LATENT DIFFUSION FOR REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Despite the widespread adoption of large language models with hundreds of billions of parameters, these models still struggle on complex reasoning benchmarks. In this paper, we argue that the autoregressive nature of current language models are not suited for reasoning due to fundamental limitations, and that reasoning requires slow accumulation of knowledge through time. We show that combining latent diffusion models with an encoder-decoder transformer architecture provides a scalable way to address some of the fundamental shortcomings posed by autoregressive models. Diffusion models can arrive at predictions through many forward passes in latent space and its reasoning is not handicapped by the order of the tokens in the dataset. Through our experiments, we show that latent diffusion language models is a feasible approach towards scalable language models that have general complex reasoning abilities.

1 INTRODUCTION

In recent years, autoregressive large language models have become the de-facto for natural language generation. (Team et al., 2024; Radford et al., 2019) The excellent scalability of transformers combined with the availability of large datasets has led to many practical applications where language models elicit impressive emergent capabilities. However, even the biggest corporate LLMs still struggle with complex reasoning benchmarks. Prior work has shown that LLMs are limited by its autoregressive nature because the FLOPs used to generate each token is constant regardless of the difficulty of the token. Additionally, the model has to generate tokens in the order of the dataset it is trained on and therefore cannot solve easier subproblems first. (Bachmann & Nagarajan, 2024) The model will not explicitly generate easier reasoning chains first unless explicitly fine tuned to do so on a subset of tasks.

Numerous approaches have tried to tackle this problem such as Chain-of-Thought (CoT) prompting (Wei et al., 2022), enhancing models with longer context (Dai et al., 2019), or encoding recurrence into transformers (Hutchins et al., 2022; Bulatov et al., 2022). These approaches, however, are not a general solution to these shortcomings because pretraining datasets are rarely CoT prompted, and compute allocated to each token is still constant. This is a fundamental limitation when solving math problems because not all tokens have equal difficulty, and often times the answer to easier subproblems lead to better answers for harder subproblems (e.g. geometry, algebra). Even though recurrent models can perform many forward passes in latent space, prior work has not been able to scale efficiently due to its memory requirements and it has been observed that long unrolls lead to exploding or vanishing gradients. (Vicol et al., 2021)

In this paper, we propose to combine latent diffusion models (LDM) with encoder-decoder transformers in an attempt to solve the mentioned shortcomings that are posed by autoregressive large language models. In contrast to traditional LLMs, LDMs generate a latent vector by iteratively denoising gaussion noise throughout many timesteps, which intuitvely makes it more suitable for tasks that require extrapolating many facts over long horizons. Prior work has shown that text diffusion models elicit self correction abilities, where reasoning steps can be generated non sequentially and that the model learns to correct its wrong answers from easier (and correct) reasoning steps. LDMs perform reasoning in latent space which is semantically richer than discrete tokens. For a specific task, the required semantics might not be representable by its token embeddings (e.g. spatial reasoning). LDMs also do not suffer from memory requirements and instabilities

encountered by backpropagation through time (where it is common practice to set max timesteps to 1000 or 4000) (Ho et al., 2020; Nichol & Dhariwal, 2021) which makes it an appealing candidate to solve the aforementioned shortcomings. It has also been shown that latent diffusion text models outperform discrete diffusion text models, strengthening our claim that operating in latent space yield improvements over operating with discrete tokens. (He et al., 2022; Lovelace et al., 2024b)

We summarize the potential benefits of combining latent diffusion models with encoder-decoder language models for complex reasoning tasks.

1. It can do reasoning in semantic space and does not rely on discrete tokens where the accumulation of knowledge per forward pass only amounts to that particular generated token.
2. It can perform reasoning non-sequentially regardless of the order of the tokens in the training data. Throughout denoising steps, LDMs elicit self correction where correct reasoning steps lead to corrections on harder reasoning steps.
3. It does not run into memory bottlenecks and instabilities that are encountered by recurrent transformers as we scale to bigger and bigger unroll lengths.

