Lecture 9

Architecture 2: Expressivity

CS312

Tatsu H

Reminders

A5 (Capacity) due Tue 11/10, 1pm

Quiz Tue 11/10, in class

Old quiz: make sure to put your rationales/citations in *each subpart*

Today's lecture: expressivity

Last two lectures: training a model by thinking about signal propagation

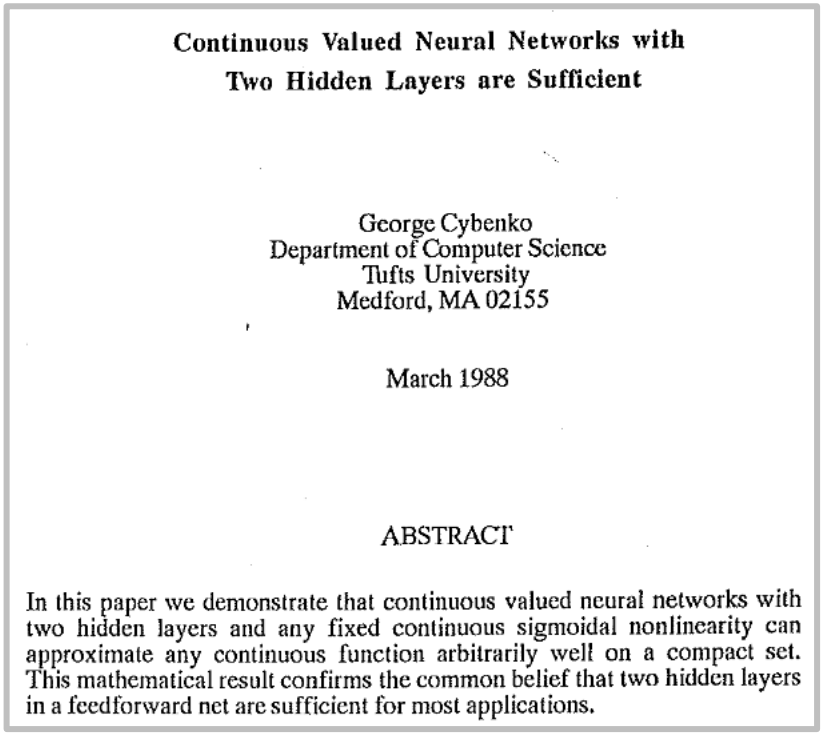
Today: building more powerful and expressive neural networks.

What is expressivity?

- *Theoretical:* what functions can a neural network represent?
- *Practical*: what functions does a neural network prefer to represent?

A classic result on the expressivity of neural nets

Classic result: ‘neural nets are universal function approximators’



Continuous Valued Neural Networks with Two Hidden Layers are Sufficient

George Cybenko
Department of Computer Science
Tufts University
Medford, MA 02155

March 1988

ABSTRACT

In this paper we demonstrate that continuous valued neural networks with two hidden layers and any fixed continuous sigmoidal nonlinearity can approximate any continuous function arbitrarily well on a compact set. This mathematical result confirms the common belief that two hidden layers in a feedforward net are sufficient for most applications.

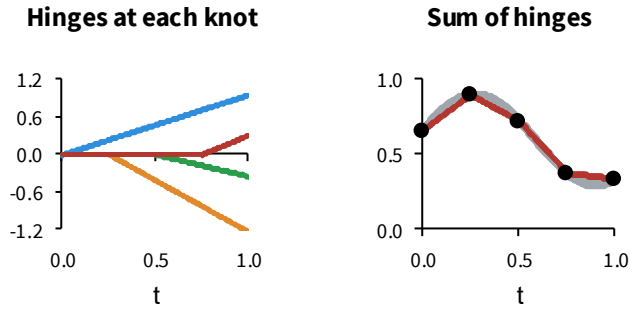
Universal approximation: one hidden layer is enough

We'll give a slightly different proof for the general (non polynomial) activations

For any σ that is not polynomial, $f(x) \approx \sum_{i=1}^{N} a_i \, \sigma(w_i^{\mathsf{T}} x + b_i)$ for every continuous f on a compact set

$$\hat{g}(t) = g(t_0) + s_0 \, \sigma(t - t_0) + \sum_{i=1}^{N-1} (s_i - s_{i-1}) \, \sigma(t - t_i)$$

- **Simple case of a ReLU:** decompose the space into a grid, add up and down slopes to get the individual ReLUs.
- **Why not a polynomial**: polynomial units only sum to polynomials of the same degree



Leshno et al. 1993; Pinkus 1999

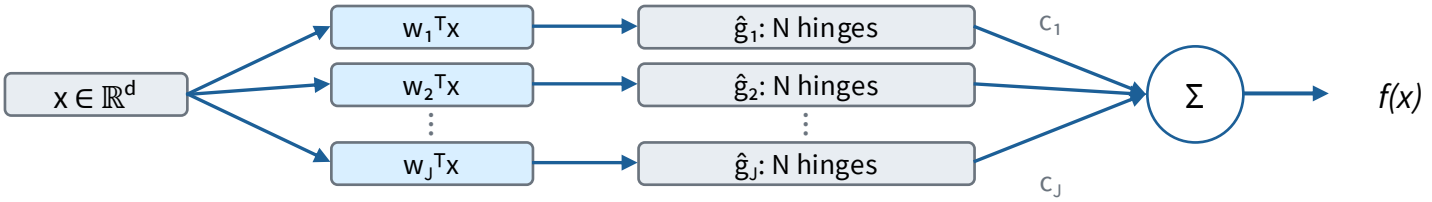
Step 2: from one input to d inputs

Write the multivariate input as a polynomial, each monomial power is approximated 1d

$$P(x) = \sum_{j=1}^{J} c_j \, (w_j^\top x)^{k_j} \quad \Longrightarrow \quad f(x) \approx \sum_{j=1}^{J} c_j \, \hat{g}_{k_j}(w_j^\top x)$$

- **Weierstrass approximation**: f is close to a polynomial P, and $P = \Sigma_j c_j (w_j^T x)^{kj}$
- **Replace each power** by its 1-D hinge interpolant $\hat{g}$

Other activations: sigmoids (Cybenko; no size), any non-polynomial σ via derivatives in w (Leshno)



Cybenko 1989; Hornik 1991; Leshno et al. 1993

Is the lecture over? no

Representability with (near) infinite units is not particularly interesting.

$$\text{grid construction: } (L/\varepsilon)^d \text{ units,}$$
$$\text{lower bound on } 1-\text{Lipschitz functions: } \gtrsim \varepsilon^{-d/2} \text{ weights}$$

The size might be very big! the grid needs $(L/\varepsilon)^d$ units this is exponentially bad in dimension, and for accurate approx, will blow up very badly

The representability may not actually appear in practice it may be very brittle (near measure zero), or it may be a very difficult to find minimum.

The rest of this lecture: a refined understanding of expressiveness.

Yarotsky 2017

Lecture today

Part 1
Depth vs width expressivity

Part 2
Realized expressivity

Part 3
Expressivity in practice

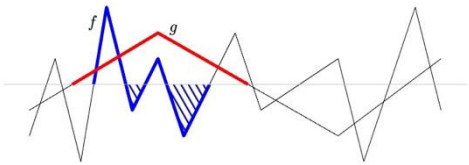
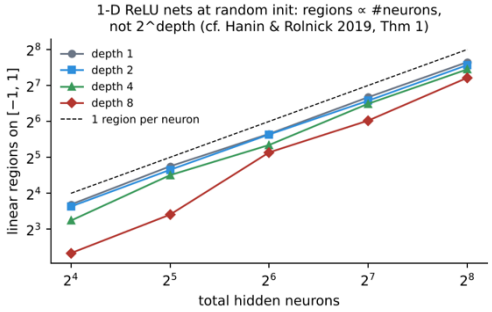
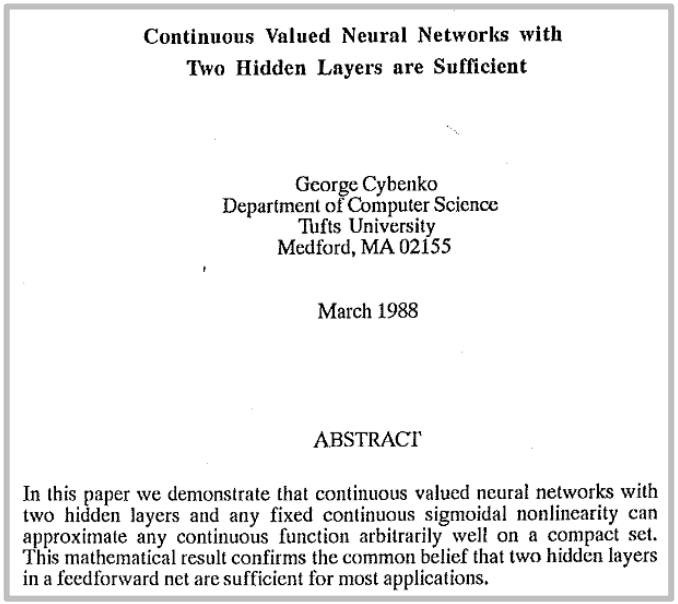


Figure 1: f crosses more than g.



	1	2	3	4	5	6	7	8	12	16
Dyck	99.3	100	100	100	100	100	100	100	100	100
Grid_9	92.2	100	100	100	100	100	100	100	100	100
C_2	77.6	99.8	99.9	100	100	99.5	100	99.7	100	100
C_3	54.6	94.6	96.7	99.4	100	100	99.8	100	100	100
C_2^3	65.0	77.9	99.9	97.9	100	99.8	98.2	99.9	95.9	80.6
D_6	25.4	27.2	47.4	75.2	100	100	100	100	100	100
D_8	45.6	98.0	100	100	100	100	100	100	100	100
Q_8	31.6	49.2	59.6	60.4	73.5	99.3	100	100	100	100
A_5	12.5	23.1	32.5	46.7	71.2	98.8	100	100	100	100
S_5	7.9	11.8	14.6	19.7	26.0	28.4	32.8	51.8	97.2	99.9

What should a theory of expressivity have?



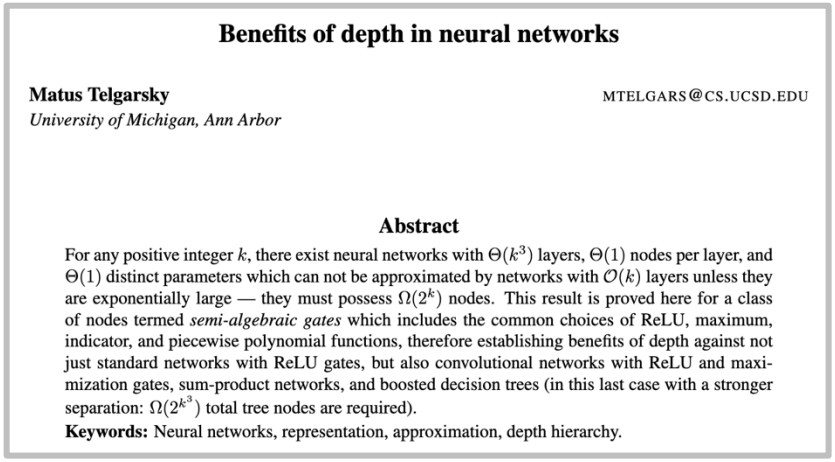
Continuous Valued Neural Networks with Two Hidden Layers are Sufficient

George Cybenko
Department of Computer Science
Tufts University
Medford, MA 02155

March 1988

ABSTRACT

In this paper we demonstrate that continuous valued neural networks with two hidden layers and any fixed continuous sigmoidal nonlinearity can approximate any continuous function arbitrarily well on a compact set. This mathematical result confirms the common belief that two hidden layers in a feedforward net are sufficient for most applications.



