The Neural Data Router: Adaptive Control Flow in Transformers Improves Systematic Generalization

Róbert Csordás, Kazuki Irie, Jürgen Schmidhuber

USI/SUPSI
Istituto
Dalle Molle
di studi
sull'intelligenza
artificiale
IDSIA

ICLR

جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology

ICLR 2022

Systematic generalization

- Ability to solve problems that are governed by **novel combinations** of rules seen during the training.
  - Novel combination of known constituents (systematicity)
  - **Generalization to longer problems (productivity)**
- Learning generally applicable rules instead of pure pattern matching

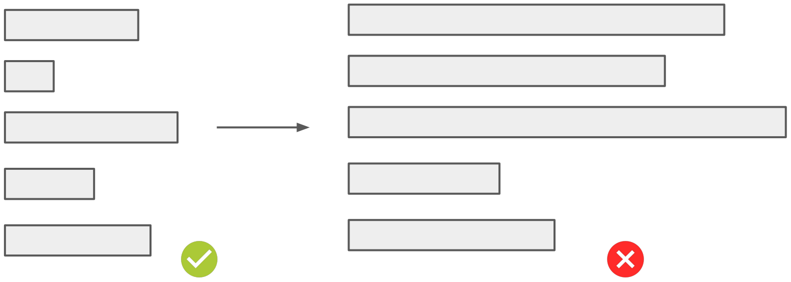
Probably one of the major obstacles toward general AI

1+2 → 3

3*3 → 9

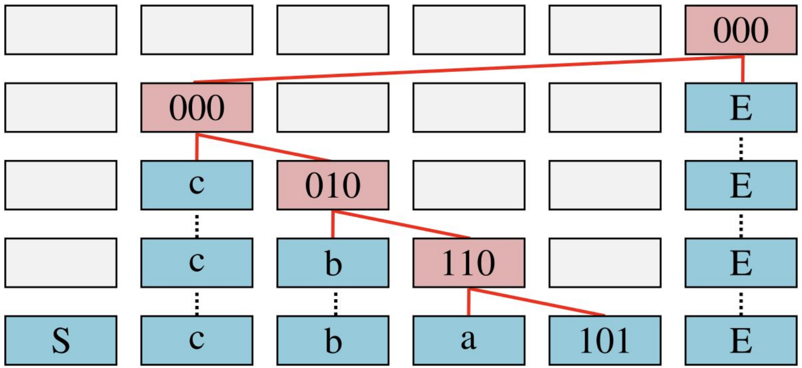
(1+1)*2 → 4

(1+2)*3 → ?



Revisiting Transformers

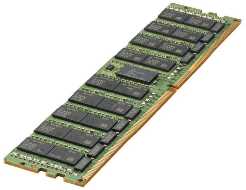
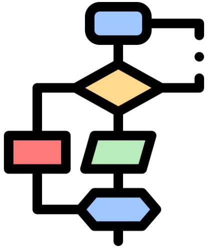
- Transformers have a structure seemingly well-suited for the task
- They should be able to build a computation graph in their layers



So why do they perform so badly?

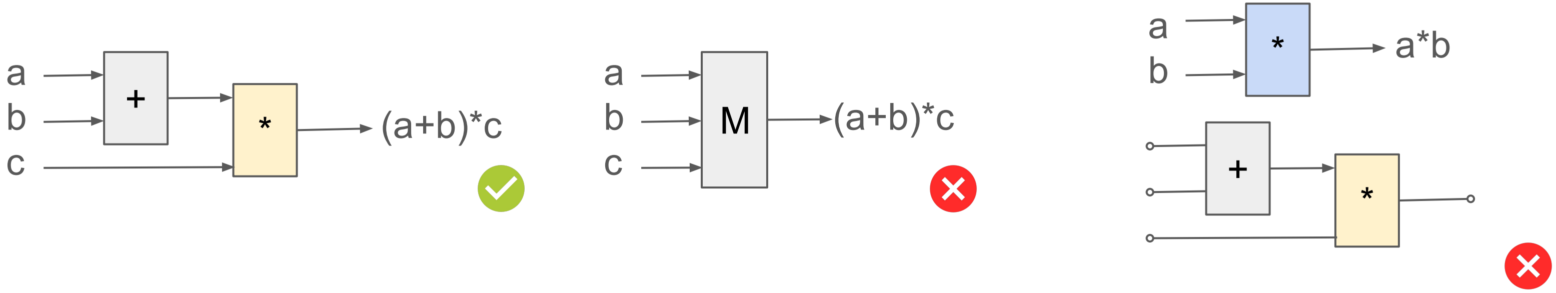
What’s missing?

