# Learning to Translate in Real-time with Neural Machine Translation

**Jiatao Gu**[†], **Graham Neubig**[◇], **Kyunghyun Cho**[‡] **and Victor O.K. Li**[†]

[†]The University of Hong Kong [◇]Carnegie Mellon University [‡]New York University
[†]{jiataogu, vli}@eee.hku.hk [◇]gneubig@cs.cmu.edu
[‡]kyunghyun.cho@nyu.edu

## Abstract

Translating in real-time, a.k.a. simultaneous translation, outputs translation words before the input sentence ends, which is a challenging problem for conventional machine translation methods. We propose a neural machine translation (NMT) framework for simultaneous translation in which an agent learns to make decisions on when to translate from the interaction with a pre-trained NMT environment. To trade off quality and delay, we extensively explore various targets for delay and design a method for beam-search applicable in the simultaneous MT setting. Experiments against state-of-the-art baselines on two language pairs demonstrate the efficacy of the proposed framework both quantitatively and qualitatively.[1]

## 1 Introduction

Simultaneous translation, the task of translating content in real-time as it is produced, is an important tool for real-time understanding of spoken lectures or conversations (Fügen et al., 2007; Bangalore et al., 2012). Different from the typical machine translation (MT) task, in which translation quality is paramount, simultaneous translation requires balancing the trade-off between translation quality and time delay to ensure that users receive translated content in an expeditious manner (Mieno et al., 2015). A number of methods have been proposed to solve this problem, mostly in the context of phrase-based machine translation. These methods are based on a *segmenter*, which receives the input one word at a time, then decides when to send it to a MT system that translates each segment independently (Oda et al., 2014) or with a minimal amount of language model context (Bangalore et al., 2012).

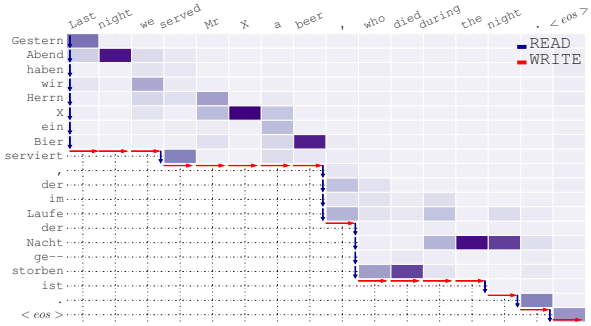



Figure 1: Example output from the proposed framework in DE → EN simultaneous translation. The heat-map represents the soft alignment between the incoming source sentence (left, up-to-down) and the emitted translation (top, left-to-right). The length of each column represents the number of source words being waited for before emitting the translation. Best viewed when zoomed digitally.

Independently of simultaneous translation, accuracy of standard MT systems has greatly improved with the introduction of neural-network-based MT systems (NMT) (Sutskever et al., 2014; Bahdanau et al., 2014). Very recently, there have been a few efforts to apply NMT to simultaneous translation either through heuristic modifications to the decoding process (Cho and Esipova, 2016), or through the training of an independent segmentation network that chooses when to perform output using a standard NMT model (Satija and Pineau, 2016). However, the former model lacks a capability to learn the appropriate timing with which to perform translation, and the latter model uses a standard NMT model as-is, lacking a holistic design of the modeling and learning within the simultaneous MT context. In addition, neither model has demonstrated gains over previ-

[1]Code and data can be found at https://github.com/nyu-dl/dl4mt-simul-trans.

ous segmentation-based baselines, leaving questions of their relative merit unresolved.

In this paper, we propose a unified design for learning to perform neural simultaneous machine translation. The proposed framework is based on formulating translation as an interleaved sequence of two actions: READ and WRITE. Based on this, we devise a model connecting the NMT system and these READ/WRITE decisions. An example of how translation is performed in this framework is shown in Fig. 1, and detailed definitions of the problem and proposed framework are described in §2 and §3. To learn which actions to take when, we propose a reinforcement-learning-based strategy with a reward function that considers both quality and delay (§4). We also develop a beam-search method that performs search within the translation segments (§5).

We evaluate the proposed method on English-Russian (EN-RU) and English-German (EN-DE) translation in both directions (§6). The quantitative results show strong improvements compared to both the NMT-based algorithm and a conventional segmentation methods. We also extensively analyze the effectiveness of the learning algorithm and the influence of the trade-off in the optimization criterion, by varying a target delay. Finally, qualitative visualization is utilized to discuss the potential and limitations of the framework.

## 2 Problem Definition

Suppose we have a buffer of input words $X = \{x_1, ..., x_{T_s}\}$ to be translated in real-time. We define the simultaneous translation task as sequentially making two interleaved decisions: READ or WRITE. More precisely, the translator READs a source word $x_\eta$ from the input buffer in chronological order as translation context, or WRITEs a translated word $y_\tau$ onto the output buffer, resulting in output sentence $Y = \{y_1, ..., y_{T_t}\}$, and action sequence $A = \{a_1, ..., a_T\}$ consists of $T_s$ READs and $T_t$ WRITEs, so $T = T_s + T_t$.

Similar to standard MT, we have a measure $Q(Y)$ to evaluate the translation quality, such as BLEU score (Papineni et al., 2002). For simultaneous translation we are also concerned with the fact that each action incurs a time delay $D(A)$. $D(A)$ will mainly be influenced by delay caused by READ, as this entails waiting for a human speaker to continue speaking (about 0.3s per word for an average speaker), while WRITE consists of generating a few words from a machine translation system, which is possible on the order of milliseconds. Thus, our objective is finding an optimal policy that generates decision sequences with a good trade-off between higher quality $Q(Y)$ and lower delay $D(A)$. We elaborate on exactly how to define this trade-off in §4.2.

