BigSSL: Exploring the Frontier of Large-Scale Semi-Supervised Learning for Automatic Speech Recognition

Daniel S. Park

Google

ISCSLP 2022
BigSSL: Exploring the Frontier of Large-Scale Semi-Supervised Learning for Automatic Speech Recognition

Yu Zhang*, Daniel S. Park*, Wei Han*, James Qin, Anmol Gulati, Joel Shor, Aren Jansen, Yuanzhong Xu, Yanping Huang, Shibao Wang, Zongwei Zhou, Bo Li, Min Ma, William Chan, Jiahui Yu, Yongqiang Wang, Liangliang Cao, Khe Chai Sim, Bhuvana Ramabhadran, Tara N. Sainath, Françoise Beaufays, Zhifeng Chen, Quoc V. Le, Chung-Cheng Chiu, Ruoming Pang† and Yonghui Wu
Table of Contents

• Introduction

• Semi-supervised Learning

• BigSSL
Introduction

Semi-supervised learning (SSL) refers to the utilization of unlabeled data for improving a supervised machine learning task.

Due to key advancements in SSL for speech recognition, we are at the point of being able to:

• Leverage large amounts of data,
• and as a result, benefit from scaling up the size of ASR models.
Introduction

In our work, we studied the effects of scaling both the dataset size and model size up, and attempted to explore the frontier of both axes of scaling.
Table of Contents

• Introduction

• Semi-supervised Learning

• BigSSL
Semi-supervised Learning (SSL)

The goal of SSL is to utilize unlabeled data to improve the performance of a task that has labeled data.
Semi-supervised Learning (SSL)

SSL has been especially instrumental for improving the performance of speech models, since data labeling for speech recognition is difficult/expensive compared to many standard ML tasks.
Semi-supervised Learning (SSL)

Methods for effectively utilizing unlabeled data can enable one to harness the full potential of large, deep models, as one is able access practically an infinite amount of data.
Emergence of Attention-based Models

Attention-based [Vaswani et al. (2017)] ASR models:

• Transformer-Transducer [Zhang et al. (2020)]
• Speech Transformer [Dong et al. (2020)]
• Conformer [Gulati et al. (2020)]
• ...
Unsupervised learning refers to a semi-supervised learning method where a loss that is helpful for learning the representation of the unlabeled data is designed, and used to train the model.

A prime example of such a task is the next-token prediction task that is used to train language models.
Unsupervised Learning

There have been a bevy of unsupervised learning methods developed for speech recognition (see Yu’s tutorial for more) but for now we stick to “Wav2Vec 2.0” by Baevski et al.
• Unsupervised “fill in the blank” task for spectrograms to “pretrain” networks (cf. BERT [Devlin et al. (2018)]) ⇒ “Fine-tuned” ASR tasks

• Spectrogram representations too dense ⇒ Compressed vector representation prediction

• Contrastive loss used to prevent cheating

• Better than NST for smaller labeled datasets
Wav2Vec 2.0
Baevski et al. (2020)

cf. BERT [Devlin et al. (2018)]
In self training, we machine-label the unlabeled data to help improve machine performance.

We want to train $S$ (student) with data labeled by $T$ (teacher), but we want $S$ to perform better than $T$. 
Noisy Student Training (NST)
Xie et al. (2020)

- Make $S$ to have larger capacity than $T$.
- Make $T$ produce labels from clean input.
- Augment the $T$-labeled data when training $S$.
- Set $S \rightarrow T$ and iterate.
Improved NST for Speech
DSP et al. (2020)
Improved NST for Speech
DSP et al. (2020)
Table of Contents

• Introduction

• Semi-supervised Learning

• BigSSL
Giant Speech Models?

Now we know how to leverage very large amounts of unlabeled data.

Can we make a giant speech model that can perform well on a wide range of speech tasks with only a small amount of labeled data?

cf. Large Language Models (GPT, LaMDA, etc.)
• Giant amount of unlabeled data: 1M hours of audio
• Giant models: 8B parameter models
• Target wide range of tasks spanning
  ▶ multiple domains
  ▶ multiple dataset sizes: $O(10)$ hrs to $O(10k)$ hrs
  ▶ semantic and non-semantic tasks
BigSSL: Main Takeaways

Voice Search Performance

Downstream ASR Tasks

Downstream Non-ASR Tasks
“Scaling-up” improves data efficiency.
"Scaling-up" pushes existing benchmarks (large or small) further.
BigSSL: Main Takeaways

Representations produced by “scaling-up” improve non-semantic tasks too.
BigSSL: Necessity of Pre-training

LibriSpeech 960h, dev-other

Voice Search 34kh, test
BigSSL: Good for Efficiency

- **P-model**: Unsupervised training.
- **PS-model**: Unsupervised + Self-Supervised training.
BigSSL: Cross-lingual Benefits
BigSSL: Deep Non-semantic Information

![Graph showing performance of different models over Conformer Layer Number]

- Conformer XL Non-RA
- Conformer XXL
- Conformer XL
- Conformer G
- ASR Encoder
- FRILL
- TRILL
- YAMNet layer 10

Average accuracy measure vs Conformer Layer Number.
Advances in SSL for speech recognition has opened a new era for ASR, where researchers are now looking beyond improving public speech benchmarks and aspiring to train giant universal speech models.
Thank you very much!
JUST: Joint Unsupervised and Supervised Training for Multilingual ASR

Bo Li

ISCSLP 2022
Based on Work

Massive Multilingual ASR

- Concerned with dealing with a large number of languages, each with large amount of data, within a single model

- In a production setting
  - Training, deploying and maintaining one model per language, especially on long tail of low-resource languages, can quickly become cumbersome as the number of languages increases
  - A single model for all languages can simplify the production pipeline significantly
  - Training multilingual ASR models on a small set of similar languages can improve recognition performance
  - Support the use case of code-switching

- Goal: A single E2E model for multiple languages
  - Improve automatic speech recognition (ASR) performance on low-resource languages
  - Overall simplify deployment of ASR systems to support diverse languages
Motivations

● Self-supervised learning
  ○ An effective method in unveiling the useful and general latent representations from large-scale unlabeled data
  ○ pretrain a sequence-to-sequence model and facilitate downstream tasks
  ○ For example, finetuning wav2vec 2.0 pretrained on 60k hours with only 1h labeled data can outperform most fully supervised models

● Concerns
  ○ catastrophic forgetting
    ■ The model might forget the previously learnt knowledge
  ○ pretrained checkpoint selection
    ■ The downstream performance varies from one checkpoint to another
    ■ The one pretrained longer is not necessarily the best one
    ■ These issues are even more severe in multilingual ASR, since different languages are often heterogeneous and the corpus is often imbalanced
    ■ (ref?)
XLSR

