Unroll Autoregressive Generation via Next-N-Token Prediction

Anonymous ACL submission

Abstract

Most of current large language models (LLMs) based on next-token prediction suffer from the failures both in teacher-forcing training and autoregressive inference. Although non-autoregressive approaches offer alternative solutions to mitigate these problems, the difficulty in model inference and enormous cost for long text generation greatly impedes the application in tasks a LLM is good at. We here present a framework to predict next n tokens at once, which bridges the gap between autoregressive and non-autoregressive generation. According to this framework, we propose to exploit multiple identical mask tokens appended after input context together with a novel mask recipe called future-aware self-attention mask for generation. Using this method, we finetune the pretrained models of Qwen2 series and evaluate the derived models on five benchmarks. Our finetuned model evidently surpasses those trained using two existing methods under the same condition. We also verify the great potential of our method in unrolling the autoregressive generation and discuss several directions for further improvement.

1 Introduction

In recent years, large language models (LLMs) profoundly advance the application of text generation, ranging from text summarization (Lewis et al., 2020; Zhang et al., 2020; Du et al., 2022) and question-answering (Brown et al., 2020; Raffel et al., 2020; Touvron et al., 2023) to code completion (Chen et al., 2021; Guo et al., 2022) and mathematical problem solving (Touvron et al., 2023; Frieder et al., 2024; OpenAI et al., 2024). Nearly all of these distinguished LLMs such as GPT-3 (Brown et al., 2020), LLaMA (Touvron et al., 2023) and Qwen2 (Yang et al., 2024) take **n**ext-**t**oken **p**rediction (NTP) as the typical training task and inference method. However, several types of failures induced from NTP cannot be neglected despite of the superior performance of these LLMs on many downstream tasks (Bachmann and Nagarajan, 2024). Consequently, it is necessary and meaningful for generative LLMs to explore new approaches that essentially alter the conventional methods of training and inference.

There are actually plenty of related work focusing on non-autoregressive generation to avoid the disadvantages of NTP-based approaches. One popular kind is to exploit diffusion process in continuous or discrete space to reconstruct the real sentence from a corrupted sequence via iterative denoising (Li et al., 2023), such as D3PM (Austin et al., 2021), Diffusion-LM (Li et al., 2022), DiffusionBERT (He et al., 2023) and AR-Diffusion (Wu et al., 2023). Other methods employ an autoencoder or an energy function to generate an expected sentence iteratively (Ghazvininejad et al., 2019; Savinov et al., 2022; Mireshghallah et al., 2022), theoretically different from predicting the next token conditioning on its preceding context. Although effective in certain settings, the difficulty of determining the length of target sequence dynamically and enormous computational cost for long text generation impede the application of these methods. Despite all this, non-autoregressive methods still inspire us to pursue more efficient and effective way by generating multiple tokens in parallel.

In fact, predicting multiple tokens from one pass has been reinvestigated recently to fulfill the need of inference speedup. Stern et al. (2018) proposed blockwise parallel decoding by predicting future tokens at once and then verifying them in parallel. Since the emergence of speculative decoding (Leviathan et al., 2023; Chen et al., 2023), a lot of researches focus on more practicable and efficient means to produce future candidates for the target LLM, such as predicting from different decoding heads (Cai et al., 2024; Gloeckle et al., 2024) or though look-ahead embeddings (PaSS) (Monea

et al., 2023). However, most of these methods consider multi-token prediction as an auxiliary means to speed up the generation of NTP-based models, instead of mitigating NTP-inherent failures.

Combining these excellent ideas, we formalize the theoretical framework of **n**ext-**n**-**t**oken **p**rediction (NNTP). As the term suggests, NNTP is just a generalized version of classical NTP to produce multiple tokens simultaneously (e.g., Figure 1). As in Bachmann and Nagarajan's (2024) work, we propose to exploit multiple identical mask tokens appended after input context, together with a delicate mask recipe called future-aware self-attention mask, to generate the continuation. We then finetune several pretrained models of Qwen2 series using different configurations of certain hyper-parameters through this method, and evaluate the derived models on five benchmarks. We firstly choose the best configuration and verify the great potential of our method in unrolling the autoregressive generation in the meantime. Then we finetune the Qwen2 7B model using the best configuration and compare our method with others. Although our finetuned models exhibits slightly inferior performance than the original, we still observe evident improvement compared to two existing methods under the same condition. Our method significantly surpasses the work of Gloeckle et al. (2024) with linear output heads by 0.059 accuracy increase on MMLU, 0.185 on HellaSwag, 0.046 on WinoGrande, 0.058 on RACE and 47.8 perplexity decrease on LAMBADA, and slightly outperforms PaSS (Monea et al., 2023) by 0.077 accuracy increase on HellaSwag, 0.011 on RACE and 1.93 perplexity decrease on LAMBADA. Finally, we analyze the error distribution over positions corresponding to newly-generated tokens and discuss several directions for further improvement.

Although predicting multiple tokens has been studied before (Cai et al., 2024; Gloeckle et al., 2024; Monea et al., 2023), the present work offers the following contributions by conducting a thorough investigation of NNTP at both training and inference time.

1. We present a framework of NNTP to predict n subsequent tokens simultaneously by summarizing previous work and explain three features of this framework;

2. For the NNTP framework, we propose to exploit identical mask tokens together with a novel mask recipe for generation and validate its effectiveness on five benchmarks;

3. The potential in unrolling the process of autoregressive generation is clearly verified and a simple yet effective technique is unveiled to improve the ultimate performance of our method.

We will introduce the framework of NNTP and a new method based on it, and then present experimental results in following sections.

