Improving Non-Autoregressive Translation with Reordering and Hybrid CTC/Attention Decoding

Shuheng Wang and Dongbai Li
School of Computer and Software, Nanyang Institute of Technology
shwang@dl4mt.cn db.li@foxmail.com

Shumin Shi*
School of Computer Science and Technology, Beijing Institute of Technology
bjssm@bit.edu.cn

Abstract

Connectionist Temporal Classification (CTC) based non-autoregressive translation (NAT) has attracted increasing attention in recent years. Despite the fact that the CTC loss mechanism is capable of handling monotonic latent alignments, CTC remains limited in handling non-monotonic alignments, which is a common challenge in machine translation. In this paper, we propose a reordering-based framework for NAT that utilizes the CTC loss function. Our approach employs an encoder-only model for NAT and integrates a reordering module into the encoder to effectively address the limitations of CTC under non-monotonic alignments. Furthermore, we present a hybrid CTC/attention decoding mechanism that takes advantage of the strengths of both models. We evaluate our model on benchmark datasets that include WMT14 English → German, WMT16 English → Romanian, and IWSLT14 German → English. Experimental results show consistent improvements over strong NAT baselines.

1 Introduction

Non-Autoregressive Translation (NAT), which generates target words simultaneously, was introduced by Gu et al. (2017) and substantially improves decoding speed over autoregressive neural machine translation (NMT) (Vaswani et al., 2017; Bahdanau et al., 2014; Gehring et al., 2017). However, NAT's benefit often comes at a cost of reduced translation quality, resulting from a mismatch in training objectives. NAT models are trained using cross-entropy loss, which demands model output precisely match target sentences and penalizes even small variances in word position. Given that a single sentence can have multiple possible translations (Gu et al., 2017), it can be challenging to achieve the precise alignment that cross-entropy loss necessitates. Consequently, there is a fundamental conflict between cross-entropy loss and NAT models (Shao and Feng, 2022).

The challenge of achieving high translation quality in NAT models trained with the cross-entropy loss has resulted in significant research toward identifying more effective training objectives. For example, Wang et al. (2019) proposed an auxiliary regularization to limit the number of repetitions in the generated target sentences. Meanwhile, Shao et al. (2019, 2020); Xie et al. (2020) introduced new training objectives capable of enabling NAT models to learn global or partial sentence-level information. Researchers such as Ghazvininejad et al. (2020) and Du et al. (2021) attempted to ease the alignment constraint of the cross-entropy loss for NAT models, which led to significant performance gains. Other approaches used directed acyclic graphs, as demonstrated by Sun et al. (2019) and Huang et al. (2022b), to aid learning target word dependencies. Furthermore, researchers have explored the use of latent training objectives that relax the alignment requirements by marginalizing all possible monotonic latent alignments using the Connectionist Temporal Classification (CTC) loss for NAT models (Higuchi et al., 2020; Chan et al., 2020; Shao and Feng, 2022), which have shown promising improvements in performance and attracted considerable attention in recent years.

At present, most NAT models retain the encoder-decoder architecture, with an encoder that maps input sentences into hidden representations and a decoder that generates target words based on both predicted words and source representations. The primary difference between NAT and NMT models is that the NAT decoder generates target words all at once without conditioning on previously generated target tokens. To align with NMT models, NAT models use specific tokens as placeholders or copy embeddings from source sentences as decoder input. However, these inputs may not provide effective information and introduce extrane-

*Corresponding author

ous noise that negatively impacts the NAT model performance. As a remedy, we propose an encoder-only model for NAT models in this study. However, the CTC loss function of the latent training objective only models monotonic alignments, which is problematic for encoder-only NAT models.

The CTC loss function assumes that there is a monotonic alignment between source-side representations and target tokens. This assumption, however, makes it unsuitable for translation tasks that require significant word reordering. An example of this limitation is illustrated in Figure 1a, where a CTC-based translation model fails to translate the target sentence "I would argue" correctly, producing "I argue" instead. The limitation occurs because the CTC loss function is not designed to accommodate non-monotonic alignments between the output and the intended target, resulting in an inaccurate penalty on the model and an under-translated output. The nature of non-autoregressive translation (NAT), whereby the target sentence is generated exclusively based on the source input, raises particular challenges when incorporating an encoder-only model. To address this issue, we introduce a reorder-based module to improve the performance of our encoder-only model. This module works by converting the source sentence into a new sentence that conforms to the word order of the target sentence, as shown in Figure 1b. By doing so, we encourage more monotonic alignments between the source and target words and in line with the requirements of the CTC loss function.

This paper presents a novel encoder-only model for non-autoregressive translation, built on an encoder-only architecture with stacked attention blocks, and optimized using the CTC loss function. To enhance the attention model's capacity for generating target words, we use an auxiliary attention objective to complement CTC training. For decoding, we propose a joint decoding approach that uses attention-based and CTC scores in a rescoring beam search algorithm. The joint decoding approach eliminates irregular alignments and enhances translation quality.

To evaluate the effectiveness of our approach, we conducted a series of experiments on standard benchmark translation tasks, including WMT14 EN $\rightarrow$ DE, WMT16 EN $\rightarrow$ RO, and IWSLT14 DE $\rightarrow$ EN. Our results demonstrate that our proposed method significantly improves NAT models' translation quality, particularly for the IWSLT14 DE $\rightarrow$ EN task, where our approach improves BLEU by 5.0 points over the strong baseline NAT-CTC model. Additionally, we conducted a detailed analysis of each module's influence in our method, focusing primarily on the IWSLT14 DE $\rightarrow$ EN task. The results show that each module proposed in our approach results in significant performance gains for NAT models.

