

SpikingNeRF: Making Bio-inspired Neural Networks See through the Real World

Xingting Yao^{1,2*}, Qinghao Hu^{1*}, Fei Zhou³, Tielong Liu^{1,2},
 Zitao Mo¹, Zeyu Zhu^{1,2}, Zhengyang Zhuge¹, Jian Cheng^{1†}

¹Institute of Automation, Chinese Academy of Sciences

²School of Future Technology, University of Chinese Academy of Sciences

³China Electric Power Research Institute Co., Ltd

{yaoxingting2020, huqinghao2014, jian.cheng}@ia.ac.cn,

Abstract

In this paper, we propose SpikingNeRF, which aligns the temporal dimension of spiking neural networks (SNNs) with the radiance rays, to seamlessly accommodate SNNs to the reconstruction of neural radiance fields (NeRF). Thus, the computation turns into a spike-based, multiplication-free manner, reducing energy consumption and making high-quality 3D rendering, for the first time, accessible to neuromorphic hardware. In SpikingNeRF, each sampled point on the ray is matched to a particular time step and represented in a hybrid manner where the voxel grids are maintained as well. Based on the voxel grids, sampled points are determined whether to be masked out for faster training and inference. However, this masking operation also incurs irregular temporal length, making it intractable for hardware processors, e.g., GPUs, to conduct parallel training. To address this problem, we develop the temporal padding strategy to tackle the masked samples to maintain regular temporal length, i.e., regular tensors, and further propose the temporal condensing strategy to form a denser data structure for hardware-friendly computation. Experiments on various datasets demonstrate that our method can reduce energy consumption by an average of 70.79% and obtain comparable synthesis quality with the ANN baseline. Verification on the neuromorphic hardware accelerator also shows that SpikingNeRF can further benefit from neuromorphic computing over the ANN baselines on energy efficiency. Codes and the appendix are in <https://github.com/Ikarosy/SpikingNeRF-of-CASIA>.

Introduction

Spiking neural networks (SNNs) are considered the third generation of neural networks, and their bionic modeling encourages much research attention to explore the prospective bio-plausible intelligence that features multitasking and extreme energy efficiency as the human brain does (Maass 1997; Roy, Jaiswal, and Panda 2019). While much dedication has been devoted to SNN research, the gap between the expectation of SNN boosting a wider range of intelligent tasks and the fact of artificial neural networks (ANNs) dominating deep learning in the majority of tasks still exists.

Recently, more research interests have been invested in narrowing the gap and have promoted notable milestones in various tasks, including image classification (Zhou et al.

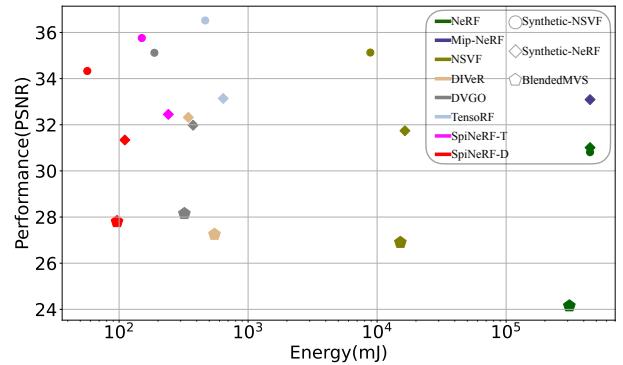


Figure 1: Comparisons of our SpikingNeRF with other NeRF-based works in synthesis quality and model rendering energy. Different colors represent different works, and our SpikingNeRF with two different frameworks are denoted in red and violet, respectively. A detailed notation explanation is specified in the Experiments section. Different testing datasets are denoted by different shapes.

2022), object detection (Zhang et al. 2022), graph prediction (Zhu et al. 2022b), natural language processing (Zhu, Zhao, and Eshraghian 2023), etc. Besides multi-task supporting, SNN research is also thriving in performance lifting and energy efficiency exploration at the same time.

However, we have not yet witnessed the establishment of SNN in the real 3D reconstruction task with advanced performance. Let alone enabling the high-quality real 3D rendering on neuromorphic hardwares, e.g., Loihi (Davies et al. 2018), PTB (Lee, Zhang, and Li 2022), and SpikeSim (Moitra et al. 2023), where neuromorphic computing can essentially acquire low-energy consumption. Meanwhile, high-quality real 3D reconstruction, specifically for NeRF (Mildenhall et al. 2021), has suffered huge computation overhead during rendering, consuming a significant magnitude of energy (Garbin et al. 2021). Naturally, this raises a question: *could bio-inspired spiking neural networks reconstruct the real 3D scene with advanced quality at low energy consumption?* This paper investigates the rendering of neural radiance fields with a spiking approach to answer the question.

We propose SpikingNeRF to reconstruct volumetric scene

*Equal contribution.

†Corresponding author.

representations of neural radiance fields. For fast synthesis, voxel grids methods (Hedman et al. 2021; Liu et al. 2022; Sun, Sun, and Chen 2022) are considered to explicitly store the volumetric parameters. For efficient computation, the spiking multilayer perceptron (sMLP) is utilized to implicitly yield volumetric information in an addition-only and spike-driven approach. With such an explicit-and-implicit hybrid, fast and energy-efficient neural radiance rendering becomes feasible.

Inspired by the imaging process of the primate fovea in the retina that accumulates the intensity flow over time to stimulate the photoreceptor cell (Masland 2012; Wässle 2004), we take one step forward to associate the accumulation process of rendering with the temporal accumulation process of SNNs, which ultimately stimulates the spiking neurons to fire. Concretely, we align the radiance ray with the temporal dimension of the sMLP, and individually match each sampled point on the ray to a time step during rendering. Thus, the geometric consecutiveness of the ray is transformed into the temporal continuity of the SNN. As a result, SpikingNeRF seizes the nature of both worlds to make the NeRF rendering in a spiking manner, and brings about a novel and effective data encoding approach for SNN-based 3D rendering. Different from other SNN-based image reconstructions (Zhu et al. 2022a; Mei et al. 2023), which focus on the event-based gray 2D reconstruction, we are the first to explore the reconstruction of the real RGB world with SNNs.

