A Theoretical Analysis of the Incremental Counting Ability of LSTM in Finite Precision

Volodimir Mitarchuk, Rémi Eyraud

A Theoretical Analysis of the Incremental Counting Ability of LSTM in Finite Precision

Volodimir Mitarchuk VOLODIMIR.MITARCHUK@SUNIV-ST-ETIENNE.FR.COM
Rémi Eyraud REMI.EYRAUD@UNIV-ST-ETIENNE.FR
Université Jean Monnet Saint-Etienne, CNRS, Inria, Laboratoire Hubert Curien UMR 5516

Abstract

Since the introduction of the LSTM (Long Short-Term Memory), this type of Recurrent Neural Networks has been the subject of extensive studies to determine its expressiveness. This work is an attempt to fill the gap between the observations made during empirical studies and the theory. We propose in this article to study theoretically the functioning and the limits of the, empirically observed, LSTMs counting mechanism. Our results show that, in finite precision, LSTMs have a limited ability to retrieve the counting information it has deposited in its carry. This reveals an asymmetry in the information storage and retrieval mechanisms in LSTMs.

Keywords: Recurrent Neural Networks, LSTM, Finite Precision, Counting Ability

1. Introduction

Recurrent Neural Networks (RNNs) are a powerful and expressive class of models designed to process sequential data. When focusing on RNNs having unbounded precision and computation time, or bounded precision and growing memory, it is possible to show that this class of models is Turing complete (Siegelmann and Sontag, 1992; Chung and Siegelmann, 2021). On the other hand, the question of precisely characterizing their expressiveness under more realistic assumptions remains open. To this extent, researchers have investigated connections with classical models of the formal language theory and properties such as long-distance dependencies and counting (Weiss et al., 2018, 2022; Eyraud and Ayache, 2021; Li et al., 2022; Merrill, 2019; Merrill et al., 2020; Merrill, 2021; Delétang et al., 2022; Hao et al., 2022; Mali et al., 2021). In this paper, we are particularly interested in analyzing the counting ability of Long Short-Term Memory (LSTM) networks (Hochreiter and Schmidhuber, 1997). Gers and Schmidhuber (2001); Weiss et al. (2018) and Suzgun et al. (2019a) experimentally show that, unlike GRUs, LSTM networks are capable to use a counting mechanism relying on the incrementation of a counter. This allows them to recognize some strictly context-free and context-sensitive languages.

It is therefore natural to wonder if the networks are actually able to use the information deriving from the counting mechanism to its full potential, or if there is a limit number of increments that can be tracked. Specifically, it is not clear if a network with bounded activation function can keep a discrimination capacity on unbounded count.

This work proposes a theoretical analysis of the ability of LSTMs to count elements in a sequence of symbols while in a finite precision configuration. It is trivial that this configuration comes with a maximal value that each parameter of the network can achieve, and

thus on the number of elements that can be enumerated. But this number is astronomical in classical implementations of LSTM, and thus cannot be considered as a real limitation. However, the intrinsic computing process of these models implies much tighter limits that we describe in this article. The core idea is that when the RNN simulates an incremental mechanism on its carry, the values stored will rapidly saturate the activation function and thus the information will be no more retrievable from the hidden state. As this latter is the only element outputted by the recurrent part of the network, no correct decision can then be made based on the count.

After introducing the main notions and notations in Section 2, we first state the problem of LSTM counting abilities in Section 3. In Section 4, we analyse the limitation, in finite precision, of a fixed increment counting mechanism, a behavior that has been observed experimentally in trained LSTM. In Section 5 we discuss some implications of the theoretical results obtained in Section 4. Finally after discussing our results in Section 6, we conclude in Section 7.

2. Preliminaries

2.1. Notations

2.1.1. Numbers, vectors and matrices

Numbers, besides properly defined constants, are noted by a lowercase italic letter, *e.g.*, $x \in \mathbb{R}$, the vectors of dimension higher than 1 by a bold lowercase letter, *e.g.*, $\mathbf{x} \in \mathbb{R}^d$ for $d > 1$, and the matrices by a bold uppercase letter, *e.g.*, $\mathbf{M} \in \mathbb{R}^{d_1 \times d_2}$. For $\mathbf{x} \in \mathbb{R}^d$, $\mathbf{x}[j]$ represents the j^{th} coordinate of the vector $\mathbf{x}$.

