Temporal Coding in Spiking Neural Networks With Alpha Synaptic Function: Learning With Backpropagation

Iulia-Maria Comşa, Krzysztof Potempa, Luca Versari, Thomas Fischbacher, Andrea Gesmundo, and Jyrki Alakuijala

***Abstract*— The timing of individual neuronal spikes is essential for biological brains to make fast responses to sensory stimuli. However, conventional artificial neural networks lack the intrinsic temporal coding ability present in biological networks. We propose a spiking neural network model that encodes information in the relative timing of individual spikes. In classification tasks, the output of the network is indicated by the first neuron to spike in the output layer. This temporal coding scheme allows the supervised training of the network with backpropagation, using locally exact derivatives of the postsynaptic spike times with respect to presynaptic spike times. The network operates using a biologically plausible synaptic transfer function. In addition, we use trainable pulses that provide bias, add flexibility during training, and exploit the decayed part of the synaptic function. We show that such networks can be successfully trained on multiple data sets encoded in time, including MNIST. Our model outperforms comparable spiking models on MNIST and achieves similar quality to fully connected conventional networks with the same architecture. The spiking network spontaneously discovers two operating modes, mirroring the accuracy-speed tradeoff observed in human decision-making: a highly accurate but slow regime, and a fast but slightly lower accuracy regime. These results demonstrate the computational power of spiking networks with biological characteristics that encode information in the timing of individual neurons. By studying temporal coding in spiking networks, we aim to create building blocks toward energy-efficient, state-based biologically inspired neural architectures. We provide open-source code for the model.**

***Index Terms*— Backpropagation, biological neural networks, feedforward neural networks, image classification, supervised learning.**

I. Introduction

INSPIRED by the biology of the nervous system, artificial neural networks have recently been used to achieve resounding success in solving many real-world problems, sometimes surpassing the performance of humans [1]–[3]. However, conventional artificial networks lack the intrinsic ability to encode information using temporal coding schemes in the same way as biological brains do. In the nervous system, the timing of individual neuronal spikes is fundamental during behaviors requiring rapid encoding and processing of perceptual stimuli. The relative firing time of neurons has not only been shown to encode stimulus information in the visual [4], auditory [5], and tactile [6] sensory systems but also in higher-level neural structures, such as the thalamus [7] and the hippocampus [8]. Moreover, by observing *in vivo* the low latency of responses to visual stimuli in the temporal cortex, it can be concluded that the response is produced by individual spikes occurring at every synaptic stage across the visual areas of the brain [9].

Architectures for atemporal networks that emulate the processing of information in a temporal fashion using memory mechanisms have been proposed [10]. However, these do not have the advantages conferred by encoding information in the temporal domain. One disadvantage is that every neuron needs to wait for the activation of all the neurons in the previous layer in order to produce an answer. Compared with biological brains, this is inefficient in terms of energy consumption. Moreover, information in the real world almost always has a temporal dimension. Therefore, additional processing is needed for encoding it in an atemporal network, potentially losing information in the process.

Unlike the field of conventional artificial neural networks, the field of artificial spiking networks that encode information in time has not been thoroughly explored. So far, spiking networks have only achieved modest results. The main difficulty in advancing the field of spiking networks has been their training process, as the techniques usually used for supervised learning in atemporal networks cannot be directly applied when information is encoded in a temporal sequence of asynchronous events. Atemporal networks owe much of their success to the development of the backpropagation algorithm [11], [12]. Backpropagation exploits the existence of an end-to-end differentiable relationship between a loss function and the network inputs and outputs, which can be expressed in terms of local derivatives at all hidden network parameters. One can, thus, find local updates that minimize the loss function and apply them as incremental updates to train the network. On the other hand, in spiking networks, which encode information in sequences of binary spike events, differentiable relationships do not naturally exist.

The problem of training in spiking networks has been addressed in several ways. Many spiking models have adopted a rate coding scheme. In contrast to the temporal coding of

Manuscript received 24 March 2020; revised 3 December 2020 and 30 March 2021; accepted 31 March 2021. Date of publication 26 April 2021; date of current version 6 October 2022. *(Corresponding author: Iulia-Maria Comşa.)*

Iulia-Maria Comşa, Luca Versari, Thomas Fischbacher, Andrea Gesmundo, and Jyrki Alakuijala are with Google Research Zürich, 8002 Zürich, Switzerland (e-mail: iuliacomsa@google.com; veluca@google.com; tfish@google.com; agesmundo@google.com; jyrki@google.com).

Krzysztof Potempa was with Google Research Zürich, 8002 Zürich, Switzerland. He is now with GSA Capital, London, U.K. (e-mail: krzysztof.potempa@gmail.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2021.3071976.

Digital Object Identifier 10.1109/TNNLS.2021.3071976

information, which is based on individual spike timing, rate coding averages over multiple spikes. Approximate gradients have been proposed in order to allow gradient descent optimization in such networks [13]–[15]. Various other learning rules have been proposed with the objective of producing custom spiking patterns or spike distributions [16]–[25]. Alternatively, atemporal deep neural networks can be trained and then converted to spiking networks using rate-coding schemes [26]–[29]. Spiking networks can also be trained using methods, such as evolutionary algorithms [30] or reinforcement learning [31]. However, such rate-coding schemes may still be redundant considering the evidence that biological systems can react to stimuli on the basis of single spikes [4], [6], [9].

In contrast to rate-coding models, here, we are interested in the temporal encoding of information into single spikes. Crucially, this change in the coding scheme shifts the differentiable relationships into the temporal domain. To find a backpropagation scheme for temporal coding, we need a differentiable relationship of the time of a postsynaptic spike with respect to the weights and times of the presynaptic spikes. Encoding information in the temporal domain makes this possible.

A similar idea was proposed in the SpikeProp model [32]. In this model, a spiking network successfully learns the supervised XOR problem encoded in the temporal domain using neurons that generate single spikes. The model was able to implement backpropagation by approximating the differentiable relationship for small learning rates. It was also necessary to explicitly encode inhibitory and excitatory neurons in order for training to converge. Various extensions to SpikeProp have been proposed [33]–[38]. Recently, Mostafa [39] trained a spiking network to solve temporally encoded XOR and MNIST problems by deriving locally exact gradients with nonleaky spiking neurons. Other training approaches include spike-timing-dependent plasticity [40] and reinforcement learning [41].

One important choice when modeling a spiking neural network is that of the synaptic transfer function, which defines the dynamics of the neuron membrane potential in response to a presynaptic spike. From a machine learning perspective, this is equivalent to the activation function in conventional networks, but, importantly, it operates in time. Historically, the Hodgkin–Huxley model [42] was the first to offer a detailed description of the process of neuronal spiking using differential equations describing the dynamics of sodium and potassium ionic channels. In practice, however, this model is often needlessly complex. Instead, many applications use the reduced spike response model (SRM) [43]. In this model, the membrane potential is described by the integration of kernels reflecting the incoming currents at the synapse. The neuron spikes when its membrane potential reaches a threshold and then enters a refractory period. Provided that an appropriate kernel function is used [44], the SRM can approximate the dynamics of the Hodgkin–Huxley model to a satisfactory extent, while demanding less computational power and being easier to analyze. Integrate-and-fire neurons and their leaky counterparts are examples of SRM. A commonly used SRM impulse response function is the exponential decay, e^{-t}.

A more biologically realistic but less explored alternative is the alpha function, of the form te^{-t}, produced by the integration of exponentially decaying kernels. In contrast with the single exponential function, the alpha function gradually rises before slowly decaying [Fig. 1(a)], which allows more intricate interactions between presynaptic inputs. The alpha function has been proposed [45] as a close match to the shape of postsynaptic potentials measured *in vitro* [46], [47]. It, hence, provides a biologically plausible model for exploring the problem-solving abilities of spiking networks with temporal coding schemes. As we show in the following, it is possible to derive exact gradients with respect to spike times using this model.

With these considerations in mind, here, we propose a spiking network model that uses the alpha function for synaptic transfer and encodes information in relative spike times. This work is an extension of preliminary work presented elsewhere [48], with open-source code available online [49]. The spiking network that we describe is fully trained in the temporal domain using exact gradients over domains where relative spiking order is preserved. To the best of our knowledge, this model has not been previously used for supervised learning with backpropagation. We explore the capacity of this model to learn standard benchmark problems, such as Boolean logic gates and MNIST, encoded in individual spike times. To facilitate transformations of the class boundaries, we use sets of *pulses*, which are neurons that send spikes at input-independent, learned times. The model is easily able to solve temporally encoded Boolean logic and other benchmark problems. We perform a search for the best set of hyperparameters for this model using evolutionary-neural hybrid agents [50] in order to solve temporally encoded MNIST. The result improves the state-of-the-art accuracy on nonconvolutional spiking networks and is comparable to the performance of atemporal nonconvolutional networks. Furthermore, we analyze the behavior of the spiking network during training and show that it spontaneously displays two operational regimes that reflect a tradeoff between speed and accuracy: a slow regime that is slow but very accurate, and a fast regime that is slightly less accurate but makes decisions much faster.

With this work, we also aim to increase the familiarity of the research community with the concept of temporal coding in spiking neural networks. Given that biological brains have evolved for millions of years to use temporal coding mechanisms in order to process information efficiently, we expect that an equivalent development will also be a key step in advancing artificial intelligence in the future. The present work is an early building block in this direction that invites further exploration of more complex recurrent architectures, spike-based state machines, and interfacing between artificial and biological spiking neural networks.

II. Methods

A. Temporal Coding

In this model, information is encoded in the relative timing of individual spikes. Similar encoding schemes have been previously used [27], [32], [39], [40]. The input features are encoded in the temporal domain as the spike times of

individual input neurons, with each neuron corresponding to a distinct feature. More salient information about a feature is encoded as an earlier spike in the corresponding neuron. Information propagates through the network in a temporal fashion. Each hidden and output neuron spikes when its membrane potential rises above a fixed threshold. Similar to the input layer, the output layer of the network encodes a result in the relative timing of output spikes. In other words, the computational process consists of producing a temporal sequence of spikes across the network in a particular order, with the result encoded in the ordering of spikes in the output layer.

We use this model to solve standard classification problems. Given a classification problem with m inputs and n possible classes, the inputs are encoded as the spike times of individual neurons in the input layer and the result is encoded as the index of the neuron that spikes first among the neurons in the output layer. An example drawn from class k is classified correctly if and only if the kth output neuron is the first to spike. An earlier output spike can reflect more confidence in the network in classifying a particular example, as it implies more synaptic efficiency or a smaller number of presynaptic spikes. In a biological setting, the winning neuron could suppress the activity of neighboring neurons through lateral inhibition [51], while in a machine learning setting the spike times of the nonwinning neurons can be useful in indicating alternatives predictions of the network. The learning process aims to change the synaptic weights, and thus, the spike timings in such a way that the target order of spikes is produced.

