Sparse Attentive Backtracking: Temporal credit assignment through reminding

Nan Rosemary Ke\textsuperscript{1,2}, Anirudh Goyal\textsuperscript{1}, Olexa Bilaniuk \textsuperscript{1}, Jonathan Binas\textsuperscript{1} Chris Pal\textsuperscript{2,4}, Mike Mozer \textsuperscript{3}, Yoshua Bengio\textsuperscript{1,5}

\textsuperscript{1}Mila, Université de Montréal
\textsuperscript{2}Mila, Polytechnique Montreal
\textsuperscript{3}University of Colorado, Boulder
\textsuperscript{4}Element AI
\textsuperscript{5}CIFAR Senior Fellow
Overview

- Recurrent neural networks
  - sequence modeling
- Training RNNs
  - backpropagation through time (BPTT)
- Attention mechanism
- Sparse attentive backtracking
Sequence modeling

Variable length input and (or) output.

- Speech recognition
  - variable length input, variable length output

- Image captioning
  - Fixed size input, variable length output

Sequence modeling

More examples

- Text
  - Language modeling
  - Language understanding
  - Sentiment analysis

- Videos
  - Video generation.
  - Video understanding.

- Biological data
  - Medical imaging
Recurrent neural networks (RNNs)

Handling variable length data

- **Variable** length input or output
- **Variable** order
  - ”In 2014, I visited Paris.”
  - ”I visited Paris in 2014.”
- Use **shared parameters** across time
Vanilla recurrent neural networks

- Parameters of the network
  - U, W, V
  - unrolled across time

\[ h_{t-1} = \begin{array}{c}
  h_0 \\
  h_1 \\
  h_2 \\
  \vdots \\
  h_T
\end{array} \]

*Christopher Olah – Understanding LSTM Networks*
Training RNNs

Backpropagation through time (BPTT)

\[
\frac{dE_2}{dU} = \frac{dE_2}{dh_2} (x_T^2 + \frac{dh_2}{dh_1} (x_T^1 + \frac{dh_1}{dh_0} x_T^0))
\]

\[\text{Christopher Olah – Understanding LSTM Networks}\]
Challenges with RNN training

Parameters are shared across time

- Number of parameters do not change with sequence length.
- Consequences
  - Optimization issue
  - Exploding or vanishing gradients
  - Assumption that same parameters can be used for different time steps.
Challenges with RNN training

Train to predict the future from the past

- $h_t$ is a lossy summary of $x_0, ..., x_t$
- Depending on criteria, $h_t$ decides what information to keep
- **Long term dependency**: if $y_t$ depends on distant past, then $h_t$ has to keep information from many timesteps ago.
Long term dependency

Example of long term dependency

- Question answering task.
- Answer is the first word.
Exploding and vanishing gradient

Challenges in learning long-term dependencies

- Exploding and vanishing gradient
Long short term memory (LSTM)

Gated recurrent neural networks that helps with long term dependency.

- Self-loop for gradients to flow for many steps
- Gates for learning what to remember or forget
- Long-short term memory (LSTM)

- Gated recurrent neural networks (GRU)
Long short term memory (LSTM)

Recurrent neural network with gates that dynamically decides what to put into, forget about and read from memory.

- Memory cell $c_t$
- Internal states $h_t$
- Gates for writing into, forgetting and reading from memory
Encoder decoder model

Summarizes the input into a single $h_t$ and decoder generates outputs conditioned on $h_t$.

- Encoder summarizes entire input sequence into a single vector $h_t$.
- Decoder generates outputs conditioned on $h_t$.
- Applications: machine translation, question answering tasks.
- Limitations: $h_t$ in encoder is bottleneck.
Attention mechanism

Removes the bottleneck in encoder decoder architecture using an attention mechanism.

- At each output step, learns an attention weight for each $h_0, ..., h_t$ in the encoder.
  \[
  a_j = \frac{e^{A(z_j, h_j)}}{\sum_{j'} e^{A(z_{j'}, h_{j'})}}
  \]
- Dynamically encodes into context vector at each time step.
- Decoder generates outputs at each step conditioned on context vector $c_{x_t}$. 

![Diagram of attention mechanism](image)
Limitations of BPTT

The most popular RNN training method is backpropagation through time (BPTT).

- Sequential in nature.
- Exploding and vanishing gradient
- Not biologically plausible
  - Detailed replay of all past events.
Credit assignment

- **Credit assignment**: The correct division and attribution of blame to one's past actions in leading to a final outcome.
- Credit assignment in **recurrent neural networks** uses backpropagation through time (BPTT).
  - Detailed memory of all past events
  - Assigns soft credit to almost all past events
  - Diffusion of credit? difficulty of learning long-term dependencies
• Humans selectively recall memories that are relevant to the current behavior.

• Automatic reminding:
  • Triggered by contextual features.
  • Can serve a useful computational role in ongoing cognition.
  • Can be used for credit assignment to past events?

• Assign credit through only a few states, instead of all states:
  • Sparse, local credit assignment.
  • How to pick the states to assign credit to?
Example: Driving on the highway, hear a loud popping sound. Didn’t think too much about it, 20 minutes later stopped by side of the road. Realized one of the tire has popped.

