Autonomous Driving with Spiking Neural Networks

Rui-Jie Zhu[1]*, **Ziqing Wang**[2]*, **Leilani Gilpin**[1], **Jason K. Eshraghian**[1]†

[1]University of California, Santa Cruz, USA
[2]Northwestern University, USA

Abstract

Autonomous driving demands an integrated approach that encompasses perception, prediction, and planning, all while operating under strict energy constraints to enhance scalability and environmental sustainability. We present Spiking Autonomous Driving (SAD), the first unified Spiking Neural Network (SNN) to address the energy challenges faced by autonomous driving systems through its event-driven and energy-efficient nature. SAD is trained end-to-end and consists of three main modules: perception, which processes inputs from multi-view cameras to construct a spatiotemporal bird's eye view; prediction, which utilizes a novel dual-pathway with spiking neurons to forecast future states; and planning, which generates safe trajectories considering predicted occupancy, traffic rules, and ride comfort. Evaluated on the nuScenes dataset, SAD achieves competitive performance in perception, prediction, and planning tasks, while drawing upon the energy efficiency of SNNs. This work highlights the potential of neuromorphic computing to be applied to energy-efficient autonomous driving, a critical step toward sustainable and safety-critical automotive technology. Our code is available at `https://github.com/ridgerchu/SAD`.

1 Introduction

Autonomous driving, often considered the 'holy grail' of computer vision, integrates complex processes such as perception, prediction, and planning to achieve higher levels of vehicle automation as classified by the SAE J3016 standard [1]. Many new vehicles now feature Level 2 autonomy, with transitions to Level 3 marking notable advancements. However, these systems must adhere to energy constraints of 50-60 W/h [2] and face increasing environmental concerns. Sudhakar *et al.* highlight the need for hardware efficiency to double every 1.1 years to maintain 2050 emissions from autonomous vehicles below those of 2018 data center levels [3].

Spiking Neural Networks (SNNs) offer a promising solution for energy-efficient intelligence by using sparse, event-driven, single-bit spiking activations for inter-neuron communication, mimicking biological neurons [4, 5, 6, 7]. Such workloads can be accelerated for low latency and low energy, when processed on neuromorphic hardware that utilizes asynchronous, fine-grain processing to efficiently handle spiking signals and parallel operations [8]. Much like in the brain, spikes are thought to encode information over time, and have shown improvements in the energy efficiency of sequence-based computer vision tasks by several orders of magnitude in a variety of workloads [9, 10, 11].

In the past several years, SNNs have rapidly improved in performance across various tasks, including image classification [12, 13, 14, 15, 16, 17], object detection [18, 19, 20], semantic segmentation [21], low-level image reconstruction [22, 23, 24, 25, 26], and language modeling [27, 28, 29], with most of these works focused on computer vision. These advancements have brought SNN performance

*Equal contribution
†Corresponding author, jsn@ucsc.edu

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

 https://doi.org/10.52202/079017-4346

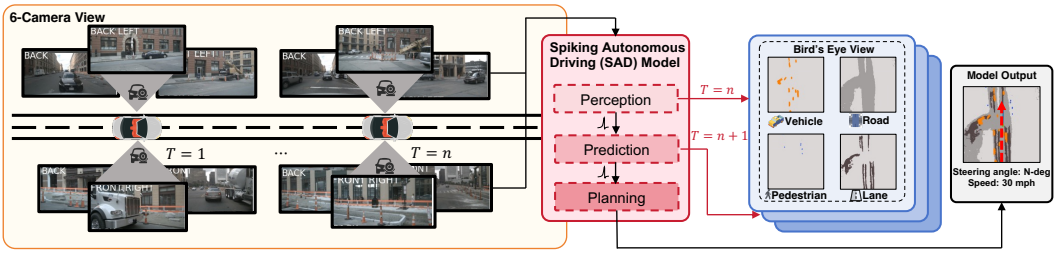

Figure 1: How SAD enables autonomous driving from vision to planning: The system processes inputs from six cameras across multiple frames. The perception module encodes feature information related to the present input frame ($T = n$), the prediction module predicts feature information of the next frame using sequential information ($T = n + 1$), and the model output generates a steering and acceleration plan. This process creates a bird's eye view (BEV) and trajectory plan for navigation.

closer to that of Artificial Neural Networks (ANNs) in fundamental computer vision tasks. Despite this progress, SNNs have not yet proven effective in complex real-world computer vision applications that involve multiple subtasks.

We introduce the first SNN designed for end-to-end autonomous driving, integrating perception, prediction, and planning into a single model. Achieving this milestone for SNNs involved spatiotemporal fusion of visual embeddings for enhanced perception, probabilistic future modeling for accurate prediction, and and a high performance temporal mixing spiking recurrent unit that effectively incorporates safety and comfort considerations into high-level planning decisions. By leveraging the event-driven and energy-efficient properties of SNNs, our model processes visual inputs, forecasts future states, and calculates the final trajectory for autonomous vehicles. Previously, GRUs or 3D convolutions were used in spatiotemporal visual tasks, though these operators have been entirely replaced with spiking neurons for time-mixing. This work marks a significant advancement in neuromorphic computing, demonstrating the potential of SNNs to handle the complex requirements of low-power autonomous driving. Our experiments show that this SNN-based system performs competitively with traditional deep learning approaches, while offering improved energy efficiency and reduced latency.

2 Related Works

Spiking Neural Networks in Autonomous Systems are particularly effective for low-power, edge intelligence applications, leveraging deep learning and neuroscience principles to boost operational efficiency [6, 4, 30, 31, 32]. Many neuromorphic autonomous systems use SNNs as a Proportional Derivative-Integral (PID) controller to adapt to changing conditions, such as different payloads in unmanned aerial vehicles [33, 34, 35, 36], or to prevent drift in non-neutral buoyancy blimps [37]. Much of this work successfully deployed SNNs as PID controllers in real-world systems on neuromorphic hardware, highlighting the potential for low-power autonomous control. Moving from the sky to the ground, SNN-based PID controllers have been used for lane-keeping tasks in simulated environments with reference trajectories provided by the lane [38, 39, 40], as well as with LiDAR for collision avoidance in simulated environments [41]. These tasks all show successful use of SNNs in adaptive control, though the objective of a PID controller is to maintain a desired setpoint which is often a well-defined and simpler goal than end-to-end autonomous driving in the face of complex and noisy environments. We push the frontier of what SNNs are capable of in this paper.

End-to-end Autonomous Driving directly maps sensory inputs to vehicle control outputs using a single, fully differentiable model. Existing approaches can be broadly classified into two categories: imitation learning and reinforcement learning paradigms [42]. Imitation learning methods, such as behavior cloning [43, 44, 45, 46, 47] and inverse optimal control [48, 49, 50, 51, 52], learn a driving policy by mimicking expert demonstrations. On the other hand, reinforcement learning techniques [53, 54, 55, 56] enable the driving agent to learn through interaction with the environment by optimizing a reward function. Recent advancements, such as multi-modal sensor fusion [57, 58, 59, 60, 61], attention mechanisms [57, 62, 58], and policy distillation [43, 63, 64, 65, 66] have significantly improved the performance of end-to-end driving systems.

https://doi.org/10.52202/079017-4346

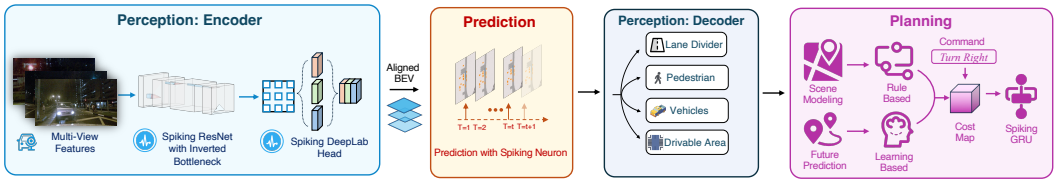

Figure 2: Overview of SAD. The multi-view features from the perception encoder, including a spiking ResNet with inverted bottleneck and spiking DeepLab head, are fed into a prediction module using spiking neurons. The perception decoder then generates lane divider, pedestrian, vehicle and drivable area predictions. Finally, the planning module models the scene and generates future predictions to inform rule-based command decisions for turning, stopping, and goal-directed navigation.

3 Method

This section presents the Spiking Autonomous Driving (SAD) method, an end-to-end framework that integrates perception, prediction, and planning using SNNs (Fig. 2). SAD's biologically-inspired architecture enables efficient spatiotemporal processing for autonomous driving, with the spiking neuron layer at its core. This layer incorporates spatiotemporal information and enables spike-driven computing, making it well-suited for the dynamic nature of autonomous driving tasks.

The perception module is the first stage of the SAD framework. It constructs a bird's eye view (BEV) representation from multi-view camera inputs, providing a human-interpretable understanding of the environment. This representation serves as the foundation for the subsequent prediction and planning modules. The prediction module uses the BEV to forecast future states using a 'dual pathway', which allows data to flow through two separate paths, providing a pair of alternative data embeddings. One pathway focuses on encoding information from the past, while the other pathway specializes in predicting future information. Subsequently, the embeddings from these two pathways are fused together, integrating the past and future information to facilitate temporal mixing. This enables the anticipation of dynamic changes in the environment, which is crucial for safe and efficient autonomous driving. Leveraging the perception and prediction outcomes, the planning module generates safe trajectories by considering predicted occupancy of space around the vehicle, traffic rules, and ride comfort. To optimize the entire pipeline, SAD is trained end-to-end using a composite loss that combines objectives from perception, prediction, and planning. The following subsections describe each module in detail.

