Explicit Gradient Linking Mechanism in Neural Networks with External Memory

Shousei Masuda
Hokuto City Kobuchizawa Junior High School (individual research)

August 2025

*The author is a non-native English speaker and would like to acknowledge that the manuscript was translated from their native language.

1 Abstract

This study addresses a fundamental challenge in neural networks with external memory: the disruption of gradient flow between memory write and read operations. We propose an explicit gradient linking mechanism that uses time-series ID tokens to reconnect these interrupted gradients.
Results on the Copy Task clearly demonstrate the method's superior efficiency and also reveal its sensitivity to hyperparameters. Furthermore, analysis of attention heatmaps provides visual evidence that the model autonomously learns an efficient and coordinated memory utilization strategy. However, experiments also revealed that the mechanism's inherent property of gradient activation can lead to instability under high learning rates.
These results indicate that our proposed method effectively reconstructs the feedback loop, paving a new path for stable and efficient learning in memory networks.

2 Introduction

2.1 Background

Recent remarkable advancements in deep learning have significantly enhanced the capability of neural networks to process complex information through the introduction of the concept of external memory. Approaches such as Neural Turing Machines (NTM) [1] and Differentiable Neural Computers (DNC) [2] have opened new possibilities for reasoning and time-series tasks by allowing models to temporarily store and retrieve information as needed.

2.2 Problem Statement

However, a fundamental challenge persists in learning such memory-augmented neural networks: the disruption of gradient flow between the **write** operation to memory and the **read** operation from it. This gradient discontinuity hinders the model's ability to learn effective memory utilization strategies, leading to learning stagnation and inefficiency.

2.3 Overview of the Proposed Mechanism

This study proposes a novel approach to address this gradient discontinuity problem: an explicit gradient linking mechanism between **read** and **write** operations using time-series ID tokens. Specifically, gradients arising from read operations are temporarily stored in a global memory (`grad_list`) along with unique ID tokens. During the backpropagation of subsequent write operations, these stored gradients are summed with the existing gradients corresponding to the same ID token, thereby reconstructing the interrupted gradient flow. This mechanism enables the model to effectively receive feedback on how to write information now so that it is useful for future reads, facilitating more efficient learning of memory utilization strategies. This method aims to stabilize gradient propagation and significantly improve learning performance even in complex models.

3 Related Work

The advancement of deep learning, particularly recurrent neural networks (RNNs) and their derivatives, has significantly improved the ability to learn long-term dependencies in complex sequence data [3]. In recent years, attempts have been made to further enhance the information processing capabilities and reasoning abilities of these models by introducing the concept of external memory.

3.1 Memory Augmented Neural Networks (MANNs)

Memory Augmented Neural Networks (MANNs) represent an effort to significantly expand the memory capacity and reasoning capabilities of recurrent models by incorporating external readable and writable memory interfaces. Prominent examples in this field include Neural Turing Machines (NTM) [1] and their evolution, Differentiable Neural Computers (DNC) [2]. These models have demonstrated remarkable performance on complex algorithmic tasks, such as Copy Task and Sorting Task, as well as reasoning tasks like question answering, through their interaction with external memory.

However, a major challenge recognized in conventional MANNs like NTM and DNC has been the instability of gradient flow, particularly the problem of vanishing or exploding gradients during long-term memory operations. This complexity arises because memory access operations form intricate computational graphs, making learning difficult. Consequently, models have been limited in their ability to fully exploit their potential by effectively learning memory utilization strategies. Attention Mechanisms The Attention Mechanism has been widely adopted in recent years, notably in Transformers [4], serving as a powerful mechanism that dynamically learns the relevance between different positions within a sequence or the importance of information relative to a query. This enables models to efficiently capture long-range dependencies and complex patterns by focusing on specific information. Our proposed Explicit Gradient Linking Mechanism shares common ground with attention in that it employs an attention-like mechanism for addressing memory during write and read operations. However, while general Attention Mechanisms primarily aim at integrating information within a sequence or learning relationships, our proposed mechanism fundamentally differs by focusing on addressing a distinct challenge: explicitly reconstructing long-term gradient propagation paths between external memory and the controller, thereby enhancing gradient flow between the past and future via memory.

