About Sequence Classification for Sound Event Detection and end-to-end ASR

Thomas Pellegrini
Outline

Bird’s eye view on Recurrent Neural Networks

Case study: Weakly-labeled Sound Event Detection

Speech recognition
  Motivation for end-to-end systems
  From sounds to characters: CTC models
  Attention-based encoder-decoder models: Listen, Attend, Spell
  A few words on transfer learning
Examples of sequence data

- Speech recognition
- Music generation
- Sentiment classification: “There is nothing to like in this movie.”
- DNA sequence analysis: AGCCCTGTGAGGAATCTAG
- Machine translation: Voulez-vous chanter avec moi?
- Video activity recognition: Running
- Name entity recognition: Yesterday, Harry Potter met Hermione Granger.
Outline

Bird’s eye view on Recurrent Neural Networks

Case study: Weakly-labeled Sound Event Detection

Speech recognition
  Motivation for end-to-end systems
  From sounds to characters: CTC models
  Attention-based encoder-decoder models: Listen, Attend, Spell
  A few words on transfer learning
Generic RNNs

RNN:
- a hidden state \( \mathbf{h} \)
- an optional output \( \mathbf{y} \) which operates on a variable-length sequence \( \mathbf{x} = (x_1, \ldots, x_T) \)

At each time step \( t \), \( \mathbf{h}_t \) is updated by:

\[
\mathbf{h}_t = f(\mathbf{h}_{t-1}, x_t)
\]

where \( f \) is a non-linear function such as a sigmoid or an LSTM, or a GRU, etc.
Simple RNN (keras version)

$$h_t = x_t \cdot W$$
$$y_t = \tanh(h_t + y_{t-1} \cdot U)$$

Example: 10 cells, $x_t \in \mathbb{R}^d \rightarrow W \in \mathbb{R}^{d \times 10}$, $U \in \mathbb{R}^{10 \times 10}$, $h_t, y_t \in \mathbb{R}^{10}$

Unrolled RNN
Long Short Term Memory (LSTM)

[Hochreiter and Schmidhuber, 1997], [Gers, 2000]
Image from https://www.deeplearningbook.org/contents/rnn.html
Gated Recurrent Units (GRU)

- Two gating units instead of three in LSTM
- Shown to exhibit better performance on smaller datasets

[Cho et al, 2014], [Chung et al, 2014]
Generic RNNs

- An RNN can be trained to predict the next symbol in a sequence.
- In that case, the output at each timestep $t$ is the conditional distribution $p(x_t|x_{t-1}, \ldots, x_1)$.
- For language modeling for instance, we would use a softmax function:

$$p(x_t,j|x_{t-1}, \ldots, x_1) = \frac{\exp(w_j h_t)}{\sum_{i=1}^{K} \exp(w_i h_t)}$$

for all possible words $j = 1 \ldots K$, where $w_j$ are the rows of a weight matrix $W$.

- By combining these probabilities, we can compute the probability of the sequence $x$ using

$$p(x) = \prod_{t=1}^{T} p(x_t|x_{t-1}, \ldots, x_1)$$
Forward pass and backpropagation

RNN trained with the backpropagation through time algorithm (BPTT)
Summary of RNN types

One to one
\[ a^{<0>} \rightarrow x^{<1>} \rightarrow \hat{y}^{<1>} \]

One to many
\[ a^{<0>} \rightarrow x \rightarrow \hat{y}^{<1>} \rightarrow \hat{y}^{<2>} \rightarrow \cdots \rightarrow \hat{y}^{<T_y>} \]

Many to one
\[ a^{<0>} \rightarrow x^{<1>} \rightarrow x^{<2>} \rightarrow \cdots \rightarrow x^{<T_x>} \rightarrow \hat{y} \]

Many to many
\[ a^{<0>} \rightarrow x^{<1>} \rightarrow x^{<2>} \rightarrow \cdots \rightarrow x^{<T_x>} \rightarrow \hat{y}^{<1>} \rightarrow \hat{y}^{<2>} \rightarrow \cdots \rightarrow \hat{y}^{<T_y>} \]

Andrew Ng
Outline

Bird’s eye view on Recurrent Neural Networks

Case study: Weakly-labeled Sound Event Detection

Speech recognition
  Motivation for end-to-end systems
  From sounds to characters: CTC models
  Attention-based encoder-decoder models: Listen, Attend, Spell
  A few words on transfer learning
Case study: Sound Event Detection

