Fast Interleaved Bidirectional Sequence Generation

Biao Zhang[1] **Ivan Titov**[1,2] **Rico Sennrich**[3,1]

[1]School of Informatics, University of Edinburgh
[2]ILLC, University of Amsterdam
[3]Department of Computational Linguistics, University of Zurich
B.Zhang@ed.ac.uk, ititov@inf.ed.ac.uk, sennrich@cl.uzh.ch

Abstract

Independence assumptions during sequence generation can speed up inference, but parallel generation of highly inter-dependent tokens comes at a cost in quality. Instead of assuming independence between neighbouring tokens (semi-autoregressive decoding, SA), we take inspiration from bidirectional sequence generation and introduce a decoder that generates target words from the left-to-right and right-to-left directions simultaneously. We show that we can easily convert a standard architecture for unidirectional decoding into a bidirectional decoder by simply interleaving the two directions and adapting the word positions and self-attention masks. Our interleaved bidirectional decoder (IBDecoder) retains the model simplicity and training efficiency of the standard Transformer, and on five machine translation tasks and two document summarization tasks, achieves a decoding speedup of $\sim 2\times$ compared to autoregressive decoding with comparable quality. Notably, it outperforms left-to-right SA because the independence assumptions in IBDecoder are more felicitous. To achieve even higher speedups, we explore hybrid models where we either simultaneously predict multiple neighbouring tokens per direction, or perform multi-directional decoding by partitioning the target sequence. These methods achieve speedups to $4\times$–$11\times$ across different tasks at the cost of <1 BLEU or <0.5 ROUGE (on average).[1]

1 Introduction

Neural sequence generation aided by encoder-decoder models (Bahdanau et al., 2015; Vaswani et al., 2017) has achieved great success in recent years (Bojar et al., 2018; Song et al., 2019; Raffel et al., 2019; Karita et al., 2019), but still suffers from slow inference. One crucial bottleneck lies in its generative paradigm which factorizes the conditional probability along the target sequence $\mathbf{y} = \{y_1, y_2, \ldots, y_n\}$ of length n as follows:

$$p(\mathbf{y}|\mathbf{x}) = \prod_{t=1}^{n} p\left(y_t|\mathbf{y}_{<t}, \mathbf{x}\right), \qquad (1)$$

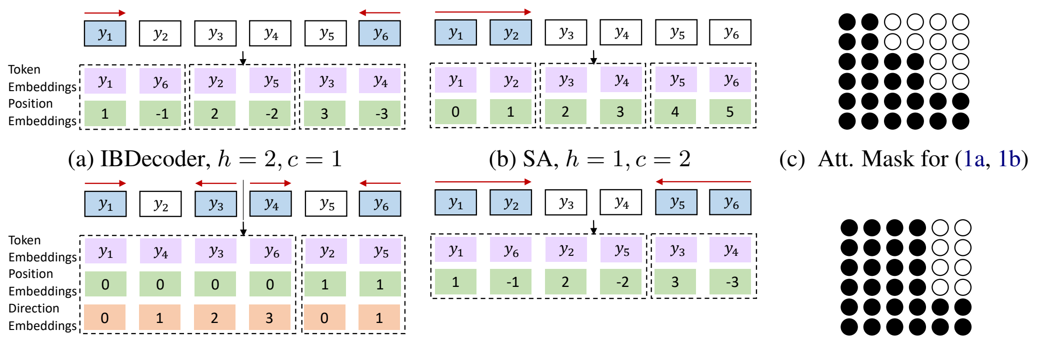
where $\mathbf{x} = \{x_1, x_2, \ldots, x_m\}$ is the source sequence of length m. This factorization determines that target words can only be generated one-by-one in a sequential and unidirectional manner, which limits the decoding efficiency.

A promising direction to break this barrier is to generate multiple target words at one decoding step to improve the parallelization of inference (Gu et al., 2018; Stern et al., 2018). However, this introduces independence assumptions that hurt translation quality, since words produced in parallel are in fact likely to be inter-dependent. We hypothesize that there are groups of words that are less likely to be strongly inter-dependent than neighbouring words, which will allow for better parallelization. Inspired by bidirectional modeling (Zhang et al., 2019b, 2020), we resort to an alternative probabilistic factorization:

$$p^{\text{BD}}(\mathbf{y}|\mathbf{x}) = \prod_{t=1}^{\lceil n/2 \rceil} p\left(\overrightarrow{y_t}, \overleftarrow{y_{t'}}|\overrightarrow{\mathbf{y}_{<t}}, \overleftarrow{\mathbf{y}_{>t'}}, \mathbf{x}\right), \qquad (2)$$

Introducing an independence assumption between t and $t' = n - t + 1$ allows for parallel word prediction from both the $\overrightarrow{\text{left-to-right}}$ and $\overleftarrow{\text{right-to-left}}$ directions. Based on this factorization, Zhou et al. (2019) propose synchronous bidirectional translation using a dedicated interactive decoder, and report quality improvements compared to left-to-right semi-autoregressive decoding (Wang et al., 2018, SA) in translation quality. However, their success comes along with extra computational overhead brought by the specialized decoder. Empirically, Zhou et al. (2019) only report a decoding

[1]Source code is released at https://github.com/bzhangGo/zero.


Token Embeddings
Position Embeddings
Direction Embeddings
y_1 y_2 y_3 y_4 y_5 y_6
y_1 y_6 y_2 y_5 y_3 y_4
1 -1 2 -2 3 -3
0 1 2 3 4 5
y_1 y_4 y_3 y_6 y_2 y_5
0 0 0 0 1 1
0 1 2 3 0 1

(a) IBDecoder, $h = 2, c = 1$ (b) SA, $h = 1, c = 2$ (c) Att. Mask for (1a, 1b)

(d) IMDecoder, $h = 4, c = 1$ (e) IBDecoder + SA, $h = 2, c = 2$ (f) Att. Mask for (1d, 1e)

Figure 1: Overview of the interleaved bidirectional decoder (IBDecoder, 1a), the semi-autoregressive decoder (SA, 1b), the interleaved multi-directional decoder (IMDecoder, 1d) and the bidirectional semi-autoregressive decoder (IBDecoder+ SA, 1e) on target sequence $\mathbf{y} = \{y_1, y_2, \ldots, y_6\}$. We reorganize the target sequence (purple), the word positions (green) and the self-attention mask (circles) to reuse the standard Transformer decoder. During inference, multiple target words are generated simultaneously at each step (dashed rectangles), improving the decoding speed. The self-attention masks are given in (1c) and (1f), where sold black circles indicate allowed attention positions. Red arrows indicate generation directions (h is the direction number), whose length denotes the number of words produced per direction (c). Blue rectangles denote words generated at the first step. The direction embedding (red rectangles) reflects the direction each target word belongs to. Apart from the left-to-right generation, IBDecoder jointly models the right-to-left counterpart within a single sequence. IMDecoder extends IBDecoder by splitting the sequence into several equal segments and performing bidirectional generation on each of them, while IBDecoder+SA allows each direction to produce multiple words.

speedup of 1.38×, slower than SA, although the factorization halves the decoding steps.

