frontiers | Frontiers in Neuroscience

TYPE Original Research
PUBLISHED 24 August 2022
DOI 10.3389/fnins.2022.857513

OPEN ACCESS

EDITED BY
Jie Yang,
Shanghai Jiao Tong University, China

REVIEWED BY
Timothée Masquelier,
Centre National de la Recherche Scientifique (CNRS), France
Chankyu Lee,
Nvidia, United States

*CORRESPONDENCE
Feng Xu
fengxu@fudan.edu.cn

SPECIALTY SECTION
This article was submitted to Neuromorphic Engineering, a section of the journal Frontiers in Neuroscience

RECEIVED 18 January 2022
ACCEPTED 27 July 2022
PUBLISHED 24 August 2022

CITATION
Lu S and Xu F (2022) Linear leaky-integrate-and-fire neuron model based spiking neural networks and its mapping relationship to deep neural networks. *Front. Neurosci.* 16:857513. doi: 10.3389/fnins.2022.857513

Linear leaky-integrate-and-fire neuron model based spiking neural networks and its mapping relationship to deep neural networks

Sijia Lu and Feng Xu*

The Key Laboratory for Information Science of Electromagnetic Waves (MoE), Fudan University, Shanghai, China

Spiking neural networks (SNNs) are brain-inspired machine learning algorithms with merits such as biological plausibility and unsupervised learning capability. Previous works have shown that converting Artificial Neural Networks (ANNs) into SNNs is a practical and efficient approach for implementing an SNN. However, the basic principle and theoretical groundwork are lacking for training a non-accuracy-loss SNN. This paper establishes a precise mathematical mapping between the biological parameters of the Linear Leaky-Integrate-and-Fire model (LIF)/SNNs and the parameters of ReLU-AN/Deep Neural Networks (DNNs). Such mapping relationship is analytically proven under certain conditions and demonstrated by simulation and real data experiments. It can serve as the theoretical basis for the potential combination of the respective merits of the two categories of neural networks.

KEYWORDS

leaky integrate-and-fire model, spiking neural networks, rectified linear unit, equivalence, deep neural networks

1. Introduction

In recent decades, Artificial Intelligence (AI) has taken a path that has been rising, then falling, and is now under steady development. Based on the understanding of the human cerebral cortex's mechanism, ANN is formulated and becomes one of the primary directions for AI, called connectionism (McCulloch and Pitts, 1943). ANNs are composed of artificial neurons (ANs) connected as a graph. The weights of the connections, mimicking the cerebral cortex's synapses, represent the network's plasticity and can be trained *via* gradient descent (Ruder, 2016) in supervised learning tasks. With a large amount of annotated training data, a deep large-scale network structure, and computing power, DNNs have achieved great success in many application fields. They have become the most popular AI technology. The performance of a DNN can reach the human level on specific tasks, such as image recognition (Krizhevsky et al., 2012; He et al., 2016; Jiang et al., 2018; Zhao et al., 2019), instance segmentation (Cao et al., 2020), speech understanding (Hinton et al., 2012), strategic game playing (Mnih et al., 2013), etc.

DNNs employ a hierarchical structure with an exponentially-growing representation capacity. Such deep network structure was studied as early as the 1980s, but it was found difficult to train due to the vanishing of backpropagated gradients (Ivakhnenko and Lapa, 1965; Ivakhnenko, 1971; Schmidhuber, 2015). This problem was not solved until the deep learning era when the much simpler activation function called Rectified Linear Unit (ReLU) was used instead of conventional nonlinear functions such as the sigmoid (Jarrett et al., 2009; Glorot et al., 2011; Choromanska et al., 2014). Equipped with the ReLU activation function, DNNs have gained a powerful fitting capability on large-scale complex data. DNN is considered second-generation neural networks (Maass, 1997). It is widely considered that DNN's great success is attributable to big data, powerful computational technology (such as GPU), and training algorithms.

As DNNs are widely applied in real applications, limitations are becoming apparent. For example, strong dependence on labeled data and non-interpretability are considered drawbacks of deep learning. With the increase of layers and parameters, DNNs require many annotated data and computing power for training. However, current research mainly focuses on network architecture and algorithms designed for specific AI tasks. A technical approach to general artificial intelligence aims to break the limitations that remain studied. In this regard, many methods have been proposed, including SNN (Maass, 1997), which is regarded as the third generation of neural networks. SNN uses spiking neurons primarily of the leaky-Integrate-and-Fire (LIF) type (Lapicque, 1907), which exchange information *via* spikes. Due to its accurate modeling of biological neural network dynamics, SNN is the most popular brain-inspired AI approach (Tan et al., 2020). There have been extensive studies of SNN-derived neural networks, such as full connected SNN (Diehl and Cook, 2015), deep SNN (Illing et al., 2019; Tavanaei et al., 2019) and convolution SNN (Kheradpisheh et al., 2018). The learning mechanism of SNN includes supervised learning (such as spike backward propagation) (Kulkarni and Rajendran, 2018; Wang et al., 2020), unsupervised learning (such as spiking timing-dependent plasticity) (Tavanaei and Maida, 2016; Nazari and faez, 2018), and reinforcement learning (Mozafari et al., 2018).

However, SNNs have not yet achieved the performance of DNNs in many tasks. One of the most effective training algorithms is to transfer the trained weights of DNNs to SNNs with the same structure (Cao et al., 2015; Sengupta et al., 2019; Kim et al., 2020; Rathi et al., 2020). Establishing an effective SNN training algorithm or transformation mechanism is a challenging task. The fundamental question on the relationship between the second and third-generation neural networks is unclear.

The major contributions of this paper are as follows:

- The parameter mapping relationship between the Linear LIF neuron model and the ReLU-AN model is established.
- Inspired by the perspective of biology as well as the proposed equivalence, the ReLU activation function is proved to be the bridge between SNNs and DNNs.
- Experiments conducted on MNIST and CIFAR-10 datasets demonstrate the effectiveness and superiority of the proposed SNN composed of the Linear LIF model. The experimental validation under various simulation conditions is presented to prove the equivalence.

The rest of the paper is organized as follows. Section 2 explains the motivation of this study. Section 3 summarizes the related studies on ReLU-AN and the LIF model. Section 4 defines equivalence and presents the mapping relationship between Linear LIF model and ReLU-AN model. Simulations and analyses from single neuron to deep neural networks are carried out in Section 5. Finally, we make a brief conclusion and state the future opportunities in Section 6.

2. Motivation

2.1. Bridge the Gap between ANN and SNN

Brain science and cognitive neuroscience have been one of the essential sources of inspiration for artificial intelligence (Bear et al., 2007; Marblestone et al., 2016). From this perspective, we want to establish the relationship between ANNs and SNNs, which may bridge artificial intelligence and computational neuroscience. We believe that ANNs, the most powerful AI in real applications, and SNNs, the most biologically plausible technology, can learn from each other.

The biological neural model's complex dynamics and non-differentiable operations make SNNs lack scalable training algorithms. In this paper, we focus on the mechanism of transferring trained weights of DNN into SNN. While this method has achieved good results in target classification tasks, it has relatively strict limitations on pre-trained DNN, especially bias transformation. In the SNN conversion toolbox (SNN-TB) (Rueckauer et al., 2017), the bias is represented as a constant input current or an external spike input of constant rate proportional. However, we believe that bias in neuron model can be reflected in the biological neuron model, which we will show in the following simulations. In addition, the thickness and length of the axon of a neuron are different, and the neuron model parameters should also be different. This is not reflected in SNNs while some DNN-to-SNN algorithms use dynamic spiking threshold. We intend to establish the equivalent relationship between spiking neurons and artificial neurons and then the transformation mechanism of ANN.

2.2. A biological explanation of ReLU

DNNs use many layers of nested nonlinearity to fit massive amounts of data and perform better in machine learning tasks with ReLU. We focus on the nonlinearity and sparsity of ReLU, but we do not have a deep understanding of why the ReLU performs better than the other activation function. Glorot et al. (2011) indicates that ReLU can bridge the gap between the computational neuroscience model and the machine learning neural network model. But under what conditions, i.e. coding algorithms and parameters, RELU can be equivalent to the biological model, and what is the mathematical mapping relationship between the two models. This is still a fundamental question in biologically inspired AI that remains unanswered.

2.3. A new approach of unsupervised learning

The unsupervised learning mechanism employed in SNN has a good biological basis and emphasizes the causal relations between the signals, which complements conventional machine learning. Unsupervised learning is generally regarded as a representation learning that estimates a model representing the distribution for a new input x_n given previous inputs $x_1, x_2, \ldots, x_{n-1}$, expressed as $\mathrm{P}\left(x_n \mid x_1, x_2, \ldots, x_{n-1}\right)$ (Ghahramani, 2003). Computational neuroscience has provided a new idea for unsupervised learning mechanisms. Spiking Time Dependent Plasticity (STDP) (Abbott and Nelson, 2000; Song et al., 2000; Caporale and Dan, 2008; Tavanaei et al., 2018; Falez et al., 2019), is a temporally asymmetric form of Hebbian learning and is the most widely used unsupervised learning mechanism in SNN. In the temporal dimension, the relation between the presynaptic action potential and the postsynaptic action potential regulates the neurons' weights, which is a feature unique to SNN. Suppose we want to migrate such a natural learning mechanism in the time domain from SNNs to DNNs. In that case, we first need to establish a mathematical mapping relationship between SNN's neuron model and DNN's neuron model.

2.4. Inspire the development of artificial intelligence

SNN has its unique advantages in information transmission and learning mechanisms. Although ANN is historically brain-inspired, ANN and SNN are entirely different. First, SNN uses event-driven characteristics to reduce power consumption. SNN transfer and process the information *via* spike train (Tavanaei et al., 2019), while DNN uses scalars to represent the neural signals. For the same task, e.g., image and voice recognition, the human brain typically consumes 10–20 watts (Jeong et al., 2016), compared to hundreds of thousands of watts for DNNs running on a computer. Secondly, the neurokinetic calculation is not a conventional von Neumann architecture but adopts an integrated structure of storage and calculation, storing information in neurons. The mechanism of time-domain processing in spike trains and Hebbian learning-based synaptic plasticity are considered potential routes to a more advanced artificial intelligence (Hebb, 1949; Song et al., 2000; Denham, 2001).

3. Related work

3.1. ReLU artificial neuron

Artificial Neuron (AN) is a mathematical function that can model a biological neuron. McCulloch and Pitts proposed the Artificial Neuron model in 1943. It is also known as the M-P model and is still used today. As the basic unit of the neural network, it receives input signals from previous layer units or perhaps from an external source. Each input has an associated weight ω, which can be adjusted to model the synaptic plasticity. Through an activation function $f(\ )$, the unit converts the integrated signal, i.e., the weighted sum of all the inputs, to obtain its output

$$y_i = f(\sum_j \omega_{ij} x_j + b_i) \tag{1}$$

Here, ω_{ij} is the weight from unit j to unit i, b_i is the bias of unit i, and $f(\ )$ is the activation function. For the M-P model, the form of activation function is the Heaviside step function. The working model of neurons has two states, activation (1) and inhibition (0). The main idea of deep learning is still very similar to the perceptron proposed by Frank Rosenblatt many years ago, but the binary Heaviside step function is no longer used. Neural networks mostly use the ReLU activation function.

ReLU activation function has been developed for a long time. Cognitron (Fukushima, 1975) is considered the first artificial neural network using a multi-layered and hierarchical design. This paper also proposed the initial form of ReLU, i.e., $max(0, x)$, shown in Figure 1. The activation function, a rectification nonlinearity theory, applies to the Symmetric Threshold-Liner network dynamics (Hahnloser et al., 2003). Pinto and Cox proposed a V1-like recognition system, in which the outputs of the Gabor filter will pass through a standard output nonlinearity—a threshold and response saturation (Pinto et al., 2008). Jarrett and Kavukcuoglu have proved that rectified nonlinearities are the single most crucial ingredient for deep learning (Jarrett et al., 2009). ReLU was then introduced to enhance the ability in the feature learning of restricted Boltzmann machines. Compared with the sigmoid function,

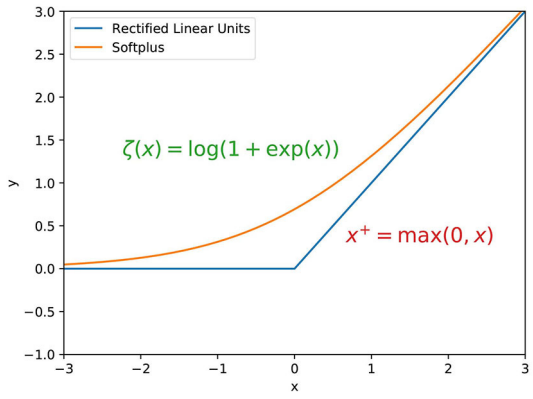

FIGURE 1
Rectified Linear Unit and softplus activation functions.

it has achieved better image classification accuracy (Nair and Hinton, 2010). Glorot et al. (2011) showed that neurons with ReLU activation function have a better performance than hyperbolic tangent networks and analyzed the advantage of sparsity. In the deep learning era, ReLU was crucial to the training of deep neural networks.

The critical characteristics of ReLU are: A. Nonlinearity: Introducing nonlinearity is critical for deep neural networks. With the simple rectification, it provides the fundamental nonlinearity required for data fitting. B. Sparsity: Nearly half of the neuron's outputs are suppressed. This mechanism is similar to lateral inhibition in biological neural networks (Amari, 1977) and increases the neural network's sparsity.