2 Background

2.1 Non-Autoregressive Translation

Traditional NMT systems predict target words one-by-one from left to right in an autoregressive manner (Bahdanau et al., 2014). This process involves several decoding steps based on the length of the target sentence, limiting the decoding speed of the model. To address this issue, Gu et al. (2017) introduced NAT models, which generate the entire target sentence simultaneously and achieve significant decoding speedup. NAT models decouple the dependencies between target words by factorizing the probability of target words as a joint probability instead of a chain conditional probability as in NMT models:

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

where N represents the number of words in the target sentence, X represents the encoder's mapped hidden representations of the source words, $Y = \{y_1, ...y_N\}$ represents the target sentence, and $p(y_t|X)$ denotes the translation probability at position t.

NAT models typically employ the cross-entropy loss function during training to optimize the entire model. This loss function aligns the model outputs explicitly with the target words:

$$\mathcal{L}_{CE} = -\sum_{t=1}^{N} \log p(y_t|X) \tag{2}$$

The cross-entropy loss function is a reliable method to evaluate the accuracy of a model's predictions. It imposes a penalty for each inaccurately predicted word, even if only the slightest difference exists between the target and predicted sequences. Additionally, this loss function penalizes any word wrongly placed in the output sequence (Ghazvininejad et al., 2020). The NMT models are capable of learning to avoid penalties associated with predicting words out of order, as they generate



(a) CTC Loss

(b) Source Reordering for CTC

Figure 1: Illustration of the monotonic alignment assumption of CTC: (a) CTC loss where global word reordering leads to non-monotonic alignments. (b) Reorder Source for CTC Loss where we reorder the words in the source sentence for monotonic alignment.

words conditioned on the sentence prefix. Conversely, NAT models lack the precise knowledge of the sentence prefix and may encounter challenges in avoiding these penalties. To alleviate this issue, the Connectionist Temporal Classification (CTC) loss was introduced for the NAT models (Graves et al., 2006; Libovický and Helcl, 2018; Saharia et al., 2020).

2.2 NAT with CTC Loss Function

In contrast to the cross-entropy loss function, which models the joint probabilities of target words, the CTC loss function models the alignments between the source and target words. Consider a source sentence X and a target sentence $Y = \{y_1, ...y_N\}$, where $y_i \in \mathcal{V}$ represents a word in the target sentence, the CTC loss function includes a latent alignment α between X and Y, where $\alpha_i \in \mathcal{V}^+$ and $\mathcal{V}^+ = \mathcal{V} \cup \{\texttt{<blank>}\}$, $\mathcal{V}$ being the target vocabulary. The symbol <blank> is the CTC blank token, which is not a word of $\mathcal{V}$ and only serves to align X and Y. In modeling the alignment α, the CTC loss function assumes conditional independence between token frame predictions:

$$P(\alpha|X) = \prod_i p(\alpha_i|X), \tag{3}$$

where $p(\alpha_i|X)$ is modeled by a non-autoregressive Transformer in NAT models. The CTC loss learns the alignment α by maximizing the log-likelihood:

$$\mathcal{L}_{CTC} = -\log \sum_{\alpha \in \Gamma(Y)} P(\alpha|X) \tag{4}$$

The CTC loss, presented in Eq 4, enables the learning of all possible alignments that are consistent with the target sentence Y. However, despite the improved performance of NAT models with the CTC loss, CTC-based NAT models still struggle with non-monotonic alignment patterns and this issue is even more severe in encoder-only models.

3 Approach

Previous studies have indicated the effectiveness of incorporating local information in the NAT models, as shown in prior work (Wang et al., 2023; Ding et al., 2020). In this study, we intend to integrate the local information approach into our own NAT model. Additionally, our aim is to overcome the limitations of the Connectionist Temporal Classification (CTC) loss function by introducing an upsample layer that addresses the mismatch between the length of the source sentence and the target sentence. A detailed diagram of our model's methodology is provided in Figure 2, and the remainder of this section describes our approach in detail.

3.1 Global Information Layer

To extract global information from any word in the sentence, we employ a self-attention network in our work, which is inspired by the structure of Vaswani et al. (2017). In the global information layer, we utilize a multi-head attention mechanism to extract the relevant information as follows:

$$\begin{aligned} m_l = \text{Concat}(&\text{SelfAttn}_1(h_{l-1}), \\ &\text{SelfAttn}_2(h_{l-1}), \ldots, \text{SelfAttn}_k(h_{l-1})), \end{aligned} \tag{5}$$

Here, h_{l-1} refers to the output of the previous layer, m_l denotes the concatenated multi-head attention output, and k represents the number of attention heads. We then combine the output of the multi-head attention with the previous layer's output using a residual connection.