We combine the strengths of bidirectional modeling and SA, and propose interleaved bidirectional decoder (IBDecoder) for fast generation. As shown in Figure 1a, we interleave target words from the left-to-right and right-to-left directions and separate their positions to support reusing any standard unidirectional decoders, such as the Transformer decoder (Vaswani et al., 2017). We reorganize the self-attention mask to enable inter- and intra-direction interaction (Figure 1c) following SA. Unlike SA, we show through experiments that distant tokens from different directions are less inter-dependent, providing a guarantee for better performance. Compared to previous studies (Zhang et al., 2018d, 2019b, 2020; Zhou et al., 2019), our approach has no extra model parameters and brings in little overhead at training and decoding.

IBDecoder is speedup-bounded at 2×. To push this ceiling up, we explore strategies for multi-word simultaneous generation, including *multi-directional decoding* (IMDecoder, Figure 1d) and *SA* (Figure 1b). The former extends Eq. 2 by inserting more generation directions, while the latter allows each direction to produce multiple target words (Wang et al., 2018). These strategies offer us a chance to aggressively improve the decoding speed albeit at the risk of degenerated performance. To encourage multi-word generation in parallel, we propose a modified beam search algorithm.

We extensively experiment on five machine translation tasks and two document summarization tasks, with an in-depth analysis studying the impact of batch size, beam size and sequence length on the decoding speed. We close our analysis by examining the capacity of our model in handling long-range dependencies. On these tasks, IBDecoder yields ∼2× speedup against Transformer at inference, and reaches 4×–11× after pairing it with SA. Still, the overall generation quality is comparable. When we pair our method with sequence-level knowledge distillation (Kim and Rush, 2016), we outperform a Transformer baseline on 6 out of 7 tasks.

Our contributions are summarized below:

- We propose IBDecoder, following a bidirectional factorization of the conditional probability, for fast sequence generation. IBDecoder retains the training efficiency and is easy to implement.
- We extend IBDecoder to enable multi-word simultaneous generation by investigating integration with IMDecoder and SA. Results show that IBDecoder + SA performs better than IMDecoder.
- We propose a modified beam search algorithm to support step-wise parallel generation.
- On several sequence generation benchmarks, IBDecoder yields ∼2× speedup against Transformer at inference, and reaches 4×–11× af-

ter pairing it with SA. Still, the overall generation quality is comparable.

2 Related Work

Efforts on fast sequence generation come along with the rapid development of encoder-decoder models (Vaswani et al., 2017). A straightforward way is to reduce the amount of computation. Methods in this category range from teacher-student model (Kim and Rush, 2016; Hayashi et al., 2019), constrained softmax prediction (Hu et al., 2015), beam search cube pruning (Zhang et al., 2018c), float-point quantization (Wu et al., 2016; Bhandare et al., 2019), model pruning (See et al., 2016), to simplified decoder architectures, such as lightweight recurrent models (Zhang et al., 2018b; Zhang and Sennrich, 2019; Kim et al., 2019), average attention network (Zhang et al., 2018a), merged attention network (Zhang et al., 2019a), dynamic convolution (Wu et al., 2019), and hybrid attentions (Shazeer, 2019; Wang et al., 2019), .etc.

Nonetheless, the above methods still suffer from the inference bottleneck caused by the sequential nature of autoregressive models. Instead, Gu et al. (2018) propose non-autoregressive generation where target words are predicted independently, leading to great speedup, albeit at a high cost to generation quality. Follow-up studies often seek solutions to recover the performance (Libovický and Helcl, 2018; Guo et al., 2019; Shao et al., 2020; Ghazvininejad et al., 2020; Ran et al., 2020), but also reveal the trade-off between the quality and speed in terms of autoregressiveness. This motivates researchers to discover the optimal balance by resorting to semi-autoregressive modeling (Wang et al., 2018; Stern et al., 2018), iterative refinement (Lee et al., 2018; Stern et al., 2019; Ghazvininejad et al., 2019) or in-between (Kaiser et al., 2018; Akoury et al., 2019).

We hypothesize that generation order affects the felicity of independence assumptions made in semi-autoregressive modelling. Unlike generation with flexible orders (Emelianenko et al., 2019; Stern et al., 2019; Gu et al., 2019a), we employ deterministic generation order for model simplicity and training efficiency, specifically focusing on bidirectional decoding. The study of bidirectional modeling dates back to the era of phase-based statistical machine translation (Watanabe and Sumita, 2002; Finch and Sumita, 2009) and recently gained popularity in neural machine translation (Liu et al., 2016; Sennrich et al., 2016a; Zhang et al., 2019c,b; Zheng et al., 2019). Unfortunately, these methods either design complex neural decoders, which hurts training efficiency, and/or perform the left-to-right and right-to-left inference separately followed by rescoring, which slows down decoding. By contrast, our model speeds up inference while maintaining training speed.

Our work is closely related to SA (Wang et al., 2018) and synchronous bidirectional generation (Zhou et al., 2019). IBDecoder extends SA to incorporate information from different directions. In contrast to Zhou et al. (2019), we only make minimal changes to the standard Transformer decoder, which benefits efficiency during training and inference, and makes our method easy to implement. We also find improvements in both decoding speed and translation quality compared to (Wang et al., 2018; Zhou et al., 2019).

3 Autoregressive Transformer

Transformer (Vaswani et al., 2017), the state-of-the-art neural sequence generation model, follows the autoregressive factorization as in Eq. 1. To handle the dependency of target word y_t on previous target words $\mathbf{y}_{<t}$, Transformer relies on a masked self-attention network in the decoder:

$$\textsc{Att}(\mathbf{Y}^l, \mathbf{M}) = f\left(\frac{\mathbf{Q}^l \mathbf{K}^{l^T}}{\sqrt{d}} + \mathbf{M}\right)\mathbf{V}^l \tag{3}$$

where $\mathbf{Q}^l, \mathbf{K}^l, \mathbf{V}^l = \mathbf{W}_q^l \mathbf{Y}^l, \mathbf{W}_k^l \mathbf{Y}^l, \mathbf{W}_v^l \mathbf{Y}^l \in \mathbb{R}^{n \times d}$, $f(\cdot)$ denotes softmax operation, d is model dimension and l is layer depth. $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v \in \mathbb{R}^{d \times d}$ are trainable parameters.

