Imperial College London

Department of Earth Science and Engineering

MSc in Applied Computational Science and Engineering

Independent Research Project
Final Report

Spiking Forward-Forward: A Spiking Neural Network-Based Implementation of Forward-Forward Algorithm

by

Boxuan Zhu

Email: bz422@imperial.ac.uk
GitHub username: acse-bz422
Repository: https://github.com/ese-msc-2022/irp-acse-bz422

Supervisors:

Dr. Dan Goodman
Dr. Lluis Guasch

1st September 2023

Abstract

Spiking neural networks, next-generation neural networks inspired by biological neurons, are on the eve of extensive application. However, the non-differentiable characteristic that grant spiking neural networks their biological plausibility also pose challenges in their training. At the end of 2022, an algorithm called forward-forward was proposed, which is a learning algorithm that does not require back-propagation and allows local optimization of each hidden layer. This project aims to implement the Spiking Forward-Forward algorithm, a combination of spiking neural network and Forward-Forward, to alleviate the training difficulties of spiking neural networks. By completing the implementation of the Spiking Forward-Forward algorithm and conducting a series of experiments, the effectiveness of the algorithm was proven in this project. Furthermore, the Spiking Forward-Forward algorithm provides a new perspective for the study of spiking neural networks.

1 Introduction

The neurons of spiking neural networks are more similar to biological neurons than traditional artificial neural networks. By using spike trains to transmit information, spiking neural networks exhibit sparsity and are non-differentiable: since the spiking neural network processes data in the time dimension, and its neurons only release spikes when the membrane potential exceeds a threshold, only some of the neurons in the neural network are active at a specific moment [1]. Furthermore, since the spiking signal is binary, this leads to the fact that the activation functions in spiking neural networks are essentially non-differentiable [2]. Therefore, sparsity endows spiking neural networks with low power consumption and the possibility of being deployed on edge hardware; non-differentiability makes it difficult for spiking neural networks to be trained by back-propagation-based learning rules.

On the other hand, in the traditional neural network field, the forward-forward algorithm uses two forward passes instead of backpropagation. The forward-forward algorithm no longer uses error, but uses ”goodness” (how excited a layer of neurons) as the core of the learning rule. Specifically, the two forward passes use the original data and the adjusted adversarial data respectively, and the learning goal is to make the neural network excited for positive data but not excited for negative data [3].

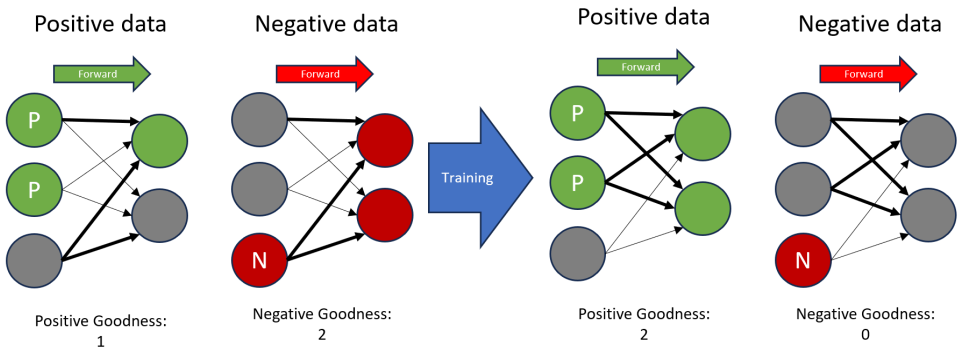

Figure 1: Forward-Forward algorithm increases the activity vector length of the layer for positive data, and decreases the activity vector length of the layer for negative data.

In addition, the algorithm allows the neural network to learn layer by layer - each layer of the neural network has its own goodness and a corresponding threshold. Each layer is trained through local gradient calculations to reach the parameter space position where the goodness of positive data is high enough (much greater than the threshold) and the goodness of negative

data is low enough (much smaller than the threshold). This characteristic also means that in the neural network trained by the Forward-Forward algorithm, each layer can complete its own training independently. The fact that the hidden layers no longer depend on each other makes the decentralization of Forward-Forward training possible.

