SpikingVTG: A Spiking Detection Transformer for Video Temporal Grounding

Malyaban Bal[1,2]*, **Brian Matejek**[2], **Susmit Jha**[2], **Adam D. Cobb**[2]

[1]The Pennsylvania State University
[2]Computer Science Laboratory, SRI International

Abstract

Video Temporal Grounding (VTG) aims to retrieve precise temporal segments in a video conditioned on natural language queries. Unlike conventional neural frameworks that rely heavily on computationally expensive dense matrix multiplications, Spiking Neural Networks (SNNs)—previously underexplored in this domain—offer a unique opportunity to tackle VTG tasks through bio-plausible spike-based communication and an event-driven accumulation-based computational paradigm. We introduce SpikingVTG, a multi-modal spiking detection transformer, designed to harness the computational simplicity and sparsity of SNNs for VTG tasks. Leveraging the temporal dynamics of SNNs, our model introduces a Saliency Feedback Gating (SFG) mechanism that assigns dynamic saliency scores to video clips and applies multiplicative gating to highlight relevant clips while suppressing less informative ones. SFG enhances performance and reduces computational overhead by minimizing neural activity. We analyze the layer-wise convergence dynamics of SFG-enabled model and apply implicit differentiation at equilibrium to enable efficient, BPTT-free training. To improve generalization and maximize performance, we enable knowledge transfer by optimizing a Cos-L2 representation matching loss that aligns the layer-wise representation and attention maps of a non-spiking teacher with those of our student SpikingVTG. Additionally, we present Normalization-Free (NF)-SpikingVTG, which eliminates non-local operations like softmax and layer normalization, and an extremely quantized 1-bit (NF)-SpikingVTG variant for potential deployment on edge devices. Our models achieve competitive results on QVHighlights, Charades-STA, TACoS, and YouTube Highlights, establishing a strong baseline for multi-modal spiking VTG solutions.

1 Introduction

The rapid expansion of various social medias and portable smart technologies has triggered an unprecedented surge in video content. This vast influx of data has intensified the need for efficient methods to retrieve and analyze video information. Consequently, the field of Video Temporal Grounding (VTG) [1] has emerged as an important area of research. The main objective of VTG is to identify the precise segment of a video that corresponds to a given natural language query, enabling accurate and context-driven video content retrieval. In this paper, we focus on two tasks: moment retrieval [2, 3], which aims to identify video intervals relevant to a given query, and highlight detection [4], which retrieves the best candidate segment of the video in response to the query. Our work involves analyzing multimodal data—combining video content with natural language queries—to develop an effective solution to the problem. With the rise of transformer based architectures the field of VTG has seen significant advancements [5, 6]. However, these models demand substantial

*Work completed while at SRI.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

power and energy [7] to operate. Furthermore, VTG is inherently resource-intensive, requiring the analysis of long video sequences, leading to significant computational overhead. Additionally, applications such as **edge-based event detection** in video—e.g., identifying accidents from traffic cameras—require deploying VTG models on resource-constrained devices near the data source, in order to reduce data transfer to the cloud. These devices often operate under limited energy availability. Inspired by neural dynamics in the brain, this paper leverages bio-plausible neuronal models and learning dynamics to develop an efficient and brain-inspired solution for VTG tasks. Our model enables flexible **inference-time tradeoff** between **energy consumption and accuracy**, making it well-suited for **edge-based deployment in VTG tasks**.

SpikingVTG Model: We introduce SpikingVTG, a spiking detection transformer designed for efficient and accurate VTG. Built on the sparse, event-driven communication and accumulation-based computation of Spiking Neural Networks (SNNs) [8], SpikingVTG leverages the intrinsic dynamics of SNNs to offer a lightweight yet competitive alternative to conventional transformer-based approaches—eliminating the need for costly dense, real-valued matrix multiplications. The architecture comprises three key components: (i) a spiking transformer core, (ii) a Saliency Feedback Gating (SFG) mechanism, and (iii) a spiking decoder for output generation. The spiking transformer captures temporal and cross-modal dependencies, while the decoder produces task-specific outputs. The SFG mechanism addresses a critical challenge unique to VTG: given an input video composed of many clips, how can we identify those most relevant to the query? Tailored specifically for multi-modal VTG tasks, the SFG module leverages the temporal dynamics of SNNs to attend more towards the most salient segments while suppressing irrelevant clips.

Saliency Feedback Gating Mechanism: Operating over discrete time steps, SpikingVTG uses the intermediate spiking activity of the transformer core to dynamically estimate the relevance of each video segment. We compute a feedback-based saliency score for each segment based on the *average spiking rate* (ASR) of the output of the transformer, conditioned on the query. These scores serve as soft attention masks within a multiplicative gating mechanism, suppressing less informative segments to reduce computational overhead while enhancing focus on the most relevant candidate clips. From a neuroscience perspective, **feedback connections** are known to play a critical role in object recognition in the visual cortex [9]. Furthermore, our **feedback mechanism maintains layer-wise convergence of ASR** to equilibrium, enabling us to adopt an efficient training mechanism leveraging the equilibrium dynamics [10]. This approach circumvents the need for computationally intensive BPTT [11], and instead updates model parameters using a single backward pass, significantly improving memory efficiency.

Cos-L2 Representation Matching (CLRM): While our base SpikingVTG model achieves performance comparable to non-pretrained, non-spiking VTG models such as M-DETR and UniVTG [1, 6], achieving state-of-the-art results on VTG tasks typically requires transformer-based, non-spiking models like UniVTG to undergo extensive pre-training. This pre-training significantly boosts their generalization and task-specific capabilities. To enable our SpikingVTG to benefit from similar pre-training advantages—without incurring the high computational cost of training on large-scale datasets like Ego4D, Video-CC [12]— we propose a direction and scale aware CLRM loss for efficient knowledge transfer. Optimizing CLRM loss aligns the hidden states and attention score maps of a pre-trained non-spiking multi-modal transformer with the hidden state converged ASR and mean attention scores of our SpikingVTG model. By minimizing this alignment loss on the downstream task, we allow the student model to imbibe the generalization capability learnt by the pre-trained teacher without having to perform the extensive pre-training from scratch.

Optimizations and Application to VTG Tasks: Traditional transformer-based VTG solutions [6, 13] rely heavily on non-local normalization operations, such as softmax and layer normalization, which pose significant challenges for efficient implementation on neuromorphic hardware [14]. To address this limitation, we develop and evaluate a Normalization-Free (NF)-SpikingVTG model, which eliminates all layer normalization operations and substitutes softmax spiking attention with a ReLU and scaling-based spiking attention mechanism. Although alternative Softmax-free attention approaches have been explored in the literature [15, 16], they have primarily been applied uni-modal vision tasks. We empirically demonstrate that while these optimizations enhance computational efficiency, they result in minimal performance degradation.

