10707
Deep Learning: Spring 2020

Andrej Risteski

Machine Learning Department

Lecture 2:
Benefits of depth

Recap

Recall from previous lecture:

(1): Neural networks are **universal approximators**: given any (reasonably nice) function f: $\mathbb{R}^d \to \mathbb{R}$, a **shallow** (3-layer) neural network with $\sim \left(\frac{1}{\epsilon}\right)^{\mathrm{d}}$ neurons can approximate it to within ϵ error.

(2): Neural networks are **circumvent** curse of dimensionality for functions f: $\mathbb{R}^d \to \mathbb{R}$ with appropriately decaying Fourier coefficients: **shallow** (-layer) neural networks with $\sim \left(\frac{1}{\epsilon}\right)$ neurons can approximate them to within ϵ error.

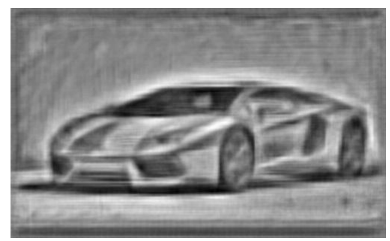
Is there any point to depth?
Are deeper networks more powerful?

Part of the deep learning story

Data → Feature extraction → Learning algorithm

Object detection

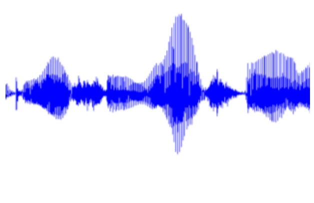
Image



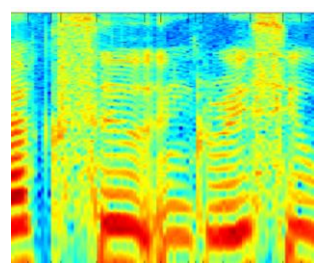
vision features

