**Abstract**

How to learn a better speech representation for end-to-end speech-to-text translation (ST) with limited labeled data? Existing techniques often attempt to transfer powerful machine translation (MT) capabilities to ST, but neglect the representation discrepancy across modalities. In this paper, we propose the Speech-TExt Manifold Mixup (STEMM) method to calibrate such discrepancy. Specifically, we mix up the representation sequences of different modalities, and take both unimodal speech sequences and multimodal mixed sequences as input to the translation model in parallel, and regularize their output predictions with a self-learning framework. Experiments on MuST-C speech translation benchmark and further analysis show that our method effectively alleviates the cross-modal representation discrepancy, and achieves significant improvements over a strong baseline on eight translation directions.

**1 Introduction**

Speech-to-text translation (ST) aims at translating acoustic speech signals into text in a foreign language, which has wide applications including voice assistants, translation for multinational video conferences, and so on. Traditional ST methods usually combine automatic speech recognition (ASR) and machine translation (MT) in a cascaded manner (Sperber et al., 2017; Cheng et al., 2018; Sperber et al., 2019; Dong et al., 2019b; Zhang et al., 2019a; Lam et al., 2021b), which might suffer from error propagation and high latency. To break this bottleneck, end-to-end ST systems attracted much attention recently (Wang et al., 2020b,c; Dong et al., 2021a,b; Han et al., 2021; Inaguma et al., 2021a; Tang et al., 2021a), which learn a unified model to generate translations from speech directly. Some recent work has shown great potential for end-to-end speech translation, even surpassing traditional cascaded systems (Ye et al., 2021; Xu et al., 2021).

As a cross-modal task, a major challenge in training an end-to-end ST model is the representation discrepancy across modalities, which means there is a modality gap between speech representations and text embeddings, as shown in the left sub-figure of Figure 1. Existing approaches often adopt a sophisticated MT model to help the training of ST, with some techniques like pretraining (Wang et al., 2020c; Ye et al., 2021; Xu et al., 2021), multi-task learning (Ye et al., 2021; Han et al., 2021; Tang et al., 2021a) and knowledge distillation (Liu et al., 2019; Gaido et al., 2020; Inaguma et al., 2021b; Tang et al., 2021a). Although these methods have achieved impressive improvements in ST task, these methods are not necessarily the best way to leverage the MT knowledge. Considering that during training, the input of the translation module only include speech sequences or text sequences, the lack of multimodal contexts makes it difficult for the ST model to learn from the MT model. Inspired by recent studies on some cross-
lingual (Lample and Conneau, 2019; Liu et al., 2020a; Lin et al., 2020) and cross-modal (Li et al., 2021b; Zhou et al., 2020; Dong et al., 2019a) tasks, we suggest that building a shared semantic space between speech and text, as illustrated in the right sub-figure of Figure 1, has the potential to benefit the most from the MT model.

In this paper, we propose the Speech-TEXT Manifold Mixup (STEMM) method to bridge the modality gap between text and speech. In order to calibrate the cross-modal representation discrepancy, we mix up the speech and text representation as the input and keep the target sequence unchanged. Specifically, STEMM is a self-learning framework, which takes both the speech representation and the mixed representation as parallel inputs to the translation model, and regularizes their output predictions. Experimental results show that our method achieves promising performance on the benchmark dataset MuST-C (Di Gangi et al., 2019a), and even outperforms a strong cascaded baseline. Furthermore, we found that our STEMM could effectively alleviate the cross-modal representation discrepancy, and project two modalities into a shared space.

2 Method

In this section, we will begin with the basic problem formulation (Section 2.1) and introduce the model architecture (Section 2.2). Then, we introduce our proposed Speech-TEXT Manifold Mixup (STEMM) in Section 2.3. Finally, we introduce our proposed self-learning framework with STEMM in Section 2.4 and present two mixup ratio strategies in Section 2.5. Figure 2 illustrates the overview of our proposed method.