3.2 Our Contribution in Context

While existing MANNs have demonstrated the potential to overcome the limitations of memory capacity through the introduction of external memory, the fundamental challenge of disrupted gradient flow due to complex memory operations has remained a major bottleneck in their learning. The explicit gradient linking mechanism proposed in this study directly addresses this long-standing problem.

Our method temporarily stores gradients generated by read operations (`grad_read`) in a `grad_list` along with ID tokens, and then explicitly links this gradient information to subsequent write operations. This approach strengthens and reactivates the gradient propagation paths between the controller and memory, which tend to be interrupted in conventional MANNs over extended time steps. Consequently, this enables models to learn memory utilization strategies more efficiently, mitigating vanishing gradients and achieving improved learning stability and performance. This represents the primary novelty and academic contribution of this study, distinguishing it from any existing MANNs or Attention-based approaches.

4 Algorithmic Details

4.1 Overview: Architecture of the Gradient-Linked Memory Interface

The proposed memory interface is represented by the `AttentionMemoryInterface` class. This class receives input data and performs write and read operations on its internal memory (`self.memory`). The main components are as follows:

- `write_net`: Transforms input data into values to be written to memory (`mem_values`).
- `write_query_net`: Generates a query that determines the memory address during write operations.
- `read_query_net`: Generates a query that retrieves information from memory during read operations.
- `read_out_net`: Transforms the read result into the final output format.
- `self.memory`: The actual memory cells, with a fixed number of slots (`num_slots`) and dimensions (`mem_dim`).

The most important feature of this method is that the interface can operate in a **gradient link mode.** In gradient link mode, in addition to standard automatic differentiation, the custom gradient propagation mechanism described below is activated.

4.2 Explicit Gradient Linking Mechanism

In conventional memory networks, gradients originating from read operations often fail to propagate correctly back to the past write operations that produced the content. This issue arises from in-place memory updates and non-continuous processing steps between read and write operations. Our proposed method addresses this problem with an innovative two-stage process

4.2.1 Gradient Storage during Read Operations

When the `read_with_attention_grad_link` function is called, a unique `id_token` (e.g., the time step t) associated with each read operation is introduced. A custom gradient hook is then registered to the final output of the read operation, `read_output`. This hook is triggered when a gradient (grad) backpropagates to `read_output`, saving that gradient along with the corresponding `id_token` temporarily in a global dictionary `grad_list`. This ensures that once a future read operation is completed and its loss gradient is calculated, this gradient is not immediately lost but is held, associated with a specific `id_token`. This `grad_list` then serves as "feedback from the future" to be utilized during the backpropagation of subsequent write operations.

4.2.2 Gradient Reconnection during Write Operations (`WriteFunctionGradLink`)

The core of the proposed method lies in a custom class `WriteFunctionGradLink` that inherits from PyTorch's `torch.autograd.Function`. This class fully controls the `forward` and `backward` passes of the memory write operation.

- `forward` method: It receives `input_tensor` (the value to write), the corresponding `id_token`, the current `memory_tensor`, and `write_weights` (attention weights for where to write in memory). It then updates the memory based on this information and returns the new memory state (`memory_out`). During this process, `ctx.save_for_backward` is used to store tensors and the `id_token` necessary for backpropagation.
- `backward` method: Upon receiving the gradient `grad_output` for the memory update operation, it uses the stored `ctx.id_token` to retrieve the corresponding "gradient from the read operation" (`grad_read`) from `grad_list`. The retrieved `grad_read` is immediately removed from `grad_list`.

This `grad_read` is added to `grad_output`. This gradient summation is the novelty of this method. It ensures that the gradient to the write operation is guided not only by its own forward pass result but also by the **direction expected by future read operations.** In other words, information on how past writes affect future reads is directly fed back as gradients. Specific gradient calculations for `input_tensor`, `memory_tensor`, and `write_weights` are performed efficiently in a vectorized manner, using the saved `input_tensor`, `write_weights`, `memory_tensor`, and `grad_read`.

