Deep Thinking by Markov Chain of Continuous Thoughts

Jiayu Liu [1 †] **Zhenya Huang** [1] **Xuan Yang** [2 3] **Tianyun Ji** [1] **Anya Sims** [4] **Hao Xu** [5] **Enhong Chen** [1]
Yee Whye Teh [4] **Ning Miao** [2 3]

Abstract

Transformer-based models can perform complicated reasoning by generating reasoning paths token by token. While effective, this approach often requires generating thousands of tokens to solve a single problem, which can be slow and computationally expensive. More importantly, it involves a discrete sampling operation at the end of each time step, creating an information bottleneck across time steps. In this work, we propose MarCos, an improvement of the transformer structure that allows fully continuous reasoning at the thought level. Unlike traditional transformer layers, which focus on refining token predictions at each time step, layers in MarCos map a continuous representation of a stepwise thought to the distribution of the next thought. This enables us to achieve multi-step reasoning in a single pass of MarCos. Preliminary experimental results on synthetic and real-world math tasks show the great potential of MarCos. Notably, we observe that the increased information bandwidth of MarCos elicits the ability of parallel thinking, in contrast to single-threaded thinking in traditional transformers. Meanwhile, in real-world math tasks, MarCos achieves more than $10\times$ speedup in wall-clock time with the same level of accuracy. Our code is available at https://github.com/Ljyustc/MarCos .

1. Introduction

Transformer-based models exhibit strong reasoning capabilities by autoregressively generating reasoning paths token by token (Wei et al., 2022). Despite its simplicity and applicability, this token-wise autoregressive generation can lead to inefficient inference and suboptimal reasoning behavior. To solve a problem, we often need to serially invoke the transformer thousands of times, resulting in significant latency. Meanwhile, transformers are forced to think and speak simultaneously. In other words, they need to generate one token immediately after a single forward computation, leaving little room for thoughtful planning (Valmeekam et al., 2023; Kambhampati et al., 2024). Moreover, the discrete operation of token selection at the end of each time step heavily restricts the information bandwidth between time steps. For example, the information generated at the last transformer layer can only be passed to the next step through the sampled tokens, leading to severe information loss.

To mitigate these issues, previous works (Deng et al., 2023; 2024) have attempted to replace discrete tokens in transformers with continuous ones. For example, Coconut (Hao et al., 2024) gradually replaces blocks of discrete tokens with fewer continuous tokens. CoLaR (Tan et al., 2025) further introduces an additional loss to align the continuous tokens with the representation of discrete tokens in ground-truth reasoning paths. Although these methods reduce the number of time steps to reach a final answer, they mainly work by compressing multiple discrete tokens into a continuous one, which means that their latent thinking essentially remains a form of token-based decision-making. Moreover, most of them adopt a deterministic transition function between continuous tokens during reasoning, which is unsuitable for learning the diverse and multi-modal solution space.

In this paper, we propose **MarCos**, a variant of transformer that natively supports continuous reasoning at the thought level. In MarCos, we model the step-by-step reasoning process as a Markov chain of continuous thoughts. Different from traditional transformers, whose layers are in charge of refining the prediction of the next token (Lioubashevski et al., 2024), MarCos employs dedicated thinking layers to directly predict the distribution of thoughts at the next reasoning step based on the current thought, as illustrated in Figure 1. By representing each thought as the states of a group of neurons, MarCos essentially thinks internally by updating neuron states. To peek into and add supervision on intermediate thoughts, we use a separate speaking layer to translate neuron states to natural language. This design es-

[†]Work done during Jiayu's visit to the University of Oxford. [1]State Key Laboratory of Cognitive Intelligence, University of Science and Technology of China [2]Department of Data Science, City University of Hong Kong [3]Hong Kong Institute of AI for Science, City University of Hong Kong [4]Department of Statistics, University of Oxford [5]Li Auto Inc. Correspondence to: Ning Miao <ningmiao@cityu.edu.hk>, Zhenya Huang <huangzhy@ustc.edu.cn>.

Preprint. May 5, 2026.

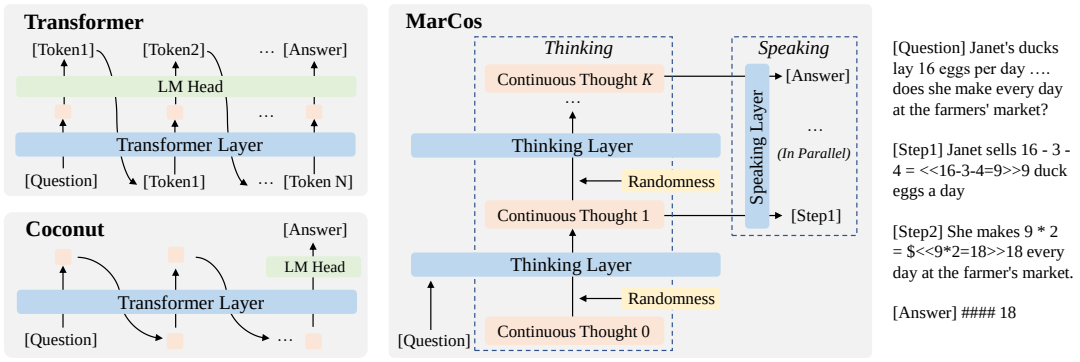


Figure 1. Comparison of a traditional transformer, the representative continuous reasoning method Coconut, and our MarCos. In MarCos, the *thinking* process is modeled as iterative transitions of continuous thoughts (0 to K) with incorporated randomness. The *speaking* process (optionally) translates these thoughts into natural language (e.g., [Step1] to [Answer]).

sentially disentangles the internal *thinking* process from the *speaking* process, which allows the model to fully deliberate before speaking. It also aligns with recent neuroscience findings that language is primarily a tool for communication rather than thought (Fedorenko et al., 2024; Fedorenko & Varley, 2016). To model the diverse distribution of possible reasoning paths, we further incorporate a variational module that explicitly controls the randomness of reasoning at the step level. With all these designs, we enable MarCos to think completely in a continuous space, which creates the possibility for parallel and more efficient reasoning.

We first evaluate MarCos on synthetic tasks, where we uncover that its superior bandwidth enables parallel thinking, a capability that is difficult to achieve in traditional transformers. Then, on three grade-school level mathematical benchmarks (GSM8K (Cobbe et al., 2021), SVAMP (Patel et al., 2021), and MultiArith (Roy & Roth, 2015)), MarCos **obtains performance on par with or even superior to transformer-based CoT reasoning**, with a 4.7% accuracy improvement on GSM8K and up to $15.7\times$ inference-time speedup. This highlights the great potential of our architecture. Beyond accuracy, our analysis reveals the functional independence between contextual processing and randomness determination in variational modules, and uncovers that different randomness dimensions control different speaking properties such as sentence length and step depth. This opens up the possibility of manually controlling reasoning behaviors. Furthermore, we show that our model remains competitive when applying non-autoregressive (NAR) decoding in the speaking stage, indicating that thinking and speaking play well-separated roles in our architecture. In summary, our contributions in this paper are:

- We propose MarCos, a new architecture that models reasoning as a Markov chain of continuous thoughts and operates in decoupled *thinking-speaking* stages.
- We introduce a principled way to model the randomness in reasoning, which yields two key advances: (1) enabling explicit control over the reasoning process, and (2) verifying the feasibility of pre-determining step-level randomness before token generation.
- Experiments on synthetic and math tasks show the strong potential of MarCos, which can achieve up to $15.7\times$ faster inference without loss in accuracy.

2. Related Work

Various continuous reasoning methods have emerged to mitigate the issues of token-by-token reasoning, which can be categorized into time-axis and depth-axis approaches.

Time-axis latent reasoning. Most time-axis approaches aim to improve inference efficiency by compressing the discrete tokens in explicit CoT into a few continuous steps. For example, iCoT-KD (Deng et al., 2023; 2024) distills entire reasoning path into a single inference step, while curriculum learning is adopted in iCoT-SI (Deng et al., 2024) to gradually encourage the model to shift from explicit rationales toward fully latent reasoning. However, compressing an entire CoT reasoning path, which may consist of thousands of tokens, into a single inference step, is too aggressive and degrades model performance. Consequently, more recent works adopt intermediate continuous vectors, each representing a block of consecutive discrete tokens in CoT reasoning. For example, Coconut (Hao et al., 2024) progressively replaces blocks of discrete word tokens with continuous vectors. CoLaR (Tan et al., 2025), CODI (Shen et al., 2025), SIM-CoT (Wei et al., 2025), and CoT2 (Zhang et al., 2025) further add supervision to align the latent states with token blocks they replace in explicit CoT, with CoT2 forming continuous tokens as convex combinations of vocabulary embeddings. While improving inference efficiency, these methods essentially remain a form of token compression and could still be restricted by token-level reasoning. Moreover, their continuous vectors are updated either in a fixed manner or via simple Gaussian transitions, which limits their ability to model the diverse and exploratory thinking.

