APPENDIX

A BACKPROPAGATION THROUGH TIME IN SPIKING NEURAL NETWORKS

The content of this section is mostly referred to Lv et al. (2023).

Given a loss function L like Equation 7 and 8, the losses at every time step can be summed together to give the following global gradient:

$$\frac{\partial L}{\partial W} = \sum_t \frac{\partial L_t}{\partial W} = \sum_i \sum_{j \leq i} \frac{\partial L_i}{\partial W_j} \frac{\partial W_j}{\partial W} \tag{9}$$

where i and j denote different time steps, and L_t is the loss calculated at time step t. No matter which time step is, the weights of an SNN are shared across all steps. Therefore, we have $W_0 = W_1 = \cdots = W$, which also indicates that $\frac{\partial W_j}{\partial W} = 1$. Thus, Equation (9) can be written as follows:

$$\frac{\partial L}{\partial W} = \sum_i \sum_{j \leq i} \frac{\partial L_i}{\partial W_j} \tag{10}$$

Based on the chain rule of derivatives, we obtain:

$$\begin{aligned} \frac{\partial L}{\partial W} &= \sum_i \sum_{j \leq i} \frac{\partial L_i}{\partial S_i} \frac{\partial S_i}{\partial U_i} \frac{\partial U_i}{\partial W_j} \\ &= \sum_i \frac{\partial L_i}{\partial S_i} \frac{\partial S_i}{\partial U_i} \sum_{j \leq i} \frac{\partial U_i}{\partial W_j} \end{aligned} \tag{11}$$

where $\frac{\partial L_i}{\partial S_i}$ is the derivative of the cross-entropy loss at the time step i with respect to S_i, and $\frac{\partial S_i}{\partial U_i}$ can be easily derived using surrogate gradients like Equation 3. As to the last term of $\sum_{j \leq i} \frac{\partial U_i}{\partial W_j}$, we can split it into two parts:

$$\sum_{j \leq i} \frac{\partial U_i}{\partial W_j} = \frac{\partial U_i}{\partial W_i} + \sum_{j \leq i-1} \frac{\partial U_i}{\partial W_j} \tag{12}$$

From Equation (2), we know that $\frac{\partial U_i}{\partial W_i} = X_i$. Therefore, Equation (9) can be simplified as follows:

$$\frac{\partial L}{\partial W} = \sum_i \underbrace{\frac{\partial L_i}{\partial S_i} \frac{\partial S_i}{\partial U_i}}_{\text{constant}} \left(\underbrace{\frac{\partial U_i}{\partial W_j}}_{\text{constant}} + \sum_{j \leq i-1} \frac{\partial U_i}{\partial W_j} \right) \tag{13}$$

By the chain rule of derivatives over time, $\frac{\partial U_i}{\partial W_j}$ can be factorized into two parts:

$$\frac{\partial U_i}{\partial W_j} = \frac{\partial U_i}{\partial U_{i-1}} \frac{\partial U_{i-1}}{\partial W_j} \tag{14}$$

It is easy to see that $\frac{\partial U_i}{\partial U_{i-1}}$ is equal to β from Equation (2), and Equation (9) can be written as:

$$\frac{\partial L}{\partial W} = \sum_i \underbrace{\frac{\partial L_i}{\partial S_i} \frac{\partial S_i}{\partial U_i}}_{\text{constant}} \left(\underbrace{\frac{\partial U_i}{\partial W_j}}_{\text{constant}} + \sum_{j \leq i-1} \underbrace{\frac{\partial U_i}{\partial U_{i-1}}}_{\text{constant}} \frac{\partial U_{i-1}}{\partial W_j} \right) \tag{15}$$

We can treat $\frac{\partial U_{i-1}}{\partial W_j}$ recurrently as Equation (12). Finally, we can update the weights W by the rule of $W = W - \eta \frac{\partial L}{\partial W}$, where η is a learning rate.

