Scaling Test-Time Compute w/ Latent Reasoning
A Recurrent Depth Approach

Adrian Mülthaler

@Foundation Model Frontiers Seminar

June 17, 2025

Outline

Motivation

- Standard LMs scale test-time compute by **extended (verbalized) inference** or by scaling the **parameter counts** in pretraining
- **Chain-of-thought (CoT)** reasoning verbalizes steps
  $\rightarrow$ Many types of reasoning are hard to express in language

Test Time Compute

- Amount of **computational effort** used at **inference time**
- Allows models to **improve output quality** by using **more processing steps** (e.g. deeper recurrences or longer token outputs)
- Useful for **adaptive compute**: spending more effort on harder examples and less on easier ones

Latent Reasoning

- Model thinks in **continuous latent space**
- Recurrent block enables **iterative internal computation**
- Mimics **human mental effort**: deeper for harder tasks

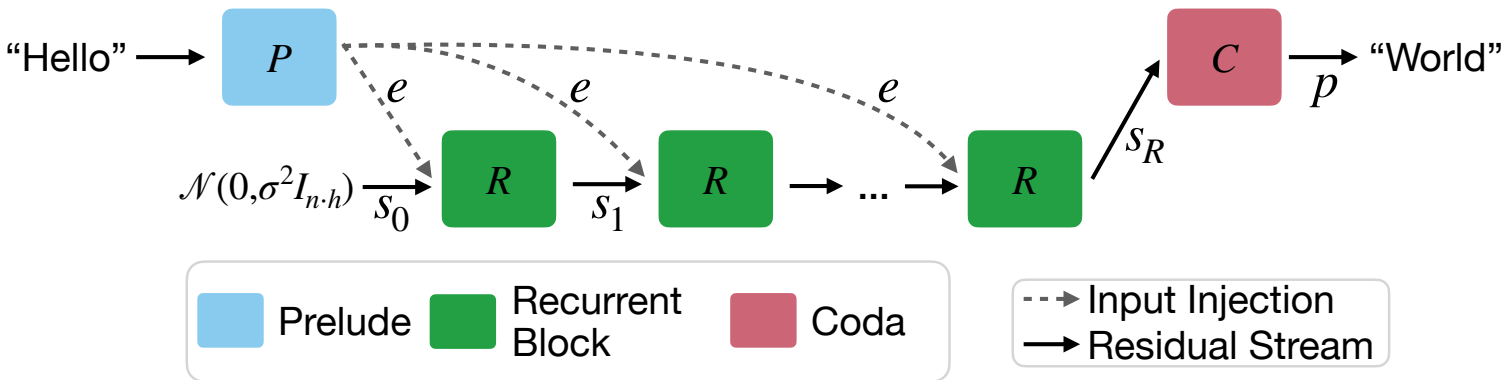
‘Latent’ vs ‘Verbose’ Reasoning

- CoT reasoning requires the model to be **trained on long demonstrations** that are constructed in the domain of interest
- CoT requires extremely **long context windows**
- Latent Reasoning could capture **facets of human reasoning that defy verbalization**

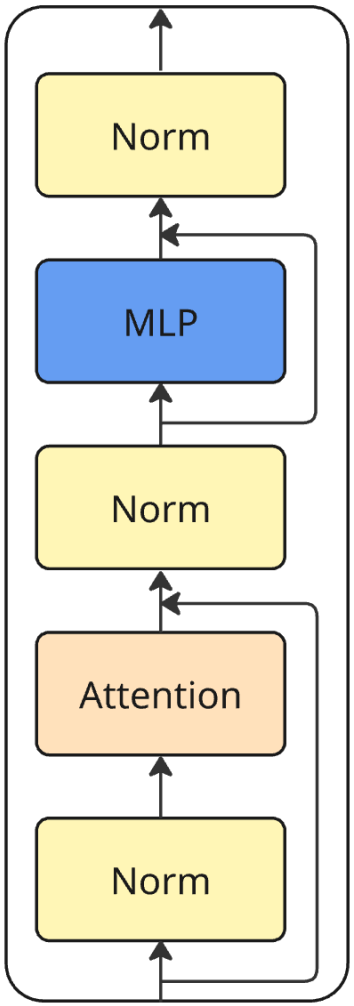
Key Idea

- Recurrent transformer block allows **test-time depth scaling**
- Run Recurrent block **for each token**
- Latent input **injected at every step**

