Non-autoregressive End-to-end Speech Translation with Parallel Autoregressive Rescoring

Hirofumi Inaguma, Student Member, IEEE, Yosuke Higuchi, Student Member, IEEE, Kevin Duh, Member, IEEE Tatsuya Kawahara, Fellow, IEEE and Shinji Watanabe, Senior Member, IEEE

Abstract—This article describes an efficient end-to-end speech translation (E2E-ST) framework based on non-autoregressive (NAR) models. End-to-end speech translation models have several advantages over traditional cascade systems such as inference latency reduction. However, conventional AR decoding methods are not fast enough because each token is generated incrementally. NAR models, however, can accelerate the decoding speed by generating multiple tokens in parallel on the basis of the token-wise conditional independence assumption. We propose a unified NAR E2E-ST framework called Orthros, which has an NAR decoder and an auxiliary shallow AR decoder on top of the shared encoder. The auxiliary shallow AR decoder selects the best hypothesis by rescoring multiple candidates generated from the NAR decoder in parallel (parallel AR rescoring). We adopt conditional masked language model (CMLM) and a connectionist temporal classification (CTC)-based model as NAR decoders for Orthros, referred to as Orthros-CMLM and Orthros-CTC, respectively. We also propose two training methods to enhance the CMLM decoder. Experimental evaluations on three benchmark datasets with six language directions demonstrated that Orthros achieved large improvements in translation quality with a very small overhead compared with the baseline NAR model. Moreover, the Conformer encoder architecture enabled large quality improvements, especially for CTC-based models. Orthros-CTC with the Conformer encoder increased decoding speed by 3.63× on CPU with translation quality comparable to that of an AR model.

Index Terms—End-to-end speech translation, non-autoregressive decoding, rescoring

I. INTRODUCTION

BREAKING language barriers by using machines is an ultimate goal for international communications. Automatic speech translation (ST) has been studied for this purpose for decades [1]–[4]. Cascade approaches combining automatic speech recognition (ASR) and machine translation (MT) systems have been the de facto standard, but the research paradigm is shifting to end-to-end speech translation (E2E-ST) models thanks to advances in deep learning [5]–[7]. End-to-end models have several attractive properties, such as avoiding ASR error propagation and low-latency decoding. However, the translation quality of E2E models still lags behind that of cascade systems when additional resources are available, although the gap is closing [8]–[10]. Various solutions have been proposed to bridge the quality gap between cascade and E2E models, such as multi-task learning with auxiliary tasks [6], [7], pre-training [7], [11]–[13], knowledge distillation [14], [15], and semi-supervised training [16], [17]. Most studies on E2E systems have focused on autoregressive (AR) models, which generate a target sequence from left to right. However, the decoding speed is not fast enough for real-world applications because of the incremental update of decoder states. This slows the decoding speed, especially when the Transformer architecture [18] is adopted because of the self-attention operation to all past tokens in the decoder at each generation step.

Recently, non-AR (NAR) models have attracted attention to increase the decoding speed by discarding the conditional dependency of outputs in AR models. Gu et al. [19] proposed the first single-step NAR MT model, but it sacrifices translation quality. Various models have been proposed to address this issue; improved single-step NAR models [20]–[28], iterative-refinement models [29]–[32], latent alignment models [33]–[36], insertion-based models [37]–[39], and latent variable models [40]–[43]. NAR models have been successfully extended to other tasks such as text-to-speech synthesis [44], [45], ASR [46]–[48] and ST [49], [50].

However, what differentiates the NAR ST task from other NAR tasks is many-to-many non-monotonic mapping. In other words, source inputs, even though they correspond to the same word sequence, can vary significantly depending on speaker attributes, speaking rate, recording conditions and so on. These problems do not exist in text-based tasks because discrete tokens represent the input instead of continuous signals. It is also challenging to determine the target length from the speech in advance because it includes many silent frames and the input length is much longer than that of text.

To increase the decoding speed for the E2E-ST task, we propose an efficient unified NAR framework called Orthros, which introduces an auxiliary shallow AR decoder on top of the shared speech encoder (see Fig. 1). The AR decoder is jointly trained with the NAR decoder and is used to rescore multiple candidates of different lengths generated from the NAR decoder, referred to as parallel AR rescoring. This is based on an observation that sequence-level scores from the NAR decoder are not sufficient for accurately selecting the best candidate. Because outputs from the NAR decoder can be fed to the AR decoder in parallel, the NAR decoding is still maintained. Because a shallow AR decoder works for
the rescoring purpose only, additional computation cost is minimal.

We investigate the conditional masked language model (CMLM) \cite{30} and connectionist temporal classification (CTC)-based model \cite{51} as NAR decoders for Orthros, referred to as Orthros-CMLM and Orthros-CTC, respectively. The CMLM is an iterative refinement model while the CTC-based model is a single-step latent alignment model. However, any other NAR decoder topology can be used for Orthros in theory as long as multiple candidates can be generated. To enhance the training of the CMLM decoder, we further propose multi-mask training (MMT) and joint training with an auxiliary text-input NAR MT task. With these methods, the CMLM decoder can take decoder inputs from multiple views per sample for improving translation quality. To enhance the encoder representation, we also investigated the Conformer encoder \cite{52}.

Experimental evaluations on three benchmark corpora, including six language directions, are conducted to show that Orthros significantly improves the BLEU scores with small additional latency. We show that the rescoring is more effective than refining predictions through further iterations. We also argue that joint training with the auxiliary shallow AR decoder improves the BLEU scores in most cases. MMT and joint training with the NAR MT task also boosts the BLEU score for the Transformer encoder while maintaining the decoding speed. The Conformer encoder improves BLEU scores across models, especially for CTC-based models. We show that Orthros-CTC with the Conformer encoder achieved the best BLEU scores among NAR models. Compared with the AR model, it achieved 3.63× faster decoding speed on a CPU with comparable BLEU scores.

The contributions of this article are summarized as follows:\footnote{This study is an extension of our previous study \cite{49}, in which only the first item was investigated.}

- We propose Orthros and show its effectiveness with two NAR models, i.e., CMLM and CTC-based model as examples.
- We propose two enhanced training methods for the CMLM.
- We thoroughly compare AR and NAR models on the basis of both Transformer and Conformer encoder architectures.
- We present strong AR baselines with and without sequence-level knowledge distillation (SeqKD) \cite{53}.

which will be helpful for future E2E-ST studies.
- We conducted various analyses to evaluate the effectiveness of Orthros in terms of model capacity, robustness against long-form speech, and searchability.

II. BACKGROUND

In this section, we review AR and NAR E2E-ST models. Let $X = (x_1, \ldots, x_U)$ denote input speech features in a source language, $Y = (y_1, \ldots, y_N)$ denote the target translation text, where $U$ and $N$ denote the input length and output length, respectively.

A. AR sequence model

Sequence-to-sequence models, so-called encoder-decoder models, are typically using AR sequence models such as the Transformer \cite{18}. An AR model factorizes a conditional probability of $Y$ given $X$ into a chain of conditional probabilities from left to right as follows:

$$P(Y|X) = \prod_{i=1}^{N} P_{AR}(y_i|y_{<i}, X),$$

(1)

where $P_{AR}$ is a probability density function of the AR model, and $y_{<i}$ are all previous tokens before the $i$-th token. The parameters are updated with a cross-entropy (CE) loss $L_{AR}$ formulated as

$$L_{AR} = -\log P_{AR}(Y|X)$$

$$= -\sum_{i=1}^{N} \log P_{AR}(y_i|y_{<i}, X).$$

When using the Transformer architecture, training can be carried out efficiently by feeding all ground-truth tokens to the decoder in parallel, i.e., teacher-forcing \cite{54}. During inference, however, beam search is used as a heuristic to find the most plausible sequence. Because this incrementally expands a prefix of each hypothesis token by token, the decoding speed is typically not fast enough. Specifically, the Transformer decoder slows the speed because it performs self-attention to all past tokens generated thus far. The decoding complexity is $O(UN)$.

