Generalised Training of Spiking Neural Networks

Anonymous Author(s)
Affiliation
Address
email

Abstract

Spiking neural networks (SNNs) can be trained by first training an equivalent ANN and then transferring the trained weights to the SNN, but there are two significant problems to be solved. First, an accurate activation function is needed to model the neural dynamics of spiking neurons, and our previously proposed activation function, Noisy Softplus, has shown to be a good match to the response activity of Leaky Integrate-and-Fire (LIF) neurons. The second problem is mapping the abstract numerical values of the ANN to concrete physical units in the SNN, such as current and firing rate. In this paper, we introduce the parametric activation function (PAF) to tackle the second problem. With these problems solved, SNNs can be trained exactly the same way as ANNs, and the trained weights can be used directly in the spiking version of the same network without any conversion. More importantly, the PAF can be generalised to activation functions other than Noisy Softplus, such as the Rectified Linear Unit (ReLU).Therefore, SNN training can be simplified as: estimate the parameters for PAF according to biological configurations of LIF neurons, and use parametric ReLU as the activation function when training. In addition, we propose a fine tuning method as an option which helps the trained network to match the SNN more closely. Based on this generalised training method, we achieve the best SNN accuracy oN the MNIST task using LIF neurons, 98.85%, on a 6-layer spiking convolutional neural network (ConvNet).

1 Introduction

Advances in computing power and deep learning have benefited computers with a rapid growing performance in cognitive tasks, such as recognising objects [1] and playing GO [2]). These tasks were once dominated by human intelligence and solved by biological neurons in the brain. However, humans and many other animals still win against computers in practical tasks and outperform in terms of size and energy cost over several orders of magnitude. For instance, AlphaGO [2] consumed 1 MW of power on its 1920 CPUs and 280 GPUs when playing the game with one of the best human players whose brain only consumed about 20 W. Although we are still far from understanding the brain thoroughly, it is believed that the performance gap between computation in the biological nervous system and in a computer lies in the fundamental computing units and the way they act. Computers employ Boolean logic and deterministic digital operations based usually on synchronous clocks while nervous systems employ parallel-distributed, event-driven, stochastic unreliable components [3]. The impressive disparities in cognitive capabilities and energy consumption drives the research into biologically-plausible spiking neurons.

A regular artificial neuron (Figure 1(a)) composes a weighted summation of input data, $\sum x_i w_i$, and an activation function, f, applying on the sum. Usually, a bias is included in the weighted summation which can be seen as an extra input $x_b = 1$ with its weight set to b. However, in this paper we exclude biases for both artificial and spiking neurons to simplify neural models and to reduce parameters. Meanwhile the inputs of a spiking neuron (Figure 1(b)) are pre-synaptic spike

Submitted to 31st Conference on Neural Information Processing Systems (NIPS 2017). Do not distribute.

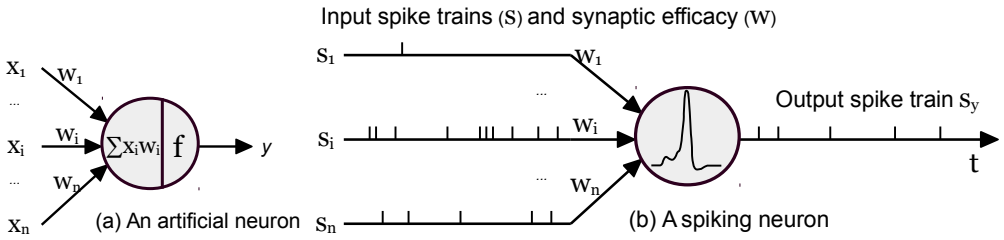


Figure 1: Comparisons of processing mechanisms of an artificial and a spiking neuron. (a) An artificial neuron takes numerical values of vector **x** as input, works as a weighted summation followed by an activation function f. (b) Spike trains flow into a spiking neuron as input stimuli, trigger linearly summed-up PSPs through synapses with different synaptic efficacy **w**, and post-synaptic neuron generates output spikes when membrane potential reaches some threshold.

trains, which create Post-synaptic Potentials (PSPs) and trigger spikes as outcomes when the neuron's membrane potential reaches some threshold. Neural dynamics of the membrane potentials, PSPs, and spike trains are all time dependent, while neurons of ANNs, e.g. sigmoid units, only cope with abstract numerical values representing spiking rate, without timing information These fundamental differences on input/output representation and neural computation raise the research problem of how to operate and train biologically-plausible SNNs to be competent in cognitive tasks just as ANNs.