Recognition

Audio classification



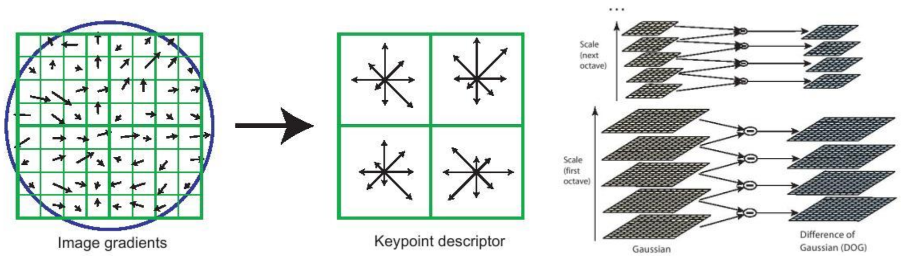
Audio



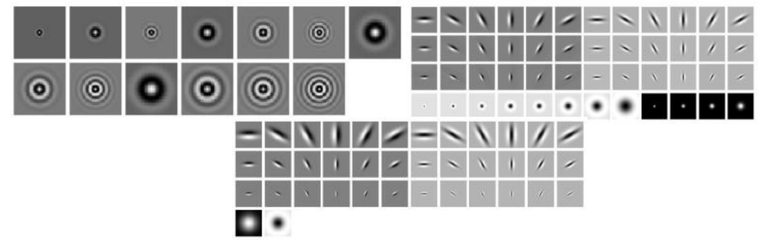
audio features

Speaker identification

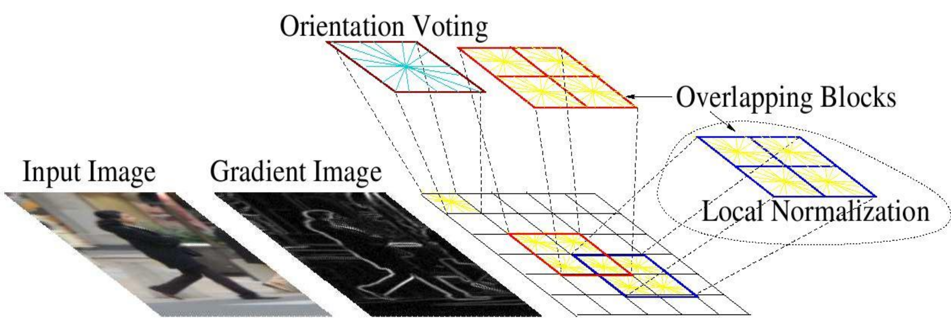
Old school: hand-craft features



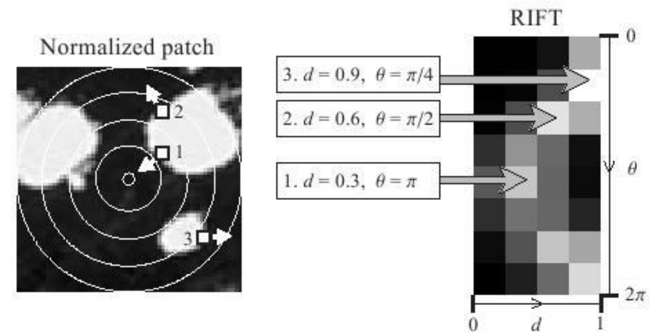
SIFT



Textons

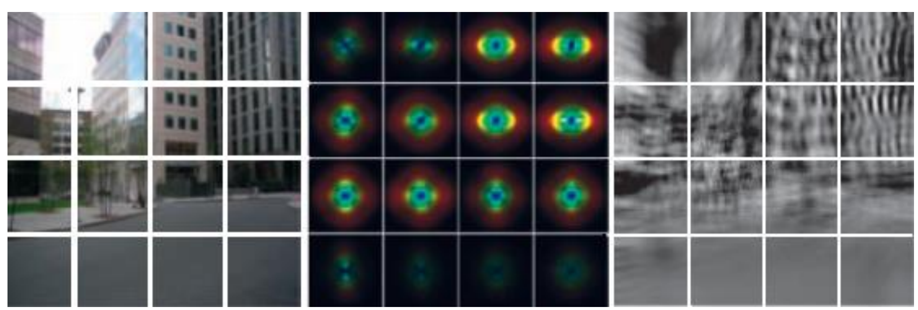


HoG

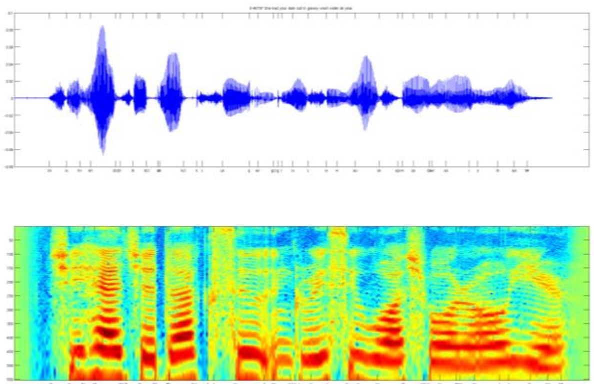


RIFT

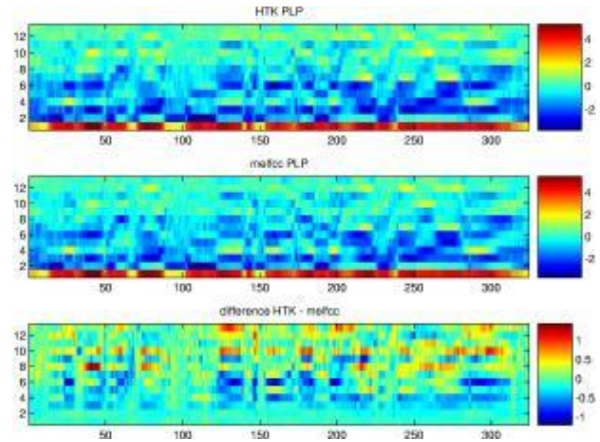
GIST



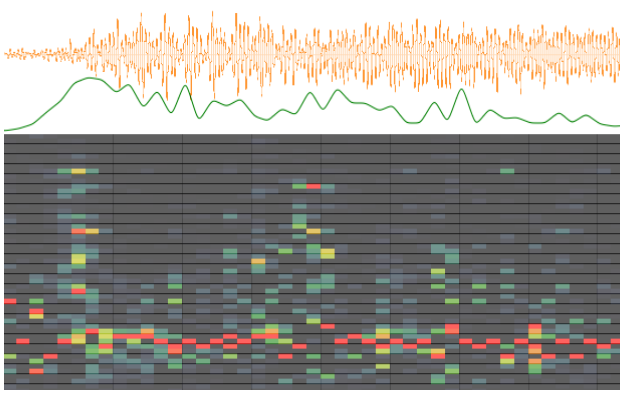
Old school: hand-craft features



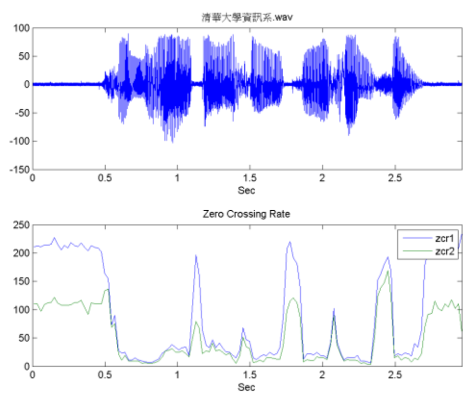
Spectrogram



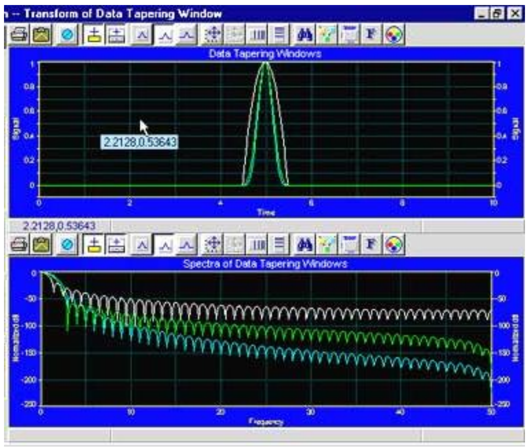
MFCC



Flux

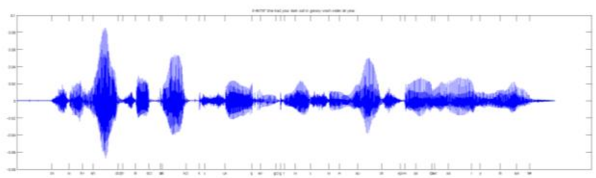


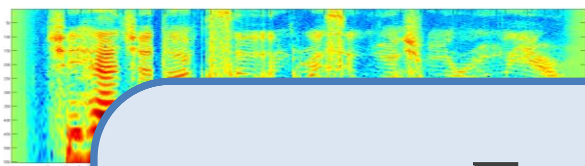
ZCR

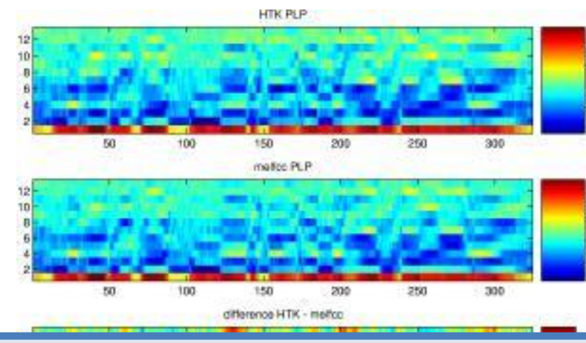


Rolloff

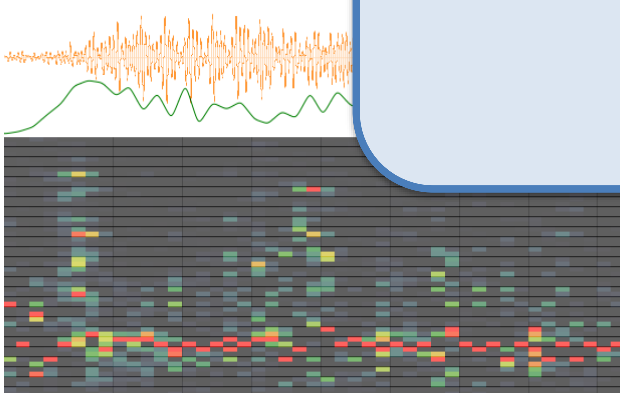
Old school: hand-craft features

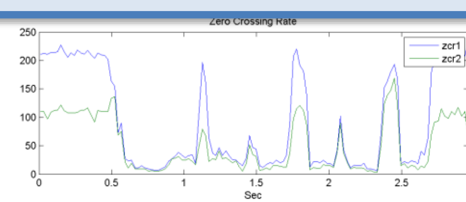


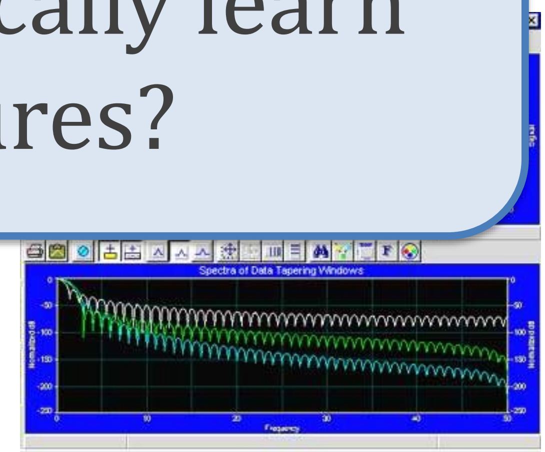




Feature learning:
Can we automatically learn useful features?