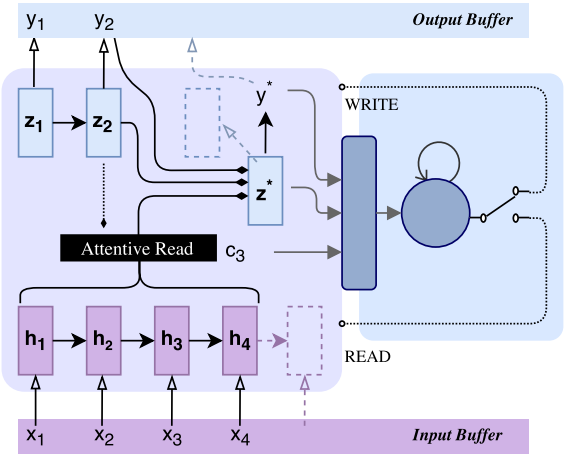



Figure 2: Illustration of the proposed framework: at each step, the NMT environment (left) computes a candidate translation. The recurrent agent (right) will the observation including the candidates and send back decisions–READ or WRITE.

In the following sections, we first describe how to connect the READ/WRITE actions with the NMT system (§3), and how to optimize the system to improve simultaneous MT results (§4).

## 3 Simultaneous Translation with Neural Machine Translation

The proposed framework is shown in Fig. 2, and can be naturally decomposed into two parts: environment (§3.1) and agent (§3.2).

### 3.1 Environment

**Encoder: READ** The first element of the NMT system is the encoder, which converts input words $X = \{x_1, ..., x_{T_s}\}$ into context vectors $H = \{h_1, ..., h_{T_s}\}$. Standard NMT uses bi-directional RNNs as encoders (Bahdanau et al., 2014), but this is not suitable for simultaneous processing as using a reverse-order encoder requires knowing the final word of the sentence before beginning processing. Thus, we utilize a simple left-to-right uni-directional RNN as our encoder:

$$h_\eta = \phi_{\text{UNI-ENC}}\left(h_{\eta-1}, x_\eta\right) \quad (1)$$

**Decoder: WRITE** Similar with standard MT, we use an attention-based decoder. In contrast, we

only reference the words that have been read from the input when generating each target word:

$$\begin{aligned} c_\tau^\eta &= \phi_{\text{ATT}}\left(z_{\tau-1}, y_{\tau-1}, H^\eta\right) \\ z_\tau^\eta &= \phi_{\text{DEC}}\left(z_{\tau-1}, y_{\tau-1}, c_\tau^\eta\right) \\ p\left(y|y_{<\tau}, H^\eta\right) &\propto \exp\left[\phi_{\text{OUT}}\left(z_\tau^\eta\right)\right], \end{aligned} \tag{2}$$

where for $\tau$, $z_{\tau-1}$ and $y_{\tau-1}$ represent the previous decoder state and output word, respectively. $H^\eta$ is used to represent the incomplete input states, where $H^\eta$ is a prefix of $H$. As the WRITE action calculates the probability of the next word on the fly, we need greedy decoding for each step:

$$y_\tau^\eta = \arg\max_y p\left(y|y_{<\tau}, H^\eta\right) \tag{3}$$

Note that $y_\tau^\eta, z_\tau^\eta$ corresponds to $H^\eta$ and is the candidate for $y_\tau, z_\tau$. The agent described in the next section decides whether to take this candidate or wait for better predictions.

### 3.2 Agent

A trainable agent is designed to make decisions $A = \{a_1, .., a_T\}, a_t \in \mathcal{A}$ sequentially based on observations $O = \{o_1, ..., o_T\}, o_t \in \mathcal{O}$, and then control the translation environment properly.

**Observation** As shown in Fig 2, we concatenate the current context vector $c_\tau^\eta$, the current decoder state $z_\tau^\eta$ and the embedding vector of the candidate word $y_\tau^\eta$ as the continuous observation, $o_{\tau+\eta} = [c_\tau^\eta; z_\tau^\eta; E(y_\tau^\eta)]$ to represent the current state.

**Action** Similarly to prior work (Grissom II et al., 2014), we define the following set of actions:

- **READ**: the agent rejects the candidate and waits to encode the next word from input buffer;
- **WRITE**: the agent accepts the candidate and emits it as the prediction into output buffer;

**Policy** How the agent chooses the actions based on the observation defines the policy. In our setting, we utilize a stochastic policy $\pi_\theta$ parameterized by a recurrent neural network, that is:

$$\begin{aligned} s_t &= f_\theta\left(s_{t-1}, o_t\right) \\ \pi_\theta(a_t|a_{<t}, o_{\leq t}) &\propto g_\theta\left(s_t\right), \end{aligned} \tag{4}$$

where $s_t$ is the internal state of the agent, and is updated recurrently yielding the distribution of the action $a_t$. Based on the policy of our agent, the overall algorithm of greedy decoding is shown in Algorithm 1, The algorithm outputs the translation result and a sequence of observation-action pairs.

**Algorithm 1** Simultaneous Greedy Decoding

**Require:** NMT system $\phi$, policy $\pi_\theta$, $\tau_{\text{MAX}}$, input buffer $X$, output buffer $Y$, state buffer $S$.

