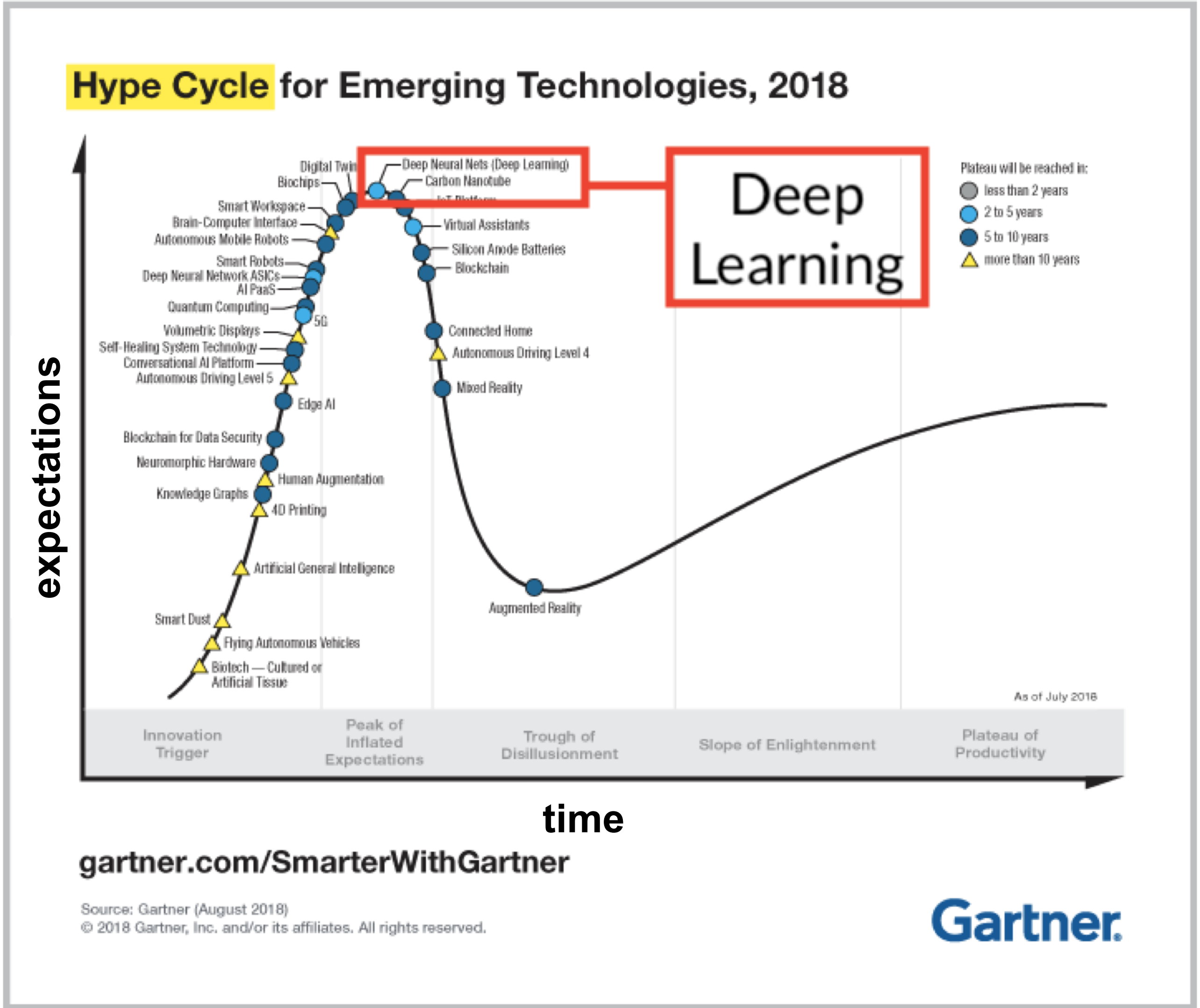
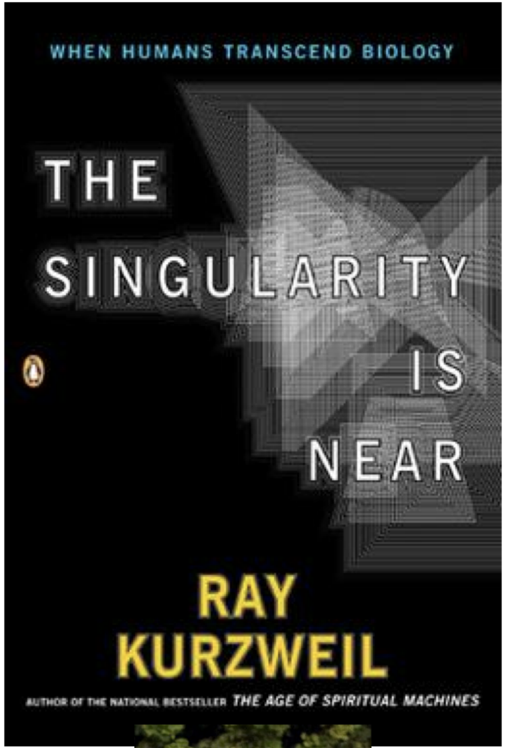
Mad Max:

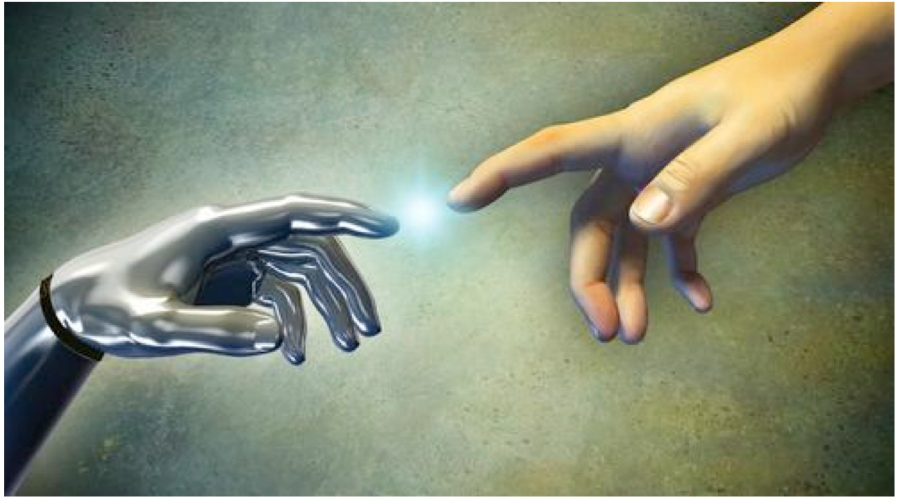
Affine Spline Insights into Deep Learning

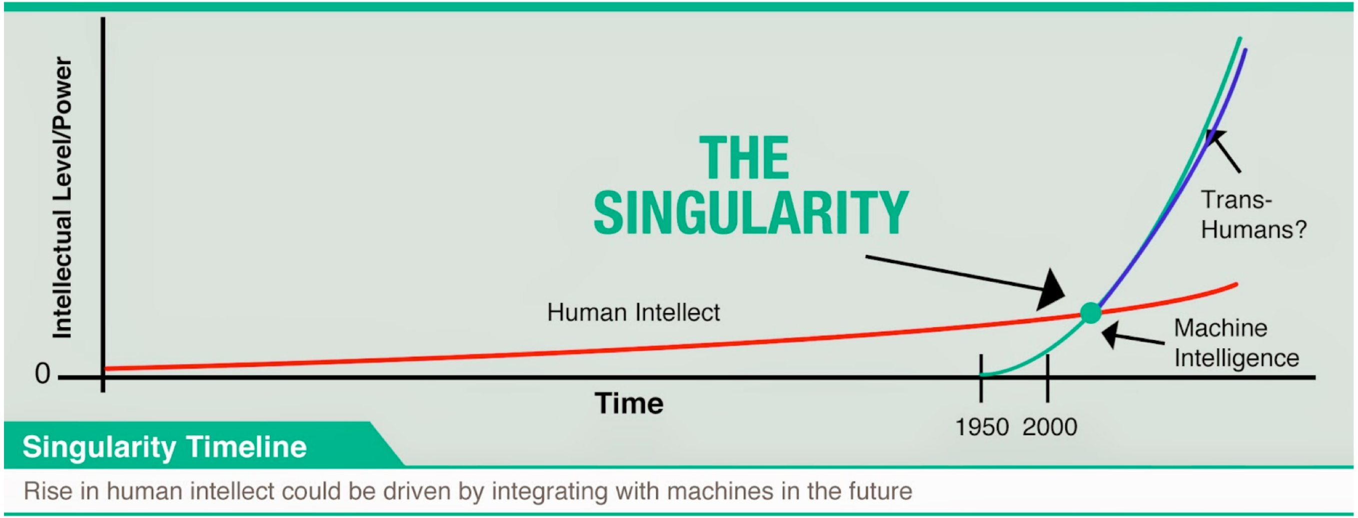
Richard Baraniuk

RICE UNIVERSITY


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


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




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

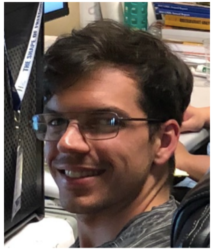

IWDRM

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 & B
"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)

