Unleashing the Potential of Spiking Neural Networks with Dynamic Confidence

Chen Li*
The University of Manchester
Manchester, United Kingdom
chen.li@manchester.ac.uk

Edward G Jones*
The University of Manchester
Manchester, United Kingdom
edward.jones-3@manchester.ac.uk

Steve Furber
The University of Manchester
Manchester, United Kingdom
steve.furber@manchester.ac.uk

Abstract

This paper presents a new methodology to alleviate the fundamental trade-off between accuracy and latency in spiking neural networks (SNNs). The approach involves decoding confidence information over time from the SNN outputs and using it to develop a decision-making agent that can dynamically determine when to terminate each inference.

The proposed method, Dynamic Confidence, provides several significant benefits to SNNs. 1. It can effectively optimize latency dynamically at runtime, setting it apart from many existing low-latency SNN algorithms. Our experiments on CIFAR-10 and ImageNet datasets have demonstrated an average 40% speedup across eight different settings after applying Dynamic Confidence. 2. The decision-making agent in Dynamic Confidence is straightforward to construct and highly robust in parameter space, making it extremely easy to implement. 3. The proposed method enables visualizing the potential of any given SNN, which sets a target for current SNNs to approach. For instance, if an SNN can terminate at the most appropriate time point for each input sample, a ResNet-50 SNN can achieve an accuracy as high as 82.47% on ImageNet within just 4.71 time steps on average. Unlocking the potential of SNNs needs a highly-reliable decision-making agent to be constructed and fed with a high-quality estimation of ground truth. In this regard, Dynamic Confidence represents a meaningful step toward realizing the potential of SNNs. Code is available[1].
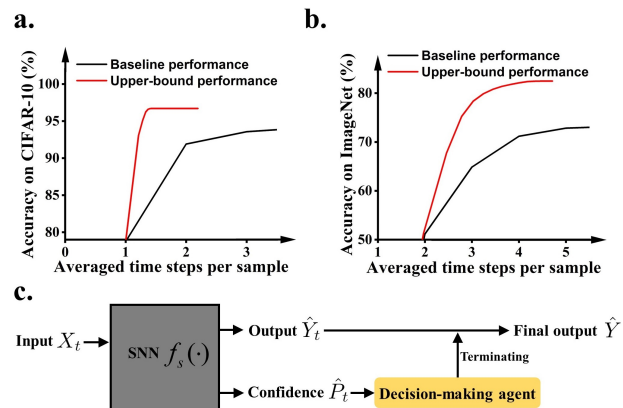
Figure 1. The upper-bound performance of SNNs when fully utilizing dynamic strategies at runtime, as shown by the red curves in **a.** ResNet18 on CIFAR-10 and **b.** ResNet-50 on ImageNet. The black curves represent baseline SNN performance without dynamic strategies. Additional figures with other settings can be found in the Supplementary Material. **c.** The diagram of the proposed Dynamic Confidence, which can be implemented on-the-fly.

1. Introduction

Deep artificial neural networks (ANNs) have achieved remarkable success in computer vision tasks [24, 37, 30]. These improvements have been accompanied by ever-growing model complexity and neural network depth. Though the classification accuracy has improved dramatically, the latency and energy cost have also increased, which poses a challenge for real-world AI applications on edge devices, such as mobile phones, smartwatches, and IoT hardware.

Deep spiking neural networks (SNNs) are promising to offer power and latency advantages over counterpart deep artificial neural networks during inference. However, when

---

*These authors contributed equally to this work.

[1] https://github.com/chenlicodebank/
Dynamic-Confidence-in-Spiking-Neural-Networks

their power (e.g. averaged spike counts per inference) or latency (e.g. averaged inference time steps per inference) are strictly constrained, SNNs suffer huge accuracy degradation. In recent years, much scholarly attention has been paid to the conflict between accuracy and latency in SNNs, growing the field of low-latency SNNs (also known as fast SNNs). The primary goal of fast SNN research is achieving equivalent or slightly-lower accuracy than the baseline ANN accuracy with as few inference time steps as possible. With the reduction in the number of inference time steps, the spike counts required for each inference can potentially be reduced as well, if the firing rates can be kept the same when latency decreases. In hardware that can take scale power with spike processing operations, this research on low-latency SNNs can contribute to reducing the power requirements of SNNs.

In this paper, we detail a runtime optimization method called Dynamic Confidence and show that it can effectively reduce the inference latency in low-latency SNNs on CIFAR-10 and ImageNet, without the need to sacrifice accuracy or to increase firing rates. Our paper has several new contributions:

- We propose a method (Dynamic Confidence) to decode a temporal series of confidence values from the inference results of an SNN, and utilize this confidence information to dynamically guide inference. It is the first study to formulate confidence in SNNs and introduce the formal concept of dynamic strategies to SNNs.

- Dynamic Confidence can further reduce the inference latency by 40% on average on the state-of-the-art low-latency SNNs (QFFS[26] and QCFS[3]) on CIFAR-10 and ImageNet. By reducing the inference latency and maintaining the same firing rates, the power consumption of SNN inference is reduced consequently.

- We provide a method to calculate the upper-bound performance that an SNN can achieve when applying dynamic strategies at runtime. Our findings shed light on the vast untapped potential within current SNNs, providing valuable insights to answer the fundamental question of why to use SNNs instead of ANNs.

- Dynamic Confidence is a lightweight, parameter-insensitive, on-the-fly method. These features make Dynamic Confidence extremely easy to be applied on a variety of low-latency SNN algorithms whenever further latency and power reductions are desirable.

2. Motivation

- A main research method adopted in current SNN studies is improving SNNs by leveraging knowledge from ANNs. Some successful examples of this include, but are not limited to, straight-through estimators [32], quantization [26], Backpropagation through time (BPTT) [2], network architectures [28], neural architecture search [22], and neural network simulation tools [39]. The primary concern during this process is how to make ANN knowledge applicable to SNNs, which usually requires additional optimization of spike dynamics in SNNs to ensure smooth knowledge transfer. The study presented in this paper also follows this research method. Specifically, this study is intended to explore whether confidence, a key concept in Bayesian neural networks and interpretable AI can be applied to SNNs to help reduce inference latency as well as inference power. We discovered that the temporal dimension of SNNs provides richer information in confidence compared to ANNs. This paper exemplifies a simple way to leverage the information in SNN confidence to further boost the performance of SNNs.