2.1 Problem Formulation

The speech translation corpus usually contains \textit{speech-transcription-translation} triples, which can be denoted as \( D = \{(s, x, y)\} \). Here \( s \) is the sequence of audio wave, \( x \) is the transcription in the source language, and \( y \) is the translation in the target language. End-to-end speech translation aims to generate translation \( y \) directly from the audio wave \( s \), without generating intermediate transcription \( x \).

2.2 Model Architecture

Inspired by recent works (Dong et al., 2021b; Xu et al., 2021) in end-to-end speech translation, we decompose the ST model into three modules: \textit{acoustic encoder}, \textit{translation encoder}, and \textit{translation decoder}. The \textit{acoustic encoder} first encodes the original audio wave into hidden states, fed into the \textit{translation encoder} to learn further semantic information. Finally, the \textit{translation decoder} generates the translation based on the output of the \textit{translation encoder}.

\textbf{Acoustic Encoder} As recent works (Ye et al., 2021; Han et al., 2021) show that Wav2vec2.0 (Baevski et al., 2020) can improve the performance of speech translation, we first use a pretrained Wav2vec2.0 to extract speech representations \( c \) from the audio wave \( s \). We add two additional convolutional layers to further shrink the length of speech representations by a factor of 4, denoted as \( a = \text{CNN}(c) \).

\textbf{Translation Encoder} Our \textit{translation encoder} is composed of \( N_c \) transformer (Vaswani et al., 2017) encoder layers, which includes a self-attention layer, a feed-forward layer, normalization layers,
and residual connections. For MT task, the input of the translation encoder is the embedding of transcription $e = \text{Emb}(x)$. For ST task, it is the output sequence of the acoustic encoder $a$. The input can also be the multimodal mixed sequence with our proposed STEMM (see details in Section 2.3). Generally, for the input sequence $\chi$, we obtain the contextual representations $h(\chi)$ after $N_e$ transformer (Vaswani et al., 2017) layers, which are fed into the translation decoder for predicting the translation.

**Translation Decoder** Our translation decoder is composed of $N_t$ transformer decoder layers, which contain an additional cross-attention layer compared with transformer encoder layers. For the input sequence $\chi$, the cross entropy loss is defined as:

$$L_{CE}(\chi, y) = - \sum_{i=1}^{|y|} \log p(y_i | y_{<i}, h(\chi)).$$  \tag{1}$$

**Pretrain-finetune** We follow the pretrain-finetune paradigm to train our model. First, we pretrain the translation encoder and translation decoder with parallel transcription-translation pairs, derived from both the speech translation corpus and the external MT dataset. Also, the acoustic encoder is pretrained on large amounts of unlabeled audio data in a self-supervised manner. We combine those pretrained modules and finetune the whole model for ST.

### 2.3 Speech-Text Manifold Mixup (STEMM)

As we mentioned in Section 1, to alleviate the representation discrepancy due to the lack of multimodal contexts, we present the Speech-Text Manifold Mixup (STEMM) method to mix up the sequence of speech representations and word embeddings. We first introduce STEMM in this section and later show how to use it to help the training of ST.

Note the sequence of sub-word embeddings as $e = [e_1, e_2, ..., e_{|e|}]$ and the sequence of speech representations as $a = [a_1, a_2, ..., a_{|a|}]$, where the sequence lengths usually follow $|a| \geq |e|$. We first perform a word-level forced alignment between speech and text transcriptions to determine when particular words appear in the speech segment. Formally, the aligner recognizes a sequence of word units $w = [w_1, w_2, ..., w_T]$, and for each word $w_i$, it returns the start position $l_i$ and end position $r_i$ in the sequence of speech representation $a$. Meanwhile, we denote the corresponding sub-word span for word $w_i$ as $[x_{m_i} : x_{n_i}]$, with its embeddings matrix $[e_{m_i} : e_{n_i}]$, where $m_i$ and $n_i$ are the start position and end position in the sequence of sub-words. To mix up both sequences, for each word unit $w_i$, we choose either the segment of speech representations $[a_{l_i} : a_{r_i}]$ or sub-word embeddings $[e_{m_i} : e_{n_i}]$ with a certain probability $p^*$, referred to mixup ratio in this paper.

$$m_i = \begin{cases} [a_{l_i} : a_{r_i}] & p \leq p^* \\ [e_{m_i} : e_{n_i}] & p > p^* \end{cases}$$  \tag{2}$$

where $p$ is sampled from the uniform distribution $\mathcal{U}(0, 1)$.