prediction problem

- Unknown **function/operator** f mapping data to labels

$$\mathbf{y} = f(\mathbf{x})$$

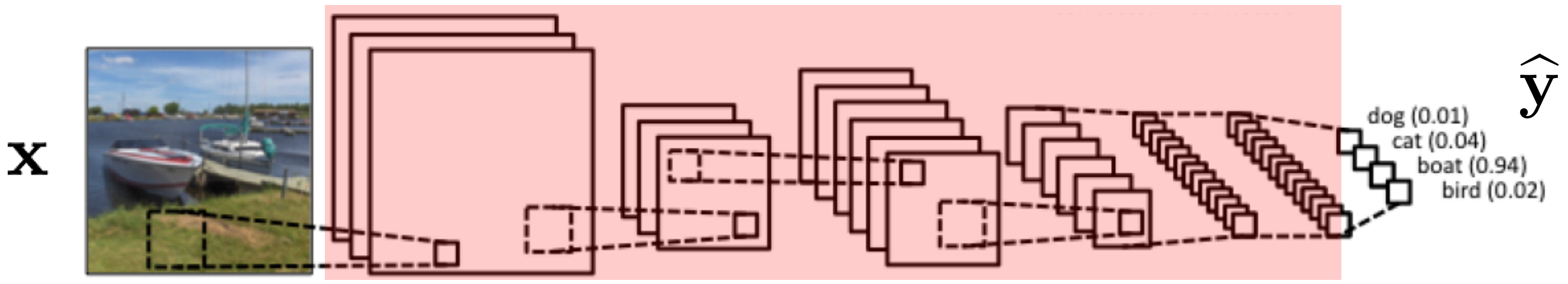
label data (signal, image, video, …)

- **Goal:** Learn an **approximation** to f using **training data**

$$\widehat{\mathbf{y}} = f_{\Theta}(\mathbf{x}) \qquad \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{n}$$

deep nets approximate

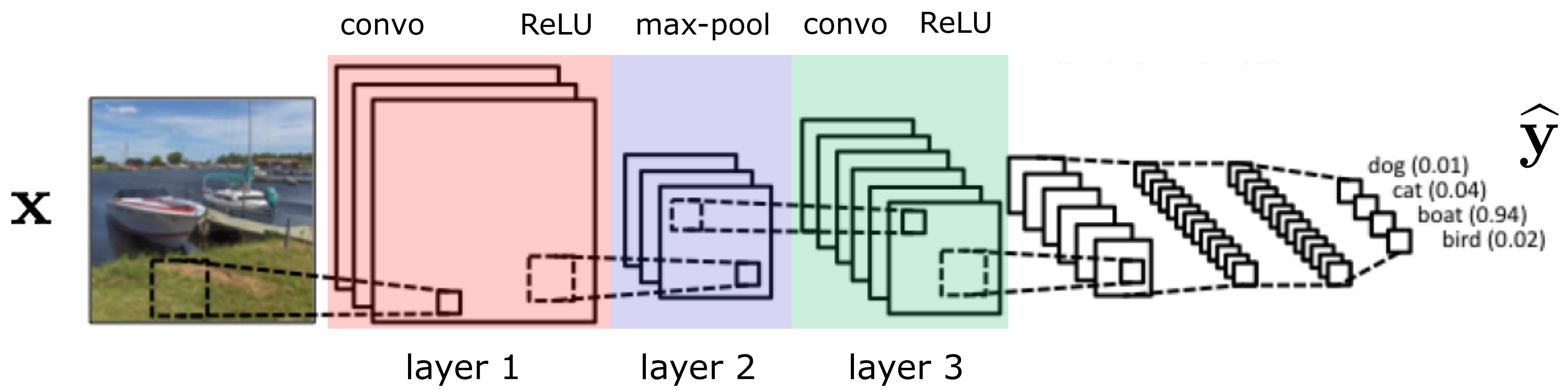
- Deep nets solve a **function approx** problem (black box)



$$\widehat{\mathbf{y}} = f_{\Theta}(\mathbf{x})$$

deep nets approximate

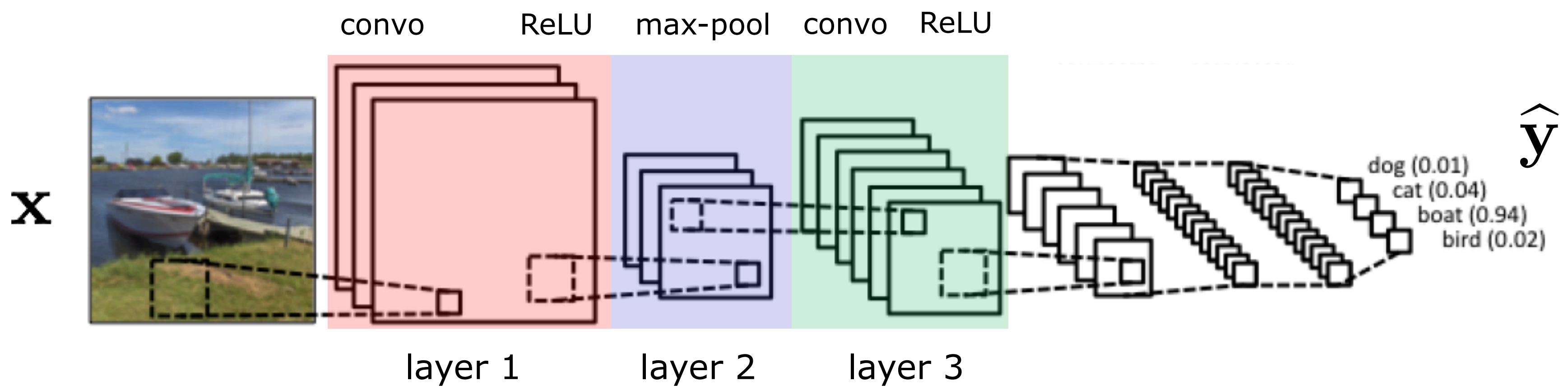
- Deep nets solve a **function approx** problem **hierarchically**



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

deep nets and splines

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



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

deep nets and splines

Piecewise convexity of artificial neural networks

Blaine Rister [a,*], Daniel L. Rubin [b]

[a] Stanford University, Department of Electrical Engineering, 1201 Welch Rd, Stanford, CA, 94305, USA
[b] Stanford University, Department of Radiology (Biomedical Informatics Research), 1201 Welch Rd Stanford, CA, 94305, USA

On the Number of Linear Regions of Deep Neural Networks

Guido Montúfar
Max Planck Institute for Mathematics in the Sciences
montufar@mis.mpg.de

Razvan Pascanu
Université de Montréal
pascanur@iro.umontreal.ca

Kyunghyun Cho
Université de Montréal
kyunghyun.cho@umontreal.ca

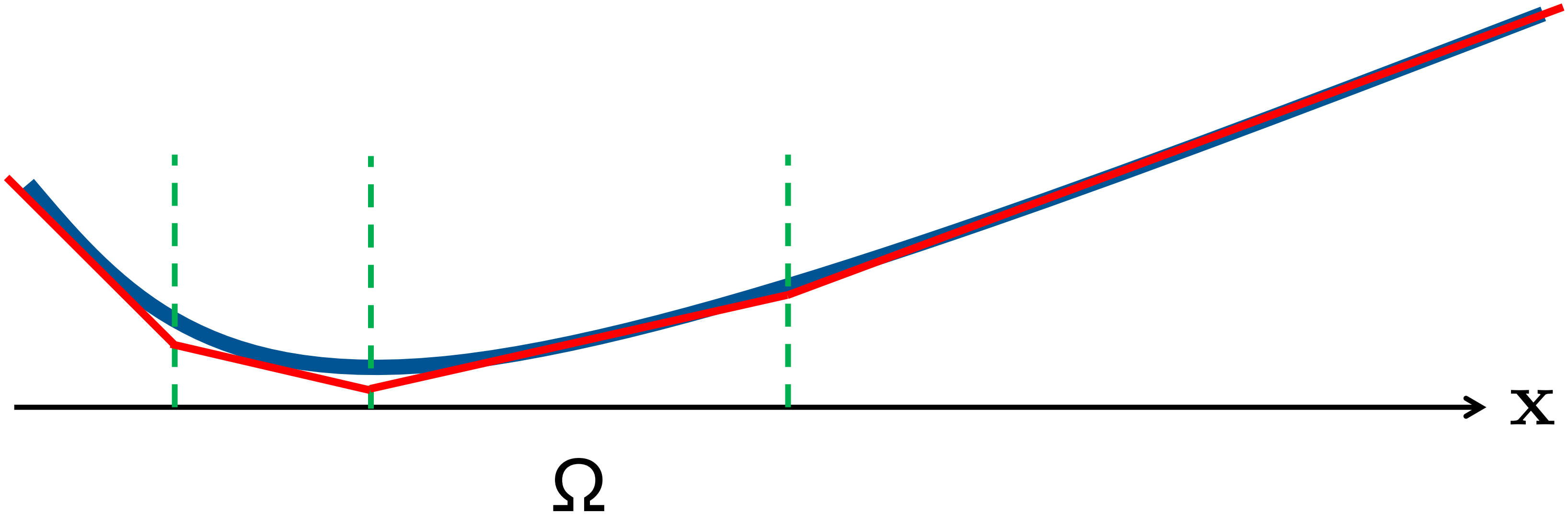
Yoshua Bengio
Université de Montréal, CIFAR Fellow
yoshua.bengio@umontreal.ca

A representer theorem for deep neural networks

Michael Unser

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)