- Dynamic strategies have received considerable critical attention in ANNs due to their effectiveness in bringing considerable speedup and computational savings without degrading prediction accuracy [40]. In contrast, there is not any research studying the application of dynamic strategies to SNNs and the evaluation of possible speed and power gains on nontrivial datasets such as CIFAR-10 and ImageNet. This paper seeks to address an existing gap by sharing the preliminary findings of implementing a dynamic strategy in SNNs. Different from other dynamic strategy research in ANNs, we found in this paper that spikes make it convenient to apply a dynamic strategy explicitly in the dimension of precision. This task has been notably challenging in existing ANN dynamic strategy literature.

- Information is in the central position in SNN research. For example, a primary concern in directly-trained SNNs is whether better spatial-temporal information representation and computing can be achieved by surrogate gradients. This paper also emphasizes the significance of the information, but from a different perspective. Instead of seeking more spatial-temporal information by replacing ANN-to-SNN conversion with spatial-temporal training by surrogate gradients, we stick to using ANN-to-SNN conversion but try to seek if any information in the SNN output is missed by other research and has not been fully exploited. Our results show that confidence in SNN outputs contains valuable information and can be leveraged to improve SNN performance on latency and power.

3. Related Work

Spiking Neural Networks (SNNs). Spiking neural networks (SNNs) are biologically-inspired models that have time-evolving states, unlike artificial neural networks (ANNs) [31, 25, 9, 11]. SNNs use spikes, binary pulses that model action potentials in biology, to transmit information between neurons. Because data processing in SNNs need only take place when spikes are received, SNNs can be implemented in an event-based manner and can benefit from improved power efficiency over ANNs, especially when sparsity can be exploited in hardware [21, 5, 20].

The potential for increased efficiency and biological plausibility that SNNs promise has led to ANN-to-SNN conversion being an active area of research. Converting ANN activations to spike rates in SNNs has proven possible [8] with marginal loss in accuracy when data-based normalisation techniques are applied [38]. The significant downsides to these kinds of rate-coding approaches are that they require long integration times, which can impact the potential latency benefits of SNNs.

Low-latency SNNs (fast SNNs). SNN latency optimizations have received increasing scholarly attention in recent years [36, 7, 3, 26, 10], and they can be coarsely divided to two categories: methods based on ANN-to-SNN conversion, and methods based on surrogate gradients. This research falls into the first category. Deng and Gu combine threshold balancing and soft-reset to correct conversion errors and show a significant reduction in the number of SNN simulation time steps needed [7]. The QCFS method [3] uses a quantized activation function to achieve ultra-low latency in SNNs. The QFFS method [26] proposes to reduce accuracy loss in low-latency SNNs by compressing information and suppressing noise.

Dynamic Strategies. Dynamic strategies [40, 19, 4, 13, 1, 18, 29], also known as dynamic networks, are network optimization algorithms that are input-dependent and can dynamically decide which part of the network to execute at runtime on a per-input basis. The basic assumption in a dynamic strategy is that in real-world applications, examples are not equal for a given network model. Novel examples require the full model capacity to generate a reliable inference result, while simple examples can be solved confidently with fewer computing resources. Dynamic strategies optimize the inference of an ANN by taking both the model and input examples into considerations, and are promising for achieving fast and energy-efficient edge computing.

Compared to these studies on dynamic strategies, our method does not require any modifications to the internal structure of standard models or the addition of heavy auxiliary sub-networks as a dynamic controller to generate instructions. Instead, the proposed Dynamic Confidence can simply be calculated based on the neural network output during inference, bringing lower overhead and higher porta-

bility than other dynamic strategies. Also, most of the existing dynamic strategy research optimizes the number of channels and layers, while our method implicitly optimizes the activation precision, thanks to the event-based nature of SNNs. A visualization illustrating the dimensions of dynamic strategies can be found in the Supplementary Material.

4. SNN preliminaries

The detailed equations that describing spiking neuronal dynamics and ANN-to-SNN conversion are provided in the Supplementary Material. The following sections focus on introducing the main ideas of the proposed method in a comprehensive manner.

5. Approach

The goal of our method is to allocate computing resources (specifically, inference latency) dynamically for each inference by using a confidence metric determined by model output. By this way, the time steps required for each inference are heterogeneous, and the averaged time steps per inference can potentially be reduced compared to using homogeneous time steps for all inferences. The dynamic resource allocation mentioned above can be modeled as a decision-making agent with two main points to consider:

1. The decisions made regarding the allocation of computing resources to the model needs to facilitate fast and efficient inference while maintaining a model accuracy that is as high as possible. In other words, there is a minimal trade-off of accuracy for decreased latency.

2. The decision-making mechanism itself needs to be lightweight and intelligible so as not to add unnecessary overhead and complexity to the model and to enable it to allow users to balance accuracy against latency.

In our approach, we calculate the confidence associated with network output at runtime and use this confidence information to decide when to end inference in advance to get lower average latency per example without accuracy degradation. A confidence threshold is adopted to balance accuracy and latency. Section 5.1 describes the preliminary confidence concept, the quality of confidence, and confidence calibration. Section 5.2 explains the difference between confidence in SNNs and ANNs, highlights the challenges that must be addressed before utilizing confidence in SNNs, and presents our proposed solution. Section 5.3 presents how to construct a decision-making agent and exemplifies to use a Pareto Front to compute its threshold, which can be done with minimal cost.
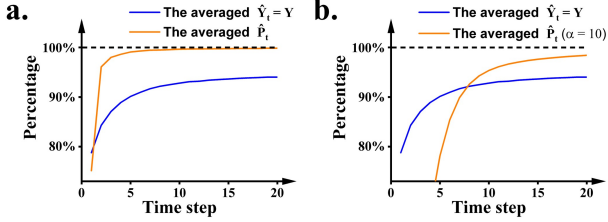
Figure 2. The trend of the averaged $\hat{Y}_t = Y$ and the averaged $\hat{P}_t$ over time t on the test set of CIFAR-10. **a.** $\hat{P}_t$ saturates to 100% quickly with time. **b.** After introducing a scalar regulation parameter α, $\hat{P}_t$ is softened.