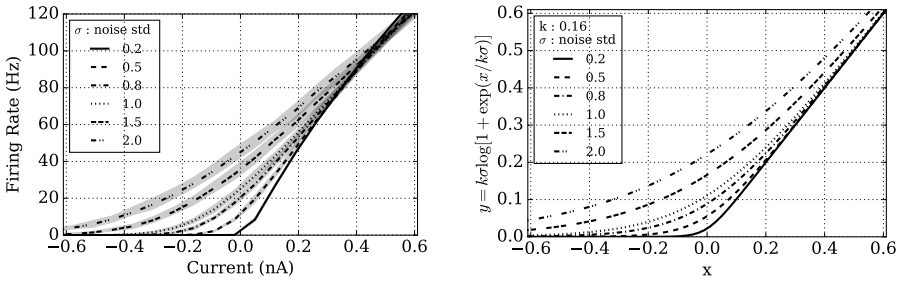
An intuitive idea is to train SNNs on an equivalent ANN and then transfer the trained weights to the SNN. Jug et al. [4] first proposed the use of the Siegert formula [5] as the activation function in training Deep Belief Networks, which maps incoming currents driven by Poisson spike trains to the response firing rate of an LIF neuron. The activation function is in actual physical units, thus the trained weights can be transferred directly into SNNs. However, most importantly, the Siegert formula is inaccurate taking no notice of the coloured noise generated by the synaptic time constant τ_{syn} of spikes arrivals, since the current noise is only a white noise when $\tau_{syn} \to 0$ Moreover, the high complexity of the Siegert function causes much longer training time and more energy, let alone the high-cost computation on its derivative to obtain the error gradient.

A similar activation function of Soft LIF [6] was introduced to simplify the computation complexity. However, the model ignored the current noise introduced by input spikes, assuming static current influx into neurons. Therefore the training required additional noise on the response firing rate and on the training data, thus included hyperparameters in the model. What's more, the activation function devoted the modelling accuracy for computational simplicity.

Therefore, here comes the first problem of accurately model the neural response activity of LIF neurons with abstract activation functions. We call these activation functions 'abstract' referring to the ones without physical units which used in the ANNs, and select them for LIF modelling due to their simplicity and generalised training in ANNs. Noisy softplus [7], was proved to be a close match to the response activity of LIF neurons by including noise as the second factor in the activation function and taking account of coloured noise driven by τ_{syn}.

Then the second problem appeared to map the abstract activation functions to actual physical units: current in nA and firing rates in Hz. Instead of solving the problem, the alternative way of converting ANN-trained weights to be fitted in SNNs [8, 9] was successfully applied on less biologically-realistic and simplified integrate-and-fire (IF) neurons. Normalising the ReLU-trained weights for use on simplified IF neurons was relatively straightforward, so this method keeps the state-of-the-art performance. However, this paper aims to address the second problem of mapping activation functions to actual response firing activity of biologically-plausible LIF neurons, thus to complete the generalised SNN training mechanism.

This paper will start with a brief review on modelling the LIF response function with Noisy Softplus in Section 2, and introduce the PAF in Section 3 to address the second problem mentioned above and complete the generalised SNN training method. In Section 4 we will demonstrate the training of a spiking ConvNet, and compare the proposed method to existing training algorithms.



(a) Noisy Softplus (b) Noisy Softplus in 3D

Figure 2: Noisy Softplus models the LIF response function. (a) Actual firing rates measured by simulations on an LIF neuron driven by different input currents and discrete noise levels. Bold lines show the average and the grey colour fills the range between the minimum and the maximum. (b) Noisy Softplus activates the input x according to different noise levels where $k = 0.16$.