In the models, we use for the current experiments, we allow each neuron to spike once per input example, but it is possible to use the same model with multiple spikes per neuron and custom architectures, as described in Section II-F in the following.

We note that the function describing the membrane potential over time in response to inputs, until a spike is produced, is continuous. Therefore, although spikes are single events occurring at specific points in time, there is no discontinuity from a temporal perspective, since the computation is represented by their *timing*. As we will see in the following, it is possible to obtain exact derivatives of spike times with respect to time and weights, and the only discontinuity appears in the optimization process in the loss function, at points where spike order is reversed.

B. Alpha Activation Function

We use an alpha biologically inspired function to model the neuronal membrane dynamics [44], [45]. The potential of the membrane in response to a single-incoming spike is of the form $u(t) = te^{-\tau t}$, where τ is the decay constant. The shape of this function is similar to that of the difference between two exponential decay functions, which we verified also achieves similar results when used in the model presented here. This is a possible instantiation of the SRM [43, Sec. 4.2.1]. The alpha function we propose has a gradual rise and a slow decay, peaking at $t_{\max} = \tau^{-1}$. Every synaptic connection has a weight, representing its efficiency. The decay rate has the effect of scaling the induced potential in amplitude and time, while the weight of the synapse has the effect of scaling the amplitude only [Fig. 1(a)].

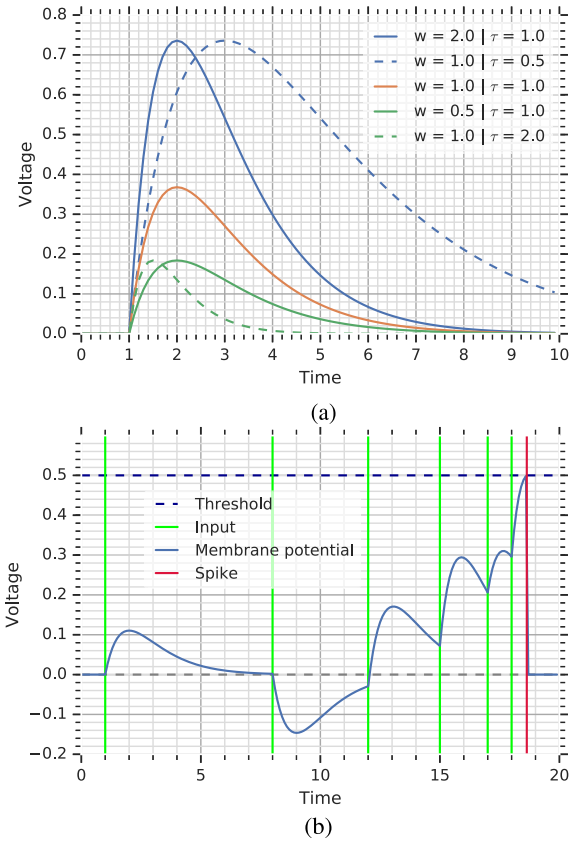

Fig. 1. Description of the neuron model with alpha synaptic function. (a) Alpha potential function with different sets of weights w and decay constants τ. The weight scales the function in amplitude, whereas the decay constant scales it in both amplitude and time. (b) Example of potential membrane dynamics in response to excitatory and inhibitory inputs, followed by a spike. In this example, $\tau = 1$, $w = \{0.3, -0.4, 0.5, 0.7, 0.5, 0.8\}$, $t = \{1, 8, 12, 15, 17, 18\}$ and the spike occurs at $t_{\text{out}} = 18.64$.

The following procedure for computing the spike time of a neuron is applied for all neurons in the model. In a fully connected model, the inputs to a neuron in layer h_j consist of all of the neurons in layer h_{j-1}. In the first layer of the network, the spike times are given by the temporal encoding scheme described above, while the spike times in the following layers are computed according to the procedure in the following.

Given a set I of presynaptic inputs i arriving at times $t_i \leq t$ with weights w_i and assuming the postsynaptic neuron has not yet spiked, its membrane potential at time t is given by

$$V_{\text{mem}}(t) = \sum_{i \in I} w_i (t - t_i) e^{\tau (t_i - t)}. \tag{1}$$

The neuron spikes when the membrane potential crosses the firing threshold [Fig. 1(b)]. To compute the spike time t_{out} of a neuron, we determine the minimal subset of all presynaptic inputs $I_{t_{\text{out}}}$ with $t_i \leq t_{\text{out}}$, which cause the membrane potential

to reach the threshold θ while rising

$$\sum_{i \in I_{t_{\text{out}}}} w_i (t_{\text{out}} - t_i) e^{\tau(t_i - t_{\text{out}})} = \theta. \tag{2}$$

To find the spike time of a neuron (if any), the inputs need to be temporally sorted; in an event-based model, the inputs will be naturally sorted. For every newly arriving input t_i, we add it to $I_{t_{\text{out}}}$ and solve (2). If there is no solution, then the current set of inputs does not cause a spike. If there is a predicted spike, it is still possible that an input t_{i+1} might come between t_i and the predicted t_{out}, so we repeat the process until t_{i+1} is larger than the predicted t_{out} or there are no more inputs. If the membrane voltage never crosses θ, then the neuron does not spike; otherwise, it spikes at the last predicted t_{out}.

Equation (2) has two potential solutions—one on the rising part of the function and one on the decaying part. If a solution exists (in other words, if the neuron spikes), then its spike time is the earlier of the two solutions.

We solve (2) as follows. For a set of inputs I, for which we are currently solving this equation to check when or whether the neuron spikes, we denote $A_I = \sum_{i \in I} w_i e^{\tau t_i}$ and $B_I = \sum_{i \in I} w_i e^{\tau t_i} t_i$. Note that a spike can occur if and only if A_I is strictly positive. We then rewrite (2) as

$$\begin{aligned}
& t_{\text{out}} e^{-\tau t_{\text{out}}} \sum_i w_i e^{\tau t_i} - e^{-\tau t_{\text{out}}} \sum_i w_i t_i e^{\tau t_i} = \theta \\
\Leftrightarrow \quad & t_{\text{out}} e^{-\tau t_{\text{out}}} A_I - e^{-\tau t_{\text{out}}} B_I = \theta \\
\Leftrightarrow \quad & e^{-\tau t_{\text{out}}} \left(\tau \frac{B_I}{A_I} - \tau t_{\text{out}} \right) = -\tau \frac{\theta}{A_I} \\
\Leftrightarrow \quad & e^{\tau \frac{B_I}{A_I} - \tau t_{\text{out}}} \left(\tau \frac{B_I}{A_I} - \tau t_{\text{out}} \right) = -\tau \frac{\theta}{A_I} e^{\tau \frac{B_I}{A_I}}.
\end{aligned}$$

This formulation allows us to use the Lambert W function [52], [53], using the following property: $e^w w = z \Leftrightarrow w = W_k(z)$. This gives us the spike time t_{out}

$$\begin{aligned}
\tau \frac{B_I}{A_I} - \tau t_{\text{out}} &= W_k \left(-\tau \frac{\theta}{A_I} e^{\tau \frac{B_I}{A_I}} \right) \\
\Leftrightarrow \quad t_{\text{out}} &= \frac{B_I}{A_I} - \frac{1}{\tau} W_k \left(-\tau \frac{\theta}{A_I} e^{\tau \frac{B_I}{A_I}} \right).
\end{aligned} \tag{3}$$

A spike will occur whenever the Lambert W function has a valid argument and the resulting t_{out} is larger than all input spikes. The Lambert W function has two real-valued branches, corresponding to the points where the membrane threshold is crossed from below and from above; as we are interested in the earlier solution of this equation, we employ the main branch ($k = 0$). Note that the Lambert W function is real-valued when its argument is larger than or equal to $-e^{-1}$. It can be proven that this is always the case when (2) has a solution, by expanding $V_{\text{mem}}(t_{\max}) \geq \theta$, where $t_{\max} = (B/A) + (1/\tau)$ is the peak of the membrane potential function corresponding to the presynaptic set of inputs I.

In the Appendix, we prove that the proposed model is powerful enough to represent any sufficiently well-behaved function. Here, we provide an intuitive explanation of the proof. We rely on the fact that two inputs, t_0 and t_1, with opposite weights, w and $-w$, respectively, coming at precisely the same time (i.e., $t_0 = t_1$), will always cancel each other out. If one of these inputs is shifted in time even by a very small amount, the sign of the membrane potential that received these two inputs will always be given by the sign of the weight of the input that came later. Consider then two additional inputs, t_2 and t_3, arriving very close to a fixed time T: the first one has a (large) positive weight and would determine a spike at time T if the neuron potential were initially at 0, while another one has a (large) negative weight and would come at time T, i.e., exactly at the time of the spike caused by t_3 if the neuron potential were initially at 0. If the neuron potential were even slightly below 0, i.e., if the input with negative weight (t_1) came at any time after the input with positive weight (t_0), then there would be no spike. Using this observation, we can construct conditions of the form $t_1 < t_2$, which allows us to split the function input space into many small hyperboxes and to approximate the function value in each box by adjusting T.

C. Error Backpropagation

The spiking network learns to solve problems whose inputs and solutions are encoded in the times of individual input and output spikes. Therefore, the goal is to adjust the output spike times so that their relative order is correct. This is achieved by adjusting weights across the network so that the timing of spikes in the hidden and output layers is adjusted in the right direction (earlier or later).

Given a classification problem with n classes, the neuron corresponding to the correct label should be the earliest to spike. Therefore, we choose a loss function that minimizes the spike time of the target neuron and maximizes the spike time of the other output neurons. Note that this is the opposite of the usual classification setting involving probabilities, where the value corresponding to the correct class is maximized and those corresponding to incorrect classes are minimized. To achieve this, we use the softmax function on the *negative* values of the spike times o_i (which are always positive) in the output layer: $p_j = e^{-o_j} / \sum_{i=1}^{n} e^{-o_i}$.

We then employ the cross-entropy loss in the usual form: $L(y_i, p_i) = -\sum_{i=1}^{n} y_i \ln p_i$, where y_i is an element of the one-hot encoded target vector of output spike times. Taking the negative values of the spike times ensures that minimizing the cross-entropy loss minimizes the spike time of the correct label and maximizes the rest.

