BEYOND AUTOREGRESSION: DISCRETE DIFFUSION FOR COMPLEX REASONING AND PLANNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Autoregressive language models, despite their impressive capabilities, struggle with complex reasoning and long-term planning tasks. We introduce discrete diffusion models as a novel solution to these challenges. Through the lens of subgoal imbalance, we demonstrate how diffusion models effectively learn difficult subgoals that elude autoregressive approaches. We propose Multi-granularity Diffusion Modeling (MDM), which prioritizes subgoals based on difficulty during learning. On complex tasks like Countdown, Sudoku, and Boolean Satisfiability Problems, MDM significantly outperforms autoregressive models without using search techniques. For instance, MDM achieves 91.5% and 100% accuracy on Countdown and Sudoku, respectively, compared to 45.8% and 20.7% for autoregressive models. Our work highlights the potential of diffusion-based approaches in advancing AI capabilities for sophisticated language understanding and problem-solving tasks.

1 INTRODUCTION

In recent years, autoregressive language models (LMs; Bengio et al. 2000) have dominated the landscape of natural language processing and artificial intelligence. Empowered by scaling laws (Kaplan et al., 2020), these models have demonstrated impressive performance across various applications (OpenAI, 2022; Achiam et al., 2023; Anthropic, 2023; Team et al., 2023, *inter alia*). However, this apparent success masks significant limitations that are becoming increasingly evident. Autoregressive models inherently struggle with tasks requiring complex reasoning, long-term planning, and maintaining global coherence (Bubeck et al., 2023; Valmeekam et al., 2023; 2024; Dziri et al., 2024; Kambhampati et al., 2024). These shortcomings represent substantial challenges in developing AI systems capable of robust problem-solving and adaptable cognition (Wu et al., 2022; Zhao et al., 2023; Trinh et al., 2024; Yao et al., 2023; Shinn et al., 2024, *inter alia*). While autoregressive approaches have driven considerable progress, their limitations suggest that they may not be the optimal solution for all aspects of machine intelligence. As the field evolves, it becomes increasingly important to explore alternative paradigms that can address these inherent drawbacks and potentially offer new avenues for advancement in AI capabilities.

In response to these limitations, recent research has focused on addressing the inherent constraints of autoregressive models. Various strategies have been explored, including the integration of search algorithms at inference (Yao et al., 2024; Besta et al., 2024) and the incorporation of backtracking supervision during training (Lehnert et al., 2024; Gandhi et al., 2024). However, these approaches are not without their own drawbacks: the former often incurs significant computational costs, while the latter frequently results in verbose inputs and suboptimal performance.

To address this challenge, we argue for a fundamentally different modeling approach: discrete diffusion models. While most contemporary language models are autoregressive, diffusion-based models have become predominant in image (Dhariwal & Nichol, 2021; Rombach et al., 2022; Peebles & Xie, 2023) and video domains (Ho et al., 2022; Wu et al., 2023a; Brooks et al., 2024). Diffusion models are also gaining traction in various other applications, such as protein desiging (Xu et al., 2022; Hoogeboom et al., 2022b; Corso et al., 2023) and planning in reinforcement learning (Janner et al., 2022; Ajay et al., 2022; Chi et al., 2023). In this work, we reveal that discrete diffusion models demonstrate significantly superior performance compared to the autoregressive counterparts, particularly in tasks requiring complex planning and reasoning.

To substantiate this argument, we first examine the problem through the lens of *subgoal imbalance* (§3.1). We present both theoretical and empirical evidence via a synthetic planning task (Figure 1) to illustrate why autoregressive models struggle with these types of problems, often achieving near-random performance. In contrast, we demonstrate how diffusion models effectively learn the subgoals that challenge autoregressive models (§3.2). The key insight lies in the training objective of diffusion models, where difficult subgoals are decomposed into a diverse range of interrelated views within a multi-view learning framework (Xu et al., 2013). Each of these views is more manageable, resulting in an overall easier and more effective learning process.

Building upon these insights, we propose a natural extension to current discrete diffusion models, which we term multi-granularity diffusion modeling (MDM; §3.3). This approach prioritizes different subgoals based on their difficulty during the learning process, leading to more effective learning outcomes and faster convergence.

In our experimental evaluation (§4), we focus on substantially more complex problem-solving tasks, such as Countdown (Gandhi et al., 2024) and Sudoku (Garns, 1979). These tasks demand extensive planning over a large number of combinations and pose challenges even for commercial Large Language Models (e.g., GPT-4 Achiam et al. 2023). Notably, without employing any search techniques, MDM achieves 91.5% and 100% accuracy on Countdown and Sudoku respectively, while its autoregressive counterpart only solves 45.8% and 20.7% of the problems. Additionally, we conduct experiments on the Boolean Satisfiability Problem (SAT), an NP-complete problem (Cook, 1971) that represents a wide range of constraint satisfaction problems. Our model exhibits superior performance in solving SAT problems with higher accuracy compared to the autoregressive alternative, particularly when dealing with an increased number of variables and constraints. Through this systematic exploration, we aim to demonstrate the potential advantages of diffusion-based approaches in addressing sophisticated language understanding and generation challenges. All associated code is available at Anonymous.

2 Background

2.1 Auto-regressive Modeling

Let $\mathbf{x} \coloneqq (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$ denote a sequence drawn from a data distribution $q(\mathbf{x})$. For decades, it has been common to factorize the joint probabilities of a sequence of tokens as the product of conditional probabilities (Jelinek, 1980; Bengio et al., 2000):

$$p_{\boldsymbol{\theta}}(\mathbf{x}) = p_{\boldsymbol{\theta}}(\boldsymbol{x}_1) \prod_{n=2}^{N} p_{\boldsymbol{\theta}}(\boldsymbol{x}_n \mid \boldsymbol{x}_{1:n-1}), \tag{1}$$

where $\boldsymbol{\theta}$ parameterizes the model distribution and $\boldsymbol{x}_{1:n-1} \coloneqq \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n-1}$. In order to optimize the generative model $p_{\boldsymbol{\theta}}(\mathbf{x})$ to fit the data distribution $q(\mathbf{x})$, we optimize the negative log-likelihood:

$$L_{\text{AR}} = -\mathbb{E}_{q(\mathbf{x})} \log p_{\boldsymbol{\theta}}(\mathbf{x}) = -\mathbb{E}_{q(\mathbf{x})} \sum_{n=1}^{N} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_n \mid \boldsymbol{x}_{1:n-1}). \tag{2}$$

2.2 Discrete Diffusion Modeling

Discrete diffusion models (Sohl-Dickstein et al., 2015; Hoogeboom et al., 2021; Austin et al., 2021) are a class of latent variable models characterized by a forward noising process and a learned reverse denoising process. The forward process $q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$ corrupts the original data $\mathbf{x}_0 \coloneqq \mathbf{x}$ into a sequence of increasingly noisy latent variables $\mathbf{x}_{1:T} \coloneqq \mathbf{x}_1, \ldots, \mathbf{x}_T$. The backward process learns to gradually denoise the latent variables to the data distribution given by:

$$p_{\boldsymbol{\theta}}(\mathbf{x}) = \sum_{\mathbf{x}_{1:T} \sim q} p(\mathbf{x}_T) \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t). \tag{3}$$

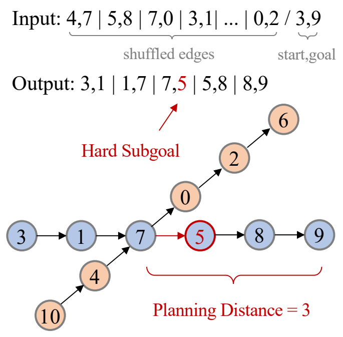
Input: 4,7 | 5,8 | 7,0 | 3,1| ... | 0,2 / 3,9
shuffled edges
start,goal
Output: 3,1 | 1,7 | 7,5 | 5,8 | 8,9
Hard Subgoal
Planning Distance = 3

Figure 1: The planning task.

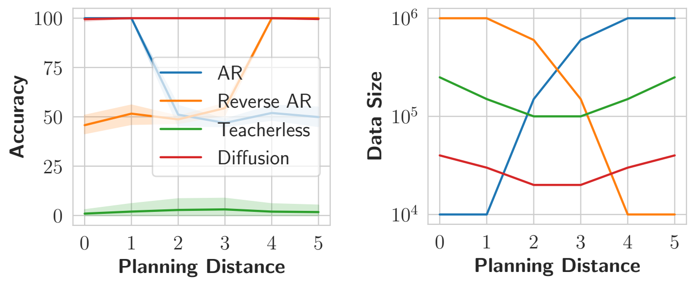

Figure 2: **(Left)** Accuracy of different method given 50k training data. **(Right)** Minimum data size required to solve (i.e., accuracy above 90%) subgoal at each planning distance.

Due to the intractable marginalization, we typically optimize a variational upper bound on the negative log-likelihood:

$$L_{\text{DM}} = \mathbb{E}_{q(\mathbf{x}_0)}\Bigg[\underbrace{D_{\text{KL}}[q(\mathbf{x}_T|\mathbf{x}_0)||p(\mathbf{x}_T)]}_{L_T} + \sum_{t=2}^{T}\underbrace{\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}\left[D_{\text{KL}}[q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)||p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t)]\right]}_{L_{t-1}}$$
$$\underbrace{-\mathbb{E}_{q(\mathbf{x}_1|\mathbf{x}_0)}[\log p_{\boldsymbol{\theta}}(\mathbf{x}_0|\mathbf{x}_1)]}_{L_0}\Bigg], \tag{4}$$

where L_T is a constant when one uses a fixed prior $p(\mathbf{x}_T)$. By defining both the forward and backward distribution as categorical distribution, e.g., $q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \text{Cat}(\mathbf{x}_t; \boldsymbol{p} = \boldsymbol{Q}_t^\top \mathbf{x}_{t-1})$ where $\boldsymbol{Q}_t$ is a pre-defined $K \times K$ transition matrix and K is the size of categories, and $p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t) = q(\mathbf{x}_{t-1}|\mathbf{x}_t, f(\mathbf{x}_t;\boldsymbol{\theta}))$, the forward process posterior $q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)$ and each KL term can be calculated analytically (Hoogeboom et al., 2021; Austin et al., 2021).

