Hybrid-Regressive Paradigm for Accurate and Speed-Robust Neural Machine Translation

Qiang Wang[1,2], **Xinhui Hu**[2], **Ming Chen**[2*]
[1]Zhejiang University, Hangzhou, China
[2]RoyalFlush AI Research Institute, Hangzhou, China
{wangqiang3, huxinhui}@myhexin.com, chm@zju.edu.cn

Abstract

This study provides empirical evidence that non-autoregressive translation (NAT) is less robust in decoding batch size and hardware settings than autoregressive translation (AT). To address this issue, we demonstrate that incorporating a small number of AT predictions can significantly reduce the performance gap between AT and NAT through synthetic experiments. In line with this, we propose hybrid-regressive translation (HRT), a two-stage translation prototype that combines the strengths of AT and NAT. Specifically, HRT initially generates discontinuous sequences using autoregression (e.g., making predictions for every k tokens, $k > 1$), and then fills in all previously skipped tokens simultaneously in a non-autoregressive manner. Experimental results on five translation tasks show that HRT achieves comparable translation quality to AT while providing at least 1.5x faster inference, irrespective of batch size and device. Moreover, HRT successfully retains the desirable characteristics of AT in the deep-encoder-shallow-decoder architecture, enabling further speed improvements without sacrificing BLEU scores.[1]

1 Introduction

Autoregressive translation (AT) such as Transformer has been the *de facto* standard for Neural Machine Translation (NMT) (Vaswani et al., 2017). However, AT predicts only one target word at a time, resulting in slow inference speed. To overcome this limitation, non-autoregressive translation (NAT) attempts to generate the entire target sequence in one step by assuming conditional independence among target tokens (Gu et al., 2018). While NAT offers efficiency, it often suffers from significant degradation in translation quality. Achieving a better trade-off between inference speed and translation quality remains an active area of research for NAT (Wang et al., 2018a; Ran et al., 2020; Qian et al., 2021; Huang et al., 2022b,a).

One of the most successful approaches to this issue is the iterative refinement mechanism (IR-NAT) proposed by Lee et al. (2018), which has been widely adopted by several leading systems (Ghazvininejad et al., 2019; Kasai et al., 2020a; Guo et al., 2020; Saharia et al., 2020; Geng et al., 2021; Huang et al., 2022b). Specifically, IR-NAT, also known as multi-shot NAT, takes the translation hypothesis from the previous iteration as a reference to refine the new translation until it reaches the predefined iteration count I or no translation changes. Although a larger I can improve translation accuracy, it may also lead to a speedup degradation (Kasai et al., 2020b).

In this work, we build upon the findings of Kasai et al. (2020b) and examine the robustness of IR-NAT compared to AT. Our comprehensive examinations confirm that the inference speed of IR-NAT is consistently less robust than that of AT when involving various decoding batch sizes and computing hardware. For example, when using a GPU, the ten-iteration non-autoregressive model has 1.7/1.2/0.7/0.4 times the inference speed of the AT model for decoding batch sizes of 1/8/16/32, respectively. However, when switching to CPU, the relative speed ratio drops to 0.8/0.4/0.3/0.3 times. Previous studies have highlighted the complementary nature of AT and NAT in terms of both translation quality (AT being superior) and inference speed (NAT being superior) (Wang et al., 2018a; Ran et al., 2020). Our findings, however, suggest that there is also complementary robustness in inference speed (AT being superior).

Taking a further step, we investigate how much target context (i.e., the number of target tokens) is sufficient for one-shot NAT to rival multi-shot NAT through synthetic experiments. Our findings suggest that given a well-trained CMLM model, even if 70% of AT translations are masked, the

[*]Corresponding author.
[1]https://github.com/wangqiangneu/hrt

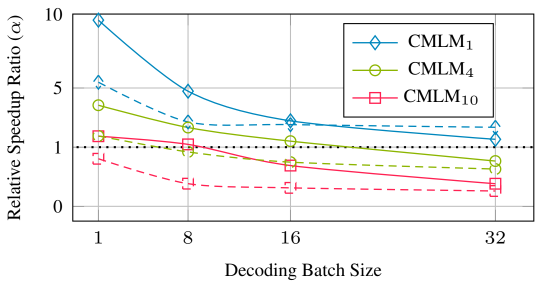

Figure 1: Relative speedup ratio (α) of CMLM compared to AT on GPU (solid) and CPU (dashed). $\alpha < 1$ indicates that CMLM is slower than AT. CMLM$_1$ can be regarded as the representative of one-shot NAT in inference speed.

remaining target context can help the CMLM$_1$ with greedy search compete with the standard CMLM$_{10}$ with beam search (see Figure 2). This could enable us to build the desired target context more cheaply, replacing expensive multiple iterations. To our best knowledge, this is the first study of the masking rate issue in the inference phase of NAT.

Based on the observations from these experiments, we have proposed a novel two-stage translation prototype called hybrid-regressive translation (HRT). This method combines the advantages of autoregressive translation (AT) and non-autoregressive translation (NAT) by first using an autoregressive decoder to generate a discontinuous target sequence with an interval of k ($k > 1$), and then filling the remaining slots in a lightweight non-autoregressive manner. We have also created a multi-task learning framework, enhanced by curriculum learning, for effective and efficient training without adding any model parameters. Results on WMT En$\leftrightarrow$Ro, En$\leftrightarrow$De, and NIST Zh$\rightarrow$En show that HRT outperforms prior work combining AT and NAT, and is competitive with state-of-the-art IR-NAT models. Specifically, HRT achieved a BLEU score of 28.27 on the WMT En$\rightarrow$De task, and is 1.5x faster than AT regardless of batch size and device. Additionally, HRT equipped with a deep-encoder-shallow-decoder architecture achieved up to 4x/3x acceleration on GPU/CPU, respectively, without sacrificing BLEU.

2 Background

Given a source sentence $\boldsymbol{x} = \{x_1, x_2, \ldots, x_M\}$ and a target sentence $\boldsymbol{y} = \{y_1, y_2, \ldots, y_N\}$, there are several ways to model $P(\boldsymbol{y}|\boldsymbol{x})$:

Autoregressive Translation (AT) AT is the predominant technique in NMT, decomposing $P(\boldsymbol{y}|\boldsymbol{x})$ using the chain rule: $P(\boldsymbol{y}|\boldsymbol{x}) = \prod_{t=1}^{N} P(y_t|\boldsymbol{x}, y_{<t})$, where $y_{<t}$ denotes the prefix translation generated before time step t. Nevertheless, autoregressive models must wait for y_{t-1} to be generated before predicting y_t, thus hindering parallelism over the target sequence.

