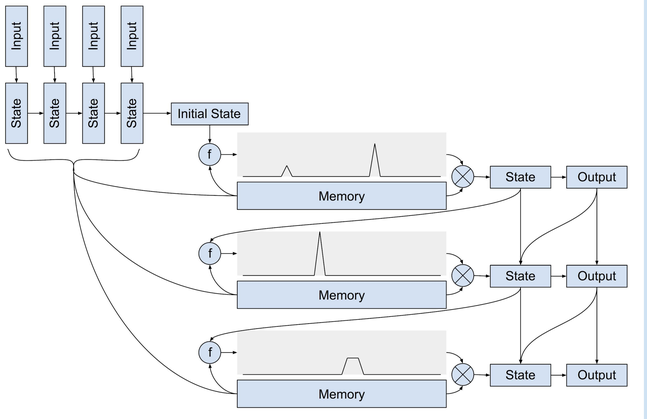
Monotonic Chunkwise Attention

Chung-Cheng Chiu* and Colin Raffel* (equal contribution)

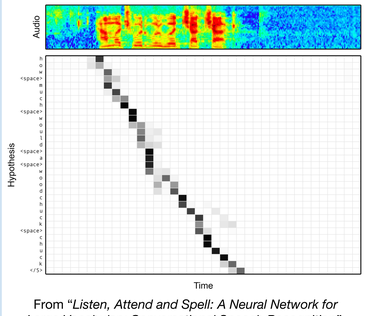
Abstract

Sequence-to-sequence models with soft attention have been successfully applied to a wide variety of problems, but their decoding process incurs a quadratic time and space cost and is inapplicable to real-time sequence transduction. To address these issues, we propose Monotonic Chunkwise Attention (MoChA), which adaptively splits the input sequence into small chunks over which soft attention is computed. We show that models utilizing MoChA can be trained efficiently with standard backpropagation while allowing online and linear-time decoding at test time. When applied to online speech recognition, we obtain state-of-the-art results and match the performance of a model using an offline soft attention mechanism. In document summarization experiments where we do not expect monotonic alignments, we show significantly improved performance compared to a baseline monotonic attention-based model.

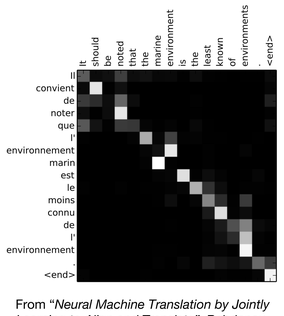
Sequence-to-sequence framework



Attention is often roughly monotonic

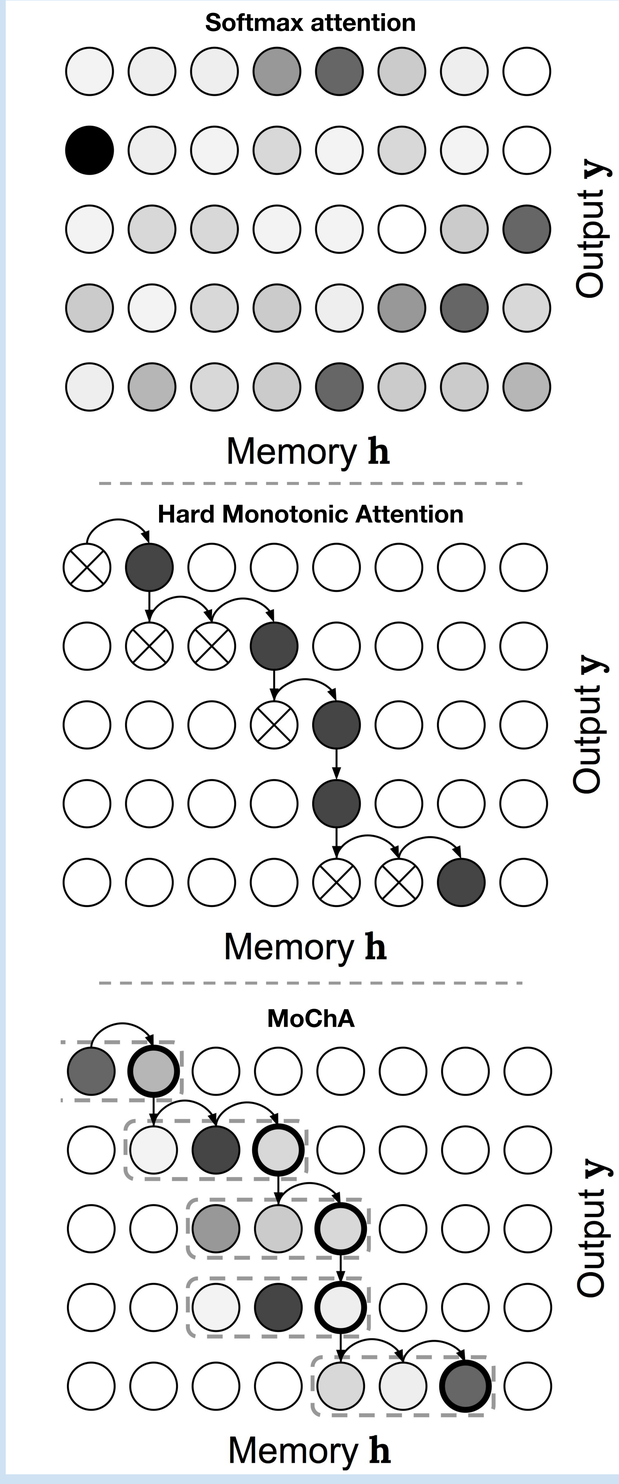


From "*Listen, Attend and Spell: A Neural Network for Large Vocabulary Conversational Speech Recognition*", Chan, Jaitly, Le, Vinyals.



