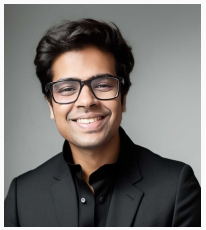
A Formal Framework For Length Generalization in Transformers

Xinting Huang[1*] Andy Yang[2*] Satwik Bhattamishra[3] Yash Sarrof[1]
Andreas Krebs[4] Hattie Zhou[5] Preetum Nakkiran[6] Michael Hahn[1]

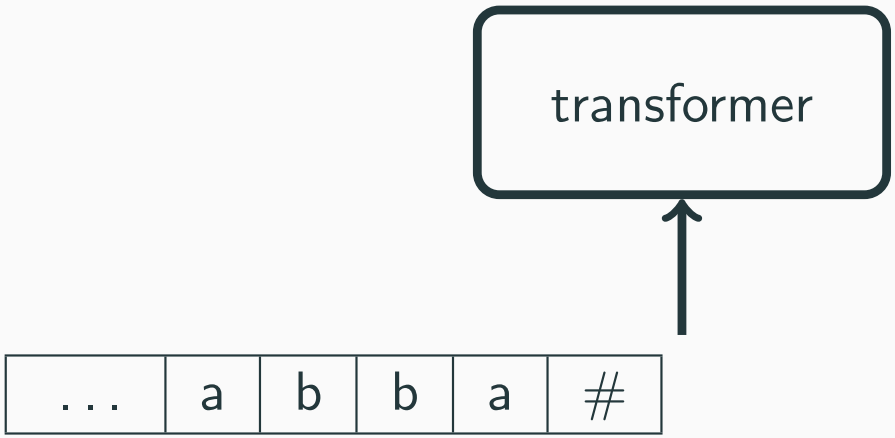
[1]Saarland University
[2]University of Notre Dame
[3]University of Oxford
[4]University of Tübingen
[5]Mila, Université de Montréal
[6]Apple



