Context Modulation Enables Multi-tasking and Resource Efficiency in Liquid State Machines

Peter Helfer
peter.helfer@utsa.edu
University of Texas at San Antonio
San Antonio, Texas, USA

Corinne Teeter
cmteete@sandia.gov
Sandia National Laboratories
Albuquerque, New Mexico, USA

Aaron Hill
ajhill@sandia.gov
Sandia National Laboratories
Albuquerque, New Mexico, USA

Craig M. Vineyard
cmviney@sandia.gov
Sandia National Laboratories
Albuquerque, New Mexico, USA

James B. Aimone
jbaimon@sandia.gov
Sandia National Laboratories
Albuquerque, New Mexico, USA

Dhireesha Kudithipudi
dk@utsa.edu
University of Texas at San Antonio
San Antonio, Texas, USA

ABSTRACT

Memory storage and retrieval are context-sensitive in both humans and animals; memories are more accurately retrieved in the context where they were acquired, and similar stimuli can elicit different responses in different contexts. Researchers have suggested that such effects may be underpinned by mechanisms that modulate the dynamics of neural circuits in a context-dependent fashion. Based on this idea, we design a mechanism for context-dependent modulation of a liquid state machine, a recurrent spiking artificial neural network. We find that context modulation enables a single network to multitask and requires fewer neurons than when several smaller networks are used to perform the tasks individually.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**.

KEYWORDS

context modulation, neuromorphic, spiking neural network, liquid state machine

ACM Reference Format:
Peter Helfer, Corinne Teeter, Aaron Hill, Craig M. Vineyard, James B. Aimone, and Dhireesha Kudithipudi. 2023. Context Modulation Enables Multi-tasking and Resource Efficiency in Li quid St ate Ma chines. In *International Conference on Neuromorphic Systems (ICONS '23), August 1–3, 2023, Santa Fe, NM, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3589737.3605975

1 INTRODUCTION

Memory storage and retrieval in humans and animals is context-sensitive; memories are more accurately retrieved in the context where they were acquired [4, 18, 24], and similar stimuli may elicit different responses in different contexts [22]. A context may be spatial or temporal or may consist of any collection of cues that serve to distinguish one learning situation from another. The establishment of a context for memory retrieval may be achieved by a physical return to the place of learning or the reinstatement of relevant environmental cues, but can also be effected by subtle reminders [10]. From a neuroscience perspective, it has been suggested that brain regions such as the CA3 subfield of the hippocampus may be modulated by context signals that bias the excitability of neurons, thereby adjusting the way a neural subsystem responds to inputs [1, 22].

Here, we explore potential benefits of context modulation in biologically inspired artificial neural networks. We design a context modulation mechanism for a liquid state machine (LSM), a recurrent spiking architecture [16] that can be deployed on energy-efficient neuromorphic hardware [14, 21, 30]. In analogy with the purported neural processes, our mechanism modulates the firing thresholds of the LSM's spiking neurons, thus altering its dynamics in a context-dependent fashion.

We investigate whether context modulation can improve accuracy, enable multi-tasking and reduce the amount of resources required for a task. For evaluation, we use two publicly available datasets: FSDD, the Free Spoken Digit Dataset [11], and MotionSense, a human activity recognition dataset [17]. We find that context modulation provides only modest accuracy improvements within a single network. However, context modulation does enable a single network to perform multiple tasks and substantially decreases the number of neurons required compared to the case where smaller individual networks perform the same tasks separately. In essence, context modulation allows a network to reuse its computational components for more than one task.

2 PREVIOUS WORK

Work with brain-inspired neural networks falls largely into two categories. The first involves models of hypothetical neural mechanisms, which are built in order to test whether the putative mechanisms can account for phenomena that have been observed in biological systems. Conversely, observation of such models can lead to new hypotheses about biological systems, which can be tested in neuroscience laboratories. For a review of models of contextual modulation of this kind, see [15]. The other category, to which this work belongs, concerns the application of mechanisms observed in biological brains to practical networks, in particular in the field of artificial intelligence. There are several notable examples of the use of contextual modulation to enhance the performance of neural networks. Contextual information was used in [5] to control

ICONS '23, August 1–3, 2023, Santa Fe, NM, USA

ACM ISBN 979-8-4007-0175-7/23/08.
https://doi.org/10.1145/3589737.3605975

where learning takes place in a compartmentalized architecture for few-shot learning. In [31], an online learning model derives its goal from environmental clues and uses it to modulate its attention mechanism toward objects relevant to the goal. A neuromodulatory network in [2] is trained to selectively inhibit neurons in a prediction network in order to mitigate catastrophic forgetting. In [1], the use of superclass information to modulate neuron biases in a convolutional neural network was shown to improve the accuracy on image classification tasks. Previous work involving liquid state machines includes [27–29].

3 MATERIALS AND METHODS

3.1 The Liquid State Machine

The concept of reservoir computing was independently introduced by the publication of two algorithms, the Echo State Network (ESN) [13] and the Liquid State Machine (LSM) [16]. The two architectures differ in that the ESN uses conventional (rate-based) neurons in the reservoir, whereas the LSM uses spiking neurons, but are otherwise very similar. Here, we focus on the LSM, the spiking variety of reservoir computing.

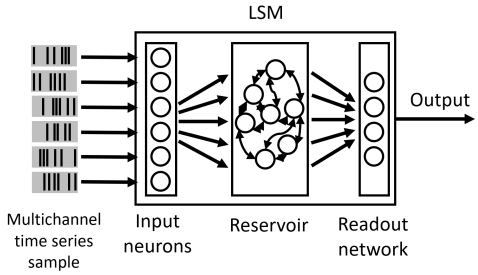

Figure 1: Liquid state machine (LSM). An LSM consists of an input layer, a recurrent spiking reservoir, and a readout network. Connections from input to reservoir and between reservoir neurons are sparse, randomly initialized and fixed; only output connection weights are trained. Samples of multichannel time series data are fed to the input neurons, which project the received values onto the reservoir, resulting in spiking activity. Once a sample has been fully processed, the readout network classifies the resulting reservoir state and outputs its predicted label for the sample.

