CADE: Cosine Annealing Differential Evolution for Spiking Neural Network

Runhua Jiang[†]*, Guodong Du[‡]*, Shuyang Yu[†], Yifei Guo[†], Sim Kuan Goh[†], Ho-Kin Tang[‡]
[†]*School of Electrical Engineering and Artificial Intelligence, Xiamen University Malaysia*, Selangor, Malaysia
[‡]*School of Science, Harbin Institute of Technology (Shenzhen),* Shenzhen, China.
*contributed equally to this paper.

***Abstract*—Spiking neural networks (SNNs) have gained prominence for their potential in neuromorphic computing and energy-efficient artificial intelligence, yet optimizing them remains a formidable challenge for gradient-based methods due to their discrete, spike-based computation. This paper attempts to tackle the challenges by introducing Cosine Annealing Differential Evolution (CADE), designed to modulate the mutation factor (F) and crossover rate (CR) of differential evolution (DE) for the SNN model, i.e., Spiking Element Wise (SEW) ResNet. Extensive empirical evaluations were conducted to analyze CADE. CADE showed a balance in exploring and exploiting the search space, resulting in accelerated convergence and improved accuracy compared to existing gradient-based and DE-based methods. Moreover, an initialization method based on a transfer learning setting was developed, pretraining on a source dataset (i.e., CIFAR-10) and fine-tuning the target dataset (i.e., CIFAR-100), to improve population diversity. It was found to further enhance CADE for SNN. Remarkably, CADE elevates the performance of the highest accuracy SEW model by an additional 0.52 percentage points, underscoring its effectiveness in fine-tuning and enhancing SNNs. These findings emphasize the pivotal role of a scheduler for F and CR adjustment, especially for DE-based SNN. Source Code on Github: https://github.com/Tank-Jiang/CADE4SNN.**

***Index Terms*—Differential Evolution, Spiking Neural Network, Robustness, Spiking Element Wise model**

I. Introduction

In the last decade, there has been a huge development in artificial neural network theories and applications [1]–[3]. It begins with the first-generation multi-layer perceptron to the many state-of-the-art techniques in the second-generation deep neural networks (DNNs) trained using gradient descent. Despite this great advancement, ANNs still lag behind the biological neural networks in terms of energy efficiency and abilities for online learning. Though ANNs are brain-inspired, they are fundamentally different in structure, neural computations, and learning rules compared to the biological neural network. This leads to the third generation of neural networks, spiking neural networks (SNNs), which are considered a breakthrough in addressing the bottlenecks of ANNs. Unlike ANNs that continuously compute over all nodes, SNNs only activate specific neurons in response to stimuli, significantly reducing energy consumption [1]. This attribute is particularly beneficial for processing sensor data or performing tasks where only a subset of the data changes over time. Moreover, SNNs excel in dynamic and evolving environments through mechanisms such as online learning, enabling them to adapt continuously and in real-time to new data. This makes SNNs highly effective for applications that require immediate processing of incoming data streams without the need to pause for retraining, as they leverage updated models instantaneously [4]. With the ability to address the major challenges of deep neural networks, such as high resource requirements including energy consumption, data storage, and computational costs, Spiking Neural Networks (SNNs) [5], [6] have emerged as a promising alternative to conventional Artificial Neural Networks (ANNs) in recent years.

However, the training of SNNs can be challenging due to their non-differentiable nature of spiking events. Most prior SNN methods use ANN-like architectures (e.g., VGGNet or ResNet), which could provide sub-optimal performance for temporal sequence processing of binary information in SNNs. Traditional gradient-based learning methods often suffer when it comes to training SNN. An alternative to gradient-based methods Evolution algorithms (EAs). By using EAs to optimize the weights and architecture of SNNs, researchers can bypass the need for gradient computation, offering a robust alternative to train SNNs for a variety of tasks. The combination of DE's global search capability and SNN's efficient information processing holds promise for developing advanced computational models that are both powerful and energy-efficient.

EAs are grounded in Darwinian evolutionary theory and are recognized for their effectiveness in tackling complex problems. They mimic the natural process of survival of the fittest within a population, displaying remarkable robustness and adaptability in identifying global solutions to various optimization challenges. The initial development of evolutionary computing traces back to the 1950s with the introduction of the Genetic Algorithm (GA) [8], which incorporates Charles Darwin's theory of natural selection. Subsequent decades, particularly the 1960s and 1970s, saw the emergence of other evolutionary algorithm forms like Evolutionary Programming (EP) [9] and Evolution Strategies (ES) [10]. While Differ-

This research is supported by the Ministry of Higher Education Malaysia through the Fundamental Research Grant Scheme (FRGS/1/2023/ICT02/XMU/02/1), and Xiamen University Malaysia through Xiamen University Malaysia Research Fund (XMUMRF/2022-C10/IECE/0039 and XMUMRF/2024-C13/IECE/0049). This work also acknowledges support from the National Natural Science Foundation of China (Grant No. 12204130), Shenzhen Start-Up Research Funds (Grant No. HA11409065) and the HITSZ Start-Up Funds (Grant No. X2022000). Corresponding: simkuan.goh@xmu.edu.my, denghaojian@hit.edu.cn

ential Evolution (DE) has been applied in optimizing Deep Neural Networks (DNNs), the effectiveness of this approach in training conventional DNN models is not fully established. This is partly because most existing research tends to focus on custom-designed network architectures rather than standard ones.

