Speech recognition

• Many exciting & valuable applications

Content captioning

Cars / Hands-free interfaces

Home devices

Mobile devices
Speech recognition

- Many exciting & valuable applications

“Speech Is 3x Faster than Typing for English and Mandarin Text Entry on Mobile Devices”

[Ruan et al., 2016]
Speech recognition

• Given speech audio, generate a transcript.

Important goal of AI: historically hard for machines, easy for people.
Traditional ASR pipeline

• Traditional systems break problem into several key components:

\[ W^* = \arg\max_W P(W|X) \]

\[ = \arg\max_W P(O|W)P(W) \]

Audio wave \( X \)

Feature representation \( O \)

Decoder

Acoustic Model

Language Model

Gales & Young, 2008
Jurafsky & Martin, 2000
Traditional ASR pipeline

• Usually represent words as a sequence of “phonemes”:

\[ w_1 = \text{“hello”} = [\text{HH AH L OW}] = [q_1 q_2 q_3 q_4] \]

• Phonemes are the perceptually distinct units of sound that distinguish words.
  – Quite approximate... but sorta standardized-ish.
  – Some labeled corpora available (e.g., TIMIT)

<table>
<thead>
<tr>
<th>Phone Label</th>
<th>Example</th>
<th>Phone Label</th>
<th>Example</th>
<th>Phone Label</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>iy</td>
<td>beet</td>
<td>22</td>
<td>ch</td>
<td>choke</td>
</tr>
<tr>
<td>2</td>
<td>ih</td>
<td>bit</td>
<td>23</td>
<td>b</td>
<td>bee</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>43</td>
<td>en</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>44</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>eng</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Washington</td>
</tr>
</tbody>
</table>
Traditional ASR pipeline

- Traditional systems usually model phoneme sequences instead of words. This necessitates a dictionary or other model to translate.

\[
W^* = \arg\max_W P(W|X) = \arg\max_W \sum_Q P(O|Q)P(Q|W)P(W)
\]
Traditional ASR pipeline

• Traditional pipeline is highly tweak-able, but also hard to get working well.
• Historically, each part of system has own set of challenges.
  – E.g., choosing feature representation.
Deep Learning in ASR

• Where to apply DL to make ASR better?
  – Good start: improve acoustic model $P(O|Q)$
Raw audio

• Simple 1D signal:

Typical sample rates for speech: 8KHz, 16KHz. Each sample typically 8-bit or 16-bit.

• 1D vector: $X = [x_1 x_2 \ldots]$
Spectrogram

- Take a small window (e.g., 20ms) of waveform.
  - Compute FFT and take magnitude. (i.e., power)
  - Describes frequency content in local window.

“Hello world”

20ms

\[ \log |\text{FFT}(X)|^2 \]

Power

Frequency

1 Frame
Spectrogram

• Concatenate frames from adjacent windows to form “spectrogram”.

$O_1 \quad \cdots \quad O_t \quad \cdots \quad O_T$
Acoustic Model

• Goal: create a neural network (DNN/RNN) from which we can extract transcription, $y$.
  – Train from labeled pairs $(x,y^*)$

$y = "Hello"$
Acoustic model

- Main issue: \( \text{length}(x) \neq \text{length}(y) \)
  - Don’t know how symbols in \( y \) map to frames of audio.
  - Traditionally, try to bootstrap alignment – painful!

```
“the quick brown”
\[ \text{Expand to phonemes} \]
\[ \text{th ah qw eh ke ba ra ow en} \]
\[ + \text{audio} \]
\[ \text{Bootstrapped Recognizer} \]
\[ \text{alignment} \]
```

```
th th th ah ah qw qw eh eh ke ke ke ba ba ba ra ra ow ow ow en en en
```