The core idea of an LSM is to cast input data into a much higher dimensional representation with the hope of improving class separability of the data. Then, a readout network is trained to classify on the higher dimensional representation. This is implemented by projecting time-series data onto a recurrently connected set of spiking neurons (the *reservoir* or *liquid*), then read the state of the reservoir and classify it using a simple feedforward *readout network*, which is fully connected to the reservoir; see Figure 1.

The connections from the input neurons to the reservoir neurons, as well as the connections between reservoir neurons, are sparse and randomly initialized, but thereafter never change during training or inference. Learning occurs only in the readout connections. In this manner, the readout network learns to discern patterns of activity in the liquid to differentiate between how different inputs drive different activity dynamics.

The reservoir consists of leaky-integrate-and-fire (LIF) neurons whose dynamics are defined by the following equation:

$$V_i(t+1) = V_i(t) + \frac{\Delta t}{\tau_{mem}}\big(-V_i(t) + I_i(t)R\big), \tag{1}$$

where $V_i(t)$ is the membrane potential of neuron i at time t, $I_i(t)$ is the input current to neuron i, R is the membrane resistance, and τ_{mem} is the membrane time constant. Δt is the length of a time step.

The input current to neuron i is the sum of the incoming currents from other neurons:

$$I_i(t) = \sum_{j=1}^{N_{inp}} w_{ij}A_j(t) + \sum_{k=1}^{N_{res}} w_{ik}S_k(t). \tag{2}$$

w_{ij} is the connection weight from input neuron j to reservoir neuron i, A_j is the activation level of input neuron j, w_{ik} is the connection weight from reservoir neuron k to reservoir neuron i, and S_k is 1 if reservoir neuron k spikes at the current time step, otherwise 0. N_{inp} and N_{res} are the numbers of input and reservoir neurons, respectively.

When a neuron's membrane potential crosses a threshold V_{thresh}, a spike is emitted ($S_i = 1$) and the membrane potential is reset to zero ($V_i = 0$). After a neuron spikes, there is a short time interval during which it cannot spike; this is known as the refractory period.

Each reservoir neuron i is equipped with an "x-trace", X_i, a leaky integrator that serves as a decaying memory of the neuron's spiking activity:

$$X_i(t+1) = X_i(t)\Big(1 - \frac{\Delta t}{\tau_{xtrace}}\Big) + S_i(t), \tag{3}$$

where τ_{xtrace} defines the decay rate and $S_i(t)$ is 1 if neuron i spikes at time step t, 0 otherwise.

The readout layer consists of sigmoid neurons:

$$f_i(t) = \frac{1}{1 - e^{-Y_i(t)}}, \tag{4}$$

where $f_i(t)$ is readout neuron i's activation level at time t, and $Y_i(t)$ is its instantaneous input, a weighted sum of the x-trace values:

$$Y_i(t) = \sum_{j=1}^{N_{res}} w_{ij}X_j(t), \tag{5}$$

where w_{ij} is the connection weight from x-trace j to output neuron i. The readout network is trained to map input samples to one-hot encodings of the corresponding labels, i.e., the index of the readout neuron with the highest activation level is the network's prediction for the label:

$$pred = \underset{i}{\operatorname{argmax}}\, f_i(t). \tag{6}$$

3.2 Context Modulation

The central contribution of the present work is the introduction of a mechanism for context-dependent modulation of the LSM's reservoir. The idea is to bias the reservoir neurons' firing thresholds so that, depending on the current context, different subpopulations of the reservoir are more or less prone to fire. This can be thought of as remodeling the reservoir's "energy landscape" so that spiking activity is directed to different parts of the reservoir depending on which context is active, with the aim of improving pattern separation and facilitating classification.

To implement this mechanism, we replace the LSM's globally defined firing threshold V_{thresh} with a neuron-specific firing threshold $vthresh_i$, and the condition for spiking becomes:

$$S_i(t) = \begin{cases} 1, & \text{if } V_i(t) >= vthresh_i \\ 0, & \text{otherwise,} \end{cases} \tag{7}$$

where $S_i(t)$ indicates whether neuron i is firing and $V_i(t)$ is neuron i's membrane potential at time t.

Whenever a context is activated, the neuron-specific firing thresholds $vthresh_i$ are set to a set of values specific to that context. Those values are determined as follows:

Let N_{ctx} be the number of contexts for a given classification task. We identify the different contexts with integer IDs $c = 0, 1, 2, ..., N_{ctx} - 1$.

For each context ID c, we create a unique vector of firing threshold biases with length N_{res}, the number of neurons in the reservoir. So we have an array of biases, $bias_{ci}$, with N_{ctx} rows, one per context ID, and each row is a vector of N_{res} bias values, one for each reservoir neuron. When a particular context, say context c, is activated, we set the firing threshold for each reservoir neuron i to a base threshold $vthresh_{base}$ plus the value of the corresponding element of the bias vector for context c:

$$vthresh_i = vthresh_{base} + bias_{ci} \tag{8}$$

To achieve a smooth variation of biases, the bias vectors are created by randomly permuting a template vector $templ$ that is initialized with a Gaussian distribution of bias values:

$$templ_i = max_{bias} * e^{-\frac{k_{bias}}{N_{res}}(i-\frac{N_{res}}{2})^2}, \tag{9}$$

where max_{bias} and k_{bias} are configurable parameters.

