NeurST: Neural Speech Translation Toolkit

Chengqi Zhao  Mingxuan Wang  Qianqian Dong  Rong Ye  Lei Li
ByteDance AI Lab, Shanghai, China
{zhaochengqi,d,wangmingxuan.89,dongqianqian,yerong,lileilab}@bytedance.com

Abstract

NeurST is an open-source toolkit for neural speech translation. The toolkit mainly focuses on end-to-end speech translation, which is easy to use, modify, and extend to advanced speech translation research and products. NeurST aims at facilitating the speech translation research for NLP researchers and building reliable benchmarks for this field. It provides step-by-step recipes for feature extraction, data preprocessing, distributed training, and evaluation. In this paper, we will introduce the framework design of NeurST and show experimental results for different benchmark datasets, which can be regarded as reliable baselines for future research. The toolkit is publicly available at https://github.com/bytedance/neurst and we will continuously update the performance of NeurST with other counterparts and studies at https://st-benchmark.github.io/.

1 Introduction

Speech translation (ST), which translates audio signals of speech in one language into text in a foreign language, is a hot research subject nowadays and has widespread applications, like cross-language videoconferencing or customer support chats.

Traditionally, researchers build a speech translation system via a cascading manner, including an automatic speech recognition (ASR) and a machine translation (MT) subsystem (Ney, 1999; Casacuberta et al., 2008; Kumar et al., 2014). Cascade systems, however, suffer from error propagation problems, where an inaccurate ASR output would theoretically cause translation errors. Owing to recent progress of sequence-to-sequence modeling for both neural machine translation (NMT) (Bahdanau et al., 2015; Luong et al., 2015; Vaswani et al., 2017) and end-to-end speech recognition (Chan et al., 2016; Chiu et al., 2018; Dong et al., 2018), it becomes feasible and efficient to train an end-to-end direct ST model (Berard et al., 2016; Duong et al., 2016; Weiss et al., 2017). This end-to-end fashion attracts much attention due to its appealing properties: a) modeling without intermediate ASR transcriptions obviously alleviates the propagation of errors; b) a single and unified ST model is beneficial to deployment with lower latency in contrast to cascade systems.

Recent studies show that end-to-end ST models achieve promising performance and are comparable with cascaded models (Ansari et al., 2020). The end-to-end solution has great potential to be the dominant technology for speech translation, however challenges remain. The first is about benchmarks. Many ST studies conduct experiments on different datasets. Liu et al. (2019) evaluate the method on TED English-Chinese; and Dong et al. (2021) use libri-trans English-French and IWSLT2018 English-German dataset; and Wu et al. (2020) show the results on CoVoST dataset and the FR/RO portions of MuST-C dataset. Different datasets make it difficult to compare the performance of their approaches. Further, even for the same dataset, the baseline results are not necessarily kept consistent. Take the libri-trans English-French dataset as an example. Dong et al. (2021) report the pre-trained baseline as 15.3 and the result of Liu et al. (2019) is 14.3 in terms of tokenized BLEU, while Inaguma et al. (2020) report 15.5 (detokenized BLEU). The mismatching baseline results in an unfair comparison on the improvements of their approaches. We think one of the primary reasons is that the preprocessing of audio data is complex, and the ST model training involves many tricks, such as pre-training and data augmentation.

Therefore a reproducible and reliable benchmark is required. In this work, we present NeurST, a toolkit for easily building and training end-to-end ST models, as well as end-to-end ASR and
NMT for cascade systems. We implement state-of-the-art Transformer-based models (Vaswani et al., 2017; Karita et al., 2019) and provide step-by-step recipes for feature extraction, data preprocessing, model training, and inference for researchers to reproduce the benchmarks. Though there exist several counterparts, such as Lingvo (Shen et al., 2019), fairseq-ST (Wang et al., 2020a) and Kaldi style ESPnet-ST (Inaguma et al., 2020), NeurST is specially designed for speech translation tasks, which encapsulates the details of speech processing and frees the developers from data engineering. It is easy to use and extend. The contributions of this work are as follows:

- NeurST is designed specifically for end-to-end ST, with clean and simple code. It is lightweight and independent of Kaldi, which simplifies installation and usage, and is more compatible for NLP researchers.
- We report strong benchmarks with well-designed hyper-parameters and show best practice on several ST corpora. We provide a series of recipes to reproduce them, which serves as reliable baselines for the speech translation field.