Retrain recognizer
DL for End-to-end Speech

- No perceptual features (MFCC). No feature transformation. No phonetic inventory. No transcription dictionary. No HMM.
- The output of the RNN are characters including space, apostrophe, (not CD phones)
- Connectionist Temporal Classification (No fixed alignment speech/character)
- Data augmentation. 5,000 hours (9600 speakers) + noise = 100,000 hours. Optimizations: data parallelism, model parallelism
- Good results in noisy conditions
Connectionist Temporal Classification (CTC)

[Graves et al., 2006]

• Basic idea:

1. RNN output neurons $c$ encode distribution over symbols. Note length(c) == length(x).

   For phoneme-based model: $c \in \{AA, AE, AX, \ldots, ER1, \text{blank}\}$

   For grapheme-based model: $c \in \{A, B, C, D, \ldots, Z, \text{blank, space}\}$

2. Define a mapping $\beta(c) \rightarrow y$.

3. Maximize likelihood of $y^*$ under this model.
Deep Speech - Recurrent Neural Network

Output alphabet, space, & blank

[Diagram showing the output alphabet, space, and blank symbols with corresponding visual representations]
Deep Speech - CTC

No alignment needed!

\[ P(\_\_T\_H\_\_\_E\_\_\_C\_\_\_A\_A\_\_\_T\_T\_\_) \]

\[ + \]

\[ \cdot \]

\[ + \]

\[ P(\_\_T\_\_H\_\_\_E\_\_\_E\_\_\_\_C\_\_\_A\_A\_\_\_T\_\_\_\_\_) \]

\[ \{ P(\text{THE-CAT-}) \} \]
Connectionist Temporal Classification (CTC)

1. RNN output neurons $c$ encode distribution over symbols. Note $\text{length}(c) = \text{length}(x)$.

For grapheme-based model: $c \in \{A, B, C, D, ..., Z, \text{blank}, \text{space}\}$

$$c_{1,17} = P(c_{17} = 'B' \mid x)$$
Connectionist Temporal Classification (CTC)

2. Define a mapping $\beta(c) \rightarrow y$.
   
   – Given a specific character sequence $c$, squeeze out duplicates + blanks to yield transcription:

   $$y = \beta(c) = \beta(\text{HHH\_E\_\_LL\_LO\_\_\_}) = \text{“HELLO”}$$
Connectionist Temporal Classification (CTC)

• Mapping implies a distribution over possible transcriptions $y$:

$$P(c|x) = \{ 0.1 \ \text{HHH}\_\_\_\_\_LL\_\_\_LO\_\_\_,
0.02 \ \text{HH}\_\_\_\_\_E\_\_\_\_\_\_\_\_\_\_\_LL\_\_\_\_LO\_\_\_,
0.01 \ \text{HHH}\_\_\_\_\_E\_\_\_\_\_L\_\_\_\_\_\_\_\_\_OH\_\_,
0.01 \ \text{HHH}\_\_\_\_\_E\_\_\_\_\_E\_\_\_\_\_\_\_\_\_\_\_LL\_\_\_\_\_LO\_\_\_,
\ldots \ \text{YY}\_\_\_\_\_E\_\_\_\_\_\_\_\_\_\_\_LL\_\_\_\_\_LO\_\_\_W\_\_,
\ldots \$$

$$P(y|x) = \sum_{c: \beta(c) = y} P(c|x)$$

$$P(“HELLO”) = 0.1 + 0.02 + 0.01 + \ldots$$
Connectionist Temporal Classification (CTC)

3. Update network parameters $\theta$ to maximize likelihood of correct label $y^*$:

$$\theta^* = \arg\max_\theta \sum_i \log P(y^*(i) | x(i))$$

$$= \arg\max_\theta \sum_i \log \sum_{c: \beta(c) = y^*(i)} P(c | x(i))$$

- [Graves et al., 2006] provides an efficient dynamic programming algorithm to compute the inner summation and its gradient.
Connectionist Temporal Classification (CTC)