Moreover, since the number of sampled points on different rays always varies, the temporal lengths of different rays become irregular. Consequently, the querying for the volumetric information can hardly be parallelized during rendering, severely hindering the training process on *graphics processing units (GPUs)*. To solve this issue, we first investigate the temporal padding (TP) method to attain the regular temporal length in a querying batch, i.e., a regular-shaped tensor, thus ensuring parallelism and GPU training feasible. Furthermore, we propose the temporal condensing-and-padding (TCP), to fully constrain the tensor size and condense the data distribution, which is hardware-friendly to *neuromorphic hardware accelerators and GPUs*. Our thorough experimentation proves that TCP can maintain both the energy merits of SNNs and the high quality of NeRF rendering as shown in Fig. 1.

To sum up, our main contributions are as follows:

- We propose SpikingNeRF that aligns the temporal dimension of SNNs with the radiance rays of NeRF, exploiting the temporal characteristics of SNNs. *To the best of our knowledge, this is the first work to accommodate SNNs to reconstructing 3D scenes, making high-quality 3D rendering feasible on the neuromorphic hardware.*
- We propose TP and TCP to solve the irregular temporal lengths, ensuring the training and inference parallelism on GPUs. TCP can also further keep SpikingNeRF hardware-friendly to the neuromorphic hardware.
- Our experiments demonstrate the effectiveness of SpikingNeRF on four mainstream tasks, achieving advanced energy efficiency as shown in Fig. 1. For another specific

example, SpikingNeRF-D can achieve 72.95% energy reduction with a 0.33 PSNR drop on Tanks&Temples.

In order to avoid misunderstanding, we additionally cite (Liao et al. 2023) which shares the same title as ours. Their work, based on the ANN implementation, essentially builds a non-linear and non-spiking function named B-FIF to post-process the particular density-related output of the original ANN-based NeRF. Overall, they do not use SNNs, do not aim at the neuromorphic hardware, focus on geometric reconstruction, and report only on the Chamfer metric. So, it is rational to deem that (Liao et al. 2023) is irrelevant and our work is impossible to conduct quantitative comparisons with theirs. Different from the above work, we are the first to explore the rendering of real RGB with SNNs and benefit NeRF rendering from neuromorphic computing.

Preliminaries

Neural radiance field. To reconstruct the scene for the given view, NeRF (Mildenhall et al. 2021) first utilizes an MLP, which takes in the location coordinates $\mathbf{p} \in \mathbb{R}^3$ and the view direction $\mathbf{v} \in \mathbb{R}^2$ and yields the density $\sigma \in \mathbb{R}$ and the color $\mathbf{c} \in \mathbb{R}^3$, to implicitly maintain continuous volumetric representations:

$$\mathbf{e}, \sigma = \text{MLP}_\theta(\mathbf{p}), \quad (1)$$

$$\mathbf{c} = \text{MLP}_\gamma(\mathbf{e}, \mathbf{v}), \quad (2)$$

where θ and γ denote the parameters of the separate two parts of the MLP, and \mathbf{e} is the embedded features. Next, NeRF renders the pixel of the expected scene by casting a ray \mathbf{r} from the camera origin point to the direction of the pixel, then sampling K points along the ray. Through querying the MLP as in Eq. (1-2) K times, K color values and K density values can be retrieved. Finally, following the principles of the discrete volume rendering proposed in (Max 1995), the expected pixel RGB $\hat{\mathbf{C}}(\mathbf{r})$ can be rendered:

$$\alpha = 1 - \exp(-\sigma_i \delta_i), \quad w_i = \prod_{j=1}^{i-1} (1 - \alpha_j), \quad (3)$$

$$\hat{\mathbf{C}}(\mathbf{r}) \approx \sum_{i=1}^K w_i \alpha_i \mathbf{c}_i, \quad (4)$$

where \mathbf{c}_i and σ_i denotes the color and the density values of the i -th point respectively, and δ_i is the distance between the adjacent point i and $i + 1$.

After rendering all the pixels, the expected scene is reconstructed. With the ground-truth pixel color $C(\mathbf{r})$, the parameters of the MLP can be trained end-to-end by minimizing the MSE loss:

$$\mathcal{L} = \frac{1}{|\mathcal{R}|} \sum_{\mathbf{r} \in \mathcal{R}} \|\hat{\mathbf{C}}(\mathbf{r}) - C(\mathbf{r})\|_2^2, \quad (5)$$

where \mathcal{R} is the mini-batch containing the sampled rays.

Hybrid volumetric representation. The number of sampled points K in Eq. 4 is usually big, leading to the heavy MLP querying burden as displayed in Eq. (1-2). To alleviate this problem, voxel grid representation is utilized to contain the volumetric parameters directly, e.g., the embedded

feature e and the density σ in Eq. 1, as the values of the voxel grid. Thus, querying the MLP in Eq. 1 is substituted to querying the voxel grids and operating the interpolation, which is much easier:

$$\sigma = \text{act}(\text{interp}(\mathbf{p}, \mathbf{V}_\sigma)), \quad (6)$$

$$\mathbf{e} = \text{interp}(\mathbf{p}, \mathbf{V}_f), \quad (7)$$

where \mathbf{V}_σ and \mathbf{V}_f are the voxel grids related to the volumetric density and features, respectively. “interp” denotes the interpolation operation, and “act” refers to the activation function, e.g., ReLU or the shifted softplus (Barron et al. 2021).

Furthermore, those irrelevant points with low density or unimportant points with low weight can be **masked** through predefined thresholds λ , then Eq. 4 turns into:

$$A \triangleq \{i : w_i > \lambda_1, \alpha_i > \lambda_2\}, \quad (8)$$

$$\hat{C}(\mathbf{r}) \approx \sum_{i \in A} w_i \alpha_i \mathbf{c}_i. \quad (9)$$

Thus, the queries of the MLP for sampled points in Eq. 2 are significantly reduced. With such computational benefits, hybrid volumetric representation is prevalent in neural radiance rendering (Sun, Sun, and Chen 2022; Chen et al. 2022). **Spiking neuron.** The spiking neuron is the most fundamental unit in spiking neural networks, which essentially differs SNNs from ANNs. The modeling of spiking neurons commonly adopts the leaky integrate-and-fire (LIF) model:

$$\mathbf{U}^t = \mathbf{V}^{t-1} + \frac{1}{\tau} (\mathbf{X}^t - \mathbf{V}^{t-1} + V_{reset}), \quad (10)$$

$$\mathbf{S}^t = \mathbb{H}(\mathbf{U}^t - V_{th}), \quad (11)$$

$$\mathbf{V}^t = \mathbf{U}^t \odot (1 - \mathbf{S}^t) + V_{reset} \mathbf{S}^t. \quad (12)$$

Here, we follow the renowned SpikingJelly (Fang et al. 2023) to implement the LIF neurons. \odot denotes the Hadamard product. \mathbf{U}^t is the intermediate membrane potential at time-step t and can be updated through Eq. 10, where \mathbf{V}^{t-1} is the actual membrane potential at time-step $t-1$ and \mathbf{X}^t denotes the input vector at time-step t , e.g., the activation vector from the previous layer in MLPs. The output spike vector \mathbf{S}^t is given by the Heaviside step function $\mathbb{H}(\cdot)$ in Eq. 11, indicating that a spike is fired when the membrane potential exceeds the potential threshold V_{th} . Dependent on whether the spike is fired at time-step t , the membrane potential \mathbf{V}^t is set to \mathbf{U}^t or the reset potential V_{reset} through Eq. 12.