Benefits of depth in neural networks

Matus Telgarsky MTELGARS@CS.UCSD.EDU
University of Michigan, Ann Arbor

Abstract

For any positive integer k, there exist neural networks with $\Theta(k^3)$ layers, $\Theta(1)$ nodes per layer, and $\Theta(1)$ distinct parameters which can not be approximated by networks with $\mathcal{O}(k)$ layers unless they are exponentially large — they must possess $\Omega(2^k)$ nodes. This result is proved here for a class of nodes termed *semi-algebraic gates* which includes the common choices of ReLU, maximum, indicator, and piecewise polynomial functions, therefore establishing benefits of depth against not just standard networks with ReLU gates, but also convolutional networks with ReLU and maximization gates, sum-product networks, and boosted decision trees (in this last case with a stronger separation: $\Omega(2^{k^3})$ total tree nodes are required).

Keywords: Neural networks, representation, approximation, depth hierarchy.

Can we approximate it with infinite neurons?

Not useful – unrealistically strong
(and no useful prescriptions)

Which networks are *more* expressive?

Deep vs wide

Telgarsky's theorem 1: depth k^3 vs width 2^k

Consider a standard multilayer ReLU network

A shallow networks cannot match a deep one unless they are exponentially wide

$$\text{depth } 2k^3 + 8, \text{ width } 3 \quad \text{vs} \quad \text{depth} \leq k, \text{ width} < 2^k: \qquad \int_{[0,1]^d} |f - g| \, dx \geq \frac{1}{64}$$

Exponentially more neurons needed in the shallow model!

The gap is a constant fraction of the domain

Holds for any semi-algebraic gate (ReLU, max, piecewise polynomials)

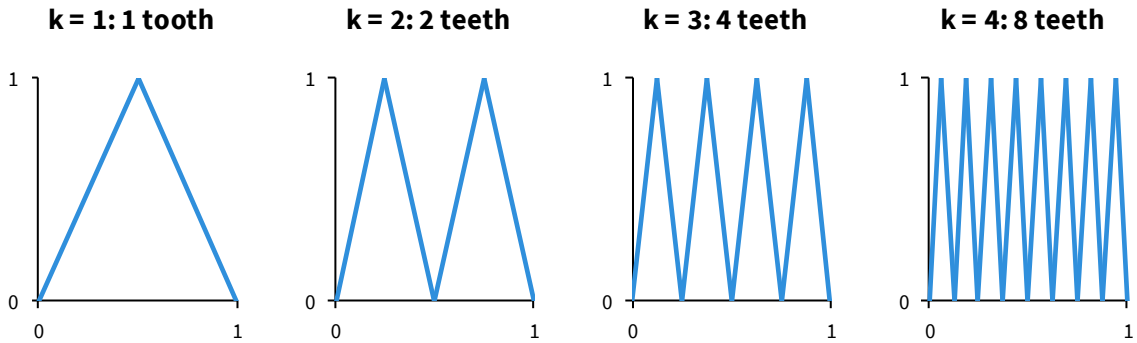
Telgarsky 2016

Proof: depth allows for 'high frequency' constructions

Two layers of ReLUs can fold [0, 1] onto itself

$$f(z) = \sigma(2\sigma(z) - 4\sigma(z - 0.5)) = \min(2z,\ 2 - 2z) \text{ on } [0,1]$$

$$f^k = f \circ f \circ \cdots \circ f: \quad 2^{k-1} \text{ teeth}, \quad 2^k \text{ pieces}, \quad 2k \text{ ReLUs}$$



This is hard for a shallow net to approximate!

- A one-hidden-layer net with w units is a sum of w hinges: at most w + 1 pieces on a line
- To match f^k it needs **$w \geq 2^k$**. Deep: 2k units. Wide: 2^k units

Telgarsky 2016

No shallow approximation is possible

Basic argument

- A function with few linear pieces crosses a horizontal line few times
- The sawtooth crosses ½ exactly 2^k times.
- If g crosses far fewer times than f, then on a constant fraction of the teeth g is on the wrong side, so ∫|f − g| is bounded below.

$$\mathrm{Cr}(g) \le 2\left(\frac{2m}{l}\right)^{l} \quad (l \text{ layers}, m \text{ ReLUs}) \qquad \mathrm{Cr}(f^k) = 2^k + 1$$

$$\int_0^1 |f^k - g| \;\ge\; 2^{k-2} \cdot \frac{2^{-k}}{4} = \frac{1}{16} \qquad \text{whenever } \mathrm{Cr}(g) \le 2^{k-2}$$

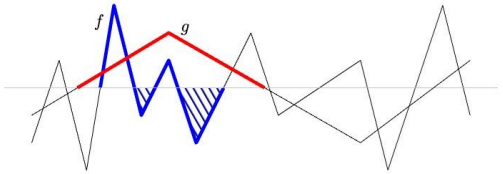
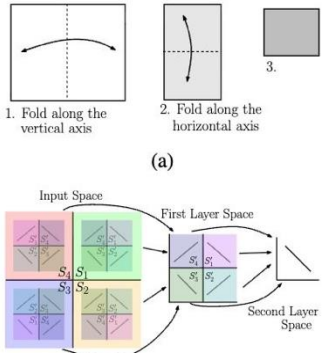
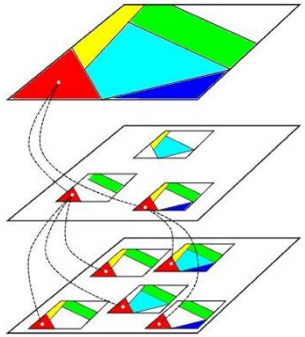


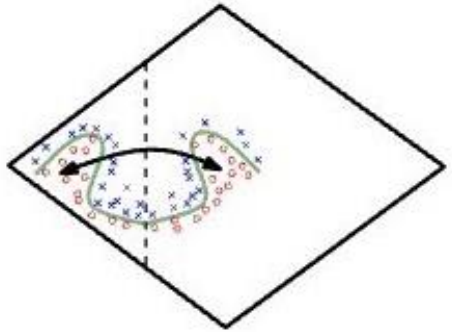
Figure 1: f crosses more than g.

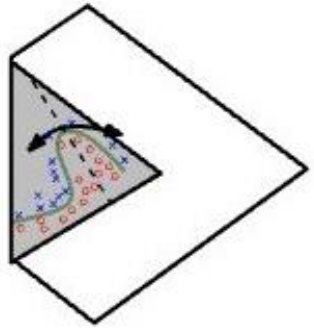
In d dimensions, similar folding arguments

There's a more general 'folding' mechanism that exponentially increases expressivity









$$\#\text{regions} = \Omega\left(\left(\frac{n}{n_0}\right)^{(L-1)\, n_0}\, n^{n_0}\right) \quad \text{vs} \quad O(N^{n_0}) \text{ for one layer of } N \text{ units}$$

L hidden layers, width n, input dim n0

Montúfar, Pascanu, Cho & Bengio 2014

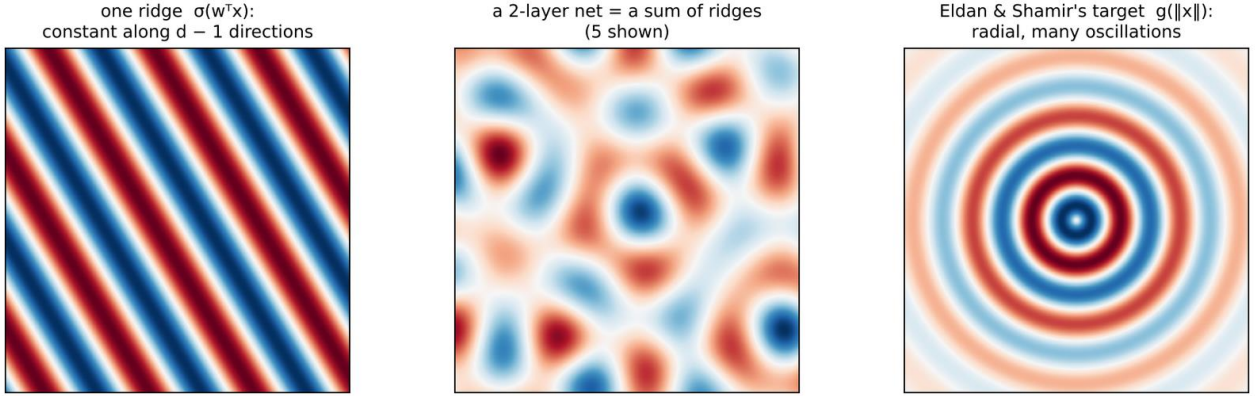
Exponential separation even between 2 and 3 layers

A radial function: easy with 3 layers, exponentially hard with 2

- A 2-layer net is a sum of ridges, no obvious radial symmetry. A 3 layer net can approximate this
- A radially symmetric sine/cosine requires exponentially many units in 2 layers, but not 3

$$\text{3 layers: width } \mathrm{poly}(d) \qquad \text{2 layers: width } \geq c\,e^{cd} \text{ to reach error } < c$$

Matching the radial oscillations with ridges takes exponentially many of them in d; a third layer does it with poly(d) width