- 2-stage pretrain+finetune framework based on wav2vec2
- A shared quantization module over feature encoder representations produces multilingual quantized speech units/tokens
  - embeddings are then used as targets in contrastive learning
XLSR pretrain

Feature Encoder
- 7 temporal convolutional blocks
- Extract latent representations from raw input audios

Quantization module
- A Gumbel-Softmax layer + codebook (learnable)
- Assign code vectors to each encoded feature

Contrastive Module
- A stack of Transformer layers
- Extract context vectors from feature encoder output for computing the w2v2 contrastive loss

pre-training loss = $L_{w2v2} + 0.1 \cdot L_{div}$

Thanks to @andyyuan for sharing slide components
XLSR finetune

Decoder
- convolutional language model
Joint Training

- We propose to train the model with both supervised and unsupervised losses jointly
  - Reconcile the gradients provided by both types of losses
  - Mutually regularize each other

- Unsupervised losses
  - Contrastive loss
  - Masked language model (MLM) loss

- Supervised loss
  - RNN-T
Feature Encoder
- 2 Convolutional layers
- Extract latent representations from the surface features (log-mel filter bank)
- Filter size 3x3, stride 2
- Down-sample (4x length reduction)

Quantization module
- Encoded features are passed to a quantizer without masking
- Quantizer “summarizes” all the latent speech representations to a finite set (codebook) of representative discriminative speech tokens
- Codebook learnable
- Output both the quantized token + token ID
JUST

Contrastive Module
- A stack of Conformer blocks
- Read the encoded features with masking
  - sample 6.5% of all time steps and replace each of the selected time steps and its subsequent 10 time steps with random normal vectors
- Extract context vectors from feature encoder output for computing the w2v2 contrastive loss

MLM Module
- A stack of Conformer blocks
- Continue to extract context vectors (from the contrastive module’s output) for computing the MLM loss
- Cross-entropy with ground-truth token IDs

Decoder Module
- 2-layer LSTM
- RNN-T loss

Final JUST loss = \( L_C + L_m + 0.1 \cdot L_d + L_s \)
Multilingual LibriSpeech (MLS)

- Monolingual LibriSpeech
  - derived from the LibriVox data
  - ships with about 1000 hours of labeled audio, obtained by leveraging alignments between textbooks and their read (audio) counterpart
  - A great success as a standard, freely available ASR benchmark

- LibriVox top 15 languages

<table>
<thead>
<tr>
<th>Language</th>
<th>Hours</th>
<th>Books</th>
<th>Speakers</th>
</tr>
</thead>
<tbody>
<tr>
<td>English</td>
<td>71,506.79</td>
<td>12421</td>
<td>4214</td>
</tr>
<tr>
<td>German</td>
<td>3,287.48</td>
<td>593</td>
<td>244</td>
</tr>
<tr>
<td>Dutch</td>
<td>2,253.68</td>
<td>206</td>
<td>91</td>
</tr>
<tr>
<td>Spanish</td>
<td>1,438.41</td>
<td>285</td>
<td>120</td>
</tr>
<tr>
<td>French</td>
<td>1,333.35</td>
<td>224</td>
<td>114</td>
</tr>
<tr>
<td>Multilingual*</td>
<td>516.82</td>
<td>130</td>
<td>19</td>
</tr>
<tr>
<td>Portuguese</td>
<td>284.59</td>
<td>68</td>
<td>31</td>
</tr>
<tr>
<td>Italian</td>
<td>279.43</td>
<td>61</td>
<td>28</td>
</tr>
<tr>
<td>Russian</td>
<td>172.34</td>
<td>44</td>
<td>29</td>
</tr>
<tr>
<td>Latin</td>
<td>138.93</td>
<td>20</td>
<td>16</td>
</tr>
<tr>
<td>Polish</td>
<td>137.00</td>
<td>25</td>
<td>16</td>
</tr>
<tr>
<td>Church Slavonic</td>
<td>136.42</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>Hebrew</td>
<td>125.72</td>
<td>23</td>
<td>13</td>
</tr>
<tr>
<td>Japanese</td>
<td>97.67</td>
<td>38</td>
<td>24</td>
</tr>
<tr>
<td>Ancient Greek</td>
<td>69.77</td>
<td>43</td>
<td>8</td>
</tr>
</tbody>
</table>
Multilingual LibriSpeech (MLS)

- While English is the most dominant language in LibriVox, there is a significant amount of audio hours present for other languages.

- MLS selects English (en), German (de), Dutch (nl), Spanish (es), French (fr), Portuguese (pt), Italian (it), Polish (pl)
  - Based on the number of audiobook hours and the availability of the corresponding text sources.
  - All the audio data is downsampling from 48kHz to 16kHz for further processing.
Multilingual LibriSpeech (MLS)

- Audio segmentation
  - Raw audio files from LibriVox vary from few minutes to hours
  - Segment each sequence at silence frames or 20s mark
    - Each segment between 10s to 20s
  - Models trained on their in-house datasets consisting of audios (from videos) publicly shared by users

- Pseudo label generation
  - A beam-search decoding with a 4-gram language model on the same acoustic models used for audio segmentation
Multilingual LibriSpeech (MLS)

- Text sources for audiobook data
  - Online parsers and manual effort
  - Text normalization
    - NFKC normalization
    - remove all the unwanted characters like punctuations, subscript/superscripts, etc.
    - prepare a list of valid unicode characters based on the language’s orthography

- Transcript retrieval
  - Split the source text into multiple overlapping documents of 1250 words each and striding by 1000 words
  - Retrieve the documents which best matches with the pseudo label for the audio segments
    - using term-frequency inverse document-frequency (TF-IDF) similarity score
Multilingual LibriSpeech (MLS)

- Order the speakers based on their durations
  - If lower than the threshold, assigned for training
  - If higher than the threshold, assigned for validation and testing

<table>
<thead>
<tr>
<th>Language</th>
<th>Duration (hrs)</th>
<th># Speakers</th>
<th># Hours / Gender</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>train</td>
<td>dev</td>
<td>test</td>
</tr>
<tr>
<td>English</td>
<td>44,659.74</td>
<td>15.75</td>
<td>15.55</td>
</tr>
<tr>
<td>German</td>
<td>1,966.51</td>
<td>14.28</td>
<td>14.29</td>
</tr>
<tr>
<td>Dutch</td>
<td>1,554.24</td>
<td>12.76</td>
<td>12.76</td>
</tr>
<tr>
<td>French</td>
<td>1,076.58</td>
<td>10.07</td>
<td>10.07</td>
</tr>
<tr>
<td>Spanish</td>
<td>917.68</td>
<td>9.99</td>
<td>10</td>
</tr>
<tr>
<td>Italian</td>
<td>247.38</td>
<td>5.18</td>
<td>5.27</td>
</tr>
<tr>
<td>Portuguese</td>
<td>160.96</td>
<td>3.64</td>
<td>3.74</td>
</tr>
<tr>
<td>Polish</td>
<td>103.65</td>
<td>2.08</td>
<td>2.14</td>
</tr>
</tbody>
</table>
Monolingual baselines