B. NAR sequence model

To increase the decoding of AR models by generating all tokens in a target sequence in parallel, the conditional independence for each token position in the output probability is assumed with NAR models. The conditional probability in Eq. (1) is decomposed as

$$P(Y|X) = \prod_{i=1}^{N} P_{NAR}(y_i|X),$$

(2)

where $P_{NAR}$ is a probability density function of the NAR model. Because $P_{NAR}$ is not conditioned on $y_{<i}$, this formulation enables the generation of a sequence with a single iteration, which can achieve large increases in decoding speed. However, such a strong assumption degrades translation quality because of the multimodality problem, in which multiple
correct translations are predicted given the same source sentence [19]. SeqKD is an effective method for mitigating this problem by transforming reference translations in the training data into a more deterministic form by using a teacher AR model [55].

To relax the conditional independence assumption, iterative refinement methods [29] have also been studied by modifying Eq. (2) to a chain of $T$ iterations as

$$P(Y|X) = \prod_{t=1}^{T} P_{NAR}(Y^{(t)}|Y^{(t-1)}, X)$$

where $Y^{(t)} = (y_1^{(t)}, \ldots, y_{N(t)}^{(t)})$ is a sequence of tokens at the $t$-th iteration, where $N(t)$ is the output length at $t$-th iteration and can be changed in some models [37]–[39]. Although iterative refinement methods slow the decoding speed of pure NAR models, they can achieve better translation quality in general and flexibly control the speed by changing $T$.

1) CTC: CTC is a loss function for a single-step latent alignment model, which eliminates the necessity of frame-level supervision for learning input-output mapping [51]. A linear projection layer is stacked on the encoder to generate a probability distribution $P_{CTC}$ so the model is typically a decoder-free architecture. The conditional independence per encoder frame is assumed with CTC, and CTC marginalizes posterior probabilities of all possible alignment paths efficiently with the forward-backward algorithm. The gap in the sequence lengths between the input and output is filled by introducing blanks labels. The CTC loss $L_{CTC}$ is defined as the negative log-likelihood:

$$L_{CTC} = -\log P_{CTC}(Y|X).$$

During inference, CTC-based models can be regarded as fully NAR models if the best label class is taken from the vocabulary at every encoder frame, which is known as greedy search. The decoding complexity is $O(U)$. CTC assumes the length of $X$ to be longer than that of $Y$ (i.e., $|X| \geq |Y|$), so previous studies using CTC for the text-input NAR MT task adopted the upsampling technique to expand the input sequence length [34]–[36]. However, this is not necessary for speech-to-text generation tasks because $|X|$ is generally much longer than $|Y|$. Instead, we usually compress input sequence lengths for the E2E-ST task [59]. Another advantage of CTC for NAR models is that an explicit target length prediction is not necessary. It can be obtained as a by-product after collapsing frame-level outputs.

2) CMLM: The CMLM [30] is an iterative refinement-based model with a token in-filling adopted in Bidirectional Encoder Representations from Transformers (BERT) [60]. During inference, CMLM adopts the Mask-Predict algorithm, which alternates Mask and Predict steps at every iteration and refines predictions through $T$ iterations. Therefore, the decoding complexity is $O(UT)$.

Inference Let $\hat{Y}_t^{(t)}$ and $Y_t^{(t)}$ be masked and observed tokens in the prediction $Y_t^{(t)}$ at the $t$-th iteration $(0 \leq t \leq T)$, respectively. Given a target length $N$ estimated using a length predictor stacked on the encoder, the CMLM starts generation from a placeholder filled with a $\text{[MASK]}$ token at all $N$ positions ($t = 0$). In the Predict step, the most plausible token is selected from the vocabulary according to the probability of the CMLM $P_{CMLM}$ at each masked position $i$ in $Y_t^{(t)}$ as

$$y_i^{(t)} = \arg \max_{w \in V} P_{CMLM}(y_i = w| Y_t^{(t)}),$$

$$P^{(t)}_{CMLM} \leftarrow \max_{w \in V} P_{CMLM}(y_t = w| Y_t^{(t)}),$$

where $V$ is the output vocabulary, and $P^{(t)}_{CMLM}$ is a score of the $i$-th token at the $t$-th iteration. Note that $P^{(t)}_{CMLM}$ is updated for masked positions only.

In the Mask step, $k(t)$ tokens having the lowest confidence scores in the previous prediction $Y_t^{(t)}$ are replaced with $\text{[MASK]}$, where $k(t)$ is a linear decay function defined as

$$k(t) = \lfloor \hat{N} \cdot \frac{T - t}{T} \rfloor.$$

As the target length must be determined before starting the token generation, length parallel decoding (LPD) [23] is typically used by generating multiple length candidates in parallel. The $l$ length candidates are determined randomly.

A length predictor is implemented as a linear classifier and trained to predict the ground-truth target sequence length $N$ given $X$ as

$$L_{lp} = -\log P_{lp}(N|X),$$

where $L_{lp}$ is a length prediction loss. Unlike the text-input MT task, where the encoder output corresponding to a spacial token [LENGTH] is used as an input to the linear classifier [30], time-averaged encoder outputs are used for the E2E-ST task. The total objective $L_{total}$ is formulated as follows:

$$L_{total} = L_{CMLM} + \lambda_{lp} L_{lp},$$

where $\lambda_{lp}$ is a weight for the length-prediction loss.

2 We use top-$l$ classes instead of $[\hat{l} - \Delta, \hat{l} + \Delta]$ centering on the best class $\hat{l}$ as in [23].
3) Semi-autoregressive Training: Semi-AR training (SMART) is an improved training method to mitigate the gap in the behaviors of the CMLM between training and test [31]. To resemble the test-time behavior based on the Mask-Predict algorithm during training, the decoder input is replaced with the model prediction at each training step. To obtain the prediction, the most plausible token is taken at all $N$ positions by adopting the same masking as the original CMLM training while the gradients are truncated. Part of the resultant tokens $\hat{Y} = (\hat{y}_1, \ldots, \hat{y}_N)$ are masked out to generate a new decoder input $Y_{\text{obs}}$, which is fed to the decoder again to calculate the CE loss at all positions as

$$L_{\text{CMLM}} = - \sum_{y \in Y} \log P_{\text{CMLM}}(y|Y_{\text{obs}}, X).$$

During inference, the Mask-Predict algorithm is used, but tokens at all positions are updated at every iteration, unlike the original CMLM training. Although SMART slightly slows the training speed by doubling the forward pass, it does not increase the decoding cost during inference.

4) Mask-CTC: Since the CMLM is designed for iterative refinement starting from [MASK] tokens at all positions, the translation quality at the early decoding iterations is likely to be poor. To provide useful contexts from a single-step NAR E2E-ST model based on CTC, Mask-CTC initializes the decoder input of the CMLM with a greedy CTC prediction [47]. The CTC module is attached to the top encoder layer and trained jointly with the CMLM decoder. The less confident CTC predictions, the probabilities of which are smaller than $p^{\text{thres}}$, are replaced with [MASK] tokens. However, unlike Mask-CTC for the ASR task, masked positions change depending on the score at each iteration because of the non-monotonic sequence generation in the ST task [48]. To keep from modifying most CTC outputs by using the original Mask-Predict algorithm, the restricted Mask-Predict algorithm was proposed [48], where $k(t)$ is truncated by the number of masked tokens at $t = 0$. We refer to this type of CMLM as $\text{CTC-CMLM}$ in this paper.

