# Neural Networks learn Representation Theory:
## Reverse Engineering how Networks perform Group Operations

Bilal Chughtai, Lawrence Chan, Neel Nanda

# PROGRESS MEASURES FOR GROKKING VIA MECHANISTIC INTERPRETABILITY

**Neel Nanda**
Independent
neelnanda27@gmail.com

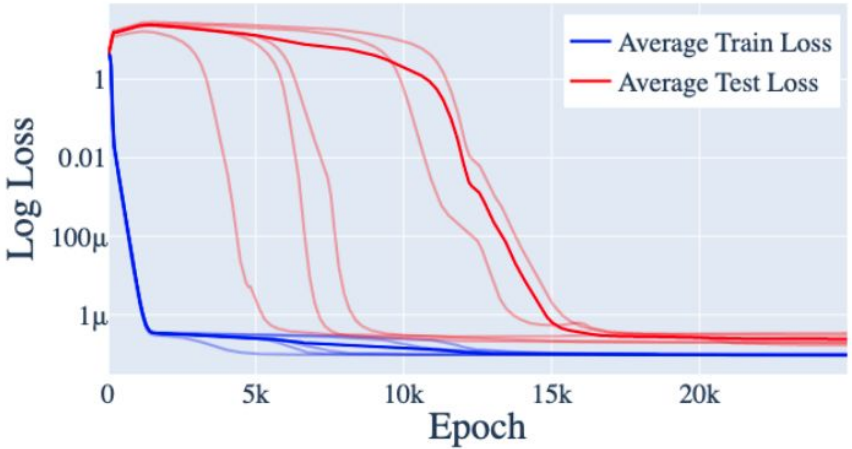
**Lawrence Chan**
UC Berkeley
chanlaw@berkeley.edu

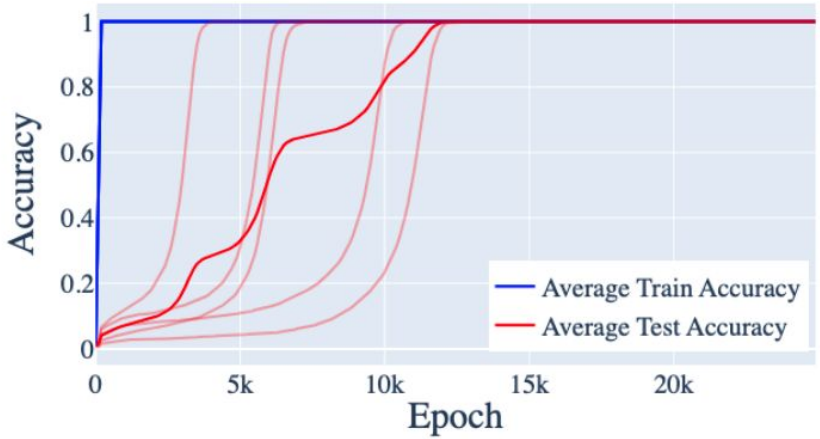
**Tom Lieberum**
Independent
tlieberum3141@gmail.com

**Jess Smith**
Independent
smith.jessk@gmail.com

**Jacob Steinhardt**
UC Berkeley
jsteinhardt@berkeley.edu

# **Mystery:** Why do models grok?



# GROKKING: GENERALIZATION BEYOND OVERFITTING ON SMALL ALGORITHMIC DATASETS

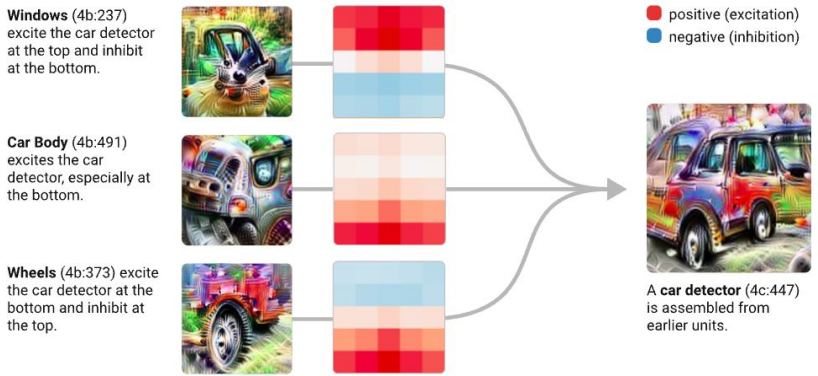
Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin
OpenAI