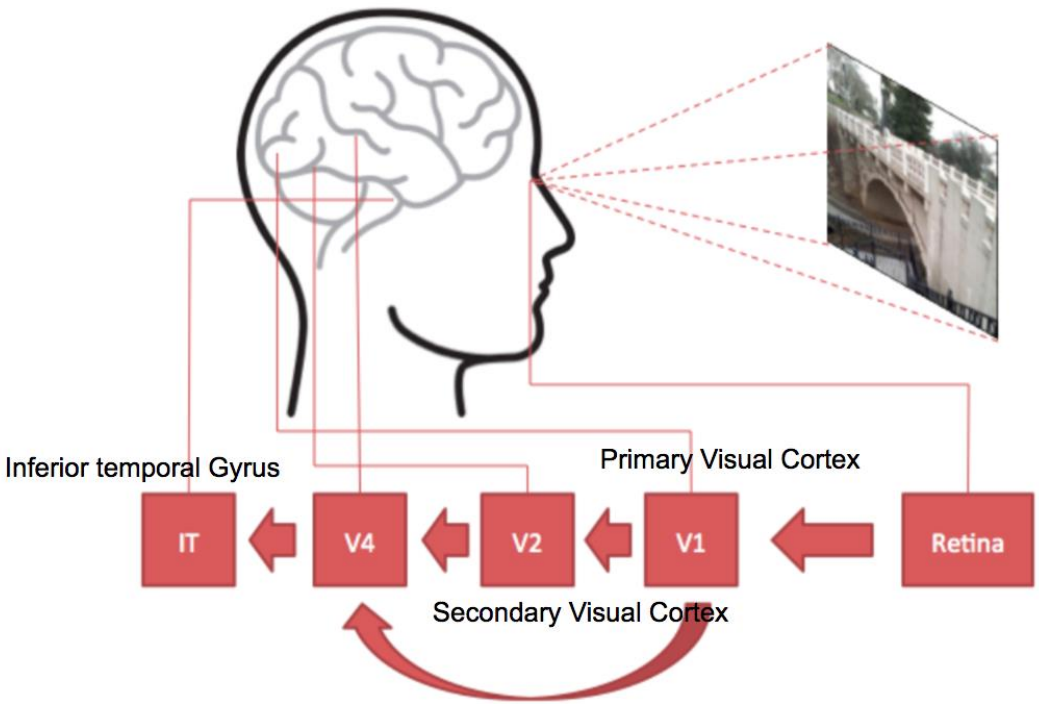


Flux

ZCR

Rolloff

Early inspirations from visual cortex



V1: Edge detection, etc.

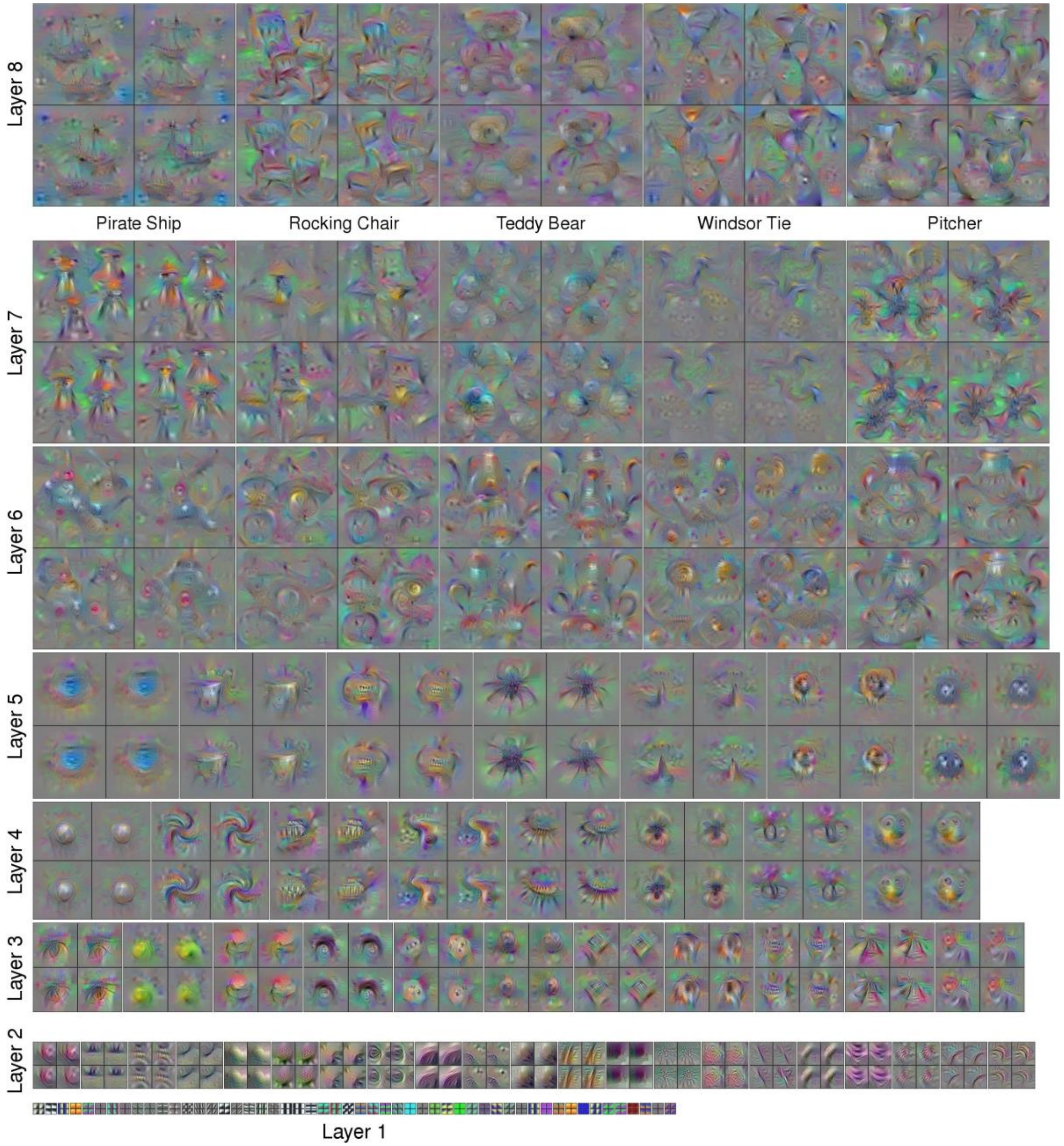
V2: Extract simple visual properties (orientation, spatial frequency, color, etc)

V4: Detect object features of intermediate complexity

TI: Object recognition.

Image: Wang, Raj, and Xing. [“On the Origin of Deep Learning.”]()

What do deep networks learn?



