The JHU/KyotoU Speech Translation System for IWSLT 2018

Hirofumi Inaguma[†]*, Xuan Zhang, Zhiqi Wang, Adithya Renduchintala, Shinji Watanabe, Kevin Duh*

Center for Language and Speech Processing, Johns Hopkins University, Baltimore, USA
[†]Graduate School of Informatics, Kyoto University, Japan
{xuanzhang,zwang132,adithya.renduchintala,shinjiw}@jhu.edu
inaguma@sap.ist.i.kyoto-u.ac.jp, kevinduh@cs.jhu.edu

Abstract

This paper describes the Johns Hopkins University (JHU) and Kyoto University submissions to the Speech Translation evaluation campaign at IWSLT2018. Our end-to-end speech translation systems are based on ESPnet and implements an attention-based encoder-decoder model. As comparison, we also experiment with a pipeline system that uses independent neural network systems for both the speech transcription and text translation components. We find that a transfer learning approach that bootstraps the end-to-end speech translation system with speech transcription system's parameters is important for training on small datasets.

1. Introduction

We report on our efforts on the IWSLT 2018 Speech Translation task. The goal of the 2018 task is to build and evaluate English-to-German speech translation systems on the domain of lectures and TED talks. We build two systems:

- Pipeline System: English (EN) speech transcription system using a joint CTC-attention model (Section 3.1), followed by a English-to-German (EN-DE) text translation system using a RNN-based sequence-to-sequence model (Section 3.3).
- End-to-End System: English-to-German (EN-DE) speech translation system using an RNN-based sequence-to-sequence model transferred from the joint CTC-attention model (Section 4).

The main challenge is to develop end-to-end neural systems that are trainable given the small amount of data (of English speech matched to German text). We find that bootstrapping the end-to-end system with the parameters of an English-only speech transcription system (i.e. ASR of English speech to English text) was helpful.

Generally, we are interested in comparing the relative merits of end-to-end vs. pipeline approaches. Currently, our pipeline system outperforms the end-to-end system, even when trained on the same number of utterances, suggesting that there is much room for future work in end-to-end models.

[†] Work carried out as a visiting scholar at JHU.

2. Data

We build our systems on the following provided corpora:

1. Speech-Translation TED corpus (ST TED): This data contains English speech (EN-s), the corresponding English transcription (EN-t), as well as the German translation (DE-t). We use this to train both pipeline and end-to-end systems. In particular, (EN-s,EN-t) is used to train the pipeline's speech transcription component (Section 3.1); (EN-t, DE-t) is used to train the pipeline's text translation component (Section 3.3); and (EN-s, DE-t) is used to train the end-to-end sytem (Section 4).
2. TED LIUM corpus (TEDLIUM2): This data contains English speech (EN-s) and their English transcription (EN-t). We use this as additional data to train the pipeline system's speech transcription component, which is also used to initialize the end-to-end system.
3. WMT 2018 data, filtered to the TED domain using Moore-Lewis data selection [1] (WMT-Filtered): We trained 5-gram Engish language models on TED (LM_{TED}) and a random sample of the WMT data (LM_{WMT}), then selected the top 1 million WMT bitext according to the perplexity difference between LM_{TED} and LM_{WMT}. Finally, we filtered all sentences that were longer than 100 tokens or had an out-of-vocabulary rate (with respect ST TED dictionary) of 10% or larger. This is used to augment the training data for the pipeline's text translation component.

For data preprocessing of transcriptions and translations in all languages, we normalized punctuation and performed tokenization using the Moses scripts[1]. For both the pipeline's speech transcription and the end-to-end speech translation, we used a fixed vocabulary of 5k or 10k wordpieces, which were composed from characters to words and generated using sentencepiece[2]. We used the same dictionary including both EN and DE wordpieces to capture the common words in both languages.

[1] normalize-punctuation.perl and tokenizer.perl in https://github.com/moses-smt/mosesdecoder
[2] https://github.com/google/sentencepiece

For the pipeline's text translation component, we experimented with different kinds of subword units. For simplicity, we did not use any truecase models for any systems and worked directly with the natural casing. Table 1 shows the data sizes in each corpus.