Since the Heaviside step function $\mathbb{H}(\cdot)$ is not differentiable, the surrogate gradient method is utilized to solve this issue, which is defined as :

$$\frac{d\mathbb{H}(x)}{dx} = \frac{1}{1 + \exp(-\alpha x)}, \quad (13)$$

where α is a predefined hyperparameter. Thus, spiking neural networks can be optimized end-to-end.

Methodology

Data encoding

In this subsection, we explore two naive data encoding approaches for converting the input data to SNN-tailored formats, i.e., direct-encoding and Poisson-encoding. Both of

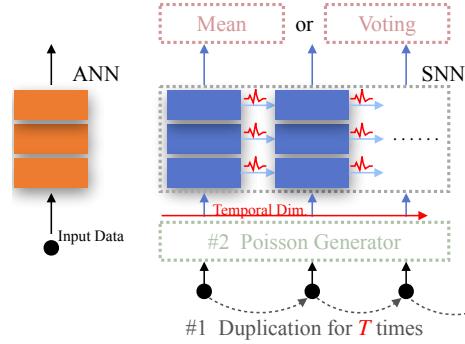


Figure 2: Conventional data encoding schemes. For direct-encoding, only the operation #1 is necessary that it duplicates the input data T times to fit the length of the temporal dimension. For Poisson-encoding, both operation #1 and #2 are utilized to generate the input spike train. The “Mean” or “Voting” operation is able to decode the SNN output.

them are proven to perform well in the direct learning of SNNs (Fang et al. 2021; Han, Srinivasan, and Roy 2020; Shrestha and Orchard 2018; Cheng et al. 2020).

As described in Preliminaries, spiking neurons receive data with an additional dimension called the temporal dimension, which is indexed by the time-step t in Eq. (10-12). Consequently, original ANN-formatted data need to be encoded to fill the temporal dimension as illustrated in Fig. 2. In the direct-encoding scheme, the original data is duplicated T times to fill the temporal dimension, where T represents the total length of the temporal dimension. As for the Poisson-encoding scheme, besides the duplication operation, it perceives the input value as the probability and generates a spike according to the probability at each time step. Additionally, a decoding method is entailed for the subsequent operations of rendering, and the mean (Li et al. 2021) and the voting (Fang et al. 2021) decoding operations are commonly considered. We employ the former approach since the latter one is designed for classification tasks (Diehl and Cook 2015; Wu et al. 2019).

Thus, with the above two encoding methods, we build two naive versions of SpikingNeRF and are able to conduct experiments on various datasets to verify the feasibility.

Time-ray alignment (TRA)

This subsection further explores the potential of accommodating the SNN to the NeRF rendering process in a more natural and novel way, where we attempt to retain the real-valued input data as direct-encoding does, but do not fill the temporal dimension with the duplication-based approach.

We first consider the MLP querying process in the ANN philosophy. For an expected scene to reconstruct, the volumetric parameters of sampled points, e.g., e and v in Eq. 2, are packed as the input data with the shape of $[batch, c_e]$ or $[batch, c_v]$, where $batch$ represents the sample index and c is the channel index of the volumetric parameters. Thus, the MLP can query these data and output the corresponding color information in parallel. However, from the geometric

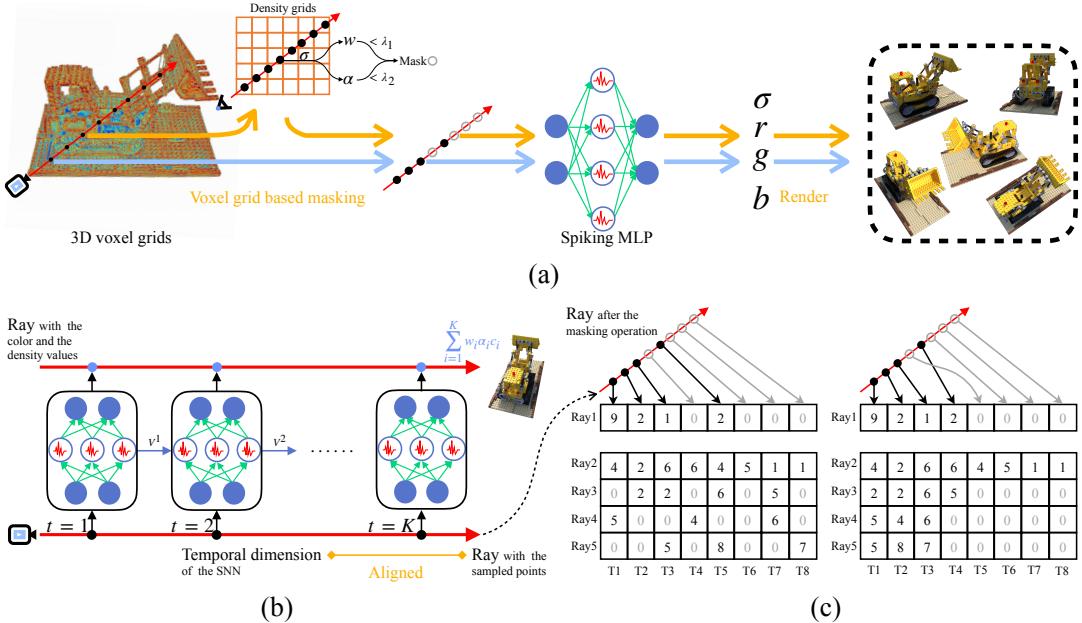


Figure 3: Overview of the proposed SpikingNeRF. (a) The rendering process of SpikingNeRF. The whole 3D volumetric parameters are stored in the voxel grids. The irrelevant or unimportant samples are masked before the sMLP querying. The expected scenes are rendered with the volumetric information yielded by the sMLP. (b) Alignment between the temporal dimension and the ray. The sMLP queries each sampled point step-by-step to yield the volumetric information. (c) Proposed temporal padding (left) and temporal condensing-and-padding (right) methods. For simplification, the channel length of the volumetric parameters is set to 1.

view, the input data should be packed as $[ray, pts, c]$, where ray is the ray index and the pts is the index of the sampled points.