5.1. Confidence

In multi-class classification problems, for a given input example X, a label Y, and an artificial neural network $f(\cdot)$, the outputs are $f(X) = (\hat{Y}, \hat{P})$. $\hat{Y}$ is the predicted output label and $\hat{P} \in [0, 1]$ is the confidence of this prediction. Confidence $\hat{P}$ is an estimate of the ground truth inference correctness likelihood $P(\hat{Y} = Y)$, and it is usually calculated by Softmax in the multi-class classification problems discussed above. In some application scenarios where the safety and trustworthiness of the neural network models are central, such as medical diagnosis and autonomous driving, the confidence of the prediction is as crucial as its accuracy. The confidence can also be used to understand the interpretability and explainability of the neural network and can interact with other probabilistic models such as Bayesian neural networks.

Recognising the significance of $\hat{P}$, there is a growing body of literature that delivers calibration algorithms to improve the reliability of $\hat{P}$ [33]. Note that calibration can only improve the quality of confidence; a perfectly calibrated model is impossible to achieve. Our approach applies temperature scaling [15], a post-training calibration algorithm, to guarantee $\hat{P}$ to be of high quality and to keep the overhead of calibration low.

5.2. Dynamic Confidence in Spiking Neural Networks

After conducting ANN-to-SNN conversion, an ANN model $f(\cdot)$ will be converted into an SNN model $f_s(\cdot)$. Unlike a non-spiking neural network that only generates a single pair $(\hat{Y}, \hat{P})$, a spiking neural network $f_s(\cdot)$ with inputs X_t will generate a series of outputs $(\hat{Y}_t, \hat{P}_t)$ during inference, where $t \in \mathcal{T} = \{1, \ldots, T\}$ represents a time step during SNN inference. One feature of SNNs is that their prediction accuracy in the test set will increase with more time steps [36]. This feature is associated with the event-based computing nature of SNNs: in each time step, only a fraction of the information is represented by spikes and processed by spiking neurons in an event-based manner; with more time steps, more information is accumulated and more

reliable decision can be made. This well-known SNN feature means that: $\hat{Y}_{t_2} = Y$ is more likely to happen than $\hat{Y}_{t_1} = Y$ when $t_2 > t_1$. Here, we go one more step further and state that: If $\hat{P}_t$ is a good estimate of $\hat{Y}_t = Y$, $\hat{P}_{t_2}$ should also be more likely to be higher than $\hat{P}_{t_1}$ when $t_2 > t_1$.

To further clarify these, we plot the trend of $\hat{Y}_t = Y$ and $\hat{P}_t$ over time t in Figure 2.a., averaged on 10,000 test samples in CIFAR-10. The blue curve (the averaged $\hat{Y}_t = Y$) actually corresponds to the traditional accuracy-vs-latency response curve in SNNs [36]. The orange curve (the averaged $\hat{P}_t$) is an estimate of the correctness likelihood $P(\hat{Y}_t = Y)$, averaged on 10,000 test samples. We can see that both these curves show a monotonic increase with time t with minor fluctuations, which is in line with the statements above. The same trend of these two curves suggests that $\hat{P}_t$ is a good estimator of $\hat{Y}_t = Y$: A low confidence value $\hat{P}_t$ indicates a low likelihood of correctness $\hat{Y}_t = Y$, and vice versa. Note that during SNNs inference, $\hat{Y}_t = Y$ is unknown because it needs to access the ground truth Y. However, this ground truth information can be implicitly fetched by estimating it by the confidence information $\hat{P}_t$. This provides a huge opportunity on optimizing SNNs by utilizing confidence.

One obstacle must be overcome before optimizing SNNs with confidence. As shown in Figure 2.a., the orange curve saturates to 100% quickly with time t. In other words, an overconfident averaged $\hat{P}_t$ is produced after only a few time steps, after which there is little room for further improvement. We call this problem confidence saturation. Confidence saturation has a detrimental effect on our proposed method, as we want the $\hat{P}_t$ to act as a high-quality estimate of $\hat{Y}_t = Y$. A saturated $\hat{P}_t$ can no longer provide a good estimate of $\hat{Y}_t = Y$ since the saturated confidence is too trivial to distinguish from each other. The saturated confidence will also make the decision-making agent we are constructing later more sensitive to its parameter, which will be discussed in the following sections.

The confidence saturation problem is caused by the format of the SNN outputs. The outputs of a rate-coded SNN are accumulated spikes [26] or, in some cases that need higher output precision, the membrane potential of integrate-but-not-fire neurons [27, 3]. In both situations, the output logits of SNNs $f_s(X_t)$ increase with t continuously, which causes the saturated confidence $\hat{P}_t$ after applying Softmax to the SNN outputs. To prevent saturated confidence we introduce a scalar regulation parameter α to restrict the scale of the output logits in the SNNs, and calculate confidence according to

$$\hat{P}_t = \max\left(\frac{e^{\mathbf{z}_t^i/\alpha}}{\sum_{j=1}^{K} e^{\mathbf{z}_t^j/\alpha}}\right) \quad for \ i = 1, 2, \ldots, K. \quad (1)$$

$\mathbf{z}_t$ is the output logits of the SNN, that is

$$\mathbf{z}_t = f_s(X_t) \qquad (2)$$

α serves to regulate the SNN's confidence level, ensuring it does not become overly saturated $\hat{P}_t \asymp 1$ before the end of the SNN simulation at time T. While various alpha values can fulfill this purpose, our selected alpha is intended to align the SNN's confidence value at T with that of the ANN. This alignment guarantees a one-to-one match between the ANN and SNN models, allowing tools designed for ANN confidence to be applied to SNNs. With this consideration, α is set as $2^b - 1$ when SNNs are built by QFFS [26], where b is the bit precision in all hidden layers. However, when SNNs are built by QCFS [3], deriving an exact alpha value becomes challenging because of the lack of existing research that can accurately estimate the effect of occasional noise [26] (also called unevenness error [3]) on SNN output without accessing ground truth. The empirical solution we propose sets alpha to the same value as the number of simulation time steps T when using QCFS. We plot the averaged $\hat{P}_t$ after applying scaling factor α in Figure 2.b. It can be seen from this figure that after scaling the average $\hat{P}_t$ is softened.

