Appendix of Adaptive Calibration: A Unified Conversion Framework of Spiking Neural Networks

Ziqing Wang[1, 2*], **Yuetong Fang**[1*], **Jiahang Cao**[1], **Hongwei Ren**[1], **Renjing Xu**[1†]

[1]The Hong Kong University of Science and Technology (Guangzhou), China
[2]Northwestern University, USA
ziqingwang2029@u.northwestern.edu,
{yfang870, jcao248, hren066}@connect.hkust-gz.edu.cn, renjingxu@hkust-gz.edu.cn

Energy Evaluation

Design of Energy Estimator

For assessing SNNs' energy consumption for Pareto-frontier driven search, we draw upon established methodologies (Wang et al. 2023; Ding et al. 2021; Cao, Chen, and Khosla 2015), calculating energy based on the total number of spikes and their associated energy cost per spike, μ Joules:

$$E = \frac{\text{total spikes}}{1 \times 10^{-3}} \times \mu \quad \text{(in Watts)} \tag{1}$$

Theoretical Energy Consumption in Experiments

In the experimental section of our paper, we employ energy consumption as the metric for evaluating efficiency. This appendix delineates the methodology used to compute the theoretical energy consumption associated with our SNNs architecture. The computation process encompasses two primary phases: identifying the synaptic operations (SOPs) for each architectural component and estimating the cumulative energy consumption predicated on these operations.

Synaptic operations within each block of the SNN are calculated using the equation:

$$\text{SOPs}(l) = fr \times T \times \text{FLOPs}(l) \tag{2}$$

where l represents the block's ordinal number in the SNN, fr signifies the firing rate of the block's input spike train, T denotes the neuron's time step, and $\text{FLOPs}(l)$ refers to the block's floating-point operations, encapsulating the count of multiply-and-accumulate (MAC) operations. Herein, SOPs quantify the spike-based accumulation (AC) operations.

To ascertain the SNN's theoretical energy expenditure, we posit the implementation of MAC and AC operations on 45 nm technology, entailing energy costs of $E_{MAC} = 4.6\,\text{pJ}$ and $E_{AC} = 0.9\,\text{pJ}$, respectively. Following the methodologies delineated in (Cao et al. 2024; Yao et al. 2023), the formula for computing the SNN's theoretical energy consumption is as follows:

$$E_{SNN} = E_{MAC} \times \text{FLOP}^{1}_{\text{SNN}_{\text{Conv}}} + E_{AC} \times \left(\sum_{n=2}^{N} \text{SOP}^{n}_{\text{SNN}_{\text{Conv}}} + \sum_{m=1}^{M} \text{SOP}^{m}_{\text{SNN}_{\text{FC}}} \right) \tag{3}$$

Here, N and M denote the total count of convolutional (Conv) and fully connected (FC) layers, respectively. E_{MAC} and E_{AC} represent the energy costs per operation for MAC and AC, respectively. $\text{FLOP}_{\text{SNN}_{\text{Conv}}}$ pertains to the FLOPs of the initial Conv layer, while $\text{SOP}_{\text{SNN}_{\text{Conv}}}$ and $\text{SOP}_{\text{SNN}_{\text{FC}}}$ refer to the SOPs for the n^{th} Conv and m^{th} FC layers, respectively.

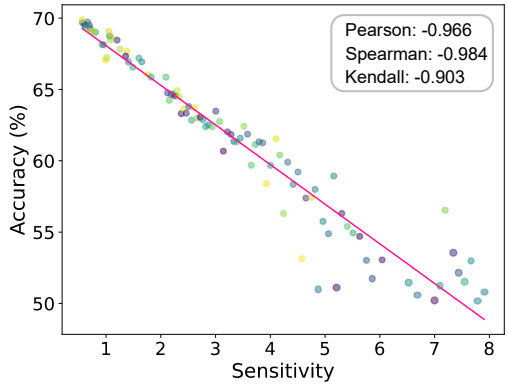


Figure 1: **Illustration of the Inverse Relationship between Accuracy and Sensitivity.** This figure details the layer-specific parameter settings and their impact on model performance. Sensitivity values are plotted on the x-axis against the corresponding performance metrics on the y-axis. The demonstrated negative correlation highlights that configurations with lower sensitivity tend to yield higher performance.

Performance Evaluation

Design of Performance Estimator

In the body of our paper, we introduced the concept of sensitivity to evaluate the performance of Spiking Neural Networks (SNNs). This appendix aims to further elucidate and expand upon the inverse relationship between sensitivity and

*These authors contributed equally.
†Corresponding Author

SNN performance, a foundational aspect of our findings. Herein, we delve into the quantitative analysis that substantiates this relationship, utilizing Kullback-Leibler (KL) divergence as our primary tool for measuring layer sensitivity.

Kullback-Leibler (KL) divergence, a statistical measure for gauging the difference between two probability distributions, serves as our metric for assessing the disparity in output distributions between Artificial Neural Networks (ANNs) and SNNs configurations across each neural layer. The sensitivity of a given layer i with respect to parameter k is expressed mathematically as:

$$S_i(k) = \frac{1}{N}\sum_{j=1}^{N} \mathrm{KL}\left(\mathcal{M}\left(\mathrm{ANN_i}; x_j\right), \mathcal{M}\left(\mathrm{SNN_i(k)}; x_j\right)\right) \tag{4}$$

where N denotes the number of inputs x_j assessed. A lower $S_i(k)$ signifies that the output of the SNN model closely mirrors that of the ANN model for the specified layer and parameter, indicating reduced sensitivity to parameter variations and, consequently, higher model performance.

Fig. 1 showcases the empirical relationship between layer sensitivity and overall SNN performance. By plotting sensitivity values against performance metrics for various configurations, we observe a pronounced negative correlation (Pearson coefficient = -0.966, $p < 10^{-122}$; Spearman coefficient = -0.984, $p < 10^{-156}$; Kendall coefficient = -0.903, $p < 10^{-77}$). This observation is pivotal, as it underscores the principle that diminishing sensitivity strongly correlates with the performance of SNNs.

Conversion Errors in ANN-to-SNN Conversion

The performance gap between ANNs and SNNs can largely be attributed to three main conversion errors: clipping error, quantization error, and unevenness error (Bu et al. 2021a; Hao et al. 2023a,b). These errors arise during the process of converting continuous values in ANNs to discrete spike-based representations in SNNs.

Clipping Error

Clipping error occurs due to the different ranges in ANNs and SNNs. In ANNs, outputs are continuous, while in SNNs, they are discrete. Clipping error arises when high values in ANNs are mapped to a single maximum value in SNNs.

Quantization Error

Quantization error results from mapping continuous ANN values to discrete SNN values. This error occurs because each continuous interval in ANNs is mapped to a fixed discrete value in SNNs.

Unevenness Error

Unevenness error is caused by the irregular timing of spikes in SNNs, leading to deviations from expected output values. This happens when spike arrival times vary, affecting the number of spikes received by neurons in deeper layers.

Upon analyzing these errors, we find that unevenness error is the most significant contributor to the performance gap. Therefore, our method primarily targets the reduction of this error to enhance the overall accuracy and efficiency of SNNs.