Obviously, the ANN-based MLP querying process can not reflect such geometric relations between the ray and the sampled points. Then, we consider the computation modality of SNNs. As illustrated in Eq. (10-12), SNNs naturally entail the temporal dimension to process the sequential signals. This means a spiking MLP naturally accepts the input data with the shape of $[batch, time, c]$, where $time$ is the temporal index. Therefore, we can reshape the volumetric parameters back to $[ray, pts, c]$, and intuitively match each sample along the ray to the corresponding time step:

$$\begin{aligned} \text{Input}_{MLP} &:= [batch, c] \\ &\Rightarrow [ray, pts, c] \\ &\Rightarrow [batch, time, c] := \text{Input}_{sMLP}, \end{aligned} \quad (14)$$

which is also illustrated in 3(b). Such an alignment does not require any input data pre-process such as duplication (Zhou et al. 2022) or Poisson generation (Garg, Chowdhury, and Roy 2021) as prior arts commonly do.

Temporal condensing-and-padding (TCP)

The masking operation on sampled points, as illustrated in Preliminaries, makes the time-ray alignment intractable. Although such masking operation improves the rendering speed and quality by curtailing the computation cost of redundant samples, it also causes the number of queried sam-

ples on different rays to be irregular, which indicates the reshape operation of Eq. 14, i.e., shaping into a tensor, is unfeasible on GPUs after the masking operation.

To ensure computation parallelism on GPUs, we first propose to retain the indices of those masked samples but discard their values. As illustrated in Fig. 3(c) Left, we arrange both unmasked and masked samples sequentially to the corresponding ray -indexed vector, and pad zeros to the vacant tensor elements. Such that, a regular-shaped input tensor is built, *making GPU training feasible*. We refer to this simple approach as the temporal padding (**TP**) method.

However, TP does not handle those masked samples effectively because those padded zeros will still get involved in the following computation and cause the membrane potential of sMLP to decay, implicitly affecting the outcomes of the unmasked samples in the posterior segment of the ray. Even for a sophisticated hardware accelerator that can skip those zeros, the sparse data structure still causes computation inefficiency such as imbalanced workload (Zhang et al. 2020). To solve this issue, we design the temporal condensing-and-padding (**TCP**) scheme, which is illustrated in Fig. 3(c) Right. Different from TP, TCP completely discards the parameters and indices of the masked samples, and adjacently rearranges the unmasked sampled points to the corresponding ray vector. For the vacant tensor elements, zeros are filled as TP does. Consequently, valid data is condensed to the left side of the tensor. Notably, the ray dimension can be sorted according to the valid data number to further increase the density. As a result, TCP has fully elim-

Algorithm 1: Overall algorithm of the DVGO-based SpikingNeRF (SpikingNeRF-D) in the rendering process.

Input: The density and the feature voxel grids V_σ and V_f , the spiking MLP $sMLP(\cdot)$, the view direction of the camera v , the rays from the camera origin to the directions of N pixels of the expected scene $R_{\{N\}} = \{r_1, r_2, \dots, r_N\}$, the number of the sampled points per ray M , the ground-truth RGB $\mathbf{C} = \{C_1, C_2, \dots, C_N\}$ of the expected N pixels.

Output: The expected RGB $\hat{\mathbf{C}} = \{\hat{C}_1, \hat{C}_2, \dots, \hat{C}_N\}$ of the expected N pixels, the training loss \mathcal{L} .

- 1: The coordinates of sampled points $P_{\{N \times M\}} = \{p_{1,1}, p_{1,2}, \dots, p_{N,M}\} \leftarrow Sample(R)$.
- 2: $\alpha_{\{N \times M\}}, w_{\{N \times M\}} \leftarrow Weigh(P, V_\sigma)$ as in Eq. 6 and Eq. 3.
- 3: Filtered coordinates $P' \leftarrow Mask(P, \alpha, w)$ as in Eq. 8.
- 4: $Input_{MLP} \leftarrow ExtractFeatures(P', V_f, v)$ as described in Eq. 7 and Eq. 2.
- 5: The temporal length $T \leftarrow$ The maximum point number among the batched rays.
- 6: $Input_{sMLP} \leftarrow$ The **TP or TCP** transformation on $Input_{MLP}$ as described in Eq. 14 and Sec. TCP.
- 7: The RGB values $c_{\{N,T\}} \leftarrow sMLP(Input_{sMLP})$
- 8: $\hat{\mathbf{C}} \leftarrow Accumulate(P', \alpha, w, c)$ as in Eq. 9.
- 9: $\mathcal{L} \leftarrow MSE(\mathbf{C}, \hat{\mathbf{C}})$ as in Eq. 5

Note: Dependent on the specific NeRF framework, the functions, e.g., $Sample(\cdot)$, $Mask(\cdot)$, may be different.

inated the impact of the masked samples and made SpikingNeRF more hardware-friendly.

Although such data condensing operation can incur extra overhead on hardware, *a regular and condensed data structure commonly brings far more benefits to efficiency, covering the cost* (Zhang et al. 2020). Such benefits not only cater for DNN hardware accelerators and GPUs, but also apply to neuromorphic hardware, as proposed and proved in PTB (Lee, Zhang, and Li 2022) and STELLAR (Mao et al. 2024). Therefore, we choose *TCP as our mainly proposed method*.

Overall algorithm

This section summarizes the overall algorithm of SpikingNeRF based on the DVGO (Sun, Sun, and Chen 2022) framework. And, the pseudo code is given in Algorithm 1.

As illustrated in Fig. 3(a), SpikingNeRF first establishes the voxel grids filled with learnable volumetric parameters. In the case of the DVGO implementation, two groups of voxel grids are built as the input of Algorithm 1, which are the density and the feature voxel grids. Given an expected scene with N pixels to render, Step 1 is to sample M points along each ray shot from the camera origin to the direction of each pixel. With the $N \times M$ sampled points, Step 2 queries the density grids to compute the weight coefficients, and Step 3 uses these coefficients to mask out those irrelevant points. Then, Step 4 queries the feature grids for the filtered points and returns each point’s volumetric parameters. Step 5 and 6 prepare the volumetric parameters into a receivable

data format for $sMLP$ with TP or TCP. Step 7 and 8 compute the RGB values for the expected scene. If a backward process is required, Step 9 calculates the MSE loss between the expected and the ground-truth scenes.

