Decoder-only Streaming Transformer for Simultaneous Translation

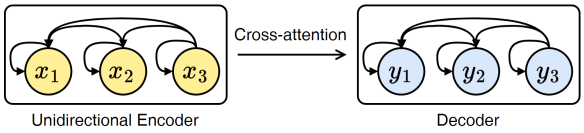
Shoutao Guo [1,3], **Shaolei Zhang** [1,3], **Yang Feng** [1,2,3]*
[1]Key Laboratory of Intelligent Information Processing,
Institute of Computing Technology, Chinese Academy of Sciences (ICT/CAS)
[2] Key Laboratory of AI Safety, Chinese Academy of Sciences
[3] University of Chinese Academy of Sciences, Beijing, China
guoshoutao22z@ict.ac.cn, zhangshaolei20z@ict.ac.cn, fengyang@ict.ac.cn

Abstract

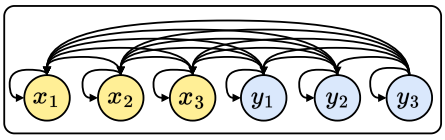
Simultaneous Machine Translation (SiMT) generates translation while reading source tokens, essentially producing the target prefix based on the source prefix. To achieve good performance, it leverages the relationship between source and target prefixes to exact a policy to guide the generation of translations. Although existing SiMT methods primarily focus on the Encoder-Decoder architecture, we explore the potential of Decoder-only architecture, owing to its superior performance in various tasks and its inherent compatibility with SiMT. However, directly applying the Decoder-only architecture to SiMT poses challenges in terms of training and inference. To alleviate the above problems, we propose the first Decoder-only SiMT model, named Decoder-only Streaming Transformer (DST). Specifically, DST separately encodes the positions of the source and target prefixes, ensuring that the position of the target prefix remains unaffected by the expansion of the source prefix. Furthermore, we propose a Streaming Self-Attention (SSA) mechanism tailored for the Decoder-only architecture. It is capable of obtaining translation policy by assessing the sufficiency of input source information and integrating with the soft-attention mechanism to generate translations. Experiments demonstrate that our approach achieves state-of-the-art performance on three translation tasks[1].

1 Introduction

Simultaneous Machine Translation (SiMT) (Gu et al., 2017; Ma et al., 2019) is designed for generating translations in real-time scenarios such as online conferences and real-time subtitles. It predicts the target tokens (i.e., target prefix) based on the already read source tokens (i.e., source prefix), aiming to achieve good tradeoffs between latency and translation quality. During training, SiMT models need to learn the correspondence between source and target prefixes, crucial for extracting policies that ensure superior performance during inference (Zhang and Feng, 2022c).



(a) Encoder-Decoder Architecture.



(b) Decoder-only Architecture.

Figure 1: Comparison of Encoder-Decoder architecture and Decoder-only architecture.

Existing research on SiMT primarily focuses on the Encoder-Decoder architecture and is categorized into fixed and adaptive policies. For fixed policy (Dalvi et al., 2018; Ma et al., 2019; Elbayad et al., 2020), the model utilizes heuristic rules to determine the source prefix used for generating translations, which ignores the correspondence between the source and target prefixes. This may lead to redundant or missing source information during translation, resulting in inferior performance (Zhang and Feng, 2022a). For adaptive policy (Ma et al., 2020b), the model dynamically decides whether to read or output tokens based on the relationship between the source and target prefixes. This dynamic adjustment of policy in response to the translation status allows for improved tradeoffs (Zhao et al., 2023). However, there is a lack of exploration in SiMT regarding the Decoder-only architecture.

With the rise of language models, the Decoder-only architecture has exhibited superior performance across diverse tasks (Touvron et al., 2023; Team et al., 2024). As illustrated in Figure 1,

* Corresponding author: Yang Feng.
[1]Code is at https://github.com/ictnlp/DST

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 8851–8864

the Decoder-only architecture, compared to the Encoder-Decoder architecture with an equivalent number of parameters, can support more layers, thereby offering parameter efficiency and better scalability (Liu et al., 2023). Importantly, SiMT relies on unidirectional encoding (Elbayad et al., 2020), and the Decoder-only architecture seamlessly accommodates this requirement. Therefore, we explore the capability of Decoder-only architecture in SiMT.

However, directly applying the Decoder-only architecture to the SiMT task poses challenges in both training and inference. During inference, with the arrival of each source token, there is a position increase of the generated target prefix, necessitating the model to re-encode the target prefix. This exacerbates the inference cost, particularly at low latency (Wang et al., 2024). During training, the model learns to predict the corresponding target prefix based on a given source prefix. Consequently, it is necessary to construct corresponding target prefixes for all possible source prefixes in a sentence pair. This limitation hinders the model from learning the translation policy and leads to an increase in training costs compared to the Encoder-Decoder architecture.

To overcome the above limitations, we propose the first SiMT model based on Decoder-only architecture, named the **D**ecoder-only **S**treaming **T**ransformer (DST). To alleviate the issue of re-encoding, DST encodes the positional information of the source prefix and the target prefix separately. This ensures that the expansion of the source prefix does not impact the position of the generated target prefix, thereby reducing the inference costs. To assess the contribution of partial source information to generating target tokens, DST uses the proposed Streaming Self-Attention (SSA) in replace of the conventional masked self-attention in the Decoder layer to decrease training costs and derive a translation policy.

During training, SSA can consider all possible source prefixes for the target prefixes in a sentence pair. Specifically, SSA predicts attention allocation for different source prefixes and combines it with the soft-attention mechanism to obtain expected attention for all source tokens and tokens in the target prefix. This expected attention is then utilized to derive the context vector. By leveraging SSA, the model learns the importance of all source prefixes in translating the target prefix, thereby reducing training costs. During inference, SSA accumulates the allocated attention from all prefixes of the input source tokens, enabling an assessment of the sufficiency of input source information for generating translation. The model utilizes this assessment to determine whether to read or generate tokens, thereby acquiring the translation policy. Experiments demonstrate that DST achieves state-of-the-art performance on three tasks.