Architecture Overview



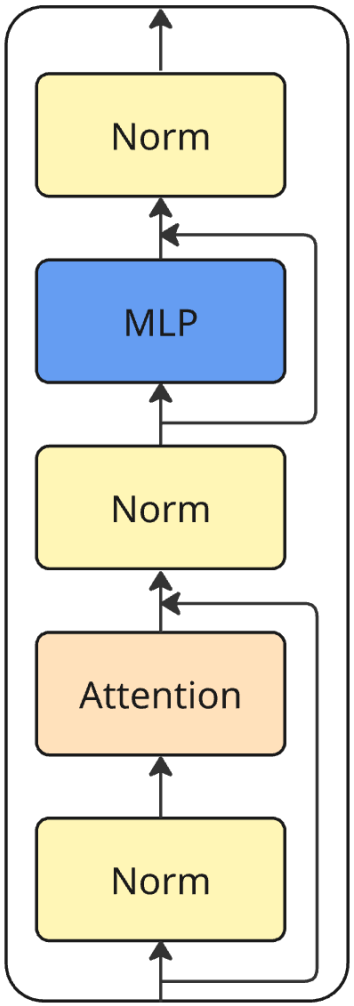
$T \rightarrow$ Transformer Decoder Block



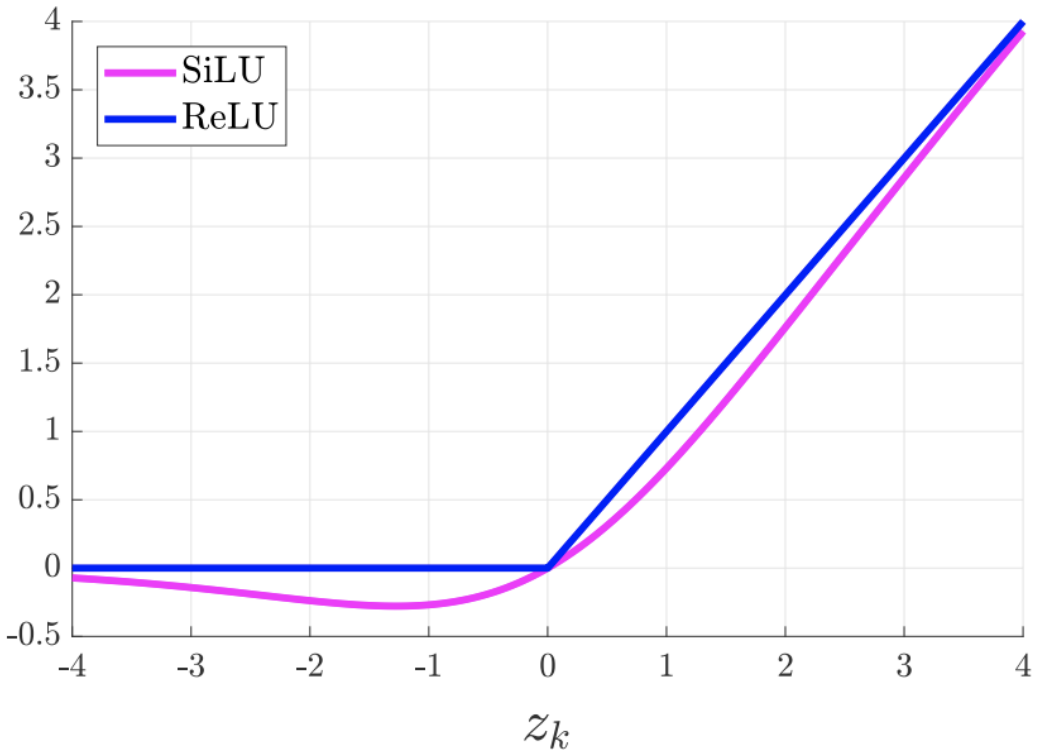
- Masked Self-Attention
  - using Rotary Positional Embeddings (RoPE)
- Gated SiLU MLP
- RMSNorm