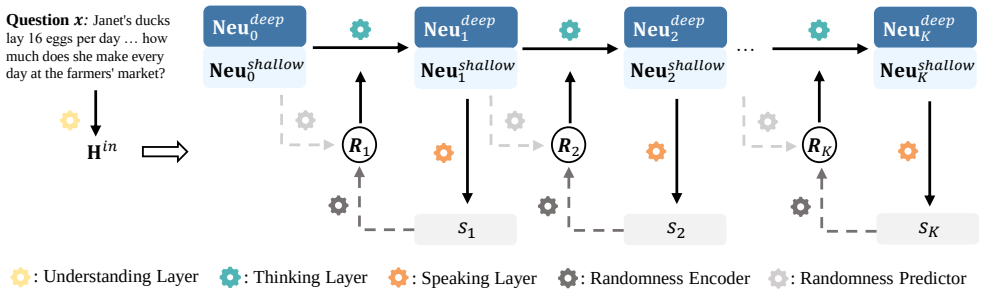


Figure 2. Illustration of our MarCos model. In the thinking stage, MarCos reasons in the continuous space for K steps by updating neurons $\mathbf{Neu}_k^{deep}$ and $\mathbf{Neu}_k^{shallow}$. In the speaking stage, $\{\mathbf{Neu}_k^{shallow}, k = 1, ..., K\}$ are translated to natural language in parallel. If intermediate steps are not required, we only need to translate $\mathbf{Neu}_K^{shallow}$ to a final answer. During training, the random variable $\boldsymbol{R}_k$ is derived from the ground-truth sentence $\boldsymbol{s}_k$ using the randomness encoder f, while in inference, it is sampled from a distribution predicted by the randomness predictor g based on $\mathbf{Neu}_k^{deep}$, $\mathbf{Neu}_k^{shallow}$, and H^{in}.

Depth-axis latent reasoning. These approaches modify the transformer architecture to support latent reasoning. One representative is the Looped Transformer (Giannou et al., 2023; Yang et al., 2024), which reuses the same layer parameters across iterations to deepen internal deliberation. For example, Geiping et al. (2025) apply transformer blocks recurrently for each token, enabling test-time scaling up without generating additional tokens. Zhu et al. (2025) further incorporate adaptive early-exit mechanisms, allowing the model to dynamically determine the number of recurrent steps. Compared with MarCos, these methods lack supervision for intermediate reasoning states. Consequently, they can be seen as parameter-efficient ways to increase the depths of transformers, which are orthogonal to our focus.

3. Our MarCos Model

In this section, we introduce MarCos, a new architecture for fully continuous, stepwise reasoning at the thought level. Specifically, we model the reasoning process as a conditional hidden Markov model (cHMM), where the input questions are the conditions, the internal thoughts are hidden variables, and spoken sentences are observable variables. This formulation enforces a clear disentanglement between thinking and speaking: thinking corresponds to transitions of continuous thoughts, and speaking is the emission process from these thoughts to natural language. Crucially, continuous thought at each time step represents a *reasoning step*, which enables the model to plan thoroughly before producing a sentence. Compared with traditional transformers, our paradigm offers two key advantages: (1) by disentangling thinking from speaking, it enables more structured and deliberate reasoning beyond token-by-token generation; and (2) by carrying out the entire thinking process in continuous space, we significantly increase the information bandwidth between reasoning steps, which enables parallel thinking within a single forward pass of MarCos.

For the transition between thoughts, the simplest idea is to have a deterministic mapping from the current thought to the next one. However, this could be suboptimal, since there are usually multiple plausible reasoning paths for a math problem, which makes deterministic mappings insufficient to model the diverse distribution of possible thoughts, limiting model exploration during inference.

To tackle this problem, MarCos maps each thought to a multi-modal distribution of possible thoughts at the next step. By doing so, we can explicitly parameterize the randomness in the thinking process without relying on token-level sampling.

We will first introduce the structure of MarCos in Section 3.1. Then, we will show the training recipe of MarCos in Section 3.2. Specifically, we propose a two-phase training scheme, inspired by VAE (Kingma & Welling, 2013), to address the intractability of sample probabilities caused by the stochastic mapping between latent thoughts.

3.1. Model Architecture

This section explains how MarCos "thinks" and generates the solution to a question. As shown in Figure 2, it consists of three separate layers: understanding, thinking, and speaking, which provide conditions, transition probabilities, and emission probabilities for cHMM.

Understanding. Given the input question x of length n, we use an understander layer to convert it into a sequence of feature vectors $\mathrm{H}^{in} = \{\boldsymbol{h}_1, ..., \boldsymbol{h}_n\}$:

$$\mathrm{H}^{in} = \text{Understanding}(x) \in \mathbb{R}^{n \times d}, \tag{1}$$

where d denotes the latent dimension.

Thinking. As the hidden variable of the cHMM, our continuous thought is realized as the states of a group of neurons. Specifically, inspired by neuroscience evidence of functional specialization in the human brain (Kanwisher, 2010), we have two types of neurons: $\mathbf{Neu}^{deep} \in \mathbb{R}^{T \times d}$ and $\mathbf{Neu}^{shallow} \in \mathbb{R}^{S \times d}$, in charge of slow and fast think-

ing, respectively. The intuition is that $\mathbf{Neu}^{deep}$ facilitates deeper reasoning, while $\mathbf{Neu}^{shallow}$ represents shallower and more readily verbalizable reasoning, as it interacts more closely with the speaking layer (see Eq. 4). Here, T and S denote the numbers of neurons in each group.

Then, each thinking step corresponds to an update of $\mathbf{Neu}^{deep}$ and $\mathbf{Neu}^{shallow}$ within a thinking layer, conditioned on the input feature H^{in} and their own previous states.

As described earlier, a key difference between MarCos and existing continuous reasoning approaches is that it explicitly parameterizes the randomness of the thinking process at the *step level*. Concretely, we introduce an auxiliary random variable $\boldsymbol{R}_k \in \mathbb{R}^{\tau \times d}$ to control the reasoning behavior at step k. τ here is the length scaling factor of $\boldsymbol{R}_k$. Instead of directly mapping the current thought at time step $k-1$ to the multi-modal and high-dimensional distribution of thoughts at the next step k, we first map it to a distribution of $\boldsymbol{R}_k$, which lies in a better-behaved, lower-dimensional space. Then, we can map the current thought and the controlling variable $\boldsymbol{R}_k$ deterministically to the next thought. To encourage each dimension of $\boldsymbol{R}_k$ to represent a single random factor and stabilize the training process, we further add a sparsity constraint to $\boldsymbol{R}_k$, which will be elaborated in Section 3.2. As observed in Sections 4.1.2 and 4.2.3, the introduction of $\boldsymbol{R}_k$ facilitates more controllable randomness in the reasoning process. For example, we find that different dimensions of $\boldsymbol{R}_k$ effectively controls different aspects of thinking, including high-level factors (e.g., search directions, reasoning depth) and low-level choices (e.g., expression format, sentence length).

Specifically, we employ a learnable randomness predictor g to predict the distribution of $\boldsymbol{R}_k$ based on input information H^{in} and the current thought $\mathbf{Neu}_{k-1}^{deep}$ and $\mathbf{Neu}_{k-1}^{shallow}$:

$$\boldsymbol{\mu}_k, \boldsymbol{\sigma}_k = g(\mathbf{Neu}_{k-1}^{deep}, \mathbf{Neu}_{k-1}^{shallow}, \mathrm{H}^{in}). \tag{2}$$

Then, a sample of $\boldsymbol{R}_k \sim \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\sigma}_k)$ is drawn to guide the transition from the current thought to the next one[1]:

$$\mathbf{Neu}_k^{deep}, \mathbf{Neu}_k^{shallow} = \mathrm{Thinking}(\mathbf{Neu}_{k-1}^{deep}, \mathbf{Neu}_{k-1}^{shallow}, \mathrm{H}^{in}, \boldsymbol{R}_k). \tag{3}$$

The thinking layer uses bi-directional attention to facilitate this information flow and iteratively updates the neurons to stimulate step-by-step reasoning. The initial values of neurons, $\mathbf{Neu}_0^{deep}$ and $\mathbf{Neu}_0^{shallow}$, are learnable parameters.

