*INTERSPEECH 2020*
October 25–29, 2020, Shanghai, China

# Efficient Wait-$k$ Models for Simultaneous Machine Translation

*Maha Elbayad[1], Laurent Besacier[1], Jakob Verbeek[2]*

[1] Univ. Grenoble Alpes, CNRS, Grenoble INP, Inria, LIG, LJK, F-38000 Grenoble France
[2] Facebook AI Research

[1] firstname.lastname@univ-grenoble-alpes.fr

## Abstract

Simultaneous machine translation consists in starting output generation before the entire input sequence is available. Wait-$k$ decoders offer a simple but efficient approach for this problem. They first read $k$ source tokens, after which they alternate between producing a target token and reading another source token. We investigate the behavior of wait-$k$ decoding in low resource settings for spoken corpora using IWSLT datasets. We improve training of these models using unidirectional encoders, and training across multiple values of $k$. Experiments with Transformer and 2D-convolutional architectures show that our wait-$k$ models generalize well across a wide range of latency levels. We also show that the 2D-convolution architecture is competitive with Transformers for simultaneous translation of spoken language.

## 1. Introduction

Neural Sequence-to-Sequence (S2S) models are state-of-the-art for sequential prediction tasks including machine translation, speech recognition, speech translation, text-to-speech synthesis, *etc*. The most widespread models are composed of an encoder that reads the entire input sequence, while a decoder (often equipped with an attention mechanism) iteratively produces the next output token given the input and the partial output decoded so far. While these models perform very well in the typical *offline* decoding use case, recent works studied how S2S models are affected by *online* (or simultaneous) constraints, and which architectures and strategies are the most efficient. Online decoding is desirable for applications such as real-time speech-to-speech interpretation. In such scenarios, the decoding process starts before the entire input sequence is available, and online prediction generally comes at the cost of reduced translation quality. In this paper we improve training and decoding of deterministic *wait-k* models that are simple and efficient for online decoding [1, 2]. These decoders first read $k$ tokens from the source, after which they alternate between producing a target token and reading another source token, see Figure 1.

In summary our contributions are: (1) we propose improved training techniques for wait-$k$ by first using unidirectional encoders and training across multiple values of $k$. (2) we show that 2D convolutional architectures are competitive with transformers for simultaneous (online) translation, especially in lower resource settings such as encountered for spoken corpora (IWSLT datasets), (3) we show that training along multiple wait-$k$ paths achieves good online performance without the need to set a suitable $k$ *a priori* for training. Moreover, models trained in this manner generalize well across a wide range of latency levels.

## 2. Related work

After pioneering works on online statistical MT [3, 4, 5, 6, 7], one of the first works on online translation to use attention-based sequence-to-sequence models is that of [8], which uses

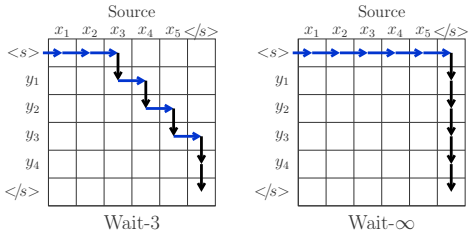



Figure 1: *Wait-$k$ decoding as a sequence of reads (horizontal) and writes (vertical) over a source-target grid. After first reading $k$ tokens, the decoder alternates between reads and writes. In Wait-$\infty$, or Wait-until-End (WUE), the entire source is read first.*

manually designed criteria that dictate whether the model should make a read/write operation. [9] reads equally-sized chunks of the source sequence and generates output sub-sequences of variable lengths, each ending with a special end-of-segment token. [10] proposed a deterministic decoding algorithm that starts with $k$ read operations then alternates between blocks of $l$ write/read operations. This simple approach outperforms the information based criteria of [8], and allows complete control of the translation delay. [1] trained Transformer models [11] with a *wait-k* decoding policy that first reads $k$ source tokens then alternate single read-writes. Wait-$k$ approaches were found most effective by [2] when trained for the specific $k$ that is used to generate translations. This, however, requires training separate models for each potential value of $k$ used for translation.