Notably, the proposed methods, **functioning in a plug-in way**, can be applied to other NeRF frameworks, e.g., the SoTA TensoRF (Chen et al. 2022), and is also orthogonal and applicable to those efficient NeRFs such as Fast-NeRF(Garbin et al. 2021) and KiloNeRF(Reiser et al. 2021).

Experiments

In this section, we demonstrate the effectiveness of our proposed SpikingNeRF. **A**) We first build SpikingNeRF on the voxel-grid based DVGO framework (Sun, Sun, and Chen 2022), and compare the proposed TRA encoding with the naive data encodings. **B**) We extend SpikingNeRF to the TensoRF framework (Chen et al. 2022) to show the flexibility of our method, and compare SpikingNeRF with the original DVGO and TensoRF along with other NeRF-based works in both rendering quality and energy cost. **C**) We evaluate SpikingNeRF on the *neuromorphic accelerator SpikeSim* (Moitra et al. 2023) and SATA (Yin et al. 2022) and GPU to compare the hardware friendliness of the proposed TCP over TP, meanwhile showcase the energy advantage on neuromorphic accelerators over ANNs. **D**) we further discuss the alignment direction issue and extensively compare the merits of our edge-friendly spiking approach with the classic quantization method. Note that, for text saving, we defer the results of unbounded inward-facing and forward-facing datasets and all visualizations to the appendix, and the results of SATA evaluation is also deferred along with the detailed hardware descriptions. For clarity, we term the DVGO-based SpikingNeRF as **SpikingNeRF-D** and the TensoRF-based as **SpikingNeRF-T**. If not stated otherwise, TCP is utilized by default.

Experimental settings

We conduct experiments mainly on the four inward-facing datasets, including Synthetic-NeRF(Mildenhall et al. 2021), Synthetic-NSVF(Liu et al. 2020), BlendedMVS(Yao et al. 2020), and Tanks&Temples(Knapitsch et al. 2017). We refer to *DVGO as the ANN counterpart to SpikingNeRF-D*. We refer to *TensoRF as the ANN counterpart to SpikingNeRF-T*. In terms of the energy computation, we follow the prior arts (Zhou et al. 2022; Horowitz 2014) to estimate the theoretical rendering energy cost in most of our experiments except for those in Tab. 4 whose results are produced by the neuromorphic accelerator SpikeSim. **Very detailed implementation hyper-parameters, the experiment fairness declaration, and thorough SpikeSim evaluation details are all specified in the appendix.**

Comparisons and ablations

Comparisons with the conventional data encodings. As described in Data encoding, we propose two naive versions of SpikingNeRF-D that adopt two different conventional data encoding schemes: direct-encoding and Poisson-encoding. On the one hand, Poisson-encoding, severely losing the feature information and producing **at most 24.83**

Table 1: Comparisons with direct-encoding under different time step settings.

| Metric | PSNR↑ | SSIM↑ | Energy(mJ) ↓ | PSNR↑ | SSIM↑ | Energy(mJ) ↓ | PSNR↑ | SSIM↑ | Energy(mJ) ↓ |
|------------------------|--------------|--------------|---------------|-------------------|--------------|---------------|--------------|--------------|---------------|
| <i>Direct-Encoding</i> | TimeStep=1 | | | TimeStep=2 | | | TimeStep=4 | | |
| Synthetic-NeRF | 31.22 | 0.947 | 113.03 | 31.51 | 0.951 | 212.20 | 31.55 | 0.951 | 436.32 |
| Synthetic-NSVF | 34.17 | 0.969 | 53.73 | 34.49 | 0.971 | 104.05 | 34.56 | 0.971 | 217.86 |
| <i>TRA</i> | | | | Dynamic Time Step | | | | | |
| Synthetic-NeRF | 31.34 | 0.949 | 110.80 | 31.59 | 0.951 | 185.78 | 31.64 | 0.952 | 308.84 |
| Synthetic-NSVF | 34.33 | 0.970 | 56.69 | 34.63 | 0.972 | 98.39 | 34.57 | 0.972 | 165.17 |

TRA denotes the proposed time-ray alignment with TCP.

Table 2: Comparisons with direct-encoding on the same sampling density levels.

| Density Level | 1 (Base) | | | 2 | | | 4 | | |
|------------------------------------|--------------|--------------|---------------|--------------|--------------|---------------|--------------|--------------|---------------|
| Metric | PSNR↑ | SSIM↑ | Energy(mJ) ↓ | PSNR↑ | SSIM↑ | Energy(mJ) ↓ | PSNR↑ | SSIM↑ | Energy(mJ) ↓ |
| <i>Direct-Encoding</i> | | | | | | | | | |
| Synthetic-NeRF | 31.22 | 0.947 | 113.03 | 31.40 | 0.949 | 192.81 | 31.46 | 0.950 | 337.98 |
| Synthetic-NSVF | 34.17 | 0.969 | 53.73 | 34.45 | 0.970 | 94.58 | 34.56 | 0.971 | 168.10 |
| <i>Time-ray Alignment with TCP</i> | | | | | | | | | |
| Synthetic-NeRF | 31.34 | 0.949 | 110.80 | 31.59 | 0.951 | 185.78 | 31.64 | 0.952 | 308.84 |
| Synthetic-NSVF | 34.33 | 0.970 | 56.69 | 34.63 | 0.972 | 98.39 | 34.57 | 0.972 | 165.17 |

PSNR among all time-step settings and datasets, achieves far-from-acceptable synthesis quality, which indicates it does not work at all. The corresponding quantitative and qualitative results of this ineffective scheme are deferred to the appendix. On the other hand, as listed in Tab. 1, direct-encoding obtains good synthesis performance with only one time-step, and can achieve higher PSNR as the time step increases. Inheriting the good startup of direct-encoding, our proposed TRA shows better energy efficiency and rendering ability over direct-encoding. In Tab. 1, we change the TRA’s time step by adjusting the default sampling density to compare with Direct-Encoding (DE) of different time steps since it is unfeasible to explicitly set TRA’s time step due to its dynamic temporal length. Tab. 1 shows that TRA has *better rendering quality under the same energy levels*. We also compare TRA with DE under the same sampling densities in Tab. 2 for fairness, and the outcome still holds. The appendix contains the full statistics of Tab. 1 and Tab. 2 for each specific scene, which also show TRA consistently outperforms these conventional encodings. Conclusively, TRA exploits SNN’s temporal characteristics in 3D rendering and proves simple and effective.