Speaking. After each thinking step, some thoughts in $\mathbf{Neu}_k^{shallow}$, such as plans and intermediate conclusions, may become ready for verbalization. We use a speaking layer to translate them into the discrete space of sentences:

$$\boldsymbol{s}_k = \mathrm{Speaking}(\mathbf{Neu}_k^{shallow}). \tag{4}$$

Unlike traditional transformers, thinking in MarCos does not depend on speaking, so our speaking stage is optional for all intermediate steps and can be skipped except for the final step when only an answer is required. Moreover, speaking at each step is independent to each other, making them highly parallelizable compared with traditional models.

In addition, since randomness in MarCos is resolved at the step level in the thinking stage *before* the speaking layer generates specific sentences, complex randomness can be simulated "in one pass" rather than through multiple "iterative" sampling. This provides a new perspective for the development of non-autoregressive (NAR) language models with one-pass generation, and suggests a promising avenue for accelerating diffusion LLMs (Gulrajani & Hashimoto, 2023; Sahoo et al., 2024). To verify this idea, we performed a preliminary experiment by replacing the speaking layer of MarCos with a fully NAR decoder. As shown in Section 4.2.3, even the simplest NAR decoding strategy still obtains competitive performance, which provides further acceleration to MarCos. This also highlights that our model achieves a clear separation between thinking and speaking, ensuring that the reasoning quality remains stable while allowing flexibility in the choice of speaking strategy.

3.2. Training

In this part, we discuss how to effectively train MarCos. Assume we have a training sample that consists of a question x and its step-by-step solution $y = \{y_1, ..., y_N\}$. The training objective is to find the parameters that maximize the probability of generating all y_k. However, because sampling occurs in thinking rather than in speaking, we no longer have a closed-form conditional probability for y_k.

To solve this, we adapt the evidence lower bound (ELBO) used in VAE training to maximize the marginal probability of y_k at each step. Specifically, the general ELBO is:

$$\mathcal{L}_k^{\mathrm{ELBO}} = \mathbb{E}_{q(\boldsymbol{R}_k|y_k)}\big[\log p(y_k|\boldsymbol{R}_k)\big] - \mathrm{KL}\big(q(\boldsymbol{R}_k|y_k) \,\|\, p(\boldsymbol{R}_k)\big), \tag{5}$$

which is a combination of a reconstruction term and a KL term. In our case, the VAE encoder q is realized by a learnable function f (referred to as randomness encoder in Figure 2) to map y_k to the posterior mean and variance of $\boldsymbol{R}_k$:

$$\hat{\boldsymbol{\mu}}_k, \hat{\boldsymbol{\sigma}}_k = f(y_k). \tag{6}$$

The VAE decoding is simply a step of thinking and speaking in Eqs. 3 and 4, which maps $\boldsymbol{R}_k$ back to the sentence space. Then the reconstruction loss can be expanded as:

$$\mathcal{L}_k^{re} = \mathbb{E}_{\boldsymbol{R}_k \sim \mathcal{N}(f(y_k))}\big[\log p(\boldsymbol{s}_k = y_k | \mathbf{Neu}_{k-1}^{deep}, \mathbf{Neu}_{k-1}^{shallow}, \mathrm{H}^{in}, \boldsymbol{R}_k)\big]. \tag{7}$$

[1] Eq. (3) can be extended to include all preceding thinking neurons as input. In Appendix A, we show that this first-order setting does not create an information bottleneck compared to CoT.

Unlike vanilla VAEs with fixed isotropic Gaussian priors, our randomness predictor g in Eq. 2 provides a different prior for different questions and thoughts as described in Section 3.1. Consequently, our KL divergence between the prior (Eq. 2) and the posterior (Eq. 6) is:

$$\mathcal{L}_k^{KL} = \mathrm{KL}\big(\mathcal{N}(f(y_k)) \,\|\, \mathcal{N}(g(\mathbf{Neu}_{k-1}^{deep}, \mathbf{Neu}_{k-1}^{shallow}, \mathrm{H}^{in}))\big). \tag{8}$$

In addition, without a proper control, the model may encode excessive semantic information of y_k into $\boldsymbol{R}_k$ via the randomness encoder f. This information can be easily merged into $\mathbf{Neu}_k^{shallow}$ in Eq.3 for the reconstruction of y_k, creating a training "shortcut" that prevents the model from learning genuine reasoning. To prevent this, we introduce an extra sparsity loss $\mathcal{L}_k^{sparse} = \|\boldsymbol{R}_k\|_1$. This forces $\boldsymbol{R}_k$ to store only the minimal amount of information that cannot be inferred from context, while providing a stronger inductive bias for structured and interpretable reasoning. For example, at each thinking step, only a small set of dimensions need to be active. Dimensions useful for breadth-first reasoning may not be activated during depth-first reasoning. To reflect this inductive bias and encourage each dimension of $\boldsymbol{R}_k$ to capture a "mono-semantic" factor in thinking, we draw inspiration from sparse autoencoders and implement the function f as a d-dimensional transformer followed by an MLP that expands the dimension from d to $\tau \times d$. The overall training loss of MarCos is:

$$\mathcal{L} = \sum_{k\in\{1,2,\ldots,N\}} \big[-\mathcal{L}_k^{re} + \mathcal{L}_k^{KL} + \lambda \mathcal{L}_k^{sparse}\big], \tag{9}$$

where λ is a hyperparameter that controls the strength of the sparsity loss. To eliminate the need to pre-select λ, we adopt a dynamic weighting strategy that adaptively controls λ to balance the reconstruction and sparsity regularization during training (see Appendix B). Moreover, to avoid model collapse caused by simultaneously tuning the posterior and prior, we adopt a two-phase training scheme, where we first optimize $\mathcal{L}_k^{re}$ and $\mathcal{L}_k^{sparse}$ with fixed g and then train g to minimize $\mathcal{L}_k^{KL}$ with all other parameters fixed.

4. Experiments

4.1. Evaluation on Synthetic Benchmarks

To systematically compare the fundamental characteristics of MarCos and transformers in performing reasoning and randomness modeling, we first design three synthetic reasoning tasks, with examples provided in Figure 3.

4.1.1. Experimental Setup

Level-1 Task: Random Summation (RS). Starting from a random sequence of $2N$ 1-digit non-negative integers, at each step, another 1-digit non-negative integer is uniformly sampled and added to each number in the sequence. This process is repeated for K steps. In the last step, the sum of the first two numbers is computed as the final answer. This task requires the most basic arithmetic skills and modeling of very simple randomness in the addends.

Level-2 Task: Random Pairing (RP). Given a random sequence of $2N$ 1-digit non-negative integers, at each step, the model needs to randomly partition it into N pairs. For each pair (a, b), the first position is updated to $(a + b)$ and the second position to $(a - b)$. The other settings are the same as in the level-1 RS task. Compared with RS, this task not only involves moderately more complex calculations, but also induces more complicated randomness of partitioning. Because a global partitioning configuration must be determined before token generation, the randomness is more difficult to be modeled via token-level sampling.

Level-3 Task: Parallel 24 (P24). The classic Game of 24 requires a model to perform trial and error until reaching 24. In P24, we ask models to think about different paths in parallel and find a correct one in K steps. This task requires parallel thinking and a larger information bandwidth, as the model needs to consider as many reasoning paths as possible at each step. It simulates how humans dynamically adjust their reasoning strategy during problem-solving.

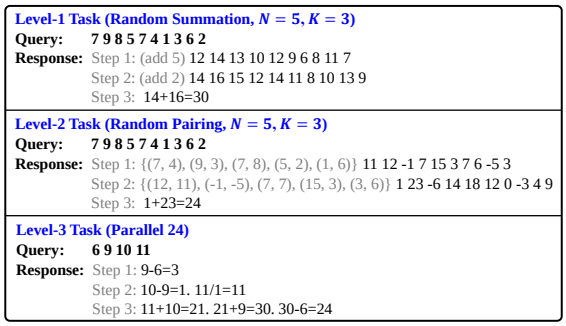


Figure 3. Overview of synthetic tasks. The output sentence is shown in black, while the gray segments are added only to facilitate step delineation and to understand the reasoning process.

