Exploring Non-Monotonic Latent Alignments for Non-Autoregressive Machine Translation

Anonymous Author(s)
Affiliation
Address
email

Abstract

Non-autoregressive translation (NAT) models are typically trained with the cross-entropy loss, which forces the model outputs to be aligned verbatim with the target sentence and will highly penalize small shifts in word positions. Latent alignment models relax the explicit alignment by marginalizing out all monotonic latent alignments with the CTC loss. However, they cannot handle non-monotonic alignments, which is non-negligible as there is typically global word reordering in machine translation. In this work, we explore non-monotonic latent alignments for NAT. We extend the alignment space to non-monotonic alignments to allow for the global word reordering and further consider all alignments that overlap with the target sentence. We non-monotonically match the alignments to the target sentence and train the latent alignment model to maximize the F1-score of non-monotonic matching. Extensive experiments on major WMT benchmarks show that our method substantially improves the translation performance and achieves comparable performance to the autoregressive Transformer with only one-iteration parallel decoding.

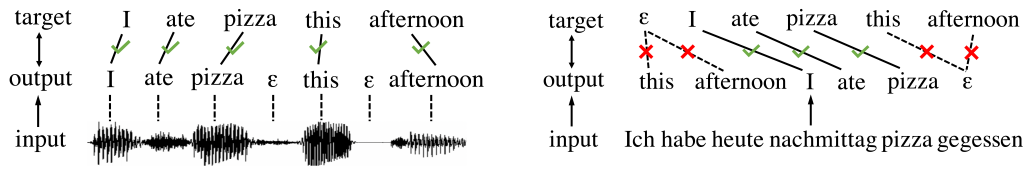
1 Introduction

Non-autoregressive translation (NAT) models achieve significant decoding speedup in neural machine translation [NMT, 1, 43] by generating target words simultaneously [17]. This advantage usually comes at the cost of translation quality due to the mismatch of training objectives. NAT models are typically trained with the cross-entropy loss, which forces the model outputs to be aligned verbatim with the target sentence and will highly penalize small shifts in word positions. The explicit alignment required by the cross-entropy loss cannot be guaranteed due to the multi-modality problem that there exist many possible translations for the same sentence [17], making the cross-entropy loss intrinsically mismatched with NAT.

As the cross-entropy loss can not evaluate NAT outputs properly, many efforts have been devoted to designing better training objectives for NAT [28, 38, 39, 13, 35, 9]. Among them, latent alignment models [28, 35] relax the alignment restriction by marginalizing out all monotonic latent alignments with the connectionist temporal classification loss [CTC, 15], which receive much attention for the ability to generate variable length translation.

Latent alignment models generally make a strong monotonic assumption on the mapping between model output and the target sentence. As illustrated in Figure 1a, the monotonic assumption holds in classic application scenarios of CTC like automatic speech recognition (ASR) as there is a natural monotonic mapping between the speech input and ASR target. However, non-monotonic alignments are non-negligible in machine translation as there is typically global word reordering, which is a common source of the multi-modality problem. As Figure 1b shows, when the target sentence is "I ate pizza this afternoon" but the model produces a translation with a different but correct word

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.



(a) CTC for ASR (b) CTC for NAT

Figure 1: Illustration of the monotonic alignment assumption of CTC: (a) CTC for ASR where there is a natural monotonic mapping between the speech input and ASR target, (b) CTC for NAT where there exists global word reordering that induces non-monotonic alignment. ϵ is the blank token that means 'output nothing'.

ordering "this afternoon I ate pizza", the CTC loss cannot handle this non-monotonic alignment between output and target and wrongly penalizes the model.

In this paper, we propose to model non-monotonic latent alignments for non-autoregressive machine translation. We first extend the alignment space from monotonic alignments to non-monotonic alignments to allow for the global word reordering in machine translation. Without the monotonic structure, we have to optimize the best alignment found by the Hungarian algorithm [42, 27, 7, 9] since it becomes difficult to marginalize out all non-monotonic alignments with dynamic programming. This difficulty can be overcome by not requiring an exact match between alignments and the target sentence. In practice, it is not necessary to have the translation include exact words as contained in the target sentence, but it would be favorable to have a large overlap between them. Therefore, we further extend the alignment space by considering all alignments that overlap with the target sentence. Specifically, we are interested in the overlap of n-grams, which is the core of some evaluation metrics (e.g., BLEU). We accumulate n-grams from all alignments regardless of their positions and non-monotonically match them to target n-grams. The latent alignment model is trained to maximize the F1-score of n-gram matching, which reflects the translation quality to a certain extent [31].

We conduct experiments on major WMT benchmarks for NAT (WMT14 En↔De, WMT16 En↔Ro), which shows that our method substantially improves the translation performance and achieves comparable performance to autoregressive Transformer with only one-iteration parallel decoding.

2 Background

2.1 Non-Autoregressive Machine Translation

[17] proposes non-autoregressive neural machine translation, which achieves significant decoding speedup by generating target words simultaneously. NAT breaks the dependency among target tokens and factorizes the joint probability of target words in the following form:

$$p(y|x,\theta) = \prod_{t=1}^{T} p_t(y_t|x,\theta), \tag{1}$$

where x is the source sentence belonging to the input space $\mathcal{X}$, $y = \{y_1, ..., y_T\}$ is the target sentence belonging to the output space $\mathcal{Y}$, and $p_t(y_t|x,\theta)$ indicates the translation probability in position t.

The vanilla-NAT has a length predictor that takes encoder states as input to predict the target length. During the training, the target length is set to the golden length, and the vanilla-NAT is trained with the cross-entropy loss, which explicitly aligns model outputs to target words:

$$\mathcal{L}_{CE}(\theta) = -\sum_{t=1}^{T} \log(p_t(y_t|x,\theta)). \tag{2}$$

2.2 NAT with Latent Alignments

The vanilla-NAT suffers from two major limitations. The first limitation is the explicit alignment required by the cross-entropy loss, which cannot be guaranteed due to the multi-modality problem

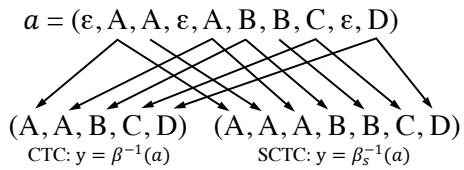


Figure 2: An example of target sentences obtained from collapsing functions of CTC and SCTC. The collapsing function of SCTC is β_s^{-1}, which only remove blanks in the alignment a, where the collapsing function of CTC removes repetitions first and then removes blanks.

and therefore leads to the inaccuracy of the loss. The second limitation is the requirement of target length prediction. The predicted length may not be optimal and cannot be changed dynamically, so it is often required to use multiple length candidates and re-score them to produce the final translation.