$\odot$ is the Hadamard product between vectors: for $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ the Hadamard product of $\mathbf{x}$ by $\mathbf{y}$ denoted $\mathbf{x} \odot \mathbf{y}$ is defined by : for all $1 \leq j \leq d$, $\mathbf{x} \odot \mathbf{y}[j] = \mathbf{x}[j] \cdot \mathbf{y}[j]$.

2.1.2. Activation functions

The logistic sigmoidal function is $\sigma(x) = \frac{1}{1+\exp(-x)}$ and the hyperbolic tangent is $\tanh(x) = \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}$. Given a derivable function $f : \mathbb{R} \to \mathbb{R}$, we note $f'(x) = \frac{df}{dx}(x)$.

2.1.3. Basics of language theory

A finite set of symbol is called an alphabet and is usually denoted Σ. A sequence on Σ is denoted $w = \alpha_1\alpha_2 \dots \alpha_T$, with $\alpha_i \in \Sigma, \forall i$; T is called the length of the sequence. Given a sequence w and a symbol a, $|w|_a$ denotes the number of a's in w. The set of all sequences over the alphabet is denoted Σ^* and any subset $L \subseteq \Sigma^*$ is called a language. For instance, on the alphabet $\Sigma = \{a, b\}$, the infinite language $L = \{a^n b^n : n > 0\}$ is composed of sequences that start with a certain number of a's followed by the exact same number of b's, and nothing else after.

We will trivially extend these formal language elements to sequences of 1-hot vectors $\{v_k\}_{k=1}^T$ where $\forall k, \mathbf{v}_k \in \mathbb{R}^{|\Sigma|}$ and $\exists j, 1 \leq j \leq |\Sigma|$ such that $\mathbf{v}_k[j] = 1$ and $\mathbf{v}_k[i] = 0$, for $i \neq j$.

2.2. Finite Precision

Definition 1 (Finite precision configuration) *A finite precision configuration is defined by a tuple of positive integers (B, M, Ex), B being the basis, M and Ex representing the number of digits allocated for the "mantissa" and the "exponent", respectively.*

In this configuration a float number x is represented as a triplet:

$$float_x = (sign, mantissa, exponent)$$

where $sign \in \{-1, 1\}$, $mantissa \in \{0, 1, \dots, B-1\}^M$, $exponent \in \{0, 1, \dots, B-1\}^{Ex}$ are word of length M and Ex respectively, on the alphabet $\{0, 1, \dots, B-1\}$

In order to go from the representation to the number itself, one needs to:

$$x = (sign) \times (mantissa) \times B^{shift(exponent)}$$

where: $\{0, 1, \dots, B-1\}^{Ex} \ni x \mapsto shift(x) = x - \left\lfloor \frac{B^E}{2} \right\rfloor$

The function $shift$ allows to represent positive and negative exponents with only positive integers.

For example in the configuration $(B, M, Ex) = (10, 5, 4)$ the number $x := -12.35$ is represented by the triplet $(-1, 12350, 4997)$ and when we switch to the actual number we get

$$x = -1 \times 12350 \times 10^{shift(4997)} = -1 \times 12350 \times 10^{-3}$$

where in this case $shift(n) \ = n - \frac{10^4}{2} \ = n - 5000$.

One can remark that this notation allows multiple representations for the same number, indeed both representations $(-1, 12350, 4997)$ and $(-1, 01235, 4998)$ will produce the same float number -12.35 in the finite precision configuration $(10, 5, 4)$. Therefor to avoid all ambiguities, we say that a non zero float number is in its *normal form* $(sign, mantissa, exponent)$ if the $mantissa = x_1 \cdots x_M$ is such that $x_1 \neq 0$.

The setting (B, M, Ex) generates the set $\mathbb{G}_{(B,M,Ex)}$ composed of all numbers representable following these restrictions. We will abusively use the notation $\mathbb{G}$ instead of $\mathbb{G}_{(B,M,Ex)}$ for the sake of simplicity when there is no ambiguity.

As an example, in the IEEE 754 norm, the simple floats are encoded in base $B = 2$ on 32 bits, with 1 bit for the sign, $M = 23$ for the mantissa and $Ex = 8$ for the exponent.