We minimise the cross-entropy loss by changing the value of the weights across the network. This has the effect of delaying or advancing spike times across the whole network. For any presynaptic spike arriving at time $t_j \in I$ with weight w_j, we denote $W_I = W\left(-\tau \frac{\theta}{A_I} e^{\tau \frac{B_I}{A_I}}\right)$ and compute the exact derivative of the postsynaptic spike time with respect to any presynaptic spike time t_j and its weight w_j as:

$$\frac{\partial t_{out}}{\partial t_j} = \frac{w_j e^{\tau t_j} \left(\tau \left(t_j - \frac{B_i}{A_I} \right) + W_I + 1 \right)}{A_I (1 + W_I)} \tag{4}$$

$$\frac{\partial t_{out}}{\partial w_j} = \frac{e^{\tau t_j} \left(t_j - \frac{B_I}{A_I} + \frac{W_I}{\tau} \right)}{A_I (1 + W_I)} \tag{5}$$

These derivatives are applicable to all neurons in the network. In a fully connected model, the inputs of any neuron in layer h_j are the spike times of the neurons in layer h_{j-1}. The backpropagation procedure performed here is the same procedure used in conventional neural networks, with the difference that the adjusted quantity represents a spike time as opposed to an activation value.

As the postsynaptic spike time moves earlier or later in time when $I_{t_{\text{out}}}$ changes to include or exclude presynaptic spikes, the landscape of the loss function also changes. Furthermore, the loss function exhibits discontinuities where an output neuron stops spiking; we counter this problem using a penalty, as described below. In practice, we find that optimization is possible in spite of these challenges.

D. Pulses

In order to adjust the class boundaries in the temporal domain, a temporal form of bias is also needed to adjust spike times, i.e., to delay or advance them in time. In this model, we introduce "pulses" acting as additional inputs across every layer or the network, in order to provide temporal bias across the network. These can be thought of as similar to internally generated rhythmic activity in biological networks, such as alpha waves in the visual cortex [54] or theta and gamma waves in the hippocampus [55].

A set of pulses can be connected to all neurons in the network, to neurons within individual layers, or to individual neurons. A per-neuron bias is biologically implausible and more computationally demanding; hence, in this model, we use either a single set of pulses per network, to solve easier problems, or a set of pulses per layer, to solve more difficult problems. All pulses are fully connected to either all noninput neurons in the network or to all neurons of the noninput layer they are assigned to.

Each pulse spikes at a predefined and trainable time. This has the effect of providing a reference spike delay. Each set of pulses is initialized to spike at times evenly distributed in the interval $(0, 1)$. Subsequently, the spike time of each pulse is learned using (4), while the weights between pulses and neurons are trained using (5), in the same way as all other weights in the network. Hence, the pulses are updated at the same time and under the same backpropagation scheme as the rest of the network, with the aim of minimizing the chosen loss function.

The network can also operate without the pulses. However, in practice, as indicated in Section III, we find that the pulses are essential for synchronizing spike times across the network and with respect to target times and for forcing some of the neurons to spike during training and keep the flow of gradients through the network.

E. Hyperparameters

We train fully connected feedforward networks with topology `n_hidden` (a vector of hidden layer sizes). We use Adam optimization [56] with mini-batches of size `batch_size` to minimize the cross-entropy loss, as this optimizer performed better than stochastic gradient descent. Updates are applied only when the classification label is wrong. We use different learning rates for the pulse spike time (`learning_rate_pulses`) and the weights of both pulse and nonpulse neurons (`learning_rate`). We use a fixed firing threshold (`fire_threshold`) and decay constant (`decay_constant`).

Network weight initialization is crucial for the subsequent training of the network. In a spiking network, it is important that the initial weights are large enough to cause at least some of the neurons to spike; in absence of spike events, there will be no gradient to use for learning. We, therefore, use a modified form of Glorot initialization [57], where the weights are drawn from a normal distribution with standard deviation $\sigma = (2.0/(\text{fan}_{\text{in}} + \text{fan}_{\text{out}}))^{1/2}$ (as in the original scheme) and custom mean $\mu = \texttt{multiplier} \times \sigma$. If the multiplication factor of the mean is 0, this is the same as the original Glorot initialization scheme. We set different multiplication factors for pulse (`pulse_init_multiplier`) and nonpulse (`nonpulse_init_multiplier`) weights. This allows the two types of neurons to prespecialize into inhibitory and excitatory roles. In biological brains, internal oscillations are thought to be generated through inhibitory activities that regulate the excitatory effects of incoming stimuli [58], [59].

Despite careful initialization, the network might still become quiescent during training. We prevent this problem by adding a fixed small penalty (`penalty_no_spike`) to the derivative of all presynaptic weights of a neuron that has not fired. In practice, after the training phase, some of the neurons will spike too late to matter in the classification, and thus, they do need to spike at all.

Another challenge is that the gradients become very large as a spike becomes closer to, but not sufficient for the postsynaptic neuron to reach the firing threshold. In this case, in (4) and (5), the value of the Lambert W function will approach its minimum (-1) as its argument approaches $-e^{-1}$, the denominator of the derivatives will approach zero and the derivatives will approach infinity. To counter this, we clip the derivatives to a fixed value `clip_derivative`. Note that this behavior will occur in any activation function that has a maximum (hence, a biologically plausible shape), is differentiable, and has a continuous derivative.

In addition to these hyperparameters, we explored several other heuristics. These included weight decay, adding random noise during training to the spike times of either the inputs or all nonoutput neurons in the network, averaging over brightness values in a convolutional-like manner, and adding additional input neurons responding to the inverted version of the image, akin to the ON/OFF bipolar cells in the retina. In addition, to improve the computation time of the network, we tried removing presynaptic neurons from the presynaptic set once their individual contribution to the potential decayed below a decay threshold. This can be achieved by solving an equation similar to (2) for reaching a decay threshold on the decaying part of the function, using the -1 branch of the Lambert W function. None of these techniques improved our results and, therefore, we did not include them here.

TABLE I

HYPERPARAMETERS OF THE MODEL. THE FIRST COLUMN SHOWS THE DEFAULT PARAMETERS CHOSEN TO SOLVE BOOLEAN LOGIC PROBLEMS. THE SECOND COLUMN SHOWS THE SEARCH RANGE USED IN THE HYPERPARAMETER SEARCH. ASTERISKS (*) MARK RANGES THAT WERE PROBED ACCORDING TO A LOGARITHMIC SCALE; ALL OTHERS WERE PROBED LINEARLY. THE LAST COLUMN SHOWS THE VALUE CHOSEN FROM THESE RANGES TO SOLVE MNIST

Parameter	Default value (Boolean tasks)	Search range	Chosen value (MNIST)
`batch_size`	1	$[1, 1000]^*$	5
`clip_derivative`	100.0	$[1, 1000]$	539.7
`decay_constant` (τ)	1.0	$[0.1, 2]$	0.181769
`fire_threshold` (θ)	1.0	$[0.1, 1.5]$	1.16732
`learning_rate`	0.001	$[10^{-5}, 1.0]^*$	$10^{-4} \times 2.01864$
`learning_rate_pulses`	0.001	$[10^{-5}, 1.0]^*$	$10^{-2} \times 5.95375$
`n_hidden`	1×2	$[0, 4] \times [2, 1000]^*$	1×340
`n_pulses`	1	$[0, 10]$	10
`nonpulse_init_multiplier`	0.0	$[-10, 10]$	-0.275419
`penalty_no_spike`	1.0	$[0, 100]$	48.3748
`pulse_init_multiplier`	0.0	$[-10, 10]$	7.83912

* - logarithmic search space

F. Event-Based Multispike Extension

We have so far shown an algorithm applicable to feed-forward spiking neural networks, where each neuron emits at most one spike. This can easily be extended to multi-spike models with custom architectures using an event-based implementation. In this version, the network architecture can be manually specified and can be cyclic. Pulses can be connected to any subset of neurons in the network. Neurons can spike more than once and a `refractory_period` parameter controls the amount of time for which a neuron ignores new inputs after spiking. The spikes are processed using a priority queue. The network can still be trained with backpropagation by accumulating updates for each spike, with a penalty applied to input connections where there was an input spike and no output spike. We provide an implementation in the `event_based` branch of our open-sourced code.

G. Comparison With SpikeProp Methods

Our algorithm can be considered to belong to the family of SpikeProp methods [32]. It is based on the same idea of temporal encoding for the inputs, but the encoding is analog (proportionally to pixel brightness) as opposed to discretized. Like SpikeProp, our model learns by adjusting spike timings through backpropagation. As in some extensions of SpikeProp [38], the output is encoded using rank order, first-to-spike coding. Learning is performed using the cross-entropy loss with a softmax layer, as also done in [39]. We arrive at a similar form for the derivatives, but our mathematical derivation is different: we operate on the alpha synaptic function directly to produce, solve, and differentiate the membrane potential function equation using the Lambert W function. In contrast with SpikeProp, there are no delays in our model; however, we introduce trainable pulses, which represent an alternative form of temporal delay while also being an essential component during training and in the spike time dynamics. In comparison, other extensions to SpikeProp [34] introduce trainable delays, time constants, and thresholds, which in our model we keep constant. We also provide our own weight initialization technique.

Most importantly, compared with other SpikeProp models, we provide a more in-depth analysis of the analog spiking dynamics on a larger network, an illustration of the reconstruction abilities of the trained spiking model, a mathematical proof for its universal approximation capabilities, and open-source implementation.

III. EXPERIMENTS

A. Boolean Logic Problems

We first tested the problem on noisy Boolean logic problems: AND, OR, and XOR. For each example, we encoded the two inputs as the individual spike times of the two input neurons. All spikes occurred at times between 0 and 1. We drew True and False values from uniform distributions between $[0.0, 0.45]$ and $[0.55, 1.0]$, respectively. The result is encoded as the first neuron to spike in the output layer. We also used a single pulse connected to all noninput neurons of the network.

We also solved a concentric circles problem, where the value represents a 2-D coordinate uniformly drawn from either an inner circle with a radius of 0.3 or an outer circle with an inner radius of 0.4 and an outer radius of 0.5.

We trained small networks containing two hidden neurons and one pulse to solve these problems. The networks were trained using the default parameters from Table I for a maximum of 100 epochs on 1000 training examples. They were tested on 150 randomly generated test examples from the same distribution.

We were able to train such spiking networks with 100% accuracy on all four problems. Fig. 2 exemplifies class boundaries for such networks. All trained networks in these examples had only positive weights.

