Hybrid-Regressive Neural Machine Translation

Anonymous ACL submission

Abstract

Non-autoregressive translation (NAT) with iterative refinement mechanism has shown comparable performance with the auto-regressive counterpart. However, we have empirically found that decoding acceleration is fragile when using a large batch size and running on the CPU. We demonstrate that one-pass NAT is sufficient when providing a few target contexts in advance through synthetic experiments. Inspired by this, we propose a two-stage translation prototype – Hybrid-Regressive Translation (HRT) to combine the strengths of autoregressive and non-autoregressive. Specifically, HRT first generates a discontinuous sequence by autoregression (e.g., make a prediction every k tokens, $k > 1$) and then fills all previously skipped tokens at once in a non-autoregressive manner. We also propose a bag of techniques to effectively and efficiently train HRT, with almost no increase in parameters. Experimental results on WMT En↔Ro, En↔De, and NIST Zh→En show that our model outperforms existing semi-autoregressive models and is competitive with current state-of-the-art non-autoregressive models. Moreover, compared to its autoregressive counterpart, HRT has a stable 1.5x acceleration, regardless of batch size and device [1].

1 Introduction

Recently, increasing attention has been paid to accelerating the autoregressive Transformer (Vaswani et al., 2017) decoding for Neural Machine Translation, such as network architecture design (Zhang et al., 2018; Xiao et al., 2019; Kasai et al., 2020b), model compression (Kim and Rush, 2016; Lin et al., 2021; Li et al., 2021), quantization (Bhandare et al., 2019; Lin et al., 2020) etc. Unlike them, non-autoregressive translation (NAT) (Gu et al., 2017) attempts to circumvent the slow autoregressive translation (AT) by predicting the target sequence in parallel in one shot. Although attractive, early one-shot NAT usually suffers from severe translation quality degradation due to the lack of necessary target word dependencies (Gu et al., 2017).

Aimed at this issue, researchers have proposed many approaches, such as optimizing the training objective (Li et al., 2019; Shao et al., 2019; Wang et al., 2019b; Sun et al., 2019; Qian et al., 2021), enhancing the decoder input (Guo et al., 2019a; Ma et al., 2019) etc. Beyond that, another route is to combine the strengths of autoregressive translation and non-autoregressive, called semi-autoregressive translation (Semi-AT) (Wang et al., 2018a; Kaiser et al., 2018; Akoury et al., 2019; Ran et al., 2019, 2020). However, there is still a noticeable gap between the above methods and AT in terms of BLEU despite the effectiveness. Perhaps the most promising solution is to extend the one-shot NAT by introducing an iterative refinement mechanism (IR-NAT). Concretely, IR-NAT takes the translation hypothesis from the previous iteration as a reference and regularly polishes the new translation until reaching the predefined iteration count I or no translation changed. A common brief is that IR-NAT can run faster than AT and have comparable translation accuracy.

In this work, we continue the line of research and go towards a fast and accurate translation paradigm. Our contributions are threefold:

- We analyze and empirically reveal that *increasing batch size* and/or *running on CPU* can significantly reduce the efficiency of parallel computation, resulting in severe acceleration degradation of IR-NAT. In contrast, AT is less sensitive. For example, when decoding with a batch size of 32 on CPU, the IR-NAT model (i.e., CMLM (Ghazvininejad et al., 2019)) with 10 iterations even runs 3x slower than auto-regressive (cf. Figure 1).

- We designed a synthetic experiment to demon-

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

strate that iterative decoding is unnecessary when providing a good (partial) target context. Specifically, given a well-trained CMLM model, we notice that under the appropriate masking strategy, even if 70% of the AT translation is masked, the remaining target context can help the CMLM(beam=1, I=1) compete with the standard CMLM(beam=5, I=10) (see Figure 2).

- We proposed a two-stage translation prototype – Hybrid-Regressive Translation (HRT). Concretely, HRT first uses an autoregressive decoder to generate a discontinuous target sequence with the interval k. Then, HRT fills these remaining slots at once in a non-autoregressive manner. We further propose joint training guided by curriculum learning and mixed distillation to effectively and efficiently train HRT.

Experimental results on WMT En↔Ro, En↔De, and NIST Zh→En show that HRT has a large BLEU improvement compared with previous Semi-AT methods and can compete with the state-of-the-art IR-NAT models. Moreover, HRT has a consistent 50% decoding speedup compared with the autoregressive counterparts regardless of batch sizes and devices.

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) is the dominant approach in NMT, which decomposes $P(\boldsymbol{y}|\boldsymbol{x})$ by chain rules:

$$P(\boldsymbol{y}|\boldsymbol{x}) = \prod_{t=1}^{N} P(y_t|\boldsymbol{x}, y_{<t}) \quad (1)$$

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 parallel computation over the target sequence.

