Poster Presentations

• Examples:
  https://nips.cc/Conferences/2017/Schedule?type=Poster

• Use your favourite tool
  (LaTeX, LibreOffice Impress, Adobe Illustrator, Powerpoint, …)

• Format: A0 landscape

• LaTeX templates:
  https://www.latextemplates.com/cat/conference-posters

• Reimbursement for poster printing at uCreate
  • Printing is paid for using Charged Balance credit
  • Plain (non-glossy) A0 poster: £8.00
  • Please ask for receipts

• Short interactive presentation: 10-15 min
  • You explain the poster to other students!

• Non-presenting students write summaries for two other poster presentations: select 2 out of 5 (4 non-presenting) poster sessions
  • One summary per selected poster session
Example Poster

Analyzing Hidden Representations in End-to-End Automatic Speech Recognition Systems
Yonatan Belinkov, James Glass
MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA 02139, USA
{belinkov, glass}@mit.edu

Motivation
- Traditional Automatic Speech Recognition (ASR) systems are complex with many moving parts: acoustic model, language model, lexicon, etc.
- End-to-end ASR maps acoustics directly to text, jointly optimizing for the recognition task
- End-to-end models do not require explicit phonetic supervision (e.g. phonemes)
- Research questions:
  - Do end-to-end models implicitly learn phonetic representations (“g” in “bought”)?
  - Which components capture more phonetic information?
  - Do more complicated ASR models learn better representations for phonology?

Methodology and Data
- Methodology
  - Train ASR model on transcribed speech
  - Extract features from the pre-trained model on a supervised dataset with phonetic segmentation
  - Train a simple classifier on a frame classification task: predict phones using the extracted features
- Classifier
  - One hidden layer, dropout, ReLU, softmax
  - Adam optimizer, cross-entropy loss
- Data
  - ASR training: LibriSpeech, 1000 hours of read speech
  - Frame classifier: TIMIT, time segmentation of phones

Analysis
- Effect of blank symbols
  - With strides, better representations at blanks
  - Without strides, better representations at non-blanks

Results
- Main results
  - Conv1 improves the input representation, but conv2 degrades it
  - RNN layers initially improve, then drop
  - Higher layers capture more global information like dependencies between characters (e.g. “bought”)
  - Similar trends in different configurations (layers, phone classes, input futures)
- Model complexity
  - LSTM layer representations are better than RNN, but the respective conv layers are worse
  - Deeper model has better WER (12 vs 15) but worse representations for phonology

Conclusion
- End-to-end CTC models learn substantial phonetic information
- Phonetic information persists until mid-layers, but the top layers lose phonetic information
- Separability in vector space corresponds to representation quality

ASR Model
- DeepSpeech2 (Amodei et al. 2017):
  - Map spectrograms to characters (or blanks)
  - Stack of CNNs and RNNs

<table>
<thead>
<tr>
<th>Layer</th>
<th>Type</th>
<th>Input Size</th>
<th>Output Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>cnn1</td>
<td>161</td>
<td>41x11</td>
</tr>
<tr>
<td>2</td>
<td>cnn2</td>
<td>41x11</td>
<td>21x11</td>
</tr>
<tr>
<td>3</td>
<td>rnn1</td>
<td>1312</td>
<td>1760</td>
</tr>
<tr>
<td>4</td>
<td>rnn2</td>
<td>1760</td>
<td>1760</td>
</tr>
<tr>
<td>5</td>
<td>rnn3</td>
<td>1760</td>
<td>1760</td>
</tr>
<tr>
<td>6</td>
<td>rnn4</td>
<td>1760</td>
<td>1760</td>
</tr>
<tr>
<td>7</td>
<td>rnn5</td>
<td>1760</td>
<td>1760</td>
</tr>
<tr>
<td>8</td>
<td>rnn6</td>
<td>1760</td>
<td>1760</td>
</tr>
<tr>
<td>9</td>
<td>rnn7</td>
<td>1760</td>
<td>1760</td>
</tr>
<tr>
<td>10</td>
<td>fc</td>
<td>1760</td>
<td>29</td>
</tr>
</tbody>
</table>

CTC loss (Graves 2006)
- Map spectrograms x to characters \( \mathbf{y} \) by considering all possible alignments \( \pi \)

\[
p(\mathbf{y} | \mathbf{x}) = \sum_{\pi \in B^{-1}(\mathbf{y})} \prod_{t = 1}^{T} \phi_t(x_t | \pi_t)
\]

where \( \phi_t(x_t) \in \mathbb{R}^V \) = output at time \( t \)