Universal Simultaneous Machine Translation with Mixture-of-Experts Wait-k Policy

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

Abstract

Simultaneous machine translation (SiMT) generates translation before reading the entire source sentence and hence it has to trade off between translation quality and latency. To fulfill the requirements of different translation quality and latency in practical applications, the previous methods usually need to train multiple SiMT models for different latency levels, resulting in large computational costs. In this paper, we propose a universal SiMT model with *Mixture-of-Experts Wait-k Policy* to achieve the best translation quality under arbitrary latency with only one trained model. Specifically, our method employs multi-head attention to accomplish the mixture of experts where each head is treated as a wait-k expert with its own waiting words number, and given a test latency and source inputs, the weights of the experts are accordingly adjusted to produce the best translation. Experiments on three datasets show that our method outperforms all the strong baselines under different latency, including the state-of-the-art adaptive policy.

1 Introduction

Simultaneous machine translation (SiMT) (Cho and Esipova, 2016; Gu et al., 2017; Ma et al., 2019; Arivazhagan et al., 2019) begins outputting translation before reading the entire source sentence and hence has a lower latency compared to full-sentence machine translation. In practical applications, SiMT usually has to fulfill the requirements with different levels of latency. For example, a live broadcast requires a lower latency to provide smooth translation while a formal conference focuses on translation quality and allows for a slightly higher latency. Therefore, an excellent SiMT model should be able to maintain high translation quality under different latency levels.

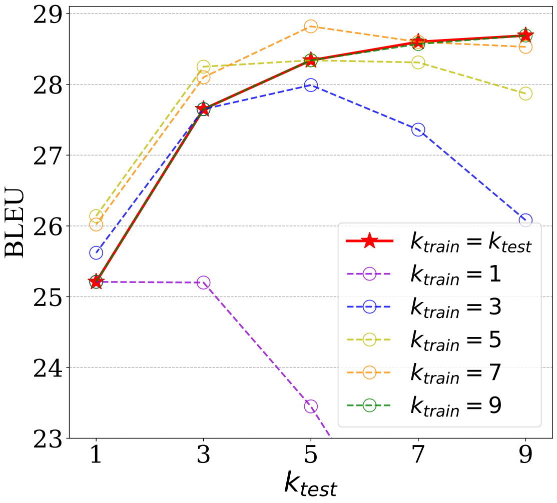


Figure 1: Performance of wait-k models with different k_{train} v.s. k_{test} on IWSLT15 En→Vi SiMT task. k_{train} and k_{test} mean the number of source tokens to wait before performing translation during training and testing, respectively.

However, the existing SiMT methods, which usually employ fixed or adaptive policy, cannot achieve the best translation performance under different latency with only one model (Ma et al., 2019, 2020). With fixed policy, e.g., wait-k policy (Ma et al., 2019), the SiMT model has to wait for a fixed number of source words to be fed and then read one source word and output one target word alternately. In wait-k policy, the number of words to wait for can be different during training and testing, denoted as k_{train} and k_{test} respectively, and the latency is determined by k_{test}. Figure 1 gives the performance of the model trained with k_{train} under different k_{test}, and the results show that under different k_{test} the SiMT model with the best performance corresponds to different k_{train}. As a result, multiple models should be maintained for the best performance under different latency. With adaptive policy, the SiMT model dynamically adjusts the waiting of source tokens for better translation by directly involving the latency in the loss function (Arivazhagan et al., 2019; Ma et al., 2020). Although the adaptive policy achieves the state-of-

[*] Corresponding author: Yang Feng.

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 7306–7317
November 7–11, 2021. ©2021 Association for Computational Linguistics

the-art performance on the open datasets, multiple models need to be trained for different latency as the change of model latency is realized by the alteration of the loss function during training. Therefore, to perform SiMT under different latency, both kinds of methods require training multiple models for different latency, leading to large costs.

Under these grounds, we propose a universal simultaneous machine translation model which can self-adapt to different latency, so that only one model is trained for different latency. To this end, we propose a *Mixture-of-Experts Wait-k Policy* (*MoE wait-k policy*) for SiMT where each expert employs the wait-k policy with its own number of waiting source words. For the mixture of experts, we can consider that different experts correspond to different parameter subspaces (Zhang et al., 2021), and fortunately the multi-head attention is designed to explore different subspaces with different heads (Vaswani et al., 2017). Therefore, we employ multi-head attention as the implementation manner of MoE by assigning different heads with different waiting words number (wait-1,wait-3,wait-5,$\cdots$). Then, the outputs of different heads (aka experts) are combined with different weights, which are dynamically adjusted to achieve the best translation under different latency.

Experiments on IWSLT15 En→Vi, WMT16 En→Ro and WMT15 De→En show that although with only a universal SiMT model, our method can outperform strong baselines under all latency, including the state-of-the-art adaptive policy. Further analyses show the promising improvements of our method on efficiency and robustness.

2 Background

Our method is based on mixture-of-experts approach, multi-head attention and wait-k policy, so we first briefly introduce them respectively.

2.1 Mixture of Experts

Mixture of experts (MoE) (Jacobs et al., 1991; Eigen et al., 2013; Shazeer et al., 2017; Peng et al., 2020) is an ensemble learning approach that jointly trains a set of expert modules and mixes their outputs with various weights:

$$\mathrm{MoE} = \sum_{i=1}^{n} G_i \cdot \mathbf{E}_i \tag{1}$$

where n is the number of experts, $\mathbf{E}_i$ and G_i are the outputs and weight of the i^{th} expert, respectively.

