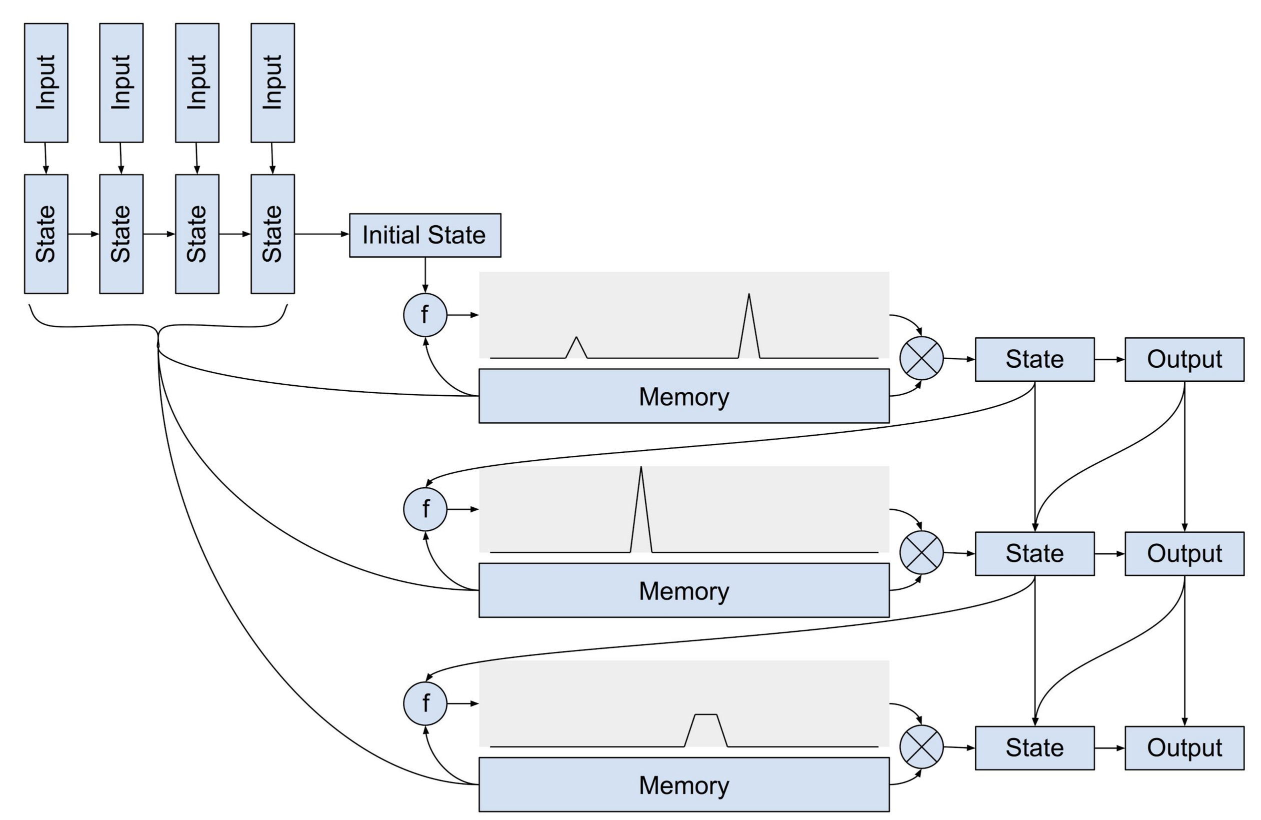
Online and Linear-Time Attention by Enforcing Monotonic Alignments

