Going off the Deep End

Deep Learning for Signal Recovery

Richard Baraniuk

Rice University

cycles

models ***data***

babylon

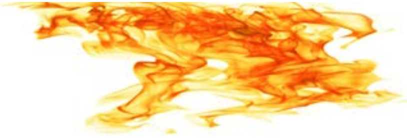
- Babylonians were **obsessed with data** and calculating (polynomial curve fitting)

- Knew about all the key theorems of the day and were extraordinary at **predicting** astronomical events

- Students learned math by working out **large numbers of problems** until they "understood" the general concept

- **Incapable** of scaffolding theorems together to create something larger

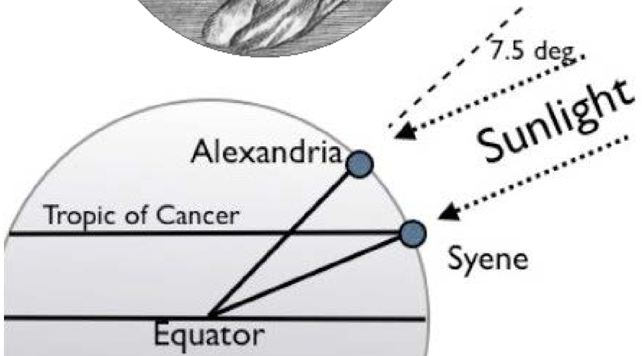
greece



- Ancient Greeks were **obsessed with models**
  - Ex: stars, sun, planets, moon are holes in a colossal cosmic colander that reveals the eternal fire beyond

- Such a (bad) model, inspired Eratosthenes to use geometry to deduce the **radius of the earth**

- Such a **deduction** would never have occurred to a Babylonian curve fitter



[Feynman, 1964 ...]

cycles

models

data

easy inference problems

- Great success on **"easy" inference problems** where accurate, tractable models are easy to develop (Greek approach)

- **Least squares** for optimal estimation
- **Optimal matched filtering** to detect signals in noise
- **Optimal Wiener filtering** to separate signals from noise
- **Optimal Kalman filtering** to track signals in noise
- **AR / ARMA** modeling
- **Sparse recovery/Lasso** (when the features are actually sparse)
- **SVM** (when the data is actually linearly separable)

...

wicked hard inference problems

- Historically, less success on **“wicked hard” problems** like **machine perception**
  - Ex: object/speech recognition, image priors, robot navigation, …

wicked hard inference problems

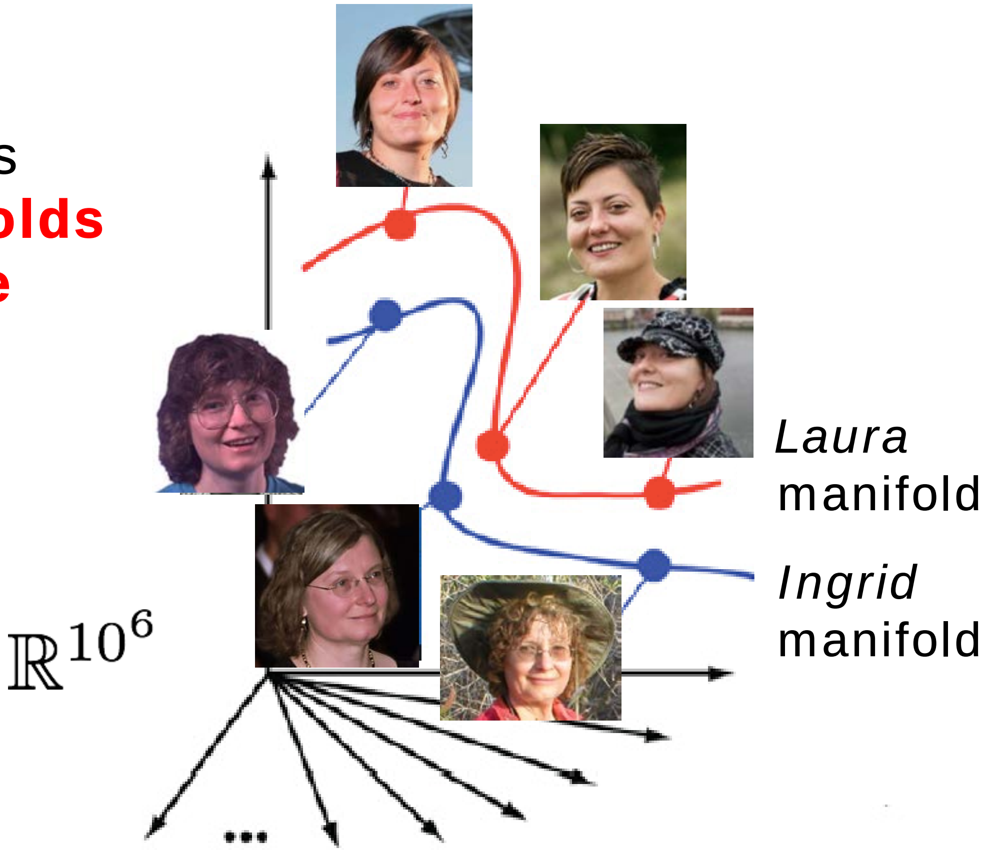
- Historically, less success on **"wicked hard" problems** like **machine perception**
  - Ex: object/speech recognition, image priors, robot navigation, ...

- Key challenge: Perception plagued by large amounts of **nuisance variation**
  - Ex: in object recognition: changes in location, pose, viewpoint, lighting, expression, occlusion, ...
  - Some nuisances might not be known explicitly

what makes perception wicked hard?

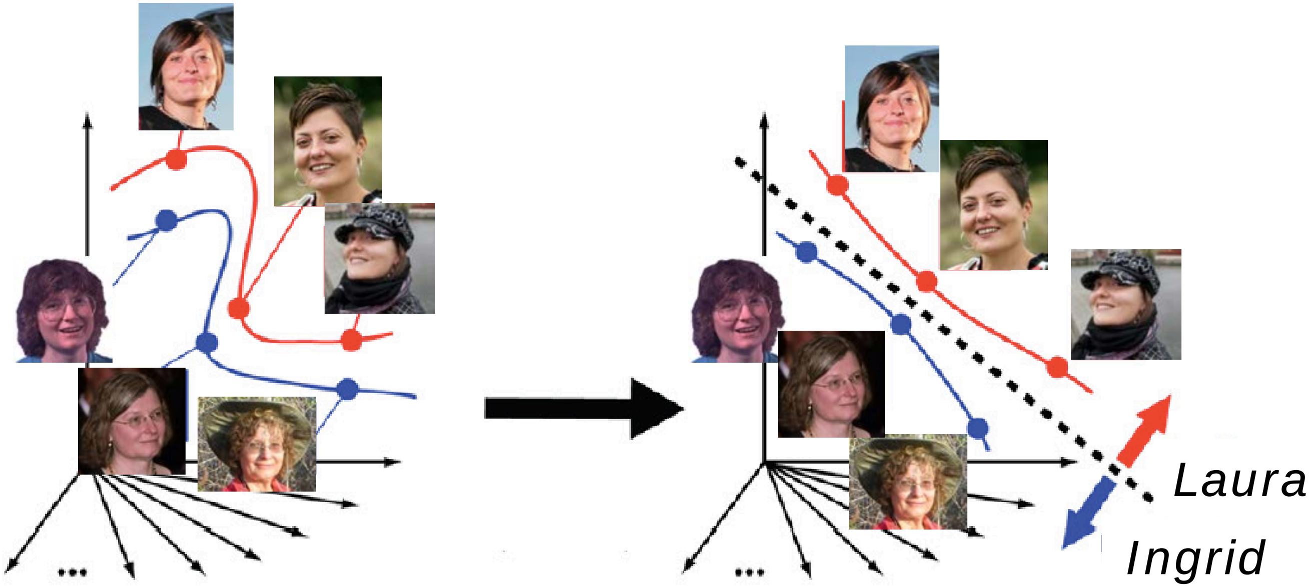
- Problem: Nuisance variations generate **entangled manifolds** in **high-dimensional space**

- Ex: Classify *Ingrid* vs. *Laura*



[DiCarlo et al. Neuron 2012]

holy grail of machine perception

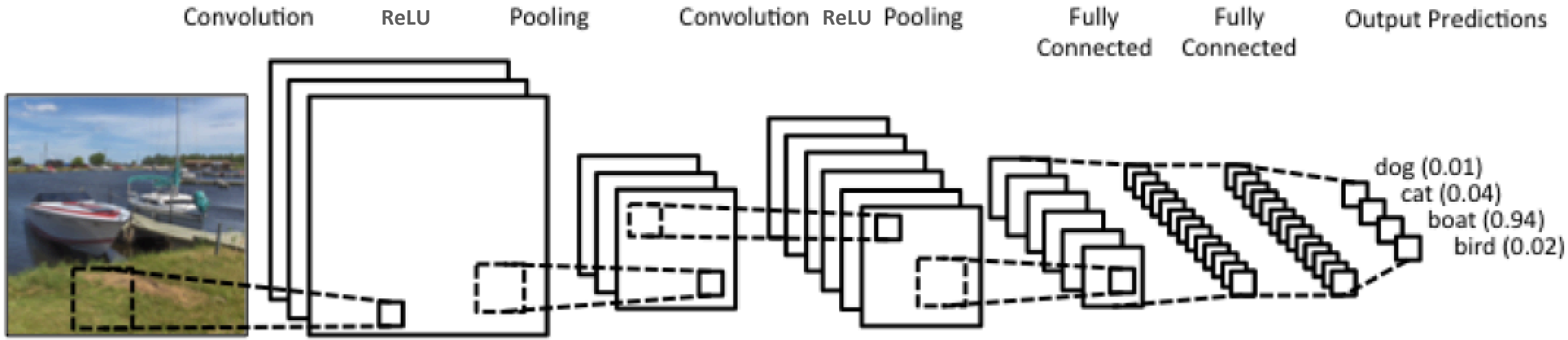


- Learn a **model** that **disentangles** (factors out) nuisance variations, leaving meaningful intrinsic degrees of freedom
  - In the past: Fourier transform, cepstrum, wavelets, K-SVD, SIFT, ...

deep learning

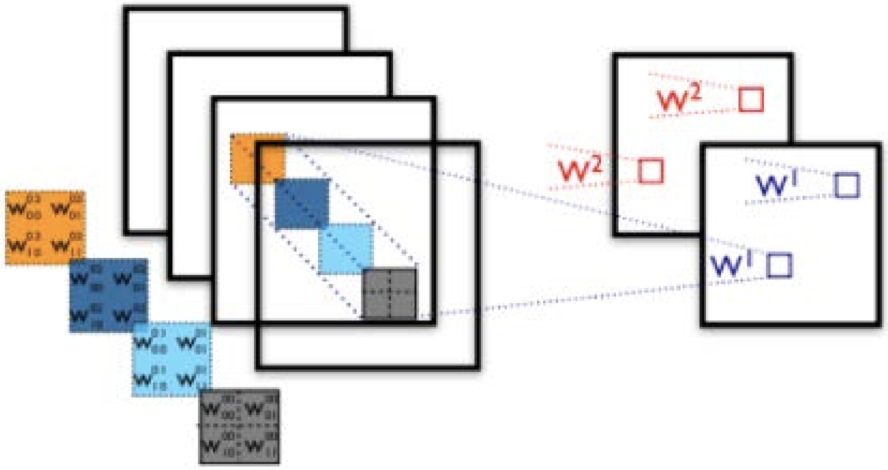
- Today, **Babylonians** are making all the progress

- How? **Black box deep neural networks**
  - **multi-scale** architecture combining adaptive filters with simple nonlinearities
  - **convolution | thresholding | sub-sampling**



convolution

- **Convolve image** with a set of filters
  - When local image patch **resembles** filter weights, then output is **large**
  - Otherwise, output is **small**



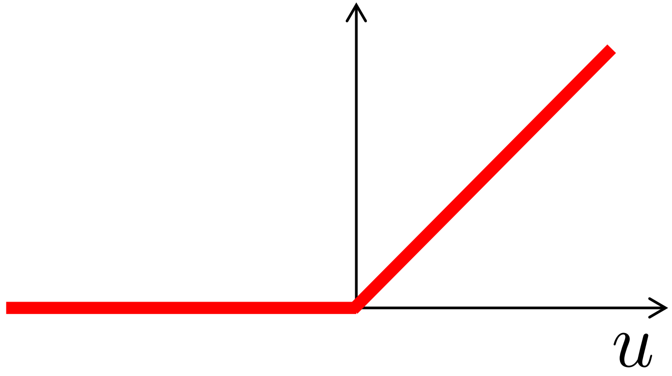
- Most nets also add a **bias** to the convolution output (affine)
- Filter **weights** and **biases** are the only **parameters** in the net