2.2 Multi-head Attention

Multi-head attention is the key component of the state-of-the-art Transformer architecture (Vaswani et al., 2017), which allows the model to jointly attend to information from different representation subspaces. Multi-head attention contains h attention heads, where each head independently calculates its outputs between queries, keys and values through scaled dot-product attention. Since our method and wait-k policy are applied to cross-attention, the following formal expressions are all based on cross-attention, where the queries come from the t^{th} decoder hidden state $\mathbf{S}_t$, and the keys and values come from the encoder outputs $\mathbf{Z}$. Thus, the outputs $\widetilde{\mathbf{H}}_i^t$ of the i^{th} head when decoding the t^{th} target token is calculated as:

$$\begin{aligned}\widetilde{\mathbf{H}}_i^t &= f_{att}\left(\mathbf{S}_t, \mathbf{Z}, \mathbf{Z}; \boldsymbol{\theta}_i\right) \\ &= \mathrm{softmax}\left(\frac{\mathbf{S}_t\mathbf{W}_i^Q\left(\mathbf{Z}\mathbf{W}_i^K\right)^\top}{\sqrt{d_k}}\right)\mathbf{Z}\mathbf{W}_i^V\end{aligned} \tag{2}$$

where $f_{att}\left(\cdot; \boldsymbol{\theta}_i\right)$ represents dot-product attention of the i^{th} head, $\mathbf{W}_i^Q$, $\mathbf{W}_i^K$ and $\mathbf{W}_i^V$ are learned projection matrices, $\sqrt{d_k}$ is the dimension of keys. Then, the outputs of h heads are concatenated and fed through a learned output matrix $\mathbf{W}^O$ to calculate the context vector $\mathbf{C}_t$:

$$\mathbf{C}_t = \mathrm{MultiHead}\left(\mathbf{S}_t, \mathbf{Z}, \mathbf{Z}\right) = \left[\widetilde{\mathbf{H}}_1^t, \cdots, \widetilde{\mathbf{H}}_h^t\right]\mathbf{W}^O \tag{3}$$

2.3 Wait-k Policy

Wait-k policy (Ma et al., 2019) refers to first waiting for k source tokens and then reading and writing one token alternately. Since k is input from the outside of the model, we call k the *external lagging*. We define $g(t)$ as a monotonic non-decreasing function of t, which represents the number of source tokens read in when generating the t^{th} target token. In particular, for wait-k policy, given external lagging k, $g(t; k)$ is calculated as:

$$g(t; k) = \min\{k + t - 1, |\mathbf{Z}|\}, \ t = 1, 2, \cdots \tag{4}$$

In the wait-k policy, the source tokens processed by the encoder are limited to the first $g(t; k)$ tokens when generating the t^{th} target token. Thus, each head outputs in the cross-attention is calculated as:

$$\mathbf{H}_i^t = f_{att}\left(\mathbf{S}_t, \mathbf{Z}_{\leq g(t;k)}, \mathbf{Z}_{\leq g(t;k)}; \ \boldsymbol{\theta}_i\right) \tag{5}$$

where $\mathbf{Z}_{\leq g(t;k)}$ represents the encoder outputs when the first $g(t; k)$ source tokens are read in.

The standard wait-k policy (Ma et al., 2019) trains a set of SiMT models, where each model is trained through a fixed wait-k_{train} and tested with corresponding wait-k_{test} ($k_{test}=k_{train}$). Elbayad et al. (2020a) proposed multipath training, which uniformly samples k_{train} in each batch during training. However, training with both $k_{train}=1$ and $k_{train}=\infty$ definitely make the model parameters confused between different subspace distributions.

3 The Proposed Method

In this section, we first view multi-head attention from the perspective of the mixture of experts, and then introduce our method based on it.

3.1 Multi-head Attention from MoE View

Multi-head attention can be interpreted from the perspective of the mixture of experts (Peng et al., 2020), where each head acts as an expert. Thus, Eq.(3) can be rewritten as:

$$\begin{aligned}\mathbf{C}_t &= \mathrm{MultiHead}\left(\mathbf{S}_t,\mathbf{Z},\mathbf{Z}\right)=\left[\widetilde{\mathbf{H}}_1^t,\cdots,\widetilde{\mathbf{H}}_h^t\right]\mathbf{W}^O\\ &=\left[\widetilde{\mathbf{H}}_1^t,\cdots,\widetilde{\mathbf{H}}_h^t\right]\left[\mathbf{W}_1^O,\cdots,\mathbf{W}_h^O\right]^\top\\ &=\sum_{i=1}^{h}\widetilde{\mathbf{H}}_i^t\mathbf{W}_i^O \;=\sum_{i=1}^{h}\frac{1}{h}\cdot h\widetilde{\mathbf{H}}_i^t\mathbf{W}_i^O\\ &=\sum_{i=1}^{h}\widetilde{G}_i^t\cdot\widetilde{\mathbf{E}}_i^t \qquad (6)\end{aligned}$$

$$\text{where}\quad \widetilde{G}_i^t=\frac{1}{h}\quad,\quad \widetilde{\mathbf{E}}_i^t=h\widetilde{\mathbf{H}}_i^t\mathbf{W}_i^O \qquad (7)$$

$\left[\mathbf{W}_1^O,\cdots,\mathbf{W}_h^O\right]^\top$ is a row-wise block sub-matrix representation of $\mathbf{W}^O$. $\widetilde{\mathbf{E}}_i^t$ is the outputs of the i^{th} expert at step t, and $\widetilde{G}_i^t\in\mathbb{R}$ is the weight of $\widetilde{\mathbf{E}}_i^t$. Therefore, multi-head attention can be regarded as a mixture of experts, where experts have the same function but different parameters ($\widetilde{\mathbf{E}}_i^t=h\widetilde{\mathbf{H}}_i^t\mathbf{W}_i^O$) and the normalized weights are equal ($\widetilde{G}_i^t=\frac{1}{h}$).

3.2 Mixture-of-Experts Wait-k Policy

To get a universal model which can perform SiMT with a high translation quality under arbitrary latency, we introduce the *Mixture-of-Experts Wait-k Policy (MoE wait-k)* into SiMT to redefine the experts $\widetilde{\mathbf{E}}_i^t$ and weights $\widetilde{G}_i^t$ in multi-head attention (Eq.(7)). As shown in Figure 2, experts are given different functions, i.e., performing wait-k policy with different latency, and their outputs are denoted as $\left\{\mathbf{E}_i^t\right\}_{i=1}^h$. Meanwhile, under the premise of normalization, the weights of experts are no longer equal but dynamically adjusted according to source input and latency requirement, denoted as $\left\{G_i^t\right\}_{i=1}^h$. The details are introduced following.