2 Method

2.1 Framework of Next-N-Token Prediction

With the help of a language model, we can fundamentally estimate the likelihood of an arbitrary sentence through the token sequence encoded from it. Given the sequence $\mathbf{x} = \{x_1, x_2, \ldots, x_L\}$ of a sentence, the joint probability can be typically factorized into the product of conditional probabilities of next token via the chain rule

$$p(\mathbf{x}) = \prod_{i=0}^{L-1} p(x_{i+1}|\mathbf{x}_{\leq i}), \tag{1}$$

where $\mathbf{x}_{\leq i}$ denotes the prefix sequence of token x_{i+1}. According to this formulation, it is natural to induce the framework of **n**ext-**t**oken **p**rediction (NTP), which predicts a subsequent token conditioning on its context at every step. So far, almost all of causal LLMs are built upon NTP via teacher-forcing training and autoregressive inference (Bachmann and Nagarajan, 2024). We generalize this concept and then formalize the framework of **n**ext-**n**-**t**oken **p**rediction (NNTP) by gathering inspiration of recent advances in speculative decoding and multi-token prediction (Cai et al., 2024; Gloeckle et al., 2024; Monea et al., 2023).

As implied by its name, a language model based on NNTP produces n subsequent tokens simultaneously conditioning on a given context. Supposing $\{x_1, x_2, \ldots, x_c\}$ $(c < L)$ is the given context of a complete sentence mentioned above, we formalize the language modeling via NNTP by refactorizing the joint probability

$$\begin{aligned} p(\mathbf{x}) &= p(x_1, x_2, \ldots, x_L) \\ &= p(x_1, x_2, \ldots, x_c) \\ &\quad p(x_{c+1}, \ldots, x_{c+n}|\mathbf{x}_{\leq c}) \\ &\quad p(x_{c+n+1}, \ldots, x_{c+2n}|\mathbf{x}_{\leq c+n}) \\ &\quad \cdots \\ &\quad p(x_{c+kn+1}, \ldots, x_{\min(L, c+(k+1)n)}|\mathbf{x}_{\leq c+kn}), \end{aligned} \tag{2}$$

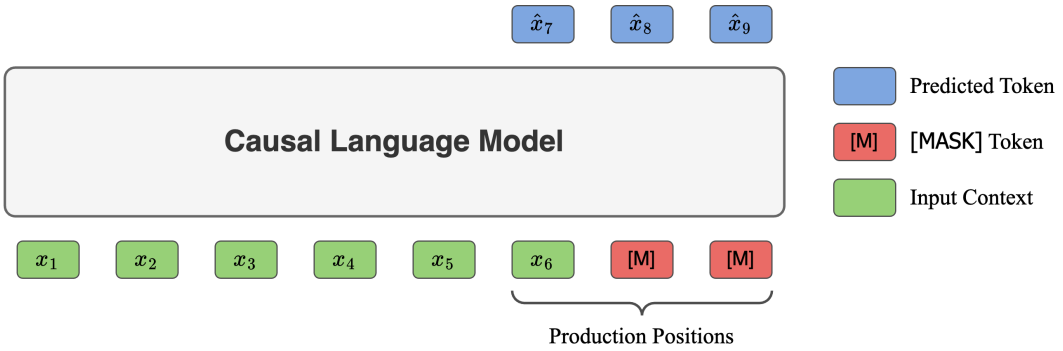

Figure 1: An illustration of a causal language model with parallel inputs for next-n-token prediction. By appending two mask tokens after input context, the model will generate three new tokens from one forward pass in this case. The position in the sequence corresponding to each new token ($\hat{x}_7$, $\hat{x}_8$, $\hat{x}_9$) is referred to as production position for brevity.

where $k+1 = \lceil \frac{L-c}{n} \rceil$, indicating the total steps to generate the whole sequence autoregressively.

We refer to the number of tokens n predicted simultaneously from a single pass of a LLM as **unrolled stride** for convenience. Accordingly, the positions corresponding to these newly-generated tokens as the continuation of input context are referred to as **production positions** for brevity (e.g., Figure 1). It is demonstrable that the steps of autoregression is inversely proportional to unrolled stride at inference. This simple correlation gradually bridges the gap between autoregressive and non-autoregressive generation by varying the unrolled stride from small to large. Moreover, since the unrolled stride during training is not required to be identical to that at inference time, we use n_{train} and n_{infer} as notations in order to distinguish one from the other. We formally conclude three exciting features from NNTP framework:

1. Although it seems apparent that a NNTP-based model produces tokens autoregressively (Eq. 2), it actually behaves like non-autoregressive generation when the inference stride n_{infer} becomes large enough;

2. A LLM trained with a large stride n_{train} can be normally used with any n_{infer} smaller than or equal to n_{train} at inference, i.e. $n_{\text{infer}} \leq n_{\text{train}}$;

3. With particularly designed architecture, a LLM can generate more tokens simultaneously than the predicted number during training, namely $n_{\text{infer}} > n_{\text{train}}$.

Note that NNTP framework only works for Transformer-based language models processing inputs parallelly (Vaswani et al., 2017), not recurrent models emitting one token at a time like LSTM (Merity et al., 2018) and Mamba (Gu and Dao, 2024).

2.2 Predict with Mask Tokens

According to the framework of NNTP, we introduce a simple modification to generative LLMs with Transformer as backbone (Vaswani et al., 2017). As depicted in Figure 1, multiple identical mask tokens are appended after the given context as inputs. The number of mask tokens is evidently determined by the unrolled stride n_{train} or n_{infer} specified beforehand. A clear difference from PaSS is that we replace all "look ahead" tokens with a same mask token (Monea et al., 2023). Such modification essentially leads to a similar model specialized for teacherless training in Bachmann and Nagarajan's (2024) work, which is also derived from PaSS (Monea et al., 2023). An incredible advantage of identical mask tokens in different production positions is that such architecture implements the third feature of NNTP framework. In other words, we can predict any number of subsequent tokens (i.e., unrestricted n_{infer}) from a forward pass, once the mask embedding has been learned in training.

