N-SPIKFORMER: NESTED SPIKING TRANSFORMER

Anonymous authors
Paper under double-blind review

ABSTRACT

Spiking Transformers achieve strong representation capability while retaining the energy-efficient characteristics of spiking neural networks. However, adapting them to different resource constraints typically requires repeated training or fine-tuning, which is computationally expensive. In this paper, we propose a nested spiking Transformer (N-Spikformer) that directly provides multiple sub-models with different accuracy–size trade-offs from a single network to meet different resource constraints. First, we construct a nested spiking Transformer architecture by scoring and reordering attention heads and MLP hidden neurons according to their importance. Within the nested model, smaller sub-models use the higher-ranked heads and hidden neurons. In this way, each sub-model is nested within a larger one, and all sub-models share weights. Second, we develop a nested training strategy to efficiently optimize these shared-weight sub-models. Instead of training all sub-models at each update step, we sample one representative sub-model from four representative width ratios and progressively introduce smaller sub-models during training, reducing training overhead and alleviating optimization interference. Extensive experiments across multiple spiking Transformer architectures, image classification datasets, and dense prediction tasks show that N-Spikformer directly provides customized sub-models under different resource constraints while achieving comparable or better performance than independently trained models. These results demonstrate the practical value of N-Spikformer for deployment scenarios with varying resource constraints.

1 INTRODUCTION

Spiking Neural Networks (SNNs) have emerged as a promising energy-efficient alternative to conventional artificial neural networks (ANNs) (Maass, 1997; Gerstner & Kistler, 2002). Specifically, by transmitting information through discrete binary spikes, SNNs enable sparse event-driven computation and replace computationally intensive multiply-and-accumulate operations with energy-efficient accumulation operations (Yao et al., 2023). This energy efficiency has also driven the development of neuromorphic hardware, such as SpiNNaker (Painkras et al., 2013), TrueNorth (Merolla et al., 2014), Darwin (Shen et al., 2016), Loihi (Davies et al., 2018), and Tianjic (Pei et al., 2019), which in turn allows the low-power advantage of SNNs to be fully realized. However, early SNNs still suffered from limited performance in complex tasks (Fang et al., 2021; Hu et al., 2024).

Recent studies have explored Transformer architectures for SNNs and developed diverse spiking Transformers that substantially improve representation capability while retaining the energy-efficient nature of SNNs (Zhou et al., 2023; Yao et al., 2024b; Zhou et al., 2026; Yao et al., 2024a; Zhou et al., 2024a). However, due to their inherent multi-timestep nature, training spiking Transformers is time- and memory-intensive. For example, on ImageNet with 8 NVIDIA A100 GPUs, a 10M Spike-Driven Transformer V2 requires 63 hours for 200 epochs of training (Yao et al., 2024a; 2025), while an 11.7M ResNet-18 requires only about 6 hours for 90 epochs (He et al., 2016; Ismayilov & Mammadova, 2024). Consequently, customizing spiking Transformers for various deployment scenarios characterized by limited compute resources through repeated training is impractical.

To efficiently obtain customized spiking Transformers for different deployment scenarios, existing methods typically compress spiking Transformers through techniques such as quantization (Wei et al., 2025; Qiu et al., 2025; Cao et al., 2025; Wei et al., 2026a) and pruning (Liu et al., 2024; Zhou et al., 2024b; Zhuge et al., 2024; Wei et al., 2026b). These methods provide effective ways to reduce model size and computational cost. However, most of them still require additional fine-tuning

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

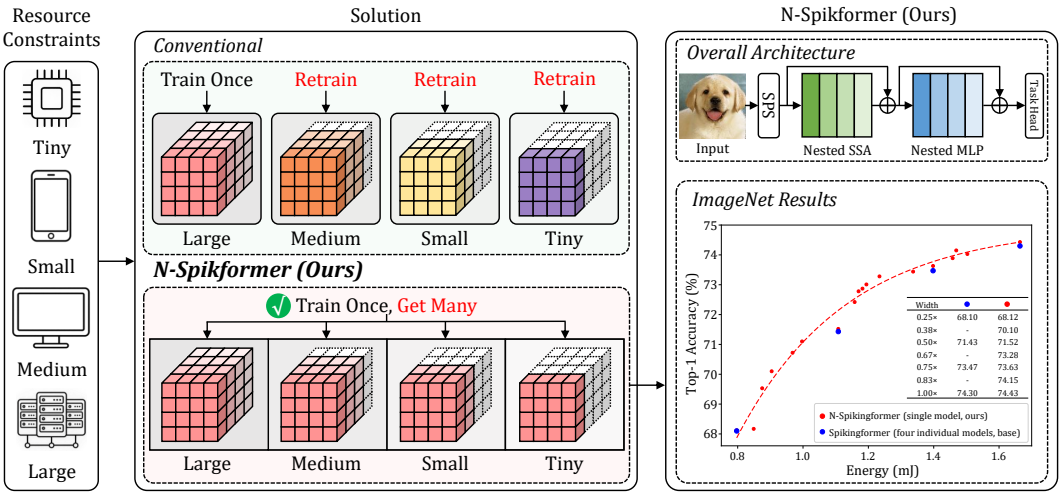

Figure 1: Illustration of nested architecture. Under different resource constraints, conventional methods require retraining individual models of different sizes, whereas nested architecture directly provides multiple shared-weight sub-models from a single nested model without additional training. The overall architecture and ImageNet accuracy–efficiency trade-offs of nested spiking Transformer (N-Spikformer) are also shown in the right panel.

(Wei et al., 2025; 2026a) or retraining (Liu et al., 2024; Zhuge et al., 2024; Zhou et al., 2024b; Qiu et al., 2025; Cao et al., 2025) when adapting to a new deployment scenario. This raises a question: "Given different resource constraints, can a single spiking Transformer directly yield corresponding customized models with different accuracy-size trade-offs without additional training?"

Realizing this goal is non-trivial and involves two challenges. First, existing spiking Transformer architecture do not support directly obtaining different models with favorable accuracy-size trade-offs, as shown in Table 3. A natural solution is to design the spiking Transformer as a nested architecture (Kusupati et al., 2022; Devvrit et al., 2024; Haberer et al., 2024; Cai et al., 2024; Zhu et al., 2025), as shown in Figure 1. In nested spiking Transformer, models of different sizes (sub-models) with shared weights are contained within a single network and each model can be directly extracted after training. Second, while a nested spiking transformer helps direct sub-model extraction, simultaneously training all sub-models at each step is computationally expensive, and naively optimizing multiple sub-models with shared weights may cause them to interfere with each other (Yu et al., 2018; Cai et al., 2020). This makes it difficult for all sub-models to perform well.

In this paper, we propose a nested spiking Transformer (N-Spikformer), which provides a set of sub-models with favorable accuracy-size trade-offs for diverse deployment scenarios through a single training process. Specifically, first, we construct a nested model by scoring and reordering attention heads and MLP hidden neurons. Attention heads and MLP hidden neurons with higher scores are placed at the front. For the nested model, smaller sub-models use the front heads and hidden neurons. In this way, each sub-model is nested within a larger one, and all sub-models share weights. Second, we develop a nested training strategy to efficiently train the sub-models and alleviate interference among them. Instead of training all sub-models at each update step, we select four representative sub-models (25%, 50%, 75%, and 100%) and uniformly sample one for each step. The training process consists of three stages: the first stage trains the 75% and 100% sub-models, while the 50% and 25% sub-models are progressively introduced in the subsequent stages. The experimental results demonstrate the effectiveness of N-Spikformer. The main contributions are summarized as follows:

- We propose a nested spiking Transformer (N-Spikformer) that scores and reorders attention heads and MLP hidden neurons according to their importance. N-Spikformer consists of multiple nested models of different sizes (sub-models) that share weights, enabling sub-models to be directly extracted from a single network without additional training.
- We develop a nested training strategy for N-Spikformer that samples a single representative sub-model during each update step and trains the sub-models progressively from larger to

smaller sizes. This strategy reduces the overhead of jointly training all sub-models and alleviates optimization interference among these shared-weight sub-models.

- Extensive experiments on image classification and dense prediction tasks across multiple spiking Transformer architectures demonstrate that N-Spikformer can directly provide sub-models with different accuracy–size trade-offs without additional retraining, while achieving comparable or better performance than independently trained models.

2 Related work

2.1 Nested architecture

Nested architecture aims to obtain multiple models of different sizes from a single network and has been extensively explored in ANNs (Hou et al., 2020; Kusupati et al., 2022; Beyer et al., 2023). Slimmable Networks (Yu et al., 2018) enable a single network to operate at several fixed widths, providing different accuracy-size trade-offs without independently training multiple models. Universal Slimmable Networks (Yu & Huang, 2019) extend this idea by supporting arbitrary widths within a continuous range. Once-for-All (Cai et al., 2020) further expands the nested space to width, depth, kernel size, and input resolution. MatFormer (Devvrit et al., 2024) introduces nested FFN blocks into Transformers, enabling multiple sub-models with different FFN widths. HydraViT (Haberer et al., 2024) expands the nested space in vision Transformers to embedding dimension and the number of attention heads. EA-ViT (Zhu et al., 2025) additionally considers network depth and designs a router to select sub-models under different computational budgets. Flextron (Cai et al., 2024) extends nested architectures to large language models that can meet different requirements. These studies have advanced nested learning in ANNs, enabling multiple sub-models to be obtained from a single network, which reduces the cost for repeated training. However, it remains underexplored in SNNs, where the inherent multi-timestep nature leads to much higher training cost.

