Recurrent Transformers Trade-off Parallelism for Length Generalization on Regular Languages

Paul Soulos, Aleksandar Terzić, Michael Hersche, Abbas Rahimi

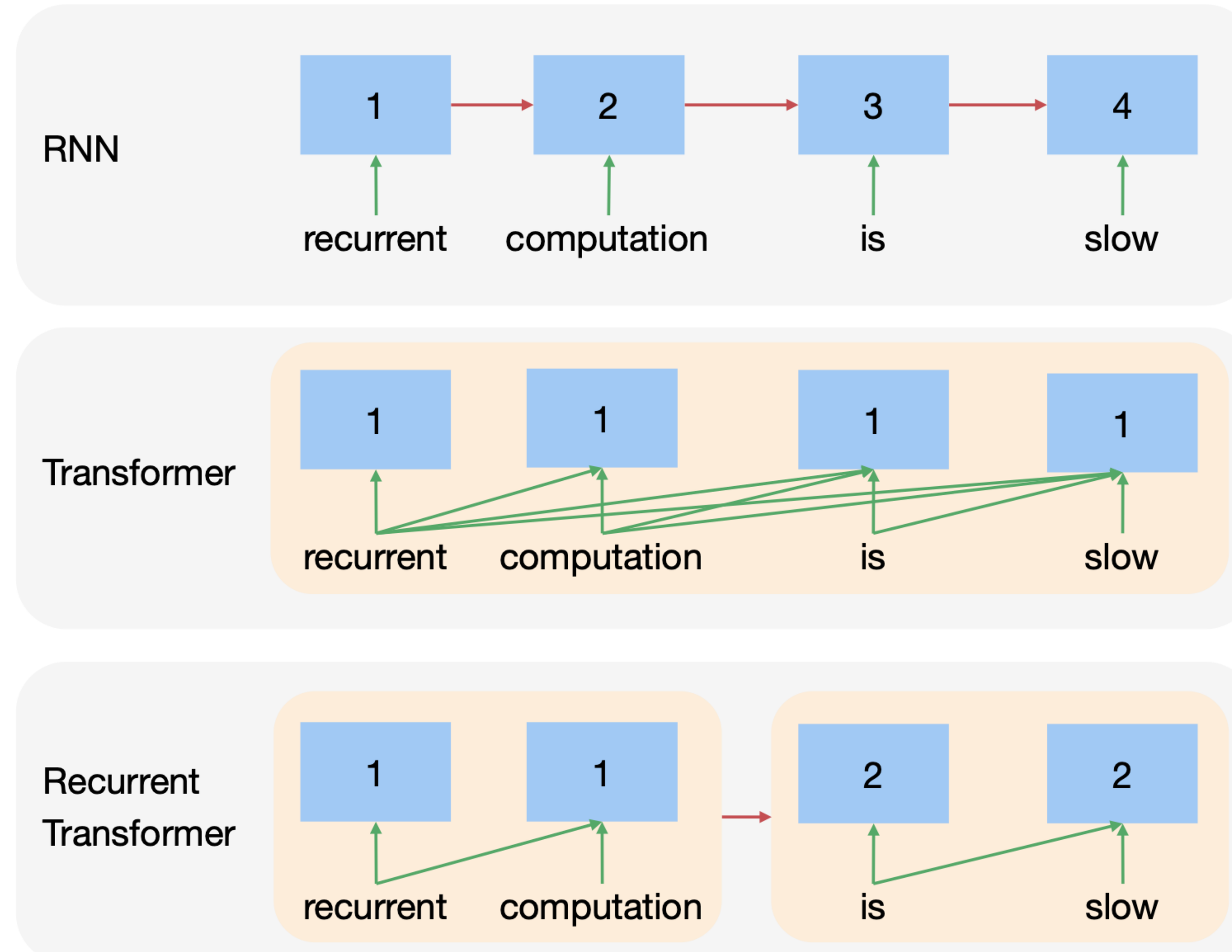
psoulos1@jh.edu