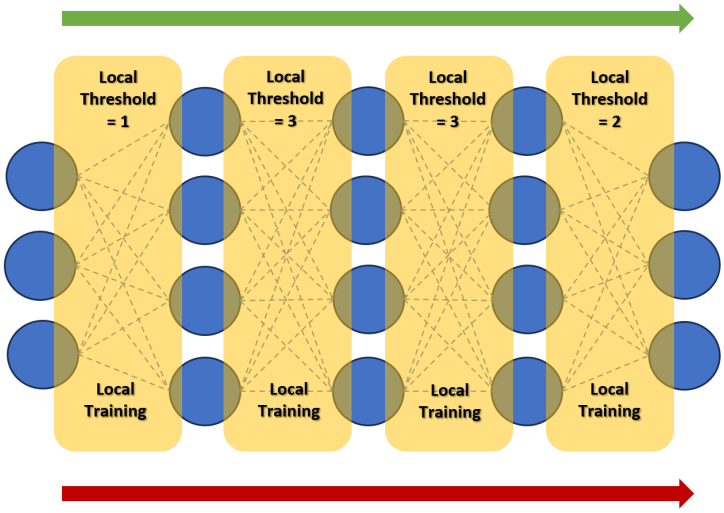

Figure 2: Forward-Forward algorithm allows each layer to have local goodness and threshold values, and allows the network to be trained layer by layer.

It can be found that the combination of spiking neural network and forward-forward algorithm has certain technical prospects: the combination has the potential to solve the training problem of spiking neural networks and at the same time provides a practical deployment scenario for the forward-forward algorithm - low-power neuromorphic hardware using spiking neural networks.

Contributions of the author of this project:

1. Researched and discovered the complementarity between Forward-Forward algorithm and spiking neural network.

2. Identify and solve the key issues of implementing Forward-Forward algorithm in spiking neural network.

3. Based on the above contributions, the author implemented the Spiking Forward-Forward algorithm independently. After that, the author designed and implemented a series of experiments using data sets in various forms and task fields. These experiments proved the effectiveness, robustness of the Spiking Forward-Forward algorithm, and discovered the limitations of the algorithm.

4. Based on the characteristics of the Spiking Forward-Forward algorithm, the experimental findings and experimental conclusions, the author conducted an in-depth discussion on the algorithm.

2 Spiking Forward-Forward

2.1 Algorithm Design

For the purpose of achieving the combination of the two in the new algorithm, two technical

problems had to be overcome:

1. The forward-forward algorithm needs to be redesigned to accommodate the binary data format of the spiking neural network and gain the ability to process spike trains in the time dimension; 2. The spiking neural network needs to provide a way for the forward-forward algorithm to perform local training at the layer level. This approach should not destroy the inherent sparsity of spiking neural networks.

Initially, consider the core idea of Forward-Forward: classify positive data and negative data by differentiating the excitement of the neural network between the two. Mathematically, this idea can be expressed as a probability calculation (only positive data is demonstrated for convenience):

$$p(\text{ positive }) = \sigma\left(\sum_i \gamma_i - \theta\right) \tag{1}$$

Among them, σ represents the logistic function, γ_i represents the square of the activity of the i-th unit in the hidden layer, and θ represents the threshold.

The training process of the neural network by the Forward-Forward algorithm is essentially to adjust the weights to achieve the minimum value of the following function:

$$f = \log\left(e^{\theta-\gamma} + 1\right) \tag{2}$$

In order to force the hidden layer of the neural network to learn more complex advanced features (instead of making judgments based on the length of the output vector of the previous layer), the FF algorithm will normalize the output vector of each layer [3]:

$$\gamma_i = \left(\max\left(0, (\mathbf{W}\frac{\mathbf{x}}{||\mathbf{x}||_2} + \mathbf{b})_i\right)\right)^2 \tag{3}$$

Where $\mathbf{W}$ represents the weight matrix, $\mathbf{x}$ represents the input vector, $\mathbf{b}$ represents the bias, and the activation function is ReLU.