The two limitations can be addressed by using CTC-based latent alignment models [15, 28, 35], which extend the output space $\mathcal{Y}$ with a blank token ϵ that means 'output nothing'. We define the extended output space as $\mathcal{Y}^*$. Following prior works [14, 35], we refer to the elements $a \in \mathcal{Y}^*$ as alignments, since the location of the blank tokens determines an alignment between the extended output and target sentence. Assume the generated alignments have length T_a, we define a function $\beta(y)$ which returns a subset of $\mathcal{Y}^*$ including all possible alignments of length T_a for y, where $\beta^{-1} : \mathcal{Y}^* \mapsto \mathcal{Y}$ is the collapsing function that first collapses all consecutive repeated words in a and then removes all blank tokens to obtain the target sentence. As illustrated in Figure 2, the alignment $a = (\epsilon, A, A, \epsilon, A, B, B, C, \epsilon, D)$ is collapsed to the target sentence $y = (A, A, B, C, D)$ with a monotonic mapping from target positions to alignment positions. During the training, latent alignment models marginalize out all alignments with the CTC loss [15]:

$$\log p(y|x, \theta) = \log \sum_{a \in \beta(y)} p(a|x, \theta), \tag{3}$$

where the alignment probability $p(a|x, \theta)$ is modeled by a non-autoregressive Transformer:

$$p(a|x, \theta) = \prod_{t=1}^{T_a} p_t(a_t|x, \theta). \tag{4}$$

NAT models with latent alignments achieve superior performance by overcoming the two major limitations of NAT [28, 35, 16, 24, 32, 45]. The weakness in handling non-monotonic latent alignments is noticed in previous work [35, 16] but remains unsolved.

3 Approach

In this section, we explore non-monotonic latent alignments for NAT. We introduce a simplified CTC loss in section 3.1, which has a simpler structure that helps further analysis and derivation under the regular CTC loss. We explore non-monotonic alignments under the simplified CTC loss in section 3.2, and utilize the results to derive non-monotonic alignments under the regular CTC loss in section 3.3. Finally, we introduce the training strategy to combine monotonic and non-monotonic latent alignments in section 3.4.

3.1 Simplified Connectionist Temporal Classification

In CTC, the collapsing function β^{-1} maps an alignment to a target sentence in two steps: (1) collapses all consecutive repeated words, (2) removes all blank tokens. The two operations make the alignment space $\beta(y)$ complex. Therefore, we first introduce the simplified connectionist temporal classification loss (SCTC), which has a simpler structure that helps the derivation of non-monotonic alignments, and the results on SCTC are helpful for further analysis under the regular CTC loss.

To simplify the alignment structure, we consider only using one operation in the collapsing function. One option is only collapsing all consecutive repeated words in the alignment. The concern is that it will limit the expressive power of the model. For example, the target sentence $y = (A, A, B, C, D)$

has probability 0 since it has repeated words that cannot be mapped from any alignment. Therefore, we favor another option which only removes all blank tokens. In this way, the expressive power stays the same and the alignment space becomes much simpler, where alignments of y simply contain T target words and $T_a - T$ blank tokens.

We denote this simplified loss as SCTC, and illustrate the difference between CTC and SCTC in Figure 2. The alignment space for the target sentence y is defined as $\beta_s(y)$, and the collapsing function is defined as β_s^{-1}. We can still obtain the translation probability $p(y|x, \theta)$ with dynamic programming. We define the forward variable $\alpha_t(s)$ to be the total probability of $y_{1:s}$ considering all possible alignments $a_{1:t}$, which can be calculated recursively from $\alpha_{t-1}(s)$ and $\alpha_{t-1}(s-1)$:

$$\alpha_t(s) = \alpha_{t-1}(s-1)p_t(y_s|x, \theta) + \alpha_{t-1}(s)p_t(\epsilon|x, \theta). \tag{5}$$

Finally, the total translation probability $p(y|x, \theta)$ is given by the forward variable $\alpha_{T_a}(T)$, so we can train the model with the cross-entropy loss $\mathcal{L}_{sctc}(\theta) = -\log \alpha_{T_a}(T)$.

3.2 Non-Monotonic Alignments under SCTC

3.2.1 Bipartite Matching

In this section, we explore non-monotonic alignments under the SCTC loss, which are helpful for further analysis under the regular CTC loss. We first extend the alignment space from monotonic alignments $\beta_s(y)$ to non-monotonic alignments $\gamma_s(y)$ to allow for the global word reordering in machine translation, where $\gamma_s(y)$ is defined as:

$$\gamma_s(y) = \bigcup_{y' \in P(y)} \beta_s(y'). \tag{6}$$

In the above definition, $P(y)$ represents all permutations of y. For example, $P((A, B)) = \{(A, B), (B, A)\}$. By enumerating $P(y)$, we consider all possible word reorderings of the target sentence. Ideally, we want to traverse all alignments in $\gamma_s(y)$ to calculate the log-likelihood loss:

$$\mathcal{L}_{sum}(\theta) = -\log \sum_{a \in \gamma_s(y)} p(a|x, \theta). \tag{7}$$

However, without the monotonic structure, it becomes difficult to marginalize out all latent alignments with dynamic programming. Alternatively, we can minimize the loss of the best alignment, which is an upperbound of Equation 7:

$$\mathcal{L}_{max}(\theta) = -\log \max_{a \in \gamma_s(y)} p(a|x, \theta). \tag{8}$$

Following prior work [42, 7, 9], we formulize finding the best alignment as a maximum bipartite matching problem and solve it with the Hungarian algorithm [27]. Specifically, we observe that the alignment space $\gamma_s(y)$ is simply permutations of T target words and $T_a - T$ blank tokens. Therefore, finding the best alignment is equivalent to finding the best bipartite matching between the T_a model predictions and the T target words plus $T_a - T$ blank tokens, where the two sets of nodes are connected by edges with the prediction log-probability as weights.

3.2.2 N-Gram Matching

Without the monotonic structure desired for dynamic programming, calculating Equation 7 becomes difficult. This difficulty is also caused by the strict requirement of exact match for alignments, which makes it intractable to simplify the summation of probabilities. In practice, it is not necessary to force the exact match between alignments and the target sentence since a large overlap is also favorable. Therefore, we further extend the alignment space by considering all alignments that overlap with the target sentence. Specifically, we are interested in the overlap of n-grams, which is the core of some evaluation metrics (e.g., BLEU).

We propose a non-monotonic and non-exclusive n-gram matching objective based on SCTC to encourage the overlap of n-grams. Following the underlying idea of probabilistic matching [37], we introduce the probabilistic variant of the n-gram count to make the objective differentiable.

Specifically, we use $C_g(y)$ to denote the occurrence count of n-gram $g = (g_1, ..., g_n)$ in the target sentence y, and use $C_g^s(\theta)$ to denote the probabilistic count of n-gram g for the model with input x

and parameter θ, where the superscript s indicates SCTC. For simplicity, we omit the source sentence x in $C_g^s(\theta)$ and the following notations. The probabilistic n-gram count is obtained by accumulating n-grams from all possible target sentences:

$$C_g^s(\theta) = \sum_{y \in \mathcal{Y}} p(y|x,\theta) C_g(y) = \sum_{a \in \mathcal{Y}^*} p(a|x,\theta) C_g(\beta_s^{-1}(a)). \tag{9}$$