2 Background

Simultaneous Machine Translation The SiMT model (Ma et al., 2019) dynamically reads the source sentence $\mathbf{x} = (x_1, ..., x_J)$ with length J and generates translation $\mathbf{y} = (y_1, ..., y_I)$ with length I based on a policy. To articulate the policy, we introduce g_i, representing the number of source tokens involved in translating y_i. Thus, the translation policy from $\mathbf{x}$ to $\mathbf{y}$ can be denoted as $\mathbf{g} = (g_1, ..., g_I)$. During training, the SiMT model undergoes optimization by minimizing the cross-entropy loss:

$$\mathcal{L}_{simt} = -\sum_{i=1}^{I} \log p(y_i^\star \mid \mathbf{x}_{\leq g_i}, \mathbf{y}_{<i}), \quad (1)$$

where $y_i^\star$ represents the ground-truth token.

Masked Self-Attention Masked self-attention allows each position in the decoder to attend to all positions in the decoder up to and including that position (Vaswani et al., 2017), ensuring the auto-regressive generation. Given the target hidden states $\mathbf{s} = (s_1, ..., s_I)$, the context vector is computed as follows:

$$e_{i,k} = s_i W^Q (s_k W^K)^\intercal / \sqrt{d}, \quad (2)$$

$$\alpha_{i,k} = \begin{cases} \exp(e_{i,k}) / \sum_{l=1}^{i} \exp(e_{i,l}) & \text{if } k \leq i \\ 0 & \text{otherwise} \end{cases}, \quad (3)$$

$$c_i = \sum_{k=1}^{i} \alpha_{i,k} (s_k W^V), \quad (4)$$

where W^Q, W^K and W^V are projection parameters, and d denotes the dimension of inputs.

3 Method

In this section, we introduce the Decoder-only Streaming Transformer (DST). DST adopts the Decoder-only architecture and employs the proposed Streaming Self-Attention (SSA) in place of masked self-attention at each layer. During inference, SSA accumulates the attention assigned to

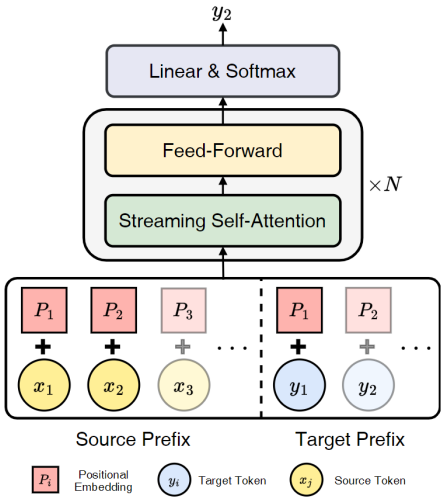


Figure 2: The architecture of DST. It shows the moment when DST generates y_2 after reading two source tokens.

the input source tokens to assess the sufficiency of source information and obtain the translation policy accordingly. During training, DST leverages our proposed constraints to ensure the learning of the SSA mechanism. The details of DST are introduced as follows.

3.1 Model Architecture

We present the model architecture of DST in Figure 2. DST takes the concatenation of the source and target prefixes as input, encoding the positional information of both the source and target prefixes separately. In this way, the expansion of the source prefix does not affect the position of the generated target prefix, preventing the re-encoding of the generated target prefix. In each layer, DST replaces the masked self-attention module (Vaswani et al., 2017) in the conventional Decoder-only architecture with our proposed Streaming Self-Attention (SSA) module. During inference, the SSA module will determine the policy and integrate with the soft-attention mechanism to derive the context vector and predict the next token.

3.2 Streaming Self-Attention

In order to address the SiMT task, DST incorporates Streaming Self-Attention (SSA) in place of masked self-attention. By concatenating the source sentence and the target sentence as input, SSA adopts masked self-attention within the source sentence (Vaswani et al., 2017). For each target token, SSA initially allocates its attention probability to all possible source prefixes and subsequently combines the allocated attention with the soft-attention mechanism to compute the expected attention probability for source tokens and preceding target tokens. The context vectors of target tokens are computed by utilizing the expected attention. We then provide a detailed explanation of the SSA mechanism as applied to target tokens during training.

Given the source sentence $\mathbf{x}$ and the target sentence $\mathbf{y}$, the hidden state sequence input to SSA is denoted as $\mathbf{h} = (h_1^s, ..., h_J^s, h_1^t, ..., h_I^t)$, where h_j^s denotes the source hidden state and h_i^t represents the target hidden state. To articulate the allocated attention to the source prefixes, we define $p_{i,j}$, which signifies the allocated attention of the target token y_i to the source prefix $\mathbf{x}_{\leq j}$. The calculation of $p_{i,j}$ is computed by utilizing the relationship between the source and target prefixes:

$$p_{i,j} = \text{sigmoid}(\frac{\overline{h}_i^t U^Q (\overline{h}_j^s U^K)}{\sqrt{d}}), \tag{5}$$

where U^Q and U^K are learnable parameters, and d denotes the dimension of hidden state. $\overline{h}_i^t$ and $\overline{h}_j^s$ represents the mean pooling of hidden states corresponding to the target prefix $\mathbf{y}_{\leq i}$ and source prefix $\mathbf{x}_{\leq j}$, respectively. Subsequently, we utilize $p_{i,j}$ to obtain the expected attention of each target token towards both source tokens and preceding target tokens.

Taking into account all possible source prefixes when translating the target token y_i, the expected attention of target token y_i to the source token x_j is calculated as:

$$\alpha_{i,j}^s = \sum_{m=j}^{J} \frac{p_{i,m} \exp(e_{i,j}^s)}{\sum_{l=1}^{m} \exp(e_{i,l}^s)}, \tag{6}$$

where $e_{i,j}^s$ denotes the scaled dot-product of h_i^t and h_j^s, as illustrated in Eq.(2). The first summation considers all possible source prefixes that include x_j, and the term inside summation reflects the attention probability associated with $p_{i,m}$ and soft-attention. Correspondingly, the expected attention of y_i to the target token $y_k (k \in [1, i])$ is:

$$\alpha_{i,k}^t = (1 - \sum_{j=1}^{J} p_{i,j}) \frac{\exp(e_{i,k}^t)}{\sum_{l=1}^{i} \exp(e_{i,l}^t)}, \tag{7}$$

where $\sum_{j=1}^{J} p_{i,j}$ represents the total attention to the source tokens and $e_{i,k}^t$ is the scaled dot-product of h_i^t and h_k^t. Therefore, we can get the context vector of y_i by considering both source tokens and

previous target tokens:

$$c_i^t = \sum_{j=1}^{J} \alpha_{i,j}^s (h_j^s W^V) + \sum_{k=1}^{i} \alpha_{i,k}^t (h_k^t W^V), \quad (8)$$

where W^V denotes the projection parameters.

