Temporal Learning with Biologically Fitted SNN Models

Yuan Zeng, Lehigh University, USA, yuz615@lehigh.edu
Terrence C Stewart, National Research Council Canada, Canada, terrence.stewart@nrc-cnrc.gc.ca
Zubayer Ibne Ferdous, Lehigh University, USA, zui216@lehigh.edu
Yevgeny Berdichevsky, Lehigh University, USA, yeb211@lehigh.edu
Xiaochen Guo, Lehigh University, USA, xig515@lehigh.edu

DOI: https://doi.org/10.1145/3477145.3477153
ICONS 2021: International Conference on Neuromorphic Systems 2021, Knoxville, TN, USA, July 2021

Spiking Neural Networks (SNN) can model biological neural networks with different levels of details. There are trade-offs between model fidelity and computation efficiency. Which model is the most appropriate one to use depends on the goal and the computation task. Temporal learning is an important feature of the brain, which requires neural networks to integrate information from the past to solve present computation tasks. Prior work has proposed different SNN models for temporal learning, which includes the Leaky-Integrate-and-Fire (LIF), the Adaptive Leaky-Integrate-and-Fire (ALIF), and the Exponential Adaptive Leaky-Integrate-and-Fire (AdEx). These models capture different biological details and exhibit different learning properties.

This work aims to compare the model fidelity and learning performance of these three SNN models. Experimental data for *in vitro* living neural networks is used to first fit parameters of these three models. An automatic fitting tool is used to match the precise spike timing of the *in vitro* neurons and the modeled neurons. ALIF and AdEX can match with the spiking timing of the biological neuron better than the LIF does. The fitted models are then compared on a delay task, where the network needs to output values that were input into the network in the recent past. To compute the delay task, the Neural Engineering Framework (NEF) is used to implement a Legendre Memory Unit. Good performance is demonstrated on the delay task using ALIF, which suggests the possibility of implementing the algorithm on *in vitro* living neural networks. This work proposes a new neuron parameter fitting

approach, compares three SNN models, and is the first to use detailed adaptive neurons on the delay task with the NEF approach.

CCS Concepts: • **Networks** → **Network performance modeling**; • **Networks** → **Network experimentation**;

Keywords: spiking neural networks, living neural networks, spike frequency adaptation, precise spike timing, temporal learning, neural engineer framework, legendre memory unit

ACM Reference Format:
Yuan Zeng, Terrence C Stewart, Zubayer Ibne Ferdous, Yevgeny Berdichevsky, and Xiaochen Guo. 2021. Temporal Learning with Biologically Fitted SNN Models. In *International Conference on Neuromorphic Systems 2021 (ICONS 2021), JULY 27-29, 2021, Knoxville, TN, USA.* ACM, New York, NY, USA 8 Pages. https://doi.org/10.1145/3477145.3477153

1 INTRODUCTION

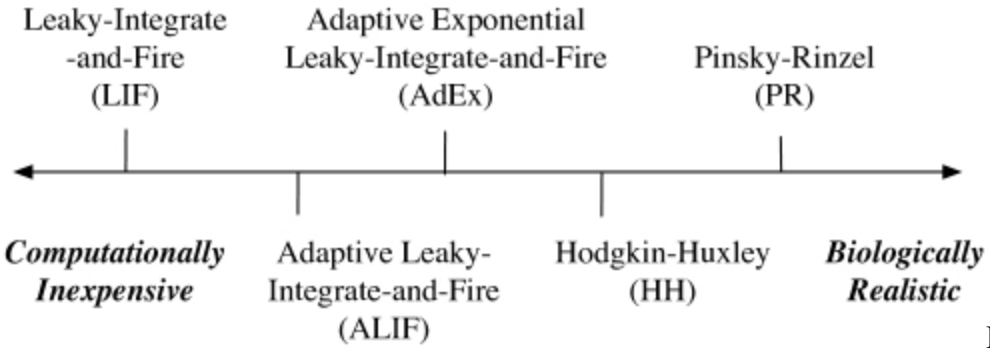


Figure 1: SNN model comparison.

Spiking Neural Networks (SNN), which are thought to be the next generation of neural networks [15], mimic biological neural networks more closely than traditional non-spiking Artificial Neural Networks (ANN) do. SNNs not only model neuronal and synaptic dynamics but also incorporate the concept of time. They are widely used in the area of computational neuroscience to study brain functions [13] [8], as well as in the area of artificial intelligence to explore new learning algorithms and energy-efficient neuromorphic implementations in solving real-world problems [9] [5] [14].

There are many kinds of SNN models [18] [22] [11] [17] [23] [10], which capture different levels of details of a biological neural network using one or several differential equations. These equations represent the electrical properties of neurons, synapses, dendrites, and axons, and aim to match some observed behaviors captured by experimental data [16]. The most commonly used one is the Leaky Integrate-and-Fire (LIF) model [18], which represents the lipid bilayer of a cell membrane as capacitance and the leaky channel across that membrane, mostly made up of chloride ions, as conductance. The Hodgkin-Huxley (HH) [11] model further simulates the sodium current and potassium current. Unlike the LIF model, where a spike happens when the membrane potential crosses a pre-defined threshold, a spike in the HH model is implicitly generated through the dynamics of the model. Furthermore, the HH model describes how the membrane potential changes with time when an action potential occurs more precisely. More

complicated models, such as the Pinsky-Rinzel (PR) [26] model, add calcium current and calcium-dependent potassium current. This allows the model to represent more realistic and more complex neuron properties, which includes the spike frequency adaptation behavior, where the firing rate of a specific neuron reduces through the time when a constant injection current is given. The PR model also describes the soma and the dendrite as two compartments interacting through a coupling conductance. By increasing the number of compartments, more complicated dendritic and axonal morphology can be captured [29].