- There is no optimization pressure for generalization
- Only hope: inductive biases
- But what biases?

	Memorize	Learn the rules
		
Train set	✓	✓
Test set - generalization	✗	✓

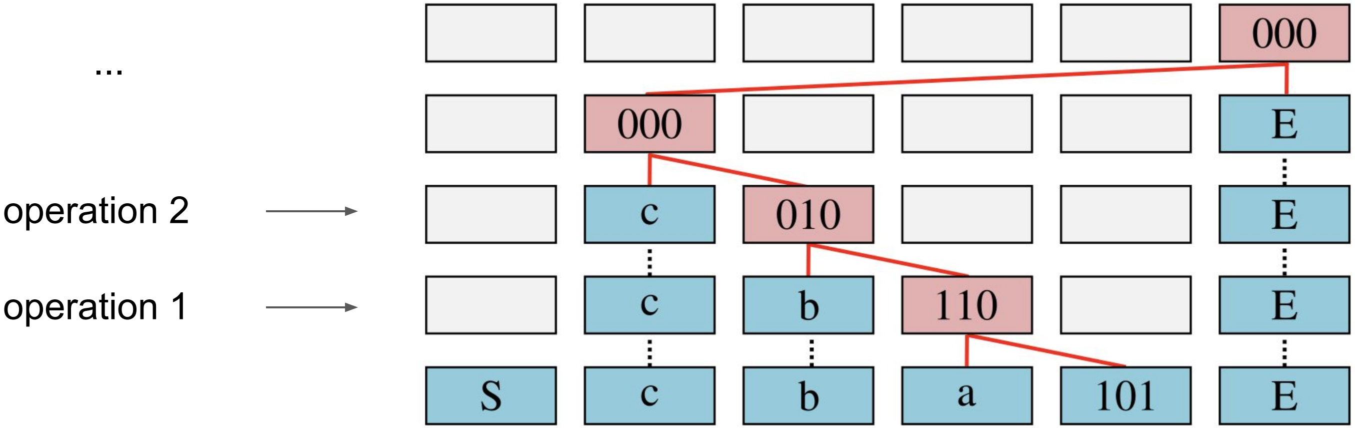
Hypothesis 1

- The basis of generalization should be **compositionality**
- Decomposing the problem into elementary, reusable components should boost generalization



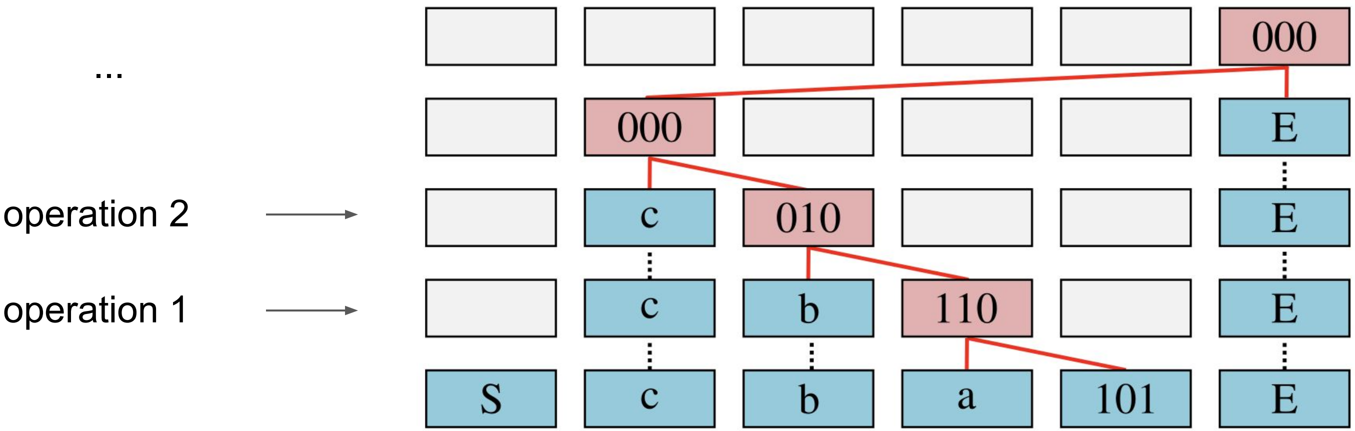
Hypothesis 1

- In Transformers, the output of an operation is available only to the next layer
- For composition, all levels should have all functions available
  to enable compositions of elementary functions in any *arbitrary orders*