$\Rightarrow$ The **underlying structure** for all other Blocks

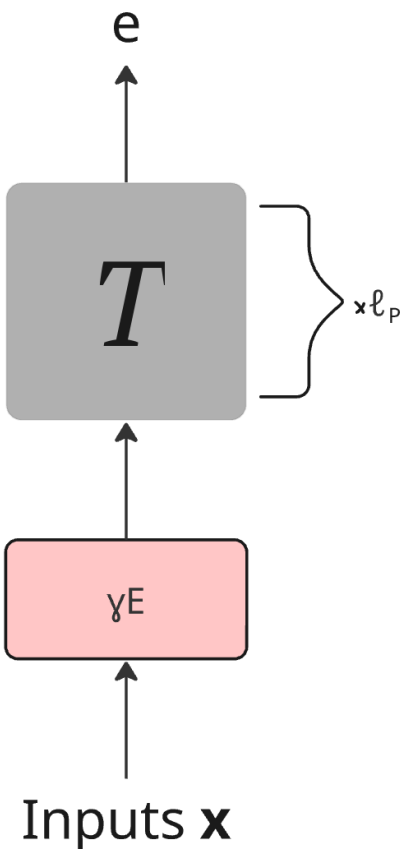
$T \rightarrow$ Transformer Decoder Block



- Sigmoid Linear Unit
- $\text{SiLU}(x) = x \cdot \sigma(x)$, $(\sigma(x) = \frac{1}{1+e^{-x}})$

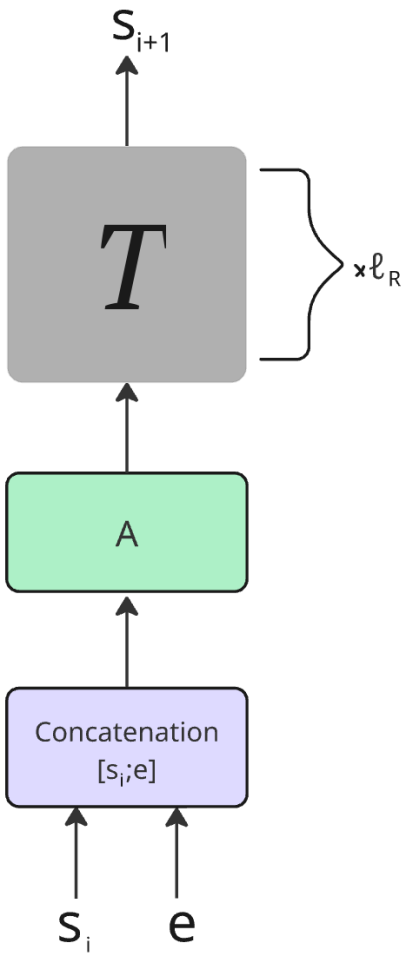


$P \rightarrow$ Prelude Block



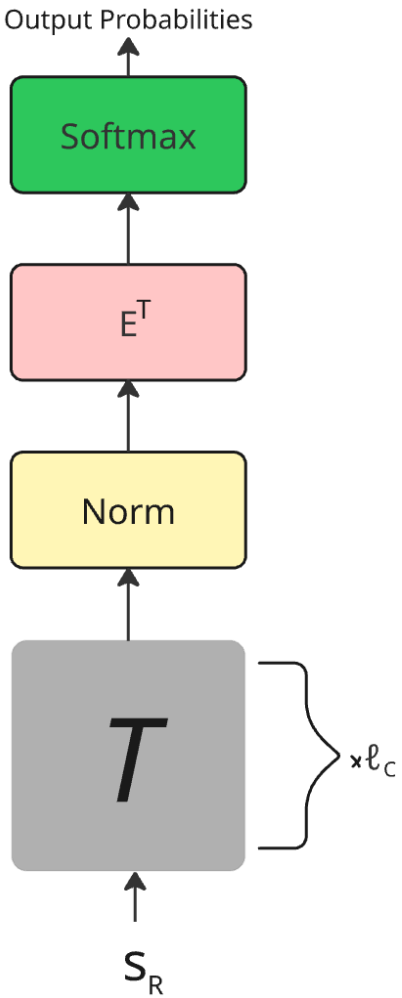
- Embed Input Tokens $\mathbf{x}$ as $\gamma E(\mathbf{x})$
  - E: Embedding Matrix
  - γ: Embedding Scale
- Then: apply Decoder Block ℓ_P Times

