

# 000 001 002 003 004 005 C-VOTING: CONFIDENCE-BASED TEST-TIME VOTING 006 WITHOUT EXPLICIT ENERGY FUNCTIONS 007 008 009

010 **Anonymous authors**  
011 Paper under double-blind review  
012  
013  
014  
015  
016  
017  
018  
019  
020  
021  
022  
023  
024  
025  
026  
027  
028  
029

## 030 ABSTRACT

031 Neural network models with latent recurrent processing, where identical layers  
032 are recursively applied to the latent state, have gained attention as promising mod-  
033 els for performing reasoning tasks. A strength of such models is that they en-  
034 able test-time scaling, where the models can enhance their performance in the  
035 test phase without additional training. Models such as the Hierarchical Reason-  
036 ing Model (HRM) and Artificial Kuramoto Oscillatory Neurons (AKOrN) can  
037 facilitate deeper reasoning by increasing the number of recurrent steps, thereby  
038 enabling the completion of challenging tasks, including Sudoku, Maze solving,  
039 and AGI benchmarks. In this work, we introduce confidence-based voting (C-  
040 voting), a test-time scaling strategy designed for recurrent models with multiple  
041 latent candidate trajectories. Initializing the latent state with multiple candidates  
042 using random variables, C-voting selects the one maximizing the average of top-1  
043 probabilities of the predictions, reflecting the model’s confidence. Additionally, it  
044 yields 4.9% higher accuracy on Sudoku-hard than the energy-based voting strat-  
045 egy, which is specific to models with explicit energy functions. An essential  
046 advantage of C-voting is its applicability: it can be applied to recurrent mod-  
047 els without requiring an explicit energy function. Finally, we introduce a simple  
048 attention-based recurrent model with randomized initial values named ItrSA++,  
049 and demonstrate that when combined with C-voting, it outperforms HRM on  
050 Sudoku-extreme (95.2% vs. 55.0%) and Maze (78.6% vs. 74.5%) tasks.  
051  
052

## 053 1 INTRODUCTION

054 In recent years, reasoning has been recognized as crucial for achieving Artificial General Intelligence  
055 (AGI). Recurrent models, which repeat identical layers, have emerged as a promising approach to  
056 achieve the goal. A key advantage of recurrent models is that their performance can be enhanced at  
057 test time without additional training, a technique known as test-time scaling. The test-time scaling  
058 is typically realized in two ways: (1) increasing the number of inference recurrence steps, and (2)  
059 selecting a “good” trajectory from multiple candidates with random sampling, or *voting*. The method  
060 of increasing inference steps (1) has been investigated in (Anil et al., 2022) and other studies. The  
061 method of selecting “good” trajectories (2) has recently attracted attention in large language models  
062 (LLMs), particularly in the context of self-consistency (Wang et al., 2023) as well as search and  
063 decoding strategies (Yao et al., 2023; Zhou et al., 2022).

064 Recent recurrent models, such as the Hierarchical Reasoning Model (HRM) (Wang et al., 2025) and  
065 the Artificial Kuramoto Oscillatory Neurons (AKOrN) (Miyato et al., 2025), have become capable  
066 of solving complex tasks, including Sudoku and Maze by utilizing test-time scaling. Sudoku and  
067 Maze are regarded as benchmarks for reasoning because they require consistent logical inference  
068 under complex constraints. These are challenging tasks for conventional models, including leading  
069 LLMs as reported in (Wang et al., 2025). HRM mainly relies on increasing recurrence depth,  
070 whereas AKOrN introduces energy-based voting (E-voting) in addition to increasing the depth. E-  
071 voting samples multiple initial latent states and selects the final trajectory with the lowest energy,  
072 yielding large performance gains in Sudoku. It enables further improvement even in cases where  
073 performance improvement from increasing the number of recurrent steps is saturated. Specifically,  
074 it has been reported that using 4096 candidates resulted in an approximately 40% increase in the  
075 board accuracy (Miyato et al., 2025). Although highly effective, it has the limitation that it cannot  
076 be used unless the energy function is explicitly defined. In most promising reasoning models, such

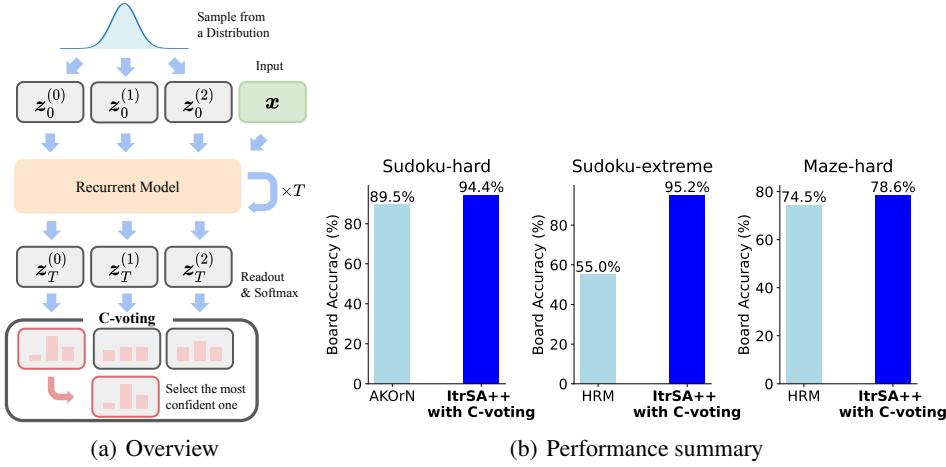


Figure 1: (a): The overview of confidence-based voting (C-voting). The candidates of the latent state are initialized by random variables, and the recurrent model updates the latent states from  $t = 0$  to  $t = T$  using input  $x$ . Calculating probabilities from the final states, C-voting selects the most confident prediction. (b): The performance comparisons of ItrSA++ with C-voting to other state-of-the-art models. The proposed method outperforms in Sudoku-hard, Sudoku-extreme, and Maze-hard tasks.

as HRM and recurrent transformers (Geiping et al., 2025), the energy is not explicitly defined, making it impossible to use E-voting. This limitation raises natural questions: (RQ1) Can we facilitate a model-agnostic test-time voting strategy without explicit energy functions, and (RQ2) Does there exist a simple, lightweight recurrent architecture matching or surpassing state-of-the-art performance with C-voting?

In this work, we propose confidence-based voting (C-voting), a novel voting strategy at test time, which enhances performance on a broader range of recurrent models. C-voting only requires the recurrent models to start from randomly sampled initial values, and does not require explicit energy functions. The model initializes the latent states with random variables, updates them applying an identical model repeatedly, and performs inferences. With these sampled trajectories, the rule of C-voting is very simple: take the most confident one (Figure 1(a)). We measure the confidence of a trajectory through the average top-1 probability that the trajectory finally provides. We experimentally find that C-voting offers better performance than E-voting in the case of Sudoku-solving with AKOrN.

C-voting can be applied not only to existing models, but also to newly designed, more powerful recurrent models. As an example, we introduce ItrSA++, a simple recurrent model with randomized initial values. It outperforms HRM on Sudoku-extreme (95.2% vs. 55.0%) and Maze-hard tasks (78.6% vs. 74.5%), and AKOrN on Sudoku-hard task (94.4% vs. 89.5%) as shown in Figure 1(b). The model only requires about 3 million parameters, which is about one-ninth the number of parameters in HRM, and comparable to that of AKOrN. **We also demonstrate that C-voting can be applied to HRM with C-voting.** This represents that C-voting is a simple and portable test-time voting strategy that enables recurrent models to achieve state-of-the-art performance with significantly fewer parameters.

## 2 PRELIMINARIES

### 2.1 RECURRENT MODELS WITH RANDOMIZED INITIALIZATION

Recent works show that recurrent models, which apply identical layers iteratively, are a powerful and promising framework for a variety of complex reasoning tasks (Dehghani et al., 2019; Anil et al., 2022; Geiping et al., 2025; Jaegle et al., 2021; Gladstone et al., 2025; Wang et al., 2025; Miyato et al., 2025; Saunshi et al., 2025; Darlow et al., 2025). Concretely, these recurrent models typically

108 have the form of

$$110 \quad z_{i,t+1} = f(z_{i,t}, x_i; \theta), \quad (1)$$

111 where  $f$  is a neural network,  $z_{i,t} \in \mathbb{R}^{L \times C}$  the latent state at layer  $t (\in \mathbb{N})$ ,  $x_i \in \mathbb{R}^{L \times C}$  the  $i$ -th input  
 112 (e.g., image, Sudoku board with blanks) embedded in the latent space, and  $\theta$  the parameters.  $L \in \mathbb{N}$   
 113 is the number of tokens, and  $C \in \mathbb{N}$  is the embedding dimensions. For models that solve tasks with  
 114 complex reasoning, such as Sudoku or Maze tasks,  $f$  is oftentimes set to have a Transformer-based  
 115 architecture (Vaswani et al., 2017; Dehghani et al., 2019; Saunshi et al., 2025). Note that the network  
 116  $f(\cdot, \cdot; \theta)$  is identical for every  $t$ , hence equation 1 can be thought of as a time-invariant nonlinear  
 117 dynamical system.