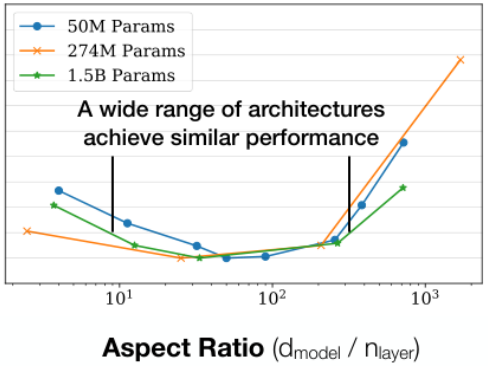


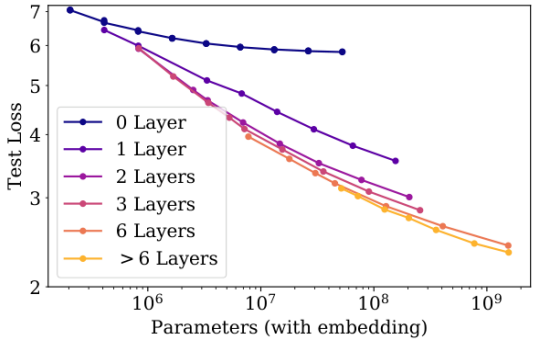
Eldan & Shamir 2016

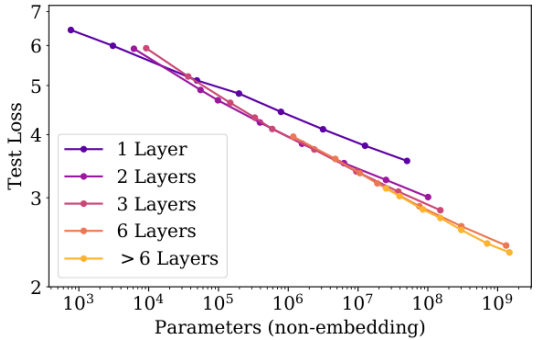
Depth vs width empirically

LM loss depends mostly on param count, not shape

- Can have very deep or very wide networks (see aspect ratio) without breaking the model
- Only truly destructive choice is 0 or 1 layer models







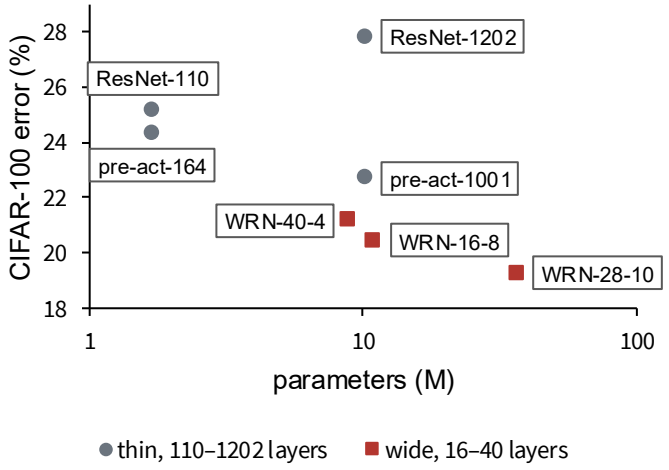
Kaplan et al. 2020

Depth vs width in vision

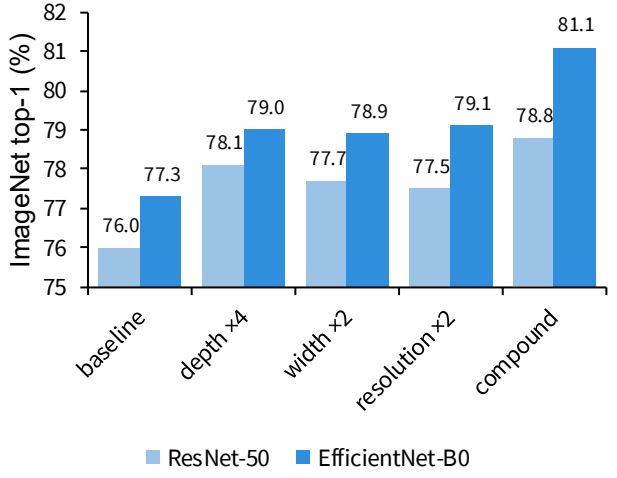
Generally, fixed aspect ratio scaling is better than depth or width alone

- **Wide ResNets**: 40 layers is better than 1001 at similar size
- **EfficientNet**: joint scaling ('compound') is better than either depth or width alone

Wide ResNets: 40 layers beat 1001



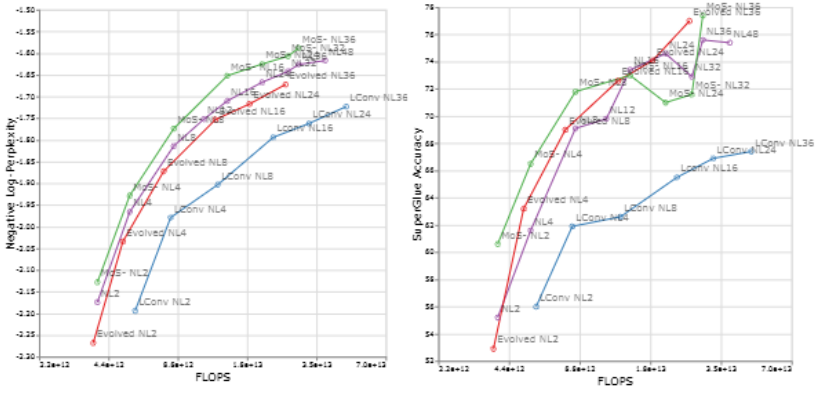
EfficientNet: at ~equal FLOPs



Zagoruyko & Komodakis 2016; Tan & Le 2019

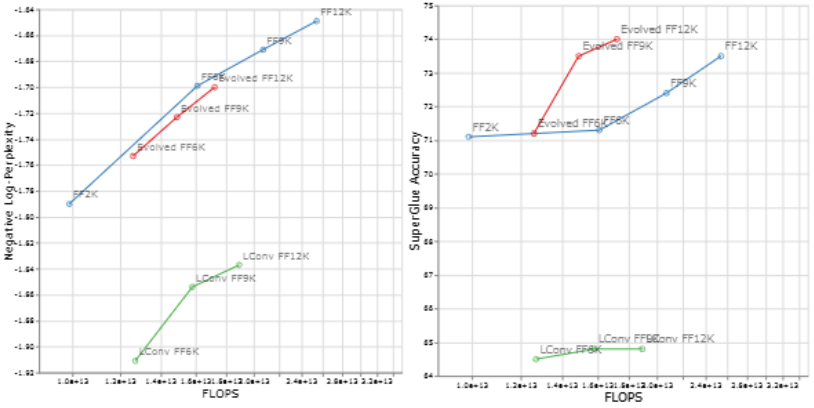
Depth vs width downstream

But depth may matter more in thinking about downstream performance



(a) Upstream Neg. Log-PPL. (b) Downstream Accuracy.

Figure 4: Scaling depth



(a) Upstream Neg. Log-PPL. (b) Downstream Accuracy.

Figure 5: Scaling width of FFN

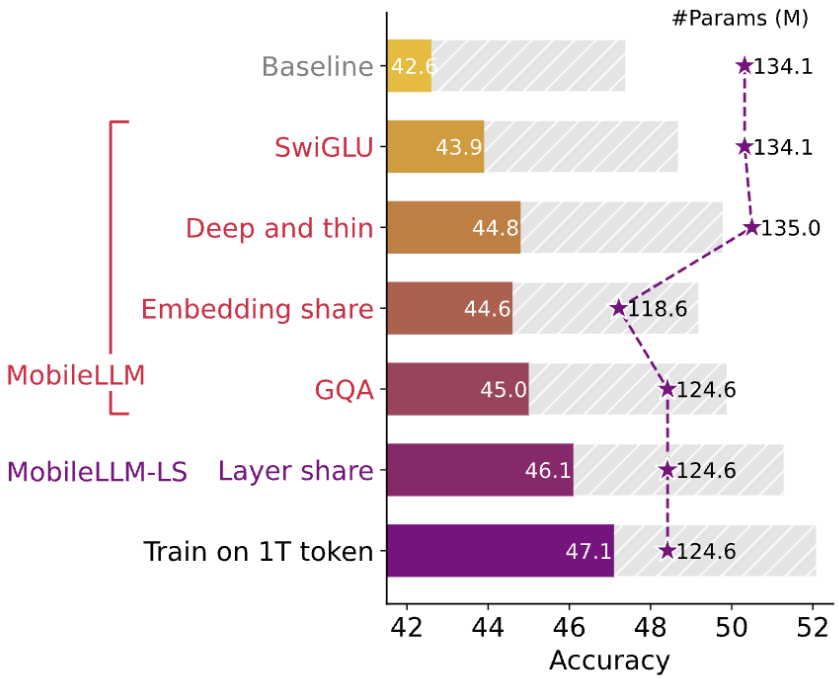
Tay et al 2022

Aside: systems concerns and architecture shapes

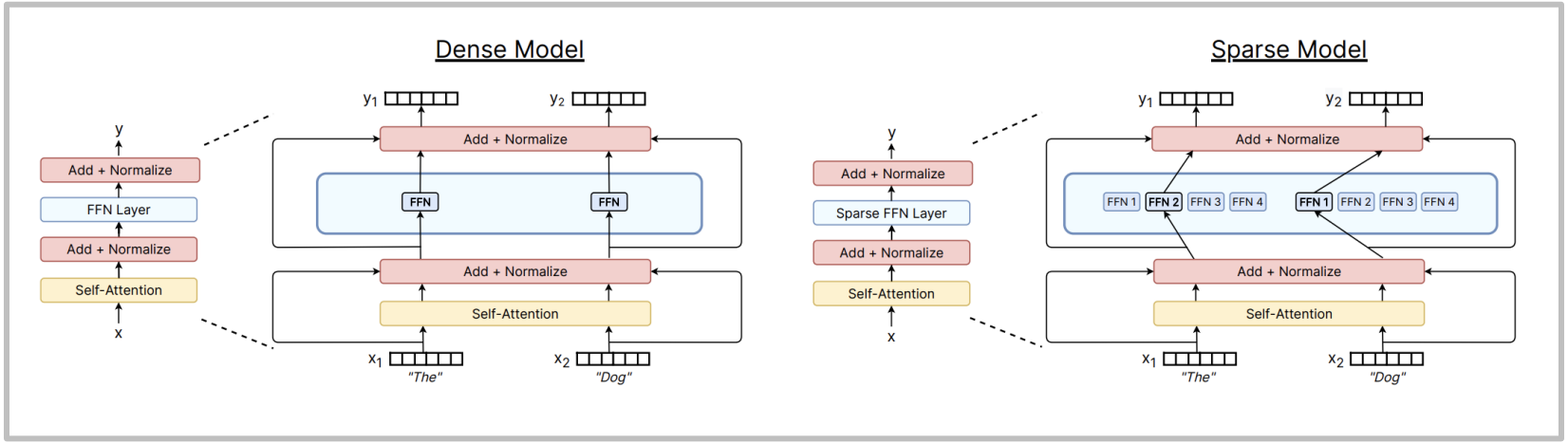
Systems concerns (and constraints) lead to very different optima

In MobileLLM (small, on device model)

- Very deep networks (minor)
- Share embeddings (save params)
- Repeat weights (looping)



Advanced: more parameters without cost



[Fedus et al 2022]