Colin Raffel, Minh-Thang Luong, Peter J. Liu, Ron J. Weiss, Douglas Eck

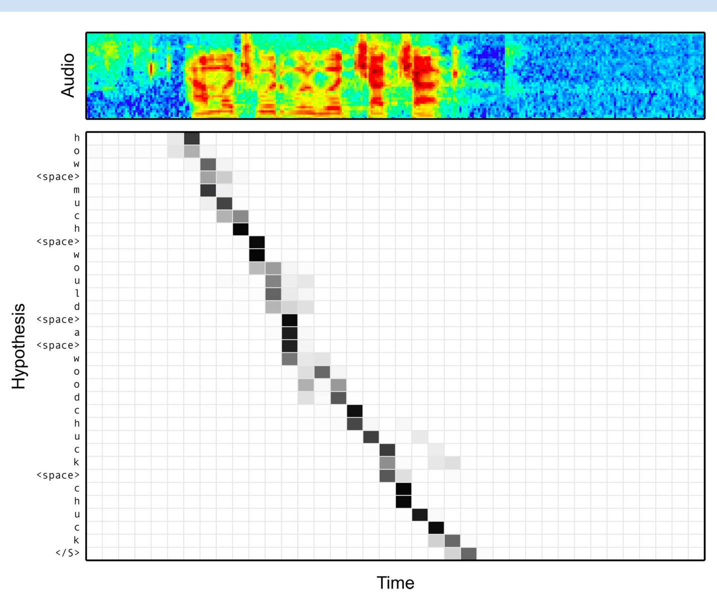
Abstract

Recurrent neural network models with an attention mechanism have proven to be extremely effective on a wide variety of sequence-to-sequence problems. However, the fact that soft attention mechanisms perform a pass over the entire input sequence when producing each element in the output sequence precludes their use in online settings and results in a quadratic time complexity. Based on the insight that the alignment between input and output sequence elements is monotonic in many problems of interest, we propose an end-to-end differentiable method for learning monotonic alignments which, at test time, enables computing attention online and in linear time. We validate our approach on sentence summarization, machine translation, and online speech recognition problems and achieve results competitive with existing sequence-to-sequence models.

