HYBRID-REGRESSIVE NEURAL MACHINE TRANSLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

In this work, we empirically confirm that non-autoregressive translation with an iterative refinement mechanism (IR-NAT) suffers from poor acceleration robustness because it is more sensitive to decoding batch size and computing device setting than autoregressive translation (AT). Inspired by it, we attempt to investigate how to combine the strengths of autoregressive and non-autoregressive translation paradigms better. To this end, we demonstrate through synthetic experiments that prompting a small number of AT's predictions can promote one-shot non-autoregressive translation to achieve the equivalent performance of IR-NAT. Following this line, we propose a new two-stage translation prototype called hybrid-regressive translation (HRT). Specifically, HRT first generates discontinuous sequences via autoregression (e.g., make a prediction every k tokens, $k > 1$) and then fills in all previously skipped tokens at once in a non-autoregressive manner. We also propose a bag of techniques to effectively and efficiently train HRT without adding any model parameters. HRT achieves the state-of-the-art BLEU score of 28.49 on the WMT En→De task and is at least 1.5x faster than AT, regardless of batch size and device. In addition, another bonus of HRT is that it successfully inherits the good characteristics of AT in the deep-encoder-shallow-decoder architecture. Concretely, compared to the vanilla HRT with a 6-layer encoder and 6-layer decoder, the inference speed of HRT with a 12-layer encoder and 1-layer decoder is further doubled on both GPU and CPU without BLEU loss. [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 each time, resulting in a slow inference speed. To address this problem, non-autoregressive translation (NAT) attempts to generate the entire target sequence in parallel in one shot, assuming that the generation of target tokens is conditional independent Gu et al. (2018). While efficient, one-shot NAT suffers from severe translation quality degradation. How to achieve a better balance between inference speed and translation quality is still an active field for NAT Wang et al. (2018a); Ran et al. (2020); Qian et al. (2021); Huang et al. (2022b).

The iterative refinement mechanism first introduced by Lee et al. (2018), is one of the most successful approaches to this issue and has been 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, iterative refinement-based NAT (abbreviated as IR-NAT), also known as multi-shot NAT, "thinks more" than one-shot NAT: IR-NAT takes the translation hypothesis from the previous iteration as a reference and regularly polishes the new translation. The iteration stops when reaching the predefined iteration count I or no translation changed. A larger I generally improves translation accuracy while facing the risk of speedup degradation Kasai et al. (2020b).

In this work, we devote ourselves to understanding the limitations of IR-NAT and attempting to build a new fast-and-accurate translation paradigm beyond it. Our contributions are the following:

- We comprehensively study the acceleration robustness problem in IR-NAT and extend the finding of Kaiser et al. (2018) that IR-NAT is more sensitive not only to the decod-

[1] We will release the source code once accepted.

ing batch size but also to the computing device compared with AT. For example, when the decoding batch size is 1/8/16/32, the ten-iteration non-autoregressive model achieves 1.7x/1.2x/0.7x/0.4x inference speed of the AT model on GPU, respectively. However, when switching to CPU, the relative speed ratio drops to 0.8x/0.4x/0.3x/0.3x. These results prove that the two translation paradigms are well complementary to each other.

- We design synthetic experiments to investigate how much target context (i.e., the number of target tokens) is sufficient for one-shot NAT to rival multi-shot NAT. The value of the answer is that we could build the desired target context more cheaply, replacing expensive multiple iterations. Specifically, given a well-trained CMLM model, we notice that under the appropriate masking strategy, even if 70% of AT translations are masked, the remaining target context can help the CMLM$_1$ [2] with greedy search compete with the standard CMLM$_{10}$ with beam search (see Figure 2). To our best knowledge, we are the first to study the masking rate issue in the inference phase of NAT.
- Inspired by the observations above, we proposed a novel two-stage translation prototype – hybrid-regressive translation (HRT), to incorporate the advantages of AT and NAT. Concretely, HRT first uses an autoregressive decoder to generate a discontinuous target sequence with the interval k ($k > 1$). Then, HRT fills the remaining slots at once in a lightweight non-autoregressive manner. We propose to use a multi-task learning framework enhanced by curriculum learning and mixed distillation for effective and efficient training without adding any model parameters.
- Experimental results on WMT En↔Ro, En↔De, and NIST Zh→En show that HRT significantly outperforms prior work combining AT and NAT and has competitive BLEU with state-of-the-art IR-NAT models. Specifically, HRT achieves a BLEU score of 28.49 on the WMT En→De task and is robust 1.5x faster than AT regardless of batch size and device. Moreover, HRT equipped with deep-encoder-shallow-decoder architecture achieves up to 4x/3x acceleration on GPU/CPU, respectively, without BLEU loss.

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 dominant approach in NMT, which decomposes $P(\boldsymbol{y}|\boldsymbol{x})$ by chain rules: $P(\boldsymbol{y}|\boldsymbol{x}) = \prod_{t=1}^{N} P(y_t|\boldsymbol{x}, y_{<t})$, where $y_{<t}$ denotes the generated prefix translation before time step t. However, autoregressive models have to wait for the generation of y_{t-1} before predicting y_t, which hinders the parallelism over the target sequence.