rectifier

- **Rectified linear unit (ReLU)**

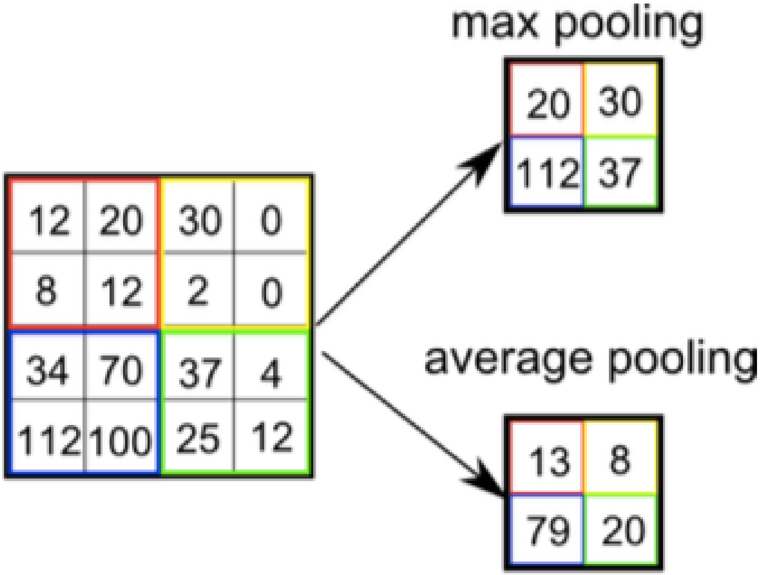
- Discard negative values

$$\mathrm{ReLU}(u) = \max(u, 0)$$

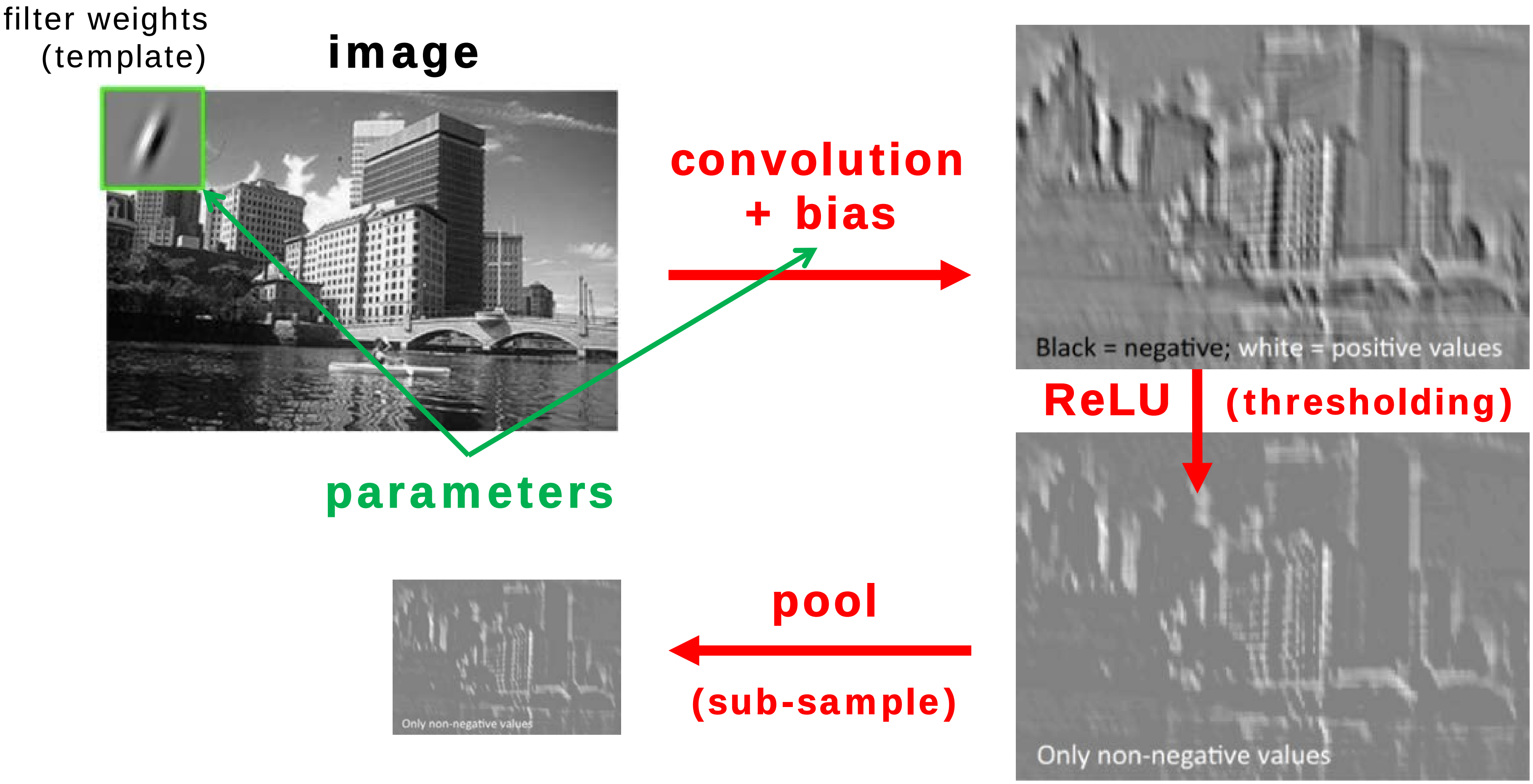


sub-sampling

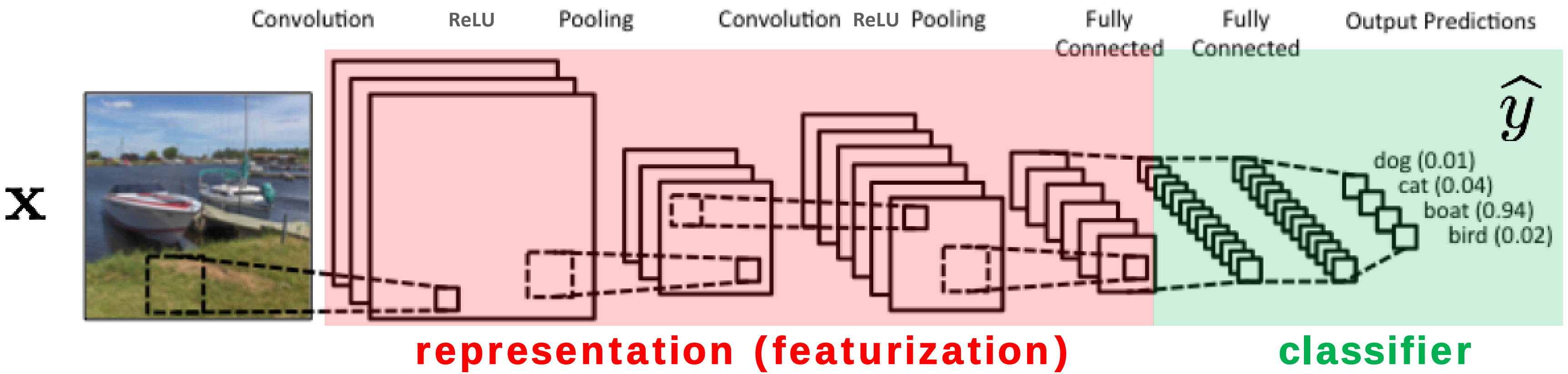
- Down-sample to **reduce dimensionality** of subsequent layers
  - Average pooling (linear)
  - Max pooling (nonlinear)
  - Channel pooling



convo | ReLU | max-pool

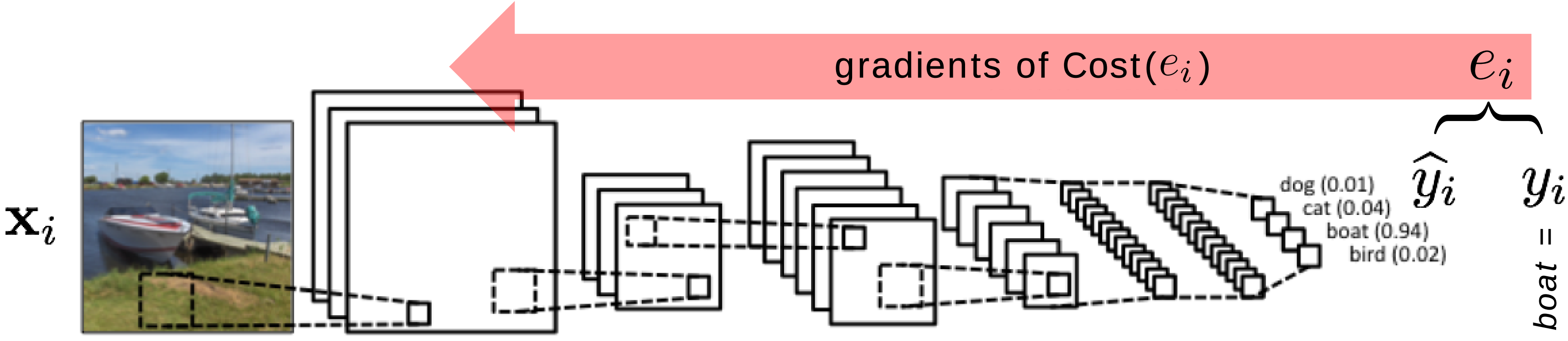


inference



- Output **convnet** into an **old-school neural network** (1990s) (**multinomial logistic regression** aka “softmax”)
- For **classification**, use a **“1-hot encoding”** of the classes (likelihood **histogram** over classes)

learning



- Estimate the **parameters** (convo filter weights, biases) given a (large) set of **labeled training data** $(\mathbf{x}_i, y_i)_{i=1}^{n}$
- **Cost function** that quantifies prediction **errors** on training data ($\widehat{y_i}$ vs. y_i)
  - Ex: **cross-entropy** for classification, squared error for regression
- Optimize cost function via **stochastic gradient descent** with the gradient computed via **backpropagation** (chain rule of calculus)

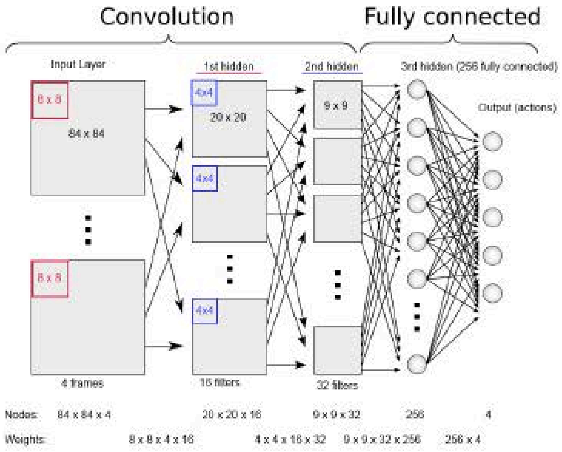
deep nets – a perfect storm

ca. 2012

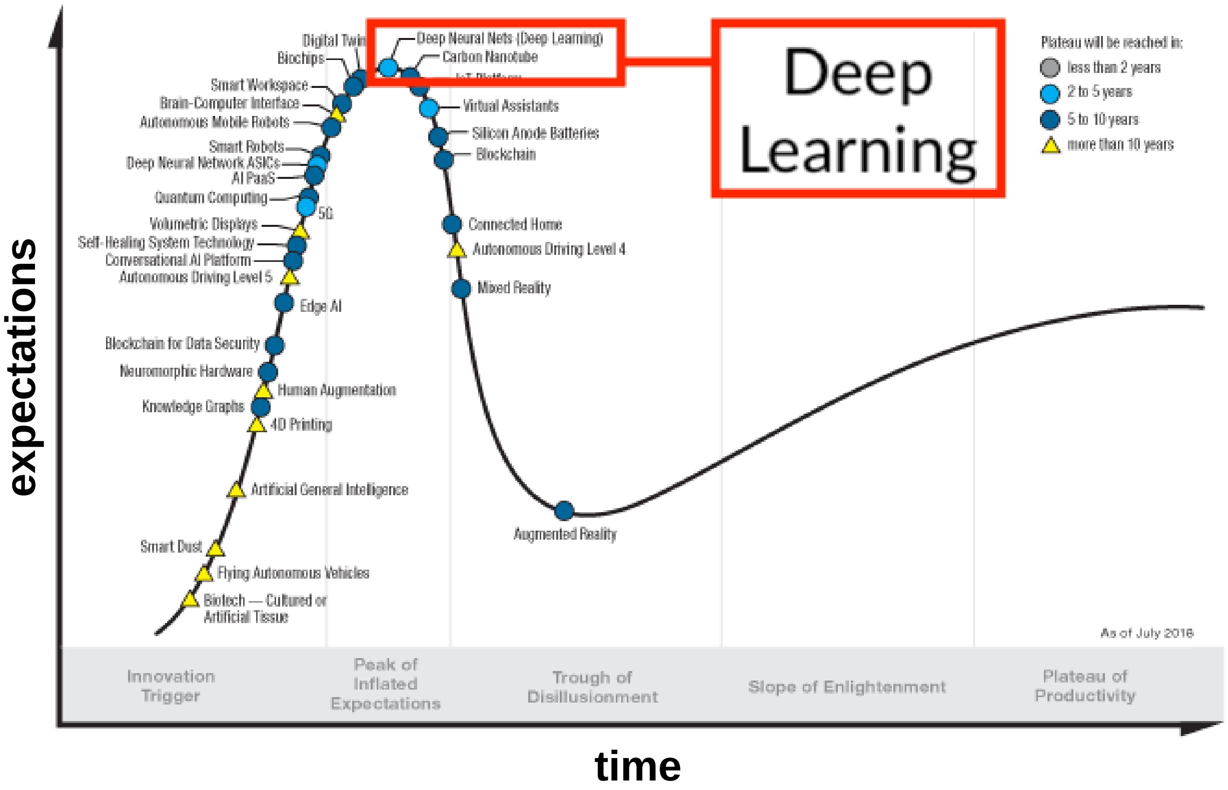
Big Data

Big Computers

Deep Architectures



Hype Cycle for Emerging Technologies, 2018



gartner.com/SmarterWithGartner

Source: Gartner (August 2018)

Gartner.

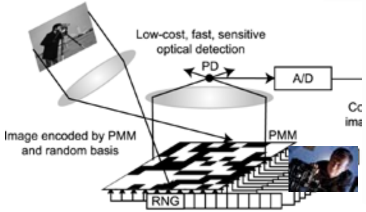
the cranberries

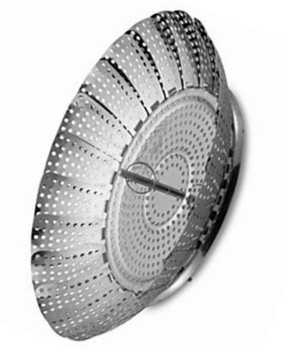
EVERYBODY ELSE IS DOING IT, SO WHY CAN'T WE?