$$\tilde{h}_l = \text{LayerNorm}(m_l + h_{l-1}), \tag{6}$$

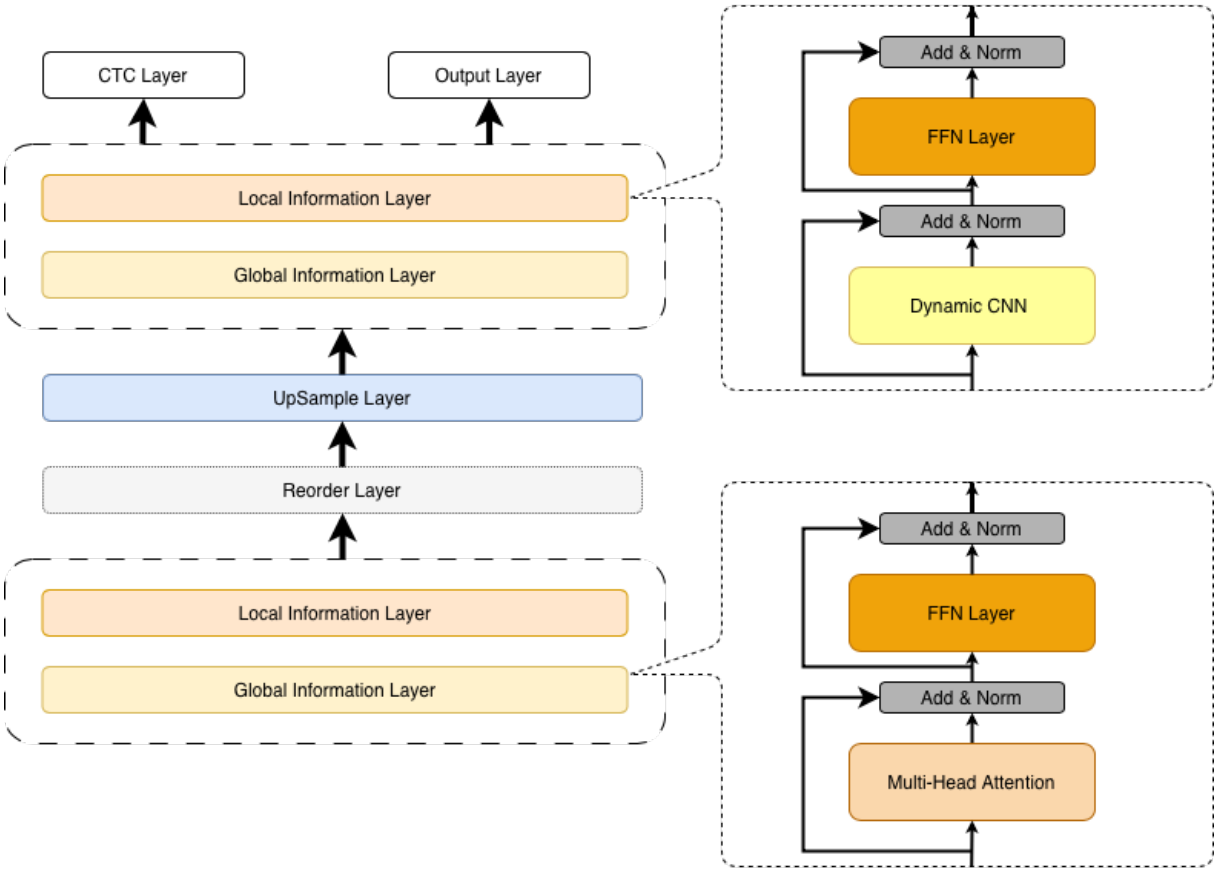


Figure 2: Overview of the proposed model. The reorder layer calculates reorder loss. The upsample layer matches CTC loss length requirements.

where $\tilde{h}_l$ is the normalized residual output. A feed-forward neural network follows, which consists of two linear layers with a ReLU activation function:

$$\begin{aligned} \hat{h}_l &= \mathrm{ReLU}(\tilde{h}_l W_1 + b_1) \\ h_l &= \hat{h}_l W_2 + b_2 \end{aligned} \tag{7}$$

where $\hat{h}_l$ is the hidden representation produced by the first linear layer, W_1, W_2 are weight matrices and b_1, b_2 are bias vectors. After a second residual connection, we obtain the output h_l of the global information layer.

3.2 Local Information Layer

As highlighted in previous works such as (Wang et al., 2023) and (Ding et al., 2020), incorporating local information is also crucial for improving the performance of the NAT models. In our work, we have also integrated the local information into our model.

In the local information layer, we follow the structure of Vaswani et al. (2017), but instead of using a self-attention network, we utilize a dynamic CNN network to extract the local information. The dynamic CNN network of Wu et al. (2019) can predict different kernels at each position. This is similar to locally connected layers in the sense that the weights change at every position, but the key difference is that the weights are dynamically generated by the model rather than fixed after training.

Integrating the dynamic CNN network into our model allows us to effectively capture local information from the input sentence, which is then combined with the global information extracted by the self-attention module. This enables our model to learn both local and global contextual representations of the input sentence, leading to improved performance on the NAT models.

After the global contextual representation is output by the global information layer, we use the dynamic CNN network to extract the local contextual information. To achieve this, we apply the LightConv function of Wu et al. (2019), which takes in the input representation h and a dynamically generated filter $f(h)$ defined as a linear module with learned weight W. The output of the DynamicCNN module is given by:

$$\text{DynamicCNN}(h) = \text{LightConv}(h, f(h)) \quad (8)$$

Similar to self-attention, the DynamicCNN network changes the weights assigned to context elements over time. However, unlike self-attention, the weights of the DynamicCNN network only depend on the current time-step, rather than the entire context. This allows the model to effectively capture local contextual information by dynamically generating different filters at each time-step, resulting in improved performance on the NAT models.

3.3 Reorder Layer

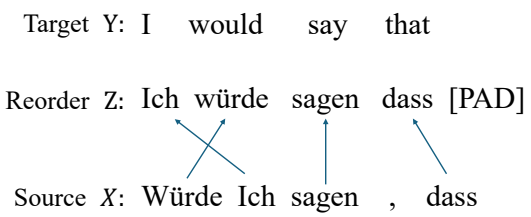


Figure 3: An example of extraction. We extract the partial source words according to the order of target words. In this example, because "," cannot be aligned to any target token, we use a special token "[PAD]" to match the length of source sentence.

Generally, because source and target sentences belong to different languages, the order of target words is not always consistent with the order of the source words. Therefore, reordering is an important component of machine translation. While traditional NAT models typically rely on a cross-attention network to learn the order of target words without an explicit reorder layer, our approach is different because we use an encoder-only model for the NAT task. As a result, we need a reorder layer to reorder the source words, since the CTC layer can only learn monotonic alignments between the source and target words. However, instead of adding an explicit reorder layer (Ran et al., 2021), which can increase model complexity, we leverage the bottom of the encoder-only model to perform source word reordering. In order for the bottom of the encoder-only model to learn the reorder information, we add an extra loss:

$$\mathcal{L}_R = -\log P(Z|X), \quad (9)$$

where $P(Z|X)$ is the reorder source sentence probability. Because we perform the reordering with the bottom of the encoder-only model, which enables us to model the reorder probability in a non-autoregressive manner:

$$P(Z|X) = \prod_i p(z_i|X), \quad (10)$$

Here, $p(z_i|X)$ is calculated by the lower half of the encoder, and we use an output layer (Reorder Layer in Fig.2) to obtain the reorder sentence probability $p(z_i|X)$. Z is the source words extracted according to the order of target words. Note that the target words are not always aligned to the source words, so we only extract partial source words and pad the reordered source sentence with a special token to match the length of the source sentences. Figure 3 illustrates this process.