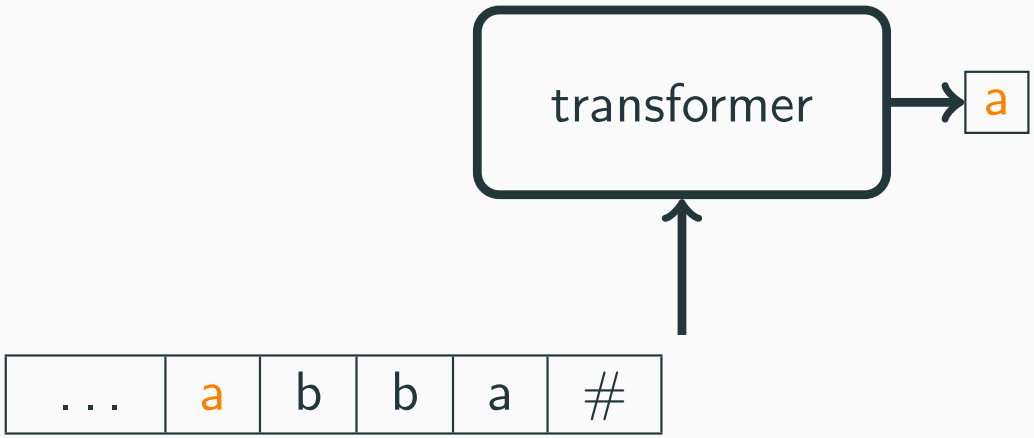





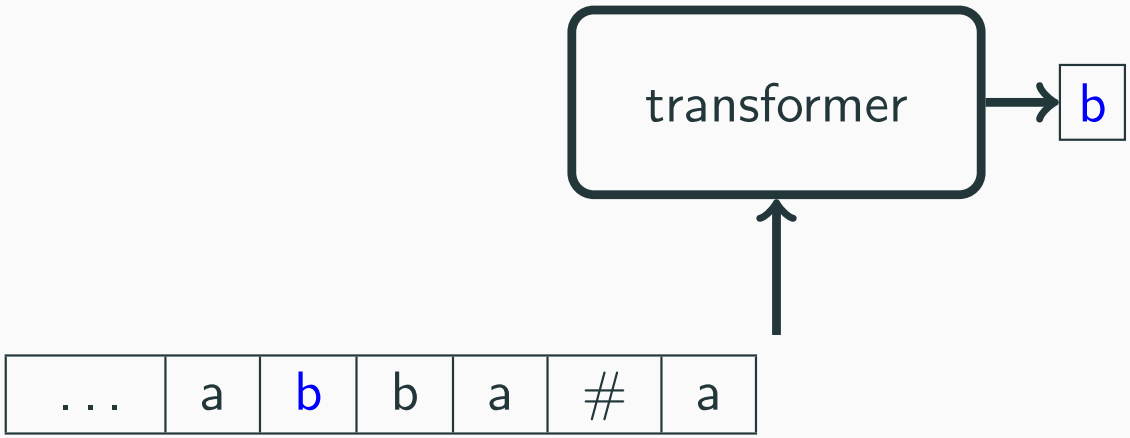
Copy



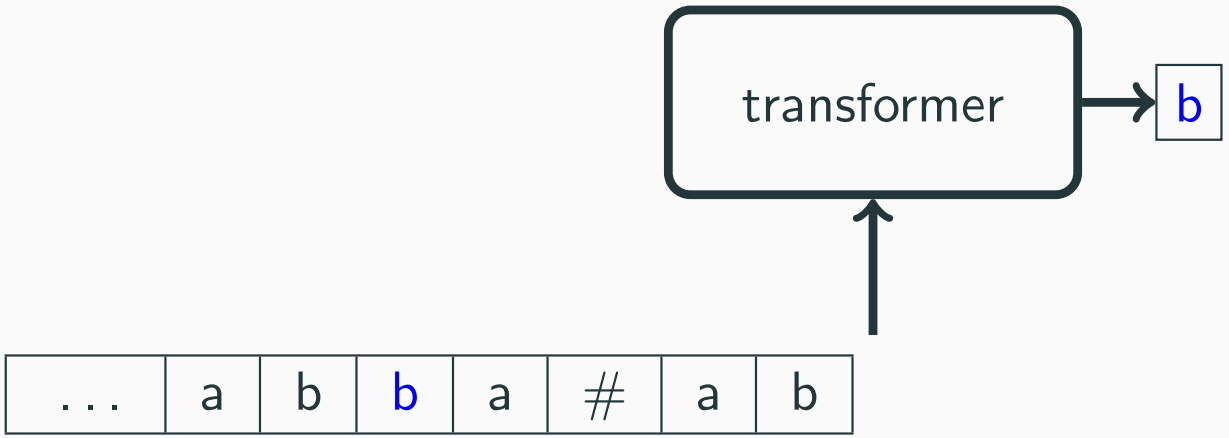
Copy



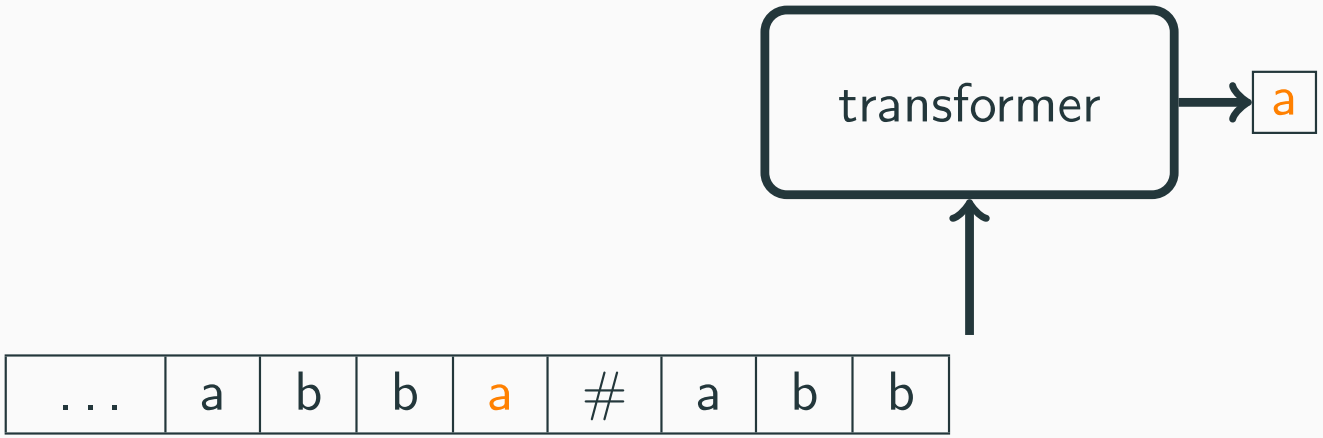
Copy



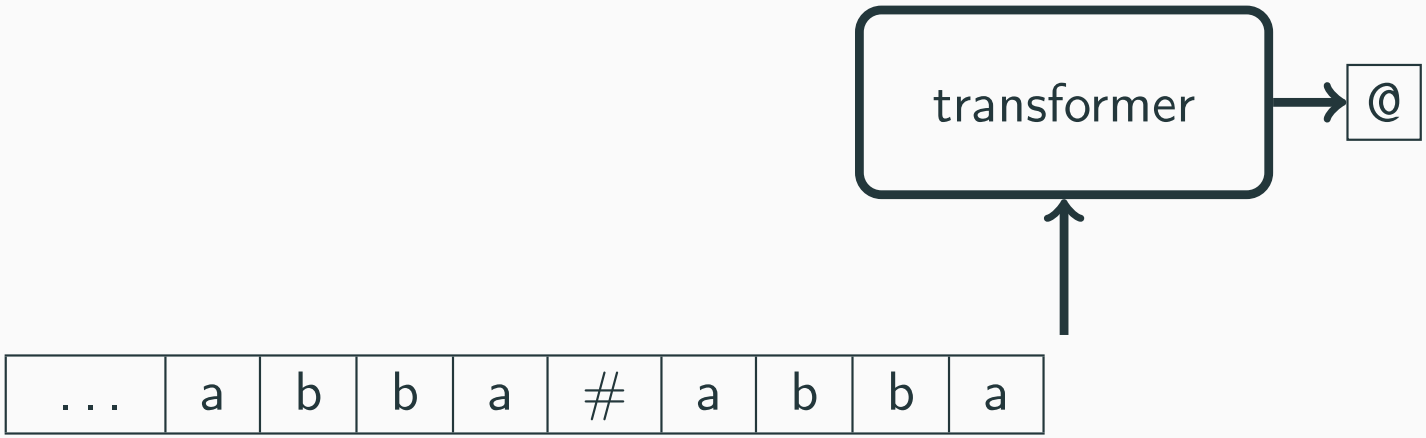
Copy



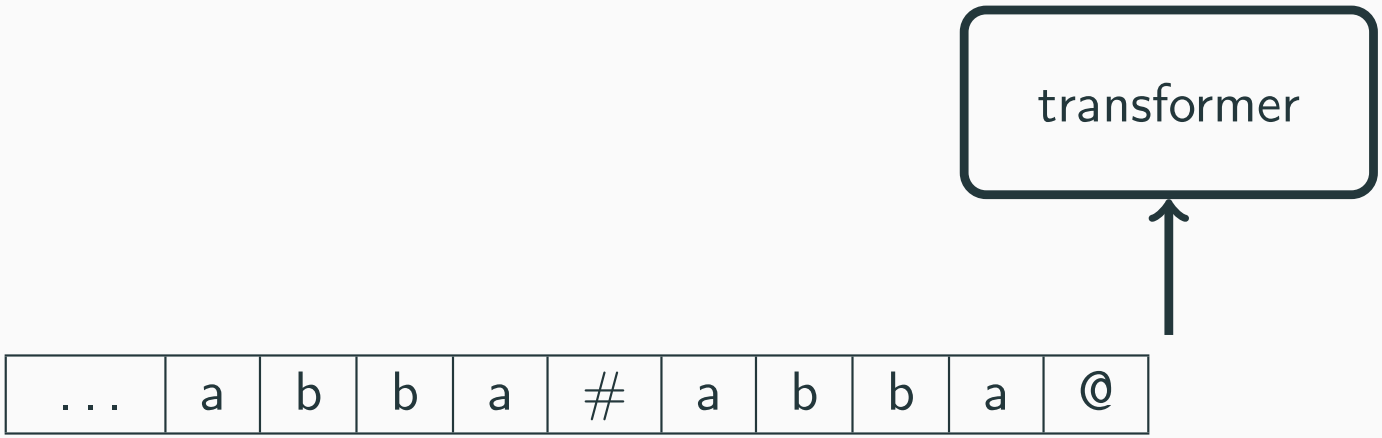
Copy



Copy

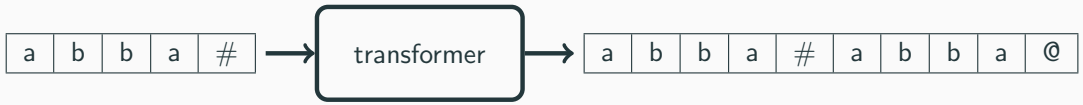


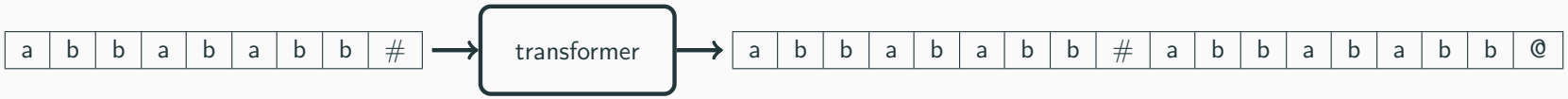
