LC-TTFS: Towards Lossless Network Conversion for Spiking Neural Networks with TTFS Coding

Qu Yang*, Malu Zhang*, Jibin Wu, Kay Chen Tan, *Fellow, IEEE* and Haizhou Li, *Fellow, IEEE,*

***Abstract*—The biological neurons use precise spike times, in addition to the spike firing rate, to communicate with each other. The time-to-first-spike (TTFS) coding is inspired by such biological observation. However, there is a lack of effective solutions for training TTFS-based spiking neural network (SNN). In this paper, we put forward a simple yet effective network conversion algorithm, which is referred to as LC-TTFS, by addressing two main problems that hinder an effective conversion from a high-performance artificial neural network (ANN) to a TTFS-based SNN. We show that our algorithm can achieve a near-perfect mapping between the activation values of an ANN and the spike times of an SNN on a number of challenging AI tasks, including image classification, image reconstruction, and speech enhancement. With TTFS coding, we can achieve up to orders of magnitude saving in computation over ANN and other rate-based SNNs. The study, therefore, paves the way for deploying ultra-low-power TTFS-based SNNs on power-constrained edge computing platforms.**

***Index Terms*—Deep Spiking Neural Network, Time-to-first-spike Coding, ANN-to-SNN Conversion, Image Classification, Image Reconstruction, Speech Enhancement**

I. Introduction

OVER the last decade, we have witnessed tremendous progress in artificial intelligence technologies, that include computer vision [1]–[4], speech processing [5], [6], natural language processing [7], [8], and robotics [9], [10]. However, the core computational model behind this revolution, i.e., artificial neural network (ANN), is computationally expensive to operate. This prompts the researchers to improve the computational efficiency of ANNs, for example, through model compression [11], [12], network accelerator [13], and reduction of on-chip data movements [14]. Nevertheless, the high computational cost remains a major roadblock to the deployment of ANNs on power-constrained platforms, such as wearable and mobile devices [15].

The human brains evolve over many millennia under strong ecological pressure to be highly efficient and effective, therefore, it is worthwhile to look into the computation principles adopted by biological neural networks. Motivated by this, the spiking neural networks (SNNs), which were initially introduced to simulate neural computations [16], are now considered a power-efficient alternative to the mainstream ANNs, with a great potential to become the solution for power-constrained platforms.

SNNs, which emulate the information processing mechanism of biological neural networks [17]–[21], represent and transmit information through asynchronous action potentials or spikes. Due to the complex spatial-temporal dependency of spike trains and the discontinuity at the spike generation time, the canonical back-propagation (BP) algorithm is not directly applicable to the training of SNNs [22], [23]. The surrogate gradient learning method [24] has been introduced recently to address these problems. It models the spiking neuron as a self-recurrent neural network that explicitly captures the spatial-temporal dependency between input and output spike trains. Furthermore, the continuous surrogate gradient functions are introduced during gradient back-propagation that effectively addresses the discontinuity issue at the spike generation time. Despite much progress on a host of machine learning and neuromorphic benchmarks [25]–[31], it is computationally prohibitive to exactly model the temporal dynamics of SNNs due to exorbitant memory requirement, even for a network of less than ten layers [27]. Besides, this method suffers from the vanishing and exploding gradient problem that is notorious for recurrent neural networks, making long-range temporal credit assignments ineffective. A more biologically plausible way entails considering propagating spikes only when the neuron fires spikes, which reduces the overall number of gradient propagations on neuromorphic hardware [32]. Zhu et al. [32] have rigorously proved that event-based propagation allocates gradients from one output spike to its corresponding input spikes, thus preserving the sum of gradients between layers. Armed with this insight, they successfully trained SNNs with temporal gradients on the CIFAR-100 dataset for the first time.

In another vein of research, ANN-to-SNN conversion methods are introduced as an alternative solution to address the difficulties in direct SNN training. A large body of these network conversion methods takes the firing rate of spiking neurons to approximate the activation value of analog neurons

This research work was supported in part by the IAF, A*STAR, SOITEC, NXP and National University of Singapore under FD-fAbrICS: Joint Lab for FD-SOI Always-on Intelligent Connected Systems (Award I2001E0053), the Agency for Science, Technology and Research (A*STAR) under its AME Programmatic Funding Scheme (Project No. A18A2b0046), A*STAR under its RIE 2020 Advanced Manufacturing and Engineering Human (AME) Programmatic Grant (Grant No. A1687b0033), and by the Research Grants Council of the Hong Kong SAR (Grant No. PolyU11211521, PolyU15218622, and PolyU25216423), and in part by The Hong Kong Polytechnic University (Project IDs: P0039734, P0035379, P0043563, and P0046094), in part by the National Natural Science Foundation of China (NSFC) under Grant U21A20512, 62106038 and 62306259, and in part by the Sichuan Science and Technology Program under Grant 2023YFG0259. (* Contributed equally in this work, Corresponding Author: J. Wu).

Q. Yang is with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore.

H. Li is with Shenzhen Research Institute of Big Data, School of Data Science, The Chinese University of Hong Kong, Shenzhen (CUHK-Shenzhen), China.

M. Zhang is with University of Electronic Science and Technology of China, China.

J. Wu and K. C. Tan are with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong SAR, China.

used in the ANN, which we refer to as *rate conversion* in this paper. By carefully determining the firing threshold of spiking neurons or the weight normalization factor, the pre-trained network parameters of ANN can be directly transferred to the SNN with an equivalent network structure [33]–[47]. In this way, we can avoid the expensive spatio-temporal credit assignments in surrogate gradient learning. The rate conversion methods map ANN to SNN with high precision on major machine learning benchmarks, such as ImageNet-12 [43], [48]–[51], with a high latency [42] due to the requirement of a large simulation time window. The benefits that rate-based SNNs bring are hence limited.

It has long been identified in the biological neural systems that the precise spike firing time, in addition to the spike firing rate, carries a significant amount of information [52]. Based on these insights, the time-to-first-spike (TTFS) coding scheme was formulated [53]–[56], where only one single spike is allowed within each time window as shown in Fig. 1 (a), and a stronger stimulus is transduced into an earlier firing spike. In this way, with a fewer number of spikes, the TTFS coding scheme is expected to be more efficient computationally than rate-based coding.

Embracing the TTFS coding, Rueckauer et al. [42] proposed an ANN-to-SNN conversion algorithm, where the activation value of ANN was treated as the instantaneous firing rate, and subsequently converted to the equivalent spike time in SNN. However, the scalability of this conversion algorithm to deeper network architecture was not demonstrated. The TDSNN [57] algorithm has been proposed to achieve better scalability, which introduces an auxiliary ticking neuron for each operating spiking neuron and a TTFS-like reverse coding scheme. However, the additional spikes generated from auxiliary ticking neurons adversely affect the model efficiency. In contrast, our method uses the same network architecture as the ANN, which does not require any additional neurons or spikes. Recently, the T2FSNN [58] algorithm has been introduced with improved conversion performance on TTFS-based SNNs, while their results still lag behind rate-based SNNs. Moreover, their kernel-based dynamic threshold is considered more computationally expensive on hardware implementation than ours layer-wise dynamic thresholds. Similar to the TDSNN work, Han and Roy [59] proposed a Temporal-Switch-Coding (TSC) scheme and a corresponding TSC spiking neuron model. However, this coding scheme requires two spikes to encode the information of each analog neuron, whereby the information is represented as the time difference between these two spikes. Besides, the introduced TSC neuron model is computationally more expensive than the Integrate-and-Fire neuron model adopted in other works. In what follows, in contrast to the rate conversion introduced earlier, we refer to the ANN-to-SNN conversion based on the TTFS coding as *TTFS conversion*.

In this work, we aim to bridge the accuracy gap between TTFS conversion and rate conversion, thereby fully realizing the computational advantages of SNNs on power-constrained platforms. Toward this goal, we make the following three contributions:

1) We perform a comprehensive analysis on the problems underlying TTFS conversion, including temporal dynamics problem and time-warping problem.
2) We propose a simple yet effective TTFS conversion algorithm that can effectively address all the above problems. As shown in Fig. 1 (c), it establishes a near-perfect mapping between activation values of ANNs and spike times of SNNs, which leads to a near-lossless TTFS conversion.
3) We successfully implement the TTFS conversion algorithm for image classification, image reconstruction, and speech enhancement tasks. To the best of our knowledge, this is the first work that applies TTFS-based SNN in solving challenging signal reconstruction tasks.

The rest of this paper is organized as follows: In Section II, we introduce the research problems to set the stage for our study. In Section III, we present the proposed TTFS conversion algorithm to address the identified problems. In Section IV, we first evaluate the proposed conversion algorithm on the image classification task. Then, we thoroughly evaluate the effectiveness of the proposed algorithm through a series of ablation studies. In Section V, we further evaluate the proposed algorithm on two signal reconstruction tasks, i.e. image reconstruction and speech enhancement. Finally, Section VI concludes the study.

II. Preliminaries and Problem Analysis

We begin by introducing the analog and spiking neuron models, as well as the TTFS coding scheme. We then analyze the problems underlying TTFS conversion.

A. Preliminaries

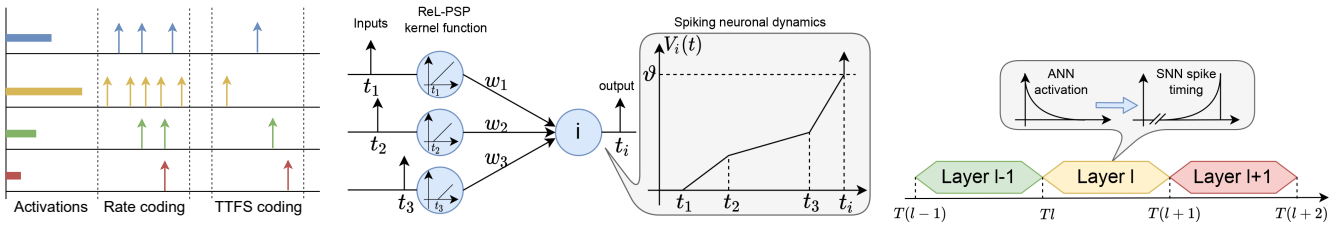
1) Analog neuron model: In ANN-to-SNN conversion, an ANN is first trained, wherein analog neurons are employed and formulated as follows,

$$a_i^l = f(\sum_i^N w_{ij}^l a_j^{l-1} + b_i^l) \tag{1}$$

where a_j^{l-1} and a_i^l are the input and output of neuron i at layer l, respectively. w_{ij}^l is the synaptic weight between pre-synaptic neuron j and post-synaptic neuron i, and b_i^l is the bias term of neuron i. $f(\cdot)$ denotes the activation function, for which we use a modified version of the Rectified Linear Unit (ReLU) function. Specifically, we clamp the activation value to be within $[0, 1]$, similar to the ReLU6 function [60], and we refer to it as ReLU1 hereafter. As will be explained in the following section, this ensures a one-to-one correspondence can be established between ANN and SNN for lossless TTFS conversion.

