Darjan Salaj, BSc

Spike-based LSTM-like Modules in Neural Networks

Master's Thesis

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme: Computer Science

submitted to

Graz University of Technology

Supervisor:
Em.Univ.-Prof. Dipl.-Ing. Dr.rer.nat. Wolfgang Maass
Institute of Theoretical Computer Science

Graz, February 2018

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

Date	Signature

Abstract

Working memory is considered an essential component of a system displaying cognitive abilities. Artificial neural networks are successfully endowed with the working memory capabilities by including specific implementations of recurrent networks like LSTM. Underlying processes endowing spiking networks with the same capabilities have remained elusive. In this work we address this issue by developing a novel model that incorporates multiple types of neurons in a structured network. Through experiments we evaluate our model, demonstrating performance comparable to that of artificial networks. Further we analyze the roles of different components within our model and experimentally verify the conclusions.

Zusammenfassung

Das Arbeitsgedächtnis wird als wesentlicher Bestandteil eines Systems angesehen, das kognitive Fähigkeiten aufweist. Künstliche neuronale Netzwerke werden erfolgreich mit Arbeitsspeicherfähigkeiten ausgestattet, indem spezifische Implementierungen von wiederkehrenden Netzwerken wie LSTM eingeschlossen werden. Zugrunde liegende Prozesse, die Spiking-Netzwerke mit den gleichen Fähigkeiten ausstatten, blieben schwer erfassbar. In dieser Arbeit befassen wir uns mit diesem Problem, indem wir ein neuartiges Modell entwickeln, das mehrere Arten von Neuronen in ein strukturiertes Netzwerk integriert. Durch Experimente evaluieren wir unser Modell und demonstrieren eine Performance, die mit der von künstlichen Netzwerken vergleichbar ist. Darüber hinaus analysieren wir die Rollen verschiedener Komponenten des Modells und verifizieren die Schlussfolgerungen experimentell.

Acknowledgment

Special thanks are due to...

Wolfgang Maass, for the guidance and opportunity to work on rewarding cutting edge research topic.

Guillaume Bellec, for the too many hours spent helping, supervising, and advising me throughout my time at IGI, and for his work that mine builds upon.

All the colleagues at IGI, for constantly providing a friendly working environment, social events, and a journal club!

Fabian Tschiatschek, for his invaluable help and discussions throughout my studies.

My wife Anida, for endless support and life outside academia.

My parents, sister and grandmother, for their love and support in everything I do.

Pokola Mrdžić, for unwavering companionship in fostering the occult.

And finally I would like to thank Michael Müller for providing me with this great template. Everything else was easy. As a thank you, I will cite his BSc thesis[1] because no one else will ever do that[2].[3]

"Wubba lubba dub dub!"

Rick Sanchez
Rick and Morty

[1] M. Müller and M. Kampelmühler. Reduced Precision MLPs. Bachelor Thesis 2015

[2] this was a joke

[3] NOT!

Contents

List of Figures

List of Tables

1 Introduction

Humans and animals (to various extents) possess abilities to acquire knowledge through sensory inputs and use it to solve the challenges they face in life. These abilities are embodied in form of observation, comprehension, evaluation, reasoning, memory, planning, and are commonly labeled cognitive abilities. Historically, the cognitive abilities have been taken for granted, and investigated mostly in fields of philosophy and psychology without much formalism. In past two centuries this has changed by the rise of more formal and experimental research in fields of biology and neuroscience. More importantly, past few decades have seen the rise in machine learning which has sparked the interest of understanding the cognitive abilities in a formal manner and to such a degree where they could be reproduced in artificially constructed models. This interest resulted in emergence of field of computational neuroscience. Eric L. Schwartz introduced the term "computational neuroscience" in 1985 for the field which was, before that point, labeled by different names such as neural modeling and brain theory.

Computational neuroscience is a field dedicated to producing theories of brain function by means of information processing view of nervous system [2]. It is a cooperative field between neurobiologists and computer scientists. This cooperation is necessary due to the fact that applying only bottom-up or top-down approach will likely fail in explaining the underlying functional systems of neural networks [2].

One of the key cognitive systems contributing to the brains ability to solve complex tasks is working memory. Human brain requires working memory to solve even the most common daily tasks such as reading and comprehending, communication through speech, and planning of motion. More concretely this can be described as ability to hold, process, and use information for some period of time during problem solving. Artificial neural networks are endowed with this ability by including layers of recurrent memory units (usually of LSTM [3] or GRU [4] type).

1.1 Motivation

In presence of artificial neural networks as successful solutions to the working memory related problems, one might ask why is the development of more constrained, biologically plausible models attractive. The motivation behind this effort is manifold:

- Spiking models implemented on neuromorphic hardware is significantly more energy efficient then the artificial counterparts. If the spiking models can only approach the performance levels of artificial networks withing application tolerance range, the benefits are still very high due to energy efficiency of application.
- Successful spiking models offer new insights into the inner working of the brain and give basis to further brain research which has impact on variety of fields (e.g. medicine).
- Original artificial neural network models were developed through inspiration from biological neural networks. Investigating biological neural networks offers opportunity to learn about mechanism that could be further adapted and used in artificial implementation to improve the performance. One prominent example is the learning in the brain on many different timescales, which has been popularized in the machine learning community under "learning to learn" term.

1.2 Contributions

In this work we develop simple model of spiking neurons with adaptive threshold and combine them with other neurons in network with imposed structure. Our aim is to endow spiking neural networks with many of the properties of artificial networks with memory units. We demonstrate the emerging features of such networks through tasks that require memory. We compare the performance of our model against the spiking networks without the adaptive mechanism and against the artificial neural networks. Through this we demonstrate the competitive performance of our novel model against the artificial counterpart. Finally we analyze the roles of different components within our model and experimentally verify the conclusions.

2 Background

Humans and animals exhibit complex behaviors which include capacity to learn, ability to solve complex tasks, planning, and creativity. Such abilities emerge from the signal processing and neural computation that is happening in the nervous system and the brain, commonly called neural networks.

Basic building block of biological neural networks is the spiking neuron (Figure 1). It produces discrete output by integrating the information it receives from the neighboring neurons. Most of the neurons in the cortex are of spiking type, however there exist specialized neurons such as analog neurons in the retina. In this work we will not focus on non-spiking types of cells. We describe the inner mechanics of biological spiking neurons in Section 2.1. In addition to neurons, biological neural networks consist of number of glia cells that support the functioning of the network but are not directly involved in information processing [5].

In the Section 2.2.1 we discuss the first formal neuron model and its extensions to the artificial neural networks. In the Section 2.3 we discuss the leaky integrate and fire spiking neuron model which we will use throughout the experiments in this work.

2.1 Biological Neuron

Neurons are cells that build nervous system and enable neural computation through their electrical properties. They control the electric potential between their cellular walls which is termed membrane potential. This dynamic involves transport of different types of ions which are gated through their corresponding ion channels. There exist many types of neurons, however their fundamental architecture and electrical properties are common among all of them. Three functionally distinct parts that compose a neuron are: dendrites, soma, and axon.

Dendrites are fiber which are attached to neighboring neurons through synapses. Their role is to collect the input signals and relay them to the soma. *Soma* is the cell body whose role is to integrate all incoming signals and generate the output in form of action potentials i.e. spikes. *Axon* propagates these spikes to the connections with other neurons.

Connections between neurons are called a synapses. More specifically this is the connection between an axon fiber of pre-synaptic neuron and the dendrite of post-synaptic neuron. These synapses are chemical in nature and do not propagate the spikes of pre-synaptic neuron to the post-synaptic neuron directly. Instead the signal is processed and transmitted through chemical processes in the synapse: Arriving spike at the axon terminal causes opening of the calcium ion channels in the axon which allows the flow of calcium ions through the membrane. Higher calcium concentration

triggers the vesicles to merge with axon membrane, releasing the containing neurotransmitters to the extracellular fluid in the synaptic cleft. These neurotransmitters bind to the receptors on the post-synaptic dendrite membrane which causes opening of the ion channels through which cations and anions flow. This causes the post-synaptic potential in the dendrites of the receiving neuron [5].

Usually the dendrites of a neuron are connected to many axons of other neurons receiving spike trains which produce many post-synaptic potentials. Such post-synaptic potentials have high variance in shape and amplitude which results from the complex ion channel mechanisms and other bio-chemical processes in the synaptic efficacy. The cell body integrates these post-synaptic potentials into a single membrane potential.

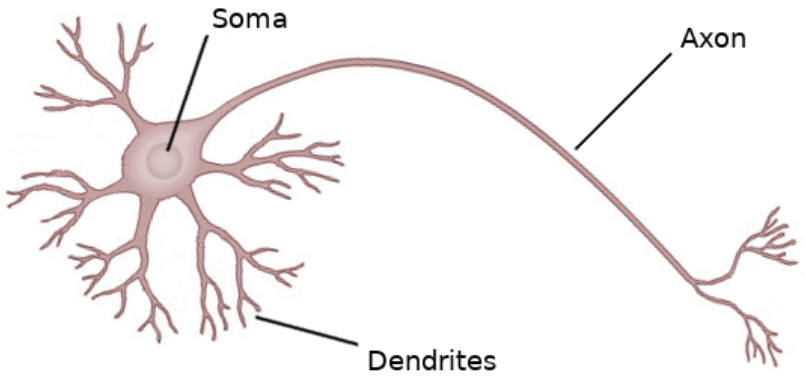

Figure 1: **Structure of a typical biological neuron**: Dendrites make synaptic connections with neurons from which they receive the input. Soma integrates the inputs from dendrites and generate spikes which are propagated by axon to the synaptic connections of the downstream neurons.

Functionally, synapses are categorized as excitatory or inhibitory. Excitatory synapses depolarize (increase) the membrane potential of the post-synaptic neuron, where as the inhibitory synapses decrease the membrane potential pushing it in the direction of cells resting potential, thus reducing its excitability. All of the post-synaptic currents at dendrites are propagated to the soma. If the membrane potential at the soma crosses the cell-specific threshold, a spike fires (is generated) and is propagated over the axon to synaptic connections with other neurons. After the spike fires, the fast repolarization of the membrane potential occurs, lowering the membrane potential to a value bellow resting potential. At this point the membrane potential does not change for a brief period of time (termed refractory period). This phenomenon is called hyperpolarization and it effectively inhibits spike

generation by increasing the input stimulus required to move the membrane potential over the threshold.

The spikes generated in soma are discrete in nature, having similar shape and duration across neurons. Information transmitted between neurons is not solely contained in the spike train of the pre-synaptic neuron, but is augmented by the complex dynamics of the synapse itself which is conditioned by previous activity and other factors. This has strong effect on the resulting post-synaptic potential.

One of the main learning mechanism in the brain is the adaptation of the strengths of the synaptic connections. In addition, the biological neural network not only adapt the strength of, but also create and remove synaptic connections (effectively changing the architecture of the network).

Different neuron models with various levels of detail have been developed by different research communities. In the machine learning domain, the artificial neural networks have been largely developed to efficiently solve complex tasks on contemporary hardware. These belong to the first and second generation of neural networks [6] and are discussed in the following section. On the other hand, computational neuroscience and related fields developed more biologically realistic spiking neuron models which are considered the third generation [6]. Inspired by real neurons, these models implement spike based communication and in turn incorporate spatio-temporal information in the computation. In Section 2.3 we will discuss a specific spiking neuron model which we will use extensively for the experiments in Section 6.

2.2 Artificial models

2.2.1 McCulloch and Pitts neuron

First attempt to describe fundamental behavior of neurons has been made in 1943. by McCulloch and Pitts [7]. This initial model had binary activation, excitatory and inhibitory inputs, unit weights and fixed threshold. This meant that the weights and thresholds of the network had to be analytically determined and could not be learned. Despite these limitations, network of McCulloch-Pitts neurons is able to model any logic function.

