Research Article

Contextual Learning for Interruption Tolerance and Multitasking

Thomas Portegys[1]

1. Dialectek, United States

Natural environments abound in event streams that require interruption tolerance and multitasking. This project introduces Mandala, a neural network that improves multitasking in the form of dealing with intervening events from overlaid causation chains, a capability that a conventional recurrent artificial neural network (RNN) struggles with, as evidenced by the results. This capability is also tolerant of interrupting events. Mandala achieves this by accumulating contextual tiers of temporal states that are fed into a multilayer perceptron (MLP) at each time step. Mandala is also an effort to combine the Morphognosis and Mona neural network models into a comprehensive model for learning and behavior. Mona features a contextual causation learning with goal-directed motivation. Morphognosis features contextual MLP learning. In addition, accumulating temporal information discretely labels hierarchical cause-and-effect relationships that can be used for augmented processing. In the case of Mona, channeling motivation through the network for the purpose of goal-seeking requires this feature.

Corresponding author: Thomas E. Portegys, portegys@gmail.com

Introduction

A mated pair of eagles, Jackie and Shadow, are popular on the internet these days as viewed with a hidden camera and microphone[1]. They spend their days and nights building and repairing their nest, brooding eggs, hunting, and fending off threats. In one segment Shadow is seen screaming for Jackie to help with intruding ravens that have in the past destroyed their eggs. He knows he must remain sitting atop the eggs. Jackie has been gone overlong and he must be hungry. Such chaotic conditions are frequent for many animals and the neural networks that orchestrate them: they must be able to multitask in the midst of interruptions and interferences.

Artificial neural networks (ANNs) currently deal with what might be termed controlled surprise, defined as being able to formulate an output before dealing with the next input. Large Language Models (LLMs)[2] can

sensibly respond to just about any query, but while composing said responses they do not deal with a crying baby or a flat tire.

This project is an effort to equip an ANN to be more tolerant of interruptions and accomplish multitasking more effectively. The resulting architecture, named Mandala, achieves this by accumulating contextual tiers of temporal states that are fed into a multilayer perceptron (MLP) at each time step. Mandala is named after a geometric pattern that is typically composed of concentric rings or tiers as shown in Figure 1.

Figure 1. Mandala

Mandala is also a step toward synthesizing the Morphognosis[3] and Mona [4][5] neural network models into a comprehensive model for learning and behavior. Mona features a contextual causation learning with goal-directed motivation. Morphognosis features contextual MLP learning. In addition, accumulating non-hidden temporal information discretely labels hierarchical cause-and-effect relationships that can be used for augmented processing. In the case of Mona, channeling motivation through the network for the purpose of goal-seeking requires this feature.

Architecturally Mandala bears a resemblance to a Jordan recurrent neural network[6], as shown in Figure 2.

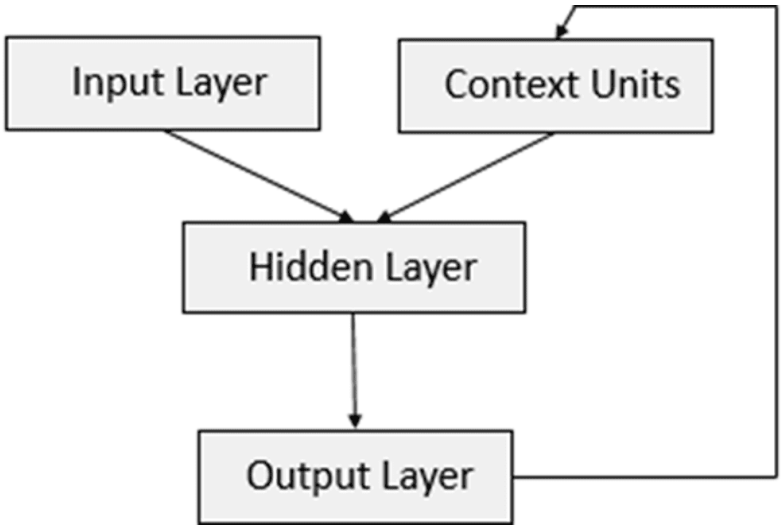

Figure 2. Jordan recurrent neural network

In a Jordan network, the output layer is fed into context units, which in turn are fed into the hidden layer along with the input layer. In Mandala, the output layer produces not only a prediction, but also temporal state to combine with external input.