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

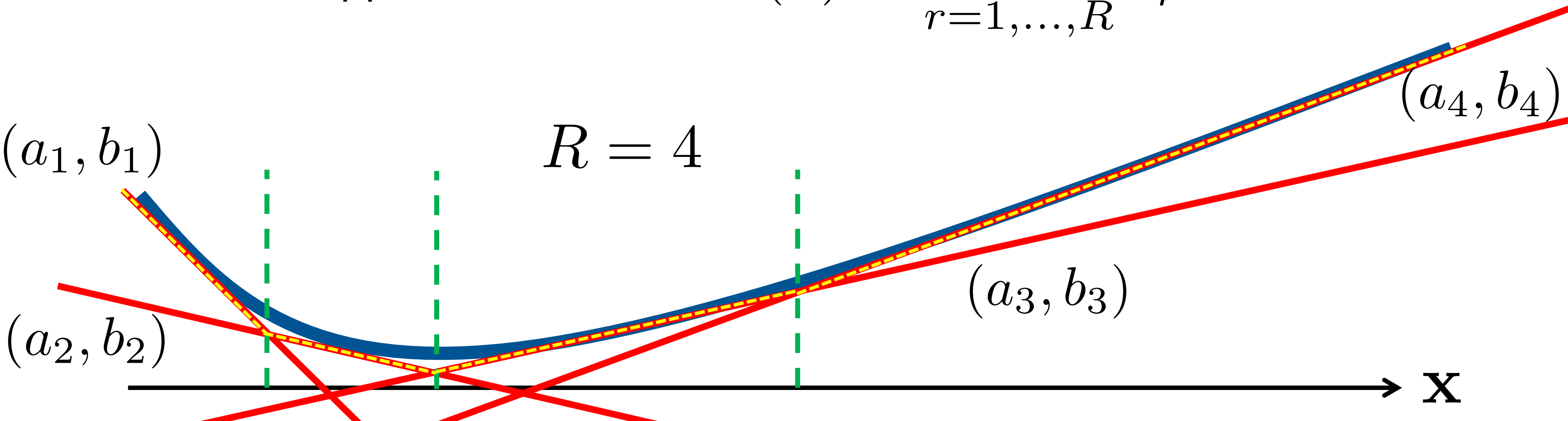
- **Powerful splines**
  - free, unconstrained partition Ω (ex: **“free-knot” splines**)
  - jointly optimize both the partition and local mappings (highly nonlinear, computationally intractable)

- **Easy splines**
  - fixed partition (ex: uniform grid, dyadic grid)
  - need only optimize the local mappings

max-affine spline (MAS)

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

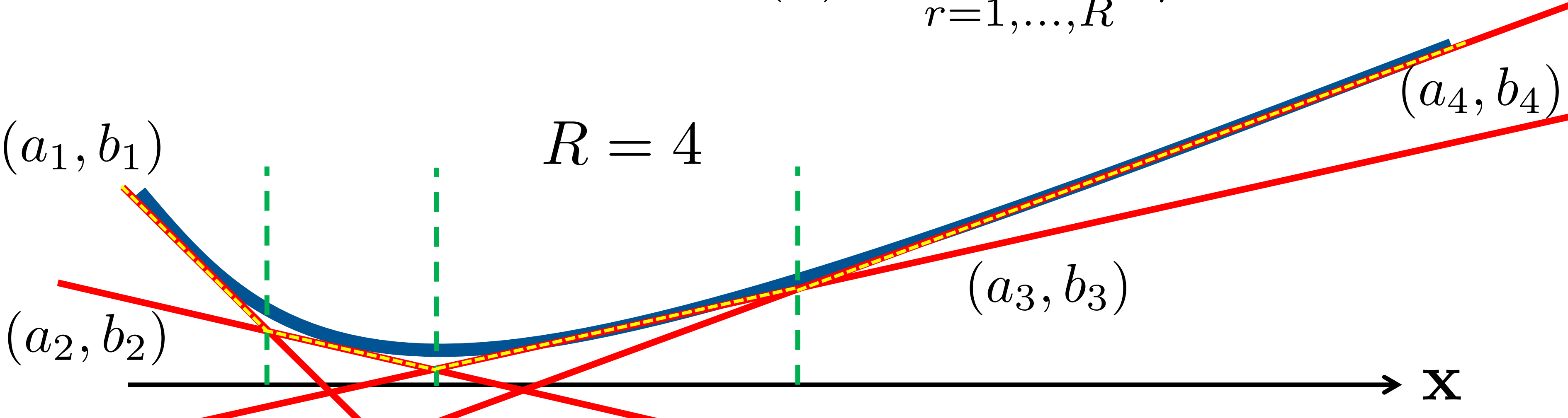
- Consider **piecewise-affine approximation** of a **convex function** over *R* regions
  - Affine functions: $$a_r^\mathsf{T}\mathbf{x} + b_r, \quad r = 1, \ldots, R$$
  - Convex approximation: $$z(\mathbf{x}) = \max_{r=1,\ldots,R} a_r^\mathsf{T}\mathbf{x} + b_r$$



max-affine spline (MAS)

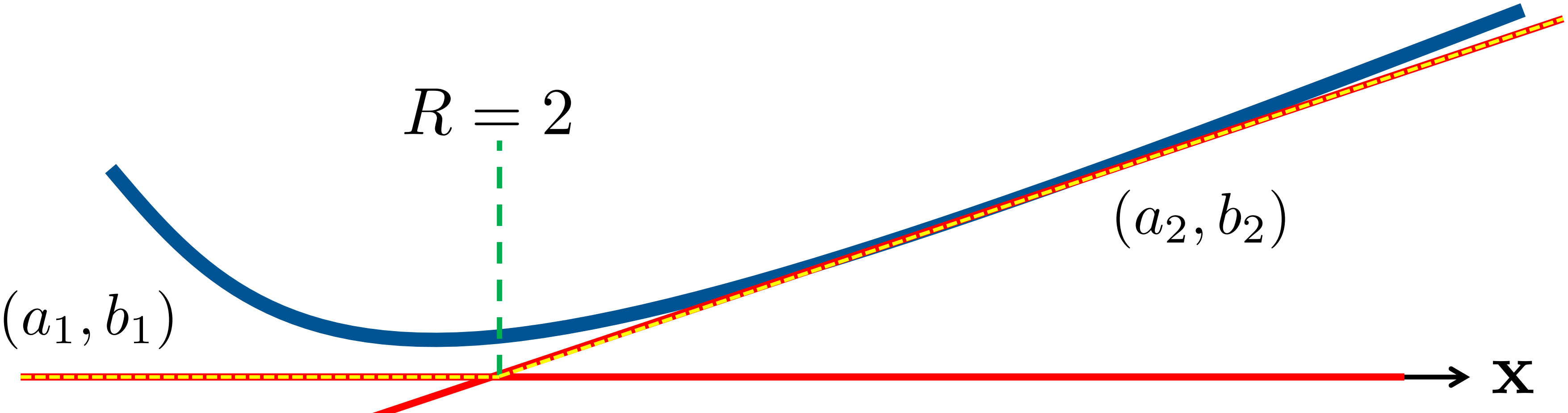
[Magnani & Boyd, 2009; Hannah & Dunson, 2013]

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



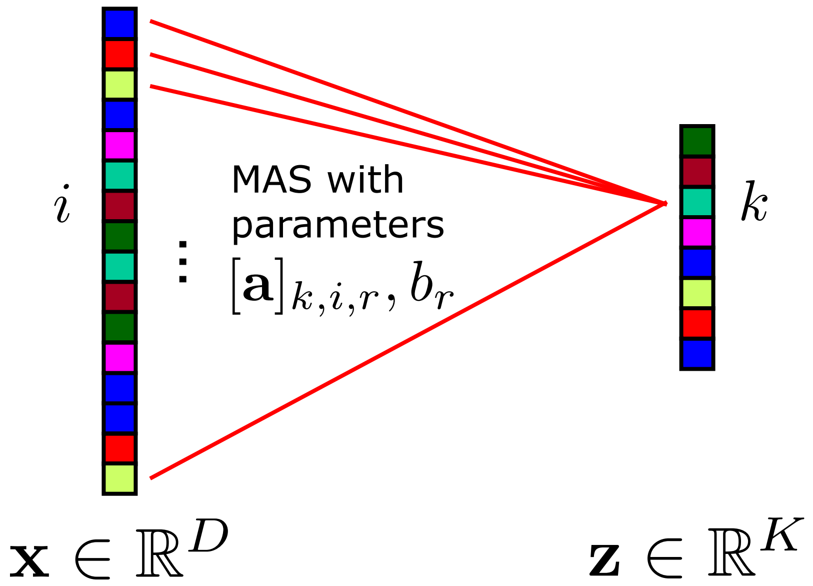
scale + bias | ReLU is a MAS

- 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: $z(\mathbf{x}) = \max_{r=1,2} a_r^\top \mathbf{x} + b_r$