In this work, we hypothesize existing SEW that compete for survival and breeding in evolution will show higher accuracy and stronger tolerance to data set and design an experiment, to examine the influence of EA on DNNs' corruption robustness. Specifically, pre-trained SEW CIFAR-100 evolved without changing the original dataset, loss function, and network architecture.

II. Related Works

A. Training methods for Spiking Neural Network

Recent research revealed that deep spiking neural networks (SNNs) have the potential for improving the latency and energy efficiency of deep neural networks through data-driven event-based computation [11]. However, SNNs have not quite reached the same accuracy levels of ANNs in traditional machine learning tasks practically. A main reason for this is the lack of efficient training algorithms for deep SNNs due to the non-differentiable architecture. Meanwhile current adequate algorithms are well prepared for deep neural networks. There is a research gap here. Spike-Time-Dependent Plasticity (STDP) [12] is a learning rule commonly used in spiking neural networks (SNNs) to update the synaptic weights between neurons based on the precise timing of pre- and post-synaptic spikes. It is inspired by the observed phenomenon in biological neurons where the strength of synaptic connections is modulated by the relative timing of pre- and post-synaptic spikes.

To deal with this non-differentiable transfer function, various methods have been proposed to address the non-differentiable transfer function. On the one side, some researches improve the standard backpropagation. On the other side, non-differentiable spike generation function was handled using surrogate models (e.g., Gaussian process [18]–[20]). A novel method enables error backpropagation in deep spiking neural networks (SNNs) by treating neuron membrane potentials as differentiable signals and spike discontinuities as noise [13]. This approach, which uses principles from optimal control theory, supports efficient training across various SNN architectures without needing approximations. Spike-based backpropagation [14] is proposed that can enable training deep convolutional SNNs directly (with input spike events). This derivative method can work well for leaky behavior of Leaky integrate and fire (LIF) neurons.

On the other side, recently proposed indirect methods involve using different data representations between training and processing. For example, training a conventional artificial neural network (ANN) and then using conversion algorithms to transfer the weights into equivalent deep spiking neural networks (SNNs). This includes advanced techniques such as Logarithmic Temporal Coding, which efficiently reduces computational costs by adjusting the spike encoding during the conversion of a trained ANN to an SNN, offering a streamlined method for achieving competitive classification performance with reduced computational demand [15]. But converting deep ANNs into SNNs may have come at the cost of performance losses without a notion of time, to sparsely firing, event-driven SNNs. To address these issues, recent advancements introduce optimization methods that minimize performance loss during the conversion of ConvNets and fully connected networks into SNNs, significantly improving both accuracy and latency [16]. However, the main problem is that details of statistics in spike trains that go beyond ideal mean rate modeling, such as required for processing practical event-based sensor data cannot be precisely represented by the signals used for training. So far it has only been possible to train single layers with the learning rules operating directly on spike trains.

B. Differential Evolution Algorithm

Differential Evolution is a robust, stochastic evolutionary algorithm that has gained widespread recognition for its efficacy in solving complex optimization problems across diverse domains. However, it faces a significant challenge in the realm of hyperparameter tuning. Hyperparameters in DE, primarily comprising the population size, mutation factor, and crossover rate, play a crucial role in dictating the algorithm's performance.

Consequently, numerous research efforts have been initiated to propose various mechanisms to address the hyperparameter challenge in DE. For instance, the SADE algorithm includes an adaptive mechanism that autonomously adjusts strategy parameters like mutation and crossover strategies and rates during the evolutionary process [17]. This self-adaptive approach allows SADE to dynamically tailor its behavior according to the specific requirements of the optimization problem, thereby enhancing its efficiency and robustness. Similarly, the Success-History based Adaptive Differential Evolution (SHADE) algorithm introduces a sophisticated adaptation technique that relies on historical performance data [21]. SHADE maintains a memory that records successful parameter settings from previous generations. These records are then used to guide the mutation and crossover rates, enabling a more informed and targeted search process. By incorporating this historical success information, SHADE effectively navigates complex optimization landscapes, achieving superior convergence rates and solution quality compared to traditional DE and other variants.

While SADE and SHADE were innovative, the field has evolved with new algorithms that integrate machine learning to enhance performance in complex scenarios, often outperforming traditional methods [22]. This highlights the continuous advancement and dynamic nature of optimization technologies.

C. Robustness

In the field of computer vision, a key objective is to enhance deep learning systems to rival or surpass the robustness of the