As an extension to the McCulloch-Pitts model, in 1958. the perceptron model has been developed [8]. This model allowed for parameter learning and application of efficient minimization heuristics to solve linear separability problems. In the perceptron the complex dynamics of the synapses are simplified to a weight factor. The membrane potential of the neuron is defined as a weighted sum of the activation of input neurons:

$$u = \sum_{i=0}^{N} w_i x_i + w_0 \tag{1}$$

where w_i is the synaptic weight to the input neuron i, w_0 the weight bias determining the resting potential, and N the number of input neurons. The output is defined as a threshold function of the membrane potential:

$$z = \begin{cases} 1 \text{ if } u \geq \theta \\ 0 \text{ if } u \leq \theta \end{cases} \tag{2}$$

where θ is the threshold parameter.

Next generation of artificial neuron models replaced the threshold function at the output with a non-linear activation functions, usually sigmoid or tangens hyperbolicus. As opposed to the perceptron, networks of neurons with non-linear activations are able to distinguish data that is not linearly separable. First networks with these properties were multilayer perceptron and madaline networks, but are today colloquially referred to as "vanilla" neural networks.

These networks, despite their simplicity, have proven to be powerful models serving as universal function approximators. This along with the existence of efficient algorithms for training such networks made neural networks a popular candidate for applications from different fields such as image recognition [9].

2.2.2 Recurrent networks

A major shortcoming of the traditional neural networks is their inability to incorporate information of the previously seen inputs in the computation of the current state. This renders them unusable for the problems which are sequential in nature and require the memory capabilities in order to be solved efficiently. To overcome this the recurrent neural networks have been developed. These networks contain loops withing layers allowing the information to persist over many computation steps. Developments in the recurrent networks allowed for the efficient application of the network models to problems of machine translation, language modeling, speech recognition, and more [9].

2.3 Leaky integrate and fire neuron

In this section we discuss a formal spiking neuron model called the leaky integrate and fire (LIF) model. LIF neuron belongs to the third generation of neuron modes due to its biologically realistic communication and is one of the most popular formal spiking neuron models used in research. Information transmitted in the networks of LIF neurons are encoded in the spike trains the neurons produce.

The cell membrane is modeled as electrical circuit of parallel resistor R and capacitor C elements which are driven by current I. The current I is

interpreted as incoming current from the dendrites. The current component discharging over the resistor R is interpreted as leaking of ions through cells open channels, where as the second current component charges the capacitor C. The capacitor voltage is interpreted as membrane potential u and is defined by

$$\tau_m \frac{\delta u}{\delta t} = -u(t) + RI(t) \tag{3}$$

where $\tau_m = RC$ is the membrane time constant of the neuron [5].

When the membrane potential rises above the threshold parameter θ, an action potential (spike) is generated. In this model the spikes are formal events defined by their firing time $t^{(f)}$ which is determined by the threshold criterion

$$t^{(f)} : u(t^{(f)}) = \theta \tag{4}$$

Immediately after the spike fires, the membrane potential is set to the resting potential $u_r < \theta$ [5].

In general, the LIF neuron can implement an absolute refractory period, which we do in our simulations. In this case when the membrane potential u reaches the threshold θ at time $t = t^{(f)}$, we interrupt the dynamics (3) for the period of absolute refractory time Δ^{abs} and continue the dynamics at $t^{(f)} + \Delta^{\text{abs}}$ with the membrane potential $u = u_r$ [5].

LIF model does not define the form of a spike explicitly, thus the driving current I of interconnected LIF neurons is undefined. In the biological neural network the activity of pre-synaptic neurons determine the driving current. Spikes produce current pulses at the synapse which are proportional to the excitatory post synaptic potential in the receiving dendrite. In this setting the input current is the weighted sum over received current pulses, where weights represent the synaptic efficacies:

$$I^{(t)} = \sum_i w_i \sum_f \alpha_i(t - t_{i,f}) \tag{5}$$

where kernel $\alpha_i(t)$ is the current pulse shape produced at synapse i, $t_{i,f}$ is a list of input spike times arriving at synapse i, and w_i the weight or efficacy of the synapse i. The kernel α is usually a Dirac δ-pule, but can also be a double exponential kernel parametrized with rise time τ_r and fall time τ_s:

$$\alpha_i(t) = \frac{q}{\tau_s - \tau_r}\left(exp\left(-\frac{t - \Delta_i}{\tau_s}\right) - exp\left(-\frac{t - \Delta_i}{\tau_r}\right)\right)\Theta(t - \Delta_i) \tag{6}$$

where q is the total charge of a synapse, Δ_i the synapse delay, and $\Theta(t)$ the Heaviside step function [5].

2.4 Working Memory

Working memory is an essential part of a system that exhibits complex cognitive abilities. Humans solve tasks that make heavy use of working memory on daily basis. These include reading articles, cooking, comparing prices, adapting and propagating information through speech. For successful completion of such tasks, the human is required to do multiple steps while temporarily keeping the intermediate results in mind [10].

Term "Working memory" is the theoretical construct used in cognitive psychology that refers to the mechanism underlying the maintenance of task-relevant information during the performance of a cognitive task. The term originated in the 1960. work of Miller et al. [11]. It has been stated that the working memory is "perhaps the most significant achievement of the human mental evolution" [12]. Due to its significance, it has also become a central construct studied in cognitive neuroscience. It is important to note, however, that despite the prominence of the term, there is a lot of confusion and contradiction in the definition and meaning of the term used throughout different research communities.

In this work we use the term working memory as described in [13, 14]: Working memory is the ability to hold information and manipulate it as opposed to short-term memory which refers to just holding the memory. These two concepts are linked to different neural subsystems. The distinction is shown by separate factor clustering in factor analyses, different reliance on dorsolateral prefrontal cortex, and different developmental progression between the two features.

2.5 Related Work

In this work we hypothesize about potential implementation of working memory mechanism in the spiking networks. We propose a form of neural network with multiple types of neurons and imposed architecture. Previous works approached the same problem in different ways. Here we briefly discuss these related works and their contributions.

Cortical microcircuits as gated-recurrent neural networks [15]
Authors of this work observe the stereotypical structure in cortical circuits, interpreting them as the common computational units. They develop a modified LSTM architecture, termed subLSTM, which gates the information in subtractive manner. This unit is still able to approach the LSTM performances on the sequential MNIST and language modeling tasks used for evaluation. The key point of this work is that the architecture of LSTM unit lends itself to a natural mapping onto known cannonical cortical circuits with few differences which have been bridged with the development of subLSTM.

Memory cell of an LSTM unit is controlled by gates through a multiplicative factor. This operation is however not biologically plausible, thus the subLSTM model has been developed to address this issue. In subLSTM the memory cell is gated in a subtractive manner, which is interpreted as inhibition through inhibitory neurons. This however poses a constraint to the model since this subtractive inhibitory mechanism needs to match the excitatory input well in order to act as a gate effectively. This implies that in order to work the network needs to be well balanced, which would be a challenge to implement in spiking networks which are the end target for this model.

Another critique of this work is that through the proposed mapping, only a single LSTM-like unit is implemented in a cortical circuit containing many neurons, which renders this mapping very inefficient. In our work we present an efficient implementation of LSTM-like spiking network approaching the performance of artificial LSTM network with same number of parameters and training iterations. Finally the use of LSTM architecture as target when developing a spiking models of memory is not a good approach since the architecture of LSTM units have been primarily developed to mitigate the issue of learning in automated gradient frameworks, namely exploding and vanishing gradients, which do not exist in biological neural networks.

Gating out sensory noise in a spike-based Long Short-Term Memory network [16]
In this work a simplified LSTM cell is targeted. Authors develop an Adaptive Analog Neuron (AAN) which is used to build a simple LSTM cell version containing only an input gate and a memory cell. They also develop an Adaptive Spiking Neuron (a variant of LIF neuron with fast adaptation) which can be used as drop-in replacement for the AAN. Main idea of their approach is to use a network of ANN units to train the network on specific task, and finally replace the ANN units of the trained network with the Adaptive Spiking Neurons. With this approach the authors were able to solve Sequence Prediction task (Hochreiter & Schmidhuber 1997) and T-Maze task (Bakker 2002).

Prefrontal cortex basal ganglia working memory (PBWM) [17, 18, 19, 20, 21, 22]
PBWM is an algorithm that has been put forward in many works as a hypothesis for working memory model in the basal ganglia and the prefrontal cortex. Functionally it also is comparable to the LSTM model. In the [17], the authors try to shed light on the underlying mechanism of executive function of cortical homunculus. They approach the problem by constructing a learning mechanism that endow perfrontal cortex model with ability to control different brain areas in strategic, task-oriented manner. The proposed

learning mechanism is an actor-critic architecture formed from subcortical structures (found in midbrain, basal ganglia, amygdala) and is designed to solve the structural and temporal credit assignment problem.

3 Neuron Model

In this section we describe the neuron models used throughout the simulations in Section 6. To take advantage of modern frameworks for training networks with automatic differentiation and backward propagation of errors (backpropagation), we simulate the spiking neural networks in discrete time. Details on how the backpropagation has been applied to recurrent networks of spiking neurons are discussed in Section 5.

3.1 LIF implementation

We perform the neuron simulations in discrete time with time step of $\delta t = 1$ ms. Each neuron j is given a membrane time constant $\tau_{m,i}$ chosen from uniform distribution in [15, 30] ms. The membrane potential of neuron i is given by:

$$u_i(t + \delta t) = \alpha_i u_i(t) + (1 - \alpha_i) I_i(t) - I_i^{ref}(t) \tag{7}$$

$$\alpha_i = exp\left(\frac{-\delta t}{\tau_{m,i}}\right) \tag{8}$$

Each synapse is associated with a unique delay d in [1, 10]ms with uniform probability so that: $W_{i,j}^d$ is the synaptic weight and $W_{i,j}^{d'}$ with $d' \neq d$ is zero. The input current $I_i(t)$ of the neuron i at time t is defined as:

$$I_i(t) = \sum_{d,j} W_{j,i}^d x_j(t) \tag{9}$$

where x represents the spike trains of neurons which axons are connected to the dendrites of neuron i.

The refractory current $I_i^{ref}(t) = z_i(t)(\theta - V^{\text{reset}})$ resets the membrane potential after an action potential to $V^{\text{reset}} = 0$. A spike is initiated when the neurons membrane potential rises above the threshold:

$$z_i(t) = \text{HSS}(u_i(t) - \theta) \tag{10}$$

where HSS denotes the Heaviside step function, and threshold $\theta = 0.01$.

3.2 LIF with adaptive bias

In addition to the vanilla LIF neuron model, we implement LIF neuron model with adaptive bias i.e. threshold, which we term ALIF in the following. As opposed to LIF model which has the fixed threshold potential $\theta = 0.01$,

the ALIF model has adaptive threshold. For ALIF neuron k the adaptive threshold is defined as:

$$\theta_k = \theta_0 + b_k(t)\beta \tag{11}$$
$$b_k(t) = (1 - \alpha^{\text{thr}})z_k(t) + \alpha^{\text{thr}} b_k(t-1) \tag{12}$$

where $z_k(t)$ is the spike train of ALIF neuron k at time t, α^{thr} a parameter in (0, 1), β the update scaling factor, and $\theta_0 = 0.01$ the baseline threshold. We set $\alpha^{\text{thr}} = exp(\frac{-\delta t}{\tau_a})$ with τ_a set to number of milliseconds matching the expected memory length requirement of a given task. This however applies only to artificial tasks where the memory length requirement is predictable. For more complex tasks like speech recognition, and more biological realism, a wide spread of τ_a values over neuron population is more suitable.

4 Network Model

There exist many models of artificial recurrent neural networks that approach state of the art performance on memory related tasks, most notable being the LSTM[3] and GRU[4]. In the following Section 4.1 we describe the artificial recurrent neural network that served as inspiration for development of our spiking network model discussed in Section 4.2.

4.1 Structurally Constrained Recurrent Network

Structurally Constrained Recurrent Network (SCRN) has been developed by Mikolov et al. [23] and gives as an insight into the effects of imposing a structural constraint on a simple artificial recurrent model which by itself is considered inadequate for solving memory related tasks.

Original Elman neural network [24] as shown in Figure 2(a) consists of an input layer, a recurrent hidden layer, and an output layer. The recurrent hidden layer allows the information about previous inputs to be propagated through time and influence the future outputs. Formally this model is defined by following equations:

$$h_t = \sigma(Ax_t + Rh_{t-q})$$
$$y_t = f(Uh_t)$$

where $\sigma(x) = \frac{1}{1+\exp(x)}$ is the sigmoid function, and f is the soft-max function.