2 Related Work

Generative Pretrained Transformers (GPT) have taken over natural language processing (and the entire world) by storm. It's remarkable scaling properties and its performance on downstream tasks has made it an indispensible tool for state-of-the-art language tasks such as translation, summarization, and instruction following. Meanwhile, image generation also had a renaissance powered by latent diffusion models. By iteratively denoising an image distribution from gaussian noise, diffusion models has been able to outperform GANs on important image generation benchmarks. Continuing research on LDMs have also found that LDMs generate more diverse image samples, and techniques such as Min-SNR-γ and progressive distillation improved the efficiency of the training and inference such that LDMs can now generate high quality images and videos at a fraction of the cost. (Rombach et al., 2022)

3 Preliminaries

3.1 Denoising Diffusion Probabilistic Models

DDPMs (Ho et al., 2020) are a class of diffusion models that iteratively construct an image from random gaussian noise. The forward process, which converts the original image into a corrupted image (by adding gaussian noise), can be formulated as $x_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$ where we sample $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ and $\bar{\alpha}_t$ which is a noise scheduling hyperparameter for different timesteps. In order to learn the reverse process to reconstruct the denoised image, a neural network is learned to predict the noise at each timestep, which is optimized by minimizing the loss $\mathcal{L}_{\text{simple}}(\theta) = \|\epsilon_\theta(x_t, t) - \epsilon\|_2^2$. During each iteration of inference, random gaussian noise can then be turned into an image according to a target data distribution by iteratively removing $\epsilon_\theta(x_t, t)$. For each sampling step, the denoised image for the next step in given by $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t\mathbf{z}$ where $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$.

3.2 Improved Denoising Diffusion Probabilistic Models

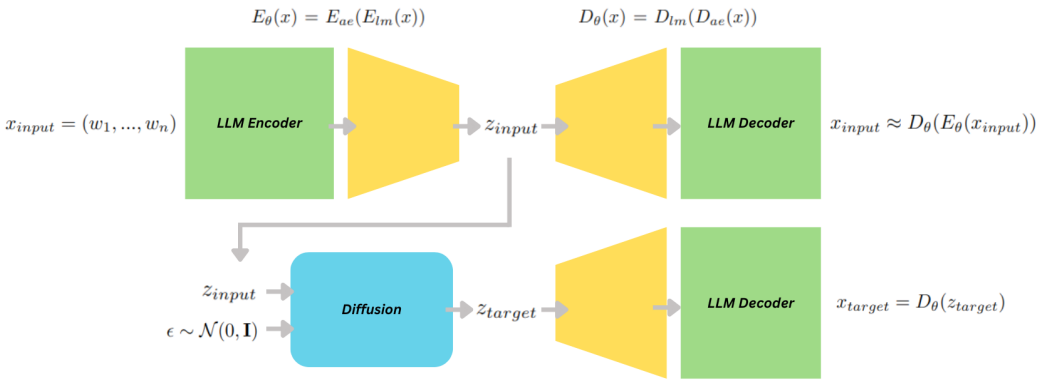
Improved DDPM (Nichol & Dhariwal, 2021) proposes various improvements to training DDPM. First, the original linear noise schedule for $\bar{\alpha}_t$ is replaced by the cosine noise schedule, which has shown to denoise the sample more evenly throughout timesteps, instead of disproportionately towards the end of the sampling timesteps. Second, they propose to have the model learn the variance using the variational lower bound instead of fixing it to σ_t^2 to improve the log likelihood of the samples. Since variances have small magnitude, the variances are learned as an interpolation between

β_{min} and β_{max} using the equation $\Sigma_\theta\left(x_t, t\right) = \exp\left(v \log \beta_t + (1-v) \log \tilde{\beta}_t\right)$ where the loss is $\mathcal{L}(\theta) = -\log p(x_0|x_1) + \sum_t \mathcal{D}_{KL}(q^*(x_{t-1}|x_t, x_0) \| p_\theta(x_{t-1}|x_t))$