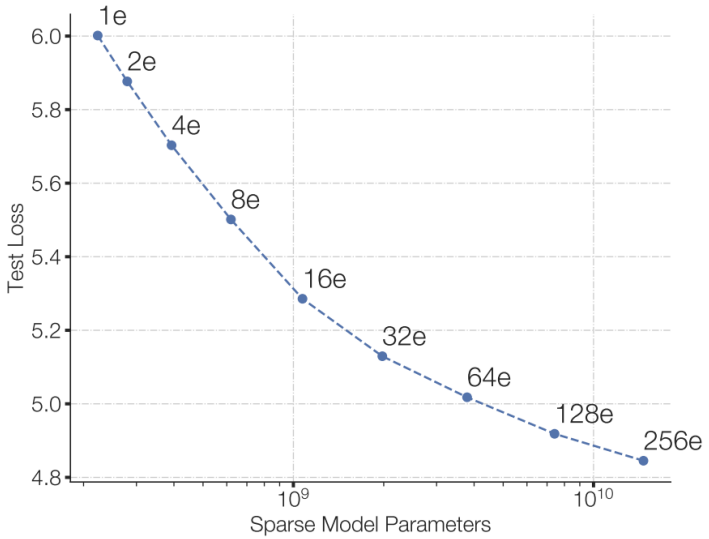
Replace big feedforward with (many) big feedforward networks and a selector layer

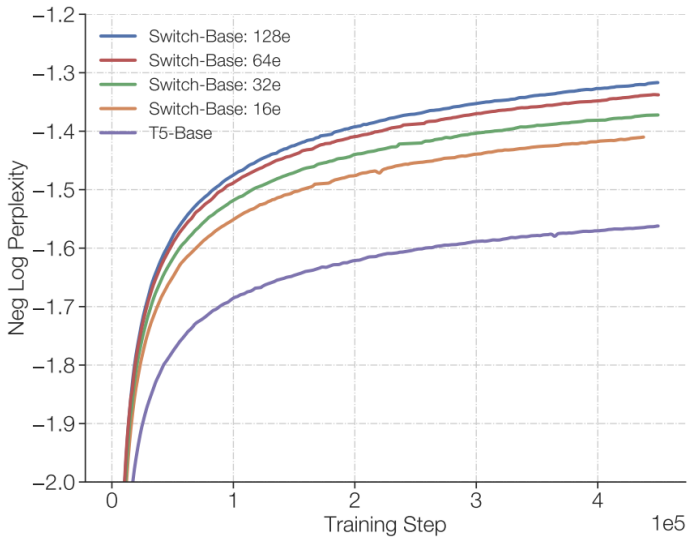
You can increase the # experts without affecting FLOPs

Why MoE?

Same total compute as a dense model, but can have way more inactive parameters

Significant compute and memory wins





[Fedus et al 2022]

Routing and sparse computation

How do we have ‘sparse layers’? Have a simple sparse gate as an inner product.

Gating

$$\mathbf{h}_t^l = \sum_{i=1}^{N} \left(g_{i,t} \, \mathrm{FFN}_i \left(\mathbf{u}_t^l \right) \right) + \mathbf{u}_t^l,$$

$$g_{i,t} = \begin{cases} s_{i,t}, & s_{i,t} \in \mathrm{Topk}(\{s_{j,t} | 1 \leqslant j \leqslant N\}, K), \\ 0, & \text{otherwise}, \end{cases}$$

$$s_{i,t} = \mathrm{Softmax}_i \left(\mathbf{u}_t^{l\,T} \mathbf{e}_i^l \right),$$

Gates selected by a logistic regressor

Used broadly across a large number of modern MoE based models, as well as other exotic ‘routing’ settings (h-nets)

[Dai et al 2024]

Heuristics to make experts evenly used

Systems efficiency requires even utilization of experts, but this is hard to optimize for

For each Switch layer, this auxiliary loss is added to the total model loss during training. Given N experts indexed by $i = 1$ to N and a batch $\mathcal{B}$ with T tokens, the auxiliary loss is computed as the scaled dot-product between vectors f and P,

$$\text{loss} = \alpha \cdot N \cdot \sum_{i=1}^{N} f_i \cdot P_i \tag{4}$$

where f_i is the fraction of tokens dispatched to expert i,

$$f_i = \frac{1}{T} \sum_{x \in \mathcal{B}} \mathbb{1}\{\text{argmax } p(x) = i\} \tag{5}$$

and P_i is the fraction of the router probability allocated for expert i, [2]

$$P_i = \frac{1}{T} \sum_{x \in \mathcal{B}} p_i(x). \tag{6}$$

From the Switch Transformer [Fedus et al 2022]

The derivative with respect to $p_i(x)$ is $\frac{\alpha N}{T^2} \sum 1_{argmax\ p(x)=i}$,
so more frequent use = stronger downweighting

Summary

Beyond universal approximation
We can prove when deeper models have provable advantage over shallow

The nature of depth advantage
Constructions rely on ‘oscillatory’ structures that are hard to capture.

Depth advantage in practice
In practice, what matters is the parameter count more than depth

Getting more parameters
total params and not just active ones are important

Lecture today

Part 1
Depth vs width expressivity

Part 2
Realized expressivity

Part 3
Expressivity in practice

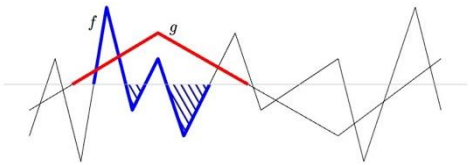
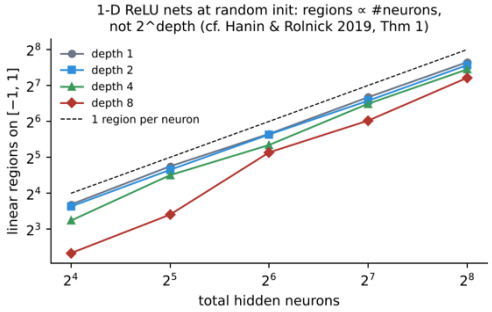


Figure 1: f crosses more than g.



	1	2	3	4	5	6	7	8	12	16
Dyck	99.3	100	100	100	100	100	100	100	100	100
Grid_9	92.2	100	100	100	100	100	100	100	100	100
C_2	77.6	99.8	99.9	100	100	99.5	100	99.7	100	100
C_3	54.6	94.6	96.7	99.4	100	100	99.8	100	100	100
C_2^3	65.0	77.9	99.9	97.9	100	99.8	98.2	99.9	95.9	80.6
D_6	25.4	27.2	47.4	75.2	100	100	100	100	100	100
D_8	45.6	98.0	100	100	100	100	100	100	100	100
Q_8	31.6	49.2	59.6	60.4	73.5	99.3	100	100	100	100
A_5	12.5	23.1	32.5	46.7	71.2	98.8	100	100	100	100
S_5	7.9	11.8	14.6	19.7	26.0	28.4	32.8	51.8	97.2	99.9

Why don’t we see exponential depth advantage?

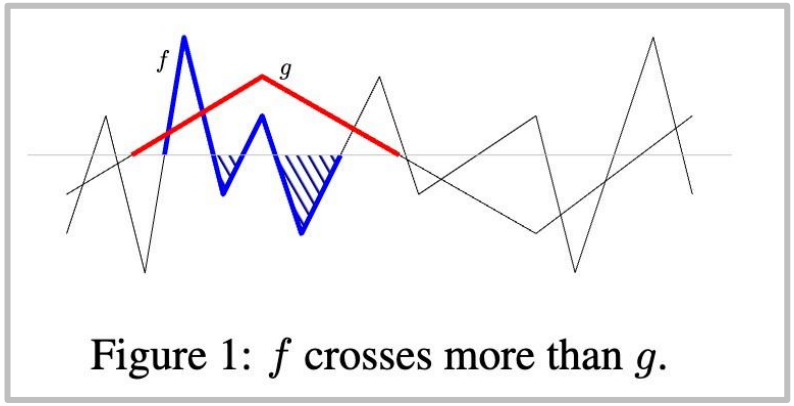
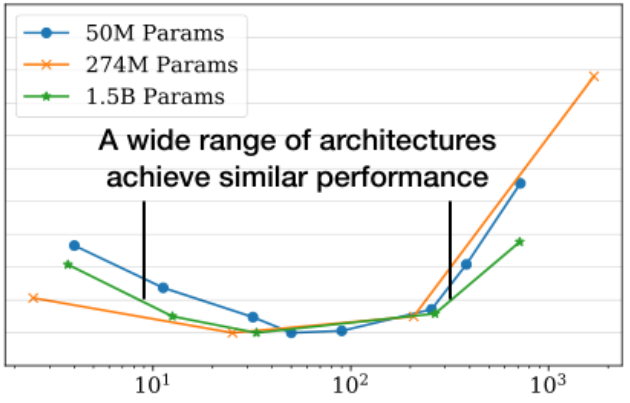


Figure 1: f crosses more than g.

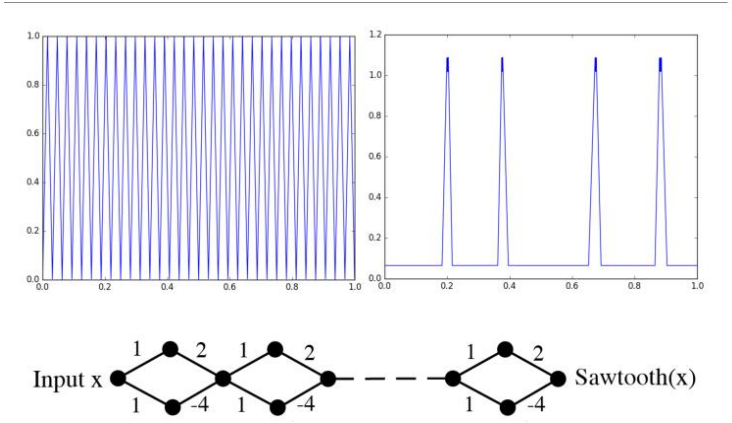
The theory: exponential benefits to depth over width



In practice: relatively low dependence on depth

The sawtooth construction is fragile

From Hanin and Rolnick: adding a bit of noise collapses most of the sawtooth



Why?: Each layer has slope ± 2, so noise at layer j is amplified 2^{k-j} times

$$\sup_x |f_\varepsilon^k(x) - f^k(x)| \approx 2^k\sigma \quad \Longrightarrow \quad \sigma \lesssim 2^{-k}$$ is the max allowable noise to retain sawtooth

Hanin & Rolnick 2019

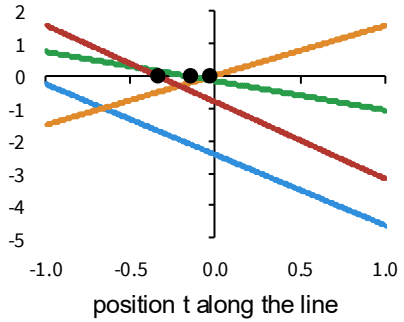
Let's count the complexity of models *in practice*

We can count the complexity of a network in a 1d-slice of the input space

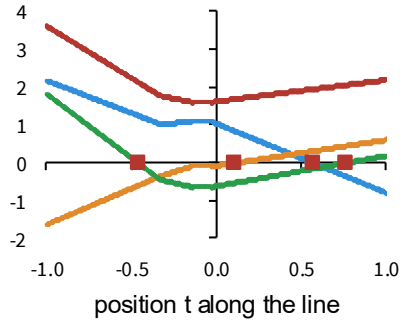
Consider only a 1d-slice of the input space $x = vt + c$. How 'rich' can the network be?

