# Exploring Counting Abilities in Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) Networks

Balaji Vijayaraj
Data Science
Dalarna University
Borlänge, Sweden
v22balvi@du.se

Charu Bisht
Data Science
Dalarna University
Borlänge, Sweden
h21chabi@du.se

*Abstract*—**This study focuses on evaluating and comparing the counting abilities of RNNs and LSTMs. The task involves predicting the number of consecutive occurrences of 'a' before encountering a 'b' in input strings (e.g., aaaaabb). Utilizing a series of experiments under various levels of task complexity, we analyze the correlation between the networks' hidden states and the actual count of 'a's to gain insights into their counting mechanisms. The results indicate that while RNNs exhibit inconsistent performance with a wide range of values in their hidden states, LSTMs consistently demonstrate superior ability to maintain accurate counts over extended sequences, proving to be more reliable for language processing tasks that require understanding of context and sequences. It is noteworthy to mention that these results are surprising, particularly in the case of RNNs. The unexpected observation of significant correlation in RNNs contradicts initial expectations, adding a layer of intrigue to their performance characteristics.**

***Keywords: Recurrent Neural Networks (RNNs), Long Short-Term Memory networks (LSTMs) Counting Abilities, Sequential Data Analysis.***

## I. Introduction

The field of sequential data analysis has experienced notable advancements in recent times, primarily influenced by developments in machine learning algorithms (Mikolov et al., 2013). Central to the effective processing of sequential data is the capability of accurately counting and tracking occurrences within sequences. This skill is particularly crucial in neural networks like Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks (Hochreiter & Schmidhuber, 1997). Their proficiency in correctly identifying and counting elements within data sequences plays a vital role in enhancing their overall efficiency and reliability in a variety of sequence-dependent tasks. The improved handling of sequential data by these networks is not just beneficial for language-based applications, but also for a broader range of applications where sequence recognition and memory are essential.

This study is inspired by previous work by Rodriguez, Wiles, and Elman (1999), which showed the potential of RNNs in learning to count. This work laid a foundation for understanding how neural networks function with sequential data. The paper explores the counting mechanism in RNNs by capturing movement in phase space which is generated by conducting PCA of the time series values of the 12 hidden states of RNNs. They however abstained from investigating correlations within RNN states based on their assumption rested on the belief that RNN states would not correlate, given their representation within a higher-dimensional space shaped by the neural network's state vector.

They refrained from investigating correlations within RNN states, guided by the assumption that RNN states would not exhibit correlations. This assumption was rooted in the understanding that RNN states are represented within a higher-dimensional space defined by the neural network's state vector. Motivated by this, we aim to delve into the counting abilities of both RNNs and LSTMs, specifically examining the correlation perspective. The primary focus lies in assessing how effectively these networks can anticipate the count of a specific pattern in each input string. The experiment is designed to analyze this research question under different levels of task complexity, evaluating the performance of RNNs and LSTMs using different neural network architectures.

The analysis investigates the correlation between the hidden states of these neural networks and the actual count of 'a's in the input sequences, aiming to understand how each model learns and processes sequential information. By examining the correlation trends across different hidden states, this experiment hopes to provide a clearer understanding of how RNNs and LSTMs handle counting tasks. LSTMs, by their construction are expected to excel at counting due to their utilization of an additional cell state model, enabling the isolation of information in these cell states. The expectation is that multiple cell states in LSTMs should correlate with the actual count, reflecting the capacity of LSTMs to capture nuanced counting patterns. Contrastingly, the hypothesis for RNNs posits that correlations should not occur during counting. This assumption is attributed to the nature of RNNs, where nodes change values independently, making it challenging to track trajectories.

This research endeavor aims to bridge the existing knowledge gap by meticulously examining the counting mechanisms inherent in Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTMs). We establish an initial baseline to identify differences in the counting mechanisms of RNNs and LSTMs. The insights gained aim to contribute to the broader development of sequential data analysis and artificial intelligence technologies. This foundational understanding could optimize diverse AI applications beyond language processing.

## II. Literature Review

In 1999, Rodriguez, Wiles, and Elman wrote a paper titled "A Recurrent Neural Network that Learns to Count" (Rodriguez, Wiles, & Elman, 1999). This paper showed that RNNs can learn to count when trained properly. The authors demonstrated that counting is a basic but crucial task for RNNs, especially when dealing with sequential data. This work is one of the early attempts to show how RNNs can learn counting tasks, which is important for understanding how they process and predict sequential data. Their paper involves a detailed exploration of the dynamics within phase space, a construct derived from the application of Principal Component Analysis (PCA) to the time series values of the 12 hidden states within Recurrent Neural Networks (RNNs). To unravel the counting mechanism embedded in RNNs, it is essential to comprehend the intricacies of PCA.