- **Multi-label classification problem**
- **Weak labels**: audio tags, $y \in \{0, 1\}^C \rightarrow \text{many-to-one RNN}$
- **Strong labels**: frame-level tags, $Y \in \{0, 1\}^{T \times C} \rightarrow \text{many-to-many RNN}$
Generic approach

- **audio tagging**
  - CNN
  - bin CE loss
  - $y \in \{0, 1\}^c$

- **localization**
  - CRNN
  - MIL loss + penalty
  - scores $\in \mathbb{R}^{T \times C}$

- rescaling
  - smoothing
  - threshold

- 0.00 2.34 speech
- 3.19 4.52 dog
- ...

**Diagram notes:**
- "scores $\in \mathbb{R}^{T \times C}$" refers to the output of the localization CRNN.
- "rescaling" and "smoothing" are applied to the output scores before thresholding.
- The final output includes labels such as "speech" and "dog," indicating the classification results.
Weakly-labeled SED: problem statement

- We want to perform:
  - "audio tagging": infer weak labels, $\hat{y}$
  - "localization" or SED: infer strong labels, $\hat{Y}$

- Input: a sequence of T.F. representation vectors, $X = \{x_1, \ldots, x_T\}$

- Only weak labels are available for learning

- In a recording weakly labeled as *Dog*, some acoustic frames will comprise dog barking and others will not

Options:

1. False strong labeling and then refine the strong labels
2. Models with attention, e.g. with Gated Linear Units
3. Multiple Instance Learning setting (MIL)
Baseline model: Convolutional Recurrent Neural Network with "false strong labeling"
Loss function for classification: cross-entropy

"False strong labeling" (FSL): strong labels == weak label

\[
\text{loss}\(\{X^k, y_c^k\}\) = \text{binCE}(y_c^k, \hat{y}_{tc}^k)
\]

\[
\text{binCE}(y_c^k, \hat{y}_{tc}^k) = -y_c^k \log \hat{y}_{tc}^k - (1 - y_c^k) \log (1 - \hat{y}_{tc}^k)
\]

- FSL achieves poor results for short duration events
Option 2: attention mechanisms

Language Modeling with Gated Convolutional Networks

Yann N. Dauphin¹  Angela Fan¹  Michael Auli¹  David Grangier¹

- Pre-dominant approach to language modeling: recurrent neural networks
- Gated Linear Units (GLU): non-recurrent alternative with stacked convolutions, allows parallelization over sequential tokens
Gated Linear Units (GLU)

\[ h_l(X) = (X \ast W + b) \odot \sigma(X \ast V + c) \]

- \( X \in \mathbb{R}^{N \times m} \) input of layer \( l \)
- learnable: \( W, V \in \mathbb{R}^{k \times m \times n}, b, c \in \mathbb{R}^n \)
- Similar to LSTM, the \( \sigma(X \ast V + c) \) gates control the information pass
- The gates replace standard activation functions (such as ReLU)
- Linear activation to help gradients flow
- Input shift to prevent using information from future words
- GLUs wrapped in a residual block
- Output activation function: adaptive Softmax [Grave et al, 2016]
Gated CNN in SED

LARGE-SCALE WEAKLY SUPERVISED AUDIO CLASSIFICATION USING GATED CONVOLUTIONAL NEURAL NETWORK

Yong Xu*, Qiuqiang Kong*, Wenwu Wang, Mark D. Plumbley

Diagram of the proposed model, including gated CNN blocks and their connections.
Final audio tag prediction:

$$y_c = \frac{\sum_t^T z_c^{\text{cla}}(t) \odot z_c^{\text{att}}(t)}{\sum_t^T z_c^{\text{att}}(t)}$$
Option 3: Multiple Instance Learning

- MIL terminology
  - $X = \{x_1, \ldots, x_T\}$: a bag or a set
  - $x_i$: an instance
  - No dependency nor ordering among each other
  - A single binary label $Y$
  - Unknown individual labels: $y = \{y_1, \ldots, y_T\}$

MIL assumptions:

$$Y = \begin{cases} 0, & \text{iff } \sum_t y_t = 0, \\ 1, & \text{otherwise} \end{cases}$$