Mandala also shares some commonalities with networks that make use of functionality external to the core network. Some of these are discussed below.

Recurrent Neural Networks (RNNs) with external temporal memory are architectures designed to overcome the limitations of standard RNNs (like LSTMs or GRUs) in memorizing long-term dependencies. While standard RNNs rely on internal hidden states that often suffer from vanishing gradients over long sequences, these models integrate a separate, addressable memory bank to store and retrieve past information over extended time horizons.

Neural Turing Machines (NTMs)[7]. These are models that couples a neural network controller to an external memory bank, capable of learning to manipulate and manage the memory to solve tasks like sorting or copying.

A Dynamic Memory Network (DMN)[8] is a neural network architecture that uses an external episodic memory to answer questions about input sequences (stories, text).

External Addressable Long-Term and Working Memory (EALWM)[9] splits external memory into distinct long-term and working memory parts, improving the ability to learn long-term dependencies without gradient issues.

The above models retain an RNN at their core, using external storage as an adjunct memory. In Mandala context input and output serve as the repository of temporal state for a non-recurrent MLP.

Description

Problem Generation

Training and testing event streams are generated using randomly generated grammars that produce binary hierarchies of cause-and-effect pairs of terminals. A simple such hierarchy is presented in Figure 3.

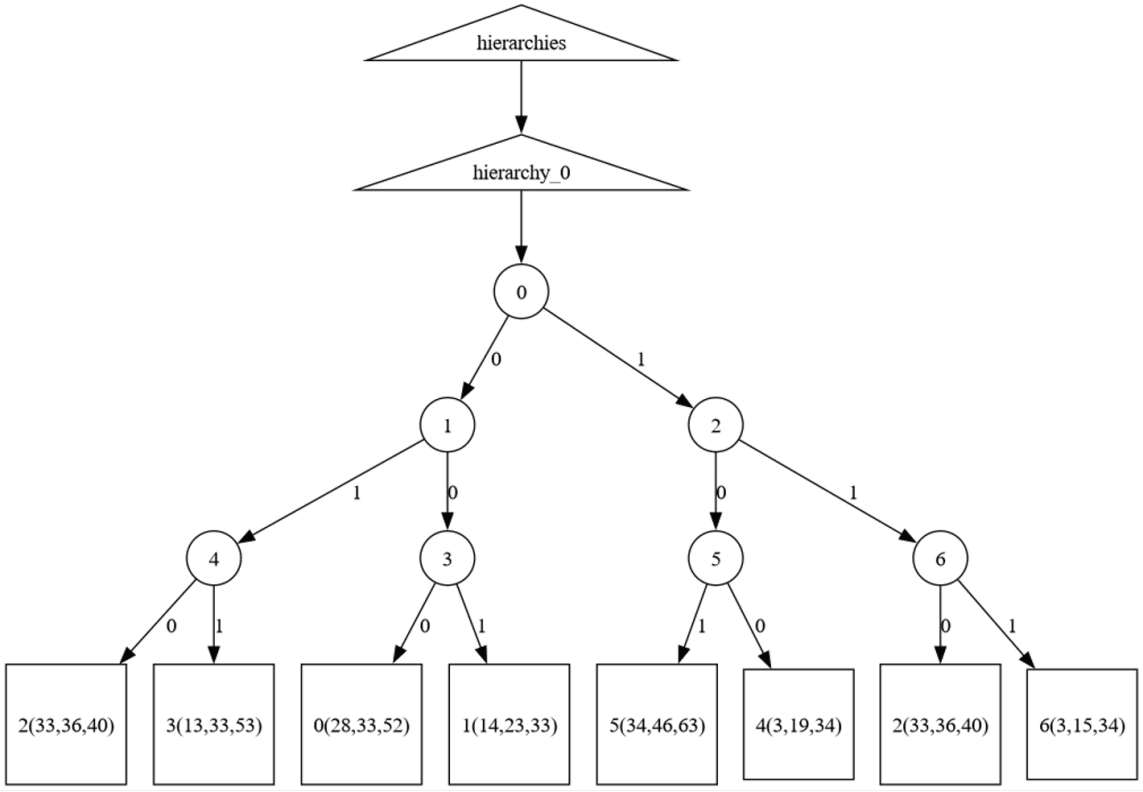

Figure 3. Simple generated hierarchy

Here there are seven non-terminals (0-6), indicated by circles, and seven terminals, indicated by squares. The numbers on the edges indicate the child order. The parenthesized numbers in the terminals are sparse encodings for the neural network.