25th Anniversary Edition

Remastered at Abbey Road Studios

inverse problems



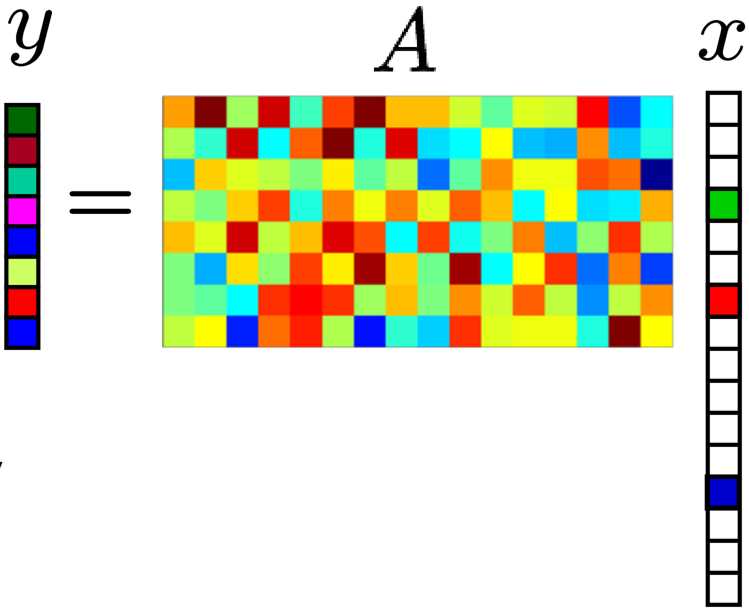


- Ex: **Compressive** cameras, radars, MRIs

- Typically assume:

  known **forward operator** A

  known **image model** for x



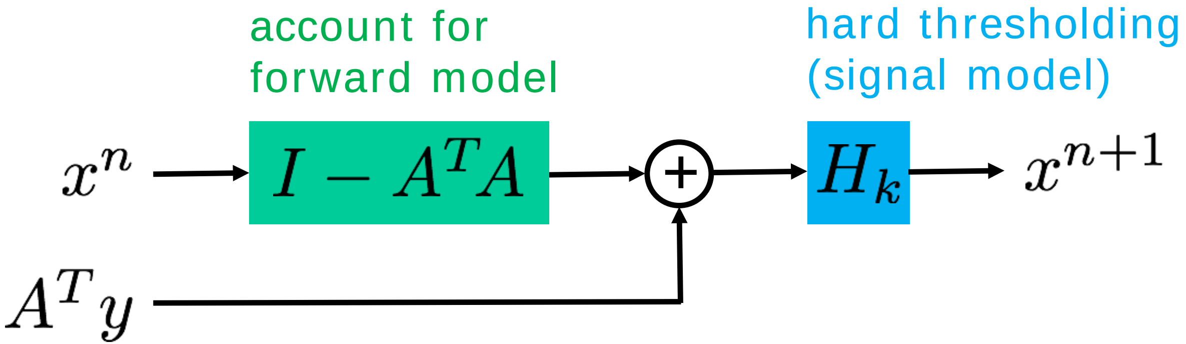
- Challenges:
  - Image modeling is **wicked hard**
  - Forward operator never known exactly

recovery by optimization

- **Goal:** Given $y = Ax$ find x
- Now-classical signal model: x is k-**sparse** in some basis
- Numerous **iterative optimizations** for recovery
  - Lasso
  - BPDN
  - AMP
  - ADMM
  - Bregman
  - **Iterative hard thresholding**
  - ...

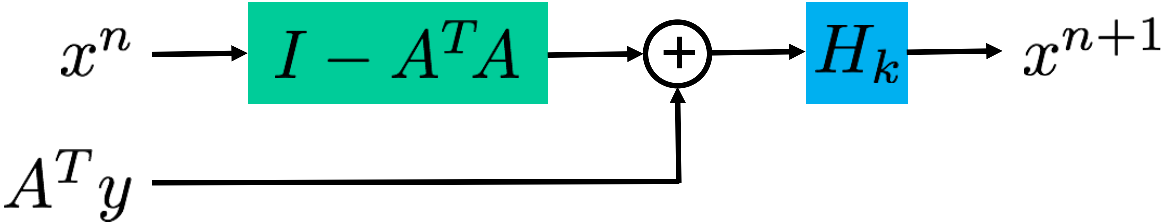
iterative hard thresholding

- **Goal:** Given $y = Ax$ find x
- Standard signal prior: x is k-sparse in some basis
- **Iterative hard thresholding** algorithm
  - Given an initial guess x^0
  - For $n = 0, 1, \ldots$ do $x^{n+1} = H_k[x^n + A^T(y - Ax^n)]$

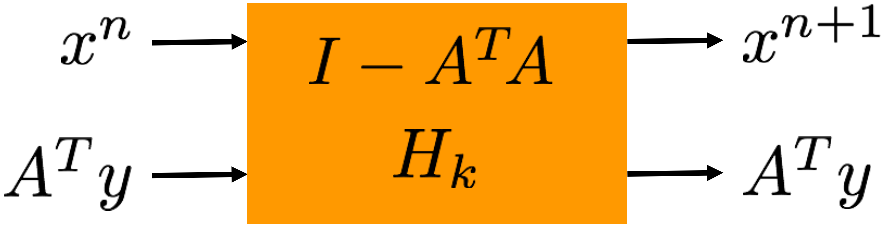


network for recovery

- Encapsulate operations from one **iteration** of the algorithm

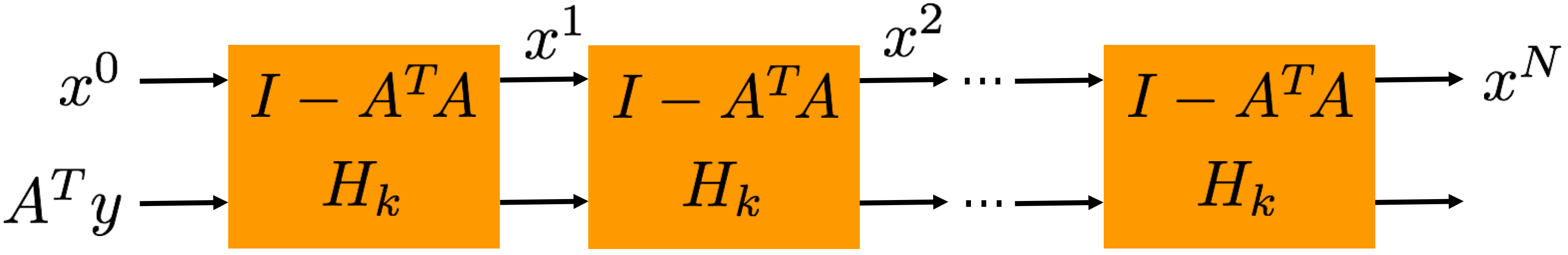


... into one block of the **network**



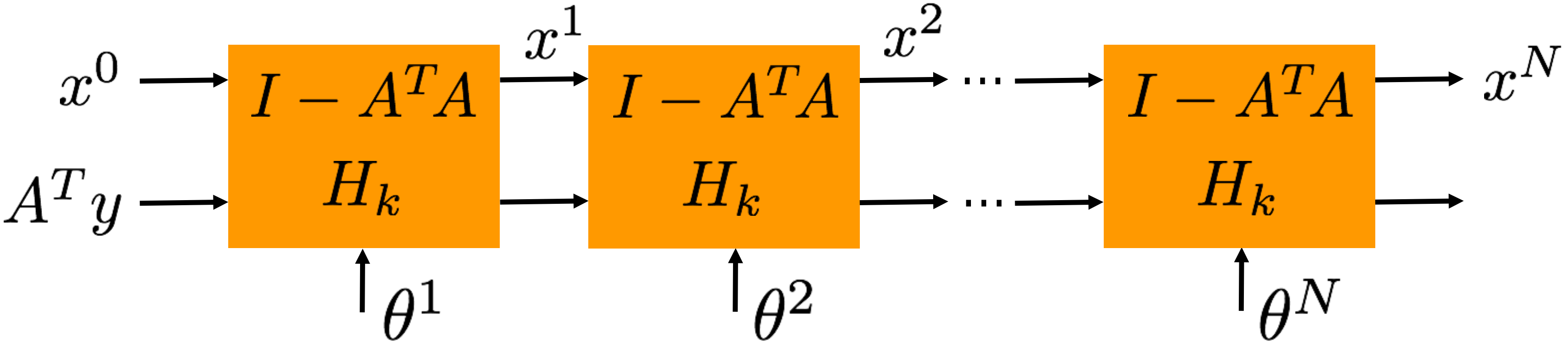
network for recovery

- Unrolled network **equivalent** to *N* iterations of the algorithm



network for recovery

- Unrolled network **equivalent** to *N* iterations of the algorithm



- Replace assumed **forward model** and **signal model** with **more flexible models** that can be learned from data
- Ex: Replace wavelet thresholding H_k with **deep net denoisers**

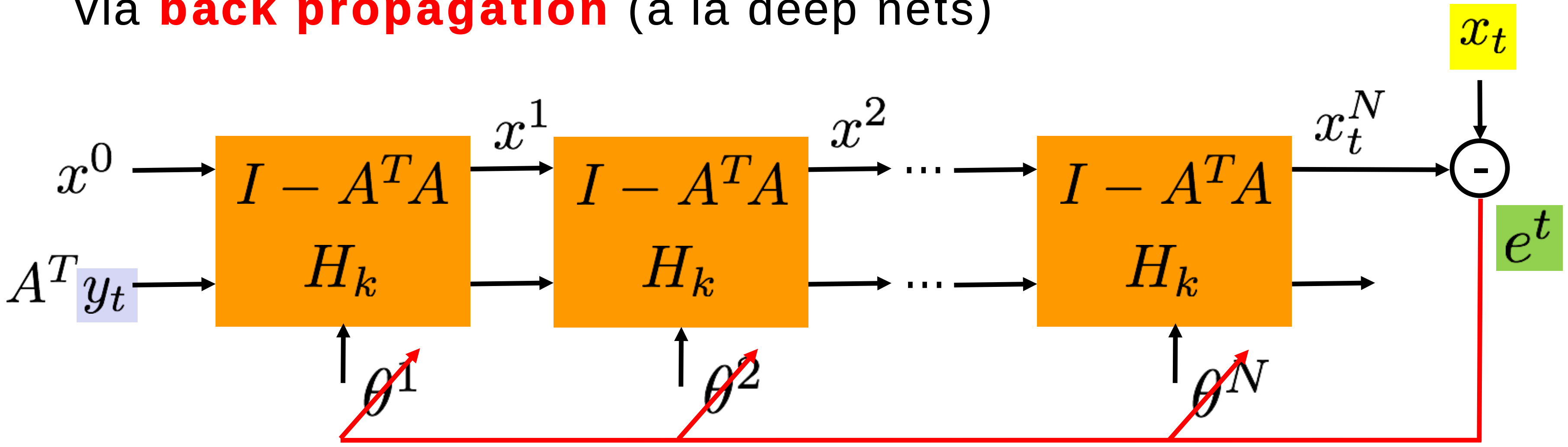
(ex: DnCNN, ECNDNet, ...)

optimized network

- Construct training data $\{x_t, y_t = Ax_t\}_{t=1}^{T}$ and use **training error**

$$e_t = x_t - x_t^N$$

to optimize the parameters of the network via **back propagation** (a la deep nets)



sparse recovery (20x undersampling)

TVAL3
6.85 sec
26.4 dB

BM3D-AMP
75.04 sec
27.2 dB

Learned D-DAMP
1.22 sec
28.1 dB

A. Mousavi, C. Metzler, RB, “Learned D-AMP: Principled Neural-Network-based Compressive Image Recovery,” NIPS 2017

phase retrieval

(4x Fourier measurements w/ Poison noise)

Hybrid Input/Ouput
(40s)

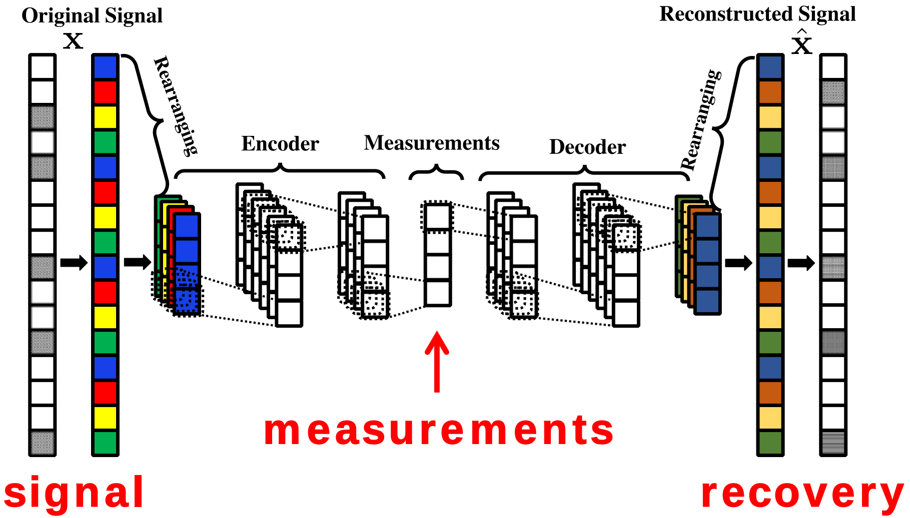
BM3D-AMDD
(144s)

prDeep
(75s)

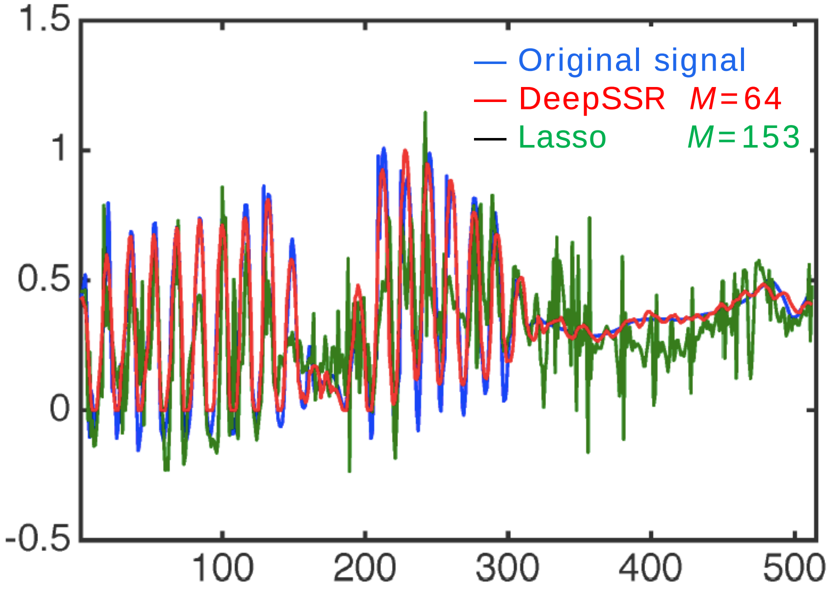
C. Metzler, P. Schniter, A. Veeraraghavan, RB, “prDeep: Robust Phase Retrieval with a Flexible Deep Network,” ICML 2018