- On the first layer, find the 'zeros' of $Wvt + b$, which is the piecewise linear boundary
- On future layers, each interval is linear, so solve for zeros within the interval

Layer 1: affine in t; zeros = 3 breakpoints



Layer 2: affine on each piece; 4 new zeros



8 regions on the line (exact)



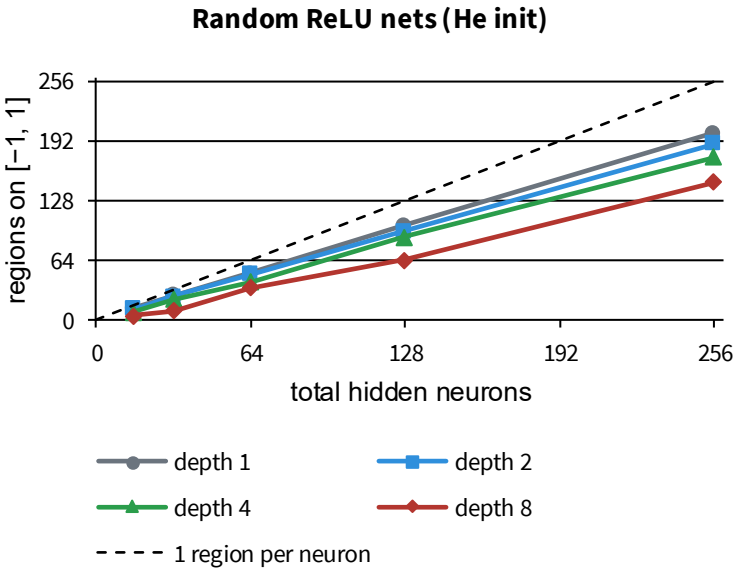
Hanin & Rolnick 2019

Model complexity at init (for a 1d slice)

Along any line, the number of regions grows with neurons

$$\mathbb{E}\,\#\{\text{regions along a line segment } I\} \;\approx\; |I| \cdot \#\text{neurons} \cdot C_{\text{grad}}\, C_{\text{bias}}$$

- For a 1 to 8 layer network, number of distinct regions grows *linearly* with the number of total neurons
- More consistent with the empirical 'total neurons matter' theory than the sawtooth constructions for depth vs width

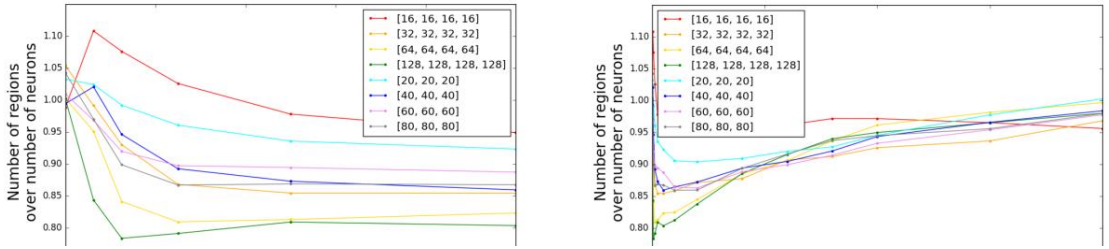


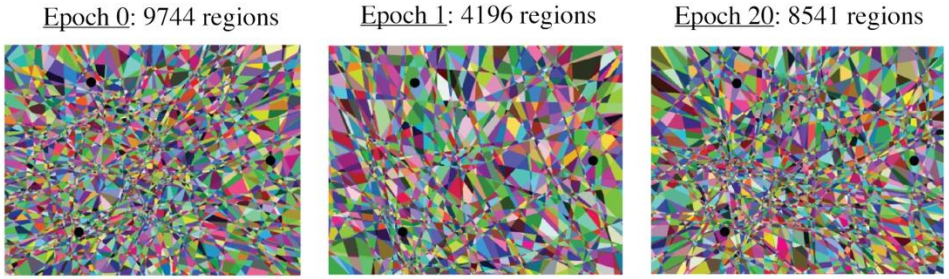
Hanin & Rolnick 2019

Complexity after training resembles those at init

The complexity during 20 epochs of MNIST training is generally stable

- Initial drop, and then stabilization to near-init values, no depth exponentials





(2d version of the 1d count method)

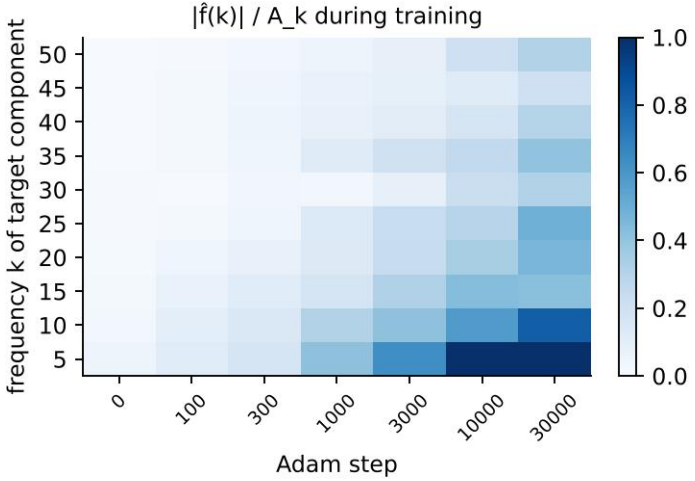
Hanin & Rolnick 2019

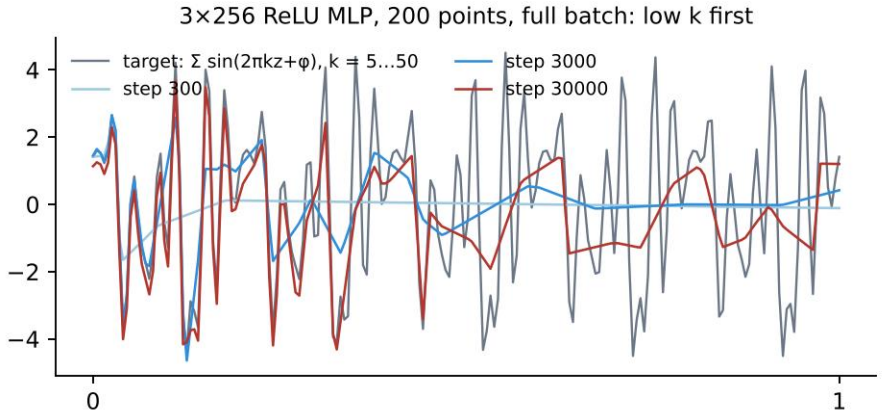
Other observations: low frequency bias of deep nets

In a mixture of different frequencies, low frequency is modeled first

High frequencies need many regions or large weights;

- At init, relatively few regions
- GD grows the weights slowly.





Rahaman et al. 2019

Summary

Complexity in theory: exponential in depth
Complexity in practice (?): linear in neurons

Main observation is that the exponential complexity is very brittle!

- Any noise would propagate exponentially and break the sawtooth

- Learning high frequency functions require very large norms / gradients

Lecture today

Part 1
Depth vs width expressivity

Part 2
Realized expressivity

Part 3
Expressivity in practice

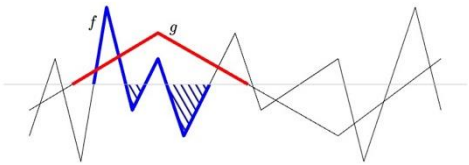
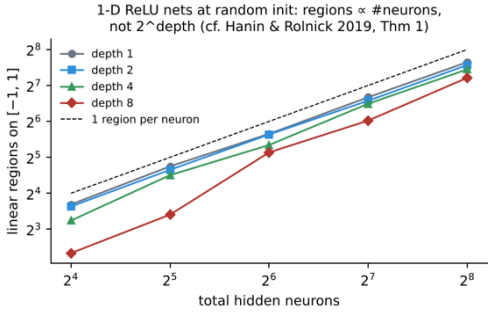


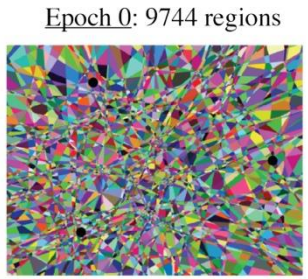
Figure 1: f crosses more than g.



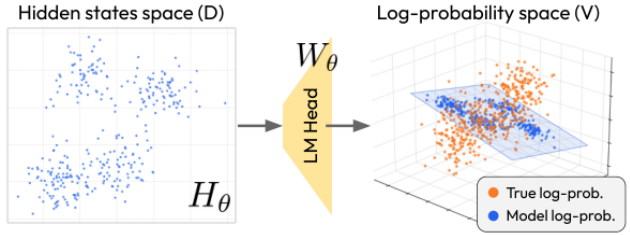
	1	2	3	4	5	6	7	8	12	16
Dyck	99.3	100	100	100	100	100	100	100	100	100
Grid_9	92.2	100	100	100	100	100	100	100	100	100
C_2	77.6	99.8	99.9	100	100	99.5	100	99.7	100	100
C_3	54.6	94.6	96.7	99.4	100	100	99.8	100	100	100
C_2^3	65.0	77.9	99.9	97.9	100	99.8	98.2	99.9	95.9	80.6
D_6	25.4	27.2	47.4	75.2	100	100	100	100	100	100
D_8	45.6	98.0	100	100	100	100	100	100	100	100
Q_8	31.6	49.2	59.6	60.4	73.5	99.3	100	100	100	100
A_5	12.5	23.1	32.5	46.7	71.2	98.8	100	100	100	100
S_5	7.9	11.8	14.6	19.7	26.0	28.4	32.8	51.8	97.2	99.9

Expressivity in transformers

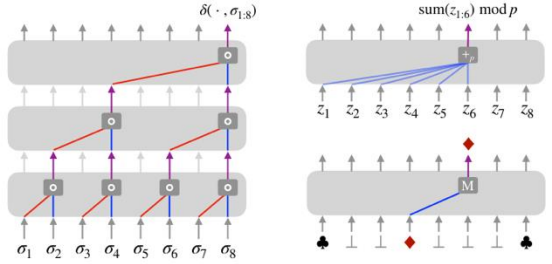
A more concrete set of examples..