Non-autoregressive translation (NAT) allows generating all target tokens simultaneously (Gu et al., 2017). NAT replaces $y_{<t}$ with target-independent input $\boldsymbol{z}$ and rewrites Eq. 1 as:

$$P(\boldsymbol{y}|\boldsymbol{x}) = P(N|\boldsymbol{x}) \prod_{t=1}^{N} P(y_t|\boldsymbol{x}, \boldsymbol{z}) \quad (2)$$

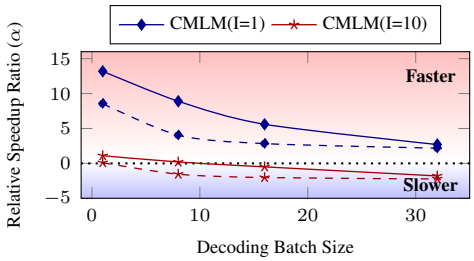


Figure 1: Relative speedup ratio (α) compared CMLM with AT on GPU (solid) and CPU (dashed). The value of α denotes running faster (positive) or slower (negative) $|\alpha|$ times than AT. The beam size is 5.

We can model $\boldsymbol{z}$ as source embedding (Gu et al., 2017; 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) extends the traditional one-shot NAT by introducing a multi-round decoding mechanism (Lee et al., 2018; Ghazvininejad et al., 2019; Gu et al., 2019; Ghazvininejad et al., 2020b). We choose CMLM as IR-NAT representative in this work 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. Denote $\boldsymbol{y}^m/\boldsymbol{y}^r$ as the masked/residual tokens of $\boldsymbol{y}$, then we have:

$$P(\boldsymbol{y}|\boldsymbol{x}) = \prod_{t=1}^{|\boldsymbol{y}^m|} P(\boldsymbol{y}_t^m|\boldsymbol{x}, \boldsymbol{y}^r) \quad (3)$$

At inference, CMLM deterministically masks tokens from the hypothesis in the previous iteration $\hat{\boldsymbol{y}}^{(i-1)}$ according to the prediction confidence. This process is repeated until $\hat{\boldsymbol{y}}^{(i-1)} = \hat{\boldsymbol{y}}^{(i)}$ or i reaches the maximum iteration count.

3 Acceleration Degradation in IR-NAT

This section empirically compares the practical inference speed of CMLMs and autoregressive models on different batch sizes (1, 8, 16, 32) and devices (GPU, CPU) to demonstrate the speed degradation problem.

Setup Inference speed is meassured on the widely used WMT En→De *newstest2014* test set with a beam size of 5. We use the official CMLM models released by Ghazvininejad et al. (2019) [2]. Unless otherwise stated, we use TITAN X (Pascal)

[2] https://github.com/facebookresearch/Mask-Predict

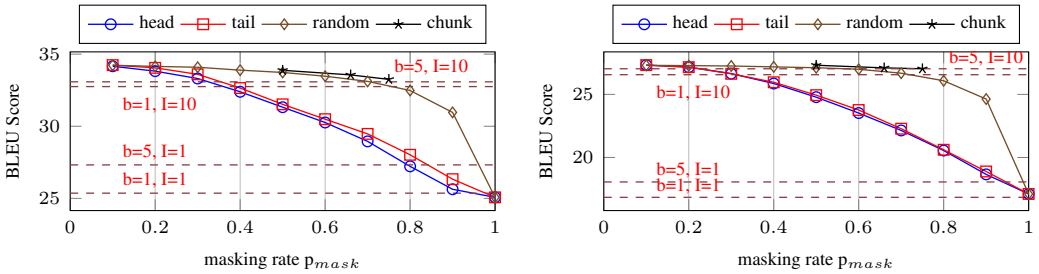


Figure 2: Comparison of four masking strategies {`Head`, `Tail`, `Random`, `Chunk`} in synthetic experiments on WMT En→Ro (Left) and En→De (Right) test sets. For `Chunk`, we test the chunk size from {2, 3, 4}. Dashed lines represent Mask-Predict's scores reported by Ghazvininejad et al. (2019). b stands for "beam size" while I stands for "the number of iterations".

GPU and Intel Xeon(R) E5-2680 v3 @ 2.50GHz CPU in this work. We measure all speeds five times and report the average value.

Results As illustrated in Figure 1, we can see that: (1) Relative speedup ratio (CMLM/AT) decreases as the increase of decoding batch size regardless of the number of iterations; (2) The speed on the CPU is consistently worse than that on the GPU; (3) Unlike the single iteration CMLM that can achieve stable acceleration, the CMLM with I=10 is even three times slower than autoregressive when decoding with a batch size of 32 on the CPU.