Arithmetic operations in a finite precision configuration might sometimes be counter intuitive. One of the particularities is the rounding error when applying the addition operator. For instance, suppose one wants to compute the addition of two numbers x and s. Let $x = (sign_x) \times x_1 \cdots x_M \times B^{exponent_x}$ and $s = (sign_s) \times s_1 \cdots s_M \times B^{exponent_s}$ represented in their normal form and such that $exponent_s < exponent_x$. If $exponent_x - exponent_s \geq M$ i.e. the difference between the exponents exceeds the size of the mantissa fixed by the finite precision configuration, then $x \pm s = x$. In other words, distant magnitudes do not interact with each other by the addition operator. This particularity of finite precision arithmetic's

is central in our study, where we try to establish theoretical limitation for an incremental counting mechanism.

In the following Lemma we discuss another phenomenon, the saturation of the bounded activation function. In theory, functions as the sigmoidal function and the hyperbolic tangent, never reach their boundaries when evaluated on real numbers. However in finite precision they do reach the boundaries, and the following lemma characterises the smallest float number to *saturate* the sigmoidal function.

Lemma 2 (Saturation integer)

Let (B, M, Ex) the finite precision configuration. We define $u := \lfloor \frac{B^{Ex}}{2} \rfloor$ and $l := B^{Ex} - u$. If $M < \min\{u, l\}$ then there exists $N_\sigma, N_{\tanh} \in \mathbb{G}_{(B,M,Ex)}$ two positive float numbers such that $\sigma(N_\sigma) = 1, \tanh(N_{\tanh}) = 1$, $\sigma(-N_\sigma) = 0, \tanh(-N_{\tanh}) = -1$ and

$$\sigma(-(N_\sigma - 1)) \neq 0 \text{ and } \tanh(-(N_{\tanh} - 1)) \neq -1 \, .$$

In this work, we do not exploit the fact that the saturation integer is the smallest to cause overflow in the activation functions. What interests us here is the existence of a float number that saturate the activation functions. Therefore, for the rest of this work, we'll define the integer N as the maximum of the numbers N_σ and $N_{\tanh}$. So, whatever the activation function σ or tanh we are guaranteed to have:

$$\sigma(N) = 1 \;\; \sigma(-N) = 0 \;\; \tanh(N) = 1 \;\; \tanh(-N) = -1 \tag{1}$$

In finite precision norm IEEE 754 we have $N_\sigma = 89$ and $N_{\tanh} \leq 45$. The computations for these values are in Appendix 8.3. A particular consequences of the existence of N is that for all $x \geq N$ we have:

$$\sigma(x) = 1 \;\; \sigma(-x) = 0 \;\; \tanh(x) = 1 \;\; \tanh(-x) = -1$$

Remark 3 *A finite precision setup induces the existence of a smallest float number $\varepsilon > 0$ that is non zero. A property of this float number is: for all float number α in the finite precision setup (B, M, Ex) we have $\forall \;\; 0 \leq \vartheta < \varepsilon \;\; : \alpha + \vartheta = \alpha$.*

In other words all positive real numbers strictly smaller then ε will be rounded to zero. In finite precision norm IEEE 754 we have $\varepsilon = 2^{-128}$.

2.3. Recurrent Neural Networks (RNNs)

This work focuses on the class of RNNs called Long Short Term Memory (LSTM).

Definition 4 (LSTM (Hochreiter and Schmidhuber, 1997)) *Let $\{\mathbf{x}_k\}_{k=1}^T$ a sequence of real vectors of dimension $u \geq 1$, and $\mathbf{h}_0, \mathbf{c}_0$ vectors in $\mathbb{R}^d$. The execution of a LSTM on the sequence $\{\mathbf{x}_k\}_{k=1}^T$ is the sequence of vectors $\{(\mathbf{h}_k, \mathbf{c}_k)\}_{k=1}^T$ obtained recursively by the*

following rule:

$$\begin{aligned}
\mathbf{f}_k &= \sigma\left(\mathbf{U}_f \mathbf{x}_k + \mathbf{W}_f \mathbf{h}_{k-1} + \mathbf{b}_f\right) \\
\mathbf{i}_k &= \sigma\left(\mathbf{U}_i \mathbf{x}_k + \mathbf{W}_i \mathbf{h}_{k-1} + \mathbf{b}_i\right) \\
\mathbf{o}_k &= \sigma\left(\mathbf{U}_o \mathbf{x}_k + \mathbf{W}_o \mathbf{h}_{k-1} + \mathbf{b}_o\right) \\
\mathbf{g}_k &= \tanh\left(\mathbf{U}_g \mathbf{x}_k + \mathbf{W}_g \mathbf{h}_{k-1} + \mathbf{b}_g\right) \\
\mathbf{c}_k &= \mathbf{f}_k \odot \mathbf{c}_{k-1} + \mathbf{i}_k \odot \mathbf{g}_k \\
\mathbf{h}_k &= \mathbf{o}_k \odot \tanh\left(\mathbf{c}_k\right).
\end{aligned}$$