3 Subgoal Imbalance and Multi-granularity Diffusion Models

In this section, we employ a motivation example (§3.1) to elucidate the challenges faced by autoregressive models in specific scenarios. Through this analysis, we introduce the concept of *subgoal imbalance*—wherein some subgoals are inherently more difficult than others—which offers insights into these difficulties. We then extend our discussion in §3.2 to examine how diffusion models can more effectively address and learn these *hard subgoals*, effectively overcoming the limitations observed in autoregressive approaches. We finally propose Multi-granularity Diffusion Modeling (MDM; §3.3) as a natural extension of discrete diffusion models to better address these challenges and improve performance on complex tasks requiring planning and reasoning.

3.1 Subgoal Imbalance in Autoregressive and Diffusion Modeling

We designed a simple planning task to serve as our running example. Consider the example in Figure 1, where the input for the task consists of a set of shuffled edges from the graph shown below. At the end of the input sequence, the start and goal nodes are specified to indicate the path the model needs to find. The objective of this task is to identify the correct path in the graph and output its constituent edges. The complexity of this problem arises from distracting factors (highlighted in orange) that potentially mislead the path selection. For instance, at node 7, with the goal being node 9, the model must plan over a distance of 3 nodes to determine that the correct next choice should be node 5 rather than 0. We define this span as the Planning Distance (PD), a parameter adjustable in our synthetic task data. Intuitively, as the PD increases, the model faces greater difficulty in learning to determine the correct subsequent node. We formalize this intuition as *subgoal imbalance*.

Proposition 1 ***(Subgoal imbalance due to the unknown data distribution*** $q(\mathbf{x})$***)*** *Given the true data distribution $q(\mathbf{x})$ is usually unknown, the difficulty of learning each subgoal $\boldsymbol{x}_n$ can differ significantly based on how we parametrize the model distribution, and some subgoals may require substantially more data to learn or may even be infeasible to learn.*

Subgoal imbalance in autoregressive modeling. The autoregressive modeling parametrizes the model distribution $p_{\boldsymbol{\theta}}(\mathbf{x})$ into $p_{\boldsymbol{\theta}}(\boldsymbol{x}_1)\prod_{n=2}^{N} p_{\boldsymbol{\theta}}(\boldsymbol{x}_n \mid \boldsymbol{x}_{1:n-1})$. Given the true data distribution $q(\mathbf{x})$ is unknown, the generation of individual tokens may not inherently follow an autoregressive pattern (i.e., $\boldsymbol{x}_n \not\sim q(\boldsymbol{x}_n \mid \boldsymbol{x}_{1:n-1})$). Consequently, the difficulty of learning these subgoals can differ significantly. Given only the left context, some subgoals may require substantially more data to learn or may even be infeasible to learn.

Setup. We synthesize the data with only one distracting path. We randomize node numbers in $[0, 10]$ and the intersection positions in $[0, 5]$. We further designed this task to be symmetric, ensuring that simply training with reversed output, as suggested by Bachmann & Nagarajan (2024), cannot solve subgoals with all PDs. For comparison, we include Auto-regressive (AR), reverse AR (Bachmann & Nagarajan, 2024), and teacherless training (Monea et al., 2023; Bachmann & Nagarajan, 2024), which can be seen as a lookahead method that produce all target tokens from the source input, and our proposed diffusion model (detailed in §3.2). For all the models, we keep the model architecture fixed as the same 3-layer Transformer with approximately 6M parameters. More details can be found in Appendix §C.

Discussion. We examine the performance of all the models in two scenarios. In the first scenario, we generate a fixed number of 50k instances with mixed planning distance. We plot the accuracy on the held-out evaluation set for each model in the left figure of Figure 2. Our findings indicate that autoregressive models (AR and Reverse AR) are only effective in solving cases where the PD equals 0 or 1 (or equivalently, 5 and 4 in the reverse setting). Due to the aforementioned subgoal imbalance phenomenon, when PD is less than 2, the task barely involves any planning, allowing models to simply copy from the input with ease. However, for larger PDs, AR models barely outperform random guessing. Teacherless training fails to adequately fit the training data, resulting in the production of illegal paths. In contrast, our diffusion model achieves perfect accuracy across all PD values.

In the second scenario, we investigate whether the challenging subgoals can be naturally resolved through data or model scaling, akin to the success observed in large language models (Kaplan et al., 2020; Wei et al., 2022a). To investigate this question, we gradually increase the size of the dataset for each model with different PDs and plot the minimum data size required to solve the subgoal in the right figure of Figure 2. We find that the autoregressive models (AR and Reverse AR) can learn the easy cases of PD equal to 0 and 1 (or equivalently, 5 and 4 in the reverse setting) with only 10k data points. However, exponentially larger amounts of data are required to address increasingly challenging subgoals. Both teacherless training and diffusion models exhibit a similar U-shaped curve in their performance. This similarity can be attributed to the fact that teacherless training can be conceptualized as a special case of diffusion without an iterative noising and denoising process. In these models, solving edge PDs necessitates slightly more data. We hypothesize that this phenomenon occurs because the distance to other positions is shorter from the middle position (i.e., higher closeness centrality), thus providing the middle position with more nearby tokens to aid in prediction. Overall, autoregressive models require significantly more data to address all PDs compared to diffusion models, highlighting their relative data inefficiency.

In addition to our previous experiments, we conducted a series of tests to examine the effect of increasing the parameter count in autoregressive models while maintaining a fixed dataset size of 50,000 instances. Our findings reveal that scaling the original 6 million parameter model to 85 million, 303 million, and 1.5 billion parameters fails to resolve all PDs. Only upon fine-tuning a substantially larger model, specifically the LLaMA 7B model (Touvron et al., 2023), did we observe successful resolution of all PD subgoals.

3.2 Effective Hard Subgoal Learning in Diffusion Modeling

These experiments collectively indicate that diffusion models are significantly more effective in learning challenging subgoals arising from subgoal imbalance. To elucidate why diffusion models exhibit this superior capability, we first establish a connection between autoregressive (AR) models and diffusion models by reformulating Equation (4). Instead of evaluating the KL divergence between two complicated categoricals (Hoogeboom et al., 2021), we consider discrete diffusion with absorbing state and simplify it as the weighted cross-entropy losses (Austin et al., 2021; Zheng et al., 2023; Shi et al., 2024; Sahoo et al., 2024):

$$D_{\text{KL}}[q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)||p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t) = -w(t)\sum_{n=1}^{N}\mathbf{1}_{\mathbf{x}_{t,\text{n}}\neq\mathbf{x}_{0,\text{n}}}\mathbf{x}_{0,n}^{\top}\log f(\mathbf{x}_t;\boldsymbol{\theta})_n, \tag{5}$$

where $w(t) = \frac{\alpha_{t-1}-\alpha_t}{1-\alpha_t} \in (0,1]$ is a time-dependent reweighting term which places higher weight when t approaching 0. We then rewrite Equation (4) as:

$$L_{\text{DM}} = \mathbb{E}_{q(\mathbf{x}_0)}\sum_{n=1}^{N}\underbrace{\sum_{t=1}^{T}w(t)\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}u(\mathbf{x}_0,\mathbf{x}_t,n;\boldsymbol{\theta})}_{-\log p_{\text{DM}}(\boldsymbol{x}_n|\boldsymbol{x}_{\neq n})}, \tag{6}$$

where $u(\mathbf{x}_0,\mathbf{x}_t,n;\boldsymbol{\theta}) := -\mathbf{1}_{\mathbf{x}_{t,\text{n}}\neq\mathbf{x}_{0,\text{n}}}\mathbf{x}_{0,n}^{\top}\log f(\mathbf{x}_t;\boldsymbol{\theta})_n$ is the cross entropy loss on token n.

We can now systematically compare the losses of autoregressive (AR) and diffusion models (DM), specifically $-\log p_{\text{AR}}(\boldsymbol{x}_n \mid \boldsymbol{x}_{1:n-1})$ and $-\log p_{\text{DM}}(\boldsymbol{x}_n \mid \boldsymbol{x}_{\neq n})$, as expressed in Equations (2) and (6), respectively. In Figure 3, we examine a specific hard subgoal with Planning Distance (PD) equals 3 in both model types. The loss levels of AR and diffusion models are depicted using blue and red lines, respectively. The overall loss $-\log p_{\text{DM}}(\boldsymbol{x}_n \mid \boldsymbol{x}_{\neq n})$ in the diffusion model remains relatively low compared to its autoregressive counterpart $-\log p_{\text{AR}}(\boldsymbol{x}_n \mid \boldsymbol{x}_{1:n-1})$, corroborating the superior performance of the diffusion model on these challenging subgoals in our experiments.

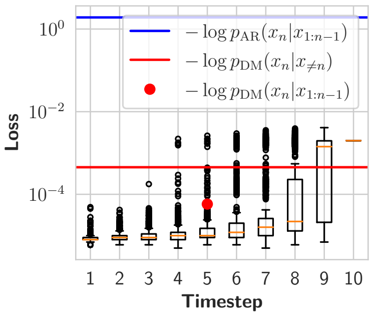

Figure 3: Loss for a specific hard subgoal, i.e., PD=3, in Diffusion and AR modeling. We also show the unweighted loss $u(\mathbf{x}_0,\mathbf{x}_t,n;\boldsymbol{\theta})$ at different timestep t and context $\mathbf{x}_t$ in diffusion modeling.

Further analysis of the unweighted loss $u(\mathbf{x}_0,\mathbf{x}_t,n;\boldsymbol{\theta})$ in the diffusion model, based on 1,000 samples of $\mathbf{x}_t \sim q(\mathbf{x}_t|\mathbf{x}_0)$, reveals a clear trend: as the number of timesteps increases, resulting in more noise in $\mathbf{x}_t$, objectives in smaller timesteps (i.e., recovery from less noisy data) become significantly easier to learn. From a multi-view learning perspective (Xu et al., 2013), each $\mathbf{x}_t$ can be interpreted as a distinct view of $\mathbf{x}_0$, where each view provides different information about $\mathbf{x}_0$. In the diffusion process, by exploring the consistency and complementary properties of different views offered by a diverse range of interrelated objectives $u(\mathbf{x}_0,\mathbf{x}_t,n;\boldsymbol{\theta})$, our findings suggest that objectives challenging to learn in AR models become more effective, promising, and exhibit better generalization in diffusion models.

