Cross Attention Augmented Transducer Networks for Simultaneous Translation

Dan Liu[1,2], Mengge Du[2], Xiaoxi Li[2], Ya Li[2] and Enhong Chen [1]

[1]University of Science and Technology of China, Hefei, China
[2]iFlytek Research, Hefei, China
danliu@mail.ustc.edu.cn , *cheneh@ustc.edu.cn*
{danliu,mgdou,xxli8,yali8}@iflytek.com

Abstract

This paper proposes a novel architecture, Cross Attention Augmented Transducer (CAAT), for simultaneous translation. The framework aims to jointly optimize the policy and translation models. To effectively consider all possible READ-WRITE simultaneous translation action paths, we adapt the online automatic speech recognition (ASR) model, RNN-T, but remove the strong monotonic constraint, which is critical for the translation task to consider reordering. To make CAAT work, we introduce a novel latency loss whose expectation can be optimized by a forward-backward algorithm. We implement CAAT with Transformer while the general CAAT architecture can also be implemented with other attention-based encoder-decoder frameworks. Experiments on both speech-to-text (S2T) and text-to-text (T2T) simultaneous translation tasks show that CAAT achieves significantly better latency-quality trade-offs compared to the state-of-the-art simultaneous translation approaches. [1]

1 Introduction

Simultaneous translation, which starts to translate input sentences before they are finished, is of importance to many real-life applications such as teleconference systems and time-sensitive spoken document analysis and conversion. While a substantial progress has been made on offline machine translation (Wu et al., 2016; Vaswani et al., 2017; Hassan et al., 2018), more research on simultaneous translation is yet highly desirable. Central to the task is performing high-quality low-latency translation, which involves the key challenges of developing optimal policies for the READ-WRITE action paths as well as generating high-quality target sequences based only on partial source sequences.

This paper aims to optimize the policy and translation model jointly, by expanding target sequences with *blank* symbols for READ actions. The loss function can be defined as negative log-likelihood (NLL) of marginal distribution through all expanded paths. A similar problem in automatic speech recognition (ASR) has been tackled with RNN-T (Recurrent Neural Network Transducer) (Graves, 2012) by an efficient forward-backward algorithm. However, RNN-T is trained based on the monotonic alignment between source and target sequences, which is not suitable for simultaneous translation, as it cannot properly consider reordering. On the other hand, the forward-backward algorithm is not available for attention-based encoder-decoder (Bahdanau et al., 2015) architectures, including Transformer (Vaswani et al., 2017), due to the deep coupling between source contexts and target history contexts.

To solve this problem, we separate the cross attention mechanism from target history representation in attention-based encoder-decoder, which can also be viewed as RNN-T with the joiner being augmented by cross attention mechanism, resulting in Cross Attention Augmented Transducer (CAAT). However, cross attention mechanism removes the alignment constraint in RNN-T which originally encourages an appropriate latency. To ensure latency under control, jointly minimizing a latency loss is required. Both the NLL loss and latency loss can be efficiently optimized by a forward-backward algorithm.

The main contributions of this paper are threefold: (1) We propose a novel architecture, Cross Attention Augmented Transducer, which jointly optimizes the policy and translation model by considering all possible READ-WRITE simultaneous

[1]The code is available at https://github.com/danliu2/caat.

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 39–55
November 7–11, 2021.

translation action paths. (2) We introduce a novel latency loss whose expectation can be optimized by a forward-backward algorithm. Training with this latency loss ensures the latency of CAAT simultaneous translation model to be under control. (3) The proposed model achieves significantly better latency-quality trade-offs compared to the state-of-the-art simultaneous translation approaches.

2 Related Work

Recent work on simultaneous translation falls into two categories. The first category uses a fixed policy for the READ/WRITE actions. Cho and Esipova (2016) propose simultaneous translation with the *wait-if-** policy for an offline model. Ma et al. (2019) propose a *wait-k* policy for both the training and inference period. The second category includes models with a flexible policy learned and/or adaptive to current context. Gu et al. (2017) introduce an agent trained by reinforcement learning from the interaction with a pre-trained offline neural machine translation model. Zheng et al. (2019a) train the agent by supervise learning with label sequences generated via the rank of golden target words given partial input. A special subcategory of flexible policy jointly optimize policy and translation by monotonic attention customized to translation model, e.g., Monotonic Infinite Lookback (MILk) attention (Arivazhagan et al., 2019) on RNN encoder-decoder (Bahdanau et al., 2015) and Monotonic Multihead Attention (MMA) (Ma et al., 2020c) on Transformer (Vaswani et al., 2017).

End-to-end speech-to-text (S2T) simultaneous translation has been investigated in (Ma et al., 2020b,d; Ren et al., 2020), among which Ma et al. (2020b) adapt latency metrics from T2T simultaneous translation to S2T simultaneous translation, and experiment with both the fixed and flexible policy. Ma et al. (2020d) study the effect of speech block processing on S2T simultaneous translation. Ren et al. (2020) experiment with the *wait-k* policy based on a source language CTC segmenter.

In our work, we optimize the marginal distribution of all expanded paths motivated by RNN-T (Graves, 2012). Unlike RNN-T, the CAAT model removes the monotonic constraint, which is critical for considering reordering in machine translation tasks. The optimization of our latency loss is motivated by Sequence Discriminative Training in ASR (Povey, 2005).

3 Preliminaries

3.1 Notations and formulation

Let $\mathbf{x}$ and $\mathbf{y}$ denote the source sequence and target sequence, and f and g the encoder and decoder function, respectively. For simultaneous translation, let a_j denotes the length of source sequence processed when deciding the target y_j. The policy of simultaneous translation is denoted as an action sequence $p \in \{R, W\}^{|\mathbf{x}|+|\mathbf{y}|}$ where R denotes the READ action and W the WRITE action. If the READ action is replaced with a *blank* symbol $\varnothing$, the policy can also be represented by the expanded target sequence $\hat{y} \in (V \cup \{\varnothing\})^{|\mathbf{x}|+|\mathbf{y}|}$, where V is the vocabulary of the target language. Note that removing all $\varnothing$ in $\hat{y}$ results in the original target sequence $\mathbf{y}$. The mapping from $\mathbf{y}$ to sets of all possible expansion $\hat{y}$ is denoted as $H(\mathbf{x}, \mathbf{y})$.

3.2 Recurrent Neural Network Transducer

RNN-T (Graves, 2012) draws condition probability $\Pr(\mathbf{y}|\mathbf{x})$ by marginalizing all possible alignment paths as :

$$\begin{aligned}\Pr(y|x) &= \sum_{\hat{y} \in H(x,y)} \Pr(\hat{y}|x) \\ &= \sum_{\hat{y} \in H(x,y)} \prod_{k=1}^{|\mathbf{x}|+|\mathbf{y}|} \Pr(\hat{y}_k|i_k, j_k)\end{aligned} \tag{1}$$

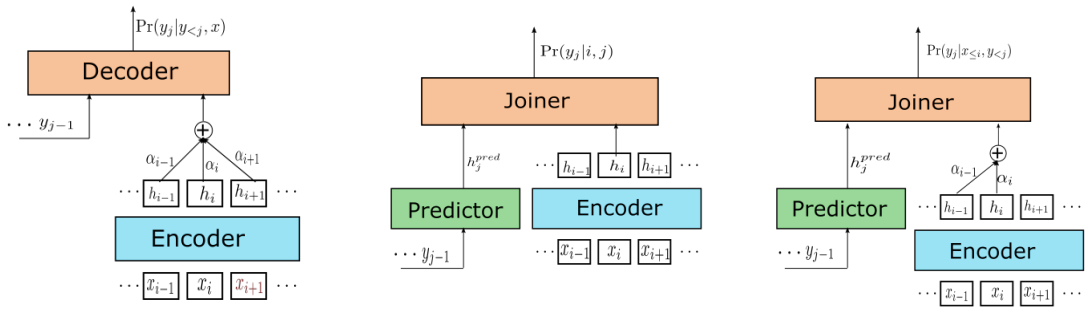
where i_k and j_k denote the source and target position of the k-th element in $\hat{\mathbf{y}}$, respectively, and $\hat{\mathbf{y}} = (\hat{y}_1, \hat{y}_2, ..., \hat{y}_{|\mathbf{x}|+|\mathbf{y}|}) \in H(\mathbf{x}, \mathbf{y}) \subset \{\mathcal{V} \cup \varnothing\}^{|\mathbf{x}|+|\mathbf{y}|}$ corresponds to a possible expansion path which yields $\mathbf{y}$ after removing the *blank* symbol $\varnothing$.