To further **reduce computational complexity** and **memory footprint** [17, 18], we introduce 1-bit (NF)-SpikingVTG model, which rely primarily on **integer accumulation operations** and **eliminate**

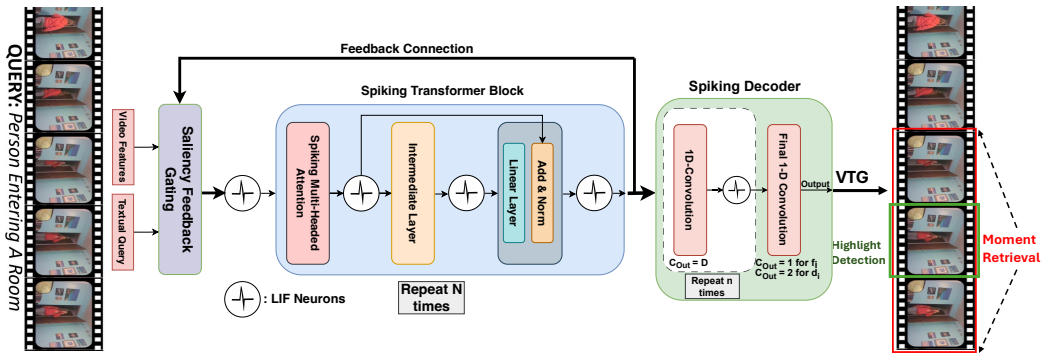

Figure 1: High-level overview of the proposed SpikingVTG model. The model employs a spiking transformer core that utilizes Saliency Feedback Gating through temporal feedback connections. The model also incorporates a spiking decoder module that takes the output of the transformer core to predict parameters for the VTG task.

all non-local operations. This design makes it well-suited for deployment on resource-constrained edge devices. To our knowledge, this work is the first to evaluate an operational spiking detection transformer across VTG tasks, including moment retrieval and highlight detection, on datasets such as QVHighlights, Charades-STA, TACoS and Youtube Highlights. To further highlight the benefits of using SpikingVTG, we present an energy-accuracy tradeoff analysis. This demonstrates that our model is well-suited for deployment on edge devices, where its performance can be dynamically adjusted based on the available energy budget.

2 Related Works

VTG Advancements: Moment-DETR [1], a transformer encoder-decoder model introduced alongside the QVHighlights dataset, laid a strong foundation for subsequent VTG architectures. UMT [5] introduced an unified framework for solving both highlight detection and moment retrieval tasks. Due to the limited availability of trainable video data, UniVTG [6] proposed an innovative solution by unifying various VTG tasks and labels under a single formulation. This enables the development of an LLM-like pretraining framework, achieving state-of-the-art performance on VTG tasks. Although no fully spiking-based model has been explored for VTG tasks, SpikeMba [19]—primarily a non-spiking model—integrates SNN components to generate proposal sets from video data. However, since its core framework is derived from Mamba [20] and relies on floating-point matrix multiplications, SpikeMba cannot be considered a baseline for spiking models, which predominantly use accumulation-based operations. While recent VTG models have significantly improved task-specific performance, adopting a spiking framework enables us to leverage energy-accuracy tradeoff—making suitable solutions on edge devices with limited or dynamic energy supply.

Spiking neural networks (SNNs): SNNs have been implemented in neuromorphic systems like IBM TrueNorth [21] and Intel Loihi 2 [22], demonstrating approximately $75\times$ greater energy efficiency compared to traditional networks running on low-power GPUs [23]. SNNs, with their energy-efficient computational framework, offer a promising solution to the resource-intensive demands of multimodal VTG tasks. While SNNs for a long time were confined in simpler vision-based tasks [24] with simple architectures, recent developments have scaled them to transformer-based models for tasks ranging from vision to language modelling [25, 26, 27], however majority of them are uni-modal and rely on non-local operations not implementable on a neuromorphic chip.

3 Video Temporal Grounding (VTG)

For a given video V and language query Q, we start by segmenting V into a sequence of L_v fixed-length clips, denoted as $\{v_1, \ldots, v_{L_v}\}$. Each clip v_i has a length l and is centered at timestamp t_i. The textual query Q consists of L_q tokens, denoted as $Q = \{q_1, \ldots, q_{L_q}\}$. Following previous studies on VTG [6], we define three parameters for each clip $v_i = (f_i, d_i, s_i)$, where $f_i = 1$ if the

clip is in foreground, i.e. relevant else $f_i = 0$. $d_i = [d_{s_i}, d_{e_i}] \in \mathbb{R}^2$ represent the temporal distance that converts the clip timestamp t_i to its interval boundaries. Here, d_i is valid when $f_i = 1$. The term d_{s_i} denotes the distance between the start of the interval and t_i, while d_{e_i} denotes the distance between the end of the interval and t_i. $s_i \in [0, 1]$ is a continuous score that quantifies the relevance between the visual content of clip v_i and the query Q. Our proposed SpikingVTG predicts these three parameters for each video clip. In this paper, we focus on the following VTG tasks:

Moment Retrieval: We rank the predicted clip boundaries $\{\tilde{b}_i\}_{i=1}^{L_v}$, where $b_i = [t_i - d_{s_i}, t_i + d_{e_i}]$, based on their associated probabilities given by $\{\tilde{f}_i\}_{i=1}^{L_v}$. Since the predicted L_v boundaries are dense, we employ a 1-dimensional Non-Maximum Suppression (NMS) [28] with a threshold of 0.7 to eliminate highly overlapping boundary boxes, resulting in a final prediction.

Highlight Detection For each clip, we rank all clips based on their combined scores $\{\tilde{f}_i + \tilde{s}_i\}_{i=1}^{L_v}$. This combined value represents how well the chip i match with the underlying query. We then return the top clips (e.g., Top-1) as predictions.

4 SpikingVTG: Architecture Overview

The core computational unit of the proposed SpikingVTG model is a leaky integrate-and-fire (LIF) neuron [29]. Neurons communicate with each other using sparse, spike-based activations instead of real-valued signals, thus we can replace floating-point matrix multiplications with accumulative operations, resulting in improved computational efficiency.

4.1 Spiking Neural Networks

The discrete time dynamics of an LIF-based spiking neuron can be given as follows,

$$\begin{aligned} u_i[t+\delta] &= \gamma u_i[t] + W_{(i-1)}(s_{(i-1)}[t]) + b_i, \\ u_i[t+1] &= u_i[t+\delta] - V_{th_i} s_i[t+1], \\ a_i[t] &= \frac{\sum_{\tau=1}^{t} \gamma^{t-\tau} s_i[\tau]}{\sum_{\tau=1}^{t} \gamma^{t-\tau}}. \end{aligned} \tag{1}$$

where, at time t, $u_i[t]$ is the membrane potential of the i^{th} neuronal layer; b_i indicates a bias term and γ is the leaky term. $W_{(i-1)}$ represents the layer-specific operation; $t + \delta$ is an intermediate time step to determine if the neuron fired; V_{th_i} is the threshold of layer i. We use a ternary spiking model [30] in our work for spike ($s[t + 1]$) generation. This improves performance while avoiding the introduction of additional floating-point multiplicative and accumulative (FP-MAC) operations. The average spiking rate (ASR $a_i[t]$) of neurons within each layer i at time t can be defined as a weighted-average function.

4.2 Spiking Transformer Core

The high-level overview of each encoder block of our spiking transformer architecture is demonstrated in Fig. 1. The model consists of N encoder layers, each consists of a spiking multi-headed attention block, followed by an intermediate layer and an output layer. Communication within and between encoder layers occurs via spikes. Furthermore, all matrix multiplications involved in linear layers and attention layer comprises of more efficient fp-accumulative (FP-ACC) operations instead of FP-MAC operations in conventional neural architectures. Detailed descriptions of each layer are provided in the Appendix A. Following architectural optimizations (Section 4.6), we replace softmax-based attention with a ReLU and scaling-based spiking attention mechanism, remove all layer normalization operations, and explore extreme quantization of linear weights.