This explicit gradient linking enables the model not only to store information in memory but also to learn strategies for optimizing its future use. This facilitates meta-learning on **not just what to write to memory, but how to write it,** significantly improving learning stability and efficiency. Furthermore, by linking each operation with an `id_token`, the method accurately maintains gradient correspondences even when multiple writes and reads occur in parallel.

4.3 Mathematical Formulation of Memory Operations

Let the memory state at time t be $M_t \in \mathbb{R}^{S \times D}$, where S is the number of memory slots and D is the dimension of each slot. The input vector to be written is $x_t \in \mathbb{R}^D$, and the attention weights for the write and read operations are $w_t, r_t \in \mathbb{R}^S$.

4.3.1 Write Operation

The write operation updates the memory state based on the input vector $x_t$and the write attention weights w_t . The new memory state M_{t+1} is a combination of the previous memory Mt and the new information. The update rule is formulated as

$$M_{t+1} = M_t \odot (1 - w_t \mathbf{1}^T) + w_t x_t^T$$

where 1 is a vector with all elements equal to 1. This expression means that we first decay the existing memory contents according to w_t (an erase operation), and then add new information x_t weighted by w_t (a write operation).

4.3.2 Read Operation

The read operation retrieves a new output vector $o_t \in \mathbb{R}^D$by taking a weighted sum of the memory rows, using the read attention weights r_t

$$o_t = \sum_{j=1}^{S} r_t[j] M[j]$$

4.4 Explicit Gradient Linking

The core of the proposed method is the custom gradient propagation. During the backward pass, the gradient from a future read operation, identified by its time step t, is stored in a global list

$$G_{read,t} = \nabla o_t \mathcal{L}$$

This stored gradient is then retrieved during the backward pass of the corresponding write operation at time $t - 1$. The standard gradient propagating through the memory update, $\nabla M_t \mathcal{L}standard$, is augmented by adding this feedback gradient from the read operation. The final effective gradient for the memory is:

$$\nabla M_t \mathcal{L}total = \nabla M_t \mathcal{L}standard + G_{read,t}$$

This combined gradient is then used to calculate the gradients for the parameters of the write operation, such as the `write_net` and `write_query_net`, ensuring that the write operation learns how to store information in a way that is beneficial for future reads

5 Experiments

5.1 Experimental Setup

To evaluate the effectiveness of the explicit gradient linking mechanism proposed in this study, we set up the Copy Task, a representative benchmark for measuring the memory capabilities of neural networks. This task requires the model to accurately remember a given input sequence and then output the exact same sequence. This makes it a crucial task for precise coordination between memory write and read operations, clearly highlighting the difference in learning efficiency and performance with and without gradient linking.

5.2 Task Details

The model receives a randomly generated fixed-length input sequence, where each time step consists of a high-dimensional vector (e.g., `input_dim=80`). The model aims to record this input sequence in its internal memory and then reconstruct and output the sequence in the exact correct order.

5.3 Model

We used the `AttentionMemoryInterface` model incorporating the proposed method.

- `input_dim = 80` : Dimension of the input vector at each time step.
- `mem_dim = 160` : Dimension of each element in a memory slot.
- `num_slots = 100` : Total number of available slots in memory.
- `seq_len = 50` : Length of the input sequence
- batch_size = 32: Number of sequences in each training batch.

5.4 Optimization

- Loss Function: Mean Squared Error (MSELoss) was used to minimize the difference between the model's output sequence and the target input sequence.
- Optimizer: AdamW optimizer($lr = 1 \times 10^{-4}$) was used.

5.5 Gradient Clipping

Gradient norm clipping with `clip_value = 1.0` was applied to prevent gradient explosion. It should be noted that when using a relatively high learning rate such as $lr = 1 \times 10^{-3}$, gradient explosion can occur depending on the magnitude of the gradients, making this gradient clipping an indispensable element for learning stability.