Finally, we concatenate all $m_i$ together and obtain the mixup sequence:

$$m = \text{Concat}(m_1, m_2, ..., m_T).$$  \tag{3}$$

Note that in terms of the mixup representation sequence length, we have $|e| \leq |m| \leq |a|$. Considering the positions of tokens have changed after mixup, we add positional encodings to the token embeddings. We further perform layer normalization to normalize the embeddings:

$$\text{Mixup}((s, x), p^*) = \text{LayerNorm}(m + \text{Pos}(m)),$$  \tag{4}$$

where $\text{Pos}(\cdot)$ is the sinusoid positional embedding (Vaswani et al., 2017). Mixup $((s, x), p^*)$ indicates the mixup sequence of speech $s$ and text $x$ with probability $p^*$, which is fed into the translation encoder for predicting the translation.

### 2.4 Self-learning with STEMM

With the help of our proposed STEMM, we are now able to access multimodal mixed sequences, in addition to the unimodal speech sequences. We integrate them into a self-learning framework. Specifically, we input both unimodal speech sequences and multimodal mixed sequences into the translation module (translation encoder and translation decoder). In this way, translation of unimodal speech sequences focuses on the ST task itself, while the translation of multimodal mixed sequences is devoted to capture the connections between representations in different modalities. Besides, we try to regularize above two output predictions by minimizing the Jensen-Shannon Divergence (JSD) between two output distributions,
which is
\[
\mathcal{L}_{\text{ISD}}(s, x, y, p^*) = \sum_{i=1}^{|y|} \text{JSD}\{p_\theta(y_i | y_{<i}, h(s)) || p_\theta(y_i | y_{<i}, h(\text{Mixup}((s, x), p^*)))\},
\]
where \(h(\cdot)\) is the contextual representation outputted by the translation encoder. \(p_\theta(y_i | y_{<i}, h(s))\)
\[
= \lambda p_\theta(y_i | y_{<i}, h(\text{Mixup}((s, x), p^*)))
\]
where \(\lambda\) is the coefficient weight to control \(\mathcal{L}_{\text{ISD}}\).

### 2.5 Mixup Ratio Strategy

When using our proposed STEMM, an important question is the strategy to determine the mixup ratio \(p^*\). Here we try two strategies: static mixup ratio and uncertainty-aware mixup ratio.

#### Static Mixup Ratio

We use the same mixup ratio \(p^*\) for all instances throughout the whole training process. We will show how we determined this important hyper-parameter in Section 4.3.

#### Uncertainty-aware Mixup Ratio

With this strategy, we determine the mixup ratio for each instance according to the prediction uncertainty of the ST task, defined as the average entropy of predicted distributions of all target tokens:
\[
u = \frac{1}{|y|} \sum_{i=1}^{|y|} \text{Entropy}(p_\theta(y_i | y_{<i}, h(s))),
\]
and then we set the mixup ratio \(p^*\) as follows:
\[
p^* = \sigma\left(\frac{u}{U} - \frac{1}{2}\right),
\]
where \(U\) is a normalization factor which rescales \(u\) to \([0, 1]\), \(\sigma(\cdot)\) is a sigmoid function to prevent \(p^*\) from dropping too quickly.

### 3 Experiments

#### 3.1 Datasets

**MuST-C** We conduct experiments on MuST-C (Di Gangi et al., 2019a) dataset. MuST-C is a multilingual speech translation dataset, which contains translations from English (En) to 8 languages: German (De), French (Fr), Russian (Ru), Spanish (Es), Italian (It), Romanian (Ro), Portuguese (Pt), and Dutch (Nl). It is one of the largest speech translation datasets currently, which contains at least 385 hours of audio recordings from TED Talks, with their manual transcriptions and translations at the sentence level. We use dev set for validation and tst-COMMON set for test.