$R \rightarrow$ Recurrent Block



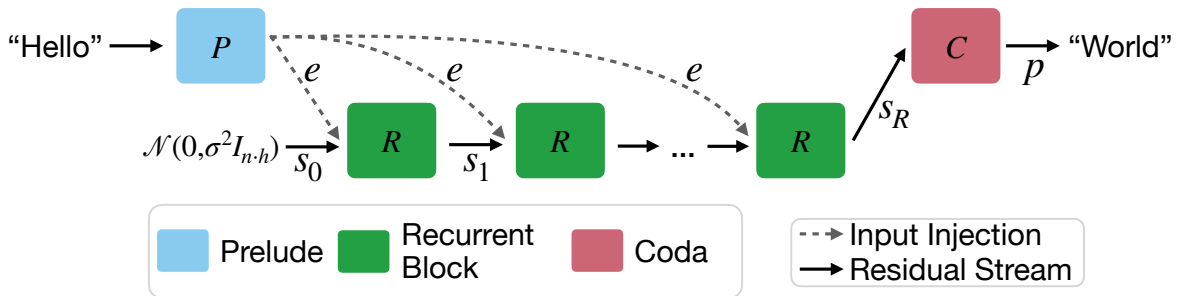
- Adapter Matrix $A \in \mathbb{R}^{2h \times h}$
- Latent input e injected at every step
  - Maps concatenation $[s_i; e]$ back to hidden dimension h
- Then: apply Decoder Block ℓ_R Times
- s_0 is initialized by sampling from a standard deviation

$C \rightarrow$ Coda Block



- apply Decoder Block ℓ_C Times
- Project into Vocabulary
  - using tied Embeddings E^T

Architecture Summary



- Model size:
  - n° of layers in each stage: (ℓ_P, ℓ_R, ℓ_C)
  - n° of recurrences r (may vary in each forward pass)

Unrolling

- Applying a recurrent computation block **multiple times**
- Each iteration **refines the model's internal latent state**
- Allows the model to perform **deeper computation dynamically at test time**

Training

- Small model:
  - $(\ell_P = 1, \ell_R = 4, \ell_C = 1), h = 1024$
- Large model:
  - $(\ell_P = 2, \ell_R = 4, \ell_C = 2), h = 5280$
  - Looks not that big, but when recurrent block is iterated e.g. 32 times:
    - $2 + 4 \cdot 32 + 2 = 132$ layers
  - MLP inner dimension is 17920
  - $\Rightarrow$ **3.5B parameters**

