Doctoral Dissertation

Efficient Implementations of Neural Network Powered by Spike Coding and Scalable Bisection Spanning

Man Wu

March 16, 2022

Graduate School of Science and Technology
Nara Institute of Science and Technology

A Doctoral Dissertation
submitted to Graduate School of Science and Technology,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Man Wu

Thesis Committee:

Professor Yasuhiko Nakashima	(Supervisor)
Professor Yuichi Hayashi	(Co-supervisor)
Associate Professor Renyuan Zhang	(Co-supervisor)

Efficient Implementations of Neural Network Powered by Spike Coding and Scalable Bisection Spanning[*]

Man Wu

Abstract

Deep neural networks (DNNs) have demonstrated state-of-the-art performances in the broad field of AI applications. However, these algorithms have intensive computational and colossal memory, making it challenging to develop on hardware platforms with limited computational resources. To address this challenge, this dissertation focuses on constructing efficient elastic neural networks (NNs) with spike coding and scalable bisection spanning to make NN inference less complicated, less redundancy, and more efficient to support the fully parallel reconfigurable NN platforms.

We present directly-trained spiking neural networks (SNNs) with ternary weight to achieve a good trade-off between complexity, latency, and performance to improve capacity extension of fully parallel and reconfigurable NN platforms by reducing its parameter computation and time steps. The proposed approach achieves 98.43%, 89.07%, 65.24% accuracy on N-MNIST, CIFAR-10, and CIFAR-100 with 4 time steps, respectively, and achieved up to 16x model compression. On the other hand, we develop and evolve a spatially scalable bisection NN architecture for fully parallel and reconfigurable NN platforms by improving the efficiency during the reconfiguration, which also supports on-demand array partitioned and reconfiguring (seen as "DiaNet"), and processes multiple applications in fully parallel through multi-grained reconfigurable architecture (MGRA). A byproduct of the model bisection paradigm, the proposed scalable bisection NN

[*]Doctoral Dissertation, Graduate School of Science and Technology, Nara Institute of Science and Technology, March 16, 2022.

minimizes computation and memory costs by reducing the number of operations and model size. This proposed DiaNet can achieve 90.8% parameters reduction with no loss of accuracy on MNIST. Moreover, we demonstrate that MGRA can process multiple applications in parallel and achieve a 10.8% power reduction simultaneously. Finally, we introduce the spike coding into spatially scalable bisection NN architecture to achieve temporal-spatial combined bisection NN architecture. The combined NN takes full advantage of the robustness and low power of SNN and ultra-sparse and multi-grained reconfigurable of DiaNet. In this sense, the model becomes linearly separable while processing information in temporal and spatial domains, better yet, reducing the computational and memory costs. Finally, the model achieves 96.10% accuracy with a 90.86% compression ratio with 8 time steps on MNIST and 98.15% accuracy with 69.38% parameters reduction with 6 time steps on N-MNIST.

Keywords:

DiaNet, sparse neural network, spiking neural network, multi-grained re-configurable, low complexity

Contents

List of Figures

List of Tables

1 Introduction

Recently, deep neural networks (DNNs) have demonstrated state-of-the-art performances in the broad field of AI applications including but not limited to image classification [1], speech recognition [2][3], computer vision [4] and natural language processing [5]. For example, AlphaGo [6] defeated world champion Lee Sedol in the Go game in 2016 and Hassan [7] appeared potential comparable performance to human brains in Chinese-to-English translation application in 2018. The high quality of service (QoS) of DNNs is always powered by employing massive computations. For instance, a widely applied VGG-16 model with 138 million parameters improves accuracy by 71% [8], the ResNet50 model requires 4 giga floating-point operations (FLOPs) during inference phases [9]. The implementation of such heavy computations is challenging for hardware platform.

In order to implement such intensive computations and colossal memory at the application end, efficient deep learning hardware has developed rapidly in recent years. The noticeable trend is to implement large-scale neural networks (NNs) on the Von Neumann processors with massively parallel and reconfigurable design, such as general-purpose graphics processing units (GPGPUs) [10] and tensor processing units (TPUs) [11]. However, these processors suffer from expensive overhead. A general trade-off between performances and costs lies in the parallelism of accelerations. By employing the feasibly massive computational cores, the calculations for NNs are allocated to a set of corresponding cores in a specific time- and space-division. This strategy, known as partially parallel, helps to improve processing speed with limited hardware resources. Along with the scaling down of VLSI technologies, the processing capacity of the computing platform greatly increases. Several domain-specified accelerators (DSAs) with tailored architecture are developed to DNN, which achieves higher energy efficiency compared with GPUs [12][13]. Unfortunately, the specified application domain leads to a narrow and inflexible scope of those parallel DSAs. Additionally, the field-programmable gate arrays (FPGAs) appear promising regarding system-on-chip (SoC) due to fine-grained reconfigurable architecture, which suffers from compilation time, and reprogramming overheads [14]. In order to further improve the energy efficiency, the dataflow–based coarse-grained reconfigurable architecture (CGRA) is used for accelerating multiply-accumulate calculations (MACs) computations of DNN [15][16][17]. However, these CGRA–based processors require more efficient processing units to achieve a good trade-off between flexibility and implementation costs. The NN–based approximate computing technologies are employed to construct efficient processing units with slight performance loss [18][19][20]. Furthermore, Since Von Neumann's architecture suffers from the "memory wall," the architecture alone cannot solve the problem of intensive computing and colossal memory of NN hardware [21]. In this sense, the ideal hardware platform of NN implementations is expected to be fully parallel (for high speed) and re-configurable (for application-flexibility) simultaneously to support various NN models and applications.

TrueNorth of IBM [22] and Loihi by Intel [23] provided massively parallel and higher energy efficiency inference engines by building millions of neurons and synapses in pre-silicon with a maximum estimation of utilization scale. Those processors are implemented based on spiking neural network (SNN) model, which is positioned for temporal coding information processing and low-power hardware evaluation [24]. In general, the fully parallel and re-configurable NN platforms meet two challenges for the capacity extension, which lies in reducing the parameters/computations for ubiquitous NNs and improving the efficiency during reconfiguration.

Addressing the former, a promising solution is to explore energy-efficient NN hardware by the algorithm co-design. Recent works have shown that the DNN models can be compressed and re-designed before developing on hardware towards maximizing the performance while minimizing the hardware cost [25] [26] [27] [28]. We found that the co-design approaches can be divided into reducing operations precision, and reducing the model size and the number of operations. For the former, it minimizes the bit-width by going from the float-point to the fixed point; furthermore, it eliminates the multiplication by ternarized/ binarized network [29][30][31]. In this sense, the efficient and compact DNN models could optimize the computation-intensive and memory-intensive issues from the unit- and topology levels. For the latter, it introduces the sparsity of the model to avoid a large number of multiplications, and memory requirements [32][33][34][35]. Nevertheless, the pruning method obtains a compressed topology at the cost of irregularities in the DNN model. Still, it requires additional index memory to address the non-zero weights, which increases the memory footprint, latency, and energy consumption [36]. Besides, the existing compressed method only focuses on optimizing DNN; it has yet to be well studied in SNN domain. Therefore, we need to construct efficient and compact SNN model to improve the capacity extension of parallelism and reconfigurability

On the other hand, the hardware resource for inactivated synapses/neurons still remains. since the processing elements (PEs) and interconnections are arranged in pre-silicon, noticeable parts of PEs and wire connections might be inactivated when the NN is reconfigured, leading to remarkable redundancy. Namely, while the platforms are reconfigured toward various tasks, a notable amount of hardware remains redundant, damaging the efficiency, even parallelism, and pipeline of the entire system. Therefore, the unit-, architecture- level and the topology of NN should be reconsidered to maximize the parallelism and reconfiguration while minimizing the cost. In our early study, a bisection topology of NN named DiaNet has been proven feasible and efficient [38]. By partitioning this symmetrical bisection network into diamond-shaped pieces, the reconfiguration is achieved without redundancy. However, the prototype of DiaNet can hardly be applied for NNs with large scales due to depth explosion. We therefore need to optimize the DiaNet for improving the efficiency during reconfiguration

This dissertation focuses on constructing efficient elastic neural network architectures with spike coding and scalable bisection spanning to improve the capacity scaling of NN platform. We aim to break the boundaries between algorithms and hardware stacks to provide a larger design space for the ideal NN hardware platform. Furthermore, a byproduct of our proposal, the proposed NN architecture minimizes computation and memory costs by reducing multiplication operations and the number of weights.

We investigate how to compress SNNs to achieve a good trade-off between complexity, latency, and performance from the temporal perspective. We aggressively compressed the SNNs by up to 16$\times$ with 4 timesteps (inference latency) and no accuracy loss on N-MNIST, CIFAR-10, and CIFAR-100. Moreover, we eliminate the multiplication operands of synaptic weight during the inference stage. On the other hand, from the spatial perspective, we develop and evolve spatially expanded bisection NN architecture, which supports on-demand array portioning and reconfiguring for parallel processing of different applications through multi-grained reconfigurable architecture (MGRA). A byproduct of the model bisection paradigm, the proposed spatial NN architecture minimizes computation and memory costs by reducing multiplication operations and the model's size. This spatial NN architecture can achieve 90.86% parameters reduction with no loss of accuracy on MNIST. Moreover, we demonstrate that MGRA can process multiple applications in parallel and achieve a 10.8% power reduction simultaneously [37]. To efficiently utilize the benefit of spike coding and scalable bisection spanning, we introduce the low-power spiking paradigm into spatially expanded bisection NN architecture to achieve a temporal-spatial combined NN structure. Specifically, we integrate both spatially expanded-based topology and temporal dynamical-based SNN model so that the network becomes linearly separable while processing information in temporal and spatial domains, better yet, reducing hardware overhead.

1.1 Motivation

The philosophy of this dissertation is to construct elastic neural network with spike coding and scalable bisection spanning to make NN inference less complicated, less redundancy and make it more efficient, to support the fully parallel and re-configurable NN platforms.

Motivation for Ternarized Spiking Neural Networks: As the third generation NN, SNNs have shown promising towards enabling low-power AI hardware thanks to their event-driven sparsity. Time flexibility of SNN is mainly manifested in time-based coding and time-based neuron dynamics, and time-based adjustable precision. Whereas the massive amount of floating-point parameters in SNNs still takes away considerable energy efficiency. Furthermore, the large time step of the SNN model leads to high computational cost and significant latency. In a nutshell, a ternarized SNN model is proposed to maximize accuracy while minimizing energy consumption and latency.

Motivation for Elastic Neural Networks with Scalable Bisection Spanning: In order to explore the ideal energy-efficient hardware platform for various NN applications, spatially expanded bisection NN architecture is developed for efficiently multi-grained reconfigurable on-chip in our early feasibility studies [38]. Instead of full connection between the adjacent layers, the architecture is designed with a symmetrical bisection connection paradigm to avoid global inter-communication. In this sense, a large scale of elastic bisection neural network (BNN) is expected to be symmetrically implemented by the computational hardware on-chip, which supports on-demand array partitioning and reconfiguring for any application without redundancy (seen as “DiaNet”). Specifically, the behaviors of synapse and neurons in entire network can be reconfigured for different DiaNet (seen as “fine-grained”); the organization of synapse and neurons in entire network can be reshaped for various applications (“see as mid-grained”); the organization of multi DiaNet for specific applications (“seen as coarse–grained ”). To verify the feasibility of BNN and multi-grained reconfigurable architecture (MGRA), a series of simple BNN-based approximate functions (1- and 2-operands) and its PE hardware implementation have been implemented. However, the simple approximate function of BNN is not enough to support various NN applications, which is inflexible in NN hardware platform. Furthermore, by directly applying DiaNet for complex pattern recognition applications, all the features are fed into the input layer leading to the remarkable depth of the structure, resulting in vanishing gradient or explosion problem.

Motivation for Temporal-spatial Combined Bisection Neural Networks: DiaNet topology minimizes the computation cost by reducing the number of multiplication operations and the model’s size. Moreover, the DiaNet takes full advantage of network-on-chip (NoC) resources for highly parallel and efficient dynamically composable to improve energy efficiency further. However, it still suffers from the overhead of floating-point operands and poor robustness performance (e.g., hardware variation). Simultaneously, SNNs have great promise in continuous spatial-temporal information processing with lower energy consumption and better robustness. Therefore, to efficiently utilize the benefit of temporal and spatial architecture, we introduce the low-power spiking paradigm into spatially expanded bisection NN architecture to achieve a temporal-spatial combined NN structure.

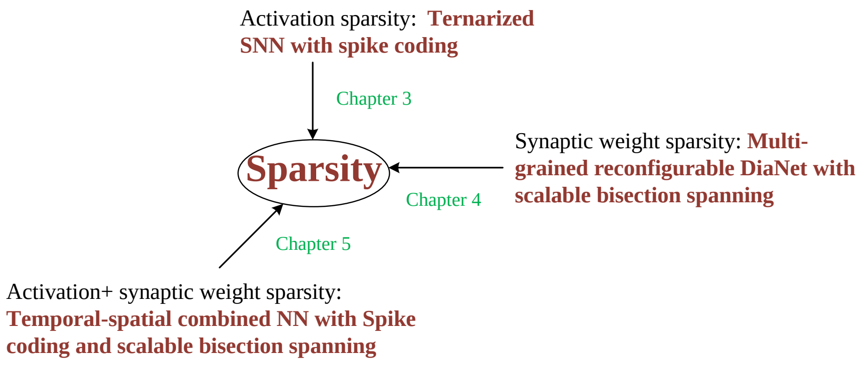

Activation sparsity: Ternarized SNN with spike coding
Chapter 3
Sparsity
Synaptic weight sparsity: Multi-grained reconfigurable DiaNet with scalable bisection spanning
Chapter 4
Chapter 5
Activation+ synaptic weight sparsity: Temporal-spatial combined NN with Spike coding and scalable bisection spanning

Figure 1. We exploit sparsity to improve the efficiency of NNs from multiple perspective

1.2 Outline of the Dissertation

All of these techniques revolve around exploiting the sparsity of neural networks to improve the efficiency for NN parallel hardware implementation, as shown in Fig. 1. The remaining chapters of the dissertation are summarized as follows:

Chapter 2. This chapter introduces the learning rules, architectures, and applications of application-oriented artificial neural networks and neuroscience-oriented spiking neural networks. We also survey the related work on sparse neural networks, efficient and compact spiking neural networks, and DiaNet topology.

Chapter 3. The feasibility of ternarizing spiking neural networks is studied in this chapter toward trading a slight accuracy for significantly reducing computation complexity and memory costs. We also introduce a spiking ResNet structure to mitigate the accuracy gap. The method described in this chapters will also be used for chapter 5.

Chapter 4. This chapter discusses the DiaNet topology toward a multi-grained reconfigurable architecture for NN applications. We also propose a series of evolutionary techniques of DiaNet topology, including I/O layer integration and skip connections, to prevent the depth explosion and gradient vanishing problem. Then, the sensitivity to the decline of computational precision and bit-width is investigated to suggest the guideline for efficient hardware implementations. Finally, the effectiveness of DiaNet is verified by the proposed multi-grained reconfigurable architecture on FPGA. The method described in this chapters will also be used for chapter 5.

Chapter 5. Previous chapters discuss how to compress the temporal network and how to optimize the spatial network efficiently. This chapter explores the temporal-spatial combined bisection neural network for further reducing the computation complexity and memory costs of DiaNet topology, and to provide flexible and powerful implementation framework for various NN applications.

Chapter6. This chapter summarizes the dissertation and discuss remaining challenges and future works for temporal-spatial combined NN architecture.

2 Background

This chapter presents the background material and related work to facilitate understanding of the remaining chapters. Firstly, we survey the learning rule, architecture, and application of neural networks, both conventional and spiking. Then, we describe related work about efficient and compact neural networks, including sparse NN and compressed SNN. Finally, we describe DiaNet topology, how it works, its application, and its limitation.

2.1 Neural Networks

Neural networks are inspired by the intelligence of biological brain and are composed of many synapses and neurons. A schematic representation of a biological neuron and its neural network counterpart is shown in Fig .2. Dendrite accumulates inputs x via its corresponding synaptic weights w. After dendrites operation, the soma integrates accumulation results and transmits them to the output. Hence, the neuron can be described as:

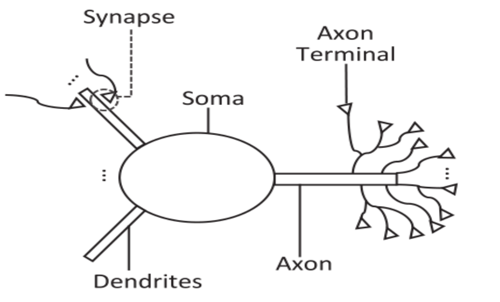
$$y = \sigma(\sum_{i} w_{i,j} \times x_i + b_i) \tag{1}$$

where the b is bias, and σ is the activation function. The neurons are organized into layers, and the cascade of these layers is connected for a NN model. If the number of layers between input neurons and output neurons is large, it is called a deep neural network. Modern DNNs usually have hundreds of layers. As well known, the modern DNN models can approximate any objective function based on Eq. (1).

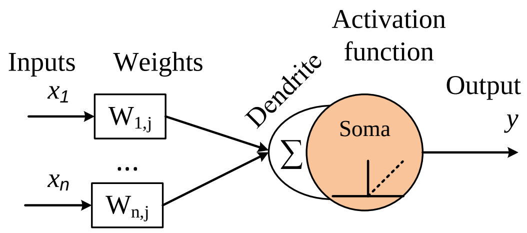
To some extent, the NN models can be loosely grouped into two categories [39]:

• Application-oriented artificial neural networks (ANNs). ANN integrates the spatial complexity through nonlinear activation functions and uses continuous analog signals for inter-neuron communication.

• Neuroscience-oriented spiking neural networks. SNN integrates temporal information through neuronal dynamics, and uses binary spikes for communication between spiking neurons.



(a) Biological neuron [24]



(b) Neuron in neural network

Figure 2. Schematic representation of a biological neuron and artificial neuron

2.1.1 Artificial Neural Network

Artificial Neural Networks have gained tremendous success in the broad field of AI applications. According to topology and application, ANN can be roughly divided into three types: (1) multi-layer perceptron (MLP); (2) convolutional neural network (CNN); (3) recurrent neural network (RNN). In MLP, each neuron from the previous layer is connected to the post layer, and the matrix-vector multiplication is computation-intensive due to the straight forward full connection (FC) fashion. In that vein, MLP calculations account for more than 61% of Google's TPU workload [11]. Empirically, MLP has plenty of room to explore a good trade-off between the complexity of the model and its performance. CNN makes full use of the shared weights to significantly reduce the number of weights. Specifically, CNN takes advantage of the spatial locality of the input images to share the weights in the space domain. RNN is good at processing sequential data by capturing temporal information through dynamic architecture. In this dissertation, we focus on the optimization of CNN and MLP architecture. Herein, MLP and CNN models used well-known gradient descent and backpropagation algorithm for training [40]. LetNet-5 [41] was proposed in 1998 for the handwritten digit recognition dataset (MNIST), which consists of two convolutional layers and two FC layers (as shown in Fig. 3). VGGNet [8] has strong generalization ability due to 138 million parameters across 13 convolutional layers and 2 FC layers. As shown in Fig. 4, each convolutional layer has the same kernel size, which is 3×3.