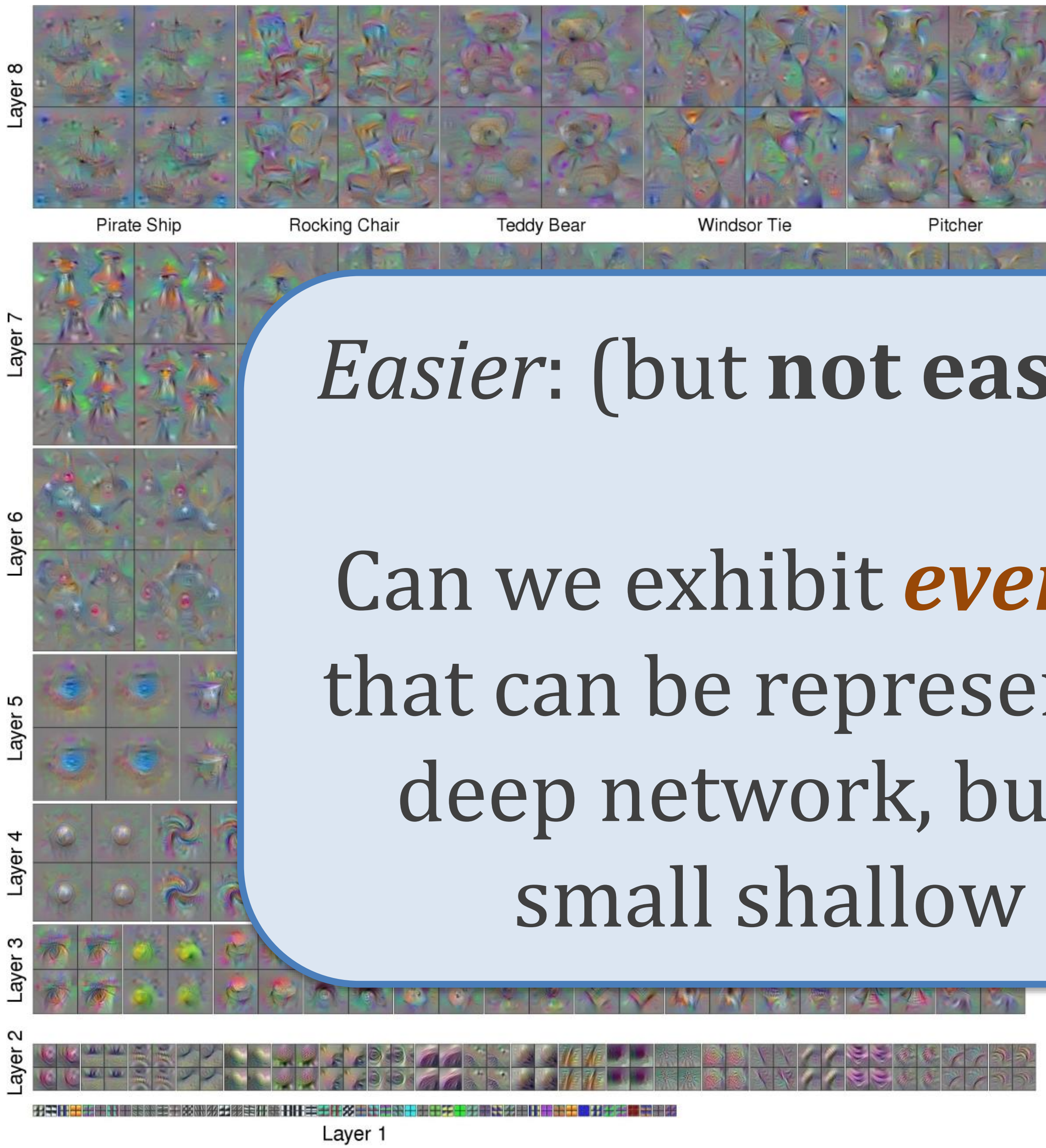
Yosinski et al ‘15: http://yosinski.com/deepvis

Q: What “patterns” do neurons respond to?

A: From random start, do gradient descent to find an input for which neuron activation* is high.

*: This produces completely unrecognizable images – they are regularized w/ an image prior.

What do deep networks learn?



Yosinski et al '15:
http://yosinski.com/deepvis

Easier: (but **not easy** it turns out!)

Can we exhibit ***even one*** function that can be represented as a small deep network, but cannot as a small shallow network?

A brief history of depth separation

Early roots: theoretical computer science

Boolean circuits: a directed acyclic graph model for computation over binary inputs; each node ("*gate*") performs an operation (e.g. OR, AND, NOT) on the inputs from its predecessors.

Relaxation of the P vs NP question: separate more "structured" models of computation – e.g. shallow vs deep Boolean circuits.

Seminal result by *Håstad ('86):* parity function (calculates parity of number of ones in input) cannot be approximated by a small constant-depth circuit with OR and AND gates.
[Highly non-trivial; **Gödel Prize!!**]

Modern iterations of depth separation

Related architectures/models of computation

Sum-product networks [*Bengio, Delalleau '11*]

Weaker measures of complexity

Bound on number of linear regions for ReLU networks [*Montufar, Pascanu, Cho, Bengio '14*]

True approximation error results

A small deep network cannot be approximated by a smaller shallow network [*Telgarsky '15*]

Separating deep and shallow networks

Can be imitated for higher dim too

Family of functions for ever L

Theorem (Telgarsky '15): For every $L \in \mathbb{N}$,

there is a function **f:** $[0,1] \to \mathbb{R}$ representable as a network of depth $\Theta(L^2)$, with $\Theta(L^2)$ nodes, and ReLU activation, s.t.

for every network **g :** $[0,1] \to \mathbb{R}$ of depth $O(L)$ and $O(2^L)$ nodes, and ReLU activation, we have

$$\int_{[0,1]} |f(x) - g(x)| dx \geq \frac{1}{32}$$

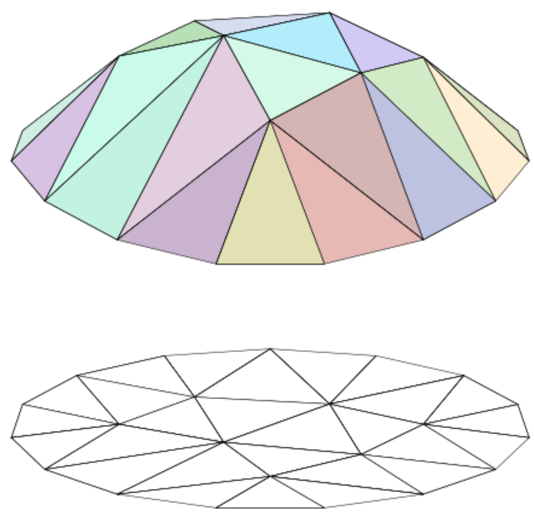
You cannot approximate f well using any g, ***even*** *in an* ***averaged*** (l_1) *sense.*

Outputting trivial approximation g = 0 will give error 1, so this is a really bad approximation.

Q: what can deep networks do easily?

A ReLU network f is **piecewise linear**: we can subdivide domain of f into a finite number of *polyhedral* pieces $(P_1, P_2, \ldots, P_N)$, s.t. in each piece, f is linear. In other words, $\forall x \in P_i, f(x) = A_i x + b_i$.

(Once we know which ReLUs are in the linear regime, and which are zeroed out, the function f calculates is linear.)



Deeper networks can make a larger number of pieces.

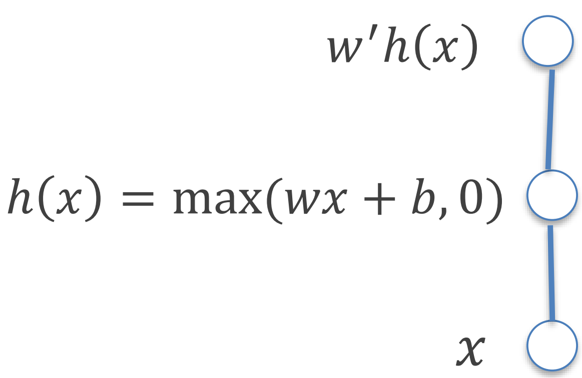
Shallow functions have few linear pieces

Let's reason how the number of linear pieces behaves under compositions.

Claim: if $f: \mathbb{R} \to \mathbb{R}$ is a ReLU network with hidden layer widths $(m_1, m_2, \dots, m_L)$. Then, f has at most $2^{L-1}(m_1 + 1)\, m_2 \dots m_L$ linear pieces.