- Monolingual Baseline WER with different decoding strategies using wav2letter++

<table>
<thead>
<tr>
<th>Language</th>
<th>Viterbi</th>
<th>Zero LM</th>
<th>5-gram LM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dev</td>
<td>test</td>
<td>dev</td>
</tr>
<tr>
<td>English</td>
<td>6.01</td>
<td>6.99</td>
<td>5.89</td>
</tr>
<tr>
<td>German</td>
<td>5.55</td>
<td>6.93</td>
<td>5.80</td>
</tr>
<tr>
<td>Dutch</td>
<td>17.00</td>
<td>13.18</td>
<td>16.45</td>
</tr>
<tr>
<td>French</td>
<td>7.79</td>
<td>6.88</td>
<td>7.43</td>
</tr>
<tr>
<td>Spanish</td>
<td>5.94</td>
<td>6.90</td>
<td>5.90</td>
</tr>
<tr>
<td>Italian</td>
<td>14.55</td>
<td>12.35</td>
<td>14.01</td>
</tr>
<tr>
<td>Portuguese</td>
<td>18.62</td>
<td>21.70</td>
<td>17.22</td>
</tr>
<tr>
<td>Polish</td>
<td>19.25</td>
<td>19.40</td>
<td>18.73</td>
</tr>
</tbody>
</table>
XLSR results

- XLSR is pretrained on the 53 languages of MLS, CommonVoice and BABEL
- Finetuned on MLS **per language**
  - Outperforms monolingual baseline on **nl, it, pt, pl**

<table>
<thead>
<tr>
<th>Model</th>
<th>#pt</th>
<th>#ft</th>
<th>en</th>
<th>de</th>
<th>nl</th>
<th>fr</th>
<th>es</th>
<th>it</th>
<th>pt</th>
<th>pl</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of training hours</td>
<td>44.7k</td>
<td>2k</td>
<td>1.6k</td>
<td>1.1k</td>
<td>918</td>
<td>247</td>
<td>161</td>
<td>104</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LibriVox</td>
<td>1</td>
<td>1-1h</td>
<td>13.2</td>
<td>25.5</td>
<td>30.3</td>
<td>37.0</td>
<td>22.5</td>
<td>23.3</td>
<td>37.8</td>
<td>38.4</td>
</tr>
<tr>
<td>LibriVox</td>
<td>1</td>
<td>1-10h</td>
<td>10.6</td>
<td>14.5</td>
<td>19.6</td>
<td>19.6</td>
<td>13.5</td>
<td>16.7</td>
<td>25.0</td>
<td>32.0</td>
</tr>
<tr>
<td>XLSR-53</td>
<td>53</td>
<td>1-1h</td>
<td>17.3</td>
<td>10.6</td>
<td>15.6</td>
<td>17.0</td>
<td>10.4</td>
<td>15.1</td>
<td>21.4</td>
<td>31.9</td>
</tr>
<tr>
<td>XLSR-53</td>
<td>53</td>
<td>1-10h</td>
<td>14.6</td>
<td>8.4</td>
<td>12.8</td>
<td>12.5</td>
<td>8.9</td>
<td>13.4</td>
<td>18.2</td>
<td>21.2</td>
</tr>
<tr>
<td>XLSR-53</td>
<td>53</td>
<td>1-100h</td>
<td>13.2</td>
<td>7.4</td>
<td>10.9</td>
<td>9.8</td>
<td>7.9</td>
<td>12.0</td>
<td>15.7</td>
<td>18.9</td>
</tr>
<tr>
<td>XLSR-53</td>
<td>53</td>
<td>1-full</td>
<td>-</td>
<td>7.0</td>
<td>10.8</td>
<td>7.6</td>
<td>6.3</td>
<td>10.4</td>
<td>14.7</td>
<td>17.2</td>
</tr>
<tr>
<td>Pratap et al. (2020)</td>
<td>-</td>
<td>1-full</td>
<td><strong>5.88</strong></td>
<td><strong>6.49</strong></td>
<td>12.02</td>
<td><strong>5.58</strong></td>
<td><strong>6.07</strong></td>
<td>10.54</td>
<td>19.49</td>
<td>20.39</td>
</tr>
</tbody>
</table>
Experiments

- Training schemes
  - Loss components: contrastive loss ($L_c$), MLM loss ($L_m$), diversity ($L_d$), RNN-T ($L_s$)
  - Single-stage, two-stage

- Single-stage
  - $L_s$
  - $L_s + L_c + 0.1 x L_d$
  - $L = L_s + L_c + 0.1 x L_d + L_m$ (JUST)

- Two-stage
  - Pretrain
    - $L_c + 0.1 x L_d$
    - $L_u = L_m + L_c + 0.1 x L_d$
    - $L$
  - Finetune
    - $L_s$: Pure finetune, no updates on codebook
    - $L$: Joint finetune: updates on codebook on top of the one learnt in pretraining
Experiments

- Denote the unsupervised loss as $L_u$, supervised loss as $L_s$
- $L = L_s + \beta L_u$

- Compared methods
  - XLSR
  - B0 (size roughly same as JUST), E3 (1B) from transfer learning methods
### Baselines

- **JUST ($\beta=0$)** is simply a standard supervised baseline using JUST’s architecture.

<table>
<thead>
<tr>
<th>Method</th>
<th>External data</th>
<th>de</th>
<th>en</th>
<th>es</th>
<th>fr</th>
<th>it</th>
<th>nl</th>
<th>pl</th>
<th>pt</th>
<th>Avg</th>
<th>Avg (w/o en)</th>
</tr>
</thead>
<tbody>
<tr>
<td>XLSR-53 [16]</td>
<td>Y</td>
<td>7.0</td>
<td>-</td>
<td>6.3</td>
<td>7.6</td>
<td>10.4</td>
<td>10.8</td>
<td>17.2</td>
<td>14.7</td>
<td>10.6</td>
<td>10.6</td>
</tr>
<tr>
<td>B0 (random init.) [13]</td>
<td>Y</td>
<td>5.5</td>
<td>6.1</td>
<td>5.8</td>
<td>6.9</td>
<td>11.9</td>
<td>11.9</td>
<td>15.4</td>
<td>16.2</td>
<td>10.0</td>
<td>10.5</td>
</tr>
<tr>
<td>B0 (15-language model init.) [13]</td>
<td>Y</td>
<td>5.0</td>
<td>6.6</td>
<td>4.7</td>
<td>6.1</td>
<td>10.1</td>
<td>11.1</td>
<td>10.9</td>
<td>15.5</td>
<td>8.8</td>
<td>9.1</td>
</tr>
<tr>
<td>E3 (15-language model init.) [13]</td>
<td>Y</td>
<td>4.3</td>
<td>5.8</td>
<td>4.2</td>
<td>4.9</td>
<td>8.8</td>
<td>9.9</td>
<td>10.4</td>
<td>15.2</td>
<td>7.9</td>
<td>8.2</td>
</tr>
<tr>
<td>JUST ($\beta = 0$)</td>
<td>N</td>
<td>5.5</td>
<td>6.9</td>
<td>4.1</td>
<td>6.0</td>
<td>9.3</td>
<td>10.3</td>
<td>11.3</td>
<td>9.4</td>
<td>7.8</td>
<td>8.0</td>
</tr>
</tbody>
</table>
On average WER of all 8 languages, all JUST-based methods outperform previous works. In particular, JUST (with $\beta=0.07$) outperforms the monolingual baseline with 5-gram LM by 33.3%, XLSR-53 by 32.0%, B0 by 18.2%, E3 by 8.8%.

