VILO-TM: VISUAL-LOGIC MAPPING FROM DEEP CNNS TO TSETLIN MACHINES

Anonymous authors
Paper under double-blind review

ABSTRACT

Tsetlin Machines (TMs) provide explicit conjunctive rules and gradient-free adaptation, yet learning expressive visual features jointly with their discrete clauses remains difficult. We introduce ViLo-TM, a family of vision TMs that jointly learns CNN representations and visually traceable Boolean rules. Its learned spatial-evidence-to-literal (SE2L) interface maps pooled convolutional channel responses to ternary predicate states and paired literals, while continuous relaxations of literal inclusion and conjunction enable end-to-end optimization. Quantization-aware training (QAT) and export yield an integer CNN encoder and a Boolean TM in which each fired clause exposes its literal conditions and signed vote, linked to the source channel maps. The Boolean TM supports bitwise inference and multiplication-free learning through standard TM feedback (Type I/II) with the encoder and thresholds fixed. On CIFAR-10, the deployed Boolean TM exceeds the strongest compared vision-TM baseline (90.15% vs. 82.8%; initializer selected on test accuracy). On CUB-200-2011, Boolean conversion retains most of the QAT accuracy (75.92% $\rightarrow$ 75.01%, mean of three runs), and the deployed TM exactly matches the reference Boolean clause outputs and class scores on all test images (5,794).

1 INTRODUCTION

Tsetlin Machines (TMs) (Granmo, 2018) learn classification rules as Boolean conjunctions with signed votes. Explicit clause supports specify the conditions required for each vote, while Boolean clause evaluation supports bitwise inference. Standard TM learning (Type I/II feedback) updates Tsetlin-Automaton (TA) states and clause weights without backpropagation, supporting adaptation on resource-constrained hardware (Rahman et al., 2025; Mao et al., 2025). The recognition performance of vision TMs depends on the discriminative information encoded by the Boolean input predicates. With predefined image encodings, clause learning optimizes combinations of fixed visual measurements while leaving the input representation unchanged. Hard threshold comparisons, discrete literal selection, and Boolean conjunction obstruct direct gradient propagation from TM class scores to convolutional neural network (CNN) parameters. Differentiable relaxations and surrogate gradients allow the TM classification loss to update the CNN encoder.

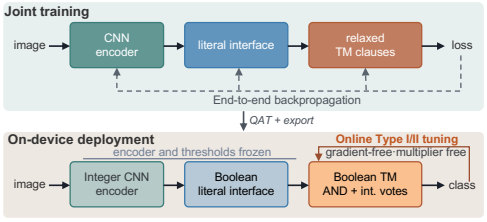


Figure 1: **How ViLo-TM operates.** Joint training updates the encoder, thresholds, and TM clauses; after export, the Boolean TM adapts through standard TM learning (Type I/II feedback) without multiplication or gradient computation.

We introduce **ViLo-TM**, a family of vision Tsetlin Machines that jointly learns CNN representations and visually traceable Boolean rules (Figure 1). Its learned spatial-evidence-to-literal (SE2L) **literal interface** maps pooled CNN channel responses to ternary predicate states and paired literals. Smooth threshold comparisons in the literal interface and relaxed literal selection and conjunction in the classifier, termed **ContinuousTM**, allow classification gradients to update the encoder and classifier jointly.

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.

Quantization-aware training (QAT) and export preserve the correspondence between channels and clause literals in an integer CNN encoder and a Boolean TM. Each fired clause exposes its literal conditions and signed vote, linked to the source channel maps. The deployment goal is to combine explicit conjunctive decisions with classifier updates that require neither gradients nor floating-point arithmetic: standard TM feedback (Type I/II) updates clause inclusions and weights while the encoder and thresholds stay fixed.

Contributions.

- **Joint learning of CNN representations and TM clauses.** A learned literal interface and ContinuousTM jointly optimize CNN representations, predicate thresholds, and clause parameters, allowing the TM classification objective to shape both the visual evidence and the rules that combine it.
- **Boolean deployment with multiplication-free learning.** QAT and export produce an integer visual front end and a Boolean TM. Starting from the exported clauses, standard TM feedback (Type I/II) updates literal inclusions and integer votes without gradients or multiplication, with the encoder and thresholds fixed.
- **Clause-level visual traceability.** Each clause's literal conditions and signed vote are linked to source channel maps, exposing which combinations of learned visual predicates enable a vote and which conditions block it.
- **Progress over prior vision TMs.** The deployed TM surpasses the compared vision-TM baselines on CIFAR-10 and retains most QAT accuracy after Boolean conversion on CUB. Corruption experiments further demonstrate that the Boolean classifier can adapt while keeping the learned visual representation fixed.

2 Background and Related Work

2.1 Tsetlin Machines

A TM (Granmo, 2018) augments K binary features with their complements to obtain literals $\ell \in \{0,1\}^{2K}$. For class c, clause j includes a subset $\mathcal{S}_{c,j}$ and computes a conjunction. Class scores sum the signed, magnitude-weighted clause outputs:

$$C_{c,j}(\ell) = \bigwedge_{i \in \mathcal{S}_{c,j}} \ell_i, \qquad v_c(\ell) = \sum_{j=1}^{M} \pi_{c,j} w_{c,j} C_{c,j}(\ell), \qquad \widehat{y} = \arg\max_c v_c(\ell). \tag{1}$$

In Eq. 1, M denotes the number of clauses per class, $\pi_{c,j} \in \{-1,+1\}$ is a fixed clause polarity, and $w_{c,j}$ is a nonnegative weight magnitude. Integer-weighted TMs learn the magnitudes $w_{c,j}$ (Abeyrathna et al., 2021). Excluded literals do not enter the conjunction; every included literal must be true for a nonempty clause to evaluate to one.

Each clause–literal pair is assigned a Tsetlin Automaton whose state determines whether the literal is included. Rewards and penalties update the state within or across the exclude and include regions. Type I feedback reinforces class-consistent patterns, while Type II feedback suppresses false activations by promoting the inclusion of a currently false literal. Bitwise clause evaluation and local automaton updates support efficient inference and on-chip learning (Rahman et al., 2025; Mao et al., 2025).

2.2 Related Work

Visual learning with TMs. The visual representation determines which distinctions are available to TM clauses. Convolutional TMs (CTM) reuse clauses across image patches and aggregate patch-level clause outputs by Boolean OR (Granmo et al., 2019). Coalesced TMs (CoTM) share clauses across outputs (Glimsdal & Granmo, 2021), and Drop Clause regularizes learning by stochastically dropping clauses during training (Sharma et al., 2023). TM Composites combine specialist TMs trained on different engineered image encodings (Granmo, 2023; Grønningsæter et al., 2024). Graph TMs (GraphTM) extend logical learning to graph-structured representations (Granmo et al., 2025). Convolutional and composite vision TMs use predefined image encodings; ViLo-TM instead lets classification gradients jointly shape CNN features, predicate thresholds, and TM clauses.

Visual explanations and prototype representations. Visual explanations can identify influential evidence, expose semantic intermediates, or reveal the conditions governing a decision. Grad-CAM weights spatial activation maps by class gradients to localize influential image regions (Selvaraju et al., 2017). Concept bottleneck models predict through semantic variables learned from concept annotations (Koh et al., 2020). ProtoPNet and PIP-Net express class scores as weighted sums of prototype evidence (Chen et al., 2019; Nauta et al., 2023); PIP-Net uses sparse, nonnegative classifier weights. ProtoTree exposes combinations of prototype-presence and prototype-absence tests along decision paths (Nauta et al., 2021). ViLo-TM expresses class scores as sums of signed conjunctive votes over learned visual predicates. In the deployed Boolean TM, each clause casts its signed vote only if all included literals are satisfied; any false included literal blocks that vote. Linking literal conditions to their source channel maps shows which combinations of visual responses enable a vote and which conditions rule it out.

Neuro-symbolic and differentiable logic. Logic Tensor Networks learn neural groundings of first-order symbols by optimizing formula satisfaction under differentiable fuzzy semantics (Badreddine et al., 2022). Analyses of differentiable fuzzy operators characterize how the choice of logical relaxation affects gradient propagation (van Krieken et al., 2022). ContinuousTM relaxes literal selection and conjunction to learn clause structure jointly with the CNN encoder and predicate thresholds.

Continuous training and discrete deployment. Differentiable logic gate networks learn continuous gate-selection parameters and discretize the learned networks into Boolean gates (Petersen et al., 2022). Convolutional differentiable logic gate networks extend the architecture with logic-gate convolutions and OR pooling (Petersen et al., 2024). Truth Table Net trains differentiable filters with a dual truth-table representation from which Boolean circuits can be extracted (Benamira et al., 2024). ViLo-TM's exported clause supports set the initial include/exclude actions of Tsetlin automata; local Type I/II feedback then updates inclusions and integer votes without multiplication, with the encoder and thresholds fixed.

3 METHOD

ViLo-TM jointly learns a CNN and TM clauses: classification gradients pass through the differentiable literal interface and ContinuousTM to the encoder, thresholds, and clauses. The exported classifier keeps the channel–literal correspondence but uses Boolean conjunctions and integer votes (Figure 2).

3.1 VISUAL ENCODER AND LITERAL INTERFACE

Channel responses. Following PIP-Net (Nauta et al., 2023), a CNN produces K spatial channels $A(x) \in \mathbb{R}^{K \times H_s \times W_s}$ over an $H_s \times W_s$ feature grid for image x. Channel-wise softmax at each location u and spatial max pooling give

$$S_k(x,u) = \frac{\exp A_k(x,u)}{\sum_{k'=1}^{K} \exp A_{k'}(x,u)}, \qquad r_k(x) = \max_{u \in \Omega} S_k(x,u), \tag{2}$$

where $\Omega = \{1,\dots,H_s\} \times \{1,\dots,W_s\}$ is the feature grid. The map $S_k(x,u)$ provides spatial localization, with peak $u_k^*(x) = \arg\max_u S_k(x,u)$ identifying the region used to inspect channel k. The pooled response $r_k(x)$ determines the predicate state supplied to the classifier.