Such simple models of recurrent neural networks, although elegant, suffered from the exploding and vanishing gradient problem. During training

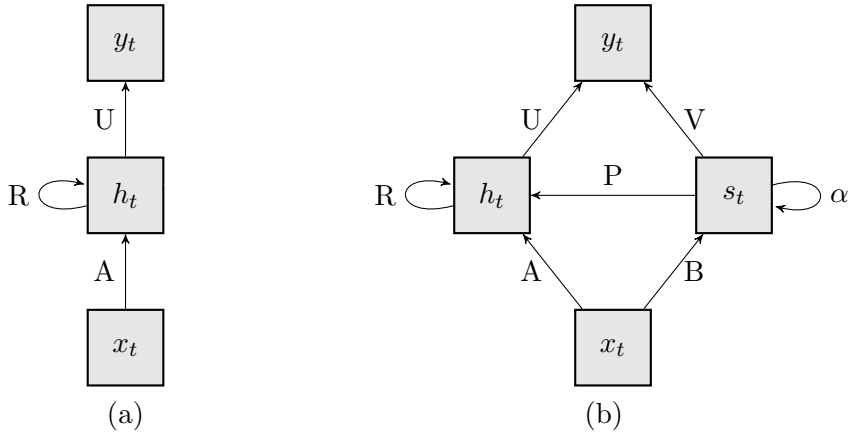

Figure 2: **Architecture comparison of SRN and SCRN**: Architecture of (a) simple recurrent network and (b) structurally constrained recurrent network.

of a network, if the accumulation of error gradients during an update results in large values, this leads to large weight updates and unstable training. In extreme case these large values can cause an overflow in the training framework. The growth of the gradients is a result of repeated multiplication through the network layers, which is especially the case with recurrent neural networks which need to be unrolled in time. This is called the exploding gradient problem, and can be worked around by means of gradient clipping.

The problem of vanishing gradient results from propagating the gradients through layers which have small weights, which in turn shrink the value of the propagated gradients quickly to values close to zero. This results in no updates being made to recurrent layers with dependencies reaching further in the past, which makes learning longer term patterns difficult. Vanishing gradient problem can not be mitigated easily as exploding gradient problem. This motivated the development of much more complex and biologically unfeasible recurrent units such as LSTM [3] and GRU [4]. SCRN however offers network architecture that overcomes the problems of exploding and vanishing gradients and is also able to retain information about longer patterns while still retaining the architectural simplicity. Formally the SCRN is defined:

$$
\begin{aligned}
s_t &= (1-\alpha)Bx_t + \alpha s_{t-1} \\
h_t &= \sigma(Ps_t + Ax_t + Rh_{t-1}) \\
y_t &= f(Uh_t + Vs_t)
\end{aligned}
$$

where α is a parameter in (0,1). Architecture of SCRN is shown in Figure 2(b). Notice however that the self pointing edge on s_t with label α

represents the dependence of s_t on its state from previous time step, and not the self recurrent connection as is the case with node h_t.

4.2 Spike based Memory Unit

Inspired by SCRN [23], we propose a spiking model which we term Spike based Memory Unit (SMU). Architecture of proposed SMU is shown in Figure 3.

In Figure 3, orange pool represents a recurrent network of LIF neurons which we label O, while blue pool represents a non-recurrent layer of ALIF neurons we label M. Both O and M receive input from a pool of input neurons X. Pool O additionally receives input from M.

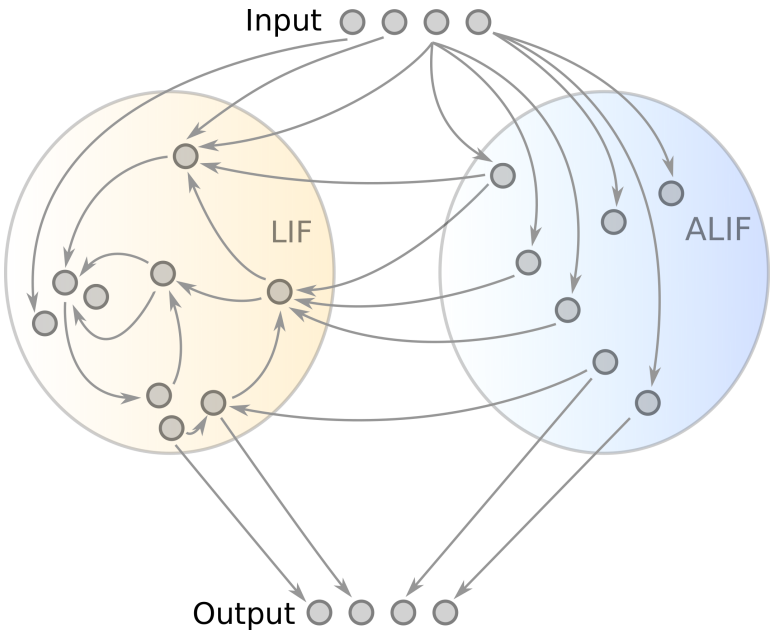

Figure 3: **Architecture of SMU network**: Gray circles represent spiking neurons, and gray arrows the synaptic connection. Recurrent neuron pool marked with large orange circle, contains only LIF neurons, while non-recurrent pool marked with blue circle contains only the ALIF type neurons.

Inner neuron dynamics of LIF and ALIF neurons are described in Section 3. The input current is defined by equations 13 and 14 for neurons in O and M respectively.

$$I_j(t) = \sum_{d,i} W_{i,j}^{\mathtt{in},d} x_i(t) + \sum_{d,k} W_{k,j}^{\mathtt{mem},d} m_k(t) + \sum_{d,l} W_{l,j}^{\mathtt{rec},d} o_l(t) \tag{13}$$

$$I_k(t) = \sum_{d,i} W_{i,k}^{\mathtt{in},d} x_i(t) \tag{14}$$

where x, m, and o are the spike trains of input neurons X, ALIF neurons M, and LIF neurons O respectively.

The output Y receives the inputs from both O and M and is defined as softmax over weighted sum of post synaptic potentials:

$$\boldsymbol{y}(t) = softmax(U\widehat{\boldsymbol{o}}(t) + V\widehat{\boldsymbol{m}}(t))$$

where $\widehat{\boldsymbol{o}}(t)$ and $\widehat{\boldsymbol{m}}(t)$ are vectors of exponentially convolved spike trains of $\boldsymbol{o}(t)$ and $\boldsymbol{m}(t)$ respectively:

$$\widehat{\boldsymbol{o}}(t) = \sum_{t'=0}^{t} \boldsymbol{o}(t') exp(\frac{-t-t'}{\tau_{out}}) \tag{15}$$

$$\widehat{\boldsymbol{m}}(t) = \sum_{t'=0}^{t} \boldsymbol{m}(t') exp(\frac{-t-t'}{\tau_{out}}) \tag{16}$$

with $\tau_{out} = 22$ms, and $\boldsymbol{o}(t)$ and $\boldsymbol{m}(t)$ being the vectors of neuron spiking outputs at time t.

The output dimensionality is chosen to match the requirements of the task at hand. For the supervised learning classification problems it is number of classes, and for the reinforcement learning tasks it is the number of possible actions. We apply softmax function over this output and define loss as cross-entropy between the target values and the softmax output. Role of softmax on the PSPs of the output pool Y can be interpreted as winner-take-all circuits in context of classification tasks.

5 Training Method

Artificial neural networks have been successfully trained by gradient-based methods, most notably backpropagation (BP). For the recurrent neural networks the extension of this method is used which is called backpropagation through time (BPTT).

However, the BP and BPTT methods can not be directly applied to the spiking neural networks due to the non-differentiable nature of the spiking neuron output. To overcome this problem Bellec et al. developed a novel variant of BPTT which is applicable to the spiking neural networks simulated in discrete time. Section 5.1 describes the detail of this novel BPTT and is based on [25].

5.1 SSD-BPTT

A spiking neuron i at time t produces a non-differentiable output $z_i(t) \in \{0, 1\}$. This output is produced by the spike generation dynamics of the neuron which is defined as thresholding of the membrane potential:

$$z_i = \mathrm{HSS}(u_i(t) - \theta) \tag{17}$$

where HSS is the Heaviside step function, $u_i(t)$ the membrane potential of the neuron i at time t, and θ the threshold voltage.

Solution to the non-differentiability of the Heaviside step function is to interpret its theoretical derivative as a Dirac distribution. Following [26, 27] the derivative of HSS is approximated as smoothed Dirac:

$$\frac{\delta z_i(t)}{\delta v_i(t)} \approx \mathrm{SD}(v_i(t)) := \max\{0, 1 - |v_i(t)|\} \tag{18}$$

where SD is the smoothed Dirac function, $v_i(t)$ the normalized membrane potential $v_i(t) = \frac{u_i(t)-\theta}{\theta}$. This approximation allowed for deep neural-networks of deterministic binary neurons to be trained by means of back-propagation, but has shown to be unstable for training of recurrent networks of spiking neurons where the gradients are propagated through the spiking mechanism of a neuron many thousands of time steps [25].

To overcome this, further adaptations have been made to (18). The gradients that are propagated through neurons are sparsified by a dampening factor γ. This method has been termed sparsified smoothed Dirac (SSD):

$$\frac{\delta z_i(t)}{\delta v_i(t)} \approx \mathrm{SSD}(v_i, t) := \gamma\ \mathrm{SD}(v_i(t)) \tag{19}$$

5.2 Evolutionary Strategy

Possible alternative training method which solves biological realism problem of back propagation methods is the evolutionary strategy [28]. In this work we opted for not using this approach firstly due to the inherent computational cost that comes with it, and secondly due to performance inferiority in comparison with gradient based approaches in the context of supervised learning tasks on which we focus in this work.

6 Experiments

To evaluate if our proposed SMU model endows spiking networks with many of the features of LSTM artificial networks, we consider a set of different memory related tasks. We first solve a distribution of artificial toy problems labeled store-recall task in Section 6.1. Our aim with solving these tasks is to demonstrate the ability of our spiking models to replicate the basic behaviors of LSTM networks like storing and recalling information on demand, and processing this information to produce the required output. Next we consider a more demanding sequential MNIST task which we discuss in Section 6.2. Lastly we apply our models to TIMIT task of speech recognition, in Section 6.3.

6.1 Store-Recall task

The store-recall task is simple working memory task testing if the model can memorize and recall input values on demand. The task is defined in discrete steps, each lasting for τ_c milliseconds. Since these steps dictate the changes in the network input values (characters), we label them 'character steps'. The input is defined through a set of input channels. Two channels are reserved for 'store' and 'recall' signals to the network which instruct the network to either store the currently shown input signal on value channels, or recall the last stored signal respectively. All other input channels are labeled value channels and are used to encode the input values (which are stored and recalled during the task).

In our setup we define two value channels, termed '0' and '1' representing the bit values (characters) that should be stored and recalled. These two channels are activated exclusively (either one or the other) and constantly feed random binary stream to the network. When the store signal is received, the network should remember the currently present binary value from the the value encoding channels. When the recall signal is present, the network has to output the previously stored value. Dimensionality of the network output is equal to the number of value channels, which in this case is two. Input values, shown on value channels, are picked with equal probability and the store-recall channels are activated in alternating fashion with probability of $p_{SR} = 0.2$. Another implementation detail is that no value is shown at value encoding channels during the recall signal.

Input channels ('store', 'recall', '0', '1') were encoded in a population of 100 spiking neurons. Each of the input channels is encoded in a sub population of 25 neurons that fired with 30Hz firing rate during its activation.

We first trained the networks on sequences of length 12 (character steps). With the probability of alternating store-recall signal being $p_{SR} = 0.2$, we calculate the expected number of character steps between consecutive store-recall signals $E[k] = \frac{1-p_{SR}}{p_{SR}} = 4$. This converts to expected delay of $\tau_c *$

$E[k]$ms in biological time. In first setup we extend this delay by increasing the character step length τ_c.