There are trade-offs between "biologically realistic" and "computationally inexpensive" among different SNN models. Solving differential equations can be time-consuming through computer simulations and the time increases dramatically when more state variables and parameters are added into the model. The Pinsky-Rinzel model has eight state variables and more than 20 parameters, while the leaky integrate and fire model has only one state variable and five parameters. Many prior works attempt to reduce the model complexity while keeping the important feature. For instance, the Adaptive Leaky-Integrate-and-Fire (ALIF) model [23] captures the spike frequency adaptation behavior; the Adaptive Exponential Leaky-Integrate-and-Fire (AdEx) model takes account of the membrane potential changes when an activation happens [10]. All these models reduce the computational complexity as compared to the PR and the HH model.

Which model is appropriate to use depends on the purpose. For SNN learning algorithm designs, the early works are mostly focusing on capturing the spike activity to achieve a more energy-efficient neuromorphic implementation. Therefore, LIF is the most popular choice [33] [27]. Some recent works [25] [3] have shown that more detailed neuron features, such as spike frequency adaptation, play an important role in capturing the long-term temporal information and improves the capability of the algorithm. In these works, ALIF is used for algorithm exploration. In addition, there are also attempts to use *in vitro* living neural networks to perform the learning tasks instead of the silicon implementation [34] [35] [19]. Dissociated animal cortex maintained in physiological conditions within an adhesive dish could make random synaptic connections with each other. The *in vitro* neural cultures could respond to stimuli delivered through microelectrode arrays or optogenetics interface [4] [12] [21] [28] [32], which can be used to perform learning tasks. Prior works [34] [35] in this direction simulate detailed SNN models such as HH and PR for living neural network algorithm design before conducting experiments on the *in vitro* neural cultures.

This paper studies learning algorithms that would be suitable for living neural networks. In contrast with earlier attempts in this direction, this work focuses on exploring the temporal learning property for recurrent networks, with the consideration of the precise spike timing. For temporal learning tasks, the neuron networks need to integrate information from the recent past into current computational processing, which is an important feature of the brain. There are some existing learning algorithms proposed for recurrent spiking neural network on temporal tasks [3] [31] [36], however, those algorithms are designed for silicon implementations and the model parameters are chosen either for computational simplicity (e.g., setting the resting potential to zero) or algorithm performance (e.g., setting the membrane or synapse constant based on the task requirement). It is not clear whether these algorithms still perform well with realistic neuron parameters. On the other hand, previous works [34] [35] simulate detailed

neuron models such as HH and PR to capture more realistic biological properties to explore learning algorithms for living neural networks, which are not scalable due to the computational complexity of these models. Since the temporal learning task focuses only on the spike timing instead of other detailed neuron features, this work uses simplified spiking neuron models in the study.

In this paper, we fitted three existing simplified neuron models (LIF, ALIF, and AdEx) to biological neuron data at a given current. The goal of the task is to find the best parameters in these models to match the exact timing of each spike between the experimental recording and the simulation. A machine learning based parameter tuning tool is used [1]. Results show that without adaptation, the models (e.g., LIF) cannot capture the spike timing when multiple spikes happen. ALIF and AdEx models can both match the spike timing well. The AdEx model has a better fit on the neuron's after-hyperpolarization voltage, but the ALIF model is more computationally efficient, which is a better choice for the temporal learning task.

We then test whether a network of ALIF neurons with the fitted parameters can be trained to perform learning tasks. A temporal delay task is considered in particular, where the output of the network should be the input to the network at some point in the past. This requires building a recurrent neural network, and the Neural Engineering Framework is used [7] [30] to implement a Legendre Memory Unit [31]. That is, we construct a differential equation that converts a scalar input into an internal state representation that uses shifted Legendre polynomials to represent some window of the past history of the input. We then train a single-hidden-layer feed-forward neural network and then connect it back to itself to produce the final recurrent model. This task has previously been done with LIF neurons [31] and with physical analog neurons [20], but not with adaptive spiking neuron models such as ALIF. Results in this paper suggest that, by taking the exact spike timing into account in the algorithm design process, the ALIF model with fitted parameters performs well in the delay task. This suggests the possibility of implementing the same algorithm with living neural networks in the future.

Contributions of this work are: 1) This paper proposes a general approach for fitting neuron parameters by utilizing an existing machine learning based hyper-parameters tuning tool. 2) An error function is proposed for the parameter fitting to achieve a good matching on spike timing. And 3) This paper is the first to show that the fitted adaptive neuron could achieve a good performance on a delay task using the Neural Engineering Framework approach, especially at a low firing rate.

2 NEURON MODEL PARAMETERS FITTING

$$\tau_m \frac{du(t)}{dt} = -(u(t) - V_{rest}) + \Delta_T \exp\left(\frac{u(t) - V_T}{\Delta_T}\right) + R(I_{inject}(t) + I_{syn(t)}) - R_{adp} R I_w(t) \quad (1)$$

$$\tau_w \frac{dI_w(t)}{dt} = -a(u(t) - V_{rest}) - I_w(t) + b\tau_w \delta(t - t_0) \quad (2)$$

$$\tau_{syn} \frac{I_{syn}(t)}{dt} = -I_{syn(t)} + \sum_j W_{ji} \delta(t - t_0)$$

(3)
$$u(t)=\{V_{ahp},\text{if } u(t-1)>V_{th}\text{or in refractory period}u(t)\text{otherwise}$$

(4)