Ternary states and Boolean literals. The literal interface (SE2L) converts each channel response into conditions that a clause can use to rule out its activation. For each channel k, two learned thresholds $0 < \theta_k^- < \theta_k^+ < 1$ define LOW and HIGH states separated by an intermediate DON'T CARE interval:

$$z_k(x) = \begin{cases} 0 & r_k(x) \le \theta_k^- \quad \text{(LOW)}, \\ X & \theta_k^- < r_k(x) < \theta_k^+ \quad \text{(DON'T CARE)}, \\ 1 & r_k(x) \ge \theta_k^+ \quad \text{(HIGH)}. \end{cases} \tag{3}$$

With the indicator function $\mathbb{I}[\cdot]$, positive and negative literals encode the ternary state as

$$\widetilde{\ell}_k^+ = \mathbb{I}[z_k \neq 0], \qquad \widetilde{\ell}_k^- = \mathbb{I}[z_k \neq 1], \qquad 0 \mapsto (0,1), \quad X \mapsto (1,1), \quad 1 \mapsto (1,0). \tag{4}$$

In a deployed Boolean clause, an included positive literal blocks the entire clause when its predicate is LOW, and an included negative literal blocks it when its predicate is HIGH. DON'T CARE acts

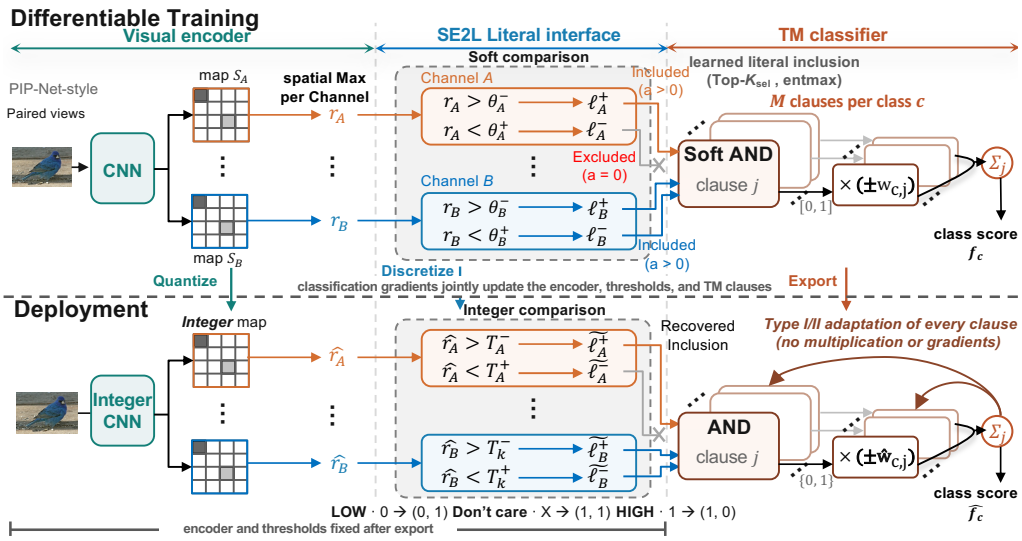


Figure 2: **Joint CNN–TM learning and Boolean deployment.** Top: classification gradients connect the CNN, literal interface, and ContinuousTM. Bottom: exported Boolean clauses support integer voting and multiplication-free Type I/II adaptation with the encoder and thresholds fixed. Colors link literals to source channels; stacked boxes: M clauses per class; a: entmax coefficient.

as a wildcard: both literal types are true, so that channel does not veto the clause. Interpreting these response conditions as the presence or absence of a named visual concept requires separate semantic validation.

Continuous approximation. For gradient-based learning, we approximate the two threshold comparisons with sigmoid transitions $\sigma(t) = (1 + e^{-t})^{-1}$ of sharpness $s_k > 0$:

$$\ell_k^+ = \sigma\big(s_k(r_k - \theta_k^-)\big), \qquad \ell_k^- = \sigma\big(s_k(\theta_k^+ - r_k)\big), \qquad \ell(x) = [\ell^+(x), \ell^-(x)] \in [0,1]^{2K}. \quad (5)$$

The overlap between the high-valued regions of the positive- and negative-literal sigmoids approximates state X, and the sigmoids propagate classification gradients to r_k, θ_k^-, and θ_k^+.

3.2 Differentiable Tsetlin Machine

Sparse literal routing. Each clause learns which channel conditions to combine. We use noisy grouped Top-K_{sel} selection, adapted from sparsely gated mixture-of-experts models (Shazeer et al., 2017), to propose candidate literals and entmax to learn their sparse routing coefficients. For class c, clause j, and literal i, training perturbs a learned preference $g_{c,j,i}$:

$$h_{c,j,i} = g_{c,j,i} + \nu_{c,j}\epsilon_{c,j,i}, \qquad \epsilon_{c,j,i} \sim \mathcal{N}(0,1), \quad (6)$$

where $\nu_{c,j} > 0$ is a learned noise scale. Grouped selection first retains the higher-scoring of each channel's positive and negative literals, then selects the K_{sel} highest-scoring of these retained literals. The candidate set $\mathcal{I}_{c,j}$ therefore contains at most one literal per channel, excluding pairs whose conjunction would be satisfied only in the DON'T CARE state. Routing noise perturbs candidate rankings across training steps, changing the literal candidates selected for gradient updates.

Entmax (Peters et al., 2019) assigns sparse nonnegative coefficients to the candidates:

$$a_{c,j,\mathcal{I}_{c,j}} = \text{entmax}_{\alpha_{c,j}}\left(h_{c,j,\mathcal{I}_{c,j}}/\tau_g\right), \qquad \mathcal{S}_{c,j} = \{i : a_{c,j,i} > 0\}. \quad (7)$$

In Eq. 7, $\tau_g > 0$ denotes the routing temperature and $\alpha_{c,j} \in (1, 2]$ controls sparsity. The coefficients sum to one over the candidate set and are zero outside it. The support $\mathcal{S}_{c,j}$ identifies the literals included in clause (c, j). Gradients propagate through the entmax coefficients to the selected routing preferences.

Soft conjunction and class votes. For a nonempty set of binary literals, Boolean conjunction equals the minimum literal value. ContinuousTM approximates the minimum over the selected

literals using an entmax-weighted log-sum-exp operator with sharpness $\beta > 0$:

$$C_{c,j}^{\text{soft}}(\ell) = -\frac{1}{\beta} \log \sum_{i \in \mathcal{S}_{c,j}} a_{c,j,i} \exp(-\beta \ell_i), \qquad f_c(x) = \sum_{j=1}^{M} \pi_{c,j} w_{c,j} C_{c,j}^{\text{soft}}(\ell(x)). \tag{8}$$

For fixed nonempty support and fixed positive coefficients, the soft conjunction in Eq. 8 converges to $\min_{i \in \mathcal{S}_{c,j}} \ell_i$ as $\beta \to \infty$. Clause polarities $\pi_{c,j}$ alternate between positive and negative, while learned magnitudes $w_{c,j} \geq 0$ scale the signed clause outputs.

Learning conjunctive conditions. For the soft conjunction in Eq. 8, the derivative with respect to an included literal is

$$\frac{\partial C_{c,j}^{\text{soft}}}{\partial \ell_i} = \frac{a_{c,j,i} \exp(-\beta \ell_i)}{\sum_{i' \in \mathcal{S}_{c,j}} a_{c,j,i'} \exp(-\beta \ell_{i'})}. \tag{9}$$

For equal routing coefficients, Eq. 9 assigns larger derivatives to lower literal values, focusing the training signal on the conditions that limit the conjunction. The classification loss determines whether a clause is reinforced or suppressed, so these derivatives do not always drive included literals toward satisfaction. Classification gradients update the corresponding channel responses, thresholds, and routing preferences. Clause-routing parameters are shared across images, defining a common preference over visual predicates for each clause.

3.3 Joint Learning

We jointly optimize the CNN encoder, literal-interface thresholds, and ContinuousTM parameters using

$$\mathcal{L} = \lambda_{\text{CE}} \mathcal{L}_{\text{CE}} + \lambda_{\text{align}} \mathcal{L}_{\text{align}} + \lambda_{\text{usage}} \mathcal{L}_{\text{usage}} + \lambda_{\text{load}} \mathcal{L}_{\text{load}} + \lambda_{\text{div}} \mathcal{L}_{\text{div}}, \tag{10}$$

where each $\lambda_{(\cdot)} \geq 0$ weights the corresponding loss term. Cross-entropy $\mathcal{L}_{\text{CE}}$ is computed from the ContinuousTM class scores and ground-truth labels. The optional alignment and usage losses follow PIP-Net (Nauta et al., 2023): alignment promotes consistent channel activations at corresponding locations in paired views, while usage promotes channel activation across the image batch. Load balancing regularizes expected selection frequencies across channels. Clause diversity penalizes squared cosine similarities between distinct clause-routing preference vectors within each class, encouraging different literal preferences.

We initialize the literal interface and TM against a fixed visual representation before allowing classification gradients to reshape the encoder. An optional representation warm-up precedes this initialization, and the alignment and usage losses may also regularize subsequent joint training. Appendix B specifies threshold initialization, loss coefficients, and the joint-learning and QAT schedules.

3.4 Discrete Deployment and Adaptation

Integer visual encoding and literals. QAT prepares the visual encoder and literal interface for deployment by using the same integer forward computations and hard literal comparisons as the exported model; straight-through surrogate gradients (Jacob et al., 2018) let classification learning continue through these discrete operations. Appendix C specifies quantization and rounding conventions.

The pooled integer response $\widehat{r}_k$ and both thresholds share a probability scale Q. With $T_k^- = \lfloor Q\theta_k^- \rfloor$ and $T_k^+ = \lceil Q\theta_k^+ \rceil$, the literal interface evaluates

$$\widetilde{\ell}_k^+ = \mathbb{I}[\widehat{r}_k > T_k^-], \qquad \widetilde{\ell}_k^- = \mathbb{I}[\widehat{r}_k < T_k^+]. \tag{11}$$

Concatenating the positive and negative literal vectors produces the Boolean TM input $\widetilde{\ell}(x) = [\widetilde{\ell}^+(x), \widetilde{\ell}^-(x)] \in \{0,1\}^{2K}$. Responses strictly between the integer thresholds retain the DON'T CARE encoding $(1,1)$.

Boolean clauses and integer votes. Export disables routing noise and calibrates a global entmax parameter α_{exp} to a target mean clause length. Let $\widehat{a}_{c,j,i}$ denote the resulting sparse routing coefficients. A positive coefficient includes literal i in clause (c,j), while a zero coefficient excludes it:

$$m_{c,j,i}^{(0)} = \mathbb{I}[\widehat{a}_{c,j,i} > 0], \qquad \mathcal{S}_{c,j}^{\text{exp}} = \{i : m_{c,j,i}^{(0)} = 1\}. \tag{12}$$

At export, grouped selection permits at most one literal per channel, and each clause uses the same support across inputs. A shared scale quantizes magnitudes to integer weights $\widehat{w}_{c,j}$ while preserving clause polarities. Boolean inference computes

$$\widetilde{C}_{c,j}(x) = \begin{cases} \bigwedge_{i \in \mathcal{S}^{\exp}_{c,j}} \widetilde{\ell}_i(x), & \mathcal{S}^{\exp}_{c,j} \neq \varnothing, \\ 0, & \text{otherwise}, \end{cases} \qquad \widehat{f}_c(x) = \sum_{j=1}^{M} \pi_{c,j} \widehat{w}_{c,j} \widetilde{C}_{c,j}(x). \tag{13}$$

The predicted class maximizes $\widehat{f}_c$. Retained channel indices and literal types (positive or negative) identify the spatial channel response behind each condition of an exported rule.