This approach enables our model to learn reorder information during training. During inference, we do not need to precisely predict the reordered source words, but only need the reordered source representations. In our model, the reorder layer is used to make the pre-half of the encoder-only model learn the reorder information. Once training converges, the lower half of the encoder can perform the source word reordering without the output layer.

3.4 Upsample Layer

To ensure that the CTC layer can be applied to machine translation, where the source and target sentences may differ in length, we propose an upsample layer that increases the length of the reordered source sequence. For a reordered source representation $Z \in \mathbb{R}^{B\times T_z\times d}$, where B is the batch size, T_z the sequence length, and d the feature dimension, the layer applies a linear map to the feature dimension of every frame:

$$\hat{Z} = W_{up}Z + b_{up}, \qquad \hat{Z} \in \mathbb{R}^{B\times T_z\times \lambda d}, \quad (11)$$

where $W_{up} \in \mathbb{R}^{d\times\lambda d}$ and $b_{up} \in \mathbb{R}^{\lambda d}$ are the weight matrix and the bias vector of the map, and λ is the upsample ratio. We write the whole operation as $\text{Up}(\cdot)$: reshaping the enlarged feature dimension of $h \in \mathbb{R}^{B\times T_z\times d}$ into the sequence dimension gives $\text{Up}(h) \in \mathbb{R}^{B\times \lambda T_z\times d}$, whose length λT_z is no shorter than the target length N, so that the CTC layer can be applied to the encoder-only model.

3.5 Multi-objective Learning

In our approach, we propose utilizing an attention output layer as an auxiliary task to train the CTC

layer within the multi-objective learning framework of Kim et al. (2017). As shown in Figure 2, the CTC layer and attention output layer share the main body of the model except for the last layer. This way, the attention layer can estimate the desired alignments in long sequences, which helps the CTC layer to learn the monotonic alignment between the source and target words. The objective to be maximized is a combination of the CTC and attention objectives:

$$\begin{aligned}\mathcal{L}_o &= \log P_{\text{ctc}}(\alpha|\hat{Z}, Z, X) \\ &\quad + \log P_{\text{att}}(Y|\hat{Z}, Z, X),\end{aligned} \tag{12}$$

where $P_{\text{ctc}}(\alpha|\hat{Z}, Z, X)$ is learned by the CTC layer, and $P_{\text{att}}(Y|\hat{Z}, Z, X)$ is learned by the attention output layer with cross-entropy loss. Because we add an upsample layer in the model, generally, the length of output is greater than the length of target sentence, and the CTC layer can align the output to the target words by adding the blank token. But the cross-entropy loss needs the strict alignments between the output and target words, which is not compatible in our model. To address this issue, we borrow the roll-in policy of Chan et al. (2020) to generate a pseudo-target sentence of the same length. With pseudo-target sentence, the cross-entropy loss can be utilized to train the attention output layer.

The upsample layer gets no explicit supervision from Eq. 12, which only constrains the shared encoder, so we add an upsample loss, i.e. the CTC objective evaluated on the upsampled frames:

$$\mathcal{L}_U = -\log \sum_{\alpha \in \Gamma(Y)} \prod_{i=1}^{\lambda T_z} p(\alpha_i \mid \text{Up}(h)), \tag{13}$$

where $\text{Up}(\cdot)$ is defined in Section 3.4, h is the reordered source representation, and $\Gamma(Y)$ is the set of length-λT_z alignments over $\mathcal{V}^+$ that collapse to Y under the CTC merge rule (Eq. 4). Here $p(\alpha_i \mid \text{Up}(h))$ is the frame-level label distribution produced by the shared encoder followed by the CTC output layer, and α_i is the label or the blank token emitted at frame i. Like the CTC term of $\mathcal{L}_o$, $\mathcal{L}_U$ keeps the upsampled frames aligned with the target words.

Therefore, given the dataset of parallel training pairs $\{x^m, y^m\}_{m=1}^M$, the parameters of the whole model are learned by minimizing the loss $\mathcal{L}(\theta)$:

$$\mathcal{L}(\theta) = -\frac{1}{M} \sum_{m=1}^{M} (\mathcal{L}_R + \mathcal{L}_o + \lambda_u \mathcal{L}_U) \tag{14}$$

where M denotes the number of parallel training pairs and λ_u is the weight of the upsample loss.

3.6 Joint decoding

Both the CTC layer and the attention layer produce a distribution over target sequences after training, and joint decoding combines their sequence-level scores in the log domain, as illustrated in Figure 4. The decoding objective is

$$\begin{aligned}\hat{Y} = \arg\max_{Y} [&\beta \log P_{\text{ctc}}(Y|X) \\ &+ (1-\beta) \log P_{\text{att}}(Y|X)]\end{aligned} \tag{15}$$

where $P_{\text{ctc}}(Y|X)$ and $P_{\text{att}}(Y|X)$ are the sequence-level probabilities given by Eq. 4 and by the attention layer, both normalized over target sequences of the same length; $\beta \in [0, 1]$ is the CTC weight and $1 - \beta$ is the attention weight.

