Supplementary Materials: Attention Spiking Neural Networks

Man Yao *Student Member, IEEE*, Guangshe Zhao, Hengyu Zhang, Yifan Hu, Lei Deng *Member, IEEE*, Yonghong Tian *Fellow, IEEE*, Bo Xu, and Guoqi Li *Member, IEEE*

S1 Analysis of Energy Consumption

The measure of energy cost in CNNs and SNNs are shown in Table S1. Almost all FLOPs in CNNs are MAC. By contrast, in the vanilla SNN, FLOPs of the first encoder layer are MAC while all other Conv or FC layers are AC.

TABLE S1: FLOPs for CNN and SNN models. i_m and o_m are the input and output dimensions of the FC layer, respectively. When the inputs are static images, $\Phi^0_{Conv}=1$. When the inputs are event frames, Φ^0_{Conv} is the ratio of non-zero pixels. Moreover, $\Phi^0_{FC}=\Phi^N_{Conv}$.

Model	FLOPs of a CONV or FC layer		
	Variable	Value	FLOP Type
CNN [1]	FL^n_{Conv}	$(k_n)^2 \cdot h_n \cdot w_n \cdot c_{n-1} \cdot c_n$	MAC
	FL^m_{FC}	$i_m \cdot o_m$	MAC
SNN [2]	$FL^n_{SNNConv}$	$T \cdot FL^n_{Conv} \cdot \Phi^{n-1}_{Conv}$	MAC ($n=1$) or AC ($n>1$)
	FL^m_{SNNFC}	$T \cdot FL^m_{FC} \cdot \Phi^{m-1}_{FC}$	AC

Energy Cost of Vanilla SNNs. Similar to [2], [3], we define the layer average spiking activity rate (LASAR) to analyze the energy cost related to the spiking activity: at time step t, a layer's spiking activity rate (LSAR) is the ratio of spikes produced over all the neurons to the total number of neurons in that layer; then we define the LASAR which averages LSAR across all time steps T. The LASAR of the vanilla SNN at n-th Conv layer and m-th FC layer are Φ^n_{Conv} and Φ^m_{FC}, respectively. As shown in Table S1, the number of FLOPs needed for n-th Conv and m-th FC layer of CNNs, which can be separated as FL^n_{Conv} and FL^m_{FC}, are easy to compute [2]. The formula of CNN FLOPs can be easily adjusted for an SNN. Considering the simulation step T and LASARs, we obtain FLOPs of SNNs in n-th Conv and m-th FC layer, denoted as $FL^n_{SNNConv}$ and FL^m_{SNNFC} respectively, in row 5 and 6 of Table S1. Then, we can calculate the energy cost of vanilla SNN by Table S1.

In the encoder layer of SNNs ($n=1$), FLOPs are MAC operations that are the same as CNNs, because the work of this layer is to transform analog inputs into spikes. In addition, all other Conv and FC layers transfer spikes and execute AC operations to accumulate weights of postsynaptic neurons. Thus the inference energy cost of a vanilla SNN E_{Base} can be quantified as

$$E_{Base} = E_{MAC} \cdot FL^1_{SNNConv} + E_{AC} \cdot \left(\sum_{n=2}^{N} FL^n_{SNNConv} + \sum_{m=1}^{M} FL^m_{SNNFC}\right), \tag{S1}$$

where N and M are the total numbers of layers of Conv and FC, E_{MAC} and E_{AC} represent the energy cost of MAC and AC operation, respectively. Refer to previous SNN works [2], [3], [4], [5], [6], we assume the data for various operations are 32-bit floating-point implementation in 45nm technology [7], in which $E_{MAC}=4.6pJ$ and $E_{AC}=0.9pJ$.

Additional Model and Computational Complexity. Table S2 shows the additional parameters and computational burden induced by three dimensions of attention modules. We individually consider the additional parameters and computational burden induced by three dimensions of attention module. We assume the attention module is used in each Conv layer. Since the inputs of the attention module are analog values generated by pooling, the additional computation is MAC operation. We first consider the additional parameters, which are solely from the two FC layers or one Conv layer, and therefore constitute a small fraction of the total network capacity. The results are shown in column 2 of Table S2. Then we consider the additional computation burden Δ_{MAC1} and Δ_{MAC2}, where the former comes from generating attention weights and the latter derives from refinement membrane potential. For TA, each layer executes the attention module only once, and Δ_{MAC1} has nothing to do with time step; for CA and SA, modules are repeatedly performed at each time step when executing the inference. Results of the number of additional MAC operations are shown in column 3 and 4 of Table S2.

