NLP & Transformers

“Abstract cubist art representing the Transformer multi-head inner product query-key-value self-attention neural network architecture”, according to a vision-language model (DALL-E 2)

Cyril Zhang
Microsoft Research NYC

ML, NLP, and "AGI"

- **This course:** foundations of algorithmic prediction
- **One weird prediction problem:** $\widehat{\Pr}_\theta[\text{next word} \mid \text{previous words}]$

- Statistical LM
- Parsing
- Transduction
- Knowledge retrieval
- Symbolic reasoning

`I know you are, but what am _____`

`In "AI can write", the verb phrase is: _____`

`English: "pigeons can't swim". French: _____`

`Shakespeare's 99th sonnet begins with: _____`

`Python code that finds the n²-th prime: _____`

- **2020-present:** all these capabilities within one system

“Sparks of AGI”

- GPT-4 demonstrates capabilities in *every* domain, many at a superhuman level

Conventional…

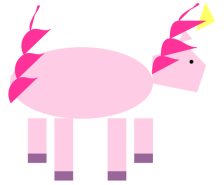
Exam	GPT-4
Uniform Bar Exam (MBE+MEE+MPT)	298 / 400 (~90th)
LSAT	163 (~88th)
SAT Evidence-Based Reading & Writing	710 / 800 (~93rd)
SAT Math	700 / 800 (~89th)
Graduate Record Examination (GRE) Quantitative	163 / 170 (~80th)
Graduate Record Examination (GRE) Verbal	169 / 170 (~99th)
Graduate Record Examination (GRE) Writing	4 / 6 (~54th)
USABO Semifinal Exam 2020	87 / 150 (99th - 100th)
USNCO Local Section Exam 2022	36 / 60
Medical Knowledge Self-Assessment Program	75 %
Codeforces Rating	392 (below 5th)
AP Art History	5 (86th - 100th)
AP Biology	5 (85th - 100th)
AP Calculus BC	4 (43rd - 59th)
AP Chemistry	4 (71st - 88th)

…and unconventional intelligence tests

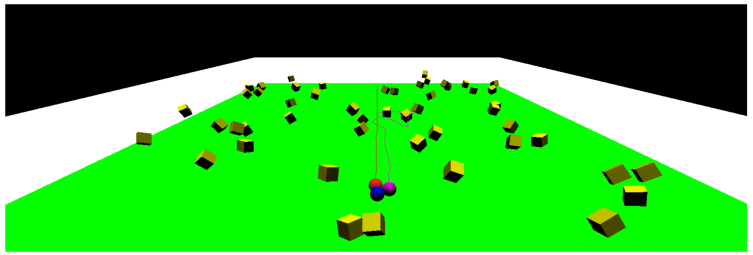
```
Write a poem proving the
infinitude of primes.
```

```
I'll start by noting Euclid's proof,
Which shows that primes aren't just aloof.
Assume we have a finite list,
Of primes, and that none have been missed.
Multiply them all together,
And add one, just to be clever.
```

```
Draw a unicorn in LaTeX.
```



```
Write a 3D game...
```



OpenAI, ‘23. *GPT-4 Technical Report.*
Bubeck et al., ’23. *Sparks of Artificial General Intelligence: Early experiments with GPT-4.*

Agenda

- *What's going on? How did we get here?*

- **Basics:** *generalization + optimization, not magic!*
  - *Models:* a case study in architecture design
  - *Capabilities:* parsing, generation, transfer, in-context learning
  - *Scaling:* datasets, systems

- **Research:** *mysteries & emergent phenomena*
  - *Theory:* computational models within LMs
  - *Empirical science:* explaining, interpreting, & isolating behaviors
  - *Innovation:* what's next?

- **Freeform discussion**

Statistical language modeling

- **Task:** estimate $\widehat{\Pr}_\theta[w_1, \ldots, w_T]$ from examples (sequences of natural language)
- **Autoregressive model:** an LM of the form $\widehat{\Pr}[w_{t+1} \mid w_1, \ldots, w_t]$

$$\widehat{\Pr}[w_1, w_2, w_3] = \widehat{\Pr}[w_1] \cdot \widehat{\Pr}[w_2 | w_1] \cdot \widehat{\Pr}[w_3 | w_2, w_1]$$

- **Self-supervised learning problem:** fit θ to (context, next word) pairs

$$\min_\theta \mathop{\mathbb{E}}_{(w_{1:t}, w_{t+1})} \left[\ell\left(w_{t+1}, f(w_{1:t}; \theta)\right)\right]$$

SGD  GPT-4

- **Which loss?** Multiclass logistic loss (a.k.a. cross entropy)

