Direct Training of Spiking Transformers

Anosh Billimoria[1] and Katharina Bendig[2]

[1] billimor@rptu.de
[2] katharina.bendig@dfki.de

Abstract. As the focus on energy efficiency grows in research, the race of sustainable solutions for the planet while minimizing performance compromise emerges as beneficial. The rise of Spiking Neural Networks (SNNs) is subject to their alignment with this goal over Artificial Neural Networks (ANNs). The Transformer architecture, renowned for its versatility across diverse AI tasks such as text generation, object recognition and music generation with commendable performance. A combination of the two results in architectures that are challenging to train up to the same standard as their counterparts. To address this, numerous approximation techniques have been suggested. This study aims to execute, compare, and present several metrics, facilitating an exploration of insights into the direct training approach of Spiking Transformers.

Keywords: Spiking Neural Networks, Transformers, Survey

1 Introduction

Similar to Neural Networks that are computational models inspired by the human brain's interconnected neuron structure, spiking neural networks employ spatio-temporal dynamics to replicate neural actions and utilize binary spike signals for inter-neuron communication. They harness the advantages of an event-driven processing paradigm, where computations are triggered solely upon the arrival of spike events. This event-driven and spatial-temporal manner makes the SNNs very efficient and good at handling temporal signals, thus receiving a lot of research attention, especially recently [38,30,22,18]. The input data is differently interpreted in training procedures such that ANN have static inputs but SNNs run through binary data as a function of time. This single dynamic pass over the data is called a timestep and there can be several during an epoch. Two popular strategies exist for training SNNs. One involves the conversion of a pre-trained ANN into an SNNs model, typically necessitating a considerable number of time steps, as indicated by prior research [28]. The alternative approach entails training SNNs directly using gradient descent, thereby bypassing the dependency on pre-trained ANNs and reducing the need for an excessive number of time steps.
Conversely, transformers, as introduced by [31], are founded upon the notion of multi-head self-attention. Originally conceived as an advancement over sequence-to-sequence models for automated translation by [29], transformers garnered

their initial acclaim in NLP tasks such as machine translation [31] and language modeling[23]. Subsequently also demonstrated state-of-the-art prowess in Computer Vision tasks such as image classification [8], object detection [20] and image generation [14]. Other areas as well such as audio processing [4,36]. The idea of seamlessly merging the effectiveness of Transformers with the energy-efficient traits of SNNs suggests a compelling and intriguing proposition. To effectively understand how can this be useful, we introduce a set of new metrics called spiking metrics. Which can only be studied under the SNNs domain and go deeper into the way the network learns the given inputs and makes predictions. Various encoding techniques, neuron models, datasets and surrogate functions are compared with each other to experimentally establish a correlation between their learning patterns.

2 Background

2.1 Spiking Neural Networks

SNNs [21] are a biologically more accurate implementation unlike traditional ANNs that do not emulate the impulse cycle of organic neurons.

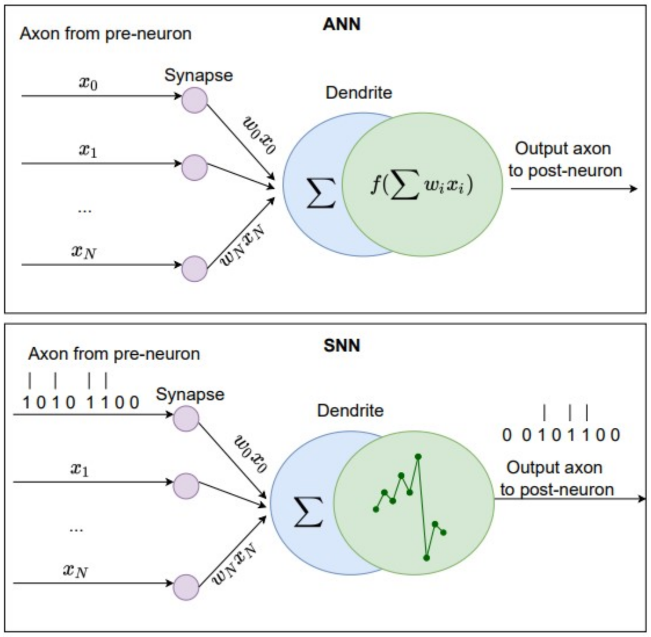

Fig. 1: Comparison between Artificial and Spiking neuron. [35]

As seen in Figure 1 input spikes $x_0 \dots x_N$ enter the spiking neurons from the synapse and are combined with weight parameters $w_0 \dots w_N$ of the network which are accumulated in the dendrite and accordingly fire as a spike vector from one hidden unit to the next instead of real valued vectors. Within each neuron the accumulation of such spikes results in increased membrane potential which upon crossing a numerical threshold is fired on to the next neuron. This mimics the spiking phenomenon as described in [11]. The most widely used implementation of this phenomenon is the Leaky Integrate-and-Fire (LIF) spiking neurons [6]. It is represented by the formulation:

$$V_l^t = \tau V_l^{t-1} + W_l O_{l-1}^t, V_l^t < V_{th} \tag{1}$$

where V_l^t is the membrane potential at t-th time-step for layer, O_{l-1}^t is the spike output from the previous layer, W_l is the weight matrix at l-th layer, V_{th} is the firing threshold, and τ is a time leak constant for the membrane potential, which is in (0, 1). When τ is 1, the above equation will degenerate to the Integrate-and-Fire (IF) spiking neuron. When the membrane potential exceeds firing threshold indicated by the second part of 1, a spike is fired and neuron is reset to resting potential,

$$O_l^t = \begin{cases} 1, & V_l^t \geq V_{th} \\ 0, & otherwise \end{cases} \tag{2}$$

Following is a list of some distinct properties of SNNs.
Property 1. Unique Spatio temporal behaviour compared to ANNs.
Property 2. Efficient. As the output takes the form of a binary tensor, the multiplicative operations involving activations and weights can be substituted with additive operations, leading to a notable enhancement in energy efficiency. Moreover, in cases where no spike output is produced, the neuron remains inactive. This event-driven mechanism presents an additional avenue for energy conservation.
Property 3. Distillation of information. Evidently, conveying information through the quantization of real-valued membrane potentials into binary output spikes leads to the reduction of information. A single time-step proves insufficient to accommodate an ample amount of information. But this can be improved by using Temporal encoding techniques described later.
Property 4. Non-differentiability. Shown by,

$$\frac{\partial L}{\partial W_l} = \sum_t \left(\frac{\partial L}{\partial O_l^t} \frac{\partial O_l^t}{\partial W_l} + \frac{\partial L}{\partial V_l^{t+1}} \frac{\partial V_l^{t+1}}{\partial V_l^t} \right) \frac{\partial V_l^t}{\partial W_l} \tag{3}$$

where $\frac{\partial V_l^t}{\partial O_l^t}$ is the gradient of the firing function at t-th time step for l -th layer and is 0 everywhere else, while infinity at V_{th}. Consequently, gradient descent updates to infinity or is frozen.