118 The behavior of the latent dynamics equation 1 depends on the initialization of  $z_{i,0}$ , which possibly  
 119 improves or deteriorates the model’s performance. One strategy is to sample  $z_{i,0}$  from a probability  
 120 distribution, e.g., a standard Gaussian distribution. Such randomized initialization can lead the  
 121 model to learn path independence, or the convergence to the same final steady state (Anil et al.,  
 122 2022), which helps the model to generalize better than initializing the dynamics with a fixed value,  
 123 e.g.,  $z_{i,0} \equiv 0$ .

124 After repeating equation 1 from  $t = 0$  to  $t = T - 1 (T \in \mathbb{N})$ , the final latent state  $z_{i,T}$  is passed to  
 125 the readout module. In classification problems such as Sudoku, the readout module produces logits,  
 126 which are subsequently converted to class probabilities by the softmax function or its variant.

127 Recurrent models with randomized initialization are beneficial not only in parameter efficiency but  
 128 also in that they allow *test-time scaling*. Test-time scaling is a phenomenon where the model en-  
 129 hances its performance without further training, and is observed in a number of recurrent models  
 130 (Gladstone et al., 2025; Geiping et al., 2025; Miyato et al., 2025; Hu et al., 2025). Test-time scaling  
 131 can be realized in the following two ways: one is to run more iterations at inference than in training;  
 132 the second is to select from the trajectories, which are initialized randomly, the “best” one based on  
 133 a pre-determined criterion.

134 Although the term “test-time scaling” typically refers only to the concept of extending iterative  
 135 steps, which has been studied extensively (Anil et al., 2022; Geiping et al., 2025), here we place  
 136 our focus on the other concept, that is, selecting one trajectory from the sampled ones. We consider  
 137 that this “sample & choose”, or *voting* strategy, is as significant as extending iterative steps, as the  
 138 voting strategy can compensate for the next two shortcomings of iterative step extension. First, the  
 139 performance of a recurrent model cannot grow monotonously with extending iterative steps: the per-  
 140 formance can saturate. Second, extending the iterative steps in the test phase requires long iterative  
 141 computations and cannot be parallelized. The voting strategy works well with parallel computing  
 142 and can enhance the performance even when it is saturated. Therefore, voting can enhance the  
 143 reasoning ability to the point where the step extension cannot be solely reached.

### 144 3 RELATED WORK

#### 145 3.1 RECURRENT MODELS WITH RANDOMIZED INITIALIZATION

146 As representative examples of the recurrent models with randomized initial values, we focus on  
 147 HRM (Wang et al., 2025) and AKOrN (Miyato et al., 2025), as they demonstrate outstanding per-  
 148 formance in logical tasks such as Sudoku and Maze.

149 **HRM** HRM (Wang et al., 2025) is a recurrent model that is gaining attention for its strong rea-  
 150 soning abilities for the relatively small size ( $\approx 27$  million parameters), reaching SoTA in Sudoku-  
 151 extreme, Maze-hard, and ARC-AGI challenges (Chollet, 2019; Chollet et al., 2025). The latent  
 152 dynamics is a two-fold model that has *higher* and *lower* order processing given as follows:

$$153 \quad z_{i,t+1}^L = f^L(z_{i,t}^L, z_{i,t}^H, x; \theta^L), \\ 154 \quad z_{i,t+1}^H = \begin{cases} f^H(z_{i,t}^H, z_{i,t}^L; \theta^H) & \text{if } t \equiv 0 \pmod{\tau} \\ z_{i,t}^H & \text{otherwise} \end{cases}, \quad (2)$$

155 where  $\tau$  is an update cycle for higher-level layers. The superscripts H and L respectively represent  
 156 higher and lower order processings. This two-fold structure was first proposed as the main break-  
 157 through but later shown to have a limited effect on the performance (Team, 2025). HRM makes use

of other techniques to boost its performance, such as one-step gradient approximation (taking its origin from Deep Equilibrium Model (Bai et al., 2019)) or adaptive computational time (ACT).

**AKOrN** AKOrN (Miyato et al., 2025) is another recurrent model that shows SoTA performance in the Sudoku-hard task (Palm et al., 2018). AKOrN utilizes the  $C$ -dimensional Kuramoto oscillators (Kuramoto, 1975; Lipton et al., 2019) as the latent dynamics and given as:

$$\Delta \mathbf{z}_{i,t} = \Omega(\mathbf{z}_{i,t}) + \text{Proj}_{\mathbf{z}_{i,t}}(\mathbf{x}_i + \mathbf{J}(\mathbf{z}_{i,t})), \quad (3)$$

$$\mathbf{z}_{i,t+1} = \Pi(\mathbf{z}_{i,t} + \eta \Delta \mathbf{z}_{i,t}). \quad (4)$$

Here, all rows of  $\mathbf{z}_{i,t}$  are confined on the unit hypersphere (thus have norm one).  $\Omega : \mathbb{R}^{L \times C} \rightarrow \mathbb{R}^{L \times C}$  is a linear map that rotates each token on the hypersphere, and thus is represented as a set of skew-symmetric matrices.  $\mathbf{J} : \mathbb{R}^{L \times C} \rightarrow \mathbb{R}^{L \times C}$  is the connectivity strength between each token pair, and is implemented using self-attention. The projection onto the tangent space  $\text{Proj}$  and normalization onto the hypersphere  $\Pi$  are applied token-wise.

### 3.2 TEST-TIME SCALING WITH E-VOTING

E-voting is a test-time voting strategy introduced in (Miyato et al., 2025). E-voting can be applied to the recurrent model equipped with an energy (scalar) function —i.e., the recurrent models where a scalar function  $E : \mathbb{R}^{L \times C} \rightarrow \mathbb{R}$  exists, such that

$$\mathbf{z}_{i,t+1} = f(\mathbf{z}_{i,t}, \mathbf{x}_i; \boldsymbol{\theta}) = \mathbf{z}_{i,t} - \alpha \nabla_{\mathbf{z}} E(\mathbf{z}_{i,t}; \boldsymbol{\theta}), \quad \alpha \in \mathbb{R}_+. \quad (5)$$

This dynamics proceeds in the direction in which the energy (expectedly) decreases when  $\alpha$  is small enough. For example, AKOrN has a closed-form energy function<sup>1</sup>, whereas energy-based transformer (Gladstone et al., 2025) explicitly models an energy using a Transformer-based architecture. As the model learns the dynamics to decrease the energy function, a smaller energy value is expected to reflect higher accuracy. E-voting is based on this assumption, and proceeds as follows. We first sample initial states  $\mathbf{z}_{i,0}$  for  $K \in \mathbb{N}$  times from a probability distribution, which we denote by  $\{\mathbf{z}_{i,0}^{(k)}\}_{k \in [1, K]}$ . We then select the sample with the lowest energy at the final step  $t = T$ , i.e.,

$$k^* = \arg \min_k E(\mathbf{z}_{i,T}^{(k)}; \boldsymbol{\theta}). \quad (6)$$

This voting strategy is effective in the Sudoku-solving task; in fact, AKOrN achieves about a 40% improvement with E-voting.

Although E-voting is an effective voting strategy, it is not available in many models since the energy function is not explicitly obtained. For example, in recurrent models with a residual connection

$$\mathbf{z}_{i,t+1} = \mathbf{z}_{i,t} + g(\mathbf{z}_{i,t}; \boldsymbol{\theta}), \quad (7)$$

$g$  has to be written as the gradient of some scalar function, which does not exist in general.

## 4 CONFIDENCE-BASED VOTING

We describe the detailed procedure of the proposed voting method, C-voting. C-voting, as the name indicates, is a simple voting strategy applicable at test time. Concretely, C-voting applies the following steps to the trained model. We first sample initial states  $\mathbf{z}_{i,0}$  for  $K \in \mathbb{N}$  times from a probability distribution, which we denote by  $\{\mathbf{z}_{i,0}^{(k)}\}_{k \in [1, K]}$ . From these sampled initial states, we update the latent state with a neural network  $f$  by equation 1. Reading out the final state  $\mathbf{z}_{i,T}^{(k)}$  as logits, we obtain the probability of the class  $j$  at the position  $l$  of  $k$ -th candidate as

$$P_{j,l}(\mathbf{z}_{i,T}^{(k)}) = \text{Softmax} \left( \text{Readout}(\mathbf{z}_{i,T}^{(k)}) \right)_{j,l}, \quad (8)$$

<sup>1</sup>Note that equation 5 does not strictly hold when  $\mathbf{J}$  in equation 3 is asymmetric. Nevertheless, E-voting is shown to be effective for AKOrN in such a case.