Vedant Misra*
Google

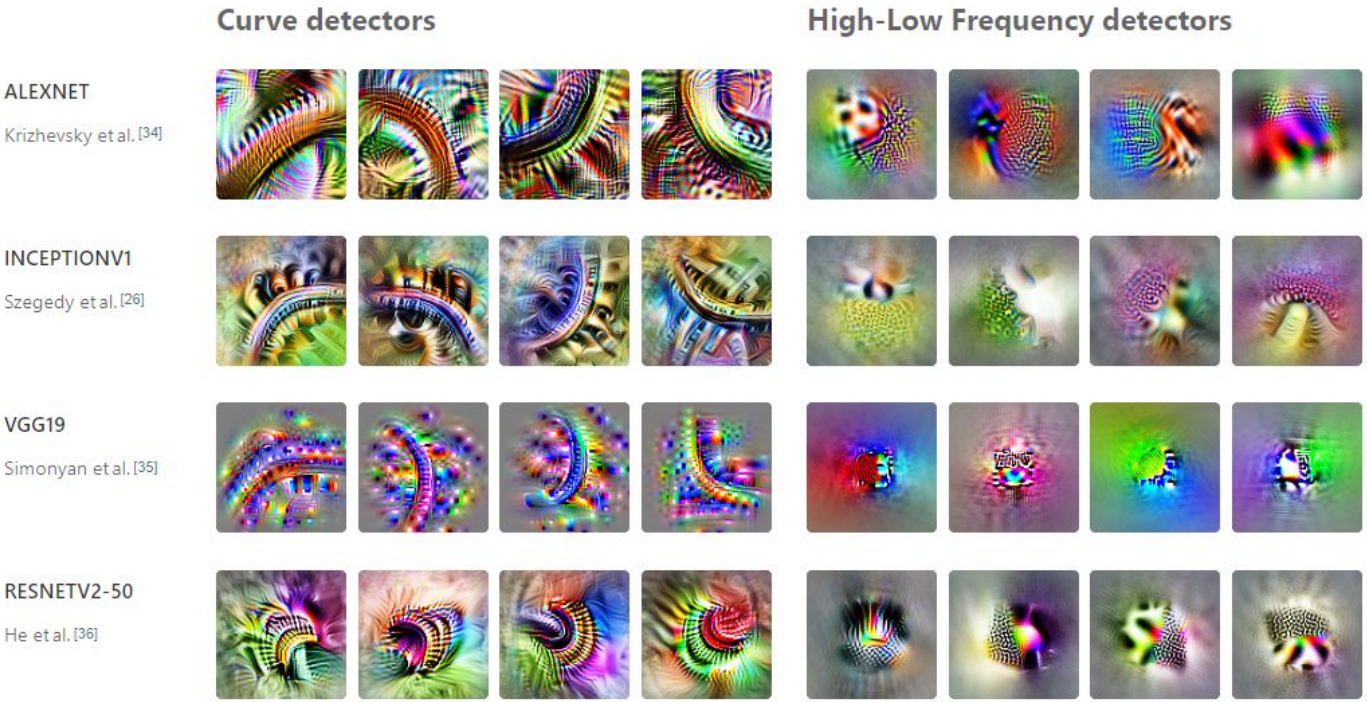
**Methodology:** Apply mechanistic interpretability

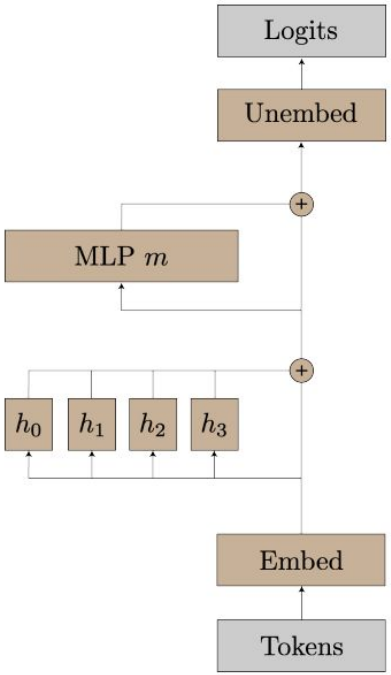
# Inspiration: Mechanistic Interpretability

- **Goal:** Reverse engineer neural networks
- **Hypothesis**: Models learn human-comprehensible algorithms and can be understood, if we learn how to make it legible
- Models learn **circuits**, algorithms encoded in the weights
- **Motivation**: A deep knowledge of circuits is crucial to understand and predict model behaviour



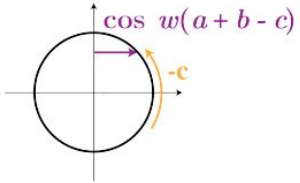
# Universality





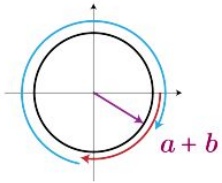

Computes logits using further trig identities:

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




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

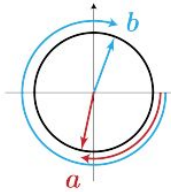
$$\sin(w(a+b)) = \sin(wa)\cos(wb) + \cos(wa)\sin(wb)$$
$$\cos(w(a+b)) = \cos(wa)\cos(wb) - \sin(wa)\sin(wb)$$