To closely capture the biological neural activity, parameters of the neuron model need to be carefully fitted. The adaptive exponential neuron model (AdEx) is described by Eq. 1-Eq. 4. Eq. 1 models how membrane potential changes with time. In this equation, u, V_{rest}, and V_T represent a neuron's membrane potential, resting potential, and depolarization threshold respectively. I_{inject} is the injection current. I_{syn} is the synaptic current that comes from other connected neurons. R and τ_m represents the membrane resistance and membrane time constant respectively. Δ_T is the slope factor, which determines the sharpness of spike initiation. The term $\Delta T\exp(\frac{du(t)-V_T}{\Delta T})$

models an additional current that dependence on the exponential of the voltage. The term $R_{adp}I_w(t)$ models the adaptation behavior. $I_w(t)$ is the adaptation current and R_{adp} is the adaptation resistance. The dynamics of the adaptation current are described in Eq. 2. In this equation, τ_w is the adaptation time constant and a is an adaptation coupling parameter. The term $b\delta(t-t_0)$ means that, while a neuron is firing, the adaptation current is increased by an amount b when each spike happens. Synapse dynamics are represented by Eq. 3, where τ_{syn} is the synapse time constant and W_{ji} is the synapse weight between the modeled neuron i and its neighboring neurons j. The membrane potential resets when it reaches a pre-defined numerical threshold V_{th}. Unlike standard neuron models, the membrane potential resets to a pre-defined value V_{ahp} rather than the resting potential, to better capture the living neuron properties. A refractory period is also implemented: after a neuron fires, it will enter a period where no other spike can happen. These are described by Eq. 4. The simulation results shown in this paper use a sampling rate of 1ms, and the discrete-time implementation is based on the forward Euler method. Eq. 1 and Eq. 2 can be modified and changed to other models. If the $\Delta T\exp(\frac{du(t)-V_T}{\Delta T})$

is removed, then the model will become the Adaptive Leaky-Integrate-and-Fire (ALIF) model. If the entire adaptation term $R_{adp}I_w(t)$ is removed, then the model will become the Leaky-Integrate-and-Fire (LIF) model.

Finding parameter settings that cause this model to produce behavior that matches a particular neuron by hand-tuning is not an easy task. There are some previous attempts to make an efficient automatic tuning tool [6] [24]. In recent years, machine learning plays a more and more important role in solving such problems, including efficient hyperparameter optimization. Since Python and Python-based software libraries such as Tensorflow and Pytorch have become standard in the machine learning community, there is a trend to do SNN learning algorithm studies with these tools. Therefore, it is also a natural step to bring the advanced machine learning approach to the SNN model fitting area.

```
import optuna
numTrial=1000
def optimization(trial):
    set parameter tuning range
    pError = CalcError()
    return pError
study = optuna.create_study(direction='minimize')
study.optimize(optimization, n_trails=numTrial)
```

Figure 2: OPTUNA interface for error optimization [1].

In this work, Optuna [1] is used for parameter fitting, which is an open-source automatic hyperparameter optimization framework based on python. It could efficiently search large spaces and prune unpromising trials for faster results. It is also easy to parallelize. Fig. 2 shows an example Optuna tuning interface. By defining the error function, setting the number of trials, parameter tuning range, and optimization direction, the tool could return the optimized fitting result for each trail. Table 1 shows all the tunable parameters and the range set for Optuna tuning.

Table 1: Tunable parameter settings and fitting results for OPTUNA tuning.

	Tunable parameters setting			**Single-current fitting**		
Num	Name	Range	Step	AdEx	ALIF	LIF
1	$V_{ahp}(mV)$	[-60, -15]	0.5	-23.5	-43.5	-30.5
2	$V_T(mV)$	[-60, -15]	0.5	-23.5	-41	-23.5
3	$V_{th}(mV)$	[10, 40]	0.5	27	18.5	12
4	$t_{ref}(ms)$	[0.5, 20]	0.5	6	8	17.5
5	$R(m\Omega)$	[0.1, 1]	0.1	0.4	0.3	0.6
6	$tau_m(ms)$	[0.5, 100]	0.5	28	62.5	60
7	$R_{adp}(m\Omega)$	[0.1, 1]	0.1	0.8	0.9	/
8	$tau_w(ms)$	[0.5, 100]	0.5	38	57.5	/
9	b	[0.5, 100]	0.5	70	42.5	/
10	$\Delta_T(ms)$	[0.5, 10]	0.5	1.5	/	/
Peak Error				2	2	82