The networks compared in this and following sections are: LIF network with 80 neurons, and SMU network with 60 neurons in LIF pool and 20 neurons in ALIF pool. Using this setup both networks have the same number of spiking neurons, note however that the SMU network has less synapses (parameters) due to the constrained architecture.

Results: Networks of LIF neurons are only able to reliably solve the store-recall task up to 200 ms expected delay, see Table 1, where as the SMU with same number of neurons is able to solve the task with arbitrarily long delays. Figure 4 (containing the spike raster plots of input, LIF and ALIF neurons, traces of ALIF neuron biases, and network outputs) shows performance of successfully trained SMU network on instance of store-recall task with $\tau_c = 250$ms containing a delay of 1250ms between store and recall signals. In the raster plot of ALIF neuron pool we can observe that the stored information is maintained in the activity silent manner, through the internal bias state, showcasing the functional difference of the ALIF neurons to the LIF neurons in this setting. Improvement to the performance by adding ALIF neurons to the model also introduces the additional hyper-parameters: speed of bias decay τ_α and the scaling factor of bias change β, see (11). In general we can set τ_α to the expected delay (in milliseconds) of the task instance if we want to ensure that the ALIF neurons can easily memorize values for those delays. For the scaling factor β we do a task specific parameter search in the following section.

We also investigate a more natural approach to choosing the τ_α parameter. Instead of using the insight into the task specifics to choose the bias decay τ_α, we set it to wide spread of values which is something we can observe happening in biological neural networks for a semantically similar parameter [29]. Allen Institute for Brain Science provides us with database of single cell feature measurements of both human and mouse brains. Here we observe that brains of both species contain a spread of adaptation index, with human brains having a wider spread. The adaptation index in the data is defined as: "The rate at which firing speeds up or slows down during a stimulus" [29], which is semantically equivalent to the adaptiveness of the threshold in our implementation of ALIF neuron.

6.1.1 Optimizing scaling factor β

For the β we perform a simple parameter search to optimize for the learning speed of the SMU network. More specifically we measure the number of training iterations required to reach error bellow 10%. Plots in Figure 5 show the learning speed of the network as a function of different β values in two experiment setups. First setup is performed with $\tau_c = 100$ms resulting in 400ms expected delay time, and $\tau_\alpha = 200$ms. In second setup we set

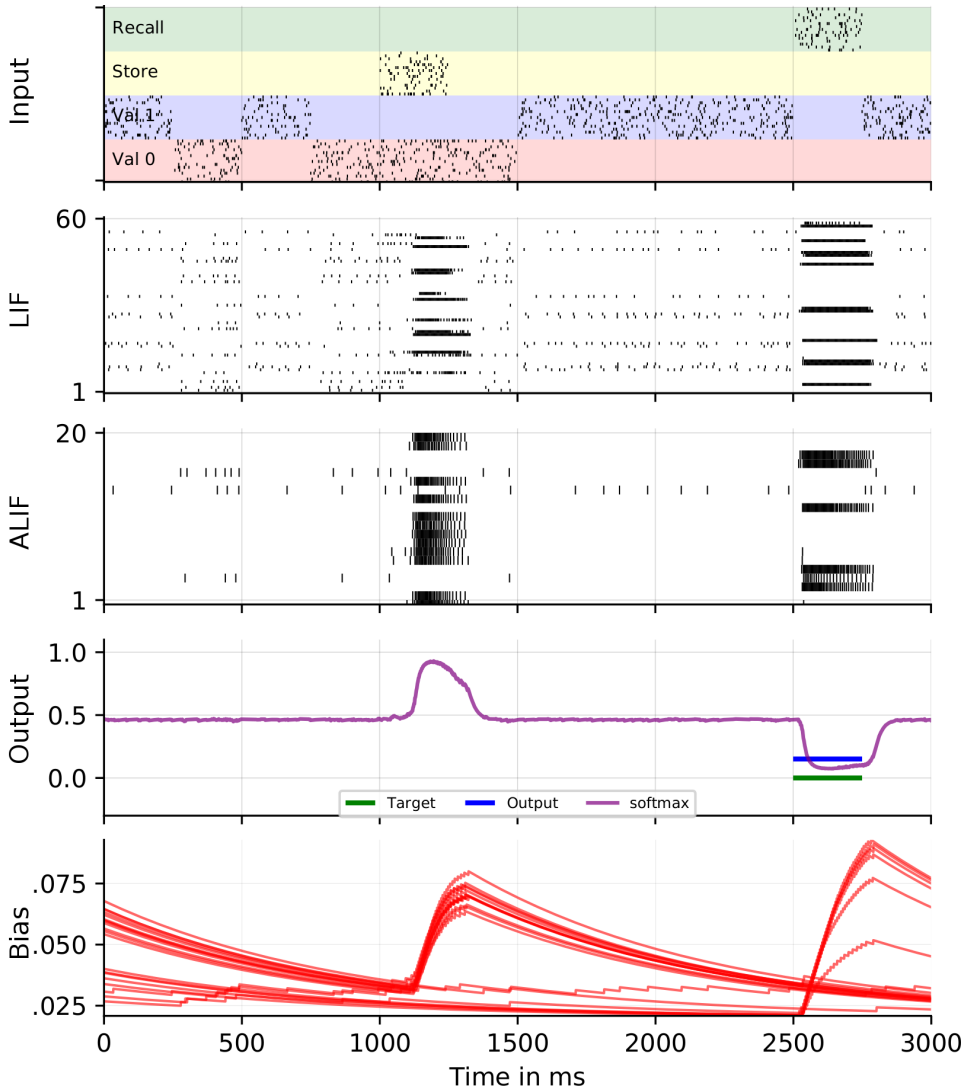

Figure 4: **Trained SMU network solving store-recall task**: Character step is set to $\tau_c = 250$ms. From top to bottom: Spike raster of input neurons, spike raster of LIF neurons, spike raster of ALIF neurons, softmax output (with target and average of softmax for character step). and bias plot of ALIF neurons.

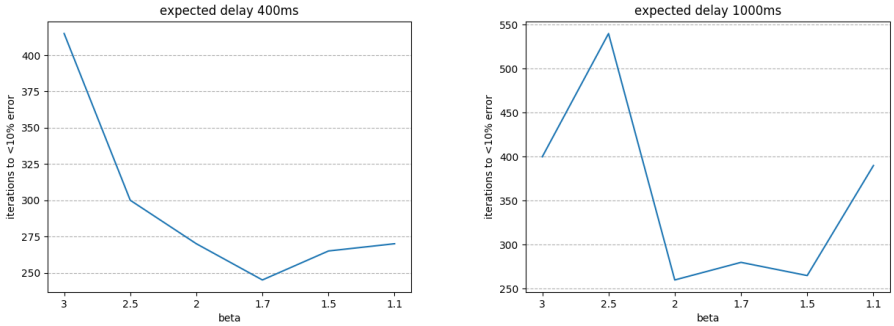

Figure 5: **Impact of bias update scaling factor β on network learning speed**: We perform a limited grid search over the bias update scaling parameter β to optimize the learning speed of SMU network. Plots show the number of training iterations needed to reach error bellow 10% (y-axis) for simulations of store-recall task with expected delay of 400ms for left plot and 1000ms for the right.

$\tau_c = 250$ms (1000ms expected delay) and $\tau_\alpha = 400$ms. For every β value we performed two simulations with random initializations and plot the mean number of iterations. We set the τ_α to values smaller then expected delay on purpose to make it harder for the network to store the values over longer periods. This should result in more spiking in the ALIF pool being required to successfully store a value. This firing highly depends on parameter β which dictates the strength of bias update on every spike.

Based on Figure 5, we can observe that the optimal value for β is found in the range (1.5, 2), so we fix the parameter to the value $\beta = 1.7$ for further store-recall experiments.

6.1.2 Evaluation on large delays

In this section we optimize the network to solve the task for longer delays between the store and recall signals. We extend the delay by increasing the τ_c parameter. Table 1 contains the results of different training runs for SMU and vanilla LIF networks.

From the results we can observe the importance of ALIF neurons for the input memorization over longer periods. On the other hand the network with more LIF neurons is able to memorize the inputs more efficiently for shorter time periods by encoding the data in network activity.

Extending the τ_c parameter not only extends the expected delay, but also extends the character step duration. This implies that the network has longer period for storing the value, which makes the task easier. To verify the robustness of SMU to smaller τ_c with longer expected delays, we test the network on longer input sequence lengths, fixed $\tau_c = 250$ms, and smaller store-recall signal probability to $p_{SR} = 0.1$. In this setup the SMU was still able to reach errors lower then 5% withing same training period

Task τ_c (ms)	50	125	250	500	1000
expected delay (ms)	200	500	1000	2000	4000
SMU with τ_α = expected delay	0.0414	**0.0150**	0.0078	0.0073	**0.0197**
SMU with spread τ_α in [200, 4000]ms	0.0186	0.0285	**0.0013**	**0.0064**	0.0373
LIF	**0.0039**	0.3070	0.3183	0.3503	0.3748

Table 1: **Accuracy on store-recall task**: Smallest mean test errors reached by different networks over 500 training iterations. Mean test error is calculated over any ten consecutive iterations. Final numbers represent the average values over multiple random initializations. Number in boldface are the best results reached for particular delay.

of 500 iterations. Appendix A contains Figures 13 and 14 showcasing the SMU solving the task in this setup with delay between store-recall signals of 4250ms and 8750ms respectively.

6.1.3 Architecture evaluation

To evaluate the influence of specific architecture proposed by Mikolov et al. [23], we implement three additional architectures based on the original SCRN by adding the possible additional connections to the model until the fully connected pool of mixed LIF and ALIF neurons is reached. New architectures emerging from this process are shown in Figure 6 with labels A, B, C, and D. Architecture A is the original SCRN architecture, while D is the fully connected model.

The changes to the architecture are also reflected in the input current dynamic (14) of the M neuron pool in the following way:

$$\text{arch. B}: I_k(t) = \sum_{d,i} W_{i,k}^{\mathtt{in},d} x_i(t) + \sum_{d,l} W_{l,k}^{\mathtt{rec},d} o_l(t) \tag{20}$$

$$\text{arch. C}: I_k(t) = \sum_{d,i} W_{i,k}^{\mathtt{in},d} x_i(t) + \sum_{d,l} W_{l,k}^{\mathtt{mem},d} m_l(t) \tag{21}$$

$$\text{arch. D}: I_k(t) = \sum_{d,i} W_{i,k}^{\mathtt{in},d} x_i(t) + \sum_{d,l} W_{l,k}^{\mathtt{rec},d} o_l(t) + \sum_{d,n} W_{n,k}^{\mathtt{mem},d} m_n(t) \tag{22}$$

Finally we compare the learning speed of above developed architectures on the store-recall task. We repeat the same setup as in previous section. With goal to find the fastest learning architecture, we measure the number of training iterations needed to reach the error rate bellow 5%. Simulations that do not reach this threshold in 600 iterations are stopped and marked as failure. We perform this evaluation for store-recall task instances with ex-

