Supplementary Material - On the Practical Computational Power of Finite Precision RNNs for Language Recognition

1 Simplified K-Counter Machines

We use a simplified variant of the k-counter machines (SKCM) defined in (Fischer et al., 1968), which has no autonomous states and makes classification decisions based on a combination of its current state and counter values. This variant consumes input sequences on a symbol by symbol basis, updating at each step its state and its counters, the latter of which may be manipulated by increment, decrement, zero, or no-ops alone, and observed only by checking equivalence to zero. To define the transitions of this model its accepting configurations, we will introduce the following notations:

Notations We define $z : \mathbb{Z}^k \to \{0,1\}^k$ as follows: for every $n \in \mathbb{Z}^k$, for every $1 \leq i \leq k$, $z(n)_i = 0$ iff $n_i = 0$ (this function masks a set of integers such that only their zero-ness is observed). For a vector of operations, $o \in \{-1, +1, \times 0, \times 1\}^k$, we denote by $o(n)$ the pointwise application of the operations to the vector $n \in \mathbb{Z}^k$, e.g. for $o = (+1, \times 0, \times 1)$, $o((5,2,3)) = (6,0,3)$.

We now define the model. An *SKCM* is a tuple $M = \langle \Sigma, Q, q_o, k, \delta, u, F \rangle$ containing:

1. A finite input alphabet Σ
2. A finite state set Q
3. An initial state $q_0 \in Q$
4. $k \in \mathbb{N}$, the number of counters
5. A state transition function

$$\delta : Q \times \Sigma \times \{0,1\}^k \to Q$$

6. A counter update function[1]

$$u : \Sigma \to \{-1, +1, \times 0, \times 1\}^k$$

7. A set of accepting masked[2] configurations

$$F \subseteq Q \times \{0,1\}^k$$

The set of *configurations* of an SKCM is the set $C = Q \times \mathbb{Z}^k$, and the initial configuration is $c_0 = (q_0, \bar{0})$ (i.e., the counters are initiated to zero). The transitions of an SKCM are as follows: given a configuration $c_t = (q, n)$ ($n \in \mathbb{Z}^k$) and input $w_t \in \Sigma$, the next configuration of the SKCM is $c_{t+1} = (\delta(q, w_t, z(n)), u(w_t)(n))$.

The language recognized by a k-counter machine is the set of words w for which the machine reaches an accepting configuration — a configuration $c = (q, n)$ for which $(q, z(n)) \in F$.

Note that while the counters can and are increased to various non-zero values, the transition function δ and the accept/reject classification of the configurations observe only their zero-ness.

1.1 Computational Power of SKCMs

We show that the SKCM model can recognize the context-free and context-sensitive languages $a^n b^n$ and $a^n b^n c^n$, but not the context free language of palindromes, meaning its computational power differs from the language classes defined in the Chomsky hierarchy. Similar proofs appear in (Fischer et al., 1968) for their variant of the k-counter machine.

[1] We note that in this definition, the counter update function depends only on the input symbol. In practice we see that the LSTM is not limited in this way, and can also update according to some state-input combinations — as can be seen when it it is taught, for instance, the language $a^n b a^n$ We do not explore this here however, leaving a more complete characterization of the learnable models to future work.

[2] i.e., counters are observed only by zero-ness.

a^nb^n**:** We define the following SKCM over the alphabet $\{a, b\}$:

1. $Q = \{q_a, q_b, q_r\}$
2. $q_0 = q_a$
3. $k = 1$
4. $u(a) = +1,\ u(b) = -1$
5. for any $z \in \{0, 1\}$:
   $\delta(q_a, a, z) = q_a, \quad \delta(q_a, b, z) = q_b,$
   $\delta(q_b, a, z) = q_r, \quad \delta(q_b, b, z) = q_b$
   $\delta(q_r, a, z) = q_r, \quad \delta(q_r, b, z) = q_r$
6. $C = \{(q_b, 0)\}$

The state q_r is a rejecting sink state, and the states q_a and q_b keep track of whether the sequence is currently in the "a" or "b" phase. If an a is seen after moving to the b phase, the machine moves to (and stays in) the rejecting state. The counter is increased on input a and decreased on input b, and the machine accepts only sequences that reach the state q_b with counter value zero, i.e., that have increased and decreased the counter an equal number of times, without switching from b to a. It follows easily that this machine recognizes exactly the language a^nb^n.