2 Background

To model the response function of LIF neurons, see Figure 2(a), whose output firing rates are determined by the mean and variance of the noisy input currents, we proposed the Noisy Softplus:

$$y = f_{ns}(x, \sigma) = k\sigma \log[1 + \exp(\frac{x}{k\sigma})] \,, \tag{1}$$

where x and σ refer to the mean and standard deviation of the input current, y indicates the intensity of the output firing rate, and k, determined by the biological configurations on the LIF neurons (listed in Table 1), controls the shape of the curves. Note that the novel activation function we proposed contains two parameters, the mean current and its noise, which can be estimated by:

$$x = \tau_{syn} \sum_i w_i \lambda_i \;,\; \sigma^2 = \frac{1}{2} \tau_{syn} \sum_i w_i^2 \lambda_i \;, \tag{2}$$

where λ_i indicates the firing rate of an input spike train. Figure 2(b) shows the activation function in curve sets corresponding to different discrete noise levels which mimics the actual responding activities of LIF neurons. The derivative of the Noisy Softplus is the logistic function scaled by $k\sigma$:

$$\frac{\partial f_{ns}(x, \sigma)}{\partial x} = \frac{1}{1 + exp(-\frac{x}{k\sigma})} \;, \tag{3}$$

which could be easily applied to back propagation in any network training.

Table 1: Default parameter settings for the current-based LIF neurons used through this paper, for PyNN [10] simulations.

cm	tau_m	tau_refrac	v_reset	v_rest	v_thresh	i_offset
0.25 nF	20.0 ms	1.0 ms	-65.0 mV	-65.0 mV	-50.0 mV	0.1 nA

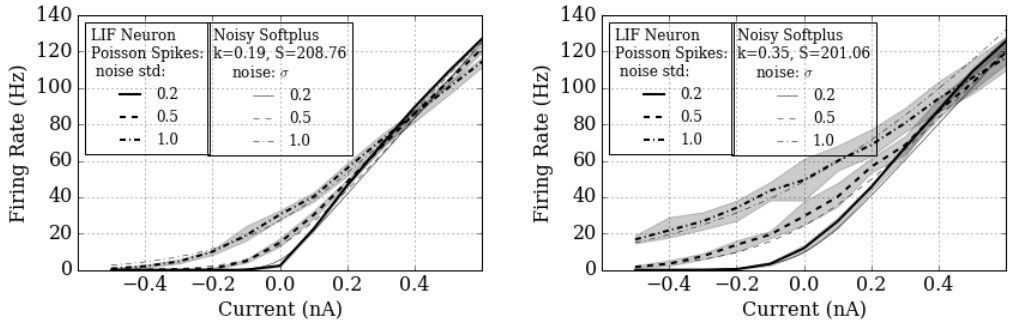
3 Methods

3.1 Mapping Noisy Softplus to Concrete Physical Units

The inputs of the Noisy Softplus, x and σ, are obtained from physical variables as stated in Equation 2, thus already in physical unit: nA. Therefore, linearly scaling up the activation function by a factor $S\ Hz$ can approximate the output firing rate λ_{out} of an LIF neuron in Hz:

$$\lambda_{out} \simeq f_{ns}(x, \sigma) \times S = k\sigma \log[1 + \exp(\frac{x}{k\sigma})] \times S \,. \tag{4}$$

Suitable calibrations of k and S in Equation 4 enables Noisy Softplus to closely match the practical response firing rates of LIF neurons given various biological parameters. The parameter pair of (k, S)