If we exclude English WER, JUST outperforms monolingual, XLSR-53, B0, E3 by 36.5%, 31.1%, 19.8%, 11.0% respectively. Compared to JUST with $\beta=0$, JUST with joint training improves the average WER by 7.7%.
Per language

B0 vs JUST

JUST (β=0) vs JUST

E3 vs JUST
JUST finetune

- Two finetuning schemes are attempted
- First, we take a pretrained checkpoint trained with $L_u$, and finetune it with JUST objective $L$
  - Compared to Pretrain+pure Finetune, it also improves on de, en, fr, it, nl, pt
  - It is interesting to compare JUST and JUST Finetune on pt, pl. Different training schemes lead to different quantized tokens and cause the discrepancy. Empirically, JUST from scratch can better facilitate the low-resource languages and reduce WER of each language to below 10

<table>
<thead>
<tr>
<th>Method</th>
<th>External data</th>
<th>de</th>
<th>en</th>
<th>es</th>
<th>fr</th>
<th>it</th>
<th>nl</th>
<th>pl</th>
<th>pt</th>
<th>Avg</th>
<th>Avg (w/o en)</th>
</tr>
</thead>
<tbody>
<tr>
<td>JUST ($\beta = 0$)</td>
<td>N</td>
<td>5.5</td>
<td>6.9</td>
<td>4.1</td>
<td>6.0</td>
<td>9.3</td>
<td>10.3</td>
<td>11.3</td>
<td>9.4</td>
<td>7.8</td>
<td>8.0</td>
</tr>
<tr>
<td>Pretrain ($L_u$)+pure Finetune ($L_s$)</td>
<td>N</td>
<td>4.3</td>
<td>6.6</td>
<td>3.8</td>
<td>5.0</td>
<td>9.1</td>
<td>9.9</td>
<td>8.1</td>
<td>14.6</td>
<td>7.7</td>
<td>7.8</td>
</tr>
<tr>
<td>Pretrain ($L_u$)+JUST Finetune ($\mathcal{L}$)</td>
<td>N</td>
<td>4.2</td>
<td>6.6</td>
<td>4.0</td>
<td>5.0</td>
<td>9.0</td>
<td>9.5</td>
<td>7.6</td>
<td>15.1</td>
<td>7.6</td>
<td>7.8</td>
</tr>
<tr>
<td>JUST ($\mathcal{L}$)</td>
<td>N</td>
<td>4.6</td>
<td>6.8</td>
<td>3.9</td>
<td>5.7</td>
<td>9.1</td>
<td>9.9</td>
<td>9.1</td>
<td>8.6</td>
<td>7.2</td>
<td>7.3</td>
</tr>
<tr>
<td>JUST ($\mathcal{L}$)+pure Finetune ($L_s$)</td>
<td>N</td>
<td>4.2</td>
<td>6.7</td>
<td>3.9</td>
<td>5.6</td>
<td>8.1</td>
<td>9.8</td>
<td>7.2</td>
<td>9.5</td>
<td>6.9</td>
<td>6.9</td>
</tr>
</tbody>
</table>
Second, we take a checkpoint from JUST trained from scratch, and finetune it with only supervised loss $L_s$.

- It further improves over JUST and achieves the best average WER.
- On de, it, pl, this scheme outperforms all compared methods and remains competitive on other languages.

<table>
<thead>
<tr>
<th>Method</th>
<th>External data</th>
<th>de</th>
<th>en</th>
<th>es</th>
<th>fr</th>
<th>it</th>
<th>nl</th>
<th>pl</th>
<th>pt</th>
<th>Avg</th>
<th>Avg (w/o en)</th>
</tr>
</thead>
<tbody>
<tr>
<td>JUST ($\beta = 0$)</td>
<td>N</td>
<td>5.5</td>
<td>6.9</td>
<td>4.1</td>
<td>6.0</td>
<td>9.3</td>
<td>10.3</td>
<td>11.3</td>
<td>9.4</td>
<td>7.8</td>
<td>8.0</td>
</tr>
<tr>
<td>Pretrain ($\mathcal{L}_u$)+pure Finetune ($\mathcal{L}_s$)</td>
<td>N</td>
<td>4.3</td>
<td>6.6</td>
<td>3.8</td>
<td>5.0</td>
<td>9.1</td>
<td>9.9</td>
<td>8.1</td>
<td>14.6</td>
<td>7.7</td>
<td>7.8</td>
</tr>
<tr>
<td>Pretrain ($\mathcal{L}_u$)+JUST Finetune ($\mathcal{L}$)</td>
<td>N</td>
<td>4.2</td>
<td>6.6</td>
<td>4.0</td>
<td>5.0</td>
<td>9.0</td>
<td>9.5</td>
<td>7.6</td>
<td>15.1</td>
<td>7.6</td>
<td>7.8</td>
</tr>
<tr>
<td>JUST ($\mathcal{L}$)</td>
<td>N</td>
<td>4.6</td>
<td>6.8</td>
<td>3.9</td>
<td>5.7</td>
<td>9.1</td>
<td>9.9</td>
<td>9.1</td>
<td>8.6</td>
<td>7.2</td>
<td>7.3</td>
</tr>
<tr>
<td>JUST ($\mathcal{L}$)+pure Finetune ($\mathcal{L}_s$)</td>
<td>N</td>
<td>4.2</td>
<td>6.7</td>
<td>3.9</td>
<td>5.6</td>
<td>8.1</td>
<td>9.8</td>
<td>7.2</td>
<td>9.5</td>
<td><strong>6.9</strong></td>
<td><strong>6.9</strong></td>
</tr>
</tbody>
</table>
When $\beta=0.07$, the unsupervised and the supervised losses are balanced, resulting in the best performance.

