

Hybrid Transducer and Attention based Encoder-Decoder Modeling for Speech-to-Text Tasks

Yun Tang*, Anna Y. Sun*, Hirofumi Inaguma*, Xinyue Chen[◇]*,
Ning Dong*, Xutai Ma*, Paden D. Tomasello*, Juan Pino*

Meta AI*, Carnegie Mellon University[◇]
yuntang.email@gmail.com, hirofumii@meta.com

Abstract

Transducer and Attention based Encoder-Decoder (AED) are two widely used frameworks for speech-to-text tasks. They are designed for different purposes and each has its own benefits and drawbacks for speech-to-text tasks. In order to leverage strengths of both modeling methods, we propose a solution by combining Transducer and Attention based Encoder-Decoder (TAED) for speech-to-text tasks. The new method leverages AED's strength in non-monotonic sequence to sequence learning while retaining Transducer's streaming property. In the proposed framework, Transducer and AED share the same speech encoder. The predictor in Transducer is replaced by the decoder in the AED model, and the outputs of the decoder are conditioned on the speech inputs instead of outputs from an unconditioned language model. The proposed solution ensures that the model is optimized by covering all possible read/write scenarios and creates a matched environment for streaming applications. We evaluate the proposed approach on the MuST-C dataset and the findings demonstrate that TAED performs significantly better than Transducer for offline automatic speech recognition (ASR) and speech-to-text translation (ST) tasks. In the streaming case, TAED outperforms Transducer in the ASR task and one ST direction while comparable results are achieved in another translation direction.¹

1 Introduction

Neural based end-to-end frameworks have achieved remarkable success in speech-to-text tasks, such as automatic speech recognition (ASR) and speech-to-text translation (ST) (Li, 2021). These frameworks include Attention based Encoder-Decoder modeling (AED) (Bahdanau et al., 2014), connectionist

temporal classification (CTC) (Graves et al., 2006) and Transducer (Graves, 2012) etc. They are designed with different purposes and have quite different behaviors, even though all of them could be used to solve the mapping problem from a speech input sequence to a text output sequence.

AED handles the sequence-to-sequence learning by allowing the decoder to attend to parts of the source sequence. It provides a powerful and general solution that is not bound to the input/output modalities, lengths, or sequence orders. Hence, it is widely used for ASR (Chorowski et al., 2015; Chan et al., 2015; Zhang et al., 2020; Gulati et al., 2020; Tang et al., 2021), and ST (Berard et al., 2016; Weiss et al., 2017; Li et al., 2021; Tang et al., 2022).

CTC and its variant Transducer are designed to handle monotonic alignment between the speech input sequence and text output sequence. A hard alignment is generated between speech features and target text tokens during decoding, in which every output token is associated or synchronized with an input speech feature. CTC and Transducer have many desired properties for ASR. For example, they fit into streaming applications naturally, and the input-synchronous decoding can help alleviate over-generation or under-generation issues within AED. Sainath et al. (2019); Chiu et al. (2019) show that Transducer achieves better WER than AED in long utterance recognition, while AED outperforms Transducer in the short utterance case. On the other hand, CTC and Transducer are shown to be sub-optimal in dealing with non-monotonic sequence mapping (Chuang et al., 2021), though some initial attempts show encouraging progress (Xue et al., 2022; Wang et al., 2022).

In this work, we propose a hybrid Transducer and AED model (TAED), which integrates both AED and Transducer models into one framework to leverage strengths from both modeling methods. In TAED, we share the speech encoder between

*Xinyue Chen contributed to this work during her internship at Meta.

¹https://github.com/facebookresearch/fairseq/tree/main/examples/transformer_transducer.

AED and Transducer. The predictor in Transducer is replaced with the decoder in AED. The AED decoder output assists the Transducer’s joiner to predict the output tokens. Transducer and AED models are treated equally and optimized jointly during training, while only Transducer’s joiner outputs are used during inference. We extend the TAED model to streaming applications under the chunk-based synchronization scheme, which guarantees full coverage of read/write choices in the training set and removes the training and inference discrepancy. The relationship between streaming latency and AED alignment is studied, and a simple, fast AED alignment is proposed to achieve low latency with small quality degradation. The new approach is evaluated in ASR and ST tasks for offline and streaming settings. The results show the new method helps to achieve new state-of-the-art results on offline evaluation. The corresponding streaming extension also improves the quality significantly under a similar latency budget. To summarize, our contributions are below:

1. TAED, the hybrid of Transducer and AED modeling, is proposed for speech-to-text tasks
2. A chunk-based streaming synchronization scheme is adopted to remove the training and inference discrepancy for streaming applications
3. A simple, fast AED alignment is employed to balance TAED latency and quality
4. The proposed method achieves SOTA results on both offline and streaming settings for ASR and ST tasks

2 Preliminary

Formally, we denote a speech-to-text task training sample as a (\mathbf{x}, \mathbf{y}) pair. $\mathbf{x} = x_{1:T}$ and $\mathbf{y} = y_{1:U}$ are the speech input features and target text tokens, respectively. T and U are the corresponding sequence lengths. $y_u \in \mathcal{V}$ and \mathcal{V} is the target vocabulary. The objective function is to minimize the negative log likelihood $\log p(\mathbf{y}|\mathbf{x}, \theta)$ over the training set

$$\mathcal{L}_{\text{aed}} = - \sum_{(\mathbf{x}, \mathbf{y})} \log p(y_u | y_{1:u-1}, x_{1:T}). \quad (1)$$

In the streaming setting, the model generates predictions at timestamps denoted by $a =$

$(t_1, \dots, t_u, \dots, t_U)$, rather than waiting to the end of an utterance, where $t_u \leq t_{u+1}$ and $0 < t_u \leq T$. We call the prediction timestamp sequence as an alignment a between speech $x_{1:T}$ and token labels $y_{1:U}$. $\mathcal{A}_T^U = \{a\}$ denotes all alignments between $x_{1:T}$ and $y_{1:U}$. The streaming model parameter θ_s is optimized through

$$\min_{\theta_s} \sum_{(\mathbf{x}, \mathbf{y})} \sum_{a \in \mathcal{A}_T^U} \sum_{u=1}^U -\log p(y_u | y_{1:u-1}, x_{1:t_u}). \quad (2)$$

The offline modeling can be considered a special case of streaming modeling, i.e., the alignment is unique with all $t_u = T$. The following two subsections briefly describe two modeling methods used in our hybrid approach.

