Homeostasis-aware Direct Spike Encoding for Deep Spiking Neural Networks

Yechan Kang[1,2]* **Mingyeong Seo**[1]* **Jeonghee Jo**[1] **Hyun Jae Jang**[1] **Jongkil Park**[1]
Jaewook Kim[1] **Suyoun Lee**[1] **Jinkyu Kim**[2] **Seongsik Park**[1]†
[1]Center for Semiconductor Technology, Korea Institute of Science and Technology, Korea
[2]Department of Computer Science and Engineering, Korea University, Korea
{kyccj,mgseo,jh.jo2,hjjang,jongkil,jaewookk,slee_eels}@kist.re.kr
jinkyukim@korea.ac.kr, seong.sik.park@kist.re.kr

Abstract

Deep spiking neural networks (SNNs), gaining attention as the next generation of artificial neural networks, have been successfully applied to many applications thanks to the development of various algorithms, such as spike encoding. Spike encoding represents input information as discrete spikes, significantly influencing the performance and efficiency of deep SNNs. Most state-of-the-art deep SNN models have greatly improved their performance by using direct encoding. However, performance and efficiency have been limited by the lack of consideration for the brain's efficiency mechanisms, such as homeostasis. To overcome this limitation, we propose H-Direct, a spike encoding technique designed to balance both effectiveness and efficiency, based on a comprehensive analysis of conventional direct encoding. Furthermore, experimental results confirm that our proposed encoding surpasses traditional direct encoding in both performance and efficiency across multiple image classification benchmarks.

1 Introduction

Deep learning has improved AI performance but comes with high computational and energy costs [1, 2]. Neuromorphic computing, inspired by the brain, offers a more energy-efficient solution using spiking neural networks (SNNs) [3–5]. Recently, deep SNNs, which merge the structure of DNNs with the energy-efficient nature of SNNs, are expected to be the next-generation artificial neural networks for energy-efficient AI that can simultaneously achieve high learning performance and low-energy operation. To maximize the advantages of deep SNNs, an efficient neural coding scheme is essential, particularly for input spike encoding, which greatly impacts performance and efficiency. Direct encoding, used in most state-of-the-art deep SNN models, preserves more input information and learns from data, offering superior performance. However, the existing direct encoding does not adequately consider the efficacy and efficiency of spike encoding, which restricts the overall performance and efficiency of deep SNNs. For optimal results, spikes should be encoded at an appropriate firing rate with dynamically determined features based on input.

In this work, to overcome the aforementioned limitation, we first investigated conventional direct spike encoding. Based on a comprehensive analysis of the observed phenomena and inspired by the human brain, we propose H-Direct, which promotes dynamic feature selection while suppressing both over- and under-firing. First, we define homeostasis in H-Direct and propose a dynamic feature encoding loss that normalizes the input distribution of each channel in the encoding neurons to

*Equal contribution
†Corresponding author

1st Workshop on NeuroAI @ 38th Conference on Neural Information Processing Systems (NeurIPS 2024).

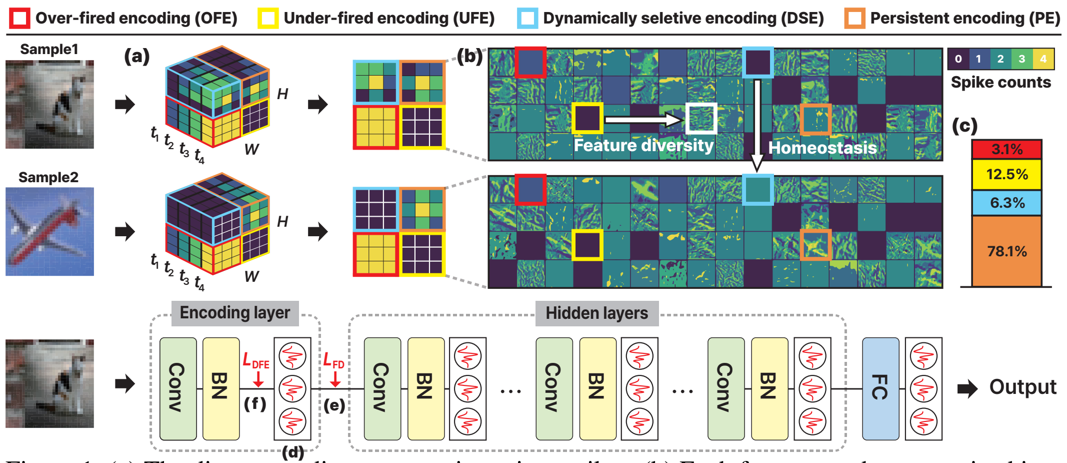

Figure 1: (a) The direct encoding converts input into spikes. (b) Each feature can be categorized into four types. (c) Proportions of each categorized encoding. Our encoding achieves this with three main modules: (d) adaptive threshold, (e) feature diversity loss, and (f) dynamic feature encoding loss.

improve homeostasis. Next, to suppress over- and under-firing, we introduce an adaptive threshold and feature diversity loss to the encoding neurons to maximize the entropy of encoded features. Our experiments demonstrate that these mechanisms improve both the learning performance and efficiency of deep SNNs.

2 Preliminaries and Related Work

2.1 Deep Spiking Neural Networks

SNNs, which mimic the operation of the brain, have been considered the next generation of artificial neural networks [6]. SNNs propagate information using spikes through a network of neurons and synapses, enabling energy-efficient operations with asynchronous event-driven computing. Deep SNNs can simultaneously achieve high learning ability and energy-efficient operation by integrating the synaptic topology of DNNs with the event-based operation of SNNs [7]. Leaky integrate-and-fire (LIF) neurons are widely used in deep SNNs due to their low computational cost. The integration process of LIF neurons is explained in more detail in Sec. A.1 Recently, various deep learning applications and models have been implemented with deep SNNs [8–15]. Most of these SOTA deep SNN models adopted spatio-temporal back-propagation (STBP) with surrogate gradient [16], threshold-dependent batch normalization (tdBN) [17], and direct spike encoding [17–20]. Despite these advancements, performance gaps between DNNs and deep SNNs persist. Despite efforts to reduce these gaps, such as improving learning algorithms [19, 21], and addressing gradient mismatch [22, 23], spike encoding has not received sufficient attention.

2.2 Spike Encoding

Spike encoding transforms input signals into spikes, allowing SNNs to process data [24]. This process significantly impacts the performance and efficiency of SNNs. Previous research has proposed various encoding schemes such as rate [25] and temporal encoding, including phase [26, 27], and time-to-first-spike (TTFS) [28–31] for the efficient processing of input in deep SNNs, but these methods limited the performance due to the loss of input information. SOTA deep SNN models address this issue by adopting direct encoding [13, 14, 18], designating the first layer as the encoding layer and training it end-to-end. While effective across various datasets and models [13, 17–20], this method is limited by the lack of sufficient consideration for SNN characteristics.

2.3 Homeostasis in SNNs

Homeostasis is essential for maintaining stability in biological systems [32], and its absence degrades information processing and efficiency [33–35]. Thus, the learning process of the neural network should incorporate homeostatic mechanisms to maintain appropriate firing rate [36]. Few studies have introduced the biological efficiency of homeostasis on SNNs. In [37], homeostasis was introduced via an adaptive threshold to improve training but could not be applied to deep SNNs. Another recent study demonstrated adversarial robustness through homeostasis with an adaptive threshold, but it focused only on stability and didn't address improvements in information processing [38]. Hence, it is imperative to investigate methods for enhancing training performance and efficiency of deep SNNs with homeostasis.