3.1 Spiking Neuron Layer

All modules consist of spiking neurons rather than artificial neurons, and so a formal definition of spiking neurons is provided below. Spiking neuron layers integrate spatio-temporal information into the hidden state of each neuron (membrane potential) which are converted into binary spikes emitted to the next layer. Spiking neurons can be represented as recurrent neurons with binarized activations and a diagonal recurrent weight matrix such that the hidden state of a neuron is isolated from all other neurons (see Ref. [5] for a derivation). We adopt the standard Leaky Integrate-and-Fire (LIF) [7] model, whose dynamics are described by the following equations:

$$U[t] = H[t-1] + X[t], \tag{1}$$
$$S[t] = \Theta\left(U[t] - u_{th}\right), \tag{2}$$
$$H[t] = V_{\text{reset}} S[t] + \left(\beta U[t]\right)\left(1 - S[t]\right), \tag{3}$$

where $X[t]$ is the input to the neuron at time-step t, and is typically generated by convolutional or dense operators. $U[t]$ denotes the membrane potential of the neuron, and integrates $X[t]$ with the temporal input component $H[t-1]$. $\Theta(\cdot)$ is the Heaviside step function, which is 1 for $x \geq 0$ and 0 otherwise. If $U[t]$ exceeds the firing threshold u_{th}, the spiking neuron emits a spike $S[t] = 1$ as its activation, and the temporal output $H[t]$ is reset to V_{reset}. Otherwise, no spike is emitted ($S[t] = 0$) and $U[t]$ decays to $H[t]$ with a decay factor $\beta < 1$. For brevity, we refer to Eq. 2 as $\mathcal{SN}(\cdot)$, where the input U is a tensor of membrane potential values fed into multiple spiking neurons, and the output S is an identically-shaped tensor of spikes.

https://doi.org/10.52202/079017-4346

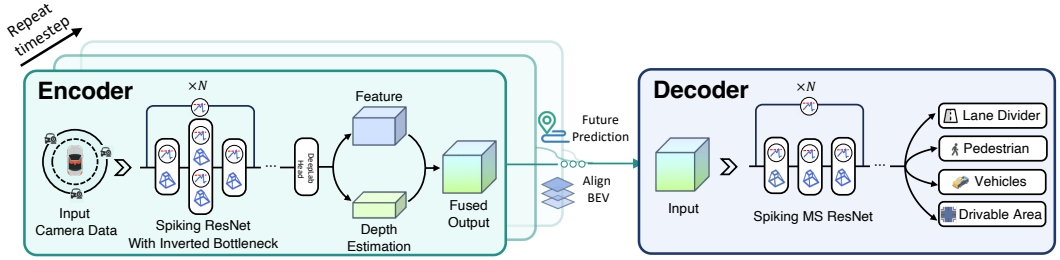

Figure 3: The perception module. The encoder takes multi-camera input data, passes it through a spiking ResNet with inverted bottleneck to generate feature representations, each of which has its own depth estimation. These are fused and passed to the decoder, which generates predictions for lane dividers, pedestrians, vehicles and drivable areas.

3.2 Perception: Distinct Temporal Strategies for Encoder and Decoder

Fig. 3 illustrates the overall architecture of the perception module. The perception stage constructs a spatiotemporal BEV representation from multi-view camera inputs over t time-steps through spatial and temporal fusion of features extracted from the cameras. It consists of an encoder, which processes each camera input to generate features and depth estimations, and a decoder, which generates BEV segmentation and instructs the planning module. A future prediction module is depicted between the encoder and decoder in Fig. 3. It is not used in the first stage where the perception module is trained alone, but it is included in the second stage once the prediction module is included.

The temporal dimension processing in the Encoder/Decoder architecture is a crucial design consideration, as both SNNs and autonomous driving data inherently possess a temporal structure. There are two approaches to handle this:

- **Sequential Alignment (SA):** sequential input data is passed to the SNN step-by-step by aligning the time-varying dimension of the input data with the model
- **Sequence Repetition (SR):** sequential input data is aligned with the batch dimension for better parallelism during training, and individual frames are repeated T times over the model sequence, so as to create virtual timesteps. SR is commonly used for pre-training sequence-based models on static image datasets.

Given these two encoding options, we test all four combinations of these options applied to both the encoder and decoder of the perception block. Based on our experiments (detailed in Sec. 4.3.1), the best performing approach is using SR for the encoder and SA for the decoder. The encoder is also pre-trained on ImageNet-1K which requires the use of repeated images to create virtual timesteps. Further details regarding the pre-training of the encoder can be found in Appendix D.1. Conversely, the decoder is trained from scratch, which naturally assumes a temporal-mixing role, making the alignment of sequential data with the model sequence a more effective approach.

The training process involves first training the encoder-decoder, followed by the prediction module. This approach integrates spatial and temporal information for comprehensive BEV representation in autonomous vehicle perception and planning.

Encoder: Spiking Token Mixer with Sequence Repetition The encoder module can be thought of as a spiking token mixer (STM). The STM consists of 12-layers of spiking CNN pre-trained on ImageNet-1K [67] to generate vision patch embeddings, which is effectively a deeper version of the 'spiking patch embedding' from Ref. [13, 14]. Across these 12 layers, the number of channels in each layer is designed to first increase and then decrease so as to act as an inverted bottleneck. While SPS layers are usually terminated by self-attention, we replaced this with dense layers instead, which both reduces computational resources and leads to improved performance. In doing so, we achieved a 72.1% ImageNet top-1 classification accuracy with only 12M parameters. In contrast, the previous spiking vision transformers that employs self-attention reached 70.2% with the same number of parameters [14].

The encoder extracts feature embeddings and depth estimations while squeezing the image into a smaller latent space. The overall workflow of the encoder can be summarized as follows:

$$
\begin{aligned}
X &= \text{STM}(I), && I \in \mathbb{R}^{N\times C_{in}\times T\times L\times H\times W}, && X \in \{0,1\}^{C\times T\times L\times H\times W} \\
\mathcal{F} &= \text{Head}_{\text{feature}}(X), && X \in \{0,1\}^{C\times T\times L\times H\times W}, && \mathcal{F} \in \{0,1\}^{C_f\times L\times H\times W}, \\
\mathcal{D} &= \text{Head}_{\text{depth}}(X), && X \in \{0,1\}^{C\times T\times L\times H\times W}, && \mathcal{D} \in \{0,1\}^{C_d\times L\times H\times W} \\
Y &= \mathcal{F}\otimes\mathcal{D}, && Y \in \{0,1\}^{C_f\times C_d\times T\times L\times H\times W}
\end{aligned}
$$

The STM encoder described above is used to extract feature embeddings and depth estimates from each camera frame $I_t \in \mathbb{R}^{N\times C_{in}\times H\times W}$, where $N = 6$ is the number of cameras, $C_{in} = 3$ refers to the number of input channels (RGB), and $H \times W$ refers to the video resolution. Note that the use of sequence repetition means that T is the number of times the same frame is repeated over the sequence, while L is the number of frames in a continuous camera recording. As such, the dimensions $N \times L$ are stacked so as to speed up processing. The encoder generates two outputs: feature embeddings $\mathcal{F} \in \{0,1\}^{N\times L\times C_f\times H\times W}$ and depth estimates $\mathcal{D} \in \{0,1\}^{N\times L\times C_d\times H\times W}$, where C_f and C_d represent the number of channels for feature embeddings and depth estimates, respectively. These outputs are then fused to form a single tensor $Y \in \{0,1\}^{N\times L\times(C_f+C_d)\times H\times W}$, effectively combining the C_f and C_d dimensions into a single dimension. The fused tensor Y is subsequently passed to the decoder for further processing. For a more comprehensive understanding of the encoder architecture and its internal workings, please refer to Appendix A.

Decoder: Sequential Alignment with Streaming Feature Maps The recurrent decoder aligns feature maps sequentially, introducing a new instance of data at each time-step, contrasting with the encoder's repeated inputs. Using SA rather than SR for the decoder improves performance for two reason: 1) the decoder does not need to be pre-trained on static, repeated data, and 2) the decoder acts as a temporal mixer. In this architecture, time-mixing is achieved using LIF neurons and allows them to take on the role of self-attention without the same computational burden. The LIF neurons are composed as a shared backbone as a set of layers used to extract features from data before being passed to various specialized heads, each dedicated to a specific task. The high-level dataflow summarized as follows:

$$
\begin{aligned}
X &= \text{SharedBackbone}(I_D), && I \in \mathbb{R}^{C_{in}\times L\times H\times W}, && X \in \{0,1\}^{C_{med}\times T\times L\times H\times W} \\
Y_k &= \text{Head}_k(X), && Y_k \in \mathbb{R}^{C_{out}\times L\times H\times W}, && k \in \{\text{seg, ped, map, inst}\}
\end{aligned}
$$

where I_D is the input tensor to the decoder with dimensions (C_{in}, L, H, W), X is the output of the shared backbone with dimensions (C_{med}, T, L, H, W),Y_k is the output of the k-th head with dimensions (C_{out}, L, H, W), and k indexes into the heads of the different tasks: vehicle segmentation (seg), pedestrian (ped), HD map (map), and future instance (inst). The shared backbone is implemented using the first three layers of MS-ResNet18 [68] followed by three upsampling layers with a factor of 2 and skip connections. More details about MS-ResNet can be found in Appendix B. The resulting features have 64 channels and are then passed to different heads according to the task requirements. Each head consists of a spiking convolutional layer.