5.3. Optimizing Latency by Pareto Front

5.3.1 Optimization Targets

Before conducting any optimization, we first clarify the targets of SNN latency optimization research. Certainly, the primary target of this type of research is achieving low latency in an SNN. On the other hand, we do not wish this low-latency SNN to have a low accuracy or to even be nonfunctional, which means that high accuracy should be considered as one of the optimization objectives as well. Hence, the goals of SNN latency optimization research should be at least twofold: achieving low latency and maintaining high accuracy. This is a typical multi-objective optimization problem, which allows a Pareto Front to be used to find efficient solutions.

5.3.2 Proposed Method and Upper Bound

In our proposed method, we dynamically optimize SNN latency based on the estimated correctness likelihood $\hat{P}_t$ for each example, while trying to ensure that $\hat{Y}_t = Y$. To accomplish this goal, we develop a simple decision-making agent that determines the optimal time step for terminating inference on a given sample. At runtime, a decision-making agent is placed at the output layer and it compares whether the estimated correctness likelihood $\hat{P}_t$ is higher than a confidence threshold th_c. If it is, a decision to terminate SNN inference will be made. For example, if $th_c = 0.6$ and $\hat{P}_t = [0.1, 0.3, 0.5, 0.7, 0.9]$ at the first 5 time steps for a
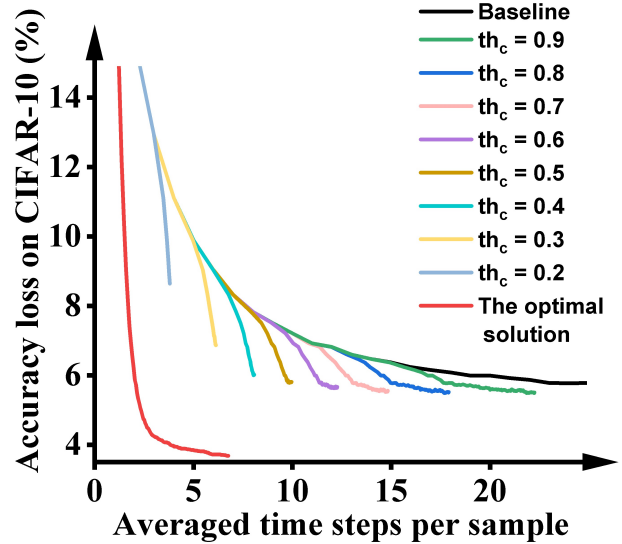

Figure 3. The Pareto Front of ResNet-18 on CIFAR-10 as well as the baseline performance (without a dynamic strategy) and the optimal solution (Assuming the ground truth is accessible). Note that unlike the general form of Pareto Front, SNNs have a temporal dimension so their outputs are curves instead of points. The different settings of th_c have different accuracy-vs-latency response curves.

given example, the SNN will be terminated in advance at the fourth time step as $\hat{P}_4 = 0.7$, higher than $th_c = 0.6$. Different examples will have different termination times.

The optimal performance that can be achieved by our proposed dynamic strategy can be calculated in the following way: Firstly, assuming the ground truth label Y is accessible during inference. In Dynamic Confidence, this corresponds to the assumption that our formulated confidence $\hat{P}_t$ is highly reliable and is a perfect estimate of $\hat{Y}_t = Y$. Secondly, the SNN inference will always be terminated for a given example at t as soon as $\hat{Y}_t = Y$. This scenario represents the perfect capture of the correct termination time by the constructed decision-making agent. This upper-bound performance fully exploits dynamic strategies at runtime and represents the best possible performance that a given SNN can achieve.

We compared this upper-bound performance to the baseline performance on CIFAR-10 and ImageNet as displayed in Figure 1. On CIFAR-10, the optimal solution achieved 95.1% accuracy in 1.28 time steps and 96.7% accuracy in 1.43 time steps, which is substantially better than the baseline that took 4 time steps to achieve 94.11% accuracy. On ImageNet, the baseline performance was 72.68% accuracy in 14 time steps, and it only surpassed 70% accuracy in 4 time steps. In contrast, the optimal performance achieved 82.47% accuracy in only 4.71 time steps. This significant performance gap suggests that there is potential for further improvements in SNNs by enabling runtime optimization.

5.3.3 Calculating th_c by Pareto Front

The value of the confidence threshold th_c is crucial for the final inference performance. Intuitively, a high th_c is a conservative dynamic strategy. It makes $\hat{P}_t > th_c$ happen less frequently and fewer examples will terminate early, limiting the latency improvements. In contrast, a low th_c is a radical dynamic strategy, and making it easy for $\hat{P}_t$ to surpass th_c and more examples will terminate early even though their output confidence $\hat{P}_t$ is low, bringing low latency but large accuracy drops. In other words, the selection of th_c reflects the fundamental latency-accuracy trade-off in SNNs. To compute the precise confidence threshold th_c, we propose using a low-cost Pareto Front method.