4.3 Saliency Feedback Gating (SFG)

SpikingVTG operates over a specific number of convergence time steps (T_c), with the convergence dynamics detailed in Section 4.5. This temporal processing allows us to leverage intermediate temporal outputs to dynamically update the input to the model at every time step for improved performance. This approach conforms to the feedback connections observed in the human visual

cortex [9], providing a bio-plausible explanation for its efficacy. The ASR of the final encoder layer of the Spiking Transformer core is used as a temporal feedback to compute a dynamic saliency score with the input query enabling the design of a gating mechanism (Fig. 2a), allowing selective focusing on relevant segments of the video while minimizing computation on irrelevant segments. The saliency feedback gating mechanism is shown below,

$$
\begin{aligned}
F_s^{v_i}[t] = \cos(\mathbf{a}^{\mathbf{i}}_{\mathbf{N_v}}[\mathbf{t}], \mathbf{M}) := \frac{\mathbf{a}^{\mathbf{i}}_{\mathbf{N_v}}[\mathbf{t}] \cdot \mathbf{M}}{\|\mathbf{a}^{\mathbf{i}}_{\mathbf{N_v}}[\mathbf{t}]\|_2 \|\mathbf{M}\|_2}, \\
\bar{V}[t+1] = V[t] * \bar{F}_s^v[t], \\
\mathbf{SFG}(V, \mathbf{Q}, \mathbf{a}_{\mathbf{N_v}}) = \bar{V}[t+1] \oplus \mathbf{Q},
\end{aligned}
\tag{2}
$$

where, using attentive pooling operation, sentence representation $\mathbf{M} = \mathbf{Q}^T Softmax(\mathbf{Q}\mathbf{W_p})$, $\mathbf{M} \in \mathbb{R}^D$, input textual query features $\mathbf{Q} \in \mathbb{R}^{L_q \times D}$, input video features $V \in \mathbb{R}^{L_v \times D}$ and $\mathbf{W_p} \in \mathbb{R}^{D \times 1}$ is a learnable embedding and D is the hidden dimension. $F_s^{v_i}[t]$ is the dynamic saliency score, at time t, for the i-th segment of the video. We compute $\bar{F}_s^v$ by applying min-max normalization to F_s^v, allowing us to obtain per-clip scores within the range $[0, 1]$. The ASR of the output of the spiking transformer core is given as $a_N[t] \in \mathbb{R}^{(L_v+L_q)\times D}$. $a^i_{N_v}[t]$ is ASR of output of the spiking transformer core, corresponding to video segment i, at time t. The output of SFG module is the concatenation of saliency feedback gated video features and query features and serves as the input to the spiking transformer core at time $t+1$.

To ensure compatibility between the two modalities during cosine similarity computation, both representations are projected into a shared latent space of dimensionality D. The query representation is aggregated into a global sentence embedding via attention-based pooling with a learnable embedding vector. This adaptive mapping aligns the textual representation with the spatiotemporal semantics of the spiking video features, thereby facilitating effective cross-modal compatibility.

SFG Computational Overhead: The SFG layer comprises $O(L_v \cdot D)$ floating-point multiplication operations; however, the computational overhead of this layer is significantly less than that of the transformer core which has a complexity of $O(L^2 \cdot D + L \cdot D^2)$, where $L = L_v + L_q$.

4.3.1 Visualizing effect of SFG:

The dynamic saliency score (F_s^v) achieves an equilibrium value F_s^* following the convergence of ASR of the neuronal layers of the SpikingVTG model (Fig. 3). As shown in Fig. 2b, we empirically analyze the scores per clip at equilibrium to gain insights into the functioning of the SFG based multiplicative gating mechanism. Neighboring salient clips of the target clip exhibit higher $\bar{F}_s$ scores at equilibrium, while irrelevant clips show lower scores, highlighting the effectiveness of the SFG mechanism.

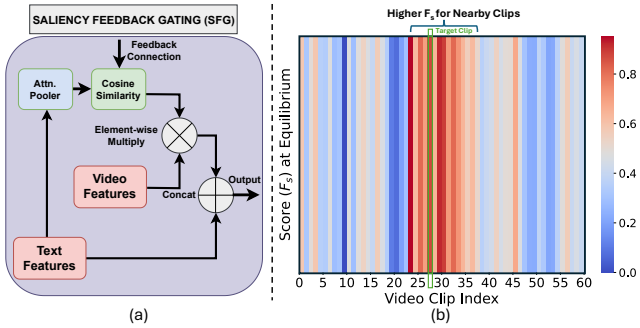

Figure 2: (a) Overview of the internal operations of the saliency-feedback gating mechanism. The ASR of the output of the spiking transformer core at each time step is leveraged as the feedback signal (Fig. 1). (b) Heatmap showing the scores per clip ($\bar{F}_s$) at equilibrium, with the target frame for highlight detection corresponding to clip index 28.

The SFG mechanism not only results in **better performance** of our SpikingVTG architecture (Table 5) but also **reduces overall neural activity** by sparsifying input spikes. Empirical results (Fig. 3b) confirm that the model with the gating mechanism exhibits a lower neural activity, particularly in the input and spiking attention layers, compared to the model without SFG.

4.4 Spiking Decoder Module

The spiking decoder comprises of stacked 1-D convolutions followed by integrate-fire (IF) neuron layers ($\gamma = 1$ in Eqn. 1), for spike generation. The spiking decoder used for predicting foreground indicator (f_i) per clip, applies n_1 1-D convolution operations with kernel size k_1, each followed by an

IF layer. The final layer consists of a single output channel, and its temporal mean is passed through a sigmoid activation to produce the prediction. The spiking decoder used for d_i applies n_2 1-D convolution operations with kernel size k_2, each followed by an IF layer, and the final convolution layer has two output channels to predict $d_i = [d_{s_i}, d_{e_i}]$, after which we compute b_i.

4.5 SpikingVTG: Convergence Dynamics

The membrane potential (u_1) of the input LIF layer to the spiking transformer core, following the SFG mechanism can be formulated as below,

$$u_1[t+1] = \gamma u_1[t] + \textbf{SFG}(V, Q, a_{N_v}[t]) + b_1 - V_{th_1} s_1[t+1] \tag{3}$$

where, $a^{N_v}[t]$ is the ASR of the final layer corresponding to the video features at time t, V is the input video features, Q is the query features and **SFG** is defined in Eqn. 2. The layer-wise convergence dynamics of the SpikingVTG with SFG is demonstrated in Fig. 3.

Following Eqn. 1, the layer-wise ASR is, $a_i[t+1] = \frac{1}{V_{th_i}}(\hat{f}(a_{(i-1)}[t+1]) + b_i - \frac{u_i[t+1]}{\sum_{j=0}^{t}\gamma^j})$ where, $\hat{f}$ is operation of layer i. Following empirical evidence (Fig. 3) and theoretical formulation [10, 26] as time $t \to \infty$, the layer-wise ASRs converge to equilibrium, enabling the derivation of layer-wise steady-state equations given as,

$$a_i^* = \sigma(\frac{1}{V_{th_i}}(\hat{f}(a_{i-1}^*) + b_i)) \tag{4}$$

The equation describing the steady-state ASR dynamics of the input layer is given as,

$$a_1^* = \sigma(\frac{1}{V_{th_1}}(\textbf{SFG}(V, Q, a_{N_v}^*) + b_1)) \tag{5}$$

where, clipping function $\sigma(x)$ clamps the values within $[-1, 1]$. This is because we allow ternary spikes thus ASR must be with $[-1, 1]$. Furthermore, the dynamic saliency score also achieves an equilibrium value F_s^* since the ASR at the final layer achieves equilirbrium state $a_{N_v}^*$. Convergence dynamics of other layers are given in Appendix A. To analyze the overall layer-wise neural activity, which includes both positive and negative spiking event, we present the layer-wise dynamics of the absolute spiking activity events in Fig. 3b, i.e. $act_i[t] = \frac{\sum_{i=1}^{t}|s_i[t]|}{t}$.

Training: As described in the Section 4.4, the Spiking decoder is responsible for predicting $\tilde{f}_i$ and $\tilde{d}_i$ for individual video clip i and $\tilde{s}_i$ is computed using the SFG module at equilibrium. Using these three predictions, we design a loss function that combines various components. The total loss over N_T clips in the training set is given by $L = \frac{1}{N_T}\sum_{i=1}^{N_T}(L_{f_i} + L_{d_i} + L_{c_i})$, where L_f is the binary cross-entropy loss for the indicator variable f_i, L_d combines smooth L1 loss with IoU loss [31] for the predicted boundaries, and L_c is an optional loss incorporating intra- and inter-video contrastive learning [32]. Detailed formulation of the loss functions is in Appendix B.