2) Spiking neuron model: For SNN, to be converted from the pre-trained ANN, we employ the Rectified Linear Postsynaptic Potential (ReL-PSP) spiking neuron model proposed in [61], whose membrane potential $V_i^l(t)$ can be expressed as

$$V_i^l(t) = \sum_{j=1}^{N^{l-1}} w_{ij}^l K(t - t_j^{l-1}) \tag{2}$$



(a) (b) (c)

Fig. 1: (a) Comparison of rate and TTFS coding schemes. (b) Illustration of spiking neuronal dynamics with ReL-PSP kernel function. (c) Illustration of inference pipeline of the TTFS converted SNN, wherein each layer operates in consecutive but non-overlapping time windows. The inset on top shows the data distribution of activation values in an ANN and spike times of the converted SNN. The activation values are mapped to spike times in a one-to-one correspondence following the TTFS encoding scheme.

where $K(\cdot)$ refers to the PSP kernel function, which is defined as

$$K(t-t_j^{l-1}) = \begin{cases} t-t_j^{l-1} & \text{if} \quad t > t_j^{l-1} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

For $t > t_j^{l-1}$, Eq. (2) can be further simplified as

$$V_i^l(t) = \sum_{j=1}^{N^{l-1}} w_{ij}^l (t - t_j^{l-1}) \tag{4}$$

The neuron i fires a spike once its membrane potential exceeds the firing threshold ϑ, whose spike time t_i^l is defined as

$$t_i^l = \mathcal{F}\left\{t | V_i^l(t) = \vartheta, t \geq 0\right\} \tag{5}$$

The neuronal dynamics of the ReL-PSP spiking neuron model are illustrated in Fig. 1 (b). Without loss of generality, we set the firing threshold ϑ to 1 in this work.

3) TTFS encoding scheme: To encode the first spike time of a ReL-PSP spiking neuron with the activation value of a ReLU1 neuron, we follow the TTFS encoding scheme. For each layer that with separate time window, as shown in Fig. 1 (a) and (c), we encode the activation value a_i^l into the spike time t_i^l as per

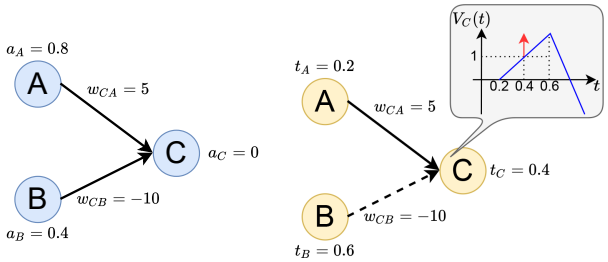
$$\frac{t_i^l - t_{min}^l}{t_{max}^l - t_{min}^l} = 1 - a_i^l \tag{6}$$

where t_{max}^l and t_{min}^l are the maximum and minimum permissible spike time of SNN layer l. We define the time window $T^l = t_{max}^l - t_{min}^l$. Without loss of generalizability, we fix this value to be 1 for all layers, i.e., $T = 1$. Hence, we can establish the following encoding function

$$t_i^l = l + 1 - a_{i,n}^l, \tag{7}$$

B. TTFS Conversion Problems

1) Temporal dynamics problem: Following the above TTFS encoding scheme, a larger activation value in the ANN layer shall be encoded into an earlier spike in the corresponding SNN layer, and vice versa. The additional temporal dynamics introduced during the conversion process may, however, give rise to missing spikes and premature spikes. The missing spike problem happens, during ANN-to-SNN conversion, when an analog neuron receives multiple tiny weighted inputs, while their corresponding spikes as a whole are insufficient to trigger an output spike within the given time window. In contrast, the premature spike problem, which has been also described in [42], happens when the positive weighted spikes arrive earlier than the negative ones, causing the actual number of output spikes more than expected.

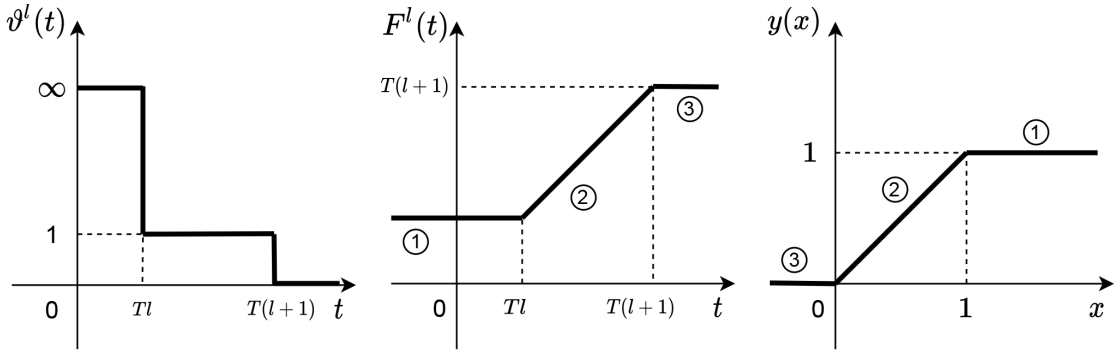


(a) ANN (b) SNN

Fig. 2: Illustration of the premature spike problem. (a) For ANN, the net inputs from neurons A and B are summed to 0, causing neuron C to remain inactivated. (b) For SNN, after TTFS conversion, the earlier spike from neuron A will cause neuron C to fire a 'premature' spike at $t_c = 0.4$, before the arrival of the inhibition input from neuron B.

To better understand the premature spike problem, let's consider a network formed by two input neurons A and B that is connected to an output neuron C, with synaptic weights $w_{CA} = 5$ and $w_{CB} = -10$. As shown in Fig. 2(a), assuming the activation value of analog neurons A and B is $a_A = 0.8$ and $a_B = 0.4$. As these two weighted inputs cancel out each other, the net input received by the neuron C will be 0. According to Eq. (6), the converted spike time of spiking neurons A and B are $t_A = 0.2$ and $t_B = 0.6$. Assuming the firing threshold is 1, the earlier input spike from neuron A will trigger neuron C to fire a spike at $t_C = 0.4$, before receiving the inhibition input from neuron B. In this example, both neurons A and B contribute to neuron C in the ANN, while the contribution of neuron B has been discarded in the SNN due to the additional temporal dynamics introduced after the TTFS conversion.

We refer to both missing spike and premature spike prob-



(a) (b) (c)

Fig. 3: (a) Illustration of the dynamic firing threshold. (b) Illustration of the effective spike time transformation achieved by the dynamic firing threshold. The x- and y-axis represent the spike time before and after applying the dynamic firing threshold, respectively. (c) Illustration of the ReLU1 activation function used in the ANN. Note that the activation values of analog neurons in (c) can be mapped to spike times in (b) in a one-to-one correspondence, while the order of the mapping is reversed following the TTFS encoding scheme.

lems as the temporal dynamics problem, which will compound across network layers and may eventually lead to a fatal mismatch between the outputs of ANN and SNN. To eliminate this problem, we propose a dynamic firing threshold mechanism to ensure spiking neurons in the l-th layer only fire within the permissible time window $[lT, (l+1)T)$, after all their input spikes are received. This ensures the contributions from pre-synaptic spiking neurons are fully considered by the post-synaptic spiking neuron. The details of this mechanism will be explained in Section III-A.

2) Time-warping problem: As we already introduced in Section II-A2, the spiking neuron will fire a spike when the membrane potential $V_i^l(t)$ reaches the firing threshold ϑ. According to Eq. (5), the spike time t_i^l can be calculated as

$$t_i^l = \frac{\vartheta + \sum_{j=0}^{N^{l-1}} w_{ij}^l t_j^{l-1}}{\sum_{j=0}^{N^{l-1}} w_{ij}^l} \tag{8}$$

Following Eq. (6), the activation value a_j^{l-1} of the analog neuron should be encoded into spike time t_j^{l-1} as per

$$t_j^{l-1} = l - a_j^{l-1} \tag{9}$$

Taking Eq. (9) into Eq. (8), we can establish the following relationship between the activation value a_j^{l-1} and the spike time t_i^l

$$t_i^l = \frac{\vartheta + l \times \sum_{j=0}^{N^{l-1}} w_{ij}^l - \sum_{j=0}^{N^{l-1}} w_{ij}^l a_j^{l-1}}{\sum_{j=0}^{N^{l-1}} w_{ij}^l} \tag{10}$$

According to Eq. (10), the spike time t_i^l depends on both weighted inputs $\sum_{j=0}^{N^{l-1}} w_{ij}^l a_j^{l-1}$ and weight sum $\sum_{j=0}^{N^{l-1}} w_{ij}^l$. When transferring an activation value a_j^{l-1} from an analog neuron to a spiking neuron, this additional dependency on the weight sum will cause a significant mismatch between the ANN and SNN outputs if not properly addressed. Ideally, identical inputs should produce consistent outputs in t_i^l. Yet, variations in the weight sum for each neuron i cause deviations in the output for the same input value. We refer to this problem as the *time-warping problem*. As will be introduced in Section III-B, we propose a weight regularization strategy to ensure the weight sum equals 1, thereby eliminating this time-warping problem.