2.2 Training approaches

Widely speaking two major training approaches can be undertaken in context of SNNs.

ANN-to-SNN Conversion Several methods for converting artificial neural networks (ANNs) to spiking neural networks (SNNs) have been developed. These methods primarily focus on converting ReLU neurons to IF neurons. These works [7,27,28] all have the conversion method in common along with some normalizing and residual learning methods. A common hypothesis behind these ANN-to-SNN conversion designs is that the high computational cost in traditional ANNs arises from continuously transmitting real-valued activities between connected nodes, along with subsequent matrix multiplications or convolutions. ANN-to-SNN conversion aims to maintain information transmission and functionality while reducing the costs associated with signal transmission and computation. Binary-valued spikes play a crucial role in this process by reducing the number of bits per transmission, transforming real-valued signals into binary ones, and making signal transmission sparse in time. These conversion methods rely on importing pre-trained parameters like weights and biases from ANNs to SNNs. They have demonstrated comparable results in deep SNNs compared to original ANNs such as VGG and ResNet. Therefore, ANN-to-SNN conversion methods offer a potential solution to the energy-efficiency challenges posed by ANNs during deployment. However, conversion methods impose limitations on the weights of pre-trained ANNs for conversions to work properly, also converted model requires a large number of timesteps to reach the same accuracy.

Direct As described previously, the non-differentiability is a problem limiting in using Back Propagation Through Time (BPTT) techniques. Consequently, an alternative avenue known as the surrogate gradient (SG) based learning method [32] emerges, aiming to identify a differentiable surrogate function to substitute the non-differentiable firing activity during the back-propagation process of spiking neurons. However, a significant problem arises because there is a noticeable mismatch between the gradient of the firing function and the surrogate gradient. This mismatch can result in poorly optimized SNNs and a significant drop in performance. To mitigate this issue, researchers are exploring the development of more carefully crafted surrogate gradients. Recently, the authors of [2] argue that the future of training SNNs lies in the direct training approach. Furthermore, SNNs are more prone to the problem of gradient explosion/vanishing compared to ANNs because many SG methods use tanh-like functions.This SG approach is gaining prominence [24,9] due to its ability to effectively manage temporal data, delivering noteworthy results even with a limited number of time-steps when applied to extensive datasets.

2.3 Spiking Neuron Models

There has been some modifications to the simpler LIF neurons. Generally these adapt to an additional aspect from the biological process or just beneficial mathematically.

Parametric LIF Authors of [10], address the uniformity of membrane-related parameters in spiking neural networks (SNNs). They note that traditional SNNs assume the same membrane time constants for all neurons, despite evidence showing variability in these parameters across brain regions. To address this limitation, they propose a novel training algorithm capable of learning both synaptic weights and membrane time constants. This approach offers several advantages: firstly, it reduces sensitivity to initial values and accelerates learning. Secondly, it acknowledges the significance of diverse time constants in neural dynamics, which play a crucial role in functions such as working memory and learning. The algorithm optimizes the membrane time constants automatically during training, rather than relying on manual hyperparameter tuning. It also enforces shared time constants among neighboring neurons in the same layer while allowing for distinct time constants between layers, enhancing the expressiveness and biological plausibility of the resulting SNNs. From 1 τ can be replaced with $\frac{1}{k(a)}$ as seen in Equation 4 here, experimentally $k(a)$ is chosen as the sigmoid activation function. Directly optimization of τ is not feasible as it can cause numerical instability, thus it is redefined as a well-known bounded function and a becomes the learnable parameter in this case.

$$k(a) = \frac{1}{1 + exp(-a)} \tag{4}$$

KLIF In the paper by [13], a novel k-based leaky Integrate-and-Fire (KLIF) neuron model is introduced to enhance the learning capabilities SNNs. In contrast to the traditional LIF model, KLIF incorporates a dynamic scaling factor (parameter k) that adjusts the slope and width of the surrogate gradient curve during training. Additionally, it integrates a ReLU activation function, enabling selective membrane potential delivery for spike firing and resetting. This approach aligns more closely with biological processes, where neurons control ion concentration differences across their membranes by modulating ion channels. KLIF allows for the regulation of input currents and internal potentials during training, making it a biologically plausible improvement over LIF neurons. This is done by replacing V_l^{t-1} with $F(k)$ where,

$$F(k) = ReLU(k.V_l^{t-1}) \tag{5}$$

2.4 Encodings

Biological nervous systems can convert natural stimulus into "all or nothing" pulses to encode and transmit information. Looking at the various stimulus modalities in nature, each one needs an encoding scheme to represent visual, acoustic, or somatic data from the different senses. Resulting impulses transmit motor commands received by muscles again decode them to execute the action. This biological model suggests, that coding schemes exist which are better or worse depending on their source and application. Most of the data is collected in static format, such that they lack a temporal component, due to the nature of the hardware we use in real world. Such a format is incompatible with SNNs which need data in the form of spike trains and not real-valued vectors. On the other hand, we also have data collected from event cameras which is naturally in binary format, sometimes also called dynamic data as they capture the movements across time, do not need any encoding. Major methods of encoding are described further.

Count Rate [25] Rate coding encodes information by utilizing the instantaneous or averaged rate at which spikes are generated by either an individual neuron or a cluster of neurons. In practice a Poisson distribution is used to randomly generate spike times. The pixel intensities x are used as firing probability and a rate value r is drawn from H distribution. Then converted to binary using x as upper bound.

$$
\begin{aligned}
& r \sim H(x); 0 \leq x \leq 1 \\
& spike = \begin{cases} 1, & r \leq x \\ 0, & otherwise \end{cases}
\end{aligned} \tag{6}
$$

Temporal Temporal encoding relies on the precise timing of spikes to convey information, with more important details represented by earlier spike times. Unlike rate encoding, which uses spike frequency, temporal encoding results in sparser spikes. However, it is vulnerable to input noise or timing variations. For image encoding, each pixel's brightness (ranging from 0 to 255) determines its spike time. Brighter pixels lead to earlier spikes, while pure white pixels (brightness 1 or 255) do not generate spikes as they are deemed information-free in grayscale images.
Latency The basic temporal coding scheme, Time To First Spike (TTFS) [15], encodes information using the time difference (delta t) between stimulus onset and a neuron's first spike. It can be inversely proportional to stimulus amplitude $\delta t = 1/a$ or linear $\delta t = 1 - a$, with larger amplitudes resulting in earlier spikes and lower amplitudes leading to delayed or no spikes.
Weighted Phase [16] A background periodic oscillation function serves as a reference in this approach. The spike's value depends on the phase of the reference function when the spike arrives. Weighted spikes assign varying weights to different phases or spikes as seen in Figure 2 within those phases, enabling more information to be encoded compared to traditional rate coding, which assigns equal weight to all spikes.