3.3 Scalable Diffusion Models with Transformers

Diffusion Transformers (DiT) is a variant of transformer that has been modified to incorporate conditioning information for diffusion. Conditional image generation can be formulated as $p_\theta(x|c)$ where c is additional information (such as the class of an image). The DiT architecture consists of adaLN-zero blocks which incorporates conditioning information by regressing dimension-wise γ (scale) and β (shift) parameters from the sum of the embedding vectors of t and c for adaptive layer normalization. In addition to adaptive layer normalization, DiT also regress dimension-wise α (scaling) parameters that are added prior to any residual connections. They further initialize all MLP to output the zero-vector for α since this initializes the residual block to an identity block which leads to faster convergence. DiT empirically outperforms traditional U-Net as backbone for diffusion.

4 Methods



In this paper, we propose to merge the encoder-decoder language model with latent diffusion models in an attempt to enhance reasoning in natural language processing. We use pretrained encoder-decoder LLMs as our base model since these LLMs already contain high level semantics that have been learned from large corpus of text. We use BART extensively throughout our experiments to obtain its encoder representations. In constrast to next sequence prediction, the decoder is fine tuned to generate the original sequence given the encoder representation.

The main training consists of two stages. First, we fine-tune the decoder and an autoencoder such that the variable length encoder representation can be compressed to a fixed length latent, which can then be decoded back to its original token sequenceq. Second, we train a diffusion model such that the diffusion transformer denoises the target sequence compressed latent conditioned on the input sequence compressed latent.

During inference, the input sequence is fed into the encoder and autoencoder to obtain the compressed input latent which is then used to condition the diffusion transformer to generate a compressed target latent, which is then passed through the autoencoder and decoder to generate the predicted target sequence.

4.1 Latent Models Fine-tuning

Formally, we define a sequence of tokens as $x = (w_1, ..., w_n)$ and $(x_{input}, x_{target}) \sim \mathcal{D}$. Then, we aim to learn a language autoencoder such that $x \approx D_\theta(E_\theta(x))$. In our setting, $E_\theta(x) =$

$E_{ae}(E_{lm}(x))$, and $D_\theta(x) = D_{lm}(D_{ae}(x))$. Since changes in the dimensionality of the latent can lead to drastic changes in final performance, (He et al., 2022; Nichol & Dhariwal, 2021) we compress the encoder representation to a fixed latent space with $l_{ae} = 16$ and $d_{ae} = 256$. The autoencoder architecture consists of only cross attention transformer blocks where first block queries are learned from learnable hyperparameters of the target dimensions, and key and values are learned from the encoder representation or compressed representations. Our goal is to have the compressed latents contain both low level features and high level semantics instead of simply compressing the token sequences, so we freeze E_{model} during all training stages.

4.2 Latent Diffusion

The second stage simply consists of learning a diffusion model in the latent space learned by the autoencoder. The compressed latents with $l_{ae} = 16$ and $d_{ae} = 256$ are then projected up to d_{proj} which is then reshaped into $l_{diffusion}$ and $d_{diffusion} = 768$. Since DiT scales with decreasing patch size (or increasing sequence length), we ablate sequence length for DiT to determine whether the same scaling law holds for latent text diffusion. We follow standard DDPM approach to train x_0 and train Σ_θ with the full $\mathcal{L}$. In preliminary experiments, we observed that predicting x_θ instead of ϵ_θ produces more coherent generations, and that pretrained encoder-decoder transformers are not sensitive to small pertubations of encoder representations. We use a cosine schedule with $T = 1000$ since higher T improves the log-likelihood, and that we can always sample more efficiently using different samplers at the expense of sample quality. (Nichol & Dhariwal, 2021; Peebles & Xie, 2023)

4.3 Improvements

In addition to using diffusion to predict the target tokens, we could alternatively concatenate the input token sequences or representation to the decoder input to allow the decoder to do additional inference. If we add noise to the encoder representation during training, it learns whether to discard some of its inputs, teaching it that the diffusion output contains useful semantics but are not always reliable. Alternatively, we could also use encoders trained with contrastive learning to improve the expressibility of our diffused representations. This allows the architecture to retain GPT performance while being able to solve additional reasoning tasks with diffusion output. (Lovelace et al., 2024a) We opt to only give diffusion output to the decoder since the performance improvements would also depend on the decoder (a.k.a autoregressive language model) which would not reflect the capabilities of diffusion.