2.2 Efficient spiking Transformer

Existing studies mainly employ quantization and pruning to obtain customized spiking Transformers for different resource-constrained deployment scenarios. For quantization, QP-SNN (Wei et al., 2025) combines weight quantization and structured pruning to construct lightweight spiking Transformers. QSD-Transformer (Qiu et al., 2025) extends low-bit quantization to spike-driven Transformers. BESTformer (Cao et al., 2025) further binarizes both weights and spiking self attention maps to 1 bit. S^2NN (Wei et al., 2026a) pushes weight quantization below 1 bit for further compression. For pruning, existing studies on spiking Transformers mainly focus on token-level pruning. SparseSpikformer (Liu et al., 2024) jointly performs weight and token pruning to reduce model parameters and computation. STATA (Zhuge et al., 2024) introduces an anchor token for token pruning to accelerate both training and inference. STSFP (Zhou et al., 2024b) further exploits spatial and temporal redundancy in spiking features to reduce training cost. Recently, TP-Spikformer (Wei et al., 2026b) develops an information-retaining token pruning method to reduce the overhead and maintain performance. These studies provide effective ways to obtain efficient spiking Transformers for specific deployment targets. However, most of them still require additional fine-tuning or retraining each time they adapt to a new deployment scenario.

3 Preliminary

Spiking neuron model. Spiking neurons serve as the basic computational units of SNNs, which integrates spatiotemporal inputs into membrane potentials and generates binary spikes for information transmission. Among various spiking neuron models (Maass & Bishop, 2001; Gerstner & Kistler, 2002; Gerstner, 1995), the Leaky Integrate-and-Fire (LIF) neuron model (Maass, 1997; Wu et al., 2018) is widely adopted due to its biological plausibility and implementation simplicity:

$$\mathbf{U}[t] = \beta \mathbf{H}[t-1] + \mathbf{X}[t], \tag{1}$$

$$\mathbf{S}[t] = \Theta(\mathbf{U}[t] - V_{\mathrm{th}}), \tag{2}$$

$$\mathbf{H}[t] = \begin{cases} \mathbf{U}[t](1-\mathbf{S}[t]) + V_{\mathrm{reset}}\mathbf{S}[t], & \text{hard reset}, \\ \mathbf{U}[t] - V_{\mathrm{th}}\mathbf{S}[t], & \text{soft reset}. \end{cases} \tag{3}$$

Here, t denotes the timestep, $\mathbf{X}[t]$ is the input from a convolution or linear operation, $\mathbf{U}[t]$ is the membrane potential before spike generation, and β is the decay factor. $\mathbf{S}[t]$ denotes the binary spike generated by the Heaviside step function $\Theta(\cdot)$ with firing threshold V_{th}. $\mathbf{H}[t]$ is the membrane potential after reset, and V_{reset} is the reset potential. At each timestep, the neuron integrates the temporal input $\beta\mathbf{H}[t-1]$ with the spatial input $\mathbf{X}[t]$. When $\mathbf{U}[t]$ exceeds the firing threshold, the neuron generates a spike and then updates the membrane potential through hard or soft reset.

Spiking Transformer. Spiking Transformers combine the energy efficiency of SNNs with the strong representation capability of Transformers. Typically, a spiking Transformer consists of an input embedding module, L spiking encoder blocks, and a classification head (CH). Given an input image sequence $\mathbf{X}_{\text{in}} \in \mathbb{R}^{T\times H_{\text{in}}\times W_{\text{in}}\times 3}$, the overall framework can be formulated as

$$\mathbf{X}_0 = \text{InputEmbedding}(\mathbf{X}_{\text{in}}),\ \mathbf{X}_l = \text{Encoder}(\mathbf{X}_{l-1}),\ l = 1,\dots,L,\ \mathbf{Y} = \text{CH}(\text{GAP}(\mathbf{X}_L)). \tag{4}$$

where $\mathbf{X}_l \in \mathbb{R}^{T\times H\times W\times C}$ denotes the output feature of the l-th block. Each encoder block follows a residual structure consisting of a spiking self-attention (SSA) block and an MLP block:

$$\hat{\mathbf{X}}_l = \mathbf{X}_{l-1} + \text{SSA}(\mathbf{X}_{l-1}), \quad \mathbf{X}_l = \hat{\mathbf{X}}_l + \text{MLP}(\hat{\mathbf{X}}_l), \tag{5}$$

For $\text{SSA}(\cdot)$, the spiking query $\mathbf{Q}$, key $\mathbf{K}$, and value $\mathbf{V}$ are generated through linear operations and spiking neurons. Then, they are used to compute the spiking self-attention output:

$$\mathbf{Q},\mathbf{K},\mathbf{V} = \text{SN}(\text{BN}(\text{Linear}(\text{SN}(\mathbf{X})))),\ \text{SSA}(\mathbf{X}) = \text{BN}(\text{Linear}_o(\text{SN}(\mathbf{Q}\mathbf{K}^{\text{T}}\mathbf{V}\cdot\alpha))). \tag{6}$$

where $\text{SN}(\cdot)$ denotes the LIF spiking neuron, $\text{BN}(\cdot)$ denotes batch normalization, $\text{Linear}(\cdot)$ denotes a learnable linear projection, and α is a scaling factor to control the attention output.

4 METHOD

In this section, we propose a nested spiking Transformer (N-Spikformer) that directly provides multiple sub-models with different accuracy–size trade-offs from a single network without additional retraining for different resource constraints. N-Spikformer contains two parts. First, we construct a nested architecture to enable direct sub-model extraction. Second, we develop a nested training strategy to reduce training overhead and alleviate interference among shared-weight sub-models.

4.1 NESTED SPIKING TRANSFORMER ARCHITECTURE

In this subsection, we aim to construct a set of nested sub-models $\{\mathcal{N}_w \mid w \in \mathcal{W}\}$ with different width ratios for spiking Transformer $\mathcal{N}$, where $\mathcal{W}$ denotes the set of width ratios. First, based on a pretrained spiking Transformer, we evaluate the importance of spiking attention heads and MLP neurons and rank them according to their scores. Second, based on the ranking results, we reorder the corresponding parameters and move higher-ranked heads and neurons to the front. Third, based on the reordered model, we construct nested SSA and MLP blocks for different width ratios. Finally, we prove that the nested spiking Transformer does not change the output of the pretrained model.

Scoring and ranking. Starting from a pretrained $\mathcal{N}$, we select $\mathcal{D}_s$ from the training data $\mathcal{D}_{\text{train}}$ to score each spiking attention head and MLP neuron. During scoring, the importance is measured by the average absolute input current. Specifically, for the i-th attention head in the l-th SSA block, we use the attention feature represented by the input current as the basis for importance scoring:

$$\mathbf{I}^{\text{SSA}} = (\mathbf{Q}\mathbf{K}^{\text{T}})\mathbf{V}\cdot\alpha, \quad s_{l,i}^{\text{SSA}} = \frac{1}{|\mathcal{D}_s|}\sum_{\mathbf{x}\in\mathcal{D}_s}\frac{1}{TN_l d_l}\sum_{t,n,c}\left|\mathbf{I}_{l,i}^{\text{SSA}}(\mathbf{x})_{t,n,c}\right|. \tag{7}$$

Here, $\mathbf{I}^{\text{SSA}}$ denotes the input current, and the inner summation is performed over all T timesteps, N_l tokens, and d_l channels for each attention head. Similarly, for the j-th hidden neuron in the l-th MLP block, we use its input current $\mathbf{I}^{\text{MLP}}$ as the feature for importance estimation:

$$\mathbf{I}^{\text{MLP}} = \text{BN}(\text{Conv}(\text{SN}(\hat{\mathbf{X}}))), \quad s_{l,j}^{\text{MLP}} = \frac{1}{|\mathcal{D}_s|}\sum_{\mathbf{x}\in\mathcal{D}_s}\frac{1}{TN_l}\sum_{t,n}\left|\mathbf{I}_{l,j}^{\text{MLP}}(\mathbf{x})_{t,n}\right|. \tag{8}$$