Because CTC can also be used as a length predictor, a separate length-prediction layer or LPD is not necessary. The total training objective is formulated by interpolating $L_{\text{CMLM}}$ and $L_{\text{CTC}}$ as

$$L_{\text{total}} = (1 - \lambda_{\text{CTC}})L_{\text{CMLM}} + \lambda_{\text{CTC}}L_{\text{CTC}},$$

where $\lambda_{\text{CTC}}$ is a CTC loss weight.

C. Conformer

The Conformer is a Transformer-based encoder architecture augmented by the convolution module in each block [52]. It was introduced in the ASR task to capture global and local features by self-attention and convolution, respectively. Because this property is compatible with speech, its effectiveness has been demonstrated in various speech-related tasks [61], including E2E-ST. Each Conformer block introduces an additional convolution module right after the self-attention module. An additional position-wise feed-forward network (FFN) is introduced right before the self-attention module, following the Macaron-Net [62]. Relative positional encoding is also used in each self-attention module. In our study, we investigated how the Conformer impacts the translation quality of both AR and NAR models. Although the Conformer encoder introduces additional parameters, it is not a bottleneck of decoding speed.

III. PROPOSED FRAMEWORK: ORTHROS

In this section, we propose a unified NAR E2E-ST framework, Orthros, which enhances the NAR decoder by joint training and parallel rescoring with an auxiliary shallow AR decoder.

A. Model architecture

An overview of Orthros is presented in Fig. 1. Orthros has three main components: speech encoder, NAR decoder, and auxiliary shallow AR decoder. The motivation to introduce the AR decoder is based on an observation that sequence-level scores obtained from the NAR decoder are not suitable for selecting the best translation from multiple candidates. This is because of the conditional independence assumption made with the NAR decoder, which makes the NAR decoder not fully leverage the effectiveness of generating multiple candidates.

In Orthros, the speech encoder is shared between the NAR and AR decoders, which greatly reduces the model size and computational overhead compared with using another AR model such as that proposed by Gu et al. [19]. For models except for a CTC-based one, a length predictor is also stacked on the speech encoder. The entire architecture is jointly trained.

For Orthros, we mainly focus on the CMLM as the NAR decoder because we can control the decoding speed by changing the number of iterations, and it is widely used in the MT literature [26], [27], [30], [31], [63]–[68]. We refer to this model as Orthros-CMLM. Because of powerful Conformer-CTC, we also investigated equipping the AR decoder of with CTC, referred to as Orthros-CTC. Orthros can be applied to any NAR decoder topology as long as multiple candidates can be generated.

B. Inference: parallel AR rescoring

During inference, we use sequence-level scores from the AR decoder to select the most plausible translation after generating multiple candidates with the NAR decoder. Because we can feed tokens at all positions to the AR decoder simultaneously in Eq. 1, we refer to this as parallel AR rescoring.

When using the CMLM as the NAR decoder, we use the Mask-Predict algorithm described in Section II-B2 to generate $l$ candidates through $T$ iterations. When using a CTC-based model as the NAR decoder, we use left-to-right beam search with a beam width of $l$. Although this is not purely NAR decoding, there is no matrix multiplication with weight parameters. After the NAR decoding, we immediately feed the resulting tokens to the auxiliary AR decoder. We use the sequence-level log probabilities $\frac{1}{N} \sum_{i} \log P_{\text{AR}}(y_i|y_{<i}, X)$ obtained from the AR decoder to select the best translation
among the \( l \) candidates. Note that scores from the NAR decoder are not used for the rescoring.\(^3\)

For Orthros-CMLM, this rescoring step corresponds to performing one more decoding iteration. However, as shown in Section V parallel AR rescoring with \( T - 1 \) is more effective than the Mask-Predict with \( T \) without the rescoring. A shallow AR decoder is used for rescoring purpose, so the additional decoding cost is much smaller than a single iteration of the CMLM decoder.

C. Orthros-CMLM

In this section, we present Orthros-CMLM, which uses the CMLM as the NAR decoder in Orthros. We also propose two enhanced training methods for the CMLM.

1) MMT: In the training of the CMLM, the decoder observes a single set of tokens per sample, and the CE loss is calculated at only masked positions. This is not as data-efficient as AR models, especially when training data are sparse such as in the E2E-ST task. To mitigate this problem, we introduce MMT by calculating CE losses for \( M \) forward passes with different \( Y_{\text{obs}} = Y \setminus Y_{\text{mask}} \). At each forward pass, a random mask for \( Y_{\text{mask}} \) is sampled independently. Accordingly, the CE loss in Eq. (3) is modified to the average CE losses in all forward passes as

\[
\mathcal{L}_{\text{CMLM}} = \frac{1}{M} \sum_{m=1}^{M} \mathcal{L}_{\text{CMLM}}(y^{(m)}_{\text{mask}}|Y^{(m)}_{\text{obs}}, X) = \frac{1}{M} \sum_{m=1}^{M} \sum_{y \in Y^{(m)}_{\text{mask}}} -\log P_{\text{CMLM}}(y|Y^{(m)}_{\text{obs}}, X),
\]

where \( Y^{(m)}_{\text{mask}} \) and \( Y^{(m)}_{\text{obs}} \) are masked and observed tokens in the \( m \)-th forward pass, respectively. This method corresponds to augmenting the training data by a factor of \( M \) in total. However, the number of parameter updates does not increase.

2) Auxiliary NAR MT objective: To further enhance the CMLM decoder further, we train the NAR E2E-ST model jointly with an auxiliary text-input NAR MT task by sharing the parameters of the CMLM decoder. The effectiveness of an auxiliary MT task has been studied to improve the translation quality of CTC-based models greatly improves using the Conformer encoder. Therefore, we focus only on the Conformer encoder. The total objective \( \mathcal{L}_{\text{total}} \) is formulated as

\[
\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{CMLM}} + \lambda_{\text{lm}} \mathcal{L}_{\text{lm}} + \lambda_{\text{AR}} \mathcal{L}_{\text{AR}} + \lambda_{\text{MT}} \mathcal{L}_{\text{MT}},
\]

where \( \lambda \) are the corresponding loss weights.\(^6\)

D. Orthros-CTC

In this section, we describe Orthros-CTC, which uses a CTC-based model as the NAR decoder. Unlike Orthros-CMLM, Orthros-CTC does not have any NAR decoder parameters except for the linear projection layer, so it is more parameter-efficient. Note again that CTC does not require a separate length predictor. As discussed in Section V, translation quality of CTC-based models greatly improves using the Conformer encoder. Therefore, we focus only on the Conformer encoder. The total objective \( \mathcal{L}_{\text{total}} \) is formulated as

\[
\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{CTC}} + \lambda_{\text{AR}} \mathcal{L}_{\text{AR}}.
\]

To generate multiple candidates, left-to-right prefix beam search can be used in the CTC branch. Although this is no longer NAR decoding, we can use an efficient implementation with C++\(^6\) as done in a previous study\(^3\) because there is no matrix multiplication.