As shown in Figure 1(b), to calculate $P(\hat{y}_k|h_{i_k}, y_{<j_k})$, RNN-T divides decoder into *predictor* and *joiner*, where the predictor, denoted f^{pred}, produces target history representation (Eq. (2)), and the joiner products output probability $\Pr(y|i, j)$ by joint representations from predictor and encoder (Eq. (3)).

$$h_j^{pred} = f^{pred}(y_{<j}) \tag{2}$$

$$\Pr(y|i, j) = softmax(W^h h_i + W^p h_j^{pred}) \tag{3}$$

Though named as RNN Transducer, other sequence processing architectures work well as the encoder or predictor, e.g., Transformer (Zhang et al., 2020; Yeh et al., 2019). Online decoding is natural for RNN-T if the encoder works with streaming input, which makes RNN-T widely



(a) Attention-based Encoder-Decoder (b) RNN-T (c) CAAT

Figure 1: The difference between Attention-based Encoder-Decoder, RNN-T and CAAT.

adopted in both the online and offline ASR tasks. One drawback of RNN-T is that it is based on a monotonic alignment between the input and output sequence, making it unsuitable for sequence-to-sequence tasks with reordering, e.g., machine translation.

4 The Proposed Method

4.1 Cross Attention Augmented Transducer

4.1.1 Model Architecture

The goal of simultaneous translation is to achieve high translation quality and low latency. A natural loss function hence measures the NLL loss of marginal conditional distribution and expectation of latency metric through all possible expanded paths:

$$\begin{aligned}\mathcal{L}(x,y) &= \mathcal{L}_{NLL}(x,y) + \mathcal{L}_{latency}(x,y) \\ &= -\log \sum_{\hat{y}} \Pr(\hat{y}|x) + \mathbb{E}_{\hat{y}} l(\hat{y}) \\ &= -\log \sum_{\hat{y}} \Pr(\hat{y}|x) + \sum_{\hat{y}} \Pr(\hat{y}|y,x) l(\hat{y})\end{aligned} \tag{4}$$

where $\Pr(\hat{y}|y,x) = \frac{\Pr(\hat{y}|x)}{\sum_{\hat{y}' \in H(x,y)} \Pr(\hat{y}'|x)}$, $\hat{y} \in H(x,y)$ is one of the expanded paths of the target sequence $\mathbf{y}$, and $l(\hat{y})$ is the latency loss for path $\hat{y}$.

As the total number of expanded paths is exponential with regard to $|x| + |y|$, computing the marginal probability $\sum_{\hat{y} \in H(x,y)} \Pr(\hat{y}|x)$ is non-trivial. RNN-T solves this with a forward-backward algorithm (Graves, 2012), which inherently requires paths in the graph to be *mergeable*. That is, the representations of the same location in different paths should be identical. Conventional attention-based encoder-decoder architectures as shown in Figure 1(a), however, do not satisfy this requirement. Take Figure 2 as an example, the decoder hidden states for the red path $\hat{y}^1$ and the blue path $\hat{y}^2$ are described below (we denotes s_i^n as the representation of the i-th decoder step in the expanded path $\hat{y}^n$) :

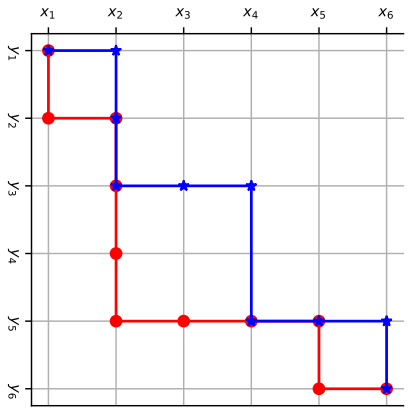


Figure 2: Expanded paths in simultaneous translation

$$\begin{aligned} s_2^1 &= g(s_1^1, h_{\leq 2}) \\ &= g(g(s_0, h_{\leq 2}), h_{\leq 2}) \end{aligned} \tag{5a}$$

$$\begin{aligned} s_2^2 &= g(s_1^2, h_{\leq 2}) \\ &= g(g(s_0, h_{\leq 1}), h_{\leq 2}) \end{aligned} \tag{5b}$$

The decoder states at output step 2 with different history paths, s_2^1 and s_2^2, are not identical. This is due to the coupling of source and previous target representation by the attention mechanism in the decoder. The same problem exists in Transformer, from the coupling of self-attention and encoder-decoder cross attention in each block.

To solve this, we separate the cross attention mechanism from the target history representation, which is similar to the joiner and predictor in RNN-T. The novel architecture, as shown in Figure 1(c),

can be viewed as an extended version of RNN-T with the joiner augmented by cross attention mechanism, and is named as Cross Attention Augmented Transducer (CAAT). Different from RNN-T, the joiner in CAAT is a complex architecture with attention mechanisms as in Eq. (6):

$$s_{i,j} = s(attn(h_j^{pred}, h_{\leq i}^{enc}), h_j^{pred}) \tag{6}$$

Note that $s_{i,j}$ is independent of previous nodes $s_{i',j'}$ in path $\hat{\mathbf{y}}$, and the same location from different paths in Figure 2 produces the same state representation. By analyzing the diffusion of the output probability through the lattice in Figure 2, we can find that $\Pr(\mathbf{y}|\mathbf{x})$ is equal to the sum of probabilities over any top-right to bottom-left diagonal nodes. Defining the *forward variable* $\alpha(i,j)$ as the probability of outputting $\mathbf{y}_{[1:j]}$ during $x_{[1:i]}$, and the *backward variable* $\beta(i,j)$ as the probability of outputting $y_{[j+1:|\mathbf{y}|]}$ during $x_{[i:|\mathbf{x}|]}$, we can draw the marginal likelihood $\Pr(\mathbf{y}|\mathbf{x})$ as :

$$\begin{aligned} \Pr(\mathbf{y}|\mathbf{x}) &= \sum_{(i,j):i+j=m} \alpha(i,j) \cdot \beta(i,j) \\ \mathcal{L}_{NLL}(x,y) &= -\log \Pr(\mathbf{y}|\mathbf{x}) \end{aligned} \tag{7}$$

where $1 \leq m \leq |\mathbf{x}| + |\mathbf{y}|$. The detailed derivation of NLL loss of CAAT can be found in Appendix A.1.

The proposed CAAT can be implemented with a variety of attention-based encoder-decoder frameworks. In this paper, we implemented CAAT with Transformer, by dividing Transformer's decoder into the predictor and joiner module. As shown in Figure 3, the predictor and joiner share the same number of transformer blocks as the conventional transformer decoder, but there are no cross-attention blocks in the predictor module and no self-attention blocks in the joiner.

4.1.2 Multi-Step Decision

The CAAT architecture gains the ability of handling source-target reordering at the cost of an expensive joiner. The complexity of joiner is $\mathcal{O}(|\mathbf{x}| \cdot |\mathbf{y}|)$ during training. For RNN-T, the joiner is efficient because only softmax operates at $\mathcal{O}(|\mathbf{x}| \cdot |\mathbf{y}|)$. But for CAAT, joiner takes up half of the parameters of decoder, which means the complexity of CAAT is about $\frac{|\mathbf{x}|}{4}$ times higher than the conventional encoder-decoder framework during training.

RNN-T needs to ensure the output timing of y_j is the corresponding source frame $a_j = align(x, y_j)$.

