Machine Translation (2021) 35:595–609
https://doi.org/10.1007/s10590-021-09285-x

Enhanced encoder for non-autoregressive machine translation

Shuheng Wang[1] · Shumin Shi[2] · Heyan Huang[2]

Received: 26 October 2020 / Accepted: 21 October 2021 / Published online: 16 November 2021

Abstract

Non-autoregressive machine translation aims to speed up the decoding procedure by discarding the autoregressive model and generating the target words independently. Because non-autoregressive machine translation fails to exploit target-side information, the ability to accurately model source representations is critical. In this paper, we propose an approach to enhance the encoder's modeling ability by using a pre-trained BERT model as an extra encoder. With a different tokenization method, the BERT encoder and the Raw encoder can model the source input from different aspects. Furthermore, having a gate mechanism, the decoder can dynamically determine which representations contribute to the decoding process. Experimental results on three translation tasks show that our method can significantly improve the performance of non-autoregressive MT, and surpass the baseline non-autoregressive models. On the WMT14 EN→DE translation task, our method achieves 27.87 BLEU with a single decoding step. This is a comparable result with the baseline autoregressive Transformer model which obtains a score of 27.8 BLEU.

Keywords Machine translation · Pre-training language model · Non-autoregressive

✉ Shumin Shi
bjssm@bit.edu.cn

Shuheng Wang
wsh@njust.edu.cn

Heyan Huang
hhy63@bit.edu.cn

[1] School of Computer Science and Engineer, Nanjing University of Science and Technology, Nanjing, China

[2] School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China

1 Introduction

Neural machine translation (NMT) has achieved great improvements over the past few years (Bahdanau et al. 2014; Gehring et al. 2017; Vaswani et al. 2017). NMT models are generally based on the encoder-decoder framework. The encoder maps the input sentences to distributed representations, and the decoder generates the output sentences from these representations in a word-by-word manner. That is, when predicting the next word, the decoder feeds the previous output as input. This word-by-word generation manner of NMT models limits the application of parallel computing methods during the inference phase and leads to high translation latency which restricts the application scenarios of NMT.

With the introduction of parallel computing methods in the NMT training phase, the question of how to perform parallel decoding has attracted researchers' attention. To avoid the autoregressive property and produce the outputs in parallel, Gu et al. (2017) proposed a non-autoregressive translation (NAT) model. Drawing on the parallel computing power of Transformer (Vaswani et al. 2017), although a NAT model still uses the encoder-decoder architecture, the NAT model does not use the previously generated words as input which avoids the problems inherent in autoregressive models. The NAT model takes other signals (transferred from the source inputs (Gu et al. 2017; Guo et al. 2019), translation results from other systems (Lee et al. 2018; Guo et al. 2019), or latent variables (Kaiser et al. 2018)) as decoder inputs, which enables the independent and simultaneous generation of the target words and reduces translation latency.

In recent years, research on NAT has mainly focused on improving the decoder (Ghazvininejad et al. 2019; Gu et al. 2019; Shu et al. 2019; Sun et al. 2019; Shao et al. 2020). While those methods improve the performances of NAT model by modifying the inputs of the decoder or the training objective, the information that the decoder can rely on still comes from the encoder. The question, then, is whether the performance of the NAT model can continue to improve with a strong encoder. The effectiveness of an enhanced encoder has been demonstrated in NMT (Bastings et al. 2017; Imamura and Sumita 2019; Wei et al. 2019; Xiao et al. 2019), but the encoder used in NAT models is still the vanilla encoder from Transformer. To address this question, in this paper, we explored the effect of enhanced encoders in NAT models.

Given developments in pre-training methods (Devlin et al. 2018; Radford et al. 2019; Yang et al. 2019b), there has been some work on enhancing encoders in NMT (Clinchant et al. 2019; Yang et al. 2019a; Zhu et al. 2020), but those methods have not been investigated yet in NAT models. Drawing on the success of pre-training in NMT, in this paper, we exploited a straightforward method to enhance the encoder in NAT models with pre-training methods.