Fig. 3 shows the dynamics of the membrane potential of every individual neuron during the classification of a typical XOR example. In this particular case, all neurons spike. With larger spacing between inputs, one of the hidden neurons no longer needs to spike. Given the limited range of inputs with respect to the decay constant, the classification occurs before any significant decay has occurred. However, the network can also be successfully trained with scaled inputs or decay constant so that the time to reach the alpha function peak is exceeded during the classification, in which case the

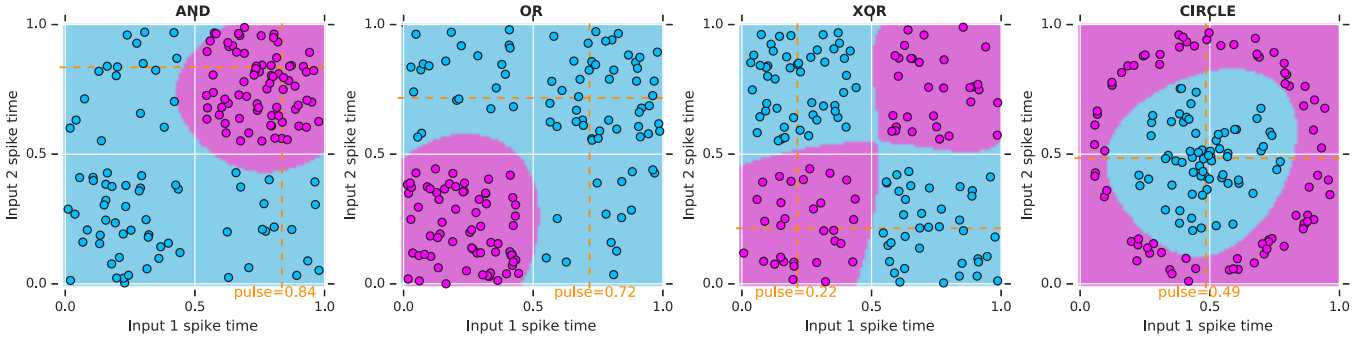

Fig. 2. Example of temporal class boundaries produced by small spiking networks with two hidden neurons and one pulse connected to all noninput neurons. All networks are trained for a maximum of 100 epochs on 1000 training examples, while 150 test examples are shown in Fig. 2. The default hyperparameters used for training are shown in Table I.

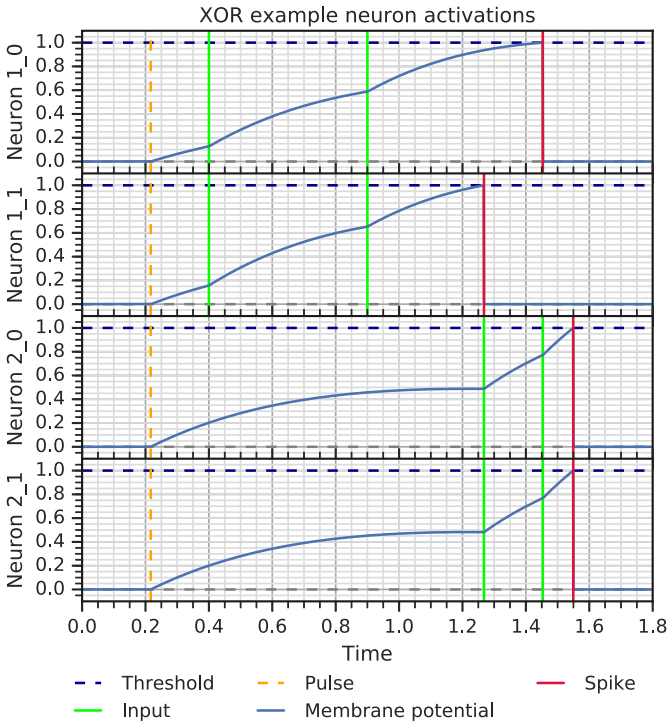

Fig. 3. Membrane potentials of hidden (1_∗) and output (2_∗) neurons during the classification of one XOR example, with incoming inputs at $t = 0.4$ and $t = 0.6$. Note that the output spike times are very close but not equal.

information will also propagate during the decay phase of the function.

The purpose of solving a set of simple problems was to demonstrate that spiking networks of small size can be trained without hyperparameter tuning, and hence, we did not focus on performance metrics for these problems.

B. Real-World Small Data Sets

We achieved 100% leave-one-out cross-validation accuracy on three widely used classification data sets obtained from [60]: iris petals [61], wine characteristics [62], and the Wisconsin breast cancer data set [63]. The data were scaled between 0 and 1; missing values were encoded as no spikes in the corresponding neurons. The examples were randomly

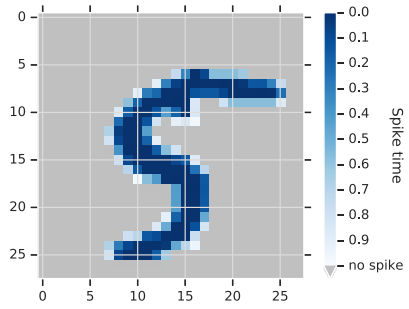

Fig. 4. Example of an MNIST digit temporal representation. Each of the 784 input neurons spikes at a time proportional with the brightness of the corresponding pixel in the flattened row-major brightness matrix.

shuffled. Hyperparameter searches were performed in each case, similar to the description given in Table I, but with smaller layer sizes.

C. Nonconvolutional MNIST

To solve the MNIST benchmark, we encoded the raw pixel brightness values in the time delay of the 784 neurons of the input layer (Fig. 4). All values were encoded as spikes occurring in the interval (0, 1). Darker pixels were encoded as earlier spikes compared with brighter pixels, as they represent more salient information. The scale of the brightness-to-temporal encoding was linear. Input neurons corresponding to white pixels did not spike.

The output of the network was encoded as the first neuron to spike among the ten neurons in the output layer. We used individual sets of pulses with fixed sizes connected to each individual layer, which allows for deeper architectures where the spike times of different layers become considerably delayed.

To find the optimal parameters for solving MNIST, we performed a hyperparameter search using evolutionary-neural hybrid agents [50]. This technique combines deep reinforcement learning with the method of evolutionary agents and has been shown to outperform both approaches in architecture search for a variety of classification tasks. It consists of an evolutionary agent encoding an architecture whose mutations are guided by a neural network trained with reinforcement

TABLE II

ACCURACY OF THE SPIKING NETWORK TRAINED WITH THE BEST SET OF HYPERPARAMETERS DURING THE SLOW AND FAST REGIMES THAT OCCURRED SPONTANEOUSLY DURING TRAINING

	Slow regime	Fast regime
Training accuracy (%)	99.9633	99.885
Training loss (mean)	0.002884	0.00444
Test accuracy (%)	97.96	97.4
Test loss (mean)	0.173248	0.19768

learning. We used Google Vizier as infrastructure for the hyperparameter search [64]. The parameter ranges are given in Table I.

The hyperparameter search includes the decay constant of the neurons, whereas the spike times of the inputs are fixed. In this work, we make no claims about the biological meaning of the temporal scales themselves, in particular, the spike times of the encoded problem relative to the decay constant of the model. Rather, we encode well-known problems in time in order to show the general capability of spiking models with temporal coding to solve nonlinear, complex problems.

Each trial in the hyperparameter search trained a network on a set of parameters suggested by the hyperparameter search agent within the given range. The 60 000 MNIST training examples were randomly split into a training set (90%) and a validation set (10%). The final reported objective to be maximized by the search agent was the accuracy on the validation set after 100 epochs. The search was run for 3 394 trials. We then chose the hyperparameters of the model as the set that produced the best objective value during the search, which is given in Table I. The hyperparameter values are given in Table I up to six-decimal precision, but in practice, an amount of imprecision can be tolerated.

By looking at the set of well-performing hyperparameters, we noted that, in general, they had a low batch size (up to 5), a small decay rate (between 0.1 and 0.3), and a nonpulse learning rate usually under 10^{-3}, whereas the other hyperparameters had comparatively more varied values.

We used the best set of hyperparameters to train three networks for 1000 epochs. This time we trained on the whole MNIST training data with no validation set and tested on the MNIST test data. As shown in Table II, the best out of the three networks reached a maximum accuracy of 99.96% and 97.96% on the MNIST train and test sets, respectively. This result improves on recent spiking models published recently, as detailed in Section IV.

For comparison, we also trained a fully connected conventional multilayer perceptron with rectified linear unit (ReLU) neurons with the same architecture as the spiking network (784–340–10 neurons) on MNIST. We used the same training setting as for the spiking network: Glorot initialization, Adam optimizer (learning rate 0.001), the same train/test split, and the same number of epochs. We tested batch sizes 5, 32, and 128. For each batch size, we ran the training three times. During the nine runs, we obtained a maximal accuracy of 97.9% on the test set. Although it can be claimed that various techniques can improve the performance of conventional

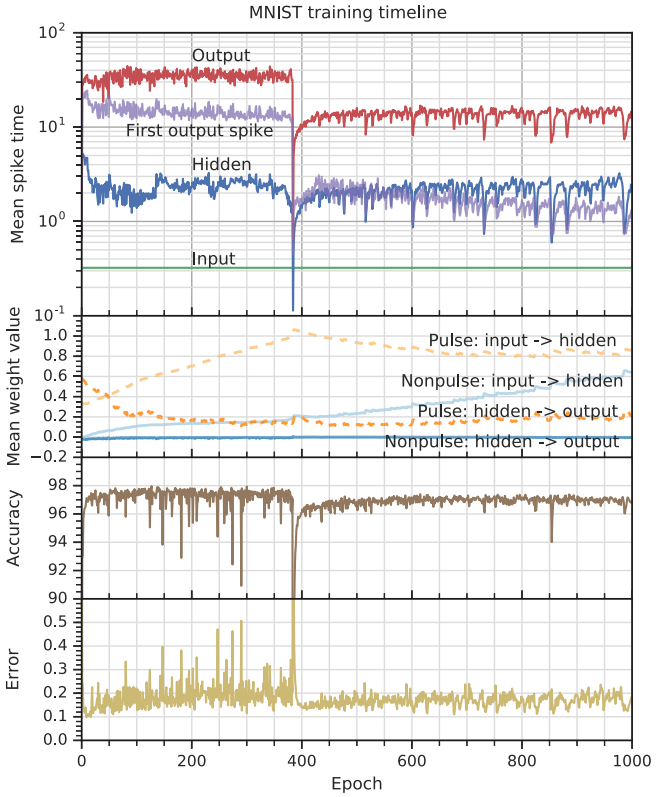

Fig. 5. Learning dynamics during the training of a spiking network with the best set of hyperparameters. The plots show a change in regime at epoch 384 from slow to fast classification. All statistics are shown for the test set.

neural networks (as it also may well be the case with the spiking model presented here), we aim to show that, generally speaking, the performances of the two types of networks are comparable.