**MT Datasets** Our model architecture allows us to utilize external parallel sentence pairs in large-scale machine translation datasets. Therefore, we incorporate data from WMT for En-De, En-Fr, En-Ru, En-Es, En-Ro, and OPUS100\(^1\) for En-Pt, En-It, En-Nl, as pretraining corpora. The detailed statistics of all datasets included are shown in Table 1.

<table>
<thead>
<tr>
<th>MT</th>
<th>ST (MuST-C)</th>
<th>hours</th>
<th>#sents</th>
</tr>
</thead>
<tbody>
<tr>
<td>De</td>
<td>408</td>
<td>234K</td>
<td></td>
</tr>
<tr>
<td>Fr</td>
<td>492</td>
<td>280K</td>
<td></td>
</tr>
<tr>
<td>Ru</td>
<td>489</td>
<td>270K</td>
<td></td>
</tr>
<tr>
<td>Es</td>
<td>504</td>
<td>240K</td>
<td></td>
</tr>
<tr>
<td>Ro</td>
<td>432</td>
<td>240K</td>
<td></td>
</tr>
<tr>
<td>It</td>
<td>465</td>
<td>258K</td>
<td></td>
</tr>
<tr>
<td>Pt</td>
<td>385</td>
<td>211K</td>
<td></td>
</tr>
<tr>
<td>Nl</td>
<td>442</td>
<td>253K</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Statistics of all datasets

\(^1\)http://opus.nlpl.eu/opus-100.php
\(^2\)https://github.com/MontrealCorpusTools/
\(^3\)https://github.com/google/sentencepiece
Wav2vec2.0 (Baevski et al., 2020) following the without finetuning

Training and Inference
We train our model in a

Translation decoder
We train the MT model i.e.,

Translation encoder
pretrain-finetune manner. During pretraining, we
former layers comprises 512 hidden units, 8 atten-
transformer decoder layers. Each of these trans-
the audio, with kernel size 5, stride size 2, padding
dimensional convolutional layers to further shrink
data from LibriSpeech (Panayotov et al., 2015)
base configuration, which is pretrained on audio
three modules. For the
Our model consists of
Model Configuration
Our model consists of three modules. For the acoustic encoder, we use Wav2vec2.0 (Baevski et al., 2020) following the base configuration, which is pretrained on audio data from LibriSpeech (Panayotov et al., 2015) without finetuning. We add two additional 1-dimensional convolutional layers to further shrink the audio, with kernel size 5, stride size 2, padding 2, and hidden dimension 1024. For the translation encoder, we use \( N_e = 6 \) transformer encoder layers. For the translation decoder, we use \( N_d = 6 \) transformer decoder layers. Each of these transformer layers comprises 512 hidden units, 8 attention heads, and 2048 feed-forward hidden units.

Training and Inference
We train our model in a pretrain-finetune manner. During pretraining, we train the MT model i.e., translation encoder and translation decoder, with transcription-translation pairs. The learning rate is 7e-4. We train the model with at most 33k input tokens per batch. During finetuning, the learning rate is set to 1e-4. We finetune the whole model up to 25 epochs to avoid overfitting, with at most 16M source audio frames per batch. The training will early-stop if the loss on dev set did not decrease for ten epochs. During both pretraining and finetuning, we use an Adam optimizer (Kingma and Ba, 2015) with \( \beta_1 = 0.9, \beta_2 = 0.98 \), and 4k warm-up updates. The learning rate will decrease proportionally to the inverse square root of the step number after warm-up. The dropout is set to 0.1, and the value of label smoothing is set to 0.1. We use the uncertainty-aware mixup ratio strategy by default, and the mixup ratio \( p^* \) is set to 0.4 when using static strategy. The weight \( \lambda \) of JSD loss is set to 1.0.

