ESPnet: End-to-end speech processing toolkit

Shinji Watanabe
Center for Language and Speech Processing
Johns Hopkins University

Joint work with Takaaki Hori, Shigeki Karita, Tomoki Hayashi, Jiro Nishitoba, Yuya Unno, Nelson Enrique Yalta Soplin, Jahn Heymann, Matthew Wiesner, Nanxin Chen, Adithya Renduchintala, Tsubasa Ochiai
Lab instruction

- [https://hackmd.io/s/rJ6TDZPeQ](https://hackmd.io/s/rJ6TDZPeQ)
ESPnet
https://github.com/espnet/espnet

- Open source (Apache2.0) end-to-end ASR toolkit
  - Developed for the 2018 JSALT workshop “Multilingual End-to-end ASR for Incomplete Data”
- Actively developed by researchers all over the world (JHU, MERL, Nagoya Univ., NTT, Paderborn Univ., PFN, ...)
- Chainer or Pytorch backend
- Follows the Kaldi style
  - Data processing
  - Feature extraction/format
  - Recipes to provide a complete setup for speech recognition and other speech processing experiments
Software architecture

**NN libraries**
- Chainer
- PyTorch
- Warp_CTC

**Numerical libraries**
- Numpy
- Scipy

**ESPnet libraries (python)**
- **Chainer backend**
  - asr_chainer.py
  - lm_chainer.py
  - e2e_asr_attctc.py
  - asr_utils.py
  - lm_utils.py
  - etc.
- **Pytorch backend**
  - asr_pytorch.py
  - lm_pytorch.py
  - e2e_asr_attctc_th.py

**ESPnet Executables (python)**
- asr_train.py
- asr_recog.py
- lm_train.py

**ESPnet Recipes (bash)**
- csj/asr1/run.sh
- wsj/asr1/run.sh
- etc.

**ASR libraries**
- Kaldi, Sclite
Functionalities

• Kaldi style data preprocessing
  1) fairly compare the performance obtained by Kaldi hybrid systems
  2) make use of data preprocessing developed in the Kaldi recipe

• Attention-based encoder-decoder
  • Subsampled BLSTM and/or VGG-like encoder
  • location-based attention (+10 attentions)

• CTC
  • WarpCTC, label-synchronous decoding

• Hybrid CTC/attention
  • Multitask learning, joint decoding

• Use of language models
  • Combination of RNNLM and label-synchronous hybrid CTC/attention decoding
Backends

- Use Chainer and PyTorch

<table>
<thead>
<tr>
<th></th>
<th>Chainer</th>
<th>PyTorch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance</td>
<td>◎</td>
<td>○</td>
</tr>
<tr>
<td>Speed</td>
<td>○</td>
<td>◎</td>
</tr>
<tr>
<td>Multi-GPU</td>
<td>supported</td>
<td>supported</td>
</tr>
<tr>
<td>VGG-like encoder</td>
<td>supported</td>
<td>no support</td>
</tr>
<tr>
<td>RNNLM integration</td>
<td>supported</td>
<td>supported</td>
</tr>
<tr>
<td>#Attention types</td>
<td>3 (no attention, dot, location)</td>
<td>12 including variants of multihead</td>
</tr>
</tbody>
</table>
Lines of code, etc.

- Kaldi
  
  ```
  $ cat kaldi/src/*/*.{cc,cu,h} | wc -l
  ~330k
  ```

- ESPnet
  
  ```
  $ cat espnet/src/*/*.{sh,py} | wc -l
  ~6.9k
  ```

- Chainer/Pytorch as a main deep learning engine
- Use Kaldi feature extraction, and python-based reader/writer
Basic flow of recipes

- Very simple flow
  - No Gaussian construction
  - No FST
  - No alignments
  - No lattice outputs
- Easily ported from existing Kaldi recipes
Supported recipes (15 recipes)

- ami
- an4
- babel
- chime4
- chime5
- csj
- fisher_swbd
- hkust
- hub4_Spanish
- librispeech
- li10
- swbd
- tedlium
- voxforge
- wsj
Supported languages (25 languages)

Major English tasks (WSJ, Fisher+Switchboard, Librispeech)

Japanese (Corpus of Spontaneous Japanese)