However, in the spiking neural network, the spiking neurons uniformly emit a spiking signal with a value of 1 only when the membrane potential reaches the threshold, which in fact erases the length information of the output vector. Therefore, the Spiking Forward-Forward algorithm directly completes the normalization through the sending of spiking signals, and discards the L2 normalization part in the original algorithm.

$$\gamma_i = \left(H\left((\mathbf{W}\mathbf{x} + \mathbf{b})_i\right)\right)^2 \tag{4}$$

Where H represents the Heaviside step function of the spiking neural network.

Subsequently, for the data in the form of spike trains, the effective information contained in a single moment is less, and the spiking signal itself is more susceptible to noise interference. A reasonable goodness calculation and weight optimization strategy is:

1. Integrate the spiking signal of the sample in the form of discrete frames [4]; 2. When training each layer, the average value of all frames of a certain sample will be calculated for weight optimization; 3. The output with frame dimensions is passed intact to the next layer.

$$\mathbf{x}_n = \sum_{t=(n-1)\times\frac{T}{N}+1}^{n\times\frac{T}{N}} \mathbf{s}_t \tag{5}$$

$$\bar{\gamma} = \frac{1}{N}\sum_{n=1}^{N}\gamma_n \tag{6}$$

Where the original time length is T, the target number of frames is N, $\mathbf{x}_n$ is the integral of the nth frame, and $\mathbf{s}_t$ is the spike s at time t.

Furthermore, In the field of spiking neural networks, there are two methods to overcome the non-differentiable property to train spiking neural networks:

The spiking timing-dependent plasticity (STDP) model and the surrogate gradient model. STDP model adjusts the value of connection weights by tracking the firing timing of pre-synaptic and post-synaptic neurons [5]. The surrogate gradient model keeps the original non-differentiable activation function unchanged during the forward propagation of the spiking neural network, while in the backward propagation, it uses a surrogate function that approximates the original function but is differentiable [2]. Both models can be modified to be applied to the layer training required by the Forward-Forward algorithm, but there are differences in learning efficiency and learning ability.

Therefore, Spiking Forward-Forward adopts a combination of the two modes: For the connection weights of the input layer and the first hidden layer, the algorithm only uses the surrogate gradient for training; For the connection between hidden layers, the algorithm first uses the STDP scheme training based on reward modulation to help the hidden layer capture advanced features; After that, the algorithm is further trained using the surrogate gradient scheme.

Specifically, the surrogate gradient training used by Spiking Forward-Forward has been modified as follows: First of all, the function involved in the surrogate gradient is no longer a loss function, but a new function (goodness function) composed of goodness and threshold; Secondly, the algorithm no longer calculates the gradients of all layers at once, but uses the original non-leadable activation function in two forward propagations. When a layer needs to update weights, only the gradient of the layer's goodness function is calculated to indicate the direction and magnitude of the weight update [3].

2.2 Algorithm Implementation

The algorithms and series of experiments involved in this article can be reproduced through the Python language and the SpikingJelly neural network framework based on Pytorch. In order to make the algorithm run efficiently on mainstream platforms, it is recommended to use a GPU with at least 16GB of video memory and enable CUDA acceleration.

The implementation of the Spiking Forward-Forward algorithm is based on operating a data structure composed of objects of two classes: the class representing the layer and the class representing the net. Each net class object has a list variable consisting of a variable number of layer class objects.

In addition, the net class object also has a member function used to coordinate the training of layer class objects, and a member function used to perform category prediction - it iterates through each category and returns the category with the largest sum of goodness of all layers.

Each layer class has various member variables required by the spiking neuron (basic components of the spiking neuron, tau value, weight extremum and STDP learner), and contains a pair of float type thresholds corresponding to positive data and negative data. At the same time, the layer object has four member functions, the functions are: 1. Use both positive and negative data for training; 2. Two forward passes in a spiking neural network (positive and negative data); 3. Single forward pass; 4. Calculation of goodness. The first two member functions of the layer object will be called by the net object in the network training function, and the member functions that perform a single forward pass will be called in the category prediction. In addition, the algorithm also needs a global function that encodes the labels of the training data set during supervised learning.