During inference, we decode by one-pass prefix search in the log domain. Let $y_{1:t}$ be a target prefix and $\Gamma(y_{1:t})$ the set of length-λT_z alignments $\alpha = \{\alpha_1, \ldots, \alpha_{\lambda T_z}\}$ over $\mathcal{V}^+$ that collapse to $y_{1:t}$ under the CTC merge rule, where α_i is the label or the blank token emitted at frame i. The CTC prefix probability marginalizes over these alignments:

$$\begin{aligned}&P_{\text{ctc}}(y_{1:t}|\hat{Z}, Z, X) \\ &\qquad = \sum_{\alpha \in \Gamma(y_{1:t})} \prod_{i=1}^{\lambda T_z} p_{\text{ctc}}(\alpha_i|\hat{Z}, Z, X),\end{aligned} \tag{16}$$

where the sum runs over $\Gamma(y_{1:t})$ and the product over the λT_z frames. Since Eq. 15 interpolates sequence-level log probabilities, the same rule applies frame by frame: extending a prefix with the word y_t adds

$$\begin{aligned}\Delta(y_{1:t}) = &\beta \log p_{\text{ctc}}(y_t|\hat{Z}, Z, X) \\ &+ (1-\beta) \log p_{\text{att}}(y_t|\hat{Z}, Z, X)\end{aligned} \tag{17}$$

to its accumulated score, where $p_{\text{ctc}}(y_t|\cdot)$ and $p_{\text{att}}(y_t|\cdot)$ are the frame-level CTC and position-level attention probabilities. Equations 15 and 17 thus define a single log-space objective, evaluated over sequences in Eq. 15 and over frames in Eq. 17, and no raw probability is ever interpolated. The dynamic program keeps the forward variable $\alpha(t, i)$,

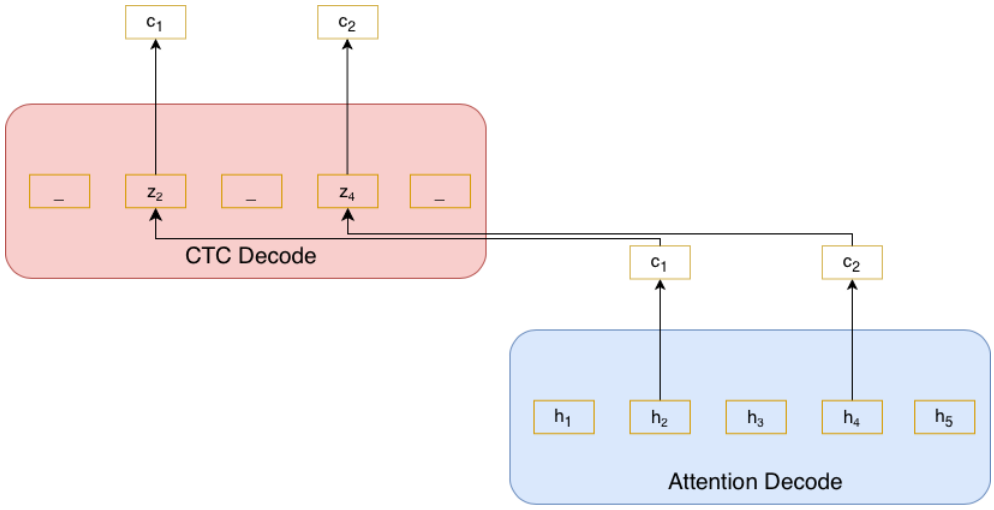


Figure 4: The Illustration of joint decoding. The probabilities of target words calculated by attention decoding are combined with CTC probabilities.

the score of the alignments that have emitted the first t target words and end at frame i, initialized as $\alpha(0,0) = 0$ and updated by the standard CTC recursion over the blank and repeated-label transitions, where each extension adds $\Delta(y_{1:t})$ and the final hypothesis score is $\alpha(N, \lambda T_z)$.

4 Experiments

4.1 Experimental Setup

Datasets: Our method is evaluated on several widely used benchmark translation tasks, including WMT14 En→De, WMT16 EN→Ro, and IWSLT14 De→EN, which consist of approximately 4.5 million , 610,000, and 196,000 bilingual sentences, respectively. First, we generate the reordered source sentence using the self-on-the-fly alignment tool, FastAlign[1]. We then tokenize the sentences into subword units using BPE (Sennrich et al., 2016). Finally, we utilize the method proposed by (Lee et al., 2018) to preprocess the datasets.

Sequence-Level Distillation We apply sequence-level distillation (Kim and Rush, 2016) on the training set for all tasks, as its effectiveness has been demonstrated in (Zhou et al., 2019). As pointed out by (Wang et al., 2019), the performance of teacher models can affect the performance of NAT models. Hence, to ensure a fair comparison, we use the distilled datasets provided in Mask-Predict[2].

Metrics We adopt the widely-used BLEU (Papineni et al., 2002) metric to evaluate the accuracy of our model. To compare the decoding speed, we compute the latency, which is the average decoding time per sentence, on the test set of WMT14 EN→DE.

Model Detail: For the IWSLT14 benchmark, we used a hyperparameter configuration consisting of 12 layers with 9.4M parameters (compared to the vanilla NAT model's 9.6M parameters). Our model had an embedding size of 256, hidden size of 512, and 4 attention heads. The local information layer utilized kernel sizes of 7, 15, 31, 31, 15, 7. For the WMT14 EN→DE and WMT16 EN→RO benchmarks, our model had 14 layers and the same hyperparameter values as the vanilla NAT model (Gu et al., 2017), which consists of 66M parameters. This included an embedding size of 512, hidden size of 2048, and 8 attention heads. For the WMT tasks, the local information layer had kernel sizes of 7, 15, 31, 31, 31, 31, 15, 7. Our proposed model was implemented using the fairseq [3] library. We used the same training parameters as the paper (Gu et al., 2017), including the Adam optimizer and word batches of 64K for WMT tasks and 8K for the IWSLT14 benchmarks. All models were trained for 300K steps using Nvidia 1080Ti GPUs.

[1] https://github.com/clab/fast_align
[2] https://github.com/facebookresearch/Mask-Predict
[3] https://github.com/facebookresearch/fairseq

4.2 Main Results

Besides our baseline NAT-CTC (Saharia et al., 2020), we also compare our model with vanilla NAT (Gu et al., 2017), NAT-IR (Lee et al., 2018), Hint-NAT (Li et al., 2018), DCRF-NAT (Sun et al., 2019),TCL-NAT (Guo et al., 2020), FlowSeq (Ma et al., 2019),GLAT (Qian et al., 2020), DSLP (Huang et al., 2022a), LaNAT (Shu et al., 2020), CMLM (Ghazvininejad et al., 2019). We show the results in Table 1.