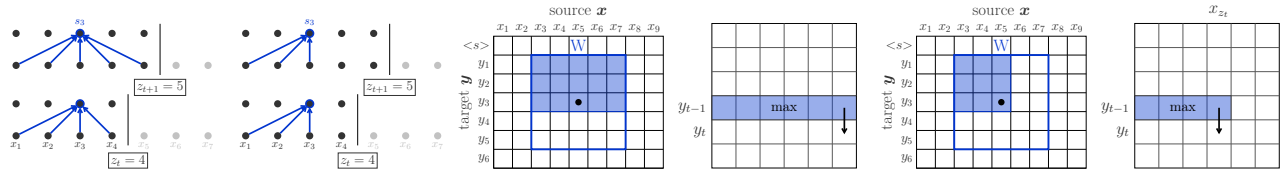
For *dynamic* online decoding, [12, 13] rely on reinforcement learning (RL) to optimize a read/write policy. [12] learns an LSTM model that emits read/write decisions based on the input and output prefixes processed so far. [13] trains an RNN fed with the encoder and decoder current hidden states to generate read/write decisions. RL based models are first pre-trained offline, and then fine-tuned with policy gradient to optimize a reward balancing translation quality and latency.

To combine the end-to-end training of *wait-k* models with the flexibility of dynamic online decoding, [14, 15, 2] use imitation learning (IL). [14] estimates a decoding path from the source-target alignments obtained with an off-the-shelf alignment model then trains a recurrent network to jointly encode the two sequences following the alignment path. [15] adds end-of-segment tokens to the target w.r.t. two wait-$k$ paths during training. At test time, decoding is controlled with the end-of-segment token and constrained to lie in between the two training paths. [2] supervises the training of a decoder that controls the read/write decision using an oracle derived from an offline translation model.

Recent work on *dynamic* online translation use monotonic alignments [16]. They were first introduced as a substitute for attention that enables progressive reading of the source context and where only a single (last read) encoder state is fed to the decoder. *MoChA* [17] adds chunkwise attention on top of mono-



http://dx.doi.org/10.21437/Interspeech.2020-1241




(a) *Bidirectional att.* (b) *Unidirectional att.* (c) *Offline convolution* (d) *Offline pooling* (e) *Online convolution* (f) *Online pooling*

Figure 2: *Illustration of bi/uni-directional attention, and causal two-dimensional convolutions. At the position marked with •, the convolution only includes signals from the highlighted blue area, the other weights are zeroed out.*

tonic alignments to attend to a window of the last encoder states and *MILk* [18] broadens this window with an infinite lookback to boost the translation quality. [19] adapted *MILk*'s monotonic attention for multi-headed Transformer decoders.

Simultaneous translation models usually operate under a *streaming* constraint where an emitted output cannot be altered, alternatively, [20, 21, 22] propose decoding strategies that allow for revision to correct past outputs.

In our work, we focus on wait-$k$ decoding, but unlike [1] we opt for unidirectional encoders, and show that they are more effective and efficient for online translation. We also show that it is possible to train a single model that is effective across a large range of latency levels.

## 3. Online translation models

In the following we describe how we adapt the transformer [11] and pervasive attention [23] architectures to the online translation setting, and how we train them for wait-$k$ decoding.

### 3.1. Online Transformer (TF)

The key component of the transformer model [11] is multi-headed attention, which concatenates the outputs of multiple attention heads. Attention aggregates encoder/decoder states from other positions in the form of a weighted sum, where the weights depend on the current state.

Wait-$k$ online decoding [1, 10] starts by reading $k$ source tokens, and then alternates between reading and writing a single token at a time, until the full source has been read, or the target generation has been terminated. Formally, we denote with $z_t$ the number of source tokens read when decoding $y_t$. For a wait-$k$ decoding path we have $z_t = \min(k + t - 1, |\boldsymbol{x}|)$.

