JANUS: Joint Autoregressive and Non-autoregressive Training with Auxiliary Loss for Sequence Generation

Xiaobo Liang[1] **Juntao Li**[1] **Lijun Wu**[2] **Min Zhang**[1]
[1]Soochow University, [2]Microsoft Research
xbliang3@stu.suda.edu.cn, {ljt,minzhang}@suda.edu.cn
lijuwu@microsoft.com

Abstract

Transformer-based autoregressive and non-autoregressive models have played an essential role in sequence generation tasks. The autoregressive model can obtain excellent performance, while the non-autoregressive model brings fast decoding speed for inference. In this paper, we propose **JANUS**, a **J**oint **A**utoregressive and **N**on-autoregressive training method using a**U**xiliary los**S** to enhance the model performance in both AR and NAR manner simultaneously and effectively alleviate the problem of distribution discrepancy. Further, we pre-train BART with JANUS on a large corpus with minimal cost (16 GPU days) and make the BART-JANUS capable of non-autoregressive generation, demonstrating that our approach can transfer the AR knowledge to NAR. Empirically, we show our approach and BART-JANUS can achieve significant improvement on multiple generation tasks, including machine translation and GLGE benchmarks. Our code is available at Github[1].

1 Introduction

The transformer-based autoregressive (Vaswani et al., 2017; So et al., 2019) (AR) generation model has achieved high-quality results in various natural language generation tasks. Meanwhile, the non-autoregressive (NAR) (Gu et al., 2018; Lee et al., 2018; Libovický and Helcl, 2018; Su et al., 2021) generation methods show great potential to reduce the inference latency by introducing parallel decoding. Especially, iterative generative paradigms like CMLM (Wang et al., 2019a; Ghazvininejad et al., 2019) can dynamically adapt the trade-off between performance and latency. Recent works (Qi et al., 2021; Guo et al., 2020; Tian et al., 2020) have successfully combined these two mechanisms by joint training. However, these approaches only consider the relevance of model parameters, ignoring the correlations between the two manners, which require efforts for improvement. Therefore, in this paper, we attempt to leverage the merits of both AR and NAR mechanisms while avoiding their weaknesses to improve their performance.

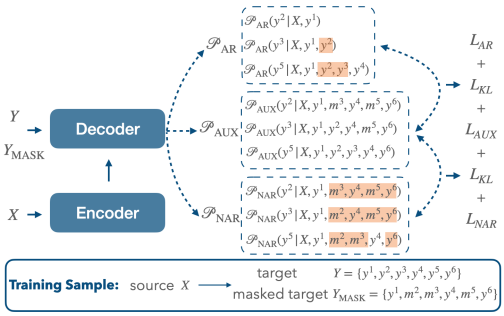

Figure 1: The example of JANUS. We exhibit some cases that show the distribution discrepancy due to the difference in context with orange background.

The principal difference between AR and NAR is that they use distinct attention mechanisms. In particular, the AR uses unidirectional attention (Vaswani et al., 2017) to simplify the sentence probability distributions by introducing Markov Hypothesis, which outputs the next token only depending on the previous context. This pattern makes the output distribution of each token accurate and unambiguous but also lacks diversity in inference. The NAR introduces bi-directional attention (Devlin et al., 2019) that can capture bi-directional context by considering the whole sentence information. This pattern needs to predict multiple tokens simultaneously, which causes the token distribution to be ambiguous, stemming from the multi-modality problem (Zhou et al., 2019). Inspired by knowledge distillation (Hinton et al., 2015) and deep mutual learning (Zhang et al., 2018b), we try to regularize the model predictions by minimizing the distribution distance between the output generated by the two manners. However, the conditional output probability of each target position is inconsistent between AR and NAR. We present an example in Figure 1 to il-

[1]https://github.com/dropreg/JANUS

Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 8050–8060
December 7-11, 2022

lustrate such a discrepancy, where the AR output probability $\mathcal{P}_{\mathrm{AR}}(y^3|X, y^1, \mathbf{y^2})$ for token y^3 depends on the context information $\{X, y^1, y^2\}$, and the NAR output $\mathcal{P}_{\mathrm{NAR}}(y^3|X, y^1, m^2, \mathbf{y^4}, m^5, \mathbf{y^6})$ relies on more context $\{X, y^1, m^2, y^4, m^5, y^6\}$. The $\mathcal{P}_{\mathrm{AR}}(y^3)$ lack the token $\mathbf{y^4}, \mathbf{y^6}$ as condition compare to $\mathcal{P}_{\mathrm{NAR}}(y^3)$, in contrast, the $\mathcal{P}_{\mathrm{NAR}}(y^3)$ misses the context $\mathbf{y^2}$. Such distinct token distributions bring difficulty in directly regularizing the prediction distances $\mathcal{D}_{(\mathrm{AR}|\mathrm{NAR})}$.