5.6 Comparison

To clearly demonstrate the effectiveness of the proposed method, we trained the model under two conditions and compared their performance:

5.6.1 With Gradient Link (Proposed Method)

A model with explicit gradient linking between read and write operations using `id_token`.

5.6.2 Without Gradient Link (Baseline)

A model with the gradient linking mechanism disabled, exhibiting conventional behavior where gradient flow between memory operations is interrupted. In this model, write gradients do not receive direct feedback in a manner that efficiently guides memory write operations based on future reads.

5.6.3 Number of Training Epochs

Each model was trained for 300 epochs (or more) to observe the loss convergence behavior.

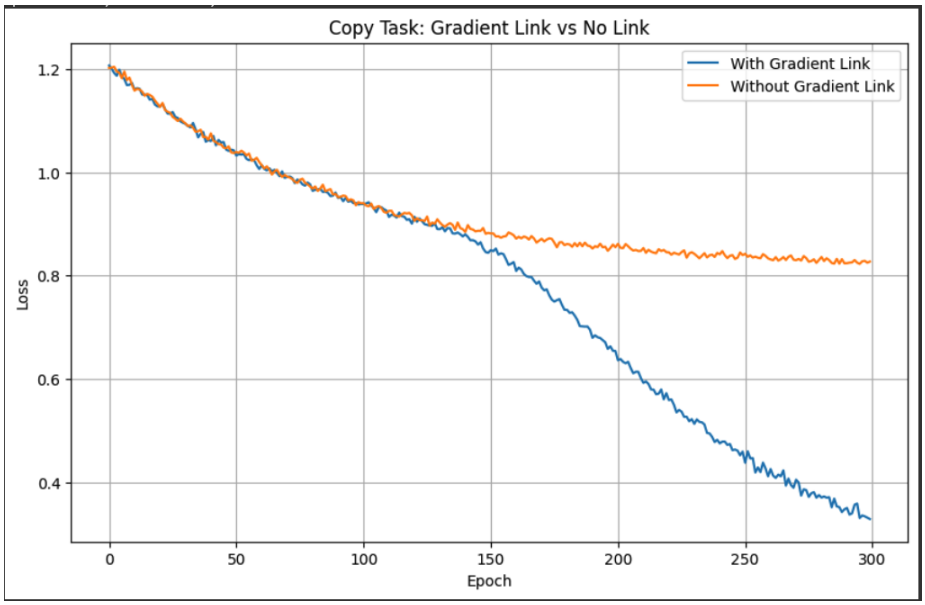

Figure 1: shows the progression of the loss curve for the Copy Task with and without gradient linking.

6 Results and Discussion

The results of this experiment clearly demonstrate that the proposed explicit gradient link mechanism significantly impacts the learning performance of neural networks with external memory. Although the baseline model occasionally exhibits temporary decreases in loss due to random initialization or optimization variance, these fluctuations do not surpass the performance of the proposed method. Even when such transient improvements occur, the model with explicit gradient linking consistently maintains lower overall loss and faster convergence.

6.1 Comparison of Learning Performance

- The model without gradient links (orange line) exhibited stagnation in loss reduction after approximately 150 epochs in this specific run, stabilizing at a high loss value of around 0.85. This suggests that the model failed to establish a learning progression where writing to memory consistently leads to successful subsequent retrieval.

- It should be noted that, while in some runs with different initial parameter or hyperparameter settings, loss reduction was eventually observed after a prolonged plateau, this phenomenon is not the norm. This eventual success represents a trial-and-error process stemming from unstable gradient propagation and cannot be considered an efficient, reproducible

learning strategy. Due to gradient discontinuities, the model struggles to autonomously establish an effective feedback loop for efficient memory utilization, resulting in highly inefficient task solving.

- In contrast, the model with gradient linking (blue line) consistently reduced loss from the early learning stages and ultimately converged to a low loss value of approximately 0.35. This remarkable performance improvement strongly indicates that the proposed gradient link mechanism effectively reconstructs the flow of gradients between memory write and read operations, enabling the model to learn long-term dependencies essential for task success with overwhelming speed and stability.