This hierarchy produces a path of: {0, 1, 2, 3, 4, 5, 2, 6}.

The terminal children of a non-terminal are considered to be a cause-and-effect pair. The goal of the neural network is to predict the effect given the cause as they are presented in the input stream.

Generating inputs using this method presents context-related challenges. For example, the appearance of cause 2 cannot solely be used to determine which of the effects 3 or 6 is correct. The correct prediction depends on the context provided by the previous cause-and-effect pair.

Parameters determine the number and shapes of hierarchies:

- NUM_CAUSATION_HIERARCHIES
- NUM_NONTERMINALS
- NUM_TERMINALS
- TERMINAL_PRODUCTION_PROBABILITY

Multiple paths are generated by multiple hierarchies. Node ids are scoped by hierarchy, so the same id in different hierarchies will be unique to each hierarchy. The terminal production probability affects the hierarchy height: a greater probability causes non-terminals to be more likely to produce terminal children, thus stunting the hierarchy depth.

An example of two hierarchies having identical structure is presented in Figure 4.

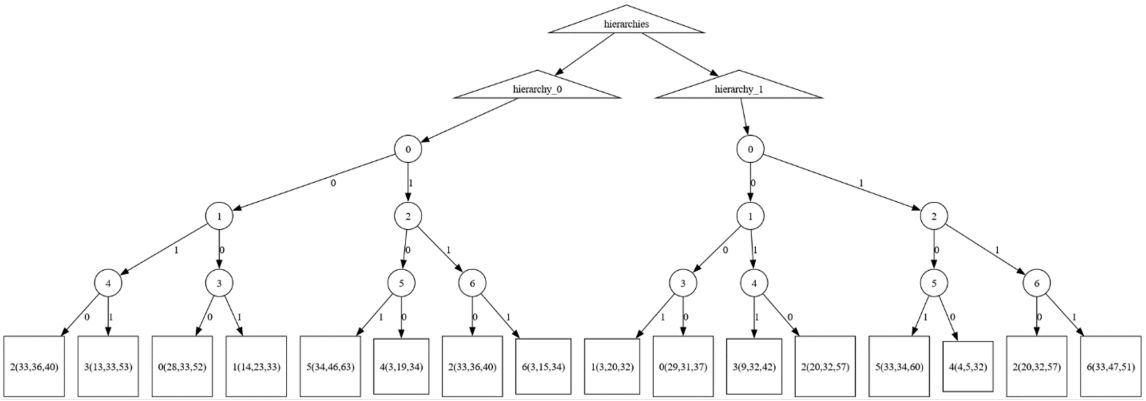

Figure 4. Two hierarchies

Note that a cause terminal is itself not caused, which means that temporal gaps containing interstitial events can precede it, presenting another learning challenge. For example, with interstitial events the stream given by the hierarchy in Figure 3 might look like:

{a, b, 0, 1, b, 2, 3, b, c, 4, 5, 2, 6}

Where the letters a-c represent interstitial events.

A final challenge, addressing multitasking, can be shown by interleaving the streams from the hierarchies in Figure 4:

doi.org/10.32388/ZNZO4Q

{0/0, 1/0, 2/0, 3/0, 0/1, 1/1, 4/0, 5/0, 2/1, 3/1, 2/0, 6/0, 4/1, 5/1, 2/1, 6/1}

Where the hierarchy number is appended to the terminal id after a slash (“/”) character.

Algorithm

The generated terminal ids are encoded into a sparse representation of inputs to the neural network. Sparse encoding is also used to represent higher tiered information. The advantage of sparsity in neural networks is delineated in the Numenta White Paper[10].

During training, paths are presented to the network without interstitial or interleaved events. Those are supplied during testing. This is how training is done:

The input to the network at time *t* is denoted X_t . The output is y_t . Concatenated to the external input and prediction output, there are *n* tiers of encoded temporal states.