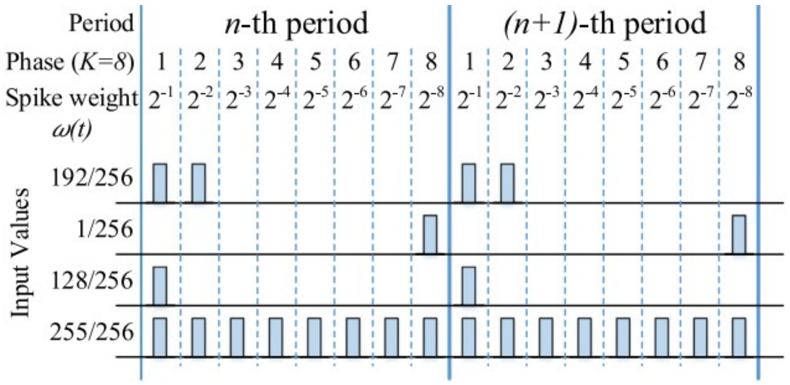

Fig. 2: Example of input spike trains when the number of phases in a period (K) is 8. [16]

Gaussian Tuning [3] Sparse TTFS coding is defined based on the relative time differences between spikes and stimuli, where this timing contains essential information. Input neurons cover the data range and employ Gaussian receptive fields to map continuous input values to specific delay times. Significant data results in small delays, while irrelevant data does not trigger action potentials within a defined time window, introducing sparsity. Gaussian receptive fields encode real-valued inputs into spike trains, requiring knowledge of the input variable's maximum and minimum values to calculate sufficient statistics (μ and σ) for the Gaussian field in the following way.

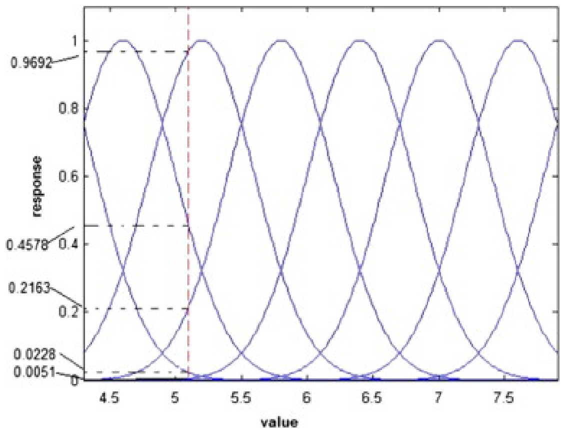

Fig. 3: Real-valued input and it's response value according to different Gaussian profiles.[3]

$$\begin{aligned}\mu &= x_{min} + \frac{(2i-3).(x_{max}-x_{min})}{2.(N-2)} \\ \sigma &= \frac{x_{max}-x_{min}}{\beta.(N-1)}\end{aligned} \tag{7}$$

Where, N is the number of neurons used to calculate Gaussian profiles for each variable, β is the parameter for controlling profile width and $i = 1 \dots N$. Once the intensities of the current variable for all N GRF profiles are determined then a certain threshold value such as max spike time is used to determine which GRF values will be represented with 1 (spike) or -1(silent). Hence we have a spike train for each input variable.

Convolution Direct coding [33] uses the floating-point inputs directly in the first layer. Pass the input image (or RGB pixel values) through the first convolution layer as shown in Figure 4 which generates floating point outputs. The float output values are repeated for T time-steps of SNN processing. These outputs are then processed through a layer of spiking neurons that generates binary spikes.

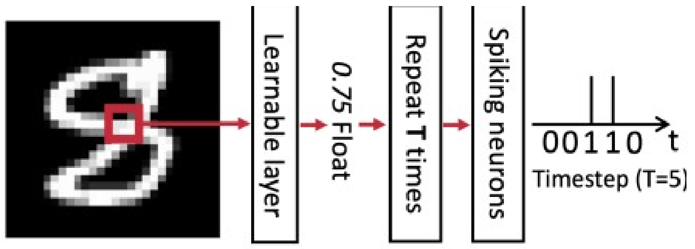

Fig. 4: Direct coding. [17]

2.5 Surrogate Functions

To alleviate the non-differentiability problem of spiking neurons the gradient is approximated by well known functions.

ArcTan Inspired by [5] ArcTan function is also a popular choice as a surrogate gradient function.

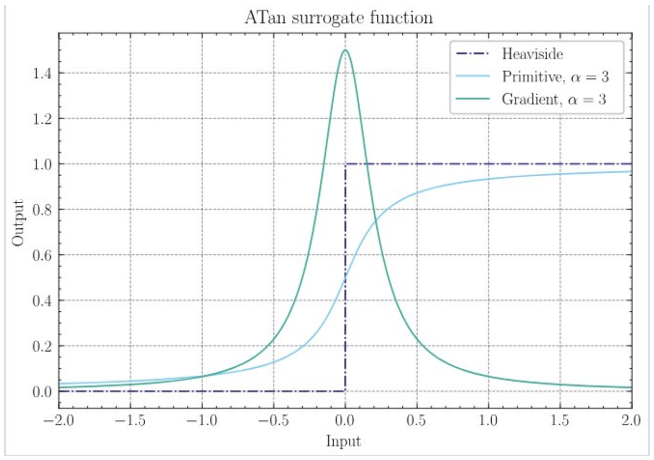

Fig. 5: ArcTan comparision with spiking step function.

Sigmoid First seen in [26]. Such an approximation outputs a 1, i.e. a spike with a probability determined by the sigmoid of its input. The derivative is then simply the derivation of the sigmoid and the approximation can be viewed in Figure 6

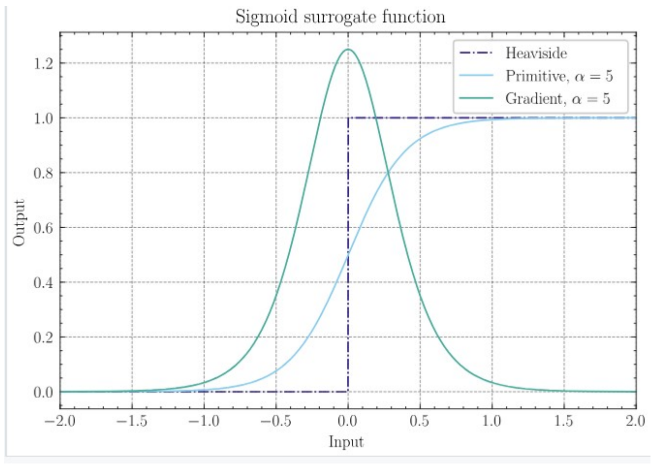

Fig. 6: Sigmoid surrogate compare to Heaviside Spike function.

