# Turing Completeness of Bounded-Precision Recurrent Neural Networks

**Stephen Chung*, Hava T. Siegelmann***

Previous works have proved that recurrent neural networks (RNNs) are Turing-complete. However, in the proofs, the RNNs allow for neurons with unbounded precision, which is neither practical in implementation nor biologically plausible. To remove this assumption, we propose a dynamically growing memory module made of neurons of fixed precision. The memory module dynamically recruits new neurons when more memories are needed, and releases them when memories become irrelevant. We prove that a 54-neuron bounded-precision RNN with growing memory modules can simulate a Universal Turing Machine, with time complexity linear in the simulated machine's time and independent of the memory size. The result is extendable to various other stack-augmented RNNs. Furthermore, we analyse the Turing completeness of both unbounded-precision and bounded-precision RNNs, revisiting and extending the theoretical foundations of RNNs.

*Equal contribution

## Background & Motivation

### Turing Completeness of RNNs

Symbolic (such as Turing Machines (TMs)) and sub-symbolic processing (such as adaptive neural networks) are two competing methods of representing and processing information. An ultimate way to combine symbolic and sub-symbolic capabilities is by enabling the running of algorithms on a neural substrate, which means a neural network that can simulate a Universal Turing Machine (UTM).

Previous works [1-3] have shown that this is possible – there exists a recurrent neural network (RNN) that can simulate a UTM. These proofs assumed a couple of neurons with unbounded precision that equals the number of symbols used in the tape. This capability is enabled due to the use of **fractal encoding** to encode the tape symbols by two neurons:

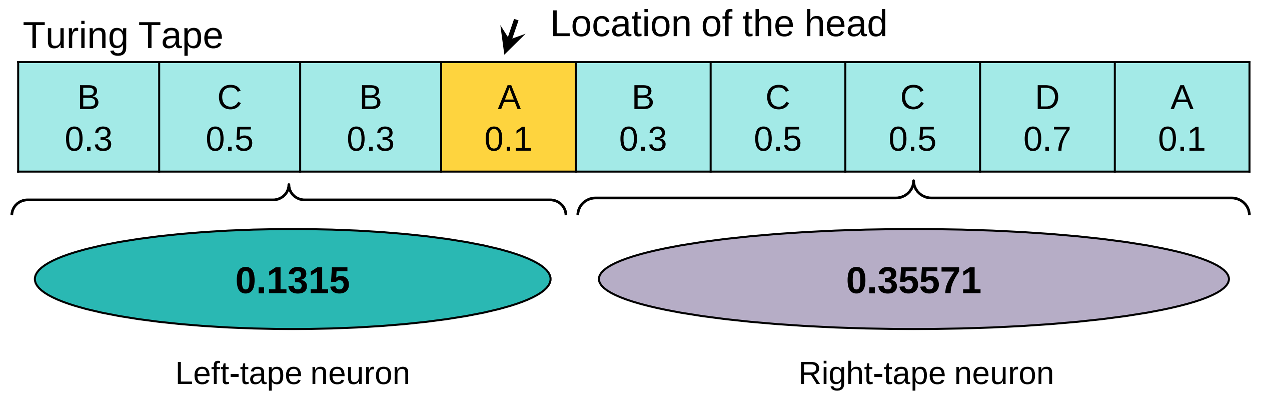

**Fig 1 Illustration of fractal encoding.** Assume that there are four symbols {A, B, C, D}, which are be encoded by value {0.1, 0.3, 0.5,0.7}. To store the entire left tape (the string of symbols starting at the symbol under the read/write head and extending to the left) into a single neuron, we add the tape's symbols together after shifting one decimal point per distance from the head, which is 0.1315 in this example. Similar encoding is used for the right tape (the remaining string of symbols).

However, these previous work required the tape neurons to have the same precision as the size of the active (non-blank) part of the tape, which is usually not applicable. Therefore, in this paper, we consider how to simulate a UTM with different forms of bounded-precision RNNs.

## 1. Unbounded-precision RNNs

We show that **a 40-neuron unbounded precision network can simulate a UTM with linear time complexity (Theorem 1)**, which is the smallest Turing-complete RNN to date. See Fig 4 for the architecture.