nonlinear sensing

- Can design matched **nonlinear sensing / recovery** schemes based on deep nets

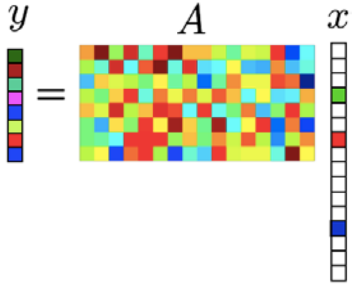


N=512 Nyquist samples
K=64 wavelet-sparse

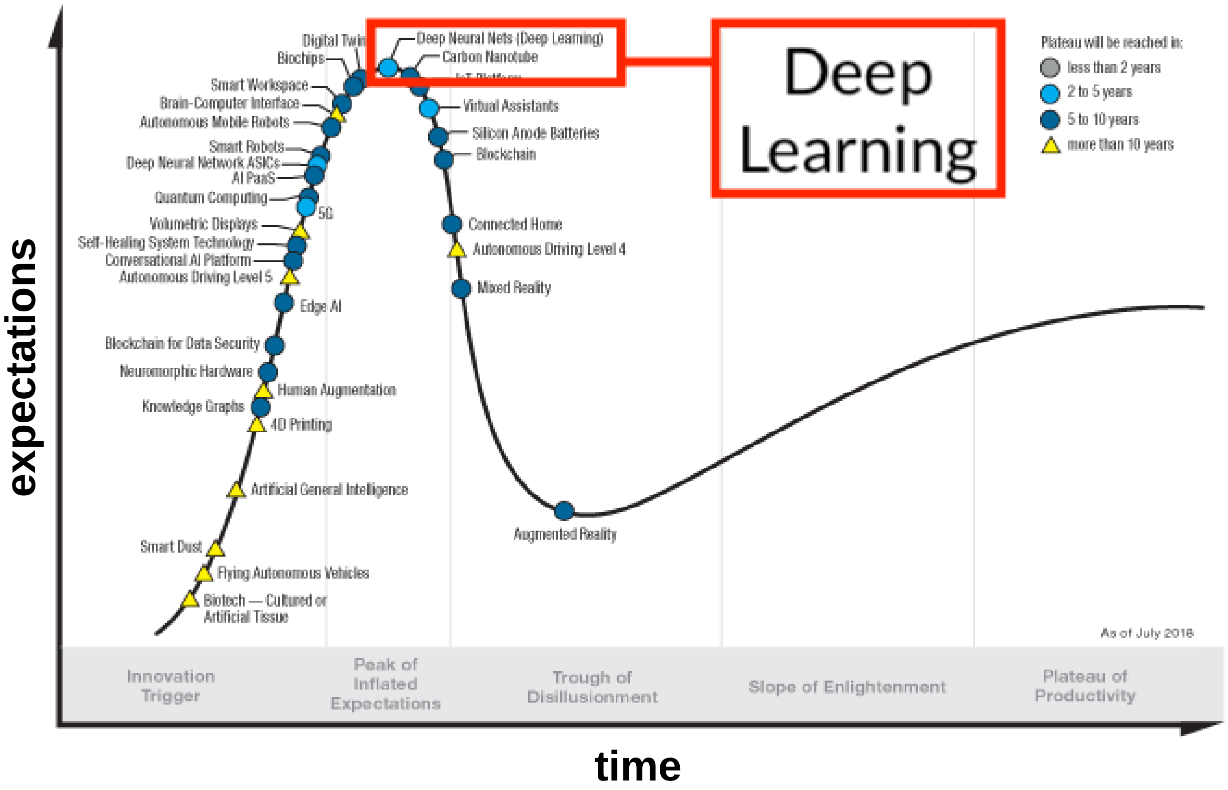


A. Mousavi, G. Dasarathy, RB, “DeepCodec: Adaptive Sensing and Recovery via Deep Convolutional Neural Networks,” ICLR 2019

CS is growing up

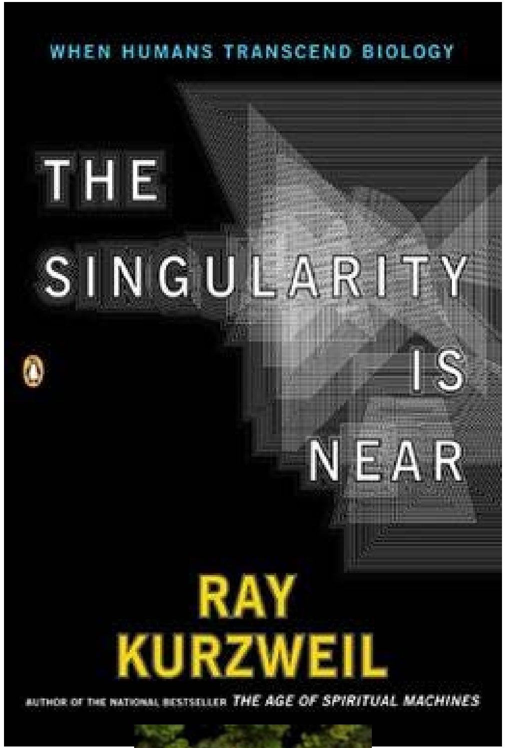


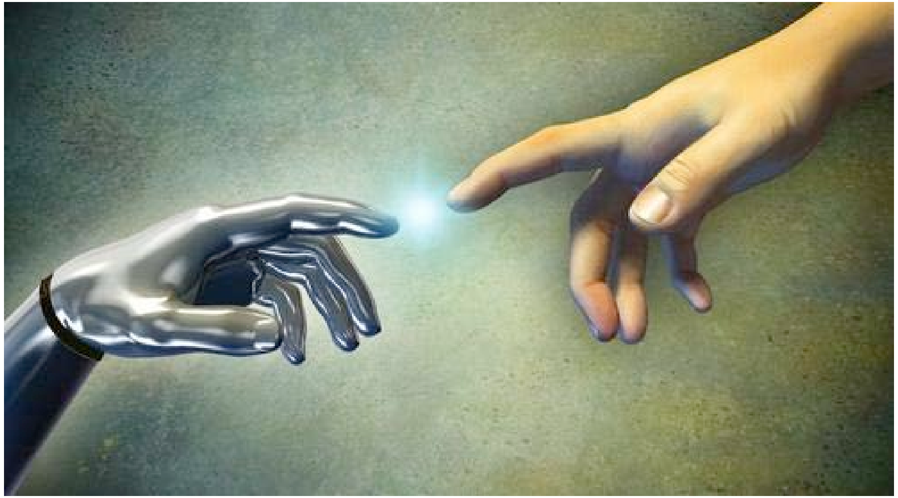
Hype Cycle for Emerging Technologies, 2018

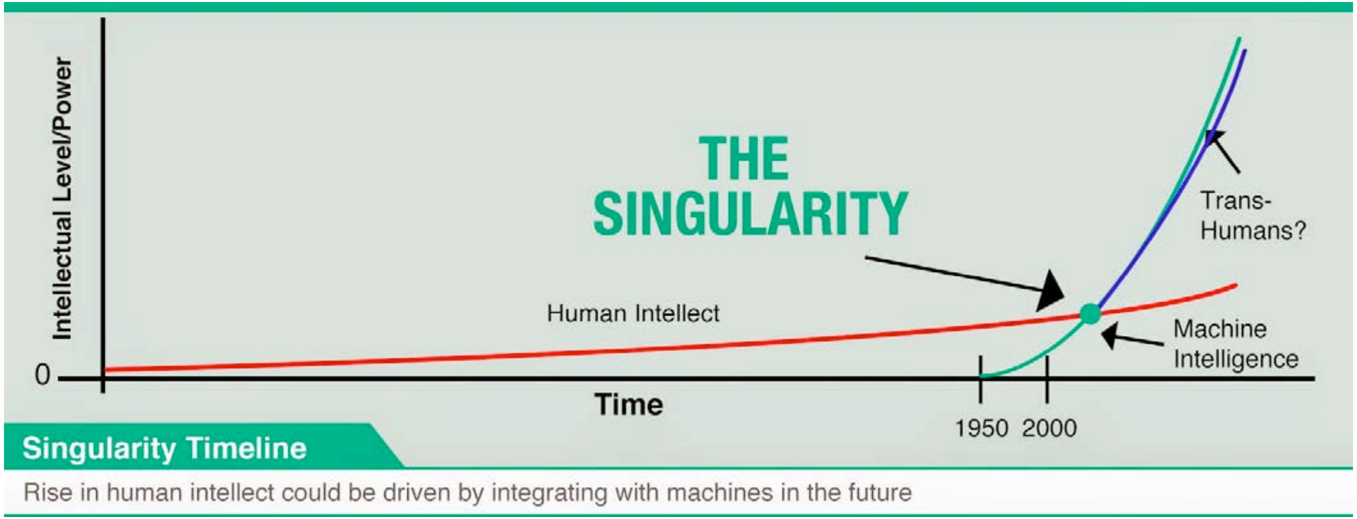


gartner.com/SmarterWithGartner

Source: Gartner (August 2018)


WHEN HUMANS TRANSCEND BIOLOGY
THE
SINGULARITY
IS
NEAR
RAY
KURZWEIL
AUTHOR OF THE NATIONAL BESTSELLER THE AGE OF SPIRITUAL MACHINES




Intellectual Level/Power
0
Time
THE
SINGULARITY
Human Intellect
Trans-
Humans?
Machine
Intelligence
1950
2000
Singularity Timeline
Rise in human intellect could be driven by integrating with machines in the future


IWDRM

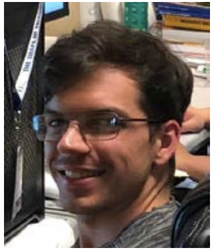
greek questions for the *babylonians*

- Why is deep learning so **effective**?
- Can we derive deep learning systems from **first principles**?
- When and why does deep learning **fail**?
- How can deep learning systems be improved and extended in a **principled** fashion?
- Where is the **foundational framework** for theory?

See also Mallat, Soatto, Arora, Poggio, Tishby, [growing community] ...

splines

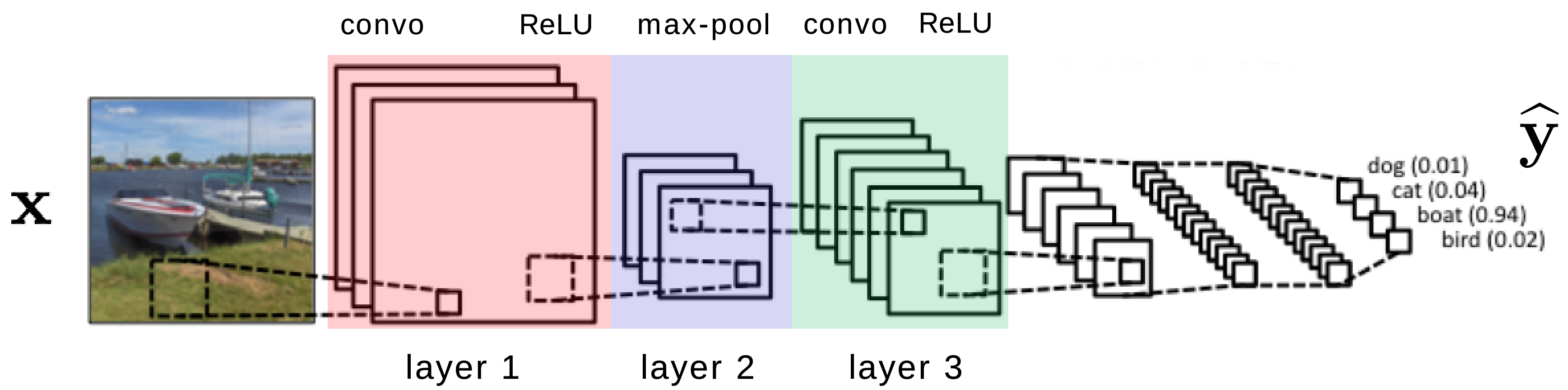
and deep learning



R. Balestriero & RB
"A Spline Theory of Deep Networks," *ICML* 2018
"Mad Max: Affine Spline Insights into Deep Learning," arxiv.org/abs/1805.06576, 2018
"From Hard to Soft: Understanding Deep Network Nonlinearities...," *ICLR* 2019
"A Max-Affine Spline Perspective of RNNs," *ICLR* 2019 (w/ J. Wang)

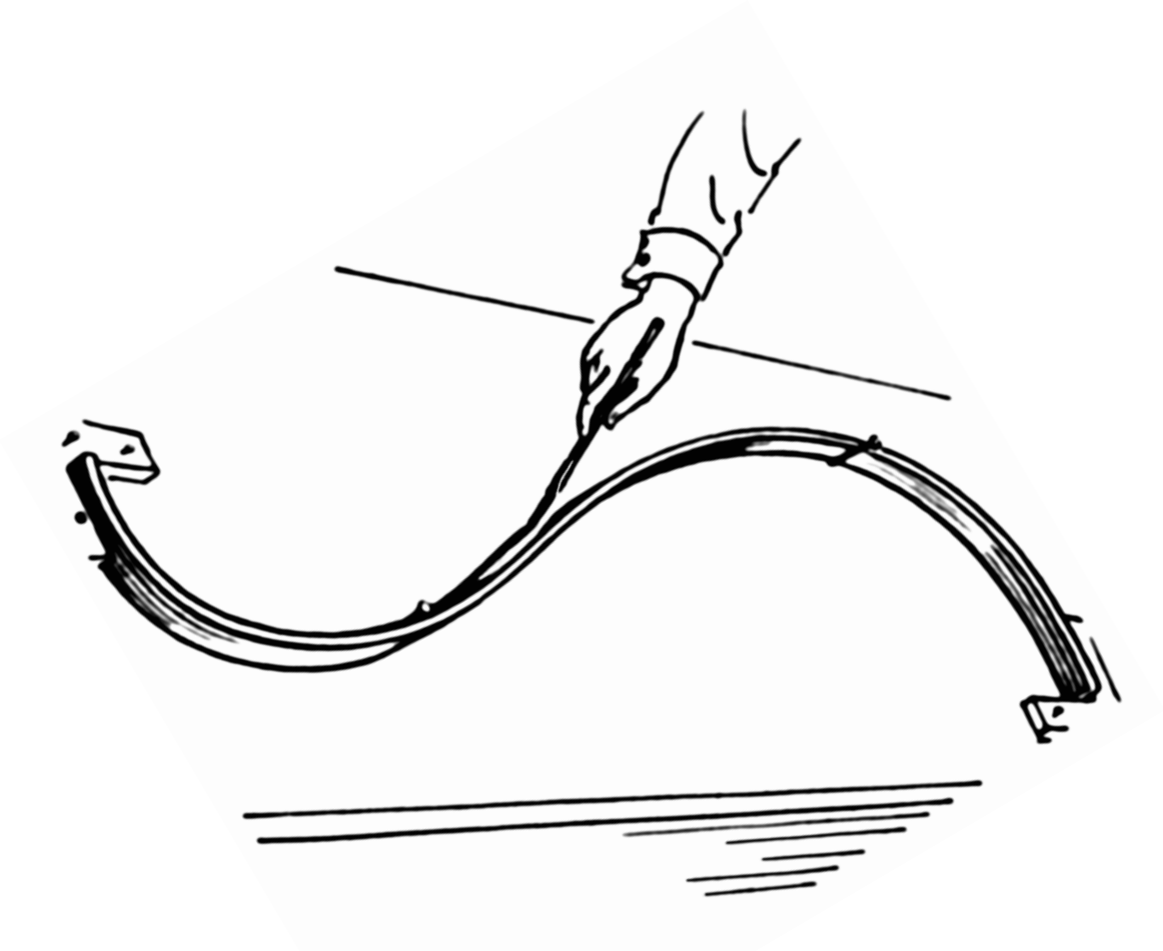
deep nets and splines

- Deep nets solve a **function approx** problem **hierarchically** using a very special family of **splines**