2 Design and Features

NeurST is implemented with both TensorFlow2 and PyTorch backends. In this section, we will introduce the design components and features of this toolkit.

2.1 Design

NeurST divides one running job into four components: Dataset, Model, Task and Executor.

Dataset NeurST abstracts out a common interface Dataset for data input. For example, we can train a speech translation model from either a raw dataset tarball or pre-extracted record files. The Dataset iterates on the data files and standardizes the read records, e.g., ST tasks only accept key-value pairs storing audio signals/features and translations. One can implement their logic to accept the data of various modalities.

Model NeurST provides an optimal implementation of Transformer and its adaptation to speech-to-text tasks, which achieve state-of-the-art performance on standard benchmarks. Moreover, one can customize various models using TensorFlow2/PyTorch APIs or combine the encoders, decoders, and layers inside the NeurST.

Task NeurST abstracts out Task interface to bridge Dataset and Model. In detail, Task defines data pipelines to match the data samples from Dataset to the input formats of Model. For examples, ST task does tokenization on the text translations and transforms each token to index. In this way, user-defined Dataset and Model can be efficiently integrated into NeurST, as long as they share the same Task.

Executor NeurST provides the execution logic for handling basic workflows of training, validation, and inference. Researchers can either define their specific process of training and evaluation, or pay less attention to API details in Executor but reuse them by simply customizing Dataset, Model and Task.

2.2 Features

Computation NeurST has high computation efficiency and it can be further optimized by enabling mixed-precision (Micikevicius et al., 2018) and XLA (Accelerated Linear Algebra). Furthermore, NeurST supports fast distributed training using Horovod (Sergeev and Balso, 2018) and Byteps (Peng et al., 2019; Jiang et al., 2020) on large-scale scenarios.

Data Preprocessing NeurST supports on-the-fly data preprocessing via a number of lightweight python packages, like python_speech_features2 for extracting audio features (e.g. mel-frequency cepstral coefficients and log-mel filterbank coefficients). And for text processing, NeurST integrates some effective tokenizers, including moses tokenizer3, byte pair encoding (BPE) (Sennrich et al., 2016b) and SentencePiece4. Alternatively, the training data can be preprocessed and stored in binary files (e.g., TFRecord) beforehand, which is guaranteed to improve the I/O performance during training. Moreover, to simplify such operations, NeurST provides the command-line tool to create such record files, which automatically iterates on various data formats defined by Dataset, preprocesses data samples according to Task and writes to the disk.

2https://github.com/jameslyons/python_speech_features
3The python version: https://github.com/alvations/sacremoses
4https://github.com/google/sentencepiece
**Transfer Learning** NeurST supports initializing the model variables from well-trained models as long as they have the same variable names. As for ST, we can initialize the ST encoder with a well-trained ASR encoder and initialize the ST decoder with a well-trained MT decoder, which facilitates to achieve promising improvements. Besides, NeurST also provides scripts for converting released models from other repositories, like wav2vec2.0 (Baevski et al., 2020) and BERT (Devlin et al., 2019). Researchers can conveniently integrate these pre-trained components to the customized models.

**Simultaneous Translation** NeurST keeps up with the recent progress of simultaneous translation. The models are extended to train with streaming audio or text input.

**Validation while Training** NeurST supports customizing validation process during training. By default, NeurST offers evaluation on development data during training and keeps track of the checkpoints with the best evaluation results.