Automata-based adaptation. For Tsetlin-Automaton states $A^{\mathrm{TA}}_{c,j,i} \in \{0, \ldots, 2R-1\}$ with R states per action, the action after t updates is $m^{(t)}_{c,j,i} = \mathbb{I}[A^{\mathrm{TA},(t)}_{c,j,i} \geq R]$. Exported actions $m^{(0)}$ select each automaton's initial include/exclude region, and normalized routing preferences set its state offset within that region; subsequent Type I/II updates change $m^{(t)}$ through the automaton states.

Type I/II feedback updates automaton states and integer clause weights using labeled samples. Labels and clipped class votes determine feedback selection; clause outputs, literal values, and specificity determine reward and penalty probabilities. State transitions across the action boundary change inclusion, while weight updates change vote magnitudes. The encoder and thresholds remain fixed.

4 Experiments

We evaluate the contribution of joint learning to recognition and the accuracy retained after Boolean conversion. We then examine the visual evidence and semantic consistency of clause conditions, and compare adaptation accuracy and computational requirements across classifiers on shared frozen literals.

4.1 Experimental Setup

To span a range of visual complexity, recognition experiments cover handwritten digits and clothing images (MNIST (LeCun et al., 1998), Fashion-MNIST (Xiao et al., 2017)), multi-class object recognition (CIFAR-10 and CIFAR-100 (Krizhevsky, 2009)), and fine-grained bird species (CUB-200-2011 (Wah et al., 2011)). Classifier adaptation is evaluated under corruption on MNIST-C (Mu & Gilmer, 2019) and CIFAR-10-C (Hendrycks & Dietterich, 2019). The CUB configuration (Table 1) uses an initializer, a pretrained CNN checkpoint, trained only on the model-training split (4,794 images), with both initialization and checkpoint selection based on internal validation (1,200 images), excluding the official test split. Initializers for the four smaller benchmarks were selected using official-test accuracy (run records), so these results are not independent held-out estimates. Appendix A provides the data partitions, training schedules, and run details.

4.2 Recognition Accuracy and Joint Learning

ViLo-TM improves on the reported TM Composites result on CIFAR-10 (90.15%; $+7.35$ percentage points, pp; Table 1(a)) and extends recognition to fine-grained bird classification on CUB-200-2011 (75.01% across 200 species). ViLo-TM also records 70.04% on CIFAR-100 versus 52.2% for TM Composites; both ViLo-TM CIFAR runs use test-selected initializers (Section 4). On MNIST and Fashion-MNIST, performance remains below the strongest reported TM baselines (99.24% vs. 99.45% and 92.83% vs. 94.68%, respectively) (Sharma et al., 2023; Granmo, 2023; Hnilov, 2025).

To examine the contribution of encoder updates, we compare joint learning with a frozen-encoder control that shares the same initializer and trains the literal interface and TM while keeping CNN weights and batch-normalization statistics fixed. On CUB, joint training with delayed encoder updates and without the alignment and usage objectives outperforms the frozen-encoder control (75.63% vs. 69.35%; $+6.28$ pp; matched run with the same initializer). The standard schedule gives lower accuracy (63.20%) and mixed changes on MNIST, Fashion-MNIST, and CIFAR-10 (-1.04 to $+0.44$ pp), so the gain is obtained with delayed, classification-driven encoder updates.

Table 1: **Recognition accuracy and Boolean conversion.** (a) Official-test accuracy. (b) Cumulative accuracy change from QAT after each conversion step (percentage points). ViLo-TM: mean ± s.d. over three runs in (a) and mean over three runs in (b). Bold: best per column among literature TMs and deployed ViLo-TM.

(a) Recognition performance

	Accuracy				
Method	MNIST	FMNIST	C10	C100	CUB
Tsetlin Machine vision models					
CTM	99.40%	91.50%	69.30%	35.50%	–
Drop Clause	**99.45%**	92.50%	75.10%	45.20%	–
CoTM	99.31%	91.83%	66.42%[†]	–	–
TM Composites	–	**93.0%**[‡]	82.80%	52.2%[‡]	–
GraphTM	$98.42 \pm 0.05\%$	$87.07 \pm 0.08\%$	$70.28 \pm 0.17\%$	–	–
ViLo-TM					
ViLo-TM (QAT)	$99.29 \pm 0.08\%$	$93.12 \pm 0.14\%$	$90.37 \pm 0.19\%$	$71.25 \pm 0.65\%$	$75.92 \pm 0.40\%^*$
ViLo-TM (Boolean)	$99.24 \pm 0.06\%$	$92.83 \pm 0.26\%$	$\mathbf{90.15 \pm 0.22\%}$	$\mathbf{70.04 \pm 0.78\%}$	$\mathbf{75.01 \pm 0.59\%}^*$

Sources: CTM (Granmo et al., 2019; Sharma et al., 2023); Drop Clause (Sharma et al., 2023); CoTM (Glimsdal & Granmo, 2021); TM Composites (Grønningsæter et al., 2024); GraphTM (Granmo et al., 2025); [‡]Granmo (2023); FPTM, not listed, reports 94.68% on Fashion-MNIST (Hnilov, 2025). [†]CoTM CIFAR-10 uses the input encoding and clause budget from the GraphTM benchmark (Granmo et al., 2025). [*]CUB uses a validation-selected initializer (Section 4.2). FMNIST/C10/C100: Fashion-MNIST/CIFAR-10/CIFAR-100. (b) uses fixed-point votes with 16 fractional bits; three-bit votes are used for adaptation (Table 3) and the Figure 3 case study, while the quantitative clause and intervention analyses use $F = 16$ votes.

(b) Cumulative conversion to a Boolean TM

Conversion stage (cumulative)	MNIST	FMNIST	C10	C100	CUB
QAT accuracy	99.29%	93.12%	90.37%	71.25%	75.92%
+ integer CNN and literal interface	0.00	0.00	0.00	0.00	0.00
+ global export routing	−0.03	−0.19	−0.08	−0.80	−0.51
+ exact Boolean AND	−0.05	−0.29	−0.22	−1.21	−0.91
+ integer votes (deployed classifier)	−0.05	−0.29	−0.22	−1.21	−0.91

After the first QAT epoch, the jointly trained model shows a smaller drop in internal-validation accuracy (76.1% → 73.8%) than the frozen-encoder control (76.2% → 68.7%; Appendix F).

We compare classifiers using the same frozen CUB encoder and Boolean literals: linear and multilayer perceptron (MLP) classifiers achieve higher accuracy than the Boolean TM (77.51% and 76.88% vs. 75.63%; Appendix A). Floating-point CNNs with linear classifiers also outperform the corresponding Boolean models across the five benchmarks (by 0.28–5.27 pp; Table 4); because these references also differ in representation learning and numerical precision, this gap is not attributable to the classifier alone.

4.3 Conversion to a Boolean TM

Boolean conversion retains most QAT accuracy (mean losses of 0.05–1.21 pp on the four smaller benchmarks; 0.91 pp on CUB; Table 1(b)). Because QAT already uses the deployed integer front end and hard literals, the integer CNN and literal interface convert without change (0.00 pp). Accuracy decreases at global routing recalibration, which changes literal supports and soft-conjunction coefficients, and at Boolean AND; vote quantization causes no further change.

On all 5,794 CUB test images, the deployed classifier exactly matches the reference Boolean clause outputs and class scores. QAT and Boolean predictions agree on 91.35% of these images.

4.4 Visual Evidence and Clause-Level Interpretation

Following Co-12 (Nauta & Seifert, 2023), we evaluate computational traceability, which reconstructs clause votes from included literals and source channels, and semantic interpretability, which measures correspondence between channel responses and recognizable visual concepts. We evalu-

Table 2: **Visual grounding.** (a) Official-test concept localization for 511 channels with training-assigned concepts. Peak: annotated point inside the peak-centered 64×64 patch; Control: same-image grid. (b) CUB part purity over 512 channels (mean $\pm$ s.d.), with the best-matching part reselected within each split.

(a) Concept localization

Test	Retrievals	Peak	Control
Concept	Concept visible	48.57%	9.03%
Concept	All top-20	38.35%	7.06%
+ color/pattern	Visible, attr. known	40.24%	7.32%
+ color/pattern	All top-20	32.25%	5.74%

(b) CUB part purity

Split	Images	Purity ↑
Model training	4,794	0.42 ± 0.14
Internal validation	1,200	0.36 ± 0.11
Official test	5,794	0.39 ± 0.12

ate ViLo-TM through clause visualization, training-assigned concept localization, anatomical part purity, semantic agreement, and spatial interventions. Appendix E specifies the evaluation protocols.

Figure 3 shows how included literal conditions determine a clause's vote. Spatial interventions further connect clause activation to image evidence: replacing the peak region of a channel whose predicate is HIGH with the ImageNet mean moves that predicate out of the HIGH state and switches off the clause more often than replacing a control region matched for size and bird-box coverage (27.3% vs. 3.2% and 20.1% vs. 8.6%; Appendix E).

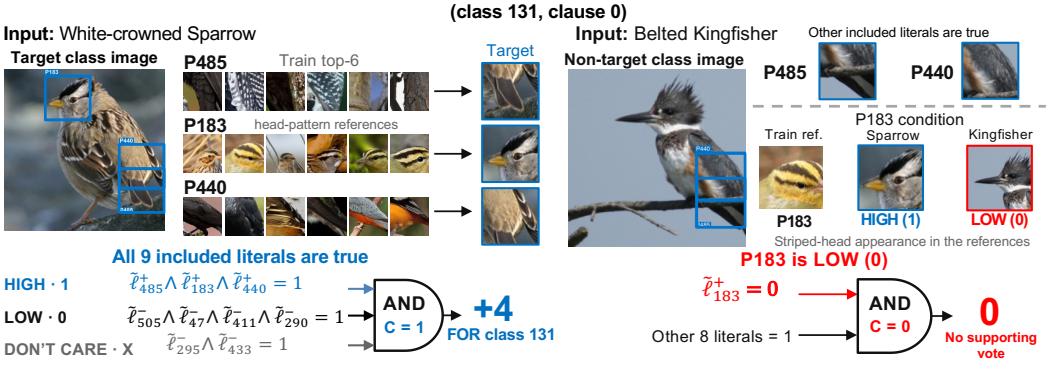


Figure 3: **One false literal blocks a conjunctive vote.** The shown clause contributes $+4$ to White-crowned Sparrow when all nine included literals are true (left), but zero when LOW predicate P183 falsifies its positive literal (right). Blue boxes mark channel peaks; training patches are visual references. Pk: predicate of channel k; C: deployed clause output.

