Gating Sensory Noise in a Spiking Subtractive LSTM

Isabella Pozzi, Roeland Nusselder, Davide Zambrano, and Sander Bohté

Centrum Wiskunde & Informatica, Amsterdam

Abstract. Spiking neural networks are being investigated both as biologically plausible models of neural computation and also as a potentially more efficient type of neural network. Recurrent neural networks in the form of networks of gating memory cells have been central in state-of-the-art solutions in problem domains that involve sequence recognition or generation. Here, we design an analog Long Short-Term Memory (LSTM) cell where its neurons can be substituted with efficient spiking neurons, where we use subtractive gating (following the subLSTM in [1]) instead of multiplicative gating. Subtractive gating allows for a less sensitive gating mechanism, critical when using spiking neurons. By using fast adapting spiking neurons with a smoothed Rectified Linear Unit (ReLU)-like effective activation function, we show that then an accurate conversion from an analog subLSTM to a continuous-time spiking subLSTM is possible. This architecture results in memory networks that compute very efficiently, with low average firing rates comparable to those in biological neurons, while operating in continuous time.

Keywords: Spiking neurons, LSTM, recurrent neural networks, supervised learning, reinforcement learning.

1 Introduction

With the manifold success of biologically inspired deep neural networks, networks of spiking neurons are being investigated as potential models for computational and energy efficiency. Spiking neural networks mimic the pulse-based communication in biological neurons: in brains, neurons spike only sparingly – on average 1-5 spikes per second [2]. A number of successful convolutional neural networks based on spiking neurons have been reported [3–7], with varying degrees of biological plausibility and efficiency. Still, while spiking neural networks have thus been applied successfully to solve image-recognition tasks, many deep learning algorithms use recurrent neural networks (RNNs), especially variants of Long Short-Term Memory (LSTM) layers [8] to implement dynamic kinds of memory. Compared to convolutional neural networks, LSTMs use memory cells to store select information and various gates to direct the flow of information in and out of the memory cells. The state-changes in such networks are iterative and lack an intrinsic notion of continuous time. To translate LSTMs-like networks into networks, such a notion of time has to be included. At present, the only spike-based version of LSTM has been realized for the IBM TrueNorth platform [9]: this work

proposes an approximate LSTM specifically for TrueNorth's constrains by using a store-and-release mechanism synchronized across its modules, effectively still iterative and synchronized model of computation; Intel recently introduced the first semi-commercial spike-based hardware [10], obviating the need for efficient and effective spiking neural network algorithms. Here, we propose a biologically plausible spiking LSTM network based on an asynchronous approach. While a continuous time model in LSTMs can be implemented by taking small, finite time-steps, a key problem in *spiking* LSTM models is the multiplicative nature of the gating mechanism: such gating requires a graded response from spiking neurons to create a gradient for learning the proper degree of gating. We found that multiplicative gating also needs to be precise, in that noisy gating signal disturbed the learning of memory tasks. We exploit subtractive gating, the "sub-LSTM" [1], to use spiking neurons that effectively compute a fast ReLU function, enabling a spiking subLSTM network to operate in continuous time. We construct a spiking subLSTM network and successfully demonstrate the efficacy of this approach on two standard machine learning tasks: we show that it is indeed possible to use standard analog neurons for the training phase of the modified subLSTM and accurately convert the networks into spiking versions, such that during inference phase spike-based computation is sparse (comparable to active biological neurons) and efficient.

2 Model

To construct a spiking subLSTM network, we first describe the Adaptive Spiking Neurons we aim to use, and we show how we can approximate their effective corresponding activation function. We then show how an LSTM network comprised of a spiking memory cell and a spike-driven input-gate can be constructed and we discuss how analog versions of this subLSTM network are trained and converted to spiking networks.