To tackle this issue, we introduce an auxiliary distribution $\mathcal{P}_{\mathrm{AUX}}$ to bridge the discrepancy between $\mathcal{P}_{\mathrm{AR}}$ and $\mathcal{P}_{\mathrm{NAR}}$ with the help of minimizing the distribution distance $\mathcal{D}_{(\mathrm{AR}|\mathrm{AUX})}$ and $\mathcal{D}_{(\mathrm{NAR}|\mathrm{AUX})}$ rather than the direct $\mathcal{D}_{(\mathrm{AR}|\mathrm{NAR})}$, as shown in Figure 1. This approach benefits AR and NAR to learn from each other with the following advantages: (1) The $\mathcal{P}_{\mathrm{AUX}}$ possesses rich context information compared with AR and NAR. For example, comparing to $\mathcal{P}_{\mathrm{AR}}$ and $\mathcal{P}_{\mathrm{NAR}}$, $\mathcal{P}_{\mathrm{AUX}}(y^3|X, y^1, \mathbf{y^2}, \mathbf{y^4}, m^5, \mathbf{y^6})$ contains all token $\mathbf{y^2}, \mathbf{y^4}, \mathbf{y^6}$ as condition. (2) There is an autoregressive dependency between the predicted tokens, which guarantees the accuracy of output distribution and without ambiguity for each token. (3) The random mask mechanism applied to this distribution, similar to NAR, enables the model to learn various token distributions. For effective implementation, we draw on the experience of two-stream self-attention (Yang et al., 2019b) and position compensation (Song et al., 2020). We build the P_{AUX} by altering the attention matrix of NAR, which merely modifies the training procedure without affecting the inference.

We first launched experiments on multiple NMT datasets to verify whether JANUS can help improve performance in two manners. Then, we explored our method on existing autoregressive pretraining models, in which we pre-train our BART-JANUS initialized by BART-base on a large corpus and fine-tuned to adapt various downstream tasks. Here we use the GLGE datasets as our benchmark to validate the model performance. Meanwhile, we also conducted comparative ablation studies to illustrate the effectiveness of our proposed method. Experimental results show that our method can achieve similar results to the state-of-the-art NAR model without distillation data. It simultaneously improves the AR model performance by more than 1.5 BLEU scores on average. Furthermore, our model exceeds the non-autoregressive pretraining model BANG on the same GLGE tasks. Perhaps surprisingly, it can achieve comparable performance with the AR manner at least two times speedup based on the iterative inference mechanism.

2 Related Work

How to take advantage of both AR and NAR paradigms for sequence generation has been discussed heatedly. Gu et al. (2018) and Zhou et al. (2019) introduced knowledge distillation (Hinton et al., 2015) into the training procedure of NAR, which can reduce the data complexity and allow the model to learn the variations from data. Some approaches (Guo et al., 2020; Sun and Yang, 2020; Hao et al., 2021) try to transfer AR knowledge to NAR by applying some specific learning strategy, which can gradually guide the model training. Zhou et al. (2020) proposed that the NAR model can effectively improve the AR performance by utilizing the NAR output with slight speed degradation. Several other works (Tian et al., 2020; Wang et al., 2022) have also proposed to improve the NAR performance by combining multiple decoding paradigms into a unified model.

The most related work is BANG (Qi et al., 2021), which proposed a novel cross-stream visible n-stream self-attention structure to bridge the gap between AR and NAR generation. Specifically, it supports different decoding manners using AR, semi-NAR, and NAR loss functions as training objectives. Unlike it, our approach not only focuses on joint training and parameter sharing but also considers the inner relationship of output distributions. Compared with previous work, we can integrate the merits of two different ways while circumventing their drawbacks by introducing an auxiliary loss. Some works (Zhang et al., 2018b; Shen et al., 2020; Liang et al., 2021) have explored that distribution regularization can make the model more robust. Nevertheless, the purpose of our work is more inclined to constrain two distinct training optimization objectives rather than the same. Besides, the pretraining models (Raffel et al., 2020; Fedus et al., 2022; Brown et al., 2020) can achieve robust generalization but suffer from huge model size. The model with massive parameters is prone to lead the unacceptable inference latency for generation tasks. Our approach attempts to leverage an existing autoregressive model (Lewis et al., 2020) to make it obtain the abilities of non-autoregressive rather than pre-train a NAR model from scratch.

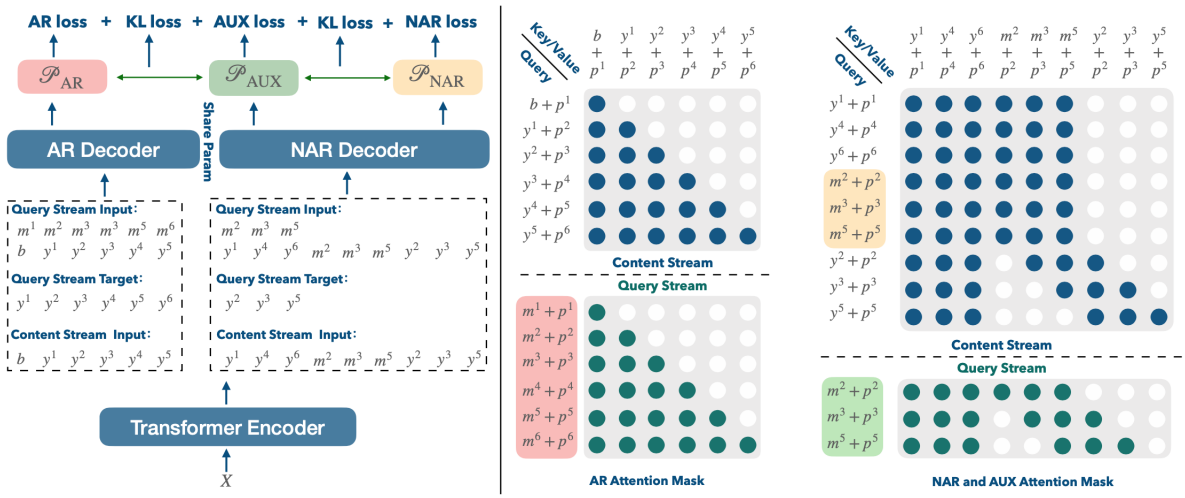

Figure 2: The overview of JANUS. The left side is the model architecture, in which we use the two-stream attention to train the model. The query streams require different keys and values for each input. The right side is the attention matrix of the decoder self-attention layer, and the white dots represent the masked similarity score. The symbol y is the word embedding, and p is the position embedding for decoder modules.

