Response to Hochreiter and Schmidhuber (1997)

Matthew Evans

March 27, 2025

Overview

The Long Short-Term Memory (LSTM)[1] model addresses a critical limitation in learning systems: the inability to retain information over long time spans. Traditional methods often fail in tasks requiring long-term dependencies, such as speech recognition or contextual language understanding, due to issues like forgetting or misinterpreting past inputs.

LSTM introduces a novel architecture designed to store and retrieve key information over extended periods. By leveraging *memory cells* and *gating* mechanisms, it enables effective learning of patterns across long sequences, overcoming challenges that hindered earlier approaches.

Approach

Prior Efforts

Prior to LSTM, recurrent neural networks (RNNs) trained with methods like backpropagation through time (BPTT) or real-time recurrent learning (RTRL) faced severe challenges due to vanishing or exploding gradients, which hindered their ability to learn long-term dependencies and limited their effectiveness to short sequences. These issues arose from the repeated multiplication of weights during backpropagation, causing gradients to either shrink exponentially, making learning ineffective, or grow uncontrollably, destabilizing training. Attempts to address these problems included introducing time-delayed neural networks (TDNNs)[2], which incorporated fixed delays to capture temporal dependencies, and designing hierarchical RNNs (chunker systems)[3] with multiple layers to model complex sequences. However, these approaches achieved limited success, often requiring extensive manual tuning of delay parameters or layer configurations, further complicating the training process.

Novelty of LSTM

LSTM's key innovation is the *memory cell*, which maintains information over long periods by leveraging *gates* that learn to store, update, and control its influence. The LSTM memory cell is substituted for the plain hidden state of a plain RNN. This design ensures stable error flow, addressing vanishing and exploding gradients, while remaining computationally efficient and scalable to complex, long-sequence tasks.

Memory Cells and Gates

LSTM memory cells are designed to preserve information across time by default, only modifying it when necessary. Central to this design is the self-connected unit, or *constant error carousel* (CEC), which ensures consistent gradient flow during backpropagation. The authors' architecture includes an input gate, which controls when new information enters the memory cell, and an output gate, which controls when stored information is read out.

Gates, whose parameters are learnable, use multiplicative operations to regulate the reading and writing of memory cells c_t. This structure allows the network to selectively retain, access, or discard information, making it well-suited for tasks involving long-range dependencies. The result is a model that is both robust and trainable, addressing key limitations of traditional RNNs.

Input Gates

Input gates control whether new information is allowed into the memory cell. The input gate's activation i_t is given by

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i).$$

The *candidate value* $\tilde{c}_t$ is given by

$$\tilde{c}_t = \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c).$$

The memory cell's state is then incremented by $\tilde{c}_t$ proportional to the input activation i_t, and is given by

$$c_t = c_{t-1} + i_t \cdot \tilde{c}_t.$$

Hence, if the input gate's activation is low, the memory will remain relatively unchanged.

Output Gate

Output gates control the proportion of c_t to be revealed as the output h_t. The gate activation is given by

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o).$$

And the revealed output is given by

$$h_t = o_t \cdot \tanh(c_t).$$

Note the intentional choice of activation functions. $\sigma(x) : x \to [0, 1]$ throttles the update while $\tanh(x) : x \to [-1, 1]$ allows for both negative and positive updates to memory, helping to center activations around zero.

Constant Error Carousel

Like other networks, networks utilizing LSTM are trained via backpropagation, using gradients. Observe the gradient for the memory cell update is constant.

$$\frac{\partial}{\partial c_{t-1}}(c_t = c_{t-1} + i_t \cdot \tilde{c}_t)$$
$$\frac{\partial c_t}{\partial c_{t-1}} = 1$$

This *constant* error ensures that gradients can flow backward through time without diminishing. As a result, it prevents the vanishing gradient problem, enabling the network to learn dependencies from earlier time steps effectively.

Measures of Success

The authors validated LSTM through experiments on tasks with long time lags, comparing it to methods like BPTT[4, 5], RTRL[6], and Elman networks[7]. They measured success by the model's ability to classify sequences with perfect accuracy, often achieving zero errors across thousands of test examples. On benchmarks such as the embedded Reber grammar and synthetic tasks designed to test long-term memory, LSTM consistently outperformed competing methods. It required fewer training iterations and excelled on tasks with time lags exceeding 1,000 steps, where other approaches failed, demonstrating its efficiency and robustness.

Considerations

LSTM's strength lies in its ability to learn long-term dependencies by maintaining stable error flow through memory cells, effectively addressing vanishing and exploding gradients. Its gating mechanisms provide precise control over information storage and retrieval, ensuring adaptability across diverse sequence learning tasks, while maintaining computational efficiency.

Despite its strengths, LSTM has limitations. Its complexity, with more parameters than traditional RNNs, can lead to longer training times and require careful tuning of hyperparameters. Early in training, memory cells may act as static biases rather than learning useful information. While the authors suggest mitigation strategies, such as sequential network construction and bias initialization, these add design overhead. Nonetheless, LSTM's robustness in handling long-term dependencies solidifies its significance.

Impact

The innovations in this paper revolutionized sequence modeling by enabling effective learning of long-term dependencies. LSTM addressed challenges in tasks like language modeling, speech recognition, and time-series forecasting, transitioning from theoretical benchmarks to widespread real-world applications, solidifying its foundational role in machine learning.

LSTM has become a cornerstone of modern machine learning, powering advancements in natural language processing, machine translation, handwriting recognition, and more. Its architecture inspired gated recurrent units and marked a pivotal shift, overcoming theoretical barriers to enable impactful real-world applications.

References

[1] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[2] Kevin J. Lang, Alex H. Waibel, and Geoffrey E. Hinton. A time-delay neural network architecture for isolated word recognition. *Neural Networks*, 3(1):23–43, 1990.

[3] Jürgen Schmidhuber. Learning complex, extended sequences using the principle of history compression. *Neural Computation*, 4(2):234–242, 1992.

[4] Paul J. Werbos. Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, 1(4):339–356, 1988.

[5] Ronald J. Williams and David Zipser. Gradient-based learning algorithms for recurrent networks and their computational complexity. In Yves Chauvin and David E. Rumelhart, editors, *Back-propagation: Theory, architectures and applications*, pages 433–486. Erlbaum, Hillsdale, NJ, 1992.

[6] A. J. Robinson and F. Fallside. The utility driven dynamic error propagation network. Technical Report CUED/F-INFENG/TR.1, Cambridge Univ., 1987.

[7] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.