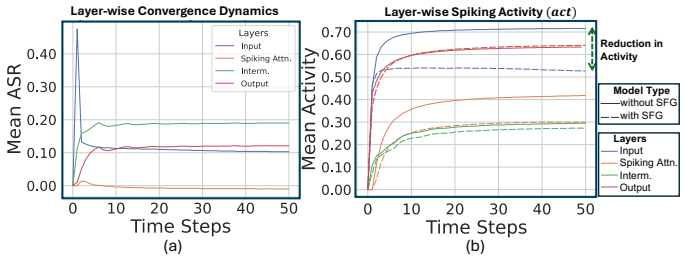

Figure 3: Results from a random QVHighlights input passed through SpikingVTG models. (a) Layer-wise mean ASR convergence over operating time steps for a sample spiking transformer encoder layer (Fig. 1); note ASR can be negative due to ternary spikes. (b) Layer-wise mean spiking activity ($act_i[t]$, averaged over neurons) versus time steps. Model with SFG exhibit reduced activity in both input and spiking attention layers, highlighting SFG's role in minimizing neural activity.

During training, leveraging implicit differentiation [33] at equilibrium, only ASR values at equilibrium are used, $\frac{\partial L(a^*)}{\partial \theta} = -\frac{\partial L(a^*)}{\partial a^*}(J_{g_\theta}^{-1}|_{a^*})\frac{\partial f_\theta(a^*)}{\partial \theta}$ where, θ is the model parameters, $g_\theta(a) = f_\theta(a) - a$,

f is the steady-state equation of ASR, J^{-1} is the inverse Jacobian of g_θ when $a = a^*$, i.e., at equilibrium. Thus, unlike BPTT, we do not need to store the intermediate computational graph and the model parameters can be updated using a single backpropagation step.

4.5.1 Cos-L2 Representation Matching (CLRM)

We train the base SpikingVTG model on VTG tasks such as QVHighlights [1], achieving comparable performance to a similarly scaled non-spiking transformer-based model, UniVTG [6], as demonstrated in Table 1. The later to improve its performance further uses pre-training on large datasets such as Video-CC and Ego4D. Since, extensive pretraining of the SpikingVTG is considerably resource intensive we propose an effective knowledge transfer strategy. This mechanism enables SpikingVTG to inherit the generalization capabilities learned by the pre-trained detection transformer models like UniVTG model. Importantly, since the primary benefit of spiking architectures manifests during inference on resource-constrained edge devices, this knowledge transfer is a one-time process, which significantly improves performance.

Table 1: **Ablation study of the effect of CLRM** on SpikingVTG evaluated on the evaluation set of QVHighlights.

Method	QVHighlights-MR			QVHighlights-HD	
	@0.5	@0.7	mAP@avg	mAP	HIT@1
UniVTG [6]	59.74	40.90	36.13	38.83	61.81
SpikingVTG w/o CLRM	60.12	39.68	36.23	38.84	62.49
UniVTG with PT [6]	67.35	52.65	45.44	41.34	68.77
SpikingVTG w/ CLRM	**67.58**	**50.82**	**44.07**	**40.81**	**68.64**

Hidden State Matching Loss: To align the hidden state (output of individual encoder layer) of the non-spiking multi-modal transformer model with the converged ASR (4.5) of the corresponding layer of the SpikingVTG model, we propose a hybrid loss function combining a squared cosine similarity based directional loss and a L2-norm based loss for minimizing the scale difference. For each layer, the student representation is projected into the feature space of the teacher via a learnable linear transformation $W_d \in \mathbb{R}^{d_s \times d_t}$. The total loss is:

$$\mathcal{L}_{\text{rep}} = \frac{1}{B \times L} \sum_{i=1}^{N} \sum_{j=1}^{B} \sum_{k=1}^{L} \left[\lambda_{\cos} \cdot \left(1 - \cos\left(\theta_{i,j,k}^{\text{rep}}\right)\right)^2 + \lambda_{\ell_2} \cdot \left\| \mathbf{s}_i^{(j,k)} - \mathbf{t}_i^{(j,k)} \right\|_2^2 \right] \tag{6}$$

where N is total number of encoder layers, B is batch size, L is length of sequence, $\lambda_{\cos}$, λ_{l_2} are hyperparameters and the cosine similarity is computed as: $\cos(\theta_{i,j,k}^{\text{rep}}) = \frac{\mathbf{s}_i^{(j,k)} \cdot \mathbf{t}_i^{(j,k)}}{\|\mathbf{s}_i^{(j,k)}\|_2 \cdot \|\mathbf{t}_i^{(j,k)}\|_2}$, where, $\mathbf{s}_i^{(j,k)} = a_{r_i}^{*(j,k)} W_d \in \mathbb{R}^{d_t}$ is the projected student representation and $\mathbf{t}_i^{(j,k)} = T_{r_i}^{(j,k)} \in \mathbb{R}^{d_t}$ is the teacher representation at layer i, sequence position k, and batch index j. $a_{r_i}^* \in \mathbb{R}^{B \times L \times d_s}$ and $T_{r_i} \in \mathbb{R}^{B \times L \times d_t}$ are the pre-activation outputs from student and teacher, respectively.

Attention Score Matching Loss: We align the attention score map ($\mathbf{A_i^T} \in \mathbb{R}^{B \times L \times L}$) of encoder layer i of the non-spiking model with the mean attention score ($\mathbf{A_i^{S^*}} = \frac{1}{T_c} \sum_{t=1}^{T_c} A_i^S[t] \in \mathbb{R}^{B \times L \times L}$) of our corresponding converged SpikingVTG model. The total attention map alignment loss ($\mathcal{L}_{\text{att}}$) is then computed following Eqn. 6. This loss term encourages the student to learn cross-modal attention behavior consistent with the pre-trained teacher.

The cumulative representation matching loss is given as $\mathcal{L}_{\text{CLRM}} = \mathcal{L}_{\text{rep}} + \mathcal{L}_{\text{att}}$. Optimizing this loss enables our SpikingVTG model to learn the generalability enabling better performance as demonstrated in Table 1.

4.6 Normalization-Free and Quantized SpikingVTG

We perform two key optimizations: removal of non-local operations and 1-bit weight quantization. Although SpikingVTG replaces floating-point MACs with ternary spikes, it retains softmax and layer normalization, which are inefficient for resource-constrained hardware. We eliminate these by introducing a Normalization-Free (NF) SpikingVTG.

In NF-SpikingVTG, softmax in the spiking attention is replaced with a ReLU followed by scaling with $1/L$, reducing compute overhead. Given d-dimensional queries, keys, and values $\{q_i[t], s_{k_i}[t], s_{v_i}[t]\}_{i=1}^{L}$, at time t, the attention weights α_{ij} are computed as follows:

$$\alpha_{ij}[t] = \phi\left(\left[q_i[t]^\top s_{k_1}[t], \cdots, q_i[t]^\top s_{k_L}[t]\right]\right)_j \tag{7}$$

where, ϕ is a custom kernel consisting of ReLU operation followed by scaling with L^{-1}. We further remove all layer normalization layers to eliminate additional non-local operations.

To reduce memory and computational cost, we introduce 1-bit NF-SpikingVTG by binarizing weights. Each weight matrix $W \in \mathbb{R}^{n\times m}$ is zero-centered and quantized as $W_q = \text{sgn}\left(W - \frac{1}{nm}\sum_{i,j} W_{ij}\right), \quad \beta = \frac{1}{nm}\sum_{i,j}|W_{ij}|$, where $W_q \in \{-1, +1\}$. The output of each quantized linear layer is scaled by β, yielding a binary-weight, ternary-activation model that supports efficient integer-ACC operations. 1-bit NF-SpikingVTG achieves competitive performance while drastically reducing memory and compute, making it well-suited for edge deployment.