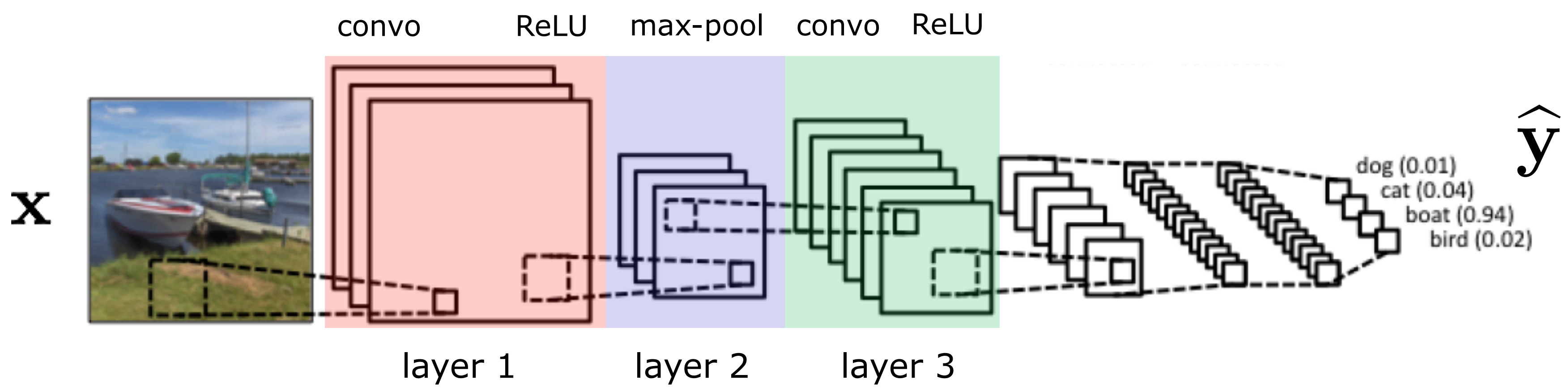
max-affine spline operator (MASO)

- **MAS** for $\mathbf{x} \in \mathbb{R}^D$ has affine parameters $\mathbf{a}_r \in \mathbb{R}^D, b_r \in \mathbb{R}$
- A **MASO** is simply a concatenation of *K* MASs



modern deep nets

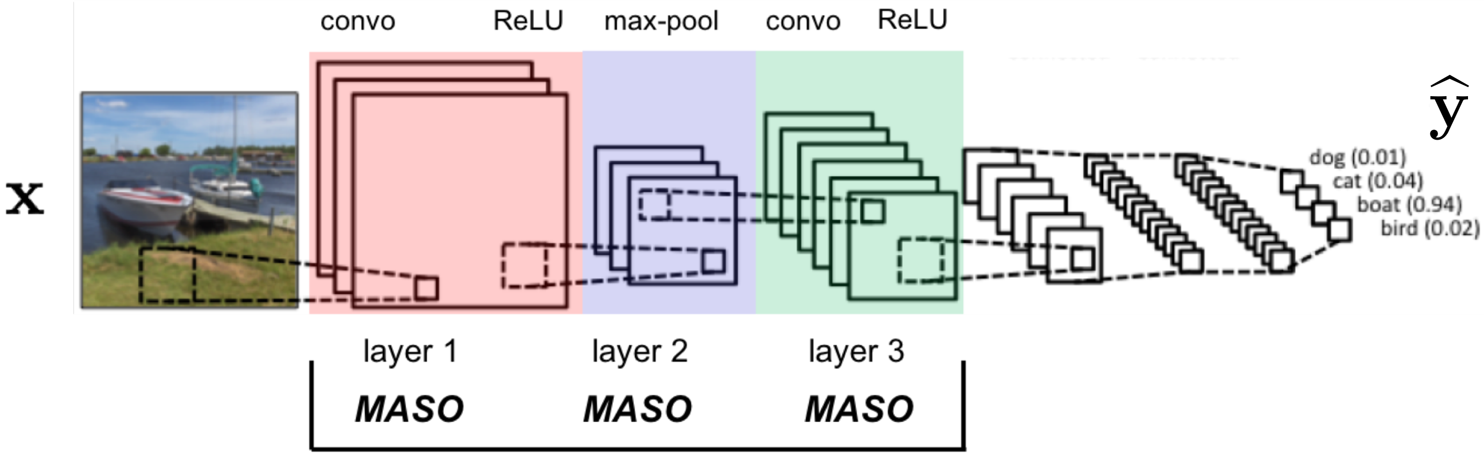
- **Focus:** The lion-share of today's deep net architectures (convnets, resnets, skip-connection nets, inception nets, recurrent nets, …) employ **piecewise linear (affine) layers** (fully connected, conv; (leaky) ReLU, abs value; max/mean/channel-pooling)



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

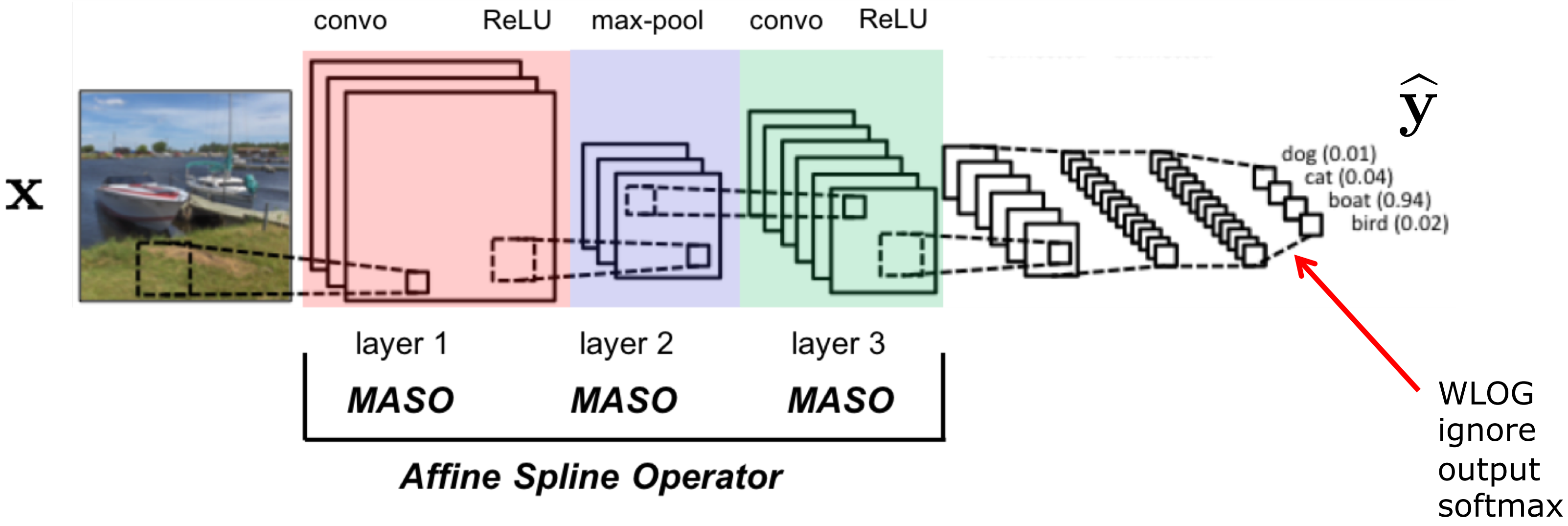
theorems

- Each deep net **layer** is a **MASO**
  - **convex** wrt each output dimension, piecewise-affine operator



theorems

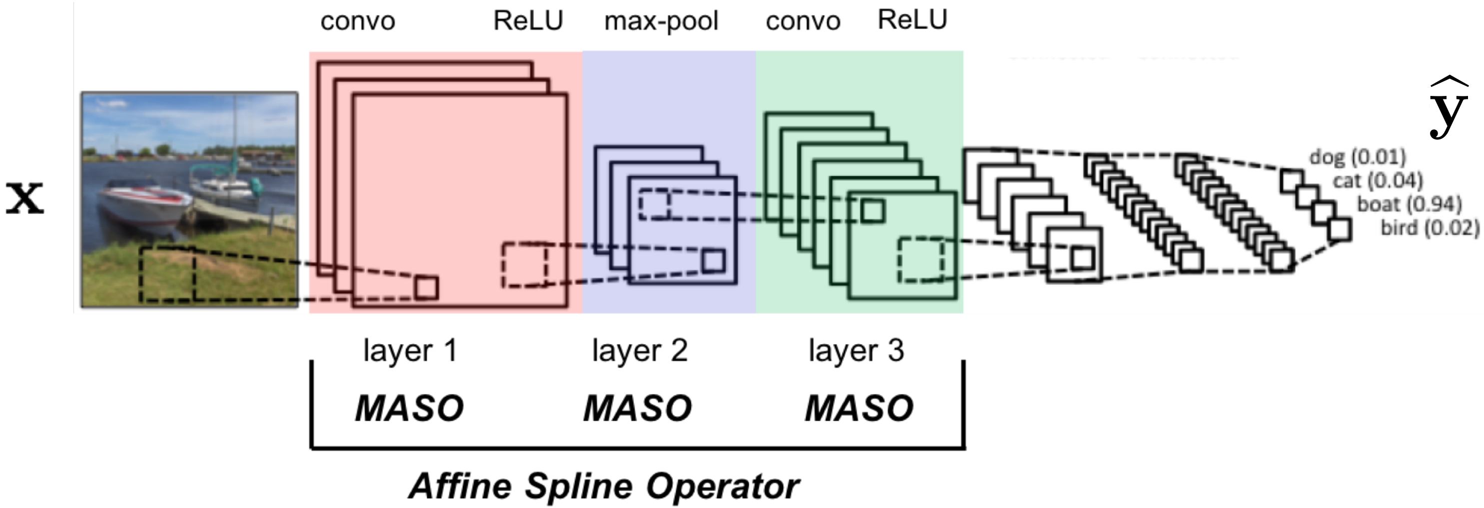
- Each deep net **layer** is a **MASO**
  - **convex**, piecewise-affine operator