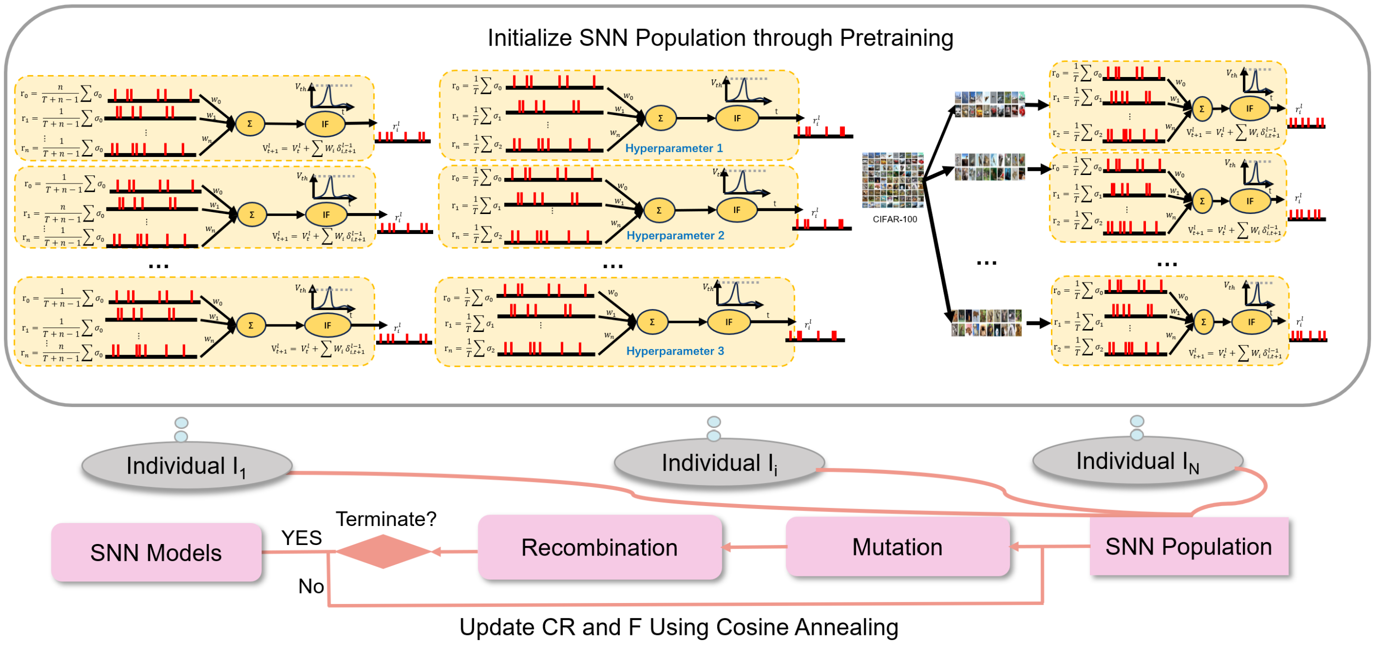

Fig. 1. Pipeline of the proposed CADE for Spiking Neural Network.

human visual system. This pursuit involves addressing various challenges, with researchers exploring different approaches. Carilini and Wagner have made strides in increasing the resilience of networks against adversarial examples, while Hendrycks et al. have concentrated on dealing with unknown unknowns, anomalous inputs, and bolstering networks against input corruption [14]. In this context, the introduction of datasets like Imagenet-C, featuring 75 common visual corruptions at varying intensities, and CIFAR-100-C, designed for assessing corruption robustness, has been significant.

Moreover, the exploration of spiking neural networks (SNNs) has opened new avenues in the quest for robustness. SNNs, which more closely mimic the biological processes of the human brain, offer potential advantages in terms of energy efficiency and latency. Their inherent temporal dynamics and spike-based information processing can lead to enhanced robustness against certain types of noise and distortions, making them a promising area of research in developing resilient deep learning models for computer vision tasks.

III. Proposed Work

In our work, we propose the Cosine Annealing Differential Evolution (CADE) algorithm to improve the accuracy and robustness of the SEW model without changing the original dataset, loss function, and network architecture, illustrated in Fig. 1. Subsequently, the overall performance of robustness of our proposed optimization scheme is evaluated on CIFAR-100-C.

A. CADE Algorithm

DE is a globally-oriented stochastic optimization technique that operates on a population basis. Its primary function is to conduct an extensive search for optimal solutions. In DE, a set of potential solutions, referred to as ”individuals,” constitutes the population. Each individual represents a candidate solution to the optimization problem. The evolution of these individuals is a systematic process, where new candidate solutions emerge through a ”mutation” process. This is subsequently followed by a ”crossover” phase, which amalgamates these new solutions with the existing individuals, thus forming a fitter population. This iterative cycle of evolution continues until the process either reaches a pre-set number of iterations or the optimization goal attains a level of convergence deemed satisfactory.

The mutation factor (F), and crossover rate (CR) are critical hyperparameters in the DE algorithm. Properly tuning these hyperparameters is essential to strike a balance between exploration and exploitation, leading to efficient and effective optimization for various real-world problems. However, the optimal setting of these parameters is not straightforward and is highly problem-dependent, leading to the primary challenge in DE applications. Regarding the hyperparameter issue, this paper also proposes a variant of the DE algorithm. In CADE, the F and CR are dynamically varied throughout the evolutionary process, following the pattern of a cosine function. This concept is inspired by the annealing learning rate approach, where the learning rate is adjusted using a cosine function. The algorithm pseudocode is shown in the algorithm 1.

B. Strategies for CR and F update

Hyperparameter tuning plays a pivotal role in the performance of differential evolution algorithms, impacting their optimization capabilities across various iterations. Traditional

Algorithm 1 CADE Algorithm

```
Initialize population $P$ with random solutions
Initialize parameters: $F_{\text{min}}, F_{\text{max}} = F_{\text{init}}, CR_{\text{min}}, CR_{\text{max}} = CR_{\text{init}}$
Set termination criteria: max iterations, convergence threshold, etc.
Initialize current iteration $t = 0$
while not reached termination criteria do
    for each individual $x_i$ in population $P$ do
        Randomly select three distinct individuals $a, b, c$ from $P$
        Generate a trial solution $u$ using the mutation strategy: $u = a + F \cdot (b - c)$
        Apply crossover with probability
        $CR$: $v_j = \begin{cases} u_j, & \text{if rand()} \le CR \text{ or } j = \text{rand()}(1, D) \\ x_{ij}, & \text{otherwise} \end{cases}$
        Evaluate the fitness of $v$
        if $v$ is better than $x_i$ then
            Replace $x_i$ with $v$ in population $P$
        end if
    end for
    Update iteration counter: $t = t + 1$
    Strategy 1, 2, 3 or 4 for CR and F update
end while
return Best solution found in $P$
```

approaches often rely on static or manually adjusted hyperparameters, which may not optimally adapt to dynamic problem landscapes encountered during optimization processes. This paper introduces a novel update strategy that utilizes a cosine function transformation to dynamically adjust hyperparameters during iterations. By implementing this strategy, the algorithm can search and adapt to the most suitable hyperparameters, thereby enhancing model optimization effectively. We provide a comprehensive analysis of how different hyperparameters influence the algorithm's performance and demonstrate the superiority of our proposed cosine-based adaptation strategy through extensive experiments on benchmark functions. The results indicate significant improvements in convergence rates and solution accuracy, underscoring the importance of adaptive hyperparameter strategies in evolutionary computations.