216 where the softmax function is applied in the dimension of classes. The position  $l$  indicates the  
 217 location at which the probability is computed, e.g., a cell in Sudoku. Then, the top-1 probability at  
 218 the position  $l$  is

$$219 \quad 220 \quad 221 \quad \hat{P}_l(\mathbf{z}_{i,T}^{(k)}) = \max_j P_{j,l}(\mathbf{z}_{i,T}^{(k)}). \quad (9)$$

222 The average of top-1 probabilities over all positions reflects the confidence of the model. Therefore,  
 223 we define the model’s confidence for the  $i$ -th input and the  $k$ -th candidate as

$$224 \quad 225 \quad 226 \quad C_i^{(k)} = \frac{1}{|\mathcal{L}|} \sum_{l \in \mathcal{L}} \hat{P}_l(\mathbf{z}_{i,T}^{(k)}), \quad (10)$$

227 where  $\mathcal{L}$  denotes the set of positions to be predicted, e.g., indices of blank cells in Sudoku, and  $|\cdot|$   
 228 denotes the number of elements in the set. We select the index of the most confident candidate by

$$229 \quad 230 \quad 231 \quad k_i^* = \arg \max_k C_i^{(k)}. \quad (11)$$

232 Thus, the prediction with the C-voting for input  $i$  at position  $l$  is obtained by

$$233 \quad 234 \quad 235 \quad \hat{y}_{i,l}^* = \arg \max_j P_{j,l}(\mathbf{z}_{i,T}^{(k^*)}). \quad (12)$$

236 If the model is well calibrated, the following approximation holds

$$237 \quad 238 \quad 239 \quad \Pr[y_{i,l} = \hat{y}_{i,l}^* | \hat{P}_l(\mathbf{z}_{i,T}^{(k)})] \simeq \hat{P}_l(\mathbf{z}_{i,T}^{(k)}), \quad (13)$$

240 where  $\hat{y}_{i,l}^* = \arg \max_j P_{j,l}(\mathbf{z}_{i,T}^{(k)})$ . Using equation 13, we obtain the index of the candidate that  
 241 maximizes average accuracy across all positions by

$$242 \quad 243 \quad 244 \quad \arg \max_k \frac{1}{|\mathcal{L}|} \sum_{l \in \mathcal{L}} \Pr[y_{i,l} = \hat{y}_{i,l}^* | \hat{P}_l(\mathbf{z}_{i,T}^{(k)})] \simeq \arg \max_k \frac{1}{|\mathcal{L}|} \sum_{l \in \mathcal{L}} \hat{P}_l(\mathbf{z}_{i,T}^{(k)}) = k_i^*. \quad (14)$$

245 Note that the average accuracy across all positions may not be the same as the objectives. Nevertheless,  
 246 it would be a good proxy for them.

## 249 5 ItrSA++

251 As introduced in subsection 2.1, recurrent models, such as AKOrN and HRM, incorporate various  
 252 techniques that may be redundant with C-voting. To verify if higher performance can be achieved  
 253 with a simpler and optimized model for C-voting, we introduce ItrSA++, a simple recurrent model to  
 254 which C-voting can be applied. The design principles of ItrSA++ are as follows: (1) Cross-attention  
 255 is employed to mix random initialization with input, similar to Perceiver (Jaegle et al., 2021). As in  
 256 (Geiping et al., 2025), a linear layer was also tested, but experiments showed that cross-attention per-  
 257 formed better. (2) Rather than recurrent transformers, we adopt recurrent attention layers to maintain  
 258 a simple architecture. Experiments showed that periodically inserting SwiGLU (Shazeer, 2020) out-  
 259 performs, so we adopt it. (3) The normalization method was determined experimentally. As noted  
 260 in Geiping et al. (2025), normalization has a significant impact on the performance. [Appendix D](#)  
 261 [confirms these design choices](#).

262 Concretely, ItrSA++ is defined as follows. At first, we embed the input  $\mathbf{x}_i$  using an embedding layer  
 263 and apply normalization.

$$264 \quad 265 \quad \mathbf{x}_i^{\text{emb}} = \text{Norm}_{\text{attn}}(\text{Embedding}(\mathbf{x}_i)). \quad (15)$$

266 Hereafter, we use RMSNorm (Zhang & Sennrich, 2019) for the normalization. The initial latent  
 267 state is sampled from a standard normal distribution. To mix the information of the input and the  
 268 latent state, we use cross-attention as follows;

$$269 \quad \tilde{\mathbf{z}}_{i,t} = \text{Norm}_{\text{attn}}(\mathbf{z}_{i,t}) + \text{CrossAttn}(q = \text{Norm}_{\text{attn}}(\mathbf{z}_{i,t}), k = \mathbf{x}_i^{\text{emb}}, v = \mathbf{x}_i^{\text{emb}}). \quad (16)$$

270 After mixing the information, we apply the self-attention layer  $S \in \mathbb{N}$  times.  
 271

$$\bar{z}_{i,t,s+1} = \text{Norm}_{\text{attn}}(\bar{z}_{i,t,s}) + \text{SelfAttn}(\text{Norm}_{\text{attn}}(\bar{z}_{i,t,s})), \quad \bar{z}_{i,t,0} = \bar{z}_{i,t}, \quad (17)$$

273 At the end of the iterative self-attention, we apply SwiGLU (Shazeer, 2020) and obtain  
 274

$$z_{i,t+1} = \text{Norm}_{\text{mlp}}(\bar{z}_{i,t+1,S}) + \text{SwiGLU}(\text{Norm}_{\text{mlp}}(\bar{z}_{i,t+1,S})). \quad (18)$$

276 By repeating a block composed of equation 16, equation 17, and equation 18  $T$  times, we obtain the  
 277 final latent state  $z_{i,T}$ . The readout module consists of a normalization and a linear layer, defined by  
 278

$$\text{logit}_i = W_O \text{Norm}_{\text{out}}(z_{i,T}). \quad (19)$$

280 We use Geometry-Aware Attention Mechanism (Miyato et al., 2024) as the position embedding.  
 281

## 282 6 EXPERIMENTS

### 284 6.1 BENCHMARKS

286 Sudoku is a logical puzzle in which the digits 1 to 9 are placed in  $9 \times 9$  cells. The cells must be  
 287 filled with digits to satisfy the following constraints: no digit appears more than once in any row,  
 288 any column, and any of the nine  $3 \times 3$  sub-grids. Several cells are filled with digits in advance as  
 289 clues, and players fill in the digits referencing these clues. The difficulty of the Sudoku depends on  
 290 the number of given digits and their placement. We adopt three kinds of datasets: Sudoku, Sudoku-  
 291 hard, and Sudoku-extreme datasets. Sudoku and Sudoku-hard datasets are identical to those defined  
 292 in (Miyato et al., 2025). Sudoku dataset (Wang et al., 2019) consists of 10,000 boards with (31–42)  
 293 given digits. The first 9,000 samples are extracted for training, and the remaining 1,000 samples  
 294 are extracted for validation. On the other hand, the Sudoku-hard dataset (Palm et al., 2018) contains  
 295 (17–34) given digits, which is much fewer than the Sudoku dataset. Sudoku-extreme is introduced in  
 296 (Wang et al., 2025) and is a challenging dataset. It is composed of Sudoku problems from multiple  
 297 datasets. We use 1,000 samples for training and 1,000 samples for testing and apply the same data  
 298 augmentation for Sudoku-extreme datasets as in (Wang et al., 2025).

299 Maze-hard is also a logical puzzle in which the players find the optimal path given  $30 \times 30$  cells  
 300 with passable and impassable cells, a start cell, and a goal cell. We use the same dataset as in (Wang  
 301 et al., 2025), which contains 1,000 samples in both the training and test datasets.

302 For the metric of the model performance, we use the board accuracy, which is defined as the proportion  
 303 of test instances for which the model produces an entirely correct board, i.e., all cells satisfy the  
 304 given constraints without any errors.

### 305 6.2 C-VOTING VS. E-VOTING IN AKORN

307 To compare the performance of C-voting and E-voting, we conduct test-time scaling experiments  
 308 using AKOrN. For the inference, we use C-voting instead of E-voting and run E-voting on the same  
 309 pre-trained model. Note that no modification is needed to integrate C-voting into AKOrN.

310 Figure 2 shows that the board accuracy under C-voting outperforms that of E-voting as the number  
 311 of random samples increases. For Sudoku-hard test in Figure 2(a), we use the Sudoku dataset  
 312 for training and the Sudoku-hard dataset for inference during testing. This experimental setting is  
 313 identical to that in (Miyato et al., 2025). The board accuracy in C-voting with 4096 samples is  
 314  $94.4 \pm 0.1\%$ , which is 4.9% higher than that of E-voting ( $89.5 \pm 2.5\%$ ) reported in (Miyato et al.,  
 315 2025). We also compare the performance with the Sudoku-extreme dataset in Figure 2(b). This  
 316 result indicates that C-voting outperforms E-voting and is useful for models even with an explicit  
 317 energy function. For Maze-hard, AKOrN fails to learn, resulting in a test board accuracy of 0.