The optimization problem we are solving is essentially finding a series of th_c values that achieve the highest accuracy for each latency. Computing a Pareto Front requires the search space to be finite, but there are infinitely many choices of th_c in $[0, 1]$. To address this, we use a searching resolution of 0.1 to discretize the search space of th_c, which is sufficient to get an appropriate th_c that significantly reduces SNN inference time according to our experimental results. This suggests that our developed decision-making agent is highly robust to its parameter th_c. After discretizing the search space, there are only 11 settings of th_c. Then we draw the accuracy-vs-latency curve of $f_s(\cdot)$ with these 11 th_c in Figure 3 (The setting of $th_c = 0$ is the same as $th_c = 0.1$ on CIFAR-10 and they are both terminated at the first time step with a very low accuracy of 78.76%, and the setting of $th_c = 1$ is equivalent to the baseline that does not apply any dynamic strategy. Thus, we only show 8 candidate solutions in this figure.), by which a series of Pareto efficient choices of th_c can be obtained. Then a th_c which can give the lowest latency and no accuracy loss compared to the baseline solution is chosen from these 8 candidate solutions and used in Dynamic Confidence. For example, $th_c = 0.6$ will be selected in Figure 3.

6. Experiment

6.1. Experimental Setup

Datasets. The proposed Dynamic Confidence is validated on CIFAR-10 and ImageNet. CIFAR-10 [23] contains 60,000 images, divided into 10 classes. There are 50,000 training images and 10,000 test images. The size of each image is 32x32 pixels and their format is RGB. ImageNet [6] is a large object recognition dataset, and the version we use in our experiments is ILSVRC-2012. ILSVRC-2012 has over one million labeled RGB examples, and 1000 object classes. It has 50,000 images for validation.
Network Architectures. For CIFAR-10 we experiment with VGG-16 and ResNet-18 and for ImageNet we experiment with VGG-16 and ResNet-50. Max-pooling is replaced by average-pooling to facilitate ANN-to-SNN con-

version.
QCFS and QFFS. One of the essential features of the proposed Dynamic Confidence can be applied to a wide range of ANN-to-SNN conversion methods on the fly including the standard data-based normalization [8] and 99.9% data-based normalization [38], resulting in significant reductions in inference latency. However, this paper is focused on exploiting the fast-response potential of SNNs, so we only highlight the results on ANN-to-SNN conversion methods that achieve ultra-low inference latency. Specifically, Dynamic Confidence is demonstrated on SNNs built using QCFS [3] and QFFS [26], both of which have demonstrated ultra-low latency with competitive accuracy on nontrivial datasets. For example, QCFS and QFFS have reported latencies of 64 and 4 time steps, respectively, achieving 72% accuracy on ImageNet. Both methods reduce SNN latency by reducing the information in the ANN model (refer to [26] for a comprehensive introduction to why reducing ANN activation precision can lead to a reduction in SNN latency) and suppressing noise in the SNN model. QCFS achieves noise suppression by simulating longer to amortize the negative impact of noise, while QFFS generates negative spikes to correct noise. Some recent low-latency SNN algorithms [16, 17] also follow these two methods to amortize or correct noise.

In our experiments, we duplicate QCFS and QFFS by quantizing the activation in ANNs to 2-bit by LSQ [12] in all hidden layers and keeping the output layer at full precision. All weights are in full precision. We adopt the soft-reset IF neuron and its variant in SNNs as described in [3, 26], and the neurons in the output layer only accumulate input currents [27]. Detailed equations about LSQ, QCFS, and QFFS are provided in Supplementary Material.
Dynamic Confidence. As described in the approach section, there are three steps to configure Dynamic Confidence:

1. Calibration using an ANN.

2. Scaling the output logits by α in the SNN converted from the calibrated ANN.

3. Calculating the confidence threshold th_c from the Pareto Front.

Steps 1 and 3 are conducted on the same validation set. The validation set is collected randomly from the training set and its size is 5,000 on CIFAR-10 and 50,000 on ImageNet.

During inference, the confidence is calculated in SNN outputs and sent to the decision-making agent whose threshold is th_c. A binary decision of terminating the inference in advance is made if the confidence value surpasses th_c.
Training Configurations. The quantized ANNs are fine-tuned on the pre-trained full-precision ANN models for 60 epochs, with a momentum of 0.9 and a weight decay of

Table 1. Latency advantages brought by Dynamic Confidence in 8 different experimental settings.

Dataset	Architecture	Method	Acc(ANN)(%)	Acc(SNN)(%)	Averaged time steps	Latency saving(%)
CIFAR-10	VGG-16	QCFS	92.41	92.50	30	
		QCFS + Dynamic Confidence	92.41	92.50	12.69	58%
		QFFS	92.41	92.41	6	
		QFFS + Dynamic Confidence	92.41	92.41	3.17	47%
	ResNet-18	QCFS	93.79	94.27	27	
		QCFS + Dynamic Confidence	93.79	**94.27**	11.51	57%
		QFFS	93.79	94.11	4	
		QFFS + Dynamic Confidence	93.79	94.11	**2.52**	41%
		Best Reported[3]	95.52	93.96	4	
		Best Reported[26]	93.12	93.14	4	
ImageNet	VGG-16	QCFS	72.40	73.30	74	
		QCFS + Dynamic Confidence	72.40	**73.30**	49.54	33%
		QFFS	72.40	72.52	4	
		QFFS + Dynamic Confidence	72.40	72.52	**2.86**	29%
	ResNet-50	QCFS	72.60	70.72	128	
		QCFS + Dynamic Confidence	72.60	70.72	81.81	36%
		QFFS	72.60	73.17	6	
		QFFS + Dynamic Confidence	72.60	73.17	4.42	26%
		Best Reported[3]	74.29	72.85	64	
		Best Reported[26]	71.88	72.10	4	

*A concurrent research [17] based on QCFS reported a higher accuracy. However, note that Dynamic Confidence is a runtime latency optimization method and can also apply to this SNN algorithm to further reduce its latency.

2.5×10^{-5}, and a loss function of cross entropy. The initial learning rate is 0.01 with an exponential learning rate decay. The batch size is 32. The training is implemented in PyTorch.