Copy



Length Generalizing on Copy

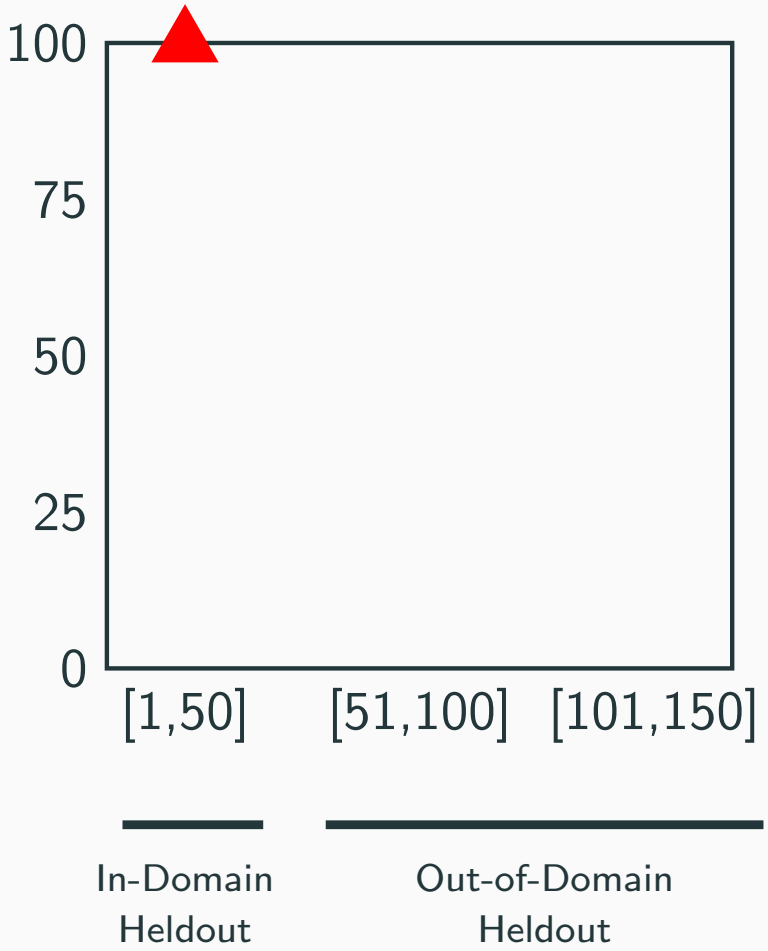
If we train on length up to N . . .





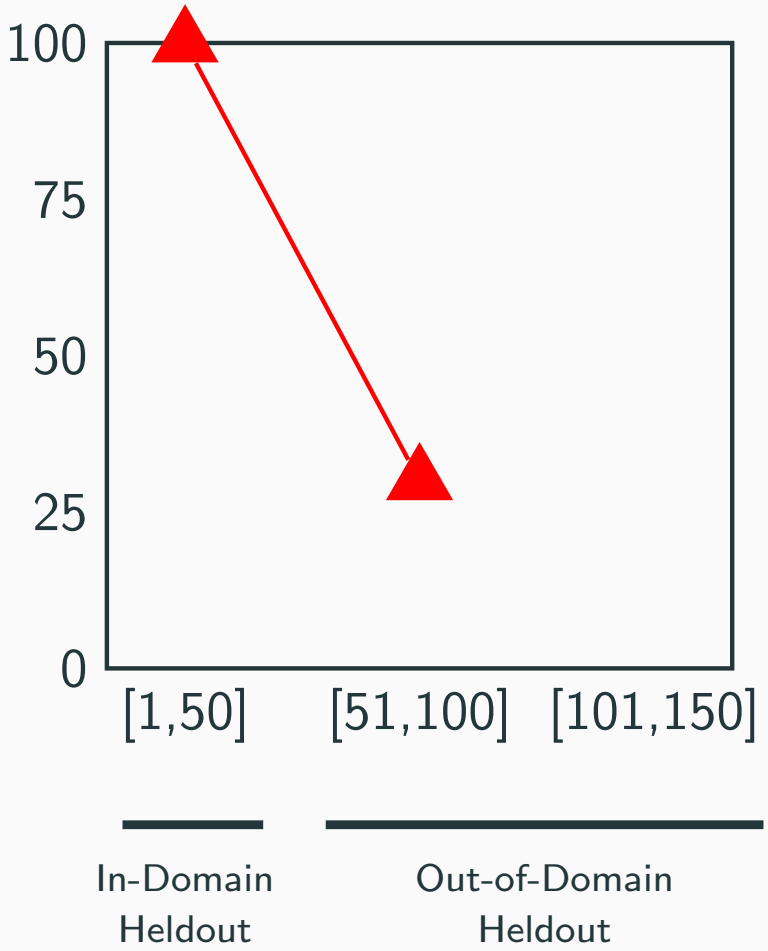
Will we succeed on lengths up to $2N$ . . .?

Length Generalizing on Copy



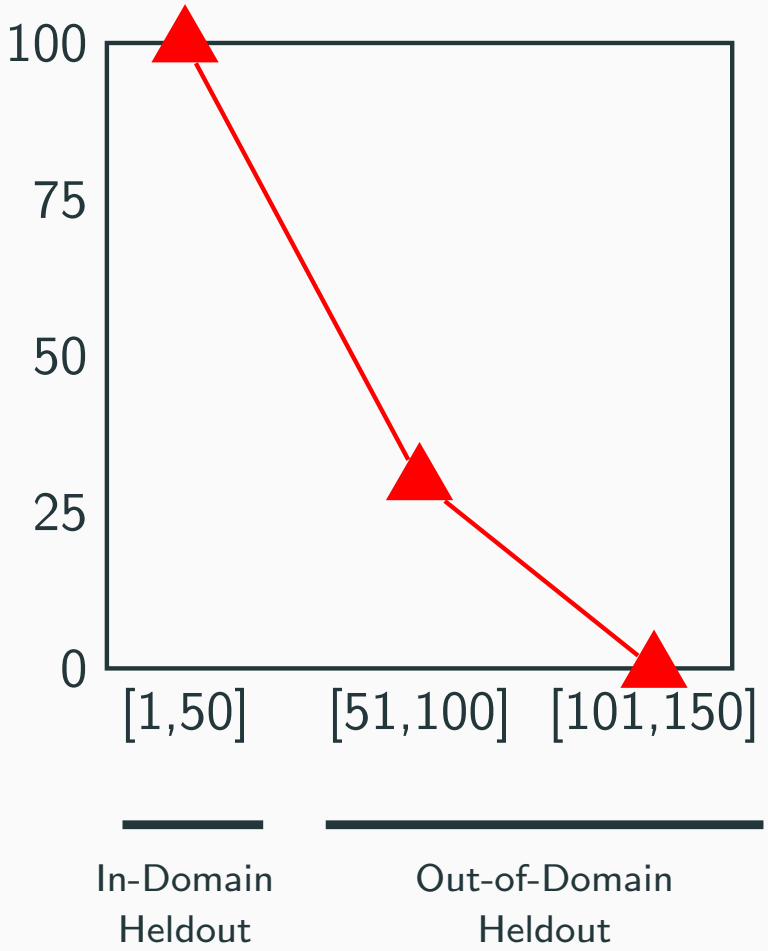
Zhou et al 2024 ICLR; Jelassi et al 2024 ICML; Kazemnejad et al 2023 NeurIPS.