Quantitative comparisons with the ANN counterparts and other NeRFs. As shown in Tab. 3, our SpikingNeRF-D with TCP can achieve a 70.79% energy saving with a 0.53 PSNR drop on average over the ANN counterpart. Such a trade-off between synthesis quality and energy cost is reasonable because a significant part of inference is conducted with the addition operations in the sMLP of SpikingNeRF-D rather than the multiplication operations in the original DVGO. On the one hand, compared with those methods, e.g., NeRF(Mildenhall et al. 2021), Mip-NeRF(Barron et al. 2021), JaxNeRF(Deng, Barron, and Srinivasan 2020), that do not perform the masking operation, SpikingNeRF-D can reach orders of magnitude lower energy consumption. On the other hand, compared with those methods, e.g., NSVF(Liu et al. 2020), DIVeR(Wu et al.

2022), DVGO(Sun, Sun, and Chen 2022), TensoRF(Chen et al. 2022), that significantly exploit the masking operation, SpikingNeRF-D can still obtain better energy efficiency and comparable synthesis quality. Even compared with Kilo-NeRF(Reiser et al. 2021) that is aiming at fast rendering (which takes days to train), SpikingNeRF-D (that takes minutes to train) still performs better. Furthermore, following (Alyamkin et al. 2018; Lee, Yang, and Fan 2023), we adopt the analogous PSNR/Energy to further estimate the energy efficiency of SpikingNeRF and the ANN baselines. As listed in Tab. 3, SpikingNeRF-D achieves superior energy efficiency among these competitors. Given Fig. 1, the superiority of our SpikingNeRF in energy efficiency is vivid. Moreover, SpikingNeRF-T also reduces energy consumption by 62.80% with a 0.69 PSNR drop on average. Except for Tanks&Temples, SpikingNeRF-T outperforms DVGO in both PSNR and energy cost. Notably, SpikingNeRF-T only uses two FC layers as TensoRF does. One layer is for encoding data with full precision, the other for spiking with binary computation, leading to only half of the computation burden being tackled with the addition operations. This explains why SpikingNeRF-T performs slightly worse than SpikingNeRF-D in terms of energy reduction ratio. In conclusion, these results demonstrate the effectiveness of our proposed SpikingNeRF in improving energy efficiency.

Qualitative comparisons. Due to text limitation, we defer piles of visualizations of SpikingNeRF-D rendering results to the appendix. Basically, SpikingNeRF-D shares the analogous issues with the ANN counterpart on texture distortion.

Advantages of temporal condensing on the hardware. To demonstrate the advantages of the proposed temporal condensing on hardware accelerators as described in Sec. TCP, we evaluate SpikingNeRF-D with TCP and TP on SpikeSim using the SpikeFlow architecture. For one thing, as listed in Tab. 4, TCP consistently outperforms TP in both inference latency and energy overhead by a significant margin over the two datasets. Specifically, The gap between TCP

Table 3: Comparisons with the ANN counterpart and other NeRF-based methods.

| Dataset Metric | Synthetic-NeRF | | | | Synthetic-NSVF | | | | BlendedMVS | | | | TanksTemples | | | |
|----------------------|----------------|--------------|---------------|--------------|----------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|---------------|--------------|
| | PSNR↑ | SSIM↑ | Energy↓ (mJ) | P/E↑ | PSNR↑ | SSIM↑ | Energy↓ (mJ) | P/E↑ | PSNR↑ | SSIM↑ | Energy↓ (mJ) | P/E↑ | PSNR↑ | SSIM↑ | Energy↓ (mJ) | P/E↑ |
| NeRF | 31.01 | 0.947 | 4.5e5 | 6.9e-5 | 30.81 | 0.952 | 4.5e5 | 6.8e-5 | 24.15 | 0.828 | 3.1e5 | 7.8e-5 | 25.78 | 0.864 | 1.4e6 | 1.8e-5 |
| Mip-NeRF | 33.09 | 0.961 | 4.5e5 | 7.4e-5 | - | - | - | - | - | - | - | - | - | - | - | - |
| JaxNeRF | 31.65 | 0.952 | 4.5e5 | 7.0e-5 | - | - | - | - | - | - | - | - | 27.94 | 0.904 | 1.4e6 | 2.0e-5 |
| NSVF | 31.74 | 0.953 | 16427 | 1.9e-3 | 35.13 | 0.979 | 8864 | 4.0e-3 | 26.90 | 0.898 | 15149 | 1.8e-3 | 28.40 | 0.900 | 101443 | 2.8e-4 |
| DIVeR | 32.32 | 0.960 | 343.96 | 0.094 | - | - | - | - | 27.25 | 0.910 | 548.65 | 0.050 | 28.18 | 0.912 | 1930.67 | 0.015 |
| KiloNeRF | 31.00 | 0.95 | 185.12 | 0.167 | 33.37 | 0.97 | 99.89 | 0.334 | 27.39 | 0.92 | 170.71 | 0.160 | 28.41 | 0.91 | 723.79 | 0.039 |
| DVGO* | 31.98 | 0.957 | 374.72 | 0.085 | 35.12 | 0.976 | 187.85 | 0.187 | 28.15 | 0.922 | 320.66 | 0.088 | 28.42 | 0.912 | 2147.86 | 0.012 |
| TensoRF* | 33.14 | 0.963 | 641.17 | 0.052 | 36.74 | 0.982 | 465.09 | 0.079 | - | - | - | - | 28.50 | 0.920 | 2790.03 | 0.010 |
| SpikingNeRF-D w/ TP | 31.34 | 0.949 | 111.59 | 0.281 | 34.34 | 0.970 | 57.57 | 0.596 | 27.80 | 0.912 | 97.38 | 0.285 | 28.00 | 0.892 | 483.48 | 0.057 |
| SpikingNeRF-D w/ TCP | 31.34 | 0.949 | 110.80 | 0.283 | 34.34 | 0.970 | 56.69 | 0.606 | 27.80 | 0.912 | 96.37 | 0.288 | 28.09 | 0.896 | 581.04 | 0.048 |
| SpikingNeRF-T w/ TCP | 32.45 | 0.956 | 240.81 | 0.134 | 35.76 | 0.978 | 149.98 | 0.238 | - | - | - | - | 28.09 | 0.904 | 1165.90 | 0.024 |

* denotes an ANN counterpart implemented by the official codes.
P/E abbreviates the “PSNR/Energy”.