IBM Research · Johns Hopkins University · ETH zürich

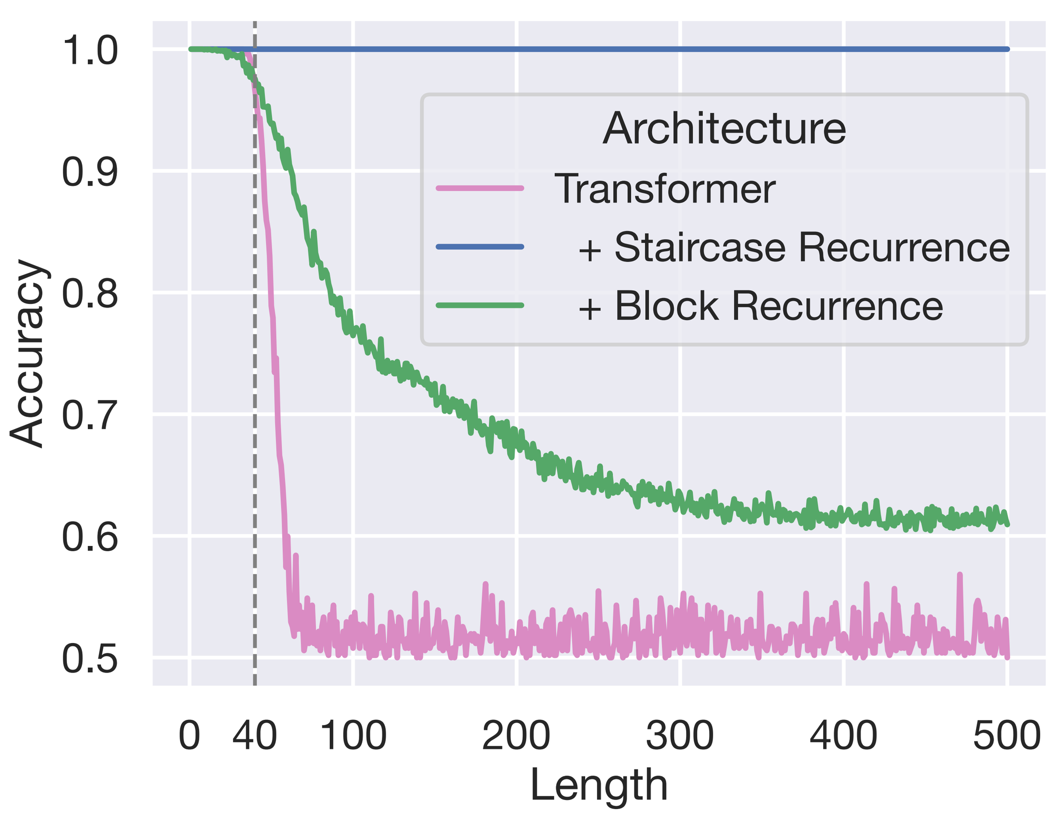
PAPER · CODE

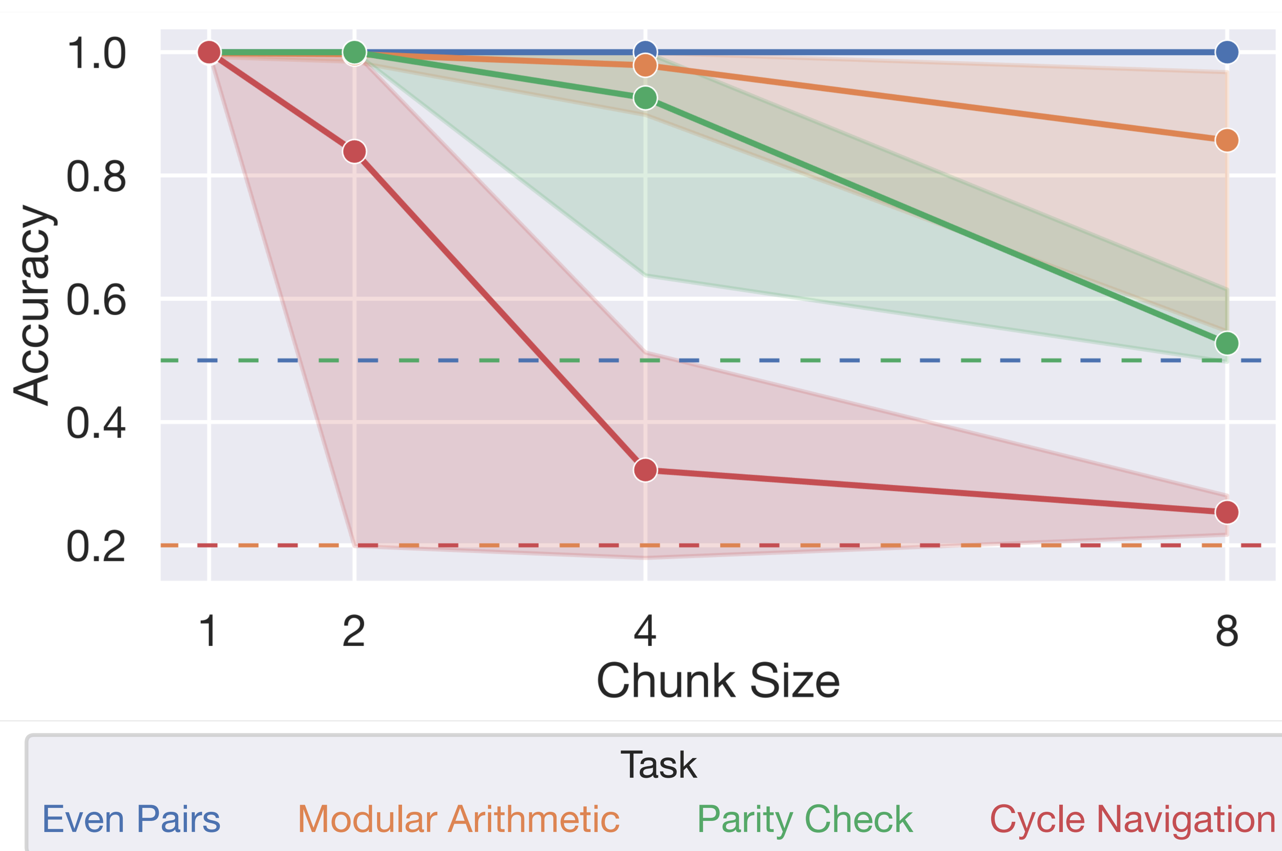
Goal: improve the Transformer's ability to model Regular