Non-Autoregressive Translation (NAT) NAT has been proposed to generate target tokens simultaneously (Gu et al., 2018). This approach replaces the traditional autoregressive formulation of $y_{<t}$ with a target-independent input $\boldsymbol{z}$, resulting in the following formulation: $P(\boldsymbol{y}|\boldsymbol{x}) = P(N|\boldsymbol{x}) \times \prod_{t=1}^{N} P(y_t|\boldsymbol{x}, \boldsymbol{z})$. Various approaches have been proposed for modeling $\boldsymbol{z}$, such as using source embedding (Gu et al., 2018; Guo et al., 2019), reordering the source sentence (Ran et al., 2019), or using a latent variable (Ma et al., 2019; Shu et al., 2019).

Iterative Refinement based Non-Autoregressive Translation (IR-NAT) IR-NAT extends the traditional one-shot NAT by introducing an iterative refinement mechanism (Lee et al., 2018). We choose CMLM (Ghazvininejad et al., 2019) as the representative of IR-NAT due to its excellent performance and simplification. During training, CMLM randomly masks a fraction of tokens on $\boldsymbol{y}$ as the alternative to $\boldsymbol{z}$, and is trained as a conditional masked language model (Devlin et al., 2019). Denoting $\boldsymbol{y}^m/\boldsymbol{y}^r$ as the masked/residual tokens of $\boldsymbol{y}$, we have: $P(\boldsymbol{y}|\boldsymbol{x}) = \prod_{t=1}^{|\boldsymbol{y}^m|} P(\boldsymbol{y}_t^m|\boldsymbol{x}, \boldsymbol{y}^r)$. At inference, CMLM deterministically masks tokens from the hypothesis in the previous iteration $\hat{\boldsymbol{y}}^{(i-1)}$ according to prediction confidences. This process is iterated until $\hat{\boldsymbol{y}}^{(i-1)}$=$\hat{\boldsymbol{y}}^{(i)}$ or i reaches the maximum iteration count.

3 Acceleration Robustness Problem

In this section, we comprehensively analyze the inference acceleration robustness problem in IR-NAT. Without loss of generality, we take CMLM as the agency of IR-NAT.[2]

Problem Description The inference overhead of the autoregressive translation model mainly con-

[2]From the perspective of inference speed, we note that most one-shot NAT models are closed to CMLM$_1$. Especially, existing one-shot NAT models with CTC loss, such as GLAT and Fully-NAT, are theoretically slower than CMLM$_1$ because they require a longer target sequence for inference.

centrates on the decoder side(Hu et al., 2020). Suppose that the decoder's computational cost is proportional to the size of its input tensor (B, N, H), where B is the batch size, N is the target sequence length, and H is the network dimension. We omit H for convenience due to its invariance in NAT and AT. Thus, the total cost of AT model is about $C_{at} \propto N \times \mathcal{O}(B \times 1)$ [3]. Likely, the cost of I-iteration NAT is $C_{nat} \propto I \times \mathcal{O}(B \times N)$. Given a fixed test set, We can use $\mathcal{T}_D(\cdot)$ to represent the translation time on computing device D. This allows us to calculate the relative speedup ratio α between I-iteration NAT and AT as:

$$\alpha = \frac{\mathcal{T}_D(C_{at})}{\mathcal{T}_D(C_{nat})} \propto \frac{N}{I} \times \mathcal{E}(B, D), \qquad (1)$$

where $\mathcal{E}(B,D)=\frac{\mathcal{T}_D(\mathcal{O}(B\times 1))}{\mathcal{T}_D(\mathcal{O}(B\times N))} \leq 1$, denotes the parallel computation efficiency over sequence under batch size B and device D. When fixing N and I, α is completely determined by $\mathcal{E}(B, D)$. We note that most previous NAT studies only report the inference speed with D=GPU and B=1, without considering cases where B or D change.

Setup We systematically investigate the inference speed of CMLM [4] and AT under varying environments, including batch size $B \in \{1, 8, 16, 32\}$, device $D \in \{\text{GPU}, \text{CPU}\}$[5], and the number of iterations $I \in \{1, 4, 10\}$, using a beam size of 5. We test inference speed on the widely used WMT En→De *newstest2014* test set and report the average results over five runs (see Appendix A for details).

Results We plot the curve of relative speedup ratio (α) in Figure 1 and observe that:

i. α decreases as decoding batch size increases regardless of the number of iterations, as noted by Kasai et al. (2020b).

ii. α on CPU generally performs worse than GPU, except when using one iteration.

iii. The benefit of non-autoregressive decoding is more prone to disappear for larger numbers of iterations (I).

[3]Though the decoder self-attention module considers the previous i tokens, we omit it here for the sake of clarity.

[4]We use the officially released CMLM models from `https://github.com/facebookresearch/Mask-Predict`

[5]We use 2080Ti GPUs and Intel Xeon(R) E5-2683 v4 CPU, unless otherwise stated.

For instance, when decoding a single sentence on the GPU, the inference speed of the ten-iteration non-autoregressive model is 170% that of the autoregressive model. However, when switching to batches of 32 on CPU, the IR-NAT model only reaches 30% of the AT model's inference speed. These results demonstrate that AT and NAT possess different strengths, and combining the advantages of both models could be an effective way to achieve robust acceleration.

4 Synthetic Experiments

According to Equation 1, reducing the iteration count I helps to increase α. Recalling the refinement process of IR-NAT, we hypothesize that the essence of multiple iterations is to provide the decoder with a good enough target context (deterministic target tokens). This raises the question of *how many target tokens need to be provided to make one-shot NAT competitive with IR-NAT?* To answer it, we conduct synthetic experiments on WMT En→Ro and En→De to control the size of the target context by masking the partial translations generated by a pre-trained AT model. We then use a pre-trained CMLM model to predict these masks and observe the BLEU score curves under different masking rates.

Models We use the official CMLM models. Since the authors did not release the AT baselines, we used the same data to retrain AT models with the standard Transformer-Base configuration (Vaswani et al., 2017) and obtain comparable performance with the official ones (see Appendix B for details).

Decoding AT models decode with beam sizes of 5 on both tasks. Then we replace a certain percentage of AT tokens with `[MASK]` and feed them to CMLM. The used CMLM model only iterates once with beam size 1. We substitute all `[MASK]`s with CMLM's predictions to obtain the final translation. We report case-sensitive tokenized BLEU scores by *multi-bleu.perl*.