### 318 6.3 ITRSA++ WITH C-VOTING

320 To show the performance of ItrSA++ with C-voting, we train ItrSA++ for Sudoku, Sudoku-extreme,  
 321 and Maze-hard datasets and compare the test-time accuracy to HRM and AKOrN.  
 322

323 The results are shown in Figure 3. Even without the voting, ItrSA++ outperforms AKOrN and HRM  
 324 for all the tasks. Note that for the HRM experimental results for Sudoku-extreme and Maze-hard,

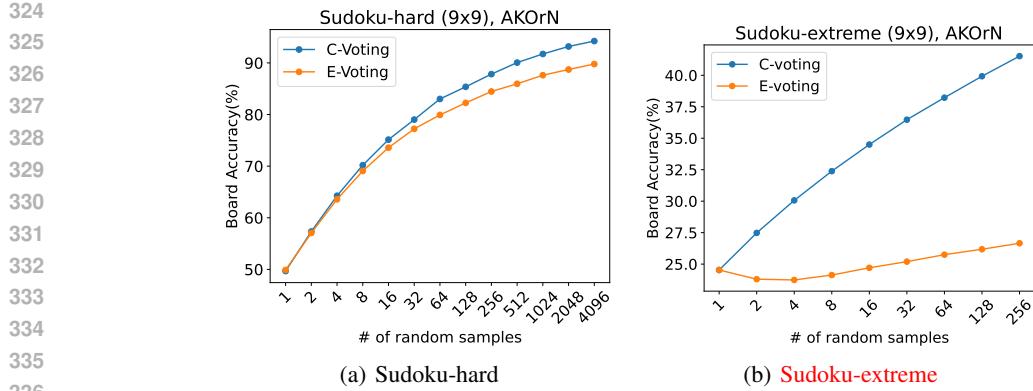


Figure 2: A performance comparison between C-voting and E-voting in AKOrN.

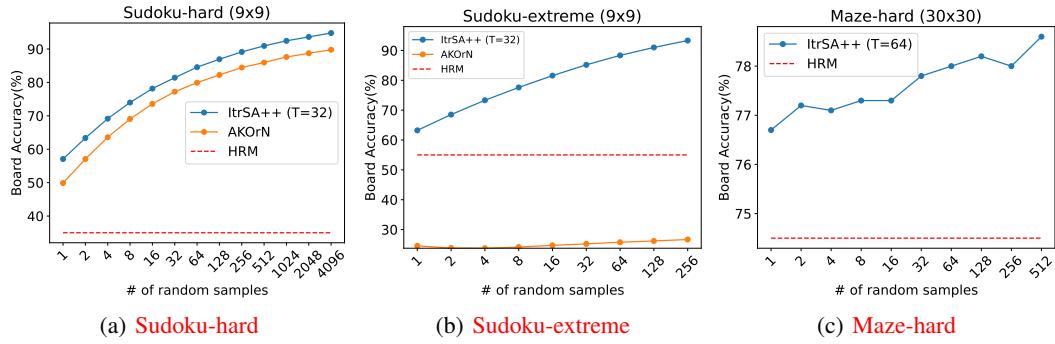


Figure 3: Scaling analysis of ItrSA++ with C-voting.

355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
we utilize those reported in (Wang et al., 2025). On the other hand, for the HRM results on Sudoku-hard, we employ the original HRM code and modify only the dataset. For all of the AKOrN results, we employ the original code (Miyato et al., 2025) and implement datasets for Sudoku-extreme and Maze-hard. As mentioned in subsection 6.2, the test board accuracy of AKOrN for Maze-hard is 0 and not plotted in the figure. ItrSA++ with C-voting scales similarly to AKOrN for the Sudoku tasks when the number of random samples increases, and also exhibits similar scaling for the Sudoku-extreme task. For Maze-hard, the improvement with scaling is weaker compared to Sudoku-hard or Sudoku-extreme, but it nonetheless surpasses HRM and confirms the effectiveness of the voting.

#### 6.4 HRM WITH C-VOTING

378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
To demonstrate the general applicability of C-voting, we combine C-voting with HRM and evaluate the scaling ability on the Sudoku-extreme dataset. While HRM initializes the latent state with random variables, it fixes throughout training and inference. To integrate C-voting with HRM, we modify to use a different initial latent state for each input. We also set the exploration rate for ACT to 1, which means that ACT is not actually used. With these modifications, we train HRM and use C-voting at test-time.