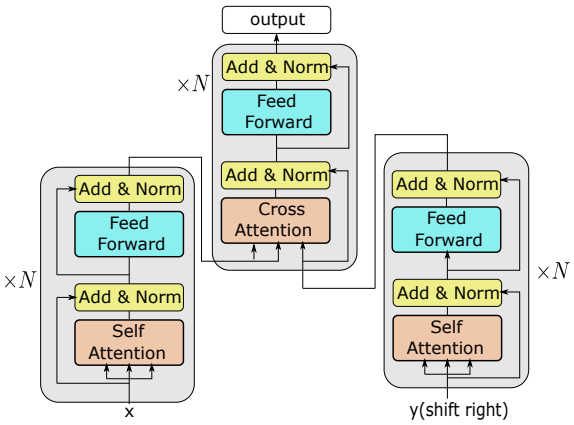


Figure 3: Architecture of CAAT Transformer

However, based on attention mechanism, CAAT only needs to ensure output timing to be *after* the corresponding position ($a_j \geq align(x, y_j)$). Therefore, it is no longer necessary to make decision each encoder frame; the decision step size $d > 1$ is appropriate for CAAT, which reduces the complexity of the joiner from $\mathcal{O}(|\mathbf{x}| \cdot |\mathbf{y}|)$ to $\mathcal{O}\left(\frac{|\mathbf{x}| \cdot |\mathbf{y}|}{d}\right)$. Besides, the decision step size is also an effective way to adjust latency-quality trade-off.

4.1.3 Latency Loss

CAAT relaxes the restriction of output timing by attention mechanism, which means all source step $i \geq align(x, y_j)$ should be appropriate for output y_j, including the offline path ($\forall j : a_j = |\mathbf{x}|$). To avoid the CAAT model bypassing online policy by choosing the offline path, the latency loss $\mathcal{L}_{latency}(x, y)$ as defined in Eq. (4) is required.

Motivated by Sequence Criterion Training in ASR (Povey, 2005), we optimize the latency loss with the forward-backward algorithm. To calculate the expectation of latency loss through all paths $\hat{\mathbf{y}}$, *mergeable* is also a requirement to the latency loss definition, which means the latency loss through path $\hat{y}$ may be defined as $l(\hat{y}) = \sum_{k=1}^{|\mathbf{x}|+|\mathbf{y}|} l(\hat{y}_k)$ and $l(\hat{y}_k)$ is independent of $l(\hat{y}_{k' \neq k})$. However, both Average Lagging (Ma et al., 2019) and Differentiable Average Lagging (Arivazhagan et al., 2019) do not meet this requirement. We hence introduce a novel latency function as follows:

$$l(i,j) = \frac{1}{|\mathbf{y}|} \max\left(i - j \cdot \frac{|\mathbf{x}|}{|\mathbf{y}|}, 0\right) \tag{8}$$

$$l(\hat{y}_k) = \begin{cases} 0 & \text{if } \hat{y}_k = \varnothing \\ l(i_k, j_k) & \text{else} \end{cases} \tag{9}$$

$$l(\hat{y}) = \sum_{k=1}^{|\hat{\mathbf{y}}|} l(\hat{y}_k) \tag{10}$$

where $i_k = \sum_{k'=1}^{k} I(\hat{y}_{k'} = \varnothing)$ and $j_k = \sum_{k'=1}^{k} I(\hat{y}_{k'} \neq \varnothing)$ denote the number of READ and WRITE actions before $\hat{y}_k$, respectively. The maximization operation is used to avoid encouraging over-aggressive decision paths. This latency definition is not rigorous enough to be an evaluation metric for the under-estimation after source ended, as analyzed in (Arivazhagan et al., 2019), but it can still be used as a loss function.

By defining the *forward latency variable* $\alpha^{lat}(i,j)$ as the expectation of latency of outputting $\mathbf{y}_{[1:j]}$ during $x_{[1,2,\cdots,i]}$, and the *backward latency variable* $beta^{lat}(i,j)$ as the expectation of latency of outputting $\mathbf{y}_{[j+1:|\mathbf{y}|]}$ during decision steps $\mathbf{x}_{[i,\cdots,|\mathbf{x}|]}$, the latency loss can be drawn as:

$$\begin{aligned} c(i,j) &= \alpha^{lat}(i,j) + \beta^{lat}(i,j) \\ \mathcal{L}_{latency}(x,y) &= \mathbb{E}_{\hat{y} \in H(x,y)} l(\hat{y}) \\ &= \frac{\sum_{(i,j):i+j=m} \alpha(i,j)\beta(i,j)c(i,j)}{\Pr(\mathbf{y}|\mathbf{x})} \end{aligned}$$

where $1 \leq m \leq |\mathbf{x}| + |\mathbf{y}|$. The detailed derivation of latency loss of CAAT can be found in Appendix A.2.

4.1.4 Offline Auxiliary Loss

We add the negative log-likelihood loss of the offline translation path as an auxiliary loss to CAAT model training for two reasons. First, we hope the CAAT model falls back to offline translation in the worst case; second, the CAAT translation is carried out in accordance with offline translation when a source sentence finishes. The final loss function for CAAT training is defined as follows:

$$\begin{aligned} \mathcal{L}(x,y) &= \mathcal{L}_{NLL}(x,y) + \lambda_{latency}\mathcal{L}_{latency}(x,y) \\ &\quad + \lambda_{offline}\mathcal{L}_{offline}(x,y) \\ &= -\log \sum_{\hat{y}} \Pr(\hat{y}|x) \\ &\quad + \lambda_{latency} \sum_{\hat{y}} \Pr(\hat{y}|y,x) l(\hat{y}) \\ &\quad - \lambda_{offline} \log P_{offline}(y|x) \end{aligned} \tag{11}$$

where $\lambda_{latency}$ and $\lambda_{offline}$ are the scaling factors corresponding to $\mathcal{L}_{latency}$ and $\mathcal{L}_{offline}$, respectively. And we set $\lambda_{latency} = \lambda_{offline} = 1.0$ if not specified.

4.2 Streaming Encoder

Unidirectional Transformer encoder (Arivazhagan et al., 2019; Ma et al., 2020c) is not effective for speech data processing, because of the close relatedness to the right context for speech feature x_i. Block processing (Dong et al., 2019; Wu et al., 2020) is introduced for online ASR, but it lacks direct observation to infinite left context.

We process the streaming encoder for speech data by block processing with the right context and infinite left context. First, input representations $\mathbf{h}$ is divided into overlapped blocks with block shift step m and block size $m + r$. Each block consists of two parts, the main context $\mathbf{m}_n = \left[h_{m*n+1}, \cdots, h_{m*(n+1)}\right]$ and the right context $\mathbf{r}_n = \left[h_{(n+1)*m+1}, \cdots, h_{(n+1)*m+r}\right]$. The query, key, and value of block $\mathbf{b}_n$ in self-attention can be described as follows:

$$\mathbf{Q} = \mathbf{W}_q \left[\mathbf{m}_n, \mathbf{r}_n\right] \tag{12}$$

$$\mathbf{K} = \mathbf{W}_k \left[\mathbf{m}_1, \cdots, \mathbf{m}_n, \mathbf{r}_n\right] \tag{13}$$

$$\mathbf{V} = \mathbf{W}_v \left[\mathbf{m}_1, \cdots, \mathbf{m}_n, \mathbf{r}_n\right] \tag{14}$$

By reorganizing the input sequence and designed self-attention mask, training is effective by reusing conventional transformer encoder layers. And unidirectional transformer can be regarded as a special case of our method with $\{m = 1, r = 0\}$. Note that the look-ahead window size in our method is fixed, which enables us to increase transformer layers without increasing latency. We set the main context size and right context size to 8 and 4, respectively, for our experiments on speech-to-text simultaneous translation, and conventional unidirectional transformer encoder $\{m = 1, r = 0\}$ for

experiments on text-to-text simultaneous translation.

4.3 Inference of CAAT Simultaneous Translation

The online inference for CAAT is adapted from beam search for RNN-T (Graves, 2012), and the changes are as follows[2] : (1) We only merge paths between decision steps, as the cost of the joiner of CAAT is significantly more expensive than that of RNN-T. (2) We extract common prefix of existing hypotheses as determined target output at each decision time step. (3) Different beam sizes are introduced for intra-decision (b_1) and inter-decision (b_2) pruning, to ensure timely determination of outputs. b_1 and b_2 are set to be 5 and 1, respectively, if not otherwise specified.