• Use usual gradient descent methods to optimize. Tune entire network with backpropagation.
  – Given network outputs, many off-the-shelf packages to compute CTC loss (likelihood) from $c$ and $y^*$, and gradient w.r.t. $c$.
    • Warp CTC: github.com/baidu-research/warp-ctc
    • Stanford CTC: github.com/amaas/stanford-ctc
    • Tensorflow: tf.nn.ctc_loss

\[
L(\theta) = \log P(y^*(i) \mid x^{(i)}) = \text{CTC}(c^{(i)}, y^*(i))
\]
Connectionist Temporal Classification (CTC)

- The framewise network receives an error for misalignment
- The CTC network predicts the sequence of phonemes / characters (as spikes separated by ‘blanks’)
- No force alignment (initial model) required for training.
Training tricks

• Getting RNN to train well is tricky.

“SortaGrad”: order utterances by length during first epoch.

Batch normalization

See [“Curriculum Learning”, Bengio et al., ICML 2009]

See [Ioffe & Szegedy, 2015]
Decoding

- Network outputs $P(c|\mathbf{x})$. How do we find most likely transcription from $P(y|\mathbf{x})$?
- Simple (approximate) solution:

$$\beta\left(\operatorname{arg\ max}_c P(c|\mathbf{x})\right)$$

- Often terrible, but a useful diagnostic to "eyeball" models.
Example

- RNN to predict graphemes (26 characters + space + blank):
  - Spectrograms as input.
  - 1 layer of convolutional filters.
  - 3 layers of Gated Recurrent Units.
    - 1000 neurons per layer.
  - 1 full-connected layer to predict $c$.
  - Batch normalization
    [Ioffe & Szegedy, 2015]
- CTC loss function (Warp-CTC)
- Train with SGD+Nesterov momentum.

Typical model family:
Example

• Wall Street Journal:
  – [https://catalog.ldc.upenn.edu/ldc93s6a](https://catalog.ldc.upenn.edu/ldc93s6a)
  – Reading WSJ articles.

Free alternative: LibriSpeech
  – Read speech from public domain audiobooks.
Example

• What happens inside?
Network outputs ($c$) at Iteration 300
(Thresholding / contrast added for clarity.)

Max decoding: h
Network outputs ($c$) at Iteration 1500
(Thresholding / contrast added for clarity.)

Max decoding: bhe y uar just hin fro ton
Network outputs ($c$) at Iteration 2500
(Thresholding / contrast added for clarity.)

Max decoding: bhey yore j esstand fromghtte
Network outputs \((c)\) at Iteration 5500
(Thresholding / contrast added for clarity.)

Max decoding: they are just in front

Ground truth: “There just in front”
Max Decoding

• Examples:

Max Decoding
“put pore lottle thank and sr crits sinpt the atting to them
having been turned ef the wal al thes years con”

True Label
“the poor little things cried cynthia think of them
having been turned to the wall all these years”

Max Decoding
“That is true baddel gre”

True Label
“That is true badauderie”
Decoding

• Network outputs $P(c|x)$. How do we find most likely transcription from $P(y|x)$?

• No efficient solution in general. Resort to search!
  – See [Graves et al., 2006] for prefix decoding strategy.
Language models

• Even with better decoding, CTC model tends to make spelling + linguistic errors. E.g.:

<table>
<thead>
<tr>
<th>RNN output</th>
</tr>
</thead>
<tbody>
<tr>
<td>what is the weather like in bostin right now</td>
</tr>
<tr>
<td>prime miniter nerenr modi</td>
</tr>
<tr>
<td>arther n tickets for the game</td>
</tr>
</tbody>
</table>

From Hannun et al., 2014.

• $P(y|x)$ modeled directly from audio.
  – But not enough audio data to learn complicated spelling and grammatical structure.
  – Only supports small vocabulary.
  – For grapheme models: “Tchaikovsky” problem.
Language models

• Two solutions
  – Fuse acoustic model with language model: \( P(y) \)
  – Incorporate linguistic data:
    • Predict phonemes + pronunciation lexicon + LM.