2.3 Future-aware Self-Attention Mask

In addition to identical mask tokens, we also propose a novel mask recipe termed as future-aware mask for self-attention computation. Inspired by the researches of UNILMv2 (Bao et al., 2020) and GLM (Du et al., 2022), we come up with two types of future-aware mask to enable certain queries of interest to attend all future keys except for padding ones:

- **Type I** Only queries in production positions

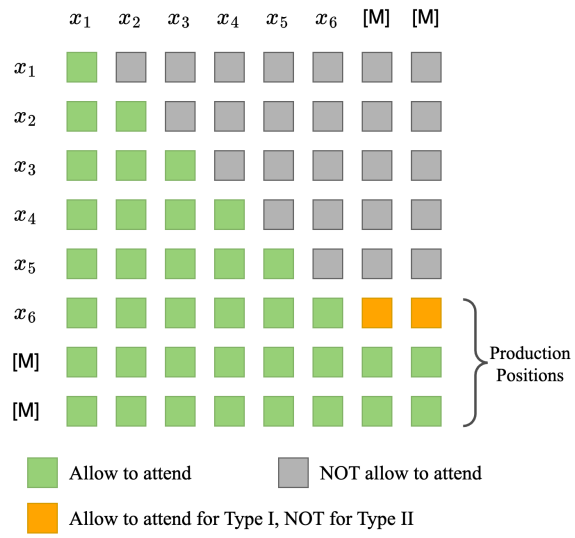

Figure 2: Two types of future-aware self-attention mask. All mask tokens can attend each other and their preceding keys (green). For future-aware mask Type I, the query in the first production position x_6 can attend all mask tokens appended after input context (orange), while on the contrary for Type II. Other queries are strictly prevented from attending future keys (grey), same as causal mask in Transformer (Vaswani et al., 2017).

(x_6 and [M] in Figure 2) can attend all future keys, while others just attend themselves or their history as in causal mask used in Transformer (Vaswani et al., 2017);

- **Type II** Slightly different from Type I, the query in the first production position (x_6 in Figure 2) only attends itself or keys preceding it.

Even though both types of future-aware mask basically share similar mask matrices added to self-attention logits before softmax, Type II makes it possible to cache the keys and values of the last context token (x_6 in Figure 2) for next inference step. Such minor improvement eventually avoids some redundant computation especially in generating very long sentence with a small unrolled stride. We use future-aware mask Type I in following experiments for the sake of better performance, unless otherwise specified.

3 Experiments

We preprocess the English Wikipedia as training corpus and then finetune different LLMs using two approaches (See Appendix A). To evaluate the performance of finetuned models, we carefully choose five benchmarks about text comprehension and continuation: MMLU (Hendrycks et al., 2021b,a), HellaSwag (Zellers et al., 2019), LAMBADA (Paperno et al., 2019), WinoGrande (Sakaguchi et al., 2019) and RACE (Lai et al., 2017). At last, experimental results are gathered and presented in following subsections.

3.1 Unrolled Stride in Training

First of all, we investigate the effect of unrolled stride during training on the ultimate performance of finetuned models. Using Qwen2 1.5B as the base model, we finetune it with the stride n_{train} varying from 2 to 16, and evaluate each derived model with inference stride $n_{\text{infer}} = 2, 4, 8$ respectively. We use future-aware mask Type I by default in this experiment. Experimental results on two kinds of tasks, HellaSwag for multiple choice question and LAMBADA for word prediction, are illustrated in Figure 3a and 3b.

As we can see, too small strides (e.g., $n_{\text{train}} = 2$) always lead to poor performance on both HellaSwag and LAMBADA under all circumstances. An interesting phenomenon is that all models finetuned with large strides consistently produce high accuracies on HellaSwag when the inference strides are small (i.e., $n_{\text{train}} > n_{\text{infer}}$). We conjecture that this is due to the ability of foreseeing future tokens beyond current production positions in every generation step, which is probably acquired from finetuning via NNTP. The abnormal trend of $n_{\text{train}} > n_{\text{infer}} = 4$ on LAMBADA may be caused by very short answers restricting the potential of NNTP (See Table 5). From Figure 3a and 3b, we can find the best performance for $n_{\text{infer}} = 2$, moderate for $n_{\text{infer}} = 4$ and the worst for $n_{\text{infer}} = 8$ using each finetuned model on both benchmarks. We leave this to next subsection for further investigation. If we intentionally keep the inference stride same as the training one, a consecutive performance drop can be observed by increasing the unrolled strides. Through a rough theoretical analysis, the explosive complexity of NNTP at training and inference time naturally takes responsibility for this degradation (See Appendix C). According to the results on HellaSwag and LAMBADA, we choose $n_{\text{train}} = 4$ for remaining experiments if not specified.

3.2 Unrolled Stride at Inference

Besides the training stride, the unrolled stride at inference may also have a significant impact on