Languages and generalize out-of-distribution.

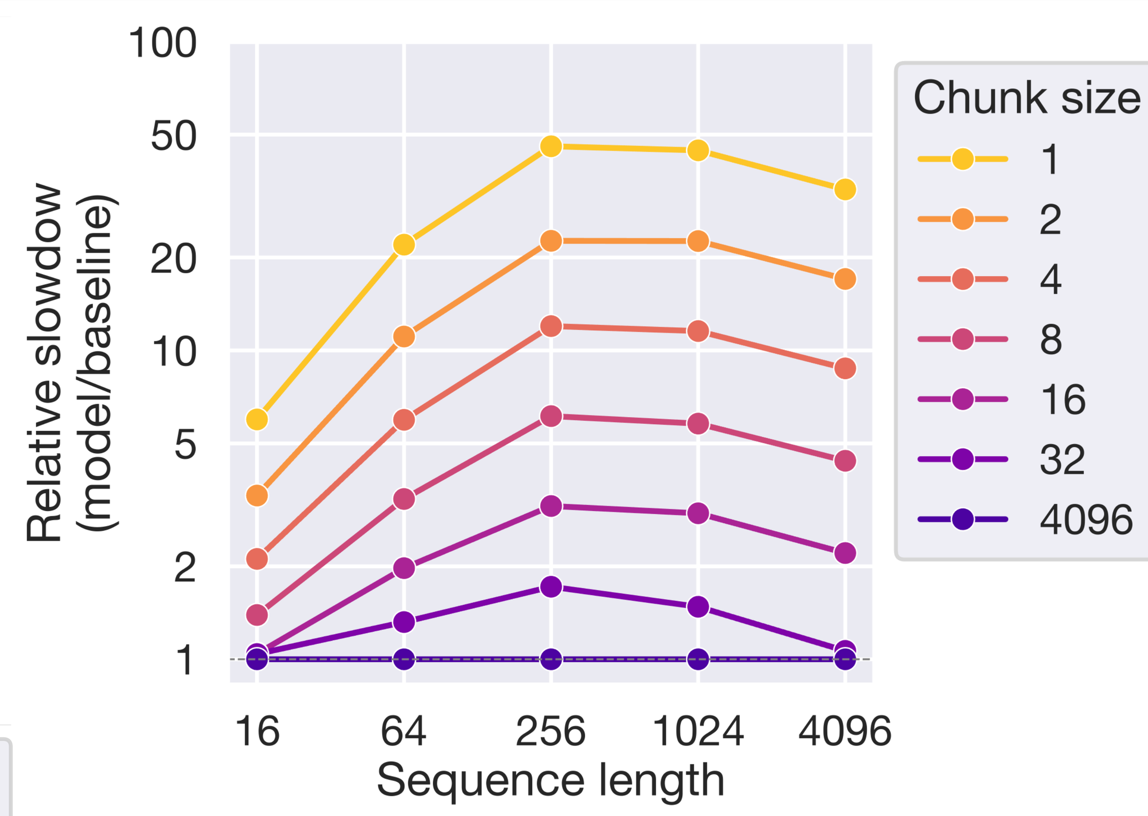
Task	RNN	Transformer
Even Pairs	100.0	96.4
Modular Arithmetic	100.0	24.2
Parity Check	100.0	52.0
Cycle Navigation	100.0	61.9