The tier values for y_t are computed as follows:

```
update_y_tiers:
    tier_A = encode(external input)
    for tier i from 0 to n:
       tier_B = Xt values of tier i
       if tier_B exists:
          negate values for tier_B in tier i
          tier_A = encode(tier_A, tier_B)
       else:
          enter values for tier_A in tier i
          break
```

The tier values of $X_{t+1} = X_t + y_t$

The network is thus trained to compute temporal context information. Note that the *y* values are a delta for the *X* values of the current hierarchy, leaving the *X* values generated by other concurrent hierarchies to propagate unmodified.

Here is an example scenario, where the numeric triples are encodings:

```
t = 0, X: [sense=[4, 14, 41]], y: [sense=[21, 24, 40], tier0=[4,
14, 41]]

t = 1, X: [sense=[21, 24, 40], tier0=[4, 14, 41]], y: [sense=[5,
42, 43], tier0=[-4, -14, -41], tier1=[8, 9, 34]]

t = 2, X: [sense=[5, 42, 43], tier0=[], tier1=[8, 9, 34]], y:
[sense=[38, 40, 42], tier0=[5, 42, 43]]
```

The algorithm also times-out values to limit their “range”. Values in higher tiers, having longer histories, persist longer.

For testing, novel interstitial events and interleaved events are presented to the network to evaluate resilience to interruptions and multitasking performance. Prediction outputs are used to update the context tiers.

Note that the algorithm could also be used to recognize out-of-order streams, a topic explored by Portegys[11].

The code is available on GitHub at: https://github.com/morphognosis/Mandala

Results

Mandala/RNN Comparisons

Mandala was evaluated against an LSTM (Long Short-Term Memory) RNN. The Keras 3.11.3 machine learning library was used.

Mandala neural network parameters:

- Three hidden layers of 128 neurons.
- Tanh activation.
- Mean squared error loss.
- Adam optimizer.

RNN parameters:

- One hidden layer of 128 neurons.
- Mean squared error loss.
- Adam optimizer.

Parameter combinations are as follows:

- NUM_CAUSATION_HIERARCHIES: 1, 2, 3.
- NUM_NONTERMINALS: 10, 15, 20.

- NUM_TERMINALS: 10, 15, 20.
- TERMINAL_PRODUCTION_PROBABILITY: .25, .5, .75

Training for 500 epochs. Average of 10 runs per parameter combination. An error was counted when the effect terminal event was not correctly predicted by its cause terminal event.

To get a baseline comparison, Mandala and the RNN were run with no interstitial or interleaved events. Figure 5 shows nearly perfect performance for both.

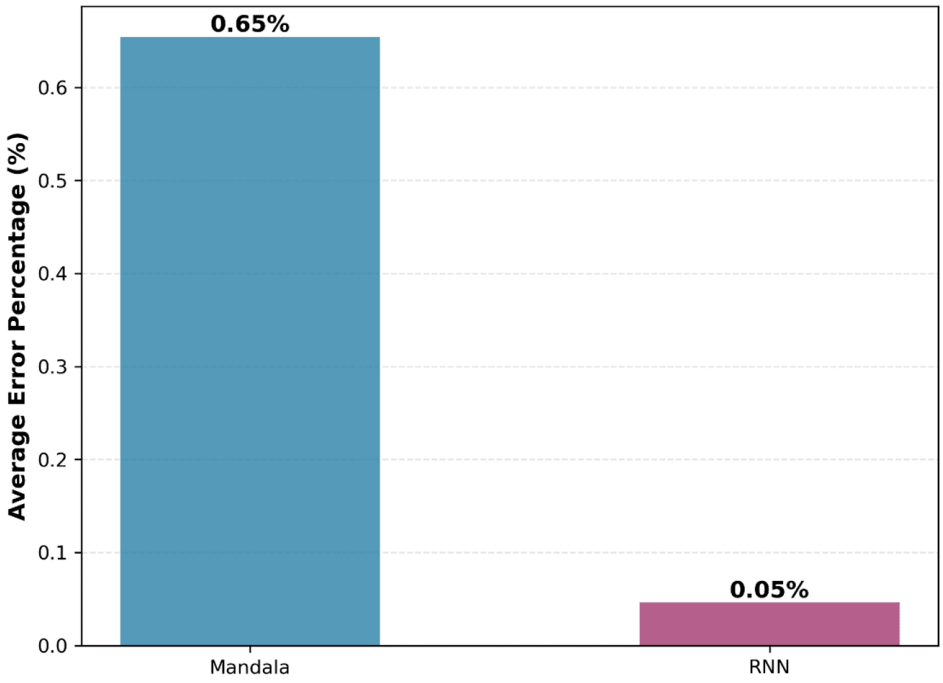

Figure 5. Baseline testing performance: no interstitial or interleaved events

The remainder of the tests were done with interstitial events inserted into the stream, randomly chosen from 10 possible patterns.