IV. EXPERIMENTAL SETTING

A. Dataset

We used Must-C \( \{ \text{De, Nl, Fr, Es, It} \} \), Libritrans \( \text{En} \rightarrow \{ \text{Fr} \} \), and Fisher-CallHome Spanish \( \text{Es} \rightarrow \text{En} \) corpora for the experimental evaluations. We used the recipes provided in the ESPNet-ST toolkit\(^7\).

1) Must-C \( \text{En} \rightarrow \{ \text{De, Nl, Fr, Es, It} \} \): We used \( \text{En} \rightarrow \text{De} \) (399 h), \( \text{En} \rightarrow \text{Nl} \) (433 h), \( \text{En} \rightarrow \text{Fr} \) (483 h), \( \text{En} \rightarrow \text{Es} \) (495 h), and \( \text{En} \rightarrow \text{It} \) (456 h) directions on Must-C. This corpus contains spontaneous English lecture speech extracted from TED talks, the corresponding source transcriptions, and the target translations. Non-verbal speech labels, such as "(Applause)" and "(Laughter)" were kept during training but removed during evaluation as post-processing. We report case-sensitive detokenized BLEU scores\(^6\) on the \text{test-COMMON} set.

2) Libritrans (\( \text{En} \rightarrow \text{Fr} \)): This corpus contains 100 h of English read speech, the corresponding English transcriptions, and the French translations. The French translation in the training set is augmented with Google Translate for each utterance, and we used both references following the standard practice. We report case-insensitive detokenized BLEU scores on the \text{test} set.

\(^3\)We also investigated linearly interpolating scores from the AR and NAR decoders in the log domain, but it did not lead to improvement.

\(^4\)This corresponds to augmenting the training data by a factor of 4.

\(^5\)Unlike our previous study\(^8\), we removed the auxiliary CTC-based ASR objective because it can be removed without quality degradation when pre-training the encoder with the ASR task. Instead, an auxiliary NAR MT task was newly introduced in this article.

\(^6\)https://github.com/parlance/ctcdecode
3) Fisher-CallHome Spanish (Es→En): This corpus contains 171 h of Spanish conversational telephone speech, the corresponding Spanish transcriptions as well as the English translations. Following the standard practice of this corpus, all punctuation marks except for apostrophes were removed from both transcriptions and translations. We report case-insensitive BLEU scores on Fisher-{dev, dev2, test}, and CallHome-{dev, test, evltest}. We report case-insensitive detokenized BLEU scores on these five sets. Note that BLEU scores on the Fisher splits were evaluated with four references. We used the Fisher-dev set as the validation set.

B. Pre-processing

We extracted 80-channel log-mel filterbank coefficients computed with 25-ms window size and shifted every 10ms with three-dimensional pitch features with the Kaldi toolkit [75]. We augmented speech data with speed perturbation [76] and SpecAugment [77] for both ASR and E2E-ST task. Utterances having more than 3000 speech frames or more than 400 characters were removed from the training data to fit the GPU memory.

We tokenized all sentences with the tokenizer.perl script in the Moses toolkit [78]. Source transcriptions were lowercased, and the punctuation marks except for apostrophes were removed. We constructed shared source and target vocabularies on the basis of the byte pair encoding (BPE) algorithm [79] with the Sentencepiece toolkit [80]. ASR vocabularies were built only on source transcriptions. For the AR models, we used 1k units for all tasks on the Fisher-CallHome Spanish and Libri-trans corpora, while 5k and 8k units were used for ASR and E2E-ST/MT tasks on the Must-C corpus, respectively. For NAR models, we used 16k units on all corpora, unless otherwise noted. These vocabulary sizes were selected to achieve the best performance for each model following [49].

C. Architecture

We implemented models based on the ESPnet-ST toolkit. All models in the ASR and E2E-ST tasks consisted of 12 encoder blocks and 6 decoder blocks, while MT models used 6 Transformer encoder blocks. We also used the Transformer decoder when using the Conformer encoder. The speech encoder in the ASR and E2E-ST tasks had two convolutional neural network (CNN) blocks before the first encoder block, which had a kernel size of 3 and channel size of 256. Each CNN block down-sampled features on both time and frequency axes with a stride of two, which resulted in the four-fold time reduction. The dimensions of the self-attention layer \(d_{\text{model}}\) and FPN \(d_q\) were set to 256 and 2048, respectively, and the number of attention heads \(H\) was set to four. The kernel size of depthwise separable convolution in each Conformer block was set to 15. The CTC-based models had the same encoder architecture while the decoder was replaced with a linear projection layer.

D. Training

The Adam optimizer [83] was used for training with \(\beta_1 = 0.9, \ \beta_2 = 0.98, \ \text{and} \ \epsilon = 10^{-9}\). We used the Noam learning rate schedule [18] with warmup steps of 25k and learning-rate constant of 5.0. The effective batch size was set to 256 utterances for the NAR models. We used dropout and label smoothing [84] with a probability of 0.1 and 0.1, respectively. We set \((\lambda_\text{sp}, \lambda_\text{AR}, \lambda_\text{MT})\) to \((0.1, 0.3, 0.3)\) in Eq. (6) throughout the experiments and empirically confirmed that they work well under various conditions. The number of masks for MMT \(M\) was set to 2. We trained AR models for 30 epochs, CTC-based models and CTC-CMLM for 100 epochs, the other models for 50 epochs. The last five best checkpoints based on the validation score were used for model averaging, except that the last ten best checkpoints were used for the CTC-based models.

We initialized encoder parameters of all E2E-ST models with those of the pre-trained ASR model trained on the same speech data. We did not use any external resources for pre-training. The decoder parameters of all E2E-ST models were initialized on the basis of a strategy in BERT [60], where weight parameters were sampled from \(N(0, 0.02)\), biases were set to zero, and layer normalization parameters were set to \(\beta = 0, \ \gamma = 1\). This technique was used for the CMLM models in the MT task [30].

Following the standard practice in NAR models [19], [30], we used SeqKD with an AR Transformer MT model as a teacher, except for Libri-trans [4]. The teacher MT models used a beam width of 5.

E. Decoding

For the AR models, we used a beam width \(b_{\text{ST}} \in \{1, 4, 10\}\). The ASR models used the joint CTC/Attention decoding [85] with shallow fusion of an external long short-term memory (LSTM) LM. For the CMLM decoders, we used a length beam size \(l = 5\) and number of iterations \(T \in \{4, 10\}\) as default settings. We mainly used a relatively small \(l\) to avoid slowing the decoding speed on the CPU. We also used a de-duplication technique [27], [29], [86] for the CMLM decoders, where repeated tokens were collapsed to a single token as post-processing, except for the Fisher-CallHome Spanish corpus [9]. For Orthros-CTC, we used \(l = 20\) without de-duplication. The decoding speed was measured with a batch size of 1 on an NVIDIA TITAN RTX GPU and Intel(R) Xeon(R) Gold 6128 CPU @ 3.4GHz by averaging on five runs. We calculated BLEU scores with SacreBLEU [87].

V. MAIN RESULTS

A. Must-C

1) Transformer encoder: Results with the Transformer encoder on Must-C are shown in Table I. The single step NAR decoding with CTC showed poor BLEU scores although

7Unlike [49], we did not use a pre-trained MT model for initialization of decoder parameters. We did not observe any improvement in BLEU scores with SeqKD for both AR and NAR models on this corpus. This is probably because the teacher MT model was too weak due to a small amount of the training bitext. De-duplication was effective for corpora having long-form speech, such as Must-C and Libri-trans. We set \((\lambda_\text{sp}, \lambda_\text{AR}, \lambda_\text{MT})\) to \((0.1, 0.3, 0.3)\) in Eq. (6) throughout the experiments and empirically confirmed that they work well under various conditions. The number of masks for MMT \(M\) was set to 2. We trained AR models for 30 epochs, CTC-based models and CTC-CMLM for 100 epochs, the other models for 50 epochs. The last five best checkpoints based on the validation score were used for model averaging, except that the last ten best checkpoints were used for the CTC-based models.