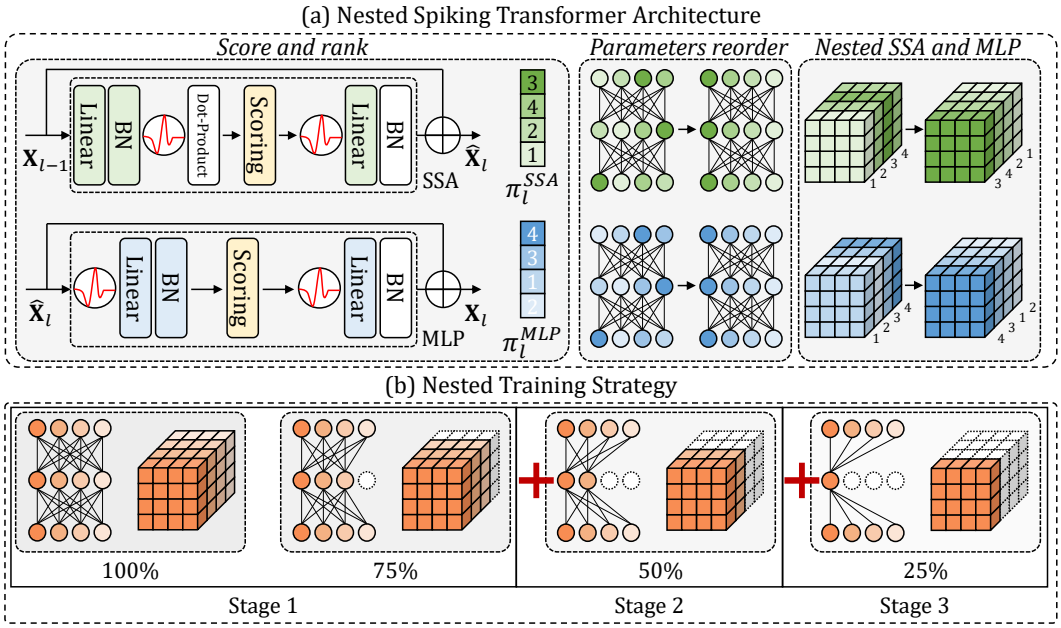

Figure 2: Overview of the proposed N-Spikformer. (a) The nested spiking Transformer architecture ranks attention heads and MLP hidden neurons by importance, reorders the corresponding parameters, and constructs nested SSA and MLP blocks. (b) The nested training strategy progressively introduces smaller sub-models from larger to smaller widths.

The inner summation is performed over all T timesteps and N_l tokens. After obtaining the importance scores, we rank the attention heads and MLP hidden neurons in descending order within each block. The resulting ranking orders are denoted as π_l^{SSA} and π_l^{MLP}, respectively. Since each attention head and MLP hidden neuron corresponds to fixed parameters, their ranking orders can be directly applied to the associated parameters to reorder.

Parameter reordering. Generally, for an attention head or hidden neuron, the corresponding parameters include the $\mathrm{Linear}(\cdot)$ and $\mathrm{BN}(\cdot)$ layers that generate its feature, as well as the subsequent $\mathrm{Linear}(\cdot)$ layer that takes this feature as input:

$$\hat{\mathbf{Z}}_l = \underline{\mathrm{BN}}(\underline{\mathrm{Linear}}(\mathbf{Z}_{l-1})), \quad \mathbf{Z}_l = \mathrm{BN}(\underline{\mathrm{Linear}}(\hat{\mathbf{Z}}_l)). \tag{9}$$

The underlined modules contain the parameters that need to be reordered. For both SSA and MLP blocks, we permute these associated parameters according to the corresponding ranking orders, such that higher-ranked attention heads and hidden neurons are placed at the front. Specifically, for the l-th SSA block, let $\tilde{\pi}_l^{\mathrm{SSA}}$ denote the channel permutation obtained by expanding each attention head in π_l^{SSA} to its d_l channels. The QKV $\mathrm{Linear}(\cdot)$ and output $\mathrm{Linear}(\cdot)$ are permuted as

$$\mathbf{W}'_{\mathrm{QKV}} = \mathrm{Permute}(\mathbf{W}_{\mathrm{QKV}}, \tilde{\pi}_l^{\mathrm{SSA}}), \quad \mathbf{W}'_o = \mathrm{Permute}(\mathbf{W}_o, \tilde{\pi}_l^{\mathrm{SSA}}). \tag{10}$$

The $\mathrm{BN}(\cdot)$ parameters and statistics following the QKV $\mathrm{Linear}(\cdot)$ are permuted similarly:

$$\boldsymbol{\gamma}' = \mathrm{Permute}(\boldsymbol{\gamma}, \tilde{\pi}_l^{\mathrm{SSA}}), \quad \boldsymbol{\beta}' = \mathrm{Permute}(\boldsymbol{\beta}, \tilde{\pi}_l^{\mathrm{SSA}}), \tag{11}$$

$$\boldsymbol{\mu}' = \mathrm{Permute}(\boldsymbol{\mu}, \tilde{\pi}_l^{\mathrm{SSA}}), \quad \boldsymbol{\sigma}'^2 = \mathrm{Permute}(\boldsymbol{\sigma}^2, \tilde{\pi}_l^{\mathrm{SSA}}). \tag{12}$$

For the l-th MLP block, let π_l^{MLP} denote the permutation order of hidden neurons. The weights of the first and second $\mathrm{Linear}(\cdot)$ layers are permuted as

$$\mathbf{W}'_1 = \mathrm{Permute}(\mathbf{W}_1, \pi_l^{\mathrm{MLP}}), \quad \mathbf{W}'_2 = \mathrm{Permute}(\mathbf{W}_2, \pi_l^{\mathrm{MLP}}), \tag{13}$$

where $\mathbf{W}_1$ and $\mathbf{W}_2$ denote the weights of the first and second $\mathrm{Linear}(\cdot)$ layers in the MLP block, respectively. The $\mathrm{BN}(\cdot)$ parameters following $\mathbf{W}_1$ are permuted using the same neuron order. In this way, we construct a reordered spiking Transformer architecture. Only the parameters associated with attention heads and hidden neurons are permuted, while other parameters remain unchanged.

Nested SSA block. After parameter reordering, different-width SSA blocks can be directly obtained by activating different numbers of attention heads. Specifically, for the l-th SSA block with H_l attention heads, a sub-model with width ratio w activates the first $h_l(w)$ attention heads:

$$h_l(w) = \max\left(1, \mathrm{round}(wH_l)\right). \tag{14}$$

Activating these heads is achieved by selecting the corresponding weights. Considering that each attention head contains d_l channels, the QKV and output $\mathrm{Linear}(\cdot)$ weights are extracted as

$$\mathbf{W}_{\mathrm{QKV},w} = \mathbf{W}'_{\mathrm{QKV}}[1 : h_l(w) \cdot d_l, :], \quad \mathbf{W}_{o,w} = \mathbf{W}'_o[:, 1 : h_l(w) \cdot d_l]. \tag{15}$$

Similarly, the corresponding $\mathrm{BN}(\cdot)$ parameters are selected from the first $h_l(w)d_l$ channels:

$$\boldsymbol{\gamma}_w = \boldsymbol{\gamma}'[1 : h_l(w)d_l],\ \boldsymbol{\beta}_w = \boldsymbol{\beta}'[1 : h_l(w)d_l],\ \boldsymbol{\mu}_w = \boldsymbol{\mu}'[1 : h_l(w)d_l],\ \boldsymbol{\sigma}^2_w = \boldsymbol{\sigma}'^2[1 : h_l(w)d_l]. \tag{16}$$

The selected attention heads are then used to compute the attention feature:

$$\mathbf{I}^{\mathrm{SSA}}_{l,w} = \mathrm{Concat}^{h_l(w)}_{i=1}\left[(\mathbf{Q}_i\mathbf{K}_i^{\mathrm{T}})\mathbf{V}_i \cdot \alpha\right],\ \mathrm{SSA}(\mathbf{X}_{l,w}) = \mathrm{Linear}_{o,w}\left(\mathrm{SN}(\mathbf{I}^{\mathrm{SSA}}_{l,w})\right). \tag{17}$$

Since smaller sub-models select subsets of the heads contained in larger sub-models, different-width SSA blocks naturally form a nested structure.

Nested MLP block. Similarly, different-width MLP blocks can be obtained by activating different numbers of hidden neurons from the reordered MLP block. Specifically, for the l-th MLP block with M_l hidden neurons, a sub-model with width ratio w activates the first $m_l(w)$ hidden neurons:

$$m_l(w) = \max\left(1, \mathrm{round}(wM_l)\right). \tag{18}$$

Then, the weights of the first $\mathrm{Linear}(\cdot)$, followed $\mathrm{BN}(\cdot)$ and second $\mathrm{Linear}(\cdot)$ are extracted as

$$\mathbf{W}_{1,w} = \mathbf{W}'_1[1 : m_l(w), :], \quad \mathbf{W}_{2,w} = \mathbf{W}'_2[:, 1 : m_l(w)]. \tag{19}$$

$$\boldsymbol{\gamma}_{1,w} = \boldsymbol{\gamma}'_1[1 : m_l(w)],\ \boldsymbol{\beta}_{1,w} = \boldsymbol{\beta}'_1[1 : m_l(w)],\ \boldsymbol{\mu}_{1,w} = \boldsymbol{\mu}'_1[1 : m_l(w)],\ \boldsymbol{\sigma}^2_{1,w} = \boldsymbol{\sigma}'^2_1[1 : m_l(w)]. \tag{20}$$

Using the selected parameters, the hidden feature and MLP output are computed as

$$\mathbf{I}^{\mathrm{MLP}}_{l,w} = \mathrm{BN}_{1,w}(\mathrm{Linear}_{1,w}(\mathrm{SN}(\hat{\mathbf{X}}_{l,w}))),\ \mathrm{MLP}(\hat{\mathbf{X}}_{l,w}) = \mathrm{Linear}_{2,w}(\mathrm{SN}(\mathbf{I}^{\mathrm{MLP}}_{l,w})). \tag{21}$$

In this way, the reordered spiking Transformer architecture enables direct extraction of multiple sub-models with different width ratios through nested SSA and MLP blocks. We provide an intuitive example in Appendix D for better understanding.