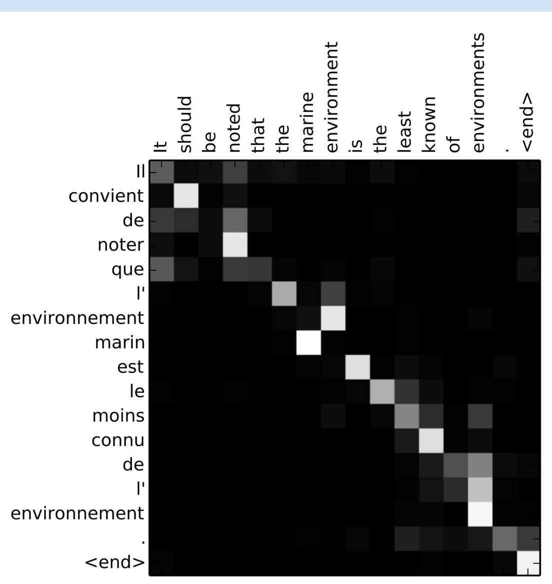
Sequence-to-sequence framework



Attention is often roughly monotonic

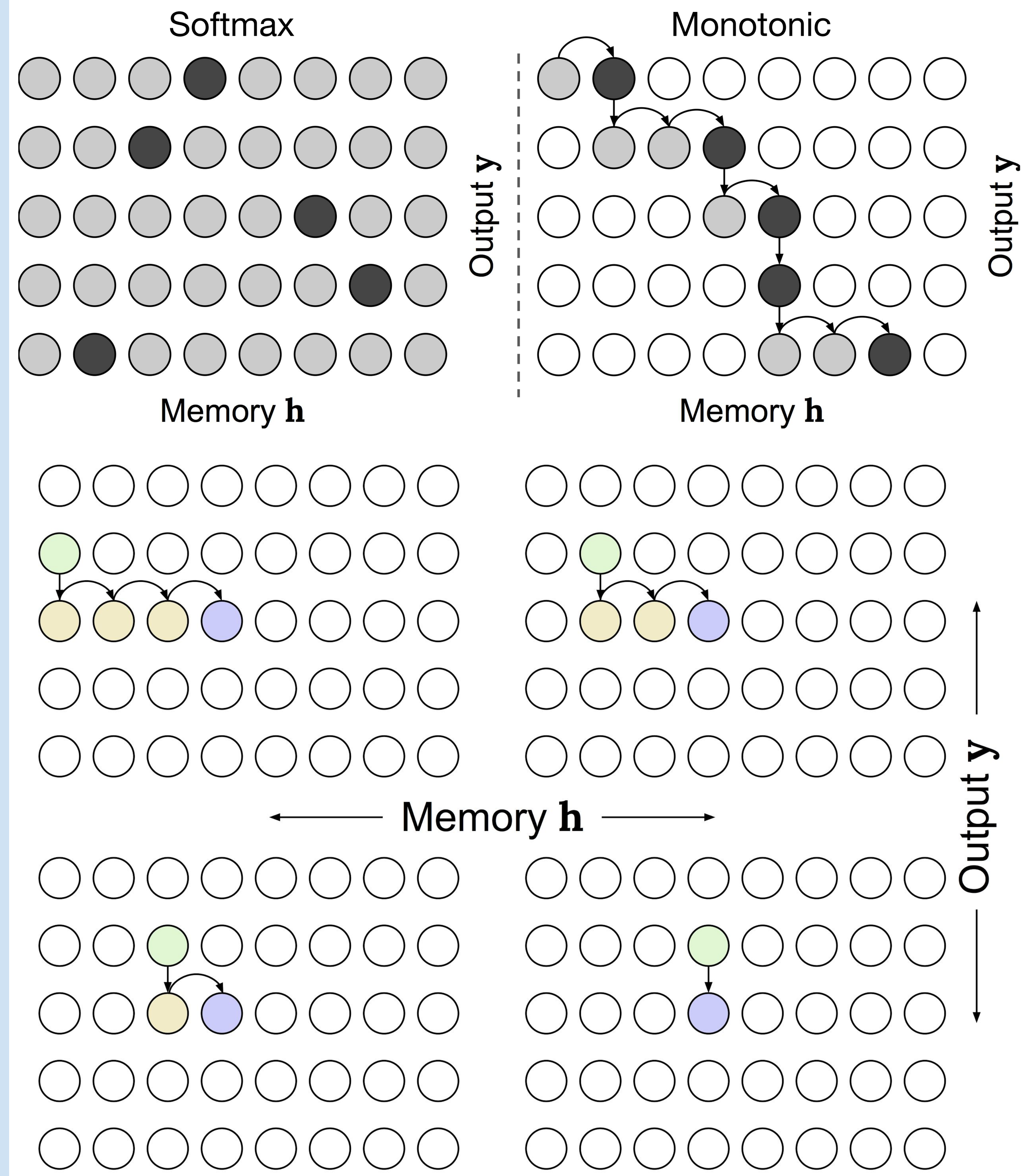


From *"Listen, Attend and Spell: A Neural Network for Large Vocabulary Conversational Speech Recognition"*, Chan, Jaitly, Le, Vinyals.



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

Monotonic attention

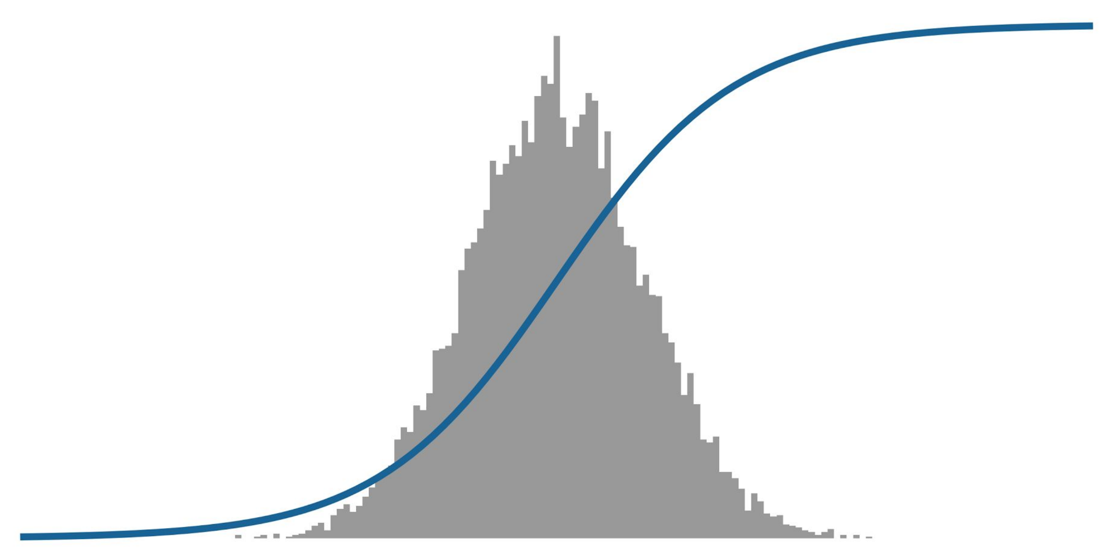


$$\text{attention}_{i,j} = \text{select}_{i,j} \sum_{k=1}^{j} \left(\text{attention}_{i-1,k} \prod_{l=k}^{j-1} (1 - \text{select}_{i,l}) \right)$$

$$\text{attention}_{i,j} = \text{select}_{i,j} \left((1 - \text{select}_{i,j-1}) \frac{\text{attention}_{i,j-1}}{\text{select}_{i,j-1}} + \text{attention}_{i-1,j} \right)$$

$$\text{attention}_{i} = \text{select}_{i}\, \texttt{cumprod}(1 - \text{select}_{i})\, \texttt{cumsum}\left(\frac{\text{attention}_{i-1}}{\texttt{cumprod}(1 - \text{select}_{i})} \right)$$