SoftSign Used in [37] the primitive function is defined as

$$g(x) = \frac{1}{2}(\frac{\alpha x}{1 + |\alpha x|} + 1) \tag{8}$$

where, α is experimentally set to 2.0 and x is the spike input. It's closeness and derivative function can be seen in 7

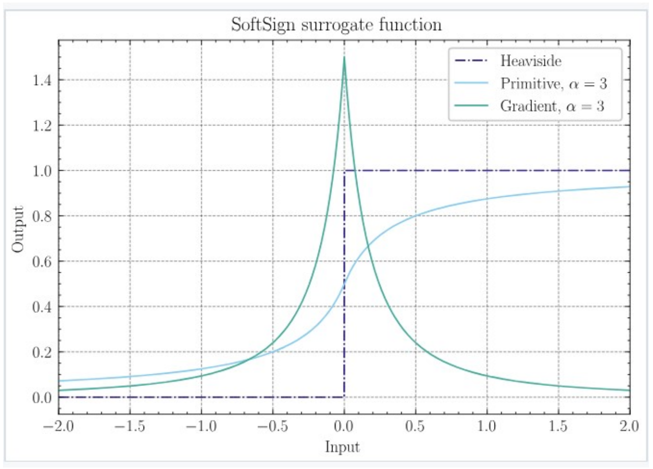

Fig. 7: SoftSign surrogate compared to Heaviside Spike function.

2.6 Event camera data

Event cameras, like the Dynamic Vision Sensor (DVS) [19], differ from traditional cameras by capturing pixel-level brightness changes instead of static intensity frames. They provide benefits like a wide dynamic range, no motion blur, and microsecond-level latency. However, due to their asynchronous event-based output, conventional vision algorithms are incompatible, necessitating the development of new algorithms to harness their high temporal resolution and asynchronous nature. It's output contains the time, location, sign of brightness change in the pixel so [timestep,x,y,polarity]. Polarity gives information on whether the pixel is increasing(+1) in brightness or reducing(-1) since the last timestep. A research compares traditional RNNs and SNNs to demonstrate their differences/ similarities on such neuromorphic data [12].

2.7 Transformers

The Transformer architecture [31] depicted in 8 is structured as a sequence-to-sequence model, composed of both an encoder and a decoder block. Within the encoder, a crucial component is the multi-head self-attention module. This mechanism involves the conceptual breakdown of the input sequence into components referred to as Query, Key, and Values.

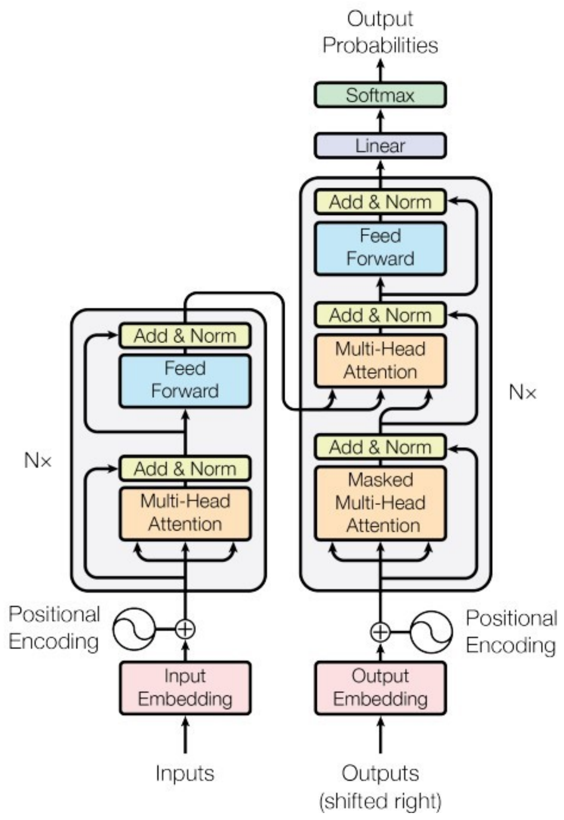

Fig. 8: Transformer Architecture[31]

The underlying concept is to enable the transformer to learn the most pertinent relationships between a token or patch and both itself and other elements present within the sequence. Subsequently, this knowledge can be effectively accessed, resembling an information retrieval system with key-value pairs. The term 'multi-head' signifies that the attention pertaining to each sequence element is consolidated into a singular vector, then subject to multiplication with its corresponding weight matrix. The Transformer architecture has become a standard for natural language processing but, its use in computer vision was limited initially, until Vision Transformer[8] where attention is applied directly to sequences of image patches can perform well on image classification tasks.

3 Implementation

This section will cover our implementation of direct training a Vision Transformer by substituting it's GELU neuron with spiking neurons on a static and a dynamic dataset.

3.1 Architecture

In this paper we use a simplified version of the Vision Transformer[8] shown in 9. The original version accepts static input image that is split into patches and modifies their dimensions, additionally the channels are converted to embedding as patches are split using a convolution layer. Next, the image dimension are flattened, and every patch sequence is assigned a class token and it's positional embedding. After this step we have achieved a representation similar to Text Classification problems. Now attention can be applied in the Transformer encoder block, which consists of two fully connected layers and activation functions with layer norm before and after each iteration of the block. Finally, the classifier or MLP block consists of one fully connected layer and activation function before a final layer to determine the class token.
In all the following experiments the GELU activation function used in the original paper will be replaced with a version of the Integrate and Fire(IF) neuron or simply spiking neuron. Another change would be that the input image will be spike encoded using different techniques to achieve a temporally representative version of the original static images.

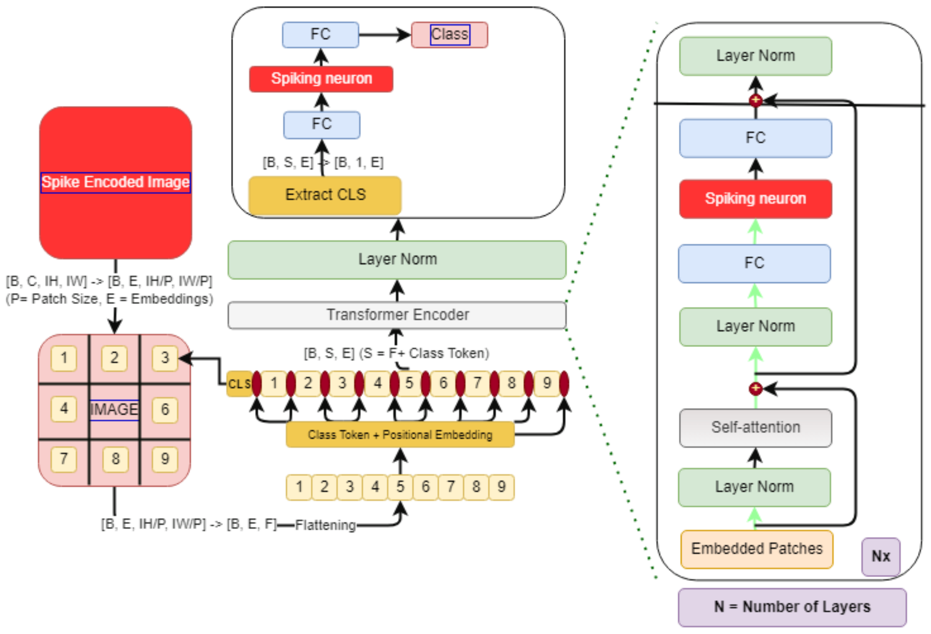

Fig. 9: Base Architecture used for all experiments.