- What we tend to do?
  - Memory replay of event in context: Immediately brings back the memory of the loud popping sound 20min ago.

- what BPTT does?
  - BPTT will replay all events within the past 20min.
Maybe something more biologically inspired?

- What we tend to do?
  - Memory replay of event in context: Immediately brings back the memory of the loud popping sound 20min ago.

- What BPTT does?
  - BPTT will replay all events within the past 20min.
Credit assignment through a few states?

- Can we assign credit only through a few states?
- How to pick which states to assign credit to?
- RNN models does not support such operations in the past. Needs to make **architecture changes**.
  - Can change both forward and backward.
  - Or just change backward pass.
- Change both forward and backward pass
  - Forward dense, backward sparse
  - Forward sparse, backward sparse
Humans are trivially capable of assigning credit or blame to events even a long time after the fact, and do not need to replay all events from the present to the credited event sequentially and in reverse to do so.

- **Avoids competition** for the limited information-carrying capacity of the sequential path
- A simple form of **credit assignment**
- Imposes a **trade-off** that is absent in previous, dense self-attentive mechanisms: opening a connection to an interesting or useful timestep must be made at the price of excluding others.
Sparse attentive backtracking

- Use attention mechanism to select previous timestep to do backprop
  - Local backprop: truncated BPTT
  - Select previous hidden states - **sparsely**.
  - Skip-connections: natural for long-term dependency.
Algorithm 1 SAB-augmented LSTM

1: procedure SABCell \((h^{(t-1)}, c^{(t-1)}, x^{(t)})\)

Require: \(k_{top} > 0, k_{att} > 0, k_{trunc} > 0\)

Require: Memories \(m^{(i)} \in \mathcal{M}\)

Require: Previous hidden state \(h^{(t-1)}\)

Require: Previous cell state \(c^{(t-1)}\)

Require: Input \(x^{(t)}\)

2: \(\hat{h}^{(t)}, c^{(t)} \leftarrow \text{LSTMCell}(h^{(t-1)}, c^{(t-1)}, x^{(t)})\)

3: for all \(i \in 1 \ldots |\mathcal{M}|\) do

4: \(d^{(t)}_i \leftarrow W_1 m^{(i)} + W_2 \hat{h}^{(t)}\)

5: \(a^{(t)}_i \leftarrow W_3 \tanh(d^{(t)}_i)\)

6: \(a^{(t)}_{k_{top}} \leftarrow \text{sorted}(a^{(t)})[k_{top}+1]\)

7: \(\tilde{a}^{(t)} \leftarrow \text{ReLU}(a^{(t)} - a^{(t)}_{k_{top}})\)

8: \(s^{(t)} \leftarrow \sum_{m^{(i)} \in \mathcal{M}} \frac{\tilde{a}^{(t)}_i m^{(i)}}{\sum_{i} \tilde{a}^{(t)}_i}\)

9: \(h^{(t)} \leftarrow \hat{h}^{(t)} + s^{(t)}\)

10: \(y^{(t)} \leftarrow V_1 h^{(t)} + V_2 s^{(t)} + b\)

11: if \(t \equiv 0 \pmod{k_{att}}\) then

12: \(\mathcal{M}.\text{append}(h^{(t)})\)

13: return \(h^{(t)}, c^{(t)}, y^{(t)}\)
Sparse Attentive Backtracking

Forward pass

[Diagram of Sparse Attentive Backtracking with labels for Concat, Broadcast, MLP, Sparsifier, and RNN Cell]
Sparse Attentive Backtracking

Backward pass

Diagram: Network structure with nodes labeled as Concat, Broadcast, MLP, Sparsifier, and RNN Cell. Arrows indicate the flow of information between nodes.
## Copy task

<table>
<thead>
<tr>
<th>LSTM</th>
<th>$k_{trunc}$</th>
<th>$k_{top}$</th>
<th>Copying (T=100)</th>
<th>Copying (T=200)</th>
<th>Copying (T=300)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>acc. CE CE</td>
<td>acc. CE CE</td>
<td>acc. CE CE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CE CE CE</td>
<td>CE CE CE</td>
<td>CE CE CE</td>
</tr>
<tr>
<td>full BPTT</td>
<td></td>
<td></td>
<td>99.8 0.030 0.002</td>
<td>56.0 1.07 0.046</td>
<td>35.9 0.197 0.047</td>
</tr>
<tr>
<td>full self-attn.</td>
<td></td>
<td></td>
<td>100.0 0.0008 0.0000</td>
<td>100.0 0.001 0.000</td>
<td>100.0 0.002 7.5e-5</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>-</td>
<td>20.6 1.984 0.165</td>
<td>17.1 2.03 0.092</td>
<td>14.0 2.077 0.065</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>-</td>
<td>31.0 1.737 0.145</td>
<td>20.2 1.98 0.090</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>-</td>
<td>29.6 1.772 0.148</td>
<td>35.0 1.596 0.073</td>
<td></td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>-</td>
<td>30.5 1.714 0.143</td>
<td>35.8 1.61 0.073</td>
<td></td>
</tr>
<tr>
<td></td>
<td>150</td>
<td>-</td>
<td>- - -</td>
<td>25.7 1.848 0.197</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>24.4 1.857 0.058</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAB</td>
<td>1</td>
<td>1</td>
<td>57.9 1.041 0.087</td>
<td>39.9 1.516 0.069</td>
<td>43.1 0.231 0.045</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>5</td>
<td>100.0 0.001 0.000</td>
<td></td>
<td>89.1 0.383 0.012</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>5</td>
<td>100.0 0.000 0.000</td>
<td>100.0 0.000 0.000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>10</td>
<td>100.0 0.000 0.001</td>
<td>100.0 0.000 0.000</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Test accuracy and cross-entropy (CE) loss performance on the copying task with sequence lengths of T=100, 200, and 300. Accuracies are given in percent for the last 10 characters. CE_{10} corresponds to the CE loss on the last 10 characters. These results are with mental updates; Compare with Table 4 for without.
## Comparison to Transformers