During the training process, we observed the same spiking network spontaneously learning two different operating modes for the classification of inputs: a slow regime and a fast regime (Fig. 5). In the slow regime, the network spikes in a typical feedforward manner, with the neurons in the output layer usually waiting for all hidden neurons and pulses to spike before spiking themselves. The best accuracy is reported during the slow regime. On the other hand, in the fast regime, the network makes very fast decisions, with the first spike in the output layer occurring before the mean spike in the hidden layer. The transitions between the two regimes are sudden.

Investigating this transition, we found that the input layer pulses had an oscillatory behavior during training, likely due to a high learning rate as selected using hyperparameter search. The transition occurred when the pulses from the input layer synchronized and simultaneously reached a minimal spike time (0.0). This drove the hidden layer neurons and pulses to spike considerably earlier and closer in time to each other (Fig. 6). Consequently, through the earlier synchronization of the hidden layer spikes, the output spikes could also occur earlier, without the need to wait for the pulses. A suitable valley of the loss function was found and the configuration remained stable, despite the later recovery of the input layer pulses to larger and nonsynchronous spike times.

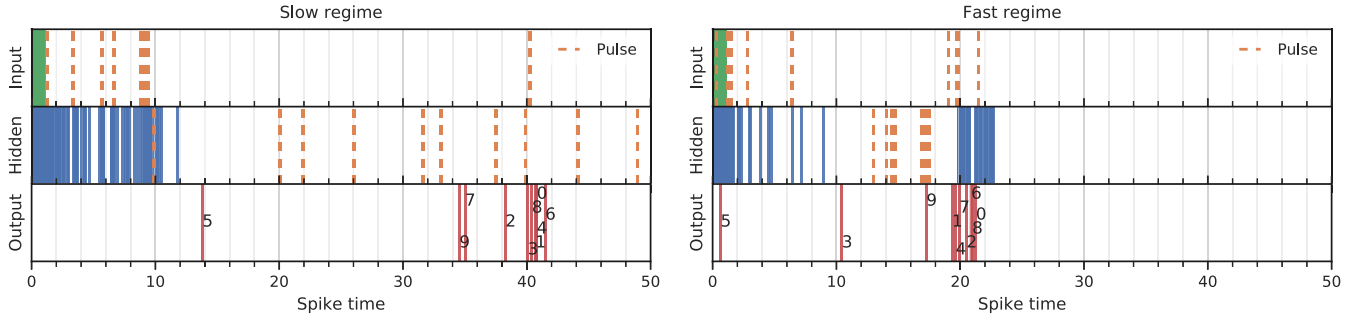

Fig. 6. Raster plot of spikes in each layer during the classification of the digit shown in Fig. 4 by the best-performing networks with slow and fast regimes. The pulses shown in each layer are connected to the neurons in the next layer. The number labels indicate the digit encoded by an output neuron.

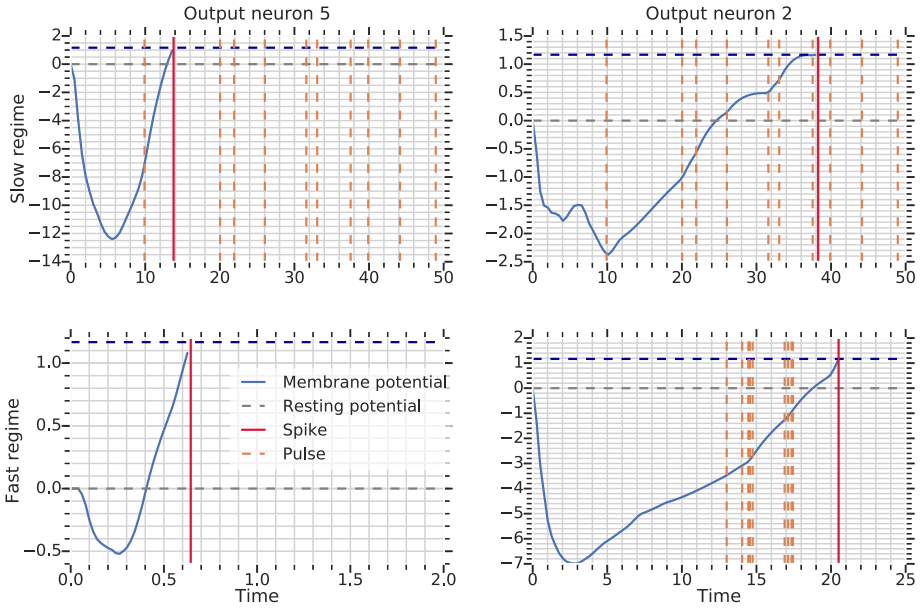

Fig. 7. Membrane potentials of two output target and nontarget neurons during the classification of the digit shown in Fig. 4 by the best-performing networks with slow and fast regimes. The axis scales are different for each plot, reflecting the considerable difference in speed of classification between the two regimes. Inhibition levels (reflected in negative amplitudes) are also considerably different in the two regimes. The membrane potential always becomes hyperpolarized (achieves negative potential) in output neurons before they spike.

It is, therefore, likely that, during the slow regime, the network learned enough that it no longer needed the pulses to produce some of the spikes accurately.

We observed the same transition from a slow to a fast regime occurring in one of the other two trained networks, whereas the third network went directly into a fast regime. With this set of hyperparameters, in particular the relatively high learning rate for pulses leading to oscillatory changes across training epochs, it is likely that the network has a preference for settling into the fast regime.

In the best performing slow and fast networks, we investigated the individual membrane potentials of the output neurons during the classification of individual examples (Fig. 7). We found that all output neurons were first inhibited before producing a spike, even in the case of the fast network. Moreover, in both networks, the winning neuron spiked before most or all of the hidden layer pulses spiked. This suggests that one role of the pulses was to produce late spikes where neurons would not have normally spiked, and thus, to allow the flow of gradients from nontarget neurons during training.

The optimal value for the decay constant (0.181769) chosen by the hyperparameter search means that the alpha function has its maximum at $t = 5.56$. The input spikes always occurred between 0 and 1. However, the pulses learned to produce spikes at considerably later times. This means that the decaying part of the function was exploited indeed, as the output neurons generally spiked only after a subset of the pulses.

D. Feature Visualization

We explored the representations learned by the spiking network using the family of "deep dream" methods as commonly applied in atemporal neural networks [65]–[67]. Given a network with fixed weights and pulses, a blank input image can be gradually adjusted in order to minimize the spike time of a particular neuron. Optionally, at the same time, the spike times of the other neurons in the same layer can be maximized. To do this, a one-hot encoded vector is set as the target for a noninput layer of the network and the derivative with respect

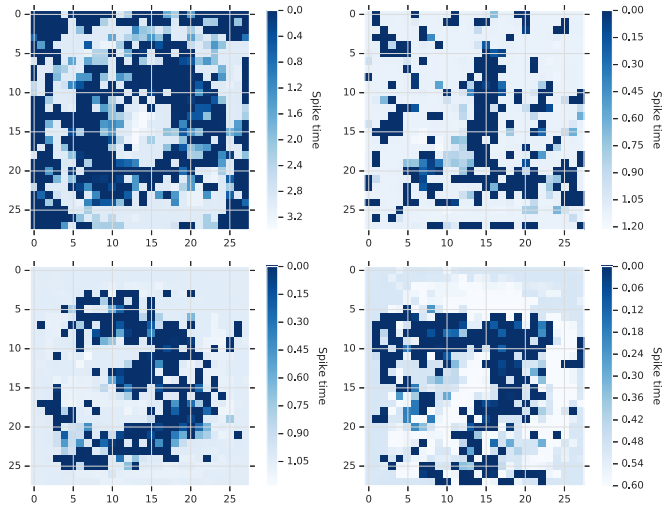

Fig. 8. Example of reconstructed input maps that produce a particular output class in the best performing slow regime network. The target classes are, in row-major order, 0, 1, 3, and 7. The start image is filled with zeros and adjusted using standard gradient descent until the target class is produced ten epochs in a row. Only nonnegative spike times are allowed.

to the spike time [(4)] is used to backpropagate errors to the input layer. Then, the same procedure is repeated with the adjusted image, until the output is close enough to the target.

We performed feature visualization on the network that performed best. We started from a blank image with each pixel initialized to spike at $t = 0.0$, though we confirmed that starting from a random image is also feasible for this procedure. We gradually adjusted the input image once per epoch with a learning rate of 0.1, until the classification produced the correct target class for ten consecutive epochs. As an additional constraint, we only allowed nonnegative spike times.

Using this method, we were able to produce recognizable digits (Fig. 8).

IV. Discussion

In this article, we proposed a spiking neural network with a biologically plausible alpha synaptic function that encodes information in the relative timing of individual spikes. The activation function is differentiable with respect to the time when the presynaptic set is fixed, thus making the network trainable using gradient descent. A set of pulses is used to provide bias and extra degrees of freedom during training. We showed that the network is able to successfully solve benchmark problems encoded in time, such as noisy Boolean logic problems and MNIST. Our code is openly available at [49].

Our model improves upon existing spiking network models, in particular those from the SpikeProp family, in multiple ways. First, we use a biologically plausible alpha synaptic function. The alpha function includes a decay component that allows earlier inputs to be forgotten when a neuron has not spiked, which can help in correcting potentially false alarms. We also use a continuous (as opposed to a discretized) scheme where the spike time of an input neuron is proportional to the value of a feature, such as the brightness of a pixel. Finally, we use trainable pulses connected to individual layers of the network, which provide temporal bias and allow extra flexibility during training. We presented a more detailed comparison in Section II-G.

Our results on the MNIST benchmark compare favorably with other recent models from literature, including similar models with larger or deeper architectures. Recent works on nonconvolutional spiking network whose benchmark results we improve include [28], [39], [68]–[71]. Recently, performant spiking networks have been obtained by conversion from atemporal neural networks [72]; however, we are not aware of such networks performing computations by encoding information in the timing of individual spikes.

The main challenge posed by this type of model is the discontinuous nature of the loss function at points where an output neuron stops spiking and due to the landscape changes at the points where the order of presynaptic and postsynaptic spikes changes. In general, discontinuities also appear in other deep learning models, such as the derivative of the ReLU activation function in atemporal networks. We found that the training process is able to overcome these challenges.

Another practical challenge is the computational complexity of the activation function and the training process. In conventional atemporal networks, the feedforward pass can be performed as a sequence of matrix multiplications, which can be efficiently parallelized on the GPU or other specialized hardware. In spiking networks, the feedforward pass cannot be parallelized in the same efficient fashion, leading to slower training times. In addition, solving the threshold crossing equation in the spiking network requires the computation of exponentials and solving the Lambert W function, which is relatively expensive. The networks presented in this work are trainable in the order of a few seconds to a few hours.