The LSTM is an architecture based on the so called gate mechanism. In this context the gates are the vectors $\mathbf{f}_k, \mathbf{i}_k, \mathbf{o}_k, \mathbf{g}_k$ which regulate the flow of information from one iteration to the next one, through the vector $\mathbf{c}_k$ called the *carry*. Finally the vector $\mathbf{h}_k$ is called the *hidden state vector*, or simply the *hidden vector*. The hidden vector is in charge of transmitting the information from the carry to the gates and to the output. When we talk in general terms about the carry, we omit to mention the index k and denote it by $\mathbf{c}$. The same will be applied to the hidden vector and the gates.

The interaction between gates is complex and is nearly impossible to predict in the general setting. Nevertheless, by simplifying the behavior of the gates it is possible to study the LSTM architecture, and one way of doing that is the one of saturation:

Definition 5 (Saturated LSTM (Merrill, 2019)) *Given a LSTM $\mathcal{R}$ and a parameter θ, we defined the LSTM $\mathcal{R}_\theta$ by multiplying all parameters of $\mathcal{R}$ by θ. The saturated version of $\mathcal{R}$ is $\mathcal{R}_{Sat} = \lim_{\theta \to \infty} \mathcal{R}_\theta$.*

The existence of the saturation integer N in finite precision implies that there exist a finite value for θ such that $\mathcal{R}_\theta = \mathcal{R}_{Sat}$.

By analyzing the carry $\mathbf{c}_k = \mathbf{f}_k \odot \mathbf{c}_{k-1} + \mathbf{i}_k \odot \mathbf{g}_k$, it was observed that when a LSTM is saturated, the coordinates of $\mathbf{c}_{k-1}$ can be incremented by a quantity $\gamma \in \{-1, 1\}$ (Merrill et al., 2020; Weiss et al., 2018; Merrill, 2019). In detail, for some $1 \leq j \leq d$ we have :

$$\mathbf{f}_k[j] = 1, \quad \mathbf{g}_k[j] = 1, \quad \mathbf{i}_k[j] = \gamma$$

and obtain $\mathbf{c}_k[j] = \mathbf{c}_{k-1}[j] + \gamma$.

3. LSTM counting ability in finite precision

In this section, we introduce the definition of reliable incremental mechanisms. Leveraging the finite precision assumption, we derive results on the limits of the interaction that the output of a LSTM can have with its carry.

3.1. Reliable incremental counting mechanism

From the definition of LSTM networks, the k^{th} output of a LSTM is a function of the hidden state vector $\mathbf{h}_k$, which is defined by: $\mathbf{h}_k = \mathbf{o}_k \odot \tanh\left(\mathbf{c}_k\right)$. If the LSTM has a counting

mechanism based on incrementing a coordinate j of the carry $\mathbf{c}$, it may not be easy to retrieve from the hidden vector $\mathbf{h}$ the exact number of increments that have been added to $\mathbf{c}[j]$. In fact, we show that there can only be a finite amount of retrievable increments from the hidden state vector $\mathbf{h}$, and will call this amount Γ. We say that an incremental mechanism is *reliable* if it is possible to retrieve from the hidden state vector the exact amount of increments that was added at every time step k, for $1 \leq k \leq T$.

In the following section we discus, from a theoretical point of view, the limitation of the incremental counting mechanism in LSTMs.

4. Incremental counting

It has been observed experimentally (Weiss et al., 2018; Suzgun et al., 2019b) that trained LSTMs can use the coordinates of their carry $\mathbf{c}$ to count. This allows them to correctly recognize context-free languages such as $\{a^n b^n : b > 0\}, \{a^n b^n c^n : n > 0\}, \{w : |w|_a = |w|_b\}$ and is crucial for the languages of brackets.

The framework analysed here is the one where a LSTM is fed with a sequence $\{\mathbf{x}_k\}_{k=1}^T$ that requires the model to count the number of occurrences of some of the elements in the sequence. As we study the maximal amount of increments Γ that can be retrieved from the hidden vector $\mathbf{h}$, without loss of generality, we can suppose that $\mathbf{x}_1 = \mathbf{x}_2 = \cdots = \mathbf{x}_T$ and thus that the goal is to store the number of elements in the sequence. We will denote by the Greek letter γ the increment added to a coordinate of the carry during the incremental counting.