**Init** $x_1 \Leftarrow X, h_1 \leftarrow \phi_{\text{ENC}}\left(x_1\right), H^1 \leftarrow \{h_1\}$
  $z_0 \leftarrow \phi_{\text{INIT}}\left(H^1\right), y_0 \leftarrow \langle s \rangle$
  $\tau \leftarrow 0, \eta \leftarrow 1$
**while** $\tau < \tau_{\text{MAX}}$ **do**
  $t \leftarrow \tau + \eta$
  $y_\tau^\eta, z_\tau^\eta, o_t \leftarrow \phi\left(z_{\tau-1}, y_{\tau-1}, H^\eta\right)$
  $a_t \sim \pi_\theta\left(a_t; a_{<t}, o_{<t}\right), S \Leftarrow (o_t, a_t)$
  **if** $a_t =$ READ and $x_\eta \neq \langle /s \rangle$ **then**
    $x_{\eta+1} \Leftarrow X, h_{\eta+1} \leftarrow \phi_{\text{ENC}}\left(h_\eta, x_{\eta+1}\right)$
    $H^{\eta+1} \leftarrow H^\eta \cup \{h_{\eta+1}\}, \eta \leftarrow \eta + 1$
    **if** $|Y| = 0$ **then** $z_0 \leftarrow \phi_{\text{INIT}}\left(H^\eta\right)$
  **else if** $a_t =$ WRITE **then**
    $z_\tau \leftarrow z_\tau^\eta, y_\tau \leftarrow y_\tau^\eta$
    $Y \Leftarrow y_\tau, \tau \leftarrow \tau + 1$
    **if** $y_\tau = \langle /s \rangle$ **then break**

## 4 Learning

The proposed framework can be trained using reinforcement learning. More precisely, we use policy gradient algorithm together with variance reduction and regularization techniques.

### 4.1 Pre-training

We need an NMT environment for the agent to explore and use to generate translations. Here, we simply pre-train the NMT encoder-decoder on full sentence pairs with maximum likelihood, and assume the pre-trained model is still able to generate reasonable translations even on incomplete source sentences. Although this is likely sub-optimal, our NMT environment based on uni-directional RNNs can treat incomplete source sentences in a manner similar to shorter source sentences and has the potential to translate them more-or-less correctly.

### 4.2 Reward Function

The policy is learned in order to increase a reward for the translation. At each step the agent will receive a reward signal $r_t$ based on $(o_t, a_t)$. To evaluate a good simultaneous machine translation, a reward must consider both quality and delay.

**Quality** We evaluate the translation quality using metrics such as BLEU (Papineni et al., 2002). The BLEU score is defined as the weighted geometric average of the modified n-gram precision $\text{BLEU}^0$, multiplied by the brevity penalty BP to punish a short translation. In practice, the vanilla

BLEU score is not a good metric at sentence level because being a geometric average, the score will reduce to zero if one of the precisions is zero. To avoid this, we used a smoothed version of BLEU for our implementation (Lin and Och, 2004).

$$\mathrm{BLEU}(Y, Y^*) = \mathrm{BP} \cdot \mathrm{BLEU}^0(Y, Y^*), \quad (5)$$

where $Y^*$ is the reference and $Y$ is the output. We decompose BLEU and use the difference of partial BLEU scores as the reward, that is:

$$r_t^Q = \begin{cases} \Delta\mathrm{BLEU}^0(Y, Y^*, t) & t < T \\ \mathrm{BLEU}(Y, Y^*) & t = T \end{cases} \quad (6)$$

where $Y^t$ is the cumulative output at $t$ ($Y^0 = \emptyset$), and $\Delta\mathrm{BLEU}^0(Y, Y^*, t) = \mathrm{BLEU}^0(Y^t, Y^*) - \mathrm{BLEU}^0(Y^{t-1}, Y^*)$. Obviously, if $a_t = \textsc{read}$, no new words are written into $Y$, yielding $r_t^Q = 0$. Note that we do not multiply BP until the end of the sentence, as it would heavily penalize partial translation results.

**Delay** As another critical feature, delay judges how much time is wasted waiting for the translation. Ideally we would directly measure the actual time delay incurred by waiting for the next word. For simplicity, however, we suppose it consumes the same amount of time listening for one more word. We define two measurements, global and local, respectively:

- **Average Proportion (AP)**: following the definition in (Cho and Esipova, 2016), $X$, $Y$ are the source and decoded sequences respectively, and we use $s(\tau)$ to denote the number of source words been waited when decoding word $y_\tau$,

$$\begin{aligned} 0 < d(X, Y) &= \frac{1}{|X||Y|} \sum_\tau s(\tau) \leq 1 \\ d_t &= \begin{cases} 0 & t < T \\ d(X, Y) & t = T \end{cases} \end{aligned} \quad (7)$$

  $d$ is a global delay metric, which defines the average waiting proportion of the source sentence when translating each word.

- **Consecutive Wait length (CW)**: in speech translation, listeners are also concerned with long silences during which no translation occurs. To capture this, we also consider on how many words were waited for (READ) consecutively between translating two words. For each action, where we initially define $c_0 = 0$,

$$c_t = \begin{cases} c_{t-1} + 1 & a_t = \textsc{read} \\ 0 & a_t = \textsc{write} \end{cases} \quad (8)$$

- **Target Delay:** We further define "target delay" for both $d$ and $c$ as $d^*$ and $c^*$, respectively, as different simultaneous translation applications may have different requirements on delay. In our implementation, the reward function for delay is written as:

$$r_t^D = \alpha \cdot [\mathrm{sgn}(c_t - c^*) + 1] + \beta \cdot \lfloor d_t - d^* \rfloor_+ \quad (9)$$

  where $\alpha \leq 0, \beta \leq 0$.

**Trade-off between quality and delay** A good simultaneous translation system requires balancing the trade-off of translation quality and time delay. Obviously, achieving the best translation quality and the shortest translation delays are in a sense contradictory. In this paper, the trade-off is achieved by balancing the rewards $r_t = r_t^Q + r_t^D$ provided to the system, that is, by adjusting the coefficients $\alpha, \beta$ and the target delay $d^*, c^*$ in Eq. 9.