B DATASETS

The benchmark we used in Table 1 includes the following datasets:

- **MR**: MR stands for Movie Review and it consists of movie-review documents labeled with respect to their overall sentiment polarity (positive or negative) or subjective rating (Pang & Lee, 2005).
- **SST-5**: SST-5 contains $11,855$ sentences extracted from movie reviews for sentiment classification (Socher et al., 2013). There are 5 categories (very negative, negative, neutral, positive, and very positive).
- **SST-2**: The binary version of SST-5. There are just 2 classes (positive and negative).
- **Subj**: The task of this dataset is to classify a sentence as being subjective or objective[1].
- **ChnSenti**: ChnSenti comprises about $7,000$ Chinese hotel reviews annotated with positive or negative labels[2].
- **Waimai**: There are about $12,000$ Chinese user reviews collected by a food delivery platform for binary sentiment classification (positive and negative)[3] in this dataset.

C PERFORMANCE ON GLUE

General Language Understanding Evaluation (GLUE) benchmark is a collection of diverse natural language understanding tasks. We report the performance of SpikeBERT on GLUE benchmark on Table 4

Table 4: Classification accuracy achieved by different models on the GLUE benchmark. A BERT model fine-tuned on the dataset is denoted as "FT BERT". "SNN-TextCNN" is a SNN baseline proposed by Lv et al. (2023). $*$ indicates that the model fails to converge. All reported experimental results are averaged across 10 random seeds.

Task Metric	SST-2 Acc	MRPC F1	RTE Acc	QNLI Acc	MNLI-(m/mm) acc	QQP F1	CoLA Matthew's corr	STS-B Spearman's corr
FT BERT	92.31	89.80	69.31	90.70	83.82/83.41	90.51	60.00	89.41
SNN-TextCNN	80.91	80.62	47.29*	56.23*	64.91/63.69	0.00*	−5.28*	0.00*
SpikeBERT	85.39	81.98	57.47	66.37	71.42/70.95	68.17	16.86*	18.73*

Although SpikeBERT significantly outperforms the SNN baseline on all tasks, we find that the performance of SpikeBERT on the Natural Language Inference (NLI) task (QQP, QNLI, RTE) is not satisfactory compared to fine-tuned BERT. The possible reason is that we mainly focus on the semantic representation of a single sentence in the pre-training distillation stage. Meanwhile, we have to admit that SpikeBERT is not sensitive to the change of certain words or synonyms, for it fails to converge on CoLA and STS-B datasets. We think that's because spike trains are much worse than floating-point data in representing fine-grained words. In the future, we intend to explore the incorporation of novel pre-training loss functions to enhance the model's ability to model sentence entailment effectively.

D THEORETICAL ENERGY CONSUMPTION CALCULATION

According to Yao et al. (2022), for spiking neural networks (SNNs), the theoretical energy consumption of layer l can be calculated as:

$$Energy(l) = E_{AC} \times SOPs(l) \tag{16}$$

where SOPs is the number of spike-based accumulate (AC) operations. For traditional artificial neural networks (ANNs), the theoretical energy consumption required by the layer b can be estimated by

$$Energy(b) = E_{MAC} \times FLOPs(b) \tag{17}$$

---

[1] `https://www.cs.cornell.edu/people/pabo/movie-review-data/`

[2] `https://raw.githubusercontent.com/SophonPlus/ChineseNlpCorpus/master/datasets/ChnSentiCorp_htl_all/ChnSentiCorp_htl_all.csv`

[3] `https://raw.githubusercontent.com/SophonPlus/ChineseNlpCorpus/master/datasets/waimai_10k/waimai_10k.csv`

where FLOPs is the floating point operations of b, which is the number of multiply-and-accumulate (MAC) operations. We assume that the MAC and AC operations are implemented on the 45nm hardware (Horowitz, 2014), where $E_{MAC} = 4.6pJ$ and $E_{AC} = 0.9pJ$. Note that 1J = 10^3 mJ $= 10^{12}$ pJ. The number of synaptic operations at the layer l of an SNN is estimated as

$$SOPs(l) = T \times \gamma \times FLOPs(l) \tag{18}$$

where T is the number of times step required in the simulation, γ is the firing rate of input spike train of the layer l.