2.3 Algorithm Optimization

As of the writing of this article, there are many optimization strategies for the forward-forward algorithm in the traditional neural network field. The Spiking Forward-Forward algorithm jointly adopts the following two strategies to improve the convergence speed and reduce the error rate:

Consolidation of weight updates

As of this writing, no optimization strategy that completely separates positive and negative updates has been reported. Instead, how closely positive updates alternate with negative updates directly determines the final test error rate. Therefore, for each layer, after performing two forward propagations, combining positive data goodness and negative data goodness to perform gradient calculation and weight update can effectively reduce the amount of calculation and reduce the error rate. Using this strategy in the Spiking Forward-Forward algorithm can reduce the error rate by 3%-5% on the Poission Spike MNIST dataset (locking epoch number, learning rate, random seed, threshold).

Layer Collaboration

Many papers have reported that the Forward-Forward algorithm has the problem of weak learning ability of the hidden layer. An intuitive and concise optimization strategy is to introduce a constant term into the goodness function required for gradient calculation—the sum of the goodness of each layer when it is transmitted through the neural network [6]. After the weight updates are combined, there are actually two constant terms in the function (for positive and negative data). Introducing a constant term ι actually allows the trained layer to see the subsequent layers and make more reasonable optimizations:

$$f = \log\left(e^{\theta-\gamma-\iota} + 1\right) \tag{7}$$

By using layer cooperation strategies, the error rate of traditional neural networks can be reduced by 1% to 3%. Using this strategy in the Spiking Forward-Forward algorithm can reduce the error rate by 10%-15% on the Poission Spike FashionMNIST dataset (locking epoch number, learning rate, random seed, threshold).

3 Experiments and Results

In order to extensively test the effectiveness and robustness of the Spiking Forward-Forward algorithm, this paper conducts experiments on four data sets: 1. MNIST processed with a Poisson encoder; 2. Spiking Heidelberg Digits (SHD) dataset; 3. Neuromorphic-MNIST; 4. FashionMNIST processed with a Poisson encoder.

Since the experiment is mainly based on supervised learning, and Spiking Forward-Forward does not have a loss function using label data in the usual sense. Therefore, this paper refers to the experimental method of Dr. Hinton's paper, uses one-hot encoding to encode the label data into the sample, and then performs training. When running the test data set, each test sample traverses each category, and returns the category with the largest sum of goodness of all layers [3].

3.1 Neuromorphic-MNIST-R-STDP Experiment

Before the key part of the algorithm, the method of weight optimization, was determined to use the combination of surrogate gradient calculation and STDP, a more radical solution using only STDP for weight optimization was developed and tested. This solution uses the STDP derivative algorithm R-STDP (STDP based on reward modulation) to update the weight in each layer. Specifically, the R-STDP learner is updated once at each time step of each forward propagation (including positive data and negative data): First, the learner captures and records the time it takes for each propagating presynaptic neuron and postsynaptic neuron to fire a spike. After that, depending on the source of this propagation (positive data or negative data), the algorithm will determine whether to use standard STDP rules (positive data) or reverse STDP rules (negative data) for this weight update.

This experiment uses a 2312 * 1000 * 1000 IF spiking neural network to integrate the event flow of each sample into 20 frames for goodness calculation and weight update. The learning rate is from 0.1 to 0.00003, and the threshold is 0.02.

This experiment reflects a series of problems: First, there is currently no dedicated hardware suitable for the STDP algorithm. Both the spiking neuron simulation and the spiking time monitoring involved in the algorithm need to be simulated using general-purpose hardware. Even with GPU and CUDA acceleration, the STDP Spiking Forward-Forward scheme consumes a lot of computing resources and training time. Second, as a derivative of STDP, R-STDP is essentially still an unsupervised learning algorithm. It is difficult to adapt to supervised learning tasks - although the reward function can be manually designed to force the neural network to distinguish between positive and negative data more effectively, the ability of the R-STDP algorithm to capture features is still weak.

As a result, the error rate of STDP Spiking Forward-Forward is 87% (±3%). Changing the epoch value and the learning rate will not significantly change the error rate. Therefore, it can be considered that STDP Spiking Forward-Forward cannot converge on Neuromorphic-MNIST.