doi.org/10.32388/ZNZO4Q

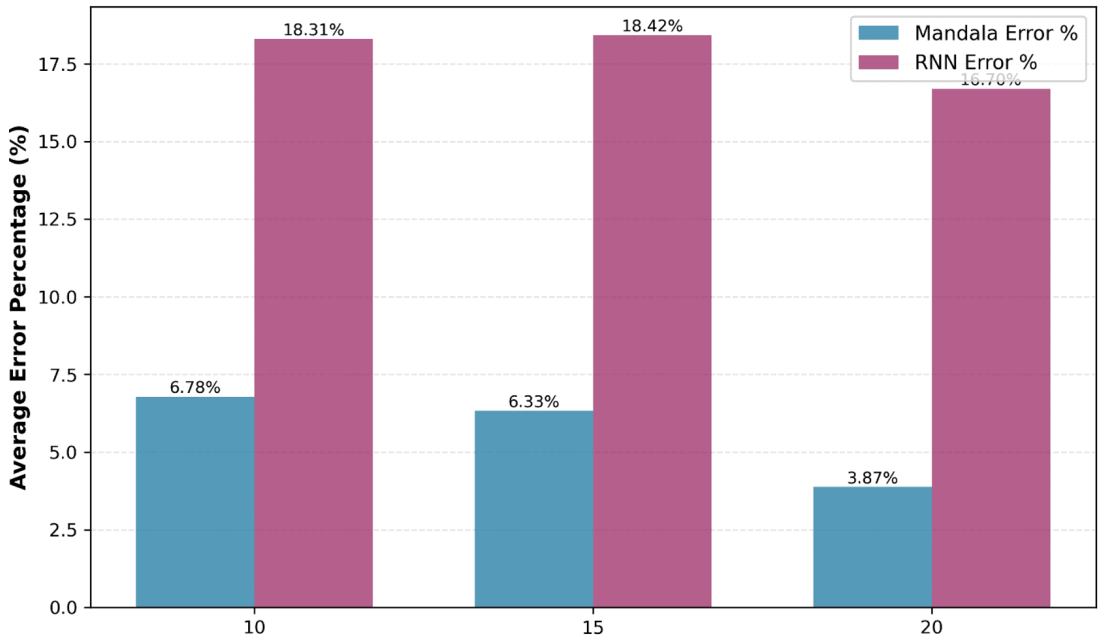

Figure 6. Number of terminals

Figure 6 shows the results of varying the number of possible terminals.