3.2. LIF neuron model

In this subsection, we present the development of biological neuron models and analysis of LIF model's dynamic properties.

3.2.1. Development of the biological neuron

Research on biological neuron models can be dated back to the 1900s, referred to Lapicque model (Lapicque, 1907; Abbott, 1999), which is employed in the calculation of firing times. The Lapicque model is considered the earliest form of the "integrate-and-fire model," and it becomes the LIF model after adding an attenuation term. The LIF model is one of the most popularly used models for analyzing the nervous system's behavior (Diehl and Cook, 2015; Kheradpisheh et al., 2018). Unlike the neuron models used for computing, some neuron models have also been created and applied to simulate real neuron propagation potentials. The Hodgkin–Huxley model (HH model) (Hodgkin and Huxley, 1952) was proposed by analyzing the electric current flow through the surface membrane. We call it a simulation-oriented neuron model. However, it is not practically applied in general neural networks due to its computational complexity. For computational feasibility, simplified models have emerged, referred to as computation-oriented models. Izhikevich model (Izhikevich, 2003) is a simplification of the HH model based on the theory of dynamic systems. In this paper, we focus on the dynamic properties of the LIF model and analyze its equivalence with ReLU.

3.2.2. The dynamic of LIF neuron model

The LIF model can be modeled as a circuit composed of a resistor and a capacitor in parallel, which, respectively, represent the leakage and capacitance of the membrane (Tuckwell, 1988). The integrate-and-fire neuron model is described by the dynamics of the neuron's membrane potential (MP), $V(t)$,

$$C_m \frac{dV(t)}{dt} + \frac{V(t) - V_0}{R_m} = I_{inj}, \tag{2}$$

where C_m and R_m denote the membrane capacitance and resistance, respectively. V is the membrane potential of LIF model, V_0 is the resting potential, and I_{inj} is the current injected into the neuron. The driving current can be split into two components, $I(t) = I_{C_m} + I_{R_m}$ (Gerstner and Kistler, 2002). The first part on the left of Equation (2) represents the current passing through the capacitor during charging. According to the definition of capacitor $C = Q/U$ (where Q is the charge and U is the voltage), we find that the membrane capacitance current $I_{C_m} = C_m dV/dt$. The second part represents the leak of the membrane through the linear resistor R_m, and the membrane time constant $\tau_m = C_m R_m$ of the "leaky integrator" is introduced (Burkitt, 2006).

Given the initial value of membrane potential and injected current I_{inj}, we can use the method of integrating factors (Maday et al., 1990) to solve the differential equations which define the change of membrane potential, and the simplified equation can be expressed as

$$V(t) = e^{-\frac{t-t_0}{\tau_m}} \left[\int_{t_0}^{t} \frac{I_{inj}(t')}{C_m} e^{\frac{t'-t_0}{\tau_m}} dt' + V(t_0) \right] \tag{3}$$

where $V(t_0)$ is the membrane potential at the initial time t_0, and we take the reset potential to be $V_{reset} = 0$ for the sake of simplicity. When the neuron has no input, i.e. $I_{inj} = 0$, the integral term in Equation (3) is 0, and the membrane potential decays exponentially on the basis of the initial potential $V(t_0)$; when there is input, the input current is integrated into the post-neuronal membrane potential.

To explore a self-consistent neuron model, neurons' input and output forms should be the same. The input current, $I_{inj}(t)$,

is defined as the weighted summation of pre-synaptic spikes at each time step,

$$I_{inj}(t) = \sum_{i=1}^{n^l} \omega_i \cdot \sum_{j=1}^{n} \delta(t - t_j) \tag{4}$$

where n^l indicates the number of pre-synaptic weights, n is the number of spikes of pre-synaptic spike train, ω_i gives the connection weights between the pre-synaptic neuron i and post-synaptic neuron, and δ is the Dirac function. The top of Figure 2 shows the dynamics of membrane potentials of an LIF neuron with multiple input spikes.

Assuming that a spiking input signal with a period of T (frequency $f = 1/T$ and $t_j = j/f$), the value of MP can be described as:

$$V(nT^+) = \frac{\omega}{C_m} \cdot \frac{1 - e^{-nT/\tau_m}}{1 - e^{T/\tau_m}} \tag{5}$$

The membrane potential accumulates with the presence of inputs $I(t)$. Once the membrane potential $V(t)$ exceeds the spiking threshold V_{th}, the neuron fires an action potential, and the membrane potential $V(t)$ goes back to the resting potential V_0. The LIF model is a typical nonlinear system. Three discrete equations can describe the charge, discharge, and fire of the LIF model:

$$\begin{aligned} H(t) &= f(V(t-1), I(t)) \\ S(t) &= \Theta(H(t) - V_{th}) \end{aligned} \tag{6}$$

where the $H(t)$ is the membrane potential before spike, $S(t)$ is the spike train and $f(V(t-1), I(t))$ is the update equation of membrane potential.

4. The mapping relationship between LIF neuron model and ReLU-AN model

Converting CNNs into SNNs is an effective training method that enables mapping CNNs to spike-based hardware architectures. Many scholars believe that the theoretical equivalence between the spiking neuron model and the artificial neuron model is the basis of the transformation method. This section presents a mapping relationship between the Linear LIF model and ReLU-AN model.

Many differences exist between the neural network models used in machine learning and those used in computational neuroscience. Glorot et al. (2011) shows that the ReLU activation function can bridge the gap between these two neuron models, including the sparse information coding and non-linear. Mainly based on changing the activation function from *tanh*() to $HalfRect(x) = max(x, 0)$, which is named ReLU and is nowadays the standard model for the neuron in DNNs. Cao et al. (2015) proposed a method for converting trained CNN to SNN with slight performance loss. However, the theoretical groundwork of converting basic principles is lacking, and related research only shows the similarity between the LIF neuron model and the AN model. Rueckauer et al. (2017) present a one-to-one correspondence between an ANN unit and an SNN neuron and an analytical explanation for the approximation. Han et al. (2020) proposed a loss-less ANN-SNN conversion method using "soft reset" spiking neuron model, and an illustration of ReLU-IF mapping. On this basis, this paper further presents an exact correspondence between the parameters of LIF neuron model and ReLU-AN model.

4.1. Linear leaky-integrate-and-fire model

Once the membrane potential reaches the spiking threshold, an action potential will be exceeded. Then the membrane potential will be reset: Reset-to-Zero, used, e.g., in Diehl and Cook (2015), reset the membrane potential to zero. Linear-Reset retains the attenuation term that exceeds the threshold:

$$V(t) = \begin{cases} H(t) \cdot (1 - S(t)) & \text{Reset-to-Zero} \\ H(t) \cdot (1 - S(t)) + (H(t) - V_{reset}) \cdot S(t) & \text{Linear-Reset} \end{cases} \tag{7}$$

The LIF neuron model with "Linear Reset" is named Linear LIF model. Diehl et al. (2016) and Rueckauer et al. (2017) analyzed the difference between these two MP reset modes and chose the Linear LIF model for simulation. We analyze the two models from the perspective of physics and information theory and determine the advantages of the Linear LIF model. The membrane potential of the Reset-to-Zero LIF model does not satisfy the law of conservation of energy. There are two parts of membrane potential attenuations: "leaky", the attenuations as the form of conductance in the circuit which keeps the nonlinear dynamic properties. The other part is that when the action potential is exceeded, the membrane potential exceeding the spike threshold will be lost directly, resulting in energy non-conservation. From the perspective of information, the Linear LIF neuron model maintains the nonlinearity and retains the completion of information to the greatest extent. The Linear LIF model's membrane potential is shown in Figure 2 compared with Reset-to-Zero LIF model under the same input. More detail can be seen in Supplementary material 1.1.

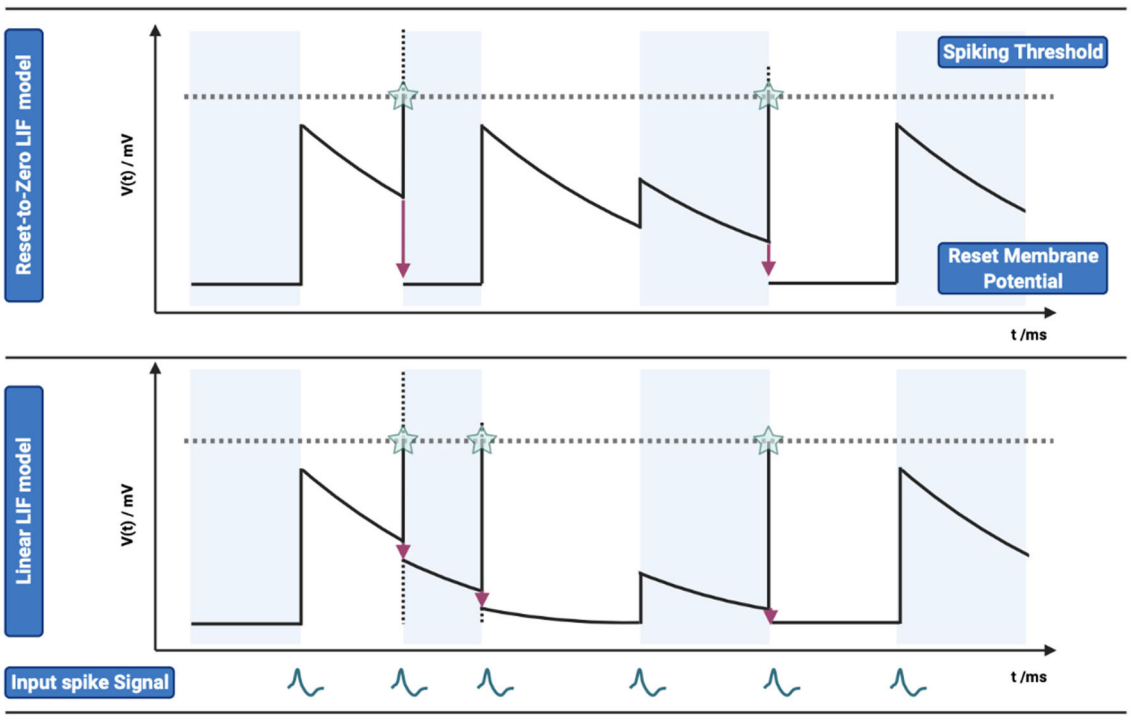

FIGURE 2
Comparison of membrane potential changes between Reset-to-Zero LIF model and Linear LIF model. **(TOP)** Reset-to-Zero LIF model, the membrane potential will be reset to zero ($V_{reset} = 0\ mV$). The subfigure shows the input action potential, membrane potential change, spiking threshold and output action potential from bottom to top. **(Bottom)** Linear LIF model, membrane potential will subtract the spiking threshold at the time when it exceeds the threshold.

4.2. Information transmission between LIF neuron models

The linear LIF neuron model has two steps in information transmission. The information is integrated by connection weight ω, and then the data is processed and transmitted, as shown in Figure 3. We can see that the LIF model and Linear LIF model are the same for subthreshold membrane potential changes in Figure 2. However, the resulting simulation proves that Linear LIF model is more similar to ReLU-AN model than LIF model.

4.3. Information coding

Neuronal coding is a key step of the simulation. Neurons use sequences of action potentials, which can be considered as a point process, to carry information from one node to another in the brain. This spiking train can be considered an element of neural coding. The shape and duration of the individual spikes generated by a given neuron are very similar, so we think that the spike train can be described as a train of one-or-none point events in time (Kostal et al., 2007).

The information received by the LIF neuron is a series of spike sequences, so the original data needs to be encoded into a spike train (Richmond, 2009). Many coding methods are used in biological neuron models, such as one-dimensional coding and sparse coding mechanism. We assume that the neuron encodes the information as a spike frequency and encodes it according to the following rules:

$$I_l^i = \omega_i \cdot \sum_{j=1}^{n} \delta(t - \frac{j}{x_i}), \quad \frac{n}{x_i} <= T_w \tag{8}$$

where x_i ($x_i \in [0, 1]$) is the input. We encode the input as a series of spike sequences with a fixed frequency, which is proportional to the input data. The encoding time (time window) is T_w, sampling frequency is the maximum frequency of encoding $R_{max} = 1/\Delta t$. In our experiment, we set the minimum time unit Δt as 0.01s and the time window T_w as 3s. That is, a pixel is encoded into a sequence with a length of 300 (time steps).