`Once upon a` → f → $\widehat{\Pr}[\texttt{ the} \mid w_{1:t}]$, $\widehat{\Pr}[\texttt{ cat} \mid w_{1:t}]$, $\widehat{\Pr}[\texttt{ dog} \mid w_{1:t}]$, …, $\widehat{\Pr}[\texttt{ time} \mid w_{1:t}]$

penalize "surprisal"

$$\ell_{\mathrm{CE}}(y, \widehat{\Pr}[\cdot]) \coloneqq -\log \widehat{\Pr}[y]$$

What's a good f for text? Attempt 1

- **n-gram models** (1948+)**:** directly estimate $\widehat{\Pr}[w_t | w_{t-n}, \ldots, w_{t-1}]$ by counting

observed less frequently

I know you are, but what am _____

observed in training data

bias-variance tradeoff:
longer n = larger capacity = needs more data

- **Smoothing:** $\widehat{\Pr}[w_6 \mid w_{1:5}] = \lambda_5 \widehat{\Pr}^{(6)}[w_6 \mid w_{1:5}] + \cdots + \lambda_2 \widehat{\Pr}^{(2)}[w_6 \mid w_5] + \lambda_1 \widehat{\Pr}^{(1)}[w_6]$
- Can these models have "AGI-like" capabilities? Intuitively, no…

In "AI can write", the verb phrase is: _____

English: "pigeons can't swim". French: _____

Shakespeare's 99th sonnet begins with: _____

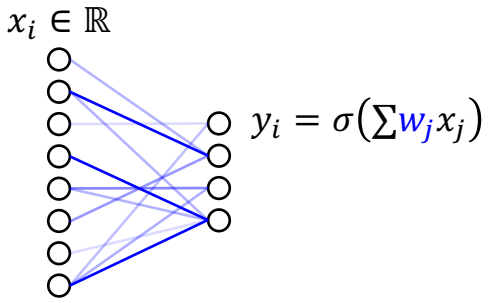
Python code that finds the n^2-th prime: _____

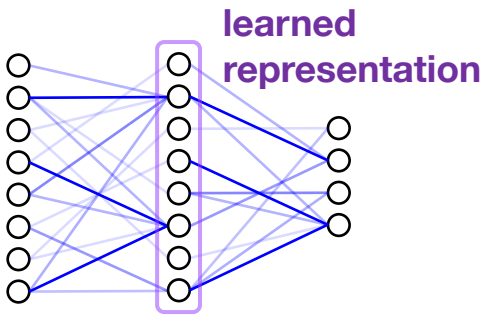
Challenge: long-range dependencies
- $(\text{vocab size})^n$ parameters
- Needs $(\text{vocab size})^n$ data to generalize

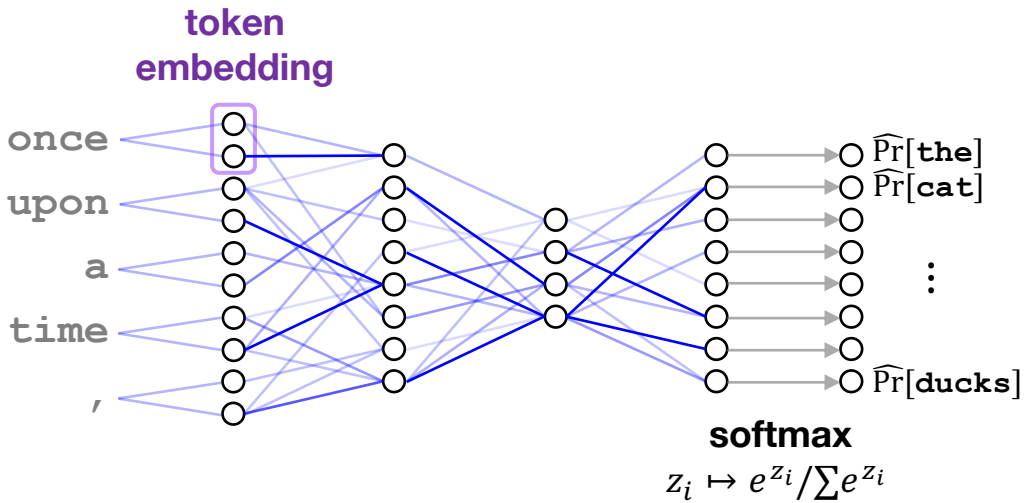
Needs a lot more "parameter sharing"…

What's a good f for text? Attempt 2

- **Neural networks:** arithmetic circuits







- **Weights:** coefficients in these circuits
- **Representations:** intermediate variables in these circuits

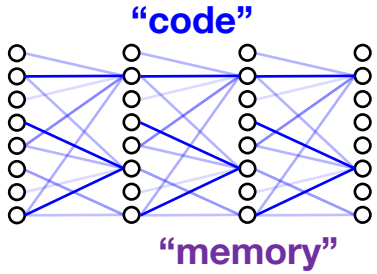
- **Neural LM:** a circuit which guesses a distribution over the next word:

$$w_1, \ldots, w_t \mapsto \widehat{\Pr}[w_{t+1} | w_1, \ldots, w_t]$$

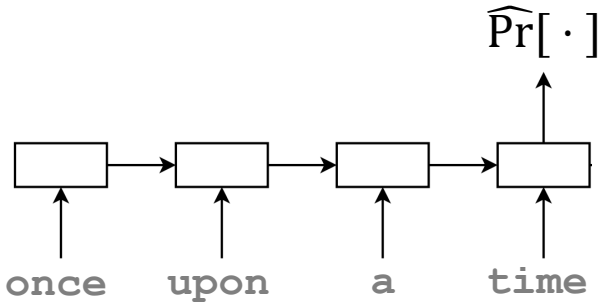
- **Training:** SGD on next-word-prediction loss