In the encoder-decoder attention, the decoder state $h_t$, used to predict $y_{t+1}$, attends to the first $z_{t+1}$ source states, and each source state should only encode signal from the $z_t$ source tokens read so far. Self-attention makes the source encoder bidirectional, *i.e.* the encoder state at a given position includes signals from past as well as future time-steps. This means that, as in [1, 2], the bidirectional source encoding has to be recomputed each time a source token is read, so that past source tokens can attend to the newly available source tokens, see Figure 2a.

To alleviate the cost of re-encoding the input sequence after each read operation, we propose unidirectional encoders for online translation, by masking the self-attention to only consider previous time-steps, as in Figure 2b. In this manner, unlike [1, 2], both during training and deployment, source sequences are encoded once, without having to update the encoder states as new source tokens become available.

### 3.2. Online Pervasive Attention (PA)

In Pervasive Attention [23], the source and target sequences are jointly encoded with a two-dimensional convolutional neural network (CNN). For decoding, [23] uses causal convolutions where the filters mask future target positions, see Figure 2c. In a similar way, to adapt to the online translation task, we mask the filters in the source direction in order to encode the source unidirectionally, see Figure 2e. In our online version of Pervasive Attention, the CNN's ultimate features $\mathrm{H}^{\text{conv}}$ at a given position $(t, j)$ of the source-target grid encode source context up to $x_j$ and target context up to $y_t$. In the offline Pervasive Attention, a single representation per target position is needed to predict the next output token, which is obtained using max-pooling across the source positions, see Figure 2d. In the online task, we would like to make a prediction at every position where the model could be asked to write. When predicting $y_t$ at position $(t-1, z_t)$ of the grid we max-pool the activations in $\mathrm{H}^{\text{conv}}_{t-1,\leq z_t}$, see Figure 2f.

One major difference between Pervasive Attention (PA) and Transformer is that in PA, the source-target representation at any given position $(t, z_t)$ is independent from the decoding path $\boldsymbol{z}$ taken to get to that point. In a transformer model, however, the representation at $(t, z_t)$ depends on the order in which tokens were read and written up to that point.

### 3.3. Training wait-$k$ models

In [1, 10] the model is optimized by maximum likelihood estimation w.r.t. a single wait-$k$ decoding path $\boldsymbol{z}^k$:

$$\log p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{z}^k) = \sum_{t=1}^{|\boldsymbol{y}|} \log p_\theta(y_t | \boldsymbol{y}_{<t}, \boldsymbol{x}_{\leq z_t^k}, \boldsymbol{z}^k_{<t}). \quad (1)$$

Note that the dependency on $\boldsymbol{z}^k_{<t}$ is only relevant for the Transformer model where the path history matters.

Instead of optimizing a single decoding path, we propose to jointly optimize across multiple wait-$k$ paths. The additional loss terms provide a richer training signal, and potentially yield models that could perform well in different latency regimes. Due to the dependence of the decoder hidden states on the full decoding path $\boldsymbol{z}_{<t}$ in the transformer-based model, we can only train in parallel across a limited set of paths. We consider wait-$k$ paths $\{\boldsymbol{z}^k, \forall k \in \mathrm{K}\}$. During training, we encode the source sequence once, and uniformly sample $k \in \mathrm{K}$ to decode:

$$\mathbb{E}_{\mathrm{K}}\Big[\log p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{z}^k)\Big] \approx \sum_{k \sim \mathcal{U}(\mathrm{K})} \log p_\theta(\boldsymbol{y} | \boldsymbol{x}, \boldsymbol{z}^k). \quad (2)$$

To cover all possible wait-$k$ paths for an input $(\boldsymbol{x}, \boldsymbol{y})$, we set $\mathrm{K} = [1, \ldots, |\boldsymbol{x}|]$. We will refer to this training with *multi-path*.