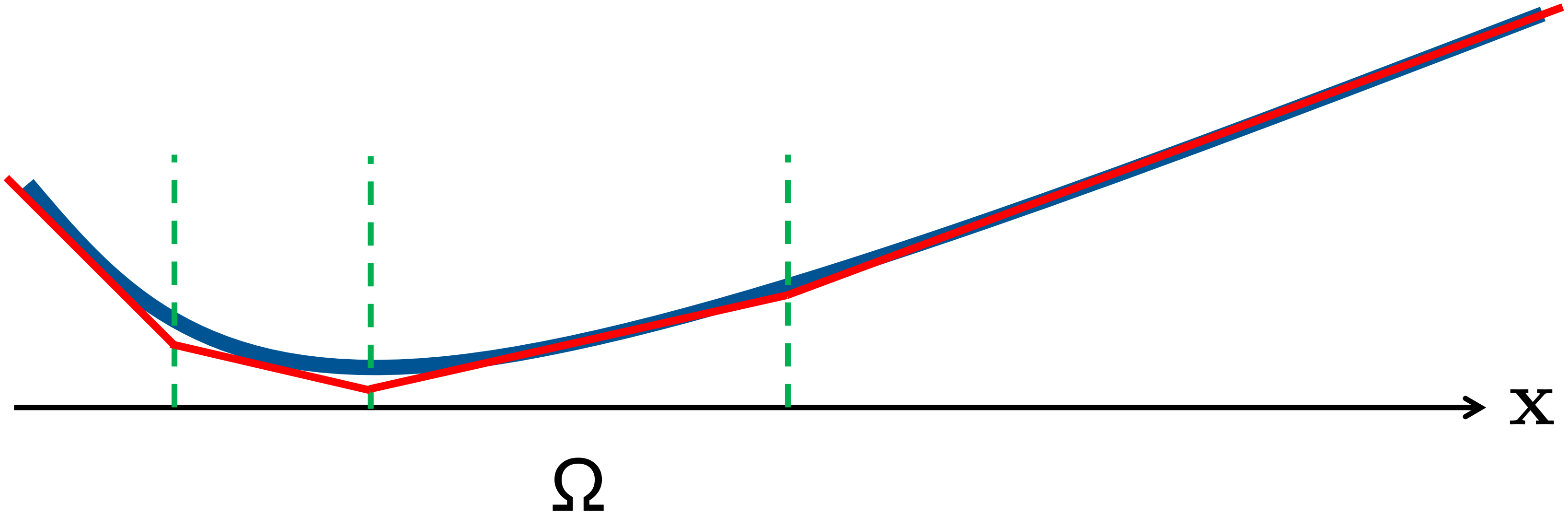
$$\widehat{\mathbf{y}} = f_\Theta(\mathbf{x}) = \left(f^{(L)}_{\theta^{(L)}} \circ \cdots \circ f^{(3)}_{\theta^{(3)}} \circ f^{(2)}_{\theta^{(2)}} \circ f^{(1)}_{\theta^{(1)}} \right)(\mathbf{x})$$

spline approximation



spline approximation

- A **spline** function approximation consists of
  - a **partition** Ω of the independent variable (input space)
  - a (simple) **local mapping** on each region of the partition (our focus: piecewise-affine mappings)



max-affine spline (MAS)

[Magnani & Boyd, 2009; Hannah & Dunson, 2013]

- Consider **piecewise-affine approximation** of a **convex function** over *R* regions
  - Affine functions: $a_r^\top \mathbf{x} + b_r, \quad r = 1, \ldots, R$

↑ ↑

fixed parameters

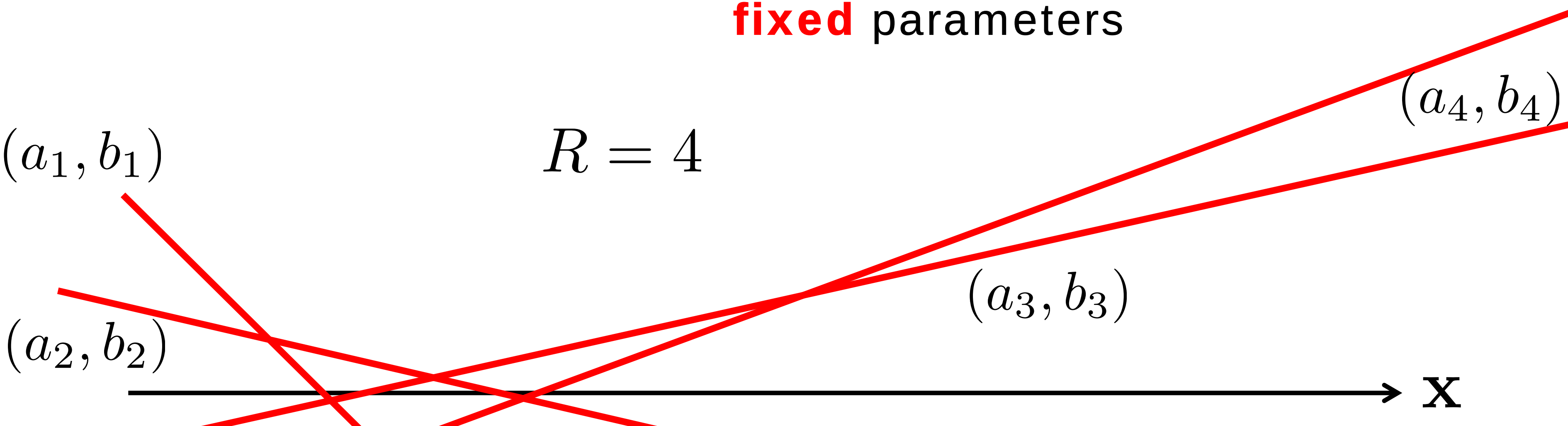
max-affine spline (MAS)

[Magnani & Boyd, 2009; Hannah & Dunson, 2013]

- Consider **piecewise-affine approximation** of a **convex function** over *R* regions
  - Affine functions: $a_r^\mathsf{T}\mathbf{x} + b_r, \quad r = 1, \ldots, R$

fixed parameters

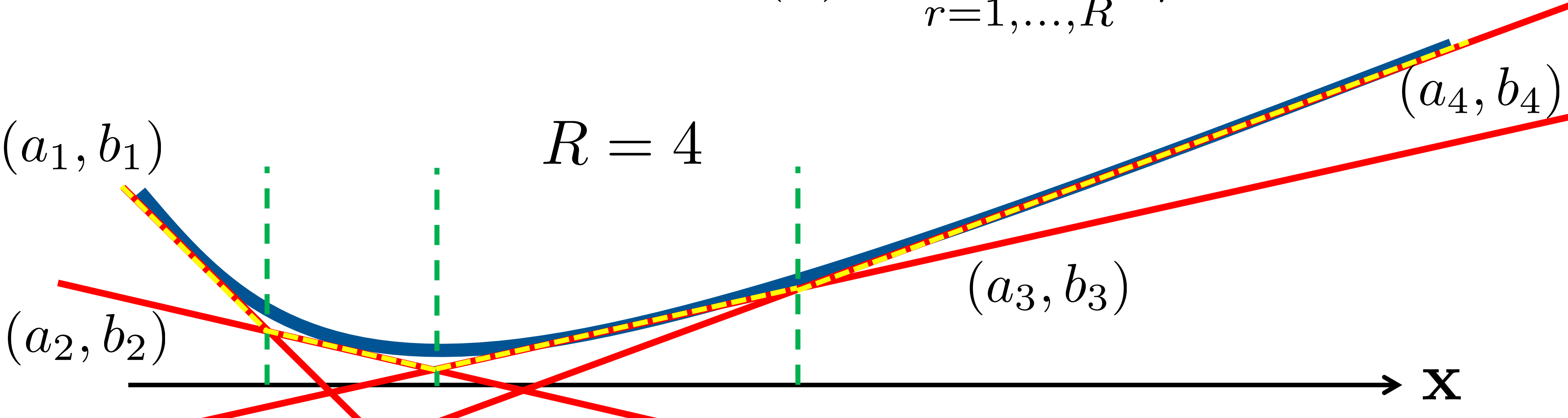
$R = 4$



max-affine spline (MAS)

[Magnani & Boyd, 2009; Hannah & Dunson, 2013]

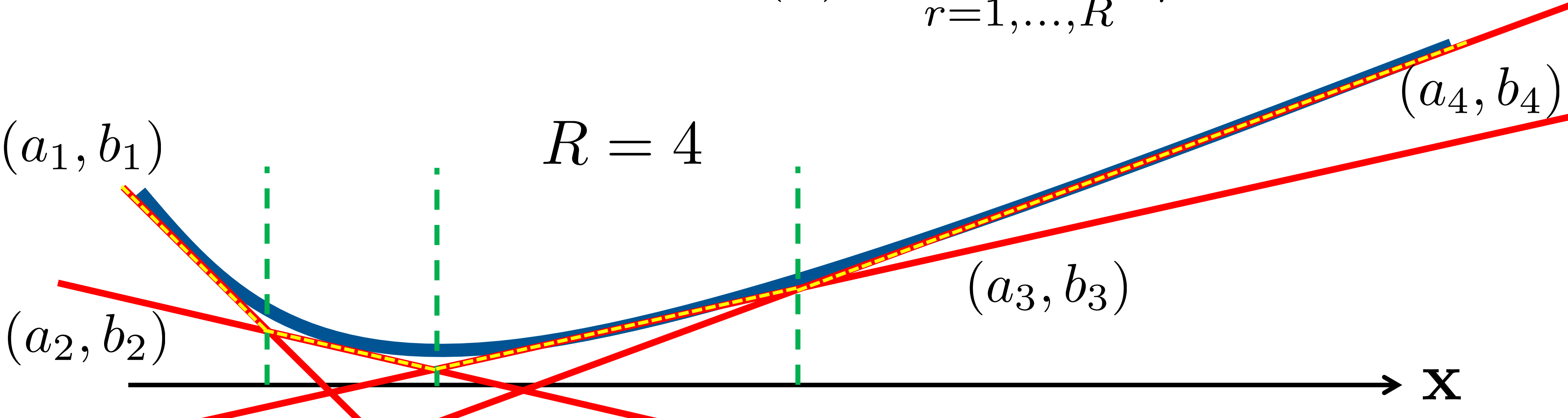
- Consider **piecewise-affine approximation** of a **convex function** over *R* regions
  - Affine functions: $a_r^\top \mathbf{x} + b_r, \quad r = 1, \ldots, R$
  - Convex approximation: $z(\mathbf{x}) = \max_{r=1,\ldots,R} a_r^\top \mathbf{x} + b_r$



max-affine spline (MAS)

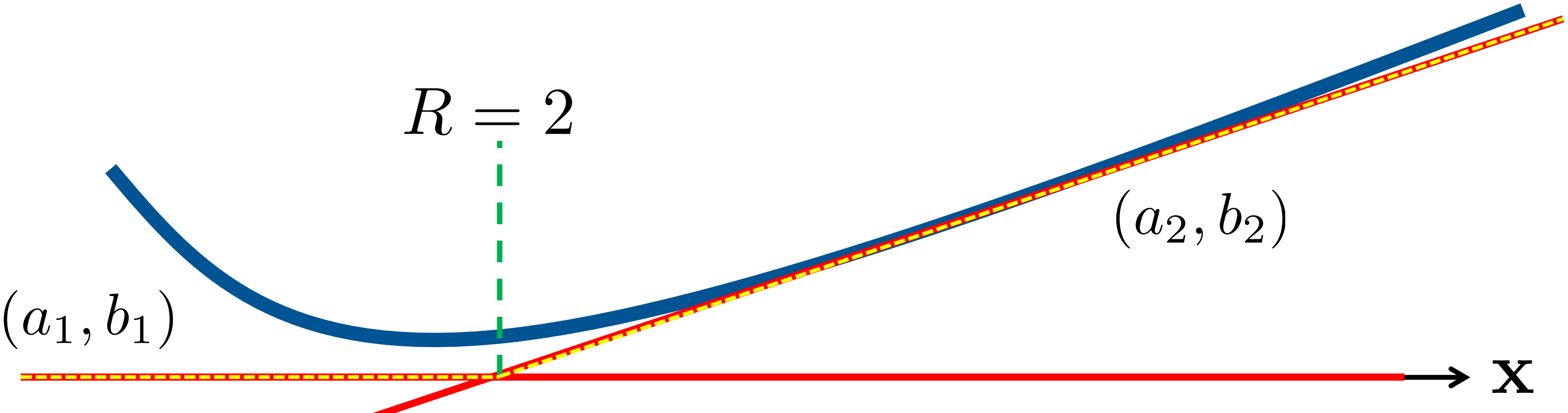
[Magnani & Boyd, 2009; Hannah & Dunson, 2013]

- **Key:** Any set of affine parameters $(a_r, b_r),\ r = 1, \ldots, R$ **implicitly** determines a spline **partition**
  - Affine functions: $a_r^\mathsf{T}\mathbf{x} + b_r, \quad r = 1, \ldots, R$
  - Convex approximation: $z(\mathbf{x}) = \max_{r=1,\ldots,R} a_r^\mathsf{T}\mathbf{x} + b_r$