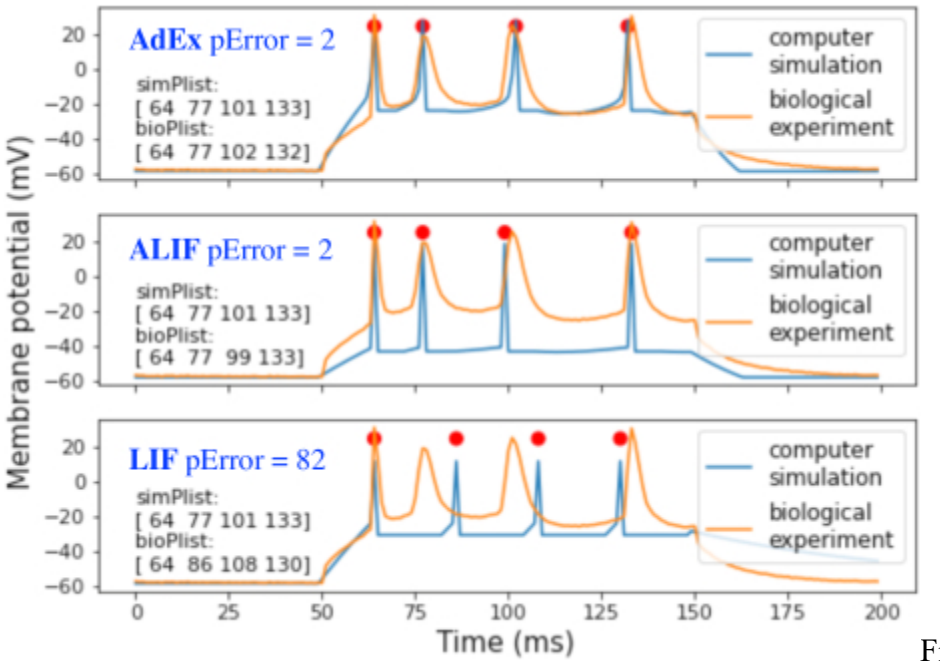


Figure 3: Neuron model fitting results.

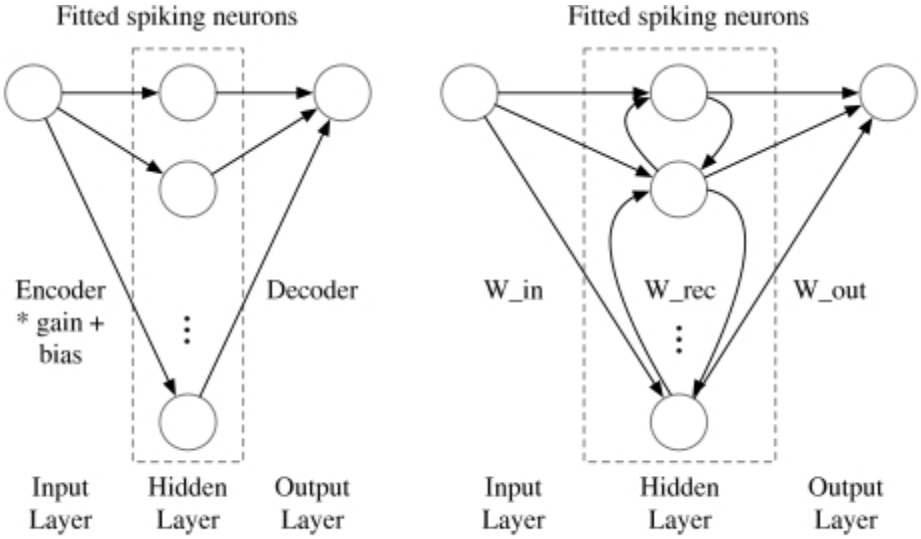
To use the automatic platform for neuron model parameter tuning, defining the error function is critical. Algorithm 1 shows the steps to measure and calculate an error score, *pError*, to quantify the difference between biological experiment and computer simulation results. For the biological experiment, neural cultures were obtained by dissociating cortices of postnatal day 0 Sprague Dawley rats and plating neurons onto poly-D-lysine coated tissue culture dishes. On days in vitro (DIV) 12-19, neuron IV characteristics were obtained by injecting a 250pA current in current-clamp mode. This method is not limited to a specific injection current and this value is chosen because a clear adaptation behavior could be observed under 250pA. Membrane potential of the neuron is recorded for 200ms and assigned to *bioVlist* in Algorithm 1 . The injection current is given between 50-150ms. In the computer simulation, a single neuron model with random parameters within the pre-defined range is initialized, then the same biological experiment is repeated in simulation. The recorded membrane potential of the SNN model is assigned to *simVlist* (line 5-15).

Since the fitting goal in this paper is to match the precise spike timing, the absolute time difference for each pair of the spike peak is used for error measurement (line 24). For biological neurons, the adaptation behavior is more obvious when the injection current is just given to the neuron, and time to the first spike is more important as compared to the timing of the following spikes. Therefore, a weight parameter is added to each spike to give more emphasis on matching the timing of the first two spikes (line 19-22). There are cases when the number of spikes does not match between biological experiment results and computer simulation results. In these cases, a noPairPenalty is added to the error (line 26). The values of Weight and noPairPenalty shown in Algorithm 1 are chosen empirically.

The auto-fitting approach is tested with LIF, ALIF, and AdEx model to compare their effectiveness in capturing the exact spike timing of the biological neurons. All of the fitting

experiments were run three times, each time with Optuna tuning for 1000 trials. The best result among all of the runs and all of the trials are listed in Table 1 and the best-fitted curves are shown in Fig. 3. As shown in the result, the LIF model cannot capture the precise timing (peak error 82) because it lacks the adaptation mechanism. The ALIF and AdEx models can both reduce the peak error to 2, and thus accurately model the timing of the biological system. At a closer look of the fitting result, the exponential term in the AdEx model generates the spike shape, hence the after-hyperpolarization voltage could be matched better as compared to the ALIF model. However, since after-hyperpolarization voltage does not directly influence the temporal learning performance and ALIF is more computationally inexpensive, ALIF becomes a better-suited model in studying the temporal learning property of *in vitro* living neural networks.

3 DELAY TASK WITH FITTED NEURON MODEL



(a) Network to compute decoder **(b) Network to approximate delay function** Figure 4: Delay network topology.