5 Results

Throughout our experiments, we study the potential benefits and the scalability of our approach to augment encoder-decoder LLMs with diffusion to enhance reasoning. We ablate different architecture variants, diffusion sequence length, model layers, and pretrained LLMs to determine the best architecture for scaling. We then summarize our findings to provide insights on how to scale this hybrid architecture.

5.1 Mock Spatial Reasoning

To study the benefits of latent diffusion augmented LLMs against conventional LLMs, we create a mock spatial reasoning problem where 4 numbers are presented as input and the model is tasked with coming up with the answer as the first token and subsequent reasoning. The reasoning consists of rotations of *up* $\rightarrow$ *down* $\rightarrow$ *left* $\rightarrow$ *right* $\rightarrow$ *up*. Initially, we start with *up*, then rotate n times where n is the first number, and each subsequent number reverses the direction of the rotation. For example, given input *1 3*, the output should be ***left*** *up down up right* where ***left*** is the final answer. We first start with *up*, then rotate one time to *down*, then reverse direction and rotate 3 times to *left*.

The problem consists of easy reasoning chains which is required for the first token, however, coming up with the first token directly is nearly impossible. In practice, reasoning might not be representable by tokens but the model could still rely on high level semantics learned by the encoder-decoder.

Architecture	Accuracy↑
Latent Diffusion (T=500)	90.4
Latent Diffusion (T=1000)	92.3
Latent Diffusion (T=4000)	89.5
Encoder-Decoder	0.0

The Encoder-Decoder performed as good as the random baseline throughout training, and predicted the end of text token as the answer near the end of training. We show from the results that latent diffusion does not have to rely on the order of the dataset and can do easy reasoning chains to extrapolate harder answers. Many hard reasoning problems in the real world are impractical to be represented by chain-of-thought tokens, therefore, doing reasoning in latent space could be a promising alternative. Augmenting latent diffusion also has an additional benefit when there is many repetition in the reasoning chain. For example, if reasoning requires many multiplication arithmetics, diffusion is able to reuse its multiplication logic neurons throughout many timesteps whereas autoregressive models can only use it once to produce a token unless there are many duplicate multiplication logic neurons in its layers which takes up model capacity.

5.2 Single Digit Additions

To further analyze the performance of this hybrid architecture with downstream math tasks, we created single digit addition problem sets where 3-5 single digit numbers are added together. Chain-of-thought reasoning chains are provided as the target where the model is trained to iteratively add the first two digits. The model is required to output the first token as answer along with its subsequent reasoning.

Architecture for single digit additions	Accuracy↑
Latent Diffusion (T=500)	97.19
Latent Diffusion (T=1000)	96.68
Latent Diffusion (T=4000)	97.27
Encoder-Decoder (First token as answer)	1.31
Encoder-Decoder (Last token as answer)	0.34

We further studied its performance by testing out different arithmetic tasks that mirrors the experiments done for GPT-3. We found out that latent diffusion performs remarkably well for its given model size, however, we acknowledge that this might not be a fair comparison because GPT-3 is not fine tuned for arithmetic tasks.

Architecture for two double digit addition	Accuracy↑
Latent Diffusion (T=1000, 140M)	87.18
GPT-3 (400M)	5.00
GPT-3 (13B)	57.00
GPT-3 (175B)	99.00

Architecture for two double digit addition	Accuracy↑
Latent Diffusion (T=1000, 140M)	

Given the same number of iterations, we show that our hybrid architecture learned the task while the encoder-decoder architecture was unable. Predicting the first token as answer lead to worse performance for the encoder-decoder because it is unable to self correct after giving an incorrect answer and have to give subsequent reasoning based on a wrong answer. This is an advantage of diffusion because pretraning data that are scraped from the internet are rarely well behaved.