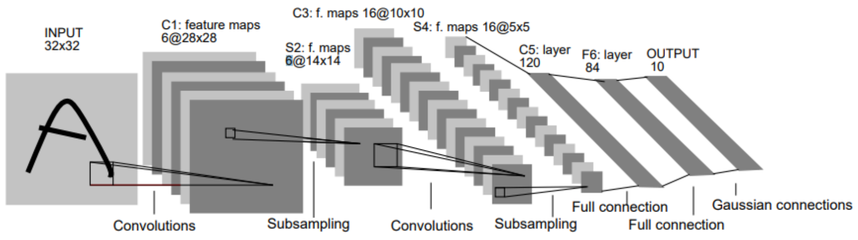


Figure 3. Architecture of LeNet-5 [41]

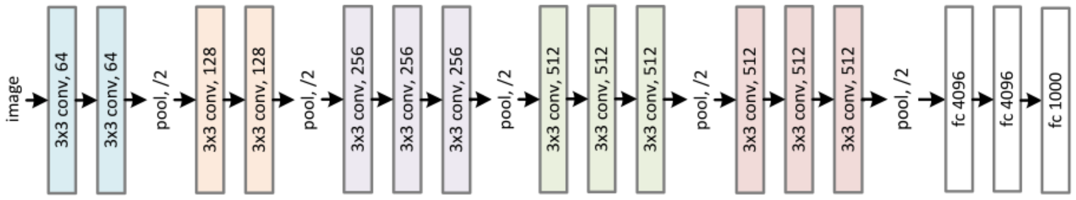


Figure 4. Architecture of VGG-16 [8]

Additionally, ResNet [42] proposed a deep residual learning framework with residual units to address the degradation problem of a very deep NN model, showing compelling performance in terms of accuracy and convergence. Each "Residual Units" can be described as:

$$\begin{gathered} \mathbf{y}_l = h\left(\mathbf{x}_l\right) + \mathcal{F}\left(\mathbf{x}_l, \mathcal{W}_l\right) \\ \mathbf{x}_{l+1} = f\left(\mathbf{y}_l\right) \end{gathered} \tag{2}$$

where x_l and x_{l+1} are input and output of the residual unit, $h(x_l)$ represents identity mappings, which implement by shortcut connections. In addition, $F(\mathbf{x}_l), \mathcal{W}_l$ denotes the residual mapping to be learned, and f is activation function. Besides, a series ReNet variants are implemented in [43]. The above NN architectures are used in the subsequent chapter.

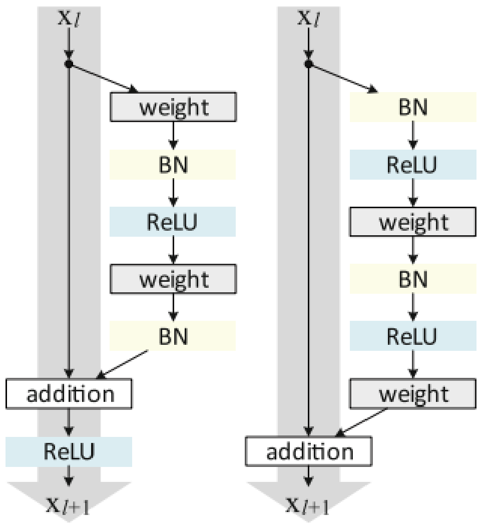


Figure 5. Architecture of residual unit [43]

2.1.2 Spiking Neural Network

Spiking neural network imitates how the cerebral cortex represents, processes and learns spatial-temporal information, as well as spike coding schemes and various learning rules [44] [45]. The schematic representation of SNN with multiple cascaded layers is shown in Fig. 6, in which neurons integrate the binary input spikes via synaptic weights and in turn generate spikes to post-layer when the membrane potential exceeds the threshold potential. Thus, SNNs have shown promising towards enabling ultra-low power AI hardware thanks to their event-driven sparsity [46] [47] [48]. Moreover, by accumulating spatial and temporal information, SNNs demonstrated great potential in robustness compared to conventional ANNs [49] [50]. Nevertheless, the training algorithms of SNNs still remain challenging. Specifically, the learning algorithm of conventional NN (such as backpropagation) cannot be directly used in SNNs due to the complex neuronal dynamics and non-differentiable characteristics of the activation function (e.g. Leaky-integrate-and-fire (LIF), and integrate-and-fire (IF) neuron model).

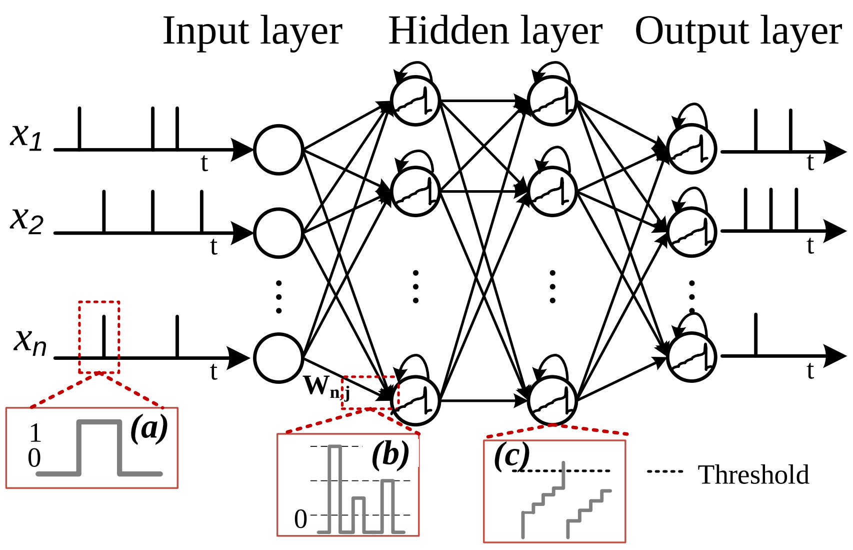


Figure 6. Schematic representation of spiking neural networks with multiple cascaded layers. (a) represents a binary spike of inputs; (b) describes the accumulation of spikes and synaptic weight; (c) illustrates the firing and resetting mechanism of a neuron.

The learning algorithm of SNN can be divided into three categories: (1) unsupervised learning algorithms; (2) direct supervised learning approaches; (3) indirect supervised learning methods. Spike-timing-dependent plasticity (STDP) is a well-known unsupervised learning algorithm of SNNs that updates the synaptic weights based on the correlation between the firing activities of pre-neuron and post neuron [51] [52]. Yet STDP still performs poorly due to local neuronal activity, rather than a global supervisor. More recently, the supervised learning algorithms of SNN, including ANN to SNN conversion (ANN2SNN) [53] and directly-trained methods [54], break through the limitation of traditional STDP that can only be applied to simple tasks, even reaching closed to DNN performance. The ANN2SNN refers to trained by ANN model then converts trained ANN model to SNN, which achieves near DNN accuracies on non-trivial datasets with thousands or hundreds of time steps [55]. The ”time steps” refers to repeating the inference process t times to accumulate the final membrane potential of the output layer, which is also inference latency. The major drawback of this method is high accuracy at the cost of high inference latency because the absence of the timing information and the firing threshold is the maximum pre-activation of the IF neuron. By contrast, directly-trained SNN was proposed based on surrogate gradient backpropagation through time (BPTT) [56] [57], which achieves satisfactory performance with fewer time steps through the comprehensive utilization of spatial-temporal information [58]. In that vein, Yujie Wu et al. [59] [60] proposed a spatio-temporal backpropagation (STBP) and developed efficient programming frameworks for directly training SNNs with high performance. Based on the STBP learning rule, Nitin Rathi et al. [61] proposed a hybrid training methodology including ANN2SNN for initialization and spike-timing-dependent backpropagation (STDB) for retaining progress, which achieves 10x faster compared to ANN2SNN with similar accuracy.

2.2 Efficient and Compact Neural Network

Although the DNN has gained tremendous success in many AI applications, the DNN model is usually over-parameterized, computationally intensive, and has colossal memory, resulting in higher computational and storage overhead [62]. Several different approaches have been proposed to address these challenges, including reducing the model size, reducing the number of operations, and reducing the precision of the operations. For instance, the quantization technology is introduced to reduce the bit-width of operation [29], ternarized/ binarized NN is proposed to eliminate the multiplications [30] [31][63], and the Knowledge distillation is proposed to reduce the model size [35].

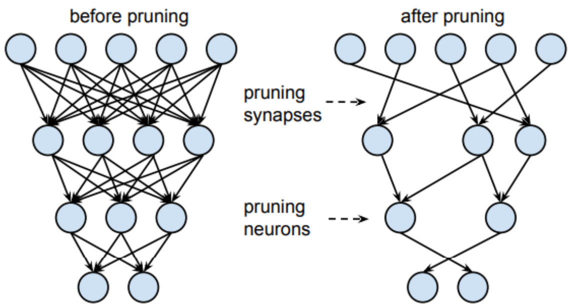


Figure 7. Illustration of pruning a neural network [64]

Besides, network pruning technology is a popular and universal solution that pruning the original NN topology into a sparse NN, as shown in Fig. 7. By removing the parts of parameters (e.g., synapses or biases), the computational complexity of NN can be greatly reduced. Song Han et al. [64] proposed a seminal work that prunes redundant connections in an iterative fashion, which can achieve 92% parameters reduction with no loss of accuracy on the ImageNet dataset. After that, pruning technology is one of the popular approaches to reduce the complexity of the ANN model [33][34][65]. These methods are all pruning during or after training, which means that the NN model must be pre-trained before pruning. By contrast, the NNs can be sparsified in the training phase by the pre-defined training technology in recent reports [66]. For example, Sourya Dey et al. [67] proposed pre-defined sparse NN reduce the complexity of training and inference stages. Souvik Kundu et al. [68] reported a pre-defined sparsity for CNN for low complexity, which achieves 83.3% parameter reduction on Tiny ImageNet dataset with 2.1% accuracy loss. However, these sparse NN models obtain a compressed topology at the cost of irregularities in the DNN model. Still, it requires additional index memory to address the non-zero weights, which increases the memory footprint, latency, and energy consumption [36][69]. Besides, Ryan Robinett et al. [70] reported structured sparse matrices for the DNN model. Guotian Xie et al. [71] proposed an interleaved structured sparse convolution block to eliminate the redundancy in convolution kernels.

2.3 Efficient and Compact Spiking Neural Network

As aforementioned, SNNs have shown promising for real-time embedded AI systems and edge devices as the third generation of neural networks. Whereas the massive amount of floating-point parameters in SNNs still take away considerable computation efficiency and memory footprint. Similar to conventional ANN, SNN also exits two directions to reduce the storage and computational requirements: reducing the number of parameters and reducing the precision of parameters (such as synaptic weight). Besides, the number of timesteps is also one of the factors that affect the computation overhead. To this end, a few recent works have tried to compress the SNN model through the mentioned approach. Steven K. Esser et al. [72] converted the ANN model with a ternary weight and binary activation into SNN with ternary weight, and in turn, developed them on the TrueNorth chip. Nitin Rathi et al. [73] presented a sparse SNN topology based on the STDP learning rule, which achieves 91.5% accuracy on MNIST when both pruning and quantization techniques were employed. Yuhua Shi et al. [74] reported a soft-pruning method on SNN, which maintained 90% accuracy on the MNIST with 75% weight reduction. Ruizhi Chen et al. [75] proposed a novel deep multi-strength SNN architecture to maintain the low-power benefit of SNN, which reduces the computational operation by 85% with slight accuracy loss. At the same time, these works are suffering from large timesteps. Lei Deng et al. [76] realized a comprehensive SNN compression method with fewer timesteps by introducing the alternating direction method of multipliers (ADMM), which is validated in NMNIST, CIFAR-100 datasets. However, their compressed SNN still can not escape from the FC manner of the first and last layers.

On the other hand, several works explored the SNN with binary weight, which is able to implement neuromorphic or "In memory" hardware with extremely low power consumption, thereby overcoming the memory bottleneck. Sen Lu et al. [77] converted binary neural networks (BNNs) to binary spiking neural networks (BSNNs), which demonstrated near full precision SNN accuracies on CIFAR-100 and ImageNet datasets with hundreds of timesteps. But at the same time, a significant latency and computational cost are caused by its large timesteps. Hong-Han Lien et al. [78] directly trained BSNNs based on the spatio-temporal back-propagation (STBP) learning rule, which can significantly reduce the timesteps. Yixuan Wang et al. [79] proposed a weights-threshold balance conversion method to binarize the weights for object recognition. Saeed Reza Kheradpisheh et al. [80] binarized SNN with temporal coding based on a directly supervised training algorithm and validated the proposal on MNIST and Fashion MNIST. The main drawbacks of these works are restricted to the small-scale dataset and oversize network architecture.

2.4 DiaNet Topology

To configure the computing unit on the chip without any redundancy, a hardware friendly bisection neural network (BNN) is developed instead of traditional full connection neural networks (FC-NNs) [38]. The topology of BNN is described in

Fig.8. Instead of conventional full connection topology, the entire neural network is organized by the connection of bisection structure. Each neuron communicates to two pre-synapses and two post-synapses in adjacent. Then, the feedforward propagation of BNN can be described as:

$$z^{l+1} = \sigma(w_{i-1}^{l+1} a_{i-1}^{l} + w_{i}^{l+1} a_{i}^{l} + b) \tag{3}$$

where the b is bias and w is weight. For a and b subscript denotes the layer number and superscript denotes the particular neuron in a layer. For example, the w_{i-1}^{l+1} denotes the weight of neuron $(i-1)$ in $(l+1)^{th}$ layer. The σ indicates activation function, i.e., *sigmoid* and ReLU can be formulated as:

$$sigmoid(x) = \frac{1}{1+e^{-x}} \tag{4}$$

$$ReLU(x) = \max(0, x) \tag{5}$$

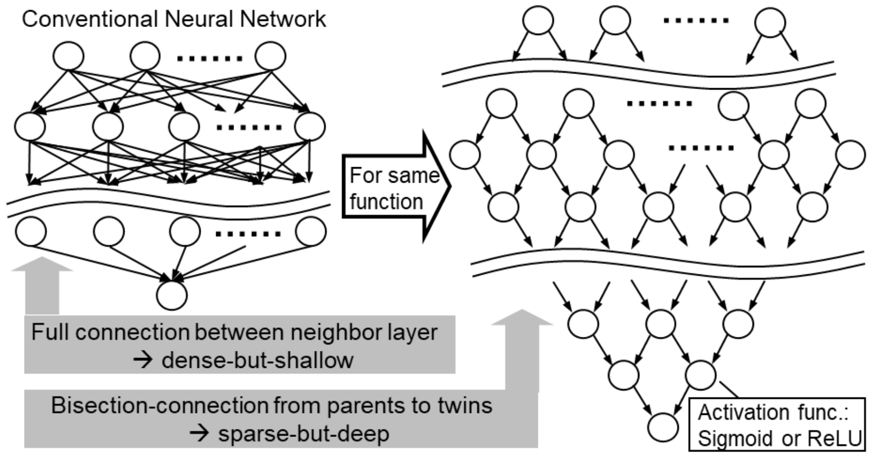

Conventional Neural Network
For same function
Full connection between neighbor layer → dense-but-shallow
Bisection-connection from parents to twins → sparse-but-deep
Activation func.: Sigmoid or ReLU

Figure 8. Topology of proposed BNN [38]

The reasonable combinations of neurons perform as the diamond-shaped NNs for specific applications such as regression for the approximate non-linear function. Thus, the structure of this NN is known as DiaNet. A large scale of BNN is expected to be symmetrically implemented by the computational hardware, which supports on-demand array partitioning and re-configuring for parallel processing of different applications. Figure 9 schematically illustrates that the entire network can be effectively partitioned into multiple DiaNets through multi-grained reconfigurable (MuGR), where two synapses and one neuron with activation function as packaged as a processing element (PE). Herein, the hardware implementations of DiaNets are fine-grained reconfigurable by applying various weights inside PEs; mid-grained reconfigurable by applying various DiaNet shapes for different tasks; coarse-grained reconfigurable by organizing massive DiaNets in any scheme.

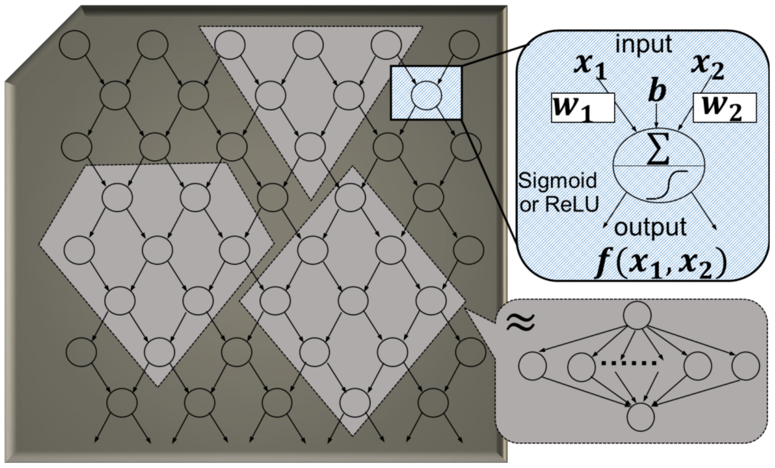


Figure 9. DiaNet topology: a large scale BNNs are symmetrically implemented in a large scale of network-on-chip; entire network can be re-configured and partitioned into multiple DiaNets [38].

3 Ternarizing Spiking Neural Network

Spiking neural networks show great potential in continuous spatial-temporal information processing with lower energy consumption, and yet, existing approaches usually introduce hyper-parameters and large latency that taking away significant computation efficiency and latency. Alternatively, we present directly-trained SNN with ternary weight to achieve a good trade-off between complexity, latency and performance. By leveraging a parametric integrate-and-fire (PIF) neuron with learnable threshold and STDB learning rule, the ternary spiking neural networks (TSNNs) enable directly trained with low latency and negligible loss of accuracy. Subsequently, a paradigm for binary-ternary dot-product operation is realized during the inference; therefore, the TSNNs achieve up to 16x model compression in contrast to the full precision SNNs. Moreover, to mitigate the accuracy gap, an optimized TSNN with a spiking ResNet structure is introduced into TSNN. In a nutshell, temporal-based SNN with ternary weight is discussed in this chapter.

3.1 Ternarizing Spiking Neural Network

In this section, we first propose the PIF neuron model to define the internal state of SNNs. Secondly, we propose TSNNs and implement binary-ternary dot product operation during the inference. Then, we introduce a one-layer-skip spiking ResNet structure to TSNN for compensating the accuracy loss. Finally, based on the STDB learning rule, the whole training process of TSNN is presented.

3.1.1 Parametric Integrate-and-Fire Neuron Model with Learnable Threshold

Compared with the LIF neuron, the IF neuron without leak parameters is more superficial in calculation and easier to implement in hardware with lower computational complexity. However, the IF neuron has worse adaptability for different tasks. To this end, we proposed PIF neuron model with a learnable threshold to increase heterogeneity, adaptability, and the expressiveness of neurons while keeping lower computational costs. The dynamic behavior of PIF neuronal activities can be formulated as:

$$u_i^l(t+1) = u_i^l(t) + \sum_j W_{ij}^{l-1} \cdot o_j^{l-1}(t+1) - V_{th} o_i^l(t) \tag{6}$$

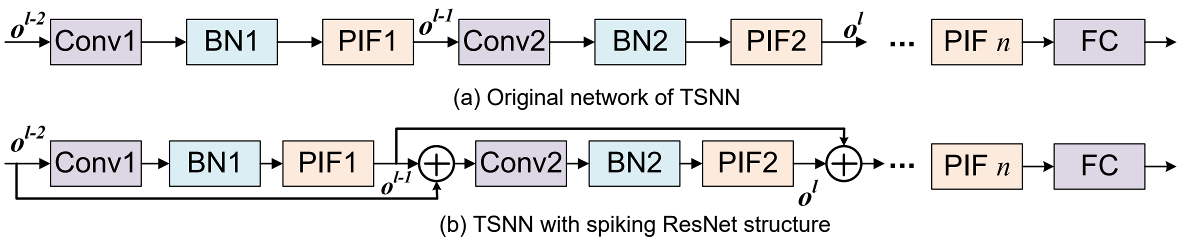
$$o_i^l(t) = \begin{cases} 1, & \text{if } u_i^l(t) > V_{th} \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

where $u_i^l(t)$ represents the membrane potential of $i-th$ neuron in $l-th$ layer at time t, W_{ij}^{l-1} denotes the synaptic weight between pre-neuron $j-th$ in $(l-1)-th$ layer and post-neuron $i-th$ in $l-th$ layer, o_j^{l-1} is binary spike output. Crucially, V_{th} is a learnable firing threshold potential rather than fixed hyperparameters. It is adaptively optimized during the training phase of the whole model, and its initial value is 1.0. Additionally, the threshold V_{th} is shared by all channels of one layer, which conforms to the similar characteristics of neighboring neurons in the biological. In this sense, PIF introduces a tiny number of extra parameters compared with the total number of parameters, which do not lead to the additional risk of overfitting.