Length Generalizing on Copy



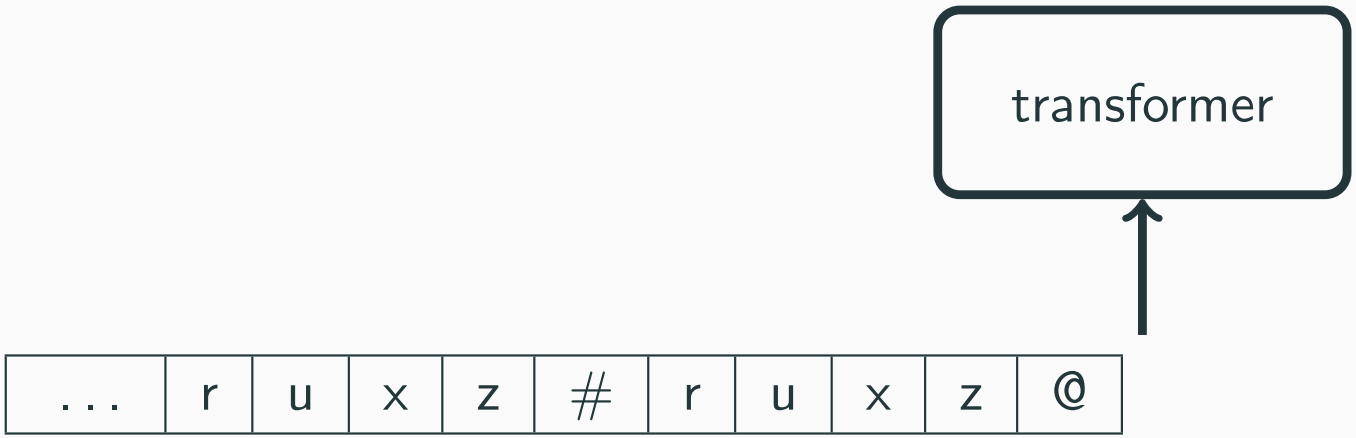
Zhou et al 2024 ICLR; Jelassi et al 2024 ICML; Kazemnejad et al 2023 NeurIPS.

Length Generalizing on Copy

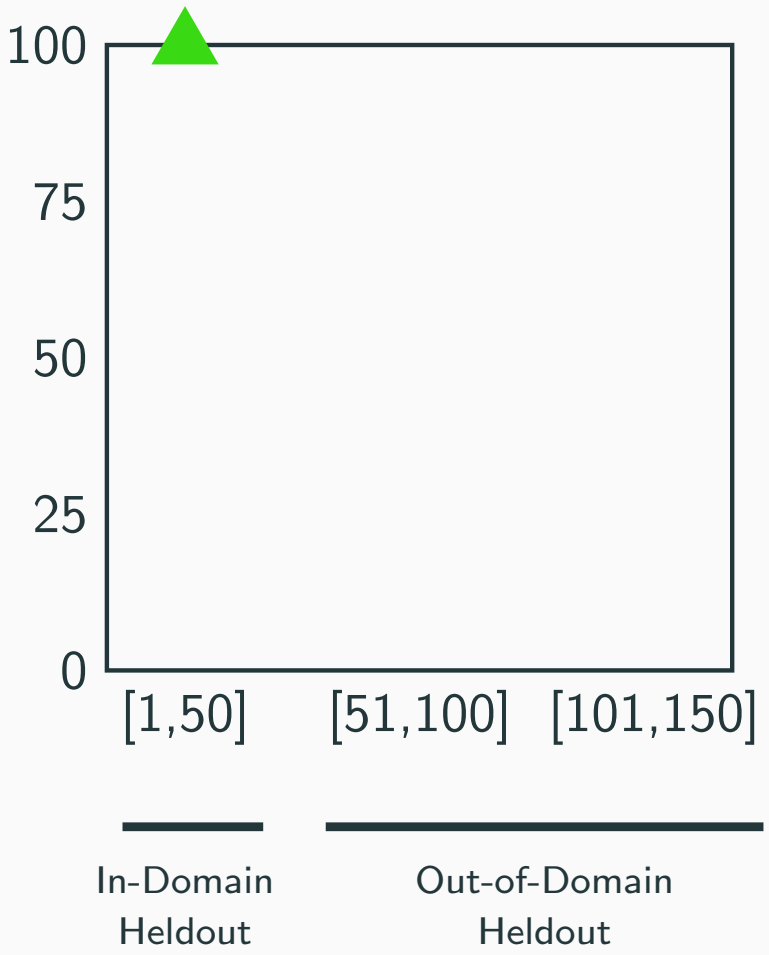


Zhou et al 2024 ICLR; Jelassi et al 2024 ICML; Kazemnejad et al 2023 NeurIPS.

