An Idea of a Neural Web

Okerew

January 25, 2025

1 Introduction

This document outlines the mathematical formulas used in the provided neural network code and provides a step-by-step explanation of how the neural network operates. Additionally, it includes a detailed explanation of the design principles behind the neural network.

2 Memory Vector Computation

The memory vector is computed by combining neuron states, neuron outputs, and input tensor values. The formula for computing the memory vector is:

$$\text{memory\_vector}[i] = \begin{cases} \text{neurons}[i].\text{state} & \text{if } i < \text{MAX\_NEURONS} \\ \text{neurons}[i - \text{MAX\_NEURONS}].\text{output} & \text{if MAX\_NEURONS} \leq i < 2 \times \text{MAX\_NEURONS} \\ \text{input\_tensor}[i - 2 \times \text{MAX\_NEURONS}] & \text{if } 2 \times \text{MAX\_NEURONS} \leq i < \text{MEMORY\_VECTOR\_SIZ} \\ 0 & \text{otherwise} \end{cases}$$

The memory vector is a concatenation of neuron states, neuron outputs, and input tensor values. If the vector size exceeds the available data, the remaining elements are set to zero. This ensures that the memory vector has a fixed size, which is necessary for consistent processing.

3 Importance Score Calculation

The importance score for a memory vector is calculated as the average of the absolute values of its elements:

$$\text{importance} = \frac{1}{\text{MEMORY\_VECTOR\_SIZE}} \sum_{i=0}^{\text{MEMORY\_VECTOR\_SIZE}-1} |\text{memory\_vector}[i]|$$

The importance score measures the overall significance of a memory vector. Higher absolute values in the vector indicate greater importance. This score is used to determine which memories should be retained, consolidated, or forgotten.

4 Memory Decay

Memories decay over time based on a decay factor. The decay formula is:

$$\text{importance}_{\text{new}} = \text{importance}_{\text{old}} \times \text{DECAY\_FACTOR}$$

The decay factor reduces the importance of memories over time, simulating the natural forgetting process. This ensures that less important memories are gradually phased out, making room for new information.

5 Memory Consolidation

Memories are consolidated based on their importance. If the importance exceeds a threshold, the memory is strengthened:

$$\text{importance}_{\text{new}} = \text{importance}_{\text{old}} \times \text{STRENGTHEN\_FACTOR}$$

Memories with high importance are strengthened, making them more likely to be retained in long-term memory. This process mimics how the brain prioritizes and retains important information.

6 Neuron State Update

The state of a neuron is updated based on its current state, weighted inputs, and input tensor values:

$$\text{state}_{\text{new}} = \text{state}_{\text{old}} \times 0.7 + \text{weighted\_output} \times 0.2 + \text{input\_tensor}[i\%\text{INPUT\_SIZE}] \times 0.1$$

The new state of a neuron is a weighted combination of its previous state, the weighted sum of its inputs, and the influence of the input tensor. This ensures that the neuron's state reflects both its history and the current input.

7 Neuron Output Calculation

The output of a neuron is calculated using the hyperbolic tangent (tanh) function:

$$\text{output} = \tanh(\text{state} \times \text{scale} + \text{bias})$$

The tanh function is used to introduce non-linearity into the neuron's output, ensuring that the output is bounded between -1 and 1. This non-linearity is essential for the network to model complex relationships in the data.

8 Cosine Similarity

The cosine similarity between two vectors is calculated as:

$$\text{similarity} = \frac{\sum_{i=0}^{n-1} \text{vec1}[i] \times \text{vec2}[i]}{\sqrt{\sum_{i=0}^{n-1} \text{vec1}[i]^2} \times \sqrt{\sum_{i=0}^{n-1} \text{vec2}[i]^2}}$$

Cosine similarity measures the cosine of the angle between two vectors, providing a value between -1 and 1. A value of 1 indicates perfect similarity. This metric is used to compare memory vectors and identify similar patterns.