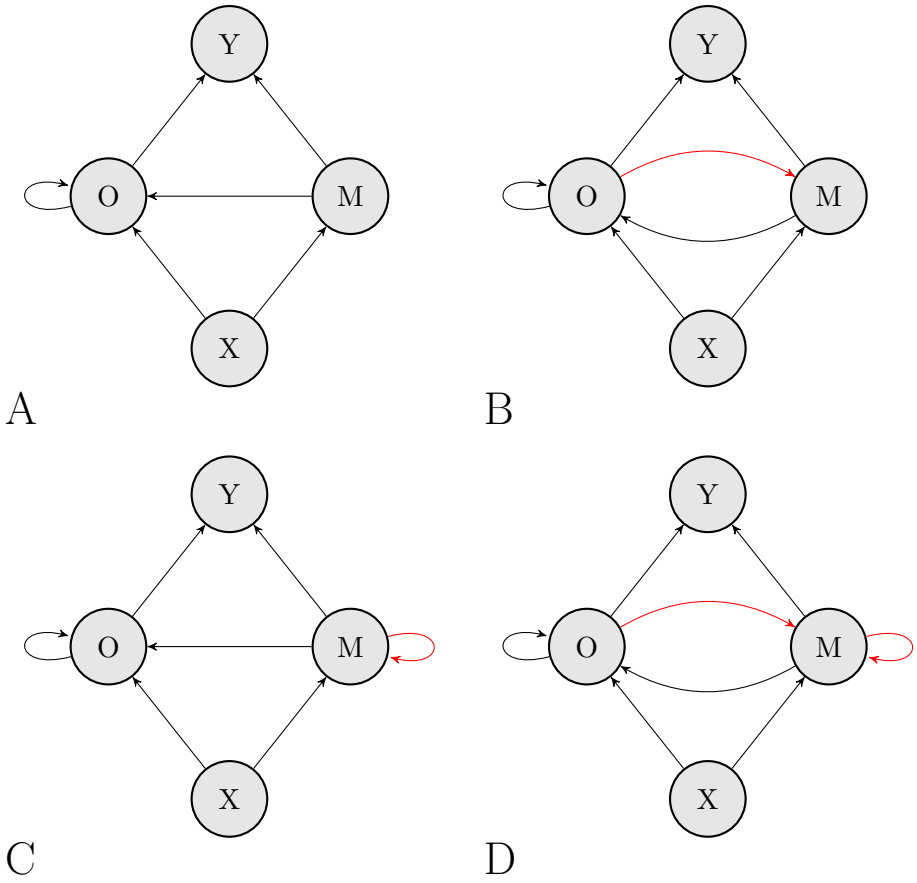

Figure 6: **SCRN architecture alternatives**: Proposed modifications to the original SCRN architecture. In out implementation M represents the pool of ALIF neurons while O represents the pool of LIF neurons (for more details see Section 4.2). Red edges represent the added synaptic connections. A: original SCRN inspired architecture. B: additional connection from recurrent LIF neurons to the ALIF neurons. C: recurrent connection within ALIF neurons. D: both changes from B and C effectively making a fully connected pool of mixed LIF and ALIF neurons.

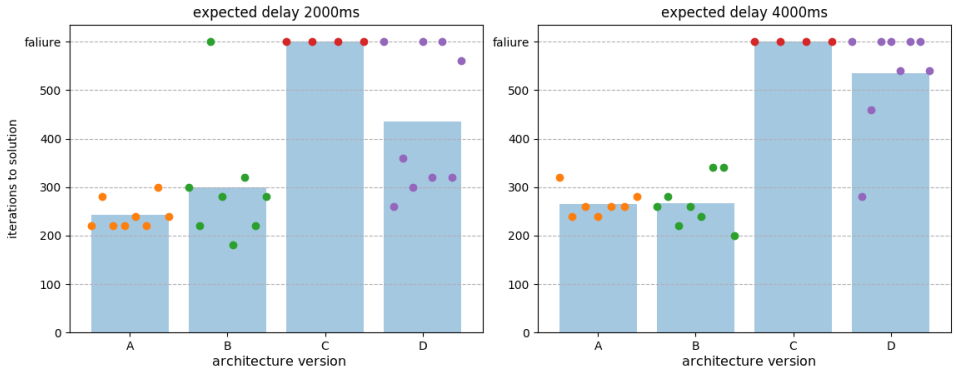

Figure 7: **Evaluation of alternative SMU network architectures**: Performance comparison of different architectures of SMU model on store-recall task. Points represent the number of training iterations needed to reach an error bellow 5% on the test set, each point representing a result from different random initialization. Bars that contain points are the mean values of results of individual architectures.

pected delay of 2000ms and 4000ms. Figure 7 shows the individual training times along with the mean values for every architecture.

Through this comparative analysis we have shown that the imposed structure on the network plays a large role in the resulting performance. Architectures C and D show significant degradation in learning speeds which indicates that their common factor, recurrence in the pool of ALIF neurons, has negative impact on the models capabilities to capture long term dependencies in sequential patterns.

Another discovery is the second promising architecture B which performs as well as original architecture A but with higher variance in learning speed. However, when considering only the best results reached by individual architecture, the architecture B is the winner. The additional connection from LIF to ALIF pool in architecture B might be responsible for better processing ability of the model but with a cost of additional parameters that need to be trained.

6.2 Sequential MNIST

The MNIST dataset [30] is a popular toy problem in the deep learning community. It is a dataset of handwritten digits and respective labels used as a supervised learning image classification task. Since image data is lacking the dimension of time, this dataset needed to be adapted for benchmarking of memory capable neural networks. This resulted in the sequential MNIST developed by Le et al. [31]. In this task the 784 pixels of an image are presented in a sequential manner to the recurrent network. Classifying such sequences requires a network capable of learning longer time dependencies

between the presented pixels.

This same problem has been tackled by different publications, which report very different performances using the LSTM networks. In the [31] the maximum test accuracy reached is ∼65% after 10^6 training iterations of an LSTM network with 100 hidden units. They also performed a grid search to optimize the initial forget gate bias to improve the learning of long term dependencies. In contrast to these results, in the [15] the authors report the performance of 97.96% without sharing any details on the training duration. They report also using single layer LSTM network with 100 hidden units, optimizing the network with RMSProp with momentum with learning rate 10^{-4}, and initializing forget gate to 1. Lastly the authors of [32] report the test accuracy of only 52% with 10k training iterations.

6.2.1 Pixel to spike encoding

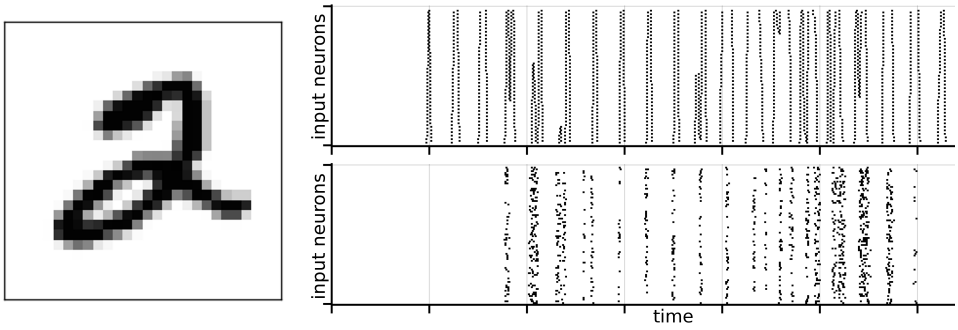

Figure 8: **Sequential MNIST input encoding**: Original MNIST image (left) is fed to the network one pixel at a time in a sequential manner from top left to bottom right. Artificial LSTM network is fed with analog pixel values directly, where as spiking network models require additional encoding of analog values to spikes. Spike rasters produced from the MNIST image on the left: Threshold crossings method (top right), Population rate encoding (probabilistic firing) method (bottom right)

Pixel values in the MNIST data sets are normalized to floating point number in range [0, 1]. To encode such pixel sequences to spikes we implemented two different approaches. First, a simple pool of neurons fire with probability of pixel value, effectively implementing the population rate encoding. The probabilities are scaled down so the stochastic neurons fire in the range [0, 200] Hz. Resulting spike trains are shown in the Figure 8 bottom right plot. The second approach was inspired by the onset-offset method used for audio preprocessing in [33]. Here we apply an onset and offset threshold operation on the pixel values in the sequence. Every threshold-direction tuple is assigned to separate neuron which would fire in respect to the change of the pixel value. Resulting spike trains from this method are shown in the Figure 8 top right plot.

In comparison we observe that the threshold crossing method of spike generation preserves much more information about the pixel values showing spikes in the train for even the very low pixel values. This of course depends on the number of thresholds, however the previous statement always holds if the number of neurons is matched by both methods. In contras the population rate encoding shows more degradation in the information detected (pixels with very low values are not detected), but much more irregular and biologically plausible firing rates. This method would require very high number of input neurons to detect the weak pixels with same accuracy as the threshold crossing method.

A potential improvement to the population rate encoding would be to apply a logarithmic transform to the pixel intensities making the input more sensitive to the weak pixels, analog to the human logarithmic perception of many stimuli (sound, light, etc.). Exploration of this potential improvement to the input preprocessing we leave for the future works, and focus here only on previously described methods.

With the two encoding methods described above, it is obvious that increasing the number of input neurons encoding the pixel values would increase the information resolution in the spike trains. This however comes with a cost of increase in parameter number for the network model which is significant for the effectiveness of the network.

Additionally for the resulting spiking input, we extend the sequence by 100ms and add a neuron that fires constantly for this period of time at the end of input. This serves as the output cue to the spiking models. During training, the softmax output of the spiking models is averaged over this cue period and the output channel with highest probability is used as final classification output of the network. This output is compared with the target values to compute the classification accuracy of the network. Plot of probabilities of the softmax output of a trained SMU network is shown in fourth row of Figure 9.

6.2.2 Evaluation

In the following we compare performances of different network models with different input encodings of sequential MNIST task. We first perform the experiment with recurrent networks of LIF neurons to serve as a baseline for spiking network performance. We fix the SMU network architecture to version B (see Section 6.1.3) and test it on both thresholding and population rate pixel encoding methods described in previous section. Finally we compare the results of spiking networks to the artificial LSTM network. To facilitate a fair comparison we match the number of parameters between the networks (∼66k), the batch size (512), and the number of training iterations (12k, 24k, 36k). An important factor to note is that the LSTM network has been fed with the original analog input pixel values, potentially giving it a

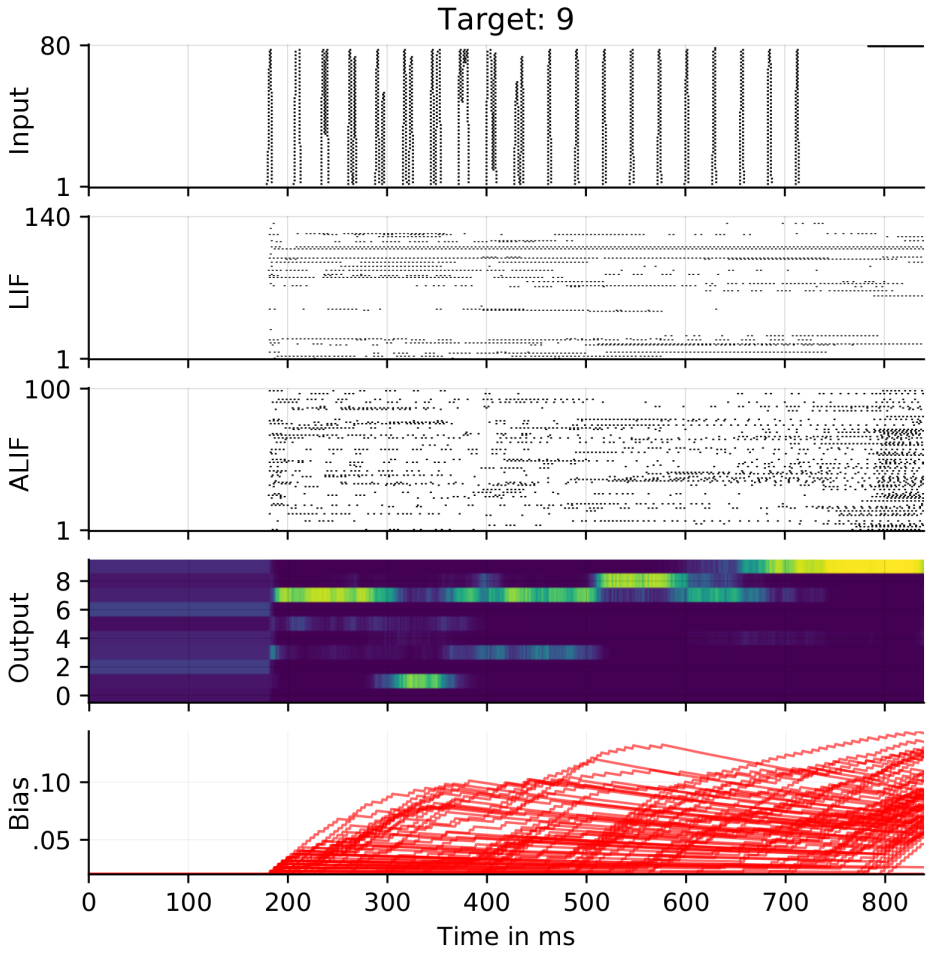

Figure 9: **Trained SMU network solving sequential MNIST**: SMU network classifying the sequential MNIST image of digit 9 encoded with onset-offset thresholding method. First three rows show raster plots of input, LIF, and ALIF neurons respectively. Fourth output row shows the plot of softmax probabilities over output classes. Last row shows plot of adaptive thresholds over time of the ALIF neurons.

significant advantage over the spiking network which are fed with the lossy input encoding. Further details of the training setups is described in Appendix B. Table 6 contains the accuracy on the test set of different network models after the training. The reported results are the best runs out of 20 random network initializations that showed the fastest learning speed in initial training period.