Translates one-hot a, b to Fourier basis:

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

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



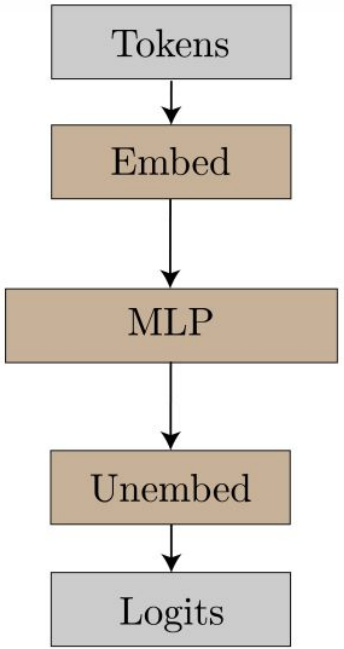
# Representation Theory

**Definition 11.2.1** *A* representation *of a group $G$ on a vector space $V$ is a group homomorphism*

$$\rho : G \to \mathbf{GL(V)}.$$

*We say that $\rho$ is a representation of $G$.*

$$\rho(g_1 g_2) = \rho(g_1)\rho(g_2), \qquad \text{for all } g_1, g_2 \in G.$$




Translates one-hot a,b to representation matrices:

$a, b \mapsto \rho(a), \rho(b)$

Performs matrix multiplication on representations via ReLUs:

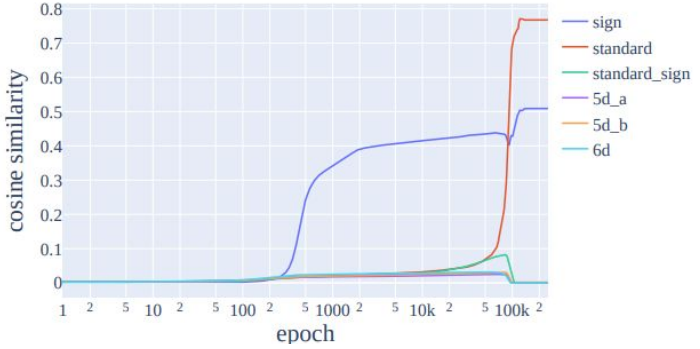
$\rho(a), \rho(b) \mapsto \rho(a)\rho(b) = \rho(ab)$

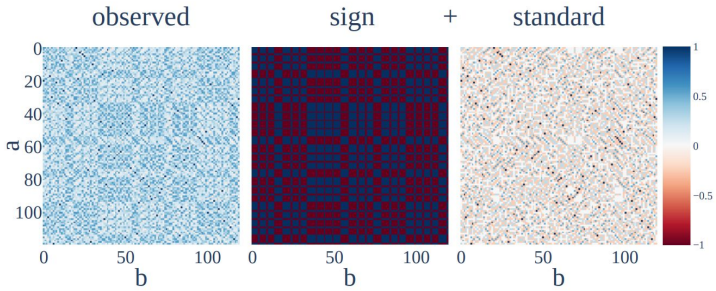
Computes logits by multiplying by $\rho(c^{-1})$ and taking the trace:

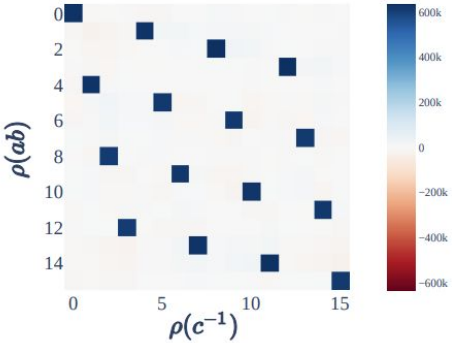
$\text{Logit}(c) \propto \chi_\rho(abc^{-1}) = \text{tr}(\rho(abc^{-1}))$

# Reverse Engineering S5

1. Logit similarity
2. Embeddings
3. MLP activations & the MLP - Logit map
4. Ablations




| | $W_a$ | $W_b$ | $W_U$ |
|---|---|---|---|
| SIGN | 6.95% | 6.95% | 9.58% |
| STANDARD | 93.0% | 93.0% | 84.5% |
| RESIDUAL | 0.00% | 0.00% | 5.96% |