Adaptive Spiking Neuron. The requirements of the network architectures guide us in the demands put on spiking neuron models. Here, we use Adaptive Spiking Neurons (ASNs) as described in [11]. ASNs are a variant of an adapting Leaky Integrate & Fire (LIF) neuron model that includes fast adaptation to the dynamic range of input signals. The behavior of the ASN is determined by the following equations:

$$\text{incoming postsynaptic current:} \quad I(t) = \sum_i \sum_{t_s^i} w_i \vartheta(t_s^i) \exp\left(\frac{t_s^i - t}{\tau_\beta}\right), \tag{1}$$

$$\text{input signal:} \quad S(t) = (\phi * I)(t), \tag{2}$$

$$\text{threshold:} \quad \vartheta(t) = \vartheta_0 + \sum_{t_s} m_f \vartheta(t_s) \exp\left(\frac{t_s - t}{\tau_\gamma}\right), \tag{3}$$

$$\text{internal state:} \quad \hat{S}(t) = \sum_{t_s} \vartheta(t_s) \exp\left(\frac{t_s - t}{\tau_\eta}\right), \tag{4}$$

where w_i is the weight (synaptic strength) of the neuron's incoming connection; $t_s^i < t$ denote the spike times of neuron i, and $t_s < t$ denote the spike times of the neuron itself; $\phi(t)$ is an exponential smoothing filter with a short time constant τ_ϕ; ϑ_0 is the resting threshold; m_f is a variable controlling the speed of spike-rate adaptation; $\tau_\beta, \tau_\gamma, \tau_\eta$ are the time constants that determine the rate of decay of $I(t), \vartheta(t)$ and $\hat{S}(t)$ respectively. The ASN emits spikes following a firing condition defined as $S(t) - \hat{S}(t) > \frac{\vartheta(t)}{2}$, and, instead of sending binary spikes, the ASNs here communicate with "analog" spikes of which the height is equal to the value of the threshold at the time of firing; note that this model speculatively implies a tight coupling between spike-triggered adaptation and short-term synaptic plasticity (see [12] and [11] for more details).

Activation Function of the Adaptive Analog Neuron. In order to create a network of ASNs that performs correctly on typical LSTM tasks, our approach is to train a network of Adaptive Analog Neurons (AANs) and then convert the resulting analog network into a spiking one, similar to [6, 5, 11]. We define the activation function of the AANs as the function that maps the input signal S to the average PSC I that is perceived by the *next* (receiving) ASN. We then fit the normalized spiking activation function with a softplus-shaped function as:

$$\mathrm{AAN}(S) = a \cdot \log\left(1 + b \cdot \exp(c \cdot S)\right), \tag{5}$$

with derivative:

$$\frac{\mathrm{dAAN}(S)}{\mathrm{d}S} = \frac{a \cdot b \cdot c \cdot \exp(c \cdot S)}{1 + b \cdot \exp(c \cdot S)}, \tag{6}$$

where, for the neuronal parameters used, we find $a = 0.04023$, $b = 1.636$ and $c = 23.54$. Using this mapping from the AAN to the ASN (see Figure 1), the activation function can be used during training of the network with analog AANs: thereafter, the ASNs are used as "drop in" replacements for the AANs. The ASNs use $\tau_\eta = \tau_\beta = \tau_\gamma = 10$ ms, and ϑ_0 and m_f are set to 0.3 and 0.18 for all neurons.

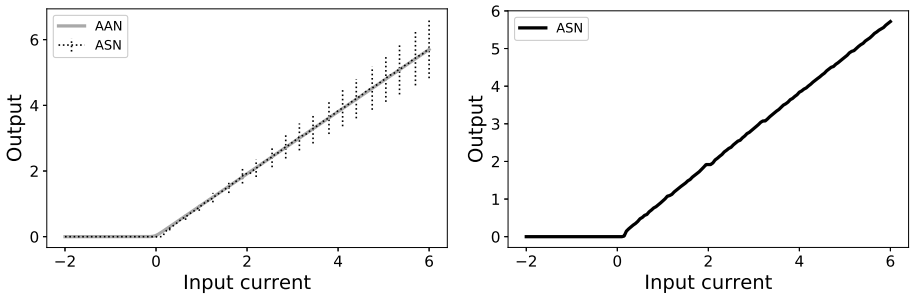

Fig. 1. Left panel: average output signal of the ASN as a function of its incoming PSC I, where the error bars indicate the standard deviation of the spiking simulation, and the corresponding AAN curve. The shape of the ASN curve is well described by the AAN activation function, Equation 5; right panel: the output signal of the ASN alone.