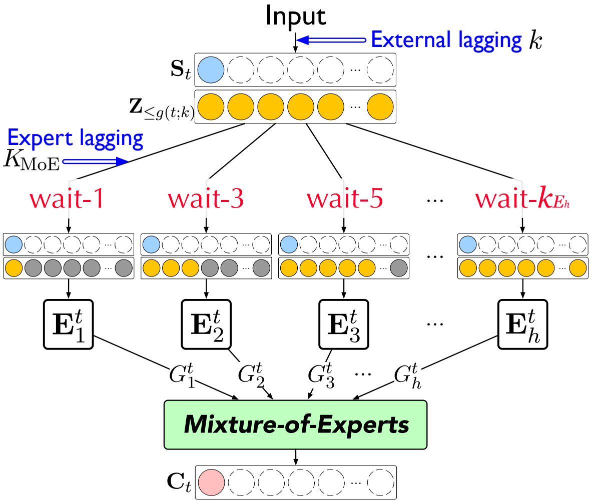


Figure 2: The architecture of the mixture-of-experts wait-k policy. Each expert performs wait-k under different lagging (such as wait-1,wait-3,wait-5,$\cdots$), and then their outputs are combined with different weights.

3.2.1 Experts with Different Functions

The experts in our method are divided into different functions, where each expert performs SiMT with different latency. In addition to the external lagging k in standard wait-k policy, we define *expert lagging* $K_{\mathrm{MoE}}=[k_{E_1},\cdots,k_{E_h}]$, where k_{E_i} is the hyperparameter we set to represent the fixed lagging of the i^{th} expert. For example, for a Transformer with 8 heads, if we set $K_{\mathrm{MoE}}=[1,3,5,7,9,11,13,15]$, then each expert corresponds to one head and 8 experts concurrently perform wait-1, wait-3, wait-5,$\cdots$, wait-15 respectively. Specifically, given K_{MoE}, the outputs $\mathbf{H}_i^t$ of the i^{th} head at step t is calculated as:

$$\begin{aligned}\mathbf{H}_i^t=f_{att}\Big(\mathbf{S}_t,\ &\mathbf{Z}_{\leq\min\left(g\left(t;k_{E_i}\right),g(t;k)\right)},\\ &\mathbf{Z}_{\leq\min\left(g\left(t;k_{E_i}\right),g(t;k)\right)};\ \boldsymbol{\theta}_i\Big) \qquad (8)\end{aligned}$$

where $g\left(t;k_{E_i}\right)$ is the number of source tokens processed by the i^{th} expert at step t and $g\left(t;k\right)$ is the number of all available source tokens read in at step t. During training, k is uniformly sampled in each batch with multipath training (Elbayad et al., 2020a). During testing, k is the input test lagging.

Then, the outputs $\mathbf{E}_i^t$ of the i^{th} expert when generating t^{th} target token is calculated as:

$$\mathbf{E}_i^t=h\mathbf{H}_i^t\mathbf{W}_i^O \qquad (9)$$

3.2.2 Dynamic Weights for Experts

Each expert has a clear division of labor through expert lagging K_{MoE}. Then for different input and latency, we dynamically weight each expert with the predicted $\left\{G_i^t\right\}_{i=1}^h$, where $G_i^t \in \mathbb{R}$ can be considered as the confidence of expert outputs $\mathbf{E}_i^t$. The factor to predict G_i^t consists of two components:

- e_i^t: The average cross-attention scores in the i^{th} expert at step t, which are averaged over all source tokens read in (Zheng et al., 2019a).
- k: External lagging k in Eq.(8).

At step t, all e_i^t and k are concatenated and fed through the multi-layer perceptron (MLP) to predict the confidence score β_i^t of the i^{th} expert, which are then normalized to calculate the weight G_i^t:

$$\beta_i^t = \tanh\left([e_1^t; \cdots; e_h^t; k]\mathbf{W}_i + b_i \right) \tag{10}$$

$$G_i^t = \frac{\exp(\beta_i^t)}{\sum_{l=1}^{h} \exp(\beta_l^t)} \tag{11}$$

where $\mathbf{W}_i$ and b_i are parameters of MLP to predict G_i^t. Given expert outputs $\left\{\mathbf{E}_i^t\right\}_{i=1}^h$ and weights $\left\{G_i^t\right\}_{i=1}^h$, the context vector $\mathbf{C}_t$ is calculatas:

$$\mathbf{C}_t = \sum_{i=1}^{h} G_i^t \cdot \mathbf{E}_i^t \tag{12}$$

The algorithm details of proposMoE wait-k policy are shown in Algorithm 1. At decoding step t, each expert performs the wait-k policy with different latency according to the expert lagging K_{MoE}, and then the expert outputs are dynamically weighted to calculate the context vector $\mathbf{C}_t$.

3.2.3 Training Method

We apply a two-stage training, both of which apply multipath training (Elbayad et al., 2020a), i.e., randomly sampling k (k in Eq.(8)) in every batch during training. *First-stage*: Fix the weights G_i^t equal to $\frac{1}{h}$ and pre-train expert parameters. *Second-stage*: jointly fine-tune the parameters of experts and their weights. In the inference time, the universal model is tested with arbitrary latency (test lagging). In Sec.5, we compare the proposed two-stage training method with the one-stage training method which directly trains the parameters of experts and their weights together.

We tried the block coordinate descent (BCD) training (Peng et al., 2020) which is proposed to train the experts in the same function, but it is not suitable for our method, as the experts in MoE wait-k have already assigned different functions. Therefore, our method can be stably trained through back-propagation directly.

Algorithm 1: MoE Wait-k Policy

```
Input  : Encoder output $\mathbf{Z}$ (incomplete),
         Decoder hidden state $\mathbf{S}_t$,
         Expert lagging $K_{\mathrm{MoE}}$,
         Test lagging $k_{test}$ (only in testing)
Output : Context vector $\mathbf{C}_t$

if is_Training then              // In training
    $k \leftarrow$ Sample from( $[1, 2, \cdots, |\mathbf{Z}|]$ )
else                             // In testing
    $k \leftarrow k_{test}$
end

for $k_{E_i}$ in $K_{\mathrm{MoE}}$ do
    calculate $\mathbf{Z}_{\leq \min\left(g\left(t; k_{E_i}\right), g(t;k)\right)}$
end

for $i \leftarrow 1$ to $h$ do
    calculate $\mathbf{E}_i^t$ according to Eq.(8, 9)
    calculate $\mathbf{G}_i^t$ according to Eq.(10, 11)
end
calculate $\mathbf{C}_t$ according to Eq.(12)

Return $\mathbf{C}_t$
```

4 Related Work

Mixture of experts MoE was first proposed in multi-task learning (Jacobs et al., 1991; Caruana et al., 2004; Liu et al., 2018; Ma et al., 2018; Dutt et al., 2020). Recently, Shazeer et al. (2017) applied MoE in sequence learning. Some work (He et al., 2018; Shen et al., 2019; Cho et al., 2019) applied MoE in diversity generation. Peng et al. (2020) applied MoE in MT and combined $h-1$ heads in Transformer as an expert.