2.1 Attention based encoder decoder

AED consists of an encoder, a decoder, and attention modules, which connect corresponding layers in the encoder and decoder as demonstrated in Figure 1(a). The encoder generates the context representation $h_{1:T}$ from input $x_{1:T}$ ². The decoder state s_u^l is estimated based on previous states and encoder outputs

$$s_u^l = f_{\theta_{\text{dec}}}(h_{1:T}, s_{1:u-1}^l, y_{u-1}), \quad (3)$$

where $f_{\theta_{\text{dec}}}$ is the neural network parameterized with θ_{dec} and $l \in [1, L]$ is the layer index.

When the AED is extended to the streaming applications (Raffel et al., 2017; Arivazhagan et al., 2019), a critical question has been raised: *how do we decide the write/read strategy for the decoder?*

Assuming the AED model is Transformer based, and tokens $y_{1:u-1}$ have been decoded before timestep t during inference. The next AED decoder state $s_u^l(t)$ is associated with partial speech encoder outputs $h_{1:t}$ as well as a partial alignment $a' \in \mathcal{A}_t^{u-1}$ between $h_{1:t}$ and $y_{1:u-1}$. The computation of a Transformer decoder layer (Vaswani et al., 2017) includes a self-attention module and a cross-attention module. The self-attention module models the relevant information from previous decoder states

$$\hat{s}_{a'}^l = [s_1^l(t_1), \dots, s_{u-1}^l(t_{u-1})], \quad (4)$$

where t_{u-1} is the prediction timestamp for token $u - 1$ in alignment a' . The cross-attention module

²A down-sampling module might be applied in the speech encoder. For simplicity, we still use T as the encoder output sequence length.

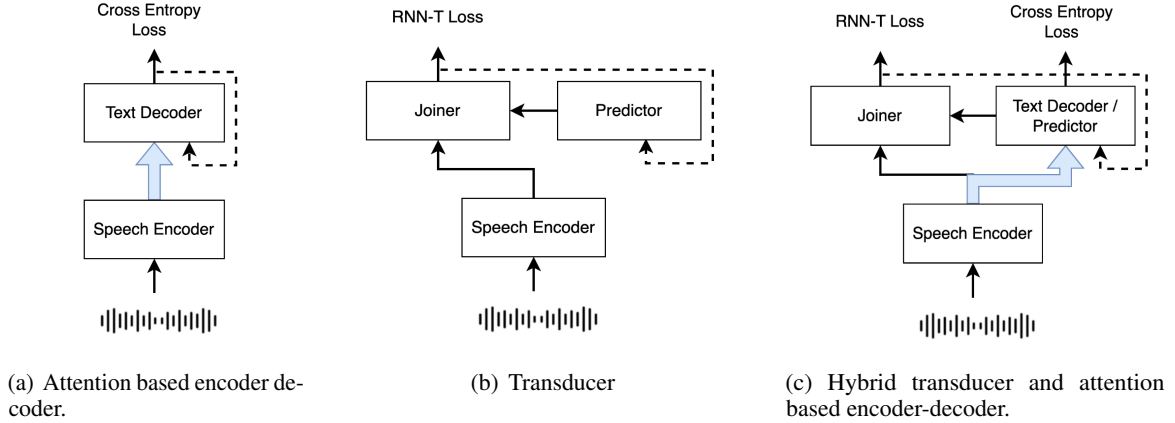


Figure 1: Comparison among different frameworks: AED, Transducer and TAED.

extracts information from the encoder outputs $h_{1:t}$. The decoder state computation is modified as

$$s_u^l(t) = f_{\theta_{\text{dec}}}(h_{1:t}, \hat{s}_{a'}^l, y_{u-1}). \quad (5)$$

To cover all read/write paths during training, we need to enumerate all possible alignments at every timestep given the output token sequence $y_{1:U}$. The alignment numbers would be $O(\frac{T'!(T'-U)!}{U!})$ and it is prohibitively expensive. In AED based methods, such as Monotonic Infinite Lookback Attention (MILk) (Arivazhagan et al., 2019) and Monotonic Multihead Attention (MMA) (Ma et al., 2020b), an estimation of context vector is used to avoid enumerating alignments. In Cross Attention Augmented Transducer (CAAT) (Liu et al., 2021), the self-attention modules in the joiner are dropped to decouple $y_{1:u-1}$ and $h_{1:t}$.

2.2 Transducer

A Transducer has three main components. A speech encoder θ_{enc} forms the context speech representation $h_{1:T}$ from speech input $x_{1:T}$, a predictor θ_{pred} models the linguistic information conditioned on previous target tokens, and a joiner θ_{joiner} merges acoustic and linguistic representations to predict outputs for every speech input feature, as shown in 1(b). The encoder and predictor are usually modeled with a recurrent neural network (RNN) (Graves, 2012) or Transformer (Zhang et al., 2020) architecture. The joiner module is a feed-forward network which expands input from speech encoder h_t and predictor output s_u^L to a $T \times U$ matrix with component $z(t, u)$:

$$z(t, u) = f_{\theta_{\text{joiner}}}(h_t, s_u^L). \quad (6)$$

A linear projection $W^{\text{out}} \in \mathcal{R}^{d \times |\mathcal{V} \cup \emptyset|}$ is applied to $z(t, u)$ to obtain logits for every output token $k \in \mathcal{V} \cup \emptyset$. A blank token \emptyset is generated if there is no good match between non-blank tokens and current h_t . The RNN-T loss is optimized using the forward-backward algorithm:

$$\alpha_{t,u} = \text{LA} \left(\alpha_{t,u-1} + \log p(y_u | z(t, u-1)), \right. \\ \left. \alpha_{t-1,u} + \log p(\emptyset | z(t, u)) \right), \quad (7)$$

$$\mathcal{L}_{\text{rnn-t}} = -\alpha_{T,U} - \log p(\emptyset | T, U), \quad (8)$$

where $\text{LA}(x, y) = \log(\exp^x + \exp^y)$ and $\alpha_{0,\phi}$ is initialized as 0. Transducer is well suited to the streaming task since it can learn read/write policy from data implicitly, i.e., a blank token indicates a read operation and a non-blank token indicates a write operation.

3 Methods

In this study, we choose the Transformer-Transducer (T-T) (Zhang et al., 2020) as the backbone in the proposed TAED system. For the streaming setting, the speech encoder is based on the chunkwise implementation (Chiu and Raffel, 2017; Chen et al., 2020), which receives and computes new speech input data by chunk size N instead of one frame each time.

3.1 TAED

TAED combines both Transducer and AED into one model, as illustrated in Figure 1(c). The speech Transformer encoder is shared between Transducer and AED models. The predictor in Transducer is replaced by the AED decoder. Outputs of the new predictor are results of both speech encoder

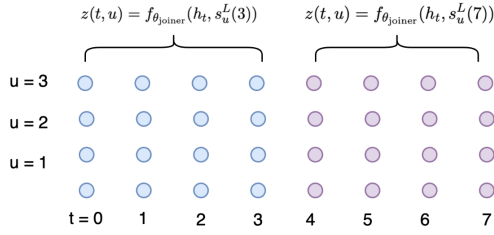


Figure 2: Chunk-based RNN-T synchronization. Frames from $t \in [0, 3]$ are from the first chunk (cyan) and $t \in [4, 7]$ belong to the second chunk (violet).

outputs and predicted tokens, hence they are more informative for the joiner.