401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
7010  
7011  
7012  
7013  
7014  
7015  
7016  
7017  
7018  
7019  
7020  
7021  
7022  
7023  
7024  
7025  
7026  
7027  
7028  
7029  
7030  
7031  
7032  
7033  
7034  
7035  
7036  
7037  
7038  
7039  
7040  
7041  
7042  
7043  
7044  
7045  
7046  
7047  
7048  
7049  
7050  
7051  
7052  
7053  
7054  
7055  
7056  
7057  
7058  
7059  
7060  
7061  
7062  
7063  
7064  
7065  
7066  
7067  
7068  
7069  
7070  
7071  
7072  
7073  
7074  
7075  
7076  
7077  
7078  
7079  
7080  
7081  
7082  
7083  
7084  
7085  
7086  
7087  
7088  
7089  
7090  
7091  
7092  
7093  
7094  
7095  
7096  
7097  
7098  
7099  
70100  
70101  
70102  
70103  
70104  
70105  
70106  
70107  
70108  
70109  
70110  
70111  
70112  
70113  
70114  
70115  
70116  
70117  
70118  
70119  
70120  
70121  
70122  
70123  
70124  
70125  
70126  
70127  
70128  
70129  
70130  
70131  
70132  
70133  
70134  
70135  
70136  
70137  
70138  
70139  
70140  
70141  
70142  
70143  
70144  
70145  
70146  
70147  
70148  
70149  
70150  
70151  
70152  
70153  
70154  
70155  
70156  
70157  
70158  
70159  
70160  
70161  
70162  
70163  
70164  
70165  
70166  
70167  
70168  
70169  
70170  
70171  
70172  
70173  
70174  
70175  
70176  
70177  
70178  
70179  
70180  
70181  
70182  
70183  
70184  
70185  
70186  
70187  
70188  
70189  
70190  
70191  
70192  
70193  
70194  
70195  
70196  
70197  
70198  
70199  
70200  
70201  
70202  
70203  
70204  
70205  
70206  
70207  
70208  
70209  
70210  
70211  
70212  
70213  
70214  
70215  
70216  
70217  
70218  
70219  
70220  
70221  
70222  
70223  
70224  
70225  
70226  
70227  
70228  
70229  
70230  
70231  
70232  
70233  
70234  
70235  
70236  
70237  
70238  
70239  
70240  
70241  
70242  
70243  
70244  
70245  
70246  
70247  
70248  
70249  
70250  
70251  
70252  
70253  
70254  
70255  
70256  
70257  
70258  
70259  
70260  
70261  
70262  
70263  
70264  
70265  
70266  
70267  
70268  
70269  
70270  
70271  
70272  
70273  
70274  
70275  
70276  
70277  
70278  
70279  
70280  
70281  
70282  
70283  
70284  
70285  
70286  
70287  
70288  
70289  
70290  
70291  
70292  
70293  
70294  
70295  
70296  
70297  
70298  
70299  
70300  
70301  
70302  
70303  
70304  
70305  
70306  
70307  
70308  
70309  
70310  
70311  
70312  
70313  
70314  
70315  
70316  
70317  
70318  
70319  
70320  
70321  
70322  
70323  
70324  
70325  
70326  
70327  
70328  
70329  
70330  
70331  
70332  
70333  
70334  
70335  
70336  
70337  
70338  
70339  
70340  
70341  
70342  
70343  
70344  
70345  
70346  
70347  
70348  
70349  
70350  
70351  
70352  
70353  
70354  
70355  
70356  
70357  
70358  
70359  
70360  
70361  
70362  
70363  
70364  
70365  
70366  
70367  
70368  
70369  
70370  
70371  
70372  
70373  
70374  
70375  
70376  
70377  
70378  
70379  
70380  
70381  
70382  
70383  
70384  
70385  
70386  
70387  
70388  
70389  
70390  
70391  
70392  
70393  
70394  
70395  
70396  
70397  
70398  
70399  
70400  
70401  
70402  
70403  
70404  
70405  
70406  
70407  
70408  
70409  
70410  
70411  
70412  
70413  
70414  
70415  
70416  
70417  
70418  
70419  
70420  
70421  
70422  
70423  
70424  
70425  
70426  
70427  
70428  
70429  
70430  
70431  
70432  
70433  
70434  
70435  
70436  
70437  
70438  
70439  
70440  
70441  
70442  
70443  
70444  
70445  
70446  
70447  
70448  
70449  
70450  
70451  
70452  
70453  
70454  
70455  
70456  
70457  
70458  
70459  
70460  
70461  
70462  
70463  
70464  
70465  
70466  
70467  
70468  
70469  
70470  
70471  
70472  
70473  
70474  
70475  
70476  
70477  
70478  
70479  
70480  
70481  
70482  
70483  
70484  
70485  
70486  
70487  
70488  
70489  
70490  
70491  
70492  
70493  
70494  
70495  
70496  
70497  
70498  
70499  
70500  
70501  
70502  
70503  
70504  
70505  
70506  
70507  
70508  
70509  
70510  
70511  
70512  
70513  
70514  
70515  
70516  
70517  
70518  
70519  
70520  
70521  
70522  
70523  
70524  
70525  
70526  
70527  
70528  
70529  
70530  
70531  
70532  
70533  
70534  
70535  
70536  
70537  
70538  
70539  
70540  
70541  
70542  
70543  
70544  
70545  
70546  
70547  
70548  
70549  
70550  
70551  
70552  
70553  
70554  
70555  
70556  
70557  
70558  
70559  
70560  
70561  
70562  
70563  
70564  
70565  
70566  
70567  
70568  
70569  
70570  
70571  
70572  
70573  
70574  
70575  
70576  
70577  
70578  
70579  
70580  
70581  
70582  
70583  
70584  
70585  
70586  
70587  
70588  
70589  
70590  
70591  
70592  
70593  
70594  
70595  
70596  
70597  
70598  
70599  
70600  
70601  
70602  
70603  
70604  
70605  
70606  
70607  
70608  
70609  
70610  
70611  
70612  
70613  
70614  
70615  
70616  
70617  
70618  
70619  
70620  
70621  
70622  
70623  
70624  
70625  
70626  
70627  
70628  
70629  
70630  
70631  
70632  
70633  
70634  
70635  
70636  
70637  
70638  
70639  
70640  
70641  
70642  
70643  
70644  
70645  
70646  
70647  
70648  
70649  
70650  
70651  
70652  
70653  
70654  
70655  
70656  
70657  
70658  
70659  
70660  
70661  
70662  
70663  
70664  
70665  
70666  
70667  
70668  
70669  
70670  
70671  
70672  
70673  
70674  
70675  
70676  
70677  
70678  
70679  
70680  
70681  
70682  
70683  
70684  
70685  
70686  
70687  
70688  
70689  
70690  
70691  
70692  
70693  
70694  
70695  
70696  
70697  
70698  
70699  
70700  
70701  
70702  
70703  
70704  
70705  
70706  
70707  
70708  
70709  
70710  
70711  
70712  
70713  
70714  
70715  
70716  
70717  
70718  
70719  
70720  
70721  
70722  
70723  
70724  
70725  
70726  
70727  
70728  
70729  
70730  
70731  
70732  
70733  
70734  
70735  
70736  
70737  
70738  
70739  
70740  
70741  
70742  
70743  
70744  
70745  
70746  
70747  
70748  
70749  
70750  
70751  
70752  
70753  
70754  
70755  
70756  
70757  
70758  
70759  
70760  
70761  
70762  
70763  
70764  
70765  
70766  
70767  
70768  
70769  
70770  
70771  
70772  
70773  
70774  
70775  
70776  
70777  
70778  
70779  
70780  
70781  
70782  
70783  
70784  
70785  
70786  
70787  
70788  
70789  
70790  
70791  
70792  
70793  
70794  
70795  
70796  
70797  
70798  
70799  
70800  
70801  
70802  
70803  
70804  
70805  
70806  
70807  
70808  
70809  
70810  
70811  
70812  
70813  
70814  
70815  
70816  
70817  
70818  
70819  
70820  
70821  
70822  
70823  
70824  
70825  
70826  
70827  
70828  
70829  
70830  
70831  
70832  
70833  
70834  
70835  
70836  
70837  
70838  
70839  
70840  
70841  
70842  
70843  
70844  
70845  
70846  
70847  
70848  
70849  
70850  
70851  
70852  
70853  
70854  
70855  
70856  
70857  
70858  
70859  
70860  
70861  
70862  
70863  
70864  
70865  
70866  
70867  
70868  
70869  
70870  
70871  
70872  
70873  
70874  
70875  
70876  
70877  
70878  
70879  
70880  
70881  
70882  
70883  
70884  
70885  
70886  
70887  
70888  
70889  
70890  
70891  
70892  
70893  
70894  
70895  
70896  
70897  
70898  
70899  
70900  
70901  
70902  
70903  
70904  
70905  
70906  
70907  
70908  
70909  
70910  
70911  
70912  
70913  
70914  
70915  
70916  
70917  
70918  
70919  
70920  
70921  
70922  
70923  
70924  
70925  
70926  
70927  
70928  
70929  
70930  
70931  
70932  
70933  
70934  
70935  
70936  
70937  
70938  
70939  
70940  
70941  
70942  
70943  
70944  
70945  
70946  
70947  
70948  
70949  
70950  
70951  
70952  
70953  
70954  
70955  
70956  
70957  
70958  
70959  
70960  
70961  
70962  
70963  
70964  
70965  
70966  
70967  
70968  
70969  
70970  
70971  
70972  
70973  
70974  
70975  
70976  
70977  
70978  
70979  
70980  
70981  
70982  
70983  
70984  
70985  
70986  
70987  
70988  
70989  
70990  
70991  
70992  
70993  
70994  
70995  
70996  
70997  
70998  
70999  
71000  
71001  
71002  
71003  
71004  
71005  
71006  
71007  
71008  
71009  
71010  
71011  
71012  
71013  
71014  
71015  
71016  
71017  
71018  
71019  
71020  
71021  
71022  
71023  
71024  
71025  
71026  
71027  
71028  
71029  
71030  
71031  
71032  
71033  
71034  
71035  
71036  
71037  
71038  
71039  
71040  
71041  
71042  
71043  
71044  
71045  
71046  
71047  
71048  
71049  
71050  
71051  
71052  
71053  
71054  
71055  
71056  
71057  
71058  
71059  
71060  
71061  
71062  
71063  
71064  
71065  
71066  
71067  
71068  
71069  
71070  
71071  
71072  
71073  
71074  
71075  
71076  
71077  
71078  
71079  
71080  
71081  
71082  
71083  
71084  
71085  
71086  
71087  
71088  
71089  
71090  
71091  
71092  
71093  
71094  
71095  
71096  
71097  
71098  
71099  
71100  
71101  
71102  
71103  
71104  
71105  
71106  
71107  
71108  
71109  
71110  
71111  
71112  
71113  
71114  
71115  
71116  
71117  
71118  
71119  
71120  
71121  
71122  
71123  
71124  
71125  
71126  
71127  
71128  
71129  
71130  
71131  
71132  
71133  
71134  
71135  
71136  
71137  
71138  
71139  
71140  
71141  
71142  
71143  
71144  
71145  
71146  
71147  
71148  
71149  
71150  
71151  
71152  
71153  
71154  
71155  
71156  
71157  
71158  
71159  
71160  
71161  
71162  
71163  
71164  
71165  
71166  
71167  
71168  
71169  
71170  
71171  
71172  
71173  
71174  
71175  
71176  
71177  
71178  
71179  
71180  
71181  
71182  
71183  
71184  
71185  
71186  
71187  
71188  
71189  
71190  
71191  
71192  
71193  
71194  
71195  
71196  
71197  
71198  
71199  
71200  
71201  
71202  
71203  
71204  
71205  
71206  
71207  
71208  
71209  
71210  
71

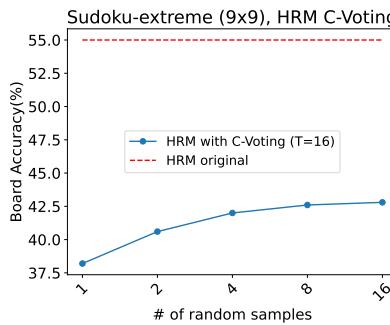


Figure 4: Board accuracy for Sudoku-extreme task of HRM with C-voting.

While the computational complexity of C-voting increases linearly with the number of votes, the computational complexity increase in HRM when increasing the maximum ACT step is expected to be sublinear. Note, however, that our results demonstrate C-voting can be scaled in a manner distinct from scaling in the direction of recurrence and does not focus on computational complexity.

## 6.5 VISUALIZATION OF CONFIDENCE

To analyze the reasons why the performance doesn't improve much on the Maze dataset compared to the Sudoku dataset, where it scales nicely, we visualize the confidence in this section.

First, we show the reliability diagram with several temperatures and the relationship between the expected calibration error (ECE) and average accuracy across all positions with 16 different initial states in Figure 5. When varying the temperature, the ranking of the probability for each position remains unchanged, but the average confidence slightly changes, allowing candidate rankings to fluctuate a little. In fact, calibration does change, with ECE reaching a minimum at  $T=2$ . However, we also observe that the impact of calibration on average accuracy across all positions is limited. In our setting, C-voting relies on the relative ordering of confidence among sampled candidates, rather than the absolute calibration measured by ECE. Temperature scaling changes the absolute confidence values but preserves the top-1 ordering within each cell, explaining why ECE varies while accuracy remains nearly unchanged.

We show the distributions of confidence for 32 samples with 32 different initial states grouped by whether predictions are correct in Figure 6. For the Sudoku-extreme samples, we observe a broader distribution for incorrect samples than for correct samples. This indicates that when problems are difficult, predictions become unstable due to the randomness of the initialization. On the other hand, the lower histogram, which is from the Maze-hard samples, shows that even for the incorrect predictions, the distribution is tight. This suggests that the model possesses incorrect confidence in the Maze-hard samples, regardless of its initial state.