Previous works always applied MoE for diversity. Our method makes the experts more regular in parameter space, which provides a method to improves the translation quality with MoE.

SiMT Early read / write policies in SiMT used segmented translation (Bangalore et al., 2012; Cho and Esipova, 2016; Siahbani et al., 2018). Grissom II et al. (2014) predicted the final verb in SiMT. Gu et al. (2017) trained a read / write agent with reinforcement learning. Alinejad et al. (2018) added a predict operation based on Gu et al. (2017).

Recent read / write policies fall into two categories: fixed and adaptive. For the fixed policy, Dalvi et al. (2018) proposed STATIC-RW, and Ma et al. (2019) proposed wait-k policy, which always generates target k tokens lagging behind the source. Elbayad et al. (2020a) enhanced wait-k policy by sampling different k during training. Han et al. (2020) applied meta-learning in wait-k. Zhang et al. (2021) proposed future-guided training for wait-k policy. Zhang and Feng (2021) proposed a char-level wait-k policy. For the adaptive policy, Zheng et al. (2019a) trained an agent with gold read / write sequence. Zheng et al. (2019b) added a "delay" token $\{\varepsilon\}$ to read. Arivazhagan et al. (2019) proposed MILk, which used a Bernoulli variable to determine writing. Ma et al. (2020) proposed MMA, which is the implementation of MILk on the Transformer. Zheng et al. (2020a) ensembled multiple wait-k models to develop a adaptive policy. Zhang and Zhang (2020) and Zhang et al. (2020) proposed adaptive segmentation policies. Bahar et al. (2020) and Wilken et al. (2020) proposed alignment-based chunking policy.

A common weakness of the previous methods is that they all train separate models for different latency. Our method only needs a universal model to complete SiMT under all latency, and meanwhile achieve better translation quality.

5 Experiments

5.1 Datasets

We evaluated our method on the following three datasets, the scale of which is from small to large.

IWSLT15[1] English→Vietnamese (En-Vi) (133K pairs) (Cettolo et al., 2015) We use TED tst2012 (1553 pairs) as the validation set and TED tst2013 (1268 pairs) as the test set. Following Raffel et al. (2017) and Ma et al. (2020), we replace tokens that the frequency less than 5 by $\langle unk \rangle$. After replacement, the vocabulary sizes are 17K and 7.7K for English and Vietnamese, respectively.

WMT16[2] English→Romanian (En-Ro) (0.6M pairs) (Lee et al., 2018) We use news-dev2016 (1999 pairs) as the validation set and news-test2016 (1999 pairs) as the test set.

WMT15[3] German→English (De-En) (4.5M pairs) Following the setting from Ma et al. (2019) and Ma et al. (2020), we use newstest2013 (3000 pairs) as the validation set and newstest2015 (2169 pairs) as the test set.

For En-Ro and De-En, BPE (Sennrich et al., 2016) is applied with 32K merge operations and the vocabulary is shared across languages.

5.2 System Settings

We conducted experiments on following systems.

Offline Conventional Transformer (Vaswani et al., 2017) model for full-sentence translation, decoding with greedy search.

Standard Wait-k Standard wait-k policy proposed by Ma et al. (2019). When evaluating with the test lagging k_{test}, we apply the result from the model trained with k_{train}, where $k_{train} = k_{test}$.

Optimal Wait-k An optimal variation of standard wait-k. When decoding with k_{test}, we traverse all models trained with different k_{train} and apply the optimal result among them. For example, if the best result when testing with wait-1 ($k_{test} = 1$) comes from the model trained by wait-5 ($k_{train} = 5$), we apply this optimal result. 'Optimal Wait-k' selects the best result according to the reference, so it can be considered as an oracle.

Multipath Wait-k An efficient training method for wait-k policy (Elbayad et al., 2020a). In training, k_{train} is no longer fixed, but randomly sampled from all possible lagging in each batch.

MU A segmentation policy base on meaning units proposed by Zhang et al. (2020), which obtains comparable results with SOTA adaptive policy. At each decoding step, if a meaning unit is detected through a BERT-based classifier, 'MU' feeds the received source tokens into a full-sentence MT model to generate the target token and stop until generating the $<EOS>$ token.

MMA[4] Monotonic multi-head attention (MMA) proposed by (Ma et al., 2020), the state-of-the-art adaptive policy for SiMT, which is the implementation of 'MILk' (Arivazhagan et al., 2019) based on the Transformer. At each decoding step, 'MMA' predicts a Bernoulli variable to decide whether to start translating or wait for the source token.

MoE Wait-k A variation of our method, which directly trains the parameters of experts and their weights together in one-stage training.

Equal-Weight MoE Wait-k A variation of our method. The weight of each expert is fixed to $\frac{1}{h}$.

MoE Wait-k + FT Our method in Sec.3.2.

[1] nlp.stanford.edu/projects/nmt/

[2] www.statmt.org/wmt16/

[3] www.statmt.org/wmt15/

[4] github.com/pytorch/fairseq/tree/master/examples/simultaneous_translation

Architecture	Expert Lagging K_{MoE}
Transformer-Small (4 heads)	[1, 6, 11, 16]
Transformer-Base (8 heads)	[1, 3, 5, 7, 9, 11, 13, 15]
Transformer-Big (16 heads)	[1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15,16]

Table 1: The value of expert lagging K_{MoE} for different Transformer settings.

We compare our method with 'MMA' and 'MU' on De-En(Big) since they report their results on De-En with Transformer-Big.

