frontiers | Frontiers in Neuroscience

TYPE Original Research
PUBLISHED 23 July 2024
DOI 10.3389/fnins.2024.1372257

OPEN ACCESS

EDITED BY
Mohammed Fouda,
University of California, Irvine, United States

REVIEWED BY
Pengju Ren,
Xi'an Jiaotong University, China
Rachmad Vidya Wicaksana Putra,
New York University Abu Dhabi,
United Arab Emirates
Jundong Liu,
Ohio University, United States

*CORRESPONDENCE
Li Yuan
✉ yuanli-ece@pku.edu.cn
Yonghong Tian
✉ yhtian@pku.edu.cn

RECEIVED 17 January 2024
ACCEPTED 05 July 2024
PUBLISHED 23 July 2024

CITATION
Che K, Zhou Z, Niu J, Ma Z, Fang W, Chen Y, Shen S, Yuan L and Tian Y (2024) Auto-Spikformer: Spikformer architecture search. *Front. Neurosci.* 18:1372257. doi: 10.3389/fnins.2024.1372257

Auto-Spikformer: Spikformer architecture search

Kaiwei Che[1,2], Zhaokun Zhou[1,2], Jun Niu[1], Zhengyu Ma[2], Wei Fang[1,2], Yanqi Chen[1,2], Shuaijie Shen[3], Li Yuan[1,2]* and Yonghong Tian[1,2]*

[1]School of Electronic and Computer Engineering, Shenzhen Graduate School, Peking University, Shenzhen, Guangdong, China, [2]Peng Cheng Laboratory, Shenzhen, Guangdong, China, [3]Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, Guangdong, China

Introduction: The integration of self-attention mechanisms into Spiking Neural Networks (SNNs) has garnered considerable interest in the realm of advanced deep learning, primarily due to their biological properties. Recent advancements in SNN architecture, such as Spikformer, have demonstrated promising outcomes. However, we observe that Spikformer may exhibit excessive energy consumption, potentially attributable to redundant channels and blocks.

Methods: To mitigate this issue, we propose a one-shot Spiking Transformer Architecture Search method, namely Auto-Spikformer. Auto-Spikformer extends the search space to include both transformer architecture and SNN inner parameters. We train and search the supernet based on weight entanglement, evolutionary search, and the proposed Discrete Spiking Parameters Search (DSPS) methods. Benefiting from these methods, the performance of subnets with weights inherited from the supernet without even retraining is comparable to the original Spikformer. Moreover, we propose a new fitness function aiming to find a Pareto optimal combination balancing energy consumption and accuracy.

Results and discussion: Our experimental results demonstrate the effectiveness of Auto-Spikformer, which outperforms the original Spikformer and most CNN or ViT models with even fewer parameters and lower energy consumption.

KEYWORDS

spiking neural network (SNN), transformer, transformer architecture search, network architecture search (NAS), evolutionary algorithm (EA)

1 Introduction

Spiking neural networks (SNNs) show promise for the next generation of artificial intelligence, owing to their biological inspiration and appealing features such as sparse activation and temporal dynamics. The performance of SNNs has improved by employing advanced architectures from ANNs, such as ResNet-like SNNs (Fang et al., 2021a; Hu et al., 2021a,b; Zheng et al., 2021), or Spiking Recurrent Neural Networks (Lotfi Rezaabad and Vishwanath, 2020). Transformer, originally developed for natural language processing (Vaswani et al., 2017), has proven successful in various computer vision applications, including image classification (Dosovitskiy et al., 2020; Yuan et al., 2021a), object detection (Carion et al., 2020; Zhu et al., 2020; Liu et al., 2021), and semantic segmentation (Wang et al., 2021; Yuan et al., 2021b). The self-attention mechanism, a crucial component of the Transformer model, selectively attends to relevant information and is analogous to an important feature of the human biological system (Caucheteux and King, 2022; Whittington et al., 2022). The integration of self-attention into SNNs for advanced deep learning has gained attention due to the biological properties of both mechanisms. Spikformer (Zhou et al., 2022), a recent SNN architecture, has demonstrated promising

results on both static and neuromorphic datasets using its Spiking Self-Attention (SSA) and Spiking Patch Splitting (SPS) modules.

While SNNs are known for their low energy consumption compared to ANNs, our observations revealed that the energy consumption of Spikformer can be significantly reduced as it contains potentially redundant channels and blocks. In Figure 1, we observed suboptimal architecture parameters in the original Spikformer, with redundancy channels, particularly in higher-order channels (See Section 3 for more details). These phenomenons motivates us to search to design a better Spikformer architectures. Nevertheless, designing and training such hybrid models remains a challenging task (Dosovitskiy et al., 2020; Touvron et al., 2021).

We address the Spikformer search problem by dividing it into two main parts: the Transformer part and the SNN neuron part. Transformer Architecture Search (TAS) (Chen B. et al., 2021; Chen M. et al., 2021; Su et al., 2022) has gained attention as an automated way to search for multiple configurations of Vision Transformer (ViT) architectures. The one-shot NAS scheme (Dong and Yang, 2019; Chen M. et al., 2021), leveraged in TAS, obtains reliable performance estimations on various ViT architectures. We choice weight entanglement supernet training strategy (Chen M. et al., 2021) as base search method to optimize the Transformer architecture. However, directly applying TAS may not be the most optimal solution for Spiking Transformers. The original TAS method does not consider the SNN search space and the energy consumption, which is vital in the field of SNN.

To optimize the internal parameters of SNN neurons, we propose a method that leverages the concept of natural selection and evolutionary algorithms. While previous studies have focused on improving SNN performance through network structure exploration, the significance of individual neuron parameters has also been identified (Che et al., 2022; Kim et al., 2022; Na et al., 2022). We draw inspiration from Darwin's theory of evolution, which suggests that organisms adapt to their environment through natural selection over time (Slowik and Kwasnicka, 2020; Jordan et al., 2021). Similarly, SNN neurons can evolve and optimize their internal parameters to enhance network performance. By treating traits such as the threshold, decay, and time-step parameters of a neuron as candidate solutions and the input stimuli as the environment, we can apply simulated evolution to find optimal parameter sets that improve accuracy and efficiency. This novel approach, referred to as Discrete Spiking Parameters Search (DSPS), utilizes an evolutionary algorithm to search for the internal parameters of SNN neurons. Our study is the first to apply the evolutionary algorithm to search for the internal parameters of SNN neurons.

Our method for optimizing Spikformer explores the optimal combination of key factors but doesn't ensure lower energy consumption. To address this, we introduce a joint fitness function, $\mathcal{F}_{AEB}$, balancing energy consumption and accuracy. This allows us to achieve a Pareto optimal combination, striking a balance between these two objectives.

We summarize our contributions are as follows:

- We provide the first systematic and in-depth analysis of the channel redundancy in SNNs by analysing the performance curve and Structural Similarity (SSIM), which are crucial to the high energy efficiency.
- To the best of our knowledge, this study is the first to use NAS for spiking-based ViT, namely Auto-Spikformer. By employing Discrete Spiking Parameters Search (DSPS) and the weight entanglement supernet training method, Auto-Spikformer enhances the efficiency and accuracy of spiking-based ViT architectures.
- Auto-Spikformer integrates an accuracy and energy balanced fitness function $\mathcal{F}_{AEB}$ to optimize the Spikformer search space by considering both energy consumption and accuracy simultaneously.

2 Related work

2.1 Spiking neural networks

Unlike traditional deep learning models that perform computations using floating-point values, SNNs leverage discrete spike sequences for information processing and transmission. Spiking neurons endow SNNs with temporal dynamics and biological properties. Common types include the leaky integrate-and-fire (LIF) neuron (Wu et al., 2018), PLIF (Fang et al., 2021b), etc. Two main approaches for obtaining deep SNNs are ANN-to-SNN conversion and direct training. In ANN-to-SNN conversion, a pre-trained ANN with high performance is transformed into an SNN by substituting the ReLU activation layers with spiking neurons (Cao et al., 2015; Hunsberger and Eliasmith, 2015; Rueckauer et al., 2017; Bu et al., 2021; Meng et al., 2022; Wang et al., 2022). However, this method requires large time-steps to approximate ReLU activation accurately, leading to high latency (Han et al., 2020). In direct training, SNNs are trained by backpropagation through time (BPTT) (Werbos, 1990). A challenge for direct training is the non-differentiability of the event-triggered mechanism in spiking neurons. To address this challenge, surrogate gradients are employed for backpropagation (Neftci et al., 2019; Lee et al., 2020; Xiao M. et al., 2021) adopts implicit differentiation on the equilibrium state to train SNNs.