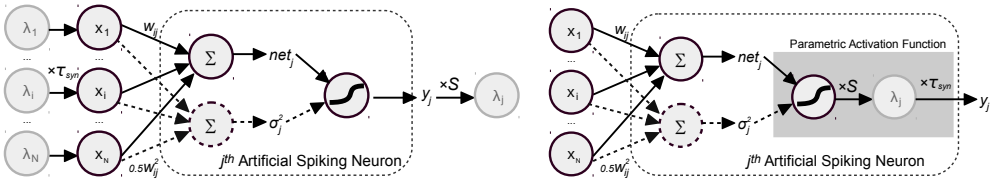


(a) τ_{syn}=1 ms (b) τ_{syn}=10 ms

Figure 3: Noisy Softplus fits to the actual response firing rates of LIF neurons in concrete physical units. Recorded response firing rate of an LIF neuron driven by synaptic current with two synaptic time constants: (a) τ_{syn}=10 ms and (b) τ_{syn}=10 ms. Averaged firing rates of simulation trails are shown in bold lines, and the grey colour fills the range between the minimum to maximum of the firing rates. The thin lines are the scaled Noisy Softplus.

is curve-fitted with the triple data points of $(\lambda_{out}, x, \sigma)$ and the calibration currently is conducted by linear least squares regression. The output firing rate λ_{out} is measured from SNN simulations where an LIF neuron is driven by synaptic input current of Poisson spike trains. Figure 3 shows two calibration results that the parameters were fitted to $(k, S) = (0.19, 208.76)$ when the synaptic constant is set to $\tau_{syn} = 1$ ms and was fitted to $(k, S) = (0.35, 201.06)$ when $\tau_{syn} = 10$ ms.

3.2 Parametric Activation Functions (PAFs)



(a) Noisy Softplus (b) Parametric Noisy Softplus

Figure 4: The PAF links the firing activity of a spiking neuron to the numerical value of ANNs.

Neurons in ANNs take inputs from their previous layer, and feed the weighted sum of their input, $net_j = \sum_i w_{ij} x_i$, to the activation function. The transformed signal then forms the output of an artificial neuron, which can be denoted as $y_j = f(net_j)$, see Figure 1(a). However, a spiking neuron modelled by Noisy Softplus takes firing rate as input/output thus Equation 2 can be written as:

$$net_j = \sum_i w_{ij}(\lambda_i \tau_{syn}) \ , \ \sigma_j^2 = \sum_i (\frac{1}{2} w_{ij}^2)(\lambda_i \tau_{syn}) \ , \tag{5}$$

and x_i can be seen as $\lambda_i \tau_{syn}$, see Figure 4(a).

If we move the left end process of $\times \tau_{syn}$ to the right end after λ_j, Figure 4(b) forms the same neuron model and structure as a typical neuron in ANNs, See Figure 1(a), that neurons take x as input and outputs y. The only difference lies in the activation function where the artificial spiking neuron takes PAF, which is a simple linearly-scaled activation function with the scaling parameter S and the synaptic time constant, τ_{syn}:

$$y = PAF(x) = f(x) \times S \times \tau_{syn} \ , \tag{6}$$

and its derivative function which is used when back propagates is:

$$\frac{\partial y}{\partial x} = f'(x) \times S \times \tau_{syn} \ . \tag{7}$$

Excitingly, PAF not only allows Noisy Softplus to model spiking LIF neurons onn ANNs, but also can be generalised to other activation functions. Note that the calculation of noise level is not necessary for other activation functions, for example, it can be set to a constant for Softplus or 0 for ReLU.

3.3 Generalised SNN Training

Most excitingly, SNN training then can be simplified as: calibrate the parameters of (k, S) for Noisy Softplus which models actual response firing rates of LIF neurons, and use S and τ_{syn} for PAF-ReLU as the activation function when training. Note that, (k, S) are dependant on the biological configurations of an LIF neuron, but independent from core activation functions used in PAF.

This generalised SNN training allowed using widely-used activation functions in ANNs which are of low complexity and their corresponding derivative functions can be directly used for back propagation. Especially, ReLU, the simplest and most effective activation function may improve the training performance of SNNs. Ideally, the method can be applied for any feedforward network using ReLU-like activation functions, including deep architecture. Most significantly, the ANN-trained weights are ready for use to transfer to SNNs without any conversion, and the output firing rate of a spiking neuron can be obtained in the ANN simulation thus to estimate the power use in Neuromorphic systems (hardware SNN simulators).