In this paper, based on pre-training, we proposed a BERT-based method to enhance the encoder in NAT models. Since the generation of target words is independent of the previously generated words, the decoder cannot acquire information from the previous target words, so all information including dependencies or word order comes from the encoder. Accordingly, we proposed a BERT-based encoder for

NAT models to enhance its modeling capability. Considering the performance degradation (Zhu et al. 2020) of directly using BERT to initialize the encoder of NMT models, we learned from a BERT-fused model(Zhu et al. 2020) and designed our enhanced encoder for NAT models, where the BERT representations are combined with the representations of the vanilla encoder. When the input sentences are fed into our model, the raw encoder and the BERT encoder simultaneously map the input sentences as distributed representations. As different sequence lengths are derived from different word segmentation rules, the representations from the raw encoder and the BERT encoder cannot be directly concatenated or added together. Therefore, we used an extra attention module in the decoder to fuse the BERT encoder representations with those from the raw encoder. In addition, a gate module is used to dynamically select information from the BERT and Raw encoder representations. Our enhanced encoder has the following advantages over other methods:

- It does not assume a NAT-specific model architecture and is suitable for any NAT model architecture with recent technology.
- It strengthens the ability to model the input sentence.
- It provides rich information to the decoder.

To evaluate the performance of our model, we compare it with previous work (Gu et al. 2017; Lee et al. 2018; Libovickỳ and Helcl 2018; Ghazvininejad et al. 2019; Gu et al. 2019) and conduct experiments on three benchmark tasks: WMT17 EN→ ZH, WMT14 EN↔DE and WMT16 EN↔RO. In addition, we conduct further analysis on IWSLT16 EN→DE. Experimental results and analysis show that our enhanced encoder surpasses the baseline NAT system by a significant margin in terms of translation quality without decelerating decoding speed. Furthermore, with distilled knowledge, our model can achieve comparable performance with an autoregressive MT baseline.

The main contributions of this paper include:

- We are first to exploit the influence of the encoder in NAT models.
- We use the BERT-based model to enhance the encoder of NAT models.
- We achieve a new state-of-the-art 27.87 BLEU on WMT'14 En→De for single-step non-autoregressive MT.

2 Related work

Gu et al. (2017) introduced a non-autoregressive Transformer model to reduce the translation latency of NMT, but this comes at the cost of translation quality. Instead of feeding the previous target tokens into the decoder, NAT models use other signals such as latent variables (Gu et al. 2017; Shu et al. 2019) as the input to the decoder. However, there is still a gap between the autoregressive and non-autoregressive models. In recent years, there has been some significant previous work on non-autoregressive models to improve the performance of NAT models. Lee et al. (2018) introduced an iterative decoding method for NAT models, which significantly improved the performance of NAT models. Inspired by the mask language model, Devlin et al. (2018), Ghazvininejad et al. (2019) and Gu et al. (2019) utilized the mask-based method to improve the decoder in NAT models. until now, research has mainly focused on the decoder in NAT models such as Shao et al. (2020). Recently, some research has started to migrate the successful experience on autoregressive NMT into non-autoregressive NMT such as Shao et al. (2019), Shao et al. (2020) and Zhou and Keung (2020). Following this idea and considering the absence of research on enhancing the encoder in NAT models, we utilize a BERT model as an extra encoder to enhance the modeling ability of the encoder.

There is some work on how to incorporate BERT into autoregressive NMT. Imamura and Sumita (2019) directly used a BERT model as an extra encoder to strengthen the representations generated by the encoder. Because of the limited improvement using BERT directly as encoder, Yang et al. (2019a) and Zhu et al. (2020) utilized the BERT model as an extra encoder to strengthen the encoding process. In this paper, we also use a BERT model as an extra encoder. Note too that different tokenizations have different effects on the performance of NMT models (Bahdanau et al. 2014; Sennrich et al. 2015). In this work, with the different tokenization methods, the BERT encoder and the raw encoder may model the different aspects of the input sentence.

3 Background

3.1 Autoregressive neural machine translation

At present, both autoregressive and non-autoregressive MT adapt the encoder-decoder framework. This framework has achieved great success in NMT (Bahdanau et al. 2014; Gehring et al. 2017; Vaswani et al. 2017). Compared with RNN-based models, CNN- and self attention-based models have a highly parallelized architecture and solve parallelization problems during training. However, during inference, as is the way in autoregressive models, the translation is still generated word-by-word.

Given an input sentence $X = \{x_1, x_2, \dots, x_n\}$ and the target sequence $Y = \{y_1, y_2, \dots, y_n\}$, an autoregressive translation model models the conditional probability as in (1):

$$P(Y|X, \theta) = \prod_{t-1}^{T} p(y_t|y_{<t}, X, \theta), \tag{1}$$

where θ are the parameters of the autoregressive translation models and $y_{<t}$ denote the previously generated words. During inference, with θ and the previously generated words $y_{<t}$, the autoregressive model generates the current word. During training, θ are learned by maximizing the log-likelihood of the training data, as in (2) and (3):

$$\theta = arg\,\underset{\theta}{max}(L(\theta)) \tag{2}$$

$$L(\theta) = \sum_{n=1}^{N} \sum_{t=1}^{T} log(p(y_t^n | y_{<t}^n, X^n, \theta)), \tag{3}$$

where N denotes the pairs of sentences in the training set.