2.2 Vision transformer

The Vision Transformer (ViT) facilitates the transformation from NLP to CV by partitioning visual information into patches and processing it accordingly. For image classification, a Transformer encoder comprises a patch splitting module, multiple Transformer encoder blocks, and a linear prediction head. Each Transformer encoder block includes a self-attention layer and a multi-perception layer. Self-attention is a fundamental component contributing to ViT's success. It captures global dependence and interest representation by weighing feature values of image patches via the dot product of the query and key, followed by the application of the softmax function (Katharopoulos et al., 2020; Qin et al., 2022). Researchers have made improvements to the visual transformer, including the Transformer architecture (Hassani et al., 2021; Xiao T. et al., 2021), more advanced self-attention mechanisms (Choromanski et al., 2020; Rao et al., 2021; Song, 2021; Yang et al., 2021), and pre-training techniques (He et al.,

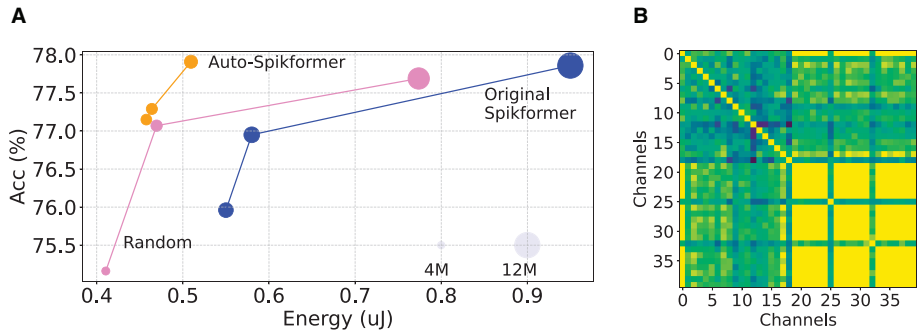

FIGURE 1
Analysis of redundancy of Spikformer **(A)** Relationship among energy consumption, number of parameters, and accuracy for various Spikformer candidates. Original Spikformer candidates are obtained from Zhou et al. (2022). We select 100 candidates from the Spikformer large search space $\mathcal{S}_{T_S}$ using our proposed Auto-Spikformer method and random selection method, then plot their Pareto frontier onto the figure. Note that larger circles represent a higher number of parameters. Detailed results can be found in Section 5.3. **(B)** The Structural Similarity (SSIM) matrix between channels after embedding (also called SPS) in Spikformer. Both the X and Y axes represent channels. The color indicates the SSIM value: yellow denotes higher similarity, while green denotes lower similarity. The matrix reveals significant redundancy in channels, particularly in higher-order channels, after embedding.

2022), among others. Spikformer (Zhou et al., 2022), a recent SNN-based Transformer, has demonstrated promising results on both static and neuromorphic datasets. Observations reveal potential redundancy in channels and blocks, motivating us to explore a more efficient SNN-based Transformer automatically.

2.3 One-shot NAS

Designing high-performance network architectures for specific tasks often requires expert experience and trial-and-error experiments. Neural architecture search (NAS) (Elsken et al., 2019) aims to automate this manual process and has recently achieved highly competitive performance in tasks such as image classification (Zoph and Le, 2016; Liu C. et al., 2018; Pham et al., 2018; Zoph et al., 2018; Real et al., 2019), object detection (Zoph et al., 2018; Chen Y. et al., 2019; Guo et al., 2020; Wang et al., 2020), and semantic segmentation (Liu et al., 2019; Nekrasov et al., 2019; Zhang et al., 2019; Lin et al., 2020), etc. However, searching over a discrete set of candidate architectures often results in a massive number of potential combinations, leading to explosive computational cost. The recently proposed differentiable architecture search (DARTS) method (Liu H. et al., 2018) and its variations (Chen X. et al., 2019; Xu et al., 2019; Chu et al., 2020) address this problem using a continuous relaxation of the search space, enabling learning a set of architecture coefficients by gradient descent. They have achieved competitive performances with the state-of-the-art using orders of magnitude fewer computation resources (Liu H. et al., 2018; Liu et al., 2019; Cheng et al., 2020). Recently, Na et al. (2022) studied pooling operations for downsampling in SNNs and applied NAS to reduce the overall number of spikes. Kim et al. (2022) applied NAS to improve SNN initialization and explore backward connections. However, both works only searched for different SNN cells or combinations of them within traditional CNNs. There is a lack of work on searching for SNN internal parameters and SNN-based transformer architectures.

3 Problem analysis

We conducted several experiments and metrics to analyze the redundancy in Spikformer. Our observations revealed three key phenomena: (1) the original Spikformer architecture parameters are not optimal; (2) there is redundancy in the channels after embedding; and (3) most of the redundant channels are found at higher-order channels.

Exploration of the optimal combination of key factors in Spikformer. As depicted in Figure 1A, we trained a supernet and randomly selected candidates to evaluate their performance, plotting their Pareto frontier. Surprisingly, some randomly selected candidates performed optimally in both energy and accuracy. Upon analyzing these high-performing candidates, we discovered that their blocks and channels were both fewer than those in the original Spikformer, indicating that the original architecture parameters are suboptimal.

Analysis of redundancy in Spikformer. Previous work has identified high sparsity and redundancy in spike features in spiking convolutional neural networks (Yao et al., 2023). Spiking Transformer also exhibits redundancy in both channels and the number of blocks. We further analyzed the Structural Similarity (SSIM) to measure the similarity between features at different scales. As shown in Figure 1B, we inferred the trained 8–384 Spikformer model and calculated SSIM between each channel's feature map after embedding. The feature map consists of 384 channels. We selected the top 20 channels with the highest and lowest SSIM scores, preserving their original order to construct a matrix. Our analysis revealed redundancy in the channels after embedding, with most of the redundant channels found at higher-order channels.

To address these issues, we propose a hybrid architecture search, using the Transformer Architecture Search (TAS)

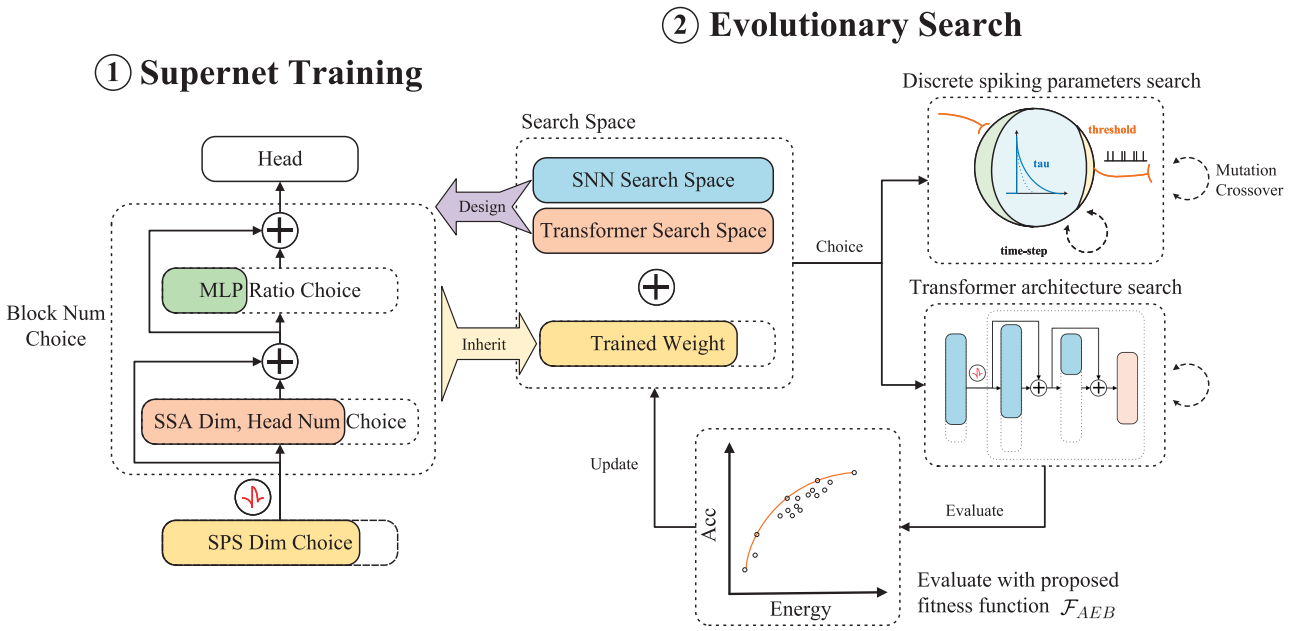

FIGURE 2
Auto-Spikformer including two stages: supernet training stage and evolutionary search stage. We first design a search space that includes both SNN inner parameters and transformer architecture. Within this search space, we train the supernet based on weight entanglement for the transformer space and an alternate choice method for the SNN space. After supernet training, we use evolutionary search to choose the best transformer architecture and optimal SNN inner parameters with weights inherited from the supernet. Note that we denote the process of designing the SNN search space, using the alternate choice method to train the SNN space supernet, and searching the SNN space with evolutionary search as Discrete Spiking Parameters Search (DSPS). We then evaluate the performance of the searched subnet with our proposed energy-accuracy balanced fitness function $\mathcal{F}_{AEB}$. Spiking Self-Attention (SSA) is the attention block in Spikformer. And Spiking Patch Splitting (SPS) is the embedding module in Spikformer.