During training, the context ID for each sample is supplied from the environment. During testing, context IDs may similarly be supplied to the network ("known context mode"), simulating a scenario where each test sample is presented in the same context where it was learned (Figure 2 B). In an alternate scenario, context IDs are provided during training, but not during testing. For this case, an auxiliary context identifier LSM is trained to infer context IDs from samples, and the inferred context ID is used to modulate the network during testing ("inferred context mode"), see Figure 2 C. Using speech recognition as an example, the first scenario may correspond to a situation where the speaker can be identified independently of the audio signal, e.g. visually, whereas in the second scenario no such extra information is available, but accuracy may be improved if the listener can identify the voice of the speaker.

3.3 Datasets

The *FSDD* dataset [12] consists of 3000 voice recordings of six speakers pronouncing the digits zero through nine in English. We preprocess the recordings into standard 13 MEL frequency cepstral coefficients (MFCCs). See Section 3.4 for further details.

The *MotionSense* human activity recognition dataset [17] consists of recordings from a smartphone's acceleration, attitude, and gyroscope sensors when worn by 24 participants engaged in any of 6 activities (sitting, standing, walking, jogging, walking upstairs, and walking downstairs). There are 216 recordings designated for training and 144 for testing. The lengths of the recordings vary considerably; to obtain a uniform dataset, we split the recordings into five-second samples, resulting in 4214 training samples and 1247 test samples, labeled with activity type.

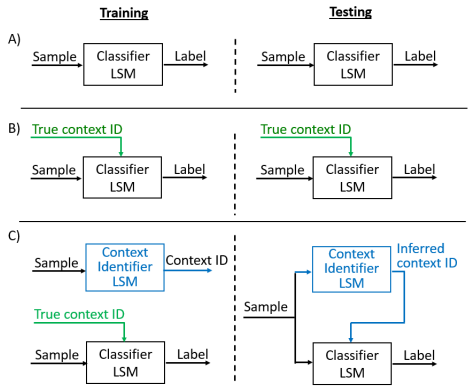

Figure 2: Context modulation of LSM. Scenario A: Without context modulation; Scenario B "Known context": As each sample is trained, an associated ("true") context ID (green) is received from the environment and is used to modulate the LSM. During testing, as each sample is being classified, the LSM is modulated using the same context ID with which the sample was trained; Scenario C "Inferred context": During training, an auxiliary LSM (blue) is trained to infer context IDs from the samples, using the true context IDs as targets. During testing, the true context IDs are unavailable; context IDs inferred by the context identifier LSM are used to modulate the main (classifier) LSM.

3.4 Training and Testing

The LSM's performance on a dataset is evaluated by executing a series of train/test cycles. For the FSDD dataset, each cycle consists of the following steps:

(1) The dataset, consisting of 3000 labeled samples, is randomly split into a training set (2700 samples) and a test set (300 samples).
(2) The LSM is then trained on the training set for 50 epochs. The order of the training samples is randomized for each epoch. Each sample is processed through the LSM as described in Section 3.1, whereupon the readout weights are updated using gradient descent (online training).
(3) The accuracy of the trained LSM is then tested by processing the test samples and calculating the proportion of correctly labeled samples.

Each accuracy value reported in the results section was calculated by executing ten train/test cycles and taking the mean and standard deviation of the test accuracies.

The procedure is the same for the MotionSense dataset, except that a) the training and test datasets are predefined, so there is no random splitting into train/test samples, and b) the number of samples is 4214 for training and 1247 for testing.

3.5 Optimization of the LSM

The configuration of an LSM is controlled by a number of hyperparameters. Some hyperparameters directly control attributes of network elements, for example τ_{mem}, the membrane time constant for the LIF neurons; others are used to parameterize the random initialization of the LSM. For example, the probability of a connection between any two reservoir neurons is $C \cdot e^{-(D/\lambda)^2}$, where D is the distance between the two neurons and C and λ are hyperparameters [16]. We use a genetic algorithm (GA) to find a good set of hyperparameter values for a classification task, using a train/test cycle as defined above to evaluate the fitness of a set of parameter values. However, the random initialization still leaves room for considerable variation in accuracy between LSMs configured with the same set of hyperparameter values. The problem of finding a "good reservoir" has been discussed in the literature and several approaches have been suggested [9, 19, 25]. Here we use a simple heuristic: We repeatedly (e.g. 50 or 100 times) instantiate and initialize LSMs using the optimized set of hyperparameters, execute a single train/test cycle with each such randomly initialized LSM instance on the task, and select the instance that achieves the highest accuracy. This means that, when evaluating an LSM's performance for a given task, we use the best set of input and reservoir connections that we have found for that task, together with a fixed set of threshold bias vectors (when using context modulation). The weights in the readout network are still randomly initialized and trained in each train/test cycle.

4 RESULTS

4.1 Context Modulation Improves Accuracy

We start our investigation by asking if context modulation can improve the classification accuracy within a single network. We envision potential applications in two different scenarios (see Section 3.2 and Figure 2). In the first scenario, the context signal is *known* and is supplied by the environment both during training (memory acquisition) and during testing (recall). This is likely the best case scenario and characterizes how well a network can perform in a situation where context is unambiguous. In the second scenario, we assume that the context signal is available during training, but not during testing. In this case, an auxiliary LSM is trained to learn the context. The *inferred* context signal is then provided to the main LSM during testing. Unless the auxiliary network learns to infer context perfectly, the context signal will contain some noise and likely limit the benefits of context modulation. We look at the "inferred context" scenario because it is likely there are many real-world applications where the context signal is not available during inference. We wanted to get a rough idea of how much realistic errors in a context signal would affect classification accuracy.