To explore further the update strategy, this work consider four distinct approaches to dynamically adjust the parameters CR and F within the Differential Evolution algorithm framework. Each strategy offers a different method for parameter adaptation, potentially impacting the algorithm's ability to navigate the search space and converge on optimal solutions.

Strategy 1 updates both CR and F according to a cosine function that depends on the current iteration (t) and the maximum number of iterations (max_iterations), if the new individual (a solution candidate) does not lead to a better result. The updates aim to adjust the parameters dynamically over the course of the algorithm's run, enabling the balance between exploration and exploitation.

Strategy 1 Update rules for CR and F

```
if New individual is not better then
    $F = F_{\text{min}} + \frac{F_{\text{max}} - F_{\text{min}}}{2} \left(1 + \cos\left(\frac{\pi \cdot t}{\text{max\_iterations}}\right)\right)$
    $CR = CR_{\text{min}} + \frac{CR_{\text{max}} - CR_{\text{min}}}{2} \left(1 + \cos\left(\frac{\pi \cdot t}{\text{max\_iterations}}\right)\right)$
end if
```

Strategy 2 straightforwardly updates CR and F using the same cosine function as in Strategy 1, but it does not condition the update on the performance of the new individual. This suggests a consistent, periodic adjustment of the parameters regardless of the immediate outcomes.

Strategy 2 Update rules for CR and F

```
~~if New individual is not better then~~
    $F = F_{\text{min}} + \frac{F_{\text{max}} - F_{\text{min}}}{2} \left(1 + \cos\left(\frac{\pi \cdot t}{\text{max\_iterations}}\right)\right)$
    $CR = CR_{\text{min}} + \frac{CR_{\text{max}} - CR_{\text{min}}}{2} \left(1 + \cos\left(\frac{\pi \cdot t}{\text{max\_iterations}}\right)\right)$
end if
```

CR and F are updated as in Strategy 2, but after the deterministic update, a stochastic component is added by using random value in **strategy 3**, which introduces randomness into the new values of CR and F. The random.uniform() function generates a random number between the given min and max values for each parameter, adding an element of randomness to the updating process. **Strategy 4** combines the conditional approach of Strategy 1 with the stochastic element of Strategy 3. If the new individual is not better, both CR and F are first updated deterministically using the cosine function, and then they are further adjusted by adding a random value from a uniform distribution. This strategy seems to adapt the parameters based on the current success of the algorithm and introduces variability to escape local optima or to search new areas of the solution space.

Strategy 3 Update rules for CR and F

```
$F = F_{\text{min}} + \frac{F_{\text{max}} - F_{\text{min}}}{2} \left(1 + \cos\left(\frac{\pi \cdot t}{\text{max\_iterations}}\right)\right)$
$F = F + \text{random.uniform}(F_{\text{min}}, F_{\text{max}})$
$CR = CR_{\text{min}} + \frac{CR_{\text{max}} - CR_{\text{min}}}{2} \left(1 + \cos\left(\frac{\pi \cdot t}{\text{max\_iterations}}\right)\right)$
$CR = CR + \text{random.uniform}(CR_{\text{min}}, CR_{\text{max}})$
```

Strategy 4 Update rules for CR and F

```
if New individual is not better then
    $F = F_{\text{min}} + \frac{F_{\text{max}} - F_{\text{min}}}{2} \left(1 + \cos\left(\frac{\pi \cdot t}{\text{max\_iterations}}\right)\right)$
    $F = F + \text{random.uniform}(F_{\text{min}}, F_{\text{max}})$
    $CR = CR_{\text{min}} + \frac{CR_{\text{max}} - CR_{\text{min}}}{2} \left(1 + \cos\left(\frac{\pi \cdot t}{\text{max\_iterations}}\right)\right)$
    $CR = CR + \text{random.uniform}(CR_{\text{min}}, CR_{\text{max}})$
end if
```

C. Spiking Element Wise Model

Deep Spiking Neural Networks (SNNs) pose unique challenges for optimization using gradient-based methods due to their discrete, binary activations and intricate spatial-temporal dynamics. Given the remarkable achievements of ResNet in the field of deep learning, applying residual learning to train deep SNNs emerges as a logical progression. Inspired by this, the Spike-Element-Wise (SEW) residual block have been developed [7], specifically designed for SNNs to enable efficient residual learning. This block not only facilitates identity mapping but also effectively addresses the issues of vanishing and exploding gradients. As demonstrated in Fig. 2, the SEW residual block is illustrated as follows:

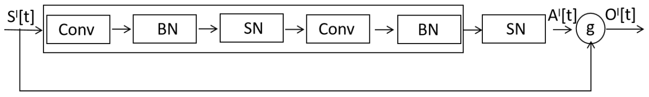

Fig. 2. SEW residual block

$$O'[t] = g(\mathrm{SN}(\mathrm{SF}(s'[t])), s[t]), \quad S'[t] = g(A'[t], S'[t]), \quad (1)$$

It's intriguing to note that SEW ResNet can easily implement identity mapping, leveraging the binary property of spikes. By capitalizing on the unique characteristics of spiking neurons and their binary spiking behavior, different element-wise functions, denoted as g, can be identified that satisfy identity mapping. The binary property of spikes introduces a distinctive approach to realizing identity mapping in SEW ResNet, offering a novel perspective on the implementation of this fundamental concept in the context of spiking neural networks.The specifics of these element-wise functions, as indicated in Table I. The two keys are formulation of downsample block and

TABLE I
LIST OF ELEMENT-WISE FUNCTIONS g.