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

theorems

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



- A deep net is a **convex MASO** iff the convolution/fully connected weights in all but the first layer are nonnegative and the intermediate nonlinearities are nondecreasing

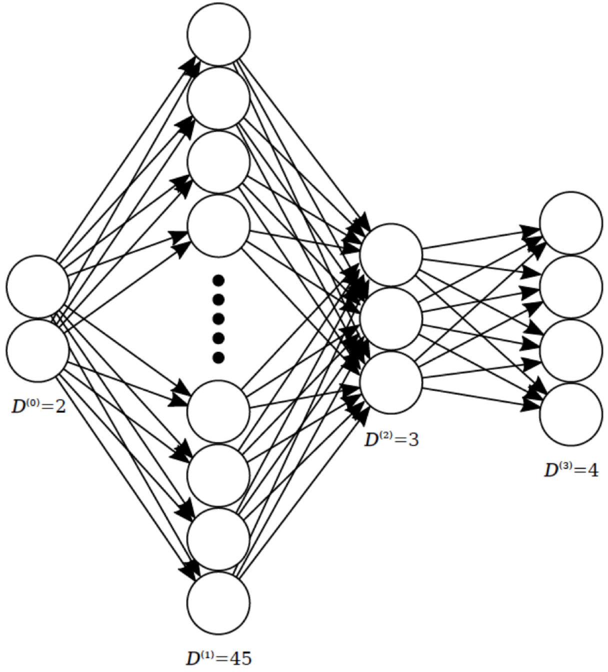
MASO spline partition

- The parameters of each **deep net layer** (MASO) induce a **partition** of its input space with **convex regions**
  - vector quantization (info theory)
  - *k*-means (statistics)
  - Voronoi tiling (geometry)

MASO spline partition

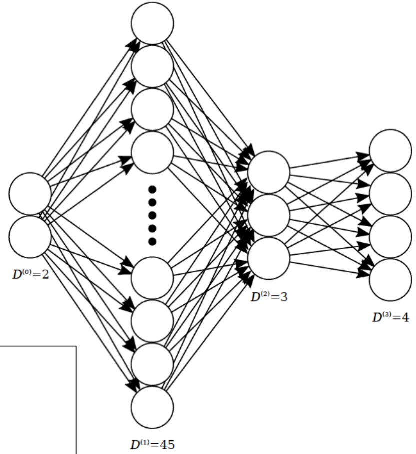
- The L layer-partitions of an L-layer deep net combine to form the **global input signal space partition**
  - affine spline operator
  - **non-convex regions**

- 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

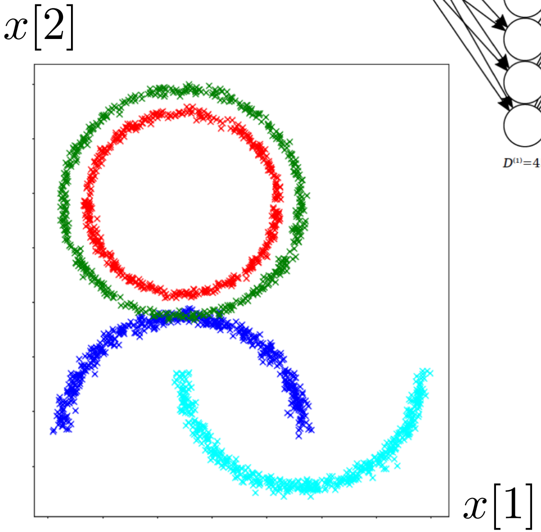


MASO spline partition

- 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**



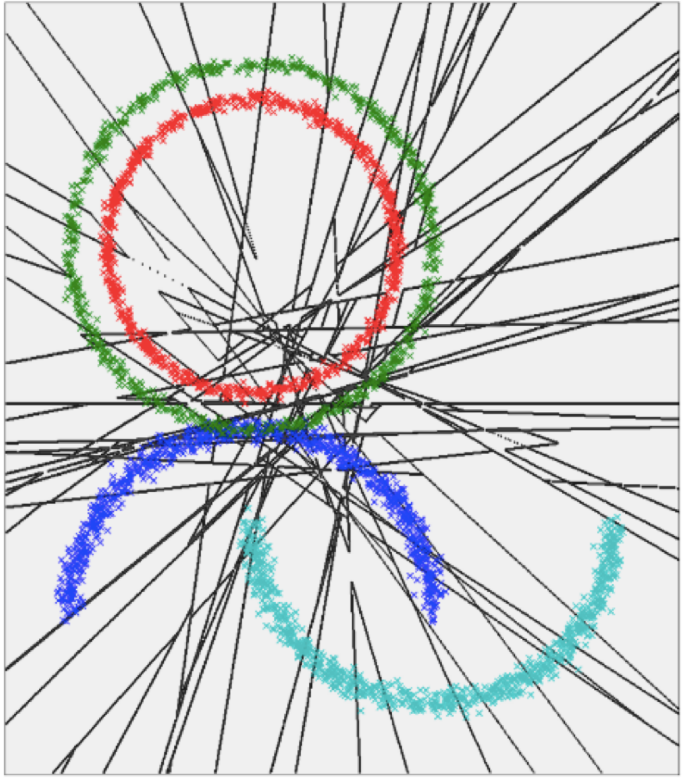
- Toy example: 3-layer "deep net"
  - **Input *x*: 2-D (4 classes)**
  - Fully connected | ReLU (45-D output)
  - Fully connected | ReLU (3-D output)
  - Fully connected | (softmax) (4-D output)
  - Output ***y***: 4-D



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

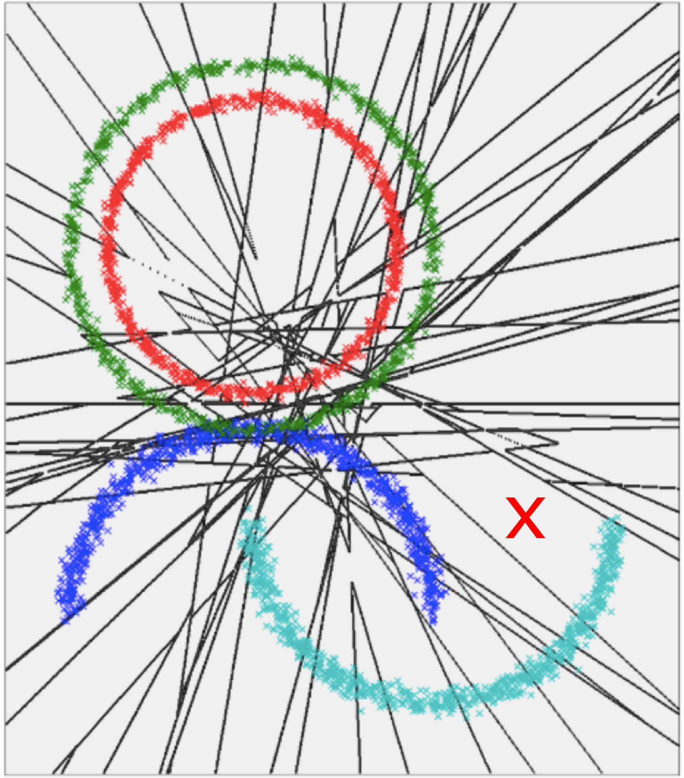


MASO spline partition

- 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**

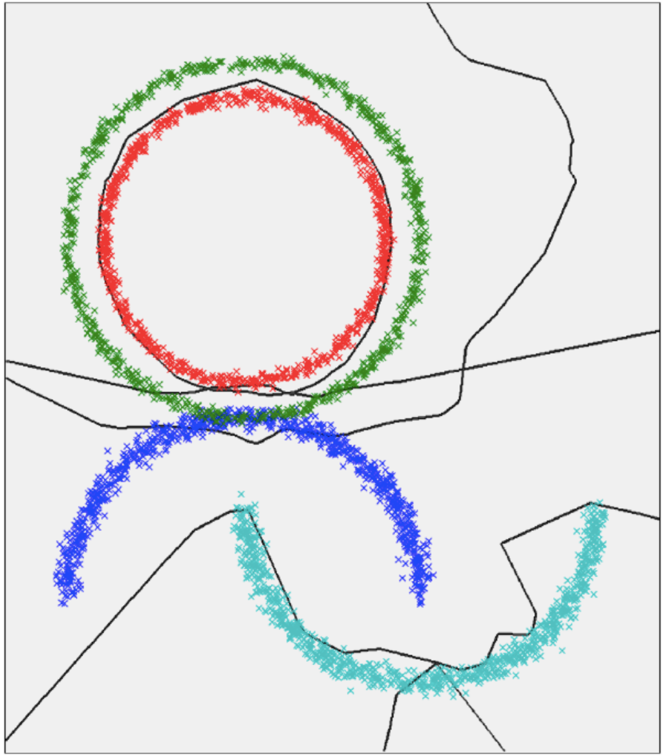
$$\mathbf{z}(\mathbf{x}) = \mathbf{A}_{Q(\mathbf{x})}\mathbf{x} + \mathbf{b}_{Q(\mathbf{x})}$$



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 **layer 2** depicted in the input space
  - **non-convex** regions due to visualization in the input space