Analysis Suppose that the computational cost is proportional to the size of decoder input tensor $(BH \times BM, N, H)$, where BH is the batch size, BM is the beam size, and H is the network dimension. For convenience, we omit BM and H due to their invariance in NAT and AT. Thus, the total cost of I-iterations NAT is $C_{nat} \propto I \times \mathcal{O}(BH \times N)$. Similarly, the cost of AT model is about $C_{at} \propto L \times \mathcal{O}(BH \times 1)$ [3]. We use $\mathcal{T}(\cdot)$ to represent the elapsed time. In this way, we can denote the relative speedup ratio α as $\alpha = \frac{\mathcal{T}(C_{at})}{\mathcal{T}(C_{nat})} \propto \frac{N}{I} \times \mathcal{E}$, where $\mathcal{E} = \frac{\mathcal{T}(\mathcal{O}(BH \times 1))}{\mathcal{T}(\mathcal{O}(BH \times N))} \leq 1$. Therefore, it is easy to determine that fewer iterations I and more efficient parallel computation (larger $\mathcal{E}$) are the keys to the acceleration in IR-NAT. Unfortunately, due to hardware limitations, $\mathcal{E}$ decreases significantly as the computation cost (BH) increases (see Appendix A for details). For example, when BH increases from 1 to 32, AT's decoding latency on the GPU for a fixed test set (En→De *newstest14*) reduces by 22.4 times, while CMLM (I=10) only reduces by 3.6 times [4]. If $\mathcal{E}$ is too small, the advantage of parallel generation will disappear. And once $\mathcal{E} < \frac{N}{I}$, NAT will perform slower than AT.

4 Synthetic Experiments

One way to alleviate the problem above is to reduce the iteration count. To this end, we design a synthetic experiment on WMT En→Ro and En→De to study *how much target context is needed to make one-shot NAT compete with IR-NAT*? We change the target context by masking the translation generated by the pre-trained AT model.

Models We use the official CMLM models (Ghazvininejad et al., 2019). 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*> 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, $\lfloor N \times p_{mask} \rfloor$). Then `Head`/`Tail` always masks the first/last N_{mask} tokens, while `Random` masks the translation randomly. `Chunk` is slightly different from the above

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

[4] We observed similar results on CPU.

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}. To exclude randomness, we ran `Random` three times with different seeds and report the average results.

Results The experimental results are illustrated in Figure 2, where we can see that *balanced bidirectional context is optimal*. Specifically, `Chunk` is moderately but consistently superior to `Random` and both of them significantly outperform `Tail` and `Head`. We attribute the success of `Chunk` to two aspects: (1) The use of bidirectional context (compared to `Head` and `Tail` (Devlin et al., 2019)); (2) Uniformly distributed deterministic tokens (compared to `Rand`) [5]. In addition, when using the `Chunk` strategy, exposing 30% AT tokens as the input of the decoder is sufficient to make our CMLM(beam=1, I=1) compete with the official CMLM(beam=5, I=10), which indicates the importance of a good partial target context.

5 Hybrid-Regressive Translation

Inspired by the `Chunk` strategy's success, we propose a two-stage translation paradigm called Hybrid-Regressive Translation (HRT). Briefly speaking, HRT autoregressively generates a discontinuous sequence with chunk size k (stage I), and then non-autoregressively fills the skipped tokens (stage II). The idea of HRT is similar to SynST (Akoury et al., 2019) that carries out AT and NAT sequentially, but HRT does not require any additional supervision from the external parser tree.

5.1 Architecture

Overview Our 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). To make the single model compatible with the generation of continuous and discontinuous sequences simultaneously, we additionally equip each decoder self-attention sublayer with a simplified relative position representation (SRPR) (Shaw et al., 2018) for the awareness of word position information [6]. The two decoders have the same network structure and share model parameters, leading to almost the same parameter size as the standard CMLM (except for a few parameters in SRPR). The only difference between the two decoders lies in the masking pattern in the self-attention sublayer: The Skip-AT decoder masks future tokens to guarantee strict left-to-right generation. In contrast, the Skip-CMLM decoder eliminates it to leverage the bi-directional context.

No target length predictor Existing NAT models generally train the translation model with an independent translation length predictor. However, just like previous Semi-AT models (Wang et al., 2018a; Akoury et al., 2019; Ran et al., 2020), such a length predictor is unnecessary for us because the translation length is a by-product of Skip-AT, e.g., N_{nat}=$k \times N_{at}$, where N_{at} is the sequence length produced by Skip-AT [7]. There are two main advantages to avoiding the independent length predictor: (1) No need to carefully tune the weighting coefficient between the length prediction loss (sentence level) and the target token prediction loss (word level). (2) The length predicted by (Skip-)AT may be more accurate because it can access the already generated sequence information to obtain better translation performance (Ghazvininejad et al., 2019).

5.2 Training

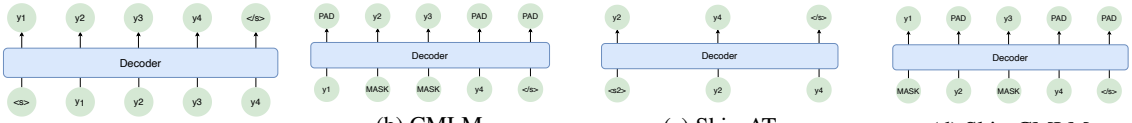
This section will introduce how to train the HRT model efficiently and effectively. Please refer to Appendix C for the entire training algorithm.

Training samples Figure 3 illustrates the differences of training samples among AT, CMLM, Skip-AT, and Skip-CMLM. Compared with AT, Skip-AT shrinks the sequence length from N to N/k. It should be noted that, although the sequence feeding to Skip-AT is shortened, the input position still follows the original sequence. For example, in Figure 3 (c), the position of Skip-AT input (<*s2*>, y_2, y_4) is (0, 2, 4) instead of (0, 1, 2). Moreover, CMLM has the opportunity to mask any token on the target sequence, while the masking pattern in Skip-CMLM is deterministic.