What's a good f for text? Attempt 2

- **Recurrent LMs** (1990+)**:** architectures which learn *dynamics*

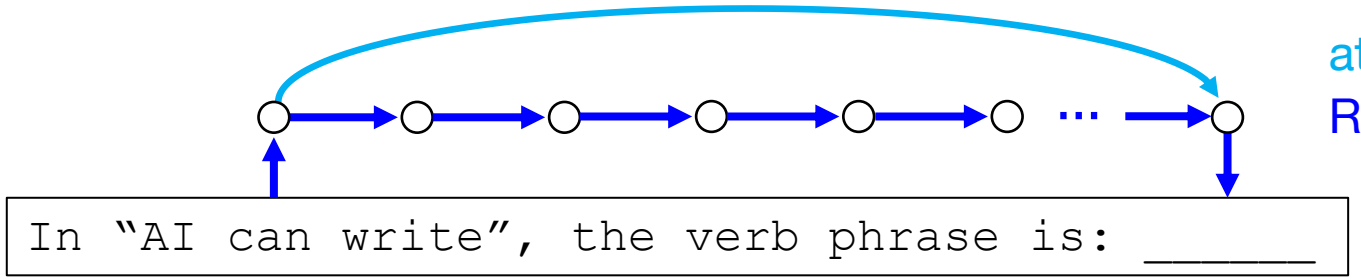


$$h_{t+1} = f_\theta(h_t, u_t)$$



Challenge: non-convexity
Even when fitting $y = A^n x$…
- Local minima
- Vanishing/exploding gradients

- **Attention:** direct paths to long-range dependencies



attention: read a vector over a long range
RNN information flow

- "AGI-like"? Enough to learn grammar & revolutionize NLP…

What's a good f for text? Attempt 3

- **Transformer** (2017+)**:** an attention-only neural architecture

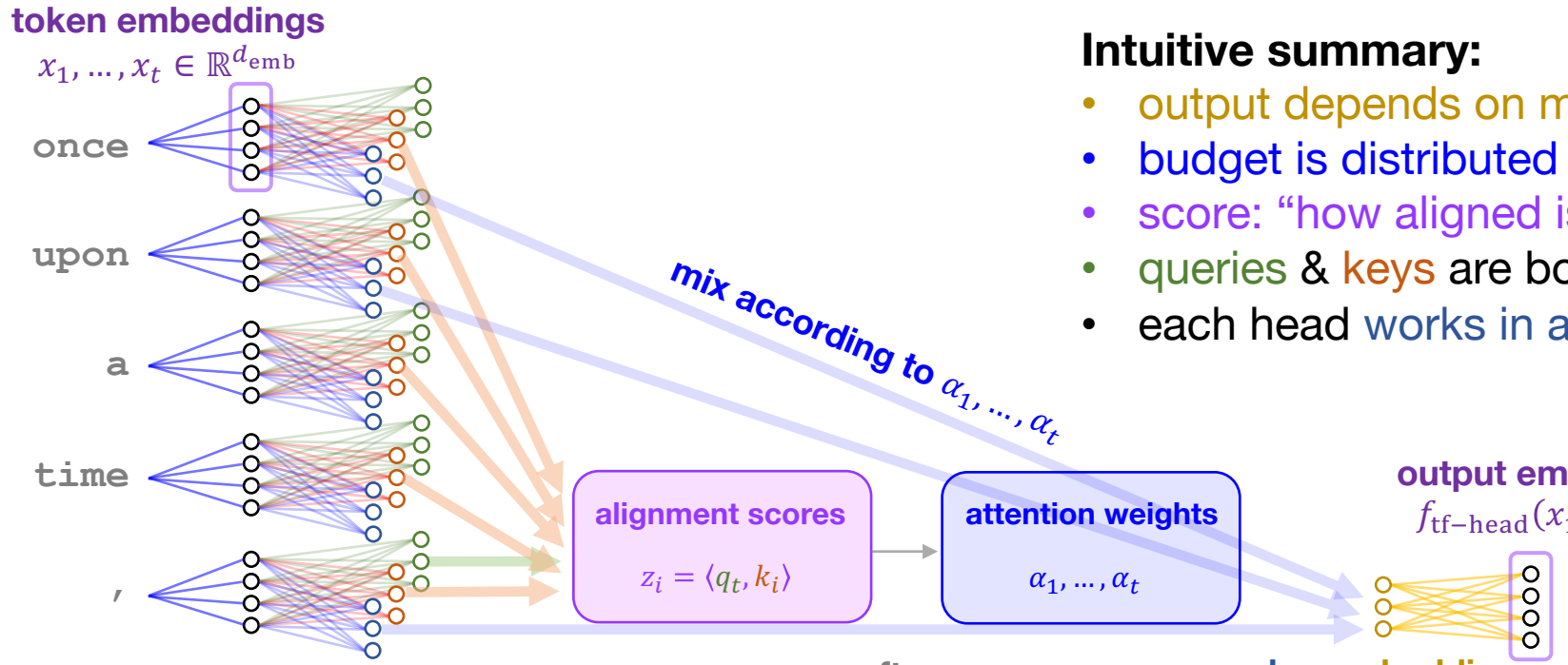
- *What could be improved about RNNs?*
  - **Statistical:** do they underfit or overfit?
  - **Computational:** could the fitting procedure be much faster?