With Pervasive Attention, we can leverage more training signals. In fact, the grid nature of the model allows us to efficiently compute the output distributions $p(y_t | \boldsymbol{y}_{<t}, \boldsymbol{x}_{\leq j})$ all over the grid in a single forward pass. Consequently, we optimize the writing log-likelihoods in the area above the diagonal with:

$$\sum_{t=1}^{|\boldsymbol{y}|} \sum_{j=1}^{|\boldsymbol{x}|} \log p_\theta(y_t | \boldsymbol{y}_{<t}, \boldsymbol{x}_{\leq j}) [\![ j \geq t ]\!]. \quad (3)$$

We will refer to this training with *multi-path* as well.

Table 1: *Evaluation of Pervasive Attention (PA) and Transformer (TF) for offline translation with greedy decoding.*

| Encoder | Unidirectional | | Bidirectional | |
|---|---|---|---|---|
| Architecture | PA | TF | PA | TF |
| IWSLT'14 En→De | **26.81** | 26.58 | 27.23 | **27.46** |
| IWSLT'14 De→En | 32.40 | **32.81** | 33.43 | **33.64** |
| IWSLT'15 En→Vi | **29.22** | 28.90 | **29.81** | 29.33 |
| IWSLT'15 Vi→En | **26.81** | 25.73 | 27.43 | **28.09** |
| WMT'15 De→En | 28.08 | **31.14** | 28.78 | **31.96** |

# 4. Experimental evaluation

## 4.1. Datasets and experimental setup

We evaluate our approach on IWSLT14 En↔De [24], IWSLT'15 En↔Vi [25], and WMT15 De→En datasets.[1] We train offline unidirectional and bidirectional Transformer (TF) and Pervasive Attention (PA) models on all tasks. On IWSLT'14 De↔En, similar to [23, 26], we train on 160K pairs, develop on 7K held out pairs and test on TED dev2010+tst2010-2013 (6,750 pairs). All data is tokenized and lower-cased and we segment sequences using byte pair encoding [27] with 10K merge operations. The resulting vocabularies are of 8.8K and 6.6K types in German and English respectively. On IWSLT'15 En↔Vi, similar to [19, 25], we train on 133K pairs, develop on TED tst2012 (1,553 pairs) and test on TED tst2013 (1,268 pairs). The corpus was simply tokenized resulting in 17K and 7.7K word vocabularies in English and Vietnamese respectively. On WMT'15 De→En, we reproduce the setup of [1, 18] with a joint vocabulary of 32K BPE types. We train on 4.5M pairs, develop on newstest2013 (3,000 pairs) and test on newstest15 (2,169 pairs).

Our Pervasive Attention models use residual-cumulative skip connections and stack $N = 14$ layers with $11 \times 11$ convolutions. We train Transformer *small* on IWSLT'14 De↔En, a modified *base* [19] on IWSLT'15 En↔Vi and Transformers *base* and *big* on WMT'15 De→En.

We evaluate all models with greedy decoding, and measure translation quality measured with tokenized word-level BLEU [28] with *multi-bleu.pl*. We measure decoding latency with Average Lagging (AL) [1], which is designed to indicate the source steps by which we lag behind an ideal translator (wait-0), it can however be negative if the system finishes decoding prematurely before the full source is read. Other measures of lagging include *Average proportion* (AP) [8] and *Differentiable Average Lagging* (DAL) [18]. AP is unfavorable to short sequences and is incapable of highlighting improvement as it occupies a narrow range [1, 18, 29]. DAL is a differentiable version of AL used to regularize trainable decoders, and behaves similarly to AL.