<table>
<thead>
<tr>
<th>Method</th>
<th>Ext.</th>
<th>de</th>
<th>en</th>
<th>es</th>
<th>fr</th>
<th>it</th>
<th>nl</th>
<th>pl</th>
<th>pt</th>
<th>Avg</th>
<th>Avg (w/o en)</th>
</tr>
</thead>
<tbody>
<tr>
<td>JUST ($\beta = 0$)</td>
<td>N</td>
<td>5.5</td>
<td>6.9</td>
<td>4.1</td>
<td>6.0</td>
<td>9.3</td>
<td>10.3</td>
<td>11.3</td>
<td>9.4</td>
<td>7.8</td>
<td>8.0</td>
</tr>
<tr>
<td>JUST ($\beta = 0.03$)</td>
<td>N</td>
<td>5.0</td>
<td>7.4</td>
<td>4.3</td>
<td>6.3</td>
<td>9.3</td>
<td>10.3</td>
<td>8.7</td>
<td>9.1</td>
<td>7.5</td>
<td>7.6</td>
</tr>
<tr>
<td>JUST ($\beta = 0.05$)</td>
<td>N</td>
<td>5.2</td>
<td>6.8</td>
<td>4.4</td>
<td>5.7</td>
<td>9.4</td>
<td>9.9</td>
<td>9.3</td>
<td>8.8</td>
<td>7.4</td>
<td>7.5</td>
</tr>
<tr>
<td>JUST ($\beta = 0.07$)</td>
<td>N</td>
<td>4.6</td>
<td>6.8</td>
<td>3.9</td>
<td>5.7</td>
<td>9.1</td>
<td>9.9</td>
<td>9.1</td>
<td>8.6</td>
<td>7.2</td>
<td>7.3</td>
</tr>
<tr>
<td>JUST ($\beta = 0.1$)</td>
<td>N</td>
<td>5.8</td>
<td>6.8</td>
<td>4.1</td>
<td>5.8</td>
<td>10.3</td>
<td>10.0</td>
<td>9.7</td>
<td>8.6</td>
<td>7.6</td>
<td>7.8</td>
</tr>
</tbody>
</table>
Attention

- We compare two attention mechanisms for JUST from scratch:
  - a local attention mechanism with both left and right context
  - a global attention mechanism with full context
  - Global attention clearly outperforms local attention on all languages.
Codebook

- Original w2v2 doesn't update codebook in the finetuning phase. JUST finetuning, however, keeps the unsupervised loss and could further update the codebook.
- We compare the performance with learnable or fixed codebook during JUST finetuning:
  - Their results are close.
  - This implies that fixing codebook in JUST finetuning would not degrade the performance.
Given the observation that updating codebook or not might not affect the final performance, it is important to learn a good codebook at the beginning. L_u vs L, leading to different codebooks and thus affecting downstream performance.
Pretrained checkpoints

- Different checkpoints can lead to different downstream performance. The later checkpoints do not necessarily lead to better downstream WERs
- To verify this, we finetune multiple pretrained checkpoints and evaluate their finetuning quality
  - Despite the constantly descending unsupervised loss $L_u$, downstream average WERs don't follow the same trend
Conclusion

- In our work, we propose a novel uniform multilingual ASR system JUST for the end-to-end speech recognition on multiple languages, by jointly training supervised and unsupervised losses.

- On low-resource languages, our JUST can consistently bring gains and boost performance.

- In the future, we will investigate:
  - How the objective function affects the codebook learning
  - Explore the joint training with more languages and other unsupervised losses
  - Incorporate text into joint training
  - More datasets (VS, YT, etc.)
Text Injection for speech pre-training

Yu Zhang

ISCSLP 2022

Slides made by Yu Zhang, Colin Cherry, Zhehuai Chen and Rohit Prabhavalkar

Google Research
Based on Work


To build a pre-trained encoder that can encode speech or text in many different languages into compatible representations.

- Pre-training allows us to leverage large amounts unlabeled speech and text data to prime our networks, so our supervised data goes further.

- **Multilingual** to share representations and data across languages, improving performance on low-resource languages.
  - **Multi-modal** to do the same, but across modalities.
Why now?

- We are seeing a convergence in neural architectures for artificial intelligence.
  - The differences between a speech recognizer and a machine translation system become smaller each year.
- Pre-training on unlabeled data is seeing a similar convergence:
  - The masked language modeling strategies that have been so successful for text are seeing as much or more success for speech.
Motivating problems

- For medium-resource languages, availability of speech data can be a bottleneck:
  - There is a lot of speech, but far far more text.
  - This goes for both labeled and unlabeled data.
  - It would be nice to leverage text data for speech understanding tasks.

- For very low-resource languages, the situation can be reversed
  - There may be more spoken language available through sources like YouTube videos than there is through text.

- Ideally, this technology would help both scenarios.
Background: BERT – masked language modeling for text
Devlin et al. (2018)

- Now ubiquitous in NLP
- Typically mask about 15% of input text
- SpanBERT (Joshi et al., 2020): replaces random masks with masks of contiguous spans

```
the [MASK] blue
```

```
Transformers
```

```
curtains were
```

Transformers
Background: w2v-BERT – self-supervised modeling of speech
Chung et al., 2021

Conformer → MLM Loss

Contrastive Loss

Discrete Features → Diversity loss

Conformer

Masking

Features

Quantization

Speech ConvNet

Features
Background: TLM – masked languaging modeling for bitext
Lample and Conneau (2019)

- A method to make use of parallel data
- Transformer self-attention allows sharing bilingual information through concatenated bitext, which in turn encourages cross-lingual representation similarity.
Background: Multimodal TLM
Zheng et al, (2021)

- Text should be a transcript of the audio
- Q’s are quantiles of the masked audio, as in w2v-BERT
Background: Connectionist Temporal Classification for ASR

CTC Loss Dynamic Program
Enumerate all ways to remove _ and collapse duplicates

Certain

Conformer
mSLAM: Multilingual encoder for speech and text

**Figure 1: Multilingual Speech-Text Pretraining** We pre-train a large multilingual speech-text Conformer on 429K hours of unannotated speech data in 51 languages, 15TBs of unannotated text data in 101 languages, as well as 2.3k hours of speech-text ASR data.
Zooming in: bottom layers

- **Character embeddings**
  - Absolutely crucial that these be characters
  - Improves length match and phonetic sharing between text and speech

- **Speech Convnet**
  - Uses 1D convolutions to perform a 4x length reduction to improve length match
Directly asks the speech representations to be interpretable as text
Crucial to use the same softmax over character as used by BERT
  ○ Otherwise, speech and text representations would have no reason to merge
Implemented with TLM inputs, meaning we see the characters on the input and the output
  ○ Makes the learning task easier, but prevents it from taking over