	τ_α (ms)	input type	test accuracy at		
			12k iter.	24k iter.	36k iter.
LIF		thresholding	$36 \pm 5.4\%$	$55 \pm 8.9\%$	$61 \pm 4.2\%$
SMU	700	population rate	$77 \pm 5.3\%$	$86 \pm 2.4\%$	$86 \pm 2.3\%$
SMU	[200, 2000]	thresholding	$80 \pm 6.3\%$	$89 \pm 1.7\%$	$92 \pm 1.4\%$
SMU	700	thresholding	$83 \pm 5.6\%$	$90 \pm 1.7\%$	$93 \pm 1.2\%$
LSTM		analog	$83 \pm 1.8\%$	$94 \pm 1.7\%$	$96 \pm 1.6\%$

Table 2: **Accuracy on sequential MNIST**: Performance comparison of different networks on sequential MNIST task. Presented are the best runs out of 20 random network initializations. Results are mean values and standard deviations of network classification accuracy on test batch containing 512 images.

With the results from Table 6 we can confirm that the information degradation through the population rate pixel encoding has a direct impact on the performance of spiking network (compare SMU results with different input types). Results also show the superiority of our SMU model to the recurrent network of LIF neurons in the context of task where temporal dependencies in the sequences that need to be exploited are over the 200ms limit.

Finally the highlight of this experiment is the performance of SMU network which approaches the performance of optimized artificial LSTM network (see last two rows of Table 6). We point out that both networks use the approximately the same number of parameters, and were trained for the same number of training iterations, while the LSTM network had an advantage of more precise analog input.

The SMU network with spread of adaptation time constant τ_α in [200, 2000]ms performed slightly worse then the same network with fixed $\tau_\alpha =$ 700ms. This is caused by the two factors. First, the potential maximum of the required time constant to hold the information from beginning of input is 700ms. This indicates that ALIF neurons with lower time constants would not be able to successfully store the information until the output cue. Second, with the onset offset thresholding method of encoding the pixels, there is no redundancy in the input (as opposed to store-recall task where values are presented for $\tau_c$ms which allows even the ALIF neurons with short adaptive time constant to significantly change their threshold bias which can then decay sufficiently long until the recall output cue). Both of these factors contributed to training difficulty of this network which could

potentially match the performance of SMU network with fixed τ_α. It is important to note that the SMU network with spread of τ_α values is much more flexible and applicable to tasks with very different memory demands.

Raster plots of trained SMU network solving the sequential MNIST classification of onset-offset threshold encoded image of digit 9 is shown in Figure 9. Additional plots for the trained network solving the task encoded with population rate method is shown in Figure 15 (in Appendix B).

6.3 Speech recognition task

In this section we present the supervised learning phoneme classification task on TIMIT corpus. The dataset on which we train the model is described in the Section 6.3.1. The encoding of audio signals to spike trains by Lyon's cochlear model is described in Section 6.3.2. Finally we evaluate and compare models in Section 6.3.3.

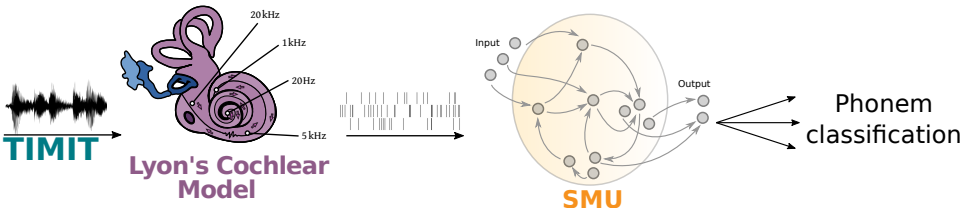

Figure 10: **Setup for audio classification with spiking models**: High level view of setup used to solve TIMIT phoneme classification task with spiking models.

Motivation for solving a speech recognition task is twofold. First it is a real world problem still being researched by the machine learning communities. Methods used to solve it are continuously being improved by many handcrafted algorithms, mostly using artificial neural networks. So far, to the best of our knowledge, no work has attempted to solve this task using spiking neural networks. Second, the task is something human brain solves on the daily basis, making it a well fitted task for a biologically plausible spiking network models.

With this in mind, during our experiments in the following sections we will opt for the more biologically plausible methods in different aspects of experiments. We acknowledge that this will result in worse performance of our spiking models then if we optimized all aspects of the experiments toward maximum performance.

6.3.1 TIMIT corpus

The TIMIT corpus is a collection of reading speech data made with automatic speech recognition systems evaluation as its primary target [34]. The

original label		target label
aa, ao	→	aa
ah, ax, ax-h	→	ah
er, axr	→	er
hh, hv	→	hh
ih, ix	→	ih
l, el	→	l
m, em	→	m
n, en, nx	→	n
ng, eng	→	ng
sh, zh	→	sh
pcl, tcl, kcl, bcl, dcl, gcl, h#, pau, epi	→	sil
uw, ux	→	uw
q	→	–

Table 3: **TIMIT phoneme reduction**: Reduction of the number of phoneme labels in the corpus is achieved through mapping of multiple phonemes in the left column to a single label in the right column. The phoneme 'q' is discarded. This mapping reduces the the original 61 phonemes to 39 labels.

corpus contains recordings of 630 speakers reading ten phonetically rich sentences. The speakers dialects belong to the eight major dialects of American English, providing the high variation in the utterance of phonemes. A data point (single utterance of a sentence) from the corpus consists of recorded waveform, and phonetic, orthographic, and word labels which are time-aligned. The waveform recording are 16-bit with 16kHz sampling rate. All the labels contained are hand verified. Additionally test and training groupings are balanced for phonetic and dialectal coverage [34].

The transcription labels of this corpus belong to the dictionary of 61 phonemes. Throughout experiments in this chapter we us a reduced set of phonemes which is a common practice from speech recognition domain [35]. We achieve this by mapping multiple phonemes to single label using the common mapping from [36] shown in Table 3. Through this reduction the resulting phoneme dictionary contains 39 entries.

6.3.2 Audio to spike encoding

State of the art ANNs commonly use the Mel-frequency cepstral coefficients (MFCCs) as the audio preprocessing method of choice for tasks related to speech recognition [37]. Many authors have been credited for development of this method, most notable ones include Paul Mermelstein, John S. Bridle and M. D. Brown.

There exist many variations of the algorithm for deriving MFCCs, but the essential steps include:

- Fourier transform of windowed signal
- Mapping the powers of previous step on the mel scale, using triangular overlapping windows
- Applying logarithm to the powers at each mel frequency
- Discrete cosine of previous result
- Finally the output is made of amplitudes of the spectrum from previous step

Even thou the MFCCs are a powerful and proven audio preprocessing methods for speech recognition methods, it is ill-suited for the spiking network models. In context of human speech recognition and audio processing in general, the preprocessing task of the incoming sound waves is performed by ear organs. More specifically the outer and middle ear partially process the acoustic pressure waves which are finally transformed into nerve firings in the cochlea, located in the inner ear. The resulting spike trains are used in the higher levels of the brain for speech recognition and related tasks.

A well known biologically realistic audio preprocessing method is the Lyon's Cochlear Model [38]. This method provides us with a simple model for the middle and outer ear and most importantly the input/output characteristics of the cochlea from the inner ear. The output of this cochlear model is a discrete time series of neuron spiking probabilities. Sampling from these probabilities produces spike trains which are compatible with our discretized simulation of spiking neurons (described in Section 3). This is equivalent to the population rate encoding we investigate in previous sequential MNIST experiment where we learn the inferiority of this encoding method to the

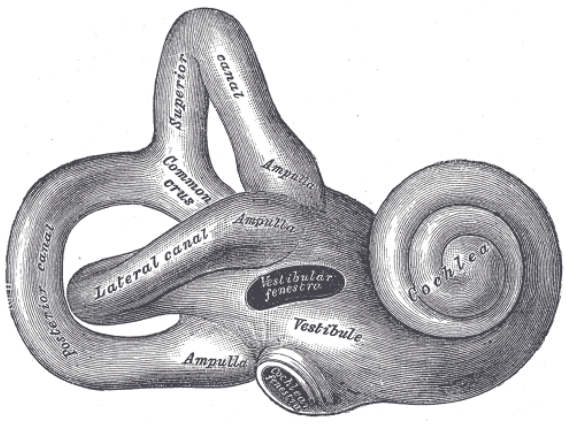

Figure 11: **Inner ear anatomy**: Right osseous (bony) labyrinth of the inner ear showing the cochlea (spiral-shaped cavity). Lateral view.

thresholding method. Nevertheless, for this experiment we will use the population rate encoding since this is the intended and biologically justified encoding to spikes through cochlear model.

For the purposes of our experiments we used the Malcolm Slaney's implementation of Lyon's passive cochlear model in the Auditory Toolbox [39] to produce the firing probability matrices.

6.3.3 Evaluation

Despite the popularity of the TIMIT corpus as a task used for evaluating recurrent models, in the following analysis we show that relatively short memory is required for successfully solving the task. Through statistical analysis of the corpus sampled with 1ms window we discover that the maximal length of single phoneme in the corpus does not cross the 438ms limit. Overall the mean phoneme length is only 77ms in the training dataset, and the mean length for the longest phoneme is only 160ms. Figure 12 shows the distribution of maximum phoneme lengths in milliseconds for half of the randomly chosen phonemes in the corpus. Fact that most state of the art results were achieved through applying LSTM on MFCC sampled with 10ms window size (resulting in average phoneme length of 7.7 steps) point to the relatively short memory demand of the task. These statistical properties

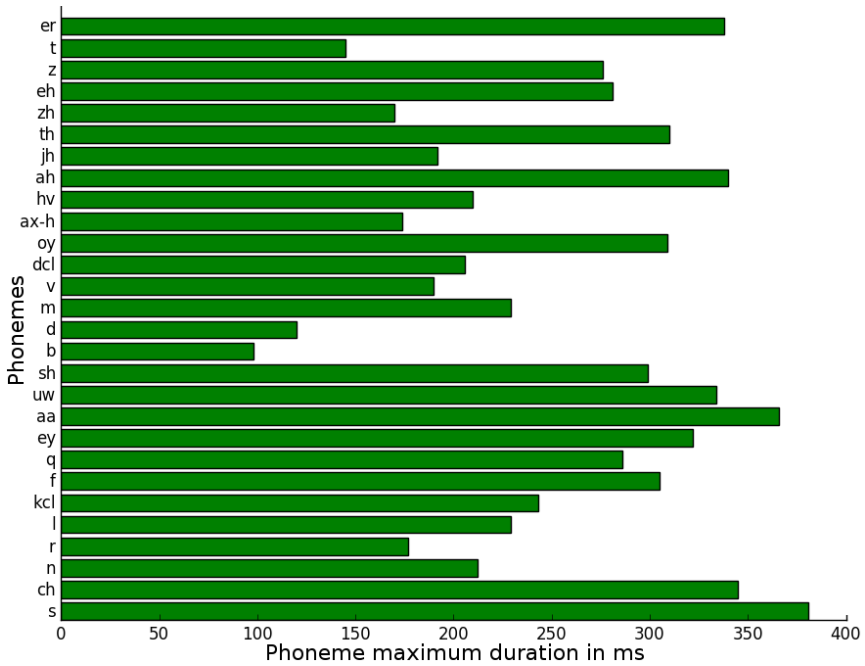

Figure 12: **TIMIT maximum phoneme lengths in ms**: Maximum pronunciation time per phoneme in the train set of TIMIT corpus. For brevity, only half of the phonemes from the training set are shown.

suggests that the ability of SMU to learn long term dependencies will not be exploited in this task. Based on our findings during store-recall task, we would expect that the LIF network would perform better on TIMIT task then the SMU model.