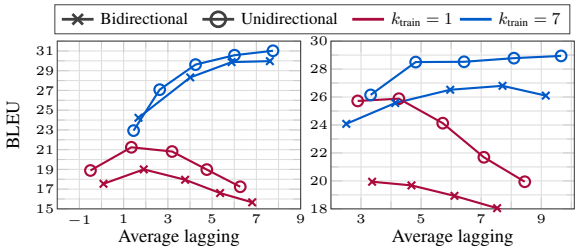
## 4.2. Offline comparison

Table 1 reports offline performance of Pervasive Attention (PA) and Transformer (TF) models with both a unidirectional encoder and a bidirectional encoder. Overall, and as expected, bidirectional encoders in the offline setup are better than their unidirectional counterparts. The gain for PA is of 0.65 on average while for TF the addition of bidirectionality improves BLEU by 1.1 on average. The first two columns of Table 1 show that pervasive attention (PA) is competitive with TF on these datasets when using unidirectional encoders: PA improves upon TF on three of the five tasks.

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

Table 2: *Decoding speed of Transformers with uni/bi-directional encoders for De→En on IWSLT'14 and WMT'15.*

| | Encoder | Decoding (tok/s) GPU | CPU |
|---|---|---|---|
| IWSLT De→En | Unidirectional | 21.7k | 130 |
| | Bidirectional | 7.3k | 54 |
| WMT De→En | Unidirectional | 6.3k | 77 |
| | Bidirectional | 2.9k | 32 |




(a) *IWSLT'14 De→En - TF* (b) *IWSLT'15 En→Vi - TF*

Figure 3: *Transformer models with bi/uni-directional encoders trained on wait-1 and wait-7 decoding paths.*

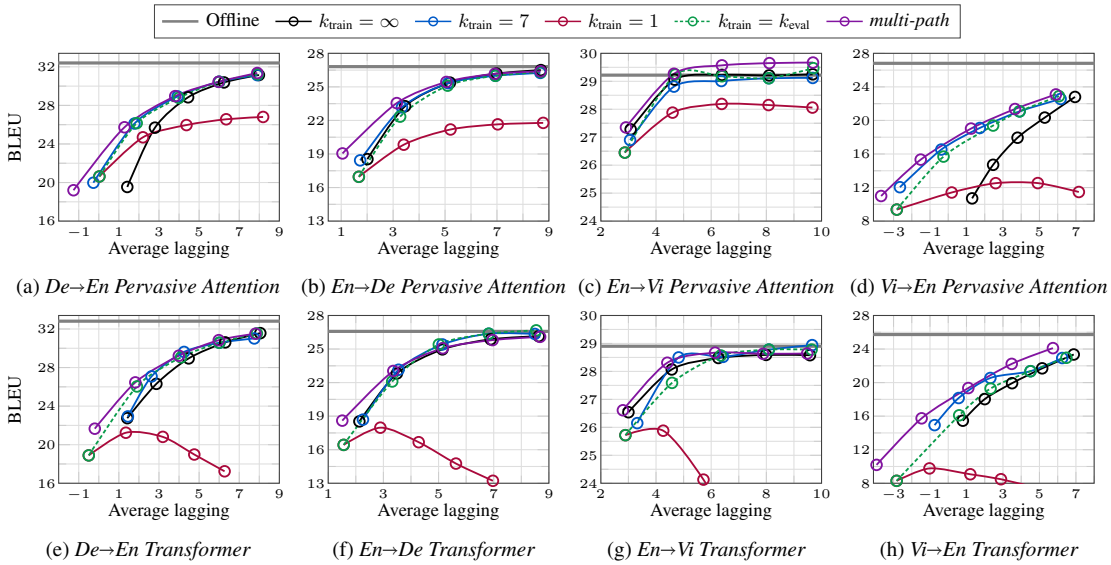
## 4.3. Online comparison

For the Transformer model, we initially consider a bidirectional encoder similar to [1, 2], in which case the encoder states have to be updated after each read. The timing results in Table 2 show that using a bidirectional encoder rather than a unidirectional one slows decoding down by a factor two to three. Second, in Figure 3 we assess the impact of the uni/bidirectional encoder on online decoding quality. We look at models trained using either of two wait-$k$ paths: $k_{\text{train}}$=1 and $k_{\text{train}}$=7. We observe that in the case of online decoding unidirectional encoding performs best, in contrast to the case for offline decoding. For both $k_{\text{train}}$=1 and $k_{\text{train}}$=7, the unidirectional encoder is consistently providing better performance. For the experiments below we therefore use unidirectional encoders.