Stacked ReLU networks

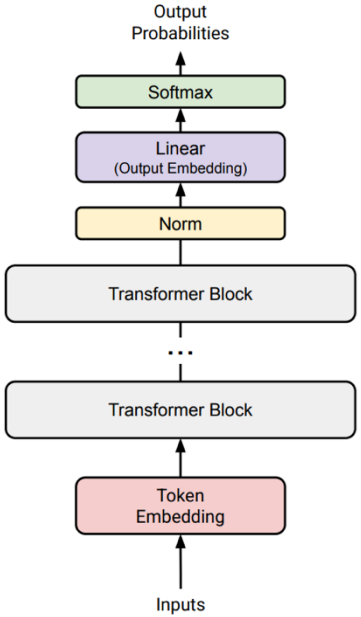


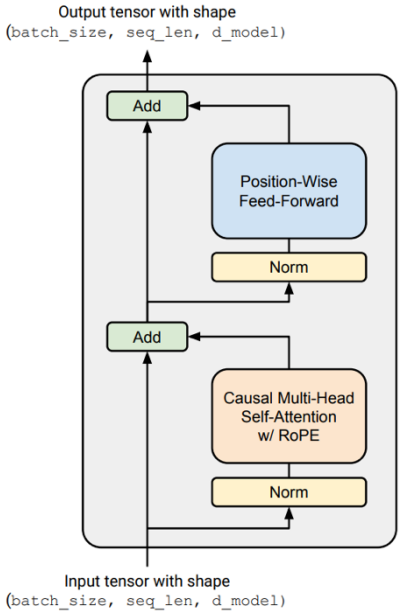
Architecture constraints



Temporal dependence

Bottlenecks in transformers..





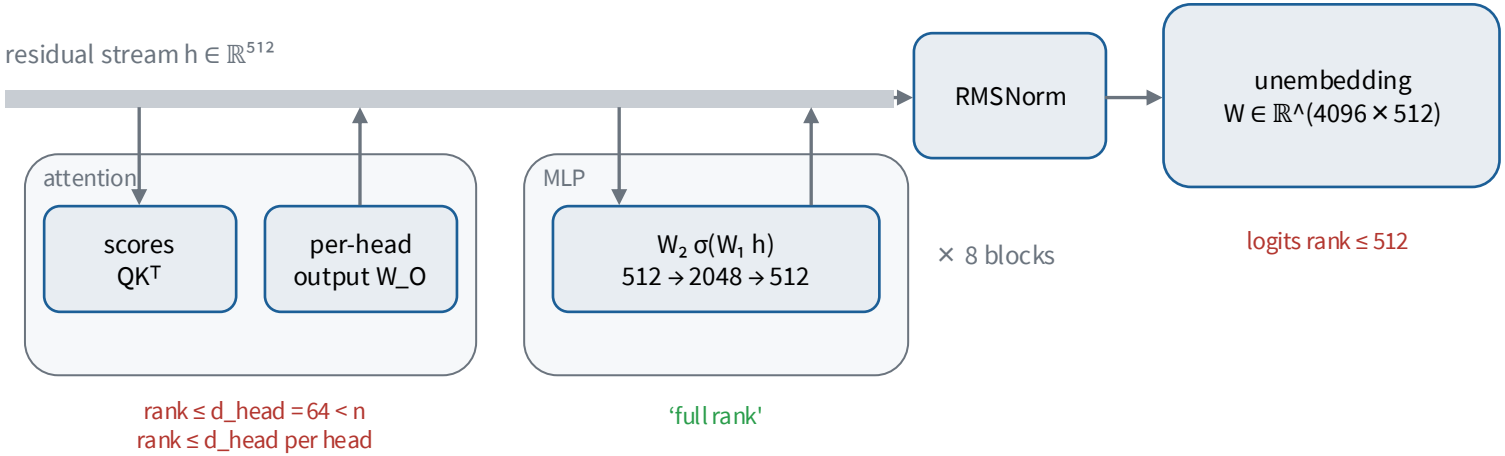
What are limitations of transformers?

- **Residual** – blocks can incrementally transform the residual stream
- **Embedding** – limited, but can be transformed by later layers.
- **Output embedding** - .. ?

A closer look at the transformer matmuls

What are the matmuls? Lets look at the dimensions (for d8)..

- Attention scores QK^T: rank ≤ d_head = 64 < n
- Readout: logits rank ≤ 512, with a 4096-word output vocab



Bhojanapalli et al. 2020

The softmax bottleneck (2018)

The log probability matrix of a transformer *cannot have rank greater than d+1*

$$\text{rank}(HW^\top) \leq d \;\Rightarrow\; \text{rank}(\underbrace{HW^\top - \log Z\,\mathbf{1}^\top}_{\text{logprobs}}) \leq d + 1$$

The most consequential bottleneck is at the output

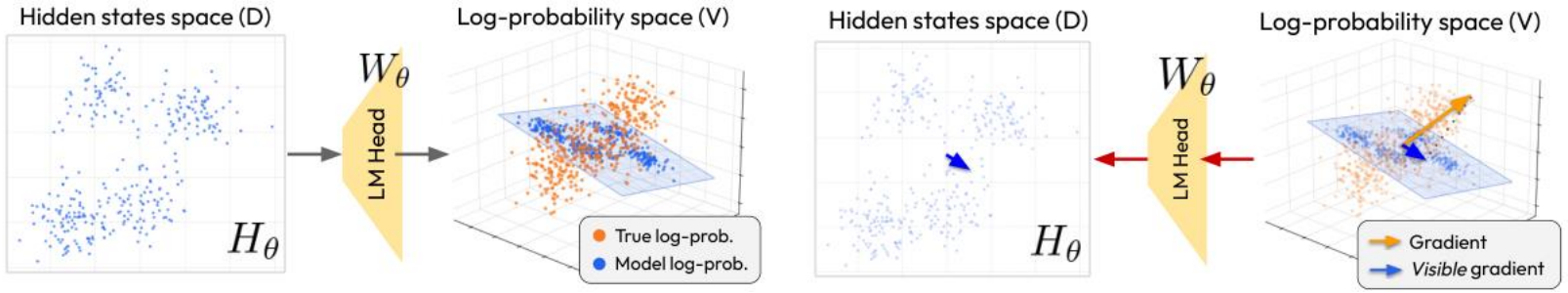
- There is a single output matmul (output unembed)
- At the end of the model, so no way to ‘increase’ expressiveness after
- This matmul is low rank (d << vocab)

Yang, Dai, Salakhutdinov & Cohen 2018, Kanai et al. 2018

A representation bottleneck is a gradient bottleneck

Gradients are projected onto a d-dimensional subspace in the backward pass

95–99% of the gradient norm is lost in GPT-2, Pythia, Llama-3 and Qwen-3



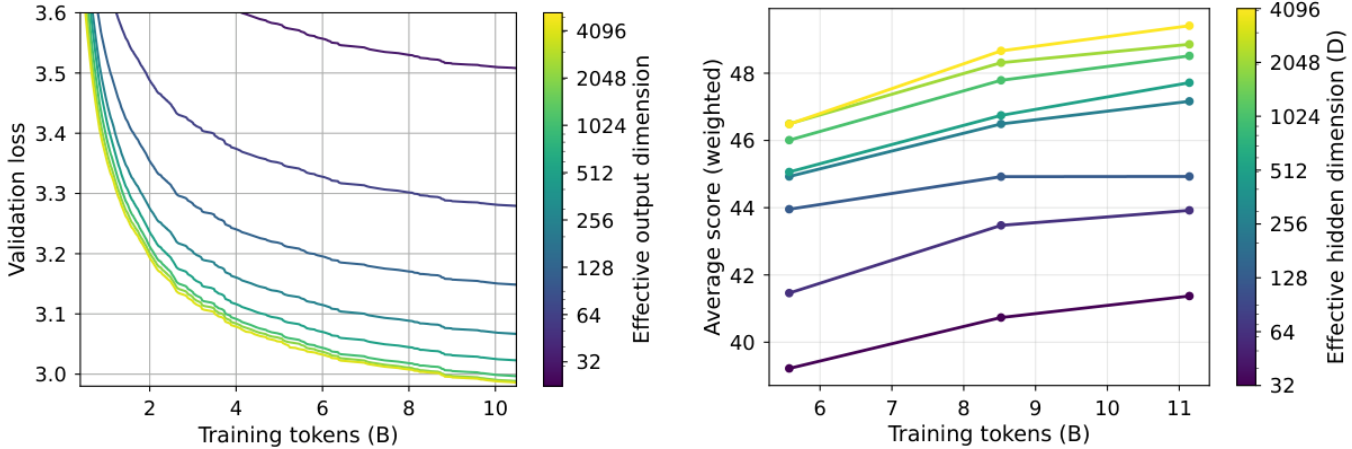
This can cause problems for learning the entire network below the bottleneck, not just impact the top LM head part

Godey & Artzi 2026

Impacts of changing the head rank

2B-scale models, head factored to rank D = 32 … 4096: higher rank trains much faster

- Dramatic gaps between low and high rank (16x faster)



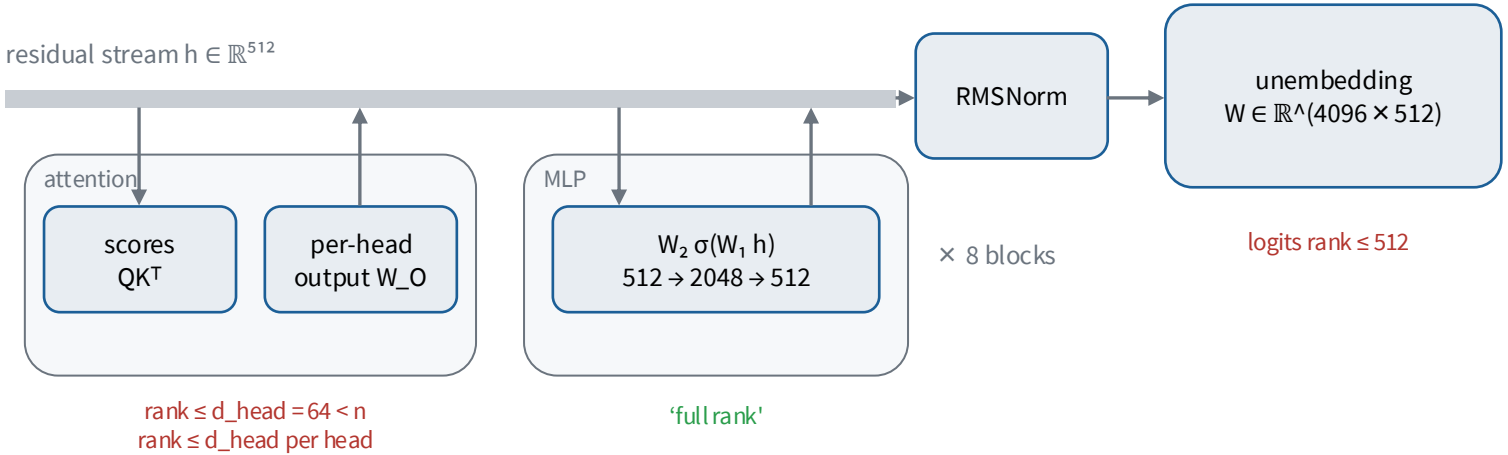
Godey & Artzi 2026

The other bottleneck..