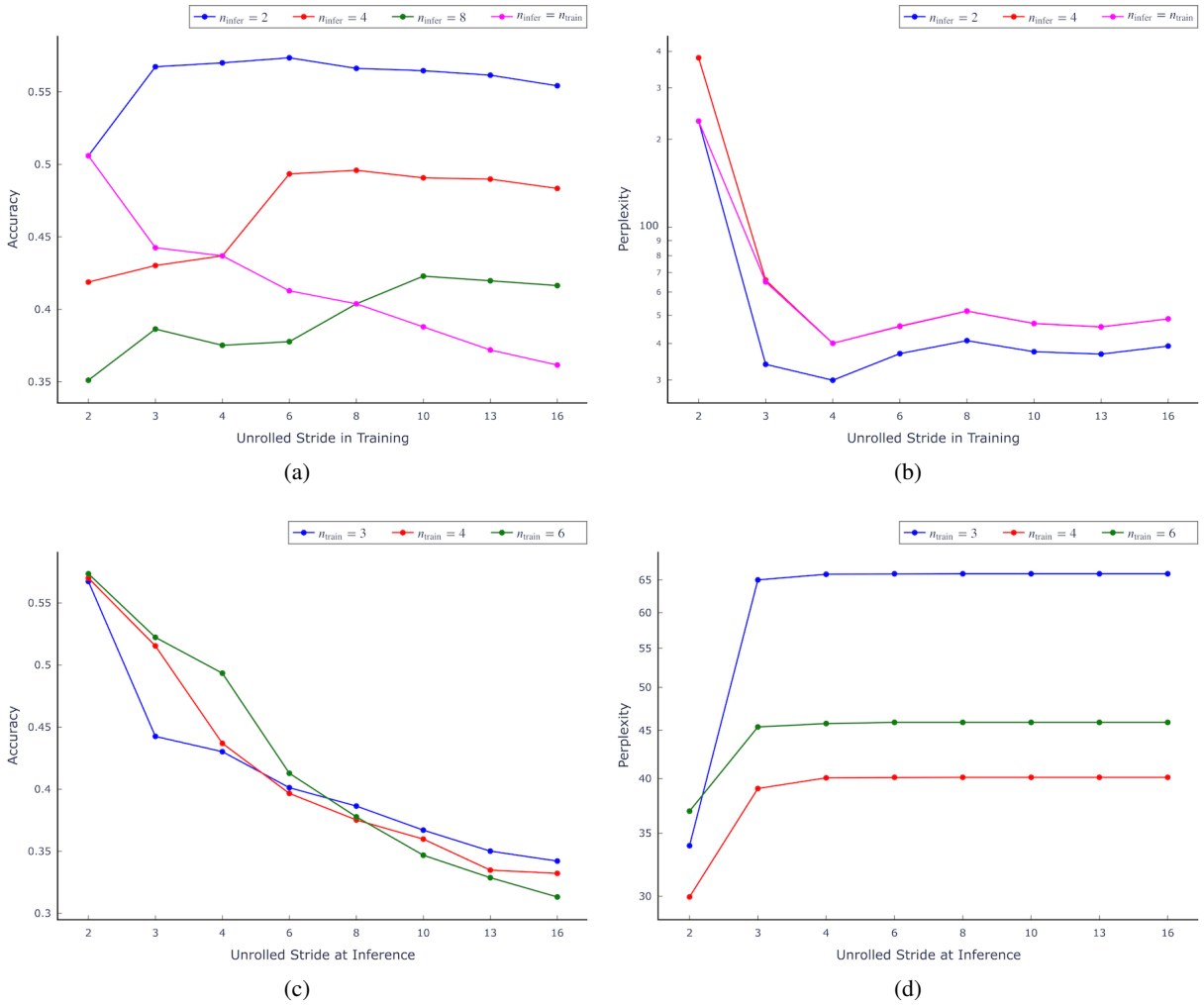

Figure 3: Evaluation results of finetuned models corresponding to different unrolled strides in training and at inference. The accuracies of derived models after finetuning Qwen2 1.5B with different training strides on HellaSwag are illustrated in (a). (b) presents the perplexity of same models as (a) on LAMBADA benchmark. (c) and (d) exhibit the performance of finetuned Qwen2 1.5B models when evaluating with different inference strides on HellaSwag and LAMBADA respectively.

the evaluation results. We still finetune the Qwen2 1.5B model with different training stride $n_{\text{train}} = 3, 4, 6$ respectively, and then evaluate each derived model with unrolled stride varying from 2 to 16. As in previous experiment, we use future-aware mask Type I as the default setting.

As depicted in Figure 3c, we discover that the accuracy on HellaSwag decreases monotonically with the increase of unrolled stride at inference. A similar trend can be observed on LAMBADA benchmark in Figure 3d, where the perplexity of generated tokens becomes worse and worse if the inference stride gets large. It is not surprising since the search space of possible outputs grows exponentially with the increasing inference stride (See Appendix C). We also find the perplexity on LAMBADA remains unchanged for $n_{\text{infer}} \geq 4$, due to small answer length of each example in this dataset (See Table 5). Specially, all of our finetuned models achieve relatively poor yet acceptable performance when the unrolled stride at inference exceeds the training one (i.e., $n_{\text{infer}} > n_{\text{train}}$). This exciting results clearly demonstrate the superiority of the third feature of NNTP framework implemented through identical mask tokens (Section 2.1). In order to examine the potential of NNTP and expect relatively better performance using finetuned models, we choose $n_{\text{infer}} = 4$ to conduct following experiments unless specified otherwise.

3.3 Future-aware Self-attention Mask

In spite of unrolled strides, the mask recipe also plays an important role in improving the effectiveness of NNTP-based language models. We fine-

	HellaSwag (acc ↑)			LAMBADA$_{std}$ (ppl ↓)		
	$n_{\text{infer}} = 2$	$n_{\text{infer}} = 4$	$n_{\text{infer}} = 8$	$n_{\text{infer}} = 2$	$n_{\text{infer}} = 4$	$n_{\text{infer}} = 8$
Future-aware Mask Type I	**0.7059**	**0.5946**	0.4732	**10.77**	**11.92**	**11.96**
Future-aware Mask Type II	0.7050	0.5909	**0.4755**	11.45	12.68	12.71
Causal Mask	0.7023	0.5823	0.4373	12.90	14.38	14.43

Table 1: Comparison results of different mask recipes. We apply each mask recipe to the base Qwen2 7B model and then finetune it with $n_{\text{train}} = 4$. As for evaluation, we also employ the same mask recipe as training for each derived model. Each model is evaluated with three different inference strides to reduce accidental bias.

tune Qwen2 7B model by employing three kinds of mask recipes: two types of future-aware mask and causal mask. The causal mask is firstly proposed in Transformer (Vaswani et al., 2017) and widely used in generative LLMs (Touvron et al., 2023; Yang et al., 2024), which only allows each query to attend its preceding keys or itself in self-attention computation. We use the same mask recipe as in finetuning when evaluating each derived model. The resulting models are evaluated on HellaSwag and LAMBADA benchmarks with different inference strides for fair comparison.

From Table 1, we can conclude that future-aware self-attention mask substantially surpasses the conventional causal mask, and future-aware mask Type I generally works slightly better than Type II. Obviously, the modification that allows queries in production positions to attend future keys evidently enhances the capability of generative LLMs. The superior performance of future-aware mask further proves the great potential of foreseeing the future when generating text via NNTP. As a consequence, we use future-aware mask Type I during training and at inference by default.

3.4 LLMs of Different Sizes

Using the chosen settings discussed in previous subsections, we experiment with LLMs of various sizes via NNTP. Based on pretrained models of Qwen2 series, we present the performance of both original and finetuned models. The evaluation results of the former on five benchmarks are marked with NTP and the later with NNTP as shown in Table 2.