Figure 7 shows the relationship between the number of iteration steps and confidence for 32 random initial states. In all cases, it can be seen that confidence increases as the number of iteration steps increases. For the Sudoku-extreme dataset, the confidence is low when predictions are incorrect. In contrast, this trend is not observed in Maze-hard, which again indicates that the model has incorrect confidence for incorrectly-predicted samples in the Maze-hard dataset.

## 7 DISCUSSION AND CONCLUSION

We propose C-voting, a new test-time voting strategy applicable even when the explicit energy function is not defined. It starts from multiple initial random latent states in models with recurrent structures and selects the best candidate based on confidence.

It provides test-time scaling other than increasing recursion depth. We show that C-voting can be integrated into AKOrN, and find that it outperforms E-voting in subsection 6.2. Although the reason

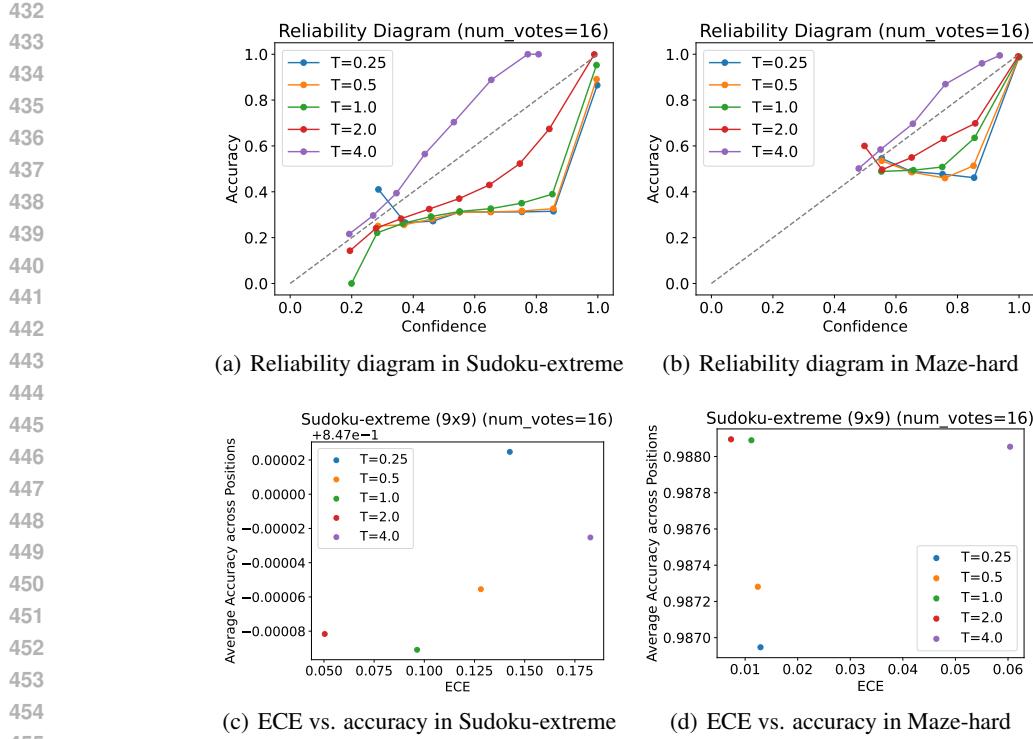


Figure 5: Visualization of calibration of ItrSA++.

for the improvement is not yet clear, at least in the Sudoku task, the confidence may be a more direct proxy for board accuracy than energy.

C-voting achieves state-of-the-art performance by integrating with a new lightweight recurrent model. We introduce ItrSA++ in section 5 as a simple but powerful example of recurrent models for which C-voting can be applied. In subsection 6.3, we demonstrate that ItrSA++ integrated with C-voting outperforms AKOrN in *Sudoku tasks*, and HRM in *Sudoku* and *Maze-hard* tasks.

For models such as AKOrN and ItrSA++, which are trained to operate with random initializations, the sampled candidates evolve into meaningfully different predictions, making C-voting effective. For HRM, however, the original design relies on a fixed initial state shared across all inputs. Introducing randomness at inference time breaks this assumption and can lead to similar or unstable hidden-state evolutions across samples, which fundamentally limits the improvement obtainable from C-voting. Nevertheless, even under this mismatch, we still observe non-trivial gains when the number of samples increases.

As seen in subsection 6.5, performance gains from voting are limited when the model holds incorrect confidence or makes similar predictions regardless of initial random variables. On the other hand, it would also be possible that even if the prediction accuracy of the models cannot be improved, C-voting may still enhance performance by making the models produce more diverse outputs.

Since voting based on confidence corresponds to the maximization of average accuracy across all positions, as shown in equation 14, it would be useful for masked language models (Devlin et al., 2019; Joshi et al., 2020; Li et al., 2022), text infilling (Zhu et al., 2019; Raffel et al., 2020), or image inpainting (Lugmayr et al., 2022; Pathak et al., 2016). Furthermore, using other uncertainty metrics, such as entropy or the sum of log probabilities, may potentially improve performance for specific tasks. We would like to address these as topics for future research.

**Reproducibility Statement** We provide anonymized codes containing training and evaluation scripts for HRM with C-voting, AKOrN with C/E-voting, and ItrSA++. We specify all hyper-parameters in Tables 1–3, including optimizer settings, gradient clipping, EMA, iteration steps T,

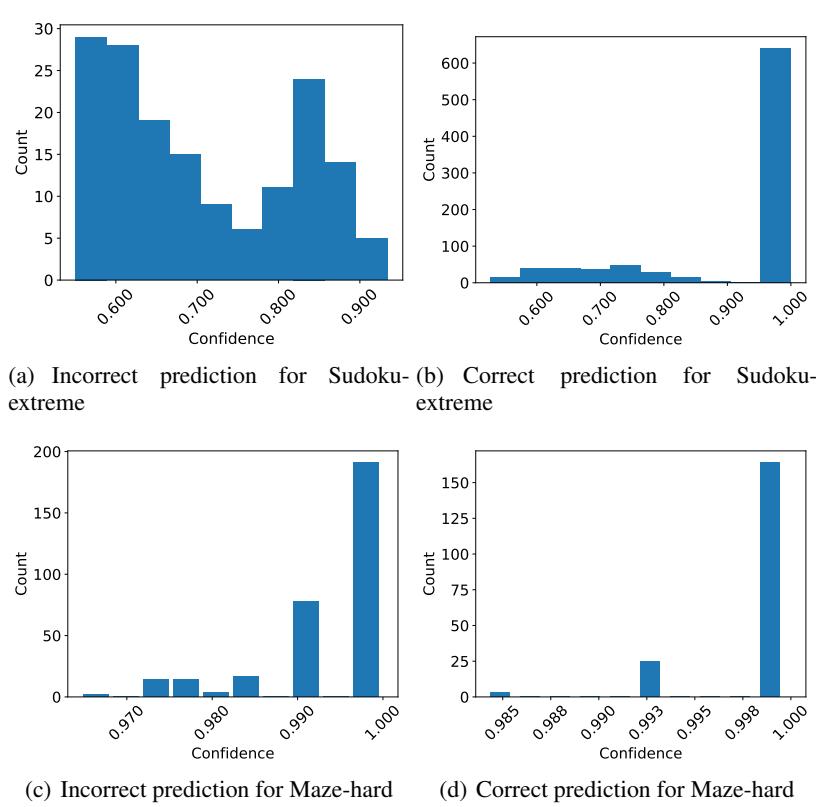
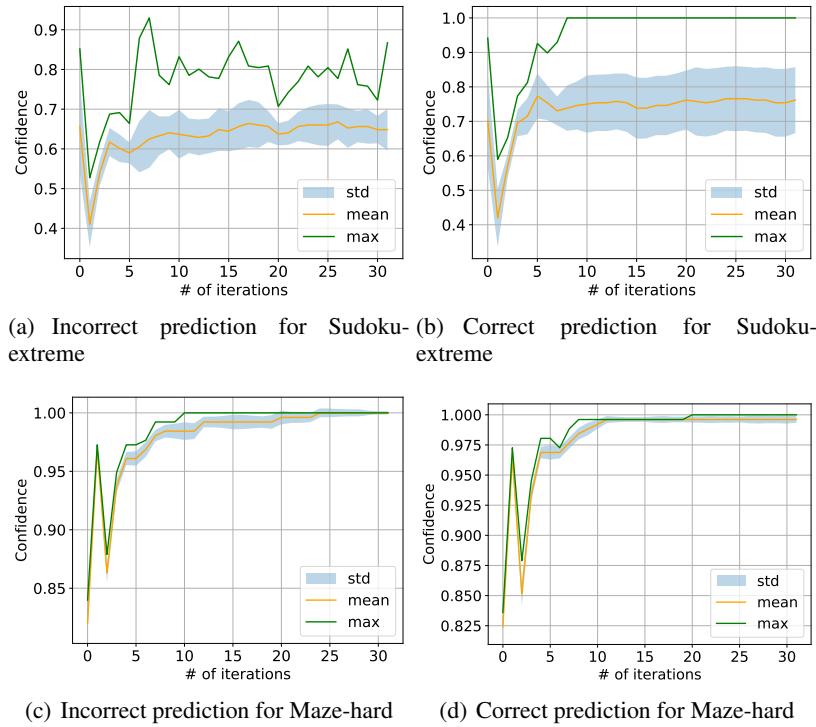
Figure 6: Distributions of confidence of ItrSA++ at  $t = 32$ .