During inference, We average the checkpoints of the last 10 epochs for evaluation. We use beam search with a beam size of 5. We use sacreBLEU (Post, 2018) to compute case-sensitive detokenized BLEU (Papineni et al., 2002) scores and the statistical significance of translation results with paired bootstrap resampling (Koehn, 2004) for a fair comparison. All models are trained on 8 Nvidia Tesla-V100 GPUs. We implement our models based on fairseq (Ott et al., 2019).

Baseline Systems
We compare our method with several strong end-to-end ST systems including:

---

4Model can be downloaded at https://dl.fbaipublicfiles.com/fairseq/wav2vec/wav2vec_small.pt

5https://github.com/mjpost/sacrebleu

6sacreBLEU signature: nrefs:1 l bs:1000 l seed:12345 l case:mixed l eff:no l tok:13a l smooth:exp l version:2.0.0

7https://github.com/pytorch/fairseq

Table 2: BLEU scores on MuST-C tst-COMMON set. “Speech” denotes unlabeled audio data. † use OpenSubtitles (Lison and Tiedemann, 2016) as external MT data. * and ** mean the improvements over W2V2-Transformer baseline is statistically significant (\( p < 0.05 \) and \( p < 0.01 \), respectively).

<table>
<thead>
<tr>
<th>Models</th>
<th>WER↓</th>
<th>MT BLEU↑</th>
<th>ST BLEU↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cascaded</td>
<td>9.9</td>
<td>31.7</td>
<td>27.5</td>
</tr>
<tr>
<td>W2V2-Transformer</td>
<td>-</td>
<td>31.7</td>
<td>26.9</td>
</tr>
<tr>
<td>STEMM</td>
<td>-</td>
<td>31.7</td>
<td>28.7**</td>
</tr>
</tbody>
</table>

Table 3: Comparison with cascaded baseline on MuST-C En-De tst-COMMON set. ** mean the improvements over cascaded baseline is statistically significant (\( p < 0.01 \)).

<table>
<thead>
<tr>
<th>Models</th>
<th>Pretrain w/ external MT data</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>En-De</td>
</tr>
<tr>
<td>Pretrain w/o external MT data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Faireseq ST (Wang et al., 2020a)</td>
<td>× × ×</td>
<td>22.7</td>
</tr>
<tr>
<td>AFS (Zhang et al., 2020)</td>
<td>× × ×</td>
<td>22.2</td>
</tr>
<tr>
<td>DTT (Le et al., 2020)</td>
<td>× × ×</td>
<td>23.6</td>
</tr>
<tr>
<td>Self-training (Pino et al., 2020)</td>
<td>✓ ✓ ✓</td>
<td>25.2</td>
</tr>
<tr>
<td>BfKD (Inaguma et al., 2021a)</td>
<td>× × ×</td>
<td>25.3</td>
</tr>
<tr>
<td>SATE (Xu et al., 2021)</td>
<td>× × ×</td>
<td>25.2</td>
</tr>
<tr>
<td>XSTNet (Ye et al., 2021)</td>
<td>✓ ✓ ✓</td>
<td>25.5</td>
</tr>
<tr>
<td>W2V2-Transformer</td>
<td>✓ ✓ ✓</td>
<td>24.1</td>
</tr>
<tr>
<td>STEMM</td>
<td>✓ ✓ ✓</td>
<td>25.6**</td>
</tr>
</tbody>
</table>
Fairseq ST (Wang et al., 2020a), AFS (Zhang et al., 2020), DDT (Le et al., 2020), MTL (Tang et al., 2021b), Self-training (Pino et al., 2020), BiKD (Inaguma et al., 2021a), FAT-ST (Zheng et al., 2021a), JT-S-MT (Tang et al., 2021a), SATE (Xu et al., 2021b), Chimera (Han et al., 2021) and XSTNet (Ye et al., 2021). Besides, we implement a strong baseline W2V2-Transformer based on Wav2vec2.0. It has the same model architecture as our proposed STEMM and is pretrained in the same way. The only difference is that it is only finetuned on the ST task, while we adopt a self-learning framework during finetuning.