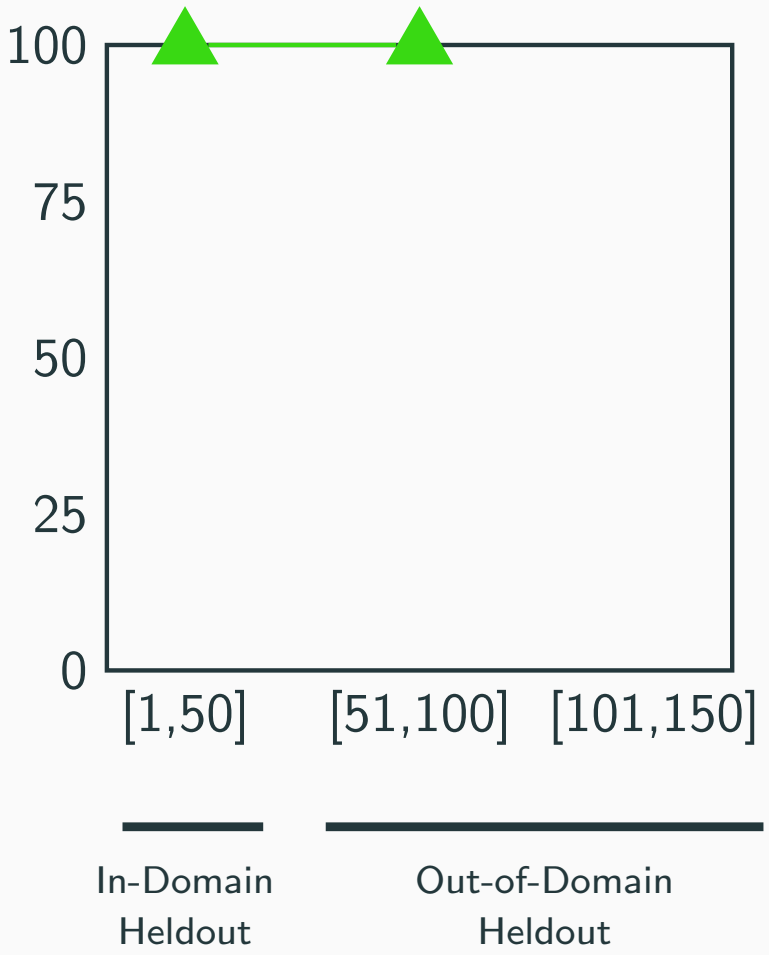
Unique Copy



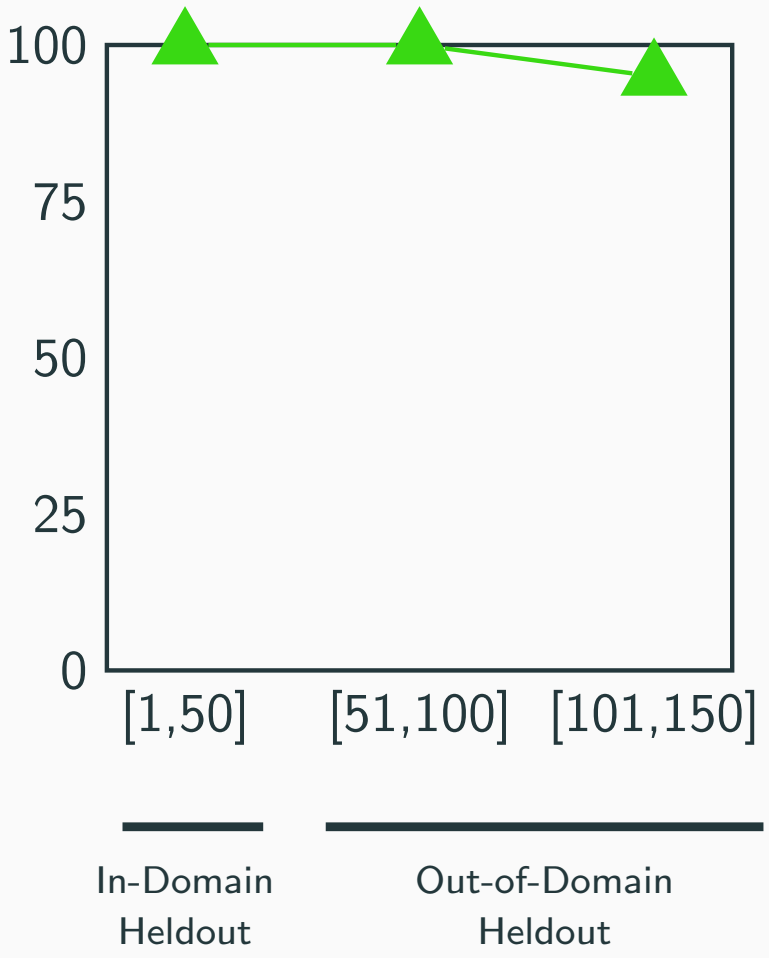
Length Generalizing on Unique Copy



Length Generalizing on Unique Copy



Length Generalizing on Unique Copy



On which tasks do transformers length-generalize?

What Algorithms can Transformers Learn? A Study in Length Generalization

Hattie Zhou Arwen Bradley Etai Littwin Noam Razin
Omid Saremi, Josh Susskind Samy Bengio Preetum Nakkiran

RASP-L Conjecture

RASP-L Conjecture (paraphrased): Transformers length-generalize on problems with simple programs in the RASP-L language.

RASP-L Conjecture

RASP-L Conjecture (paraphrased): Transformers length-generalize on problems with simple programs in the RASP-L language.

Challenge: RASP-L hasn't been formalized. Expressiveness not understood

RASP-L Conjecture

RASP-L Conjecture (paraphrased): Transformers length-generalize on problems with simple programs in the RASP-L language.

Challenge: RASP-L hasn't been formalized. Expressiveness not understood

Our Contribution: Formalize it based on C-RASP.

C-RASP Example: Majority

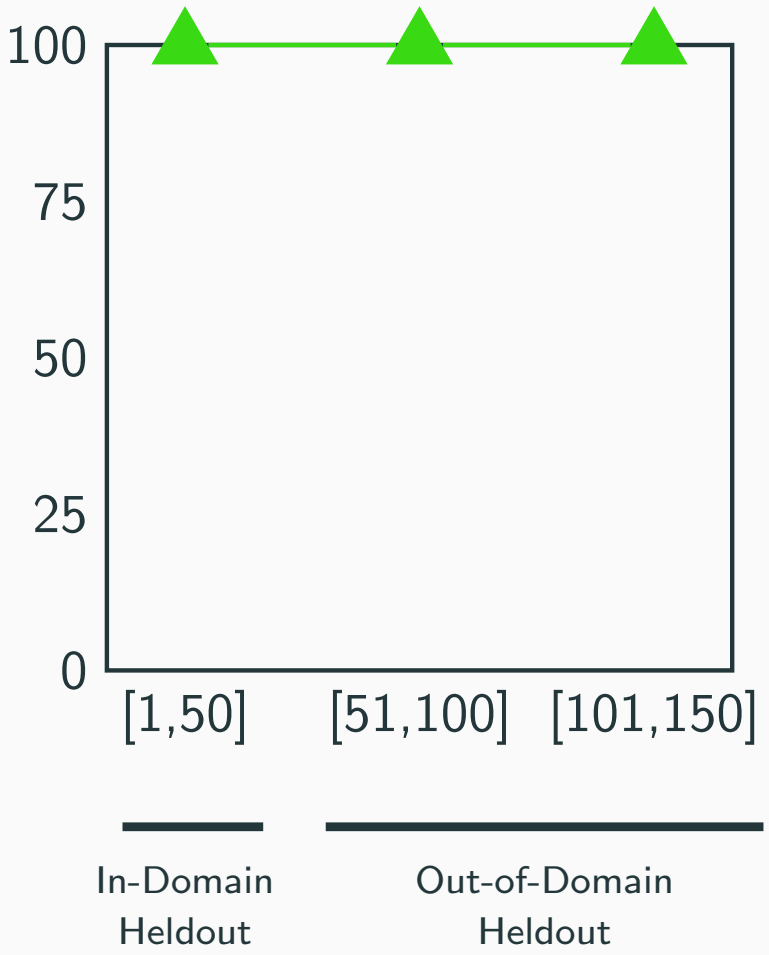
MAJORITY

$$C_1(i) := \#[j \leq i]\ Q_1(j) \qquad \text{count } \# \text{ of 1's} \quad (1)$$