Table 1. Accuracy on synthetic tasks ($N = 5, K = 3$).

Model	RS	RP	P24
Transformer	0.999	0.589	0.170
MarCos	1.000	0.712	0.668

4.1.2. Results

Table 1 shows the performance of MarCos and traditional transformer on all three tasks. On the RS task, where the randomness can be effectively absorbed into a sequential token-generation process, both MarCos and the transformer achieve perfect performance. However, the performance of transformer degrades sharply on the RP task, which requires the modeling of the more structured stochasticity of random partitioning. This result suggests that token-based sampling

Table 2. Results of models **trained from scratch**. All continuous reasoning baselines (0.5B) are trained on equation data, following their original papers. We report the best continuous models in bold and the runner-ups with underline.

Model	Train (h)	GSM8K		SVAMP		MultiArith	
		Acc (%)	Test (s)	Acc (%)	Test (s)	Acc (%)	Test (s)
Training Data: Text							
CoT-SFT	0.31±0.01	13.27±0.59	80.95±5.78	23.25±0.39	216.56±11.88	25.20±0.47	18.32±1.81
MarCos	1.13±0.01	**17.97**±0.96	17.53±0.29	**24.39**±1.10	34.14±0.49	**23.67**±0.53	5.90±0.17
$-\ w/o\ \mathbf{Neu}^{deep}$	1.12±0.01	16.91±1.60	17.28±0.13	22.04±0.29	34.83±0.26	20.28±0.25	5.68±0.16
$-\ w/o\ \boldsymbol{R}_k$	0.77±0.01	16.07±0.00	17.20±0.10	20.27±0.00	32.43±0.11	16.66±0.00	5.65±0.12
Training Data: Equation							
CoT-SFT	0.38±0.01	20.31±0.85	32.97±1.03	27.92±0.36	162.12±7.43	41.48±0.50	9.61±0.91
iCoT-SI	0.43±0.03	14.81±0.55	13.11±1.62	18.46±0.54	17.30±0.31	21.33±0.43	2.32±0.05
Coconut	1.50±0.25	15.27±0.05	59.53±0.53	17.76±0.01	106.74±1.46	15.50±0.01	15.98±0.28
CoLaR	1.17±0.00	15.45±0.42	17.20±0.45	23.09±0.09	35.80±0.45	38.60±0.55	3.80±0.45
CODI	0.68±0.09	<u>16.07</u>±0.17	8.58±0.25	<u>26.24</u>±0.22	24.88±0.26	<u>40.03</u>±0.25	3.48±0.24
MarCos	1.03±0.01	**24.11**±0.97	17.76±0.04	**27.77**±0.32	34.02±0.11	**42.33**±0.56	6.02±0.28
$-\ w/o\ \mathbf{Neu}^{deep}$	0.99±0.01	21.07±0.76	17.09±0.30	25.83±0.19	35.24±0.11	38.38±1.04	6.00±0.29
$-\ w/o\ \boldsymbol{R}_k$	0.74±0.01	23.99±0.00	17.67±0.09	26.90±0.00	33.78±0.54	42.17±0.00	5.99±0.08

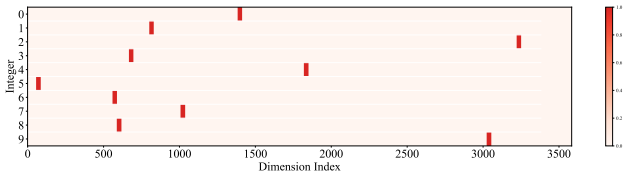


Figure 4. Emergent one-to-one encoding of randomness in $\boldsymbol{R}_k$. Each randomly chosen integer consistently activates a unique dimension in $\boldsymbol{R}_k$, regardless of the input sequence.

is better suited to randomness that admits a sequential factorization, but becomes severely limited when dealing with structured randomness that involves global thinking.

On the more challenging P24 task, MarCos achieves a substantial improvement compared with transformer. This large gap highlights the advantage of high-bandwidth continuous thinking in our thinking layer. By implicitly maintaining all parallel reasoning paths, MarCos explores the solution space far more effectively than transformer, which is important in settings with heavy branching and frequent replanning.

An interesting phenomenon is that our model learns to store perfectly clean, context-independent randomness in the random variable $\boldsymbol{R}_k$. For example, on the RS task, each randomly chosen integer consistently activates a unique dimension of $\boldsymbol{R}_k$, while leaving all other dimensions inactive (Figure 4). This pattern holds across ALL inputs. More importantly, during inference, if we fix $\mathbf{Neu}_{k-1}^{deep}, \mathbf{Neu}_{k-1}^{shallow}$ and manually modify $\boldsymbol{R}_k$ at a given step k, the decoded $\boldsymbol{s}_k$ and all subsequent reasoning steps change consistently according to this modification, without any error (accuracy = 100%). These findings demonstrate that MarCos explicitly separates randomness from contextual processing in a structured and interpretable manner. The thinking neurons are responsible for processing contextual information and performing the core reasoning computations, while $\boldsymbol{R}_k$ controls the direction of reasoning (i.e., the randomness).

4.2. Evaluation on Real-World Benchmarks

Having validated the efficacy of our MarCos on synthetic tasks, we now turn to larger-scale real-world tasks to further evaluate its potential on more challenging problems.

4.2.1. Experimental Setup

Datasets and Tasks. Following (Tan et al., 2025), we train our model on **GSM8K-Aug** dataset (Deng et al., 2023), which contains 385K training samples. Then, we evaluate on the original GSM8K test set (in-domain task), as well as two out-of-domain benchmarks: (1) **SVAMP** (Patel et al., 2021), consisting of 4,138 arithmetic problems constructed by subtle variations in wording and semantic perturbations, which aims to evaluate the robustness of reasoning, (2) **MultiArith** (Roy & Roth, 2015), a collection of 600 problems from MAWPS (Koncel-Kedziorski et al., 2016) that require multi-step reasoning. Our evaluation metrics include (1) Accuracy (Acc), which reflects the reasoning ability; (2) Training Time (Train), which measures the time cost for model training; and (3) Test Time (Test), which measures inference efficiency by counting the total time required to produce solutions for all test samples. More detailed descriptions of the evaluation setup are provided in Appendix C.

Baselines. We compare with (1) **CoT** (Wei et al., 2022), which trains transformer-based models on problem-solution pairs with explicit reasoning paths, (2) **iCoT-SI** (Deng et al., 2024), which gradually removes explicit reasoning steps during training to encourage implicit reasoning, (3) **Co-**

Table 3. Results of models initialized from pretrained checkpoints. For MarCos, we remove $\boldsymbol{R}_k$ since pretrained weights are not naturally suited for its stochastic role.

Model	GSM8K	SVAMP	MultiArith
Backbone: Qwen2.5-0.5B			
CoT	41.6	71.9	88.3
Coconut	28.6	60.7	88.0
CoLaR	36.0	61.7	82.2
CODI	40.8	71.2	87.6
MarCos(w/o $\boldsymbol{R}_k$)	**43.8**	**75.1**	**98.3**
Backbone: LLaMA-3.2-1B-Instruct			
CoT	**61.6**	66.7	**99.3**
Coconut	45.3	48.8	90.1
CoLaR	40.1	54.9	96.1
CODI	55.6	61.1	91.3
MarCos(w/o $\boldsymbol{R}_k$)	56.5	**68.5**	97.8

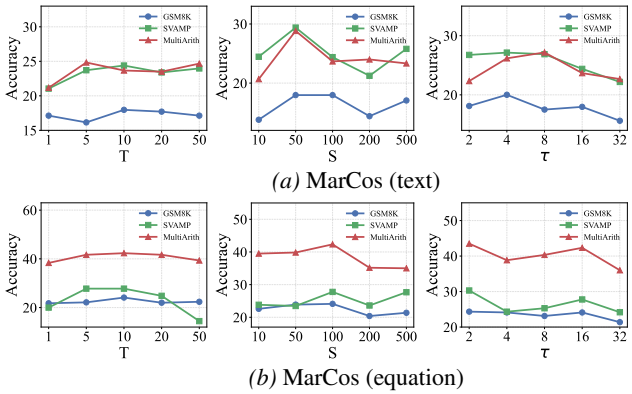
conut (Hao et al., 2024), which treats latent representations as a special token and inputs them alongside word tokens into an autoregressive model, (4) **CoLaR** (Tan et al., 2025), which compresses reasoning tokens and learns to predict the distribution of these compressed embeddings, and (5) **CODI** (Shen et al., 2025), which self-distills hidden activations across all layers at the selected distillation token.