In PCA, the states that contribute to the counting process are initially represented in a higher-dimensional space, constructed from the state vector of the neural network. Contrary to the notion that a single value or dimension accounts for counting, the phenomenon unfolds across a multidimensional vector space. Each dimension in this space captures unique aspects of the counting process, and their combined influence shapes the overall counting behavior of the neural network. PCA assumes a pivotal role in this process. It serves to discern and retain the most significant dimensions, discarding those of lesser importance. By effectively condensing the information into a two-dimensional representation, PCA facilitates the visualization of the counting process within phase space. Interpreting the phase space provides insights into the trajectory of the 12 hidden nodes over time. It acts as a visual guide, revealing the intricate patterns and movements that signify how the counting unfolds within the neural network.

In a different study, Kvam and Hintze (2018) explored decision-making strategies in their paper, "Rewards, Risks, and Reaching the Right Strategy: Evolutionary Paths from Heuristics to Optimal Decisions." Although this paper did not directly study counting abilities in neural networks, it discussed how decision-making strategies evolve. By looking at how simpler heuristic strategies can evolve into more optimal ones, this paper provides some insight into how counting mechanisms in neural networks might also evolve. This work helps to understand decision-making and counting in RNNs and LSTMs, especially when dealing with complex tasks.

The two works reviewed here provide a starting point for studying the counting abilities of neural networks. The paper by Rodriguez, Wiles, and Elman (1999) directly studied counting in RNNs, while the paper by Kvam and Hintze (2018) discussed decision-making strategies related to counting. Our research aims to build on these works to better understand how RNNs and LSTMs counts.

## III. Method description

In this section, we describe the methodological framework employed to examine the counting capabilities of Recurrent Neural Networks (RNNs) and Long Short-Term Memory networks (LSTMs) in language processing tasks. The procedure is broken down into the following steps:

### *A. Data Generation:*

A dataset is generated using a custom-built function `generate_data`, which creates sequences comprising a certain number of 'a's followed by an equal number of 'b's. The actual count of 'a's serves as the label for each sequence. The `generate_data` function is invoked with a specified number of samples to create the dataset for training and testing the models.

### *B. Data Preprocessing:*

The sequences are converted into a one-hot encoded format using the `sequences_to_one_hot` function, facilitating the feeding of data into the neural network models (Lv, Ding, Wang, & Zou, 2021). Each letter 'a' and 'b' gets its own special pattern of zeros and ones. This way, the neural network can tell the letters apart and learn to count them.

### *C. Model Construction and Training:*

Two types of neural network models, RNN and LSTM, are constructed using the TensorFlow and Keras libraries. Each model comprises a recurrent layer (RNN or LSTM) followed by a Dense layer with a linear activation function. The models are compiled using the Adam optimizer and Mean Squared Error loss function. Both models are trained on the training data for 20 epochs with a batch size of 32.

### *D. Extracting Hidden States:*

Post training, new models are constructed to extract the hidden states of the RNN and LSTM models. For the RNN model, a new model `rnn_hidden_model` is constructed with the same input but outputs the hidden states. Similarly, for the LSTM model, a new model `lstm_hidden_model` is constructed to output both the hidden states (`hidden_h`) and cell states (`hidden_c`).

### *E. Correlation Analysis:*

A correlation analysis is performed to measure the relationship between the hidden states (and cell states for LSTM) and the true count of 'a's in the sequences. The Pearson correlation coefficient is computed for each hidden state across all sequences, providing a measure of the linear relationship between the hidden states and the true count.

### *F. Computing Loss Function*

We utilize Mean Absolute Error (MAE) to gauge the precision of our Recurrent Neural Network (RNN) and Long Short-Term Memory network (LSTM) models. MAE measures the average magnitude of errors between predicted and actual counts of consecutive 'a's before a 'b'. The computed MAE values for both training and validation sets are plotted over each epoch, offering insights into the models' overall performance dynamics.

### *G. Visualization:*

The correlations are visualized using line plots, where the x-axis represents the index of the hidden state, and the y-axis represents the correlation coefficient. Separate plots are generated for RNN, LSTM hidden states (`hidden_h`), and LSTM cell states (`hidden_c`).

The correlation values and plots are analyzed to infer how well the hidden states of RNN and LSTM models are correlated with the true count of 'a's in the sequences. The analysis aims to provide insights into the counting abilities of RNNs and LSTMs, shedding light on how each model processes and learns from sequential data.

## IV. RESULTS AND ANALYSIS

By investigating two different sets of parameters, our goal was to shed light on the distinct mechanisms these networks employ for counting and to assess how these mechanisms influence their overall performance.

