Bridging the Gap between Pre-Training and Fine-Tuning for End-to-End Speech Translation

Chengyi Wang,1 Yu Wu,2 Shujie Liu,2 Zhenglu Yang,1 Ming Zhou2
1Nankai University, Tianjin, China
2Microsoft Research Asia, Beijing, China
cywang@mail.nankai.edu.cn, Wu.Yu@microsoft.com,
shujliu@microsoft.com, yangzl@nankai.edu.cn, mingzhou@microsoft.com

Abstract
End-to-end speech translation, a hot topic in recent years, aims to translate a segment of audio into a specific language with an end-to-end model. Conventional approaches employ multi-task learning and pre-training methods for this task, but they suffer from the huge gap between pre-training and fine-tuning. To address these issues, we propose a Tandem Connectionist Encoding Network (TCEN) which bridges the gap by reusing all subnets in fine-tuning, keeping the roles of subnets consistent, and pre-training the attention module. Furthermore, we propose two simple but effective methods to guarantee the speech encoder outputs and the MT encoder inputs are consistent in terms of semantic representation and sequence length. Experimental results show that our model outperforms baselines 2.2 BLEU on a large benchmark dataset.

1 Introduction
Speech-to-Text translation (ST) is essential for a wide range of scenarios: for example in emergency calls, where agents have to respond emergent requests in a foreign language (Munro 2010); or in online courses, where audiences and speakers use different languages (Jan et al. 2018). To tackle this problem, existing approaches can be categorized into cascaded method (Ney 1999; Ma et al. 2019), where a machine translation (MT) model translates outputs of an automatic speech recognition (ASR) system into target language, and end-to-end method (Duong et al. 2016; Weiss et al. 2017), where a single model learns acoustic frames to target word sequence mappings in one step towards the final objective of interest. Although the cascaded model remains the dominant approach due to its better performance, the end-to-end method becomes more and more popular because it has lower latency by avoiding inferences with two models and rectifies the error propagation in theory.

Since it is hard to obtain a large-scale ST dataset, multi-task learning (Weiss et al. 2017; Berard et al. 2018) and pre-training techniques (Bansal et al. 2019) have been applied to end-to-end ST model to leverage large-scale datasets of ASR and MT. A common practice is to pre-train two encoder-decoder models for ASR and MT respectively, and then initialize the ST model with the encoder of the ASR model and the decoder of the MT model. Subsequently, the ST model is optimized with the multi-task learning by weighing the losses of ASR, MT, and ST. This approach, however, causes a huge gap between pre-training and fine-tuning, which are summarized into three folds:

• **Subnet Waste**: The ST system just reuses the ASR encoder and the MT decoder, while discards other pre-trained subnets, such as the MT encoder. Consequently, valuable semantic information captured by the MT encoder cannot be inherited by the final ST system.

• **Role Mismatch**: The speech encoder plays different roles in pre-training and fine-tuning. The encoder is a pure acoustic model in pre-training, while it has to extract semantic and linguistic features additionally in fine-tuning, which significantly increases the learning difficulty.

• **Non-pre-trained Attention Module**: Previous work (Berard et al. 2018) trains attention modules for ASR, MT and ST respectively, hence, the attention module of ST does not benefit from the pre-training.

To address these issues, we propose a Tandem Connectionist Encoding Network (TCEN), which is able to reuse all subnets in pre-training, keep the roles of subnets consistent, and pre-train the attention module. Concretely, the TCEN consists of three components, a speech encoder, a text encoder, and a target text decoder. Compared to prior works, the encoder of TCEN is a concatenation of an ASR encoder and an MT encoder, and our model does not have an ASR decoder, so the subnet waste issue is solved. Furthermore, the two encoders work at tandem, disentangling acoustic feature extraction and linguistic
Figure 1: An illustration of multi-task learning for speech translation. Networks inherited from pre-trained models are labeled by rectangles.

feature extraction, ensuring the role consistency between pre-training and fine-tuning. Moreover, we reuse the pre-trained MT attention module in ST, so we can leverage the alignment information learned in pre-training.