3.3 Prediction: Fusing Parallel Spike Streams

Predicting future agent behavior is crucial for an autonomous vehicle to be reactive and make informed decisions in real-time. In our approach, we accumulate historical BEV features and predict the next few timesteps using purely LIF neurons. However, the stochastic nature of interactions among agents, traffic elements and road conditions, makes it challenging to accurately predict future trajectories. To address this, we model future uncertainty with a conditional Gaussian distribution.

To accomplish this, two layers of LIF neurons are used. The first parallel layer takes the present and prior output BEV feature maps from the encoder of the perception model $(x_1, \ldots, x_t)$ as inputs. The first BEV x_1 is also used as the initial membrane potential for this LIF layer. The second parallel layer accounts for the uncertainty distribution of future BEV predictions. The uncertainty distribution is generated by passing the present feature x_t through 4 spiking MS-ResNet blocks, average pooling, and another 2D spiking convolution with a kernel size of $(1, 1)$ to transform the channel depth to double the output of the first parallel layer. Another averaging pooling operator compresses the

 https://doi.org/10.52202/079017-4346

feature map down to a vector. The vector is split in two sub-vectors representing mean $\mu \in \mathbb{R}^L$ and variance $\sigma \in \mathbb{R}^L$, and these values populate a diagonal Gaussian distribution of the latent feature map. Using the μ and standard deviation σ, we can construct the Gaussian distribution at timestep t, denoted as η_t. This distribution, represented by the parameters μ and σ, can then be concatenated with the input at the current timestep x_t.

Simultaneously, all prior spiking outputs are concatenated to the present input x_t, denoted below $x_{0:t}$. The predicted BEV feature for the next timestep is calculated below:

$$\hat{x}_{t+1} = \text{LIF}(\text{BN}(\text{Conv2d}(\text{concatenate}(x_t, \eta_t))) \oplus \text{LIF}(\text{BN}(\text{Conv2d}((x_{0:t}))), \tag{4}$$

where $\hat{x}_{t+1}$ represents the predicted BEV features for the next timestep, $\text{LIF}(\cdot)$ denotes the LIF neuron layer, concatenate$(\cdot)$ represents the concatenation operation, and $\oplus$ denotes element-wise addition of the outputs from the two LIF layers, and the inner $\text{Conv2d}(\cdot)$ and $\text{BN}(\cdot)$ layers are used to ensure consistency in the output dimensions of the first and second LIF layers. The mixture prediction serves as the basis for the subsequent prediction steps. By recursively applying this dual-pathway prediction method, we obtain the predicted future states $(\hat{x}_{t+1}, \ldots, \hat{x}_{t+n})$. The overall datapath is illustrated in Fig. 4. Following the dual-pathway prediction, all the features are fed into a Spiking ResNet using SA for additional temporal mixing. The historical features $(x_1, \ldots, x_t)$ and the predicted future features $(\hat{x}_{t+1}, \ldots, \hat{x}_{t+n})$ are then passed to the perception decoder, which consists of multiple output heads to generate various interpretable intermediate representations.

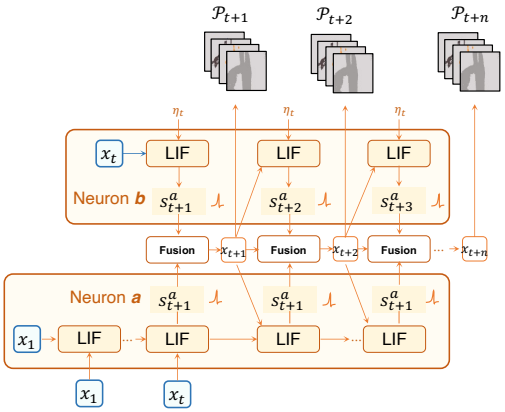

Figure 4: Dual pathway modeling for prediction. Neuron a captures future multi-modality by incorporating uncertainty distribution. Neuron b compensates for information gaps using past variations. Inputs x_1 and x_t from both pathways are used for the next prediction step.

3.4 Planning: Temporal Spiking Temporal for Trajectory Refinement

The primary objective of the SAD system is to plan a trajectory that is both safe and comfortable, aiming towards a designated target. This is accomplished through a planning pipeline that integrates predicted occupancy grids o, map representations m, and temporal dynamics of environmental elements like pedestrians.

Motion Planning. Initially, our motion planner generates a diverse set of potential trajectories using the bicycle model [69]. Among these, the trajectory that minimizes a predefined cost function is selected. This function integrates various factors including learned occupancy probabilities (from segmentation maps generated in the Prediction section) and compliance with traffic regulations to ensure the selected trajectory optimizes for safety and smoothness.

Cost Function. The cost function f employed is a multi-component function, where:

$$f(\tau, o, m; w) = f_o(\tau, o, m; w_o) + f_v(\tau; w_v) + f_r(\tau; w_r) \tag{5}$$

where $w = (w_o, w_v, w_r)$ represents the learnable parameters associated with each cost component, and τ denotes a set of trajectory candidates. Specifically, f_o evaluates trajectory compliance with static and dynamic obstacles, f_v is derived from the prediction decoder assessing future states, and f_r addresses metrics of ride comfort and progress towards the goal. The aim is to select the optimal set of trajectories τ^* that minimize this cost function. This cost function is adapted from Hu et al. [70] and is detailed in Appendix C.

Additionally, trajectories are filtered based on high-level commands (e.g., go forward, turn left, turn right) which tailor the trajectory selection to the immediate navigational intent.

Optimization with Spiking Gated Recurrent Unit. Following the initial selection, the "best" trajectory τ^* undergoes further refinement using a Spiking Gated Recurrent Unit (SGRU), as inspired by [71]. In this optimization phase, the hidden state h_t of the SGRU incorporates features derived from the front camera's encoder. The input state x_t is formulated by concatenating the vehicle's

Table 1: **Perception results.** We report the BEV segmentation IoU (%) of intermediate representations and their mean value.

Method	Spike	Drivable Area	Lane	Vehicle	Pedestrian	Avg.
VED [72]$^{\text{RAL}}$	✗	60.82	16.74	23.28	11.93	28.19
VPN [73]$^{\text{RAL}}$	✗	65.97	17.05	28.17	10.26	30.36
PON [74]$^{\text{CVPR}}$	✗	63.05	17.19	27.91	13.93	30.52
Lift-Splat [75]$^{\text{ECCV}}$	✗	72.23	19.98	31.22	15.02	34.61
IVMP [76]$^{\text{ICRA}}$	✗	74.70	20.94	34.03	17.38	36.76
FIERY [77]$^{\text{ICCV}}$	✗	71.97	33.58	38.00	17.15	40.18
ST-P3 [70]$^{\text{ECCV}}$	✗	75.97	33.85	38.00	17.15	42.69
SAD (Ours)	✓	64.74	27.78	34.82	15.12	35.62

current position, the corresponding position from the selected trajectory τ^*, and the designated target point. The SGRU model processes inputs as follows:

$$r_t = \Theta(W_{ir}x_t + b_{ir} + W_{hr}h_{t-1} + b_{hr}) \tag{6}$$
$$z_t = \Theta(W_{iz}x_t + b_{iz} + W_{hz}h_{t-1} + b_{hz}) \tag{7}$$
$$n_t = \Theta(W_{in}x_t + b_{in} + r_t \odot (W_{hn}h_{t-1} + b_{hn})) \tag{8}$$
$$h_t = (1 - z_t) \odot n_t + z_t \odot h_{t-1} \tag{9}$$

where h_t denotes the hidden state at time t, x_t represents the input, and r_t, z_t, n_t are the reset, update, and new candidate state gates respectively. The Heaviside function Θ ensures sparse and binarized operations in the state of the SGRU, thus preserving the advantages of SNNs.

This optimization step enhances trajectory reliability by mitigating uncertainties inherent in perceptual and predictive analyses, and by integrating dynamic traffic light information directly into the trajectory planning process.

After the optimization, the model can be trained end-to-end, from perception to planning. Details about the stage-wise training process and the end-to-end learning methodology can be found in Appendices D.2 and D.3, respectively.

4 Experiments

We evaluate the proposed model using the nuScenes dataset [78] with 20 epochs with 4 $\times$ NVIDIA A100 80GB. as is done in ST-P3 [70]. For our experiments, we consider $1.0s$ of historical context and predict $2.0s$ into the future, which corresponds to processing 3 past frames and predicting 4 future frames. Additional experimental details are elaborated in the Appendix E.

4.1 Experimental Results on nuScenes

Perception. Our evaluation focuses on the model's ability to interpret map representations and perform semantic segmentation. For map representation, we specifically assess the identification of drivable areas and lanes, which are critical for safe navigation as they dictate where the Self-Driving Vehicle (SDV) can travel and help maintain the vehicle's position within the lanes. Semantic segmentation tests the model's ability to recognize dynamic objects, such as vehicles and pedestrians, which are pivotal in urban driving scenarios.