Transducer and AED models are optimized together with two criteria, RNN-T loss for the Transducer’s joiner outputs and cross entropy loss for the AED decoder outputs. The overall loss $\mathcal{L}_{\text{taed}}$ is summation of two losses

$$\mathcal{L}_{\text{taed}} = \mathcal{L}_{\text{rnn-t}} + \mathcal{L}_{\text{aed}}. \quad (9)$$

The model is evaluated based on the outputs from the Transducer’s joiner.

3.2 Transducer optimization with chunk-based RNN-T synchronization scheme

When we attempt to extend TAED to the streaming scenario, we encounter the same streaming read/write issue discussed in §2.1. In order to avoid enumerating exponential increased alignments, we adopt a different approach and modify the inference logic to match the training and inference conditions.

In the conventional streaming decoder inference, when the new speech encoder output h_t is available, the new decoder state $s_u^l(t)$ is estimated via $h_{1:t}$ and previous computed decoder states $\hat{s}_{a'}^l$, which are based on $h_{1:t'}$ and $t' \leq t$, as shown in Eq. (5). In the proposed solution, we update all previous decoder states given speech encoder outputs $h_{1:t}$, and $\hat{s}_{a'}^l$ is replaced by $\hat{s}_{a(t)}^l$,

$$\hat{s}_{a(t)}^l = [s_1^l(t), \dots, s_{u-1}^l(t)], \quad (10)$$

where $a(t)$ stands for a special alignment where all tokens are aligned to timestamp $t_u = t$.

There are two reasons behind this modification. First, we expect the state representation would be more accurate if all decoder states are updated when more speech data is available. Second, the modification helps to reduce the huge number of alignments between $y_{1:U}$ and $h_{1:t}$ to one during

training, i.e., $a(t)$. Compared with the conventional AED training, it only increases the decoder forward computation by T times.

The computation is further reduced when the chunk-based encoder is used. Given two decoder states $s_u^l(t)$ and $s_u^l(\delta(t))$, where $\delta(t)$ is the last frame index of the chunk which frame t belongs to and $t \leq \delta(t)$, $s_u^l(\delta(t))$ is more informative than $s_u^l(t)$ since the former is exposed to more speech input. During inference, $s_u^l(t)$ and $s_u^l(\delta(t))$ are available at the same time when the speech chunk data is available. Therefore, we replace all $s_u^l(t)$ with corresponding $s_u^l(\delta(t))$ for both inference and training. If N is the number of speech encoder output frames from one chunk of speech input, chunk-based computation helps to reduce the decoder computation cost by N times during training, since we only need to update the decoder states every N frames instead of every frame. In summary, the computation of $s_u^l(t)$ is modified from Eq. (5) as below

$$s_u^l(t) = f_{\theta_{\text{dec}}}(h_{1:\delta(t)}, \hat{s}_{a(\delta(t))}^l, y_{u-1}), \quad (11)$$

and the joiner output $z(t, u)$ in Eq. (6) is updated as

$$z(t, u) = f_{\theta_{\text{joiner}}}(h_t, s_u^L(t)). \quad (12)$$

The chunk-based RNN-T synchronization is depicted in Figure 2. The number of speech encoder output frames in one chunk is 4. $z(t, u)$ s in the first chunk (cyan) are calculated with $s_u^L(3)$ and the second chunk (violet) is based on $s_u^L(7)$.

When the chunk size N is longer than the utterance length T , the chunk-based TAED streaming training is the same as the offline training with similar computation cost as Transducer.

3.3 AED optimization with fast alignment

Besides optimizing the model with the RNN-T loss aforementioned, we also include another auxiliary loss \mathcal{L}_{aed} for the AED model, as shown in Eq. (9). A straightforward approach is to optimize the AED modules as an offline AED model as Eq. (1). However, an offline AED model could lead to high latency if it is used for streaming applications. Hence, we also introduce a simple “streaming”-mode AED training by creating an even alignment a^e of the target tokens against the speech encoder outputs, i.e., $t_u^e = \lfloor u * T' / U \rfloor$, where $\lfloor x \rfloor$ is the floor operation on x . Furthermore, we can manipulate the alignment pace with an alignment speedup factor $\lambda > 0$, and the new alignment a^λ is with timestep

$t_u^\lambda = \max(T, \lfloor \frac{u * T'}{U * \lambda} \rfloor)$. When $\lambda > 1.0$, the streaming AED model is trained with a fast alignment and is encouraged to predict new tokens with less speech data. On the other hand, if $\lambda < \frac{1}{U}$, then $t_u^\lambda = T$ and it is equivalent to the offline training. The auxiliary AED task is optimized via

$$\mathcal{L}_{\text{aed}} = - \sum_{(\mathbf{x}, \mathbf{y})} \sum_u \log p(y_u | y_{1:u-1}, h_{1:t_u^\lambda}). \quad (13)$$

Note an accurate alignment is not required in this approach, which could be difficult to obtain in translation-related applications.

3.4 Blank penalty during inference

The ratio between the number of input speech frames and the number of target tokens could be varied due to many factors, such as different speech ratios or target language units. In the meantime, the prediction of a blank token \emptyset indicates a read operation, and a non-blank token represents a write operation, as discussed in §2.2. During inference, a blank penalty τ is introduced to adjust the target token fertility rate by penalizing the blank token \emptyset emission probability. It acts as word insertion penalty used in ASR (Takeda et al., 1998):

$$\hat{e}(t, u)[i_\emptyset] = e(t, u)[i_\emptyset] - \tau, \quad (14)$$

where $e(t, u) = \text{LOGSOFTMAX}_u(W^{\text{out}}z(t, u))$ and i_\emptyset is the index for the blank token \emptyset .

4 Comparison of streaming algorithms

When to read new input and write new output is a fundamental question for the streaming algorithm. Based on the choices of streaming read/write policies, they can roughly be separated into two families: pre-fixed and adaptive. The pre-fixed policy, such as Wait- k (Ma et al., 2019), adopts a fixed scheme to read new input and write new output. On the other hand, the adaptive algorithms choose read/write policies dynamically based on the input speech data presented. The adaptive algorithms could be further separated into two categories based on input and output synchronization.