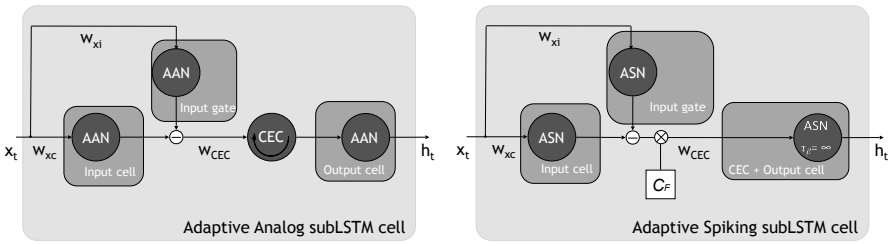

Fig. 2. Overview of the construction of an Adaptive Analog subLSTM and an Adaptive Spiking subLSTM cell. This compares to a subLSTM with only an input gate.

Adaptive Spiking subLSTM. An LSTM cell usually consists of an input and output gate, an input and output cell and a CEC [8]. Deviating from the original formulation and more recent versions where forget gates and peepholes were added [13], the LSTM architecture as we present it here only consists of a (subtractive) input gate, input and output cells, and a CEC. Moreover, the original formulation, an LSTM unit uses a sigmoidal activation function in the input gate and input cell. However, when using spiking neurons, this causes inaccuracies between the analog and spiking network, as, due to the variance in the spike-based approximation, the gates are never completely closed nor completely open. In a recently proposed variation from the original LSTM architecture, called subLSTM [1], the typical multiplicative gating mechanism is substituted with a subtractive one, not requiring thus for the gates to output values exclusively in the range $[0, 1]$. This allows us to use neurons characterized by a smoothed ReLU as activation function. Mathematically, the difference between the integration in the CEC in the LSTM and subLSTM is given as:

$$\text{LSTM:} \quad c_t = c_{t-1} + \mathbf{z}_t \odot \mathbf{i}_t \,, \qquad | \qquad \text{subLSTM:} \quad c_t = c_{t-1} + \mathbf{z}_t - \mathbf{i}_t \,, \tag{7}$$

with c_t value of the memory cell at time t, $\mathbf{z}_t$ and $\mathbf{i}_t$ represent the signal coming from the input cell and the input gate, respectively.

As noted, to obtain a working Adaptive Spiking subLSTM, we first train its analog equivalent, the Adaptive Analog subLSTM. Figure 2 shows the schematic of the Adaptive Analog subLSTM and its spiking analogue: we aim for a one-on-one mapping from the Adaptive Analog subLSTM to the Adaptive Spiking subLSTM. This means that while we train the Adaptive Analog subLSTM network with the standard time step representation, the conversion to the continuous-time spiking domain is achieved by presenting each input for a time window of size Δt, which is determined by the neuronal parameters and by the size of the network. We find that by simply multiplying the signal incoming to the spiking CEC times a conversion factor (i.e. C_F in Figure 2), the two architectures process inputs identically, even if the time component is treated differently.

Spiking Input Gate and Spiking Input Cell. The AAN functions are used in the Adaptive Analog LSTM cell for the input gate, input cell and output

cell. From the activation value of the input cell the activation value of the input gate is subtracted, before it enters the CEC, see Figure 2. Correspondingly, in the spiking version of the input gate, the outgoing signal is subtracted from the spikes that move from the ASN of the input cell to the ASN of the output cell. This leads to a direct mapping from the Adaptive Analog subLSTM to the Adaptive Spiking subLSTM.

Spiking Constant Error Carousel (CEC) and Spiking Output Cell. The Constant Error Carousel (CEC) is the central part of the LSTM cell and avoids the vanishing gradient problem [8]. In the Adaptive Spiking subLSTM, we merge the CEC and the output cell to one ASN with an internal state that does not decay – in the brain could be implemented by slowly decaying (seconds) neurons [14]. The value of the CEC in the Adaptive Analog LSTM corresponds with state I of the ASN output cell in the Adaptive Spiking LSTM. In the Adaptive Spiking subLSTM, we set τ_β in Equation 1 to a very large value for the CEC cell to obtain the integrating behavior of a CEC. Since no forget gate is implemented this results in a spiking CEC neuron that fully integrates its input. When τ_β is set to ∞, every incoming spike is added to a non-decaying PSC I. So if the state of the sending neuron (ASN_in in Figure 3) has a stable inter-spike interval (ISI), then I of the receiving neuron (ASN_out) is increased with incoming spike height h every ISI, so $\frac{h}{\text{ISI}}$ per time step. The same integrating behavior needs to be translated to the analog CEC. Since the CEC cell of the Adaptive Spiking subLSTM integrates its input S every time step by $\frac{S}{\tau_\eta}$, we can map this to the CEC of the Adaptive Analog subLSTM. The CEC of a traditional LSTM without a forget gate is updated every time step by $\text{CEC}(t) = \text{CEC}(t-1) + S$, with S its input value (i.e. $\mathbf{z}_t - \mathbf{i}_t$ for a subtractive LSTM). The CEC of the Adaptive Analog subLSTM is updated every time step by $\text{CEC}(t) = \text{CEC}(t-1) + \frac{S}{\tau_\eta}$. This is depicted in Figure 2 via a weight after the input gate with value $\frac{1}{\tau_\eta}$. To allow a correct continuous-time representation after the spike-coding conversion, we divide the incoming connection weight to the CEC, W_CEC, by the time window Δt. In our approach then, we train the Adaptive Analog subLSTM as for the