5 Experiments

5.1 Speech-to-Text Simultaneous Translation

5.1.1 Experiment Setup

Datasets and Models We use the MuST-C v1.0[3] (Di Gangi et al., 2019) English→German (EN→DE) and English→Spanish (EN→ES) speech translation datasets in our experiments. We use the dev set for validation and report performance on the tst-COMMON set. The 80-dimensional log-Mel filter bank features are extracted for speech feature with a $25ms$ window size and a $10ms$ window shift; SpecAugment (Park et al., 2019) were performed on the training data. We use SentencePiece (Kudo and Richardson, 2018) to generate a unigram vocabulary of size 20,000 for the source and target language jointly. Our experiments on speech-to-text simultaneous translation are based on Transformer (Vaswani et al., 2017). Since the variance of the length of speech frames is more significant than that of text length, we use both cosine positional embedding (Vaswani et al., 2017) and relative positional attention (Shaw et al., 2018) for speech encoder, and only cosine positional embedding for the decoder. Detailed hyper-parameters of our models can be found in Appendix C.1.

Training and Inference Training speech translation models is often regarded to be more difficult than training text machine translation or ASR models. We use two methods to improve the performance and stability of model training. The first is to pre-train encoder with ASR task (Ma et al., 2020b), and the second is to leverage sequence-level knowledge distillation with text machine translation model (Ren et al., 2020).

Training CAAT models require significantly larger GPU memory than that used in conventional Transformer due to the spatial complexity $\mathcal{O}(\frac{|\mathbf{x}||\mathbf{y}|}{d})$ of the joiner module; we solve this by splitting hidden states into small pieces before sending them into the joiner and recombining them during back-propagation.

Our implementation is based on the Fairseq library (Ott et al., 2019); the NLL and latency loss for CAAT are implemented based on warp-rnnt [4].

Evaluation We evaluate our models with SimulEval (Ma et al., 2020a). Translation quality is measured by detokenized case-sensitive BLEU (Papineni et al., 2002); latency is measured with the adapted version of *word-level* Average Lagging (AL) (Ma et al., 2020a).

5.1.2 Results

We compare CAAT to the current state-of-the-art model in speech-to-text simultaneous translation (Ma et al., 2020b), which uses *wait-k* with a fixed pre-decision step size of 320ms. All our simultaneous speech translation models, both *wait-k* and CAAT are trained with encoder pretrained on ASR task and sequence-level knowledge distillation with text translation model. Two inference methods are used for *wait-k*, conventional beam search only on target tail (when source finishes) and speculative beam search (SBS) (Zheng et al., 2019b), both with a beam size of 5; the forecast steps in SBS is set to be 2. For CAAT we set the intra-decision beam size $b_1 = 5$ and inter-decision beam size $b_2 = 1$ as described in Sec. 4.3. The latency-quality curves of CAAT are produced by varying decision step size $d \in \{8, 16, 32, 48, 64, 80, +\infty\}$, and *wait-k* by varying $k \in \{1, 2, 4, 6, 8, 10, 12, +\infty\}$. The AL-BLEU curves on the MuST-C EN→DE and EN→ES test sets are shown in Figure 4.[5]

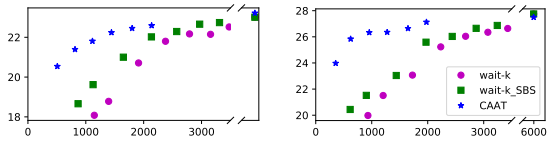
From the figure we can observe that: (1) In general CAAT significantly outperforms *wait-k* (with and without SBS) in both the EN→DE and

[2]Details of inference algorithm can be found in Appendix C.

[3]https://ict.fbk.eu/must-c/

[4]https://github.com/1ytic/warp-rnnt

[5]Full-size graphs for all latency metrics (AL, AP, and DAL) along with the corresponding numeric scores are available in Appendix D.



(a) EN→DE (b) EN→ES

Figure 4: Translation quality vs. Average Lagging on the EN→DE and EN→ES speech translation test set. The y-axis is BLEU and x-axis Average Laggging (AL).

EN→ES task. Especially in the low-latency region ($AL < 1000ms$) (Ansari et al., 2020), CAAT outperforms *wait-k* with SBS by more than 3 BLEU points. (2) The Offline models of CAAT and *wait-k* obtain similar BLEU, suggesting that the adapted architecture of CAAT performs comparably with conventional Transformer in an offline scenario. (3) With the same wait step k, SBS can produce lower latency. This is due to the word-level latency metrics we used requires an additional token to ensure complete word submitted, which can be offset by the forward exploration in SBS.

5.1.3 Ablation Study

Effectiveness of Streaming Encoder The performance of our offline models with full-sentence encoder compared to the state-of-the-art offline speech translation systems (Wang et al., 2020; Inaguma et al., 2020) are demonstrated in Table 1. We also show the ablation analyses on sequence-level knowledge distillation with text translation model (KD) and pretrain encoder with ASR task (Pretrain).

Model	EN→DE	EN→ES
(Wang et al., 2020)	22.7	27.2
(Inaguma et al., 2020)	22.9	28.0
Full-Sentence Encoder	24.3	28.6
-KD	22.4	26.8
-Pretrain	20.2	25.1
$\{m = 1, r = 0\}$ (unidirectional)	22.0	25.3
$\{m = 8, r = 0\}$	22.4	26.3
$\{m = 4, r = 4\}$	22.8	27.3
$\{m = 8, r = 4\}$	23.0	27.8
$\{m = 12, r = 4\}$	23.1	27.6

Table 1: Offline translation performance of different streaming methods on the MuST-C EN→DE and EN→ES test set.

We further compare offline translation models with streaming encoders to those with the conventional full-sentence encoder. As shown in Table 1, the performance of the translation model with a unidirectional encoder drops 2-3 BLEU points compared to that with a full-sentence encoder, and the gap is gradually narrowed by the increase of main block size m and introduction of right context. Considering the effect on latency, we choose $\{m = 8, r = 4\}$.

# Joiner layers	d	AL	BLEU
0	1	2715.7	9.74
0	32	2154.4	9.42
1	32	1156.2	20.94
4	32	1141.2	21.78
6	32	1114.9	21.81

Table 2: Effect of the number of joiner layers on quality and latency on the MuST-C EN→DE test set.

Effectiveness of Joiner Layers The performance of CAAT models with different numbers of joiner layers are shown in Table 2. Note that in the table, the first two rows (# joiner layers=0) corresponds to the conventional Transducer without cross attention, in which encoder representations are downsampled d times using average-pooling and then directly fused with predictor outputs by addition. We can find that the introduction of the cross attention mechanism significantly improves the performance of simultaneous translation, and the BLEU scores are close when the number of joiner layers is greater than 4.

$\lambda_{offline}$	AL	BLEU
0	1111.6	19.84
0.5	1106.5	20.83
1	1114.9	21.81
1.5	1144.0	21.77
2.0	1176.5	21.87

Table 3: Effect of the $\lambda_{offline}$ on quality and latency on the MuST-C EN→DE test set.

Effectiveness of $\lambda_{latency}$ and $\lambda_{offline}$ The effectiveness of $\lambda_{offline}$ is demonstrated in Table 3. Furthermore, as shown in Figure 5, though $\lambda_{latency}$ may affect the trade-off between translation quality and latency, varying $\lambda_{latency}$ is not as effective as varying the decision step size d, and we

found the model training will be unstable when $\lambda_{latency} \geq 2.0$.

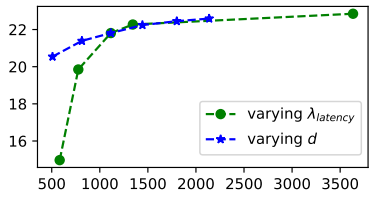


Figure 5: AL-BLEU curves drawn by varying the decision step size d and latency loss scale $\lambda_{latency}$ on the MuST-C EN→DE test set. *varying* d means setting $d = \{8, 16, 32, 48, 64, 80\}$ with $\lambda_{latency} = 1.0$; *varying* $\lambda_{latency}$ means setting $\lambda_{latency} = \{2, 1.5, 1, 0.5, 0\}$ with decision step size $d = 32$.

