LAPIS: LAPLACIAN SPIKING ATTENTION VIA FIRST-SPIKE TIMING AND MEMBRANE LEAKAGE

Anonymous authors
Paper under double-blind review

ABSTRACT

Self-attention has become central to spiking vision transformers, yet its query-key scoring is still largely inherited from dense networks. Existing spiking variants either simplify dot product scoring or replace it with discrete operators, but spike timing, the native variable of a spiking network, does not directly define how tokens are related. We propose Lapis, a spiking attention mechanism that scores each token pair by the ℓ_1 distance between its query and key first-spike latency vectors under time-to-first-spike coding, and maps this distance to an affinity through a Laplacian kernel. The kernel's exponential decay matches the impulse response of a leaky integrate-and-fire membrane, so the accumulated latency difference determines the decay of a membrane trace, while row normalization reduces to a bit shift under power-of-two rounding. Scoring therefore needs only subtraction, absolute value, and accumulation, and removes all multiplication between query and key channels. Under a matched backbone and training schedule, Lapis reaches 96.56% top-1 accuracy on CIFAR-10, within 0.53 points of dot-product scoring. On ImageNet-1K, it reduces the estimated arithmetic energy of the attention path by $15.2\times$ relative to dense dot-product attention. The deployed 6-bit model attains 83.25% top-1 accuracy at an estimated arithmetic energy of 3.04 mJ per image.

1 INTRODUCTION

Spiking neural networks (SNNs) offer a brain-inspired and energy-efficient computing paradigm by processing information through sparse, event-driven spikes (Hu et al., 2024). With their growing success in visual recognition and other tasks, increasingly powerful architectures developed for artificial neural networks (ANNs) have been introduced into the spiking domain. In vision, the Vision Transformer (ViT) has made self-attention a central operation for recognition (Touvron et al., 2021; Bao et al., 2021), since attention relates every pair of tokens and captures the long-range dependencies that local operations miss. However, these architectures are primarily designed for dense, continuous-valued computation and therefore do not naturally align with the temporal representation and local dynamics of SNNs.

This mismatch has motivated a range of mechanisms that reduce the cost of standard self-attention. Some binarize queries, keys, and values to simplify interactions (Zhou et al., 2022), while others replace pairwise multiplication with masking, addition, or spike-gated operations (Yao et al., 2023; Zhou et al., 2024). Although effective in reducing arithmetic cost, these methods largely treat spikes as a computational constraint imposed on attention, rather than as a distinct representation of information. Their pairwise token similarity functions are therefore derived from simplified dot-product interactions or newly designed discrete operators (Xiao et al., 2025; Lee et al., 2025), while the temporal structure carried by spikes remains outside the relation function. This motivates a different design direction, which is defining token relations directly from spike timing.

TTFS coding makes this opportunity explicit by representing each activation with its first-spike latency (Park et al., 2020; Göltz et al., 2021). Nearby activation values produce nearby firing times, so the latency pattern can serve as a direct relation signal between tokens. Lapis therefore compares query and key tokens by the ℓ_1 distance between their first-spike latency vectors, rather than by a dot product over encoded activations. To convert temporal distance into interaction strength, Lapis applies a Laplacian kernel. Its ℓ_1 distance accumulates channel-wise latency differences, while

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

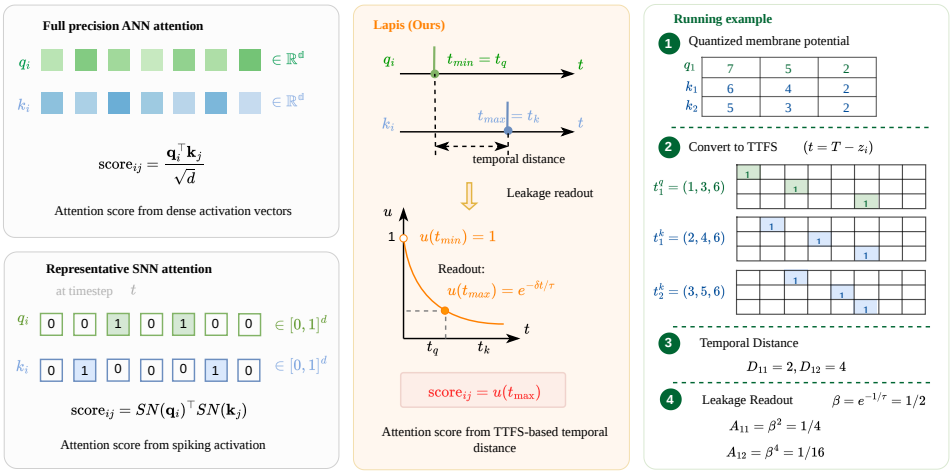

Figure 1: Rethinking query-key scoring with Lapis. ANN attention compares dense activations, while representative spiking attention compares binary states at one timestep. Lapis instead scores from first-spike timing. Channel-wise latency differences are accumulated into a single temporal distance, which a leaky trace converts into an affinity by decaying for that duration. The running example uses quantized membrane potentials as integer TTFS codes, with larger values mapped to earlier spikes.

its exponential decay assigns larger affinities to token pairs with closer first-spike patterns. The same exponential form follows the subthreshold decay of a leaky integrate-and-fire (LIF) membrane, linking the attention affinity to membrane leakage dynamics. Thus, spike timing determines the relation, and membrane decay determines how strongly the tokens interact. In this way, attention is no longer a dot-product operation imposed on spikes, but a temporal operation defined by the same dynamics that represent and process them.

In this paper, we propose Lapis, a novel temporal relation mechanism for spiking attention that defines token proximity from first-spike timing and realizes the resulting affinity through membrane leakage. This couples the attention relation to both the temporal representation and neuronal dynamics. Query-key distance computation requires only subtraction, absolute value, and accumulation, eliminating all multiplications between query and key channels. Power-of-two normalization further replaces general division with bit shifts.

Lapis is evaluated as an attention operator rather than as a complete architecture. All models are trained with quantized activations and converted losslessly to TTFS spiking networks. ImageNet models start from a pretrained vision transformer, and ablations use a fixed backbone and training schedule for controlled comparison. Under this protocol, the Laplacian relation stays within 0.53 points of dot-product scoring on CIFAR-10. On ImageNet-1K, Lapis-L reduces the estimated energy of the attention path from 100.01 to 6.59 mJ per image compared with dense dot product attention using the same architecture. It reaches 83.39% top-1 at 12.87 mJ, and 83.25% at 3.04 mJ once weights are quantized to 6 bits.

Our contributions are summarized as follows:

- We define token relations in spiking attention directly from first-spike latency vectors under TTFS coding. The resulting query-key score uses subtraction, absolute value, and accumulation, eliminating multiplication between query and key channels.
- We propose Lapis, which maps temporal distance to attention affinity through a Laplacian kernel aligned with LIF membrane leakage. This design replaces dot-product score computation and softmax normalization with a spike-native temporal process.
- We validate Lapis from CIFAR to ImageNet-1K. Lapis reaches 96.56% top-1 accuracy on CIFAR-10 and 83.25% on ImageNet-1K, while reducing the estimated arithmetic energy of the attention path by $15.2\times$ relative to dense dot-product attention under the same 45-nm operation-level model.