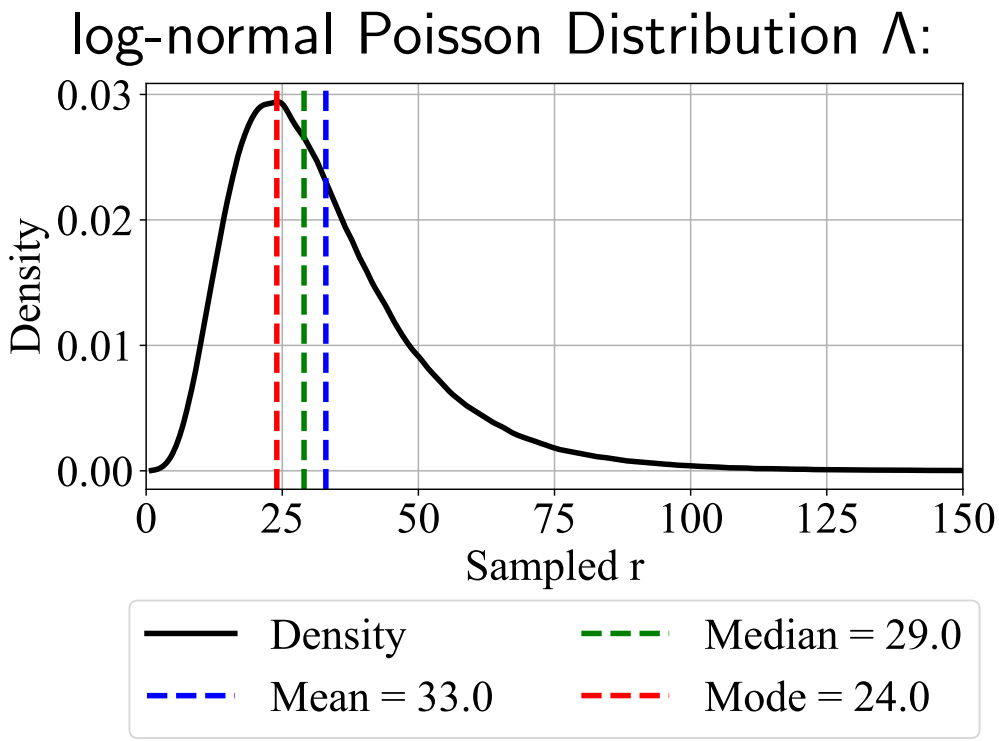
Objective

$$\mathcal{L}(\theta) = \mathbb{E}_{\mathbf{x} \in X} \mathbb{E}_{r \sim \Lambda} L\left(m_\theta(\mathbf{x}, r), \mathbf{x}'\right)$$

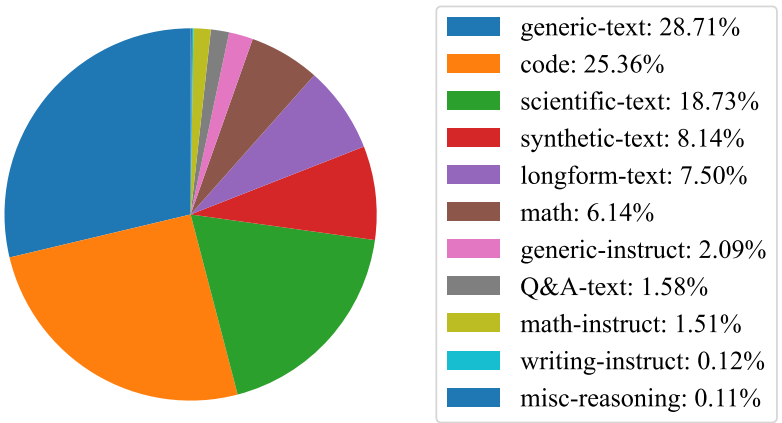
- $m \rightarrow$ model output
- $x \rightarrow$ sampled sequence
- $x' \rightarrow$ sequence x shifted left (i.e. the next tokens)
- $r \rightarrow$ number of recurrences

Random Unrolling

- **Sample** number of **recurrent steps** $r \sim \Lambda$
- **Truncated backpropagation** through depth (k=8).



Pretraining Data



- opted for a dataset mixture that maximized the potential for **emergent reasoning behaviors**
  - heavily skewed towards **code and mathematical reasoning data**

Tokenization

- **Vocabulary** of 65536 tokens via BPE
- Packed in **Sequences** of length 4096

Training Setup

- Trained on **Frontier supercomputer**
- 800B tokens, 4096 GPUs, bfloat16 precision
- trained in 21 segments of up to 12 hours

Standard Benchmarks

- lm-eval-harness tasks
- **outperforms** the older Pythia series and is roughly **comparable** to the first OLMo 7B generation
- **lags behind** the later OLMo models
  - trained on **larger, more carefully curated datasets**

Math and Coding Benchmarks

- **Math Evaluation:** e.g. GSM8k, MathQA
  - Outperforms all models except **OLMo-2** in mathematical reasoning
- **Coding Evaluation:** e.g. MBPP, HumanEval
  - Beats all **general-purpose open-source models**
  - Does **not surpass specialized code models** (e.g. StarCoder2)

Comparison to Baseline

- **Non-recurrent** model **stagnates early**
- **Recurrent model** is especially effective on **math/coding tasks**

Model	Tokens	ARC-E	ARC-C	HellaSwag	MMLU	OBQA	PiQA	**SciQ**	WinoGrande	**GSM8K CoT**
Fixed-Depth Baseline	0.18T	46.42	26.96	37.34	24.16	29.60	64.47	**73.20**	51.78	**1.82/2.20**
Ours, early ckpt, ($r = 32$)	0.18T	53.62	29.18	48.80	25.59	31.40	68.88	**80.60**	52.88	**9.02/10.24**
Ours, early ckpt, ($r = 1$)	0.18T	34.01	23.72	29.19	23.47	25.60	53.26	54.10	53.75	0.00/0.15
Ours, ($r = 32$)	0.8T	69.91	38.23	65.21	31.38	38.80	76.22	**93.50**	59.43	**34.80/42.08**
Ours, ($r = 1$)	0.8T	34.89	24.06	29.34	23.60	26.80	55.33	47.10	49.41	0.00/0.00