## 2. Stack-augmented Bounded-precision RNNs

From now on, we assume that each neuron can hold at most p symbols. Therefore, while we still use the fractal encoding introduced before, we divide the string of symbols to short strings of p symbol :

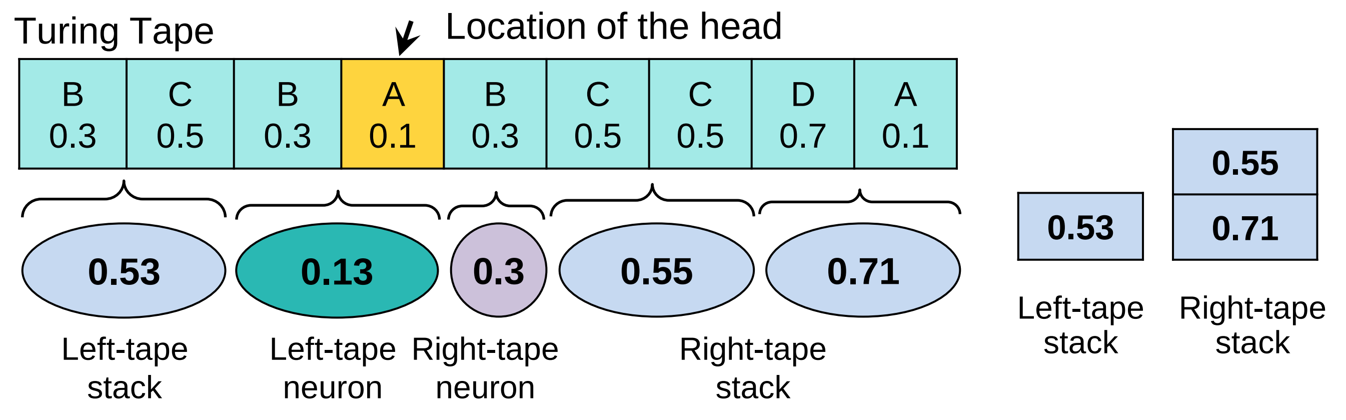

**Fig 2. Illustration of fractal encoding with bounded-precision neurons.** Instead of storing the whole tape by two neurons, we now use one neuron to store each p=2 symbols.

We only keep the two neurons which store values closest to the head in the RNN, and the other neurons reside in a stack-like memory. We propose a simple mechanism for the RNN to interact with the stack-like memory by two neurons, called push (u) and pop (o) neurons, as follows:

**Push neuron (*u*)**
Whenever *u* is non-zero, its value is pushed to the stack and *u* is reset to zero

**Pop neuron (*o*)**
Whenever *o* is zero, the top neuron in the stack is popped and *o* gets the value of the current top neuron in the stack

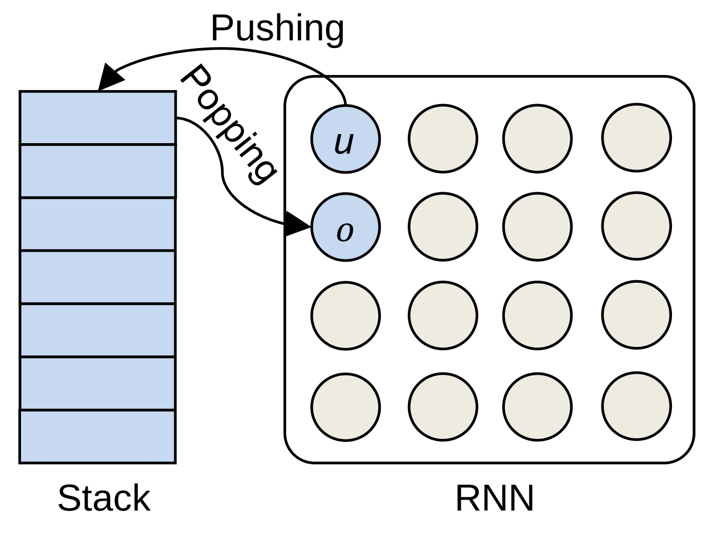

**Fig 3. Illustration of how the RNN control pushing and popping of the growing memory module.**

We call this stack-like memory a **growing memory module**. The advantage of this module is that most neurons there are not updated, saving computational cost. This module shares similarities with other stack-augmented RNNs proposed but with a simpler mechanism [4-6], making the proof extendable to stack-augmented RNNs in general.