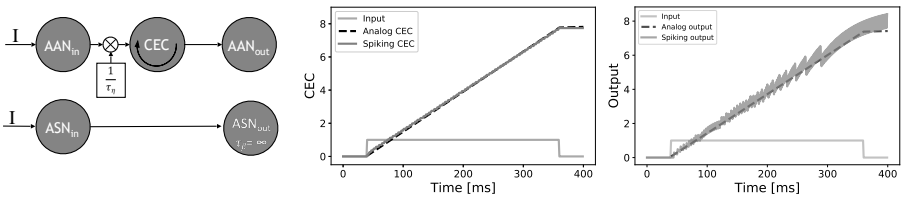

Fig. 3. A simulation to illustrate how the analog CEC integrates its input signal with the same speed as an ASN with $\tau_\beta = \infty$ provided that the input signal does not change and that 1 analog time step corresponds to $\Delta t = 40$ ms (middle). In the right panel, the spiking output signal approximates the analog output.

traditional LSTM (without the τ_η factor), which effectively corresponds to set a continuous-time time window $\Delta t = \tau_\eta$. Thus, to select a different Δt, in the spiking version W_{CEC} has to be set to $W_{\text{CEC}} = \tau_\eta / \Delta t$. The middle plot in Figure 3 shows that setting τ_β to ∞ for ASN_{out} in a spiking network results in the same behavior as using an analog CEC that integrates with $\text{CEC}(t) = \text{CEC}(t-1) + S$, since the slope of the analog CEC is indeed the same as the slope of the spiking CEC. Here, every time step in the analog experiment corresponds to $\Delta t = 40$ ms.

Learning Rule To train the analog subLSTMs on the supervised tasks, a customized truncated version of real-time recurrent learning (RTRL) was used. This is the same algorithm used in [13], where the partial derivatives w.r.t. the weights W_{xc} and W_{xi} (see Figure 2) are truncated. For the reinforcement learning (RL) tasks we used RL-LSTM [15], which uses the same customized, truncated version of RTRL that was used for the supervised tasks. RL-LSTM also incorporates eligibility traces to improve training and Advantage Learning [16]. All regular neurons in the network are trained with traditional backpropagation.

3 Experiments

Since the presented Adaptive Analog subLSTM only has an input gate and no output or forget gate, we present four classical tasks from the LSTM literature that do not rely on these additional gates.

Sequence Prediction with Long Time Lags. The main concept of LSTM, the ability of a CEC to maintain information over long stretches of time, was demonstrated in [8] in a Sequence Prediction task: the network has to predict the next input of a sequence of $p + 1$ possible input symbols denoted as $a_1, ..., a_{p-1}, a_p = x, a_{p+1} = y$. In the *noise free* version of this task, every symbol is represented by the $p + 1$ input units with the $i - th$ unit set to 1 and all the others to 0. At every time step a new input of the sequence is presented. As in the original formulation, we train the network with two possible sequences, $(x, a_1, a_2, ..., a_{p-1}, x)$ and $(y, a_1, a_2, ..., a_{p-1}, y)$, chosen with equal probability. For both sequences the network has to store a representation of the first element in the memory cell for the entire length of the sequence (p). We train 50 networks on this task for a total of $200k$ trials, with $p = 100$, on an architecture with $p+1$ input units and $p + 1$ output units. The input units are fully connected to the output units without a hidden layer. The same sequential network construction method from the original paper was used to prevent the "abuse problem": the Adaptive Analog subLSTM cell is only included in the network after the error stops decreasing [8]. In the *noisy* version of the sequence prediction task, the network still has to predict the next input of the sequence, but the symbols from a_1 to a_{p-1} are presented in random order and the same symbol can occur multiple times. Therefore, only the final symbols a_p and a_{p+1} can be correctly predicted. This version of the sequence prediction task avoids the possibility that