For each channel, we assign a concept from its 20 highest-response training images using annotations from the frozen VLM InternVL3.5-8B (Wang et al., 2025). Among its 20 highest-response test images with an annotated point for that concept, channel peaks localize the concept more often than same-image spatial controls (Table 2). Among positive-vote firing events for correctly predicted classes, 33.8% of positive-literal conditions are VLM-confirmed, and all non-wildcard conditions are confirmed in 0.17% of eligible clause-firing events (Appendix E). Unannotated and unqueried conditions count as unconfirmed, so this comparison also depends on annotation coverage.

Votes from firing events containing at least one non-wildcard condition favor the predicted class over its original runner-up on 96.8% of correctly classified images (Appendix F).

We evaluate training-assigned attributes on test images using state-conditioned presence rates averaged over eligible channels, excluding annotations marked not visible. The training-assigned attributes are annotated as present more often when their predicates are in state 1 than in state 0 (39.6% versus 24.5%), and keeping each channel's training-selected part fixed gives an official-test part purity of 0.23 (Appendix E).

4.5 Classifier Adaptation after Deployment

We compare classifier adaptation on the same 512 Boolean inputs while holding the CNN and literal-interface thresholds fixed. Linear and MLP classifiers adapt using 32-bit floating-point (FP32) stochastic gradient descent (SGD), whereas TM classifiers use Type I/II feedback on 8-bit TA states and 3-bit vote magnitudes. TM-100 and TM-200 use 100 and 200 clauses per class, respectively. Labeled adaptation samples are disjoint from evaluation images. The adaptation experiments and FPGA design share the Type I/II settings ($T = 50$, $s = 5$); field-programmable gate array (FPGA) synthesis provides counts of digital signal processing (DSP) blocks, the dedicated hardware multipliers, and lookup tables (LUTs), and high-level synthesis (HLS) provides cycle estimates (Appendix D).

Table 3: **Classifier adaptation under corruption.** (a) Update arithmetic, DSP usage, HLS cycle estimates, and mutable state. (b) Accuracy before and after adaptation on shared frozen Boolean inputs (mean over three adaptation seeds).

(a) Arithmetic, latency estimates, and state

Classifier	Arith.	DSP↓	HLS best↓	HLS worst↓	State (KiB)↓
Linear	FP32	16	5,372	87,689	**20.0**
MLP	FP32	23	70,347	1,123,434	261.5
TM-100	int	**0**	**2,258**	**58,136**	500.4
TM-200	int	**0**	4,258	115,936	1,000.7

(b) Adaptation accuracy (%)

Classifier	MNIST-C	CIFAR-10-C
Linear	72.0→**90.0**	72.5→74.8
MLP	70.1→85.9	72.1→**76.4**
TM-100	72.2→82.9	69.6→73.4
TM-200	–	71.7→75.0

HLS values are cycle estimates, not measured latency (Appendix D). –: not evaluated. Bold: best per column.

The exported Boolean TM adapts to corrupted inputs without changing the CNN or literal thresholds. Type I/II feedback raises accuracy from 72.2% to 82.9% on MNIST-C and from 71.7% to 75.0% on CIFAR-10-C (Table 3), demonstrating adaptation within the discrete classifier. The linear classifier achieves the highest post-adaptation accuracy on MNIST-C, and the MLP on CIFAR-10-C.

FPGA synthesis on an AMD Zynq UltraScale+ ZU15EG shows that the TM's local Type I/II update datapath requires no DSP blocks, that is, no hardware multipliers. The TM implements these updates through Boolean logic and bounded integer state changes, avoiding gradients, floating-point arithmetic, and multiplication. Its higher state-storage and LUT requirements relative to the FP32 classifiers identify the corresponding hardware trade-off: multiplication-free adaptation with greater storage and logic requirements (Table 3; Appendix D).

Limitations. Accuracy after Boolean conversion and the memory requirements of classifier adaptation have not been evaluated at ImageNet scale. The hardware evidence consists of synthesis results and HLS estimates; on-board learning latency and energy have not been measured. The semantic consistency of the learned predicates remains partial.

5 Conclusion

ViLo-TM jointly learns CNN representations, predicate thresholds, and TM clauses, then exports a Boolean classifier whose literal conditions and signed integer votes remain linked to visual evidence. Its conjunctive structure makes explicit which combinations of learned conditions enable or block each clause vote. The deployed TM improves CIFAR-10 recognition over the compared vision-TM baselines (90.15% vs. 82.8%), and Boolean conversion retains most QAT accuracy (75.92% → 75.01% on CUB). With the encoder and thresholds fixed, the TM classifier adapts through standard TM feedback (Type I/II) without gradients, floating-point arithmetic, or multiplication.

AI USE STATEMENT

Generative AI tools assisted manuscript organization and language editing. A frozen InternVL3.5-8B vision-language model supplied the post-hoc visual annotations described in Section 4.4. The authors are responsible for the methods, experimental measurements, and manuscript.

ETHICS STATEMENT

The study uses public image-classification benchmarks and their part and attribute annotations. The visual-grounding evaluation depends on the coverage and potential biases of these benchmark annotations.

REPRODUCIBILITY STATEMENT

Sections 3.1–3.4 define the literal interface, joint objective, and discrete model. Appendices A–E provide training schedules, quantization conventions, adaptation settings, and visual-grounding protocols; Appendices F and G report the frozen-encoder, training-configuration, and clause-diversity ablations.

REFERENCES

K. Darshana Abeyrathna, Ole-Christoffer Granmo, and Morten Goodwin. Extending the Tsetlin Machine with integer-weighted clauses for increased interpretability. *IEEE Access*, 9:8233–8248, 2021. doi: 10.1109/ACCESS.2021.3049569.

Samy Badreddine, Artur d'Avila Garcez, Luciano Serafini, and Michael Spranger. Logic Tensor Networks. *Artificial Intelligence*, 303:103649, 2022. doi: 10.1016/j.artint.2021.103649.

Adrien Benamira, Thomas Peyrin, Trevor Yap, Tristan Guérand, and Bryan Hooi. Truth Table Net: Scalable, compact & verifiable neural networks with a dual convolutional small Boolean circuit networks form. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pp. 13–21, 2024. doi: 10.24963/ijcai.2024/2.

Chaofan Chen, Oscar Li, Chaofan Tao, Alina Jade Barnett, Jonathan Su, and Cynthia Rudin. This looks like that: Deep learning for interpretable image recognition. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Sondre Glimsdal and Ole-Christoffer Granmo. Coalesced multi-output Tsetlin Machines with clause sharing. *arXiv preprint arXiv:2108.07594*, 2021.

Ole-Christoffer Granmo. The Tsetlin Machine—a game theoretic bandit driven approach to optimal pattern recognition with propositional logic. *arXiv preprint arXiv:1804.01508*, 2018.

Ole-Christoffer Granmo. TMComposites: Plug-and-play collaboration between specialized Tsetlin Machines. *arXiv preprint arXiv:2309.04801*, 2023.

Ole-Christoffer Granmo, Sondre Glimsdal, Lei Jiao, Morten Goodwin, Christian W. Omlin, and Geir Thore Berge. The convolutional Tsetlin Machine. *arXiv preprint arXiv:1905.09688*, 2019.

Ole-Christoffer Granmo, Youmna Abdelwahab, Per-Arne Andersen, Karl Audun K. Borgersen, Paul F. A. Clarke, Kunal Dumbre, Ylva Grønningsæter, Vojtech Halenka, Runar Helin, Lei Jiao, Ahmed Khalid, Rebekka Omslandseter, Rupsa Saha, Mayur Shende, and Xuan Zhang. The Tsetlin Machine goes deep: Logical learning and reasoning with graphs. *arXiv preprint arXiv:2507.14874*, 2025.

Ylva Grønningsæter, Halvor S. Smørvik, and Ole-Christoffer Granmo. An optimized toolbox for advanced image processing with Tsetlin Machine Composites. In *2024 International Symposium on the Tsetlin Machine (ISTM)*, pp. 1–8, 2024. doi: 10.1109/ISTM62799.2024.10931429.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2019.

Artem Hnilov. Fuzzy-Pattern Tsetlin Machine. *arXiv preprint arXiv:2508.08350*, 2025.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2704–2713, 2018.

Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5338–5348, 2020.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. URL `https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf`.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

Gang Mao, Tousif Rahman, Sidharth Maheshwari, Bob Pattison, Zhuang Shao, Rishad Shafik, and Alex Yakovlev. Dynamic Tsetlin Machine accelerators for on-chip training using FPGAs. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 72(11):6962–6975, 2025. doi: 10.1109/TCSI.2025.3564875.

Norman Mu and Justin Gilmer. MNIST-C: A robustness benchmark for computer vision. *arXiv preprint arXiv:1906.02337*, 2019.

Meike Nauta and Christin Seifert. The Co-12 recipe for evaluating interpretable part-prototype image classifiers. In *Explainable Artificial Intelligence*, volume 1901 of *Communications in Computer and Information Science*, pp. 397–420. Springer, 2023. doi: 10.1007/978-3-031-44064-9_21.

Meike Nauta, Ron van Bree, and Christin Seifert. Neural prototype trees for interpretable fine-grained image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14933–14943, 2021.

Meike Nauta, Jörg Schlötterer, Maurice van Keulen, and Christin Seifert. PIP-Net: Patch-based intuitive prototypes for interpretable image classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2744–2753, 2023.

Ben Peters, Vlad Niculae, and André F. T. Martins. Sparse sequence-to-sequence models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 1504–1519, 2019. doi: 10.18653/v1/P19-1146.

Felix Petersen, Christian Borgelt, Hilde Kuehne, and Oliver Deussen. Deep differentiable logic gate networks. In *Advances in Neural Information Processing Systems*, volume 35, pp. 2006–2018, 2022. doi: 10.52202/068431-0146.

Felix Petersen, Hilde Kuehne, Christian Borgelt, Julian Welzel, and Stefano Ermon. Convolutional differentiable logic gate networks. In *Advances in Neural Information Processing Systems*, volume 37, pp. 121185–121203, 2024. doi: 10.52202/079017-3851.