Proof: Induction on L:

L = 1:



If only one hidden node: one value if $x \geq -\dfrac{b}{w}$

another if $x < -\dfrac{b}{w}$

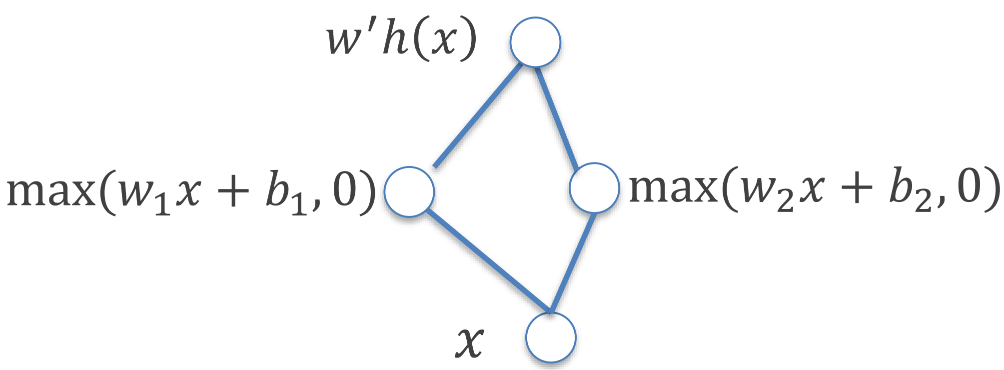
Shallow functions have few linear pieces

Let's reason how the number of linear pieces behaves under compositions.

Claim: if $f: \mathbb{R} \to \mathbb{R}$ is a ReLU network with hidden layer widths $(m_1, m_2, \ldots, m_L)$. Then, f has at most $2^{L-1}(m_1 + 1)\, m_2 \ldots m_L$ linear pieces.

Proof: Induction on L:

L = 1:



Each node introduces at most one breakpoint, e.g.:

Let's assume $-\frac{b_2}{w_2} \geq -\frac{b_1}{w_2}$

Different values in

$$\left[0, -\frac{b_1}{w_1}\right], \left[-\frac{b_1}{w_1}, -\frac{b_2}{w_2}\right], \left[-\frac{b_2}{w_2}, 1\right]$$

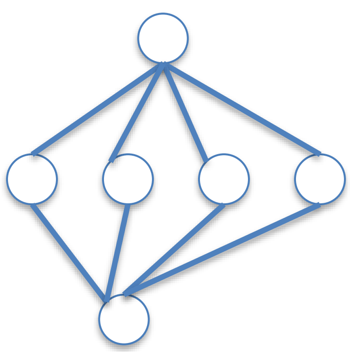
Shallow functions have few linear pieces

Let's reason how the number of linear pieces behaves under compositions.

Claim: if $f: \mathbb{R} \to \mathbb{R}$ is a ReLU network with hidden layer widths $(m_1, m_2, \ldots, m_L)$. Then, f has at most $2^{L-1}(m_1 + 1)\, m_2 \ldots m_L$ linear pieces.

Proof: Induction on L:

L = 1:

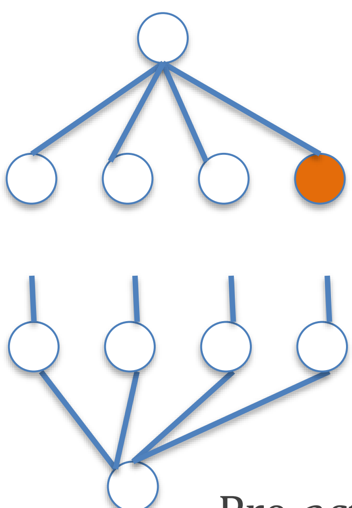


Continuing, number of pieces is at most $m_1 + 1$, as we need.

Shallow functions have few linear pieces

Claim: if $f : \mathbb{R} \to \mathbb{R}$ is a ReLU network with hidden layer widths $(m_1, m_2, \ldots, m_L)$. Then, f has at most $2^{L-1}(m_1 + 1)\, m_2 \ldots m_L$ linear pieces.

Proof: Inductive step:



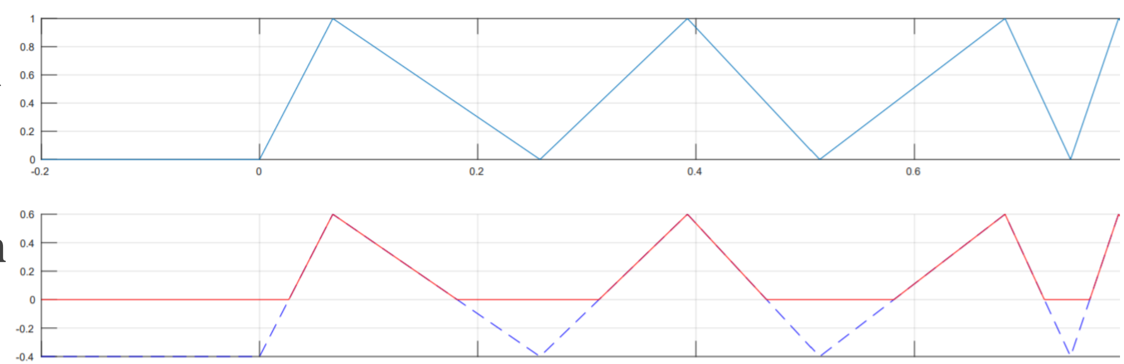
Take any **node** in penultimate layer:

By inductive hypothesis, pre-activation of node is piecewise linear, $\leq 2^{L-2}(m_1 + 1)\, m_2 \ldots m_{L-1}$ pieces.

Applying ReLU to pre-activation of **node** can at most double the number of pieces for that node !

Pre-activation fn for node

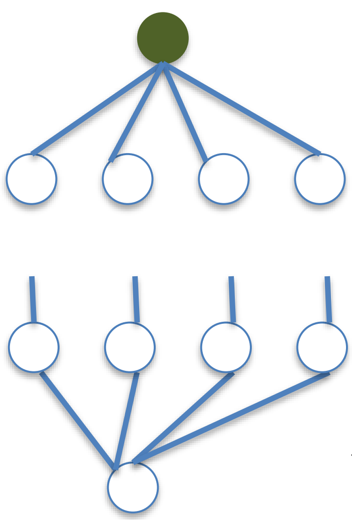
Post-activation fn for node



Shallow functions have few linear pieces

Claim: if $f: \mathbb{R} \to \mathbb{R}$ is a ReLU network with hidden layer widths $(m_1, m_2, \ldots, m_L)$. Then, f has at most $2^{L-1}(m_1 + 1)\, m_2 \ldots m_L$ linear pieces.