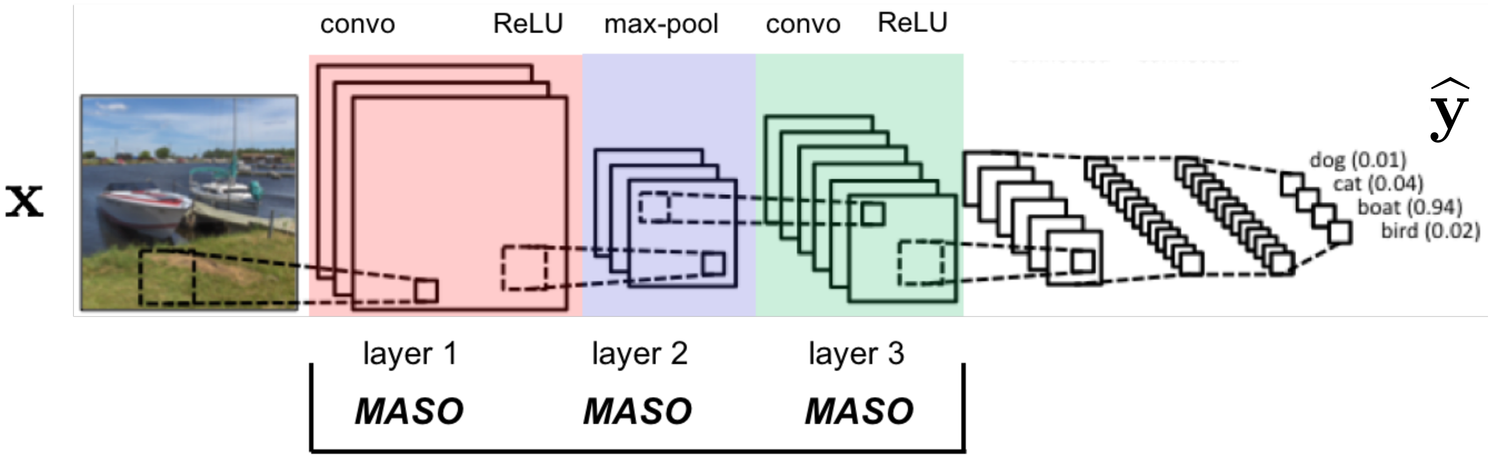
scale + bias | ReLU is a MAS

- Scale *x* by *a* + bias *b* | ReLU: $z(x) = \max(0, ax + b)$
  - Affine functions: $(a_1, b_1) = (0, 0),\ (a_2, b_2) = (a, b)$
  - Convex approximation: $z(\mathbf{x}) = \max_{r=1,2} a_r^\top \mathbf{x} + b_r$



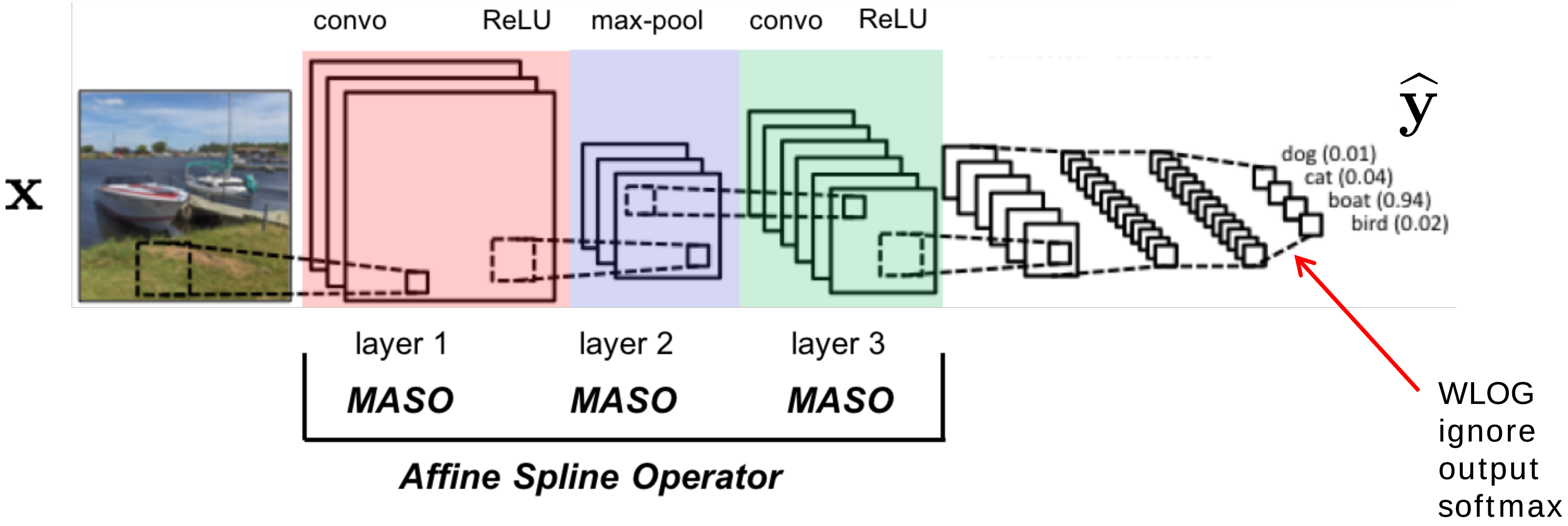
theorems

- Standard deep net **layers** are **Max Affine Spline Operators**
  - fully connected, convo | (leaky) ReLU, abs value
  - max/mean/channel-pooling
  - **convex** wrt each output dimension, piecewise-affine operator



theorems

- Standard deep net **layers** are **MASOs**
  - **convex** wrt each output dimension, piecewise-affine operator

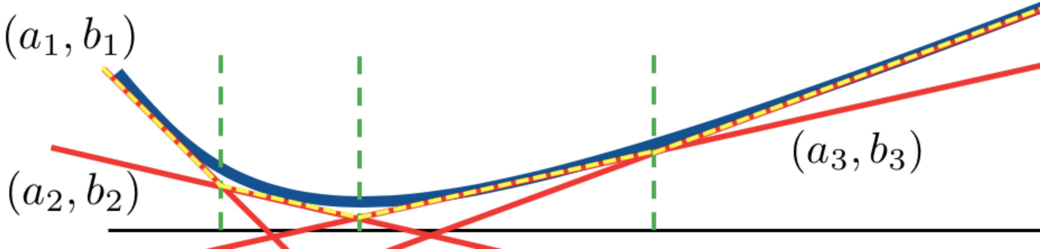


- A deep net is a **composition of MASOs**
  - **non-convex** piecewise-affine spline operator


And now, things really get **interesting**...

MASO spline partition

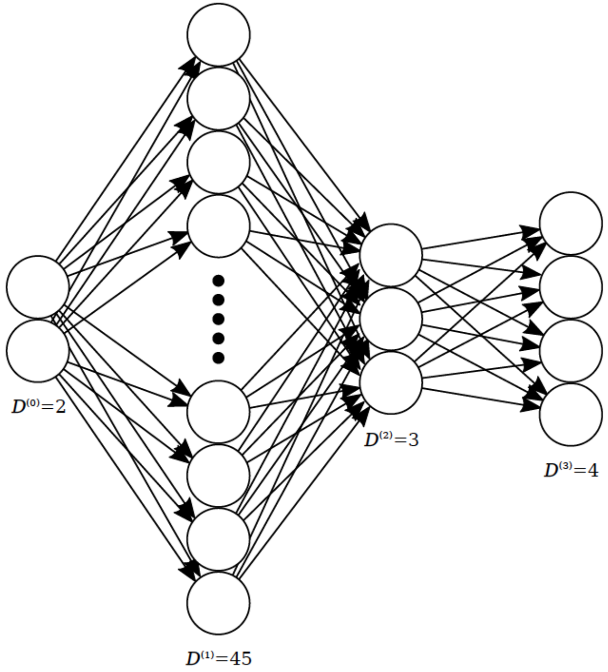
- The parameters of a **deep net layer** (MASO) induce a **partition** of the layer's input space with **convex regions**
- The **composition** of several layers progressively subdivides a **non-convex partition** of the deep net input space
- Partition **links** deep nets to
  - vector quantization (info theory)
  - *k*-means (statistics)
  - Voronoi tiling (geometry)



MASO spline partition

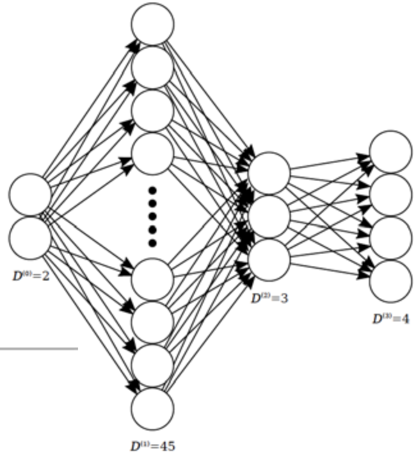
- The L layer-partitions of an L-layer deep net combine to form the **global input signal space partition**
  - affine spline operator
  - **non-convex regions**

- Toy example: **3-layer "deep net"**
  - Input $\boldsymbol{x}$: 2-D (4 classes)
  - Fully connected | ReLU (45-D output)
  - Fully connected | ReLU (3-D output)
  - Fully connected | (softmax) (4-D output)
  - Output $\boldsymbol{y}$: 4-D

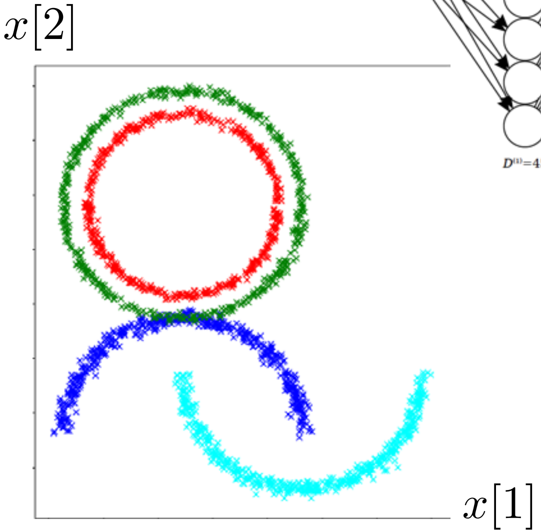


MASO spline partition

- The *L* layer-partitions of an *L*-layer deep net combine to form the **global input signal space partition**
  - affine spline operator
  - **non-convex regions**



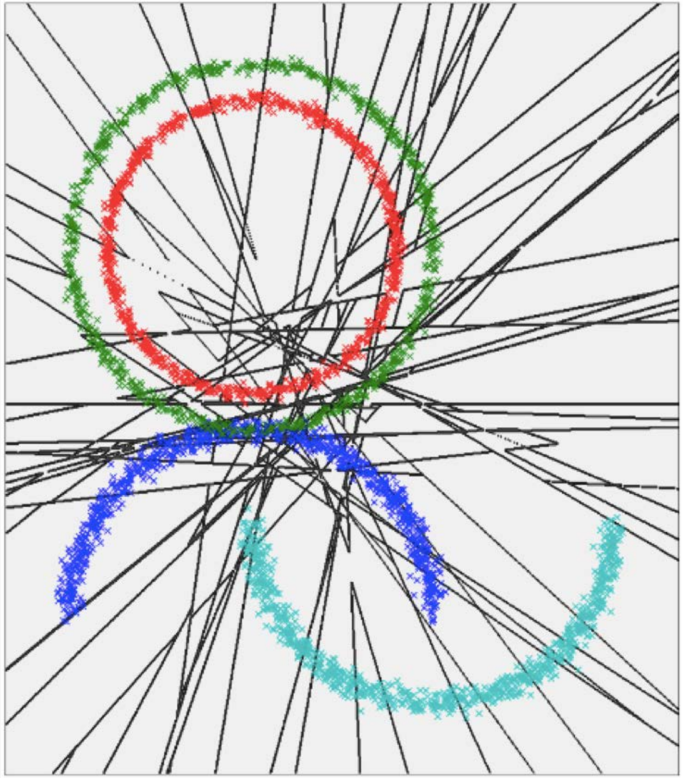
- Toy example: 3-layer "deep net"
  - **Input *x*: 2-D (4 classes)**
  - Fully connected | ReLU (45-D output)
  - Fully connected | ReLU (3-D output)
  - Fully connected | (softmax) (4-D output)
  - Output ***y***: 4-D



MASO spline partition

- Toy example: 3-layer "deep net"
  - Input ***x***: 2-D (4 classes)
  - **Fully connected | ReLU (45-D output)**
  - Fully connected | ReLU (3-D output)
  - Fully connected | (softmax) (4-D output)
  - Output ***y***: 4-D

- VQ partition of **layer 1** depicted in the input space
  - **convex** regions

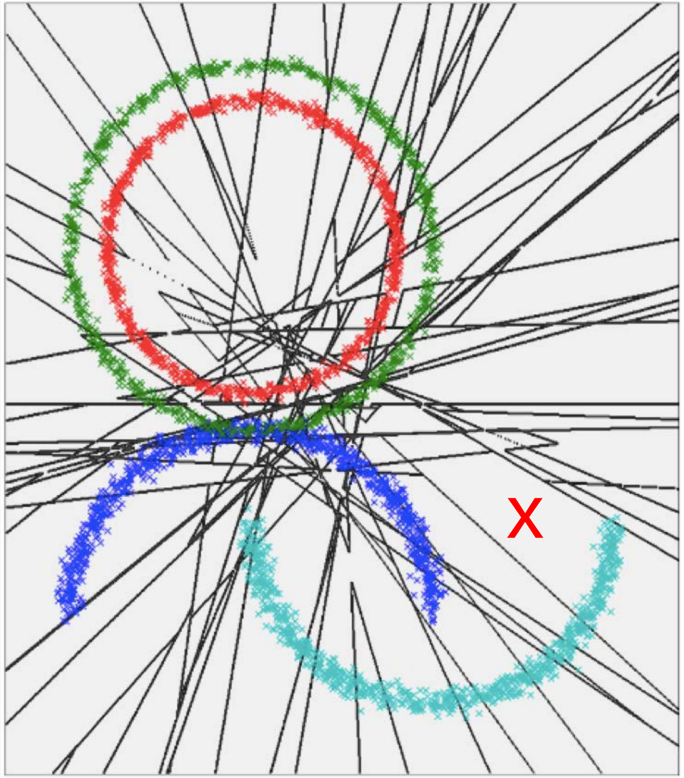


MASO spline partition

- Toy example: 3-layer "deep net"
  - Input ***x***: 2-D (4 classes)
  - **Fully connected | ReLU (45-D output)**
  - Fully connected | ReLU (3-D output)
  - Fully connected | (softmax) (4-D output)
  - Output ***y***: 4-D