This phenomenon is particularly evident when examining scenarios where mask noise is applied to positions after the hard token, i.e., $\mathbf{x}_t = \boldsymbol{x}_{1:n-1}$, where the diffusion model learns the hard subgoal similarly to AR models. We plot this loss as $-\log p_{\text{DM}}(\boldsymbol{x}_n \mid \boldsymbol{x}_{1:n-1})$ in the figure. Unlike in the AR model, where this learning is consistently difficult, in diffusion models, this challenging subgoal is addressed at a much more manageable level during the learning process.

3.3 Multi-granularity Diffusion Modeling

These observations provide valuable insights, i.e., diffusion modeling builds on a diverse range of interrelated views from the data $\mathbf{x}_0$ to handle a challenging subgoal. To handle multiple challenging subgoals in real data, we should prioritize different subgoals based on their difficulty during the learning process to achieve more effective learning outcomes and faster convergence, and this naturally translates to prioritizing difficult views as the learning of a subgoal depends on learning interrelated views related to it. Building on this, we propose the multi-granularity diffusion model as a natural extension of the discrete diffusion model.

In practice, to optimize Equation (6), we typically employ Monte Carlo sampling, which results in:

$$L_{\text{DM}} = \sum_{n=1}^{N}\sum_{t=1}^{T}w(t)u(\mathbf{x}_0,\mathbf{x}_t,n;\boldsymbol{\theta}). \tag{7}$$

For a sequence of length N, the probability of sampling the same $\mathbf{x}_t$ in AR is 1. However, in diffusion, this probability reduces to $1/C_{N-1}^{t(N-1)/T}$ due to the randomness in sampling $\mathbf{x}_t$, potentially reducing the training efficiency of diffusion models. We note that Equation (7) employs a sequence-level reweighting term $w(t)$ to indicate the importance of $\mathbf{x}_t$. However, individual tokens within the sequence, given their imbalanced difficulties, are not properly reweighted. To address this, we propose multi-granularity diffusion modeling (MDM), which introduces an additional token-level reweighting mechanism to enhance training efficiency:

$$L_{\text{MDM}} = \sum_{n=1}^{N}\sum_{t=1}^{T} w(t)v(\mathbf{x}_{t,n})u(\mathbf{x}_0, \mathbf{x}_t, n; \boldsymbol{\theta}), \tag{8}$$

where $v(\mathbf{x}_{t,n}) = \alpha(1-\exp(-u(\cdot)))^{\beta}$ is the adaptive token-level reweighting term. Setting $\beta > 0$ reduces the relative loss for easy tokens while emphasizing harder tokens, and α is used to control the relative reweighting magnitude. For inference, we employ an easy-first TopK decoding strategy, which has demonstrated superior performance compared to the random decoding method used by Austin et al. (2021). This finding aligns with similar observations documented in prior studies (Savinov et al., 2021; Zheng et al., 2023). We provide a detailed derivation and algorithm of the training and inference process in Appendix §A and §B, respectively.

4 EXPERIMENTS

In Section §3.1 we show our model works well on a straightforward planning task with only one hard subgoal. However, it is important to note that real-world scenarios often involve instances with multiple challenging subgoals. In this section, we aim to assess the performance of our model in tackling three considerably more complex problem-solving tasks that necessitate deliberate planning. Detailed experimental setup can be found in Appendix §C.

4.1 COUNTDOWN

Countdown (Countdown, 2024) is a mathematical reasoning challenge and is a generalized version of the game of 24, which even advanced models such as GPT-4 struggle with (Yao et al., 2024). The goal of Countdown is to use the given numbers and arithmetic operations $(+ - */)$ to obtain a target number. For example, given 4 numbers "97,38,3,17" and a target number "14", a step-by-step solution is "97-38=59,59-17=42,42/3=14".

Setup. We follow Gandhi et al. (2024) to generate 500k problems with target numbers ranging from 10 to 100 and randomly hold out 10% of the targets for 'out-of-distribution' evaluation. We consider three subtasks with increasing complexity by varying the number of input digits in $\{3,4,5\}$. Given that search-augmented prompting approaches (Yao et al., 2024) have recently been employed to address the limitations of AR, we also compare with such approaches by training on Countdown 4 and evaluating on the same game of 24 test set as Yao et al. (2024).

Table 1: Results on the Countdown (CD) task with increasing complexity.

	Params	CD 3	CD 4	CD 5
Autoregressive				
	6M	94.1	31.9	4.3
GPT-2 Scratch	85M	95.9	45.8	5.1
	303M	96.4	41.3	4.5
Stream-of-Search	250M	-	54.2	-
LLaMA	7B	95.7	41.1	6.7
	13B	96.5	51.1	7.4
Diffusion				
VDM	85M	99.1	73.4	16.3
D3PM	85M	99.4	83.1	27.6
RDM	85M	99.5	87.0	45.8
	6M	98.1	52.0	27.0
MDM (Ours)	85M	99.5	**91.5**	**46.6**
	303M	**99.9**	88.3	39.0

Baselines. Our primary comparison involves autoregressive models trained from scratch, employing the GPT-2 architecture (Radford et al., 2019) with parameter sizes ranging from 6M, 85M, and 303M (denoted as GPT-2 Scratch). We also include larger pre-trained AR models LLaMA (Touvron et al., 2023) with sizes 7B and 13B. These models are fine-tuned using the same dataset. In addition, we compare with Stream-of-Search (Gandhi et al., 2024), which augments the dataset with search trajectory such that the AR model can be taught to search. Furthermore, we compare with several existing diffusion models, both continuous models VDM (Kingma et al., 2021) and discrete models D3PM Austin et al. (2021) and RDM (Zheng et al., 2023). By default, we use the absorbing noise for discrete diffusion as it significantly outperforms the multinomial one (Austin et al., 2021; Zheng et al., 2023). Finally, we consider in-context learning (Brown et al., 2020; Wu et al., 2023c; Ye et al., 2023a) based on GPT-4, including vanilla input-output (IO), chain-of-thought (CoT; Wei et al. 2022b), CoT with Self-consistency (CoT-SC; Wang et al. 2023) and Tree-of-thought (ToT; Yao et al. 2024). We use 5 in-context examples following Yao et al. 2024.

Results on Countdown. As shown in Table 1, diffusion-based approaches demonstrate superior performance across all three Countdown tasks compared to autoregressive models, especially as the complexity of the tasks increases. We have several key findings based on the result. Firstly, the 6M diffusion model outperforms both the 303M GPT-2 model trained from scratch and the pretrained 13B LLaMA model, indicating that the modeling approach sometimes outweighs the sheer number of parameters. Secondly, while training with search trajectory supervision (Stream-of-search) does provide some benefits, its effectiveness is limited. Importantly, training the entire search trajectory as a sequence poses additional challenges due to its long length, such as in the case of Countdown 5 where the search trajectories can span 60,000 tokens. Lastly, our model surpasses all previous diffusion models, demonstrating the efficacy of the multigranularity loss.

Results on Game of 24. As shown in Table 2, the performance of the GPT-4 with IO, CoT, and CoT-SC prompting methods from Yao et al. (2024) is unsatisfactory for the given task, with only accuracy below 10%. The introduction of ToT, which incorporates a search algorithm designed by human experts into the decoding process, significantly enhances the performance of GPT-4. This integration allows the AR model to backtrack as needed, resulting in notable improvements. However, this paradigm requires the assessment of intermediate steps using LLM, resulting in considerable computational costs due to the need for multiple LLM calls. We list the token cost in Table 2 with more details in Appendix D.1. ToT consumes 186 times more tokens than MDM, showcasing the 'internal' search capability by promoting global consistency in diffusion modeling. In summary, our model, despite having a parameter size of only 85M, significantly outperforms both the AR task-specific model of the same size (GPT-2 Scratch) in terms of performance and the larger general pre-trained model (GPT-4) in computation cost, indicating it is challenging for model scaling and decoding strategies to substitute the advantages of modeling paradigm.

Table 2: Accuracy and token cost on game of 24.

	Acc.	# Token
Prompting		
GPT-4 IO	7.3	x28
GPT-4 CoT	4.0	x61
GPT-4 CoT-SC	9.0	x241
GPT-4 ToT	74.0	x186
Supervised training		
GPT-2 Scratch	18.8	**x1**
MDM	**76.0**	**x1**

4.2 SUDOKU

Sudoku is a classical logic-based number placement puzzle that has gained popularity due to its rigorous intellectual demands. The goal of Sudoku is to meticulously fill a 9×9 grid with numerical digits, ensuring that every column, row, and 3×3 subgrid contains all the numbers from 1 to 9.

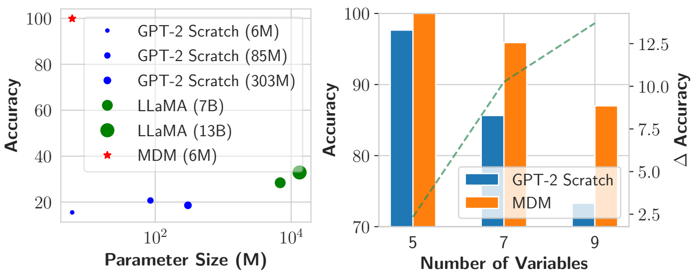

Figure 4: **(Left)** Accuracy on Sudoku. **(Right)** Accuracy on boolean satisfiability problem with increasing difficulty.

Setup. We collect one million solved games from Park (2016) and use the first 100k as our training set and the subsequent 1k as the testing set. We employ the digit 0 to represent the vacant position that needs to be filled. We then transform the 9×9 grid into a sequence of 81 digits, which serves as the model input. To illustrate, an example input appears as "080050060...603100007" (omitted for brevity), while the corresponding output is represented as "789251364...653184297". During tokenization, we treat each digit as a separate token.

	Random	TopK
No reweighting	82.1	87.3
Original sequence-reweighting	83.1	88.5
+ token-reweighting (α=0.25, β=1)	84.9	**90.4**
+ token-reweighting (α=1, β=1)	82.4	89.3
+ token-reweighting (α=0.25, β=2)	82.4	87.9
Linear sequence-reweighting	79.6	87.0
+ token-reweighting (α=0.25, β=1)	83.2	88.0
+ token-reweighting (α=1, β=1)	86.7	90.4
+ token-reweighting (α=0.25, β=2)	85.6	**91.5**

Table 3: Ablation on training reweighting strategies and inference decoding methods.

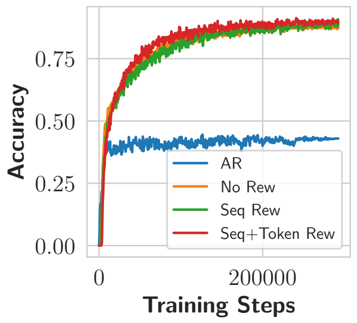

Figure 5: Evaluation accuracy throughout the training process for AR and MDM with different reweighting strategies.