III. LC-TTFS CONVERSION ALGORITHM

A. Solve temporal dynamics problem with dynamic threshold

As discussed in Section II-B1, the temporal dynamics problem will result in a mismatch between the ANN and SNN outputs. To address this problem, we propose a dynamic firing threshold for spiking neurons. As shown in Fig. 3(a), the dynamic firing threshold $\vartheta^l(t)$ for neurons in the l-th layer is determined according to the following piecewise linear function

$$\vartheta^l(t) = \begin{cases} \infty & \text{if} \quad t < Tl \\ 1 & \text{if} \quad t \in [Tl, T(l+1)) \\ -\infty & \text{if} \quad t \geq T(l+1) \end{cases} \tag{11}$$

The role of the proposed dynamic firing threshold is to set the spike time outside the permissible time window to the two boundary values. This is equivalent to applying a transformation function $F^l(t)$, defined as in Eq. (12) and illustrated in Fig. 3(b). Following this dynamic firing threshold, the earliest spike time for neurons in the l-th layer is Tl, and the latest spike should fire before $T(l+1)$. As such, the time window is non-overlapping for each layer, and spiking neurons at layer l only start to fire after all input spikes from layer $l-1$ are being integrated, therefore, overcoming the temporal dynamics problem.

$$F^l(t) = \begin{cases} Tl & \text{if} \quad t < Tl \\ t & \text{if} \quad t \in [Tl, T(l+1)) \\ T(l+1) & \text{if} \quad t \geq T(l+1) \end{cases} \tag{12}$$

It is worth noting that there are two main differences between our proposed dynamic threshold and the method proposed in [42]. Concretely, 1) the dynamic threshold introduced

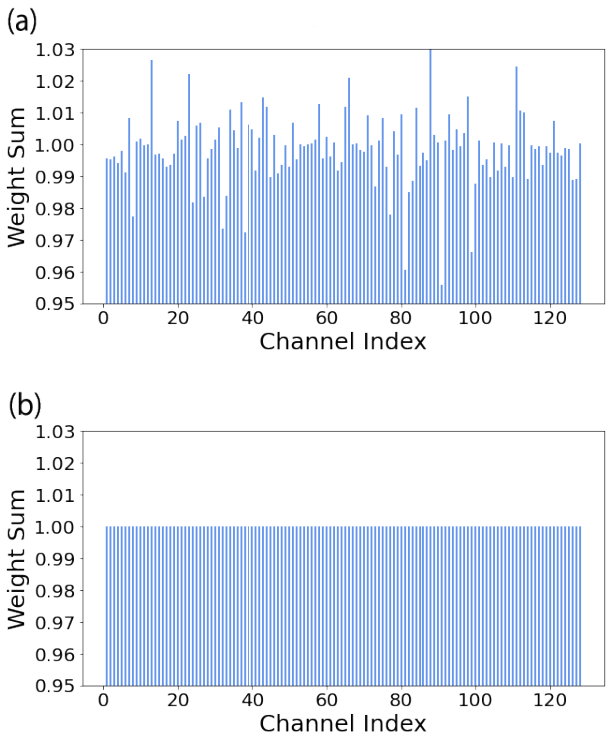


Fig. 4: Illustration of weight sum distributions of a randomly selected network layer, trained on the CIFAR-10 dataset, after imposing the (a) soft constraint, and (b) both soft and hard constraints.

in [42] depends on the weight of each neuron, resulting in the firing threshold varying from neuron to neuron that will cause significant hardware overhead. In contrast, our method shares one firing threshold for all neurons in the same layer. 2). our method does not require calculating the missing spikes which is computationally expensive.

The dynamic firing threshold ensures that the spiking neurons fire only within their permissible time window. To achieve a lossless conversion from the ANN, the activation value of each ANN layer should also be bounded within a particular interval. To this end, we employ the ReLU1 activation function, which is formulated as follows

$$y(x) = \begin{cases} 0 & \text{if} \quad x \leq 0 \\ x & \text{if} \quad x \in (0, 1] \\ 1 & \text{if} \quad x > 1 \end{cases} \tag{13}$$

As shown in Figs. 3(b) and 3(c), with the proposed ReLU1 function, the ANN activation region is one-to-one mapped to the spike time region. The ablation study performed in Section IV-D5 highlights the necessity of this dynamic threshold mechanism toward a lossless TTFS conversion.

B. Solve time-warping problem with weight regularization

To deal with the time-warping problem introduced in Section II-B2, we proposed a two-step weight regularization strategy to ensure the weight sum in Eq. (10) will take a constant value of 1. In the first step, we impose a soft constraint by penalizing those weight sums that are not equal to 1 with an L1 loss function $\mathcal{L}_W$ that is defined as follows

$$\mathcal{L}_W = \sum_{l=0}^{L-1} \sum_{i=0}^{N^l} \left| \sum_{j=0}^{N^{l-1}} w_{ij}^l - 1 \right| \tag{14}$$

As shown in Fig. 4(a), this step drives the overall weight sum distribution towards 1, which can largely alleviate the time-warping problem. However, these seemingly small deviations from the ideal value of 1 will compound across layers and significantly deteriorate the performance of converted SNNs.

To fully resolve the time-warping problem, we further impose a hard constraint by distributing the weight sum deviation evenly across all contributing synapses for each neuron. As shown in Fig. 4(b), by introducing soft and hard constraints during ANN pre-training, we ensure all weight sums are exactly equal to 1. As the soft constraint already drives the weight sum to approach the ideal value, the additional hard constraint has little interference to the learning dynamics and network convergence. This has been confirmed by the ablation study that will be introduced in Section IV-D.

With the time-warping problem resolved, we now can establish the relationship between t_i^l and a_i^l. Considering weight regularization, we obtain

$$t_i^l = 1 + l - \sum_{j=0}^{N^{l-1}} w_{ij}^l a_j^{l-1}, \tag{15}$$

here, $\sum_{j=0}^{N^{l-1}} w_{ij}^l = 1$ and $\vartheta = 1$. With the Eqns. (12, 13) and visualization in Fig.3 (b,c), we yield

$$F^l(x) = T(l+1) - y(T(l+1) - x). \tag{16}$$

As mentioned in Section II-A3, we adopt $T = 1$ in this work, thereby obtaining

$$F^l(x) = l + 1 - y(l + 1 - x). \tag{17}$$

Substitute Eqn. (15) into Eqn. (17), we have

$$\begin{aligned} F^l(t_i^l) &= l + 1 - y(l + 1 - t_i^l) \\ &= l + 1 - y(l + 1 - 1 - l + \sum_{j=0}^{N^{l-1}} w_{ij}^l a_j^{l-1}) \\ &= l + 1 - a_i^l \end{aligned} \tag{18}$$

Since $F^l(t_i^l) = t_i^l$ holds true within the permissible time window, we ultimately obtain

$$t_i^l = l + 1 - a_i^l, \tag{19}$$

This result is consistent with the encoding scheme for neurons in layer $l-1$, as depicted by Eqn. (9). However, it introduces a T steps shift to have a non-overlapping window.

C. Pre-activation normalization strategy

The BN is an important technique to accelerate the deep neural network training, which normalizes the pre-activation of each layer to follow normal distribution so as to mitigate the internal covariate shift problem. We indeed concur that

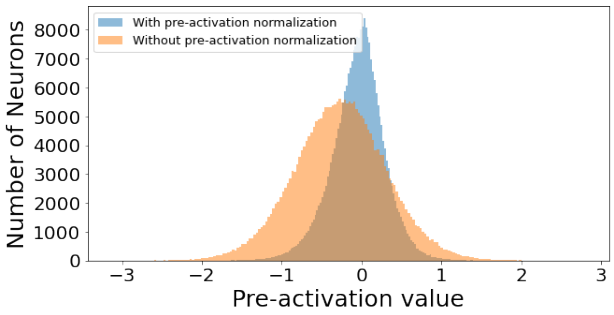


Fig. 5: Comparison of the pre-activation value distributions with (blue) and without (brown) applying the pre-activation normalization technique proposed in this work. Note that the data distribution approaches a normal distribution of N(0,1/9) after applying the pre-activation normalization. The data is extracted from a randomly selected network layer trained on the CIFAR-10 dataset. It is better to view this figure in color.

the bias term from an ANN can be transposed into an SNN by injecting a constant input current at the beginning of each applicable time window. This strategy has proven effective in addressing the internal covariate shift issue in many cases [51], [62]. However, in this work, this approach is not directly applicable to our proposed TTFS conversion method. The reason for this lies in the fact that once the BN parameters are merged with the weight sum constraint inherent in the pre-trained ANN model, the constraint becomes invalidated. Essentially, the preservation of this constraint is crucial for the functionality of our proposed framework.

To compensate for the absence of BN, we propose to normalize the pre-activation distribution of each layer implicitly by introducing a new loss term $\mathcal{L}_A$ as given in Eq. (20). Since we expect the activation values to lie within $(0, 1]$ as desired for ReLU1, therefore, we apply an L1 loss to the pre-activation of each layer, encouraging them to fit a normal distribution with zero mean and standard deviation of 1/3, i.e., $\mathcal{N}(0, 1/9)$. This ensures 99.7% pre-activation values will lie within the interval $[-1, 1]$, such that the activation values will mostly lie within $(0, 1]$.

$$\mathcal{L}_A = \sum_{l=0}^{L-1} \sum_{i=0}^{N^l} \left| \sum_{j=0}^{N^{l-1}} w_{ij}^l a_j^{l-1} - A_i^l \right| \tag{20}$$

where A_i^l is a vector, wherein entry i is the normalized form, draw from $\mathcal{N}(0, 1/9)$, of the pre-activation $\sum_{j=0}^{N^{l-1}} w_{ij}^l a_j^{l-1}$.

In Fig. 5, we show an example of how the pre-activation normalization strategy effectively drives the distribution of pre-activation values towards a normal distribution of $\mathcal{N}(0, 1/9)$. Without the normalization, the pre-activation values are skewed towards a mean of -0.3. The effectiveness of this strategy in pre-training high-performance ANNs will be further demonstrated in our ablation study introduced in Section IV-D4.