A MIL model must be **permutation invariant**

In a compact form:

$$Y = \max_t \{y_t\}$$
MIL loss for WL-SED

\[
\text{loss}(\{X, y_c\}) = \text{binCE}(y_c, \max_t \hat{y}_{tc})
\]  

(1)

where

- \( y_c \in \{0, 1\} \): ground truth label for class \( c \)
- \( \hat{y}_{tc} \in [0, 1]^T \): temporal predictions for class \( c \)
Example
MIL loss for WL-SED

- Issue in distinguishing some classes
Adding a similarity penalty

\[
\text{loss}(\{X, y_c\}) = \text{binCE}(y_c, \max_t \hat{y}_{tc}) \\
+ \alpha y_c \sum_{l \neq c} y_l \max(0, \cos(\hat{y}_l, \hat{y}_c))
\]
Adding a similarity penalty

Joint work with Léo Cances
Audio demo?
Outline

Bird’s eye view on Recurrent Neural Networks

Case study: Weakly-labeled Sound Event Detection

Speech recognition
- Motivation for end-to-end systems
- From sounds to characters: CTC models
- Attention-based encoder-decoder models: Listen, Attend, Spell
- A few words on transfer learning
Conventional ASR pipeline

→ A single model instead of all these modules??

End-to-end speech recognition

“A system which directly maps a sequence of input acoustic features into a sequence of graphemes or words”

- Since about 2014, first attempt: CTC-based models
- Alternative: attention models
Connectionist Temporal Classification (CTC)

- CTC proposed by [Graves et al, 2006]
- A way to train acoustic models without frame-level alignments
• Find most likely symbol sequence given inputs

\[ S_0 \ldots S_{K-1} = \arg\max_{S'_0 \ldots S'_{K-1}} \text{prob}(S'_0 \ldots S'_{K-1}|X_0 \ldots X_{N-1}) \]
Option 1: Simply select the most probable symbol at each time
- Option 1: Simply select the most probable symbol at each time
  - Merge adjacent repeated symbols, and place the actual emission of the symbol in the final instant
• Option 2: Impose external constraints on what sequences are allowed
  – *E.g.* only allow sequences corresponding to dictionary words
  – *E.g.* Sub-symbol units (like in HW1 – what were they?)
How do we train these models?

- Either just define Divergence as:
  \[ \text{DIV} = \text{Xent}(Y_2, B) + \text{Xent}(Y_4, IY) + \text{Xent}(Y_6, F) + \text{Xent}(Y_9, IY) \]
- Or..
Either just define Divergence as:

$$DIV = Xent(Y_2, B) + Xent(Y_4, IY) + Xent(Y_6, F) + Xent(Y_9, IY)$$

Or repeat the symbols over their duration

$$DIV = \sum_t Xent(Y_t, symbol_t) = -\sum_t \log Y(t, symbol_t)$$
Problem: No timing information provided

Only the sequence of output symbols is provided for the training data
— But no indication of which one occurs where

How do we compute the divergence?
— And how do we compute its gradient w.r.t. $Y_t$
Constraining the alignment: Try 1

- Block out all rows that do not include symbols from the target sequence
  - E.g. Block out rows that are not /B/ /IY/ or /F/

Slide from Bhiksha Raj
Blocking out unnecessary outputs

Compute the entire output (for all symbols)
Copy the output values for the target symbols into the secondary reduced structure
Constraining the alignment: Try 1

- Only decode on reduced grid
  - We are now assured that only the appropriate symbols will be hypothesized

- Problem: This still doesn’t assure that the decode sequence correctly expands the target symbol sequence
  - E.g. the above decode is not an expansion of /B///IY///F///IY/

- Still needs additional constraints
Try 2: Explicitly arrange the constructed table