According to these formulae, the unique characteristic of autoregressive models is that it requires the the previously generated words in the decoding procedure. Due to this unique characteristic, parallelization is not possible and the decoding is limited, which restricts the application of the autoregressive model.

However, at present, the performance of non-autoregressive models falls far behind autoregressive models. In this work, we attempt to improve the performance of non-autoregressive models using pre-trained models.

3.2 Pre-training for autoregressive NMT

Pre-training has been used in natural language processing (NLP) (Mikolov et al. 2013; Dai and Le 2015) for years. At the beginning, because improvements were not comparable with pre-training in computer vision, the scope of pre-training in NLP was still relatively small. Recently, with increases in both computing resources and available data, pre-training techniques have received increasing attention from NLP researchers. The pre-training approach has refreshed state-of-the-art results on some tasks (Devlin et al. 2018; Yang et al. 2019b). Pre-training in MT has seen some research (Yang et al. 2019a; Zhu et al. 2020), but .this work has been conducted only for autoregressive MT. However, in this work, we use the pre-training language model as an extra encoder to model the sentence in the source language for non-autoregressive MT.

3.3 Non-autoregressive NMT

The aim of non-autoregressive NMT proposed by Gu et al. (2017) is to accelerate the decoding speed. Compared to autoregressive models, non-autoregressive NMT can simultaneously and independently generate the words in the translation. Compared to conditional probability in autoregressive MT, the translation the probability from X to Y in non-autoregressive MT is modeled as in (4):

$$P(Y|X) = \prod_{t=1}^{T} p(y_t|X, \theta) \tag{4}$$

Given a training set $D = \{X^N, Y^N\}$ with N sentence pairs, the training objective of non-autoregressive MT is to maximize the log-likelihood of the training data, as in (5):

$$\theta = argmax_{\theta}(L(\theta)) \tag{5}$$

in which $L(\theta)$ is computed as in (6):

$$L(\theta) = \sum_{n=1}^{N} \sum_{t=1}^{T} \log(p(y_t^n | X^n, \theta)) \tag{6}$$

Eq. (1) shows that when non-autoregressive MT generates the target words, it does not need access to the previously generated words. During inference, the target words can be generated by taking the word with the maximum probability in each time step, as in (7):

$$\hat{y}_t = \text{argmax}_{y_t}(p(y_t | X, \theta)) \tag{7}$$

No longer needing the previous target words at each time step, non-autoregressive MT can be computed in parallel both in the training and in the decoding phases. However, there are some weaknesses in non-autoregressive MT. For example, non-autoregressive MT still has a great gap in translation quality compared to autoregressive MT and tends to generate repetitive words or wrong words. In this work, we introduce a stronger encoder for the NAT model to improve its performance. Inspired by previous work using pre-training methods in NMT, in this work, we demonstrate how pre-training can be incorporated into NAT models.

4 Approaches

In this section, we first define the necessary notation, and then introduce our proposed enhanced encoder model.

Notation Let X and Y denote the input sentences and target sentences, respectively. We denote the raw encoder and BERT encoder as Enc_R, Enc_B, respectively, and we let *attn* be the attention module.

Since our proposed model mainly focuses on the encoder of NAT models, and does not restrict the decoder, in this section, for ease of description, we use the architecture of Ghazvininejad et al. (2019) to describe our model. An illustration of our model is shown in Figure 1.

Firstly, given an input $x \in X$, the BERT-encoder and Raw-encoder encode it into representations $H_B = Enc_B(x)$ and $H_R = Enc_R(x)$, respectively. H_B and H_R are the output of the last layer in the BERT-encoder and Raw-encoder, respectively.