Mask Strategies We tested four strategies to mask AT results: `HEAD`, `TAIL`, `RANDOM`, and `CHUNK`. Given the masking rate p_{mask} and the translation length N, the number of masked tokens is N_{mask}=max(1, $\lfloor N \times \text{p}_{mask} \rfloor$). Then `HEAD`/`TAIL` always masks the first/last N_{mask} tokens, while `RANDOM` masks the translation randomly. `CHUNK` is slightly different from the above strategies. It first divides the target sentence into C chunks, where

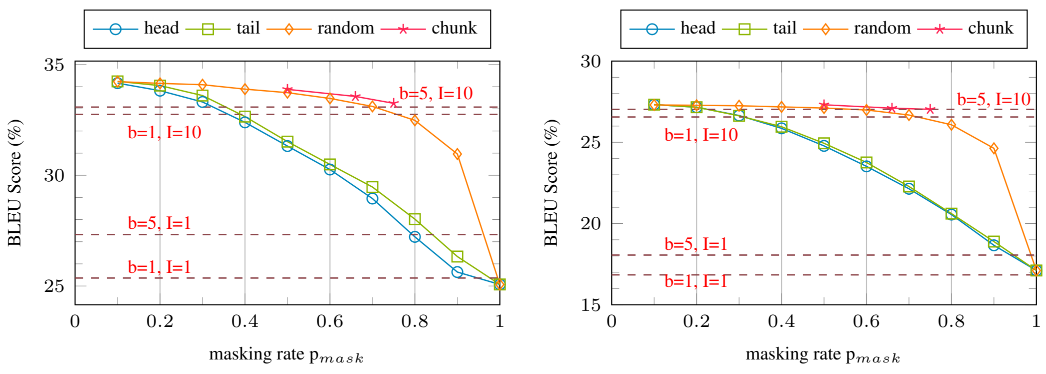

Figure 2: Comparison of four masking strategies {HEAD, TAIL, RANDOM, CHUNK} in synthetic experiments on WMT En→Ro (Left) and En→De (Right) test sets. For CHUNK, we test the chunk size from {2, 3, 4}. Dashed lines are the official CMLM scores. b stands for "beam size," and I stands for "the number of iterations".

$C = \text{Ceil}(N/k)$ and k is the chunk size. Then in each chunk, we retain the first token but mask other $k-1$ tokens. Thus, the actual masking rate in CHUNK is $1 - 1/k$ instead of p_{mask}. We ran RANDOM three times with different seeds to exclude randomness and report the average results.

Results The experimental results in Figure 2 demonstrate that CHUNK is moderately and consistently superior to RANDOM, and both strategies significantly outperform HEAD and TAIL. We attribute this success to the use of (1) bidirectional context (Devlin et al., 2019) (vs. HEAD and TAIL), and (2) the uniform distribution of deterministic tokens (vs. RANDOM)[6]. Furthermore, when using the CHUNK strategy, we find that exposing 30% AT tokens as the input of the decoder is enough to make our CMLM_1(beam=1) competitive with the official CMLM_{10}(beam=5), which emphasizes the importance of a good partial target context.

5 Hybrid-Regressive Translation

We propose a novel two-stage translation paradigm, Hybrid-Regressive Translation (HRT), which imitates the CHUNK process. In HRT, a discontinuous sequence with a chunk size of k is autoregressively generated in stage I, followed by non-autoregressive filling of the skipped tokens in stage II.

5.1 Architecture

Overview HRT consists of three components: encoder, Skip-AT decoder (for stage I), and Skip-CMLM decoder (for stage II). All components adopt the Transformer architecture (Vaswani et al., 2017). The two decoders have the same network structure, and we share them to make the parameter size of HRT the same as the vanilla Transformer. The only difference between the two decoders lies in the masking pattern in self-attention: The Skip-AT decoder masks future tokens to guarantee strict left-to-right generation like the autoregressive Transformer. In contrast, the Skip-CMLM decoder eliminates it to leverage the bi-directional context like CMLM (Ghazvininejad et al., 2019).

No Target Length Predictor Thanks to Skip-AT, we can obtain the translation length as a by-product: N_{nat}=$k \times N_{at}$, where N_{at} is the sequence length produced by Skip-AT. Our approach has two major advantages over most NAT models, which jointly train both the translation length predictor and the translation model. Firstly, there is no need to carefully adjust the weighting coefficient between the sentence-level length prediction loss and the word-level target token prediction loss. Secondly, the length predicted by Skip-AT is more accurate due to its access to the already-generated sequence information.

[6] CHUNK ensures that each masked token (except the last k-1 ones in the sequence) meets two deterministic tokens within the window size of k. However, RANDOM may degrade into HEAD/TAIL in extreme cases.

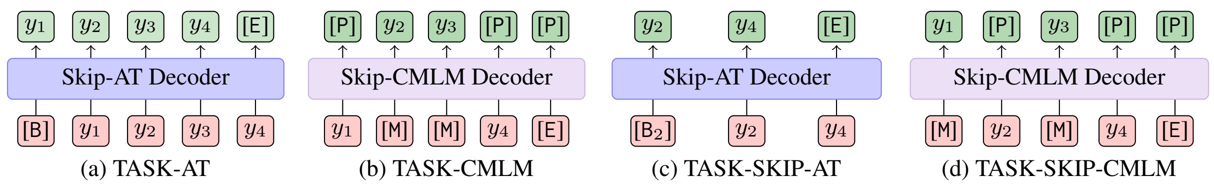

Figure 3: Examples of training samples for four tasks, where (a) and (b) are auxiliary tasks and (c) and (d) are primary tasks. To make the explanation clearer, the source sequence has been omitted. The special tokens [BOS], [EOS], [PAD], and [MASK] are represented by [B], [E], [P], and [M], respectively. Additionally, [B$_2$] is the [BOS] for k=2 and the loss at [P] is not taken into account.

5.2 Training

Next, we elaborate on how to train the HRT model efficiently and effectively. Please refer to Appendix C for the entire training algorithm.

Multi-Task Framework We learn HRT by jointly training four tasks: two primary tasks (TASK-SKIP-AT, TASK-SKIP-CMLM) and two auxiliary tasks (TASK-AT, TASK-CMLM). All tasks use cross-entropy as the training objective. Figure 3 illustrates the differences in training samples among these tasks. Notably, TASK-SKIP-AT shrinks the sequence length from N to N/k, while preserving the token positions from the original sequence. For example, in Figure 3 (c), the position of TASK-SKIP-AT input ([B$_2$], y_2, y_4) is (0, 2, 4). Auxiliary tasks are necessary to leverage all tokens in the sequence, as the two primary tasks are limited by the fixed k. For example, in Figure 3 (c) and (d), y_1 and y_3 cannot be learned as the decoder input of either TASK-SKIP-AT or TASK-SKIP-CMLM.