- *M. Yao is with the School of Automation Science and Engeneering, Xi'an Jiaotong University, Xi'an, Shaanxi, China, and also with Peng Cheng Laboratory, China.*
- *G. Zhao is with the School of Automation Science and Engeneering, Xi'an Jiaotong University, Xi'an, Shaanxi, China.*
- *H. Zhang is with Tsinghua Shenzhen International Graduate School, Tsinghua University, Shenzhen, China.*
- *Y. Hu, and L. Deng are with Center for Brain-Inspired Computing Research, Department of Precision Instrument, Tsinghua University, Beijing, China.*
- *Y. Tian is with Institute for Artificial Intelligence, Peking University, Beijing, China, and also with Peng Cheng Laboratory, China*
- *B. Xu and G. Li are with Institute of Automation, Chinese Academy of Sciences, Beijing, China.*
  The corresponding author: Guoqi Li (E-mail:guoqi.li@ia.ac.cn).

TABLE S2: Additional Model and Computational Complexity. Additional parameters induced by attention modules are very small compared with baseline parameters, which can be ignored. Δ_{MAC1} is caused by the computation of attention weights. Δ_{MAC2} is induced by the refinement of membrane potential, where $N_{Conv-neuron}$ means the number of Conv neurons. Δ_{AC} derives from the drop of network spiking activity, where $\Delta\Phi^n_{TA-Conv} = \Phi^n_{Conv} - \Phi^n_{TA-Conv}$ and $\Delta^m_{TA-FC} = \Phi^m_{FC} - \Phi^m_{TA-FC}$ indicate the shift of LASAR between baseline SNN and TA-SNN in n-th Conv layer and m-th FC layer, respectively. And so on, we can get $\Delta\Phi^n_{CA-Conv}$, $\Delta\Phi^m_{CA-FC}$, $\Delta\Phi^n_{SA-Conv}$, and $\Delta\Phi^m_{SA-FC}$.

Attention	Additional Para.(↑)	Additional Computational Complexity		
		Δ_{MAC1} (MAC ↑)	Δ_{MAC2} (MAC ↑)	Δ_{AC} (AC ↓)
TA	$N \cdot \left(2 \cdot T \cdot \lfloor \frac{T}{r_t} \rfloor\right)$	$N \cdot \left(2 \cdot T \cdot \lfloor \frac{T}{r_t} \rfloor\right)$	$T \cdot N_{Conv-neuron}$	$T \cdot (\sum_{n=1}^{N-1} FL^n_{Conv} \cdot \Delta\Phi^{n-1}_{TA-Conv} + \sum_{m=1}^{M} FL^m_{FC} \cdot \Delta\Phi^{m-1}_{TA-FC})$
CA	$\sum_{n=1}^{N} \left(2 \cdot c_n \cdot \lfloor \frac{c_n}{r_c} \rfloor\right)$	$T \cdot \sum_{n=1}^{N} \left(2 \cdot c_n \cdot \lfloor \frac{c_n}{r_c} \rfloor\right)$	$T \cdot N_{Conv-neuron}$	$T \cdot (\sum_{n=1}^{N-1} FL^n_{Conv} \cdot \Delta\Phi^{n-1}_{CA-Conv} + \sum_{m=1}^{M} FL^m_{FC} \cdot \Delta\Phi^{m-1}_{CA-FC})$
SA	$N \cdot (2 \cdot 7 \cdot 7)$	$T \cdot \sum_{n=1}^{N} 2 \cdot 7 \cdot 7 \cdot h_n \cdot w_n$	$T \cdot N_{Conv-neuron}$	$T \cdot (\sum_{n=1}^{N-1} FL^n_{Conv} \cdot \Delta\Phi^{n-1}_{SA-Conv} + \sum_{m=1}^{M} FL^m_{FC} \cdot \Delta\Phi^{m-1}_{SA-FC})$