2 Related Work

Visual Spiking Neural Networks Visual SNNs are commonly developed through ANN-to-SNN conversion or direct training with surrogate gradients (Hu et al., 2024; Tavanaei et al., 2019). Recent conversion methods improve low latency inference through membrane initialization, activation shaping, and residual error correction (Bu et al., 2022; 2023; Huang et al., 2025), while direct training has scaled to deeper architectures through improved spike propagation and residual design (Fang et al., 2021; Zheng et al., 2021; Deng et al., 2022; Li et al., 2025). These advances primarily address trainability and efficient execution. Lapis instead studies how visual tokens should be related when information is represented by spike timing.

Spiking and Laplacian Transformers Existing spiking Transformers mainly adapt attention to binary or sparse computation. Spikformer uses spike-form queries, keys, and values without softmax, Spike-driven Transformer replaces dense interactions with sparse operations, and QKFormer develops linear complexity Q-K attention (Zhou et al., 2022; Yao et al., 2023; Zhou et al., 2024). Sorbet further explores shift based normalization, while STAtten and α-XNOR modify the interaction domain or similarity rule (Tang et al., 2024; Lee et al., 2025; Xiao et al., 2025). These methods still derive token relations from adapted attention operators or discrete matching rules rather than channel-wise first spike patterns. Outside the spiking domain, EcoTransformer replaces dot product scoring with an ℓ_1 Laplacian kernel, and LaplacianFormer further studies Laplacian relations for efficient dense attention (Gao et al., 2025; Feng et al., 2026). Lapis brings the Laplacian form into the temporal domain, where the distance is defined by first spike patterns and its decay is realized through membrane leakage.

Time-to-First-Spike Coding TTFS represents information through first spike latency, often with at most one spike per neuron, making it attractive for sparse inference (Göltz et al., 2021). Prior work improves TTFS training, thresholds, and decoding (Park et al., 2020; Wei et al., 2023; Che et al., 2026). TTFSFormer extends lossless TTFS conversion to conventional Transformer computations, while Otters exploits physical decay in optoelectronic devices for efficient temporal computation (Zhao et al., 2025; Yan et al., 2026). Lapis addresses the complementary problem of defining token relations directly from spike timing. It compares first spike patterns and maps their temporal distance to interaction strength through membrane leakage.

3 Preliminaries

Leaky Integrate-and-Fire Model The leaky integrate-and-fire (LIF) model describes a neuron by its membrane potential $u(t)$, which integrates incoming input while decaying toward a resting potential u_{rest}. Its subthreshold dynamics are given by

$$\tau_m \frac{\mathrm{d}u(t)}{\mathrm{d}t} = -\big(u(t) - u_{\mathrm{rest}}\big) + I(t), \tag{1}$$

where $I(t)$ denotes the input drive and $\tau_m > 0$ is the membrane time constant. When the membrane potential reaches a firing threshold, the neuron emits a spike and its potential is reset. In the absence of input and threshold crossings, the membrane potential follows

$$u(t + \Delta) - u_{\mathrm{rest}} = \big(u(t) - u_{\mathrm{rest}}\big) \exp\left(-\frac{\Delta}{\tau_m}\right). \tag{2}$$

The membrane time constant therefore determines how quickly the effect of a past input diminishes with elapsed time (Gerstner et al., 2014).

Time-to-First-Spike Coding Time-to-first-spike (TTFS) coding represents a neuronal response by the latency of its first spike relative to the beginning of a finite coding window. An active neuron therefore requires at most one spike, while silence represents an inactive response. The precise mapping between values and spike times is specified by the encoding rule (Park et al., 2020; Göltz et al., 2021).

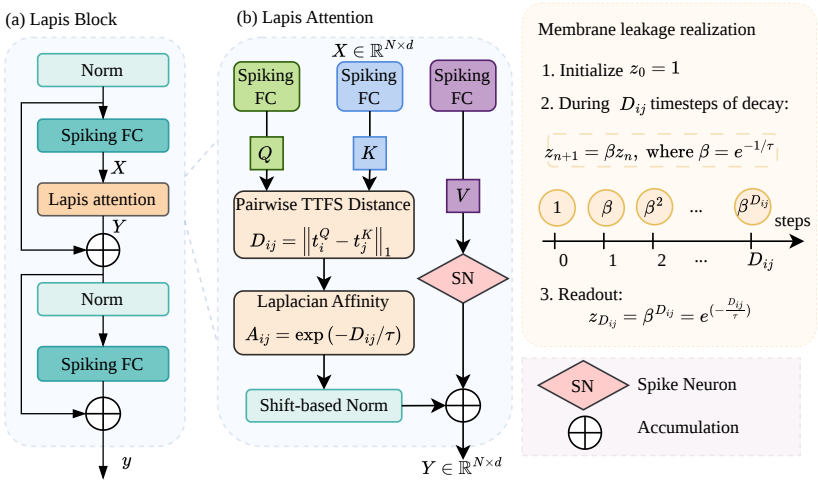

Figure 2: **Overview of the Lapis block and attention module.** A Lapis block follows a residual Transformer structure with Lapis attention and a spiking feed-forward network. Within Lapis attention, the query and key branches provide first-spike latencies whose pairwise ℓ_1 distances are mapped to Laplacian affinities. A power-of-two row denominator enables shift-based normalization. The value branch passes through an independent spiking neuron before weighted accumulation. The inset shows the membrane leakage realization. Initializing a unit state and applying D_{ij} decay steps yields $\beta^{D_{ij}}$, equal to the corresponding Laplacian affinity under a unit simulation step.

4 Method

4.1 Overview

The central idea of Lapis is to form attention weights from spike timing through neuronal leakage. Lapis retains the multi-head query, key, and value projections. For head h, let $\mathbf{t}_i^{Q,h}$ and $\mathbf{t}_j^{K,h}$ denote the channel-wise first-spike latency vectors of query token i and key token j, respectively. We define their temporal distance as

$$D_{ij}^h = \left\| \mathbf{t}_i^{Q,h} - \mathbf{t}_j^{K,h} \right\|_1, \tag{3}$$

where a smaller D_{ij}^h indicates closer agreement between the two first-spike patterns.

We convert the temporal distance into a positive, unnormalized affinity using a Laplacian kernel:

$$A_{ij}^h = \exp\left(-\frac{D_{ij}^h}{\tau_h} \right), \tag{4}$$

where $\tau_h > 0$ is a per-head learnable temporal scale. A smaller τ_h causes the affinity to decay more rapidly with temporal distance, whereas a larger τ_h permits larger differences in first-spike timing. The exponential decay follows the subthreshold dynamics of an LIF membrane, providing a leakage-based realization of the temporal affinity. For each query, Lapis accumulates the affinities over all keys and quantizes the resulting normalization factor to a power of two. Under fixed-point arithmetic, the final scaling can therefore be implemented by a bit shift.