The implementation of all systems are adapted from Fairseq Library (Ott et al., 2019), and the setting is exactly the same as Ma et al. (2019) and Ma et al. (2020). To verify that our method is effective on Transformer with different head settings, we conduct experiments on three types of Transformer, where the settings are the same as Vaswani et al. (2017). For En-Vi, we apply **Transformer-Small** (4 heads). For En-Ro, we apply **Transformer-Base** (8 heads). For De-En, we apply both **Transformer-Base** and **Transformer-Big** (16 heads). Table 2 reports the parameters of different SiMT systems on De-En(Big). To perform SiMT under different latency, both 'Standard Wait-k', 'Optimal Wait-k' and 'MMA' require multiple models, while 'Multipath Wait-k', 'MU' and 'MoE Wait-k' only need one trained model.

Expert lagging K_{MoE} in MoE wait-k is the hyperparameter we set, which represents the lagging of each expert. We did not conduct many searches on K_{MoE}, but set it to be uniformly distributed in a reasonable lagging interval, as shown in Table 1. We will analyze the influence of different settings of K_{MoE} in our method in Sec.6.5.

We evaluate these systems with BLEU (Post, 2018) for translation quality and Average Lagging (AL[5]) (Ma et al., 2019) for latency. Given $g(t)$, latency metric AL is calculated as:

$$\text{AL} = \frac{1}{\tau}\sum_{t=1}^{\tau} g(t) - \frac{t-1}{|\mathbf{y}| \,/\, |\mathbf{x}|} \tag{13}$$

$$\text{where} \quad \tau = \operatorname*{argmax}_{t} \left(g(t) = |\mathbf{x}| \right) \tag{14}$$

where $|\mathbf{x}|$ and $|\mathbf{y}|$ are the length of the source sentence and target sentence respectively.

Systems	#Para. per Model	Model Num.	Total #Para.
Offline	209.91M	1	209.91M
Wait-k	209.91M	5	1049.55M
Optimal	209.91M	5	1049.55M
Mulitpath	209.91M	1	209.91M
MMA	222.51M	7	1557.57M
MU	319.91M	1	319.91M
MoE Wait-k	209.91M	1	209.91M

Table 2: The parameters of SiMT systems on De-En(Transformer-Big) in our experiments. '#Para. per model': The parameters of a single SiMT model. 'Model Num.': The number of SiMT models required to perform SiMT under multiple latency. 'Total #Para.': The total parameters of the SiMT system.

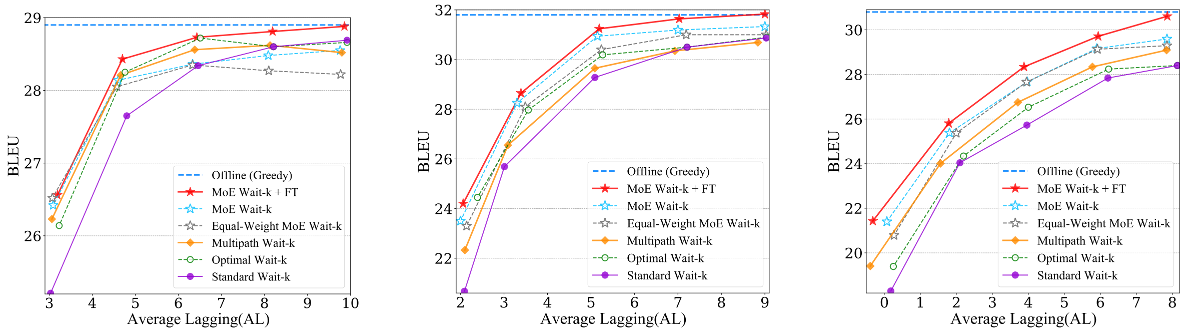
5.3 Main Results

Figure 3 and Figure 4 show the comparison between our method and the previous methods on Transformer with the various head settings. In all settings, 'MoE wait-k + FT' outperforms the previous methods under all latency. Our method improves the performance of SiMT much closer to the offline model, which almost reaches the performance of full-sentence MT when lagging 9 tokens.

Compared with 'Standard Wait-k', our method improves 0.60 BLEU on En-Vi, 2.11 BLEU on En-Ro, 2.33 BLEU on De-En(Base), and 2.56 BLEU on De-En(Big), respectively (average on all latency). More importantly, our method only needs one well-trained universal model to complete SiMT under all latency, while 'Standard wait-k' requires training different models for each latency. Besides, 'Optimal Wait-k' traverses many models to obtain the optimal result under each latency. Our method dynamically weights experts according to the test latency, and outperforms 'Optimal Wait-k' under all latency, without searching among many models.

Both our method and 'Multipath Wait-k' can train a universal model, but our method avoids the mutual interference between different sampled k during training. 'Multipath Wait-k' often improves the translation quality under low latency, but on the contrary, the translation quality under high latency is poor (Elbayad et al., 2020b). The reason is that sampling a slightly larger k in training improves the translation quality under low latency (Ma et al., 2019; Zhang et al., 2021), but sampling a smaller k destroys the translation quality under high latency. Our method introduces expert lagging and dynam-

[5] github.com/SimulTrans-demo/STACL.



(a) En-Vi, Transformer-Small

(b) En-Ro, Transformer-Base

(c) De-En, Transformer-Base

Figure 3: Translation quality (BLEU) against latency (AL) on the En-Vi(Small), En-Ro(Base), De-En(Base). We show the result of our methods, Standard wait-k, Optimal Wait-k, Multipath Wait-k and offline model.

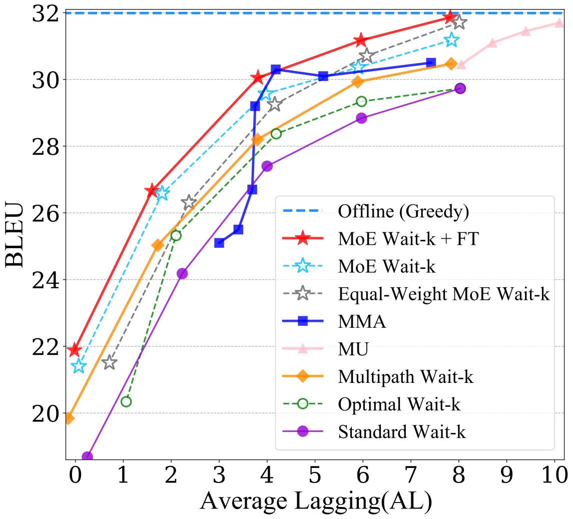


Figure 4: Translation quality (BLEU) against latency (AL) on the De-En with Transformer-Big. We show the result of our methods, Standard wait-k, Optimal Wait-k, Multipath Wait-k, MU, MMA (the current SOTA adaptive policy) and offline model.

ical weights, avoiding the interference caused by multipath training.