3.4 Fine Tuning

There are two aspects to the fine tuning which makes the ANN closer to SNNs: Firstly, using Noisy Softplus activation functions in a whole trained network operates every single neuron running in a similar noise level as in SNNs, thus the weights trained by other activation functions will be tuned to fit closer to SNNs. Secondly, the output firing rate of any LIF neuron is greater than zero as long as noise exists in their synaptic input. Thus adding up a small offset on the labels directs the model to approximate to practical SNNs.

The labels of data are always converted to binary values for ANN training. This enlarges the disparities between the correct recognition label and the rest to train the network for better classification capability. Consequently, we can train the network with any activation function and then fine-tune it with Noisy Softplus to take account of both accuracy and practical network activities of SNNs. However, we add a small number, for example 0.01, to all the binary values of the data labels. Doing so helps the training to loosen the strict objective function to predict exact labels with binary values. Instead, it allows a small offset to the objective. An alternative method is to use Softmax function at the top layer, which aims to map real vectors to the range of $(0, 1)$ that add up to 1. However, without a limit on the input of Softmax, it will be easy to reach or even exceed the highest firing rate of a spiking neuron. The result of fine tuning on a Convnet will be demonstrated in subsection 4.2.

4 Results

A convolutional network model was trained on MNIST, a popular database in neuromorphic vision, using the ANN-trained SNN method stated above. The architecture contains 28×28 input units, followed by two convolutional layers 6c5-2s-12c5-2s, and 10 output neurons fully connected to the last pooling layer to represent the classified digit.

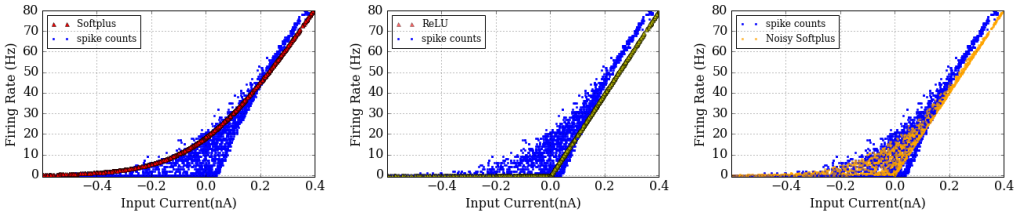
The training only employed Noisy Softplus units that all the convolution, average sampling, and the fully-connected neurons use Noisy Softplus function with no bias. The parameters of the activation function were calibrated as, $(k = 0.30, S = 201)$, for LIF neurons. The input images were scaled by 100 Hz to present the firing rates of input spikes. The weights were updated using a decaying learning rate, 50 images per batch and 20 epochs. The ANN-trained weights were then directly applied in the corresponding convolutional SNN without any conversion for recognition tasks.

4.1 Neural Activity

To validate how well the Noisy Softplus activation fits to the response firing rate of LIF neurons in a real application, we simulated the model on NEST using the Poisson MNIST dataset [11] and the neurons of a convolutional map were observed.

A small test of ten MNIST digits presented in Poisson spike trains for 1 s each. A trained 5×5 kernel was convolved with these input digits, and the convolved output of the feature map, the output firing rate was recorded during a real-time SNN simulation on NEST, and compared to the modelled activations of Equation 6 in ANNs.

With three PAFs of ReLU, Softplus and Noisy Softplus, we compare the output to the recorded SNN simulations. The experiment took the sequence of 10 digits to the same kernel and the estimated spike counts using Noisy Softplus fit to the real recorded firing rate much more accurately than ReLU and Softplus, see 5. The Euclidean distance, $\sqrt{\sum_j (y_j/\tau_{syn} - \lambda_j)}$, between the spike counts and the predicted firing rates by Noisy Softplus, ReLU and Softplus was 184.57, 361.64 and 1102.76 respectively. We manually selected a static noise level of 0.45 for Softplus, whose estimated firing rates located roughly on the top slope of the real response activity. This resulted in longer Euclidean distance than using ReLU, since most of the input noisy currents were of relatively low noise level in this experiment. Hence, the firing rate driven by lower noise level is closer to ReLU curve than Softplus.