The calculation of $C_g^s(\theta)$ is the core of this method and will be described in detail later. We use $M_g^s(\theta)$ to denote the match count of n-gram g between the latent alignment model and the target sentence y (we omit y in $M_g^s(\theta)$ for simplicity), which is defined as follows:

$$M_g^s(\theta) = \min(C_g(y), C_g^s(\theta)). \tag{10}$$

Our objective is to maximize the n-gram overlap between the target sentence and model output, so we can train the model to maximize the precision or recall of n-gram matching:

$$P_n^s(\theta) = \frac{\sum_{g \in G_n} M_g^s(\theta)}{\sum_{g \in G_n} C_g^s(\theta)}, R_n^s(\theta) = \frac{\sum_{g \in G_n} M_g^s(\theta)}{\sum_{g \in G_n} C_g(y)}, \tag{11}$$

where we use G_n to denote the set of all n-grams. However, since the latent alignment model have the ability to control the translation length, both precision and recall cannot accurately reflect the translation quality. The precision will encourage short translations to reduce the denominator, while the recall prefers long translations. Therefore, we consider both of them and maximize the F1 score:

$$\mathcal{L}_n^s(\theta) = -\text{F1}_n^s(\theta) = -\frac{2 \cdot \sum_{g \in G_n} M_g^s(\theta)}{\sum_{g \in G_n} (C_g(y) + C_g^s(\theta))}, \tag{12}$$

where we use G_n to denote the set of all n-grams. In the numerator of Equation 12, $M_g^s(\theta)$ is non-zero only if $C_g(y)$ is non-zero, so we only need to calculate probabilistic n-gram counts $C_g^s(\theta)$ for n-grams g in the target sentence. In the denominator, $\sum_{g \in G_n} C_g(y)$ equals to the constant $T-n+1$. Therefore, there are only two tasks left: calculating $C_g^s(\theta)$ and the summation $\sum_{g \in G_n} C_g^s(\theta)$.

For the first task of calculating $C_g^s(\theta)$, we first consider the simple case $n=1$. The main difference between 1-gram matching and bipartite matching is that 1-gram matching is non-exclusive, which allows us to marginalize out all alignments using the property of non-autoregressive generation. Due to the space limit, we refer readers to Appendix.A for the derivation and directly give the probabilistic 1-gram count as follows:

$$C_g^s(\theta) = \sum_{t=1}^{T_a} p_t(g_1|x,\theta), g \in G_1, \tag{13}$$

where G_1 denotes the set of all 1-grams $g = (g_1)$. Both 1-gram matching and bipartite matching are completely non-monotonic, which aim at generating correct target words regardless of the word ordering. When the target sentence is "I ate pizza this afternoon", they will think that "pizza ate I this afternoon" is also a good translation. We consider 2-gram matching to overcome this limitation, which is also non-monotonic but takes target dependency into consideration. We first introduce a transition matrix A of size $T_a \times T_a$, where $A_{i,j}$ is the probability that all positions between i and j are blank tokens:

$$A_{i,j} = \begin{cases} 0 & j < i+1 \\ 1 & j = i+1 \\ \prod_{t=i+1}^{j-1} p_t(\epsilon|x,\theta) & j > i+1 \end{cases}. \tag{14}$$

Then we give the probabilistic 2-gram count for $g = (g_1, g_2)$ (see Appendix.A for derivation):

$$C_g^s(\theta) = \sum_{i=1}^{T_a} \sum_{j=1}^{T_a} p_i(g_1|x,\theta) A_{i,j} p_j(g_2|x,\theta), g \in G_2. \tag{15}$$

For a larger n, the probabilistic n-gram count is:

$$C_g^s(\theta) = \sum_{i_1=1}^{T_a} \sum_{i_2=1}^{T_a} ... \sum_{i_n=1}^{T_a} p_{i_1}(g_1|x,\theta) \times \prod_{k=2}^{n} A_{i_{k-1},i_k} p_{i_k}(g_k|x,\theta),\ g \in G_n. \tag{16}$$

Equation 15 can be efficiently calculated in matrix form with $\mathcal{O}(T_a^2)$ time complexity. For Equation 16, we need $\mathcal{O}(nT_a^n)$ operations to calculate it by brute force, which is impractical when n is large. We refer readers to Appendix.B for the $\mathcal{O}(nT_a^2)$ algorithm, which is similar to calculating the n-step transition probability of a Markov chain.

For the second task of efficiently calculating the summation of probabilistic n-gram counts $\sum_{g \in G_n} C_g^s(\theta)$, we also consider the simple case $n = 1$ first, where the summation can be calculated by directly summing all probabilities:

$$\sum_g C_g^s(\theta) = \sum_{t=1}^{T_a} \sum_g p_t(g_1|x, \theta), g \in G_1. \tag{17}$$

For $n > 1$, the search space is too big for the direct summation. Fortunately, the definition of $C_g^s(\theta)$ in Equation 9 allows us to safely extend the result of $n = 1$ to a larger n since n-grams count in a sentence is always $n - 1$ less than 1-grams count.

$$\sum_{g \in G_n} C_g^s(\theta) = \sum_{g \in G_1} C_g^s(\theta) - n + 1. \tag{18}$$

In summary, with the above results (Equation 13-18), we are able to efficiently calculate the loss defined in Equation 12 to maximize the F1 score non-monotonic n-gram matching, which reflects the translation quality to a certain extent.

3.3 Non-Monotonic Alignments under CTC

In this section, we explore non-monotonic latent alignments under the regular CTC loss. Under SCTC, we can use both the bipartite matching and n-gram matching to train the latent alignment model. However, bipartite matching is infeasible under the regular CTC loss. Recall that the search space $\gamma_s(y)$ is simply permutations of target words and $T_a - T$ blank tokens, which allows us to find the best alignment with the Hungarian algorithm. However, as consecutive repeated words will be collapsed in the regular CTC loss, alignments in $\gamma(y)$ may contain different words, so the best alignment cannot be found by the Hungarian algorithm.

Fortunately, for n-gram matching, we can utilize previous results of SCTC to help the analysis and derivation under the regular CTC loss. As the output of the collapsing function $\beta_s^{-1}(a)$ simply contains more repeated words than $\beta^{-1}(a)$, we can first identify how much influence these repeated words have, and then remove this influence from the results of SCTC. We first formally define the probabilistic n-gram count under the regular CTC:

$$C_g(\theta) = \sum_{a \in \mathcal{Y}^*} p(a|x, \theta) C_g(\beta^{-1}(a)). \tag{19}$$

We define $R_g(\theta)$ to be the gap bewteen $C_g(\theta)$ and $C_g^s(\theta)$ when $n = 1$, which represents the number of repeated words g_1 removed by the collapsing function. $R_g(\theta)$ can be efficiently calculated as follows (see Appendix.C for derivation):

$$R_g(\theta) = C_g^s(\theta) - C_g(\theta) = \sum_{t=1}^{T_a - 1} p_t(g_1|x, \theta) p_{t+1}(g_1|x, \theta), g \in G_1. \tag{20}$$

The above equation enables us to efficiently calculate $C_g(\theta)$ and $\sum_g C_g(\theta)$ when $g \in G_1$. For 2-grams $g = (g_1, g_2)$, we divide it into two cases $g_1 = g_2$ and $g_1 \neq g_2$. The collapsing of consecutive repeated words will only remove 2-grams of the first case, where each removed word corresponds to a removed 2-gram. Taking Figure 2 as an example, the output of SCTC $\beta_s^{-1}(a)$ have two more 2-grams $\{(A, A), (B, B)\}$ of the $g_1 = g_2$ case than the output of CTC $\beta^{-1}(a)$, and they contain exactly the same 2-grams of the $g_1 \neq g_2$ case. Therefore, we can directly reuse previous results of SCTC and the number of repeated words $R_g(\theta)$ to calculate the probabilistic 2-gram count:

$$C_g(\theta) = \begin{cases} C_g^s(\theta) - R_g(\theta), & g_1 = g_2 \\ C_g^s(\theta), & g_1 \neq g_2 \end{cases}, g \in G_2. \tag{21}$$