Effectiveness of Beam Search The effectiveness of the intra-decision beam size b_1 and inter-decision beam size b_2 on simulation translation performance is shown in Table 4. We can find that beam search in one decision step brings an improvement of about 0.7 BLEU over the greedy search. And if we allow multiple hypothesizes between decision steps we may get another 0.5 BLEU improvement at the cost of latency (AL increases from 1114.9 to 2433.5). However, this may be useful in the scenarios where revision is allowed (Arivazhagan et al., 2020), e.g., simultaneous translation for subtitle.

b_1	b_2	AL	BLEU
1	1	1116.0	21.18
3	1	1109.0	21.75
5	1	1114.9	21.81
10	1	1126.2	21.86
5	2	1929.7	22.1
5	3	2433.5	22.3

Table 4: Performance of CAAT models with different b_1 and b_2 on the EN→DE test set, all with the decision step $d = 32$.

Case Study We perform case study to demonstrate the advantages of CAAT model over *wait-k* with SBS, we compare *wait-k* $k = 2$ with CAAT $d = 32$ for they have similar AL latency. As shown in Figure 7, *wait-k* generates meaningless translation by 'predict' in the place of pauses and changes in speech rate, while CAAT does not suffer from this problem. As a result, CAAT outperforms *wait-k* with SBS.

5.2 Text-to-Text Simultaneous Translation

We further performed experiments on the text-to-text simultaneous translation task. Experiments are carried out on the WMT15 German-English (DE→EN) dataset with newstest2013 as the validation set and newstest2015 as the test set. We strictly follow the same settings of (Arivazhagan et al., 2019), and latency is measured with the *subword-level* latency metric. Detailed hyper-parameters of our models can be found in Appendix C.2. We compare CAAT models to *wait-k* with/without SBS, and Infinite Lookbak MMA (MMA-IL)[6] (Ma et al., 2020c). The latency-quality curves of CAAT are produced by varying decision step size $d \in \{1, 2, 4, 8, 12, 16, 20, +\infty\}$ [7], *wait-k* by varying $k \in \{2, 4, 6, 8, 10, 12, 16, +\infty\}$, and MMA-IL by varying weighted average latency loss scale $\lambda_{avg} \in \{0.8, 0.6, 0.4, 0.2, 0.1, 0.0\}$.

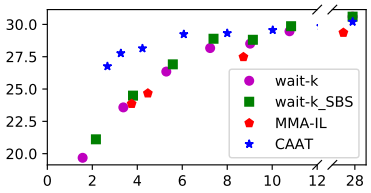


Figure 6: Translation quality vs. Average Lagging on the WMT15 DE→EN text translation test set.The y-axis is BLEU and x-axis Average Laggging (AL).

The AL-BLEU trade-off curves are shown in Figure 6.[8] We can see that CAAT outperforms *wait-k* and *wait-k* with SBS, but the gap is narrowing compared to that of S2T simultaneous translation in Figure 4. Considering the case analyze in Sec. 5.1.3, we believe that flexible policy is more important for speech translation because of the speech rate changing.

6 Conclusions

This paper proposes Cross Attention Augmented Transducer (CAAT), a novel simultaneous translation model that jointly optimizes policy and translation model by considering all possible READ-WRITE action paths. Crucial to the model is a

[6] https://github.com/pytorch/fairseq/tree/master/examples/simultaneous_translation

[7] Limited by GPU memory, we failed to train CAAT with $d < 4$, so we just set $d = \{1, 2\}$ in inference on model trained with $d = 4$.

[8] Full-size graphs for all latency metrics along with the corresponding numeric scores are available in Appendix D.

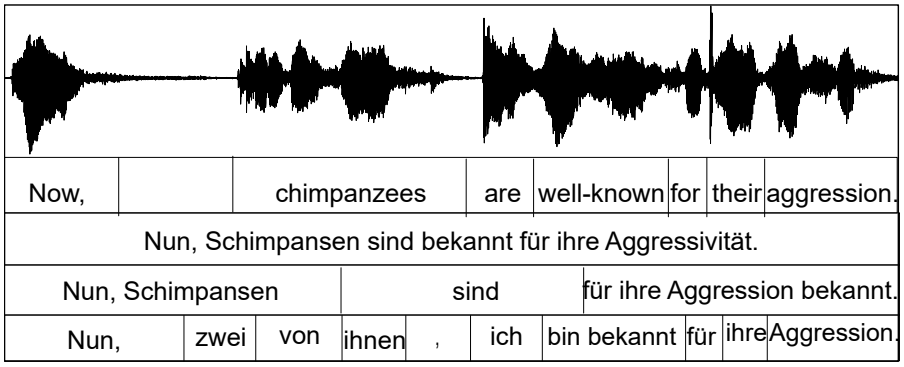


Figure 7: An example from the EN→DE test set, which demonstrate that CAAT outperforms *wait-k* with close latency. The five components from top to bottom: speech, source transcription (aligned to speech audio), reference translation, hypothesis from CAAT, and hypothesis from *wait-k* with SBS.

properly designed latency loss incorporated to ensure latency to be under control. Experiments demonstrate that CAAT achieves better latency-quality trade-offs compared to the state-of-the-art approaches in speech-to-text and text-to-text simultaneous translation tasks. We provide detailed analyses to demonstrate how CAAT works and improves the performance.

Acknowledgements

This paper is funded by the National Key R&D Program of China (No. 2018YFC0831601).

References

Ebrahim Ansari, Amittai Axelrod, Nguyen Bach, Ondřej Bojar, Roldano Cattoni, Fahim Dalvi, Nadir Durrani, Marcello Federico, Christian Federmann, Jiatao Gu, Fei Huang, Kevin Knight, Xutai Ma, Ajay Nagesh, Matteo Negri, Jan Niehues, Juan Pino, Elizabeth Salesky, Xing Shi, Sebastian Stüker, Marco Turchi, Alexander Waibel, and Changhan Wang. 2020. FINDINGS OF THE IWSLT 2020 EVALUATION CAMPAIGN. In *Proceedings of the 17th International Conference on Spoken Language Translation*, pages 1–34, Online. Association for Computational Linguistics.

Naveen 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 *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1313–1323, Florence, Italy. Association for Computational Linguistics.

Naveen Arivazhagan, Colin Cherry, Wolfgang Macherey, and George Foster. 2020. Re-translation versus streaming for simultaneous translation. In *Proceedings of the 17th International Conference on Spoken Language Translation*, pages 220–227, Online. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Kyunghyun Cho and Masha Esipova. 2016. Can neural machine translation do simultaneous translation? *arXiv preprint arXiv:1606.02012*.

Mattia A. Di Gangi, Roldano Cattoni, Luisa Bentivogli, Matteo Negri, and Marco Turchi. 2019. MuST-C: a Multilingual Speech Translation Corpus. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2012–2017, Minneapolis, Minnesota. Association for Computational Linguistics.

Linhao Dong, Feng Wang, and Bo Xu. 2019. Self-attention aligner: A latency-control end-to-end model for ASR using self-attention network and chunk-hopping. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019, Brighton, United Kingdom, May 12-17, 2019*, pages 5656–5660. IEEE.

Alex Graves. 2012. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*.

Jiatao Gu, Graham Neubig, Kyunghyun Cho, and Victor O.K. Li. 2017. Learning to translate in real-time with neural machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1053–1062, Valencia, Spain. Association for Computational Linguistics.

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt,

William Lewis, Mu Li, et al. 2018. Achieving human parity on automatic chinese to english news translation. *arXiv preprint arXiv:1803.05567*.

Hirofumi Inaguma, Shun Kiyono, Kevin Duh, Shigeki Karita, Nelson Yalta, Tomoki Hayashi, and Shinji Watanabe. 2020. ESPnet-ST: All-in-one speech translation toolkit. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 302–311, Online. Association for Computational Linguistics.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

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 *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3025–3036, Florence, Italy. Association for Computational Linguistics.

Xutai Ma, Mohammad Javad Dousti, Changhan 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*, pages 144–150, Online. Association for Computational Linguistics.

Xutai Ma, Juan Pino, and Philipp Koehn. 2020b. SimulMT to SimulST: Adapting simultaneous text translation to end-to-end simultaneous speech translation. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 582–587, Suzhou, China. Association for Computational Linguistics.

Xutai Ma, Juan Miguel Pino, James Cross, Liezl Puzon, and Jiatao Gu. 2020c. Monotonic multihead attention. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Xutai Ma, Yongqiang Wang, Mohammad Javad Dousti, Philipp Koehn, and Juan Pino. 2020d. Streaming simultaneous speech translation with augmented memory transformer. *arXiv preprint arXiv:2011.00033*.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le. 2019. Specaugment: A simple data augmentation method for automatic speech recognition. *arXiv preprint arXiv:1904.08779*.

