KIT's IWSLT 2020 SLT Translation System

Ngoc-Quan Pham, Felix Schneider, Tuan-Nam Nguyen, Thanh-Le Ha,
Thai-Son Nguyen, Maximilian Awiszus, Sebastian Stüker, Alexander Waibel
Karlsruhe Institute of Technology
firstname.lastname@kit.edu

Abstract

This paper describes KIT's submissions to the IWSLT2020 Speech Translation evaluation campaign. We first participate in the simultaneous translation task, in which our simultaneous models are Transformer-based and can be efficiently trained to obtain low latency with minimized compromise in quality. On the offline speech translation task, we applied our new Speech Transformer architecture to end-to-end speech translation. The obtained model can provide translation quality which is competitive to a complicated cascade. The latter still has the upper hand, thanks to the ability to transparently access to the transcription, and resegment the inputs to avoid fragmentation.

1 Introduction

The Karlsruhe Institute of Technology (KIT) participated in the IWSLT 2020 Evaluation Campaign (Ansari et al., 2020) in two main tracks: Offline Speech Translation task (SLT) and Simultaneous Text Translation. Our highlight s the proposal of a novel method for training simultaneous translation models, with the Adaptive Computation Time technique (Graves, 2016) incorporated to the Transformer models (Vaswani et al., 2017). On the other hand, the end-to-end speech translation models have observed a single deep Speech Transformer (Pham et al., 2019b) approaching the performance of a heavily powered cascade. The latter, however, is more transparent because of visible inputs and outputs to each components. It is still the dominant approach, thanks to the segmentation module that adds punctuations and sentence boundaries, so the MT models do not suffer from fragmentation.

2 Data

The overall data that the project employed can be divided into two main sections: speech and text corpora.

Speech Corpora. We gathered the allowed training data included MuST-C and Speech-Translation TED Talks containing both parallel data for audio to English and German. The TEDLIUM3 and the Mozilla Common Voice data are speech recognition-specific. Furthermore we also considered the How2 dataset (the Portuguese translation is ignored). The data is further cleaned with ASR models (the details are unveiled in Section 4.4) to obtain the training time as shown in Table 1.

Table 1: Speech Training data

Data	Segments	Total time
MuST-C	229K	408h
Speech Translation	142K	160h
TEDLIUM	264K	415h
Common Voice	854K	1490h
How2	217K	360h

Text Corpora. We collected the text parallel training data as presented in Table 2.

Table 2: Text Training Data

Dataset	Sentences
TED Talks (TED)	220K
Europarl (EPPS)	2.2MK
CommonCrawl	2.1M
Rapid	1.21M
ParaCrawl	25.1M
OpenSubtitles	12.6M
WikiTitle	423K
Back-translated News	26M

Proceedings of the 17th International Conference on Spoken Language Translation (IWSLT), pages 55–61
July 9-10, 2020.
https://doi.org/10.18653/v1/P17

3 Simultaneous Speech Translation

For simultaneous speech translation, we deploy a novel model on the text-to-text task based on Adaptive Computation Time (ACT, Graves (2016)). At each decoder step, the model makes a decision on whether to READ another input token or to WRITE an output token (c. f. Raffel et al. (2017)).

In our case, these decisions are made by a mechanism based on ACT: At each decoder step, we calculate a probability distribution over the encoder timesteps, representing the prediction where the decoder should halt and WRITE an output. Specifically, for decoder step i, we calculate:

$$p_i^n = \sigma(\textsc{Energy}(s_i^n)) \tag{1}$$

$$N(i) = \min\{n' : \sum_{n=1}^{n'} p_i^n \geq 1 - \epsilon\} \tag{2}$$

$$R(i) = 1 - \sum_{n=1}^{N(i)-1} p_i^n \tag{3}$$

$$\alpha_i^n = \begin{cases} R(i) & \text{if } n = N(i) \\ p_i^n & \text{otherwise} \end{cases} \tag{4}$$

It follows from the definition that α_i is a probability distribution. We use this distribution along with the attention mechanism from Arivazhagan et al. (2019) to calculate the encoder-decoder attention. In order to incentivise the model to keep the delays short, we employ the *ponder loss* in addition to the usual cross-entropy:

$$L(\theta) = -\sum_{(\mathrm{x},\mathrm{y})} \log p(\mathrm{y}|\mathrm{x};\theta) + \lambda \mathcal{C}(\mathrm{n}) \tag{5}$$

$$\mathcal{C} = \sum_{i=1}^{|\mathrm{x}|} N(i) + R(i) \tag{6}$$

For more information on the ponder loss, see (Graves, 2016). By varying the parameter λ, we can produce systems with different latency regimes. However, each model produces many different latency-quality tradeoffs during training.