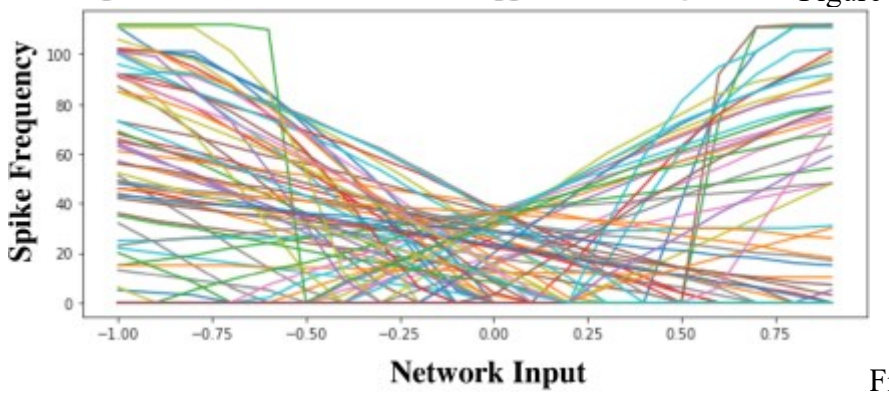


Figure 5: Neuron tuning curve for 100 neurons.

In this section, we use a network of ALIF neurons with fitted parameters to compute a delay task: the output should be the same as the input one second in the past. To construct this network, we use the Neural Engineering Framework (NEF) [7] [30] approach, where we first train a feed-forward neural network that approximates the identity function (i.e. an auto-encoder) and then connect the output back to the input such that the synapse dynamics and the transformed weight matrices will cause the overall system to approximate the differential equation that approximates the ideal delay function. This is known as a Legendre Memory Unit [31] and the learning algorithm is shown in Algorithm 2 .

The training process contains two steps. In the representation phase, a feed-forward network with a single hidden layer of neurons is constructed (Fig. 4 (a)) with randomly generated input weights (encoder × gain) and bias, and no nonlinearities at the input or output. The goal is to compute the network output weights (decoder) so that the output of the network is exactly the same as the input. The input and output dimensionality is n_dim

, so in this step the n_dim dimensional time-varying signal is represented by a population of n_hidden neurons. To maximize the neuron diversity and reach a better representation, the input weights are generated by randomly sampling n_hidden values on the surface of a n_dim hyper-sphere (line 6). The gain and bias are generated following the same rule, which is to have each neuron respond differently to different values of the input signal. This kind of diversity is described by the neuron tuning curve (Fig. 5). In this curve, the x-axis describes the network input values, the y-axis is the spike frequency, and each color represents a neuron. After recording how spike frequency changes with the injection current in the *propertyList* (line 8-13), an *intercept* and *maxFreq* value will be randomly chosen for each neuron from the given range (line 16-17). *minFreq*, which is the minimum spike frequency above zero, will be found by checking the *propertyList* (line 18). Then two points: (intercept, *minFreq*) and (1, *maxFreq*) on the tuning curve for a given neuron are used to form the linear equation described in Algorithm 2 line 19-21. Based on the equation, a unique *gain* and *bias* can be solved for this neuron. To find the output weights, n_hidden number of neurons with fitted parameters are initialed and ridge regression is used to compute the decoder and minimize the mean squared error between the network input and the output (line 23-24). When calculating the network output at each time step, the output of the hidden layer is represented with a vector of n_hidden

, where a spike at this time step is represented as one and no spike is represented as zero for each neuron. With this representation, the exact timing of a spike is taking into account for training and prediction.

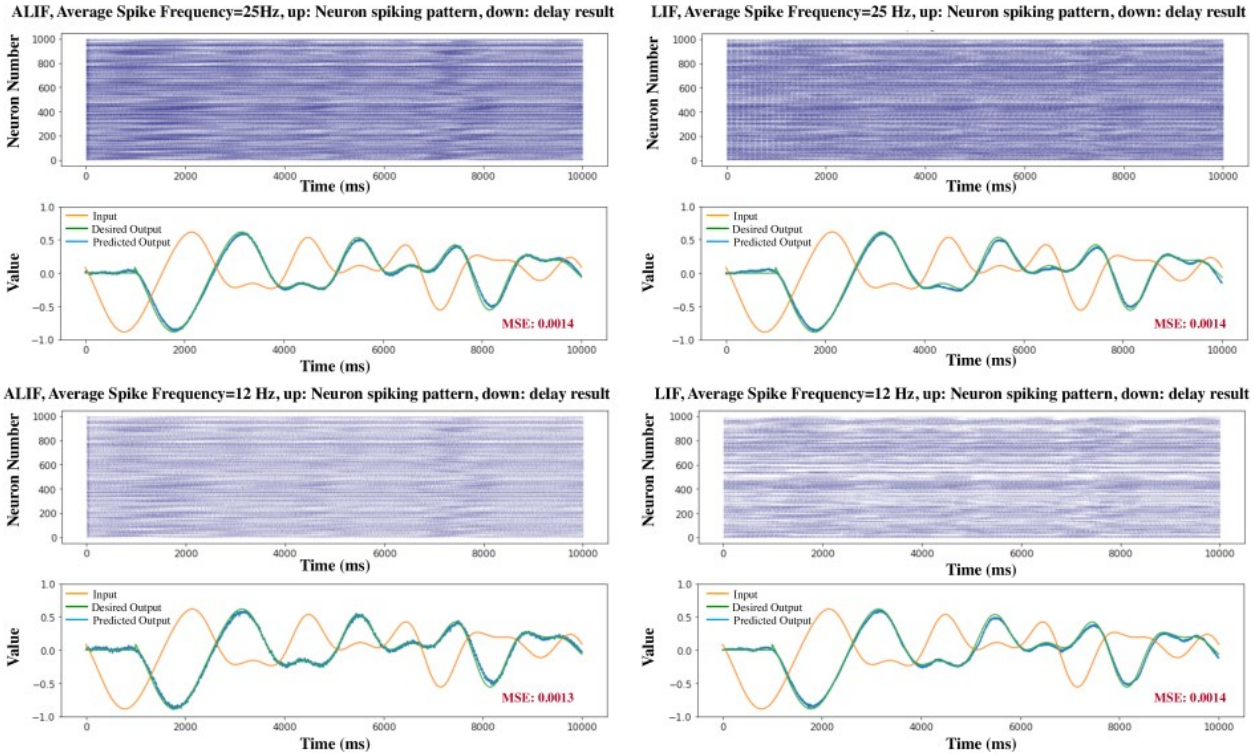


Figure 6: Delay task study with different hidden layer spike frequency.