**Pervasive Attention and Transformer for online translation.** We evaluate models trained for different wait-$k$ decoding paths. We denote with $k_{\text{train}}$=$\infty$ the wait-until-end training where the full source is read before decoding. We report offline results for reference, the offline model has a latency of AL = $|\boldsymbol{x}|$.

Figure 4 presents the performance of models trained for a single wait-$k$ decoding path, with $k_{\text{train}} \in \{1, 7, \infty\}$. Each trained model is represented by a curve, by evaluating it across different wait-$k$ decoding paths $k_{\text{eval}} \in \{1, 3, 5, 7, 9\}$.

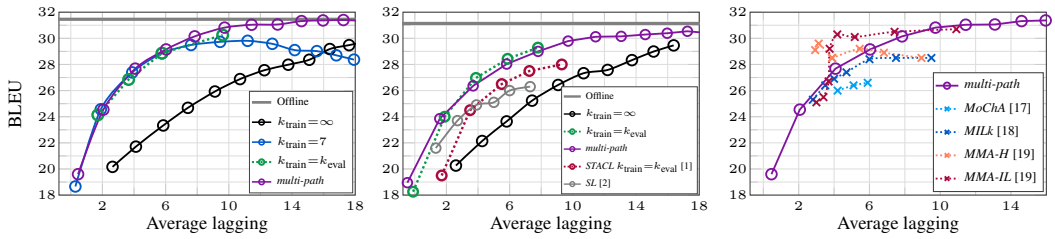
Initial experiments with both architectures across the four IWSLT tasks showed that models trained on wait-7 generalize well on other evaluation paths. Thus, unlike [1], we note that we can train a single model to use in different latency regimes, *i.e.* we do not achieve better BLEU scores when training with $k_{\text{eval}} = k_{\text{train}}$. This generalization to other wait-$k$ paths is notably stronger with pervasive attention (PA) models. Where TF models drop in performance far from the training path (*e.g.* $k_{\text{train}} = 1$ and $k_{\text{eval}} = 9$), the PA models continue to improve for larger $k_{\text{eval}}$. Overall, for tasks where PA performs better offline the model consistently outperforms TF online and vice-versa. It is worth noting that for some translation directions, we can outperform the offline model's performance at a considerably lower latency.




(a) *De→En Pervasive Attention* (b) *En→De Pervasive Attention* (c) *En→Vi Pervasive Attention* (d) *Vi→En Pervasive Attention*

(e) *De→En Transformer* (f) *En→De Transformer* (g) *En→Vi Transformer* (h) *Vi→En Transformer*

Figure 4: *IWSLT'14 De↔En and IWSLT'15 En↔Vi: wait-k online decoding with Pervasive Attention (top) and Transformer (bottom), both with unidirectional encoder. Each curve represents a model trained on a single decoding path, evaluated with* $k_{eval} \in \{1, 3, 5, 7, 9\}$. *Offline models have an average lagging of 22.97, 22.21, 21.08 and 26.56 on De→En, En→De, En→Vi and Vi→En, respectively.*




(a) *Our TF big models* (b) *SoTA comparison, TF base* (c) *SoTA comparison, TF big*

Figure 5: *Evaluation of our models on WMT'15 De→En, and comparison to the state of the art (SoTA). Offline models have an average lagging of 26.96. Note that in panel (b) STACL* $k_{train} = k_{eval}$ *shows the results of [2] where an end-of-sequence marker was added to the source to improve over [1].*

This is in particular the case for En→Vi where the online PA with an AL of 4.65 matches the performance of the offline model with an AL of 21.08, see Figure 4c.