There is an obvious trend that large-sized models outperform those of small sizes in terms of accuracy or perplexity on all benchmarks. It clearly verifies one aspect of the well-known "scaling law" that the performance of LLMs scales with model size (Kaplan et al., 2020). Another phenomenon is that all models finetuned via NNTP always fail to achieve comparable performance to corresponding base models without finetuning. According to Table 2, our finetuned models only reach about 97% of the original in terms of accuracy on MMLU, 75% on HellaSwag, 85% on WinoGrande and 80% on RACE. Furthermore, we discover that the performance gap between finetuned and original models correlates with the answer length of different benchmark datasets. For example, we can observe an evident degradation of our NNTP-based models on HellaSwag with longest answers (29.44 tokens in average), while our models exhibit comparable performance on MMLU with just one token as answers (Table 5). We conjecture that this is because the complexity of NNTP grows with the increasing unrolled strides at both training and inference time (See Appendix C). It is worth to note that our method essentially turns into a non-autoregressive solution for tasks requiring short answers, such as MMLU and LAMBADA. Therefore, our proposed method exhibits a promising advantage in unrolling the process of autoregressive generation.

3.5 Comparison with Other Methods

In this subsection, we compare our method with two existing methods adapted from Gloeckle et al. (2024) and Monea et al. (2023) respectively. In order to adapt to the requirement of NNTP, we reimplement both methods by changing the speculative decoding into predicting multiple tokens at once. We exploit the terms "Multi-token Heads" and "Look-ahead Tokens" to represent the major characteristic of each method and to distinguish them from ours. Note that only linear output heads are implemented in the "Multi-token Heads" model for simplicity. Using Qwen2 7B pretrained weights, we initialize and train a model through each of these methods. The evaluation of two resulting models and ours (marked with NNTP) is conducted under a same condition, that is predicting 4 tokens simultaneously from one pass. All experimental results are presented in Table 3.

From Table 3, we can see that all models fine-

	MMLU acc ↑	HellaSwag acc ↑	LAMBADA$_{std}$ ppl ↓	WinoGrande acc ↑	RACE acc ↑
Qwen2 0.5B (NTP)	0.4405	0.4906	15.38	0.5777	0.3445
Qwen2 0.5B (NNTP)	0.4247 (0.96x)	0.3614 (0.74x)	213.56	0.5296 (0.92x)	0.2670 (0.78x)
Qwen2 1.5B (NTP)	0.5508	0.6548	6.93	0.6598	0.3665
Qwen2 1.5B (NNTP)	0.5397 (0.98x)	0.4369 (0.67x)	40.07	0.5580 (0.85x)	0.2852 (0.78x)
Qwen2 7B (NTP)	0.6946	0.7882	4.67	0.7222	0.3990
Qwen2 7B (NNTP)	0.6743 (0.97x)	0.5946 (0.75x)	11.92	0.6054 (0.84x)	0.3234 (0.81x)

Table 2: Performance of Qwen2 models of different sizes before and after finetuning. We choose original models of Qwen2 series without instruction tuning as base models. The original models are marked with NTP, while those after finetuning with NNTP.

	MMLU acc ↑	HellaSwag acc ↑	LAMBADA$_{std}$ ppl ↓	WinoGrande acc ↑	RACE acc ↑
Original (NTP)	0.6946	0.7882	4.67	0.7222	0.3990
Multi-token Heads	0.6156	0.4095	59.72	0.5596	0.2651
Look-ahead Tokens	**0.6769**	0.5180	13.85	**0.6101**	0.3129
Ours (NNTP)	0.6743	**0.5946**	**11.92**	0.6054	**0.3234**
Ours (NTP)	0.6743	0.7743	4.35	0.7088	0.3876
Ours (NNTP + NTP)	0.6743	0.7783	4.27	0.6993	0.4010

Table 3: Comparison results of different methods. The original is Qwen2 7B model without instruction tuning, which serves as the base model for comparing other methods. "Multi-token Heads" and "Look-ahead Tokens" represent two existing methods adapted from Gloeckle et al. (2024) and Monea et al. (2023) respectively. Our finetuned model is particularly evaluated via three different ways, including next-token prediction (marked with NTP), next-n-token prediction (NNTP) and using dynamic strides in decoding (NNTP + NTP). Three models in the middle rows are actually finetuned and evaluated using the same setting, i.e., predicting 4 future tokens simultaneously.

tuned and evaluated via NNTP are inferior to the original model using NTP on five benchmarks, including those trained through "Multi-token Heads" and "Look-ahead Tokens". It is quite reasonable since NNTP is inherently more complex than NTP in both stages of training and inference (See Appendix C). Regardless of such inferiority for now, our method clearly outperforms other two methods, especially on HellaSwag. "Multi-token Heads" performs much worse than "Look-ahead Tokens" and our method on all benchmarks, which may caused by the lack of dependence among different output heads of linear type. Contrarily, "Look-ahead Tokens" can capture the dependence among tokens in production positions partly through causal mask, and ours through future-aware self-attention mask. Moreover, our method obviously surpasses two others by implementing the third feature of NNTP framework, which is predicting any number of tokens from one pass at inference (Section 2.1).

We further investigate the underlying reasons responsible for the degradation of NNTP-based models. Using the finetuned model via NNTP, we firstly evaluate it by autoregressively predicting next token (marked with NTP). Inspired by the idea of speculative decoding, we then use dynamic unrolled stride at inference: (1) predicting n subsequent tokens simultaneously as a candidate sequence at each step; (2) accepting the first token (or first few tokens) in the candidate sequence and feeding it together with preceding context to the model for next step. The results of such decoding are marked with "NNTP + NTP" for convenience.

According to Table 3, our model (NTP) exhibits slightly inferior performance than the original if decoding via NTP. Such phenomenon partly demonstrates that NNTP is a much harder task for training than NTP, resulting in a less effective next-token predictor than before finetuning. The contrary trend on LAMBADA may be caused by short answers in that dataset (See Appendix D). In spite of the performance drop induced from training, we still observe more severe degradation if changing the decoding method via NTP to NNTP using the same model (0.7743 to 0.5946 on HellaSwag, 4.35 to 11.92 on LAMBADA, 0.7088 to 0.6054 on Wino-

Grande and 0.3876 to 0.3234 on RACE). This fact indicates that the decoding method becomes the major bottleneck for development of our NNTP-based method. At last, we experiment with a novel decoding method using dynamic unrolled stride at inference (NNTP + NTP), and surprisingly achieve a slightly better results than the plain decoding via NTP. Once again, this explicitly verifies the benefit of foreseeing future tokens in every decoding step, which is learned from NNTP.