We employ the Intersection-over-Union (IoU) metric to quantify the accuracy of our BEV segmentation tasks. The results, as summarized in Tab. 1, show that our SAD method, which is fully implemented with spiking neural networks (SNNs), competes favorably against state-of-the-art, non-spiking artificial neural networks (ANNs). Notably, our model achieves a superior mean IoU on the nuScenes dataset compared to existing leading methods such as VED [72], VPN [73], PON [74], and Lift-Splat [75]. Specifically, our SAD method outperforms the VED [72] model by **7.43%** in mean IoU. This enhancement is significant, considering that our network utilizes spiking neurons across all layers, which contributes to greater computational efficiency. Despite the inherent challenges of using SNNs, such as the binary nature of spikes and potential information loss compared to traditional

https://doi.org/10.52202/079017-4346

Table 2: **Prediction results.** We report semantic segmentation IoU (%) and instance segmentation metrics from the video prediction area. The *static* method assumes all obstacles static in the prediction horizon.

Method	Spike	Future Semantic Seg. IoU ↑	Future Instance Seg. PQ ↑	SQ ↑	RQ ↑
Static	✗	32.20	27.64	70.05	39.08
FIERY [77]^{ICCV}	✗	37.00	30.20	70.20	42.90
ST-P3 [70]^{ECCV}	✗	38.63	31.72	70.15	45.22
SAD (Ours)	✓	32.74	20.00	68.74	29.39

Table 3: **Planning results.**

Method	Spike	L2 (m) ↓			Collision (%) ↓			Energy (mJ)
		1s	2s	3s	1s	2s	3s	
NMP [48]^{CVPR}	✗	0.61	1.44	3.18	0.66	0.90	2.34	-
Freespace [51]^{CVPR}	✗	0.56	1.27	3.08	0.65	0.86	1.64	344.11
ST-P3 [70]^{ECCV}	✗	1.33	2.11	2.90	0.23	0.62	1.27	3520.40
SAD (Ours)	✓	1.53	2.35	3.21	0.62	1.26	2.38	46.92

ANNs, our results demonstrate that SAD is capable of delivering competitive perception accuracy in autonomous driving scenarios.

Prediction. We assess the predictive capabilities of our model using multiple metrics tailored for video prediction, specifically IoU, existing Panoptic Quality (PQ), Recognition Quality (RQ), and Segmentation Quality (SQ) for evaluating our prediction quality. The definition of these metrics can be found in Appendix E.2. The results, presented in Tab. 2, demonstrate that while our model does not employ an additional temporal module, the inherent temporal dynamics of spiking neurons facilitate effective information processing. However, our SAD model still shows a gap in performance when compared with state-of-the-art ANN methods.

Planning. In the planning domain, we evaluate our model using two primary metrics: L2 error and collision rate. To ensure fairness, the planning horizon is adjusted to $3.0s$. The L2 error measures the deviation between the SDV's planned trajectory and the human driver's actual trajectory, providing a quantitative measure of planning accuracy. The collision rate assesses the model's ability to safely navigate the driving environment without incidents. Results, detailed in Tab. 3, reveal that our SAD method achieves an L2 error and collision rate comparable to those of the state-of-the-art ANN-based methods, underscoring the safety and reliability of our planning approach.

Energy Efficiency. Neuromorphic hardware is able to take advantage of small activation bit-widths and dynamical sparsity [8], and as such, SNNs are able to significantly reduce energy consumption during inference − provided there are sufficiently sparse activation patterns amongst spiking neurons. As detailed in Tab. 3, we present an estimation of the energy usage of each SOTA model based on dynamical sparsity (detailed calculation methods described in Appendix F). Owing to the utilization of spiking neurons, our model achieves substantial energy reductions: **7.33** × less than the Freespace model [51] and **75.03** × lower compared to the ST-P3 model [70]. This exceptional energy efficiency makes our model highly suitable for real-world applications.

Table 4: Ablation Study on different modules for the encoder and the decoder on Planning tasks.

Decoder		Prediction		Results		
MS	**SEW**	**SP**	**DP**	**PQ**	**SQ**	**RQ**
	✓		✓	59.28	0.75	0.44
✓		✓		67.55	13.35	19.77
✓			✓	**68.16**	**16.57**	**24.11**

Table 5: Ablation study on different timestep alignment strategies for the encoder and decoder on perception tasks.

Encoder		Decoder			Results				
SR	**SA**	**SR**	**SA**	**w/o T**	**Drivable**	**Lane**	**Vehicle**	**Pedestrian**	**Avg.**
	✓		✓		43.84	15.25	4.41	1.62	16.28
✓		✓			0.00	0.00	0.00	0.42	0.11
✓				✓	61.80	20.92	31.78	13.46	31.99
✓			✓		**61.81**	**25.31**	**33.89**	**15.24**	**34.06**

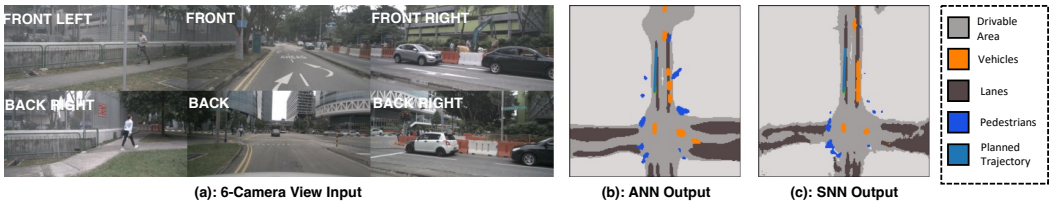

Figure 5: **Qualitative Results of the SAD Model on the nuScenes Dataset.** (a) displays six camera view inputs utilized by the model. (b) illustrates the planning result of the ANN model, and (c) presents the planning results of our SAD model. The comparison shows that our SAD model can achieve performance comparable to that of the ANN model and successfully generate a safe trajectory.

4.2 Visualization

To evaluate the effectiveness of our SAD model in a more interpretable manner, we provide qualitative results using the nuScenes dataset. Fig. 5 illustrates the outputs of our model. The SAD model effectively generates a safe trajectory that enables straight-ahead motion while avoiding collisions with curbsides and the vehicle ahead. Additionally, we conducted a comparative analysis with the ANN model, which demonstrated that our SAD model can achieve comparable performance, ensuring accurate and reliable planning outcomes.More visual results can be found in the Appendix E.3.

4.3 Ablation Study

4.3.1 Timestep Strategy

The large space of architecture design decisions came with a large number of ablation studies before determining the best performing model. We conducted an ablation study to examine the effects of different timestep alignment strategies on the performance of the encoder and decoder in perception tasks. The strategies include 'SR' for the repetition of vision inputs over timesteps, 'SA' where sequential inputs are aligned with the model's inherent temporal dimension, and 'w/o T' where recurrent dynamics in the decoder are removed, thus disconnecting the hidden states between timesteps. The results are summarized in Tab. 5. The last row is the baseline configuration of our model that serves as a reference point for these experiments. All ablation experiments are run for 5 epochs.

Impact of Timestep Repetition in Encoder. (Row 1) When repeating timesteps in the encoder which was pre-trained on the ImageNet-1K dataset for classification tasks, we notice a substantial benefit. This approach leverages the encoder's ability to capture spatial information through repeated images, a technique effective during its pre-training phase. Adopting this strategy in the SAD model enhances perception performance by maintaining consistency with the pre-training approach.

Impact of Timestep Alignment in Decoder. (Row 2) In contrast to the encoder, the decoder, which is trained from scratch, benefits from aligning timesteps with the model's inherent temporal dimension ('SA'). This strategy leverages the decoder's capacity to mix temporal information, improving performance over the repeat strategy ('SR'), which fails to show any performance gains.

Single Timestep Input. (Row 3) The purpose of this experiment was to study how well spiking neurons perform as temporal mixers. In this setup, all inputs are processed in one timestep without inter-frame connections, leading to a slight performance decrease compared to our baseline. Therefore, it confirms that spiking neurons inherently possess temporal processing capabilities.

4.3.2 Effectiveness of Different Modules

Tab. 4 outlines the results of an ablation study that assesses the impact of various structural modifications to the encoder and decoder on planning tasks. The study differentiates between 'MS' for MS-ResNet structure [68], 'SEW' for SEW-ResNet structure [79], 'SP' for single pathway model, and 'DP' for dual pathway model. We analyze the planning performance of each configuration. The last row represents the configuration of our final model, which we use as the baseline for comparison.

Decoder Structure Evaluation. (Row 1) This part of the study compares the MS-ResNet and SEW-ResNet structures in their roles as decoders, where both of them are mainly used in SNN

https://doi.org/10.52202/079017-4346

architectures. The difference between MS-ResNet and SEW-ResNet can be found in Appendix B. Our results indicate that the MS-ResNet structure is more effective for planning tasks, likely due to its enhanced capability to handle the spatial-temporal dynamics required in this context.

Pathway Model Comparison. (Row 2) Here, we explore the performance difference between single and dual pathway prediction models. The dual pathway model integrates information through two distinct processing streams. Our experimental results show that the dual pathway model significantly outperforms the single pathway model in planning tasks.