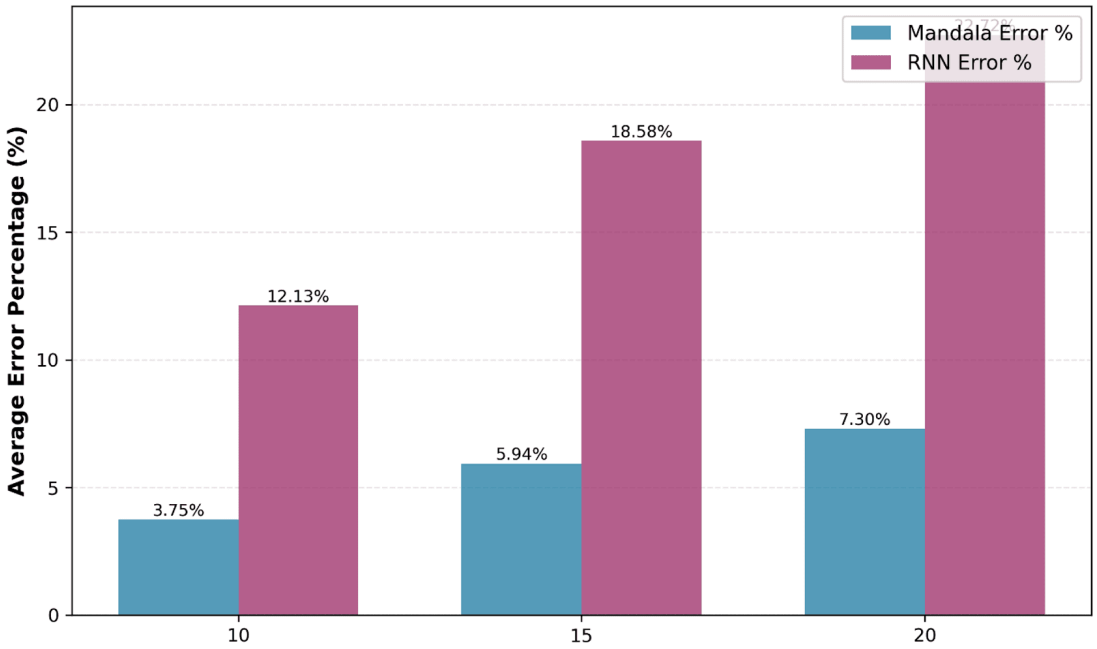

Figure 7. Number of non-terminals

doi.org/10.32388/ZNZO4Q

Figure 7 varies the number of non-terminals, which appears to indicate a linear-like increase in the error rate.

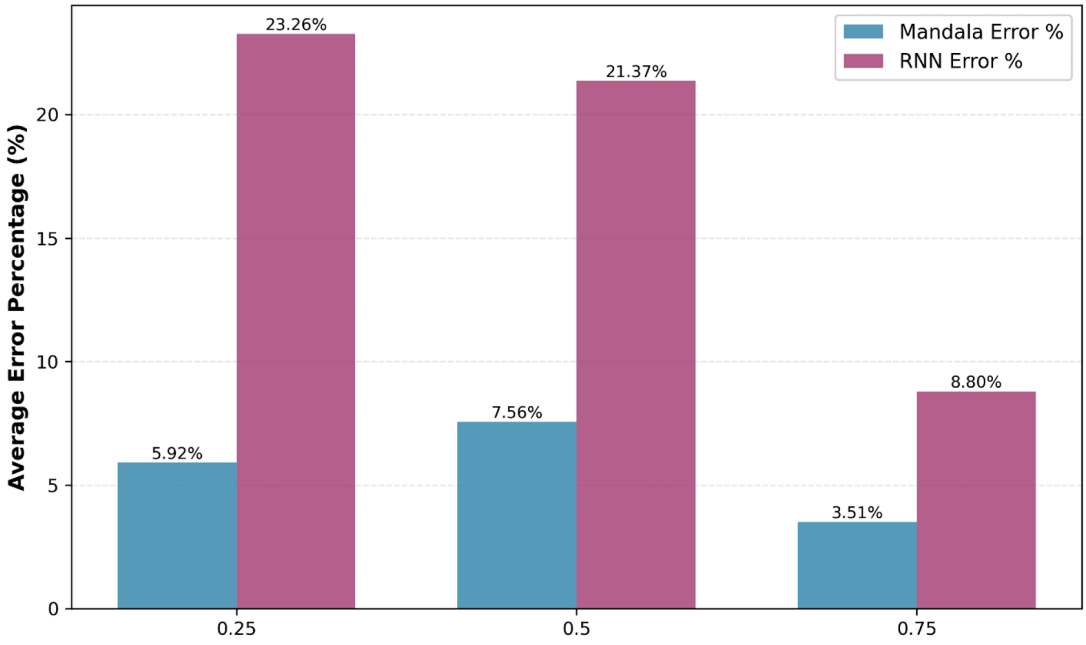

Figure 8. Terminal production probability

Figure 8 shows how making the hierarchies shallower by increasing the probability of terminating branches reduces the error rate for both Mandala and RNN.