Name	Expression of $g(A'[t], S[t])$
ADD	$A'[t] + S[t]$
AND	$A'[t] \wedge S[t] = A'[t] \cdot S[t]$
IAND	$\neg A'[t] \wedge S[t] = (1 - A'[t]) \cdot S[t]$

SEW ResNet can overcome vanishing/exploding gradient. For the downsample, in cases where the dimensions of the input and output of a residual block differ, the shortcut connection is adapted to perform downsampling. This is achieved by using convolutional layers with a stride greater than 1. This strategy ensures that the shortcut connection aligns with the spatial dimensions of the feature maps using spiking neurons(SN) in shortcut. The experimental results suggest that the ReLU before addition (RBA) block does not perform as well as the basic block . The SEW block is described as an extension of the RBA block. This implies that the SEW block incorporates certain characteristics of the RBA block while potentially introducing additional features or modifications that address the limitations observed in the RBA block. The SEW block utilizes AND, IAND, and ADD as potential element-wise functions (g). When AND and IAND are used, the output is spikes represented as binary tensors. This ensures that the "infinite outputs problem" in Artificial Neural Networks (ANNs), where unbounded activations can lead to numerical instability, does not occur in SNNs with SEW blocks. In the experiment, SEW ResNet-34 was used.

IV. EXPERIMENT

A. Dataset

In the experimental section of our study, this work focuses on two benchmark datasets, **CIFAR-10** and **CIFAR-100**, which are widely used in the field of machine learning for evaluating image classification models.

CIFAR-100-C stands as an extension to the CIFAR-100 dataset, designed specifically for the evaluation of model robustness. Unlike the standard CIFAR-100 dataset, CIFAR-100-C introduces a challenging set of conditions by intentionally corrupting the images with various types and levels of distortions. This dataset serves as a rigorous benchmark to assess the ability of machine learning models to generalize and perform well under diverse and adverse circumstances. CIFAR-100-C encompasses a broad spectrum of corruption types, including but not limited to blur, noise, and brightness variations. Each corruption type introduces a unique challenge, testing the model's resilience against a range of real-world distortions. The dataset provides images corrupted at different severity levels, allowing for a granular evaluation of a model's performance under varying degrees of perturbation.

B. Initial Population

In this paper, transfer learning is used to initialize population. This approach involved using pre-trained models from the CIFAR-10 dataset and subsequently fine-tuning them on the CIFAR-100 dataset. Based on the result, the transfer learning not only markedly accelerated the training process, but also improved accuracy compared to those trained from scratch or initialized differently.

Finetuning not only offers the advantage of time efficiency in the initial population creation but also presents a unique opportunity for diversification. The algorithm can generate individuals with distinct characteristics by employing various hyperparameter configurations during the fine-tuning phase. This diversification strategy plays a crucial role in enriching the population's genetic pool, promoting a broader exploration of the solution space. Consequently, the increased diversity among individuals enhances the algorithm's ability to adapt to different facets of the optimization landscape, potentially leading to more robust and effective solutions. It is obvious that transfer learning effectively leverages the feature extraction capabilities acquired on CIFAR-10 for application to CIFAR-100. Furthermore, this approach not only saves significant time but also enhances accuracy.

During the fine-tuning process, various data augmentation techniques were employed, including Smoothing [24], Mixup [25], and AutoAugment (AA) [26]. Smoothing involves attenuating the true labels from 1.0 to a smaller value (typically close to 1.0). This is a method to smooth the labels, which helps mitigate overfitting and enhances generalization performance. The main parameter is used to reduce the value of the true labels, for example, decreasing from 1.0 to 0.9. This value can be adjusted based on the specific task to balance the smoothness of the model and the fitting of the training data. Mixup:

is a data augmentation technique that generates new training samples by linearly combining the inputs and labels of two different samples. This encourages the model to be more robust during learning. AutoAugment is an augmentation strategy that automatically discovers the best data augmentation policies for a given dataset. It optimizes augmentation policies to improve model robustness and generalization. AutoAugment typically involves a set of sub-policies, each containing a specific data augmentation operation and its corresponding probability. Key parameters include the number of sub-policies, the number of operations in each sub-policy, the intensity of each operation, etc. These parameters can be adjusted by searching on the training set or using default values.

Another direct approach to initializing the population for the DE algorithm is to use the last-performing model obtained during a single fine-tuning process, which spans a duration of the last population size epochs, as the initial population.

V. Results

A. Hyperparameter effect

Based on the experimental results, it was found that hyperparameters continue to have a significant impact on the performance of the DE algorithm. The experiments revealed that when the crossover rate (CR) and mutation factor (F) are relatively low, there is an improvement in the optimization of Spiking Neural Networks (SNNs) by DE. This finding contrasts with previous observations where DE was applied to optimize Artificial Neural Networks (ANNs), suggesting that DE requires smaller values of F and CR to effectively optimize SNNs.