$$C_0(i) := \#[j \leq i]\ Q_0(j) \qquad \text{count } \# \text{ of 0's} \quad (2)$$

$$M(i) := C_1(i) \geq C_0(i) \qquad \text{compare them} \quad (3)$$

Length Generalizing on Majority



C-RASP Example: $(aa)^*$

$(aa)^*$

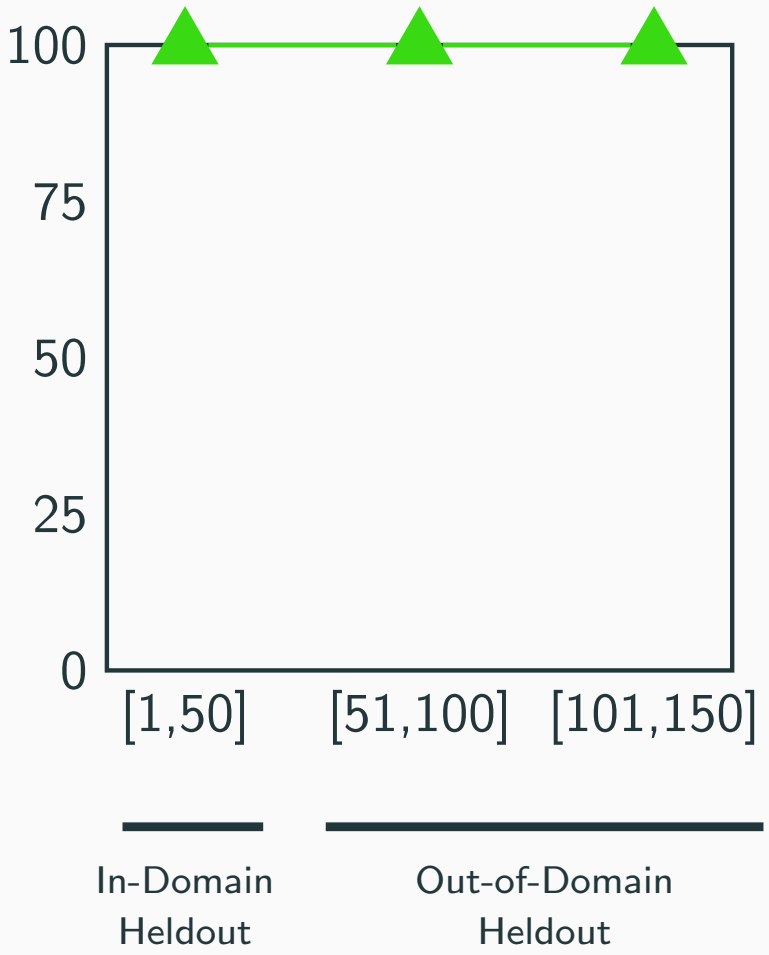
$$\phi(i) := i \equiv 0 \mod 2 \qquad \text{check position modulo 2} \quad (1)$$

$$C_{\neg a}(i) := \#\,[j \leq i]\ \neg Q_a(j) \qquad (2)$$

$$A(i) := C_{\neg a}(i) = 0 \qquad \text{check that no } b\text{'s occur} \quad (3)$$

$$D(i) := \phi(i) \wedge A(i) \qquad (4)$$

Length Generalizing on $(aa)^*$



C-RASP Example: Unique Copy

Unique Copy (Induction Head)

$$\vdots \tag{1}$$

$$CP_a(i) := \#\,[j \le i, j = i-1]\;\; Q_a(j) \tag{2}$$

$$PRED_a(i) := CP_a(i) \ge 1 \qquad \text{check preceding token} \tag{3}$$

$$\vdots \tag{4}$$

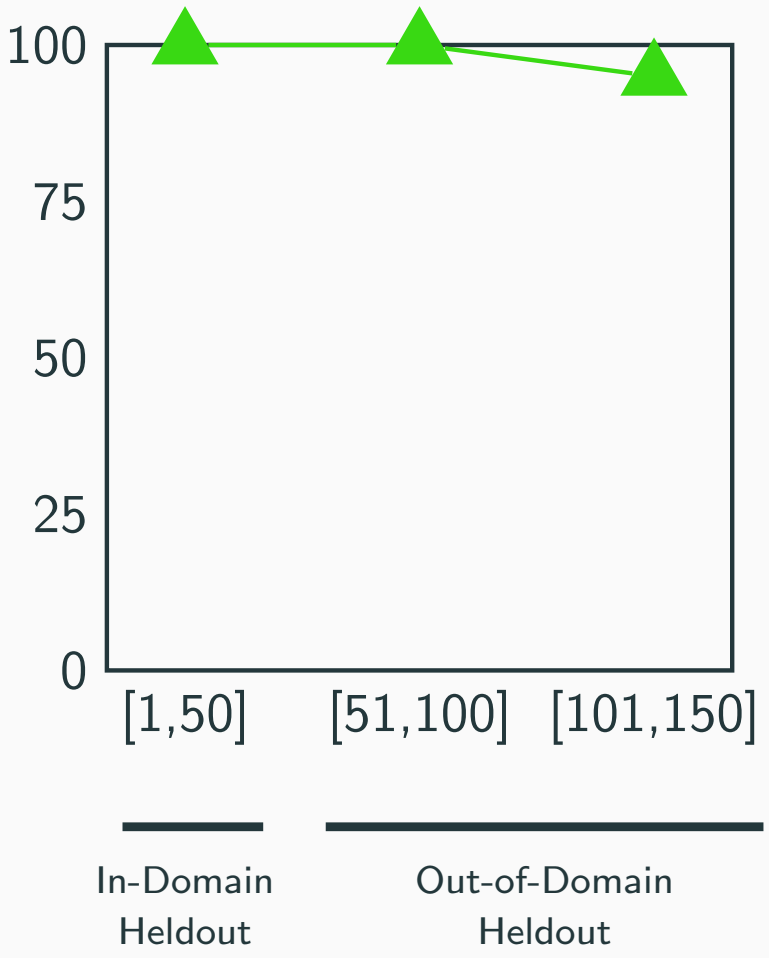
$$CBIGRAM_{ab} := \#\,[j \le i]\;\; Q_b(j) \wedge PRED_a(j) \qquad \text{collect bigrams} \tag{5}$$

$$EXISTS_{ab} := CBIGRAM_{ab}(i) \ge 1 \tag{6}$$

$$\vdots \tag{7}$$

$$NEXT_a(i) := \bigvee_{\sigma \in \Sigma} [Q_\sigma(i) \wedge EXISTS_{\sigma a}(i)] \qquad \text{check if bigram matches} \tag{8}$$