Tousif Rahman, Gang Mao, Bob Pattison, Sidharth Maheshwari, Marcos Sartori, Adrian Wheeldon, Rishad Shafik, and Alex Yakovlev. Runtime tunable Tsetlin Machines for edge inference on eFPGAs. In *2025 IEEE Sensors Applications Symposium (SAS)*, pp. 1–6, 2025. doi: 10.1109/SAS65169.2025.11105163.

Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 618–626, 2017.

Jivitesh Sharma, Rohan Yadav, Ole-Christoffer Granmo, and Lei Jiao. Drop Clause: Enhancing performance, robustness and pattern recognition capabilities of the Tsetlin Machine. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(11):13547–13555, 2023. doi: 10.1609/aaai.v37i11.26588.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

Emile van Krieken, Erman Acar, and Frank van Harmelen. Analyzing differentiable fuzzy logic operators. *Artificial Intelligence*, 302:103602, 2022. doi: 10.1016/j.artint.2021.103602.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The Caltech-UCSD Birds-200-2011 dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011. URL `https://authors.library.caltech.edu/records/cvm3y-5hh21`.

Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, Zhaokai Wang, Zhe Chen, Hongjie Zhang, Ganlin Yang, Haomin Wang, Qi Wei, Jinhui Yin, Wenhao Li, Erfei Cui, Guanzhou Chen, Zichen Ding, Changyao Tian, Zhenyu Wu, Jingjing Xie, Zehao Li, Bowen Yang, Yuchen Duan, Xuehui Wang, Zhi Hou, Haoran Hao, Tianyi Zhang, Songze Li, Xiangyu Zhao, Haodong Duan, Nianchen Deng, Bin Fu, Yinan He, Yi Wang, Conghui He, Botian Shi, Junjun He, Yingtong Xiong, Han Lv, Lijun Wu, Wenqi Shao, Kaipeng Zhang, Huipeng Deng, Biqing Qi, Jiaye Ge, Qipeng Guo, Wenwei Zhang, Songyang Zhang, Maosong Cao, Junyao Lin, Kexian Tang, Jianfei Gao, Haian Huang, Yuzhe Gu, Chengqi Lyu, Huanze Tang, Rui Wang, Haijun Lv, Wanli Ouyang, Limin Wang, Min Dou, Xizhou Zhu, Tong Lu, Dahua Lin, Jifeng Dai, Weijie Su, Bowen Zhou, Kai Chen, Yu Qiao, Wenhai Wang, and Gen Luo. InternVL3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *arXiv preprint arXiv:2508.18265*, 2025.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

A Experimental Settings

Datasets and encoders. MNIST and Fashion-MNIST provide 60,000 official training and 10,000 test images; CIFAR-10 and CIFAR-100 provide 50,000 training and 10,000 test images. The CUB evaluation uses 4,794 model-training images, 1,200 internal-validation images, and 5,794 official-test images. Recognition accuracy is measured on each complete official-test split. The CUB configuration uses an encoder initializer trained only on the 4,794 model-training images and selected on internal validation. According to the run records, the MNIST, Fashion-MNIST, CIFAR-10, and CIFAR-100 initializers were selected by official-test accuracy.

MNIST, Fashion-MNIST, and CIFAR-10 use dataset-specific plain CNN initializers with 256 output channels. CIFAR-100 uses 512 channels. CUB uses the 512 native output channels of a VGG19-BN encoder (Simonyan & Zisserman, 2015), with a 14×14 grid for 224×224 inputs. The channel outputs pass directly to channel-wise softmax and spatial max pooling. Routing retains $K_{\mathrm{sel}} = 10$ candidates per clause.

Reported runs and controls. Table 1(a) summarizes seeds 0–2 for MNIST, Fashion-MNIST, CIFAR-10, and CIFAR-100 using population standard deviations. CUB reports three runs of the current configuration (seeds 0–2) with the validation-selected initializer: $75.92 \pm 0.40\%$ QAT and $75.01 \pm 0.59\%$ Boolean accuracy (seed 0: 76.41/75.63%; seed 1: 75.94/75.18%; seed 2: 75.42/74.21%). Seed 0 selects its checkpoint by internal-validation QAT accuracy over the 100 scheduled epochs; for seeds 1 and 2, the checkpoint is selected by the same criterion within the first 63 (seed 1) and 60 (seed 2) of 100 scheduled epochs. The CUB column of Table 1(b) averages the three conversions: -0.51 pp after global export routing and -0.91 pp cumulatively after exact Boolean conjunction, with no further change from integer votes and the deployed implementation. The matched-classifier, clause-composition, and margin analyses use the seed-0 model; the frozen-encoder control and the other CUB ablations use one run each.

Appendix F compares seed-0 joint training with a frozen-encoder TM that shares each dataset's initializer and holds CNN weights and batch-normalization statistics fixed while learning the literal interface and classifier. Table 4 gives a non-TM accuracy reference. For MNIST through CIFAR-100, the encoder and a linear classifier on pooled channels are trained jointly. For CUB, the entry is the validation-selected initializer itself, a floating-point CNN with a linear classifier on pooled channels.

Table 4: **Floating-point CNN–linear reference.** Official-test accuracy, seed 0. CUB: the validation-selected initializer with its own linear classifier; other datasets: encoder and linear head trained jointly.

Method	MNIST	FMNIST	C10	C100	CUB
Linear-head reference	99.58%	94.66%	92.15%	75.06%	78.65%

Matched backends on fixed literals. Table 5 keeps the CUB model's integer encoder, literal thresholds, and preprocessing fixed and replaces only the classifier. Linear and MLP heads are trained with cross-entropy and Adam on the 4,794 model-training images, with learning rate, weight decay, and early stopping selected on internal-validation accuracy (grid: learning rate $\{3 \cdot 10^{-4}, 10^{-3}, 3 \cdot 10^{-3}\}$, weight decay $\{0, 10^{-4}, 10^{-3}\}$), and are evaluated once on official test over three seeds. Re-extracting the test literals reproduces the exported TM's recognition result (4,382 of 5,794 correct predictions). All backends fit the model-training images perfectly (100% accuracy), while their held-out performance differs. On Boolean inputs, the first layer of the linear and MLP heads reduces to additions over the true literals (717 per image on average); the MLP's second layer requires multiplications. The TM thresholds were trained jointly with the TM, whereas the heads are fitted post hoc on fixed features.

Recorded training schedules. The three-run aggregates include different stopping epochs. MNIST, Fashion-MNIST, and CIFAR-10 seed 0 complete 100 epochs; seeds 1/2 stop at 91/75, 61/69, and 84/63 epochs, respectively. CIFAR-100 seeds 0/1/2 stop at epochs 89/58/56. The means and standard deviations summarize the recorded runs and their schedules.

Table 5: **Matched classifiers on the fixed CUB encoder and literals.** Official-test accuracy; heads: mean ± s.d. (population) over three seeds; r_k rows use the 512 pooled integer responses instead of literals. Params: included literals plus votes (TM) or weights and biases. Memory: sparse literal indices, clause lengths, and 18-bit votes (TM) or FP32 weights. Ops: included-literal AND evaluations and at most one integer addition per fired clause (TM; 12,636 fired on average) or dense MAC-equivalent counts (heads; on Boolean inputs the first layer needs only additions). Memory is sparse inference storage, not the dense adaptation state of Table 3.

Classifier	Input	Test (%)	Params	Memory	Ops per image
Boolean TM (export)	1,024 literals	75.63	182,873	289 KB	143k AND, ≤40k add
Linear	1,024 literals	$\mathbf{77.51 \pm 0.12}$	205,000	820 KB	205k MAC
MLP, H=128	1,024 literals	76.18 ± 0.20	157,000	628 KB	157k MAC
MLP, H=512	1,024 literals	76.88 ± 0.23	627,400	2,510 KB	627k MAC
Linear	512 r_k	76.15 ± 0.12	102,600	410 KB	102k MAC
MLP, H=512	512 r_k	74.17 ± 0.05	365,256	1,461 KB	365k MAC

B Losses and Training Schedule

PIP-Net representation objectives. The standard schedule uses the alignment and usage objectives of Nauta et al. (2023); the CUB configuration sets their weights to zero. Two augmented views share spatial geometry. Let $S^{(1)}, S^{(2)}$ denote their channel distributions, B the image count, U the number of matched spatial locations, and sg stop-gradient. The alignment objective is

$$\mathcal{L}_{\text{align}} = -\frac{1}{2BU} \sum_{n,u} \left[\log \sum_k S^{(1)}_{n,k,u} \operatorname{sg}(S^{(2)}_{n,k,u}) + \log \sum_k S^{(2)}_{n,k,u} \operatorname{sg}(S^{(1)}_{n,k,u}) \right]. \tag{14}$$

Writing $r^{(v)}_{n,k} = \max_u S^{(v)}_{n,k,u}$, usage is

$$\mathcal{L}_{\text{usage}} = -\frac{1}{2K} \sum_{v=1}^{2} \sum_{k=1}^{K} \log \left[\tanh \left(\sum_{n=1}^{B} r^{(v)}_{n,k} \right) + 10^{-8} \right]. \tag{15}$$

Alignment averages matched locations; usage pools channel responses across the image batch.

Routing regularizers. Load balancing uses the squared coefficient of variation of the expected selection frequencies across channels, averaged over classes. Under grouped routing, the higher preference of the channel's positive and negative literals defines each channel's load logit. For each augmented view n in the batch $\mathcal{B}$ of one backend call, let $t_{n,c,j}$ be the noise-perturbed Top-K_{sel} cutoff with gradients stopped. With N_C classes, standard normal CDF Φ (argument clipped to $[-50, 50]$), and channel load logit $\bar{h}_{c,j,k} = \max(g_{c,j,k}, g_{c,j,K+k})$,

$$p_{c,k} = \frac{1}{|\mathcal{B}|} \sum_{n \in \mathcal{B}} \sum_{j=1}^{M} \Phi\left(\frac{\bar{h}_{c,j,k} - t_{n,c,j}}{\max(\nu_{c,j}, 10^{-5})} \right), \qquad \mathcal{L}_{\text{load}} = \frac{1}{N_C} \sum_{c=1}^{N_C} \left(\frac{\operatorname{std}_k p_{c,k}}{\operatorname{mean}_k p_{c,k} + 10^{-8}} \right)^2, \tag{16}$$

where std_k is the sample standard deviation over channels.