Table 1. Summary of the results. The number of iterations necessary for the network to learn is shown both for the original [8][15] and current implementation. Successfully trained networks (%), ASN accuracy (%) over the number of successfully trained networks, total number of spikes per task and average firing rate (Hz) are also reported.

Task	Orig. Conv. (%)	AAN Conv. (%)	ASN (%)	N_{spikes} (Hz)
Seq. Prediction	5040 (100)	4562 (100)	100	2578 ± 18 (129)
noisy Seq. Prediction	5680 (100)	64428 (100)	100	2241 ± 22 (112)
T-maze	$1M$ (100)	15633 (86)	97	1901 ± 249 (77)
noisy T-Maze	$1.75M$ (100)	20440 (94)	92	1604 ± 216 (65)

the network learns local regularities in the input stream. We train 50 networks with the same architecture and parameters of the previous task, for $200k$ trials.

T-Maze Task. In order to demonstrate the generality of our approach, we trained a network with Adaptive Analog subLSTM cells on a Reinforcement Learning task, originally introduced in [15]. In the T-Maze task, an agent has to move inside a maze to reach a target position in order to be rewarded while maintaining information during the trial. The maze is composed of a long corridor with a T-junction at the end, where the agent has to make a choice based on information presented at the start of the task. The agent receives a reward of 4 if it reaches the target position and -0.2 if it moves against the wall. If it moves to the wrong direction at the T-junction it also receives a reward of -0.2 and the system is reset. The agent has 3 inputs and 4 outputs corresponding to the 4 possible directions it can move to. At the beginning of the task the input can be either 011 or 110 (which indicates on which side of the T-junction the reward is placed). Here, we chose the corridor length $N = 20$. A noiseless and a noisy version of the task were defined: in the noiseless version the corridor is represented as 101, and at the T-junction 010; in a noisy version the input in the corridor is represented as $a0b$ where a and b are two uniformly distributed random variables in a range of $[0, 1]$. While the noiseless version can be learned by LSTM-like networks without input gating [17], the noisy version requires the use of such gates. The network consists of a fully connected hidden layer with 12 AAN units and 3 Adaptive Analog subLSTMs. The same training parameters are used as in [15]; we train 50 networks for each task and all networks have the same architecture. As a convergence criteria we checked whenever the network reached on average a total reward greater than 3.5 in the last 100 trials.

4 Results

As shown in Table 1, for the noise-free and noisy Sequence Prediction tasks all of the networks were both successfully trained and could be converted into spiking networks. The top panels in Figure 4 show the last 5 inputs of a noise-free Sequence Prediction task before (left) and after (right) the conversion, demonstrating the correct predictions made in both cases. In the noisy task, all the successfully trained networks were also still working after the conversion. Finally,

we found that the number of trials needed to reach the convergence criterion were, on average, lower than the one reported in [8] for the noiseless task, while much higher for the noisy task. Both the training and the conversion resulted harder for the T-Maze task, with a few networks non converting correctly into spiking. The bottom panels in Figure 4 show the Q-values of a noisy T-Maze task, demonstrating the correspondence between the analog and spiking representation even in presence of noisy inputs. In general, we see that the spiking CEC value is close to the analog CEC value, while always exhibiting some deviations. Table 1 reports also the average firing rate per neuron, showing reasonably low values compatible with those recorded from real (active) neurons.

