A General Framework for Adaptation of Neural Machine Translation to Simultaneous Translation

Yun Chen[*1], **Liangyou Li**[2], **Xin Jiang**[2], **Xiao Chen**[2], **Qun Liu**[2]

[1]Shanghai University of Finance and Economics, Shanghai, China

[2]Huawei Noah's Ark Lab, Hong Kong, China

yunchen@sufe.edu.cn, {liliangyou, jiang.xin, chen.xiao2, qun.liu}@huawei.com

Abstract

Despite the success of neural machine translation (NMT), simultaneous neural machine translation (SNMT), the task of translating in real time before a full sentence has been observed, remains challenging due to the syntactic structure difference and simultaneity requirements. In this paper, we propose a general framework for adapting neural machine translation to translate simultaneously. Our framework contains two parts: prefix translation that utilizes a consecutive NMT model to translate source prefixes and a stopping criterion that determines when to stop the prefix translation. Experiments on three translation corpora and two language pairs show the efficacy of the proposed framework on balancing the quality and latency in adapting NMT to perform simultaneous translation.

1 Introduction

Simultaneous translation (Fügen et al., 2007; Oda et al., 2014; Grissom et al., 2014; Niehues et al., 2016; Cho and Esipova, 2016; Gu et al., 2017; Ma et al., 2018), the task of producing a partial translation of a sentence before the whole input sentence ends, is useful in many scenarios including outbound tourism, international summit and multilateral negotiations. Different from the consecutive translation in which translation quality alone matters, simultaneous translation trades off between translation quality and latency. The syntactic structure difference between the source and target language makes simultaneous translation more challenging. For example, when translating from a verb-final (SOV) language (e.g., Japanese) to a verb-media (SVO) language (e.g., English), the verb appears much later in the source sequence than in the target language. Some premature translations can lead to significant loss in quality (Ma et al., 2018).

Recently, a number of researchers have endeavored to explore methods for simultaneous translation in the context of NMT (Bahdanau et al., 2015; Vaswani et al., 2017). Some of them propose sophisticated training frameworks explicitly designed for simultaneous translation (Ma et al., 2018; Arivazhagan et al., 2019). These approaches are either memory inefficient during training (Ma et al., 2018) or with hyper-parameters hard to tune (Arivazhagan et al., 2019). Others utilize a full-sentence base model to perform simultaneous translation by modifications to the encoder and the decoding process. To match the incremental source context, they replace the bidirectional encoder with a left-to-right encoder (Cho and Esipova, 2016; Satija and Pineau, 2016; Gu et al., 2017; Alinejad et al., 2018) or recompute the encoder hidden states (Zheng et al., 2019). On top of that, heuristic algorithms (Cho and Esipova, 2016; Dalvi et al., 2018) or a READ/WRITE model trained with reinforcement learning (Satija and Pineau, 2016; Gu et al., 2017; Alinejad et al., 2018) or supervised learning (Zheng et al., 2019) are used to decide, at every step, whether to wait for the next source token or output a target token. However, these models either cannot directly use a pretrained consecutive neural machine translation (CNMT) model with bidirectional encoder as the base model or work in a sub-optimal way in the decoding stage.

In this paper, we study the problem of adapting neural machine translation to translate simultaneously. We formulate simultaneous translation as two nested loops: an outer loop that updates input buffer with newly observed source tokens and an inner loop that translates source tokens in the buffer updated at each outer step. For the outer loop, the input buffer can be updated by an ASR system with

[*] Part of the work was done when Yun is working at Huawei Noah's Ark Lab.

an arbitrary update schedule. For the inner loop, we translate using the pretrained CNMT model and stop translation with a stopping controller. Such formulation is different from previous work (Satija and Pineau, 2016; Gu et al., 2017; Alinejad et al., 2018; Zheng et al., 2019) which define simultaneous translation as sequentially making interleaved READ or WRITE decisions. We argue that our formulation is better than the previous one in two aspects: (i) Our formulation can better utilize the available source tokens. Under previous formulation, the number of source tokens observed by the CNMT model is determined by the number of READ actions that has been produced by the policy network. It is likely that the CNMT model does not observe all the available source tokens produced by the ASR system. In contrast, the CNMT model observes all the available source tokens when performing inner loop translation in our framework. (ii) Previous formulation makes T_η+T_τ READ or WRITE decisions regardless of the ASR update schedule, where T_η and T_τ are source sentence and translation length, respectively. For an ASR system that outputs multiple tokens at a time, this is computational costly. Consider an extreme case where the ASR system outputs a full source sentence at a time. Previous work translates with a sequence of T_η+T_τ actions, while we translate with a sequence of T_τ decisions ($T_\tau - 1$ CONTINUE and 1 STOP).

Under our proposed framework, we present two schedules for simultaneous translation: one stops the inner loop translation with heuristic and one with a stopping controller learned in a reinforcement learning framework to balance translation quality and latency. We evaluate our method on IWSLT16 German-English (DE-EN) translation in both directions, WMT15 English-German (EN-DE) translation in both directions, and NIST Chinese-to-English (ZH→EN) translation. The results show our method with reinforced stopping controller consistently improves over the de-facto baselines, and achieves low latency and reasonable BLEU scores.