Then let S^l denotes the hidden state of l-th layer in the decoder, where we have (8)–(10):

$$\hat{S}^l = \text{attn}_s(S^{l-1}, S^{l-1}, S^{l-1}) \tag{8}$$

$$S_B^l = \text{attn}_B(\hat{S}^l, H_B, H_B) \tag{9}$$

$$S_R^l = \text{attn}_R(\hat{S}^l, H_R, H_R) \tag{10}$$

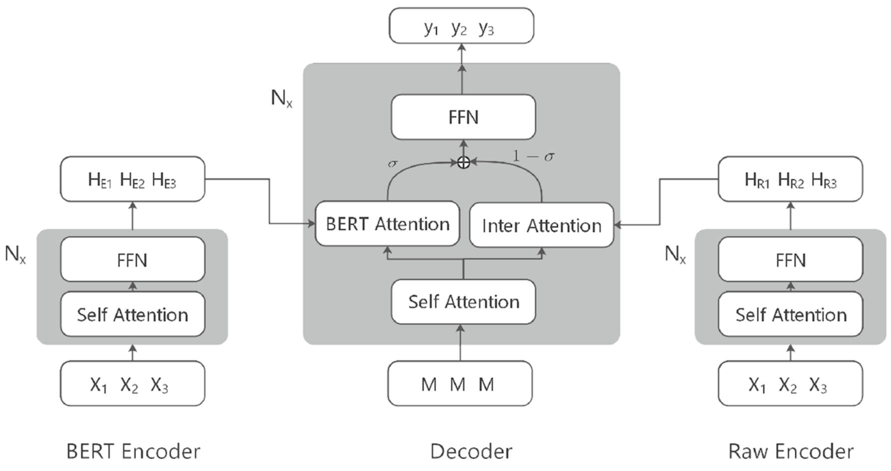

Fig. 1 The overall enhanced NAT encoder. The BERT encoder is an extra encoder, and the Raw Encoder is the vanilla Transformer encoder. The decoder can dynamically control the information flowing from the BERT encoder and Raw Encoder. "M" denotes "MASK" in MASK-Predict

S_B and S_R denote the information learned from H_B and H_R, respectively. $attn_s$, $attn_B$ and $attn_R$ represent the self-attention module, BERT-encoder-decoder attention module and encoder-decoder module, respectively. We use a gate module to make the decoder dynamically combine the information from BERT and the Raw encoder, and control the information flowing to the next layer, as in (11).

$$g = \sigma(W[H_R : H_B]) \tag{11}$$

$$S^l = g \times H_R + (1 - g) \times H_B \tag{12}$$

S^l in (12) is the output of the l-th layer in the decoder. With a stack decoder, we can derive the final hidden state of the decoder S. Finally, with *softmax*, we can obtain the conditional probability in (13):

$$P(y|x) = \text{softmax}(WS) \tag{13}$$

In our proposed model, BERT is used as an auxiliary encoder, and generates different representations of the input. With a different tokenization model, the BERT-encoder and Raw-encoder can learn to express the input from different angles, and the decoder can dynamically utilize the representations from different angles via the gate module.

5 Experimental settings

5.1 Datasets

We use several commonly adopted benchmark datasets to evaluate the performance of our proposed methods: WMT17 EN→ZH, WMT14 EN↔DE, and WMT16 EN↔RO. We also add experiments and analysis on IWSLT16 EN→DE. These datasets consist of 20*M*, 4.5*M*, 610*k*, and 196*k* sentence pairs, respectively. For IWSLT16 EN→DE, we use the test2013 dataset for validation purposes. For WMT14 EN↔DE, we use newstest2013 for our validation set and newstest2014 for our test set. For WMT16 EN↔RO, we use newsdev2016 and newstest2014 as our development and test sets. For WMT17 EN→ZH, we use newsdev2017 and newstest2017 as our validation and test sets, respectively. For all tasks, we use the script from Moses (Koehn et al. 2007) as our tokenization tools, and we segment each word into subword units with BPE (Sennrich et al. 2015). The vocabulary size for all tasks is 40*k* and is shared for source and target languages. We use BLEU (Papineni et al. 2002) as our evaluation metric.

5.2 Baselines

We use the Transformer model (Vaswani et al. 2017) as our autoregressive baseline . We choose several recently proposed NAT methods as our NAT baselines:

- **NAT** (Gu et al. 2017) is the first non-autoregressive model.
- **I-NAT** (Lee et al. 2018) is the first model that utilizes iterative refinement to refine the translations.
- **Mask-Predict** (Ghazvininejad et al. 2019) introduce a masked language model to train the NAT model.
- **LevT** (Gu et al. 2019) utilizes three decoders to determine which operation (Deletion, Insertion, or Filling) should be done.
- **CTC** (Libovickỳ and Helcl 2018) utilizes CTC model to learn the alignment between source and target sentences.
- **SMART** (Ghazvininejad et al. 2020b) adapts semi-autoregressive training to improve the non-autoregressive model.
- **NAT-REG** (Wang et al. 2019) introduces explicit regularization to reduce repetitive words in the NAT model.
- **BoN-NAT** (Shao et al. 2020) introduce bag-of-ngram loss to improve the performance of the NAT model.
- **Hint-NAT** (Li et al. 2019) distil the output of the attention module by an autoregressive model to improve the performance of the NAT model.
- **FlowSeq** (Ma et al. 2019) models the generation flow as latent variables.
- **CRF-NAT** (Sun et al. 2019) introduce an approximate CRF model to model the structure of the target sentence.
- **AXE-NAT** (Ghazvininejad et al. 2020a) introduce a new loss function to align target words with source words.