Since the text encoder consumes word embeddings of plausible texts in MT task but uses speech encoder outputs in ST task, another question is how one guarantees the speech encoder outputs are consistent with the word embeddings. We further modify our model to achieve semantic consistency and length consistency. Specifically, (1) the projection matrix at the CTC classification layer for ASR is shared with the word embedding matrix, ensuring that they are mapped to the same latent space, and (2) the length of the speech encoder output is proportional to the length of the input frame, so it is much longer than a natural sentence. To bridge the length gap, source sentences in MT are lengthened by adding word repetitions and blank tokens to mimic the CTC output sequences.

We conduct comprehensive experiments on the IWSLT18 speech translation benchmark \cite{jan2018real}, demonstrating the effectiveness of each component. Our model is significantly better than previous methods by 3.6 and 2.2 BLEU scores for the subword-level decoding and character-level decoding strategies, respectively.

Our contributions are three-folds: 1) we shed light on why previous ST models cannot sufficiently utilize the knowledge learned from the pre-training process; 2) we propose a new ST model, which alleviates shortcomings in existing methods; and 3) we empirically evaluate the proposed model on a large-scale public dataset.

2 Background

2.1 Problem Formulation

End-to-end speech translation aims to translate a piece of audio into a target-language translation in one step. The raw speech signals are usually converted to sequences of acoustic features, e.g. Mel filterbank features. Here, we define the speech feature sequence as $x = (x_1, \cdots, x_{T_x})$. The transcription and translation sequences are denoted as $y^s = (y^s_1, \cdots, y^s_{T_y})$, and $y^t = (y^t_1, \cdots, y^t_{T_y})$ respectively. Each symbol in $y^s$ or $y^t$ is an integer index of the symbol in a vocabulary $V_{src}$ or $V_{trg}$ respectively (e.g. $y^s_i = k, k \in [0, |V_{src}| - 1]$). In this work, we suppose that an ASR dataset, an MT dataset, and a ST dataset are available, denoted as $A = \{(x_t, y^s_t)\}_{t=0}^{L}$, $M = \{(y^t_j, y^t_{j+1})\}_{j=0}^{J}$ and $S = \{(x_t, y^t_t)\}_{t=0}^{L}$ respectively. Given a new piece of audio $x$, our goal is to learn an end to end model to generate a translation sentence $y^t$ without generating an intermediate result $y^s$.

2.2 Multi-Task Learning and Pre-training for ST

To leverage large scale ASR and MT data, multi-task learning and pre-training techniques are widely employed to improve the ST system. As shown in Figure[1] there are three popular multi-task strategies for ST, including 1) one-to-many setting, in which a speech encoder is shared between ASR and ST tasks; 2) many-to-one setting in which a decoder is shared between MT and ST tasks; and 3) many-to-many setting where both the encoder and decoder are shared.

A many-to-many multi-task model contains two encoders as well as two decoders. It can be jointly trained on ASR, MT, and ST tasks. As the attention module is task-specific, three attentions are defined.

Usually, the size of $A$ and $M$ is much larger than $S$. Therefore, the common training practice is to pre-train the model on ASR and MT tasks and then fine-tune it with a multi-task learning manner. However, as aforementioned, this method suffers from subnet waste, role mismatch and non-pre-trained attention issues, which severely limits the end-to-end ST performance.

3 Our method

In this section, we first introduce the architecture of TCEN, which consists of two encoders connected in tandem, and one decoder with an attention module. Then we give the pre-training and fine-tuning strategy for TCEN. Finally, we propose our solutions for semantic and length inconsistency problems, which are caused by multi-task learning.
3.1 TCEN Architecture

Figure 2 sketches the overall architecture of TCEN, including a speech encoder $\text{enc}_s$, a text encoder $\text{enc}_t$ and a decoder $\text{dec}$ with an attention module $\text{att}$. During training, the $\text{enc}_s$ acts like an acoustic model which reads the input $x$ to word or subword representations $h^s$, then $\text{enc}_s$ learns high-level linguistic knowledge into hidden representations $h^t$. Finally, the $\text{dec}$ defines a distribution probability over target words. The advantage of our architecture is that two encoders disentangle acoustic feature extraction and linguistic feature extraction, making sure that valuable knowledge learned from ASR and MT tasks can be effectively leveraged for ST training. Besides, every module in pre-training can be utilized in fine-tuning, alleviating the subnet waste problem.