Curriculum Learning To ensure the model is not overly biased towards auxiliary tasks, we propose gradually transferring the training tasks from auxiliary tasks to primary tasks through curriculum learning (Bengio et al., 2009). We start with a batch of original sentence pairs $\mathcal{B}$, and let the proportion of primary tasks in $\mathcal{B}$ be p_k=0. We construct the training samples of TASK-AT and TASK-CMLM for all pairs, then gradually increase p_k to introduce more learning signals for TASK-SKIP-AT and TASK-SKIP-CMLM until p_k=1. We schedule p_k by $p_k = (t/T)^\lambda$, where t and T are the current and total training steps, and λ is a hyperparameter set to 1 for linear increase.

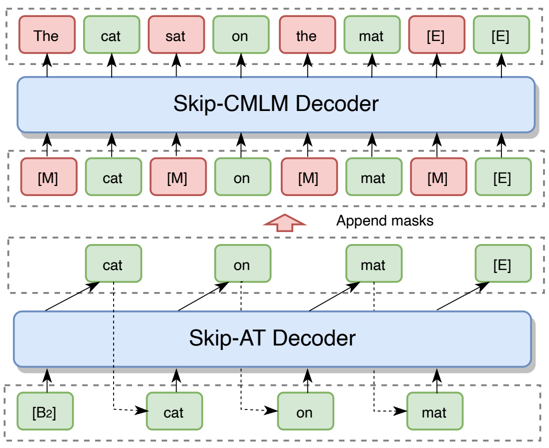

Figure 4: The decoding process of HRT with k=2. For the sake of clarity, we omit the source sentence.

5.3 Inference

As illustrated in Figure 4, HRT adopts a two-stage generation strategy. In the first stage, the Skip-AT decoder autoregressively generates a discontinuous target sequence $\hat{\boldsymbol{y}}_{at} = (z_1, z_2, \ldots, z_m)$ with chunk size k, starting from [BOS$_k$] and ending with [EOS]. Then, the input of Skip-CMLM decoder $\boldsymbol{y}_{nat}$ is constructed by appending $k-1$ [MASK]s before every z_i. The final translation is generated by replacing all [MASK]s with the predicted tokens after one iteration of the Skip-CMLM decoder. If multiple [EOS]s exist, we truncate to the first [EOS]. The beam sizes b$_{at}$ and b$_{nat}$ can be different from each other, as long as b$_{at} \geq$ b$_{nat}$. In our implementation, we use standard beam search in Skip-AT (b$_{at}$ >1) and greedy search in Skip-CMLM (b$_{nat}$=1). Table 3 provides more details on the beam size setting in HRT. The translation hypothesis with the highest score $\mathrm{S}(\hat{y})$ is chosen by summing the Skip-AT score and the Skip-CMLM score:

$$\underbrace{\sum_{i=1}^{m} \log P(z_i|\boldsymbol{x}, \boldsymbol{z}_{<i})}_{\text{Skip-AT score}} + \underbrace{\sum_{i=0}^{m-1}\sum_{j=1}^{k-1} \log P(\hat{y}_{i\times k+j}|\boldsymbol{x}, \boldsymbol{y}_{nat})}_{\text{Skip-CMLM score}} \quad (2)$$

where $z_i = \hat{y}_{i\times k}$.

5.4 Discussion

The basic idea of HRT is to apply Autoregressive Transformation (AT) and Non-Autoregressive Transformation (NAT) in sequence. This concept has been investigated before (Kaiser et al., 2018), Ran et al. (2019), and Akoury et al. (2019). The main differences between these methods lie in the content of the AT output, such as latent variable (Kaiser et al., 2018), reordered source tokens (Ran et al., 2019), and syntactic labels (Akoury et al., 2019). In contrast, our approach uses the deterministic target token following Ghazvininejad et al. (2019).

HRT is related to chunk-wise decoding, another line incorporating AT and NAT. Table 1 shows the differences between HRT and prior studies, including SAT (Wang et al., 2018a), RecoverSAT (Ran et al., 2020), and LAT (Kong et al., 2020). SAT and LAT follow the generation order of from-left-to-right, behaving similarly to HEAD as described in Section 4. In contrast, RecoverSAT and HRT generate discontinuous target contexts, which have been shown to perform better than HEAD according to our synthetic experiments. However, RecoverSAT cannot accurately generate the discontinuous context "a,c,e" via non-autoregression, resulting in error propagation for generating "b,d,f". However, HRT produces "a,c,e" through accurate autoregression. Additionally, although HRT requires more decoding steps, its non-autoregressive process is inexpensive due to greedy searches. In contrast, other methods require larger beams to explore translations of different lengths.

Method	Generation
SAT	$a, b \rightarrow c, d \rightarrow e, f$
RecoverSAT	$a, c, e \rightarrow b, d, f$
LAT	$\{\text{a} \rightarrow b \rightarrow c, d \rightarrow e \rightarrow f\}$
HRT (Our)	$a \rightarrow c \rightarrow e \dashrightarrow b, d, f$

Table 1: Examples of generating the sequence of "a, b, c, d, e, f" by different methods. The elements in "$\{\}$" are generated in parallel. "$\rightarrow$" denotes a new decoding step conditioned on the prefix with beam search, while "$\dashrightarrow$" is its greedy search version.

b$_{at}$	b$_{nat}$	En→Ro	En→De	α(AG)	α(AC)
1	1	34.16	28.19	**2.1**	**2.8**
5	1	**34.37**	28.27	1.7	1.6
5	5	34.36	**28.50**	N/A	1.1

Table 3: Effects of different beam sizes in HRT. α(AG) and α(AC) denotes the average relative speedup ratio with batch size {1,8,16,32} on GPU and CPU, respectively (see Appendix A for details). "N/A" denotes decoding failed with batch size 32 due to insufficient GPU memory.

6 Experimental Results

Setup We conduct experiments on five tasks, including WMT'16 English↔Romanian (En↔Ro, 610k), WMT'14 English↔German (En↔De, 4.5M) and long-distance language pair NIST Chinese-English (Zh→En, 1.8M). For fair comparisons, we replicate the same data processing as Ghazvininejad et al. (2019) in four WMT tasks and follow the setup of Wang et al. (2018b) for Zh→En. Like previous work, we train HRT through sequence-level knowledge distillation (Kim and Rush, 2016). Specifically, we use the standard Transformer-Base as teacher models for En↔Ro and Zh→En, while we use the deep PreNorm Transformer-Base with a 20-layer encoder for harder En↔De. We run all experiments on four 2080Ti GPUs. Unless noted otherwise, we use the chunk size k=2. We fine-tune HRT models on pre-trained AT models and take 100k/300k/100k training steps for En↔Ro/En↔De/Zh→En, respectively. Other training hyperparameters are the same as Vaswani et al. (2017) or Wang et al. (2019) (deep-encoder). We report both case-sensitive tokenized BLEU scores and SacreBLEU [7]. We also report COMET as suggested by Helcl et al. (2022).