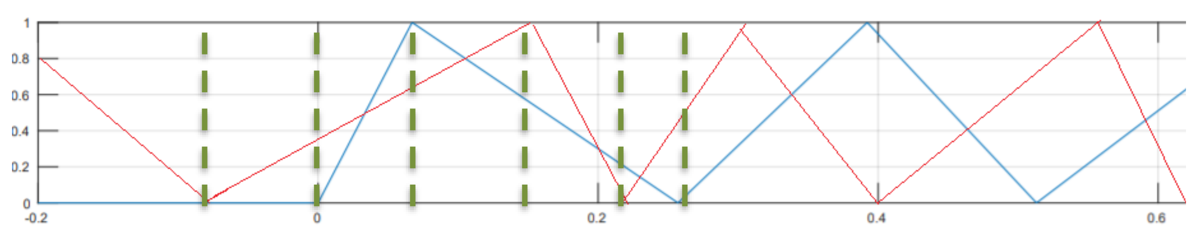
Proof: Induction on L:



So, **output** is linear combination of m_L piecewise linear function, each with at most $2^{L-1}(m_1 + 1)\, m_2 \ldots m_{L-1}$ pieces.

A linear comb. of 2 piecewise lin. fns, each w/ at most **a** and **b** pieces gives a piecewise lin. fn w/ at most **a+b** pieces:

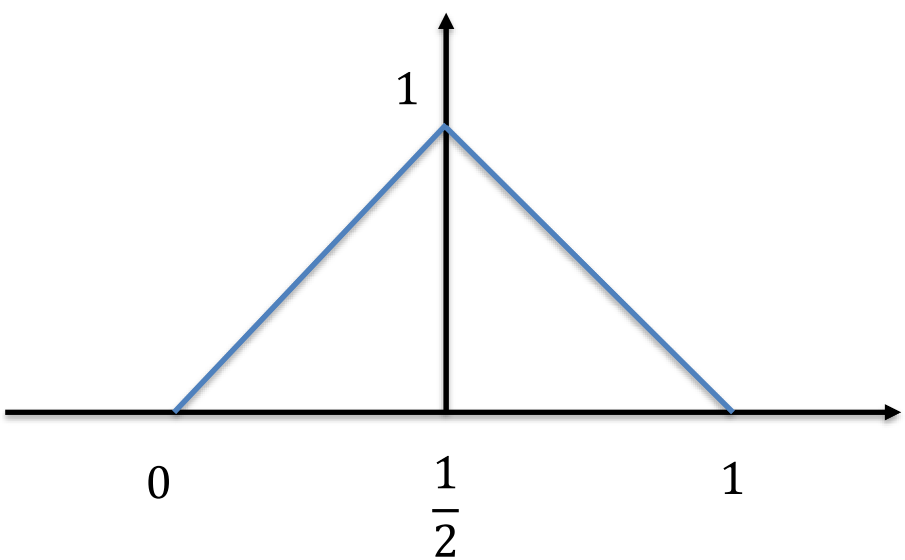
Hence, output has at most $2^{L-1}(m_1 + 1)\, m_2 \ldots m_L$ pieces.



Deep function with many linear pieces

The player : the "triangle" map $\Delta: [0,1] \to \mathbb{R}$

$$\Delta(\mathrm{x}) = 2\sigma(x) - 2\sigma(2x-1) = \begin{cases} 2x, & \text{if } x \leq 1/2 \\ 2-2x, & \text{if } x > 1/2 \end{cases}$$

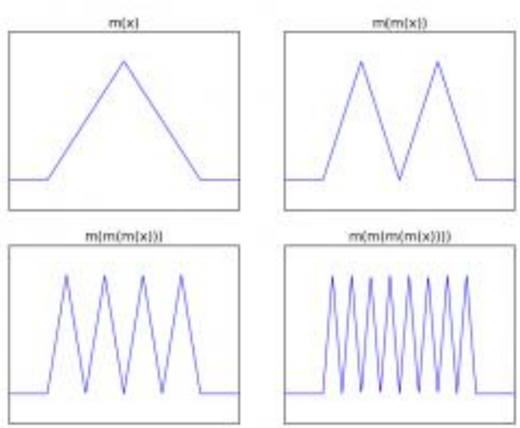


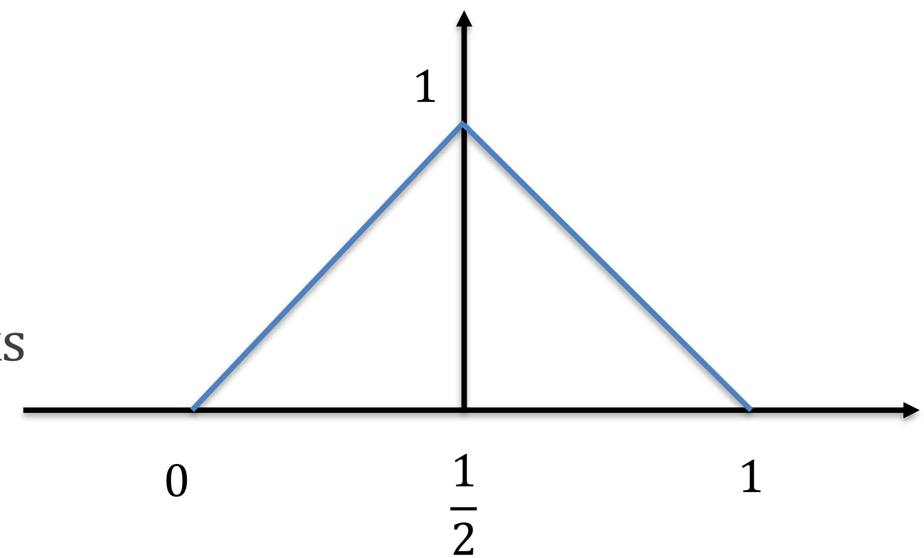
Deep function with many linear pieces

$$\Delta(x) = 2\sigma(x) - 2\sigma(2x-1) = \begin{cases} 2x, & \text{if } x \le 1/2 \\ 2-2x, & \text{if } x > 1/2 \end{cases}$$

Idea: compose the triangle function w/ itself many times!

Idea: composing it k times should look like sawtooth with $\sim 2^{k-1}$ peaks



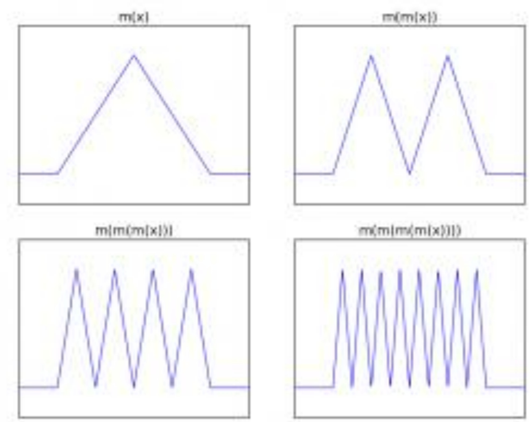


We'll show that function w/ small number of oscillations can't approximate this well.