Routing cosine is the default diversity objective. With clause preference vectors $g_{c,j} = (g_{c,j,1}, \ldots, g_{c,j,2K})$ normalized as $\bar{g}_{c,j} = g_{c,j} / \max(\|g_{c,j}\|_2, 10^{-12})$, the implemented normalization is

$$\mathcal{L}_{\text{div}} = \frac{1}{N_C M} \sum_{c=1}^{N_C} \sum_{j=1}^{M} \sum_{j' \neq j} \left(\bar{g}^{\mathsf{T}}_{c,j} \bar{g}_{c,j'} \right)^2. \tag{17}$$

The routing-cosine penalty compares clause preferences within each class. Joint learning uses $(\lambda_{\text{CE}}, \lambda_{\text{align}}, \lambda_{\text{usage}}, \lambda_{\text{load}}, \lambda_{\text{div}}) = (2, 5, 2, 0.001, 0.001)$ in the standard schedule. The CUB configuration uses $(2, 0, 0, 0.001, 0.001)$.

Threshold parameterization. Unconstrained parameters a_k, b_k define ordered thresholds with a margin $0 < \delta < 1/2$:

$$\theta^-_k = (1 - 2\delta)\sigma(a_k), \qquad \theta^+_k = \theta^-_k + \delta + (1 - \theta^-_k - \delta)\sigma(b_k). \tag{18}$$

The lower and upper thresholds are initialized from the 40th and 60th percentiles of training responses. Sharpness is calibrated per channel as three divided by the response standard deviation, clipped to $[10, 10000]$, and stored as a fixed buffer.

Optimization phases. The schedule used for MNIST, Fashion-MNIST, and CIFAR-10 has representation learning in epochs 1–10, literal-interface and TM warm-up in 11–13, partial joint learning in 14–20, full joint learning in 21–50, and QAT from epoch 51 within a 100-epoch run. The conversion records use $\beta = 20$ for all four smaller benchmarks.

The CUB configuration follows a 100-epoch schedule with $\beta = 10$, as do the CUB ablations in Appendix F (the smaller-benchmark controls there use $\beta = 20$), with CNN weights and batch-normalization statistics fixed through epoch 13. Alignment and usage weights are zero throughout training. Epochs 1–10 perform no parameter updates; thresholds are initialized from the pretrained encoder responses at epoch 11, and epochs 11–13 optimize thresholds and TM parameters. Joint learning begins at epoch 14, includes the full encoder from epoch 21, and enters QAT at epoch 51. Internal-validation QAT accuracy selects the checkpoint. Export uses $F = 16$ fixed-point votes and target mean clause length 3.5; the calibrated entmax parameter reaches $\alpha = 2$, with mean length 3.57.

Soft-conjunction implementation. The MNIST, Fashion-MNIST, CIFAR-10, and CIFAR-100 checkpoints apply a 10^{-6} lower bound to the partition sum inside the logarithm in Eq. 8. The bound is inactive at $\beta = 10$ and limits the clause response to $\log(10^6)/\beta$ for larger β. While the bound is active, the clause response is constant, so its derivative with respect to the literals is zero; the derivative in Section 3.2 applies otherwise. Stagewise evaluation retains the soft conjunction with the partition-sum lower bound through the relaxed-conjunction stages. The CUB configuration evaluates Eq. 8 directly in log space, preserving an all-true clause value of one.

C Quantization and TM Conversion

Integer encoder. The encoder folds batch normalization into each convolution. Folded weights use per-output-channel power-of-two scales and INT8 quantization; activations use per-layer power-of-two scales. Integer convolution accumulates products, then applies rounded shifts and ReLU clipping. QAT evaluates the integer convolution, rounded shifts, and ReLU clipping with straight-through backward operators. Normalized CUB inputs use a 2^{-5} grid and final logits a 2^{-6} grid.

Integer softmax and literal thresholds. For the integer channel logits $\widehat{A}_k(u)$ of the quantized encoder and differences $d_k(u) = \max_{k'} \widehat{A}_{k'}(u) - \widehat{A}_k(u)$, a lookup stores $\widehat{E}_k(u) = \mathrm{round}(2^{24} \exp(-d_k(u)/64))$ for differences from 0 to 1024; larger differences map to zero. Integer probability maps and pooled responses are

$$\widehat{S}_k(u) = \mathrm{round}\left(Q \frac{\widehat{E}_k(u)}{\sum_{k'} \widehat{E}_{k'}(u)} \right), \qquad \widehat{r}_k = \max_u \widehat{S}_k(u), \qquad Q = 65535. \tag{19}$$

Thresholds use $T_k^- = \lfloor Q\theta_k^- \rfloor$ and $T_k^+ = \lceil Q\theta_k^+ \rceil$, with $\delta = 1/Q$ in Eq. 18. The literal bits are

$$\widetilde{\ell}_k^+ = \mathbb{I}[\widehat{r}_k > T_k^-], \qquad \widetilde{\ell}_k^- = \mathbb{I}[\widehat{r}_k < T_k^+]. \tag{20}$$

QAT and deployment share the rounding and comparison conventions.

Exported supports and votes. Export disables routing noise. A global entmax parameter determines coefficients $\widehat{a}_{c,j,i}$, and inclusion is $m_{c,j,i}^{(0)} = \mu_{c,j,i} \mathbb{I}[\widehat{a}_{c,j,i} > 0]$, where μ is the auxiliary binary mask. Standard training fixes $\mu_{c,j,i} = 1$, yielding the inclusion rule $m_{c,j,i}^{(0)}$ in Eq. 12. Training and export use routing temperature $\tau_g = 0.1$. Global sparsity calibration uses routing parameters and a target mean inclusion. The four smaller benchmarks target mean inclusion 3.5; the CIFAR-100 seed-0 export reaches 3.6255 at the upper entmax bound.

Throughout integer conversion, round rounds to the nearest integer, with ties away from zero. Fixed-point exports use $\widehat{w}_{c,j} = \mathrm{round}(2^{16} w_{c,j})$. The CUB configuration uses the same $F = 16$ format. The adaptation experiments and Figure 3 use a global weight scale with three-bit magnitudes in $\{0, \ldots, 7\}$. Clause evaluation uses exact Boolean conjunction over included literals, with an empty

clause defined as false. An activated clause contributes its integer weight with the sign determined by its polarity.

Automaton initialization. For $N_{\mathrm{TA}} = 2R$ states indexed from 0 to $2R-1$, states below R exclude and states at or above R include. Normalize each clause's routing preferences to $\tilde{g}_{c,j,i} \in [0,1]$ using their minimum and maximum over all $2K$ literals, taking $\tilde{g} = 0$ when they are equal. Initialization is

$$A_{c,j,i}^{\mathrm{TA},(0)} = \mathrm{round}\left((R-1)\tilde{g}_{c,j,i}\right) + Rm_{c,j,i}^{(0)}, \qquad m_{c,j,i}^{(t)} = \mathbb{I}[A_{c,j,i}^{\mathrm{TA},(t)} \geq R]. \tag{21}$$

The adaptation branch uses $N_{\mathrm{TA}} = 256$ and bounded integer vote magnitudes. Feedback updates the states and votes while retaining the visual encoder and thresholds.

D Adaptation and Hardware Evaluation

All three classifiers use the same 512 Boolean literals from frozen exported MNIST and CIFAR-10 models. Each corruption starts from the clean classifier state. Linear and MLP use online cross-entropy SGD with learning rates 0.01 and 0.001; the MLP has dimensions 512–128–10 and updates both layers. TM uses Type I/II updates to eight-bit automaton states and three-bit clause votes. Feedback uses threshold $T = 50$ and specificity $s = 5$: each clause of the target class, and of one randomly drawn non-target class, receives feedback with probability $(T \mp v_c)/(2T)$ for the clipped class sum $v_c \in [-T, T]$, and Type I feedback draws a fixed mask of $\mathrm{round}(512/s) = 102$ literals. The MLP uses ReLU hidden units. The TM configurations use 100 clauses per class for MNIST-C and 200 for CIFAR-10-C. Before adaptation, the Linear and MLP heads are trained on frozen clean-training literals (AdamW; 30 epochs; learning rate 0.001; batch size 256), and the TM is initialized from the exported joint-training checkpoint. The comparison therefore fixes the adaptation inputs and data protocol, while the source classifiers retain their own initial-training procedures. Type I/II updates act on each literal's automaton independently and do not enforce the one-literal-per-channel restriction of grouped selection; after t updates, inference uses $\mathcal{S}_{c,j}^{(t)} = \{i : m_{c,j,i}^{(t)} = 1\}$ in place of $\mathcal{S}_{c,j}^{\mathrm{exp}}$.

MNIST-C uses 5,000 adaptation images and 10,000 disjoint evaluation images for each of 15 corruptions. CIFAR-10-C uses 1,000 adaptation and 9,000 evaluation base IDs across five severities, giving 5,000 and 45,000 samples per corruption across 19 corruptions. Adaptation and evaluation base IDs are disjoint. Seeds 0–2 set the adaptation sample order and, for TM, the feedback pseudo-random stream. All runs reuse the same pretrained encoder and initial classifier states, so their variation characterizes the adaptation procedure for a fixed source model. Metrics are macro-averaged across corruptions per seed, followed by the mean and sample standard deviation across seeds.

Table 6 gives the complete comparison. Resource counts are post-synthesis Vivado results for a Zynq UltraScale+ ZU15EG device (xczu15eg-ffvb1156-2-e) (LUT: lookup tables; FF: flip-flops; DSP: digital signal processing blocks), and clock periods are post-synthesis estimates at a 10 ns target rather than per-sample latency. Cycle counts are the HLS best- and worst-case latency estimates of each kernel; we do not identify them with separate inference-only and adaptation-enabled runs, which would require C/RTL co-simulation with the update disabled and enabled. State sizes are logical packed sizes of the mutable classifier parameters (floating-point weights for Linear and MLP; eight-bit automaton states, three-bit votes, and a 32-bit pseudo-random generator state for TM; fixed clause polarities are excluded), not measured on-chip memory. Including biases, the Linear and MLP heads contain 5,130 and 66,954 trainable FP32 scalars; TM-100 and TM-200 maintain 512,000 and 1,024,000 eight-bit automaton states, 1,000 and 2,000 three-bit votes, and a 32-bit PRNG state. For C classes, M clauses per class, and L input literals, this dense adaptation state requires $8CML + 3CM + 32$ bits, which grows linearly with the number of classes for fixed M and L. Linear and MLP keep FP32 weights and use a softmax and cross-entropy gradients during adaptation; TM increments or decrements integer automaton states and bounded votes without floating-point arithmetic or gradients. The adaptation experiments and the HLS kernel apply the same feedback rule; they differ only in whether empty clauses enter the class sum that sets the feedback probability. HLS C-simulation of the TM kernel matched the scalar reference model over eight consecutive updates.