### 4.3 Reinforcement Learning

**Policy Gradient** We freeze the pre-trained parameters of an NMT model, and train the agent using the policy gradient (Williams, 1992). The policy gradient maximizes the following expected cumulative future rewards, $J = \mathbb{E}_{\pi_\theta}\left[\sum_{t=1}^T r_t\right]$, whose gradient is

$$\nabla_\theta J = \mathbb{E}_{\pi_\theta}\left[\sum_{t'=1}^T \nabla_\theta \log \pi_\theta(a_{t'}|\cdot) R_t\right] \quad (10)$$

$R_t = \sum_{k=t}^T \left[r_k^Q + r_k^D\right]$ is the cumulative future rewards for current observation and action. In practice, Eq. 10 is estimated by sampling multiple action trajectories from the current policy $\pi_\theta$, collecting the corresponding rewards.

**Variance Reduction** Directly using the policy gradient suffers from high variance, which makes learning unstable and inefficient. We thus employ the variance reduction techniques suggested by Mnih and Gregor (2014). We subtract from $R_t$ the output of a baseline network $b_\varphi$ to obtain $\hat{R}_t = R_t - b_\varphi(o_t)$, and centered re-scale the reward as $\tilde{R}_t = \frac{\hat{R}_t - b}{\sqrt{\sigma^2 + \epsilon}}$ with a running average $b$ and standard deviation $\sigma$. The baseline network is trained to minimize the squared loss as follows:

$$L_\varphi = \mathbb{E}_{\pi_\theta}\left[\sum_{t=1}^T \| R_t - b_\varphi(o_t) \|^2\right] \quad (11)$$

We also regularize the negative entropy of the policy to facilitate exploration.

**Algorithm 2** Learning with Policy Gradient

**Require:** NMT system $\phi$, agent $\theta$, baseline $\varphi$

Pretrain the NMT system $\phi$ using MLE;
Initialize the agent $\theta$;
**while** stopping criterion fails **do**
  Obtain a translation pairs: $\{(X, Y^*)\}$;
  **for** $(Y, S) \sim$ Simultaneous Decoding **do**
    **for** $(o_t, a_t)$ in $S$ **do**
      Compute the quality: $r_t^Q$;
      Compute the delay: $r_t^D$;
      Compute the baseline: $b_\varphi(o_t)$;
  Collect the future rewards: $\{R_t\}$;
  Perform variance reduction: $\{\tilde{R}_t\}$;
  Update: $\theta \leftarrow \theta + \lambda_1 \nabla_\theta [J - \kappa \mathcal{H}(\pi_\theta)]$
  Update: $\varphi \leftarrow \varphi - \lambda_2 \nabla_\varphi L$

The overall learning algorithm is summarized in Algorithm 2. For efficiency, instead of updating with stochastic gradient descent (SGD) on a single sentence, both the agent and the baseline are optimized using a minibatch of multiple sentences.

## 5 Simultaneous Beam Search

In previous sections we described a simultaneous greedy decoding algorithm. In standard NMT it has been shown that beam search, where the decoder keeps a beam of $k$ translation trajectories, greatly improves translation quality (Sutskever et al., 2014), as shown in Fig. 3 (A).

It is non-trivial to directly apply beam-search in simultaneous machine translation, as beam search waits until the last word to write down translation. Based on our assumption WRITE does not cost delay, we can perform a simultaneous beam-search when the agent chooses to consecutively WRITE: keep multiple beams of translation trajectories in temporary buffer and output the best path when the agent switches to READ. As shown in Fig. 3 (B) & (C), it tries to search for a relatively better path while keeping the delay unchanged.

Note that we do not re-train the agent for simultaneous beam-search. At each step we simply input the observation of the current best trajectory into the agent for making next decision.

## 6 Experiments

### 6.1 Settings

**Dataset** To extensively study the proposed simultaneous translation model, we train and evaluate it on two different language pairs: "English-German (EN-DE)" and "English-Russian (EN-RU)" in both directions per pair. We use the parallel corpora available from WMT'15[2] for both pre-training the NMT environment and learning the policy. We utilize newstest-2013 as the validation set to evaluate the proposed algorithm. Both the training set and the validation set are tokenized and segmented into sub-word units with byte-pair encoding (BPE) (Sennrich et al., 2015). We only use sentence pairs where both sides are less than 50 BPE subword symbols long for training.

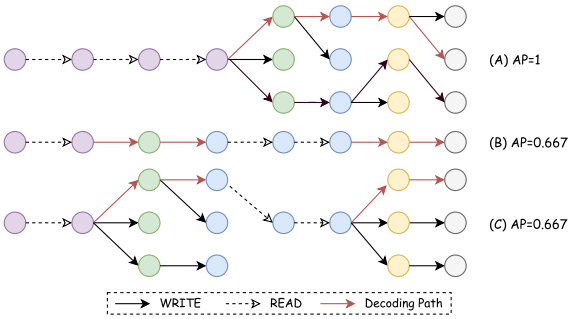


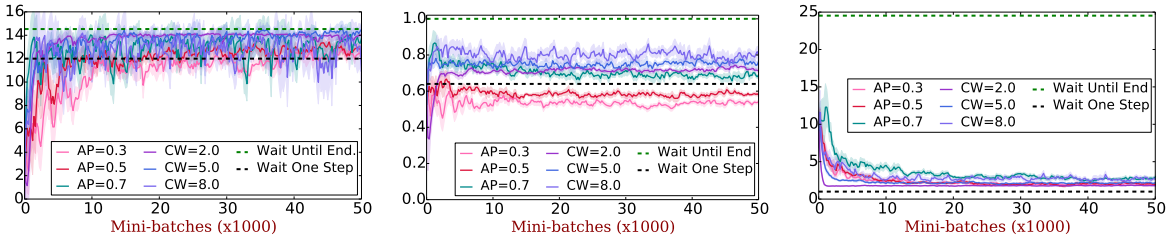

Figure 3: Illustrations of (A) beam-search, (B) simultaneous greedy decoding and (C) simultaneous beam-search.