Influence of CR and F: Experiments as shown II have demonstrated that lower values of CR and F tend to improve CADE's effectiveness in optimizing Spiking Neural Networks (SNNs). A lower CR means that a smaller portion of the trial individual's components will be inherited by the target individual during the crossover operation. A lower F value results in a more conservative amplification of the difference between parent individuals when generating the trial individual.

F-init	CR-init	Strategy	t	Improvement	Accuracy
2e-3	2e-3	2	1	0.05	77.81
1e-5	1e-5	2	1	0.41	78.17
1e-6	1e-6	2	1	0.19	77.95
1e-7	1e-7	2	1	0.18	77.94
1e-5	1e-5	2	2	0.42	78.18
1e-6	1e-6	2	2	0.12	77.88
1e-7	1e-7	2	2	0.12	77.88
1e-5	1e-5	3	5	**0.52**	78.28
1e-5	1e-5	3	5	0.33	78.09
2e-3	2e-8	2	5	0.16	77.92
1e-5	1e-5	2	10	0.41	78.17

TABLE II
HYPERPARAMETERS

Our experimentation with hyperparameter tuning in CADE demonstrates the algorithm's sensitivity and responsiveness to parameter adjustments. The ability to enhance performance through fine-tuning underscores the importance of hyperparameter optimization in achieving the full potential of evolutionary algorithms like CADE when applied to intricate neural network models such as SEW.

B. Comparison of different initial population

In our study, the initial population for the CADE framework was generated using three distinct strategies, each designed to optimize the diversity and robustness of the models. The first strategy involved fine-tuning models by adjusting different time ratios, allowing us to explore a range of computational efforts and model complexities. The second strategy segmented the CIFAR-100 dataset into five equal parts, with each segment used to train a unique model. This approach aimed to develop distinct capabilities in each model by exposing them to different data subsets. The third strategy employed a sophisticated transfer learning technique, where models were first trained on the CIFAR-10 dataset to harness broad generalizable features and then fine-tuned on CIFAR-100 to adapt these models to more specific tasks. This method effectively combines the benefits of generalization with specialized adaptation, enhancing the evolutionary potential of the population. These strategies and their impacts on optimization outcomes are comprehensively documented in Table III of our research findings.

The table III above illustrates the optimization outcomes of various initial populations in the CADE framework. It presents the performance metrics both before and after CADE's optimization process. Notably, it is observed that initial populations derived from pretrained models exhibit the most substantial enhancement in optimization. This comparison underscores the impact of different initial populations on the efficacy of the CADE optimization approach. The initial population should ideally encompass a diverse range of solutions. This diversity ensures a broader exploration of the solution space, increasing the likelihood of finding globally optimal solutions rather than being trapped in local optima. In the context of machine learning and particularly with CADE, using pre-trained models as the initial population can be highly advantageous. Pretrained models are already trained on vast datasets and have learned general features that can provide a significant head start in the optimization process.

Initialization population	Different Time Ratio	Parts of CIFAR-100 Datasets	Pretrain Model from CIFAR-10
Acc Improvement (From→To) %	0.6 76.24→76.84	0.01 74.57→74.58	0.65 77.37→78.02

TABLE III
DIFFERENT INITIAL POPULATION

C. Comparison with CADE, SADE and SHADE

The notable performance improvement observed with CADE in optimizing the Spike-Element-Wise (SEW) model can be attributed to several factors.

	SADE	SHADE	CADE
Improvement	0.56	0.11	**0.65**

TABLE IV
COMPARISION OF DE VARIATION

First of all, the introduction of the Cosine Adaptive Mechanism in CADE allows for a more nuanced adjustment of the F and CR. The cosine-based variation provides a smoother transition during the optimization process, preventing abrupt changes that might disrupt the delicate balance required for optimizing the SEW model.

In our comparative study, we analyzed the performance of two adaptive differential evolution algorithms, SHADE (Success-History Adaptation Differential Evolution) and SADE (Self-Adaptive Differential Evolution), based on the same initial population generated through our outlined strategies. The effectiveness of each algorithm was assessed by measuring the improvement in accuracy over the initial population. This approach allowed us to directly compare how each algorithm adapts and optimizes under identical starting conditions. By quantifying the accuracy improvements, we were able to evaluate the relative efficacy of SHADE and SADE in refining the models' performance, providing a clear benchmark for evolutionary success within our CADE framework. This comparison is crucial in understanding which adaptation strategies yield the most gains in practical applications.

The comparative analysis may suggest that the self-adaptive mechanisms in SADE and adaptation of SHADE may not be as well-suited for the specific challenges posed by the optimization of SNNs. CADE, tailored with the cosine-based adaptive mechanism, proves to be more effective in navigating the complex optimization landscape associated with SEW. In summary, the success of CADE in optimizing the SEW model can be attributed to its adaptability, smooth parameter variation, and its alignment with the unique characteristics of spiking neural networks. These qualities make CADE particularly well-suited for the challenges posed by the optimization of SNNs compared to SADE and SHADE.

D. Robustness Experiment

In the experimental phase, this paper not only compared the efficacy of CADE, SADE, and SHADE in optimizing the SEW model but also conducted robustness experiments to assess the performance of the CADE-optimized model against the original pretrained model. **CADE-Optimized Model vs. Pretrained Model:** (the pretrained model is derived from the model obtained during the final popsize epochs of a fine-tuning process.) From the table V, the CADE-optimized SEW model demonstrated robustness across varying corruption levels, maintaining competitive accuracy under different degrees of noise and perturbations. Despite the challenging conditions introduced by CIFAR-100-C, the CADE-optimized model exhibited the retention of adaptive features acquired during the optimization process, contributing to its resilience. In specific corruption types, the CADE-optimized model outperformed the pretrained model, suggesting that the optimization process enhanced the model's ability to handle certain types of distortions and corruptions.