Deep function with many linear pieces

$$\Delta(\mathrm{x}) = 2\sigma(x) - 2\sigma(2x-1) = \begin{cases} 2x, & \text{if } x \le 1/2 \\ 2-2x, & \text{if } x > 1/2 \end{cases}$$

Claim: $\Delta^k(\mathrm{x}) = \Delta\left(\underbrace{2^{k-1}\mathrm{x} - \lfloor 2^{k-1}x \rfloor}_{\text{“Squish” triangle in every } 1/2^k \text{ -sized interval}}\right)$



Proof: Induction:

K=1: by definition

K ⇒ K+1:

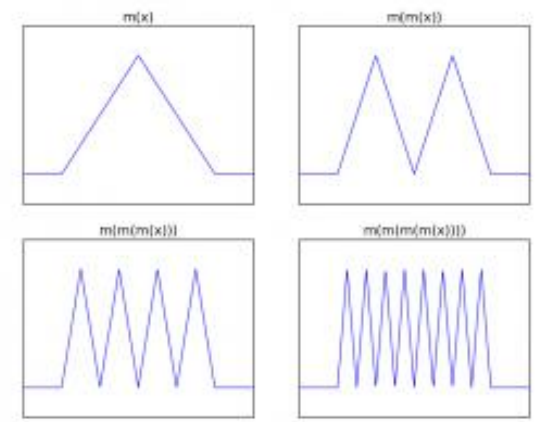
$x \le \frac{1}{2}$: $\quad \Delta^{k+1}(\mathrm{x}) = \Delta^k\big(\Delta(x)\big) = \Delta^k(2x) = \Delta\left(2^k\mathrm{x} - \lfloor 2^k x \rfloor\right)$

Deep function with many linear pieces

$$\Delta(\mathrm{x}) = 2\sigma(x) - 2\sigma(2x-1) = \begin{cases} 2x, & \text{if } x \leq 1/2 \\ 2-2x, & \text{if } x > 1/2 \end{cases}$$

Claim: $\Delta^k(\mathrm{x}) = \Delta\underbrace{\left(2^{k-1}\mathrm{x} - \lfloor 2^{k-1}x \rfloor\right)}$

"Squish" triangle in every $1/2^k$ -sized interval



Proof: Induction:

K=1: by definition

K $\Rightarrow$ K+1:

$$x > \frac{1}{2}: \quad \Delta^{k+1}(\mathrm{x}) = \Delta^k\big(\Delta(x)\big) = \Delta^k(2-2x) = \Delta^{\mathrm{k}-1}(\Delta(2-2x)) = \Delta^{\mathrm{k}-1}(\Delta(1-(2-2x)))$$

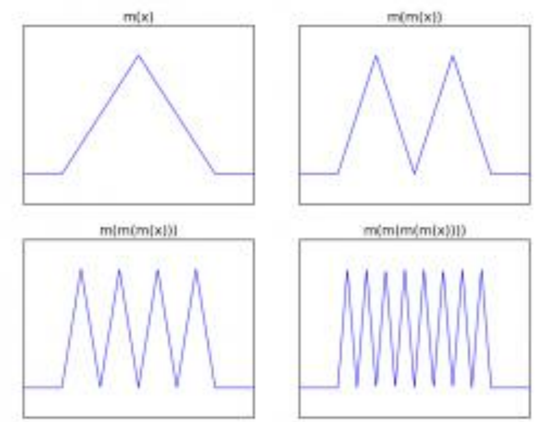
$$= \Delta^{\mathrm{k}-1}\,(\Delta(2x-1)) = \Delta\big(2^k\mathrm{x} - 2^{\mathrm{k}} - \lfloor 2^k\mathrm{x} - 2^{\mathrm{k}} \rfloor\big) = \Delta\big(2^k\mathrm{x} - \lfloor 2^k\mathrm{x} \rfloor\big)$$

Deep function with many linear pieces

$$\Delta(\mathrm{x}) = 2\sigma(x) - 2\sigma(2x-1) = \begin{cases} 2x, & \text{if } x \le 1/2 \\ 2-2x, & \text{if } x > 1/2 \end{cases}$$

Claim: $\Delta^k(\mathrm{x}) = \Delta\left(\underbrace{2^{k-1}\mathrm{x} - \lfloor 2^{k-1}x \rfloor}\right)$

"Squish" triangle in every $1/2^k$ -sized interval

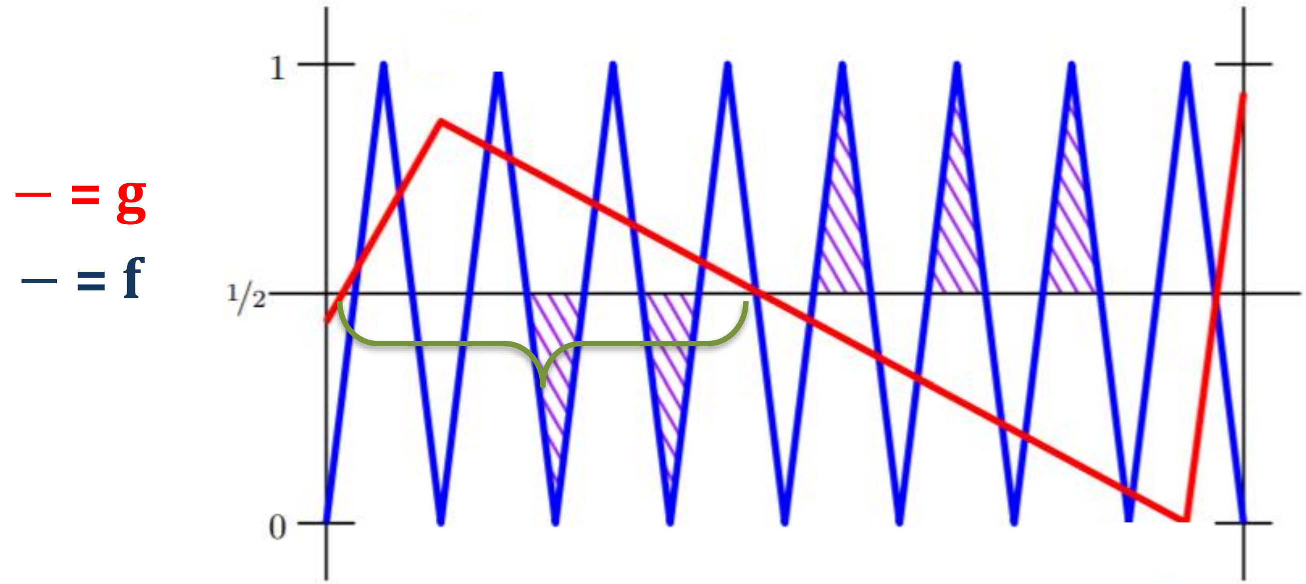


Some C we will choose

*The function **f** in the theorem will be $\Delta^{C \cdot L^2}(x)$*

We will show shallow g's can't approximate f.

