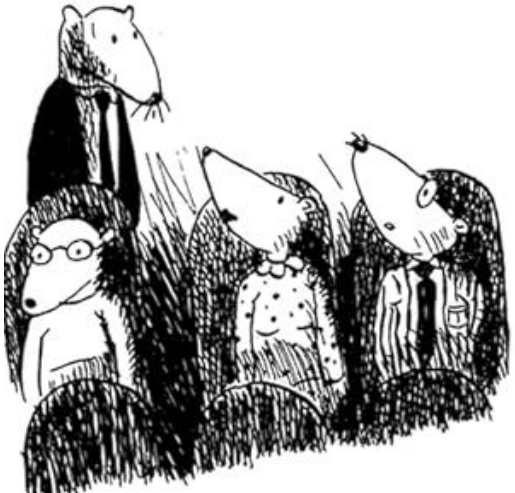
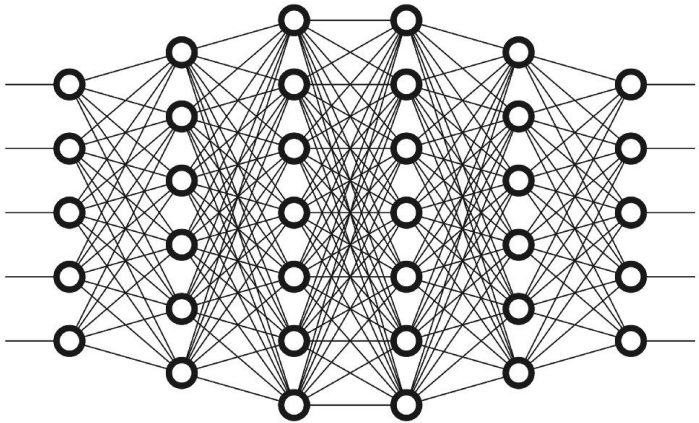
GROKKING

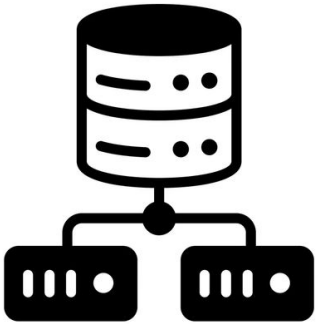
Ameli Alaeva



1

1





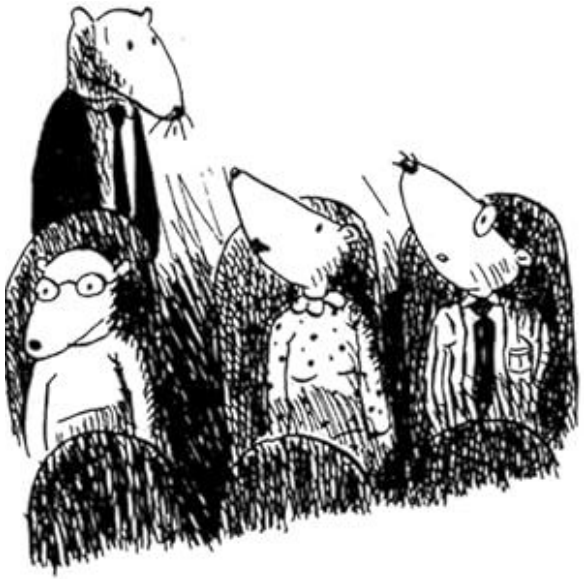
Dataset

generalization

PLAN

- reference
- 1st paper
  - observe dataset + model architecture
  - the goal
  - experiments + results
- 2d paper
- additional info
- conclusion

Q&A in the end



grok

/grɒk/

verb **INFORMAL • US**

gerund or present participle: **grokking**

understand (something) intuitively or by empathy.
"corporate leaders seemed to grok this concept fairly quickly"

GROKKING: GENERALIZATION BEYOND OVERFITTING ON SMALL ALGORITHMIC DATASETS

Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin
OpenAI

Vedant Misra*
Google

All of our experiments used a small transformer trained on datasets of equations of the form $a \circ b = c$, where each of "a", "$\circ$", "b", "$=$", and "c" is a separate token. Details of the operations studied, the architecture, training hyperparameters and tokenization can be found in Appendix A.1.