4.1. Limits of incremental counting with fixed increments

In this subsection we consider the case where the value of the increment is fixed. The page limitation does not allow us to develop on the case where the value of the increment is dynamic. Nevertheless, the fixed increment counting mechanism is the one empirically observed by Weiss et al. (2018) and Suzgun et al. (2019b) in trained LSTMs. We recall that ε denotes the smallest float number in finite precision. We say that a coordinate j of the carry $\mathbf{c}_k$ simulates a fixed increment mechanism on the sequence $\{\mathbf{x}_k\}_{k=1}^T$ if there exists $\gamma > 0$, such that: $\quad \mathbf{c}_k[j] = \mathbf{c}_0[j] + k \cdot \gamma$.

If the coordinate $1 \leq j \leq d$ of the carry vector has a fixed incremental mechanism, then by definition we have $\mathbf{h}_k[j] = \mathbf{o}_k[j] \cdot \tanh\left(\mathbf{c}_0[j] + k \cdot \gamma\right)$. We also assume here that the j^{th} coordinate of $\mathbf{o}_k$ is fixed and equal to 1 for all k. We make this hypothesis to simplify the computations, but also because this assumption corresponds to the best case scenario for the transmission of information from $\mathbf{c}_k[j]$ to $\mathbf{h}_k[j]$. Our goal here is to consider the distance between the coordinates of two consecutive hidden state vectors. We consider:

$$\begin{aligned}
\left|\mathbf{h}_{k+1}[j] - \mathbf{h}_k[j]\right| &= \left|\mathbf{o}_{k+1}[j] \cdot \tanh\left(\mathbf{c}_0[j] + (k+1) \cdot \gamma\right) - o_k[j] \cdot \tanh\left(\mathbf{c}_0[j] + (k) \cdot \gamma\right)\right| \\
&= \left|\tanh\left(\mathbf{c}_0[j] + (k+1) \cdot \gamma\right) - \tanh\left(\mathbf{c}_0[j] + (k) \cdot \gamma\right)\right| \\
&= \tanh\left(\mathbf{c}_0[j] + (k+1) \cdot \gamma\right) - \tanh\left(\mathbf{c}_0[j] + k \cdot \gamma\right) \text{ because tanh is increasing} \\
&= \gamma \tanh'(w_k) \quad \text{for } w_k \text{ s.t. } k \cdot \gamma < w_k - \mathbf{c}_0[j] < (k+1) \cdot \gamma,
\end{aligned}$$

where the last equality follows from the mean value theorem. We are sure to loose the reliability of the counting when the distance $\left|\mathbf{h}_{k+1}[j] - \mathbf{h}_k[j]\right|$ becomes smaller than the finite precision. Since $\tanh'$ is a decreasing function on $[0, +\infty[$, we have:

$$\gamma \tanh'(w_k) \leq \gamma \tanh'(\mathbf{c}_0[j] + k \cdot \gamma)$$

and thus $\left|\mathbf{h}_{k+1}[j] - \mathbf{h}_k[j]\right| \leq \gamma \tanh'(\mathbf{c}_0[j] + k \cdot \gamma)$. This last inequality motivates the following lemma.

Lemma 6 (Bound on the incremental counting on one coordinate of the carry) *Let α, γ be positive real numbers so that $0 < \gamma \leq 1$. Let c_0 be a real number. Then we have* $\left(K \geq \frac{\tanh^{-1}\left(\sqrt{1-\varepsilon/\alpha}\right) - c_0}{\gamma}\right) \Rightarrow \left(\alpha \tanh'(c_0 + K\gamma) < \epsilon\right)$.

The proof of Lemma 6 is based on basic calculus and is in Appendix 8.1.

From Lemma 6 one can deduce that:

$$\left(\frac{\tanh^{-1}\left(\sqrt{1-\epsilon/\gamma}\right) - \mathbf{c}_0}{\gamma} < K\right) \Longrightarrow \left(\left|\mathbf{h}_{k+1} - \mathbf{h}_k\right| < \varepsilon\right). \tag{2}$$

With this Lemma we can compute an upper bound on the number of distinguishable increments. As an example in IEEE 754 single float numbers, if we consider the experimentally observed case where $\gamma = 1$ and $\mathbf{c}_0 = 0$, one can compute an upper bound on the maximal number of distinguishable increments $K \approx 65 \ln(2) \leq 45.06$ (see Appendix 8.2 for detail).