Table 1 lists the primary results of our Hybrid-CTC model and other baselines. Our model is identified as Hybrid-CTC. As shown in Table 1, our model significantly enhances translation quality, outperforming most previous models, including iterative refinement NAT models. We make several observations about the results from Table 1:

- Our non-autoregressive Hybrid-CTC model outperforms most iterative refinement NAT models. Although CMLMC outperforms our approach with an iterative step of 10, our model maintains the advantage of performing a single decoding step, making it faster than iterative refinement models.

- Our non-autoregressive model outperforms all other fully non-autoregressive models in all tasks, even those that use noisy parallel decoding. While the NAG-BERT model is based on an encoder-only model and utilizes the knowledge of the pre-trained BERT model, which has more parameters than other NAT models, it still does not match the performance of our Hybrid-CTC model. The NAG-BERT model achieves a BLEU score of 30.45 on the IWSLT14 DE→EN task, while our model achieves a better performance of 33.27 BLEU using only an encoder-only model trained without pre-training. Despite lacking the ability to extract knowledge from pre-trained models, our model also achieves better performance in a simple way. Additionally, Our model also achieves the improvements compared to other CTC models, such as CTC-NMLA, which includes n-gram information in its CTC models.

- Compared to our baseline model NAT-CTC, which only utilizes an encoder-only model and a CTC layer to learn the alignments between the source and target words, our proposed model shows significant improvements. On the IWSLT De→En task, NAT-CTC achieved a BLEU score of 28.35, but Hybrid-CTC achieved a score of 33.27, resulting in a gain of 5.0 BLEU. On the WMT14 En→De task, our model achieved a score of 27.83 BLEU, which is a 2.5 BLEU improvement. Due to the similarity in model structures, we attribute these improvements to the proposed approach, including reorder layer and hybrid decoding.

In addition, following previous work (Gu et al., 2017; Lee et al., 2018; Gehring et al., 2017; Qian et al., 2020), we evaluated the decoding latency on the WMT14 English→German test sets. We set the batch size to 1, and performed this experiment on a single Nvidia 1080Ti GPU. We present a scatter plot in Figure 5 to compare the decoding speed and translation accuracy of our model with other non-autoregressive machine translation models.

As can be observed, our model outperforms other non-autoregressive machine translation models at a comparable speedup. While the Hint-NAT model achieves faster speed, its translation accuracy is inferior to our model. Furthermore, compared with the autoregressive model Transformer, our model achieves comparable results while providing a speed gain of around 14.0×.

4.3 Analyses

Effect on different length of sentences: To assess the influence of sentence length on our model's performance, we conducted experiments on the IWSLT14 German→English test set, dividing the sentences into different buckets based on the length of the target sentences. The results are presented in Figure 6. It can be observed that the autoregressive model performs better in predicting translations when the sentences are long. However, compared to the vanilla NAT model, our model achieves comparable accuracy to the autoregressive machine translation model, particularly on longer sentences.

Compared to the previous state-of-the-art model GLAT, our model achieves better performance across various sentence lengths. While the difference between our model and GLAT is small for shorter sentences, it becomes more significant for longer ones. In particular, our model maintains stable performance even for sentences longer than 60 words, whereas GLAT's performance deteriorates.

The Effect of the CTC Weight: In Eq. 15, β is the

Table 1: The results of our model on WMT14 EN→DE, WMT16 EN→RO, and IWSLT14 DE→EN translation tasks. "k" denotes the number of decoding iterations. "NPD" represents noisy parallel decoding. "/" indicates that the result is not reported. "*" denotes the results are obtained by our reimplementation.

Model			WMT14 EN→DE	WMT16 EN→RO	IWSLT14 DE→EN
AT model		Transformer	27.2	33.70	34.65
Iterative-Based NAT models		NAT-IR(k=10)(Lee et al., 2018)	21.61	29.32	23.94
		LaNAT(k=4)(Shu et al., 2020)	26.30	/	/
		CMLM(k=4)(Ghazvininejad et al., 2019)	25.94	32.53	30.42
		CMLM(k=10)	27.03	33.08	31.71
		RewriteNAT (k=2.7)(Geng et al., 2021)	27.83	33.63	/
		CMLMC (k=10)(Huang et al., 2022c)	28.37	34.57	/
Fully NAT models		NAT(Gu et al., 2017)	17.69	26.22	/
		Hint-NAT(Li et al., 2018)	21.11	/	/
		TCL-NAT(Guo et al., 2020)	21.94	/	28.16
		DCRF-NAT(Sun et al., 2019)	23.44	/	/
		Flowseq(Ma et al., 2019)	21.45	29.34	27.55
		CMLM	18.05	27.32	/
		GLAT(Qian et al., 2020)	25.21	31.19	29.80*
		ReorderNAT (NAT)(Ran et al., 2021)	22.79	29.30	/
		CTC(Saharia et al., 2020)	25.7	32.2	/
		DSLP(Huang et al., 2022a)	27.02	34.17	/
		NAG-BERT(Su et al., 2021)	/	/	30.45
		CTC-NMLA(Shao and Feng, 2022)	27.57	33.86	/
	w/ NPD	NAT(NPD=100)	19.17	29.79	24.21
		Hint-NAT(NPD=9)	25.20	/	/
		DCRF-NAT(NPD9)	26.07	/	29.99
		GLAT(NPD=7)	26.55	32.87	31.23*
	Ours	NAT-CTC	25.32	31.83	28.35
		Hybrid-CTC	27.83	33.88	33.27

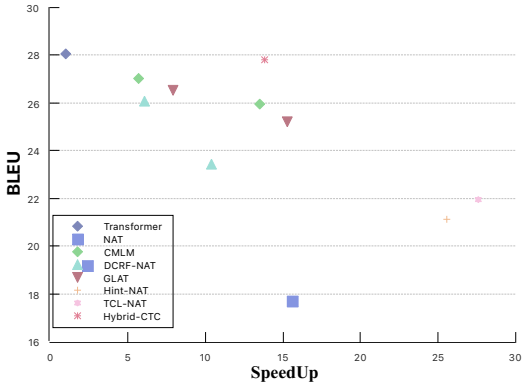


Figure 5: The trade-off between speed-up and BLEU on WMT14 English→German.