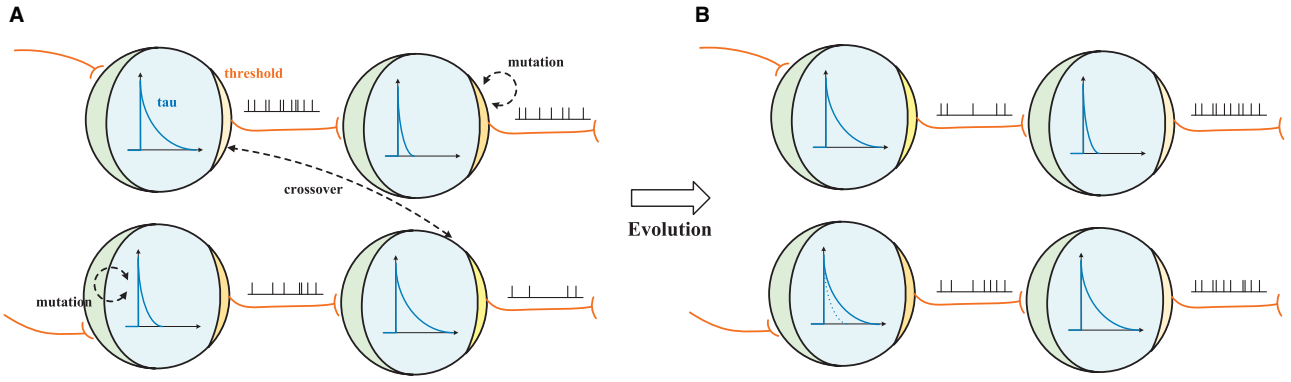

FIGURE 3
Discrete Spiking Parameters Search (DSPS) process. **(A, B)** show two candidate parameter sets before and after applying the mutation and crossover operators. The spike from the previous neuron is transmitted to the current neuron during the charging process. If the membrane potential is above the threshold (yellow area, where the darker color indicates a higher value), a spike is delivered; if the membrane potential is below the threshold, it decays exponentially with a time constant τ (blue curve). During the supernet training stage, these parameters are searched using the alternate choice method.

method to explore optimal combinations in Spikformer. In the TAS field, the weight entanglement method is used to train and select channels in order, as shown in Figure 4. Therefore, in this work, we use the weight entanglement method to optimize the transformer factors. However, this method cannot optimize discrete parameters like tau, threshold, and time-step, which are important in SNNs. Therefore, we propose a discrete method to optimize these spiking parameters, addressing the limitations in current optimization approaches.

4 Auto-Spikformer

We propose Auto-Spikformer, a one-shot Spiking Transformer Architecture Search method combining the search of Transformer

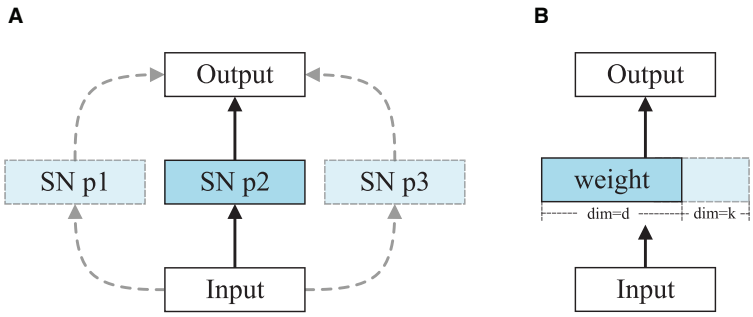

FIGURE 4
(A) Alternate choice method for training SNN search space. SN p1 denotes spiking neuron parameter setting 1, like $[u_{th} = 1.2, \tau = 1.25, t = 4]$. Note that they are all discrete values and without trainable parameters. **(B)** Weight entanglement method for training transformer search space (Chen M. et al., 2021).

and SNN neurons simultaneously. Auto-Spikformer consists of two stages: the supernet training stage and the evolutionary search stage. We first briefly introduce the spiking neuron, followed by an overview of Auto-Spikformer, the DSPS method, and the fitness function.

As shown in Figure 2, during the supernet training stage, we use Spikformer (Zhou et al., 2022) as our base model to construct the supernet. We then train the supernet using the weight entanglement method for the Transformer space and the alternate choice method for the SNN space. After supernet training, we employ evolutionary search to select the optimal transformer architecture and SNN inner parameters with weights inherited from the supernet (as discussed in Section 4.2). Note that the fitness function aims to find a Pareto optimal combination balancing energy consumption and accuracy, as shown in Section 4.4.

TABLE 1 Auto-Spikformer search space.

	$\mathcal{S}_{T_s}$	$\mathcal{S}_{T_l}$
(A) Transformer search space		
Embed dim	(336,384,12)	(336,480,48)
MLP ratio	(3,4,0.2)	(3,5,0.2)
Head num	(6,12,6)	(6,12,6)
Depth	(2,4,1)	(2,6,1)
(B) SNN search space		
	$\mathcal{S}_S$	
Threshold u_{th}	(0.6,2,0.2)	
Decay τ	(1.25,10,0.25)	
Time-step t	(2,4,1)	

$\mathcal{S}_{T_s}$ denotes the transformer smaller search space, $\mathcal{S}_{T_l}$ denotes the transformer larger search space, and $\mathcal{S}_S$ denotes the SNN search space. Each element in the table represents the lower limit, upper limit, and step size, such as depth (2,4,1) represents the depth range from 2 to 4, step size is 1.

4.1 LIF

We adopt the iterative LIF neuron model (Wu et al., 2019) described by (Equation 1)

$$u^{t,n} = (1 - \frac{1}{\tau})u^{t-1,n}(1 - y^{t-1,n}) + I^{t,n} \quad (1)$$

where superscripts n and t denote the layer index and time-step, respectively. The decay τ is the membrane time constant, u is the membrane potential, y denotes the spike output, and I denotes the synaptic input with $I^{t,n} = \sum_j w_j y_j^{t,n-1}$, where w is the weight. The neuron fires a spike $y^{t,n} = 1$ when $u^{t,n}$ exceeds a threshold V_{th}; otherwise, $y^{t,n} = 0$. In this work, we set $\tau = 2$ and $u_{th} = 0.5$.

4.2 Discrete spiking parameters search

Motivation. The performance of SNN neurons is influenced by both their interconnections and internal parameters. While previous research has primarily focused on enhancing SNN performance through modifications to the network's structure, the importance of optimizing the internal parameters within individual neurons cannot be overlooked. Darwin's theory of evolution posits that organisms adapt to their surroundings through natural selection, favoring traits that enhance survival and reproduction. This concept can be applied to the context of SNN, where individual neurons can undergo an evolutionary process. In this context, the internal parameters of a neuron, such as the threshold (u_{th}), decay (τ), and time-step (t), can be seen as analogous to traits, while the input stimuli received by the neuron can be likened to the environment in which it operates. Previous work (Fontaine et al., 2014) suggests that the threshold can be viewed as an adaptation to membrane potentials at short timescales, influencing how signals received by a neuron are encoded into a spike. Decay τ has a similar effect to the threshold, but it only affects the decay of unfired neurons, influencing the firing of the next timestep. In contrast, the threshold affects the firing of all neurons at the current moment.

Discrete spiking parameters search process. As shown in Figure 3, the spike from the previous neuron is transmitted to the current neuron during the charging process. If the membrane potential is above the threshold, a spike is delivered; if the membrane potential is below the threshold, it decays at the rate of τ. The DSPS begins with a population of randomly generated parameter sets (candidates) like $[u_{th} = 1.2, \tau = 1.25, t = 4]$. In each generation, the algorithm evaluates the fitness of

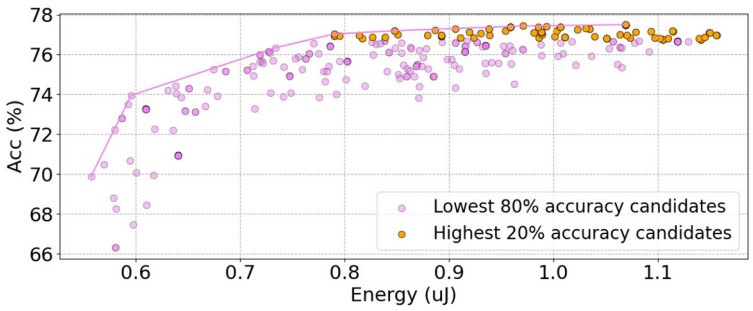

FIGURE 5
The energy and accuracy of all candidates in $\mathcal{S}_S$ in CIFAR100. We use $\mathcal{F}_{AEB}$ as the fitness function to select the top 300 candidates. The purple points represent the candidates with the lowest 80% accuracy, while the orange points represent the candidates with the highest 20% accuracy. The purple line represents the Pareto frontier, indicating the optimal trade-off between accuracy and energy consumption.