By reducing γ it is possible to increase the number of distinguishable increments. However in finite precision, the rounding mechanism imposes an absolute upper bound on the number of distinguishable increments. Indeed in Section 2.2 we discus how in a finite precision setup (B, M, Ex) it is possible to have $x + s = x$ if $exponent_x - exponent_s \geq M$. This particularity of arithmetic in finite precision yields an upper bound on Γ which is

$$\Gamma \leq 2 \cdot B^M .$$

The factor 2 comes from the fact that it is, theoretically, possible to initialize $\mathbf{c}_0[j] = -\gamma \cdot B^M$ making it possible to add $2\gamma \cdot B^M$ to $\mathbf{c}_0[j]$ before facing the finite precision boundary.

In conclusion, we can combine the upper bounds we have discussed above and derive the maximal number of distinguishable increments on the carry coordinate $\mathbf{c}_0[j]$ is :

$$\Gamma \leq \min\left\{2 \cdot B^M, \left\lfloor \tanh^{-1}\left(\sqrt{1 - \tfrac{\varepsilon}{\gamma}} - \mathbf{c}_0[j]\right)\right\rfloor + 1\right\} \tag{3}$$

In the following section we discuss some possible implications of Bound 3.

5. Theoretical consequences

5.1. Increment on several coordinates

Let us assume that a LSTM can accept the language $a^n b^n$ but the counting of a and b is independent and done on different coordinates j and j'. By the previous section, we

know that the maximum amount of distinguishable increments cannot exceed Γ. Therefore for $n > \Gamma$ the LSTM will no longer be able to distinguish between $a^n b^n$ and $a^n b^{n+1}$. This problem can be addressed by incrementing a single coordinate of the carry when an a is seen and decrementing it when a b is seen. By doing so there is no more theoretical limitation since in order to accept a word the count in $\mathbf{c}_T[j]$ has to be equal to 0. This is exactly how the LSTMs from Weiss et al. (2018) and Suzgun et al. (2019b) are operating.

5.2. LSTMs have a limited ability to adapt their behavior based on the count

We assume to have a LSTM that simulates a fixed incremental mechanism on the j^{th} coordinate of the carry vector $\mathbf{c}$ with fixed increment γ. We consider the case where a gate $\mathbf{i}, \mathbf{f}, \mathbf{g}$ or $\mathbf{o}$ has to change its behavior (for example $\mathbf{g}[l]$ goes from 1 to -1) based uniquely on a specific value n of the count $\mathbf{c}[j]$. The existence of Γ demonstrates that this kind of change is not possible if n is an unbounded variable. Indeed suppose that we have $n > \Gamma$, with $\Gamma = \min\left\{2 \cdot B^M, \left\lfloor \tanh\left(\sqrt{1 - \frac{\varepsilon}{\gamma}} - \mathbf{c}_0[j]\right)\right\rfloor + 1\right\}$. If $\Gamma = 2 \cdot B^M$ then by the rounding mechanism in finite precision we will have $\gamma \cdot \Gamma + \gamma = \gamma \cdot \Gamma$, hence the carry $\mathbf{c}[j]$ will not be able to count till n. If $\Gamma = \left\lfloor \tanh\left(\sqrt{1 - \frac{\varepsilon}{\gamma}} - \mathbf{c}_0[j]\right)\right\rfloor + 1$ than we will have $\tanh(\mathbf{c}[j] + \gamma) - \tanh(\mathbf{c}[j]) < \varepsilon$, i.e. the distance between two consecutive counts will be smaller then the finite precision, implying that if a change occurs in $\mathbf{g}[l]$ it will occur before reaching n (or any other gate than $\mathbf{g}$). Thus in any case, the gates $\mathbf{i}, \mathbf{f}, \mathbf{g}$ and $\mathbf{o}$ will not be able to change their behavior only based on n if the tipping value n exceeds Γ. Leading us to believe that LSTMs do not have the ability to recognise the language $\{a^n b^{n^2} : n \geq 1\}$ because the machine that accepts this language adapts its counting behavior according to n. Page limitation does not allow us to detail further on this result, thus leaving room for a more developed work on this topic.

6. Discussion

This work is based on the fact that the information contained in the carry is transmitted to the hidden state vector through the tanh, in other words $\mathbf{h}_k = \mathbf{o}_k \odot \tanh(\mathbf{c}_k)$. This is not the case in peephole LSTM (Gers and Schmidhuber, 2001) where the gates receive the information directly from the carry. Therefore peephole LSTMs have the potential to be more expressive than LSTMs defined by Hochreiter and Schmidhuber (1997). This statement is to be proven theoretically and might constitute a future work.