Implementation Details. We implement the understanding, thinking, speaking layers and randomness encoder as 0.5B models (structure detailed in Appendix C). The randomness predictor is a two-layer MLP. For fair comparison, we implement baselines with both 0.5B and 3B parameters. For our model and CoT-SFT, we include both an equation-based version for direct comparison and a text-based version for real-world applications. For other baselines, we follow their original recipes to first train an explicit CoT for initialization and then finetune with only structured equations.

Since MarCos is a fundamentally new reasoning architecture, we primarily evaluate its efficacy by training from scratch to ensure that the observed reasoning capabilities stem from the architecture itself rather than external knowledge from pretraining. For a fair comparison, all baselines in the main experiments are trained from scratch using the same data. More details of experimental settings are provided in Appendix C. We also briefly discuss the pretraining of MarCos in Appendix G, which we leave for future work.

4.2.2. Main Results

As shown in Table 2 and 5, MarCos outperforms existing baselines in accuracy, while providing up to 15.7× speedup compared with traditional transformer-based CoTs. Specifically, MarCos (equation) achieves an 8.04% accuracy improvement over the best CODI baseline on GSM8K. Beyond in-domain performance, MarCos yields gains of up to 2.30% on SVAMP and MultiArith, which demonstrates strong out-of-domain generalization. Notably, MarCos also exceeds CoT-SFT in most settings. In particular, when trained with text supervision, MarCos improves accuracy by 4.70% on GSM8K. These results highlight its ability to execute implicit and deep reasoning effectively.



(a) MarCos (text)

(b) MarCos (equation)

Figure 5. Performance of MarCos with different numbers of neurons and random variables.

In terms of efficiency, MarCos not only outperforms token-based CoTs, but also achieves faster inference than many of the 0.5B baselines. This underscores that modeling reasoning at the thought level and decoupling the thinking and speaking bring substantial computational advantages. Remarkably, MarCos requires less training time than all 3B models, and even some 0.5B models. These findings demonstrate that our two-stage training scheme is both effective and exceptionally efficient.

Ablation Study. To show the importance of different components of MarCos, we compare it with two variants: w/o $\mathbf{Neu}^{deep}$, which removes the deep thinking neurons; w/o $\boldsymbol{R}_k$, which removes the randomness controller.

In Table 2, we observe that removing any of these components leads to a substantial performance drop. We also find that the relative importance of $\mathbf{Neu}^{deep}$ and $\boldsymbol{R}_k$ depends on the supervision type. When trained with equations, removing $\boldsymbol{R}_k$ has less impact than $\mathbf{Neu}^{deep}$, whereas with text supervision the opposite holds. This is because equations carry very little inherent randomness, while texts can exhibit more variability in aspects such as length and format, which will be validated in Section 4.2.3. We provide additional ablation studies, including attention direction, initialization strategy, and fixed λ, in Appendix D.

Attempting Initialization with Pretrained Transformers. In Table 3, we try to initialize all layers of MarCos from the pretrained Qwen2.5-0.5B and LLaMA-3.2-1B-Instruct checkpoint, with all baselines initialized accordingly for a fair comparison. However, we find that pretrained transformer weights are not naturally suited for the randomness

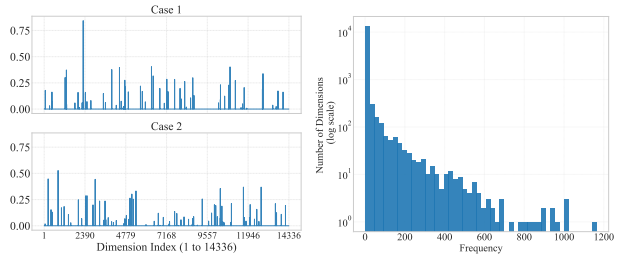


Figure 6. **Left**: $\boldsymbol{R}_k$ of step 1 for three cases, **Right**: Frequency distribution of all dimensions in $\boldsymbol{R}_k$.

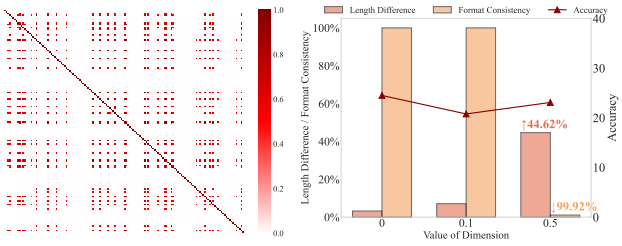


Figure 7. **Left**: Correlation heatmap of the top 1% most frequently activated dimensions in $\boldsymbol{R}_k$, **Right**: Results of intervening a clique of highly correlated dimensions in $\boldsymbol{R}_k$.

factor $\boldsymbol{R}_k$, as it serves a fundamentally different role from standard tokens. Therefore, we remove $\boldsymbol{R}_k$ in this setting to ensure a clean comparison. Under this setting, MarCos continues to achieve superior performance across all three benchmarks. Notably, it achieves substantial gains over CoT on SVAMP (+3.2% for Qwen; +1.8% for LLaMA) and MultiArith (+10.0% for Qwen). These results reflect that our architecture can initialize partial components from existing checkpoints to leverage their pretrained knowledge.

4.2.3. Analysis and Discussions

More thinking neurons, better performance? We first investigate how the scale of thinking neurons and random variables influences the model's performance by varying the number of deep thinking neurons $T \in \{1, 5, 10, 20, 50\}$, shallow thinking neurons $S \in \{10, 50, 100, 200, 500\}$, and random variables $\tau \in \{2, 4, 8, 16, 32\}$ in Figure 5.

For both types of data, as the numbers of neurons T and S increase, the model performance exhibits an initial increase, followed by a period of stabilization or decline. This suggests that performance is initially constrained by the model's thinking capacity, but beyond a certain point, adding more neurons may introduce noise or lead to overfitting.

Regarding the scale of randomness τ, we observe different phenomena on text- and equation-based models. For the equation setup, a large τ generally leads to worse performance, consistent with our earlier hypothesis in Section 4.2.2 that equation supervision contains relatively little variability. In contrast, for the text setup, good performance is achieved only with a large τ, indicating that a higher degree of stochasticity is required to adequately capture the diverse variations in natural language.

How does $\boldsymbol{R}_k$ control the randomness of reasoning? Beyond analyzing random variable $\boldsymbol{R}_k$ on synthetic tasks in Section 4.1.2, we extend our study to real-world reasoning tasks. As shown in Figure 6, different data activate different dimensions of $\boldsymbol{R}_k$. To quantify this effect, we define dimensions with values greater than 0.1 as *activated* and count activation frequencies across all samples. The right part of Figure 6 shows a long-tail pattern: while most dimensions are activated only once, a small subset of dimensions are activated at a very high frequency. This suggests that general randomness information may be controlled by these high-frequency dimensions, whereas case-specific randomness is reflected in the low-frequency ones.

To verify this, we select the top 1% most frequently activated dimensions and compute their pairwise Pearson correlation coefficients in Figure 7. Interestingly, we find that several dimensions are highly correlated with each other. In Appendix E, we further visualize this correlation structure as a graph, where we observe a clique of four dimensions that may represent a form of "combinational control". Specifically, we identify that these dimensions mainly control two factors: (1) *output length*, i.e., the number of tokens in the reasoning steps, and (2) *output format*, i.e., whether the final answer is preceded by the marker "####". To illustrate, we manually intervene on these dimensions by setting their values to $\{0, 0.1, 0.5\}$ for all data samples. Figure 7 shows that with higher values, the generated solutions become substantially longer. For example, at 0.5, the average length increases by 44.62%. Meanwhile, the strictness of format diminishes: at 0 and 0.1, almost all answers preserve the "####" prefix, whereas at 0.5 the proportion drops sharply to 1.08%. Crucially, despite these changes, reasoning accuracy remains largely unaffected (red line). These findings indicate that these dimensions independently control randomness factors such as verbosity and format, without encoding the essential reasoning content.

For the case-specific information, we perform a case-by-case analysis in Appendix F. First, we observe that several dimensions affect the *reasoning depth*. For example, when setting the dimensions A or C to 0.5, both cases exhibit deeper reasoning at Step 1, completing the original Step 1 and Step 2 simultaneously. Second, we identify distinct functional roles of certain dimensions. In Case 1, dimensions B control whether units are explicitly expressed in the equations (e.g., "4" vs. "4 pens"), whereas in Case 2, dimensions D determine whether numbers are represented as fractions ("$20/100$") or decimals ("0.20"). These findings reflect that our variational module effectively captures nuanced variations in reasoning.