In addition, the first term on the right side in Eq. (6) is membrane potential inherited from the previous timestep, the second term accumulates membrane potential from the last layer, and the third term defines soft resetting mechanism. Concretely, if the neuron generates a spike at $(t-1)$, the $u_i^l(t)$ at t will be lowered by V_{th}; on the contrary, the membrane potential will remain its value, seen as soft resetting mechanism. The soft resetting mechanism minimizes information loss during forwarding propagation. Equation (6)-(7) reveal that the firing mechanism of the PIF neuron, which emits spikes when the membrane potential $u_i^l(t+1)$ greater than the V_{th}, and vice versa. The firing mechanism also reveals that the deeper SNNs are easier to trigger notorious vanishing/exploding gradient problems. Additionally, the output layer of the PIF model only accumulates the inputs with synaptic weight overall timesteps but do not emits the output spike, which can be formulated as:

$$u_i^l(t) = u_i^l(t-1) + \sum_j w_{ij} \cdot o_j^l(t) \tag{8}$$

where, Eq. (8) reduces the information loss caused by the firing mechanism of the last layer.



o^{l-2}
Conv1
BN1
PIF1
o^{l-1}
Conv2
BN2
PIF2
o^l
PIF n
FC
(a) Original network of TSNN
(b) TSNN with spiking ResNet structure

Figure 10. Topology of TSNN: (a) TSNN topology; (b) TSNN with one-skip layer spiking ResNet structure to further minimize the accuracy degradation

3.1.2 Ternarizing Spiking Neural Networks

Following the ternary weight network (TWN) [63], we employ a similar paradigm to train ternary weight for SNNs. Specifically, using ternary weight $\mathbf{T} \in \{-1, 0, 1\}$ and a scaling factor α to approximate full precision weight during the forward and backward propagations, which can be formulated as:

$$\mathbf{W}^l = \alpha \mathbf{W}_{\mathbf{T}}^{\mathbf{l}} \tag{9}$$

where $\mathbf{W}^l$ and $\mathbf{W}_{\mathbf{T}}^{\mathbf{l}}$ represent full precision weight and ternary weight of SNN, respectively. Afterward, the approximated optimal solution was proposed to solve the problem of minimizing the $L2$ distance between $\mathbf{W}_l$ and $\mathbf{W}_{\mathbf{l}}^{\mathbf{T}}$, which can be described as:

$$W_T^l = \begin{cases} +1 * \alpha, & \text{if } \boldsymbol{W}^l > \Delta \\ 0, & \text{if } \boldsymbol{W}^l \leq \Delta \\ -1 * \alpha, & \text{if } \boldsymbol{W}^l < -\Delta \end{cases} \tag{10}$$

Here, Δ is a adjustable threshold parameters, which is calculated as follows:

$$\Delta = 0.7 \times E\left(\boldsymbol{W}^l\right) \tag{11}$$

Thus, as depicted in Fig. 10(a), TSNN uses spiking convolution layer (Conv), batch normalization (BN), and PIF neuron to construct our TSNN model. Note that, we ternarize the SNN model from scratch, including the first layer and last layer.

Binary-ternary dot product of TSNN. Once we obtain the SNN with ternary weights, achieving an effective binary-ternary dot-product operation is our next goal. Firstly, we employ batch normalization (BN) fusion to remove its multiply operation during the inference phase [81], thereby keeping the event-driven paradigm. Secondly, the dot-product operations of SNN with ternary weights can be expressed as:

$$o_j^{l-1}(t+1) \cdot \boldsymbol{W}_T^l = o_j^{l-1}(t+1) \cdot (\alpha \boldsymbol{T}) = \alpha \left(o_j^{l-1}(t+1) \cdot \boldsymbol{T} \right) \tag{12}$$

Thus, to accelerate the dot product with binary activation and ternary weight, Eq. (6) can be reformulated as:

$$u_i^{l'}(t+1) = u_i^{l'}(t) + \sum_j \left(o_j^{l-1}(t+1) \cdot \boldsymbol{T} \right) - \frac{V_{th}}{\alpha} o_i^l(t) \tag{13}$$

where, the second and third term of right side in Eq. (13) can be easily obtained. While the first term, the membrane potential $u_i^{l'}(t)$ is accumulated by:

$$u_i^{l'}(t) = \sum_{t=0}^{t-1} (\alpha \sum_j \left(o_j^{l-1}(t) \cdot \boldsymbol{T} \right) - V_{th} o_i^l(t)) \tag{14}$$

By incorporating the Eq. (12)-(14), we can reformulated Eq. (6) with binary-ternary dot product operation as below:

$$u_i^{l'}(t+1) = \frac{u_i^l(t+1)}{\alpha} = \sum_{t=0}^{t} (\sum_j \left(o_j^{l-1}(t) \cdot \boldsymbol{T} \right) - \frac{V_{th}}{\alpha} o_i^l(t)) \tag{15}$$

Subsequently, the $u_i^l(t+1)$ compares with $\frac{V_{th}}{\alpha}$ rather than V_{th}, Eq. (7) is reformulated as:

$$o_i^l(t) = \begin{cases} 1, \text{ if } \boldsymbol{u_i^l(t)} > \boldsymbol{\frac{V_{th}}{\alpha}} \\ 0, \text{ otherwise} \end{cases} \tag{16}$$

where, the α and V_{th} are the fixed parameters in the inference phase. So, the dot product between $\mathbf{T} \in \{-1, 0, 1\}$ and $o_j^{l-1}(t) \in \{0, 1\}$ are able to implement by fewer simple logic operations in contrast to full precision spiking neural networks (FPSNNs). Additionally, the bias is a constant in the inference stage, which can be formulated as:

$$u_i^{l'}(t+1) = \frac{u_i^l(t+1)}{\alpha} = \sum_{t=0}^{t} (\sum_j \left(o_j^{l-1}(t) \cdot \boldsymbol{T} \right) - \frac{V_{th}}{\alpha} o_i^l(t)) + \frac{b}{\alpha} \tag{17}$$

3.1.3 Spiking ResNet Structure to Improve accuracy

Due to the gradient vanishing and internal covariate shift issue, it is challenging to directly train deep SNN, thereby restricting the directly-trained method of SNN to the shallow architecture and small-scale datasets. Inspired by the superior performance of ResNet in very deep structures, a few works explored the spiking ResNet structure. Hanle Zheng et al. [58] proposed a threshold dependent BN method to replace native BN and applied it to the deep residual spiking network. Wei Fang et al. [82] introduced the spiking ResNet structure into SNN, which achieved higher performance on the ImageNet dataset by directly training 100 layers of SNN. However, these works are implemented with FC fashion. Empirically, TSNNs also suffer accuracy degradation by aggressive compression. To this end, we introduced one-layer-skip residual units into TSNN analogous to SNN with ResNet structures to avoid accuracy degradation. As illustrated in Fig. 10(b), the residual spiking units can be formulated as:

$$o^l(t) = PIF\left(F_s\left((o^{l-2}(t) + o^{l-1}(t)), W_l^T\right)\right) + o^{l-1}(t) \tag{18}$$

Here, F_s is spiking residual function, which is described by Eq. (6) and Eq. (7). On the one hand, our structure is similar to ResNet with $ReLU\ before\ addition$ structure. On the other hand, different from ResNet, the internal state of F_s not only depends on the input $o^{l-1}(t)$, identity mapping $o^{l-2}(t)$ and W_l^T, but also depends on the membrane potential of previous timesteps and V_{th}. Additionally, PIF is explained by Eq. (7), which follows the firing mechanism.

3.1.4 Overall Training Implementation

To compute gradients by leveraging the temporal and spatial information, we employ STDB learning rule and cross-entropy loss function L to directly train the overall TSNN model. The weight and threshold update is computed as:

$$\Delta W_{ij}^l = \sum \frac{\partial L}{\partial W_{ij}^l} = \sum \frac{\partial L}{\partial o_i^l(t)} \frac{\partial o_i^l(t)}{\partial u_i^l(t)} \frac{\partial u_i^l(t)}{\partial w_{ij}^l} \tag{19}$$

$$\Delta V_{th}^l = \sum \frac{\partial L}{\partial V_{th}^l} = \sum \frac{\partial L}{\partial o_i^l(t)} \frac{\partial o_i^l(t)}{\partial u_i^l(t)} \frac{\partial u_i^l(t)}{\partial V_{th}^l} \tag{20}$$

wherein, we introduce the surrogate gradient to approximate real gradient [57], which is described as below:

$$\frac{\partial o_i^l(t)}{\partial u_i^l(t)} = 0.3 * \max\left\{0, 1 - \left|u_i^l(t)\right|\right\} \tag{21}$$

Therefore, We present an informal programmatic construction for implementing the overall training of proposed TSNNs, as shown in algorithm 2.

Algorithm 1 Overall Training Algorithm

```
Input: Minibatch of inputs ($X$) and labels ($Y$), initial parameters: weight ($W_l$),
    membrane threshold $V_{th}$, membrane $u_l(t)$, times step($T$), learning rate $\eta$
Output: updated parameters $W_l^T$,$V_{th}$
 1: Forward propagation:
 2: for $t = 1$ to $T$ do
 3:     for $l = 1$ to $L - 1$ do
 4:         Obtain the weight threshold $\Delta$ by initial weight($W_l$) and (11)
 5:         Compute the ternary weight $W_l^T$ based on (10)
 6:         Obtain the scaling factor $\alpha^l$ and $\mathbf{T}$ by (9)
 7:         Calculate the membrane potential $u_i^l(t)$ with $\alpha^l$ and $\mathbf{T}$ by (15)
 8:         Generate the spike by (16) through pre-trained $V_{th}$
 9:     end for
10:     for $l = L - 1$ to $L$ do
11:         Obtain the scaling factor $\alpha^l$ and $\mathbf{T}$ by (9)   (11)
12:         Calculate the membrane potential $u_i^L(t)$ by (15) and do not generate
    spike
13:         Compute the loss $L$
14:     end for
15: end for
16: Backward propagation:
17: $W_l$ update: $W_l \leftarrow W_l - \eta\Delta W^l$ based on (19)
18: $V_{th}$ update: $V_{th} \leftarrow V_{th} - \eta\Delta V_{th}$ based on (20)
19: return $W_l$, $V_{th}$
```

3.2 Experiments and Results

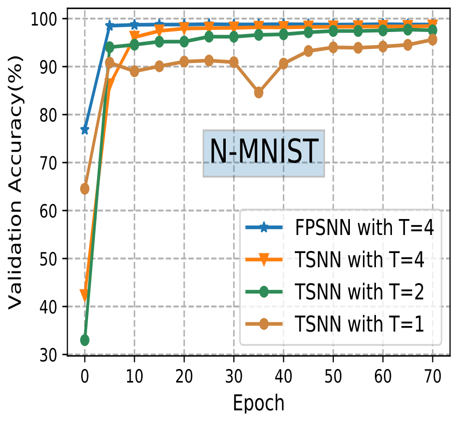
3.2.1 Datasets and Implementation

For proof of feasibility, we evaluate our TSNN on two public static datasets, including CIFAR-10 and CIFAR-100, and one neuromorphic dataset like N-MNIST [83]. CIFAR-10 comprises ten categories and 60000 colored images (including 50000 training images and 10000 testing images), whose sizes are 32 $\times$ 32. CIFAR-100 has the same configuration as CIFAR-10 but contains 100 categories. N-MNIST dataset is a spiking version of the original MNIST, whose size is 34 $\times$ 34, and the training-testing ratio is the same as CIFAR-10. We use normalization technology to pre-process input images and $L2$ weight decay of 0.00001 for regularization. In training, we employ the SGD optimizer with default parameters and the dropout technique with 0.2 probabilities. The learning rate is set to 0.1 at the beginning and is lowered by 0.5$\times$ at epoch 50. N-MNIST experiments are trained for 70 epochs with a batch size of 128, and CIFAR-10/100 experiments are trained for 200 epochs with a batch size of 64. Note that, due to straightforward terinarization operation of weight, we employ a direct encoding scheme [84] in this work to make the input features are the same in each timestep during the ternarized process. Furthermore, we use max-pooling technology in our TSNN model to maintain the binary output of each neuron. Additionally, all analysis experiments are implemented in the Pytorch framework [85].

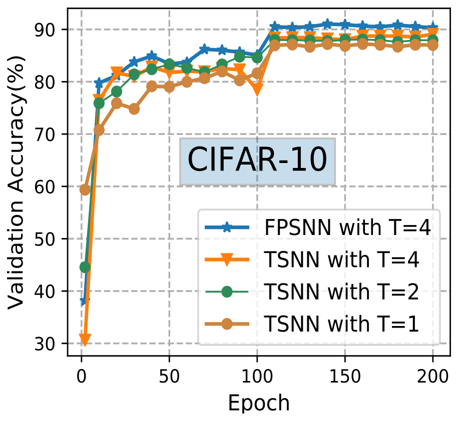
3.2.2 Experimental Results

Firstly, we investigate the performance gap between full precision SNNs and TSNNs. Then, we compare the performance between IF neuron with a fixed V_{th} and PIF neuron.

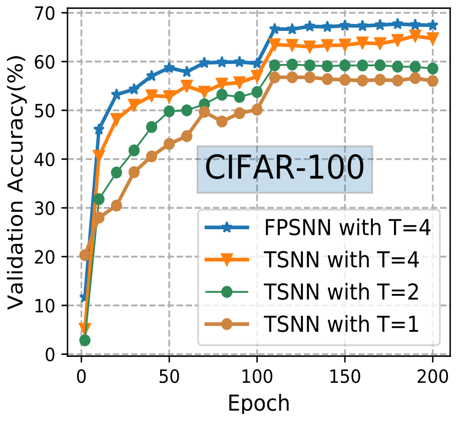
Comparison between TSNN and full precision SNN. Empirically, the performance of full precision SNNs is the upper-bound performance of any compressed SNNs. The directly-trained compressed SNNs have two factors that affect performance: on the one hand, the performance degradation (including accuracy and latency) caused by model compression; on the other hand, the unique nature of SNN makes them more challenging to train than the DNNs. To allow a fair comparison, FPSNN and TSNN models perform the same task in the same topol-



(a) N-MNIST



(b) CIFAR-10



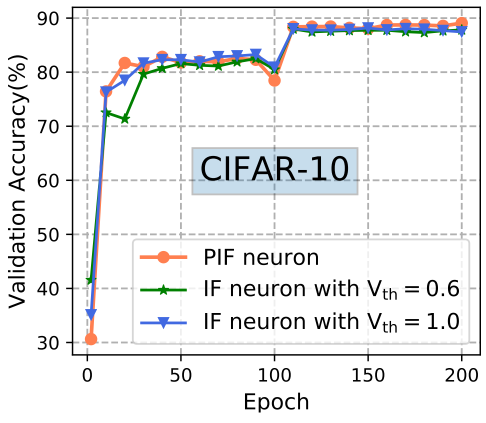
(c) CIFAR-100

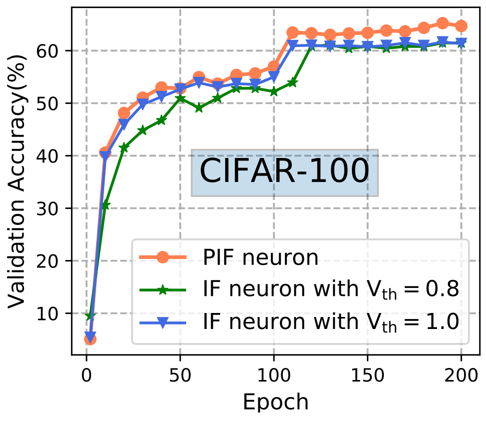
Figure 11. Validation accuracy curves of full precision SNNs and TSNNs with different timesteps over N-MNIST, CIFAR-10 and CIFAR-100 datasets

ogy with the same configuration. Fig. 11 depicts the validation accuracy curves of FPSNNs and TSNNs with different timesteps on N-MNIST, CIFAR-10, and CIFAR-100 datasets, where T represents timestep. As illustrated in Fig. 11(a), with 4 time steps, the N-MNIST with ternary weight achieves almost the same performance as FPSNN, which are superior to TSNN with 1 timestep. Indeed, the full precision SNN converges faster, but the TSNN achieves similar performance within 20 epochs. Additionally, as shown in Fig. 11(b), the FPSNN and

TSNN achieve 91.02% and 89.07% classification accuracy with just 4 timesteps over CIFAR-10, respectively. The TSNN can also get acceptable accuracy with extreme 1 timestep due to the direct encoding and VGG7-128 topology. However, as shown in Fig. 11(c), when processing a larger-scale of CIFAR-100, the accuracy of TSNN with 4 timesteps is increased by 8.56% compared with TSNN with a 1 timestep; compared with FPSNN, there is still a 2.26% accuracy gap. As a result, although longer timesteps lead to better accuracy, our TSNNs can achieve near full precision performance with only 4 timesteps and have a 16x compression rate compared to FPSNN counterparts.

Comparison between PIF and IF neuron. Since N-MNIST is easy to implement on the directly-trained SNN, the performance comparison of PIF and IF neurons are difficult to reflect on this dataset. Thus, we compare the validation accuracy of PIF neurons and IF neurons in TSNN on CIFAR-10 and CIFAR-100. The threshold potential V_{th} used to balance the membrane potential ensures the firing rate of the SNN, which also guarantees the performance of the model. Specifically, if the V_{th} is too small, the neuron emits spikes all the time, thereby the neuron insensitive to the feature. On the other hand, if the V_{th} is too large, the neuron is challenging to exceed the V_{th} for generating spikes, thereby leading to vanishing spike propagation. As a rule of thumb, we use 0.6&1.0 and 0.8&1.0 as V_{th} for CIFAR-10 and CIFAR-100, respectively. Fig. 12(a) shows the validation accuracy of mentioned experiments on CIFAR-10; the PIF neuron outperforms the IF neuron with a 1.05% accuracy gap. In addition, the comprehensive performance of PLF neuron on CIFAR-100 is better than that of IF neuron, such as 3.13% accuracy gap and convergence speed, as shown in the Fig. 12(b). In this sense, PIF has more prominent advantages for deeper SNN models. Moreover, we take V_{th} update process of the first layer as an example to analyze its self-optimization process, as shown in Fig. 13. The V_{th} update is consistent with Eq. (19) and backpropagation. Furthermore, the result in Fig. 12 and Fig. 13 demonstrate that PIF improves the performance of the SNN model without any extra computational cost.