Table 6: **Classifier adaptation on shared Boolean literals.** (a) Arithmetic and synthesis resources; TM entries a/b refer to MNIST-C/CIFAR-10-C with 100/200 clauses per class. (b) Macro accuracy (%) before→after adaptation, mean ± s.d. over three adaptation seeds. (c) HLS latency estimates in cycles per kernel call on 512 literals, not measurements; the TM worst case depends on a loop trip-count estimate and is not a guaranteed bound. Bold: best per column.

(a) Arithmetic and synthesis

Classifier	Update	FP32	FP multiply/ softmax/gradient	DSP ↓	Est. post-synth. clock period (ns) ↓	State (KiB) ↓	LUT ↓	FF ↓
Linear	SGD	yes	yes/yes/yes	16	7.632	**20.0**	**4,601**	**4,723**
MLP	Backpropagation	yes	yes/yes/yes	23	7.947	261.5	8,471	9,985
TM	Type I/II	no	no/no/no	**0**	**2.254/2.149**	500.4/1,000.7	39,518/39,747	18,251/18,295

(b) Accuracy before and after adaptation

Classifier	MNIST-C ↑	CIFAR-10-C ↑
Linear	$72.00 \rightarrow \mathbf{89.998} \pm 0.704$ (+18.00)	$72.50 \rightarrow 74.796 \pm 0.633$ (+2.30)
MLP	$70.05 \rightarrow 85.931 \pm 0.153$ (+15.88)	$72.15 \rightarrow \mathbf{76.368} \pm 0.001$ (+4.22)
TM	$72.17 \rightarrow 82.888 \pm 0.208$ (+10.72)	$71.69 \rightarrow 74.968 \pm 0.215$ (+3.28)

(c) HLS latency estimates (cycles)

Classifier	Best case ↓	Worst-case estimate ↓
Linear	5,372	87,689
MLP	70,347	1,123,434
TM, 100 clauses/class	**2,258**	**58,136**
TM, 200 clauses/class	4,258	115,936

E VISUAL-GROUNDING EVALUATION

The validation-selected checkpoint yields 75.63% official-test Boolean accuracy, with $F = 16$ votes and mean exported clause length 3.57. Model-training images determine channel concepts, anatomical parts, and attributes before official-test evaluation.

Train-assigned concept localization. Each of the 512 CUB channels ranks images by its QAT spatial-max response. Top-20 model-training retrievals assign a concept, and Top-20 official-test retrievals evaluate the fixed assignment. A frozen InternVL3.5-8B vision-language model (VLM) annotates each clean full image independently with concept names, color or pattern attributes, and point locations. It receives the image without the channel index or peak marker. The 8,632 unique-image annotation maps passed output-format checks. One full-image annotation is reused for all channels that retrieve that image.

For each channel, training retrievals determine the concept with the highest point-overlap rate in peak-centered 64×64 patches. The joint evaluation also assigns a color or pattern value. The concept and attribute assignments remain fixed during test evaluation, which includes 511 evaluable channels.

Denominators and controls. Concept-visible localization conditions on retrievals where the fixed concept has an annotated point. The joint visible row additionally requires an annotated attribute. The all-Top-20 rows retain the full retrieval denominator. A success requires the fixed concept point to lie within the peak-centered patch and, for the joint test, its attribute to match. The same-image control is the exact expected overlap when placing an equally sized patch at each of the 196 feature-grid centers. Localization is computed as a rate within each channel and then averaged across evaluable channels.

Localization controls. The bird-box control averages over grid centers inside the annotated CUB bird bounding box. The other-channel control evaluates the target channel's fixed concept label at the peaks of other channels in the same image. Table 7 reports the localization rates and paired differences, with 95% channel-bootstrap intervals; it uses the 508 channels whose test retrievals contain an annotated point for the assigned concept. Peak localization exceeds the controls based on image geometry, bird position, and other channel responses.

CUB part purity. Part purity follows the PIP-Net-style Top-10 evaluation with peak-centered 32×32 patches and visible CUB part points. Within each split, a channel's purity is the maximum,

Table 7: **Localization controls.** Mean concept localization over 508 channels with annotated test retrievals; brackets: 95% channel-bootstrap intervals.

Patch center	Localization (%) ↑	Peak minus control (pp)
Channel peak	**47.6**	–
Same-image grid	9.1	38.5 [36.0, 41.0]
Grid inside bird box	15.6	32.0 [29.4, 34.6]
Other-channel peaks	14.3	33.2 [30.6, 35.9]

Table 8: **CUB part purity.** Mean ± s.d. with Top-10 peak-centered 32×32 patches and visible part points. Reference rows are from PIP-Net (Nauta et al., 2023) (R/C: ResNet/ConvNeXt; sup./self-sup.: backbone pretraining, without part annotations). ViLo-TM rows use all 512 channels of VGG19-BN; train: 4,794-image model-training split; test: official test. "Training-selected part" keeps the part chosen on training retrievals fixed at test.

Model	Purity (train) ↑	Purity (test) ↑
ProtoPNet R	0.44 ± 0.21	0.46 ± 0.22
ProtoTree R	0.13 ± 0.14	0.14 ± 0.16
ProtoPShare R	0.43 ± 0.21	0.43 ± 0.22
ProtoPool R	0.35 ± 0.20	0.36 ± 0.21
PIP-Net R (sup.)	0.63 ± 0.25	0.65 ± 0.25
PIP-Net R (self-sup.)	0.29 ± 0.31	0.29 ± 0.32
PIP-Net C (sup.)	$\mathbf{0.92} \pm 0.16$	$\mathbf{0.93} \pm 0.15$
PIP-Net C (self-sup.)	0.61 ± 0.38	0.60 ± 0.38
ViLo-TM (VGG19-BN)	0.42 ± 0.14	0.39 ± 0.12
ViLo-TM, training-selected part	0.42 ± 0.14	0.23 ± 0.16

over parts, of the fraction of its Top-10 retrievals whose patch contains the part point, counting only retrievals in which that part is visible. Table 2(b) reports the mean and standard deviation across all 512 channels. Keeping each channel's training-selected part fixed gives official-test purity 0.23 ± 0.16, compared with 0.39 ± 0.12 when the part is reselected on test retrievals.

Training-fixed part and attribute assignments. With part assignments reselected within each split, ViLo-TM obtains similar purity on model training, internal validation, and official test (0.42 ± 0.14, 0.36 ± 0.11, and 0.39 ± 0.12, respectively). Fixing each channel's training-selected part gives an official-test purity of 0.23 ± 0.16 (quartiles 0.10, 0.20, 0.31). All 512 channels are evaluable, and the fixed part is visible in 4,705 of the 5,120 test retrievals (91.9%). The selected parts are belly (128 channels), wing (127), breast (79), back (76), leg (48), nape (26), throat (16), eye (4), beak (4), crown (3), and forehead (1).

We also fix one CUB attribute per channel on model-training images: among the 312 image-level attributes, the one whose presence rate in state-1 images most exceeds its rate over all training images. Presence requires an annotation with certainty "probably" or "definitely"; attributes marked not visible are excluded from the denominator rather than counted as absent. States come from the deployed integer front end. On training images, the selected attributes are annotated as present more often in images whose predicate is in state 1 than in the overall training population (state 1: 41.3%; overall: 31.3%). On official test, attribute presence increases from predicate state 0 to state 1 (state 0: 24.5%; state X: 29.8%; state 1: 39.6%), with a positive paired state-1 minus state-0 difference (15.0 pp; 95% channel-bootstrap interval 14.5–15.5; 511 channels). The ordering learned on training images therefore persists on test, although the mean annotation-presence rate in state 1 across eligible channels remains below 50%. No human evaluation was performed.

Clause visualization. Figure 3 traces the vote of class 131, clause 0. In the White-crowned Sparrow image, three included positive literals have predicates in state 1, four negative literals have predicates in state 0, and two negative literals are passed by predicates in state X. All nine literal bits are one, producing the clause's $+4$ vote. In the Belted Kingfisher image, P183 is in state 0, so its positive literal is zero, while the other included literals are one. The clause output is therefore zero. Pk denotes CNN channel k. Displayed clause and channel IDs are zero-based. Top-6 training

patches provide visual references for each displayed channel; the evaluated model determines the peak locations and predicate states.

Conditions in fired clauses. A firing event is a nonempty clause with $\widetilde{C}_{c,j}(x) = 1$ for a predicted, true, or runner-up class on one image. Repeated firings on different images count as separate events. In a firing event, a non-wildcard condition is an included positive literal whose predicate is HIGH (state 1) or an included negative literal whose predicate is LOW (state 0); a wildcard pass occurs when an included literal's predicate is in state X. The wildcard fraction is averaged over events. All-X vote share divides the sum of absolute vote magnitudes from entirely wildcard events by the corresponding sum over all events. Zero-weight clauses enter condition and length statistics but are excluded from vote-sign groups and contribute zero to vote magnitudes.

VLM agreement for positive-literal conditions requires the fixed channel concept to have an annotated point with certainty at least 3 inside its peak-centered box. VLM certainty uses a four-level scale from very uncertain to very clear. Agreement for negative-literal conditions requires an explicit absence response to a full-image query about the fixed concept. Up to five test images per channel and state are sampled with a fixed seed; queries provide the image and concept without the channel state or peak. Complete-clause agreement requires every non-wildcard condition to agree. Unannotated and unqueried conditions count as unconfirmed, and all-X events are excluded from the complete-agreement denominator. Table 9 reports agreement together with condition composition.

Positive-vote clauses of the predicted class contain more non-wildcard conditions and receive a smaller all-X vote share on correctly classified images than on incorrect predictions (non-wildcard conditions: 4.33 vs. 4.20; all-X vote share: 0.06% vs. 0.10%). Agreement is also higher for correct predictions, both for positive-literal conditions (33.8% vs. 29.9%) and for complete clauses (0.17% vs. 0.11%). Assessment of negative-literal conditions has limited query coverage (14.1% of these conditions), with explicit absence confirmed for a small fraction of queried conditions (1.7%). Complete-clause agreement therefore reflects both semantic correspondence and the coverage of condition annotations and queries.

Concept presence across predicate states. For each labeled channel, up to five official-test images per state are sampled with a fixed seed, and the VLM answers present, absent, or cannot determine for the fixed concept without seeing the channel, state, or peak. Averaged over channels, it answers present for 94.7%, 94.8%, and 92.5% of state-1, state-X, and state-0 images (512, 510, and 510 channels; 2,559, 2,547, and 2,550 judgments), and absent for 3.7%, 3.5%, and 4.9%. A LOW predicate (state 0) satisfies its negative literal; the VLM's frequent presence answers for these images, where most assigned concepts are bird parts visible in nearly every CUB image, indicate limited correspondence between a LOW predicate and the absence of the assigned concept.