The mask matrix $\mathbf{M} \in \mathbb{R}^{n \times n}$ limits the access of attention to only the past target words. Formally, given the target sequence length n, this matrix can be constructed by the following masking function:

$$\mathcal{M}_{i,j}(h, c) = \begin{cases} 0, & \text{if } \lceil i/(h \cdot c) \rceil \geq \lceil j/(h \cdot c) \rceil \\ -\infty, & \text{otherwise} \end{cases} . \tag{4}$$

where $0 < i, j < n$, h denotes the number of generation directions, and c is the number of target words predicted per direction. By default, the Transformer decoder is unidirectional and generates words one-by-one. Thus, $\mathbf{M} = \mathcal{M}(1, 1)$. The infinity here forces softmax output a probability of 0, disabling invalid attentions.

The input layer to Transformer's decoder is the addition of target word embedding $\mathbf{E}_\mathbf{y}$ and word position encoding $\text{PE}_{\mathcal{T}}$, i.e $\mathbf{Y}^0 = \mathbf{E}_{\mathbf{y}} + \text{PE}_{\mathcal{T}} \in \mathbb{R}^{n \times d}$. $\mathcal{T}$ maps $\mathbf{y}$ to its word position sequence, which is a simple indexing function (Figure 1b):

$$\mathcal{T}_t = t - 1, \tag{5}$$

where $t = 1 \dots n$. Transformer adopts the sinusoidal positional encoding to project these indexes to real-space embeddings, and uses the last-layer decoder output $\mathbf{Y}^L$ to predict the respective next target word. We explain how to accelerate generation by reordering $\mathbf{y}$, adjusting h, c and $\mathcal{T}$ next.

4 Interleaved Bidirectional Decoder

The structure of Transformer is highly parallelizable, but the autoregressive schema $(h = 1, c = 1)$ blocks this parallelization during inference. We alleviate this barrier by exploring the alternative probabilistic factorization in Eq. 2 to allow words predicted from different directions simultaneously.

We propose IBDecoder as shown in Figure 1a. We reuse the standard decoder's architecture in a bid to largely inherit Transformer's parallelization and avoid extra computation or parameters, rather than devising dedicated decoder architectures (Zhou et al., 2019; Zhang et al., 2020). To make the left-to-right and right-to-left generation collaborative, we reorganize the target sequence and the word positions below (purple and green rectangles in Figure 1a):

$$\mathbf{y}^{\text{BD}} = \left[y_1 y_n, y_2 y_{n-1}, \dots, y_{\lfloor n/2 \rfloor + 1}\right], \tag{6}$$
$$\mathcal{T}_t^{\text{BD}} = (-1)^{(t-1)} \lceil t/2 \rceil. \tag{7}$$

By following the generation order defined by Eq. 2, the sequence $\mathbf{y}^{\text{BD}}$ interleaves $\mathbf{y}_{1:\lfloor n/2 \rfloor}$ and $\mathbf{y}_{\lfloor n/2 \rfloor + 1:n}$ and converts a bidirectional generation problem to a unidirectional one. We introduce negative positions to $\mathcal{T}^{\text{BD}}$ to retain the locality bias of sinusoidal positional encodings in $\mathbf{y}^{\text{BD}}$.[2] Compared to $(\mathbf{y}, \mathcal{T})$, the reorganized sequences $(\mathbf{y}^{\text{BD}}, \mathcal{T}^{\text{BD}})$ have the same length, thus with no extra overhead.

We also adapt the self-attention mask to permit step-wise bidirectional generation:

$$\mathbf{M}^{\text{BD}} = \mathcal{M}(2, 1), \tag{8}$$

where IBDecoder has $h = 2$ generation directions. This corresponds to the relaxed causal mask by Wang et al. (2018), which ensures access to all predictions made in previous time steps[3] and allows for interactions among the tokens to be produced per time step. Although two words are predicted independently at each step, the adapted self-attention mask makes their corresponding decoding context complete; each word has full access to its corresponding decoding history, i.e. the left-to-right $(\mathbf{y}_{1:t})$ and right-to-left $(\mathbf{y}_{n-t+1:n})$ context. Except for $(\mathbf{y}^{\text{BD}}, \mathbf{M}^{\text{BD}}, \mathcal{T}^{\text{BD}})$, other components in Transformer are kept intact, including training objective.

4.1 Beyond Two-Word Generation

Eq. 2 only supports two-word generation, which indicates an upper bound of $2\times$ speedup at inference. To improve this bound, we study strategies for multi-word generation. We explore two of them.

Multi-Directional Decoding Similar to IBDecoder, IMDecoder also permutes the target sequence. It inserts multiple generation directions (i.e. increases h), with each direction producing one word per step (i.e. $c = 1$). As shown in Figure 1d, it splits the target sequence into several roughly equal segments followed by applying IBDecoder to each segment (thus an even h required). Formally, IMDecoder reframes the target sequence and word positions as follows:

$$\mathbf{y}^{\text{MD}} = \left[\mathbf{y}^{\text{BD}}_{1,k}, \mathbf{y}^{\text{BD}}_{2,k}, \dots, \mathbf{y}^{\text{BD}}_{h/2,k}\right]_{k=1}^{\lceil n/h \rceil}, \tag{9}$$
$$\mathcal{T}_t^{\text{MD}} = \left(\lfloor t-1/h \rfloor, t - 1 \bmod h\right), \tag{10}$$

where $\mathbf{y}^{\text{BD}}_{i,k}$ denotes the k-th word of $\mathbf{y}^{\text{BD}}_i$, which is the i-th segment of $\mathbf{y}$ reordered by IBDecoder($h/2$ segments in total). $\mathcal{T}^{\text{MD}}$ decomposes the word position into two parts. The first one represents the index of decoding step where each word is predicted; the second one denotes the generation direction each target word belongs to. Specifically, we record the corresponding direction indices and add a group of trainable direction embeddings (red rectangles in Figure 1d) into the decoder input. IMDecoder uses the following self-attention mask:

$$\mathbf{M}^{\text{MD}} = \mathcal{M}(h, 1) \tag{11}$$

Semi-Autoregressive Decoding Instead of partitioning the target sequence, another option is to produce multiple target words per direction at each

[2]Consider Figure 1a. We cannot reorder position encodings along with embeddings (1,6,2,5,...) because we do not know sentence length at test time. Simply using vanilla position encodings (1,2,3,4,...) would increase the embedding distance between positions within a direction.