(a) CIFAR-10 (b) CIFAR-100

Figure 12. Validation accuracy curves of PIF v.s. IF over CIFAR-10 and CIFAR-100 datasets

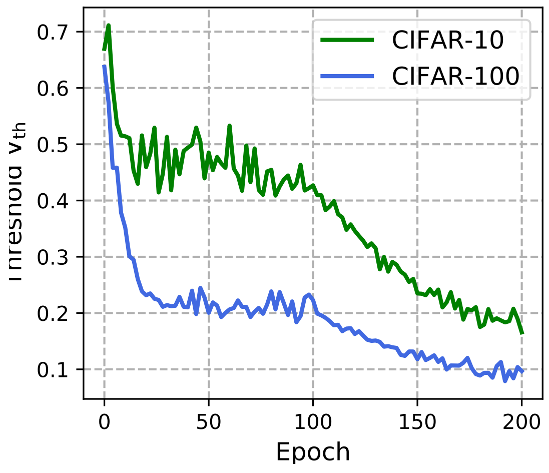


Figure 13. Update process of the first layer V_{th} during training over CIFAR-10 and CIFAR-100 datasets

3.2.3 Accuracy Evaluation on Spiking ResNet Structure

As aforementioned, we introduce the spiking ResNet structure into TSNN model to improve the accuracy, which can still benefit from the computation reduction and memory savings. To verify the spiking ResNet structure that can effectively compensate for the accuracy loss caused by the ternarization, we perform the experiments in a similar configuration on CIFAR-10 and CIFAR-100. On the basis of the TSNN prototype, the optimized TSNN with spiking ResNet improved by 0.84% and 0.51% on the CIFAR-10 and CIFAR-100, respectively, due to the one-layer-skip identity mappings. All configurations and experimental comparison results of TSNN and optimized TSNN are listed in Tab. 1. Indeed, the inputs of each neuron are changed, which provide a trade-off between the computational costs and accuracy.

Table 1. Comparison between the TSNN and TSNN with spiking ResNet structure.

Case	**Weights**	**Inputs***	**CIFAR-10**	**CIFAR-100**
TSNN	(-1,0,1)	(0,1)	89.07%	65.24%
TSNN with Spiking ResNet	(-1,0,1)	(0,1,2)	+0.84%	+0.51%

$*$ Here the inputs represent the input of each neuron.

3.3 Comparisons with State-of-the-art Works

We compare the proposed TSNNs with the state-of-the-art works on several benchmark datasets regarding the accuracy, timesteps, and compression rate. As illustrated in Tab. 2 and Tab. 3, our directly-trained TSNN not only compresses SNNs but also achieves high performance even on more complicated datasets. For neuromorphic N-MNIST dataset, compared to [60], this work performs a 16x compression rate with a slight accuracy loss and fewer time steps. Additionally, compared with BSNNs based on ANN2SNN [77] and [79], TSNNs improve accuracy and reduce the timesteps by 25× on CIFAR-10; moreover, our TSNNs achieve 3.17% 5.02% accuracy improvement and 26.25× 75× timestep reduction by sacrificing the compression rate of the model. The performance of TSNN is lower than BSNNs based on a directly-trained method with oversize architecture [78] because the model size has a significant impact on the compressed network.

However, our TSNN has a large model capacity compared to directly-trained BSNN. As a result, our TSNNs can compress parameters by 16x in various applications while maintaining close to full precision accuracy and low latency, thereby significantly reducing computational and memory costs.

Table 2. Network structures and training methods used for accuracy performance on various datasets

	Dataset	**Training Method**	**Architecture**
AAAI'2019 [60]*	N-MNIST	Directly-trained	128C3*2
This work	N-MNIST	Directly-trained	16C3*3
TCDS'2020 [79]	CIFAR-10	ANN2SNN	VGG9-128
ISCAS'2021 [78]	CIFAR-10	Directly-trained	VGG13-128
This work	CIFAR-10	Directly-trained	VGG7-128
This work with spiking ResNet	CIFAR-10	Directly-trained	ResNet9-128
TCDS'2020 [79]	CIFAR-100	ANN2SNN	VGG8-128
Front. Neurosci'2020 [77]	CIFAR-100	ANN2SNN	VGG16-64
This work	CIFAR-100	Directly-trained	VGG11
This work witht spiking ResNet	CIFAR-100	Directly-trained	ResNet11-128

[*] ∗ This model employs average-pooling in SNNs, which results in the activation output being float point rather than binary output.

Table 3. Performance comparison between the proposed method and the state-of-the-art quantization methods on various datasets.

	Dataset	Time **steps**	Precision **of weight**	**Accuracy**	Compression **rate**
AAAI'2019 [60]*	N-MNIST	10	Full precision	99.53%	1x
This work	N-MNIST	4	Ternary	98.43%	16x
TCDS'2020 [79]	CIFAR-10	100	Binary	90.19%	32x
ISCAS'2021 [78]	CIFAR-10	8	Binary	90.28%	32x
This work	CIFAR-10	4	Ternary	89.07%	16x
This work with spiking ResNet	CIFAR-10	4	Ternary	89.91%	16x
TCDS'2020 [79]	CIFAR-100	300	Binary	60.22%	32x
Front. Neurosci'2020 [77]	VGG16-64	105	Binary	62.07%	32x
This work	CIFAR-100	4	Ternary	65.24%	16x
This work witht spiking ResNet	CIFAR-100	4	Ternary	65.75%	16x

3.4 Conclusion

This section develops directly-trained TSNNs based on PIF neurons to achieve a good trade-off between complexity, latency, and accuracy by reducing computational and memory costs with low latency and slight accuracy loss. The proposed TSNN is evaluated on N-MNIST, CIFAR-10, CIFAR-100, which achieved 98.43%, 89.07%, 65.24% accuracy with 4 timesteps, respectively, and achieved up to 16x model compression. Furthermore, we introduce spiking ResNet structure into TSNNs to mitigate the accuracy gap. Based on this prototype, the optimized TSNN improves by 0.84% and 0.51% over CIFAR-10 and CIFAR-100 datasets, respectively. Besides, a paradigm for binary-ternary dot-product operation is proposed to eliminate MAC operation in the inference stage.

4 DiaNet: An Elastic Neural Network for Effectively Re-configurable Implementation

Since the PE and interconnection are arranged in pre-silicon, the noticeable PE and wire connections might be inactivated when the FC-NN is reconfigurable, leading to remarkable redundancy. It has been shown that the spatially expanded-based DiaNet behave as massive approximate calculation units (few operands) in fully parallel with rich flexibility and ultra-low cost [38]. Considering the global-grained re-configurability, it is expected to migrate more complex tasks (pattern recognition and complex vector regression for instance) onto DiaNet. In Chapter 4, we raised a key question:

It is possible to implement complex applications on the DiaNet topology with ultra-low sparse connections to achieve the same performance as FC-NN?

We answer this question in Chapter 4. In this chapter, we will implement the complex application by the improved and optimized scheme of DiaNet. Moreover, a series evolutionary technique of DiaNets topology is proposed to optimize the prototype.

4.1 DiaNet for Neural Network Applications

Compared to reconfigurable FC-NN in parallel, such as Loihi [23] or TrueNorth [22], the re-configuration of DiaNet avoids the enormous redundancy of synapses during the configurations. The efficiency of structure re-configuration, as shown in Fig. 14, has been investigated. Here, since the application is unpredictable, neuron and local connections should be planned on-chip. When the entire network is partitioned into two sets, $(n_1 + n_2) \times (m_1 + m_2)$ synapses are utilized and $m_1n_2 + m_2n_1$ synapses are inactive during reconfiguration in the FC-NNs. In contrast, reconfiguration in the DiaNets only requires $2 \times (n_1 + n_2)$ synapses and wastes only 2 synapses. It shows that the bisection topology can avoid numerous redundancy during reconfiguration as that in the conventional FC-NNs.

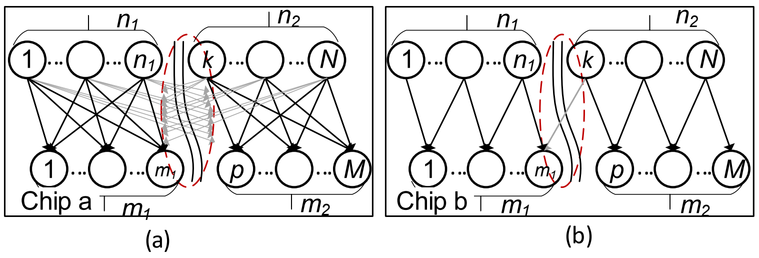


Figure 14. The redundancy comparisons of different NN topology; (a) partitioning two sets of FC-NNs; (b) partitioning two sets of BNNs.

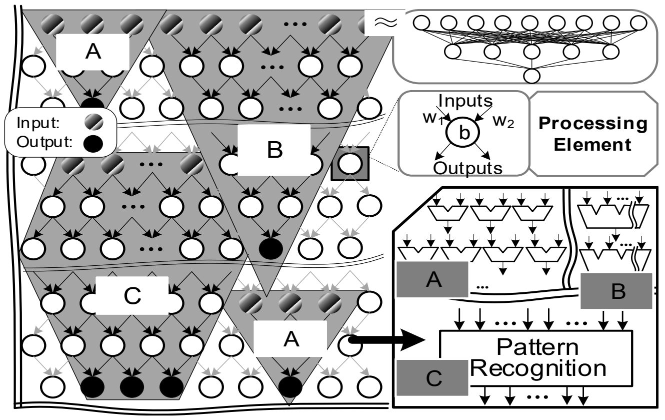


Figure 15. The layout of PE array with DiaNet topology: a large scale PE arrays can be re-configured and partitioned into various tasks, which are executed in independent DiaNets parallelly.

In this sense, the entire network with bisection mesh structure on hardware can be efficiently partitioned into arbitrary DiaNet for various application rather than simple regression with few operands with any redundancy. Figure 15 schematically illustrates that the entire network can be effectively partitioned into multiple DiaNets to support various tasks, which are configured by MuRA. Besides, global-grained re-configurable by directly mapping a complex NN inference task to a large scale of DiaNet. Herein, global-grained re-configurable is a direct mapping of complex NN inference task to a large scale of DiaNet, which is regarded as a complement to MuRA. Specifically, By configuring the entire network into pieces of DiaNets, three types of behaviors and computations can be carried out.

Type-A performs the operation of non-linear functions with few operands by small DiaNets; type-B is used to retrieve vector calculations; and type-C migrates the behaviors of complex full connection neural networks. In the real-world applications, computer vision for instance, type-A and B are expected to speed up the pre-processing and convolutions, and type-C helps to construct a full connection NN-based classifier in fully parallel.

As aforementioned, a reasonable connections of BNNs perform as DiaNet topology for specific applications. BNNs are essentially sparse NN with a fixed degree and fixed path-connection properties. How to ensure the effectiveness of the DiaNet network with extremely sparse properties? Interestingly, several works have proposed the necessary condition for sparse NN model with good representation ability, that is, sufficient information flows through the entire network [66] [86]. Namely, each output should be sensitive to all input features. Here, we provide the following three guidelines on designing DiaNet topology:

1. Recognition applications. When output node m greater than 1, to ensure that each output node depends upon all n input nodes, the DiaNet topology is composed of at least $(n + m - 2)$ hidden layers, wherein the upper part of DiaNet are expanded layers l_e, which require $(m - 1)$ layers. Hence, its lower part requires $(n - 1)$ layers, which are shrinkage layer l_s.
2. Regression application with n input greater than 3. On the premise of the necessary condition, the DiaNet topology is also composed of $(n + m - 2)$ layers, which presents inverted triangle-shaped and only has shrinkage layer l_s.
3. Regression application with n input less than 3. These DiaNet topologies are special diamond-shape [38].

Thus, as motivated by the above guidelines, an informal programmatic construction is described in the algorithm 2. Wherein l and i denote the number of layers and neurons, respectively. However, by directly applying DiaNet for complex pattern recognition applications, all the features are fed into the input layer leading to the remarkable depth of the structure. In that vein, the very deep DiaNets result in vanishing gradient problem [87], which leads to the network unable to converge.

Algorithm 2 Generating DiaNet1.0 Topology

```
Input: n inputs and m outputs
Output: DiaNet Topology
 1: if $m = 1 \rightarrow$ regression applications $(n \geq 3)$ then
 2:     // DiaNet topology only has $l_s$
 3:     for $l = 0, l \leq n + m - 2, l{+}{+}$ do
 4:         for $i = 1, i \leq n + l, i{+}{+}$ do
 5:             create neuron $(l, i)$
 6:             connect $(l - 1, i)$ and $(l, i)$
 7:             connect $(l, i)$ and $(l - 1, i + 1)$
 8:         end for
 9:     end for
10:     return DiaNet Topology
11: else $m \geq 2 \rightarrow$ recognition applications
12:     // DiaNet topology has $l_s$ and $l_e$
13:     for $l = 0, l \leq n + m - 2, l{+}{+}$ do
14:         if $l \leq m - 1$ then
15:             // Constructing $l_e$
16:             for $i = 1, i \leq n + l, i{+}{+}$ do
17:                 create neuron $(l, i)$
18:                 connect $(l - 1, i - 1)$ and $(l, i)$
19:                 connect $(l, i)$ and $(l - 1, i)$
20:             end for
21:         else
22:             // Constructing $l_s$
23:             for $i = 1, i \leq n + m + 1 - i, i{+}{+}$ do
24:                 create neuron $(l, i)$
25:                 connect $(l - 1, i)$ and $(l, i)$
26:                 connect $(l, i)$ and $(l - 1, i + 1)$
27:             end for
28:         end if
29:     end for
30:     return DiaNet Topology
31: end if
```

4.2 Evolution of DiaNet Topology

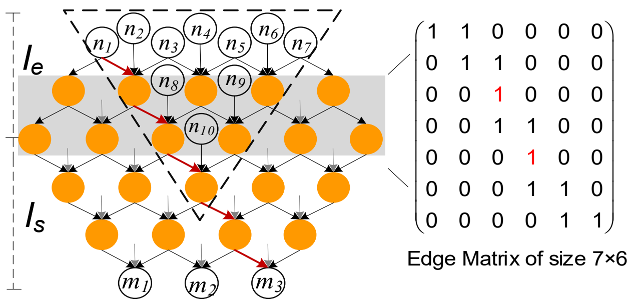
In this subsection, we address the remarkable depth problem in the structure of DiaNet topology by introducing I/O layer integration and skip connection techniques.

4.2.1 I/O Layer Integration

Instead of feeding all features into the input layer, they are suggested to feed into neurons in the hidden layer of the DiaNet prototype. Hence, developing the structure of I/O layer integration based on the DiaNet1.0 topology, which knows as DiaNet2.0. The adjacent matrices ($Ad_l \in \mathbb{R}$) between l^{th} and $(l+1)^{th}$ layer of SNN can be described as [66]:

$$Ad_l[i][j] = \begin{cases} 1, & if(i,j) \in E_l \\ 0, & otherwise \end{cases} \tag{22}$$

where, (i,j) denotes the connection between i^{th} neuron in the l^{th} layer and j^{th} neuron in the $(l+1)^{th}$ layer. E_l indicates the connections between consecutive layers; zero denotes absence of connections. Thus, the weight matrix of DiaNet2.0 topology is described in Fig. 16.



$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Figure 16. Layout of DiaNet 2.0 topology with adjacency matrices

The $(l+1)^{th}$ layer of DiaNet2.0 topology receives the information from preceding layer (l^{th} layer) and an additional feature x_i ($x_i \in n_i$). Then the feedforward can be formulated as:

$$z^{l+1} = \begin{cases} \sigma(w_{i-1}^{l+1}a_{i-1}^{l} + w_{i}^{l+1}a_{i}^{l} + w^{l+1}x_i + b), & l \in l_e \\ \sigma(w_{i-1}^{l+1}a_{i+1}^{l} + w_{i}^{l+1}a_{i}^{l} + w^{l+1}x_i + b), & l \in l_s \end{cases} \tag{23}$$

Crucially, DiaNet2.0 topology has to satisfy the necessary condition of constructing DiaNet. We explore the effectiveness of DiaNets2.0 by designing an "isosceles triangle" structure that accepts input features. Namely, neurons inside the "isosceles triangle" structure based on the DiaNet1.0 topology can directly receive input features that ensure each output depends upon all input features. In this manner, the number of features in the input layer (denoted as n_0) are reconsidered and can be formulated as:

$$\begin{cases} n_0 + (n_0 - 1) + (n_0 - 2) + ... + 2 + 1 \geq n_{total} \\ n_0 \geq 3 \end{cases} \tag{24}$$

where n_{total} denotes the total number of the inputs, and $(n_0 - 1)$ indicates the number of neurons in the first hidden layer that can accept input features.

Algorithm 3 Generating DiaNet2.0 Topology

```
Input: n inputs and m outputs
Output: DiaNet2.0 Topology
 1: // Calculate the minimum value of $n_0$ based on equation. 24
 2: // Call algorithm1 function to construct DiaNet1.0 topology with $n_0$ inputs
    and $m$ outputs. Thus, total number of layers are $n_0 + m - 2$; $k = n - n_0$
    features are suggested to feed into neurons inside the "isosceles triangle"
 3: if $k$>0 then
 4:     // Constructing "isosceles triangle"
 5:     for $l = 0, l \leq l_{n_0-1}, l++$ do
 6:         for $i = n_0 + 1, i \leq (n - n_0), i++$ do
 7:             Add additional input feature $x_i$
 8:         end for
 9:     end for
10: else $k = 0$
11:     return DiaNet1.0 Topology
12: end if
```

Analogously, with the number of hidden layers increases, the neuron that receives features decrease until reach the vertex layer l_{n_0} of “isosceles triangle”. Thus, we present an efficient algorithm that generates DiaNet2.0 topology (see informal Algorithmic program 3).

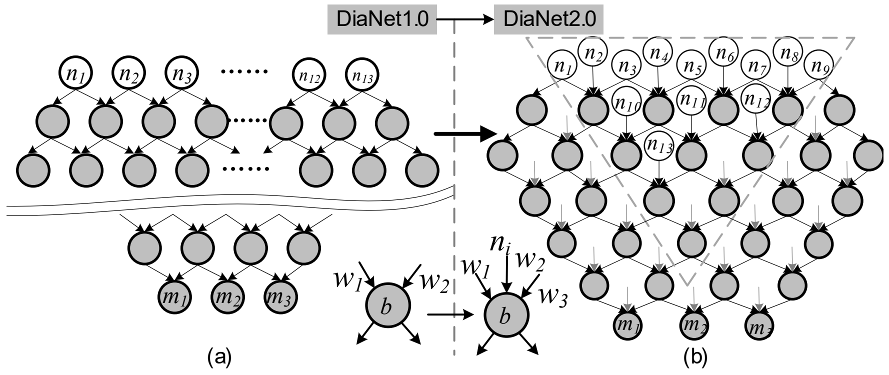

DiaNet1.0
DiaNet2.0
n_1 n_2 n_3 ...... n_{12} n_{13}
m_1 m_2 m_3
w_1 w_2 b
n_i w_1 w_2 w_3 b
n_1 n_2 n_3 n_4 n_5 n_6 n_7 n_8 n_9 n_{10} n_{11} n_{12} n_{13}
(a)
(b)

Figure 17. The layout of different DiaNet typologies for Wine dataset: (a) the DiaNet1.0 topology with 14 layers; (b) the DiaNet2.0 topology with 6 layers, which towards depth reduction.

Figure 17 illustrates the DiaNet1.0 and DiaNet2.0 topology of the Wine dataset with 13 attributes and three labels. Evolving from DiaNet1.0 to DiaNet 2.0 topology, the accuracy is increased from 88.89% to 98.15% and the number of layers towards depth reduction. The depth is one of the greatest problems causing the decay on inference quality of DiaNets. Assuming N inputs are fed into DiaNet1.0, the depth increases in $\mathcal{O}(N)$ for fanning the contribution of all inputs out to end as output. By DiaNet2.0, the depth grows in $\mathcal{O}(\sqrt{N})$ since the inputs are integrated to even the inner PEs. As a result, the theoretically necessary depth of DiaNet2.0 is smaller than the prototype along with N increases. In this manner, the inference quality for complex tasks is improved. Although the depth problem is alleviated and the accuracy is improved, the performance is still unstable and even crash on some complex tasks. For instance, to realize the recognition of MNIST, the construction of DiaNet topology requires at least 36 layers of BNN (according to Algorithm 2), and the accuracy is still about 20% because of the vanishing gradients. Fortunately, this problem has been largely addressed by additional techniques, including initial normalization [88] [89] and batch normalization [90]. However, these additional optimization mechanisms not only suffer from expensive computation but also show case-by-case behaviors that even damage the original topology.