4 Results and Analysis

4.1 Results on MuST-C Dataset

Comparison with End-to-end Baselines As shown in Table 2, our implemented W2V2-Transformer is a relatively strong baseline, which proves the effectiveness of Wav2vec2.0 module and MT pretraining. Without external MT data, our method achieves an improvement of 1.0 BLEU (average over 8 directions) over the strong baseline, which proves our proposed self-learning framework could effectively improve the performance of the ST task. It even outperforms baselines with external MT data on En-Es, En-It, En-Ro, En-Pt, and En-Nl. When we introduce additional MT data, our method also yields a 0.8 BLEU improvement compared with baseline. Note that our performance is slightly worse than XSTNet (Ye et al., 2021). However, our method is orthogonal with theirs, which focuses on the training procedure of end-to-end ST model. We will investigate how to combine them together in the future.

Comparison with Cascaded Baseline We also implement a strong cascaded system, whose ASR part is composed of a pretrained Wav2vec2.0 module and 6 transformer decoder layers, and the MT part is the same as our pretrained MT module. Both cascaded systems and end-to-end models are trained with the same data ($D$ and $D_{MT}$). As shown in Table 3, the end-to-end baseline W2V2-Transformer is inferior to the cascaded system, but our method significantly outperforms it, which shows the potential of our STEMM method.

4.2 Ablation Studies

Is Each Learning Objective Effective? As shown in Equation 6, our training objective contains three terms. Besides the cross-entropy objective $L_{CE}(s, y)$ for speech translation, we investigate the effects of the other two auxiliary training objectives. As shown in Table 4, when we input the additional multimodal mixed sequence into the model and optimize the cross-entropy loss (Line 3), it can already outperform the baseline (Line 4) significantly. When we regularize two output predictions with JSD loss (Line 2), the performance can be further boosted. The uncertainty-aware strategy reduces the cost for searching mixup ratio and has better performance. We present two different mixup ratio strategies in Section 2.5. To evaluate their impacts, we conduct another ablation study on MuST-C En-De. We observe that the BLEU scores on tst-COMMON set with different auxiliary training objectives. STEMM Trans. indicates the criterion entropy loss of translation of multimodal mixed sequence $L_{CE}(Mixup((s, x), p^*), y)$. ** mean the improvements over W2V2-Transformer baseline (last row in the table) is statistically significant ($p < 0.01$).

Mixup Ratio | STEMM Trans. | JSD | BLEU
--- | --- | --- | ---
uncertainty-aware | ✓ | ✓ | 28.7**
static | ✓ | x | 28.5**
static | ✓ | x | 27.9**
static | x | x | 26.9

Table 4: BLEU scores on MuST-C En-De tst-COMMON set with different auxiliary training objectives. STEMM Trans. indicates the criterion entropy loss of translation of multimodal mixed sequence $L_{CE}(Mixup((s, x), p^*), y)$. ** mean the improvements over W2V2-Transformer baseline (last row in the table) is statistically significant ($p < 0.01$).

4.3 What is the Optimal Mixup Ratio?

When using static mixup ratio strategy, it is important to choose the mixup ratio $p^*$. We constrain $p^*$ in $[0.0, 0.2, 0.4, 0.6, 0.8]$ for experiments on MuST-C En-De tst-COMMON set, as shown in Figure 3. When $p^* = 0.0$, the translation task with the mixed sequence as input degrades to the MT task. We interestingly find that self-learning with MT tasks performed the worst (i.e. lowest BLEU) than self-learning with STEMM at other mixup ratios. This confirms what we mentioned in Section 1, that the representation discrepancy between speech and text makes the MT task an inferior boost to ST.