Beam Size on HRT We first verify the influence of two beam sizes of HRT (b$_{at}$ and b$_{nat}$) on the BLEU score and relative speedup ratio by testing three different setups. The results are listed in Table 3. Consistent with our observations in synthetic experiments, using b$_{nat}$=1 only slightly reduces BLEU but significantly improves decoding efficiency. Considering the trade-off between translation quality and speed, and for a fair comparison with other baselines (most prior related work uses beam size of 5), we use b$_{at}$=5 and b$_{nat}$=1 unless

[7]Signature: BLEU+case.mixed+lang.*source-target*+numrefs.1+smooth.exp+tok.13a+version.1.5.1

	System	Param.	Iter.	WMT'16		WMT'14		COMET
				En-Ro	Ro-En	En-De	De-En	
	Existing systems							
NAT	AXE (Ghazvininejad et al., 2020a)	-	1	30.75	31.54	23.53	27.90	-
NAT	GLAT+CTC (Qian et al., 2021)	-	1	32.79	33.84	26.39	29.54	-
NAT	Fully-NAT (Gu and Kong, 2021)	-	1	33.79	34.16	27.49	31.39	-
NAT	DA-Transformer (Huang et al., 2022a)	73M	1	-	-	27.91	31.95	-
Iterative NAT	CMLM (Ghazvininejad et al., 2019)	76M	10	33.08	33.31	27.03	30.53	0.4338
Iterative NAT	LevTransformer (Gu et al., 2019)	-	Adaptive	-	-	27.27	-	-
Iterative NAT	JM-NAT (Guo et al., 2020)	-	10	33.52	33.72	27.69	32.24	-
Iterative NAT	SMART (Ghazvininejad et al., 2020b)	-	10	-	-	27.65	31.27	-
Iterative NAT	DisCO (Kasai et al., 2020a)	-	Adaptive	33.22	33.25	27.34	31.31	-
Iterative NAT	Imputer (Saharia et al., 2020)	-	8	34.40	34.10	28.20	31.80	-
Iterative NAT	RewriteNAT (Geng et al., 2021)	-	Adaptive	33.63	34.09	27.83	31.52	-
Iterative NAT	CMLMC (Huang et al., 2022b)	-	10	34.57	34.13	28.37	31.41	-
Semi-NAT	SAT (Wang et al., 2018a)	-	$N/2$	-	-	26.90	-	-
Semi-NAT	SynST (Akoury et al., 2019)	-	$N/6+1$	-	-	20.74†	25.50†	-
Semi-NAT	ReorderNAT (Ran et al., 2019)	-	$N+1$	31.70	31.99	26.49	31.13	-
Semi-NAT	RecoverSAT (Ran et al., 2020)	-	$N/2$	32.92	33.19	27.11	31.67	-
	Our implementations							
Raw	AT (teacher for En↔Ro)	61M	N	34.25(34.2†)	34.40(34.0†)	27.45(26.9†)	31.86(31.6†)	0.4779
Raw	AT$_{20-6}$ (teacher for En↔De)	105M	N	-	-	28.79(28.2†)	33.02(32.8†)	0.5201
Raw	HRT	61M	$N/2+1$	33.59(33.5†)	32.98(32.9†)	26.69(26.2†)	30.58(30.3†)	0.4331
Distillation	AT	61M	N	34.14(33.9†)	34.06(33.8†)	**28.24(27.7†)**	31.95(31.7†)	0.4922
Distillation	SAT	61M	$N/2$	-	-	26.47(25.9†)	29.40(29.1†)	0.1848
Distillation	GLAT-CTC	62M	1	-	-	26.59(26.0†)	29.73(29.4†)	0.1712
Distillation	HRT	61M	$N/2+1$	34.37(34.2†)	34.14(33.9†)	**28.27(27.7†)**	32.02(31.7†)	0.4881
Distillation	HRT$_{20-6}$	105M	$N/2+1$	-	-	**29.06(28.5†)**	33.20(32.9†)	0.5098

Table 2: Results of BLEU, SacreBLEU (denoted by †), and COMET on four WMT tasks. By default, these models have a 6-layer encoder and a 6-layer decoder. The subscript $X-Y$ is used to denote the X-layer encoder and Y-layer decoder. Boldface results are significantly better (p<0.01) than those of the autoregressive counterparts in the same network capacity trained by raw data, as indicated by paired bootstrap resampling (Koehn, 2004).

Model	MT04	MT05	MT08
AT (teacher)	**43.86**	52.91	33.94
CMLM$_{10}$	42.47	52.16	33.09
HRT	43.81	**52.99**	**34.17**

Table 4: BLEU scores on NIST Zh→En task.

otherwise stated.

Main Results We compare the performance of HRT with existing systems in different translation paradigms on four WMT tasks, as shown in Table 2. HRT with distillation data consistently outperforms that of raw data and most existing NAT, IR-NAT, and Semi-NAT models, obtaining a BLEU score of 28.27 on the widely used En→De task. Compared to the re-implemented typical semi-autoregressive model (SAT) and one-shot non-autoregressive model (GLAT-CTC), HRT obtains an improvement of approximately 1.7 BLEU points, with a more significant margin in COMET score. Moreover, HRT$_{20-6}$ can improve by 0.7 BLEU and 0.02 COMET when using a deeper encoder. Interestingly, the evaluation results of BLEU and COMET are inconsistent, as observed by Helcl et al. (2022). For instance, HRT$_{20-6}$ has higher BLEU score than AT$_{20-6}$ on En→De, but its COMET score is still lower. Furthermore, the experimental results on the Zh→En task, as reported in Table 4, demonstrate that the effectiveness of HRT is agnostic to language pairs, as it is close or superior to the original AT and CMLM model. We attribute this to two reasons: (1) HRT is fine-tuned on a well-trained AT model; (2) Multi-task learning on autoregressive and non-autoregressive tasks has better regularization than training alone.

7 Analysis

Impact of Chunk Size We tested chunk size k on the En→De task, as shown in Table 5. We observed that larger values of k had a more significant speedup on the GPU, as fewer autoregressive steps were required. However, as k increased, the performance of HRT dropped sharply; for example, k=4 was about 1.24 BLEU points lower than k=2 on the test set. This suggests that the training difficulty of Skip-AT increases as k becomes larger. Further investigation into more sophisticated training algorithms to address this is left for our future work.

Chunk	Valid	Test	α(AG)	α(AC)
2	**26.44**	**28.27**	1.7	1.6
3	26.34	27.92	2.5	2.3
4	25.60	27.03	**3.2**	**2.9**

Table 5: Effects of chunk size (k) on BLEU and α.