As shown in Figure 3, context modulation modestly improves classification accuracy on both the FSDD and MotionSense datasets.

When we trained a single LSM to identify spoken digits in the FSDD dataset without a context signal, we achieved an accuracy of 0.958 ± 0.012. We then applied context modulation to the LSM by using the speaker as a context signal. Using the inferred speaker IDs resulted in an accuracy of 0.969 ± 0.014, a 1.1% improvement. Using known speaker IDs resulted in an accuracy of 0.973 ± 0.010, a 1.5% improvement over baseline. The difference in accuracy between the inferred-context and known-context modes is due to the error in speaker inference, which was 98.2% accurate.

We see similar results when applying context modulation to the MotionSense dataset. Here participant ID (0-23) is used as the context signal and activity type (one of six) is used for the classification target. With this dataset, accuracy without context modulation was 0.946 ± 0.003, with inferred participant ID used as context signal, 0.948 ± 0.002, a 0.7% improvement, and with known participant ID, 0.954 ± 0.002, a 0.8% improvement over the baseline.

Although these improvements are small, they do show that our context modulation technique can aid in classification on two separate datasets.

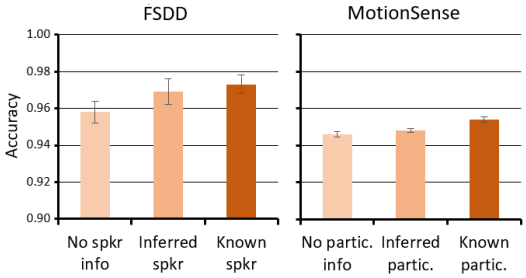

Figure 3: Context modulation improves accuracy. Left: digit classification with FSDD using speaker ID as context signal. Right: activity classification with MotionSense using participant ID as context signal. The error bars indicate standard deviation over ten train/test cycles of each test case. The smaller variability with MotionSense is due to the fact that train and test subsets are predefined in the dataset; only sample ordering varies between train/test cycles.

4.2 Context Modulation Reduces Network Size

Above we showed that a context signal can improve the ability of a network to perform classification. The best results were achieved when the context signal was available both during training and testing. This raises the question: If context is known, why perform classification in different contexts on one network? Why not use separate networks for each context? Here we use the FSDD dataset to show that using one network with a context signal reduces the number of neurons needed to reach equivalent accuracy between individual networks and a context network.

We trained six separate LSMs to each classify samples from one of the six speakers. The mean accuracy for individual speakers varied between 0.974 and 0.992 depending on speaker, with an overall mean of 0.986 ± 0.017 (see Figure 4). The accuracy of a single LSM with context modulation is somewhat lower than the mean accuracy for single-speaker networks. As also reported in Section 4.1, when we trained a single LSM on the complete FSDD dataset without context, we achieved 0.958 ± 0.012 accuracy, 2.8% below the mean single-speaker performance. Context modulation improved the accuracy to 0.969 ± 0.014 when using inferred speaker IDs, 1.7% below mean single-speaker accuracy and to 0.973 ± 0.010 using known speaker IDs, 1.3% below mean single-speaker accuracy.

Although using a single LSM with context modulation does not improve accuracy over using multiple individual networks, it does substantially reduce the number of neurons needed to achieve high accuracy. The accuracy values shown at the top of Figure 4 were

obtained using a cube-shaped reservoir with $10^3 = 1000$ neurons for each network. Thus, although individual networks achieve higher average accuracy, more neurons are used to achieve this result (6*1000 neurons using multiple individual networks versus 1000 neurons for one network with context modulation). To quantify how large individual networks need to be to achieve high accuracy, we decrease the size of the networks (bottom panel of Figure 4). The size of the individual-speaker digit classification networks can be reduced to 343 neurons and still achieve the accuracy obtained by the single 1000-neuron LSM with context modulation. Although 343 is less than 1000, six networks are required. Using context modulation to handle all six speakers with a single 1000-neuron LSM thus resulted in an overall reduction of reservoir size by 51.4% (1000 vs. $6 * 343$). Thus, the use of individual networks comes at the cost of a larger combined network size. Interestingly, these results also show that the context-modulated network does not simply use separate subsets of neurons while in different contexts; instead, individual neurons contribute to more than one context.

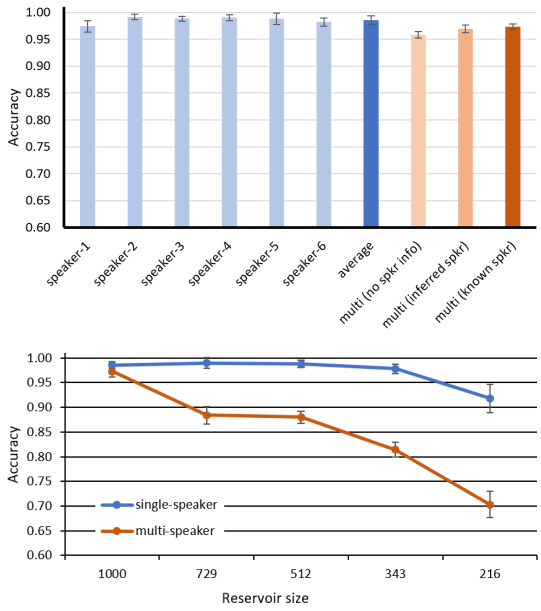

Figure 4: Top: Single-speaker vs. multi-speaker accuracy. The first six columns show accuracy achieved by LSMs classifying FSDD samples spoken by each individual speaker and the seventh column shows the average across speakers. The last three columns indicate the accuracy of a single LSM classifying the complete FSDD dataset (all six speakers), without context modulation, with inferred speaker ID as context signal, and with known speaker ID, respectively. Note that the last three columns are the same as in Figure 3. Bottom: Accuracy versus reservoir size for single-speaker and multi-speaker FSDD digit recognition. The error bars in both panels indicate standard deviation over ten train/test cycles for each test case.