Theoretical Analysis of SNN Calibration

In SNNs, inputs are transmitted through the neuronal units, typically the Integrate-and-Fire (IF) spiking neuron in ANN-to-SNN conversions (Ding et al. 2021; Li et al. 2021a; Bu et al. 2021b):

$$u^{(\ell)}(t+1) = v^{(\ell)}(t) + W^{(\ell)} s^{(\ell)}(t) \tag{5}$$

$$v^{(\ell)}(t+1) = u^{(\ell)}(t+1) - s^{(\ell+1)}(t) \tag{6}$$

$$s^{(\ell+1)}(t) = \begin{cases} V_{th}^{(\ell)} & \text{if } u^{(\ell)}(t+1) \geq V_{th}^{(\ell)} \\ 0 & otherwise \end{cases} \tag{7}$$

where $u^{(\ell)}(t+1)$ denotes the membrane potential of neurons before spike generation, $v^{(\ell)}(t+1)$ denotes the membrane potential of neurons in layer ℓ at time step $t+1$, corresponding to the linear transformation matrix W^{ℓ}, the threshold $V_{th}^{(\ell)}$, and binary input $s^{\ell}(t)$ of layer ℓ.

The fundamental principle of ANN-to-SNN conversion is to ensure that the converted SNN closely approximates the input-output function mapping of the original ANN:

$$x^{(\ell)} \approx \overline{s}^{(\ell)} = \frac{1}{T}\sum_{t=0}^{T} s^{(\ell)}(t) \tag{8}$$

where $x^{(\ell)}$ represents the activation input of the ANN model, and $\overline{s}^{(\ell)}$ denotes the averaged binary input over T timesteps in the converted SNN. It is important to note that this approximation becomes valid only as T approaches infinity.

To address this limitation, Li et al.(Li et al. 2021b) introduced a layer-wise Calibration algorithm that minimizes the discrepancy between the outputs of the original ANN and the converted SNN.

We first establish the relationship between $\overline{s}^{(\ell)}$ and $\overline{s}^{(\ell+1)}$. Assuming an initial membrane potential $v^{(\ell)}(0) = 0$, by substituting Eq. 3 into Eq. 4 and summing over T, we obtain:

$$v^{(\ell)}(T) = W^{(\ell)}\left(\sum_{t=0}^{T} s^{(\ell)}(t)\right) - \sum_{t=0}^{T} s^{(\ell+1)}(t) \tag{9}$$

At each time step, the output can be either 0 or $V_{th}^{(\ell)}$. Therefore, the accumulated output $\sum_{t=0}^{T} s^{(\ell+1)}(t)$ can be written as $mV_{th}^{(\ell)}$, where $m \in 0, 1, \ldots, T$ denotes the total number of spikes. Assuming that the terminal membrane potential $v^{(\ell)}(T)$ lies within the range $\left[0, V_{th}^{(\ell)}\right)$, we have:

$$TW^{(\ell)}\overline{s}^{(\ell)} - V_{th}^{(\ell)} < mV_{th}^{(\ell)} \leq TW^{(\ell)}\overline{s}^{(\ell)} \tag{10}$$

Given this, we can determine m using the floor and clip operations:

$$m = Clip\left(\left\lfloor \frac{T}{V_{th}^{(\ell)}} W^{(\ell)}\overline{s}^{(\ell)} \right\rfloor, 0, T\right) \tag{11}$$

The clip function sets an upper bound T and a lower bound of 0, while the floor function $\lfloor x \rfloor$ returns the greatest integer less than or equal to x. The expected output spike can then be calculated as:

$$\begin{aligned}\overline{s}^{(\ell+1)} &= ClipFloor\left(W^{(\ell)}\overline{s}^{(\ell)}, T, V_{th}^{(\ell)}\right) \\ &= \frac{V_{th}^{(\ell)}}{T} Clip\left(\left\lfloor \frac{T}{V_{th}^{(\ell)}} W^{(\ell)}\overline{s}^{(\ell)} \right\rfloor, 0, T\right)\end{aligned} \tag{12}$$

Then SNN Calibration optimizes the threshold of spiking neurons using Eq.12:

$$\min_{V_{th}} \left(ClipFloor\left(\overline{s}^{(\ell+1)}, T, V_{th}^{(\ell)}\right) - ReLU\left(\overline{s}^{(\ell+1)}\right)\right)^2 \tag{13}$$

Moreover, to align the outputs of ANNs and SNNs, SNN Calibration incorporates the expected conversion errors into the bias terms:

$$b_i^{(\ell)} = b_i^{(\ell)} + \mu_i\left(e^{(\ell+1)}\right) \tag{14}$$

where $\mu_i\left(e^{(\ell+1)}\right)$ computes the mean error between the ANN and SNN outputs in the i^{th} channel.

In our work, we adopt the SNN Calibration framework to reduce the need for re-training. However, previous studies have neglected the energy efficiency of the converted SNNs and require extended inference timesteps to achieve performance comparable to directly trained SNNs. To address these issues, we propose an Adaptive Calibration method as a unified framework that enhances both performance and efficiency.

Pareto-frontier driven Search Algorithm of Adaptive Calibration

Motivation

To achieve an optimal balance between performance and energy consumption in our model, we utilize a Pareto-frontier driven Search Algorithm. This approach is driven by the need to simultaneously optimize two conflicting objectives: minimizing energy consumption and maximizing model performance. In our framework, performance is quantified through layer-wise sensitivity, which serves as a surrogate for direct performance metrics.

The search process is divided into two key components, each leveraging the Pareto frontier:

1. **Max Firing Pattern Search**: Given a fixed energy budget E_{target}, we search for the combination of firing patterns φ across layers that minimizes the total layer-wise sensitivity S_{sum}. This process allows us to find the optimal configuration that maximizes performance within the energy constraints, effectively moving along the Pareto frontier to identify the best trade-off point.
2. **Threshold Ratio Search**: In the second stage, we focus on reducing energy consumption further. Given a fixed sensitivity threshold S_{target}, we search for the combination of threshold amplification factors ρ across layers that minimize energy consumption E_{sum}. By doing so, we identify the configuration that achieves the desired performance while minimizing energy use, again optimizing along the Pareto frontier.

Both components of the algorithm are designed to ensure that the model's performance and energy consumption are optimally balanced, providing a robust solution that meets the specified constraints for either sensitivity or energy. This approach allows for a systematic exploration of the trade-offs between performance and energy, ensuring that the final model configuration is both efficient and effective.

Analysis of Search Space

Defining the search space is a critical step in optimizing both the firing patterns φ and threshold ratios ρ across the layers of the model. The search space for each component is outlined as follows:

Max Firing Pattern Search Space To identify the optimal firing pattern φ for each layer, we first establish a target energy budget E_{target}, which corresponds to the energy consumption when all layers adopt the target φ. The search space for φ is defined as follows:

- The minimum firing pattern count $\min \varphi$ is set to 1 (or 2 for the ImageNet dataset to ensure stability).
- The maximum firing pattern count $\max \varphi$ is set to twice the value of the target φ, providing a broad range for optimization.

This search space allows for a comprehensive exploration of possible configurations, ensuring that the algorithm can identify the optimal φ for each layer under the given energy constraints.

Threshold Ratio Search Space For the threshold ratio ρ, which aims to minimize energy consumption under a fixed sensitivity threshold S_{target}, the search space is defined as follows:

- The minimum threshold amplification factor $\min \rho$ is set to 1, representing the baseline configuration.
- The maximum threshold amplification factor $\max \rho$ is set to 10, with incremental steps of 0.5 to provide fine-grained control over the compression process.

This search space is designed to allow the algorithm to explore various configurations that balance energy savings with the preservation of model accuracy.

Pseudo-code

The optimization process for determining the optimal maximum firing pattern φ utilizes a dynamic programming-based approach, inspired by the method proposed by Cai et al. (2020). The optimization of the threshold ratio ρ follows a similar method, with a different target function corresponding to energy minimization under a fixed sensitivity threshold. The pseudo-code for this optimization process is presented in Algorithm 1.