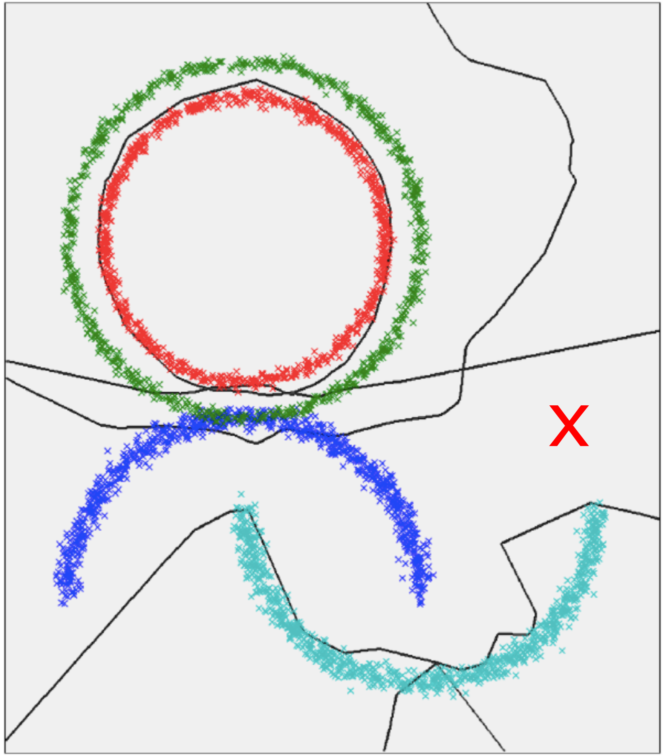


MASO spline partition

- 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

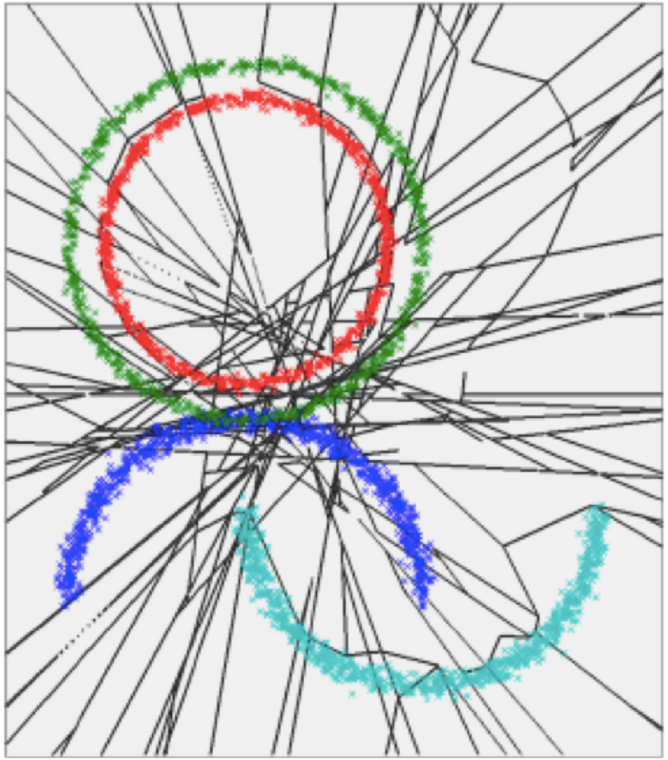
- Given the partition region $Q(\mathbf{x})$ containing $\mathbf{x}$ the layer **input/output mapping is affine**

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



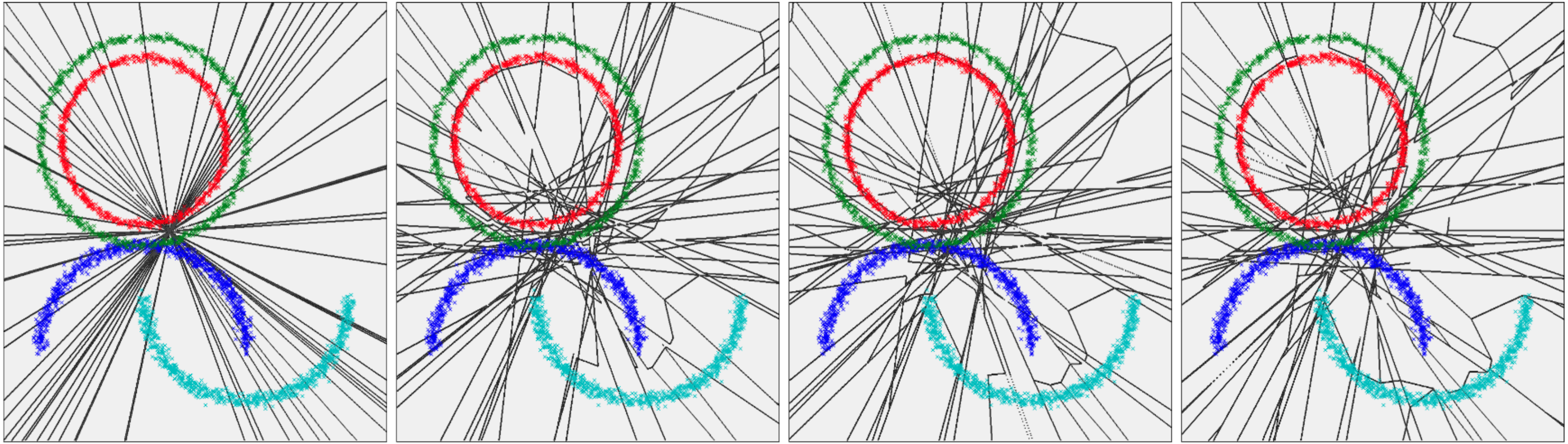
MASO spline partition

- Toy example: "Deep" net layer
  - 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



learning

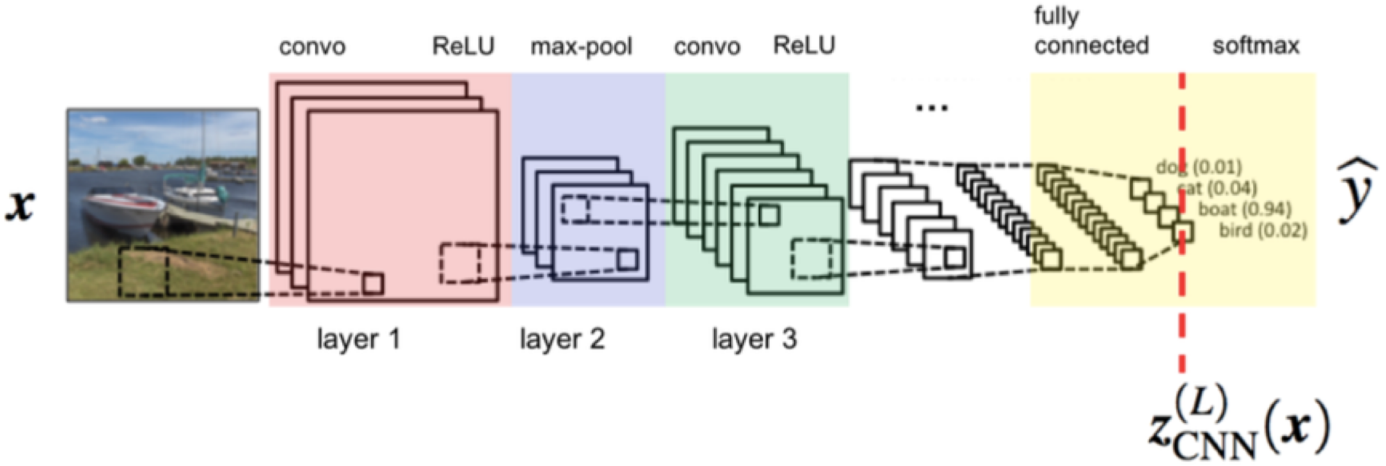
layers 1 & 2



learning epochs (time)

local affine mapping – CNN

Example: Classical CNN architecture with conv/ReLU/max-pooling layers terminating in a linear classifier comprising one fully connected layer and softmax

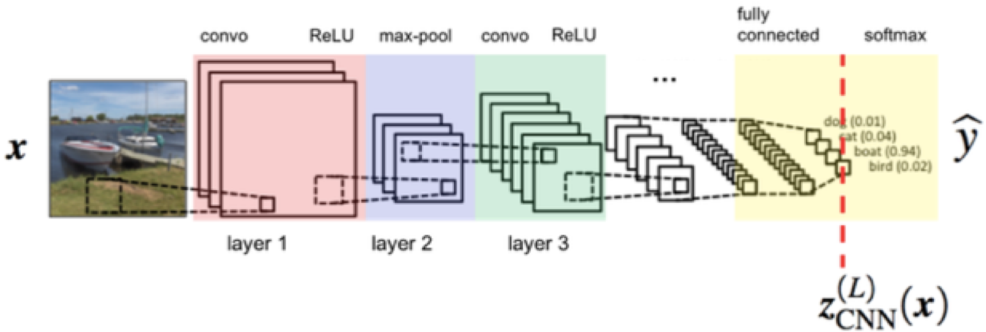


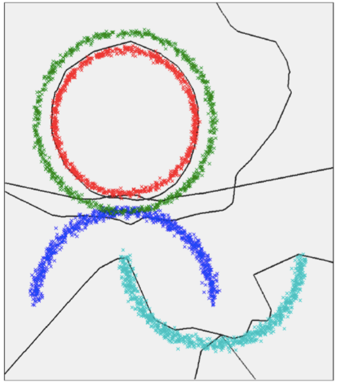
WLOG ignore output softmax

Result Input $(\boldsymbol{x})$ to output $\left(z_{\mathrm{CNN}}^{(L)}(\boldsymbol{x})\right)$ mapping is a **region-dependent affine transform**