A.1.1 BINARY OPERATIONS

The following are the binary operations that we have tried (for a prime number $p = 97$):

$$x \circ y = x + y \pmod{p} \text{ for } 0 \leq x, y < p$$
$$x \circ y = x - y \pmod{p} \text{ for } 0 \leq x, y < p$$
$$x \circ y = x/y \pmod{p} \text{ for } 0 \leq x < p, 0 < y < p$$
$$x \circ y = [x/y \pmod{p} \text{ if } y \text{ is odd, otherwise } x - y \pmod{p}] \text{ for } 0 \leq x, y < p$$
$$x \circ y = x^2 + y^2 \pmod{p} \text{ for } 0 \leq x, y < p$$
$$x \circ y = x^2 + xy + y^2 \pmod{p} \text{ for } 0 \leq x, y < p$$
$$x \circ y = x^2 + xy + y^2 + x \pmod{p} \text{ for } 0 \leq x, y < p$$
$$x \circ y = x^3 + xy \pmod{p} \text{ for } 0 \leq x, y < p$$
$$x \circ y = x^3 + xy^2 + y \pmod{p} \text{ for } 0 \leq x, y < p$$
$$x \circ y = x \cdot y \text{ for } x, y \in S_5$$
$$x \circ y = x \cdot y \cdot x^{-1} \text{ for } x, y \in S_5$$
$$x \circ y = x \cdot y \cdot x \text{ for } x, y \in S_5$$

For each binary operation we constructed a dataset of equations of the form $\langle x \rangle \langle op \rangle \langle y \rangle \langle = \rangle \langle x \circ y \rangle$, where $\langle a \rangle$ stands for the token corresponding to element a.

★	a	b	c	d	e
a	a	d	?	c	d
b	c	d	d	a	c
c	?	e	d	b	d
d	a	?	?	b	c
e	b	b	c	?	a

★	a	b	c	d	e
a	a	d	?	c	d
b	c	d	d	a	c
c	?	e	d	b	d
d	a	?	?	b	c
e	b	b	c	?	a

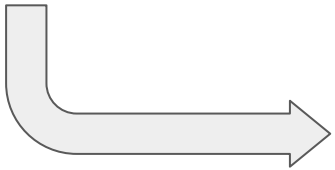
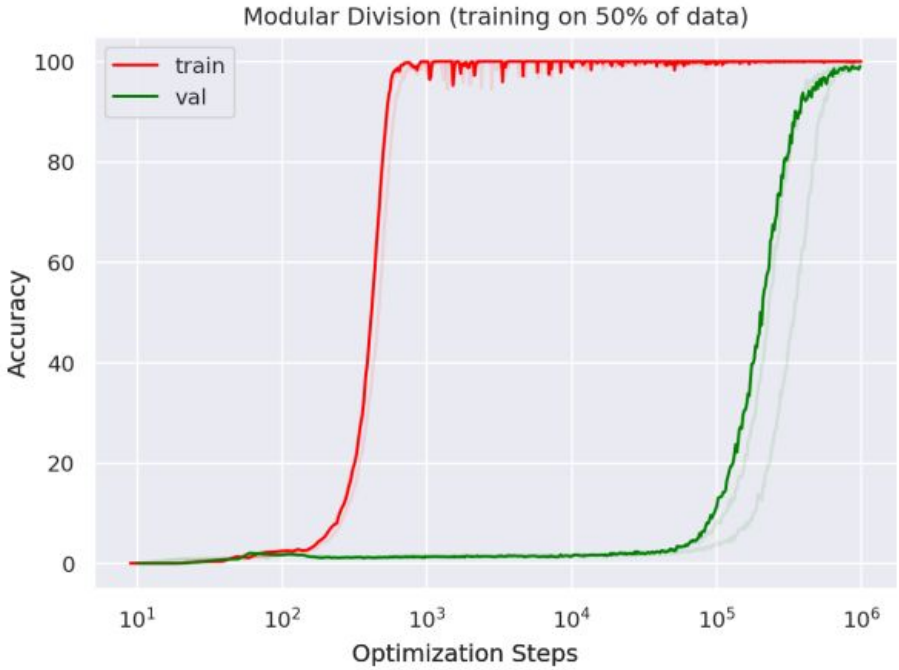