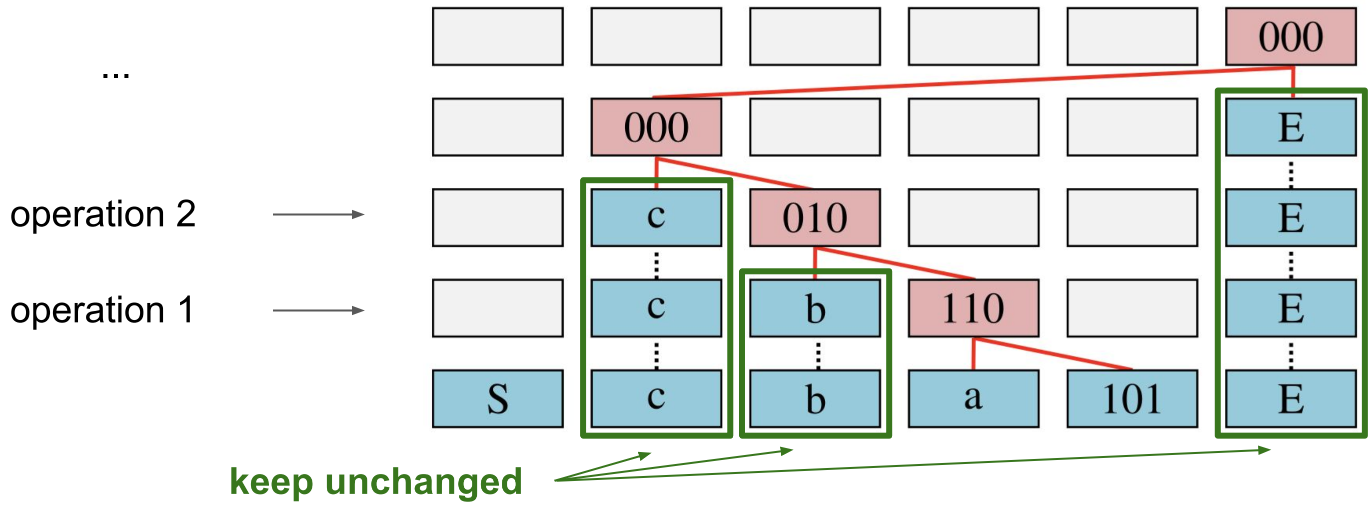
Hypothesis 1

- Should use shared layers
- Should use many layers
    - At least as many as the depth of the “computation graph”