doi.org/10.32388/ZNZO4Q

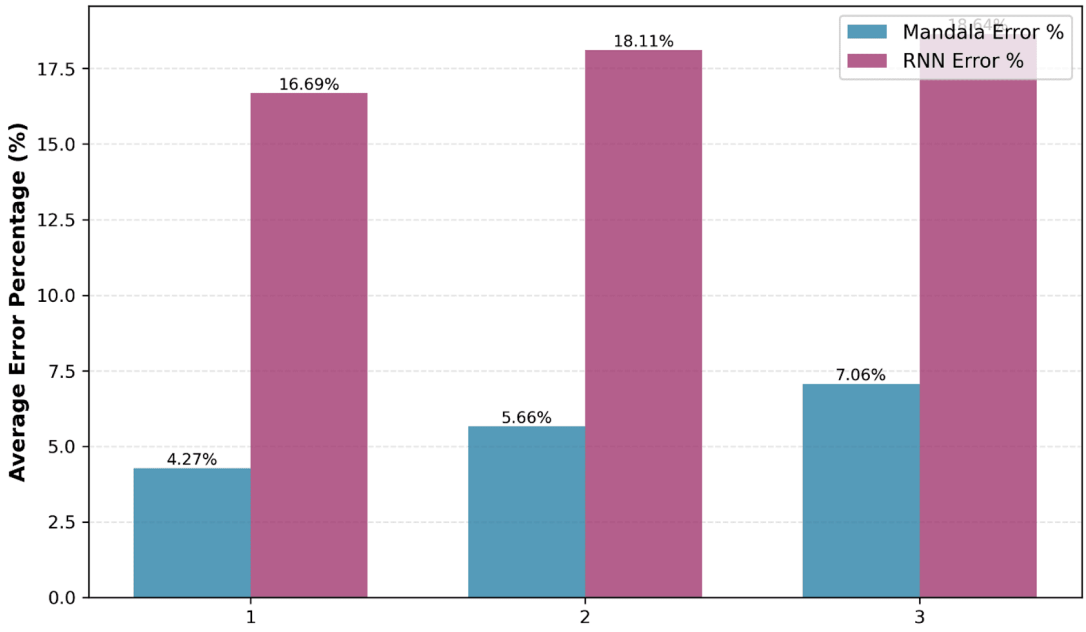

Figure 9. Causation hierarchies

Figure 9 demonstrates how multitasking, which interleaves events, affects performance. The error rate for Mandala increases as the number of tasks goes from 1 to 3. RNN shows worse performance for multitasking in general.

doi.org/10.32388/ZNZO4Q

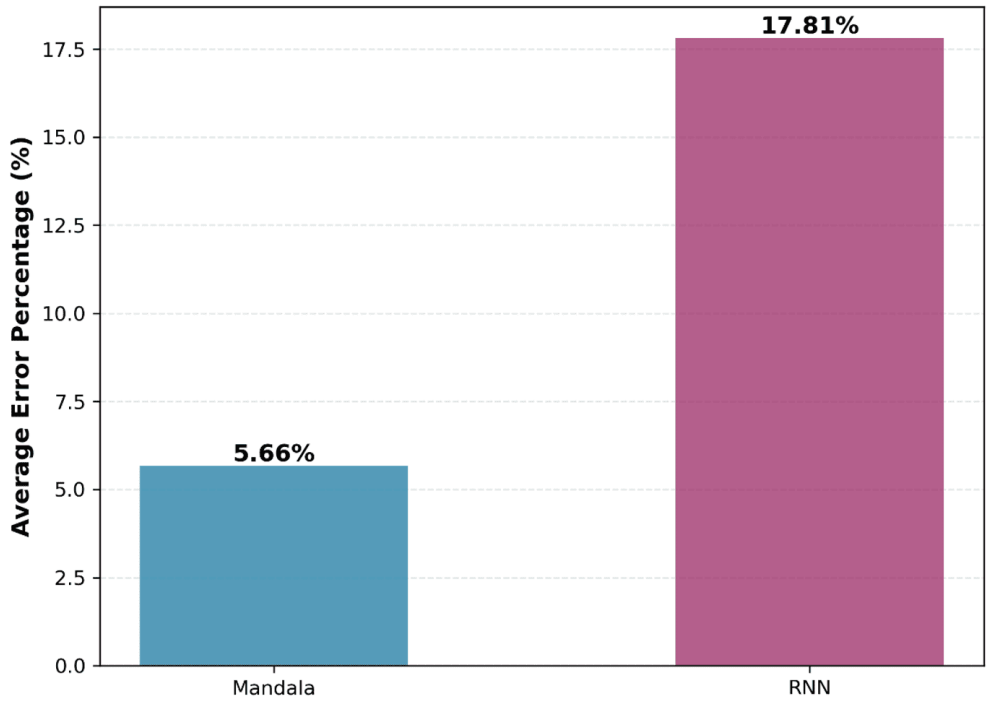

Figure 10. Overall error rates

Figure 10 shows the overall error rate comparison across parameter combinations.