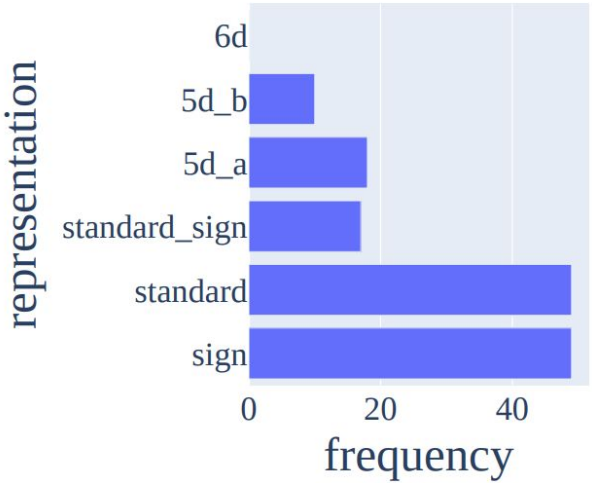
| CLUSTER | $\rho(a)$ | $\rho(b)$ | $\rho(ab)$ | RESIDUAL |
|---|---|---|---|---|
| SIGN | 33.3% | 33.3% | 33.3% | 0.00% |
| STANDARD | 39.6% | 37.1% | 11.3% | 12.1% |

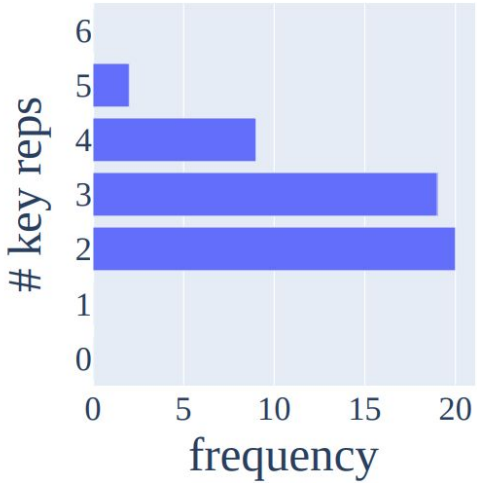
# Weak Universality

*Table 3.* Results from all groups on both MLP and Transformer architectures, averaged over 4 seeds. We find that that features for matrices in the key representations are learned consistently, and explain almost all of the variance of embeddings and unembeddings. We find that terms corresponding to $\rho(ab)$ are consistently present in the MLP neurons, as expected by our algorithm. Excluding and restricting to these terms in the key representations damages performance/does not affect performance respectively.

| | **MLP** | | | | | | | | **Transformer** | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FVE | | | | | Loss | | | FVE | | | | Loss | | |
| *Group* | $W_a$ | $W_b$ | $W_U$ | *MLP* | $\rho(ab)$ | *Test* | *Exc.* | *Res.* | $W_E$ | $W_L$ | *MLP* | $\rho(ab)$ | *Test* | *Exc.* | *Res.* |
| $C_{113}$ | 99.53% | 99.39% | 98.05% | 90.25% | 12.03% | 1.63e-05 | 5.95 | 6.88e-03 | 95.18% | 99.52% | 92.12% | 16.77% | 2.67e-07 | 9.42 | 2.12e-02 |
| $C_{118}$ | 99.75% | 99.74% | 98.43% | 95.84% | 13.26% | 5.39e-06 | 8.72 | 3.60e-03 | 94.05% | 99.64% | 94.63% | 17.11% | 1.73e-07 | 15.93 | 2.55e-01 |
| $D_{59}$ | 99.71% | 99.73% | 98.52% | 87.68% | 12.44% | 6.34e-06 | 12.37 | 1.60e-06 | 98.58% | 98.53% | 85.01% | 10.85% | 3.20e-06 | 46.42 | 2.82e-05 |
| $D_{61}$ | 99.26% | 99.45% | 98.26% | 87.61% | 12.48% | 1.79e-05 | 12.00 | 1.69e-06 | 98.33% | 97.40% | 85.59% | 11.11% | 1.63e-02 | 41.64 | 9.60e-02 |
| $S_5$ | 100.00% | 99.99% | 94.14% | 88.91% | 12.13% | 1.02e-05 | 11.72 | 2.21e-07 | 99.84% | 99.97% | 85.28% | 10.23% | 1.43e-07 | 17.77 | 4.44e-09 |
| $S_6$ | 99.65% | 99.78% | 93.67% | 86.38% | 8.98% | 4.95e 05 | 12.17 | 2.66e 06 | 99.94% | 99.93% | 86.32% | 9.35% | 2.21e 06 | 291.67 | 1.05e 06 |
| $A_5$ | 99.04% | 99.31% | 93.27% | 86.69% | 10.26% | 1.94e-05 | 9.82 | 5.28e-07 | 97.53% | 97.40% | 83.56% | 8.22% | 4.88e-02 | 19.76 | 7.70e-04 |

# Strong Universality



# Takeaways

- Models naturally learn representation theory.
- Mechanistic interpretability is useful.
- Reverse engineering a single network is insufficient for understanding behaviour in general.

# Further Work

- Reverse engineering more group theoretic tasks.
- Understanding universality better in algorithmic / realistic tasks.
- Understanding network inductive biases better.