We initialized encoder parameters of all E2E-ST models with those of the pre-trained ASR model trained on the same speech data. We did not use any external resources for pre-training. The decoder parameters of all E2E-ST models were initialized on the basis of a strategy in BERT [60], where weight parameters were sampled from \(N(0, 0.02)\), biases were set to zero, and layer normalization parameters were set to \(\beta = 0, \ \gamma = 1\). This technique was used for the CMLM models in the MT task [30].

Following the standard practice in NAR models [19], [30], we used SeqKD with an AR Transformer MT model as a teacher, except for Libri-trans [4]. The teacher MT models used a beam width of 5.

E. Decoding

For the AR models, we used a beam width \(b_{\text{ST}} \in \{1, 4, 10\}\). The ASR models used the joint CTC/Attention decoding [85] with shallow fusion of an external long short-term memory (LSTM) LM. For the CMLM decoders, we used a length beam size \(l = 5\) and number of iterations \(T \in \{4, 10\}\) as default settings. We mainly used a relatively small \(l\) to avoid slowing the decoding speed on the CPU. We also used a de-duplication technique [27], [29], [86] for the CMLM decoders, where repeated tokens were collapsed to a single token as post-processing, except for the Fisher-CallHome Spanish corpus [9]. For Orthros-CTC, we used \(l = 20\) without de-duplication. The decoding speed was measured with a batch size of 1 on an NVIDIA TITAN RTX GPU and Intel(R) Xeon(R) Gold 6128 CPU @ 3.4GHz by averaging on five runs. We calculated BLEU scores with SacreBLEU [87].

V. MAIN RESULTS

A. Must-C

1) Transformer encoder: Results with the Transformer encoder on Must-C are shown in Table I. The single step NAR decoding with CTC showed poor BLEU scores although
the speed increases were very large; $24.51 \times$ and $14.62 \times$ on the GPU and CPU, respectively. The iterative NAR models, however, improved BLEU scores at the cost of speed. We first compared the CMLM and SMART without the auxiliary AR decoder, and the SMART outperformed the CMLM when $T = 4$. However, the gains were diminished when $T = 10$. Therefore, we did not use SMART for Orthros. Orthros-CMLM consistently improved the CMLM for all language pairs regardless of $T$. The additional latency with the parallel AR rescoring was about 5 and 100 ms on the GPU and CPU, respectively. The iterative NAR models, which are negligible when we take the gains of BLEU scores into account. The enhanced training gains of BLEU scores by more than 4 BLEU. One limitation of the CTC-based models was that it took more training steps to converge. We trained them for 100 epochs while the Conformer encoder consistently improved BLEU scores across models. The CTC-based models showed the largest gains of BLEU scores and the speedup by more than 2.99× and 2.64× increase in speed on the GPU and CPU, respectively. However, we found that SeqKD improved the AR model and the gains were much larger than those reported in the MT literature [36]. Compared with the cascade system, the enhanced Orthros-CMLM with $T = 10$ achieved 6.44× and 3.88× faster decoding on GPU and CPU while the average BLEU score underperformed by 1.0.

2) Conformer encoder: Results with the Conformer encoder on Must-C are shown in Table II. We observed that the Conformer encoder consistently improved BLEU scores across models. The CTC-based models showed the largest gains of BLEU scores by more than 4 BLEU. One limitation of the CTC-based models was that it took more training steps to converge. We trained them for 100 epochs while we trained the CMLM models for 50 epochs. We evaluated Orthros-CTC considering the strong BLEU scores of Conformer-CTC. It resulted in further gains by parallel AR

### Table I: BLEU scores with Transformer encoder on Must-C tst-COMMON

| Model | $T$ | BLEU (↑) | Speedup (↑) |
|-------|-----|----------|-------------|
|       |     | De | Nl | Fr | Es | It | Avg | GPU | CPU |
| ESPnet-ST (b$_{ST}$ = 10) |      | 22.9 | 27.4 | 32.8 | 28.0 | 23.8 | 27.0 | – | – |
| Fairseq S2T (b$_{S2T}$ = 10) |     | 22.7 | 27.3 | 32.9 | 27.2 | 22.7 | 26.6 | – | – |
| NearST [62] |   | 22.8 | 27.2 | 33.3 | 27.4 | 22.9 | 26.7 | – | – |
| Transformer (b$_{TR}$ = 1) |     | 21.0 | 26.1 | 31.6 | 26.2 | 22.1 | 25.4 | 1.64× | 3.26× |
| Transformer (b$_{TR}$ = 4) |     | 22.8 | 27.3 | 33.3 | 27.8 | 23.3 | 26.9 | 1.00× | 1.00× |
| Transformer + SeqKD (b$_{TR}$ = 1) | | 23.8 | 27.7 | 33.7 | 28.0 | 23.3 | 27.3 | 1.64× | 3.26× |
| Transformer + SeqKD (b$_{TR}$ = 4) |   | 24.3 | 28.4 | 34.5 | 28.9 | 24.2 | 28.1 | 1.00× | 1.00× |

| Model | $T$ | BLEU (↑) | Speedup (↑) |
|-------|-----|----------|-------------|
|       |     | De | Nl | Fr | Es | It | Avg | GPU | CPU |
| CTC | 1 | 19.8 | 23.9 | 28.9 | 22.8 | 19.5 | 23.0 | 24.5× | 14.6× |
| CMLM | 4 | 20.4 | 23.6 | 29.1 | 24.0 | 20.5 | 23.4 | 1.00× | 4.53× |
| SMART | 4 | 21.8 | 25.3 | 30.0 | 25.3 | 21.5 | 24.8 | 4.69× | 5.10× |
| Orthros-CMLM + MMT + NAR MT | | 22.5 | 26.1 | 31.5 | 26.0 | 22.7 | 25.8 | 5.69× | 10.0× |
| CMLM | 10 | 21.5 | 25.1 | 30.7 | 25.3 | 21.8 | 24.9 | 3.13× | 2.75× |
| SMART | 10 | 21.4 | 25.2 | 31.0 | 25.4 | 22.1 | 25.0 | 5.13× | 2.75× |
| Orthros-CMLM + MMT + NAR MT | | 23.5 | 26.4 | 31.8 | 26.1 | 22.5 | 26.0 | 2.99× | 2.64× |
| Cascade AR Trf ASR → Trf MT | 2 | 23.5 | 28.5 | 33.9 | 28.6 | 24.3 | 27.8 | 0.45× | 0.68× |