3 Approach

In this section, we elaborate on the details of our JANUS method, which supports AR and CMLM-based NAR generation and mutually enhances their performance. As shown in Figure 2, we introduce the two-stream self-attention mechanism to unify these two manners into our single model.

3.1 Model Architecture

We first introduce an auxiliary distribution (AUX) as a bridge between AR and NAR models. Considering that the predicted tokens in CMLM are independent of each other, ignoring their dependency, we integrate AUX with the NAR generation procedure using the two-stream self-attention mechanism, which is proposed by XLNet (Yang et al., 2019b) to overcome the limitations of MLM by introducing autoregressive information. More concretely, we use as an alternative of XLNet, i.e., MPNet (Song et al., 2020), to tackle the position discrepancy problem by position compensation, which inputs auxiliary position information to make the model see the whole sentence. Unlike its original function in addressing position discrepancy between pre-training and fine-tuning, we introduce MPNet to alleviate the distribution discrepancy between AR and NAR patterns.

In particular, given the training pairs (X, Y), the transformer encoder takes the source sentence X as input. The decoder accepts the multiple input sequences to build the attention matrix for different streams. For AR, we input an original sentence $\{b, y^1, ..., y^5\}$[2] and a fully masked sentence $\{m^1, ..., m^6\}$ into the decoder. For NAR and AUX, we first sample the masked target Y_{MASK} from the original sentence Y, e.g. $Y_{\text{MASK}} = \{y^1, m^2, m^3, y^4, m^5, y^6\}$. Then, we expand the masked sequence with the ground-truth token using position compensation and reorder the position of the whole sentence to obtain $Y_{\text{MASK}} = \{y^1, y^4, y^6, m^2, m^3, m^5, y^2, y^3, y^5\}$. Finally, we take the reordered sentence Y_{MASK} and masked sentence $\{m^2, m^3, m^5\}$ as decoder inputs. The decoder can output the distributions according to the corresponding generation paradigm by manipulating the attention matrix of each decoder self-attention layer.

AR Pattern The AR decoder suffers from the position shift of the predicted token due to sharing parameters with the NAR decoder. That is, the decoder is confused about whether to output the current or the next token. So, we use the MASK token in the query stream instead of the content stream to generate the output distribution. Specifically, we build the lower triangular matrix for both the query and content stream. The content stream is responsible for encoding the context information, and the query stream is for prediction. The content stream takes the hidden states of $\{b, y^1, ..., y^5\}$ as input to ensure each position can see the previous position. The query stream predicts each masked token according to the previous content stream, e.g.,

[2]We append a special token b in front of sentence Y.

m^2 generates y^2 according to $\{b, y^1\}$.

NAR Pattern We only use the content stream for the NAR manner and leverage the bi-directional attention to generate the output distribution. The bi-directional attention operation only depends on reordered target sequences $\{y^1, y^4, y^6, m^2, m^3, m^5\}$, which is the front part of Y_{MASK}. It is worth noting that the bidirectional self-attention operation is position independent, so sentence reordering does not affect the training procedure.

AUX Pattern We employ a specific attention mask to build the auxiliary distribution. Specifically, the content stream take the Y_{MASK} as input, and split the tokens in a sequence into original and compensation parts. For example, the previous sentence $\{y^1, y^4, y^6, m^2, m^3, m^5\}$ is original parts, and $\{y^2, y^3, y^5\}$ is the compensation part. The content stream ensures that each position in compensation part can see the input token instead of the MASK, e.g. y^3 based on context $\{y^1, y^4, y^6, y^2, y^3, m^5\}$ instead of original parts. Unlike it, the query stream predicts the next token of masked sequence in an autoregressive manner, e.g. m^3 based on context $\{y^1, y^4, y^6, y^2, m^3, m^5\}$, and m^5 based on context $\{y^1, y^4, y^6, y^2, y^3, m^5\}$.

In this way, we can generate various output distributions by switching the attention matrix, which is a sample-aware operation because each sentence and masked spans have an arbitrary length.

3.2 Training and Inference

AR The training objective of AR generation manner is to minimize the following cross-entropy loss:

$$\mathcal{L}_{\text{AR}} = -\sum_{i}^{N} \log \mathcal{P}_{\text{AR}}(y_i | X, Y_{<i}), \quad (1)$$

where $Y_{<i}$ refers to the tokens before the i-th time step, N is the target length, and the output probability $\mathcal{P}(y_i)$ equals $\mathcal{P}(m_i)$ generated by the query stream. We still use the output probability of query stream during inference to predict the results.

NAR We use span masking as a token sampling method, which masks contiguous random spans rather than tokens. We select each span by sampling from a Poisson distribution ($\lambda = 2$) inspired by (Joshi et al., 2020; Lewis et al., 2020), and each token in the span is replaced with MASK token. The training loss function is to minimize the sum of negative log-likelihood for masked sequence:

$$\mathcal{L}_{\text{NAR}} = -\sum_{i}^{M} \log \mathcal{P}_{\text{NAR}}(y_i | X, Y_{\text{MASK}}), \quad (2)$$

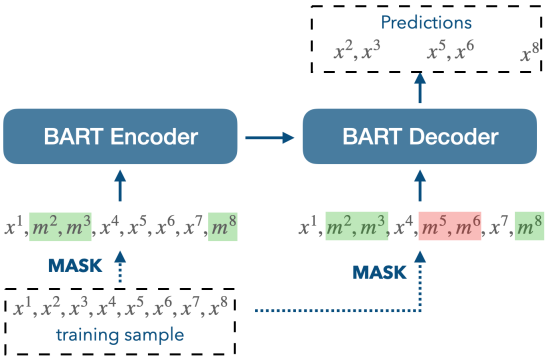

Figure 3: Different Span masking is applied to BART encoder and decoder input to avoid decoder degradation.

where M is the masked sequence length. During inference, the decoder refines the low-probability tokens starting with a fully masked sentence by iterative masking and prediction.