4.4. Mapping relationship between linear LIF model and ReLU-AN model

This subsection presents the mapping relationship between the Linear LIF model and the ReLU-AN model from three

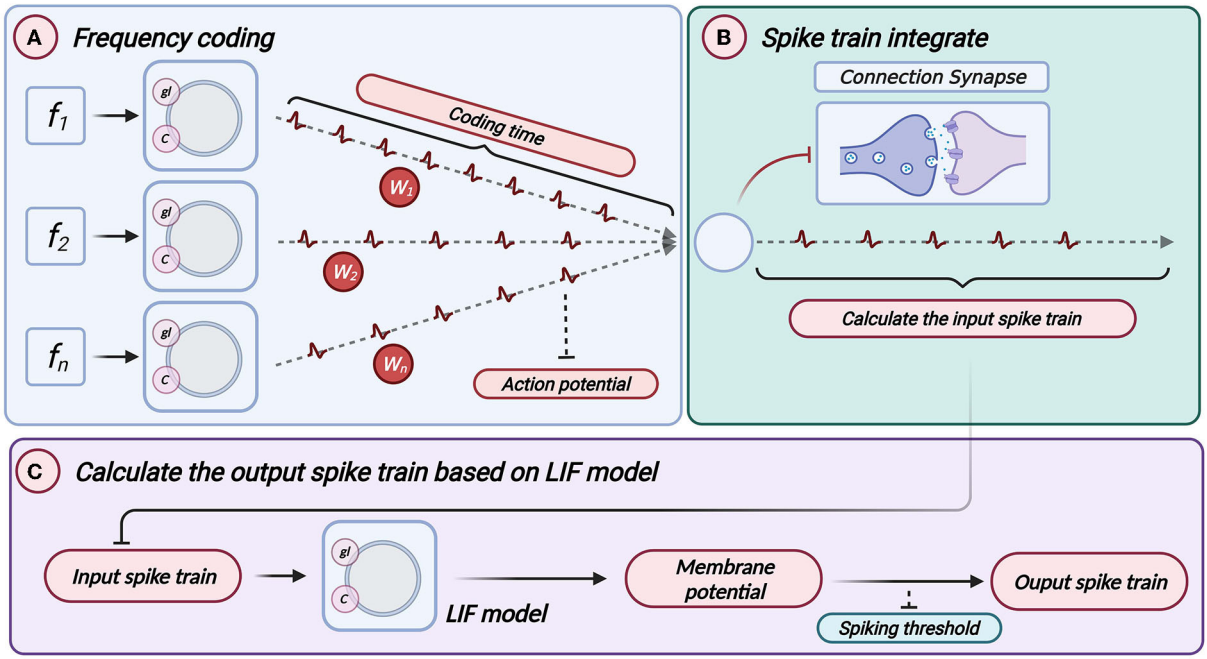

FIGURE 3
Information transmission of Linear LIF model. **(A)** All inputs are encoded as a spike train, multiplied by a weight. This is the information integration shown in the left part of the figure. **(B)** Add all input spikes to get the input spike sequence of the neuron. **(C)** The output action potential is obtained according to the input spike sequence through the LIF neuron's information processing mechanism.

aspects, i.e., weights, bias, and slope of the ReLU activation function. In comparison, the parameters of the linear LIF neuron are more biologically plausible, including the membrane capacitance C_m and the membrane resistance R_m. Here we give the parameter mapping we established in Table 1, which will be discussed in detail later in this paper. A more detailed derivation process is shown in Supplementary material 1.2.

4.4.1. Mapping of the weights

We assume that the spiking frequency of the input signal is f_j and the amplitude is 1, and then the signal can be expressed as:

$$I_l = \sum_{i=1}^{n^l} \omega_i \cdot \sum_{j=1}^{n} \delta(t - j\frac{1}{f_i}) \tag{9}$$

The ω_i is the synaptic weight between presynaptic neuron i and post-synaptic neuron, n^l represents the number of neurons in layer l, j represents the j_{th} action potential in the input spike train, n is the number of action potentials and T is the time windows of simulation.

Compared with the weight integration process in ANNs, we integrate the input signal I_l in the time window $[0, T_w]$ and obtain:

$$\int_0^{T_w} I_l dt = \sum_{i=1}^{n^l} \omega_i \cdot \sum_{j=1}^{n} \int_0^{T_w} \delta(t - j\frac{1}{f_i}) dt = T \cdot \sum_{i=1}^{n^l} \omega_i f_i \tag{10}$$

This information integration mechanism is similar to the ReLU-AN model ($f = \sum \omega x$), except that for time-domain signals, the weight is reflected in the amplitude of the input action potential. We encode the weights in the amplitude of the spiking train but not in the Linear LIF model's conductance. Note that the encoding mechanism and information integration mechanism play a vital role in the network's information transmission process.

4.4.2. Mapping of the bias

The bias is an additional parameter in the ReLU model used to adjust the output along with the weighted sum of the inputs. Moreover, a learnable bias allows one to shift the activation function to either the right or the left. The neuron model's bias has a similar role with the threshold, determining whether the input activates the output. Based on Equation (1) and the activation function, we know that the output of the neuron is equal to 0 if $\sum_{i=1}^{n} w_i x_i < -b$.

Similarly, it cannot be fired if the input frequency is less than a threshold. According to the spike excitation rules of Linear LIF neurons, the action potential is generated when the membrane potential reaches the spiking threshold. If the integrated spike train can excite an action potential within the time window T_w (set to 4s in the simulation here), it should satisfy

TABLE 1 Parameter mapping between ReLU-AN model and Linear LIF model.

Parameter of ReLU		Params of linear LIF	
Symbol	**Description**	**Symbol**	**Description**
ω	Connection weight	ω	Synaptic weight
b	Bias	$-\sum \omega/R_m C_m \cdot \ln(1-\sum \omega/(V_{th}C_m))$	
k	Slope of activation function	$1/V_{th}C_m$	

$$V(T_w = n/f_{in}) > V_{th} \tag{11}$$

where T_w refers to the time window, and n is the number of action potentials within the time window at the current frequency. According to Equation (5), we can get the relation between the number of action potentials and the other parameters.

$$\sum_{i=1}^{n^l} \omega_i f_i > \frac{-\sum_{i=1}^{n^l} \omega_i}{\tau_m \cdot \ln(1-\sum_{i=1}^{n^l} \omega_i/(V_{th}C_m) \cdot (1-e^{-T/\tau_m}))} \tag{12}$$

The Linear LIF model can excite the action potential in the time window only if the inequality is satisfied, that is when the input frequency is greater than a value determined by the parameters of the Linear LIF model. This mechanism has the same effect as the bias, which filters the input signal before the information processing. Bias of the ReLU-AN model also can be expressed as a function of membrane resistance R_m and membrane capacitance C_m. The existence of membrane resistance R_m endows the Linear LIF model with nonlinearity, which is very important for neural networks.

4.4.3. Mapping of activation function

The activation function determines the relationship between input and output after integration. We focus on the non-negativity and linear relationship of ReLU. For non-negativity, the Linear LIF model's output is based on the number of action potentials, a non-negative value. So, for the input-output relationship where the input is greater than 0. The relationship between integrated input and output spike frequencies of the LIF neurons can be expressed as

$$f_o = \frac{f_{in}}{\lfloor n \rfloor} \quad \{n \mid V(n/f_{in}) \geq V_{th}\} \tag{13}$$

where f_{in} is the frequency of input signal, f_o is the frequency of the output signal, and n is the minimum number of input spikes capable of firing an action potential within a time window, defined by Equation (5).

We rounded up the number of input spikes and approximated the relation between the input and output frequency as

$$f_o = \tau_m [\frac{[(V_{th}C_m)/\sum_{i=1}^{n^l} \omega_i](1-e^{-1/f_i\tau_m})}{1-[(V_{th}C_m)/\sum_{i=1}^{n^l} \omega_i](1-e^{-1/f_i\tau_m})}] \tag{14}$$

Based on the action potential frequency in biological neurons, we assume that the input frequency f_{in} satisfies $-1/(f_i\tau_m) \approx 0$. According to the approximation formulas $\ln(1+x) \approx x$ and $e^x \approx 1+x$, the above formula can be simplified to

$$f_o = \frac{1}{V_{th}C_m} \cdot \sum_i \omega_{i=1}^{n^l} f_i \tag{15}$$

The input frequency f_i is proportional to the output frequency f_o, as shown above. If $1/(V_{th}C_m) = 1$, we can conclude that this function is equivalent to the ReLU, in the case of the same input frequency. No matter how the parameters of the LIF model change, this proportional relation remains true. Given the spiking threshold, the LIF model introduces nonlinearity into the network. Simultaneously, because of frequency coding, the input frequency cannot be less than 0, which increases the sparsity of the network. We conclude that the LIF model can be equivalent to the ReLU under certain parameter mapping principles.

4.5. Definition of model equivalence

In this subsection, we define the equivalence in two aspects, i.e., structural equivalence and behavioral equivalence (shown in Figure 4):

- **Structural equivalence** is mainly reflected in the structures of the ReLU-AN model and LIF model. The parameters of the two models should have a mapping relationship represented by a transformation function R. R can be described as a binary relation satisfying reflexive (xRx), symmetrical xRy $\Rightarrow$ yRx), and transitive properties ((xRy $\wedge$ yRz) $\Rightarrow$ xRz). We will present a perfect parameter mapping between Linear LIF/SNN (model A) and ReLU/ANN (model B) in Section 4.4.
- **Behavioral equivalence** focuses on the functional equivalence of the two models, requiring that model A can

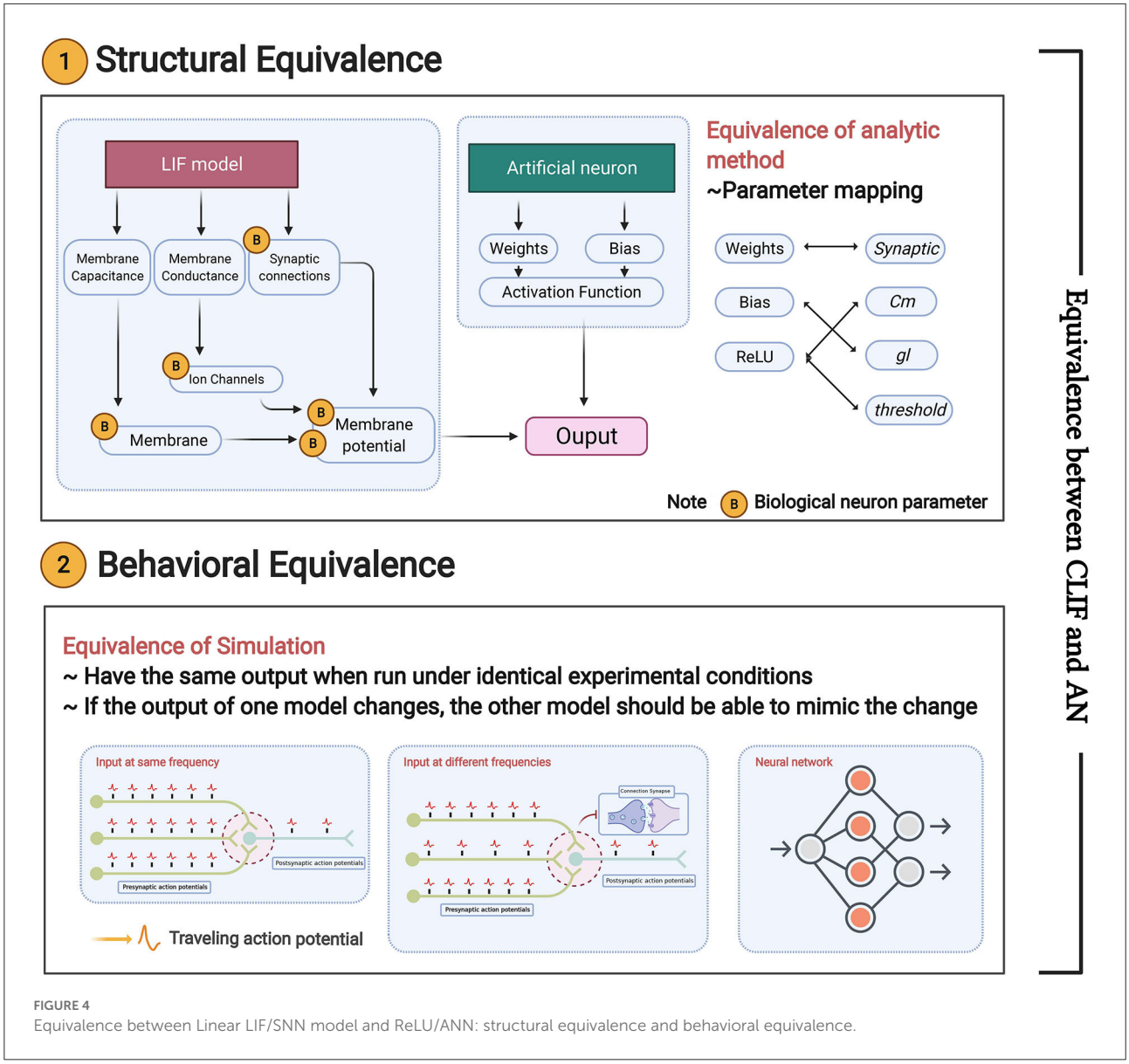

FIGURE 4
Equivalence between Linear LIF/SNN model and ReLU/ANN: structural equivalence and behavioral equivalence.

complete the functions of model B, and vice versa. We define behavioral equivalence as: "Model A and Model B have the same output if run under identical experimental conditions. Given a parameter mapping rule, there always exists a small error bound ε that $\|F_A(x) - F_B(x)\| \leq \varepsilon$ can be guaranteed for any valid input x, where F_A,F_B denotes the function of model A and model B."

5. Experiments and analysis

In this section, we demonstrate the equivalence of LIF/SNN and ReLU-AN/DNN model and the advantages of the Linear LIF model compared to the Reset-to-Zero LIF model. As shown in Figure 5, it mainly includes: 1. Verify the structural and functional equivalence of LIF/SNN and ReLU/DNN through simulation. 2. Reduce the simulation error by increasing the sampling frequency and coding time, demonstrating a convergence toward ideal conditions.

The simulation experiment in this section is mainly divided into two parts:

1. Simulation 1: Proof of structural equivalence
   (a) Compare the Linear LIF model with the ReLU-AN model (with bias) when the input signal frequencies are the same.
2. Simulation 2: Prove of behavioral equivalence
   (a) Compare the Linear LIF model and the ReLU-AN model (with bias) under the condition that the two input signal frequencies are different.