Shallow g's can't approximate f



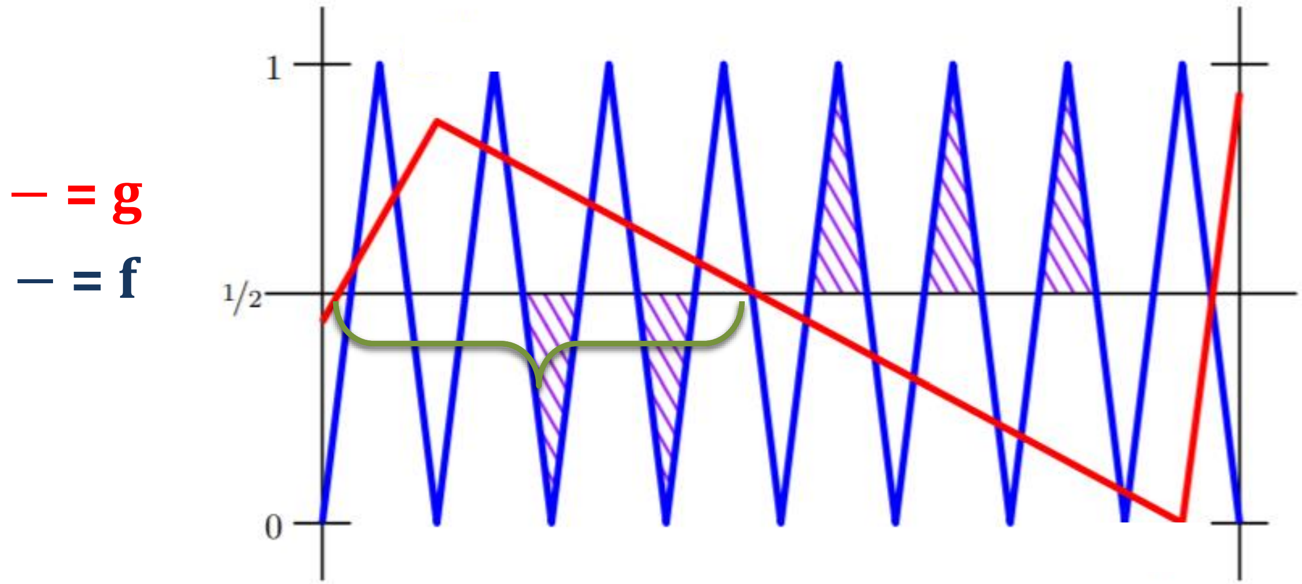
Consider the line **y = ½** . We will count triangles of **f** on a different side than the **g**. Each contributes at least ½ x (**triangle area**) to $\int|f-g|dx$.

Let $\boldsymbol{N_f = 2^{C \cdot L^2} - 2}$ and $\boldsymbol{N_g}$ be the number of triangles of **f**, **g**.

For any **interval** of **g** corresponding to triangle: if there are **m** triangles of **f** contained entirely in interval, $\geq \frac{\mathbf{m-1}}{\mathbf{2}}$ lie on opposite side of **g**.

We "miss" at most $\boldsymbol{N_g}$ triangles (i.e. they are not contained entirely in an **interval** of **g**). Hence, we "contribute" at least $(\boldsymbol{N_f - N_g - 1})/\mathbf{2}$ triangles.

Shallow g's can't approximate f



Let N_f and N_g be the number of triangles of **f**, **g**. We "contribute" at least $(N_f - N_g - 1)/2$ triangles.

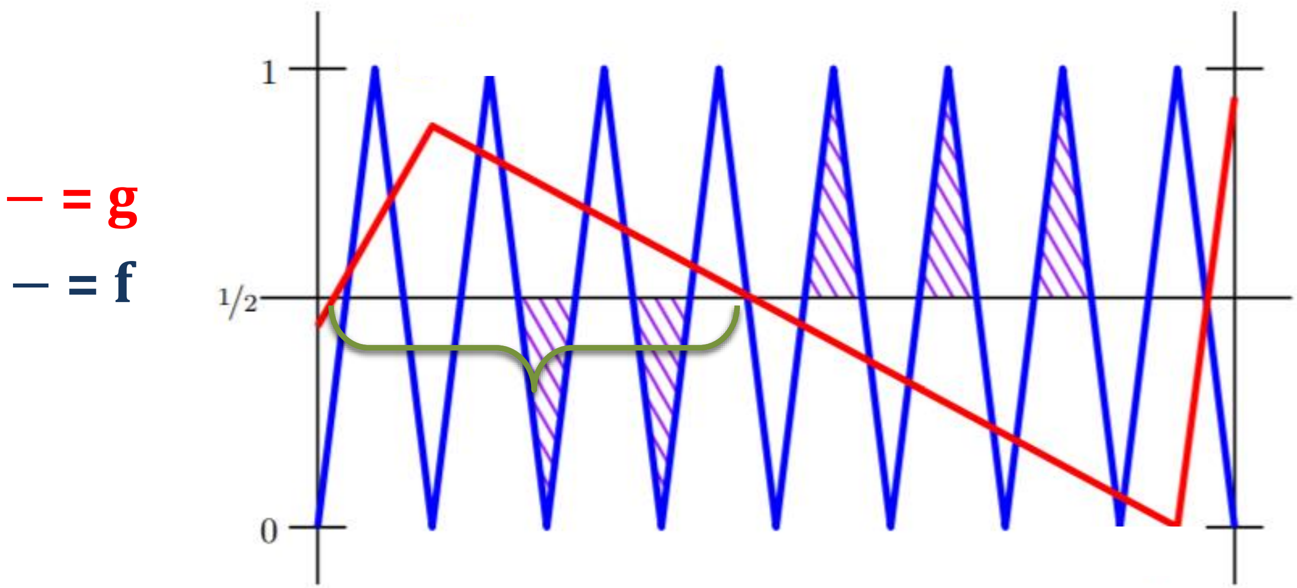
Size of g

Now, since shallow nets have small N_g: $N_g \leq (2n)^L \leq 2^{O(L^2)}$

We also showed that $N_f = 2^{C \cdot L^2} - 2$

$\Rightarrow$ We "contribute" at least $\Omega\left(2^{C \cdot L^2}\right)$ triangles.

Shallow g's can't approximate *f*



We "contribute" at least $\Omega\left(2^{C\cdot L^2}\right)$ triangles.

Each triangle has area ½ x (base) x (height)

$$= \tfrac{1}{2} \times \left(\frac{1}{2^{C\cdot L^2}}\right) \times \tfrac{1}{2} = \frac{1}{2^{C\cdot L^2+4}}$$

Total contribution: $\Omega\left(2^{C\cdot L^2} \times \frac{1}{2^{C\cdot L^2+4}}\right) \geq \frac{1}{32}$

For appropriate constants

Parting thoughts

Thm can be generalized to **d dimensions**, lots of other activation functions. (Need to bound # of pieces for multidim compositions.)

There can be benefits of depth to approximating **smooth** functions

Thm (*Yarotsky '16*). Suppose $\boldsymbol{f}: [\mathbf{0}, \mathbf{1}]^{\boldsymbol{d}} \to \mathbb{R}$ has all partial derivs of order r coordinate-wise bdd in $[-1, +1]$ and let $\boldsymbol{\epsilon} > \mathbf{0}$ be given. Then there exists a $\boldsymbol{O}\left(\ln \frac{\mathbf{1}}{\boldsymbol{\epsilon}}\right)$ – depth and $\left(\frac{\mathbf{1}}{\boldsymbol{\epsilon}}\right)^{\boldsymbol{O}\left(\frac{\boldsymbol{d}}{\boldsymbol{r}}\right)}$ –size network so that $\sup_{\boldsymbol{x} \in [\mathbf{0},\mathbf{1}]^{\boldsymbol{d}}} |\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{g}(\boldsymbol{x})| \leq \boldsymbol{\epsilon}$

Interaction of depth w/ **architecture**: depth tends to be problematic from an optimization point of view ("*vanishing gradient problem*").

Lots of proposal architectures to remedy this (**ResNet, DenseNet**)

In general, interplay of depth with both optimization and generalization is **not** well understood.

Stay tuned!