Non-Autoregressive Translation (NAT) NAT allows generating all target tokens simultaneously Gu et al. (2018). NAT replaces $y_{<t}$ with target-independent input $\boldsymbol{z}$ and rewrites as: $P(\boldsymbol{y}|\boldsymbol{x}) = P(N|\boldsymbol{x}) \times \prod_{t=1}^{N} P(y_t|\boldsymbol{x}, \boldsymbol{z})$. We can model $\boldsymbol{z}$ as source embedding (Gu et al., 2018; Guo et al., 2019a), reordered source sentence (Ran et al., 2019), latent variable (Ma et al., 2019; Shu et al., 2019) etc.

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 as the IR-NAT representative in this work due to its excellent performance and simplification Ghazvininejad et al. (2019). 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). Denote $\boldsymbol{y}^m/\boldsymbol{y}^r$ as the masked/residual tokens of $\boldsymbol{y}$, then 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 repeated until $\hat{\boldsymbol{y}}^{(i-1)}$=$\hat{\boldsymbol{y}}^{(i)}$ or i reaches the maximum iteration count.

[2] We use the subscript to denote the number of iteration count.

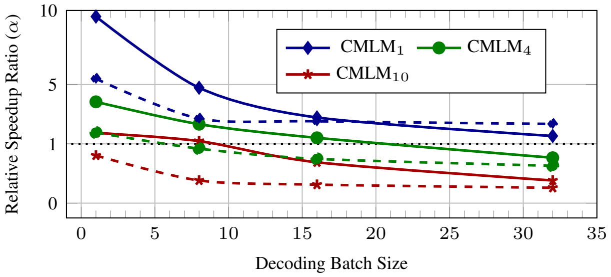


Figure 1: Relative speedup ratio (α) compared CMLM with AT on GPU (solid) and CPU (dashed). We test different batch sizes from $\{1, 8, 16, 32\}$. $\alpha < 1$ denotes that CMLM runs slower than AT. CMLM$_1$ can be approximately regarded as the representative of one-shot NAT.

3 Acceleration Robustness Problem

In this section, we attempt to comprehensively understand the inference acceleration robustness problem in IR-NAT. Without loss of generality, we take CMLM as the agency of IR-NAT.[3]

Problem Description The inference overhead of the autoregressive translation model mainly concentrates on the decoder sideHu 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. For convenience, we omit H due to its invariance in NAT and AT. Thus, the total cost of AT model is about $C_{at} \propto N \times \mathcal{O}(B \times 1)$ [4]. Likely, the cost of I-iteration NAT is $C_{nat} \propto I \times \mathcal{O}(B \times N)$. Given a fixed test set, We use $\mathcal{T}_D(\cdot)$ to represent the translation time on computing device D. In this way, we get the relative speedup ratio α between I-iteration NAT and AT by:

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

where $\mathcal{E}(B, D)$=$\frac{\mathcal{T}_D(\mathcal{O}(B\times1))}{\mathcal{T}_D(\mathcal{O}(B\times N))} \leq 1$, 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 when B or D change.

Setup To systematically investigate this problem, we compared the actual inference speed of CMLM [5] and AT under varying environments, including batch size $B \in \{1, 8, 16, 32\}$, device $D \in \{\text{GPU}, \text{CPU}\}$ [6], and the number of iteration $I \in \{1, 4, 10\}$. We use a beam size of 5 in all experiments. 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, where we can see that:

i. α decreases as decoding batch size increases regardless of the number of iterations, which is in line with Kasai et al. (2020b).

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

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

More concretely, the ten-iteration non-autoregressive model is 70% faster than the autoregressive counterpart when decoding a single sentence on GPU. In contrast, the IR-NAT model only reaches

[3] 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 long enough target sequence for inference.

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

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

[6] Unless otherwise stated, we use 2080Ti GPU and Intel Xeon(R) E5-2683 v4 CPU in this work.

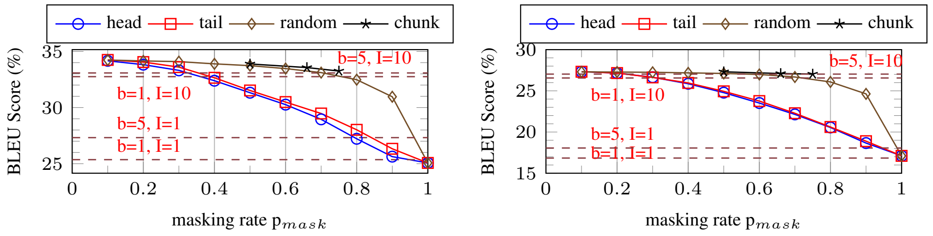


Figure 2: Comparison of four masking strategies {HEAD, TAIL, RANDOM, CHUNK} in synthetic experiments on WMT En→Ro (Left) and En→De (Right) test sets. For CHUNK, we test the chunk size from {2, 3, 4}. Dashed lines are the official CMLM scores. b stands for “beam size,” and I stands for “the number of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

30% inference speed of the AT model when switching to batches of 32 on CPU. These results indicate that the two translation paradigms enjoy different decoding setups and complement each other. Therefore, combining the advantages of AT and NAT could be an effective way for robust acceleration.