(a) Softplus (b) ReLU (c) Noisy Softplus

Figure 5: Noisy Softplus fits to the neural response firing rate in an SNN simulation. The recorded firing rate of the same kernel convolved with 10 images in SNN simulation, comparing to the prediction of activations of Softplus, ReLU, and Noisy Softplus.

The SNN successfully classified the digits where the correct label neuron fired the most. We trained the network with binary labels on the output layer, thus the expected firing rate of correct classification was $1/\tau_{syn} = 200$ Hz according to Equation 4. The firing rates of the recognition test fell to the valid range around 0 to 200 Hz. This shows another advantage of the proposed ANN-trained method that we can constrain the expected firing rate of the top layer, thus preventing SNN from exceeding its maximum firing rate, for example 1000 Hz when time resolution of SNN simulation set to 1 ms.

4.2 Recognition Performance

Here we focus on the recognition performance of the proposed ANN-trained SNN method. Before looking into the recognition results, it is significant to see the learning capability of the proposed activation function, Noisy Softplus. We compared the training using ReLU, Softplus, and Noisy Softplus by their loss during training averaged over 3 trials, see Figure 6. ReLU learned fastest with the lowest loss, thanks to its steepest derivative. In comparison, Softplus accumulated spontaneous firing rates layer by layer and its derivative may experience vanishing gradients during back propagation, which result in a more difficult training. Noisy Softplus performance lay between these two in terms of loss and learning speed. However, the loss stabilised fastest, which means a possible shorter training time.

The recognition test took the whole testing dataset of MNIST which contains $10,000$ images. At first, all trained models were tested on the same artificial neurons as used for training in ANNs, and these experiments were called 'DNN' test since the network had a deep structure (6 layers). Subsequently, the trained weights were directly applied to SNN without any transformation, and these 'SNN' experiments tested their recognition performance on the NEST simulator. The LIF neurons had the same parameters as in training. The input images were converted to Poisson spike trains and presented for 1 s each. The output neuron which fired the most indicated the classification of an input image. Moreover, a 'Fine tuning' test took the trained model for fine tuning, and the tuned weights were tested on the same SNN environment. The tuning only ran for one epoch, 5% cost of the ANN training (20 epochs), using Noisy Softplus neurons with labels shifted for $+0.01$.

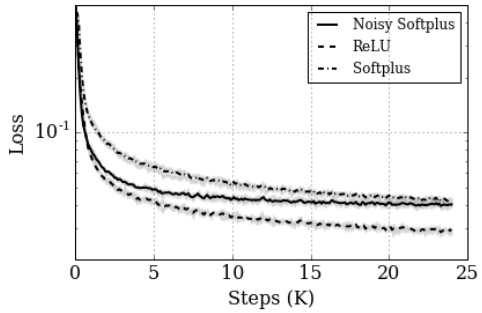


Figure 6: Comparisons of Loss during training using Noisy Softplus, ReLU and Softplus activation functions. Bold lines show the average of three training trials, and the grey colour illustrates the range between the minimum and the maximum values of the trials.

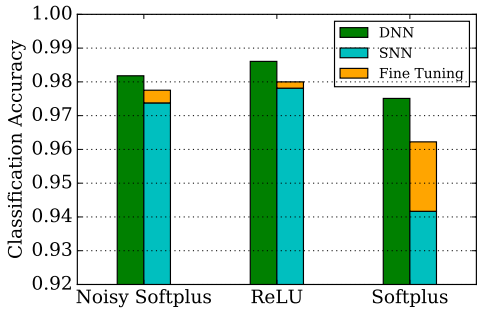


Figure 7: Classification accuracy compared among trained weights of Noisy Softplus, ReLU, Softplus on DNN, SNN and fine-tuned SNN.