3.2 Datasets

FMNIST Fashion-MNIST (FMNIST)[34] is a popular dataset in machine learning and computer vision. It comprises 60,000 training images and 10,000 testing

images, each belonging to one of 10 different clothing categories, such as T-shirts, dresses, and shoes. FMNIST is a grayscale dataset, with images sized at 28x28 pixels. It serves as a benchmark for image classification tasks, providing a diverse set of fashion items for training and evaluation.
DVS128 Gesture The DVS128 Gesture Dataset [1] is a collection of data designed for neuromorphic vision research and gesture recognition. It features recordings captured by a DVS128 event-based camera, which records visual information asynchronously as spikes when pixel intensity changes occur. Researchers use this dataset to develop and test spiking neural networks (SNNs) and event-based vision algorithms, exploring their potential for real-time and low-power gesture recognition applications. It comprises 1,342 instances of a set of 11 hand and arm gestures, grouped in 122 trials collected from 29 subjects under 3 different lighting conditions.

4 Experiments and Results

Following the Architecture from 9 we have trained several models with some minor differences between each training run. Firstly, we establish a vanilla case as a baseline. This model includes a ParametricLIF Node with a time constant τ of 2.5, Poisson encoding on the FMNIST dataset. Other important parameters were 15 epochs, 10 timesteps, 5-e4 decaying learning rate, 4 encoder layers and image patch size of 4. The encodings, neuron model and dataset are modified between experiments while keeping all other parameters constant.
This aim of such a setup was to ensure that we can notice any subtle changes in their training patterns. Under slightly varying conditions we have been able to capture the spike behaviour and potential accumulation. In addition to the usual accuracy and loss metrics we track and analyse metrics that are specific to the spiking model.

4.1 Neuron modification

In this experiment we trained several models by changing the activation node namely, IF, LIF and KLIF. With PLIF being already part of the baseline model we can then look at the performance of each based on it's validation accuracy. From figure 10 we can clearly notice that the model using PLIF neuron which is also the baseline model performs the best as compared to the other three. This is maybe in part because of the fact that PLIF employs a learnable time decay constant parameter τ while the LIF has a steady τ and IF is non-decaying. KLIF on the other hand does not perform so well probably because of resetting all of it's negative potentials back to zero after every successful firing. This tells us that the negative potentials with neurons might hold some relevancy.

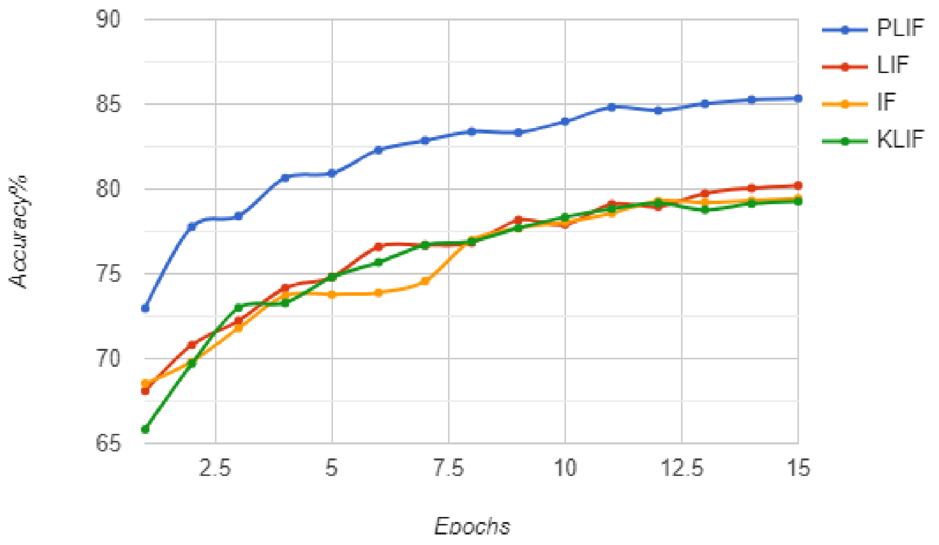

Fig. 10: Comparing Validation Accuracy between neurons.

4.2 Encoding modification

This experiment works on the same principle but here we change the encodings from Rate coding in the baseline to Weighted phase, latency, gaussian tuning and convolutional.

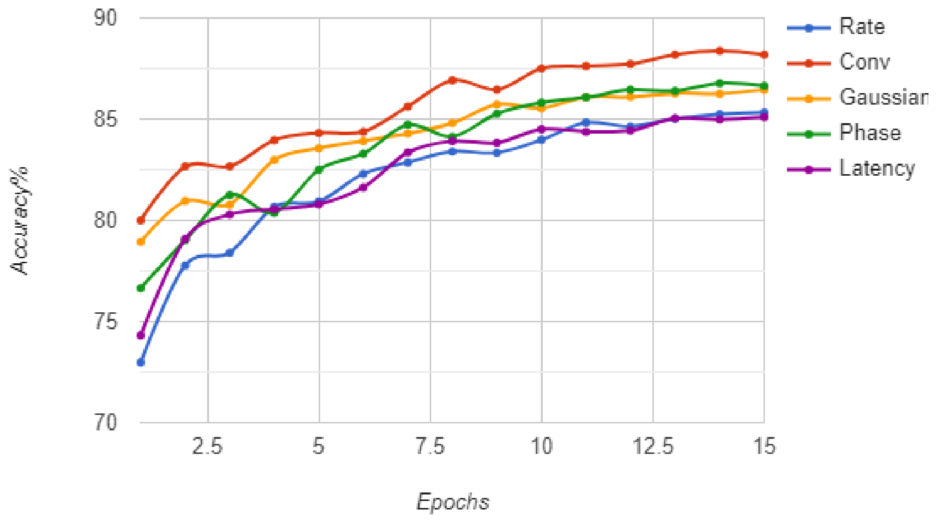

Fig. 11: Comparing Validation Accuracy between encodings

In 11 we noticed that the direct encoding method using convolutional layers followed by a spiking neuron performs the best amongst other encodings. The explanation here could simply be that the learnablity of the direct encoding is precise enough to formulate incoming static data into the respective time steps and enable the model to perform well. We can also notice that the temporal coding approaches have performed better than rate coding approach used in the Vanilla baseline. Which is also no surprise considering that randomness trumps structured temporal arrangement based on existing image properties.