The first category of adaptive streaming algorithms is based on the AED framework, including hard monotonic attention (HMA) (Raffel et al., 2017), MILk (Arivazhagan et al., 2019), MoChA (Chiu and Raffel, 2017), MMA (Ma et al., 2020b) and continuous integrate-and-fire (CIF) (Dong and Xu, 2020; Chang and yi Lee,

2022). Those methods extract acoustic information from the encoder outputs via attention between the encoder and decoder. The acoustic information is fused with linguistic information, which is estimated from the decoded token history, within the decoder. There is no explicit alignment between the input and output sequence; in other words, the outputs are **asynchronized** for the inputs. As discussed in §2.1, AED models don't fit the streaming application easily, and approximations have been taken during training. For example, the alignment-dependent context vector extracted via attention between the encoder and decoder is usually replaced by a context vector expectation from alignments. It differs from inference, which is based on a specific alignment path sampled during decoding. Hence a training and inference discrepancy is inevitable, potentially hurting the streaming performance.

The second category of adaptive streaming methods is with **synchronized** inputs and outputs, in which every output token is associated with a speech input frame. This includes CTC, Transducer, CAAT, and the proposed TAED. They combine acoustic and linguistic information within the joiner if linguistic modeling is applied. Specific read/write decisions are not required during training. This considers all alignments and is optimized via CTC loss or RNN-T loss. Hence, there is no training and inference discrepancy. The detailed comparison of different methods is listed in Table 1.

5 Experiments

5.1 Experimental setup

Data Experiments are conducted on two MUST-C (Gangi et al., 2019) language pairs: English to German (EN→DE) and English to Spanish (EN→ES). Sequence level knowledge distillation (Kim and Rush, 2016) is applied to boost the ST quality (Liu et al., 2021). The English portion of data in the EN→ES direction is used for English ASR development and evaluation. The models are developed on the dev set, and the final results are reported on the tst-COMMON set. We also report LIBRISPEECH (Panayotov et al., 2015) ASR results in Appendix C for convenient comparison with other ASR systems.

Evaluation The ASR quality is measured with word error rate (WER), and the ST quality is reported by case-sensitive detokenized BLEU, which is based on the default SACREBLEU options (Post,

Method	Synchronization	Merge Module	R/W decision	Training/Inference
CIF (Dong and Xu, 2020)	Async	Decoder	$h_{\leq j}$	sampling+scaling/sampling
HMA (Raffel et al., 2017)	Async	Decoder	$h_j, s_{\leq i}$	expectation/sampling
MILk (Arivazhagan et al., 2019)	Async	Decoder	$h_{\leq j}, s_{< i}$	expectation/sampling
MMA (Ma et al., 2020b)	Async	Decoder	$h_{\leq j}, s_{< i}$	expectation/sampling
CTC (Graves et al., 2006)	Sync	None	h_j	all paths/sampling
Transducer (Graves, 2012)	Sync	Joiner	$h_j, s_{k < i}$	all paths/sampling
CAAT (Liu et al., 2021)	Sync	Joiner	$h_{\leq j}, s_{< i}$	all paths/sampling
TAED (this work)	Sync	Predictor, Joiner	$h_{\leq j}, s_{< i}$	all paths/sampling

Table 1: Comparison of different streaming methods. ‘‘R/W decision’’ column lists information needed for R/W decision.

2018)³. Latency is measured with Average Lagging (AL) (Ma et al., 2019) using SimualEval (Ma et al., 2020a).

Model configuration Input speech is represented as 80-dimensional log mel-filterbank coefficients computed every 10ms with a 25ms window. Global channel mean and variance normalization is applied. The SpecAugment (Park et al., 2019) data augmentation with the LB policy is applied in all experiments. The target vocabulary consists of 1000 ‘‘unigram’’ subword units learned by SentencePiece (Kudo and Richardson, 2018) with full character coverage of all training text data.

We choose the Transformer-Transducer (T-T) (Zhang et al., 2020) as our Transducer baseline model. The speech encoder starts with two casual convolution layers with a kernel size of three and a stride size of two. The input speech features are down-sampled by four and then processed by 16 chunk-wise Transformer layers with relative positional embedding (Shaw et al., 2018). For the streaming case, the speech encoder can access speech data in all chunks before and one chunk ahead of the current timestep (Wu et al., 2020; Shi et al., 2020; Liu et al., 2021). We sweep over chunk size from 160ms to 640ms. For the offline model, we simply set a chunk size larger than any utterance to be processed as discussed in §3.2. There are two Transformer layers in the predictor module. The Transformer layers in both the speech encoder and predictor have an input embedding size of 512, 8 attention heads, and middle layer dimension 2048. The joiner module is a feed-forward neural network as T-T (Zhang et al., 2020). The TAED follows the same configuration as the T-T baseline, except the predictor module is replaced by an AED decoder with extra attention modules to connect the outputs from the speech encoder. The total number of pa-

³case.mixed+numrefs.?+smooth.exp+tok.none+version.1.5.1

Model	BLEU (↑)	
	EN→DE	EN→ES
AED (Wang et al., 2020)	22.7	27.2
AED (Inaguma et al., 2020)	22.9	28.0
CAAT (Liu et al., 2021)	23.1	27.6
Transducer	24.9	28.0
TAED	25.7	29.6

Table 2: Comparison of offline ST on the MUST-C t_{st} -COMMON set.

rameters is approximately 59M for both Transducer and TAED configurations.

Hyper-parameter setting The model is pre-trained with the ASR task using the T-T architecture. The trained speech encoder is used to initialize the TAED models and the T-T based ST model. The models are fine-tuned up to 300k updates using 16 A100 GPUs. The batch size is 16k speech frames per GPU. It takes approximately one day to train the offline model and three days for the streaming model due to the overhead of the lookahead chunk and chunk-based synchronization scheme. Early stopping is adopted if the training makes no progress for 20 epochs. The RAdam optimizer (Liu et al., 2020) with a learning rate 3e-4 is employed in all experiments. Label smoothing and dropout rate are both set to 0.1. We choose blank penalty τ by grid search within [0, 4.0] with step=0.5 on the dev set. The models are trained with FAIRSEQ (Wang et al., 2020). The best ten checkpoints are averaged for inference with greedy search (beam size=1).

5.2 Offline results

The results for the offline models are listed in Table 2 and Table 3. In Table 2, our models are compared with systems reported using MUST-C data only. The first two rows are based on the AED framework, and the third one is the results from CAAT, which is the backbone in the

Model	WER (\downarrow)	
	dev	tst-COMMON
Transducer	14.3	12.7
TAED	11.9	10.9

Table 3: Comparison of offline ASR on the MUST-C dev and tst-COMMON sets.