For feature extraction for speech transcription and translation, we extracted 80-channel log-mel filterbank outputs with 3-dimensional pitch features computed with a 25ms window and shifted every 10 ms using Kaldi [2]. The features were normalized by the mean and the standard deviation on the whole training set (excluding our development set). We removed utterances having more than 3000 frames or more than 400 characters due to the GPU memory capacity.

corpus	#utterance	speech datasize
Speech-Translation TED	166,214	271 hours
TEDLIUM2	258,943	210 hours
WMT-Filtered	988,697	-

Table 1: Data size in each corpus.

3. Pipeline System

3.1. Speech Transcription Component

In this section, we briefly describe the joint CTC-attention framework for the speech transcription (i.e. ASR) component. Let $\mathbf{x} = (x_1, \ldots, x_T)$ be T-length acoustic features and $\mathbf{y} = (y_1, \ldots, y_U)$ be the corresponding U-length target sentence in the same language as $\mathbf{x}$.

3.1.1. Connectionist Temporal Classification (CTC)

Connectionist Temporal Classification (CTC) [3] is a latent variable model which directly maps the input sequence into the output sequence of shorter length. To compensate the differences of sequence lengths, CTC introduces an additional "blank" symbol. The CTC loss function is defined as the summation of negative log probabilities of all possible paths mapped from ground truth labels interleaved with blank labels.

$$\begin{aligned} L_{ctc} &= -\ln P(\mathbf{y}|\mathbf{x}) \\ &= -\ln \sum_{\pi \in B^{-1}(\mathbf{y})} P(\pi|\mathbf{x}) \end{aligned}$$

where π represents a CTC path, and B represents a collapse function which maps all the CTC paths into the unique ground truth labels $\mathbf{y}$ by removing all blank labels. Based on the conditional independence assumption, posterior probabilities $P(\pi|\mathbf{x})$ is factorized frame by frame as follows:

$$P(\pi|\mathbf{x}) = \prod_{t=1}^{T} P(\pi_t|h_t)$$

where h_t represents an activation of the top layer of the encoder. $P(\pi|\mathbf{x})$ is effectively calculated by dynamic programming.

3.1.2. Attention-based encoder-decoder

Attention-based encoder-decoder [4, 5] is another sequence labeling model which directly predicts output sequences. Unlike the CTC framework, this approach does not make any conditional independence assumptions, where the model predicts each token conditioned on all previous tokens.

$$P(\mathbf{y}|\mathbf{x}) = \prod_{u=1}^{U} P(y_u|y_1, \ldots, y_{u-1}, \mathbf{x})$$

Attention-based encoder-decoder model consists of two modules: the encoder and decoder. The encoder network maps input features $\mathbf{x}$ into high-level distributed representation $\mathbf{h}$, and the decoder network picks up a portion of $\mathbf{h}$ with a scoring function given encoder and decoder hidden states, which is called the attention mechanism. We used the location-aware scoring function, which takes previous attention weights into account. The loss function is designed as the negative log probabilities as follows:

$$L_{att} = -\ln P(\mathbf{y}|\mathbf{x})$$

3.1.3. Joint CTC-attention

We introduce the multitask learning (MTL) framework with the CTC objective in the training of the attention-based encoder-decoder model [6]. This approach has two advantages: 1) it encourages monotonic alignments in the encoder network, which leads to fast convergence and removes inappropriate alignments in long sequences, 2) it leads to sequence-level optimization. The loss function of the joint CTC-attention framework is designed as an interporation of L_{ctc} and L_{att} with a tunable parameter λ ($0 \leq \lambda \leq 1$):

$$L_{mtl} = \lambda L_{ctc}(\mathbf{y}|\mathbf{x}) + (1-\lambda) L_{att}(\mathbf{y}|\mathbf{x})$$

In addition, scores from CTC outputs are taken into account in the beam search decoding of the attention-based model during the inference stage [7, 8]. Because CTC is frame-synchronous, hyper-parameters tuning such as length penalty and coverage penalty are not neccesary any more in order to prune inappropriate hypotheses.