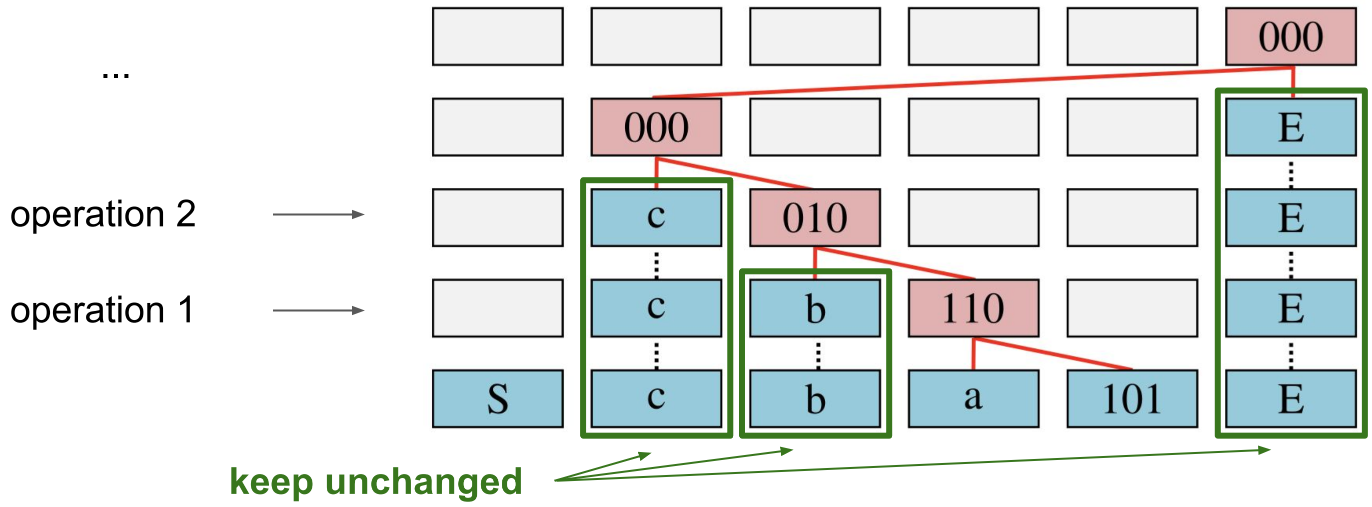
Hypothesis 2

- An operation should be performed only if its inputs are ready
- Otherwise, the columns should remain unchanged



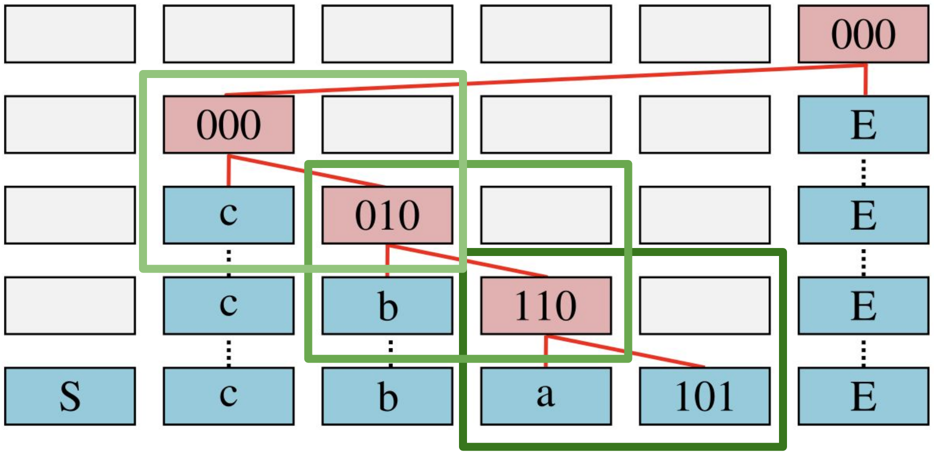
Hypothesis 2

- Transformers is not well suited for keeping the states unchanged
  - Layernorm makes it especially difficult
- Add a mechanism to bias towards keeping the states unchanged until it's their turn to be processed



Hypothesis 3

- Long compositions are often made of multiple local compositions
- Bias, but not restrict to local computation



Copy Gate

- Makes it easy to keep the columns unchanged
- Gating to skip the entire Transformer layer
- Adds a bias towards Hypothesis 2

***Copy gate**, not just yet another gate!*

Copy Gate

- Makes it easy to keep the columns unchanged
- Removes layernorm, and skips the whole transformation
- Adds a bias towards Hypothesis 2

$$\boldsymbol{a}^{(i,t+1)} = \mathrm{LayerNorm}(\mathrm{MultiHeadAttention}(\boldsymbol{h}^{(i,t)}, \mathbf{H}_t, \mathbf{H}_t) + \boldsymbol{h}^{(i,t)}) \quad \leftarrow \text{standard Transformer}$$

$$\hat{\boldsymbol{h}}^{(i,t+1)} = \mathrm{LayerNorm}(\mathrm{FFN}^{\mathrm{data}}(\boldsymbol{a}^{(i,t+1)})) \quad \leftarrow \text{no residual, otherwise like standard Transformer}$$