**Environment & Agent Settings** We pre-trained the NMT environments for both language pairs and both directions following the same setting from (Cho and Esipova, 2016). We further built our agents, using a recurrent policy with 512 GRUs and a softmax function to produce the action distribution. All our agents are trained using policy gradient using Adam (Kingma and Ba, 2014) optimizer, with a mini-batch size of 10. For each sentence pair in a batch, 5 trajectories are sampled. For testing, instead of sampling we pick the action with higher probability each step.

**Baselines** We compare the proposed methods against previously proposed baselines. For fair comparison, we use the same NMT environment:

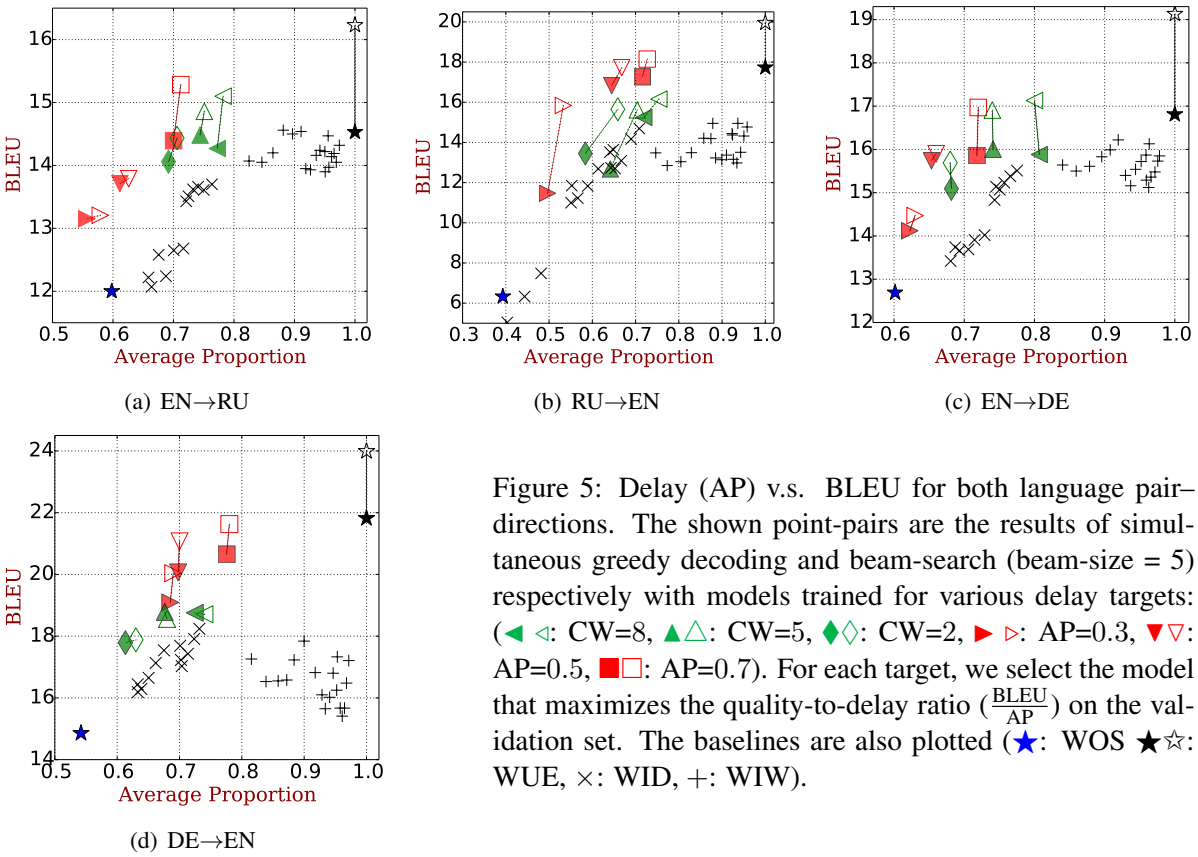
- **Wait-Until-End (WUE)**: an agent that starts to WRITE only when the last source word is seen. In general, we expect this to achieve the best quality of translation. We perform both greedy decoding and beam-search with this method.
- **Wait-One-Step (WOS)**: an agent that WRITEs after each READs. Such a policy is problematic when the source and target language pairs have different word orders or lengths (e.g. EN-DE).

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




(a) BLEU (EN → RU) (b) AP (EN → RU) (c) CW (EN → RU)

Figure 4: Learning progress curves for variant delay targets on the validation dataset for EN → RU. Every time we only keep one target for one delay measure. For instance when using target AP, the coefficient of $\alpha$ in Eq. 9 will be set 0.




(a) EN→RU (b) RU→EN (c) EN→DE

(d) DE→EN

Figure 5: Delay (AP) v.s. BLEU for both language pair–directions. The shown point-pairs are the results of simultaneous greedy decoding and beam-search (beam-size = 5) respectively with models trained for various delay targets: (◀ ◁: CW=8, ▲△: CW=5, ◆◇: CW=2, ▶ ▷: AP=0.3, ▼▽: AP=0.5, ■□: AP=0.7). For each target, we select the model that maximizes the quality-to-delay ratio ($\frac{\text{BLEU}}{\text{AP}}$) on the validation set. The baselines are also plotted (★: WOS ★☆: WUE, ×: WID, +: WIW).