3.2. Evaluation of Speech Transcription Component

Preprocessing: We used the Speech-Translation TED corpus augmented with TED LIUM corpus, totaling 481h. With regard to the official development sets provided by the IWSLT organizers (*dev2010*, *tst2013* etc.), there is no segmentation information of the start and end time of utterances. Therefore, we sampled 4k utterances from the Speech-Translation TED corpus as the validation set, and

removed them from the original training data. For evaluation, we segmented each audio file in the development sets with the LIUM SpkDiarization tool [9] first, then performed MWER segmentation with the toolkit from RWTH [10] as in the baseline implementation provided by organizers[3].

Architecture: We built end-to-end ASR models with the ESPnet toolkit [11] with a pytorch backend [12]. For the encoder part, we used 2 CNN blocks, where each blocks composed of 2-layer CNNs with max-pooling, followed by 3 or 5-layers of 1024 dimensional bidirectional LSTM [13]. The max-polling operation in each CNN block performs subsampling for every two frames, resulting in 4-fold time reduction in the encoder. For the decoder part, we used 2-layers of 1024 dimensional LSTM. We did not conduct regularization such as dropout, label smoothing [14, 15], scheduled sampling [16] for speech transcription.

Optimization: Our systems were optimized with the AdaDelta algorithm with epsilon annealing for 15 epochs. The weight for CTC loss λ was empirically set to 0.5.

Decoding: We conducted beam search decoding with beam width 20. LSTM language model of 2 layers with 650 hidden units trained on the same parallel corpus was used.

Results: Results for the TEDLIUM2 corpus are shown in Table 2. 5k wordpiece units are always better than 10k in this corpus. We also conformed the consistent improvements with deeper encoders (3 layers $\rightarrow$ 5 layers). Results for the official development sets are shown in Table 3. As in Table 2, although 5k units are better than 10k units, we cannot observe improvements by adding encoder layers. The results of Table 3 were obtained through audio segmentation by the LIUM SpkDiarization tool and utterance matching by the RWTH MWER tool, and we expect that such complex systems make it difficult to reflect the ASR performance improvement achieved with the Table 2 condition to the final system.

#unit	#layer	dev	test
10k	3	13.8	12.3
5k	3	12.1	11.1
10k	5	13.3	12.5
5k	5	**11.6**	**10.7**

Table 2: Word error rate (WER) evaluated on the TEDLIUM2 corpus. **#unit** represents the number of untis in the softmax layer. **#layer** represents the number of BiLSTM layers following CNN layers in the encoder network.

3.3. Text Translation Component

Preprocessing: We built neural machine translation (NMT) systems for the English-German text translation component

[3]https://github.com/isl-mt/SLT.KIT

#unit	#layer	dev2010	test2010	test2013	test2014	test2015
10k	3	28.2	29.5	31.9	32.6	46.5
5k	3	**25.6**	**27.7**	**30.6**	**31.1**	**44.4**
10k	5	28.8	31.2	33.1	34.5	45.6
5k	5	27.4	30.8	32.0	33.7	47.9

Table 3: Word error rate (WER) evaluated on the official development sets.

of our pipeline system. These systems were trained on the ST TED corpus, with English manual transcript for speech recognition on the source side and corresponding German translation on the target. Training data were tokenized and split into subwords using Byte Pair Encoding (BPE) [17]. We set the number of BPE merge operations to be 20k for the source side — same for the target side. The validation set used for early stopping consists of around 4k utterances, and they were randomly sampled from the corpus.

Architecture: The attention-based NMT models consist of two components: an encoder network, which is a recurrent neural network (RNN), that provides a representation of the input sentence, and a decoder network, which is also a RNN, that generates translation based on the input context with attention mechanism [5, 18] applied.

We trained our NMT systems with Sockeye [19]. In the model we used based on hyper-parameter tuning, the encoder and decoder both had 2 layers with 512 LSTM hidden units on each layer and we applied dot product attention for RNN decoders. Both the source and target embedding vectors were set to 512. We used word-count based batch of size 4096 words and maximum sequence length 100. For regularization, the RNN inputs and states dropout rates for both the encoder and the decoder were set to 0.1.

Optimization: Our systems employed the Adam optimizer to reduce the cross-entropy loss with an initial learning rate 0.0005. We made a checkpoint after every 2000 batch updates, and if the model had not improved in perplexity on the validation data for more than 8 checkpoints, we would perform early stopping for the training process. In general, it takes around 50 epochs (about 10 hours) for the model to converge.