Overall, Lapis forms attention weights from an ℓ_1 temporal distance through leakage-based decay and power-of-two normalization. Computing the query–key distance requires only subtraction, absolute value, and accumulation over channels, avoiding the channel-wise multiplications required by dot-product scoring.

4.2 Temporal Query and Key Relations

We now define the latency vectors used in Eq. 3. For each head h, let$S^{Q,h}, S^{K,h} \in \{0,1\}^{T\times N\times C}$ be the query and key spike tensors, where T is the simulation length, N is the number of tokens, and C is the channel dimension of the head. We take one simulation step as the unit of latency.

For any token spike train $X \in \{0,1\}^{T\times C}$, define the first-spike map $\phi(X) \in \{0,\dots,T\}^C$ by

$$[\phi(X)]_c = \begin{cases} \min\{n \mid X_{n,c} = 1,\ 0 \le n < T\}, & \text{if such } n \text{ exists,} \\ T, & \text{otherwise.} \end{cases} \tag{5}$$

The terminal value T denotes no spike within the simulation window. In our TTFS conversion, it is the code for zero activation.

The query and key latency vectors are

$$\mathbf{t}_i^{Q,h} = \phi(S_{:,i,:}^{Q,h}), \qquad \mathbf{t}_j^{K,h} = \phi(S_{:,j,:}^{K,h}). \tag{6}$$

Lapis defines their temporal relation by the ℓ_1 distance between the two latency code vectors as Equation 3.

4.3 Laplacian Affinity from Membrane Leakage

Given the temporal distance D_{ij}^h, Lapis assigns an affinity to each query-key pair by Eq. 4. Channels enter the affinity only through the additive ℓ_1 reduction. Lapis first accumulates the scalar D_{ij}^h and then maps it to the single affinity A_{ij}^h.

The exponential form in Eq. 4 matches the subthreshold decay of an LIF membrane. Let

$$\beta_h = \exp\left(-\frac{1}{\tau_h}\right), \qquad 0 < \beta_h < 1, \tag{7}$$

and consider the discrete leaky recurrence

$$u_{r+1} = \beta_h u_r, \qquad u_0 = 1, \tag{8}$$

with no input and no reset.

Proposition 1 (Leakage realization of Lapis affinity)**.** *For any query-key pair (i,j), the affinity in Eq. 4 is obtained by running the leaky recurrence in Eq. 8 for D_{ij}^h steps:*

$$u_{D_{ij}^h} = A_{ij}^h. \tag{9}$$

Proof. By Eq. 8, $u_r = \beta_h^r$ for any nonnegative integer r. Taking $r = D_{ij}^h$ and substituting Eq. 7 gives

$$u_{D_{ij}^h} = \beta_h^{D_{ij}^h} = \exp\left(-\frac{D_{ij}^h}{\tau_h}\right) = A_{ij}^h. \tag{10}$$

□

The leakage realization starts from a unit membrane trace and applies D_{ij}^h successive updates with the head-specific leakage factor β_h. The terminal membrane state gives the affinity A_{ij}^h. Thus, the accumulated first-spike distance directly controls the number of leak updates: closer latency patterns undergo less decay and produce stronger interactions. This realizes the Laplacian mapping through membrane dynamics without query–key channel multiplications.

Proposition 2 (Properties of the temporal affinity)**.** *For every head h and all token pairs (i,j):*

(i) Finite affinity set. *Since every first-spike latency lies in $\{0,\dots,T\}$, we have $D_{ij}^h \in \{0,\dots,CT\}$, and$A_{ij}^h \in \{\beta_h^r : r = 0,\dots,CT\}$, so A_{ij}^h takes at most $CT+1$ distinct values per head.*

(ii) Positivity and order consistency. $A_{ij}^h \in (0,1]$, *with* $A_{ij}^h = 1$ *if and only if* $\mathbf{t}_i^{Q,h} = \mathbf{t}_j^{K,h}$. *Moreover, for any keys* j, ℓ,

$$A_{ij}^h > A_{i\ell}^h \iff D_{ij}^h < D_{i\ell}^h,$$

i.e., the affinity is strictly decreasing in the temporal distance.

(iii) Latency stability. *If the query and key latencies are perturbed by* $\boldsymbol{\delta}_i^{Q,h}$ *and* $\boldsymbol{\delta}_j^{K,h}$ *with* $\|\boldsymbol{\delta}_i^{Q,h}\|_1 \le \eta_Q$ *and* $\|\boldsymbol{\delta}_j^{K,h}\|_1 \le \eta_K$, *then*

$$\left|\widetilde{A}_{ij}^h - A_{ij}^h\right| \le \frac{\eta_Q + \eta_K}{\tau_h}.$$

The proof is given in appendix.

4.4 Lapis Attention

Because the affinity row sum varies across queries, Lapis normalizes each row before value aggregation. For query i, let $Z_i^h = \sum_{j=1}^N A_{ij}^h$. Exact row normalization requires division by the general positive value Z_i^h. We instead approximate it by the nearest power of two:

$$k_i^h = \mathrm{round}\big(\log_2 Z_i^h\big), \qquad \widehat{Z}_i^h = 2^{k_i^h}. \tag{11}$$

The Lapis attention weight is then

$$\widehat{W}_{ij}^h = \frac{A_{ij}^h}{\widehat{Z}_i^h} = A_{ij}^h 2^{-k_i^h}. \tag{12}$$

Since k_i^h is an integer, the scaling in Eq. 12 can be implemented by a bit shift under fixed-point arithmetic.

For comparison, exact row normalization gives

$$W_{ij}^{*,h} = \frac{A_{ij}^h}{Z_i^h} = \mathrm{softmax}_j\left(-\frac{D_{ij}^h}{\tau_h}\right). \tag{13}$$

The following result bounds the approximation introduced by power-of-two scaling.

Lemma 1 (Power-of-two approximation)**.** *For every query* i, *key* j, *and head* h,

$$\frac{1}{\sqrt{2}} W_{ij}^{*,h} \le \widehat{W}_{ij}^h \le \sqrt{2} W_{ij}^{*,h}. \tag{14}$$

Since $\left|k_i^h - \log_2 Z_i^h\right| \le 1/2$, we have

$$2^{-1/2} \le \frac{Z_i^h}{\widehat{Z}_i^h} \le 2^{1/2},$$

which gives Lemma 1. A complete derivation is provided in appendix. The same scaling factor is applied to every key associated with a query. Consequently, power-of-two scaling preserves the relative affinities and their ordering, while keeping the row mass within $[1/\sqrt{2}, \sqrt{2}]$.