IWSLT2021 (Anastasopoulos et al., 2021) and IWSLT2022 (Anastasopoulos et al., 2022) streaming winning systems. Our Transducer baseline achieves competitive results and is comparable with the three systems listed above. The quality improves by 0.8 to 1.6 BLEU after we switch to the proposed TAED framework. Table 3 demonstrates the corresponding ASR quality, and TAED achieves 14% relative WER reduction compared with the Transducer baseline on the tst-COMMON set.

The results indicate Transducer can achieve competitive results with the AED based model in the ST task. A predictor conditioned with speech encoder outputs could provide a more accurate representation for the joiner. The TAED can take advantage of both the Transducer and AED and achieve better results.

In the next experiment, we compare the impact of the AED task weight for the offline model. In Eq. (9), the RNN-T loss and AED cross entropy loss are added to form the overall loss during training. In Table 4, we vary the AED task weight during training from 0.0 to 2.0. The 2nd, 3rd, and 4th columns correspond to the AED task weight, ASR WER, and ST BLEU in the “EN→ES” direction, respectively. AED weight 0.0 indicates only RNN-T loss is used while AED weight = 1.0 is equivalent to the proposed method in Eq. (9). Without extra guidance from the AED task (AED weight=0.0), the models still outperform the Transducer models in both ASR and ST tasks, though the gain is halved. When the AED task is introduced during training, i.e., AED weight is above 0, we get comparable results for three AED weights: 0.5, 1.0, and 2.0. This demonstrates that the AED guidance is essential, and the task weight is not very sensitive for the final results. In the following streaming experiments, we follow Eq. (9) without changing the AED task weight.

5.3 Streaming results

We first study the impact of the AED alignment speedup factor described in §3.3 in Table 5 and Ta-

Model	AED wts.	WER (\downarrow)	BLEU (\uparrow)
Transducer	–	12.7	28.0
TAED	0.0	11.9	28.9
	0.5	10.9	30.1
	1.0	10.9	29.6
	2.0	10.8	30.2

Table 4: Comparison of the TAED models trained with different AED weights on the MUST-C tst-COMMON set. “BLEU” stands for the ST results from “EN→ES” and “WER” column includes corresponding ASR results.

ble 6. In those experiments, the chunk size is set to 320ms. The ASR results are presented in Table 5. The first row indicates the alignment speedup factor λ . “Full” means the AED model is trained as an offline ST model. “1.0” stands for the alignment created by evenly distributing tokens along the time axis. The streaming TAED model trained with the offline AED model (“Full”) achieves 12.7 WER with a large latency. We examine the decoding and find the model tends to generate the first non-blank token near the end of the input utterance. The joiner learns to wait to generate reliable outputs at the end of utterances and tends to ignore the direct speech encoder outputs. When the fast AED alignment is adopted, i.e., $\lambda \geq 1.0$, the latency is reduced significantly from almost 6 seconds to less than 1 second. The larger λ is, the smaller AL is. One surprising finding is that both WER and latency become smaller when λ increases. The WER improves from 14.7 to 12.5 when λ increases from 1.0 to 1.2, slightly better than the TAED trained with the offline AED module. We hypothesize that the joiner might achieve better results if it gets a synchronized signal from both the speech encoder and AED decoder outputs. When λ is small, i.e., 1.0, AED decoder output might be lagged behind the speech encoder output when they are used to predict the next token.

A similar observation is also found in ST as demonstrated in Table 6 that the fast AED alignment helps to reduce TAED latency, though the best BLEU are achieved when the offline AED module is used. Compared to TAED models trained with offline AED module, the latency is reduced from 4+ seconds to less than 1.2 seconds for both translation directions, at the expense of BLEU score decreasing from 0.9 (EN→ES) to 1.8 (EN→DE).

In the following experiments, we compare the quality v.s. latency for TAED and Transducer. We

λ	Full	1.0	1.2	1.4
WER (\downarrow)	12.7	14.7	12.5	12.7
AL (\downarrow)	5894	1654	849	907

Table 5: Comparison of AED alignment speedup factor impact for the streaming ASR performance on the MUST-C EN t_{st} -COMMON set.

λ	EN \rightarrow ES		EN \rightarrow DE	
	BLEU (\uparrow)	AL (\downarrow)	BLEU (\uparrow)	AL (\downarrow)
Full	28.3	4328	24.1	4475
1.0	27.1	1715	22.8	1611
1.2	27.6	1228	22.6	1354
1.4	27.6	1120	22.3	1208

Table 6: Comparison of AED alignment speedup factor impact for the streaming ST performance on the MUST-C t_{st} -COMMON set. We set chunk size to 320ms for both EN \rightarrow ES and EN \rightarrow DE.

build models with different latency by changing the chunk size from 160, 320, and 480 to 640 ms. We present the WER v.s. AL curve in Figure 3. The dash lines are the WERs from the offline models, and the solid lines are for the streaming models. The figure shows that the proposed TAED models achieve better WER than the corresponding Transducer model, varied from 1.2 to 2.1 absolute WER reduction, with similar latency values.

The BLEU v.s. AL curves for ST are demonstrated in Figure 4 and Figure 5 for EN \rightarrow ES and EN \rightarrow DE directions, respectively. Besides the results from Transducer and TAED, we also include CAAT results from Liu et al. (2021) for convenient comparison. First, TAED consistently outperforms Transducer at different operation points in the EN \rightarrow ES direction and is on par with Transducer in the EN \rightarrow DE direction. We expect the TAED model outperforms the Transducer model for the EN \rightarrow DE direction when more latency budget is given since the offline TAED model is clearly better than the corresponding offline Transducer model. Second, CAAT performs better at the extremely low latency region (~ 1 second AL), and TAED starts to excel CAAT when AL is beyond 1.1 seconds for EN \rightarrow ES and 1.3 seconds for EN \rightarrow DE. TAED achieves higher BLEU scores than the offline CAAT model when the latency is more than 1.4 seconds for both directions. The detailed results are included in Appendix B.

6 Related work

Given the advantages and weaknesses of AED and CTC/ Transducer, many works have been done to

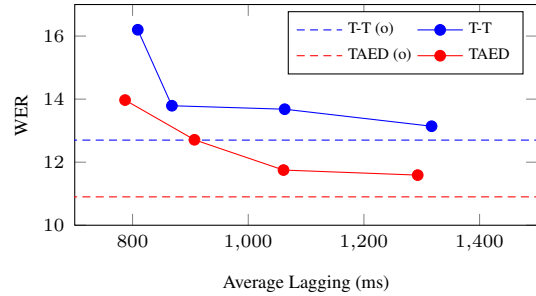


Figure 3: WER (\downarrow) v.s. Average Lagging (\downarrow) on the MUST-C EN t_{st} -COMMON dataset ($\lambda = 1.4$). “o” stands for the offline model.