4.3 Surrogate function modification

Here, we introduce models using different SG functions such as ArcTan(baseline), Sigmoid and SoftSign. From Figure 12 we unfortunately note than there isn't

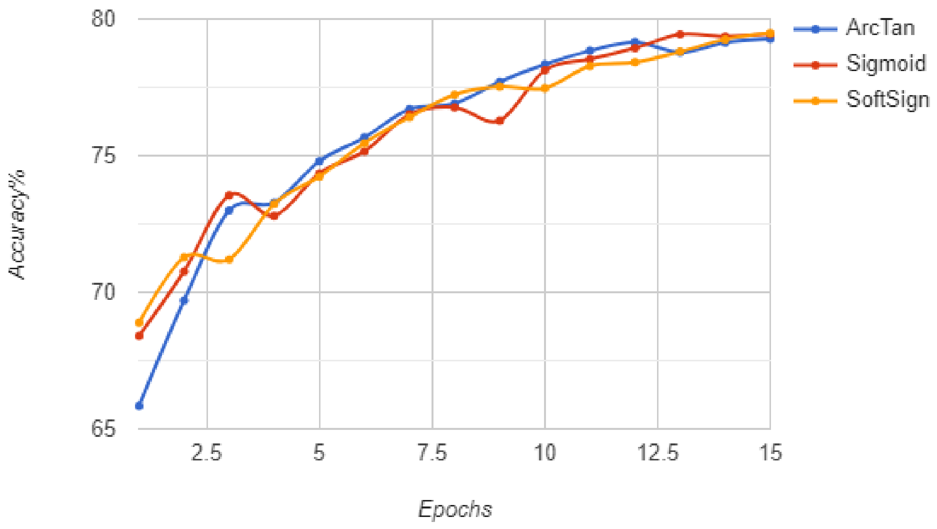

Fig. 12: Comparing Validation Accuracy between Surrogate Function.

much difference in the performance of the models. This might be due to the fact that nearly all surrogate function can be approximated well with the heaviside function by a smart choice of α as discussed in 2.5.

4.4 Dataset modification

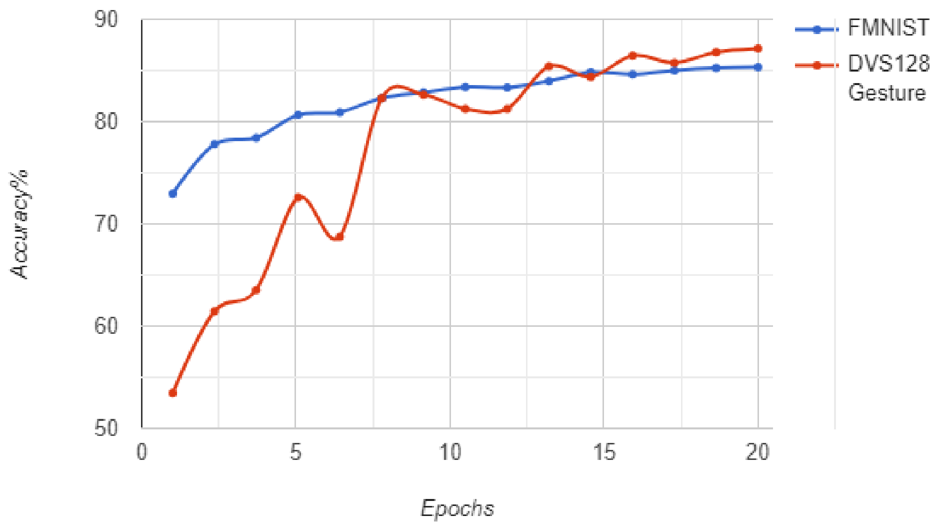

Fig. 13: Comparing Validation Accuracy between Datasets.

Finally, we have the comparison between a static and neuromorphic dataset as seen in Figure 13. The DVS128 gesture dataset has a rocky start but eventually exceeds performance and crosses the Fashion-MNIST scores. This dataset was also allowed to train a bit longer (20 epochs) to see how much it can improve but soon started to overfit on the trainset.

4.5 Spiking Percent

The key difference in SNNs is the spiking binary input that causes enough potential to fire into the next layer, this makes it intriguing to findout how many spikes are actually present in each neuron. We decide to track the number of neurons that fire after every layer activation and thus came up with spiking percent. The output of every individual neuron is arranged in a tensor with dimension [batchSize(B), data embedding(S), channel embedding(E)]. We count the number of spikes(1) in each element and calculate a percent across the entire batch size. This metric helps to identify the amount of information the model deems necessary based on the firing rates(weights) of each neuron as the learning progresses. For FMNIST in particular the spiking percentages are low since most of the images in a dataset are comprised of 0 intensity background pixels as seen in 14. This also demonstrates the efficiency of SNNs. We track a correctly classified example and a miss-classified example then compare the percentage of spikes against all of the test dataset. In 15 we see that the positive result

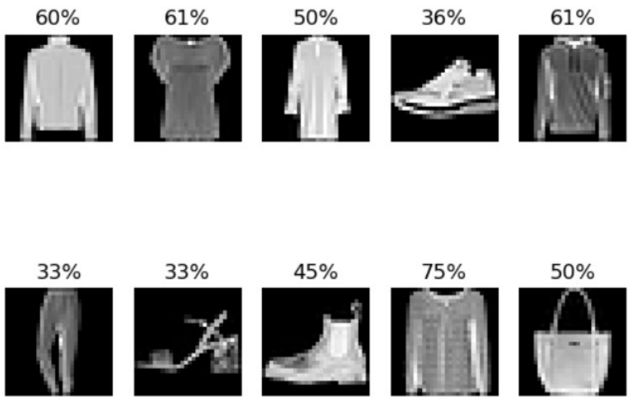

Fig. 14: Amount of pixels that are non-zero.

curves upward at the 4th encoder layer while there is no such increase for the negative result. Furthermore we can note that this increase is instead noticed on the line representing spiking percentage values for the entire testset. Due to the high accuracy most of the data samples are correctly classified and thus follow a similar trend even after being singled out. Whereas, for a wrong sample it is visibly obvious that the model learns a different pattern and miss-classifies.