The above is the operational mechanism of SSA during training. During inference, SSA utilizes the input source tokens and the previously generated target tokens to generate the context vector.

3.3 Inference

After introducing SSA, DST employs it to derive a policy, which instructs the model to generate translation. The SiMT policy generally assesses the adequacy of the source information to determine whether to proceed with generating the translation or to read additional source tokens (Zhang and Feng, 2023b). In DST, SSA initially allocates attention to different source prefixes using $p_{i,j}$, subsequently integrating the allocated attention with the soft-attention. Consequently, $\sum_{j=1}^{m} p_{i,j}$ signifies the accumulated attention to the current available source prefix $\mathbf{x}_{\leq m}$ and quantifies the sufficiency of source information. By setting an appropriate threshold δ_{infer}, DST deems the source information sufficient under the condition:

$$\sum_{j=1}^{m} p_{i,j} > \delta_{infer}, \quad (9)$$

where m is the number of input source tokens. At this point, DST is capable of generating the translation based on the existing source tokens. Otherwise, the model should continue reading the source tokens until the aforementioned condition is met or the entire sentence is input to our model. It is noteworthy that DST will generate the translation only when most layers meet the conditions.

3.4 Training Method

To facilitate the learning of SSA mechanism, we introduce the training method. The essence lies in the learning of $p_{i,j}$. Although we may not directly provide an optimization objective for $p_{i,j}$, we can leverage the characteristics of SiMT tasks (Zhang et al., 2022) and the properties of $p_{i,j}$ to induce its learning. Therefore, in addition to the cross-entropy loss $\mathcal{L}_{simt}$, we propose three additional constraints and a curriculum learning strategy to aid in the training of DST.

Summation Constraint Based on the inherent property of $p_{i,j}$, we introduce the summation constraint. As described in the section 3.2, $p_{i,j}$ represents the attention probability allocated to the source prefix $x_{\leq j}$ and $\sum_{j=1}^{J} p_{i,j}$ reflects the total attention on all source tokens. Therefore, it should be equivalent to the sum of attention to source tokens in masked self-attention (Vaswani et al., 2017):

$$\mathcal{L}_{sum} = \sum_{i=1}^{I} \| \sum_{j=1}^{J} p_{i,j} - \beta_i \|_2, \quad (10)$$

where β_i denotes the attention to the source tokens in masked soft-attention. It is computed as:

$$\beta_i = \frac{\sum_{j=1}^{J} \exp(e_{i,j}^s)}{\sum_{j=1}^{J} \exp(e_{i,j}^s) + \sum_{k=1}^{i} \exp(e_{i,k}^t)}, \quad (11)$$

where $e_{i,j}^s$ is the scaled dot-product of target token y_i and source token x_j, and $e_{i,k}^t$ represents the scaled dot-product between the target token y_i and its preceding target token y_k.

Latency Constraint In addition to the summation constraint, we introduce a latency constraint for DST. Without the latency constraint, SSA tends to allocate excessive attention to longer source prefixes during training (Ma et al., 2020b). This tendency encourages the model to read more source tokens during inference, resulting in high latency. According to Zhang and Feng (2022b), the alignment favoring low latency tends to concentrate near the diagonal between source and target sentences. Expressing the attention allocation for source prefixes in sentence pair $(\mathbf{x}, \mathbf{y})$ as $\mathbf{p} = (p_{i,j})_{I \times J}$, the latency constraint aims to encourage SSA to allocate more attention to source prefixes closer to the diagonal. To make it, we introduce the cost matrix $\mathbf{C} = (C_{i,j})_{I \times J}$, where $C_{i,j}$ is computed as:

$$C_{i,j} = \frac{1}{\max(I, J)} \max(|j - i \times \frac{J}{I}| - \epsilon, 0), \quad (12)$$

where $|j - i \times \frac{J}{I}|$ quantifies the degree of deviation from the diagonal between the target token y_i and the source prefix $\mathbf{x}_{\leq j}$. Therefore, $C_{i,j}$ supports attention distributions that are close to the diagonal. The hyperparameter ϵ controls the tolerance level for the deviation. We then obtain the latency constraint as:

$$\mathcal{L}_{lat} = \sum_{i=1}^{I} \sum_{j=1}^{J} p_{i,j} C_{i,j}. \quad (13)$$

Further explanation about latency constraint is provided in the Appendix A.

Consistency Constraint During training, DST utilizes summation and latency constraints to ensure the learning of $p_{i,j}$ in each layer. However, there are variations among different layers (Yang et al., 2020), and our policy is obtained by integrating decisions from different layers. Without constraints on the decisions between different layers, the presence of outlier layers may result in excessively high latency for the learned policy. Therefore, we propose a consistency constraint to ensure consistency across decisions at different layers. Denoting the $p_{i,j}$ in the n-th layer as $p_{i,j}^{(n)}$, and the consistency constraint is calculated as follows:

$$\mathcal{L}_{con} = \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{n=1}^{N}\frac{1}{N}\|p_{i,j}^{(n)} - \overline{p}_{i,j}\|_2, \quad (14)$$

where N is the number of layers and $\overline{p}_{i,j}$ is computed as follows:

$$\overline{p}_{i,j} = \frac{1}{N}\sum_{n=1}^{N} p_{i,j}^{(n)}. \quad (15)$$

Therefore, we can get the overall training objective of DST:

$$\mathcal{L} = \mathcal{L}_{simt} + \mathcal{L}_{sum} + \mathcal{L}_{lat} + \mathcal{L}_{con}. \quad (16)$$

Curriculum Learning Strategy If we train DST based on the training objective $\mathcal{L}$ directly, the model learns to generate translation using the entire source sentence. However, DST generates translations based on source prefixes during inference, leading to a discrepancy between training and inference (Zhang and Feng, 2022d). To mitigate this problem, we train DST to generate translations based on the source prefix by masking out subsequent source tokens during training. Specifically, by setting the threshold δ_{train}, DST masks out the source tokens after the shortest prefix $\mathbf{x}_{\leq m}$ that satisfies the condition $\sum_{j=1}^{m} p_{i,j} > \delta_{train}$. Additionally, we introduce curriculum learning (Bengio et al., 2009), which gradually transitions the available source information from the entire sentence to the prefix consistent with the inference. During training, we implement this strategy by adjusting δ_{train} as:

$$\delta_{train} = \delta_{infer} + (1 - \delta_{infer}) \times \exp\left(-\frac{\text{update}}{T}\right), \quad (17)$$

where T is a hyperparameter and update represents the number of updates. During this process, DST gradually adapts to the scenario of generating translations based on prefixes (Guo et al., 2024).