Potential for non-autoregressive decoding in speaking stage. A key advantage of MarCos is the explicit sepa-

ration between thinking and speaking. This means that a clear idea about what to say is already formed before entering the speaking stage. As a result, there will be less stochasticity and lower correlation between tokens during sentence decoding, which makes MarCos well-suited for faster non-autoregressive generation. We verify this idea by directly training the speaking layer to generate 128 tokens in one pass. In Table 7, even with such a naive training recipe, MarCos still achieves performance comparable to baselines such as iCoT-SI in Table 2. This finding highlights the potential of MarCos to integrate more advanced NAR strategies, which is an important future direction.

5. Conclusion

In this work, we introduced MarCos, an improvement of the transformer architecture that performs stepwise reasoning entirely in a continuous space, modeled as a hidden Markov chain. Through extensive experiments, we showed that MarCos not only surpasses traditional transformers but also outperforms existing continuous reasoning methods while providing substantial inference speedup. Looking ahead, our top priority is to verify the effectiveness of MarCos when pretrained on massive corpus and develop dedicated reinforcement learning algorithms for the post-training of MarCos. For more discussions about this work's limitations and future directions, please refer to Appendix G.

Impact Statement

This paper presents work whose goal is to advance the field of machine learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Deng, Y., Prasad, K., Fernandez, R., Smolensky, P., Chaudhary, V., and Shieber, S. Implicit chain of thought reasoning via knowledge distillation. *arXiv preprint arXiv:2311.01460*, 2023.

Deng, Y., Choi, Y., and Shieber, S. From explicit cot to implicit cot: Learning to internalize cot step by step. *arXiv preprint arXiv:2405.14838*, 2024.

Fedorenko, E. and Varley, R. Language and thought are not the same thing: evidence from neuroimaging and neurological patients. *Annals of the New York Academy of Sciences*, 1369(1):132–153, 2016.

Fedorenko, E., Piantadosi, S. T., and Gibson, E. A. Language is primarily a tool for communication rather than thought. *Nature*, 630(8017):575–586, 2024.

Geiping, J., McLeish, S., Jain, N., Kirchenbauer, J., Singh, S., Bartoldson, B. R., Kailkhura, B., Bhatele, A., and Goldstein, T. Scaling up test-time compute with latent reasoning: A recurrent depth approach. *arXiv preprint arXiv:2502.05171*, 2025.

Giannou, A., Rajput, S., Sohn, J.-y., Lee, K., Lee, J. D., and Papailiopoulos, D. Looped transformers as programmable computers. In *International Conference on Machine Learning*, pp. 11398–11442. PMLR, 2023.

Gulrajani, I. and Hashimoto, T. B. Likelihood-based diffusion language models. *Advances in Neural Information Processing Systems*, 36:16693–16715, 2023.

Hao, S., Sukhbaatar, S., Su, D., Li, X., Hu, Z., Weston, J., and Tian, Y. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024.

Kambhampati, S., Valmeekam, K., Guan, L., Verma, M., Stechly, K., Bhambri, S., Saldyt, L. P., and Murthy, A. B. Position: Llms can't plan, but can help planning in llm-modulo frameworks. In *Forty-first International Conference on Machine Learning*, 2024.

Kanwisher, N. Functional specificity in the human brain: a window into the functional architecture of the mind. *Proceedings of the national academy of sciences*, 107 (25):11163–11170, 2010.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Koncel-Kedziorski, R., Roy, S., Amini, A., Kushman, N., and Hajishirzi, H. Mawps: A math word problem repository. In *Proceedings of the 2016 conference of the north american chapter of the association for computational linguistics: human language technologies*, pp. 1152–1157, 2016.

Lioubashevski, D., Schlank, T., Stanovsky, G., and Goldstein, A. Looking beyond the top-1: Transformers determine top tokens in order. *arXiv preprint arXiv:2410.20210*, 2024.

Miao, N., Rainforth, T., Mathieu, E., Dubois, Y., Teh, Y. W., Foster, A., and Kim, H. Learning instance-specific augmentations by capturing local invariances. In *International Conference on Machine Learning*, pp. 24720–24736. PMLR, 2023.

Patel, A., Bhattamishra, S., and Goyal, N. Are nlp models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2080–2094, 2021.

Roy, S. and Roth, D. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1743–1752, 2015.

Sahoo, S., Arriola, M., Schiff, Y., Gokaslan, A., Marroquin, E., Chiu, J., Rush, A., and Kuleshov, V. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37:130136–130184, 2024.

Shen, Z., Yan, H., Zhang, L., Hu, Z., Du, Y., and He, Y. Codi: Compressing chain-of-thought into continuous space via self-distillation. *arXiv preprint arXiv:2502.21074*, 2025.

Sprague, Z. R., Yin, F., Rodriguez, J. D., Jiang, D., Wadhwa, M., Singhal, P., Zhao, X., Ye, X., Mahowald, K., and Durrett, G. To cot or not to cot? chain-of-thought helps mainly on math and symbolic reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025.

Tan, W., Li, J., Ju, J., Luo, Z., Luan, J., and Song, R. Think silently, think fast: Dynamic latent compression of llm reasoning chains. *arXiv preprint arXiv:2505.16552*, 2025.

Valmeekam, K., Marquez, M., Sreedharan, S., and Kambhampati, S. On the planning abilities of large language models-a critical investigation. *Advances in Neural Information Processing Systems*, 36:75993–76005, 2023.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Wei, X., Liu, X., Zang, Y., Dong, X., Cao, Y., Wang, J., Qiu, X., and Lin, D. Sim-cot: Supervised implicit chain-of-thought. *arXiv preprint arXiv:2509.20317*, 2025.

Yang, L., Lee, K., Nowak, R. D., and Papailiopoulos, D. Looped transformers are better at learning learning algorithms. In *The Twelfth International Conference on Learning Representations*, 2024.

Zhang, C. et al. Cot2: Scaling reasoning via continuous thought augmentation. *arXiv preprint*, 2025.

Zhu, R.-J., Wang, Z., Hua, K., Zhang, T., Li, Z., Que, H., Wei, B., Wen, Z., Yin, F., Xing, H., et al. Scaling latent reasoning via looped language models. *arXiv preprint arXiv:2510.25741*, 2025.

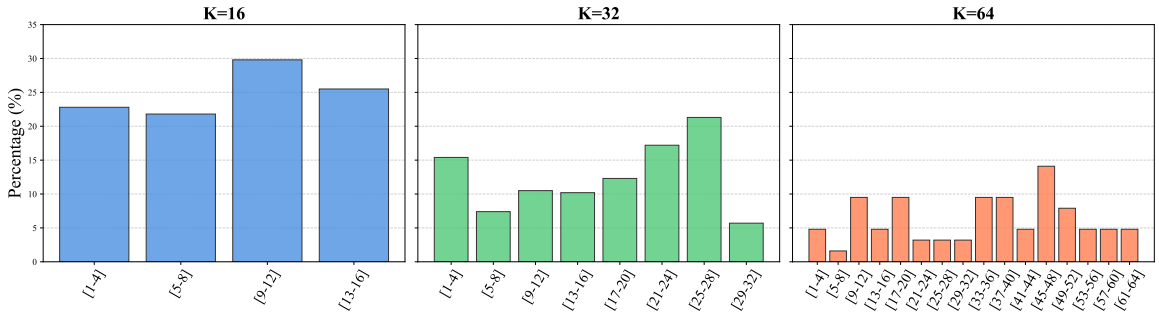


Figure 8. Distribution of selected source steps at test time for the long-range dependency task (%). Each bin covers 4 prior steps.

A. Markov Assumption and Long-Range Dependencies

A natural concern is whether the first-order Markov assumption in MarCos limits its ability to retain information over many reasoning steps. We emphasize that the thought at each step is represented by high-dimensional continuous vectors updated without discretization or information bottleneck, so the model has the inductive capacity to encode all necessary prior information. To empirically validate this, we design a task where the model randomly generates 5 numbers for $K-1$ steps and must randomly copy the numbers from one of these prior steps at the K-th step, requiring it to retain all $K-1$ prior steps' information. As shown in Table 4, MarCos successfully learns long-range dependencies at $K=16$ (88.2%) and maintains non-trivial performance even at $K=32$ and $K=64$, where CoT collapses entirely.