9 Dynamic Parameter Update

Dynamic parameters are updated based on performance and stability:

$$\text{current\_adaptation\_rate} = \text{learning\_momentum} \times \text{current\_adaptation\_rate} + (1 - \text{learning\_momentum}) \times \text{target\_rate}$$

The adaptation rate is adjusted using momentum, ensuring smooth updates to the network's parameters based on performance trends. This dynamic adjustment allows the network to adapt to changing conditions and improve over time.

10 Hierarchical Memory System

The neural network employs a hierarchical memory system consisting of short-term, medium-term, and long-term memory clusters. This design allows the network to prioritize and retain important information while discarding less relevant data.

10.1 Short-Term Memory

Short-term memory stores recent memories with high detail. Memories in this cluster decay quickly unless they are consolidated into medium-term memory.

10.2 Medium-Term Memory

Medium-term memory stores consolidated memories with moderate detail. Memories in this cluster are less detailed than those in short-term memory but are more stable and longer-lasting.

10.3 Long-Term Memory

Long-term memory stores highly consolidated, abstract memories. Memories in this cluster are the most stable and are retained for the longest duration.

11 Predictive Coding

Predictive coding is used to enhance the network's ability to anticipate future states based on current inputs and past experiences. The predictive coding formula is:

$$\text{prediction\_error} = \text{actual\_input} - \text{predicted\_input}$$

The network adjusts its weights and states based on the prediction error, allowing it to improve its predictive accuracy over time.

12 Advanced Neuron Management

The network employs advanced neuron management techniques to dynamically add or remove neurons based on their performance. Neurons that consistently underperform are removed, while new neurons are added to explore new patterns and improve network performance.

13 Self-Reflection System

The self-reflection system is designed to analyze the network's performance and coherence. It includes functions to detect confabulation, regenerate responses, and perform self-reflection. The main components are:

13.1 Reflection History

The reflection history tracks confidence, coherence, and consistency thresholds. It initializes with default values and updates based on the network's performance.

13.2 Reflection Metrics

Reflection metrics include coherence score, confidence score, novelty score, consistency score, and reasoning. These metrics are used to evaluate the network's performance and detect potential confabulation.

13.3 Self-Reflection Integration

The self-reflection system is integrated into the main loop, where it performs self-reflection at regular intervals and adjusts parameters based on the reflection metrics.

14 Self-Identity System

The self-identity system manages the network's identity components, including core values, beliefs, and identity markers. It updates these components based on behavioral patterns and experiences.

14.1 Identity Components

Identity components include core values, belief systems, identity markers, experience history, and behavioral patterns. These components are updated based on the network's state and experiences.

14.2 Identity Reflection

The identity reflection provides insights into the network's consistency score, confidence level, core value stability, belief system coherence, identity marker prominence, and temporal coherence.

15 Knowledge Filter System

The knowledge filter system categorizes inputs and records problem instances. It analyzes category statistics and integrates knowledge based on the network's state and input tensor.

15.1 Knowledge Categories

Knowledge categories are initialized with default values and updated based on usage statistics and confidence scores. The system records problem instances and analyzes category statistics to update category importance and confidence.

15.2 Knowledge Integration

The knowledge filter system is integrated with the memory system to strengthen memories based on their importance and update category statistics based on the network's performance.

16 Design of the Neural Network

The neural network is designed with several key principles in mind:

1. **Hierarchical Memory System**: The network uses a hierarchical memory system with short-term, medium-term, and long-term memory clusters. This design allows the network to prioritize and retain important information while discarding less relevant data.

2. **Dynamic Adaptation**: The network dynamically adjusts its parameters based on performance and stability. This ensures that the network can adapt to new data and changing conditions, improving its robustness and accuracy.

3. **Non-linear Activation**: The use of the tanh function introduces non-linearity into the neuron outputs, enabling the network to model complex relationships in the data.