Result. We show the results in the left figure of Figure 4. As the size of the AR model increases, the performance remains unsatisfactory. For instance, the LLaMA model achieves a performance of only 32.9 with 13B parameters. In contrast, our model, which has only 6M parameters, is able to perfectly solve all the problems, demonstrating the significant advantage brought by the modeling architecture.

4.3 Boolean Satisfiability Problem

The Boolean satisfiability problem, commonly known as SAT, is a foundational problem in computer science that has been rigorously proven to be NP-complete (Cook, 1971). This challenging combinatorial problem is attractive as a broad range of search problems from domains such as software verification, test pattern generation, planning, scheduling, and combinatorics can all routinely be solved by reducing to an appropriate SAT problem (Gomes et al., 2008). The goal of SAT is to determine whether a given Boolean formula represented in conjunctive normal form (CNF) can be assigned a set of values (0 or 1) to its variables, such that the formula evaluates to true (1). An example formula with three variables can be $(x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge \neg x_1$ and an corresponding assignment is $x_1 = 0, x_2 = 0, x_3 = 1$.

Setup. Given the number of variables n and clauses m, we iteratively sample $k = 3$ variables (and their polarities) uniformly at random until m clauses are obtained. To ensure that we get relatively hard instances of SAT, we take advantage of the well-studied family of random k-SAT problem (Ding et al., 2015) and set the m to be close to $m = 4.258n + 58.26n^{-2/3}$ given n, as it has been observed that SAT solvers are slow to determine the satisfiability of a formula when m is near the threshold (Crawford & Auton, 1996). We consider increasing numbers of variables from $\{5,7,9\}$ and generate 50k training data for $n = 5, 7$ and 100k for $n = 9$, as well as additional 1k testing data for each n.

Result. As shown in the right figure of Figure 4, MDM performs well in solving scenarios with five variables, while the AR model falls slightly short. As the number of variables increases, both our model and the AR model experience a certain degree of decrease in accuracy. However, the performance gap between the two models widens as the difficulty of the task increases. This indicates that our diffusion model exhibits a more pronounced advantage in handling more challenging tasks than the AR counterpart.

4.4 Analysis

On the Effect of Training and Decoding Strategies. As listed in Table 3, we find that changing the sequence-reweighting strategies has only led to a slight improvement in performance. However, when a suitable parameter is selected for token-reweighting, a more significant improvement can be observed. Additionally, the easy first decoding (TopK) outperforms the random one, which aligns with previous findings (Ghazvininejad et al., 2019; Zheng et al., 2023). We compare the evaluation accuracy along the training process in Figure 5. By aligning the AR training steps with the diffusion process, we can see AR converges rapidly, with the performance tends to plateau afterward. The utilization of our multi-granularity loss, which incorporates sequence and token reweighting, demonstrates superior performance, particularly during the middle stages of training. This implies that the inclusion of such a loss function contributes to enhanced convergence during the training process.

On Decoding Speed. We assess the trade-off between accuracy and decoding speed by comparing the performance of the AR model (GPT-2 Scratch 85M) and MDM (85M). The speed metric is determined by the number of samples processed using a batch size of 1 on the NVIDIA GeForce V100 GPU. As shown in Figure 6(a), MDM can flexibly control the trade-off between accuracy and decoding speed by varying the diffusion timesteps. Notably, by employing just one diffusion step, MDM demonstrates a remarkable 10x improvement in speed compared to AR, while maintaining superior accuracy with 75% and 12.7% compared to 45.8% and 5.1% of AR on Countdown 4 and 5, respectively. We observed that the slope of countdown 4 is smaller compared to countdown 5 in the trade-off. This suggests that for tasks with lower complexity, diffusion demonstrates a more noticeable speed advantage by setting a smaller diffusion step. In addition, it also indicates that sacrificing some efficiency for performance improvements becomes particularly evident when dealing with more intricate tasks.

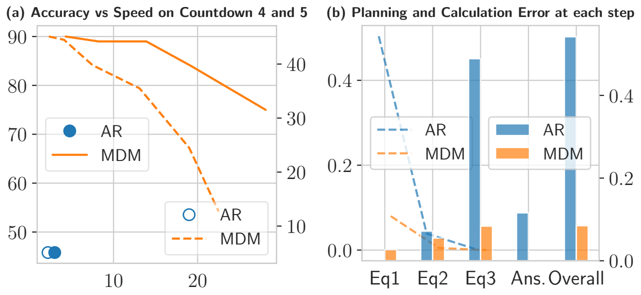

Figure 6: **(a)** Accuracy and speed (samples per second) trade-off by varying the diffusion timesteps on Countdown 4 (left y-axis) and 5 (right y-axis).**(b)** Ratio of planning (left y-axis) and calculation (right y-axis) error at each reasoning step on Countdown 4.

Error Analysis: The Regretful Compromise. To gain a deeper understanding of error patterns in AR and MDM, we conducted an error analysis on Countdown 4. For instance, given the input “7,38,3,1” and the target number 14, a correct solution would be “97-38=59,59-17=42,42/3=14”. We divide the solution into four parts: equation 1, equation 2, equation 3, and answer checking. First, from a calculation perspective, we assess the error ratio in each equation by comparing the left-hand side and the right-hand side, regardless of whether the correct number was chosen. As shown in Figure 6(b), the bar plot demonstrates that the majority of calculation errors for AR are concentrated in Equation 3. For example, given input “16,4,40,51” and target 87, the prediction of AR is “51-40=11,16*4=64,11+64=87” while the correct solution is “16/4=4,40+51=91,91-4=87”. We can observe that in the last equation, the model realizes that it needs to output 87 as the final result. Therefore, even though the model most likely knows that 11 + 64 actually equals 75 given the low calculation error in the former equations, it reluctantly outputs 87 due to it being the last equation. This significantly increases the number of calculation errors in the third equation. We call this phenomenon ‘The Regretful Compromise’. The reason for this is that the AR model made incorrect choices of numbers or operations in previous equations. This is demonstrated by examining the step at which the models fail the task, as depicted in Figure 6(b). It is evident that there is a notable frequency of planning errors in the first equation for the AR model, where the number of errors is significantly higher compared to our model. This highlights the limitations of the left-to-right decoding approach in AR, which adversely affects its planning ability.

5 Related Work

5.1 Autoregressive Modeling

Starting from Bengio et al. (2000) and later Sutskever et al. (2011), the autoregressive modeling paradigm, where the prediction of a token only depends on the preceding context, is widely adopted in modeling language, until recently (OpenAI, 2022; Achiam et al., 2023; Anthropic, 2023; Team et al., 2023; Touvron et al., 2023; Jiang et al., 2023; Bai et al., 2023, *inter alia*). Theoretically, the autoregressive Transformers have limited expressive power, but their capabilities can be expanded given sufficient chain-of-thought intermediate steps (Wei et al., 2022b; Merrill & Sabharwal, 2023; Malach, 2023). However, Lin et al. (2021) demonstrates that the expressing of some next-tokens requires super-polynomial computational resources and is NP-hard to approximate. Numerous advancements have been made upon the AR paradigm to compensate for modeling deficiencies, such as reverse training (Lee et al., 2023; Golovneva et al., 2024),fill-in-the-middle training (Bavarian et al., 2022), future-token prediction (Qi et al., 2020; Gloeckle et al., 2024), lookahead attention (Du et al., 2023) during the training stage, as well as search-augmented decoding (Lu et al., 2022; Xie et al., 2023; Yao et al., 2024, *inter alia*) during inference. In practice, autoregressive next-token predictors are shown to be ill-suited for planning tasks (Bubeck et al., 2023; Valmeekam et al., 2023; 2024; Dziri et al., 2024; Kambhampati et al., 2024). Besides, Bachmann & Nagarajan (2024); Lin et al. (2024) find not all tokens are equal and some tokens are hard to learn in the AR pretraining stage, implying the introduced subgoal imbalance phenomenon also exists in the general text corpus.

5.2 Non-autoregressive Modeling

The non-autoregressive (NAR) generation method, which produces all target tokens simultaneously given the source context, is first proposed by (Gu et al., 2017) in the text field for machine translation, primarily due to the efficiency consideration. While a series of advancements have been made afterward (Lee et al., 2018; Gu et al., 2019; Ghazvininejad et al., 2019; Qian et al., 2021; Huang et al., 2022, *inter alia*), traditional NAR models still tend to underperform AR models in terms of generation quality (Xiao et al., 2023). Diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020), a powerful class of generative models known for their impressive image-generation capabilities (Dhariwal & Nichol, 2021), have recently been applied to the field of text generation (Hoogeboom et al., 2021; Austin et al., 2021; Li et al., 2022; Campbell et al., 2022; Dieleman et al., 2022; Chen et al., 2023; Ye et al., 2023b; Lovelace et al., 2024), reinforcement learning (Janner et al., 2022; Chi et al., 2023) and protein design (Xu et al., 2022; Hoogeboom et al., 2022b; Corso et al., 2023). In essence, diffusion models perform a multi-step denoising process to progressively convert a random noise into a data sample, and the denoising procedure can be seen as parameterizing the gradients of the data distribution (Song & Ermon, 2019), connecting them to score matching (Hyvärinen & Dayan, 2005) and energy-based models (LeCun et al., 2006). For text, the diffusion model can be seen as an extension of the traditional iterative NAR models (Gong et al., 2022) and has been shown to approach or outperform AR models on text perplexity (Han et al., 2023; Lou et al., 2023; Gulrajani & Hashimoto, 2024), diversity (Gong et al., 2022; 2023; Zhang et al., 2023) as well as various seq-to-seq tasks (Wu et al., 2023b; Zheng et al., 2023; Ye et al., 2024). In contrast, we compare diffusion with AR from a perspective of subgoal imbalance and demonstrates the effectiveness of diffusion in tasks requiring complex reasoning and planning.

6 Conclusion

This paper presents an extensive analysis of the limitations of auto-regressive (AR) language models when applied to planning tasks that involve deliberate planning, both in controlled settings and real-world contexts. Based on an advanced understanding, we propose an improved diffusion model, named MDM, that performs significantly better than AR and previous diffusion models on various sophisticated planning tasks. Our findings underscore the necessity to reevaluate the sequence modeling paradigm for modern large language models, especially in tackling challenging problem-solving tasks.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Anurag Ajay, Yilun Du, Abhi Gupta, Joshua B Tenenbaum, Tommi S Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? In *NeurIPS 2022 Foundation Models for Decision Making Workshop*, 2022.