TABLE 2 Subsets of the candidates in $\mathcal{S}_S$.

Candidates (threshold ×4, tau ×4, time-step)	Fr	Energy (μJ)	Acc (%)
(A) Candidates on the Pareto frontier			
(1.6, 0.6, 0.8, 2.0, 10, 10, 10, 2, 2)	0.20	0.52	72.12
(1.8, 0.8, 1.4, 1.2, 10, 10, 10, 2, 2)	0.20	0.52	72.18
(1.6, 0.6, 2.0, 1.8, 5, 10, 10, 1.5, 4)	0.24	0.63	75.58
(1.0, 1.0, 1.4, 1.0, 10, 10, 2, 3, 4)	0.25	0.66	76.44
(1.8, 1.6, 0.6, 0.8, 5, 10, 2, 3, 4)	0.26	0.68	76.77
(0.8, 1.2, 1.6, 2.0, 5, 10, 2, 1.5, 4)	0.27	0.72	77.20
(1.0, 2.0, 1.6, 2.0, 5, 2, 2, 3, 4)	0.30	**0.79**	**77.87**
(1.0, 2.0, 1.4, 1.6, 2, 2, 1.25, 5, 4)	0.37	0.99	77.95
(1.0, 1.0, 1.0, 1.0, 2, 2, 2, 2, 4)*	0.35	0.95	77.86
(B) Candidates with the top 20% accuracy			
(1.0, 2.0, 1.6, 2.0, 5, 2, 2, 3, 4)	0.30	**0.79**	**77.87**
(1.2, 1.8, 1.8, 1.6, 2, 10, 1.5, 5, 4)	0.33	**0.87**	**77.90**
(1.6, 1.2, 1.8, 2.0, 1.5, 5, 10, 2, 4)	0.33	**0.88**	**77.86**
(1.6, 0.8, 1.2, 1.8, 2, 10, 1.25, 3, 4)	0.34	0.91	77.74
(0.6, 1.4, 1.2, 0.6, 2, 3, 1.25, 5, 4)	0.35	0.94	77.78
(0.8, 0.8, 1.8, 1.8, 2, 2, 1.5, 2, 3)	0.36	**0.95**	**77.92**
(1.0, 1.4, 1.8, 0.6, 2, 2, 1.5, 3, 4)	0.36	0.96	77.77
(1.0, 2.0, 1.4, 1.6, 2, 2, 1.25, 5, 4)	0.37	0.99	77.95
(1.0, 1.0, 1.0, 1.0, 2, 2, 2, 2, 4)*	0.35	0.95	77.86

Each selected candidate includes 9 features: (threshold ×4, tau ×4, time-step). Using the first candidate as an example, (1.6, 0.6, 0.8, 2.0) means the thresholds of SNN for four blocks are (1.6, 0.6, 0.8, 2.0) separately. Fr denotes the firing rate. The last row corresponds to Spikformer 4-384, denoted as $*$, obtained by running the open-source code of Spikformer. The bold font represents that the energy consumption and accuracy are superior to Spikformer 4-384.

candidates and selects the best ones as the parents for the next generation. The parents produce offspring by applying mutation and crossover operators with some probabilities. The mutation operator randomly modifies one parameter of a parameter set, while the crossover operator combines two parameters from different parents. As illustrated in Figure 3, for example, the decay τ of a candidate changes from 1.25 to 2.5 after mutation. The thresholds u_{th} of two candidates are swapped after crossover, affecting the firing rate of each candidate. The algorithm repeats this process for a fixed number of generations and returns the best architecture found. Through a process of simulated evolution, the threshold, decay, and time-step parameters of individual neurons can be adjusted to improve the performance of the network as a whole. According to our experiments, this approach can lead to the network becoming better adapted to the input stimuli it receives, resulting in increased accuracy and efficiency.

Alternate choice and weight entanglement. Figure 4 illustrates the differences between the weight entanglement method and the alternate choice method. The weight entanglement method allows different transformer blocks within a supernet to share weights for common parts in each layer. This strategy leads to faster convergence, lower memory cost, and better performance of subnets compared to classical weight-sharing methods. We apply this strategy to train the transformer block in Spikformer. However, the SNN search space is discrete and lacks trainable parameters, limiting the application of the weight entanglement method. It requires the search space to undergo continuous changes, or in other words, the search space should share some common parts; it cannot be entirely discrete. When we use weight entanglement, channels change from 380 to 384, with the previous 380 channels sharing the same weights, leaving only the last 4 channels different. However, for SNN inner parameters such as threshold, changing from 0.3 to 0.5, there are no common parts to share. Therefore, we employ the alternate choice method for training instead of weight entanglement.

4.3 Search space

We design a search space including both SNN inner parameters and transformer architecture as shown in Table 1.

SNN search space, denoted $\mathcal{S}_S$, includes three variable factors: the threshold u_{th}, decay τ, and time-step t. The structured

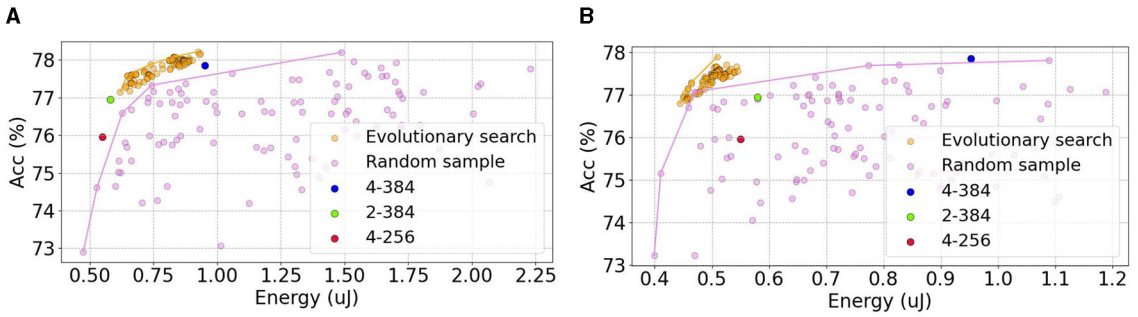

FIGURE 6
The energy and accuracy of all searched candidates in $\mathcal{S}_{T_s}$ **(A)** and $\mathcal{S}_{T_l}$ **(B)** in CIFAR100. We use $\mathcal{F}_{AEB}$ as the fitness function (purple points) and randomly (orange points) select the 100 candidates, respectively. The other color points represent the different architectures of Spikformer, derived from the original paper (Zhou et al., 2022).

TABLE 3 Subsets of the candidates in $\mathcal{S}_T$.