[3]Note that with two tokens produced per time step, decoder inputs are shifted by two.

Algorithm 1 Beam search with step-wise multi-word generation.

```
Input: Decoder $dec$, beam size $B$, word number $z = h \cdot c$, maximum length $T$
Output: Top-$B$ finished hypothesis
        ▷ initial hypothesis (z start symbols, score 0)
 1: $\mathcal{H}_0 \leftarrow \{([\text{'[s]'}]^z, 0)\}$
 2: $\mathcal{H}_{finish} \leftarrow \emptyset$
 3: $t \leftarrow 0$
 4: while $|\mathcal{H}_{finish}| < B$ & $t < T$ do
 5:     for $(h_t, s_t) \in \mathcal{H}_t$ do
            ▷ words $W_p$ of probability $P \in \mathbb{R}^{z \times B}$
 6:         $\mathbf{P}, \mathbf{W}_p \leftarrow top_B(dec(h_t))$
            ▷ $\oplus$: outer addition for vectors
 7:         $\mathbf{s}, \mathbf{W}_s \leftarrow top_B(\oplus_{i=1}^{z} \log \mathbf{P}_i)$
            ▷ extract words by index, $W \in \mathbb{R}^{B \times z}$
 8:         $\mathbf{W} \leftarrow tracewords(\mathbf{W}_s, \mathbf{W}_p)$
 9:         for $(\mathbf{w}, s)$ in $(\mathbf{W}, \mathbf{s})$ do
                ▷ meet end-of-hypothesis condition
10:             if $finish(\mathbf{w})$ then
11:                 add $([h_t, \mathbf{w}], s + s_t)$ to $\mathcal{H}_{finish}$
12:             else
13:                 add $([h_t, \mathbf{w}], s + s_t)$ to $\mathcal{H}_{t+z}$
14:             end if
15:         end for
16:     end for
17:     prune $\mathcal{H}_{t+z}$ to keep top-$B$ hypothesis
18:     $t \leftarrow t + z$
19: end while
    ▷ $post(\cdot)$: process $h_t$ to recover word order
20: return sort $(post(h_t), s_t) \in \mathcal{H}_{finish}$ by $\frac{s_t}{t}$
```

step (i.e. increase c, Wang et al., 2018). SA assumes that neighbouring words are conditionally independent, despite the fact that tokens in natural language are typically highly inter-dependent.

We combine SA with IBDecoder (Figure 1e) with the expectation that producing 2 neighbouring tokens independently per direction is less harmful than producing 4 neighbouring words in parallel. We reuse the sequence $\mathbf{y}^{\text{BD}}$ and $\mathcal{T}^{\text{BD}}(n)$ for the decoder input, but enlarge the attention range in the self-attention mask to assist multi-word generation (Figure 1f):

$$\mathbf{M}^{\text{SA}} = \mathcal{M}(2, c). \tag{12}$$

4.2 Inference

To handle multiple predicted words per decoding step simultaneously, we adjust the beam search algorithm as in Algorithm 1. For each partial hypothesis h_t, we predict $z = h \cdot c$ words in parallel. We thus first extract the B top-scoring predictions $\mathbf{W}_p$ of probability $\mathbf{P}$ for all z positions (line 6), followed by pruning the resulting search space of size $\mathcal{O}(B^z)$ through an outer-addition operation to size B (line 7). The scores $\mathbf{s} \in \mathbb{R}^B$ (line 7) and the backtraced words $\mathbf{W} \in \mathbb{R}^{B \times z}$ (line 8) are then used for normal decoding. Note that each complete hypothesis requires a simple deterministic post-processing to recover its original word order (line 20). In contrast to Zhou et al. (2019), we do not separate the left-to-right beam from the right-to-left beam.

End-of-Hypothesis Condition With multiple predicted target words, determining whether one hypothesis is complete or not becomes challenging. We adopt a simple strategy: one hypothesis is assumed complete once any word in the predictions hits the end-of-sentence symbol ("[/s]") (line 10). We leave the study of alternatives for the future.

5 Experiments

Setup We test our model on machine translation (MT) and document summarization. We train MT models on five different language pairs: WMT14 English-German (En-De, Bojar et al., 2014), WMT14 English-French (En-Fr, Bojar et al., 2014), WMT16 Romanian-English (Ro-En, Bojar et al., 2016), WMT18 English-Russian (En-Ru, Bojar et al., 2018) and WAT17 Small-NMT English-Japanese (En-Ja, Nakazawa et al., 2017). Translation quality is measured by BLEU (Papineni et al., 2002), and we report detokenized BLEU using the toolkit *sacreBLEU* (Post, 2018)[4] except for En-Ja. Following Gu et al. (2019b), we segment Japanese text with KyTea[5] and compute tokenized BLEU. We train document summarization models on two benchmark datasets: the non-anonymized version of the CNN/Daily Mail dataset (CDMail, Hermann et al., 2015) and the Annotated English Gigaword (Gigaword, Rush et al., 2015). We evaluate the summarization quality using ROUGE-L (Lin, 2004).

We provide details of data preprocessing and model settings in Appendix A. We perform thorough analysis of our model on WMT14 En-De. We also report results improved by knowledge distillation (KD, Kim and Rush, 2016).

[4]Signature BLEU+c.mixed+#.1+s.exp+tok.13a+v.1.4.3

[5]http://www.phontron.com/kytea/

ID	Model	B	h	c	BLEU↑	+KD↑	Latency↓	Speedup↑	Train↑
1	Transformer	4	1	1	26.9	27.3	387	1.00×	1.00×
		1			26.0	26.8	294	1.32×	
2	IBDecoder	4	2	1	26.2	27.1	204	1.90×	0.98×
		1			25.0	26.8	166	2.33×	
3	2 + SA	4	2	2	23.0	26.3	117	3.31×	0.98×
		1			21.7	26.0	89	4.35×	
4	IMDecoder	4	4	1	21.5	24.6	102	3.79×	0.98×
		1			19.7	24.1	85	4.55×	

Table 1: Performance on WMT14 En-De for different models with respect to beam size (B), generation direction number (h, Eq. 4) and predicted token number per step (c, Eq. 4). *BLEU*: detokenized BLEU for models trained from scratch, *+KD*: detokenized BLEU for models trained with knowledge distillation. *Latency* (in millisecond) and *Speedup* are evaluated by decoding the test set with a batch size of 1, averaged over three runs. We report the latency and speedup for ②, ③ and ④ trained with KD. *Train* compares the training speed averaged over 100 steps. Time is measured on GeForce GTX 1080.