4 Synthetic Experiments

According to Equation 1, we know that reducing the iteration count I helps to increase α. Recalling the refinement process of IR-NAT, we think the essence of multiple iterations is to provide the decoder with a good enough target context (deterministic target tokens). Thus, an unexplored question raises that *how many target tokens need to be provided to make one-shot NAT compete with IR-NAT*? In this section, we try to answer this question through synthetic experiments on WMT En→Ro and En→De. Specifically, we control the size of the target context by masking the partial translations generated by a pre-trained AT model. Then we use a pre-trained CMLM model to predict these masks. Finally, we observe the BLEU score curves under different masking rates.

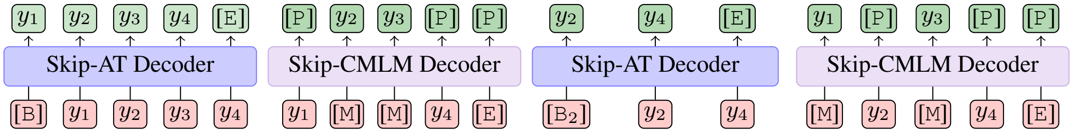
Models We use the official CMLM models. Since the authors did not release the AT baselines, we used the same data to retrain AT models with the standard Transformer-Base configuration (Vaswani et al., 2017) and obtain comparable performance with theirs (see Appendix B for details).

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 $C = \text{Ceil}(N/k)$ 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 are illustrated in Figure 2, where we can see that CHUNK is moderately but consistently superior to RANDOM and both of them significantly outperform HEAD and TAIL. We attribute the success of CHUNK to two aspects: the use of (1) bidirectional context (Devlin et al., 2019) (vs. HEAD and TAIL); (2) uniformly distributed deterministic tokens (vs. RANDOM) [7]. In addition, when using the CHUNK strategy, we find that exposing 30% AT tokens

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



(a) TASK-AT (b) TASK-CMLM (c) TASK-SKIP-AT (d) TASK-SKIP-CMLM

Figure 3: Examples of training samples for four tasks, in which (a) and (b) are auxiliary tasks and (c) and (d) are primary tasks. For the sake of clarity, we omit the source sequence. [B]/[E]/[P]/[M] represents the special token for [BOS]/[EOS]/[PAD]/[MASK], respectively. [B$_2$] is the [BOS] for k=2. Loss at [P] is ignored.

as the input of the decoder is sufficient to make our CMLM$_1$(beam=1) compete with the official CMLM$_{10}$(beam=5), which indicates the importance of a good partial target context.

5 Hybrid-Regressive Translation

Inspired by the observations above, we propose a novel two-stage translation paradigm called hybrid-regressive translation (HRT) to imitate the process of CHUNK. Briefly speaking, HRT autoregressively generates a discontinuous sequence with chunk size k (stage I), and then non-autoregressively fills the skipped tokens (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 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 its by-product: N_{nat}=$k \times N_{at}$, where N_{at} is the sequence length produced by Skip-AT . Compared to most NAT models that jointly train the translation length predictor and the translation model, our approach benefits in two aspects: (1) There is no need to carefully tune the weighting coefficient between the sentence-level length prediction loss and the word-level target token prediction loss; (2) The length predicted by Skip-AT could be more accurate because it can access the already generated sequence information.

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 through joint training of four tasks, including 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. It should be noted that TASK-SKIP-AT shrinks the sequence length from N to N/k compared to TASK-AT, while the token positions follow 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) instead of (0, 1, 2). Involving auxiliary tasks is necessary because the two primary tasks cannot fully leverage all tokens in the sequence due to the fixed k. For example, in Figure 3 (c) and (d), y_1 and y_3 have no chance to be learned as the decoder input of either TASK-SKIP-AT or TASK-SKIP-CMLM.

Method	Generation
SAT	$\{a, b\} \rightarrow \{c, d\} \rightarrow \{e, f\}$
RecoverSAT	$\{a, c, e\} \rightarrow \{b, d, f\}$
LAT	{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	34.09	28.34	**2.1**	**2.8**
5	1	34.24	28.49	1.7	1.6
5	5	**34.36**	**28.65**	N/A	1.1

Table 2: 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.

Curriculum Learning To ensure that 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). More concretely, given a batch of original sentence pairs $\mathcal{B}$ and let the proportion of primary tasks in $\mathcal{B}$ be p_k, we start with p_k=0 and construct the training samples of `TASK-AT` and `TASK-CMLM` for all pairs. Then we gradually increase p_k to introduce more learning signals for `TASK-SKIP-AT` and `TASK-SKIP-CMLM` until p_k=1. In implementation, we schedule p_k by:

$$p_k = (t/T)^{\lambda}, \tag{2}$$

where t and T are the current and total training steps. λ is a hyperparameter, and we use λ=1 to increase p_k linearly for all experiments.