Equivalence analysis. Nested spiking Transformer changes the order of weights and features. A natural question is whether the reordered 100%-width model preserves the output of the original pretrained model. Let $\mathbf{P}$ denote the channel permutation matrix corresponding to the parameter reordering in Eq. 10–13. The reordered feature and output can be written as

$$\mathbf{Y} = \mathbf{W}_2\mathbf{W}_1\mathbf{Z}, \quad \mathbf{Y}' = \mathbf{W}'_2\mathbf{W}'_1\mathbf{Z} = \mathbf{W}_2\mathbf{P}^{-1}\mathbf{P}\mathbf{W}_1\mathbf{Z} = \mathbf{W}_2\mathbf{W}_1\mathbf{Z} = \mathbf{Y}. \tag{22}$$

Therefore, the proposed nested spiking Transformer preserves full mapping of the original model.

4.2 Nested training strategy

Although nested spiking Transformer allows sub-models with different width ratios to share weights, jointly training all sub-models at each update step introduces considerable training overhead and may cause optimization interference among them. To solve these two challenges, first, we select four representative sub-models with width ratios $\mathcal{W} = \{0.25, 0.50, 0.75, 1.00\}$ and sample only one sub-model at each update step to reduce the training overhead. Specifically, given the active width set $\mathcal{W}_s$, we uniformly sample a width w, apply it to all nested SSA and MLP blocks, and optimize the corresponding sub-model $\mathcal{N}_w$ using the cross-entropy loss:

$$w \sim \mathrm{Uniform}(\mathcal{W}_s), \qquad \mathcal{L}_w = \mathcal{L}_{\mathrm{CE}}(\mathcal{N}_w(\mathbf{x}), \mathbf{y}). \tag{23}$$

Notably, training all representative sub-models at each update step does not yield better performance, as shown in Table 3. Second, to further alleviate optimization interference among different-width sub-models, we divide the training process into three stages and progressively introduce smaller sub-models. The active width sets $\mathcal{W}_s$ in the three training stages are

$$\mathcal{W}_1 = \{0.75, 1.00\}, \qquad \mathcal{W}_2 = \{0.50, 0.75, 1.00\}, \qquad \mathcal{W}_3 = \{0.25, 0.50, 0.75, 1.00\}. \tag{24}$$

In this way, the larger sub-models are optimized first, while smaller sub-models are gradually introduced in subsequent stages. This ensures that each newly introduced sub-model is optimized on top of well-trained shared weights, thereby reducing interference during shared-weight optimization.

Table 1: Comparison of base models (Spikingformer (Zhou et al., 2026), QKFormer (Zhou et al., 2024a), and SDT-V3 (Yao et al., 2025)) with the proposed N-Spikformer sub-models at different width ratios on image classification datasets. Base models are trained individually for each ratio, while all nested N-Spikformer sub-models are directly obtained from a single training process and achieve comparable accuracy. All base results are reproduced by us for a fair comparison.

Dataset	Model	Ratio	Params (M)	Base Model		**Nested Sub-Model (Ours)**	
				Train	Acc. (%)	Train	Acc. (%)
CIFAR-10	Spikingformer	1.00×	7.11	Train Once	95.47	Train Once	95.87 (+0.40)
		0.75×	5.34	Retrain	95.37		95.93 (+0.56)
		0.50×	3.56	Retrain	95.20		95.82 (+0.62)
		0.25×	1.79	Retrain	95.12		95.28 (+0.16)
	QKFormer	1.00×	4.07	Train Once	96.11	Train Once	96.20 (+0.09)
		0.75×	3.09	Retrain	96.06		96.15 (+0.09)
		0.50×	2.11	Retrain	96.02		96.06 (+0.04)
		0.25×	1.13	Retrain	95.66		95.91 (+0.25)
CIFAR-100	Spikingformer	1.00×	7.11	Train Once	78.99	Train Once	79.63 (+0.64)
		0.75×	5.34	Retrain	78.74		79.88 (+1.14)
		0.50×	3.56	Retrain	78.49		79.64 (+1.15)
		0.25×	1.79	Retrain	78.23		78.85 (+0.62)
	QKFormer	1.00×	4.07	Train Once	80.92	Train Once	80.94 (+0.02)
		0.75×	3.09	Retrain	80.85		80.88 (+0.03)
		0.50×	2.11	Retrain	80.15		80.36 (+0.21)
		0.25×	1.13	Retrain	78.87		79.31 (+0.44)
CIFAR10-DVS	Spikingformer	1.00×	1.59	Train Once	81.90	Train Once	82.10 (+0.20)
		0.75×	1.19	Retrain	80.90		81.90 (+1.00)
		0.50×	0.79	Retrain	80.10		81.80 (+1.70)
		0.25×	0.40	Retrain	81.60		81.10 (−0.50)
	QKFormer	1.00×	0.48	Train Once	84.00	Train Once	83.90 (−0.10)
		0.75×	0.37	Retrain	83.50		84.10 (+0.60)
		0.50×	0.27	Retrain	82.90		83.00 (+0.10)
		0.25×	0.16	Retrain	81.50		81.90 (+0.40)
DVS-Gesture	Spikingformer	1.00×	1.59	Train Once	97.92	Train Once	98.26 (+0.34)
		0.75×	1.19	Retrain	98.26		98.26 (+0.00)
		0.50×	0.79	Retrain	97.92		98.26 (+0.34)
		0.25×	0.40	Retrain	98.26		98.26 (+0.00)
	QKFormer	1.00×	0.98	Train Once	98.61	Train Once	98.26 (−0.35)
		0.75×	0.75	Retrain	98.26		98.26 (+0.00)
		0.50×	0.51	Retrain	98.26		98.26 (+0.00)
		0.25×	0.28	Retrain	98.26		98.26 (+0.00)
ImageNet	SDT-V3	1.00×	4.91	Train Once	74.30	Train Once	74.43 (+0.13)
		0.75×	3.96	Retrain	73.47		73.63 (+0.16)
		0.50×	3.01	Retrain	71.43		71.52 (+0.09)
		0.25×	2.06	Retrain	68.10		68.12 (+0.02)

5 EXPERIMENT

In this section, we first introduce the experimental setup, including datasets, backbone architectures, and training details. Second, we conduct comparative studies on image classification and dense prediction tasks to evaluate the effectiveness of N-Spikformer. Finally, we perform ablation studies to analyze the contributions of each component and the sub-model selection strategy.

5.1 EXPERIMENTAL SETUP

We evaluate N-Spikformer across multiple spiking Transformer architectures and vision tasks. For small-scale classification, we use CIFAR-10, CIFAR-100 (Krizhevsky, 2009), DVS128-Gesture (Amir et al., 2017) and CIFAR10-DVS (Li et al., 2017) with Spikingformer (Zhou et al., 2026)

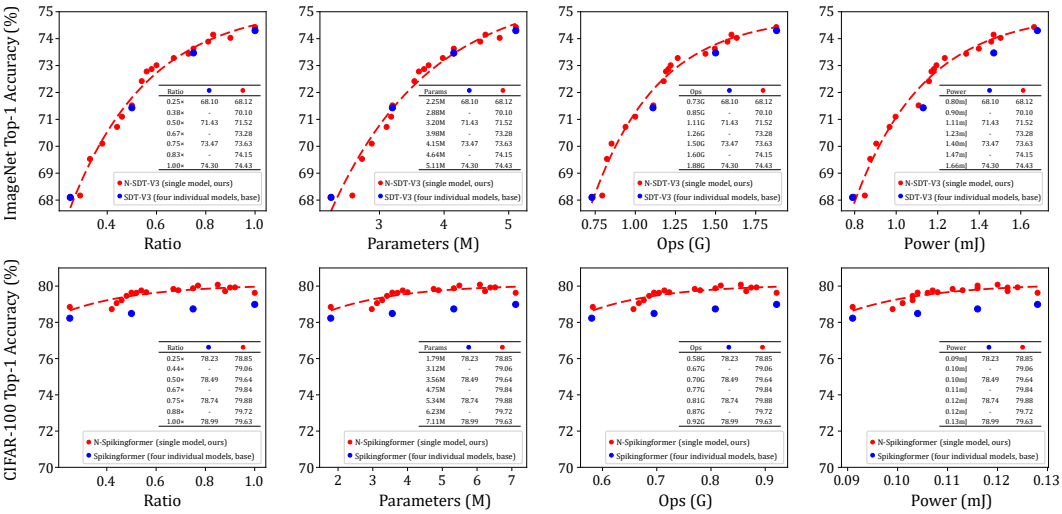

Figure 3: Accuracy–efficiency trade-offs of N-Spikformer sub-models on ImageNet and CIFAR-100 under different resource constraints. Without additional retraining, sub-models from a single N-Spikformer provide favorable trade-offs in width ratio , parameter, computation and power compared with independently trained SDT-V3 (Yao et al., 2025) and Spikingformer (Zhou et al., 2026).