Algorithm 1: Pareto Frontier Driven Search Algorithm

Input : Training data $train\_loader$, AdaFire model $model$, configuration range φ_range, energy budget E_{target}

Output : Optimal configuration $optimal\_phi$

// Initialize model for ANN mode to compute baseline
$model$.set_mode(ANN);
$model$.eval();
for *each* $(input, target)$ *in* $train\_loader$ **do**
  $input$ = $input$.to(device=$model$.device);
  $target$ = $target$.to(device=$model$.device);
  gt_output = $model$(input);
  break; `// Only need one batch for baseline`

// Sensitivity and energy analysis for each layer configuration in SNN mode
for *each layer* i *in* $model.layers$ **do**
  for *each configuration* φ *in* φ_range **do**
    $model$.layers[i].set_configuration(φ);
    $model$.set_mode(SNN);
    $output$ = $model$(input);
    kl_div = symmetric_kl(F.softmax($output$, dim=1), F.softmax(gt_output, dim=1));
    $spike\_count$ = $model$.layers[i].spike_counter;

// Pareto frontier search for optimal configuration
Initialize root node with cost 0 and profit 0;
$current\_list$ = [root];
for *each layer* $layer\_id$ **do**
  $next\_list$ = [];
  for *each node* n *in* $current\_list$ **do**
    for *each configuration* φ *in* φ_range **do**
      new_cost = $n.cost + energy\_result[layer\_id][\varphi]$;
      new_profit = $n.profit + sen\_result[layer\_id][\varphi]$;
      // Create new node with updated cost and profit
      new_node = Node(new_cost, new_profit, configuration=φ, parent=n);
      $next\_list$.append(new_node);
  // Prune nodes based on cost and profit
  $next\_list$.sort(key=lambda x: x.cost);
  $current\_list$ = prune_nodes($next\_list$, constraint=E_{target});

// Retrieve the best configuration
$best\_node$ = min($current\_list$, key=lambda node: node.profit);
$optimal\_phi$ = trace_back_config($best\_node$);

return $optimal\_phi$ `// Return the optimal configuration that minimizes sensitivity`

Results and Analysis

The results of the optimization process for both the firing pattern φ and the threshold ratio ρ are presented in Tables 1 and 2, respectively. These results highlight the effectiveness of the Pareto-frontier driven search in achieving a balanced trade-off between energy and performance.

Firing Pattern Search Results Table 1 displays the optimal firing pattern configurations for the CIFAR-10/100 datasets. The allocation of higher φ values in the initial and final layers underscores the importance of these layers in feature extraction and classification. The gradual reduction in resolution as the network deepens allows for moderate increases in φ values in later layers, optimizing energy consumption without significantly affecting overall performance.

Threshold Ratio Search Results Table 2 shows the optimal threshold amplification factors ρ for the CIFAR-10/100 datasets. The strategic selection of ρ values, with lower values in the initial layers and higher values in the final layers, reflects an adaptive approach to compression. This approach ensures that energy consumption is minimized while maintaining model accuracy. The higher ρ values in the final layers, particularly for CIFAR-10, indicate a tolerance for more aggressive compression in these layers, allowing for significant energy savings without compromising performance.

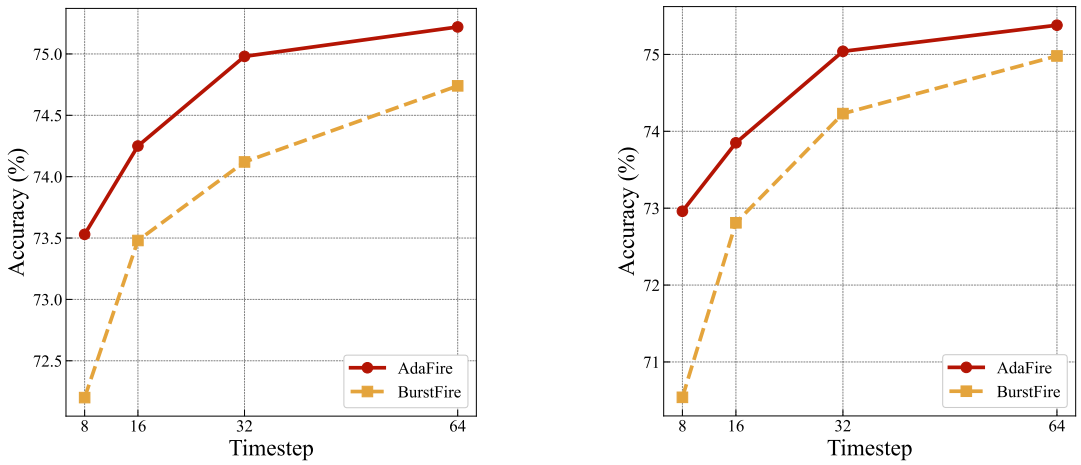
Comparison of AdaFire and BurstFire To further validate the effectiveness of the optimized firing patterns, we compare our Adaptive-Firing Neuron Model (AdaFire), which utilizes the optimal configurations derived from the Pareto-frontier driven search, with the conventional Burst-Firing Neuron Model (BurstFire), where each layer uniformly adopts the target firing pattern φ.

As shown in Figure 2, AdaFire consistently outperforms BurstFire, particularly in low timestep settings. For instance, Figure 2b demonstrates a notable performance gain of 2.45% at a timestep of 8 on the ResNet34 architecture. These results highlight the advantages of our adaptive approach in tailoring firing patterns to the specific sensitivities of each layer, leading to enhanced accuracy and efficiency compared to the static approach of BurstFire.

Summary of Results In summary, the Pareto-frontier driven Search Algorithm effectively balances the conflicting objectives of energy and performance, providing a robust framework for optimizing both firing patterns and threshold ratios across different layers of the model. The comparison between the AdaFire and BurstFire models further underscores the advantages of an adaptive approach, demonstrating superior accuracy and efficiency in the AdaFire model, particularly under low timestep constraints.

Input-aware Adaptive Timesteps

The efficiency and performance of SNNs are significantly impacted by the number of computational timesteps (T) during operation. Traditionally, T has been a fixed hyperparameter, leading to a trade-off: increasing T can enhance accuracy at the cost of higher computational latency. This trade-off is evident in the performance of spiking ResNet architectures on the CIFAR-10 dataset across different timesteps.



(a) VGG-16 (b) ResNet-34

Figure 2: **Effectiveness of AdaFire Neuron Model**. Compared with the BurstFire method, our method can largely improve performance on both VGG16 and ResNet34 in the ImageNet dataset, especially at low timestep.

Table 1: Adaptive-firing Neuron Model Configuration Search Results for CIFAR datasets. Target φ is set to 4.

Dataset	**Layer Number (φ Value)**																		
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
CIFAR-10	7	5	3	1	3	3	4	4	6	5	5	2	7	7	7	7	8	8	5
CIFAR-100	8	3	4	1	4	1	4	8	6	4	6	3	6	8	8	8	8	8	8

Table 2: Sensitivity Spike Compression (SSC) Search Results for CIFAR datasets

Dataset	**Layer Number (ρ Value)**																		
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
CIFAR-10	1.0	1.5	1.0	3.5	1.5	2.0	1.5	1.0	2.5	2.0	2.0	2.5	1.5	2.5	1.5	1.5	3.5	4.5	10.0
CIFAR-100	1.0	1.0	1.0	3.0	1.5	2.0	2.0	1.5	2.0	1.0	1.0	1.0	1.0	2.0	1.0	1.5	1.5	2.5	6.0

Motivation of Adaptive Timestep Strategy

Our analysis (see Fig. 3) reveals the inherent trade-off between accuracy and latency in fixed timestep configurations. For instance, increasing the timestep count from 4 to 8 boosts top-1 accuracy by only 0.4%, while doubling the computational time. This observation suggests that many inputs do not require the maximum number of timesteps for correct classification, highlighting the potential for efficiency gains through a more adaptive approach.