#BPE merge ops	20k	30k	40k	50k
avg dev BLEU	23.83	24.18	24.28	23.77

Table 4: Effect of different number of BPE merge operations on average BLEU score on development sets. The initial learning rate was set to 0.0007.

Hyper-parameter Tuning: Hyper-parameters were tuned based on systems' average decoding performance (BLEU score) on *dev2010*, *tst2010*, *tst2013*, *tst2014* and *tst2015* set. We searched the number of BPE merge operations from 20k, 30k, 40k and 50k, word embedding size and

system	dev2010	tst2010	tst2013	tst2014	tst2015
Manual: NMT (ST TED) on EN reference	23.96	27.54	26.23	22.30	25.07
Pipeline: NMT (ST TED) on ASR output	15.47	20.54	16.68	14.35	16.21
Manual: NMT (ST TED + WMT-Filtered) on EN reference	28.07	30.59	29.47	26.04	26.70

Table 5: BLEU comparison of NMT translating English reference (Manual) or ASR ouput (Pipeline). BLEU scores are evaluated on development sets using multi-bleu.perl with the Moses tokenization.

the number of RNN hidden units from 512 and 1024, batch size from 4096 and 6000, initial learning rate from 0.0002 to 0.0007, dropout probability from 0.1 and 0.2[4].

When searching for a good hyper-parameter configuration, we found that a more complex model, in terms of RNN hidden size, was not necessarily needed to get better performance on this corpus: when we increased the number of RNN hidden units from 512 to 1024, the average BLEU score dropped from 24.71 to 24. Another interesting finding was that with other hyper-parameters fixed, when the number of BPE operations increased, the BLEU score on the development data tended to first go up and then decrease (see Table 4). Finally, the initial learning rate turned out to be an important hyper-parameter to tune. For example, we got 23.44 BLEU with initial learning rate 0.0003, but 24.18 BLEU with 0.0007.

3.4. Evaluation of Pipeline System

We show the main results of our pipeline systems in Table 5. For the purpose of comparison, we provided the NMT systems with either the manual transcripts (*Manual*) or the output of our ASR system (*Pipeline*), which is described in Section 3.1. As expected for error cascading in pipeline systems, BLEU scores drop substantially, by up to 36.4%, when translating noisy ASR outputs compared to translating the clean English transcript.

A paired permutation test shows that *Manual* outperforms *Pipeline* statistically significantly with p-value $< 1\%$. NMT systems trained with good manual transcripts might be intolerant to various ASR errors, and it is very likely they will propagate the errors during decoding.

Additionally, the final row in Table 5 we show the BLEU scores of the NMT system trained with additional WMT-Filtered data. There is a large improvement, for example from 23.96 to 28.07 on the *dev2010*. This confirms that adding more bitext helps.

While the corpus-level BLEU score of the pipeline is lower than the manual system, we did observe some interesting variances at the level of individual sentences: it is not the case that translations of ASR outputs are always worse than translations of manual, clean transcripts. Figure 1 compares the sentence-level BLEU scores in three different scatter plots. For each sentence in *tst2010*, we have the English transcript (EN-ref) and the resulting translation (DE-manual) by our NMT system; we also have the English ASR output (EN-ASR) and the resulting translation (DE-pipeline). Finally we have the correct German reference (DE-ref). We then computed three sentence-level BLEU scores (with add-one smoothing) as follows:

- Manual BLEU: BLEU of DE-manual vs. DE-ref
- Pipeline BLEU: BLEU of DE-pipeline vs. DE-ref
- ASR BLEU: BLEU of EN-ASR vs. EN-ref

Our goal is to compare ASR BLEU (which measures whether the English sentence was difficult to transcribe) with Manual/Pipeline BLEU. Our original hypothesis is that sentences with low ASR BLEU should result in a larger difference in Manual BLEU minus Pipeline BLEU.

Interestingly, as seen in Figure 1 (c), there are individual sentences where Manual BLEU is less than Pipeline BLEU. An example is shown in Table 6. The difference between the English reference and the ASR output is "where it gets" vs "what gets", which are arguably both correct. However, the NMT result is very different, one translating perfectly and the other not. It appears that since NMT output has high variance, i.e. it can output very different translations even when the inputs are semantically similar.