5.3 Ablations

We first ablate different architectures to incorporate input text conditioning since text latents could have different properties compared to image class labels. Although cross attention yields the best results, it could simply be due to additional parameters in the cross attention module. Throughout experiments, we found that in context conditioning has minimal compute overhead while having negligible difference on BertScore, hence we adopt in context conditioning for most of the ablations.

We use Common Crawl (C4) for all of the ablation experiments since it includes a variety of different sequences from most domains.

Conditioning	BertScore↑	Perplexity↓
In Context	67.9	662.73
Cross Attention	68.3	425.83
AdaLN-Zero	68.2	453.70

In Context - We concatenate the noised target representation sequence with the input representation sequence. At the end, the sequence is split into two sequences where the first sequence is the model output, and the second sequence is the predicted variance.

Cross Attention - An additional cross attention module is added after self attention for each DiT block to incorporate input text conditioning.

AdaLN-Zero - Input text conditioning information is incorporated by adding to the timestep representation which is fed into the AdaLN-Zero block similar to the DiT architecture for class labels.

We further observe improved performance with increasing depth. However, we observe a weak negative correlation between the metrics and loss with increasing diffusion sequence length which is in contrast to image diffusion transformers. This suggests that further architecture improvements can be made or scaling should be done through increasing layers and not sequence length. We further observe that BertScore does not always correlate with Perplexity, which leads us to hypothesize that the loss function that optimizes representation could sacrifice coherence for semantic similarity, hence we use BertScore as the main metric for determining performance since it also correlates better with loss whereas perplexity has very high variance and depends significantly on the target sequence length and architecture. One hypothesis is that images are more parallelizable within layers since there are more independent patches whereas text data are more interdependent.

Depth (In Context)	BertScore↑	Perplexity↓
6 Layers	70.0	660.72
12 Layers	70.2	653.24
24 Layers	70.6	644.55

Sequence Length (In Context)	BertScore↑	Perplexity↓
Length 16	70.0	660.72
Length 32	70.2	838.61
Length 64	69.6	741.21

Depth (Cross Attention)	BertScore↑	Perplexity↓
6 Layers	70.2	621.17
12 Layers	70.4	669.70
24 Layers	70.4	600.57

Sequence Length (Cross Attention)	BertScore↑	Perplexity↓
Length 16	70.2	621.17
Length 32	70.0	622.25
Length 64	68.1	693.81

Depth (AdaLN-Zero)	BertScore↑	Perplexity↓
6 Layers	69.7	622.09
12 Layers	69.8	734.45
24 Layers	70.1	749.93

Sequence Length (AdaLN-Zero)	BertScore↑	Perplexity↓
Length 16	69.7	622.09
Length 32	69.0	590.55
Length 64	70.1	618.18

To further study the effects of high level semantics learned by the encoder-decoder architecture we compare the performance of BART-base (140M parameters) and BART-large (406M parameters) to determine whether the improved quality of both low and high level representations also carries over after augmenting with latent diffusion models of the same size. Our results show that diffusing better representations from pretrained weights improves BertScore and Perplexity.

Sequence Length	BertScore↑	Perplexity↓	Mauve↑
BART-base	67.64	903.31	28.16
BART-large	69.80	1168.88	7.23

Instead of compressing the last representation of the encoder, we compress the concatenation of the first and last representation of the encoder since the decoder wasn't able to recontruct the original text from only the last encoder representation of BART-large.

We've found that generation length decreases with longer training times which led to uncomparable BertScores since longer constrasting sentences have higher BertScores on average and that shorter sentences have lower Perplexity on average. One hypothesis is that the diffusion model only denoised signals for earlier tokens because earlier tokens have lower variance (e.g. the next token is easier to predict than the 16th token) leading to later positions denoised as paddings. Since experiments are trained with different hyperparameters with different learned generation lengths, metrics cannot be compared between different ablations. Further research on how to better evaluate these variable length diffusion models will be required to improve the reliability of current metrics.

6 Implementation Details

Throughout our experiments, we adopt classifer-free guidance to improve sample quality at the expense of sample diversity. We also use Min-SNR-γ since it is able to improve the training efficiency by many folds.