Compared with 'MMA' and 'MU', our method performs better. 'MU' sets a threshold to perform SiMT under different latency and achieves good translation quality, but it is difficult to complete SiMT under low latency as it is a segmentation policy. As a fixed policy, our method maintains the advantage of simple training and meanwhile catches up with the adaptive policy 'MMA' on translation quality, which is uplifting. Furthermore, our method only needs a universal model to perform SiMT under different latency and the test latency can be set artificially, which is impossible for the previous adaptive policy.

5.4 Ablation Study

We conducted ablation studies on the dynamic weights and two-stage training, as shown in Figure 3 and Figure 4. The translation quality decreases significantly when each expert is set to equal-weight. Our method dynamically adjusts the weight of each expert according to the input and test lagging, resulting in concurrently performing well under all latency. For the training methods, the two-stage training method makes the training of weights more stable, thereby improving the translation quality, especially under high latency.

6 Analysis

We conducted extensive analyses to understand the specific improvements of our method. Unless otherwise specified, all the results are reported on De-En with Transformer-Base(8 heads).

6.1 Performance on Various Difficulty Levels

The difference between the target and source word order is one of the challenges of SiMT, where many word order inversions force to start translating before reading the aligned source words. To verify the performance of our method on SiMT with various difficulty levels, we evenly divided the test set into three parts: EASY, MIDDLE and HARD. Specifically, we used `fast-align`[6] (Dyer et al., 2013) to align the source with the target, and then calculated the number of crosses in the alignments (number of reversed word orders), which is used as a basis to divide the test set (Chen et al., 2020; Zhang et al., 2021). After the division, the alignments in the EASY set are basically monotonous,

[6] https://github.com/clab/fast_align

k_{test}	EASY			MIDDLE			HARD		
	Wait-k	Ours	Δ	Wait-k	Ours	Δ	Wait-k	Ours	Δ
1	19.27	21.79	+2.52	18.70	21.87	+3.17	16.14	20.04	**+3.90**
3	28.79	30.19	+1.40	24.88	25.65	+0.77	21.30	23.81	**+2.51**
5	31.15	33.80	**+2.65**	26.56	29.03	+2.47	24.02	25.73	+1.71
7	32.62	34.68	**+2.06**	28.52	30.42	+1.90	25.65	27.37	+1.72
9	32.52	35.08	**+2.56**	28.94	31.42	+2.48	26.66	28.40	+1.74

Table 3: Improvement of our method on SiMT with various difficult levels, evaluated with wait-k_{test}. The difficult levels are divided according to the word order difference between the source sentence and the target sentence.

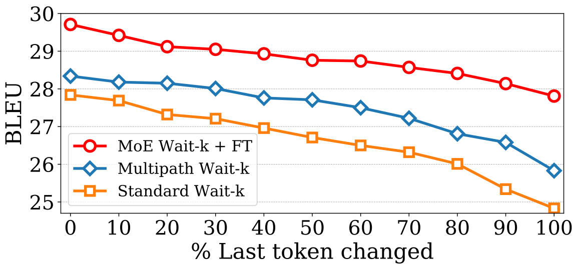
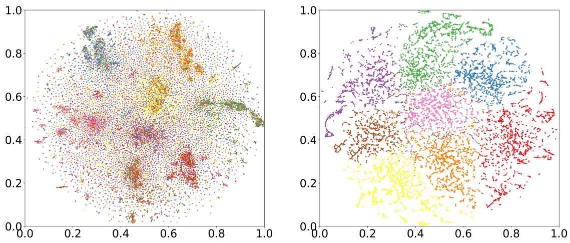


Figure 5: Degradation of performance as the noise of last source token increases, evaluated with wait-7.



(a) Multipath Wait-k
(b) MoE Wait-k + FT

Figure 6: Subspace distribution of expert outputs. Each color represents the outputs of an expert.

and the sentence pairs in the HARD set contains at least 12 reversed word orders.

Our method outperforms the standard wait-k on all difficulty levels, especially improving 3.90 BLEU on HARD set under low latency. HARD set contains a lot of word order reversal, which is disastrous for low-latency SiMT such as testing with wait-1. The standard wait-k enables the model to gain some implicit prediction ability (Ma et al., 2019), and our method further strengthens it. MoE wait-k introduces multiple experts with varying expert lagging, of which the larger expert lagging helps the model to improve the implicit prediction ability (Zhang et al., 2021), while the smaller expert lagging avoids learning too much future information during training and prevents the illusion caused by over-prediction (Chen et al., 2020). With MoE wait-k, the implicit prediction ability is stronger and more stable.

6.2 Improvement on Robustness

Robustness is another major challenge for SiMT (Zheng et al., 2020b). SiMT is often used as a downstream task of streaming automatic speech recognition (ASR), but the results of streaming ASR are not stable, especially the last recognized source token (Li et al., 2020; Gaido et al., 2020; Zheng et al., 2020b). In each decoding step, we randomly modified the last source token with different proportions, and the results are shown in Figure 5.

Our method is more robust with the noisy last token, owing to multiple experts. Due to different expert lagging, the number of source tokens processed by each expert is different and some experts do not consider the last token. Thus, the noisy last token only affects some experts, while other experts would not be disturbed, giving rise to robustness.

6.3 Differentiation of Experts Distribution

Our method clearly divides the experts into different functions and integrates the expert outputs from different subspaces for better translation. For 'Multipath Wait-k' and our method, we sampled 200 cases and reduced the dimension of the expert outputs (evaluating with wait-5) with the t-Distributed Stochastic Neighbor Embedding (tSNE) technique, and shown the subspace distribution of the expert outputs in Figure 6.