Staircase recurrence can model Regular Languages by is sensitive to chunk size.







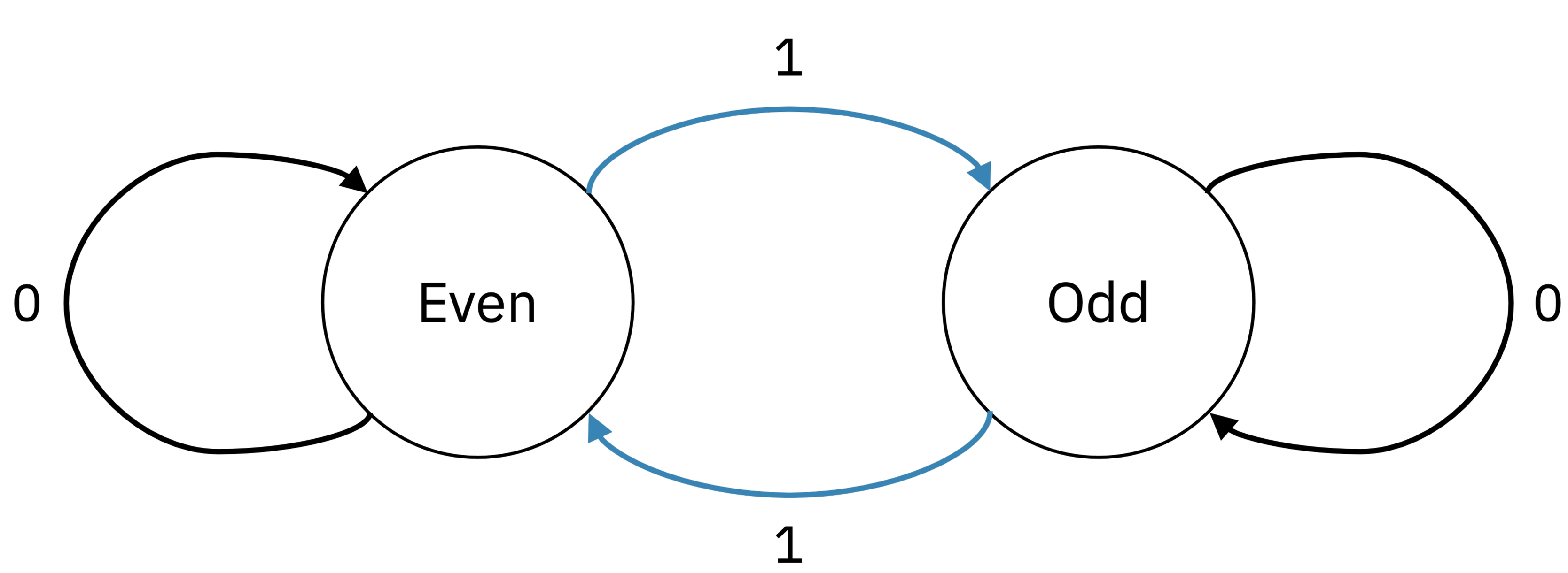
Only Staircase Recurrence achieves good out-of-distribution generalization.

Staircase Recurrence is highly sensitive to chunk size for good OOD generalization.

Staircase recurrence is impractical for pre-training because it is up to 50x slower.

Why do Transformers fail on Regular Languages?