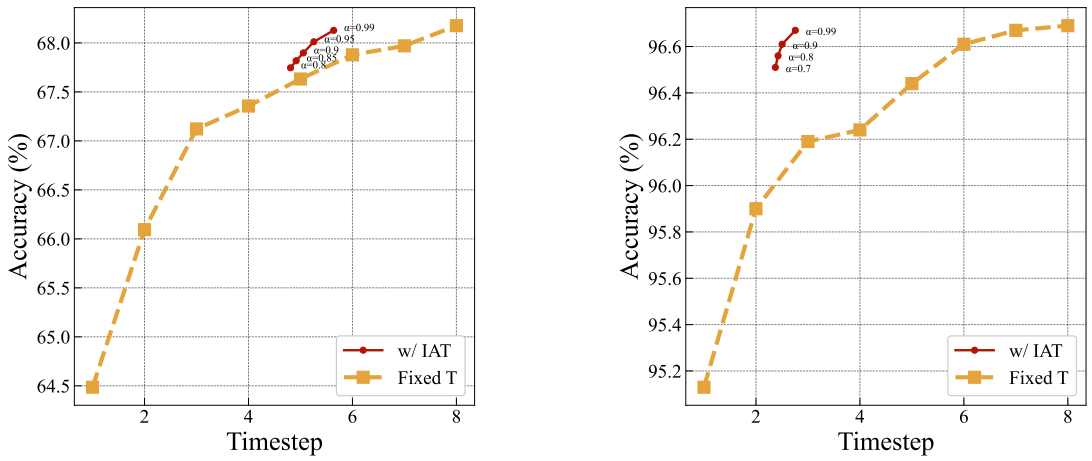
Building on this insight, we propose an adaptive timestep technique that dynamically adjusts T based on the complexity of each input. By defining a difficulty factor for each image, the SNN can determine the necessary number of timesteps (t) for accurate classification, thereby reducing redundant computations for simpler inputs. This approach is grounded in the following assumption:

Monotonic Correctness Assumption: *If an SNN can accurately predict an input x within t timesteps ($t \leq T$), it will continue to do so for any $t' \geq t$, up to T.*

This assumption guarantees that once an accurate classification is achieved at a specific timestep, any computations beyond that point up to T will not affect the outcome. Determining the earliest t for precise prediction enables the elimination of superfluous computations, thereby significantly enhancing network efficiency while preserving accuracy.

Implementation

The implementation of the input-aware adaptive timestep technique involves creating a mechanism within the SNN to evaluate the complexity of each input in real-time and adjust T accordingly. Unlike previous approaches that use a single fixed threshold α across all timesteps, our method considers the unique impact of each timestep on the network's final accuracy. Analyzing the entropy distribution across timesteps for the training dataset reveals significant variances, underscoring the limitation of a uniform threshold. This leads to the



(a) ImageNet (b) CIFAR-10

Figure 3: **Effectiveness of IAT method**. Compared with the baseline SNN with fixed timesteps, our method can largely reduce the latency while achieving higher performance.

development of our input-aware adaptive timestep technique, which sets a dynamic threshold of confidence scores at each timestep. In the body of the paper, we design the threshold of confidence score at each timestep by the following formula:

$$\alpha_t = \alpha_{base} + \beta e^{-\frac{\bar{E}_t - \bar{E}_{min}}{\delta}} \tag{15}$$

where α_{base} is the base threshold, β is the scaling factor, δ represents the decay constant, $\bar{E}_t$ denotes the average entropy of the network's output distribution at timestep t, and $\bar{E}_{min}$ the minimum average entropy observed across all timesteps.

By configuring β to $1 - \alpha_{base}$ and δ to 1, the dynamic range of α_t effectively spans from α_{base} to 1. This configuration allows for a flexible and responsive adaptation of the confidence threshold based on the entropy observed at each timestep, optimizing the SNN's computational efficiency by customizing timesteps to the complexity of each input, thus reducing unnecessary computations for simpler inputs while maintaining accuracy for more complex ones.

The pseudo-code for this algorithm is shown in Algorithm 2.

Results and Analysis

As shown in Fig. 3, the application of the IAT technique significantly reduces inference latency while maintaining or improving performance. For instance, employing the IAT technique allows our method to achieve 67.75% accuracy with only 4.8 timesteps on one dataset, outperforming the fixed timestep approach by 0.39%. Similarly, on the CIFAR-10 dataset, the IAT technique achieves 96.61% accuracy with just 2.5 timesteps, which is 0.42% higher than the SNN with three fixed timesteps. These examples demonstrate that our method effectively reduces latency and enhances performance simultaneously.

Algorithm 2: Input-aware Adaptive Timesteps

Input : Pretrained ANN, training samples, test dataset, α_{base}, β, δ
Output : Test accuracy, Acc_{test}

// Calibration and setup
for *each layer* $i = 1$ *to* n *in the ANN* **do**
  Calibrate parameters of layer i using training samples.

// Compute baseline for adaptive thresholding
for *each timestep* $t = 1$ *to* T **do**
  Compute $\bar{E}_t$, the average entropy of the network's output at timestep t, over the training dataset.
  Compute $\alpha(t)$ using the formula:
  $\alpha_t = \alpha_{\text{base}} + \beta e^{-\frac{\bar{E}_t - \bar{E}_{\text{min}}}{\delta}}$.

$\text{Acc}_{\text{test}} = 0$ `// Initialize test accuracy`

// Inference, processing images one by one
for *each image* i *in the test dataset* **do**
  $O = 0$ `// Initialize accumulated output`
  for $t = 1$ *to* T **do**
    $O_t = \text{ANN}(i, t)$ `// Output at timestep` t
    $O = O + O_t$ `// Accumulate output`
    $H(t) = \text{Entropy}(O)$
    if $H(t) > \alpha(t)$ **then**
      $\hat{y} = \arg\max O$ `// Make prediction`
      break
  Update Acc_{test} based on prediction accuracy of $\hat{y}$ for image i.

return Acc_{test} `// Final test accuracy`

Implementation Details

Computer Resources

We use RTX 4090 for training and inference. All the GPU time evaluations and performance evaluations are on the 4090 server.

Static Classification Datasets

This section details the static image datasets employed in our study, underscoring their pivotal role in training and testing our models. These datasets are selected for their diverse and comprehensive representation of real-world visual categories, which is crucial for evaluating the model's performance across a wide range of object classes.

CIFAR-10. Introduced by LeCun et al. (LeCun et al. 1998), the CIFAR-10 dataset consists of 60,000 32×32 color images in 10 classes, with 6,000 images per class. The dataset is divided into 50,000 training images and 10,000 testing images. Classes include vehicles and animals, such as "truck," "car," "bird," and "cat," making it a popular choice for evaluating classification algorithms.

CIFAR-100. The CIFAR-100 dataset (Krizhevsky, Hinton et al. 2009) is similar in scale to CIFAR-10 but with a finer classification granularity, featuring 100 classes containing 600 images each. This dataset is split into 50,000 training and 10,000 testing images, covering a wide array of objects from "apple" to "wolf," providing a challenging variety for object recognition tasks.

ImageNet-1k. As curated by Deng et al. (Deng et al. 2009), the ImageNet-1k dataset is a large-scale collection composed of approximately 1.28 million training images, 50,000 validation images, and 100,000 test images, spread across 1000 object classes. Its extensive variety and volume make it a benchmark for evaluating advanced image classification models.