C T E r t t a i n n

CTC Loss Dynamic Program
Enumerate all ways to remove _ and collapse duplicates

Text input is used only for Conformer self attention

Conformer
Tasks

- Multilingual Speech Understanding
  - Translation
  - Classification
  - Recognition
- Multilingual Text Understanding
  - Translation
  - Classification
- Will always compare to a speech-only baseline
  - Identical architecture, leave out all text and supervised losses
    - Only w2v-BERT loss

Google Research
Multilingual Speech Translation

- CoVoST-2 Many-to-English speech translation task
  - Audio is read text collected through the Common Voice Project
  - Transcripts are professionally translated into English
  - 21 source languages
  - Tiny by MT standards:
    - Biggest is French-English with 200k sentence pairs
    - Smallest is Japanese-English with 1k sentence pairs

- Fine tune with a 6-layer Transformer decoder
  - Decoder is tiny (34M params) compared to the mSLAM encoder (0.6B params)
  - Languages are mixed together
    - No attempt to identify different source languages
    - No upweighting languages with fewer data
Speech Translation Results

CoVoST-2 21-to-English Average BLEU

<table>
<thead>
<tr>
<th>Model</th>
<th>BLEU Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>XLS-R 2.0B</td>
<td>22.1</td>
</tr>
<tr>
<td>Speech-Only 0.6B</td>
<td>20.4</td>
</tr>
<tr>
<td>mSLAM 0.6B</td>
<td>20.6</td>
</tr>
</tbody>
</table>

*XLS-R: Self-supervised Cross-lingual Speech Representation Learning at Scale, Babu et al. 2021 ([arxiv](https://arxiv.org))
A competing pre-trained multilingual speech encoder
Adding MT data with Joint Fine Tuning

- mSLAM was built to consume both speech and text during pre-training
  - This makes it ideal to also consume speech and text during fine-tuning
  - We still evaluate on just the ST task

- Add in MT data using the same encoder
  - Each batch has an equal number of speech and text examples
  - Using relevant WMT when available; otherwise using TED talks
  - Now have much more data:
    - Millions for the WMT languages
    - Tens of thousands for the TED languages
  - Put 5x more weight on text during fine-tuning, decrease drop-out
    - (tuned on Speech-only baseline)
ST + MT Results

CoVoST-2 21-to-English Average BLEU

<table>
<thead>
<tr>
<th>Model</th>
<th>ST</th>
<th>ST+MT</th>
</tr>
</thead>
<tbody>
<tr>
<td>XLS-R 2.0B</td>
<td>22.1</td>
<td></td>
</tr>
<tr>
<td>Speech-Only 0.6B</td>
<td>20.4</td>
<td>21</td>
</tr>
<tr>
<td>mSLAM 0.6B</td>
<td>20.6</td>
<td>22.4</td>
</tr>
</tbody>
</table>
ST: Scaling up

CoVoST-2 21-to-English Average BLEU

<table>
<thead>
<tr>
<th>Model</th>
<th>ST</th>
<th>ST+MT</th>
</tr>
</thead>
<tbody>
<tr>
<td>XLS-R 2.0B</td>
<td>22.1</td>
<td></td>
</tr>
<tr>
<td>Speech-Only 0.6B</td>
<td>20.4</td>
<td>21</td>
</tr>
<tr>
<td>mSLAM 0.6B</td>
<td>20.6</td>
<td>22.4</td>
</tr>
<tr>
<td>mSLAM 2.0B</td>
<td>22.4</td>
<td>24.8</td>
</tr>
</tbody>
</table>
Speech Classification

- **MINDS-14**: Spoken intent classification
  - 14 intents, 14 languages; merged all languages into a single dataset
  - Intents: goals in a natural language interface:
    - IE: cash withdrawal, currency exchange rates or using credit cards abroad

- **Fleurs-LangID**: Spoken Language Identification:
  - Speech extension of FLORES multilingual MT benchmark
  - 1,000 training examples per language for 102 languages (400 dev, 500 test)

- **Fine-tuning**:
  - Max-pooled mSLAM-encoded representations, then fed into a classification layer
Speech Classification Results

Speech Classification Accuracy

- Speech-Only 0.6B
- mSLAM 0.6B
- mSLAM 2.0B

- MINDS-14
- Fleurs-LangID
Multilingual Speech Recognition (ASR)

- **VoxPopuli:**
  - 14 languages, Euorparl event recordings (>1,700 hours)
- **MLS-10Hr:**
  - 8 languages from Librispeech (read books), 10-hour (per language)
- **Babel:**
  - Telephone speech from Africa and Asia, train and test on a 5-language subset
- **Fine-tuning:**
  - 2-layer LSTM with RNN-Transducer loss
  - No language-model fusion
ASR Results

*XLS-R results are for 1B for VoxPop, 2B are not available
*XLR-R results benefit from LM Fusion for Babel
At this stage, our goal was not to improve text modeling; however, looking at text classification results can be instructive.

**XNLI: Cross-lingual natural language inference**
- Sentence-pair classification (entailment, neutral, contradiction)
- 15 languages
- Zero-shot: Train on English, test on other languages
- Translate-train: Train on English + machine translations into other languages

**Fine-tuning**
Text Classification: Results

mT5: An encode-decoder pre-training scheme trained on the same multilingual text data at mSLAM
Summary

- We have presented the first massively multilingual, speech + text pre-trained encoder
- Establishes a new state-of-the-art on X-to-English Speech Translation
- Strong results on Speech Classification, Speech Recognition, Text Classification.
- We still behind on ASR task, with same model capacity.
Can we match the two modality better?

- **Sequence** self-alignment
- **Modality matching** in the intermediate layer
- Reuse **duration** part of Parallel Tacotron
- **Unified** framework for text-speech representation learning

*Figure 1: Proposed architecture of Maestro to learn unified representations from speech and text. The purple and red boxes denote differences from mSLAM [11]. The Text Encoder block utilizes alignments to explicitly resample text representations $c_t$ to match speech encoder output $c_s$.***
Data and tasks

<table>
<thead>
<tr>
<th>Task</th>
<th>Lang</th>
<th>Speech (hours)</th>
<th>Text</th>
<th>Paired speech</th>
</tr>
</thead>
<tbody>
<tr>
<td>SpeechStew</td>
<td><strong>Monolingual ASR</strong></td>
<td>1 (5 genres)</td>
<td>60k</td>
<td>5k</td>
</tr>
<tr>
<td>VoxPopuli</td>
<td><strong>Multilingual ASR</strong></td>
<td>14</td>
<td>430k</td>
<td>1.3k</td>
</tr>
<tr>
<td>CoVoST</td>
<td><strong>Speech-to-text Translation</strong></td>
<td>21</td>
<td>430k</td>
<td>2.9k</td>
</tr>
</tbody>
</table>
## Monolingual ASR