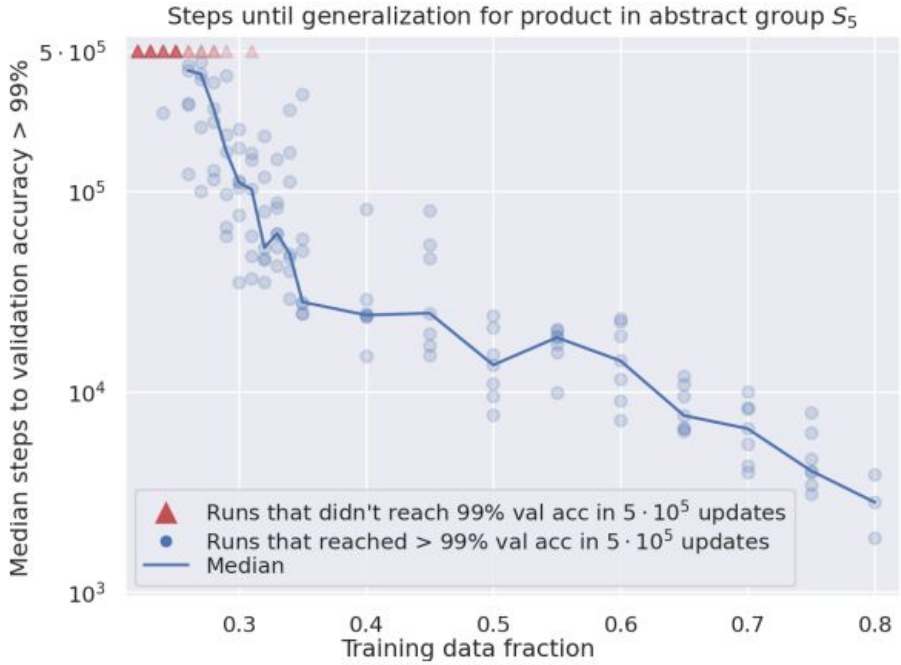
Figure 5: One of the binary operation tables presented to the networks that the network can perfectly fill in. Each symbol is represented as a letter in English, Hebrew, or Greek alphabet for reader's convenience. We invite the reader to guess which operation is represented here.

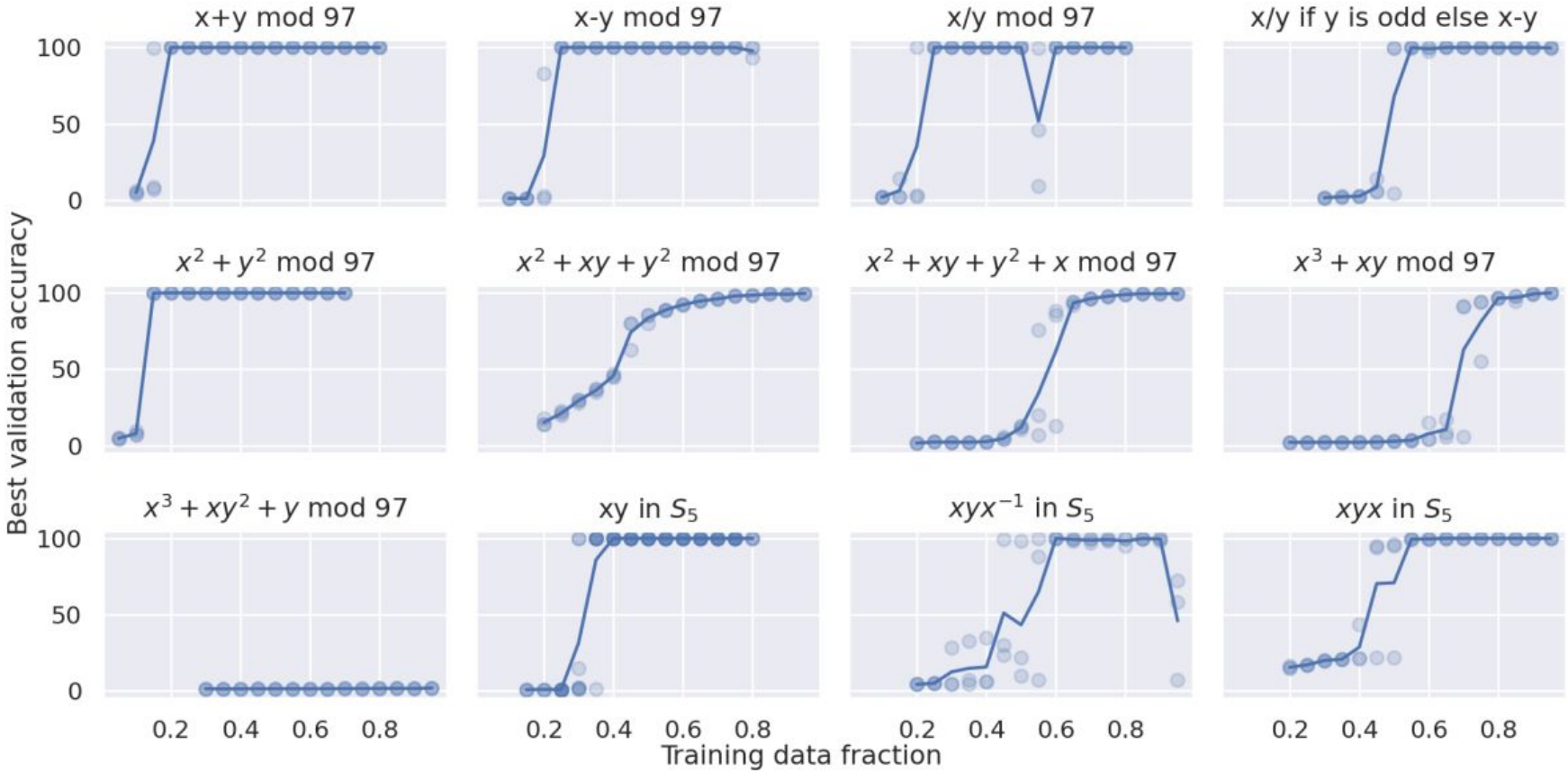
“a”, “b”, “c”, “=” and “∘”