3 Analysis of Conventional Direct Encoding

To improve spike encoding, we analyzed the conventional direct encoding used in deep SNNs, which employs the first layer for feature extraction and encoding into spikes [8, 13–15, 17]. The direct encoding extracts features from input data and encodes them into spikes in a channel-wise manner according to the time step t, as shown in Fig. 1-(a). The accumulated encoded spikes during a total time step T are depicted in Fig. 1-(b). From an encoding perspective, we found that the encoded features (channels) can be categorized into four types. Over- and under-fired encoding (OFE and UFE, respectively) are caused by an inadequate firing rate during the encoding process. These inappropriately encoded channels cannot encode any features because all neurons in the channel are fired with the same value. As in many other studies [39, 40], such inappropriate encoding should be avoided since it limits the training ability of deep SNNs. Persistent encoding (PE) consistently converts the extracted features into spikes regardless of input, as in DNNs. In this case, the generation of encoded spikes in every input results in inefficient deep SNNs which rely on event-driven computing. Dynamically selective encoding (DSE), however, encodes only essential features based on the input, reducing spike count and improving energy efficiency. As illustrated in Fig. 1-(b), such selective encoding according to inputs can reduce the number of spikes, thereby improving the energy efficiency of deep SNNs. Fig. 1-(c) shows the ratio of the four encoding types, with PE being the most common and inappropriate channels also present. This indicates that conventional direct encoding needs improvement. To enhance deep SNN performance and efficiency, the encoding layer must maintain an appropriate firing rate and selectively encode features.

4 Homeostasis-aware Direct Spike Encoding

To enhance spike encoding efficiency and effectiveness, we propose a homeostasis-aware direct spike encoding. Homeostasis in H-Direct pertains to the constancy of a feature channel across different inputs, highlighting the importance of selective feature encoding. To improve this, we introduce dynamic feature encoding loss. Additionally, we introduce an adaptive threshold to suppress under-firing. And we introduce feature diversity loss to maximize the entropy of encoded features. The proposed encoding is applied to training the encoding layer, which is the first layer as in the conventional direct encoding.

4.1 Dynamic Feature Encoding Loss

Firstly, we introduce a method to enhance homeostasis, which denotes the consistency enabling a channel to maintain a suitable firing rate across inputs. This can be achieved through DSE, which was identified in our analysis. We confirmed that each encoding type defined in Fig. 1-(b) has a distinct distribution, and each these types can be defined by the ratio of the shift parameter (β) to the scale parameter (γ) in tdBN in the encoding layer. Inspired by this, we propose a dynamic feature encoding loss, which is stated as

$$L_{\text{DFE}} = \sum_{c} \left\| \frac{\beta_c}{\gamma_c + \epsilon} - \alpha \right\|_2, \tag{1}$$

where c, α, and ϵ are the channel index, the target value of the ratio(β/γ), and a small positive number for the numerical stability, respectively. This loss encourages the ratio to be trained to the target value α at which DSE channels are likely to occur. The gradients of each parameter for the loss and more detailed derivation are Sec. A.3. This loss dynamically selects essential features to be encoded as spikes based on the input, improves homeostasis by adjusting the firing rate across channels, and thereby enhances the efficiency and performance of deep SNNs.

Table 1: Experimental results for various models and datasets configurations.

		Accuracy (%)			# of total spikes (K)			# of encoded spikes (K)		
Datasets	Architecture	Baseline	Ours	Δ	Baseline	Ours	Δ	Baseline	Ours	Δ
CIFAR10	VGG16	93.47±0.14	**93.67±0.06**	**0.21**	148±8.0	**144±6.0**	**-3.00%**	59±2.00	**52±0.8**	**-12.00%**
	ResNet19	95.61±**0.03**	**95.72**±0.18	**0.12**	825±12.0	**781±11.0**	**-5.30%**	190±**0.04**	**188**±0.8	**-1.05%**
	ResNet20	94.99±**0.02**	**95.09**±0.04	**0.11**	480±11.0	**463±5.0**	**-3.54%**	92±1.20	**83±0.9**	**-9.78%**
CIFAR100	VGG16	69.03±0.13	**69.29±0.05**	**0.38**	160±**0.6**	**151**±1.0	**-6.00%**	64±1.00	**57±0.9**	**-11.00%**
	ResNet19	76.86±**0.05**	**77.07**±0.10	**0.23**	1003±8.0	**987±7.0**	**-1.60%**	222±2.00	**217±0.6**	**-2.30%**
	ResNet20	74.92±**0.03**	**75.13**±0.12	**0.28**	629±5.0	**624±0.8**	**-0.78%**	121±0.60	**118±0.5**	**-2.29%**
ImageNet	ResNet18	64.07±0.08	**64.30±0.03**	**0.36**	2175±15.0	**2051±7.0**	**-5.70%**	872±8.00	**722±5.0**	**-17.20%**
CIFAR10-DVS	VGG16	75.10±**0.16**	**76.15**±0.62	**1.40**	413±**1.2**	**273**±1.8	**-33.90%**	146±0.50	**18±0.2**	**-87.74%**

4.2 Adaptive Threshold in Encoding Neurons

As we discussed in the previous section, the conventional direct encoding leads to UFE due to an improper firing rate, limiting the encoding layer's performance. To overcome this, we introduce an adaptive threshold in encoding neurons as follows:

$$V_{\text{th},c}(t) = \begin{cases} \eta V_{\text{th},c}(t-1) & \text{if} \quad \sum_{\{i \in Channel_c\}} s_i[t] = 0 \\ V_{\text{th},c}(0) & \text{otherwise} \end{cases}, \tag{2}$$

where c, η, and $V_{\text{th}}(0)$ denote channel index, adjust rate, and initial threshold, respectively. The proposed threshold is adjusted in a channel-wise manner to ensure computational efficiency and precise adjustment. This adjustment can be cumulative, but once firing occurs, the threshold is restored to its initial value for the subsequent time steps. This method enables the encoding layer to fully utilize its potential by promoting the encoding of non-firing channels.

4.3 Feature Diversity Loss

In contrast to DNNs, SNNs output binary spikes, limiting feature diversity, further constrained by inadequate encodings like OFE and UFE. To address this issue, we propose an entropy-based feature diversity loss, which fits the feature with a probability density function (PDF) and trains the encoding layer's weights to maximize entropy. The general form of the proposed loss can be represented as

$$L_{\text{FD}} = -\sum_k p(x_k) \log p(x_k), \tag{3}$$

where x_k and $p(x_k)$ denote the feature and PDF, respectively. We used each neuron's accumulated spikes as the feature to reduce the distortion of PDF fitting. In order for the proposed loss to be compatible with a gradient-based training algorithm, the PDF must be differentiable. Thus, we used a normal distribution $\mathcal{N}(\mu, \sigma)$ as the PDF, where μ and σ are the mean and standard deviation of accumulated spikes. Details on the gradient of the encoding layer for feature diversity loss are in Sec. A.4. This method improves encoding performance by encouraging diverse feature encoding.