- **Self-attention:** a new component for NN architectures
  - *RNNs underfit, due to parameter sharing. Let's keep it that way.*
  - *RNNs contain non-parallelizable computations.*
  - **How?** Discard the RNN…

Vaswani et al. '17. *Attention is all you need.*
Pseudocode digestion: Phuong & Hutter '22. *Formal algorithms for Transformers.*
Theoretical digestion: Edelman et al. '22. *Inductive biases and variable creation in self-attention mechanisms.*

Anatomy of a Transformer

- **Self-attention head:** a "learned soft pointer" into $x_{1:T}$ which depends on $x_{1:T}$ itself



Intuitive summary:

- output depends on many inputs, with a fixed "budget"
- budget is distributed proportionally to exp(score)
- score: "how aligned is each key with this head's query?"
- queries & keys are both linear functions of the input
- each head works in a restricted subspace

trainable parameters:

$W_Q, W_K, W_V \in \mathbb{R}^{d_{\text{emb}} \times d_{\text{att}}}$

$W_C \in \mathbb{R}^{d_{\text{att}} \times d_{\text{emb}}}$

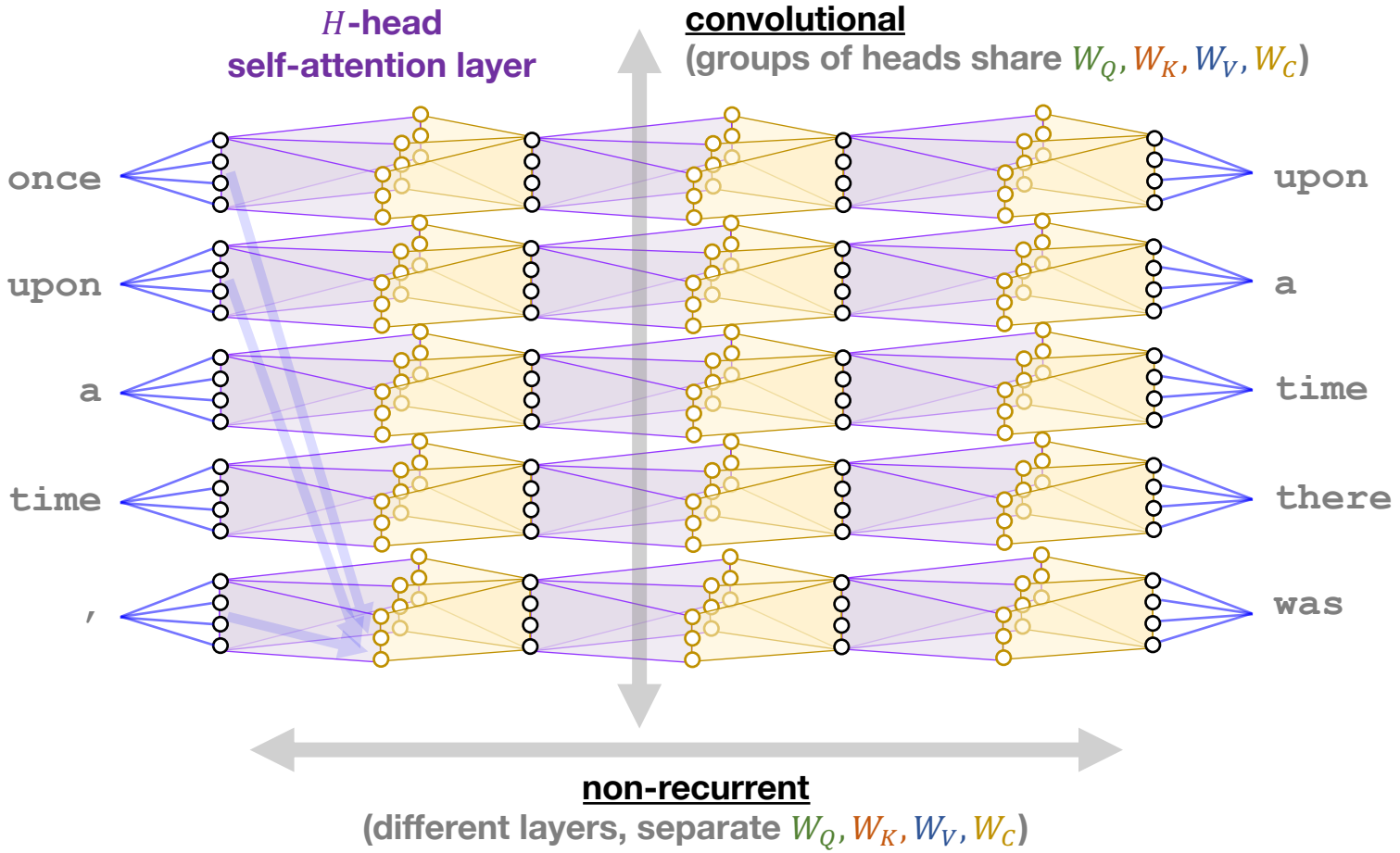
overall computation:

$f_{\text{tf-head}}: \mathbb{R}^{T \times d_{\text{emb}}} \to \mathbb{R}^{d_{\text{att}}}$

$X \mapsto \text{softmax}\left(x_t^\top W_Q W_K^\top X\right) X W_V W_C$

Anatomy of a Transformer

- **Self-attention head:** a module $\mathbb{R}^{T \times d_{\text{emb}}} \to \mathbb{R}^{d_{\text{emb}}}$, with weights W_Q, W_K, W_V, W_C
- **Transformer:** a cascade of self-attention heads, $\mathbb{R}^{T \times d_{\text{emb}}} \to \mathbb{R}^{T \times d_{\text{emb}}}$



Remaining details:

- 2-layer MLPs after each
- normalization layers
- residual connections
- position encodings

GPT-3:

- vocab size $= 50257$ (subwords)
- $d_{\text{emb}} = 12288$, $d_{\text{att}} = 128$, $H = 96$
- 96 sequential layers
- context length $T = 2048$
- trained on 45TB of Internet text

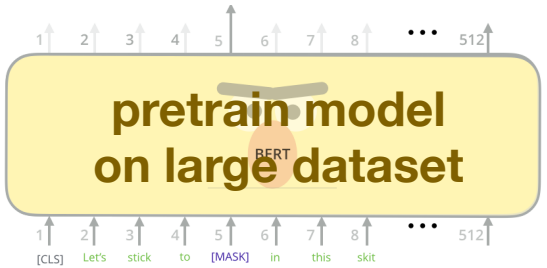
What can large Transformers do?

- **Predict the next word very accurately** (2017+)

```
Once upon a time, there was …
```

```
Toronto is the capital of Ontario, Canada.
```

- **Adapt to new distributions & tasks** (2018+)



finetune on smaller datasets

sentiment analysis
question answering
logical entailment
commonsense reasoning

grammar checking
syntactic parsing
coreference resolution
multilingual understanding

- **Understand data & instructions in-context** (2020+)

```
...
"You don't want to miss it."
This review is positive.
```

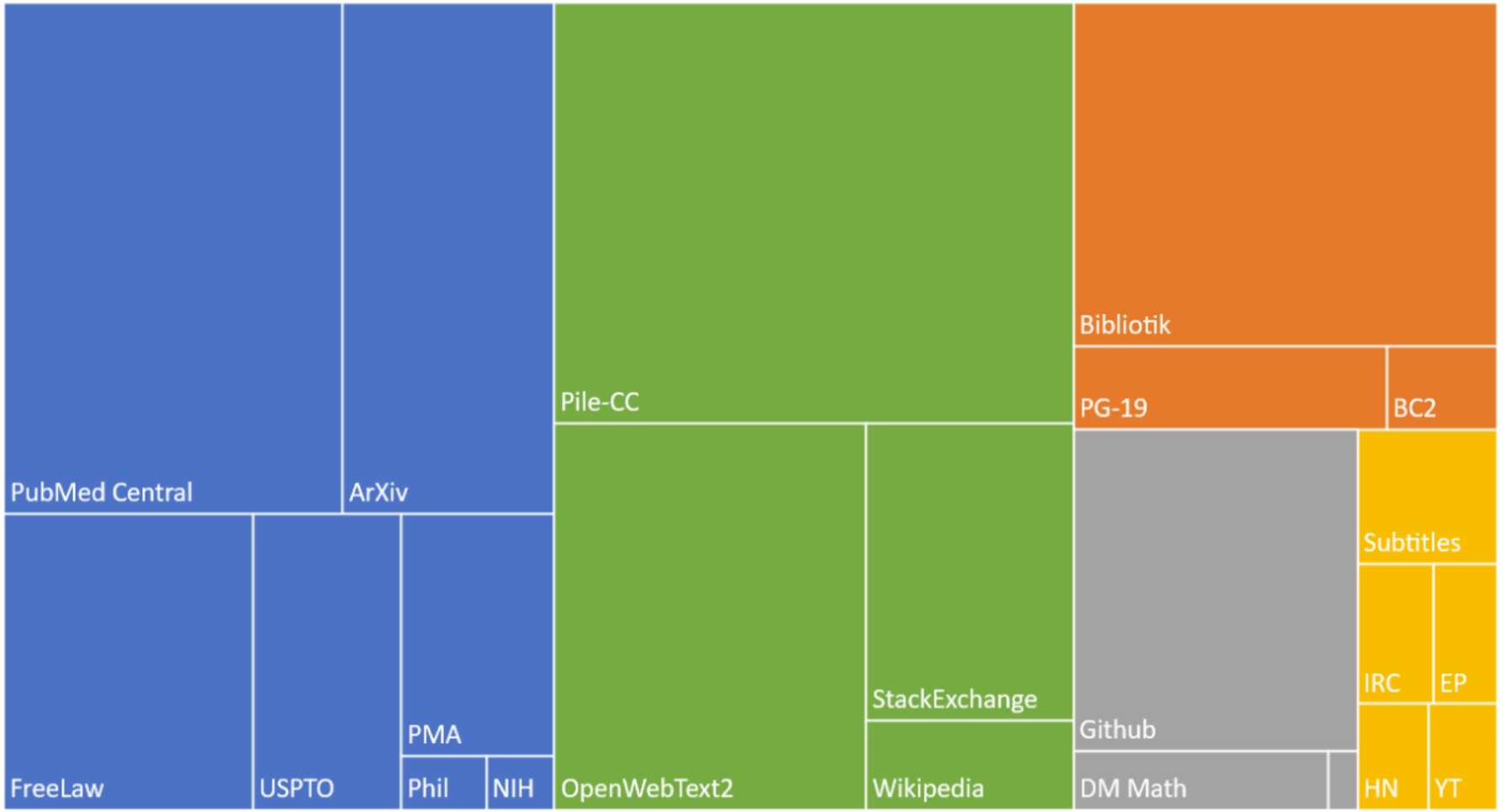
```
// shortest paths on graph G
for(int i=0; i<n; i++) {
  for(int j=0; j<n; j++) { ...
```

```
User: Write my lecture.
Bot: Sure! Your talk on
NLP/Transformers: ...
```