- **Wait-If-Worse/Wait-If-Diff (WIW/WID)**: as proposed by Cho and Esipova (2016), the algorithm first pre-READs the next source word, and accepts this READ when the probability of the most likely target word decreases (WIW), or the most likely target word changes (WID).
- **Segmentation-based (SEG)** (Oda et al., 2014): a state-of-the-art segmentation-based algorithm based on optimizing segmentation to achieve the highest quality score. In this paper, we tried the simple greedy method (SEG1) and the greedy method with POS Constraint (SEG2).

### 6.2 Quantitative Analysis

In order to evaluate the effectiveness of our reinforcement learning algorithms with different reward functions, we vary the target delay $d^* \in \{0.3, 0.5, 0.7\}$ and $c^* \in \{2, 5, 8\}$ for Eq. 9 separately, and trained agents with $\alpha$ and $\beta$ adjusted to values that provided stable learning for each language pair according to the validation set.

**Learning Curves** As shown in Fig. 4, we plot learning progress for EN-RU translation with different target settings. It clearly shows that our algorithm effectively increases translation quality for all the models, while pushing the delay close, if not all of the way, to the target value. It can also be noted from Fig. 4 (a) and (b) that there exists strong correlation between the two delay measures, implying the agent can learn to decrease both AP and CW simultaneously.

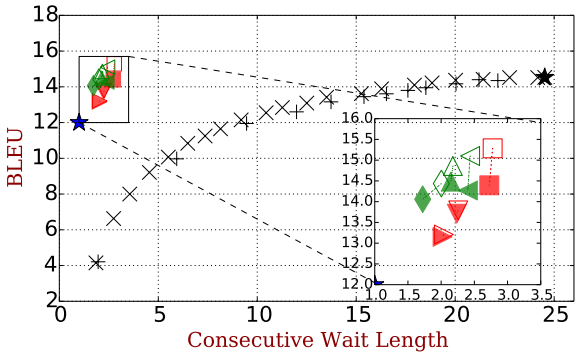



Figure 6: Delay (CW) v.s. BLEU score for EN → RU, (◀ ◁: CW=8, ▲△: CW=5, ◆◇: CW=2, ▶ ▷: AP=0.3, ▼▽: AP=0.5, ■□: AP=0.7), against the baselines (★: WOS ★: WUE, +: SEG1, ×: SEG2).

**Quality v.s. Delay** As shown in Fig. 5, it is clear that the trade-off between translation quality and delay has very similar behaviors across both language pairs and directions. The smaller delay (AP or CW) the learning algorithm is targeting, the lower quality (BLEU score) the output translation. It is also interesting to observe that, it is more difficult for "→EN" translation to achieve a lower AP target while maintaining good quality, compared to "EN→". In addition, the models that are optimized on AP tend to perform better than those optimized on CW, especially in "→EN" translation. German and Russian sentences tend to be longer than English, hence require more consecutive waits before being able to emit the next English symbol.

**v.s. Baselines** In Fig. 5 and 6, the points closer to the upper left corner achieve better trade-off performance. Compared to WUE and WOS which can ideally achieve the best quality (but the worst delay) and the best delay (but poor quality) respectively, all of our proposed models find a good balance between quality and delay. Some of the proposed models can achieve good BLEU scores close to WUE, while have much smaller delay.

Compared to the method of Cho and Esipova (2016) based on two hand-crafted rules (WID, WIW), in most cases our proposed models find better trade-off points, while there are a few exceptions. We also observe that the baseline models have trouble controlling the delay in a reasonable area. In contrast, by optimizing towards a given target delay, our proposed model is stable while maintaining good translation quality.

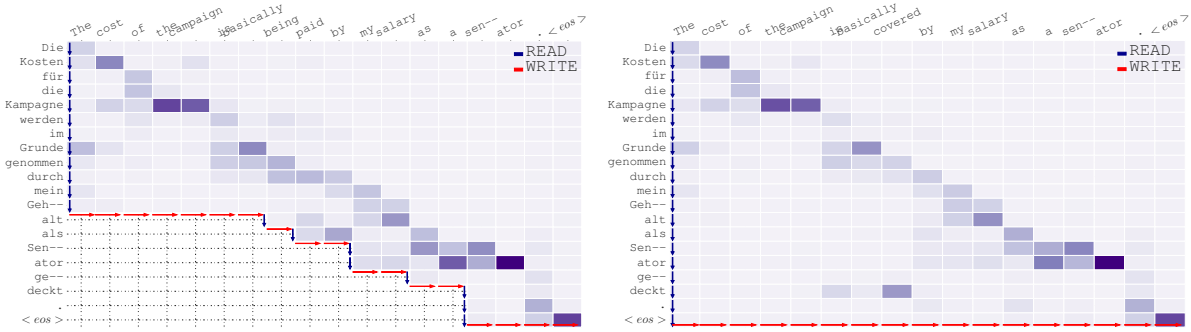
We also compared against Oda et al. (2014)'s state-of-the-art segmentation algorithm (SEG). As shown in Fig 6, it is clear that although SEG can work with variant segmentation lengths (CW), the proposed model outputs high quality translations at a much smaller CW. We conjecture that this is due to the independence assumption in SEG, while the RNNs and attention mechanism in our model makes it possible to look at the whole history to decide each translated word.

**w/o Beam-Search** We also plot the results of simultaneous beam-search instead of using greedy decoding. It is clear from Fig. 5 and 6 that most of the proposed models can achieve an visible increase in quality together with a slight increase in delay. This is because beam-search can help to avoid bad local minima. We also observe that the simultaneous beam-search cannot bring as much improvement as it did in the standard NMT setting. In most cases, the smaller delay the model achieves, the less beam search can help as it requires longer consecutive WRITE segments for extensive search to be necessary. One possible solution is to consider the beam uncertainty in the agent's READ/WRITE decisions. We leave this to future work.