2 Background

Given a set of source–target sentence pairs $\langle \mathbf{x}_m, \mathbf{y}^*_m \rangle_{m=1}^{M}$, a consecutive NMT model can be trained by maximizing the log-likelihood of the target sentence from its entire source side context:

$$\hat{\phi} = \underset{\phi}{\text{argmax}} \left\{ \sum_{m=1}^{M} \log p(\mathbf{y}^*_m | \mathbf{x}_m; \phi) \right\}, \qquad (1)$$

where ϕ is a set of model parameters. At inference time, the NMT model first encodes a source language sentence $\mathbf{x} = \{x_1, ..., x_{T_\eta}\}$ with its encoder and passes the encoded representations $\mathbf{h} = \{h_1, ..., h_{T_\eta}\}$ to a greedy decoder. Then the greedy decoder generates a translated sentence in target language by sequentially choosing the most likely token at each step t:

$$y_t = \text{argmax}_y \, p(y|y_{<t}, \mathbf{x}). \qquad (2)$$

The distribution of next target word is defined as:

$$p(y|y_{<t}, \mathbf{x}) \propto \exp\left[\phi_{\text{OUT}}(z_t)\right]$$
$$z_t = \phi_{\text{DEC}}(y_{t-1}, z_{<t}, \mathbf{h}), \qquad (3)$$

where z_t is the decoder hidden state at position t. In consecutive NMT, once obtained, the encoder hidden states $\mathbf{h}$ and the decoder hidden state z_t are not updated anymore and will be reused during the entire decoding process.

3 Simultaneous NMT

In SNMT, we receive streaming input tokens, and learn to translate them in real-time. We formulate simultaneous translation as two nested loops: the outer loop that updates an input buffer with newly observed source tokens and the inner loop that translates source tokens in the buffer updated at each outer step.

More precisely, suppose at the end of outer step $s-1$, the input buffer is $\mathbf{x}^{s-1} = \{x_1, ..., x_{\eta[s-1]}\}$, and the output buffer is $\mathbf{y}^{s-1} = \{y_1, ..., y_{\tau[s-1]}\}$. Then at outer step s, the system translates with the following steps:

1 The system observes $c_s > 0$ new source tokens and updates the input buffer to be $\mathbf{x}^s = \{x_1, ..., x_{\eta[s]}\}$ where $\eta[s] = \eta[s-1] + c_s$.

2 Then, the system starts inner loop translation and writes $w_s >= 0$ target tokens to the output buffer. The output buffer is updated to be $\mathbf{y}^s = \{y_1, ..., y_{\tau[s]}\}$ where $\tau[s] = \tau[s-1] + w_s$.

The simultaneous decoding process continues until no more source tokens are added in the outer loop. We define the last outer step as the terminal outer step S, and other outer steps as non-terminal outer steps.

For the outer loop, we make no assumption about the value of c_s, while all previous work assumes

$c_s = 1$. This setting is more realistic because (i) increasing c_s can reduce the number of outer steps, thus reducing computation cost; (ii) in a real speech translation application, an ASR system may generate multiple tokens at a time.

For the inner loop, we adapt a pretrained vanilla CNMT model to perform partial translation with two important concerns:

1. Prefix translation: given a source prefix $\mathbf{x}^s = \{x_1, ..., x_{\eta[s]}\}$ and a target prefix $\mathbf{y}^s_{\tau[s-1]} = \{y_1, ..., y_{\tau[s-1]}\}$, how to predict the remaining target tokens?

2. Stopping criterion: since the NMT model is trained with full sentences, how to design the stopping criterion for it when translating partial source sentcnes?

3.1 Prefix Translation

At an outer step s, given encoder hidden states $\mathbf{h}^s$ for source prefix $\mathbf{x}^s = \{x_1, ..., x_{\eta[s]}\}$ and decoder hidden states $\mathbf{z}^s_{\tau[s-1]}$ for target prefix $\mathbf{y}^s_{\tau[s-1]} = \{y_1, ..., y_{\tau[s-1]}\}$, we perform prefix translation sequentially with a greedy decoder:

$$\begin{aligned} z^s_t &= \phi_{\text{DEC}}\left(y_{t-1}, z^s_{<t}, \mathbf{h}^s\right) \\ p(y|y_{<t}, \mathbf{x}^s) &\propto \exp\left[\phi_{\text{OUT}}\left(z^s_t\right)\right] \\ y_t &= \text{argmax}_y\, p(y|y_{<t}, \mathbf{x}^s), \end{aligned} \tag{4}$$

where t starts from $t = \tau[s-1] + 1$. The prefix translation terminates when a stopping criterion meets, yielding a translation $\mathbf{y}^s = \{y_1, ..., y_{\tau[s]}\}$.

However, a major problem comes from the above translation method: how can we obtain the encoder hidden states $\mathbf{h}^s$ and decoder hidden states $\mathbf{z}^s_{\tau[s-1]}$ at the beginning of prefix translation? We propose to rebuild all encoder and decoder hidden states with

$$\mathbf{h}^s = \phi_{\text{ENC}}\big(\mathbf{x}^s\big), \tag{5}$$

$$\mathbf{z}^s_{\tau[s-1]} = \phi_{\text{DEC}}\big(\mathbf{y}^s_{\tau[s-1]}, \mathbf{h}^s\big). \tag{6}$$

During full sentence training, all the decoder hidden states are computed conditional on the same source tokens. By rebuilding encoder and decoder hidden states, we also ensure that the decoder hidden states are computed conditional on the same source. This strategy is different from previous work that reuse previous encoder (Cho and Esipova, 2016; Gu et al., 2017; Dalvi et al., 2018; Alinejad et al., 2018) or decoder (Cho and Esipova, 2016; Gu et al., 2017; Dalvi et al., 2018; Ma et al., 2018) hidden states. We carefully compare the effect of rebuilding hidden states in Section 4.2 and experiment results show that rebuilding all hidden states benefits translation.

	src		trans
1	晓莹	→	xiaoying
2	晓莹 你	→	xiaoying you
3	晓莹 你 好	→	xiaoying you are good
4	晓莹 你 好 。	→	xiaoying you are good .