<table>
<thead>
<tr>
<th>/B/</th>
<th>$y_0^B$</th>
<th>$y_1^B$</th>
<th>$y_2^B$</th>
<th>$y_3^B$</th>
<th>$y_4^B$</th>
<th>$y_5^B$</th>
<th>$y_6^B$</th>
<th>$y_7^B$</th>
<th>$y_8^B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>/IY/</td>
<td>$y_0^{IY}$</td>
<td>$y_1^{IY}$</td>
<td>$y_2^{IY}$</td>
<td>$y_3^{IY}$</td>
<td>$y_4^{IY}$</td>
<td>$y_5^{IY}$</td>
<td>$y_6^{IY}$</td>
<td>$y_7^{IY}$</td>
<td>$y_8^{IY}$</td>
</tr>
<tr>
<td>/F/</td>
<td>$y_0^F$</td>
<td>$y_1^F$</td>
<td>$y_2^F$</td>
<td>$y_3^F$</td>
<td>$y_4^F$</td>
<td>$y_5^F$</td>
<td>$y_6^F$</td>
<td>$y_7^F$</td>
<td>$y_8^F$</td>
</tr>
<tr>
<td>/IY/</td>
<td>$y_0^{IY}$</td>
<td>$y_1^{IY}$</td>
<td>$y_2^{IY}$</td>
<td>$y_3^{IY}$</td>
<td>$y_4^{IY}$</td>
<td>$y_5^{IY}$</td>
<td>$y_6^{IY}$</td>
<td>$y_7^{IY}$</td>
<td>$y_8^{IY}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>/AH/</th>
<th>$y_0^{AH}$</th>
<th>$y_1^{AH}$</th>
<th>$y_2^{AH}$</th>
<th>$y_3^{AH}$</th>
<th>$y_4^{AH}$</th>
<th>$y_5^{AH}$</th>
<th>$y_6^{AH}$</th>
<th>$y_7^{AH}$</th>
<th>$y_8^{AH}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>/B/</td>
<td>$y_0^B$</td>
<td>$y_1^B$</td>
<td>$y_2^B$</td>
<td>$y_3^B$</td>
<td>$y_4^B$</td>
<td>$y_5^B$</td>
<td>$y_6^B$</td>
<td>$y_7^B$</td>
<td>$y_8^B$</td>
</tr>
<tr>
<td>/D/</td>
<td>$y_0^D$</td>
<td>$y_1^D$</td>
<td>$y_2^D$</td>
<td>$y_3^D$</td>
<td>$y_4^D$</td>
<td>$y_5^D$</td>
<td>$y_6^D$</td>
<td>$y_7^D$</td>
<td>$y_8^D$</td>
</tr>
<tr>
<td>/EH/</td>
<td>$y_0^{EH}$</td>
<td>$y_1^{EH}$</td>
<td>$y_2^{EH}$</td>
<td>$y_3^{EH}$</td>
<td>$y_4^{EH}$</td>
<td>$y_5^{EH}$</td>
<td>$y_6^{EH}$</td>
<td>$y_7^{EH}$</td>
<td>$y_8^{EH}$</td>
</tr>
<tr>
<td>/IY/</td>
<td>$y_0^{IY}$</td>
<td>$y_1^{IY}$</td>
<td>$y_2^{IY}$</td>
<td>$y_3^{IY}$</td>
<td>$y_4^{IY}$</td>
<td>$y_5^{IY}$</td>
<td>$y_6^{IY}$</td>
<td>$y_7^{IY}$</td>
<td>$y_8^{IY}$</td>
</tr>
<tr>
<td>/F/</td>
<td>$y_0^F$</td>
<td>$y_1^F$</td>
<td>$y_2^F$</td>
<td>$y_3^F$</td>
<td>$y_4^F$</td>
<td>$y_5^F$</td>
<td>$y_6^F$</td>
<td>$y_7^F$</td>
<td>$y_8^F$</td>
</tr>
<tr>
<td>/G/</td>
<td>$y_0^G$</td>
<td>$y_1^G$</td>
<td>$y_2^G$</td>
<td>$y_3^G$</td>
<td>$y_4^G$</td>
<td>$y_5^G$</td>
<td>$y_6^G$</td>
<td>$y_7^G$</td>
<td>$y_8^G$</td>
</tr>
</tbody>
</table>

Arrange the constructed table so that from top to bottom it has the exact sequence of symbols required
Explicitly constrain alignment

- Constrain that the first symbol in the decode must be the top left block.
- The last symbol must be the bottom right.
- The rest of the symbols must follow a sequence that monotonically travels down from top left to bottom right. 
  - I.e. never goes up.
- This guarantees that the sequence is an expansion of the target sequence.
  - /B/ /IY/ /F/ /IY/ in this case.
Explicitly constrain alignment