[5] `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`.

[6] Shaw et al. (2018) inject the relative positional representation in both `key` and `value`, while our SRPR only involves `key`. We found that this simplification has no negative impact on performance but saves memory footprint.

[7] More precisely, N_{nat} is the maximum length rather than the realistic length because multiple </*s*> may appear in the last k tokens.



(a) AT (b) CMLM (c) Skip-AT (d) Skip-CMLM

Figure 3: Examples of training samples fed to the decoder. For the sake of clarity, we omit the source sequence. <*s2*> is a special <*s*> for k=2. PAD is ignored when computing the loss function.

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

Table 1: Examples of different methods to generate the sequence of "a, b, c, d, e, f". "$\rightarrow$" denotes a new decode step conditioned on the prefix with beam search, while "$\dashrightarrow$" is its greedy search version.

Curriculum learning Unfortunately, the direct joint training of Skip-AT and Skip-CMLM is problematic because their training samples cannot fully use all the tokens in the sequence. 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 Skip-AT or Skip-CMLM. However, there is no such problem in AT and CMLM. Therefore, we propose to gradually transit from joint training {AT, CMLM} to {Skip-AT, Skip-CMLM} through curriculum learning (Bengio et al., 2009). In other words, the model is trained from chunk size 1 to k ($k > 1$). More concretely, given a batch of original sentence pairs $\mathcal{B}$ and let the proportion of chunk size k in $\mathcal{B}$ be p_k, we start with p_k=0 and construct the training samples of AT and CMLM for all pairs. Then we gradually increase p_k to introduce more learning signals for Skip-AT and Skip-CMLM until p_k=1. In the implementation, we schedule p_k by:

$$p_k = (t/T)^{\lambda} \tag{4}$$

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

Mixed distillation NAT models generally use the distillation data generated by AT models due to the smoother data distribution (Zhou et al., 2020). However, using only distillation data may lose some important information (e.g., rare words) contained in the original data (Ding et al., 2020). To combine the best of both worlds, we propose a simple but effective approach – *Mixed Distillation* (MixDistill). 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]. We empirically found that MixDistill makes the HRT model less prone to overfitting in some simple tasks (e.g., WMT'16 En→Ro). Compared with recent related studies (Ding et al., 2020, 2021), our method is easier to implement: HRT does not rely on external word alignment (Ding et al., 2020), and also avoids the time-consuming bidirectional distillation process (Ding et al., 2021).

5.3 Inference

Thanks to the joint training under chunk size one and k, it is flexible for HRT to trade-off translation quality and speedup by switching different decoding chunk size $C_d \in [1, k]$.

Autoregressive decoding When C_d=1, HRT behaves like the standard AT model: Feed <*s*> to Skip-AT decoder and increase the target position by one in each step. Skip-CMLM decoder is needless. In this way, HRT has no faster speed advantage than AT, but we can regard C_d=1 as the performance upper bound of C_d=k.

Hybrid-regressive decoding When C_d=k, the Skip-AT decoder firstly starts from <*sk*> to autoregressively generate a discontinuous target sequence $\hat{\boldsymbol{y}}_{at} = (z_1, z_2, \ldots, z_m)$ with chunk size k until meeting <*/s*>. Then we construct the input of Skip-CMLM decoder $\boldsymbol{y}_{nat}$ by appending $k-1$ <*mask*> before every z_i. The final translation is generated by replacing all <*mask*> with the predicted tokens by Skip-CMLM decoder with one iteration. If there are multiple <*/s*> existing, we truncate to the first <*/s*>. 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 st. $\mathrm{b}_{at} \geq \mathrm{b}_{nat}$: We

[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
AT	Transformer	N	34.25	34.40	27.45	31.86
	Transformer-20L	N	-	-	28.79	33.02
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 and Kong, 2021)	1	33.79	34.16	27.49	31.39
Iterative NAT	iNAT (Lee et al., 2018)	Adaptive	29.66	30.30	21.54	25.43
	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
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(k=2) (Ran et al., 2020)	$N/2$	32.92	33.19	27.11	31.67
Our	HRT (b_{at}=5, b_{nat}=1)	$N/2+1$	34.36	34.55	27.98	31.93
	HRT (b_{at}=5, b_{nat}=5)	$N/2+1$	**34.53**	**34.80**	28.10	32.07
	HRT-20L (b_{at}=5, b_{nat}=1)	$N/2+1$	-	-	28.90	33.06
	HRT-20L (b_{at}=5, b_{nat}=5)	$N/2+1$	-	-	**28.99**	**33.08**

Table 2: The BLEU scores of our proposed HRT and the baseline methods on four WMT tasks. Unless otherwise stated, the used beam size is 5. "Adaptive" denotes dynamic iterations. "20L" stands for using a 20-layer encoder. All HRT models only iterate once by non-autoregression. * means `sacrebleu` score, which is uncomparable to others. All the HRT results are significantly better ($p<0.01$) than the autoregressive counterparts, measured by paired bootstrap resampling (Koehn, 2004) .