4.2.2 Skip Connections

Skip connections are additional connections between nodes in non-consecutive layers of NN that skip one or more layers. Several works have recently demonstrated that skip connection has substantially improved the degradation problem of the extremely deep neural network [42] [91]. Moreover, other works have proposed novel explanations for skip connections: the introduction of short-circuit skip-connection can avoid the vanishing gradient problem and eliminate the linear dependence of singularities in the network[92] [93]. Inspired by these benefits, skip connection is introduced to address the vanishing gradient without destroying the original topology structure. Based on DiaNet2.0 topology, the evolution of DiaNet topology is constructed by adding a skip two-layer connection between each layer (note that skip connections are not allowed in the input layer), as shown in Fig. 18. This structure is called as DiaNet3.0 topology. In this sense, the Eq. (23) can be modified to Eq. (25), which describes the feedforward of DiaNet3.0 topology.

$$z^{l+1} = \begin{cases} \sigma(w_{i-1}^{l+1}a_{i-1}^{l} + w_i^{l+1}a_i^l + w^{l+1}x_i + b) + z^{l-1}, & l \in l_e \\ \sigma(w_{i-1}^{l+1}a_{i+1}^{l} + w_i^{l+1}a_i^l + w^{l+1}x_i + b) + z^{l-1}, & l \in l_s \end{cases} \tag{25}$$

where, Eq. (9) indicates the skip connectivity between $(l+1)^{th}$ layer and $(l-1)^{th}$ layer. The z^{l-1} denotes the identity matrix of $(l-1)^{th}$ layer. Detailed real-world experiments in Section 4.3 are supported by these architecture.

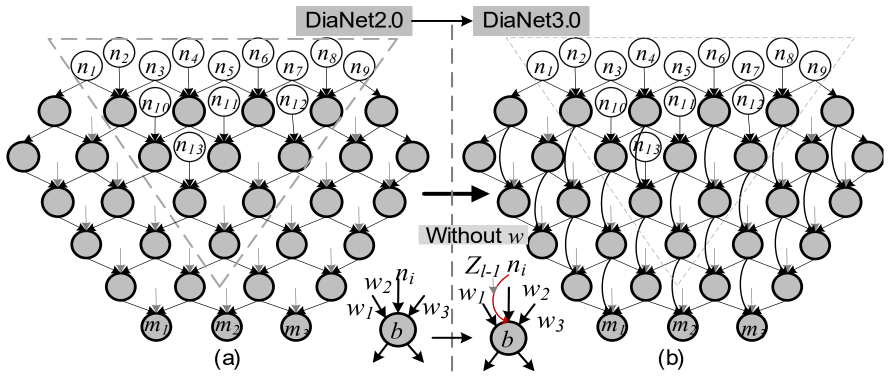


Figure 18. The layout of different DiaNet typologies for Wine dataset: (a) the DiaNet2.0 topology; (b) the DiaNet3.0 topology towards performance improvement by adding skip connections.

4.3 Experimental Results

In this subsection, we present experiments to analyze the performance of DiaNet topology with its evolution techniques and validate the effectiveness of the proposed methods.

4.3.1 Datasets and Implementation

Firstly, the UCI database [94] including Haberman's Survival, Iris, Car Evaluation, Pima India Diabetes, and Wine is employed to verified the recognition performance of DiaNet1.0. Secondly, we perform the experiments on the MNIST dataset to understand the typical behavior of series evolutionary technique of DiaNet. Similar to LeNet5 architecture [41], we setup the MNIST experiment consists of 2 convolution blocks for extracting features and DiaNet topology for classification. Each convolution block has convolution layers, max-pooling layer, ReLU activation function, and batch normalization technique. In this work, we only study the feasibility of classifier. In this sense, the output from the 2nd convolution block is flattened into a vector with 400 features. Namely, these 400 features and ten labels are employed to analyze the evolution of DiaNet topology. Thirdly, To further verify the performance of DiaNet3.0 , we perform the experiments on Fashion-MNIST [95] and CIFAR-10 datasets. Similar to MNIST and CIFAR-10, the Fashion-MNIST also consists of ten categories and 28×28 gray-scale images.

Additionally, the Wine dataset is employed to compare the performance of DiaNet prototype and its evolution techniques. Besides, we compare the performance between DiaNet and other sparse NNs on MNIST, Fashion MNIST and CIFAR-10 datasets. Finally, the Ionosphere and Waveform Database Generator dataset (version 2) from the UCI database are used to verify the feasibility of multiple independent DiaNets for parallel processing. The Ionosphere dataset consists of 34 attributes and two classifications. Besides, the Waveform Database Generator dataset (Waveform dataset) comprises 40 features and three labels. Our implementation is based on PyTorch library [85].

4.3.2 Experimental Results of DiaNet1.0

By applying the DiaNet topology, the recognition accuracy and the amount of synapses (see Tab. 4) are similar to the baseline of FC-NN (detailed in the following section) for all the examples [96]. However, the straight implementation of DiaNet leads to the remarkable depth of network structure. For instance, the Wine Dataset is performed by the thirteen-feature-three-class classifier, which is carry out by a DiaNet organized as the scheme of 13-14-15-14-13-12-11-…-4-3. The very deep structure results in the large number of synapses (even though smaller than FC-NN still) and difficulty on training [97]. In this sense, the complex tasks such as pattern recognition demand additional optimization technologies on the basis of DiaNet topology.

Table 4. Performance of DiaNet prototype over five datasets

Dataset	♯ instances	♯ feature	♯ class	Accuracy	♯ synapse
Haberman's Survival	306	3	2	69.57%	16
Iris	150	4	3	97.80%	42
Car.Eval	1728	6	4	95.57%	102
Pima India Diabetes	768	8	3	80.52%	86
Wine	178	13	3	88.89%	156

♯: represents the number

4.3.3 Experimental Results of Evolutionary DiaNet

For proof of feasibility, we analyze the behavior of DiaNet2.0 and DiaNet3.0 topology through various experiments. First, we explore the effect of evolution methods based on Section 4.2. Second, we analyze the impact of varying activation functions. All analysis experiments are trained for 70 epochs with a fixed batch size of 128. The learning rate is set to 0.01 initially and is lowered five times at epoch 10. The L2 weight decay of 10^{-6} is applied for regularization. The Adam optimizer [98] with default parameters is employed during the training phase. Additionally, we use the testing accuracy as a performance metric.

Analysis of optimized DiaNet topology. According to algorithm 2, the DiaNet1.0 topology requires at least 408 layers for classifying the MNIST dataset. Meanwhile, according to Algorithm 3, the number of layers of DiaNet2.0 is reduced to 8.8% of the DiaNet prototype. In this sense, the problem of remarkable deep and broad structure can be resolved effectively, thereby reducing the computation complexity and compressing a massive DiaNets into a specific state. Unfortunately, the accuracy is only about 20% because vanishing gradient descent exists, hampering the convergence from the beginning. As aforementioned, this problem has been largely addressed by an additional mechanism, such as a batch normalization technique. The training procedures is shown in the blue line of Fig. 19. This result demonstrates that the DiaNet2.0 topology with 36 layers can converge, but its learning slow-down and accuracy quickly saturates; compared to the baseline model, its accuracy loss is about 3%. Moreover, these additional optimization mechanisms not only suffer from expensive computation but also damage the original topology. Hence, we perform the MNIST experiments based on DiaNet3.0, and the training procedure is shown in the red line of Fig. 19. Compared with DiaNet2.0 topology and its optimization scheme using batch normalization, DiaNet3.0 not only avoids the problem of gradient vanishing but also solves the problem of learning slow down and degradation. We further explore the performance of DiaNet3.0 with an optimized batch normalization scheme. We find that the performance of DiaNet3.0 with the optimized scheme and DiaNet3.0 topology are almost the same. The DiaNet3.0 reduces batch normalization parameters, which becomes a primary reason to use the DiaNet3.0 topology for performance improvement.

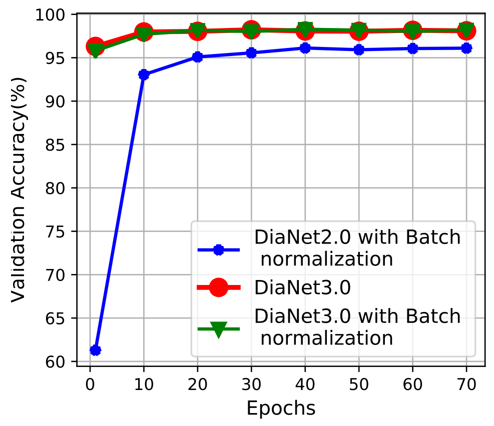


Figure 19. Comparison of MNIST classification accuracy as a function of different optimized schemes

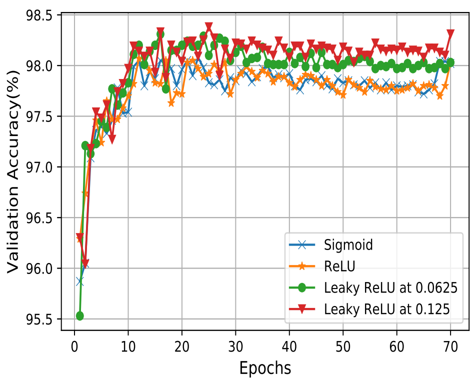


Figure 20. Validation accuracy during training phase on the MNIST dataset, for various activation functions.

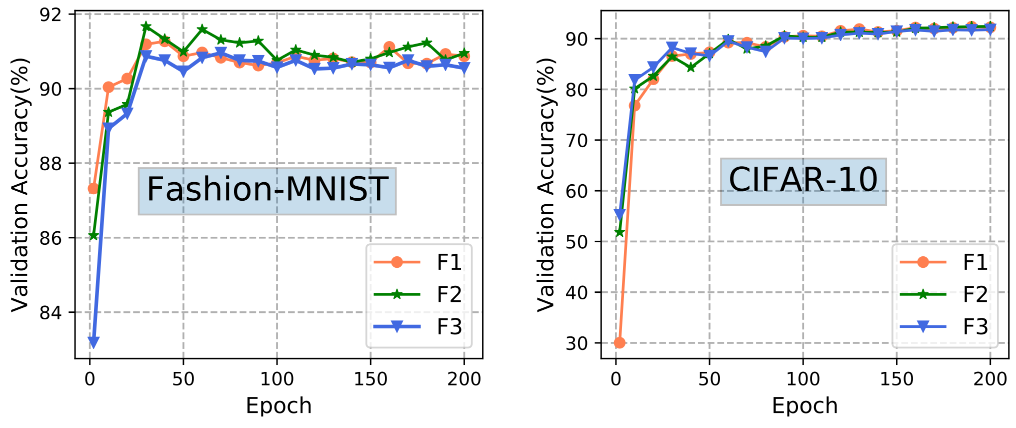
Effect of activation function. Although the parameters of proposed model are reduced compared to other SNN topologies, the number of neurons has increased. This means that the effect of activation function and bias on DiaNet topology is increased. Thus, we investigate the influence of various activation functions. Sigmoid and ReLU activation functions are well-known. The LeakyReLU activation function [99] can be formulated as equation (26):

$$LeakyReLU(x) = \begin{cases} x, & if x \geq 0 \\ negative\_slope \times x, & otherwise \end{cases} \tag{26}$$

We employ the values 0.125 and 0.0625 for the negative_slope of LeakyReLU functions, which are feasible to implement by shifted techniques and comparators in hardware and incurs negligible hardware costs. Figure 20 shows the validation accuracy of various activation functions. The LeakyReLU function at 0.125 performs the best, and the sigmoid function performs the worst, in terms of accuracy, which is due to the LeakyReLU function at 0.125 makes full use of input features and sigmoid function has saturated neurons problem. While the ReLU function exacerbates the loss of negative part of the feature information. Besides, the sigmoid function uses the exponential function, which is inflexible to implement in hardware. This motivates us to use the LeakyReLU activation function.

Table 5. The network architecture of Fashion-MNIST and CIFAR-10 datasets

Datasets	Architecture	Mark
Fashion-MNIST	16C3-M2-32C5-M2-512-10	F1
	16C3-M2-32C5-M2-512-256-10	F2
	16C3-M2-32C5-M2-512-DiaNet-10	F3
CIFAR-10	64C3-M2-128C3-M2-256C3*2-M2-512C3*2-M2-512C3*2-M2-512-10	C1
	64C3-M2-128C3-M2-256C3*2-M2-512C3*2-M2-512C3*2-M2-512-512-512-10	C2
	64C3-M2-128C3-M2-256C3*2-M2-512C3*2-M2-512C3*2-M2-512-DiaNet-10	C3



(a) Fashion-MNIST (b) CIFAR-10

Figure 21. Validation accuracy curves of DiaNet over Fashion-MNIST and CIFAR-10 datasets

Application of DiaNet. To further analyze the capacity of DiaNet3.0 with the LeakyReLU function, we perform the experiments on Fashion-MNIST and CIFAR-10. Similar to the MNIST experiment, these experiments also use CNN for feature extraction and DiaNet for classification. We use the following VGG-like architectures, which is shown in Tab. 5.

These experiments are trained for 200 epochs using the above settings. Especially, the 0.78 is used for the DiaNet with negative_slope of LeakyReLU functions on the CIFAR10 experiment. The training procedure is shown in Fig. 21. The results demonstrate that the DiaNet can achieve near full connection precision and capacity. In summary, the DiaNet series has successfully implemented three types of applications, including non-linear approximate function (regression), convolution operations, and classifiers. Namely, the proposed DiaNet can be applied in image classification (see Fig.21), biomedical classification (see Tab. 4), and regression. To some extent, since these applications have limited labels, the corresponding DiaNet architecture with appropriate depth can be generated and successfully trained. By contrast, the DiaNet model is hard to execute text/video tasks with one hundred of the label because the optimized scheme only focuses on input integration.

4.3.4 Comparison Results

The proposed DiaNet topologies are compared to other state-of-the-art works, including sparse NNs, pruning NNs and pre-defined sparse NNs, in terms of accuracy and computation cost. Table 6 demonstrates the performance comparison of different DiaNet topologies over the Wine dataset. Along with the evolution from DiaNet1.0 to DiaNet3.0, the network is deeply compressed but the accuracy is improved. From the training algorithms of DiaNets mentioned above, the essence of our proposed topology is eliminating (almost) all of ineffective synapses and neurons. Theoretically, DiaNets have a potential to achieve similar even beyond accuracy of conventional FC-NNs with compressed parameters. However, all versions of DiaNets lead to the depth-explosion, which damage the inference accuracy due to the gradient vanish problem. By integrating the input synapses (addressing DiaNet2.0) and attaching one additional skip connection to each neuron (addressing DiaNet3.0), the effects of all input can efficiently sink to output. As a result, the depth of proposed DiaNets is reduced along with the evolution.

On the other hand, we compare the characteristic of DiaNet3.0 topology and other sparse NNs on the MNIST dataset, including the testing accuracy, the number of synapses, and the reduction of parameters, as shown in Tab. 7. Compared with the baseline LeNet5 model, this work can achieve 90.86% parameter reductions with a negligible loss in accuracy. Moreover, this work achieves favorable results when compared to the other sparse NN typologies. The proposed topology

Table 6. Performance comparison of different NN topologies on Wine dataset

Index	FC-NN	DiaNet1.0	DiaNet2.0	DiaNet3.0
Synapse	160	156	65	65
Parameters	173	273	96	96
Accuracy	96.30%	88.89%	98.15%	98.15%
Configuration cost	N/A	$\rightarrow$	$\nearrow$	$\nearrow$

Table 7. Comparison with other sparse neural network topologies on MNIST dataset

Method	Accuracy	Synapse	Parameters	Parameters reduction (%)
Baseline model (LeNet5) [41]	99.20%	61376	61654	-
Architecture learning [100]	99.04%	-	40.9K	33.66%
Pruning [64]	99.23%	-	36K	41.61%
Sparsely-connected [34]	96.84%	-	8961	85.47%
Sparse NN [33]	99.19%	-	18K	70.80%
Pre-defined sparse NN*[67]	97.2 %	21000	22110	64.14%
	93.3%	2000	3110	94.96%
This work	98.41%	4660	5642	90.86%

* The architecture of this pre-defined sparse NN is $N_{net} = (800, 100, 100, 100, 10)$, and its out degree is (20,20,20,10) and (1,2,2,10), respectively.

has additional benefits compared to its counterparts: DiaNet is a sparse neural network with regularity, symmetry, and predefined properties, which leads to a decrease of memory complexity.

Upgrade of DiaNet series. The generations of DiaNets can be simply upgraded by symmetrically attaching additional connections to each PEs. It is an obvious but endless trade-off between pre-/post-reconfiguration costs and inference quality. However, the cost-quality balance lies on the principle that redundancy should be minimized while the entire array is assumed to be fully partitioned. In this sense, the bisection-based architecture is the best effort to fit above. The I/O integration technology helps to shrink the estate of a specific DiaNet piece; then, reduce the partitioning edges to evaporate the redundancy in further. From this aspect, DiaNet2.0 offers the optimal cost-quality balance. However, the quality-greedy users are able to trade additional hardware resource for slight improvement on the inference accuracy. DiaNet3.0 indicates a potential strategy: additional skip-connection without weight (neither parameters nor multiplier is needed, namely) can be symmetrically attached to each PE. From the NN theory, the synaptic feedback or skip propagation greatly help to solve linear inseparable problems. The DiaNet3.0-like upgrading strategy appears higher efficiency than reverting to the FC fashion from bisection fashion. The comparisons in Tab. 7 show some evidences. The inference accuracy of MNIST by DiaNet3.0 is higher than most of sparse NNs with fewer parameters. The upgrade strategy by adding skip connections is obviously more efficient than that of reverting to FC fashion. On the other hand, the exploration of upgrades is still an open challenge for huge networks and accuracy-greedy applications, which is out of the scope of this discussion.

4.4 Discussion on How to Co-design of Algorithm and Hardware for DiaNet

Our goal is to develop an algorithm and hardware co-design framework to improve throughput and energy efficiency. DiaNet topology is an elastic neural network with scalable bisection spanning, supporting multi-grained re-configurable from the hardware perspective. Figure 22 shows the software-hardware co-design framework for the proposed DiaNet, which mainly consists of the GPU implementation part and the Vivado implementation part. Specifically, the framework will generate the DiaNet topology based on application inputs and labels, and algorithm 3, and obtain a trained DiaNet model along with parameters after training, which executes on GPU platform (python). From the hardware perspective, the framework will partition the PE array to the corresponding area according to the DiaNet topology and reconfigure and allocate the memory and resource to generate accelerator, which is implemented by system C/ Verilog code. Finally, the framework will construct the accelerator-based hardware used to accelerate

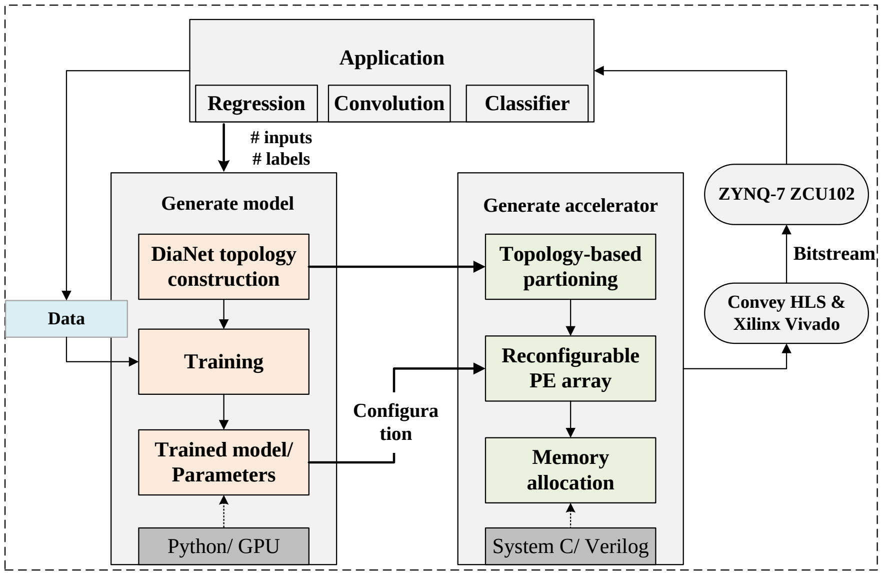


Figure 22. Software-hardware co-design framework for proposed DiaNet

real-life applications.