5 Discussion

Gating is a crucial ingredient in recurrent neural networks that are able to learn long-range dependencies [8, 18]. Input gates in particular allow memory cells to maintain information over long stretches of time regardless of the presented -

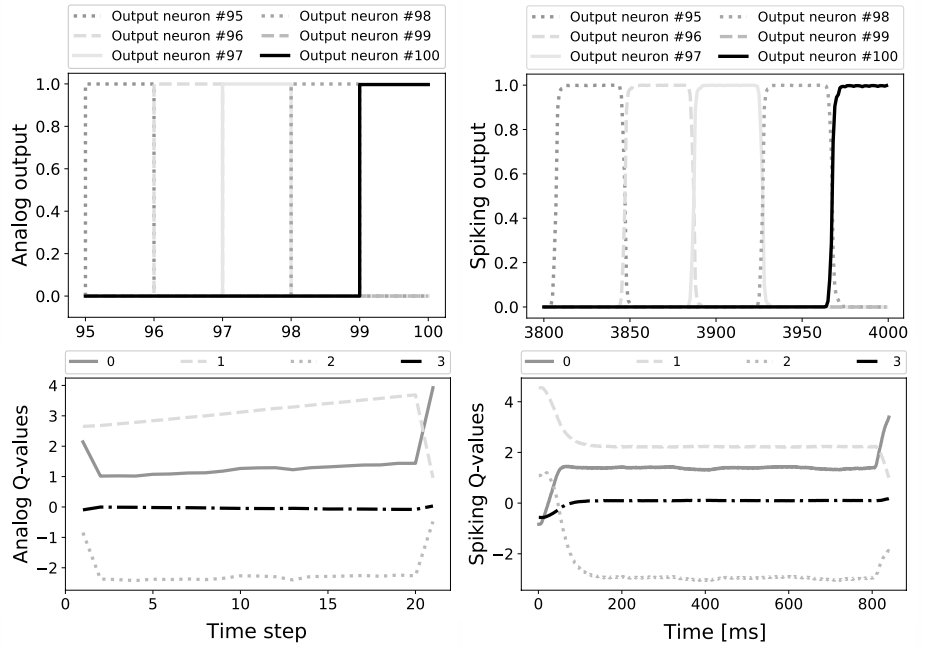

Fig. 4. Top panels: output values of the analog (left) and spiking (right) network for the noise-free Sequence Prediction task. Only the last 5 input symbols of the series are shown. The last symbol y (black) is correctly predicted both in the last time step (analog) and in the last 40 ms (spiking). Bottom panels: Q-values of the analog (left) and spiking (right) network for the noisy T-Maze task. At the last time step/40 msit correctly selects the right action (solid gray line).

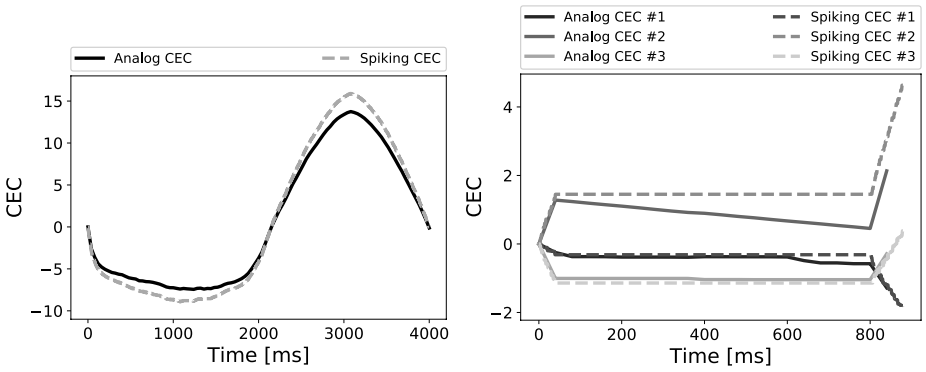

Fig. 5. The values of the analog CECs and spiking CECs for the noise-free Sequence Prediction (left panel) and noisy T-maze (right panel)tasks. The spiking CEC is the internal state $\hat{S}$ of the output cell of the Adaptive Spiking LSTM.