Unfortunately, for $n > 2$, there is no such a clear relationship between $C_g(\theta)$ and $C_g^s(\theta)$, where the CTC output can contain n-grams (for example, (A, B, C) in Figure 2) that the SCTC output does not

have. If we directly formulize $C_g(\theta)$ like in Equation 16, it will be more complex and the $\mathcal{O}(nT_a^2)$ algorithm is no longer applicable. Therefore, we do not use n-gram matching with $n > 2$ under the regular CTC loss. Finally, we define the match count $M_g(\theta)$ similarly as in Equation 10 and train the CTC-based latent alignment model to maximize the F1 score:

$$\mathcal{L}_n(\theta) = -\frac{2 \cdot \sum_{g \in G_n} M_g(\theta)}{\sum_{g \in G_n (C_g(y) + C_g(\theta))}}. \tag{22}$$

3.4 Training

In the above, we propose non-monotonic training objectives for the latent alignment model. They need to be combined with monotonic training objectives, otherwise, they may generate translations like "pizza ate I this afternoon". Therefore, we first pretrain the latent alignment model with monotonic training objectives like SCTC or CTC. The model pretrained by SCTC can be finetuned with the bipartite matching objective (Equation 8) or the n-gram matching objective (Equation 12). The model pretrained by CTC can be finetuned with the n-gram matching objective (Equation 22), and the choices of n are limited to $\{1, 2\}$.

4 Experiments

4.1 Experimental Setup

Datasets We evaluate our methods on the most widely used public benchmarks in previous NAT studies: WMT14 English↔German (En↔De, 4.5M sentence pairs) [5] and WMT16 English↔Romanian (En↔Ro, 0.6M sentence pairs) [6]. For WMT14 En↔De, the validation set is *newstest2013* and the test set is *newstest2014*. For WMT16 En↔Ro, the validation set is *newsdev-2016* and the test set is *newstest-2016*. Following prior works, we use tokenized BLEU [31] to evaluate the translation quality. Considering that BLEU might be biased as our method optimizes the n-gram overlap, we also report METEOR [2], which is not directly based on n-gram overlap. We learn a joint BPE model [36] with 32K operations to process the data and share the vocabulary for source and target languages.

Knowledge Distillation Following prior works [17], we apply sequence-level knowledge distillation [25] to reduce the complexity of training data. We use Transformer-base setting for the teacher model and train NAT on the distilled data.

Implementation Details We adopt Transformer-base [43] as our autoregressive baseline as well as the teacher model. We uniformly copy encoder outputs to construct decoder inputs. For CTC-based NAT models, the length for decoder inputs is $3\times$ as long as the source length. On WMT14 En↔De, we use a dropout rate of 0.2 to train NAT models and use a dropout rate of 0.1 for finetuning. On WMT16 En↔Ro, the dropout rate is 0.3 for both the pretraining and finetuning. We use the batch size 64K and train NAT models for 300K steps on WMT14 En↔De and 150K steps on WMT16 En↔Ro. During the finetuning, we train NAT models for 2K steps with the batch size 256K. All models are optimized with Adam [26] with $\beta = (0.9, 0.98)$ and $\epsilon = 10^{-8}$. The learning rate warms up to $5 \cdot 10^{-4}$ within 10K steps in the pretraining and warms up to $2 \cdot 10^{-4}$ within 500 steps in the finetuning. We use the GeForce RTX 3090 GPU to train models and measure the translation latency. We implement our models based on the open-source framework of `fairseq` [MIT License, 30].

Decoding For autoregressive models, we use beam search with beam size 5 for the decoding. For Vanilla-NAT, we use argmax decoding to generate the translation. For CTC-based models, we can also use argmax decoding to generate the alignment and then collapse it to obtain the translation. Besides, following [28, 16], we can use beam search decoding combined with a 4-gram language model [20] to find the translation that maximizes:

$$\log p(y|x, \theta) + \alpha \log p_{LM}(y) + \beta \log(|y|), \tag{23}$$

where α and β are hyperparameters for language model scores and length bonus. We use a fixed beam size 20 with grid-search α, β based on the performance on the validation set. The beam search does not contain any neural network computations and can be implemented efficiently in C++[1].

[1] https://github.com/parlance/ctcdecode

4.2 Main Results

We first compare the performance of different non-monotonic training objectives for latent alignment models. Baseline models include the Transformer, vanilla-NAT, SCTC, and CTC, and we finetune latent alignment models with bipartite matching and n-gram matching objectives. In Table 1, we report BLEU and METEOR scores along with the speedup on WMT14 En-De test set.

Table 1: Performance comparison between baselines and our methods on WMT14 En-De test set.

	Model	BLEU	METEOR	Speed
Base	Transformer	27.54	54.38	1.0×
	Vanilla-NAT	19.32	45.79	15.5×
SCTC	SCTC	25.26	52.06	14.7×
	+bipartite	26.13	53.17	
	+1-gram	26.22	53.20	
	+2-gram	**26.79**	53.54	
	+3-gram	26.66	53.48	
	+4-gram	26.62	53.43	
CTC	CTC	26.34	53.15	14.7×
	+1-gram	27.16	53.95	
	+2-gram	**27.57**	**54.50**	

Table 2: Performance comparison between our models and existing methods. The speedup is measured on WMT14 En-De test set with batch size 1. AT means autoregressive. **Iter.** denotes the number of iterations at inference time. '–' indicates that the result is not reported. '12-1' means the Transformer with 12 encoder layers and 1 decoder layer [23]. 'beam&lm' means beam search decoding combined with a 4-gram language model.

Models		Iter.	Speed	WMT14		WMT16	
				EN-DE	DE-EN	EN-RO	RO-EN
AT	Transformer (teacher)	N	1.0×	27.54	**31.57**	**34.26**	**33.87**
	Transformer (12-1)	N	2.6×	26.09	30.30	32.76	32.39
	+ KD	N	2.7×	**27.61**	31.48	33.43	33.50
Non-CTC NAT	NAT-FT [17]	1	15.6×	17.69	21.47	27.29	29.06
	AXE [13]	1	–	23.53	27.90	30.75	31.54
	GLAT [32]	1	15.3×	25.21	29.84	31.19	32.04
	AligNART [41]	1	13.4×	26.40	30.40	32.50	33.10
	CMLM [12]	10	–	27.03	30.53	33.08	33.31
	LevT [18]	2.05	4.0×	27.27	–	–	33.26
	JM-NAT [19]	10	–	27.69	**32.24**	33.52	33.72
	RewriteNAT [11]	2.70	–	**27.83**	31.52	**33.63**	**34.09**
CTC-based NAT	CTC [28]	1	–	16.56	18.64	19.54	24.67
	Imputer [35]	1	–	25.80	28.40	32.30	31.70
	REDER [45]	1	15.5×	26.70	30.68	33.10	33.23
	+ beam&reranking	1	5.5×	27.36	31.10	33.60	34.03
	Fully-NAT [16]	1	16.8×	27.20	31.39	33.71	**34.16**
	Imputer [35]	8	–	**28.20**	**31.80**	**34.40**	34.10
Our Implementation	Bag-of-ngrams [38]	1	15.5×	25.28	29.66	31.37	31.51
	+ rescore 5 candidates	1	9.5×	25.95	30.43	32.78	32.92
	OAXE [9]	1	15.3×	24.70	29.30	31.06	31.37
	+ rescore 5 candidates	1	14.0 ×	25.78	30.29	32.45	32.63
	GLAT+CTC [32]	1	14.7×	26.71	29.93	32.85	33.12
	+ beam&lm	1	5.0×	**27.58**	**31.25**	**33.59**	**33.98**
Our work	Vanilla-NAT	1	15.5×	19.32	24.23	29.03	29.46
	CTC	1	14.7×	26.34	29.58	33.45	33.32
	NMLA	1	14.7×	27.57	31.28	33.86	33.94
	NMLA + beam&lm	1	5.0×	**28.35**	**32.27**	**34.72**	**34.95**