Our overall loss function including cross-entropy loss for the image classification is as follows:

$$L = \lambda_{\text{CE}} L_{\text{CE}} + \lambda_{\text{DFE}} L_{\text{DFE}} + \lambda_{\text{FD}} L_{\text{FD}}, \tag{4}$$

where λ_{CE}, λ_{DFE}, and λ_{FD} denote the weighting factors of L_{CE}, L_{DFE}, and L_{FD}, respectively.

5 Experiments

Effectiveness and Efficiency For details about experiments please refer to Sec. A.5. We investigated the impact of improved homeostasis on the training performance of deep SNNs, focusing on test accuracy and the number of spikes. As shown in Table 1, the proposed encoding method improves accuracy and reduces spike counts by up to 34% compared to the baseline. The deviation in the number of encoding spikes is drastically reduced, leading to a reduction in the variation of spike firing in the entire neural network. This suggests that enhancing the homeostasis in the spike encoding contributes to improved stability in deep SNNs. Through this section, we validate that the proposed homeostasis-based encoding can achieve higher accuracy with fewer spikes. Figs. 6 and 7 illustrate the spike feature maps produced by our method.

Noise Robustness Noise robustness in the encoding layer impacts neural network performance. Thus, we evaluated the noise robustness of the proposed encoding with input and integration noise, occurring before the encoding layer and in the neurons' membrane potential. We injected Gaussian noise $\mathcal{N}(0, \sigma)$ to input and membrane potential (u in Eq. 5). Fig. 2 demonstrates that our homeostasis-based encoding method, with VGG16 on CIFAR10, is robust to both types of noise, confirming that homeostasis improves training efficiency and noise robustness.

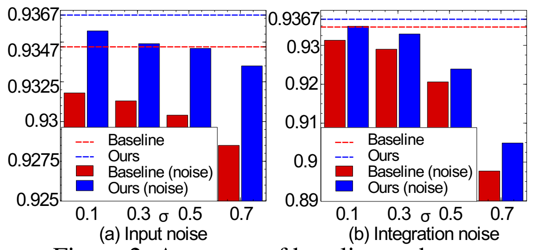

Figure 2: Accuracy of baseline and ours.

Compatibility Our encoding method is compatible with models and training algorithms from other studies, allowing for improved performance when applied to the latest models. To prove this, we experimentally demonstrated improvements in performance and efficiency by applying our encoding method to two latest models [18, 41]. For detailed experimental results, please refer to Tab. 5 and 6.

5.1 Ablation Studies

We conducted ablation studies with CIFAR10 on VGG16 and ResNet20 to determine how each proposed method affects the spike encoding and training performance. The cases included baseline, homeostasis (DFE), maximize the entropy of encoded feature (AT+FD), and proposed encoding (AT+FD+DFE). Results in Tab. 2 show that DFE improves accuracy and reduces spikes in VGG16. AT and FD enhance accuracy but reduce efficiency. The proposed method shows the most significant enhancement in accuracy with improved efficiency. There is a slight difference in trend in ResNet20, but our method improves performance and efficiency. Detailed results for ResNet20 are in Tab. 7.

Table 2: Ablation study results with and without the proposed method.

Methods	Accuracy (%)	# of total spikes (K)	# of encoded spikes (K)
Baseline	93.47±0.14	148±8.0	59±2.0
w/ DFE	93.53±0.11	**135**±**4.0**	**48**±1.0
w/ AT+FD	93.60±0.10	147±**4.0**	67±5.0
w/ AT+FD+DFE	**93.67**±**0.06**	144±6.0	52±**0.8**

To understand in more detail the impact of the proposed method on spike encoding, we investigated encoding patterns of neurons in the encoding layer. As shown in Fig. 3-(a), DFE reduces the proportion of neurons with low spike counts (1 or 2) and increases the proportion of non-spiking neurons. This shows that the dynamic selectivity proposed to improve homeostasis only affects low-firing neurons. In contrast, AT and FD increase spike diversity. Our proposed method expands the application of selective encoding, which is caused by homeostasis, to high-firing neurons with a spike count of three. This can be interpreted as an enhancement of the feature selectivity due to the maximization of the entropy of encoded features. As shown in Fig. 3-(b), DFE suppresses low-firing features, while entropy maximization promotes diverse features with balanced frequencies(Fig. 4-(c)). Compared to the baseline, our proposed method promotes low-firing features and prevents high-firing features (Fig. 3-(b)). Additionally, we measured the correlation between features [42], the average number of channels used for encoding, and the proportion of each channel type. For details, refer to Sec. A.7. Through these studies, we confirm that the proposed methods work together, enabling the encoding layer to learn efficient and effective spike encoding.

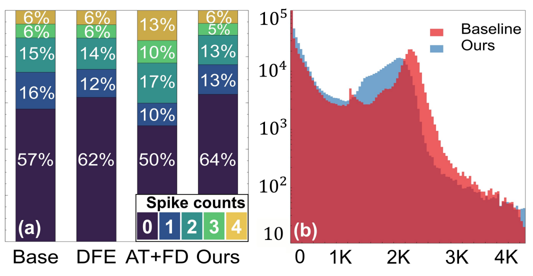

Figure 3: (a) Proportion and (b) distributions of spike counts of encoding neurons and channels, respectively (VGG16, CIFAR10).

5.2 Comparisons with Other Works

We compared the training performance with other related works to evaluate the proposed encoding method. For detailed comparisons, please refer to Tab. 8. We omitted the comparisons for spike count because most studies do not report them. While some encoding methods show lower accuracy with longer time steps, our method, which only modifies the baseline encoding, achieves performance comparable to SOTA-level studies using direct encoding.

6 Conclusion

In this work, we proposed H-direct, which is a homeostasis-aware direct spike encoding to improve the efficiency and effectiveness of encoding based on an analysis of conventional direct encoding. Experiments on several models and datasets showed that our method improved the efficiency and stability of spike encoding. In addition, we demonstrate that it is compatible with various models and performances through experiments. The proposed method demonstrates that a brain-inspired mechanism can improve the performance and efficiency of deep SNNs.

Acknowledgements

This work was supported in part by the Korea Institute of Science and Technology (KIST) through 2E32960 and the National Research Foundation of Korea (NRF) grant funded by the Korea government (Ministry of Science and ICT) [NRF-2021R1C1C2010454].

References

[1] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(3):1–45, 2024.

[2] Yuhuai Wu, Albert Qiaochu Jiang, Wenda Li, Markus Rabe, Charles Staats, Mateja Jamnik, and Christian Szegedy. Autoformalization with large language models. *Advances in Neural Information Processing Systems*, 35:32353–32368, 2022.

[3] Kaushik Roy, Akhilesh Jaiswal, and Priyadarshini Panda. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 575(7784):607–617, 2019.

[4] Kashu Yamazaki, Viet-Khoa Vo-Ho, Darshan Bulsara, and Ngan Le. Spiking neural networks and their applications: A review. *Brain Sciences*, 12(7):863, 2022.

[5] Christoph Ostrau, Christian Klarhorst, Michael Thies, and Ulrich Rückert. Benchmarking neuromorphic hardware and its energy expenditure. *Frontiers in neuroscience*, 16:873935, 2022.