### Table II: BLEU scores with Conformer encoder on Must-C tst-COMMON

| Model | $T$ | BLEU (↑) | Speedup (↑) |
|-------|-----|----------|-------------|
|       |     | De | Nl | Fr | Es | It | Avg | GPU | CPU |
| E2E AR |     |     |     |     |     |     |     |     |     |
| Conformer (b$_{CTC}$ = 1) | | 23.2 | 28.3 | 34.5 | 29.7 | 24.6 | 28.1 | 1.59× | 2.82× |
| Conformer (b$_{CTC}$ = 4) | | 25.0 | 29.7 | 35.5 | 30.5 | 25.4 | 29.2 | 1.00× | 1.00× |
| Conformer + SeqKD (b$_{CTC}$ = 1) | | 25.7 | 29.7 | 35.8 | 30.6 | 25.5 | 29.5 | 1.59× | 2.82× |
| Conformer + SeqKD (b$_{CTC}$ = 4) | | 26.3 | 30.6 | 36.4 | 31.0 | 25.9 | 30.0 | 1.00× | 1.00× |
| E2E NAR |     |     |     |     |     |     |     |     |     |
| CTC (l = 1) | 1 | 24.1 | 28.5 | 34.6 | 29.0 | 24.3 | 28.1 | 13.83× | 8.32× |
| Orthonl-CTC (l = 20) | | 25.3 | 29.9 | 36.2 | 30.4 | 25.4 | 29.4 | 1.14× | 3.63× |
| CMLM | 4 | 22.6 | 26.3 | 31.3 | 26.5 | 21.6 | 25.7 | 4.54× | 4.20× |
| CTC-CMLM | | 24.1 | 27.8 | 34.7 | 29.1 | 24.5 | 28.0 | 5.07× | 5.97× |
| Orthros-CMLM | | 23.4 | 27.6 | 33.3 | 28.3 | 23.2 | 27.2 | 4.92× | 4.03× |
| Orthonl-CMLM + MMT + NAR MT | | 23.3 | 27.5 | 34.0 | 28.7 | 23.7 | 27.4 | 4.92× | 4.03× |
| CMLM | 10 | 23.5 | 27.6 | 33.0 | 27.5 | 22.7 | 26.9 | 2.89× | 2.33× |
| CTC-CMLM | | 24.2 | 28.2 | 34.8 | 29.3 | 24.6 | 28.1 | 3.06× | 4.43× |
| Orthonl-CMLM + MMT + NAR MT | | 24.5 | 28.5 | 34.6 | 29.1 | 24.0 | 28.1 | 2.73× | 2.31× |
| Cascade AR CfM ASR → Trf MT | 2 | 24.1 | 29.4 | 35.0 | 29.5 | 24.7 | 28.5 | 0.46× | 0.67× |
rescoring and addressed the slow convergence issue. In terms of decoding speed, Orthros-CTC achieved a large gain for the CPU (3.63×) while that for the GPU was small (1.14×). We also investigated CTC-CMLM, which refined a CTC output through multiple iterations. Although it outperformed Orthros-CMLM in speed with similar BLEU scores, we did not observe any BLEU improvement from the pure CTC-based model. Therefore, increasing the number of candidates was more effective than refining the single output multiple times. Similar to the Transformer encoder, Orthros-CMLM improved the CMLM by 1.5 and 1.2 BLEU scores on average for $T = 4$ and $T = 10$, respectively. The enhanced training slightly improved the BLEU scores (+0.2) except for En$\rightarrow$De and En$\rightarrow$Fr. We reason that the Conformer encoder already enhanced the CMLM decoder by providing better encoder representations and augmenting masks for the CMLM training was not complementary. Comparing the enhanced Orthros-CMLM ($T = 10$) with the cascade system, the gap in the average BLEU score was reduced from 1.0 to 0.2 by using the Conformer encoder. To summarize, Orthros-CTC showed the best BLEU scores in all language pairs.

### B. Libri-trans

The results from the Transformer and Conformer encoders on Libri-trans are listed in Tables III and IV, respectively. We did not use SeqKD for this corpus because it was not effective in our preliminary experiments. We also referred to the results of previous studies that did not leverage additional resources.

For Transformer-based models, we confirmed that SMART was not effective and Orthros outperformed both CMLM and SMART regardless of $T$. The enhanced training was also helpful, boosting the BLEU score by 0.9 for $T = 4$ and 0.5 for $T = 10$. The enhanced Orthros with $T = 10$ matched the AR model in terms of BLEU score with 4.39× and 4.99× increase in decoding speed on the GPU and CPU, respectively. The relative increase in speed were much larger than those on Must-C because the test set on Libri-trans had many long-form utterances up to about 50 s.

For Conformer-based models, we observed large improvements across all models. The AR model achieved the state-of-the-art BLEU score of 19.2. The other trend was consistent with that on Must-C. Conformer-CTC improved upon the Transformer-CTC by 4.1 BLEU, and Orthros-CTC further boosted it by 1.0 BLEU. Orthros-CMLM outperformed the CMLM by 1.4 BLEU for $T = 4$ and 1.8 BLEU for $T = 10$, while the enhanced training did not improve it further.

### C. Fisher-CallHome Spanish

The results with the Transformer and Conformer encoders on Fisher-CallHome Spanish are listed in Tables V and VI, respectively. Note that the CallHome sets are out-of-domain because the models were trained in the Fisher domain. We confirmed similar trends for Must-C and Libri-trans. However, unlike previous experiments, we observed that Conformer-based Orthros-CMLM outperformed Conformer-CTC on the CallHome test sets by a large margin. Considering the fact that the word error rates on the CallHome test sets were much larger than those on the Fisher test sets (40% vs. 20% [61], [90]), we can conclude that Orthros is more robust in noisy acoustic conditions.

### VI. Analysis

#### A. Effective number of auxiliary AR decoder layers

We first investigated the effective number of auxiliary AR decoder layers $N_{AR}$ for Orthros-CMLM. We increased $N_{AR}$ from zero to six [11]. We used the Conformer encoder without any enhanced training method. The results on the dev set of Must-C En$\rightarrow$De, Libri-trans, and Fisher-CallHome Spanish in Table VII indicate that a shallow AR decoder works.

\[^{11}\text{In our previous study [49], we used } N_{AR} = 6 \text{ and did not study its impact on BLEU scores.}\]
TABLE V: BLEU scores with Transformer encoder on Fisher-CallHome Spanish En→En. $N_{AR}$ was set to 3. Decoding speed was measured on Fisher-Test set.

| Model                        | T       | BLEU (↑) | Speedup (↑) |
|------------------------------|---------|----------|-------------|
|                              |         | dev2     | test        | dev test   | evl test |
| Fisher                       |         |          |             |            |          |
| CNN                          | 1       | 45.8     | 46.4       | 46.2       | 15.6     | 16.0     | 22.11x   | 11.55x   |
| Must-C                       | 2       | 45.3     | 46.2       | 45.4       | 17.5     | 16.8     | 5.76x    | 4.50x    |
| Libri-trans                  | 4       | 45.3     | 46.1       | 45.9       | 17.2     | 17.0     | 5.76x    | 4.50x    |
| CMLM                         | + MMT + NAR MT | 49.1     | 49.9       | 48.5       | 19.0     | 18.9     | 4.95x    | 4.05x    |
| CMLM                         | ST (l = 1) | 49.7     | 48.1       | 46.9       | 18.8     | 18.5     | 3.19x    | 2.47x    |
| CMLM                         | ST (l = 20) | 50.3     | 50.7       | 48.9       | 20.0     | 18.9     | 4.18x    | 3.44x    |
| CMLM                         | + MMT + NAR MT | 50.3     | 50.7       | 49.0       | 20.3     | 20.0     | 4.18x    | 3.44x    |
| CMLM                         | ST (l = 1) | 49.8     | 49.7       | 48.9       | 19.7     | 19.4     | 2.89x    | 2.30x    |
| CMLM                         | ST (l = 20) | 51.3     | 51.7       | 50.7       | 17.9     | 18.3     | 3.40x    | 3.89x    |
| CMLM                         | + MMT + NAR MT | 51.8     | 52.3       | 50.9       | 20.7     | 20.0     | 2.70x    | 2.20x    |
| Cascade AR                   | Trf ASR → Trf MT | 2N     | 41.4     | 43.3       | 42.3     | 20.4     | 19.9     | 0.50x    | 0.64x    |