$a^nb^nc^n$**:** We define the following SKCM over the alphabet $\{a, b\}$. As its state transition function ignores the counter values, we use the shorthand $\delta(q, \sigma)$ for $\delta(q, \sigma, z)$, for all $z \in \{0, 1\}^2$.

1. $Q = \{q_a, q_b, q_c, q_r\}$
2. $q_0 = q_a$
3. $k = 2$
4. $u(a) = (+1, \emptyset)$,
   $u(b) = (-1, +1)$,
   $u(c) = (\emptyset, -1)$
5. for any $z \in \{0, 1\}$:
   $\delta(q_a, a) = q_a,\ \delta(q_a, b) = q_b,\ \delta(q_a, c) = q_r,$
   $\delta(q_b, a) = q_r,\ \delta(q_b, b) = q_b,\ \delta(q_b, c) = q_c,$
   $\delta(q_c, a) = q_r,\ \delta(q_c, b) = q_r,\ \delta(q_c, c) = q_c,$
   $\delta(q_r, a) = q_r,\ \delta(q_r, b) = q_r,\ \delta(q_r, c) = q_r$
6. $C = \{(q_c, 0, 0)\}$

By similar reasoning as that for a^nb^n, we see that this machine recognizes exactly the language $a^nb^nc^n$. We note that this construction can be extended to build an SKCM for any language of the sort $a_1^na_2^n...a_m^n$, using $k = m - 1$ counters and $k + 1$ states.

Palindromes: We prove that no SKCM can recognize the language of palindromes defined over the alphabet $\{a, b, x\}$ by the grammar $S \rightarrow x|aSa|bSb$. The intuition is that in order to correctly recognize this language in an one-way setting, one must be able to reach a unique configuration for every possible input sequence over $\{a, b\}$ (requiring an exponential number of reachable configurations), whereas for any SKCM, the number of reachable configurations is always polynomial in the input length.[3]

Let M be an SKCM with k counters. As its counters are only manipulated by steps of 1 or resets, the maximum and minimum values that each counter can attain on any input $w \in \Sigma^*$ are $+|w|$ and $-|w|$, and in particular the total number of possible values a counter could reach at the end of input w is $2|w| + 1$. This means that the total number of possible configurations M could reach on input of length n is $c(n) = |Q| \cdot (2n + 1)^k$.

$c(n)$ is polynomial in n, and so there exists a value m for which the number of input sequences of length m over $\{a, b\}$ — 2^m — is greater than $c(m)$. It follows by the pigeonhole principle that there exist two input sequences $w_1 \neq w_2 \in \{a, b\}^m$ for which M reaches the same configuration. This means that for any suffix $w \in \Sigma^*$, and in particular for $w = x \cdot w_1^{-1}$ where w_1^{-1} is the reverse of w_1, M classifies $w_1 \cdot w$ and $w_2 \cdot w$ identically—despite the fact that $w_1 \cdot x \cdot w_1^{-1}$ is in the language and $w_2 \cdot x \cdot w_1^{-1}$ is not. This means that M necessarily does not recognize this palindrome language, and ultimately that no such M exists.

Note that this proof can be easily generalized to any palindrome grammar over 2 or more characters, with or without a clear 'midpoint' marker.

2 Impossibility of Counting in Binary

While we have seen that the SRNN and GRU cannot allocate individual counting dimensions, the question remains whether they can count using a more elaborate mechanism, perhaps over several dimensions. We show here that one such mechanism — a binary counter — is not implementable in the SRNN.