Table 2: Comparison of the base model SDT-V3 (Yao et al., 2025) with the proposed N-Spikformer sub-models at different width ratios on semantic segmentation and object detection. For semantic segmentation, Acc. denotes mIoU (%), while for object detection, Acc. denotes mAP@50 (%).

Dataset	Ratio	Params (M)	Base Model			**Nested Sub-Model (Ours)**		
			Train	Power (mJ)	Acc.	Train	Power (mJ)	Acc.
ADE20K	1.00×	9.79	Train Once	15.16	40.10	Train Once	15.09	40.46 (+0.36)
	0.75×	7.88	Retrain	12.61	38.76		12.67	40.49 (+1.73)
	0.50×	5.97	Retrain	9.90	38.27		9.96	39.42 (+1.15)
	0.25×	4.06	Retrain	6.80	36.33		6.83	38.16 (+1.83)
COCO	1.00×	9.78	Train Once	40.54	43.40	Train Once	41.96	43.60 (+0.20)
	0.75×	7.87	Retrain	29.77	43.10		30.19	43.80 (+0.70)
	0.50×	5.97	Retrain	24.35	42.55		24.44	43.70 (+1.15)
	0.25×	4.06	Retrain	18.64	39.70		17.42	41.00 (+1.40)

and QKFormer (Zhou et al., 2024a) as backbones. For ImageNet-1K classification (Deng et al., 2009), ADE20K semantic segmentation (Zhou et al., 2017), and MS COCO object detection (Lin et al., 2014), we adopt Spike-Driven Transformer V3 (SDT-V3) (Yao et al., 2025). During training, stage 3 fully follows the original from-scratch training schedule, while stage 1 and stage 2 each use approximately 20% of the stage 3 epochs. The remaining settings, including optimizer, learning-rate schedule, weight decay, batch size, data augmentation (Zhang et al., 2018; Yun et al., 2019; Cubuk et al., 2020), and timesteps, follow the original training recipes.

5.2 Comparative Study

Image classification. Table 1 compares proposed N-Spikformer with independently trained base models across five image classification datasets and three spiking Transformer architectures. Under different resource constraints, such as parameter budgets, conventional spiking Transformers require retraining a separate model for each width ratio, whereas a single N-Spikformer directly yields the corresponding customized sub-models while achieving comparable or better accuracy. For example, On ImageNet with SDT-V3, the $0.75\times$ N-Spikformer sub-model achieves 73.63% accuracy, which is 0.16 percentage points higher than the independently trained base counterpart. More customized sub-models for CIFAR-100 and ImageNet datasets are reported in Figure 3, Table 5 and Table 6.

Semantic segmentation and object detection. Table 2 and Figure 4 report the semantic segmentation results on ADE20K and object detection results on MS COCO. Compared with independently trained SDT-V3 models, N-Spikformer directly provides sub-models at different width ratios from a single training process and consistently achieves higher accuracy. For example, on

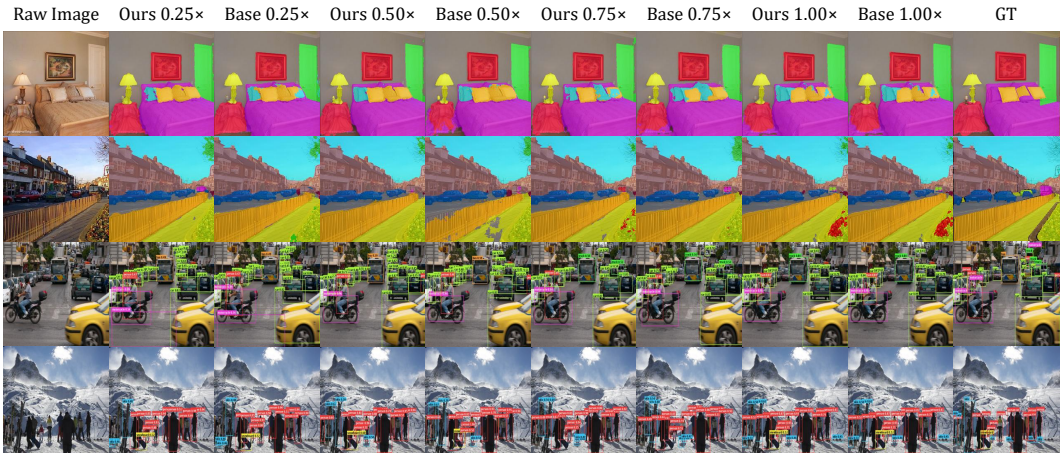

Figure 4: Qualitative results on semantic segmentation and object detection, demonstrating the generalization of N-Spikformer to dense prediction tasks.

Table 3: Ablation study of different variants.

Variant	0.25×	0.50×	0.75×	1.00×
Baseline	78.85	79.64	79.88	79.63
− Nested architecture	20.83	49.56	67.78	78.99
− Nested training	77.22	79.32	79.46	79.48
+ KD	77.86	79.67	79.72	79.62
→ Lower lr	75.49	78.42	78.74	79.03
→ Train all sub-models	76.04	78.09	78.60	79.04
→ Spike-weight scoring	78.23	79.38	79.92	79.38
→ Random scoring	77.19	79.26	79.34	79.60

Table 4: Ablation of different select strategies.

Config.	Order	Ops (G)	Acc@1
8S4M	Small-to-Large	0.823	68.82
4M8S	Large-to-Small	0.960	66.86
4S8M	Small-to-Large	0.881	69.42
8M4S	Large-to-Small	1.018	67.40
5M7L	Small-to-Large	1.264	72.30
7L5M	Large-to-Small	1.401	71.12
8L4X	Small-to-Large	1.589	73.78
4X8L	Large-to-Small	1.726	72.64

ADE20K with SDT-V3, the 0.25× N-Spikformer sub-model achieves 38.16% mIoU, outperforming the independently trained SDT-V3 model by 1.83 percentage points. These results demonstrate that N-Spikformer also generalizes well to dense prediction tasks.

5.3 Ablation Study

Table 3 evaluates the main design choices of N-Spikformer from three aspects: nested architecture, nested training strategy and scoring mechanism. First, removing the nested architecture causes severe accuracy degradation, showing that the original architecture cannot directly yield effective sub-models. Second, without the proposed nested training strategy, performance decreases across all width ratios. Besides, introducing knowledge distillation or using a lower learning rate does not improve performance. Training all sub-models at each update step also degrades accuracy while increasing the multiple training cost. Third, replacing our scoring with spike-weight scoring or random scoring brings no improvement, validating the proposed scoring strategy.

Table 4 compares different sub-model selection orders. Here, S, M, L, and X denote width ratios of 0.25×, 0.50×, 0.75×, and 1.00×, respectively. Small-to-Large consistently achieves higher accuracy with lower Ops than Large-to-Small. This indicates that, when extracting sub-models under the same parameter budget, allocating more parameters to deeper modules yields a better accuracy–efficiency trade-off. The results in Table 5 further support this observation.

6 Conclusion

This paper proposes N-Spikformer, a nested spiking Transformer that directly provides multiple sub-models with different accuracy–size trade-offs from a single model without additional retraining. First, a nested spiking Transformer architecture is developed by scoring and reordering attention heads and MLP hidden neurons, enabling sub-models with different heads and hidden neurons to be directly extracted. Second, a nested training strategy is introduced to effectively train the nested model. Extensive experiments across multiple architectures, classification datasets, and dense prediction tasks demonstrate its effectiveness for resource-constrained deployment scenarios.

AI USE STATEMENT

(This section is **required** and does not count toward the page limit.)

In this work, we used generative AI tools to assist with language polishing and minor code modification. All AI-assisted text was manually reviewed and revised by the authors to ensure accuracy and consistency with the intended meaning. AI-assisted code was inspected and tested by the authors to verify its correctness. We did not use generative AI tools to generate research results, experimental data, or scientific claims. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

(This section is **recommended** and does not count toward the page limit.)

Our implementation is based on SpikingJelly (Fang et al., 2023), PyTorch (Paszke et al., 2019), and the official implementations of Spikingformer (Zhou et al., 2026), QKFormer (Zhou et al., 2024a), and Spike-Driven Transformer V3 (Yao et al., 2025). The experimental results in this paper are reproducible. We provide the model architecture, training strategy, and experimental settings in the main text. All datasets used in this work are publicly available, and the corresponding preprocessing and data augmentation follow the original training recipes. Our codes will be available on GitHub.