Preprocessing Details. Prior to training, we apply several data augmentation techniques to enhance the diversity of the training sets. Specifically, we employ random cropping with a crop size of 224×224 pixels for ImageNet-1k and 32×32 for CIFAR datasets, and a 50% chance of horizontal flipping across all datasets. Data normalization is conducted using the dataset-specific mean and variance values to standardize the input images to zero mean and unit variance. These preprocessing steps are critical for mitigating overfitting and enhancing the model's ability to generalize from the training data to unseen images.

Event-driven Classification Datasets

This subsection delves into datasets tailored for neuromorphic computing, emphasizing their event-driven nature. Unlike traditional static images, event-driven sensors capture changes in illumination per pixel, providing a dynamic and temporal perspective of the scene. This characteristic makes them particularly suited for tasks requiring temporal information processing, such as action recognition and dynamic scene analysis.

CIFAR10-DVS. Developed by Li et al. (Li et al. 2017), the CIFAR10-DVS dataset transforms the classic CIFAR-10 dataset into an event-based format using a Dynamic Vision Sensor (DVS). It includes 10,000 event-based images across 10 classes at a resolution of 128×128 pixels, divided into 9,000 training and 1,000 test samples. This dataset enables the exploration of neuromorphic approaches in recognizing static image categories within a dynamic, event-driven paradigm.

N-Caltech101. The N-Caltech101 dataset, introduced by Orchard et al. (Orchard et al. 2015), consists of 8,831 event-based representations of the original Caltech101 dataset, captured at a resolution of 180×240 pixels across 101 classes. By converting static images into an event-based format, this dataset challenges models to maintain high classification performance in a more complex, temporally rich input space.

N-Cars. Sironi et al. (Sironi et al. 2018) created the N-Cars dataset, which comprises 24,029 event-based images at a resolution of 100×120 pixels, categorized into car and background classes. The dataset is split into 15,422 training samples (7,940 cars and 7,482 background samples) and 8,607 test samples (4,396 cars and 4,211 background samples). This dataset is particularly useful for binary classification tasks within the neuromorphic computing domain, focusing on vehicle detection in dynamic environments.

Action Recognition. The Action Recognition dataset by Miao et al. (Miao et al. 2019) features 10 action classes at a 346×260 resolution. It includes diverse actions such as arm-crossing, getting up, and waving, with 30 recordings per class. To accommodate the dataset's limited size, the Surface of Active Events (SAE) encoding technique is employed to convert raw data into 4,670 frame images. This dataset pushes the boundaries of neuromorphic vision by enabling the study of complex human activities through event-based sensing.

Preprocessing Details. Given the intrinsic characteristics and challenges posed by neuromorphic datasets, such as their limited size and potential for overfitting, specialized data augmentation techniques are crucial. Following the recommendations of Li et al. (Li et al. 2022), we apply augmentation methods specifically designed for neuromorphic data. These techniques, including temporal jittering and spatial transformations, enhance the diversity and robustness of the training samples. By implementing these targeted preprocessing steps, we aim to bolster the model's generalization capabilities, ensuring it does not overfit to the nuances of the training data but rather learns to perform accurately across varied neuromorphic inputs.

Object Detection and Semantic Segmentation Datasets

In the realm of computer vision, object detection and semantic segmentation are pivotal tasks that drive numerous applications, from autonomous driving to interactive augmented reality. To benchmark and refine our models for these tasks, we employ two of the most influential datasets: PASCAL VOC 2012 and MS COCO 2017. These datasets are not only vast in their image variety and annotations but also present unique challenges that push the limits of current computer vision technologies.

PASCAL VOC 2012. The PASCAL Visual Object Classes (VOC) 2012 dataset (Everingham et al. 2010) serves as a fundamental benchmark for object detection and semantic segmentation. It encompasses over 11,000 images spanning 20 object categories, ranging from people and animals to vehicles and household items. Each image is meticulously annotated with object instance boundaries, facilitating detailed object detection and precise semantic segmentation tasks. The dataset's diversity and complexity make it an essential tool for evaluating the performance and robustness of computer vision models. PASCAL VOC challenges models with real-world scenarios, including varied scales, poses, lighting conditions, and occlusions, offering insights into their generalization capabilities and limitations.

MS COCO 2017. The Microsoft Common Objects in Context (COCO) 2017 dataset (Lin et al. 2014) significantly extends the scope and depth of dataset challenges for object detection and semantic segmentation. With over 200,000 labeled images and 1.5 million object instances across 80 object categories, MS COCO provides a comprehensive landscape for training and evaluating advanced models. The dataset is renowned for its rich annotations, including object segmentation masks, multiple objects per image, and a wide range of object sizes. Its emphasis on object detection in the context of scene understanding (i.e., identifying objects in their natural environment with contextual relationships) and the inclusion of complex, crowded scenes makes it a stringent testbed for assessing model efficacy in more intricate visual environments.

Dataset Utilization. For our studies, both PASCAL VOC 2012 and MS COCO 2017 are integral in developing and testing our object detection and semantic segmentation frameworks. The datasets' varied and comprehensive annotations allow us to train models that are not only precise in identifying and segmenting objects but also robust against a wide array of real-world visual challenges. To leverage the full potential of these datasets, we adopt a suite of data augmentation techniques, such as scale augmentation, rotation, and flipping, to enhance the diversity of the training data. Furthermore, we utilize mean Average Precision (mAP) to ensure a thorough assessment of model performance across both datasets.

3D Task Datasets

Our study evaluates the proposed method for 3D point cloud classification and 3D part segmentation, pivotal tasks in understanding complex 3D environments. We leverage the ShapeNet dataset, as detailed by Yi et al. (Yi et al. 2016), using the PointNet architecture (Qi et al. 2017) as the foundation for our experiments.

3D Point Cloud Classification involves identifying the category of a whole object based on its 3D point cloud representation. PointNet, a groundbreaking neural network, processes point clouds directly, enabling effective feature extraction and classification without the need for volumetric representations. This approach is critical for handling the intricacies of 3D geometric data efficiently.

Part Segmentation represents a more granular task than classification, aiming to categorize individual components of an object within a 3D scan or mesh model. As shown in 4, for example, in a 3D representation of a chair, the task would involve accurately identifying and segmenting the chair's legs, backrest, and other constituent parts. This task is inherently challenging due to the fine-grained nature of the segmentation and the complexity of 3D shapes.

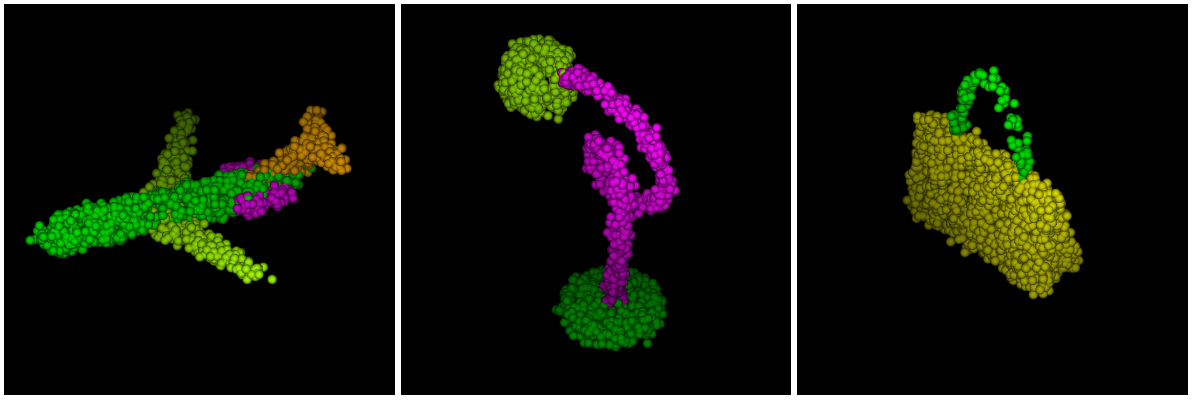
ShapeNet Part Dataset. We utilize the ShapeNet part dataset, which encompasses 16,881 3D shapes across 16 categories, annotated with 50 distinct parts. This dataset is particularly suited for evaluating part segmentation methodologies due to its diverse range of objects and detailed part annotations. Objects in the dataset are typically labeled with two to five parts, and ground truth annotations are provided for points sampled on the objects' surfaces.