Stage I	Stage II	BLEU	Δ
HRT	HRT	28.27	ref.
HRT_{20-6}	HRT_{20-6}	29.06	+0.79
HRT	HRT_{20-6}	28.42	+0.15
HRT_{20-6}	HRT	28.88	+0.61

Table 6: Swapping two decoding stages between HRT (weak model) and HRT_{20-6} (strong model).

Which decoding stage is more important? To understand the importance of the two decoding stages of HRT, we exchange the intermediate results of two HRT models (A and B). Specifically, we use the Skip-AT decoder of A to generate its discontinuous target sequence, which is then forced decoded by B's Skip-AT decoder to obtain corresponding encoding representations and autoregressive model scores. Finally, B's Skip-CMLM decoder generates the complete translation result based on these. We can reverse the order of A and B as well. We use two models (HRT and HRT_{20-6}) with a large performance gap as A and B, respectively. As shown in Table 6, we find that using the result of stage I of the strong model brings a greater improvement (+0.61 BLEU) than that of stage II (+0.15 BLEU). This result supports our hypothesis that a good partial target context is essential.

Deep-encoder-shallow-decoder Architecture Kasai et al. (2020b) showed AT with deep-encoder-shallow-decoder architecture can speed up translation without sacrificing accuracy, while CMLM fails. To validate whether HRT can inherit this, we compared HRT and AT with a 12-layer encoder and 1-layer decoder (HRT_{12-1} and AT_{12-1}), using the same distillation data. As Table 7 shows, both AT_{12-1} and HRT_{12-1} benefit from the layer allocation, achieving comparable BLEU scores and double the decoding speed of the vanilla models. Specifically, HRT_{12-1} achieved an average acceleration of 4.2x/3.1x over the AT baselines. This suggests HRT_{12-1}'s success was due to Skip-AT rather than Skip-CMLM. However, its COMET scores are lower than 6-6 architecture.

Model	BLEU	COMET	α(AG)	α(AC)
AT	28.24	**0.4922**	ref.	ref.
AT_{12-1}	**28.40**	0.4539	2.7	2.1
HRT	28.27	0.4881	1.7	1.6
HRT_{12-1}	28.24	0.4152	**4.2**	**3.1**

Table 7: Effects of deep-encoder-shallow-decoder architecture on En→De test set. All models use distillation data.

System	BLEU	Δ
HRT (T=300k)	28.27	ref.
−FT	28.00	-0.27
−CL (p_k=1)	27.53	-0.74
−CL (p_k=0.5)	27.75	-0.52
−TS (T=100k)	27.82	-0.45
−ALL	26.59	-1.68

Table 8: Ablation study on En→De task.

Further research into decoder depth and COMET correlation will be conducted.

Ablation Study In Table 8, we conduct an ablation study on the En→De task to investigate the contribution of fine-tuning from pre-trained AT (FT), training steps (TS), and curriculum learning (CL). We test two settings about CL: Fixing p_k=1 is equivalent to removing auxiliary tasks; Fixing p_k=0.5 assigns the same probability to the primary and auxiliary tasks. The results show that all components contribute to the performance, but CL and TS are the most critical, with a reduction of 0.74 and 0.45 BLEU points, respectively. Excluding all components from the vanilla HRT (-ALL) leads to a total reduction of 1.68 BLEU points.

Case study Table 9 presents a translation example from En→De validation set. Comparing CMLM_5 and HRT, both having the same masking rate (50%), two main distinctions can be observed: (1) The distribution of masked tokens in CMLM is more discontinuous than in HRT (as indicated by the blue marks); (2) The decoder input of HRT contains more accurate target tokens than CMLM, due to the Skip-AT decoder (as indicated by the wavy marks). These differences make our model more effective in producing high-quality translations than CMLM, and suggest that our model can generate appropriate discontinuous sequences.

Source	Also problematic : civil military jurisdiction will continue to be uph@@ eld .
Reference	Auch problematisch : Die zivile Militär@@ geri@@ chts@@ barkeit soll weiter aufrechterhalten bleiben .
CMLM$_{10}$ (5th)	**Problem@@ atisch : Die zivile militärische** Gerichts@@ barkeit wird weiterhin **aufrechterhalten** . [EOS]
HRT	**Auch** problematisch **:** Die **zivile** Militär@@ **geri@@** chts@@ **barkeit** wird **weiterhin** aufrechterhalten **werden** . **[EOS]** [EOS]

Table 9: A case study in En→De validation set. **Blue** denotes the original input is [MASK]. We add a wavy line under the target context tokens (black) that hit the reference translation. We also report the CMLM$_{10}$ in the 5th iteration that has closing mask rate to HRT.

8 Conclusion

We noted that IR-NAT has robustness issues with inference acceleration. Inspired by our findings in synthetic experiments, we proposed HRT to take advantage of the strengths of both AT and NAT. Our experiments demonstrated that our approach surpasses existing semi-autoregressive and IR-NAT methods, providing competitive performance and consistent speedup, making it a viable alternative to autoregressive translation.

9 Limitations

The main limitation of HRT is that its upper bound on the inference speedup is lower than that of single-iteration NAT under the same network architecture. As demonstrated in Appendix A, the average speedup of single-iteration NAT (i.g., CMLM$_1$) is 4.7x/3.2x on GPU/CPU, respectively, while that of HRT is 1.7x/1.6x. To achieve higher acceleration, HRT needs to employ the deep-encoder-shallow-decoder architecture. Increasing the chunk size is a simple way to reduce the autoregressive cost, yet it results in severe BLEU degradation (see Table 5). Further research should be conducted to maintain high translation performance with fewer autoregressive prompts.

Acknowledgements

We would like to thank the anonymous reviewers for their helpful comments. We also thank Shuqin Pan for the writing suggestions.

References

Nader Akoury, Kalpesh Krishna, and Mohit Iyyer. 2019. Syntactically supervised transformers for faster neural machine translation. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 1269–1281.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In Proceedings of the 26th Annual International Conference on Machine Learning, ICML '09, page 41–48, New York, NY, USA. Association for Computing Machinery.

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.

Xinwei Geng, Xiaocheng Feng, and Bing Qin. 2021. Learning to rewrite for non-autoregressive neural machine translation. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 3297–3308, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Marjan Ghazvininejad, Vladimir Karpukhin, Luke Zettlemoyer, and Omer Levy. 2020a. Aligned cross entropy for non-autoregressive machine translation. In ICML 2020: 37th International Conference on Machine Learning, volume 1, pages 3515–3523.

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.

Marjan Ghazvininejad, Omer Levy, and Luke Zettlemoyer. 2020b. Semi-autoregressive training improves mask-predict decoding. arXiv preprint arXiv:2001.08785.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O.K. Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In International Conference on Learning Representations.

Jiatao Gu and Xiang Kong. 2021. Fully non-autoregressive neural machine translation: Tricks of the trade. In ACL 2021: 59th annual meeting of the Association for Computational Linguistics, pages 120–133.