REFERENCES

Arnon Amir, Brian Taba, David Berg, Timothy Melano, Jeffrey McKinstry, Carmelo Di Nolfo, Tapan Nayak, Alexander Andreopoulos, Guillaume Garreau, Marcela Mendoza, et al. A low power, fully event-based gesture recognition system. In *2017 IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 7388–7397. IEEE, 2017.

Lucas Beyer, Pavel Izmailov, Alexander Kolesnikov, Mathilde Caron, Simon Kornblith, Xiaohua Zhai, Matthias Minderer, Michael Tschannen, Ibrahim Alabdulmohsin, and Filip Pavetic. Flexivit: One model for all patch sizes. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14496–14506. IEEE, 2023.

Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HylxE1HKwS.

Ruisi Cai, Saurav Muralidharan, Greg Heinrich, Hongxu Yin, Zhangyang Wang, Jan Kautz, and Pavlo Molchanov. Flextron: many-in-one flexible large language model. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Honglin Cao, Zijian Zhou, Wenjie Wei, Yu Liang, Ammar Belatreche, Dehao Zhang, Malu Zhang, Yang Yang, and Haizhou Li. Binary event-driven spiking transformer. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence*, IJCAI '25, 2025. ISBN 978-1-956792-06-5. doi: 10.24963/ijcai.2025/458. URL https://doi.org/10.24963/ijcai.2025/458.

Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pp. 702–703, 2020.

M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, and et al. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99, 2018.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Fnu Devvrit, Sneha Kudugunta, Aditya Kusupati, Tim Dettmers, Kaifeng Chen, Inderjit S Dhillon, Yulia Tsvetkov, Hannaneh Hajishirzi, Sham M Kakade, Ali Farhadi, et al. Matformer: Nested

transformer for elastic inference. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Wei Fang, Zhaofei Yu, Yanqi Chen, Tiejun Huang, Timothée Masquelier, and Yonghong Tian. Deep residual learning in spiking neural networks. *Advances in neural information processing systems*, 34:21056–21069, 2021.

Wei Fang, Yanqi Chen, Jianhao Ding, Zhaofei Yu, Timothée Masquelier, Ding Chen, Liwei Huang, Huihui Zhou, Guoqi Li, and Yonghong Tian. Spikingjelly: An open-source machine learning infrastructure platform for spike-based intelligence. *Science Advances*, 9(40):eadi1480, 2023.

Wulfram Gerstner. Time structure of the activity in neural network models. *Physical review E*, 51 (1):738, 1995.

Wulfram Gerstner and Werner M Kistler. *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002.

Janek Haberer, Ali Hojjat, and Olaf Landsiedel. Hydravit: Stacking heads for a scalable vit. *Advances in Neural Information Processing Systems*, 37:40254–40277, 2024.

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*, pp. 770–778, 2016.

Mark Horowitz. 1.1 computing's energy problem (and what we can do about it). In *2014 IEEE international solid-state circuits conference digest of technical papers (ISSCC)*, pp. 10–14. IEEE, 2014.

Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. Dynabert: Dynamic bert with adaptive width and depth. *Advances in Neural Information Processing Systems*, 33: 9782–9793, 2020.

Yifan Hu, Lei Deng, Yujie Wu, Man Yao, and Guoqi Li. Advancing spiking neural networks toward deep residual learning. *IEEE transactions on neural networks and learning systems*, 36(2):2353–2367, 2024.

Elviz Ismayilov and Narmin Mammadova. Optimizing deep learning model training with cyclic data parallelism: A distributed computing approach. In *International Conference on Data Science and Management*, pp. 209–216. Springer, 2024.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, et al. Matryoshka representation learning. *Advances in Neural Information Processing Systems*, 35:30233–30249, 2022.

Hongmin Li, Hanchao Liu, Xiangyang Ji, Guoqi Li, and Luping Shi. Cifar10-dvs: an event-stream dataset for object classification. *Frontiers in neuroscience*, 11:244131, 2017.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pp. 740–755. Springer, 2014.

Yue Liu, Shanlin Xiao, Bo Li, and Zhiyi Yu. Sparsespikformer: A co-design framework for token and weight pruning in spiking transformer. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6410–6414. IEEE, 2024.

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

Wolfgang Maass and Christopher M Bishop. *Pulsed neural networks*. MIT press, 2001.

P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, and et al. A million spiking neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, 2014.

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.

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, 2019.

J. Pei, L. Deng, S. Song, M. Zhao, Y. Zhang, S. Wu, G. Wang, Z. Zou, Z. Wu, W. He, and et al. Towards artificial general intelligence with hybrid tianjic chip architecture. *Nature*, 572(7767): 106–111, 2019.

Xuerui Qiu, Malu Zhang, Jieyuan Zhang, Wenjie Wei, Honglin Cao, Junsheng Guo, Rui-Jie Zhu, Yimeng Shan, Yang Yang, and Haizhou Li. Quantized spike-driven transformer. In *International Conference on Learning Representations*, volume 2025, pp. 14477–14501, 2025.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pp. 618–626, 2017.

Juncheng Shen, De Ma, Zonghua Gu, Ming Zhang, Xiaolei Zhu, Xiaoqiang Xu, Qi Xu, Yangjing Shen, and Gang Pan. Darwin: a neuromorphic hardware co-processor based on spiking neural networks. *Science China Information Sciences*, 59(2):1–5, 2016.

Wenjie Wei, Malu Zhang, Zijian Zhou, Ammar Belatreche, Yimeng Shan, Yu Liang, Honglin Cao, Jieyuan Zhang, and Yang Yang. Qp-snn: Quantized and pruned spiking neural networks. In *International Conference on Learning Representations*, volume 2025, pp. 96470–96495, 2025.

Wenjie Wei, Malu Zhang, Jieyuan Eric Zhang, Ammar Belatreche, Shuai Wang, Yimeng Shan, Hanwen Liu, Honglin Cao, Guoqing Wang, Yang Yang, et al. S2nn: Sub-bit spiking neural networks. *Advances in Neural Information Processing Systems*, 38:45459–45487, 2026a.

Wenjie Wei, Xiaolong Zhou, Malu Zhang, Ammar Belatreche, Qian Sun, Yimeng Shan, Dehao Zhang, Zijian Zhou, Zeyu Ma, Yang Yang, et al. Tp-spikformer: Token pruned spiking transformer. In *The Fourteenth International Conference on Learning Representations*, 2026b.

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.

Man Yao, Guangshe Zhao, Hengyu Zhang, Yifan Hu, Lei Deng, Yonghong Tian, Bo Xu, and Guoqi Li. Attention spiking neural networks. *IEEE transactions on pattern analysis and machine intelligence*, 45(8):9393–9410, 2023.

Man Yao, JiaKui Hu, Tianxiang Hu, Yifan Xu, Zhaokun Zhou, Yonghong Tian, Bo Xu, and Guoqi Li. Spike-driven transformer v2: Meta spiking neural network architecture inspiring the design of next-generation neuromorphic chips. In *International conference on learning representations*, volume 2024, pp. 52885–52907, 2024a.

Man Yao, Jiakui Hu, Zhaokun Zhou, Li Yuan, Yonghong Tian, Bo Xu, and Guoqi Li. Spike-driven transformer. *Advances in neural information processing systems*, 36, 2024b.

Man Yao, Xuerui Qiu, Tianxiang Hu, Jiakui Hu, Yuhong Chou, Keyu Tian, Jianxing Liao, Luziwei Leng, Bo Xu, and Guoqi Li. Scaling spike-driven transformer with efficient spike firing approximation training. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

Jiahui Yu and Thomas S Huang. Universally slimmable networks and improved training techniques. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1803–1811, 2019.

Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. *arXiv preprint arXiv:1812.08928*, 2018.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6023–6032, 2019.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *2017 IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 5122–5130. IEEE, 2017.

Chenlin Zhou, Han Zhang, Zhaokun Zhou, Liutao Yu, Liwei Huang, Xiaopeng Fan, Li Yuan, Zhengyu Ma, Huihui Zhou, and Yonghong Tian. Qkformer: Hierarchical spiking transformer using qk attention. *Advances in Neural Information Processing Systems*, 37:13074–13098, 2024a.

Chenlin Zhou, Liutao Yu, Zhaokun Zhou, Han Zhang, Jiaqi Wang, Huihui Zhou, Zhengyu Ma, and Yonghong Tian. Spikingformer: A key foundation model for spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 2236–2244, 2026.

Zhaokun Zhou, Yuesheng Zhu, Chao He, Yaowei Wang, Shuicheng Yan, Yonghong Tian, and Li Yuan. Spikformer: When spiking neural network meets transformer. In *The Eleventh International Conference on Learning Representations*, 2023.

Zhaokun Zhou, Kaiwei Che, Jun Niu, Man Yao, Guoqi Li, Li Yuan, Guibo Luo, and Yuesheng Zhu. Spatial–temporal spiking feature pruning in spiking transformer. *IEEE Transactions on Cognitive and Developmental Systems*, 17(3):644–658, 2024b.

Chen Zhu, Wangbo Zhao, Huiwen Zhang, Yuhao Zhou, Weidong Tang, Shuo Wang, Zhihang Yuan, Yuzhang Shang, Xiaojiang Peng, Kai Wang, et al. Ea-vit: Efficient adaptation for elastic vision transformer. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1–10. IEEE, 2025.