***Copy gate**, not just yet another gate!*

Copy Gate

- Makes it easy to keep the columns unchanged
- Removes layernorm, and skips the whole transformation
- Adds a bias towards Hypothesis 2

$$\boldsymbol{a}^{(i,t+1)} = \mathrm{LayerNorm}(\mathrm{MultiHeadAttention}(\boldsymbol{h}^{(i,t)}, \mathbf{H}_t, \mathbf{H}_t) + \boldsymbol{h}^{(i,t)})$$ ← standard Transformer

$$\hat{\boldsymbol{h}}^{(i,t+1)} = \mathrm{LayerNorm}(\mathrm{FFN}^{\mathrm{data}}(\boldsymbol{a}^{(i,t+1)}))$$ ← no residual, otherwise like standard Transformer

$$\boldsymbol{g}^{(i,t+1)} = \sigma(\mathrm{FFN}^{\mathrm{gate}}(\boldsymbol{a}^{(i,t+1)}))$$ ← gate, parallel branch to $\hat{\boldsymbol{h}}^{(i,t+1)}$

***Copy gate**, not just yet another gate!*

Copy Gate

- Makes it easy to keep the columns unchanged
- Removes layernorm, and skips the whole transformation
- Adds a bias towards Hypothesis 2

$$\boldsymbol{a}^{(i,t+1)} = \mathrm{LayerNorm}(\mathrm{MultiHeadAttention}(\boldsymbol{h}^{(i,t)}, \mathbf{H}_t, \mathbf{H}_t) + \boldsymbol{h}^{(i,t)})$$ ← standard Transformer

$$\hat{\boldsymbol{h}}^{(i,t+1)} = \mathrm{LayerNorm}(\mathrm{FFN}^{\mathrm{data}}(\boldsymbol{a}^{(i,t+1)}))$$ ← no residual, otherwise like standard Transformer

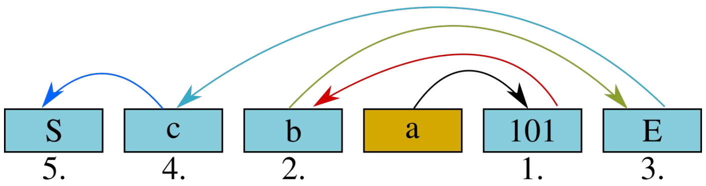
$$\boldsymbol{g}^{(i,t+1)} = \sigma(\mathrm{FFN}^{\mathrm{gate}}(\boldsymbol{a}^{(i,t+1)}))$$ ← gate, parallel branch to $\hat{\boldsymbol{h}}^{(i,t+1)}$

$$\boldsymbol{h}^{(i,t+1)} = \boldsymbol{g}^{(i,t+1)} \odot \hat{\boldsymbol{h}}^{(i,t+1)} + (1 - \boldsymbol{g}^{(i,t+1)}) \odot \boldsymbol{h}^{(i,t)}$$ ← highway-network style gate

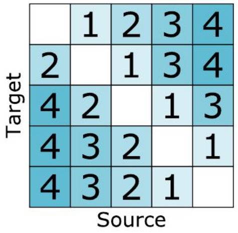
***Copy gate**, not just yet another gate!*

Geometric attention

- Bias towards attending to the *nearest* match
    - The distance does not matter
- Inspired by geometric distribution
- Define an order of preference of the positions, radiating out from each column



The order in which the geometric attention considers the nodes in a single layer. Shown for column “a”.



The order of visiting the nodes for each row in the attention matrix

Geometric attention

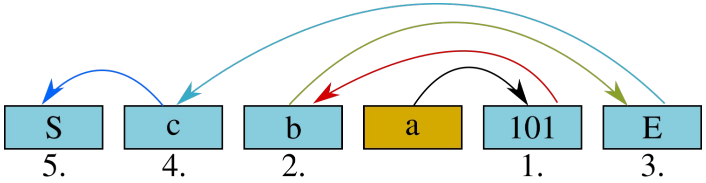
- Bias towards attending to the *nearest* match
- Each position has a probability of attending to it, produced by sigmoid

$$\boldsymbol{P}_{i,j} = \sigma(\boldsymbol{k}^{(j)\top}\boldsymbol{q}^{(i)})$$

- The final attention score is then the probability of attending to that node, multiplied by the probability of *not* attending to any closer ones

$$\boldsymbol{A}_{i,j} = \boldsymbol{P}_{i,j} \prod_{k \in \mathbb{S}_{i,j}} (1 - \boldsymbol{P}_{i,k})$$

- Efficient implementation possible in log-space with cumulative sum and scalar subtraction