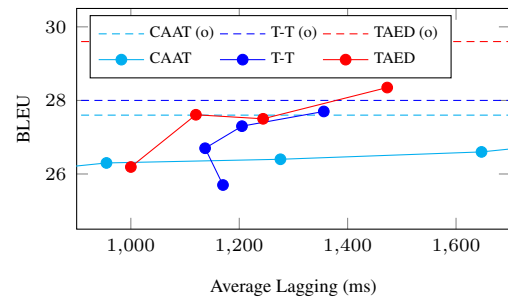


Figure 4: BLEU (\uparrow) v.s. Average Lagging (\downarrow) on the MUST-C EN \rightarrow ES t_{st} -COMMON dataset ($\lambda = 1.4$)

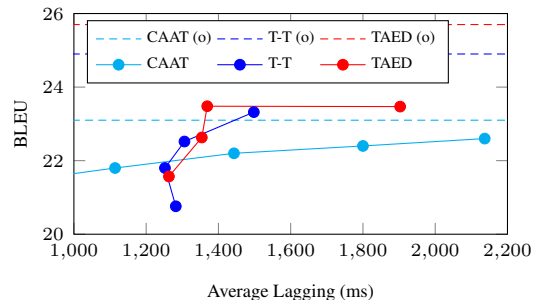


Figure 5: BLEU (\uparrow) v.s. Average Lagging (\downarrow) on the MUST-C EN \rightarrow DE t_{st} -COMMON dataset ($\lambda = 1.2$)

combine those methods together.

Transducer with attention (Prabhavalkar et al., 2017), which is a Transducer variant, also feeds the encoder outputs to the predictor. Our method is different in two aspects. First, we treat the TAED as a combination of two different models: Transducer and AED. They are optimized with equal weights during training, while Transducer with attention is optimized with RNN-T loss only. It is critical to achieve competitive results as shown in Table 4. Second, our method also includes a streaming solution while Transducer with attention can only be applied to the offline modeling.

Another solution is to combine those two methods through a two-pass approach (Watanabe et al., 2017; Sainath et al., 2019; Moriya et al., 2021).

The first pass obtains a set of complete hypotheses using beam search. The second pass model rescores these hypotheses by combining likelihood scores from both models and returns the result with the highest score. An improvement along this line of research replaces the two-pass decoding with single-pass decoding, which integrates scores from CTC/Transducer with AED during the beam search (Watanabe et al., 2017; Yan et al., 2022). However, sophisticated decoding algorithms are required due to the synchronization difference between two methods. They also lead to high computation cost and latency (Yan et al., 2022). Furthermore, the two-pass approach doesn't fit streaming applications naturally. Heuristics methods such as triggered decoding are employed (Moritz et al., 2019; Moriya et al., 2021). In our proposed solution, two models are tightly integrated with native streaming support, and TAED predictions are synergistic results from two models.

7 Conclusion

In this work, we propose a new framework to integrate Transducer and AED models into one model. The new approach ensures that the optimization covers all read/write paths and removes the discrepancy between training and evaluation for streaming applications. TAED achieves better results than the popular AED and Transducer modelings in ASR and ST offline tasks. Under the streaming scenario, the TAED model consistently outperforms the Transducer baseline in both the EN ASR task and EN→ES ST task while achieving comparable results in the EN→DE direction. It also excels the SOTA streaming ST system (CAAT) in medium and large latency regions.

8 Limitations

The TAED model has slightly more parameters than the corresponding Transducer model due to the attention modules to connect the speech encoder and AED decoder. They have similar training time for the offline models. However, the optimization of the streaming model would require more GPU memory and computation time due to the chunk-based RNN-T synchronization scheme described in §2.1. In our experiments, the streaming TAED model takes about three times more training time than the offline model on the 16 A100 GPU cards, each having 40GB of GPU memory.

In this work, we evaluate our streaming ST al-

gorithms on two translation directions: EN→ES and EN→DE. The word ordering for English and Spanish languages are based on Subject-Verb-Object (SVO) while German is Subject-Object-Verb (SOV). The experiments validate the streaming algorithms on both different word ordering pair and similar word ordering pair. Our future work will extend to other source languages besides English and more language directions.

References

- Antonios Anastasopoulos, Loïc Barrault, Luisa Bentivogli, Marceley Zanon Boito, Ondřej Bojar, Roldano Cattoni, Anna Currey, Georgiana Dinu, Kevin Duh, Maha Elbayad, et al. 2022. Findings of the IWSLT 2022 evaluation campaign. In *International Workshop on Spoken Language Translation*.
- Antonios Anastasopoulos, Ondřej Bojar, Jacob Bremerman, Roldano Cattoni, Maha Elbayad, Marcello Federico, Xutai Ma, Satoshi Nakamura, Matteo Negri, Jan Niehues, Juan Miguel Pino, Elizabeth Salesky, Sebastian Stüker, Katsuhito Sudoh, Marco Turchi, Alexander H. Waibel, Changhan Wang, and Matthew Wiesner. 2021. Findings of the IWSLT 2021 evaluation campaign. In *International Workshop on Spoken Language Translation*.
- N. Arivazhagan, Colin Cherry, Wolfgang Macherey, Chung-Cheng Chiu, Semih Yavuz, Ruoming Pang, Wei Li, and Colin Raffel. 2019. Monotonic infinite lookback attention for simultaneous machine translation. In *ACL*.
- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *ICLR*.
- A. Berard, O. Pietquin, C. Servan, and L. Besacier. 2016. Listen and translate: A proof of concept for end-to-end speech-to-text translation. In *NIPS*.
- William Chan, Navdeep Jaitly, Quoc V. Le, and Oriol Vinyals. 2015. Listen, attend and spell. *ArXiv*, abs/1508.01211.
- Chih-Chiang Chang and Hung yi Lee. 2022. Exploring continuous integrate-and-fire for adaptive simultaneous speech translation. In *Interspeech*.
- Xie Chen, Yu Wu, Zhenghao Wang, Shujie Liu, and Jinyu Li. 2020. Developing real-time streaming transducer for speech recognition on large-scale dataset. In *ICASSP*, pages 5904–5908.
- Chung-Cheng Chiu, Wei Han, Yu Zhang, Ruoming Pang, Sergey Kishchenko, Patrick Nguyen, Arun Narayanan, Hank Liao, Shuyuan Zhang, Anjali Kannan, Rohit Prabhavalkar, Z. Chen, Tara N. Sainath, and Yonghui Wu. 2019. A comparison of end-to-end models for long-form speech recognition. In *ASRU*.