Despite challenges, there are two prominent goals motivating progress in the field of spiking neural networks. On the one hand, spiking neural networks are interesting as computational models of biological processes. They can contribute to understanding low-level information processing occurring in the brain and help us in the search for neural correlates of cognition [73]–[75]. On the other hand, they can be deployed in neuromorphic hardware [22], [76], [77] to implement rapid and energy-efficient computations.

From a computational biology perspective, the main contribution of this work is showing that it is possible to perform complex, nonlinear computations in networks with biologically plausible activation functions that encode information in the timing of single spikes. The timing of neuronal spikes, as opposed to spike rates, must play an important role in biological brains during the fast processing of perceptual stimuli. Neurons involved in the processing of sensory stimuli, such as ganglion cells in the retina, fire with very high precision, and convey more information through their timing than through their spike count [78]. The relative difference between the first spikes of different ganglion cells can encode the spatial structure of an image [4]. In the primary sensory neurons in human fingertips, the relative timing of the first spikes encodes information about tactile pressure and texture [6]. Further

constraints can be inferred from the response times of cortical neurons. In macaque brains, neurons in the temporal area can respond to a visual stimulus within 100 ms; given that the anatomy of the visual cortex indicates that at least ten synaptic stages must be passed to reach the temporal cortex, and given the conduction speed of intracortical fibers and the distance between the brain areas involved, it must be that the response is generated by single, or at most double, spikes in individual neurons [9]. It must, therefore, be that the temporal encoding of stimuli into single spikes fired by individual neurons is essential in rapid stimulus processing.

More generally, relative spike times are reflected in phase synchronization between neuronal populations, which can be observed both in individual-neuron studies [5] and in macroscale recording of brain activity [79]. Such synchronization has been shown to be reliable across different neural stages [5], [7], [8], [80], particularly in dynamic contexts [81]. The relative encoding of information in the precise timing of spikes produced by single neurons is thought to represent a solution to the binding problem—the ability of the brain to coordinate over features highly distributed neural representations, such as associating a color and a shape with a particular object in the environment [82], [83].

We do not claim that temporal coding is solely responsible for all of the complex cognitive processes performed by biological brains, but rather aim to draw attention that this type of encoding can be an efficient and computationally powerful alternative to rate coding. A reasonable hypothesis is that the two encoding methods are complementary in the brain. A temporal code can be transformed into a rate codes [84] and vice versa [85].

Another point that deserves discussion is backpropagation in biological networks. Our investigation was concerned with the representational capacity, as opposed to the learning mechanism of the network. We, therefore, used backpropagation to teach the spiking network. The idea of backpropagation in neural networks has for a long time been considered implausible [86], due to requirements, such as the existence of symmetrical connections between neuronal layers. However, recent works have begun to challenge this viewpoint and have made promising attempts to understand how backpropagation-like mechanisms might work in a biological network [87]–[91]. Notably, it has been shown that random feedback connections are able to support learning with backpropagation [90]. The main learning mechanism observed in biological neurons is spike-time-dependent plasticity [92], which can theoretically implement weight updates as required by backpropagation [91], [93]. This topic is nevertheless still an open question. Neuroscience is still uncovering the role of neural elements previously thought to have passive roles, which in fact appear to actively participate in synaptic regulation; they might also have significant roles for learning in the brain [94], [95].

V. Conclusion

We showed that a spiking neural network model with biologically plausible characteristics is able to solve standard machine learning benchmark problems using temporal coding. These results invite further exploration into the possibility of using such networks to generate behaviors resembling those of biological organisms and investigations into the computational capabilities of spiking neural networks with more complex and recurrent architectures. Further analyses of the spike dynamics and phase transitions will also be highly informative.

On a more speculative note, we expect that further advances in spiking networks with temporal coding will open exciting new possibilities for artificial intelligence, in the same way, that it has for biological brains. We envision that neural spike-based state machines will offer a natural solution for energy-efficient modeling and processing of real-world analog signals and interfacing between artificial and biological spiking networks. By observing and testing models, such as the one presented in this study, we aim to increase the familiarity of the research community with the temporal coding paradigm and to create building blocks toward recurrent and state-based architectures for neural computing.

Appendix
Universal Approximation Proof

Here, we prove that the proposed spiking network model is a universal approximator for sufficiently well-behaved functions.

Recall that the maximum voltage of a spike of positive weight w is obtained $(1/\tau)$ units of time after the spike itself, reaching a value of $(w/\tau e)$.

Lemma 1: For every $\epsilon > 0$, here is a spiking network with n inputs, each of them constrained to the [0, 1] range, and a single output that produces a spike in the interval $(t, t+\epsilon)$, with $t \geq 2+(2/\tau)$ if and only if each of the inputs belongs to a specified interval. Otherwise, no spike is produced. Moreover, it is possible to build one such network by using $2n+4$ neurons.

Proof: We will first prove the following.

Lemma 2: We can use one neuron and three auxiliary pulse inputs to conditionally produce a spike in the time interval $(t_{\text{out}}, t_{\text{out}}+\epsilon)$ if and only if the input spike happens at a time t_i that is smaller than (or bigger than) a given constant $t_0 < t_{\text{out}}$, provided that $t_{\text{out}} > 2+(1/\tau)$.

Proof: Let us consider the function

$$\Delta_x(t) = w(te^{-\tau t} - (t-T)e^{-\tau t+\tau T}) = we^{-\tau t}(t-(t-T)e^{\tau T}).$$

This describes the potential of a neuron that receives one spike at time 0 with positive weight w, and one at time $T \in [-1, 1]$ with negative weight $-w$; note that this formula is valid for $t \geq \max\{0, T\}$. The sign of this function is determined fully by the term $t-(t-T)e^{\tau T}$. Note that $t-(t-T)e^{\tau T} > 0 \Leftrightarrow t(e^{\tau T}-1) < Te^{\tau T}$.

If $T < 0$, then $e^{\tau T} < 1$, and thus, $t(e^{\tau T}-1) < Te^{\tau T} \Leftrightarrow t > -T(e^{\tau T})/(1-e^{\tau T})$. As $-T(e^{\tau T})/(1-e^{\tau T}) < (1/\tau)$ for $T \in [-1, 0)$, it follows that if $t > (1/\tau)$ then $\Delta_T(t) > 0$.

If $T > 0$, we have $t(e^{\tau T}-1) > Te^{\tau T} \Leftrightarrow t > T(e^{\tau T})/(e^{\tau T}-1)$. As $T(e^{\tau T})/(e^{\tau T}-1) < 1+(1/\tau)$, it follows that if $t > 1+(1/\tau)$ then $\Delta_T(t) < 0$.

Thus, we can consider this setup as follows.

1) An input spike at time $t_i \in [0, 1]$ with positive weight w.

2) A fixed spike at time $t_0 \in [0, 1]$ with negative weight $w < \theta\tau e$.
3) A fixed spike at time $t_{\text{out}} > 2+(1/\tau)$ with weight $v \geq \theta\tau e$.
4) A fixed spike at the time when the spike starting at t_{out} would reach the firing threshold if no other spikes were present, with a big enough negative weight to ensure that that spike reaches but does not cross the threshold by itself.

If $t_i \leq t_0$, then at time t_{out} the potential of the neuron is positive (as proven above). Thus, the fixed spike at t_0 will cross the firing threshold and produce an output spike after a small delay (which can be made smaller than ϵ by suitably increasing v). On the other hand, if $t_i > t_0$, the potential will be negative, and no output spike is produced.

In a similar way, we can have a configuration of neurons and weights that will produce a spike in the interval $(t_{\text{out}}, t_{\text{out}} + \epsilon)$ if and only if $t_i \geq t_0$. □

By connecting $2n$ of the configurations shown in Lemma 2 to a single neuron, having connections with weight $(\theta\tau e)/(2n-1) - \delta$ for a fixed $\delta > 0$ that depends on the desired ϵ, we obtain an output spike in the interval $(t_{\text{out}} + (1/\tau), t_{\text{out}} + (1/\tau) + \epsilon)$ if and only if the input belongs to a specified product of intervals. This requires the output spikes of Lemma 2 to belong to an interval that is small enough to ensure that any configuration of input spikes will still reach a potential of θ. This is always possible if the chosen ϵ in Lemma 2 is small enough.

Note that in the configuration from Lemma 2, all the fixed input spikes can be shared, except for the one defining the threshold. Thus, this setup requires at most $2n + 3 + 1$ neurons. □

Theorem 1: Let f be a continuous function from $[0, 1]^n$ to $(2 + (2/\tau), +\infty)$. Then, $\forall\epsilon > 0$, there is a spiking network with two hidden layers and decay constant τ that computes a function g satisfying $|f(x) - g(x)| < \epsilon \ \forall x \in [0, 1]^n$.

If f is also Lipschitz with constant K, then it is possible to realize such a network using at most $(3K\sqrt{n}\epsilon^{-1})^n(4n+3)+1$ neurons.

Proof: Since f is continuous on a compact set, it is uniformly continuous. Thus, for any $\epsilon > 0$, there is a δ such that, in each subdivision of the domain in n-dimensional boxes with extension along each coordinate axis being no more than δ, the difference between the maximum and the minimum of f is at most $(\epsilon/3)$. Moreover, if the function is Lipschitz with constant K, we can choose $(\epsilon/3)/(K\sqrt{n})$.

By Lemma 1, for each such box B there is a network that produces a spike in the time interval $(\min_B f, \min_B f + (\epsilon/3))$ if and only if the input belongs to the box. Connecting the outputs of those networks to a single neuron with sufficiently high weights, we can ensure that it will produce a spike with a delay of at most $(\epsilon/3)$ after receiving one as an input.

Thus, as a whole the network will produce a spike for input x with a time that is at most ϵ away from the value $f(x)$, proving the thesis. □

Note that the requirement for the function domain to be greater than some prefixed values is necessary: any network that uses temporal coding takes two inputs x and y and produces as an output $(x + y/2)$ would violate causality.

ACKNOWLEDGMENT

The authors would like to thank Robert Obryk for the valuable contributions in the early phase of this work.