5.1 Results on WMT14 En-De

Table 1 compares the performance of our models on WMT14 En-De. Relaxing the autoregressiveness with IBDecoder yields slightly worse translation quality compared to Transformer (-0.7 BLEU, ①→②, w/o KD, $B = 4$). Unlike Zhang et al. (2020), we observe no quality improvement, but our model delivers a speedup of 1.90×∼2.33× at inference, clearly surpassing the simple greedy decoding baseline (1.32×) and BIFT (0.89×) (Zhang et al., 2020). The dropped quality is easily recovered with knowledge distillation (+0.2 BLEU, ①→②, w/ KD, $B = 4$).

Going beyond two-word generation, which enhances independence, greatly decreases the performance (②→③,④, w/o KD) while enlarging the speedup to 3.3×–4.5×. Compared to SA, the quality degradation with IMDecoder is larger, both w/ and w/o KD. We ascribe this to the difficulty of structure planning, as IMDecoder has to guess words in the middle of the sequence at the start of generation. We employ SA for the following experiments.

In contrast to existing work (Zhang et al., 2018d, 2019b, 2020; Zhou et al., 2019), our models marginally affect the training efficiency (0.98× vs 0.61× (Zhang et al., 2020)), and require no extra linguistic information (Akoury et al., 2019). Our results also suggest that the degree each model benefits from KD varies. Follow-up studies should report performance w/ and w/o KD.

Ablation Study We carry out an ablation study as shown in Table 2. Replacing the attention mask with the vanilla one (①→②) introduces unnecessary independence assumptions and reduces performance by 0.5 BLEU. Using vanilla positional encodings (③) also reduces performance -0.3 BLEU, indicating that we benefit from preserving the locality bias of sinusoidal encodings within each direction. Changing the generation direction from the side-to-middle (①) to the middle-to-side (④) dramatically increases the learning difficulty (-5.5 BLEU).

ID	Model	h	c	BLEU↑
1	IBDecoder	2	1	26.2
2	1 + vanilla mask	2	1	25.7
3	1 + vanilla positions	2	1	25.9
4	1 + middle-to-side	2	1	20.7
5	1 + indep. directions	2	1	23.9
6	vanilla SA	1	2	24.1
7	1 + SA	2	2	23.0
8	vanilla SA	1	4	18.7

Table 2: Ablation study on WMT14 En-De. Beam size 4. All models are trained from scratch. *vanilla mask*/*vanilla positions*: the self-attention mask ($\mathcal{M}(1, 1)$, Eq. 4) and word positions ($\mathcal{T}$, Eq. 5) used in Transformer. *middle-to-side*: generate words from the middle of the sequence to its two ends, a reverse mode of IBDecoder. *indep. directions*: disable cross-direction interaction. *vanilla SA*: predict multiple target words per step following one direction (Wang et al., 2018).

In IBDecoder, the two translation directions are interlinked, i.e. predictions are conditioned on the history of both directions. We can remove cross-direction attention, essentially forcing the model to produce the left and right half of sequences independently. Such an independent generation performs poorly (-2.3 BLEU, ①→⑤), which supports the importance of using bidirectional context and resonates with the finding of Zhou et al. (2019).

Vanilla SA vs. IBDecoder Our IBDecoder shares architectural properties with vanilla SA (Wang et al., 2018), namely the independent generation of two tokens per time step, and the

	Left-to-Right	Bidirectional
Autoregressive	4.04	4.86
Semi-Autoregressive	6.95	4.72
Estimated PMI	0.235	-0.014

Table 3: Perplexity of autoregressive and semi-autoregressive models with different factorizations, and estimated average point-wise mutual information between words that are predicted independently. Measured on WMT14 En-De test set. *Left-to-Right*: $h = 1$, *Bidirectional*: $h = 2$; Autoregressive: $z = 1$, Semi-autoregressive: $z = 2$. The estimated PMI shows that the inter-dependence of word pairs predicted in parallel by vanilla SA is stronger than for those predicted simultaneously by IBDecoder.

Model	$L/h/c$	BLEU↑	Speedup↑
Transformer	6/1/1	26.9	1.00×
+ student	2/1/1	26.0	2.19×
+ KD	2/1/1	26.7	2.32×
IBDecoder	6/2/1	26.2	1.90×
+ student	2/2/1	25.0	4.29×
+ KD	2/2/1	26.6	4.41×
IBDecoder + SA	6/2/2	23.0	3.31×
+ student	2/2/2	21.5	7.13×
+ KD	2/2/2	24.5	7.24×

Table 4: Detokenized BLEU and decoding speedup for student models on WMT14 En-De with reduced decoder depth L (encoder depth remains constant). Beam size 4.

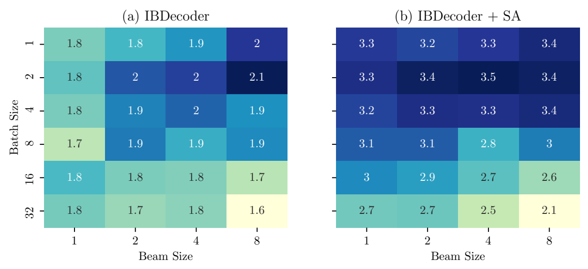


Figure 2: Speedup against Transformer vs. batch size and beam size on WMT14 En-De. Comparison is conducted under the same batch size and beam size. IBDecoder (+SA) is trained with KD. Our model consistently accelerates decoding.

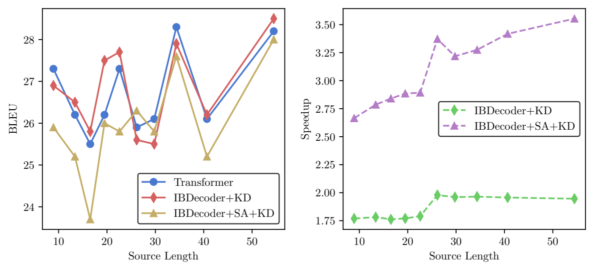


Figure 3: BLEU (solid lines, left) and speedup (dashed lines, right) as a function of source sentence length on WMT14 En-De. We sort the test set according to the source sentence length and uniformly divide it into 10 bins (274 sentences each). IBDecoder (+SA) is trained with KD. Beam size 4.

adapted self-attention mask, but crucially differ in their generation order and independence assumptions, with vanilla SA operating from left-to-right, and IBDecoder interleaving left-to-right and right-to-left decoding.