For the purposes of this discussion, we first define a binary counter in an RNN.

Binary Interpretation In an RNN with hidden state values in the range $(-1, 1)$, the *binary interpretation* of a sequence of dimensions $d_1, ..., d_n$ of its hidden state is the binary number obtained

[3]This will hold even if the counter update function can rely on any state-input combination.

by replacing each positive hidden value in the sequence with a '1' and each negative value with a '0'. For instance: the binary interpretation of the dimensions 3,0,1 in the hidden state vector $(0.5, -0.1, 0.3, 0.8)$ is 110, i.e., 6.

Binary Counting We say that the dimensions $d_1, d_2, ..., d_n$ in an RNN's hidden state implement a *binary counter* in the RNN if, in every transition, their binary interpretation either increases, decreases, resets to 0, or doesn't change.[4]

A similar pair of definitions can be made for state values in the range $(0, 1)$.

We first note intuitively that an SRNN would not generalize binary counting to a counter with dimensions beyond those seen in training — as it would have no reason to learn the 'carry' behavior between the untrained dimensions. We prove further that we cannot reasonably implement such counters regardless.

We now present a proof sketch that a single-layer SRNN with hidden size $n \geq 3$ cannot implement an n-dimensional binary counter that will consistently increase on one of its input symbols. After this, we will prove that even with helper dimensions, we cannot implement a counter that will consistently increase on one input token and decrease on another — as we might want in order to classify the language of all words w for which $\#_a(w) = \#_b(w)$.[5]

Consistently Increasing Counter: The proof relies on the linearity of the affine transform $Wx + Uh + b$, and the fact that 'carry' is a non-linear operation. We work with state values in the range $(-1, 1)$, but the proof can easily be adapted to $(0, 1)$ by rewriting h as $h' + 0.5$, where $h' = h - 0.5$ is a vector with values in the range $(-0.5, 0.5)$.

Suppose we have a single-layer SRNN with hidden size $n = 3$, such that its entire hidden state represents a binary counter that increases every time it receives the input symbol a. We denote by x_a the embedding of a, and assume w.l.o.g. that the hidden state dimensions are ordered from MSB to LSB, e.g. the hidden state vector $(1, 1, -1)$ represents the number 110=6.

Recall that the binary interpretation of the hidden state relies only on the signs of its values. We use p and n to denote 'some' positive or negative value, respectively. Then the number 6 can be represented by any state vector (p, p, n).

Recall also that the SRNN state transition is

$$h_t = \tanh(Wx_t + Uh_{t-1} + b)$$

and consider the state vectors $(-1, 1, 1)$ and $(1, -1, -1)$, which represent 3 and 4 respectively. Denoting $\tilde{b} = Wx_a + b$, we find that the constants U and $\tilde{b}$ must satisfy:

$$\begin{aligned} tanh(U(-1,1,1) + \tilde{b}) &= (p, n, n) \\ tanh(U(1,-1,-1) + \tilde{b}) &= (p, n, p) \end{aligned}$$

As tanh is sign-preserving, this simplifies to:

$$\begin{aligned} U(-1,1,1) &= (p, n, n) - \tilde{b} \\ U(1,-1,-1) &= (p, n, p) - \tilde{b} \end{aligned}$$

Noting the linearity of matrix multiplication and that $(1, -1, -1) = -(-1, 1, 1)$, we obtain:

$$U(-1,1,1) = U(-(1,-1,-1)) = -U(1,-1,-1)$$

$$(p, n, n) - \tilde{b} = \tilde{b} - (p, n, p)$$

i.e. for some assignment to each p and n, $2\tilde{b} = (p, n, n) + (p, n, p)$, and in particular $\tilde{b}[1] < 0$.