• Possible to train language model from massive text corpora.
  – Learn spelling + grammar
  – Greatly expand vocabulary
  – Elevate likely cases (“Tchaikovsky concerto”) over unlikely cases (“Try cough ski concerto”).
Language models

• Standard approach: n-gram models
  – Simple n-gram models are common, well supported.
    • KenLM: kheafield.com/code/kenlm/
  – Train easily from huge corpora.
  – Quickly update to follow trends in traffic.
  – Fast lookups inside decoding algorithms.
Decoding with LMs

• Given a word-based LM of form $P(w_{t+1} | w_{1:t})$, Hannun et al., 2014 optimize:

$$\arg\max_w P(w|x) P(w)^{\alpha}[\text{length}(w)]^{\beta}$$

$P(w|x) = P(y|x)$ for characters that make up $w$.

$\alpha$ and $\beta$ are tunable parameters to govern weight of LM and a bonus/penalty for each word.
Decoding with LMs

• Basic strategy: beam search to maximize

\[ \arg \max_w P(w|x)P(w)^\alpha[\text{length}(w)]^\beta \]

Start with set of candidate transcript prefixes, \( A = \{ \} \).
For \( t = 1..T \):

For each candidate in \( A \), consider:
1. Add blank; don’t change prefix; update probability using AM.
2. Add space to prefix; update probability using LM.
3. Add a character to prefix; update probability using AM.
Add new candidates with updated probabilities to \( A_{\text{new}} \).

\( A := K \) most probable prefixes in \( A_{\text{new}} \).
Decoding with LMs: Examples

<table>
<thead>
<tr>
<th>RNN output</th>
<th>Decoded Transcription</th>
</tr>
</thead>
<tbody>
<tr>
<td>what is the weather like in bostin right now</td>
<td>what is the weather like in boston right now</td>
</tr>
<tr>
<td>prime miniter nerenr modi</td>
<td>prime minister narendra modi</td>
</tr>
<tr>
<td>arther n tickets for the game</td>
<td>are there any tickets for the game</td>
</tr>
</tbody>
</table>

From Hannun et al., 2014.
Rescoring

- Another place to plug in DL algorithms:
  Systems usually produce N-best list.
  Use fancier models to “rescore” this list.
Rescoring with Neural LM

- Example: train neural language model and rescore word transcriptions.
  - Cheap to evaluate $P(w_k|w_{k-1}, w_{k-2}, \ldots, w_1)$ NLM on many sentences.
  - In practice, often combine with N-gram trained from big corpora.

1. (-25.45) I’m a connoisseur looking for wine and porkchops.  -24.45
2. (-26.32) I’m a connoisseur looking for wine and port shops.  -23.45
3. ...
4. ...
5. ...
Data

• Transcribing speech data isn’t cheap, but not prohibitive:
  – Roughly 50¢ to $1 per minute.

• Typical speech benchmarks offer 100s to few 1000s of hours.
  – LibriSpeech (audiobooks)
  – LDC corpora (Fisher, Switchboard, WSJ) ($$)
  – VoxForge
## Types of speech data

- **Application matters**
  - We want to find data that matches our goals.

<table>
<thead>
<tr>
<th>Styles of speech</th>
<th>Issues</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read</td>
<td>Disfluency / stuttering</td>
<td>Dictation</td>
</tr>
<tr>
<td>Conversational</td>
<td>Noise</td>
<td>Meeting transcription</td>
</tr>
<tr>
<td>Spontaneous</td>
<td>Mic quality / #channels</td>
<td>Call centers</td>
</tr>
<tr>
<td>Command/control</td>
<td>Far field</td>
<td>Device control</td>
</tr>
<tr>
<td></td>
<td>Reverb / echo</td>
<td>Mobile texting</td>
</tr>
<tr>
<td></td>
<td>Lombard effect</td>
<td>Home / IoT / Cars</td>
</tr>
<tr>
<td></td>
<td>Speaker accents</td>
<td></td>
</tr>
</tbody>
</table>
Read speech