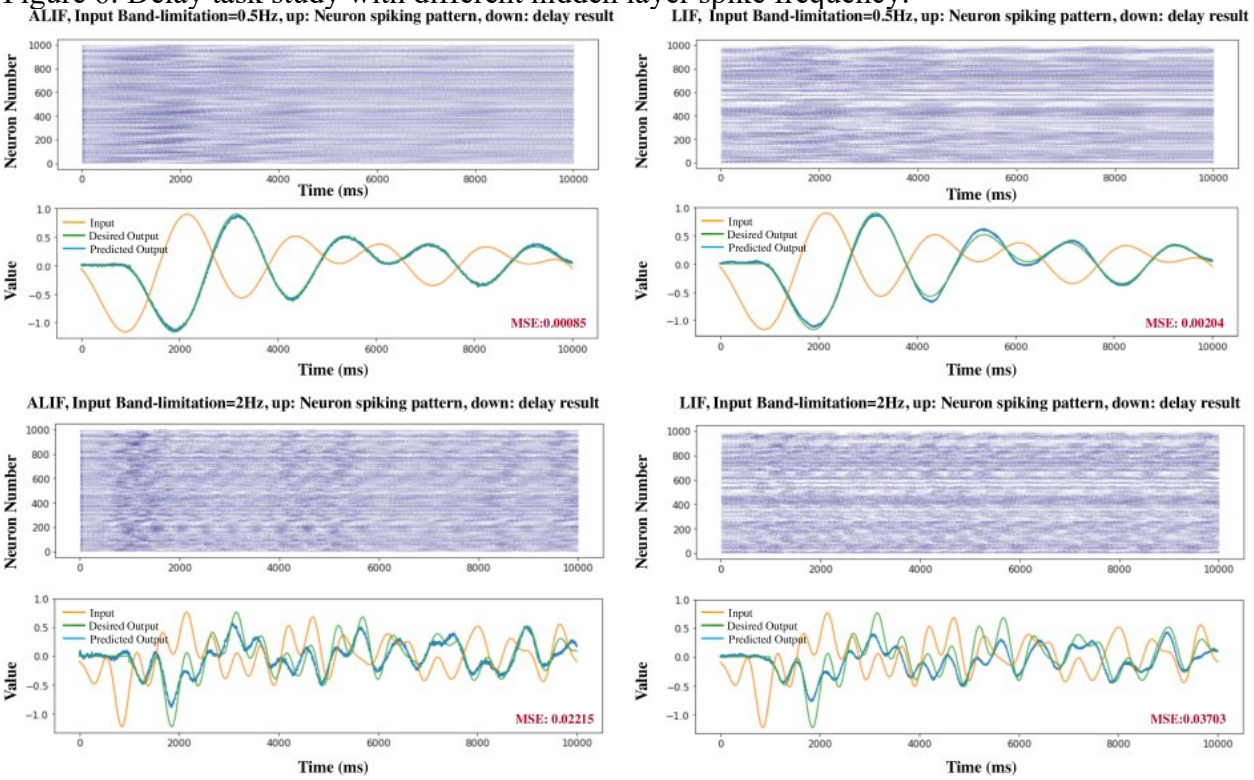


Figure 7: Delay task task with with different input band-limitation.

To turn this feed-forward network into a system capable of computing a differential equation, recurrent connections are added to the network (fig. 4). The input, output, and recurrent weights

of the recurrent network is calculated by equations described in line 26-29, where the A, B, D dynamic matrices are calculated according to the LMU's mathematical derivation with a pre-defined window size. Here dot means dot product multiplication; × means element-wise multiplication. The window size represents the length of the past history this network could record. This specifies the particular differential equation that we want the recurrent neural network to approximate. As has been shown in general [7], the resulting neural network will approximate the desired differential equation $\dot{m}=Am+Bx$

and $y = Dm$. As shown in specific [31], the particular A, B, and D matrices used here will approximate the desired delay function $y(t) = x(t - \theta)$.

After the network is constructed, a randomly sampled continuous-time white noise process is given as the network input and the mean square error of the actual and desired network output is computed to indicate the performance of the constructed delay network (line 31-33). Note that only the hidden layer neurons are fitted spiking neurons, the input and output layers have no nonlinearity.

Results of the delay task for fitted ALIF and LIF neurons with different hidden layer spike frequency are shown in Fig. 6. Different high_freq

and low_freq

values in Algorithm 2 are tested with both models to reach a different hidden layer spike frequency for a given input. Results show that both fitted ALIF and LIF neurons perform well even with a very low average spike frequency (12Hz), which is within the range of the spike frequency of a living neuron. To further explore the difference between using the ALIF and the LIF model for the delay task, different input band-limitations for the white noise process are tested and results are shown in Fig. 7. With an increase in the input frequency, the mean square error for both ALIF and LIF models increased. This is due to the limitation of the Padé approximation [2] approach which is used by the LMU. However, when comparing the ALIF and LIF results, an interesting finding is that, when the input band-limitation increases, the ALIF model tends to have a better mean square error as compared to the LIF model with a similar output spike frequency.