We use sentencepiece (Kudo and Richardson, 2018) to create a shared 37000 word BPE dictionary for source and target. We then train an offline transformer (Vaswani et al., 2017) model with relative self-attention (Dai et al., 2019). Based on this, we train several ACT models with λ varying from 0.15 to 0.7. For all models, we use the Adam optimizer (Kingma and Ba, 2015). We train the offline model for 200 000 steps, varying the learning rate from $2.5 \cdot 10^{-4}$ to 0 with a cosine schedule, then train each of the simultaneous models for 1000 steps with initializing parameters from the offline model. All models use the transformer "base" configuration (layer size 512, feed-forward size 2048, 8 attention heads, 6 layers in encoder and decoder). Because the evaluation primarily measures delay in terms of tokens, not time, we could have used a larger model, but we decided to choose our model for a more realistic scenario where evaluation time is an important factor.

4 Offline Speech Translation

We participate to the offline speech translation task using two different approaches: cascade and end-to-end. In the cascade, the audio inputs are fed into our Speech Recognition component (ASR - Section 4.1), then the outputs will go through a Segmentation module (Section 4.2) to have well-formed inputs prior to our Machine Translation module (MT - Section 4.3). The outputs of our MT are the final outputs of the cascade system. On the other hand, the end-to-end approach, as its name suggests, performs trainings for a single model from the English audio inputs to produce text outputs in German (Section 4.4).

4.1 Speech Recognition

Data preparation and Segmentation tool We used two different training data sets for this evaluation. Having collected all audios from the TED-LIUM and How2 corpora provided by the organizer, we then generated 40 features of Mel-filterbank coefficients for ASR training models using Janus Recognition Toolkit. We use SentencePiece toolkit (Kudo and Richardson, 2018) to train and create 4000 different byte-pair-encoding (BPE) for all models. After that, the WerRTCVAD toolkit (Wiseman, 2016) was used to segment the audio in two unsegmented datasets.

Model We only focus on sequence-to-sequence ASR models, which are based on two different network architectures: The long short-term memory (LSTM) and the Transformer. Our LSTM-based models consist of 6 bidirectional layers of 1024 units for the encoder and 2 unidirectional layers for the decoder (Nguyen et al., 2019). Our transformer-based models presented in (Pham et al., 2019b) consist of 32 blocks for the encoder and 12 blocks for the decoder. Inputs to the LSTM model are

Mel-filterbank features with 40 coefficients. For the Transformer model, we concatenated 4 consecutive features, then combined them with the position information and put them to the self-attention blocks. For LSTM regularization, we applied the dropout rate 0.35 in all LSTM layers, and the embedding dropout rate 0.35 for LSTM. For Transformer regularization, we applied dropout of rate is 0.5 and Stochastic Layers in our models (Pham et al., 2019b).

4.2 Segmentation

Automatic speech recognition (ASR) systems typically do not generate punctuation marks or reliable casing. Using the raw output of these systems as input to MT causes a performance drop due to mismatched train and test conditions. To create segments and better match typical MT training conditions, we use a monolingual NMT system to add sentence boundaries, insert proper punctuation, and add case where appropriate before translating [15]. The idea of the monolingual machine translation system is to translate from lower-cased, without-punctuation text into text with case information and punctuation.

This year, we reuse the segmentation model from (Pham et al., 2019a). We ultilize a transformer-based NMT system to to translate from an English sentence into a sequence of punctuation and casing notations. The training data for that are EPPS, NC and a filtered version of the ParaCrawl corpus. Then, we fine-tune the model on the TED corpus. For more details, please refer to (Pham et al., 2019a).

4.3 Machine Translation

Data Preparation. This year, we use an approximating of 70 millions sentence pairs, coming from TED, EPPS, NC, CommonCrawl, ParaCrawl, Rapid and OpenSubtitles corpora, including around 26 millions back-translation sentence pairs. The data are applied tokenization and smart-casing using the Moses scripts. Furthermore, we segment words into subword units using BPE method (Sennrich et al., 2016). The smartcasing and BPE model are trained on what we call clean datasets (TED, EPPS, NC and CommonCrawl), with the number of BPE merging operation of 40000, jointly learned from English and German sides.