Table 2: Performance comparison of our SpikingVTG model against non-spiking VTG solutions on **test sets** of QVHighlights-MR and Charades-STA datasets for moment retrieval task.

Method	SNN	QVHighlights-MR				Charades-STA			
		@0.5	@0.7	mAP@0.5	mAP@0.75	@0.3	@0.5	@0.7	mIoU
M-DETR [1]	No	52.89	33.02	54.82	29.40	65.83	52.07	30.59	45.54
UMT [5]	No	56.23	41.18	53.83	37.01	-	49.35	26.16	-
UniVTG [6]	No	58.86	40.86	57.60	35.59	70.81	58.01	35.65	50.10
UniVTG+PT [6]	No	65.43	50.06	64.06	45.02	72.63	60.19	38.55	52.17
UVCOM [34]	No	63.55	47.47	63.37	42.67	-	56.69	34.76	-
SpikeMba [19]	No	64.13	49.42	-	43.67	71.24	59.65	36.12	51.74
BAM-DETR [35]	No	62.71	48.64	64.57	46.33	72.93	59.95	39.38	52.33
TR-DETR [36]	No	64.66	48.96	63.98	43.73	-	57.61	33.52	-
CG-DETR [37]	No	65.43	48.38	64.51	42.77	70.40	58.40	36.30	50.10
LLMEPET [38]	No	66.73	49.94	65.76	43.91	70.91	-	36.49	50.25
SpikingVTG	**Yes**	**65.29**	**48.18**	**64.31**	**42.25**	**71.20**	**58.73**	**37.16**	**50.62**

5 Experimentation

We evaluate SpikingVTG variants on moment retrieval and highlight detection tasks using the QVHighlights, Charades-STA, TACoS and Youtube Highlight datasets. Since, to the best of our knowledge, our proposed model is the first spiking detection transformer evaluated on VTG tasks, we compare its performance against sota non-spiking detection transformers. Additional training, dataset, evaluation metric, hyperparameter and experimental details are provided in Appendix C & E. The experiments were run on a NVIDIA RTX A6000 GPU with 48GB memory.

Table 3: Performance comparison on the **test set** of TACoS for **moment retrieval task**.

Method	SNN	TACoS			
		@0.3	@0.5	@0.7	mIoU
M-DETR [1]	No	37.97	24.67	11.97	25.49
2D-TAN [6]	No	40.01	27.99	12.92	27.22
UniVTG [6]	No	51.44	34.97	17.35	33.60
QD-DETR [39]	No	52.39	36.77	21.07	35.76
CG-DETR [37]	No	52.23	-	22.23	36.48
UniVTG+PT [6]	No	56.11	43.44	24.27	38.63
SpikeMba [19]	No	51.98	39.34	22.83	35.81
SpikingVTG	**Yes**	**54.71**	**39.27**	**21.84**	**36.02**

5.1 Results

SpikingVTG establishes a baseline for spiking models on VTG tasks. The results are shown in Table 2, 3 & 4. Our model achieves competitive results compared to the current SOTA non-spiking models.

Training & Inference Metrics: In the CLRM-based knowledge transfer stage, the memory requirement is 20GB when using a batch size of 32 on the QVHighlights dataset. In contrast, replacing our training method with BPTT would require over 100GB of memory for $T_c > 10$, making BPTT computationally infeasible. Training on true labels with similar batch size requires 8GB of

Table 4: Performance comparison of our SpikingVTG model against other non-spiking VTG solutions on the **test set** of the QVHighlights and Youtube Highlights for **highlight detection task**.

Method	SNN	QVHighlights-HD		Youtube Highlights						
		mAP	HIT@1	Dog	Gym.	Skating	Skiing	Parking	Surfing	Avg.
UMT [5]	No	38.18	59.99	65.9	75.2	71.8	72.3	81.6	82.7	74.9
UniVTG [6]	No	38.20	60.96	71.8	76.5	73.3	73.2	73.9	82.2	75.2
UniVTG+PT [6]	No	40.54	66.28	74.3	79.0	84.9	75.1	74.4	83.9	78.6
QD-DETR [39]	No	38.90	62.40	72.2	77.4	72.7	72.8	71.0	80.6	74.4
SpikeMba [5]	No	-	-	74.4	75.4	74.3	75.5	-	-	75.5
CG-DETR [37]	No	40.30	66.20	76.3	76.1	76.0	75.1	70.0	81.9	75.9
UVCOM [34]	No	39.98	65.58	73.8	77.1	76.0	75.1	75.7	82.7	76.4
LLMEPET [38]	No	38.18	59.99	73.6	73.2	75.3	74.0	72.5	82.5	75.3
SpikingVTG	**Yes**	**40.46**	**65.82**	**73.9**	**78.1**	**80.1**	**74.2**	**72.2**	**81.7**	**76.7**

memory. The clock time for 50 epochs of training on QVHighlights is $\approx$ 5 HOURS. Inference latency on entire test set of QVHighlights ranges from 25 sec ($T_c = 2$) to $\approx$ 4 mins ($T_c = 20$).

Table 5: **Ablation Study** of SpikingVTG variants on the evaluation set of QVHighlights.

Method	QVHighlights-MR		QVHighlights-HD		Operations	Local	Activity	Energy
	@0.5	@0.7	mAP	HIT@1				
Pre-trained UniVTG (sota)	67.35	52.65	41.34	68.77	FP-MAC	×	1.0	23.92mJ
SpikingVTG **without SFG**	64.94	47.21	40.49	67.37	FP-ACC	×	0.41	15.2mJ
SpikingVTG **with SFG**	**67.58**	**50.82**	**40.81**	**68.64**	FP-ACC	×	0.34	13.8mJ
(NF)-SpikingVTG w/ SFG	66.59	48.31	40.61	67.73	FP-ACC	✓	0.25	10.1mJ
1-bit (NF)-SpikingVTG w/ SFG	65.31	47.48	40.35	67.30	**INT-ACC**	✓	**0.19**	**1.3mJ**
1-bit (NF)-SpikingVTG w/ ReLU	65.91	47.04	40.16	67.07	**INT-ACC**	✓	**0.19**	**1.3mJ**

Ablation Study: As demonstrated in Table 5, the inclusion of the SFG mechanism enhances performance compared to the model without SFG. It results in reduced neuronal activity and overall less energy consumption. Thus, we enable SFG for the SpikingVTG variants we discuss next. Without non-local operations, the (NF)-SpikingVTG model achieves competitive performance even compared to other SOTA non-spiking VTG models. Although the 1-bit (NF)-SpikingVTG variant shows a slight reduction in performance, it is highly memory efficient and involves simpler INT-ACC computations, resulting in order of magnitude less energy consumption. Along with energy consumption ($T_c = 10$) Table 5, also highlights the **sparsity** in the SpikingVTG variants particularly in the 1-bit version underscoring considerably **reduced mean neural activity**. For a more hardware friendly model, we train a variant of 1-bit (NF)-SpikingVTG by replacing all GELU layers with ReLU layer [40] and observe minimal degradation in performance. Extensive hyper-parameter study is done in Appendix E.

Analysis of Energy Efficiency: We analyze the test-time energy efficiency of SpikingVTG variants compared to a non-spiking transformer-based model with same depth and hidden dimensions. The analysis considers arithmetic operation costs using a 45nm CMOS technology with 32-bit precision, where FP-MAC, FP-ACC, and INT-ACC ops. consume 4.6 pJ, 0.9 pJ, and 0.1 pJ respectively [41]. The energy cost for a non-spiking transformer encoder layer based on total FLOPs used in attention and linear layers, is $E_A = [(3LD^2)+(LD^2+L^2D)+(2LD^2)]\times 4.6\,\text{pJ}$. Considering $L = 200$, $D = 1024$, we compute $E_A = 5.98\,\text{mJ}$. Since, UniVTG has 4 encoder layers, so total energy is $23.92mJ$.

