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

*A woman is throwing a frisbee in a park.*
*A stop sign is on a road with a mountain in the background.*
*A group of people sitting on a boat in the water.*
*A giraffe standing in a forest with trees in the background.*
More examples

- Text
  - Language modeling
  - Language understanding
  - Sentiment analysis

- Videos
  - Video generation.
  - Video understanding.

- Biological data
  - Medical imaging
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

*Christopher Olah – Understanding LSTM Networks*
Training RNNs

Backpropagation through time (BPTT)

$$\frac{dE_2}{dU} = \frac{dE_2}{dh_2} (x_2^T + \frac{dh_2}{dh_1} (x_1^T + \frac{dh_1}{dh_0} x_0^T))$$

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.
Example of long term dependency

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

Challenges in learn long term dependencies

- Exploding and vanishing gradient
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)

Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." *Neural computation* 9.8 (1997): 1735-1780.

- Gated recurrent neural networks (GRU)

Cho, Kyunghyun, et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." arXiv preprint arXiv:1406.1078 (2014).
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

*Christopher Olah – Understanding LSTM Networks*
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.
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-url)
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
Sparse replay

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: \(d_{i}^{(t)} \leftarrow W_{1}m^{(i)} + W_{2}\hat{h}^{(t)}\)

5: \(a_{i}^{(t)} \leftarrow W_{3}\tanh(d_{i}^{(t)})\)

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

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

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

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
Sparse Attentive Backtracking

Backward pass
## Copy task

| k<sub>trunc</sub> | k<sub>top</sub> | Copying (T=100) | Copying (T=200) | Copying (T=300) |
|------------------|----------------|------------------|------------------|------------------|
|                  |                | acc. CE<sub>10</sub> CE | acc. CE<sub>10</sub> CE | acc. CE<sub>10</sub> CE |
| full BPTT        |                | 99.8 0.030 0.002 | 56.0 1.07 0.046 | 35.9 0.197 0.047 |
| full self-attn.  |                | 100.0 0.0008 0.0000 | 100.0 0.001 0.000 | 100.0 0.002 7.5e-5 |
| LSTM             |                |                  |                  |                  |
| 1                | -              | 20.6 1.984 0.165 | 17.1 2.03 0.092 | 14.0 2.077 0.065 |
| 5                | -              | 31.0 1.737 0.145 | 20.2 1.98 0.090 |                  |
| 10               | -              | 29.6 1.772 0.148 | 35.8 1.61 0.073 | 25.7 1.848 0.197 |
| 20               | -              | 30.5 1.714 0.143 | 35.0 1.596 0.073 | 24.4 1.857 0.058 |
| 150              | -              | -                | -                | -                |

| SAB              |                |                  |                  |                  |
| 1 1              |                | 57.9 1.041 0.087 | 39.9 1.516 0.069 | 43.1 0.231 0.045 |
| 1 5              | 100.0 0.001 0.000 |                  |                  | 89.1 0.383 0.012 |
| 5 5              | 100.0 0.000 0.000 | 100.0 0.000 0.000 | 99.9 0.007 0.001 |
| 10 10            | 100.0 0.000 0.001 | 100.0 0.000 0.000 |                  |                  |

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<sub>10</sub> 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

| Image class. | pMNIST acc. | CIFAR10 acc. |
|--------------|-------------|--------------|
| **LSTM**     |             |              |
| full BPTT    | 90.3        | 58.3         |
| 300          | -           | 51.3         |
| **SAB**      |             |              |
| 20           | 5           | 89.8         |
| 20           | 10          | 90.9         |
| 50           | 10          | 94.2         |
| 16           | 10          | 64.5         |
| Transformer (Vaswani’17) | **97.9** | **62.2** |

Table 4: Test accuracy for the permuted MNIST and CIFAR10 classification tasks.
| Language | $k_{\text{trunc}}$ | $k_{\text{top}}$ | $k_{\text{att}}$ | PTB BPC | Text8 BPC |
|----------|------------------|-----------------|-----------------|--------|---------|
| full BPTT |                  |                 |                 | 1.36   | 1.42    |
| LSTM     | 1                | -               | -               | 1.47   |         |
|          | 5                | -               | -               | 1.44   | 1.56    |
|          | 20               | -               | -               | 1.40   |         |
| SAB      | 10               | 5               | 10              | 1.42   | 1.47    |
|          | 10               | 10              | 10              | 1.40   | 1.45    |
|          | 20               | 5               | 20              | 1.39   | 1.45    |
|          | 20               | 10              | 20              | 1.37   | 1.44    |
Are mental updates important?

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

| Ablation | Copying, T=100 | Adding, T=200 CE |
|----------|----------------|-----------------|
|          | $k_{trunc}$ | $k_{top}$ | acc. | CE$_{last10}$ | CE | |
| no MU    | 1 | 1 | 49.0 | 1.252 | 0.104 | |
|          | 5 | 5 | 98.3 | 0.042 | 0.0036 | |
|          | 10 | 10 | 99.6 | 0.022 | 0.0018 | |
| 5 all    | 40.5 | 1.529 | 0.127 | 2.171e-6 | |
Generalization

- Generalization on longer sequences

### Transfer Learning Results

| Copy len. (T) | LSTM | LSTM +self-a. | SAB |
|---------------|------|---------------|-----|
| 100           | 99%  | 100%          | 99% |
| 200           | 34%  | 52%           | 95% |
| 300           | 25%  | 28%           | 83% |
| 400           | 21%  | 20%           | 75% |
| 2000          | 12%  | 12%           | 47% |
| 5000          | 12%  | OOM           | 41% |

Generalization test for models trained on copy task with T=100

![Graph showing accuracy of last 10 digits for different sequence lengths]
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