It is worth noting that STDP Spiking Forward-Forward will output some regular predictions if the threshold is carefully tuned. For example, when the threshold is 0.02 and the learning rate is 0.01, STDP Spiking Forward-Forward will output all predicted values of 1 after several epochs. Such performance is actually consistent with the performance of the Forward-Forward algorithm in traditional neural networks in the early epoch [6]. On the other hand, using thresholds of other orders of magnitude (such as 2), the algorithm does not exhibit the behavior described above.

Therefore, the findings of this experiment lead to two preliminary conclusions:

1. The Spiking Forward-Forward algorithm is sensitive to the magnitude of the threshold.

2. Although STDP has limited capabilities, there is the possibility of combining it with gradient calculation schemes.

3.2 Poisson-Spike-MNIST experiment

This experiment confirmed the effectiveness of Spiking Forward-Forward and fixed the weight optimization method as a combination of surrogate gradient calculation and STDP.

Unlike the Neuromorphic-MNIST dataset, which uses a neuromorphic camera to scan handwritten digits, the Poisson-Spike-MNIST dataset uses a simple but effective Poisson encoder: Specifying the number of frames, the Poisson encoder can generate a corresponding number of spike morphology samples based on the original samples - they form a spike train that conforms to the Poisson distribution [7]. The Poisson encoder also provides convenience for tag encoding. As long as the required one-hot encoding is composed of several 0s and one 1, it can ensure that the tag will pulse correctly in every frame.

This experiment uses a 784*500*500 IF spiking neural network to integrate the event stream of each sample into 50 frames for goodness calculation and weight update. The learning rate is 0.001 to 0.00001, and the thresholds are 0.05 and 2.

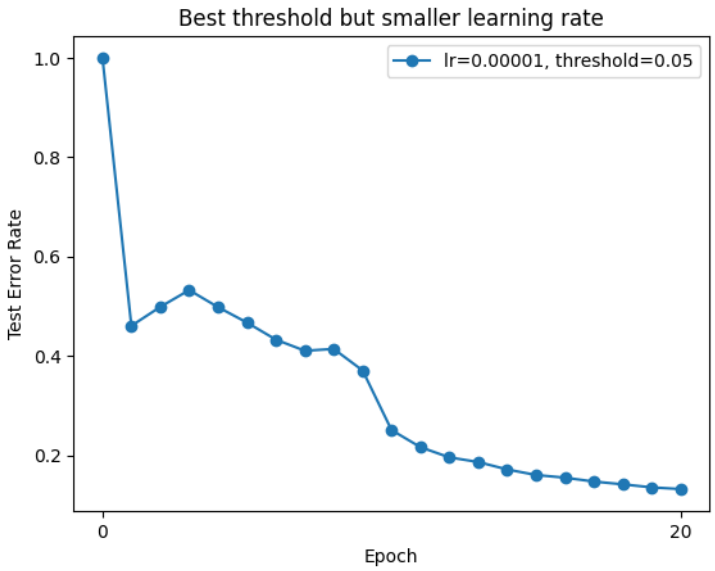

Figure 3: SFF convergence record

The experiment first uses surrogate gradient computation to update the gradient. Since the size of the neural network is reduced and the surrogate gradient calculation can be fully compatible with CUDA acceleration, the training cost is greatly reduced (in the case of batch size 500, the memory consumption is 1GB, and the training time of each epoch is about 2-3 minutes) . As a result, when the random seed is locked to 1000:

When the learning rate is 0.001 and the threshold is 0.05, the test error rate of the model at the first epoch is 24%, and the test error rate at the second epoch is 51%. This means that the model already has the ability to generalize and overfitting has occurred.

If the learning rate is reduced to 0.00001 and the threshold is 0.05, the model has a test error rate of 12% at the 20th epoch.

If the learning rate is increased to 0.0003 and the threshold is 0.05, the model will converge at the first epoch. The test error rate is 10%. Furthermore, if the STDP component is introduced into the above experimental model—the STDP weight update is performed before the surrogate gradient calculation, the test error rate can reach a minimum of 9.95%.