4 Experiments

4.1 Datasets

We evaluate our method on three translation tasks.

IWSLT15[2] **English→Vietnamese (En→Vi)** (Cettolo et al., 2015) Consistent with Ma et al. (2020b), we replace the tokens occurring less than 5 with $\langle unk \rangle$. We use TED tst2012 and TED tst2013 as the development set and the test set, respectively.

WMT16[3] **English→Romanian (En→Ro)** We use newsdev-2016 as validation set and newstest-2016 as test set. The source and target languages employ a shared vocabulary. Other settings are consistent with Gu et al. (2018).

WMT15[4] **German→English (De→En)** We use newstest2013 as validation set and newstest2015 as test set. Following Ma et al. (2020b), we apply BPE (Sennrich et al., 2016) with 32K subword units and use a shared vocabulary between source and target.

4.2 System Settings

Our experiments involve the following methods. Apart from our approach, other methods all employ Encoder-Decoder architecture.

Full-sentence represents the full-sentence translation of Encoder-Decoder architecture.

Wait-k (Ma et al., 2019) reads k tokens and then alternates between writing and reading a token.

MoE Wait-k (Zhang and Feng, 2021) treats each head as a Wait-k expert and integrates the outputs of all experts to generate translation.

MMA (Ma et al., 2020b) makes each head determine the policy by predicting a Bernoulli variable and generates translation by integrating the results of multiple heads.

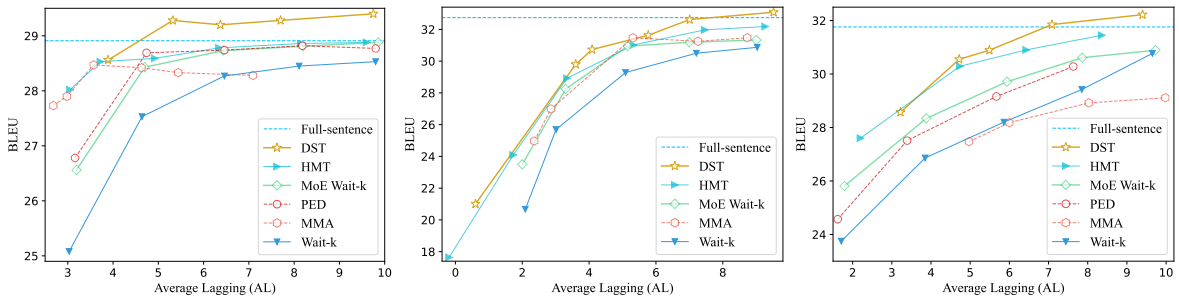
PED (Guo et al., 2022) implements the adaptive policy by integrating post-evaluation into the translation policy.

HMT (Zhang and Feng, 2023a) utilizes the Hidden Markov Model to model the policy as hidden events and translation as observed events, and achieves the current state-of-the-art performance.

DST denotes our method described in Section 3.

All systems are adapted from Fairseq Library (Ott et al., 2019). Regarding the compared methods, we apply Transformer-Small (6 layers, 4 heads) for En→Vi task and Transform-Base (6 layers, 8 heads) for En→Ro and De→En tasks. For our approach, we set the number N of layers in DST to 16

[2] https://nlp.stanford.edu/projects/nmt/
[3] www.statmt.org/wmt16/
[4] www.statmt.org/wmt15/



(a) En→Vi (b) En→Ro (c) De→En

Figure 3: Comparison of our approach with other SiMT methods on En→Vi, En→Ro and De→En tasks.

Method	δ_{infer}	AL	BLEU
DST	0.3	**4.72**	**30.55**
w/o $\mathcal{L}_{lat}$	0.3	7.95	31.78
w/o $\mathcal{L}_{con}$	0.3	26.5	24.05
w/o CL Strategy	0.8	22.48	29.69

Table 1: Ablation study on the training method of DST. 'w/o $\mathcal{L}_{lat}$' removes the latency constraint. 'w/o $\mathcal{L}_{con}$' removes the consistency constraint. 'w/o CL Strategy' removes curriculum learning strategy. The experiments are conducted on the De→En task.

and utilize the top 8 layers to determine the translation policy. For the latency constraint, we empirically choose ϵ as 1. Other detailed system settings are shown in Appendix B. We obtain the SiMT models under different latency by adjusting δ_{infer}. To accelerate the convergence and achieve better performance, our approach is fine-tuned from conventional Decoder-only architecture with masked self-attention mechanism (Vaswani et al., 2017). We use greedy search during inference and evaluate all methods with latency measured by Average Lagging (AL) (Ma et al., 2019) and quality estimated by BLEU (Papineni et al., 2002).

4.3 Main Results

The performance comparison of our approach with other SiMT methods is illustrated in Figure 3, demonstrating that our approach achieves state-of-the-art performance on three tasks.

Compared to the Wait-k and MoE Wait-k methods, our method brings significant improvement. Both Wait-k and MoE Wait-k employ fixed policy and rely on heuristic rules to determine the source prefixes for translation (Ma et al., 2019; Zhang and Feng, 2021). This usually leads to the omission or redundancy of source information during translation (Zhang and Feng, 2022c), resulting in degraded performance. In contrast, our method adaptively determines the translation policy by leveraging the correspondence between the source and target prefixes. This adaptability allows our method to achieve superior tradeoffs. Additionally, our method outperforms PED, MMA, and HMT, which belong to the adaptive policy. Previous adaptive methods are based on the Encoder-Decoder architecture, determining the policy based on extracted features from the source and target prefixes (Ma et al., 2023; Zhang and Feng, 2023a). Our method relies on the Decoder-only architecture and utilizes the accumulated attention to existing source tokens to assess the sufficiency of translation, resulting in more effective policies. Besides, our method surpasses the performance of Full-sentence translation based on the Encoder-Decoder architecture with lower latency. This can be attributed to the advantages of the decoder-only architecture.