Our ablation results in Table 2 show that IBDecoder substantially outperforms vanilla SA (2.1/4.3 BLEU, ①→⑥/⑦→⑧). To further investigate the difference in independence assumptions between the two approaches, we compare estimated point-wise mutual information (PMI) of the words being predicted independently by IBDecoder and vanilla SA.[6] Results in Table 3 show that the PMI in IBDecoder (-0.014) is significantly smaller than that in vanilla SA (0.235), supporting our assumption that distant words are less inter-dependent on average. This also explains the smaller quality loss in IBDecoder compared to vanilla SA.

On Teacher-Student Model One classical approach to improving decoding efficiency is training a small student model w/ KD. Results in Table 4 support this: Transformer with a student model produces similar performance w/ KD but runs 2.32× faster, even better than IBDecoder (1.90×). Combining the student schema with IBDecoder increases the speedup to 4.41× without hurting the performance (26.6 BLEU, w/ KD). In exchange of 2.4 BLEU, we could reach 7.24× faster decoding with SA. The compatibility of our model with the teacher-student framework reflects the generalization of our bidirectional modeling. The results also demonstrate that efficiency improvements from faster autoregressive decoding, here obtained by reducing the number of decoder layers L[7], and from bidirectional decoding, are orthogonal.

Impact of Batch and Beam Size Figure 2 shows speedups over a standard Transformer with varying batch and beam sizes. When batch size < 8, increasing beam size improves the speedup; while the impact becomes negative with batch size ≥ 8. Overall, our model is consistently faster than Transformer at inference, regardless of the batch and beam size.

Impact of Source Sentence Length Although translation quality fluctuates over the source sentence length, Figure 3 shows that our model shares the same performance pattern with the baseline.

[6]Details about PMI estimation are given in Appendix B

[7]Also note the concurrent work by (Kasai et al., 2020).

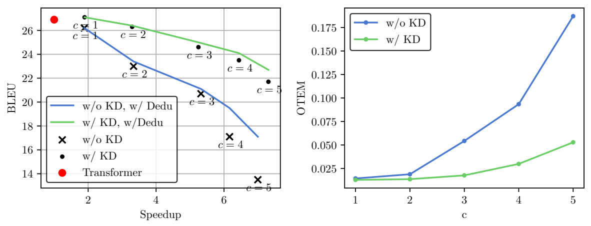


Figure 4: BLEU versus speedup (left) and OTEM (right) for different c on WMT14 En-De. Generation directions: $h = 2$. Beam size 4. OTEM↓: a metric measuring the degree of over-translation (Yang et al., 2018). Larger c indicates more independence between neighbouring tokens and results in more severe over-translation. Deduplication (Dedu) improves translation quality for large c.

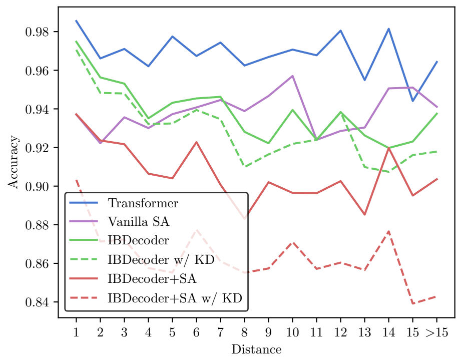


Figure 5: Accuracy of different models over distances on the subject-verb agreement task in *Lingeval97*.

With respect to the speedup, our model performs better when translating longer source sentences.

Effect of c Results in Figure 4 show that c controls the trade-off between translation quality and speedup. With larger c, more target tokens are predicted per decoding direction, leading to better speedup, but causing a larger performance drop w/ and w/o KD. Further analysis reveals that, as the dependency between predicted target words weakens, our model suffers from more serious over-translation issue, yielding larger OTEM (Yang et al., 2018). Although n-gram deduplication slightly improves quality[8], it does not explain the whole performance drop, echoing with Wang et al. (2018). We recommend using $c = 2$ for a good balance. In addition, the reduction of OTEM by KD in Figure 4 partially clarifies its improvement on quality.

Analysis on Long-range Dependency We adopt the subject-verb agreement task from *Lingeval97* (Sennrich, 2017) for analysis. We can see from the results in Figure 5 that IBDecoder

[8]we only applied deduplication for results in Figure 4.

Model		BLEU↑	SU↑
Existing work			
SAT (Wang et al., 2018)*		26.09†	2.07 ×
SBSG (Zhou et al., 2019)*		27.22†	1.61 ×
SynST (Akoury et al., 2019)		20.74	4.86×
Levenshtein (Gu et al., 2019b)*		27.27†	4.01×
CMLM (Ghazvininejad et al., 2019)*		27.03†	-
AXE (Ghazvininejad et al., 2020)*		23.53†	-
This work	SacreBLEU↑		
IBDecoder	25.0	25.73†	2.48×
w/ SA	22.3◇	22.95†	4.53×
w/ student	25.0	25.33†	4.29×
IBDecoder*	26.8	27.50†	2.33×
w/ SA*	26.0◇	26.84†◇	4.35×
w/ student*	26.6	27.00†	4.41×

Table 5: Comparison to several recent fast sequence generation models on WMT14 En-De. *: trained w/ KD. †: tokenized BLEU. ◇: deduplication applied. *SU*: short for speedup.

performs similarly to the original Transformer for agreement over short distances, but agreement over longer distances drops on average. In contrast, models that include SA show steep drops in accuracy for short distances.

Curiously, KD seems to harm agreement scores even though it led to higher BLEU. Overall, these results suggest that BLEU does not show the full quality loss incurred by our independence assumptions. This deficiency also provides evidence for the performance drop in Figure 4.

Comparison to Previous Work Results in Table 5 show that our model outperforms SynST (Akoury et al., 2019) in quality, and slightly surpasses the Levenshtein Transformer (Gu et al., 2019b) in speed. Particularly, our model ($27.50^\dagger/2.33\times$) surpasses SAT (Wang et al., 2018) ($26.09^\dagger/2.07\times$) and SBSG (Zhou et al., 2019) ($27.22^\dagger/1.61\times$) in terms of both quality and speed. Our model doesn't heavily rely on extra linguistic knowledge (Akoury et al., 2019), neither requires complex pseudo training data construction (Gu et al., 2019b). Compared to these prior studies, our approach is simple but effective.