5 Conclusion

In this work, we presented Spiking Autonomous Driving (SAD), the first end-to-end spiking neural network designed for autonomous driving. By integrating perception, prediction, and planning into a unified neuromorphic framework, SAD demonstrates competitive performance on the nuScenes dataset while exhibiting exceptional energy efficiency compared to state-of-the-art ANN-based methods. The perception module effectively constructs interpretable bird's eye view representations, the prediction module accurately forecasts future states using a novel dual-pathway architecture with spiking neurons, and the planning module generates safe and comfortable trajectories. Crucially, SAD showcases the immense potential of SNNs in complex real-world applications, marking a significant step towards realizing low-power, intelligent systems for safety-critical domains like autonomous vehicles. However, further validation through real-world on-vehicle testing is necessary. Moving forward, we believe this work will inspire further research into neuromorphic computing for sustainable and robust autonomous driving solutions.

Acknowledgements

This work was supported in part by National Science Foundation (NSF) awards CNS-1730158, ACI-1540112, ACI-1541349, OAC-1826967, OAC-2112167, CNS-2100237, CNS-2120019, the University of California Office of the President, and the University of California San Diego's California Institute for Telecommunications and Information Technology/Qualcomm Institute. Thanks to CENIC for the 100Gbps networks. This project was also supported in-part by the National Science Foundation RCN-SC 2332166.

References

[1] SAE Committee. Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems, 2014.

[2] Kshitij Bhardwaj, Zishen Wan, Arijit Raychowdhury, and Ryan Goldhahn. Real-time fully unsupervised domain adaptation for lane detection in autonomous driving. In *2023 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1–2, 2023.

[3] Soumya Sudhakar, Vivienne Sze, and Sertac Karaman. Data centers on wheels: Emissions from computing onboard autonomous vehicles. *IEEE Micro*, 43(1):29–39, 2022.

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

[5] Jason K Eshraghian, Max Ward, Emre O Neftci, Xinxin Wang, Gregor Lenz, Girish Dwivedi, Mohammed Bennamoun, Doo Seok Jeong, and Wei D Lu. Training spiking neural networks using lessons from deep learning. *Proceedings of the IEEE*, 2023.

[6] Guoqi Li, Lei Deng, Huajing Tang, Gang Pan, Yonghong Tian, Kaushik Roy, and Wolfgang Maass. Brain inspired computing: A systematic survey and future trends. *Authorea Preprints*, 2023.

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

[8] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Pradip Joshi, Nabil Imam, Shweta Jain, et al. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99, 2018.

[9] Mostafa Rahimi Azghadi, Corey Lammie, Jason K Eshraghian, Melika Payvand, Elisa Donati, Bernabe Linares-Barranco, and Giacomo Indiveri. Hardware implementation of deep network accelerators towards healthcare and biomedical applications. *IEEE Transactions on Biomedical Circuits and Systems*, 14(6):1138–1159, 2020.

[10] Charlotte Frenkel, Jean-Didier Legat, and David Bol. Morphic: A 65-nm 738k-synapse/mm2 quad-core binary-weight digital neuromorphic processor with stochastic spike-driven online learning. *IEEE transactions on biomedical circuits and systems*, 13(5):999–1010, 2019.

[11] Fabrizio Ottati, Chang Gao, Qinyu Chen, Giovanni Brignone, Mario R Casu, Jason K Eshraghian, and Luciano Lavagno. To spike or not to spike: A digital hardware perspective on deep learning acceleration. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 2023.

[12] Wei Fang, Zhaofei Yu, Yanqi Chen, Timothée Masquelier, Tiejun Huang, and Yonghong Tian. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2661–2671, October 2021.

[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] Zhaokun Zhou, Yuesheng Zhu, Chao He, Yaowei Wang, Shuicheng Yan, Yonghong Tian, and Li Yuan. Spikformer: When spiking neural network meets transformer. *arXiv preprint arXiv:2209.15425*, 2022.

[15] Rui-Jie Zhu, Qihang Zhao, Tianjing Zhang, Haoyu Deng, Yule Duan, Malu Zhang, and Liang-Jian Deng. Tcja-snn: Temporal-channel joint attention for spiking neural networks. *arXiv preprint arXiv:2206.10177*, 2022.

[16] Ziqing Wang, Yuetong Fang, Jiahang Cao, Qiang Zhang, Zhongrui Wang, and Renjing Xu. Masked spiking transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1761–1771, 2023.

[17] Ziqing Wang, Qidong Zhao, Jinku Cui, Xu Liu, and Dongkuan Xu. Autost: Training-free neural architecture search for spiking transformers. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3455–3459. IEEE, 2024.

[18] Qiaoyi Su, Yuhong Chou, Yifan Hu, Jianing Li, Shijie Mei, Ziyang Zhang, and Guoqi Li. Deep directly-trained spiking neural networks for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6555–6565, 2023.

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

[20] Man Yao, JiaKui Hu, Tianxiang Hu, Yifan Xu, Zhaokun Zhou, Yonghong Tian, XU Bo, and Guoqi Li. Spike-driven transformer v2: Meta spiking neural network architecture inspiring the design of next-generation neuromorphic chips. In *The Twelfth International Conference on Learning Representations*, 2023.

[21] Youngeun Kim, Joshua Chough, and Priyadarshini Panda. Beyond classification: Directly training spiking neural networks for semantic segmentation. *Neuromorphic Computing and Engineering*, 2(4):044015, 2022.

[22] Xue-Rui Qiu, Zhao-Rui Wang, Zheng Luan, Rui-Jie Zhu, Ma-Lu Zhang, and Liang-Jian Deng. Vtsnn: a virtual temporal spiking neural network. *Frontiers in neuroscience*, 17:1091097, 2023.

 https://doi.org/10.52202/079017-4346

[23] Hiromichi Kamata, Yusuke Mukuta, and Tatsuya Harada. Fully spiking variational autoencoder. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 7059–7067, 2022.

[24] Qiugang Zhan, Xiurui Xie, Guisong Liu, and Malu Zhang. Esvae: An efficient spiking variational autoencoder with reparameterizable poisson spiking sampling. *arXiv preprint arXiv:2310.14839*, 2023.

[25] Xuerui Qiu, Zheng Luan, Zhaorui Wang, and Rui-Jie Zhu. When spiking neural networks meet temporal attention image decoding and adaptive spiking neuron. 2023.

[26] Jiahang Cao, Ziqing Wang, Hanzhong Guo, Hao Cheng, Qiang Zhang, and Renjing Xu. Spiking denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 4912–4921, 2024.

[27] Rui-Jie Zhu, Qihang Zhao, Guoqi Li, and Jason K Eshraghian. Spikegpt: Generative pre-trained language model with spiking neural networks. *arXiv preprint arXiv:2302.13939*, 2023.

[28] Changze Lv, Tianlong Li, Jianhan Xu, Chenxi Gu, Zixuan Ling, Cenyuan Zhang, Xiaoqing Zheng, and Xuanjing Huang. Spikebert: A language spikformer trained with two-stage knowledge distillation from bert. *arXiv preprint arXiv:2308.15122*, 2023.

[29] Malyaban Bal and Abhronil Sengupta. Spikingbert: Distilling bert to train spiking language models using implicit differentiation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 10998–11006, 2024.

[30] Alexander Henkes, Jason K Eshraghian, and Henning Wessels. Spiking neural networks for nonlinear regression. *arXiv preprint arXiv:2210.03515*, 2022.

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

[32] Samuel Schmidgall, Jascha Achterberg, Thomas Miconi, Louis Kirsch, Rojin Ziaei, S Hajiseyedrazi, and Jason Eshraghian. Brain-inspired learning in artificial neural networks: a review. *arXiv preprint arXiv:2305.11252*, 2023.

[33] Antonio Vitale, Alpha Renner, Celine Nauer, Davide Scaramuzza, and Yulia Sandamirskaya. Event-driven vision and control for uavs on a neuromorphic chip. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 103–109. IEEE, 2021.

[34] Sebastian Glatz, Julien Martel, Raphaela Kreiser, Ning Qiao, and Yulia Sandamirskaya. Adaptive motor control and learning in a spiking neural network realised on a mixed-signal neuromorphic processor. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9631–9637. IEEE, 2019.

[35] Rasmus Karnøe Stagsted, Antonio Vitale, Alpha Renner, Leon Bonde Larsen, Anders Lyhne Christensen, and Yulia Sandamirskaya. Event-based pid controller fully realized in neuromorphic hardware: A one dof study. In *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 10939–10944. IEEE, 2020.

[36] Rasmus Stagsted, Antonio Vitale, Jonas Binz, Leon Bonde Larsen, Yulia Sandamirskaya, et al. Towards neuromorphic control: A spiking neural network based pid controller for uav. RSS, 2020.

[37] Tim Burgers, Stein Stroobants, and Guido de Croon. Evolving spiking neural networks to mimic pid control for autonomous blimps. *arXiv preprint arXiv:2309.12937*, 2023.

[38] Zhenshan Bing, Claus Meschede, Kai Huang, Guang Chen, Florian Rohrbein, Mahmoud Akl, and Alois Knoll. End to end learning of spiking neural network based on r-stdp for a lane keeping vehicle. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 4725–4732. IEEE, 2018.