The classification errors for the tests are investigated in the averaged classification accuracy, shown in Figure 7. From DNN to SNN, the classification accuracy declines by 0.80%, 0.79% and 3.12% on average for Noisy softplus, ReLU and Softplus The accuracy loss was caused by the mismatch between the activations and the practical response firing rates, see example in Figure 5, and the strict binary labels for Noisy Softplus and Softplus activations. Fortunately, the problem is alleviated by fine tuning which increased the classification accuracy by 0.38%, 0.19% and 2.06%, and resulted in the total loss of 0.43%, 0.61%, and 1.06% respectively. The improvement of ReLU is not as great as the others, because there is no problem of strict labels during training. Softplus benefits the most from fine tuning, since not only the huge mismatch of response firing rate is greatly corrected, but also the offset on the labels helps the network to fit SNNs.

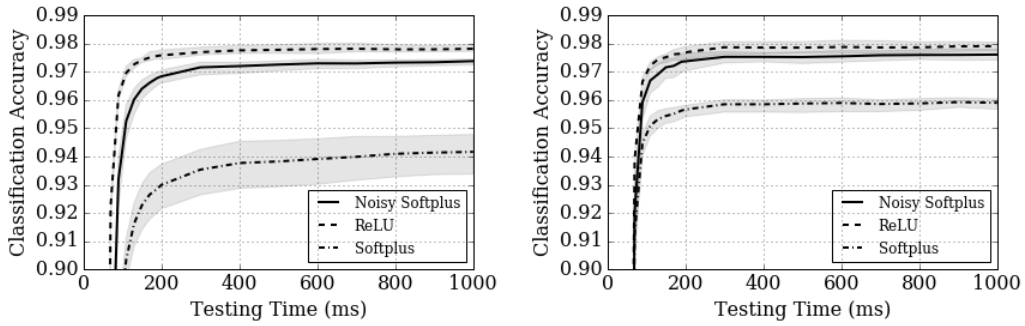
The most efficient training in terms of both classification accuracy and algorithm complexity, takes ReLU for ANN training and Noisy Softplus for fine tuning. Softplus does not exhibit better classification capability and more importantly the manual selected static noise level hugely influences the mismatch between the predicted firing rates and the real data. Although Noisy Softplus shows the least classification drop from ANNs to SNNs, the training performance is still worse than ReLU.

The best classification accuracy achieved by SNN was 98.85%, a 0.20% drop from ANN test (99.05%), which was trained with ReLU and fine-tuned by Noisy Softplus. It is useful to compare with existing SNN training methods in Table where we order them on the computation complexity (descending).

Table 2: SNN training methods comparisons.

complexity	activation function	conversion	biologically-plausible	extra tricks
1 [4]	Siegert	No	Yes	No
2 [6]	Soft LIF	No	Yes	Noisy input & AF
3 [9]	ReLU	Normalisation	No	Dropout
4 This Paper	PAF-ReLU	No	Yes	No

As it is a major concern in neuromorphic vision, the recognition performance over short response times is also estimated in Figure 8. After fine tuning, Softplus significantly reduced the mismatch since the randomness among the three trials shrinks to a range similar to other experiments. More obviously, fine tuning improved its classification accuracy and the response latency. Notice that all of the networks trained by three different activation functions showed a very similar stabilisation curve against time, which means they all reached an accuracy close to their best by only taking 300 ms of test.



(a) Before fine tuning

(b) After fine tuning.

Figure 8: The classification accuracy of 3 trials (averaged in bold lines, grey shading shows the range between minimum to maximum) over short response times, with (a) trained weights before fine tuning, and (b) after fine tuning.

4.3 Power Consumption

Noisy Softplus can easily be used for energy cost estimation for SNNs. For a single neuron, the energy consumption of the synaptic events it triggers is:

$$E_j = \lambda_j N_j T E_{syn} = \frac{y_j N_j T E_{syn}}{\tau_{syn}} \quad , \tag{8}$$

where λ_j is the output firing rate, N_j is the number of post-synaptic neurons it connects to, T is the testing time, and E_{syn} is the energy cost for a synaptic event of some specific neuromorphic hardware, for example, about 8 nJ on SpiNNaker [12]. Thus to estimate the whole network, we can sum up all the synaptic events of all the neurons:

$$\sum_j E_j = \frac{T E_{syn}}{\tau_{syn}} \sum_j y_j N_j. \tag{9}$$

Thus, it may cost SpiNNaker 0.064 W, 192 J running for $3,000$ s with synaptic events of $8 \times 10^6/s$ to classify $10,000$ images (300 ms each) with an accuracy of 98.02%. The best performance reported using the larger network may cost SpiNNaker 0.43 W operating synaptic event rate at $5.34 \times 10^7/s$, consume 4271.6 J to classify all the images for 1 s each.