- Chung-Cheng Chiu and Colin Raffel. 2017. Monotonic chunkwise attention. In *ICLR*.
- Kyunghyun Cho and Masha Esipova. 2016. Can neural machine translation do simultaneous translation? *ArXiv*, abs/1606.02012.
- J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio. 2015. Attention-based models for speech recognition. In *NIPS*.
- Shun-Po Chuang, Yung-Sung Chuang, Chih-Chiang Chang, and Hung yi Lee. 2021. Investigating the re-ordering capability in CTC-based non-autoregressive end-to-end speech translation. In *Findings of ACL*.
- Linhao Dong and Bo Xu. 2020. CIF: Continuous integrate-and-fire for end-to-end speech recognition. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6079–6083.
- Mattia Antonino Di Gangi, Roldano Cattoni, Luisa Bentivogli, Matteo Negri, and Marco Turchi. 2019. MuST-C: a multilingual speech translation corpus. In *NAACL-HLT*.
- Alex Graves. 2012. Sequence transduction with recurrent neural networks. *ArXiv*, abs/1211.3711.
- Alex Graves, Santiago Fernández, Faustino J. Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*.
- Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, and Ruoming Pang. 2020. Conformer: Convolution-augmented transformer for speech recognition. In *Interspeech*.
- H. Inaguma, S. Kiyono, K. Duh, S. Karita, N. Sproll, T. Hayashi, and S. Watanabe. 2020. ESPnet-ST: All-in-one speech translation toolkit. In *ACL*.
- Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Conference on Empirical Methods in Natural Language Processing*.
- T. Kudo and J. Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *EMNLP*.
- Jinyu Li. 2021. Recent advances in end-to-end automatic speech recognition. *APSIPA Transactions on Signal and Information Processing*.
- Xian Li, Changan Wang, Yun Tang, C. Tran, Yuqing Tang, Juan Miguel Pino, Alexei Baevski, Alexis Conneau, and Michael Auli. 2021. Multilingual speech translation from efficient finetuning of pretrained models. In *ACL/IJCNLP*.
- Dan Liu, Mengge Du, Xiaoxi Li, Ya Li, and Enhong Chen. 2021. Cross attention augmented transducer networks for simultaneous translation. In *EMNLP*.
- Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. 2020. On the variance of the adaptive learning rate and beyond. In *ICLR*.
- Mingbo Ma, Liang Huang, Hao Xiong, Renjie Zheng, Kaibo Liu, Baigong Zheng, Chuanqiang Zhang, Zhongjun He, Hairong Liu, Xing Li, Hua Wu, and Haifeng Wang. 2019. STACL: Simultaneous translation with implicit anticipation and controllable latency using prefix-to-prefix framework. In *ACL*.
- Xutai Ma, Mohammad Javad Dousti, Changan Wang, Jiatao Gu, and Juan Pino. 2020a. SIMULEVAL: An evaluation toolkit for simultaneous translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics.
- Xutai Ma, Juan Pino, James Cross, Liezl Puzon, and Jiatao Gu. 2020b. Monotonic multihead attention. In *ICLR*.
- Niko Moritz, Takaaki Hori, and Jonathan Le Roux. 2019. Triggered attention for end-to-end speech recognition. In *ICASSP*.
- Takafumi Moriya, Tomohiro Tanaka, Takanori Ashihara, Tsubasa Ochiai, Hiroshi Sato, Atsushi Ando, Ryo Masumura, Marc Delcroix, and Taichi Asami. 2021. Streaming end-to-end speech recognition for hybrid RNN-T/attention architecture. In *Interspeech*.
- Vassil Panayotov, Guoguo Chen, D. Povey, and S. Khudanpur. 2015. Librispeech: An asr corpus based on public domain audio books. In *ICASSP*.
- Sara Papi, Marco Gaido, Matteo Negri, and Marco Turchi. 2022. Over-generation cannot be rewarded: Length-adaptive average lagging for simultaneous speech translation. *ArXiv*, abs/2206.05807.
- D. Park, W. Chan, Y. Zhang, C. Chiu, B. Zoph, E. Cubuk, and Q. Le. 2019. SpecAugment: A simple data augmentation method for automatic speech recognition. In *Interspeech*.
- Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*.
- Rohit Prabhavalkar, Kanishka Rao, Tara N. Sainath, Bo Li, Leif M. Johnson, and Navdeep Jaitly. 2017. A comparison of sequence-to-sequence models for speech recognition. In *Interspeech*.
- Colin Raffel, Minh-Thang Luong, Peter J. Liu, Ron J. Weiss, and Douglas Eck. 2017. Online and linear-time attention by enforcing monotonic alignments. In *ICML*.

- Tara N. Sainath, Ruoming Pang, David Rybach, Yanzhang He, Rohit Prabhavalkar, Wei Li, Mirkó Visontai, Qiao Liang, Trevor Strohman, Yonghui Wu, Ian McGraw, and Chung-Cheng Chiu. 2019. Two-pass end-to-end speech recognition. In *Interspeech*.
- Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *North American Chapter of the Association for Computational Linguistics*.
- Yangyang Shi, Yongqiang Wang, Chunyang Wu, Ching feng Yeh, Julian Chan, Frank Zhang, Duc Le, and Michael L. Seltzer. 2020. Emformer: Efficient memory transformer based acoustic model for low latency streaming speech recognition. In *ICASSP*.
- K. Takeda, Atsunori Ogawa, and Fumitada Itakura. 1998. Estimating entropy of a language from optimal word insertion penalty. In *ICSLP*.
- Yun Tang, Hongyu Gong, Ning Dong, Changan Wang, Wei-Ning Hsu, Jiatao Gu, Alexei Baevski, Xian Li, Abdelrahman Mohamed, Michael Auli, and Juan Miguel Pino. 2022. Unified speech-text pre-training for speech translation and recognition. In *Annual Meeting of the Association for Computational Linguistics*.
- Yun Tang, Juan Miguel Pino, Changan Wang, Xutai Ma, and Dmitriy Genzel. 2021. A general multi-task learning framework to leverage text data for speech to text tasks. In *ICASSP*.
- Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.
- Changan Wang, Yun Tang, Xutai Ma, Anne Wu, Dmytro Okhonko, and Juan Miguel Pino. 2020. Fairseq S2T: Fast speech-to-text modeling with fairseq. In *AAACL*.
- Peidong Wang, Eric Sun, Jian Xue, Yu Wu, Long Zhou, Yashesh Gaur, Shujie Liu, and Jinyu Li. 2022. LAMASSU: Streaming language-agnostic multilingual speech recognition and translation using neural transducers. *ArXiv*, abs/2211.02809.
- Shinji Watanabe, Takaaki Hori, Suyoun Kim, John R. Hershey, and Tomoki Hayashi. 2017. Hybrid CTC/attention architecture for end-to-end speech recognition. *IEEE Journal of Selected Topics in Signal Processing*, 11:1240–1253.
- R. Weiss, J. Chorowski, N. Jaitly, Y. Wu, and Z. Chen. 2017. Sequence-to-sequence models can directly translate foreign speech. In *Interspeech*.
- Chunyang Wu, Yongqiang Wang, Yangyang Shi, Ching feng Yeh, and Frank Zhang. 2020. Streaming transformer-based acoustic models using self-attention with augmented memory. In *Interspeech*.
- Jian Xue, Peidong Wang, Jinyu Li, Matt Post, and Yashesh Gaur. 2022. Large-scale streaming end-to-end speech translation with neural transducers. In *Interspeech*.
- Brian Yan, Siddharth Dalmia, Yosuke Higuchi, Graham Neubig, Florian Metze, Alan W. Black, and Shinji Watanabe. 2022. CTC alignments improve autoregressive translation. *ArXiv*, abs/2210.05200.
- Qian Zhang, Han Lu, Hasim Sak, Anshuman Tripathi, Erik McDermott, Stephen Koo, and Shankar Kumar. 2020. Transformer transducer: A streamable speech recognition model with transformer encoders and rnn-t loss. In *ICASSP*, pages 7829–7833.