As aforementioned, during partitioning and reconfiguration, DiaNet does not destroy the structure but significantly reduce the resource redundancy. Moreover, the scale of PE array on-chip can be conveniently and indefinitely extended by combining multiple chips because of this property. It is obvious that the performance of this multi-grained re-configurable depends on the amount of PEs in BNN. Fortunately, the ultra-large-scale of BNN is feasible since only local connections are necessary. Thus, the critical limitation of capacity lies in the implementation cost of PEs. As the immediate solution of software, the use of inaccurate calculation units and bit-width consideration is expected to suggest the guideline for hardware design.

Here, we perform the experiments to analyze the behavior of the Ionosphere and Waveform datasets with DiaNet3.0 topology, firstly. According to algorithm 3, the DiaNet3.0 topology of Ionosphere and Waveform are built with 44 PEs and 73 PEs, respectively. Additionally, the remaining PE arrays support other applications. Theoretically, the PE array can be partitioned into multi-task with slight redundancy, as shown in Fig. 15. According to task requirements, automatically finding the optimization of PE array partitions will be an interesting

Table 8. Performance comparison between DiaNet3.0 topology and other models over two datasets from the UCI database

Dataset	Ionosphere			Waveform database generator		
	Accuracy	Parameters	Parameter reduction (%)	Accuracy	Parameters	Parameters reduction (%)
FC-NN*	95.28%	742	-	84.60%	443	-
N2PS[101]	94.9%	-	-	85.5%	-	-
This work	95.28%	156	78.98%	85%	245	44.70%

* The optimal baseline of FC-NN

exploring issue. Table 8 demonstrates that DiaNet3.0 topology achieves significant parameter reduction without quality loss compared to conventional FC-NN and N2PS model [101]. Then, the sensitivity to computational precision is investigated for DiaNets. Assuming the random noise in computation distributes into each calculates operation in DiaNets such as synapses and neurons, the noise with various bit-length is formulated as follows:

$$y_{noise} = \frac{1}{2^{bit-length}} \times \frac{1}{\sqrt{2\pi}} \times e^{\frac{x^2}{-2}} \tag{27}$$

where x is a random value, the expectation and variation of Gaussian distribution are set as 0 and 1. The error rate of inferences decreases along with the decay of bit-precision. Figure 23 shows that the impacts of bit-width variations on calculate precision in pattern recognition. For most of NN implementations, the synapse-neuron interactions are conducted by a non-linear activation function outside and a MAC operation inside. The long MACs for conventional FC fashion result in a large range range of function values. Therefore, simply reducing the global bit-width is hardly feasibly for shrinking the hardware implementations of conventional FC-NNs. By using the proposed DiaNet topology, the length of MAC is reduced to two, three, or four anywhere with a smaller range of MAC function values. Thus, fewer bits implementation is feasible for greatly reducing the hardware cost.

Finally, the bit-width consideration is investigated for optimizing on-chip inference and hardware resource saving. We employ parameter constraint technique (constraint the parameters to $[0, 1]$) (the performance as shown in Tab. 8) and collect the output of each PE for avoiding the diminishing returns. According to Eq. 25, the absolute value of each PE output is statisticized in Fig .24, which shows that the minimum and maximum absolute value of the PE of the Ionosphere are close to 2^{-8} and 2^{3}, respectively; namely, 8 bits and 3 bits for fractional bits and integer bits, respectively. Similarly, the bit-widths for the Waveform dataset are 9 bits fractional and 6 bits integer. Combining Fig. 23, the predicted bit-widths are sufficient to implement DiaNet without additional quality loss. However, to implement multiple independent DiaNets (relatively small tasks) on one chip in parallel, the performed tasks must ensure the same bit-width configuration. In this sense, 16 bits configuration is predicted to execute two tasks on one chip.

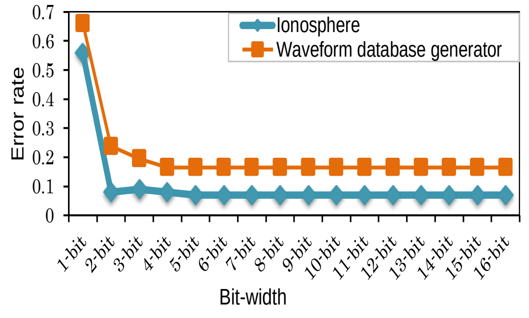


Figure 23. Error rate over various bit-length for two datasets from the UCI database

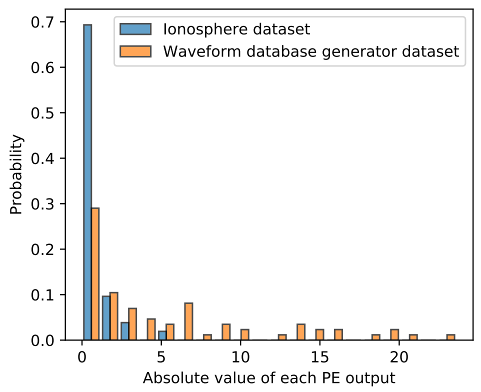


Figure 24. Histograms of absolute values of each PE output over two datasets from UCI database

4.5 Towards Hardware Implementation of DiaNet

In this section, we use proposed architecture to verify the effectiveness of DiaNet topology and the feasibility of multi-core parallel processing.

In spatially expanded in parallel architecture architecture [102], the weight parameters are stored in synapses and executed synaptic operations without off-chip accessing [103, 104]. The model of this architecture is an elastic neural

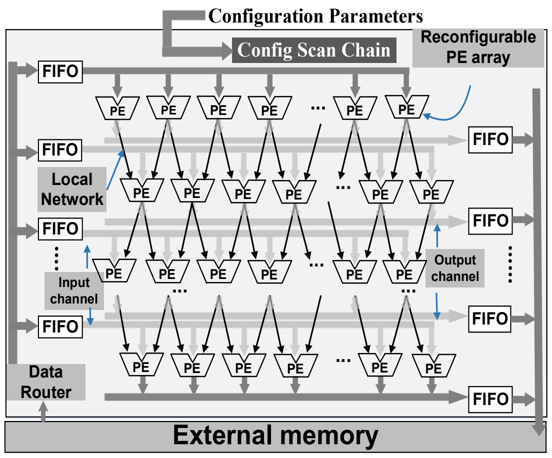


Figure 25. Architecture overview

network model, which enables fully parallel and efficient dynamically composable to improve the energy efficiency [105]. The BNN is mapped to PE on the hardware instead of time-multiplexed PEs for different synaptic connections, data flow to output continuously, and parallelly. Inspired by our previous multi-grained re-configurable accelerator for approximate computing [106], we design a neuromorphic architecture with 20×20 PE arrays to verify the performance of DiaNet topology for pattern recognition. The overview of the architecture, as shown in Fig. 25, is composed of PE arrays, input/output channel, FIFO, config scan chain, and external memory. Each PE can access FIFOs via input/output channel and communicate with adjacent PEs through local network. The inference process of independent DiaNet consists of four steps: first of all, the accelerator loads the configuration parameters into config scan chain serially to partition and reconfigure the entire PE array; secondly, all input data are loaded from the external memory to the configured PE array by the local input controller and data router mechanism through the input FIFO; thirdly, each PE can start processing as soon as data arrives instead of proceeding as a systolic array. Finally, the outputs are written back to external memory via output FIFO to find the label corresponding to the maximum value (implemented by comparators), thereby realizing pattern recognition.

As aforementioned, the experiment results show that the proposed DiaNet topology can achieve 78.98% and 44.70% parameter reduction in the Ionosphere

Table 9. Accuracy and bit-width comparison among various strategies for FPGA simulation over two datasets

Dataset	Method	Accuracy (%)	Bit width
Ionosphere	Hierarchical [107]*	94.28%	12 bits
Ionosphere	**This work**	93.10%	12 bits
Waveform	**This work**	83.47%	16 bits
Ionosphere + Waveform	**This work**	Without accuracy loss	16 bits

and Waveform dataset, respectively. The computational complexity and hardware cost of these topology decreased significantly compared to the conventional FC-NN model. For proof of feasibility, we perform the Ionosphere and Waveform dataset with DiaNet3.0 topology on the proposed neuromorphic architecture. The architecture design is implemented with RTL, which is synthesized in Vivado (v2018.3) based on Xilinx FPGA ZYNQ-7 ZCU102. The 20 $\times$ 20 PE arrays are symmetrically implemented by the FPGA, supporting on-demand array partitioning and re-configuring for processing different DiaNets in parallel. In this manner, 44 PEs and 73 PEs are partitioned to perform the mentioned tasks, respectively. Additionally, the remaining PE arrays support other DiaNet tasks. Moreover, according to the Subsection. 4.4, the Ionosphere and Waveform dataset are executed on FPGA with 12 bit-widths and 16 bit-widths, respectively. The results are shown in Tab. 10, compared to Hierarchical FPGA implementation, this work achieves 10.8% power reduction with 1% accuracy loss. Additionally, to implement these two independent DiaNets on one chip in parallel, the executed tasks must ensure the same bit-width configuration. Namely, Ionosphere and Waveform applications are processed paralelly with 16 bits-width on the FPGA. Table 9 and Tab. 10 show that two independent tasks can run in parallel on the same PE array without any accuracy loss. Moreover, the proposed multi-grained re-configurable spatially expanded architecture, which supports multi-core parallel processing instead of single-core processing to save multiple configuration costs and time.

Table 10. Power and resource utilization comparison among various strategies for FPGA simulation over two datasets

Dataset	Method	Utilization (LUT)	Frequency (MHz)	Power (W)	Re-configurable
Ionosphere	Hierarchical [107]*	3920	-	0.37	temporal architecture
Ionosphere	**This work**	4342	100	0.33	spatial architecture
Waveform	**This work**	8851	100	0.55	spatial architecture
Ionosphere + Waveform	**This work**	14187	100	0.88	spatial architecture

* This work implemented on Virtex-7 FPGA.

4.6 Conclusion

In this section, an elastic neural network is developed and evolved towards effectively re-configurable hardware in fully parallel on-chip. We proposed various evolution techniques of DiaNet to prevent the depth explosion and gradient vanishing problem, including I/O layer integration and skip connection. Experiments proved the effectiveness of our method for various applications. The number of layers is reduced to 8.8% of the DiaNet prototype for MNIST recognition while improving accuracy to 98.41%. Moreover, compared with the LeNet5 model as state-of-art, the evolved DiaNet topology achieves the parameter reduction of 90.86% with the negligible loss of accuracy. As a co-design of algorithm and hardware, this work investigated inaccuracy tolerance and bit-width consideration to suggest the guideline for efficient hardware implementations. Based on that, the effectiveness of DiaNet is verified by the proposed re-configurable architecture on FPGA with the power reduction of 10.8% compared to state-of-the-art implementations.

5 Temporal-spatial Combined Bisection Neural Network

As aforementioned, the DiaNet topology minimizes computation cost by reducing the number of multiplication operations and the model's size. Moreover, the DiaNet takes full advantage of network-on-chip (NoC) resources for highly parallel and efficient dynamically composable to improve energy efficiency further [108]. However, it still suffers from the overhead of synaptic weight storage and poor robustness performance (e.g., hardware variation). Simultaneously, SNNs have great promise in continuous spatial-temporal information processing with lower energy consumption and better robustness. To this end, we develop a temporal-spatial combined bisection neural network architecture based on spatial-temporal dynamic-based neuron and spatially expanded parallel architecture. Specifically, a spatial-temporal dynamic-based neuron with bisection connection is developed instead of a traditional neuron (e.g., ReLU, sigmoid) in spatially expanded-based DiaNet topology, so that the network becomes linearly separable while processing information in temporal and spatial domains, better yet, reducing the memory bandwidth and hardware overhead and improving the robustness. To this end, the temporal-spatial combined bisection neural network will be explored in the chapter.

From the architecture point of view, in spatial domain, DiaNet takes advantage of parallel computing resources and spatial complexity to perform various computations. Besides, in temporal domain, SNN makes full use of temporal information to perform computation, including time-based spike coding and time-based neuron dynamics and time-based adjustable precision. In this sense, the temporal-spatial combination provides a more flexible and powerful implementation framework for various NN applications. From the model's complexity point of view, DiaNet can minimize computational and memory requirements because of ultra-sparse synaptic connection. SNN may easily enable low-power computational overhead due to the event-driven computing paradigm. Therefore, the temporal-spatial combined NN architecture may leads to lower complexity of model.

Nevertheless, it is challenging to directly train the SNNs with binary spike activity of neurons and ultra-sparse synaptic connection with satisfactory performance. We describe two issues to be solved for directly training ultra-sparse SNNs:

• Gradient vanishing problem. Because of binary spike activity and the non-differentiable of spiking neurons, gradient propagation tends to vanish when training very deep SNN directly [58]. On the other hand, due to the inherent connection of neurons in DiaNet topology, the depth remains a challenge.

• Information vanishing problem. Due to the firing mechanism of the spiking neuron, the firing rate depends on the input membrane potential and threshold. Specifically, when the input potential is smaller than the threshold, its neuron will not emit a spike, and the neuronal potential remains. This issue can be addressed by accumulating the membrane potential with large timesteps or more synaptic connections. Besides, DiaNet has inherent bisection connections, and a large timestep will cause significant latency and additional computational cost.

5.1 Exploration of Temporal-spatial Combined Bisection Neural Network

In this section, we discuss the temporal-spatial combined bisection neural network architecture in detail. Firstly, the iterative LIF neurons are introduced into DiaNet topology. Then, the details of the proposed temporal-spatial combined NN architecture are presented. Finally, we provide the overall training algorithm.

5.1.1 Iterative Leaky Integrate-And-Fire Model in DiaNet

As aforementioned in section 2.1, the SNN is composed of neurons interconnected through synapses, which receives the input spikes. The spiking neuron defines the neuronal dynamics and firing mechanism, and its differential equation can be formulated as:

$$\tau \frac{du(t)}{dt} = -u(t) + I(t) \tag{28}$$

where $u(t)$ is membrane potential at time t, τ denotes a constant decay parameters and $I(t)$ represents inputs from synaptic connections from last layer. In order to take full advantage of spatial-temporal information, the iterative LIF neuron was proposed [59], the Eq. (28) can be reformulated as :

$$u_i^l(t+1) = \tau u_i^l(t)\left(1 - o_i^l(t)\right) + \sum_j W_{ij}^{l-1} \cdot o_j^{l-1}(t+1) + b \tag{29}$$

The real firing mechanism is similar to Eq. (7), which can be described as:

$$o_i^l(t+1) = f(u_i^l(t+1) - V_{th}) \tag{30}$$

where $f(x)$ is step function, which satisfies $f(x) = 1$ when $x > 0$, and vice versa. If neuron emits a spike ($o_i^l(t+1) = 1$) at timestep $t+1$, the membrane potential will be set to 0 at timestep $t+1$ via $(1 - o_i^l(t+1))$; on the contrary, the membrane potential will remain its value.

Subsequently, we introduce the LIF neuron to DiaNet1.0 topology. To this end, in spatial domain, the LIF neuron only communicates with two synapses from its previous neurons in adjacent, and fans the data out to two neurons in the post layer. Besides, in temporal domain, the LIF neuron accumulates the inputs from previous timesteps. In that vein, the forward propagation of DiaNet topology with LIF neurons in temporal-spatial domain can be reformulated as:

$$u_i^l(t+1) = \tau u_i^l(t)\left(1 - o_i^l(t)\right) + W_{ii}^{l-1} \cdot o_i^{l-1}(t+1) + W_{i(i+1)}^{l-1} \cdot o_{i+1}^{l-1}(t+1) + b \tag{31}$$

The forward propagation dataflow of DiaNet topology with LIF neurons in the temporal-spatial domain is shown in Fig. 26. In the spatial field, the forward propagation in the layer-by-layer like ANN. In addition, each neuron integrates the membrane potential from previous timesteps by self-feedback in the temporal domain.

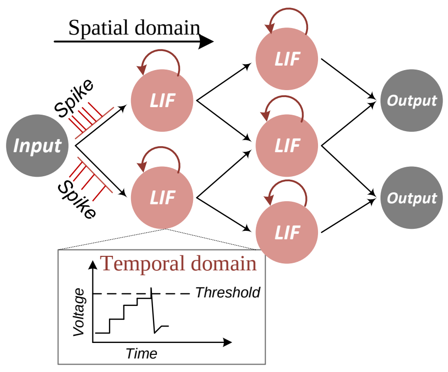


Figure 26. Illustration of the spatial-temporal characteristic of DiaNet1.0 with LIF neuron

However, due to the binary spike activity of LIF neuron and the inherent bisection connection of DiaNet topology, its neuron takes a long time to accumulate the membrane potential that exceeds the threshold. To some extent, the firing rate are depend on the input membrane potential and timesteps. Yet large timesteps not only causes expensive computational costs, but also comes at the cost of large inference latency. In this section, we reasonably increase the intensity of the input membrane potential based on the DiaNet paradigm.

5.1.2 Combined Bisection Neural Network Architecture

DiaNet2.0 and DiaNet3.0 typologies are evolved for depth reduction and accuracy improvement, respectively. We introduce the SNN paradigm with LIF neuron into DiaNet3.0 topology to construct combined NN architecture for better accuracy and timesteps, seen as DiaNet4.0 topology. As aforementioned in Eq. (25) of section 4, the addition operations are processed after activation function, which is inspired from ResNet with ReLU before addition structure. Thus, By incorporating the Eq. (29) and Eq. (31), the Eq. (25) can be reformulated as below:

$$u_i^l(t+1) = \begin{cases} \tau u_i^l(t)\left(1-o_i^l(t)\right) + W_{i-1}^{l-1}\cdot\left(o_{i-1}^{l-1}(t+1) + o_{i-1}^{l-2'}(t+1)\right) + \\ \qquad W_i^{l-1}\cdot\left(o_i^{l-1}(t+1) + o_i^{l-2'}(t+1)\right) + W^{l-1}x_i + b, \quad l \in l_e \\ \tau u_i^l(t)\left(1-o_i^l(t)\right) + W_{i+1}^{l-1}\left(o_{i+1}^{l-1}(t+1) + o_{i+1}^{l-2'}(t+1)\right) + \\ \qquad W_i^{l-1}\cdot\left(o_i^{l-1}(t+1) + o_i^{l-2'}(t+1)\right) + W^{l-1}x_i + b, \quad l \in l_s \end{cases} \tag{32}$$

where l_e and l_s are expanded layers and shrinkage layers of DiaNet topology, respectively. Here, $u_i^l(t+1)$ represents the membrane potential of $i-th$ neuron in $l-th$ layer at time t, x denotes the input spikes after Poisson rate coding, which are binary spikes. Additionally, the first term on the right side in Eq. (32) is leakage in membrane potential, which is inherited from the previous timesteps. The fourth item directly receives the input features rather than activation outputs. The first and fourth terms on the right side of the Eq. (32) in the spatial-temporal NN architecture are easy to obtain and understand. The second and the third

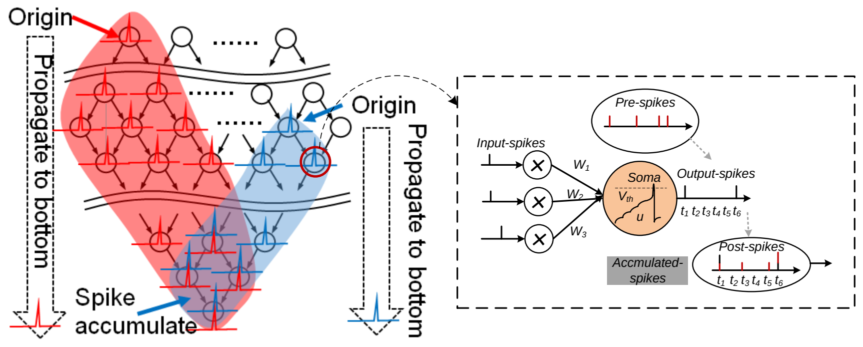


Figure 27. Illustration of accumulated spikes in temporal-spatial combined NN architecture

item represent membrane potentials from its previous neurons via two synaptic connections in the $(l-1)-th$ layer. Herein, $o_{i-1}^{l-1}(t+1)+o_{i-1}^{l-2'}(t+1)$ represents that the neuron not only has its firing-spike after $f(x)$ step function in $(l-1)th$ layer, but also accumulated the spike from $(l-2)-th$ via addition after activation function. Besides, $o_{i-1}^{l-2'}(t+1)$ directly inherits the binary spike from the previous layers through skip connection, which can alleviate the information vanishing problem and gradient vanishing problem through the spikes accumulation.

