **Qianxiao LI**'s talk yesterday (PMP in continuous case)
- It suffers from vanishing of gradients due to the chain rule of gradient map

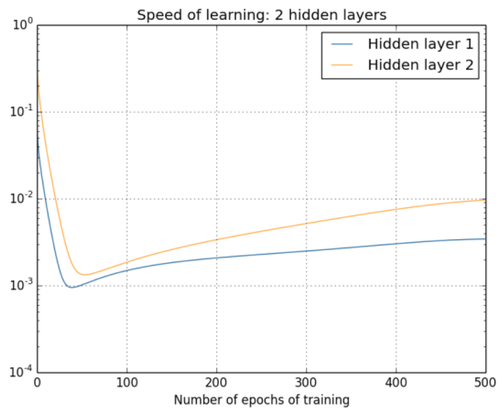
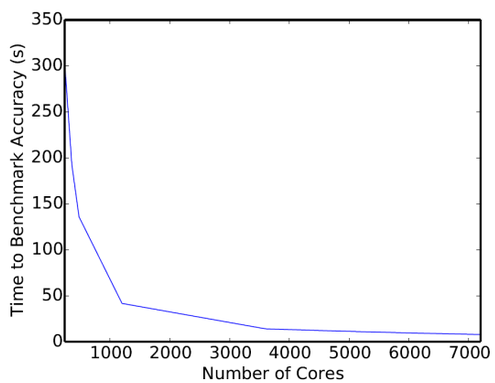


Figure: BP on MNIST[2]: Two hidden layers, speed of learning

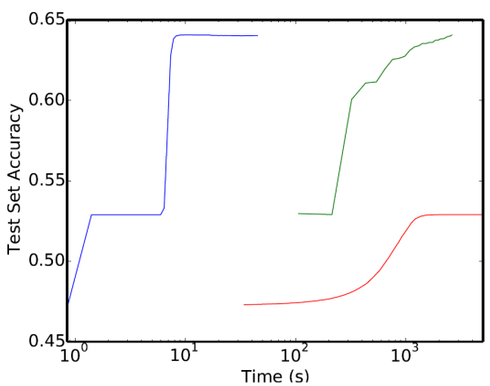
Alternative: (Augmented) Lagrangian Multiplier with Block Coordinate Descent

- ADMM-type: Taylor et al. ICML 2016
- BCD with zero Lagrangian multiplier: Zhang et al. NIPS 2017
- Discrete EMSA of PMP: **Qianxiao LI** et al 2017
  - No-vanshing gradients and parallelizable
- Some convergence theory: preliminary results on ADMM with **Jinshan Zeng'17**

Experiment results on Higgs dataset from Taylor et al'16



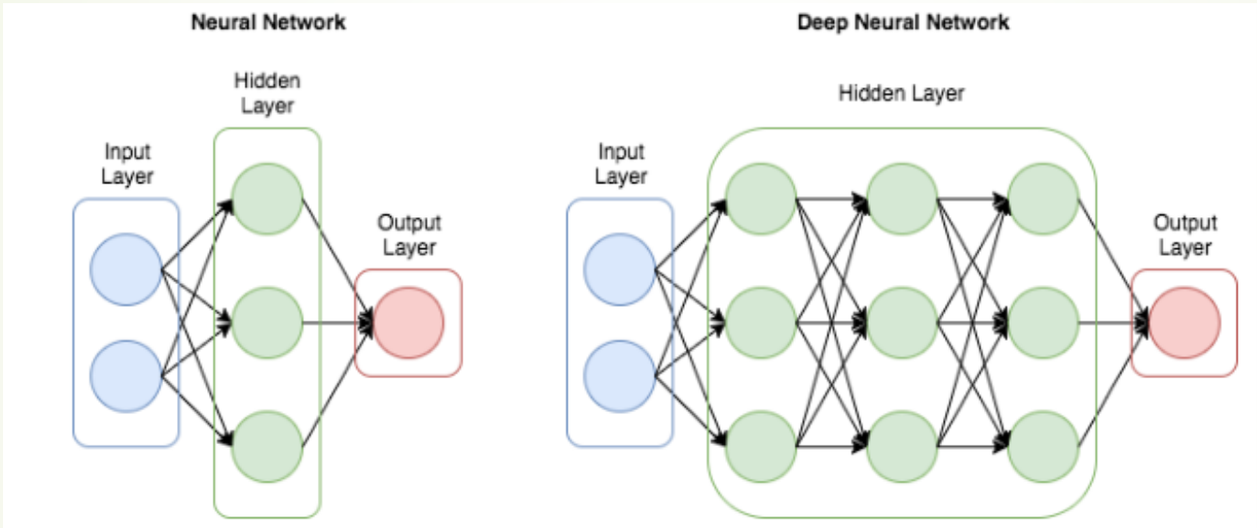
(a) **Time required for ADMM to reach 64% test accuracy when parallelized over varying levels of cores**. L-BFGS on a GPU required 181 seconds, and conjugate gradients required 44 minutes. SGD never reached 64% accuracy.