From Table 1, we have the following observations: (1) Non-monotonic training objectives can effectively improve the performance of latent alignment models, which improves SCTC by 1.53 BLEU and 1.48 METEOR and improves CTC by 1.23 BLEU and 1.35 METEOR, illustrating the importance of non-monotonic alignments. (2) Despite having the same expressive power, SCTC underperforms CTC in the task of non-autoregressive translation. We speculate that it is because of the difference in the distribution of the alignment space. Most target sentences do not contain repeated words, in which case the alignment space of SCTC $\beta_s^{-1}(y)$ is a subset of $\beta^{-1}(y)$, making the ability of SCTC relatively weak. (3) SCTC achieves the best performance with 2-gram finetuning, and then the performance decreases with the increase of n due to the sparsity of higher rank n-grams. It suggests that we do not need a large n and relieves the concern on CTC that we cannot calculate n-gram matching for $n > 2$. (4) The correlation between the reported BLEU and METEOR scores is very strong, suggesting that BLEU is not biased when evaluating our methods. Therefore, we can safely use BLEU to evaluate the translation quality in the following experiments.

We call the CTC+2-gram method NMLA since it represents our best non-monotonic training method for latent alignment models. We report the performance of NMLA comparing with existing approaches in Table 2. Notably, NMLA achieves comparable performance to autoregressive Transformer on all benchmarks, demonstrating the effectiveness of our method. With the help of beam search and 4-gram language model, NMLA even outperforms other iterative NAT models and the autoregressive Transformer with only one-iteration parallel decoding. Compared to our implementations of prior works that explore non-monotonic matching for NAT [38, 9], we can see that they underperform the CTC baseline even when rescoring 5 candidates, and their speedup after rescoring also lays behind CTC. It demonstrates the importance of exploring non-monotonic matching on latent alignment models, which is more desirable for its flexibility of generating predictions with variable lengths.

5 Related Work

[17] first proposes non-autoregressive machine translation to accelerate the decoding process of NMT. The cross-entropy loss is applied to train the Vanilla-NAT, which requires the explicit alignment between the model output and target sentence. The explicit alignment cannot be guaranteed due to the multi-modality problem, and many efforts have been devoted to mitigating this issue.

One research direction is to enable explicit alignment by leaking part of the target-side information during the training. [17] uses fertilities as a latent variable to specify the number of output words corresponding to each input word. [22, 34, 4] apply vector quantization to train NAT with discrete latent variables, and [29, 40, 16] apply variational inference to train NAT with continuous latent variables. [33, 3, 41] also introduce latent variables that carry the reordering information. Besides using latent variables, [12] directly feeds the partially masked target sentence as the decoder input, and [32] applies a glancing sampling strategy to control the mask ratio.

Another research direction is to propose better training objectives for NAT. [44] introduces two auxiliary regularization terms to reduce errors of repeated translations and incomplete translations. [21] trains NAT with deep supervision and feed additional layer-wise predictions. Prior to this work, [38, 13, 9] have explored ways to establish an alignment between the model output and target sentence, which can promote the training of NAT. Their limitation is that they are basically built on the framework of Vanilla-NAT, which requires a target length predictor and cannot dynamically adjust the output sequence length. CTC-based latent alignment models [28, 35, 16] are more desirable due to their superior performance and the flexibility of generating predictions with variable lengths. However, latent alignment models have a weakness in handling non-monotonic alignments, which is noticed in previous work [35, 16] and solved in this work.

6 Conclusion

Latent alignment models cannot handle non-monotonic alignments during the training, which is nonnegligible in machine translation. In this paper, we explore non-monotonic latent alignments for NAT and propose two matching objectives named bipartite matching and n-gram matching. Experiments show that our method achieves strong improvements on latent alignment models.

References

[1] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1409.0473.

[2] S. Banerjee and A. Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. URL https://aclanthology.org/W05-0909.

[3] Y. Bao, H. Zhou, J. Feng, M. Wang, S. Huang, J. Chen, and L. Lei. Non-autoregressive transformer by position learning. *arXiv preprint arXiv:1911.10677*, 2019.

[4] Y. Bao, S. Huang, T. Xiao, D. Wang, X. Dai, and J. Chen. Non-autoregressive translation by learning target categorical codes. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5749–5759, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.458. URL https://aclanthology.org/2021.naacl-main.458.

[5] O. Bojar, C. Buck, C. Federmann, B. Haddow, P. Koehn, J. Leveling, C. Monz, P. Pecina, M. Post, H. Saint-Amand, R. Soricut, L. Specia, and A. Tamchyna. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA, June 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W14/W14-3302.

[6] O. Bojar, R. Chatterjee, C. Federmann, Y. Graham, B. Haddow, M. Huck, A. Jimeno Yepes, P. Koehn, V. Logacheva, C. Monz, M. Negri, A. Neveol, M. Neves, M. Popel, M. Post, R. Rubino, C. Scarton, L. Specia, M. Turchi, K. Verspoor, and M. Zampieri. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 131–198, Berlin, Germany, August 2016. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W16/W16-2301.

[7] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer, 2020.

[8] S. Castilho, J. Moorkens, F. Gaspari, I. Calixto, J. Tinsley, and A. Way. Is neural machine translation the new state of the art? *Prague Bull. Math. Linguistics*, 108:109–120, 2017. URL http://ufal.mff.cuni.cz/pbml/108/art-castilho-moorkens-gaspari-tinsley-calixto-way.pdf.

[9] C. Du, Z. Tu, and J. Jiang. Order-agnostic cross entropy for non-autoregressive machine translation. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 2849–2859. PMLR, 2021. URL http://proceedings.mlr.press/v139/du21c.html.

[10] M. Fomicheva and L. Specia. Taking MT Evaluation Metrics to Extremes: Beyond Correlation with Human Judgments. *Computational Linguistics*, 45(3):515–558, 09 2019. ISSN 0891-2017. doi: 10.1162/coli_a_00356. URL https://doi.org/10.1162/coli_a_00356.

[11] X. Geng, X. Feng, and B. Qin. Learning to rewrite for non-autoregressive neural machine translation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3297–3308, Online and Punta Cana, Dominican Republic, Nov. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.265. URL https://aclanthology.org/2021.emnlp-main.265.

[12] M. Ghazvininejad, O. Levy, Y. Liu, and L. Zettlemoyer. 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, 2019. URL https://www.aclweb.org/anthology/D19-1633.

[13] M. Ghazvininejad, V. Karpukhin, L. Zettlemoyer, and O. Levy. Aligned cross entropy for non-autoregressive machine translation. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 3515–3523. PMLR, 2020. URL http://proceedings.mlr.press/v119/ghazvininejad20a.html.

[14] A. Graves. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*, 2012.

[15] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, page 369–376, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595933832. doi: 10.1145/1143844.1143891. URL https://doi.org/10.1145/1143844.1143891.