	Error		**mCE**	
	Finetune SGD	CADE	Finetune	CADE SGD
Gaussian Noise	80.98%	**80.94%**	100%	**99.94%**
Shot Noise	**75.43%**	75.47%	**100%**	100.05%
Impulse Noise	80.50%	**80.39%**	100%	**99.86%**
Defocus Blur	71.19%	**71.16%**	100%	**99.95%**
Glass Blur	91.60%	**91.59%**	100%	**99.98%**
Motion Blur	80.10%	**80.01%**	100%	**99.88%**
Zoom Blur	**80.70%**	80.66%	100%	**99.94%**
Snow	62.59%	62.45%	100%	99.77%
Frost	66.32%	**66.16%**	100%	**99.76%**
Fog	**71.50%**	71.60%	**100%**	100.13%
Brightness	45.71%	**45.64%**	100%	**99.83%**
Contrast	80.96%	**80.95%**	100 %	**99.98%**
Elastic Transform	**73.78%**	73.82%	**100%**	100.05%
Pixelate	67.30%	67.18%	100%	**99.81%**
Jpeg Compression	66.09%	**66.11%**	**100%**	100.03%
Average	72.98%	**72.94%**	100%	**99.93%**

TABLE V
PERFORMANCE (ERROR AND MCE) ON CIFAR-100-C.

mCE - mean Corruption Error (mCE). To assess robustness, the experiments employ the mean Corruption Error (mCE) as a performance measure, which quantifies the error rate of the current model compared to the base model for various types of corruption.

For both the CIFAR-100-C datasets, each corruption type comprises five distinct sensitivity levels. It's worth noting that while the base model employed in [23] was pretrain model finetuned by stochastic gradient descent algorithm, in our experiments, this paper use pretrain model as the base model. From the results, it is shown that the CADE algorithm can improve the robustness of SEW on CIFAR-100-C.

E. Different CR and F update methods

The experiment also involved contrasting the results of the trigonometric approaches with scenarios where the learning rate (LR) was fixed at its maximum and minimum values. This comparison was crucial to assess the effectiveness of dynamic versus static parameter settings.

	F_sin_CR_sin	F_sin_CR_cos	F_cos_CR_sin
Acc Improvement (From→To)%	0.7 77.37→78.07	0.7 77.37→78.07	0.55 77.37→77.92
	F_cos_CR_cos	Fixed_1e-5	Fixed_1e-9
Acc Improvement (From→To)	0.65 77.37→78.02	0.63 77.37→78	0.53 77.37→77.9

TABLE VI
DIFFERENT CR AND F UPDATE METHOD

The outcomes of these experiments demonstrated that the sine and cosine functions' application to F and CR variations is impactful. Both these trigonometric methods yielded better results compared to keeping the parameters fixed. It underscores the importance of adaptive parameter control in enhancing the performance of evolutionary algorithms.

VI. Discussion

The study noted that in SNNs, the weights often exhibit extreme values in comparison to traditional DNNs. This distinctive characteristic of SNNs implies that the optimization process needs to be handled with greater care. Aggressive exploration, potentially facilitated by higher values of CR and F, could disrupt the delicate balance within the structure of SNNs. As a result, a more nuanced and cautious approach to optimization is required to preserve the intricate dynamics of SNNs and avoid perturbing their unique structural characteristics.

	Max	Min	Most value distribution interval
SEW	195300	-10.2669	[-0.03,0.03]
ResNet-50	50040	-1.0489	[-0.04,0.05]

TABLE VII
Weight Distribution

VII. Conclusion

This paper proposed the CADE algorithm, integrating DE principles with SNN characteristics to enhance the performance and robustness of SNNs. The CADE algorithm dynamically adjusts hyperparameters F and CR using cosine functions, significantly improving the Spike-Element-Wise (SEW) model's optimization efficacy. Our experiments demonstrated that CADE not only outperforms traditional DE, SADE, and SHADE in terms of optimization but also showcases remarkable enhancements in model robustness across diverse conditions, particularly in handling data corruption scenarios like CIFAR-100-C.

Reflecting on the broader implications of our findings, it was observed that SNNs often exhibit extreme values in weights compared to traditional Deep Neural Networks (DNNs), highlighting the need for a nuanced optimization approach. The delicate balance within SNN structures requires cautious and precise optimization strategies, as aggressive exploration through high values of CR and differential weight factor F could potentially disrupt the intrinsic dynamics of SNNs. This insight reinforces the need for adaptive and thoughtful approaches in the optimization of neural networks, particularly those that mimic biological processes more closely than conventional models.

References

[1] Dongjin Lee, Seongsik Park, Jongwan Kim, Wuhyeong Doh, Sungroh Yoon, "Energy-efficient Knowledge Distillation for Spiking Neural Networks", ArXiv, 2021, vol. abs/2106.07172, doi=

[2] Goh, S. K., Abbass, H. A., Tan, K. C., Al-Mamun, A., Thakor, N., Bezerianos, A., & Li, J. (2018). Spatio–spectral representation learning for electroencephalographic gait-pattern classification. IEEE Transactions on Neural Systems and Rehabilitation Engineering, 26(9), 1858-1867.