D. Overall LC-TTFS algorithm

The proposed LC-TTFS conversion algorithm consists of two stages. In the first stage, we pre-train an ANN with the constraints described above, and the overall loss function is defined as follows

$$\mathcal{L} = \mathcal{L}_{ce} + \lambda_W \mathcal{L}_W + \lambda_A \mathcal{L}_A \tag{21}$$

where the $\mathcal{L}_{ce}$ is the cross-entropy loss for classification tasks, $\mathcal{L}_W$ is the weight regularization loss defined in Eq. (14), and $\mathcal{L}_A$ is the pre-activation normalization loss defined in Eq. (20). λ_W and λ_A are hyperparameters that balance the contribution of each individual loss term. Additionally, the hard constraints are imposed after each weight update to ensure a unity weight sum for all the neurons.

In the second stage, the weights of the pre-trained ANN are directly copied to the SNN to perform inference. We set the threshold of neurons in the last SNN layer to be infinity, such that the decision can be made based on the accumulated membrane potential. By directly mapping the pre-activation of ANN into the neuron membrane potential of SNN, it frees the ANN from applying the activation function in the output layer that deteriorates the pre-training.

IV. Experiments on Image Classification

In this section, we first evaluate the effectiveness of the proposed LC-TTFS conversion algorithm on the image classification task. Then, we perform a comprehensive analysis of the conversion efficacy and computational efficiency of converted SNNs. Finally, we present ablation studies that are designed to validate the effectiveness of each individual component of the proposed algorithm.

A. Experimental Setup

1) Datasets: We perform image classification experiments on CIFAR-10 and CIFAR-100 [63] datasets, which are commonly used for benchmarking SNN learning algorithms. The CIFAR-10 dataset consists of 60,000 colored images with a standard dataset split of 45,000, 5,000, and 10,000 for train, validation, and testing. These images are categorized into 10 classes with an image size of 32×32×3. The CIFAR-100 dataset is an extended version of CIFAR-10, which includes the 32×32×3 images from 100 classes. We follow the same image pre-processing procedures as outlined in [62].

2) Implementation details: To facilitate the comparison with other existing works, we adopt VGGNet [64] and ResNet [2]. In particular, we follow the same VGG-11, VGG-16, and ResNet-20 network architectures as described in [49], except that we do not include BN layers. The dropout is applied after every layer except the pooling layers in VGG-11 and VGG-16, whereas it is only applied after the fully connected (FC) layers in ResNet-20. Given the absence of BN layers, it is important to have a proper weight initialization. To this end, we initialize the weights of convolutions layers to the normal distribution $\mathcal{N}(0, \frac{2}{k^2 n})$, where k and n correspond to the kernel size and the number of output channels at each layer. For FC layers, the weights are initialized to the normal distribution $\mathcal{N}(0, 0.0001)$.

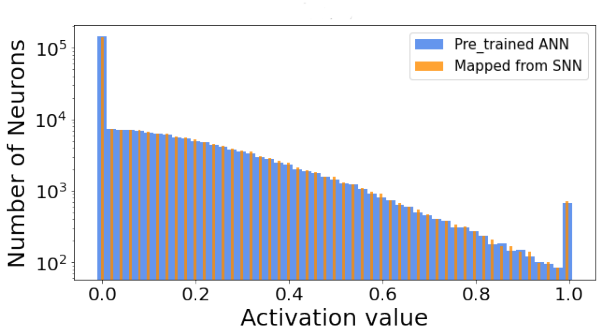


Fig. 6: Comparison of the distribution of activation values of a pre-trained ANN and those mapped back from the spike times of the converted SNN following the TTFS encoding scheme. The data is extracted from a randomly selected network layer trained on the CIFAR-10 dataset.

We use the PyTorch library for all experiments, as it supports accelerated and memory-efficient training on multi-GPU machines. We train VGG-11 and VGG-16 for 300 epochs and ResNet-20 for 200 epochs, and we adopt the SGD optimizer with a momentum of 0.9 and weight decay of 0.0005. We initialize the learning rate at 0.01 and decay its value by 10 at 0.6, 0.8, and 0.9 of the total number of epochs. We follow a discrete-time simulation for SNNs, with a time step of 0.02 ms and 50 time steps for each layer. We report the classification accuracy on the test set.

B. Experimental Results and Analysis

1) Classification Accuracy: Table I reports our experimental results on CIFAR-10 and CIFAR-100 datasets. For the CIFAR-10 dataset, our VGG-11 and ResNet-20 models achieved a SNN model classification accuracy of 91.25% and 92.67%, outperforming all other existing conversion methods. Our VGG-16 model also achieved competitive accuracy to other prior works using the same network structure. The same conclusion can also be drawn from the results of the CIFAR-100 dataset. Our ResNet-20 model achieves an accuracy of 72.36%, which is the best-reported result on this dataset as far as we know. Similarly, our VGG-16 model outperforms other works using temporal coding, which is also competitive to methods based on rate coding.

The efficacy of our algorithm in achieving a lossless TTFS conversion is pronounced when looking at the conversion error. Notably, as shown in the last column of Table I, the conversion errors from the pre-trained ANNs are negligible. To better understand the origin of the conversion errors, we have plotted the activation values of the pre-trained ANN against those mapped back from the spike times of the converted SNN (following the TTFS encoding scheme). As illustrated in Fig. 6, the data distribution of the mapped back activation values closely follows that of the pre-trained ANN, except for the quantization errors arising from the discretized SNN simulation. The effect of such quantization errors is marginal from our experimental results and can be easily addressed by increasing the temporal resolution of SNN.

C. Computational Efficiency

The TTFS-based SNNs are believed to be computationally more efficient than their rate-based counterparts. To shed light on this point, we follow the practices adopted by Intel [65] and compute the power proxy via the following equations:

$$P_{proxy} = SynOPs + NeuronOPs, \tag{22}$$

where $SynOPs$ and $NeuronOPs$ are the total number of synaptic operations and the total number of neuron updates, respectively. As extrapolated from the Loihi architecture [66], the energy weightage of a singular NeuronOPs corresponds approximately to that of around 10 SynOPs. This is due to the NeuronOPs being a multi-bit operation. Importantly, our utilized neuron model, ReL-PSP, avoids the exponential leaky mechanism present in the LIF model. To update the neuronal state, ReL-PSP requires only an addition operation at each time step, which is almost equivalent to a SynOPs.

With reference to the configuration specifics reported by Lee et al. [67], we have computed and tabulated the total P_{proxy} cross our work and that of Diehl et al. [39], and Segupta et al. [48]. As illustrated in Table II, our TTFS conversion algorithm evidently surpasses the energy efficiency metrics of traditional rate-based conversion methods.

D. Ablation Studies

Here, we present the ablation studies that are designed to validate the necessity and contribution of each individual component in the proposed algorithm. All the experiments are performed on the CIFAR-10 dataset using VGG-11. The experimental results are summarized in Table III with more detailed discussions presented in the following.

1) Model 1: We began by removing all the constraints that applied to the pre-trained ANN (i.e., soft and hard constraints for the weight sum, ReLU1 activation function, and pre-activation normalization) and the converted SNN (i.e., dynamic firing threshold). As a result, the accuracy of the ANN improved by only 0.03% over the baseline model. It suggests applying the proposed set of constraints has negligible influence on the ANN pre-training. The converted SNN, however, failed to perform the image classification task with the test accuracy dropping to a chance level, indicating it is crucial to apply the proposed set of constraints for a lossless TTFS conversion.

2) Model 2a and 2b: We further studied the necessity of soft and hard constraints for the weight sum regularization. Dropping the hard constraint while keeping all the rest constraints during the ANN pre-training had minimal impact on the ANN model, whereas the accuracy of the converted SNN dropped by 4.62%. It suggests the hard constraint for weight sum regularization is essential to eliminate the time-warping problem discussed in Section III-B. We further explored dropping both the soft and hard constraints during the ANN pre-training. Interestingly, this allows the ANN model to perform better than the baseline model. However, we noticed that the accuracy of the converted SNN dropped by 62.92%, implying the soft constraint is critical for alleviating the time-warping problem.

TABLE I: Comparison of the classification accuracy of different SNN models on the CIFAR-10 and CIFAR-100 datasets. Note that Acc. refers to the accuracy of converted SNN, and Δ Acc. refers to the difference between the pre-trained ANN and the converted SNN.

Dataset	Method	Network Architecture	Neural Coding	Acc. (%)	Δ Acc. (%)
CIFAR10	SPIKE-NORM [48]	VGG-16	Rate	91.55	-0.15
	PTL [62]	VGG-11	Rate	91.24	0.65
	CQ trained SNN [51]	VGG-11	Rate	82.09	-0.05
	CQ trained SNN [51]	VGG-16	Rate	92.48	-0.08
	RMP [43]	VGG-16	Rate	93.63	-0.01
	RMP [43]	ResNet-20	Rate	91.36	-0.11
	Hybrid Training [49]	VGG-16	Rate	91.13	-1.68
	Hybrid Training [49]	ResNet-20	Rate	92.22	-0.93
	T2FSNN [58]	VGG-16	Temporal	91.43	-
	TSC [59]	VGG-16	Temporal	93.63	-0.01
	TSC [59]	ResNet-20	Temporal	91.42	-0.05
	Ours	**VGG-11**	**Temporal**	**91.25**	**-0.05**
	Ours	**VGG-16**	**Temporal**	**92.72**	**-0.07**
	Ours	**ResNet-20**	**Temporal**	**92.67**	**-0.05**
CIFAR100	CQ trained SNN [51]	VGG-like	Rate	71.52	-0.4
	RMP [43]	VGG-16	Rate	70.93	-0.29
	RMP [43]	ResNet-20	Rate	67.82	-0.9
	Hybrid Training [49]	VGG-11	Rate	67.87	-3.34
	T2FSNN [58]	VGG-16	Temporal	68.79	-
	TSC [59]	VGG-16	Temporal	68.18	0.25
	TSC [59]	ResNet-20	Temporal	70.97	0.54
	Ours	**VGG-16**	**Temporal**	**70.15**	**-0.13**
	Ours	**ResNet-20**	**Temporal**	**72.36**	**0.13**

TABLE II: Comparison of the Total P_{proxy}.