TABLE VI: BLEU scores with Conformer encoder on Fisher-CallHome Spanish En→En

| Model                        | T       | BLEU (↑) | Speedup (↑) |
|------------------------------|---------|----------|-------------|
|                              |         | dev2     | test        | dev test   | evl test |
| Fisher                       |         |          |             |            |          |
| CNN                          | 1       | 51.0     | 51.6       | 50.8       | 18.0     | 18.7     | 11.80x   | 6.91x    |
| Must-C                       | 2       | 50.4     | 54.8       | 54.1       | 21.0     | 20.8     | 1.09x    | 2.80x    |
| Libri-trans                  | 4       | 50.3     | 50.7       | 48.9       | 20.0     | 18.9     | 4.18x    | 3.44x    |
| CMLM                         | ST (l = 1) | 50.3     | 50.7       | 49.0       | 20.3     | 20.0     | 4.18x    | 3.44x    |
| CMLM                         | ST (l = 20) | 51.3     | 51.7       | 50.7       | 17.9     | 18.3     | 3.40x    | 3.89x    |
| CMLM                         | + MMT + NAR MT | 51.8     | 52.3       | 50.9       | 20.7     | 20.0     | 2.70x    | 2.20x    |
| Cascade AR                   | Cfm ASR → Trf MT | 2N     | 42.6     | 44.4       | 43.2     | 21.7     | 21.0     | 0.50x    | 0.66x    |

TABLE VII: Effective number of auxiliary AR decoders $N_{AR}$ for Orthros-CMLM on dev sets of Must-C En→De, Libri-trans, and Fisher-CallHome Spanish. Conformer encoder was used. All models used parallel AR rescoring. No enhanced training method, such as MMT, was used.

| $N_{AR}$ | BLEU (↑) |
|----------|----------|
|          | Must-C       | Libri-trans | Fisher    |
|          | $T = 4$ | $T = 10$  | $T = 4$ | $T = 10$ | $T = 4$ | $T = 10$ |
| 0        | 21.4   | 22.7     | 16.2   | 16.9     | 47.7   | 49.8     |
| 1        | 22.2   | 23.2     | 17.4   | 18.4     | 48.7   | 50.1     |
| 2        | 21.9   | 23.1     | 17.3   | 18.2     | 49.2   | 51.0     |
| 3        | 22.0   | 23.2     | 16.8   | 18.1     | 50.3   | 51.8     |
| 4        | 21.0   | 22.4     | 17.6   | 18.5     | 49.5   | 51.6     |
| 5        | 21.8   | 23.1     | 17.3   | 18.4     | 47.9   | 49.5     |
| 6        | 21.9   | 22.8     | 16.9   | 17.8     | 49.1   | 51.1     |

well across corpora. Specifically, $N_{AR} = 1$ was best on Must-C, $N_{AR} = 4$ was best on Libri-trans while $N_{AR} = 1$ showed comparable BLEU scores. However, Fisher-CallHome Spanish reached a peak at $N_{AR} = 3$. This was probably because the input speech was much noisier compared with other corpora; therefore, stacking multiple AR decoder layers was more effective for providing better sequence-level scores. Otherwise, a single layer was enough, which minimized the additional decoding cost by the rescoring. Hence, we used $N_{AR} = 3$ on Fisher-CallHome Spanish and $N_{AR} = 1$ on the other corpora in the other experiments, regardless of the encoder type. We also used the same $N_{AR}$ for Orthros-CTC.

B. Ablation study

We conducted an ablation study of Orthros-CMLM in terms of parallel AR rescoring and the enhanced training methods on the dev sets of Must-C En→De, Libri-trans, and Fisher-CallHome Spanish shown in Table VIII. We used the Transformer encoder and first observed that joint training with an auxiliary shallow AR decoder improved the BLEU scores
increased the number of masks for the CMLM training by A9 MMT without the auxiliary NAR-MT task (training methods was beneficial (A4 A2 scores in all settings (A4 (512 A1 A3 used (A8 (NAR models based on a large Transformer architecture. We Therefore, we also compared decoding speeds of AR and coder was used. (architectures of AR and NAR models, except that the size (model did not lead to a notable BLEU improvement in our (the NAR MT literature [36], where (dmodel,H) = (512,8) was typically used. Therefore, our baseline AR model was relatively lightweight. This was because using a large AR model did not lead to a notable BLEU improvement in our preliminary experiments due to the limited parallel data in the ST corpora. However, when additional data are available, increasing model capacity would be typically beneficial. Therefore, we also compared decoding speeds of AR and NAR models based on a large Transformer architecture. We used (dmodel,H) = (512,8) in both the encoder and decoder architectures of AR and NAR models, except that the size of the auxiliary AR decoder in Orthros-CMLM was kept to (dmodel,H) = (256,4). The results on Must-C En→De tst-COMMON in Table IX indicate that using the large architecture improved the BLUE score of the AR model by only 0.4. The inference speed was not so different from that in Table IV but we observed a large BLEU improvement in Orthros-CMLM by increasing the model capacity. Compared with Table IV it improved BLEU scores by 1.2 and 1.1 when T = 4 and T = 10, respectively. The quality was further boosted by the enhanced training by 0.8 and 0.4 for T = 4 and T = 10, respectively. Finally, the enhanced Orthros-CMLM almost matched the strong AR model trained with SeqKD in quality with 3× faster decoding.

C. Large model

In the above experiments, we used (dmodel,H) = (256,4) in all models. They are much smaller than those used in the NAR MT literature [36], where (dmodel,H) = (512,8) was typically used. Therefore, our baseline AR model was relatively lightweight. This was because using a large AR model did not lead to a notable BLEU improvement in our preliminary experiments due to the limited parallel data in the ST corpora. However, when additional data are available, increasing model capacity would be typically beneficial. Therefore, we also compared decoding speeds of AR and NAR models based on a large Transformer architecture. We used (dmodel,H) = (512,8) in both the encoder and decoder architectures of AR and NAR models, except that the size

### Table VIII: Ablation study of parallel AR rescoring and enhanced training methods on dev sets of Must-C En→De, Libri-trans, and Fisher-CallHome Spanish. Transformer encoder was used.

| ID | Model                              | Must-C | Libri-trans | Fisher |
|----|------------------------------------|--------|-------------|--------|
|    |                                    | T = 4  | T = 10      | T = 4  | T = 10 |
| A1 | CMLM                               | 19.8   | 21.4        | 14.6   | 15.8   | 45.3   | 47.7   |
| A2 | Orthros-CMLM                       | 21.1   | 22.0        | 15.7   | 16.9   | 47.4   | 49.9   |
| A3 | - parallel AR rescoring            | 20.4   | 21.3        | 15.1   | 16.3   | 45.7   | 48.3   |
| A4 | Orthros-CMLM + MMT + NAR MT       | 21.9   | 22.8        | 16.3   | 17.3   | 49.1   | 50.8   |
| A5 | - parallel AR rescoring            | 21.5   | 22.4        | 15.7   | 16.7   | 47.2   | 49.0   |
| A6 | - MMT (M = 2)                     | 21.8   | 23.0        | 15.6   | 16.3   | 48.6   | 50.2   |
| A7 | - NAR MT                           | 21.2   | 22.4        | 16.4   | 17.2   | 47.5   | 49.9   |
| A8 | + MMT (M = 3)                     | 21.8   | 22.6        | 16.1   | 17.2   | 47.9   | 50.1   |
| A9 | + MMT (M = 4)                     | 21.4   | 22.4        | 16.1   | 17.0   | 48.7   | 50.4   |

(A1 vs. A3), except for T = 10 on Must-C. A3 was trained jointly with the auxiliary AR decoder but did not use it during inference. Parallel AR rescoring further improved the BLEU scores in all settings (A2 vs. A3).