5 Conclusion and Future Work

Most significantly,we proposed a generalised SNN training method to train an equivalent ANN and transfer the trained weights back to the SNN. This training procedure is simple as two stages: first, estimate PAF parameters using Noisy Softplus, and second, use PAF instead of conventional activation functions in ANN training. can be generalised to activation units other than Noisy Softplus. The training of an SNN model is exactly the same as ANN training, and the trained weights can be directly used in SNN without any transformation. This method requires least computation complexity but performs most effectively among existing training algorithms.

In terms of classification/recognition accuracy, the performance of ANN-trained SNNs is nearly equivalent as ANNs, and the performance loss can be partially solved by fine tuning. The best classification accuracy of 98.85% using LIF neurons in PyNN simulation outperforms state-of-the-art SNN models of LIF neurons and is very close to the result using IF neurons [9].

The current limitation prohibiting this off-line SNN training method from wide use lies on the lack of supporting tools. This requires developing a supporting tool which enables SNN training in popular deep learning platforms, and the other automation tool that reads platform-dependant trained weights into PyNN [10] language. The other issue is the parameter calibration on the scaling factor of the PAF, thus numerical analysis is considered for future work to express the factors with biological parameters of a LIF neuron.

Interesting applications have started with collaborations on speech recognition of cochlea generated spikes which has achieved a promising accuracy at the initial test-idea stage. A further goal is to implement deep networks fit for ImageNet [1] tasks, which will also requires modelling various structures of deep learning, for example recurrent neural networks.

Acknowledgments

To be added after reviewing.

References

[1] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: a large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255, 2009.

[2] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[3] Giacomo Indiveri, Elisabetta Chicca, and Rodney J Douglas. Artificial cognitive systems: from VLSI networks of spiking neurons to neuromorphic cognition. *Cognitive Computation*, 1(2):119–127, 2009.

[4] F. Jug, J. Lengler, C. Krautz, and A. Steger. Spiking networks and their rate-based equivalents: does it make sense to use Siegert neurons? In *Swiss Soc. for Neuroscience*, 2012.

[5] Arnold JF Siegert. On the first passage time probability problem. *Physical Review*, 81(4):617, 1951.

[6] Eric Hunsberger and Chris Eliasmith. Spiking deep networks with lif neurons. *arXiv preprint*, 2015.

[7] Qian Liu and Steve Furber. Noisy softplus: A biology inspired activation function. In *International Conference on Neural Information Processing*, pages 405–412. Springer, 2016.

[8] Yongqiang Cao, Yang Chen, and Deepak Khosla. Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision*, 113(1):54–66, 2015.

[9] 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 *Neural Networks (IJCNN), 2015 International Joint Conference on*, pages 1–8. IEEE, 2015.

[10] Andrew P Davison, Daniel Brüderle, Jochen Eppler, Jens Kremkow, Eilif Muller, Dejan Pecevski, Laurent Perrinet, and Pierre Yger. PyNN: a common interface for neuronal network simulators. *Frontiers in neuroinformatics*, 2, 2008.

[11] Qian Liu, Evangelos Pineda-García Garibaldia nd Stromatias, Teresa Serrano-Gotarredona, and Steve Furber. Benchmarking spike-based visual recognition: A dataset and evaluation. *Frontiers in Neuroscience*, 10:496, 2016. http://journal.frontiersin.org/article/10.3389/fnins.2016.00496.

[12] Evangelos Stromatias, Francesco Galluppi, Cameron Patterson, and Steve Furber. Power analysis of large-scale, real-time neural networks on SpiNNaker. In *Neural Networks (IJCNN), The 2013 International Joint Conference on*, pages 1–8, 2013.