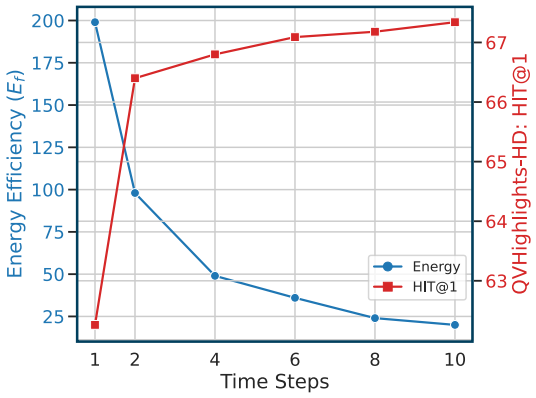

Figure 4: Graph showing tradeoff of energy efficiency (E_f) and HIT@1 on QVHighlights-HD for varying time steps.

In contrast, the 1-bit (NF)-SpikingVTG operates over T_c time steps. Its per time-step cost is: $E_{S_t} = [(3 \cdot IFR_{\text{in}} \cdot LD^2) + (IFR_{\text{k}} \cdot LD^2 + IFR_{\text{v}} \cdot L^2D) + (IFR_{\text{attn}} \cdot LD^2) + (IFR_{\text{int.}} \cdot LD^2)] \times 0.1\,\text{pJ}$, where IFR_l denotes the mean activity of component l. For our 1-bit (NF)-SpikingVTG model, which uses INT-ACC operations, we empirically measure: $IFR_{\text{in}} = 0.40$, $IFR_{\text{k}} = 0.18$, $IFR_{\text{v}} = 0.19$, $IFR_{\text{attn}} = 0.03$, and $IFR_{\text{interm.}} = 0.09$, resulting in $E_{S_t} = 0.03\,\text{mJ}$. With $T_c = 10$, the total spiking energy is $E_S = T_c \cdot E_{S_t} = 0.3\,\text{mJ}$, yielding an energy efficiency of $E_f = E_A/E_S = 19.93$. Model-wise energy comparison is shown in Table 5.

Energy-Accuracy Tradeoff: Unlike conventional non-spiking VTG solutions, SpikingVTG enables a controllable trade-off between performance and energy consumption, as total energy usage scales with the number of operating time steps. In Fig. 4, we demonstrate this performance-energy tradeoff achieved using 1-bit (NF)-SpikingVTG model by running it for different number of time steps. Notably, when running the model for just 2 time steps, SpikingVTG attains a HIT@1 score of 66.4 (compared to the SOTA score of 68.77), while achieving an energy efficiency factor $E_f \approx 100$.

6 Conclusions

In this paper we propose SpikingVTG, a bio-inspired computationally efficient solution for VTG tasks in resource constraint environment. We propose a saliency feedback gating mechanism, which leverages the temporal dynamics of the model to improve performance while lowering computational costs by reducing overall neuronal activity across the model. We empirically demonstrate the convergence dynamics of the SpikingVTG model and leverage the formulated steady-state equations to efficiently train our model using implicit differentiation at equilibrium. To improve generalization, we propose a Cos-L2 Representation Matching loss, enabling knowledge transfer from non-spiking VTG models and yielding substantial performance gains. We further optimize the model by removing non-local operations and applying extreme quantization, leading to the 1-bit NF-SpikingVTG. Our framework significantly improves computational efficiency and model compactness, enabling tunable energy-accuracy tradeoffs on low-resource devices. While this work presents the first spiking solution for VTG with competitive performance, there remains room for improvement in fully closing the performance gap. Future research can build upon our framework to further advance spiking models in this domain.

7 Acknowledgments

This work was supported in part by the United States Air Force and Defense Advanced Research Projects Agency (DARPA) under Contract No. FA8750-23-C-0519 and the U.S. Army Research Laboratory Cooperative Research Agreement W911NF-17-2-0196. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the Department of Defense or the United States Government.

References

[1] Jie Lei, Tamara L Berg, and Mohit Bansal. Detecting moments and highlights in videos via natural language queries. *Advances in Neural Information Processing Systems*, 34:11846–11858, 2021.

[2] Songyang Zhang, Houwen Peng, Jianlong Fu, and Jiebo Luo. Learning 2d temporal adjacent networks for moment localization with natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 12870–12877, 2020.

[3] Jonghwan Mun, Minsu Cho, and Bohyung Han. Local-global video-text interactions for temporal grounding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10810–10819, 2020.

[4] Fa-Ting Hong, Xuanteng Huang, Wei-Hong Li, and Wei-Shi Zheng. Mini-net: Multiple instance ranking network for video highlight detection. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIII 16*, pages 345–360. Springer, 2020.

[5] Ye Liu, Siyuan Li, Yang Wu, Chang-Wen Chen, Ying Shan, and Xiaohu Qie. Umt: Unified multi-modal transformers for joint video moment retrieval and highlight detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3042–3051, 2022.

[6] Kevin Qinghong Lin, Pengchuan Zhang, Joya Chen, Shraman Pramanick, Difei Gao, Alex Jinpeng Wang, Rui Yan, and Mike Zheng Shou. Univtg: Towards unified video-language temporal grounding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2794–2804, 2023.

[7] Siddharth Samsi, Dan Zhao, Joseph McDonald, Baolin Li, Adam Michaleas, Michael Jones, William Bergeron, Jeremy Kepner, Devesh Tiwari, and Vijay Gadepally. From words to watts: Benchmarking the energy costs of large language model inference. In *2023 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–9. IEEE, 2023.

[8] Samanwoy Ghosh-Dastidar and Hojjat Adeli. Spiking neural networks. *International journal of neural systems*, 19(04):295–308, 2009.

[9] Kohitij Kar, Jonas Kubilius, Kailyn Schmidt, Elias B Issa, and James J DiCarlo. Evidence that recurrent circuits are critical to the ventral stream's execution of core object recognition behavior. *Nature neuroscience*, 22(6):974–983, 2019.

[10] Mingqing Xiao, Qingyan Meng, Zongpeng Zhang, Yisen Wang, and Zhouchen Lin. Training feedback spiking neural networks by implicit differentiation on the equilibrium state. *Advances in Neural Information Processing Systems*, 34:14516–14528, 2021.

[11] Emre O Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, 36(6):51–63, 2019.

[12] Arsha Nagrani, Paul Hongsuck Seo, Bryan Seybold, Anja Hauth, Santiago Manen, Chen Sun, and Cordelia Schmid. Learning audio-video modalities from image captions. In *European Conference on Computer Vision*, pages 407–426. Springer, 2022.

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[14] Amar Shrestha, Haowen Fang, Zaidao Mei, Daniel Patrick Rider, Qing Wu, and Qinru Qiu. A survey on neuromorphic computing: Models and hardware. *IEEE Circuits and Systems Magazine*, 22(2):6–35, 2022.

[15] Soroush Abbasi Koohpayegani and Hamed Pirsiavash. Sima: Simple softmax-free attention for vision transformers. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2607–2617, 2024.

[16] Boxun Xu, Hejia Geng, Yuxuan Yin, and Peng Li. Ds2ta: Denoising spiking transformer with attenuated spatiotemporal attention. *arXiv preprint arXiv:2409.15375*, 2024.

[17] Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Huaijie Wang, Lingxiao Ma, Fan Yang, Ruiping Wang, Yi Wu, and Furu Wei. Bitnet: Scaling 1-bit transformers for large language models. *arXiv preprint arXiv:2310.11453*, 2023.

[18] Malyaban Bal, Yi Jiang, and Abhronil Sengupta. Exploring extreme quantization in spiking language models. In *2024 International Conference on Neuromorphic Systems (ICONS)*, pages 272–276. IEEE, 2024.