The value branch remains a spike train. Let $\mathbf{S}_{n,j}^{V,h} \in \{0,1\}^C$ denote the value spike vector of token j at step n. Lapis aggregates the value spikes as

$$\mathbf{Y}_{n,i}^h = \sum_{j=1}^N \widehat{W}_{ij}^h \mathbf{S}_{n,j}^{V,h}, \qquad 0 \le n < T. \tag{15}$$

The same token-mixing weights are applied at every simulation step, so the temporal structure of the value spikes is retained. We decode the time-resolved output through the temporal readout

$$R_i^h = \sum_{n=0}^{T-1} (T-n) Y_{n,i}^h. \tag{16}$$

The readouts are concatenated and quantized: $z_i^{\mathrm{out}} = Q_{\mathrm{out}}(\mathrm{Concat}_h\, R_i^h)$, where Q_{out} returns integer activation codes. These codes are then TTFS-encoded before the output projection.

4.5 Quantized Training and TTFS Conversion

Lapis is trained as a quantized neural network and converted to a TTFS SNN for inference. This avoids backpropagation through discrete first-spike events. During training, the quantized query and key representations are processed by the same Lapis operator defined in the preceding subsections.

Let $z \in \{0, \dots, T\}$ be an integer activation code. We convert it to a binary spike train $\mathcal{E}(z) \in \{0,1\}^T$ as

$$[\mathcal{E}(z)]_n = \mathbb{I}[z > 0 \ \wedge \ n = T - z], \qquad 0 \le n < T, \tag{17}$$

where $\mathbb{I}[\cdot]$ is the indicator function. Each positive activation code therefore produces one spike at its assigned time step, and larger codes produce earlier spikes. The encoding is applied elementwise to the quantized activations throughout the network. This encoding is lossless on the quantized codebook. Define the decoded value as $\mathcal{R}(S) = T - \phi(S)$, where $\phi(S) = T$ for a silent spike train. Then $\mathcal{R}(E(z)) = z$ for every $z \in \{0, \dots, T\}$. Thus, TTFS encoding introduces no additional representation error relative to the quantized activations. The conversion from the quantized Lapis network to its TTFS realization is lossless.

By construction, the first-spike time of the encoded activation is $\mathcal{T}(\mathcal{E}(z)) = T - z$. The conversion therefore preserves the query–key distance:

$$\begin{aligned} \left\| \mathcal{T}\big(\mathcal{E}(\mathbf{q}_i^h)\big) - \mathcal{T}\big(\mathcal{E}(\mathbf{k}_j^h)\big) \right\|_1 &= \left\| (T - \mathbf{q}_i^h) - (T - \mathbf{k}_j^h) \right\|_1 \\ &= \left\| \mathbf{q}_i^h - \mathbf{k}_j^h \right\|_1 . \end{aligned} \tag{18}$$

With the same temporal scale and power-of-two scaling rule, the QNN and the converted SNN produce identical Lapis affinities and attention weights. Under the TTFS encoding, weighting each value spike by its remaining coding-window duration recovers its integer code. The temporal readout in Eq. 16 therefore equals the QNN's weighted value aggregation, and applying the same output quantizer produces identical output codes.

5 Experiment

5.1 Experimental Setup

Datasets and models. We evaluate Lapis on CIFAR-10, CIFAR-100, and ImageNet-1K and report top-1 classification accuracy. We instantiate Lapis at three scales. Lapis-S is used for the CIFAR experiments, while Lapis-B and Lapis-L follow the ViT-B and ViT-L configurations with 86.7 M and 305.0 M parameters, respectively, and are evaluated on ImageNet-1K. The TTFS coding windows are set to $T = 15$ for Lapis-S and Lapis-B, and $T = 20$ for Lapis-L. Under the 6-bit deployment setting, the model sizes of Lapis-S are 18.0 MB on CIFAR-10 and 18.2 MB on CIFAR-100, while those of Lapis-B and Lapis-L are 70.7 MB and 238.1 MB, respectively. Relation function ablation variants use the same Lapis-S backbone and training configuration.

Training and conversion. CIFAR models are trained for 100 epochs with a batch size of 128 using AdamW, an initial learning rate of 1×10^{-4}, a weight decay of 0.05, and a cosine schedule with a 5-epoch linear warmup. Unless otherwise specified, CIFAR training uses both logit- and feature-level knowledge distillation. For ImageNet-1K, we initialize Lapis-B and Lapis-L from pretrained BEiT backbones and fine-tune them before short quantization-aware recovery stages. Lapis-L, our main model, is fine-tuned for 60 epochs with a global batch size of 162 and an initial learning rate of 8.5×10^{-5}. Full scale-specific settings are provided in the appendix.

Implementation. Our models are implemented in PyTorch 2.12 with CUDA 12.8 and trained on NVIDIA RTX PRO 6000 Blackwell Server Edition GPUs with 96 GB of memory. CIFAR experiments use a single GPU, whereas ImageNet experiments use distributed data parallel training on up to three GPUs. We use bfloat16 automatic mixed precision, while the ℓ_1 distance, exponential affinity, and normalization operations are evaluated in FP32 for numerical stability. Further training and quantization details are provided in the appendix.

Model	Type	Architecture	Energy (mJ) ↓	Time step T	Top-1 Acc. (%) ↑
DeiT (Touvron et al., 2021)	ANN	DeiT-B	80.50	–	81.80
PVT (Wang et al., 2021)	ANN	PVT-Large	45.08	–	81.70
BEiT (Bao et al., 2021)	ANN	ViT-B/16	–	–	83.20
BEiT (Bao et al., 2021)	ANN	ViT-L/16	–	–	85.20
Spikformer (Zhou et al., 2022)	DT-SNN	Spikformer-8-768	21.48	4	74.81
CML (Zhou et al., 2023)	DT-SNN	Spikingformer-8-768	16.30	4	77.64
QKFormer (Zhou et al., 2024)	DT-SNN	HST-10-768	8.52	1	81.69
			38.91	4	84.22
Max-Former (Fang et al., 2026)	DT-SNN	Max-10-768	5.27	1	78.60
			14.87	4	82.39
MST (Wang et al., 2023)	A2S	Swin-T (BN)	–	512	78.51
SpikeZIP-TF (You et al., 2024)	A2S	SViT-L-32Level	1,270.4[†]	64	83.82
Lapis	A2S	**Lapis-B**	1.50	15	80.32
		Lapis-B (FP)	5.35	15	80.58
		Lapis-L	3.04	20	83.25
		Lapis-L (FP)	12.87	20	83.39

Table 1: ImageNet-1K classification at 224×224. DT-SNN denotes direct training, while A2S denotes ANN-to-SNN conversion. Energy is the estimated per-image inference energy under the operation-level models used by the corresponding studies. For Lapis, T is the length of the TTFS coding window, with at most one spike per neuron, rather than the number of recurrent simulation steps. [†]Derived from SpikeZIP-TF's reported 19.85 W using 64 steps of 1 ms.