Follow Inaguma et al. (2018), we use CNN-BiLSTM architecture to build our model. Specifically, the input features $x$ are organized as a sequence of feature vectors in length $T_x$. Then, $x$ is passed into a stack of two convolutional layers followed by max-pooling:

$$
\begin{align}
\tilde{v}^{(l)} &= \sigma(W^{(l)} \ast v^{(l-1)} + b^{(l)}) \\
v^{(l)} &= \max(v^{(l)}, \tilde{v}^{(l)})
\end{align}
$$

where $v^{(l-1)}$ is feature maps in last layer and $W^{(l)}$ is the filter. The max-pooling layers downsample the sequence in length by a total factor of four. The down-sampled feature sequence is further fed into a stack of five bidirectional $d$-dimensional LSTM layers:

$$
\begin{align}
\tilde{h}^{(l)} &= \text{LSTM}(h^{(l-1)}), \\
\tilde{h}^{(l)} &= \tilde{h}^{(l)}(h^{(l-1)}) \\
h^{(l)} &= [\tilde{h}^{(l)}; \tilde{h}^{(l)}]
\end{align}
$$

where $[;]$ denotes the vector concatenation. The final output representation from the speech encoder is denoted as $h^s = (h^s_1, \ldots, h^s_T)$, where $h^s_k \in \mathbb{R}^d$.

The text encoder $\text{enc}_t$ consists of two bidirectional LSTM layers. In ST task, $\text{enc}_t$ accepts speech encoder output $h^s$ as input. While in MT, $\text{enc}_t$ consumes the word embedding representation $e^w$ derived from $y^w$, where each element $e^w_k$ is computed by choosing the $y^w_k$-th vector from the source embedding matrix $W_{E^s}$. The goal of $\text{enc}_t$ is to extract high-level linguistic features like syntactic features or semantic features from lower level subword representations $h^s$ or $e^w$. Since $h^s$ and $e^w$ belong to different latent space and have different lengths, there remain semantic and length inconsistency problems. We will provide our solutions in Section 3.3. The output sequence of $\text{enc}_t$ is denoted as $h^t$.

The decoder is defined as two unidirectional LSTM layers with an additive attention $\text{att}$. It predicts target sequence $y^w$ by estimating conditional probability $P(y^w| x)$:

$$
\begin{align}
&c_k = \text{att}(z_{k-1}, h^t) \\
&z_k = \text{dec}(z_{k-1}, y^w_{k-1}, c_k)
&\text{softmax}(W \cdot z_k)
\end{align}
$$

Here, $z_k$ is the the hidden state of the decoder RNN at $k$ step and $c_k$ is a time-dependent context vector computed by the attention $\text{att}$.

3.2 Training Procedure

Following previous work, we split the training procedure to pre-training and fine-tuning stages. In pre-training stage, the speech encoder $\text{enc}_s$ is trained towards CTC objective using dataset $\mathcal{A}$, while the text encoder $\text{enc}_t$ and the decoder $\text{dec}$ are trained on MT dataset $\mathcal{M}$. In fine-tuning stage, we jointly train the model on ASR, MT, and ST tasks.

### Pre-training

To sufficiently utilize the large dataset $\mathcal{A}$ and $\mathcal{M}$, the model is pre-trained on CTC-based ASR task and MT task in the pre-training stage.

For ASR task, in order to get rid of the requirement for decoder and enable the $\text{enc}_s$ to generate subword representation, we leverage connectionist temporal classification (CTC) (Graves et al. 2006) loss to train the speech encoder.
Table 1: An example of the comparison between the golden transcript and the predicted CTC paths given the corresponding speech. ‘-’ denotes the blank token and the following number represents repeat times.

| Transcript $y^*$ | CTC path $\pi_1$ | CTC path $\pi_2$ |
|------------------|------------------|------------------|
| were not v @en @en @ge @ful all | -3(1) we -4(3) were -3(4) v @en @en @ge - @ful -8(3) at -3(3) all -10(10) | -9(9) we -3(4) were -4(3) v v @en @en @en @ge - @ful -7(3) at -3(3) all -10(10) |

Given an input $x$, $\text{enc}_e$ emits a sequence of hidden vectors $h^e$, then a softmax classification layer predicts a CTC path $\pi$, where $\pi_t \in V_{src} \cup \{\text{‘-’}\}$ is the observing label at particular RNN step $t$, and ‘-’ is the blank token representing no observed labels:

$$P(\pi | x) \approx \prod_{t=1}^{T} P(\pi_t | x) = \prod_{t=1}^{T} \text{softmax}(W_{ctc} \cdot h^e_t)$$

where $W_{ctc} \in \mathbb{R}^{d \times (|V_{src}| + 1)}$ is the weight matrix in the classification layer and $T$ is the total length of encoder RNN.