4. End-to-End System

In this section, we describe our end-to-end speech translation model and transfer learning from pre-trained ASR model.

4.1. Model for End-to-End Speech Translation

We used an attention-based encoder-decoder model for the end-to-end speech translation model. The architecture of the encoder is exactly the same as that in the ASR model in Section 3.1 (VGG-like CNN layers followed by stacked BiLSTM layers). The decoder includes two modification from the ASR decoder: (1) adopting input-feeding mechanism [18], and (2) adding scheduled sampling [16].

It is possible to integrate a language model during the decoding stage (i.e. *shallow fusion* [20]) and also training stage (i.e. *deep fusion* [21] and *cold fusion* [22]), but we did not use any language models for the speech translation task in this paper. We'll leave them to the future work.

4.2. Transfer learning from ASR

In our preliminary experiments, it took too much time to train end-to-end speech translation models from scratch, i.e. many

[4]Due to time and computational resources limitation, we only tried a subset of all the possible combinations.

English Reference	But here's where it gets interesting.	**NMT Result (*Manual*)**	Aber hier ist das, was interessant wird.
ASR Output	But here's what gets interesting.	**NMT Result (*Pipeline*)**	Aber hier wird es interessant.

Table 6: An example where Pipeline system outperforms the Manual system (100 sentBLEU vs. 6.5 sentBLEU). The German reference for the utterance is *Aber hier wird es interessant* .

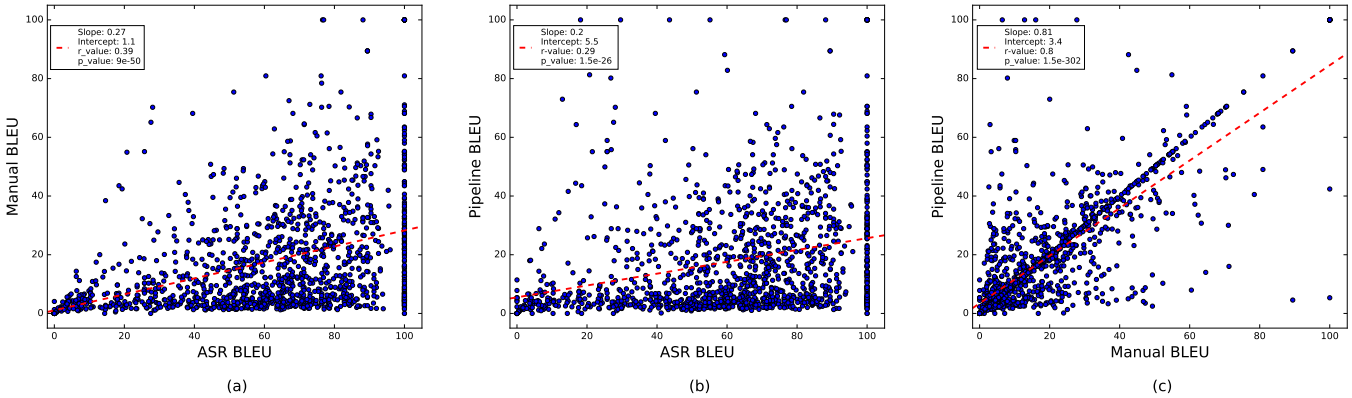

Figure 1: Sentence-level BLEU of *manual*, *pipeline* and *ASR* system on *tst2010*. A linear least-squares regression is calculated for each pair of systems.

epochs are required for convergence. Therefore, we explored methods to better initialize our end-to-end model.

Speech translation can be viewed as a combination of ASR and MT tasks, so we can treat the encoder and decoder networks as having roles in ASR and MT, respectively. Therefore, it is a natural choice to initialize the encoder with that of a pre-trained ASR model. Initialing the decoder with pre-trained MT model will be left to the future work.

Weiss et al. [23] shows improvements of BLEU scores by multi-task learning (MTL) with ASR task by sharing the encoder. Berard et al. [24] also shows both MTL and pre-training strategies lead to fast convergence and better results for end-to-end speech translation. Here we chose the pre-training strategy and transfer the weights of the encoder in the ASR model to the end-to-end model prior to training.