only feed the Skip-CMLM with the top b_{nat} Skip-AT hypothesis. Finally, we choose the translation hypothesis with the highest score $S(\hat{y})$ by:

$$\underbrace{\sum_{i=1}^{m} \log P(z_i|\boldsymbol{x}, z_{<i})}_{\text{Skip-AT score}} + \underbrace{\sum_{i=0}^{m-1}\sum_{j=1}^{k-1} \log P(\hat{y}_{i\times k+j}|\boldsymbol{x}, \boldsymbol{y}_{nat})}_{\text{Skip-CMLM score}} \quad (5)$$

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

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 from these methods lies in the content of AT output, such as latent variable (Kaiser et al., 2018), reordered source token (Ran et al., 2019), syntactic label (Akoury et al., 2019). In contrast, our approach uses the deterministic target token as Ghazvininejad et al. (2019). Another line to incorporate AT and NAT is to couple the two decoding paradigms. For example, SAT (Wang et al., 2018a) embeds chunk-level NAT into the AT process, while RecoverSAT (Ran et al., 2020) does the opposite. As shown in Table 1, although HRT has a longer decoding path, the cost of the non-autoregressive process in HRT is cheap. The reason is that our Skip-CMLM can work well with greedy search, thanks to the good context provided by the relatively slow Skip-AT (see Table 2). In contrast, SAT and RecoverSAT need larger beams to explore translations of different lengths. Another note is that HRT significantly outperforms both SAT and RecoverSAT, e.g., +1.0 BLEU scores on WMT En→De (see Table 2).

6 Experiments

Setup We mainly conducted experiments on four widely used WMT tasks: WMT'16 English↔Romanian (En↔Ro, 610k) and WMT'14 English↔German (En↔De, 4.5M). We replicated the same data processing as Ghazvininejad et al. (2019) for fair comparisons. 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). Since Ghazvininejad et al. (2019) did not release the distillation data of En↔De, we retrained the AT teacher models to produce the corresponding data. Specifically, we use the deep PreNorm Transformer-Base with a 20-layer encoder as the

Model	MT04	MT05	MT08
AT	43.86	52.91	33.94
CMLM(I=10)	42.47	52.16	33.09
HRT5-1	44.28	53.44	34.63
HRT5-5	**44.31**	**53.77**	**34.74**

Table 3: BLEU scores on the NIST Zh→En task.

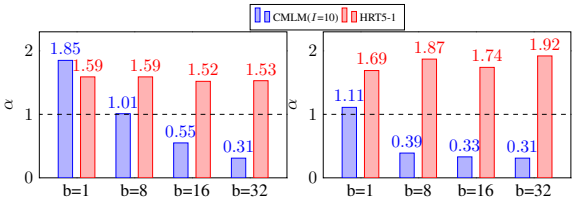


Figure 4: Relative speedup ratio w.r.t. batch size (b) and computing device (left: GPU, right: CPU) on En→De task. The dashed line at y=1 represents the corresponding autoregressive model. HRT{#1}−{#2} denotes decoding with b_{at}=#1 and b_{nat}=#2.

Lang.	C_d	Raw	Dist.	Mix Dist.
En→Ro	N/A	34.25	-	-
	k	33.92	33.41	**34.53**
	1	**34.29**	33.41	34.27
En→De	N/A	27.45	-	-
	k	26.37	28.00	**28.10**
	1	27.60	28.42	**28.51**

Table 4: The BLEU scores against different data strategies . C_d="N/A" represents the original AT model.

teacher instead of Transformer-Big for faster train and inference with comparable performance (Wang et al., 2019a). We ran all experiments on 8 TITAN X (Pascal) GPUs. Unless noted otherwise, we use the chunk size k=2. We set p_{raw}=0.5 for En↔Ro and p_{raw}=0.8 for En↔De according to validation sets. The windows size of SRPR is 16 as Shaw et al. (2018) [9]. We fine-tune HRT models on pre-trained AT models and take the same training steps as that of AT (about 1/3 training steps compared to previous NAT work) [10]. Other training hyperparameters are the same as Vaswani et al. (2017) or Wang et al. (2019a) (deep-encoder).

Translation quality Table 2 reports the BLEU scores on four WMT tasks. We first verify that greedy search (b_{nat}=1) is sufficient for our Skip-CMLM instead of cost-intensive beam search (e.g., a drop of approximately 0.1 BLEU on En↔De). In contrast, we noticed that previous methods significantly degrade performance when using greedy search. For example, when the beam size switches from 4 to 1, the BLEU score in SAT is reduced by 0.81. Therefore, we use b_{at}=5 and b_{nat}=1 (denoted by HRT5-1) in the following experiments unless noted otherwise. Our HRT outperforms most existing NAT, IR-NAT, and Semi-NAT models and establishes new state-of-the-art results on En↔Ro. Besides, in line with Guo et al. (2020), when using a deeper encoder, HRT-20 can further improve approximately +0.8 BLEU on more challenging En↔De tasks. More surprisingly, we found that HRT can be slightly better than the AT models trained from scratch. We attribute it to two reasons: (1) HRT is fine-tuned on a well-trained AT model, making training easier; (2) Mixing up AT and NAT has a better regularization effect than training alone. We also report the experimental results on the Zh→En task in Table 3. We can see that HRT is once again superior to the original AT model and CMLM model, which indicates that the effectiveness of HRT is agnostic to language pairs. Please see Appendix E for case study.