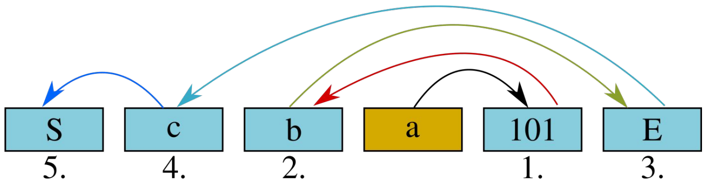


The Neural Data Router

- We call the resulting architecture **N**eural **D**ata **R**outer (NDR). Key ingredients:
    - Copy gate
    - Geometric attention
    - Shared layers
    - Sufficient depth

$$\boldsymbol{g}^{(i,t+1)} = \sigma(\mathrm{FFN}^{\mathrm{gate}}(\boldsymbol{a}^{(i,t+1)}))$$

$$\boldsymbol{h}^{(i,t+1)} = \boldsymbol{g}^{(i,t+1)} \odot \hat{\boldsymbol{h}}^{(i,t+1)} + (1 - \boldsymbol{g}^{(i,t+1)}) \odot \boldsymbol{h}^{(i,t)}$$



The **C**ompositional **T**able **L**ookup (CTL) dataset

- 8 input symbols, defined by 3 bit binary strings
- Randomly generated single-argument bijective functions, denoted by letters of alphabet
- The task is to learn to **generalize to longer compositions**
- Can be used to demonstrate failure modes of current networks
  - Generalizing to longer sequences

    101 a b c ✓ 101 a b c b a a c b ✗

  - Being order sensitive (generalization depends on the order)

    101 a b c ✓ c b a 101 ✗

Results: CTL

- None of the baselines is able to solve this simple task
- However NDR achieves 100% generalization accuracy

Model	IID		Longer		
	Forward	Backward	Forward	Backward	
LSTM	1.00 ± 0.00	0.59 ± 0.03	1.00 ± 0.00	0.22 ± 0.03	direction sensitive
DNC	1.00 ± 0.00	0.57 ± 0.06	1.00 ± 0.00	0.18 ± 0.02	
Transformer	1.00 ± 0.00	0.82 ± 0.39	0.13 ± 0.01	0.12 ± 0.01	not generalizing
+ rel	1.00 ± 0.00	1.00 ± 0.00	0.23 ± 0.05	0.13 ± 0.01	
+ rel + gate	1.00 ± 0.00	1.00 ± 0.00	0.99 ± 0.01	0.19 ± 0.04	direction sensitive
+ abs/rel + gate	1.00 ± 0.00	1.00 ± 0.00	0.98 ± 0.02	0.98 ± 0.03	dataset specific
+ geom. att.	0.96 ± 0.04	0.93 ± 0.06	0.16 ± 0.02	0.15 ± 0.02	not generalizing
+ geom. att. + gate (NDR)	1.00 ± 0.00	1.00 ± 0.00	$\boxed{1.00 \pm 0.00}$	$\boxed{1.00 \pm 0.00}$	