[16] J. Gu and X. Kong. Fully non-autoregressive neural machine translation: Tricks of the trade, 2020.

[17] J. Gu, J. Bradbury, C. Xiong, V. O. K. Li, and R. Socher. Non-autoregressive neural machine translation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018. URL https://openreview.net/forum?id=B1l8BtlCb.

[18] J. Gu, C. Wang, and J. Zhao. Levenshtein transformer. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 11179–11189, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/675f9820626f5bc0afb47b57890b466e-Abstract.html.

[19] J. Guo, L. Xu, and E. Chen. Jointly masked sequence-to-sequence model for non-autoregressive neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 376–385, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.36. URL https://aclanthology.org/2020.acl-main.36.

[20] K. Heafield. KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, July 2011. Association for Computational Linguistics. URL https://aclanthology.org/W11-2123.

[21] C. Huang, H. Zhou, O. R. Zaiane, L. Mou, and L. Li. Non-autoregressive translation with layer-wise prediction and deep supervision. *ArXiv*, abs/2110.07515, 2021.

[22] L. Kaiser, S. Bengio, A. Roy, A. Vaswani, N. Parmar, J. Uszkoreit, and N. Shazeer. Fast decoding in sequence models using discrete latent variables. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2390–2399. PMLR, 10–15 Jul 2018. URL http://proceedings.mlr.press/v80/kaiser18a.html.

[23] J. Kasai, N. Pappas, H. Peng, J. Cross, and N. Smith. Deep encoder, shallow decoder: Reevaluating non-autoregressive machine translation. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=KpfasTaLUpq.

[24] Z. Kasner, J. Libovický, and J. Helcl. Improving fluency of non-autoregressive machine translation, 2020.

[25] Y. Kim and A. M. Rush. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, Austin, Texas, Nov. 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1139. URL https://www.aclweb.org/anthology/D16-1139.

[26] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[27] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955. doi: https://doi.org/10.1002/nav.3800020109. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.3800020109.

[28] J. Libovický and J. Helcl. End-to-end non-autoregressive neural machine translation with connectionist temporal classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3016–3021, Brussels, Belgium, Oct.-Nov. 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1336. URL https://www.aclweb.org/anthology/D18-1336.

[29] X. Ma, C. Zhou, X. Li, G. Neubig, and E. Hovy. 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, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1437. URL https://www.aclweb.org/anthology/D19-1437.

[30] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

[31] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://www.aclweb.org/anthology/P02-1040.

[32] L. Qian, H. Zhou, Y. Bao, M. Wang, L. Qiu, W. Zhang, Y. Yu, and L. Li. Glancing transformer for non-autoregressive neural machine translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1993–2003, Online, Aug. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.155. URL https://aclanthology.org/2021.acl-long.155.

[33] Q. Ran, Y. Lin, P. Li, and J. Zhou. Guiding non-autoregressive neural machine translation decoding with reordering information. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Virtual Event, February 2-9, 2021*, pages 13727–13735. AAAI Press, 2021. URL https://ojs.aaai.org/index.php/AAAI/article/view/17618.

[34] A. Roy, A. Vaswani, A. Neelakantan, and N. Parmar. Theory and experiments on vector quantized autoencoders. *arXiv preprint arXiv:1805.11063*, 2018.

[35] C. Saharia, W. Chan, S. Saxena, and M. Norouzi. Non-autoregressive machine translation with latent alignments. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1098–1108, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.83. URL https://www.aclweb.org/anthology/2020.emnlp-main.83.

[36] R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. In *ACL*, August 2016. URL http://www.aclweb.org/anthology/P16-1162.

[37] C. Shao, X. Chen, and Y. Feng. Greedy search with probabilistic n-gram matching for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4778–4784, Brussels, Belgium, Oct.-Nov. 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1510. URL https://www.aclweb.org/anthology/D18-1510.

[38] C. Shao, J. Zhang, Y. Feng, F. Meng, and J. Zhou. Minimizing the bag-of-ngrams difference for non-autoregressive neural machine translation. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 198–205. AAAI Press, 2020. URL `https://aaai.org/ojs/index.php/AAAI/article/view/5351`.

[39] C. Shao, Y. Feng, J. Zhang, F. Meng, and J. Zhou. Sequence-Level Training for Non-Autoregressive Neural Machine Translation. *Computational Linguistics*, pages 1–35, 10 2021. ISSN 0891-2017. doi: 10.1162/coli_a_00421. URL `https://doi.org/10.1162/coli_a_00421`.

[40] R. Shu, J. Lee, H. Nakayama, and K. Cho. Latent-variable non-autoregressive neural machine translation with deterministic inference using a delta posterior. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8846–8853. AAAI Press, 2020. URL `https://aaai.org/ojs/index.php/AAAI/article/view/6413`.

[41] J. Song, S. Kim, and S. Yoon. AligNART: Non-autoregressive neural machine translation by jointly learning to estimate alignment and translate. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1–14, Online and Punta Cana, Dominican Republic, Nov. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.1. URL `https://aclanthology.org/2021.emnlp-main.1`.

[42] R. Stewart, M. Andriluka, and A. Ng. End-to-end people detection in crowded scenes. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2325–2333, 2016.

[43] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, u. Kaiser, and I. Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

[44] Y. Wang, F. Tian, D. He, T. Qin, C. Zhai, and T. Liu. Non-autoregressive machine translation with auxiliary regularization. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, Honolulu, Hawaii, USA, 2019*, pages 5377–5384. AAAI Press, 2019. doi: 10.1609/aaai.v33i01.33015377. URL `https://doi.org/10.1609/aaai.v33i01.33015377`.

[45] Z. Zheng, H. Zhou, S. Huang, J. Chen, J. Xu, and L. Li. Duplex sequence-to-sequence learning for reversible machine translation, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] Line 188-194, line 210-214, Appendix.E.
   (c) Did you discuss any potential negative societal impacts of your work? [No] We did not find any negative societal impacts.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] In Appendix.A, Appendix.B, Appendix.C.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] In the supplemental material.

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Section 4.

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] It would be too computationally expensive.

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] In Appendix.E.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes] We cite the WMT dataset and codebase fairseq.

(b) Did you mention the license of the assets? [Yes] line 247.

(c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We provide our code in the supplemental material.

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] Line 226, they are public benchmarks.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] They are widely used benchmarks, so we did not check.