MUST-C	hours	#W(m)
EN-DE	408	4.2
EN-ES	504	5.2

Table 7: Data statistics for MUST-C training dataset. “#W(m)” stands for “number of words (million)”.

CS(ms)	WER	AL	LAAL	AP	DAL
160	16.20	809	856	0.63	1177
320	13.79	868	914	0.65	1254
480	13.68	1063	1102	0.70	1485
640	13.14	1317	1350	0.71	1755

Table 8: Transducer WER v.s. latency for different chunk sizes.

A Statistics of the MUST-C dataset

We conduct experiments on the MUST-C (Gangi et al., 2019). The ASR experiments are based on the English portion of data in the EN→ES direction. The ST experiments are conducted in two translation directions: EN→ES and EN→DE. The detailed training data statistics are presented in Table 7. The second column is the total number of hours for the speech training data. The third column is the number of (source) words.

B Detailed streaming results

The detailed streaming experimental results are presented in this section. We report different latency metrics from SimulEval toolkit (Ma et al., 2020a), including Average Lagging (AL) (Ma et al., 2019), Average Proportion (AP) (Cho and Esipova, 2016), Differentiable Average Lagging (DAL) (Arivazhagan et al., 2019), and Length Adaptive Average Lagging (LAAL) (Papi et al., 2022). AL, DAL and LAAL are reported with million seconds. We report the evaluation results based on different chunk size, varied from 160, 320, 480 and 640 million seconds, from Table 8 to Table 13. “CS” in those tables stands for chunk size. Streaming ASR results are reported as WER (Table 8 and Table 9). BLEU scores are reported for two translation directions in Table 10, Table 11, Table 12 and Table 13.

C Librispeech ASR results

The model configure is the same as **MuST-C** experiments in §5.1. The models are trained with 16 A100 GPUs with batch size 20k speech frames per GPU for 300k updates. SpecAugment (Park

CS(ms)	WER	AL	LAAL	AP	DAL
160	13.97	787	829	0.63	1146
320	12.71	907	947	0.65	1284
480	11.75	1061	1097	0.69	1482
640	11.59	1293	1323	0.74	1742

Table 9: TAED WER v.s. latency for different chunk sizes($\lambda = 1.4$).

CS(ms)	BLEU	AL	LAAL	AP	DAL
160	25.7	1170	1379	0.70	1669
320	26.7	1137	1357	0.73	1679
480	27.3	1205	1433	0.74	1784
640	27.7	1356	1578	0.77	1951

Table 10: Transducer BLEU v.s. latency for different chunk sizes (EN→ES).

CS(ms)	BLEU	AL	LAAL	AP	DAL
160	26.19	1000	1224	0.71	1623
320	27.61	1120	1351	0.74	1795
480	27.50	1244	1477	0.77	1930
640	28.35	1473	1693	0.79	2188

Table 11: TAED BLEU v.s. latency for different chunk sizes (EN→ES)($\lambda = 1.4$).

CS(ms)	BLEU	AL	LAAL	AP	DAL
160	20.76	1282	1412	0.68	1618
320	21.80	1252	1389	0.70	1612
480	22.52	1306	1447	0.72	1717
640	23.32	1498	1630	0.75	1925

Table 12: Transducer BLEU v.s. latency for different chunk sizes (EN→DE).

CS(ms)	BLEU	AL	LAAL	AP	DAL
160	21.57	1263	1411	0.72	1823
320	22.63	1354	1530	0.74	2007
480	23.48	1369	1554	0.77	2088
640	23.47	1903	2024	0.82	2597

Table 13: TAED BLEU v.s. latency for different chunk sizes (EN→DE)($\lambda = 1.2$).

model	test		dev	
	clean	other	clean	other
Transducer(off)	3.2	8.0	3.0	8.2
Transducer(str)	4.4	11.2	4.0	11.3
TAED(off)	3.1	7.4	2.9	7.4
TAED(str)	4.2	10.4	4.1	10.8

Table 14: Comparison of offline and streaming ASR on the Librispeech datasets. “off” and “str” stand for offline and streaming models respectively.

et al., 2019) is without time warping and dropout set to 0.1. We save the checkpoints every 2500 updates and the best 10 checkpoints are averaged for the greedy search based inference. The model are trained on the 960 hours **Librispeech** (Panayotov et al., 2015) training set and evaluated on 4 test/dev sets. In Table 14, the streaming models (“str”) are trained with chunk size equals to 320ms with one right look-ahead chunk. TAED obtains similar WERs in two clean (easy) datasets and reduces WER varied by 0.5 to 0.8 in two other (hard) datasets.

ACL 2023 Responsible NLP Checklist

A For every submission:

- A1. Did you describe the limitations of your work?
section 8
- A2. Did you discuss any potential risks of your work?
section 8
- A3. Do the abstract and introduction summarize the paper’s main claims?
abstract and section 1
- A4. Have you used AI writing assistants when working on this paper?
Left blank.

B Did you use or create scientific artifacts?

We conduct experiments on MuST-C dataset and the algorithm is implemented on the top of fairseq as described in section 5

- B1. Did you cite the creators of artifacts you used?
section 5
- B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Both of data (MuST-C) and software (fairseq) are open sourced and widely used by other researchers for speech translation study.
- B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Both of data (MuST-C) and software (fairseq) are open sourced and widely used by other researchers for speech translation study. Our work is falling into the same category.
- B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
It was discussed in the MuST-C paper.
- B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
section 5
- B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
In appendix

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

C Did you run computational experiments?

section 5

- C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?

section 5

- C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?

section 5

- C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?

section 5

- C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?

section 5

D Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

- D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?

No response.

- D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?

No response.

- D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?

No response.

- D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?

No response.

- D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?

No response.