- Not just language processing: vision, protein folding, trajectory planning, …

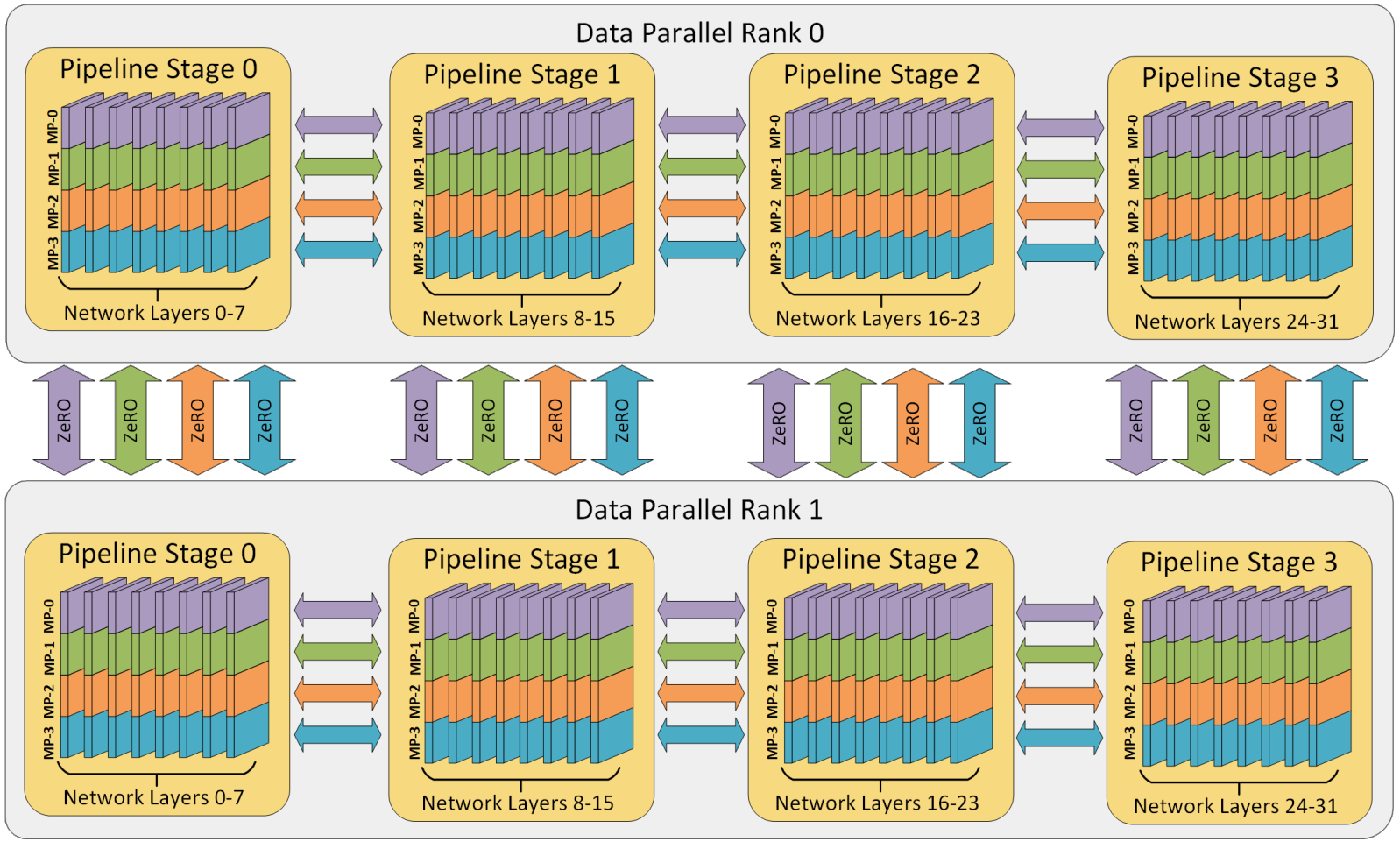
How much training data?