irrelevant - sensory input [8]. The ability to recognize and maintain information for later use is also that which makes gated RNNs like LSTM so successful in the great many sequence-related problems, ranging from natural language processing to learning cognitive tasks [15]. To transfer deep neural networks to networks of spiking neurons, a highly effective method has been to map the transfer function of spiking neurons to analog counterparts and then, once the network has been trained, substitute the analog neurons with spiking neurons [6, 5, 11]. Here, we showed how this approach can be extended to gated memory units, and we demonstrated this for a subLSTM network comprised of an input gate and a CEC. Hence, we effectively obtained a low-firing rate asynchronous subLSTM network which was then shown to be suitable for learning sequence prediction tasks, both in a noise-free and noisy setting, and a standard working memory reinforcement learning task. The learned network could then successfully be mapped to its spiking neural network equivalent for the majority of the trained analog networks. Further experiments will be needed in order to implement other gates and recurrent connections from the output cell of the subLSTM. Although the adaptive spiking LSTM implemented in this paper does not have output gates [8], they can be included by following the same approach used for the input gates: a modulation of the synaptic strength. The reasons for our approach are multiple: first of all, most of the tasks do not really require output gates; moreover, modulating each output synapse independently is less intuitive and biologically plausible than for the input gates. A similar argument can be made for the forget gates, which were not included in the original LSTM formulation: here, the solution consists in modulating the decaying factor of the CEC. It must be mentioned that which gates are really needed in an LSTM network is still an open question, with answers depending on the kind of task to be solved [19, 20].

Acknowledgments. DZ is supported by NWO NAI project 656.000.005.

References

1. Costa, R., Assael, I.A., Shillingford, B., de Freitas, N., Vogels, T.: Cortical microcircuits as gated-recurrent neural networks. In: Advances in Neural Information Processing Systems. pp. 272–283 (2017)
2. Attwell, D., Laughlin, S.: An energy budget for signaling in the grey matter of the brain. J. Cerebral Blood Flow & Metabolism 21(10), 1133–1145 (2001)
3. Esser, S., et al.: Convolutional networks for fast, energy-efficient neuromorphic computing. PNAS p. 201604850 (Sep 2016)
4. Neil, D., Pfeiffer, M., Liu, S.C.: Learning to be efficient: Algorithms for training low-latency, low-compute deep spiking neural networks (2016)
5. Diehl, P., Neil, D., Binas, J., Cook, M., Liu, S.C., Pfeiffer, M.: Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In: IEEE IJCNN. pp. 1–8 (Jul 2015)
6. O'Connor, P., Neil, D., Liu, S.C., Delbruck, T., Pfeiffer, M.: Real-time classification and sensor fusion with a spiking deep belief network. Frontiers in neuroscience 7 (2013)
7. Hunsberger, E., Eliasmith, C.: Spiking deep networks with lif neurons. arXiv preprint arXiv:1510.08829 (2015)
8. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation 9(8), 1735–1780 (1997)
9. Shrestha, A., Ahmed, K., Wang, Y., Widemann, D.P., Moody, A.T., Van Essen, B.C., Qiu, Q.: A spike-based long short-term memory on a neurosynaptic processor
10. Davies, M., Srinivasa, N., Lin, T.H., Chinya, G., Cao, Y., Choday, S.H., Dimou, G., Joshi, P., Imam, N., Jain, S., et al.: Loihi: A neuromorphic manycore processor with on-chip learning. IEEE Micro 38(1), 82–99 (2018)
11. Zambrano, D., Bohte, S.: Fast and efficient asynchronous neural computation with adapting spiking neural networks. arXiv preprint arXiv:1609.02053 (2016)
12. Bohte, S.: Efficient Spike-Coding with Multiplicative Adaptation in a Spike Response Model. In: NIPS 25. pp. 1844–1852 (2012)
13. Gers, F.A., Schraudolph, N.N., Schmidhuber, J.: Learning precise timing with lstm recurrent networks. Journal of machine learning research 3(Aug), 115–143 (2002)
14. Denève, S., Machens, C.K.: Efficient codes and balanced networks. Nature neuroscience 19(3), 375–382 (2016)
15. Bakker, B.: Reinforcement Learning with Long Short-Term Memory. In: NIPS 14. pp. 1475–1482 (2002)
16. Harmon, M., Baird III, L.: Multi-player residual advantage learning with general function approximation. Wright Laboratory pp. 45433–7308 (1996)
17. Rombouts, J., Bohte, S., Roelfsema, P.: Neurally plausible reinforcement learning of working memory tasks. In: NIPS 25. pp. 1871–1879 (2012)
18. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)
19. Greff, K., Srivastava, R.K., Koutník, J., Steunebrink, B.R., Schmidhuber, J.: Lstm: A search space odyssey. IEEE transactions on neural networks and learning systems (2017)
20. Jozefowicz, R., Zaremba, W., Sutskever, I.: An empirical exploration of recurrent network architectures. In: International Conference on Machine Learning. pp. 2342–2350 (2015)