7 Analysis

Translation speed Previous work generally only reports the decoding speed on GPUs with a batch size of 1. Instead, we systematically tested the decoding speed under varying batch sizes and devices on the WMT'14 En→De test set (see Figure 4). By default, all systems use a beam size of 5 (except the Skip-CMLM process in HRT). It can be seen that although HRT is slower than CMLM(I=10) when running on a GPU with a batch size of 1, CMLM(I=10) dramatically slows down as the batch size increases. In contrast, HRT5-1 is consistently more than 50% faster than AT without varying with the environment [11]. It indicates that HRT has a more stable acceleration than IR-NAT.

Impact of data strategy In Table 4, we compared different data strategies, including raw data (`Raw`), sequence-level knowledge distillation (`Dist.`), and mixed distillation (`Mix Dist.`). Overall, Mix Dist. is superior to other methods across the board, indicating that training with raw and distillation data is complementary. In addition, we also found that the performance of the distillation data is lower than that of the raw data on

[9] For autoregressive baselines, adding SRPR in Transformer decoders did not bring obvious improvements.

[10] Since HRT needs to train Skip-AT and Skip-CMLM jointly, the wall-clock time is about two times longer than AT in the same training epochs.

[11] The acceleration results of other HRT variants are reported in Appendix D.

Chunk (k)	BLEU C_d=k	BLEU C_d=1	Latency (sec.) GPU	Latency (sec.) CPU
2	**34.11**	33.86	20.0	70.5
3	31.15	33.78	13.0	54.6
4	28.22	**34.12**	**12.2**	**53.9**

Table 5: The effects of chunk sizes. Latency is measured in batch size of 16, C_d=k and b_{at}=b_{nat}=1.

Model	KD	BLEU	SU-GPU	SU-CPU
AT	w/o	27.45	ref.	ref.
CMLM(I=10)	w/	27.03*	0.93	0.54
HRT	w/	27.90	1.56	1.80
SD-AT	w/	**28.23**	2.08	3.47
SD-HRT	w/	28.05	**3.23**	**4.55**

Table 6: Effects of deep-shallow architecture on En→De. SU-GPU and SU-CPU denotes the average speedup ratio over batch size {1,8,16,32} on GPU and CPU, respectively. b_{at}=5, b_{nat}=1. * denotes the number comes from the original paper.

En→Ro task, which is against the previous results. As interpreted by Zhou et al. (2020), we suspect that when the translation model is strong enough, training entirely through distilled data may make learning too easy and lead to overfitting. Another note is that, even if we only use `Raw` or `Dist.` alone, our HRT can obtain equivalent or even better performance than the original AT model.

Impact of chunk size We tested chunk size k on the En→Ro test set as shown in Table 5, where we can see that: (1) A large k has more significant acceleration on the GPU because fewer autoregressive steps are required; (2) As k increases, the performance of hybrid-regressive decoding drops sharply (e.g., k=4 is 6 BLEU points lower than k=2.), but k has little effect on the autoregressive mode. It indicates that the training difficulty of Skip-AT increases as k gets bigger. We think that skip-generation may require more fancy training algorithms, which is left for our future work.

Deep-Shallow architecture Kasai et al. (2020b) point out that AT with deep-shallow architecture (i.e., deep encoder and shallow decoder) can be substantially sped up without loss in accuracy. We also compare HRT and AT under this setting: 12-layer encoder and 1-layer decoder, denoted by SD-HRT and SD-AT, respectively. Instead of the proposed mixed distillation, we use the same sequence-level KD as AT for a fair comparison. From the results listed in Table 6, we can see that: (1) SD-AT outperforms our HRT in both BLEU and speed, indicating the effectiveness of the deep-shallow layer allocation; (2) HRT also benefits from the deep-shallow architecture, achieving comparable BLEU and faster decoding with SD-AT. Note that CMLM with deep-shallow architecture degrades severely (1.1 BLEU dropped) as reported by Kasai et al. (2020b), which indicates that SD-HRT can inherit the good character from SD-AT.

System	BLEU	Δ
WMT'16 En→Ro validation set		
AT	35.12	N/A
HRT5-1	34.84	ref.
−FT	34.46	-0.38
−SRPR	34.62	-0.22
−MixDistill	34.21	-0.63
−CL(p_k=1.0)	33.85	-0.99
−ALL	33.38	-1.46
WMT'14 En→De validation set		
HRT5-1 (100k)	26.68	N/A
Official CMLM (300k)	25.51	ref.
+FT + SRPR + MD (100k)	25.64	+0.13
+FT + SRPR + MD (300k)	26.13	+0.62

Table 7: Ablation study on En→Ro and En→De.