- Compose a graph such that every path in the graph from source to sink represents a valid alignment
  - Which maps on to the target symbol sequence (/B//AH//T/)
- Edge scores are 1
- Node scores are the probabilities assigned to the symbols by the neural network
- The “score” of a path is the product of the probabilities of all nodes along the path
- Find the most probable path from source to sink using any dynamic programming algorithm
  - E.g. The Viterbi algorithm
Iterative update

• Option 1:
  – Determine alignments for every training instance
  – Train model (using SGD or your favorite approach) on the entire training set
  – Iterate

• Option 2:
  – During SGD, for each training instance, find the alignment during the forward pass
  – Use in backward pass
Iterative update: Problem

• Approach heavily dependent on initial alignment

• Prone to poor local optima

• Alternate solution: Do not commit to an alignment during any pass.
Averaging over all alignments

- Instead of only selecting the most likely alignment, use the statistical expectation over all possible alignments

\[ DIV = E \left[ - \sum_t \log Y(t, s_t) \right] \]

- Use the entire distribution of alignments
- This will mitigate the issue of suboptimal selection of alignment
A posteriori probabilities of symbols

\[ P(s_t = S_r|S,X) \propto P(s_t = S_r, S|X) \]

- \( P(s_t = S_r, S|X) \) is the total probability of all valid paths in the graph for target sequence \( S \) that go through the symbol \( S_r \) (the \( r^{th} \) symbol in the sequence \( S_1 \ldots S_K \)) at time \( t \)

- We will compute this using the “forward-backward” algorithm
A key decoding problem

• Consider a problem where the output symbols are characters
• We have a decode: R R R O O O O D
• Is this the symbol sequence ROD or ROOD?

Slide from Bhiksha Raj
A key decoding problem

- Consider a problem where the output symbols are characters
- We have a decode: R R R O O O O O D
- Is this the symbol sequence ROD or ROOD?

Note: This problem does not always occur, e.g. when symbols have sub symbols
- E.g. If O is produced as O1 and O2
  - A single O would be of the form O1 O1 .. O2 → O
  - Multiple Os would have the decode O1 .. O2.. O1..O2.. → OO
A key decoding problem

- We have a decode: R R R O O O O O D
- Is this the symbol sequence ROD or ROOD?

- Solution: Introduce an explicit extra symbol which serves to separate discrete versions of a symbol
  - A “blank” (represented by “-”)
  - RRR---OO---DDD = ROD
  - RR-R---OO---D-DD = RRODD
  - R-R-R---O-O--DD-DDDD-D = RRROODDD
    - The next symbol at the end of a sequence of blanks is always a new character
    - When a symbol repeats, there must be at least one blank between the repetitions

- The symbol set recognized by the network must now include the extra blank symbol
  - Which too must be trained
The modified forward output

- Note the extra “blank” at the output
The modified forward output

- Note the extra “blank” at the output
The modified forward output

- Note the extra “blank” at the output

Slide from Bhiksha Raj
The modified forward output

- Note the extra "blank" at the output

Slide from Bhiksha Raj
The modified forward output

- Note the extra “blank” at the output

/B/ /IY/ /F/ /F/ /IY/
Composing the graph for training

- /B/
  - /IY/
    - /F/

- With blanks
- Note: a row of blanks between any two symbols
- Also blanks at the very beginning and the very end
Composing the graph for training

- Add edges such that all paths from initial node(s) to final node(s) unambiguously represent the target symbol sequence.
Composing the graph for training

- The first and last column are allowed to also end at initial and final blanks

Slide from Bhiksha Raj
Composing the graph for training

- The first and last column are allowed to also end at initial and final blanks
- Skips are permitted across a blank, but only if the symbols on either side are different
  - Because a blank is *mandatory between repetitions of a symbol* but *not required between distinct symbols*
\[ p(Y \mid X) = \sum_{A \in A_{X,Y}} \prod_{t=1}^{T} p_t(a_t \mid X) \]

The CTC conditional probability marginalizes over the set of valid alignments computing the probability for a single alignment step-by-step.
Labels at each time index are conditionally independent (like HMMs)

Final learning objective: maximizing probabilities of true labels:

\[
CTC(X) = - \log P(Y^*|X)
\]
How to decode with a CTC model?