The effectiveness of parallel AR rescoring remained even when using the enhanced training combining MMT (M = 2) and the auxiliary NAR-MT task (A4 vs. A5). Each of these training methods was beneficial (A4 vs. A6 vs. A7). Because A4 increased the number of masks for the CMLM training by a factor of 4 per sample, we also investigated increasing M in MMT without the auxiliary NAR-MT task (A9). However, we confirmed that this was less effective, indicating that providing source information from different modalities was helpful to some extent.

### Table IX: BLEU scores of large models with (dmodel,H) = (512,8) on Must-C En→De tst-COMMON. Transformer encoder was used.

| Model                              | T | BLEU (↑) | Speedup (↑) |
|------------------------------------|---|----------|-------------|
|                                    |   |          | GPU         |
| Transformer AR + SeqKD (bST = 1)   | N | 24.0     | 1.62× 3.30× |
| Transformer AR + SeqKD (bST = 4)   | 24.7 | 1.00× 1.00× |
| Orthros-CMLM + MMT + NAR MT       | 4 | 23.0     | 8.59× 5.63× |
| Orthros-CMLM + MMT + NAR MT       | 10 | 24.0     | 3.02× 2.92× |

Fig. 2: Robustness against long-form speech on IWSLT tst2019 set. X-axis denotes maximum input length threshold in segment-merging algorithm [91].

D. Robustness against long-form speech

In speech translation, audio segmentation has a large impact on the final translation quality [91]–[95] because acoustic-driven segmentation does not necessarily correspond to sentence segmentation based on punctuation marks. Therefore, it is important to translate long-form speech robustly by incorporating long context [91]. Therefore, we investigated the robustness of E2E-ST models against long-form data on the International Conference on Spoken Language Translation (IWSLT) tst2019 set (En→De)[12] Following the segment-merging algorithm [91], we split the entire speech using a neural voice activity detection model [99] then concatenated multiple adjacent segments until reaching the desired input

12The IWSLT test sets have reference translation over the entire session, unlike Must-C.
length. Figure 2 shows BLEU scores as a function of the maximum input length threshold for the segment merging. We evaluated models trained on Must-C En→De \texttt{tst-COMMON}. Conformer encoder was used.

We did not merge adjacent segments with interval above 100ms.

**TABLE X: Oracle BLEU scores on Must-C En→De \texttt{tst-COMMON}.** Numbers inside brackets denote gains from parallel AR rescoring.

| Model                  | \( T \) | \( l \) | Oracle BLEU (\( \Delta \)) (↑) |
|------------------------|--------|-------|---------------------------------|
| **Transformer encoder** |        |       |                                 |
| Orthros-CMLM           | 4      | 5     | 24.6 (+2.8)                     |
|                        | 4      | 9     | 26.1 (+3.9)                     |
|                        | 10     | 5     | 25.9 (+3.0)                     |
| + MMT + NAR MT         | 4      | 5     | 25.7 (+3.2)                     |
|                        | 4      | 9     | 27.2 (+4.4)                     |
|                        | 10     | 5     | 26.9 (+3.4)                     |
|                        | 10     | 9     | 28.4 (+4.6)                     |
| **Conformer encoder**  |        |       |                                 |
| Orthros-CTC            | 1      | 20    | 28.6 (+3.3)                     |
| Orthros-CMLM           | 4      | 5     | 26.2 (+2.4)                     |
|                        | 4      | 9     | 27.5 (+4.1)                     |
|                        | 10     | 5     | 27.1 (+2.5)                     |
|                        | 10     | 9     | 28.5 (+4.0)                     |

**Fig. 3:** Effectiveness of length beam size \( l \) with \( T = 10 \) on Must-C En→De \texttt{tst-COMMON}. Conformer encoder was used.

**E. Searchability**

In this section, we analyze the searchability of Orthros by changing a length beam size \( l \). We investigate how it impacts the quality of hypotheses.

1) **Length beam size:** Figure 3 shows BLEU scores as a function of length beam size \( l \) with \( T = 10 \) on Must-C En→De \texttt{tst-COMMON}. We used the Conformer encoder and confirmed that parallel AR rescoring consistently improved the quality of NAR models when \( l > 1 \) and the gain increased using a larger \( l \). The baseline CMLM and CTC-based model did not improve by increasing \( l \) because the NAR decoders could not provide effective sequence-level scores. Because parallel AR rescoring introduces an additional iteration in terms of \( T \), we also investigated reducing \( T \) in Orthros-CMLM by 1. We confirmed that BLEU scores of Orthros-CMLM with \( T = 9 \) was much better than those of CMLM with \( T \) regardless of \( l \) and the gap was enlarged as \( l \) increased. Therefore, we conclude that parallel AR rescoring improves the searchability of NAR models.

2) **Oracle BLEU:** We next investigated oracle BLEU scores, where the best hypothesis was selected from candidates generated from the NAR decoder based on the sentence-level BLEU score with the reference translation. It is regarded as the upper bound that can be achieved by parallel AR rescoring. We conducted this experiment to identify a room for improvement of translation quality of NAR models. Table X indicates that increasing the number of length candidates can generate translations of higher quality than refining the predictions for more iterations. However, NAR models were still affected by selecting a better translation even with parallel AR rescoring. This was more severe when using a large \( l \). Better training and architecture also led to higher oracle BLEU scores.

To increase the decoding speed of E2E-ST models, we proposed Orthros, which introduces an auxiliary shallow AR decoder on top of the shared encoder to assist the NAR decoder and optimizes the entire network end-to-end. The AR decoder is used for rescoring outputs from the NAR decoder with a very small additional decoding cost. We investigated the CMLM and CTC as the NAR decoder for Orthros. We also compared Transformer and Conformer encoder architectures. We introduced MMT and joint training with an auxiliary text-input NAR-MT task to enhance the CMLM decoder. Experimental evaluations on three benchmark corpora including six language pairs confirmed the consistent effectiveness of Orthros; parallel AR rescoring improved the BLUE scores of the NAR model regardless of the encoder and decoder topologies. Enhanced training methods improved the quality of Orthros-CMLM by a large margin when the Transformer encoder was used. When increasing the model capacity, the enhanced Orthros-CMLM matched the strong large AR model trained with SeqKD in terms of BLEU score with 3× faster decoding. The Conformer encoder improved the overall quality across models while the effectiveness of the parallel AR rescoring was maintained. Orthros-CTC showed the best BLEU score while achieving a 3.63× increase in decoding speed on a CPU compared with the AR model.

For future work, we will further improve the translation quality of Conformer-CTC by adopting a deep encoder architecture [27]. We believe that improving the selection of a better hypothesis in NAR models would bridge the quality gap between AR and NAR models.

**REFERENCES**

[1] F. W. Stentiford and M. G. Steer, “Machine translation of speech,” *British Telecom technology journal*, vol. 6, no. 2, pp. 116–122, 1988.