Model	Diehl et al.	Segupta et al.	Ours
VGG9	730.4M	794M	117M
ResNet9	956.4M	1,226.6M	107.8M
ResNet11	854.6M	796.3M	144.4M

3) Model 3: ReLU1 activation function bounds the activation values within a particular interval, so as to establish a one-to-one correspondence between the ANN and SNN. To understand the contribution of this constraint, we replaced the ReLU1 activation function with a standard ReLU function, while keeping all the rest constraints during the ANN pre-training. Interestingly, compared to the baseline model with full constraints, the model using the ReLU function performs slightly worse than the one using the ReLU1 function. This may be explained by the fact that the ReLU1 activation function can indirectly regularize the weight sum and hence lead to easier fulfillment of the weight sum constraint. However, when changed to the ReLU activation function, it results in a severe accuracy drop of 26.43% on the converted SNN. These results indicate the ReLU1 function is imperative to ensure a one-to-one correspondence between the activation value of ANNs and the spike time of SNNs.

4) Model 4: Similar to Models 2 and 3, we pre-trained the ANN with all constraints except for the pre-activation normalization. This relaxes the pre-activation values from falling into the value range desired by the ReLU1. It, therefore, leads to poorer accuracy in the pre-trained ANN. We notice that the accuracy drop is more significant for networks with more layers (data not shown here), which is due to the internal covariate shift problem discussed in Section III-C. As expected, the absence of the pre-activation normalization technique did not affect the SNN conversion, and it can still achieve comparable accuracy to the pre-trained ANN.

5) Model 5: Finally, we follow the same ANN pre-training procedures as the baseline model, while during the network conversion, we replaced the dynamic firing threshold with a fixed threshold of 1. This modification results in a 13.48% accuracy drop during the network conversion. To further illustrate the effectiveness of applying the dynamic threshold to address the temporal dynamics problem discussed in Section II-B1, we plotted the spike time distribution using the fixed threshold and the proposed dynamic threshold. As shown in Fig. 7 (a), the spike times spread outside their allocated time intervals, and this problem becomes more obvious for deeper layers. Consequently, it leads to a high level of mismatch from the pre-trained ANN. In contrast, with the proposed dynamic threshold, the neurons at different layers only spike in their allocated time interval, i.e., $[Tl, T(l+1))$. This ensures the pre-synaptic spiking neurons fully contribute to their post-synaptic spiking neurons, eliminating the temporal dynamics problem. Although this non-overlapping time window may scarify the synchronized benefit of SNN, we highlight that our goal is to achieve lossless conversion from ANN to SNN using temporal coding without any training in SNN. To this end, the ANN

TABLE III: Summary of the result of ablation studies. Note that Δ Acc. refers to the difference between the pre-trained ANNs and the converted SNNs given in columns 7 and 8, respectively. **Soft**: soft constraint for weight sum; **Hard**: hard constraint for weight sum; **ReLU1**: ReLU1 activation function; **Dynamic**: dynamic firing threshold; **Norm**: pre-activation normalization.

Model	Soft	Hard	ReLU1	Norm	Dynamic	ANN (Acc.%)	SNN (Acc.%)	Δ Acc.%
Baseline	✓	✓	✓	✓	✓	91.30	91.25	0.05
1	✗	✗	✗	✗	✗	91.33	10.00	81.33
2a	✓	✗	✓	✓	✓	91.57	86.95	4.62
2b	✗	✗	✓	✓	✓	92.10	29.18	62.92
3	✓	✓	✗	✓	✓	90.87	64.44	26.43
4	✓	✓	✓	✗	✓	89.60	89.67	-0.07
5	✓	✓	✓	✓	✗	91.30	77.82	13.48

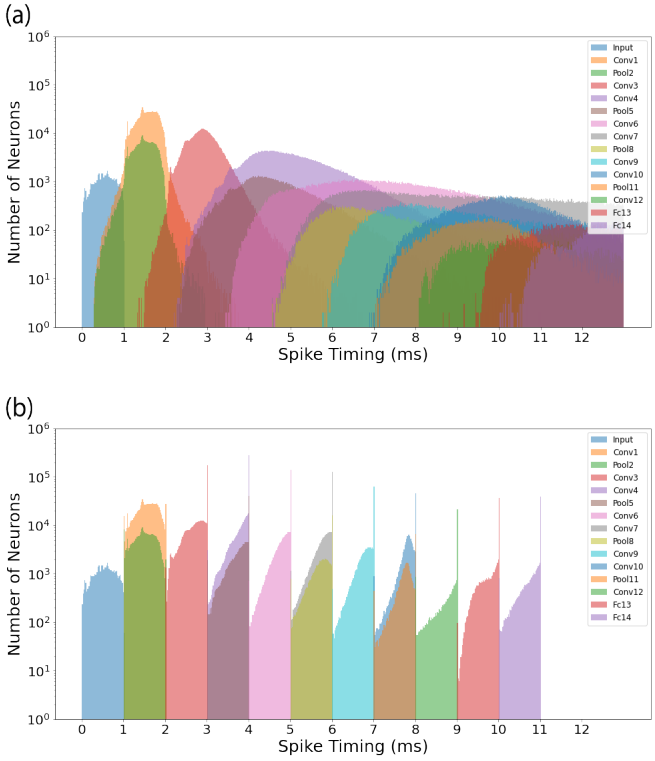


Fig. 7: Illustration of spike timing distribution of SNN models (a) with a fixed firing threshold and (b) with a dynamic firing threshold.

activation and SNN spiking time are needed to be perfectly matched with each other. Therefore, we adopt the proposed synchronized approach to realize this requirement. It is worth noting that the synchronized layer-wise processing is also used in other exciting temporal coding works, such as [57], [58], [68].

V. Experiments on Signal Reconstruction

In the previous section, we demonstrate the effectiveness and scalability of the proposed LC-TTFS conversion algorithm on image classification tasks. Existing TTFS-based learning algorithms often employ methods that achieve TTFS-based learning by either dropping or masking subsequent spikes after the first one during the training phase. Moreover, while there are gradient-based direct training algorithms cited in [45, 48], they have manifested convergence challenges, especially concerning deep neural networks. Although these methods might prove efficient for classification tasks, their efficacy diminishes notably when deployed for signal reconstruction tasks. In contrast, our proposed method adopts a conversion-based approach. This approach fundamentally eradicates the convergence issue and paves the way for establishing a direct and lossless mapping from the ANN. This ensures that information integrity remains uncompromised throughout the process. Performing TTFS encoding at each layer, allows us to preserve the information across the network and opens up the opportunity to perform signal reconstruction tasks with TTFS-based SNNs. In this section, we demonstrate the applicability and superior learning capability of the proposed LC-TTFS algorithm on two signal reconstruction tasks: image reconstruction and speech enhancement tasks.

A. Image Reconstruction with Autoencoder

Here, we first demonstrate the applicability of our algorithm to image reconstruction tasks with an autoencoder network. The autoencoder is a typical neural network that is used to learn compact latent representations of input signals via a bottleneck layer that has a reduced latent dimension. From this compact latent representation, the autoencoder then reconstructs the original input signals as accurately as possible [69].

1) Experimental Setup: The experiments on image reconstruction are performed using the MNIST dataset [70]. Following the approach from [62], we use a fully-connected autoencoder with a 784-128-64-32-64-128-784 architecture and train it to minimize the MSE between the original input and the reconstructed output signal.

The initial model was pre-trained as an ANN before being converted into a TTFS-based SNN using our proposed algorithm. We utilized the SGD optimizer for pre-training, with a cosine annealing learning rate schedule [71]. The output from spiking neurons was processed through a sigmoid function to generate the reconstructed image.

Evaluation of our model was carried out using two commonly used image quality metrics, PSNR and SSIM, both of which provided insight into the quality of reconstructed

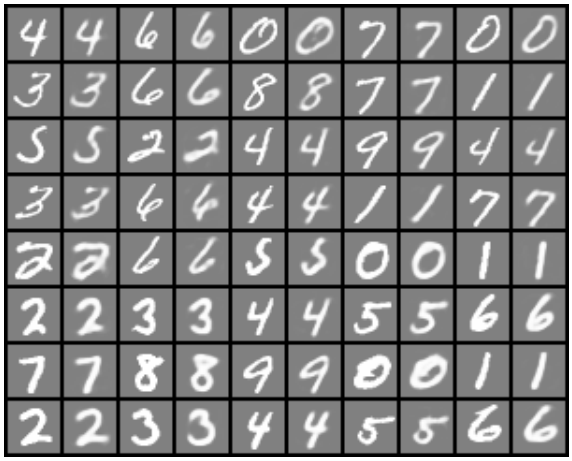


Fig. 8: Comparison of the original and the reconstructed images from our TTFS-based autoencoder. For each image pair, the left one is the original image, and the right one is the reconstructed image.

images. We provide the results of these evaluations on the test set.

2) Result and Analysis: Table IV summarises the results of on the image reconstruction task. Our TTFS-based SNN model achieves a comparable performance to the ANN-based counterpart in terms of the PSRN and SSIM metrics. In addition, the qualitative results shown in Fig. 8 demonstrate the TTFS-based SNN model can effectively reconstruct the images with high quality. Altogether, these results suggest the proposed conversion algorithm is highly effective for the image reconstruction task.

By representing information using spike times, the TTFS-based SNN is expected to greatly improve the energy efficiency over the rate-based counterparts. Following the same evaluation metrics introduced in Section IV-C, we report the energy proxy in Table IV. As shown, the energy efficiency of our TTFS-based model remains competitive with a high-optimized rate-based SNN reported in [62].

TABLE IV: Comparison of the results of different methods on the image reconstruction task.

Model	Coding	PSNR	SSIM	Energy Proxy
SNN [62]	Rate	20.74	0.84	715.8K
ANN (ours)	-	20.90	0.917	-
SNN (ours)	Temporal	20.83	0.916	581.2K

B. Time-domain Speech Enhancement

Speech enhancement, which improves the intelligibility and quality of noisy speech signals [66], has proven its importance in a vast amount of applications, including voice-based communication, hearing aids, speech recognition, and speaker identification [72]–[76]. Conventional speech enhancement methods are typically based on statistical models that estimate the data distribution of clean and noise signals, such as spectral subtraction [77] and Wiener filter [78]. However, these methods have shown limited improvements in speech quality under real-world scenarios. Recently, the ANN-based speech enhancement models have greatly improved the speech quality and intelligibility under complex acoustic environments [79]–[83]. Given a huge demand for speech enhancement technologies on mobile and IoT devices that have a limited power budget, it is, therefore, beneficial to develop power-efficient SNN-based speech enhancement models.