Method	CIFAR-10	CIFAR-100
Spikformer (2022)	95.51	78.21
SD Transformer (2023)	95.60	78.40
QKFormer (2024)	<u>96.18</u>	<u>81.15</u>
Lapis-S	96.11	81.02
Lapis-S(FP)	**96.56**	**81.41**

Table 2: Classification accuracy on CIFAR-10 and CIFAR-100. Lapis-S denotes the 6-bit deployment model, whereas Lapis-S (FP32) uses full-precision body weights.

5.2 Main Result

Table 1 compares Lapis with representative dense, directly trained, and converted models. Lapis-B and Lapis-L achieve 80.58% and 83.39% top-1 accuracy using 15-step and 20-step TTFS coding windows, respectively. The FP32 Lapis-L model requires an estimated 12.87 mJ per image. Quantizing its body weights to 6 bits reduces the energy to 3.04 mJ while retaining 83.25% accuracy, corresponding to a $4.24\times$ energy reduction. Lapis-L remains competitive in both accuracy and estimated energy among directly trained and converted spiking Transformers, reaching 83.39% top-1 with full precision weights and 83.25% with 6-bit weights. Compared with its dense BEiT-L counterpart, Lapis-L retains accuracy with a gap of only 1.81 percentage points, while reducing the estimated arithmetic energy by $22.0\times$ relative to the matched dense execution. Among conversion-based methods, it matches SpikeZIP-TF to within 0.57 points while using a TTFS coding window of 20 in which each neuron emits at most one spike, compared with 64 multi-spike simulation steps. Together, these results show that Lapis combines large-scale recognition accuracy with a compact temporal representation and low arithmetic cost.

Table 2 reports Lapis-S on CIFAR-10 and CIFAR-100. With full-precision body weights, Lapis-S reaches 96.56% and 81.41%, the highest accuracy among the compared models on both benchmarks, exceeding QKFormer by 0.38 and 0.26 points and Spikformer by 1.05 and 3.20 points. Quantizing the body weights to 6 bits costs 0.45 and 0.39 points, and the resulting deployment configuration stays within 0.13 points of QKFormer on both benchmarks. The same relation function and low-precision deployment path therefore transfer across model scales.

Table 3: Kernel ablation on CIFAR-10. All variants use the same Lapis-S backbone and training schedule. Hamming measures the number of channels with unequal first-spike latencies. It retains only binary match information and discards graded temporal proximity. "Mult.free" indicates computed without multiplication between query and key vectors.

Relation	Distance / score	Mult.free	Top-1 (%)
Softmax	Dot product	-	97.09
Gaussian	Squared ℓ_2	-	91.44
Hamming	Binary match	✓	87.92
Laplacian	ℓ_1	✓	96.56

5.3 Energy Analysis

We estimate arithmetic energy using the standard 45 nm operation-level model. The complete accounting protocol is provided in appendix. The Laplacian affinity is realized directly by membrane leakage over a duration determined by D_{ij}^h. The leakage process is part of the neuron dynamics and does not require query-key multiplications or an explicit exponential evaluation. Restricting the estimate to the attention path, dense dot-product attention requires 100.01 mJ per image under the same operation-level model, whereas the Lapis attention path requires 6.59 mJ, a 15.2× reduction, and 1.69 mJ once body weights are quantized to 6 bits. Relative to the architecture-matched dense execution, Lapis-L with full precision weights reduces the estimated energy from 283.15 mJ to 12.87 mJ per image, corresponding to a $22.0\times$ reduction before weight quantization. Using 6-bit body weights further reduces the energy to 3.04 mJ, yielding a $93.3\times$ reduction over dense execution while retaining 83.25% top-1 accuracy. The two operating points separate the sources of efficiency: single-spike TTFS execution establishes the primary gain, and low-precision weights provide an additional $4.24\times$ reduction.

5.4 Ablation Study

Relation function. Table 3 compares four relation designs using the same backbone and training configuration. The Laplacian relation reaches 96.56%, remaining within 0.53 points of softmax while outperforming the Gaussian kernel by 5.12 points. The Hamming relation, which retains only binary match information between first-spike latencies, reaches 87.92%, demonstrating the importance of preserving graded temporal proximity. In contrast, the Laplacian relation provides the desired operating point for Lapis, which is recognition accuracy close to dot product attention while computing query-key scores with subtraction, absolute value, and accumulation rather than multiplication. Its ℓ_1 form also aligns directly with the TTFS distance and membrane leakage.

Efficient deployment. On ImageNet-1K, the exact full precision configuration reaches 83.70% top-1 accuracy. Replacing exact row normalization with power-of-two scaling retains 83.39% accuracy while reducing the estimate to 12.87 mJ, showing that general division can be replaced by shift-based scaling with only a 0.31 point accuracy drop. Quantizing the body weights to 6 bits further reduces the energy to 3.04 mJ while retaining 83.25% accuracy.

6 Conclusion

We presented Lapis, a Laplacian spiking attention mechanism that forms query-key relations from first-spike latency vectors. Lapis maps ℓ_1 temporal distance to affinity through leakage-consistent decay and replaces general row normalization with power-of-two scaling. On ImageNet-1K, Lapis-L achieves 83.39% top-1 accuracy at an estimated 12.87 mJ with full precision weights and 83.25% at 3.04 mJ with 6-bit weights. The latter corresponds to a $93.3\times$ reduction in estimated arithmetic energy relative to architecture-matched dense execution. Results on CIFAR-10 and CIFAR-100 further confirm the effectiveness of the same low-precision deployment across model scales. These results establish first-spike timing as an effective and efficient basis for token relation modeling in spiking vision networks.

AI USE STATEMENT

In this work, we used generative AI tools for language polishing and improving the clarity of the manuscript. We have not used generative AI tools for the development of the methods, the conduct of experiments, or the formulation of scientific conclusions presented in this work. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021.

Tong Bu, Jianhao Ding, Zhaofei Yu, and Tiejun Huang. Optimized potential initialization for low-latency spiking neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pp. 11–20, 2022.

Tong Bu, Wei Fang, Jianhao Ding, PengLin Dai, Zhaofei Yu, and Tiejun Huang. Optimal ann-snn conversion for high-accuracy and ultra-low-latency spiking neural networks. *arXiv preprint arXiv:2303.04347*, 2023.

Kaiwei Che, Wei Fang, Peng Xue, Yifan Huang, Zhengyu Ma, and Yonghong Tian. Parallel training time-to-first-spike spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 1712–1720, 2026.

Shikuang Deng, Yuhang Li, Shanghang Zhang, and Shi Gu. Temporal efficient training of spiking neural network via gradient re-weighting. *arXiv preprint arXiv:2202.11946*, 2022.

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*, pp. 2661–2671, 2021.

Yuetong Fang, Deming Zhou, Ziqing Wang, Hongwei Ren, Lusong Li, Renjing Xu, et al. Spiking neural networks need high-frequency information. *Advances in Neural Information Processing Systems*, 38:103404–103425, 2026.