6.2. Speed and Power Advantages of Using Dynamic Confidence

While compromising accuracy can lead to even greater latency gains (by adopting a lower th_c, related results are shown in the Supplementary Material), our primary focus in the following sections is to optimize latency without sacrificing accuracy. Table 1 reports how much the average latency can be reduced by Dynamic Confidence in 8 different settings respectively. Note that after applying Dynamic Confidence to SNNs, the inference latency is heterogeneous for different examples, so the averaged latency is not necessarily an integer. The results show that Dynamic Confidence can bring a latency reduction of 41% to 58% on CIFAR-10, and a latency reduction of 26% to 36% on a more challenging dataset ImageNet without sacrificing accuracy. These substantial improvements in latency suggest the obvious benefits of adopting Dynamic Confidence to allow heterogeneous terminating time. Moreover, note that both QCFS and QFFS already provide ultra-fast solutions on rate-coded SNNs. Even though, applying Dynamic Confidence on top of these two fast SNN solutions can still render significant latency reduction.

The power cost of an SNN is roughly kept stable for each time step for a given SNN run on GPUs. Given this ob-servation, reducing the averaged time steps required by an SNN at runtime can also bring nontrivial gains in power efficiency. For example, by reducing the averaged latency from 30 time steps to 12.69 time steps in CIFAR-10; the power consumption of SNN simulations can be roughly reduced by 58%. The overhead of applying Dynamic Confidence may dilute this gain in power a little bit, which is discussed in Section 6.5.

6.3. Comparison with State-of-the-Art Methods

State-of-the-art performance of fast SNNs on CIFAR-10 and ImageNet is listed in Table 1. From this table, we can see that on CIFAR-10, the performance achieved by applying Dynamic Confidence in QFFS (94.11% in 2.52 time steps) outperforms the state-of-the-art results both in accuracy and latency. On ImageNet, we report an accuracy of 72.52% in 2.86 time steps, outperforming state-of-the-art results as well.

Not all neuromorphic hardware and SNN algorithms support the negative spikes used in QFFS. Thus, we also benchmark Dynamic Confidence on QCFS which does not use negative spikes. As shown in this table, even without negative spikes, Dynamic Confidence can still achieve an accuracy of 73.30% on ImageNet with an averaged latency of 49.54 time steps, outperforming the state-of-the-art method (72.85% in 64 time steps) by a significant margin.
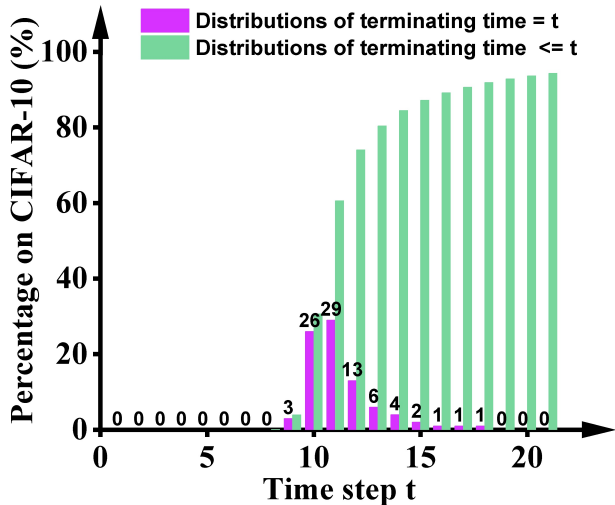
Figure 4. The distributions of terminating time after applying Dynamic Confidence on CIFAR-10. The network architecture is ResNet-18. We can see that instead of a fixing inference latency for each sample, different samples can have heterogeneous terminating times when adopting Dynamic Confidence.

6.4. Latency Distributions After Enabling Heterogeneous Terminating Time

We record the distributions of terminating time in Figure 4, to visualize how many samples are terminated by Dynamic Confidence early and facilitate a better understanding of Dynamic Confidence. The purple histogram shows the percentage of samples terminated at a time point t and the green histogram is the total terminated samples before t and at t. The dataset is CIFAR-10 and the network architecture is ResNet-18. It is apparent that the terminating time varies for different samples. There is a significant percentage of samples terminated at the time point 10 and 11 (26% and 29%), and up to 94.97% samples are terminated before the time point 22. Even though Dynamic Confidence terminates such a large percentage of samples at very early time steps, its inference accuracy is still guaranteed to be as high as 94.27%. This suggests that the terminating decision made by Dynamic Confidence is highly reliable.

6.5. Overhead of Dynamic Confidence

Dynamic Confidence has been demonstrated to provide significant improvements in terms of latency, as discussed in previous sections. Also, Dynamic Confidence is on-the-fly so it is a promising tool for further optimizing latency and spike counts on other low-latency SNN algorithms as well. However, to ensure that its benefits are maximized, it is important to be mindful of the potential overhead introduced by implementing Dynamic Confidence, as excessive costs can offset the benefits it provides. The overhead of Dynamic Confidence at the configuration phase is trivial, which is illustrated in the Supplementary Material. This section focuses on analyzing the overhead of Dynamic Confidence at runtime, particularly in the context of real-world applications where it holds greater significance.

At runtime, the main computational overhead of Dynamic Confidence is confidence calculation by Softmax, whose computational complexity is $\mathcal{O}(TL)$. T is the time step during SNN simulation and L is the number of output neurons. Benefiting from the rapid developments of low-latency SNN algorithms, T has been reduced significantly in recent years. For example, when conducting Dynamic Confidence with QFFS and CIFAR-10, T can be as small as 4. As for L, since the main application scenario of SNNs is small-scale edge-computing tasks, L is limited in most cases. For instance, L is 10 in CIFAR-10 and 11 in DVS-Gesture. Therefore, compared to the 26% to 58% gains in latency reported in previous sections, the runtime overhead of Dynamic Confidence is insignificant, especially when applying Dynamic Confidence on low-latency SNNs and edge AI applications. While low overhead is crucial to our discussion, it can be justifiable to tolerate higher overhead if a dynamic strategy can achieve performance that approaches the upper bound of an SNN, such as the 82.47% accuracy on ImageNet in 4.71 time steps depicted in Figure 1.