LLM Testing

As an attempt to observe how an LLM performs on this type of problem, the sequences depicted in Figure 3 were posed to Claude[12], a popular LLM.

Prompt:

Given these sequences:

0,1,2,3

4,5,2,6

Predict the next number in these sequences:

8,8,7,0

7,11,4

Claude correct predicts 1 and 5 respectively.

Prompt:

Given these sequences:

0,1,2,3

4,5,2,6

Predict the next number in these sequences:

7,4,5,11,11,8,10,10,10,11,2

0,1,9,11,10,2

7,9,11,9,7,8,4,5,7,10,10,8,7,2

7,0,1,11,10,7,7,10,2

4,5,10,8,8,2

11,8,0,1,8,10,2

11,7,7,11,11,0,1,11,7,11,11,9,11,10,7,2

10,8,9,7,4,5,9,8,8,10,11,11,7,9,9,11,2

Note that 3 is the correct prediction if 4,5 is in the sequence, and 6 if 0,1 is in the sequence.

Claude ran a number of analytics, but failed to predict any of the sequences.

Conclusion

Mandala suggests that external hierarchical temporal state accumulation enables neural networks to handle interruptions and multitasking more effectively than a conventional RNN as evidenced by a 68% performance improvement over an LSTM (5.66% vs. 17.81% error rate).

By maintaining separable contextual tiers, Mandala preserves information about concurrent causation chains that might otherwise interfere in standard recurrent architectures. This approach offers both predictive capability and explicit causation labeling, making it particularly suitable for goal-directed systems like Mona that require identified causal relationships.

A likely candidate for future work would be to retrofit the honey bee foraging system presented in Portegys[3] to use Mandala.

References

1. ^*Forgione M (2020). "How a Pair of Bald Eagles Became Southern California Rock Stars." Los Angeles Times.*
2. ^*Yin S, Fu C, Zhao S, Li K, Sun X, Xu T, Chen E (2024). "A Survey on Multimodal Large Language Models." Nat Sci Rev. **11**(12):nwae403. doi:10.1093/nsr/nwae403.*

3. a, b *Portegys T (2022). "Morphognostic Honey Bees Communicating Nectar Location Through Dance Movements." Nature and Biologically Inspired Computing (NaBIC 2022).*
4. ^ *Portegys T (2010). "A Maze Learning Comparison of Elman, Long Short-Term Memory, and Mona Neural Networks." Neural Netw.*
5. ^ *Portegys T (2024). "Nest-building Using Place Cells for Spatial Navigation in an Artificial Neural Network." ICAL RAI 2024: International Conference on Artificial Life, Robotics and Artificial Intelligence.*
6. ^ *Jordan MI (1986). "Serial Order: A Parallel Distributed Processing Approach. Tech. rep. ICS 8604." Institute for Cognitive Science, University of California.*
7. ^ *Graves A, Wayne G, Danihelka I (2014). "Neural Turing Machines." arXiv. https://arxiv.org/abs/1410.5401.*
8. ^ *Kumar A, Irsoy O, Ondruska P, Iyyer M, Bradbury J, Gulrajani I, Zhong V, Paulus R, Socher R (2016). "Ask Me Anything: Dynamic Memory Networks for Natural Language Processing." Proc 33rd Int Conf Mach Learn. **48**:1378–1387.*
9. ^ *Quan Z, Zeng W, Li X, Liu Y, Yu Y, Yang W (2020). "Recurrent Neural Networks With External Addressable Long-Term and Working Memory for Learning Long-Term Dependences." IEEE Trans Neural Netw Learn Syst. **31**(3):813–826. doi:10.1109/TNNLS.2019.2910302.*
10. ^ *Numenta (2021). "Sparsity Enables 100x Performance Acceleration in Deep Learning Networks, A Technology Demonstration." Numenta. https://www.numenta.com/assets/pdf/research-publications/papers/Sparsity-Enables-100x-Performance-Acceleration-Deep-Learning-Networks.pdf.*
11. ^ *Portegys T (2024). "Learning Causation Event Conjunction Sequences." J Artif Intell Auton Intell.*
12. ^ *Anthropic (2023). "Introducing Claude." Anthropic. https://www.anthropic.com/news/introducing-claude.*

Declarations

Funding: No specific funding was received for this work.

Potential competing interests: No potential competing interests to declare.