For population rate encoding of the audio data through Lyon's cochlear model we observed that sampling with 1ms resolution produced too noisy input which proved to be detrimental to the performance of spiking networks. To mitigate this we sample the input with 10ms resolution which we then stretch in time by repeating every step 10 times to produce a more stable input of the required length. The resulting firing probability matrix is also extended in space, duplicating every channel 5 times with aim to increase the sampling precision.

Training recurrent neural networks on TIMIT corpus is very computationally intensive. In an attempt to speed the training process, we split the input sequences in the chunks of 600ms. Network state at the end of one such chunk is transfered to the start of the next corresponding chunk. This adaptation only has an effect on the gradient propagation in time which in this case is limited to only 600ms. This resulted in noticeably faster training of the networks without the performance loss.

To evaluate our previous claim that only relatively short memory is required for successful learning of TIMIT task, we perform the simulation of LSTM network with sequence chunking to very short 200ms chunks which prevents any gradient propagation beyond this time limit. We observed no performance degradation in this setup which confirms our initial task analysis and conclusion.

LSTM network used in the simulations contain a single LSTM layer of 200 hidden units. Input to LSTM network is (1ms window) MFCC with 13 channels combined with first and second derivative of the same, resulting in 39 input channels. We fix the SMU network architecture to version B (see Section 6.1.3). For fair comparison we construct the SMU network with the same number of parameters as LSTM network. Lyon's cochlear model has 86 channels at output, after 5 times repetition, we have 430 input channels. This limits the SMU network to 182 LIF and 100 ALIF neurons to match the number of parameters of LSTM network. For recurrent LIF network the number of neurons is 272 to match this constraint. Details on the number of parameters per network model is described in Appendix C.

Additionally we include the results of Graves and Schmidhuber [1] who also trained a unidirectional LSTM network on a more challenging version of TIMIT using the complete set of 61 phonemes with MFCCs extended through first derivative and log-energy.

All networks have been trained with batch size 500 and up to 100 epochs, using Adam optimizer with learning rate of 0.01.

	input type	no. target phonemes	accuracy	
			test	validation
LIF	Lyon's cochlear	39	65.8%	67.2%
SMU	Lyon's cochlear	39	63.5%	65.6%
LSTM	MFCC $+\Delta + \Delta\Delta$	39	68.6%	70.9%
LSTM [1]	MFCC $+\Delta+$ logE	61	64.6%	–

Table 4: **Accuracy on TIMIT**: Performance comparison of different networks on TIMIT phoneme classification task. Last row is the result of unidirectional LSTM from [1]. Other rows are results from simulations we perform on TIMIT with reduced phoneme set.

From the experiment results shown in Table 7 we observe that the recurrent network of LIF neurons is more effective at solving the TIMIT task then the SMU network. Our previous analysis of TIMIT corpus align with the experimental results. We can conclude that the task of phoneme prediction does not require longer memory then the length of one phoneme which are largely in the range of up to 200ms. This supports our finding with the store-recall task showing superiority of LIF neurons on processing the sequences of shorter length.

Regardless of the model used, the spiking network managed to approach the performance of LSTM network in the unfavorable setting and same training period. To the best of our knowledge this is the first result of spiking networks performing comparably to the artificial networks on such a demanding task. Additional modifications to the task setup like introduction of target delays as described in [1] improve the performance of artificial networks. Since the speed of information processing in the spiking models is most likely limited by synaptic delays, we postulate that this introduction of delay in targets would in the same manner improve the performance of spiking models by allowing more time for information processing.

Further, we point out that no hyper-parameter optimization was used to achieve the results reported in this section. It is highly unlikely that extensive hyper-parameter search for our spiking models would not yield in even better performance on the TIMIT speech recognition task.

7 Conclusion and Discussion

LSTM artificial networks dominate the machine learning community as the primary model of recurrent networks. In this work we aimed to develop a spiking network model that would exhibit much of the capabilities of the LSTM model. We developed a model that uses mixture of LIF neurons with and without adaptive excitability in a network with imposed structure. To evaluate this model we based our approach on the internally developed SSD-BPTT which allows the training of spiking networks (in high resolution discrete time) in the same manner as the artificial recurrent networks.

We evaluate the model on three tasks that require memory:

- Store-recall task: test the models capability to store and recall input values over arbitrary long periods of time.
- Sequential MNIST task: test if the model is able to recognize complex sequence patters and extract the common relationships between sequence classes
- TIMIT task: test the models capability in classifying the spoken phonemes

Through the store-recall task we have showed the functional difference between novel SMU network and recurrent LIF network. We showcased the ability of our model to memorize values for arbitrary long times and the control of this ability by a single time constant parameter in the threshold adaptability of the ALIF neurons. Further we gained insights into the role and capabilities of recurrent LIF networks to effectively store and process values for short periods of time.

Using sequential MNIST task we make a significant leap in the performance of spiking networks. We show that our model can approach the performance of ANNs even in the same training setup and with the same number of parameters, despite of using a noisy input as opposed to ANN. Significance of this milestone is further supported by the facts: The spiking models are much more energy efficient when implemented in neuromorphic hardware (as opposed to ANNs); Our model is biologically plausible and can contribute to further brain research related to the topic of (working) memory.

Finally we reevaluate our functional analysis of SMU and LIF networks through phoneme classification task. Here we confirm our initial conclusion that the LIF networks are more efficient at processing shorter sequences.

A natural follow up to this work would be investigating the robustness of SMU model one wider variety of tasks, most notably evaluation in the context of reinforcement learning tasks.

Appendices

A Appendix: store-recall task

This appendix contains additional raster plots of trained SMU networks solving store-recall task. Table 5 contains parameter details of the figures 13 and 14. This setup demonstrated the robustness of SMU model to solve store-recall task for longer delays without increasing the time available for storing a value by expanding the character step τ_c.

	τ_c	sequence length	p_{SR}	delay
Figure 13	250ms	24	0.1	4250ms
Figure 14	250ms	48	0.1	8750ms

Table 5: Figure specific parameters of store-recall task modification.

B Appendix: sequential MNIST task

In this section we present the omitted details of the sequential MNIST task setup in Table 6. We also include a raster plot of trained SMU network solving the sequential MNIST task while using the population rate encoding of the input pixels.

	input type	num.params.	num.units
LIF	thresholding	66000	i:80 o:220
SMU	population rate	66800	i:80 o:140 m:100
SMU	thresholding	66800	i:80 o:140 m:100
LSTM	analog (original)	66048	i:1 h:124

Table 6: Parameter details of different networks evaluated on sequential MNIST task. In the number of units column: for LSTM ANN h represents number of hidden units and i the number of input channels; for SMU and LIF SNNs i represents number of input neurons, o number of LIF neurons, and m number of ALIF neurons

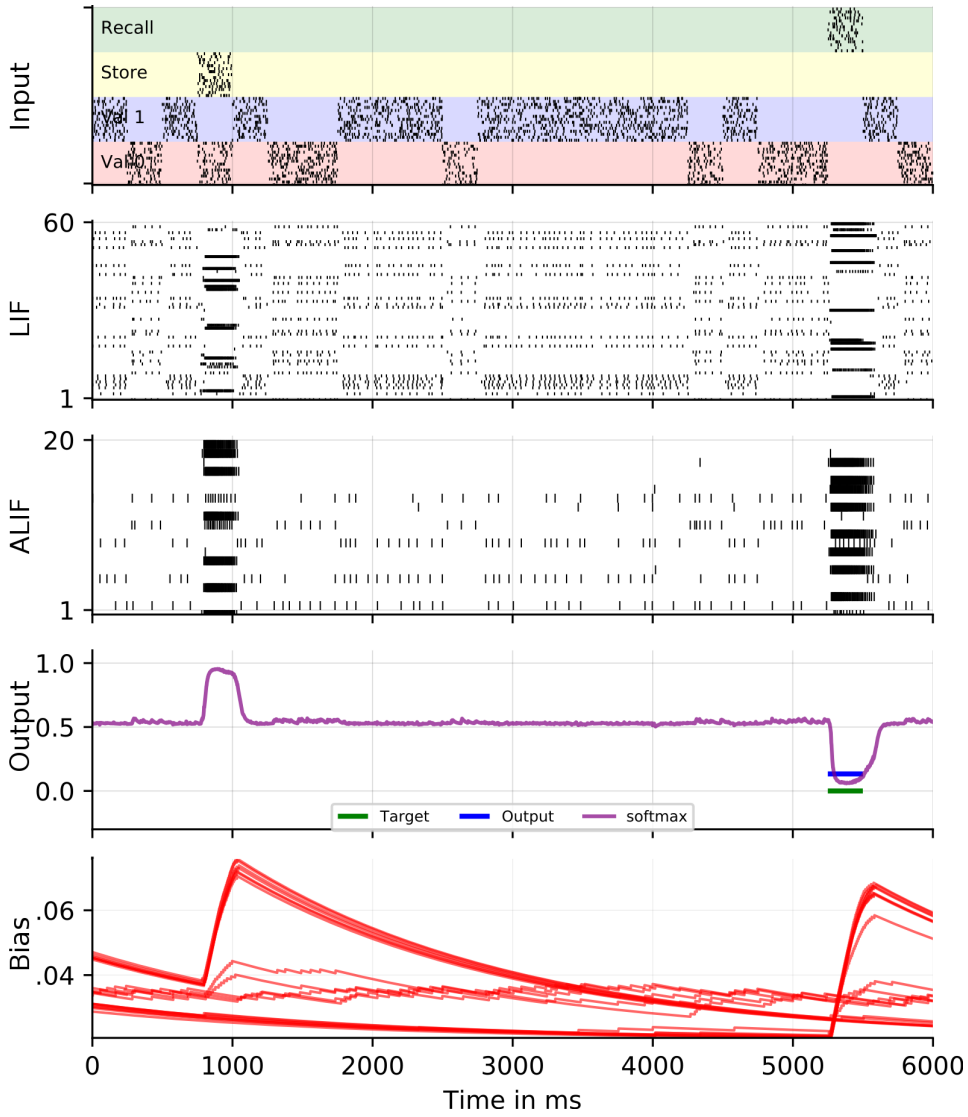

Figure 13: Trained SMU solving recall task with $\tau_c = 250$ms, sequence length 24 and $p_{SR} = 0.1$. This sequence contains delay of 4250ms between the store-recall signals. From top to bottom: Spike raster of input neurons, spike raster of LIF neurons, spike raster of ALIF neurons, softmax output (with target and average of softmax for character step), and bias plot of ALIF neurons.