**Joint training on multiple paths.** We found that training on a particular wait-$k$ path can generalize well to other paths. To avoid tuning $k_{\text{train}}$ to find the optimal path for each specific task, we consider jointly optimizing on multiple paths. Results in Figure 4 show that this joint optimization, for both architectures and on two datasets, manages to achieve comparable or better results than training on a single manually selected path.

**Experiments on the WMT15 De→En.** On WMT15 De→En we experiment with transformer *base* (comparable to [1, 2] and *big* (comparable to [18, 19]).[2] In Figure 5a we observe, as for IWSLT, that jointly training on multiple wait-$k$ paths outperforms training on a single path where the performance drops as we move away from $k_{\text{train}}$. The advantage of joint training with unidirectional encoders is confirmed in Figure 5b when comparing our results to *STACL* which trains separate bidirectional models for each decoding path with $k_{\text{train}} = k_{\text{eval}}$. Our models also outperform SL [2] that optimize dynamic agents with imitation learning (IL).

Both our *base* and *big multi-path* models match or improve the performance of the dynamic *MILk* [18] that requires training for each latency regime (each mark in the dotted curves is a different model) whereas our wait-$k$ model is simply evaluated with different values of $k_{\text{eval}}$. The more recent *MMA-H* and *MMA-IL* [19] adapting *MoChA* and *MILk* for Transformer models outperform *wait-k* models for AL $< 6$, but fail to optimize a medium lagging model.

## 5. Conclusion

In this paper, we demonstrated that unidirectional encoders for online MT achieve better translation qualities than bidirectional ones, with faster training and decoding. Moreover, we introduced joint training for *wait-k* decoders addressing the need to train a different model for each lagging value. Our models are trained end-to-end and, unlike conventional *wait-k*, can operate across the full spectrum of lagging with the quality increasing with the value of $k$. In low-resource settings, we found Pervasive Attention models to be competitive with Transformers for online translation. Our wait-$k$ models are state-of-the-art among deterministic online translation strategies, and provide a strong baseline for simultaneous translation with dynamic decoding.

[2] *MILk* is based on RNMT+ which outperforms TF *big* offline [30].

# 6. References


[1] M. Ma, L. Huang, H. Xiong, R. Zheng, K. Liu, B. Zheng, C. Zhang, Z. He, H. Liu, X. Li, H. Wu, and H. Wang, "STACL: Simultaneous translation with implicit anticipation and controllable latency using prefix-to-prefix framework," in *Proc. of ACL*, 2019.

[2] B. Zheng, R. Zheng, M. Ma, and L. Huang, "Simpler and faster learning of adaptive policies for simultaneous translation," in *Proc. of EMNLP*, 2019.

[3] C. Fügen, A. Waibel, and M. Kolss, "Simultaneous translation of lectures and speeches," *Machine translation*, 2007.

[4] M. Yarmohammadi, V. K. R. Sridhar, S. Bangalore, and B. Sankaran, "Incremental segmentation and decoding strategies for simultaneous translation," in *Proc. of NAACL-HLT*, 2013.

[5] H. He, A. Grissom II, J. Morgan, J. Boyd-Graber, and H. Daumé III, "Syntax-based rewriting for simultaneous machine translation," in *Proc. of EMNLP*, 2015.

[6] A. Grissom II, H. He, J. Boyd-Graber, J. Morgan, and H. Daumé III, "Don't until the final verb wait: Reinforcement learning for simultaneous machine translation," in *Proc. of EMNLP*, 2014.

[7] Y. Oda, G. Neubig, S. Sakti, T. Toda, and S. Nakamura, "Syntax-based simultaneous translation through prediction of unseen syntactic constituents," in *Proc. of ACL*, 2015.

[8] K. Cho and M. Esipova, "Can neural machine translation do simultaneous translation?" *ArXiv preprint*, 2016.