Mandarin (HKUST CTS)

Babel 15 languages
- Cantonese, Assamese, Bengali, Pashto, Turkish, Georgian, Tagalog, Vietnamese, Haitian, Swahili, Lao, Tamil, Zulu, Kurmanji Kurdish, Tok Pisin

VoxForge 7 languages
- German, Spanish, French, Italian, Portuguese, Russian, Dutch
Performance

- **WSJ**

<table>
<thead>
<tr>
<th>Method</th>
<th>Metric</th>
<th>dev93</th>
<th>eval92</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESPnet with VGG2-BLSTM</td>
<td>CER</td>
<td>10.1</td>
<td>7.6</td>
</tr>
<tr>
<td>+ BLSTM layers (4 → 6)</td>
<td>CER</td>
<td>8.5</td>
<td>5.9</td>
</tr>
<tr>
<td>+ char-LSTM + LM</td>
<td>CER</td>
<td>8.3</td>
<td>5.2</td>
</tr>
<tr>
<td>+ joint decoding</td>
<td>CER</td>
<td>5.5</td>
<td>3.8</td>
</tr>
<tr>
<td>+ label smoothing</td>
<td>CER</td>
<td>5.3</td>
<td>3.6</td>
</tr>
<tr>
<td></td>
<td>WER</td>
<td>12.4</td>
<td>8.9</td>
</tr>
<tr>
<td>seq2seq + CNN (no LM) [33]</td>
<td>WER</td>
<td>N/A</td>
<td>10.5</td>
</tr>
<tr>
<td>seq2seq + FST word LM [35]</td>
<td>CER</td>
<td>N/A</td>
<td>3.9</td>
</tr>
<tr>
<td></td>
<td>WER</td>
<td>N/A</td>
<td>9.3</td>
</tr>
<tr>
<td>CTC + FST word LM [11]</td>
<td>WER</td>
<td>N/A</td>
<td>7.3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Wall Clock Time</th>
<th># GPUs</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESPnet (Chainer)</td>
<td>20 hours</td>
<td>1</td>
</tr>
<tr>
<td>ESPnet (PyTorch)</td>
<td>5 hours</td>
<td>1</td>
</tr>
<tr>
<td>seq2seq + CNN [33]</td>
<td>120 hours</td>
<td>10</td>
</tr>
</tbody>
</table>

- **CSJ**

<table>
<thead>
<tr>
<th>Method</th>
<th>eval1</th>
<th>eval2</th>
<th>eval3</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESPnet</td>
<td>8.7</td>
<td>6.2</td>
<td>6.9</td>
</tr>
<tr>
<td>ESPNet (5 GPUs)</td>
<td>8.5</td>
<td>6.1</td>
<td>6.8</td>
</tr>
<tr>
<td>HMM/DNN (Kaldi nnet1)</td>
<td>9.0</td>
<td>7.2</td>
<td>9.6</td>
</tr>
<tr>
<td>CTC-syllable [43]</td>
<td>9.4</td>
<td>7.3</td>
<td>7.5</td>
</tr>
</tbody>
</table>

- **HKUST**

<table>
<thead>
<tr>
<th>Method</th>
<th>eval</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESPnet</td>
<td>28.3</td>
</tr>
<tr>
<td>HMM/LSTM (Kaldi nnet3)</td>
<td>33.5</td>
</tr>
<tr>
<td>CTC with language model [11]</td>
<td>34.8</td>
</tr>
<tr>
<td>HMM/TDNN, LF MMI [27]</td>
<td>28.2</td>
</tr>
</tbody>
</table>
Summary

• Easy to develop, easy to perform experiments
  • Thanks to the simplification of end-to-end ASR and recent developments of deep learning toolkits

• Multilingual functions
  • Make use of end-to-end ASR benefits

• Good performance
  • Moderate ASR performance on English benchmarks
  • State-of-the-art ASR performance on ideogram languages (Japanese and Mandarin)
  • Fast training

• Future development plans
  • Stabilities, faster training/recognition, performance improvement
  • Multi-*** (multilingual, multispeaker, multichannel, multimodal, etc.)
Lab instruction

• [https://hackmd.io/s/rJ6TDZPeQ](https://hackmd.io/s/rJ6TDZPeQ)