Language Independent End-to-End Architecture For Joint Language and Speech Recognition (2017)

Motivation / Goal

Recognize multiple languages at the same time

- Two tasks: identify language AND recognize speech (simultaneously)
- Use a single model for 10 languages (EN, JP, CH, DE, ES, FR, IT, NL, PT, RU)
- Find out if transfer learning between languages work
- End to end: Directly train sequence to sequence, no lexicon, phoneme pronounciation maps, or manual alignment

Problems

- How to input audio?

  → Spectral features of audio frames (e.g. in 20ms segments)

. . .

- How to output text?
  (a) word embeddings (word2vec) (would need fixed dictionary)
  (b) characters (one-hot)
    - Different char sets for languages (latin, cyrillic, CJK)
    - Just unify all character sets (5500 total)

. . .

- How to output language id?
  (a) separate one-hot output
  (b) as a special character: “`[EN]Hello`” or "
  [CH]你好

"

Model overview

Model overview

Model overview (from the paper)

Model overview

Model overview (from the paper)

Simple Model overview

1. Input: Basically a spectrogram as a 2D image
2. Encoder (CNN + LSTM)
3. Decoder
   (a) Soft Attention for each input frame to each output character
   (b) LSTM Layer
4. Output
   - N characters from union of all languages (one-hot / softmax)

Input

(Ab)use of image processing pipeline - input formatted like a RGB image

x=time, y=feature index

- first channel: spectral features

- second channel: delta spectral features
- third channel: deltadelta spectral features

Encoder - VGG Net Architecture

VGG Net for image classification

Encoder - VGG Net Architecture - First six layers

VGG Net - first 6 layers

Encoder - Bidirectional LSTM layer

320 cells for each direction $\rightarrow$ 640 outputs per time step ($\vec{h}_t$)

Bidirectional LSTM

http://colah.github.io/posts/2015-09-NN-Types-FP/

Decoder (Attention-based)

Input: $\vec{x}_1, \ldots, \vec{x}_t$

Output: $c_1, \ldots, c_l$

1. Encode whole sequence to $\vec{h}_1, \ldots, \vec{h}_t$ (via VGG+BLSTM)
2. Calculate soft attention weights a_{lt}, based on
   (a) $a_{(l-1)t}$ (attention on same input for previous output)
   (b) current encoded state $\vec{h}_t$
   (c) previous hidden decoder state $\vec{q}_{l-1}$

. . .

3. Sum encoded state with soft alignment: $\vec{r}_l = \sum_t a_{lt}\vec{h}_t$
4. Decoder = Softmax(FC(LSTM($\vec{r}_l, \vec{q}_{l-1}, c_{l-1}$)))

Problems with this simple model

- Pure temporal attention too flexible, allows nonsensical alignments
  - Intuition: In MT word order can change, in ASR it can not
- Languages must be implicitly modeled

Additions to the simple model

Problem 1: "Pure temporal attention too flexible"

Add a second, Parallel Decoder with CTC

1. Input (same as before)
2. Encoder (same as before)
3. Decoder

   softmax layer per time step (converts 640 outputs from BLSTM $\rightarrow$ N characters)
4. $\rightarrow$ One output character per input frame, using CTC Loss

CTC Crash Course

Problem: output sequence shorter than input sequence

- First, add blank character `"-"` to set. e.g. HELLO $\rightarrow \{H, E, L, O, -\}$

. . .

- Inference: Remove duplicates: HHHH-EEEEEEEE-LL-LLL----OOOOOO $\rightarrow$ H-E-L-L-O $\rightarrow$ HELLO

. . .

- Training: HELLO $\rightarrow$ H-E-L-L-O $\rightarrow$ all combinations of char duplications are ok

. . .

- Efficient computation using Viterbi / forward-backward algorithm
- Loss = - log of GT probability

→ Enforces monotonic alignment

Problem 2: "Languages must be implicitly modeled"

Add a RNN-LM

- Model distribution of character sequences in languages (ignores input speech)
- Trained seperately

Combine both decoders + RNN-LM

Hybrid CTC/attention-based end-to-end architecture (RNN-LM not shown)

Final loss function

$$\mathcal{L}_{\text{MTL}} = \lambda \log p_{\text{ctc}}(C|X) + (1-\lambda) \log p_{\text{att}}(C|X) + \gamma \log p_{\text{rnnlm}}(C)$$

$$\lambda = 0.5, \gamma = 0.1$$

Training / Inference

- AdaDelta optimization, 15 epochs

- Inference via beam search on attention output weighted by loss function

Results

Results

Character Error Rates (abbrev.)

- VGG-CNN improves it (by 7%)
- RNN-LM improves it (by 3%)
- Adding data in other languages improves it (by 9%)

Language Confusion Matrix

Language identification (LID) accuracies/error rates (%). The diagonal elements correspond to the LID accuracies while the offdiagonal elements correspond to the LID error rates

Potential problems / future work?

- Only fed with a single language utterance at a time
  - maybe we want to allow switching? (append utterances from different languages)

. . .

- Uniform random parameter initialization with $[-0.1, 0.1]$ seems statistically unsound? (use Xavier / Hu)

- Input feature convolution is weird
  - *[. . .] we used 40-dimensional filterbank features with 3-dimensional pitch features*
  - redundancy (delta, deltadelta)

. . .

- Unbalanced language sets (500h CH, 2.9h PR)
- Same latin characters are used for multiple languages, while others (RU, CH, JP) get their own character set
  - Try transliterating them to Latin?

Future Work (Opinion)

- Does not work online (without complete input utterance)
  - Bidirectional LSTM in encoder
    - Could try one directional, but Language ID would completely break

        * aggregate limited number of future frames (e.g. add 500ms latency between input and output)
    - Attention does not work in realtime
    - CTC should work online

Thank you for your *attention*

Full Results Table

Related Work

- *Multilingual Speech Recognition With A Single End-To-End Model* (Shubham Toshniwal, Google)
    - separate output for language id
    - only on 9 indian languages, hard to compare
- *Hybrid CTC/Attention Architecture for End-to-End Speech Recognition* (Watanabe et al. 2017)
    - Same as this paper except only one language and more detailed

...