Figure 7: Time step dependency of confidence.

540 and the number of attention heads. We also provide exact data preprocessing and augmentation  
 541 pipelines for Sudoku, Sudoku-hard, Sudoku-extreme, and Maze-hard.  
 542

543 **544 REFERENCES**

545 Cem Anil, Ashwini Pokle, Kaiqu Liang, Johannes Treutlein, Yuhuai Wu, Shaojie Bai, J Zico Kolter,  
 546 and Roger Baker Grosse. Path independent equilibrium models can better exploit test-time com-  
 547 putation. In *Advances in Neural Information Processing Systems*, October 2022.

548 Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Deep equilibrium models. *arXiv [cs.LG]*, September  
 549 2019.

550 François Chollet. On the measure of intelligence. *arXiv preprint arXiv:1911.01547*, 2019.

551 Francois Chollet, Mike Knoop, Gregory Kamradt, Bryan Landers, and Henry Pinkard. Arc-agi-2:  
 552 A new challenge for frontier ai reasoning systems. *arXiv preprint arXiv:2505.11831*, 2025.

553 Luke Darlow, Ciaran Regan, Sebastian Risi, Jeffrey Seely, and Llion Jones. Continuous thought  
 554 machines. *arXiv preprint arXiv:2505.05522*, 2025.

555 Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal  
 556 transformers. In *International Conference on Learning Representations*, 2019. URL <https://openreview.net/forum?id=HyzdRiR9Y7>.

557 Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep  
 558 bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of  
 559 the North American chapter of the association for computational linguistics: human language  
 560 technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

561 Katie Everett, Lechao Xiao, Mitchell Wortsman, Alexander A Alemi, Roman Novak, Peter J Liu,  
 562 Izzeddin Gur, Jascha Sohl-Dickstein, Leslie Pack Kaelbling, Jaehoon Lee, et al. Scaling expo-  
 563 nents across parameterizations and optimizers. *arXiv preprint arXiv:2407.05872*, 2024.

564 Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson,  
 565 Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with  
 566 latent reasoning: A recurrent depth approach. *CoRR*, abs/2502.05171, February 2025. URL  
 567 <https://doi.org/10.48550/arXiv.2502.05171>.

568 Alexi Gladstone, Ganesh Nanduru, Md Mofijul Islam, Peixuan Han, Hyeonjeong Ha, Aman Chadha,  
 569 Yilun Du, Heng Ji, Jundong Li, and Tariq Iqbal. Energy-based transformers are scalable learners  
 570 and thinkers, 2025. URL <https://arxiv.org/abs/2507.02092>.

571 Yunzhe Hu, Difan Zou, and Dong Xu. Hyper-SET: Designing transformers via hyperspherical  
 572 energy minimization. *arXiv [cs.LG]*, 2025.

573 Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira.  
 574 Perceiver: General perception with iterative attention. In *International conference on machine  
 575 learning*, pp. 4651–4664. PMLR, 2021.

576 Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. Span-  
 577 bert: Improving pre-training by representing and predicting spans. *Transactions of the association  
 578 for computational linguistics*, 8:64–77, 2020.

579 Tero Karras, Miika Aittala, Jaakko Lehtinen, Janne Hellsten, Timo Aila, and Samuli Laine. An-  
 580 alyzing and improving the training dynamics of diffusion models. *ArXiv*, abs/2312.02696, 2023.  
 581 URL <https://api.semanticscholar.org/CorpusID:265659032>.

582 Yoshiki Kuramoto. Self-entrainment of a population of coupled non-linear oscillators. In *Inter-  
 583 national Symposium on Mathematical Problems in Theoretical Physics*, pp. 420–422. Springer-  
 584 Verlag, Berlin/Heidelberg, 1975.

594 Hojoon Lee, Hyeonseo Cho, Hyunseung Kim, Donghu Kim, Dugki Min, Jaegul Choo, and  
 595 Clare Lyle. Slow and steady wins the race: Maintaining plasticity with hare and tortoise  
 596 networks. *ArXiv*, abs/2406.02596, 2024. URL <https://api.semanticscholar.org/CorpusID:270258586>.

598 Siyuan Li, Zicheng Liu, Juanxi Tian, Ge Wang, Zedong Wang, Weiyang Jin, Di Wu, Cheng Tan,  
 599 Tao Lin, Yang Liu, Baigui Sun, and Stan Z. Li. Switch ema: A free lunch for better flatness and  
 600 sharpness. *ArXiv*, abs/2402.09240, 2024. URL <https://api.semanticscholar.org/CorpusID:267657558>.

603 Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsumi B Hashimoto. Diffusion-  
 604 lm improves controllable text generation. *Advances in neural information processing systems*, 35:  
 605 4328–4343, 2022.

606 Max Lipton, Renato Mirollo, and Steven H Strogatz. The kuramoto model on a sphere: Explaining  
 607 its low-dimensional dynamics with group theory and hyperbolic geometry. *arXiv [math.DS]*, July  
 608 2019.

609 Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool.  
 610 Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the  
 611 IEEE/CVF conference on computer vision and pattern recognition*, pp. 11461–11471, 2022.

612 Takeru Miyato, Bernhard Jaeger, Max Welling, and Andreas Geiger. GTA: A geometry-aware attention  
 613 mechanism for multi-view transformers. In *The Twelfth International Conference on Learning  
 614 Representations*, 2024. URL <https://openreview.net/forum?id=uJVHygNeSZ>.

615 Takeru Miyato, Sindy Löwe, Andreas Geiger, and Max Welling. Artificial kuramoto oscillatory  
 616 neurons. In *The Thirteenth International Conference on Learning Representations*, 2025. URL  
 617 <https://openreview.net/forum?id=nwDRD4AMoN>.

618 Rasmus Palm, Ulrich Paquet, and Ole Winther. Recurrent relational networks. *Advances in neural  
 619 information processing systems*, 31, 2018.

620 Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context  
 621 encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer  
 622 vision and pattern recognition*, pp. 2536–2544, 2016.

623 Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi  
 624 Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text  
 625 transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

626 Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J Reddi. Reasoning  
 627 with latent thoughts: On the power of looped transformers. *arXiv preprint arXiv:2502.17416*,  
 628 2025.

629 Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

630 ARC Prize Team. The hidden drivers of hrm’s performance on arc-agi, 2025. URL <https://arcprize.org/blog/hrm-analysis>.

631 Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez,  
 632 Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information  
 633 processing systems*, 30, 2017.

634 Guan Wang, Jin Li, Yuhao Sun, Xing Chen, Changling Liu, Yue Wu, Meng Lu, Sen Song, and  
 635 Yasin Abbasi Yadkori. Hierarchical reasoning model. 2025. URL <https://arxiv.org/abs/2506.21734>.

636 Po-Wei Wang, Priya Donti, Bryan Wilder, and Zico Kolter. Satnet: Bridging deep learning and log-  
 637 ical reasoning using a differentiable satisfiability solver. In *International Conference on Machine  
 638 Learning*, pp. 6545–6554. PMLR, 2019.

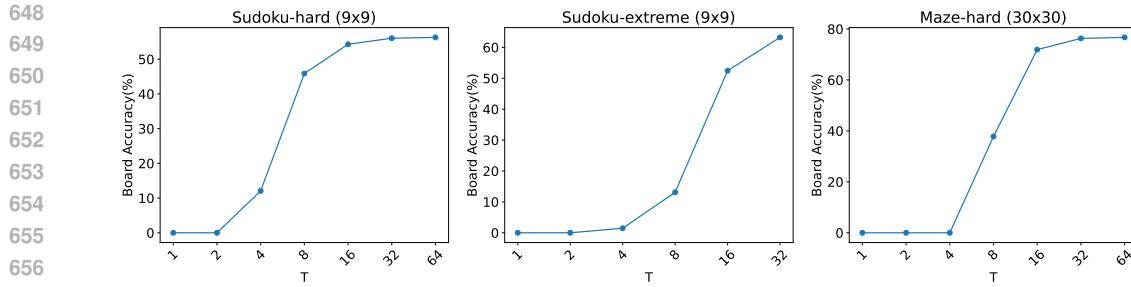


Figure 8: Test-time scaling in ItrSA++

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL <https://openreview.net/forum?id=1PL1NIMMrw>.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32, 2019.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*, 2022.

Wanrong Zhu, Zhiting Hu, and Eric Xing. Text infilling. *arXiv preprint arXiv:1901.00158*, 2019.