Zhe Feng, Sen Lian, Changwei Wang, Muyang Zhang, Tianlong Tan, Rongtao Xu, Weiliang Meng, and Xiaopeng Zhang. Laplacianformer: Rethinking linear attention with laplacian kernel. *arXiv preprint arXiv:2604.20368*, 2026.

Xin Gao, Xingming Xu, Shirin Amiraslani, and Hong Xu. Ecotransformer: Attention without multiplication. *arXiv preprint arXiv:2507.20096*, 2025.

Wulfram Gerstner, Werner M Kistler, Richard Naud, and Liam Paninski. *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014.

Julian Göltz, Laura Kriener, Andreas Baumbach, Sebastian Billaudelle, Oliver Breitwieser, Benjamin Cramer, Dominik Dold, Akos Ferenc Kungl, Walter Senn, Johannes Schemmel, et al. Fast and energy-efficient neuromorphic deep learning with first-spike times. *Nature machine intelligence*, 3(9):823–835, 2021.

Yangfan Hu, Qian Zheng, Guoqi Li, Huajin Tang, and Gang Pan. Toward large-scale spiking neural networks: A comprehensive survey and future directions. *arXiv preprint arXiv:2409.02111*, 2024.

Zhipeng Huang, Jianhao Ding, Zhiyu Pan, Haoran Li, Ying Fang, Zhaofei Yu, and Jian K Liu. Converting high-performance and low-latency snns through explicit modeling of residual error in anns. *IEEE Transactions on Neural Networks and Learning Systems*, 2025.

Donghyun Lee, Yuhang Li, Youngeun Kim, Shiting Xiao, and Priyadarshini Panda. Spiking transformer with spatial-temporal attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13948–13958, 2025.

Yudong Li, Yunlin Lei, and Xu Yang. Rethinking residual connection in training large-scale spiking neural networks. *Neurocomputing*, 616:128950, 2025.

Seongsik Park, Seijoon Kim, Byunggook Na, and Sungroh Yoon. T2fsnn: Deep spiking neural networks with time-to-first-spike coding. In *2020 57th ACM/IEEE design automation conference (DAC)*, pp. 1–6. IEEE, 2020.

Kaiwen Tang, Zhanglu Yan, and Weng-Fai Wong. Sorbet: A neuromorphic hardware-compatible transformer-based spiking language model. *arXiv preprint arXiv:2409.15298*, 2024.

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

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pp. 10347–10357. PMLR, 2021.

Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 568–578, 2021.

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*, pp. 1761–1771, 2023.

Wenjie Wei, Malu Zhang, Hong Qu, Ammar Belatreche, Jian Zhang, and Hong Chen. Temporal-coded spiking neural networks with dynamic firing threshold: Learning with event-driven backpropagation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10552–10562, 2023.

Yichen Xiao, Shuai Wang, Dehao Zhang, Wenjie Wei, Yimeng Shan, Xiaoli Liu, Yulin Jiang, and Malu Zhang. Rethinking spiking self-attention mechanism: implementing a-xnor similarity calculation in spiking transformers. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 5444–5454, 2025.

Zhanglu Yan, Jiayi Mao, Qianhui Liu, Fanfan Li, Tao Luo, Gang Pan, Bowen Zhu, and Weng-Fai Wong. Otters: An energy-efficient spiking transformer via optical time-to-first-spike encoding. In *International Conference on Learning Representations*, volume 2026, pp. 90203–90225, 2026.

Man Yao, Jiakui Hu, Zhaokun Zhou, Li Yuan, Yonghong Tian, Bo Xu, and Guoqi Li. Spike-driven transformer. *Advances in neural information processing systems*, 36:64043–64058, 2023.

Kang You, Zekai Xu, Chen Nie, Zhijie Deng, Qinghai Guo, Xiang Wang, and Zhezhi He. Spikezip-tf: Conversion is all you need for transformer-based snn. *arXiv preprint arXiv:2406.03470*, 2024.

Lusen Zhao, Zihan Huang, Jianhao Ding, and Zhaofei Yu. Ttfsformer: a ttfs-based lossless conversion of spiking transformer. In *Forty-second International Conference on Machine Learning*, 2025.

Hanle Zheng, Yujie Wu, Lei Deng, Yifan Hu, and Guoqi Li. Going deeper with directly-trained larger spiking neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 11062–11070, 2021.

Chenlin Zhou, Han Zhang, Zhaokun Zhou, Liutao Yu, Zhengyu Ma, Huihui Zhou, Xiaopeng Fan, and Yonghong Tian. Enhancing the performance of transformer-based spiking neural networks by snn-optimized downsampling with precise gradient backpropagation. *arXiv preprint arXiv:2305.05954*, 2023.

Chenlin Zhou, Han Zhang, Zhaokun Zhou, Liutao Yu, Liwei Huang, Xiaopeng Fan, Li Yuan, Zhengyu Ma, Huihui Zhou, and Yonghong Tian. Qkformer: Hierarchical spiking transformer using qk attention. *Advances in Neural Information Processing Systems*, 37:13074–13098, 2024.

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.

A Arithmetic Energy Estimation

A.1 Evaluation Protocol

We estimate the per-image arithmetic energy under the 45 nm operation model commonly adopted in prior work on spiking Transformers. We use 4.6 pJ for an FP32 multiply and accumulate operation and 0.9 pJ for an FP32 accumulation. In Lapis-W6, the linear layers use 6-bit weights, for which we use 0.1 pJ per integer accumulation. The temporal relation and value aggregation do not use quantized body weights and remain charged at 0.9 pJ per accumulation. The estimate counts explicit arithmetic operations. Membrane leakage is treated as intrinsic neuron dynamics rather than as repeated digital multiplications.

A.2 Window-Level TTFS Activity

We measure spike activity directly from each trained checkpoint. Forward hooks are attached to all activation quantizers, and the activity of operator ℓ is computed as

$$\rho_\ell = \frac{\sum_m \mathbb{I}[x_{\ell,m} > 0]}{\sum_m 1}, \tag{19}$$

where the sum is taken over all evaluated samples and tensor elements. Each positive integer activation produces exactly one spike during the complete TTFS window. Therefore, ρ_ℓ is the fraction of neurons that emit their single spike, rather than a per-timestep firing rate.

For a spike-driven operator transferred from dense operation count $N^{(\ell)}_{\text{dense}}$, the corresponding synaptic operation count is

$$N^{(\ell)}_{\text{AC}} = \rho_\ell N^{(\ell)}_{\text{dense}}. \tag{20}$$

No additional factor of T is introduced. This is a direct consequence of the single-spike TTFS representation. The equivalent average activity per timestep is ρ_ℓ/T, but it is not used to scale the operation count.