Similarly, for $(-1, -1, 1)$ and $(1, 1, -1)$, we obtain

$$\begin{aligned} U(-1,-1,1) &= (n, p, n) - \tilde{b} \\ U(1,1,-1) &= (p, p, p) - \tilde{b} \end{aligned}$$

i.e.

$$(n, p, n) - \tilde{b} = \tilde{b} - (p, p, p)$$

or $2\tilde{b} = (p, p, p) + (n, p, n)$, and in particular that $\tilde{b}[1] > 0$, leading to a contradiction and proving that such an SRNN cannot exist. The argument trivially extends to $n > 3$ (by padding from the MSB).

We note that this proof does not extend to the case where additional, non counting dimensions are added to the RNN — at least not without further assumptions, such as the assumption that the counter behave correctly for *all* values of these dimensions, reachable and unreachable. One may

[4] We note that the SKCMs presented here are more restricted in their relation between counter action and transition, but prefer here to give a general definition. Our proof will be relevant even within the restrictions.

[5] Of course a counter could also be 'decreased' by incrementing a parallel, 'negative' counter, and implementing compare-to-zero as a comparison between these two. As intuitively no RNN could generalize binary counting behavior to dimensions not used in training, this approach could quickly find both counters outside of their learned range even on a sequence where the difference between them is never larger than in training.

argue then that, with enough dimensions, it could be possible to implement a consistently increasing binary counter on a *subset* of the SRNN's state.[6] We now show a counting mechanism that cannot be implemented even with such 'helper' dimensions.

Bi-Directional Counter: We show that for $n \geq 3$, no SRNN can implement an n-dimensional binary counter that increases for one token, σ_{up}, and decreases for another, σ_{down}. As before, we show the proof explicitly for $n = 3$, and note that it can be simply expanded to any $n > 3$ by padding.

Assume by contradiction we have such an SRNN, with $m \geq 3$ dimensions, and assume w.l.o.g. that a counter is encoded along the first 3 of these. We use the shorthand $(v_1, v_2, v3)c$ to show the values of the counter dimensions explicitly while abstracting the remaining state dimensions, e.g. we write the hidden state $(-0.5, 0.1, 1, 1, 1)$ as $(-0.5, 0.1, 1)c$ where $c = (1, 1)$.

Let x_{up} and x_{down} be the embeddings of σ_{up} and σ_{down}, and as before denote $b_{up} = W x_{up} + b$ and $b_{down} = W x_{down} + b$. Then for some reachable state $h_1 \in \mathbb{R}$ where the counter value is 1 (e.g., the state reached on the input sequence σ_{up}[7])), we find that the constants U, b_{down}, and b_{up} must satisfy:

$$tanh(Uh_1 + b_{up}) = (n, p, n)c_1$$
$$tanh(Uh_1 + b_{down}) = (n, n, n)c_2$$

(i.e., σ_{up} increases the counter and updates the additional dimensions to the values c_1, while σ_{down} decreases and updates to c_2.) Removing the sign-preserving function tanh we obtain the constraints

$$Uh_1 + b_{up} = (n, p, n)\text{sign}(c_1)$$
$$Uh_1 + b_{down} = (n, n, n)\text{sign}(c_2)$$

i.e. $(b_{up} - b_{down})[0:2] = (n, p, n) - (n, n, n)$, and in particular $(b_{up} - b_{down})[1] > 0$. Now consider a reachable state h_3 for which the counter value is 3. Similarly to before, we now obtain

$$Uh_3 + b_{up} = (p, n, n)\text{sign}(c_3)$$
$$Uh_3 + b_{down} = (n, p, n)\text{sign}(c_4)$$

from which we get $(b_{up} - b_{down})[0:2] = (p, n, n) - (n, p, n)$, and in particular $(b_{up} - b_{down})[1] < 0$, a contradiction to the previous statement. Again we conclude that no such SRNN can exist.

[6](By storing processing information on the additional, 'helper' dimensions)

[7](Or whichever appropriate sequence if the counter is not initiated to zero.)

References

Patrick C. Fischer, Albert R. Meyer, and Arnold L. Rosenberg. 1968. Counter machines and counter languages. *Mathematical systems theory*, 2(3):265–283.