5 Analysis

To deepen the understanding of our method, we conduct multiple analyses. The experiments are all performed on the De→En translations task.

5.1 Ablation Study

In order to understand the settings, we conduct ablation studies on the training methods, curriculum learning strategy, and the number of layers.

Table 1 presents ablation experiments on the training method, where each setting contributes to the performance of DST. The latency constraint aids the model in acquiring the low-latency policy (Zhang and Feng, 2022b), and the curriculum learning strategy enables the model to adapt to generating translations based on prefixes (Guo et al., 2024). The consistency constraint effectively addresses the

T	δ_{infer}	AL	BLEU
20000	0.3	2.81	25.68
30000	0.3	**4.72**	**30.55**
40000	0.3	4.05	29.26

Table 2: The performance when DST employs different hyperparameters of curriculum learning strategy. The experiments are performed on the De→En dataset.

N	δ_{infer}	AL	BLEU
12	0.3	4.23	28.99
14	0.3	4.49	29.14
16	0.3	**4.72**	**30.55**

Table 3: The performance of our method with different number of layers. The results are based on De→En task.

problems of outlier layers, which ensures consistency across decisions made at different layers (Ma et al., 2020b).

We further explore the impact of hyperparameters in the curriculum learning strategy in Table 2. A smaller T indicates a quicker transition of the training environment to the scenario of SiMT. We find that transitioning to the scenario of SiMT at an appropriate pace can achieve better tradeoffs between latency and translation quality.

Additionally, we investigate the relationship between the number of layers in DST and its performance in Table 3. This suggests that the performance of DST shows incremental improvement as the number of layers increases, highlighting a certain level of scalability.

5.2 Hallucination in Translation

To assess the quality of translations generated by different SiMT methods, we evaluate the hallucination in translations. In SiMT, when the model is trained to predict target tokens based on missing essential source information, it is prone to producing hallucinations during inference (Guo et al., 2023a). Therefore, the proportion of hallucinations contained in the generated translations also serves as a reflection of the quality of the learned policy during training.

To quantify the hallucinations, we introduce the hallucination rate (HR) (Chen et al., 2021a), which measures the percentage of tokens generated that are not related to the source sentence. We then provide its detailed definition. Given the source sentence $\mathbf{x}$, we define the corresponding translation

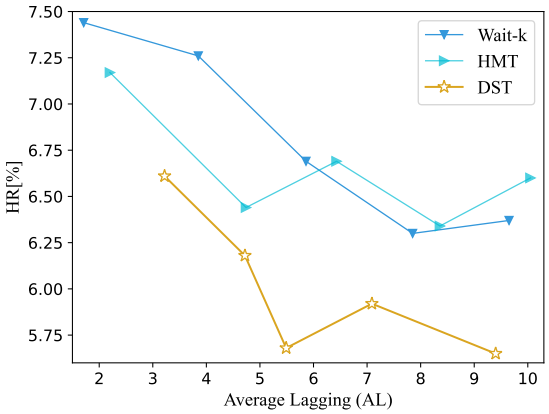


Figure 4: The comparison of hallucination in translations generated by different SiMT models. The results are based on the De→En dataset.

as $\hat{\mathbf{y}}$. Subsequently, we can get the alignment set $\mathbf{h}$, which is a set of source-target index pairs (j, i) where j^{th} source token x_j aligns to the i^{th} target token $\hat{y}_i$.

The token $\hat{y}_i$ in $\hat{\mathbf{y}}$ is seen as hallucination ($H(i, \mathbf{h})$=1) if it can not be aligned to any source token:

$$H(i, \mathbf{h}) = \mathbb{1}[\{(j, i) \in \mathbf{h}\} = \varnothing]. \quad (18)$$

Therefore, the hallucination rate can be defined as:

$$HR(\mathbf{x}, \hat{\mathbf{y}}, \mathbf{h}) = \frac{1}{|\hat{\mathbf{y}}|} \sum_{i=1}^{|\hat{\mathbf{y}}|} H(i, \mathbf{h}). \quad (19)$$

As illustrated in Figure 4, our model demonstrates a lower likelihood of generating hallucinations at all latency, indicating increased reliability in generated translations. Compared to the Wait-k approach, our method allows for flexible utilization of essential source information by adjusting its policy and obtains a lower hallucination rate. Moreover, DST considers a more extensive range of translation policies than HMT during training (Zhang and Feng, 2023a). This provides possibilities for DST to acquire more effective policies, thereby achieving better tradeoffs between latency and translation quality.

5.3 Model Efficiency

In addition to evaluating the hallucinations in translation, we also investigate the model efficiency of different SiMT methods. As shown in Table 4, our method obtains superior performance with relatively high efficiency.

Method	#Params (↓)	Inference Speed-up (↑)	Training Speed-up (↑)	AL	BLEU
Wait-k	80M	**1.907**	**3.982**	3.85	26.86
HMT	80M	1.000	1.000	4.74	30.29
DST	**67M**	1.391	2.029	**4.72**	**30.55**

Table 4: The comparison of model efficiency for different SiMT models. '#Params' represents the number of trained parameters. 'Inference Speed-up' indicates the speed-up ratio of inference compared to the HMT model. 'Training Speed-up' represents the speed-up ratio of training in comparison to the HMT model. The bolded data indicates the SiMT models that perform the best in the evaluation metrics. The experiments are conducted on the De→En task.

Method	CAAL	SacreBLEU
Wait-k	1235.72	25.83
	1704.95	27.30
DST	1672.48	27.53
	1702.00	28.17

Table 5: Comparison of different methods on MuST-C English→German task. This latency metric CAAL is measured in millisecond.

Method	δ_{infer}	**AL**	**BLEU**
Expected-Allocation	0.3	**4.72**	**30.55**
Max-Allocation	0.3	13.73	13.79

Table 6: The performance of the SiMT model when using different strategies to allocate attention to the source prefixes in the Streaming Self-Attention (SSA) mechanism. 'Expected-Allocation' represents that SSA allocates attention to all possible source prefixes. 'Max-Allocation' signifies the allocation of source attention probability to the most probable source prefix.