[6] W. Maass. Networks of spiking neurons: the third generation of neural network models. *Neural Networks*, 10(9):1659–1671, 1997.

[7] Amirhossein Tavanaei, Masoud Ghodrati, Saeed Reza Kheradpisheh, Timothée Masquelier, and Anthony Maida. Deep learning in spiking neural networks. *Neural networks*, 111:47–63, 2019.

[8] 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.

[9] Yangfan Hu, Huajin Tang, and Gang Pan. Spiking deep residual networks. *IEEE Transactions on Neural Networks and Learning Systems*, 34(8):5200–5205, 2021.

[10] Yifan Hu, Lei Deng, Yujie Wu, Man Yao, and Guoqi Li. Advancing spiking neural networks toward deep residual learning. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–15, 2024.

[11] Seijoon Kim, Seongsik Park, Byunggook Na, and Sungroh Yoon. Spiking-yolo: spiking neural network for energy-efficient object detection. *Proceedings of the AAAI conference on artificial intelligence*, 34(07):11270–11277, 2020.

[12] Seijoon Kim, Seongsik Park, Byunggook Na, Jongwan Kim, and Sungroh Yoon. Towards fast and accurate object detection in bio-inspired spiking neural networks through bayesian optimization. *IEEE Access*, 9:2633–2643, 2020.

[13] Man Yao, Jiakui Hu, Zhaokun Zhou, Li Yuan, Yonghong Tian, Bo Xu, and Guoqi Li. Spike-driven transformer. *Advances in Neural Information Processing Systems*, 36, 2024.

[14] Han Zhang, Chenlin Zhou, Liutao Yu, Liwei Huang, Zhengyu Ma, Xiaopeng Fan, Huihui Zhou, and Yonghong Tian. Sglformer: Spiking global-local-fusion transformer with high performance. *Frontiers in Neuroscience*, 18:1371290, 2024.

[15] Zhaokun Zhou, Yuesheng Zhu, Chao He, Yaowei Wang, Shuicheng YAN, Yonghong Tian, and Li Yuan. Spikformer: When spiking neural network meets transformer. In *International Conference on Learning Representations*, 2023.

[16] Yujie Wu, Lei Deng, Guoqi Li, and Luping Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in neuroscience*, 12:323875, 2018.

[17] Hanle Zheng, Yujie Wu, Lei Deng, Yifan Hu, and Guoqi Li. Going deeper with directly-trained larger spiking neural networks. *Proceedings of the AAAI conference on artificial intelligence*, 35(12):11062–11070, 2021.

[18] Yufei Guo, Xiaode Liu, Yuanpei Chen, Liwen Zhang, Weihang Peng, Yuhan Zhang, Xuhui Huang, and Zhe Ma. Rmp-loss: Regularizing membrane potential distribution for spiking neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17391–17401, 2023.

[19] Nitin Rathi and Kaushik Roy. Diet-snn: A low-latency spiking neural network with direct input encoding and leakage and threshold optimization. *IEEE Transactions on Neural Networks and Learning Systems*, 34(6):3174–3182, 2021.

[20] Shikuang Deng, Yuhang Li, Shanghang Zhang, and Shi Gu. Temporal efficient training of spiking neural network via gradient re-weighting. In *International Conference on Learning Representations*, 2022.

[21] 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*, volume 33, pages 1311–1318, 2019.

[22] Yuhang Li, Yufei Guo, Shanghang Zhang, Shikuang Deng, Yongqing Hai, and Shi Gu. Differentiable spike: Rethinking gradient-descent for training spiking neural networks. *Advances in Neural Information Processing Systems*, 34:23426–23439, 2021.

[23] Shuang Lian, Jiangrong Shen, Qianhui Liu, Ziming Wang, Rui Yan, and Huajin Tang. Learnable surrogate gradient for direct training spiking neural networks. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 3002–3010, 2023.

[24] Daniel Auge, Julian Hille, Etienne Mueller, and Alois Knoll. A survey of encoding techniques for signal processing in spiking neural networks. *Neural Processing Letters*, 53(6):4693–4710, 2021.

[25] Rufin Van Rullen and Simon J Thorpe. Rate coding versus temporal order coding: what the retinal ganglion cells tell the visual cortex. *Neural computation*, 13(6):1255–1283, 2001.

[26] Jaehyun Kim, Heesu Kim, Subin Huh, Jinho Lee, and Kiyoung Choi. Deep neural networks with weighted spikes. *Neurocomputing*, 311:373–386, 2018.

[27] Seongsik Park, Seijoon Kim, Hyeokjun Choe, and Sungroh Yoon. Fast and efficient information transmission with burst spikes in deep spiking neural networks. In *Design Automation Conference*, pages 1–6, 2019.

[28] Seongsik Park, Seijoon Kim, Byunggook Na, and Sungroh Yoon. T2fsnn: deep spiking neural networks with time-to-first-spike coding. In *Design Automation Conference*, 2020.

[29] Iulia M Comsa, Krzysztof Potempa, Luca Versari, Thomas Fischbacher, Andrea Gesmundo, and Jyrki Alakuijala. Temporal coding in spiking neural networks with alpha synaptic function. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8529–8533. IEEE, 2020.

[30] Bing Han and Kaushik Roy. Deep spiking neural network: Energy efficiency through time based coding. In *European Conference on Computer Vision*, pages 388–404. Springer, 2020.

[31] Seongsik Park and Sungroh Yoon. Training energy-efficient deep spiking neural networks with time-to-first-spike coding. *arXiv*, 2021.

[32] Dominique Fernandes and Ana Luísa Carvalho. Mechanisms of homeostatic plasticity in the excitatory synapse. *Journal of neurochemistry*, 139(6):973–996, 2016.

[33] Wickliffe C Abraham, Barbara Logan, Jeffrey M Greenwood, and Michael Dragunow. Induction and experience-dependent consolidation of stable long-term potentiation lasting months in the hippocampus. *Journal of Neuroscience*, 22(21):9626–9634, 2002.

[34] Larry F Abbott and Sacha B Nelson. Synaptic plasticity: taming the beast. *Nature*, 3(11): 1178–1183, 2000.

[35] Kenneth D. Miller and David J. C. MacKay. The role of constraints in hebbian learning. *Neural computation*, 6(1):100–126, 1994.

[36] Gina G Turrigiano and Sacha B Nelson. Homeostatic plasticity in the developing nervous system. *Nature*, 5(2):97–107, 2004.

[37] Peter U Diehl and Matthew Cook. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in computational neuroscience*, 9:99, 2015.

[38] Hejia Geng and Peng Li. Hosnn: Adversarially-robust homeostatic spiking neural networks with adaptive firing thresholds. *arXiv*, 2023.

[39] Sungmin Hwang, Jeesoo Chang, Min-Hye Oh, Jong-Ho Lee, and Byung-Gook Park. Impact of the sub-resting membrane potential on accurate inference in spiking neural networks. *Scientific reports*, 10(1):3515, 2020.

[40] Sungmin Hwang, Jeesoo Chang, Min-Hye Oh, Kyung Kyu Min, Taejin Jang, Kyungchul Park, Junsu Yu, Jong-Ho Lee, and Byung-Gook Park. Low-latency spiking neural networks using pre-charged membrane potential and delayed evaluation. *Frontiers in Neuroscience*, 15:629000, 2021.