4.3 Context Modulation Enables Multiple Representations Within a Single LSM

Our biological brains are capable of applying different classifications to the same objects depending on context. For example, in the classic Terminator movie series, featuring Arnold Schwarzenegger, Arnold is a "bad guy" in the first movie and a "good guy" in the second movie. Humans can easily classify whether Arnold is good or bad based on context (whether they are watching the first or second movie). To explore if context-modulated LSMs are capable of this type of behavior, we devised an experiment where identical data had to be classified differently depending on context.

We assigned each of the six speakers in the FSDD dataset to one of two groups, A or B, and labeled each speech sample with its speaker's group ID. As a "baseline" group assignment, we assigned the first speaker to group A, the second speaker to group B, etc.: ABABAB. The LSM was trained to classify the speech samples according to group labels (withholding speaker IDs and digit classes), achieving an accuracy of 0.931 ± 0.014.

We then created modified speaker-to-group mappings that differed from the baseline in one, two, three, four, five or all six positions: BBABAB, BAABAB, BABBAB, BABAAB, BABABB, BABABA. The LSM was trained with a mix of samples labeled either according to the baseline mapping or according to one of the modified mappings. As in the digit recognition task, we randomly split the dataset into 90% training samples and 10% test samples for each train/test cycle. During both training and testing, a context signal was supplied, indicating which mapping was in effect ("known context"). As shown in Figure 5, context modulation enabled the simultaneous learning of both mappings in the same LSM with little or no accuracy loss even when as many as four of the six speakers had different group assignments in the two mappings. The same reservoir size (1000 neurons) was used as in the previous experiments.

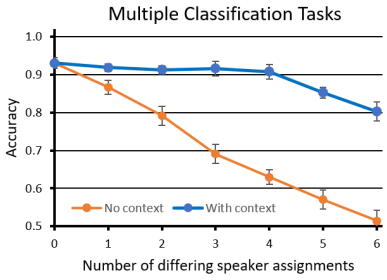

Figure 5: FSDD speaker group identification. An LSM was trained to simultaneously learn two different classifications of the FSDD dataset. Samples were classified according to speakers' group memberships. The x-axis indicates how many of the six FSDD speakers had different group assignments in the two mappings. Accuracy with and without context modulation is shown. The plots show mean accuracy over ten runs with random train/test splits of the dataset; error bars indicate standard deviation.

4.4 Resource Usage and Energy Analysis

Because an LSM's input-to-reservoir connections and intra-reservoir connections are sparse and fixed, its model size and energy consumption compare favorably to other recurrent networks. To illustrate this point, we estimate the number of compute operations and the energy requirements for our LSM implementation compared with an LSTM network with comparable performance on the FSDD task [23]. See Appendix A for details.

Table 1 compares the total number of arithmetic operations during a forward pass and the number of weight updates per training iteration, as well as memory requirements.

Table 2 compares energy consumption for the same networks, estimated for 32-bit and 16-bit floating-point operations. We also include 8-bit integer operations, which are supported in hardware by several recent ML accelerators.

As seen in the tables, the resource requirements for the LSM are considerably lower than for the LSTM: 85% smaller memory footprint and 86% lower energy cost. Even when including the auxiliary context inference network, the memory size is 71% smaller and the energy cost 71% lower than for the LSTM.

Network	# Ops (FP)	# Weight Updates	Model size (Mb)
LSM without CI	1,195,620	10,010	2.3
LSM with CI	2,383,232	16,016	4.6
LSTM	8,536,020	4,270,010	16.3

Ops: Synaptic operations; FP: Forward Pass; CI: Context inference network

Table 1: Comparison of the number of operations (additions and multiplications) in a forward pass, the number of weight updates per training iteration, and memory usage for the LSM networks applied to FSDD as described in this paper, and also for an LSTM network with comparable performance. The LSM without context inference network (CI) is used in the "no context" and "known context" scenarios, and the LSM with CI is used in the "inferred context" scenario. The model sizes are calculated for 32-bit floating-point representation. See Appendix A for calculations.

Network	Energy(32-bit FP)	Energy(16-bit FP)	Energy (8-bit INT)
LSM without CI	75.3×10^{-6} J	25.1×10^{-6} J	6.0×10^{-6} J
LSM with CI	150.1×10^{-6} J	50.0×10^{-6} J	11.9×10^{-6} J
LSTM	537.5×10^{-6} J	179.3×10^{-6} J	42.7×10^{-6} J

FP: Floating Point; INT: Integer

Table 2: Estimated computational energy requirements for each of the networks when realized with either FP-32, FP-16, or INT-8 operations [26]. See Appendix A for calculations.

Table 3 compares our model's accuracy on the FSDD and MotionSense tasks with the best-performing previously published LSM implementation and state-of-the-art non-spiking networks.

Model	Network	Task	Accuracy
Kang [14]	LSM	FSDD	0.930
Ours	LSM	FSDD	0.958 ± 0.012
Ours	LSM with CI	FSDD	0.969 ± 0.014
Reddy [23]	CNN	FSDD	**0.987**
Haresamudram [8]	CPC	MotionSense	0.918 ± 0.22
Ours	LSM	MotionSense	0.946 ± 0.003
Ours	LSM with CI	MotionSense	0.948 ± 0.002
Bernau [3]	LSTM/CNN	MotionSense	**0.99**

Table 3: Performance on the FSDD and MotionSense tasks.