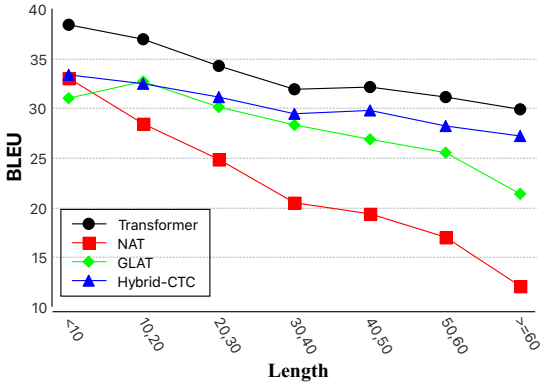


Figure 6: The BLEU scores of different lengths

weight of the CTC score and $1-\beta$ is the weight of the attention score. To evaluate the impact of this interpolation, we vary the attention weight $1-\beta$ from 0.0 to 0.5 on the IWSLT14 German→English test set; the results are listed in Table 2, where the swept value is $1-\beta$.

As shown in Table 2, the performance varies noticeably with the attention weight. It remains stable for $1-\beta$ between 0.0 and 0.07 and reaches the best BLEU score at $1-\beta = 0.06$. A further increase degrades the performance in proportion,

Table 2: The performance of our model with different attention weights $1-\beta$ in Eq. 15.

$1-\beta$	0.0	0.01	0.02	0.05	0.06	0.07	0.1	0.2	0.5
BLEU	33.15	33.18	33.20	33.23	33.27	33.24	33.2	31.73	23.42

and at $1-\beta = 0.5$ the model falls below NAT-CTC. Since $1-\beta = 0.06$ is optimal, we set the attention weight to 0.06, i.e., the CTC weight β to 0.94, for all tasks.

Ablation Analysis: We also conduct an extensive ablation study with the proposed NAT model on the IWSLT dataset. We conducted experiments on the IWSLT14 German→English dataset to evaluate the performance of models with different components. The results are presented in Table 3.

Table 3: The performance of our model with different components.

Model	BLEU
NAT-CTC	28.35
+ Reorder	31.71
+ Upsample Loss	32.11
+ Local Information Layer	32.46
+ joint decoding	33.27

As we can see from Table 3, we utilize an encoder-only CTC model NAT-CTC as our base model, and add the component one after another. When we add the reorder module on the base model NAT-CTC, we can observe a significant improvement. We suppose that due to the monotonic alignments, the encoder-only CTC model can not learn the reorder information, which always exists in machine translation tasks. In this case, the explicit reorder information will improve the performance of NAT-CTC model.

Generally, the upsample operation is unsupervised, and the model learns the upsample operation by itself. To address this issue, we also introduce the upsample loss of Eq. 13 to guide the upsampling process. As shown in Table 3, the upsample loss improves the BLEU score of the NAT-CTC model from 31.71 to 32.11.

Meantime, in our model, we also introduce a local information layer to extract the local information. In this section, we also evaluate the effect of the local information layer. From Table 3, we can see that the local information layer can improve the performance of NAT-CTC model.

Finally, we conduct the experiments to evaluate the performance of hybrid CTC/attention decode proposed in this work. From Table 3, we can see that the performance of hybrid CTC/attention decode can improve the performance of NAT-CTC model significantly.

5 Related Works

To speed up machine translation decoding, Gu et al. (2017) introduced a non-autoregressive decoding approach. However, when cross-entropy loss was adapted to train the vanilla NAT model, errors occurred even when the edit distances were small, as every misplaced word incurred a penalty.

To improve alignment between output and target sentences, several works have introduced methods such as aligned cross entropy (Ghazvininejad et al., 2020), which computes the cross-entropy loss using an alignment between the sequence of token labels and token distribution predictions. Du et al. (2021) further improved this approach by incorporating lexical matching to remove penalties for word order errors, making the loss more compatible with parallel decoding in NAT. While these methods improved the performance of cross-entropy losses, they have not fully solved the problem of explicit alignment. Libovický and Helcl (2018) proposed a CTC-based NAT model, which differs from cross-entropy loss as it relaxes the alignment restriction. However, it cannot learn non-monotonic alignments, which are common in machine translation. To address this limitation, we propose a reorder-based model that handles non-monotonic alignment.

Our model is related to Shao and Feng (2022), who introduce non-monotonic latent alignments for CTC-based models by incorporating n-gram matching into the CTC model. They do not actually solve the word-reordering problem in machine translation, but rather mitigate it with clever overlap-based matching. Meanwhile, they adapt the traditional encoder-decoder framework for NAT models, which can further alleviate the word-reordering problem. In contrast, we address the word-reordering problem using the bottom of the encoder.

Meanwhile, our model is also related to Ran et al. (2021), who incorporate an explicit reorder module into NAT models. In contrast, our model does not have an explicit reordering model, but we address word reordering through the bottom of the model.

6 Conclusion

This study addresses the limitations of CTC in handling non-monotonic alignments, a common

challenge in machine translation. The proposed model adopts a novel approach to NAT by incorporating a reorder-based mechanism. Integrating a reorder module into the encoder-only NAT model effectively addresses monotonic alignment issues caused by CTC loss. Additionally, we utilize a hybrid CTC/attention decoding method that synergizes the strengths of both models to boost translation performance. According to our experiments on benchmark datasets, our approach yields substantial improvement in the performance of non-autoregressive machine translation. Our findings could serve as a valuable guideline for future research on the application of NAT with CTC being employed.