AUX The training loss function is to minimize the sum of negative log-likelihood for masked token generated by query stream:

$$\mathcal{L}_{\text{AUX}} = -\sum_{i}^{M} \log \mathcal{P}_{\text{AUX}}(y_i | X, Y_{<i} \cup Y_{\text{Mask}}), \quad (3)$$

where the $\cup$ operation indicated removing duplicate mask tokens and keeping original tokens in context.

Then, we use the auxiliary distribution P_{AUX} as a bridge to regularize the two distributions P_{AR} and P_{NAR} by minimizing the token-level bidirectional Kullback-Leibler divergence:

$$\begin{aligned}\mathcal{L}_{\text{KL}} = \mathcal{D}_{\text{KL}}(\mathcal{P}_{\text{AR}} || \mathcal{P}_{\text{AUX}}) + \mathcal{D}_{\text{KL}}(\mathcal{P}_{\text{AUX}} || \mathcal{P}_{\text{AR}}) \\ + \mathcal{D}_{\text{KL}}(\mathcal{P}_{\text{NAR}} || \mathcal{P}_{\text{AUX}}) + \mathcal{D}_{\text{KL}}(\mathcal{P}_{\text{AUX}} || \mathcal{P}_{\text{NAR}}).\end{aligned} \quad (4)$$

The final loss we used in JANUS is a combination of cross-entropy and KL losses:

$$\mathcal{L} = \mathcal{L}_{\text{AR}} + \mathcal{L}_{\text{NAR}} + \mathcal{L}_{\text{AUX}} + \mathcal{L}_{\text{KL}}. \quad (5)$$

3.3 Pretraining and Finetuning

We run our approach on the second phase of pretraining based on an existing encoder-decoder pretrained model like BART. Specifically, we use span masking for decoder input when switching to the NAR manner and replace each token with a mask rather than each span. However, this practice causes the encoder and decoder input to be similar, and it may force the decoder to predict results through its own context, ignoring the information from the encoder. In order to prevent the phenomenon of decoder degeneration, we further mask more spans for decoder input to ensure that the decoder needs to combine the encoder information to make a prediction, as shown in Figure 3.

After pretraining, we use the traditional AR and NAR approach to finetune it on downstream tasks. We provide new target length prediction modules with random initialization for NAR models, which take this problem as a classification task. We average the encoder representations of the whole source sentence and map it to a vector with the maximal sentence size to obtain the target length. Then, we apply the predicted length to initialize the decoder input during inference.

4 Experimental Setup

4.1 Task

We validate our approach on multiple sequence generation tasks. For neural machine translation, we train our model from scratch. For other tasks, we first perform pre-training using our methods on BART-base to obtain BART-JANUS, then use the downstream task to finetune the BART-JANUS. All experiments are done using the fairseq[3] library.

Neural Machine Translation We conduct our experiments on three public datasets: IWSLT14 German→English, WMT14 German↔English[4], and WMT16 Romanian↔English[5]. We only use the raw data instead of the distilled data to verify the effectiveness of our method.

GLGE We use the GLGE[6] (General Language Generation Evaluation Benchmark) (Liu et al., 2021) to evaluate the ability of BART-JANUS on sequence generation tasks. We selected three different tasks: Abstractive Text Summarization dataset Xsum (Rush et al., 2015) (227K article and summary pairs), Answer-aware Question Generation dataset SQuAD 1.1 (Rajpurkar et al., 2016) (98K answer, passage, and question data triples), and Conversational Question Answering dataset PersonaChat (Zhang et al., 2018a) (150k persona profile, conversation history, and response data triples). In particular, we choose GLGE-Easy from different ranks of difficulty set as our dataset. For a detailed description, please refer to the original paper.

4.2 Implementation

For machine translation experiments, we use traditional `transformer_iwslt_de_en` setting for IWSLT14 dataset, which contains 6 layers in both encoder and decoder, the embedding size is 512 and the FFN layer dimension is $1,024$, the dropout and weight decay is 0.3 and 0.0001 respectively. We use the model configuration `transformer` for WMT dataset, with 6 layers, the embedding size is 512 and the FFN layer size is $2,048$, the dropout value is set to be 0.2 for De↔En and 0.3 for Ro↔En. We adopt the default optimization algorithm and learning rate schedule as in (Vaswani et al., 2017), that is Adam (Kingma and Ba, 2014) optimizer with initial learning rate 0.0005, learning rate schedule `inverse_sqrt` with $10,000$ warmup steps. Label smoothing is utilized in the loss function with a value of 0.1. The raw texts are encoded using BPE (Sennrich et al., 2016) as the subword units and report test set performance by measuring BLEU (Papineni et al., 2002). Specifically, we evaluate the final translation accuracy by averaging 5 checkpoints in both AR and NAR settings.