[41] Yufei Guo, Yuanpei Chen, Liwen Zhang, Xiaode Liu, Yinglei Wang, Xuhui Huang, and Zhe Ma. Im-loss: information maximization loss for spiking neural networks. *Advances in Neural Information Processing Systems*, 35:156–166, 2022.

[42] Gaojie Jin, Xinping Yi, Liang Zhang, Lijun Zhang, Sven Schewe, and Xiaowei Huang. How does weight correlation affect generalisation ability of deep neural networks? *Advances in Neural Information Processing Systems*, 33:21346–21356, 2020.

[43] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6023–6032, 2019.

[44] Wenjie Wei, Malu Zhang, Hong Qu, Ammar Belatreche, Jian Zhang, and Hong Chen. Temporal-coded spiking neural networks with dynamic firing threshold: Learning with event-driven backpropagation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10552–10562, 2023.

[45] Youngeun Kim and Priyadarshini Panda. Revisiting batch normalization for training low-latency deep spiking neural networks from scratch. *Frontiers in neuroscience*, 15:773954, 2021.

[46] Yufei Guo, Xinyi Tong, Yuanpei Chen, Liwen Zhang, Xiaode Liu, Zhe Ma, and Xuhui Huang. Recdis-snn: Rectifying membrane potential distribution for directly training spiking neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 326–335, 2022.

[47] Jibin Wu, Yansong Chua, Malu Zhang, Guoqi Li, Haizhou Li, and Kay Chen Tan. A tandem learning rule for effective training and rapid inference of deep spiking neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 34(1):446–460, 2021.

A Appendix / supplemental material

A.1 Leaky Integrate-and-Fire (LIF) Neuron Model

Leaky integrate-and-fire (LIF) neurons are widely used in deep SNNs due to their low computational cost. The integration process of LIF neurons can be described as

$$u_i^l[t] = 1/\tau(v_i^l[t-1] + \sum\nolimits_j w_{ij} s_j^{l-1}[t]), \tag{5}$$

where u, v, w, and s indicate the neuron's internal state, called membrane potential, intermediate state, synaptic weight, and input spike, respectively. The layer index is l, and the neuron indices are i and j. The time constant and time step are represented in τ and t, respectively. A spike is generated when the membrane potential exceeds the threshold as

$$s_i^l[t] = H(u_i^l[t] - V_{\text{th}}), \tag{6}$$

where H and V_{th} are the Heaviside step function and a threshold voltage, respectively. When a neuron fires a spike, its membrane potential is reset through the intermediate state, which can be stated as

$$v_i^l[t] = (u_i^l[t] - s_i^l[t])s_i^l[t] + u_i^l[t](1 - s_i^l[t]). \tag{7}$$

A.2 threshold dependent Batch Normalization (tdBN)

tdBN is a batch normalization method designed specifically for SNNs [17] and it can be expressed as follows:

$$x_c[t] = W \otimes s_c + B, \tag{8}$$

$$\hat{x_c} = \frac{\xi\, V_{th}(x_c - E[x_c])}{\sqrt{Var[x_c] + \epsilon}}, \tag{9}$$

$$y_c = \gamma_c \hat{x_c} + \beta_c, \tag{10}$$

where $x_c[t]$ represents the inputs at timestep t, x_c = ($x_c[1]$,$x_c[2]$,$\cdots$,$x_c[T]$), and $\hat{x}_c$ represents normalized x_c. ξ is a weight factor, and ϵ is a small positive number. γ_c and β_c are the scale and shift parameters at c-th channel, respectively. tdBN successfully adjusts the firing rate of the following spiking neurons, considering their thresholds.

A.3 Dynamic Feature Encoding Loss

The gradients of each parameter for the loss are presented as

$$\frac{\partial L_{\text{DFE}}}{\partial \beta_c} = \frac{\chi_c}{\|\chi_c - \alpha\|_2}\frac{1}{(\gamma_c + \epsilon)}, \quad \frac{\partial L_{\text{DFE}}}{\partial \gamma_c} = -\frac{\chi_c^2}{\|\chi_c - \alpha\|_2}\frac{1}{(\gamma_c + \epsilon)}, \tag{11}$$

where $\chi_c = \frac{\beta_c}{(\gamma_c + \epsilon)}$. The more detailed derivation is provided below.

Let $\chi_c = \frac{\beta_c}{(\gamma_c + \epsilon)}$,

$$\frac{\partial L_{\text{DFE}}}{\partial \chi_c} = \frac{\partial(\|\chi_c - \alpha\|_2)}{\partial \chi_c} = \frac{\chi_c}{\|\chi_c - \alpha\|_2}, \tag{12}$$

Thus,

$$\frac{\partial L_{\text{DFE}}}{\partial \beta_c} = \frac{\partial L_{\text{DFE}}}{\partial \chi_c}\frac{\partial \chi_c}{\partial \beta_c} = \frac{\chi_c}{\|\chi_c - \alpha\|_2}\frac{1}{(\gamma_c + \epsilon)}, \tag{13}$$

$$\frac{\partial L_{\text{DFE}}}{\partial \gamma_c} = \frac{\partial L_{\text{DFE}}}{\partial \chi_c}\frac{\partial \chi_c}{\partial \gamma_c} = -\frac{\chi_c^2}{\|\chi_c - \alpha\|_2}\frac{1}{(\gamma_c + \epsilon)}. \tag{14}$$

Table 3: Hyperparameters in experiments

Datasets	Architecture	η (Eq. 2)	α (Eq. 1)	λ_{CE} (Eq. 4)	λ_{FD} (Eq. 4)	λ_{DFE} (Eq. 4)
CIFAR10	VGG16	0.8	-1.0	1.0	5.00E-6	1.00E-4
	ResNet19	0.8	-0.3	1.0	3.00E-3	1.00E-4
	ResNet20	0.2	-0.4	1.0	3.00E-3	1.00E-3
CIFAR100	VGG16	0.8	-0.8	1.0	5.00E-6	1.00E-4
	ResNet19	0.8	-0.4	1.0	3.00E-3	1.00E-4
	ResNet20	0.8	-0.3	1.0	3.00E-3	1.00E-4
ImageNet	ResNet18	0.8	-1.0	1.0	3.00E-3	1.00E-3
CIFAR10-DVS	VGG16	0.8	-1.0	1.0	5.00E-6	1.00E-3

A.4 Feature Diversity Loss

The gradient of the encoding layer for the feature diversity loss can be stated as

$$\frac{\partial L_{\text{FD}}}{\partial W} \approx \sum_k -\log(p(x_k)-1)p'(x_k)\sum_t I[t]/\tau, \tag{15}$$

where I is the input. And more detailed derivation is provided below.

$$\frac{\partial L_{\text{FD}}}{\partial W} = \sum_k \left\{ \frac{\partial L_{\text{FD}}}{\partial p(x_k)} \frac{\partial p(x_k)}{\partial x_k} \sum_t \left(\frac{\partial x_k}{\partial s[t]} \frac{\partial s[t]}{\partial u[t]} \frac{\partial u[t]}{\partial W} \right) \right\} \tag{16}$$

$$\approx \sum_k -\log(p(x_k)-1)p'(x_k)\sum_t I[t]/\tau, \tag{17}$$

where I is the input, and $p(x)$ is a probability density function.