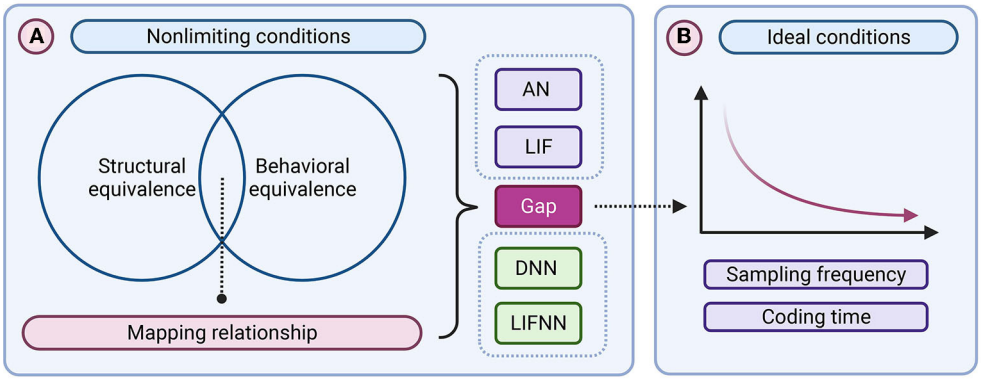

FIGURE 5
The framework of the proof of equivalence between Linear LIF model and ReLU model. **(A)** Nonlimiting conditions. **(B)** Ideal conditions.

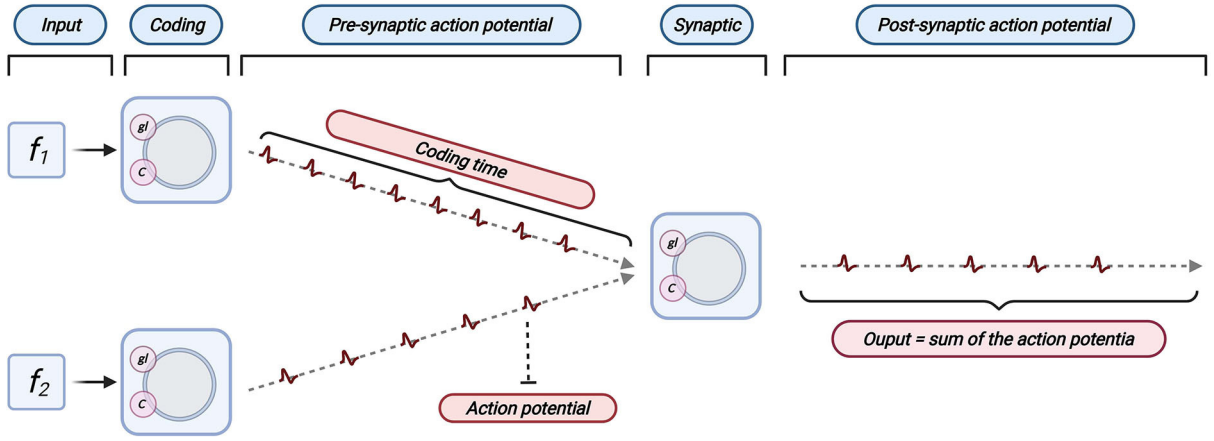

FIGURE 6
The structure of the simulation. The whole structure consists of three neurons, two of which are used for information coding and serve as presynaptic neurons; the third LIF neuron is used to process the input of the first two neurons and produce an output spike train. There are two synaptic structures in this structure.

(b) Compare the Linear LIF model and the ReLU-AN model (with bias) under the condition that the input signal frequencies are different.
(c) Compare LIFNN and DNN (without bias) based on face/motor data set and MNIST and CIFAR10 data set.

5.1. Simulation1: Proof of structural equivalence

In this subsection, we examine the dynamics of the Linear LIF model for verifying the proposed parameter mapping. The simple structure consists of a single LIF neuron and two synaptic, shown in Figure 6.

Firstly, we consider the simplest case, and the LIF model has two input spike trains from the presynaptic neurons with the same frequency. When calculating the output of the Linear LIF model and the Reset-to-Zero LIF model, we transfer the weights and obtain the values of membrane capacitance and membrane conductance according to the mapping relationship, so that the input and output relationship of the Linear LIF model is the same as the ReLU model. The parameters setting is shown in Table 2.

We recorded the output frequency/pixel of ReLU-AN model, Linear LIF model, and Reset-to-Zero LIF model under the same input, shown in Figure 7. Figure 7A shows that the output of Linear LIF model is almost equal to the output of ReLU-AN model, but Reset-to-Zero LIF model has a gap when the input frequency is large. The slope of the relationship

TABLE 2 Parameters setting of simulation 1.

Params	f_{in}	ω	b	gl	C_m	V_{th}
Value	[1, 30] Hz	[0.3, 0.2]	−2.16	3.0	1.0	1.0

between input and output, which is calculated by $1/(C_m V_{th})$, is equal to that of ReLU-AN model. Besides, the minimum input frequency that can excite an action potential is consistent with the bias of ReLU-AN model.

Figure 7B shows the relationship between the membrane conductance gl and the bias. Apparently, the gl is proportional to the minimum input frequency which can excite the action potential and good agreement with the Equation (12). By changing membrane capacitance C_m and fixing other simulation parameters, we can get the relationship between the slope of activation function and membrane capacitance C_m. Figure 7C illustrates the relationship between C_m and input-output slope of Linear LIF model (green) with respect to Reset-to-Zero LIF model (red). As the Equation (15), the slope of LIF model is $1/(V_{th}C_m)$. When we set the spiking threshold $V_{th} = 1mV$, the relationship between C_m and the slope should be an inversely proportional function. We can see that the Linear LIF model is more consistent with Equation (15), while there is an error between the LIF model and the derivation.

Based on this simulation, we verified the mapping relationship between C_m and the slope of input-output curve, gl, and the bias. So that the structural equivalence is proved that the parameters of Linear LIF model and ReLU-AN model can be one-to-one corresponded.

5.2. Simulation2: Proof of behavioral equivalence

In this subsection, we prove that the behavioral equivalence is valid under various conditions. If the difference between the Linear LIF model and the ReLU-AN model is within the error range, we believe that the behavioral equivalence is valid. In this simulation, we get the structural equivalence, that is, the parameter mapping relation is applicable in all the above cases.

5.2.1. Experiments for two input spike trains

We use the same structure and coding algorithm as Section 5.1. We map the parameters of ReLU-AN model to the Linear LIF model according to the parameter mapping relationship. When the input spiking trains have different frequencies, the output signal of Linear LIF model is not a periodic spiking train. So we count the number of action potentials in the output spiking train and divide it by the time windows to get the output frequency. The params of the simulation are given in Table 3.

The output of ReLU-AN model, Linear LIF model, and Reset-to-Zero LIF under the same condition, as shown in Figure 8A. The Linear LIF model's output can fit well with the output of ReLU-AN model, while there is an error between it and Reset-to-Zero LIF model. We perform additional experiments to explore the relationship between gl and bias. Figure 8B illustrates the bias change of the ReLU-AN model (blue) with respect to the minimum frequency of the Linear LIF model. We observe that the minimum frequency fits well with the bias, which indicates the correctness of behavioral equivalence. Figure 8C shows the relationship between C_m and slope of the input-output curve, where the blue line represents the Linear LIF model and the orange line represents the Reset-to-Zero LIF model. According to the mapping relationship, the slope of the input-output curve should be inversely proportional to the membrane capacitance C_m. We can conclude that the Linear LIF model is more consistent with the Equation (15), and the slope of the input-output function can be adjusted to 1 through parameter tuning.

Lastly, we discuss the behavioral equivalence of the Linear LIF model and ReLU-AN model under the condition that the model has three input signals with different frequencies. As shown in Figure 9, the output of Linear LIF is the same as the output of ReLU-AN model within a certain margin of error. In this case, we proved that the behavioral equivalence is valid under this condition.

5.3. Experiments for fully connected architectures

In this section, we will analyze a case of a three-layer neural network. one is a pre-trained ANN based on training dataset, and another is the weights converted linear LIF neural network (LLIFNN) based on the pre-trained ANN. We will analyze the middle-layer output, the classification accuracy of the test dataset, and the influence of parameters on the neural network.

We established an equivalent LLIFNN according to the proposed parameter mapping. We set the parameters of the Linear LIF neurons to fixed values, such as the membrane capacitance C_m and spiking threshold V_{th}, and mapping the weights of trained networks to LLIFNN. Since the bias of nodes in ANN is set to zero, so we also set the membrane capacitance C_m to a fixed value. We use the frequency coding and mark the subscripts of the node with the most action potentials in the output layer as the label. The structure and information processing algorithm are shown in Figure 10.

The structural equivalence and behavioral equivalence constitute the equivalence between networks. We use correlation coefficient (Meng et al., 1992) to quantify the functional

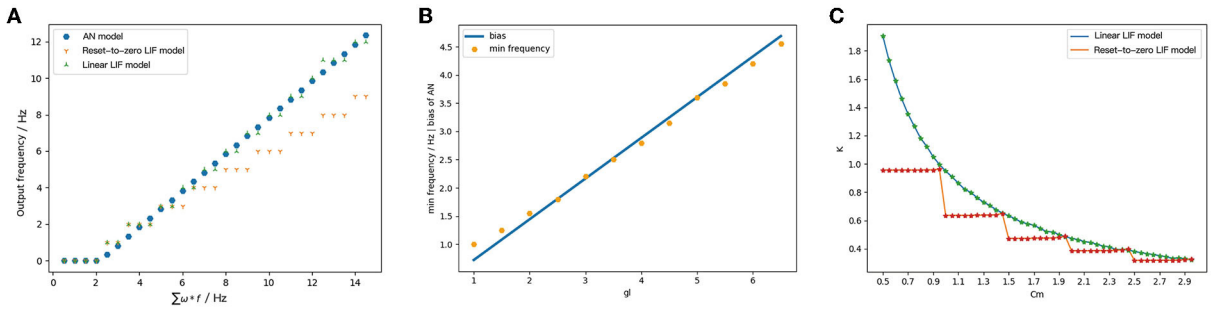

FIGURE 7
(A) The output of ReLU, LIF, and Linear LIF when the input signals have the same frequency. The abscissa is the input frequency and the ordinate is the output frequency. The blue point represents the output of ReLU, the orange represents the output of the Reset-to-Zero LIF model, and the green point represents the data of the Linear LIF model. **(B)** The relationship between min frequency and membrane conductance. **(C)** The relationship between slope of activation function and membrane capacitance.

TABLE 3 Parameters setting of simulation 2.

Params	f_{in}	ω	b	gl	C_m	V_{th}
Value	$[1, 60]$ Hz	$[0.3, 0.2]$	−2.16	3.0	1.0	1.0

equivalence of the network, which is also the most common indicator of similarity. The correlation coefficient is introduced as a metric of the similarity of neural information. It is used in statistics to measure how strong a relationship is between two random variables. The correlation coefficient between x and y is

$$\rho_{x,y} = \frac{n(\sum xy) - (\Sigma x)(\Sigma y)}{\sqrt{[n\Sigma x^2 - (\sum x)^2][n\Sigma y^2 - (\Sigma y)^2]}} \quad (16)$$

5.3.1. Face/motorbike dataset

We firstly evaluate LLIFNN based on the face and motorbike categories of the Caltech 101 dataset, including 400 training pictures and 464 test pictures. We unified the image size to 130*80 pixels. We trained and tested a three-layer fully connected neural network without bias by presenting 20 epoch training set. The data of size 130*80 is stretched into a 10,400-demensional vector and input to the neural network. The 600 neurons in the hidden layer integrate the input data of the input layer and pass them to the classification layer (of 2 neurons) through the ReLU function. The structure of LLIFNN is the same as ANN. Take the output of ANN as the label of test data, classification of LLIFNN based on face-moto dataset can achieve 99.75%. The confusion matrix is shown in Table 4.

5.3.2. MNIST dataset

The MNIST dataset is a large database of handwritten digits that contains ten object categories. It has 60000 training images and 10000 test images. The MNIST dataset is a good benchmark to show and prove the behavioral similarity between LLIFNN and ANN. We use the network with the same structure as the network in Section 5.3.1. We will analyze the behavioral similarity from two aspects: the output of the middle layer and output layer, respectively.

Here we show the output spike trains of the middle layer and mark the change of membrane potential and the action potentials. In Figure 11A, the gray lines represent the change of membrane potential and the red lines represent the action potential. When the membrane potential reaches the spiking threshold, the post-synaptic neuron will excite an action potential.

Firstly, we compare the output of the middle layer based on the correlation coefficient analysis. The framework of calculating the correlation coefficient is shown in Figure 11B. We convert the 60 series (the number of nodes in the middle layer) of spiking trains in the middle layer of LLIFNN into a 60*1 vector, named vector A. The output calculation method of Linear LIF is the same as the method mentioned in the previous chapter. With the middle layer of ANN, a 60*1 vector named vector B. We can calculate the correlation coefficient of the two vectors to quantitatively analyze the correlation between the two outputs.

Figure 12A shows the correlation coefficients for 20 groups of data, where the abscissa is the data table and the ordinate is the correlation coefficient. If the correlation coefficient is set to greater than 0.8, it can be determined that the two variables are strongly related. The orange line is the change in the correlation coefficient of the output signals of two models based on 50 groups of data, and the blue dotted line is the average value of the correlation coefficient. We mark in the image shows that the maximum correlation coefficient is 0.97, the minimum is 0.90, and the average is 0.94. It quantitatively validates the similarity between the outputs of middle layer.