Daniel Povey. 2005. *Discriminative training for large vocabulary speech recognition*. Ph.D. thesis, University of Cambridge.

Yi Ren, Jinglin Liu, Xu Tan, Chen Zhang, Tao Qin, Zhou Zhao, and Tie-Yan Liu. 2020. SimulSpeech: End-to-end simultaneous speech to text translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3787–3796, Online. Association for Computational Linguistics.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468, New Orleans, Louisiana. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Changhan Wang, Yun Tang, Xutai Ma, Anne Wu, Dmytro Okhonko, and Juan Pino. 2020. Fairseq S2T: Fast speech-to-text modeling with fairseq. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 33–39, Suzhou, China. Association for Computational Linguistics.

Chunyang Wu, Yongqiang Wang, Yangyang Shi, Ching-Feng Yeh, and Frank Zhang. 2020. Streaming transformer-based acoustic models using self-attention with augmented memory. *arXiv preprint arXiv:2005.08042*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Ching-Feng Yeh, Jay Mahadeokar, Kaustubh Kalgaonkar, Yongqiang Wang, Duc Le, Mahaveer Jain, Kjell Schubert, Christian Fuegen, and Michael L Seltzer. 2019. Transformer-transducer: End-to-end speech recognition with self-attention. *arXiv preprint arXiv:1910.12977*.

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 *2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2020, Barcelona, Spain, May 4-8, 2020*, pages 7829–7833. IEEE.

Baigong Zheng, Renjie Zheng, Mingbo Ma, and Liang Huang. 2019a. Simultaneous translation with flexible policy via restricted imitation learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5816–5822, Florence, Italy. Association for Computational Linguistics.

Renjie Zheng, Mingbo Ma, Baigong Zheng, and Liang Huang. 2019b. Speculative beam search for simultaneous translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1395–1402, Hong Kong, China. Association for Computational Linguistics.

A Derivation of CAAT Losses

A.1 Derivation of CAAT NLL loss

Given the encoder representation h_n, where $1 \leq n \leq |\mathbf{x}|$, the predictor vector h_j^{pred}, where $0 \leq j \leq J$ and $J = |\mathbf{y}|$. and decision step size $d \geq 1$. The maximum decision step is $I = \lceil \frac{|\mathbf{x}|}{d} \rceil$, and the output logits at decision step i, target position j should be

$$s(i,j) = g\left(h_{<i*d}, h_j^{pred}\right) \tag{15}$$

s(i,j) is a vector of $|V|+1$ dimension corresponding to V and *blank* symbol $\varnothing$. $s(k,i,j)$ denotes the k-th dimension of $s(i,j)$. The conditional output distribution can be yielded as :

$$\Pr(k|i,j) = \frac{e^{s(k,i,j)}}{\sum_{k'} e^{s(k',i,j)}} \tag{16}$$

To simplify notation, define

$$\begin{aligned} y(i,j) &:= \Pr(y_{j+1}|i,j) \\ \varnothing(i,j) &:= \Pr(\varnothing|i,j) \end{aligned} \tag{17}$$

Define the *forward variable* $\alpha(i,j)$ as the probability of outputting $\mathbf{y}_{[1:j]}$ during decision steps $[1,2,\cdots,i]$. The forward variables for all $1 \leq i \leq I$ and $0 \leq j \leq |\mathbf{y}|$ can be calculated recursively using

$$\begin{aligned} \alpha(i,j) = &\alpha(i-1,j) \cdot \varnothing(i-1,j) \\ &+ \alpha(i,j-1) \cdot y(i,j-1) \end{aligned} \tag{18}$$

with initial condition $\alpha(1,0) = 1$. The total output sequence probability is equal to the forward variable at the terminal node:

$$\Pr(\mathbf{y}|\mathbf{x}) = \alpha(I,J) \cdot \varnothing(I,J) \tag{19}$$

Define the *backward variable* $\beta(i,j)$ as the probability of outputting $\mathbf{y}_{[j+1:J]}$ during decision steps $[i,\cdots,I]$. Then:

$$\begin{aligned} \beta(i,j) = &\beta(i+1,j) \cdot \varnothing(i,j) \\ &+ \beta(i,j+1) \cdot y(i,j) \end{aligned} \tag{20}$$

with initial condition $\beta(I,J) = \varnothing(I,J)$. $\Pr(\mathbf{y}|\mathbf{x})$ is equal to the sum of $\alpha(i,j)\beta(i,j)$ over any top-right to bottom-left diagonal through the nodes. That is, $\forall m : 1 \leq m \leq I+J$

$$\Pr(y|x) = \sum_{(i,j):i+j=m} \alpha(i,j) \cdot \beta(i,j) \tag{21}$$

From Eqs. 18, 20 and 21, we can draw the derivation of loss function $\mathcal{L} = -\log \Pr(\mathbf{y}|\mathbf{x})$ as

$$\frac{\partial \mathcal{L}}{\partial \Pr(k|i,j)} = \frac{\alpha(i,j)}{\Pr(\mathbf{y}|\mathbf{x})} \begin{cases} \beta(i,j+1) & \text{if } k = y_{j+1} \\ \beta(i+1,j) & \text{if } k = \varnothing \\ 0 & \text{otherwise} \end{cases} \quad (22)$$

A.2 Derivation of CAAT Latency Loss

To calculate the marginal expectation in Eq. 23, we define *forward latency variable* $\alpha^{lat}(n,j)$ as the expectation latency of outputting $\mathbf{y}_{[1:j]}$ during decision steps $[1, 2, \cdots, i]$, and *backward latency variable* $beta^{lat}(i,j)$ as the expectation latency of outputting $\mathbf{y}_{[j+1:J]}$ during decision steps $[i, i+1, \cdots, i]$. Here we denote $l(n,j)$ as the latency function for output y_j at source position n.

$$\begin{aligned} \mathcal{L}_{latency}(x,y) &= \mathbb{E}_{\hat{y} \in H(x,y)} l(\hat{y}) \\ &= \sum_{\hat{y}} \Pr(\hat{y}|y,x) l(\hat{y}) \end{aligned} \quad (23)$$

The forward latency variables can be calculated recursively using

$$\begin{aligned} \alpha^{lat}(i,j) = &\, p_1(i,j) \cdot f_1(i,j) \\ &+ p_0(i,j) \cdot f_0(i,j) \end{aligned} \quad (24)$$

with initial condition $\alpha^{lat}(1,0) = 0$. Where

$$\begin{aligned} p_1(i,j) &= \frac{\alpha(i,j-1) \cdot y(i,j)}{\alpha(i,j)} \\ p_0(i,j) &= \frac{\alpha(i-1,j) \cdot \varnothing(i-1,j)}{\alpha(i,j)} \\ f_1(i,j) &= \alpha^{lat}(i,j-1) + l(i,j-1) \\ f_0(i,j) &= \alpha^{lat}(i-1,j) \end{aligned} \quad (25)$$

For backward latency variables

$$\begin{aligned} \beta^{lat}(i,j) = &\, q_1(i,j) \cdot b_1(i,j) \\ &+ q_0(i,j) \cdot b_0(i,j) \end{aligned} \quad (26)$$

with initial condition $\beta^{lat}(I,J) = 0$. Where

$$\begin{aligned} q_1(i,j) &= \frac{\beta(i,j+1) \cdot y(i,j)}{\beta(i,j)} \\ q_0(i,j) &= \frac{\beta(i+1,j) \cdot \varnothing(i,j)}{\beta(i,j)} \\ b_1(i,j) &= \beta^{lat}(i,j+1) + l(i,j) \\ b_0(i,j) &= \beta^{lat}(i+1,j) \end{aligned} \quad (27)$$

To simplify notation, define the latency expectation of all paths go through grid (n,j) as

$$\begin{aligned} c(i,j) &= \alpha^{lat}(i,j) + \beta^{lat}(i,j) \\ c(i,j,0) &= \alpha^{lat}(i,j) + \beta^{lat}(i+1,j) \\ c(i,j,1) &= \alpha^{lat}(i,j) + l(i,j) \\ &\quad + \beta^{lat}(i,j+1) \end{aligned} \quad (28)$$