	Candidates (depth (d), MLP ratio × d, head num × d, threshold × d, tau × d, time-step, embed dim)	Energy (μJ)	Acc (%)
(A) Candidates in $\mathcal{S}_{T_s}$. We denote ^{1,2} as Auto-Spikformer $\mathcal{S}_{T_s}{}^{1,2}$.			
Pareto frontier	(2, 3.2, 3.0, 12, 6, 1.2, 1.0, 5, 5, 4, 348)	0.448	76.89
	(2, 3.4, 3.2, 12, 6, 1.0, 1.0, 5, 5, 4, 348)	0.453	77.04
	(2, 3.8, 3.2, 12, 6, 0.6, 1.8, 5, 5, 4, 348)	0.458	77.15
	(2, 3.8, 3.8, 12, 6, 1.0, 1.8, 5, 5, 4, 348)	0.464	77.29
	(2, 3.6, 3.6, 6, 12, 1.8, 2.0, 5, 2, 4, 348)	**0.509**	**77.91**
Top accuracy	(2, 3.8, 3.6, 12, 12, 1.4, 2.0, 5, 2, 4, 348)[1]	0.505	77.71
	(2, 3.6, 3.6, 6, 12, 1.8, 2.0, 5, 2, 4, 348)[2]	**0.509**	**77.91**
	(2, 3.6, 3.6, 6, 12,0.6, 2.0, 5, 2, 4, 348)	**0.510**	**77.91**
	(2, 3.4, 3.8, 6, 6, 1.0, 1.8, 5, 2, 4, 360)	**0.535**	**77.89**
	(2, 3.6, 3.6, 6, 6, 0.6, 2.0, 5, 2, 4, 360)	**0.536**	**77.90**
Spikformer 4-384	(4, 4, 4, 4, 4, 12, 12, 12, 12, 1.0, 1.0, 1.0, 1.0, 2, 2, 2, 2, 4, 384)	0.95	77.86
(B) Candidates in $\mathcal{S}_{T_l}$. We denote ^{1,2,3} as Auto-Spikformer $\mathcal{S}_{T_l}{}^{1,2,3}$.			
(2, 4.8, 3.2, 6, 6, 1.4, 1.6, 5, 5, 4, 384)		0.619	77.15
(2, 3.8, 3.2, 6, 12, 0.8, 1.2, 5, 5, 4, 432)		0.647	77.59
(2, 3.8, 4.2, 12, 6, 1.4, 1.4, 3, 5, 4, 432)		**0.737**	**77.88**
(2, 3.8, 4.2, 12, 12, 0.8, 1.2, 3, 1.5, 4, 432)		**0.829**	**78.08**
(3, 3.2, 3.6, 3.2, 6, 12, 6, 1.4, 1.8, 0.8, 3, 5, 5, 4, 480)		**0.925**	**78.22**
(2, 3.8, 3.2, 6, 12, 0.8, 1.2, 3, 1.5, 4, 432)[1]		**0.826**	**78.01**
(2, 3.8, 4.2, 12, 12, 0.6, 0.8, 3, 1.5, 4, 432)		**0.829**	**78.08**
(2, 4.2, 4.2, 12, 12, 0.8, 1.2, 3, 1.5, 4, 480)[2]		**0.889**	**78.05**
(3, 3.2, 3.6, 3.2, 6, 12, 6, 1.4, 1.8, 0.8, 3, 5, 5, 4, 480)[3]		**0.925**	**78.22**
(3, 3.2, 3.6, 3.0, 6, 12, 12, 1.4, 2.0, 2.0, 3, 5, 3, 4, 480)		**0.934**	**78.17**
(4, 4, 4, 4, 4, 12, 12, 12, 12, 1.0, 1.0, 1.0, 1.0, 2, 2, 2, 2, 4, 384)		0.95	77.86

Each selected candidate includes $d \times 4 + 3$ features: (depth (d), MLP ratio × d, head num × d, threshold × d, tau × d, time-step, embed dim). Using the first candidate as an example, (2) denotes there are two only blocks. (3.2, 3.0) means MLP ratio for two blocks is (1.6, 0.6, 0.8, 2.0) separately. Blue color represents the SNN search space while orange color represents the transformer search space. Bold value means the result is more optimal.

TABLE 4 Performance comparison of Auto-Spikformer with existing methods on CIFAR10/100.

Methods	Architecture	Param (M) / Energy (μJ)	Time step	CIFAR10 Acc	CIFAR100 Acc	Model type	Design type
Hybrid training (Rathi et al., 2020)	VGG-11	9.27 / -	125	92.22	67.87	CNN	Manual
Diet-SNN (Rathi and Roy, 2020)	ResNet-20	0.27/ -	10/5	92.54	64.07	CNN	Manual
STBP (Wu et al., 2018)	CIFARNet	17.54 / -	12	89.83	-	CNN	Manual
STBP NeuNorm (Wu et al., 2019)	CIFARNet	17.54 / -	12	90.53	-	CNN	Manual
TSSL-BP (Zhang and Li, 2020)	CIFARNet	17.54 / -	5	91.41	-	CNN	Manual
STBP-tdBN (Zheng et al., 2021)	ResNet-19	12.63 / -	4	92.92	70.86	CNN	Manual
TET (Deng et al., 2022)	ResNet-19	12.63 / -	4	94.44	74.47	CNN	Manual
AutoSNN (Na et al., 2022)	AutoSNN (C=128)	21 / -	8	93.15	69.16	CNN	Auto
SNASNet (Kim et al., 2022)	SNASNet-Bw	- / -	8	94.12	73.04	CNN	Auto
SpikeDHS^{D} (Che et al., 2022)	SpikeDHS-CLA (n3s1)	14 / -	6	95.36	76.25	CNN	Auto
ANN	ResNet-19*	12.63	1	94.97	75.35	CNN	Manual
	Transformer-4-384	9.32 / 3.97	1	**96.73**	**81.02**	Transformer	Manual
Spikformer	Spikformer-4-256	4.15 / 0.553	4	93.94	75.96	Transformer	Manual
	Spikformer-2-384	5.76 / 0.582	4	94.80	76.95	Transformer	Manual
	Spikformer-4-384	9.32 / 0.952	4	95.19	77.86	Transformer	Manual
AutoST	AutoST Tiny	4.20 / -	4	95.14	76.29	Transformer	Auto
	AutoST base	29.64 / -	4	96.21	79.69	Transformer	Auto
Auto-Spikformer	Auto-Spikformer $\mathcal{S}_{T_s1}$	4.69 / **0.505**	4	**95.29**	**77.71**	Transformer	Auto
	Auto-Spikformer $\mathcal{S}_{T_s2}$	4.64 / **0.509**	4	**95.23**	**77.91**	Transformer	Auto
	Auto-Spikformer $\mathcal{S}_{T_l1}$	**7.09 / 0.826**	4	**96.19**	**78.01**	Transformer	Auto
	Auto-Spikformer $\mathcal{S}_{T_l2}$	**9.20 / 0.889**	4	**96.38**	**77.05**	Transformer	Auto
	Auto-Spikformer $\mathcal{S}_{T_l3}$	**8.46 / 0.925**	4	**96.39**	**78.22**	Transformer	Auto

Auto-Spikformer architectures $\mathcal{S}_{T_s1,2}$ and $\mathcal{S}_{T_l1,2,3}$ are selected from the candidate architectures listed in Table 3. Auto-Spikformer is the first transformer model designed through automated methods, demonstrating enhanced performance in both tasks. The symbol "*" denotes results obtained from self-implemented experiments by Deng et al. (2022). Bold value means the result is more optimal.

definition of this search space is outlined in Table 1, and its visual interpretation is depicted in Figure 3.

Transformer search space, denoted $\mathcal{S}_T$, is similar to the design of Autoformer (Chen M. et al., 2021), which includes four variable factors: embedding dimension, MLP ratio, head number, and depth. The structured definition of this search space is outlined in Table 1.

4.4 Accuracy and energy balanced fitness function ($\mathcal{F}_{AEB}$)

The original fitness function only considers accuracy. We propose a new fitness function $\mathcal{F}_{AEB}$ that balances accuracy and energy consumption. To estimate energy use, we first need to compute synaptic operations (SOPs). For a specific layer l, SOPs can be calculated as follows (Equation 2):

$$\mathrm{SOPs}(l) = fr \times t \times \mathrm{FLOPs}(l) \tag{2}$$

Here, fr denotes the firing rate of the input spike train, and t represents the time-step. Floating Point Operations (FLOPs) refer to the number of multiply-and-accumulate (MAC) operations, while SOPs contain spike-based accumulate (AC) operations only. The theoretical energy consumption of Auto-Spikformer, assuming implementation on the 45nm CMOS technology (Rathi and Roy, 2021) with $E_{MAC} = 4.6pJ$ and $E_{AC} = 0.9pJ$, is calculated as (Equation 3):

$$\mathbf{E} = E_{MAC} \times \mathrm{FL}^{1}_{\text{SNN Conv}} + E_{AC} \times \left(\sum_{n=2}^{N} \mathrm{SOP}^{n}_{\text{SNN Conv}} + \sum_{m=1}^{M} \mathrm{SOP}^{m}_{\text{SNN FC}} + \sum_{l=1}^{L} \mathrm{SOP}^{l}_{\text{SSA}} \right) \tag{3}$$

Here, $\mathbf{E}$ denotes the model energy, $\mathrm{FL}^{1}_{SNN\ Conv}$ is the first layer to encode static RGB images into spike-form, and the SOPs of n SNN Conv layers, m SNN Fully Connected Layer (FC), and l SSA are added together and multiplied by E_{AC}. For ANNs, the theoretical energy consumption of block b is calculated (Equation 4):

TABLE 5 Comparison of the performance with state-of-the-art (SOTA) methods on two neuromorphic datasets.