A legal CTC path $\pi$ is a variation of the source transcription $y^s$ by allowing occurrences of blank tokens and repetitions, as shown in Table[1]. For each transcription $y$, there exist many legal CTC paths in length $T$. The CTC objective trains the model to maximize the probability of observing the golden sequence $y^g$ which is calculated by summing the probabilities of all possible legal paths:

$$P(y | x) = \sum_{\pi \in \Phi_T(y)} P(\pi | x)$$

$$L_{CTC}(\theta) = - \sum_{(x, y^s) \in A} \log P(y^s | y^e; \theta_{\text{enc}_e}, \theta_{W_{ctc}})$$

where $\Phi_T(y)$ is the set of all legal CTC paths for sequence $y$ with length $T$. The loss can be easily computed using forward-backward algorithm. More details about CTC are provided in supplementary material.

For MT task, we use the cross-entropy loss as the training objective. During training, $y^s$ is converted to embedding vectors $e^s$ through embedding layer $W_{E^s}$, then $\text{enc}_e$ consumes $e^s$ and pass the output $h^e$ to decoder. The objective function is defined as:

$$L_{MT}(\theta) = - \sum_{(y^s, y^e) \in A} \log P(y^e | y^s; \theta_{\text{enc}_e}, \theta_{\text{dec}}, \theta_{W_{E^s}})$$

Fine-tune In fine-tune stage, we jointly update the model on ASR, MT, and ST tasks. The training for ASR and MT follows the same process as it was in pre-training stage.

For ST task, the $\text{enc}_e$ reads the input $x$ and generates $h^e$, then $\text{enc}_c$ learns high-level linguistic knowledge into $h^c$. Finally, the $\text{dec}$ predicts the target sentence. The ST loss function is defined as:

$$L_{ST}(\theta) = - \sum_{(x, y^s) \in S} \log P(y^e | x; \theta_{\text{enc}_e}, \theta_{\text{enc}_c}, \theta_{\text{dec}})$$

Following the update strategy proposed by Luong et al. (2016), we allocate a different training ratio $\alpha_i$ for each task. When switching between tasks, we select randomly a new task $i$ with probability $\frac{\alpha_i}{\sum_i \alpha_i}$.
to noisy sentences in CTC path format, and replace standard MT with denoising MT for multi-tasking.

Specifically, we first train a CTC ASR model based on dataset \( A = \{(x_i, y_i^a)\}_{i=0}^l \), and generate a CTC-path \( \pi \), for each audio \( x_i \), by greedy decoding. Then we define an operation \( S(\cdot) \), which converts a CTC path \( \pi \) to a sequence of the unique tokens \( u \) and a sequence of repetition times for each token \( l \), denoted as \( S(\pi) = (u, l) \). Notably, the operation is reversible, meaning that \( S^{-1}(u, l) = \pi \). We use the example \( \pi \) in Table 1 and show the corresponding \( u \) and \( l \) in Table 2.

| CTC path \( \pi \) | we were not v en ge ful at all |
|---------------------|-----------------|
| \( u \) | - - - - not - - - - |
| \( l \) | 11 2 3 1 3 4 1 2 1 1 8 2 3 1 10 |

Table 2: The CTC path \( \pi \) and corresponding unique tokens \( u \) and repetition times \( l \), where \( S(\pi) = (u, l) \).

Then we build a dataset \( \mathcal{P} = \{(y_i^e, u_i, l_i)\}_{i=0}^l \) by decoding all the audio pieces in \( A \) and transform the resulting path by the operation \( S(\cdot) \). After that, we train a seq2seq model, as shown in Figure 3 which takes \( y_i^e \) as input and decodes \( u_i \) and \( l_i \) as outputs. With the seq2seq model, a noisy MT dataset \( \mathcal{M}' = \{(\pi_i, y_i^e)\}_{i=0}^l \) is obtained by converting every source sentence \( y_i^e \in \mathcal{M} \) to \( \pi_i \), where \( \pi_i = S^{-1}(u_i, l_i) \). We did not use the standard seq2seq model which takes \( y_i^e \) as input and generates \( \pi \) directly, since there are too many blank tokens '-' in \( \pi \) and the model tends to generate a long sequence with only blank tokens. During MT training, we randomly sample text pairs from \( \mathcal{M}' \) and \( \mathcal{M} \) according to a hyper-parameter \( k \). After tuning on the validation set, about 30% pairs are sampled from \( \mathcal{M}' \). In this way, the \( enc_t \) is more robust toward the longer inputs given by the \( enc_e \).