• Reading is inexpensive way to get more data. < $10/hour depending on source

• Disadvantages:
  – Misses inflection/conversational tone
  – Lombard effect
  – Speaker variety sometimes a limitation.
Augmentation

• Many forms of distortion that model should be robust to:
  – Reverb, noise, far field effects, echo, compression artifacts, changes in tempo

Raw audio ($$$$)

Sound processor (e.g., SoX toolkit)

http://sox.sourceforge.net/

Novel audio
Example: additive noise

DeepSpeech 2: 10K hours of raw audio -> 100K hours of novel training data
Data Argumentation

Data Augmentation

This approach needs bigger models and bigger datasets.

Synthesis by superposition: reverberations, echoes, a large number of short noise recordings from public video sources, jitter, Lombart Effect.

[Graph showing hours of data for different datasets and a significant increase in 'DeepSpeech' data]
Results: DeepSpeech 2

- Steady fall in error rates with new raw data.

![Graph showing WER-Clean and WER-Noisy over hours of training data]
Computation

• How big is 1 experiment?

At least:

(# connections) \cdot (# frames) \cdot (# utterances) \cdot (# epochs) \cdot 3 \cdot 2 \text{ FLOPs}

E.g.: for DS2 with 10K hours of data:

\begin{align*}
100e6 \cdot 100 \cdot 10e6 \cdot 20 \cdot 3 \cdot 2 &= 1.2e19 \text{ FLOPs} \\
\end{align*}

\sim 30 \text{ days with well-optimized code on Titan X.}
Computation

CPU
~100 GigaFLOPs

Titan X (2016) GPU
~11 TeraFLOPS
Computation

• Easy: use more GPUs with data parallelism.

• Current servers support ~8 Titans.
  – Will get you < 1 week training time.
Computation

- Try to keep similar-length utterances together.

(LibriSpeech clean data.)
Computation

- Try to keep similar-length utterances together.

**Bad minibatch:**

**Good minibatch:**
Latency

• Many acoustic model structures hard to serve in practice.
  – E.g., Bi-directional RNNs.

Must wait for all audio to arrive before computing.
Latency

• Fix: Move most of context to the top.

• Can easily compute/recompute top layers online.
## Hybrid (IBM) vs. DeepSpeech 1 (Baidu)

<table>
<thead>
<tr>
<th>Feature</th>
<th>IBM 2015</th>
<th>Baidu 2014</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Features</strong></td>
<td>VTL-PLP, MVN, LDA, STC, fMMLR, i-Vector</td>
<td>80 log filter banks</td>
</tr>
<tr>
<td><strong>Alignment</strong></td>
<td>GMM-HMM 300K Gaussians</td>
<td>-</td>
</tr>
<tr>
<td><strong>DNN</strong></td>
<td>DNN(5x2048) + CNN(128x9x9+5x2048) + +RNN 32000 outputs</td>
<td>4RNN (5 x 2304) 28 outputs</td>
</tr>
<tr>
<td><strong>DNN Training</strong></td>
<td>CE + MBR Discriminative Training (ST)</td>
<td>CTC</td>
</tr>
<tr>
<td><strong>HMM</strong></td>
<td>32K states (DNN outputs) pentaphone acoustic context</td>
<td>-</td>
</tr>
<tr>
<td><strong>Language Model</strong></td>
<td>37M 4-gram + model M (class based exponential model) + 2 NNLM</td>
<td>4-gram (Transcripts)</td>
</tr>
</tbody>
</table>
## DeepSpeech 1 vs. DeepSpeech 2 (Baidu)