Jiatao Gu, Changhan Wang, and Junbo Zhao. 2019. Levenshtein transformer. In Advances in Neural Information Processing Systems, pages 11179–11189.

Junliang Guo, Xu Tan, Di He, Tao Qin, Linli Xu, and Tie-Yan Liu. 2019. Non-autoregressive neural machine translation with enhanced decoder input. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, pages 3723–3730.

Junliang Guo, Linli Xu, and Enhong Chen. 2020. Jointly masked sequence-to-sequence model for non-autoregressive neural machine translation. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 376–385.

Jindřich Helcl, Barry Haddow, and Alexandra Birch. 2022. Non-autoregressive machine translation: It's not as fast as it seems. In Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 1780–1790, Seattle, United States. Association for Computational Linguistics.

Chi Hu, Bei Li, Yinqiao Li, Ye Lin, Yanyang Li, Chenglong Wang, Tong Xiao, and Jingbo Zhu. 2020. The NiuTrans system for WNGT 2020 efficiency task. In Proceedings of the Fourth Workshop on Neural Generation and Translation, pages 204–210, Online. Association for Computational Linguistics.

Fei Huang, Hao Zhou, Yang Liu, Hang Li, and Minlie Huang. 2022a. Directed acyclic transformer for non-autoregressive machine translation. In Proceedings of the 39th International Conference on Machine Learning, volume 162 of Proceedings of Machine Learning Research, pages 9410–9428. PMLR.

Xiao Shi Huang, Felipe Perez, and Maksims Volkovs. 2022b. Improving non-autoregressive translation models without distillation. In International Conference on Learning Representations.

Lukasz Kaiser, Samy Bengio, Aurko Roy, Ashish Vaswani, Niki Parmar, Jakob Uszkoreit, and Noam Shazeer. 2018. Fast decoding in sequence models using discrete latent variables. In International Conference on Machine Learning, pages 2390–2399.

Jungo Kasai, James Cross, Marjan Ghazvininejad, and Jiatao Gu. 2020a. Non-autoregressive machine translation with disentangled context transformer. In ICML, pages 5144–5155.

Jungo Kasai, Nikolaos Pappas, Hao Peng, James Cross, and Noah A. Smith. 2020b. Deep encoder, shallow decoder: Reevaluating the speed-quality tradeoff in machine translation. arXiv preprint arXiv:2006.10369.

Yoon Kim and Alexander M Rush. 2016. Sequence-level knowledge distillation. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1317–1327.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, pages 388–395, Barcelona, Spain. Association for Computational Linguistics.

Xiang Kong, Zhisong Zhang, and Eduard Hovy. 2020. Incorporating a local translation mechanism into non-autoregressive translation. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1067–1073, Online. Association for Computational Linguistics.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1173–1182, Brussels, Belgium.

Xuezhe Ma, Chunting Zhou, Xian Li, Graham Neubig, and Eduard Hovy. 2019. Flowseq: Non-autoregressive conditional sequence generation with generative flow. 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 4273–4283.

Lihua Qian, Hao Zhou, Yu Bao, Mingxuan Wang, Lin Qiu, Weinan Zhang, Yong Yu, and Lei Li. 2021. 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), pages 1993–2003, Online. Association for Computational Linguistics.

Qiu Ran, Yankai Lin, Peng Li, and Jie Zhou. 2019. Guiding non-autoregressive neural machine translation decoding with reordering information. arXiv preprint arXiv:1911.02215.

Qiu Ran, Yankai Lin, Peng Li, and Jie Zhou. 2020. Learning to recover from multi-modality errors for non-autoregressive neural machine translation. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 3059–3069, Online. Association for Computational Linguistics.

Chitwan Saharia, William Chan, Saurabh Saxena, and Mohammad Norouzi. 2020. Non-autoregressive machine translation with latent alignments. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1098–1108.

Raphael Shu, Jason Lee, Hideki Nakayama, and Kyunghyun Cho. 2019. Latent-variable non-autoregressive neural machine translation with deterministic inference using a delta posterior. arXiv preprint arXiv:1908.07181.

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, pages 6000–6010.

Chunqi Wang, Ji Zhang, and Haiqing Chen. 2018a. Semi-autoregressive neural machine translation. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 479–488.

Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F. Wong, and Lidia S. Chao. 2019. Learning deep transformer models for machine translation. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 1810–1822, Florence, Italy.

Qiang Wang, Fuxue Li, Tong Xiao, Yanyang Li, Yinqiao Li, and Jingbo Zhu. 2018b. Multi-layer representation fusion for neural machine translation. In Proceedings of the 27th International Conference on Computational Linguistics, pages 3015–3026.

A Detailed inference speed

In Table 10, we list the exact decoding time and relative speedup ratio of different models under varying environments on the En$\rightarrow$De test set. When changing the batch size from 1 to 32, the decoding time of AT reduces 20.4x/4.6x on GPU/CPU, respectively, while that of CMLM$_{10}$ only reduces 3.7/0.8x. In contrast, HRT inherits the good character of AT and achieves an 18.7x/3.8x speedup. On the other hand, HRT has more robust acceleration than multi-shot NAT, such as CMLM$_4$, CMLM$_{10}$. When using the deep-encoder-shallow-decoder architecture, HRT$_{12-1}$ performance approaches the one-shot NAT (CMLM$_1$) on both GPU and CPU. Besides, the overall results of HRT-20L are similar to those of HRT because the translation time is mainly consumed in the decoder. We also report the change of inference speed along with chunk size k.

B AT Transformers in synthetic experiments

We trained all AT models in the synthetic experiment with the standard Transformer-Base configuration: layer=6, dim=512, ffn=2048, head=8. The difference from Ghazvininejad et al. (2019) is that they trained the AT models for 300k steps, but we updated 50k/100k steps on En$\rightarrow$Ro and En$\rightarrow$De, respectively. Although fewer updates, as shown in Table 11, our AT models have comparable performance with theirs.

C Training algorithm

Algorithm 1 describes the training process of HRT. The HRT model is pre-initialized by a pre-trained AT model (Line 1). Then according to the schedule strategy $p_k = \left(\frac{t}{T}\right)^{\lambda}$, we can divide the training batch $\boldsymbol{B}$ into two parts: $\boldsymbol{B}_p$ for primary tasks and $\boldsymbol{B}_a$ for auxiliary tasks, where $|\boldsymbol{B}_p|/|\boldsymbol{B}| = p_k$ (Line 4-5). Next, we construct four kinds of training samples based on corresponding batches: $\boldsymbol{B}_p^{at}$ (`TASK-SKIP-AT`), $\boldsymbol{B}_a^{at}$ (`TASK-AT`), $\boldsymbol{B}_p^{nat}$ (`TASK-SKIP-CMLM`) and $\boldsymbol{B}_a^{nat}$ (`TASK-CMLM`). Finally, we collect all training samples together and accumulate their gradients to update the model parameters, which results in the batch size being twice that of standard training.