[19] Wenrui Li, Xiaopeng Hong, and Xiaopeng Fan. Spikemba: Multi-modal spiking saliency mamba for temporal video grounding. *arXiv preprint arXiv:2404.01174*, 2024.

[20] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[21] Michael V DeBole, Brian Taba, Arnon Amir, Filipp Akopyan, Alexander Andreopoulos, William P Risk, Jeff Kusnitz, Carlos Ortega Otero, Tapan K Nayak, Rathinakumar Appuswamy, et al. Truenorth: Accelerating from zero to 64 million neurons in 10 years. *Computer*, 52(5):20–29, 2019.

[22] Mike Davies, Andreas Wild, Garrick Orchard, Yulia Sandamirskaya, Gabriel A Fonseca Guerra, Prasad Joshi, Philipp Plank, and Sumedh R Risbud. Advancing neuromorphic computing with loihi: A survey of results and outlook. *Proceedings of the IEEE*, 109(5):911–934, 2021.

[23] Guangzhi Tang, Neelesh Kumar, and Konstantinos P. Michmizos. Reinforcement co-learning of deep and spiking neural networks for energy-efficient mapless navigation with neuromorphic hardware. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6090–6097, 2020.

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

[25] 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.

[26] 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.

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

[28] Jan Hosang, Rodrigo Benenson, and Bernt Schiele. Learning non-maximum suppression. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4507–4515, 2017.

[29] Sangya Dutta, Vinay Kumar, Aditya Shukla, Nihar R Mohapatra, and Udayan Ganguly. Leaky integrate and fire neuron by charge-discharge dynamics in floating-body mosfet. *Scientific reports*, 7(1):8257, 2017.

[30] Yufei Guo, Yuanpei Chen, Xiaode Liu, Weihang Peng, Yuhan Zhang, Xuhui Huang, and Zhe Ma. Ternary spike: Learning ternary spikes for spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 12244–12252, 2024.

[31] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 658–666, 2019.

[32] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[33] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Deep equilibrium models. *Advances in Neural Information Processing Systems*, 32, 2019.

[34] Yicheng Xiao, Zhuoyan Luo, Yong Liu, Yue Ma, Hengwei Bian, Yatai Ji, Yujiu Yang, and Xiu Li. Bridging the gap: A unified video comprehension framework for moment retrieval and highlight detection, 2023.

[35] Pilhyeon Lee and Hyeran Byun. Bam-detr: Boundary-aligned moment detection transformer for temporal sentence grounding in videos, 2024.

[36] Hao Sun, Mingyao Zhou, Wenjing Chen, and Wei Xie. Tr-detr: Task-reciprocal transformer for joint moment retrieval and highlight detection, 2024.

[37] WonJun Moon, Sangeek Hyun, SuBeen Lee, and Jae-Pil Heo. Correlation-guided query-dependency calibration for video temporal grounding, 2024.

[38] Yiyang Jiang, Wengyu Zhang, Xulu Zhang, Xiao-Yong Wei, Chang Wen Chen, and Qing Li. Prior knowledge integration via llm encoding and pseudo event regulation for video moment retrieval. In *Proceedings of the 32nd ACM International Conference on Multimedia*, MM '24, page 7249–7258. ACM, October 2024.

[39] WonJun Moon, Sangeek Hyun, SangUk Park, Dongchan Park, and Jae-Pil Heo. Query-dependent video representation for moment retrieval and highlight detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23023–23033, 2023.

[40] Jonathan Timcheck, Sumit Bam Shrestha, Daniel Ben Dayan Rubin, Adam Kupryjanow, Garrick Orchard, Lukasz Pindor, Timothy Shea, and Mike Davies. The intel neuromorphic dns challenge. *Neuromorphic Computing and Engineering*, 3(3):034005, 2023.

[41] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks, 2015.

[42] Jiyang Gao, Chen Sun, Zhenheng Yang, and Ram Nevatia. Tall: Temporal activity localization via language query. In *Proceedings of the IEEE international conference on computer vision*, pages 5267–5275, 2017.

[43] Michaela Regneri, Marcus Rohrbach, Dominikus Wetzel, Stefan Thater, Bernt Schiele, and Manfred Pinkal. Grounding action descriptions in videos. *Transactions of the Association for Computational Linguistics*, 1:25–36, 2013.

[44] Min Sun, Ali Farhadi, and Steve Seitz. Ranking domain-specific highlights by analyzing edited videos. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*, pages 787–802. Springer, 2014.

A Extended Architecture Overview

The Spiking Transformer layer primarily consists of a spiking multi-head attention (MHA) block, followed by a spiking feedforward network comprising an intermediate layer and an output layer with both inter- and intra-layer communication happening using spikes. Details of the operations in each layer are provided below.

Spiking Attention Block: In Spiking MHA, to enable computationally efficient accumulate based operations the input to the attention layer are spikes instead of real-valued data. The spiking attention mechanism [26] is given as follows,

$$Attn(X_s[t], K_s[t], V_s[t]) = \phi(d * Q(X_s[t]) \cdot (K_s[t])^T) \cdot V_s(t) \tag{8}$$

Here, $Q(X_s(t))$ represents the Query, obtained by passing the input spikes $X_s(t)$ at time t through a linear layer (W_Q). The spikes for the Key layer ($K_s(t)$) are generated by passing $X_s(t)$ through a linear mapping (W_K), followed by an LIF neuron layer. Similarly, we generate spikes for Value. d is a scaling constant. Since the input, key, and value matrices consist of spike trains rather than real-valued data, the primary computations in all matrix multiplications are floating-point accumulation operations rather than floating point multiplicative and accumulative operations. In the (NF)-SpikingVTG variant, as discussed in the paper, we use ϕ as the $ReLU$ and scaling operation, significantly reducing the computational overhead compared to employing ϕ as the non-local $Softmax$ operation. The output of the attention layer is fed to an LIF neuron, which outputs spikes. The convergence dynamics of the layer at equilibrium is given as, $a^*_{attn} = \sigma(\frac{1}{V_{th}}(Attn(a^*_x, a^*_k, a^*_v) + b_{attn})$, where a_x represents the ASR of the layer used to generate the Query, a_k denotes the ASR of the Key, and a^*_v corresponds to the ASR of the Value. b_{attn} is a bias term.

Intermediate Layer: The intermediate layer takes as input the spikes generated from the preceding layer and maps it to an intermediate dimension with a linear layer. The output is then passed through an LIF layer. The convergence dynamics of the layer at equilibrium is given as, $a^*_{interm.} = \sigma(\frac{1}{V_{th}}(act(W_{interm.}a^*_p) + b_{interm.}))$, where $W_{interm.}$ is the linear weight and gelu() is the activation used for the layer. a^*_p is the ASR at equilibrium for the previous layer. $b_{interm.}$ is a bias term. In this paper, we have used explored different choices for act, such as $GELU$ and $ReLU$. During inference, all matrix multiplications involve accumulative operations due to the nature of the input.

Output Layer: The output layer takes as input the spikes generated from the preceding layers as shown in Fig. 1. The output is then passed through an LIF layer. The convergence dynamics of the layer at equilibrium is given as, $a^*_{output} = \sigma(\frac{1}{V_{th}}(norm(W_{output}a^*_{interm.} + a^*_p) + b_{output}))$, where W_{output} is the linear weight and layer norm is used for normalization. $a^*_{interm.}$ is the ASR at equilibrium for the previous intermediate layer. b_{output} is a bias term. During inference, all matrix multiplications involve accumulative operations due to the nature of the input. In the (NF)-SpikingVTG model we further remove the layer normalization to improve on-chip deployability.