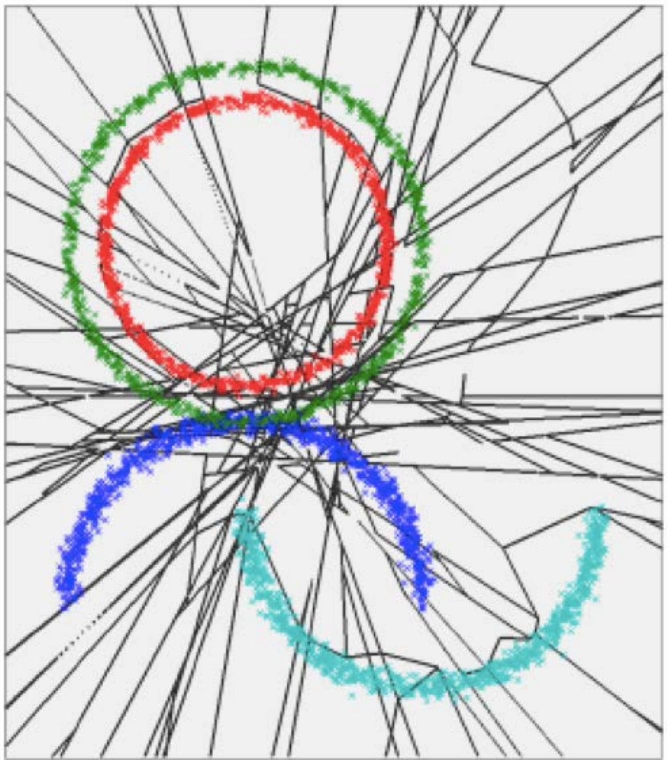
- Given the partition region $Q(\mathbf{x})$ containing $\mathbf{x}$ the layer **input/output mapping is affine**

$$\mathbf{z}(\mathbf{x}) = \mathbf{A}_{Q(\mathbf{x})}\mathbf{x} + \mathbf{b}_{Q(\mathbf{x})}$$



MASO spline partition

- Toy example: 3-layer "deep net"
  - Input ***x***: 2-D (4 classes)
  - **Fully connected | ReLU (45-D output)**
  - **Fully connected | ReLU (3-D output)**
  - Fully connected | (softmax) (4-D output)
  - Output ***y***: 4-D
- VQ partition of **layers 1 & 2** depicted in the input space
  - **non-convex** regions

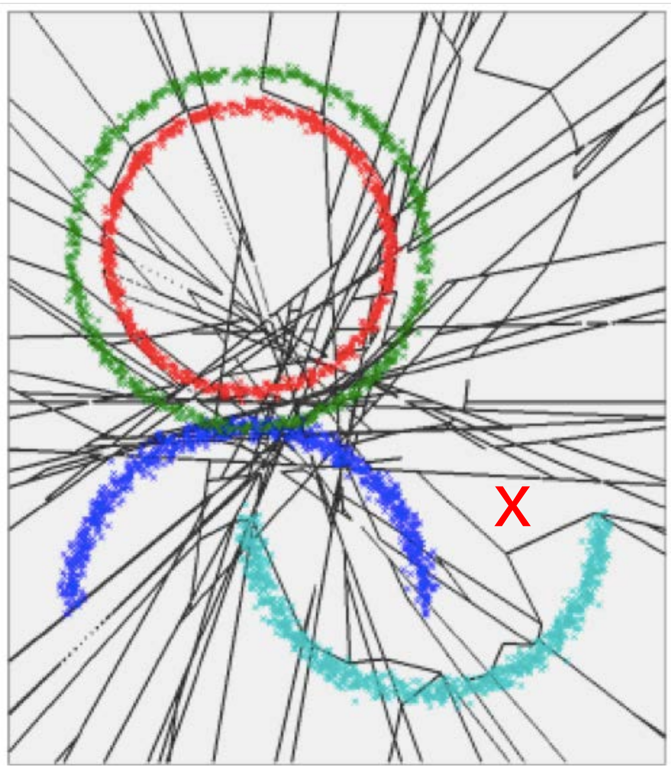


MASO spline partition

- Toy example: 3-layer "deep net"
  - Input $\boldsymbol{x}$: 2-D (4 classes)
  - **Fully connected | ReLU (45-D output)**
  - **Fully connected | ReLU (3-D output)**
  - Fully connected | (softmax) (4-D output)
  - Output $\boldsymbol{y}$: 4-D

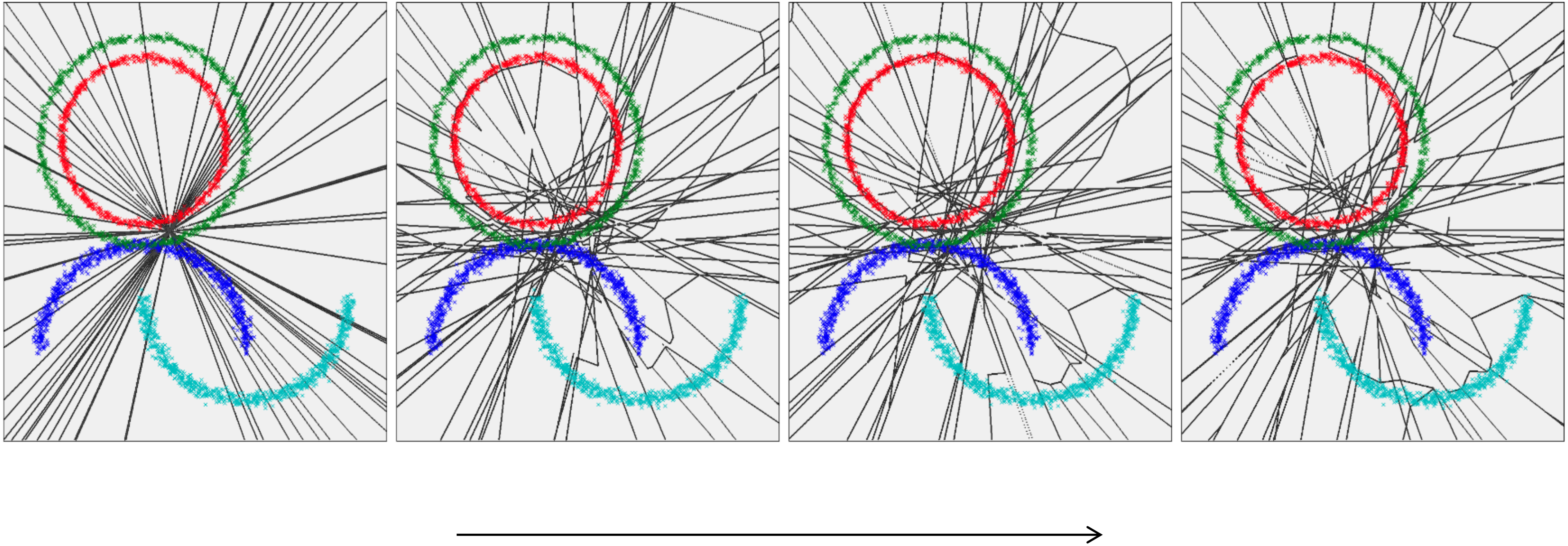
- Given the partition region $Q(\mathbf{x})$ containing $\mathbf{x}$ the two-layer **input/output mapping is affine**

$$\mathbf{z}(\mathbf{x}) = \mathbf{A}_{Q(\mathbf{x})}\mathbf{x} + \mathbf{b}_{Q(\mathbf{x})}$$



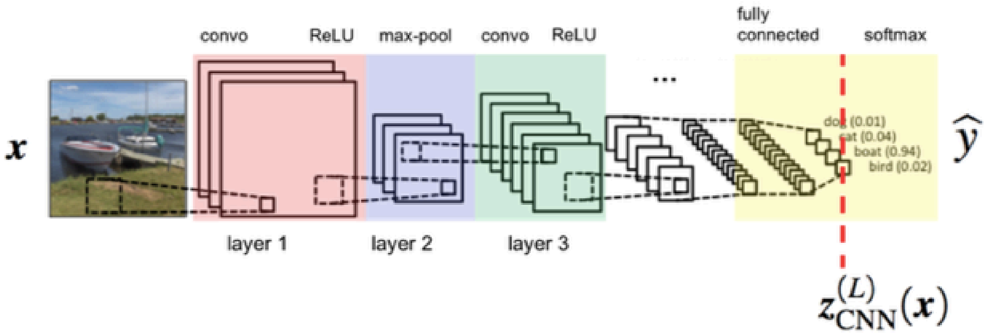
learning

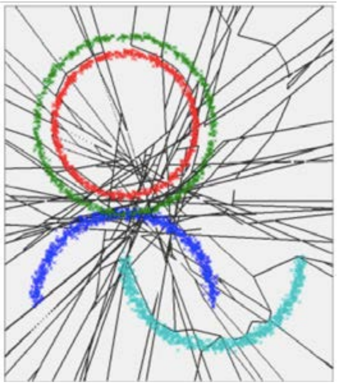
layers 1 & 2



learning epochs (time)

local affine mapping – CNN





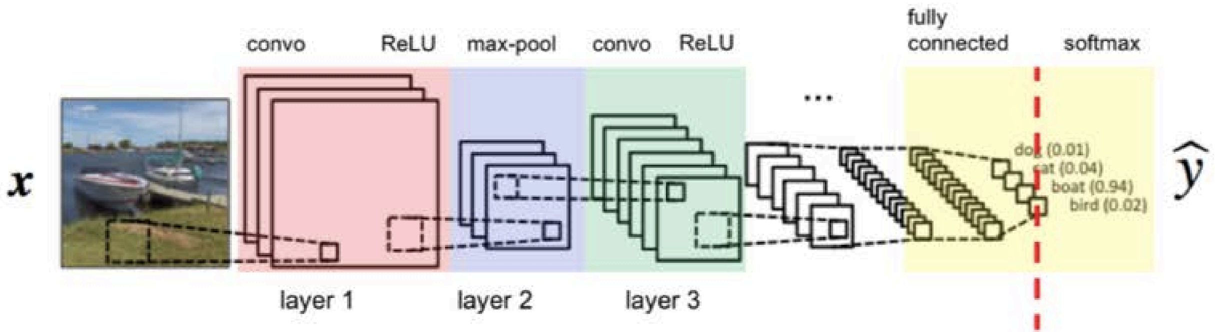
Fixed, different

$\mathbf{A}_{Q(\mathbf{x})}$, $\mathbf{b}_{Q(\mathbf{x})}$

in each partition region

$$z^{(L)}_{\text{CNN}}(\boldsymbol{x}) = \underbrace{\left(W^{(L)} \prod_{\ell=L-1}^{1} A_{\rho}^{(\ell)}[\boldsymbol{x}] A_{\sigma}^{(\ell)}[\boldsymbol{x}] C^{(\ell)}\right)}_{A_{Q(\boldsymbol{x})}\boldsymbol{x}} + \underbrace{W^{(L)} \sum_{\ell=1}^{L-1} \left(\prod_{j=L-1}^{\ell+1} A_{\rho}^{(j)}[\boldsymbol{x}] A_{\sigma}^{(j)}[\boldsymbol{x}] C^{(j)}\right)\left(A_{\rho}^{(\ell)}[\boldsymbol{x}] A_{\sigma}^{(\ell)}[\boldsymbol{x}] b_{C}^{(\ell)}\right) + b_{W^{(L)}}}_{b_{Q(\boldsymbol{x})}}$$

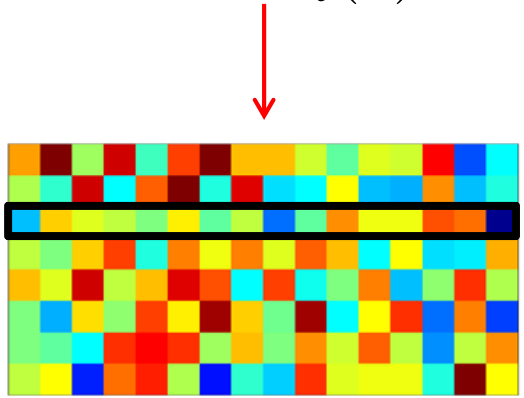
deep nets are matched filterbanks



$$\mathbf{z}^{(L)}(\mathbf{x}) = \mathbf{A}_{Q(\mathbf{x})}\mathbf{x} + \mathbf{b}_{Q(\mathbf{x})}$$

$\mathbf{z}^{(L)}(\mathbf{x})$

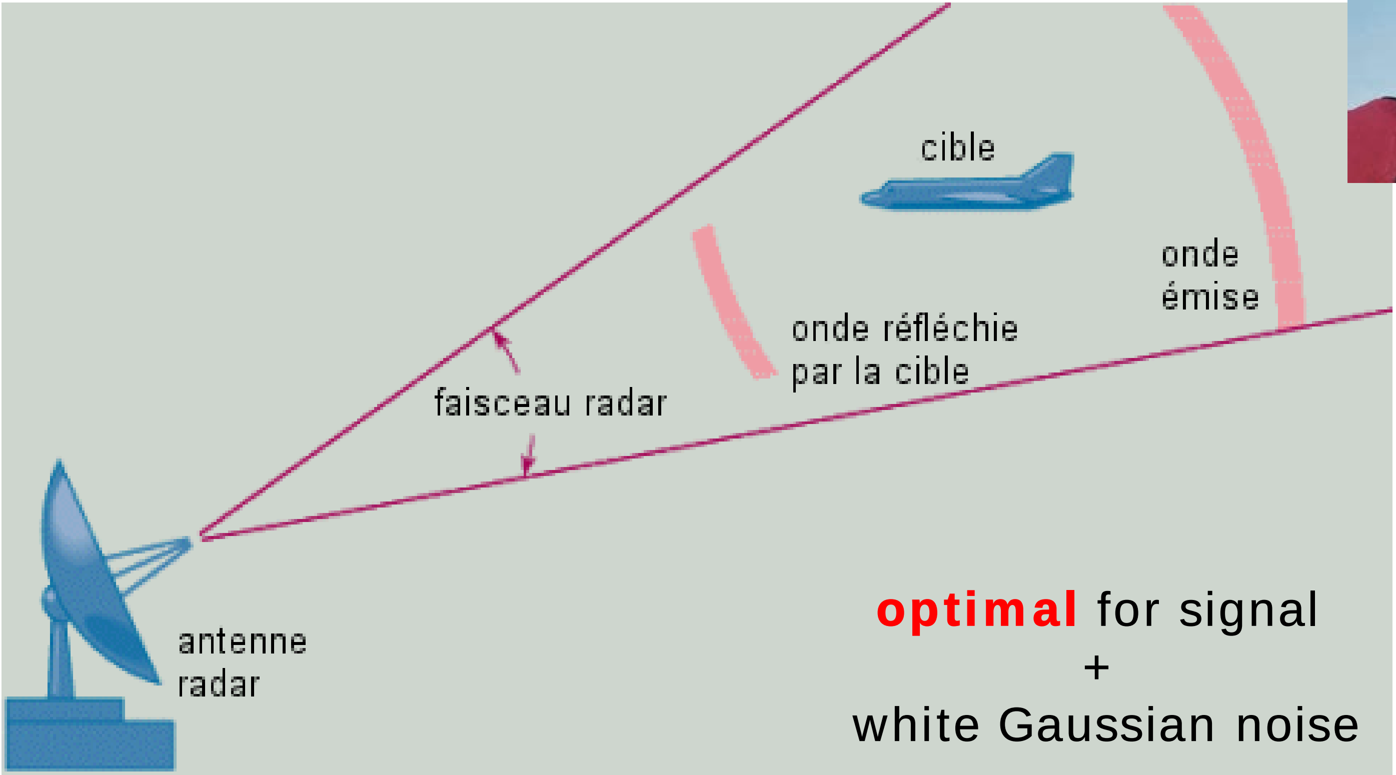
c



- Row c of $\mathbf{A}_{Q(\mathbf{x})}$ is a vectorized signal/image corresponding to class c
- Entry c of deep net output = inner product between row c and signal
- For classification, select largest output; **matched filter!**