Attention scores are also low rank

But, they happen in the middle of the network, and there are more copies

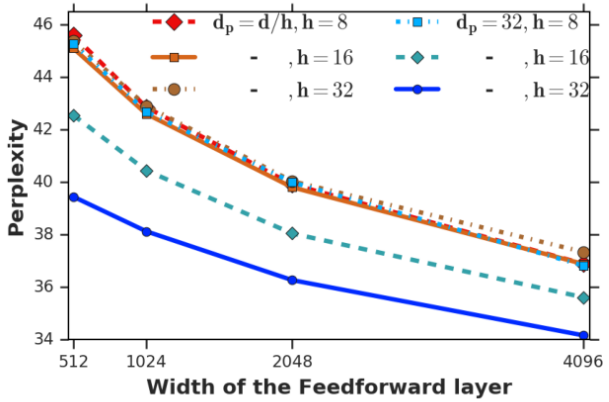
Will there be an impact?

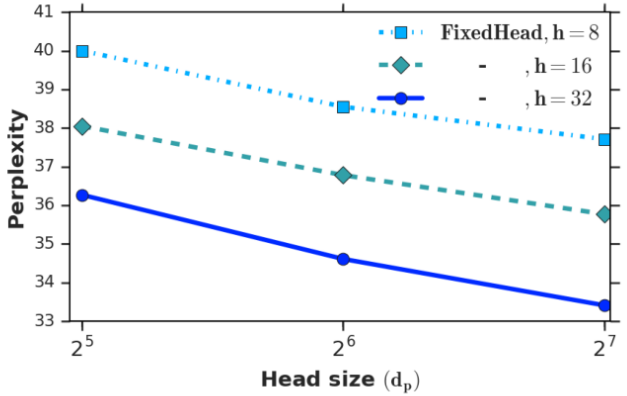


Bhojanapalli et al. 2020

The low-rank bottleneck in attention

A small d_head (< seq len) constrains the possible attention patterns





SQuAD F1 (BERT-large)	8 heads	12 heads	16 heads	32 heads
standard: d = 1024, d_head = d/h (336M)	90.89	—	90.61	90.45
fixed d_head = 128, d = 512	89.6 (168M)	90.25 (193M)	90.43 (218M)	90.95 (319M)

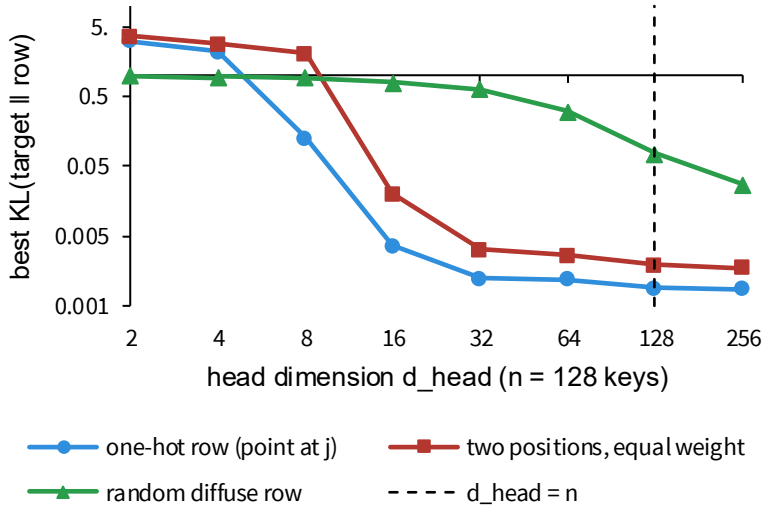
Bhojanapalli et al. 2020

Understanding the low rank attention bottleneck

Retrieval is easy with low rank, but random, diffuse patterns are harder.

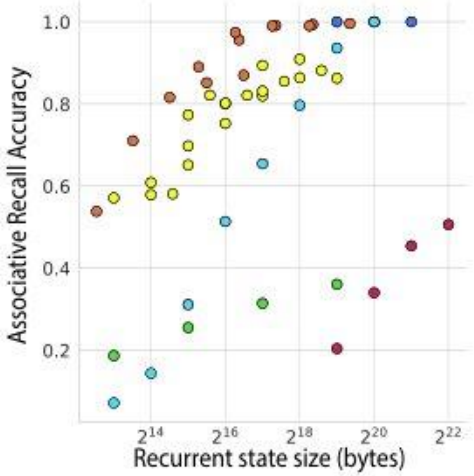
- **One-hot row**: reachable with d_head ≈ 16 ≪ n = 128
- **Random diffuse row**: needs d_head ≥ n

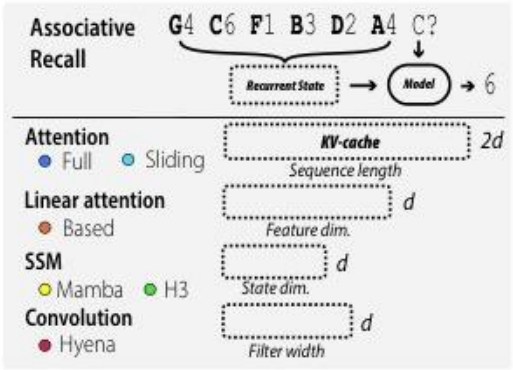
- This may be why the impacts are some what muted. Many point retrieval tasks are easy even with low rank



Sequence dependence beyond one step

Lets look back one lecture. We thought about the capacity and power of SSMs..



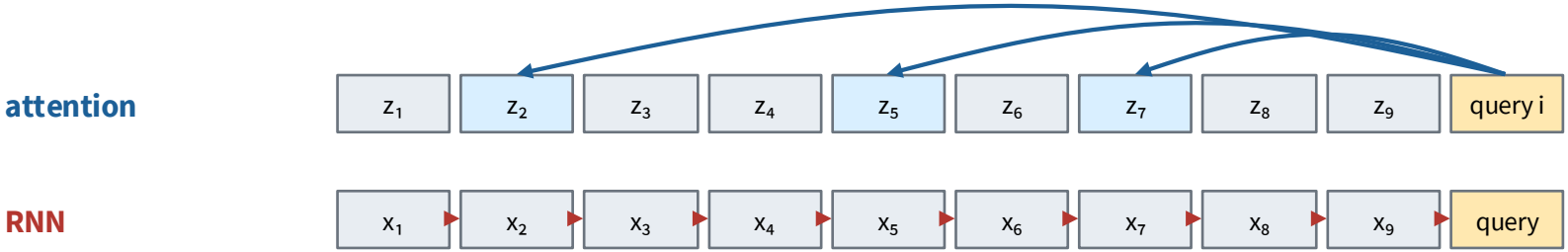


What's the (theoretical) expressiveness story for SSMs and Transformers?

Capacity bottlenecks in RNNs/SSM

Averaging q numbers identified by a 'query position' (q-sparse average)

- **Attention** requires rank ≈ q log N. **RNN** needs Ω(N) bits of hidden state.



Repeat a string of L tokens

- **Attention** can do direct lookup, but **RNNs** require full storage

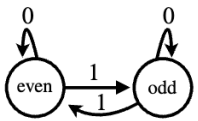


Jelassi, Brandfonbrener, Kakade & Malach 2024 , Sanford, Hsu & Telgarsky 2023

Expressivity in ‘naturally sequential’ tasks

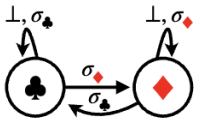
Maybe those tasks are unfair to the SSM. What about ‘naturally sequential’ tasks?

parity counter



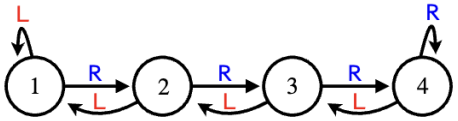
$Q = \{\text{even}, \text{odd}\}$
$\Sigma = \{0, 1\}$

memory unit

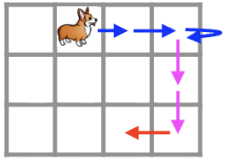


$Q = \{\clubsuit, \blacklozenge\}$
$\Sigma = \{\sigma_{\clubsuit}, \sigma_{\blacklozenge}, \perp\}$

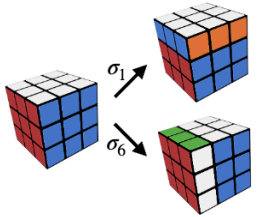
1D gridworld



$Q = \{1, 2, 3, 4\}$
$\Sigma = \{\text{L}, \text{R}\}$



$Q = \{1..3\} \times \{1..4\}$
$\Sigma = \{\leftarrow, \rightarrow, \uparrow, \downarrow\}$



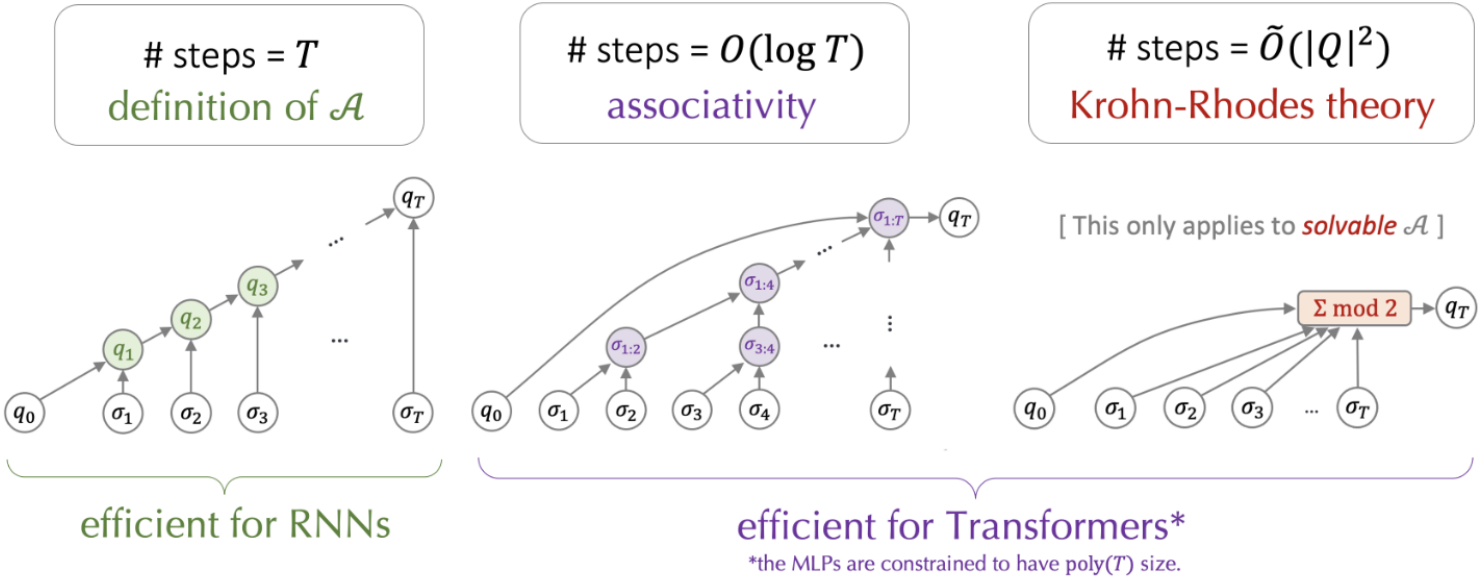
$Q = \{54 \text{ stickers}\}$
$\Sigma = \{6 \text{ face rotations}\}$

Example: parity counting using a finite state automata.