Energy Shift of Attention SNNs. By optimizing the membrane potential, the attention mechanism drops the spiking activity of SNNs in both Conv and FC layers. We can easily get how much the AC operation in the network has changed by counting the LASAR of the attention SNNs, and the computation formula is shown in column 5 of Table S2. Then, we can estimate the shift of the energy cost versus the additional computational burden $\Delta_{MAC} = \Delta_{MAC1} + \Delta_{MAC2}$ and the decreased AC operations Δ_{AC} to demonstrate the energy efficiency of the attention SNN. The absolute energy shift between vanilla and attention SNNs can be computed as

$$\Delta_E = E_{MAC} \cdot \Delta_{MAC} - E_{AC} \cdot \Delta_{AC}. \tag{S2}$$

We term the attention SNN energy consumption as E_{Att}. With the vanilla SNN as the anchor, the energy efficiency of an attention SNN is defined as

$$r_{EE} = \frac{E_{Base}}{E_{Att}} = \frac{E_{Base}}{E_{Base} + \Delta_E}. \tag{S3}$$

The higher the r_{EE}, the greater the energy efficiency of attention SNNs. Generally, we represent the r_{EE} of baseline model as $1\times$.

S2 Experimental Details

S2.1 Learning on Event-based Action Recognition

Table S3 lists details for experiments on event-based recognition datasets like learning algorithm, loss function, etc. We use the Adam optimizer for accelerating the training process and employ some standard training techniques of deep learning, such as batch normalization, dropout, etc. The hyper-parameters and specific baseline network structures of vanilla and attention SNNs are also shown in Table S3.

Moreover, in the Gait dataset, there may be some time-dependent blank areas at the beginning and end of the event stream. We remove these two parts in the process of data preprocessing by setting an event number threshold x_{th}. Specifically, we start from $t' = 0$, check the event number of each $E_{t'}$, discard the pattern $\boldsymbol{S}_{t'}$ until the event number of the $E_{t'}$ is greater than x_{th}. Similarly, we also make the discard process at the end of the data. We set $x_{th} = 25$ in this paper.

TABLE S3: Learning and hyper-parameter setting. MP4-max pooling is 4×4, nC3-Conv is 3×3 and has n output feature maps, AP2-average pooling is 2×2, nFC-Linear layer has n output feature maps.

DVS128 Gesture & DVS128 Gait		
Learning	Representation	Frame-based
	output Latency	$t_{lat} = dt \times T$
	Learning Algorithm	STBP [8]
	Data Augmentation	RCS [9]
	Loss Function	Rate Coding [8]
	Network Structure	[10]:Input-128C3-AP2 -128C3 -AP2-128C3-AP2 -128C3-AP2-128C3-AP2 -512FC-Output
		[9]:Input-MP4-64C3 -128C3-AP2-128C3 -AP2-256FC-Output
Hyper parameter	Max Epoch	100
	Batch Size	36
	Learning Rate	$1e^{-4}$
	Threshold u_{th}	0.3
	Reset potential V_{reset}	0
	Decay factor β	0.3
	Reduction factor r_c, r_t	16

S2.2 Learning on ImageNet-1K

For the experiments on ImageNet-1K, we mainly follow the network architectures of MS-ResNet [3] as in Table S5, and our training setup is detailed in Table S4. We use the SGD optimizer for training and use the cosine annealing method for learning rate scheduler. For the ImageNet-1K dataset, we use a 224×224 RandomCrop and AutoAugment as the data augmentation method. And for testing, we resize the image to size 256×256 and use 224×224 CenterCrop to obtain the input data.

S2.3 Training Efficiency on Single-time Step Res-SNNs

To evaluate the training time of single/6-time step, we use $card \cdot h/epoch$ as the normalized time unit of the measurement. Here we only compared the training cost on ImageNet-1K, results are shown in Table S6. We see that

TABLE S4: Learning and hyper-parameter setting on ImageNet-1K.

	ImageNet	
Learning	Learning Algorithm	STBP [8]
	Learning Rate Scheduler	CosineAnnealingLR
	Loss Function	CrossEntropy
	Data Augmentation	AutoAugment [11]
	Optimizer	SGD
Hyper Parameter	Max Epoch	1000
	Batch Size	600
	Learning Rate	0.1
	Momentum of SGD	0.9
	Weight Decay	1e-5
	Label Smoothness [12]	0.001
	Dropout Rate [13]	0.2
	Threshold u_{th}	0.5
	Reset potential V_{reset}	0
	Decay factor β	0.25
	Reduction factor r_c, r_t	16

TABLE S5: Network Structures for ImageNet-1K. Note, the first block in each stage contains a convolutional layer with stride size 2.