$$z_{\mathrm{CNN}}^{(L)}(\boldsymbol{x}) = A_{Q(\boldsymbol{x})}\,\boldsymbol{x} + b_{Q(\boldsymbol{x})}[\boldsymbol{x}]$$

local affine mapping – CNN





Fixed, different

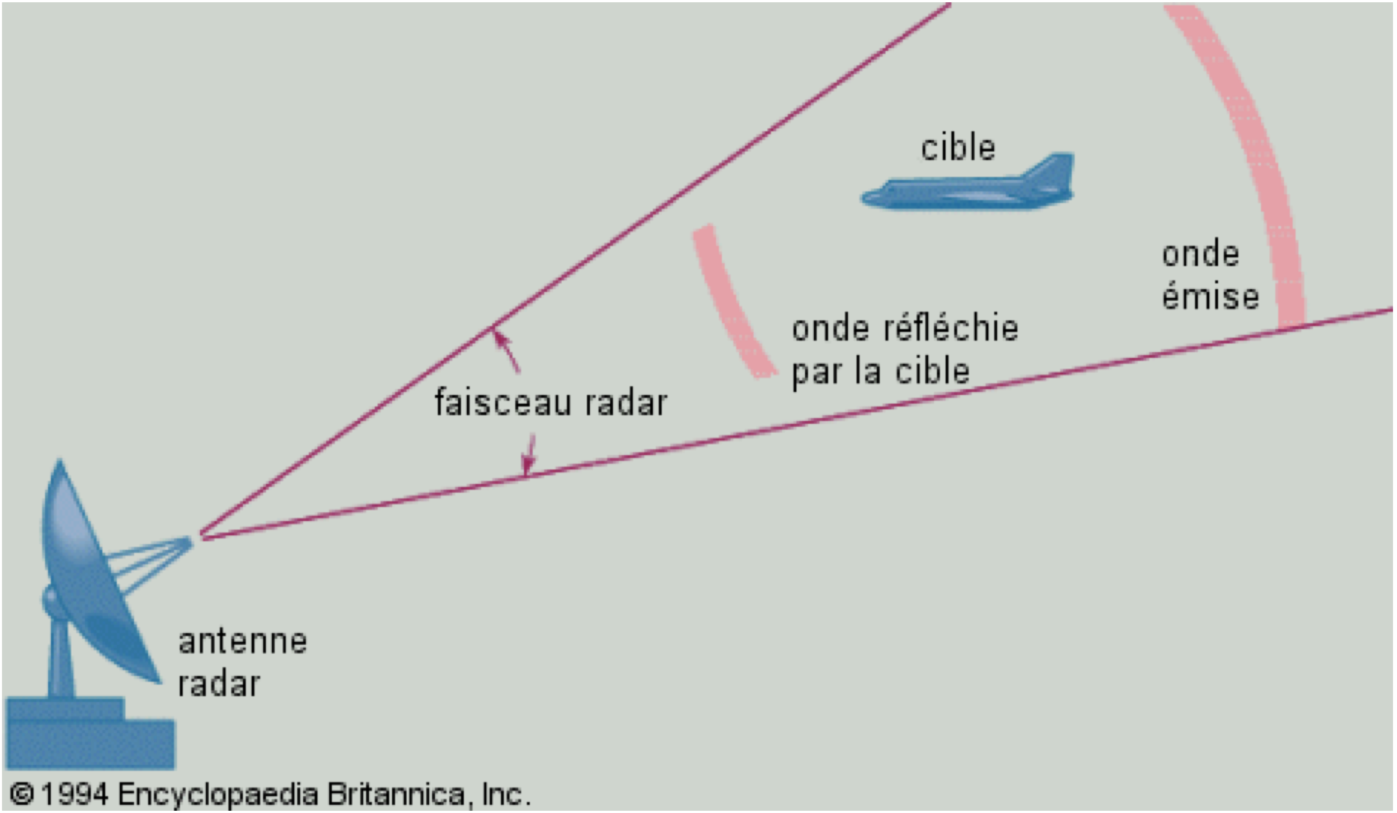
$\mathbf{A}_{Q(\mathbf{x})}, \ \mathbf{b}_{Q(\mathbf{x})}$

in each partition region

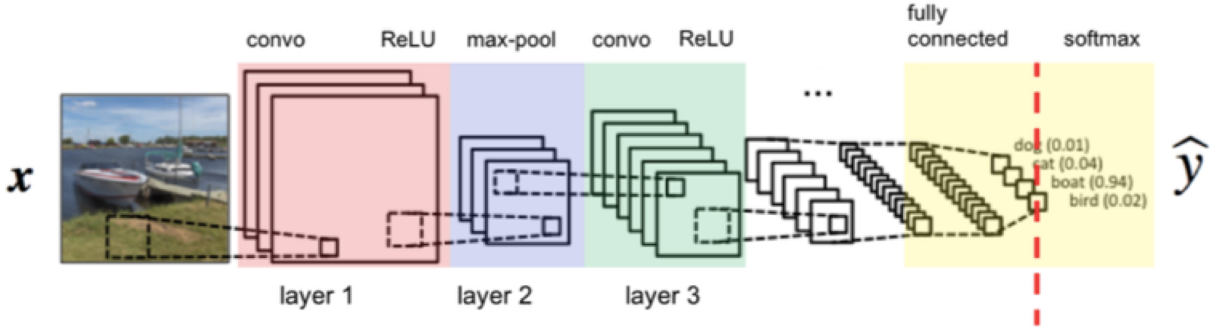
$$z_{\mathrm{CNN}}^{(L)}(\boldsymbol{x}) = \underbrace{\left(W^{(L)} \prod_{\ell=L-1}^{1} A_{\rho}^{(\ell)}[\boldsymbol{x}] A_{\sigma}^{(\ell)}[\boldsymbol{x}] C^{(\ell)} \right)}_{A_{Q(\boldsymbol{x})}\boldsymbol{x}}$$

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

matched filters



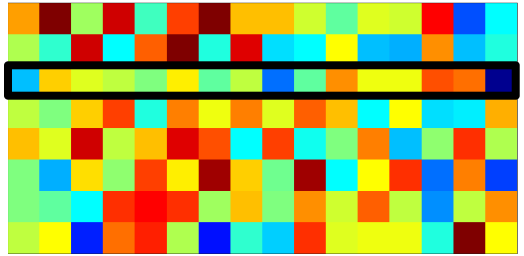
deep nets are matched filterbanks



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

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

c

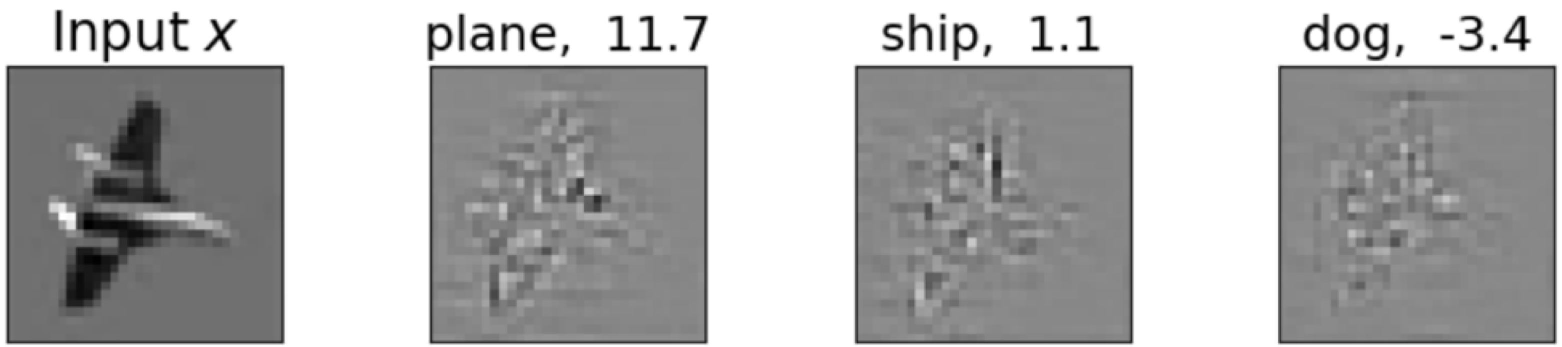


- Row c of $\mathbf{A}_{Q(\mathbf{x})}$ is a vectorized signal/image corresponding to class c
- Entry c of deep net output = inner product between row c and signal
- For classification, select largest output; **matched filter!**

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]

data memorization

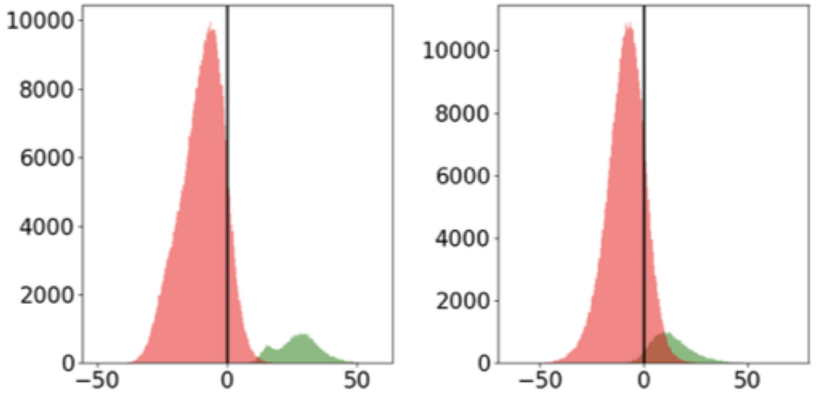
Result Matched filters of an infinite capacity deep net **memorize the training data** $\{(x_n, y_n)\}_{n=1}^{N}$

$$\text{row } c \text{ of } A_{Q(\boldsymbol{x}_n)} = \begin{cases} +\sqrt{\frac{(C-1)\alpha}{C}}\, \boldsymbol{x}_n, & c = y_n \quad \text{(correct class)} \\ -\sqrt{\frac{\alpha}{C(C-1)}}\, \boldsymbol{x}_n, & c \neq y_n \quad \text{(incorrect class)} \end{cases}$$