The expectation latency for all paths $\hat{y} \in H(\mathbf{x},\mathbf{y})$ is equal to the expectation through diagonal nodes. That is, $\forall m : 1 \le m \le N+J$:

$$\begin{aligned} \hat{c} &= c(I,J) \\ &= \frac{\sum_{(i,j):i+j=m} \alpha(i,j)\beta(i,j)c(i,j)}{\Pr(\mathbf{y}|\mathbf{x})} \end{aligned} \quad (29)$$

And the latency loss $\mathcal{L}_{latency}(\mathbf{x},\mathbf{y}) = \hat{c}$. From Eqs. 24, 26, 28 and 29, it follows that:

$$\begin{aligned} &r(i,j) = \\ &\begin{cases} \beta(i,j+1)\,(c(i,j,1) - \hat{c}) & \text{if } k = y_{j+1} \\ \beta(i+1,j)\,(c(i,j,0) - \hat{c}) & \text{if } k = \varnothing \\ 0 & \text{otherwise} \end{cases} \\ &\frac{\partial \mathcal{L}_{latency}}{\partial \Pr(k|n,j)} = \frac{\alpha(i,j)}{\Pr(\mathbf{y}|\mathbf{x})} r(i,j) \end{aligned} \quad (30)$$

B Beam Search Algorithm for CAAT

The pseudo code of beam search algorithm for CAAT is described in Algorithm 1.

C Hyper-parameters

C.1 Hyper-parameters on Speech-to-Text Simultaneous Translation

Our experiments on speech-to-text simultaneous translation are based on Transformer. For speech processing two 2D convolution blocks are introduced before the stacked Transformer encoder layers. Each convolution block consists of a 3-by-3 convolution layer with 64 channels and stride size as 2, and a ReLU activation function. Input speech features are downsampled 4 times by convolution blocks and flattened to 1D sequence as input to transformer layers. Cosine positional embedding is added to speech representations after convolutions, and relative positional attention is employed for encoder self-attention. The hidden size, feed-forward hidden size, number of heads, number of encoder and decoder layers are set to 256, 2048, 4, 12, and

Algorithm 1 Beam search for CAAT

Require: intra-block beam b_1, inter-decision beam b_2, decision step size d
Initialise: $B = \{\varnothing\}, \Pr(\varnothing) = 1, submitted = 0$
for i=1 to I step d **do**
  $A = B, B = \{\}, S = \{\}$
  while B contains less than b_2 elements more probable than the most probable in A **do**
    for $\hat{y}$ in A **do**
      $y = H^{-1}(\hat{y}))$
      $s = \Pr(\hat{y})p(\varnothing|\hat{y}, i)$
      if y in B **then**
        $S(y) = \max(S(y), s)$
      else
        $S(y) = s$
      end if
      Add y to B
      for k in **y**) **do**
        $\Pr(\hat{y} + k) = \Pr(\hat{y}) \Pr(k|\hat{y}, i)$
        Add $y + k$ to A
      end for
    end for
    Remove all but b_1 most probable from A
    Remove all but top b_1 S from B
  end while
  Remove all but top b_2 $score$ from B
  to_submit= common_prefix(B)
  if length(to_submit) > submitted **then**
    submit(to_submit[submitted:])
    submitted=length(to_submit)
  end if
end for

6. The dropout ratio is set to 0.3. Our CAAT model shares the same hyper-parameters with the conventional Transformer model, except the feed-forward hidden sizes of predictor and joiner are set to 1024 to ensure the number of the total parameters is identical. All speech translation models were trained with Adam optimizer with an initial learning rate of 5×10^{-4} and invert_sqrt scheduler. Each model was trained with 2× V-100 GPU with 32GB video memory, using a batch-size of 20000 frames; update-frequency is set to be 8.

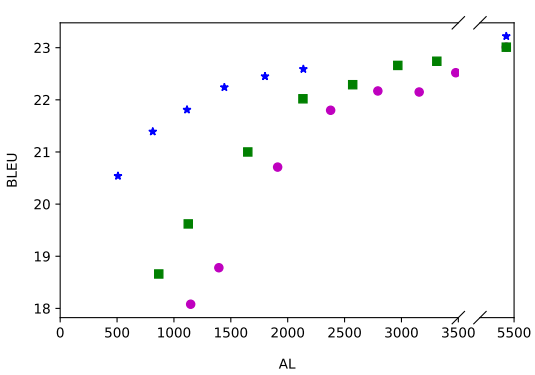
C.2 Hyper-parameters on Text-to-Text Simultaneous Translation

Our experiments on text-to-text simultaneous translation are based on Transformer_Base. That is, the hidden size, feed-forward hidden size, number of heads, number of encoder and decoder layers are set to 512, 2048, 8, 6, and 6. The dropout ratio is set to 0.3. The MMA-IL model is trained with architecture *transformer_monotonic* [9], except the noise variance is set to 2. Our CAAT model shares the same parameters number with the Transformer model by setting the feed-forward hidden size of predictor and joiner to 1024 (half of Transformer decoder feed-forward hidden size). All text translation models were trained with Adam optimizer with initial learning rate 5×10^{-4} and invert_sqrt scheduler. Each model was trained with 2× V-100 GPU with 32GB video memory, with batch-size 4096 frames, and update-frequency is set to 8.

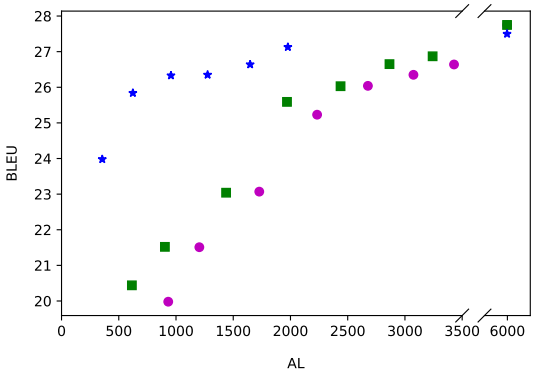
D Expanded Results

We also evaluate our work with the latency metrics Average Proportion (AP) and Differentiable Average Lagging (DAL). The full-size version of translation quality against latency (AL, AP, and DAL) curves on the MuST-C EN→DE and EN→ES speech-to-text simultaneous translation tasks are shown in Figure 8. And the quality-latency curves on the WMT15 DE→EN text-to-text translation task are shown in Figure 9. We also provide a complete table of results in Tables 5, 6 and 7 .

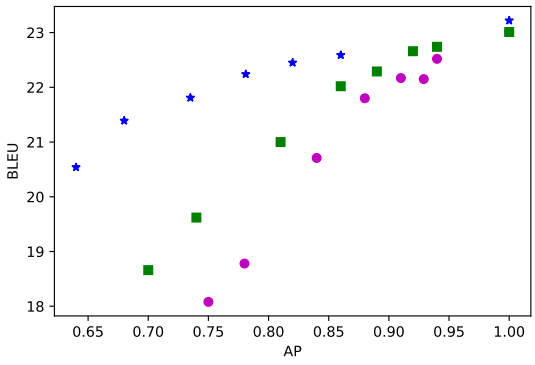
[9] https://github.com/pytorch/fairseq/tree/master/examples/simultaneous_translation/models/transformer_monotonic_attention.py