<table>
<thead>
<tr>
<th>Method</th>
<th>Librispeech</th>
<th>AMI</th>
<th>TED</th>
<th>swb/fisher</th>
<th>CV</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>clean</td>
<td>other</td>
<td>ihm</td>
<td>sdm</td>
<td>swb</td>
</tr>
<tr>
<td>W2v-BERT +LM</td>
<td>1.6</td>
<td>3</td>
<td>9.1</td>
<td>23.1</td>
<td>5.4</td>
</tr>
<tr>
<td></td>
<td>1.5</td>
<td>2.8</td>
<td>-</td>
<td>-</td>
<td>4.5</td>
</tr>
<tr>
<td>SLAM</td>
<td>1.6</td>
<td>3.1</td>
<td>9.3</td>
<td>23.5</td>
<td>5.6</td>
</tr>
<tr>
<td>Tts4pretrain2</td>
<td>1.6</td>
<td>2.8</td>
<td>8.7</td>
<td>21.9</td>
<td>5.0</td>
</tr>
<tr>
<td>Maestro +LM</td>
<td>1.5</td>
<td>2.8</td>
<td>8.5</td>
<td>21.9</td>
<td>4.9</td>
</tr>
<tr>
<td></td>
<td>1.5</td>
<td>2.7</td>
<td>-</td>
<td>-</td>
<td>4.3</td>
</tr>
<tr>
<td>Best-RQ +LM</td>
<td>1.6</td>
<td>2.9</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>1.5</td>
<td>2.7</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

1. TTS model is trained on LibriTTS.

On par with or better than SLAM and tts4pretrain2
Multilingual ASR: Voxpopuli (14 languages)

<table>
<thead>
<tr>
<th>Method</th>
<th>Model size</th>
<th>Pretrain</th>
<th>text</th>
<th>paired data</th>
<th>Avg WER</th>
</tr>
</thead>
<tbody>
<tr>
<td>XLS-R</td>
<td>1B</td>
<td>437k</td>
<td>-</td>
<td>-</td>
<td>10.6</td>
</tr>
<tr>
<td>w2v-bert</td>
<td>0.6B</td>
<td>429k</td>
<td>-</td>
<td>-</td>
<td>8.8</td>
</tr>
<tr>
<td>Maestro</td>
<td>0.6B</td>
<td>429k</td>
<td>VP-T + mC4</td>
<td>2.4k</td>
<td>8.1</td>
</tr>
</tbody>
</table>

#1 New state-of-the-art
#2 Can be extended to cover 100 languages from mC4

Details in the submission.
Does Maestro work on other tasks?

**Speech-to-text Translation** *(ST, 21 languages->en)*

<table>
<thead>
<tr>
<th>Method</th>
<th>Model size</th>
<th>Pretraining Data</th>
<th>Avg BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Finetune: ST-only; mBART decoder init</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XLS-R 1B</td>
<td>1B</td>
<td>437k</td>
<td>19.3</td>
</tr>
<tr>
<td>XLS-R 2B</td>
<td>2B</td>
<td>437k</td>
<td>22.1</td>
</tr>
<tr>
<td>Finetune: ST and Machine translation (MT) jointly</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>w2v-bert 0.6B</td>
<td>0.6B</td>
<td>429k</td>
<td>21.0</td>
</tr>
<tr>
<td>mSLAM 0.6B</td>
<td>0.6B</td>
<td>429k, mC4</td>
<td>22.4</td>
</tr>
<tr>
<td>mSLAM 2B</td>
<td>2B</td>
<td>429k, mC4</td>
<td>24.8</td>
</tr>
<tr>
<td>Maestro 0.6B</td>
<td>0.6B</td>
<td>429k, VP-T + mC4</td>
<td>24.3</td>
</tr>
<tr>
<td><strong>Maestro</strong> 0.6B</td>
<td><strong>0.6B</strong></td>
<td><strong>429k, VP-T + mC4</strong></td>
<td><strong>25.2</strong></td>
</tr>
</tbody>
</table>

Numbers other than Maestro from "mSLAM: Massively multilingual joint pre-training for speech and text." [link](#).
Moving to Prod: JOIST
Review: Cascaded Encoder [Narayanan, 2021]

- 1st-pass:
  - causal encoder → decoder
  - Low output latency decoder can be used to quickly display results to the screen
- 2nd-pass:
  - additional non-causal layers (1 second right context) → decoder
  - outputs of the non-causal decoder can be computed in parallel and displayed when we finalize
- Quality improvement with beam search while still being able to run in real time
- To unify server and on-device, and address latency issues, we use the following [Sainath, 2022]
  - 1st-pass: ~50M
  - 2nd-pass: ~100M

References:
JOIST

- Works in a **streaming**, production-style cascaded encoder model
- Inputs:
  - Paired speech-text input $S=\{(x_s, y_s)\}$
    - $x_s$: log-mel features
    - $y_s$: wordpiece outputs
  - Unpaired text-only data $T=\{(x_t, y_t)\}$
    - $x_t$: text input (phonemes or wordpieces)
    - $y_t$: wordpiece outputs
    - Note $x_t$ and $y_t$ derived from the same unpaired text
- We build on the cascaded encoder framework for JOIST
- 1st-pass:
  - $x_s \rightarrow P_C(y_s|x_s)$
  - $x_t \rightarrow P_C(y_t|x_t)$
- 2nd-pass
  - $x_s \rightarrow P_{NC}(y_s|x_s)$
  - $x_t \rightarrow P_{NC}(y_t|x_t)$
- Bulk of computation (100M params) in shared cascaded encoder!
A main novelty of JOIST is **joint training** with both speech and text modalities.

Denote:

- \( L_C(x,y) = -\log P_C(y|x) \)
- \( L_{NC}(x,y) = -\log P_{NC}(y|x) \)

Loss computation **jointly optimizes** both causal and non-causal decoders using paired audio-text as well as unpaired text.

\[
L_{CE} = \lambda_1 [L_C(y_s, x_s) + L_{NC}(y_s, x_s)] \\
+ \lambda_2 [L_C(y_t, x_t) + L_{NC}(y_t, x_t)]
\]
Objectives
- Simplicity
- Parameter-free
- Scalability to many languages, large amounts of data, and large models is very important

Towards this, we developed a `text_duration_op` that operates on the `input generator`

We explore various forms of duration modeling

Intuition: we have 100M parameters in the shared encoder, any variations in the duration perhaps will be modeled by the shared encoder.
Repeating subword units (wordpiece/phoneme) by a fixed amount, similar to [Thomas, 2022]
Advantages: Very simple, worked-well in [Thomas, 2022]
Disadvantages: Crude, does not model the actual distribution of each subword unit
Text Duration Modeling: Random Repetition