Compared to the SiMT models based on the Encoder-Decoder architecture, our model, utilizing a Decoder-only architecture, achieves superior performance with a relatively smaller number of parameters. Combining the results from Table 3, it reflects the parameter efficiency and scalability of the Decoder-only architecture (Fu et al., 2023).

Besides, we evaluate the training and inference efficiency of different SiMT models. All related experiments are conducted on NVIDIA GeForce RTX 3090. In terms of training speed, our approach is lower than Wait-k policy but higher than the adaptive policy HMT. Compared to Wait-k policy, our method involves more computations about attention, therefore requiring more training costs. Furthermore, HMT significantly increases the training complexity by expanding the target sentences several times during training. During inference, while our method is slightly slower than Wait-k, the inference speed of our method can still reach 78 tokens per second, which can fully meet the application requirements.

In addition to comparing the training and inference efficiency of SiMT methods in Table 4, we also consider model inference time and delay caused by waiting for source information concurrently. To evaluate the performance of the SiMT models in scenarios that closely resemble real-world conditions, we compute Computation-Aware Average Lagging (CAAL) (Ma et al., 2020a) for different methods utilizing MuST-C English→German dataset (Di Gangi et al., 2019). Unlike Ma et al. (2020a), our approach does not take direct speech input but rather uses the text corresponding to the speech. Therefore, we use `whisper-large-v3`[5] to align the text with the speech, considering both the length of the corresponding speech sequence and the actual machine inference time when calculating CAAL. Our settings use the Transformer-Small architecture and evaluate the performance of SiMT methods using CAAL and SacreBLEU (Post, 2018). The results are shown in Table 5. Although the inference speed of our method is slightly slower than the Wait-k policy in Table 4, it still achieves better performance in more realistic scenarios.

Therefore, our Decoder-only method can achieve state-of-the-art performance with higher parameter utilization, and relatively faster inference and training speed.

5.4 Analysis of Streaming Self-Attention

After evaluating the overall performance of our method, we delve into assessing the effectiveness of the proposed Streaming Self-Attention (SSA) mechanism. SSA initially allocates attention probability to all possible source prefixes and sub-

[5] https://huggingface.co/openai/whisper-large-v3

sequently combines the allocated attention with the soft-attention to determine the expected attention for source tokens, referred to as Expected-Allocation. To demonstrate the effectiveness of SSA, we also explore a Max-Allocation strategy, where the attention probability is allocated solely to the most probable source prefix.

As illustrated in Table 6, the Expected-Allocation method yields better latency-quality tradeoffs. The Expected-Allocation method integrates the translation policy with translation capability. It aids the model in learning the correlation between source and target prefixes, resulting in enhanced translation policies and overall performance. This underscores the importance of attention probability allocation in SSA.

6 Related Work

Simultaneous machine translation (SiMT) begins generating translation before reading the entire source sentence. To ensure the quality of generated translations and meet the latency requirements, it is necessary to determine the suitable source prefixes for translation. The process of determining the source prefixes for target tokens is referred to as the policy (Zhang et al., 2024). Depending on whether the model decides the policy utilizing the correspondence between source and target prefixes, SiMT methods can be broadly categorized into fixed policy and adaptive policy.

For fixed policy, the SiMT model reads or generates tokens according to the heuristic rules. Ma et al. (2019) proposes Wait-k policy, which reads k source tokens first, and then writes and reads one token alternately. Elbayad et al. (2020) introduces the unidirectional encoder, where the preceding source tokens cannot attend to subsequent words. Zhang and Feng (2021) proposes MoE Wait-k, which assigns a Wait-k policy to each head and combines the results from multiple heads to generate translations. There are some methods that enhance the SiMT capability by modifying the reference (Chen et al., 2021b; Guo et al., 2023b). However, the fixed policy requires the SiMT model to generate tokens even when source information is insufficient, leading to a decline in performance of SiMT.

Unlike fixed policy, adaptive policy can dynamically decide whether to read or generate tokens based on the correspondence between source prefix and target prefix. Zheng et al. (2020) implements the adaptive policy through a composition of several fixed policies. Miao et al. (2021) proposes a generative framework, which uses a reparameterized Poisson prior to regularise the policy. Zhang and Feng (2022b) models the contribution of each source token to the target tokens through optimal transport and determines the policy by accumulating the contributions of input source tokens. Zhang and Feng (2023a) utilizes the Hidden Markov Model to model the translation policy as the hidden events and the translation as the observed events. Ma et al. (2023, 2024) first proposes to conduct SiMT with non-autoregressive architecture and achieves good results. However, previous SiMT methods all employ the Encoder-Decoder architecture.

With the rise of language models, the Decoder-only architecture has demonstrated superior performance across various tasks (Touvron et al., 2023; Team et al., 2024; Jiang et al., 2024). The Decoder-only architecture has the advantages of good scalability and parameter efficiency. More importantly, its unidirectional encoding nature aligns greatly with streaming input. Therefore, we explore the application of the Decoder-only architecture in SiMT and propose the first Decoder-only SiMT model.

7 Conclusion

In this paper, we propose the first Decoder-only SiMT model, which leverages the characteristics of the Decoder-only architecture to implement the adaptive policy. Experiments show that our method achieves state-of-the-art performance and exhibits excellent scalability and model efficiency.

Limitations

Regarding the system settings, we investigate the impact of the number of layers and training methods on the performance of our DST method due to resource constraints. We think that further exploration of system settings could potentially yield even better results. We leave the exploration for future work.

Acknowledgements

This paper is supported by National Natural Science Foundation of China (Grant No. 62376260). We thank all the anonymous reviewers for their valuable feedback and thorough reviews.

References

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, Roldano Cattoni, and Marcello Federico. 2015. The IWSLT 2015 evaluation campaign. In *Proceedings of the 12th International Workshop on Spoken Language Translation: Evaluation Campaign@IWSLT 2015, Da Nang, Vietnam, December 3-4, 2015*.

Junkun Chen, Renjie Zheng, Atsuhito Kita, Mingbo Ma, and Liang Huang. 2021a. Improving simultaneous translation by incorporating pseudo-references with fewer reorderings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 5857–5864. Association for Computational Linguistics.