REFERENCES

[1] D. Silver *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.
[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017.
[3] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.
[4] T. Gollisch and M. Meister, "Rapid neural coding in the retina with relative spike latencies," *Science*, vol. 319, no. 5866, pp. 1108–1111, Feb. 2008.
[5] A. Moiseff and M. Konishi, "Neuronal and behavioral sensitivity to binaural time differences in the owl," *J. Neurosci.*, vol. 1, no. 1, pp. 40–48, Jan. 1981.
[6] R. S. Johansson and I. Birznieks, "First spikes in ensembles of human tactile afferents code complex spatial fingertip events," *Nature Neurosci.*, vol. 7, no. 2, pp. 170–177, Feb. 2004.
[7] P. Reinagel and R. C. Reid, "Temporal coding of visual information in the thalamus," *J. Neurosci.*, vol. 20, pp. 5392–5400, Jul. 2000.
[8] J. Huxter, N. Burgess, and J. O'Keefe, "Independent rate and temporal coding in hippocampal pyramidal cells," *Nature*, vol. 425, pp. 828–832, Oct. 2003.
[9] S. J. Thorpe and M. Imbert, "Biological constraints on connectionist modelling," *Connectionism Perspective*, vol. 2016, pp. 1–36, Jul. 1989.
[10] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.
[11] B. Speelpenning, "Compiling fast partial derivatives of functions given by algorithms," Ph.D. dissertation, Dept. Comput. Sci., Illinois Univ., Champaign, IL, USA, 1980.
[12] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*, vol. 1. Cambridge, MA, USA: MIT Press, 1986, pp. 318–362.
[13] Y. Wu, L. Deng, G. Li, J. Zhu, and L. Shi, "Spatio-temporal backpropagation for training high-performance spiking neural networks," *Frontiers Neurosci.*, vol. 12, pp. 1–12, May 2018.
[14] G. Bellec, D. Salaj, A. Subramoney, R. Legenstein, and W. Maass, "Long short-term memory and learning-to-learn in networks of spiking neurons," Mar. 2018, *arXiv:1803.09574*. [Online]. Available: https://arxiv.org/abs/1803.09574
[15] C. Lee, S. S. Sarwar, and K. Roy, "Enabling spike-based backpropagation in state-of-the-art deep neural network architectures," Mar. 2019, *arXiv:1903.06379*. [Online]. Available: https://arxiv.org/abs/1903.06379
[16] D. Sussillo and L. F. Abbott, "Generating coherent patterns of activity from chaotic neural networks," *Neuron*, vol. 63, no. 4, pp. 544–557, Aug. 2009.
[17] Y. Xu, X. Zeng, L. Han, and J. Yang, "A supervised multi-spike learning algorithm based on gradient descent for spiking neural networks," *Neural Netw.*, vol. 43, pp. 99–113, Jul. 2013.
[18] R. V. Florian, "The chronotron: A neuron that learns to fire temporally precise spike patterns," *PLoS ONE*, vol. 7, no. 8, Aug. 2012, Art. no. e40233.
[19] R. Gütig and H. Sompolinsky, "The tempotron: A neuron that learns spike timing-based decisions," *Nature Neurosci.*, vol. 9, pp. 420–428, Feb. 2006.
[20] D. Huh and T. J. Sejnowski, "Gradient descent for spiking neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, Jun. 2017, pp. 1433–1443.
[21] F. Zenke and S. Ganguli, "SuperSpike: Supervised learning in multilayer spiking neural networks," *Neural Comput.*, vol. 30, no. 6, pp. 1514–1541, Jun. 2018.
[22] E. Neftci, C. Augustine, S. Paul, and G. Detorakis, "Event-driven random backpropagation: Enabling neuromorphic deep learning machines," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2017, pp. 1–4.
[23] I. Sporea and A. Grüning, "Supervised learning in multilayer spiking neural networks," *Neural Comput.*, vol. 25, no. 2, pp. 473–509, Feb. 2013.

[24] S. K. Esser, R. Appuswamy, P. Merolla, J. V. Arthur, and D. S. Modha, "Backpropagation for energy-efficient neuromorphic computing," in *Proc. Neural Inf. Process. Syst. (NIPS)*, 2015, pp. 1117–1125.

[25] A. Sengupta, Y. Ye, R. Wang, C. Liu, and K. Roy, "Going deeper in spiking neural networks: VGG and residual architectures," *Frontiers Neurosci.*, vol. 13, pp. 1–16, Mar. 2019.

[26] E. Hunsberger and C. Eliasmith, "Training spiking deep networks for neuromorphic hardware," 2016, *arXiv:1611.05141*. [Online]. Available: https://arxiv.org/abs/1611.05141

[27] B. Rueckauer and S.-C. Liu, "Conversion of analog to spiking neural networks using sparse temporal coding," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2018, pp. 1–5.

[28] P. U. Diehl, G. Zarrella, A. Cassidy, B. U. Pedroni, and E. Neftci, "Conversion of artificial recurrent neural networks to spiking neural networks for low-power neuromorphic hardware," in *Proc. IEEE Int. Conf. Rebooting Comput. (ICRC)*, Oct. 2016, pp. 1–8.

[29] Y. Hu, H. Tang, Y. Wang, and G. Pan, "Spiking deep residual network," Apr. 2018, *arXiv:1805.01352*. [Online]. Available: https://arxiv.org/abs/1805.01352

[30] N. G. Pavlidis, O. K. Tasoulis, V. P. Plagianakos, G. Nikiforidis, and M. N. Vrahatis, "Spiking neural network training using evolutionary algorithms," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, Jul./Aug. 2005, pp. 2190–2194.

[31] R. V. Florian, "Reinforcement learning through modulation of spike-timing-dependent synaptic plasticity," *Neural Comput.*, vol. 19, no. 6, pp. 1468–1502, Jun. 2007.

[32] S. M. Bohte, J. N. Kok, and H. La Poutré, "Error-backpropagation in temporally encoded networks of spiking neurons," *Neurocomputing*, vol. 48, nos. 1–4, pp. 17–37, Oct. 2002.

[33] C. Hong, X. Wei, J. Wang, B. Deng, H. Yu, and Y. Che, "Training spiking neural networks for cognitive tasks: A versatile framework compatible with various temporal codes," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 4, pp. 1285–1296, 2020, doi: 10.1109/TNNLS.2019.2919662.

[34] B. Schrauwen and J. Van Campenhout, "Extending spikeprop," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, Jul. 2004, pp. 471–475.

[35] S. McKennoch, D. Liu, and L. G. Bushnell, "Fast modifications of the SpikeProp algorithm," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, Jul. 2006, pp. 3970–3977.

[36] O. Booij and H. tat Nguyen, "A gradient descent rule for spiking neurons emitting multiple spikes," *Inf. Process. Lett.*, vol. 95, no. 6, pp. 552–558, Sep. 2005.

[37] F. Y. H. Ahmed, S. M. Shamsuddin, and S. Z. M. Hashim, "Improved spikeprop for using particle swarm optimization," *Math. Problems Eng.*, vol. 2013, Sep. 2013, Art. no. 257085.

[38] J. Wang, A. Belatreche, L. P. Maguire, and T. M. McGinnity, "SpikeTemp: An enhanced rank-order-based learning approach for spiking neural networks with adaptive structure," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 1, pp. 30–43, Jan. 2017.

[39] H. Mostafa, "Supervised learning based on temporal coding in spiking neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 7, pp. 3227–3235, Jul. 2018.

[40] S. R. Kheradpisheh, M. Ganjtabesh, S. J. Thorpe, and T. Masquelier, "STDP-based spiking deep convolutional neural networks for object recognition," *Neural Netw.*, vol. 99, pp. 56–67, Mar. 2018.

[41] M. Mozafari, S. R. Kheradpisheh, T. Masquelier, A. Nowzari-Dalini, and M. Ganjtabesh, "First-spike-based visual categorization using reward-modulated STDP," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 12, pp. 6178–6190, Dec. 2018.

[42] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *J. Physiol.*, vol. 117, no. 4, pp. 500–544, Aug. 1952.

[43] W. Gerstner, "A framework for spiking neuron models: The spike response model," in *Neuro-Informatics and Neural Modelling* (Handbook of Biological Physics), vol. 4, F. Moss and S. Gielen, Eds. North-Holland, 2001, ch. 12, pp. 469–516. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1383812101800154, doi: 10.1016/S1383-8121(01)80015-4.

[44] D. Sterratt, B. Graham, A. Gillies, and D. Willshaw, "The synapse," in *Principles of Computational Modelling in Neuroscience*. Cambridge, U.K.: Cambridge Univ. Press, 2018, pp. 172–195.

[45] W. Rall, "Distinguishing theoretical synaptic potentials computed for different soma-dendritic distributions of synaptic input," *J. Neurophysiol.*, vol. 30, no. 5, pp. 1138–1168, Sep. 1967.

[46] K. Frank, "Basic mechanisms of synaptic transmission in the central nervous system," *IRE Trans. Med. Electron.*, vol. ME-6, no. 2, pp. 85–88, Jun. 1959.

[47] R. E. Burke, "Composite nature of the monosynaptic excitatory postsynaptic potential," *J. Neurophysiol.*, vol. 30, no. 5, pp. 1114–1137, Sep. 1967.

[48] I. M. Comsa, K. Potempa, L. Versari, T. Fischbacher, A. Gesmundo, and J. Alakuijala, "Temporal coding in spiking neural networks with alpha synaptic function," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2020, pp. 8529–8533.

[49] *Ihmehimmeli Repository*. Accessed: Jan. 1, 2021. [Online]. Available: https://github.com/google/ihmehimmeli

[50] K. Maziarz *et al.*, "Evolutionary-neural hybrid agents for architecture search," Nov. 2018, *arXiv:1811.09828*. [Online]. Available: https://arxiv.org/abs/1811.09828

[51] C. Blakemore, R. H. S. Carpenter, and M. A. Georgeson, "Lateral inhibition between orientation detectors in the human visual system," *Nature*, vol. 228, no. 5266, pp. 37–39, Oct. 1970.

[52] J. H. Lambert, "Observationes variae in mathesin puram," *Acta Helvetica*, vol. 3, no. 1, pp. 128–168, 1758.

[53] R. M. Corless, G. H. Gonnet, D. E. G. Hare, D. J. Jeffrey, and D. E. Knuth, "On the LambertW function," *Adv. Comput. Math.*, vol. 5, no. 1, pp. 329–359, Dec. 1996.

[54] W. Klimesch, "Alpha-band oscillations, attention, and controlled access to stored information," *Trends Cognit. Sci.*, vol. 16, no. 12, pp. 606–617, Dec. 2012.

[55] G. Buzsaki, *Rhythms of the Brain*. Oxford, U.K.: Oxford Univ. Press, 2006.

[56] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: https://arxiv.org/abs/1412.6980

[57] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," *J. Mach. Learn. Res.*, vol. 9, pp. 249–256, May 2010.

[58] W. Klimesch, P. Sauseng, and S. Hanslmayr, "EEG alpha oscillations: The inhibition–timing hypothesis," *Brain Res. Rev.*, vol. 53, pp. 63–88, Jan. 2007.

[59] X.-J. J. Wang and G. Buzsáki, "Gamma oscillation by synaptic inhibition in a hippocampal interneuronal network model," *J. Neurosci.*, vol. 16, no. 20, pp. 6402–6413, Oct. 1996.