### *A. Understanding Counting Abilities*

Starting with the first parameter set (number_of_samples=100, hidden_states=20, number_of_epochs=20), in *figure 1* the RNN displayed a wide range of values in its hidden states. This variation suggests that RNNs might have difficulty in consistently keeping track of information across longer sequences, a crucial aspect in language processing where context and sequence matter a lot. On the other hand, in *figure 2* the LSTM showcased a tendency for higher positive values, indicating its stronger ability to maintain a consistent count and hold onto information over extended sequences.

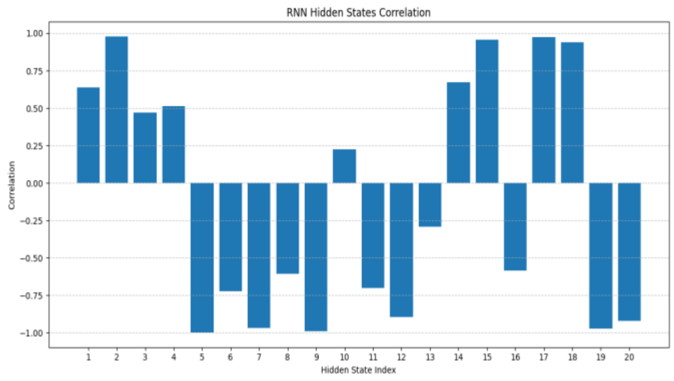

*Figure 1: Correlation of Hidden States with True Count for RNN with 20 epochs*

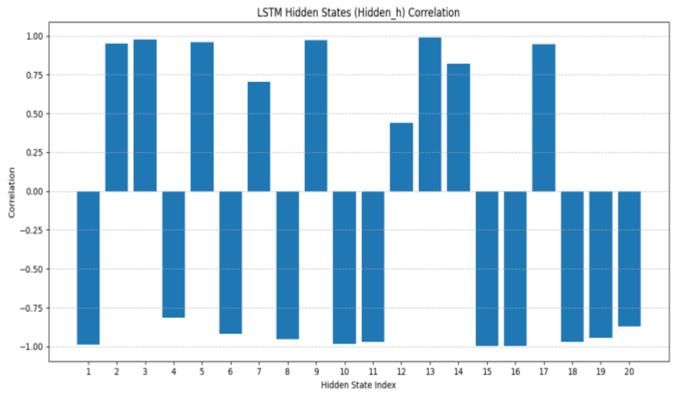

*Figure 2: Correlation of Hidden States with True Count for LSTM with 20 epochs*

### *B. Differences in Counting Mechanisms*

Moving to a more complex scenario with the second parameter set (n=1000, h=60, e=60), in *figure 3* the RNN continued to show inconsistency in its counting ability, with a broad range of correlation values. This inconsistency can be a major drawback when dealing with language processing tasks that require a deep understanding of the context. In contrast, in *figure 4* the LSTM maintained its high performance, showing a focused range of correlation values in its hidden states, further proving its proficiency in handling long sequences, and maintaining an accurate count throughout. The mechanisms within the LSTM, such as the forget gate, input gate, and output gate, play a significant role in this, helping the network decide what information to keep and what to forget, ensuring precision in sequence understanding.

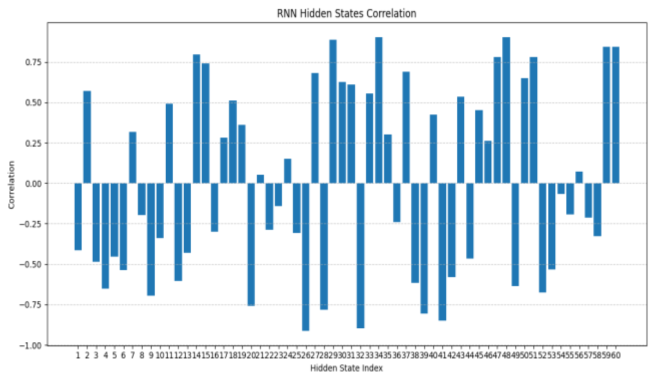

*Figure 3*: *Correlation of Hidden States with True Count for RNN with 60 epochs*

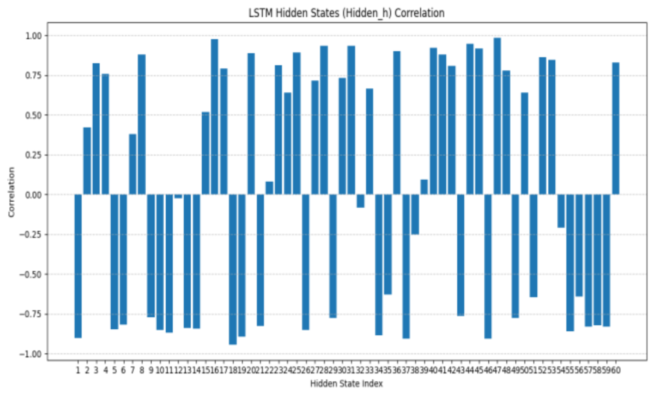

*Figure 4*: *Correlation of Hidden States with True Count for LSTM with 60 epochs*