Mixed Distillation Conventional NAT models use smoother distillation data generated by AT models instead of raw data. However, using only distillation data may lose some important information (e.g., rare words) contained in the original data Ding et al. (2020). Recent WMT competitions have proved that mixing raw data with distillation data is an effective approach to improve the performance of autoregressive translation. Akhbardeh et al. (2021); Barrault et al. (2019; 2020). To combine the best of both worlds, we propose a simple but effective approach – Mixed Distillation (MixDistill) for non-autoregressive translation. During training, MixDistill randomly samples the target sentence from the raw version $\boldsymbol{y}$ with probability p_{raw} or its distillation version $\boldsymbol{y}^*$ with probability $1 - p_{raw}$, where p_{raw} is a hyperparameter [8]. Compared with recent studies, our method is easier to implement: HRT does not rely on external word alignment Ding et al. (2020), and also avoids the time-consuming bi-directional distillation process Ding et al. (2021). MixDistill makes the HRT model less prone to overfitting in simple tasks (e.g., WMT'16 En→Ro), but what we emphasize is that the performance of HRT does not depend entirely on it. See Table 6 for more details.

5.3 Inference

HRT adopts two-stage generation strategy. In the first stage, the Skip-AT decoder starts from [BOS$_k$] to autoregressively generate a discontinuous target sequence $\hat{\boldsymbol{y}}_{at} = (z_1, z_2, \ldots, z_m)$ with chunk size k until meeting [EOS]. Then we construct the input of Skip-CMLM decoder $\boldsymbol{y}_{nat}$ by appending $k - 1$ [MASK]s before every z_i. The final translation is generated by replacing all [MASK]s with the predicted tokens by the Skip-CMLM decoder with one iteration. If there are multiple [EOS]s existing, we truncate to the first [EOS]. Note that the beam size b$_{at}$ in Skip-AT can be different from the beam size b$_{nat}$ in Skip-CMLM, as long as b$_{at} \geq$ b$_{nat}$: We only feed the top b$_{nat}$ Skip-AT hypotheses to Skip-CMLM decoder. In the implementation, we use standard beam search in Skip-AT (b$_{at}$ >1) and greedy search in Skip-CMLM (b$_{nat}$=1) because b$_{nat}$=1 is sufficient to obtain competitive BLEU scores, thanks to the good context provided by Skip-AT. Table 2 gives a detailed discussion about the beam size setting in HRT. Finally, we choose the translation hypothesis with

[8] Training with only raw data or distillation data can be regarded as the special case of MixDistill as p_{raw}=1 or p_{raw}=0.

	System	Iterations	WMT'16		WMT'14	
			En-Ro	Ro-En	En-De	De-En
	Existing systems					
NAT	FCL-NAT (Guo et al., 2019b)	1	-	-	25.75	29.50
	FlowSeq (Ma et al., 2019)	1	32.20	32.84	25.31	30.68
	AXE (Ghazvininejad et al., 2020a)	1	30.75	31.54	23.53	27.90
	GLAT (Qian et al., 2021)	1	32.87	33.84	26.55	31.02
	Fully-NAT (Gu & Kong, 2021)	1	33.79	34.16	27.49	31.39
	DA-Transformer (Huang et al., 2022a)	1	-	-	27.91	31.95
Iterative NAT	CMLM (Ghazvininejad et al., 2019)	10	33.08	33.31	27.03	30.53
	LevT (Gu et al., 2019)	Adaptive	-	-	27.27	-
	JM-NAT (Guo et al., 2020)	10	33.52	33.72	27.69	32.24
	SMART (Ghazvininejad et al., 2020b)	10	-	-	27.65	31.27
	DisCO (Kasai et al., 2020a)	Adaptive	33.22	33.25	27.34	31.31
	Imputer (Saharia et al., 2020)	8	34.40	34.10	28.20	31.80
	RewriteNAT (Geng et al., 2021)	Adaptive	33.63	34.09	27.83	31.52
	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	-
	SynST (Akoury et al., 2019)	$N/6+1$	-	-	20.74†	25.50†
	ReorderNAT (Ran et al., 2019)	$N+1$	31.70	31.99	26.49	31.13
	RecoverSAT (Ran et al., 2020)	$N/2$	32.92	33.19	27.11	31.67
	LAT (Kong et al., 2020)	$4\times N/3$	32.87	33.26	27.35	32.04
	Our implementations					
	AT + Raw	N	34.25(34.2†)	34.40(34.0†)	27.45(26.9†)	31.86(31.6†)
	AT-20L + Raw (teacher for En↔De)	N	-	-	28.79(28.2†)	33.02(32.8†)
	SAT + MixDistill	$N/2$	-	-	26.67(26.2†)	-
	HRT + MixDistill	$N/2+1$	34.24(34.2†)	34.35(34.0†)	28.49(27.9†)	32.28(32.0†)
	HRT-20L + Mixstill	$N/2+1$	-	-	**29.14(28.6†)**	**33.21(32.9†)**

Table 3: The BLEU & SacreBLEU (denoted by †) scores of our proposed HRT and the baselines on four WMT tasks. Unless otherwise stated, we use a beam size of 5. "Adaptive" denotes dynamic iterations. "20L" stands for using a 20-layer encoder. The teacher models of En→Ro and Ro→En have BLEU scores of 34.28 and 33.99, respectively, obtained from Ghazvininejad et al. (2019).