Method	Spikes	CIFAR10-DVS		DVS128	
		T Step	Acc	T Step	Acc
LIAF-Net (Wu et al., 2021)	✗	10	70.4	60	97.6
TA-SNN (Yao et al., 2021)	✗	10	72.0	60	98.6
Rollout (Kugele et al., 2020)	✓	48	66.8	240	97.2
DECOLLE (Kaiser et al., 2020)	✓	-	-	500	95.5
tdBN (Zheng et al., 2021)	✓	10	67.8	40	96.9
PLIF (Fang et al., 2021b)	✓	20	74.8	20	97.6
SEW-ResNet (Fang et al., 2021a)	✓	16	74.4	16	97.9
Dspike (Li et al., 2021)	✓	10	75.4*	-	-
SALT (Kim and Panda, 2021)	✓	20	67.1	-	-
DSR (Meng et al., 2022)	✓	10	77.3*	-	-
Spikformer (Zhou et al., 2022)	✓	10	78.9*	10	96.9
	✓	16	80.9*	16	98.3
Auto-Spikformer	✓	16	**81.2***	16	**98.6**

Bold font indicates the best; * denotes with Data Augmentation.

$$\text{Power}(b) = 4.6pJ \times \text{FLOPs}(b) \tag{4}$$

For SNNs, Power(b) is (Equation 5):

$$\text{Power}(b) = 0.9pJ \times \text{SOPs}(b) \tag{5}$$

The energy consumption of Spikformer is influenced by factors such as input image size, embedding dimension, number of blocks, firing rate fr, and time-step t. These factors can be adjusted by changing the transformer architecture and selecting suitable spike neuron parameters. For comparison, we normalized these factors using a minmax scaler and assigned different weights to both metrics. The accuracy and energy balanced fitness function $\mathcal{F}_{AEB}$ is described as follows (Equation 6):

$$\mathcal{F}_{AEB} = \alpha \times \mathbf{E} + (1 - \alpha) \times \mathcal{A} \tag{6}$$

Here, $\mathcal{A}$ denotes the top-1 accuracy, both metrics are scaled by a minmax scaler with the range (0,1), and α denotes the weight (set to 0.5 in our case).

5 Experiments

We provide comprehensive implementation details for the supernet training stage and the evolutionary search stage. Firstly, we evaluate the effectiveness of the proposed DSPS method by comparing it with random search and handcrafted design within the SNN search space. Then, we assess the effectiveness of the proposed $\mathcal{F}_{AEB}$ fitness function within the Transformer and SNN mixed search space. Finally, we evaluate the performance of the searched model on the CIFAR dataset and neuromorphic datasets, comparing it with the original Spikformer and various CNN or ViT models.

5.1 Implementation details

Supernet training stage. We followed a similar training manner as Spikformer, with an extended epoch duration of 1000 to ensure improved convergence of the supernet.

Evolutionary search stage. For the transformer search space, we adopt a similar approach to Autoformer. In the SNN search space, our proposed DSPS begins with 50 randomly generated sets, each specifying the decay rate (τ), threshold (u_{th}), and time-step (t). The fitness ($\mathcal{F}_{AEB}$) is evaluated on test data, with the top 20 selected as parents. Offspring are generated through mutation and crossover, utilizing probabilities Pd (0.3) and Pm (0.4). This process iterates for 20 generations, culminating in the identification of the optimal architecture.

Dataset. We conducted our experiments on the CIFAR dataset and neuromorphic datasets. **CIFAR** consists of 50,000 training and 10,000 test images with a resolution of 32 × 32 pixels. **CIFAR10-DVS** is a neuromorphic dataset derived from the CIFAR10 static image dataset, comprising 9,000 training and 1,000 test images with a resolution of 128 × 128 pixels. **DVS128 Gesture** is a gesture recognition dataset consisting of 11 hand gesture categories with a resolution of 128 × 128 pixels.

5.2 Effectiveness of DSPS

We train Auto-Spikformer within the SNN search space ($\mathcal{S}_S$), where only the SNN parameter sets are modified while maintaining the original Spikformer structure depicted in Table 1. We select 300 candidates through the proposed DSPS and the $\mathcal{F}_{AEB}$. We then plot energy and accuracy for each candidate and draw a Pareto frontier, as shown in Figure 5. Notably, by solely modifying the SNN inner parameter sets, a superior trade-off between energy consumption and accuracy can be achieved.

There is a moderate Kendall's tau rank correlation of 0.4 between the accuracy and the energy consumption. Some candidates exhibit lower energy consumption but higher accuracy, indicating that they are more optimal than others. The energy consumption within $\mathcal{S}_S$ is mainly determined by the firing rate, as the architecture is fixed. We select the candidates located on the Pareto frontier, as well as a subset of candidates with the top 20% accuracy, and present them in Table 2.

We observe that our fitness function and search algorithm favor a time-step of 4, which is the maximum value in $\mathcal{S}_S$. Furthermore, we aim to understand why different levels of energy consumption can result in similar accuracy. We notice that the network weights of these candidates are identical. Among them, the minimum energy consumption recorded is 0.79, while the maximum energy consumption is 0.99, resulting in a 25% difference. Remarkably, despite this significant divergence in energy consumption, the corresponding accuracies achieved are nearly equivalent.

As shown in Table 2, for a similar threshold value, the firing rate decreases as the decay parameter increases. The evolutionary search tends to adjust the tau parameter rather than the threshold to control the firing rate. The decay parameter in SNN has a profound effect on the firing rate by facilitating a memory effect for the previous membrane potential. Additionally, the decay and threshold parameters also affect the distribution of feature maps across the layers. Thus, by adjusting the tau and threshold values of each neuron, we can alter the firing rate and accuracy substantially. This shows that the proposed DSPS is a promising approach. By designing an appropriate search space and selecting a suitable fitness function, we are able to effectively decrease the overall firing rate while preserving the network's performance.

5.3 Effectiveness of $\mathcal{F}_{AEB}$

To demonstrate the superiority of $\mathcal{F}_{AEB}$, we conduct extensive experiments and illustrate the trade-off between energy and accuracy. We apply evolutionary search with $\mathcal{F}_{AEB}$ as the fitness function to generate 1000 samples in both $\mathcal{S}_{T_s}$ and $\mathcal{S}_{T_l}$. We then select the top 100 candidates based on their scores. For comparison, we also randomly sample 100 candidates from the search space. Additionally, we include the Spikformer architecture in the energy-accuracy plot.

As shown in Figure 6, the Pareto front of $\mathcal{F}_{AEB}$ dominates the random sample approach. The Kendall's tau rank correlation coefficients of evolutionary search and random sample are 0.63 and 0.08 in $\mathcal{S}_{T_s}$ and 0.60 and 0.24 in $\mathcal{S}_{T_l}$, respectively. The candidates on the Pareto front are listed in Table 3.

We observe numerous candidates that achieve a favorable balance between accuracy and energy consumption. In $\mathcal{S}_{T_s}$, some candidates on the frontier even surpass the original 4–384 Spikformer architecture in accuracy with only 2 blocks and 348 channels, meaning half of the energy consumption. $\mathcal{S}_{T_l}$ is used to further explore higher accuracy architecture, as shown in Table 3. The highest accuracy is 78.22 with lower energy consumption of 0.925 μJ. Furthermore, several candidates exhibited 10% to 25% less energy while achieving higher accuracy compared to the 4–384 Spikformer architecture.

5.4 Results on CIFAR

We select the Auto-Spikformer architecture searched in $\mathcal{S}_{T_s}$ and $\mathcal{S}_{T_l}$ in Section 5.3 and compare it with the original Spikformer and other methods. The performances are reported in Table 4. Auto-Spikformer is the first transformer model designed through automated methods. AutoST is another research work conducted during the same period, also focuses on spiking transformers but uses a training-free method to obtain suitable architecture candidates, which are then retrained from scratch. In contrast, our approach involves training a supernet and extracting candidates without the need for retraining. In terms of performance, AutoST's optimal architecture increases the number of parameters by nearly 3.5 times compared to our optimal model, achieving only a 1.5% accuracy improvement. Additionally, their minimum model's performance is lower than ours, with a 1.42% accuracy difference while using nearly the same number of parameters. Auto-Spikformer $\mathcal{S}_{T_s}2$ and $\mathcal{S}_{T_l}1,2,3$ outperform the state-of-the-art methods, including CNN or Transformer models that are manually or automatically designed, in both accuracy and energy consumption. The ANN-Transformer model is only 0.34% and 2.8% better than $\mathcal{S}_{T_l}1,2,3$ in CIFAR10/100, respectively, demonstrating that the Auto-Spikformer method is comparable to the ANN version.

5.5 Results on neuromorphic datasets

As the dimensions and depth of neuromorphic datasets differ from the CIFAR dataset, we design a new search space for neuromorphic datasets. Following the same supernet training approach and evolutionary search manner as the CIFAR dataset, we report the results in Table 5.

It can be observed that our model achieves impressive performance on both datasets while utilizing a smaller model size (Spikformer is 2.59M, our optimal choice 2.48M) and less energy. Specifically, on the DVS128 Gesture dataset, we achieve an accuracy of 98.6% using 16 time steps. Furthermore, our results are competitive with the TA-SNN model (98.6%, 60 time steps) (Yao et al., 2021), which employs floating-point spikes in the forward propagation process. Also, on the CIFAR10-DVS dataset, our Auto-Spikformer model outperforms the state-of-the-art methods in terms of accuracy. Compared to the original Spikformer, the Auto-Spikformer achieves a significant improvement in accuracy with even less energy consumption.