[60] D. Dua and C. Graff, "UCI machine learning repository," School Inf. Comput. Sci., Univ. California, Irvine, CA, USA, 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[61] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Ann. Eugenics*, vol. 7, no. 2, pp. 179–188, Sep. 1936.

[62] B. Vandeginste, "PARVUS: An extendable package of programs for data exploration, classification and correlation, M. Forina, R. Leardi, C. Armanino and S. Lanteri, Elsevier, Amsterdam, 1988, Price: US $645 ISBN 0-444-43012-1," *J. Chemometrics*, vol. 4, no. 2, pp. 191–193, 1990. [Online]. Available: https://analyticalsciencejournals.onlinelibrary.wiley.com/doi/abs/10.1002/cem.1180040210, doi: 10.1002/cem.1180040210.

[63] W. H. Wolberg and O. L. Mangasarian, "Multisurface method of pattern separation for medical diagnosis applied to breast cytology," *Proc. Nat. Acad. Sci. USA*, vol. 87, no. 23, pp. 9193–9196, Dec. 1990.

[64] D. Golovin, B. Solnik, S. Moitra, G. Kochanski, J. Karro, and D. Sculley, "Google vizier," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, New York, NY, USA, Aug. 2017, pp. 1487–1495.

[65] D. Erhan, Y. Bengio, A. Courville, and P. Vincent, "Visualizing higher-layer features of a deep network," Univ. Montreal, Tech. Rep. 1341, Jun. 2009.

[66] A. Mordvintsev, C. Olah, and M. Tyka, "Inceptionism: Going deeper into neural networks," 2015. [Online]. Available: https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html

[67] C. Olah, A. Mordvintsev, and L. Schubert, "Feature visualization," *Distill*, vol. 2, no. 11, p. e7, Nov. 2017.

[68] P. O'Connor and M. Welling, "Deep spiking networks," 2016, *arXiv:1602.08323*. [Online]. Available: https://arxiv.org/abs/1602.08323

[69] A. Tavanaei and A. Maida, "BP-STDP: Approximating backpropagation using spike timing dependent plasticity," *Neurocomputing*, vol. 330, pp. 39–47, Feb. 2019.

[70] D. Querlioz, O. Bichler, P. Dollfus, and C. Gamrat, "Immunity to device variations in a spiking neural network with memristive nanodevices," *IEEE Trans. Nanotechnol.*, vol. 12, no. 3, pp. 288–295, May 2013.

[71] J. M. Brader, W. Senn, and S. Fusi, "Learning real-world stimuli in a neural network with spike-driven synaptic dynamics," *Neural Comput.*, vol. 19, no. 11, pp. 2881–2912, Nov. 2007.

[72] A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. Maida, "Deep learning in spiking neural networks," *Neural Netw.*, vol. 111, pp. 47–63, Mar. 2019.

[73] L. M. Ward, "Synchronous neural oscillations and cognitive processes," *Trends Cognit. Sci.*, vol. 7, no. 12, pp. 553–559, Dec. 2003.

[74] M. Shanahan, "A spiking neuron model of cortical broadcast and competition," *Consciousness Cognition*, vol. 17, no. 1, pp. 288–303, Mar. 2008.
[75] D. Gamez, "Information integration based predictions about the conscious states of a spiking neural network," *Consciousness Cognition*, vol. 19, no. 1, pp. 294–310, Mar. 2010.
[76] M. Davies *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, Jan. 2018.
[77] M. Pfeiffer and T. Pfeil, "Deep learning with spiking neurons: Opportunities and challenges," *Frontiers Neurosci.*, vol. 12, p. 774, Oct. 2018.
[78] M. J. Berry, D. K. Warland, and M. Meister, "The structure and precision of retinal spike trains," *Proc. Nat. Acad. Sci. USA*, vol. 94, no. 10, pp. 5411–5416, May 1997.
[79] S. Palva and J. M. Palva, "The role of local and large-scale neuronal synchronization in human cognition," in *Multimodal Oscillation-Based Connectivity Theory*. Cham, Switzerland: Springer, 2016, pp. 51–67.
[80] M. Diesmann, M.-O. Gewaltig, and A. Aertsen, "Stable propagation of synchronous spiking in cortical neural networks," *Nature*, vol. 402, no. 6761, pp. 529–533, Dec. 1999.
[81] R. Lestienne, "Spike timing, synchronization and information processing on the sensory side of the central nervous system," *Prog. Neurobiol.*, vol. 65, no. 6, pp. 545–591, Dec. 2001.
[82] A. K. Engel, P. König, A. K. Kreiter, T. B. Schillen, and W. Singer, "Temporal coding in the visual cortex: New vistas on integration in the nervous system," *Trends Neurosci.*, vol. 15, no. 6, pp. 218–226, Jun. 1992.
[83] C. M. Gray, "Visual feature integration : Still alive and well," *Neuron*, vol. 24, pp. 31–47, Mar. 1999.
[84] E. Ahissar, "Temporal-code to rate-code conversion by neuronal phase-locked loops," *Neural Comput.*, vol. 10, no. 3, pp. 597–650, Apr. 1998.
[85] M. R. Mehta, A. K. Lee, and M. A. Wilson, "Role of experience and oscillations in transforming a rate code into a temporal code," *Nature*, vol. 417, no. 6890, pp. 741–746, Jun. 2002.
[86] F. Crick, "The recent excitement about neural networks," *Nature*, vol. 337, no. 6203, pp. 129–132, Jan. 1989.
[87] D. Hassabis, D. Kumaran, C. Summerfield, and M. Botvinick, "Neuroscience-inspired artificial intelligence," *Neuron*, vol. 95, no. 2, pp. 245–258, Jul. 2017.
[88] Q. Liao, J. Z. Leibo, and T. Poggio, "How important is weight symmetry in backpropagation?" in *Proc. AAAI Conf. Artif. Intell.*, Oct. 2015, pp. 1837–1844.
[89] Y. Bengio, "How auto-encoders could provide credit assignment in deep networks via target propagation," Jul. 2014, pp. 1–34, *arXiv:1407.7906*. [Online]. Available: https://arxiv.org/abs/1407.7906
[90] T. P. Lillicrap, D. Cownden, D. B. Tweed, and C. J. Akerman, "Random synaptic feedback weights support error backpropagation for deep learning," *Nature Commun.*, vol. 7, no. 1, p. 13276, Dec. 2016.
[91] J. C. R. Whittington and R. Bogacz, "An approximation of the error backpropagation algorithm in a predictive coding network with local Hebbian synaptic plasticity," *Neural Comput.*, vol. 29, no. 5, pp. 1229–1262, May 2017.
[92] S. Song, K. D. Miller, and L. F. Abbott, "Competitive Hebbian learning through spike-timing-dependent synaptic plasticity," *Nature Neurosci.*, vol. 3, no. 9, pp. 919–926, Sep. 2000.
[93] Y. Bengio, D.-H. Lee, J. Bornschein, T. Mesnard, and Z. Lin, "Towards biologically plausible deep learning," Feb. 2015, *arXiv:1502.04156*. [Online]. Available: https://arxiv.org/abs/1502.04156
[94] E. A. Newman, "New roles for astrocytes: Regulation of synaptic transmission," *Trends Neurosci.*, vol. 26, no. 10, pp. 536–542, Oct. 2003.
[95] C. Eroglu and B. A. Barres, "Regulation of synaptic connectivity by glia," *Nature*, vol. 468, no. 7321, pp. 223–231, Nov. 2010.

Iulia-Maria Comşa received the B.Sc. degree in computer science from Babeş-Bolyai University, Cluj-Napoca, Romania, in 2011, the M.Res. degree in computational neuroscience and cognitive robotics from the University of Birmingham, Birmingham, U.K., in 2012, and the Ph.D. degree in neurosciences from the University of Cambridge, Cambridge, U.K, in 2019.

She is currently a Research Engineer with Google Research Zürich, Zürich, Switzerland. Her current work explores biologically inspired artificial intelligence, including spiking neural networks. Her interdisciplinary interests include brain–computer interfaces, the neuroscience of consciousness, and the philosophy of mind.

Krzysztof Potempa was with Google Research Zürich, Zürich, Switzerland, where he conducted research in spiking neural networks and contributed to the JPEG XL image codec. He is currently a Quantitative Researcher with GSA Capital, London, U.K. His primary interests involve applications of modern machine learning techniques within the quantitative finance domain, optimization, generalization, and expressibility in deep learning, and game theory.

Luca Versari received the B.Sc. degree in mathematics from Scuola Normale Superiore, Pisa, Italy, in 2015, and the M.Sc. degree in computer science with a specialization in algorithms for pattern matching, graphs, hashing, and data compression from the University of Pisa, Pisa, in 2017, where he is currently pursuing the Ph.D. degree.

He is currently a Software Engineer with Google Research Zürich, Zürich, Switzerland, where he is also a Core Member of the JPEG XL Development Team. He is responsible for the algorithms and technical architecture of the image quality-related aspects of JPEG XL. He also works on the mathematical modeling and optimization aspects of spiking neural networks.

Thomas Fischbacher received the Diploma degree in physics from the Technical University of Munich, Munich, Germany, in 2000, and the Ph.D. degree in theoretical physics from the Humboldt University of Berlin, Berlin, Germany, in 2003.

He was a Lecturer in engineering physics with the University of Southampton, Southampton, U.K. He is currently a Software Engineer with Google Research Zürich, Zürich, Switzerland, where he works on a variety of compression and machine learning-related projects. His main research interests include human perception, M-theory, symbolic algebra, numerical computation, and finite-element modeling.

Andrea Gesmundo received the M.Sc. degree in computer science from the University of Padua, Padua, Italy, in 2007, and the Ph.D. degree in machine learning from the University of Geneva, Geneva, Switzerland, in 2013. He was an exchange student at the King's College, Aberdeen, U.K., for one year.

He is a Software Engineer with Google Research Zürich, Zürich, Switzerland, where he works on machine learning research.

Jyrki Alakuijala has developed computer-aided neurosurgery, radiation therapy treatment planning systems, and mobile games. He is currently a Tech Lead Manager with Google Research Zürich, Zürich, Switzerland. He is currently a Data Compression Researcher and an Active Member of the open-source software community. A sound installation co-composed by him is played daily on the Helsinki Senate Square at 05:49 PM. He has invented or coinvented the Zopfli, Butteraugli, Guetzli, WebP lossless, Brotli, WOFF2, Pik, Brunsli, and JPEG XL compression formats and algorithms. He has also developed other software tools, including hashing algorithms, CityHash and HighwayHash, a palette reduction algorithm, and the first multiuser chatbot for the Internet.