Regular Languages are the simplest type of computation in the Chomsky Hierarchy and are equivalent to the class of problems solved by Finite State Machines.



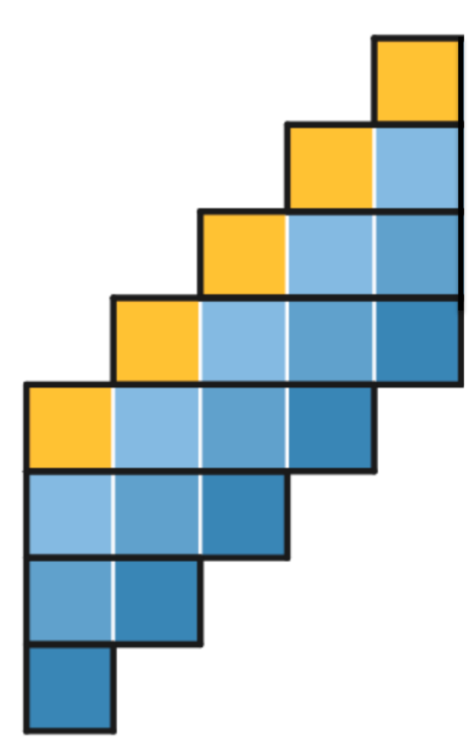
Parity Check is an example of a simple Regular Language where the task is to count whether the number of 1s in a binary string is even or odd.

Recurrent Neural Networks (RNNs) generalize well on Regular Languages. For a sequence of T tokens, RNNs perform T computation steps, allowing this architecture to maintain and update state at each step. In contrast, standard **Transformers** are non-recurrent, and a Transformer with L layers will process T tokens in L computation steps, regardless of the sequence length.

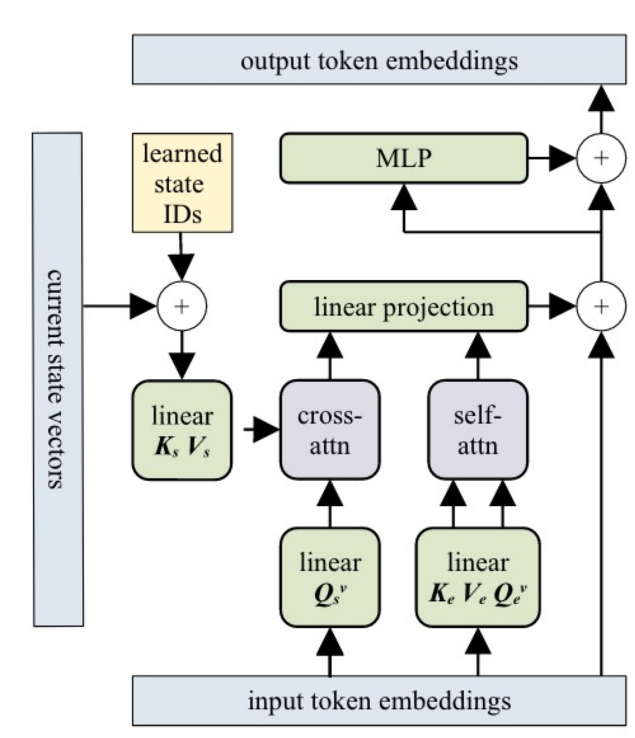
Adding recurrence to Transformers

Recurrent Transformers balance throughput by processing a **chunk** of tokens in parallel and add recurrent connections between chunks. We evaluate two types of Recurrent Transformers, Staircase Recurrence (Ju et al, 2022) and Block Recurrence (Hutchins et al, 2022).

Staircase Recurrence



Block Recurrence



Defining token and chunk recurrence

Token-layer Recurrence is the property that the output of token $X_{i,l}$ at position i after layer l depends on the output of a previous token at layer l:

$$X_{i,l} = f\left(X_{<i,l}, \ldots\right) \forall\, i > 0$$

Chunk-layer Recurrence is the property that the output of chunk $K_{i,l}$ at position i after layer l depends on the output of a previous chunk at layer l:

$$K_{i,l} = f\left(K_{<i,l}, \ldots\right) \forall\, i > 0$$

RNNs **satisfy** token-layer recurrence, whereas Transformers **do not satisfy** token-layer recurrence. For a Transformer,

$$X_{i,l} = f\left(X_{<i,l-1}, \ldots\right) \forall\, i > 0$$