4 Experiments

We conduct experiments on the IWSLT18 speech translation task (Fan et al. 2018). Since IWSLT participants use different data pre-processing methods, we reproduce several competitive baselines based on the ESPnet\(^2\) (Watanabe et al. 2018) for a fair comparison.

4.1 Dataset

Speech translation data: The organizer provides a speech translation corpus extracting from the TED talk (ST-TED), which consists of raw English wave files, English transcripts, and aligned German translations. The corpus contains 272 hours of English speech with 171k segments. We split 2k segments from the corpus as dev set and tst2010, tst2013, tst2014, tst2015 are used as test sets.

Speech recognition data: Aside from ST-TED, TED-LIUM2 corpus (Rousseau, Delégile, and Esteve 2014) is provided as speech recognition data, which contains 207 hours of English speech and 93k transcript sentences.

Text translation data: We use transcription and translation pairs in the ST-TED corpus and WIT\(^3\) as in-domain MT data, which contains 130k and 200k sentence pairs respectively. WMT2018 is used as out-of-domain training data which consists of 41M sentence pairs.

Data preprocessing: For speech data, the utterances are segmented into multiple frames with a 25 ms window size and a 10 ms step size. Then we extract 80-channel log-Mel filter bank and 3-dimensional pitch features using Kaldi (Povey et al. 2011), resulting in 83-dimensional input features. We normalize them by the mean and the standard deviation on the whole training set. The utterances with more than 3000 frames are discarded. The transcripts in ST-TED are in true-case with punctuation while in TED-LIUM2, transcripts are in lower-case and unpunctuated. Thus, we lower case all the sentences and remove the punctuation to keep consistent. To increase the amount of training data, we perform speed perturbation on the raw signals with speed factors 0.9 and 1.1.

For the text translation data, sentences longer than 80 words or shorter than 10 words are removed. Besides, we discard pairs whose length ratio between source and target sentence is smaller than 0.5 or larger than 2.0. Word tokenization is performed using the Moses scripts\(^4\) and both English and German words are in lower-case.

We use two different sets of vocabulary for our experiments. For the subword experiments, both English and German vocabularies are generated using sentencepiece\(^5\) (Kudo 2018) with a fixed size of 5k tokens. Inaguma et al. (2018) show that increasing the vocabulary size is not helpful for ST task. For the character experiments, both English and German sentences are represented in the character level.

For evaluation, we segment each audio with the LIUM SpkDiarization tool (Meignier and Merlin 2010) and then perform MWER segmentation with RWTH toolkit (Bender et al. 2004). We use lowercase BLEU as evaluation metric.

4.2 Baseline Models and Implementation

We compare our method with following baselines.

Vanilla ST baseline: The vanilla ST (Inaguma et al. 2018) has only a speech encoder and a decoder. It is trained from scratch on the ST-TED corpus.

Pre-training baselines: We conduct three pre-training baseline experiments: 1) encoder pre-training, in which the ST encoder is initialized from an ASR model; 2) decoder pre-training, in which the ST decoder is initialized from an MT model; and 3) encoder-decoder pre-training, where both the encoder and decoder are pre-trained. The ASR model has the same architecture with vanilla ST model, trained on the mixture of ST-TED and TED-LIUM2 corpus. The MT

---

\(^2\)https://github.com/espnet/espnet
\(^3\)https://wit3.fbk.eu/mnt.php?release=2017-01-trnted
\(^4\)https://github.com/moses-smt/mosesdecoder/blob/master/scripts/tokenizer/tokenizer.perl
\(^5\)https://github.com/google/sentencepiece
model has a text encoder and decoder with the same architecture of which in TCEN. It is first trained on WMT data (out-of-domain) and then fine-tuned on in-domain data.