6 Conclusion

In this work, we are the first to propose a one-shot spiking transformer architecture search method for spiking-based vision transformers, named Auto-Spikformer. Auto-Spikformer optimizes both energy consumption and accuracy by incorporating critical parameters of SNN and transformers into the search space. We introduce two novel methods: Discrete Spiking Parameters Search (DSPS), which optimizes SNN parameters, and the Accuracy and Energy Balanced Fitness Function $\mathcal{F}_{AEB}$, designed to balance energy consumption and accuracy objectives. Extensive experiments demonstrate that the proposed algorithm significantly enhances the performance of Spikformer and uncovers numerous promising architectures. As part of our future work, we plan to extend our experiments to larger benchmark datasets.

Data availability statement

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

Author contributions

KC: Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing. ZZ: Writing – original draft, Writing – review & editing. JN: Writing – review & editing. ZM: Resources, Writing – review & editing. WF: Methodology, Writing – review & editing. YC: Methodology, Writing – review & editing. SS: Validation, Writing – review & editing. LY: Funding acquisition, Investigation, Methodology, Writing – review & editing. YT: Funding acquisition, Writing – review & editing.

Funding

The author(s) declare financial support was received for the research, authorship, and/or publication of this article. The study was funded by the National Natural Science Foundation of China under contracts Nos. 62332002, 62027804, 61825101, 62206141, and 62236009, the major key project of the Peng Cheng Laboratory (PCL2021A13), and Shenzhen Basic Research Program under Grant JCYJ20220813151736001. Computing support was provided by Pengcheng Cloudbrain.

Conflict of interest

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

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

References

Bu, T., Fang, W., Ding, J., Dai, P., Yu, Z., and Huang, T. (2021). "Optimal ann-snn conversion for high-accuracy and ultra-low-latency spiking neural networks," in *International Conference on Learning Representations*.

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

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. (2020). "End-to-end object detection with transformers," in *Proceedings of the European Conference on Computer Vision (ECCV)* (Springer), 213–229. doi: 10.1007/978-3-030-58452-8_13

Caucheteux, C., and King, J.-R. (2022). Brains and algorithms partially converge in natural language processing. *Commun. Biol.* 5, 1–10. doi: 10.1038/s42003-022-03036-1

Che, K., Leng, L., Zhang, K., Zhang, J., Meng, Q., Cheng, J., et al. (2022). "Differentiable hierarchical and surrogate gradient search for spiking neural networks," in *Advances in Neural Information Processing Systems*, 24975–24990.

Chen, B., Li, P., Li, C., Li, B., Bai, L., Lin, C., et al. (2021). "Glit: neural architecture search for global and local image transformer," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 12–21. doi: 10.1109/ICCV48922.2021.00008

Chen, M., Peng, H., Fu, J., and Ling, H. (2021). "Autoformer: searching transformers for visual recognition," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 12270–12280. doi: 10.1109/ICCV48922.2021.01205

Chen, X., Xie, L., Wu, J., and Tian, Q. (2019). "Progressive differentiable architecture search: bridging the depth gap between search and evaluation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 1294–1303. doi: 10.1109/ICCV.2019.00138

Chen, Y., Yang, T., Zhang, X., Meng, G., Xiao, X., and Sun, J. (2019). Detnas: backbone search for object detection," in *Advances in Neural Information Processing Systems*, 32.

Cheng, X., Zhong, Y., Harandi, M., Dai, Y., Chang, X., Li, H., et al. (2020). "Hierarchical neural architecture search for deep stereo matching," in *Advances in Neural Information Processing Systems*, 22158–22169.

Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., et al. (2020). Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*.

Chu, X., Wang, X., Zhang, B., Lu, S., Wei, X., and Yan, J. (2020). Darts-: robustly stepping out of performance collapse without indicators. *arXiv preprint arXiv:2009.01027*.

Deng, S., Li, Y., Zhang, S., and Gu, S. (2022). Temporal efficient training of spiking neural network via gradient re-weighting. *arXiv preprint arXiv:2202.11946*.

Dong, X., and Yang, Y. (2019). "One-shot neural architecture search via self-evaluated template network," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 3681–3690. doi: 10.1109/ICCV.2019.00378

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., et al. (2020). An image is worth 16x16 words: transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Elsken, T., Metzen, J. H., and Hutter, F. (2019). Neural architecture search: a survey. *J. Mach. Learn. Res.* 20, 1997–2017. doi: 10.48550/arXiv.1808.05377

Fang, W., Yu, Z., Chen, Y., Huang, T., Masquelier, T., and Tian, Y. (2021a). "Deep residual learning in spiking neural networks," in *Advances in Neural Information Processing Systems*, 34.

Fang, W., Yu, Z., Chen, Y., Masquelier, T., Huang, T., and Tian, Y. (2021b). "Incorporating learnable membrane time constant to enhance learning of spiking neural networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2661–2671. doi: 10.1109/ICCV48922.2021.00266

Fontaine, B., Peña, J. L., and Brette, R. (2014). Spike-threshold adaptation predicted by membrane potential dynamics in vivo. *PLoS Comput. Biol.* 10:e1003560. doi: 10.1371/journal.pcbi.1003560

Guo, J., Han, K., Wang, Y., Zhang, C., Yang, Z., Wu, H., et al. (2020). "Hit-detector: Hierarchical trinity architecture search for object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11405–11414. doi: 10.1109/CVPR42600.2020.01142

Han, B., Srinivasan, G., and Roy, K. (2020). "RMP-SNN: residual membrane potential neuron for enabling deeper high-accuracy and low-latency spiking neural network," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 13558–13567. doi: 10.1109/CVPR42600.2020.01357

Hassani, A., Walton, S., Shah, N., Abuduweili, A., Li, J., and Shi, H. (2021). Escaping the big data paradigm with compact transformers. *arXiv preprint arXiv:2104.05704*.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. (2022). "Masked autoencoders are scalable vision learners," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 16000–16009. doi: 10.1109/CVPR52688.2022.01553

Hu, Y., Tang, H., and Pan, G. (2021a). "Spiking deep residual networks," in *IEEE Transactions on Neural Networks and Learning Systems*, 1–6.

Hu, Y., Wu, Y., Deng, L., and Li, G. (2021b). Advancing residual learning towards powerful deep spiking neural networks. *arXiv preprint arXiv:2112.08954*.

Hunsberger, E., and Eliasmith, C. (2015). Spiking deep networks with lif neurons. *arXiv preprint arXiv:1510.08829*.

Jordan, J., Schmidt, M., Senn, W., and Petrovici, M. A. (2021). Evolving interpretable plasticity for spiking networks. *Elife* 10:e66273. doi: 10.7554/eLife.66273

Kaiser, J., Mostafa, H., and Neftci, E. (2020). Synaptic plasticity dynamics for deep continuous local learning (DECOLLE). *Front. Neurosci.* 14:424. doi: 10.3389/fnins.2020.00424

Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. (2020). "Transformers are rnns: Fast autoregressive transformers with linear attention," in *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 5156–5165.

Kim, Y., Li, Y., Park, H., Venkatesha, Y., and Panda, P. (2022). Neural architecture search for spiking neural networks. *arXiv preprint arXiv:2201.10355*. doi: 10.1007/978-3-031-20053-3_3

Kim, Y., and Panda, P. (2021). Optimizing deeper spiking neural networks for dynamic vision sensing. *Neural Netw.* 144:686–698. doi: 10.1016/j.neunet.2021.09.022

Kugele, A., Pfeil, T., Pfeiffer, M., and Chicca, E. (2020). Efficient processing of spatio-temporal data streams with spiking neural networks. *Front. Neurosci.* 14:439. doi: 10.3389/fnins.2020.00439

Lee, C., Sarwar, S. S., Panda, P., Srinivasan, G., and Roy, K. (2020). Enabling spike-based backpropagation for training deep neural network architectures. *Front. Neurosci.* 14:119. doi: 10.3389/fnins.2020.00119

Li, Y., Guo, Y., Zhang, S., Deng, S., Hai, Y., and Gu, S. (2021). "Differentiable spike: rethinking gradient-descent for training spiking neural networks," in *Advances in Neural Information Processing Systems* 34.