6.6. Actual Inference Latency Reduction Recorded on GPUs

Table 2 presents the average latency per inference for SNN simulations on GPUs. The simulations were conducted on a NVIDIA-V100 GPU, with a batch size of 1. It is important to note that the reported latency in this table is a measurement of the actual inference speed of SNN models on GPUs, whereas the latency gains shown in Table 1 are calculated by recording the average simulated time step required per inference.

Table 2. Latency per inference with/without Dynamic Confidence. The unit is seconds. The lower the better.

Setting	no DC	DC
CIFAR-10, VGG-16, QCFS	0.097	0.047
CIFAR-10, VGG-16, QFFS	0.036	0.017
CIFAR-10, ResNet-18, QCFS	0.127	0.047
CIFAR-10, ResNet-18, QFFS	0.027	0.022
ImageNet, VGG-16, QCFS	0.315	0.213
ImageNet, VGG-16, QFFS	0.030	0.021
ImageNet, ResNet-50, QCFS	1.708	0.932
ImageNet, ResNet-50, QFFS	0.116	0.104

6.7. Spike Count Reduction

Similar to other SNN algorithm research, our proposed method is demonstrated and evaluated through the simu-

lations on GPUs, the real power saving when running on neuromorphic hardware is unknown to us. Given that neuromorphic hardware's energy consumption is dominated by the multicast routing of spikes, we report the data on the spike count reductions achieved by our method. This is to provide insight into its potential power-saving benefits.

Table 3 reveals that the implementation of Dynamic Confidence significantly lowers spike counts, most notably for QCFS. In contrast, when using QFFS configurations, the reduction is more modest. This indicates that although Dynamic Confidence invariably lowers spike counts, its effectiveness can differ based on the chosen architecture and specific settings.

Table 3. Spike counts per neuron per inference with/without Dynamic Confidence. The lower the better.

Setting	no DC	DC
CIFAR-10, VGG-16, QCFS	3.21	1.35
CIFAR-10, VGG-16, QFFS	0.32	0.31
CIFAR-10, ResNet-18, QCFS	3.02	1.29
CIFAR-10, ResNet-18, QFFS	0.34	0.25
ImageNet, VGG-16, QCFS	12.94	8.66
ImageNet, VGG-16, QFFS	0.52	0.45
ImageNet, ResNet-50, QCFS	21.63	13.84
ImageNet, ResNet-50, QFFS	0.53	0.51

7. Towards Greater Opportunities

Visualizing and Approaching the "Potential" of SNNs. Section 5.3.2 presents a method to calculate the upper-bound performance of an SNN, which opens up numerous exciting possibilities. With this approach, we can visualize the potential of an SNN when its correct terminating time is accurately captured and compare it with that of another SNN. Figure 1 depicts that SNNs can achieve highly competitive accuracy (in some cases, even surpass the capacity of the original ANN model) in just a few time steps. This potential can be explained by viewing the SNN as an ensemble network. The input-output response curve of spiking neurons in an SNN is highly dependent on its terminating time point. By stopping the simulation at different time steps, we effectively select a different subnetwork with a distinct input-output response curve. For example, if an SNN simulates 10 time steps, it is essentially an ensemble of 10 subnetworks. These ten subnetworks share the same weights, but their input-output response curve is different. For different input samples, we can select the most appropriate subnetwork from these 10 subnetworks to do inference, simply by terminating this SNN at the corresponding time point. Therefore, the temporal dimension makes SNNs a highly efficient ensemble method.

SNNs have huge latent potential, the key question is how to unleash this. In this paper, we exemplify a method to partly unlock this potential by 1). formulating a high-quality estimation of the ground truth and 2). constructing a highly-reliable decision-making agent.

Further Applications of Dynamic Confidence. To improve the implemented performance of Dynamic Confidence on neuromorphic hardware, it may be desirable to use an alternative mechanism to the Softmax function. One option is to use a k-winners-take-all (k-WTA) spiking neural architecture whereby lateral inhibition between output neurons gates outputs. This could make Dynamic Confidence more amenable to implementation directly on neuromorphic hardware. Dynamic Confidence is possible to be applied to SNNs with timing-based encoding methods such as rank order coding [14], latency coding [34], and Time-to-First-Spike Coding [35], and to SNNs with surrogate gradients training methods to further leverage the temporal information in SNNs. In timing-based encoding methods, they may need a new metric than Softmax to represent confidence. It's worth noting that, even when using a superior confidence metric, the upper-bound performance achieved with this metric still aligns with our calculations.

8. Conclusion

This study critically examines the ways of applying dynamic strategies to spike-based models to optimize inference latency by Dynamic Confidence. Essentially, we formulate confidence in SNNs and use it to decide whether terminate inference or wait more time steps to accumulate more evidence and get a more reliable prediction. The major challenge of introducing the concept of confidence to SNNs is that, unlike confidence in ANNs, confidence in SNNs evolves with time, so some modifications need to be applied to leverage the information carried by confidence. A simple decision-making agent is then constructed to decide when to stop inference. Dynamic Confidence improves the performance of SNNs in the aspect of latency and power for exploring their potential on becoming a strong candidate for low-power low-latency edge computing.

9. Acknowledgments

References

[1] Amjad Almahairi, Nicolas Ballas, Tim Cooijmans, Yin Zheng, Hugo Larochelle, and Aaron Courville. Dynamic ca-

pacity networks. In *International Conference on Machine Learning*, pages 2549–2558. PMLR, 2016.

[2] Guillaume Bellec, Franz Scherr, Anand Subramoney, Elias Hajek, Darjan Salaj, Robert Legenstein, and Wolfgang Maass. A solution to the learning dilemma for recurrent networks of spiking neurons. *Nature communications*, 11(1):1–15, 2020.