the highest score $\mathrm{S}(\hat{\mathrm{y}})$ by:

$$\underbrace{\sum_{i=1}^{m} \log P(z_i|\boldsymbol{x}, \boldsymbol{z}_{<i})}_{\text{Skip-AT score}} + \underbrace{\sum_{i=0}^{m-1}\sum_{j=1}^{k-1} \log P(\hat{y}_{i\times k+j}|\boldsymbol{x}, \boldsymbol{y}_{nat})}_{\text{Skip-CMLM score}}, \text{ where } z_i = \hat{y}_{i\times k}. \tag{3}$$

5.4 Discussion

The basic idea of HRT is to apply AT and NAT in sequence, which has been investigated by Kaiser et al. (2018); Ran et al. (2019); Akoury et al. (2019). The main difference between these methods is the content of AT output, such as latent variable (Kaiser et al., 2018), reordered source token (Ran et al., 2019), and syntactic label (Akoury et al., 2019). In contrast, our approach uses the deterministic target token as Ghazvininejad et al. (2019). Coupling different decoding paradigms in one process is another line to incorporate AT and NAT. Table 1 shows the differences between HRT and existing methods, including SAT Wang et al. (2018a), RecoverSAT Ran et al. (2020) and LAT Kong et al. (2020). Although HRT needs more decoding steps than SAT and RecoverSAT, its non-autoregressive process is inexpensive due to the use of greedy search. In contrast, these three methods require larger beams to explore translations in different lengths.

6 Experimental Results

Setup We mainly conduct experiments on four widely used WMT tasks: WMT'16 English↔Romanian (En↔Ro, 610k) and WMT'14 English↔German (En↔De, 4.5M). We replicate the same data processing for fair comparisons as Ghazvininejad et al. (2019). To verify the effectiveness in long-distance language pairs, we also test it in the NIST Chinese-English (Zh→En, 1.8M) translation task following the setup of Wang et al. (2018b). We use the En↔Ro distillation data published by Ghazvininejad et al. (2019). However, for En↔De and Zh→En, we retrained the

Model	**MT04**	**MT05**	**MT08**
AT (teacher)	43.86	52.91	33.94
CMLM$_{10}$	42.47	52.16	33.09
HRT	**43.93**	**53.02**	**34.33**

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

Chunk	**Valid**	**Test**	α**(AG)**	α**(AC)**
2	**26.45**	**28.49**	1.7	1.6
3	26.22	27.98	2.5	2.3
4	25.56	27.17	**3.2**	**2.9**

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

System	**En→Ro**			**En→De**		
	R	**D**	**MD**	**R**	**D**	**MD**
AT	**34.25**	33.19	33.92	**27.45**	28.24	28.14
SAT	-	-	-	22.07	26.45	26.67
HRT	33.59	**33.20**	**34.24**	26.69	**28.30**	**28.49**

Table 6: The BLEU scores against different data strategies (R: Raw, D: Distill, MD: MixDistill) .

Skip-AT	**Skip-CMLM**	**BLEU**	**Δ**
HRT	HRT	28.49	ref.
HRT-20L	HRT-20L	29.14	+0.65
HRT	HRT-20L	28.58	+0.09
HRT-20L	HRT	29.03	+0.54

Table 7: Swapping Skip-AT and Skip-CMLM between HRT and HRT-20L.

AT teacher models because Ghazvininejad et al. (2019) did not release the corresponding distillation data. Specifically, we use the deep PreNorm Transformer-Base with a 20-layer encoder and the standard Transformer-Base as teacher models for En↔De and Zh→En, respectively. We run all experiments on four 2080Ti GPUs. Unless noted otherwise, we use the chunk size k=2. We set p$_{raw}$=0.5 for En↔Ro and Zh→En, p$_{raw}$=0.2 for En↔De according to validation sets. 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 [9].

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 acceleration ratio. We test three different setups, and the results are listed in Table 2. 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 fair comparison with baselines [10], we use b$_{at}$=5 and b$_{nat}$=1 unless otherwise stated.

Main Results Table 3 reports the BLEU and SacreBLEU scores on four WMT tasks. Our HRT outperforms most existing NAT, IR-NAT, and Semi-NAT models and establishes new state-of-the-art results on En→De (See Appendix D for case studies.). Compared to our re-implemented SAT with the same MixDistill, HRT obtains an improvement of +2.0 BLEU points. We compared different data strategies in more detail in Section 7. Besides, in line with Guo et al. (2020), when using a deeper encoder, HRT-20L can further improve by approximately +0.5 BLEU. We highlight that HRT can achieve equivalent or even better performance than the teacher model when having the same model capacity. In Table 4, we also report the experimental results on the Zh→En task. HRT is once again superior to the original AT and CMLM model, which indicates that the effectiveness of HRT is agnostic to language pairs.