7. Conclusion

In this work we discuss theoretical limitation of LSTM fixed increment counting mechanism in finite precision. Based on a formal definition of the fixed increment counting mechanism we derived realistic upper bounds on the maximal number of distinguishable increments that can be retrieved for the hidden vector. Finally we discussed some of the implication of this limitation.

References

Stephen Chung and Hava T. Siegelmann. Turing completeness of bounded-precision recurrent neural networks. In *NeurIPS*, 2021.

Grégoire Delétang, Anian Ruoss, Jordi Grau-Moya, Tim Genewein, Li Kevin Wenliang, Elliot Catt, Marcus Hutter, Shane Legg, and Pedro A. Ortega. Neural networks and the chomsky hierarchy. *CoRR*, 2022. doi: 10.48550/arXiv.2207.02098.

Rémi Eyraud and Stéphane Ayache. Distillation of weighted automata from recurrent neural networks using a spectral approach. *Machine Learning*, 2021.

Felix A Gers and E Schmidhuber. Lstm recurrent networks learn simple context-free and context-sensitive languages. *IEEE transactions on neural networks*, 2001.

Yiding Hao, Dana Angluin, and Robert Frank. Formal language recognition by hard attention transformers: Perspectives from circuit complexity. *Transactions of the Association for Computational Linguistics*, 2022. doi: 10.1162/tacl_a_00490.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 1997.

Tianyu Li, Doina Precup, and Guillaume Rabusseau. Connecting weighted automata, tensor networks and recurrent neural networks through spectral learning. *Machine Learning*, 2022.

Ankur Mali, Alexander Ororbia, Daniel Kifer, and Lee Giles. Recognizing long grammatical sequences using recurrent networks augmented with an external differentiable stack. In Jane Chandlee, Rémi Eyraud, Jeff Heinz, Adam Jardine, and Menno van Zaanen, editors, *ICGI 2021*. PMLR, 2021.

William Merrill. Sequential neural networks as automata. In *Proceedings of the Workshop on Deep Learning and Formal Languages: Building Bridges*, pages 1–13, Florence, August 2019. Association for Computational Linguistics.

William Merrill. Formal language theory meets modern nlp. *arXiv preprint arXiv:2102.10094*, 2021.

William Merrill, Gail Weiss, Yoav Goldberg, Roy Schwartz, Noah A Smith, and Eran Yahav. A formal hierarchy of rnn architectures. *arXiv preprint arXiv:2004.08500*, 2020.

Hava T Siegelmann and Eduardo D Sontag. On the computational power of neural nets. In *COLT*, 1992.

Mirac Suzgun, Yonatan Belinkov, and Stuart M Shieber. On evaluating the generalization of lstm models in formal languages. *SCiL*, 2019a.

Mirac Suzgun, Yonatan Belinkov, Stuart M Shieber, and Sebastian Gehrmann. Lstm networks can perform dynamic counting. In *Workshop on Deep Learning and Formal Languages: Building Bridges*, 2019b.

Gail Weiss, Yoav Goldberg, and Eran Yahav. On the practical computational power of finite precision RNNs for language recognition. In *TACL*. Association for Computational Linguistics, 2018. doi: 10.18653/v1/P18-2117.

Gail Weiss, Yoav Goldberg, and Eran Yahav. Extracting automata from recurrent neural networks using queries and counterexamples. *Machine Learning*, 2022.

8. Appendix

8.1. Proof of Lemma 6

Proof [Bound on the incremental counting] Let α, γ be positive real numbers, with $\alpha \geq 0$ and $0 < \gamma \leq 1$. Let $\varepsilon > 0$. If $\alpha = 0$ there is nothing to prove and the inequality is true for $K = 1$. If $\alpha > 0$ then:

$$\begin{aligned}
\alpha \tanh'(c_0 + K\gamma) < \epsilon &\iff \tanh'(c_0 + K\gamma) < \frac{\epsilon}{\alpha} \\
&\iff 1 - \tanh^2(c_0 + K\gamma) < \frac{\epsilon}{\alpha} \\
&\iff 1 - \frac{\epsilon}{\alpha} < \tanh^2(c_0 + K\gamma) \\
&\iff \sqrt{1 - \frac{\epsilon}{\alpha}} < |\tanh(c_0 + K\gamma)| \\
&\iff \sqrt{1 - \frac{\epsilon}{\alpha}} < \tanh(c_0 + K\gamma) \quad \text{if } K \geq \frac{-c_0}{\gamma} \\
&\iff \tanh^{-1}\left(\sqrt{1 - \frac{\epsilon}{\alpha}}\right) < c_0 + K\gamma \text{ because } \tanh^{-1} \text{ is a strictly increasing function} \\
&\iff \frac{\tanh^{-1}\left(\sqrt{1 - \frac{\epsilon}{\alpha}}\right) - c_0}{\gamma} < K
\end{aligned}$$