[3] Low, W. S., Goh, K. Y., Goh, S. K., Yeow, C. H., Lai, K. W., Goh, S. L., Chuah, J. H. & Chan, C. K. (2023). Lower extremity kinematics walking speed classification using long short-term memory neural frameworks. Multimedia Tools and Applications, 82(7), 9745-9760.

[4] Mingqing Xiao, Qingyan Meng, Zongpeng Zhang, D.K. He, Zhouchen Lin, "Online Training Through Time for Spiking Neural Networks", ArXiv, 2022, vol. abs/2210.04195, doi=10.48550/arXiv.2210.04195

[5] Dennis V Christensen, Regina Dittmann, Bernabe Linares-Barranco, Abu Sebastian, Manuel Le Gallo, Andrea Redaelli, Stefan Slesazeck, Thomas Mikolajick, Sabina Spiga, Stephan Menzel, et al., "2022 roadmap on neuromorphic computing and engineering", Neuromorphic Computing and Engineering, vol. 2, no. 2, pp. 022501, 2022, IOP Publishing

[6] Souvik Kundu, Massoud Pedram, Peter A Beerel, "Hire-snn: Harnessing the inherent robustness of energy-efficient deep spiking neural networks by training with crafted input noise", Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 5209-5218, 2021

[7] Wei Fang, Zhaofei Yu, Yanqi Chen, Tiejun Huang, Timothée Masquelier, and Yonghong Tian. Deep Residual Learning in Spiking Neural Networks. *arXiv preprint arXiv:2102.04159*, 2022.

[8] J. H. Holland, Adaptation in Natural and Artificial Systems, The University of Michigan Press, Ann Arbor, 1975.

[9] L. J. Fogel, A. J. Owens, and M. J. Walsh, Artificial Intelligence through Simulated Evolution, John Wiley, New York, 1966.

[10] N. Hansen, D. V. Arnold, and A. Auger, "Evolution Strategies," in Springer Handbook of Computational Intelligence, pp. 871–898, 2015.

[11] M. Ji, Z. Wang, R. Yan, Q. Liu, S. Xu, and H. Tang, "SCTN: Event-based object tracking with energy-efficient deep convolutional spiking neural networks," Frontiers in Neuroscience, vol. 17, 1123698, 2023, Frontiers Media SA.

[12] F. Liu, W. Zhao, Y. Chen, Z. Wang, T. Yang, and L. Jiang, "SSTDP: Supervised Spike Timing Dependent Plasticity for Efficient Spiking Neural Network Training," Frontiers in Neuroscience, vol. 15, 756876, 2021.

[13] T. C. Wunderlich and C. Pehle, "EventProp: Backpropagation for exact gradients in spiking neural networks," arXiv preprint arXiv:2009.08378, 2020.

[14] C. Lee, S. S. Sarwar, P. Panda, G. Srinivasan, and K. Roy, "Enabling Spike-Based Backpropagation for Training Deep Neural Network Architectures," Frontiers in Neuroscience, vol. 119, 2020.

[15] M. Zhang, Z. Gu, N. Zheng, D. Ma, and G. Pan, "Efficient Spiking Neural Networks with Logarithmic Temporal Coding," IEEE Access, vol. 8, pp. 98156–98167, 2020.

[16] Z. Wang, S. Lian, Y. Zhang, X. Cui, R. Yan, and H. Tang, "Towards Lossless ANN-SNN Conversion under Ultra-Low Latency with Dual-Phase Optimization," arXiv preprint arXiv:2205.07473, 2022.

[17] M. G. Omran, A. Salman, and A. P. Engelbrecht, "Self-Adaptive Differential Evolution," in Proc. of the International Conference on Computational and Information Science, Berlin, Heidelberg: Springer Berlin Heidelberg, Dec. 2005, pp. 192–199.

[18] Kim, S., Park, S., Na, B., Kim, J., & Yoon, S. (2020). Towards fast and accurate object detection in bio-inspired spiking neural networks through Bayesian optimization. IEEE Access, 9, 2633-2643.

[19] Goh, S. K., Jun Lim, Z., Alam, S., & Pratap Singh, N. (2021). Tunnel gaussian process model for learning interpretable flight's landing parameters. Journal of Guidance, Control, and Dynamics, 44(12), 2263-2275.

[20] Goh, S. K., Singh, N. P., Lim, Z. J., & Alam, S. (2022). Interpretable tracking and detection of unstable approaches using tunnel gaussian process. IEEE Transactions on Aerospace and Electronic Systems.

[21] R. Tanabe and A. Fukunaga, "Success-History Based Parameter Adaptation for Differential Evolution," in Proc. of the 2013 IEEE Congress on Evolutionary Computation, 2013, pp. 71–78, IEEE.

[22] E. Morales, C. Juárez, E. García, and J. Sánchez, "Differential Evolution Based on Learnable Evolution Model for Function Optimization," in *Advances in Soft Computing: 18th Mexican International Conference on Artificial Intelligence, MICAI 2019, Xalapa, Mexico, October 27–November 2, 2019, Proceedings 18*, pp. 290–302, Springer, 2019.

[23] D. Hendrycks and T. Dietterich, "Benchmarking Neural Network Robustness to Common Corruptions and Perturbations," arXiv preprint arXiv:1903.12261, 2019.

[24] J. S. Simonoff, "Smoothing Methods in Statistics," Springer, 1996.

[25] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond Empirical Risk Minimization," arXiv preprint arXiv:1710.09412, 2018.

[26] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "AutoAugment: Learning Augmentation Policies from Data," arXiv preprint arXiv:1805.09501, 2019.