[39] Raz Halaly and Elishai Ezra Tsur. Autonomous driving controllers with neuromorphic spiking neural networks. *Frontiers in Neurorobotics*, 17:1234962, 2023.

[40] Jacques Kaiser, J Camilo Vasquez Tieck, Christian Hubschneider, Peter Wolf, Michael Weber, Michael Hoff, Alexander Friedrich, Konrad Wojtasik, Arne Roennau, Ralf Kohlhaas, et al. Towards a framework for end-to-end control of a simulated vehicle with spiking neural networks. In *2016 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, pages 127–134. IEEE, 2016.

[41] Albert Shalumov, Raz Halaly, and Elishai Ezra Tsur. Lidar-driven spiking neural network for collision avoidance in autonomous driving. *Bioinspiration & Biomimetics*, 16(6):066016, 2021.

[42] Li Chen, Penghao Wu, Kashyap Chitta, Bernhard Jaeger, Andreas Geiger, and Hongyang Li. End-to-end autonomous driving: Challenges and frontiers. *arXiv preprint arXiv:2306.16927*, 2023.

[43] Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. Learning by cheating. In *Conference on Robot Learning*, pages 66–75. PMLR, 2020.

[44] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[45] Felipe Codevilla, Matthias Müller, Antonio López, Vladlen Koltun, and Alexey Dosovitskiy. End-to-end driving via conditional imitation learning. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 4693–4700. IEEE, 2018.

[46] Jeffrey Hawke, Richard Shen, Corina Gurau, Siddharth Sharma, Daniele Reda, Nikolay Nikolov, Przemysław Mazur, Sean Micklethwaite, Nicolas Griffiths, Amar Shah, et al. Urban driving with conditional imitation learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 251–257. IEEE, 2020.

[47] Felipe Codevilla, Eder Santana, Antonio M López, and Adrien Gaidon. Exploring the limitations of behavior cloning for autonomous driving. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9329–9338, 2019.

[48] Wenyuan Zeng, Wenjie Luo, Simon Suo, Abbas Sadat, Bin Yang, Sergio Casas, and Raquel Urtasun. End-to-end interpretable neural motion planner. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8660–8669, 2019.

[49] Abbas Sadat, Sergio Casas, Mengye Ren, Xinyu Wu, Pranaab Dhawan, and Raquel Urtasun. Perceive, predict, and plan: Safe motion planning through interpretable semantic representations. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIII 16*, pages 414–430. Springer, 2020.

[50] Hao Wang, Peng Cai, Rui Fan, Yuxiang Sun, and Ming Liu. End-to-end interactive prediction and planning with optical flow distillation for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2199–2208, 2021.

[51] Peiyun Hu, Anthony Huang, John Dolan, David Held, and Deva Ramanan. Safe local motion planning with self-supervised freespace forecasting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13270–13279, 2021.

[52] Tarasha Khurana, Peiyun Hu, Achal Dave, Jason Ziglar, David Held, and Deva Ramanan. Differentiable raycasting for self-supervised occupancy forecasting. In *European Conference on Computer Vision*, pages 238–254. Springer, 2022.

[53] Alex Kendall, Jeffrey Hawke, David Janz, Przemyslaw Mazur, Daniele Reda, John-Mark Allen, Vinh-Dieu Lam, Alex Bewley, and Amar Shah. Learning to drive in a day. In *2019 international conference on robotics and automation (ICRA)*, pages 8248–8254. IEEE, 2019.

[54] Xiaodan Liang, Tairui Wang, Luona Yang, and Eric Xing. Cirl: Controllable imitative reinforcement learning for vision-based self-driving. In *Proceedings of the European conference on computer vision (ECCV)*, pages 584–599, 2018.

 https://doi.org/10.52202/079017-4346

[55] Marin Toromanoff, Emilie Wirbel, and Fabien Moutarde. End-to-end model-free reinforcement learning for urban driving using implicit affordances. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7153–7162, 2020.

[56] Raphael Chekroun, Marin Toromanoff, Sascha Hornauer, and Fabien Moutarde. Gri: General reinforced imitation and its application to vision-based autonomous driving. *arXiv preprint arXiv:2111.08575*, 2021.

[57] Aditya Prakash, Kashyap Chitta, and Andreas Geiger. Multi-modal fusion transformer for end-to-end autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7077–7087, 2021.

[58] Kashyap Chitta, Aditya Prakash, Bernhard Jaeger, Zehao Yu, Katrin Renz, and Andreas Geiger. Transfuser: Imitation with transformer-based sensor fusion for autonomous driving. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[59] Hao Shao, Letian Wang, Ruobing Chen, Hongsheng Li, and Yu Liu. Safety-enhanced autonomous driving using interpretable sensor fusion transformer. In *Conference on Robot Learning*, pages 726–737. PMLR, 2023.

[60] Xiaosong Jia, Penghao Wu, Li Chen, Jiangwei Xie, Conghui He, Junchi Yan, and Hongyang Li. Think twice before driving: Towards scalable decoders for end-to-end autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21983–21994, 2023.

[61] Bernhard Jaeger, Kashyap Chitta, and Andreas Geiger. Hidden biases of end-to-end driving models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8240–8249, 2023.

[62] Kashyap Chitta, Aditya Prakash, and Andreas Geiger. Neat: Neural attention fields for end-to-end autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15793–15802, 2021.

[63] Dian Chen, Vladlen Koltun, and Philipp Kr"ahenb"uhl. Learning to drive from a world on rails. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15590–15599, 2021.

[64] Zhejun Zhang, Alexander Liniger, Dengxin Dai, Fisher Yu, and Luc Van Gool. End-to-end urban driving by imitating a reinforcement learning coach. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 15222–15232, 2021.

[65] Penghao Wu, Xiaosong Jia, Li Chen, Junchi Yan, Hongyang Li, and Yu Qiao. Trajectory-guided control prediction for end-to-end autonomous driving: A simple yet strong baseline. *Advances in Neural Information Processing Systems*, 35:6119–6132, 2022.

[66] Jimuyang Zhang, Zanming Huang, and Eshed Ohn-Bar. Coaching a teachable student. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7805–7815, 2023.

[67] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[68] 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*, 2024.

[69] Philip Polack, Florent Altché, Brigitte d'Andréa Novel, and Arnaud de La Fortelle. The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles? In *2017 IEEE intelligent vehicles symposium (IV)*, pages 812–818. IEEE, 2017.

[70] Shengchao Hu, Li Chen, Penghao Wu, Hongyang Li, Junchi Yan, and Dacheng Tao. St-p3: End-to-end vision-based autonomous driving via spatial-temporal feature learning. In *European Conference on Computer Vision*, pages 533–549. Springer, 2022.

[71] Ali Lotfi Rezaabad and Sriram Vishwanath. Long short-term memory spiking networks and their applications. In *International Conference on Neuromorphic Systems 2020*, pages 1–9, 2020.

[72] Chenyang Lu, Marinus Jacobus Gerardus Van De Molengraft, and Gijs Dubbelman. Monocular semantic occupancy grid mapping with convolutional variational encoder–decoder networks. *IEEE Robotics and Automation Letters*, 4(2):445–452, 2019.

[73] Bowen Pan, Jiankai Sun, Ho Yin Tiga Leung, Alex Andonian, and Bolei Zhou. Cross-view semantic segmentation for sensing surroundings. *IEEE Robotics and Automation Letters*, 5(3):4867–4873, 2020.

[74] Thomas Roddick and Roberto Cipolla. Predicting semantic map representations from images using pyramid occupancy networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11138–11147, 2020.

[75] Jonah Philion and Sanja Fidler. Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV 16*, pages 194–210. Springer, 2020.

[76] Hengli Wang, Peide Cai, Yuxiang Sun, Lujia Wang, and Ming Liu. Learning interpretable end-to-end vision-based motion planning for autonomous driving with optical flow distillation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13731–13737. IEEE, 2021.

[77] Anthony Hu, Zak Murez, Nikhil Mohan, Sofía Dudas, Jeffrey Hawke, Vijay Badrinarayanan, Roberto Cipolla, and Alex Kendall. Fiery: Future instance prediction in bird's-eye view from surround monocular cameras. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15273–15282, 2021.

[78] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[79] 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.

[80] Nitin Rathi, Gopalakrishnan Srinivasan, Priyadarshini Panda, and Kaushik Roy. Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation. *arXiv preprint arXiv:2005.01807*, 2020.

[81] Jibin Wu, Chenglin Xu, Xiao Han, Daquan Zhou, Malu Zhang, Haizhou Li, and Kay Chen Tan. Progressive tandem learning for pattern recognition with deep spiking neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):7824–7840, 2021.

[82] Man Yao, Guangshe Zhao, Hengyu Zhang, Yifan Hu, Lei Deng, Yonghong Tian, Bo Xu, and Guoqi Li. Attention spiking neural networks. *IEEE transactions on pattern analysis and machine intelligence*, 2023.