Figure 1: Failure case when using EOS alone as the stopping criterion.

3.2 Stopping Criterion

In consecutive NMT, the decoding algorithm such as greedy decoding or beam search terminates when the translator predicts an EOS token or the length of the translation meets a predefined threshold (e.g. 200). The decoding for most source sentences terminates when the translator predicts the EOS token.[1] In simultaneous decoding, since we use a NMT model pretrained on full sentences to translate partial source sentences, it tends to predict EOS when the source context has been fully translated. However, such strategy could be too aggressive for simultaneous translation. Fig. 1 shows such an example. At outer step 2, the translator predicts “you EOS”, emiting target token “you”. However, “you” is not the expected translation for “你” in the context of “你好。”. Therefore, we hope prefix translation at outer step 2 can terminate without emitting any words.

To alleviate such problems and do better simultaneous translation with pretrained CNMT model, we propose two novel stopping criteria for prefix translation.

3.2.1 Length and EOS Control

In consecutive translation, the decoding process stops mainly when predicting EOS. In contrast, for prefix translation at non-terminal outer step, we stop the translation process when translation length is d tokens behind source sentence length: $\tau[s] = \eta[s] - d$. Specifically, at the beginning of outer step s, we have source prefix $\mathbf{x}^s = \{x_1, ..., x_{\eta[s]}\}$ and target prefix $\mathbf{y}^s_{\tau[s-1]} = \{y_1, ..., y_{\tau[s-1]}\}$. Prefix translation terminates at inner step w_s when

[1]We conduct greedy decoding on the validation set of WMT15 EN→DE translation with fairseq-py, and find that 100% translation terminates with EOS predicted.

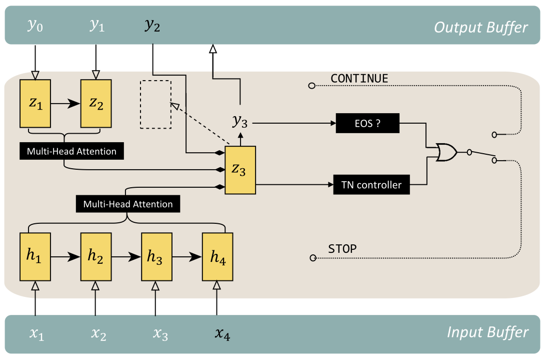

Figure 2: Framework of our proposed model with the TN controller.

predicting an EOS token or satisfying:

$$w_s = \begin{cases} \max(0, \eta[s] - \tau[s-1] - d) & s < S \\ 200 - \tau[s-1] & s = S \end{cases} \tag{7}$$

where d is a non-negative integer that determines the translation latency of the system. We call this stopping criterion as Length and EOS (LE) stopping controller.

3.2.2 Learning When to Stop

Although simple and easy to implement, LE controller lacks the capability to learn the optimal timing with which to stop prefix translation. Therefore, we design a small trainable network called trainable (TN) stopping controller to learn when to stop prefix translation for non-terminal outer step. Fig. 2 shows the illustration.

At each inner decoding step k for non-terminal outer step s, the TN controller utilizes a stochastic policy π_θ parameterized by a neural network to make the binary decision on whether to stop translation at current step:

$$\pi_\theta(a_{\tau[s-1]+k} | z^s_{\tau[s-1]+k}) = f_\theta(z^s_{\tau[s-1]+k}), \tag{8}$$

where $z^s_{\tau[s-1]+k}$ is the current decoder hidden state. We implement f_θ with a feedforward network with two hidden layers, followed by a softmax layer. The prefix translation stops if the TN controller predicts $a_{\tau[s-1]+k} = 1$. Our TN controller is much simpler than previous work (Gu et al., 2017) which implements the READ/WRITE policy network using a recurrent neural network whose input is the combination of the current context vector, the current decoder state and the embedding vector of the candidate word.

To train the TN controller, we freeze the NMT model with pretrained parameters, and optimize the TN network with policy gradient for reward maximization $\mathcal{J} = \mathbb{E}_{\pi_\theta}(\sum_{t=1}^{T_\tau} r_t)$. With a trained TN controller, prefix translation stops at inner decoding step w_s when predicting an EOS token or satisfying:

$$\begin{cases} a_{\tau[s-1]+w_s} = 1 & s < S \\ w_s = 200 - \tau[s-1] & s \leq S \end{cases}. \tag{9}$$

In the following, we talk about the details of the reward function and the training with policy gradient.

Reward To trade-off between translation quality and latency, we define the reward function at inner decoding step k of outer step s as:

$$r_t = r_t^Q + \alpha \cdot r_t^D, \tag{10}$$

where $t = \tau[s-1]+k$, and r_t^Q and r_t^D are rewards related to quality and delay, respectively. $\alpha \geq 0$ is a hyper-parameter that we adjust to balance the trade-off between translation quality and delay.