<table>
<thead>
<tr>
<th></th>
<th>Deep Speech 1 (Baidu 2014)</th>
<th>DS2 (Baidu 2015)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Features</strong></td>
<td>80 log filter banks</td>
<td>?</td>
</tr>
<tr>
<td><strong>Alignment</strong></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>DNN</strong></td>
<td>4RNN (5 x 2304) 28 outputs</td>
<td>9-layer, 7RNN, BatchNorm, Conv. Layers. (Time/Freq)</td>
</tr>
<tr>
<td><strong>DNN Training</strong></td>
<td>CTC</td>
<td>CTC</td>
</tr>
<tr>
<td><strong>HMM</strong></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>Language Model</strong></td>
<td>4-gram</td>
<td>5-gram</td>
</tr>
</tbody>
</table>
DeepSpeech 1 vs. DeepSpeech 2 (Baidu)

<table>
<thead>
<tr>
<th>Accented Speech</th>
<th>DS1</th>
<th>DS2</th>
<th>Human</th>
</tr>
</thead>
<tbody>
<tr>
<td>VoxForge American-Canadian</td>
<td>15.01</td>
<td>7.55</td>
<td>4.85</td>
</tr>
<tr>
<td>VoxForge Commonwealth</td>
<td>28.46</td>
<td>13.56</td>
<td>8.15</td>
</tr>
<tr>
<td>VoxForge European</td>
<td>31.20</td>
<td>17.55</td>
<td>12.76</td>
</tr>
<tr>
<td>VoxForge Indian</td>
<td>45.35</td>
<td>22.44</td>
<td>22.15</td>
</tr>
</tbody>
</table>

**Table 14:** Comparing WER of the DS1 system to the DS2 system on accented speech.

<table>
<thead>
<tr>
<th>Noisy Speech</th>
<th>DS1</th>
<th>DS2</th>
<th>Human</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHiME eval clean</td>
<td>6.30</td>
<td>3.34</td>
<td>3.46</td>
</tr>
<tr>
<td>CHiME eval real</td>
<td>67.94</td>
<td>21.79</td>
<td>11.84</td>
</tr>
<tr>
<td>CHiME eval sim</td>
<td>80.27</td>
<td>45.05</td>
<td>31.33</td>
</tr>
</tbody>
</table>
Results

• Scaled up models in Mandarin:

https://svail.github.io/mandarin/
Difficulties in Mandarin Speech Recognition
1. Mandarin is a *tonal* language

妈嘛马骂吗
Difficulties in Mandarin Speech Recognition

2. Thousands of characters! > 80K
Difficulties in Mandarin Speech Recognition

3. The homophone problem is ubiquitous.
Figure 1. Architecture of the DS2 system used on English versus Mandarin. The only change is the larger output layer in Mandarin that accommodates the 6000 Mandarin characters, as opposed to the 29 we use in English.
Summary

• Deep Learning makes first steps to state-of-art speech system simpler than ever.

• Performance significantly driven by data & models.
  – Focus on scaling data + compute.
  – Try more models, make more progress!

• Mature enough for production.
  – DeepSpeech model is live in Mandarin & English.

Starter code: github.com/baidu-research/ba-dls-deepspeech
References

• Dahl, Yu, Deng, Acero, “Large Vocabulary Continuous Speech Recognition with Context-Dependent DBN-HMMs”. ICASSP, 2011
• Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., Senior, A., Tucker, P., Yang, K., Le, Q.V. and Ng, A.Y. “Large scale distributed deep networks.” NIPS 2012
• Hannun, Maas, Jurafsky, Ng. “First-Pass Large Vocabulary Continuous Speech Recognition using Bi-Directional Recurrent DNNs” ArXiv: 1408.2873
• Vassil Panayotov, Guoguo Chen, Daniel Povey and Sanjeev Khudanpur, "LibriSpeech: an ASR corpus based on public domain audio books.” ICASSP 2015