5 DISCUSSION

We designed a mechanism for modulating the dynamics of an LSM in a context-dependent manner. We show that context modulation can improve classification accuracy, reduce resources needed to perform classification, and enable an LSM to classify objects differently based on context.

For both datasets, the classification accuracy improvements are small. There are several possible explanations for why we observe minimal improvements. Here, we have chosen context signals based on obvious characteristics of the dataset. For example, in the FSDD dataset, we chose to use speaker ID as a context signal, not because we had reason to believe it would be particularly informative, but because it was available in the dataset. Perhaps there are less obvious contexts that would yield better results. This work would benefit from research aimed at understanding how to identify a good context signal and determine whether datasets have properties that make them more amenable to improvements in classification accuracy from context modulation. In addition, classification of the datasets used here is already quite good without context modulation. It remains to be seen whether context modulation will make more of a difference for datasets where the baseline classification accuracy is lower.

A perhaps more interesting result is that context modulation may enable more efficient use of resources. As shown with the FSDD dataset, although the single context-modulated network showed slightly lower accuracy than the average across the single-speaker networks, the number of neurons needed to perform classification was reduced by approximately half. It will be interesting to see if this result holds for different datasets and how this scales to larger datasets. This result also implies that our context modulation mechanism does not simply divide the network into independent subpopulations for classification. Rather, it seems that neurons are reused between contexts. An interesting line of inquiry would be to understand how the dynamics of the reservoir differs between contexts, and further, to identify what aspects of this change can lead to resource conservation.

Another interesting result is that context modulation can enable multiple classifications of the same samples depending on context. In essence this also illustrates how the use of one network with context modulation enables efficient resource use. The alternative would be to have separate networks performing different classification tasks according to the context.

In addition to the efficiency benefits of context modulation as described above, LSMs are spiking networks and can be implemented on power-efficient neuromorphic hardware, further amplifying their energy advantage [6, 7]. On the other hand, our LSM implementation does not quite match the accuracy of state-of-the-art non-spiking networks; this is an area for further improvement.

In conclusion, this preliminary work provides evidence that context modulation can enable more efficient resource use in LSMs and can also enable different data classifications in the same network. It is likely that context modulation will become a powerful tool to reduce the resources needed in a variety of artificial intelligence applications.

ACKNOWLEDGMENTS

This work was supported by the Laboratory Directed Research and Development program at Sandia National Laboratories, a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. The authors own all right, title and interest in and to the article and is solely responsible for its contents. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this article or allow others to do so, for United States Government purposes. The DOE will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan https://www.energy.gov/downloads/doe-public-access-plan. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government. AUTHOR CONTRIBUTIONS: PH wrote the paper, conceived and performed experiments. CT principle investigator, conceived/consulted on experiments, edited manuscript. AH conceived and consulted on experiments. DK is part of initial ideation of the LSMs for context modulation with JBA, consulted on experiments, and edited manuscript. JBA, CV, AH, conceived of the project. All authors provided feedback on the manuscript. Thanks to Truman Hickok for helpful discussions and to Anurag Daram for helping with the compute/memory analysis.

REFERENCES

[1] James B. Aimone and William M. Severa. 2017. Context-Modulation of Hippocampal Dynamics and Deep Convolutional Networks. *arXiv:1711.09876 [cs, q-bio, stat]* (Nov. 2017). arXiv:1711.09876 [cs, q-bio, stat]

[2] Shawn Beaulieu, Lapo Frati, Thomas Miconi, Joel Lehman, Kenneth O. Stanley, Jeff Clune, and Nick Cheney. 2020. Learning to Continually Learn. *arXiv:2002.09571 [cs, stat]* (March 2020). arXiv:2002.09571 [cs, stat]

[3] Daniel Bernau, Jonas Robl, and Florian Kerschbaum. 2022. Assessing Differentially Private Variational Autoencoders under Membership Inference. arXiv:2204.07877 [cs]

[4] M. E. Bouton. 2004. Context and Behavioral Processes in Extinction. *Learning & Memory* 11, 5 (Sept. 2004), 485–494. https://doi.org/10.1101/lm.78804

[5] Anurag Daram, Angel Yanguas-Gil, and Dhireesha Kudithipudi. 2020. Exploring Neuromodulation for Dynamic Learning. *Frontiers in Neuroscience* 14 (2020).

[6] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, ..., and Hong Wang. 2018. Loihi: A Neuromorphic Manycore Processor with On-Chip Learning. *IEEE Micro* 38, 1 (Jan. 2018), 82–99. https://doi.org/10.1109/MM.2018.112130359

[7] Michael V. DeBole, Brian Taba, Arnon Amir, Filipp Akopyan, ..., and John V. Arthur. 2019. TrueNorth: Accelerating From Zero to 64 Million Neurons in 10 Years. *Computer* 52, 5 (May 2019), 20–29. https://doi.org/10.1109/MC.2019.2903009

[8] Harish Haresamudram, Irfan Essa, and Thomas Ploetz. 2023. Towards Learning Discrete Representations via Self-Supervision for Wearables-Based Human Activity Recognition. arXiv:2306.01108 [cs, eess]

[9] Emmanouil Hourdakis and Panos Trahanias. 2013. Use of the Separation Property to Derive Liquid State Machines with Enhanced Classification Performance. *Neurocomputing* 107 (May 2013), 40–48. https://doi.org/10.1016/j.neucom.2012.07.032

[10] Almut Hupbach, Rebecca Gomez, Oliver Hardt, and Lynn Nadel. 2007. Reconsolidation of Episodic Memories: A Subtle Reminder Triggers Integration of New Information. *Learning & Memory* 14, 1-2 (Jan. 2007), 47–53. https://doi.org/10.1101/lm.365707

[11] Zohar Jackson. 2020. Free Spoken Digit Dataset (FSDD) v1.0.10. https://doi.org/10.5281/zenodo.1342401

[12] Zohar Jackson, César Souza, Jason Flaks, Yuxin Pan, Hereman Nicolas, and Adhish Thite. 2018. Jakobovski/Free-Spoken-Digit-Dataset: V1.0.8. Zenodo. https://doi.org/10.5281/zenodo.1342401

[13] Herbert Jaeger. 2001. The "Echo State" Approach to Analysing and Training Recurrent Neural Networks-with an Erratum Note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report* 148, 34 (2001), 13.