The expert outputs in 'Multipath Wait-k' have a little difference but most of them are fused together, which shows some similarities in heads. In our method, due to the clear division of labor, the expert outputs are significantly different and regular in the subspace distribution, which proves to be beneficial to translation (Li et al., 2018). Besides, our method

		$\mathbf{E_1}$	$\mathbf{E_2}$	$\mathbf{E_3}$	$\mathbf{E_4}$	$\mathbf{E_5}$	$\mathbf{E_6}$	$\mathbf{E_7}$	$\mathbf{E_8}$	**Optimal Model**
Expert Lagging K_{MoE}		1	3	5	7	9	11	13	15	
Test Lagging	$k_{test}=1$	10.66	**13.90**	13.82	11.67	13.07	13.49	11.89	11.50	$k_{train}=3$
	$k_{test}=3$	9.83	12.88	**13.75**	11.62	13.70	13.51	12.55	12.16	$k_{train}=5$
	$k_{test}=5$	9.35	12.63	13.52	11.61	**13.82**	13.6	12.93	12.54	$k_{train}=9$
	$k_{test}=7$	8.65	12.55	12.82	11.58	14.04	**14.10**	13.53	12.73	$k_{train}=9$
	$k_{test}=9$	8.34	12.32	12.55	11.08	14.33	**14.69**	13.79	12.90	$k_{train}=9$

Table 4: Weight of experts under different latency, averaged on 6 decoder layers at all decoding steps. 'Optimal Model': The optimal standard wait-k model under current test latency, obtained by traversing all models trained with different wait-k_{train}.

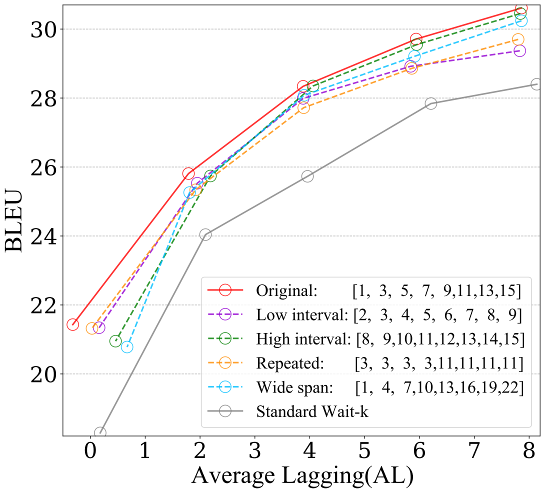


Figure 7: Results of various settings of expert lagging K_{MoE} in MoE wait-k.

has better space utilization and integrate multiple designated subspaces information.

6.4 Superiority of Dynamic Weights

Different expert outputs are dynamically weighted to achieve the best performance under the current test latency, so we calculated the average weight of each expert under different latency in Table 4.

Through dynamic weighting, the expert lagging of the expert with the highest weight is similar to the k_{train} of the optimal model with standard wait-k, meanwhile avoiding the traversal on many trained models. When the test lagging is larger, the expert with larger expert lagging has higher weight; and vice versa. Besides, the expert with a slightly larger expert lagging than k_{test} tends to get the highest weight for better translation, which is in line with the previous conclusions (Ma et al., 2019; Zhang et al., 2021). Furthermore, our method enables the model to comprehensively consider various expert outputs with dynamic weights, thereby getting a more comprehensive translation.

6.5 Effect of Expert Lagging

Expert lagging K_{MoE} is the hyperparameter we set to control the lagging of each expert. We experimented with several settings of K_{MoE} to study the effects of different expert lagging K_{MoE}, as shown in Figure 7.

Totally, all types of K_{MoE} outperform the baseline, and different K_{MoE} only has a slight impact on the performance, which shows that our method is not sensitive to how to set K_{MoE}. Furthermore, there are some subtle differences between different K_{MoE}, where the 'Original' setting performs best. 'Low interval' and 'High interval' only perform well under a part of the latency, as their K_{MoE} is only concentrated in a small lagging interval. 'Repeated' performs not well as the diversity of expert lagging is poor, which lost the advantages of MoE. The performance of 'Wide span' drops under low latency, because the average length of the sentence is about 20 tokens where the much larger lagging is not conducive to low latency SiMT.

In summary, we give a general method for setting expert lagging K_{MoE}. K_{MoE} should maintain diversity and be uniformly distributed in a reasonable lagging interval, such as lagging 1 to 15 tokens.

7 Conclusion and Future Work

In this paper, we propose Mixture-of-Experts Wait-k Policy to develop a universal SiMT, which can perform high quality SiMT under arbitrary latency to fulfill different scenarios. Experiments and analyses show that our method achieves promising results on performance, efficiency and robustness.

In the future, since MoE wait-k develops a universal SiMT model with high quality, it can be applied as a SiMT kernel to cooperate with refined external policy, to further improve performance.

Acknowledgements

We thank all the anonymous reviewers for their insightful and valuable comments. This work was supported by National Key R&D Program of China (NO. 2017YFE0192900).

References

Ashkan Alinejad, Maryam Siahbani, and Anoop Sarkar. 2018. Prediction improves simultaneous neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3022–3027, Brussels, Belgium. 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. pages 1313–1323.

Parnia Bahar, Patrick Wilken, Tamer Alkhouli, Andreas Guta, Pavel Golik, Evgeny Matusov, and Christian Herold. 2020. Start-before-end and end-to-end: Neural speech translation by AppTek and RWTH Aachen University. In *Proceedings of the 17th International Conference on Spoken Language Translation*, pages 44–54, Online. Association for Computational Linguistics.

Srinivas Bangalore, Vivek Kumar Rangarajan Sridhar, Prakash Kolan, Ladan Golipour, and Aura Jimenez. 2012. Real-time incremental speech-to-speech translation of dialogs. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 437–445, Montréal, Canada. Association for Computational Linguistics.

Rich Caruana, Alexandru Niculescu-Mizil, Geoff Crew, and Alex Ksikes. 2004. Ensemble selection from libraries of models. In *Proceedings of the Twenty-First International Conference on Machine Learning*, ICML '04, page 18, New York, NY, USA. Association for Computing Machinery.

Mauro Cettolo, Niehues Jan, Stüker Sebastian, Luisa Bentivogli, R. Cattoni, and Marcello Federico. 2015. The iwslt 2015 evaluation campaign.

Junkun Chen, Renjie Zheng, Atsuhito Kita, Mingbo Ma, and Liang Huang. 2020. Improving Simultaneous Translation with Pseudo References. *arXiv e-prints*, page arXiv:2010.11247.