**Monitoring** NeurST supports TensorBoard for monitoring metrics during training, such as training loss, training speed, and evaluation results.

**Model Serving** There is no gap between the research models and production models under NeurST, while they can be easily served with TensorFlow Serving. Moreover, for higher performance serving of standard transformer models, NeurST is able to integrate with other optimized inference libraries, like lightseq (Wang et al., 2021).

### 3 Speech Translation Benchmarks

We conducted experiments on several benchmark speech translation corpora using NeurST and compared the performance with other open-source codebases and studies. Though that would be an unfair comparison due to the different model structures and hyperparameters, the goal of NeurST is to provide strong and reproducible benchmarks for future research.

#### 3.1 Datasets

We choose the following publicly available speech translation corpora that include speech in a source language aligned to text in a target language:

<table>
<thead>
<tr>
<th>task</th>
<th>init scale</th>
<th>end scale</th>
<th>decay at</th>
<th>decay steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>MT</td>
<td>1.0</td>
<td>1.0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ASR</td>
<td>3.5</td>
<td>2.0</td>
<td>50k</td>
<td>50k</td>
</tr>
<tr>
<td>ST</td>
<td>3.5</td>
<td>1.5</td>
<td>50k</td>
<td>50k</td>
</tr>
</tbody>
</table>

Table 1: Hyperparameters of the learning rate schedule. Take the case of ST, the learning rate is scaled up by 3.5x for the first 50k steps. Then, we linearly decrease the scaling factor to 1.5 for 50k steps.

*libri-trans* (Kocabiyikoglu et al., 2018) is a small EN→FR dataset which was originally started from the *LibriSpeech* corpus, the audiobook recordings for ASR (Panayotov et al., 2015). The English utterances were automatically aligned to the e-books in French, and 236 hours of English speech aligned to French translations at utterance level were finally extracted. It has been widely used in previous studies. As such, we use the clean 100-hour portion plus the augmented machine translation from Google Translate as the training data and follow its split of dev and test data.

*MuST-C* (Di Gangi et al., 2019) is a multilingual speech translation corpus from English to 8 languages: Dutch (NL), French (FR), German (DE), Italian (IT), Portuguese (PT), Romanian (RO), Russian (RU) and Spanish (ES). MuST-C comprises at least 385 hours of audio recordings from English TED talks with their manual transcriptions and translations at sentence level for training, and we use the *dev* and *tst-COMMON* as our development and test data, respectively. To the best of our knowledge, MuST-C is currently the largest speech translation corpus available for each language pair.

#### 3.2 Data Preprocessing

Beyond the officially released version, we performed no other audio to text alignment and data cleaning on *libri-trans* and MuST-C datasets.

For speech features, we extracted 80-channel log-mel filterbank coefficients with windows of 25ms and steps of 10ms, resulting in 80-dimensional features per frame. The audio features of each sample were then normalized by the mean and the standard deviation. All texts were segmented into subword level by first applying Moses tokenizer and then BPE. In detail, we removed all punctuations and lowercased the sentences in the source side while the cases and punctuations of target sentences were

---

5 https://github.com/alicank/Translation-Augmented-LibriSpeech-Corpus

6 https://ict.fbk.eu/must-c/
Table 2: Case-insensitive BLEU scores on libri-trans test set under constrained setting (without additional ASR and MT data).