- **KERMIT** (Chan et al. 2019) is an insertion-based generative model.
- **Imputer** (Saharia et al. 2020) model the alignments as latent variables to improve the performance of the NAT model.

5.3 Model configurations

We follow the standard hyperparameters for Transformer in the base configurations (Vaswani et al. 2017): 6 layers stack, 8 attention heads per layer, 512 model dimensions, and 2048 hidden dimensions. To effectively learn the representation of the source language, for IWSLT16 EN→DE, WMT14 EN→DE, WNT17 EN→ ZH, and WMT16 EN→RO, we use bert-base-uncased[1] to initialize our enhanced encoder. For WMT14 DE→EN, we use bert-base-german-cased[2] as initialization. For WMT16 RO→ EN, we initialize our enhanced encoder with bert-base-multilingual-uncased.[3] For regularization, we use 0.3 dropout, 0.01 L_2 weight decay, and smoothed cross validation loss with $\epsilon = 0.1$. We follow Ghazvininejad et al. (2019) and train our model with batches of $128k$ tokens using Adam (Kingma and Ba 2014). We train all the models for $300k$ steps and average the 5 best checkpoints to create the final model.

5.4 Sequence-level distillation

According to previous work (Gu et al. 2017; Zhou et al. 2019), in non-autoregressive MT, sequence-level knowledge distillation (Kim and Rush 2016) is critical for NAT models. In this work, we follow this previous work and train all our models based on the translations generated by an autoregressive model. We then discuss the influence of distillation on our model.

5.5 Model architecture

Because our method is enhancing the encoder in NAT models, it does not restrict the architecture of the decoder, so it can simply be migrated to any recent method. In this paper, we implement our method on Levenshtein Transformer (LevT) (Gu et al. 2019). That is, we use the BERT-encoder and Raw-encoder of Transformer as our encoder, and we use the decoder[4] of Levenshtein Transformer as our decoder. To dynamically control the information flowing to the next layer, we add a gate module to the decoder.

[1] https://s3.amazonaws.com/models.huggingface.co/bert/bert-base-uncased.tar.gz

[2] https://int-deepset-models-bert.s3.eu-central-1.amazonaws.com/pytorch/bert-base-german-cased.tar.gz

[3] https://s3.amazonaws.com/models.huggingface.co/bert/bert-base-multilingual-uncased.tar.gz

[4] Levenshtein Transformer consists of three decoders, and the parameters of those decoders are shared. During inference, the first decoder decides which word should be deleted in the input target sentence, and the second decoder predicts the number of tokens to be inserted at every consecutive position pair and inserts the placeholders at the corresponding positions. Finally, the third decoder fills the tokens replacing the placeholders.

Table 1 Performance of various single-step decoding models. Our enhanced encoder LevT is able to outperform all prior single decoding models

Methods	Iters	WMT14		WMT16		WMT17
		EN→DE	DE→EN	EN→RO	RO→EN	EN→ZH
Autoregressive transformer						
Base transformer	*N*	27.8	31.2	34.3	34.0	34.74
Non-Autoregressive						
I-NAT	1	13.9	16.7	24.5	25.7	–
NAT	1	17.7	21.5	27.3	29.1	–
CTC	1	17.7	19.8	19.9	24.7	–
SMART	1	18.6	23.8	–	–	24.15
NAT-REG	1	20.7	24.8	–	–	–
BoN-NAT	1	20.9	24.6	28.3	29.3	–
Hint-NAT	1	21.1	25.2	–	–	–
FlowSeq	1	21.5	26.2	29.3	30.4	–
CRF-NAT	1	23.4	27.2	-	–	–
AXE-NAT	1	23.5	27.9	30.8	31.5	30.88
Imputer	1	25.8	28.4	32.3	31.7	–
Mask-Predict	1	18.0	19.3	27.3	28.2	24.23
Our work						
Enhanced encoder LevT	1	27.87	29.57	32.96	32.65	32.56

The results are obtained from the original papers

6 Results and analysis

6.1 Single step decoding

Firstly, we evaluate the performance of our method with single-step decoding, as shown in Table 1. We compare our method with other non-autoregressive single decoding models. Our enhanced encoder LevT achieves 27.87 BLEU for WMT14 EN→DE. Our method achieves an almost 2.0 BLEU point improvement over Imputer which is the state-of-the-art model for single-step non-autoregressive MT. In addition, our method has also improved the performance of NAT to varying degrees for WMT14 DE→EN, WMT16 EN→RO and WMT17 EN→ZH.