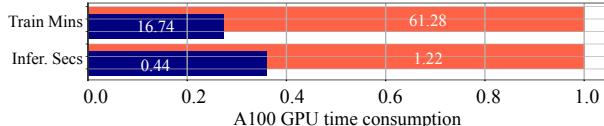
Table 4: Comparisons between TCP and TP on SpikeSim.

| Dataset | Synthetic-NeRF | | Synthetic-NSVF | |
|--------------|----------------|--------|----------------|--------|
| | w/ TCP | w/ TP | w/ TCP | w/ TP |
| Latency(s)↓ | 26.12 | 222.22 | 13.37 | 164.61 |
| Energy+(mJ)↓ | 65.78 | 559.45 | 33.68 | 414.37 |

+ denotes the energy result particularly produced by SpikeSim.

and TP is about an order of magnitude in both inference speed and energy cost. These results indicate TCP is simple but also effective at the inference stage. The same conclusion can also be drawn from the SATA (Yin et al. 2022) evaluation as shown in the appendix, which means TCP can benefit the sparsity-aware (event-driven) hardware as well. For the other, comparing the results on SpikeSim (65nm technology, Tab. 4) with those on 45nm technology general hardware (Tab. 3) further demonstrates that the proposed SpikingNeRF-D can substantially benefit from its neuromorphic computing nature on the neuromorphic hardware, achieving higher energy-efficiency over ANN baselines. Additionally, the temporal condensing will not harm the rendering quality at all as shown in Tab. 3.

In the SpikeSim evaluation, *the temporal condensing operation is done off-chip*. This causes on-chip computation can fully benefit from the dense data, thus accounting for the huge performance gap between TCP and TP. But note that due to the pipeline mechanism, the latency of such off-chip operation can be easily covered. To evaluate the substantial overhead and merits that temporal condensing can actually bring, we showcase the training and inference time on Synthetic-NeRF on single A100 GPU.



In the above figure, the orange bar is the time consumption of TP, while the blue one is that of TCP. Even roughly realizing the temporal condensing with PyTorch can still bring significant time-saving at the rendering stage on GPU.

Discussion of the alignment direction. The radiance accumulation originally has no direction but SNN has, so it’s necessary to discuss the time-ray alignment direction. We pro-

Table 5: Comparisons with temporal flip.

| Dataset | Synthetic-NeRF | | Synthetic-NSVF | |
|--------------|----------------|--------|----------------|-------|
| | w/o TF | w/ TF | w/o TF | w/ TF |
| PSNR↑ | 31.34 | 31.25 | 34.34 | 34.15 |
| SSIM↑ | 0.949 | 0.947 | 0.970 | 0.967 |
| Energy (mJ)↓ | 110.80 | 116.91 | 56.69 | 61.08 |

TF denotes temporal flip.

Table 6: Comparisons with Quantized NeRF (qNeRF).

| Dataset | Synthetic-NeRF | | Synthetic-NSVF | | |
|---------|----------------|--------------|----------------|--------------|--------------|
| | Metric | PSNR↑ | Energy↓ | PSNR↑ | Energy↓ |
| qNeRF | | 31.24 | 167.67 | 34.13 | 78.54 |
| ours | | 31.34 | 110.80 | 34.33 | 56.69 |

pose temporal flip to empirically decide the alignment direction since the querying direction of sMLP along the camera ray will affect the inference outcome. Tab. 5 lists the experimental results of SpikingNeRF-D with and without temporal flip, i.e., with the consistent and the opposite directions. Distinctly, keeping the direction of the temporal dimension consistent with that of the camera ray outperforms the opposite case on the two datasets in synthesis performance and energy efficiency. Therefore, the consistent alignment direction is important in SpikingNeRF.

Extensive comparisons with quantized ANN baselines.

To further demonstrate the merits of SpikingNeRF compared to the quantized NeRF version, we quantize the activation of DVGO to spike-bit, i.e., 1-bit, with the renowned LSQ (Esser et al. 2020), and compare it with SpikingNeRF-D in Tab. 6. The results show that SpikingNeRF-D outperforms the quantized ANN version on the two datasets in both synthesis quality and energy consumption, indicating SNNs do have an advantage over ANNs in scenario of ultra-low-energy computation.

Conclusion

This paper proposes SpikingNeRF that accommodates the spiking neural network to reconstructing real 3D scenes for the first time, improving energy efficiency. TRA is developed to encode sampled points, seamlessly combining the temporal characteristic of SNNs with the radiance ray. TCP is further proposed to improve hardware friendliness. Thorough experiments are conducted to prove the effectiveness.