<table>
<thead>
<tr>
<th>Model</th>
<th>tok</th>
<th>detok</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cascade</td>
<td>ESPnet-ST ASR transf-s + CTC → MT (Inaguma et al., 2020)†</td>
<td>-</td>
</tr>
<tr>
<td>NeurST</td>
<td>ASR transf-s → MT</td>
<td>18.2</td>
</tr>
<tr>
<td>End-to-End</td>
<td>ST BiLSTM (Bahar et al., 2019)</td>
<td>17.0</td>
</tr>
<tr>
<td></td>
<td>ST transf-s (Liu et al., 2019)</td>
<td>14.3</td>
</tr>
<tr>
<td></td>
<td>ST transf-s + KD (Liu et al., 2019)</td>
<td>17.0</td>
</tr>
<tr>
<td></td>
<td>ESPnet-ST ST transf-s (Inaguma et al., 2020)†</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>TCEN-LSTM (Wang et al., 2020b)♭</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>ST transf-s (Wang et al., 2020c)</td>
<td>16.0</td>
</tr>
<tr>
<td></td>
<td>ST transf-s + curriculum pre-training (Wang et al., 2020c)</td>
<td>17.7</td>
</tr>
<tr>
<td></td>
<td>LUT (Dong et al., 2021)</td>
<td>17.8</td>
</tr>
<tr>
<td></td>
<td>NeurST ST transf-s</td>
<td>18.7</td>
</tr>
</tbody>
</table>

Table 3: Case-sensitive detokenized BLEU scores on MuST-C tst-COMMON.

<table>
<thead>
<tr>
<th>Model</th>
<th>DE</th>
<th>ES</th>
<th>FR</th>
<th>IT</th>
<th>NL</th>
<th>PT</th>
<th>RO</th>
<th>RU</th>
<th>avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cascade</td>
<td>ESPnet-ST ASR transf-s + CTC → MT (Inaguma et al., 2020)</td>
<td>23.7</td>
<td>28.7</td>
<td>33.8</td>
<td>24.0</td>
<td>27.9</td>
<td>29.0</td>
<td>22.7</td>
<td>16.4</td>
</tr>
<tr>
<td>NeurST</td>
<td>ASR transf-s → MT</td>
<td>23.4</td>
<td>28.0</td>
<td>33.9</td>
<td>23.8</td>
<td>27.1</td>
<td>28.3</td>
<td>22.2</td>
<td>16.0</td>
</tr>
<tr>
<td>End-to-End</td>
<td>ESPnet-ST ST transf-s (Inaguma et al., 2020)</td>
<td>22.9</td>
<td>28.0</td>
<td>32.8</td>
<td>23.8</td>
<td>27.4</td>
<td>28.0</td>
<td>21.9</td>
<td>15.8</td>
</tr>
<tr>
<td></td>
<td>fairseq-ST ST transf-s (Wang et al., 2020a)</td>
<td>22.7</td>
<td>27.2</td>
<td>32.9</td>
<td>22.7</td>
<td>27.3</td>
<td>28.1</td>
<td>21.9</td>
<td>15.3</td>
</tr>
<tr>
<td></td>
<td>ST transf-base + APS♭†</td>
<td>22.4</td>
<td>26.9</td>
<td>31.6</td>
<td>23.0</td>
<td>24.9</td>
<td>26.3</td>
<td>21.0</td>
<td>14.7</td>
</tr>
<tr>
<td></td>
<td>NeurST ST transf-s</td>
<td>22.8</td>
<td>27.4</td>
<td>33.3</td>
<td>22.9</td>
<td>27.2</td>
<td>28.7</td>
<td>22.2</td>
<td>15.1</td>
</tr>
</tbody>
</table>

respected. The BPE rules were jointly learned with 8,000 merge operations and shared across ASR, MT, and ST tasks.

3.3 Benchmark Models

We implemented Transformer (Vaswani et al., 2017), the state-of-the-art sequence-to-sequence model, for all our tasks.

In detail, for MT in cascade systems, the model included 6 layers for both encoder and decoders. The embedding dimension was 256, and the size of hidden units in feedforward layer was 2,048. The attention head for self-attention and cross-attention was set to 4. We used Adam optimizer (Kingma and Ba, 2015) with $\beta_1 = 0.9, \beta_2 = 0.98$ and applied the same schedule algorithm as Vaswani et al. (2017) for learning rate. We trained the MT models with a global batch size of 25,000 tokens.