7 Analysis

Impact of Chunk Size We tested chunk size k on the En→De task as shown in Table 5, where we can see that: (1) A large k has a more significant speedup on the GPU because fewer autoregressive steps are required; (2) As k increases, the performance of HRT drops sharply. For example, k=4 is about 1.32 BLEU points lower than k=2 on the test set. It indicates that the training difficulty of Skip-AT increases as k becomes larger. We think that skip-generation may require more fancy training algorithms, which is left for our future work.

Data Strategy As shown in Table 6, we compared three data strategies, including raw data (`R`), sequence-level knowledge distillation (`D`), and proposed mixed distillation (`MD`). Overall, `MD` is superior to other methods across the board, indicating that training with raw and distillation data is

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

[10] Most prior related work uses beam size of 5.

Model	MD	BLEU	α(AG)	α(AC)
AT-20L (teacher)	✗	28.87	-	-
AT	✓	28.14	ref.	ref.
AT$_{12-1}$	✓	28.18	2.7	2.1
HRT	✓	**28.49**	1.7	1.6
HRT$_{12-1}$	✓	28.44	**4.2**	**3.1**

Table 8: Effects of deep-encoder-shallow-decoder architecture on En→De test set.

System	BLEU	Δ
HRT (T=300k)	28.49	ref.
$-$FT	28.21	-0.28
$-$MD (p$_{raw}$=0)	28.30	-0.19
$-$CL (p_k=1)	27.72	-0.77
$-$CL (p_k=0.5)	27.93	-0.56
$-$TS (T=100k)	27.91	-0.58
$-$ALL	26.96	-1.53

Table 9: Ablation study on En→De task.

complementary. We emphasize that HRT's strong performance does not mainly come from MD. For example, even if using D, HRT can achieve excellent results on the En→De task. More surprisingly, we found that HRT can be slightly better than AT models in the same network scale trained by D. We attribute it 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.

Importance of Skip-AT & Skip-CMLM We try to understand the importance of Skip-AT and Skip-CMLM for HRT. To this end, we swap the intermediate results of two different HRT models (refer to Model A and B, respectively). Specifically, we first use the Skip-AT decoder of A to generate its discontinuous target sequence. Then we let B's Skip-AT decoder force decoding this sequence to obtain corresponding encoding representations and autoregressive model scores. Finally, B uses its Skip-CMLM decoder to generate the complete translation result according to them. The order of A and B can be exchanged. In practice, we use two models (HRT and HRT-20L) with large performance gap as A and B respectively. As shown in Table 7, we find that using the strong model's Skip-AT brings more improvement (+0.54 BLEU) than using its Skip-CMLM (+0.09 BLEU). This result aligns with our claim that a good partial target context is critical.

Deep-encoder-shallow-decoder Architecture Kasai et al. (2020b) point out that AT with deep-encoder-shallow-decoder architecture can substantially speed up without losing translation accuracy. We also compare HRT and AT in this setting: 12-layer encoder and 1-layer decoder, denoted by HRT$_{12-1}$ and AT$_{12-1}$, respectively. AT$_{12-1}$ and HRT$_{12-1}$ use the same mixed distillation data for a fair comparison. As can be seen from Table 8, both AT$_{12-1}$ and HRT$_{12-1}$ benefit from the change of layer allocation, gaining comparable BLEU scores and double decoding speed as the vanilla models. Specifically, HRT$_{12-1}$ achieves an average acceleration of 4.2x/3.1x than AT baselines. Kasai et al. (2020b) report that CMLM does not enjoy this architecture, which indicates the success of HRT$_{12-1}$ comes from using Skip-AT instead of Skip-CMLM.

Ablation Study In Table 9, we conduct ablation studies on the En→De task, including fine-tuning from pre-trained AT (FT), mixed distillation (MD), 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. All components contribute to the performance, but CL(p_k=1) and TS are the most critical. We also try to exclude them all from the vanilla HRT (-ALL), resulting in a total reduction of 1.53 BLEU points.

8 Conclusion

We pointed out that IR-NAT suffers from inference acceleration robustness problems. Inspired by our findings in synthetic experiments, we proposed HRT to combine the advantages of AT and NAT. Experimental results show that our approach outperforms existing Semi-AT methods and promises to be a good substitute for AT due to competitive performance and stable speedup.

References

Farhad Akhbardeh, Arkady Arkhangorodsky, Magdalena Biesialska, Ondřej Bojar, Rajen Chatterjee, Vishrav Chaudhary, Marta R. Costa-jussa, Cristina España-Bonet, Angela Fan, Christian Federmann, Markus Freitag, Yvette Graham, Roman Grundkiewicz, Barry Haddow, Leonie Harter, Kenneth Heafield, Christopher Homan, Matthias Huck, Kwabena Amponsah-Kaakyire, Jungo Kasai, Daniel Khashabi, Kevin Knight, Tom Kocmi, Philipp Koehn, Nicholas Lourie, Christof Monz, Makoto Morishita, Masaaki Nagata, Ajay Nagesh, Toshiaki Nakazawa, Matteo Negri, Santanu Pal, Allahsera Auguste Tapo, Marco Turchi, Valentin Vydrin, and Marcos Zampieri. Findings of the 2021 conference on machine translation (WMT21). In Proceedings of the Sixth Conference on Machine Translation, pp. 1–88, Online, November 2021. Association for Computational Linguistics. URL https://aclanthology.org/2021.wmt-1.1.