Length Generalizing on Unique Copy



C-RASP

Boolean-Valued Operations	
Initial	$P(i) := Q_\sigma(i)$ for $\sigma \in \Sigma$
Boolean	$P(i) := \neg P_1(i)$ $P(i) := P_1(i) \wedge P_2(i)$
Constant	$P(i) := \top$
Positional	$P(i) := \phi(i)$ for $\phi \in \Phi$
Comparison	$P(i) := C_1(i) \leq C_2(i)$

Count-Valued Operations	
Counting	$C(i) := \#\left[j \leq i, \psi(i,j)\right]\ P(j)$ for $\psi \in \Psi \cup \{\top\}$
Conditional	$C(i) := P(i)\ ?\ C_1(i)\ :\ C_2(i)$
Addition	$C(i) := C_1(i) + C_2(i)$
Subtraction	$C(i) := C_1(i) - C_2(i)$
Constant	$C(i) := 1$

Yang & Chiang 2024

C-RASP Non-Example

Copy With Repetition

$$\vdots \tag{1}$$

$$\vdots \tag{2}$$

$$??? \tag{3}$$

Provably no C-RASP program!

Rigorous proof via communication complexity.

The Main Result

Theorem (informal):

If f has a C-RASP program, then transformers trained on an idealized training procedure will length-generalize from N to $2N$, when N is large.

Inference Procedure

Definition: Inference Procedure

Let f be the target function. For $n = 1, 2, 3, \ldots$ choose a transformer T_n

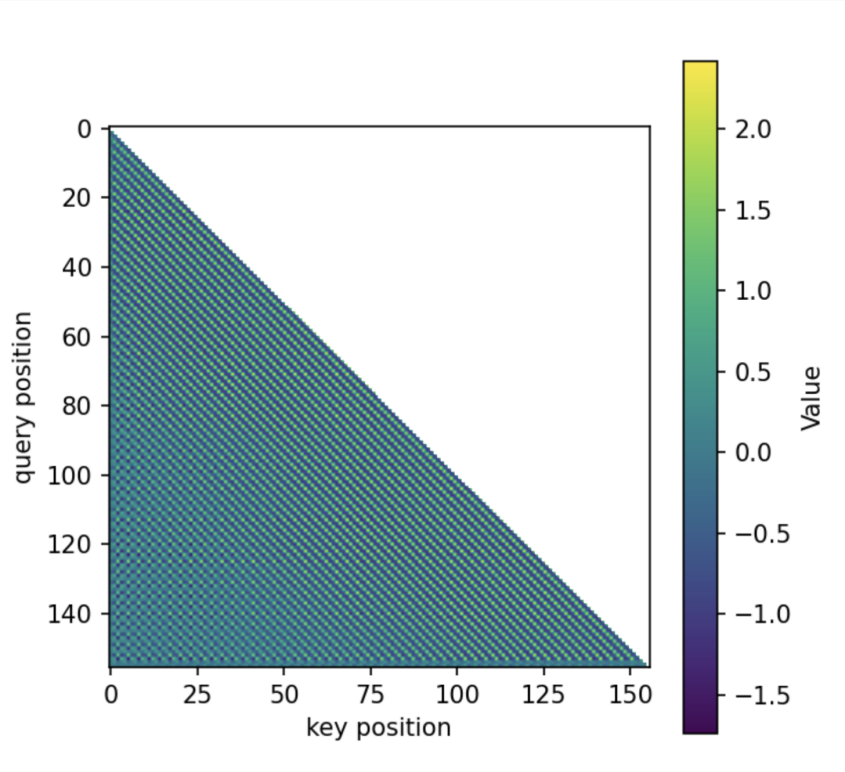
- that has context window n
- matches f on all input strings with length $\leq \frac{n}{2}$
- while minimizing a regularizer

This procedure converges on a length-generalizing transformer. There is some N such that, for $n > N$, T_n matches f on all strings up to length n.

Translation Invariance

Our hypothesis class assumes translation invariance in transformers, following the APE set up.

While not enforced by SGD, we argue translation invariance is helpful for transformers to length generalize