A.5 Experimental Settings

To evaluate the proposed encoding, we set a baseline method as STBP and tdBN, which are widely used training methods for deep SNNs. We experimented on both static datasets (CIFAR10/100, ImageNet) and a neuromorphic dataset (CIFAR10-DVS) with various model architectures including VGG16, ResNet18, ResNet19, and ResNet20. To ensure fair and accurate experiments, we repeated the experiment four times for each configuration and compared the results with the average value. As in other works, we use LIF neurons with a soft reset (Eq. 7) and set the time step to four.

Experimental Setup. The input size of the model is set to 32x32 for CIFAR10/100, 224x224 for ImageNet and 48x48 for CIFAR10-DVS. For CIFAR10/100 and CIFAR10-DVS, we trained each model for 300 epochs with SGD using a step-decay learning rate schedule (0.1 times every 100 epochs). The initial learning rate is 0.1 (0.01 for CIFAR10-DVS), and the optimizer includes L2 regularization with a lambda of 1e-4. For CIFAR10/100 and CIFAR10-DVS, the batch size was set to 100. While for ImageNet, it is set to 200 for 90 training epochs. For data augmentation, CutMix [43] was applied to static datasets, while random crop and random flip (horizontal and vertical) were used for the neuromorphic dataset. During training, the time step for all datasets was set to four. Additionally, the initial leak constant τ and threshold $V_{\text{th}}(0)$ are set to 1/0.9 and 0.5. The experiments were conducted using an NVIDIA A6000 GPU. The training times for 300 epochs were approximately 6-7 hours for VGG16 on CIFAR10 and CIFAR100 and about 20 hours for VGG16 on CIFAR10-DVS. For ResNet models, ResNet19 took around 23 hours on CIFAR10 and 20 hours on CIFAR100, while ResNet20 required roughly 12 hours on CIFAR10 and 11 hours on CIFAR100. For ImageNet, training ResNet18 for 90 epochs with 2 GPUs took about 8 days.

How to set hyper parameters? In our experiments, we used four types of hyperparameters: adjust rate η in AT, λ_{FD} that is the weight factor of L_{FD}, α that is a ratio of β/γ in tdBN of the encoding layer, and λ_{DFE} that is the weight factor of L_{DFE}. We set η empirically by varying the value between 0.1 and 1.0 in increments of 0.1. In addition, we derived α from the parameters (β and γ) of DSE channels in the baseline case. We set α to the average value of β/γ in the DSE channels. The weight

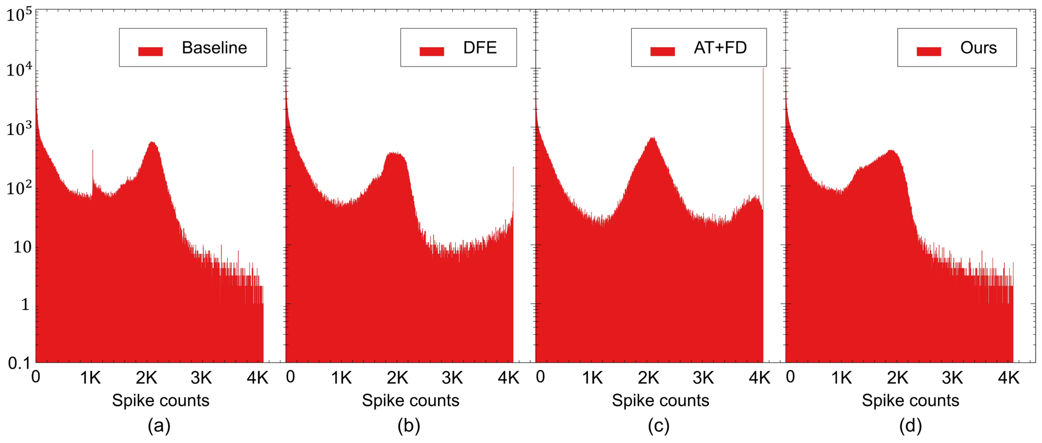

Figure 4: Distributions of each channel's spike counts on (a) baseline, (b) DFE, (c) AT+FD, and (d) ours (VGG16, CIFAR10)

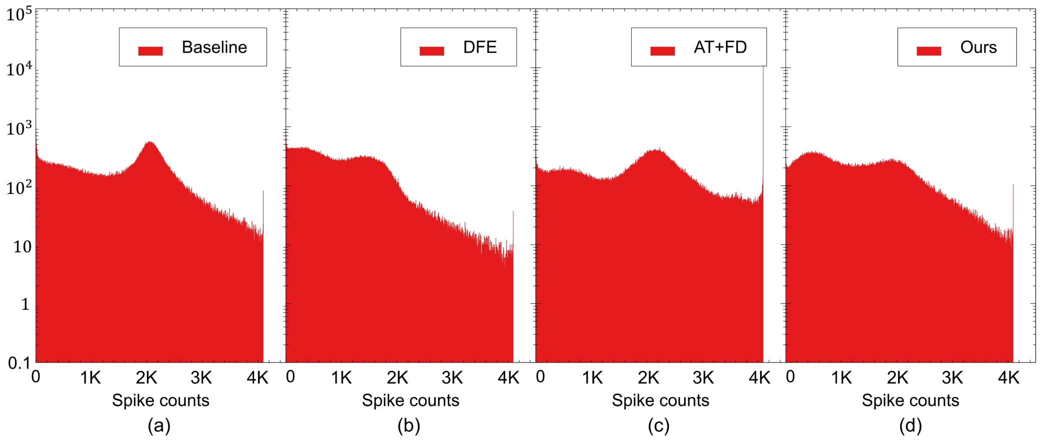

Figure 5: Distributions of each channel's spike counts on (a) baseline, (b) DFE, (c) AT+FD, and (d) ours (ResNet20, CIFAR10)

factors of total loss (λ_{CE}, λ_{FD}, and λ_{DFE}) also were determined empirically. The values of hyper parameters we used in this work are shown in Tab. 3.

A.6 Spike Counts Distributions of Each Channel

Fig. 4 and Fig. 5 show the distribution of each channels' spike counts during inference on CIFAR10 using VGG16 and ResNet20 architectures, respectively. Each histogram utilizes a logarithmic scale on the y-axis to elucidate data trends. First, in Fig. 4-(b), compared to Fig. 4-(a), there is an overall reduction in spike counts of channels, but the number of channels with spike counts greater than about 2K increases. For samples that do not require many fired channels, neurons of DSE channels exhibit reduced firing. This results in a decrease in the number of channels with spike counts less than about 2K. Conversely, in cases requiring more features, the neurons of DSE channels tend to fire, which leads to the increment of channels with spike counts above about 2K. In Fig. 4-(c), as a result of the influence of AT, the number of channels with high spike counts, especially around 4K, increases, deteriorating the encoding efficiency. Additionally, due to FD, the cumulative spike values tend to be diverse, resulting in a relatively even distribution compared to other cases. In Fig. 4-(d), by applying

Table 4: Ablation studies to compare variants of our method with and without AT (adaptive threshold), FD (feature diversity loss), and DFE (dynamic feature encoding loss). Data: CIFAR10.