Therefore, we estimate the theoretical energy consumption of SpikeBERT as follows:

$$E_{SpikeBERT} = E_{MAC} \times EMB^1_{emb} + E_{AC} \times \left(\sum_{m=1}^{M} \mathrm{SOP}^m_{\text{SNN FC}} + \sum_{n=1}^{N} \mathrm{SOP}_{\text{SSA}} \right) \tag{19}$$

where EMB^1_{emb} is the embedding layer of SpikeBERT. Then the SOPs of m SNN Fully Connected Layer (FC) and l SSA are added together and multiplied by E_{AC}.

E ENERGY REDUCTION COMPARED TO OTHER BERT VARIANTS

We compare the energy reduction between SpikeBERT, Tiny BERT(Jiao et al., 2019), and DistilBERT(Sanh et al., 2019) in Table 5.

Table 5: Energy consumption per sample of fine-tuned BERT, SpikeBERT, TinyBERT and DistilBERT during inference on 3 text classification benchmarks. "FLOPs" denotes the floating point operations of ANNs. "SOPs" denotes the synaptic operations of SpikeBERT. "Energy" denotes the average theoretical energy required for each test example prediction.

Dataset	**Model**	**Parameters(M)**	**FLOPs/SOPs(G)**	**Energy(mJ)**	**Energy Reduction**	**Accuracy(%)**
Waimai	FT BERT	109.0	22.46	103.38	-	90.27
	SpikeBERT	109.0	27.81	29.90	71.08% ↓	89.66
	TinyBERT	67.0	11.30	52.01	49.69% ↓	89.72
	DistilBERT	52.2	7.60	34.98	66.16% ↓	89.40
ChnSenti	FT BERT	109.0	22.46	103.38	-	89.48
	SpikeBERT	109.0	28.47	30.51	70.49% ↓	86.36
	TinyBERT	67.0	11.30	52.01	49.69% ↓	88.70
	DistilBERT	52.2	7.60	34.98	66.16% ↓	87.41
SST-2	FT BERT	109.0	22.23	102.24	-	92.31
	SpikeBERT	109.0	27.46	28.54	72.09% ↓	85.39
	TinyBERT	67.0	11.30	52.01	49.13% ↓	91.60
	DistilBERT	52.2	7.60	34.98	65.78% ↓	90.40

We want to state again that spiking neural networks and model compressing are two different technological pathways to achieve energy efficiency. Future advancements in neuromorphic hardware are expected to decrease energy consumption further.

F DISCUSSION OF LIMITATIONS

In the image classification task, spiking neural networks have demonstrated comparable performance to ViT on CIFAR-10-DVS and DVS-128-Gesture datasets, which are neuromorphic event-based image datasets created using dynamic vision sensors. We think that the performance gap between SNNs and ANNs in language tasks is mainly due to the lack of neuromorphic language datasets. It is unfair to evaluate SNNs on the datasets that were created to train and evaluate ANNs because these datasets are mostly processed by continuous values. However, it is quite hard to convert language to neuromorphic information without information loss. We hope there will be a new technology to transfer sentences to neuromorphic spikes.

In addition, GPU memory poses a limitation in our experiments. Spiking neural networks have an additional dimension, denoted as T (time step), compared to artificial neural networks. Increasing the number of time steps allows for capturing more information but results in an increased demand for GPU memory by a factor of T. During our experiments, we observe that maintaining the same number of time steps during training requires reducing the sentence length of input sentences, which significantly constrains the performance of our models. We remain optimistic that future advancements will provide GPUs with sufficient memory to support the functionality of SNNs.