(b) **Test set predictive accuracy as a function of time** for ADMM on 7200 cores (blue), conjugate gradients (green), and SGD (red). Note the x-axis is scaled logarithmically.

ADMM for DNN Learning

- Deep Neural Network



- l layers (i.e., $l-1$ hidden layers)
  - Input: d_1 neurons
  - Output: d_2 neurons
  - Hidden layer i: r_i neurons
  - Layer j: $W_j \in \mathbb{R}^{r_j \times r_{j-1}}$, $j = 1, \ldots, l$
  - $r_0 := d_1$, $r_l := d_2$

ADMM for DNN Learning

- Deep Neural Network (DNN) Learning

- Neural network model: (omit "bias" for simplicity)

$$f(x; \{W_j\}_{j=1}^{l}) = W_l \sigma(W_{l-1} \sigma(\cdots \sigma(W_1 x)))$$

σ : activation function (e.g., sigmoid, ReLU)

- DNN Learning: given n samples $(x_i, y_i)_{i=1}^{n}$

$$\min_{\{W_j\}_{j=1}^{l}} \underbrace{\frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i; \{W_j\}_{j=1}^{l}), y_i)}_{\text{Empirical risk}} + \underbrace{\frac{\lambda}{2} \sum_{j=1}^{l} \| W_j \|_F^2}_{\text{Regularization}}$$

ℓ : loss function, e.g., ℓ_2, logistic, hinge

ADMM for DNN Learning

- Solved via ADMM

$$\min_{\{W_j\}_{j=1}^{l}} \ \ell(f(X;\{W_j\}_{j=1}^{l}),\, Y) + \frac{\lambda}{2}\sum_{j=1}^{l} \|W_j\|_F^2$$

- Auxiliary variables: let $V_0 := X$,

$$V_i = \sigma(W_i V_{i-1}),\ \ i = 1,\ldots,l-1; \qquad V_l = W_l V_{l-1}.$$

- DNN Learning $\Rightarrow$ ADMM-ready formulation:

$$\min_{\{W_j, V_j\}_{j=1}^{l}} \ \ell(V_l,\, Y) + \frac{\lambda}{2}\sum_{j=1}^{l} \|W_j\|_F^2$$

$$\text{s.t.} \quad V_i = \sigma(W_i V_{i-1}),\ \ i = 1,\ldots,l-1$$

$$V_l = W_l V_{l-1}.$$

ADMM for DNN Learning

- Solved via ADMM (Augmented Lagrangian)

- **Augmented Lagrangian Function**

$$\mathcal{L}(\{W_i\}_{i=1}^{l}, \{V_i\}_{i=1}^{l}, \{\Lambda_i\}_{i=1}^{l})$$

$$= \ell(f(X; \{W_j\}_{j=1}^{l}), Y) + \frac{\lambda}{2} \sum_{j=1}^{l} \|W_j\|_F^2$$

$$+ \sum_{i=1}^{l-1} \langle \Lambda_i, \sigma(W_i V_{i-1}) - V_i \rangle + \langle \Lambda_l, W_l V_{l-1} - V_l \rangle$$

$$+ \sum_{i=1}^{l-1} \frac{\beta_i}{2} \|\sigma(W_i V_{i-1}) - V_i\|_F^2 + \frac{\beta_l}{2} \|W_l V_{l-1} - V_l\|_F^2$$

ADMM for DNN Learning

- Solved via ADMM (KKT conditions)