Modeling and Training. Basically our translation system employs Transformer-based encoder-decoder model (Vaswani et al., 2017). Our model comprises of a 12-layer encoder and 12-layer decoder, in which each layer' size is 1024, while the the inner size of feed-forward network inside each layer is 4096. The notable different of our translation model compared to the original Transformer lays on the attention blocks. We implemented Relative Attention following the work of (Dai et al., 2019). The self-attention layers take into account the relative distances between the states instead of using an absolute position encoding scheme by adding the position vectors to the word embeddings. For the encoder, in order to distinguish the two directions of attention (forward and backward), we use negative distances for forward, and positive distances for backward. Each attention block is multi-head attention with 16 heads. We also employ label smoothing in order to regularize the cross-entropy loss. Since we share the vocabularies of the source and target, we are able to tie the embedding weights of the encoder and decoder layers.

Since we utilize a large amount of data, we set dropout at 0.1 and trained for 300000 steps. We use the learning rate schedule with 8000 steps of warming up before linearly scaling down afterwards. We then average five best models according to perplexity on a validation set. We denote this as *Large* configuration.

Domain Adaptation. From the *Large* model, we perform fine-tuning on the TED data, which we consider the in-domain data for the task. In addition to the original TED data, we introduce some noises into a portion of that data and mix this noised data to the original one, then do the fine-tuning. The noises are simply produced by duplicating or deleting n words in some random positions conforming to some distributions[1] and inserting or deleting a punctuation from the original sentence.

The main differences between the *Fine-tuning* configuration and the *Large* configuration is that we apply more strict regularizations, since the fine-tuning data is significantly smaller. Particularly, the dropout is now 0.3, word dropout (Gal and Ghahramani, 2016) is at 0.1 and we also implement switchout (Wang et al., 2018) with the rate of 0.95. Switchout is especially useful when we want to

[1]The probability of whether the noise is introduced is $p_{w\_noise} = 0.7$. The distribution of duplicating and deleting a word is $p_{w\_manipulate} = (0.6, 0.4)$. The distribution of how many words ranging from 1 to 3 ($n = 1, 2, 3$) is $p_{w\_num} = (0.6, 0.35, 0.05)$. Those distributions are deliberately chosen after we looked into the outputs of a validation set from our ASR.

simulate the noisy conditions of speech translation, in which the automatic transcripts often contain errors. We train one fine-tuned model from the original TED, and another model with the mix of TED and noised TED with the same *Fine-tuning* configuration. Both of them are trained for 2800 steps with the learning rate of 2 and the same warm-up schedule as before, then again five best models of each are averaged. Finally, we ensemble these two averaged models to be our submitted system.

4.4 End-to-End Model

Corpora The main source of parallel data comes from the MUST-C corpus (Di Gangi et al., 2019b) (only the English-German part) and the Speech Translation data provided by the organizer. The speech features are regenerated with the in-house Janus Recognition Toolkit.

In order to utilize the English audio utterances without aligned German translations, we generate the *synthetic* translations for the available TED Talks in the TEDLIUM dataset and furthermore the large Mozilla Common Voice (CV). Even though these datasets contain their aligned transcriptions, it is still challenging to generate the translations accordingly. The audio segmentation process in the data collection process does not necessarily force the utterances to be encapsulated within sentence borders, and also the transcriptions are often lower-cased and stripped off punctuations. As a result, we used the Transformer-based punctuation model (Cho et al., 2017) to generate punctuations for each utterance. The translation models are trained with the WMT 2018 dataset combined with OpenSubtitles as in (Pham et al., 2019a) (which still satisfy the "constrained" conditions for the evaluation campaign). It is notable that, even though we can generate better translations by using the window technique as in (Cho et al., 2017) to have better sentence boundaries, such method breaks the alignment with audio utterances. Therefore, the generated translation can be incomplete or noisy compared to the translation acquired from the available parallel corpora.

The data is further cleaned from the potential errors (in alignment). These errors can be detected by first training an ASR model, that we based on the Transformer-based ASR (Pham et al., 2019b), and then decoding the audio inputs. We then compute the GLEU score (Wu et al., 2016) between the generated and the annotated transcripts. With the threshold of 0.67, we removed the utterances with the lower scores, and end up with the training SLT data as in Table 1

During training, the validation data is the Development set of the MuST-C corpus. The reason is that the SLT testsets often do not have the aligned audio and translation, while training end-to-end models often rely on perplexity for early stopping.

Modeling The main architecture is the deep Transformer (Vaswani et al., 2017) with stochastic layers (Pham et al., 2019b). Each model has 32 encoder layers and 12 decoder layers, and they are randomly dropped in training according to the linear schedule presented in the original work, with the top layer has the highest dropout rate $p = 0.5$.

In order to make training stable, we initialized the encoder of the network with the ASR model with the same configuration (so that the parameters can be transferred). We have two intermediate ASR models for this purpose, one is trained on top of TEDLIUM and MuST-C combined, and one learns from the combination of CV, TEDLIUM and MuST-C, serving two different data settings presented in the next section.