For pretraining experiments, we utilize BART-base to initialize our model, which contains 6 layers in both encoder and decoder, the embedding size is 768 and the FFN layer size is 3072, dropout and attention dropout is 0.1, and 140M model parameters in total. We mask 30% token span for encoder input and 50% for decoder input. We pre-train BART-JANUS using the 16GB corpus (Wikipedia and BookCorpus) following Qi et al. (2021) on 8 NVIDIA 40GB A100 GPUs, with a learning rate of 1e-4 and a batch size of 1024 sentences only for 2 epochs. The total training procedure uses FP16 for speedup and needs 2 days.

For GLGE downstream tasks, we load the BART-JANUS and finetune it with the traditional AR or NAR strategy. For NAR settings, we use learning rate 1e-4, warmup steps of 1000, Adam optimizer, and a label smoothness of 0.1 for 50 epochs. We set the maximal output length as 64, 64, and 32 for SQuAD, XSum and PersonaChat, respectively. For AR settings, we use learning rate 3e-5, 200 warmup steps for 10 epochs. For inference stage, we set the beam size as 5 for AR manners, and take top-5 predicted lengths and select the translation with the highest average token probability for NAR manners. We save the checkpoint for every 10 epochs and select the best checkpoint based on the performance on the validation set. The official script[7] is used to evaluate the model performance.

[3] https://github.com/facebookresearch/fairseq

[4] https://github.com/facebookresearch/fairseq/tree/main/examples/nonautoregressive_translation

[5] https://github.com/facebookresearch/DisCo/issues/5

[6] https://github.com/microsoft/glge

[7] https://github.com/microsoft/ProphetNet/blob/master/GLGE_baselines/script/eval.py

Method	IWSLT'14 De→En	WMT'16 En↔Ro		WMT'14 En↔De		Iter
	De→En	Ro→En	En→Ro	De→En	En→De	
Transformer (Vaswani et al., 2017)	34.74	34.46	34.16	31.06	27.74	#
Convolutional Transformer (Yang et al., 2019a)	-	-	-	-	28.2	#
Adversarial Training (Wang et al., 2019b)	35.2	-	-	-	28.4	#
Flowseq (Ma et al., 2019)	-	32.91	32.35	28.29	23.64	1
GLAT (Qian et al., 2021)	32.49	32.00	31.19	29.84	25.21	1
CMLM (Ghazvininejad et al., 2019)	32.10	32.87	32.86	29.40	24.61	10
DisCo (Kasai et al., 2020)	-	32.25	-	-	25.64	4
CMLMC (Huang et al., 2021)	**34.28**	34.13	**34.14**	**30.92**	26.40	10
CMLM+Corr	33.90	33.98	33.75	30.55	26.10	10
JANUS-AR	**37.24** †	**35.84**	**36.01**	**33.09**	**28.72**	#
JANUS-NAR	34.21 †	**34.36**	34.00	30.90	**26.40**	10

Table 1: Results of NMT dataset. The AR-based model does not use the average checkpoint, but JANUS uses it following the NAR standard setting. († represent the p-value < 0.01 according to significance test).

Pattern	Method	Iter	ROUGE-1	ROUGE-2	ROUGE-L	OVERALL	Latency(ms/Sample)
AR	Transformer (Vaswani et al., 2017)		30.66	10.80	24.48	21.98	262.47
	ProphetNet-base (Qi et al., 2020)		39.89	17.12	32.07	29.69	N/A
	BANG (Qi et al., 2021)		41.09	**18.37**	**33.22**	**30.89**	N/A
	BART-base (Lewis et al., 2020)		38.79	16.16	30.61	28.52	N/A
	BART-base (Our impl)		41.22	17.98	32.69	30.63	322.78* (1.0 ×)
	BART-JANUS		**41.29**	18.16	32.78	30.74	332.49*
NAR	BANG semi-NAR (Qi et al., 2021)		34.71	11.71	29.16	25.19	109.77
	BANG (Qi et al., 2021)		32.59	8.98	27.41	22.99	15.97
	BART-JANUS	0	33.29	9.74	28.21	23.74	36.66* (8.7 ×)
		1	38.05	13.50	31.18	27.57	45.51* (7.0 ×)
		4	40.65	16.56	32.86	30.02	71.85* (4.5 ×)
		10	**41.53**	**17.72**	**33.40**	**30.88**	128.94* (2.5 ×)

Table 2: Results on XSum dataset. We reimplemented the BART-base result, another result from BANG.(∗ represent the result based on our environment. ms/Sample represents the number of milliseconds consumed per sample).

5 Results

5.1 Neural Machine Translation

The Results of the NMT dataset are shown in Table 1. Our JANUS-AR model surpasses the robust baseline with the convolutional transformer or the adversarial training strategy. Moreover, the JANUS-NAR model achieves significant improvement above the CMLM baseline by nearly 1.5 BLEU points [8]. We can see that JANUS-NAR performs better than several strong baselines, such as Flowseq, GLAT, and Disco. Our model achieves similar results compared with the recent SOTA NAR model CMLMC on multiple datasets (trained with raw data) and surpasses them on Ro→En and En→De datasets. For a fair comparison, CMLM+Corr is based on the CMLM model without introducing additional parameters to address the problem of indistinguishability of tokens, and we consistently surpass it. At the same time, we can see that the JANUS-AR model outperforms all baselines with more than 1.5 BLEU scores on average. These results all support that JANUS can make two different manners benefit from each other by joint training and distribution regularization.

[8]We choose the commonly used mosesdecoder-script bootstrap-hypothesis-difference-significance.pl to conduct the significance test.

Method	Iter=1	Iter=4	Iter=10
CMLM	7.42%	1.58%	0.83%
JANUS	6.8%	1.43%	0.65%

Table 3: Token repetition of different iteration step.