Jaemin Cho, Minjoon Seo, and Hannaneh Hajishirzi. 2019. Mixture content selection for diverse sequence generation. 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 3121–3131, Hong Kong, China. Association for Computational Linguistics.

Kyunghyun Cho and Masha Esipova. 2016. Can neural machine translation do simultaneous translation?

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.

Anuvabh Dutt, Denis Pellerin, and Georges Quénot. 2020. Coupled ensembles of neural networks. *Neurocomputing*, 396:346–357.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of IBM model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648, Atlanta, Georgia. Association for Computational Linguistics.

David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever. 2013. Learning factored representations in a deep mixture of experts.

Maha Elbayad, Laurent Besacier, and Jakob Verbeek. 2020a. Efficient Wait-k Models for Simultaneous Machine Translation.

Maha Elbayad, Ha Nguyen, Fethi Bougares, Natalia Tomashenko, Antoine Caubrière, Benjamin Lecouteux, Yannick Estève, and Laurent Besacier. 2020b. ON-TRAC consortium for end-to-end and simultaneous speech translation challenge tasks at IWSLT 2020. In *Proceedings of the 17th International Conference on Spoken Language Translation*, pages 35–43, Online. Association for Computational Linguistics.

Marco Gaido, Mattia A. Di Gangi, Matteo Negri, and Marco Turchi. 2020. End-to-end speech-translation with knowledge distillation: FBK@IWSLT2020. In *Proceedings of the 17th International Conference on Spoken Language Translation*, pages 80–88, Online. Association for Computational Linguistics.

Alvin Grissom II, He He, Jordan Boyd-Graber, John Morgan, and Hal Daumé III. 2014. Don't until the final verb wait: Reinforcement learning for simultaneous machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1342–1352, Doha, Qatar. Association for Computational Linguistics.

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.

Hou Jeung Han, Mohd Abbas Zaidi, Sathish Reddy Indurthi, Nikhil Kumar Lakumarapu, Beomseok Lee, and Sangha Kim. 2020. End-to-end simultaneous translation system for IWSLT2020 using modality agnostic meta-learning. In *Proceedings of the 17th International Conference on Spoken Language Translation*, pages 62–68, Online. Association for Computational Linguistics.

Xuanli He, Gholamreza Haffari, and Mohammad Norouzi. 2018. Sequence to sequence mixture model for diverse machine translation. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 583–592, Brussels, Belgium. Association for Computational Linguistics.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. 1991. Adaptive Mixtures of Local Experts. *Neural Computation*, 3(1):79–87.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1173–1182, Brussels, Belgium. Association for Computational Linguistics.

Jian Li, Zhaopeng Tu, Baosong Yang, Michael R. Lyu, and Tong Zhang. 2018. Multi-head attention with disagreement regularization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2897–2903, Brussels, Belgium. Association for Computational Linguistics.

Minqin Li, Haodong Cheng, Yuanjie Wang, Sijia Zhang, Liting Wu, and Yuhang Guo. 2020. BIT's system for the AutoSimTrans 2020. In *Proceedings of the First Workshop on Automatic Simultaneous Translation*, pages 37–44, Seattle, Washington. Association for Computational Linguistics.

Xuanqing Liu, Minhao Cheng, Huan Zhang, and Cho-Jui Hsieh. 2018. Towards robust neural networks via random self-ensemble. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H. Chi. 2018. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, page 1930–1939, New York, NY, USA. Association for Computing Machinery.

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, Juan Miguel Pino, James Cross, Liezl Puzon, and Jiatao Gu. 2020. Monotonic multihead attention. In *International Conference on Learning Representations*.

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.

Hao Peng, Roy Schwartz, Dianqi Li, and Noah A. Smith. 2020. A mixture of h - 1 heads is better than h heads. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6566–6577, Online. 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.

Colin Raffel, Minh-Thang Luong, Peter J. Liu, Ron J. Weiss, and Douglas Eck. 2017. Online and linear-time attention by enforcing monotonic alignments. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2837–2846. PMLR.

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.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer.

Tianxiao Shen, Myle Ott, Michael Auli, and Marc'Aurelio Ranzato. 2019. Mixture models for diverse machine translation: Tricks of the trade. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5719–5728. PMLR.

Maryam Siahbani, Hassan Shavarani, Ashkan Alinejad, and Anoop Sarkar. 2018. Simultaneous translation using optimized segmentation. In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Papers)*, pages 154–167, Boston, MA. Association for Machine Translation in the Americas.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Patrick Wilken, Tamer Alkhouli, Evgeny Matusov, and Pavel Golik. 2020. Neural simultaneous speech translation using alignment-based chunking. In *Proceedings of the 17th International Conference on Spoken Language Translation*, pages 237–246, Online. Association for Computational Linguistics.

Ruiqing Zhang and Chuanqiang Zhang. 2020. Dynamic sentence boundary detection for simultaneous translation. In *Proceedings of the First Workshop on Automatic Simultaneous Translation*, pages 1–9, Seattle, Washington. Association for Computational Linguistics.

Ruiqing Zhang, Chuanqiang Zhang, Zhongjun He, Hua Wu, and Haifeng Wang. 2020. Learning adaptive segmentation policy for simultaneous translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2280–2289, Online. Association for Computational Linguistics.

Shaolei Zhang and Yang Feng. 2021. ICT's system for AutoSimTrans 2021: Robust char-level simultaneous translation. In *Proceedings of the Second Workshop on Automatic Simultaneous Translation*, pages 1–11, Online. Association for Computational Linguistics.

Shaolei Zhang, Yang Feng, and Liangyou Li. 2021. Future-guided incremental transformer for simultaneous translation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(16):14428–14436.

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

Baigong Zheng, Renjie Zheng, Mingbo Ma, and Liang Huang. 2019a. Simpler and faster learning of adaptive policies 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 1349–1354, Hong Kong, China. Association for Computational Linguistics.

Baigong Zheng, Renjie Zheng, Mingbo Ma, and Liang Huang. 2019b. 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, Kaibo Liu, Jiahong Yuan, Kenneth Church, and Liang Huang. 2020b. Fluent and low-latency simultaneous speech-to-speech translation with self-adaptive training. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3928–3937, Online. Association for Computational Linguistics.