6.2 Iterative decoding

We now analyze the performance of enhanced encoder LevT with more decoding iterations. We compare the performance of enhanced encoder LevT with other non-autoregressive models ranging from models requiring logarithmic to a constant number of decoding iterations. The results of our method are summarized in Table 2.

Table 2 Performance of various autoregressive and non-autoregressive models

Methods	Iters	WMT14		WMT16		WMT17
		EN→DE	DE→EN	EN→RO	RO→EN	EN→ZH
Autoregressive transformer						
Base transformer	*N*	27.8	31.2	34.3	34.0	34.31
Non-autoregressive						
KERMIT	$\approx log_2 n$	27.8	30.7	–	–	–
I-NAT	10	21.6	25.5	29.3	30.2	–
SMART	4	27.0	30.9	–	–	33.37
	10	27.7	31.3	–	–	34.06
Mask-Predict	4	25.9	29.9	32.5	33.2	32.63
	10	27.0	30.5	33.1	33.3	33.19
Imputer	2	27.5	30.2	33.7	33.4	–
	4	28.0	31.0	34.3	34.0	–
Our work						
LevT	2(avg)	27.27	–	–	33.26	–
Enhanced encoder LevT	2	27.95	30.31	33.73	33.52	33.74
	4	28.35	31.10	34.51	34.01	34.23

Our enhanced encoder LevT can achieve comparable results with the autoregressive Transformer baseline with just 4 decoding steps. The results are obtained from the original papers

Our enhanced encoder LevT achieves 27.95 BLEU on WMT14 EN→DE with only 2 iterative decoding steps, which is comparable with the autoregressive model Transformer. With 4 iterative decoding steps, our method achieves 28.35 BLEU, slightly outperforming the autoregressive Transformer score of 27.8 BLEU. On WMT14 DE→EN, we achieve 31.1 BLEU, which is on a par with the autoregressive Transformer. On WMT16 RO→EN, with 2 iterative decoding steps, our method also outperforms the standard LevT and achieves similar results as autoregressive Transformer in 4 iterative decoding steps. On WMT17 EN→ZH, our model and SMART achieve similar levels of performance. We think that the sequence-level distillation by the autoregressive model limits the improvement in performance.

6.3 Impact of decoding speed

Because of the introduction of the BERT encoder and the gate module, decoding speed may be adversely affected. In this section, we compare the decoding speed of our method with the standard LevT. For both models, we decode batches of 10 sentences on 1 Nvidia 1080Ti GPU. We measure the wall time from when the model and data have been loaded until the last example has been translated, and calculate the decoding speed to assess the average speed performance trade-off.

The speeds of our method are shown in Table 3. We can observe that there is a decline in speed of our enhanced encoder LevT. However, compared with Transformer, our method can still obtain a 3.12× speedup.

Table 3 Translation latency on WMT14 EN→DE

Methods	Speedup	Latency
Transformer	1.00×	607 ms
LevT	3.73×	162 ms
Enhance encoder LevT	3.12×	195 ms

Table 4 IWSLT16 EnDe BLEU comparison on the impact of distillation

Method	Iterations	Original	Distillation
Transformer	N	28.98	–
Enhanced encoder LevT	1	28.04	28.83
	2	28.84	29.54
	4	29.23	29.76

Table 5 An example of IWSLT16 EN→DE translation

Source	I want to take shipping containers and turn them into healthy cafes
Reference	ich will Schiffscontainer nehmen und sie in gesunde Cafés verwandeln
LevT	ich möchte *Schiffcontainer* nehmen und sie in gesunde *Wasserés* verwandeln
Ours	ich möchte *Schiffscontainer* nehmen und sie in gesunde *Cafés* verwandeln

6.4 Impact of distillation

We analyze the impact of distillation on our method by comparing the performance on the original training data (original data) and training data generated by a base Transformer teacher (distilled data) on IWSLT16 EN→DE.