Junkun Chen, Renjie Zheng, Atsuhito Kita, Mingbo Ma, and Liang Huang. 2021b. Improving simultaneous translation by incorporating pseudo-references with fewer reorderings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5857–5864, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, and Stephan Vogel. 2018. Incremental decoding and training methods for simultaneous translation in neural machine translation. 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 493–499, New Orleans, Louisiana. Association for Computational Linguistics.

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.

Maha Elbayad, Laurent Besacier, and Jakob Verbeek. 2020. Efficient wait-k models for simultaneous machine translation. In *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020*, pages 1461–1465. ISCA.

Zihao Fu, Wai Lam, Qian Yu, Anthony Man-Cho So, Shengding Hu, Zhiyuan Liu, and Nigel Collier. 2023. Decoder-only or encoder-decoder? interpreting language model as a regularized encoder-decoder.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O. K. Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

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.

Shoutao Guo, Shaolei Zhang, and Yang Feng. 2022. Turning fixed to adaptive: Integrating post-evaluation into simultaneous machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 2264–2278. Association for Computational Linguistics.

Shoutao Guo, Shaolei Zhang, and Yang Feng. 2023a. Learning optimal policy for simultaneous machine translation via binary search. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2318–2333, Toronto, Canada. Association for Computational Linguistics.

Shoutao Guo, Shaolei Zhang, and Yang Feng. 2023b. Simultaneous machine translation with tailored reference. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3070–3084, Singapore. Association for Computational Linguistics.

Shoutao Guo, Shaolei Zhang, and Yang Feng. 2024. Glancing future for simultaneous machine translation. In *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 11386–11390.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of experts.

Peiyu Liu, Ze-Feng Gao, Yushuo Chen, Xin Zhao, and Ji-Rong Wen. 2023. Enhancing scalability of pre-trained language models via efficient parameter sharing. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 13771–13785, Singapore. 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 Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 3025–3036. Association for Computational Linguistics.

Xutai Ma, Juan Pino, and Philipp Koehn. 2020a. 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. 2020b. Monotonic multihead attention. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Zhengrui Ma, Qingkai Fang, Shaolei Zhang, Shoutao Guo, Yang Feng, and Min Zhang. 2024. A non-autoregressive generation framework for end-to-end simultaneous speech-to-any translation. In *Proceedings of the 62th Annual Meeting of the Association for Computational Linguistics (Long Papers)*, Bangkok, Thailand. Association for Computational Linguistics.

Zhengrui Ma, Shaolei Zhang, Shoutao Guo, Chenze Shao, Min Zhang, and Yang Feng. 2023. Non-autoregressive streaming transformer for simultaneous translation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5177–5190, Singapore. Association for Computational Linguistics.

Yishu Miao, Phil Blunsom, and Lucia Specia. 2021. A generative framework for simultaneous machine translation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 6697–6706. Association for Computational Linguistics.

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: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Demonstrations*, pages 48–53. 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.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, Pier Giuseppe Sessa, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, Amélie Héliou, Andrea Tacchetti, Anna Bulanova, Antonia Paterson, Beth Tsai, Bobak Shahriari, Charline Le Lan, Christopher A. Choquette-Choo, Clément Crepy, Daniel Cer, Daphne Ippolito, David Reid, Elena Buchatskaya, Eric Ni, Eric Noland, Geng Yan, George Tucker, George-Christian Muraru, Grigory Rozhdestvenskiy, Henryk Michalewski, Ian Tenney, Ivan Grishchenko, Jacob Austin, James Keeling, Jane Labanowski, Jean-Baptiste Lespiau, Jeff Stanway, Jenny Brennan, Jeremy Chen, Johan Ferret, Justin Chiu, Justin Mao-Jones, Katherine Lee, Kathy Yu, Katie Millican, Lars Lowe Sjoesund, Lisa Lee, Lucas Dixon, Machel Reid, Maciej Mikuła, Mateo Wirth, Michael Sharman, Nikolai Chinaev, Nithum Thain, Olivier Bachem, Oscar Chang, Oscar Wahltinez, Paige Bailey, Paul Michel, Petko Yotov, Rahma Chaabouni, Ramona Comanescu, Reena Jana, Rohan Anil, Ross McIlroy, Ruibo Liu, Ryan Mullins, Samuel L Smith, Sebastian Borgeaud, Sertan Girgin, Sholto Douglas, Shree Pandya, Siamak Shakeri, Soham De, Ted Klimenko, Tom Hennigan, Vlad Feinberg, Wojciech Stokowiec, Yu hui Chen, Zafarali Ahmed, Zhitao Gong, Tris Warkentin, Ludovic Peran, Minh Giang, Clément Farabet, Oriol Vinyals, Jeff Dean, Koray Kavukcuoglu, Demis Hassabis, Zoubin Ghahramani, Douglas Eck, Joelle Barral, Fernando Pereira, Eli Collins, Armand Joulin, Noah Fiedel, Evan Senter, Alek Andreev, and Kathleen Kenealy. 2024. Gemma: Open models based on gemini research and technology.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

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.

Minghan Wang, Thuy-Trang Vu, Ehsan Shareghi, and Gholamreza Haffari. 2024. Conversational simulmt: Efficient simultaneous translation with large language models.

Yilin Yang, Longyue Wang, Shuming Shi, Prasad Tadepalli, Stefan Lee, and Zhaopeng Tu. 2020. On the sub-layer functionalities of transformer decoder. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4799–4811, Online. Association for Computational Linguistics.

Shaolei Zhang, QingKai Fang, Shoutao Guo, Zhengrui Ma, Min Zhang, and Yang Feng. 2024. Streamspeech: Simultaneous speech-to-speech translation with multi-task learning. In *Proceedings of the 62th Annual Meeting of the Association for Computational Linguistics (Long Papers)*, Bangkok, Thailand. Association for Computational Linguistics.

Shaolei Zhang and Yang Feng. 2021. Universal simultaneous machine translation with mixture-of-experts wait-k policy. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 7306–7317. Association for Computational Linguistics.

Shaolei Zhang and Yang Feng. 2022a. Gaussian multi-head attention for simultaneous machine translation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3019–3030, Dublin, Ireland. Association for Computational Linguistics.

Shaolei Zhang and Yang Feng. 2022b. Information-transport-based policy for simultaneous translation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 992–1013, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Shaolei Zhang and Yang Feng. 2022c. Modeling dual read/write paths for simultaneous machine translation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2461–2477, Dublin, Ireland. Association for Computational Linguistics.