### 6.3 Qualitative Analysis

In this section, we perform a more in-depth analysis using examples from both EN-RU and EN-DE pairs, in order to have a deeper understanding of the proposed algorithm and its remaining limitations. We only perform greedy decoding to simplify visualization.

**EN→RU** As shown in Fig 8, since both English and Russian are Subject-Verb-Object (SVO) languages, the corresponding words may share the same order in both languages, which makes simultaneous translation easier. It is clear that the larger the target delay (AP or CW) is set, the more words are read before translating the corresponding words, which in turn results in better translation quality. We also note that very early WRITE commonly causes bad translation. For example, for AP=0.3 & CW=2, both the models choose to WRITE in the very beginning the word "The", which is unreasonable since Russian has no articles, and there is no word corresponding to it. One good feature of using NMT is that the more words the decoder READs, the longer history is saved, rendering simultaneous translation easier.




(a) Simultaneous Neural Machine Translation

(b) Neural Machine Translation

Figure 7: Comparison of DE→EN examples using the proposed framework and usual NMT system respectively. Both the heatmaps share the same setting with Fig. 1. The verb "gedeckt" is incorrectly translated in simultaneous translation.

| Source | AP=0.3 | AP=0.7 | CW=2 | CW=8 |
|---|---|---|---|---|
| The | The The | | The | |
| people | p-- i-- ent the p-- ol-- s | | | |
| , | , | | p-- riv-- | |
| as | | | ers | |
| I | as I | | | |
| heard | я слышал | Люди | , | Люди |
| in | | , как я слышал | | |
| the | | | как | |
| countryside | в | в | я слышал | |
| , | | | | , |
| want | сельской местности | сельской | в | как я |
| a | | местности | | слышал |
| Government | | | сельской | |
| that | , | | | в |
| is | | | местности | сельской |
| not | | | | |
| made | хочу | | , | |
| up | правительство | | | местности |
| of | | | хочу | |
| thi-- | , | , | | |
| eves | | хотят | правительство | |
| . | | | | , |
| <eos> | которое не производится во-- ров . | , чтобы правительство , которое не в-- меши-- вается в во-- ры . | , которое не является состав-- ной частью во-- ров . | хотят , чтобы правительство , которое не в-- меши-- вается в во-- ры . |
| Summary | BLEU=39/ AP=0.46 | BLEU=64/AP=0.77 | BLEU=54/CW=1.76 | BLEU=64/CW=2.55 |

Figure 8: Given the example input sentence (leftmost column), we show outputs by models trained for various delay targets. For these outputs, each row corresponds to one source word and represents the emitted words (maybe empty) after reading this word. The corresponding source and target words are in the same color for all model outputs.

**DE→EN** As shown in Fig 1 and 7 (a), where we visualize the attention weights as soft alignment between the progressive input and output sentences, the highest weights are basically along the diagonal line. This indicates that our simultaneous translator works by waiting for enough source words with high alignment weights and then switching to write them.

DE-EN translation is likely more difficult as German usually uses Subject-Object-Verb (SOV) constructions a lot. As shown in Fig 1, when a sentence (or a clause) starts the agent has learned such policy to READ multiple steps to approach the verb (e.g. serviert and gestorben in Fig 1). Such a policy is still limited when the verb is very far from the subject. For instance in Fig. 7, the simultane-

ous translator achieves almost the same translation with standard NMT except for the verb "gedeckt" which corresponds to "covered" in NMT output. Since there are too many words between the verb "gedeckt" and the subject "Kosten für die Kampagne werden", the agent gives up reading (otherwise it will cause a large delay and a penalty) and WRITEs "being paid" based on the decoder's hypothesis. This is one of the limitations of the proposed framework, as the NMT environment is trained on complete source sentences and it may be difficult to predict the verb that has not been seen in the source sentence. One possible way is to fine-tune the NMT model on incomplete sentences to boost its prediction ability. We will leave this as future work.

## 7 Related Work

Researchers commonly consider the problem of simultaneous machine translation in the scenario of real-time speech interpretation (Fügen et al., 2007; Bangalore et al., 2012; Fujita et al., 2013; Rangarajan Sridhar et al., 2013; Yarmohammadi et al., 2013). In this approach, the incoming speech stream required to be translated are first recognized and segmented based on an automatic speech recognition (ASR) system. The translation model then works independently based on each of these segments, potentially limiting the quality of translation. To avoid using a fixed segmentation algorithm, Oda et al. (2014) introduced a trainable segmentation component into their system, so that the segmentation leads to better translation quality. Grissom II et al. (2014) proposed a similar framework, however, based on reinforcement learning. All these methods still rely on translating each segment independently without previous context.

Recently, two research groups have tried to apply the NMT framework to the simultaneous translation task. Cho and Esipova (2016) proposed a similar waiting process. However, their waiting criterion is manually defined without learning. Satija and Pineau (2016) proposed a method similar to ours in overall concept, but it significantly differs from our proposed method in many details. The biggest difference is that they proposed to use an agent that passively reads a new word at each step. Because of this, it cannot consecutively decode multiple steps, rendering beam search difficult. In addition, they lack the comparison to any existing approaches. On the other hand, we perform an extensive experimental evaluation against state-of-the-art baselines, demonstrating the relative utility both quantitatively and qualitatively.