A.3 Operation Counts

Consider a model with L Transformer blocks, N tokens, embedding dimension D, H attention heads, and MLP expansion ratio R. The architecture-matched dense execution uses

$$N_{\text{QKV}} = 3LND^2, \tag{21}$$
$$N_{\text{rel}} = LN^2D, \tag{22}$$
$$N_{\text{value}} = LN^2D, \tag{23}$$
$$N_{\text{proj}} = LND^2, \tag{24}$$
$$N_{\text{MLP1}} = RLND^2, \tag{25}$$
$$N_{\text{MLP2}} = RLND^2. \tag{26}$$

All of these operations are counted as dense multiply–accumulates for the ANN reference.

For Lapis, the corresponding counts are

$$N^{\text{Lapis}}_{\text{QKV}} = \rho_{\text{attn}}\, 3LND^2, \tag{27}$$
$$N^{\text{Lapis}}_{\text{rel}} = LN^2D, \tag{28}$$
$$N^{\text{Lapis}}_{\text{value}} = \rho_V\, LN^2D, \tag{29}$$
$$N^{\text{Lapis}}_{\text{proj}} = \rho_{\text{out}}\, LND^2, \tag{30}$$
$$N^{\text{Lapis}}_{\text{MLP1}} = \rho_{\text{mlp}}\, RLND^2, \tag{31}$$
$$N^{\text{Lapis}}_{\text{MLP2}} = \rho_{\text{hid}}\, RLND^2. \tag{32}$$

Here, ρ_{attn} is the activity at the attention input, ρ_V is the mean activity measured at the quantized value outputs, ρ_{out} is the activity at the attention output, and ρ_{mlp} and ρ_{hid} are the activities at the two MLP stages.

The temporal relation is deliberately counted densely over all token pairs. Each of its LN^2D operations is charged as an absolute-difference accumulation. We do not apply an additional sparsity factor to this term. This gives a conservative estimate and attributes the reduction entirely to replacing channel-wise multiplication with subtraction, absolute value, and accumulation.

A.4 Leakage Readout and Row Scaling

After the temporal distance is accumulated, the affinity is obtained from the terminal membrane state after D_{ij}^h leakage updates. Membrane leakage is treated as intrinsic neuron dynamics rather than as an explicit arithmetic operation in the reported energy estimate. Exact row normalization and power-of-two normalization use the same affinity row sums. They differ only in the final scaling operation. Exact normalization requires

$$N_{\mathrm{div}} = LHN^2 \tag{33}$$

general scaling operations. Charging each such operation at $4.6\,\mathrm{pJ}$ adds approximately $0.07\,\mathrm{mJ}$ to Lapis-L. Power-of-two normalization replaces these operations with bit shifts, whose arithmetic cost is treated as negligible in the reported estimate.

B Proofs

B.1 Proof of Proposition 2

Proof. We prove the three properties separately.

(i) Finite affinity set. For every channel c, both $t_{i,c}^{Q,h}$ and $t_{j,c}^{K,h}$ belong to $\{0,\dots,T\}$. Hence,

$$\left|t_{i,c}^{Q,h} - t_{j,c}^{K,h}\right| \in \{0,\dots,T\}.$$

Since

$$D_{ij}^h = \sum_{c=1}^{C} \left|t_{i,c}^{Q,h} - t_{j,c}^{K,h}\right|,$$

the distance D_{ij}^h is an integer satisfying

$$0 \le D_{ij}^h \le CT.$$

It therefore belongs to $\{0,\dots,CT\}$. Using $\beta_h = \exp(-1/\tau_h)$, the corresponding affinity is

$$A_{ij}^h = \exp\left(-\frac{D_{ij}^h}{\tau_h}\right) = \beta_h^{D_{ij}^h}.$$

Consequently,

$$A_{ij}^h \in \{\beta_h^r : r = 0,\dots,CT\},$$

which contains at most $CT+1$ values.

(ii) Positivity and order consistency. Because $\tau_h > 0$ and $D_{ij}^h \ge 0$,

$$0 < \exp\left(-\frac{D_{ij}^h}{\tau_h}\right) \le 1.$$

Thus, $A_{ij}^h \in (0,1]$. Moreover,

$$A_{ij}^h = 1 \iff D_{ij}^h = 0 \iff \mathbf{t}_i^{Q,h} = \mathbf{t}_j^{K,h},$$

where the final equivalence follows from the definiteness of the ℓ_1 norm.

The function

$$f_h(x) = \exp\left(-\frac{x}{\tau_h}\right)$$

is strictly decreasing on $[0,\infty)$. Therefore, for any two keys j and ℓ,

$$A_{ij}^h > A_{i\ell}^h \iff D_{ij}^h < D_{i\ell}^h.$$

(iii) Latency stability. Let

$$\widetilde{\mathbf{t}}_i^{Q,h} = \mathbf{t}_i^{Q,h} + \boldsymbol{\delta}_i^{Q,h}, \qquad \widetilde{\mathbf{t}}_j^{K,h} = \mathbf{t}_j^{K,h} + \boldsymbol{\delta}_j^{K,h},$$

and define

$$\widetilde{D}_{ij}^h = \left\| \widetilde{\mathbf{t}}_i^{Q,h} - \widetilde{\mathbf{t}}_j^{K,h} \right\|_1 .$$

By the reverse triangle inequality,

$$\left| \widetilde{D}_{ij}^h - D_{ij}^h \right| \le \left\| \left(\widetilde{\mathbf{t}}_i^{Q,h} - \widetilde{\mathbf{t}}_j^{K,h} \right) - \left(\mathbf{t}_i^{Q,h} - \mathbf{t}_j^{K,h} \right) \right\|_1 \tag{34}$$

$$= \left\| \boldsymbol{\delta}_i^{Q,h} - \boldsymbol{\delta}_j^{K,h} \right\|_1 \tag{35}$$

$$\le \left\| \boldsymbol{\delta}_i^{Q,h} \right\|_1 + \left\| \boldsymbol{\delta}_j^{K,h} \right\|_1 \tag{36}$$

$$\le \eta_Q + \eta_K . \tag{37}$$

For $f_h(x) = \exp(-x/\tau_h)$,

$$|f_h'(x)| = \frac{1}{\tau_h} \exp\left(-\frac{x}{\tau_h} \right) \le \frac{1}{\tau_h}, \qquad x \ge 0.$$

The mean value theorem therefore gives

$$\left| \widetilde{A}_{ij}^h - A_{ij}^h \right| = \left| f_h\left(\widetilde{D}_{ij}^h \right) - f_h\left(D_{ij}^h \right) \right| \tag{38}$$

$$\le \frac{1}{\tau_h} \left| \widetilde{D}_{ij}^h - D_{ij}^h \right| \tag{39}$$

$$\le \frac{\eta_Q + \eta_K}{\tau_h}. \tag{40}$$

This completes the proof. □

B.2 Proof of Lemma 1

Proof. Recall that

$$k_i^h = \operatorname{round}\left(\log_2 Z_i^h \right), \qquad \widehat{Z}_i^h = 2^{k_i^h}.$$