6.1 Classifier-free Guidance

Classifier-free guidance is widely known to improve sample quality. (Ho & Salimans, 2022; Nichol et al., 2021) By jointly training the unconditional $p_\theta(x)$ and conditional $p_\theta(x|c)$ model, we can sample using a linear combination of the score estimates $\tilde{\boldsymbol{\epsilon}}_\theta\left(\mathbf{z}_\lambda, \mathbf{c}\right) = (1+w)\boldsymbol{\epsilon}_\theta\left(\mathbf{z}_\lambda, \mathbf{c}\right) - w\boldsymbol{\epsilon}_\theta\left(\mathbf{z}_\lambda\right)$. This is relatively straightforward to implement by randomly setting $p(x|c = \varnothing)$ during training. Classifier-free guidance can be used to encourage the sampling procedure such that $\log p(c|x)$ is high and tradeoff between sample quality and diversity.

6.2 Min-SNR-γ

Min-SNR-γ (Hang et al., 2023) improves the training efficiency by weighing each loss term as $w_t = min\{\text{SNR}(t), \gamma\}$. By taking into account the signal-to-noise ratio (SNR), min-SNR-γ is better able to traverse the loss landscape by weighing conflicting gradients between earlier and later diffusion steps. Furthermore, Min-SNR-γ takes the minimum between SNR(t) and γ to avoid the model focusing too much on small noise levels. All training runs uses $\gamma = 5$ as our weighing strategy.

7 Discussion

It has been known that diffusion language models yield better diversity when generating text. We show from our work that augmenting latent diffusion with language models does outperform autoregressive models for certain reasoning cases. A notable limitation of diffusion is that it is

relatively inefficient to train compared to conventional language models. One hypothesis is that the possible sequences diverge as more tokens are diffused at once, hence the gradients are not as well behaved. As research on diffusion continues, we should expect that it will play a more prominent role in natural language processing to address some of our current limitations.

An exciting research direction would be to utilize this architecture for idea generation while implementing the ideas with the same architecture specialized for coding. This could be a feasible approach towards ASI because it has more diverse ideas, and it's able to directly implement the programs by reasoning about the structure of the code in latent space beforehand. This could kickstart recursive self improvement where research in deep learning becomes more and more automated. However, due to it's inefficiencies among other unknown roadblocks, it is uncertain how far we will be able to practically scale such an architecture with current hardware and algorithm.

8 Conclusion

Reasoning requires extrapolation of many facts over long horizons. In this paper, we demonstrate that augmenting latent diffusion with encoder-decoder architectures does outperform autoregressive language models for cases where tokens have different difficulty, and that following the sequential order of the dataset is not beneficial to accuracy. We believe this architecture is a viable option for solving real world reasoning tasks by reasoning in latent space. To our knowledge, this is the first research that looks into augmenting latent diffusion for reasoning tasks. As research on diffusion models continue to close the gap with autoregressive models, we are hopeful that this new architecture can achieve human level reasoning through scale and further advancements.

References

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

Aydar Bulatov, Yury Kuratov, and Mikhail Burtsev. Recurrent memory transformer. *Advances in Neural Information Processing Systems*, 35:11079–11091, 2022.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

Tiankai Hang, Shuyang Gu, Chen Li, Jianmin Bao, Dong Chen, Han Hu, Xin Geng, and Baining Guo. Efficient diffusion training via min-snr weighting strategy. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7441–7451, 2023.

Zhengfu He, Tianxiang Sun, Kuanning Wang, Xuanjing Huang, and Xipeng Qiu. Diffusionbert: Improving generative masked language models with diffusion models. *arXiv preprint arXiv:2211.15029*, 2022.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

DeLesley Hutchins, Imanol Schlag, Yuhuai Wu, Ethan Dyer, and Behnam Neyshabur. Block-recurrent transformers. *Advances in neural information processing systems*, 35:33248–33261, 2022.