Anthropic. Introducing Claude, 2023. URL `https://www.anthropic.com/index/introducing-claude`.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 17981–17993, 2021.

Gregor Bachmann and Vaishnavh Nagarajan. The pitfalls of next-token prediction. *arXiv preprint arXiv:2403.06963*, 2024.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

Mohammad Bavarian, Heewoo Jun, Nikolas Tezak, John Schulman, Christine McLeavey, Jerry Tworek, and Mark Chen. Efficient training of language models to fill in the middle. *arXiv preprint arXiv:2207.14255*, 2022.

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. A neural probabilistic language model. *Advances in neural information processing systems*, 13, 2000.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 17682–17690, 2024.

Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators. 2024. URL `https://openai.com/research/video-generation-models-as-world-simulators`.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

Andrew Campbell, Joe Benton, Valentin De Bortoli, Thomas Rainforth, George Deligiannidis, and Arnaud Doucet. A continuous time framework for discrete denoising models. *Advances in Neural Information Processing Systems*, 35:28266–28279, 2022.

Ting Chen, Ruixiang ZHANG, and Geoffrey Hinton. Analog bits: Generating discrete data using diffusion models with self-conditioning. In *The Eleventh International Conference on Learning Representations*, 2023.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023.

Stephen A Cook. The complexity of theorem-proving procedures, 1971.

Gabriele Corso, Hannes Stärk, Bowen Jing, Regina Barzilay, and Tommi S Jaakkola. Diffdock: Diffusion steps, twists, and turns for molecular docking. In *The Eleventh International Conference on Learning Representations*, 2023.

Countdown. Countdown (game show), 2024. URL `https://en.wikipedia.org/wiki/Countdown_(game_show)`.

James M Crawford and Larry D Auton. Experimental results on the crossover point in random 3-sat. *Artificial intelligence*, 81(1-2):31–57, 1996.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL `https://aclanthology.org/N19-1423`.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

Sander Dieleman, Laurent Sartran, Arman Roshannai, Nikolay Savinov, Yaroslav Ganin, Pierre H Richemond, Arnaud Doucet, Robin Strudel, Chris Dyer, Conor Durkan, et al. Continuous diffusion for categorical data. *arXiv preprint arXiv:2211.15089*, 2022.

Jian Ding, Allan Sly, and Nike Sun. Proof of the satisfiability conjecture for large k. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pp. 59–68, 2015.

Li Du, Hongyuan Mei, and Jason Eisner. Autoregressive modeling with lookahead attention. *arXiv preprint arXiv:2305.12272*, 2023.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, et al. Faith and fate: Limits of transformers on compositionality. *Advances in Neural Information Processing Systems*, 36, 2024.

Kanishk Gandhi, Denise Lee, Gabriel Grand, Muxin Liu, Winson Cheng, Archit Sharma, and Noah D Goodman. Stream of search (sos): Learning to search in language. *arXiv preprint arXiv:2404.03683*, 2024.

Howard Garns. Sudoku, 1979. URL `https://en.wikipedia.org/wiki/Sudoku`.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Mask-predict: Parallel decoding of conditional masked language models. *arXiv preprint arXiv:1904.09324*, 2019.

Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Synnaeve. Better & faster large language models via multi-token prediction. *arXiv preprint arXiv:2404.19737*, 2024.

Olga Golovneva, Zeyuan Allen-Zhu, Jason Weston, and Sainbayar Sukhbaatar. Reverse training to nurse the reversal curse. *arXiv preprint arXiv:2403.13799*, 2024.

Carla P Gomes, Henry Kautz, Ashish Sabharwal, and Bart Selman. Satisfiability solvers. *Foundations of Artificial Intelligence*, 3:89–134, 2008.

Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. Diffuseq: Sequence to sequence text generation with diffusion models. In *The Eleventh International Conference on Learning Representations*, 2022.

Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. DiffuSeq-v2: Bridging discrete and continuous text spaces for accelerated Seq2Seq diffusion models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 9868–9875, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.660. URL `https://aclanthology.org/2023.findings-emnlp.660`.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. Non-autoregressive neural machine translation. *arXiv preprint arXiv:1711.02281*, 2017.

Jiatao Gu, Changhan Wang, and Junbo Zhao. Levenshtein transformer. *Advances in neural information processing systems*, 32, 2019.

Ishaan Gulrajani and Tatsunori B Hashimoto. Likelihood-based diffusion language models. *Advances in Neural Information Processing Systems*, 36, 2024.

Xiaochuang Han, Sachin Kumar, and Yulia Tsvetkov. Ssd-lm: Semi-autoregressive simplex-based diffusion language model for text generation and modular control. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11575–11596, 2023.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in Neural Information Processing Systems*, 35:8633–8646, 2022.

Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 12454–12465, 2021.

Emiel Hoogeboom, Alexey A Gritsenko, Jasmijn Bastings, Ben Poole, Rianne van den Berg, and Tim Salimans. Autoregressive diffusion models. In *International Conference on Learning Representations*, 2022a.

Emiel Hoogeboom, Vıctor Garcia Satorras, Clément Vignac, and Max Welling. Equivariant diffusion for molecule generation in 3d. In *International conference on machine learning*, pp. 8867–8887. PMLR, 2022b.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2021.

Fei Huang, Hao Zhou, Yang Liu, Hang Li, and Minlie Huang. Directed acyclic transformer for non-autoregressive machine translation. In *International Conference on Machine Learning*, pp. 9410–9428. PMLR, 2022.

Aapo Hyvärinen and Peter Dayan. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

Frederick Jelinek. Interpolated estimation of markov source parameters from sparse data. In *Proc. Workshop on Pattern Recognition in Practice, 1980*, 1980.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

Daniel D Johnson, Jacob Austin, Rianne van den Berg, and Daniel Tarlow. Beyond in-place corruption: Insertion and deletion in denoising probabilistic models. In *ICML Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models*, 2022.

Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Kaya Stechly, Mudit Verma, Siddhant Bhambri, Lucas Saldyt, and Anil Murthy. Llms can't plan, but can help planning in llm-modulo frameworks. *arXiv preprint arXiv:2402.01817*, 2024.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707, 2021.

Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and Fujie Huang. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. Deterministic non-autoregressive neural sequence modeling by iterative refinement. *arXiv preprint arXiv:1802.06901*, 2018.

Nayoung Lee, Kartik Sreenivasan, Jason D Lee, Kangwook Lee, and Dimitris Papailiopoulos. Teaching arithmetic to small transformers. *arXiv preprint arXiv:2307.03381*, 2023.

Lucas Lehnert, Sainbayar Sukhbaatar, Paul Mcvay, Michael Rabbat, and Yuandong Tian. Beyond a*: Better planning with transformers via search dynamics bootstrapping. *arXiv preprint arXiv:2402.14083*, 2024.

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B Hashimoto. Diffusion-lm improves controllable text generation. In *Conference on Neural Information Processing Systems, NeurIPS*, 2022.

Chu-Cheng Lin, Aaron Jaech, Xin Li, Matthew R Gormley, and Jason Eisner. Limitations of autoregressive models and their alternatives. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 5147–5173, 2021.

Zhenghao Lin, Zhibin Gou, Yeyun Gong, Xiao Liu, Yelong Shen, Ruochen Xu, Chen Lin, Yujiu Yang, Jian Jiao, Nan Duan, et al. Rho-1: Not all tokens are what you need. *arXiv preprint arXiv:2404.07965*, 2024.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion language modeling by estimating the ratios of the data distribution. *ArXiv preprint*, abs/2310.16834, 2023.

Justin Lovelace, Varsha Kishore, Chao Wan, Eliot Shekhtman, and Kilian Q Weinberger. Latent diffusion for language generation. *Advances in Neural Information Processing Systems*, 36, 2024.

Ximing Lu, Sean Welleck, Peter West, Liwei Jiang, Jungo Kasai, Daniel Khashabi, Ronan Le Bras, Lianhui Qin, Youngjae Yu, Rowan Zellers, et al. Neurologic a* esque decoding: Constrained text generation with lookahead heuristics. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 780–799, 2022.

Eran Malach. Auto-regressive next-token predictors are universal learners. *arXiv preprint arXiv:2309.06979*, 2023.

William Merrill and Ashish Sabharwal. The expresssive power of transformers with chain of thought. *arXiv preprint arXiv:2310.07923*, 2023.

Giovanni Monea, Armand Joulin, and Edouard Grave. Pass: Parallel speculative sampling. *arXiv preprint arXiv:2311.13581*, 2023.

OpenAI. Introducing ChatGPT, 2022. URL `https://openai.com/blog/chatgpt`.

Kyubyong Park. 1 million sudoku games, 2016. URL `https://www.kaggle.com/bryanpark/sudoku`.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4195–4205, 2023.

Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. Prophetnet: Predicting future n-gram for sequence-to-sequence pre-training. *arXiv preprint arXiv:2001.04063*, 2020.

Lihua Qian, Hao Zhou, Yu Bao, Mingxuan Wang, Lin Qiu, Weinan Zhang, Yong Yu, and Lei Li. Glancing transformer for non-autoregressive neural machine translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 1993–2003, 2021.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Machel Reid, Vincent J Hellendoorn, and Graham Neubig. Diffuser: Discrete diffusion via edit-based reconstruction. *arXiv preprint arXiv:2210.16886*, 2022.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. *arXiv preprint arXiv:2406.07524*, 2024.

Nikolay Savinov, Junyoung Chung, Mikolaj Binkowski, Erich Elsen, and Aaron van den Oord. Step-unrolled denoising autoencoders for text generation. In *International Conference on Learning Representations*, 2021.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K Titsias. Simplified and generalized masked diffusion for discrete data. *arXiv preprint arXiv:2406.04329*, 2024.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. PMLR, 2015.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 11895–11907, 2019.

Haoran Sun, Lijun Yu, Bo Dai, Dale Schuurmans, and Hanjun Dai. Score-based continuous-time discrete diffusion models. In *The Eleventh International Conference on Learning Representations*, 2023.

Ilya Sutskever, James Martens, and Geoffrey E Hinton. Generating text with recurrent neural networks. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 1017–1024, 2011.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Trieu H Trinh, Yuhuai Wu, Quoc V Le, He He, and Thang Luong. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482, 2024.