[9] N. Jaitly, Q. V. Le, O. Vinyals, I. Sutskever, D. Sussillo, and S. Bengio, "An online sequence-to-sequence model using partial conditioning," in *Proc. of NeurIPS*, 2016.

[10] F. Dalvi, N. Durrani, H. Sajjad, and S. Vogel, "Incremental decoding and training methods for simultaneous translation in neural machine translation," in *Proc. of NAACL-HLT*, 2018.

[11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. of NeurIPS*, 2017.

[12] Y. Luo, C.-C. Chiu, N. Jaitly, and I. Sutskever, "Learning online alignments with continuous rewards policy gradient," in *Proc. of ICASSP*, 2017.

[13] J. Gu, G. Neubig, K. Cho, and V. O. Li, "Learning to translate in real-time with neural machine translation," in *Proc. of EACL*, 2017.

[14] O. Press and N. A. Smith, "You may not need attention," *ArXiv preprint*, 2018.

[15] B. Zheng, R. Zheng, M. Ma, and L. Huang, "Simultaneous translation with flexible policy via restricted imitation learning," in *Proc. of ACL*, 2019.

[16] C. Raffel, M.-T. Luong, P. J. Liu, R. J. Weiss, and D. Eck, "Online and linear-time attention by enforcing monotonic alignments," in *Proc. of ICML*, 2017.

[17] C.-C. Chiu and C. Raffel, "Monotonic chunkwise attention," in *Proc. of ICLR*, 2018.

[18] N. Arivazhagan, C. Cherry, W. Macherey, C.-C. Chiu, S. Yavuz, R. Pang, W. Li, and C. Raffel, "Monotonic infinite lookback attention for simultaneous machine translation," in *Proc. of ACL*, 2019.

[19] X. Ma, J. Pino, J. Cross, L. Puzon, and J. Gu, "Monotonic multihead attention," in *Proc. of ICLR*, 2020.

[20] J. Niehues, T. S. Nguyen, E. Cho, T.-L. Ha, K. Kilgour, M. Müller, M. Sperber, S. Stüker, and A. Waibel, "Dynamic transcription for low-latency speech translation." in *Proc. of INTERSPEECH*, 2016.

[21] N. Arivazhagan, C. Cherry, W. Macherey, and G. Foster, "Re-translation versus streaming for simultaneous translation," in *Proc. of IWSLT*, 2020.

[22] R. Zheng, M. Ma, B. Zheng, K. Liu, and L. Huang, "Opportunistic decoding with timely correction for simultaneous translation," in *Proc. of ACL*, 2020.

[23] M. Elbayad, L. Besacier, and J. Verbeek, "Pervasive attention: 2D convolutional neural networks for sequence-to-sequence prediction," in *Proc. of CoNLL*, 2018.

[24] M. Cettolo, J. Niehues, S. Stüker, L. Bentivogli, and M. Federico, "Report on the 11th IWSLT evaluation campaign," in *Proc. of IWSLT*, 2014.

[25] M.-T. Luong and C. D. Manning, "Stanford neural machine translation systems for spoken language domains," in *Proc. of IWSLT*, 2015.

[26] S. Edunov, M. Ott, M. Auli, D. Grangier, and M. Ranzato, "Classical structured prediction losses for sequence to sequence learning," in *Proc. of NAACL-HLT*, 2018.

[27] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in *Proc. of ACL*, 2016.

[28] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: a method for automatic evaluation of machine translation," in *Proc. of ACL*, 2002.

[29] A. Alinejad, M. Siahbani, and A. Sarkar, "Prediction improves simultaneous neural machine translation," in *Proc. of EMNLP*, 2018.

[30] M. Chen, O. Firat, A. Bapna, M. Johnson, W. Macherey, G. Foster, L. Jones, M. Schuster, N. Shazeer, N. Parmar, A. Vaswani, J. Uszkoreit, L. Kaiser, Z. Chen, Y. Wu, and M. Hughes, "The best of both worlds: Combining recent advances in neural machine translation," in *Proc. of ACL*, 2018.