Shaolei Zhang and Yang Feng. 2022d. Reducing position bias in simultaneous machine translation with length-aware framework. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6775–6788, Dublin, Ireland. Association for Computational Linguistics.

Shaolei Zhang and Yang Feng. 2023a. Hidden markov transformer for simultaneous machine translation. In *The Eleventh International Conference on Learning Representations*.

Shaolei Zhang and Yang Feng. 2023b. Unified segment-to-segment framework for simultaneous sequence generation. In *Advances in Neural Information Processing Systems*, volume 36, pages 45235–45258. Curran Associates, Inc.

Shaolei Zhang, Shoutao Guo, and Yang Feng. 2022. Wait-info policy: Balancing source and target at information level for simultaneous machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2249–2263, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Libo Zhao, Kai Fan, Wei Luo, Wu Jing, Shushu Wang, Ziqian Zeng, and Zhongqiang Huang. 2023. Adaptive policy with wait-k model for simultaneous translation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4816–4832, Singapore. Association for Computational Linguistics.

Baigong Zheng, Kaibo Liu, Renjie Zheng, Mingbo Ma, Hairong Liu, and Liang Huang. 2020. Simultaneous translation policies: From fixed to adaptive. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 2847–2853. Association for Computational Linguistics.

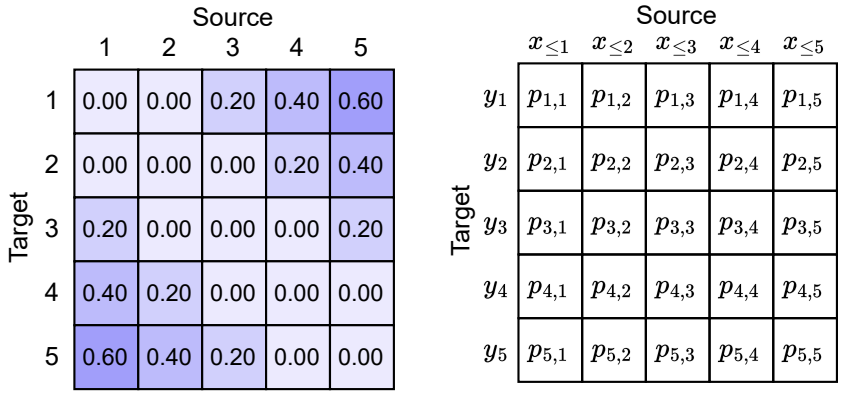
A Explanation of Latency Constraint

In the Streaming Self-Attention (SSA) mechanism, the model obtains the attention probability allocated to the source prefixes by predicting $p_{i,j}$. Subsequently, the model accumulates $p_{i,j}$ to determine the total attention on the input source tokens. The model utilizes $\sum_{j=1}^{m} p_{i,j}$ to assess the sufficiency of the source information, thereby acquiring the translation policy. The model generates target tokens when it deems the source information sufficient. To ensure that the model generates translations tightly after reading the necessary source information, we introduce the latency constraint that encourages the model to utilize as few source tokens as possible to generate translations.

According to Zhang and Feng (2022b), the alignments conducive to latency tend to concentrate near the diagonal between source and target sentences. Therefore, we introduce $\mathcal{L}_{lat}$ to encourage SSA to pay more attention to the source prefixes near the diagonal. The intuitive illustration of the cost matrix $\mathbf{C} = (C_{i,j})_{I \times J}$ and the attention allocation matrix $\mathbf{p} = (p_{i,j})_{I \times J}$ is shown in Figure 5.

B System Settings

The system settings on three translation tasks are shown in Table 7.



(a) Cost Matrix (b) Attention Allocation Matrix

Figure 5: The illustration of cost matrix and attention allocation matrix. In this diagram, I and J are both set to 5, and ϵ is set to 1.

Our method is based on fine-tuning full-sentence translation model. During the full-sentence training stage, our model adopts the traditional decoder layer (Vaswani et al., 2017) and is trained with cross-entropy loss. Once we obtain a well-performing translation model, we then add policy-specific parameters and train the model according to our proposed strategy in Section 3.4.

For more detailed implementation issues, please refer to our code.

C Detailed Results

In addition to the performance comparison in Figure 3, we also present the numerical results for our method. Table 8, 9, and 10 respectively describe the performance of DST on IWSLT15 En→Vi, WMT16 En→Ro, and WMT15 De→En translation tasks. Each task is evaluated using latency measured by AL (Ma et al., 2019) and translation quality measured by BLEU (Papineni et al., 2002).

Table 7: Hyperparameters of DST.

Hyperparameters		IWSLT15 En→Vi	WMT16 En→Ro	WMT15 De→En
Decoder	decoder_layers	16	16	16
	decoder_embed_dim	512	512	512
	decoder_ffn_embed_dim	1024	2048	2048
	decoder_attention_heads	4	8	8
Loss	ϵ	1	1	1
	T	2k	4k	30k
Training	dropout	0.1	0.3	0.3
	optimizer	adam	adam	adam
	adam_β	(0.9, 0.98)	(0.9, 0.98)	(0.9, 0.98)
	clip_norm	0	0	0
	lr	5e-4	5e-4	5e-4
	lr_scheduler	inverse_sqrt	inverse_sqrt	inverse_sqrt
	warmup_updates	4000	4000	4000
	warmup_init_lr	1e-7	1e-7	1e-7
	weight_decay	0.0	0.0	0.0
	label_smoothing	0.1	0.1	0.1
	max_tokens	16000	8192×4	8192×4

δ_{infer}	AL	BLEU
0.30	3.89	28.56
0.40	5.32	29.28
0.45	6.37	29.20
0.50	7.70	29.28
0.60	9.75	29.40

Table 8: Numerical results on IWSLT15 En→Vi.

δ_{infer}	AL	BLEU
0.25	0.60	21.01
0.30	3.60	29.80
0.40	4.09	30.73
0.50	5.77	31.63
0.55	7.01	32.62
0.65	9.52	33.08

Table 9: Numerical results on WMT16 En→Ro.

δ_{infer}	AL	BLEU
0.20	3.22	28.57
0.30	4.72	30.55
0.40	5.49	30.89
0.50	7.09	31.85
0.60	9.40	32.22

Table 10: Numerical results on WMT15 De→En.