4 Related work

Plenty of researches have explored diverse means to predict multiple tokens simultaneously, mostly under a condition of autoregressive inference. Stern et al. (2018) proposed blockwise parallel decoding by predicting future tokens at once and then verifying them in parallel. After the emergence of speculative decoding (Leviathan et al., 2023; Chen et al., 2023), Cai et al. (2024) and Gloeckle et al. (2024) improved the efficiency of autoregressive generation using future candidates produced from different decoding heads from one pass. Furthermore, parallel speculative sampling (PaSS) exploited additional look-ahead embeddings to speed up decoding (Monea et al., 2023). Derived from PaSS, Bachmann and Nagarajan (2024) repeated the same special token '$' many times for teacherless training, which happens to accord with our intuition of identical mask tokens for all look-ahead positions. Our method differs from the work mentioned above in two aspects: 1) auxiliary or draft models are not required to verify the future tokens predicted simultaneously; 2) a novel mask recipe instead of vanilla causal mask is employed when computing self-attention weights in each Transformer layer. Beyond all these, we intend to mitigate the NTP-inherent failures by unrolling the autoregressive generation via NNTP.

Many kinds of mask recipes have been devised to customize the computation of self-attention weights in Transformer-based LLMs. The most popular one is causal mask originally used in Transformer decoder (Vaswani et al., 2017). As the rising of masked language models like BERT (Devlin et al., 2019) and causal language models like GPT-2 (Radford et al., 2019), many researchers made great efforts to combine the advantages of both architectures through sophisticated self-attention mask. UNILM integrated bidirectional, unidirectional and sequence-to-sequence objectives into the unified pre-training of a Transformer-based model, each of which corresponds to a specific self-attention mask (Dong et al., 2019). UNILMv2 further employed a complicated mask for self-attention to combine autoencoding and partially autoregressive tasks together (Bao et al., 2020). GLM, from a slightly different perspective, incorporated the autoregressive objective into pre-training through a blank infilling task and a relevant mask (Du et al., 2022). Although quite similar in terms of expression, our proposed future-aware mask allows masked tokens to attend each other, which is highly concise and intuitive thanks to the lack of autoencoding task in training (See section 2.3).

5 Discussion

In this work, we formalize the framework of next-n-token prediction (NNTP), which bridges the gap between autoregressive and non-autoregressive generation. Based on existing LLMs, we then propose to exploit multiple identical mask tokens appended after input context together with a novel mask recipe called future-aware mask for generation. Through several experiments, we verify the great potential of our method in unrolling the autoregressive generation. We also demonstrate the superiority of our method compared to two others under the same condition. At last, we analyze the complexity of NNTP and investigate the error distribution over production positions.

There are several aspects concerning with NNTP and related methods to be improved in the future. The first is to sample an optimal sequence from multiple distributions of future tokens over a large vocabulary. The intuitive way of sampling each token independently may lead to an inconsistent output, such as illegal set phrase. Another aspect is to use adaptive unrolled stride at inference, similar to dynamic inference stride in Section 3.5. A trivial idea of re-generating "hard" tokens (e.g., nouns and verbs) with high uncertainty in last step may help to significantly improve the quality of generated text.

6 Limitations

Although next-n-token prediction (NNTP) exhibits great potential in unrolling the process of autoregressive generation, there still exists certain shortcomings impeding the practical application of NNTP-based methods. An obvious limitation is that NNTP-based LLMs rely on a context long

enough to generate its continuation. Fortunately, most of generative tasks usually offer an informative prompt to steer the response from a LLM, making this limitation insignificant. Another drawback lies in the inferior performance to LLMs based on next-token prediction (NTP). One major reason behind this may be the inefficiency of teacherless training adopted by NNTP-based methods, in which only a tiny portion of tokens in each example are constructed as labels for training. Additionally, due to the enormous complexity of NNTP, related methods require new techniques for both training and inference to match the performance of those based on NTP.

References

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. 2021. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems*, volume 34, pages 17981–17993. Curran Associates, Inc.

Gregor Bachmann and Vaishnavh Nagarajan. 2024. The pitfalls of next-token prediction. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 2296–2318. PMLR.

Hangbo Bao, Li Dong, Furu Wei, Wenhui Wang, Nan Yang, Xiaodong Liu, Yu Wang, Songhao Piao, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2020. Unilmv2: pseudo-masked language models for unified language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. 2024. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. In *Forty-first International Conference on Machine Learning*.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023. Accelerating large language model decoding with speculative sampling.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé, Jared Kaplan, Harrison Edwards, Yura Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 34 others. 2021. Evaluating large language models trained on code. *ArXiv*, abs/2107.03374.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *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)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. *Unified language model pre-training for natural language understanding and generation*. Curran Associates Inc., Red Hook, NY, USA.

Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. 2022. GLM: General language model pretraining with autoregressive blank infilling. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 320–335, Dublin, Ireland. Association for Computational Linguistics.

Simon Frieder, Luca Pinchetti, Ryan-Rhys Griffiths, Tommaso Salvatori, Thomas Lukasiewicz, Philipp Petersen, and Julius Berner. 2024. Mathematical capabilities of chatgpt. *Advances in neural information processing systems*, 36.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2024. A framework for few-shot language model evaluation.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Mask-predict: Parallel decoding of conditional masked language models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6112–6121, Hong Kong, China. Association for Computational Linguistics.

Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Roziere, David Lopez-Paz, and Gabriel Synnaeve. 2024. Better & faster large language models via multi-token prediction. In *Forty-first International Conference on Machine Learning*.

Aaron Gokaslan, Vanya Cohen, Ellie Pavlick, and Stefanie Tellex. 2019. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus.

Albert Gu and Tri Dao. 2024. Mamba: Linear-time sequence modeling with selective state spaces.

Daya Guo, Shuai Lu, Nan Duan, Yanlin Wang, Ming Zhou, and Jian Yin. 2022. UniXcoder: Unified cross-modal pre-training for code representation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7212–7225, Dublin, Ireland. Association for Computational Linguistics.