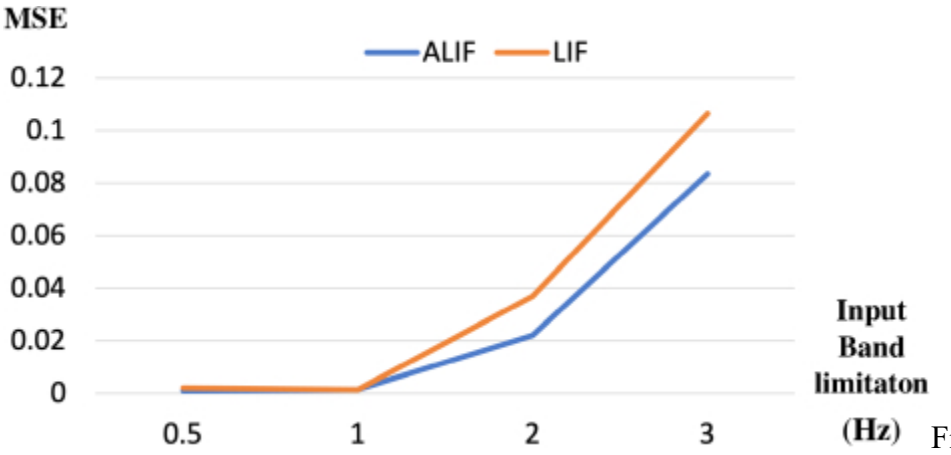


Figure 8: Mean Square Error at Different Input Band-limitation.

4 DISCUSSION

This paper shows a preliminary study on using a fitted neuron model to perform temporal tasks. A new automatic neuron parameter fitting approach is proposed and the delay task study suggests that the fitted neuron, with adaptation, can achieve good performance. Figure 8 shows that the fitted adaptive neuron achieves a lower means square error as compared to the neuron without adaption when input band-limitation increases. This result shows that the selection of the neuron model has a big impact on learning performance especially when the parameters are fitted to match with a realistic neuron. Neuron parameters typically can be tuned to achieve better learning performance in algorithm designs that are inspired by biological neural networks. This work aims to develop a tool chain to study algorithm designs for living neuron networks. Hence, tuning parameters would lost model fidelity. The code of this work can be found at:

https://github.com/yuanzenggit/temporal_learning_with_fittedSNN

REFERENCES

- Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. In *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

- George A Baker, George A Baker Jr, George Baker, Peter Graves-Morris, and Susan S Baker. 1996. *Pade Approximants: Encyclopedia of Mathematics and It's Applications, Vol. 59 George A. Baker, Jr., Peter Graves-Morris*. Vol. 59. Cambridge University Press.
- Guillaume Bellec, Darjan Salaj, Anand Subramoney, Robert Legenstein, and Wolfgang Maass. 2018. Long short-term memory and learning-to-learn in networks of spiking neurons. *arXiv preprint arXiv:1803.09574*(2018).
- Edward S Boyden, Feng Zhang, Ernst Bamberg, Georg Nagel, and Karl Deisseroth. 2005. Millisecond-timescale, genetically targeted optical control of neural activity. *Nature neuroscience* 8, 9 (2005), 1263.

- Yongqiang Cao, Yang Chen, and Deepak Khosla. 2015. Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision* 113, 1 (2015), 54–66.
- Kristofor David Carlson, Jayram M Nageswaran, Nikil Dutt, and Jeffrey L Krichmar. 2014. An efficient automated parameter tuning framework for spiking neural networks. *Frontiers in neuroscience* 8 (2014), 10.
- Chris Eliasmith and Charles H. Anderson. 2003. *Neural engineering: Computation, representation, and dynamics in neurobiological systems*. MIT Press, Cambridge, MA.
- Chris Eliasmith, Terrence C Stewart, Xuan Choo, Trevor Bekolay, Travis DeWolf, Yichuan Tang, and Daniel Rasmussen. 2012. A large-scale model of the functioning brain. *science* 338, 6111 (2012), 1202–1205.
- Edris Zaman Farsa, Arash Ahmadi, Mohammad Ali Maleki, Morteza Gholami, and Hima Nikafshan Rad. 2019. A Low-Cost High-Speed Neuromorphic Hardware Based on Spiking Neural Network. *IEEE Transactions on Circuits and Systems II: Express Briefs* (2019).
- Nicolas Fourcaud-Trocmé, David Hansel, Carl Van Vreeswijk, and Nicolas Brunel. 2003. How spike generation mechanisms determine the neuronal response to fluctuating inputs. *Journal of neuroscience* 23, 37 (2003), 11628–11640.
- Alan L Hodgkin and Andrew F Huxley. 1952. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology* 117, 4 (1952), 500–544.
- Guosong Hong, Shuo Diao, Alexander L Antaris, and Hongjie Dai. 2015. Carbon nanomaterials for biological imaging and nanomedicinal therapy. *Chemical reviews* 115, 19 (2015), 10816–10906.
- Nikola K Kasabov. 2014. NeuCube: A spiking neural network architecture for mapping, learning and understanding of spatio-temporal brain data. *Neural Networks* 52(2014), 62–76.
- Seijoon Kim, Seongsik Park, Byunggook Na, and Sungroh Yoon. 2019. Spiking-yolo: Spiking neural network for real-time object detection. *arXiv preprint arXiv:1903.06530* 1 (2019).
- Wolfgang Maass. 1997. Networks of spiking neurons: the third generation of neural network models. *Neural networks* 10, 9 (1997), 1659–1671.
- Tuomo Mäki-Marttunen, Geir Halnes, Anna Devor, Christoph Metzner, Anders M Dale, Ole A Andreassen, and Gaute T Einevoll. 2018. A stepwise neuron model fitting procedure designed for recordings with high spatial resolution: application to layer 5 pyramidal cells. *Journal of neuroscience methods* 293 (2018), 264–283.
- Skander Mensi, Richard Naud, Christian Pozzorini, Michael Avermann, Carl CH Petersen, and Wulfram Gerstner. 2012. Parameter extraction and classification of three cortical neuron types reveals two distinct adaptation mechanisms. *Journal of neurophysiology* 107, 6 (2012), 1756–1775.
- Ştefan Mihalaş and Ernst Niebur. 2009. A generalized linear integrate-and-fire neural model produces diverse spiking behaviors. *Neural computation* 21, 3 (2009), 704–718.
- Elon Musk *et al.* 2019. An integrated brain-machine interface platform with thousands of channels. *Journal of medical Internet research* 21, 10 (2019), e16194.
- Alexander Neckar, Sam Fok, Ben V. Benjamin, Terrence C. Stewart, Nick N. Oza, Aaron R. Voelker, Chris Eliasmith, Rajit Manohar, and Kwabena Boahen. 2019. Braindrop: A Mixed-Signal Neuromorphic Architecture With a Dynamical Systems-Based Programming Model. Proc. IEEE 107(2019), 144–164. Issue 1. https://doi.org/10.1109/JPROC.2018.2881432