From "*Neural Machine Translation by Jointly Learning to Align and Translate*", Bahdanau, Cho, Bengio.

Attention mechanisms



Decoding with MoChA

Training

$$e_{i,j} = \text{MonotonicEnergy}(s_{i-1}, h_j)$$
$$\epsilon \sim \mathcal{N}(0,1)$$
$$p_{i,j} = \sigma(e_{i,j} + \epsilon)$$
$$\alpha_{i,:} = p_{i,:} \,\texttt{cumprod}(1 - p_{i,:}) \,\texttt{cumsum}\left(\frac{\alpha_{i-1,:}}{\texttt{cumprod}(1-p_{i,:})}\right)$$
$$u_{i,j} = \text{ChunkEnergy}(s_{i-1}, h_j)$$
$$\beta_{i,:} = \exp(u_{i,:}) \,\text{MovingSum}\left(\frac{\alpha_{i,:}}{\text{MovingSum}(\exp(u_{i,:}), w, 1)}, 1, w\right)$$

Test-time

Input: memory $\mathbf{h}$ of length T, chunk size w
State: $s_0 = \vec{0}, t_0 = 1, i = 1, y_0 = \text{StartOfSequence}$
while $y_{i-1} \neq \text{EndOfSequence}$ **do** *// Produce output tokens until end-of-sequence token is produced*
  for $j = t_{i-1}$ **to** T **do** *// Start inspecting memory entries h_j left-to-right from where we left off*
    $e_{i,j} = \text{MonotonicEnergy}(s_{i-1}, h_j)$ *// Compute attention energy for h_j*
    $p_{i,j} = \sigma(e_{i,j})$ *// Compute probability of choosing h_j*
    if $p_{i,j} \geq 0.5$ **then** *// If $p_{i,j}$ is larger than 0.5, we stop scanning the memory*
      $v = j - w + 1$ *// Set chunk start location*
      for $k = v$ **to** j **do** *// Compute chunkwise* softmax *energies over a size-w chunk before j*
        $u_{i,k} = \text{ChunkEnergy}(s_{i-1}, h_k)$
      end for
      $c_i = \sum_{k=v}^{j} \frac{\exp(u_{i,k})}{\sum_{l=v}^{j} \exp(u_{i,l})} h_k$ *// Compute* softmax*-weighted average over the chunk*
      $t_i = j$ *// Remember where we left off for the next output timestep*
      break *// Stop scanning the memory*
    end if
  end for
  if $p_{i,j} < 0.5, \forall j \in \{t_{i-1}, t_{i-1}+1, \ldots, T\}$ **then**
    $c_i = \vec{0}$ *// If we scanned the entire memory without stopping, set c_i to a vector of zeros*
  end if
  $s_i = \text{DecoderRNN}(s_{i-1}, y_{i-1}, c_i)$ *// Update output RNN state based on the new context vector*
  $y_i = \text{Output}(s_i, c_i)$ *// Output a new symbol using the* softmax *output layer*
  $i = i + 1$
end while

Results

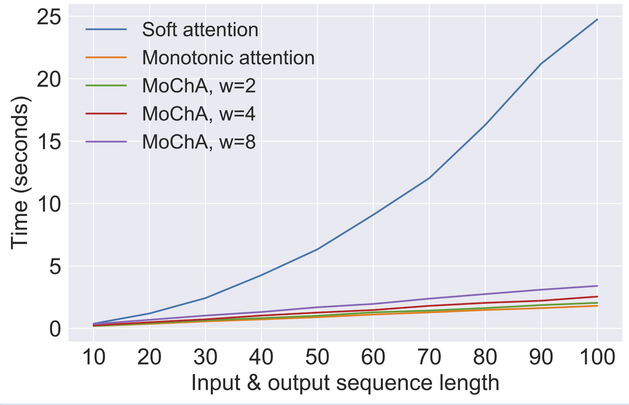
Prior Result	WER
(Raffel et al., 2017) (CTC baseline)	33.4%
(Luo et al., 2016) (Reinforcement Learning)	27.0%
(Wang et al., 2016) (CTC)	22.7%
(Raffel et al., 2017) (Monotonic Attention)	17.4%

Attention Mechanism	Best WER	Average WER
Soft Attention (offline)	14.2%	$14.6 \pm 0.3\%$
MoChA, $w = 2$	13.9%	$15.0 \pm 0.6\%$

Mechanism	R-1	R-2
Soft Attention (offline)	39.11	15.76
Hard Monotonic Attention	31.14	11.16
MoChA, $w = 8$	35.46	13.55

(left) Word Error rate for online speech recognition on WSJ. (above) ROUGE-1 and 2 for document summarization on CNN/Daily Mail.

Synthetic speed benchmark



Speech Attention Plots