Acknowledgements

We are very grateful to the reviewers for their careful work. This work was supported by the National Natural Science Foundation of China (Nos. 61732005, 61671064), the Natural Science Foundation of Henan Province (Nos. 252300423330), Henan Provincial Science and Technology Tackling Plan (No. 262102210185), the Doctoral Research Start-up Fund Project at Nanyang Institute of Technology (No. NGBJ-2024-12).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

William Chan, Chitwan Saharia, Geoffrey Hinton, Mohammad Norouzi, and Navdeep Jaitly. 2020. Imputer: Sequence modelling via imputation and dynamic programming. In *International Conference on Machine Learning*, pages 1403–1413. PMLR.

Liang Ding, Longyue Wang, Di Wu, Dacheng Tao, and Zhaopeng Tu. 2020. Context-aware cross-attention for non-autoregressive translation. *arXiv preprint arXiv:2011.00770*.

Cunxiao Du, Zhaopeng Tu, and Jing Jiang. 2021. Order-agnostic cross entropy for non-autoregressive machine translation. In *International Conference on Machine Learning*, pages 2849–2859. PMLR.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *ICML*.

Xinwei Geng, Xiaocheng Feng, and Bing Qin. 2021. Learning to rewrite for non-autoregressive neural machine translation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3297–3308.

Marjan Ghazvininejad, Vladimir Karpukhin, Luke Zettlemoyer, and Omer Levy. 2020. Aligned cross entropy for non-autoregressive machine translation. In *International Conference on Machine Learning*, pages 3515–3523. PMLR.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Mask-predict: Parallel decoding of conditional masked language models. *arXiv preprint arXiv:1904.09324*.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. 2017. Non-autoregressive neural machine translation. *arXiv preprint arXiv:1711.02281*.

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.

Yosuke Higuchi, Shinji Watanabe, Nanxin Chen, Tetsuji Ogawa, and Tetsunori Kobayashi. 2020. Mask ctc: Non-autoregressive end-to-end asr with ctc and mask predict. *arXiv preprint arXiv:2005.08700*.

Chenyang Huang, Hao Zhou, Osmar R Zaïane, Lili Mou, and Lei Li. 2022a. Non-autoregressive translation with layer-wise prediction and deep supervision. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 10776–10784.

Fei Huang, Hao Zhou, Yang Liu, Hang Li, and Minlie Huang. 2022b. Directed acyclic transformer for non-autoregressive machine translation. In *International Conference on Machine Learning*, pages 9410–9428. PMLR.

Xiao Shi Huang, Felipe Perez, and Maksims Volkovs. 2022c. Improving non-autoregressive translation models without distillation. In *International Conference on Learning Representations*.

Suyoun Kim, Takaaki Hori, and Shinji Watanabe. 2017. Joint ctc-attention based end-to-end speech recognition using multi-task learning. In *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4835–4839. IEEE.

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.

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.

Zhuohan Li, Di He, Fei Tian, Tao Qin, Liwei Wang, and Tie-Yan Liu. 2018. Hint-based training for non-autoregressive translation.

Jindřich Libovickỳ and Jindřich Helcl. 2018. End-to-end non-autoregressive neural machine translation with connectionist temporal classification. *arXiv preprint arXiv:1811.04719*.

Xuezhe Ma, Chunting Zhou, Xian Li, Graham Neubig, and Eduard Hovy. 2019. Flowseq: Non-autoregressive conditional sequence generation with generative flow. *arXiv preprint arXiv:1909.02480*.

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.

Lihua Qian, Hao Zhou, Yu Bao, Mingxuan Wang, Lin Qiu, Weinan Zhang, Yong Yu, and Lei Li. 2020. Glancing transformer for non-autoregressive neural machine translation. *arXiv preprint arXiv:2008.07905*.

Qiu Ran, Yankai Lin, Peng Li, and Jie Zhou. 2021. Guiding non-autoregressive neural machine translation decoding with reordering information. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 13727–13735.

Chitwan Saharia, William Chan, Saurabh Saxena, and Mohammad Norouzi. 2020. Non-autoregressive machine translation with latent alignments. *arXiv preprint arXiv:2004.07437*.

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.

Chenze Shao and Yang Feng. 2022. Non-monotonic latent alignments for ctc-based non-autoregressive machine translation. *arXiv preprint arXiv:2210.03953*.

Chenze Shao, Yang Feng, Jinchao Zhang, Fandong Meng, Xilin Chen, and Jie Zhou. 2019. Retrieving sequential information for non-autoregressive neural machine translation. *arXiv preprint arXiv:1906.09444*.

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*, volume 34, pages 198–205.

Raphael Shu, Jason Lee, Hideki Nakayama, and Kyunghyun Cho. 2020. Latent-variable non-autoregressive neural machine translation with deterministic inference using a delta posterior. In *Proceedings of the aaai conference on artificial intelligence*, volume 34, pages 8846–8853.

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. *arXiv preprint arXiv:2102.08220*.

Zhiqing Sun, Zhuohan Li, Haoqing Wang, Di He, Zi Lin, and Zhihong Deng. 2019. Fast structured decoding for sequence models. *Advances in Neural Information Processing Systems*, 32.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Shuheng Wang, Heyan Huang, and Shumin Shi. 2023. Better localness for non-autoregressive transformer. *ACM Transactions on Asian and Low-Resource Language Information Processing*.

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

Felix Wu, Angela Fan, Alexei Baevski, Yann N Dauphin, and Michael Auli. 2019. Pay less attention with lightweight and dynamic convolutions. *arXiv preprint arXiv:1901.10430*.

Pan Xie, Zhi Cui, Xiuyin Chen, Xiaohui Hu, Jianwei Cui, and Bin Wang. 2020. Infusing sequential information into conditional masked translation model with self-review mechanism. *arXiv preprint arXiv:2010.09194*.

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