We assume that the behavioral equivalence of neural networks can be reflected in two aspects: 1. For the same data,

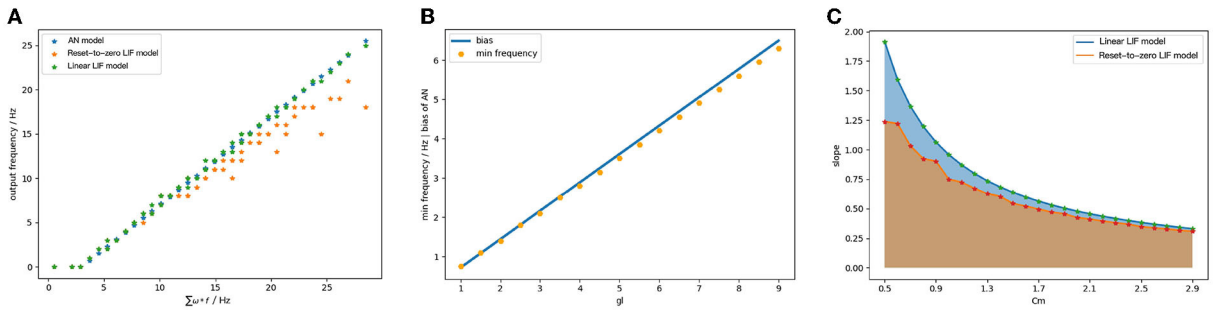

FIGURE 8
(A) The output of ReLU-AN model, Reset-to-Zero LIF model and Linear LIF model when the input signals have different frequencies. **(B)** The relationship between membrane conductance *gl* and the bias. **(C)** The relationship between membrane capacitance C_m and slope of activation function.

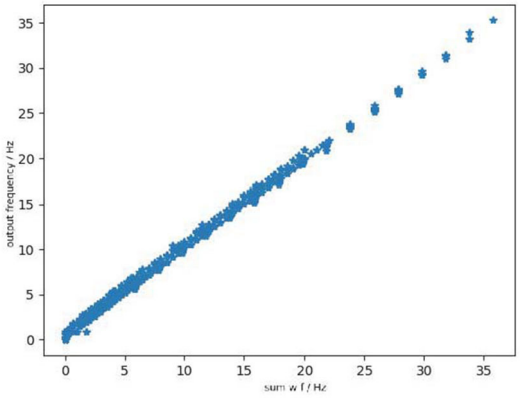

FIGURE 9
The relationship between output of ReLU-AN model and Linear LIF model when the three input signals have differen frequencies.

the outputs of ANN and SNN interneurons are equivalent. 2. The output of the neuron at the same location is equivalent under different data input situations. Figure 12B shows the correlation coefficient matrix in data dimension and neuron dimension, corresponding to the two aspects, respectively. The two correlation coefficient matrices are Symmetric Matrices and can be divided into four quadrants with the center point as the origin of the Darwin coordinate system.

We denote by x the output of the DNN, by y the output of the SNN, and by $\rho(x, y)$ the correlation coefficient. Based on 20 groups of test data, the correlation coefficient matrix between the output of the Linear LIF model and the output of the ReLU model in the middle layer is shown on the left side of Figure 12B. The first quadrant is the correlation coefficient matrix $\rho(x, x)$ of ReLU vs. ReLU among different data, the second quadrant shows the correlation coefficient matrix $\rho(x, y)$ between the output of the LIF model $x_{[600,20]}$ and the output of the ReLU model $y_{[600,20]}$, and the third quadrant is the correlation coefficient matrix $\rho(y, y)$ of LIF vs. LIF among different data. We call this the data-dimension correlation coefficient matrix. In other words, for a single data, the output of LIF is more similar to the output of ReLU under the same input than the output of the ReLU model under different inputs. Next, we analyze the neuron-dimension correlation coefficient. We analyze the correlation coefficient matrix $\rho(x_{i,j}, y_{i,j})$, where i is the subscript of the data and j is the subscript of the neuron, shown in the right part of Figure 12B. Because there are two LIF neuron that did not fire an action potential in all the test data, its variance in 20 sets of data is 0, which will cause an error in the calculation of the correlation coefficient matrix. So we ignore this neuron and only consider 19 neurons. The first quadrant is the correlation coefficient matrix of ReLU vs. ReLU, the second quadrant shows the correlation coefficient matrix of Linear LIF vs. ReLU, and the fourth quadrant is the correlation coefficient matrix of Linear LIF vs. Linear LIF, but all for different neurons. We can see that the features represented by an Linear LIF neuron are equivalent to the features extracted by the corresponding ReLU neuron.

Through the calculation of the correlation coefficient matrix, we have illustrated the behavioral equivalence of the hidden layer. To further illustrate the behavioral equivalence of network, we calculated the classification accuracy of LIF/SNN, using the output label of ANN as the benchmark. Based on the classification accuracy of MNSIT dataset, not only the equivalence of the output layer can be verified, but also the equivalence of the entire network can be proved. We used the subscript of node with the largest number of action potential as the final classification label. A confusion matrix of the SNN network classification results against that of the DNN classification results is depicted in Figure 13A. The overall accuracy reaches 99.38%, indicating that under the proposed parameter mapping, the LIF/SNN can achieve similar classification results as its equivalent ReLU/DNN.

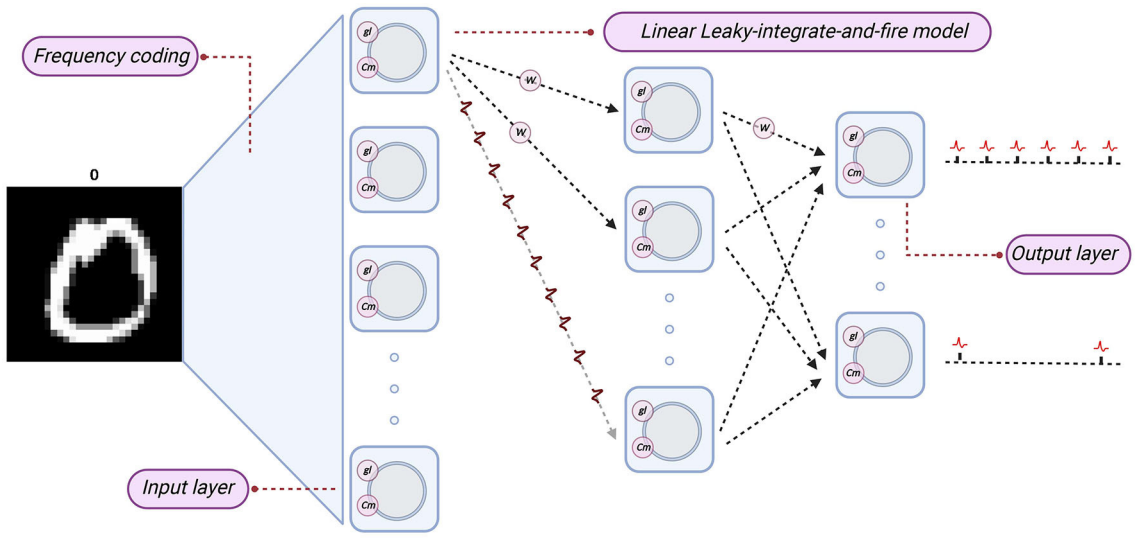

FIGURE 10
The structure of LIF-SNN. The network structure of SNN is the same as that of ANN, and the nodes are replaced by Linear LIF model. The subscript of the node with the largest number of action potentials in the output layer is taken as the output of the network.

TABLE 4 Confusion matrix of face/motorbike dataset.

Confusion matrix		Predict	
		Face	Moto
Real	Face	200	0
	Moto	1	199

The LIF model is widely used in other articles focus on SNN, and the Linear LIF model is modified based on the LIF model. With an understanding of the difference between LIF model and Linear LIF model, we further explore the advantages of Linear LIF model in the network. We change the following two parameters and analyze the gap between the classification accuracy of the two networks MNIST to quantitatively compare the two models:

- The range of coding frequency. We know that the data range of MNIST is $[0, 1]$. Based on the frequency coding algorithm, we can encode the real number into a spike train within the coding time. Since the weight which is transformed from ANN is small (the maximum value is around 0.1), we give a parameter to map the coding range to $[0, k]$.
- Membrane conductance. In the parameter mapping relationship, gl maps to the bias of ReLU, but in this ANN we set the bias to 0. The parameter gl determines the membrane potential attenuation of the LIF and Linear LIF models in the time domain, so we also make it as a variable to compare the LIF model and the Linear LIF model.

Figure 13B shows the trend of network classification accuracy based on the two models with the range of coding frequency. We can see that as the range of coding frequency increases, the recognition accuracy of the two networks is increasing. However, under the same parameters and weights, the recognition accuracy of the LIFNN is 7% higher than that of LIFNN. At the same time, as gl increases, the accuracy of the two networks also increases, as shown in Figure 13C. The theory in Section 4.4 proves that the nonlinearity of Linear LIF model is consistent with ReLU-AN model and better than Reset-to-Zero LIF model. This has also been verified in the simulation, and the similarity between the weight-converted SNN with Linear LIF model and ANN is significantly better than that with Reset-to-Zero LIF model, especially in the part with high input frequency.

However, there are still some remaining problems. For example, according to the parameter mapping relationship, the smaller the membrane conductivity parameter, the higher the equivalence between the Linear LIF model and the ReLU model. For the network structure built by multiple Linear LIF models, the larger the film capacitance parameter, the higher the classification accuracy compared to ANN. We believe that membrane conductance has a complicated relationship with the frequency coding range. We will explore in the follow-up work and believe that the convolutional SNN will eliminate this problem after adding bias.

Through the comparison of middle layers of LIF-SNN and ANN, the comparison of classification accuracy of MNIST data set, and the comparison of the classification effect of Linear LIF-SNN and LIF-SNN, we proved that LIFNN and ANN are behavioral equivalents at the network level, which confirms

the behavioral equivalence of Linear LIF model and the ReLU model.

5.4. Experiments for convolutional architectures

The previous section proved the behavioral equivalence based on fully connected neural networks (FCNNs). However, FCNNs with a large number of parameters require a longer training time and overfit the training dataset. Compared with FCNNs, convolutional neural networks (CNNs) are quite effective for image classification problems and have been applied for various learning problems. The convolutional layer, the main component of CNNs, is different from full-connection layers. To bridge the gap between Deep learning and SNNs, we verified that SNNs with the convolutional structure could also complete the task of CNN, based on the equivalence between the Linear LIF model and ReLU-AN model.

5.4.1. Spiking convolutional layer

Convolutional layers are the major building blocks used in CNNs. Through the convolution operation, the convolutional layer encodes the feature representation of the input at multiple hierarchical levels. Establishing the spiking convolutional layer is very important for building a deep SNN. The connections in convolutional layers and between the layers are similar to those in the CNN architecture. Suppose we perform a convolution operation on the output of the upper layer network, thus the input of convolutional layer can be expressed as a matrix $S^{[l-1]}$ with size $(S_H^{[l-1]}, S_W^{[l-1]}, S_C^{[l-1]}, T)$. The convolutional layer passes a series of filters $\omega^{[l]}$ over our image and gets the feature map. And the filters should have the same number of channels with input $S^{[l-1]}$. The convolutional layer is summed up in Figure 14 and the number of filters is $n_c^{[l]}$. In this simulation, we add padding around the input with zero spike trains in order to maker the output size is the same as the input size (when stride = 1). Linear LIF models receive the spike train, which is the sum of elementwise multiplication of the filter and the subcube of input spike trains. This yields :

$$conv(S,\omega)_{x,y} = \sum_{i=1}^{n^H}\sum_{j=1}^{n^W}\sum_{k=1}^{n^C} S_{x+i-1,x+j-1,k,T}\omega_{i,j,k} \tag{17}$$

where ω is the filter with the size of (n^H, n^W, n^C). We insert the spike train into the membrane (Equation 5) and solve for the output spike train.

5.4.2. Spiking max-pooling layer

Most successful CNNs use max-pooling, typically added to CNNs following individual convolutional layers, to reduce computational load and overfitting. Cao et al. (2015) used the lateral inhibition to select the winner neuron and completed the function of max-pooling layer. However, the winner neuron may not be the neuron with the largest output. Here we propose a simple method to complete the operation of the max-pooling layer. For a set of spike trains let $I_{l,h}$ denote the output of neuron. $l \in \{1, ..., L\}$ and $h \in \{1, ..., H\}$ represent the row and column of neuron, respectively. Considering that we use the frequency encoding, we integrate the input spike sequences in the time domain firstly. The number of spikes N of neuron $n_{l,h}$ is computed as:

$$N_{l,h} = \int_o^T I_{l,h}(t)dt \tag{18}$$

where T is the time window. And the we adopt the neuron with the largest number of spikes in the time window as the output neuron of the max-pooling layer, shown in Figure 15.

5.4.3. Datasets and implementation

We set a group of experiments based on MNIST and CIFAR-10 datasets. We convert the supervised-trained weights of CNN into an SNN with the same structure and verify the gap between CNN and SNN in the test dataset.

A ConvNet with two convolution layers (Conv.12 5 × 5 - Conv.64 5 × 5), ReLU activations, and two max-pooling layers are trained on the MNIST dataset. The structure of networks is the same as architectures used by authors in Diehl et al. (2015) and shown in Table 5. In the experiment, we select the best-performing model after the verification accuracy has converged, and directly transform it into LIF-SNN. The LIF-SNN uses frequency coding and sets the parameters of each Linear LIF neuron to be equivalent to the ReLU-AN model.