KKT conditions:

$$
\begin{aligned}
0 &= \lambda_1 W_1 + (\Lambda_1 \odot \sigma'(W_1 V_0)) V_0^T, \\
0 &= \lambda_i W_i + (\Lambda_i \odot \sigma'(W_i V_{i-1})) V_{i-1}^T, \;\; i = 2, \ldots, l-1 \\
0 &= \lambda_l W_l + \Lambda_l V_{l-1}^T, \\
0 &= -\Lambda_i + W_{i+1}^T (\Lambda_{i+1} \odot \sigma'(W_{i+1} V_i)), \;\; i = 1, \ldots, l-2 \\
0 &= -\Lambda_{l-1} + W_l^T \Lambda_l, \\
0 &= -\Lambda_l + (V_l - Y), \\
0 &= \sigma(W_i V_{i-1}) - V_i, \;\; i = 1, \ldots, l-1 \\
0 &= W_l V_{l-1} - V_l
\end{aligned}
$$

ADMM for DNN Learning

- Solved via ADMM (Algorithm)

Algorithm 1 ADMM for DNN Learning

Samples: $Z := \{(x_i, y_i)\}_{i=1}^n$, $X := [x_1, \ldots, x_n] \in \mathbb{R}^{d_1 \times n}$, $Y := [y_1, \ldots, y_n] \in \mathbb{R}^{d_2 \times n}$, $V_0 := X$
Initialization: $\{W_i^0, V_i^0, \Lambda_i^0\}_{i=1}^l$
Parameters: $\lambda_i,\ \beta_i > 0, i = 1, \ldots, l$

for k=1,... **do**
  (Backward Estimation)
  for i= l :-1:1 **do**
    $W_i^k \leftarrow \arg\min_{W_i}\ \mathcal{L}(W_{<i}^{k-1}, W_i, W_{>i}^k, \{V_i^{k-1}\}_{i=1}^l, \{\Lambda_i^{k-1}\}_{i=1}^l)$
  end for
  (Forward Correction)
  for j= 1: l **do**
    $V_i^k \leftarrow \arg\min_{V_i}\ \mathcal{L}(\{W_i^k\}_{i=1}^l, V_{<i}^k, V_i, V_{>i}^{k-1}, \{\Lambda_i^{k-1}\}_{i=1}^l)$
  end for
  (Updating Multipliers)
  $\Lambda_1^k \leftarrow \Lambda_1^{k-1} + \beta_1(\sigma(W_1^k V_0) - V_1^k)$
  $\Lambda_i^k \leftarrow \Lambda_i^{k-1} + \beta_i(\sigma(W_i^k V_{i-1}^k) - V_i^k),\ i = 2, \ldots, l-1$
  $\Lambda_l^k \leftarrow \Lambda_l^{k-1} + \beta_l(W_l^k V_{l-1}^k - V_l^k)$
  $k \leftarrow k + 1$
end for

ADMM for DNN Learning

- Convergence of ADMM

  - ℓ_2 loss + sigmoidal activation function (i.e., $\sigma(u) = \frac{1}{1+e^{-u}}$)

Theorem (Convergence of ADMM for DNN Learning)

If the parameters $\{\beta_i\}_{i=1}^{l}$ are sufficiently large, then ADMM converges to a KKT point of DNN learning problem from any finite initial point.

ADMM for DNN Learning

- Convergence of ADMM (3-layers for example)

- $\mathcal{B} := \max\left\{ \frac{2\sqrt{r_1}}{\|X\|_F/\sqrt{n}},\ \left(\frac{2\sqrt{r_2}}{\|X\|_F/\sqrt{n}}\right)^{1/2},\ \left(\frac{\|Y\|_F}{\|X\|_F}\right)^{1/3} \right\}$

- Assumptions on β_i's:

  (a) $\beta_3 \geq 2$,

  (b) $\beta_2 \geq 16(\beta_3+1)\mathcal{B}^2$,

  (c) $\beta_1 \geq 11(\beta_2+1)\mathcal{B}^4\|X\|_F$.

- Assumption on λ :

$$\lambda \geq \max\left\{ \frac{3}{4}\beta_1\mathcal{B}\|X\|_F^3,\ 10\beta_2\mathcal{B}^4\|X\|_F^3,\ 4\beta_3\mathcal{B}^4\|X\|_F^2 \right\}$$

Notes: regularization parameters may grow with number of layers in the worst case!

Thank you!