4.3. Experiments

We did the same data preprocessing as speech transcription in Section 3.2. We also built end-to-end speech translation models with the ESPnet toolkit with a pytorch backend. The differences of the architecture, optimization, and decoding from speech transcription models are as follows:

- We did not use the CTC framework due to its monotone assumption
- We used scheduled sampling with probability 0.2
- We ran for 30 epochs
- We did not perform beam search decoding (i.e. greedy decoding)
- We did not use any language models (due to time constraints)

We use the official scripts from the organizer and calculated case-sensitive BLEU scores with multi-bleu-detok.perl in the Moses toolkit after detokenization. We report BLEU scores in Table 7 for both pipeline systems and end-to-end speech translation models (E2E). There are several observations:

First, we can confirm that better ASR models led to better BLEU scores in the pipeline systems when comparing Table 3 and Table 7. The two ASR models with 5k units have the lowest WER scores, and the resulting two pipeline systems (b) and (d) also achieved the best BLEU scores. Second, it seems challenging to train an E2E speech translation model from scratch. Transfer learning with parameters from an existing ASR model gave consistent gains.

Finally, there are large differences between pipeline and end-to-end systems. For example, on *dev2010*, E2E trained from scratch achieved a BLEU of 4.44, E2E with transfer from ASR achieved a BLEU of 6.71, and the pipeline systems achieved BLEU in the range of 14. This may be due to data sparseness. Perhaps the explicit intermediate representation of transcripts in the language of the speech input is important for constraining the model complexity. Further, 10k wordpieces is a relatively large unit size for speech models and the data needs of an end-to-end model may be larger than that of a pipeline model.

We show some examples of the end-to-end speech translation model transferred from pre-trained ASR (system (f) in Table 7) in Table 8. Despite the low BLEU scores in general, the end-to-end model sometimes do generate reasonable sentences and correctly predicts keywords such as proper nouns and numbers. Our system was robust to misspelling because

we used 10k units for the vocabulary. The official development sets include many long sentences, and it appears that our E2E model may be doing relatively worse compared to Pipeline systems on long sentences.

5. Discussion

We described our pipeline and end-to-end speech translation systems for IWSLT 2018. For the official evaluations, we submitted the pipeline system (a) in Table 7 as a contrastive system and the E2E system (f) in Table 7 as the primary system; they were our best systems at the time of submission. Our main findings are that (1) pipeline systems can be very strong systems, and that (2) more work is needed to train end-to-end systems effectively, especially in small datasets.

For the official development sets, we had to use other tools to segment audio files before decoding and then match the number of references and hypotheses after decoding. We found that this affected WER and BLEU scores seriously due to misalignment. Therefore, the exact segmentation information for acoustic features is desired for the future evaluation in speech translation.

6. Acknowledgements

We would like to thank Tomoki Hayashi for his help on the preparation of the speech translation recip in ESPnet and Pamela Shapiro for providing the Moore-Lewis data selection toolkit.

7. References

[1] R. C. Moore and W. Lewis, "Intelligent selection of language model training data," in *Proceedings of the ACL 2010 Conference Short Papers*. Uppsala, Sweden: Association for Computational Linguistics, July 2010, pp. 220–224. [Online]. Available: http://www.aclweb.org/anthology/P10-2041

[2] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, *et al.*, "The kaldi speech recognition toolkit," in *Proc. of ASRU*, 2011.

[3] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proc. of ICML*, 2006, pp. 369–376.

[4] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *Proc. of NIPS*, 2015, pp. 577–585.

[5] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. of ICLR*, 2015.

[6] S. Kim, T. Hori, and S. Watanabe, "Joint CTC-attention based end-to-end speech recognition using multi-task learning," in *Preceedings of ICASSP*. IEEE, 2017, pp. 4835–4839.

[7] T. Hori, S. Watanabe, Y. Zhang, and W. Chan, "Advances in joint ctc-attention based end-to-end speech recognition with a deep cnn encoder and rnn-lm," *arXiv preprint arXiv:1706.02737*, 2017.

[8] S. Watanabe, T. Hori, S. Kim, J. R. Hershey, and T. Hayashi, "Hybrid ctc/attention architecture for end-to-end speech recognition," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 8, pp. 1240–1253, 2017.