5.2 Results on Other Tasks

Table 6 shows MT results for other translation directions, and for document summarization. Regardless of syntactic, morphological, transcript and sequence-length differences, our model achieves comparable generation quality and 1.75×–11.15× speedup over different tasks. With KD, our model even outperforms the Transformer baseline on 5 out of 6 tasks. In particular, our model succeeds on the

B		Model	KD	Machine Translation				Document Summarization	
				En-Fr	Ro-En	En-Ru	En-Ja	Gigaword	CDMail
4	Quality↑	Transformer	no	32.1	32.7	27.7	**43.97**	35.03	36.88
		IBDecoder	no	32.1	33.3	27.0	43.51	34.57	36.11
		+ SA	no	30.3	31.3	25.0	41.75	33.65	35.27
		IBDecoder	yes	**32.7**	**33.5**	27.5	43.76	35.12	36.46
		+ SA	yes	31.3	32.7	26.4	42.99	34.74	36.27
	Latency↓ /Speedup↑	IBDecoder	yes	231/1.75×	205/1.79×	204/1.82×	157/1.86×	83/2.35×	657/3.02×
		+SA	yes	119/3.41×	109/3.37×	112/3.30×	94/3.10×	47/4.20×	303/6.55×
1	Quality↑	Transformer	no	31.6	32.3	27.8	42.95	34.88	34.51
		IBDecoder	no	31.7	32.6	26.8	43.29	34.22	36.74
		+ SA	no	29.0	30.4	24.3	41.05	33.25	35.04
		IBDecoder	yes	32.2	33.2	**28.2**	43.79	**35.18**	**37.03**
		+ SA	yes	30.7	32.4	26.5	42.70	34.63	36.39
	Latency↓ /Speedup↑	Transformer	no	357/1.14×	333/1.10×	342/1.09×	260/1.12×	157/1.24×	1447/1.37×
		IBDecoder	yes	186/2.18×	154/2.37×	157/2.37×	121/2.40×	56/3.51×	312/6.36×
		+SA	yes	96/4.20×	88/4.17×	90/4.14×	67/4.34×	34/5.83×	178/11.15×

Table 6: Generation quality (BLEU for MT, Rouge-L for summarization) and latency(ms)/speedup on different tasks. We compare IBDecoder (+SA) with Transformer. Best quality is in **bold**.

CDMail task which previous non-autoregressive models rarely attempt due to its lengthy target sequence, although our model suffers from the long-range dependency issue as in Figure 5.

6 Conclusion and Future Work

We present interleaved bidirectional sequence generation to accelerate decoding by enabling generation from the left-to-right and right-to-left directions simultaneously. We combine the strengths of SBSG (Zhou et al., 2019) and SA (Wang et al., 2018), and propose a simple interleaved bidirectional decoder (IBDecoder) that can be easily implemented on top of a standard unidirectional decoder, like Transformer, via interleaving the target sequence and tweaking the word positions and self-attention masks. IBDecoder inherits Transformer's training parallelization with no additional model parameters, and is extensible with SA and multi-directional decoding. We show that the independence assumptions we introduce between the two directions are less harmful to translation quality than the independence assumptions in left-to-right SA. On a series of generation tasks, we report comparable quality with significant inference speedup (4×–11×) and little training overhead. We also show that the approach is orthogonal to speedups to autoregressive decoding, e.g. by reducing model size.

In the future, we would like to further improve multi-directional generation, and will investigate alternative ways to partition the target sequence and encode positional information. We are also interested in better measuring and reducing the quality loss resulting from long-distance dependencies. Finally, we would like to adapt our interleaving approach to other sequence-to-sequence architectures.

Acknowledgments

This work was performed using resources provided by the Cambridge Service for Data Driven Discovery (CSD3) operated by the University of Cambridge Research Computing Service (`http://www.csd3.cam.ac.uk/`), provided by Dell EMC and Intel using Tier-2 funding from the Engineering and Physical Sciences Research Council (capital grant EP/P020259/1), and DiRAC funding from the Science and Technology Facilities Council (`www.dirac.ac.uk`). Ivan Titov acknowledges support of the European Research Council (ERC Starting grant 678254) and the Dutch National Science Foundation (NWO VIDI 639.022.518). Rico Sennrich acknowledges support of the Swiss National Science Foundation (MUTAMUR; no. 176727).

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, Florence, Italy. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

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*.

Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA. Association for Computational Linguistics.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurélie Névéol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. 2016. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 131–198, Berlin, Germany. Association for Computational Linguistics.

Ondřej Bojar, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Philipp Koehn, and Christof Monz. 2018. Findings of the 2018 conference on machine translation (WMT18). In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 272–303, Belgium, Brussels. Association for Computational Linguistics.

Dmitrii Emelianenko, Elena Voita, and Pavel Serdyukov. 2019. Sequence modeling with unconstrained generation order. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 7700–7711. Curran Associates, Inc.

Andrew Finch and Eiichiro Sumita. 2009. Bidirectional phrase-based statistical machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1124–1132, Singapore. Association for Computational Linguistics.

Marjan Ghazvininejad, Vladimir Karpukhin, Luke Zettlemoyer, and Omer Levy. 2020. Aligned cross entropy for non-autoregressive machine translation. *ArXiv*, abs/2004.01655.

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

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

Jiatao Gu, Qi Liu, and Kyunghyun Cho. 2019a. Insertion-based decoding with automatically inferred generation order. *Transactions of the Association for Computational Linguistics*, 7:661–676.

Jiatao Gu, Changhan Wang, and Junbo Zhao. 2019b. Levenshtein transformer. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 11181–11191. Curran Associates, Inc.

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

Hiroaki Hayashi, Yusuke Oda, Alexandra Birch, Ioannis Konstas, Andrew Finch, Minh-Thang Luong, Graham Neubig, and Katsuhito Sudoh. 2019. Findings of the third workshop on neural generation and translation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 1–14, Hong Kong. Association for Computational Linguistics.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1693–1701. Curran Associates, Inc.

Xiaoguang Hu, Wei Li, Xiang Lan, Hua Wu, and Haifeng Wang. 2015. Improved beam search with constrained softmax for nmt. *Proceedings of MT Summit XV*, page 297.

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 *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2390–2399, Stockholmsmässan, Stockholm Sweden. PMLR.

Shigeki Karita, Nanxin Chen, Tomoki Hayashi, Takaaki Hori, Hirofumi Inaguma, Ziyan Jiang, Masao Someki, Nelson Enrique Yalta Soplin, Ryuichi Yamamoto, Xiaofei Wang, Shinji Watanabe, Takenori Yoshimura, and Wangyou Zhang. 2019. A comparative study on transformer vs rnn in speech applications. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 449–456.

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

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, Austin, Texas. Association for Computational Linguistics.

Young Jin Kim, Marcin Junczys-Dowmunt, Hany Hassan, Alham Fikri Aji, Kenneth Heafield, Roman Grundkiewicz, and Nikolay Bogoychev. 2019. From research to production and back: Ludicrously fast neural machine translation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 280–288, Hong Kong. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

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. Association for Computational Linguistics.