Model	Methods	Cross-correlation	# of channels	Proportions (in %)			
				OFE	UFE	DSE	PE
VGG16	Baseline	0.231	51.55	0.39	10.16	50.00	39.06
	w/ DFE	0.203	56.77	0.00	0.00	66.80	33.20
	w/ AT+FD	0.274	57.83	0.39	4.30	50.00	45.31
	w/ AT+FD+DFE (ours)	0.218	**59.51**	0.00	0.00	62.89	37.11
ResNet20	Baseline	0.433	62.07	0.00	2.34	47.66	50.00
	w/ DFE	0.320	63.61	0.00	0.00	71.48	28.52
	w/ AT+FD	0.515	63.71	0.39	0.00	33.98	65.63
	w/ AT+FD+DFE (ours)	0.363	**63.85**	0.00	0.00	51.56	48.44

all the proposed methods, we can resolve the limitations observed in previous graphs. The results in Fig. 5 show similar trends to those in Fig. 4. In conclusion, the most efficient encoding is achieved when all the proposed methods are applied, resulting in fewer spikes and improved performance.

A.7 Cross Correlation

We measured the correlation between features as in [42]. The cross-correlation is defined as follows:

$$\rho(s_c) = \frac{1}{N_b N_c} \sum_{c=0}^{N_c} \sum_{b=0}^{N_b} \frac{|s'^{T}_{c,b} s'_{c,b}|}{\|s'_{c,b}\|_2 \|s'_{c,b}\|_2}, \tag{18}$$

where $\mathbf{s}_c \in \mathbb{R}^{f\times f\times N_c}$ represents the features in the c-th channel of the encoding layer. s'_c denotes the reshaped features of s_c into $\mathbf{s}'_c \in \mathbb{R}^{f^2\times N_c}$. N_b represents the b-th sample in the batch and $s'_{c,b}$ denotes s'_c for the b-th sample. A smaller cross-correlation value indicates fewer redundant features, suggesting more efficient encoding.

For further analysis, we measured the correlation between features [42], the average number of channels used for encoding, and the proportions of each channel type, which are presented in Tab. 4. When only DFE is applied, cross-correlation is the lowest, but the accuracy improvement is not significant due to the small number of channels used for encoding, which is caused by excessive feature selection. On the contrary, the proposed method improves training performance by utilizing the encoding layer's capacity properly, which indicates the highest number of channels used in encoding with lower correlation. The feature maps of ablations are presented in Sec. A.8. Through these studies, we can confirm that our method allows the encoding layer to learn how to encode spikes efficiently and effectively.

A.8 Feature Maps of Encoded Spikes

Fig. 6 and Fig. 7 show the encoded feature maps and the proportion of each categorized encoding from ablation studies using VGG16 and ResNet20 architectures on CIFAR10 dataset, respectively. In the case of VGG16 (Fig. 6), OFE, UFE, DSE, and PE appear in the baseline, whereas in the model to which DFE is applied, OFE and UFE disappear and the proportion of DSE increases about 16.8% compared to the baseline. In addition, in the model with AT and FD, it can be seen that UFE decreases by about 5.9% and PE increases by about 12.1%. In Ours, OFE and UFE disappear, and DSE accounts for the largest portion among all types (about 62.9%). In the case of ResNet20 (Fig. 7), UFE, DSE, and PE appear in baseline. When DFE is applied, UFE disappears, and the proportion of DSE increases by about 23.8%. In addition, PE increases by about 15.6% in the model with AT and FD. In ours, UFE disappears, and DSE accounts for the largest portion (about 51.6%) of all types. The result tends to be similar to VGG16, which indicates the effectiveness and compatibility of our methods across model architectures.

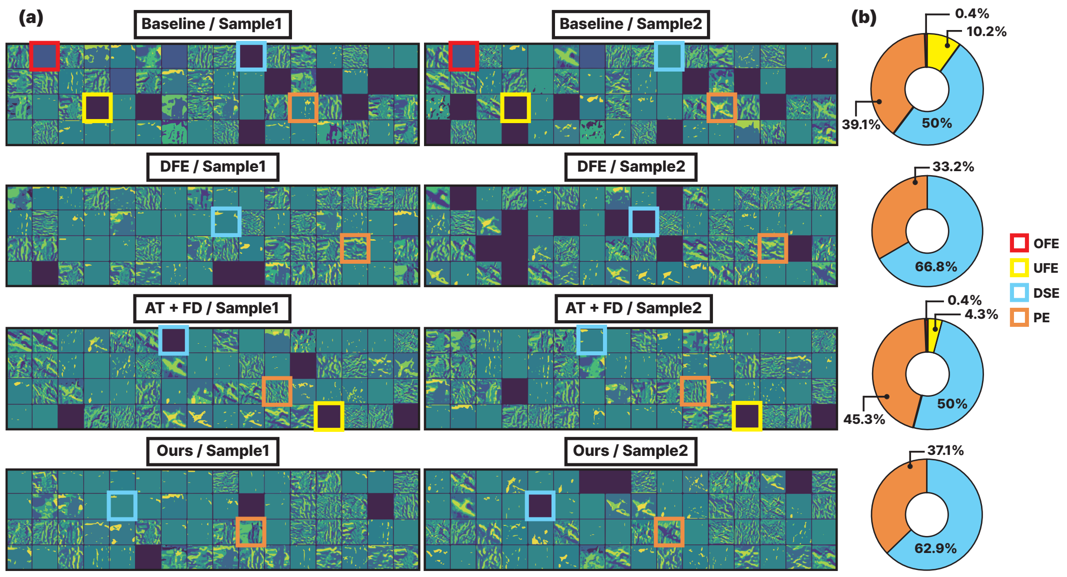

Figure 6: (a) Examples of encoded feature maps and (b) the average proportion of each categorized encoding in ablation studies (VGG16, CIFAR10).

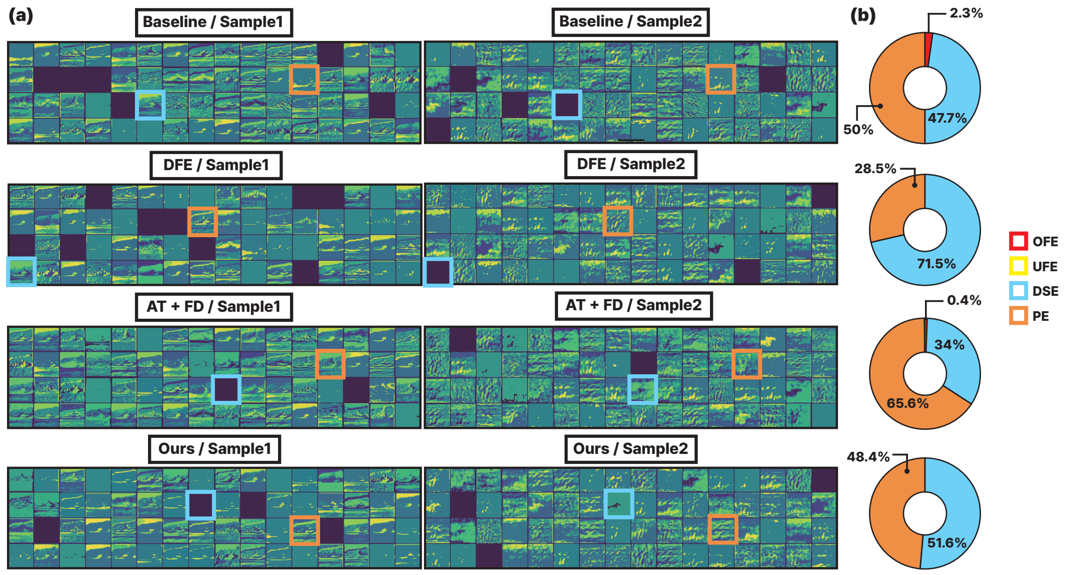

Figure 7: (a) Examples of encoded feature maps and (b) the average proportion of each categorized encoding in ablation studies (ResNet20, CIFAR10).