DiaNet4.0 topology achieves better timesteps and ultra-sparse connections at the expense of the event-driven spike communication paradigm but still benefits from spatial-temporal information processing. In essence, the DiaNet4.0 topology enhances the spike signal by adding skip connections (seen as vertical density) instead of spike accumulation by synaptic connections and more enormous timesteps. In this sense, DiaNet4.0 can alleviate the information loss caused by binary activities and ensure sufficient information flows through the entire network. Figure 27 schematically illustrates the flow and accumulation of spikes in the forward propagation process of the entire network.

Nevertheless, the inputs of each neuron in hidden layer are not events-driven paradigm, but a multi spike paradigm. Due to one layer of skip connection, the inputs of each neuron in $l-th$ hidden layer can be formulated as:

$$o_{i-1}^{l-2'}(t+1)=\begin{cases} o_{i-1}^{l-2}(t+1)+o_{i-1}^{l-4}(t+1)+...+o_{i-1}^{1}(t+1), & l_{th}\in Odd \\ o_{i-1}^{l-2}(t+1)+o_{i-1}^{l-4}(t+1)+...+o_{i-1}^{2}(t+1), & l_{th}\in Even \end{cases} \quad (33)$$

Here, $o_{i-1}^{l-2'}(t+1)$ is a positive integer. When $l_{th} \in Even$, it is smaller than $\frac{l_{th}}{2} - 1$, and otherwise smaller than $\frac{l_{th}-1}{2}$. To some extent, we quantize the inputs of each neurons to reduce the computation and memory cost, while maintains the spatial-temporal information processing. Naturally, if implement the DiaNet4.0 on digital circuit, the unrolled and tiled LUTs can be introduced to implement our DiaNet4.0 architecture replace the MAC operations.

5.1.3 Overall Training Framework

In this section, we use the Poisson rate coding [109] as a coding scheme to reduce the sampling error when converting real-value input to spike x. We employ the Poisson processes to map the normalized pixel intensity p to the spike in a stochastic manner and input the spike to the corresponding neuron. Specifically, at each time-steps of the combined NN forward propagation process, a uniform stochastic number between 0 and 1 is generated and compared with the pixel intensity p. As described in Eq. (34), if the p is greater or equal than the stochastic number, the spike will be emitted; otherwise, no spike will be produced. Herein, the spike always has the same sign as p, and the total number of spikes in t time steps is proportional to the amplitude $|p|$.

$$x = \begin{cases} \operatorname{sign}(p), & \text{if } |p| \geq s \\ -\operatorname{sign}(p), & \text{if } |p| \geq s \\ 0, & \text{if } |p| < s \end{cases} \tag{34}$$

On the other hand, to compute gradients by leveraging the temporal and spatial information, we employ spatio-temporal backpropagation (STBP) learning rule [59] and cross-entropy loss function L to directly train the overall DiaNet4.0 model. Moreover, in order to reduces the information loss caused by the firing mechanism of the last layer, the output layer only accumulates the inputs via synaptic weight and overall timesteps but do not emits the output spike. The weight and bias update is computed as:

$$\Delta W_{ij}^{l} = \sum_{i=1}^{3} \frac{\partial L}{\partial W_{ij}^{l}} = \sum_{i=1}^{3} \frac{\partial L}{\partial o_i^l(t)} \frac{\partial o_i^l(t)}{\partial u_i^l(t)} \frac{\partial u_i^l(t)}{\partial w_{ij}^l} \tag{35}$$

$$\Delta b^{l} = \sum_{i=1}^{3} \frac{\partial L}{\partial b^{l}} = \sum_{i=1}^{3} \frac{\partial L}{\partial o_i^l(t)} \frac{\partial o_i^l(t)}{\partial u_i^l(t)} \frac{\partial u_i^l(t)}{\partial b^l} \tag{36}$$

wherein, we introduce the surrogate gradient to approximate real gradient [57], which is described as below:

$$\frac{\partial o}{\partial u} = \frac{1}{\left(\sqrt{2a\pi}\right)^2} e^{-\frac{(u-V_{th})^2}{2a^2}} \tag{37}$$

where a is 0.6.

5.2 Overall Estimation of Performance

In this section, we perform the experiments to estimate the overall performance from three aspects: accuracy, parameters reduction and inference latency.

5.2.1 Datasets and Implementation

The static vision dataset (MNIST) and dynamic vision dataset (N-MNIST) are employed to evaluate the performance of temporal-spatial combined NN architecture. MNIST is a handwritten digits image with 50000 training data, 10000 validation data and 10000 testing data. N-MNIST is a frame-based neurromorphic dataset, which is captured from a moving sensor viewing MNIST dataset moving on a computer monitor from two channels. The N-MNIST uses the same sample and testing-training splits as MNIST.

We perform the experiment on the MNIST and N-MNIST dataset to analyze the performance of combined NN architecture. Similar to the LeNet5 topology and DiaNet topology, we setup the experiments consist of spiking convolution blocks for extracting features and combined NN architecture topology for classification. Each spiking convolution block composes of convolution layer, max-pool layer, batch normalization layer and conventional LIF neuron. The features from spiking convolution layer are flattened into a 400 vector. Namely, combined NN architecture classes these 400 features and 10 labels. All experiments are trained for 100 epochs with 100 batch size. Besides, the learning rate is set to 0.1 initially and is lower five times at epoch 20 and the Adam optimizer with default parameters and L2 weight decay of 10^{-6} are applied for our experiment. To this end, the network architecture and the hyper parameter for MNIST and N-MNIST experiments in this work are listed in Tab. 11. In addition, all experiments are implemented on PyTorch framework.

Table 11. The network architecture and hyper-parameters of MNIST and N-MNIST

Dataset	Network architecture	Timesteps*	Decay constant	Potential threshold
MNIST	6C3-M2-16C5-M2-DiaNet	8	0.8	0.3
N-MNIST*	16C5-M2-16C5-M2-16C5-DiaNet	6	0.95	0.1

5.2.2 Experimental Results

The complexity, accuracy, and inference latency of the model behave as a trade-off. In this sense, We analyze the results from three aspects: compression ratios of synaptic connection, time steps, and accuracy. Firstly, we analyze the relationship between accuracy and time steps. In a temporal-based model, time step T refers to repeating the inference process T times to accumulate the final membrane potential of the output layer. In this manner, the computation cost of the entire network is denoted as $\mathcal{O}(T)$. Namely, we can reduce the computational overhead by minimizing the time steps. As shown in Fig. 28 and Fig. 29, we can achieve satisfactory accuracy with a few time steps. For example, we can achieve 96.10% accuracy with 8 time steps on MNIST and 98.15% accuracy with 6 time steps on N-MNIST.

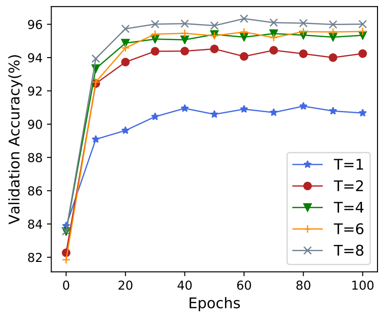


Figure 28. Influence of simulation timesteps on MNIST

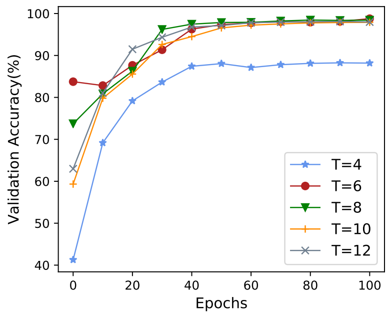


Figure 29. Influence of simulation timesteps on N-MNIST

Table 12. Performance comparison of DiaNet3.0 and DiaNet4.0 models on MNIST

Method	Domain	Accuracy	Time steps	Activation	Towards hardware implementation
DiaNet3.0	Spatial	98.10%	1	Float	Fully parallel
DiaNet4.0	Temporal+spatial	96.10%	8	4 bits	Fully parallel

The temporal-spatial combined NN architecture also takes full advantage of highly parallel and efficient dynamically composable to improve energy efficiency. As shown in Tab. 12, compared to spatial-based DiaNet3.0 architecture, DiaNet4.0 further reduces the complexity of the model by reducing the precision of operations with a 1.57% accuracy loss. Moreover, the experimental results on NMIST demonstrate that the DiaNet4.0 has better capacity and robustness of the model than the DiaNet3.0 topology.

On the other hand, we analyze the results of compression ratios of parameters, accuracy, and timesteps and compare the experimental results with other state-of-the-art compressed SNN models. Table. 13 and Tab. 14 present the current state-of-the-art results on MNIST and N-MIST. Herein, Lei Deng et al. [76] achieved 96.84% accuracy and 96.83% with 75.00% compression rate and 10 timesteps on MNIST and NMNIST, respectively. Besides, our model can achieve 96.10% with a 90.86% compression ratio and 8 timesteps; and we accomplish 98.15% accuracy with 69.38 parameters reduction with 6 timesteps. In a nutshell, the experimental results indicated that compared with full precision SNN architecture and DiaNet prototype, the temporal-spatial combined NN architecture can achieve the lower complexity with slight accuracy loss.

Table 13. Network structures and training methods of various compressed SNN methods

Dataset		Method	Architecture
MNIST	IJCNN'18 [75]	Multi-strength	32C5-64C5-1024-10 (MP)
	Front. Neurosci'19 [74]	STDP	784-398-500-10
	TNNLS'21 [76]*	STBP	128C3-128C3-512-10(AP)
	Our work	STBP	6C5-16C3-DiaNet-10 (MP)
NMNIST	TNNLS'21 [76]*	STBP	128C3-128C3-512-10(AP)
	Our work	STBP	16C5-16C5-16C5-DiaNet-10 (MP)

MP and AP represent max-pool and average-pool, respectively

Table 14. Performance comparison between DiaNet4.0 topology with other compressed SNN models

Dataset		Accuracy	Parameter reductions	Time steps
MNIST	IJCNN'18 [75]	94.00%	89.43%	80
	Front. Neurosci'19 [74]	97.05%	75.00%	50
	TNNLS'21 [76]*	96.84%	75.00%	10
	Our work	96.10%	90.86%	8
NMNIST	TNNLS'21 [76]*	96.83%	75.00%	10
	Our work	98.15%	69.36%	6

* The first and last layers are still full connection fashion.

5.3 Conclusion

This chapter explores the temporal-spatial combined bisection neural network architecture for further reducing the computational and memory cost of spatially expanded-based DiaNet by introducing a low-power spike coding. We present the skip-connection and accumulated paradigm to iterative LIF neuron with bisection synaptic connection and employ surrogate gradient-based learning rule to solve the problem of gradient vanishing and information vanishing jointly. Finally, the model is evaluated on both static MNIST and neuromorphic N-MNIS datasets.

Finally, the model achieves 96.10% accuracy with a 90.86% compression ratio with 8 time steps on MNIST and 98.15% accuracy with 69.38% parameters reduction with 6 time steps on N-MNIST. Our work estimate much better performance, including versatility and complexity of the model, than prior work and spatial-based NN architecture of this thesis. Besides, the temporal-spatial architecture takes full advantage of highly parallel and efficient dynamically composable to improve energy efficiency compared to temporal-based SNN with FC fashion.

6 Conclusion and Future Work

6.1 Conclusion

Deep neural networks have demonstrated state-of-the-art performances in the broad field of AI applications. However, these algorithms have intensive computational and colossal memory, making it challenging to develop on hardware platforms with limited computational resources. To address this challenge, this thesis focuses on constructing efficient elastic neural network architectures with spike coding and scalable bisection spanning to support the fully parallel and re-configurable NN platforms.

This dissertation focuses on constructing elastic neural network with spike coding and scalable bisection spanning to make NN inference less complicated, less redundancy and make it more efficient from from three aspects: spike coding-based ternary spiking neural network, scalable bisection spanning-based DiaNet topology, and temporal-spatial combined bisection neural network. All these aspects share a common principle: constructing elastic neural network architectures with low complexity and spatial expansion architecture, as shown in Fig. 30.

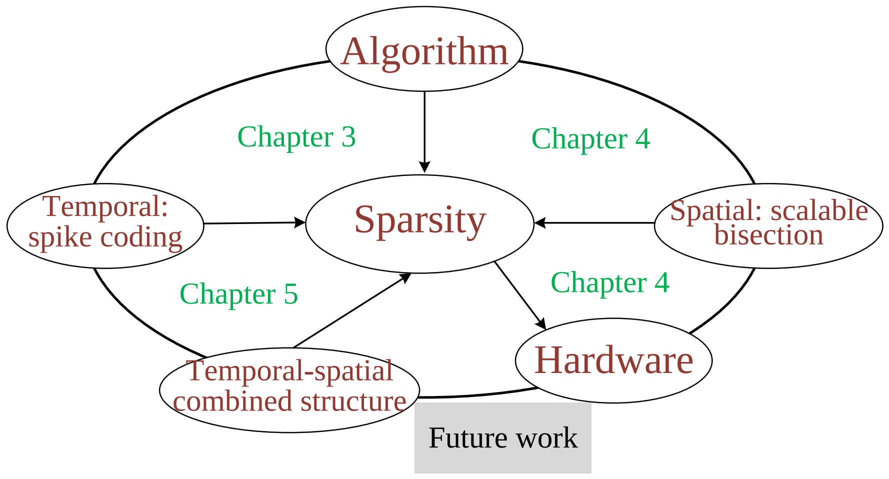


Figure 30. Summary of the thesis

Spike coding-based spiking neural network is expected to develop massively parallel and higher energy efficient inference engines. Whereas the massive amount of floating-point parameters and larger latency in SNNs still take away significant computation efficiency. To this end, we compressed SNNs with ternary weights to achieve a good trade-off between complexity, latency, and performance in Chapter 3. We propose a parametric integrate-and-fire neuron (PIF) with a learnable threshold to reduce the computation cost while maintaining the self-adaptability and expressiveness of spiking neurons. Subsequently, by introducing a paradigm similar to the ternary weight network and spike-timing-dependent backpropagation learning rule, PIF neuron-based TSNNs are directly trained on large-scale datasets with a few timesteps. Our proposal is evaluated on N-MNIST, CIFAR-10, CIFAR-100, which achieved 98.43%, 89.07%, 65.24% accuracy with 4 timesteps, respectively, and achieved up to 16x model compressions.

Since the PE and interconnection are arranged in pre-silicon, the noticeable PE and wire connections might be inactivated when the NN is reconfigurable, leading to remarkable redundancy. A spatially expanded-based bisection neural network array was developed for efficiently multi-grained reconfigurable in fully parallel on-chip. However, the original spatially expanded-based DiaNet topology is challenging to implement complex NN applications with its straightforward architecture. To this end, we proposed I/O layer integration and skipped connection into DiaNet topology to address the depth explosion and gradient vanishing problems. Experiments proved the effectiveness of our method for various applications. The layers are reduced to 8.8% of the DiaNet prototype for MNIST recognition while improving accuracy to 98.41%. Moreover, compared with the LeNet5 model as state-of-art, the evolved DiaNet topology achieves the parameter reduction of 90.86% with no accuracy loss. Besides, the effectiveness of DiaNet is verified by the proposed reconfigurable architecture on FPGA with the power reduction of 10.8% compared to state-of-the-art implementations.

DiaNet topology minimizes the computation cost by reducing the number of multiplication operations and the model's size. Moreover, the DiaNet takes full advantage of network-on-chip resources for highly parallel and efficient dynamically composable to improve energy efficiency. However, it still suffers from the overhead of floating-point operands. We explore the temporal-spatial combined bisection neural network architecture for further reducing the complexity of spatially expanded-based DiaNet by introducing a low power spiking paradigm.

We introduced the skip-connection and spike accumulation paradigm to iterative LIF neurons with bisection synaptic connection and employed surrogate gradient-based learning rule to solve the problem of gradient vanishing and information vanishing jointly. Finally, the model achieves 96.10% accuracy with a 90.86% compression ratio with 8 timesteps on MNIST and 98.15% accuracy with 69.38% parameters reduction with 6 timesteps on N-MNIST. Our work estimate much better performance, including versatility and complexity of the model, than prior work and spatial-based NN architecture of this thesis. Besides, the temporal-spatial architecture takes full advantage of highly parallel and efficient dynamically composable to improve energy efficiency compared to TSNN with FC fashion.

6.2 Future work

Although this thesis made important progress, there are still many challenges and potential directions that need to be further explored and expanded.

Spike coding-based TSNN and spatially-based DiaNet architecture provide the potential to implement neural network platforms in a highly flexible and fully parallel manner at the unit-, architecture- and topology level. There is enough space to explore hardware platforms based on these NN typologies, including compiler, in-memory computing technology and asynchronous inference engine. Firstly, the proposed multi-grained architecture can be considered as data flow-based architecture. In this manner, it is interesting to explore the efficient compilation of data flow graphs to MURA with acceptable compilation time. Besides, the efficient data-flow mapping from TSNN to pre-silicon neuromorphic hardware also a interesting challenge. Additionally, spatially expanded architecture reduces transmission cost by utilizing hierarchical memory, but it still requires significant memory cost due to low-level memory. In contrast, in-memory technology can reduce the transmission cost. Moreover, "SNN+in-memory" technology is a promising for addressing the Moore's low problem. Therefore, we will introduce the in-memory technique to the DiaNet and TSNN hardware implementation in the future.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my Ph.D. supervisor, Prof. Yasuhiko Nakashima. Prof. Nakashima is probably the most visionary, inspiring, and outstanding supervisor in my heart. I am very fortunate and grateful to receive his guidance and support in my Ph. D journey. Especially at every critical moment of my Ph. D career, he gave me visionary advice, generous support, and the most sincere and constructive feedback. His scientific rigor and endless enthusiasm for research greatly inspired me to outgrow my limits.

I would love to express my deep gratitude to associate Prof. Renyuan Zhang for his tremendous guidance and support. He patiently guided me to become an independent researcher who could define and solve scientific problems. His innovative research, endless scientific research enthusiasm, and comprehensive expertise have inspired and encouraged me consistently. I truly appreciate working with him in the past three years, and what I learn from him is beyond count.

I am sincerely grateful to Prof. Mutsumi Kimura for his tremendous help, invaluable advice, and recommendation. His scientific rigor, brilliant thinking, and technical depth have tremendously affected my attitude towards being a researcher.

Besides, I gave sincere thanks to co-advisor Prof. Yuichi Hayashi for his patient guidance and valuable advice on academic research and this thesis. I would also thank Prof. Tran Thi Hong for her encouragement. My sincere thanks are also extended to all collaborators in Computing Architecture Laboratory for their support, help, and motivation. It's a great honor to work alongside them and make progress together. Thank you all again, and looking forward to future collaborations.

I would also thanks my friends at NAIST for their encouragement and help. Additionally, I want to express my sincere appreciation to my master's mentor Prof. Guoyong Huang for his invaluable advice when I was in trouble. I also want to thanks my friends in my life for their encouragement, care, and love.