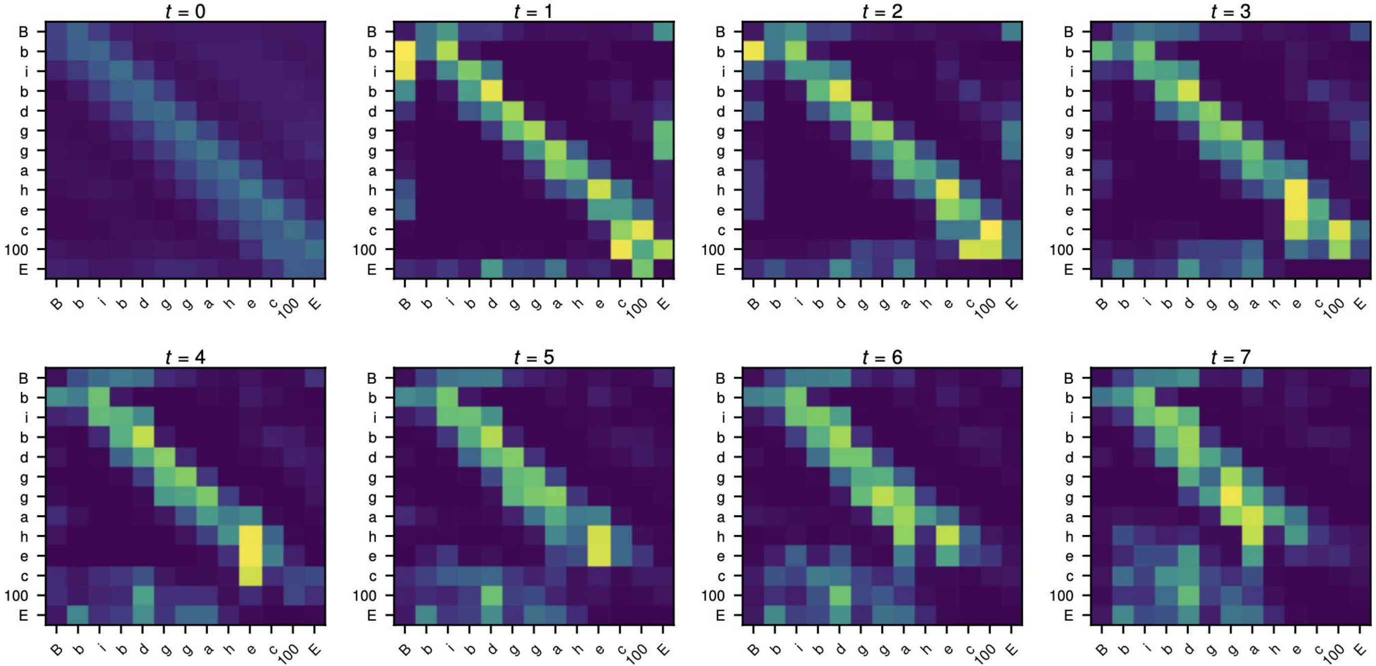
Results: Simple Arithmetics and ListOps

- Simple arithmetics (modulo 10): ((4*7)+2) = 0
- ListOps: [MED 4 8 5 [MAX 8 4 9]] = 6
- All models are good in IID setting, but only NDR generalizes

	Simple Arithmetics		ListOPS	
	IID (1..5)	Test (7..8)	IID (1..5)	Test (7..8)
LSTM	**0.99** ± **0.00**	0.74 ± 0.02	**0.99** ± **0.00**	0.71 ± 0.03
Bidirectional LSTM	**0.98** ± **0.01**	0.82 ± 0.06	**1.00** ± **0.00**	0.57 ± 0.04
Transformer	**0.98** ± **0.01**	0.47 ± 0.01	0.98 ± 0.00	0.74 ± 0.03
+ rel	**1.00** ± **0.00**	0.77 ± 0.04	**0.98** ± **0.01**	0.79 ± 0.04
+ abs/rel + gate	**1.00** ± **0.01**	0.80 ± 0.16	**1.00** ± **0.01**	0.90 ± 0.06
+ geom. att. + gate (NDR)	**1.00** ± **0.00**	**0.98** ± **0.01**	**1.00** ± **0.00**	**0.99** ± **0.01**