**Multi-task baselines**: We also conduct three multi-task baseline experiments including one-to-many setting, many-to-one setting, and many-to-many setting. In the first two settings, we train the model with $\alpha_{st} = 0.75$ while $\alpha_{asr} = 0.25$ or $\alpha_{mt} = 0.25$. For many-to-many setting, we use $\alpha_{st} = 0.6$, $\alpha_{asr} = 0.2$ and $\alpha_{mt} = 0.2$. For MT task, we use only in-domain data.

**Many-to-many+pre-training**: We train a many-to-many multi-task model where the encoders and decoders are derived from pre-trained ASR and MT models.

**Triangle+pre-train**: Anastasopoulos and Chiang (2018) proposed a triangle multi-task strategy for speech translation. Their model solves the subnet waste issue by concatenating an ST decoder to an ASR encoder-decoder model. Notably, their ST decoder can consume representations from the speech encoder as well as the ASR decoder. For a fair comparison, the speech encoder and the ASR decoder are initialized from the pre-trained ASR model. The Triangle model is fine-tuned under their multi-task manner.

All our baselines as well as TCEN are implemented based on ESPNet (Watanabe et al., 2018), the RNN size is set as $d = 1024$ for all models. We use a dropout of 0.3 for embeddings and encoders, and train using Adadelta with initial learning rate of 1.0 for a maximum of 10 epochs.

For training of TCEN, we set $\alpha_{asr} = 0.2$ and $\alpha_{mt} = 0.8$ in the pre-training stage, since the MT dataset is much larger than ASR dataset. For fine-tune, we use $\alpha_{st} = 0.6$, $\alpha_{asr} = 0.2$ and $\alpha_{mt} = 0.2$, same as the ‘many-to-many’ baseline.

For testing, we select the model with the best accuracy on speech translation task on dev set. At inference time, we use a beam size of 10, and the beam scores include length normalization with a weight of 0.2.

### 4.3 Experimental Results

Table 3 shows the results on four test sets as well as the average performance. Our method significantly outperforms the strong ‘many-to-many+pretrain’ baseline by 3.6 and 2.2 BLEU scores respectively, indicating the proposed method is very effective that substantially improves the translation quality. Besides, both pre-training and multi-task learning can improve translation quality, and the pre-training settings (2nd-4th rows) are more effective compared to multi-task settings (5th-8th rows). We observe a performance degradation in the ‘triangle+pretrain’ baseline. Compared to our method, where the decoder receives higher-level syntactic and semantic linguistic knowledge extracted from text encoder, their ASR decoder can only provide lower word-level linguistic information. Besides, since their model lacks text encoder and the architecture of ST decoder is different from MT decoder, their model cannot utilize the large-scale MT data in all the training stages.

Interestingly, we find that the char-level models outperform the subword-level models in all settings, especially in vanilla baseline. A similar phenomenon is observed by Bérand et al. (2018). A possible explanation is that learning the alignments between speech frames and subword units in another language is notoriously difficult. Our method can bring more gains in the subword setting since our model is good at learning the text-to-text alignment and the subword-level alignment is more helpful to the translation quality.

### 4.4 Discussion

**Ablation Study** To better understand the contribution of each component, we perform an ablation study on subword-level experiments. The results are shown in Table 4. In ‘-MT noise’ setting, we do not add noise to source sentences for MT. In ‘-weight sharing’ setting, we use different parameters in CTC classification layer and source embedding layer. These two experiments prove that both weight sharing and using noisy MT input benefit to the final translation quality. Performance degrades more in ‘-weight shar-

| System          | tst2010 | tst2013 | tst2014 | tst2015 | Average |
|-----------------|---------|---------|---------|---------|---------|
| TCEN            | 15.49   | 15.50   | 13.21   | 13.02   | 14.31   |
| -MT noise       | 15.01   | 14.95   | 13.34   | 12.80   |         |
| -weight sharing | 14.02   | 12.25   | 11.66   |         |         |
| -pretrain       | 8.98    | 8.42    | 7.94    | 8.08    |         |

Table 4: Ablation study for subword-level experiments.
|                | tst2010 | tst2013 | tst2014 | tst2015 |
|----------------|---------|---------|---------|---------|
| cascaded       | 13.38   | 15.84   | 12.94   | 13.79   |
| cascaded+re-seg| 17.12   | 17.77   | 14.94   | 15.01   |
| our model      | 17.61   | 17.67   | 15.73   | 14.94   |

Table 5: BLEU comparison of cascaded results and our best end-to-end results. re-seg denotes the ASR outputs are re-segmented before fed into the MT model.