Encouraging discreteness



Training: $\text{select}_{i,j} = \sigma(\text{energy}_{i,j} + \mathcal{N}(0,1))$

Testing: $\text{select}_{i,j} = \mathbb{1}(\text{energy}_{i,j} > 0)$

Decoder algorithms

Softmax:

```
for each output timestep i:
  for each memory index j:
    energy[i, j] = energy_fn(state[i - 1], memory[j])
  attention = softmax(energy[i])
  context = sum(memory[j]*attention[j], axis=j)
  state[i] = update_rnn(state[i - 1], context)
```

Soft monotonic (training):

```
for each output timestep i:
  for each memory index j:
    energy[i, j] = energy_fn(state[i - 1], memory[j])
    select[i, j] = sigmoid(energy[i, j] + noise)
  attention[i] = (select[i]*cumprod(1 - select[i])
                  *cumsum(attention[i - 1]/
                          cumprod(1 - select[i]))
  context = sum(memory[j]*attention[i, j], axis=j)
  state[i] = update_rnn(state[i - 1], context)
```

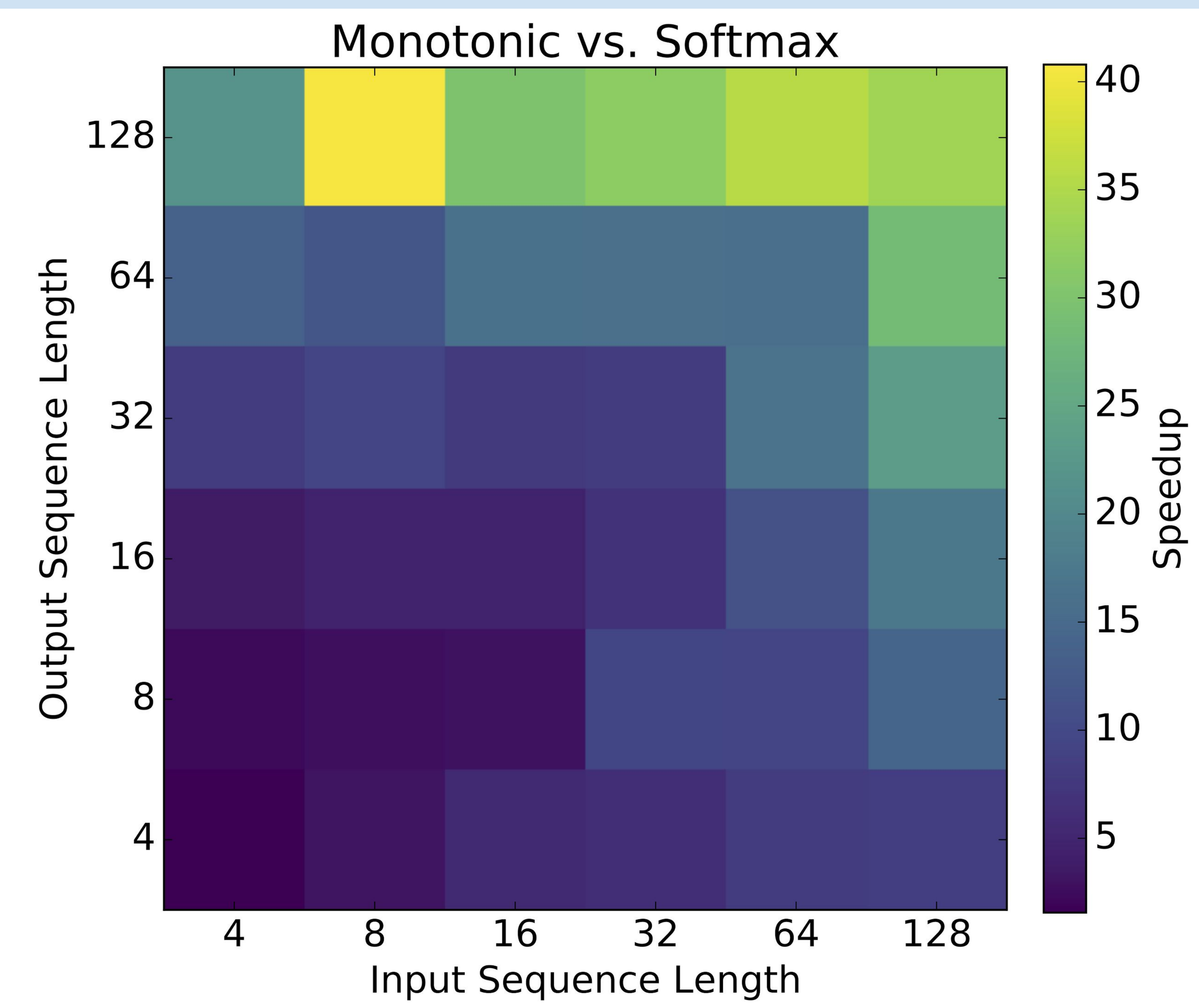
Hard monotonic (testing):

```
for each output timestep i:
  for each memory index j, starting from t[i - 1]:
    energy[i, j] = energy_fn(state[i - 1], memory[j])
    if energy[i, j] > 0:
      t[i] = j
      break
  state[i] = update_rnn(state[i - 1], memory[t[i]])
```