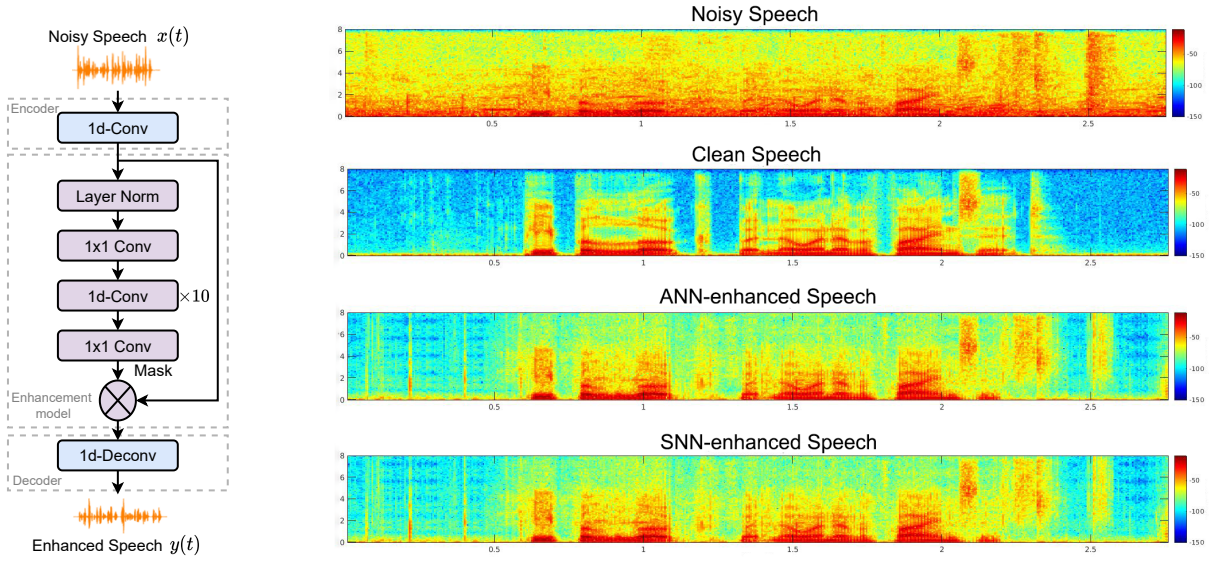
Speech enhancement can be considered as separating the human voice from the background noise. Inspired by the recent success of the time-domain speech separation model ConvTasNet [84], we proposed a TTFS-based speech enhancement model. As illustrated in Fig. 9(a), the proposed speech enhancement model takes the noisy speech waveform as input and outputs an enhanced speech waveform. This model consists of three parts: an encoder, an enhancement module, and a decoder. The **encoder** transforms the noisy speech waveform $x(t)$ into a high-dimensional feature representation, namely embedding coefficients, using a 1D convolutional layer. The 1D convolutional layer contains $N(=128)$ filters, and each filter is configured to have a time window of $L(=20)$ and a stride of $L/2(=10)$ samples. For each time window, the **enhancement module** estimates a mask, using a stack of dilated convolutional layers, to separate the noise from the human voice. A 1×1 convolution layer is first applied to normalize the encoded feature representation, and the dilated convolution layers with the filter of 128, kernel size of 1×3, and stride of 1 are repeated 10 times with doubled dilation rate of $[1,2,4,...,512]$. The mask for human voices is then estimated by applying another 1×1 convolution layer. Subsequently, the feature representation of the enhanced speech is obtained by masking the background noise with the estimated mask. Finally, the **decoder** reconstructs a high-quality speech waveform $y(t)$ from the masked feature representation using a 1D deconvolutional layer, which takes a reverse operation to the 1D convolutional layer in the encoder.

Following the ConvTasNet, we train the proposed SNN-based speech enhancement model to minimize the multi-scale scale-invariant signal-to-distortion ratio (SI-SDR) [85] loss, which is defined as:

$$\mathcal{L}_{SI-SDR} = -10log_{10}\left(\frac{\|\frac{\langle\hat{s},s\rangle}{\langle s,s\rangle}s\|^2}{\|\frac{\langle\hat{s},s\rangle}{\langle s,s\rangle}s-\hat{s}\|^2}\right) \tag{23}$$

where $\hat{s}$ and s are enhanced and reference clean speech signals, respectively. We normalize these two signals to zero means to ensure scale invariance.

1) Experimental Setup: To test our speech enhancement model, we employed a widely recognized dataset by Valentini et al. [86]. This dataset includes clean utterances from the Voice Bank corpus [87] and its noisy version created by combining clean utterances with environmental noise samples. The training set comprises 11,572 utterances mixed with ten types of noise at four different SNRs: 15 dB, 10 dB, 5 dB, and 0 dB. The test set contains 824 distinct utterances blended with five additional noise types at four SNRs: 17.5 dB, 12.5 dB, 7.5 dB, and 2.5 dB. Following the precedent set by SEGAN



(a) (b)

Fig. 9: (a) SNN-based speech enhancement network architecture. (b) From top to bottom: the power spectrum of noisy, clean, ANN-enhanced, and SNN-enhanced speech waveforms.

[79] and CNN-GAN [80], we reduced the original sampling rate to 16 kHz, without additional pre-processing.

Using the LC-TTFS algorithm, we pre-trained the ANN-based speech enhancement model for 100 epochs, utilizing an early stopping scheme and an Adam optimizer. The model was then converted into an SNN-based module, employing the membrane potential in the 1×1 spiking convolution layers.

We evaluate the speech enhancement models using the following standard metrics, which are available on the publisher's website[1].

1) PESQ: Perceptual evaluation of speech quality. The wide-band version recommended by ITU-T P.862.2 standard [88] is used in this work.
2) CSIG: Mean option score (MOS) prediction of the signal distortion attending only to the speech signal [89].
3) CBAK: MOS prediction of the intrusiveness of background noise [89].
4) COVL: MOS prediction of the overall effect [89].

All metrics are calculated by comparing the enhanced speech to the clean reference speech, we report the average values for all 824 utterances in the test set.

2) Result and Analysis: Table V compares the results of our ANN- and SNN-based speech enhancement models with other existing works using the four evaluation metrics introduced earlier. Our ANN-based model outperforms other existing methods across all the evaluation metrics, suggesting the effectiveness of the proposed model architecture. Moreover, the TTFS-based SNN model achieved comparable performance to the pre-trained ANN model, demonstrating the capability of the proposed TTFS conversion algorithm in solving the challenging speech enhancement task.

[1]https://www.crcpress.com/downloads/K14513/K14513_CD_Files.zip

We also performed subjective evaluation by listening to the enhanced speech signals generated by both the ANN- and SNN-based speech enhancement models. We find the SNN-enhanced speech samples are nearly indistinguishable from those high-quality ones generated from the ANN model. We publish some enhanced speech examples from the test set online to demonstrate our model performance[2]. In addition, we select a random speech sample from the test set and plot the power spectrum for the corresponding noisy, clean, ANN- and SNN-enhanced speech waveforms as shown in Fig. 9(b). It is clear that the SNN-enhanced speech spectrum exhibits a high level of similarity to the ANN-enhanced one, and both of them are very close to the ground truth clean speech spectrum. These results again highlight the effectiveness of the proposed model architecture and the TTFS conversion algorithm.

TABLE V: Comparison of the experimental results of different speech enhancement models. The results are higher the better.

Model	PESQ	CSIG	CBAK	COVL
Noisy	1.97	3.35	2.44	2.63
Wiener [79]	2.22	3.23	2.68	2.67
SEGAN [79]	2.16	3.48	2.94	2.80
CNN-GAN [80]	2.34	3.55	2.95	2.92
ANN (Ours)	**2.36**	**3.63**	**3.03**	**2.98**
SNN (Ours)	**2.35**	**3.64**	**3.01**	**2.98**

VI. Discussion and Conclusion

In this work, we identify and thoroughly investigate two major problems underlying the effective TTFS conversion, namely the temporal dynamics problem and time-warping

[2]The listening examples are available online at https://drive.google.com/file/d/1g5OCAtYH1B3tJE5_zGqDZam8bMcnIEvR/view?usp=sharing

problem. Based on this study, we further propose a novel TTFS conversion algorithm to address these problems, namely LC-TTFS. Firstly, to tackle the temporal dynamics problem, we introduce a dynamic firing threshold mechanism for spiking neurons that only allows neurons to fire within the allocated time window. In this way, the causal relationship between the input and output neurons is maintained throughout the network layers. Secondly, we apply a set of well-designed loss functions during the ANN pre-training to eliminate the time-warping problem. Finally, we apply the pre-activation normalization technique during the ANN pre-training to alleviate the internal covariate shift problem due to lack of BN layer. With these problems being well addressed, we establish a near-perfect mapping, apart from the marginal discretization error in the SNN simulation, between the ANN activation values and the SNN spike times, leading to a near-lossless TTFS conversion. This also enables us to go beyond the commonly considered classification tasks and opens up a new avenue for solving high-fidelity signal reconstruction tasks with TTFS-based SNNs.

The SNNs thus converted have demonstrated superior classification and signal reconstruction capabilities on image classification, image reconstruction, and challenging speech enhancement tasks. By representing information using spike times, instead of firing rates, we show TTFS-based SNNs can significantly improve the computational efficiency over both ANN and rate-based SNNs. By avoiding costly and ineffective direct SNN training, the proposed algorithm, therefore, opens up myriad opportunities for deploying efficient TTFS-based SNNs on power-constrained edge computing platforms. The carefully designed ablation studies on each individual component of the proposed algorithm highlight the necessity and synergy of these algorithmic components in achieving a near-lossless TTFS conversion. We would like to acknowledge that the proposed TTFS-based SNN requires a separate and non-overlapping time window for each layer, which adversely affects the inference speed. Therefore, we are interested in studying whether a perfect mapping can still be achieved under a shared time window to improve the inference speed, and we leave this as future work.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[3] Y. Wang, Y. Xu, R. Yan, and H. Tang, "Deep spiking neural networks with binary weights for object recognition," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 13, no. 3, pp. 514–523, 2021.