In particular, if the learning rate is kept constant at 0.0003 and the threshold is 2, the model will fail to converge (the error rate of the first epoch is 88.3%, and the error rate of the second epoch is 88.7%).

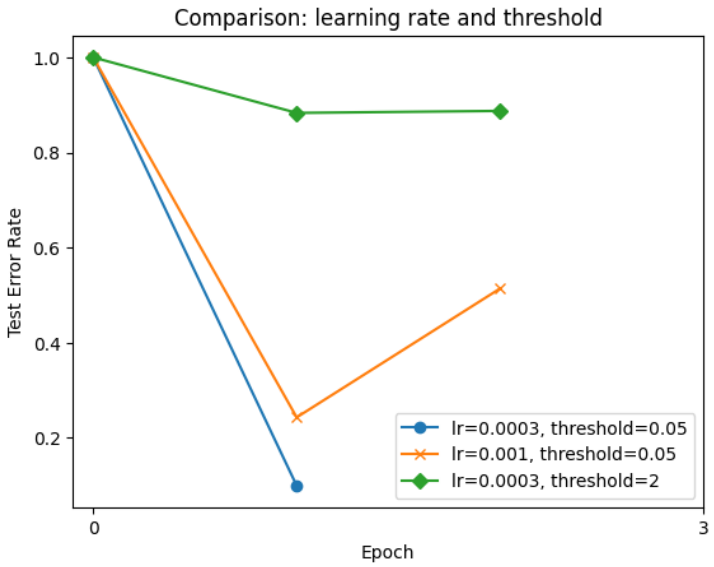

Figure 4: Hyperparameter difference comparison

As an additional error rate comparison, a SNN using Spiking Forward-Forward (learning rate 0.0003, threshold 0.05, 784*500*500), an ANN using Forward-Forward (784*500*500), and a single-layer SNN (784*10) using surrogate gradients for back-propagation were also deployed:

Dataset	Spiking Forward-Forward in SNN	Forward-Forward in ANN	BP in SNN
MNIST	11.03% (± 2% in 5 runs)	8.67% (± 2%)	7.45% (± 1%)

The findings of this experiment lead to two conclusions:

1. Spiking Forward-Forward does have learning and generalization capabilities when using surrogate gradient calculations. Moreover, Spiking Forward-Forward is enough for the spiking neural network to achieve performance close to that of the conventional (backpropagation-based) training algorithm on MNIST.

2. Before Spiking Forward-Forward performs surrogate gradient calculation, an STDP learner can be embedded for limited weight updates. This embedding does not have a significant negative impact on performance. And, in some cases, this embedding can have a positive impact on performance. With the random seed, epoch number, and learning rate locked, the STDP embedded 784*500*500 spiking neural network was used, and the test error rate on MNIST was 9.95%. This error rate is slightly better than the solution without embedding (10.12%).

3.3 Poisson-Spike-FashionMNIST experiment

This experiment is part of the robustness study of Spiking Forward-Forward and is also a test of the effectiveness of layer collaboration.

This experiment uses a 784*1000*400*200 IF spiking neural network to integrate the event stream of each sample into 50 frames for goodness calculation and weight update. The learning rate is 0.0003, the threshold is 0.06, and the number of epochs is 1. In addition, under the premise of fixing all hyperparameters, a spiking neural network of the same size but without layer collaboration is also deployed and trained simultaneously.

As a result, the test error rate of the Spiking Forward-Forward neural network without layer collaboration is 39.89% (±2%). The test error rate of the Spiking Forward-Forward neural network that introduces layer collaboration is 24.15% (±2%).

Dataset	Collaborative SFF error rate	SFF error rate
MNIST	11.03% (± 2% in 5 runs)	17.22% (± 2%)
FashionMNIST	24.15% (± 2%)	39.89% (± 2%)

For another comparison, the Forward-Forward algorithm based on traditional neural networks, which also uses only fully connected layers and introduces layer collaboration, has a test error rate of 12.0% on FashionMNIST [6]. This neural network has a learning rate of 0.001 and is trained for 150 epochs [6]. The gap between the two mainly comes from the difference between the traditional data set and the spike morphology data set (the Poisson encoder will lose certain features), the difference between the traditional neural network and the spiking neural network, and the difference in the number of epochs.