Similar to Gu et al. (2017), we utilize sentence-level BLEU (Papineni et al., 2002; Lin and Och, 2004) with reward shaping (Ng et al., 1999) as the reward for quality:

$$r_t^Q = \begin{cases} \Delta\text{BLEU}(\mathbf{y}^*, \mathbf{y}, t) & k \neq w_s \; or \; s \neq S \\ \text{BLEU}(\mathbf{y}^*, \mathbf{y}) & k = w_s \; and \; s = S \end{cases} \tag{11}$$

where

$$\begin{aligned} &\Delta\text{BLEU}(\mathbf{y}^*, \mathbf{y}, t) \\ &= \text{BLEU}(\mathbf{y}^*, \mathbf{y}_t) - \text{BLEU}(\mathbf{y}^*, \mathbf{y}_{t-1}) \end{aligned} \tag{12}$$

is the intermediate reward. Note that the higher the values of BLEU are, the more rewards the TN controller receives. Following Ma et al. (2018), we use average lagging (AL) as the reward for latency:

$$r_t^D = \begin{cases} 0 & k \neq w_s \; or \; s \neq S \\ -\lfloor d(\mathbf{x}, \mathbf{y}) - d^* \rfloor_+ & k = w_s \; and \; s = S \end{cases} \tag{13}$$

where

$$d(\mathbf{x}, \mathbf{y}) = \frac{1}{t_e} \sum_{t=1}^{\tau_e} l(t) - \frac{t-1}{\lambda}. \tag{14}$$

$l(t)$ is the number of observed source tokens when generating the t-th target token, t_e =

Dataset	Train	Validation	Test
IWSLT16	193,591	993	1,305
WMT15	3,745,796	3,003	2,169
NIST	1,252,977	878	4,103

Table 1: # sentences in each dataset.

$\mathrm{argmin}_t\,(l(t) = |\mathbf{x}|)$ denotes the earliest point when the system observes the full source sentence, $\lambda = \frac{|\mathbf{y}|}{|\mathbf{x}|}$ represents the target-to-source length ratio and $d^* \geq 0$ is a hyper-parameter called target delay that indicates the desired system latency. Note that the lower the values of AL are, the more rewards the TN controller receives.

Policy Gradient We train the TN controller with policy gradient(Sutton et al., 1999), and the gradients are:

$$\nabla_\theta \mathcal{J} = \mathbb{E}_{\pi_\theta}\left[\sum_{t=1}^{T_\tau} R_t \nabla_\theta \log \pi_\theta(a_t|\cdot)\right], \qquad (15)$$

where $R_t = \sum_{i=t}^{T_\tau} r_i$ is the cumulative future rewards for the current decision. We can adopt any sampling approach (Chen et al., 2017, 2018; Shen et al., 2018) to estimate the expected gradient. In our experiments, we randomly sample multiple action trajectories from the current policy π_θ and estimate the gradient with the collected accumulated reward. We try the variance reduction techniques by subtracting a baseline average reward estimated by a linear regression model from R_t and find that it does not help to improve the performance. Therefore, we just normalize the reward in each mini-batch without using baseline reward for simplicity.

4 Experiments

4.1 Settings

Dataset We compare our approach with the baselines on WMT15 German-English[2] (DE-EN) translation in both directions. This is also the most widely used dataset to evaluate SNMT's performance (Cho and Esipova, 2016; Gu et al., 2017; Ma et al., 2018; Arivazhagan et al., 2019; Zheng et al., 2019). To further evaluate our approach's efficacy in trading off translation quality and latency on other language pair and spoken language, we also conduct experiments with the proposed LE and TN methods on NIST Chinese-to-English[3] (ZH→EN) translation and IWSLT16 German-English[4] (DE-EN) translation in both directions. For WMT15, we use newstest2014 for validation and newstest2015 for test. For NIST, we use MT02 for validation, and MT05, MT06, MT08 for test. For IWSLT16, we use tst13 for validation and tst14 for test. All the data is tokenized and segmented into subword symbols using byte-pair encoding (Sennrich et al., 2016) to restrict the size of the vocabulary. We use 40,000 joint merge operations on WMT15, and 24,000 on IWSLT16. For NIST, we use 30,000 merge operations for source and target side separately. Without explicitly mention, we simulate simultaneous translation scenario at inference time with these datasets by assuming that the system observes one new source token at each outer step, i.e., $c_s = 1$. Table 1 shows the data statistics.

Pretrained NMT Model We use Transformer (Vaswani et al., 2017) trained with maximum likelihood estimation as the pretrained CNMT model and implement our method based on fairseq-py.[5] We follow the setting in `transformer_iwslt_de_en` for IWSLT16 dataset, and `transformer_wmt_en_de` for WMT15 and NIST dataset. Fairseq-py adds an EOS token for all source sentences during training and inference. Therefore, to be consistent with the CNMT model implemented with fairseq-py, we also add an EOS token at the end of the source prefix for prefix translation and find that the EOS helps translation.

TN Controller To train the TN controller, we use a mini-batch size of 8,16,16 and sample 5,10,10 trajectories for each sentence pair in a batch for IWSLT16, WMT15 and NIST, respectively. We set the number of newly observed source tokens at each outer step to be 1 during the training for simplicity. We set α to be 0.04, and d^* to be $2, 5, 8$. All our TN controllers are trained with policy gradient using Adam optimizer (Kingma and Ba, 2015) with 30,000 updates. We select the last model as our final TN controller.