[9] S. Meignier and T. Merlin, "Lium spkdiarization: an open source toolkit for diarization," in *CMU SPUD Workshop*, 2010.

[10] O. Bender, R. Zens, E. Matusov, and H. Ney, "Alignment templates: the rwth smt system." in *IWSLT*, 2004, pp. 79–84.

[11] S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, N. E. Y. Soplin, J. Heymann, M. Wiesner, N. Chen, *et al.*, "Espnet: End-to-end speech processing toolkit," *arXiv preprint arXiv:1804.00015*, 2018.

[12] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[13] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[14] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. of CVPR*, 2016, pp. 2818–2826.

[15] J. Chorowski and N. Jaitly, "Towards better decoding and language model integration in sequence to sequence models," in *Proc. of Interspeech*, 2017.

[16] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *Preceedings of NIPS*, 2015, pp. 1171–1179.

[17] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," *arXiv preprint arXiv:1508.07909*, 2015.

[18] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.

System	Configuration	dev2010	test2010	test2013	test2014	test2015
(a) Pipeline	ASR (10k unit, 3 layer); NMT (ST TED)	14.22	13.62	14.21	11.73	10.68
(b) Pipeline	ASR (5k unit, 3 layer); NMT (ST TED)	14.68	**14.70**	**15.08**	**12.23**	**11.59**
(c) Pipeline	ASR (10k unit, 5 layer); NMT (ST TED)	14.54	13.05	14.60	11.73	11.16
(d) Pipeline	ASR (5k unit, 5 layer); NMT (ST TED)	**14.87**	13.76	14.75	11.58	10.96
(e) E2E	train from scratch	4.44	4.10	3.57	3.52	2.42
(f) E2E	transfer learning from ASR parameters	6.71	6.21	6.01	5.08	4.51

Table 7: BLEU evaluated on the development sets using the official scripts provided by organizers. Note the results here are not comparable to Table 5 due to differences in the tokenization and evaluation scripts.

EN(Ref)	In the last five years we've added 70 million tons of CO2 every 24 hours – 25 million tons every day to the oceans.
DE (Ref)	In den letzten 5 Jahren haben wir 70 Millionen Tonnen an CO2 produziert alle 24 Stunden – 25 Millionen Tonnen jeden Tag in die Ozeane.
DE (Hyp)	In den letzten fnf Jahren haben wir die 70 Millionen Tonnen CO2 / h. Wir haben die Ostkste
EN(Ref)	But not just any mission, it's a mission that is perfectly matched with your current level in the game.
DE (Ref)	Aber nicht nur irgendeine Mission, sondern eine Mission, die perfekt zu Ihrem aktuellen Level im Spiel passt, richtig?
DE (Hyp)	Aber nicht nur die Mission, sondern nur eine Mission, die sich perfekt antreibt. Mit dem deren auf dem Spiel.

Table 8: Examples of the end-to-end speech translation model (system (f) in Table 7)

[19] F. Hieber, T. Domhan, M. Denkowski, D. Vilar, A. Sokolov, A. Clifton, and M. Post, "Sockeye: A toolkit for neural machine translation," *arXiv preprint arXiv:1712.05690*, 2017.

[20] A. Kannan, Y. Wu, P. Nguyen, T. N. Sainath, Z. Chen, and R. Prabhavalkar, "An analysis of incorporating an external language model into a sequence-to-sequence model," 2017.

[21] C. Gulcehre, O. Firat, K. Xu, K. Cho, L. Barrault, H.-C. Lin, F. Bougares, H. Schwenk, and Y. Bengio, "On using monolingual corpora in neural machine translation," *arXiv preprint arXiv:1503.03535*, 2015.

[22] A. Sriram, H. Jun, S. Satheesh, and A. Coates, "Cold fusion: Training seq2seq models together with language models," *arXiv preprint arXiv:1708.06426*, 2017.

[23] R. J. Weiss, J. Chorowski, N. Jaitly, Y. Wu, and Z. Chen, "Sequence-to-sequence models can directly translate foreign speech," *arXiv preprint arXiv:1703.08581*, 2017.

[24] A. Bérard, L. Besacier, A. C. Kocabiyikoglu, and O. Pietquin, "End-to-end automatic speech translation of audiobooks," *arXiv preprint arXiv:1802.04200*, 2018.