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

Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change. *Advances in Neural Information Processing Systems*, 36, 2024.

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.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *Transactions on Machine Learning Research*, 2022a.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022b.

Jay Zhangjie Wu, Yixiao Ge, Xintao Wang, Stan Weixian Lei, Yuchao Gu, Yufei Shi, Wynne Hsu, Ying Shan, Xiaohu Qie, and Mike Zheng Shou. Tune-a-video: One-shot tuning of image diffusion models for text-to-video generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7623–7633, 2023a.

Tong Wu, Zhihao Fan, Xiao Liu, Hai-Tao Zheng, Yeyun Gong, Jian Jiao, Juntao Li, Jian Guo, Nan Duan, Weizhu Chen, et al. Ar-diffusion: Auto-regressive diffusion model for text generation. *Advances in Neural Information Processing Systems*, 36:39957–39974, 2023b.

Yuhuai Wu, Albert Qiaochu Jiang, Wenda Li, Markus Rabe, Charles Staats, Mateja Jamnik, and Christian Szegedy. Autoformalization with large language models. *Advances in Neural Information Processing Systems*, 35:32353–32368, 2022.

Zhiyong Wu, Yaoxiang Wang, Jiacheng Ye, and Lingpeng Kong. Self-adaptive in-context learning: An information compression perspective for in-context example selection and ordering. In *The 61st Annual Meeting Of The Association For Computational Linguistics*, 2023c.

Yisheng Xiao, Lijun Wu, Junliang Guo, Juntao Li, Min Zhang, Tao Qin, and Tie-yan Liu. A survey on non-autoregressive generation for neural machine translation and beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, Xu Zhao, Min-Yen Kan, Junxian He, and Qizhe Xie. Decomposition enhances reasoning via self-evaluation guided decoding. *arXiv preprint arXiv:2305.00633*, 2023.

Chang Xu, Dacheng Tao, and Chao Xu. A survey on multi-view learning. *arXiv preprint arXiv:1304.5634*, 2013.

Minkai Xu, Lantao Yu, Yang Song, Chence Shi, Stefano Ermon, and Jian Tang. Geodiff: A geometric diffusion model for molecular conformation generation. In *International Conference on Learning Representations*, 2022.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper_files/paper/2019/file/dc6a7e655d7e5840e66733e9ee67cc69-Paper.pdf`.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023.

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, 2024.

Jiacheng Ye, Zhiyong Wu, Jiangtao Feng, Tao Yu, and Lingpeng Kong. Compositional exemplars for in-context learning. In *International Conference on Machine Learning*, pp. 39818–39833. PMLR, 2023a.

Jiacheng Ye, Shansan Gong, Liheng Chen, Lin Zheng, Jiahui Gao, Han Shi, Chuan Wu, Zhenguo Li, Wei Bi, and Lingpeng Kong. Diffusion of thoughts: Chain-of-thought reasoning in diffusion language models. *arXiv preprint arXiv:2402.07754*, 2024.

Jiasheng Ye, Zaixiang Zheng, Yu Bao, Lihua Qian, and Mingxuan Wang. Dinoiser: Diffused conditional sequence learning by manipulating noises. *arXiv preprint arXiv:2302.10025*, 2023b.

Yizhe Zhang, Jiatao Gu, Zhuofeng Wu, Shuangfei Zhai, Joshua M. Susskind, and Navdeep Jaitly. PLANNER: Generating diversified paragraph via latent language diffusion model. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Xueliang Zhao, Wenda Li, and Lingpeng Kong. Decomposing the enigma: Subgoal-based demonstration learning for formal theorem proving. *arXiv preprint arXiv:2305.16366*, 2023.

Lin Zheng, Jianbo Yuan, Lei Yu, and Lingpeng Kong. A reparameterized discrete diffusion model for text generation. *ArXiv preprint*, abs/2302.05737, 2023.

A More Background and Derivation of Discrete Diffusion Models

A.1 Background

Discrete diffusion probabilistic models are first introduced in Sohl-Dickstein et al. (2015) for binary data, and later extended to categorical data in (Hoogeboom et al., 2021). Austin et al. (2021) provides a general form of discrete diffusion and introduces multiple transition matrices, including an absorbing variant that draws close connections to masked language models (Devlin et al., 2019). Several subsequent works push this line of research further from various aspects, such as incorporating editing-based operations (Johnson et al., 2022; Reid et al., 2022), casting permuted language models (Yang et al., 2019) as diffusion models (Hoogeboom et al., 2022a), developing a continuous-time framework (Campbell et al., 2022), parameterizing the routing mechanism (Zheng et al., 2023), and investigating score functions for learning the reverse process Sun et al. (2023); Lou et al. (2023).

A.2 Derivation Setup

We now provide a detailed derivation of the loss in Equation (6). For a clear illustration, we initiate derivation with a single random variable $\boldsymbol{x}_0$ and ultimately link it with the multi-variable sequence $\mathbf{x}_0$. Suppose $\boldsymbol{x}_0 \sim q(\boldsymbol{x}_0)$ is a discrete random variable with K possible categories and represented as a one-hot vector. The forward process $q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0) = \prod_{t=1}^{T} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ corrupts the original data $\boldsymbol{x}_0$ into a sequence of increasingly noisy latent variables $\boldsymbol{x}_{1:T} := \boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$. The learned backward process $p_{\boldsymbol{\theta}}(\boldsymbol{x}_{0:T}) = p(\boldsymbol{x}_T)\prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ gradually denoises the latent variables to the data distribution. In discrete diffusion, both the forward and backward distribution are defined as categorical distribution, e.g., $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \text{Cat}(\boldsymbol{x}_t; \boldsymbol{p} = \boldsymbol{Q}_t^\top \boldsymbol{x}_{t-1})$ and $p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, f(\boldsymbol{x}_t; \boldsymbol{\theta}))$, where $\boldsymbol{Q}_t$ is a pre-defined transition matrix of size $K \times K$ (Hoogeboom et al., 2021; Austin et al., 2021).

A.3 The marginal and posterior

Starting from $\boldsymbol{x}_0$, we obtain the following t-step marginal and posterior at time $t-1$:

$$q(\boldsymbol{x}_t|\boldsymbol{x}_0) = \text{Cat}\left(\boldsymbol{x}_t; \boldsymbol{p} = \overline{\boldsymbol{Q}}_t^\top \boldsymbol{x}_0\right), \quad \text{with} \quad \overline{\boldsymbol{Q}}_t = \boldsymbol{Q}_1\boldsymbol{Q}_2\ldots\boldsymbol{Q}_t$$

$$q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) = \frac{q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_0)q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)}{q(\boldsymbol{x}_t|\boldsymbol{x}_0)} = \text{Cat}\left(\boldsymbol{x}_{t-1}; \boldsymbol{p} = \frac{\boldsymbol{Q}_t\boldsymbol{x}_t \odot \overline{\boldsymbol{Q}}_{t-1}^\top \boldsymbol{x}_0}{\boldsymbol{x}_t^\top \overline{\boldsymbol{Q}}_t^\top \boldsymbol{x}_0}\right), \tag{9}$$

where $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_0) = q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ due to the Markov property of the forward process. The KL divergence between q and $p_{\boldsymbol{\theta}}$ can be computed by simply summing over all possible values of each random variable. The cumulative products $\overline{\boldsymbol{Q}}_t$, which can be computed in closed form or precomputed for all t depending on the choice $\boldsymbol{Q}_t$, may be prohibitive for large T and number of categories. Therefore, two commonly used forms of $\boldsymbol{Q}$ are introduced by Hoogeboom et al. (2021) and Austin et al. (2021), which ensures $\overline{\boldsymbol{Q}}_t$ can still be computed efficiently, allowing the framework to scale to a larger number of categories.

A.4 Transition Matrix

Austin et al. (2021) introduced multiple types of the transition matrix $\boldsymbol{Q}_t$, such as uniform (Hoogeboom et al., 2021), absorbing, discretized Gaussian and token embedding distance. The absorbing noise for discrete diffusion has been demonstrated to outperform the others (Austin et al., 2021), where the transition matrix is given by :

$$[\boldsymbol{Q}_t]_{ij} = \begin{cases} 1 & \text{if} \quad i = j = m \\ 1 - \beta_t & \text{if} \quad i = j \neq m \\ \beta_t & \text{if} \quad j = m, i \neq m \end{cases}.$$

The transition matrix can also be written as $(1-\beta_t)I + \beta_t \mathbf{1} e_m^\top$, where e_m is a vector with a one on the absorbing state m and zeros elsewhere. Since m is an absorbing state, the corruption process converges not to a uniform distribution but to the point-mass distribution on m. The transition matrices $\overline{\boldsymbol{Q}} = \boldsymbol{Q}_1\boldsymbol{Q}_2 \dots \boldsymbol{Q}_t$ can be computed in closed form. Specifically, we transition to another token with probability β_t and stay the same with probability $1-\beta_t$ in each step. After t steps, the only operative quantity is the probability of not yet having transitioned to another token, given by $\alpha_t = \prod_{i=0}^{t}(1-\beta_i)$. Therefore, we have $\overline{\boldsymbol{Q}}_t = \alpha_t I + (1-\alpha_t)\mathbf{1}e_m^\top$.