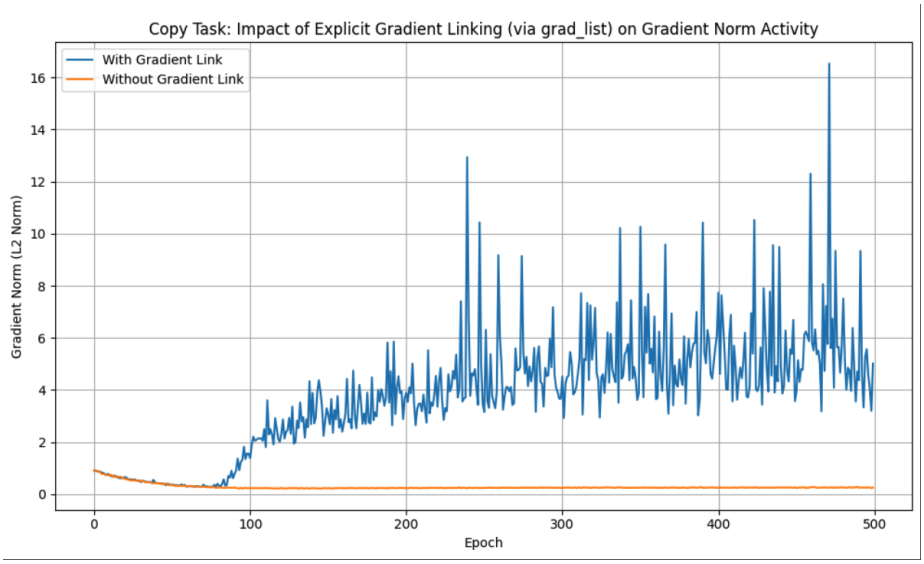

Figure 2: Gradient Norm Activity

6.2 Analysis of Gradient Norm

The figure shows the total norm (magnitude) of read gradients aggregated from `grad_list` in the proposed method. Comparing this figure with the loss curve in Figure 1, a clear correlation was observed between the epochs where the loss begins to decrease rapidly and the activity of the gradients. This suggests that the gradient link effectively transmits **feedback from future reads** to write operations, and that these gradients are directly driving the updates of model parameters, contributing to the reduction in loss.

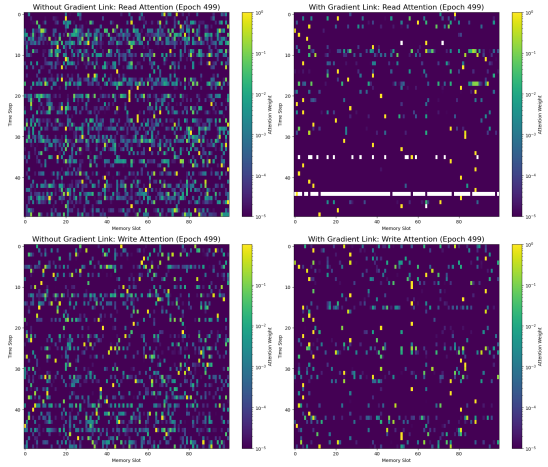

Figure 3: Changes in memory usage patterns (attention heatmap)

6.3 Changes in Memory Utilization Patterns

- Figure 3 shows a comparison of the attention patterns for models both with and without gradient linking. In the model without gradient linking (left side of the figure), both write and read attentions are relatively randomly distributed without concentrating on specific memory locations. This indicates that the model lacks a clear strategy for what information to write where, and from where to read it, which led to learning stagnation.

- In contrast, in the model with the proposed gradient linking enabled (right side of the figure), write attention shows a tendency to concentrate on specific memory locations, and accordingly, read attention also exhibits a concentrated pattern, accurately referring to the written locations. This indicates that the model, guided by gradients from id_tokens, autonomously learned an efficient and coordinated memory utilization strategy: write data for this time step to this memory location, and read from that location at this time step.