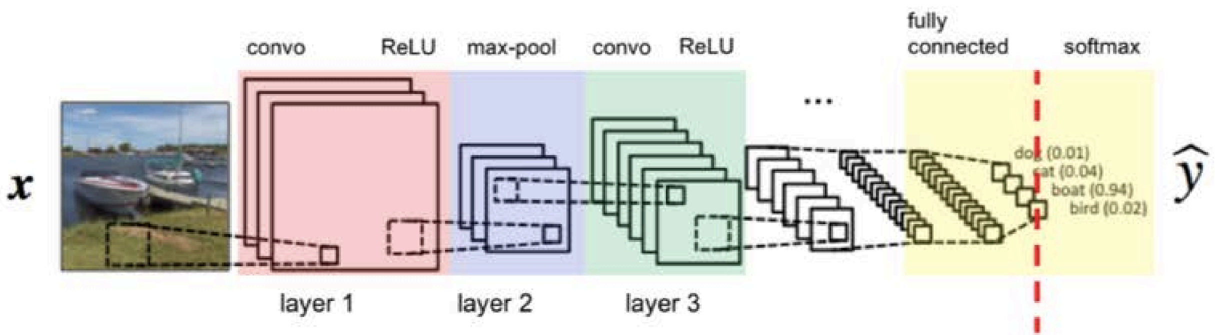
matched filter

- Aka **"sliding window cross-correlation"**



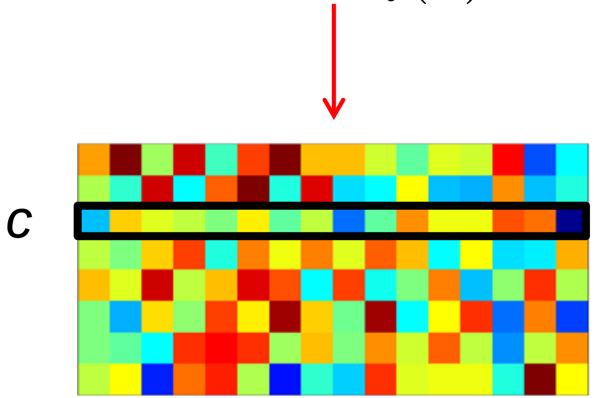
optimal for signal
+
white Gaussian noise

deep nets are matched filterbanks

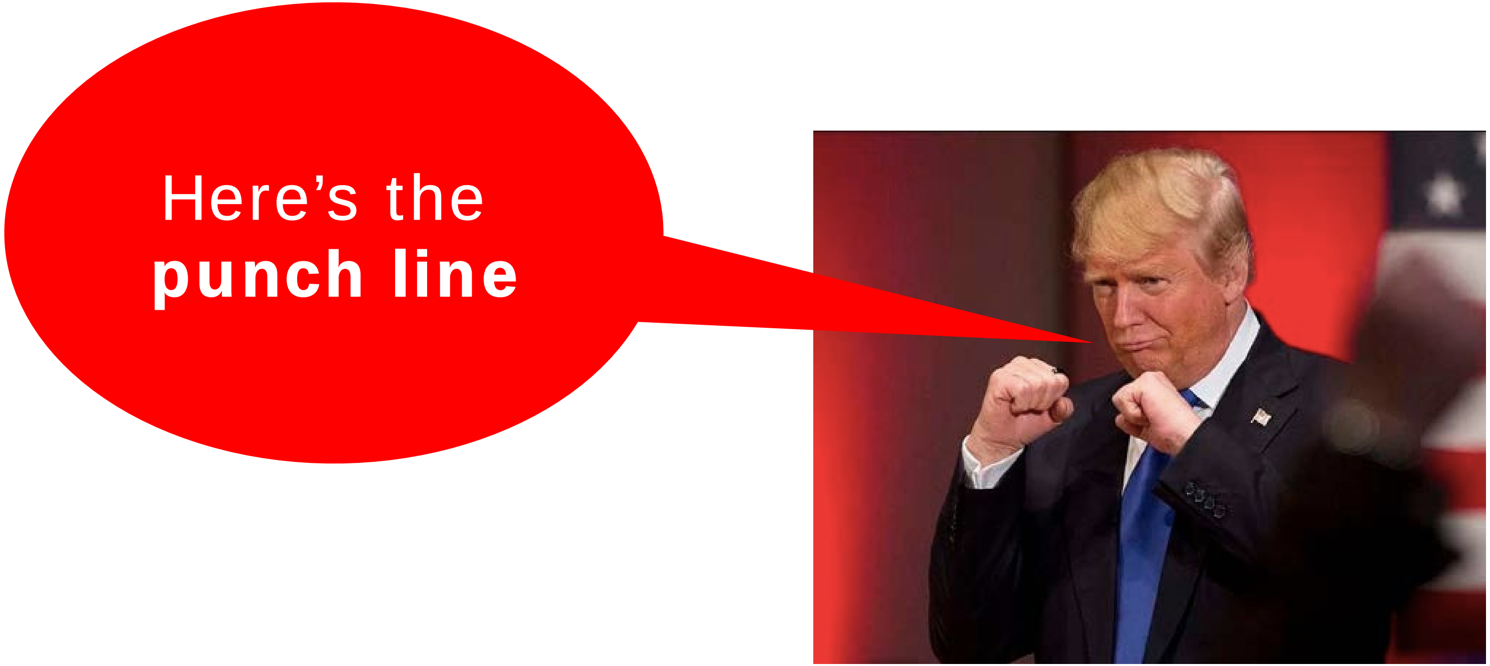


$$\mathbf{z}^{(L)}(\mathbf{x}) = \mathbf{A}_{Q(\mathbf{x})}\mathbf{x} + \mathbf{b}_{Q(\mathbf{x})}$$

$\mathbf{z}^{(L)}(\mathbf{x})$



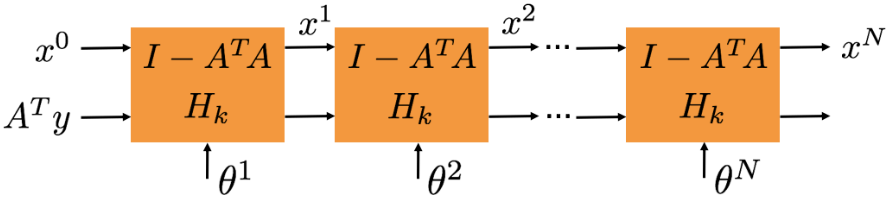
- Row c of $\mathbf{A}_{Q(\mathbf{x})}$ is a vectorized signal/image corresponding to class c
- Entry c of deep net output = inner product between row c and signal
- For classification, select largest output; **matched filter!**



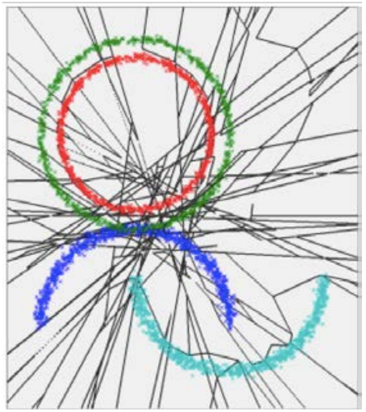
- **(Max-Affine) Splines** provide a solid mathematical foundation for a **theory of deep learning** based on familiar signal processing tools
  (like matched filtering, …)

summary

- A wide range of inference problems can be solved using **deep nets**, including **signal measurement & recovery**



- A wide range of deep nets solve function approximation problems using a **composition of max-affine spline operators (MASOs)**
  - links to **vector quantization**, *k*-means, Voronoi tiling

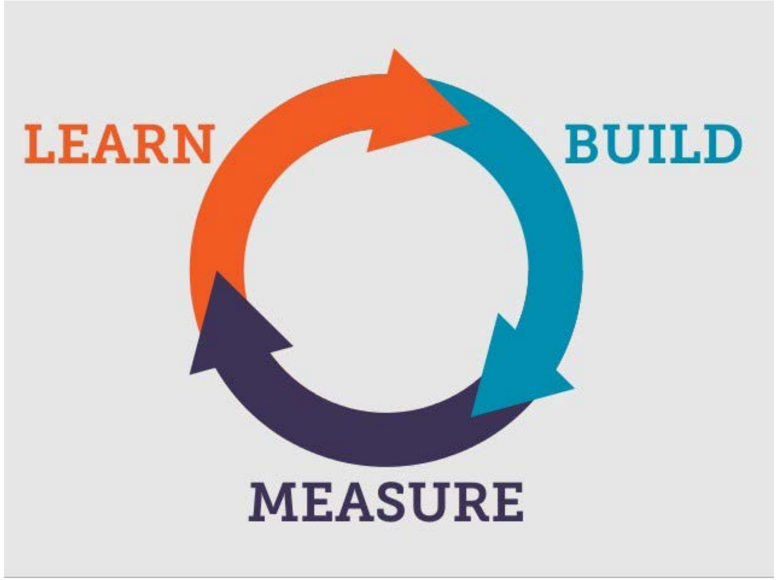


- Input/output deep net mapping is a **VQ-dependent affine transform**
- Deep nets are (learned) **matched filterbanks**

the road ahead

- Still **early days** for bringing **models** and **data** into concert to tackle wicked hard inference problems

- New theory: **Splines**
- New apps: **Signal recovery**



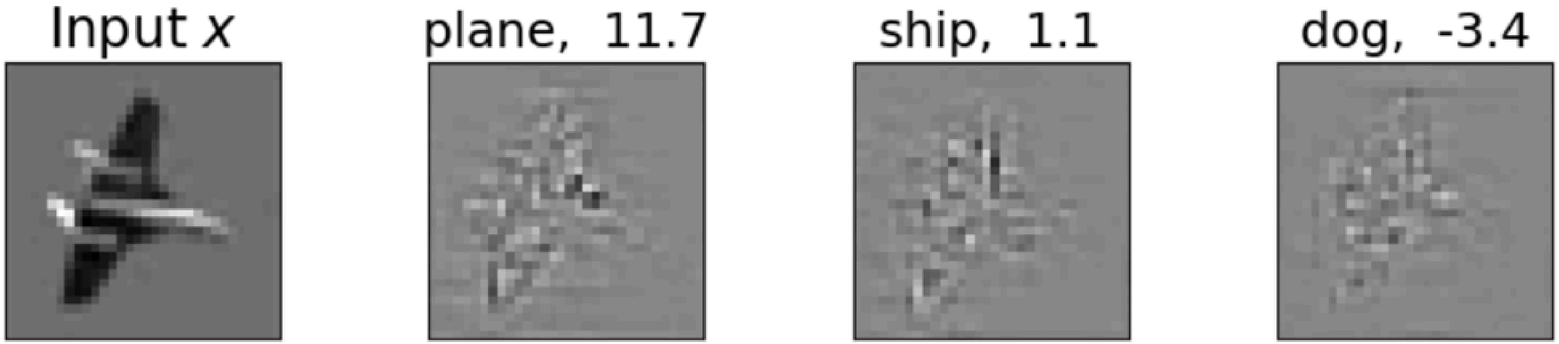
RICE UNIVERSITY DSP

dsp.rice.edu

deep nets are matched filterbanks

Result Row c of $A_{Q(\boldsymbol{x})}$ is a **matched filter** for class c that is applied to $\boldsymbol{x}$; largest inner product wins

Visualization for CIFAR10: Row of $A_{\text{net}}[\boldsymbol{x}]$, inner product with $\boldsymbol{x}$



(Converted to black & white for ease of visualization)

Matched filter can be interpreted as being applied **hierarchically** thru the layers

Link with **saliency maps** [Simonyan et al., 2013; Zeiler & Fergus, 2014]

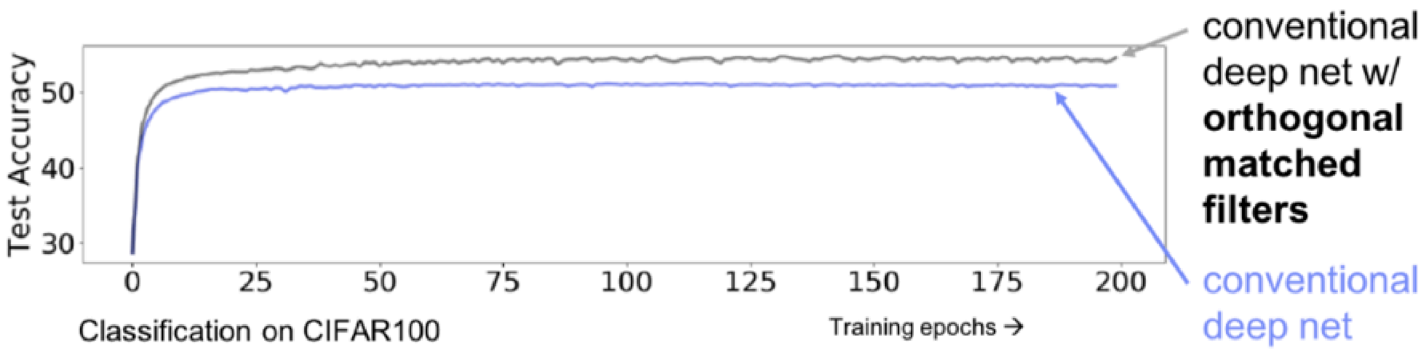
orthogonal deep nets

Matched filter classifier is optimal only for signal + white Gaussian noise (idealized)

For more general noise/nuisance models, useful to **orthogonalize** the matched filters

[Eldar and Oppenheim, 2001]

Result Easy to do with any deep net thanks to the affine transformation formula; simply add to the cost function a **penalty on the off-diagonal entries** of $W^{(L)}(W^{(L)})^T$



Bonus: Reduced overfitting