Our method achieves the best performance at $p^* = 0.4$. To find a reasonable explanation, we do a more in-depth study of the representation of the speech, text, and their mixup sequence (STEMM).
In Figure 4, we take out the sequential representation of the speech (output of acoustic encoder), text sequences (output of embedding layer), and the STEMM sequences, average them over the sequence dimension, and apply the T-SNE dimensionality reduction algorithm to reduce the 512 dimensions to two dimensions. We plot the bivariate kernel density estimation based on the reduced 2-dim representation. We find that when $p^* = 0.4$, the mixup representation just lies between the representation of speech and text sequences. That is why it calibrates the cross-modal representation discrepancy more easily and gets the best ST performance.

### 4.4 Can Our Model Alleviate Cross-modal Representation Discrepancy?

To examine whether our method alleviates the cross-modal representation discrepancy, we conduct some analysis of cross-modal word representations. As described in Section 2.3, for each word unit $w_i$, we identify the corresponding segment of speech representation $[a_{i_1}: a_{i_n}]$ and text embedding $[e_{m_1}: e_{m_n}]$. We define the word representation in each modality as follows:

$$
\alpha_i = \text{AvgPool}([a_{i_1}: a_{i_n}]),
$$
$$
\varepsilon_i = \text{AvgPool}([e_{m_1}: e_{m_n}]),
$$

where $\text{AvgPool}()$ denotes average-pooling operation across the sequence dimension, $\alpha_i$ and $\varepsilon_i$ denote the representation of word unit $w_i$ in speech and text modalities, respectively.

We calculate the average cosine similarity between $\alpha_i$ and $\varepsilon_i$ over all word units $w_i$ in MuST-C En-De tst-COMMON set. As shown in Table 5, our method could significantly improve the similarity of word representations across modalities over baseline. We believe it is because when training with our proposed STEMM, the speech segment and text segment of a word will appear in a similar multimodal context, which leads to similar representations. We also show the visualization of an example in Figure 5, we can observe that our method brings word representations within different modalities closer compared with baseline.

<table>
<thead>
<tr>
<th>Models</th>
<th>Similarity (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>W2V2-Transformer</td>
<td>32.31</td>
</tr>
<tr>
<td>STEMM</td>
<td>51.89</td>
</tr>
</tbody>
</table>

Table 5: Comparison of word-level representation similarity across modalities.
We like to play with stuff. W2V2-Transformer

Figure 5: Visualization of word representations in speech and text modalities. We visualize the representations by reducing the dimension with Principal Component Analysis (PCA). Our method brings word representation within different modalities closer.

Figure 6: Curve of BLEU scores on MuST-C En-De tst-COMMON against the size of external MT data used during pretraining.

4.5 How the Size of MT Data Influences Performance?

One important contributor to our excellent performance is the usage of external MT data. Therefore, how the amount of MT data affects the final performance is an important question. We vary the amount of available external MT data during pretraining on En-De direction. As shown in Figure 6, we observe a continuous improvement of BLEU scores with the increase of MT data, which shows that external MT data is helpful to improve ST.

4.6 Can the Final Model still Perform MT Task?

Our model is first pretrained on the MT task and then finetune for ST. An important question is whether there is a catastrophic forgetting problem during finetuning. We evaluate the model on the MT task and show the result in Table 6. We observe that when we only finetune the model on the ST task (W2V2-Transformer), the ability of text translation will be forgotten a lot. In contrast, when we use our self-learning framework during finetuning, even though there is no MT task, the MT capability can still be preserved.