Ablation study In Table 7, we first conducted ablation studies on each newly introduced technique. We can see that all techniques help to improve performance, but the most critical components are CL (-0.99) and MixDistill (-0.63). We also tried to exclude all of them from the standard HRT (-ALL), resulting in a total reduction of 1.46 BLEU points. We continued to experiment on En→De task to verify whether the optimization methods used in HRT training improve CMLM. Table 7 shows that with the help of FT+SRPR+MD, our CMLM model has improved +0.62 BLEU points compared with the official CMLM when fine-tuning 300k steps. However, there is still a large BLEU gap (0.55 BLEU points) between the enhanced CMLM and our HRT with fewer training steps.

8 Conclusion

We have pointed out that existing IR-NAT methods cannot efficiently accelerate when running with a large batch or on CPU. Inspired by synthetic experiments, we proposed a two-stage translation paradigm, HRT, to combine the advantages of AT and NAT. Experimental results show that our approach outperforms the existing Semi-AT methods and is promising to be a good substitute for AT due to competitive performance and stable 1.5x acceleration.

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.

Aishwarya Bhandare, Vamsi Sripathi, Deepthi Karkada, Vivek Menon, Sun Choi, Kushal Datta, and Vikram Saletore. 2019. Efficient 8-bit quantization of transformer neural machine language translation model. *arXiv preprint arXiv:1906.00532*.

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.

Liang Ding, Xuebo Liu Longyue Wang, Derek F. Wong, Dacheng Tao, and Zhaopeng Tu. 2021. 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*.

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

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 6111–6120, 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 OK Li, and Richard Socher. 2017. Non-autoregressive neural machine translation. *arXiv preprint arXiv:1711.02281*.

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

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.

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.

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.

Bei Li, Ziyang Wang, Hui Liu, Quan Du, Tong Xiao, Chunliang Zhang, and Jingbo Zhu. 2021. Learning light-weight translation models from deep transformer. In *AAAI*.

Zhuohan Li, Zi Lin, Di He, Fei Tian, QIN Tao, WANG Liwei, and Tie-Yan Liu. 2019. Hint-based training for non-autoregressive machine translation. 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 5712–5717.

Ye Lin, Yanyang Li, Tengbo Liu, Tong Xiao, Tongran Liu, and Jingbo Zhu. 2020. Towards fully 8-bit integer inference for the transformer model. *ArXiv*, abs/2009.08034.

Ye Lin, Yanyang Li, Ziyang Wang, Bei Li, Quan Du, Tong Xiao, and Jingbo Zhu. 2021. Weight distillation: Transferring the knowledge in neural network parameters. 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 2076–2088, Online. Association for Computational Linguistics.

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.

Chenze Shao, Yang Feng, Jinchao Zhang, Fandong Meng, Xilin Chen, and Jie Zhou. 2019. Retrieving sequential information for non-autoregressive neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3013–3024.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468, New Orleans, Louisiana. Association for Computational Linguistics.

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*.

Zhiqing Sun, Zhuohan Li, Haoqing Wang, Di He, Zi Lin, and Zhihong Deng. 2019. Fast structured decoding for sequence models. In *Advances in Neural Information Processing Systems*, pages 3011–3020.

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. 2019a. 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.

Yiren Wang, Fei Tian, Di He, Tao Qin, ChengXiang Zhai, and Tie-Yan Liu. 2019b. Non-autoregressive machine translation with auxiliary regularization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5377–5384.

Tong Xiao, Yinqiao Li, Jingbo Zhu, Zhengtao Yu, and Tongran Liu. 2019. Sharing attention weights for fast transformer. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pages 5292–5298.

Biao Zhang, Deyi Xiong, and Jinsong Su. 2018. Accelerating neural transformer via an average attention network. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1789–1798.

Chunting Zhou, Jiatao Gu, and Graham Neubig. 2020. Understanding knowledge distillation in non-autoregressive machine translation. In *ICLR 2020 : Eighth International Conference on Learning Representations*.

A Analysis of speed degradation problem

Model	BH=1	BH=8	BH=16	BH=32
AT	962s	151s	78s	43s
CMLM (I=10)	464s	125s	119s	129s

Table 8: Elapsed time of decoding *newstest2014* by different batch sizes on Titian X GPU. The beam size is 5.

As shown in Table 8, we list the elapsed time when AT and CMLM(I=10) decode the test set of WMT'14 En→De task under different batch sizes. The beam size used is 5. We test the time on Titian X GPU and report the average of 3 runs. In general, we can see that AT benefits more than CMLM(I=10) from the increase in batch size. Specifically, when the batch size is increased from 1 to 32, the delay of AT is reduced by 962/43 (about 22.4) times, while CMLM (I=10) is only reduced by 464/129 (about 3.6) times. That is, when the burden of parallel computing is too heavy, the common belief that non-autoregressive runs faster than autoregressive may not hold.