Furthermore, the model may suffer from the multi-modality problem when trained with raw data because each source has multiple target candidates. We calculate the token repetition ratio of generated results, which can reflect the degree of the multi-modality. The results are shown in Table 3. JANUS has low token repetition ratio than CMLM, demonstrating that it effectively alleviates the multi-modality problem in NAR by carrying autoregressive information. We can keep inference latency the same as CMLM since our approach only modifies the training paradigm (Section 6.2 gives a more thorough analysis).

Pattern	Method	Iter	ROUGE-L	BLEU-4	METEOR	OVERALL	Latency(ms/Sample)
AR	Transformer (Vaswani et al., 2017)		29.43	4.61	9.86	14.63	159.49
	ProphetNet-base (Qi et al., 2020)		48.00	19.58	23.94	30.51	N/A
	BANG (Qi et al., 2021)		**49.32**	**21.40**	**24.25**	**31.66**	N/A
	BART-base (Qi et al., 2021)		42.55	17.08	23.19	27.61	N/A
	BART-base (Our impl)		43.02	17.57	23.52	28.03	157.49* (1.0 ×)
	BART-JANUS		44.70	17.36	24.07	28.71	163.16*
NAR	BANG semi-NAR (Qi et al., 2021)		47.39	**17.62**	**21.69**	**28.90**	111.11
	BANG (Qi et al., 2021)		44.07	12.75	18.99	25.27	15.69
	BART-JANUS	0	44.99	11.18	17.89	24.68	33.52* (4.6 ×)
		1	47.18	14.35	20.14	27.22	42.51* (3.7 ×)
		4	47.91	16.38	21.36	28.55	68.7* (2.3 ×)
		10	**48.00**	16.87	21.58	28.81	127.51* (1.2 ×)

Table 4: Result on SQuAD 1.1.

Pattern	Method	Iter	BLEU-1	BLEU-2	Distinct-1	Distinct-2	OVERALL	Latency(ms/Sample)
AR	Transformer (Vaswani et al., 2017)		41.56	32.95	0.3	0.8	18.90	138.31
	ProphetNet-base (Qi et al., 2020)		46.00	38.40	1.3	7.3	23.25	N/A
	BANG (Qi et al., 2021)		45.77	35.54	**1.4**	**8.4**	22.78	N/A
	BART-base (Lewis et al., 2020)		47.60	39.36	1.1	6.1	23.54	N/A
	BART-base (Our impl)		50.33	40.11	1.2	7.3	24.73	183.09* (1.0 ×)
	BART-JANUS		**51.16**	**40.32**	1.2	7.2	**24.97**	187.77*
NAR	BANG semi-NAR (Qi et al., 2021)		39.82	30.72	1.9	14.2	21.66	109.17
	BANG (Qi et al., 2021)		31.11	23.90	**2.5**	**22.7**	20.05	14.89
	BART-JANUS	0	36.16	34.37	2.4	17.8	22.68	33.81* (5.4 ×)
		1	42.76	37.36	2.1	16.4	**24.65**	41.88* (4.4 ×)
		4	44.68	37.73	1.8	13.1	24.32	67.90* (2.7 ×)
		10	**45.21**	**37.90**	1.6	11.1	23.95	124.42* (1.5 ×)

Table 5: Result on PersonaChat.

5.2 GLGE

We present the results of the XSum tasks in Table 2. In AR model generation, BART-JANUS has slightly improved compared to BART-base on all metrics ROUGE-1, ROUGE-2, and ROUGE-L. These results show that it is challenging to improve the pre-trained BART model in a short time through continual training. Moreover, our focus is on whether BART-JANUS can gain non-autoregressive generation capabilities. We can see that when the number of iterations is equal to 0, the model has exceeded the BANG-NAR 0.75 score. Surprisingly, the NAR model outperforms the AR model when the iteration number reaches 10. This result implies that the bi-directional attention under NAR can effectively use the comprehensive information of the decoder input to obtain beneficial representations, which is a more suitable way than the uni-directional attention of the AR model. In inference latency, we set batch size as 1 to calculate the latency and see that the NAR model can surpass the AR model with 2.5 times speedup. Furthermore, the BART-JANUS can progressively improve the model performance by varying the iteration size.

The results of SQuAD 1.1 are shown in Table 4. Since our model is a two-stage pre-training with a relatively low training cost, the AR model may not obtain a significant improvement and limits the upper bound of the NAR model. We can observe that BART-JANUS does not exceed BANG in most metrics due to the limitation of BART-base performance. Nevertheless, our NAR model achieves comparable results to BANG semi-NAR when the iteration number reaches 10 with a similar speedup.

From Table 5, we can see that our model consistently outperforms all baseline on all evaluation scores for Personachat datasets. For dialog generation, the Distinct-1 and Distinct-2 metrics are to prevent the model from generating meaningless and boring responses. The BART-JANUS obtains a high BLEU and Distinct value, confirming that our model can simultaneously consider the fluency and diversity in the generated dialog response.

6 Analysis and Discussion

6.1 Ablation Study

We present exhaustive investigations on IWSLT14 De→En and WMT16 Ro→En datasets in both AR and NAR and use the Transformer (Vaswani et al., 2017) and CMLM (Ghazvininejad et al., 2019) as

Loss				IWSLT De→En		WMT Ro→En	
AR	NAR	AUX	KL	AR	NAR	AR	NAR
✓	✗	✗	✗	34.74	-	34.46	-
✗	✓	✗	✗	-	32.10	-	32.87
✗	✗	✓	✗	0.91	24.41	-	-
✓	✓	✗	✗	35.25	32.12	34.50	33.01
✓	✓	✗	✓	36.08	33.45	35.13	33.64
✓	✓	✓	✓	**37.24**	**34.21**	**35.84**	**34.36**

Table 6: Results and ablation study on the NMT datasets.