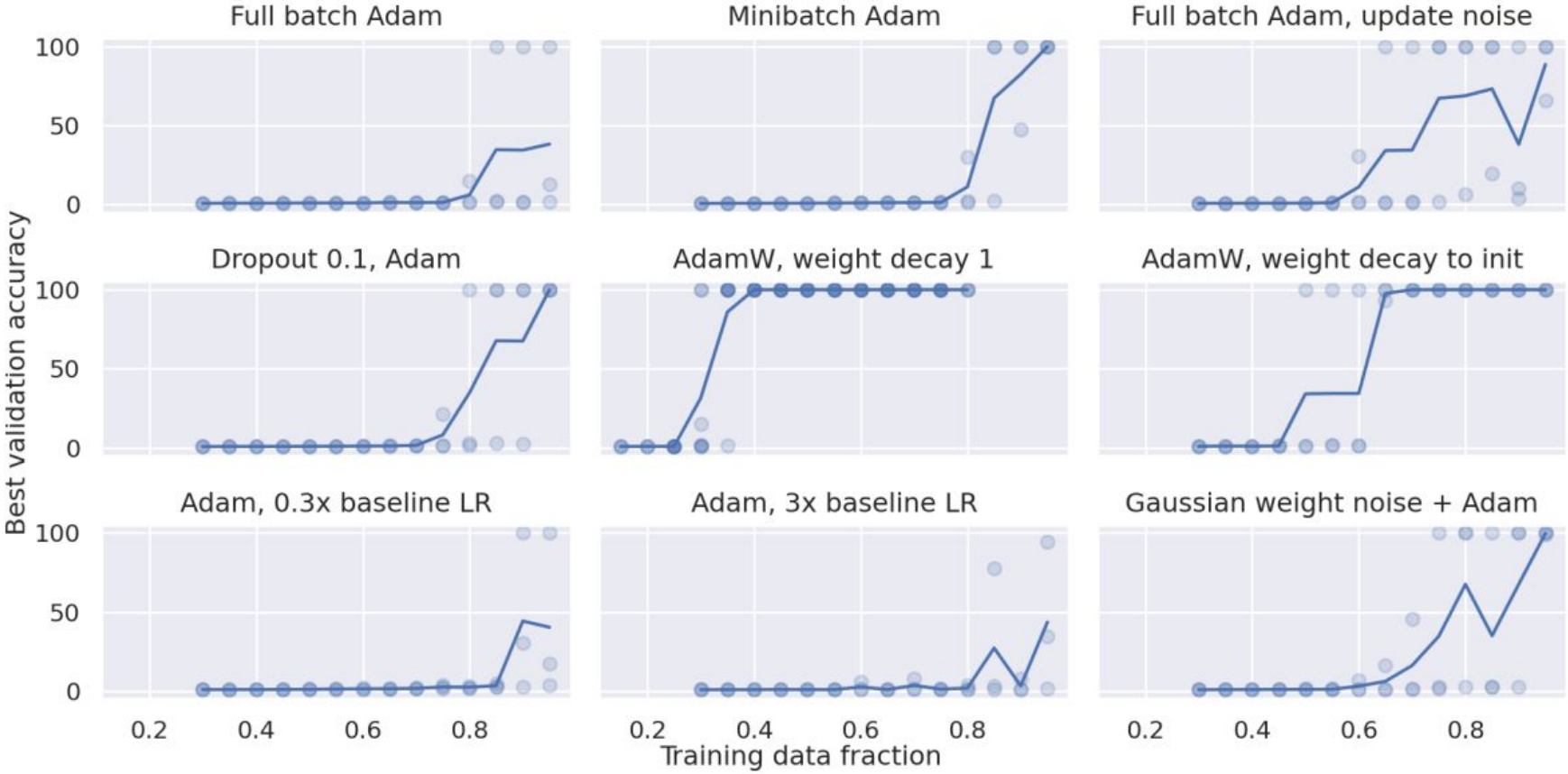
- a standard decoder-only transformer Vaswani et al. (2017)
- causal attention masking
- transformer with 2 layers,
- width 128
- 4 attention heads