- Cuong Nguyen, Hansini Upadhyay, Michael Murphy, Gabriel Borja, Emily J Rozsahegyi, Adam Barnett, Ted Brookings, Owen B McManus, and Christopher A Werley. 2019. Simultaneous voltage and calcium imaging and optogenetic stimulation with high sensitivity and a wide field of view. *Biomedical optics express* 10, 2 (2019), 789–806.
- Paul F Pinsky and John Rinzel. 1994. Intrinsic and network rhythmogenesis in a reduced Traub model for CA3 neurons. *Journal of computational neuroscience* 1, 1-2 (1994), 39–60.
- Christian Pozzorini, Skander Mensi, Olivier Hagens, Richard Naud, Christof Koch, and Wulfram Gerstner. 2015. Automated high-throughput characterization of single neurons by means of simplified spiking models. *PLoS Comput Biol* 11, 6 (2015), e1004275.
- Cyrille Rossant, Dan FM Goodman, Jonathan Platkiewicz, and Romain Brette. 2010. Automatic fitting of spiking neuron models to electrophysiological recordings. *Frontiers in neuroinformatics* 4 (2010), 2.
- Darjan Salaj, Anand Subramoney, Ceca Kraisnikovic, Guillaume Bellec, Robert Legenstein, and Wolfgang Maass. 2020. Spike-frequency adaptation provides a long short-term memory to networks of spiking neurons. *bioRxiv* (2020).
- David Sterratt, Bruce Graham, Andrew Gillies, and David Willshaw. 2011. *Principles of computational modelling in neuroscience, section 8.1.2*. Cambridge University Press.
- Christoph Stöckl and Wolfgang Maass. 2020. Classifying images with few spikes per neuron. *arXiv preprint arXiv:2002.00860*(2020).
- CA Thomas Jr, PA Springer, GE Loeb, Y Berwald-Netter, and LM Okun. 1972. A miniature microelectrode array to monitor the bioelectric activity of cultured cells. *Experimental cell research* 74, 1 (1972), 61–66.
- Roger D Traub, Robert K Wong, Richard Miles, and Hillary Michelson. 1991. A model of a CA3 hippocampal pyramidal neuron incorporating voltage-clamp data on intrinsic conductances. *Journal of neurophysiology* 66, 2 (1991), 635–650.
- Aaron R Voelker and Chris Eliasmith. 2018. Improving spiking dynamical networks: Accurate delays, higher-order synapses, and time cells. *Neural computation* 30, 3 (2018), 569–609.
- Aaron R Voelker, Ivana Kajić, and Chris Eliasmith. 2019. Legendre memory units: Continuous-time representation in recurrent neural networks. (2019).
- Ke Wang, Harvey A Fishman, Hongjie Dai, and James S Harris. 2006. Neural stimulation with a carbon nanotube microelectrode array. *Nano letters* 6, 9 (2006), 2043–2048.
- Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, Yuan Xie, and Luping Shi. 2019. Direct training for spiking neural networks: Faster, larger, better. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 1311–1318.
- Yuan Zeng, Kevin Devincentis, Yao Xiao, Zubayer Ibne Ferdous, Xiaochen Guo, Zhiyuan Yan, and Yevgeny Berdichevsky. 2018. A supervised STDP-based training algorithm for living neural networks. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 1154–1158.
- Yuan Zeng, Zubayer Ibne Ferdous, Weixiang Zhang, Mufan Xu, Anlan Yu, Drew Patel, Xiaochen Guo, Yevgeny Berdichevsky, and Zhiyuan Yan. 2019. Inference with Hybrid Bio-hardware Neural Networks. *arXiv preprint arXiv:1905.11594*(2019).
- Wenrui Zhang and Peng Li. 2019. Spike-train level backpropagation for training deep recurrent spiking neural networks. *arXiv preprint arXiv:1908.06378*(2019).

ICONS 2021, July 27–29, 2021, Knoxville, TN, USA

ACM ISBN 978-1-4503-8691-3/21/07.
DOI: https://doi.org/10.1145/3477145.3477153