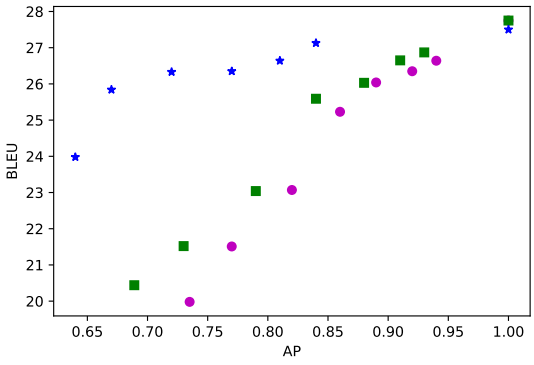
(a) BLEU-AL, EN→DE



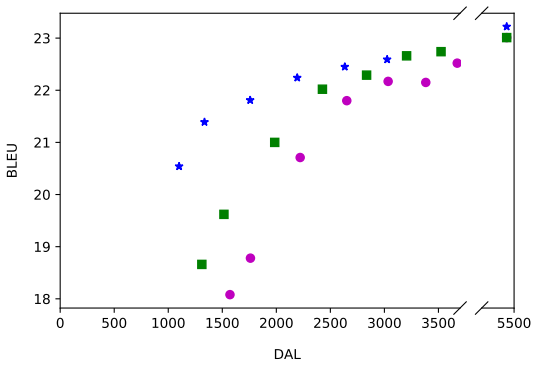
(b) BLEU-AL, EN→ES



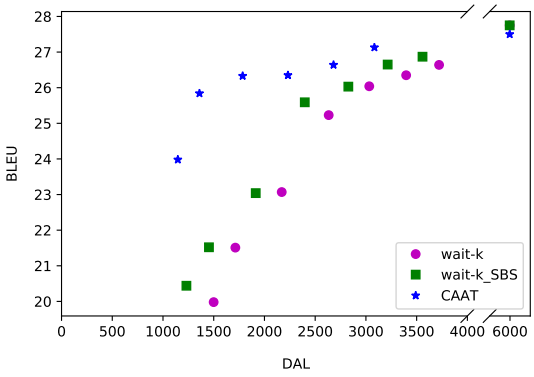
(c) BLEU-AP, EN→DE



(d) BLEU-AP, EN→ES

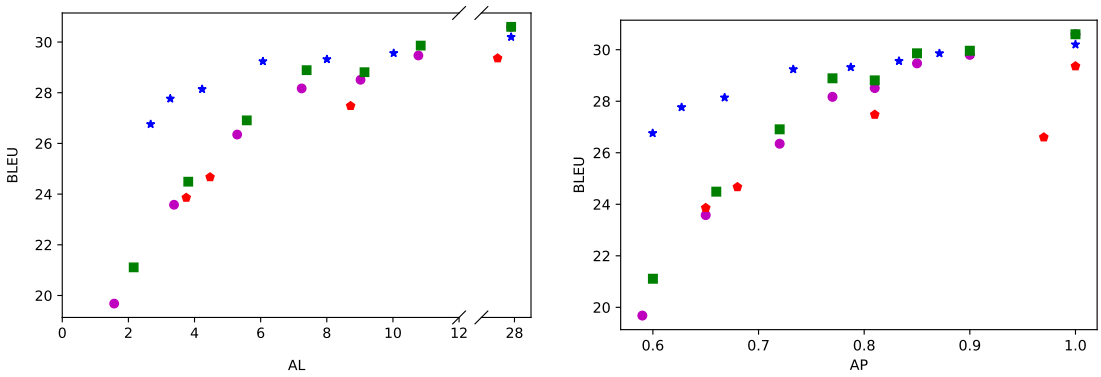


(e) BLEU-DAL, EN→DE



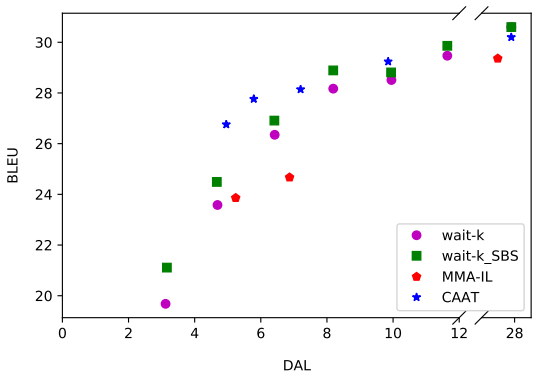
(f) BLEU-DAL, EN→ES

Figure 8: Translation quality against latency (AL, AP and DAL) on the MuST-C EN→DE and EN→ES speech translation test sets.



(a) BLEU-AL

(b) BLEU-AP



(c) BLEU-DAL

Figure 9: Translation quality against latency (AL,AP and DAL) on the WMT15 DE→EN text translation test set.

	BLEU	AL	AP	DAL
k		*wait-k*		
1	18.1	1147.0	0.75	1571.1
2	18.8	1394.9	0.78	1760.6
4	20.7	1911.6	0.84	2220.6
6	21.8	2377.4	0.88	2652.0
8	22.2	2792.5	0.91	3034.5
10	22.2	3155.7	0.929	3383.0
12	22.5	3477.3	0.94	3674.0
$+\infty$	23.0	5431.6	1.00	5431.6
k		*wait-k* with SBS		
1	18.7	866.6	0.70	1310.9
2	19.6	1125.8	0.74	1515.3
4	21.0	1649.2	0.81	1985.2
6	22.0	2134.0	0.86	2426.7
8	22.3	2571.4	0.89	2835.3
10	22.7	2967.9	0.92	3205.5
12	22.7	3310.8	0.94	3524.1
$+\infty$	23.0	5431.6	1.00	5431.6
d		CAAT		
8	20.5	508.1	0.64	1100.4
16	21.4	813.8	0.68	1335.3
32	21.8	1114.9	0.74	1758.1
48	22.2	1443.4	0.78	2193.6
64	22.4	1800.6	0.82	2633.5
80	22.6	2137.8	0.86	3025.4
$+\infty$	23.2	5431.6	1.00	5431.6

Table 5: Complete results on the MuST-C EN→DE speech translation test set.

	BLEU	AL	AP	DAL
k		*wait-k*		
1	20.0	932.1	0.74	1499.3
2	21.5	1203.9	0.77	1713.1
4	23.1	1727.5	0.82	2170.3
6	25.2	2232.7	0.86	2633.2
8	26.0	2676.5	0.89	3033.0
10	26.4	3074.3	0.92	3396.6
12	26.6	3428.9	0.94	3721.1
$+\infty$	27.8	5997.1	1.00	5997.1
k		*wait-k* with SBS		
1	20.4	614.6	0.69	1230.8
2	21.5	903.0	0.73	1453.0
4	23.0	1437.7	0.79	1914.1
6	25.6	1969.6	0.84	2397.4
8	26.0	2437.5	0.88	2826.7
10	26.6	2865.0	0.91	3214.4
12	26.9	3241.9	0.93	3557.9
$+\infty$	27.8	5997.1	1.00	5997.1
d		CAAT		
8	24.0	355.9	0.64	1146.3
16	25.8	623.2	0.67	1359.4
32	26.3	955.9	0.72	1785.0
48	26.4	1275.9	0.77	2231.4
64	26.6	1647.7	0.81	2680.7
80	27.1	1977.3	0.84	3083.7
$+\infty$	27.5	55997.1	1.00	5997.1

Table 6: Complete results on the MuST-C EN→ES speech translation test set.

	BLEU	AL	AP	DAL
k		*wait-k*		
2	19.7	1.57	0.59	3.12
4	23.6	3.38	0.65	4.69
6	26.4	5.29	0.72	6.42
8	28.2	7.24	0.77	8.19
10	28.5	9.02	0.81	9.95
12	29.5	10.77	0.85	11.64
16	29.8	13.95	0.90	14.71
$+\infty$	30.6	27.90	1.00	27.90
k		*wait-k* with SBS		
2	21.1	2.16	0.60	3.16
4	24.5	3.81	0.66	4.67
6	26.9	5.58	0.72	6.41
8	28.9	7.39	0.77	8.19
10	28.8	9.14	0.81	9.94
12	29.9	10.84	0.85	11.64
16	30.0	13.99	0.90	14.72
$+\infty$	30.6	27.90	1.00	27.90
λ_{avg}		MMA-IL		
0.8	21.1	3.26	0.63	4.65
0.6	23.9	3.74	0.65	5.24
0.4	24.7	4.47	0.68	6.87
0.2	27.5	8.72	0.81	13.0
0.1	26.6	21.37	0.97	24.48
0.	29.5	27.49	1.0	27.49
d		CAAT		
1	26.8	2.67	0.60	4.96
2	27.8	3.27	0.63	5.79
4	28.1	4.23	0.67	7.20
8	29.2	6.07	0.73	9.85
12	29.3	8.00	0.79	12.58
16	29.6	10.02	0.83	15.23
20	29.9	12.16	0.87	17.64
$+\infty$	30.2	27.90	1.00	27.90

Table 7: Complete results on the WMT15 DE→EN test translation test set.