Baseline We compare our model against three baselines that utilize a pretrained CNMT model to

[2]http://www.statmt.org/wmt15/

[3]These sentence pairs are mainly extracted from LDC2002E18, LDC2003E07, LDC2003E14, Hansards portion of LDC2004T07, LDC2004T08 and LDC2005T06

[4]https://workshop2016.iwslt.org/

[5]https://github.com/pytorch/fairseq

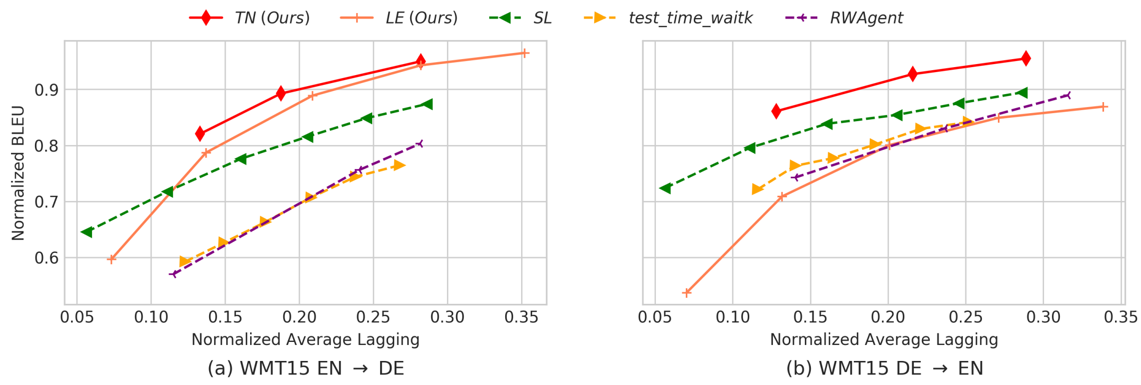

Figure 3: Comparison with the baselines on the test set of WMT15 EN→DE and WMT15 DE→EN translations. The shown points from left to right on the same line are the results of simultaneous greedy decoding with $d^* \in \{2, 5, 8\}$ for TN, $d \in \{0, 2, 4, 6, 8\}$ for LE, $\rho \in \{0.65, 0.6, 0.55, 0.5, 0.45, 0.4\}$ for SL, $k \in \{1, 3, 5, 7, 9\}$ for test_time_waitk and $CW \in \{2, 5, 8\}$ for RWAgent. The scores of **Greedy** decoding: BLEU=25.16, AL=28.10 for WMT15 EN→DE translation and BLEU=26.17, AL=31.20 for WMT15 DE→EN translation.

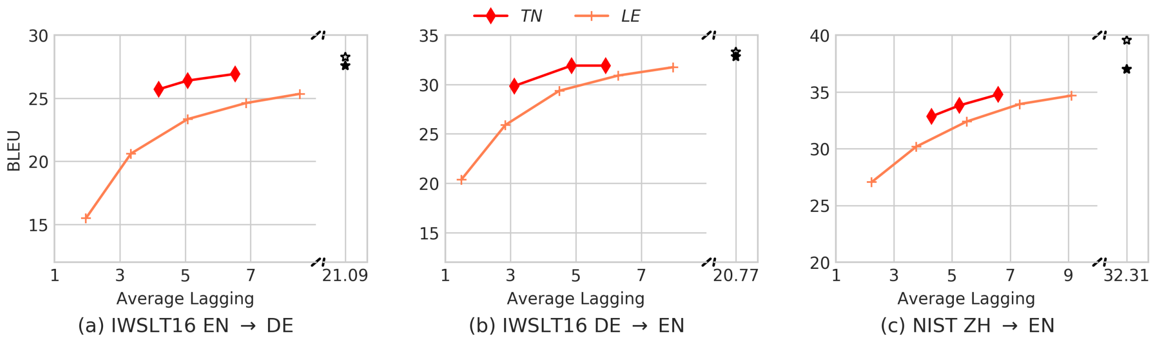

Figure 4: Performance on the test set of IWSLT16 EN→DE translation, IWSLT16 DE→EN translation and NIST ZH→EN translation. The shown points from left to right on the same line are the results of $d^* \in \{2, 5, 8\}$ for TN and $d \in \{0, 2, 4, 6, 7\}$ for LE. ★☆:full-sentence (greedy and beam-search).

perform simultaneous translation:

- **test_time_waitk** (Ma et al., 2018): the method that decodes with a waitk policy with a CNMT model. We report the results when $k \in \{1, 3, 5, 7, 9\}$.
- **SL** (Zheng et al., 2019): the method that adapts CNMT to SNMNT by learning an adaptive READ/WRITE policy from oracle READ/WRITE sequences generated with heuristics. We report the results with threshold $\rho \in \{0.65, 0.6, 0.55, 0.5, 0.45, 0.4\}$.
- **RWAgent** (Gu et al., 2017): the adaptation of Gu et al. (2017)'s full-sentence model and reinforced READ/WRITE policy network to Transformer by Ma et al. (2018). We report the results when using $CW \in \{2, 5, 8\}$ as the target delay.

We report the result with $d \in \{0, 2, 4, 6, 8\}$ for our proposed LE method and $d^* \in \{2, 5, 8\}$ for our proposed TN method. For all baselines, we cite the results reported in Zheng et al. (2019). [6]

4.2 Results

We compare our methods with the baselines on the test set of WMT15 EN→DE and DE→EN translation tasks, as shown in Fig. 3. The points closer to the upper left corner indicate better overall performance, namely low latency and high quality. We observe that as latency increases, all methods improve in quality. the TN method significantly outperforms all the baselines in both translation tasks,

[6]Since Zheng et al. (2019) did not mention the details of data preprocessing, we cannot compare the BLEU and AL scores directly with theirs. Therefore, we normalize the BLEU and AL scores with its corresponding upper bound, i.e. the BLEU and AL scores obtained when the pretrained Transformer performs standard greedy decoding (**Greedy**).