Since k_i^h is the nearest integer to $\log_2 Z_i^h$,

$$\left| k_i^h - \log_2 Z_i^h \right| \le \frac{1}{2}.$$

Equivalently,

$$-\frac{1}{2} \le \log_2 Z_i^h - k_i^h \le \frac{1}{2}.$$

Exponentiating with base two gives

$$2^{-1/2} \le 2^{\log_2 Z_i^h - k_i^h} \le 2^{1/2}.$$

Using $\widehat{Z}_i^h = 2^{k_i^h}$, this becomes

$$\frac{1}{\sqrt{2}} \le \frac{Z_i^h}{\widehat{Z}_i^h} \le \sqrt{2}.$$

The exact and power-of-two normalized weights satisfy

$$W_{ij}^{\star,h} = \frac{A_{ij}^h}{Z_i^h}, \qquad \widehat{W}_{ij}^h = \frac{A_{ij}^h}{\widehat{Z}_i^h}.$$

Therefore,

$$\widehat{W}_{ij}^h = W_{ij}^{\star,h} \frac{Z_i^h}{\widehat{Z}_i^h}.$$

Because $W_{ij}^{\star,h} > 0$, multiplying the preceding bound by $W_{ij}^{\star,h}$ yields

$$\frac{1}{\sqrt{2}} W_{ij}^{\star,h} \le \widehat{W}_{ij}^{h} \le \sqrt{2} W_{ij}^{\star,h},$$

which proves the result.

The same identity also gives

$$\sum_{j=1}^{N} \widehat{W}_{ij}^{h} = \frac{Z_i^h}{\widehat{Z}_i^h} \in \left[\frac{1}{\sqrt{2}}, \sqrt{2}\right].$$

Moreover, every element in a row is multiplied by the same positive factor. Hence, for any keys j and ℓ,

$$\frac{\widehat{W}_{ij}^{h}}{\widehat{W}_{i\ell}^{h}} = \frac{A_{ij}^{h}}{A_{i\ell}^{h}} = \frac{W_{ij}^{\star,h}}{W_{i\ell}^{\star,h}},$$

so power-of-two scaling preserves all pairwise ratios and the ordering of the affinities. □

C Training and Quantization Details

Model settings. Lapis-S follows the small vision Transformer configuration and is used for CIFAR-10, CIFAR-100, and all relation function ablations. Lapis-B and Lapis-L use the corresponding BEiT-B/16 and BEiT-L/16 backbones on ImageNet-1K. The TTFS coding window is set to $T = 15$ for Lapis-S and Lapis-B, and to $T = 20$ for Lapis-L.

CIFAR training and distillation. Lapis-S is trained for 100 epochs with a batch size of 128. We use AdamW with an initial learning rate of 1×10^{-4}, a weight decay of 0.05, and a cosine schedule with five epochs of linear warmup.

The teacher is a BEiT-v2-Large model initialized from the `timm` ImageNet checkpoint `beitv2_large_patch16_224`. Before student training, the teacher is adapted separately to CIFAR-10 and CIFAR-100 for 20 epochs with a batch size of 64 and a learning rate of 5×10^{-5}.

The student objective combines classification, logit distillation, and feature distillation:

$$\mathcal{L} = 0.5\mathcal{L}_{\mathrm{CE}} + 0.5\mathcal{L}_{\mathrm{logit}} + 0.5\mathcal{L}_{\mathrm{feat}}. \tag{41}$$

The logit term uses soft KL divergence with distillation temperature $T_{\mathrm{KD}} = 4$. For feature distillation, we first apply layer normalization and then minimize the mean squared error between selected intermediate features. Student blocks 2, 5, 8, and 11 are aligned with teacher blocks 5, 11, 17, and 23, respectively. A learned linear projection maps the student feature dimension from 384 to the teacher dimension of 1024. All relation-function ablations use the same teacher, distillation losses, and training schedule.

Relation function ablations. All variants use the same projection architecture, value aggregation, and exact row normalization. For each head with $C = 64$, dot product uses $q_i^\top k_j/\sqrt{C}$, Gaussian uses $-\|q_i - k_j\|_2^2/[C(\tau^2 + \epsilon)]$, Hamming uses the number of unequal quantized codes scaled by $(\tau + \epsilon)^{-1}$, and Laplacian uses $-\|q_i - k_j\|_1/(\tau + \epsilon)$. For Gaussian, Hamming, and Laplacian, $\tau = \exp(\theta)$ is learned independently for each head and layer. Its initial value is 5 for Gaussian and Laplacian and 1.15 for Hamming. We use $\epsilon = 10^{-6}$. Dot product and Gaussian use signed LSQ query and key representations, whereas Hamming and Laplacian use nonnegative codes.

ImageNet fine-tuning. The ImageNet models are initialized from pretrained BEiT backbones and first optimized as quantized networks with exact row normalization and full-precision body weights. Lapis-L is fine-tuned for 60 epochs with two warmup epochs, a global batch size of 162, and an initial learning rate of 8.5×10^{-5}. Training uses distributed data parallel execution on three GPUs. Lapis-B is fine-tuned for 50 epochs with a global batch size of 256 and an initial learning rate of 1.3×10^{-4}.

Starting from the exact-normalized checkpoint, we recover the reported deployment configurations through short quantization-aware fine-tuning. The power-of-two W32 model is fine-tuned for 10

epochs with a batch size of 108 and a learning rate of 2×10^{-5}. The power-of-two W6 model is fine-tuned for 10 epochs with the same batch size and learning rate. This stage jointly enables power-of-two normalization and 6-bit body weights. No additional warmup is used in either recovery stage. The W6 stage uses a light augmentation recipe with `mix_prob`=0.1. For the final Lapis-L W6 model, the learned temporal scales range from 1.53 to 2.51 across heads and layers, with a mean of 1.86.

Activation quantization and TTFS conversion. Activations are quantized with learned step-size quantization into nonnegative integer codes

$$z \in \{0, \dots, T\}.$$

The zero code denotes an inactive neuron. Each positive code emits exactly one spike at

$$n = T - z, \qquad 0 \le n < T.$$

Larger activation codes therefore correspond to earlier spikes. The encoding is applied elementwise throughout the network. As established in the main paper, this affine conversion preserves the query–key ℓ_1 distance and therefore leaves the Lapis affinities unchanged.

Weight quantization. Lapis-W32 retains full-precision body weights. Lapis-W6 uses 6-bit per-channel quantization for the weight-bearing linear layers in each Transformer block. These layers include the query, key, and value projections, the attention output projection, and the two MLP projections. The patch embedding and classification head remain in FP32.

In the deployment configurations, the concatenated temporal readouts in Eq. 16 are quantized into integer codes and TTFS-encoded before the output projection. The power-of-two denominator is enabled during quantization-aware fine-tuning. This ensures that both the reported accuracy and the measured spike activity correspond to the same deployed checkpoint.