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

Loïc Barrault, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, Shervin Malmasi, Christof Monz, Mathias Müller, Santanu Pal, Matt Post, and Marcos Zampieri. Findings of the 2019 conference on machine translation (WMT19). In Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1), pp. 1–61, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-5301. URL https://aclanthology.org/W19-5301.

Loïc Barrault, Magdalena Biesialska, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Yvette Graham, Roman Grundkiewicz, Barry Haddow, Matthias Huck, Eric Joanis, Tom Kocmi, Philipp Koehn, Chi-kiu Lo, Nikola Ljubešić, Christof Monz, Makoto Morishita, Masaaki Nagata, Toshiaki Nakazawa, Santanu Pal, Matt Post, and Marcos Zampieri. Findings of the 2020 conference on machine translation (WMT20). In Proceedings of the Fifth Conference on Machine Translation, pp. 1–55, Online, November 2020. Association for Computational Linguistics. URL https://aclanthology.org/2020.wmt-1.1.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In Proceedings of the 26th Annual International Conference on Machine Learning, ICML '09, pp. 41–48, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605585161. doi: 10.1145/1553374.1553380. URL https://doi.org/10.1145/1553374.1553380.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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), pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://www.aclweb.org/anthology/N19-1423.

Liang Ding, Longyue Wang, Xuebo Liu, Derek F. Wong, Dacheng Tao, and Zhaopeng Tu. Understanding and improving lexical choice in non-autoregressive translation. arXiv: Computation and Language, 2020.

Liang Ding, Xuebo Liu Longyue Wang, Derek F. Wong, Dacheng Tao, and Zhaopeng Tu. Rejuvenating low-frequencywords: Making the most of parallel data in non-autoregressive translation. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics, 2021.

Xinwei Geng, Xiaocheng Feng, and Bing Qin. Learning to rewrite for non-autoregressive neural machine translation. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pp. 3297–3308, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. URL https://aclanthology.org/2021.emnlp-main.265.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 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), pp. 6111–6120, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1633. URL https://www.aclweb.org/anthology/D19-1633.

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

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

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

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O.K. Li, and Richard Socher. Non-autoregressive neural machine translation. In International Conference on Learning Representations, 2018. URL https://openreview.net/forum?id=B1l8BtlCb.

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

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

Junliang Guo, Xu Tan, Linli Xu, Tao Qin, Enhong Chen, and Tie-Yan Liu. Fine-tuning by curriculum learning for non-autoregressive neural machine translation. arXiv preprint arXiv:1911.08717, 2019b.

Junliang Guo, Linli Xu, and Enhong Chen. 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, pp. 376–385, 2020.

Chi Hu, Bei Li, Yinqiao Li, Ye Lin, Yanyang Li, Chenglong Wang, Tong Xiao, and Jingbo Zhu. The NiuTrans system for WNGT 2020 efficiency task. In Proceedings of the Fourth Workshop on Neural Generation and Translation, pp. 204–210, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.ngt-1.24. URL https://aclanthology.org/2020.ngt-1.24.

Fei Huang, Hao Zhou, Yang Liu, Hang Li, and Minlie Huang. Directed acyclic transformer for non-autoregressive machine translation. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), Proceedings of the 39th International Conference on Machine Learning, volume 162 of Proceedings of Machine Learning Research, pp. 9410–9428. PMLR, 17–23 Jul 2022a. URL https://proceedings.mlr.press/v162/huang22m.html.

Xiao Shi Huang, Felipe Perez, and Maksims Volkovs. Improving non-autoregressive translation models without distillation. In International Conference on Learning Representations, 2022b. URL https://openreview.net/forum?id=I2Hw58KHp8O.

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

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

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

Xiang Kong, Zhisong Zhang, and Eduard Hovy. Incorporating a local translation mechanism into non-autoregressive translation. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1067–1073, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.79. URL https://aclanthology.org/2020.emnlp-main.79.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pp. 1173–1182, Brussels, Belgium, October-November 2018. doi: 10.18653/v1/D18-1149. URL https://www.aclweb.org/anthology/D18-1149.

Xuezhe Ma, Chunting Zhou, Xian Li, Graham Neubig, and Eduard Hovy. 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), pp. 4273–4283, 2019.

Lihua Qian, Hao Zhou, Yu Bao, Mingxuan Wang, Lin Qiu, Weinan Zhang, Yong Yu, and Lei Li. Glancing transformer for non-autoregressive neural machine translation. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pp. 1993–2003, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.155. URL https://aclanthology.org/2021.acl-long.155.

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

Qiu Ran, Yankai Lin, Peng Li, and Jie Zhou. 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, pp. 3059–3069, Online, July 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.acl-main.277.

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

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Advances in Neural Information Processing Systems, pp. 6000–6010, 2017.

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

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

Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F. Wong, and Lidia S. Chao. Learning deep transformer models for machine translation. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp. 1810–1822, Florence, Italy, July 2019. URL https://www.aclweb.org/anthology/P19-1176.