Verified by experiment, a shallow convolutional net can achieve high performance on the MNIST dataset. A more complex model should be performed to evaluate the equivalence in a deep structure. We use the AlexNet architecture (Krizhevsky et al., 2012) and VGG-16 (Simonyan and Zisserman, 2015) architecture for the CIFAR-10 dataset. In the simulation, we did not use image pre-processing and augmentation techniques and kept consistent with the AlexNet and VGG-16 model architecture. And all the CNNs in the experiment did not use bias. Because the conversion between bias and membrane conductance needs to limit the weight of the neural network, see Section 4.3 for details. The equivalence between neuron models with bias has been proved in the previous chapter through formulas and simulation experiments, see Section 5.2.

By verifying the similarity between CNN and LIF-SNN, we proved the equivalence of the Linear LIF model with

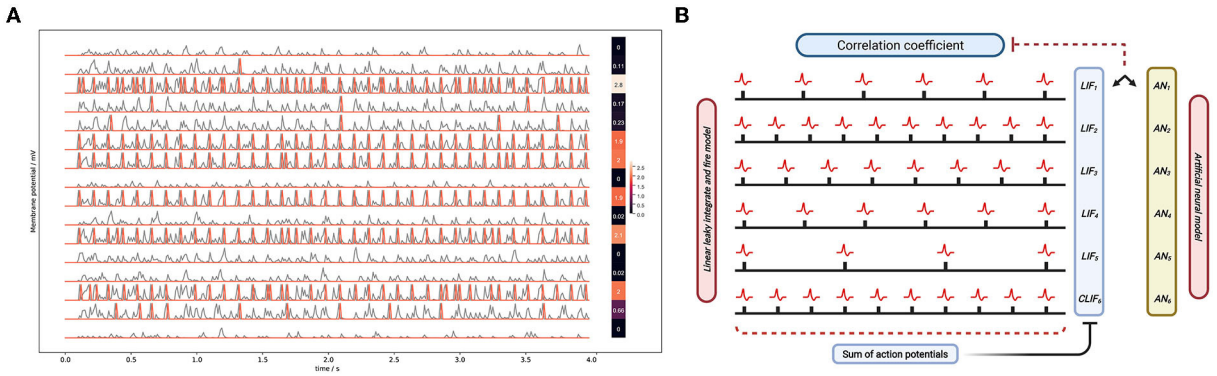

FIGURE 11
(A) The membrane potential of LIF neurons. The left curve shows the state of 16 LIF neurons. The first heat map on the right shows the number of spikes of LIF neurons as arranged in a map. Each data corresponds to the LIF membrane potential data on the left. The other heat map is the same but for the ReLU neurons. **(B)** The framework of calculating the correlation coefficient.

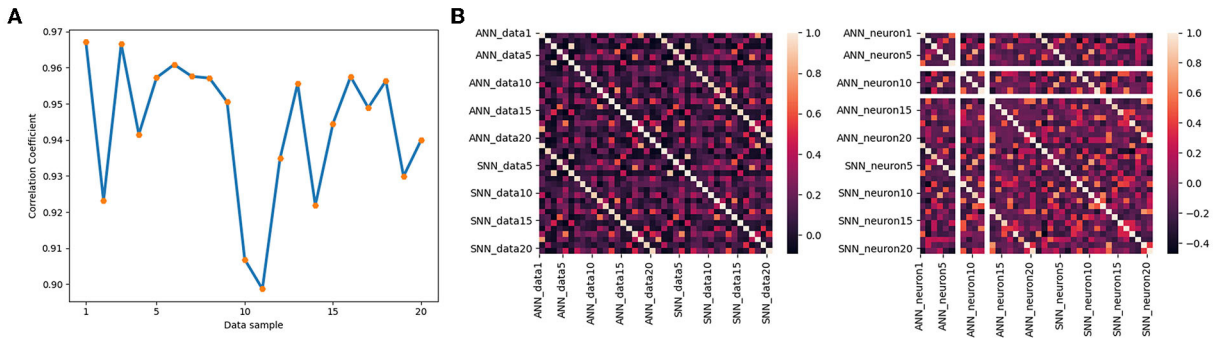

FIGURE 12
(A) Correlation of the output of the two models for different data. **(B)** The correlation coefficient matrix along data dimension (left) and neuron dimension (right).

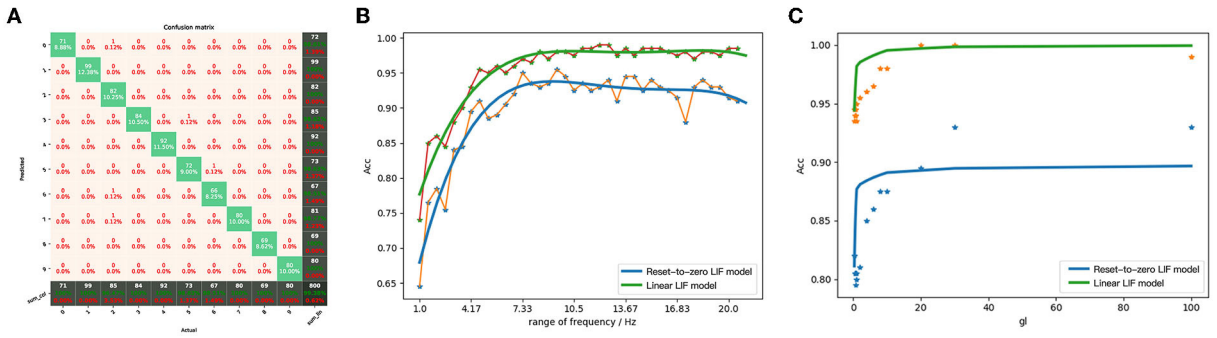

FIGURE 13
(A) The confusion matrix between the output of the network built by LIF/SNN and the predicted value of ReLU/DNN. **(B)** Comparison of classification accuracy of MNIST between LIFNN and LIFNN by changing the range of coding frequency. **(C)** Comparison of classification accuracy of MNIST between LIFNN and LIFNN by changing membrane conductance.

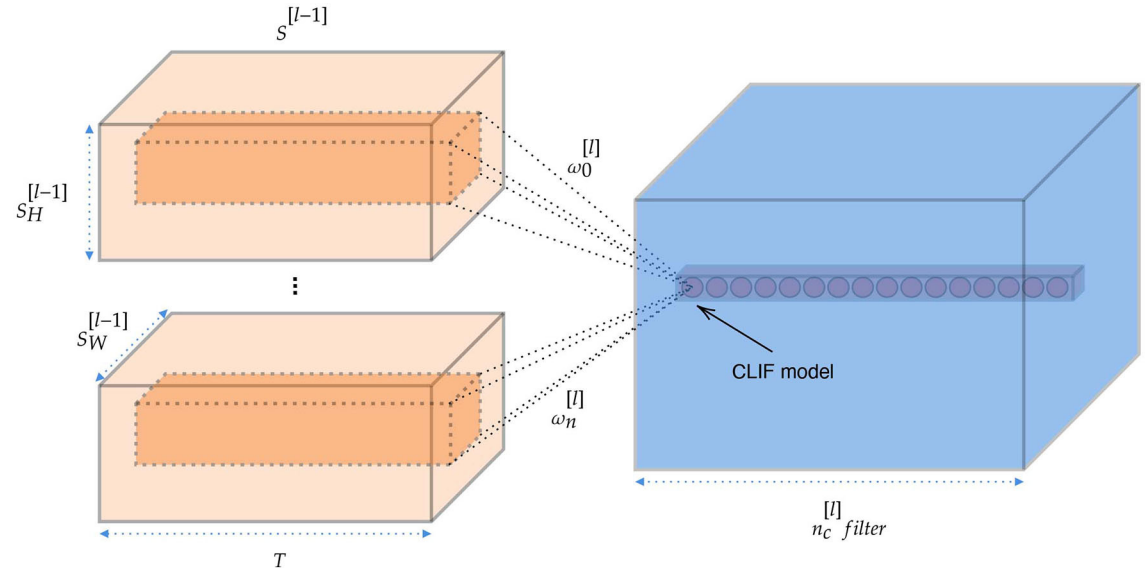

FIGURE 14
The structure and information flow of spiking convolutional layer.

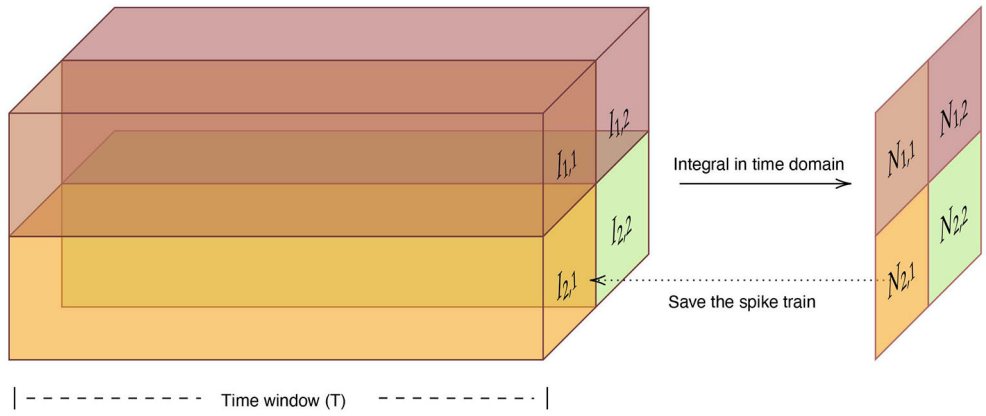

FIGURE 15
The structure and information flow of spiking max-pooling layer.

the ReLU-AN model. However, we are not aiming at the highest performance of CNNs under supervised learning, so we don't use regular operations like image pre-processing, data augmentation, batch normalization, and dropout.

5.4.4. Experiments for ConvNet architectures

The network used for the MNIST dataset is trained for 100 epochs until the validation accuracy stabilizes, and achieves 98.5% test accuracy. For LIF-SNN, we set the time window of simulation as 2s and normalized values of the MNIST images to values between 0 and 10. Based on the algorithm of information coding, spike trains between 0 and 10 *Hz* were generated and presented to the LIF-SNN as inputs. The input trains are processed by convolutional layers and max-pooling layers, and finally are vectorized and fully connected to ten Linear LIF node as the output. We counted the number of spikes in output spike trains, used the node with the highest frequency as the output of LIF-SNN.

Figure 16 shows the comparison between ReLU-based ConvNet and LIF-SNN, and the confusion matrix. We use the number of spikes to represent the spike trains. The comparisons of feature maps between ReLU-based ConvNet and LIF-SNN are shown in the left figure in Figure 16. By comparing the upper

TABLE 5 CNN baseline model for MNIST dataset (with softmax output layer).

Layer	Details
Input layer	$28 \times 28 \times 1$ in [0.0, 1.0]
Convolution 1	$1 \times 5 \times 5$ kernels, ReLU,12 output maps of 28×28
Pooling 1	2×2 max-pooling, 12 output maps of 12×12
Convolution 2	$12 \times 5 \times 5$ kernels, ReLU,64 output maps of 12×12
Pooling 2	2×2 max-pooling, 64 output maps of 6×6
Flatten 1	Flatten, ReLU, 3,136 output maps of 1×1
Fully connected 1	Fully connected, ReLU, 100 output neurons
Fully connected 2	Fully connected, ReLU, 10 output neurons

and lower figures, we can obtain that the original images have undergone convolutional and pooling operations, which are the same as the information represented by spiking convolutional and spiking max-pooling operations after frequency encoding. Ideally, that is, the encoding time is infinite and the sampling frequency is infinite, the image in the bottom row should be the same as the image in the top row. The right figure in Figure 16 shows the confusion matrix of MNIST data, the actual labels are the outputs of CNN, and the predicted labels are the outputs converted LIF-SNN. We selected 2,000 sets of images from the test dataset for testing. Compared with the output of CNN, the accuracy of LIF-SNN reached 100%. Under the structure of the convolutional and pooling layers, the two neuron models can also maintain high behavioral equivalence. The experiments also proved the equivalence of the ReLU-AN model and the CLIF model in the convolutional neural network composed of convolutional and max-pooling layers.

5.4.5. Experiments for deep convolutional architectures

In this subsection, a more thorough evaluation using more complex models (e.g., VGG, AlexNet) and datasets (e.g., CIFAR that includes color images) are given. Since the main contribution of this work is establishing the mapping relationship and not in training a SOTA model. In the training of AlexNet and VGG-16 based on the CIFAR-10 dataset, we did not use data augmentation and any hyper-parameter optimization. Although the classification accuracy based on the existing training mechanism is not the best, it is already competitive.

The AlexNet architectures network with five convolutional layers, ReLU activation, 2×2 max-pooling layers after the 1st, 2nd, and 5th convolutional layer, followed by three fully connected layers was trained on the CIFAR-10 dataset. The AlexNet network is created based on PyTorch and trained on 2 GPUs with a batchsize of 128 for 200 epochs. Classification Cross-Entropy loss and SGD with momentum 0.9 and learning rate 0.001 are used for the loss function and optimizer. We selected the best-performance model and convert the weights to the LIF-SNN with same structure. The best validation accuracies (all the test data) of AlexNet for the CIFAR-10 dataset we achieved were about 80.23%. The simulation process is the same as the simulation of the MNIST dataset. Figure 17 shows the comparison of the feature map and the confusion matrix. Based on the equivalence of Linear LIF model and ReLU-AN model, the outputs of LIF-SNN are infinitely close to the outputs of CNN. Besides, in order to quantitatively analyze the equivalence of LIF-SNN and CNN after weight conversion, we compared the classification accuracy of the two models and drew a confusion matrix. We verified all the test samples and used the output of CNN as the label. LIF-SNN achieved 99.46% accuracy on the CIFAR-10 dataset.