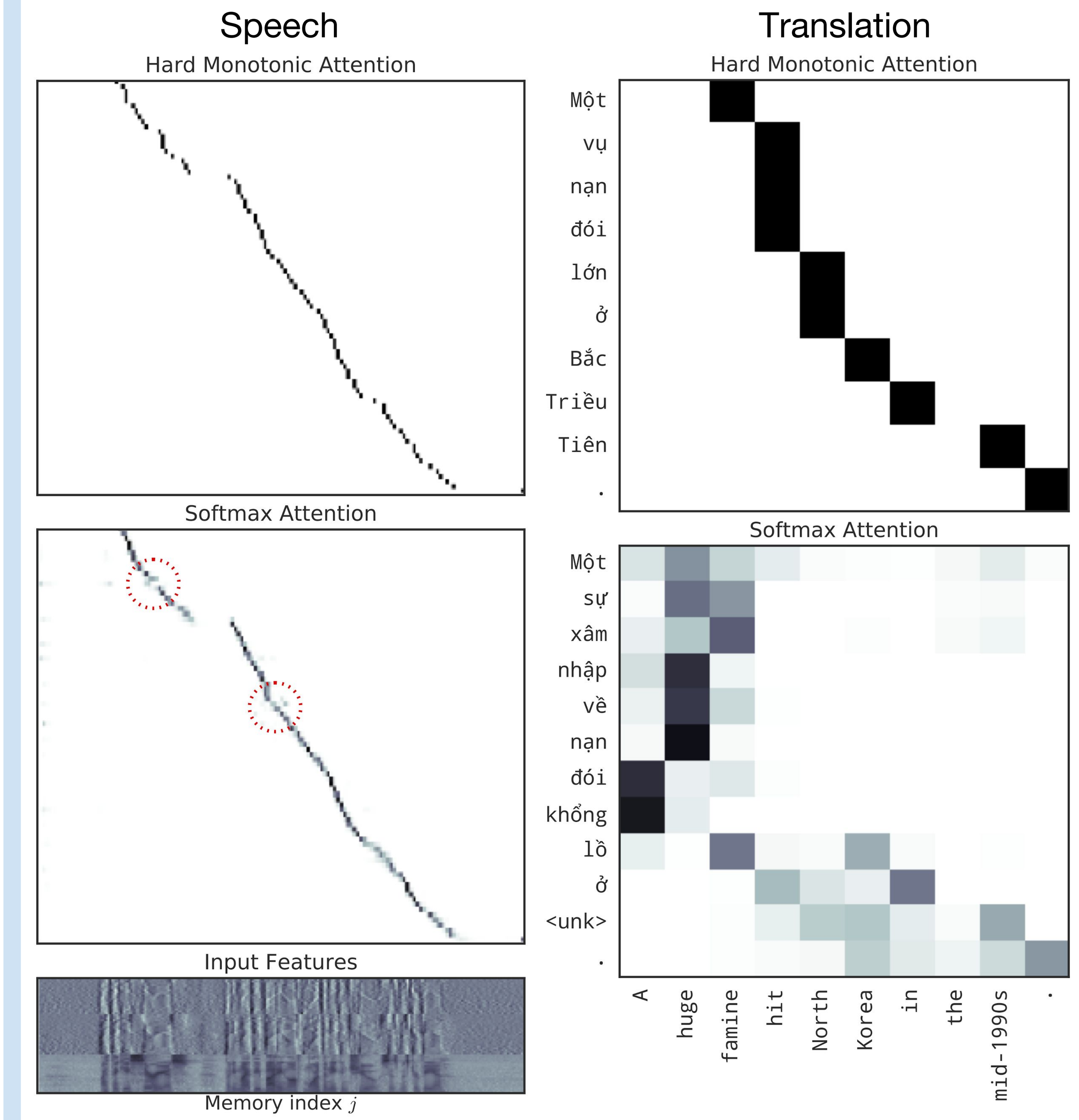
Experiments

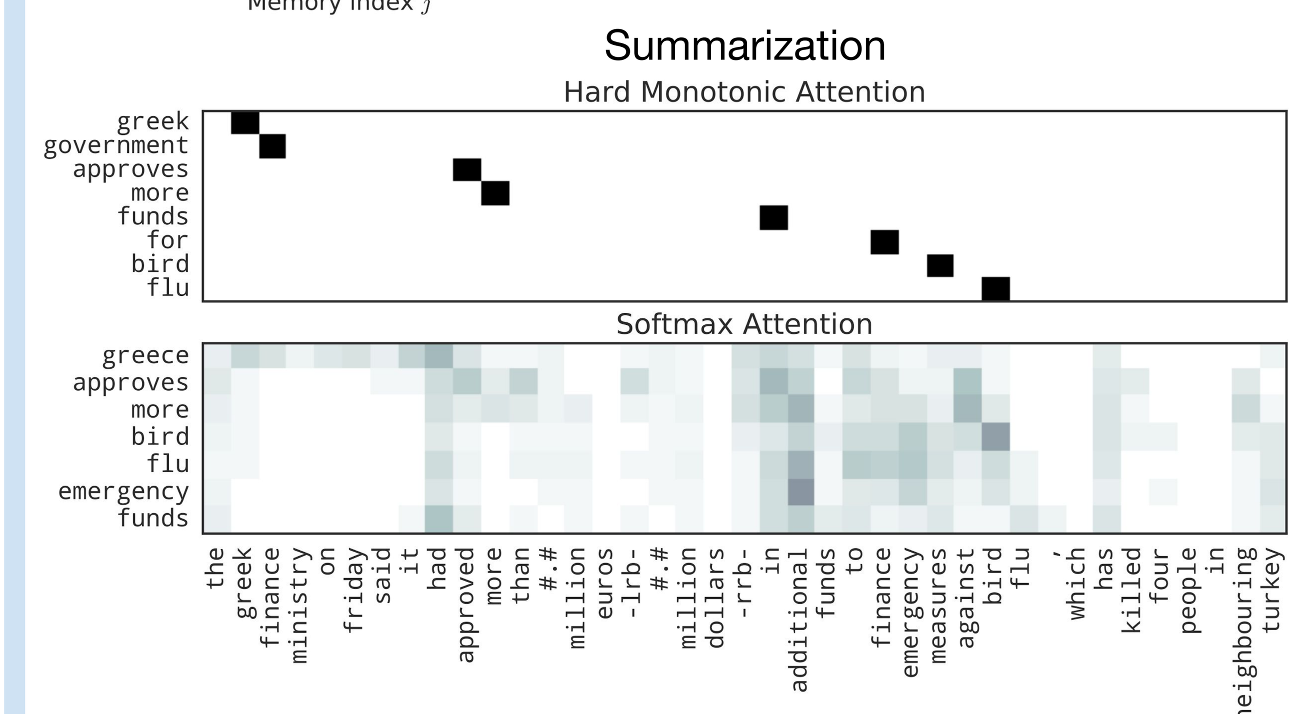
We ran experiments on machine translation, sentence summarization, and online speech recognition. In all cases, "soft" monotonic attention performed slightly worse than standard softmax attention and "hard" monotonic attention performed slightly worse than soft.

How much faster is it?



Learned alignments





Pointers

Blog post: `http://colinraffel.com/blog/`
arXiv paper: `https://arxiv.org/abs/1704.00784`
TensorFlow implementation in `tf.contrib.seq2seq`
Additional code: `https://github.com/craffel/mad`
Contact: `craffel@gmail.com`