Zhengyang Zhuge, Peisong Wang, Xingting Yao, and Jian Cheng. Towards efficient spiking transformer: a token sparsification framework for training and inference acceleration. In *Forty-first International Conference on Machine Learning*, 2024.

A THEORETICAL ENERGY EVALUATION OF N-SPIKFORMER

In this section, we provide theoretical energy evaluation of N-Spikformer. Following prior SNN works (Yao et al., 2024b; Horowitz, 2014), the energy consumed by one multiply accumulate (MAC) operation and one accumulate (AC) operation is defined as

$$E_{\mathrm{MAC}} = 4.6 \text{ pJ}, \qquad E_{\mathrm{AC}} = 0.9 \text{ pJ}. \tag{25}$$

We first calculate the dense operation count O_l of each convolutional or linear layer according to its actual tensor dimensions. For the l-th convolutional layer,

$$O_l^{\mathrm{Conv}} = H_l W_l C_l^{\mathrm{out}} \frac{C_l^{\mathrm{in}}}{G_l} K_{h,l} K_{w,l}, \tag{26}$$

where (H_l, W_l) is the output spatial resolution, C_l^{in} and C_l^{out} are the input and output channel numbers, G_l is the number of groups, and $(K_{h,l}, K_{w,l})$ is the kernel size. For the l-th linear layer,

$$O_l^{\mathrm{Linear}} = N_l^{\mathrm{out}} C_l^{\mathrm{in}}, \tag{27}$$

where N_l^{out} is the number of output elements and C_l^{in} is the input feature dimension. Then, we divide these layers into two sets. Layers receiving non-spiking inputs belong to $\mathcal{L}_{\mathrm{MAC}}$ and require MAC operations, whereas layers receiving binary spikes belong to $\mathcal{L}_{\mathrm{AC}}$ and require only AC operations. Therefore, the total number of MAC operations is

$$N_{\mathrm{MAC}} = \sum_{l \in \mathcal{L}_{\mathrm{MAC}}} O_l. \tag{28}$$

For each spike-driven layer, we measure its firing rate as

$$r_l = \frac{1}{M_l} \sum_{i=1}^{M_l} S_{l,i}, \tag{29}$$

where $S_{l,i} \in \{0, 1\}$ denotes a binary spike and M_l is the total number of spike elements over all evaluation samples, timesteps, channels, and spatial positions. The number of AC operations is then

$$N_{\mathrm{AC}} = \sum_{l \in \mathcal{L}_{\mathrm{AC}}} r_l O_l. \tag{30}$$

Finally, the theoretical inference energy is calculated as

$$E = N_{\mathrm{MAC}} E_{\mathrm{MAC}} + N_{\mathrm{AC}} E_{\mathrm{AC}}, \tag{31}$$

or equivalently,

$$E \text{ (mJ)} = \frac{4.6 N_{\mathrm{MAC}} + 0.9 N_{\mathrm{AC}}}{10^9}. \tag{32}$$

For each N-Spikformer sub-model, we perform an actual forward pass using its specific layer-wise width configuration and measure the firing rate of every spike-driven layer on the validation set. Thus, its energy consumption is calculated from the actual structure and firing activity, instead of being approximated by multiplying the full-model energy by an average width ratio.

B ADDITIONAL IMAGE CLASSIFICATION RESULTS OF N-SPIKFORMER

In this section, we report more results of N-Spikformer sub-models with different width ratios on ImageNet and CIFAR-100 in Table 5 and Table 6. Here, S, M, L, and X denote width ratios of $0.25\times$, $0.50\times$, $0.75\times$, and $1.00\times$, respectively. For ImageNet, to reduce the search cost, the attention heads and MLP hidden neurons within the same encoder use the same configuration. For CIFAR-100, the attention heads and MLP hidden neurons within the same encoder can use different configurations. For example, `MMSS:MMMS` at $0.42\times$ denotes attention-head ratios of $\{0.50, 0.50, 0.25, 0.25\}$ and MLP ratios of $\{0.50, 0.50, 0.50, 0.25\}$ across the four encoders.

Two main findings can be drawn from these results. First, beyond the four representative global ratios used during training ($0.25\times$, $0.50\times$, $0.75\times$, and $1.00\times$), a single N-Spikformer can directly provide many additional sub-models without retraining. These sub-models cover different parameter, computation, and power budgets while maintaining favorable accuracy–efficiency trade-offs, demonstrating the flexibility of N-Spikformer under different resource constraints. Second, on the more challenging ImageNet dataset, Small-to-Large configurations achieve better accuracy–efficiency trade-offs under similar parameter budgets, consistent with the findings in Section 5.3, while this tendency is less evident on the relatively simpler CIFAR-100 dataset.

Table 5: Performance of different sub-model configurations on ImageNet.

Ratio	Configuration	Params (M)	Ops (G)	Power (mJ)	Acc. (%)
0.25×	SSSSSSSSSSS	2.25	0.73	0.80	68.12
0.29×	SSSSSSSSSMM	2.58	0.79	0.85	68.17
0.33×	SSSSSSSMMMM	2.73	0.82	0.88	69.53
0.38×	SSSSSMMMMMM	2.88	0.85	0.90	70.10
0.44×	SSSMMMMMMMM	3.11	0.94	0.97	70.72
0.46×	SSMMMMMMMMM	3.18	1.00	1.00	71.10
0.50×	MMMMMMMMMMM	3.20	1.11	1.11	71.52
0.54×	MMMMMMMMMLL	3.54	1.18	1.16	72.42
0.56×	MMMMMMMMLLL	3.61	1.19	1.17	72.78
0.58×	MMMMMMMLLLL	3.69	1.21	1.18	72.87
0.60×	MMMMMMLLLLL	3.76	1.22	1.19	73.01
0.67×	MMMMLLLLLLL	3.98	1.26	1.23	73.28
0.73×	MLLLLLLLLLL	4.15	1.44	1.34	73.44
0.75×	LLLLLLLLLLL	4.15	1.50	1.40	73.63
0.81×	LLLLLLLLXXX	4.56	1.57	1.46	73.89
0.83×	LLLLLLLXXXX	4.64	1.59	1.47	74.15
0.90×	LLLLXXXXXXX	4.86	1.63	1.50	74.03
1.00×	LLLLLLLXXXX	5.11	1.88	1.66	74.43

Table 6: Performance of different sub-model configurations on CIFAR-100.

Ratio	Configuration	Params (M)	Ops (G)	Power (mJ)	Acc. (%)
0.25×	SSSS:SSSS	1.79	0.58	0.09	78.85
0.42×	MMSS:MMMS	2.97	0.66	0.10	78.73
0.44×	MMMS:MMMS	3.12	0.67	0.10	79.06
0.46×	MMMM:SMMM	3.27	0.68	0.10	79.22
0.48×	MMMS:MMMM	3.41	0.69	0.10	79.46
0.50×	MMMM:MMMM	3.56	0.70	0.10	79.64
0.52×	MMLM:MMMM	3.71	0.70	0.11	79.63
0.54×	MMXM:MMMM	3.86	0.71	0.11	79.76
0.56×	MMLM:MMLM	4.00	0.72	0.11	79.66
0.67×	MMMM:LLXM	4.75	0.77	0.11	79.84
0.69×	LLLM:LLLM	4.89	0.78	0.11	79.77
0.75×	LLLL:LLLL	5.34	0.81	0.12	79.88
0.77×	LMLL:XLXM	5.49	0.82	0.12	80.03
0.85×	XMXL:XLXL	6.08	0.86	0.12	80.08
0.88×	XMXX:XLLX	6.23	0.87	0.12	79.72
0.90×	XMXL:XLXX	6.38	0.87	0.12	79.92
0.92×	XXXX:XLXL	6.52	0.88	0.12	79.93
1.00×	XXXX:XXXX	7.11	0.92	0.13	79.63

C GRAD-CAM VISUALIZATION COMPARISION

In this section, we further provide a qualitative comparison of the visual representations on ImageNet. Specifically, Figure 5 presents Grad-CAM visualizations (Selvaraju et al., 2017) of N-Spikformer sub-models and independently trained SDT-V3 base models on ImageNet. The highlighted regions indicate the image areas that contribute most strongly to the predicted class. The results reveal two observations. First, N-Spikformer sub-models generally focus on more concentrated discriminative regions, whereas the four independently trained base models often exhibit more scattered attention regions. Second, N-Spikformer sub-models at different width ratios show highly similar highlighted regions. This is consistent with the shared-weight design of N-Spikformer, where different sub-models inherit representations from the same nested model.