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Proof of Equation 13, Equation 15, and Equation 16

Equation 13 and and Equation 15 are the special cases of Equation 16 when n is 1 or 2, so we directly provide the proof of Equation 16:

$$C_g^s(\theta) = \sum_{i_1=1}^{T_a} \sum_{i_2=1}^{T_a} \dots \sum_{i_n=1}^{T_a} p_{i_1}(g_1|x,\theta) \times \prod_{k=2}^{n} A_{i_{k-1},i_k} p_{i_k}(g_k|x,\theta), g \in G_n.$$

We begin with the definition of $C_g^s(\theta)$, which accumulates the n-gram g from all alignments:

$$C_g^s(\theta) = \sum_{a \in \mathcal{Y}^*} p(a|x,\theta) C_g(\beta_s^{-1}(a)),$$

where $C_g(\beta_s^{-1}(a))$ is the occurrence count of n-gram g in the collapsed sentence $\beta_s^{-1}(a)$. We use $i_{1:n} = \{i_1, ..., i_n\}$ to denote the possible position of $g = \{g_1, ..., g_n\} \in G_n$. There is an n-gram $g \in G_n$ in the position $i_{1:n}$ if and only if $a_{i_k} = g_k$ for all $k \in \{1, ..., n\}$ and all other words between them are blank tokens. By traversing all possible positions, we can obtain the n-gram count as follows:

$$C_g(\beta_s^{-1}(a)) = \sum_{i_1=1}^{T_a} \sum_{i_2=1}^{T_a} \dots \sum_{i_n=1}^{T_a} 1(a_{i_1} = g_1) \times \prod_{k=2}^{n} 1(i_{k-1} < i_k, a_{i_k} = g_k, a_{i_{k-1}+1:i_k-1} = \epsilon), g \in G_n.$$

where $1(\cdot)$ is the indicator function that takes value 1 when the inside condition holds, otherwise it takes value 0. Recall that A_{ij} is the probability that all positions between i and j are blank tokens, which satisfies the following equation:

$$A_{i_{k-1},i_k} = p(i_{k-1} < i_k, a_{i_{k-1}+1:i_k-1} = \epsilon|x,\theta).$$

Using the above equations, we can rewrite $C_g^s(\theta)$ as:

$$\begin{aligned}
C_g^s(\theta) &= \sum_a p(a|x,\theta) C_g(\beta_s^{-1}(a)) \\
&= \sum_{i_1=1}^{T_a} \sum_{i_2=1}^{T_a} \dots \sum_{i_n=1}^{T_a} \sum_a p(a|x,\theta) 1(a_{i_1} = g_1) \times \prod_{k=2}^{n} 1(i_{k-1} < i_k, a_{i_k} = g_k, a_{i_{k-1}+1:i_k-1} = \epsilon) \\
&= \sum_{i_1=1}^{T_a} \sum_{i_2=1}^{T_a} \dots \sum_{i_n=1}^{T_a} p(a_{i_1} = g_1|x,\theta) \times \prod_{k=2}^{n} p(a_{i_k} = g_k|x,\theta) p(i_{k-1} < i_k, a_{i_{k-1}+1:i_k-1} = \epsilon) \\
&= \sum_{i_1=1}^{T_a} \sum_{i_2=1}^{T_a} \dots \sum_{i_n=1}^{T_a} p_{i_1}(g_1|x,\theta) \times \prod_{k=2}^{n} A_{i_{k-1},i_k} p_{i_k}(g_k|x,\theta), g \in G_n. \ \square
\end{aligned}$$

B The $\mathcal{O}(nT_a^2)$ algorithm for Equation 16

Here we give the $\mathcal{O}(nT_a^2)$ algorithm to calculate $C_g^s(\theta)$ according to Equation 16. Notice that Equation 16 has a similar form with the n-step transition probability of a Markov chain, so we can try a similar method as the calculating the transition probability for a Markov chain. Specifically, we introduce a set of state vectors s_k , where $k \in \{1, 2, ..., T_a\}$ and the dimension of each vector s_k is T_a . We define the state vectors as follows:

$$s_k(i_k) = \begin{cases} p_{i_k}(g_k|x,\theta), & k = 1 \\ \sum_{i_{k-1}=1}^{T_a} s_{k-1}(i_{k-1}) A_{i_{k-1},i_k} p_{i_k}(g_k|x,\theta), & k > 1 \end{cases}.$$

From the above equation, the time complexity to calculate s_k given s_{k-1} is $\mathcal{O}(T_a^2)$ and can be efficiently calculated in matrix form. Therefore, the time complexity to sequentially calculate $s_{1:n}$ is $\mathcal{O}(nT_a^2)$. Finally, we can easily obtain $C_g^s(\theta)$ by summing the last vector s_n:

$$
\begin{aligned}
C_g^s(\theta) &= \sum_{i_1=1}^{T_a}\sum_{i_2=1}^{T_a}\dots\sum_{i_n=1}^{T_a} p_{i_1}(g_1|x,\theta) \times \prod_{k=2}^{n} A_{i_{k-1},i_k} p_{i_k}(g_k|x,\theta) \\
&= \sum_{i_n=1}^{T_a}\dots\sum_{i_2=1}^{T_a}\sum_{i_1=1}^{T_a} s_1(i_1) \prod_{k=2}^{n} A_{i_{k-1},i_k} p_{i_k}(g_k|x,\theta) \\
&= \sum_{i_n=1}^{T_a}\dots\sum_{i_3=1}^{T_a}\sum_{i_2=1}^{T_a} s_2(i_2) \prod_{k=3}^{n} A_{i_{k-1},i_k} p_{i_k}(g_k|x,\theta) \\
&= \dots\dots \\
&= \sum_{i_n=1}^{T_a}\sum_{i_{n-1}=1}^{T_a} s_{n-1}(i_{n-1}) \times A_{i_{n-1},i_n} p_{i_n}(g_n|x,\theta) \\
&= \sum_{i_n=1}^{T_a} s_n(i_n), g \in G_n. \quad \square
\end{aligned}
$$

C Proof of Equation 20

Here we provide the proof of Equation 20:

$$
R_g(\theta) = C_g^s(\theta) - C_g(\theta) = \sum_{t=1}^{T_a-1} p_t(g_1|x,\theta) p_{t+1}(g_1|x,\theta), g \in G_1,
$$

which is equivalent to proving the following equation:

$$
C_g(\theta) = p_1(g_1|x,\theta) + \sum_{t=2}^{T_a} p_t(g_1|x,\theta)(1 - p_{t-1}(g_1|x,\theta)), g \in G_1.
$$

We begin with the definition of $C_g(\theta)$, which accumulates the n-gram g from all alignments:

$$
C_g(\theta) = \sum_{a \in \mathcal{Y}^*} p(a|x,\theta) C_g(\beta^{-1}(a)).
$$

$C_g(\beta^{-1}(a))$ is the occurrence count of n-gram g in the collapsed sentence $\beta^{-1}(a)$. Besides blank tokens, repeated words will also be removed by β^{-1}. Therefore, we call a word in the alignment a valid word when it does not repeat the word before it. We can calculate the 1-gram count by collecting all valid words, which will not be removed by the collapsing function:

$$
C_g(\beta^{-1}(a)) = 1(a_1 = g_1) + \sum_{t=2}^{T_a} 1(a_t = g_1, a_{t-1} \neq g_1), g \in G_1.
$$

Using the above equation, we can rewrite $C_g(\theta)$ as:

$$
\begin{aligned}
C_g(\theta) &= \sum_a p(a|x,\theta) C_g(\beta^{-1}(a)) = p(a_1 = g_1|x,\theta) + \sum_{t=2}^{T_a} p(a_t = g_1, a_{t-1} \neq g_1|x,\theta) \\
&= p_1(g_1|x,\theta) + \sum_{t=2}^{T_a} p_t(g_1|x,\theta)(1 - p_{t-1}(g_1|x,\theta)), g \in G_1. \quad \square
\end{aligned}
$$

D Analysis

To better understand our method, we present a qualitative analysis to provide some insights on where modeling non-monotonic latent alignments improves over the CTC baseline. We analyze the generated outputs of the WMT14 En-De test set.