Table 5: Comparisons of accuracy and the number of spikes between baseline and our model with IM-Loss (* is our implementation)

Datasets	Architectures	Methods	Time steps	Accuracy (%)	# of total spikes (K)
CIFAR10	VGG16	IM-Loss [41]	5	93.85±0.11	-
		IM-Loss*	4	93.73±0.03	142±1
		IM-Loss + H-Direct	4	93.89±0.01	137±6
	ResNet19	IM-Loss [41]	4	95.40±0.08	-
		IM-Loss*	4	95.76±0.06	1116±13
		IM-Loss + H-Direct	4	95.78±0.14	1075±26
	ResNet20	IM-Loss*	4	95.16±0.13	752±18
		IM-Loss + H-Direct	4	95.23±0.06	646±7
CIFAR100	VGG16	IM-Loss [41]	5	70.18±0.09	-
		IM-Loss*	4	69.68±0.05	174±3
		IM-Loss + H-Direct	4	69.74±0.17	161±2
	ResNet19	IM-Loss*	4	76.94±0.11	1309±11
		IM-Loss + H-Direct	4	77.15±0.23	1284±9
	ResNet20	IM-Loss*	4	74.94±0.16	797±8
		IM-Loss + H-Direct	4	75.41±0.08	764±2
CIFAR10-DVS	ResNet19	IM-Loss [41]	10	72.60±0.08	-
	VGG16	IM-Loss*	4	75.48±0.56	423±2
		IM-Loss + H-Direct	4	75.68±0.33	280±2

Table 6: Comparisons of accuracy and the number of spikes between baseline and our model with RMP-Loss (* is our implementation)

Datasets	Architectures	Methods	Time steps	Accuracy (%)	# of total spikes (K)
CIFAR10	VGG16	RMP-Loss [18]	4	93.33±0.07	-
		RMP-Loss *	4	93.59±0.03	155±2
		RMP-Loss + H-Direct	4	93.69±0.05	133±7
	ResNet19	RMP-Loss [18]	4	95.51±0.08	-
		RMP-Loss *	4	95.23±0.13	963±29
		RMP-Loss + H-Direct	4	95.30±0.08	955±26
	ResNet20	RMP-Loss [18]	4	91.89±0.05	-
		RMP-Loss *	4	94.77±0.07	619±14
		RMP-Loss + H-Direct	4	94.79±0.10	586±20
CIFAR100	VGG16	RMP-Loss [18]	4	72.55±0.08	-
		RMP-Loss *	4	69.35±0.13	180±2
		RMP-Loss + H-Direct	4	69.49±0.14	157±2
	ResNet19	RMP-Loss [18]	4	78.28±0.1	-
		RMP-Loss *	4	76.13±0.08	1147±13
		RMP-Loss + H-Direct	4	76.43±0.07	1104±8
	ResNet20	RMP-Loss [18]	4	66.65±0.10	-
		RMP-Loss *	4	74.38±0.16	715±5
		RMP-Loss + H-Direct	4	74.60±0.25	703±7
CIFAR10-DVS	ResNet19	RMP-Loss [18]	10	76.20±0.20	-
	ResNet20	RMP-Loss [18]	10	75.60±0.30	-
	VGG16	RMP-Loss *	4	75.15±0.86	423±1
		RMP-Loss + H-Direct	4	75.23±0.49	295±1

Table 7: Ablation studies on accuracy and the number of spikes to compare variants of our method with and without the proposed method. Data: CIFAR10

Architecture	Methods	Accuracy (%)	# of total spikes (K)	# of encoded spikes (K)
ResNet20	Baseline	94.99±**0.02**	480±11.0	92±1.0
	w/ DFE	94.83±**0.02**	470±15.0	**62**±1.0
	w/ AT+FD	94.89±**0.02**	507±**1.0**	115±3.0
	w/ AT+FD+DFE (ours)	**95.09**±0.04	**463**±5.5	83±**0.9**

Table 8: Comparisons with other works on various models and datasets. *Abbr.*: H-Direct: Homeostasis-aware direct encoding (ours), Direct: Direct encoding, Phase: Phase encoding, TTFS: Time-to-first-spike encoding, Rate: Rate encoding.

Datasets	Architecture	Method	Input encoding	Time steps	Accuracy
CIFAR10	VGG16	ANN2SNN [27]	Phase	1500	91.41%
		T2FSNN [28]	TTFS	680	91.43%
		DTA-TTFS [44]	TTFS	160	93.05%
		BNTT [45]	Rate	25	90.05%
		Diet-SNN [19]	Direct	5	92.70%
		RMP-Loss [18]	Direct	4	93.33%±0.07
		Ours	H-Direct	4	93.67%±0.01
	ResNet19	STBP-tdBN [17]	Direct	4	92.92%
		TET [20]	Direct	4	94.44%±0.08
		RecDis-SNN [46]	Direct	4	95.53%±0.05
		RMP-Loss [18]	Direct	4	95.51%±0.08
		Ours	H-Direct	4	95.72%±0.20
	ResNet20	Diet-SNN [19]	Direct	5	91.78%
		RMP-Loss [18]	Direct	4	91.89%±0.05
		Ours	H-Direct	4	95.09%±0.04
CIFAR100	VGG16	ANN2SNN [27]	Phase	1500	68.69%
		T2FSNN [28]	TTFS	680	68.79%
		DTA-TTFS [44]	TTFS	160	69.66%
		BNTT [45]	Rate	50	66.60%
		Diet-SNN [19]	Direct	5	69.97%
		RecDis-SNN [46]	Direct	4	69.88%±0.08
		RMP-Loss [18]	Direct	4	72.55%±0.08
		Ours	H-Direct	4	69.29%±0.05
	ResNet19	TET [20]	Direct	4	74.47%±0.15
		RecDis-SNN [46]	Direct	4	74.10%±0.13
		RMP-Loss [18]	Direct	4	78.28%±0.10
		Ours	H-Direct	4	77.07%±0.10
	ResNet20	Diet-SNN [19]	Direct	5	64.07%
		RMP-Loss [18]	Direct	4	66.65%±0.10
		Ours	H-Direct	4	75.13%±0.12
ImageNet	ResNet34	STBP-tdBN [17]	Direct	6	63.72%
	ResNet18	RMP-Loss [18]	Direct	4	63.03%±0.07
		Ours	H-Direct	4	64.30%±0.03
CIFAR10-DVS	CifarNet	Tandem learning [47]	Direct	20	65.59%
	ResNet19	STBP-tdBN [17]	Direct	10	67.80%
	VGG16	Ours	H-Direct	4	76.15%±0.31