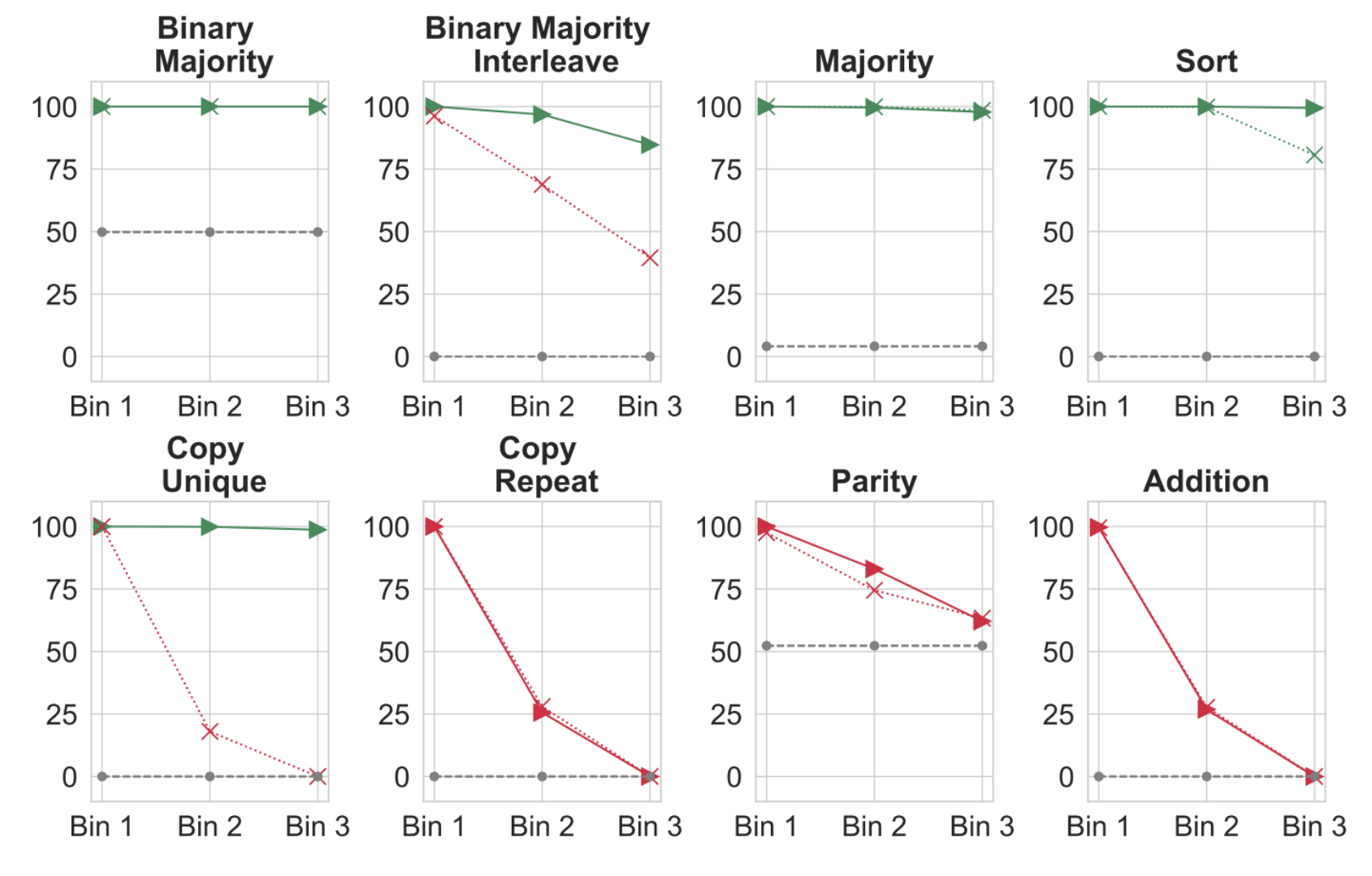


Regularizer

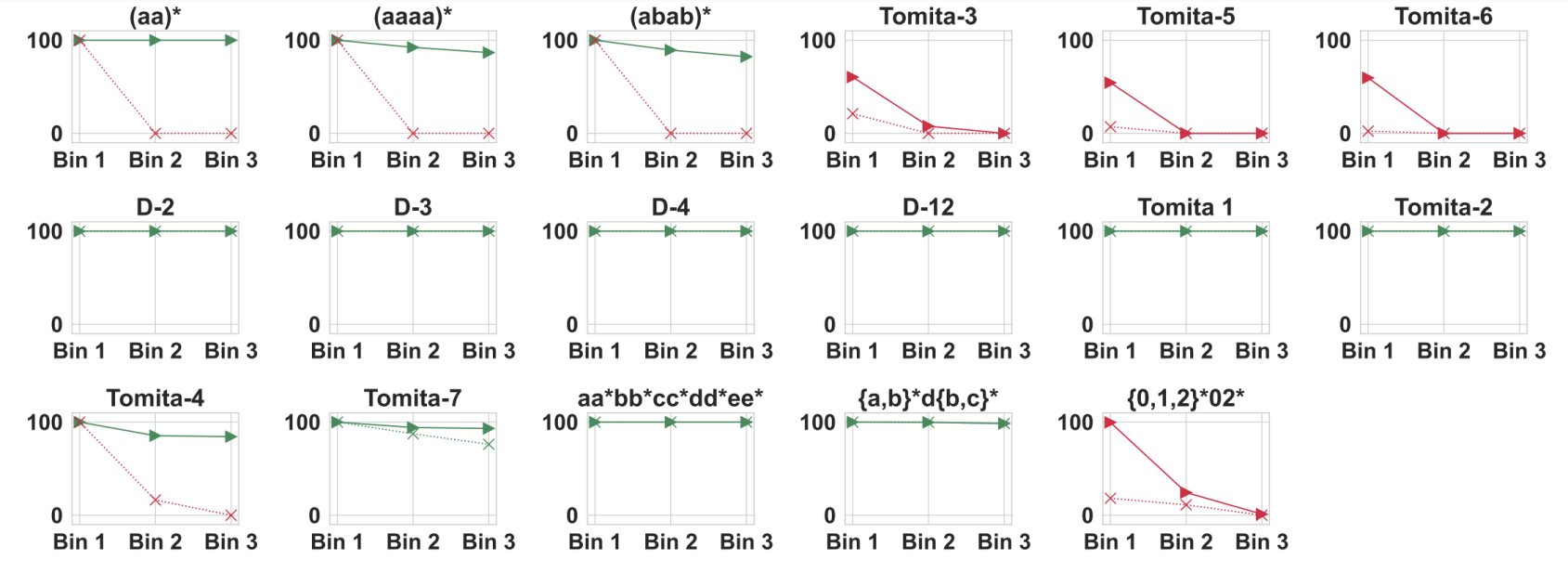
Definition (Regularizer)
Define $\mathcal{R}(T)$ as the sum of (1) $L + H$; (2) the precision p used in the hypothesis class; the precision p used for rounding attention logits and the output of $\exp(\cdot)$; (3) $\max_{l,h} rank(\boldsymbol{V}_{l,h})$; (4) $\max_{l,h} \|\boldsymbol{K}_{l,h}^T \boldsymbol{Q}_{l,h}\|$; $\max_{l,h} \|\boldsymbol{V}_{l,h}\|$; $\max_l \|\boldsymbol{A}_l\|_F$, $\|\boldsymbol{B}_l\|_F$; $\|\boldsymbol{U}\|$; (5) $\max_i \|\boldsymbol{p}_i\|_2$, $\max_\sigma \|\boldsymbol{E}_\sigma\|_2$, $\max_l \|\boldsymbol{b}_l\|_2$; (6) the term

$$\sum_{l=1}^{L} \sum_{h=1}^{H} \sum_{j=1}^{n} \left| \boldsymbol{p}_1^T \boldsymbol{K}_{l,h}^T \boldsymbol{Q}_{l,h} \boldsymbol{p}_j \right|^2 \tag{4}$$

Algorithmic Tasks



Formal Language Tasks



Bhattamishra et al. 2020

In Summary

On which tasks do transformers length-generalize?

Theorem (informal): If f has a C-RASP program, then transformers trained on an idealized training procedure will length-generalize from N to $2N$, when N is large.

MAJORITY

$$C_1(i) := \#[j \le i]\ Q_1(j) \quad (1)$$
$$C_0(i) := \#[j \le i]\ Q_0(j) \quad (2)$$
$$M(i) := C_1(i) \ge C_0(i) \quad (3)$$