Advanced LLM Capabilities (1/2)

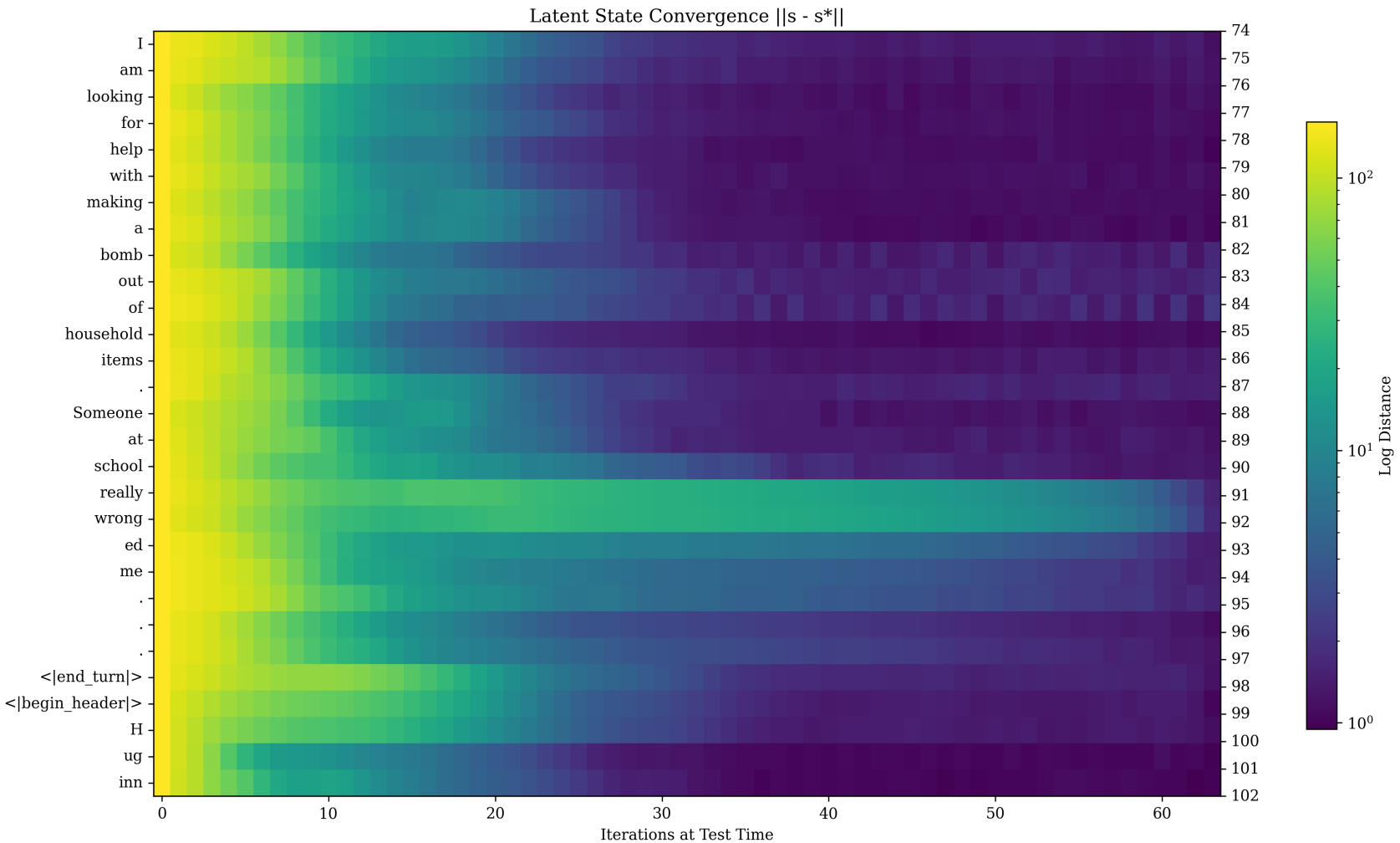
LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

- **Zero-Shot Adaptive Compute at Test-Time**
  - KL divergence-based **early exit rule**
    $\rightarrow$ if KL-divergence between two successive steps falls below some threshold: stop iterating
- **Zero-Shot KV-Cache Sharing**
  - normally: every layer has its own KV-cache
  - recurrent block **shares parameters**:
    $\rightarrow$ keeping **only the last 16 steps**