- **As much as we can find.** The Pile: 800GB open-source dataset



How much computation?

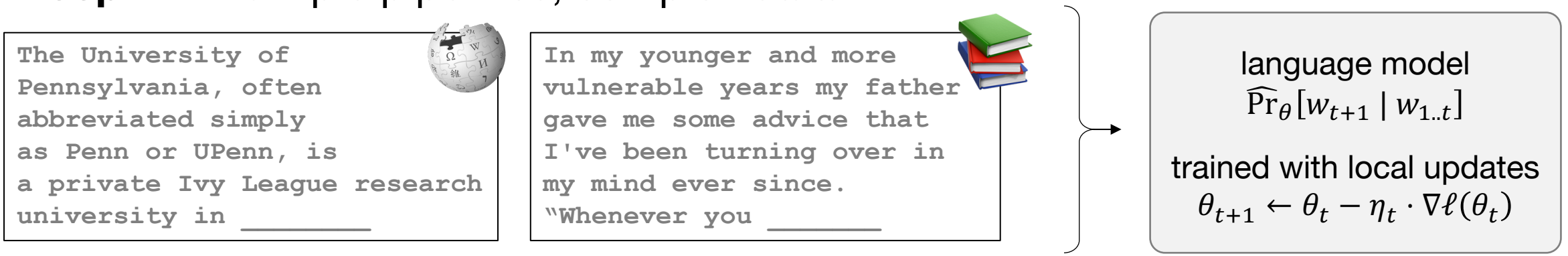
- **GPT-3 as a systems puzzle:** $\sim 10^{11}$ parameters, $\sim 10^{23}$ FLOPs, $\sim 10^{2}$ days



*parallelization schematic for a smaller Transformer

Summary of Transformers 101

- **Deep NLP:** simple pipelines, complex data



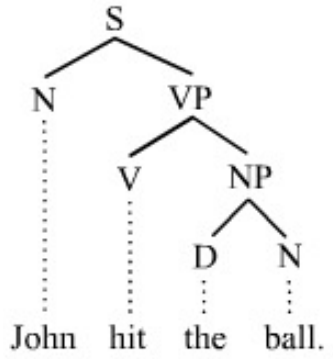
- **Principles are as usual in ML:** pick a function class where you can fit efficiently
- **Transformers:** a parallel function class, with well-chosen biases
- Out-of-distribution generalization emerges, beyond what current theory predicts

- **Next:** a taste of this new science of foundation models

Bommasani et al. '21. *On the Opportunities and Risks of Foundation Models.*

What do Transformers compute internally?

- LMs need to contain **grammar-parsing algorithms**

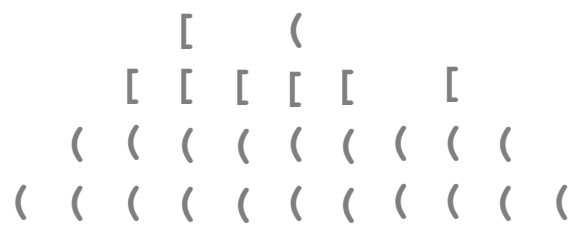



```
# Floyd-Warshall on numpy distance array A
def all_pairs_shortest_path(A):
  for i in range(len(A)):
    for j in range(len(A)):
      for k in range(len(A)):
        A[i,j] = min(A[i,j], A[i,k] + A[k,j])
  return A
```

```
((([[]()][])))
()[]()[](())
([{}[({})]])
```

Dyck languages:
correctly nested brackets

- RNNs naturally contain iterative algorithms. *What do Transformers implement?*



stack-based parser

```
Input:  ( ( [ [ ] ( ) ] [ ] ) )
Depth:  1 2 3 4 3 4 3 2 3 2 1 0
```