Given the inputs (e.g. a bitstring), how many steps do we need to compute the answer?

Low depth shortcuts

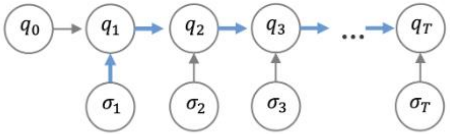
Even for such sequential tasks, transformers can do low-depth *approximations*



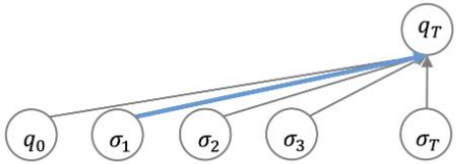
Suggests that even for ‘naturally sequential’ automata, transformers can model them well

Liu, Ash, Goel, Krishnamurthy & Zhang 2023

Example: parity computation



(a) the iterative solution $q_t = \delta(q_{t-1}, \sigma_t) = q_{t-1} \oplus \sigma_t$



(b) the parallel solution $q_t = \left(\sum_{\tau \le t} \sigma_\tau\right) \bmod 2$

Computing parity by flipping bits

Count and do mod 2

More generally, for any *finite* automata, we can find a similar ‘low depth’ solution.

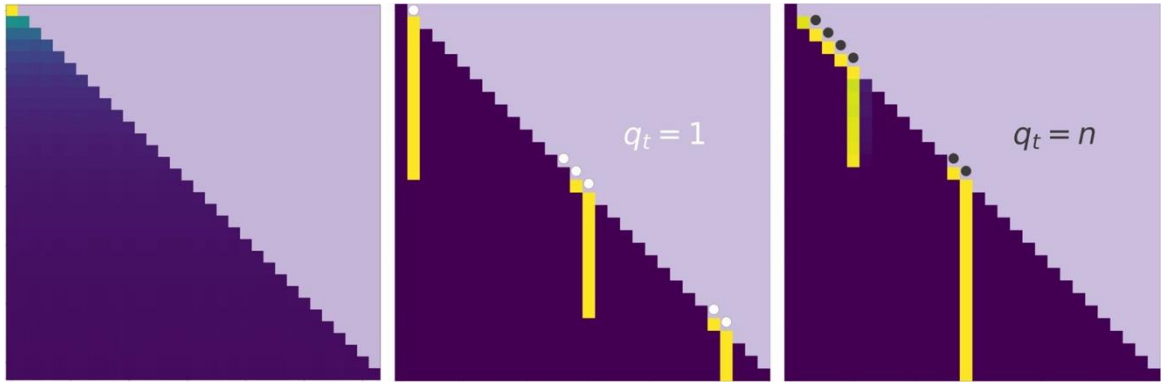
Key idea – every automata can be ‘represented’ as a sequence of matmuls, and associativity of matmuls means we can group them into a tree

Do transformers actually learn ‘shortcut’ low depth solns?

On T=100 long sequences, of the 19 different semiautomata tested

- 99% in-distribution accuracy on the task, with depths << T (left)
- In the gridworld automata (right), we expect uniform sum at the bottom, left boundary attention, then right boundary attention.

	1	2	3	4	5	6	7	8	12	16
Dyck	99.3	100	100	100	100	100	100	100	100	100
Grid_9	92.2	100	100	100	100	100	100	100	100	100
C_2	77.6	99.8	99.9	100	100	99.5	100	99.7	100	100
C_3	54.6	94.6	96.7	99.4	100	100	99.8	100	100	100
C_2^2	65.0	77.9	99.9	97.9	100	99.8	98.2	99.9	95.9	80.6
D_6	25.4	27.2	47.4	75.2	100	100	100	100	100	100
D_8	45.6	98.0	100	100	100	100	100	100	100	100
Q_8	31.6	49.2	59.6	60.4	73.5	99.3	100	100	100	100
A_5	12.5	23.1	32.5	46.7	71.2	98.8	100	100	100	100
S_5	7.9	11.8	14.6	19.7	26.0	28.4	32.8	51.8	97.2	99.9

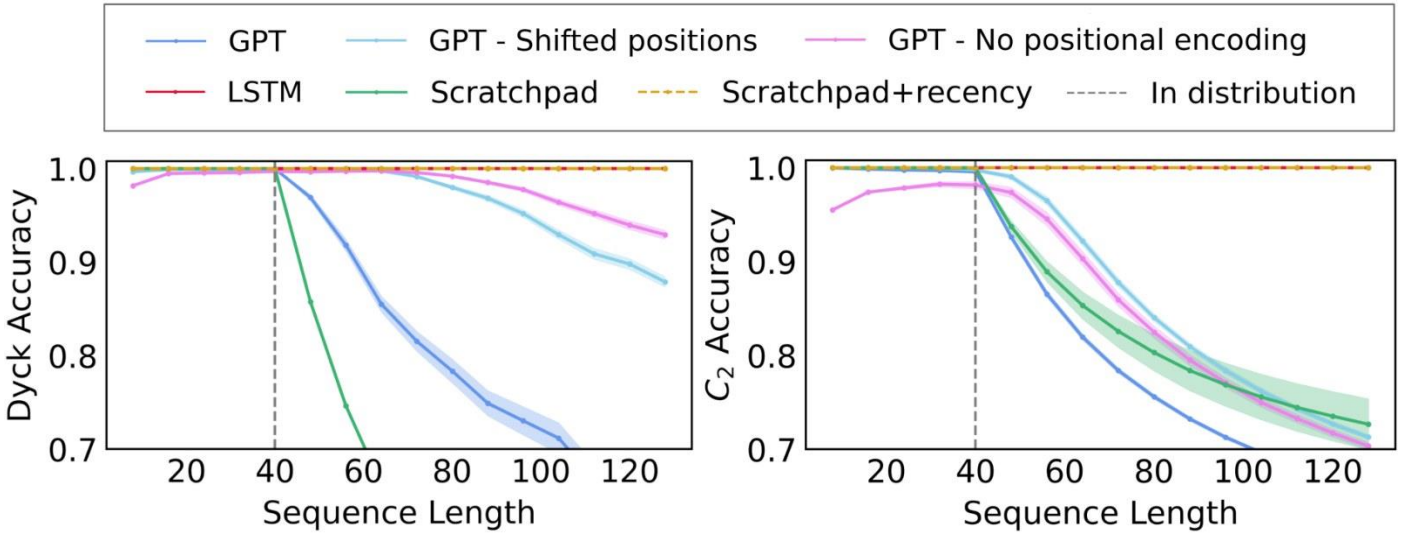


Liu et al. 2023, Figs. 3a, 4

… but these strategies are brittle

Parity by counting fails beyond the training length for transformers

- Shortcut solutions make use of the training horizon, which may not generalize
- Recurrent models do generalize beyond the training duration, which can be beneficial

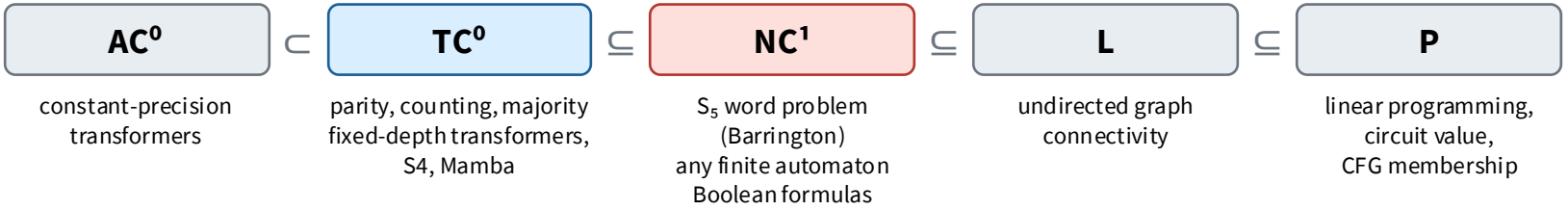


Liu et al. 2023

Hard limits to sequence expressibility

There are precise complexity hierarchies in what sequence models can do

- **Constant precision:** can't even do parity (AC0)
- **Fixed depth transformer:** can do parity (TC0), but not all finite automata (if TC0 ≠ NC1)
- **Nonlinear RNN, growing depth transformer:** can do finite automata
- **Graph connectivity**: log(n) depth suffices
- **Getting to all of P**: poly(n) depth can, polylog depth cannot unless P=NC

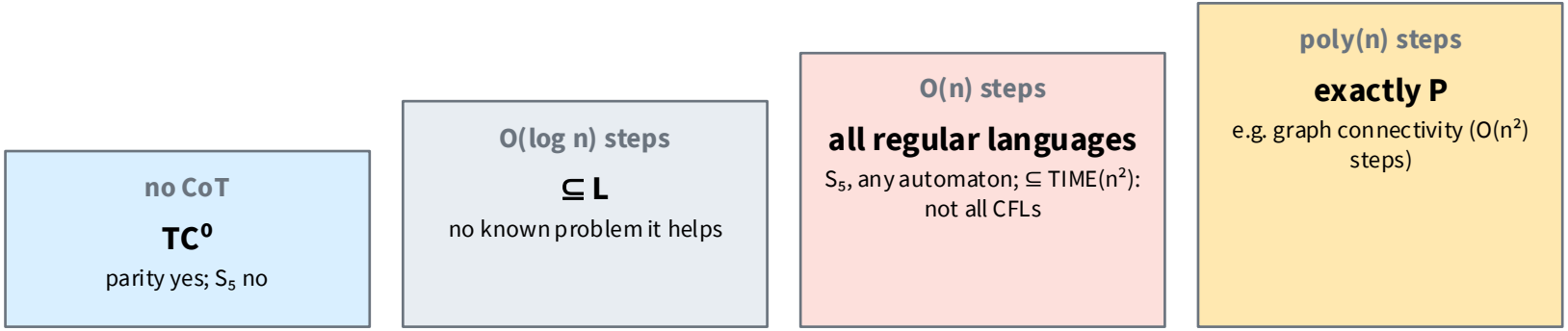


Merrill & Sabharwal 2023 and Merrill, Petty & Sabharwal 2024

Chain of thought can substitute for depth

Chain of thought produces n extra tokens before producing an answer

- This pushes expressivity up as if we had depth
- With n extra tokens, can do all regular languages
- With poly(n) steps, can even do all of P (substitutes for depth)



Merrill & Sabharwal 2024 and Li, Liu, Zhou & Ma 2024

Recap

Theoretical expressivity

- Strong benefits to depth in generic (Telgarsky) and sequence (Merrill)
- Transformers can approximate sequential computation cleanly

Practical expressivity

- LMs do not use all of this exponential expressivity in practice
- Many of the constructions are very brittle and sensitive

Transformer specific

- Low rank bottlenecks in softmax + attention