Learning Curve  It is interesting to investigate why our method is superior to baselines. We find that TCEN achieves a higher final result owing to a better start-point in fine-tuning. Figure 4 provides learning curves of subword accuracy on validation set. The x-axis denotes the fine-tuning training steps. The vanilla model starts at a low accuracy, because its networks are not pre-trained on the ASR and MT data. The trends of our model and ‘many-to-many+pretrain’ are similar, but our model outperforms it about five points in the whole fine-tuning process. It indicates that the gain comes from bridging the gap between pre-training and fine-tuning rather than a better fine-tuning process.

Compared with a Cascaded System  Table 3 compares our model with end-to-end baselines. Here, we compare our model with cascaded systems. We build a cascaded system by combining the ASR model and MT model used in pre-training baseline. Word error rate (WER) of the ASR system and BLEU score of the MT system are reported in the supplementary material. In addition to a simple combination of the ASR and MT systems, we also re-segment the ASR outputs before feeding to the MT system, denoted as cascaded+re-seg. Specifically, we train a seq2seq model (Bahdanau, Cho, and Bengio 2015) on the MT dataset, where the source side is a no punctuation sentence and the target side is a natural sentence. After that, we use the seq2seq model to add sentence boundaries and punctuation on ASR outputs. Experimental results are shown in Table 5. Our end-to-end model outperforms the simple cascaded model over 2 BLEU scores, and it achieves a comparable performance with the cascaded model combining with a sentence re-segment model.

5 Related Work

Early works conduct speech translation in a pipeline manner (Ney 1999; Matusov, Kanthak, and Ney 2005), where the ASR output lattices are fed into an MT system to generate target sentences. HMM (Juang and Rabine 1991), DenseNet (Huang et al. 2017), TDNN (Peddinti, Povey, and Khudanpur 2015) are commonly used ASR systems, while RNN with attention (Bahdanau, Cho, and Bengio 2015) and Transformer (Vaswani et al. 2017) are top choices for MT. To enhance the robustness of the NMT model towards ASR errors, Tsvetkov, Metze, and Dyer (2014) and Chen et al. (2017) propose to simulate the noise in training and inference.

To avoid error propagation and high latency issues, recent works propose translating the acoustic speech into text in target language without yielding the source transcription (Duong et al. 2016). Since ST data is scarce, pre-training (Bansal et al. 2019), multi-task learning (Duong et al. 2016; Bérard et al. 2018), curriculum learning (Kano, Sakti, and Nakamura 2018), attention-passing (Sperber et al. 2019), and knowledge distillation (Liu et al. 2019; Jia et al. 2019) strategies have been explored to utilize ASR data and MT data. Specifically, Weiss et al. (2017) show improvements of performance by training the ST model jointly with the ASR and the MT model. Bérard et al. (2018) observe faster convergence and better results due to pre-training and multi-task learning on a larger dataset. Bansal et al. (2019) show that pre-training a speech encoder on one language can improve ST quality on a different source language. All of them follow the traditional multi-task training strategies. Kano, Sakti, and Nakamura (2018) propose to use curriculum learning to improve ST performance on syntactically distant language pairs. To effectively leverage transcriptions in ST data, Anastasopoulos and Chiang (2018) augment the multi-task model where the target decoder receives information from the source decoder and they show improvements on low-resource speech translation. Their model just consumes ASR and ST data, in contrast, our work sufficiently utilizes the large-scale MT data to capture the rich semantic knowledge. Jia et al. (2019b) use pre-trained MT and text-to-speech (TTS) synthesis models to convert weakly supervised data into ST pairs and demonstrate that an end-to-end MT model can be trained using only synthesised data.

6 Conclusion

This paper has investigated the end-to-end method for ST. It has discussed why there is a huge gap between pre-training and fine-tuning in previous methods. To alleviate these issues, we have proposed a method, which is capable of reusing every sub-net and keeping the role of sub-net consistent between pre-training and fine-tuning. Empirical studies have demonstrated that our model significantly outperforms baselines.
Chen, Z. 2017. Sequence-to-sequence models can directly translate foreign speech. In Interspeech 2017, 2625–2629.