Stage	Output Size	ResNet-18	ResNet-34	ResNet-104
0	112x112	7x7, 64, stride=2		
1	56x56	[3x3, 64; 3x3, 64] * 2	[3x3, 64; 3x3, 64] * 3	[3x3, 64; 3x3, 64] * 3
2	28x28	[3x3, 128; 3x3, 128] * 2	[3x3, 512; 3x3, 128] * 4	[3x3, 128; 3x3, 128] * 8
3	14x14	[3x3, 256; 3x3, 256] * 2	[3x3, 256; 3x3, 256] * 6	[3x3, 256; 3x3, 256] * 32
4	7x7	[3x3, 512; 3x3, 512] * 2	[3x3, 512; 3x3, 512] * 3	[3x3, 512; 3x3, 512] * 8
FC	1x1	AveragePool, FC-1000		

TABLE S6: Training cost on various experiments. The unit is $card \cdot h/epoch$, which indicates the time required to use the maximum memory of one NVIDIA Tesla A100 (40G) GPU.

Models	$T = 1$	$T = 6$
Att-Res-SNN-18	0.327	2.126
Att-Res-SNN-34	0.538	3.500
Att-Res-SNN-104	1.758	11.313

TABLE S7: Effect of Different residual attention locations in Res-SNNs with $T = 1$ on ImageNet-1K

Model	Acc. (%)	NASAR
Res-SNN-18 [3]	61.70	0.224
CSA-Res-SNN-18-1	**63.97(+2.3)**	0.148
CSA-Res-SNN-18-2	63.49(+1.8)	0.137

single-time step large-scale SNNs can significantly speed up training.

S3 Additional experimental results

S3.1 Evaluation of Att-Res-SNN-1 and Att-Res-SNN-2

In Section 6.4 of the main text, we give two schemes of attention residual learning, where Att-Res-SNN-1 (our recommended method) performs attention between the residual block and shortcut. Another variant, Att-Res-SNN-2, in which the attention is moved after the shortcut. These variants are illustrated in Fig.3 and Fig.5 of the main text. The performance of each variant is reported in Table S7. We observe that both Att-Res-SNN-1 and Att-Res-SNN-2 perform well on effectiveness and efficiency concretely, where Att-Res-SNN-1 is better in the accuracy and Att-Res-SNN-2 has sparser firing. We chose Att-Res-SNN-1 as the recommended model because of its higher accuracy. Moreover, although it is beyond the scope of this work, we anticipate that further effectiveness and efficiency gains will be achievable simultaneously by tailoring backbone SNNs and attention module usage for specific tasks.

TABLE S8: Comparison with VGG-SNN baselines on CIFAR-10 (the above table) and CIFAR-100 (the below table).

Model	Top-1 Acc. (%)	NASAR	Spike Counts ($\times 10^6$)
VGG-SNN-7 ($T = 1$)	80.70	0.110	2.535
+ CSA (**This work**)	83.66**(+2.9)**	0.108	2.478
VGG-SNN-7 ($T = 6$)	84.55	0.082	11.289
+ CSA (**This work**)	87.29**(+2.7)**	0.104	14.389
VGG-SNN-11 ($T = 1$)	81.80	0.068	8.516
+ CSA (**This work**)	89.13**(+7.3)**	0.075	9.346
VGG-SNN-11 ($T = 6$)	84.62	0.047	35.476
+ CSA (**This work**)	91.91**(+7.3)**	0.057	42.871
VGG-SNN-11 ($T = 1$)	48.84	0.083	10.389
+ CSA (**This work**)	60.49**(+11.7)**	0.075	9.357
VGG-SNN-13 ($T = 1$)	49.51	0.098	14.139
+ CSA (**This work**)	58.12**(+8.6)**	0.083	11.964

TABLE S9: Effect of attention module in VGG-SNN-13 on CIFAR-100 with $T = 1$. The order of the spike counts is 10^6. Note, compared with the standard VGG-13 in CNN, VGG-SNN-13 has one less FC layer on CIFAR-100.