- These results underscore that the proposed gradient linking mechanism not only propagates gradients but also has a deeper impact, fostering cooperative learning of the model's internal state and memory strategy. This allows the model to maintain flexibility in data storage while adaptively establishing optimal memory management patterns for task achievement. This ability to maintain "a certain degree of freedom" while achieving high performance is a significant strength of this method, and it can be concluded that it offers a promising solution to the challenges of gradient propagation in existing memory networks.

7 Application of the Proposed Method and Future Prospects

As demonstrated above, the proposed explicit gradient link mechanism has been shown to enhance performance on copying tasks and improve learning quality. This enhancement is evidenced by gradient norm analysis and attention heatmaps. This approach effectively addresses the fundamental challenge of gradient flow in conventional external memory neural networks, thereby facilitating more stable and efficient learning. In this study, the mechanism was applied to a model based on the `AttentionMemoryInterface`, and its effectiveness was demonstrated. In addition, it is conceivable that this approach may be applicable to other memory-augmented neural networks, including DNC[2] and NTM [1]. Although experiments on these models were not conducted in the present study, the potential for performance enhancement can be ascertained through future verification, thereby establishing this as a significant subsequent research undertaking.

7.1 potential for application

At this stage, the efficacy of this mechanism has been demonstrated through its application to MANNs. However, we posit that the core principles of the proposed gradient link can be applied to various types of neural network architectures with temporally or structurally distant dependencies.
For instance, in other recurrent neural network (RNN) variants where long-range dependencies impede learning, or in graph neural networks (GNNs) [5] with intricate multi-step reasoning, explicit gradient links could similarly enhance performance and learning quality when gradient propagation between specific nodes becomes ineffective. This approach is considered applicable to scenarios where **an action or module in one part of the network has a subsequent impact on distant modules or operations.**
Consequently, this research offers a direct solution to the vanishing gradient problem in MANNs and proposes a novel paradigm for learning stabilization in broader deep learning models. It is expected to establish a robust foundation for solving more complex and challenging tasks in the future.

8 Conclusion

This research proposes an innovative solution to the long-standing problem of gradient discontinuity between writing and reading operations in neural networks with external memory: an explicit gradient linking mechanism that uses ID tokens. It is important to note that this research does not design a new memory-augmented neural network (MANN) itself, but rather proposes an auxiliary mechanism to enhance existing ones. Thus, this work's originality lies not in creating a new architecture but in providing a gradient control mechanism that can be retrofitted onto existing models.

8.1 Results

Extensive experiments comparing models that incorporate our proposed method with conventional models that lack gradient links clearly demonstrated the effectiveness of our method. Results from copying tasks showed that this method The proposed method significantly accelerates the initial learning phase, showing a dramatic decrease in loss during the first 150 epochs compared to the baseline (Figure 1) Gradient links effectively reconstruct broken feedback loops, enabling the model to learn memory utilization strategies more efficiently. Analysis of attention heatmaps provided visual evidence of improved learning quality. These heatmaps showed that the gradient link enabled the model to autonomously establish cooperative patterns for write and read positions.

8.2 Future Challenges

The success of this research opens new avenues for learning in memory networks. However, the proposed method presents several challenges. For example, due to its activation of gradients, it may exacerbate gradient explosion, necessitating gradient clipping as a countermeasure. Additionally, the current research artificially constructs gradient links, which limits its applicability to complex tasks. Therefore, future work must focus on developing advanced gradient control mechanisms, such as dynamic learning rate scheduling or adaptive clipping specifically designed for the gradient linking mechanism, to ensure stable and high-speed training across various hyperparameters.This will contribute to the development of more general-purpose AI capable of learning to utilize memory in complex and dynamic environments.

References

[1] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," *arXiv preprint arXiv:1410.5401*, 2014.

[2] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, E. Grefenstette, T. Ramalho, J. Agapiou, A. Specht, *et al.*, "Hybrid computing with a neural turing machine," *Nature*, vol. 538, no. 7626, pp. 471–476, 2016.

[3] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[5] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *International Conference on Learning Representations (ICLR)*, 2017.