total of about 4 · 105 non-embedding parameters


Modular Division (training on 50% of data)
train
val
Accuracy
Optimization Steps
100
80
60
40
20
0
10^1
10^2
10^3
10^4
10^5
10^6


Steps until generalization for product in abstract group S_5
Median steps to validation accuracy > 99%
Training data fraction
Runs that didn't reach 99% val acc in $5 \cdot 10^5$ updates
Runs that reached > 99% val acc in $5 \cdot 10^5$ updates
Median
$5 \cdot 10^5$
10^5
10^4
10^3
0.3
0.4
0.5
0.6
0.7
0.8


x+y mod 97
x-y mod 97
x/y mod 97
x/y if y is odd else x-y
$x^2 + y^2$ mod 97
$x^2 + xy + y^2$ mod 97
$x^2 + xy + y^2 + x$ mod 97
$x^3 + xy$ mod 97
$x^3 + xy^2 + y$ mod 97
xy in S_5
xyx^{-1} in S_5
xyx in S_5
Best validation accuracy
Training data fraction
0
50
100
0.2
0.4
0.6
0.8


Full batch Adam
Minibatch Adam
Full batch Adam, update noise
Dropout 0.1, Adam
AdamW, weight decay 1
AdamW, weight decay to init
Adam, 0.3x baseline LR
Adam, 3x baseline LR
Gaussian weight noise + Adam
Best validation accuracy
Training data fraction
0
50
100
0.2
0.4
0.6
0.8