Table 4. Long-range dependency test: accuracy of retaining information from all prior steps.

K	16	32	64
CoT	92.5	0.0	0.1
MarCos	88.2	8.8	3.2

To address the concern that the model might be "cheating" by always copying from nearby steps, we analyze the distribution of the step selected by the model at test time. The target step to copy is uniformly randomly selected during training, so the model has no incentive to degenerate into a biased selection strategy. Figure 8 shows the test-time selection distribution. At $K=16$, the model's selections are well-distributed across all prior step bins. At larger K, while the distribution becomes slightly more uneven (as expected for harder tasks), the model still covers all step positions rather than concentrating on a fixed subset, confirming genuine long-range information retention.

B. Dynamic Weighting Strategy

Since it is difficult to predefine a proper balance between the reconstruction loss and the sparsity regularization, we adopt a simple dynamic weighting strategy (Miao et al., 2023) that directly controls the *expected sparsity level* during training.

Specifically, we introduce a target sparsity β and adjust λ based on the sparsity S_{batch} on the current training batch t:

$$\lambda_{t+1} = \begin{cases} 1.01 \cdot \lambda_t, & \text{if } S_{batch} > \beta, \\ 0.99 \cdot \lambda_t, & \text{if } S_{batch} \leq \beta. \end{cases} \tag{10}$$

We initialize λ_0 to 10^{-4} to avoid overly strong regularization in the early training stage. Empirically, we find that this strategy stably drives the sparsity level to converge to the target value β, while avoiding overly aggressive sparsity optimization that could otherwise cause the reconstruction loss to prematurely fall into poor local minima.

C. Details of Implementation and Evaluation

In MarCos, the understanding layer, speaking layer, and randomness encoder are realized by stacking standard transformer layers, using the same model configuration as Qwen2.5-0.5B. The thinking layer, in contrast, is a specialized module that updates a set of neurons and random variables. Therefore, it is a stack of transformer layers with bi-directional attention to

Table 5. Results of models **trained from scratch**. All continuous reasoning baselines (3B) are trained on equation data, following their original papers. We report the best continuous models in bold and the runner-ups with underline.

		GSM8K		SVAMP		MultiArith	
Model	Train (h)	Acc (%)	Test (s)	Acc (%)	Test (s)	Acc (%)	Test (s)
Training Data: Text							
CoT-SFT	$1.81_{\pm 0.00}$	$12.66_{\pm 0.20}$	$177.37_{\pm 1.91}$	$24.24_{\pm 0.50}$	$537.78_{\pm 12.14}$	$23.44_{\pm 1.40}$	$58.20_{\pm 2.50}$
MarCos	$1.13_{\pm 0.01}$	**17.97**$_{\pm 0.96}$	$17.53_{\pm 0.29}$	**24.39**$_{\pm 1.10}$	$34.14_{\pm 0.49}$	**23.67**$_{\pm 0.53}$	$5.90_{\pm 0.17}$
Training Data: Equation							
CoT-SFT	$1.23_{\pm 0.00}$	$22.70_{\pm 0.36}$	$54.07_{\pm 2.16}$	$27.25_{\pm 0.46}$	$125.47_{\pm 7.67}$	$50.20_{\pm 1.60}$	$10.73_{\pm 0.17}$
iCoT-SI	$2.40_{\pm 0.10}$	$14.08_{\pm 0.67}$	$13.23_{\pm 0.06}$	$16.26_{\pm 0.27}$	$29.02_{\pm 0.44}$	$19.67_{\pm 0.58}$	$3.49_{\pm 0.05}$
Coconut	$12.62_{\pm 1.06}$	$9.95_{\pm 0.04}$	$127.35_{\pm 0.44}$	$13.53_{\pm 0.11}$	$735.14_{\pm 3.80}$	$8.78_{\pm 0.19}$	$26.56_{\pm 0.66}$
CoLaR	$2.21_{\pm 0.00}$	$\underline{22.60}_{\pm 0.19}$	$38.13_{\pm 0.21}$	$\underline{25.83}_{\pm 0.22}$	$129.73_{\pm 0.96}$	$\underline{41.83}_{\pm 0.44}$	$18.93_{\pm 0.73}$
MarCos	$1.03_{\pm 0.01}$	**24.11**$_{\pm 0.97}$	$17.76_{\pm 0.04}$	**27.77**$_{\pm 0.32}$	$34.02_{\pm 0.11}$	**42.33**$_{\pm 0.56}$	$6.02_{\pm 0.28}$

allow each neuron to take the states of all other neurons into consideration. In other words, we do not modify the internal computations of transformers, nor is it the goal of our work. Instead, we repurpose the standard transformer to achieve fundamentally different roles in MarCos. In particular, through the design of the thinking layer, we demonstrate that a transformer-like structure can perform one-step reasoning within a single forward pass.

The numbers of deep and shallow neurons are set to $T = 10$, $S = 100$. The scaling factor of randomness variable $\boldsymbol{R}_k$ is set to $\tau = 16$. MarCos is trained by AdamW with learning rate $1\mathrm{e}{-4}$ and weight decay 0.01. The training batch size is 256, and both training phases run for 10 epochs. To simplify prototyping, the number of thinking steps is fixed to $K = 3$, although our model also supports dynamically deciding this number. For text data, each period (".") is treated as a separate step. If a sample contains more than three steps, the original steps are evenly grouped and recombined into three steps. For fair comparison, the CoT-SFT baseline is trained for 20 epochs. All experiments are conducted on a server with 8 H200 GPUs.

In our evaluation, Training Time is defined as the wall-clock time required to complete one epoch of training. To ensure a fair comparison, for baselines that require pre-training an explicit CoT model, the reported training time is the sum of (i) the time to train the explicit CoT model for one epoch, and (ii) the time to train their own model for one epoch. For our model, the reported training time is the sum of the two training phases, each measured over one epoch. Test Time is the total time required to infer all test samples on a single GPU, using a batch size of 64. However, the original implementation of Coconut only supports inference with batch size = 1. We attempted to modify it to 64, but observed no significant improvement in efficiency. Therefore, for Coconut we report the time of its default setting, i.e., 4 GPUs with batch size = 1.

D. Additional Ablation Studies

We provide additional results to examine the sensitivity of MarCos to architectural and hyperparameter choices. (1) **Attention direction.** We compare bidirectional attention (our default) with unidirectional attention in the thinking layer. As shown in Table 6, bidirectional attention consistently outperforms unidirectional attention, confirming that allowing neurons to attend to all other neurons is beneficial for reasoning. (2) **Initialization strategy.** The performance differences between our default initialization with Xavier and Normal initialization for the thinking layer are marginal across all three benchmarks, indicating that MarCos is robust to initialization choice. (3) **Fixed λ.** Although MarCos adopts an adaptive dynamic weighting strategy for λ, we also evaluate fixed values for completeness. From Table 6, fixing λ to any of the tested values degrades performance compared to the adaptive strategy, motivating the need for dynamic balancing.

Table 6. Ablation on attention direction and initialization strategy (equation data).

Ablation	Setting	GSM8K	SVAMP	MultiArith
Attention	Bidirectional (default)	24.11	27.77	42.33
Attention	Unidirectional	22.74	25.35	38.82
λ	10^{-4}	21.32	23.98	40.54
λ	5×10^{-4}	20.83	23.90	39.43
λ	10^{-3}	20.83	23.82	36.39