Justin Lovelace, Varsha Kishore, Yiwei Chen, and Kilian Q Weinberger. Diffusion guided language modeling. *arXiv preprint arXiv:2408.04220*, 2024a.

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

Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pp. 8162–8171. PMLR, 2021.

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.

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.

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.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

Paul Vicol, Luke Metz, and Jascha Sohl-Dickstein. Unbiased gradient estimation in unrolled computation graphs with persistent evolution strategies. In *International Conference on Machine Learning*, pp. 10553–10563. PMLR, 2021.

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, 2022.

A Training Hyperparameters

A.1 Ablating Architecture

Hyperparameters	In-Context	Cross-Attention	AdaLN-Zero
Diffusion sequence length	16 (each input)	32	32
Depth		12	
Batch size		128	
Sequence Length		64	
Latents sequence length		8	
Latents dim		256	
Hidden size		768	
Number of heads		12	
Total timesteps (T)		1000	
Learning rate		1e-4	
Iterations		200k	
Floating Point		float32	

A.2 ABLATING DEPTH

Hyperparameters	6 Layers	12 Layers	24 Layers
Layers	6	12	24
Architecture		In Context	
Diffusion sequence length		16	
Batch size		128	
Sequence Length		128	
Latents sequence length		16	
Latents dim		256	
Hidden size		768	
Number of heads		12	
Total timesteps (T)		1000	
Learning rate		1e-4	
Iterations		200k	
Floating Point		float16	

A.3 ABLATING DIFFUSION SEQUENCE LENGTH

Hyperparameters	In Context 16	In Context 32	In Context 64
Diffusion sequence length	16	32	64
Batch size		128	
Sequence Length		128	
Latents sequence length		16	
Latents dim		256	
Hidden size		768	
Number of heads		12	
Layers		6	
Total timesteps (T)		1000	
Learning rate		1e-4	
Iterations		200k	
Floating Point		float16	

A.4 ABLATING ENCODER REPRESENTATIONS

Hyperparameters	BART-base	T5-large
Diffusion sequence length	64	
Batch size	128	
Sequence Length	128	
Latents sequence length	16	
Latents dim	256	
Hidden size	768	
Number of heads	12	
Layers	18	
Total timesteps (T)	1000	
Learning rate	1e-4	
Iterations	200k	
Floating Point	bfloat16	

A.5 MOCK SPATIAL REASONING

Hyperparameters	LD (T=500)	LD (T=1000)	LD (T=4000)	Encoder-Decoder
Diffusion sequence length		16		-
Sequence Length		128		-
Latents sequence length		16		-
Latents dim		256		-
Hidden size		768		-
Number of heads		12		-
Layers		24		-
Total timesteps (T)	500	1000	4000	-
Pretrained Encoder-Decoder	BART-base	BART-base	BART-base	BART-base
Batch size	128	128	128	128
Learning rate	1e-4	1e-4	1e-4	1e-4
Iterations	500k	500k	500k	500k
Floating Point	bfloat16	bfloat16	bfloat16	bfloat16

A.6 BIG TRAINING RUN

Hyperparameters	Main Run
Architecture	Cross Attention
Diffusion sequence length	32
Sequence Length	128
Latents sequence length	16
Latents dim	256
Hidden size	768
Number of heads	12
Layers	24
Total timesteps (T)	1000
Pretrained Encoder-Decoder	BART-base
Batch size	128
Learning rate	1e-4
Floating Point	bfloat16

A.7 MULTISTEP ADDITION

Hyperparameters	LD (T=500)	LD (T=1000)	LD (T=4000)	Encoder-Decoder
Diffusion sequence length		16		-
Sequence Length		128		-
Latents sequence length		16		-
Latents dim		256		-
Hidden size		768		-
Number of heads		12		-
Layers		24		-
Total timesteps (T)	500	1000	4000	-
Pretrained Encoder-Decoder	BART-base	BART-base	BART-base	BART-base
Batch size	128	128	128	128
Learning rate	1e-4	1e-4	1e-4	1e-4
Iterations	500k	500k	500k	500k
Floating Point	bfloat16	bfloat16	bfloat16	bfloat16