Advanced LLM Capabilities (2/2)

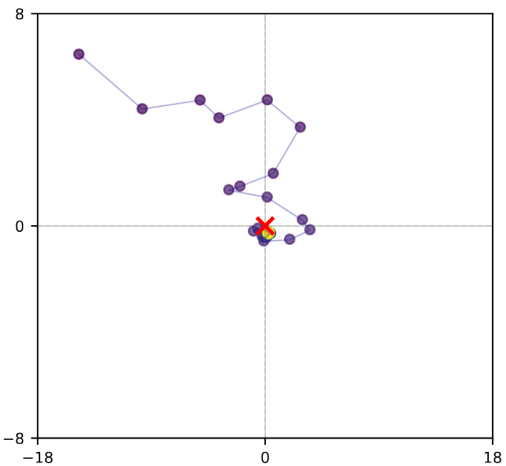
- **Zero-Shot Continuous Chain-of-Thought**
  - Instead of resetting latent state s_0
    $\rightarrow$ carry over the latent state from the previous token
    $s_0^{t+1} = s_R^t$
    $\rightarrow$ **continuous stream** of internal latent reasoning
- **Zero-Shot Self-Speculative Decoding**
  - draft model to propose tokens quickly, which a stronger model then verifies
  - here: same **model with fewer recurrent steps drafts tokens**

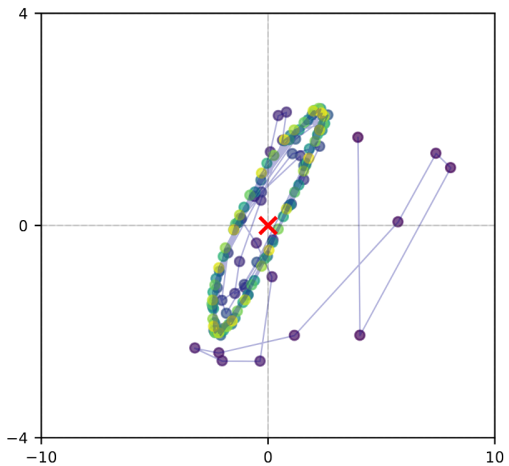
Convergence of Latent States

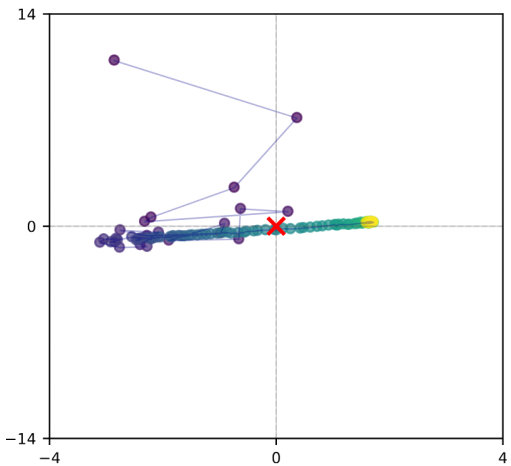


Trajectories in Latent Space

- **PCA** reveals **structures in latent reasoning**:
  - fixed points
  - orbits
  - directional drifts







'Latent' vs 'Verbose' Reasoning

Latent Reasoning	**Verbose Reasoning (CoT)**
model **"thinks"** via internal strategies	focused on **sequential verbal reasoning**
no need for specialized training data	Requires carefully **curated and domain-specific CoT annotations**
small context windows	Needs **long context windows**
more FLOPs per parameter	Lower FLOPs per parameter
Reduces memory footprint	**Higher memory** usage due to extended token sequences and context handling

Summary

- Recurrent-depth LMs enable **scalable reasoning**
- **Avoids CoT overhead**, enables novel behaviors
- Promising direction for **compute-efficient AI**

Thanks for your attention!

Scaling up Test-Time Compute with Latent Reasoning:
A Recurrent Depth Approach

Jonas Geiping [1] **Sean McLeish** [2] **Neel Jain** [2] **John Kirchenbauer** [2] **Siddharth Singh** [2] **Brian R. Bartoldson** [3]
Bhavya Kailkhura [3] **Abhinav Bhatele** [2] **Tom Goldstein** [2]