Zhengfu He, Tianxiang Sun, Qiong Tang, Kuanning Wang, Xuanjing Huang, and Xipeng Qiu. 2023. DiffusionBERT: Improving generative masked language models with diffusion models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4521–4534, Toronto, Canada. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Steven Basart, Andrew Critch, Jerry Li, Dawn Song, and Jacob Steinhardt. 2021a. Aligning ai with shared human values. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021b. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

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

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark. Association for Computational Linguistics.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. 2022. Diffusion-lm improves controllable text generation. In *Advances in Neural Information Processing Systems*, volume 35, pages 4328–4343. Curran Associates, Inc.

Yifan Li, Kun Zhou, Wayne Xin Zhao, and Ji-Rong Wen. 2023. Diffusion models for non-autoregressive text generation: a survey. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, IJCAI '23.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and optimizing LSTM language models. In *International Conference on Learning Representations*.

Fatemehsadat Mireshghallah, Kartik Goyal, and Taylor Berg-Kirkpatrick. 2022. Mix and match: Learning-free controllable text generationusing energy language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 401–415, Dublin, Ireland. Association for Computational Linguistics.

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

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2019. The lambada dataset.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1).

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Winogrande: An adversarial winograd schema challenge at scale. *arXiv preprint arXiv:1907.10641*.

Nikolay Savinov, Junyoung Chung, Mikolaj Binkowski, Erich Elsen, and Aaron van den Oord. 2022. Step-unrolled denoising autoencoders for text generation. In *International Conference on Learning Representations*.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. 2018. Blockwise parallel decoding for deep autoregressive models. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *ArXiv*, abs/2302.13971.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Tong Wu, Zhihao Fan, Xiao Liu, Hai-Tao Zheng, Yeyun Gong, yelong shen, Jian Jiao, Juntao Li, zhongyu wei, Jian Guo, Nan Duan, and Weizhu Chen. 2023. Ar-diffusion: Auto-regressive diffusion model for text generation. In *Advances in Neural Information Processing Systems*, volume 36, pages 39957–39974. Curran Associates, Inc.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 43 others. 2024. Qwen2 technical report. *Preprint*, arXiv:2407.10671.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2020. Pegasus: pre-training with extracted gap-sentences for abstractive summarization. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org.

A Training Details

We collect and preprocess the English Wikipedia (about 13G in text) as training corpus to finetune different LLMs. Due to the limitation of available computational resources, we employ two approaches to finetune base models of various sizes, either updating all parameters or a few portion with LoRA (Hu et al., 2022). The optimizer used in training is AdamW with β_1 0.9, β_2 0.95 and weight decay 0.01. Other settings including total steps, learning rate and batch size used in finetuning are listed in Table 4. "Multi-token Heads" and "Look-ahead Tokens" represent two baselines trained through existing methods (Gloeckle et al., 2024; Monea et al., 2023). We train both models for more steps in order to achieve more stable and better performance on downstream tasks.

B Dataset Statistics

To evaluate the capability of finetuned models, we carefully choose five benchmarks concerning text comprehension and continuation, including MMLU (Hendrycks et al., 2021b,a), HellaSwag (Zellers et al., 2019), LAMBADA (Paperno et al., 2019), WinoGrande (Sakaguchi et al., 2019) and RACE (Lai et al., 2017). We use the toolkit "lm-evaluation-harness" under zero-shot setting for evaluation on all benchmarks (Gao et al., 2024). We also develop the decoding method via next-n-token prediction based on the codes of "lm-evaluation-harness". Since the lengths of answers for each benchmark greatly affect the measures calculated through next-n-token prediction, we analyze the average and maximum answer length for all benchmark datasets in Table 5.

C Complexity of NNTP

We firstly review the basic problem for causal language modeling, that is *to find the best word (or token) sequence as the continuation conditioning on a given context*. Taking the following sentence with target sequence consisting of $n = 3$ words as an example, the task for training a LLM is to predict the target words (bold text).

Alice likes to chat with her friends on **the phone .**

A language model trained via next-n-token prediction (NNTP) significantly differs from one using next-token prediction (NTP). NTP-based models are typically trained via teacher-forcing rather than teacherless employed in NNTP. Teacher-forcing essentially decomposes the above task into three sub-tasks of predicting next token individually, implemented simply by causal mask in each Transformer block.

Alice likes to chat with her friends on **the**
Alice likes to chat with her friends on the **phone**
Alice likes to chat with her friends on the phone **.**

If the vocabulary size of a LLM is denoted by V, the number of possible next tokens would thus be

Base Model	n_{train}	Trainable Parameters	Total Steps Learning Rate Batch Size	Other Hyperparameters
Qwen2 0.5B	4	100% (all)	20k/3e-5/64	
Qwen2 1.5B	2/3/4/6/8/10/13/16	100% (all)	20k/3e-5/64	
Qwen2 7B	4	2.33% (LoRA)	20k/3e-4/64	$r = 64$, $\alpha = 8$
Multi-token Heads	4	100% (all)	100k/1e-5/64	
Look-ahead Tokens	4	2.33% (LoRA)	100k/1e-4/64	$r = 64$, $\alpha = 8$

Table 4: Training settings used to finetune different models. "Multi-token Heads" represents the resulting model with linear output heads trained using the method from Gloeckle et al. (2024), while "Look-ahead Tokens" is finetuned after adding multiple "look-ahead" embeddings as in PaSS (Monea et al., 2023).

Dataset	#examples	Average Answer Length (#tokens)	Maximum Answer Length (#tokens)
MMLU	14015	1.0	1
HellaSwag	40145	29.44	80
LAMBADA$_{std}$	5151	1.46	5
WinoGrande	2534	5.61	20
RACE	4033	7.65	25

Table 5: Statistics of five benchmarks for evaluation. The answer length is measured by counting the continued tokens to be generated after input context for each example. We use the tokenizer of Qwen2 7B model to produce the answer sequences and collect statistical results in this table.

	Training	Inference
NTP	$O(nV)$	$O(\frac{V^n}{n})$
NNTP	$O(V^n)$	$O(V^n)$

Table 6: Approximate complexity (i.e., intrinsic difficulty of causal language modeling) of NNTP and NTP in training and at inference. The prerequisite of the analysis here is to predict n subsequent tokens from one forward pass in different ways. V denotes the vocabulary size of a particular LLM.