With the initialized encoder, the networks can be trained with an aggressive learning rate with 2048 warm-up rate. Label-smoothing and dropout rates are set at 0.1 and 0.25 respectively for all models. Furthermore, all speech inputs are augmented with spectral augmentation (Park et al., 2019; Bahar et al., 2019). All models are trained for 100000 steps, each consists of accumulated 12000 target tokens.

Finally, in order to alleviate the weaknesses of the Transformer models when it comes to dealing with long inputs, such as speech signals, we incorporated the relative position encoding (Dai et al., 2019) into our Transformers. The self-attention layers use the relative distance between states to compute their similarity functions, instead of relying on an absolute position encoding scheme which is vulnerable for this task.

Speech segmentation A big challenge of end-to-end speech translation is audio segmentation, which could harm the performance significantly. The model does not have the ability to re-segment the audio inputs compared to the cascade. Here we simply use the WerRTCVAD toolkit (Wiseman, 2016) to provide the translation model with segments.

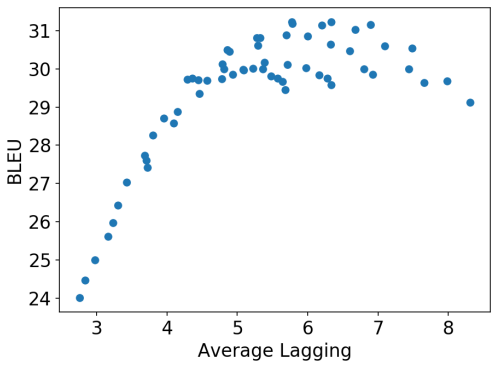

Figure 1: Quality-latency tradeoffs of various checkpoints on the *MUST-C* test set. Metrics are determined by the official evaluation script.

Model	**BLEU**	**AL**	**DAL**	**AP**
Offline	32.9	18.6	18.6	1.00
High Latency	31.5	6.3	7.2	0.81
Medium Latency	31.4	6.0	6.9	0.80
Low Latency	25.0	3.0	3.8	0.66

Table 3: Performance of our submitted models on the *MUST-C* test set.

4.5 Experimental Results

4.5.1 Simultaneous Translation

We evaluate our model on the *MUST-C* test set, `tst-COMMON`. As each model goes through many different quality-latency trade-offs during training, we evaluated a large number of checkpoints before choosing three models for the low-latency (AL $\leq$ 3), medium latency (AL $\leq$ 6) and high-latency (AL $\leq$ 12) categories. Figure 1 shows all evaluated models on a quality-latency graph. The performance peaks at around 6 Average Lagging, convenient for the medium latency category. Higher latency models can reach similar performance with longer training (the shown models are trained for 1000 steps or less), but only barely exceed the peak at 6 Average Lagging, indicating that that is this model's ideal maximum latency. Table 3 shows the performance of our models on the *MUST-C* test set.

4.5.2 Cascade Offline Speech Translation

Speech Recognition. We tested our ASR systems on two datasets, tst2015 and How2 evaluation set. The ensemble of LSTM-based and Transformer-based sequence-to-sequence model provide the best results, which are 4.1 and 10.6 WERs respectively for two evaluation sets.

Data	tst2015	How2
Transformer-based	6.5	12.5
LSTM-based	4.5	11.5
Ensemble	4.1	10.6

Table 4: WER on tst2015 and How2 sets

Machine Translation. The SLT results on tst2014 are reported in Table 5. By fine-tuning on TED and introducing noises, we are able to gain an improvements of 0.64 BLEU points from the model which is already better than the best model of last year's evaluation.

Table 5: Cascade SLT result on tst2014 (En-De)

System	tst2014 (BLEU)
Large	25.46
TED Finetune	25.90
Noised TED Finetune	26.03
Ensemble	26.10

4.5.3 End-to-end Offline Speech Translation

We tested three different data conditions. The **Small** setup uses only MuST-C as the data. The **Medium** model is trained on MuST-C, Speech-Translation and TEDLIUM. Finally the **Large** one is trained on all data we have including the Mozilla CV. This naming convention only indicates the data size, while the model size and training procedure is kept the same across all settings.

We tested the models on two different setups. The tst-COMMON is provided with the MuST-C and it is not necessary to resegment the translation afterwards to match the translation reference. On the other hand, the tst2014 set requires this step, because depending on the segmentation, the hypothesis and reference can have different alignment. All of the evaluations were performed with cased BLEU scores.