- Randomly sample each subword unit between (1, 4)
- Advantages: Provides a distribution to subword units unlike fixed
- Disadvantages: Crude, does not model the actual distribution of each subword unit

\[ x_0 \ x_0 \ x_1 \ x_2 \ x_2 \ x_2 \]

Random

\[ x_0 \ x_1 \ x_2 \]
Text Duration Modeling: Subword Distribution

- Compute \((\mu, \Sigma)\) for each subword unit over the paired training data using forced alignment.
- In training, given \((\mu, \Sigma)\) for each subword unit, sample from Gaussian.
- Advantages: More exact compared to fixed or repetition.
- Disadvantages: Does not account for contextual effects.

\[
\begin{align*}
x_0 & \quad x_0 & \quad x_1 & \quad x_1 & \quad x_2 \\
\end{align*}
\]
Text Duration Modeling: Subword Distribution + Align

- For paired speech-text: Use forced alignment information to obtain duration of subwords
- For unpaired text: Given $(\mu, \Sigma)$ for each subword unit, sample from Gaussian
- Advantages: Most exact of all schemes, still parameter-free
- Disadvantages: Requires an alignment, though we still need for natcon/endpointing

Multidomain Text Data: Alignment

$\mathbf{x}_0 \ \mathbf{x}_0 \ \mathbf{x}_0 \ \mathbf{x}_1 \ \mathbf{x}_1 \ \mathbf{x}_2 \ \mathbf{x}_2 \ \mathbf{x}_2$

Alignment

LM Text Data: Distribution

$\mathbf{x}_0 \ \mathbf{x}_0 \ \mathbf{x}_1 \ \mathbf{x}_1 \ \mathbf{x}_2$

Gaussian
MWER Training

- Standard MWER Training is to optimize the expected number of word errors

\[ \mathcal{L}^{\text{MWER}}(y^*, x) = \mathbb{E}[\mathcal{W}(y, y^*)] \]

\[ \simeq \sum_{y_i} \left[ \frac{P(y_i|x)}{\sum_i P(y_i|x)} \right] \left[ W(y_i, y^*) - \frac{\sum_i W(y_i, y^*)}{N} \right] \]

- With JOIST, we can compute the MWER loss using paired audio-text data but also using the unpaired text representations

\[ \mathcal{L} = \lambda_1 \left[ \mathcal{L}_C^{\text{MWER}}(y_s, x_s) + \mathcal{L}_{NC}^{\text{MWER}}(y_s, x_s) \right] + \lambda_2 \left[ \mathcal{L}_C^{\text{MWER}}(y_t, x_t) + \mathcal{L}_{NC}^{\text{MWER}}(y_t, x_t) \right] + \alpha \mathcal{L}_{CE} \]

Data Sets

- **Training data:**
  - ~300 million multidomain audio-text pairs
  - LM text data:

- **Test data:**
  - AH_VS standard test search test set (S)
  - 5K *synthesized* long-tail utterances across 5 domains
    - Maps (M), News (N), Play (P), QSessions (Q), YouTube (Y)
Full Context Models, Joint Training

<table>
<thead>
<tr>
<th>Exp</th>
<th>Model</th>
<th>S</th>
<th>M</th>
<th>N</th>
<th>P</th>
<th>Q</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>B0</td>
<td>CascEnc,No text</td>
<td>4.8</td>
<td>11.9</td>
<td>8.2</td>
<td>36.1</td>
<td>19.3</td>
<td>22.6</td>
</tr>
<tr>
<td>B1</td>
<td>SLAM</td>
<td>75.9</td>
<td>87.0</td>
<td>99.4</td>
<td>87.1</td>
<td>84.7</td>
<td>91.2</td>
</tr>
<tr>
<td>E0</td>
<td>No Rep</td>
<td>4.8</td>
<td>11.9</td>
<td>8.5</td>
<td>35.8</td>
<td>19.5</td>
<td>22.6</td>
</tr>
<tr>
<td>E1</td>
<td>Fixed Rep</td>
<td>4.6</td>
<td>11.4</td>
<td>7.9</td>
<td>35.7</td>
<td>18.9</td>
<td>22.1</td>
</tr>
<tr>
<td>E2</td>
<td>Random Rep</td>
<td>4.7</td>
<td>11.8</td>
<td>8.2</td>
<td>35.9</td>
<td>22.2</td>
<td>22.2</td>
</tr>
<tr>
<td>E3</td>
<td>Sub-wrd dst</td>
<td>4.9</td>
<td>11.8</td>
<td>8.0</td>
<td>36.2</td>
<td>19.5</td>
<td>22.2</td>
</tr>
</tbody>
</table>

- SLAM (B1) has a large degradation due to speech-text concatenation.
- Duration modeling (E1-E3) helps over no-text baseline (B0) and no repetition (E0).
- Best system (E1) gives between a 2-5% relative improvement in WER.
- Goal: Highlight importance of duration modeling and joint training as being a simple way to optimize mixed input systems.
Similar to full-context,
- No-repetition (E4) is no better than no-text (B2)
- Duration modeling (E5-E7) helps over no-text baseline (B2)
- Parameter-free duration modeling on par with Maestro
- Best system gives between a 2-4% relative improvement in WER.
## MWER Training

<table>
<thead>
<tr>
<th>Exp</th>
<th>Model</th>
<th>S</th>
<th>M</th>
<th>N</th>
<th>P</th>
<th>Q</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>B4</td>
<td>CascEnc, pre-MWER</td>
<td>6.2</td>
<td>13.9</td>
<td>9.4</td>
<td>37.9</td>
<td>21.6</td>
<td>24.4</td>
</tr>
<tr>
<td>B6</td>
<td>B4 + MWER</td>
<td>5.8</td>
<td>13.5</td>
<td>9.1</td>
<td>37.5</td>
<td>20.8</td>
<td>24.0</td>
</tr>
<tr>
<td>E11</td>
<td>JOIST, pre-MWER</td>
<td>6.1</td>
<td>13.1</td>
<td>9.6</td>
<td>32.6</td>
<td>18.7</td>
<td>21.2</td>
</tr>
<tr>
<td>E14</td>
<td>E11 + MWER</td>
<td>5.8</td>
<td>12.7</td>
<td>9.4</td>
<td>32.0</td>
<td>18.3</td>
<td>20.7</td>
</tr>
</tbody>
</table>

- Compare MWER training of Cascaded Encoder using paired data to JOIST using paired and unpaired data.
- After MWER training, further gains are seen with JOIST (E14) compared to the cascaded encoder (B6), particularly on rare-word sets.
Conclusion

- Text injection is useful.
- How to match the two modality are important.
- In the future:
  - Unified framework for encoder and decoder.
  - Single, efficient model for both speech and text.