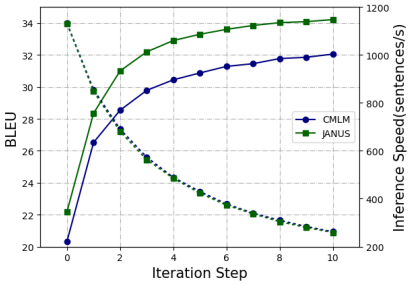

Figure 4: Results by varying the number of iterations (solid line is performance, dotted line is inference speed).

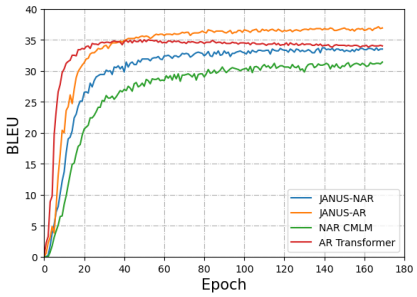

Figure 5: BLEU curves along with the model training.

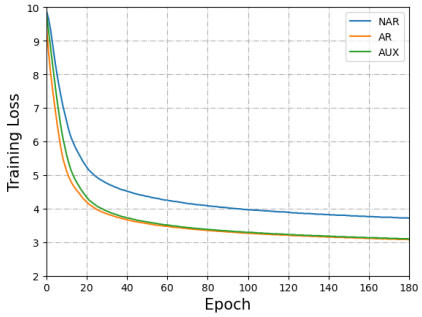

Figure 6: Training loss curves along with training epoch.

corresponding training strategies for baseline. The results are shown in Table 6. We compare the combinations of various losses to investigate the interaction between them, i.e., AR, NAR, AUX, and KL loss. The NAR and AR models outperform the model with joint training by using the KL regularization item, proving that output space is more reasonable than parameter space for enhancing each other. The AUX can further improve the performance (37.24 v.s. 36.80, 34.21 v.s. 33.25, 35.84 v.s. 35.13, and 34.36 v.s. 33.64) by alleviating the distribution discrepancy between AR and NAR patterns. Besides, we examine whether AUX is a powerful teacher that can distill knowledge to NAR and AR models in the third line. The results show AUX itself is not a good teacher, which demonstrates poor performance with both the AR and NAR inference patterns. In short, all the above evidence support that JANUS can achieve the mutual enhancement of AR and NAR rather than distilling the AUX knowledge for AR and NAR training in a mutual learning fashion.

6.2 Effect of Iteration Step

The analysis of iterative refinement is shown in Figure 4. For both CMLM and JAUNS, we compute the BLEU score at different iterations. The JANUS can significantly improve performance in each iteration step and surpass CMLM with only three iterations. The inference speed is consistent with CMLM since our solution only focuses on improving the training strategy, calculated by translating the IWSLT14 De→En test set.

6.3 Training Analysis

To better show the training cost and performance of JANUS, we study the model training from different aspects. We give detailed comparisons of the training time on the IWSLT14 De→En dataset when the model converges. Specifically, the total training cost of JANUS is about 25,235 seconds for 170 epochs on 2 GPUs, while CMLM and AR Transformers costs about 9,852 seconds and 3,975 seconds on 1 GPU, respectively. We can see that JANUS does require 4 or 5 times the training cost. Although the model needs more time to converge, JANUS has achieved significant performance improvement. Furthermore, we also study the training process and visualize the performance improvements in Figure 5. We plot the test BLEU curves along with the training updates. From the curves, we can find that JANUS exceeds the baseline at the same epoch and can continue to grow.

As shown in Figure 6, we plot the curves of three different training losses of the model on IWSLT14 De→En neural machine translation tasks. We can see that the loss value belongs to the same scale

Method	ROUGLE-L	BLEU-4	METEOR	OVERALL
BART-base	42.45	8.92	16.42	22.59
BART-CMLM	41.93	9.01	16.86	22.60
BART-JANUS	44.99	11.18	17.89	24.68

Table 7: Results on SQuAD 1.1 with various pretraining.

Method	ROUGLE-1	ROUGLE-2	ROUGLE-L	OVERALL
BART-base	31.55	8.47	26.95	22.32
BART-CMLM	32.47	8.97	27.60	23.01
BART-JANUS	33.29	9.74	28.21	23.74

Table 8: Results on XSum with various pretraining

and does not require an external balance coefficient. The auxiliary loss can keep a similar scope to AR loss in training, showing that the AUX is more reasonable to bridge AR and NAR distributions.

6.4 Effect of Various Pretraining

In this section, we consider the impact of model performance for different pretrained models as initializations on the NAR model. We provide BART-base as model initialization and then use CMLM as finetuning method. Further, We use CMLM to pre-train BART and keep consistent with BART-JANUS in other settings. Results show that the JANUS can improve NAR finetuning results significantly in both tasks. This result demonstrates that the proposed pretraining strategy via transferring the knowledge from AR to NAR is critical to achieving a better performance in NAR generation.

7 Conclusion

In this work, we propose a new training strategy named JANUS, which supports the AR, NAR, and iterative refinement generation mechanisms. Meanwhile, we tackle the problem of distribution discrepancy between the AR and NAR by introducing an auxiliary distribution. Experiments show that JANUS can significantly improve the AR and NAR model performance. Further, we pre-train BART-JANUS and achieve comparable performance with the NAR pertained model BANG. We are also interested in designing effective finetuning strategies to apply JANUS, which leaves it as future work.