References

- Alyamkin, S.; Ardi, M.; Brighton, A.; Berg, A. C.; Chen, Y.; Cheng, H.-P.; Chen, B.; Fan, Z.; Feng, C.; Fu, B.; et al. 2018. 2018 low-power image recognition challenge. *arXiv preprint arXiv:1810.01732*.
- Barron, J. T.; Mildenhall, B.; Tancik, M.; Hedman, P.; Martin-Brualla, R.; and Srinivasan, P. P. 2021. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 5855–5864.
- Chen, A.; Xu, Z.; Geiger, A.; Yu, J.; and Su, H. 2022. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision*, 333–350. Springer.
- Cheng, X.; Hao, Y.; Xu, J.; and Xu, B. 2020. LISNN: Improving Spiking Neural Networks with Lateral Interactions for Robust Object Recognition. In *IJCAI*, 1519–1525.
- Davies, M.; Srinivasa, N.; Lin, T.-H.; Chinya, G.; Cao, Y.; Choday, S. H.; Dimou, G.; Joshi, P.; Imam, N.; Jain, S.; et al. 2018. Loihi: A neuromorphic manycore processor with on-chip learning. *Ieee Micro*, 38(1): 82–99.
- Deng, B.; Barron, J. T.; and Srinivasan, P. P. 2020. JaxNeRF: an efficient JAX implementation of NeRF. URL <http://github.com/google-research/google-research/tree/master/jaxnerf>.
- Diehl, P. U.; and Cook, M. 2015. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in computational neuroscience*, 9: 99.
- Esser, S. K.; McKinstry, J. L.; Bablani, D.; Appuswamy, R.; and Modha, D. S. 2020. Learned Step Size Quantization. In *International Conference on Learning Representations*.
- Fang, W.; Chen, Y.; Ding, J.; Yu, Z.; Masquelier, T.; Chen, D.; Huang, L.; Zhou, H.; Li, G.; and Tian, Y. 2023. Spiking-Jelly: An open-source machine learning infrastructure platform for spike-based intelligence. *Science Advances*, 9(40): eadi1480.
- Fang, W.; Yu, Z.; Chen, Y.; Masquelier, T.; Huang, T.; and Tian, Y. 2021. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2661–2671.
- Garbin, S. J.; Kowalski, M.; Johnson, M.; Shotton, J.; and Valentin, J. 2021. Fastnerf: High-fidelity neural rendering at 200fps. In *Proceedings of the IEEE/CVF international conference on computer vision*, 14346–14355.
- Garg, I.; Chowdhury, S. S.; and Roy, K. 2021. Dct-snn: Using dct to distribute spatial information over time for low-latency spiking neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 4671–4680.
- Han, B.; Srinivasan, G.; and Roy, K. 2020. Rmp-snn: Residual membrane potential neuron for enabling deeper high-accuracy and low-latency spiking neural network. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 13558–13567.
- Hedman, P.; Srinivasan, P. P.; Mildenhall, B.; Barron, J. T.; and Debevec, P. 2021. Baking neural radiance fields for real-time view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 5875–5884.
- Horowitz, M. 2014. 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)*, 10–14. IEEE.
- Knapitsch, A.; Park, J.; Zhou, Q.-Y.; and Koltun, V. 2017. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)*, 36(4): 1–13.
- Lee, J.-J.; Zhang, W.; and Li, P. 2022. Parallel time batching: Systolic-array acceleration of sparse spiking neural computation. In *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 317–330. IEEE.
- Lee, Y.; Yang, L.; and Fan, D. 2023. Mf-nerf: Memory efficient nerf with mixed-feature hash table. *ArXiv*, abs/2304.12587, 2: 3.
- Li, Y.; Guo, Y.; Zhang, S.; Deng, S.; Hai, Y.; and Gu, S. 2021. Differentiable Spike: Rethinking Gradient-Descent for Training Spiking Neural Networks. *Advances in Neural Information Processing Systems*, 34.
- Liao, Z.; Zheng, Q.; Liu, Y.; and Pan, G. 2023. Spiking NeRF: Representing the Real-World Geometry by a Discontinuous Representation. *arXiv preprint arXiv:2311.09077*.
- Liu, L.; Gu, J.; Zaw Lin, K.; Chua, T.-S.; and Theobalt, C. 2020. Neural sparse voxel fields. *Advances in Neural Information Processing Systems*, 33: 15651–15663.
- Liu, Y.; Peng, S.; Liu, L.; Wang, Q.; Wang, P.; Theobalt, C.; Zhou, X.; and Wang, W. 2022. Neural rays for occlusion-aware image-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7824–7833.
- Maass, W. 1997. Networks of spiking neurons: the third generation of neural network models. *Neural networks*, 10(9): 1659–1671.
- Mao, R.; Tang, L.; Yuan, X.; Liu, Y.; and Zhou, J. 2024. Stellar: Energy-Efficient and Low-Latency SNN Algorithm and Hardware Co-Design with Spatiotemporal Computation. In *2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 172–185. IEEE.
- Masland, R. H. 2012. The neuronal organization of the retina. *Neuron*, 76(2): 266–280.
- Max, N. 1995. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2): 99–108.
- Mei, H.; Wang, Z.; Yang, X.; Wei, X.; and Delbrück, T. 2023. Deep polarization reconstruction with PDAVIS events. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 22149–22158.
- Mildenhall, B.; Srinivasan, P. P.; Tancik, M.; Barron, J. T.; Ramamoorthi, R.; and Ng, R. 2021. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1): 99–106.

- Moitra, A.; Bhattacharjee, A.; Kuang, R.; Krishnan, G.; Cao, Y.; and Panda, P. 2023. SpikeSim: An end-to-end Compute-in-Memory Hardware Evaluation Tool for Benchmarking Spiking Neural Networks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 1–1.
- Reiser, C.; Peng, S.; Liao, Y.; and Geiger, A. 2021. Kilonearf: Speeding up neural radiance fields with thousands of tiny mlps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 14335–14345.
- Roy, K.; Jaiswal, A.; and Panda, P. 2019. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 575(7784): 607–617.
- Shrestha, S. B.; and Orchard, G. 2018. Slayer: Spike layer error reassignment in time. *Advances in neural information processing systems*, 31.
- Sun, C.; Sun, M.; and Chen, H.-T. 2022. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5459–5469.
- Wässle, H. 2004. Parallel processing in the mammalian retina. *Nature Reviews Neuroscience*, 5(10): 747–757.
- Wu, L.; Lee, J. Y.; Bhattad, A.; Wang, Y.-X.; and Forsyth, D. 2022. Diver: Real-time and accurate neural radiance fields with deterministic integration for volume rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 16200–16209.
- Wu, Y.; Deng, L.; Li, G.; Zhu, J.; Xie, Y.; and Shi, L. 2019. Direct training for spiking neural networks: Faster, larger, better. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 1311–1318.
- Yao, Y.; Luo, Z.; Li, S.; Zhang, J.; Ren, Y.; Zhou, L.; Fang, T.; and Quan, L. 2020. Blendedmvs: A large-scale dataset for generalized multi-view stereo networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 1790–1799.
- Yin, R.; Moitra, A.; Bhattacharjee, A.; Kim, Y.; and Panda, P. 2022. Sata: Sparsity-aware training accelerator for spiking neural networks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 42(6): 1926–1938.
- Zhang, J.; Dong, B.; Zhang, H.; Ding, J.; Heide, F.; Yin, B.; and Yang, X. 2022. Spiking transformers for event-based single object tracking. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, 8801–8810.
- Zhang, Z.; Wang, H.; Han, S.; and Dally, W. J. 2020. Sparch: Efficient architecture for sparse matrix multiplication. In *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 261–274. IEEE.
- Zhou, Z.; Zhu, Y.; He, C.; Wang, Y.; Yan, S.; Tian, Y.; and Yuan, L. 2022. Spikformer: When spiking neural network meets transformer. *arXiv preprint arXiv:2209.15425*.
- Zhu, L.; Wang, X.; Chang, Y.; Li, J.; Huang, T.; and Tian, Y. 2022a. Event-based video reconstruction via potential-assisted spiking neural network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 3594–3604.
- Zhu, R.-J.; Zhao, Q.; and Eshraghian, J. K. 2023. Spikegpt: Generative pre-trained language model with spiking neural networks. *arXiv preprint arXiv:2302.13939*.
- Zhu, Z.; Peng, J.; Li, J.; Chen, L.; Yu, Q.; and Luo, S. 2022b. Spiking graph convolutional networks. *arXiv preprint arXiv:2205.02767*.