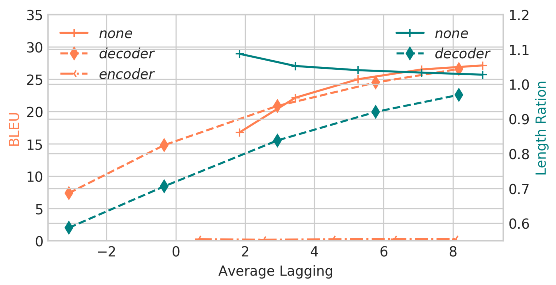

Figure 5: Comparison of whether to reuse previous encoder or decoder hidden states on WMT15 EN→DE test set with the LE controller. The left Y axis is the BLEU score and the right Y axis is the length ratio: the translation length divided by the reference length. The points on the same line are the results of $d \in \{0, 2, 4, 6, 8\}$. *none*: rebuild all encoder/decoder hidden states; *decoder*: reuse decoder hidden states and rebuild all encoder hidden states; *encoder*: reuse previous encoder hidden states and rebuild all decoder hidden states.

demonstrating that it indeed learns the appropriate timing to stop prefix translation. LE outperforms the baselines on WMT15 EN→DE translation at high latency region and performs similarly or worse on other cases.

We show the methods' efficacy in trading off quality and latency on other language pair and spoken language in Fig. 4. TN outperforms LE on all translation tasks, especially at the low latency region. It obtains promising translation quality with acceptable latency: with a lag of < 7 tokens, TN obtains 96.95%, 97.20% and 94.03% BLEU with respect to consecutive greedy decoding for IWSLT16 EN→DE, IWSLT16 DE→EN and NIST ZH→EN translations, respectively.

4.3 Analyze

We analyze the effect of different ways to obtain the encoder and decoder hidden states at the beginning of prefix translation with the LE controller. Fig. 5 shows the result. We try three variants: a) dynamically rebuild all encoder/decoder hidden states (*none*); b) reuse decoder hidden states and rebuild all encoder hidden states (*decoder*); c) reuse previous encoder hidden states and rebuild all decoder hidden states (*encoder*). The left Y axis and X axis show BLEU-vs-AL curve. We observe that if reusing previous encoder hidden states (*encoder*), the translation fails. We ascribe this to the discrepancy between training and decoding for the encoder. We also observe that when $d \in 0, 2$, reusing decoder hidden states (*decoder*) obtain negative AL. To analyze this, we plot the translation to reference length ratio versus AL curve with the right Y axis and X axis. It shows that with *decoder*, the decoding process stops too early and generates too short translations. Therefore, to avoid such problem and to be consistent with the training process of the CNMT model, it is important to dynamically rebuild all encoder/decoder hidden states for prefix translation.

Since we make no assumption about the c_s, i.e., the number of newly observed source tokens at each outer step, we also test the effect of different c_s. Fig. 6 shows the result with the LE and TN controllers on the test set of WMT15 EN→DE translation. We observe that as c_s increases, both LE and TN trend to improve in quality and worsen in latency. When $c_s = 1$, LE controller obtains the best balance between quality and latency. In contrast, TN controller obtains similar quality and latency balance with different c_s, demonstrating that TN controller successfully learns the right timing to stop regardless of the input update schedule.

We also analyze the TN controller's adaptability by monitoring the initial delay, i.e., the number of observed source tokens before emitting the first target token, on the test set of WMT15 EN→DE translation, as shown in Fig. 7. d^* is the target delay measured with AL (used in Eq. 13). It demonstrates that the TN controller has a lot of variance in it's initial delay. The distribution of initial delay changes with different target delay: with higher target delay, the average initial delay is larger. For most sentences, the initial delay is within $1 - 7$.

In speech translation, listeners are also concerned with long silences during which no translation occurs. Following Gu et al. (2017); Ma et al. (2018), we use Consecutive Wait (CW) to measure this:

$$CW(\mathbf{x}, \mathbf{y}) = \frac{\sum_{s=1}^{S} c_s}{\sum_{s=1}^{S} \mathbb{1}_{w_s > 0}}. \quad (16)$$

Fig. 8 shows the BLEU-vs-CW plots for our proposed two methods. The TN controller has higher CW than the LE controller. This is because TN controller prefers consecutive updating output buffer (e.g., it often produces w_s as 0 0 0 0 3 0 0 0 0 0 5 0 0 0 0 4 ...) while the LE controller often updates its output buffer following the input buffer (e.g., it often produces w_s as 0 0 0 0 1 1 1 1 1 1 ... when $d = 4$). Although larger than LE, the CW for TN (< 6) is acceptable for most speech translation scenarios.

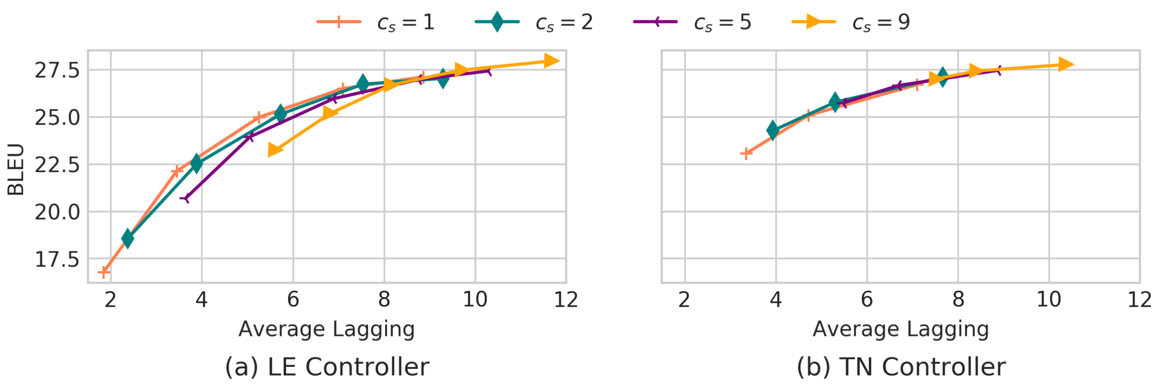

Figure 6: Performance on the test set of WMT15 EN→DE translation with different input buffer update schedule. Points on the same line are obtained by increasing $d \in 0, 2, 4, 6, 8$ for (a) and $d^* \in 2, 5, 8$ for (b).