- Greedy search ("max decoding"): sub-optimal
- Prefix beam search (PBS) proposed in [Graves, 2006] uses the forward-backward algo (also used to compute the gradients during training)

---

**Table 1. Label Error Rate (LER) on TIMIT.** CTC and hybrid results are means over 5 runs, ± standard error. All differences were significant \((p < 0.01)\), except between weighted error BLSTM/HMM and CTC (best path).

<table>
<thead>
<tr>
<th>System</th>
<th>LER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context-independent HMM</td>
<td>38.85 %</td>
</tr>
<tr>
<td>Context-dependent HMM</td>
<td>35.21 %</td>
</tr>
<tr>
<td>BLSTM/HMM</td>
<td>33.84 ± 0.06 %</td>
</tr>
<tr>
<td>Weighted error BLSTM/HMM</td>
<td>31.57 ± 0.06 %</td>
</tr>
<tr>
<td>CTC (best path)</td>
<td>31.47 ± 0.21 %</td>
</tr>
<tr>
<td>CTC (prefix search)</td>
<td>30.51 ± 0.19 %</td>
</tr>
</tbody>
</table>
Greedy search ("max decoding"): sub-optimal

Prefix beam search (PBS) proposed in [Graves, 2006] uses the forward-backward algo (also used to compute the gradients during training)

- Greedy search ("max decoding"): sub-optimal
- Prefix beam search (PBS) proposed in [Graves, 2006] uses the forward-backward algo (also used to compute the gradients during training)
- PBS with char/word LM incorporated [Hannun et al, 2014], [Maas et al, 2015]
- PBS + WFST in Eesen open-source toolkit (Kaldi based) [Miao, 2015]

Table 3. Performance of the character-based Eesen system using different vocabularies and language models, and its comparison with results presented in previous work.

<table>
<thead>
<tr>
<th>System</th>
<th>Vocabulary</th>
<th>Language Model</th>
<th>WER%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eesen</td>
<td>Original</td>
<td>Standard</td>
<td>9.07</td>
</tr>
<tr>
<td>Eesen</td>
<td>Expanded</td>
<td>Re-trained</td>
<td>7.34</td>
</tr>
<tr>
<td>Graves et al. [6]</td>
<td>Expanded</td>
<td>Re-trained</td>
<td>8.7</td>
</tr>
<tr>
<td>Hannun et al. [8]</td>
<td>Original</td>
<td>Unknown</td>
<td>14.1</td>
</tr>
</tbody>
</table>
CTC produces sparse and spiky activations

→ not a good tool for speech analysts?
Conclusions on CTC

- Alignment-agnostic end-to-end ASR approach
- CTC assumption: for efficiency, network outputs at different frames are conditionally independent
- CTC models need an external LM, greedy decoding does not perform well
- Requires several tens of hours of data to work well
- Open question: is CTC adapted to small scale speech tasks and to speech analysis studies?
RNN encoder-decoder for Machine Translation (Cho, 2014)

Recall the generic RNN update eq: \( h_t = f(h_{t-1}, x_t) \)

With the RNN encoder-decoder, it becomes: \( h_t = f(h_{t-1}, y_{t-1}, c) \)

\( c \): a summary of the input sequence
Attention models

- Issue with Cho’s 2014 model: long sentences
- Where the model should focus its attention given a sequence of inputs?
- Badhanau’s proposal: an encoder-decoder that learns to align and translate jointly

[Badhanau et al, 2014]

Figure 1: The graphical illustration of the proposed model trying to generate the $t$-th target word $y_t$ given a source sentence $(x_1, x_2, \ldots, x_T)$. 
Attention models

- $s_t = f(s_{t-1}, y_{t-1}, c_t)$

- The context vector $c_t$ is computed at each time step using:
  $$c_t = \sum_{j=1}^{T_x} \alpha_{tj} h_j$$

- $\alpha_{tj}$: amount of attention $y_t$ should pay to $h_j$

- How to compute $\alpha_{tj}$?