The experiment of the VGG-16 structure network is based on the proposal outlined by the authors in Sengupta et al. (2019). Sengupta made effort to generate an SNN with deep architecture and applied it to the VGG-16 network. Similarly, we trained a VGG-16 network based on the CIFAR-10 dataset. The best validation accuracy we achieved is about 88.58%. We only replace the ReLU-AN model with the Linear LIF model. For 800 images of the test dataset, LIF-SNN obtained a test accuracy rate of 99.88%, and the accuracy rate is calculated in the same way as Section 5.3.2. Besides, with the Spike-Norm proposed by Sengupta et al. (2019), the algorithm allows conversion of nearly arbitrary CNN architectures. The way to combine it with parameter mapping needs to be explored to minimize the accuracy loss in ANN-SNN conversion.

Table 6 summarizes the performance of converted LIF-SNN on MNIST and CIFAR-10 datasets. We list the results of some ANN-to-SNN works and compare them based on the error increment between CNN and SNN as an indicator. Error increment refers to the gap between the classification accuracies of ANN and SNN. At the same time, we also give the network structure and parameters for reference in Table 6. The transformation based on model equivalence achieved the best performance. For the shallow network, we can achieve error-free transformation, and for the deep network, we can minimize the error to 0.08%.

Through the simulation of neural networks with different structures, including shallow and deep networks, we proved the equivalence of the Linear LIF model and the ReLU-AN model. And it is verified that the conversion from CNN to SNN can also be completed in convolutional structures, deep networks, and complex data sets.

5.5. Error analysis

There is still a gap between the LIF/SNN and ReLU/DNN. We believe that the main reason for the error is that the ideal simulation conditions are not achieved. Under ideal conditions, we have infinite encoding time and infinite sampling frequency.

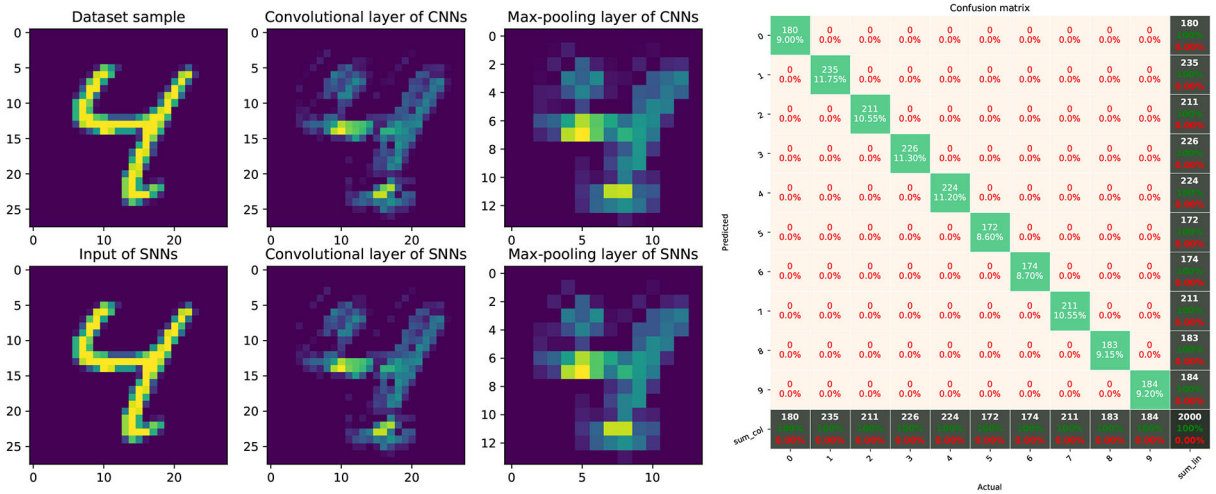

FIGURE 16
Performance of SNN with ConvNet architecture. The left figure shows comparison of feature maps between ReLU-based ConvNet and LIF-SNN. The top figure shows the test image, feature map of convolutional layer, and feature map of max-pooling layer. The images in the bottom rows show the spikes count of output of LIF-SNN. The right figure shows the confusion matrix of LIF-SNN relative to the output of CNN.

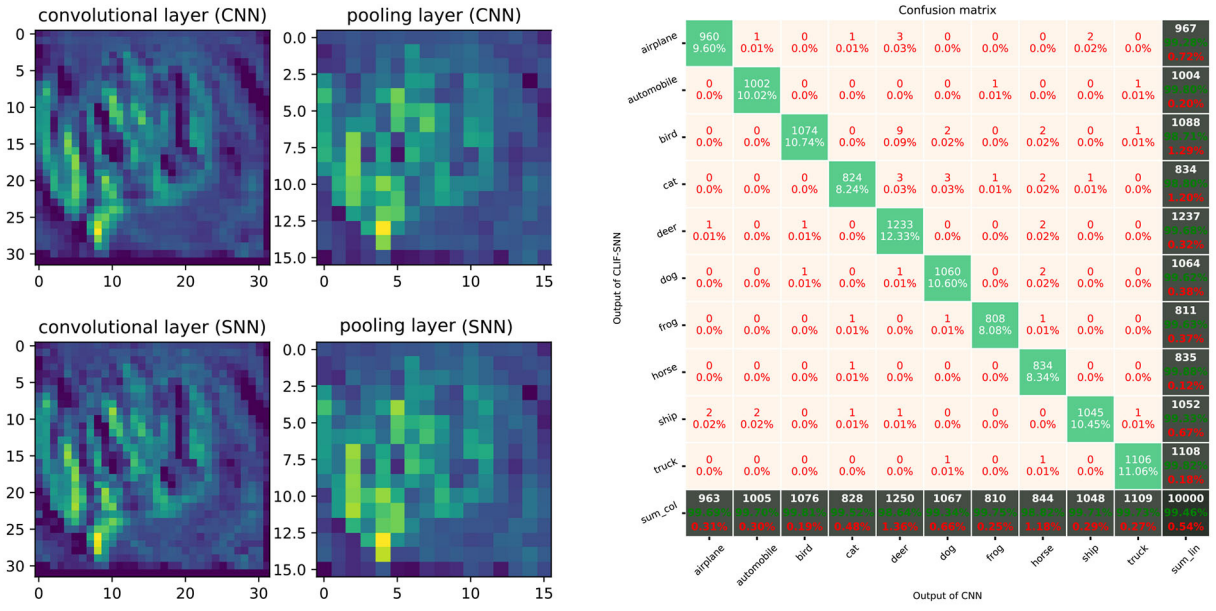

FIGURE 17
Performance of SNN with AlexNet architecture. The left part of figure shows the comparison of feature maps of convolutional layers and max-pooling layer between ReLU-based CNN and LIF-SNN with same input and connection weights. The right part of figure shows the confusion matrix for CIFAR-10 classification.

However, considering the demand for computing power, our simulations are compromised between accuracy and computing power consumption. Besides, the mapping relationship we proposed is established under the condition that multiple inputs with the same frequency. While in more general conditions, there are still errors.

Here we explore the relationship between coding time and error. We define the output of the Linear LIF model as:

$$f' = \frac{N}{T} \tag{19}$$

TABLE 6 Classification error rate on MNIST and CIFAR-10 dataset.

Dataset	Architecture	Preprocess	Synap.	ANN	SNN	Error
MNIST	7-layered ConvNet [ours]	None	0.33M	98.5	98.5	0.0
	7-layered ConvNet (Diehl et al., 2015)	Normalization	0.33M	99.14	99.12	0.02
CIFAR-10	AlexNet [ours]	None	12.98M	80.23	80.12	0.11
	8-layered ConvNet (Cao et al., 2015)	Input data preprocessing	7.4M	79.12	77.43	1.69
	6-layered ConvNet (Rueckauer et al., 2017)	Parameter normalization	23M	91.91	91.85	1.06
	8-layered Network (Hunsberger et al., 2016)	None	-	83.72	83.54	0.18
	VGG-16 [ours]	None	33M	88.58	88.46	0.12
	VGG-16 (Sengupta et al., 2019)	Spiking normalization	-	91.7	91.55	0.15

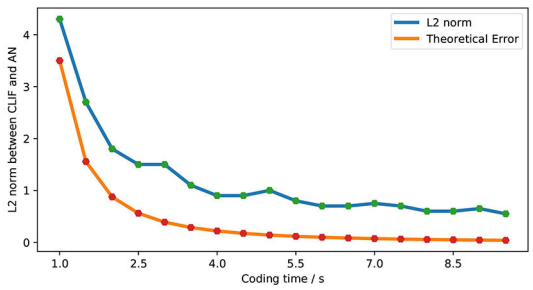

FIGURE 18
Comparison of theoretical error and actual error.

where N is the number of action potentials of spike train within the coding time, and T is the coding time. Then we assume that our expected output frequency is f, then:

$$N = \lceil f \cdot T \rceil \tag{20}$$

Then the error between the expected output frequency and the true output frequency is:

$$\left| f - f' \right| = \left| f - \frac{N}{T} \right| = \left| f - \frac{\lceil f \cdot T \rceil}{T} \right| < \frac{1}{T} \tag{21}$$

We explore the L2 norm as the error under the same frequency input condition. Figure 18 shows the relationship between simulation error and the theoretical error of LIF-AN. We can see that the actual error is consistent with the theoretical error trend, and we can reduce the error by increasing the encoding time. When the coding time is 10s, LIF-SNN achieves an error of less than 1% in the moto/face and MNIST data sets.

6. Conclusion and discussion

6.1. Brief summary

Despite the great successes of DNN in many practical applications, there are still shortcomings to be overcome. One way to overcome them is to look for inspiration from neuroscience, where SNNs have been proposed as a biologically more plausible alternative.

This paper aims to find an equivalence between LIF/SNN and ReLU/DNN. Based on a dynamic analysis of the Linear LIF model, a parameter mapping between the biological neuron model and the artificial neuron model was established. We analyzed the equivalence of the two models from the aspects of weight, bias, and slop of activation function, and verified it both theoretically and experimentally, from a single neuron simulation to a neural network simulation. It shows that such an equivalence can be established, both the structural equivalence and behavioral equivalence, and the Linear LIF model can complete the information integration and the information processing of the linear rectification.

This mapping is helpful for the combination of an SNN with an artificial neural network and increasing the biological interpretability of an artificial neural network. It is the first step toward answering the question of how to design more causal neuron models for future neural networks. Many scholars believe that interpretability is the key to a new artificial intelligence revolution.

At the same time, the equivalence relationship is the bridge between machine intelligence and brain intelligence. Exploring new neuron models is still of great importance in areas such as unsupervised learning. As brain scientists and cognitive neuroscientists unravel the mysteries of the brain, the field of machine learning will surely benefit from it. Modern deep learning takes its inspiration from many areas, and it makes sense to understand the structure of the brain and how it works at an algorithmic level.

6.2. Future opportunities

The architecture of SNN is still limited to the structure of DNN. Compared with DNN, SNN only has the synaptic connection weights which can be trained, while the weights, bias and activation function (dynamic ReLU, Microsoft Chen et al., 2020) can be trained in DNN. Therefore, we expect Linear LIF model and the parameter mapping relationship can bring innovation to SNN from those aspects.

6.2.1. A new way to convert ANN to SNN

With the new approach of converting pre-trained ANN to SNN, we will have a better expression of bias in SNN. Most conversion methods restrict the structure of ANN and directly map the weight. However, bias is also an important parameter in the deep learning network, and we can convert bias into membrane conductance *gl* based on parameter mapping relationship. In this way, SNN and ANN can maintain high consistency and improve the effect of some tasks. Especially in the convolutional neural network, the connectable region of neurons is small, which is more conducive to the conversion of bias into the parameters in the Linear LIF model.

6.2.2. Parameters training of linear LIF model

All the parameters of LIF model can be trained or transformed, which is the fundamental difference from other SNN. Based on the parameters mapping relationship, we can map the trained parameters of DNN to the biological parameters of LIF model, to ensure that each node in SNN has its own unique dynamic properties. At the same time, we know the meaning of each parameter, and we can also carry out the direct training of parameters. In biology, it is also worth investigating whether other parameters of neurons, besides weights, will change.

6.2.3. Dynamic activation function

As the number of layers in the network increases, the number of spikes decreases. We generally adjust the spiking threshold to solve this problem. But we know that the shape of the action potential is essentially fixed, and the spiking threshold of neurons does not change. The membrane capacitance represents the ability to store ions, that is, the opening and closing of ion channels. So, when the number of spikes is low, we can reduce the membrane capacitance and increase the membrane capacitance instead. In parameter mapping, it is similar to dynamic ReLU.

Data availability statement

The original contributions presented in the study are included in the article/Supplementary material, further inquiries can be directed to the corresponding author.

Author contributions

FX conceived the original idea. SL and FX conducted the theoretical derivation and algorithmic development, contributed to the development of the concepts, the analysis of the data, and the writing of the manuscript. Both authors contributed to the article and approved the submitted version.

Funding

This work has been supported by National Natural Science Foundation of China (Grant No. 61991422 to FX).