B Loss Function Details

As described in the main paper, the total loss over N clips in the training set is defined as $L = \frac{1}{N}\sum_{i=1}^{N}(L_{f_i} + L_{d_i} + L_{c_i})$, where L_f represents the binary cross-entropy loss for the indicator variable f_i, L_d combines the smooth L1 loss with the generalized IoU loss [31] for the predicted boundaries, and L_c is an optional loss term incorporating intra- and inter-video contrastive learning [32]. We follow similar loss function construction as previous works on VTG [1, 6]. The loss for fore-ground parameter is given as follows,

$$L_f = -\lambda_f \left[f_i \log \tilde{f}_i + (1 - f_i) \log(1 - \tilde{f}_i) \right] \tag{9}$$

where, f_i is the true label and $\tilde{f}_i$ is the model prediction. The loss for predicted boundaries is given as follows,

$$L_d = \mathbf{1}_{f_i=1} \left(\lambda_{\text{L1}} L_{\text{SmoothL1}}(\tilde{d}_i, d_i) + \lambda_{\text{iou}} L_{\text{iou}}(\tilde{b}_i, b_i) \right) \tag{10}$$

where, d_i, b_i are the true label and $\tilde{d}_i, \tilde{b}_i$ is the model prediction. $L_c = \lambda_{\text{inter}} L_{\text{inter}} + \lambda_{\text{intra}} L_{\text{intra}}$ is used for inter-video and intra video contranstive learning [6]. For each video V, we randomly select a clip v_i with fore-ground indicator = 1 and positive saliency score. Clips from the same video, denoted as v_j, with saliency scores $s_j < s_i$ are treated as negative samples. i.e., $A = \{j \mid s_j < s_i, 1 \leq j \leq L_v\}$, and perform intra-video contrastive learning using the loss

$$L_{\text{intra}} = -\log \frac{\exp(\tilde{s}_i/\tau)}{\exp(\tilde{s}_i/\tau) + \sum_{j \in A} \exp(\tilde{s}_j/\tau)} \tag{11}$$

. Furthermore, we treat textual queries from other samples within the batch ($k \in S$) as negative samples, enabling inter-video contrastive learning for cross-sample supervision:

$$L_{\text{inter}} = -\log \frac{\exp(\tilde{s}_i/\tau)}{\sum_{k \in S} \exp(\tilde{s}_i^k/\tau)} \tag{12}$$

, where S is the training batch, $\tilde{s}_i^k = \cos(v_i, M_k)$ and M_k is the sentence representation (Eqn. 2) and cos is cosine similarity.

C Dataset Details

QVHighlights: The QVHighlights dataset [1] stands out as the sole dataset providing annotations for both moment retrieval and highlight detection, making it an excellent resource for benchmarking on both the VTG tasks. Comprising 10,148 videos with an average duration of 150 seconds. It features a total of 10,310 queries linked to 18,367 moments, resulting in an average of 1.8 distinct moments per query within each video. The dataset spans a variety of scenarios, including daily vlogs, travel vlogs, and news events.

Charades-STA: The Charades-STA [42] dataset comprises 16,128 indoor videos, each with an average duration of 30.6 seconds. It includes 12,408 query-interval pairs designated for training and 3,720 query-interval pairs reserved for testing.

TACoS: TACoS [43] consists of 127 videos, each averaging 4.78 minutes in length. The dataset is split into 75 videos for training, 27 for validation, and 25 for testing.

Youtube Highlights: YouTube Highlights [44] consists of 433 videos across 6 domains, using the domain names as text queries.

D Evaluation Metrics:

For QVHighlights, following previous work [1] we use Recall@1 with IoU thresholds of 0.3, 0.5 and 0.7 and avg. mean average precision (mAP), mAP@0.5 and mAP@0.75 as the evaluation metric for moment retrieval tasks. For highlight detection, we use mAP and HIT@1 [1], where a clip is considered a true positive if it receives a score of "Very Good" [5]. For Charades-STA and TACoS, we employ Recall@1 with IoU thresholds of 0.3, 0.5, and 0.7, along with the mean IoU (mIoU). For Youtube Highlights we use mAP.

E Additional Experimental Details

In this subsection, we provide a concise overview of the implementation details and provide additional experimental details. The GPU specifications for the experiments are detailed in the main paper, while the CPU utilized is an AMD Ryzen Threadripper 3960X 24-Core Processor. We have used Python and the PyTorch framework to write the code. The video and textual feature are developed following previous work [1, 6]. We have used the Adam optimizer to train our model. We list the hyper-parameters used in the work in Table 6. We used grid search to find optimal values.

E.1 Training Stages

Training a multi-modal spiking architecture like SpikingVTG is resource-intensive. To enhance the efficiency of this process and develop computationally efficient variants of our model, we leverage a

Hyper-parameters	Range	Optimal
N: Encoder Layers	(2-6)	4
D: Hidden Dimension	(768-2048)	1024
n_1: f-decoder depth	(1-5)	3
k_1: f-decoder kernel size	(3-9)	3
n_2: d-decoder depth	(1-5)	3
k_2: d-decoder kernel size	(3-9)	7
T_{CLRM}: Timesteps for CLRM	(5-100)	50
T_f: Timesteps for Finetuning	(5-50)	16
V_{th}: Threshold Potential	(0.5 - 2.0)	1.0
γ: Leaky-factor	(0.9 - 1.0)	0.99
λ_f :L_f co-efficient	(1 - 20)	10
λ_{L1} :$\mathrm{L}_{SmoothL1}$ co-efficient	(1 - 20)	10
λ_{intra} :L_{intra} co-efficient	(0 - 1.0)	0.05
λ_{inter} :L_{inter}-co-efficient	(0 - 1.0)	0.01
λ_{iou} :L_{iou} co-efficient	(1 - 20)	10
λ_{cos}: Sq. cosine weight (CLRM)	(0 - 1.0)	0.2
λ_{l_2}: L_2 weight (CLRM)	(0 - 1.0)	0.8
lr: Learning Rate	($1\mathrm{e}^{-5} - 1e^{-6}$)	$8\mathrm{e}^{-6}$
w_d: weight decay	($1\mathrm{e}^{-5} - 1e^{-3}$)	$1\mathrm{e}^{-4}$
Batch Size	(8-64)	32
Epochs: CLRM	10-100	50
Epochs: Finetuning	20-200	100

Table 6: Hyper-parameters of our SpikingVTG model. Optimal values for QVHighlights dataset is also shown.

multi-staged training framework. We utilize a transformer-based non-spiking VTG model (such as UniVTG) to perform CLRM loss optimization. After this initial stage, we fine-tune SpikingVTG using the true labels. Once the base SpikingVTG model is established, we modify its architecture to remove non-local operations and perform extreme quantization, followed by additional fine-tuning to create computationally efficient variants with minimal performance degradation. The resulting computationally efficient, lightweight models are well-suited for deployment on neuromorphic chips, enabling efficient inference.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

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 claims made in the abstract and Introduction are upheld in the methodological discussions and empirical results provided in Section 4 and 5.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

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

   Answer: [Yes]

Justification: We discuss the limitation of the proposed model in the Conclusions Section.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

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: Our work is primarily empirical; we demonstrate the model's convergence dynamics and validate the performance gains of our proposed techniques through comprehensive experiments on relevant benchmarks.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

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: Section 4 and 5 extensively goes over the primary contributions. We also provide additional experimental details (hyper-parameters, dataset details, etc.) in the Appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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 are uploading the code as part of the submission. If accepted we will make it open access on github.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.

- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: We have added detailed discussion on datasets, hyper-parameters, experimental setup in Section 5 and Appendix C, E.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

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: [Yes]

   Justification: We have added the experimental results following similar literature in Section 5.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

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: We have thoroughly discussed computational resource requirement in Section 5.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

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: The research conforms to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: This work aims to develop sustainable AI solutions that reduce the carbon footprint of machine learning models, thereby contributing to environmentally responsible and socially impactful AI.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

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

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

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: Proper citations to all resources are provided.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

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.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

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.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

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.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: This research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.