Layer	NASAR	NASAR (+CSA)	Spike Counts	Spike Counts (+CSA)
Conv-1	0.176	0.242	1.153	1.585 (+0.423)
Conv-2	0.169	0.189	1.107	1.237 (+0.130)
Conv-3	0.140	0.123	1.837	1.606 (-0.231)
Conv-4	0.134	0.098	1.760	1.291 (-0.469)
Conv-5	0.120	0.103	3.158	2.706 (-0.452)
Conv-6	0.064	0.034	1.679	0.915 (-0.764)
Conv-7	0.078	0.065	1.028	0.857 (-0.171)
Conv-8	0.080	0.059	1.045	0.775 (-0.270)
Conv-9	0.064	0.062	0.844	0.821 (-0.080)
Conv-10	0.039	0.012	0.513	0.153 (-0.360)
LIF-FC-1	0.157	0.183	0.016	0.019 (+0.003)

S3.2 Results in Att-VGG-SNNs

We also assess the effect of CSA modules when operating on *non-residual deep* networks by conducting experiments with the VGG-SNN architecture. Specifically, we use VGG-7/11 (T=1, 6) and VGG-11/13 (T=1) in the open source framework SpikingJelly[1] as the benchmark model to test on CIFAR-10 and CIFAR-100. To facilitate the training of

1. https://github.com/fangwei123456/spikingjelly

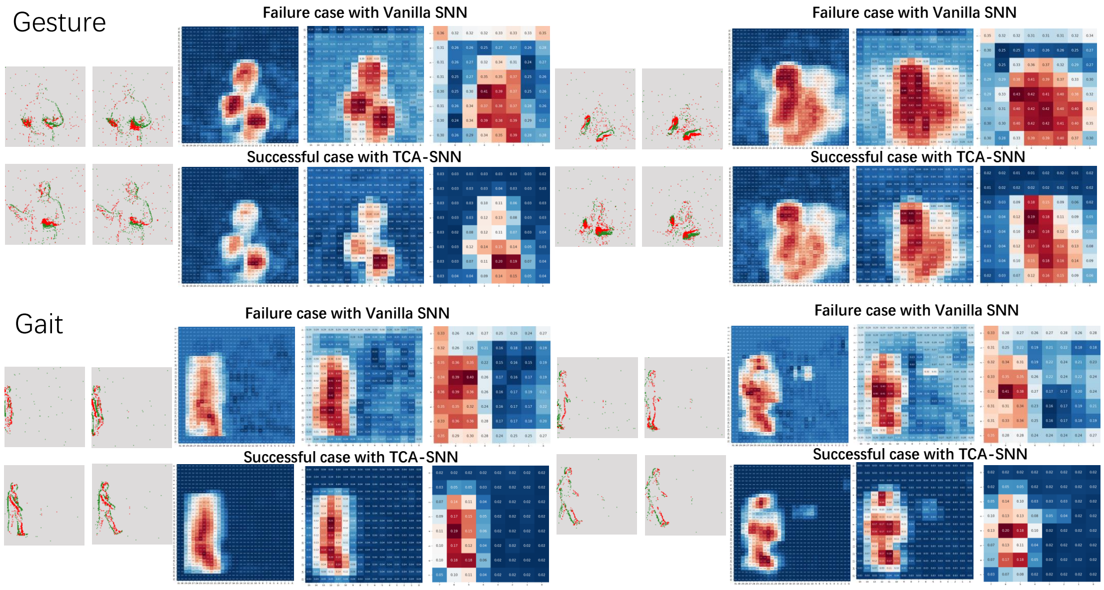

Fig. S1: Case study on event-based action recognition tasks. We can observe that attention drives SNNs to focus on the target while the vanilla model shows more decentralized spiking activations. In successful cases, the edge information in the spiking features is clearer. We see that different event streams result in distinct spiking response.

VGG-SNN from scratch, we add Batch Normalization layers for all VGG-SNN models. We plug an attention module behind each layer of VGG and exploit identical training schemes for both VGG-SNN and CSA-VGG-SNN (code is available[2]). The results of the comparison are shown in Table S8. Similarly to the results reported for the residual baseline architectures, we observe that CSA modules bring significant accuracy improvements in all experiments on the VGG-SNN settings.