[3] Tong Bu, Wei Fang, Jianhao Ding, PengLin Dai, Zhaofei Yu, and Tiejun Huang. Optimal ann-snn conversion for high-accuracy and ultra-low-latency spiking neural networks. In *International Conference on Learning Representations*, 2021.

[4] Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. Dynamic convolution: Attention over convolution kernels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11030–11039, 2020.

[5] Manon Dampfhoffer, Thomas Mesquida, Alexandre Valentian, and Lorena Anghel. Are SNNs Really More Energy-Efficient Than ANNs? an In-Depth Hardware-Aware Study. *IEEE Transactions on Emerging Topics in Computational Intelligence*, pages 1–11, 2022.

[6] 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*, pages 248–255. Ieee, 2009.

[7] Shikuang Deng and Shi Gu. Optimal conversion of conventional artificial neural networks to spiking neural networks. *arXiv preprint arXiv:2103.00476*, 2021.

[8] Peter U Diehl, Daniel Neil, Jonathan Binas, Matthew Cook, Shih-Chii Liu, and Michael Pfeiffer. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. ieee, 2015.

[9] Jianhao Ding, Tong Bu, Zhaofei Yu, Tiejun Huang, and Jian Liu. Snn-rat: Robustness-enhanced spiking neural network through regularized adversarial training. *Advances in Neural Information Processing Systems*, 35:24780–24793, 2022.

[10] Jianhao Ding, Zhaofei Yu, Yonghong Tian, and Tiejun Huang. Optimal ann-snn conversion for fast and accurate inference in deep spiking neural networks. *arXiv preprint arXiv:2105.11654*, 2021.

[11] Jianhao Ding, Jiyuan Zhang, Zhaofei Yu, and Tiejun Huang. Accelerating training of deep spiking neural networks with parameter initialization. 2021.

[12] Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. *arXiv preprint arXiv:1902.08153*, 2019.

[13] Xitong Gao, Yiren Zhao, Łukasz Dudziak, Robert Mullins, and Cheng-zhong Xu. Dynamic channel pruning: Feature boosting and suppression. *arXiv preprint arXiv:1810.05331*, 2018.

[14] Elena Garcia, Maria Antonia Jimenez, Pablo Gonzalez De Santos, and Manuel Armada. The evolution of robotics research. *IEEE Robotics & Automation Magazine*, 14(1):90–103, 2007.

[15] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR, 2017.

[16] Zecheng Hao, Tong Bu, Jianhao Ding, Tiejun Huang, and Zhaofei Yu. Reducing ann-snn conversion error through residual membrane potential. *arXiv preprint arXiv:2302.02091*, 2023.

[17] Zecheng Hao, Jianhao Ding, Tong Bu, Tiejun Huang, and Zhaofei Yu. Bridging the gap between anns and snns by calibrating offset spikes. *arXiv preprint arXiv:2302.10685*, 2023.

[18] Weizhe Hua, Yuan Zhou, Christopher M De Sa, Zhiru Zhang, and G Edward Suh. Channel gating neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[19] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens Van Der Maaten, and Kilian Q Weinberger. Multi-scale dense networks for resource efficient image classification. *arXiv preprint arXiv:1703.09844*, 2017.

[20] Xiping Ju, Biao Fang, Rui Yan, Xiaoliang Xu, and Huajin Tang. An FPGA Implementation of Deep Spiking Neural Networks for Low-Power and Fast Classification. *Neural Computation*, 32(1):182–204, 2020.

[21] Seijoon Kim, Seongsik Park, Byunggook Na, and Sungroh Yoon. Spiking-YOLO: Spiking Neural Network for Energy-Efficient Object Detection. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(07):11270–11277, 2020.

[22] Youngeun Kim, Yuhang Li, Hyoungseob Park, Yeshwanth Venkatesha, and Priyadarshini Panda. Neural architecture search for spiking neural networks. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXIV*, pages 36–56. Springer, 2022.

[23] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[25] Chen Li and Steve Furber. Towards biologically-plausible neuron models and firing rates in high-performance deep spiking neural networks. In *International Conference on Neuromorphic Systems 2021*, pages 1–7, 2021.

[26] Chen Li, Lei Ma, and Steve B Furber. Quantization framework for fast spiking neural networks. *Frontiers in Neuroscience*, page 1055, 2022.

[27] Yuhang Li, Shikuang Deng, Xin Dong, Ruihao Gong, and Shi Gu. A free lunch from ann: Towards efficient, accurate spiking neural networks calibration. *arXiv preprint arXiv:2106.06984*, 2021.

[28] Yuhang Li, Shikuang Deng, Xin Dong, and Shi Gu. Converting artificial neural networks to spiking neural networks via parameter calibration. *arXiv preprint arXiv:2205.10121*, 2022.

[29] Ji Lin, Yongming Rao, Jiwen Lu, and Jie Zhou. Runtime neural pruning. *Advances in neural information processing systems*, 30, 2017.

[30] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

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

[32] Emre O Neftci, Charles Augustine, Somnath Paul, and Georgios Detorakis. Event-driven random back-propagation: Enabling neuromorphic deep learning machines. *Frontiers in neuroscience*, 11:324, 2017.

[33] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *Advances in neural information processing systems*, 32, 2019.

[34] Zihan Pan, Jibin Wu, Malu Zhang, Haizhou Li, and Yansong Chua. Neural population coding for effective temporal classification. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019.

[35] Seongsik Park, Seijoon Kim, Byunggook Na, and Sungroh Yoon. T2fsnn: Deep spiking neural networks with time-to-first-spike coding. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2020.

[36] Michael Pfeiffer and Thomas Pfeil. Deep learning with spiking neurons: opportunities and challenges. *Frontiers in neuroscience*, page 774, 2018.

[37] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[38] Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, Michael Pfeiffer, and Shih-Chii Liu. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in neuroscience*, 11:682, 2017.

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

[40] Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S Davis, Kristen Grauman, and Rogerio Feris. Blockdrop: Dynamic inference paths in residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8817–8826, 2018.