As $\frac{\tanh^{-1}\left(\sqrt{1-\frac{\epsilon}{\alpha}}\right)-c_0}{\gamma} > \frac{-c_0}{\gamma}$ this concludes the proof. ■

8.2. A bound on Γ when $\gamma = 1$

In Lemma 6 the expression of the bound is obscured by the expression of the function $\tanh^{-1}$ therefore we exemplify this bound in the particular case where $\gamma = 1$ and $\mathbf{c}_0 = 0$. We have:

$$\frac{\tanh^{-1}\left(\sqrt{1 - \frac{\varepsilon}{\gamma}}\right) - \mathbf{c}_0}{\gamma} = \tanh^{-1}\left(\sqrt{1 - \varepsilon}\right).$$

We start by dealing with $\sqrt{1-\varepsilon}$. The Taylor expansion of $\sqrt{1-\varepsilon}$ to order 2 is:

$$\sqrt{1-\varepsilon} = 1 - \frac{\varepsilon}{2} - \frac{\varepsilon^2}{8} + o(\varepsilon^2).$$

In the case of IEEE 754 finite precision single float numbers $\varepsilon = 2^{-128}$, therefor we can consider the Taylor expansion to order 1 for our computation. We have the exact expression of $\tanh^{-1}$ with:

$$\tanh^{-1}(x) = \frac{1}{2}\ln\left(\frac{1+x}{1-x}\right)$$

thus in our case we obtain:

$$\tanh^{-1}\left(1-\frac{\varepsilon}{2}\right) = \frac{1}{2}\ln\left(\frac{2-\frac{\varepsilon}{2}}{\frac{\varepsilon}{2}}\right) = \frac{1}{2}\ln\left(4\cdot 2^{128}-1\right).$$

Consequently we obtain that:

$$\tanh^{-1}\left(\sqrt{1-\varepsilon}\right) \approx \tanh^{-1}\left(1-\frac{\varepsilon}{2}\right) \leq \frac{130}{2}\ln(2) \approx 45$$

8.3. Saturation Integers

In this section we provide the computations that allowed us to determine the values of saturation integer N_σ and $N_{\tanh}$. We start with N_σ.

The function σ is defined by:

$$\sigma : \mathbb{G} \in x \to \frac{1}{1+e^{-x}} \in [0,1].$$

In IEEE 754, 32 bits float numbers we have $(b, M, Ex) = (2, 23, 8)$. For which value on x we would have $1+e^{-x} = 1$? It would be true for x so that $e^{-x} = 2^{-M}$ therefor if we choose $x \geq \ln(2)\cdot 23$ we will have $1+e^{-x} = 1$ thus $\sigma(x) = 1$. Now, for which x we would have $\frac{1}{1+e^{-x}} < 2^{-128}$?

$$\begin{aligned}
\frac{1}{1+e^{-x}} < 2^{-128} &\iff 1+e^{-x} > 2^{128} \\
&\iff e^{-x} > 2^{128} \\
&\iff -x > \ln(2)\cdot 128 \approx 88.72.
\end{aligned}$$

One could remark that $1+e^{-x} > 2^{128} \iff e^{-x} > 2^{128}$ is not true in general, but it is true in finite precision because in this case the magnitude of e^{-x} is so large that $1+e^{-x} = e^{-x}$. Therefor by setting $89 = N_\sigma \geq \max\{23\ln(2), 126\ln(2)\}$ we obtain the saturation integer for the function σ. For the saturation integer $N_{\tanh}$ we consider the identity $\sigma(x) = \frac{1}{2}\left(\tanh\left(\frac{x}{2}\right)+1\right)$. From this identity one can deduce that $\sigma(89) = 0.5(\tanh(44.5)+1) = 1$ hence in finite precision $1 \geq \tanh(45) \geq \tanh(44.5) = 1$. Therefor $N_{\tanh} \leq 44.5$.