4. **Memory Consolidation and Decay**: Memories are consolidated based on their importance and decay over time. This mimics the natural process of memory retention and forgetting, ensuring that the network remains efficient and focused on relevant information.

5. **Performance Optimization**: The network continuously monitors its performance and optimizes its parameters to achieve the best results. This includes adjusting learning rates, batch sizes, and other hyperparameters.

6. **Pattern Matching and Similarity**: The network uses cosine similarity to compare memory vectors and identify similar patterns. This allows the network to recognize and recall relevant information from its memory.

7. **Predictive Coding**: The network employs predictive coding to anticipate future states and improve its predictive accuracy.

8. **Advanced Neuron Management**: The network dynamically manages its neurons, adding or removing them based on their performance to maintain optimal network efficiency.

9. **Meta-Controller**: The network uses a meta-controller to manage the importance and learning efficiency of different regions, enhancing overall learning and adaptation.

10. **Context Management**: The network organizes and updates context nodes to maintain a coherent global context, adapting to environmental factors and constraints.

11. **Self-Reflection**: The network performs self-reflection to analyze its performance and coherence, detect potential confabulation, and regenerate responses.

12. **Self-Identity**: The network manages its identity components based on behavioral patterns and experiences, providing insights into its consistency and confidence.

13. **Knowledge Filtering**: The network categorizes inputs and records problem instances, analyzing category statistics to update category importance and confidence.

17 Step-by-Step Explanation of the Neural Network

The neural network operates in the following steps:

1. **Initialization**: The network is initialized with neurons, connections, weights, and a memory system. This sets up the basic structure of the network and prepares it for processing data.

2. **Input Processing**: The input tensor is generated based on the current step and text input. The memory vector is computed by combining neuron states, outputs, and input tensor values. This ensures that the network has a complete representation of the current state.

3. **Memory Management**: The importance score for the memory vector is calculated. Memories are decayed over time using a decay factor, and memories with high importance are consolidated into clusters. This ensures that the network retains important information while discarding less relevant data.

4. **Neuron State Update**: Neuron states are updated based on their current state, weighted inputs, and input tensor values. Neuron outputs are calculated using the tanh function. This ensures that the network's neurons reflect both their history and the current input weights are also updated at this time.

5. **Memory Integration**: Relevant memories are retrieved based on their importance. Memory vectors are merged if they are similar. This allows the network to integrate new information with existing knowledge.

6. **Performance Measurement**: Performance metrics such as execution time, average output, and error rate are calculated. Parameters are optimized based on performance history. This ensures that the network continuously improves over time.

7. **Dynamic Adaptation**: Dynamic parameters are updated based on performance and stability. The network is adapted using the updated parameters. This ensures that the network can adapt to changing conditions and improve its performance.

8. **Output Generation**: Neuron outputs are transformed into text. Performance analysis and graph generation are performed periodically. This provides insights into the network's performance and helps identify areas for improvement.

9. **Saving State**: The network state, memory system, and parameters are saved. This ensures that the network can resume operation from its current state in future sessions.

10. **Meta-Controller Adaptations**: The meta-controller updates region priorities based on performance metrics and adapts the network's weights accordingly.

11. **Context Updates**: The context management system updates context nodes based on the network's state and environmental factors, maintaining a coherent global context.

12. **Self-Reflection**: The network performs self-reflection to analyze its performance and coherence, detect potential confabulation, and regenerate responses.

13. **Identity Update**: The network updates its identity components based on behavioral patterns and experiences, providing insights into its consistency and confidence.

14. **Knowledge Integration**: The network categorizes inputs and records problem instances, analyzing category statistics to update category importance and confidence.

These steps ensure that the neural network operates efficiently, adapts to changing conditions, and produces accurate and relevant results.

18 Conclusion

To conclude from the tests I have run, this kind of architecture is more efficient than industry standards, can adapt faster, performs better in patterns than any model currently available, is more cost-effective, and more importantly, it can be perhaps a step further to achieving AGI.