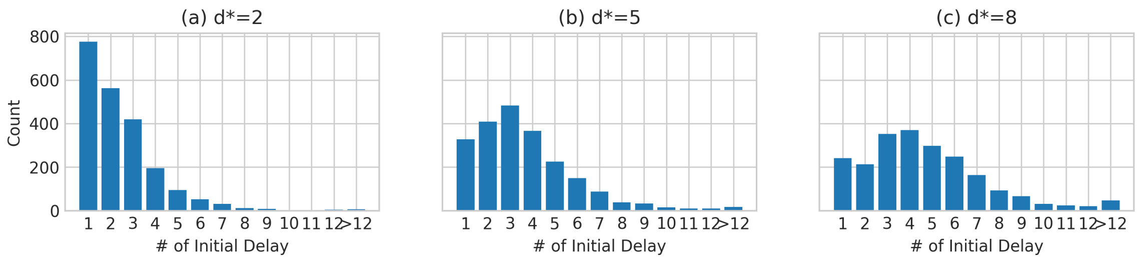

Figure 7: Number of observed source tokens before emitting the first target token for the TN controller on the test set of WMT15 EN→DE translation.

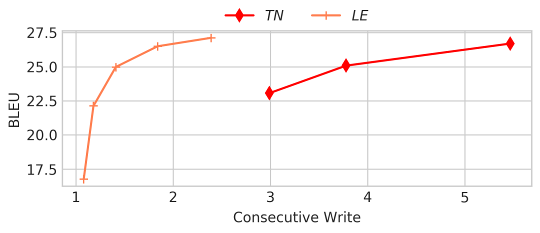

Figure 8: Average consecutive write length on the test set of WMT15 EN→DE translation.

4.4 Translation Examples

Fig. 9 shows two translation examples with the LE and TN controllers on the test set of NIST ZH→EN and WMT15 EN→DE translation. In manual inspection of these examples and others, we find that the TN controller learns a conservative timing for stopping prefix translation. For example, in example 1, TN outputs translation *"wu bangguo attended the signing ceremony"* when observing "吴邦国 出席 签字 仪式 并", instead of a more radical translation *"wu bangguo attended the signing ceremony and"*. Such strategy helps to alleviate the problem of premature translation, i.e., translating before observing enough future context.

5 Related Work

A number of works in simultaneous translation divide the translation process into two stages. A segmentation component first divides the incoming text into segments, and then each segment is translated by a translator independently or with previous context. The segmentation boundaries can be predicted by prosodic pauses detected in speech (Fügen et al., 2007; Bangalore et al., 2012), linguistic cues (Sridhar et al., 2013; Matusov et al., 2007), or a classifier based on alignment information (Siahbani et al., 2014; Yarmohammadi et al., 2013) and translation accuracy (Oda et al., 2014; Grissom et al., 2014; Siahbani et al., 2018).

Some authors have recently endeavored to perform simultaneous translation in the context of NMT. Niehues et al. (2018); Arivazhagan et al. (2020) adopt a re-translation approach where the source is repeatedly translated from scratch as it grows and propose methods to improve translation stability. Cho and Esipova (2016); Dalvi et al. (2018); Ma et al. (2018) introduce a manually designed criterion to control when to translate. Satija and Pineau (2016); Gu et al. (2017); Alinejad et al. (2018) extend the criterion into a trainable agent

	1	2	3	4	5	6	7	8	9
	吴邦国	出席	签字	仪式	并	在	协议	上	签字
LE					wu	bangguo	attended	the	signing ceremony and signed the agreement
TN					wu bangguo attended the signing ceremony				and signed the agreement
Greedy									wu bangguo attended the signing ceremony and signed the agreement
Ref									wu bangguo attends signing ceremony and signs agreement
	NATO	does	not	want	to	break	agreements	with	Russia
LE					Die	NATO	möchte	keine	Abkommen mit Russland brechen
TN		Die NATO			will			keine Abkommen	mit Russland brechen
Greedy									Die NATO möchte keine Abkommen mit Russland brechen
Ref									NATO will Vereinbarungen mit Russland nicht brechen

Figure 9: Translation examples from the test set of NIST ZH→EN (example 1) and WMT15 EN→DE translation (example 2). We compare LE with $d = 4$ and TN with $d^* = 5$ because these two models achieve similar latency. Greedy and Ref represent the greedy decoding result from consecutive translation and the reference, respectively.

in a reinforcement learning framework. However, these work either develop sophisticated training frameworks explicitly designed for simultaneous translation (Ma et al., 2018) or fail to use a pretrained consecutive NMT model in an optimal way (Cho and Esipova, 2016; Dalvi et al., 2018; Satija and Pineau, 2016; Gu et al., 2017; Alinejad et al., 2018; Zheng et al., 2019). In contrast, our work is significantly different from theirs in the way of using pretrained consecutive NMT model to perform simultaneous translation and the design of the two stopping criteria.