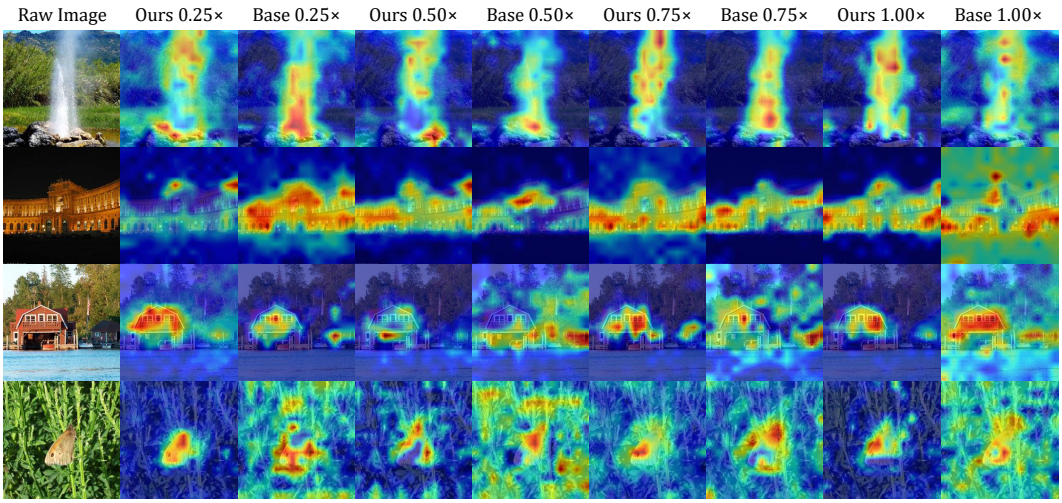

Figure 5: Grad-CAM visualizations of N-Spikformer sub-models and independently trained SDT-V3 base models at different width ratios on ImageNet. N-Spikformer sub-models generally exhibit more concentrated attention regions and more consistent highlighted regions across different widths.

D Illustrative example of N-Spikformer architecture

In this section, we further illustrate the construction process of N-Spikformer with a simple example. Consider an encoder block containing four attention heads $\{H_1, H_2, H_3, H_4\}$ and four MLP hidden neurons $\{M_1, M_2, M_3, M_4\}$. Suppose their importance scores are

$$\mathbf{s}^{\mathrm{SSA}} = [0.35, 0.62, 0.91, 0.78], \qquad \mathbf{s}^{\mathrm{MLP}} = [0.58, 0.21, 0.73, 0.90]. \tag{33}$$

Accordingly, their ranking orders are

$$\pi^{\mathrm{SSA}} = [3, 4, 2, 1], \qquad \pi^{\mathrm{MLP}} = [4, 3, 1, 2], \tag{34}$$

and the attention heads and MLP hidden neurons are reordered as

$$[H_3, H_4, H_2, H_1], \qquad [M_4, M_3, M_1, M_2]. \tag{35}$$

Their associated parameters are reordered using the same ranking orders. For SSA, since each attention head contains multiple channels, π^{SSA} is expanded to the corresponding channel permutation $\tilde{\pi}^{\mathrm{SSA}}$. Let $\mathbf{W}^i_{\mathrm{QKV}}$ denote the output-channel block of the QKV projection corresponding to the i-th attention head, and $\mathbf{W}^i_o$ denote the corresponding input-channel block of the output projection:

$$\mathbf{W}'_{\mathrm{QKV}} = \mathrm{Permute}(\mathbf{W}_{\mathrm{QKV}}, \tilde{\pi}^{\mathrm{SSA}}) = [\mathbf{W}^3_{\mathrm{QKV}}, \mathbf{W}^4_{\mathrm{QKV}}, \mathbf{W}^2_{\mathrm{QKV}}, \mathbf{W}^1_{\mathrm{QKV}}], \tag{36}$$

$$\mathbf{W}'_o = \mathrm{Permute}(\mathbf{W}_o, \tilde{\pi}^{\mathrm{SSA}}) = [\mathbf{W}^3_o, \mathbf{W}^4_o, \mathbf{W}^2_o, \mathbf{W}^1_o]. \tag{37}$$

The corresponding BN parameters and statistics are permuted using the same channel order. Similarly, for MLP, let $\mathbf{W}^i_1$ denote the output-channel block of the first linear layer corresponding to the i-th hidden neuron, and $\mathbf{W}^i_2$ denote the corresponding input-channel block of the second linear layer. Given $\pi^{\mathrm{MLP}} = [4, 3, 1, 2]$, we have

$$\mathbf{W}'_1 = \mathrm{Permute}(\mathbf{W}_1, \pi^{\mathrm{MLP}}) = [\mathbf{W}^4_1, \mathbf{W}^3_1, \mathbf{W}^1_1, \mathbf{W}^2_1], \tag{38}$$

$$\mathbf{W}'_2 = \mathrm{Permute}(\mathbf{W}_2, \pi^{\mathrm{MLP}}) = [\mathbf{W}^4_2, \mathbf{W}^3_2, \mathbf{W}^1_2, \mathbf{W}^2_2]. \tag{39}$$

The BN parameters following the first linear layer are also permuted according to π^{MLP}. After parameter reordering, sub-models with different width ratios can be directly obtained by selecting the front-ranked attention heads and MLP hidden neurons:

$$\begin{aligned} &0.25\times : \{H_3\}, && \{M_4\}, \\ &0.50\times : \{H_3, H_4\}, && \{M_4, M_3\}, \\ &0.75\times : \{H_3, H_4, H_2\}, && \{M_4, M_3, M_1\}, \\ &1.00\times : \{H_3, H_4, H_2, H_1\}, && \{M_4, M_3, M_1, M_2\}. \end{aligned} \tag{40}$$

Thus, each smaller sub-model is contained within a larger one and shares the corresponding parameters, forming the nested spiking Transformer architecture.

Table 7: More parameter and energy results of N-Spikformer across multiple datasets and models.

Model	Dataset	Ratio	$\text{Params}_{\text{Enc}}$ (M)	Params (M)	$\text{Energy}_{\text{Enc}}$ (mJ)	Energy (mJ)
Nested Spikingformer	CIFAR-10	1.00×	7.115	9.320	0.042	0.107
		0.75×	5.338	7.544	0.033	0.098
		0.50×	3.562	5.767	0.024	0.089
		0.25×	1.786	3.991	0.013	0.078
	CIFAR-100	1.00×	7.115	9.355	0.051	0.128
		0.75×	5.338	7.578	0.040	0.117
		0.50×	3.562	5.802	0.028	0.104
		0.25×	1.786	4.026	0.014	0.091
	CIFAR10-DVS	1.00×	1.585	2.569	0.008	0.083
		0.75×	1.190	2.173	0.006	0.081
		0.50×	0.794	1.778	0.004	0.080
		0.25×	0.399	1.382	0.002	0.078
	DVS-Gesture	1.00×	1.585	2.569	0.009	0.070
		0.75×	1.190	2.173	0.007	0.068
		0.50×	0.794	1.778	0.005	0.066
		0.25×	0.399	1.382	0.003	0.063
Nested QKFormer	CIFAR-10	1.00×	4.071	6.706	1.135	2.815
		0.75×	3.089	5.725	0.897	2.576
		0.50×	2.108	4.744	0.658	2.337
		0.25×	1.127	3.763	0.418	2.097
	CIFAR-100	1.00×	4.071	6.741	1.135	2.821
		0.75×	3.089	5.760	0.896	2.582
		0.50×	2.108	4.778	0.657	2.343
		0.25×	1.127	3.797	0.416	2.102
	CIFAR10-DVS	1.00×	0.481	1.504	0.136	0.572
		0.75×	0.373	1.397	0.112	0.548
		0.50×	0.266	1.290	0.088	0.523
		0.25×	0.159	1.183	0.064	0.499
	DVS-Gesture	1.00×	0.976	1.999	0.253	0.652
		0.75×	0.745	1.768	0.200	0.599
		0.50×	0.514	1.538	0.146	0.545
		0.25×	0.283	1.307	0.093	0.492
Nested SDT-V3	ImageNet	1.00×	4.913	5.106	1.727	1.727
		0.75×	3.961	4.154	1.451	1.451
		0.50×	3.009	3.202	1.147	1.147
		0.25×	2.057	2.250	0.814	0.815
	ADE20K	1.00×	9.787	11.485	15.093	40.050
		0.75×	7.879	9.578	12.671	37.506
		0.50×	5.972	7.670	9.955	34.990
		0.25×	4.064	5.762	6.831	31.730
	COCO	1.00×	9.782	21.172	41.963	49.848
		0.75×	7.874	19.265	30.191	43.985
		0.50×	5.967	17.357	24.444	36.950
		0.25×	4.059	15.449	17.418	28.037

E ADDITIONAL PARAMETER AND ENERGY RESULTS OF N-SPIKFORMER

In this section, we provide additional parameter and energy results of N-Spikformer with different backbone architectures across multiple datasets and tasks in Table 7. The results show that reducing the width ratio consistently decreases the encoder parameters ($\text{Params}_{\text{Enc}}$), full-model parameters (Params), encoder energy consumption ($\text{Power}_{\text{Enc}}$), and full-model energy consumption (Power). To better highlight the resource reduction brought by the nested SSA and MLP, we consistently report $\text{Params}_{\text{Enc}}$ and $\text{Power}_{\text{Enc}}$ in the main text as the parameter and energy metrics, respectively.