Figure 1: The graphical illustration of the proposed model trying to generate the $t$-th target word $y_t$ given a source sentence $(x_1, x_2, \ldots, x_T)$. 
- $\alpha_{tj}$: amount of attention $y_t$ should pay to $h_j$

- $c_t = \sum_{j=1}^{T_x} \alpha_{tj} h_j$

- How to compute $\alpha_{tj}$?
  - $\alpha_{tj} = \frac{\exp(e_{tj})}{\sum_{k=1}^{T_x} \exp(e_{tk})}$
  - $e_{tj} = a(s_{t-1}, h_j)$
  - $a$: alignment model scoring how well the inputs around position $j$ and the output at position $i$ match

- In practice, what is $a$?

---

Figure 1: The graphical illustration of the proposed model trying to generate the $t$-th target word $y_t$ given a source sentence $(x_1, x_2, \ldots, x_T)$. 

\[ e_{tj} = a(s_{t-1}, h_j) \]

- **a**: alignment model scoring how well the inputs around position \( j \) and the output at position \( i \) match.

- **In practice, what is a?**

- **a** is a small neural network.
Attention models

Image from https://ai.googleblog.com/2017/12/improving-end-to-end-models-for-speech.html
Listen, Attend and Spell (LAS), [Chan et al, 2015]

**Listen, Attend and Spell**

*William Chan*
Carnegie Mellon University
williamchan@cmu.edu

*Navdeep Jaitly, Quoc V. Le, Oriol Vinyals*
Google Brain
{ndjaitly, qvl, vinyals}@google.com

\[
h = \text{Listen}(x)
\]
\[
P(y|x) = \text{AttendAndSpell}(h, y)
\]

LAS is a character-level RNN encoder-decoder with attention
Listen, Attend and Spell (LAS, 2015)

- **Listen:**
  \[ h = \text{BLSTM}(x) \]

- **Attend:**
  \[ c_i = \text{AttentionContext}(s_i, h) \]

- **Spell:**
  \[ s_i = \text{RNN}(s_{i-1}, y_{i-1}, c_{i-1}) \]
Decoding and results

<table>
<thead>
<tr>
<th>Model</th>
<th>Clean WER</th>
<th>Noisy WER</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLDNN-HMM [20]</td>
<td>8.0</td>
<td>8.9</td>
</tr>
<tr>
<td>LAS</td>
<td>16.2</td>
<td>19.0</td>
</tr>
<tr>
<td>LAS + LM Rescoring</td>
<td>12.6</td>
<td>14.7</td>
</tr>
<tr>
<td>LAS + Sampling</td>
<td>14.1</td>
<td>16.5</td>
</tr>
<tr>
<td>LAS + Sampling + LM Rescoring</td>
<td>10.3</td>
<td>12.0</td>
</tr>
</tbody>
</table>

- Training set: Google voice search task, 2000 h, test set: 10 h
- Decoding: character-level beam search
- No need of a lexicon to constraint the search space to valid words, LAS spells real words
How much would a woodchuck chuck chuck
And the winner is...

STATE-OF-THE-ART SPEECH RECOGNITION WITH SEQUENCE-TO-SEQUENCE MODELS

Chung-Cheng Chiu, Tara N. Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J. Weiss, Kanishka Rao, Ekaterina Gonina, Navdeep Jaitly, Bo Li, Jan Chorowski, Michiel Bacchiani

Google, USA

<table>
<thead>
<tr>
<th>Exp-ID</th>
<th>Model</th>
<th>VS/D</th>
<th>1st pass Model Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>E8</td>
<td>Proposed LFR system</td>
<td>5.6/4.1</td>
<td>0.4 GB</td>
</tr>
<tr>
<td>E9</td>
<td>Conventional LFR system</td>
<td>6.7/5.0</td>
<td>0.1 GB (AM) + 2.2 GB (PM) + 4.9 GB (LM) = 7.2GB</td>
</tr>
</tbody>
</table>

*Table 5: Resulting WER on voice search (VS)/dictation (D). The improved LAS outperforms the conventional LFR system while being more compact. Both models use second-pass rescoring.*

Training set: 12,500 hours, 15M English sentences
Conclusions on end-to-end ASR

- Sequence-to-sequence (seq2seq) ASR is the new SOTA
- More precisely attention-based encoder-decoder architectures such as LAS are the new SOTA
- A single NN subsumes the acoustic, pronunciation and language model components
- requires massive data and compute resources
- → of any interest in our low-resource case?
A few words on transfer learning

Multilingual hybrid CTC-attention model as seed model

Performance drops when finetuning only the decoder

Gains by finetuning both the encoder and decoder