A.5 Derivation of ELBO

In order to optimize the generative model $p_{\boldsymbol{\theta}}(\boldsymbol{x}_0)$ to fit the data distribution $q(\boldsymbol{x}_0)$, we typically minimize a variational upper bound on the negative log-likelihood, defined below:

$$
\begin{aligned}
&-\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_0)\\
&= -\log \int p_{\boldsymbol{\theta}}(\boldsymbol{x}_0, \boldsymbol{x}_1, \dots, \boldsymbol{x}_T) d\boldsymbol{x}_1 \cdots d\boldsymbol{x}_T\\
&= -\log \int \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_0, \boldsymbol{x}_1, \dots, \boldsymbol{x}_T)}{q(\boldsymbol{x}_1, \dots, \boldsymbol{x}_T|\boldsymbol{x}_0)} q(\boldsymbol{x}_1, \dots, \boldsymbol{x}_T|\boldsymbol{x}_0) d\boldsymbol{x}_1 \cdots d\boldsymbol{x}_T\\
&= -\log \mathbb{E}_{q(\boldsymbol{x}_1,\dots,\boldsymbol{x}_T|\boldsymbol{x}_0)}\left[\frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_0, \boldsymbol{x}_1, \dots, \boldsymbol{x}_T)}{q(\boldsymbol{x}_1, \dots, \boldsymbol{x}_T|\boldsymbol{x}_0)}\right]\\
&\le -\mathbb{E}_{q(\boldsymbol{x}_1,\dots,\boldsymbol{x}_T|\boldsymbol{x}_0)}\left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_0, \boldsymbol{x}_1, \dots, \boldsymbol{x}_T)}{q(\boldsymbol{x}_1, \dots, \boldsymbol{x}_T|\boldsymbol{x}_0)}\right]\\
&= -\mathbb{E}_{q(\boldsymbol{x}_1,\dots,\boldsymbol{x}_T|\boldsymbol{x}_0)}\left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1)p_{\boldsymbol{\theta}}(\boldsymbol{x}_T)\prod_{t=2}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}{q(\boldsymbol{x}_T|\boldsymbol{x}_0)\prod_{t=2}^{T} q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)}\right]\\
&= -\mathbb{E}_{q(\boldsymbol{x}_1,\dots,\boldsymbol{x}_T|\boldsymbol{x}_0)}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1) - \sum_{t=2}^{T} \log \frac{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)}{p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)} - \log \frac{q(\boldsymbol{x}_T|\boldsymbol{x}_0)}{p_{\boldsymbol{\theta}}(\boldsymbol{x}_T)}\right]\\
&= -\mathbb{E}_{q}\Big[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1) - \sum_{t=2}^{T} D_{\mathrm{KL}}[q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)||p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)] - \underbrace{D_{\mathrm{KL}}[q(\boldsymbol{x}_T|\boldsymbol{x}_0)||p(\boldsymbol{x}_T)]}_{L_T(\text{const})}\Big]\\
&= \underbrace{-\mathbb{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1)}_{L_0} + \sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\Big[D_{\mathrm{KL}}[q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)||p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)]\Big]}_{L_{t-1}} + L_T(\text{const}).
\end{aligned}
$$

A.6 Derivation for Equation (6)

The categorical distribution $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ based on Equation (9) is given as:

$$
\begin{aligned}
&q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)\\
&= \frac{\boldsymbol{Q}_t\boldsymbol{x}_t \odot \overline{\boldsymbol{Q}}_{t-1}^\top \boldsymbol{x}_0}{\boldsymbol{x}_t^\top \overline{\boldsymbol{Q}}_t^\top \boldsymbol{x}_0}\\
&= \frac{[(1-\beta_t)\boldsymbol{x}_t + \beta_t\sigma_{\boldsymbol{x}_t}\mathbf{1}] \odot [\alpha_{t-1}\boldsymbol{x}_0 + (1-\alpha_{t-1})e_m]}{\alpha_t\boldsymbol{x}_t^\top\boldsymbol{x}_0 + (1-\alpha_t)\boldsymbol{x}_t^\top e_m}\\
&= \frac{(1-\beta_t)\alpha_{t-1}\boldsymbol{x}_t \odot \boldsymbol{x}_0 + (1-\beta_t)(1-\alpha_{t-1})\boldsymbol{x}_t \odot e_m + \beta_t\alpha_{t-1}\sigma_{\boldsymbol{x}_t}\mathbf{1}\odot\boldsymbol{x}_0 + \beta_t(1-\alpha_{t-1})\sigma_{\boldsymbol{x}_t}\mathbf{1}\odot e_m}{\alpha_t\boldsymbol{x}_t^\top\boldsymbol{x}_0 + (1-\alpha_t)\boldsymbol{x}_t^\top e_m}\\
&= \frac{(1-\beta_t)\alpha_{t-1}\boldsymbol{x}_t \odot \boldsymbol{x}_0 + (1-\beta_t)(1-\alpha_{t-1})\sigma_{\boldsymbol{x}_t}\boldsymbol{x}_t + \beta_t\alpha_{t-1}\sigma_{\boldsymbol{x}_t}\boldsymbol{x}_0 + \beta_t(1-\alpha_{t-1})\sigma_{\boldsymbol{x}_t}e_m}{\alpha_t\boldsymbol{x}_t^\top\boldsymbol{x}_0 + (1-\alpha_t)\sigma_{\boldsymbol{x}_t}},
\end{aligned}
$$

where $\sigma_{\boldsymbol{x}_t} := e_m(\boldsymbol{u} = \boldsymbol{x}_t)$ represents the probability of noise drawn from e_m being equal to $\boldsymbol{x}_t$. Note $\boldsymbol{x}_t \odot \boldsymbol{x}_0 = 0$ if $\boldsymbol{x}_t \neq \boldsymbol{x}_0$ otherwise 1. Thus the computation of $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ breaks down into two cases:

$$
q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) = \begin{cases} \eta_t\boldsymbol{x}_t + (1-\eta_t)\, e_m, & \text{if } \boldsymbol{x}_t = \boldsymbol{x}_0\\ \lambda_t\boldsymbol{x}_0 + (1-\lambda_t)\, e_m(\boldsymbol{x}_t), & \text{if } \boldsymbol{x}_t \neq \boldsymbol{x}_0, \end{cases}
$$

where $\eta_t \coloneqq 1 - \frac{\beta_t(1-\alpha_{t-1})e_m(\boldsymbol{u}=\boldsymbol{x}_t)}{\alpha_t+(1-\alpha_t)e_m(\boldsymbol{u}=\boldsymbol{x}_t)}$, $\lambda_t \coloneqq \frac{\alpha_{t-1}-\alpha_t}{1-\alpha_t}$, and $e_m(\boldsymbol{x}_t) = (1-\beta_t)\boldsymbol{x}_t + \beta_t e_m$ denotes a noise distribution that interpolates between $\boldsymbol{x}_t$ and e_m.

Recall the distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ is parameterized by $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, f(\boldsymbol{x}_t;\boldsymbol{\theta}))$, the KL divergence between $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)$ and $p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ becomes 0 when $\boldsymbol{x}_t = \boldsymbol{x}_0$. In the case of absorbing diffusion, $\boldsymbol{x}_t = e_m$ if $\boldsymbol{x}_t \neq \boldsymbol{x}_0$ and $e_m(\boldsymbol{x}_t) = e_m$. $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)$ has probability λ_t on index x_0 and $1-\lambda_t$ on the absorbing state. The model $f(\boldsymbol{x}_t;\boldsymbol{\theta})$ has zero-probability on the absorbing state as it never predicts the mask token. Therefore, $p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ also has $1-\lambda_t$ probability on the absorbing state. Putting them together, we derive the KL divergence as:

$$\begin{aligned} D_{\mathrm{KL}}[q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)||p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)] &= 1_{x_t\neq x_0}[\lambda_t \log \frac{\lambda_t}{f(\boldsymbol{x}_t;\boldsymbol{\theta})_{x_0}} + (1-\lambda_t)\log\frac{1-\lambda_t}{1-\lambda_t}] \\ &= -\lambda_t 1_{x_t\neq x_0}\boldsymbol{x}_0^\top \log f(\boldsymbol{x}_t;\boldsymbol{\theta}) + C, \end{aligned}$$

where $1_{x_t\neq x_0}$ is 1 if $x_t \neq x_0$ otherwise 0, and C is a constant. Moreover, given $\alpha_0 = 1$ by definition and $\lambda_0 = 1$, we have:

$$L(\boldsymbol{x}_0) = -\mathbb{E}_{q(\boldsymbol{x}_0)}\sum_{t=1}^{T}\lambda_t \mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)} 1_{\boldsymbol{x}_t\neq\boldsymbol{x}_0}\boldsymbol{x}_0^\top \log f(\boldsymbol{x}_t;\boldsymbol{\theta})$$

for a single random variable, and

$$L(\mathbf{x}_0) = -\sum_{n=1}^{N}\mathbb{E}_{q(\mathbf{x}_{0,n})}\sum_{t=1}^{T}\lambda_t \mathbb{E}_{q(\mathbf{x}_{t,n}|\mathbf{x}_{0,n})} 1_{\mathbf{x}_{t,n}\neq\mathbf{x}_{0,n}}\mathbf{x}_{0,n}^\top \log f(\mathbf{x}_{t,n};\boldsymbol{\theta})$$

for $\mathbf{x}_0$ that represents a sequence of random variables $\mathbf{x}_0 = (\boldsymbol{x}_{0,1},\ldots,\boldsymbol{x}_{0,N})$, where the λ_t also represents the reweighting term $w(t)$ in Equation (6).

B Algorithms for Training and Inference

The detailed algorithms for training and inference are illustrated in Algorithm 1 and 2, respectively. For conditional training and inference, we split $\mathbf{x}$ into $[\mathbf{x}^{\text{src}};\mathbf{x}^{\text{tgt}}]$ and freeze the condition part $\mathbf{x}^{\text{src}}$ during training and inference.

Algorithm 1 Training MDM

```
Input: neural network $f(\cdot;\boldsymbol{\theta})$, data distribution $p_{\text{data}}(\mathbf{x}_{0,1:N})$, a custom sequence reweighting term $w(t)$, token reweighting parameters $\alpha$ and $\gamma$, timesteps $T$.
Output: model parameters $\boldsymbol{\theta}$.
repeat
    Draw $\mathbf{x}_{0,1:N} \sim p_{\text{data}}(\mathbf{x}_{0,1:N})$;
    Draw $t \in \text{Uniform}(\{1,\ldots,T\})$;
    Draw $\mathbf{x}_t \sim q(\mathbf{x}_t|\mathbf{x}_0)$;
    for $n = 1,2,\ldots,N$ do
        Let $u(\mathbf{x}_0,\mathbf{x}_t,n;\boldsymbol{\theta}) \coloneqq \mathbf{1}_{\mathrm{x_{t,n}\neq x_{0,n}}}\mathbf{x}_{0,n}^\top \log f(\mathbf{x}_t;\boldsymbol{\theta})_n$;
        Let $v(\mathbf{x}_{t,n}) = \alpha(1-\exp u(\mathbf{x}_0,\mathbf{x}_t,n;\boldsymbol{\theta}))^\gamma$;
    end for
    $L_{\boldsymbol{\theta}} = -w(t)\sum_{n=1}^{N} v(\mathbf{x}_{t,n})u(\mathbf{x}_0,\mathbf{x}_t,n;\boldsymbol{\theta})$;
    Minimize $L_{\boldsymbol{\theta}}$ with respect to $\boldsymbol{\theta}$;
until converged
```

Algorithm 2 Sampling from MDM

```
Input: trained network $f(\cdot;\boldsymbol{\theta})$, mask token id $\boldsymbol{m}$, timesteps $T$, temperature $\tau$.
Output: generated sample $\mathbf{x}_0$.
for $n = 1, 2, \ldots, N$ do
    Initialize $\mathbf{x}_{T,n} = \boldsymbol{m}$;
end for
for $t = T, \ldots, 1$ do
    Define indicator $\mathbf{e}_t = \text{TopK}\left(f(\mathbf{x}_t;\boldsymbol{\theta})\right)$ with indices in top-$t/T$ values set to 1 and others 0;
    for $n = 1, 2, \ldots, N$ do
        Draw $\widetilde{\mathbf{x}}_{0,n} \sim \text{Categorical}\left(f(\mathbf{x}_t;\boldsymbol{\theta})/\tau\right)$;
        $\mathbf{x}_{t-1,n} = \mathbf{e}_{t,n}\widetilde{\mathbf{x}}_{0,n} + (1 - \mathbf{e}_{t,n})\boldsymbol{m}$;
    end for
end for
Return $\mathbf{x}_{0,1:N}$.
```

C Additional Experimental Details

C.1 Task Details

Table 4: Dataset statistics. Minimal and CD are short for the minimal planning task and Countdown, respectively.