[83] Dahun Kim, Sanghyun Woo, Joon-Young Lee, and In So Kweon. Video panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9859–9868, 2020.

 https://doi.org/10.52202/079017-4346

APPENDIX

A Encoder

The encoder consists of 12 layers, each containing a 2D convolution layer, batch normalization, and spiking neuron. The output of the encoder is a feature map $\mathcal{F} \in \{0,1\}^{C_f \times L \times H \times W}$ and a depth estimation $\mathcal{D} \in \{0,1\}^{C_d \times L \times H \times W}$, where C_f is the number of feature channels, C_d is the number of channels, each of which has a depth associated with it, and (H, W) is the spatial size. Formally, given an image sequence I:

$$X = \mathcal{SN}(\mathrm{BN}(\mathrm{Conv2d}(I))), \tag{10}$$

where Conv2d represents a 2D convolutional layer (stride: 1, 3×3 kernel size), BN is batch normalization, and $\mathcal{MP}$ is a max-pooling operator. The feature map $\mathcal{F}$ and depth estimation $\mathcal{D}$ are then averaged over the sequence T and combined using an outer product to obtain a camera feature frustum:

$$Y = \mathcal{F} \otimes \mathcal{D}, Y \in \{0,1\}^{C_f \times C_d \times T \times L \times H \times W} \tag{11}$$

The frustums from all cameras are transformed into a global 3D coordinate system centered at the ego-vehicle's inertial center at time i. Previous BEV feature maps are merged with the current BEV by applying a discount factor α to integrate these layers efficiently.

$$\tilde{x}_t = b_t + \sum_{i=1}^{t-1} \alpha^i \times \tilde{x}_{t-i} \tag{12}$$

where $\tilde{x}_t$ is the BEV at time t which has an initial condition of $\tilde{x}_1 = b_1$, and the discount factor is $\alpha = 0.5$. We then average across the T dimension to eliminate the repeated temporal dimension and obtain the average firing rate. The resulting feature map is then be passed to the decoder.

B Spiking ResNet Architecture

MS-ResNet [68] and SEW-ResNet [79] are two directly trained residual learning methods for SNNs, primarily aiming to overcome the degradation problem in training deep SNNs. In classic CNNs, ResNet achieves "very deep" neural networks by attaching an identity mapping skip connection throughout the entire network. However, directly copying the classic residual structure to SNNs leads to the degradation problem, where the training loss increases as the network deepens. The underlying reason is that CNNs generate continuous values through activation functions such as ReLU, while SNNs output discrete spikes.

To address this issue, SEW-ResNet and MS-ResNet establish shortcut connections between spikes or membrane potentials of different layers, respectively. SEW-ResNet builds shortcuts between the output spikes of different layers, thereby obtaining an identity mapping. On the other hand, MS-ResNet constructs shortcuts between the membrane potentials of spiking neurons in different layers. Fig. 6 illustrates the differences between these two methods. By building residual learning on membrane potentials, MS-ResNet achieves higher accuracy than SEW-ResNet. In summary, SEW-ResNet and MS-ResNet effectively alleviate the degradation problem faced by deep SNNs by cleverly constructing residual connections suitable for SNN characteristics. These methods significantly improve the performance of SNNs, further narrowing the gap with ANNs of the same scale, and provide effective ideas for building large-scale, high-performance SNNs.

C Detail of Cost Function

In the method section, we briefly introduced the Cost Function f, a multi-component function designed to evaluate the suitability of planned trajectories for self-driving vehicles. We follow the cost function in ST-P3 [70], each component of the cost function addresses a specific aspect of the trajectory planning, as detailed below:

Safety Cost. This component ensures that the planned trajectories do not result in collisions. The model must account for the dynamic positioning of other vehicles and road elements, avoiding overlap

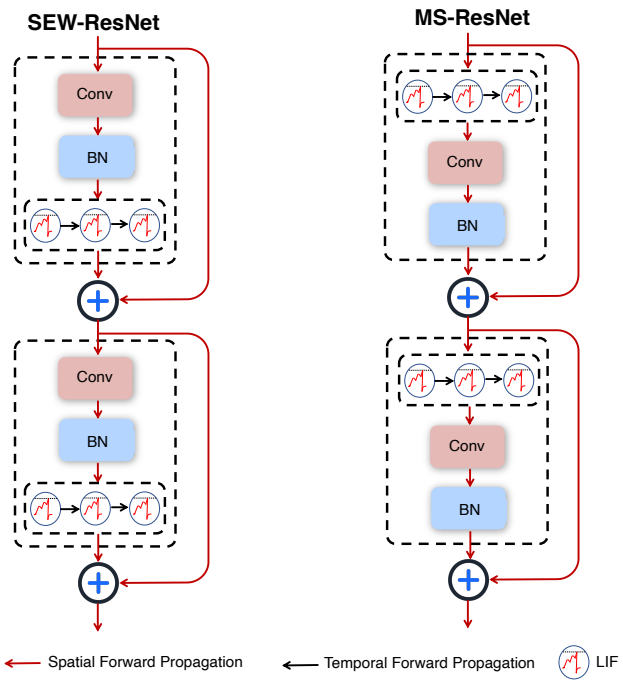

Figure 6: MS-ResNet and SEW-ResNet.

with the grids they occupy. Furthermore, a safe distance must be maintained, especially at high velocities, to prevent any potential accidents.

Cost Volume. Road environments are complex, with numerous unpredictable elements affecting trajectory planning. It is impractical to manually enumerate all possible scenarios and associated costs. To address this, we utilize a learned cost volume produced by the prediction module head mentioned earlier. To prevent the cost volume from disproportionately influencing the trajectory evaluation, we clip its values to maintain balanced decision-making.

Comfort and Progress. To ensure that the trajectories are not only safe but also comfortable for passengers, this component penalizes excessive lateral acceleration, jerk, and high curvature. Additionally, the efficiency of the trajectory is critical; thus, paths that effectively progress towards the target destination receive positive reinforcement.

These components collectively ensure that the cost function comprehensively assesses both the safety and quality of the trajectories planned by the self-driving vehicle.

D Training

D.1 Pretraining on Spiking Token Mixer (STM)

In STM, the architecture is similar to Spiking Transformers, consisting of a Spiking Patch Embedding (SPS) layer followed by a Transformer-like architecture. However, unlike Spiking Transformers, the SPS layer in STM is not followed by a self-attention layer. We pre-train the STM on ImageNet-1K for 300 epochs, aligning with other SNNs that are pre-trained on ImageNet. The input size is set to 224×224, and the batch size is set to 128 or 256 during the 310 training epochs with a cosine-decay learning rate whose initial value is 0.0005. The optimizer used is Lamb. The SPS module divides the image into $N = 196$ patches. Standard data augmentation techniques, such as random augmentation and mixup, are also employed during training. We compare our method with other SNN methods of similar size trained on ImageNet-1K, as shown in Tab. 6.

https://doi.org/10.52202/079017-4346

Table 6: Performance on ImageNet-1K [67]. Note, “Spike", “Para", and “Step" denote “Spike-driven", “Parameters", and “Timestep".

Methods	Architecture	Spike	Param (M)	Step	Acc.(%)
ANN2SNN	ResNet-34 [80]	✓	21.8	250	61.5
	VGG-16 [81]	✓	-	16	65.1
SCNN	SEW-ResNet [79]	✗	25.6	4	67.8
	MS-ResNet [68]	✓	21.8	4	69.4
	Att-MS-ResNet [82]	✗	22.1	1	69.2
Spiking-Transformer	Spikformer [14]	✗	16.8	4	70.2
	Spike-driven Transformer [13]	✓	16.8	4	72.3
SMLP	**STM(Ours)**	✓	12.2	4	72.1

D.2 Stage-wise Training of the End-to-end Model

Although our model is end-to-end differentiable, directly implementing end-to-end training can lead to unstable loss convergence. To mitigate this issue, we divide our training process into three stages:

Stage 1: Perception Module Training In the first stage, we focus on training the perception module while disabling the prediction and planning modules. This means that the prediction module within the network remains inactive, and the decoder does not output instance flow information related to prediction. During this stage, we use a learning rate of $1e-3$ and train the model for 20 epochs.

Stage 2: Prediction Module Training In the second stage, we enable the prediction module, which predicts the movement of instances on the BEV for the next three steps. We train the model for an additional 10 epochs during this stage.

Stage 3: Planning Module Training Finally, we start training the planning module, using real trajectories to supervise the model's planning results. In both the prediction and planning stages, we set the learning rate to $2e-4$ to fine-tune the model from the perception stage. Throughout all stages, we employ the Adam optimizer to update the model parameters. By incrementally introducing the prediction and planning modules, we ensure that each component is well-trained before integrating them into the end-to-end framework.

D.3 Overall Loss for End-to-End Learning

Our model integrates perception, prediction, and planning into a unified framework optimized end-to-end using a composite loss function:

$$\mathcal{L} = \mathcal{L}_{per} + \alpha\mathcal{L}_{pre} + \beta\mathcal{L}_{pla} \tag{13}$$

where α and β are learnable weights. These parameters dynamically adjust to scale the contributions of each task based on the gradients from the respective task losses.

Perception Loss. This component is multi-faceted, covering segmentation for current and previous frames, mapping, and auxiliary depth estimation. For semantic segmentation, a top-k cross-entropy loss is employed to effectively handle the large amount of background, non-salient content in the BEV images, following the approach in [77]. Instance segmentation utilizes an l_2 loss for centerness supervision and an l_1 loss for offsets and flows. Lane and drivable area predictions are evaluated using a cross-entropy loss.