[4] X. Jin, M. Zhang, R. Yan, G. Pan, and D. Ma, "R-snn: Region-based spiking neural network for object detection," *IEEE Transactions on Cognitive and Developmental Systems*, pp. 1–1, 2023.

[5] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates *et al.*, "Deep speech: Scaling up end-to-end speech recognition," *arXiv preprint arXiv:1412.5567*, 2014.

[6] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[7] J. Hirschberg and C. D. Manning, "Advances in natural language processing," *Science*, vol. 349, no. 6245, pp. 261–266, 2015.

[8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[9] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[10] A. Lele, Y. Fang, J. Ting, and A. Raychowdhury, "An end-to-end spiking neural network platform for edge robotics: From event-cameras to central pattern generation," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 14, no. 3, pp. 1092–1103, 2022.

[11] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[12] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6869–6898, 2017.

[13] A. Aimar, H. Mostafa, E. Calabrese, A. Rios-Navarro, R. Tapiador-Morales, I.-A. Lungu, M. B. Milde, F. Corradi, A. Linares-Barranco, S.-C. Liu *et al.*, "Nullhop: A flexible convolutional neural network accelerator based on sparse representations of feature maps," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 3, pp. 644–656, 2018.

[14] Y.-H. Chen, J. Emer, and V. Sze, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 367–379, 2016.

[15] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6105–6114.

[16] W. Maass, "Networks of spiking neurons: the third generation of neural network models," *Neural networks*, vol. 10, no. 9, pp. 1659–1671, 1997.

[17] A. L. Hodgkin and A. F. Huxley, "Currents carried by sodium and potassium ions through the membrane of the giant axon of loligo," *The Journal of physiology*, vol. 116, no. 4, pp. 449–472, 1952.

[18] P. Dayan and L. F. Abbott, *Theoretical neuroscience: computational and mathematical modeling of neural systems*. Computational Neuroscience Series, 2001.

[19] E. M. Izhikevich, "Simple model of spiking neurons," *IEEE Transactions on neural networks*, vol. 14, no. 6, pp. 1569–1572, 2003.

[20] R. Brette and W. Gerstner, "Adaptive exponential integrate-and-fire model as an effective description of neuronal activity," *Journal of neurophysiology*, vol. 94, no. 5, pp. 3637–3642, 2005.

[21] J. Li, H. Tang, and R. Yan, "A hybrid loop closure detection method based on brain-inspired models," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 14, no. 4, pp. 1532–1543, 2022.

[22] M. Pfeiffer and T. Pfeil, "Deep learning with spiking neurons: opportunities and challenges," *Frontiers in neuroscience*, vol. 12, p. 774, 2018.

[23] A. Zhang, Y. Gao, Y. Niu, X. Li, and Q. Chen, "Intrinsic plasticity for online unsupervised learning based on soft-reset spiking neuron model," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 15, no. 2, pp. 337–347, 2023.

[24] E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks," *IEEE Signal Processing Magazine*, vol. 36, no. 6, pp. 51–63, 2019.

[25] G. Bellec, D. Salaj, A. Subramoney, R. Legenstein, and W. Maass, "Long short-term memory and learning-to-learn in networks of spiking neurons," *arXiv preprint arXiv:1803.09574*, 2018.

[26] Y. Wu, L. Deng, G. Li, J. Zhu, Y. Xie, and L. Shi, "Direct training for spiking neural networks: Faster, larger, better," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 1311–1318.

[27] S. B. Shrestha and G. Orchard, "Slayer: Spike layer error reassignment in time," in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Curran Associates, Inc., 2018, pp. 1412–1421. [Online]. Available: http://papers.nips.cc/paper/7415-slayer-spike-layer-error-reassignment-in-time.pdf

[28] J. Wu, Y. Chua, M. Zhang, G. Li, H. Li, and K. C. Tan, "A tandem learning rule for effective training and rapid inference of deep spiking neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–15, 2021.

[29] W. Fang, Z. Yu, Y. Chen, T. Huang, T. Masquelier, and Y. Tian, "Deep residual learning in spiking neural networks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 21 056–21 069, 2021.

[30] C. Duan, J. Ding, S. Chen, Z. Yu, and T. Huang, "Temporal effective batch normalization in spiking neural networks," *Advances in Neural Information Processing Systems*, vol. 35, pp. 34 377–34 390, 2022.

[31] X. Zhu, B. Zhao, D. Ma, and H. Tang, "An efficient learning algorithm for direct training deep spiking neural networks," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 14, no. 3, pp. 847–856, 2022.

[32] Y. Zhu, Z. Yu, W. Fang, X. Xie, T. Huang, and T. Masquelier, "Training spiking neural networks with event-driven backpropagation," *Advances in Neural Information Processing Systems*, vol. 35, pp. 30 528–30 541, 2022.

[33] J. Wu, Y. Chua, M. Zhang, Q. Yang, G. Li, and H. Li, "Deep spiking neural network with spike count based learning rule," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–6.

[34] S. K. Esser, R. Appuswamy, P. Merolla, J. V. Arthur, and D. S. Modha, "Backpropagation for energy-efficient neuromorphic computing," *Advances in neural information processing systems*, vol. 28, pp. 1117–1125, 2015.

[35] E. Hunsberger and C. Eliasmith, "Spiking deep networks with lif neurons," *arXiv preprint arXiv:1510.08829*, 2015.

[36] S. K. Esser, P. A. Merolla, J. V. Arthur, A. S. Cassidy, R. Appuswamy, A. Andreopoulos, D. J. Berg, J. L. McKinstry, T. Melano, D. R. Barch *et al.*, "Convolutional networks for fast, energy-efficient neuromorphic computing," *Proceedings of the national academy of sciences*, vol. 113, no. 41, pp. 11 441–11 446, 2016.

[37] P. O'Connor, D. Neil, S.-C. Liu, T. Delbruck, and M. Pfeiffer, "Real-time classification and sensor fusion with a spiking deep belief network," *Frontiers in neuroscience*, vol. 7, p. 178, 2013.

[38] Q. Liu, Y. Chen, and S. Furber, "Noisy softplus: an activation function that enables snns to be trained as anns," *arXiv preprint arXiv:1706.03609*, 2017.

[39] P. U. Diehl, D. Neil, J. Binas, M. Cook, S.-C. Liu, and M. Pfeiffer, "Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing," in *2015 International joint conference on neural networks (IJCNN)*. ieee, 2015, pp. 1–8.

[40] P. U. Diehl, B. U. Pedroni, A. Cassidy, P. Merolla, E. Neftci, and G. Zarrella, "Truehappiness: Neuromorphic emotion recognition on truenorth," in *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2016, pp. 4278–4285.

[41] B. Rueckauer, I.-A. Lungu, Y. Hu, M. Pfeiffer, and S.-C. Liu, "Conversion of continuous-valued deep networks to efficient event-driven networks for image classification," *Frontiers in neuroscience*, vol. 11, p. 682, 2017.

[42] B. Rueckauer and S.-C. Liu, "Conversion of analog to spiking neural networks using sparse temporal coding," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2018, pp. 1–5.

[43] B. Han, G. Srinivasan, and K. Roy, "Rmp-snn: Residual membrane potential neuron for enabling deeper high-accuracy and low-latency spiking neural network," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 13 558–13 567.

[44] T. Bu, W. Fang, J. Ding, P. DAI, Z. Yu, and T. Huang, "Optimal ANN-SNN conversion for high-accuracy and ultra-low-latency spiking neural networks," in *International Conference on Learning Representations*, 2022.

[45] Z. Hao, T. Bu, J. Ding, T. Huang, and Z. Yu, "Reducing ANN-SNN conversion error through residual membrane potential," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 1, 2023, pp. 11–21.

[46] T. Bu, J. Ding, Z. Yu, and T. Huang, "Optimized potential initialization for low-latency spiking neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 1, 2022, pp. 11–20.

[47] J. Ding, Z. Yu, Y. Tian, and T. Huang, "Optimal ANN-SNN conversion for fast and accurate inference in deep spiking neural networks," 2021, pp. 2328–2336.

[48] A. Sengupta, Y. Ye, R. Wang, C. Liu, and K. Roy, "Going deeper in spiking neural networks: Vgg and residual architectures," *Frontiers in neuroscience*, vol. 13, p. 95, 2019.

[49] N. Rathi, G. Srinivasan, P. Panda, and K. Roy, "Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation," *arXiv preprint arXiv:2005.01807*, 2020.

[50] H. Zheng, Y. Wu, L. Deng, Y. Hu, and G. Li, "Going deeper with directly-trained larger spiking neural networks," *arXiv preprint arXiv:2011.05280*, 2020.

[51] Z. Yan, J. Zhou, and W.-F. Wong, "Near lossless transfer learning for spiking neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, 2021, pp. 10 577–10 584.

[52] R. Gütig and H. Sompolinsky, "The tempotron: a neuron that learns spike timing–based decisions," *Nature neuroscience*, vol. 9, no. 3, pp. 420–428, 2006.

[53] S. Thorpe, A. Delorme, and R. Van Rullen, "Spike-based strategies for rapid processing," *Neural networks*, vol. 14, no. 6-7, pp. 715–725, 2001.

[54] S. R. Kheradpisheh and T. Masquelier, "Temporal backpropagation for spiking neural networks with one spike per neuron," *International journal of neural systems*, vol. 30, no. 06, p. 2050027, 2020.

[55] S. R. Kheradpisheh, M. Mirsadeghi, and T. Masquelier, "Bs4nn: binarized spiking neural networks with temporal coding and learning," *Neural Processing Letters*, vol. 54, no. 2, pp. 1255–1273, 2022.

[56] M. Mirsadeghi, M. Shalchian, and S. R. Kheradpisheh, "Ds4nn: Direct training of deep spiking neural networks with single spike-based temporal coding," 2022.

[57] L. Zhang, S. Zhou, T. Zhi, Z. Du, and Y. Chen, "Tdsnn: From deep neural networks to deep spike neural networks with temporal-coding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 1319–1326.

[58] S. Park, S. Kim, B. Na, and S. Yoon, "T2fsnn: deep spiking neural networks with time-to-first-spike coding," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2020, pp. 1–6.