Evaluation Metric. The primary metric for evaluating part segmentation performance is the mean Intersection over Union (mIoU) calculated per point. For each object S within category C, mIoU is computed by assessing the IoU for each part type within C, between the ground truth and predicted segmentations. Part IoUs are treated as 1 in cases where both the ground truth and prediction are empty, ensuring fairness in evaluation. The mIoU for a shape is obtained by averaging the IoUs across all part types in its category. Subsequently, to derive the category's mIoU, we average the mIoUs for all shapes within that category.

More Reuslt

Performance on Event-driven Classification

As shown in Tab. 3, we provide more results of our approach over other leading SNN techniques across various neuromorphic datasets within limited timesteps. Our results show that our approach, enhanced with the AdaFire technique, consistently outperforms other leading SNN models. For instance, in the domain of Action Recognition which encapsulates sequential human actions recorded with event-based cameras, our model achieves an impressive top-1 accuracy of **88.21%**. These results, markedly better than alternatives, underscore our method's adaptability to diverse neuromorphic datasets.



(a) Airplane (b) Lamp (c) Bag

Figure 4: **Visualization of Part Segmentation results on the ShapeNet Part dataset.**

Table 3: Performance comparison between the proposed model and the state-of-the-art models on different neuromorphic datasets.

Dataset	Model	Timesteps	Accuracy (%)
N-Cars	CarSNN (Viale et al. 2021)[IJCNN]	10	86.00
	NDA (Li et al. 2022)[ECCV]	10	91.90
	AdaFire (Ours)	**8**	**96.24**
Action Recognition	STCA (Gu et al. 2019)[IJCAI]	10	71.20
	Mb-SNN (Liu et al. 2021)[IJCAI]	10	78.10
	AdaFire (Ours)	**8**	**88.21**

Table 4: Performance comparison for Semantic Segmentation.

Dataset	Method	Arch.	ANN	T	mAP
VOC	Calibration (Li et al. 2021b)[ICML]	ResNet50	73.36	128	69.11
	AdaFire (Ours)	ResNet50	73.36	16	72.17
COCO	Calibration (Li et al. 2021b)[ICML]	ResNet50	47.34	128	38.23
	AdaFire (Ours)	ResNet50	47.34	16	45.15

Table 5: Performance comparison for 3D Classification on the ShapeNet dataset.

Method	Arch.	T	Acc.
ANN	PointNet	/	97.73
Calibration (Li et al. 2021b)[ICML]	PointNet	64	95.89
AdaFire (Ours)	PointNet	**16**	**97.52**

Table 6: Performance comparison for 3D Part Segmentation on the ShapeNet dataset. Our method uses T = 16 and baseline uses T= 64.

Method	Mean	Aero	Bag	Cap	Car	Chair	Guitar	Knife	Earphone
ANN	77.46	81.55	78.74	71.87	75.15	89.1	89.22	83.81	69.55
Calibration (Li et al. 2021b)	72.25	74.11	78.07	70.15	62.53	79.48	83.52	77.88	66.97
AdaFire (Ours)	**75.65**	**78.99**	**78.91**	**70.89**	**71.95**	**88.17**	**86.03**	**82.85**	**67.27**
Method	**Mean**	**Lamp**	**Laptop**	**Motor**	**Mug**	**Pistol**	**Rocket**	**Table**	**Skateboard**
ANN	77.46	80.74	94.54	60.95	85.58	80.68	44.94	81.45	71.47
Calibration (Li et al. 2021b)	72.25	76.53	92.45	56.27	78.91	76.58	42.54	76.91	63.1
AdaFire (Ours)	**75.65**	**79.13**	**92.47**	**61.64**	**85.23**	**80.11**	**43.16**	**78.32**	**65.3**

Performance on Semantic Segmentation.

We conducted comprehensive evaluations using two widely recognized datasets: PASCAL VOC 2012 and MS COCO 2017. Table 4 presents a detailed comparison of our model's performance against the Calibration baseline. On the PASCAL VOC dataset, AdaFire achieves a mean Average Precision (mAP) of 72.17%, surpassing the Calibration baseline (69.11%) while reducing the number of timesteps from 128 to just 16. This improvement is even more pronounced on the MS COCO dataset, where AdaFire reaches an mAP of 45.15% compared to the baseline's 38.23%, again with significantly fewer timesteps.

Notably, our model maintains the same architecture (ResNet50) and ANN performance as the baseline, highlighting that the improvements are due to our AdaFire approach rather than changes in the underlying network structure. The substantial reduction in timesteps (from 128 to 16) coupled with improved accuracy demonstrates AdaFire's ability to enhance both efficiency and performance in semantic segmentation tasks.

These results affirm the effectiveness of AdaFire in pushing the boundaries of SNN capabilities in semantic segmentation tasks, offering a promising direction for future research and applications where both accuracy and computational efficiency are crucial.

Performance on 3D Classification.

We extend the application of SNNs to 3D point cloud classification, addressing the growing need for efficient 3D processing in fields such as remote sensing, augmented/virtual reality (AR/VR), robotics, and autonomous driving. Our evaluation utilizes the ShapeNet dataset (Yi et al. 2016), employing PointNet (Qi et al. 2017) as the backbone architecture.

Table 5 provides a comparative analysis of our method against existing benchmarks. AdaFire achieves an accuracy of 97.52% with only 16 timesteps, significantly outperforming the Calibration baseline (95.89% with 64 timesteps) and closely approaching the ANN performance (97.73%). This result represents a 1.63% improvement over the Calibration baseline while using just a quarter of the timesteps.

The performance of AdaFire in this 3D classification task is particularly noteworthy as it demonstrates the potential of SNNs to handle complex spatial data efficiently. By achieving near-ANN levels of accuracy with drastically reduced timesteps, our method showcases the viability of SNNs for real-world 3D classification applications where both accuracy and computational efficiency are critical.

Performance on 3D Part Segmentation.

We further extend the application of SNNs to the challenging domain of 3D Part Segmentation, a task that requires assigning specific part category labels to individual points within a 3D model. This represents a pioneering effort in applying SNNs to this nuanced 3D recognition task.

Table 6 presents a comprehensive comparison of our method against the ANN and Calibration baselines across various object categories. AdaFire achieves a mean accuracy of 75.65% across all categories, significantly outperforming the Calibration baseline (72.25%) and approaching the ANN performance (77.46%). In addition, AdaFire shows substantial improvements in challenging categories such as Car (71.95% vs. 62.53% for Calibration), Chair (88.17% vs. 79.48%), and Motor (61.64% vs. 56.27%). Moreover, these results are achieved with only 16 timesteps, compared to the 64 timesteps used by the Calibration baseline, highlighting the efficiency of our approach.