We then prove that a **54-neuron bounded-precision RNN (BP-RNN) with growing memory modules can simulate a UTM, with time complexity linear in the simulated machine's time and independent of the memory size (Theorem 2)**. The architecture of the 54-neuron RNN, which is based on the 40-neuron RNN used in Theorem 1, is shown below:

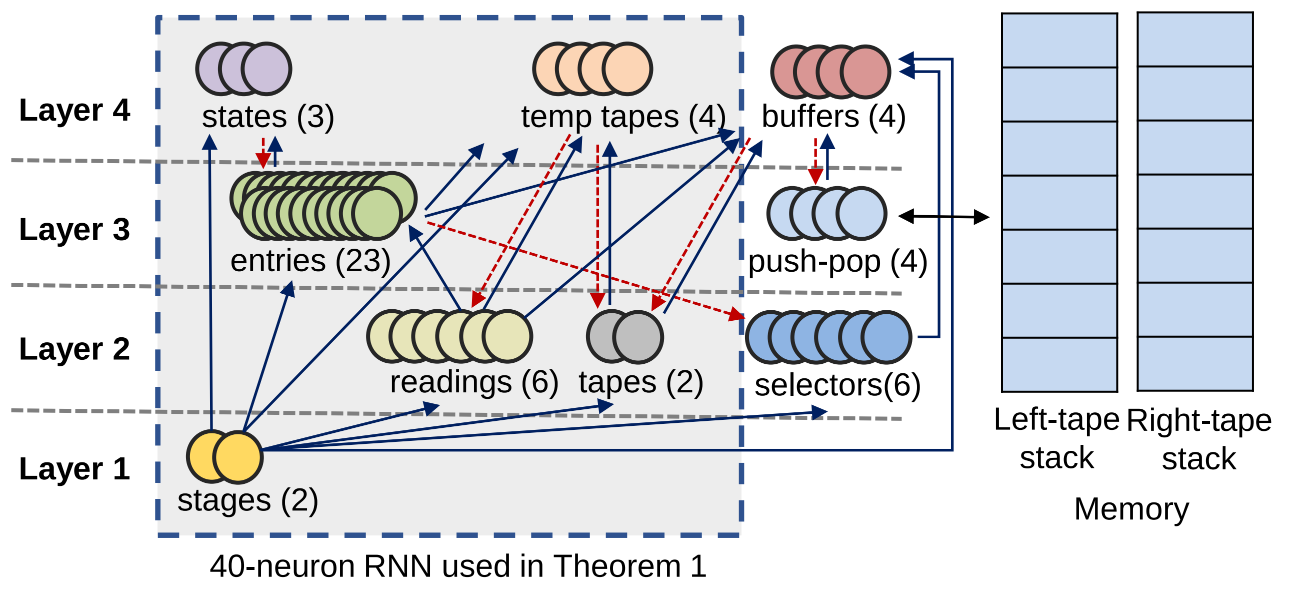

**Fig 4. The architecture of the Turing-complete 54-neuron RNN with two growing memory modules.** Neurons are grouped according to their respective functions. Notable neurons include: state neurons - represent the current state; tape neurons - represent the tape symbols near the head using fractal encoding. The remaining neurons help these two groups of neurons to be updated correctly to simulate a TM's transition.

## Summary

The paper has three main theorems (for completeness we also simulate a space-bounded TM by BP-RNNs (Theorem 3)):

| Type of RNNs | Number of neurons required | Simulated TM |
|---|---|---|
| 1. Unbounded-precision RNNs | 40 | Any TM |
| 2. Stack-augmented BP-RNNs | 54 | Any TM |
| 3. Bounded-precision RNNs (BP-RNNs) | $46+10\lceil F/p \rceil$ | Space-bounded TM |

**Table 1. The number of neurons required to simulate a UTM.** We prove the Turing completeness of 1. and 2. by showing how it can simulate UTM(4,6), a small UTM proposed by Neary & Woods [7]. 3. can only simulate space-bounded TM, where F is the space size and p is the precision of neurons, and so it is not Turing-complete.

### References

[1] Hava T Siegelmann and Eduardo D Sontag. On the computational power of neural nets. Journal of computer and system sciences, 50(1):132–150, 1995.

[2] Hava T Siegelmann. Computation beyond the turing limit. Science, 268(5210):545–548, 1995.

[3] Hava T Siegelmann. Neural networks and analog computation: beyond the Turing limit. Springer Science & Business Media, 2012.

[4] Edward Grefenstette, Karl Moritz Hermann, Mustafa Suleyman, and Phil Blunsom. Learning to transduce with unbounded memory. Advances in neural information processing systems, 28:1828–1836, 2015.

[5] Armand Joulin and Tomas Mikolov. Inferring algorithmic patterns with stack-augmented recurrent nets. Advances in neural information processing systems, 28:190–198, 2015.

[6] Ankur Arjun Mali, Alexander G Ororbia II, and C Lee Giles. A neural state pushdown automata. IEEE Transactions on Artificial Intelligence, 1(3):193–205, 2020.

[7] Turlough Neary and Damien Woods. Four small universal turing machines. Fundamenta Informaticae, 91(1):123?44, 2009.