As for ASR/ST, we referred to the recent progress of Transformer-based end-to-end ASR models (Dong et al., 2018; Karita et al., 2019) and extended the basic transformer model to be compatible with audio inputs. The audio frames were first compressed by two-layer CNN with 256 channels, $3 \times 3$ kernel and stride size 2, each of which was followed by a layer normalization. Then, we performed a linear transformation on the compressed audio representations to match the width of the transformer model. We used the same model structure as MT, except that we enlarged the number of encoder layers to 12 to obtain better performance. This configuration is labeled as transf-s (transformer small). For training, we used the same Adam optimizer as MT but set the warmup steps to 25,000, and we empirically scaled up the learning rate to accelerate the convergence. The hyperparameters of the learning rate schedule are listed in Table 1. Moreover, for GPU memory efficiency, we truncated the audio frames to 3,000 and removed training samples whose transcription length exceeded 120 and 150 for ASR and ST, respectively. The ASR models were trained with 120,000 frames per batch, while the batch size for ST was 80,000 frames. To further improve the performance of ST, we applied SpecAugment technique (Park et al., 2019) with frequency masking ($mF = 2, F = 27$).
Model tok detok
Cascade NeurST ASR transf-s → MT 17.4 16.0
End-to-End NeurST ST transf-s 17.8 16.3
ST transf-base + AFS t,f ♦ 18.6 17.2

Table 4: Case-sensitive BLEU scores on libri-trans test set under constrained setting. ♦ is from Zhang et al. (2020) with the proposed adaptive feature selection method, which uses the transformer base setting (embedding size=512).

and time masking \((mT = 2, T = 70, p = 0.2)\).

Additionally, we applied label smoothing of value 0.1 for training all three tasks. The encoder of the ST model is initialized by the ASR encoder by default unless noted.

3.4 Evaluation

For evaluation, we averaged the latest 10 checkpoints and used a beam width of 4 with no length penalty for all the above tasks.

We use word error rate (WER) to evaluate ASR models and report case-sensitive detokenized BLEU\(^8\) for MT and ST models. In order to compare with existing works, we also report case-insensitive tokenized BLEU using multi-bleu.perl in Moses for libri-trans dataset.

3.5 Main Results

The overall results and comparisons with other studies are illustrated in Table 2 and 3. It is worth noting that all results are from single models rather than ensemble models.

To make a fair comparison on libri-trans corpus, we list both tokenized and detokenized BLEU scores in Table 2 and strive to distinguish the metric of existing literature. Our transformer-based ST model, which only applies ASR pre-training and SpecAugment, achieves superior results versus recent works about knowledge distillation (Liu et al., 2019), curriculum pre-training (Wang et al., 2020c), and LUT (Dong et al., 2021). Compared with the counterpart ESPnet-ST, we also outperform by 0.5 BLEU, even though Inaguma et al. (2020) apply additional techniques like speed perturbation, pre-trained MT decoder, and CTC loss for ASR pre-training. The cascade baseline is slightly worse than that of ESPnet-ST (-0.2 BLEU) because the ASR+CTC can achieve lower WER (6.4)\(^9\) while our pure end-to-end ASR obtains 8.8. We surprisingly find that the end-to-end ST model exceeds the cascade system by 0.4~0.5 BLEU. We will discuss this in detail in section 3.7. And as a supplementary benchmark, we present case-sensitive BLEU scores in Table 4.

Table 3 illustrates the results on MuST-C tst-COMMON. The results of our end-to-end ST model are competitive with both fairseq-ST and ESPnet-ST.

3.6 Ablation Study

Training a direct ST model is more complicated than training an ASR or MT model. Our preliminary experiment based on a pure end-to-end ST model fails to converge on libri-trans corpus, which can be the result of the data scarcity. To alleviate this problem, pre-training some parts of the neural network is the most effective way and has been validated in all existing end-to-end ST studies. We show our results in Table 5 and 6 as a reference for future works. It turns out that we can obtain a reasonable or even better BLEU score by simply initializing the ST encoder with a pre-trained ASR encoder. The improvement by MT decoder initialization is relatively marginal in our setup. Furthermore, the SpecAugment technique can consistently boost ST models.