[59] B. Han and K. Roy, "Deep spiking neural network: Energy efficiency through time based coding," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part X 16*. Springer, 2020, pp. 388–404.

[60] A. Krizhevsky and G. Hinton, "Convolutional deep belief networks on cifar-10," *Unpublished manuscript*, vol. 40, no. 7, pp. 1–9, 2010.

[61] M. Zhang, J. Wang, J. Wu, A. Belatreche, B. Amornpaisannon, Z. Zhang, V. P. K. Miriyala, H. Qu, Y. Chua, T. E. Carlson, and H. Li, "Rectified linear postsynaptic potential function for backpropagation in deep spiking neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–12, 2021.

[62] J. Wu, C. Xu, X. Han, D. Zhou, M. Zhang, H. Li, and K. C. Tan, "Progressive tandem learning for pattern recognition with deep spiking neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[63] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[64] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[65] J. Timcheck, S. B. Shrestha, D. Ben Dayan Rubin, A. Kupryjanow, G. Orchard, Ł. Pindor, T. M. Shea, and M. Davies, "The intel neuromorphic dns challenge," *Neuromorphic Computing and Engineering*, 2023.

[66] P. C. Loizou, *Speech enhancement: theory and practice*. CRC press, 2007.

[67] C. Lee, S. S. Sarwar, P. Panda, G. Srinivasan, and K. Roy, "Enabling spike-based backpropagation for training deep neural network architectures," *Frontiers in neuroscience*, vol. 14, p. 119, 2020.

[68] S. Park and S. Yoon, "Training energy-efficient deep spiking neural networks with time-to-first-spike coding," *arXiv preprint arXiv:2106.02568*, 2021.

[69] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[70] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[71] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," *arXiv preprint arXiv:1608.03983*, 2016.

[72] L.-P. Yang and Q.-J. Fu, "Spectral subtraction-based speech enhancement for cochlear implant patients in background noise," *The journal of the Acoustical Society of America*, vol. 117, no. 3, pp. 1001–1004, 2005.

[73] D. Yu, L. Deng, J. Droppo, J. Wu, Y. Gong, and A. Acero, "A minimum-mean-square-error noise reduction algorithm on mel-frequency cepstra for robust speech recognition," in *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2008, pp. 4041–4044.

[74] A. Maas, Q. V. Le, T. M. O'neil, O. Vinyals, P. Nguyen, and A. Y. Ng, "Recurrent neural networks for noise reduction in robust asr," 2012.

[75] J. Ortega-García and J. González-Rodríguez, "Overview of speech enhancement techniques for automatic speaker recognition," in *Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP'96*, vol. 2. IEEE, 1996, pp. 929–932.

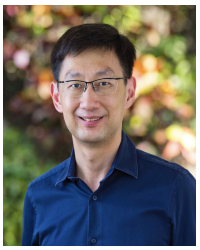
[76] C. Xu, W. Rao, J. Wu, and H. Li, "Target speaker verification with selective auditory attention for single and multi-talker speech," *IEEE/ACM*

Transactions on Audio, Speech, and Language Processing, vol. 29, pp. 2696–2709, 2021.
[77] M. Berouti, R. Schwartz, and J. Makhoul, "Enhancement of speech corrupted by acoustic noise," in *ICASSP'79. IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 4. IEEE, 1979, pp. 208–211.
[78] J. Lim and A. Oppenheim, "All-pole modeling of degraded speech," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, no. 3, pp. 197–210, 1978.
[79] S. Pascual, A. Bonafonte, and J. Serra, "Segan: Speech enhancement generative adversarial network," *arXiv preprint arXiv:1703.09452*, 2017.
[80] N. Shah, H. A. Patil, and M. H. Soni, "Time-frequency mask-based speech enhancement using convolutional generative adversarial network," in *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. IEEE, 2018, pp. 1246–1251.
[81] H.-S. Choi, J.-H. Kim, J. Huh, A. Kim, J.-W. Ha, and K. Lee, "Phase-aware speech enhancement with deep complex u-net," in *International Conference on Learning Representations*, 2018.
[82] J. Kim, M. El-Khamy, and J. Lee, "T-gsa: Transformer with gaussian-weighted self-attention for speech enhancement," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6649–6653.
[83] J. Abdulbaqi, Y. Gu, and I. Marsic, "Rhr-net: A residual hourglass recurrent neural network for speech enhancement," *arXiv preprint arXiv:1904.07294*, 2019.
[84] Y. Luo and N. Mesgarani, "Conv-tasnet: Surpassing ideal time–frequency magnitude masking for speech separation," *IEEE/ACM transactions on audio, speech, and language processing*, vol. 27, no. 8, pp. 1256–1266, 2019.
[85] J. Le Roux, S. Wisdom, H. Erdogan, and J. R. Hershey, "Sdr–half-baked or well done?" in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 626–630.
[86] C. Valentini-Botinhao, X. Wang, S. Takaki, and J. Yamagishi, "Investigating rnn-based speech enhancement methods for noise-robust text-to-speech." in *SSW*, 2016, pp. 146–152.
[87] C. Veaux, J. Yamagishi, and S. King, "The voice bank corpus: Design, collection and data analysis of a large regional accent speech database," in *2013 international conference oriental COCOSDA held jointly with 2013 conference on Asian spoken language research and evaluation (O-COCOSDA/CASLRE)*. IEEE, 2013, pp. 1–4.
[88] I. Rec, "P. 862.2: Wideband extension to recommendation p. 862 for the assessment of wideband telephone networks and speech codecs," *International Telecommunication Union, CH–Geneva*, 2005.
[89] Y. Hu and P. C. Loizou, "Evaluation of objective quality measures for speech enhancement," *IEEE Transactions on audio, speech, and language processing*, vol. 16, no. 1, pp. 229–238, 2007.

Qu Yang received a B.Eng. and an M.Sc. degree in electrical engineering from the National University of Singapore, Singapore in 2016 and 2019, respectively. She is currently a Ph.D. candidate in electrical engineering at the National University of Singapore. Her research focuses on neuromorphic computing and speech processing.

Malu Zhang (Member, IEEE) received the Ph.D. degree in computer science from the University of Electronic Science and Technology of China, Chengdu, China, in 2019. From 2019 to 2022, he was a Research Fellow with the HLT Laboratory, Department of Electrical and Computer Engineering, National University of Singapore, Singapore. He is currently a Professor with the Computational Intelligence Laboratory, School of Computer Science and Engineering, University of Electronic Science and Technology of China. His research interests include spiking neural networks, neural spike encoding, and neuromorphic applications. Dr. Zhang is now an Associate Editor of the IEEE Transactions on Emerging Topics in Computational Intelligence.

Jibin Wu received the B.E. and Ph.D degree in Electrical Engineering from National University of Singapore, Singapore in 2016 and 2020, respectively. Dr. Wu is currently an Assistant Professor in the Department of Computing, the Hong Kong Polytechnic University. His research interests broadly include brain-inspired artificial intelligence, neuromorphic computing, computational audition, speech processing, and machine learning. Dr. Wu has published over 30 papers in prestigious conferences and journals in artificial intelligence and speech processing, including NeurIPS, AAAI, TPAMI, TNNLS, TASLP, Neurocomputing, and IEEE JSTSP. He is currently serving as the Associate Editors for IEEE Transactions on Neural Networks and Learning Systems and IEEE Transactions on Cognitive and Developmental Systems. He also serves as the Editor for the Natural Language Processing Journal.



Kay Chen Tan (Fellow, IEEE) received the B.Eng. degree (First Class Hons.) and the Ph.D. degree from the University of Glasgow, U.K., in 1994 and 1997, respectively. He is currently a Chair Professor (Computational Intelligence) of the Department of Computing, the Hong Kong Polytechnic University. He has published over 300 refereed articles and seven books. Prof. Tan is currently the Vice-President (Publications) of IEEE Computational Intelligence Society, USA. He has served as the Editor-in-Chief of the IEEE Computational Intelligence Magazine from 2010 to 2013 and the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION from 2015 to 2020, and currently serves as the Editorial Board Member for more than ten journals. He is also an IEEE Distinguished Lecturer Program (DLP) Speaker and the Chief Co-Editor of Springer Book Series on Machine Learning: Foundations, Methodologies, and Applications.

Haizhou Li (Fellow, IEEE) received the B.Sc., M.Sc., and Ph.D. degrees in electrical and electronic engineering from the South China University of Technology, Guangzhou, China, in 1984, 1987, and 1990 respectively. He is currently a Presidential Chair Professor and the Executive Dean of the School of Data Science, The Chinese University of Hong Kong, Shenzhen, China. He is also an Adjunct Professor with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore. Prior to that, he taught with The University of Hong Kong, Hong Kong, during 1988–1990, and South China University of Technology, during 1990–1994. He was a Visiting Professor with CRIN, France, during 1994–1995, the Research Manager with the AppleISS Research Centre during 1996–1998, Research Director with Lernout & Hauspie Asia Pacific during 1999–2001, Vice President with InfoTalk Corp. Ltd., during 2001–2003, and Principal Scientist and Department Head of human language technology with the Institute for Infocomm Research, Singapore, during 2003–2016. His research interests include automatic speech recognition, speaker and language recognition, and natural language processing. Dr. Li was the Editor-in-Chief of IEEE/ACM TRANSACTIONS ON AUDIO, SPEECH AND LANGUAGE PROCESSING during 2015–2018, has been a Member of the Editorial Board of Computer Speech and Language since 2012, an elected Member of IEEE Speech and Language Processing Technical Committee during 2013–2015, the President of the International Speech Communication Association during 2015–2017, President of Asia Pacific Signal and Information Processing Association during 2015–2016, and President of Asian Federation of Natural Language Processing during 2017–2018. He was the General Chair of ACL 2012, INTERSPEECH 2014, ASRU 2019, and ICASSP 2022. Dr. Li is a Fellow of the ISCA, and a Fellow of the Academy of Engineering Singapore. He was the recipient of the National Infocomm Award 2002, and President's Technology Award 2013 in Singapore. He was named one of the two Nokia Visiting Professors in 2009 by the Nokia Foundation, and U Bremen Excellence Chair Professor in 2019.