<table>
<thead>
<tr>
<th>Image class.</th>
<th>pMNIST acc.</th>
<th>CIFAR10 acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LSTM</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>full BPTT</td>
<td>90.3</td>
<td>58.3</td>
</tr>
<tr>
<td>300</td>
<td>-</td>
<td>51.3</td>
</tr>
<tr>
<td><strong>SAB</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>5</td>
<td>89.8</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>90.9</td>
</tr>
<tr>
<td>50</td>
<td>10</td>
<td>94.2</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>64.5</td>
</tr>
<tr>
<td>Transformer (Vasvani’17)</td>
<td><strong>97.9</strong></td>
<td><strong>62.2</strong></td>
</tr>
</tbody>
</table>

Table 4: Test accuracy for the permuted MNIST and CIFAR10 classification tasks.
# Language modeling tasks

<table>
<thead>
<tr>
<th>Language</th>
<th>(k_{trunc})</th>
<th>(k_{top})</th>
<th>(k_{att})</th>
<th>PTB BPC</th>
<th>Text8 BPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>full BPTT</td>
<td></td>
<td></td>
<td></td>
<td>1.36</td>
<td>1.42</td>
</tr>
<tr>
<td>LSTM</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>1.47</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>-</td>
<td>-</td>
<td>1.44</td>
<td>1.56</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>-</td>
<td>-</td>
<td>1.40</td>
<td></td>
</tr>
<tr>
<td>SAB</td>
<td>10</td>
<td>5</td>
<td>10</td>
<td>1.42</td>
<td>1.47</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>1.40</td>
<td>1.45</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>5</td>
<td>20</td>
<td>1.39</td>
<td>1.45</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>10</td>
<td>20</td>
<td>1.37</td>
<td>1.44</td>
</tr>
</tbody>
</table>
Are mental updates important?

How important is backproping through the local updates (not just attention weights)?

<table>
<thead>
<tr>
<th>Ablation</th>
<th>$k_{trunc}$</th>
<th>$k_{iop}$</th>
<th>Copying, T=100</th>
<th>Adding, T=200CE</th>
</tr>
</thead>
<tbody>
<tr>
<td>no MU</td>
<td>1</td>
<td>1</td>
<td>49.0 1.252 0.104</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>5</td>
<td>98.3 0.042 0.0036</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>10</td>
<td>99.6 0.022 0.0018</td>
<td>2.171e-6</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>all</td>
<td>40.5 1.529 0.127</td>
<td></td>
</tr>
</tbody>
</table>
Generalization

- Generalization on longer sequences

<table>
<thead>
<tr>
<th>Copy len. (T)</th>
<th>LSTM</th>
<th>LSTM + self-a.</th>
<th>SAB</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>99%</td>
<td>100%</td>
<td>99%</td>
</tr>
<tr>
<td>200</td>
<td>34%</td>
<td>52%</td>
<td>95%</td>
</tr>
<tr>
<td>300</td>
<td>25%</td>
<td>28%</td>
<td>83%</td>
</tr>
<tr>
<td>400</td>
<td>21%</td>
<td>20%</td>
<td>75%</td>
</tr>
<tr>
<td>2000</td>
<td>12%</td>
<td>12%</td>
<td>47%</td>
</tr>
<tr>
<td>5000</td>
<td>12%</td>
<td>OOM</td>
<td>41%</td>
</tr>
</tbody>
</table>

Generalization test for models trained on copy task with T=100
Long term dependency tasks

Attention heat map

- Learned attention over different timesteps during training

Copy Task with $T = 200$
Future work

• Content-based rule for writing to memory
  • Reduces memory storage
  • How to decide what to write to memory?
  • Humans show a systematic dependence on many content: salient, extreme, unusual, and unexpected experiences are more likely to be stored and subsequently remembered

• Credit assignment through more abstract states/ memory?
• Model-based reinforcement learning
• The source code is now open-source, at
  https://github.com/nke001/sparse_attentive_backtracking_release