From Table 4, we can observe that in all cases, the model with distilled data outperforms the model with the original data. As the number of decoding steps increases, the gap between the model with original and distilled data decreases, which is identical to the observation of citetzhou2019understanding. Similarly, on WMT14 with distilled data, our method obtains a comparable result to autoregressive Transformer with 28.98 BLEU. Note that our model outperforms Transformer with only 2 decoding steps.

6.5 Case study

We show an example from the IWSLT16 EN→DE validation set in Table 5. For the words "shipping containers", LevT generates a wrong word "Schiffcontainer". In contrast, our model with different encoders generates the correct word "Schiffscontainer". While LevT misunderstands the meaning of the word "cafes", our model understands it correctly.

Table 6 The comparison of different encoders on the IWSLT16 EN→DE validation set

Model	Encoder type	BLEU
LevT	Raw Encoder	27.05
LevT	Bert Encoder	21.08
Ours	Raw Encoder + Bert Encoder	28.83

6.6 Ablation analysis

To evaluate the effect of different encoders, we conduct an ablation analysis on the IWSLT16 EN→DE validation set, and give the results in Table 6. We can see that only using BERT as the encoder can decrease the performance of the NAT model, which is consistent with the conclusion of Zhu et al. (2020). In contrast, using BERT as an additional encoder in our model can significantly improve the performance of the NAT model.

7 Conclusion

In this paper, we utilize a BERT model as an extra encoder to strengthen the ability of the encoder in non-autoregressive MT. Unlike most of the previous work which focused mainly on the decoder in NAT models, our method mainly focuses on enhancing the encoder. With the addition of a gate module, the decoder can dynamically select representations of the input sentences from the Raw and BERT encoders. Furthermore, with quite a simple architecture, our method can easily be incorporated seamlessly into recent work. Our enhanced encoder LevT achieves 27.87 BLEU with a single generation step, which is comparable with the Transformer baseline on the WMT14 EN→DE task.

Acknowledgements We thank the reviewers for their careful reviewing and constructive opinions. We thank Prof. Andy Way for his linguistic assistance and careful proofreading during the revision of this paper. This work is supported by the National Natural Science Foundation of China (Nos. 61732005, 61671064).

References

Bahdanau D, Cho K, Bengio Y (2014) Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473

Bastings J, Titov I, Aziz W, Marcheggiani D, Sima'an K (2017) Graph convolutional encoders for syntax-aware neural machine translation. arXiv preprint arXiv:1704.04675

Chan W, Kitaev N, Guu K, Stern M, Uszkoreit J (2019) Kermit: generative insertion-based modeling for sequences. arXiv preprint arXiv:1906.01604

Clinchant S, Jung KW, Nikoulina V (2019) On the use of bert for neural machine translation. arXiv preprint arXiv:1909.12744

Dai AM, Le QV (2015) Semi-supervised sequence learning. Advances in neural information processing systems. Montréal, Canada, pp 3079–3087

Devlin, J, Chang M-W, Lee K, Toutanova K (2018) Bert: pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805

Gehring J, Auli M, Grangier D, Yarats D, Dauphin YN (2017) Convolutional sequence to sequence learning. In: Proceedings of the 34th international conference on machine learning, vol. 70, pp. 1243–1252, Sydney, Australia

Ghazvininejad M, Levy O, Liu Y, Zettlemoyer L (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), pp. 6114–6123, Hong Kong, China

Ghazvininejad M, Karpukhin V, Zettlemoyer L, Levy O (2020a) Aligned cross entropy for non-autoregressive machine translation. arXiv preprint arXiv:2004.01655

Ghazvininejad M, Karpukhin V, Zettlemoyer L, Levy O (2020b) Semi-autoregressive training improves mask-predict decoding. arXiv preprint arXiv:2001.08785

Gu J, Bradbury J, Xiong C, Li VO, Socher R (2017) Non-autoregressive neural machine translation. arXiv preprint arXiv:1711.02281

Gu J, Wang C, Zhao J (2019) Levenshtein transformer. Advances in neural information processing systems. Vancouver, BC, Canada, pp 11179–11189

Guo J, Tan X, He D, Qin T, Xu L, Liu T-Y (2019) Non-autoregressive neural machine translation with enhanced decoder input. In: Proceedings of the AAAI conference on artificial intelligence, vol 33, pp 3723–3730, Honolulu, Hawaii, USA

Imamura K, Sumita E (2019) Recycling a pre-trained bert encoder for neural machine translation. In: Proceedings of the 3rd workshop on neural generation and translation, pp 23–31, Hong Kong, China

Roy Kaiser A, Vaswani A, Parmar N, Bengio S, Uszkoreit J, Shazeer N (2018) Fast decoding in sequence models using discrete latent variables. arXiv preprint arXiv:1803.03382