[14] Ziyang Kang, Shiying Wang, Lei Wang, Shiming Li, Lianhua Qu, and Weixia Xu. 2022. Hardware-Aware Liquid State Machine Generation for 2D/3D Network-on-Chip Platforms. *Journal of Systems Architecture* 124 (March 2022), 102429. https://doi.org/10.1016/j.sysarc.2022.102429

[15] Dhireesha Kudithipudi, Mario Aguilar-Simon, Jonathan Babb, Maxim Bazhenov, ..., and Hava Siegelmann. 2022. Biological Underpinnings for Lifelong Learning Machines. *Nature Machine Intelligence* 4, 3 (March 2022), 196–210. https://doi.org/10.1038/s42256-022-00452-0

[16] Wolfgang Maass, Thomas Natschläger, and Henry Markram. 2002. Real-Time Computing Without Stable States: A New Framework for Neural Computation Based on Perturbations. *Neural Computation* 14, 11 (Nov. 2002), 2531–2560. https://doi.org/10.1162/089976602760407955

[17] Mohammad Malekzadeh, Richard G. Clegg, Andrea Cavallaro, and Hamed Haddadi. 2018. Protecting Sensory Data against Sensitive Inferences. *Proceedings of the 1st Workshop on Privacy by Design in Distributed Systems* (April 2018), 1–6. https://doi.org/10.1145/3195258.3195260 arXiv:1802.07802

[18] M. R. Milad, S. P. Orr, R. K. Pitman, and S. L. Rauch. 2005. Context Modulation of Memory for Fear Extinction in Humans. *Psychophysiology* 42, 4 (July 2005), 456–464. https://doi.org/10.1111/j.1469-8986.2005.00302.x

[19] David Norton and Dan Ventura. 2010. Improving Liquid State Machines through Iterative Refinement of the Reservoir. *Neurocomputing* 73, 16 (Oct. 2010), 2893–2904. https://doi.org/10.1016/j.neucom.2010.08.005

[20] Christopher Olah. 2015. Understanding LSTM Networks – Colah's Blog. https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[21] Alberto Patiño-Saucedo, Horacio Rostro-González, Teresa Serrano-Gotarredona, and Bernabé Linares-Barranco. 2022. Liquid State Machine on SpiNNaker for Spatio-Temporal Classification Tasks. *Frontiers in Neuroscience* 16 (2022).

[22] Praveen K. Pilly, Michael D. Howard, and Rajan Bhattacharyya. 2018. Modeling Contextual Modulation of Memory Associations in the Hippocampus. *Frontiers in Human Neuroscience* 12 (Nov. 2018), 442. https://doi.org/10.3389/fnhum.2018.00442

[23] Sainath Reddy. 2022. Spoken_Digit_Recognition Using CNN and LSTM. https://kaggle.com/code/sainathrk/spoken-digit-recognition-using-cnn-and-lstm.

[24] William A. Roberts. 2019. The Role of Context in Animal Memory. *Learning & Behavior* 47, 2 (June 2019), 117–130. https://doi.org/10.3758/s13420-019-00380-x

[25] Subhrajit Roy and Arindam Basu. 2016. An Online Structural Plasticity Rule for Generating Better Reservoirs. *Neural Computation* 28, 11 (Nov. 2016), 2557–2584. https://doi.org/10.1162/NECO_a_00886

[26] Sadasivan Shankar and Albert Reuther. 2022. Trends in Energy Estimates for Computing in AI/Machine Learning Accelerators, Supercomputers, and Compute-Intensive Applications. In *2022 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, Waltham, MA, USA, 1–8. https://doi.org/10.1109/HPEC55821.2022.9926296

[27] Nicholas Soures and Dhireesha Kudithipudi. 2019. Deep Liquid State Machines With Neural Plasticity for Video Activity Recognition. *Frontiers in Neuroscience* 13 (July 2019), 686. https://doi.org/10.3389/fnins.2019.00686

[28] Nicholas Soures and Dhireesha Kudithipudi. 2019. Spiking Reservoir Networks: Brain-Inspired Recurrent Algorithms That Use Random, Fixed Synaptic Strengths. *IEEE Signal Processing Magazine* 36, 6 (Nov. 2019), 78–87. https://doi.org/10.1109/MSP.2019.2931479

[29] Nicholas Soures, Cory Merkel, Dhireesha Kudithipudi, Clare Thiem, and Nathan McDonald. 2017. Reservoir Computing in Embedded Systems Three Variants of the Reservoir Algorithm. *Ieee Consumer Electronics Magazine* 6, 3 (July 2017),

67–73. https://doi.org/10.1109/MCE.2017.2685159
[30] Lei Wang, Zhijie Yang, Shasha Guo, Lianhua Qu, Xiangyu Zhang, Ziyang Kang, and Weixia Xu. 2022. LSMCore: A 69k-Synapse/Mm(2) Single-Core Digital Neuromorphic Processor for Liquid State Machine. *Ieee Transactions on Circuits and Systems I-Regular Papers* 69, 5 (May 2022), 1976–1989. https://doi.org/10.1109/TCSI.2022.3147380
[31] Xinyun Zou, Soheil Kolouri, Praveen K. Pilly, and Jeffrey L. Krichmar. 2020. Neuromodulated Attention and Goal-Driven Perception in Uncertain Domains. *Neural Networks* 125 (May 2020), 56–69. https://doi.org/10.1016/j.neunet.2020.01.031