NMLA Handles Long Sequences Better We first investigate the performance of AT and NAT models for different sequence lengths. We split the test set into different buckets based on target sequence length and calculate BLEU for each bucket. We report the results in Table 3. The first observation is that the performance of Vanilla-NAT drops drastically as the sequence length increases. It is a well known property of NAT and can be explained as the increasing probability of observing misalignment between model output and target as the target sequence becomes longer. The misalignment in long sequences causes the inaccuracy of loss and therefore harms the translation quality. We have a similar observation when comparing CTC with NMLA. The performance of CTC is close to NMLA on short sequences, but the gap becomes larger as the sequence length increases. Our explanation is that the longer the sequence, the more likely the monotonic assumption fails due to the global word reordering, which causes the inaccuracy of loss and therefore harms the translation quality. NMLA models non-monotonic alignments and alleviates the inaccuracy of loss on long sequences, so it achieves strong improvements on long sequences and even outperforms AT by more than 2 BLEU, which suffers from the error accumulation on long sequences.

Table 3: The performance of AT and NAT models with respect to the target sequence length. 'Vanilla' means Vanilla-NAT. 'N' denotes the target sequence length. 'AT' means autoregressive Transformer.

Length	**Vanilla**	**CTC**	**AT**	**NMLA**
$1 \leq N < 20$	21.27	24.83	25.68	25.55
$20 \leq N < 40$	19.49	26.81	28.05	27.82
$40 \leq N < 60$	16.21	26.54	26.71	27.74
$60 \leq N$	10.69	26.69	26.44	28.89
All	19.32	26.34	27.54	27.57

NMLA Improves Model Confidence NAT suffers from the multi-modality problem, which makes the model consider many possible translations at the same time and become less confident in generating outputs. We investigate whether our method alleviates the multi-modality problem by measuring how confident the model is during the decoding. We use the information entropy to measure the confidence, which is calculated as $H(X) = -\sum_{x \in \mathcal{X}} p(x) \log p(x)$, and lower entropy indicates higher confidence. We calculate the entropy in every position when decoding the WMT14 En-De test set and use the average entropy to represent the model confidence. We do not report the entropy of AT since it varies with the decoding algorithm. The average entropy scores of NAT models are reported in Table 4. We can see that the Vanilla-NAT has low prediction confidence, which is substantially improved by CTC. NMLA further reduces the average entropy to 0.061, showing that the model confidence can be improved by modeling non-monotonic latent alignments.

Table 4: The average prediction entropy of NAT models. 'Vanilla' means Vanilla-NAT.

	Vanilla	**CTC**	**NMLA**
Entropy	2.098	0.260	0.061

NMLA Improves Generation Fluency Since NAT generates target words independently, generation fluency is a main weakness of NAT. The multi-modality problem makes NAT consider many possible translations at the same time and generate an inconsistent output, which further reduces the generation fluency. We investigate whether our method improves the generation fluency with an n-gram language model [20] trained on the target-side of WMT14 En-De training data. We use the language model to calculate the perplexity scores of model outputs and report the results in Table 5. We can see that Vanilla-NAT and CTC have high perplexity that indicates low fluency. NMLA substantially reduces the perplexity score, which is only 42.6 higher than the gold reference. For AT, its perplexity score is even lower than the gold reference, which is consistent with prior works that autoregressive NMT generally improves fluency at the cost of adequacy [8, 10].

Table 5: The perplexity scores of AT and NAT models on WMT14 En-De test set.

	Vanilla	**CTC**	**NMLA**	**AT**	**Gold**
PPL	597.0	315.5	258.8	197.0	216.2

E Training Cost

Our method requires to pretrain the CTC model first and then finetune it with non-monotonic training objectives, so more training time is required. We may also use a larger batch size for the training or finetuning to improve model performance at the cost of more training time. In Table 6, we use different batch sizes and report the BLEU scores along with the training time on WMT14 En-De. We can see that NMLA improves the CTC baseline by 1.23 BLEU with only $1.28\times$ training cost, which cannot be achieved by simply increasing the batch size for the training or finetuning.

Table 6: BLEU scores of different models on WMT14 En-De test set along with the training time measured on 8 GeForce RTX 3090 GPUs.

Model	BLEU	Time
64K GLAT+CTC [32]	26.39	–
128K Fully-NAT [16]	27.20	–
128K CTC	26.59	52.8h
64K CTC	26.34	26.4h
+512K CTC-Tune	26.56	27.8h
+64K Bigram-Tune	27.32	28.3h
+256K Bigram-Tune	27.57	33.9h

F Finetuning Steps

To illustrate the impact of the finetuning steps, we use the bigram matching objective to finetune SCTC and CTC baselines, and draw their BLEU curves on WMT14 En-De validation set in Figure 3. For both SCTC and CTC, their BLEU scores rise rapidly in the first 500 steps, and then the curves become more steady. After 2000 steps, continuing finetuning can barely bring a performance gain, so it is cost-effective to finetune the baseline for only 2000 steps.

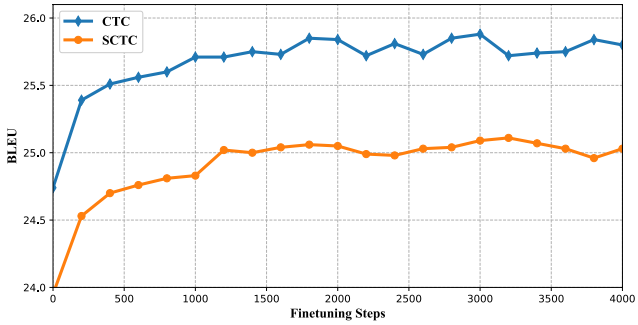


Figure 3: BLEU scores of CTC and SCTC under different finetuning steps of bigram matching.

G Combination of N-grams

In Table 1, we only finetune the baseline with different n-gram matching objectives respectively. Here we combine diffent n-gram matching objectives to see whether the combination can improve the model performance. First, we linearly combine 1-gram and 2-gram matching with a hyperparameter α:

$$\mathcal{L}(\theta) = \alpha \cdot \mathcal{L}_1(\theta) + (1-\alpha) \cdot \mathcal{L}_2(\theta), \quad \mathcal{L}^s(\theta) = \alpha \cdot \mathcal{L}_1^s(\theta) + (1-\alpha) \cdot \mathcal{L}_2^s(\theta).$$

We train CTC and SCTC baselines with different α and report the results in Table 7, which shows that the combination of 1-gram and 2-gram underperforms simply using 2-gram matching.

As $n > 2$ is allowed in SCTC, we further try to combine higher rank n-grams. We consider $n \in \{1, 2, 3, 4\}$ and combine them with arithmetic average and geometric average respectively. The

Table 7: BLEU scores of SCTC and CTC under different α on WMT14 En-De validation set.

α	0	0.25	0.5	0.75	1
SCTC	25.09	24.98	24.85	24.70	24.54
CTC	25.90	25.74	25.66	25.51	25.42

BLEU scores of arithmetic average and geometric average are respectively 24.94 BLEU and 24.88 BLEU on WMT14 En-De validation set, which also underperforms simply using 2-gram matching. Therefore, we conclude that simply combining different n-gram matching objectives cannot improve the model performance.

H Performance on Stronger Baseline

In this section, we finetune Fully-NAT [16] with the NMLA objective to show whether our method is still effective on strong baselines. We reimplement Fully-NAT following their settings and finetune it with the NMLA objective. As Table 8 shows, our method achieves consider improvement on Fully-NAT, demonstrating the effectiveness of our method on strong baselines.

Table 8: BLEU scores on WMT14 En-De test set.

Model	CTC	Fully-NAT	Fully-NAT + NMLA
BLEU	26.34	27.07	27.75