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

Supplementary material

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fnins.2022.857513/full#supplementary-material

References

Abbott, L. F. (1999). Lapicque's introduction of the integrate-and-fire model neuron (1907). *Brain Res. Bull.* 50, 303–304. doi: 10.1016/S0361-9230(99)00161-6

Abbott, L. F., and Nelson, S. B. (2000). Synaptic plasticity: taming the beast. *Nat. Neurosci.* 3, 1178–1183. doi: 10.1038/81453

Amari, S.-I. (1977). Dynamics of pattern formation in lateral-inhibition type neural fields. *Biol. Cybern.* 27, 77–87. doi: 10.1007/BF00337259

Bear, M. F., Connors, B. W., and Paradiso, M. A. (2007). *Neuroscience - Exploring the Brain*. 3rd Edn. Baltimore, MD: Lippincott Williams & Wilkins.

Burkitt, A. N. (2006). A review of the integrate-and-fire neuron model: Ii. inhomogeneous synaptic input and network properties. *Biol. Cybern.* 95, 97–112. doi: 10.1007/s00422-006-0082-8

Cao, J., Anwer, R. M., Cholakkal, H., Khan, F. S., Pang, Y., and Shao, L. (2020). "Sipmask: spatial information preservation for fast image and video instance segmentation," in *Computer Vision-ECCV 2020. ECCV 2020. Lecture Notes in Computer Science, Vol. 12359*, eds A. Vedaldi, H. Bischof, T. Brox, and J. M. Frahm (Cham: Springer).

Cao, Y., Chen, Y., and Khosla, D. (2015). Spiking deep convolutional neural networks for energy-efficient object recognition. *Int. J. Comput. Vis.* 113, 54–66. doi: 10.1007/s11263-014-0788-3

Caporale, N., and Dan, Y. (2008). Spike timing-dependent plasticity: a hebbian learning rule. *Ann. Rev. Neurosci.* 31, 25–46. doi: 10.1146/annurev.neuro.31.060407.125639

Chen, Y., Dai, X., Liu, M., Chen, D., Yuan, L., and Liu, Z. (2020). "Dynamic relu," in *16th European Conference Computer Vision (ECCV 2020)* (Glasgow: Springer), 351–367. Available online at: https://www.microsoft.com/en-us/research/publication/dynamic-relu/

Choromanska, A., Henaff, M., Mathieu, M., Arous, G. B., and Lecun, Y. (2014). The loss surface of multilayer networks. *Eprint Arxiv*, 192–204. doi: 10.48550/arXiv.1412.0233

Denham, M. J. (2001). "The dynamics of learning and memory: lessons from neuroscience," in *Emergent Neural Computational Architectures Based on Neuroscience: Towards Neuroscience-Inspired Computing*, ed S. Wermter (Berlin; Heidelberg: Springer), 333–347.

Diehl, P. U., and Cook, M. (2015). Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Front. Comput. Neurosci.* 9, 99. doi: 10.3389/fncom.2015.00099

Diehl, P. U., Neil, D., Binas, J., Cook, M., Liu, S.-C., and Pfeiffer, M. (2015). "Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing," in *2015 International Joint Conference on Neural Networks (IJCNN)* (Killarney), 1–8.

Diehl, P. U., Zarrella, G., Cassidy, A., Pedroni, B. U., and Neftci, E. (2016). "Conversion of artificial recurrent neural networks to spiking neural networks for low-power neuromorphic hardware," in *2016 IEEE International Conference on Rebooting Computing (ICRC)* (San Diego, CA: IEEE), 1–8.

Falez, P., Tirilly, P., Bilasco, I. M., Devienne, P., and Boulet, P. (2019). Unsupervised visual feature learning with spike-timing-dependent plasticity: how far are we from traditional feature learning approaches? *Pattern Recognit.* 93, 418–429. doi: 10.1016/j.patcog.2019.04.016

Fukushima, K. (1975). Cognitron: A self-organizing multilayered neural network. *Biol. Cybernet.* 20, 121–136. doi: 10.1007/BF00342633

Gerstner, W., and Kistler, W. M. (2002). *Spiking Neuron Models: Single Neurons, Populations. Plasticity*. Cambridge: Cambridge University Press.

Ghahramani, Z. (2003). "Unsupervised learning," in *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2–14, 2003, Tübingen, Germany, August 4–16, 2003, Revised Lectures*, eds O. Bousquet, U. von Luxburg, and G. Rätsch (Berlin; Heidelberg Springer), 72–112.

Glorot, X., Bordes, A., and Bengio, Y. (2011). "Deep sparse rectifier neural networks," in *Proceedings of the 14th International Conference on Artificial Intelligence and Statisitics (AISTATS) 2011, Vol. 15* (Fort Lauderdale, FL: PMLR), 315–323.

Hahnloser, R. H. R., Seung, H. S., and Slotine, J. J. (2003). Permitted and forbidden sets in symmetric threshold-linear networks. *Neural Comput.* 15, 621–638. doi: 10.1162/089976603321192103

Han, B., Srinivasan, G., and Roy, K. (2020). "Rmp-snn: residual membrane potential neuron for enabling deeper high-accuracy and low-latency spiking neural network," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Seattle, WA: IEEE).

He, K., Zhang, X., Ren, S., and Sun, J. (2016). "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (Las Vegas, NV: IEEE), 770–778.

Hebb, D. O. (1949). *The Organization of Behavior: A Neuropsychological Theory*. New York, NY: J. Wiley; Chapman & Hall.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-,r., Jaitly, N., et al. (2012). Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. *IEEE Signal Process. Mag.* 29, 82–97. doi: 10.1109/MSP.2012.2205597

Hodgkin, A. L., and Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *J. Physiol.* 117, 500–544. doi: 10.1113/jphysiol.1952.sp004764

Hunsberger, E., Eliasmith, C., and Eliasmith, C. (2016). Training spiking deep networks for neuromorphic hardware. *Salon des Refusés* 1, 6566. doi: 10.13140/RG.2.2.10967.06566

Illing, B., Gerstner, W., and Brea, J. (2019). Biologically plausible deep learning – but how far can we go with shallow networks? *Neural Networks* 118, 90–101. doi: 10.1016/j.neunet.2019.06.001

Ivakhnenko, A. (1971). Polynomial theory of complex systems. *IEEE Trans. Syst. Man Cybern. Syst.* 1, 364–378. doi: 10.1109/TSMC.1971.4308320

Ivakhnenko, A., and Lapa, V. (1965). *Cybernetic Predicting Devices*. New York, NY: CCM Information Corporation, 250.

Izhikevich, E. M. (2003). Simple model of spiking neurons. *IEEE Trans Neur. Netw.* 14, 1569–1572. doi: 10.1109/TNN.2003.820440

Jarrett, K., Kavukcuoglu, K., Ranzato, M., and LeCun, Y. (2009). "What is the best multi-stage architecture for object recognition?" in *2009 IEEE 12th International Conference on Computer Vision* (Kyoto: IEEE), 2146–2153.

Jeong, D. S., Kim, K. M., Kim, S., Choi, B. J., and Hwang, C. (2016). Memristors for energy-efficient new computing paradigms. *Adv. Electron. Mater.* 2, 1600090. doi: 10.1002/aelm.201600090

Jiang, X., Pang, Y., Sun, M., and Li, X. (2018). Cascaded subpatch networks for effective cnns. *IEEE Trans. Neural Networks Learn. Syst.* 29, 2684–2694. doi: 10.1109/TNNLS.2017.2689098

Kheradpisheh, S. R., Ganjtabesh, M., Thorpe, S. J., and Masquelier, T. (2018). Stdp-based spiking deep convolutional neural networks for object recognition. *Neural Networks* 99, 56–67. doi: 10.1016/j.neunet.2017.12.005

Kim, S., Park, S., Na, B., and Yoon, S. (2020). "Spiking-yolo: spiking neural network for energy-efficient object detection," in *Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34*, 11270–11277.

Kostal, L., Lansky, P., and Rospars, J.-P. (2007). Neuronal coding and spiking randomness. *Eur. J. Neurosci.* 26, 2693–2701. doi: 10.1111/j.1460-9568.2007.05880.x

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, eds F. Pereira, C. J. Burges, L. Bottou, and K. Q. Weinberger (Lake Tahoe: Curran Associates, Inc), 1097–1105.

Kulkarni, S. R., and Rajendran, B. (2018). Spiking neural networks for handwritten digit recognition–supervised learning and network optimization. *Neural Networks* 103, 118–127. doi: 10.1016/j.neunet.2018.03.019

Lapicque, L. (1907). Recherches quantitatives sur l'excitation electrique des nerfs traitee comme une polarization. *Journal de Physiologie et de Pathologie Generalej.* 9, 620–635.

Maass, W. (1997). Networks of spiking neurons: the third generation of neural network models. *Neural Networks* 10, 1659–1671. doi: 10.1016/S0893-6080(97)00011-7

Maday, Y., Patera, A. T., and Rønquist, E. M. (1990). An operator-integration-factor splitting method for time-dependent problems: application to incompressible fluid flow. *J. Sci. Comput.* 5, 263–292. doi: 10.1007/BF01063118

Marblestone, A. H., Wayne, G., and Kording, K. P. (2016). Toward an integration of deep learning and neuroscience. *Front. Comput. Neurosci.* 10, 94. doi: 10.3389/fncom.2016.00094

McCulloch, W. S., and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophys.* 5, 115–133. doi: 10.1007/BF02478259

Meng, X.-L., Rosenthal, R., and Rubin, D. B. (1992). Comparing correlated correlation coefficients. *Psychol. Bull.* 111, 172. doi: 10.1037/0033-2909.111.1.172

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., et al. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*. doi: 10.48550/arXiv.1312.5602

Mozafari, M., Kheradpisheh, S. R., Masquelier, T., Nowzari-Dalini, A., and Ganjtabesh, M. (2018). First-spike-based visual categorization using reward-modulated stdp. *IEEE Trans. Neural Networks Learn. Syst*. 29, 6178–6190. doi: 10.1109/TNNLS.2018.2826721

Nair, V., and Hinton, G. E. (2010). "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)* (Madison, WI: Omnipress), 807–814.

Nazari, S., and Faez, K. (2018). Spiking pattern recognition using informative signal of image and unsupervised biologically plausible learning. *Neurocomputing* 330, 196–211. doi: 10.1016/j.neucom.2018.10.066

Pinto, N., Cox, D. D., and DiCarlo, J. J. (2008). Why is real-world visual object recognition hard? *PLoS Comput. Biol*. 4, e27. doi: 10.1371/journal.pcbi.0040027

Rathi, N., Srinivasan, G., Panda, P., and Roy, K. (2020). Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation. *International Conference on Learning Representations*. Available online at: https://openreview.net/forum?id=B1xSperKvH

Richmond, B. (2009). "Information coding," in *Encyclopedia of Neuroscience*, ed L. R. Squire (Oxford: Academic Press), 137–144.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*. doi: 10.48550/arXiv.1609.04747

Rueckauer, B., Lungu, I.-A., Hu, Y., Pfeiffer, M., and Liu, S.-C. (2017). Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Front. Neurosci*. 11, 682. doi: 10.3389/fnins.2017.00682

Schmidhuber, J. (2015). Deep learning in neural networks: an overview. *Neural Networks* 61, 85–117. doi: 10.1016/j.neunet.2014.09.003

Sengupta, A., Ye, Y., Wang, R., Liu, C., and Roy, K. (2019). Going deeper in spiking neural networks: VGG and residual architectures. *Front. Neurosci*. 13, 95–95. doi: 10.3389/fnins.2019.00095

Simonyan, K., and Zisserman, A. (2015). "Very deep convolutional networks for large-scale image recognition," in *ICLR 2015: International Conference on Learning Representations 2015* (San Diego, CA).

Song, S., Miller, K. D., and Abbott, L. F. (2000). Competitive hebbian learning through spike-timing-dependent synaptic plasticity. *Nat. Neurosci*. 3, 919–926. doi: 10.1038/78829

Tan, C., Šarlija, M., and Kasabov, N. (2020). Spiking neural networks: background, recent development and the neucube architecture. *Neural Process. Lett*. 52, 1675–1701. doi: 10.1007/s11063-020-10322-8

Tavanaei, A., Ghodrati, M., Kheradpisheh, S. R., Masquelier, T., and Maida, A. (2019). Deep learning in spiking neural networks. *Neur. Netw*. 111, 47–63. doi: 10.1016/j.neunet.2018.12.002

Tavanaei, A., and Maida, A. S. (2016). Bio-inspired spiking convolutional neural network using layer-wise sparse coding and stdp learning. *arXiv preprint arXiv:1611.03000*.

Tavanaei, A., Masquelier, T., and Maida, A. (2018). Representation learning using event-based stdp. *Neural Networks* 105, 294–303. doi: 10.1016/j.neunet.2018.05.018

Tuckwell, H. (1988). *Introduction to Theoretical Neurobiology, Volume 2: Nonlinear and Stochastic Theories*. Cambridge Studies in Mathematical Biology; Cambridge University Press.

Wang, X., Lin, X., and Dang, X. (2020). Supervised learning in spiking neural networks: a review of algorithms and evaluations. *Neural Networks* 125, 258–280. doi: 10.1016/j.neunet.2020.02.011

Zhao, Z., Zheng, P., Xu, S., and Wu, X. (2019). Object detection with deep learning: a review. *IEEE Trans. Neural Netwo. Learn. Syst*. 30, 3212–3232. doi: 10.1109/TNNLS.2018.2876865