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→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 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, the performance of HRT$_{12-1}$ approaches the one-shot NAT (CMLM$_1$) on both GPU and CPU. Besides, the overall results of HRT-20L is 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.

Model	BLEU↑	B=1		B=8		B=16		B=32		Avg
		Time↓	$\alpha\uparrow$	Time↓	$\alpha\uparrow$	Time↓	$\alpha\uparrow$	Time↓	$\alpha\uparrow$	$\alpha\uparrow$
					On GPU					
AT	27.45	857.2	1.0	137.8	1.0	73.1	1.0	40.1	1.0	1.0
AT$_{12-1}$	28.18	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	26.45	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)	28.34	377.5	2.3	66.4	2.1	34.9	2.1	20.5	2.0	2.1
HRT	28.49	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)	28.65	482.4	1.8	81.2	1.7	46.5	1.6	N/A	N/A	N/A
HRT$_{12-1}$	28.44	192.5	4.6	31.4	4.3	**18.4**	**4.0**	**11.1**	**3.7**	4.2
HRT (k=3)	27.98	323.9	2.6	54.9	2.5	29.5	2.5	18.2	2.2	2.5
HRT (k=4)	27.17	256.0	3.3	43.1	3.2	23.3	3.1	12.7	3.2	3.2
					On CPU					
AT	27.45	1118.0	1.0	314.1	1.0	246.3	1.0	201.3	1.0	1.0
AT$_{12-1}$	28.18	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	26.45	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)	28.34	457.1	2.4	116.1	2.7	**82.4**	**3.0**	**65.9**	**3.1**	2.8
HRT	28.49	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)	28.65	811.0	1.4	294.5	1.1	247.6	1.0	235.2	0.9	1.1
HRT$_{12-1}$	28.44	249.6	4.5	111.5	2.8	85.1	2.9	83.9	2.4	3.1
HRT (k=3)	27.98	448.7	2.5	134.8	2.3	111.7	2.2	90.7	2.2	2.3
HRT (k=4)	27.17	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.

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→Ro and En→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). Each training sample $\boldsymbol{B}_i$ randomly selects a raw target sentence $\boldsymbol{Y}_i$ or its distilled version $\boldsymbol{Y}'$ (Line 4-6). Then according to the schedule strategy $p_k = \left(\frac{t}{T}\right)^{\lambda}$, we can divide $\boldsymbol{B}$ 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 7-8). 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.

D Case study

Table 12 shows a translation example from En→De validation set. Comparing CMLM and HRT, although both have the same masking rate (50%), we can see two main differences: (1) The distribution of masked tokens in CMLM is more discontinuous than in HRT (see blue marks); (2) The

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 $D$ including distillation targets, pretrained AT model $\mathrm{M}_{at}$, chunk size $k$, mixed distillation rate $p_{raw}$, schedule coefficient $\lambda$
Output: Hybrid-Regressive Translation model $\mathrm{M}_{hrt}$
$\mathrm{M}_{hrt} \leftarrow \mathrm{M}_{at}$    ▷ fine-tune on pre-trained AT
for $t$ in $1, 2, \ldots, T$ do
    $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}, \boldsymbol{Y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n\}, \boldsymbol{Y}' = \{\boldsymbol{y}'_1, \ldots, \boldsymbol{y}'_n\} \leftarrow$ fetch a batch from $D$
    for $i$ in $1, 2, \ldots, n$ do
        $\boldsymbol{B}_i = (\boldsymbol{X}_i, \boldsymbol{Y}^*_i) \leftarrow$ uniformly sampling $\boldsymbol{Y}^*_i \sim \{\boldsymbol{Y}_i, \boldsymbol{Y}'_i\}$ with $P(\boldsymbol{Y}_i) = p_{raw}$    ▷ mixed distillation
    end for
    $p_k \leftarrow (\frac{t}{T})^{\lambda}$    ▷ curriculum learning
    $\boldsymbol{B}_p, \boldsymbol{B}_a \leftarrow \boldsymbol{B}\big[ : \lfloor n \times p_k \rfloor\big], \boldsymbol{B}\big[\lfloor n \times p_k \rfloor : \big]$    ▷ split batch
    $\boldsymbol{B}^{at}_p, \boldsymbol{B}^{nat}_p \leftarrow$ construct training samples of primary tasks based on $\boldsymbol{B}_p$
    $\boldsymbol{B}^{at}_a, \boldsymbol{B}^{nat}_a \leftarrow$ construct training samples of auxiliary tasks based on $\boldsymbol{B}_a$
    Optimize $\mathrm{M}_{hrt}$ using $\boldsymbol{B}^{at}_p \cup \boldsymbol{B}^{at}_a \cup \boldsymbol{B}^{nat}_p \cup \boldsymbol{B}^{nat}_a$    ▷ joint training
end for
```

decoder input of HRT contains more correct target tokens than CMLM, thanks to the Skip-AT decoder (see wavy marks). These two differences make our model easier to generate good translations than CMLM. It also indicates that our model is capable of generating 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 12: 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 as its masking rate is 50%, closing to that of HRT.