Table 6: SLT BLEU scores on MuST-C test set and tst2014 (En-De)

Data	MuST-C tst-COMMON	tst2014
Small	25.2	-
Medium	30.6	25.4
Large	28.0	23.2
Large+Adapt	28.1	23.3

We obtained the results as in 6. Our **Small** setup has achieved 25.2 BLEU scores on tst-COMMON which already outperformed the best published results on this test set (Di Gangi et al., 2019a). Adding the Speech-Translation and the TEDLIUM data helped us to further improve the result to 30.6. On the other hand, the **Large** setup suffered a 2 BLEU point loss compared to the **Medium** counterpart. This could be the result of the difference in terms of domain between the Mozilla CV and TED Talks, as well as the recording environment and the translation quality obtained with the MT models. However, even adapting these models on the MuST-C and Speech-Translation corpora cannot further improve this setup.

On the tst2014 test set, our end-to-end models achieved the best result with 25.4 BLEU scores, which is closely competitive with the best system in IWSLT 2019 (Pham et al., 2019a), which was 25.7. This indicates that a deep Transformer network can potentially reach the performance of a strong cascade pipeline with mutliple models. Simplicity is the advantage of this setup, however, when the output can be obtained directly after the feature generation step, instead of having several components which have different input and output formats.

5 Conclusion

At the IWSLT2020 evaluation campaign, we first presented a novel simultaneous model that can efficiently learn to wait and translate using ACT technique. Afterwards, we built two systems for offline speech translation, namely a cascade and an end-to-end model using Deep Transformer networks. We showed that the end-to-end model can rival even the best cascade in challenging speech translation tests.

Acknowledgments

The work leading to these results has received funding from the European Union under grant agreement n°825460 and the Federal Ministry of Education and Research (Germany)/DLR Projektträger Bereich Gesundheit under grant agreement n° 01EF1803B.

References

Ebrahim Ansari, Nguyen Bach, Ondrej Bojar, Rol dano Cattoni, Marcello Federico, Christian Federmann, Jiatao Gu, Fei Huang, Ajay Nagesh, Matteo Negri, Jan Niehues, Elizabeth Salesky, Sebastian Stüker, and Marco Turchi. 2020. Findings of the IWSLT 2020 Evaluation Campaign. In *Proceedings of the 17th International Conference on Spoken Language Translation (IWSLT 2020)*, Seattle, USA.

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. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1313–1323.

Parnia Bahar, Albert Zeyer, Ralf Schlüter, and Hermann Ney. 2019. On using specaugment for end-to-end speech translation. *arXiv preprint arXiv:1911.08876*.

Eunah Cho, Jan Niehues, and Alex Waibel. 2017. NMT-based segmentation and punctuation insertion for real-time spoken language translation. In *Interspeech 2017*. ISCA.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*.

M Di Gangi, Matteo Negri, Viet Nhat Nguyen, Amirhossein Tebbifakhr, and Marco Turchi. 2019a. Data augmentation for end-to-end speech translation: FBK@IWSLT'19. In *Proceedings of the 16th International Workshop on Spoken Language Translation (IWSLT)*.

Mattia A. Di Gangi, Roldano Cattoni, Luisa Bentivogli, Matteo Negri, and Marco Turchi. 2019b. MuST-C: a Multilingual Speech Translation Corpus. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems 29 (NIPS)*.

Alex Graves. 2016. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*.

Diederik Kingma and Jimmy Ba. 2015. Adam: a method for stochastic optimization (2014). *arXiv preprint arXiv:1412.6980*, 15.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71.

Thai-Son Nguyen, Sebastian Stueker, Jan Niehues, and Alex Waibel. 2019. Improving sequence-to-sequence speech recognition training with on-the-fly data augmentation. *arXiv preprint arXiv:1910.13296*.

Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le. 2019. Specaugment: A simple data augmentation method for automatic speech recognition. *arXiv preprint arXiv:1904.08779*.

Ngoc-Quan Pham, Thai-Son Nguyen, Thanh-Le Ha, Juan Hussain, Felix Schneider, Jan Niehues, Sebastian Stüker, and Alexander Waibel. 2019a. The iwslt 2019 kit speech translation system.

Ngoc-Quan Pham, Thai-Son Nguyen, Jan Niehues, Markus Muller, and Alex Waibel. 2019b. Very deep self-attention networks for end-to-end speech recognition. *arXiv preprint arXiv:1904.13377*.

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

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Xinyi Wang, Hieu Pham, Zihang Dai, and Graham Neubig. 2018. SwitchOut: an efficient data augmentation algorithm for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 856–861, Brussels, Belgium. Association for Computational Linguistics.

John Wiseman. 2016. python-webrtcvad. `https://github.com/wiseman/py-webrtcvad`.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.