The success of AdaFire in 3D part segmentation, a task requiring a fine-grained understanding of object structure, underscores the potential of SNNs to handle intricate 3D recognition problems effectively. This advancement opens up new possibilities for applying SNNs in fields such as robotics and computer vision, where a detailed understanding of object parts is crucial.

Visualization

In this section, we present visualizations to illustrate the effectiveness of the Adaptive-Firing (AdaFire) approach in the context of Object Detection and Semantic Segmentation tasks. Our objective is to showcase how AdaFire stands in comparison to traditional ANNs and the Calibration method (Li et al. 2021b), under conditions designed to maintain equivalent energy consumption levels for a fair comparison. Specifically, we configure our method with $\varphi = 4$ and $T = 16$, while for the Calibration method, T is adjusted to 64, ensuring parity in energy requirements across both methodologies.

Object Detection Results

The visual results from the object detection task unequivocally demonstrate the superior performance of our AdaFire approach. It exhibits a remarkable ability to identify a greater number of objects with heightened confidence levels, significantly bridging the performance chasm that typically separates SNNs from their ANN counterparts. This enhanced detection capability is pivotal for applications requiring high precision and reliability in object identification under energy-constrained environments.

Semantic Segmentation Results

In the realm of semantic segmentation, the AdaFire approach reveals its prowess in capturing intricate edge details, facilitating a more comprehensive and accurate delineation of segmented areas. This capability is especially beneficial for complex scene parsing where the precise demarcation of object boundaries is crucial. The improved edge characterization afforded by AdaFire translates into segmentation outputs that are not only closer to the ground truth but also surpass the granularity achievable by conventional SNN methodologies.

Discussion

Through these visual comparisons, the AdaFire method distinctly outperforms the Calibration strategy, aligning more closely with the results obtainable through ANNs while operating within similar energy constraints. This advancement

underscores the potential of AdaFire in narrowing the performance gap between SNNs and ANNs, thereby expanding the feasibility of deploying energy-efficient SNNs in scenarios traditionally dominated by ANNs due to their superior accuracy and detail resolution.