The findings of this experiment lead to three conclusions:

1. The Spiking Forward-Forward algorithm exhibits learning and generalization capabilities in multiple Poisson datasets.

2. Layer collaboration can indeed bring gains to Spiking Forward-Forward, which is directly reflected in the test error rate.

3. Although Spiking Forward-Forward can converge at a faster speed, there is a gap between the final performance and traditional neural networks.

3.4 Neuromorphic-MNIST-Surrogate Gradient Experiment

This experiment is part of the robustness study of Spiking Forward-Forward.

This experiment uses a 2312*1000*500 IF spiking neural network and integrates the event stream of each sample into 15 frames for goodness calculation and weight update.

As a result, when the learning rate is 0.00003 and the threshold is 0.88, the test error rate of the model at the first epoch is 41.5% (±2%). The conclusions of the previous experiments are still valid in this experiment: adjusting the magnitude of the threshold will have a significant impact on the final performance (test error rate); introducing layer collaboration will have a significant positive impact on the final performance (test error rate).

In addition, this experiment has a very important finding: when training on a non-Poisson encoded data set, the spiking neurons of the hidden layer will hardly be activated. Adjusting the sample integration method, changing the learning rate and changing the tau value will not

reverse this phenomenon.

The findings of this experiment lead to two conclusions:

1. The Spiking Forward-Forward algorithm exhibits learning and generalization capabilities in multiple spike morphology data sets.

2. The Spiking Forward-Forward algorithm has the problem of insufficient learning ability of the hidden layer in some specific cases.

3.5 Spiking-Heidelberg-Digits-dataset experiment

This experiment is part of the robustness research of Spiking Forward-Forward, and it is also a further study of the problems found in the previous experiment.

This experiment uses a 700*1000*1000 LIF spiking neural network to integrate the event stream of each sample into 10 frames for goodness calculation and weight update.

As a result, when the learning rate is 0.00003, the threshold is 0.11, and the tau value is 2.0, the test error rate of the model at the 21st epoch is 57.1% (±2%). The conclusions of the previous experiments are still valid in this experiment.

In this experiment, there is still the problem that the spiking neurons in the hidden layer are difficult to activate. However, if the input data to the hidden layer is multiplied by a coefficient, this phenomenon is reversed and provides a certain positive impact on the final performance of the model. A potential inference for this phenomenon is that when certain data sets are used, the spiking firing of the spiking neural network is too sparse, causing the hidden layer to not capture enough high-level features.

For comparison, the error rate of traditional SNN using the surrogate gradient scheme is 51.9% (±1.6%) [8].

3.6 Decentralized Training Experiment

This experiment is based on the Poisson-Spike-MNIST experiment and is designed to test the flexibility of the Spiking Forward-Forward algorithm.

Since the training of the Spiking Forward-Forward algorithm is completed independently by each layer, training different layers in any order should result in similar performance to layer-by-layer training. Furthermore, this feature can be extended to: training multiple layers simultaneously on different hardware should also provide similar performance to training on the same hardware.

In the test, in order to simulate these two situations, two spiking neural networks using the same data set, with the same shape and hyperparameters were deployed. The first spiking neural network only trains the second layer, and the second spiking neural network only trains the first layer. Finally, a new spiking neural network is deployed, which is built only by referencing the layers that were trained.

As a result, this experiment yields an error rate of 12.0%, which is close to the performance of the original model with the same random seed (9.95%).

Based on the above experiments, it can be concluded that: despite the gap between the back-propagation-based algorithms and Spiking Forward-Forward, the Spiking Forward-

Forward algorithm is able to train and generalize spiking neural networks on multiple spike morphology datasets.