As shown in Table S8, in terms of spiking firing, we see that the attention module reduces the number of spikes of vanilla SNNs in three groups of experiments, while it increasing spikes in the other three sets. We scrutinized the spiking firings of all models and found that the spike counts are strongly correlated with the dataset and model structure, which is consistent with what we observed in shallow plain SNNs and deep residual SNNs. We observe that attention modules generally increase spikes in the first encoding layers and decrease spikes in deeper layers. As an example, we show the number of spikes in VGG-SNN-13 and Att-VGG-SNN-13 in Table S9. We see that among all the convolutional layers, only Conv-1 and Conv-2 increase spikes after inserting the attention module. From Conv-3 to Conv-10, the number of spikes decreases for each layer, which makes Att-VGG-SNN-13 end up with 2.175×10^6 fewer spikes than VGG-SNN-13. Therefore, if we can choose the structure of the baseline model reasonably, the attention module is able to improve the performance of VGG-SNN while reducing spikes.

S3.3 More Case Studies

Here we add more case studies for Section 7.2 of the main text. For a single event-based sample, we averaged all the 4D ($[T, C, H, W]$) spiking maps of SNN into a 2D map ($[H, W]$) over the temporal and channel dimension at each layer. Then we plot the 2D feature, which represents the average spiking response of every layer for this sample. To visualize the effectiveness and efficiency of attention SNNs, we select an example with regard to the case of the vanilla SNN failing in recognition but the attention SNN succeeds. The visualization results are given in Fig. S1.

2. https://github.com/ridgerchu/SNN_Attention_VGG

S4 Gradient Evolvement in Att-Res-SNNs

Lemma S1 (Multiplication)**.** *(Theorem 4.1 in [14]) Given* $\boldsymbol{J} := \prod_{j=L}^{1} \boldsymbol{J}_j$*, where* $\{\boldsymbol{J}_j \in \mathbb{R}^{m_j \times m_{j-1}}\}$ *is a series of independent random matrices. If* $(\prod_{j=L}^{1} \boldsymbol{J}_j)(\prod_{j=L}^{1} \boldsymbol{J}_j)^T$ *is at least the* 1^{st} *moment unitarily invariant, we have*

$$\phi\left((\prod_{j=L}^{1} \boldsymbol{J}_j)(\prod_{j=L}^{1} \boldsymbol{J}_j)^T\right) = \prod_{j=L}^{1} \phi(\boldsymbol{J}_j \boldsymbol{J}_j^T). \tag{S4}$$

Lemma S2 (Addition)**.** *(Theorem 4.2 in [14]) Given* $\boldsymbol{J} := \prod_{j=L}^{1} \boldsymbol{J}_j$*, where* $\{\boldsymbol{J}_j \in \mathbb{R}^{m_j \times m_{j-1}}\}$ *is a series of independent random matrices. If at most one matrix in* $\boldsymbol{J}_j$ *is not a central matrix, we have*

$$\phi(\boldsymbol{J}\boldsymbol{J}^T) = \sum_j \phi(\boldsymbol{J}_j \boldsymbol{J}_j^T). \tag{S5}$$

Lemma S3 (ReLU, Conv, Orthogonal)**.** *(Table 2 in [14]) First, ReLU activation function is denoted as* $ReLU(x)(P(x > 0) = p)$ *and its Jacobian matrix is expressed as* $\boldsymbol{J}_{ReLU}$*. Secondly, Conv linear transformation is defined as* $\boldsymbol{y} := \boldsymbol{K} * \boldsymbol{x}$*, where* $*$ *is the Conv operation and* $\boldsymbol{K} \in \mathbb{R}^{c_{in} c_{out} k_h k_w} \sim i.i.d. N(0, \epsilon^2))$ *is the convolution kernel. The Jacobian matrix of Conv is written as* $\boldsymbol{J}_{Conv}$*. Finally, the orthogonal linear transformation is defined* *as $\boldsymbol{y} := \boldsymbol{K}\boldsymbol{x}$ where $\boldsymbol{K}\boldsymbol{K}^T = \gamma^2 \boldsymbol{I}$, and its Jacobian matrix is denoted as $\boldsymbol{J}_{Orth}$.*