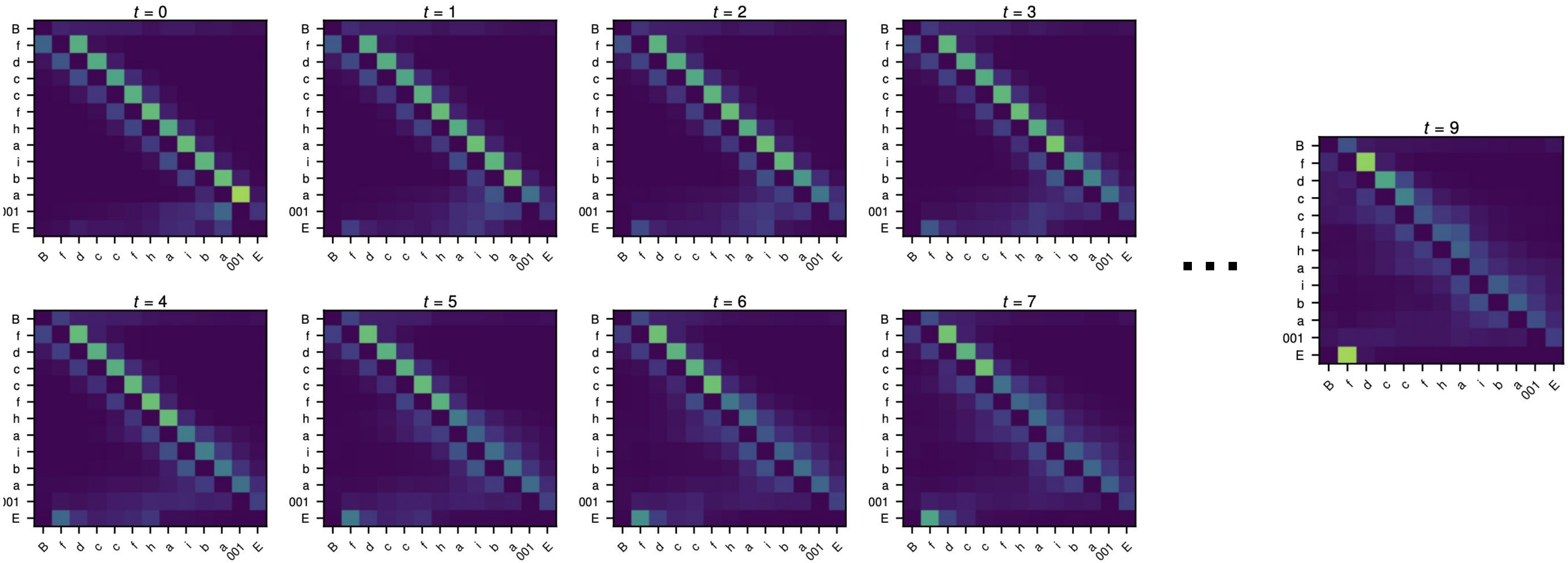
Analysis - CTL

- The attention patterns of the Transformer with relative positional embedding quickly becomes blurry



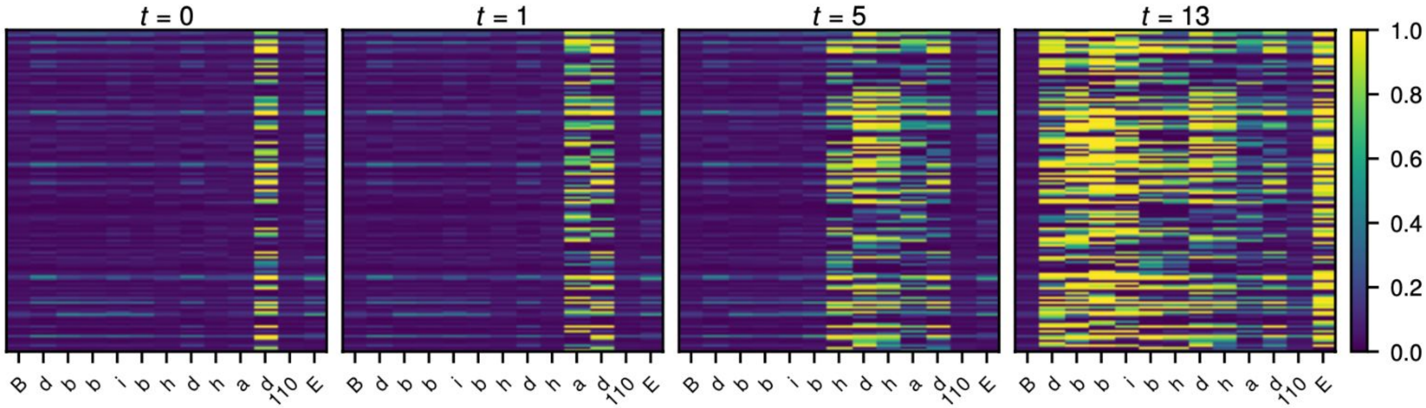
Analysis - CTL

- In contrast, the attention maps of NDR remain sharp



Analysis - CTL

- The gates open sequentially, after their input is available
  - Working as intended



Concluding remarks

- Neural models are flexible, but have poor length generalization properties
- In order to generalize, Transformers should have many, shared layers
- Copy gate enables to skip operations
- Geometric attention adds a bias to locality and serialization
- Gating with geometric attention enables length generalization on the CTL task, and parse tree depth generalization on ListOps and Simple Arithmetics

Thank you for your attention!

/robertcsordas/ndr

https://arxiv.org/abs/2110.07732