4 Discussion

Algorithm Feature

Spiking Forward-Forward is simultaneously sensitive to a variety of hyperparameters that only exist within this algorithm. For example, in supervised learning, Spiking Forward-Forward requires careful selection of the order of magnitude of the threshold. For a certain layer in the spiking neural network, the training of this layer can only converge if the threshold of the layer is of the same order of magnitude as the non-updated goodness. At the same time, the learning rate suitable for Spiking Forward-Forward is also much lower than that of conventional algorithms. Therefore, debugging the hyperparameters of Spiking Forward-Forward layer by layer will lead to better performance. Furthermore, after integrating the STDP module, whether to enable the STDP feature during training has also become one of the hyperparameters. In the three experiments in this paper using Poisson encoders, the STDP feature was turned on. In addition, schemes that only use STDP and abandon any gradient calculation still have potential and deserve further research.

On the other hand, it can be found through experiments that the timing characteristics of the spiking neural network shorten the epochs required for training. Under some specific circumstances, Spiking Forward-Forward can achieve optimal performance in the first epoch. This phenomenon essentially comes from the fact that the spiking neural network decomposes a sample into frames (or event streams) for learning, improving data utilization.

Since backpropagation is no longer used, Spiking Forward-Forward provides flexibility in training and deployment. In addition to decentralized parallel training, Spiking Forward-Forward also supports customizing the number of epochs for different layers and the training order of the layers. This means that Spiking Forward-Forward can train more on specific layers as needed, or stop training on specific layers before overfitting.

Algorithm Limitation

It can be found that the generalization ability of the Spiking Forward-Forward algorithm is still limited when facing complex data sets. This limitation mainly comes from two aspects:

In supervised learning, Spiking Forward-Forward uses a fully connected structure, which limits its learning capacity for complex images.

Spiking Forward-Forward deletes the normalization module in the traditional Forward-Forward and replaces it with the unique binary impulse firing process of the spiking neural network. Vectors composed of binary data are more abstract than traditional vectors. This makes it more difficult for the hidden layer to learn high-level features.

5 Conclusion

This project focuses on implementing the Spiking Forward-Forward algorithm, a combination of spiking neural network and Forward-Forward. In an era when spiking neural networks are moving towards practical applications, the potential of this combination has existed since the discovery of completely new algorithms liberated from backpropagation.

This project has completed the implementation of the Spiking Forward-Forward algorithm, and added original designs and improvements to the algorithm based on a series of experiments using multiple data sets. More importantly, through these experiments, this project demonstrates the effectiveness and robustness of Spiking Forward-Forward, proving the practicability of the algorithm. More importantly, the Spiking Forward-Forward algorithm not only provides a new perspective for spiking neural networks, but also provides potential applications for the mortal computation idea behind the algorithm. On the other hand, Spiking Forward-Forward also has limitations, such as insufficient generalization performance and too complex hyperparameter combinations. These problems are waiting to be explained, improved and broken through.

References

[1] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Networks*, vol. 10, no. 9, pp. 1659–1671, 1997. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608097000117

[2] E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks," *IEEE Signal Processing Magazine*, vol. 36, no. 6, pp. 51–63, 2019.

[3] G. Hinton, "The forward-forward algorithm: Some preliminary investigations," *arXiv preprint arXiv:2212.13345*, 2022.

[4] A. Kugele, T. Pfeil, M. Pfeiffer, and E. Chicca, "Efficient processing of spatio-temporal data streams with spiking neural networks," *Frontiers in Neuroscience*, vol. 14, p. 439, 2020.

[5] N. Caporale and Y. Dan, "Spike timing–dependent plasticity: a hebbian learning rule," *Annu. Rev. Neurosci.*, vol. 31, pp. 25–46, 2008.

[6] G. Lorberbom, I. Gat, Y. Adi, A. Schwing, and T. Hazan, "Layer collaboration in the forward-forward algorithm," *arXiv preprint arXiv:2305.12393*, 2023.

[7] Y. Liu, Y. Chen, W. Ye, and Y. Gui, "Fpga-nhap: A general fpga-based neuromorphic hardware acceleration platform with high speed and low power," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 69, no. 6, pp. 2553–2566, 2022.

[8] B. Cramer, Y. Stradmann, J. Schemmel, and F. Zenke, "The heidelberg spiking data sets for the systematic evaluation of spiking neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 7, pp. 2744–2757, 2020.