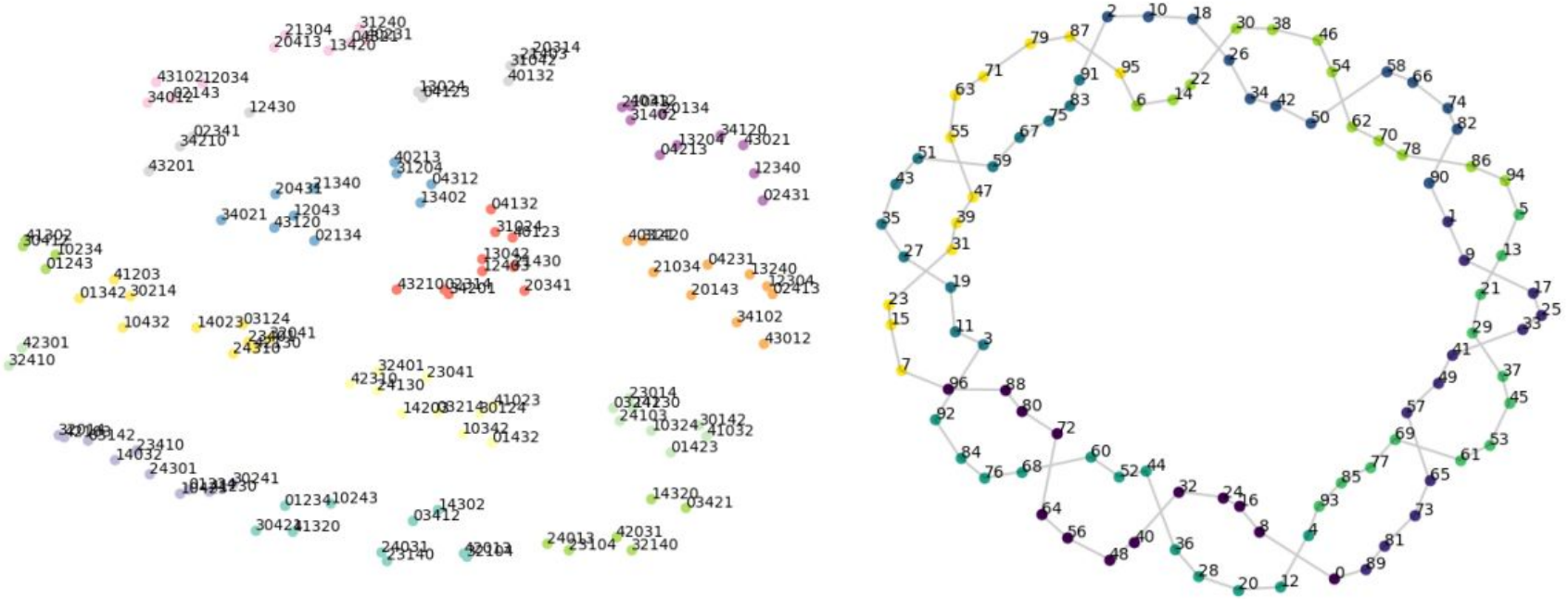


Figure 3: **Left**. t-SNE projection of the output layer weights from a network trained on S_5. We see clusters of permutations, and each cluster is a coset of the subgroup $\langle (0,3)(1,4), (1,2)(3,4) \rangle$ or one of its conjugates. **Right**. t-SNE projection of the output layer weights from a network trained on modular addition. The lines show the result of adding 8 to each element. The colors show the residue of each element modulo 8.

PROGRESS MEASURES FOR GROKKING VIA MECHANISTIC INTERPRETABILITY

Neel Nanda[*, †] **Lawrence Chan**[‡] **Tom Lieberum**[†] **Jess Smith**[†] **Jacob Steinhardt**[‡]

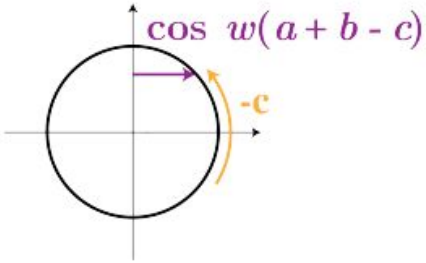
arXiv:2301.05217v3 [cs.LG] 19 Oct 2023

progress measures ~ mechanistic interpretability

reverseengineering learned behaviors into their individual components

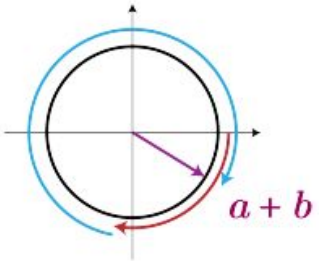
Computes logits using further trig identities:

$$\text{Logit}(c) \propto \cos(w(a+b-c))$$
$$= \cos(w(a+b))\cos(wc) + \sin(w(a+b))\sin(wc)$$



Calculates sine and cosine of $a+b$ using trig identities:

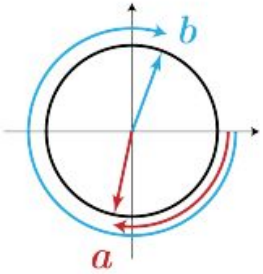
$$\sin(w(a+b)) = \sin(wa)\cos(wb) + \cos(wa)\sin(wb)$$
$$\cos(w(a+b)) = \cos(wa)\cos(wb) - \sin(wa)\sin(wb)$$



Translates one-hot a, b to Fourier basis:

$a \to \sin(wa), \cos(wa)$

$b \to \sin(wb), \cos(wb)$



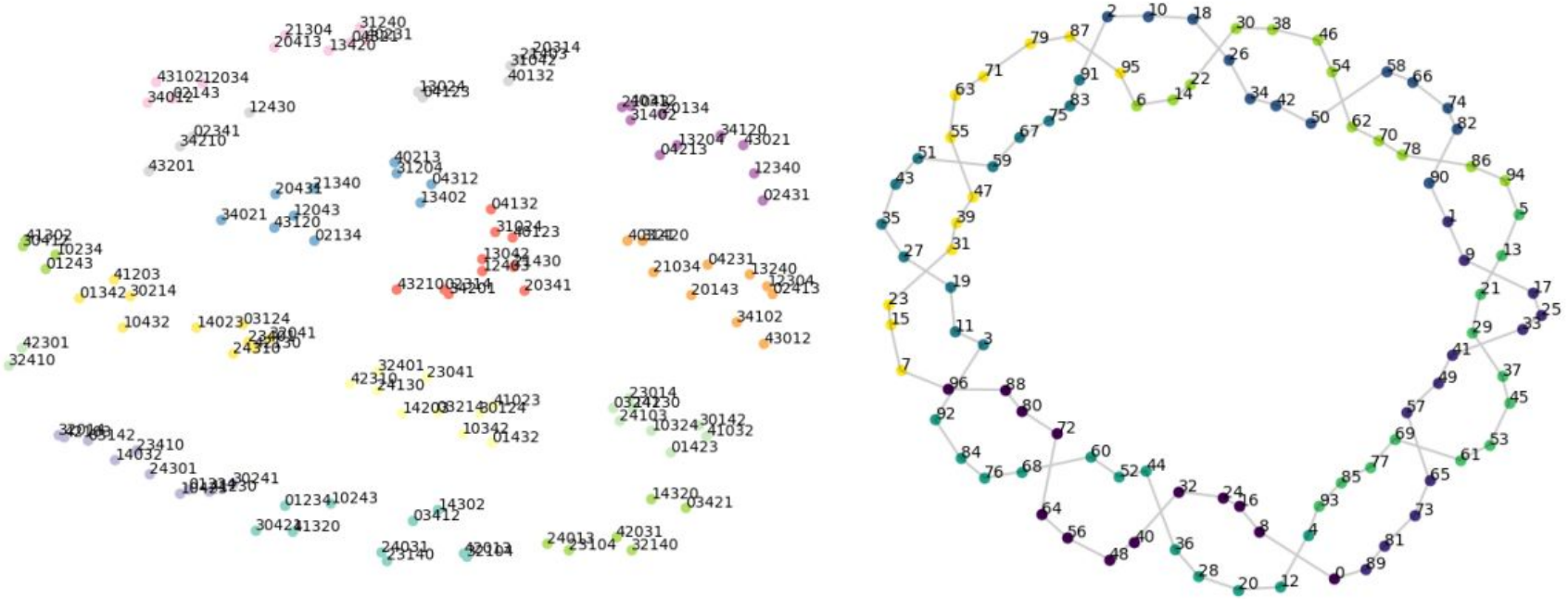
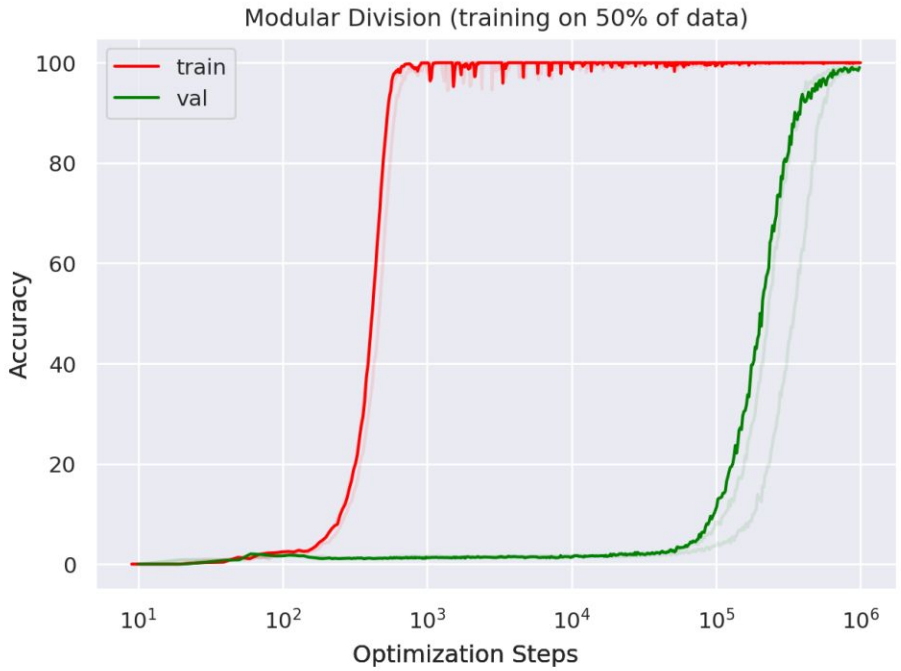


Figure 3: **Left**. t-SNE projection of the output layer weights from a network trained on S_5. We see clusters of permutations, and each cluster is a coset of the subgroup $\langle (0,3)(1,4), (1,2)(3,4) \rangle$ or one of its conjugates. **Right**. t-SNE projection of the output layer weights from a network trained on modular addition. The lines show the result of adding 8 to each element. The colors show the residue of each element modulo 8.

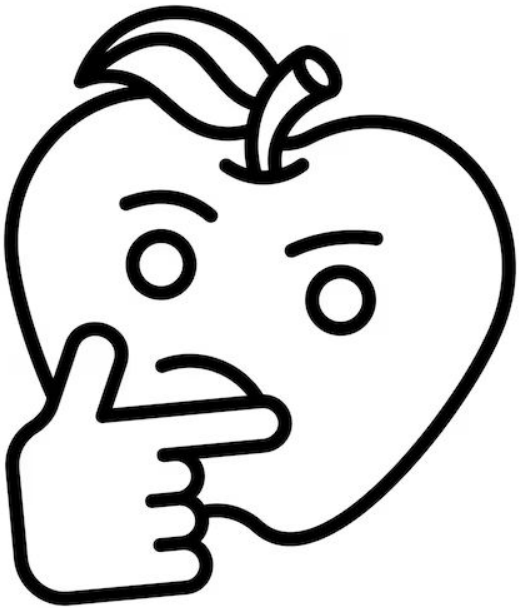
- memorization
- circuit formation
- cleanup

Cleanup = Grokking



Grokking = delayed generalization

Too Artificial?