	Minimal	CD3	CD4	CD5	Sudoku	3-SAT 5v	3-SAT 7v	3-SAT 9v
Train Instance	50k	500k	500k	500k	100k	50k	50k	100k
Test Instance	1k	1k	1k	1k	1k	1k	1k	1k
Avg Input Token	47	11	13	16	81	245	269	305
Avg Output Token	21	16	25	35	81	9	13	17
Max Input Token	49	12	15	18	81	245	269	305
Max Output Token	23	22	35	52	81	9	13	17

We show the statistics and input-output examples on each dataset in Table 4 and Table 10, respectively.

Table 5: Model parameters with varying model size.

	Tiny	Base	Medium
Parameters	6M	85M	303M
Num of Layer	3	12	24
Num of Head	12	12	16
Hidden Dim	384	768	1024

C.2 MDM Implementation Details

We conduct all the experiments on NVIDIA V100-32G GPUs, and we use 8 GPUs for training and sampling. We mainly consider comparing diffusion and AR models trained from scratch with different model sizes, with arguments for each size listed in Table 5. We use the GPT-2 architecture for both MDM and AR. We set the learning rate to 1e-3 for the tiny model and 3e-4 for others, and we set the batch size to 1024 across all the models and tasks. We train MDM for 1200 epochs on the minimal planning task, 300 epochs on Sudoku, and 600 epochs on other datasets. By default, we set the diffusion sampling steps to $T = 20$ for tasks with average output tokens larger than 20, otherwise $T = 10$. We use a decoding temperature $\tau = 0.5$ for all tasks. For all the experiments, we have verified the statistical significance by running them multiple times.

C.3 Baseline Implementation Details

We train the AR model until convergence, and the number of training epochs is set to 200 for the minimal planning task, 300 for SAT, and 40 for others. We keep other parameters, e.g., batch size and learning rate, the same as training MDM.

For LLaMA (Touvron et al., 2023), we use LoRA fine-tuning (Hu et al., 2021) with lora rank setting to 16. We use a learning rate of 1e-4, a batch size of 256, and train for a maximum of 20 epochs to ensure the model has converged. For GPT-4, we borrow the numbers from Yao et al. (2024).

For all the diffusion baselines, we use the same transformer architecture as GPT-2 to control the variables. We set the training parameters the same as MDM, e.g., number of training epochs to 600, learning rate to 3e-4, and batch size to 1024. During inference, we set decoding timesteps to 20 for all diffusion models as we didn't observe a clear performance improvement when scaling to 1024.

D Additional Experiments

D.1 Token Consumption on Game of 24

We show the detailed accuracy and token consumption on the game of 24 in Table 6.

Table 6: Detailed accuracy and token consumption on game of 24.

	Accuracy	Prompt Tokens	Generate Tokens
GPT-4 IO	7.3	1k	18
GPT-4 CoT	4.0	2.2k	67
GPT-4 CoT-SC	9.0	2.2k	6.7k
GPT-4 ToT	74.0	1.4k	2.5k
GPT2-Scratch	18.8	11	26
MDM	76.0	11	26

D.2 AR with token reweighting

We show the accuracy of AR with the same token reweighting mechanism in Equation 8 on the minimal planning task in Table 7. We find that applying token reweighting to AR models still cannot solve subgoals with PD larger than 1 (i.e., with accuracy around 50%), similar to the original AR.

Table 7: Results of AR with token reweighting.

Planning Distance	AR	AR with token reweighting
0	100	100
1	100	100
2	51.1	52.1
3	46.9	51.5
4	52.0	50.3
5	49.9	51.9

D.3 Scaling both Data and Model Size

As an extension of Table 1, we show the accuracy of AR and MDM when both data and model size are increased in Table 8. We find scaling both data and model size is effective for both AR and MDM.

Table 8: Results of scaling both model and data size.

	AR	MDM
85M model, 500k data	45.8	91.5
303M model, 500k data	41.3	88.3
303M model, 1M data	53.3	95.6

D.4 Case Study

Table 9: Example predictions on Countdown 4. For each sub-equation, we mark the planning error in red and the calculation error in **bold**. AR exhibits more calculation errors in the last equation due to incorrect planning in the previous steps.

Numbers	Goal	Groundtruth	AR Prediction	MDM Prediction
64,36,52,42	14	64-52=12,36/12=3,42/3=14	**64/36=2**,52/2=26,**42-26=14**	64-52=12,36/12=3,42/3=14
9,73,99,75	81	75-73=2,9*2=18,99-18=81	99+75=174,**174/9=16,73+16=81**	75-73=2,9*2=18,99-18=81
2,52,20,73	57	52-20=32,32/2=16,73-16=57	2*20=40,73-52=21,**40+21=57**	52-20=32,32/2=16,73-16=57
9,80,4,5	89	9+80=89,5-4=1,89*1=89	9-5=4,4/4=1,80+1=81	9+80=89,5-4=1,89*1=89
65,2,61,22	96	65-61=4,2+22=24,4*24=96	65-61=4,22*4=88,2+88=90	65-61=4,2+22=24,4*24=96
42,47,9,14	81	47-42=5,14-5=9,9*9=81	47-42=5,14*5=70,**9+70=89**	47-42=5,14*5=70,**9+70=81**
41,4,48,20	96	41*4=164,48+20=68,164-68=96	48-41=7,20-7=13,**4*13=92**	4*20=80,**41-40=2**,48*2=96
21,36,3,42	39	42-36=6,3*6=18,21+18=39	36-21=15,15/3=5,42-5=37	42-21=21,36/3=12,**21-12=39**

We show more prediction cases of the autoregressive model and our model in Table 9.

Table 10: Task details by showing example input and output for each dataset.

Task	Input Example	Output Example
Minimal Planning	2,10/10,4/11,5/2,0/8,2/0,11/6,2/1,9/5,3/4,1-8,3	8,2/2,0/0,11/11,5/5,3
Countdown 3	15,44,79,50	44-15=29,79-29=50
Countdown 4	86,28,13,31,96	86+28=114,31-13=18,114-18=96
Countdown 5	50,36,82,44,31,51	44-36=8,82*31=2542, 8+2542=2550,2550/50=51
Sudoku	080050060 460907108 005000029 970006500 000872031 300049000 004025003 010000480 603100007	789251364 462937158 135468729 978316542 546872931 321549876 894725613 217693485 653184297
3-SAT 5v	1,4,5/1,-4,-5/2,-4,5/-1,-2,5/3,4,5/-2,-4,-5/2,3,-4/-2,-3,5/1,2,4/1,-2,3/-1,3,5/1,-2,-4/1,4,-5/1,-2,-5/1,2,-5/-1,-3,-4/-1,3,-5/-1,3,4/2,-4,-5/-1,-4,5/1,-3,-5/1,3,-5/1,-3,-4/-2,3,5/1,2,5/-1,2,-4/1,-2,4/1,-4,5/3,4,-5/-1,2,-3/1,-3,5/-2,4,5/1,-2,5/-1,2,5/1,3,-4/-1,-4,-5/-2,-3,-4/2,4,5/-2,3,-4/-3,4,5/2,-3,5	1,2,3,-4,5
3-SAT 7v	-2,-3,-7/2,-4,-7/-3,4,-5/1,2,-3/1,5,-7/-5,-6,-7/2,-5,6/2,-5,-6/-3,-4,6/-1,2,-4/-3,6,7/-2,-5,6/2,3,-7/-1,2,3/-2,3,-4/-1,3,7/1,-2,-7/2,4,6/1,2,-7/2,-3,-6/1,-2,6/-1,5,7/3,-6,-7/2,6,7/-2,-6,-7/-2,3,-5/3,5,-6/-2,6,-7/-1,-2,-7/5,-6,-7/2,-6,-7/-2,5,7/-3,-4,5/2,3,-4/-3,5,-7/3,-4,5/-2,3,-6/1,2,-6/1,4,-7/1,4,7/2,4,5/1,5,-6/1,3,4/2,3,7/1,-2,4	1,2,3,4,5,6,-7
3-SAT 9v	3,-4,-6/1,3,5/2,-7,8/1,-3,6/2,-3,-8/-4,-5,-7/1,-6,-9/1,8,-9/2,3,-9/3,-5,9/-3,7,9/-2,-3,9/-1,-5,-9/-2,-7,-9/-1,3,5/2,-5,-9/4,-7,-9/-2,3,-8/2,3,7/2,-4,6/-2,3,5/-2,-6,-8/-3,-4,-8/-2,6,7/-3,4,6/-3,-6,9/2,7,-9/2,4,-5/-3,-5,8/-4,5,-7/-4,-6,-8/2,-6,9/2,-5,9/1,4,-9/5,8,9/1,-6,7/-3,6,-9/1,4,-5/4,-6,9/-1,2,6/1,-2,-5/1,-2,-9/-4,7,9/-1,-4,-7/-3,5,-8/-1,-3,6/-2,-3,6/-3,6,9/-1,-5,8/1,-5,-9/1,4,8	1,2,3,4,-5,6,-7,-8,9