## C. Impact on Performance

The LSTM employs a memory reset mechanism to concentrate information in cell states. Our findings confirm this, as we can establish a correlation between cell states and true counts. Contrary to expectations, RNNs were anticipated not to exhibit this behavior, but surprisingly, they demonstrate a considerable level of effectiveness in doing so. While not consolidating information into a single state node, almost all nodes in RNNs show correlation. In contrast, for LSTMs, it is observed that only certain nodes exhibit strong correlation, while others do not.

We find correlation for cell state and hidden state is similar which we already anticipate as by construction every cell state has a hidden state. The intention behind conducting the correlation analysis is to understand the process of counting and to assess how RNNs and LSTMs perform in each state. Although the accuracy of the task can be inferred from these observations, we do not explicitly test it in this analysis.

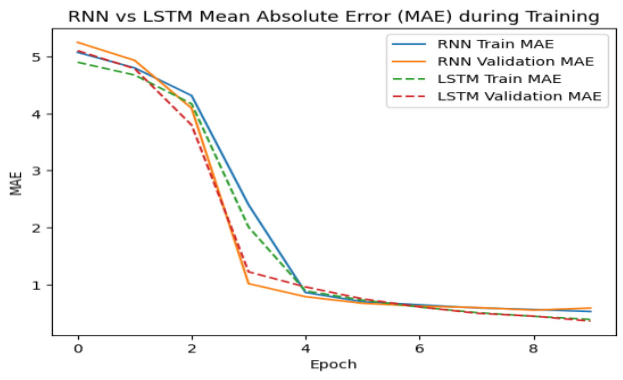

*Figure 5*: *Mean Absolute Error(MAE) of RNN and LSTM*

In *figure 5* the training results show that both the RNN and LSTM models exhibit a decreasing Mean Absolute Error (MAE) during training. The LSTM model achieves a significantly lower final MAE (0.36) compared to the RNN model (0.59), indicating that the LSTM model performs better in predicting the count of consecutive 'a's before 'b' in the given dataset.

Additionally, it is emphasized that since the test accuracy is based on random strings and training accuracy is equivalent, there is no overfitting concern.

## V. CONCLUSION

The analysis points out clear differences between Recurrent Neural Networks (RNNs) and Long Short-Term Memory networks (LSTMs) in terms of counting abilities, which is crucial for language processing tasks.

In the first scenario with simpler parameters, RNNs showed a wide range of values in their hidden states, indicating a potential difficulty in keeping track of counts over longer sequences. This could be a challenge in language processing where understanding sequences is essential. On the other hand, LSTMs displayed higher positive values, suggesting a better ability to keep a consistent count over extended sequences, which is vital for maintaining context in language processing.

In the second, more complex scenario, RNNs continued to show a broad range of correlation values, indicating an inconsistency in counting, which could lead to problems in understanding the context of language tasks. LSTMs maintained a narrow range of correlation values, highlighting their capability to handle long sequences and keep an accurate count, which is beneficial for understanding and processing language accurately.

Overall, while RNNs unexpectedly demonstrated notable counting accuracy, LSTMs consistently outperformed, showcasing their superior ability to maintain precise counts in sequential data.

## VI. LIMITATION

Despite its contributions, this study has some limitations. The dataset used is synthetically generated and focuses solely on simple sequences, which may not fully represent the complexity of real-world language processing tasks. The models examined are basic versions of RNNs and LSTMs, leaving advanced variants and other types of recurrent networks unexplored. The simplicity of the counting task and the reliance on correlation analysis may not capture the full range of the models' capabilities or the nuances of how they process sequential data. Additionally, the study’s scope is limited to the internal states of the models, without delving into other influential factors like the operation of LSTM gates or the impact of initial states. We acknowledge that generalizability of this experiment remains an open question, and its confirmation is not currently possible. However, future studies could incorporate more intricate experiments to further investigate the novel architectures or enhancements to improve counting ability in neural networks for broader applications.

## REFERENCES

1. Rodriguez, P., Wiles, J., & Elman, J. L. (1999). A Recurrent Neural Network that Learns to Count. Connection Science, 11(1), 5-40.

2. Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), 1735-1780.

3. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. In Advances in neural information processing systems (pp. 3111-3119).

4. Lv, Z., Ding, H., Wang, L., & Zou, Q. (2021). A convolutional neural network using dinucleotide one-hot encoder for identifying DNA N6-methyladenine sites in the rice genome. Neurocomputing, 422, 214-221.