V for each sub-task above. Ignoring the efficacy of the LLM, the intrinsic difficulty of causal language modeling (i.e., complexity) can be roughly estimated by the size of search space, namely the number of possible solutions for the whole task. Apparently, the complexity of teacher-forcing for a single pass using a single example is $O(nV)$ by sequentially combining different sub-tasks together. Similarly, we can infer the complexity of teacher-less training $O(V^n)$ used by NNTP, which is exponentially larger than teacher-forcing.

As for inference, a LLM trained via NNTP normally generates n subsequent tokens simultaneously from one forward pass. As a contrast, n forward passes are conducted autoregressively via NTP in order to produce a target sequence of same length as above. Although the total number of possible newly-generated sequences from NNTP is generally identical to that from NTP, the later actually makes more attempts to achieve a comparable output with the former. Ignoring the impact of different LLMs, we can hence approximate the inference complexity for each attempt with the size of search space divided by the number of forward passes, thus $O(V^n)$ for NNTP and $O(\frac{V^n}{n})$ for NTP.

To sum up, a LLM based on NNTP tries to learn from a much harder task than NTP in training, and endeavor to generate text with clearly fewer attempts at inference (Table 6). Note that the results presented in Table 6 are derived after neglecting many relevant factors (e.g., different models and capabilities). Even so, the complexity trend of NNTP versus NTP still holds true if the test LLMs share a similar structure and are in the same level of generation ability.

D Error Distribution

We additionally analyze the error distribution over production positions to uncover the underlying cause of particular phenomena presented in Section

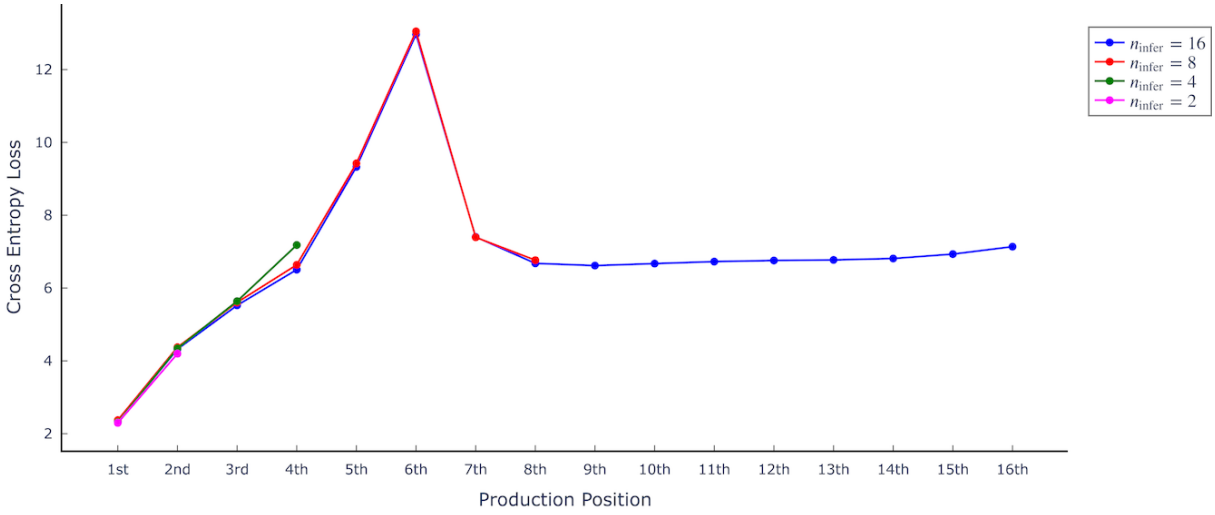

Figure 4: Error distribution over production positions. Using different inference strides, we compute the average cross-entropy loss for each production position over 30k examples of diverse lengths. The order of production position is ranked by minimum distance of corresponding token from input context in the actual sequence. For example, 1st indicates the position of last context token, and 2nd is that of the first mask token appended after input context.

3. To avoid the sample selection bias, we randomly choose 30k documents from the OpenWebText corpus (Gokaslan et al., 2019) and truncate the token sequence of each document to a random length as an example. Using our finetuned model derived from Qwen2 7B, we calculate the average cross-entropy loss in each production position for different inference strides. The experimental results are illustrated in Figure 4.

We discover an obvious trend that the errors increase with production position farther away from input context before the 7th position, and decrease dramatically and then stay stable starting from the 7th production position (Figure 4). The reason that the finetuned model tends to generate accurate tokens in the left end of production positions may be the closeness to input context according to the actual order, in which case the model is able to foresee many future tokens even if they are beyond the positions for new generations in current step. Consequently, we always achieve superior results if training with large unrolled stride and generating with a small one (Figure 3a and 3c).

On the other hand, generated tokens in the middle or right end of all production positions bring more uncertainty to the final output if using large inference stride. We conjecture that the lack of known context with strong local relations to the generated tokens may be the cause of potential errors in these tokens, which essentially accounts for the significant degradation when $n_{\text{infer}} \geq n_{\text{train}}$ in Figure 3c. However, the evident peak in the 6th position beats our knowledge about NNTP and LLM. We unfortunately fail to explain such unusual behavior of the finetuned model through extra experiments, and thus make more effort to investigate it in our future work.

As in previous experiments, we find some inconsistent phenomena on LAMBADA benchmark (Figure 3b and Table 3). The crucial fact affecting the performance of finetuned models on LAMBADA is probably the short answers in this dataset, which makes the models finetuned with large strides less effective than using small training strides (Figure 3b). As discussed above, only a few tokens in the left end of production positions are taken as final output due to small answer length, even if using a very large stride at inference. In addition, a small portion of examples have an answer length larger than one token, among which our model still benefits from the foreseeing ability learned via NNTP. As a consequence, generated tokens tend to match the actual answers with little errors (Figure 4), resulting in a better result than the original (4.35 vs 4.67 as in Table 3). Through the comprehensive analysis, we discover an effective way to improve the performance using NNTP, that is using a large stride during training and a small one at inference.