self-attention

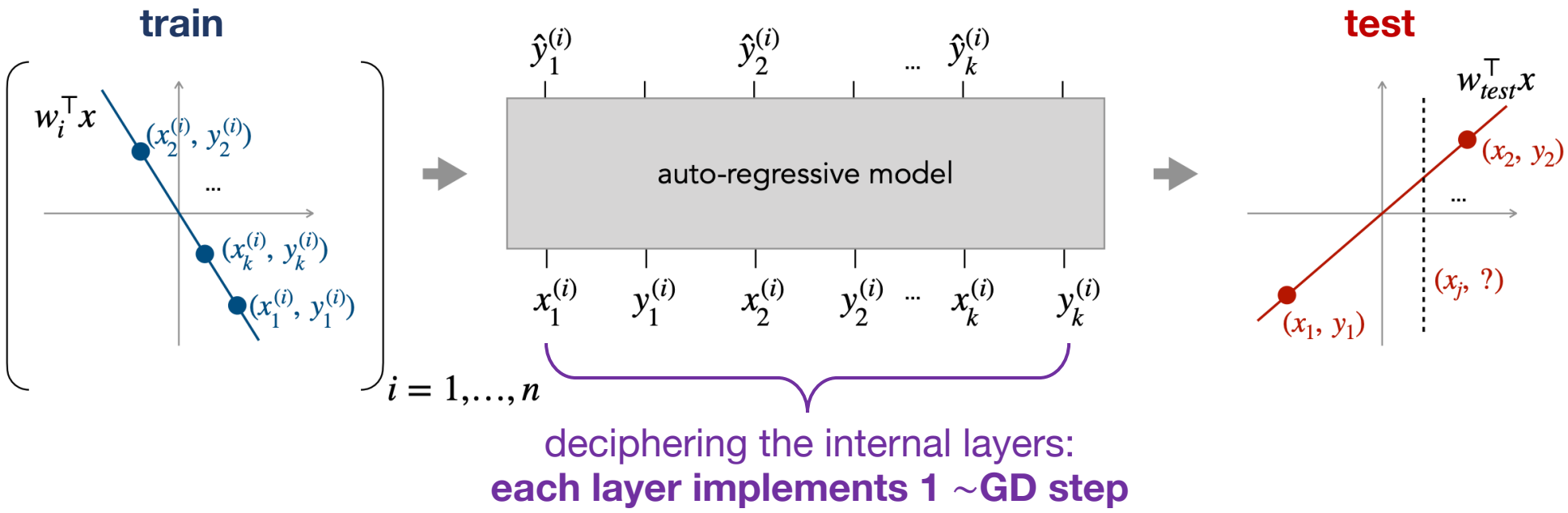
```
Stack top: ( ( [ [ [ ( [ ( [ ( ( _
```

parallel parser

Yao et al. '21. *Self-Attention Networks Can Process Bounded Hierarchical Languages.*
Liu et al., '23. *Transformers learn shortcuts to automata.*
Zhao et al. '23. *Do Transformers parse while predicting the masked word?*

Empirical science: learning learning algorithms

- **This course:** understand how to fit linear predictors
- **In-context learning:** Transformers can meta-learn how to fit linear predictors



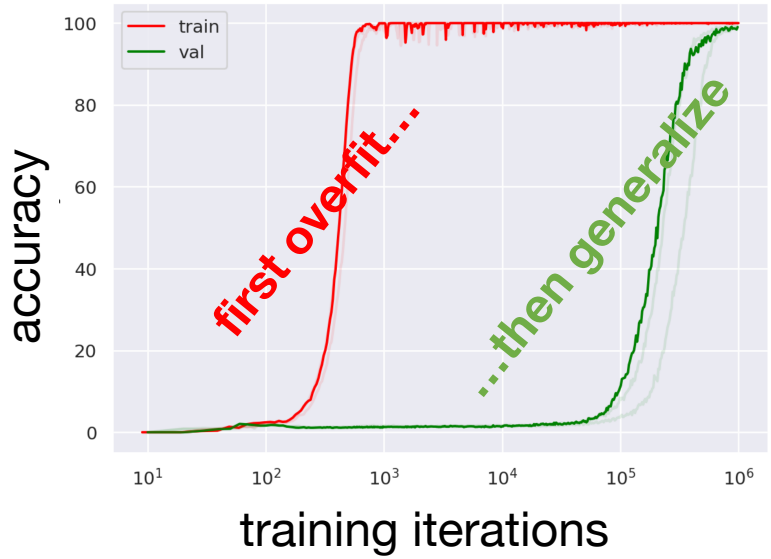
deciphering the internal layers:
each layer implements 1 ~GD step

Garg et al., '22. *What can Transformers learn in-context? A case study of simple function classes.*
Oswald et al. '22. *Transformers learn in-context by gradient descent.*
Dai et al. '22. *Why can GPT learn in-context? Language models secretly perform gradient descent as meta-optimizers.*

Innovation: what's the next breakthrough?

- *What could be improved about Transformers?*
  - **Statistical:** do they underfit or overfit?
  - **Computational:** could the fitting procedure be much faster?

grokking



hundreds of efficient Transformer variants

Model	Time	Space
Transformer	$O(T^2d)$	$O(T^2 + Td)$
Reformer	$O(T \log T d)$	$O(T \log T + Td)$
Linear Transformer	$O(Td^2)$	$O(Td + d^2)$
Performer	$O(Td^2 \log d)$	$O(Td \log d + d^2 \log d)$
AFT-simple	$O(\mathbf{Td})$	$O(\mathbf{Td})$
AFT-full	$O(T^2d)$	$O(\mathbf{Td})$
AFT-local (AFT-conv)	$O(Tsd),\ s < T$	$O(\mathbf{Td})$

Power et al., '22. *Grokking: generalization beyond overfitting on small algorithmic datasets.*
Zhai et al. '21. *An Attention-Free Transformer.*

Chat time