Jindřich Libovický and Jindřich Helcl. 2018. End-to-end non-autoregressive neural machine translation with connectionist temporal classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3016–3021, Brussels, Belgium. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Lemao Liu, Masao Utiyama, Andrew Finch, and Eiichiro Sumita. 2016. Agreement on target-bidirectional neural machine translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 411–416, San Diego, California. Association for Computational Linguistics.

Toshiaki Nakazawa, Shohei Higashiyama, Chenchen Ding, Hideya Mino, Isao Goto, Hideto Kazawa, Yusuke Oda, Graham Neubig, and Sadao Kurohashi. 2017. Overview of the 4th workshop on Asian translation. In *Proceedings of the 4th Workshop on Asian Translation (WAT2017)*, pages 1–54, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv e-prints*.

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.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal. Association for Computational Linguistics.

Abigail See, Minh-Thang Luong, and Christopher D. Manning. 2016. Compression of neural machine translation models via pruning. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 291–301, Berlin, Germany. Association for Computational Linguistics.

Rico Sennrich. 2017. How grammatical is character-level neural machine translation? assessing MT quality with contrastive translation pairs. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 376–382, Valencia, Spain. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Edinburgh neural machine translation systems for WMT 16. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 371–376, Berlin, Germany. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational*

Linguistics (Volume 1: Long Papers), pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Chenze Shao, Jinchao Zhang, Yang Feng, Fandong Meng, and Jie Zhou. 2020. Minimizing the bag-of-ngrams difference for non-autoregressive neural machine translation. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Noam Shazeer. 2019. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*.

Linfeng Song, Daniel Gildea, Yue Zhang, Zhiguo Wang, and Jinsong Su. 2019. Semantic neural machine translation using amr. *Transactions of the Association for Computational Linguistics*, 7:19–31.

Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. 2019. Insertion transformer: Flexible sequence generation via insertion operations. In *International Conference on Machine Learning*, pages 5976–5985.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. 2018. Blockwise parallel decoding for deep autoregressive models. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 10086–10095. Curran Associates, Inc.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Chengyi Wang, Shuangzhi Wu, and Shujie Liu. 2019. Accelerating transformer decoding via a hybrid of self-attention and recurrent neural network. *arXiv preprint arXiv:1909.02279*.

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

Taro Watanabe and Eiichiro Sumita. 2002. Bidirectional decoding for statistical machine translation. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Felix Wu, Angela Fan, Alexei Baevski, Yann Dauphin, and Michael Auli. 2019. Pay less attention with lightweight and dynamic convolutions. In *International Conference on Learning Representations*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Jing Yang, Biao Zhang, Yue Qin, Xiangwen Zhang, Qian Lin, and Jinsong Su. 2018. Otem&utem: Over-and under-translation evaluation metric for nmt. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 291–302. Springer.

Biao Zhang and Rico Sennrich. 2019. A lightweight recurrent network for sequence modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1538–1548, Florence, Italy. Association for Computational Linguistics.

Biao Zhang, Ivan Titov, and Rico Sennrich. 2019a. Improving deep transformer with depth-scaled initialization and merged attention. 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 898–909, Hong Kong, China. Association for Computational Linguistics.

Biao Zhang, Deyi Xiong, and Jinsong Su. 2018a. 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, Melbourne, Australia. Association for Computational Linguistics.

Biao Zhang, Deyi Xiong, Jinsong Su, Qian Lin, and Huiji Zhang. 2018b. Simplifying neural machine translation with addition-subtraction twin-gated recurrent networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4273–4283, Brussels, Belgium. Association for Computational Linguistics.

Biao Zhang, Deyi Xiong, Jinsong Su, and Jiebo Luo. 2019b. Future-aware knowledge distillation for neural machine translation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(12):2278–2287.

Jiajun Zhang, Long Zhou, Yang Zhao, and Chengqing Zong. 2020. Synchronous bidirectional inference for neural sequence generation. *Artificial Intelligence*, 281:103234.

Wen Zhang, Liang Huang, Yang Feng, Lei Shen, and Qun Liu. 2018c. Speeding up neural machine translation decoding by cube pruning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4284–4294, Brussels, Belgium. Association for Computational Linguistics.

Xiangwen Zhang, Jinsong Su, Yue Qin, Yang Liu, Rongrong Ji, and Hongji Wang. 2018d. Asynchronous bidirectional decoding for neural machine translation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Zhirui Zhang, Shuangzhi Wu, Shujie Liu, Mu Li, Ming Zhou, and Tong Xu. 2019c. Regularizing neural machine translation by target-bidirectional agreement. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 443–450.

Zaixiang Zheng, Shujian Huang, Zhaopeng Tu, Xin-Yu Dai, and Jiajun Chen. 2019. Dynamic past and future for neural 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 931–941, Hong Kong, China. Association for Computational Linguistics.

Long Zhou, Jiajun Zhang, Chengqing Zong, and Heng Yu. 2019. Sequence generation: from both sides to the middle. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 5471–5477. AAAI Press.

A Data Preprocessing and Model Settings

We use the given well-processed data for WAT17 En-Ja. For other tasks, we apply the byte pair encoding model (Sennrich et al., 2016b) with a joint vocab size of 32K except for WMT18 En-Ru (48K). We experiment with Transformer Base (Vaswani et al., 2017): $d = 512$, $L = 6$, 8 attention heads and FFN size of 2048. Dropout of rate 0.1 is used on residual connections and attention weights. We employ Adam ($\beta_1 = 0.9, \beta_2 = 0.98$) (Kingma and Ba, 2015) for parameter optimization with a scheduled learning rate of warm-up step 4K. Gradient is estimated over roughly 25K target subwords. We average the last 5 checkpoints for evaluation, and use beam search (beam size 4, length penalty 0.6) by default for inference.

B Estimation of the PMI

To evaluate the average point-wise mutual information (PMI) in Table 3, we compare IBDecoder/vanilla SA to its autoregressive counterpart in terms of testing perplexity (ppl). Take SA ($h = 1, c = 2$) as example, we have:

$$\text{PMI}(SA) = \log \text{ppl}(SA) - \log \text{ppl}(\text{Base}) \quad (13)$$

where *Base* denotes the baseline Transformer. The intuition behind our estimation is that Transformer handles neighboring words (y_1, y_2) autoregressively, thus models their joint probability: $p(y_1, y_2) = p(y_1) \cdot p(y_2|y_1)$. Instead, vanilla SA predicts those words independently, i.e. $p(y_1) \cdot p(y_2)$. Comparing the perplexity of SA and Transformer gives an estimation of the average PMI. The method for IBDecoder follows the same spirit.