5 Related Works

End-to-end ST To overcome the error propagation and high latency in the cascaded ST systems, Bérard et al. (2016); Duong et al. (2016) proved the potential of end-to-end ST without intermediate transcription, which has attracted much attention in recent years (Vila et al., 2018; Salesky et al., 2018, 2019; Di Gangi et al., 2019b,c; Bahr et al., 2019a; Inaguma et al., 2020). Since it is difficult to train an end-to-end ST model directly, some training techniques like pretraining (Weiss et al., 2017; Bérand et al., 2018; Bansal et al., 2019; Stoian et al., 2020; Wang et al., 2020b; Pino et al., 2020; Dong et al., 2021a; Alinejad and Sarkar, 2020; Zheng et al., 2021b; Xu et al., 2021), multi-task learning (Le et al., 2020; Vydana et al., 2021; Tang et al., 2021b; Ye et al., 2021; Tang et al., 2021a), curriculum learning (Kano et al., 2017; Wang et al., 2020c), and meta-learning (Indurthi et al., 2020) have been applied. To overcome the scarcity of ST data, Jia et al. (2019); Pino et al. (2019); Bahar et al. (2019b) proposed to generate synthesized data based on ASR and MT corpora. To overcome the modality gap, Han et al. (2021); Huang et al. (2021); Xu et al. (2021) further encode acoustic states which are more adaptive to the decoder. Previous works have mentioned that the modality gap between speech and text is one of the obstacles in the speech translation task, and to overcome such gap, one branch of the works (Liu et al., 2020b; Dong et al., 2021b; Xu et al., 2021) introduced a second encoder based on the conventional encoder-decoder model, to extract semantic information of speech and text. Recently, Han et al. (2021) built a shared semantic projection module that simulates the human brain, while in this work, we

<table>
<thead>
<tr>
<th>Models</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pretrained MT</td>
<td>31.7</td>
</tr>
<tr>
<td>W2V2-Transformer</td>
<td>19.5</td>
</tr>
<tr>
<td>STEMM</td>
<td>31.5</td>
</tr>
</tbody>
</table>

Table 6: BLEU scores of MT task on MuST-C En-De tst-COMMON set. Our proposed method almost preserves the text translation capability of pretrained MT model.
explored how to construct an intermediate state of the two modalities via the recent mixup method (i.e. Speech-TEXT Manifold Mixup) to narrow such gap. Note that our work is orthogonal with Ye et al. (2021)'s study in training procedure of end-to-end ST model.

Mixup Our work is inspired by the mixup strategy. Zhang et al. (2018) first proposed mixup as a data augmentation method to improve the robustness and the generalization of the model, where additional data are constructed as the linear interpolation of two random examples and their labels at the surface level. Verma et al. (2019) extended the surface-level mixup to the hidden representation by constructing manifold mixup interpolations. Recent work has introduced mixup on machine translation (Zhang et al., 2019b; Li et al., 2021a; Guo et al., 2022; Fang and Feng, 2022), sentence classification (Chen et al., 2020; Jindal et al., 2020; Sun et al., 2020), multilingual understanding (Yang et al., 2022), and speech recognition (Medennikov et al., 2018; Sun et al., 2021; Lam et al., 2021a; Meng et al., 2021), and obtained enhancements. Our approach is the first to introduce the idea of manifold mixup to the speech translation task with two modalities, speech, and text.

6 Conclusion

In this paper, we propose a Speech-TEXT Manifold Mixup (STEMM) method to mix up the speech representation sequences and word embedding sequences. Based on STEMM, we adopt a self-learning framework, which learns the translation of unimodal speech sequences and multimodal mixed sequences in parallel, and regularizes their output predictions. Experiments and analysis demonstrate the effectiveness of our proposed method, which can alleviate the cross-modal representation discrepancy to some extent and improve the performance of ST. In the future, we will explore how to further eliminate this discrepancy and fill the cross-modal transfer gap for ST.

Acknowledgements

We thank all the anonymous reviewers for their insightful and valuable comments. This work was supported by National Key R&D Program of China (NO. 2017YFE0192900).

References


Qianqian Dong, Mingxuan Wang, Hao Zhou, Shuang Xu, Bo Xu, and Lei Li. 2021a. Consecutive decoding for speech-to-text translation. In The Thirty-fifth AAAI Conference on Artificial Intelligence, AAAI.


Huiyun Yang, Huadong Chen, Hao Zhou, and Lei Li. 2022. Enhancing cross-lingual transfer by manifold mixup. In ICLR.

Rong Ye, Mingxuan Wang, and Lei Li. 2021. End-to-end speech translation via cross-modal progressive training. In Proc. of INTERSPEECH.