A MODEL SIZE AND ENERGY ESTIMATES

A.1 LSM

The number of neurons in the LSM network is $N_n = N_{inp} + N_{res} + N_{out}$. The number of synapses is $N_s = N_{inp} * N_{res} * D_{inp} + {N_{res}}^2 * D_{res} + N_{res} * N_{out}$, where $D_{inp} = 0.6$ and $D_{res} = 0.58$ are the fraction of nonzero input-to-reservoir and intra-reservoir connections, respectively. For the FSDD task, the digit classifier LSM has $N_{inp} = 13$, $N_{res} = 1000$, and $N_{out} = 10$, and the context identifier (CI) has $N_{inp} = 13$, $N_{res} = 1000$, and $N_{out} = 6$.

The memory size equals one word per input neuron (N_s), three per reservoir neuron (N_{res}) (potential, refractory timer, and xtrace), and two per readout neuron (N_{out}) (activation and bias), plus one per synaptic weight (N_s). In addition, the digit-classifier LSM (but not the CI) requires $N_{ctx} * N_{res}$ words for the context-dependent spiking thresholds, where $N_{ctx} = 6$. For FP-32 representation, this results in a total memory size of **2.3M bytes** and **4.6 Mbytes** without and with the CI, respectively.

The operation count for a forward pass is one multiplication and one addition per synapse, plus one bias addition per readout neuron. This adds up to **1,195,620** operations for the LSM without CI, and **2,383,232** for the LSM with CI.

The number of weights to update per training iteration is $N_{res} * N_{out} + N_{out}$, i.e., **10,010** for the LSM without CI and **16,016** for the LSM with CI.

A.2 LSTM

As an example of an LSTM network with FSDD performance comparable to our LSM network, the Keras-based LSTM implementation in [23] achieves 0.98 accuracy on the task when configured with a hidden state of size 1000 and applied to speech samples that have been preprocessed into 64x64 spectrograms (64 time steps, 64 frequency bands). The network consists of an LSTM layer with input size 64 and hidden/output size 1000, followed by a dense (fully connected) layer with ten linear units, and a softmax layer.

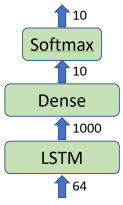

Figure 6: LSTM network. The network consists of an LSTM layer, a dense layer and a softmax function. The numbers indicate vector dimensions.

Figure 7 illustrates the components of the LSTM layer.

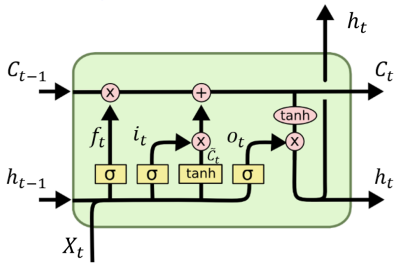

Figure 7: LSTM. The arrows are vectors and the pink circles represent element-wise vector operations, ⊗ for multiplication, ⊕ for addition, and *tanh* for the tanhyp function. The yellow boxes represent multiplication by a (trainable) weight matrix followed by element-wise application of a sigmoid (σ) or tanhyp (*tanh*) function. Merging lines represent vector concatenation and diverging lines represent copying. X_t is the *input* at time t and h_t is the *output*, also known as the *hidden state*, which is recurrently passed on to the next time step, as is C_t, the *cell state*. The three ⊗ components are called the *forget gate*, *input gate* and *output gate*, respectively, hence the names f_t, i_t and o_t for the corresponding gating vectors. Adapted from [20].

The following equations define the processing during a single timestep:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (1)$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (2)$$
$$\tilde{C}_t = tanh(W_c \cdot [h_{t-1}, x_t] + b_C) \quad (3)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (4)$$
$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \quad (5)$$
$$h_t = o_t * tanh(C_t), \quad (6)$$

where $\cdot$ represents matrix/vector multiplication and $*$ is element-wise multiplication. b_f, b_i, b_C and b_O are trainable bias vectors. The dimension of X_t is 64, and that of C_t, h_t, f_t, i_t and o_t is 1000. The four weight matrices have dimension $1000 \times (64+1000) = 1,064,000$.

The numbers of multiplications, additions and squash operations (sigmoid or tanhyp) in a forward time step are therefore:

Equation	Mul	Add	Squash
(1)	1,064,000	1,065,000	1,000
(2)	1,064,000	1,065,000	1,000
(3)	1,064,000	1,065,000	1,000
(4)	2,000	1,000	0
(5)	1,064,000	1,065,000	1,000
(6)	1,000	0	1,000
Dense	10,010	1,000	0
Softmax	10	0	0
Total	4,269,020	4,262,000	5,000

The softmax and squash operations are conservatively counted as equivalent to multiplications, for a total of **8,536,020** add or mul operations.

The LSTM's memory size is calculated as the combined sizes of the four weight matrices plus the dense output layer, each with their bias vectors: $4 * ((N_i + N_h) * N_h + N_h) + N_h * N_o + N_o = 4,270,010$. With FP-32 representation, this amounts to **16.3 Mbytes**.

A.3 Energy estimates

We estimate the energy consumption for a forward pass by multiplying the operation counts by upper-bound estimates of energy per operation for a range of AI/ML processors [26]: 6.3×10^{-11} J for FP-32, 2.1×10^{-11} J for FP-16, and 5.0×10^{-12} J for INT-8.