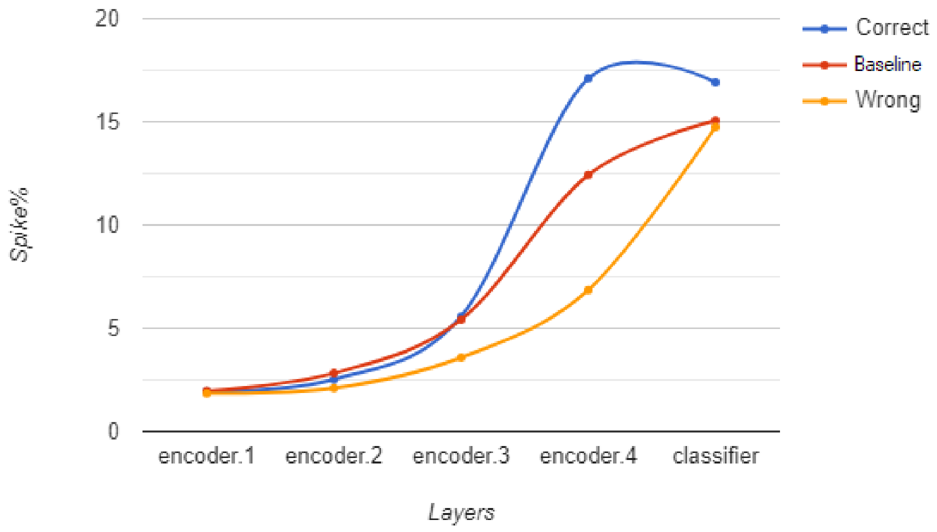

Fig. 15: Spiking metrics comparison for right and wrong class.

4.6 Negative Potential Percent

The authors of [40] argued that the membrane potential dynamically maintains the balance of excitatory and inhibitory neurons. Researchers of [39] utilised this balance as a feedback to encourage better learning patterns. In terms of our application it simply tells the machine to learn the given data in a specific pattern that discourages it to get classified and a non-target class. Therefore, this metric tracks all the negatively valued entries in the potential tensor that are present during firing of every spiking neuron. Once again we document a correctly/wrongly classified sample and the trend from the testset in Figure 16. Once again, we notice that the majority trend is followed similarly by the correct classification. While, the wrong example has an unnecessary increase at 4th encoder layer. This can actually be correlated with figure 15 where the spike percent drops in the same layer.

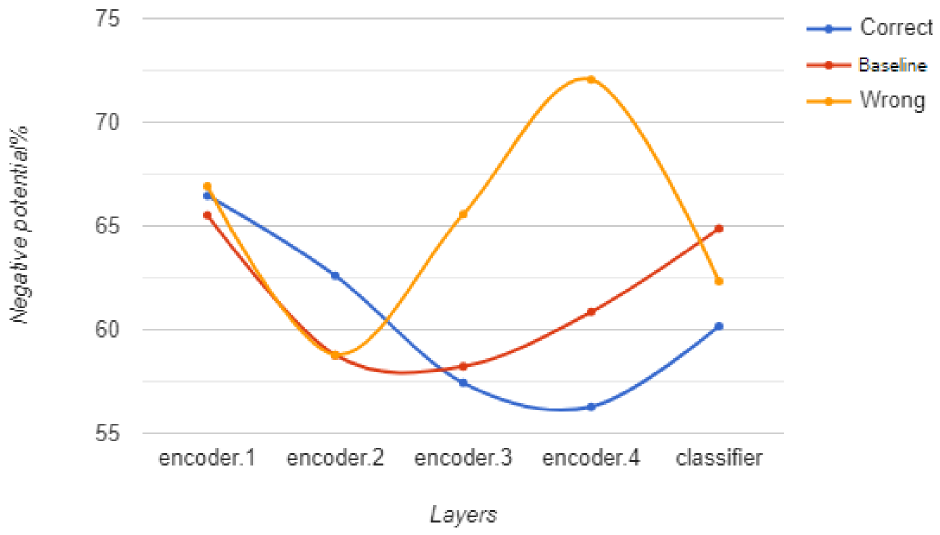

Fig. 16: Negative potential comparison for right and wrong class.

4.7 Residual potential

The potential left under utilised by each neuron after overshooting the threshold can be tracked by subtracting the potential before firing with the potential threshold V_{th}. The neuron may overshoot the V_{th} by a large margin or it maybe just enough to miss V_{th} altogether. Therefore, an absolute loss of potential or information occurs. A similar correlation can be viewed while studying the maximum potential loss after neuronal reset over all test samples in figure 17 around time step 30-40 which also corresponds to fourth encoder layer activation there is a huge dip. This indicates that a great deal of information was consumed or extracted by the model during this time which contributes to the overall prediction. In the same vein, the correctly classified example shows the exact same trajectory for 30-40 time steps also corresponding to the same encoder layer activation. While the wrongly classified example is unable to grasp a steady pattern and is visibly confused.

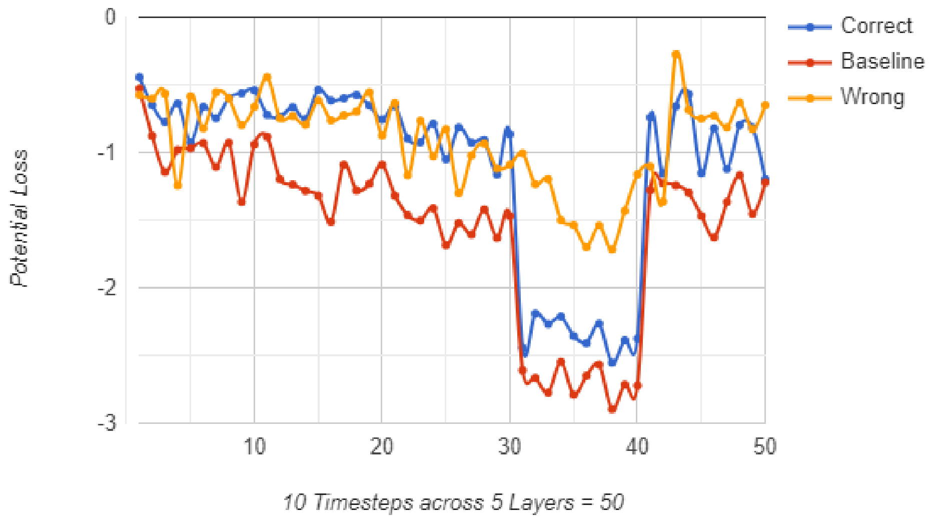

Fig. 17: Potential Loss comparison for right and wrong class.

5 Conclusion

In this article, we present an exhaustive study of the direct training approach of spiking transformers. Besides the usual well-known loss and accuracy metrics. An attempt is made at defining unique metrics relevant to the spiking behaviour. The findings show that minimal information is utilised in making successful predictions. The spiking phenomenon is highly explaninable when we look at the potential and it's changes. It's evident how even a single example can be enough to understand the learning pattern and make adjustments. This brings a fresh perspective at how the model learns and navigates it's way across various layers and time steps. Such insights are helpful as a guide to developing better training models and utilise the efficiency provided by the spiking phenomenon.