Lin, P., Sun, P., Cheng, G., Xie, S., Li, X., and Shi, J. (2020). "Graph-guided architecture search for real-time semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4203–4212. doi: 10.1109/CVPR42600.2020.00426

Liu, C., Chen, L.-C., Schroff, F., Adam, H., Hua, W., Yuille, A. L., et al. (2019). "Auto-deeplab: hierarchical neural architecture search for semantic image segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 82–92. doi: 10.1109/CVPR.2019.00017

Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L.-J., et al. (2018). "Progressive neural architecture search," in *Proceedings of the European conference on computer vision (ECCV)*, 19–34. doi: 10.1007/978-3-030-01246-5_2

Liu, H., Simonyan, K., and Yang, Y. (2018). Darts: differentiable architecture search. *arXiv preprint arXiv:1806.09055*.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., et al. (2021). "Swin transformer: hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 10012–10022. doi: 10.1109/ICCV48922.2021.00986

Lotfi Rezaabad, A., and Vishwanath, S. (2020). "Long short-term memory spiking networks and their applications," in *Proceedings of the International Conference on Neuromorphic Systems 2020 (ICONS)*, 1–9. doi: 10.1145/3407197.3407211

Meng, Q., Xiao, M., Yan, S., Wang, Y., Lin, Z., and Luo, Z.-Q. (2022). Training high-performance low-latency spiking neural networks by differentiation on spike representation. *arXiv preprint arXiv:2205.00459*.

Na, B., Mok, J., Park, S., Lee, D., Choe, H., and Yoon, S. (2022). Autosnn: towards energy-efficient spiking neural networks. *arXiv preprint arXiv*:2201.12738.

Neftci, E. O., Mostafa, H., and Zenke, F. (2019). Surrogate gradient learning in spiking neural networks: bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Process. Mag.* 36, 51–63. doi: 10.1109/MSP.2019.2931595

Nekrasov, V., Chen, H., Shen, C., and Reid, I. (2019). "Fast neural architecture search of compact semantic segmentation models via auxiliary cells," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9126–9135. doi: 10.1109/CVPR.2019.00934

Pham, H., Guan, M., Zoph, B., Le, Q., and Dean, J. (2018). "Efficient neural architecture search via parameters sharing," in *International Conference on Machine Learning* (PMLR), 4095–4104.

Qin, Z., Sun, W., Deng, H., Li, D., Wei, Y., Lv, B., et al. (2022). Cosformer: rethinking softmax in attention. *arXiv preprint arXiv:2202.08791*.

Rao, Y., Zhao, W., Liu, B., Lu, J., Zhou, J., and Hsieh, C.-J. (2021). "Dynamicvit: efficient vision transformers with dynamic token sparsification," in *Proceedings of the International Conference on Neural Information Processing Systems (NeurIPS)*, 13937–13949.

Rathi, N., and Roy, K. (2020). Diet-SNN: direct input encoding with leakage and threshold optimization in deep spiking neural networks. *arXiv preprint arXiv:2008.03658*.

Rathi, N., and Roy, K. (2021). "Diet-SNN: a low-latency spiking neural network with direct input encoding and leakage and threshold optimization," in *IEEE Transactions on Neural Networks and Learning Systems*.

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

Real, E., Aggarwal, A., Huang, Y., and Le, Q. V. (2019). "Regularized evolution for image classifier architecture search," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 4780–4789. doi: 10.1609/aaai.v33i01.33014780

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

Slowik, A., and Kwasnicka, H. (2020). Evolutionary algorithms and their applications to engineering problems. *Neural Comput. Applic.* 32, 12363–12379. doi: 10.1007/s00521-020-04832-8

Song, J.-G. (2021). Ufo-vit: high performance linear vision transformer without softmax. *arXiv preprint arXiv:2109.14382*.

Su, X., You, S., Xie, J., Zheng, M., Wang, F., Qian, C., et al. (2022). "Vitas: vision transformer architecture search," in *Computer Vision-ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XXI* (Springer), 139–157. doi: 10.1007/978-3-031-19803-8_9

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., and Jégou, H. (2021). "Training data-efficient image transformers distillation through attention," in *International Conference on Machine Learning* (PMLR), 10347–10357.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., et al. (2017). "Attention is all you need," in *Proceedings of the International Conference on Neural Information Processing Systems (NeurIPS)*, 30.

Wang, N., Gao, Y., Chen, H., Wang, P., Tian, Z., Shen, C., et al. (2020). "Nas-fcos: fast neural architecture search for object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11943–11951. doi: 10.1109/CVPR42600.2020.01196

Wang, W., Xie, E., Li, X., Fan, D.-P., Song, K., Liang, D., et al. (2021). "Pyramid vision transformer: a versatile backbone for dense prediction without convolutions," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 568–578. doi: 10.1109/ICCV48922.2021.00061

Wang, Y., Zhang, M., Chen, Y., and Qu, H. (2022). "Signed neuron with memory: towards simple, accurate and high-efficient ANN-SNN conversion," in *International Joint Conference on Artificial Intelligence*. doi: 10.24963/ijcai.2022/347

Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proc. IEEE* 78, 1550–1560. doi: 10.1109/5.58337

Whittington, J. C. R., Warren, J., and Behrens, T. E. (2022). "Relating transformers to models and neural representations of the hippocampal formation," in *International Conference on Learning Representations (ICLR)*.

Wu, Y., Deng, L., Li, G., Zhu, J., and Shi, L. (2018). Spatio-temporal backpropagation for training high-performance spiking neural networks. *Front. Neurosci.* 12:331. doi: 10.3389/fnins.2018.00331

Wu, Y., Deng, L., Li, G., Zhu, J., Xie, Y., and Shi, L. (2019). "Direct training for spiking neural networks: faster, larger, better," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 1311–1318. doi: 10.1609/aaai.v33i01.33011311

Wu, Z., Zhang, H., Lin, Y., Li, G., Wang, M., and Tang, Y. (2021). "Liaf-net: Leaky integrate and analog fire network for lightweight and efficient spatiotemporal information processing," in *IEEE Transactions on Neural Networks and Learning Systems*. doi: 10.1109/TNNLS.2021.3073016

Xiao, M., Meng, Q., Zhang, Z., Wang, Y., and Lin, Z. (2021). "Training feedback spiking neural networks by implicit differentiation on the equilibrium state," in *Advances in Neural Information Processing Systems*, 14516–14528.

Xiao, T., Singh, M., Mintun, E., Darrell, T., Dollár, P., and Girshick, R. (2021). "Early convolutions help transformers see better," in *Proceedings of the International Conference on Neural Information Processing Systems (NeurIPS)*, 30392–30400.

Xu, Y., Xie, L., Zhang, X., Chen, X., Qi, G.-J., Tian, Q., et al. (2019). Pc-darts: partial channel connections for memory-efficient architecture search. *arXiv preprint arXiv:1907.05737*.

Yang, J., Li, C., Zhang, P., Dai, X., Xiao, B., Yuan, L., et al. (2021). "Focal attention for long-range interactions in vision transformers," in *Proceedings of the International Conference on Neural Information Processing Systems (NeurIPS)*, 30008–30022.

Yao, M., Gao, H., Zhao, G., Wang, D., Lin, Y., Yang, Z., et al. (2021). "Temporal-wise attention spiking neural networks for event streams classification," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 10221–10230. doi: 10.1109/ICCV48922.2021.01006

Yao, M., Hu, J., Zhao, G., Wang, Y., Zhang, Z., Xu, B., et al. (2023). "Inherent redundancy in spiking neural networks," in *Proceedings of the*

IEEE/CVF International Conference on Computer Vision (ICCV), 16924–16934. doi: 10.1109/ICCV51070.2023.01552

Yuan, L., Chen, Y., Wang, T., Yu, W., Shi, Y., Jiang, Z.-H., et al. (2021a). "Tokens-to-token vit: training vision transformers from scratch on imagenet," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 558–567. doi: 10.1109/ICCV48922.2021.00060

Yuan, L., Hou, Q., Jiang, Z., Feng, J., and Yan, S. (2021b). Volo: vision outlooker for visual recognition. *arXiv preprint arXiv:2106.13112*.

Zhang, W., and Li, P. (2020). "Temporal spike sequence learning via backpropagation for deep spiking neural networks," in *Advances in Neural Information Processing Systems*, 12022–12033.

Zhang, Y., Qiu, Z., Liu, J., Yao, T., Liu, D., and Mei, T. (2019). "Customizable architecture search for semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11641–11650. doi: 10.1109/CVPR.2019.01191

Zheng, H., Wu, Y., Deng, L., Hu, Y., and Li, G. (2021). "Going deeper with directly-trained larger spiking neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 11062–11070. doi: 10.1609/aaai.v35i12.17320

Zhou, Z., Zhu, Y., He, C., Wang, Y., Shuicheng, Y., Tian, Y., et al. (2022). "Spikformer: when spiking neural network meets transformer," in *The Eleventh International Conference on Learning Representations*.

Zhu, X., Su, W., Lu, L., Li, B., Wang, X., and Dai, J. (2020). Deformable detr: deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*.

Zoph, B., and Le, Q. V. (2016). Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*.

Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. (2018). "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 8697–8710. doi: 10.1109/CVPR.2018.00907