Experiment with MNIST, CIFAR10

Inner products between
training image $\boldsymbol{x}_n$ and rows of $A_{\mathrm{net}}[\boldsymbol{x}_n]$

- green: correct class (large positive)
- red: incorrect classes (large negative)



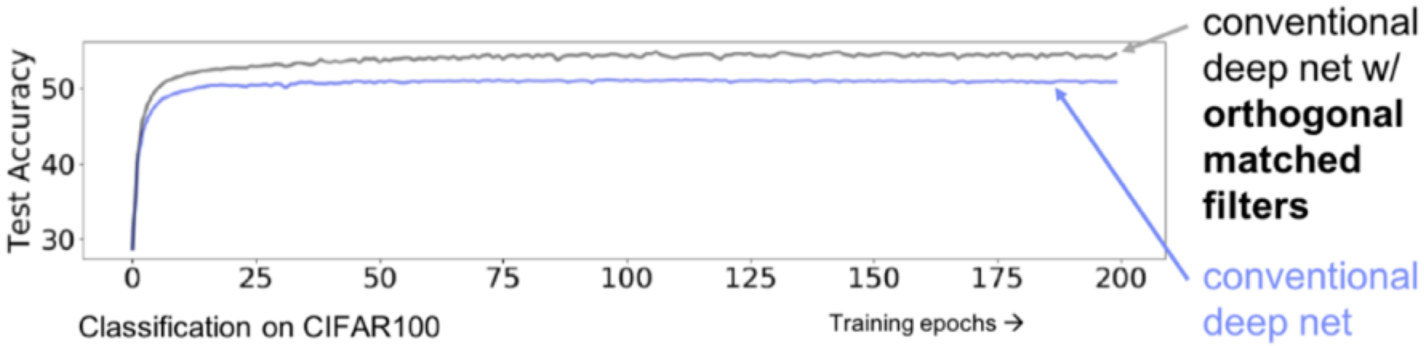
orthogonal deep nets

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

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

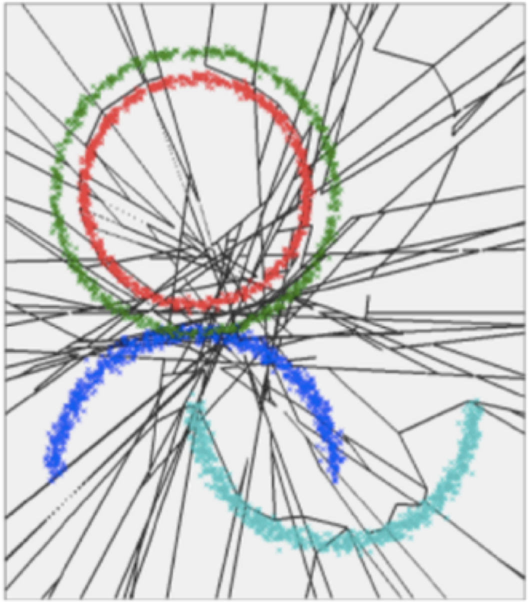
[Eldar and Oppenheim, 2001]

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

partition-based signal distance



Capture the geometry of the data space by measuring the **distance between the partition regions** inhabited by two signals $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$

Use **Hamming distance** between the codewords $Q(\boldsymbol{x}_1)$ and $Q(\boldsymbol{x}_2)$

Easily computed in terms of **activation patterns** of ReLU/max-pooling layers

Links with distance between **vector quantization encodings**

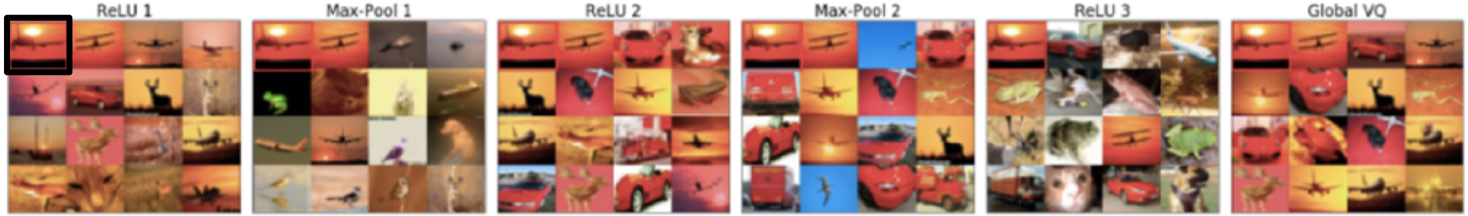
partition-based signal distance

15 **nearest neighbors** of a test image (upper left) using **spline partition (VQ) distance**

partition-based signal distance

15 **nearest neighbors** of a test image (upper left) using **spline partition (VQ) distance**

(a) Training with correct labels



(b) Training with random labels

(c) No Training

additional directions

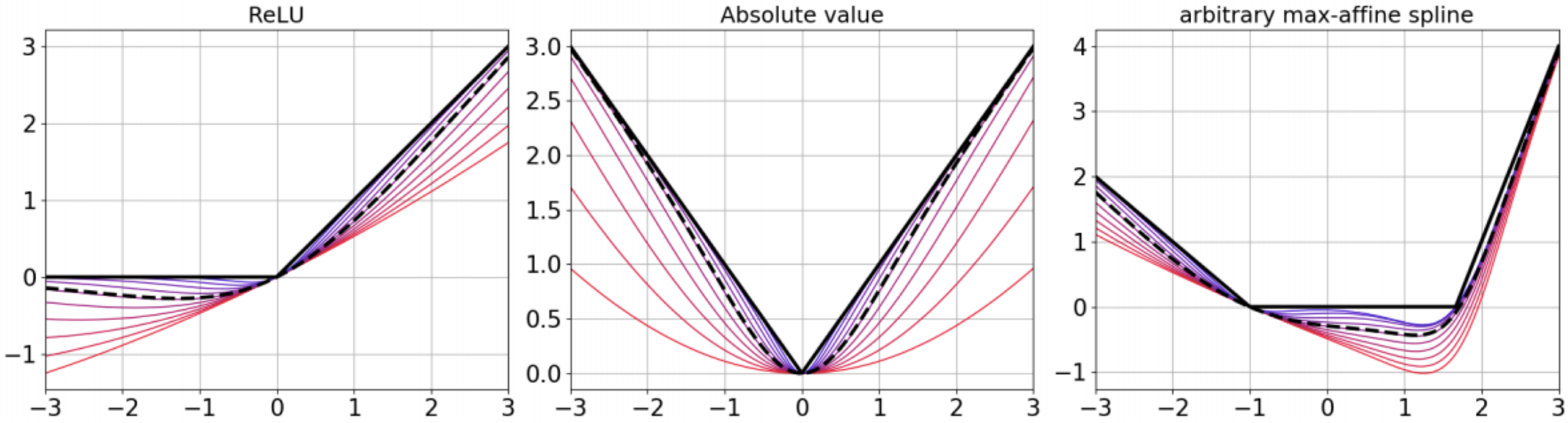
- Study the **geometry** of deep nets and signals via VQ partition

- Affine input/output formula enables explicit calculation of the **Lipschitz constant** of a deep net for the analysis of stability, adversarial examples, …

- Theory covers many **recurrent** neural networks (RNNs)

additional directions

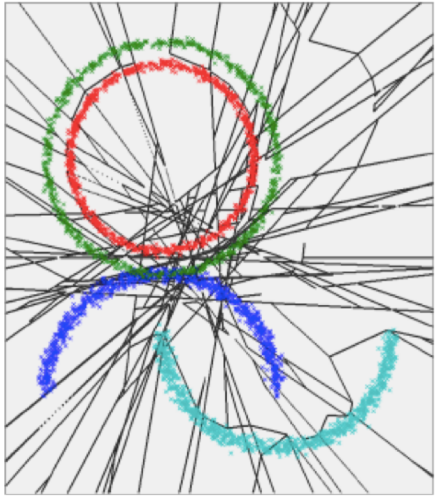
- Theory extends to non-piecewise-affine operators (ex: **sigmoid**) by replacing the "**hard VQ**" of a MASO with a "**soft VQ**"
  - soft-VQ can generate **new nonlinearities** (ex: swish)



summary

- 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**
  - enables explicit calculation of the Lipschitz constant of a deep net for the analysis of stability, adversarial examples, . . .



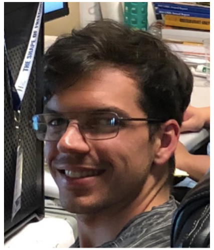
- Deep nets are (learned) **matched filterbanks**
  - new insights into dataset memorization

- Theory is **constructive**
  - inspires orthogonalized deep nets
  - new geometric distance via Hamming-VQ distance

RICE UNIVERSITY

dsp.rice.edu

max-affine **splines** and deep learning



R. Balestriero & B
"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)