## A TEST-TIME SCALING IN ITRSA++

ItrSA++ has recursive structures similar to models such as the recurrent transformer (Geiping et al., 2025; Jaegle et al., 2021), and test-time scaling can be observed. Figure 8 demonstrates that for Sudoku-hard, Sudoku-extreme, and Maze-hard tasks, board accuracy increases as the number of iterative steps grows in ItrSA++.

## B TRANSFORMER WITH C-VOTING

To validate the performance improvement by C-voting for other than recurrent models, we also integrate it into a transformer and conduct experiments. Training is performed using the Sudoku dataset, and testing is performed using the Sudoku-hard dataset. This is the same problem setting as subsection 6.2. Unlike a standard transformer, it requires a random initial latent state to use C-voting. Therefore, we mix the input  $x$  and the initial latent state  $z$  using a linear layer before feeding them into the transformer.

Figure 9 shows the dependence of board accuracy on the number of random samples for initial states. Since the transformer is not performing well on the Sudoku-hard task, performance improvements through C-voting are also limited.

## C EXPERIMENTAL DETAILS

In this section, we provide an overview of the experimental details. We apply exponential moving average (Karras et al., 2023; Lee et al., 2024; Li et al., 2024) for the model parameters of ItrSA++ and AKOrN. We adopt gradient truncation (Geiping et al., 2025) for ItrSA++. More precisely, during the training, we detach the gradient of the latent state  $z_t$  at  $t = 2$  for Sudoku, at  $t = 14$  for

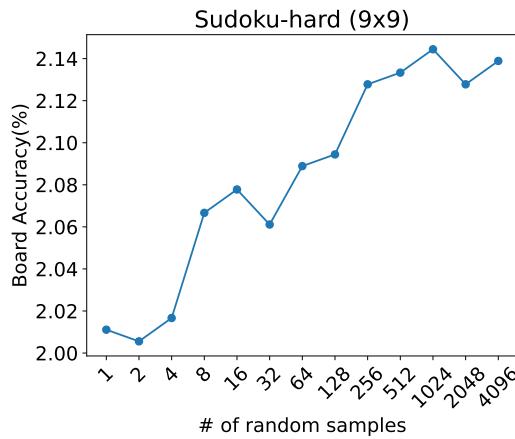


Figure 9: Board accuracy of a transformer with C-voting.

Parameter	Value
Optimizer	AdamW
$\beta$ for Adam	(0.9, 0.95)
Weight decay	0.01
Gradient clipping threshold	1.0
Learning rate	$5 \times 10^{-4}$
Batch size	64
EMA decay rate	0.995
# of heads	12
Embedding dims.	384
# of repetitions of Self-Attn.	4

Table 1: Hyperparameters for ItrSA++.

Sudoku-extreme, and at  $t = 28$  for Maze. We show the hyperparameters for ItrSA++ in Table 1, for HRM in Table 2, and for AKOrN in Table 3. The number of parameters is  $\approx 3$  million for AKOrN and ItrSA++, and  $\approx 27$  million for HRM with these settings.

Parameter	Value
Optimizer	Adam-atan2 (Everett et al., 2024)
$\beta$ for Adam	(0.9, 0.95)
Weight decay	1.0
Gradient clipping threshold	1.0
Learning rate	$1 \times 10^{-4}$
Warm-up steps	2000
Batch size	768
# of heads	8
Embedding dimension	512
Epochs	20000
# of H layers	4
# of L layers	4
Halt exploration prob.	1.0

Table 2: Hyperparameters for modified HRM.

Parameter	Value
Optimizer	Adam
$\beta$ for Adam	(0.9, 0.999)
Gradient clipping threshold	1.0
Learning rate	$5 \times 10^{-4}$
Batch size	100
EMA decay rate	0.995
# of heads	8
Embedding dimension	512
Epochs	100
# of iteration steps	16
# of Kuramoto layers	1
Oscillator dims.	4
Step size	1.0
Connectivity	Attention

Table 3: Hyperparameters for AKOrN.

Variant	Board acc. in test
w/o GT	0.0%
w/o EMA	0.0%
Cross attn. to linear	59.2%
SWiGLU to FFN	62.0%
ItrSA++(Proposed)	<b>63.2%</b>

Table 4: Ablation study for ItrSA++

## D ABLATION STUDY OF ITRSA++

To clarify the contribution of each architectural and inference component of ItrSA++, we conduct ablation experiments on Sudoku-extreme. Table 4 summarizes the results.

Overall, the ablations demonstrate that (1) ItrSA++ requires EMA and GT for stable training, (2) the cross-attention block provides meaningful improvements in reasoning ability.

We further examine how the effectiveness of voting changes when using metrics other than top-1 probability  $\hat{P}_l(\mathbf{z}_{i,T}^{(k)})$  in Equation 9. Specifically, we use negative entropy (NE) and the sum of log probability (LP) defined below.

$$\hat{P}_l^{(\text{NE})}(\mathbf{z}_{i,T}^{(k)}) = \sum_j P_{j,l}(\mathbf{z}_{i,T}^{(k)}) \log P_{j,l}(\mathbf{z}_{i,T}^{(k)}) \quad (20)$$

$$\hat{P}_l^{(\text{LP})}(\mathbf{z}_{i,T}^{(k)}) = \max_j \log P_{j,l}(\mathbf{z}_{i,T}^{(k)}) \quad (21)$$

The results with ItrSA++ are shown in Figure 10. For Sudoku-extreme, almost no difference is observed, and even in Maze-hard, the difference is only about 0.1%. This is thought to be because when the top-1 probability is dominant, there is little difference in ranking across metrics. On the other hand, since it facilitates analyses such as Equation 14, we adopt the top-1 probability.

## E PERFORMANCE DEPENDENCY ON RANDOM SEEDS

Our method uses random variables, so performance may vary when different random seeds are used. To verify this, we perform inference on ItrSA++ with C-voting using multiple random seeds. Figure 11 shows the box plots of the results obtained using five different seeds. Compared with Sudoku-extreme, Maze-hard exhibits greater seed dependence, though it remains at around 1%. However, looking at the median, board accuracy increases almost monotonically, indicating that the overall results are not significantly affected.

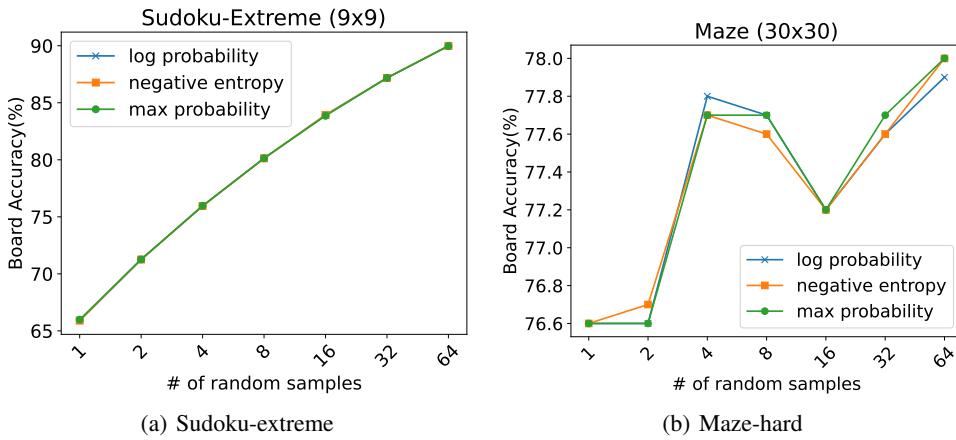


Figure 10: Comparison of voting effects across different metrics.

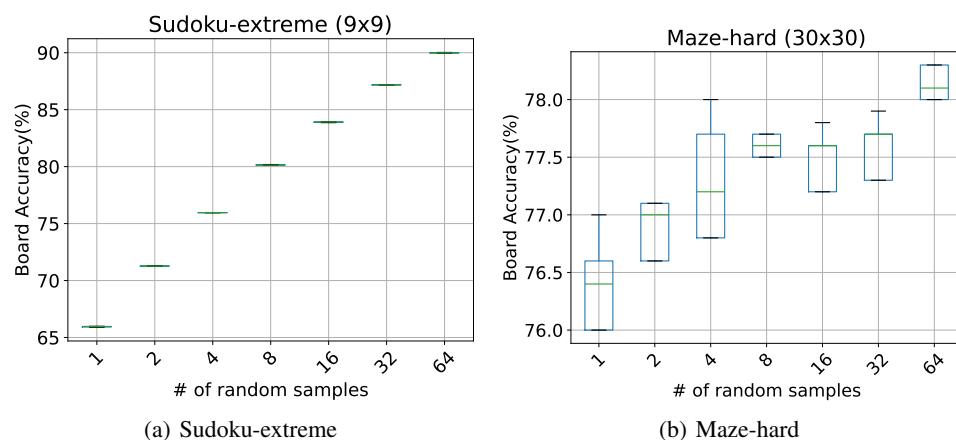


Figure 11: Random seed dependency of board accuracy.

864    **F USE OF LARGE LANGUAGE MODELS**  
865

866    We use large language models (LLMs) to refine writing for this paper and partially use them for  
867    code generation of experiments. We also use LLMs to help explore related works.  
868

869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917