8 Limitation

Our model gains noticeable performance but suffers from training costs not being neglected. However, we require a particular attention matrix to build the auxiliary distribution. This sample-aware matrix is different from the vanilla transformer in terms of implementation. The vanilla transformer attention is a 2D matrix, but JANUS needs a 3D matrix. Consequently, it leads to a lot of time consumption, as shown in section 6.3. In the future, we hope to speed up model training by exploring a better solution or using lower-level CUDA operators to create this matrix.

Acknowledgement

We would like to thank the anonymous reviewers for their constructive comments. Juntao Li is the corresponding author. This work was supported by the National Science Foundation of China (NSFC No. 62206194), the Natural Science Foundation of Jiangsu Province, China (Grant No. BK20220488), and the Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

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.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39.

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.

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

Junliang Guo, Xu Tan, Linli Xu, Tao Qin, Enhong Chen, and Tie-Yan Liu. 2020. Fine-tuning by curriculum learning for non-autoregressive neural machine translation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7839–7846.

Yongchang Hao, Shilin He, Wenxiang Jiao, Zhaopeng Tu, Michael Lyu, and Xing Wang. 2021. Multi-task learning with shared encoder for non-autoregressive machine translation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3989–3996.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Xiao Shi Huang, Felipe Perez, and Maksims Volkovs. 2021. Improving non-autoregressive translation models without distillation. In *International Conference on Learning Representations*.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Jungo Kasai, James Cross, Marjan Ghazvininejad, and Jiatao Gu. 2020. Non-autoregressive machine translation with disentangled context transformer. In *International Conference on Machine Learning*, pages 5144–5155. PMLR.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *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.

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

Xiaobo Liang, Lijun Wu, Juntao Li, Yue Wang, Qi Meng, Tao Qin, Wei Chen, Min Zhang, Tie-Yan Liu, et al. 2021. R-drop: Regularized dropout for neural networks. *Advances in Neural Information Processing Systems*, 34:10890–10905.

Jindřich Libovickỳ 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.

Dayiheng Liu, Yu Yan, Yeyun Gong, Weizhen Qi, Hang Zhang, Jian Jiao, Weizhu Chen, Jie Fu, Linjun Shou, Ming Gong, et al. 2021. Glge: A new general language generation evaluation benchmark. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 408–420.

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 4282–4292.

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.

Weizhen Qi, Yeyun Gong, Jian Jiao, Yu Yan, Weizhu Chen, Dayiheng Liu, Kewen Tang, Houqiang Li, Jiusheng Chen, Ruofei Zhang, et al. 2021. Bang: Bridging autoregressive and non-autoregressive generation with large scale pretraining. In *International Conference on Machine Learning*, pages 8630–8639. PMLR.

Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. 2020. Prophetnet: Predicting future n-gram for sequence-to-sequencepre-training. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2401–2410.

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.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

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.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. 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.

Dinghan Shen, Mingzhi Zheng, Yelong Shen, Yanru Qu, and Weizhu Chen. 2020. A simple but tough-to-beat data augmentation approach for natural language understanding and generation. *arXiv preprint arXiv:2009.13818*.

David So, Quoc Le, and Chen Liang. 2019. The evolved transformer. In *International Conference on Machine Learning*, pages 5877–5886. PMLR.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. Mpnet: Masked and permuted pre-training for language understanding. *Advances in Neural Information Processing Systems*, 33:16857–16867.

Yixuan Su, Deng Cai, Yan Wang, David Vandyke, Simon Baker, Piji Li, and Nigel Collier. 2021. Non-autoregressive text generation with pre-trained language models. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 234–243.

Zhiqing Sun and Yiming Yang. 2020. An em approach to non-autoregressive conditional sequence generation. In *International Conference on Machine Learning*, pages 9249–9258. PMLR.

Chao Tian, Yifei Wang, Hao Cheng, Yijiang Lian, and Zhihua Zhang. 2020. Train once, and decode as you like. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 280–293.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Alex Wang, Kyunghyun Cho, and CIFAR Azrieli Global Scholar. 2019a. Bert has a mouth, and it must speak: Bert as a markov random field language model. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, page 30.

Dilin Wang, Chengyue Gong, and Qiang Liu. 2019b. Improving neural language modeling via adversarial training. In *International Conference on Machine Learning*, pages 6555–6565. PMLR.

Minghan Wang, Jiaxin Guo, Yuxia Wang, Daimeng Wei, Hengchao Shang, Yinglu Li, Chang Su, Yimeng Chen, Min Zhang, Shimin Tao, et al. 2022. Diformer: Directional transformer for neural machine translation. In *Proceedings of the 23rd Annual Conference of the European Association for Machine Translation*, pages 81–90.

Baosong Yang, Longyue Wang, Derek F Wong, Lidia S Chao, and Zhaopeng Tu. 2019a. Convolutional self-attention networks. 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 4040–4045.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019b. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018a. Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 2204–2213.

Ying Zhang, Tao Xiang, Timothy M Hospedales, and Huchuan Lu. 2018b. Deep mutual learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4320–4328.

Chunting Zhou, Jiatao Gu, and Graham Neubig. 2019. Understanding knowledge distillation in non-autoregressive machine translation. In *International Conference on Learning Representations*.

Long Zhou, Jiajun Zhang, and Chengqing Zong. 2020. Improving autoregressive nmt with non-autoregressive model. In *Proceedings of the First Workshop on Automatic Simultaneous Translation*, pages 24–29.