B AT Transformer in synthetic experiments

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 9: The performance of autoregressive models in the synthetic experiment.

In the synthetic experiment, we trained all AT models with the standard Transformer-Base configuration: layer=6, dim=512, ffn=2048, head=8.

Algorithm 1 Training Algorithm for Hybrid-Regressive Translation

Input: Training data D including distillation targets, pretrained AT model M_{at}, chunk size k, mixed distillation rate p_{raw}, schedule coefficient λ

Output: Hybrid-Regressive Translation model 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: $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, $\boldsymbol{Y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n\}$, $\boldsymbol{Y}' = \{\boldsymbol{y}'_1, \ldots, \boldsymbol{y}'_n\} \leftarrow$ fetch a batch from D
4: **for** i in $1, 2, \ldots, n$ **do**
5: $\boldsymbol{B}_i = (\boldsymbol{X}_i, \boldsymbol{Y}_i^*) \leftarrow$ sampling $\boldsymbol{Y}_i^* \sim \{\boldsymbol{Y}_i, \boldsymbol{Y}'_i\}$ with $P(\boldsymbol{Y}_i) = p_{raw}$ ▷ mixed distillation
6: **end for**
7: $p_k \leftarrow (\frac{t}{T})^{\lambda}$ ▷ curriculum learning
8: $\boldsymbol{B}_{c=k}, \boldsymbol{B}_{c=1} \leftarrow \boldsymbol{B}_{:\lfloor n \times p_k \rfloor}, \boldsymbol{B}_{\lfloor n \times p_k \rfloor:}$ ▷ split batch
9: $\boldsymbol{B}^{at}_{c=k}, \boldsymbol{B}^{nat}_{c=k} \leftarrow$ construct {Skip-AT, Skip-CMLM} training samples based on $\boldsymbol{B}_{c=k}$
10: $\boldsymbol{B}^{at}_{c=1}, \boldsymbol{B}^{nat}_{c=1} \leftarrow$ construct {AT, CMLM} training samples based on $\boldsymbol{B}_{c=1}$
11: Optimize M_{hrt} using $\boldsymbol{B}^{at}_{c=k} \cup \boldsymbol{B}^{at}_{c=1} \cup \boldsymbol{B}^{nat}_{c=k} \cup \boldsymbol{B}^{nat}_{c=1}$ ▷ joint training
12: **end for**

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 9, 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: $\boldsymbol{B}_{c=1}$ and $\boldsymbol{B}_{c=k}$, where $|\boldsymbol{B}_{c=k}|/|\boldsymbol{B}| = p_k$ (Line 7-8). Next, we construct four kinds of training samples based on corresponding batches: $\boldsymbol{B}^{at}_{c=k}$, $\boldsymbol{B}^{at}_{c=1}$, $\boldsymbol{B}^{nat}_{c=k}$ and $\boldsymbol{B}^{nat}_{c=1}$. 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.

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(I=10) (5th iteration)	[Problem@@] [atisch] [:] [Die] [zivile] [militärische] Gerichts@@ barkeit wird weiterhin [aufrechterhalten] . *</s>*
HRT5-1 (C_d=1)	Auch problematisch : Die zivile Militär@@ geri@@ chts@@ barkeit wird weiterhin aufrechterhalten . *</s>*
HRT5-1 (C_d=2)	[Auch] problematisch [:] Die [zivile] militärische [Rechtsprechung] wird [weiterhin] aufrechterhalten [.] *</s>*

Table 10: A case study in En→De validation set. "[]" denotes the original token is *<mask>*. We also report the CMLM(I=10) in the 5th iteration as its masking rate is closing to that of HRT5-1(C_d=2), e.g., 50%.

D Acceleration effects of HRT variants

Model	BH=1	BH=8	BH=16	BH=32
On GPU				
HRT5-1	1.59	1.59	1.52	1.53
HRT5-5	1.40	1.38	1.34	1.29
HRT-20L(5-1)	1.55	1.52	1.53	1.49
On CPU				
HRT5-1	1.69	1.87	1.74	1.92
HRT5-5	1.43	1.47	1.41	1.40
HRT-20L(5-1)	1.62	1.75	1.77	1.81

Table 11: Speedup effects of different HRT variants.

In Table 11, we list the speedup results of HRT5-5 and HRT-20L(5-1) by the average of three runs. We can see that HRT5-5 can maintain a stable acceleration (about 30%) than AT counterparts, which is less efficient than HRT5-1. Please note that HRT5-5 is only slightly better than HRT5-1 (about 0.1 BLEU). Besides, the overall result of the 20-layer encoder is similar to that of a 6-layer encoder because the translation time is mainly consumed in the decoder.

E Case study

Table 10 shows a translation example from En→De validation set. Compared CMLM and HRT(C_d=2), although both have the same masking rate (50%), the masked positions in CMLM are more continuous than HRT. It leads to a suboptimal translation, which is consistent with the observation in the synthetic experiment. Besides, we can see that most skipped predictions (C_d=2) are the same as autoregressive ones (C_d=1). It indicates that our model is capable of generating appropriate discontinuous sequences.