The proposed framework is also related to some recent efforts about online sequence-to-sequence (SEQ2SEQ) learning. Jaitly et al. (2015) proposed a SEQ2SEQ ASR model that takes fixed-sized segments of the input sequence and outputs tokens based on each segment in real-time. It is trained with alignment information using supervised learning. A similar idea for online ASR is proposed by Luo et al. (2016). Similar to Satija and Pineau (2016), they also used reinforcement learning to decide whether to emit a token while reading a new input at each step. Although sharing some similarities, ASR is very different from simultaneous MT with a more intuitive definition for segmentation. In addition, Yu et al. (2016) recently proposed an online alignment model to help sentence compression and morphological inflection. They regarded the alignment between the input and output sequences as a hidden variable, and performed transitions over the input and output sequence. By contrast, the proposed READ and WRITE actions do not necessarily to be performed on aligned words (e.g. in Fig. 1), and are learned to balance the trade-off of quality and delay.

## 8 Conclusion

We propose a unified framework to do neural simultaneous machine translation. To trade off quality and delay, we extensively explore various targets for delay and design a method for beam-search applicable in the simultaneous MT setting. Experiments against state-of-the-art baselines on two language pairs demonstrate the efficacy both quantitatively and qualitatively.

## Acknowledgments

KC acknowledges the support by Facebook, Google (Google Faculty Award 2016) and NVidia (GPU Center of Excellence 2015-2016). GN acknowledges the support of the Microsoft CORE program. This work was also partly supported by Samsung Electronics (Project: "Development and Application of Larger-Context Neural Machine Translation").

## References


Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

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. Association for Computational Linguistics.

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

Christian Fügen, Alex Waibel, and Muntsin Kolss. 2007. Simultaneous translation of lectures and speeches. *Machine Translation*, 21(4):209–252.

Tomoki Fujita, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2013. Simple, lexicalized choice of translation timing for simultaneous speech translation. In *INTERSPEECH*.

Alvin Grissom II, He He, Jordan Boyd-Graber, John Morgan, and Hal Daumé III. 2014. Dont 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, October. Association for Computational Linguistics.

Navdeep Jaitly, Quoc V Le, Oriol Vinyals, Ilya Sutskeyver, and Samy Bengio. 2015. An online sequence-to-sequence model using partial conditioning. *arXiv preprint arXiv:1511.04868*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv: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 *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 605. Association for Computational Linguistics.

Yuping Luo, Chung-Cheng Chiu, Navdeep Jaitly, and Ilya Sutskever. 2016. Learning online alignments with continuous rewards policy gradient. *arXiv preprint arXiv:1608.01281*.

Takashi Mieno, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2015. Speed or accuracy? a study in evaluation of simultaneous speech translation. In *INTERSPEECH*.

Andriy Mnih and Karol Gregor. 2014. Neural variational inference and learning in belief networks. *arXiv preprint arXiv:1402.0030*.

Yusuke Oda, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2014. Optimizing segmentation strategies for simultaneous speech translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 551–556, Baltimore, Maryland, June. 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 on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Vivek Kumar Rangarajan Sridhar, John Chen, Srinivas Bangalore, Andrej Ljolje, and Rathinavelu Chengalvarayan. 2013. Segmentation strategies for streaming speech translation. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 230–238, Atlanta, Georgia, June. Association for Computational Linguistics.

Harsh Satija and Joelle Pineau. 2016. Simultaneous machine translation using deep reinforcement learning. *Abstraction in Reinforcement Learning Workshop, ICML2016*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Mahsa Yarmohammadi, Vivek Kumar Rangarajan Sridhar, Srinivas Bangalore, and Baskaran Sankaran. 2013. Incremental segmentation and decoding strategies for simultaneous translation. In *IJCNLP*, pages 1032–1036.

Lei Yu, Jan Buys, and Phil Blunsom. 2016. Online segment to segment neural transduction. *arXiv preprint arXiv:1609.08194*.

## A Joint learning

Following the notation, let the input pipe as $X = \{x_1, x_2, ..., x_{T_s}\}$, and the output pipe as $Y = \{y_1, y_2, ..., y_{T_t}\}$. At every time step $t = \tau + \eta$, we first pre-compute the context vector $c_\tau^\eta$, the current hidden state $z_\tau^\eta$, and output the pre-decoded word $y_\tau^\eta$ as the input $o_t$ of the agent. In our paper, we used a greedy decoder to get the pre-decoded word:

$$y_\tau^\eta = \arg\max_y p_\phi(y|y_{<\tau}, x_{\leq\eta})$$

Here, we can also use sampling as

$$y_\tau^\eta \sim p_\phi(y|y_{<\tau}, x_{\leq\eta})$$

The agent $\theta$ outputs READ/WRITE actions as $A = \{a_1, a_2, ..., a_T\}$, where $T = T_s + T_t$. When the agent chooses to WRITE, the pre-decoded word $y_\tau^\eta$ is picked as $y_\tau$. The agent receives rewards $R = \{r_1, r_2, ..., r_T\}$. Thus, we can learn the agent's policy $\pi_\theta$ using REINFORCE, by assuming the NMT model is fixed:

$$\nabla_\theta J = \mathbb{E}_{\pi_\theta}\left[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|o_t) \cdot R_t\right]$$

where $R_t = \sum_{k=t}^{T} r_k$. The idea of joint learning is that we can also optimize the underlying NMT model with REINFORCE, by assuming the policy of the agent is fixed:

$$\nabla_\phi J = \mathbb{E}_{p_\phi}\left[\sum_{\tau=1}^{T_t} \nabla_\phi \log p_\phi(y_\tau|y_{<\tau}, x_{\leq\eta}) \cdot R_{\tau+\eta}\right]$$

where we use $\eta$ to show the number of the corresponded source words at each decoded word $y_\tau$. The logic is a little similar with actor-critic. The agent tries to find the best policy $\pi_\theta$ given the current model, and the model $\phi$ tries to output the best translation under the current policy. One question is that I am not sure if we need to compute the likelihood for the pre-decoded words that are dropped.