6 Conclusion

We have presented a novel framework for improving simultaneous translation with a pretrained consecutive NMT model. The basic idea is to translate partial source sentence with the consecutive NMT model and stops the translation with two novel stopping criteria. Extensive experiments demonstrate that our method with trainable stopping controller outperforms the state-of-the-art baselines in balancing between translation quality and latency.

Acknowledgments

We thank the anonymous reviewers for their insightful feedback on this work. Yun Chen is partially supported by the Fundamental Research Funds for the Central Universities and the funds of Beijing Advanced Innovation Center for Language Resources (No. TYZ19005).

References

Ashkan Alinejad, Maryam Siahbani, and Anoop Sarkar. 2018. Prediction improves simultaneous neural machine translation. In *EMNLP*.

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. *ArXiv*, abs/1906.05218.

Naveen Arivazhagan, Colin Cherry, Isabelle Te, Wolfgang Macherey, Pallavi Baljekar, and George Foster. 2020. Re-translation strategies for long form, simultaneous, spoken language translation. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7919–7923. IEEE.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *In Proceedings of ICLR*.

Srinivas Bangalore, Vivek Kumar Rangarajan Sridhar, Prakash Kolan, Ladan Golipour, and Aura Jimenez. 2012. Real-time incremental speech-to-speech translation of dialogs. In *HLT-NAACL*.

Yun Chen, Yang Liu, Yong Cheng, and Victor OK Li. 2017. A teacher-student framework for zero-resource neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1925–1935.

Yun Chen, Yang Liu, and Victor OK Li. 2018. Zero-resource neural machine translation with multi-agent communication game. In *AAAI*.

Kyunghyun Cho and Masha Esipova. 2016. Can neural machine translation do simultaneous translation. *CoRR*, abs/1606.02012.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, and Stephan Vogel. 2018. Incremental decoding and training methods for simultaneous translation in neural machine translation. In *NAACL-HLT*.

Christian Fügen, Alexander H. Waibel, and Muntsin Kolss. 2007. Simultaneous translation of lectures and speeches. *Machine Translation*, 21:209–252.

Alvin Grissom, He He, Jordan L. Boyd-Graber, John Morgan, and Hal Daumé. 2014. Don't until the final verb wait: Reinforcement learning for simultaneous machine translation. In *EMNLP*.

Jiatao Gu, Graham Neubig, Kyunghyun Cho, and Victor O. K. Li. 2017. Learning to translate in real-time with neural machine translation. In *EACL*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Chin-Yew Lin and Franz Josef Och. 2004. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *ACL*.

Mingbo Ma, Liang Huang, Hao Xiong, Kaibo Liu, Chuanqiang Zhang, Zhongjun He, Hairong Liu, Xing Li, and Haifeng Wang. 2018. Stacl: Simultaneous translation with integrated anticipation and controllable latency. *CoRR*, abs/1810.08398.

Evgeny Matusov, Dustin Hillard, Mathew Magimai-Doss, Dilek Z. Hakkani-Tür, Mari Ostendorf, and Hermann Ney. 2007. Improving speech translation with automatic boundary prediction. In *INTERSPEECH*.

Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*.

Jan Niehues, Thai Son Nguyen, Eunah Cho, Thanh-Le Ha, Kevin Kilgour, Markus Müller, Matthias Sperber, Sebastian Stüker, and Alex Waibel. 2016. Dynamic transcription for low-latency speech translation. In *Interspeech*, pages 2513–2517.

Jan Niehues, Ngoc-Quan Pham, Thanh-Le Ha, Matthias Sperber, and Alex Waibel. 2018. Low-latency neural speech translation. *arXiv preprint arXiv:1808.00491*.

Yusuke Oda, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2014. Optimizing segmentation strategies for simultaneous speech translation. In *ACL*.

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

Harsh Satija and Joelle Pineau. 2016. Simultaneous machine translation using deep reinforcement learning. *Abstraction in Reinforcement Learning Workshop, ICML*.

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.

Shi-qi Shen, Yun Chen, Cheng Yang, Zhi-yuan Liu, Mao-song Sun, et al. 2018. Zero-shot cross-lingual neural headline generation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(12):2319–2327.

Maryam Siahbani, Ramtin Mehdizadeh Seraj, Baskaran Sankaran, and Anoop Sarkar. 2014. Incremental translation using hierarchichal phrase-based translation system. *2014 IEEE Spoken Language Technology Workshop (SLT)*, pages 71–76.

Maryam Siahbani, Hassan Shavarani, Ashkan Alinejad, and Anoop Sarkar. 2018. Simultaneous translation using optimized segmentation. In *AMTA*.

Vivek Kumar Rangarajan Sridhar, John Chen, Srinivas Bangalore, Andrej Ljolje, and Rathinavelu Chengalvarayan. 2013. Segmentation strategies for streaming speech translation. In *HLT-NAACL*.

Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*.

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 *NIPS*.

Mahsa Yarmohammadi, Vivek Kumar Rangarajan Sridhar, Srinivas Bangalore, and Baskaran Sankaran. 2013. Incremental segmentation and decoding strategies for simultaneous translation. In *IJCNLP*.

Baigong Zheng, Renjie Zheng, Mingbo Ma, and Liang Huang. 2019. Simpler and faster learning of adaptive policies for simultaneous translation. In *EMNLP*, pages 1349–1354, Hong Kong, China. Association for Computational Linguistics.