Last but not least, I would like to express my heartfelt thanks and love to my parents and family for their endless love and encouragement.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[2] Song Han, Junlong Kang, Huizi Mao, Yiming Hu, Xin Li, Yubin Li, Dongliang Xie, Hong Luo, Song Yao, Yu Wang, et al. Ese: Efficient speech recognition engine with sparse lstm on fpga. In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 75–84, 2017.

[3] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

[4] Raia Hadsell, Pierre Sermanet, Jan Ben, Ayse Erkan, Marco Scoffier, Koray Kavukcuoglu, Urs Muller, and Yann LeCun. Learning long-range vision for autonomous off-road driving. *Journal of Field Robotics*, 26(2):120–144, 2009.

[5] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(ARTICLE):2493–2537, 2011.

[6] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[7] Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt,

William Lewis, Mu Li, et al. Achieving human parity on automatic chinese to english news translation. *arXiv preprint arXiv:1803.05567*, 2018.

[8] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[9] Yang You, Zhao Zhang, Cho-Jui Hsieh, James Demmel, and Kurt Keutzer. Imagenet training in minutes. In *Proceedings of the 47th International Conference on Parallel Processing*, pages 1–10, 2018.

[10] Erik Lindholm, Mark J Kilgard, and Henry Moreton. A user-programmable vertex engine. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 149–158, 2001.

[11] Norman P Jouppi, Cliff Young, Al Patil, et al. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*, pages 1–12, 2017.

[12] William J Dally, Yatish Turakhia, and Song Han. Domain-specific hardware accelerators. *Communications of the ACM*, 63(7):48–57, 2020.

[13] Shouyi Yin, Peng Ouyang, Shibin Tang, Fengbin Tu, Xiudong Li, Shixuan Zheng, Tianyi Lu, Jiangyuan Gu, Leibo Liu, and Shaojun Wei. A high energy efficient reconfigurable hybrid neural network processor for deep learning applications. *IEEE Journal of Solid-State Circuits*, 53(4):968–982, 2017.

[14] Tianshi Chen, Zidong Du, Ninghui Sun, Jia Wang, Chengyong Wu, Yunji Chen, and Olivier Temam. Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning. *ACM SIGARCH Computer Architecture News*, 42(1):269–284, 2014.

[15] Chris Nicol. A coarse grain reconfigurable array (cgra) for statically scheduled data flow computing. *Wave Computing White Paper*, 2017.

[16] Leibo Liu, Jianfeng Zhu, Zhaoshi Li, Yanan Lu, Yangdong Deng, Jie Han, Shouyi Yin, and Shaojun Wei. A survey of coarse-grained reconfigurable

architecture and design: Taxonomy, challenges, and applications. *ACM Computing Surveys (CSUR)*, 52(6):1–39, 2019.

[17] Yu-Hsin Chen, Tien-Ju Yang, Joel Emer, and Vivienne Sze. Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(2):292–308, 2019.

[18] Qian Zhang, Ting Wang, Ye Tian, Feng Yuan, and Qiang Xu. Approxann: An approximate computing framework for artificial neural network. In *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 701–706. IEEE, 2015.

[19] Sparsh Mittal. A survey of techniques for approximate computing. *ACM Computing Surveys (CSUR)*, 48(4):1–33, 2016.

[20] Fengbin Tu, Shouyi Yin, Peng Ouyang, Leibo Liu, and Shaojun Wei. Reconfigurable architecture for neural approximation in multimedia computing. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(3):892–906, 2018.

[21] Wm A Wulf and Sally A McKee. Hitting the memory wall: Implications of the obvious. *ACM SIGARCH computer architecture news*, 23(1):20–24, 1995.

[22] Filipp Akopyan, Jun Sawada, Andrew Cassidy, Rodrigo Alvarez-Icaza, John Arthur, Paul Merolla, Nabil Imam, Yutaka Nakamura, Pallab Datta, Gi-Joon Nam, et al. Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. *IEEE transactions on computer-aided design of integrated circuits and systems*, 34(10):1537–1557, 2015.

[23] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99, 2018.

[24] Maxence Bouvier, Alexandre Valentian, Thomas Mesquida, Francois Rummens, Marina Reyboz, Elisa Vianello, and Edith Beigne. Spiking neural networks hardware implementations and challenges: A survey. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 15(2):1–35, 2019.

[25] Song Han. *Efficient methods and hardware for deep learning*. PhD thesis, Stanford University, 2017.

[26] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S Emer. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12):2295–2329, 2017.

[27] Kota Ando, Kodai Ueyoshi, Kentaro Orimo, Haruyoshi Yonekawa, Shimpei Sato, Hiroki Nakahara, Shinya Takamaeda-Yamazaki, Masayuki Ikebe, Tetsuya Asai, Tadahiro Kuroda, et al. Brein memory: A single-chip binary/ternary reconfigurable in-memory deep neural network accelerator achieving 1.4 tops at 0.6 w. *IEEE Journal of Solid-State Circuits*, 53(4):983–994, 2017.

[28] Shouyi Yin, Peng Ouyang, Jianxun Yang, Tianyi Lu, Xiudong Li, Leibo Liu, and Shaojun Wei. An energy-efficient reconfigurable processor for binary- and ternary-weight neural networks with flexible data bit width. *IEEE Journal of Solid-State Circuits*, 54(4):1120–1136, 2018.

[29] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.

[30] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, pages 3123–3131, 2015.

[31] Hande Alemdar, Vincent Leroy, Adrien Prost-Boucle, and Frédéric Pétrot. Ternary neural networks for resource-efficient ai applications. In *2017 in-*

ternational joint conference on neural networks (IJCNN), pages 2547–2554. IEEE, 2017.

[32] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*, 2017.

[33] Suraj Srinivas, Akshayvarun Subramanya, and R Venkatesh Babu. Training sparse neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 138–145, 2017.

[34] Arash Ardakani, Carlo Condo, and Warren J Gross. Sparsely-connected neural networks: towards efficient vlsi implementation of deep neural networks. *arXiv preprint arXiv:1611.01427*, 2016.

[35] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, 2021.

[36] Morteza Hosseini, Nitheesh Kumar Manjunath, Bharat Prakash, Arnab Mazumder, Vandana Chandrareddy, Houman Homayoun, and Tinoosh Mohsenin. Cyclic sparsely connected architectures for compact deep convolutional neural networks. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 29(10):1757–1770, 2021.

[37] Man Wu, Yirong Kan, Tati Erlina, Renyuan Zhang, and Yasuhiko Nakashima. Dianet: An elastic neural network for effectively re-configurable implementation. *Neurocomputing*, 464:242–251, 2021.

[38] Renyuan Zhang, Yan Chen, Takashi Nakada, and Yasuhiko Nakashima. Dianet: An efficient multi-grained re-configurable neural network in silicon. In *2019 32nd IEEE International System-on-Chip Conference (SOCC)*, pages 132–137. IEEE, 2019.

[39] Filip Ponulak and Andrzej Kasinski. Introduction to spiking neural networks: Information processing, learning and applications. *Acta neurobiologiae experimentalis*, 71(4):409–433, 2011.

[40] Robert Hecht-Nielsen. Theory of the backpropagation neural network. In *Neural networks for perception*, pages 65–93. Elsevier, 1992.

[41] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[42] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[43] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[44] Wolfgang Maass. Networks of spiking neurons: the third generation of neural network models. *Neural networks*, 10(9):1659–1671, 1997.

[45] Jing Pei, Lei Deng, Sen Song, Mingguo Zhao, Youhui Zhang, Shuang Wu, Guanrui Wang, Zhe Zou, Zhenzhi Wu, Wei He, et al. Towards artificial general intelligence with hybrid tianjic chip architecture. *Nature*, 572(7767):106–111, 2019.

[46] Giacomo Indiveri, Elisabetta Chicca, and Rodney Douglas. A vlsi array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity. *IEEE transactions on neural networks*, 17(1):211–221, 2006.

[47] Eustace Painkras, Luis A Plana, Jim Garside, Steve Temple, Francesco Galluppi, Cameron Patterson, David R Lester, Andrew D Brown, and Steve B Furber. Spinnaker: A 1-w 18-core system-on-chip for massively-parallel neural network simulation. *IEEE Journal of Solid-State Circuits*, 48(8):1943–1953, 2013.

[48] Paul A Merolla, John V Arthur, Rodrigo Alvarez-Icaza, Andrew S Cassidy, Jun Sawada, Filipp Akopyan, Bryan L Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, et al. A million spiking-neuron integrated circuit with a

scalable communication network and interface. *Science*, 345(6197):668–673, 2014.

[49] Evangelos Stromatias, Daniel Neil, Michael Pfeiffer, Francesco Galluppi, Steve B Furber, and Shih-Chii Liu. Robustness of spiking deep belief networks to noise and reduced bit precision of neuro-inspired hardware platforms. *Frontiers in neuroscience*, 9:222, 2015.

[50] Shibo Zhou, Xiaohua Li, Ying Chen, Sanjeev T Chandrasekaran, and Arindam Sanyal. Temporal-coded deep spiking neural network with easy training and robust performance. *arXiv preprint arXiv:1909.10837*, 2019.

[51] Sen Song, Kenneth D Miller, and Larry F Abbott. Competitive hebbian learning through spike-timing-dependent synaptic plasticity. *Nature neuroscience*, 3(9):919–926, 2000.

[52] Teresa Serrano-Gotarredona, Timothée Masquelier, Themistoklis Prodromakis, Giacomo Indiveri, and Bernabe Linares-Barranco. Stdp and stdp variations with memristors for spiking neuromorphic learning systems. *Frontiers in neuroscience*, 7:2, 2013.

[53] Peter U Diehl, Guido Zarrella, Andrew Cassidy, Bruno U Pedroni, and Emre Neftci. Conversion of artificial recurrent neural networks to spiking neural networks for low-power neuromorphic hardware. In *2016 IEEE International Conference on Rebooting Computing (ICRC)*, pages 1–8. IEEE, 2016.

[54] Jun Haeng Lee, Tobi Delbruck, and Michael Pfeiffer. Training deep spiking neural networks using backpropagation. *Frontiers in neuroscience*, 10:508, 2016.

[55] Seijoon Kim, Seongsik Park, Byunggook Na, and Sungroh Yoon. Spiking-yolo: Spiking neural network for energy-efficient object detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 11270–11277, 2020.

[56] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[57] Emre O Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, 36(6):51–63, 2019.

[58] Hanle Zheng, Yujie Wu, Lei Deng, Yifan Hu, and Guoqi Li. Going deeper with directly-trained larger spiking neural networks. *arXiv preprint arXiv:2011.05280*, 2020.

[59] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, and Luping Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in neuroscience*, 12:331, 2018.

[60] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, Yuan Xie, and Luping Shi. Direct training for spiking neural networks: Faster, larger, better. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1311–1318, 2019.

[61] Nitin Rathi, Gopalakrishnan Srinivasan, Priyadarshini Panda, and Kaushik Roy. Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation. *arXiv preprint arXiv:2005.01807*, 2020.

[62] Misha Denil, Babak Shakibi, Laurent Dinh, Marc'Aurelio Ranzato, and Nando De Freitas. Predicting parameters in deep learning. *arXiv preprint arXiv:1306.0543*, 2013.

[63] Fengfu Li, Bo Zhang, and Bin Liu. Ternary weight networks. *arXiv preprint arXiv:1605.04711*, 2016.

[64] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015.

[65] Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*, 2019.

[66] Alfred Bourely, John Patrick Boueri, and Krzysztof Choromonski. Sparse neural networks topologies. *arXiv preprint arXiv:1706.05683*, 2017.

[67] Sourya Dey, Kuan-Wen Huang, Peter A Beerel, and Keith M Chugg. Pre-defined sparse neural networks with hardware acceleration. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(2):332–345, 2019.

[68] Souvik Kundu, Mahdi Nazemi, Massoud Pedram, Keith M Chugg, and Peter A Beerel. Pre-defined sparsity for low-complexity convolutional neural networks. *IEEE Transactions on Computers*, 69(7):1045–1058, 2020.

[69] Shijin Zhang, Zidong Du, Lei Zhang, Huiying Lan, Shaoli Liu, Ling Li, Qi Guo, Tianshi Chen, and Yunji Chen. Cambricon-x: An accelerator for sparse neural networks. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1–12. IEEE, 2016.

[70] Jeremy Kepner and Ryan Robinett. Radix-net: Structured sparse matrices for deep neural networks. In *2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 268–274, 2019.

[71] Guotian Xie, Jingdong Wang, Ting Zhang, Jianhuang Lai, Richang Hong, and Guo-Jun Qi. Interleaved structured sparse convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8847–8856, 2018.

[72] Steven K Esser, Paul A Merolla, John V Arthur, Andrew S Cassidy, Rathinakumar Appuswamy, Alexander Andreopoulos, David J Berg, Jeffrey L McKinstry, Timothy Melano, Davis R Barch, et al. Convolutional networks for fast, energy-efficient neuromorphic computing. *Proceedings of the national academy of sciences*, 113(41):11441–11446, 2016.

[73] Nitin Rathi, Priyadarshini Panda, and Kaushik Roy. Stdp-based pruning of connections and weight quantization in spiking neural networks for energy-efficient recognition. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 38(4):668–677, 2018.

[74] Yuhan Shi, Leon Nguyen, Sangheon Oh, Xin Liu, and Duygu Kuzum. A soft-pruning method applied during training of spiking neural networks for in-memory computing applications. *Frontiers in neuroscience*, 13:405, 2019.

[75] Ruizhi Chen, Hong Ma, Shaolin Xie, Peng Guo, Pin Li, and Donglin Wang. Fast and efficient deep sparse multi-strength spiking neural networks with dynamic pruning. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2018.

[76] Lei Deng, Yujie Wu, Yifan Hu, Ling Liang, Guoqi Li, Xing Hu, Yufei Ding, Peng Li, and Yuan Xie. Comprehensive snn compression using admm optimization and activity regularization. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[77] Sen Lu and Abhronil Sengupta. Exploring the connection between binary and spiking neural networks. *Frontiers in Neuroscience*, 14:535, 2020.

[78] Hong-Han Lien, Chung-Wei Hsu, and Tian-Sheuan Chang. Vsa: Reconfigurable vectorwise spiking neural network accelerator. In *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5. IEEE, 2021.

[79] Yixuan Wang, Yang Xu, Rui Yan, and Huajin Tang. Deep spiking neural networks with binary weights for object recognition. *IEEE Transactions on Cognitive and Developmental Systems*, 2020.

[80] Saeed Reza Kheradpisheh, Maryam Mirsadeghi, and Timothée Masquelier. Bs4nn: Binarized spiking neural networks with temporal coding and learning. *Neural Processing Letters*, pages 1–19, 2021.

[81] Manoj Alwani, Han Chen, Michael Ferdman, and Peter Milder. Fused-layer cnn accelerators. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1–12. IEEE, 2016.

[82] Wei Fang, Zhaofei Yu, Yanqi Chen, Tiejun Huang, Timothée Masquelier, and Yonghong Tian. Deep residual learning in spiking neural networks. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

[83] Garrick Orchard, Ajinkya Jayawant, Gregory K Cohen, and Nitish Thakor. Converting static image datasets to spiking neuromorphic datasets using saccades. *Frontiers in neuroscience*, 9:437, 2015.

[84] Nitin Rathi and Kaushik Roy. Diet-snn: A low-latency spiking neural network with direct input encoding and leakage and threshold optimization. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[85] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037, 2019.

[86] Ameya Prabhu, Girish Varma, and Anoop Namboodiri. Deep expander networks: Efficient deep networks from graph theory. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 20–35, 2018.

[87] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[88] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[89] Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

[90] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[91] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[92] A Emin Orhan and Xaq Pitkow. Skip connections eliminate singularities. *arXiv preprint arXiv:1701.09175*, 2017.

[93] Andreas Veit, Michael J Wilber, and Serge Belongie. Residual networks behave like ensembles of relatively shallow networks. *Advances in neural information processing systems*, 29:550–558, 2016.

[94] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[95] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[96] Man Wu, Yan Chen, Yirong Kan, Takeshi Nomura, Renyuan Zhang, and Yasuhiko Nakashima. An elastic neural network toward multi-grained reconfigurable accelerator. In *2020 18th IEEE International New Circuits and Systems Conference (NEWCAS)*, pages 218–221. IEEE, 2020.

[97] Michael A Nielsen. *Neural networks and deep learning*, volume 25. Determination press San Francisco, CA, USA:, 2015.

[98] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[99] Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013.

[100] Suraj Srinivas and R Venkatesh Babu. Learning neural network architectures using backpropagation. *arXiv preprint arXiv:1511.05497*, 2015.

[101] M Gethsiyal Augasta and T Kathirvalavakumar. A novel pruning algorithm for optimizing feedforward neural network of classification problems. *Neural processing letters*, 34(3):241, 2011.

[102] Olivier Temam. A defect-tolerant accelerator for emerging high-performance applications. In *2012 39th Annual International Symposium on Computer Architecture (ISCA)*, pages 356–367. IEEE, 2012.

[103] Yu Ji, YouHui Zhang, ShuangChen Li, Ping Chi, CiHang Jiang, Peng Qu, Yuan Xie, and WenGuang Chen. Neutrams: Neural network transformation and co-design under neuromorphic hardware constraints. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1–13, 2016.

[104] Jaeyong Chung and Taehwan Shin. Simplifying deep neural networks for neuromorphic architectures. In *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, 2016.

[105] Jason Cong, Hui Huang, Chiyuan Ma, Bingjun Xiao, and Peipei Zhou. A fully pipelined and dynamically composable architecture of cgra. In *2014 IEEE 22nd Annual International Symposium on Field-Programmable Custom Computing Machines*, pages 9–16. IEEE, 2014.

[106] Kan Yirong, Man Wu, Renyuan Zhang, and Yasuhiko Nakashima. A multi-grained reconfigurable accelerator for approximate computing. In *2020 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 90–95. IEEE, 2020.

[107] Hongfei Wang, Jianwen Li, and Kun He. Hierarchical ensemble reduction and learning for resource-constrained computing. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 25(1):1–21, 2019.

[108] Yirong Kan, Man Wu, Renyuan Zhang, and Yasuhiko Nakashima. Mugra: A scalable multi-grained reconfigurable accelerator powered by elastic neural network. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2021.

[109] David Heeger et al. Poisson model of spike generation. *Handout, University of Standford*, 5(1-13):76, 2000.

List of publications

Peer review journal papers

1. Man Wu, Yirong Kan, Tati Erlina, Renyuan Zhang, and Yasuhiko Nakashima. ”DiaNet: An elastic neural network for effectively re-configurable implementation.” Neurocomputing 464 (2021): 242-251. (Section4)

2. Kan Yirong, Man Wu, Renyuan Zhang, and Yasuhiko Nakashima. ”MuGRA: A Scalable Multi-Grained Reconfigurable Accelerator Powered by Elastic Neural Network.” IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 69, no. 1, pp. 258-271, Jan. (2022)

Peer review conference papers

1. Man Wu, Yan Chen, Yirong Kan, Takeshi Nomura, Renyuan Zhang, and Yasuhiko Nakashima. ”An Elastic Neural Network Toward Multi-Grained Re-configurable Accelerator.” In 2020 18th IEEE International New Circuits and Systems Conference (NEWCAS), pp. 218-221. IEEE, 2020. (Oral) (Section4)

2. Kan Yirong, Man Wu, Renyuan Zhang, and Yasuhiko Nakashima. ”A Multi-grained Reconfigurable Accelerator for Approximate Computing.” In 2020 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), pp. 90-95. IEEE, 2020. (Oral)

Workshop

1. Man Wu, Yirong Kan, Vantinh Nguyen, Renyuan Zhang, Yasuhiko Nakashima. ”Ternarizing Deep Spiking Neural Network”,In CPSY, 2022 January. (Section3)