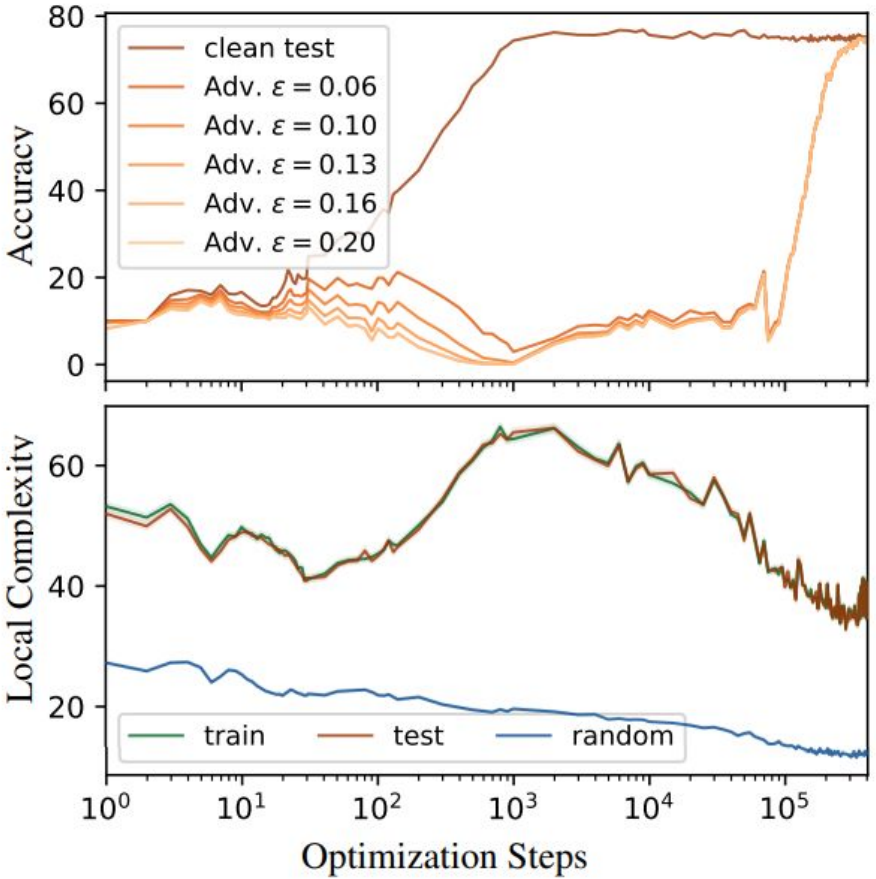


Deep Networks Always Grok and Here is Why

Ahmed Imtiaz Humayun [1] **Randall Balestriero** [2] **Richard Baraniuk** [1]

arXiv:2402.15555v2 [cs.LG] 6 Jun 2024

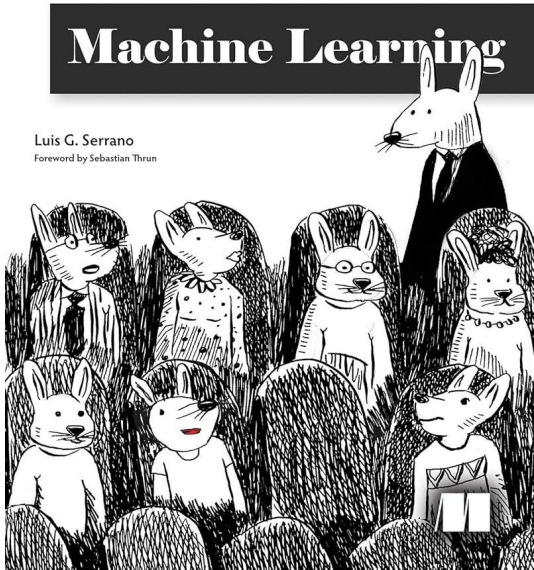
- CNN on CIFAR10
- Resnet on Imagenette



Conclusion

- **small datasets - ?**
- **grokking: accuracy ↗**
- **grokking: delayed generalization**
- **future - ?**

Thank you for attention!



Any questions?