Lemma S4 (Sigmoid Function)**.** *Sigmoid is defined as: $\sigma(x) = \frac{1}{1+e^{-x}}$, and its derivative is $\sigma'(x) = \sigma(x)(1 - \sigma(x))$. Analysis of Sigmoid is more challenging due to its complex non-linearity. Since sigmoid is point symmetric about (0, 0.5), we assume the inputs of sigmoid are around 0. Then we can simplify the $\sigma(x)$ with Taylor series around 0: $\sigma(x) \approx \sigma(0)+\sigma'(0)x = \frac{1}{2}+\frac{1}{4}x$.Therefore $\sigma'(x) \approx \frac{1}{4}$ and its Jaccobi matrix $\boldsymbol{J}_{Sig}$ is approximately $\frac{1}{4}I$, for whom we have $\phi(\boldsymbol{J}\boldsymbol{J}^T) = \frac{1}{16}$ and $\varphi(\boldsymbol{J}\boldsymbol{J}^T) = 0$.*

S5 Spiking Response of Attention SNNs

Fig.S2 shows the spiking response of vanilla SNN and TCA-SNN on Gait. The NASR of vanilla SNN is almost unchanged at each time step, which means SNN responds similarly to various inputs. With the help of data-dependent attention, the NSAR of TCA-SNN is uneven and small at the temporal axis, which induces a much lower NASAR than vanilla SNN.

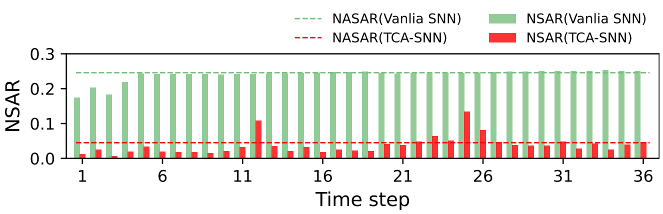

Fig. S2: Study of NSAR and NASAR on vanilla and attention SNN with Gait dataset. We set $dt = 15, T = 36$.

References

[1] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, “Pruning convolutional neural networks for resource efficient inference,” *International Conference on Learning Representations*, 2017.

[2] S. Kundu, M. Pedram, and P. A. Beerel, “Hire-snn: Harnessing the inherent robustness of energy-efficient deep spiking neural networks by training with crafted input noise,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5209–5218.

[3] Y. Hu, Y. Wu, L. Deng, and G. Li, “Advancing residual learning towards powerful deep spiking neural networks,” *arXiv preprint arXiv:2112.08954*, 2021.

[4] S. Kundu, G. Datta, M. Pedram, and P. A. Beerel, “Spike-thrift: Towards energy-efficient deep spiking neural networks by limiting spiking activity via attention-guided compression,” in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, pp. 3953–3962.

[5] B. Yin, F. Corradi, and S. M. Bohté, “Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks,” *Nature Machine Intelligence*, vol. 3, no. 10, pp. 905–913, 2021.

[6] P. Panda, S. A. Aketi, and K. Roy, “Toward scalable, efficient, and accurate deep spiking neural networks with backward residual connections, stochastic softmax, and hybridization,” *Frontiers in Neuroscience*, vol. 14, p. 653, 2020.

[7] M. Horowitz, “1.1 computing’s energy problem (and what we can do about it),” in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. IEEE, 2014, pp. 10–14.

[8] Y. Wu, L. Deng, G. Li, J. Zhu, and L. Shi, “Spatio-temporal backpropagation for training high-performance spiking neural networks,” *Frontiers in Neuroscience*, vol. 12, p. 331, 2018.

[9] M. Yao, H. Gao, G. Zhao, D. Wang, Y. Lin, Z. Yang, and G. Li, “Temporal-wise attention spiking neural networks for event streams classification,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 10 221–10 230.

[10] W. Fang, Z. Yu, Y. Chen, T. Masquelier, T. Huang, and Y. Tian, “Incorporating learnable membrane time constant to enhance learning of spiking neural networks,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 2661–2671.

[11] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, “Autoaugment: Learning augmentation strategies from data,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 113–123.

[12] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[14] Z. Chen, L. Deng, B. Wang, G. Li, and Y. Xie, “A comprehensive and modularized statistical framework for gradient norm equality in deep neural networks,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 1, pp. 13–31, 2022.