Model	**BLEU**↑	*B*=**1** Time↓	*B*=**1** α↑	*B*=**8** Time↓	*B*=**8** α↑	*B*=**16** Time↓	*B*=**16** α↑	*B*=**32** Time↓	*B*=**32** α↑	**Avg** α↑
					On GPU					
AT(raw data)	27.45	857.2	1.0	137.8	1.0	73.1	1.0	40.1	1.0	1.0
AT$_{12-1}$	28.40	294.7	2.9	49.1	2.8	28.1	2.6	16.7	2.4	2.7
CMLM$_1$	18.05	**89.4**	**9.6**	**28.8**	**4.8**	26.3	2.8	26.2	1.5	**4.7**
CMLM$_4$	25.94	223.5	3.8	59.2	2.3	52.0	1.4	52.4	0.8	2.1
CMLM$_{10}$	27.03	492.7	1.7	116.0	1.2	106.1	0.7	105.0	0.4	1.0
SAT	26.47	523.0	1.6	87.1	1.6	48.0	1.5	26.2	1.5	1.6
HRT (b$_{at}$=1, b$_{nat}$=1)	28.19	377.5	2.3	66.4	2.1	34.9	2.1	20.5	2.0	2.1
HRT	28.27	478.9	1.8	77.8	1.8	41.9	1.7	24.3	1.7	1.7
HRT (b$_{at}$=5, b$_{nat}$=5)	28.50	482.4	1.8	81.2	1.7	46.5	1.6	N/A	N/A	N/A
HRT$_{12-1}$	28.24	192.5	4.6	31.4	4.3	**18.4**	**4.0**	**11.1**	**3.7**	4.2
HRT (k=3)	27.92	323.9	2.6	54.9	2.5	29.5	2.5	18.2	2.2	2.5
HRT (k=4)	27.03	256.0	3.3	43.1	3.2	23.3	3.1	12.7	3.2	3.2
					On CPU					
AT (raw data)	27.45	1118.0	1.0	314.1	1.0	246.3	1.0	201.3	1.0	1.0
AT$_{12-1}$	28.40	405.4	2.8	149.0	2.1	130.4	1.9	110.7	1.8	2.1
CMLM$_1$	18.05	**207.3**	**5.4**	116.0	2.7	97.6	2.5	85.9	2.3	**3.2**
CMLM$_4$	25.94	635.1	1.8	341.7	0.9	329.8	0.7	319.4	0.6	1.0
CMLM$_{10}$	27.03	1390.9	0.8	820.1	0.4	789.3	0.3	776.9	0.3	0.4
SAT	26.47	737.5	1.5	248.7	1.3	205.6	1.2	158.9	1.3	1.3
HRT (b$_{at}$=1, b$_{nat}$=1)	28.19	457.1	2.4	116.1	2.7	**82.4**	**3.0**	**65.9**	**3.1**	2.8
HRT	28.27	663.1	1.7	186.3	1.7	157.8	1.6	138.0	1.5	1.6
HRT (b$_{at}$=5, b$_{nat}$=5)	28.50	811.0	1.4	294.5	1.1	247.6	1.0	235.2	0.9	1.1
HRT$_{12-1}$	28.24	249.6	4.5	111.5	2.8	85.1	2.9	83.9	2.4	3.1
HRT (k=3)	27.92	448.7	2.5	134.8	2.3	111.7	2.2	90.7	2.2	2.3
HRT (k=4)	27.03	360.0	3.1	**111.4**	**2.8**	85.8	2.9	71.9	2.8	2.9

Table 10: Compare the BLEU score, elapsed time, and relative speedup ratio (α) of decoding En→De *newstest14* under different settings. We use b$_{at}$=5, b$_{nat}$=1 and k=2 for HRT unless otherwise stated. HRT(b$_{at}$=5, b$_{nat}$=5) cannot decode data with batch size 32 (denoted by N/A) on GPU due to insufficient GPU memory. We bold the best results. Green denotes the result is worse than AT baseline.

AT Transformer	**En-Ro**	**En-De**
Vaswani et al. (2017)	-	27.3
Ghazvininejad et al. (2019)	34.28	27.74
Our implementation	34.25	27.45

Table 11: The performance of autoregressive models in the synthetic experiment.

Algorithm 1 Training Algorithm for Hybrid-Regressive Translation

```
Input: Training data $D$, pretrained AT model $\mathrm{M}_{at}$, chunk size $k$, schedule coefficient $\lambda$
Output: Hybrid-Regressive Translation model $\mathrm{M}_{hrt}$
1: $\mathrm{M}_{hrt} \leftarrow \mathrm{M}_{at}$                                    ▷ fine-tune on pre-trained AT
2: for $t$ in $1, 2, \ldots, T$ do
3:     $\boldsymbol{B} = \langle \boldsymbol{x}_i, \boldsymbol{y}_i \rangle|_{i=1}^{n}$                          ▷ fetch a batch $\boldsymbol{B}$ from $D$
4:     $p_k \leftarrow (\frac{t}{T})^{\lambda}$                                   ▷ curriculum learning
5:     $\boldsymbol{B}_p, \boldsymbol{B}_a \leftarrow \boldsymbol{B}\big[:\lfloor n \times p_k \rfloor\big], \boldsymbol{B}\big[\lfloor n \times p_k \rfloor :\big]$     ▷ split batch for different tasks
6:     $\boldsymbol{B}_p^{at}, \boldsymbol{B}_p^{nat} \leftarrow$ construct training samples of primary tasks based on $\boldsymbol{B}_p$
7:     $\boldsymbol{B}_a^{at}, \boldsymbol{B}_a^{nat} \leftarrow$ construct training samples of auxiliary tasks based on $\boldsymbol{B}_a$
8:     Optimize $\mathrm{M}_{hrt}$ using $\boldsymbol{B}_p^{at} \cup \boldsymbol{B}_a^{at} \cup \boldsymbol{B}_p^{nat} \cup \boldsymbol{B}_a^{nat}$          ▷ joint training
9: end for
```

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Left blank.

☐ A2. Did you discuss any potential risks of your work?
Not applicable. Left blank.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Left blank.

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☒ Did you use or create scientific artifacts?

Left blank.

☐ B1. Did you cite the creators of artifacts you used?
No response.

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
No response.

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
No response.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
No response.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
No response.

☐ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
No response.

C ☑ Did you run computational experiments?

Left blank.

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Footnote 5 in section 3

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section 6

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 3

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Section 6

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?
Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.