\(^8\)https://github.com/mjpost/sacrebleu

\(^9\)from https://github.com/espnet/espnet/blob/master/egs/libri_trans/asr1/RESULTS.md

<table>
<thead>
<tr>
<th>Model</th>
<th>NeurST</th>
<th>ESPnet-ST</th>
</tr>
</thead>
<tbody>
<tr>
<td>ST + ASR enc init.</td>
<td>16.5</td>
<td>15.5</td>
</tr>
<tr>
<td>+ MT dec init.</td>
<td>16.6</td>
<td>16.2</td>
</tr>
<tr>
<td>+ SpecAug.</td>
<td>17.2</td>
<td>16.7</td>
</tr>
<tr>
<td>ST + ASR enc init. + SpecAug.</td>
<td>17.2</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 5: Case-insensitive detokenized BLEU scores on libri-trans test set with different setups.

<table>
<thead>
<tr>
<th>Model</th>
<th>NeurST</th>
<th>ESPnet-ST</th>
</tr>
</thead>
<tbody>
<tr>
<td>pure ST</td>
<td>18.6</td>
<td>-</td>
</tr>
<tr>
<td>+ ASR enc init.</td>
<td>21.9</td>
<td>21.8</td>
</tr>
<tr>
<td>+ MT dec init.</td>
<td>22.1</td>
<td>22.3</td>
</tr>
<tr>
<td>+ SpecAug.</td>
<td>23.3</td>
<td>22.9</td>
</tr>
<tr>
<td>ST + ASR enc init. + SpecAug.</td>
<td>22.8</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 6: Case-sensitive detokenized BLEU scores on MuST-C EN-DE tst-COMMON with difference setups.
<table>
<thead>
<tr>
<th>Model</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>large MT (w/ punc. &amp; cased)</td>
<td>36.2</td>
</tr>
<tr>
<td>large MT (w/o punc. &amp; lc)</td>
<td>34.3</td>
</tr>
<tr>
<td>large cascade ST</td>
<td>31.4</td>
</tr>
<tr>
<td>large end-to-end ST</td>
<td>29.7</td>
</tr>
</tbody>
</table>

Table 7: Case-sensitive detokenized BLEU scores on MuST-C EN-DE \textit{tst-COMMON}.

### 3.7 Cascade versus End-to-End

Previous experiments on \textit{libri-trans} and MuST-C NL/PT show that the end-to-end systems have outperformed the cascade systems. Here we argue that the performance of the cascade systems above is hampered by a lack of quantitative data, and they should take advantage of large amounts of ASR and MT data separately. Hence, we further extended NeurST to large-scale scenarios and experimented on the allowed datasets for IWSLT 2021 evaluation campaign\textsuperscript{10}. We followed the practice of Zhao et al. (2021) to build our large cascade and end-to-end ST systems, which contains large-scale back-translation (Sennrich et al., 2016a) and pseudo labeling (also known as knowledge distillation) technologies. The results are illustrated in Table 7. As seen, there is a significant loss of 1.7 BLEU between end-to-end ST and cascade ST. And the cascade system would have the potential to narrow the gap to the pure MT system by introducing extra punctuation restoration and true-case modules.

Though the cascade system is superior under large data conditions, we believe future researches on self-supervised learning, knowledge distillation, and dataset construction would realize the potential of end-to-end models.

### 4 Conclusion

We introduce NeurST toolkit for easily building and training end-to-end speech translation models. We provide straightforward recipes for audio data pre-processing, training, and inference, which we believe is friendly with NLP researchers. Moreover, we report strong and reproducible benchmarks and will continuously catch up on advanced progress using NeurST, which can be regarded as the reliable baselines for the ST field.

---

\textsuperscript{10}https://iwslt.org/2021/offline

### References