References

1. Arnon Amir, Brian Taba, David Berg, Timothy Melano, Jeffrey McKinstry, Carmelo Di Nolfo, Tapan Nayak, Alexander Andreopoulos, Guillaume Garreau, Marcela Mendoza, et al. A low power, fully event-based gesture recognition system.

In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7243–7252, 2017.
2. Katharina Bendig, René Schuster, and Didier Stricker. On the future of training spiking neural networks. 2023.
3. Sander M. Bohte, Joost N. Kok, and Han La Poutré. Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing*, 48(1):17–37, 2002.
4. Xie Chen, Yu Wu, Zhenghao Wang, Shujie Liu, and Jinyu Li. Developing real-time streaming transformer transducer for speech recognition on large-scale dataset. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5904–5908. IEEE, 2021.
5. Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.
6. Peter Dayan and Laurence F Abbott. *Theoretical neuroscience: computational and mathematical modeling of neural systems.* MIT press, 2005.
7. Peter U Diehl, Daniel Neil, Jonathan Binas, Matthew Cook, Shih-Chii Liu, and Michael Pfeiffer. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *2015 International joint conference on neural networks (IJCNN)*, pages 1–8. ieee, 2015.
8. Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.
9. Wei Fang, Zhaofei Yu, Yanqi Chen, Tiejun Huang, Timothée Masquelier, and Yonghong Tian. Deep residual learning in spiking neural networks. *Advances in Neural Information Processing Systems*, 34:21056–21069, 2021.
10. Wei Fang, Zhaofei Yu, Yanqi Chen, Timothée Masquelier, Tiejun Huang, and Yonghong Tian. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2661–2671, 2021.
11. Wulfram Gerstner and Werner M. Kistler. *Spiking Neuron Models: Single Neurons, Populations, Plasticity.* Cambridge University Press, 2002.
12. Weihua He, YuJie Wu, Lei Deng, Guoqi Li, Haoyu Wang, Yang Tian, Wei Ding, Wenhui Wang, and Yuan Xie. Comparing snns and rnns on neuromorphic vision datasets: Similarities and differences. *Neural Networks*, 132:108–120, 2020.
13. Chunming Jiang and Yilei Zhang. Klif: An optimized spiking neuron unit for tuning surrogate gradient slope and membrane potential. *arXiv preprint arXiv:2302.09238*, 2023.
14. Yifan Jiang, Shiyu Chang, and Zhangyang Wang. Transgan: Two transformers can make one strong gan. *arXiv preprint arXiv:2102.07074*, 1(3), 2021.
15. Roland S Johansson and Ingvars Birznieks. First spikes in ensembles of human tactile afferents code complex spatial fingertip events. *Nature neuroscience*, 7(2):170–177, 2004.
16. Jaehyun Kim, Heesu Kim, Subin Huh, Jinho Lee, and Kiyoung Choi. Deep neural networks with weighted spikes. *Neurocomputing*, 311:373–386, 2018.
17. Youngeun Kim, Hyoungseob Park, Abhishek Moitra, Abhiroop Bhattacharjee, Yeshwanth Venkatesha, and Priyadarshini Panda. Rate coding or direct coding: Which one is better for accurate, robust, and energy-efficient spiking neural networks? In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 71–75. IEEE, 2022.

18. Paul Kirkland, Gaetano Di Caterina, John Soraghan, and George Matich. Spikeseg: Spiking segmentation via stdp saliency mapping. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2020.
19. Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. A 128\ times 128120db15 \mu s latency asynchronous temporal contrast vision sensor. *IEEE journal of solid-state circuits*, 43(2):566–576, 2008.
20. Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021.
21. Wolfgang Maass. Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, 10(9):1659–1671, 1997.
22. Chethan M Parameshwara, Simin Li, Cornelia Fermüller, Nitin J Sanket, Matthew S Evanusa, and Yiannis Aloimonos. Spikems: Deep spiking neural network for motion segmentation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3414–3420. IEEE, 2021.
23. Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, and Timothy P Lillicrap. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*, 2019.
24. Nitin Rathi and Kaushik Roy. Diet-snn: Direct input encoding with leakage and threshold optimization in deep spiking neural networks. *arXiv preprint arXiv:2008.03658*, 2020.
25. Fred Rieke, David Warland, Rob de Ruyter Van Steveninck, and William Bialek. *Spikes: exploring the neural code*. MIT press, 1999.
26. Deboleena Roy, Indranil Chakraborty, and Kaushik Roy. Scaling deep spiking neural networks with binary stochastic activations. In *2019 IEEE International Conference on Cognitive Computing (ICCC)*, pages 50–58. IEEE, 2019.
27. Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, Michael Pfeiffer, and Shih-Chii Liu. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in neuroscience*, 11:682, 2017.
28. Abhronil Sengupta, Yuting Ye, Robert Wang, Chiao Liu, and Kaushik Roy. Going deeper in spiking neural networks: Vgg and residual architectures. *Frontiers in neuroscience*, 13:95, 2019.
29. Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.
30. Amirhossein Tavanaei, Masoud Ghodrati, Saeed Reza Kheradpisheh, Timothée Masquelier, and Anthony Maida. Deep learning in spiking neural networks. *Neural networks*, 111:47–63, 2019.
31. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
32. Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, and Luping Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in neuroscience*, 2018.
33. Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, Yuan Xie, and Luping Shi. Direct training for spiking neural networks: Faster, larger, better. In *Proceedings of the AAAI conference on artificial intelligence*, pages 1311–1318, 2019.
34. Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

35. Kashu Yamazaki, Khoa Vo, and Darshan Bulsara. Spiking neural networks and their applications: A review. *Brain sciences*, 12, 06 2022.
36. Weiwei Yu, Jian Zhou, HuaBin Wang, and Liang Tao. Setransformer: Speech enhancement transformer. *Cognitive Computation*, pages 1–7, 2022.
37. Friedemann Zenke and Surya Ganguli. Superspike: Supervised learning in multilayer spiking neural networks. *Neural computation*, 2018.
38. Jiqing Zhang, Bo Dong, Haiwei Zhang, Jianchuan Ding, Felix Heide, Baocai Yin, and Xin Yang. Spiking transformers for event-based single object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8801–8810, 2022.
39. Dongcheng Zhao, Yi Zeng, and Yang Li. Backeisnn: A deep spiking neural network with adaptive self-feedback and balanced excitatory–inhibitory neurons. *Neural Networks*, 154:68–77, 2022.
40. Jie Zhu, Man Jiang, Mingpo Yang, Han Hou, and Yousheng Shu. Membrane potential-dependent modulation of recurrent inhibition in rat neocortex. *PLoS biology*, 9(3):e1001032, 2011.