Kim Y, Rush AM (2016) Sequence-level knowledge distillation. arXiv preprint arXiv:1606.07947

Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980

Koehn P, Hoang H, Birch A, Callison-Burch C, Federico M, Bertoldi N, Cowan B, Shen W, Moran C, Zens R et al (2007) Open source toolkit for statistical machine translation. In: Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions, pp 177–180, Prague, Czech Republic

Lee J, Mansimov E, Cho K (2018) Deterministic non-autoregressive neural sequence modeling by iterative refinement. arXiv preprint arXiv:1802.06901

Li Z, Lin Z, He D, Tian F, Qin T, Wang L, Liu T-Y (2019) Hint-based training for non-autoregressive machine translation. arXiv preprint arXiv:1909.06708

Libovickỳ J, Helcl J (2018) End-to-end non-autoregressive neural machine translation with connectionist temporal classification. arXiv preprint arXiv:1811.04719

Ma, X Zhou, C Li X, Neubig G, Hovy E (2019) Flowseq: non-autoregressive conditional sequence generation with generative flow. arXiv preprint arXiv:1909.02480

Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J (2013) Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems. Harrahs and Harveys, Lake Tahoe, pp 3111–3119

Papineni K, Roukos S, Ward T, Zhu WJ (2002) Bleu: a method for automatic evaluation of machine translation. In: Proceedings of the 40th annual meeting of the Association for Computational Linguistics, pp 311–318, Philadelphia, Pennsylvania, USA

Radford A, Wu J, Child R, Luan D, Amodei D, Sutskever I (2019) Language models are unsupervised multitask learners. OpenAI Blog 1(8):9

Saharia C, Chan W, Saxena S, Norouzi M (2020) Non-autoregressive machine translation with latent alignments. arXiv preprint arXiv:2004.07437

Sennrich R, Haddow B, Birch A (2015) Neural machine translation of rare words with subword units. arXiv preprint arXiv:1508.07909

Shao C, Feng Y, Zhang J, Meng F, Chen X, Zhou J (2019) Retrieving sequential information for non-autoregressive neural machine translation. arXiv preprint arXiv:1906.09444

Shao C, Zhang J, Feng Y, Meng F, Zhou J (2020) Minimizing the bag-of-ngrams difference for non-autoregressive neural machine translation. In: Proceedings of the AAAI conference on artificial intelligence, vol 34, pp 198–205, New York, USA

Shu R, Lee J, Nakayama H, Cho K (2019) Latent-variable non-autoregressive neural machine translation with deterministic inference using a delta posterior. arXiv preprint arXiv:1908.07181

Sun Z, Li Z, Wang H, He D, Lin Z, Deng Z (2019) Fast structured decoding for sequence models. Advances in neural information processing systems. Vancouver, BC, Canada, pp 3011–3020

Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. Advances in neural information processing systems. Long Beach Convention Center, Long Beach, pp 5998–6008

Wang Y, Tian F, He D, Qin T, Zhai C, Liu T-Y (2019) Non-autoregressive machine translation with auxiliary regularization. In: Proceedings of the AAAI conference on artificial intelligence, vol 33, pp 5377–5384, Honolulu, Hawaii, USA

Wei X, Hu Y, Xing L (2019) Gated self-attentive encoder for neural machine translation. In: International conference on knowledge science, engineering and management, pp. 655–666, Athens, Greece, 2019. Springer

Xiao F, Li J, Zhao H, Wang R, Chen K (2019) Lattice-based transformer encoder for neural machine translation. arXiv preprint arXiv:1906.01282

Yang J, Wang M, Zhou H, Zhao C, Yu Y, Zhang W, Li L (2019a) Towards making the most of bert in neural machine translation. arXiv preprint arXiv:1908.05672

Yang Z, Dai Z, Yang Y, Carbonell J, Salakhutdinov RR, Le QV (2019b) Xlnet: generalized autoregressive pretraining for language understanding. Advances in neural information processing systems. Vancouver, BC, Canada, pp 5754–5764

Zhou C, Neubig G, Gu J (2019) Understanding knowledge distillation in non-autoregressive machine translation. arXiv preprint arXiv:1911.02727

Zhou J, Keung P (2020) Improving non-autoregressive neural machine translation with monolingual data. arXiv preprint arXiv:2005.00932

Zhu J, Xia Y, Wu L, He D, Qin T, Zhou W, Li H, Liu T-Y (2020) Incorporating bert into neural machine translation. arXiv preprint arXiv:2002.06823