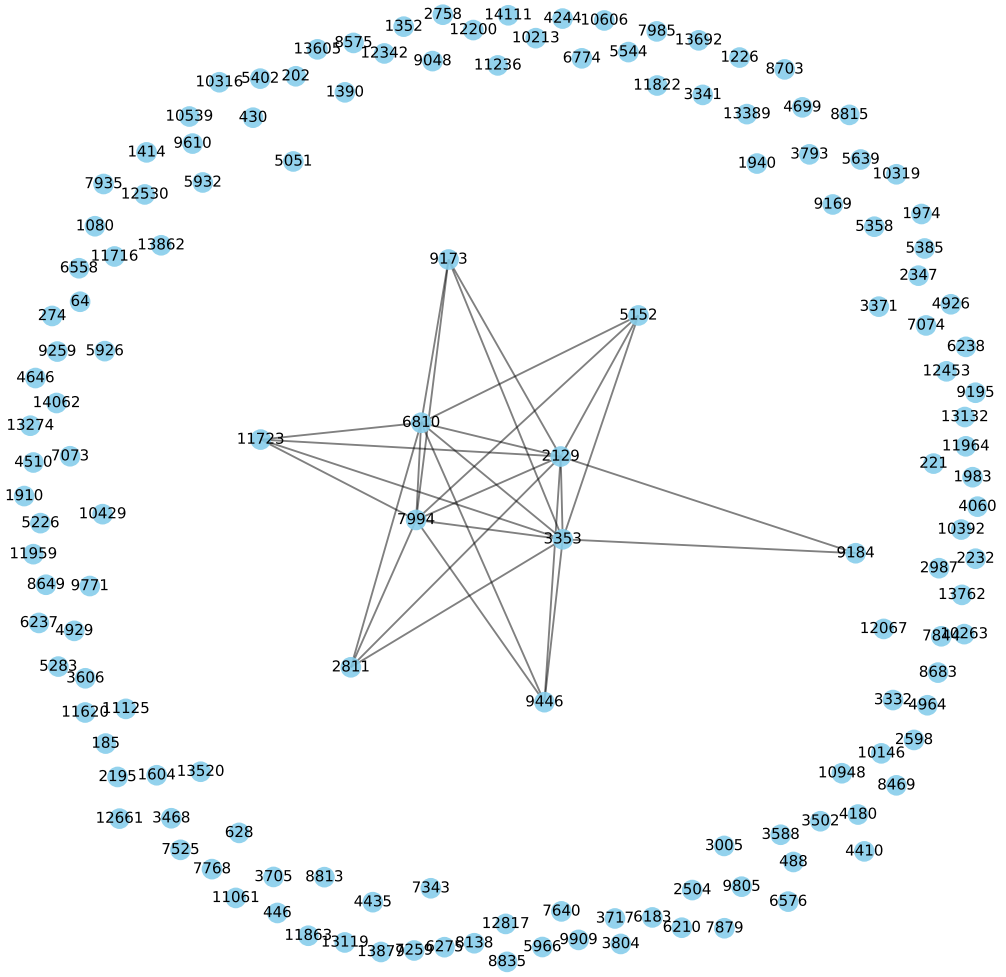


Figure 9. Correlation graph of the top 1% most frequently activated dimensions in $\boldsymbol{R}_k$.

E. Visualization of Dimension Correlation Structure

Figure 9 visualizes the correlation graph described in Section 4.2.3. In this graph, each node represents one of the top 1% most frequently activated dimensions, and an edge is drawn between two nodes if their Pearson correlation exceeds 0.9. From the figure, we can clearly observe a clustered structure, suggesting that some dimensions tend to be co-activated in a highly consistent manner. Notably, within the cluster we further identify a clique formed by dimensions $\{2129, 3353, 6810, 7994\}$, which indicates that these dimensions jointly control aspects of general randomness.

F. Case Study

Figure 10 shows how dimensions in $\boldsymbol{R}_k$ control specific information. We highlight in red the differences between the controlled outputs and the original solutions. The dimension sets are: A=$\{$ 199, 3505, 5283, 7879, 9259, 9468, 9485, 11725$\}$, B=$\{4060\}$, C=$\{1379, 1583, 5475, 7017, 7074, 7521, 8683, 9635, 10460, 10539, 11716, 12012, 13337\}$, D=$\{1226\}$.

As illustrated in Section 4.2.3, certain dimensions affect the *reasoning depth*, such as A and C. The difference between these two dimensions may arise because dimension A primarily affects the final equation, whereas dimension C influences the penultimate equation. This indicates that the stored information in random variables goes beyond surface-level linguistic style, and can in fact control diverse reasoning pathways. Other dimensions control finer-grained aspects of the solutions, such as the explicit expression of units (B) or numerical format (D).

Case 1: Raphael went to buy some school supplies. He bought 4 pens which cost $1.5 each, 2 notebooks which cost $4 each, and a rim of bond paper which cost $20. How much did Raphael spend on everything?	**Original Solution:** *Step1:* Raphael spent 4 x $1.5 = $6 on pens. He spent 2 x $4 = $8 on notebooks. *Step2:* Therefore, he spent a total of $6 (for pens) + $8 (for pens) + $20 (for notebooks) = $34. *Step3:* #### 34.	$R_k(A) = 0.5$: *Step1:* Raphael spent 4 × $1.5 = $6 on pens He spent 2 × $4 = $8 on notebooks. In total, he spent $6 + $8 + $20 = $34. *Step2:* Therefore, he spent a total of $34. *Step3:* #### 34.	$R_k(B) = 0.1$: *Step1:* Raphael spent 4 pens × $1.5/pen = $6 on pens. He spent 2 notebooks × $4/notebook = $8 on notebooks. *Step2:* Therefore, he spent a total of $6 + $8 + $20 = $34 for the items. *Step3:* #### 34.
Case 2: Julie had $500. She spent 20% of it on clothes and then 25% of the remaining money on CDs. How much money did Julie have left?	**Original Solution:** *Step1:* Julie spent $500 * 20/100 = $100 on clothes. She had $500 - $100 = $400 left. *Step2:* She then spent $400 * 25/100 = $100 on CDs. So, she had $400 - $100 = $300 left. *Step3:* #### 300.	$R_k(C) = 0.5$: *Step1:* Julie had $500 * 20/100 = $100 left after buying clothes. She had $500 - $100 = $400 left after buying CDs. She then spent 25/100 * $400 = $100 on CDs. *Step2:* She had $400 and she spent $100 so she has 400-100 = $300 left. *Step3:* #### 300.	$R_k(D) = 0$: *Step1:* Julie spent $500 * 0.20 = $100 on clothes. She had $500 - $100 = $400 left. *Step2:* She then spent $400 * 25/100 = $100 on CDs. So, she had $400 - $100 = $300 left. *Step3:* #### 300.

Figure 10. Case study of controlling four sets of dimensions in $\boldsymbol{R}_k$ ($k = 1, ..., K$). For each case, the difference of model output after intervention is highlighted in red.

Table 7. Accuracy of NAR decoding at the speaking stage.

Method	GSM8K	SVAMP	MultiArith
MarCos (text)	9.02	14.96	8.33
MarCos (equation)	14.25	22.82	30.12

G. Limitations and Future Directions

The following are some limitations in this work, which also point to promising directions for future research. First, the experiments in this paper primarily focus on mathematical reasoning tasks. This choice is motivated by the fact that transformer-based reasoning has proven to be most effective in this domain (Sprague et al., 2025), and most existing reasoning models are also benchmarked under mathematical scenarios. Second, our evaluation is conducted on relatively basic mathematical problems, rather than more challenging datasets such as MATH or AIME. On one hand, our goal in this paper is to demonstrate the effectiveness and potential of MarCos as a continuous reasoning structure, rather than pursuing leaderboard-oriented results. On the other hand, tackling complex mathematical problems may require additional techniques such as reinforcement learning, which we leave for future work. Third, as discussed in Section 4.2.1, latent thinking represents a new paradigm for transformers. To fully realize its potential, we need to explore corresponding pretraining methods and data, enabling the model to acquire broader and more robust capabilities across diverse domains.

Beyond these limitations, our study also opens several avenues for future work. First, and most importantly, is to develop a pre-training methodology for MarCos. The key challenge may be the definition and segmentation of reasoning steps, as such large, unstructured corpora lack explicit, step-by-step annotations. One potential solution is to treat each sentence as a single reasoning step. Under this setup, the pre-training objective can be formulated as given the preceding sentences (analogous to the question in our paper), the model needs to generate the subsequent K sentences. It is compatible with our existing training scheme, which could be scaled up. Second, as validated in Section 4.2.3, our random variable enables step-level control over stochasticity, offering a new perspective for exploration in reinforcement learning. Unlike conventional token-level sampling, our approach allows sampling reasoning pathways at a higher level, potentially making exploration more efficient. Third, our framework shows that randomness can be simulated "once" rather than being iteratively modeled, which may inspire new directions for diffusion language models. Last but not least, our work demonstrates the feasibility of disentangling thinking from speaking, suggests that speaking can be treated as a pure translation process. Therefore, it is worth trying to introduce more efficient decoding techniques, especially NAR decoding, to further accelerate inference.

H. Statement on the Use of LLMs

We used LLMs to improve the clarity, grammar, and fluency of our paper. The LLMs were not involved in any technical idealizations, derivations, or generation of research content. We have carefully reviewed all content generated by LLMs to ensure that no factual or technical inaccuracies were introduced, and the scientific integrity of the work is fully maintained.