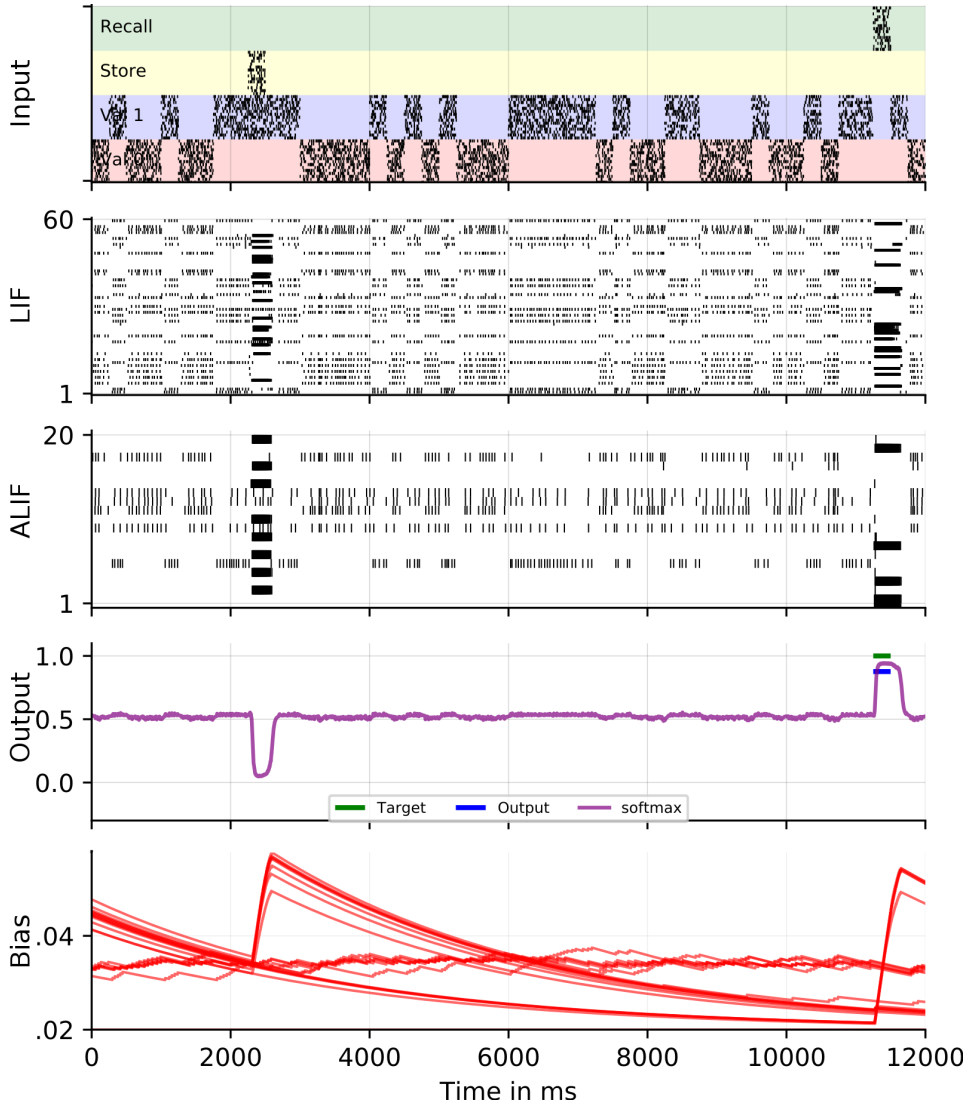

Figure 14: Trained SMU solving recall task with $\tau_c = 250$ms, sequence length 48 and $p_{SR} = 0.1$. This sequence contains delay of 8750ms between the store-recall signals. From top to bottom: Spike raster of input neurons, spike raster of LIF neurons, spike raster of ALIF neurons, softmax output (with target and average of softmax for character step), and bias plot of ALIF neurons.

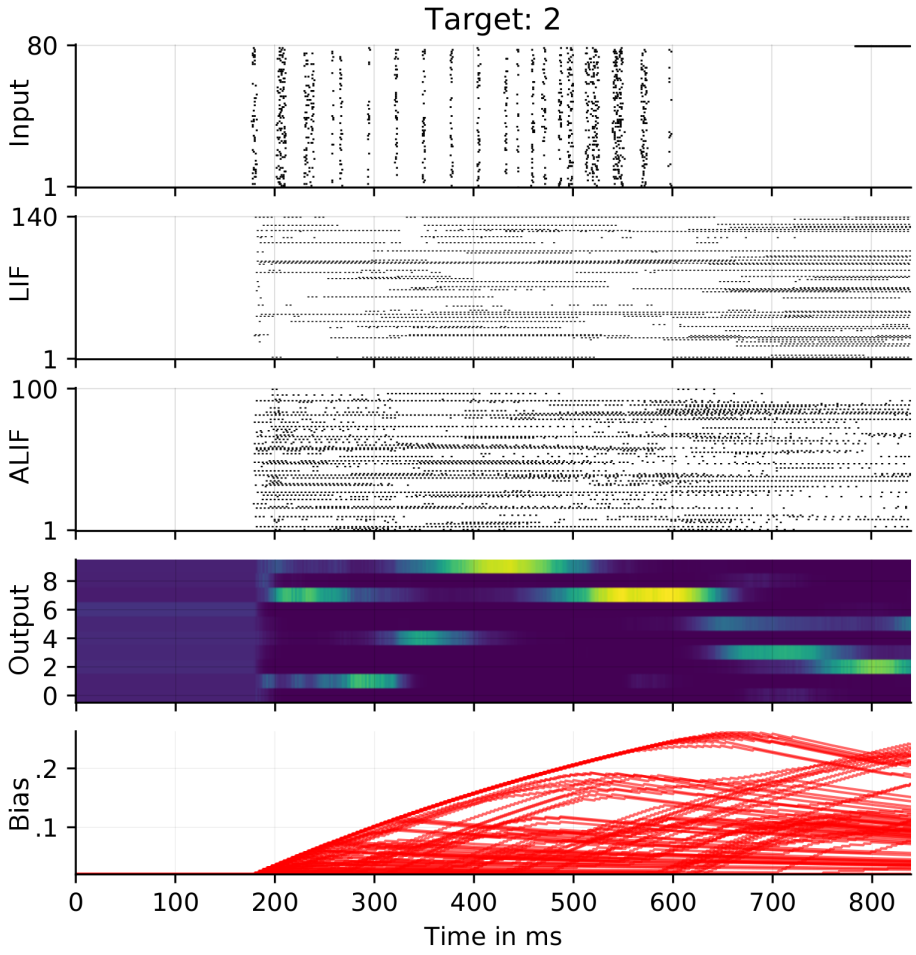

Figure 15: Trained SMU solving recall task with $\tau_c = 250$ms, sequence length 48 and $p_{SR} = 0.1$. This sequence contains delay of 8750ms between the store-recall signals. From top to bottom: Spike raster of input neurons, spike raster of LIF neurons, spike raster of ALIF neurons, softmax output (with target and average of softmax for character step), and bias plot of ALIF neurons.

C Appendix: TIMIT task

	num.params.	num.units
LIF	190944	i:430 o:272
SMU	190784	i:430 o:182 m:100
LSTM	191200	i:39 h:200

Table 7: Parameter details of different networks evaluated on TIMIT task. In the number of units column: for LSTM ANN h represents number of hidden units and i the number of input channels; for SMU and LIF SNNs i represents number of input neurons, o number of LIF neurons, and m number of ALIF neurons

D References

[1] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5-6):602–610, jul 2005.

[2] Patricia S. Churchland, Christof Koch, and Terrence J. Sejnowski. Computational neuroscience. chapter What is Computational Neuroscience?, pages 46–55. MIT Press, Cambridge, MA, USA, 1993.

[3] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. 9:1735–80, 12 1997.

[4] Rahul Dey and Fathi M. Salem. Gate-variants of gated recurrent unit (GRU) neural networks. *CoRR*, abs/1701.05923, 2017.

[5] Wulfram Gerstner and Werner Kistler. *Spiking Neuron Models: An Introduction*. Cambridge University Press, New York, NY, USA, 2002.

[6] Wofgang Maas. Networks of spiking neurons: The third generation of neural network models. *Trans. Soc. Comput. Simul. Int.*, 14(4):1659–1671, December 1997.

[7] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, Dec 1943.

[8] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, pages 65–386, 1958.

[9] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[10] Augusto Buchweitz. Models of working memory: Mechanisms of active maintenance and executive control. 04 2008.

[11] G.A. Miller. *Plans and the structure of behavior*. A Holt - Dryden book. Holt, 1960.

[12] Patricia S. Goldman-Rakic. Working memory and the mind. 267:110–7, 10 1992.

[13] Adele Diamond. Executive functions. *Annual Review of Psychology*, 64(1):135–168, jan 2013.

[14] Nelson Cowan. Chapter 20 what are the differences between long-term, short-term, and working memory? In *Progress in Brain Research*, pages 323–338. Elsevier, 2008.

[15] R. Ponte Costa, Y. M. Assael, B. Shillingford, N. de Freitas, and T. P. Vogels. Cortical microcircuits as gated-recurrent neural networks. *ArXiv e-prints*, November 2017.

[16] Roeland Nusselder Sander Bohte Davide Zambrano, Isabella Pozzi. Gating out sensory noise in a spike-based long short-term memory network, 2018.

[17] Randall C. O'Reilly and Michael J. Frank. Making working memory work: A computational model of learning in the prefrontal cortex and basal ganglia. *Neural Computation*, 18(2):283–328, feb 2006.

[18] Kai A. Krueger and Peter Dayan. Flexible shaping: How learning in small steps helps. *Cognition*, 110(3):380–394, mar 2009.

[19] David Daniel Cox and Thomas Dean. Neural networks and neuroscience-inspired computer vision. *Current Biology*, 24(18):R921–R929, sep 2014.

[20] Adam H. Marblestone, Greg Wayne, and Konrad P. Kording. Toward an integration of deep learning and neuroscience. *Frontiers in Computational Neuroscience*, 10, sep 2016.

[21] Demis Hassabis, Dharshan Kumaran, Christopher Summerfield, and Matthew Botvinick. Neuroscience-inspired artificial intelligence. *Neuron*, 95(2):245–258, jul 2017.

[22] Upinder S. Bhalla. Dendrites, deep learning, and sequences in the hippocampus. *Hippocampus*, oct 2017.

[23] Tomas Mikolov, Armand Joulin, Sumit Chopra, Michaël Mathieu, and Marc'Aurelio Ranzato. Learning longer memory in recurrent neural networks. *CoRR*, abs/1412.7753, 2014.

[24] Jeffrey L. Elman. Finding structure in time. *COGNITIVE SCIENCE*, 14(2):179–211, 1990.

[25] Guillaume Bellec et al. Medium-term synaptic plasticity endows networks of spiking neurons with functional properties of LSTM networks. *Internal report of Institute for Theoretical Computer Science TU Graz*, 2017.

[26] Matthieu Courbariaux and Yoshua Bengio. Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1. *CoRR*, abs/1602.02830, 2016.

[27] Steven K. Esser, Paul A. Merolla, John V. Arthur, Andrew S. Cassidy, Rathinakumar Appuswamy, Alexander Andreopoulos, David J.

Berg, Jeffrey L. McKinstry, Timothy Melano, Davis R. Barch, Carmelo di Nolfo, Pallab Datta, Arnon Amir, Brian Taba, Myron D. Flickner, and Dharmendra S. Modha. Convolutional networks for fast, energy-efficient neuromorphic computing. *CoRR*, abs/1603.08270, 2016.

[28] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever. Evolution Strategies as a Scalable Alternative to Reinforcement Learning. *ArXiv e-prints*, March 2017.

[29] Allen Institute for Brain Science. Allen Brain Atlas - Cell Feature Search. `http://celltypes.brain-map.org/data`. [Online; accessed 22-January-2018].

[30] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.

[31] Quoc V. Le, Navdeep Jaitly, and Geoffrey E. Hinton. A simple way to initialize recurrent networks of rectified linear units. *CoRR*, abs/1504.00941, 2015.

[32] Junier B. Oliva, Barnabás Póczos, and Jeff G. Schneider. The statistical recurrent unit. *CoRR*, abs/1703.00381, 2017.

[33] Robert Gütig. Spiking neurons can discover predictive features by aggregate-label learning. *Science*, 351(6277), 2016.

[34] William M. Fisher Jonathan G. Fiscus David S. Pallett Nancy L. Dahlgren Victor Zue John S. Garofolo, Lori F. Lamel. TIMIT Acoustic-Phonetic Continuous Speech Corpus. `https://catalog.ldc.upenn.edu/ldc93s1`, 1993. [Online; accessed 19-November-2017].

[35] Robinson A.J. Several improvements to a recurrent error propagation network phone recognition system. *Technical Report CUED/F-INFENG/TR82, University of Cambridge*, 1991.

[36] K. F. Lee and H. W. Hon. Speaker-independent phone recognition using hidden markov models. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(11):1641–1648, Nov 1989.

[37] Todor Ganchev, Nikos Fakotakis, and George Kokkinakis. Comparative evaluation of various mfcc implementations on the speaker verification task. In *in Proc. of the SPECOM-2005*, pages 191–194, 2005.

[38] M. Slaney and R.F. Lyon. *Lyon's cochlear model*. Apple technical report. Apple Computer, Advanced Technology Group, 1988.

[39] Malcolm Slaney. Auditory Toolbox. `https://engineering.purdue.edu/~malcolm/interval/1998-010/`, 1998. [Online; accessed 25-November-2017].