Prediction Loss. Our prediction module extends the perception task by forecasting future semantic and instance segmentation. It adopts the same top-k cross-entropy loss used in perception but applies an exponential discount to future timestamps to account for increasing uncertainty in predictions.

Planning Loss. The planning module begins by selecting the initial best trajectory τ^* from a set of sampled trajectories as defined in Eq. 5. This trajectory is then refined using an SGRU-based network

to produce the final trajectory output τ_o^*. The planning loss comprises two parts: a max-margin loss that differentiates between the expert behavior τ_h (treated as a positive example) and other sampled trajectories (treated as negatives), and an l_1 distance loss that minimizes the deviation between the refined trajectory and the expert trajectory.

Further details on these loss components and the training are provided in the Appendix D.2.

E Experiments

E.1 Datasets

The nuScenes dataset [78] is a comprehensive, publicly available dataset tailored for autonomous driving research. It encompasses 1,000 manually selected driving scenes from Boston and Singapore—cities renowned for their dense traffic and complex driving environments. Each scene, lasting 20 seconds, showcases a variety of driving maneuvers, traffic conditions, and unforeseen events, reflecting the intricate dynamics of urban driving. The dataset aims to foster the development of advanced methods that ensure safety in densely populated urban settings, featuring dozens of objects per scene. It includes around 1.4 million camera images, 390,000 LIDAR sweeps, 1.4 million RADAR sweeps, and 1.4 million object-bounding boxes across 40,000 keyframes.

Data Preprocessing. In preparation for modeling, we preprocess the images by cropping and resizing the original dimensions from $\mathbb{R}^{3\times900\times1600}$ to $\mathbb{R}^{3\times224\times480}$. Additionally, we incorporate temporal data by including the past three frames for each sequence, denoted as $I_t^n \in \mathbb{R}^{3\times224\times480}$, where $t \in \{1, 2, 3\}$ represents the frame indices, and $n \in \{1, \ldots, 6\}$ indexes the cameras.

E.2 Evaluation Metrics

In this section, we introduce the metrics used to assess the predictive capabilities of our model for video prediction. Specifically, we utilize Intersection over Union (IoU), Panoptic Quality (PQ), Recognition Quality (RQ), and Segmentation Quality (SQ) as defined by Kim et al. [83].

Intersection over Union (IoU) The IoU metric evaluates the overlap between the predicted and ground truth segmentation masks. It is calculated as:

$$IoU = \frac{|\text{Prediction} \cap \text{Ground Truth}|}{|\text{Prediction} \cup \text{Ground Truth}|} \tag{14}$$

where $|\cdot|$ denotes the number of pixels in the respective set.

Panoptic Quality (PQ) Panoptic Quality (PQ) is a comprehensive metric that combines both segmentation and recognition quality. It accounts for both true positive matches and penalizes false positives and false negatives. PQ is defined as:

$$PQ = \frac{\sum_{(p,g)\in TP} IoU(p, g)}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}$$

where TP represents the set of true positive matches between predicted segments p and ground truth segments g, FP denotes false positives, and FN denotes false negatives.

Recognition Quality (RQ) Recognition Quality (RQ) measures the accuracy of object recognition and classification. It is given by:

$$RQ = \frac{|TP|}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}$$

Segmentation Quality (SQ) Segmentation Quality (SQ) evaluates the quality of the predicted segments' spatial accuracy. It is defined as:

$$SQ = \frac{\sum_{(p,g)\in TP} IoU(p, g)}{|TP|}$$

 https://doi.org/10.52202/079017-4346

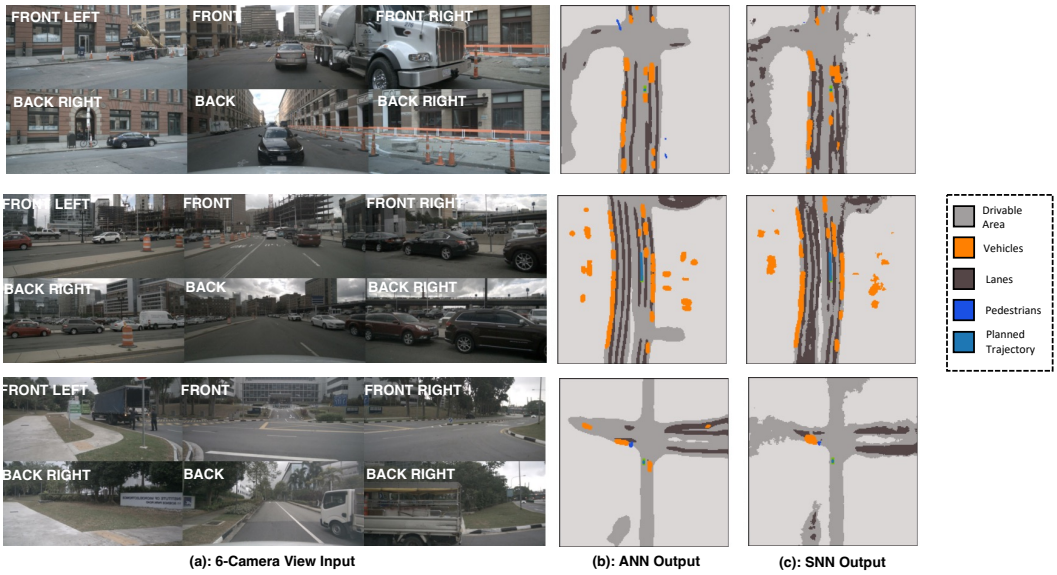

Figure 7: **More Qualitative Results of the SAD Model on the nuScenes Dataset.** (a) displays six camera view inputs utilized by the model. (b) illustrates the planning result of the ANN model, and (c) presents the planning results of our SAD model. The comparison shows that our SAD model can achieve performance comparable to that of the ANN model and successfully generate a safe trajectory.

E.3 More Visualizations

we present three additional visual examples in Fig. 7 to further showcase the performance of our SAD model on the nuScenes dataset. These examples demonstrate the model's ability to generate safe and collision-free trajectories in various driving scenarios, highlighting its effectiveness in autonomous driving applications.

F Theoretical Energy Consumption Calculation

In the experimental section of our paper, we utilize energy consumption as a metric to evaluate the efficiency of various models. This appendix outlines the methodology employed to compute the theoretical energy consumption of our SNN architecture. The calculation process involves two main steps: first, determining the synaptic operations (SOPs) for each architectural component, and second, estimating the cumulative energy consumption based on these operations.

The synaptic operations within each block of the SNN are calculated using the following equation:

$$\mathrm{SOPs}(l) = fr \times T \times \mathrm{FLOPs}(l) \tag{15}$$

where l denotes the ordinal number of the block within the SNN, fr represents the firing rate of the input spike train to the block, T indicates the time step of the neuron, and $\mathrm{FLOPs}(l)$ refers to the floating-point operations within the block, primarily consisting of multiply-and-accumulate (MAC) operations.

To compute the SNN's theoretical energy consumption, we consider both MAC and spike-based accumulation (AC) operations, utilizing 45 nm technology. The energy costs are $E_{MAC} = 4.6\,\mathrm{pJ}$ for MAC operations and $E_{AC} = 0.9\,\mathrm{pJ}$ for AC operations. Based on the methodologies outlined in [82], we calculate the energy consumption of the SNN as follows:

$$\begin{aligned} E_{SNN} = {} & E_{MAC} \times \mathrm{FLOP}^{1}_{\mathrm{SNN_{Conv}}} \\ & + E_{AC} \times \left(\sum_{n=2}^{N} \mathrm{SOP}^{n}_{\mathrm{SNN_{Conv}}} + \sum_{m=1}^{M} \mathrm{SOP}^{m}_{\mathrm{SNN_{FC}}} \right) \end{aligned} \tag{16}$$

Here, N and M denote the number of convolutional (Conv) and fully connected (FC) layers, respectively. The terms $\mathrm{FLOP}_{\mathrm{SNN}_{\mathrm{Conv}}}$ and $\mathrm{SOP}_{\mathrm{SNN}_{\mathrm{Conv}}}$, $\mathrm{SOP}_{\mathrm{SNN}_{\mathrm{FC}}}$ represent the FLOPs for the first Conv layer and SOPs for the subsequent n^{th} Conv and m^{th} FC layers.

For the ANN model, the formula for computing the theoretical energy consumption is more straightforward, as all neurons operate in floating point:

$$E_{ANN} = E_{MAC} \times \mathrm{FLOP}_{\mathrm{ANN}} \tag{17}$$

Here, $\mathrm{FLOP}_{\mathrm{ANN}}$ represents the FLOPs of the ANN model.

 https://doi.org/10.52202/079017-4346

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction reflect the paper's contributions and scope clearly.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitation of this work in the Conclusion section.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: This paper does not include theoretical results that should be proved.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: All training details have been revealed in Sec. 4.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We have attached the anonymized code for reproducing our product in supplementary material.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: All training and test details have been revealed in Sec. 4.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We do not have enough computational resources to run it multiple times to generate error bars.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper provides detailed information on the compute resources used.

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This work follows the NeurIPS Code of Ethics.

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: There is no societal impact of the work performed.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper properly credits creators and mentions the license and terms of use for existing assets.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

https://doi.org/10.52202/079017-4346