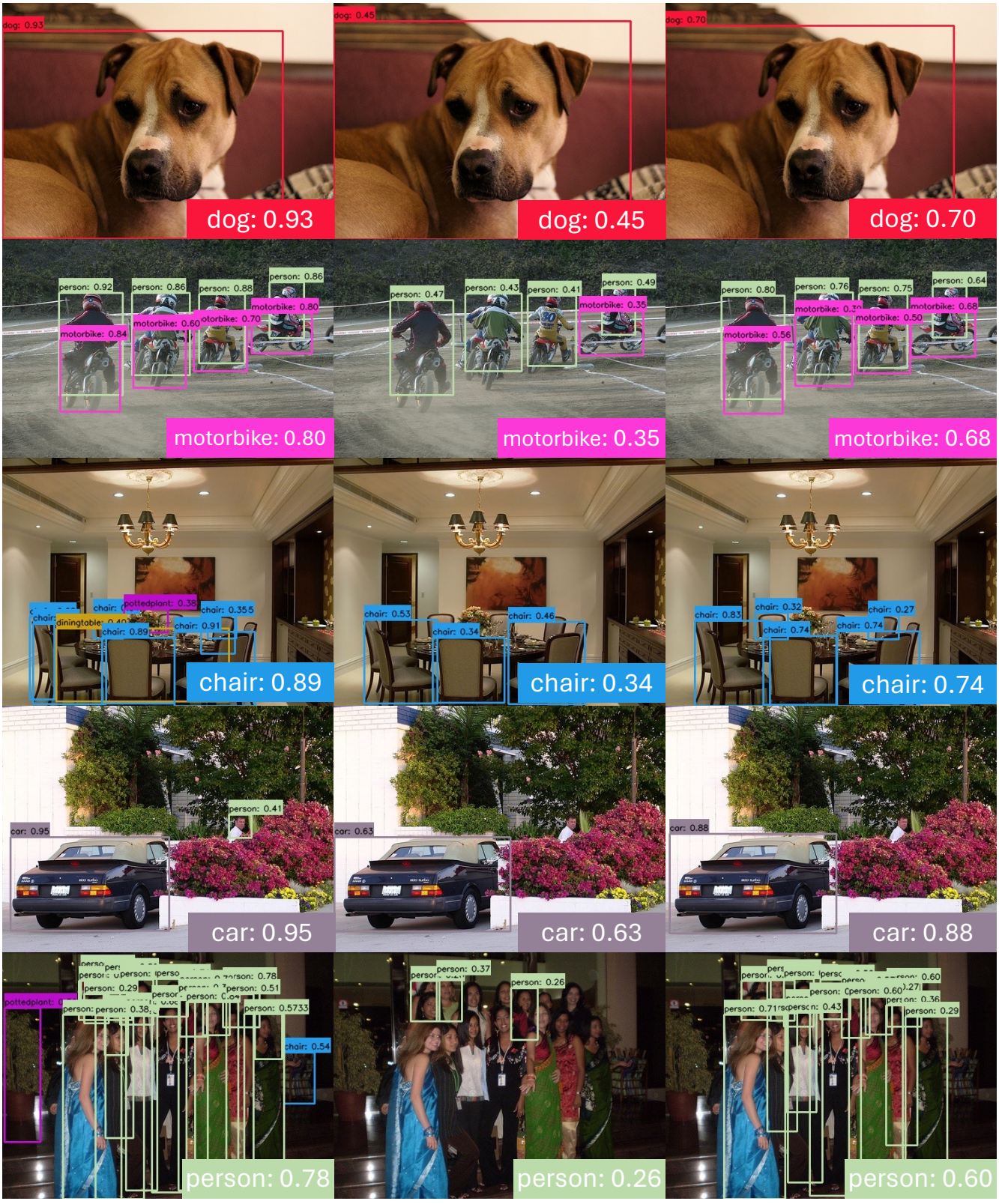


Figure 5: **Visualization of Object Detection on the VOC dataset.**

Figure 6: **Visualization of Object Detection on the COCO dataset.**

Figure 7: **Visualization of Semantic Segmentation on the VOC dataset.**

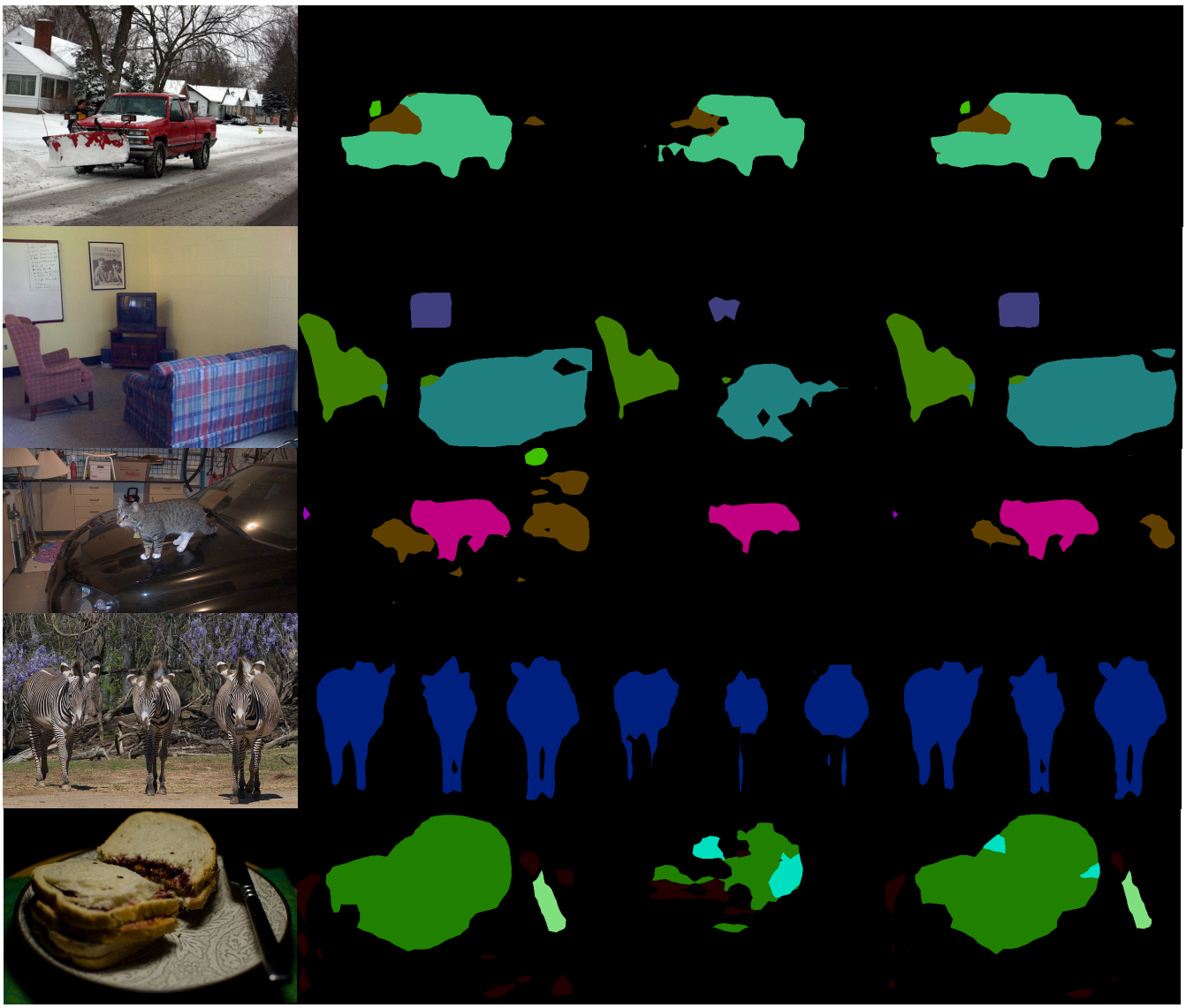


Figure 8: **Visualization of Semantic Segmentation on the COCO dataset.**

References

Bu, T.; Fang, W.; Ding, J.; Dai, P.; Yu, Z.; and Huang, T. 2021a. Optimal ANN-SNN Conversion for High-accuracy and Ultra-low-latency Spiking Neural Networks. In *International Conference on Learning Representations*.

Bu, T.; Fang, W.; Ding, J.; Dai, P.; Yu, Z.; and Huang, T. 2021b. Optimal ANN-SNN Conversion for High-accuracy and Ultra-low-latency Spiking Neural Networks. In *International Conference on Learning Representations*.

Cao, J.; Wang, Z.; Guo, H.; Cheng, H.; Zhang, Q.; and Xu, R. 2024. Spiking denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 4912–4921.

Cao, Y.; Chen, Y.; and Khosla, D. 2015. Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision*, 113: 54–66.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A Large-Scale Hierarchical Image Database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 248–255. Ieee.

Ding, J.; Yu, Z.; Tian, Y.; and Huang, T. 2021. Optimal Ann-Snn Conversion for Fast and Accurate Inference in Deep Spiking Neural Networks. *arXiv preprint arXiv:2105.11654*.

Everingham, M.; Van Gool, L.; Williams, C. K.; Winn, J.; and Zisserman, A. 2010. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88: 303–338.

Gu, P.; Xiao, R.; Pan, G.; and Tang, H. 2019. STCA: Spatio-Temporal Credit Assignment with Delayed Feedback in Deep Spiking Neural Networks. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, 1366–1372. Macao, China: International Joint Conferences on Artificial Intelligence Organization. ISBN 978-0-9992411-4-1.

Hao, Z.; Bu, T.; Ding, J.; Huang, T.; and Yu, Z. 2023a. Reducing ann-snn conversion error through residual membrane potential. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 11–21.

Hao, Z.; Ding, J.; Bu, T.; Huang, T.; and Yu, Z. 2023b. Bridging the Gap between ANNs and SNNs by Calibrating Offset Spikes. In *The Eleventh International Conference on Learning Representations*.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11): 2278–2324.

Li, H.; Liu, H.; Ji, X.; Li, G.; and Shi, L. 2017. CIFAR10-DVS: An Event-Stream Dataset for Object Classification. *Frontiers in Neuroscience*, 11.

Li, Y.; Deng, S.; Dong, X.; Gong, R.; and Gu, S. 2021a. A Free Lunch from ANN: Towards Efficient, Accurate Spiking Neural Networks Calibration. In *International Conference on Machine Learning*, 6316–6325. PMLR.

Li, Y.; Deng, S.; Dong, X.; Gong, R.; and Gu, S. 2021b. A Free Lunch from ANN: Towards Efficient, Accurate Spiking Neural Networks Calibration. In *International Conference on Machine Learning*, 6316–6325. PMLR.

Li, Y.; Kim, Y.; Park, H.; Geller, T.; and Panda, P. 2022. Neuromorphic Data Augmentation for Training Spiking Neural Networks. *arXiv preprint arXiv:2203.06145*.

Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; and Zitnick, C. L. 2014. Microsoft Coco: Common Objects in Context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, 740–755. Springer.

Liu, Q.; Xing, D.; Tang, H.; Ma, D.; and Pan, G. 2021. Event-Based Action Recognition Using Motion Information and Spiking Neural Networks. In *IJCAI*, 1743–1749.

Miao, S.; Chen, G.; Ning, X.; Zi, Y.; Ren, K.; Bing, Z.; and Knoll, A. 2019. Neuromorphic Vision Datasets for Pedestrian Detection, Action Recognition, and Fall Detection. *Frontiers in neurorobotics*, 13: 38.

Orchard, G.; Jayawant, A.; Cohen, G. K.; and Thakor, N. 2015. Converting Static Image Datasets to Spiking Neuromorphic Datasets Using Saccades. *Frontiers in Neuroscience*, 9.

Qi, C. R.; Su, H.; Mo, K.; and Guibas, L. J. 2017. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 652–660.

Sironi, A.; Brambilla, M.; Bourdis, N.; Lagorce, X.; and Benosman, R. 2018. HATS: Histograms of Averaged Time Surfaces for Robust Event-Based Object Classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1731–1740.

Viale, A.; Marchisio, A.; Martina, M.; Masera, G.; and Shafique, M. 2021. Carsnn: An Efficient Spiking Neural Network for Event-Based Autonomous Cars on the Loihi Neuromorphic Research Processor. In *2021 International Joint Conference on Neural Networks (IJCNN)*, 1–10. IEEE.

Wang, Z.; Fang, Y.; Cao, J.; Zhang, Q.; Wang, Z.; and Xu, R. 2023. Masked spiking transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 1761–1771.

Yao, M.; Zhao, G.; Zhang, H.; Hu, Y.; Deng, L.; Tian, Y.; Xu, B.; and Li, G. 2023. Attention Spiking Neural Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Yi, L.; Kim, V. G.; Ceylan, D.; Shen, I.-C.; Yan, M.; Su, H.; Lu, C.; Huang, Q.; Sheffer, A.; and Guibas, L. 2016. A scalable active framework for region annotation in 3d shape collections. *ACM Transactions on Graphics (ToG)*, 35(6): 1–12.