Table 9: **Fired-clause conditions and VLM agreement.** CUB official test, with $F = 16$ fixed-point votes. Positive-vote clauses except runner-up (both signs). Non-wild.: included conditions whose predicates are in state 0 or 1 during the firing event; All-X: vote share of entirely wildcard events.

Fired clauses	Length	Non-wild.	Wildcard (%)	All-X vote (%)	Pos. literal agree (%)	Clause agree (%)
Predicted, correct	4.49	4.33	3.2	0.06	33.8	0.17
Predicted, incorrect	4.39	4.20	4.0	0.10	29.9	0.11
True class, incorrect	4.33	4.16	3.7	0.10	29.8	0.11
Runner-up	3.67	3.50	4.7	1.13	29.3	0.21

Spatial intervention protocol. For each image, we select the highest-weight positive-vote firing clause of the predicted class that contains an included positive literal whose predicate is HIGH; among those literals, we select the channel with the largest response. The target is the channel's peak-centered 64×64 box. We replace the target region with the ImageNet normalization mean, equivalently zero in normalized input, or apply Gaussian blur with standard deviation 8 pixels, and rerun the complete integer model. The control modifies a non-overlapping box of the same size whose fraction of area inside the bird bounding box most closely matches the target's.

The evidence ratio is $\widehat{r}_k(x')/\widehat{r}_k(x)$ for original image x and edited image x'. We measure departures of the selected predicate from state 1 and deactivation of the original target clause. Evidence ratios are summarized by their median. For the spatial-intervention evaluation on all 5,794 official-test

images, replacing the selected peak region with the normalization mean reduces the median evidence ratio to 0.64 (control: 1.01), moves the selected predicate out of the HIGH state in 27.3% of images (control: 3.2%), and switches off the target clause in 20.1% (control: 8.6%). Gaussian blur produces the same pattern (0.61 vs. 1.00; 29.3% vs. 3.4%; 20.1% vs. 8.4%; Table 10). Every target-minus-control difference has a 95% image-bootstrap interval above zero (clause switch-off: 10.3–12.6 and 10.5–12.9 pp).

Table 10: **Spatial interventions.** CUB, 5,794 official-test images. Mean: replacement with the normalization mean; controls match patch size and bird-box coverage. Bold: larger effect of each target–control pair.

Edit	Region	Median evidence ratio	Leaves HIGH (%)	Clause off (%)
Mean	Peak box	**0.64**	**27.3**	**20.1**
Mean	Matched control	1.01	3.2	8.6
Blur	Peak box	**0.61**	**29.3**	**20.1**
Blur	Matched control	1.00	3.4	8.4

Input perturbations. Under $\pm 10\%$ brightness changes and an 8-pixel shift, ViLo-TM keeps 89.2–90.0% of its predictions and 85.3–85.9% of predicate states, with a fired-clause Jaccard overlap of 0.80–0.81 for the predicted class (Table 11).

Table 11: **Input perturbations.** Agreement with the unperturbed input for ViLo-TM on 5,794 official-test images; peak agreement is computed over channels whose predicate is HIGH, and the Jaccard overlap over fired clauses of the originally predicted class.

Perturbation	Prediction agreement (%)	State agreement (%)	Peak agreement (%)	Fired-clause Jaccard
Brightness $\times 0.9$	89.2	85.3	83.7	0.80
Brightness $\times 1.1$	89.5	85.5	83.9	0.80
8-pixel shift	90.0	85.9	79.8	0.81

F FROZEN-ENCODER AND TRAINING-CONFIGURATION ABLATIONS

The frozen-encoder control holds CNN weights and batch-normalization statistics fixed while training the literal interface and TM; it shares each dataset's seed-0 initializer with the joint run. MNIST, Fashion-MNIST, and CIFAR-10 controls run the full 100-epoch schedule and report the final epoch. Table 12 reports Boolean official-test accuracy. With the standard schedule, joint training changes accuracy relative to the frozen encoder by -0.01 pp on MNIST, $+0.44$ pp on Fashion-MNIST, and -1.04 pp on CIFAR-10. With the CUB configuration, the CIFAR-10 checkpoint after epoch 55 of 100 reaches 91.64/91.65% (QAT/Boolean), $+0.61$ pp relative to the frozen encoder and $+1.65$ pp relative to the standard schedule. The CIFAR-100 frozen control did not complete before submission.

Table 12: **Frozen-encoder ablation.** Boolean official-test accuracy, seed 0; schedules in Appendix B. MNIST–CIFAR-10: 100th epoch; CUB: internal-validation selection. –: not run (the CIFAR-100 frozen control did not complete before submission). †Checkpoint after epoch 55 of 100; the run had not finished at submission. Bold: highest per dataset among completed runs.

Training	MNIST ↑	FMNIST ↑	C10 ↑	C100 ↑	CUB ↑
Joint, standard schedule	99.30%	**92.62%**	90.00%	69.79%	63.20%
Joint, CUB configuration	–	–	91.65%†	–	**75.63%**
Frozen encoder	**99.31%**	92.18%	**91.04%**	–	69.35%

CUB with validation-selected initialization. All CUB runs in this section use the validation-selected initializer and seed 0 unless stated. The standard schedule reaches 66.15/63.20% (QAT/Boolean); seeds 1 and 2 reach 65.93/62.60% and 66.76/63.79%, giving $66.28 \pm 0.35/63.20 \pm 0.49\%$ over seeds 0–2 (population standard deviation). These three-seed results evaluate the standard training schedule. The CUB configuration uses delayed encoder updates without the alignment and usage objectives; its three runs are summarized in Appendix A. The frozen encoder reaches 72.52/69.35%, and the initializer's own floating-point linear classifier 78.65%. The internal-validation gap is already present at epoch 13, before classification gradients update the encoder: epochs 1–10 adapt the upper CNN layers with alignment and usage losses only, and epochs 11–13 train thresholds and clauses with the CNN fixed. At epoch 13, internal-validation accuracy is 9.83% for the standard schedule and 74.17% for the frozen encoder.

Early encoder updates and representation objectives. The delayed-update variant keeps CNN weights and batch-normalization statistics fixed through epoch 13 and then follows the standard schedule; the CUB configuration additionally sets the alignment and usage weights to zero throughout. Both begin QAT at epoch 51 and select checkpoints by internal-validation QAT accuracy. The delayed-update variant reaches 72.70/71.57% and the CUB configuration 76.41/75.63% (seed 0 of the runs in Table 1). Before QAT (epoch 50), the CUB configuration and the frozen-encoder control reach 76.1% and 76.2% internal-validation accuracy; after the first QAT epoch (epoch 51), they reach 73.8% and 68.7%. At epoch 13, the delayed-update variant reaches 74.0% internal-validation accuracy versus 9.83% for the standard schedule; its validation accuracy peaks at 76.5% (epoch 35) and declines to 69.2% by epoch 100; the CUB configuration and the frozen control also decline after QAT onset (to 74.8% and 72.4%), so the decline is not attributable to the alignment and usage losses alone. Starting from the frozen encoder's QAT checkpoint (internal validation 72.83%), 20 further epochs with the CNN still fixed do not improve the selected checkpoint (72.52/69.35%), whereas unfreezing the CNN with the classification, load, and diversity losses gives 75.20/73.59%. Unfreezing with the alignment and usage losses lowers internal validation to 63–69%, and the selected checkpoint remains the starting point.

Clause structure of the CUB export. The CUB configuration also changes clause composition. It reaches the $\alpha_{\text{exp}} = 2$ bound with mean clause length 3.57. Across fired clauses of the predicted, true, and runner-up classes, it uses fewer non-wildcard conditions per firing event than the standard schedule (1.39 vs. 2.36) and assigns a larger share of vote magnitude to all-wildcard events (23.4% vs. 3.37%), while reducing near-duplicate clauses (7.91% → 0.53%) and removable literals (0.16% → 0.045%).

Margin decomposition. For each correctly classified image with a positive margin, we split the Boolean margin between the predicted class and its runner-up into contributions from firing events

with at least one non-wildcard condition and from all-wildcard events, recomputing clause outputs and scores from the exported literals and votes; the recomputation matches the explanation cache exactly. Non-wildcard contributions alone favor the predicted class over the original runner-up on 96.8% of the 4,382 correctly classified images. The median share of the margin contributed by all-wildcard events is 10.4%, and in 29.5% of images their net contribution favors the runner-up.

G CLAUSE-DIVERSITY ABLATION

Table 13 compares four training objectives: routing-cosine diversity, Jaccard set-overlap diversity, containment set-overlap diversity, and no clause-diversity loss. Routing cosine operates on clause preference vectors as defined in Appendix B. The set-overlap variants compare included-literal sets: Jaccard normalizes their intersection by the union size, whereas containment normalizes it by the smaller set size.

Table 13: **CUB clause-diversity ablation.** Standard schedule, validation-selected initializer, seed 0, target mean clause length 3.5, internal-validation checkpoint selection, before Type I/II adaptation. Bold: best per column (mean length is fixed by calibration).

Training objective	QAT (%) ↑	Boolean (%) ↑	Mean $\lvert\mathcal{S}^{\exp}\rvert$	Removable literals (%) ↓	Near-dup. clauses (%) ↓	All-X vote (%) ↓
Routing-cosine	**66.15**	63.20	3.50	**0.16**	7.91	3.37
Jaccard set-overlap	65.83	62.20	3.50	0.18	8.10	3.31
Containment	65.64	**63.29**	3.50	**0.16**	**7.78**	3.43
No diversity	64.96	61.75	3.50	0.17	8.46	**3.00**

All four objectives lie within 1.5 pp of one another in Boolean accuracy; omitting diversity gives the lowest accuracy and the most near-duplicate clauses. With one seed per objective, these differences do not establish a ranking. The raw Jaccard and containment terms remain near 10^{-5} and 10^{-4} during training, compared with about 0.2 for routing cosine, because training-time literal sets seldom reach the 0.8 overlap threshold.

Literal removal and near-duplicate clauses. For an included literal i of clause j, let $C_{j\setminus i}$ be the clause with that literal removed. The removal analysis measures

$$d_{j,i} = \frac{\sum_x C_{j\setminus i}(x)[1 - \widetilde{\ell}_i(x)]}{\sum_x C_{j\setminus i}(x)} \tag{22}$$

on model-training images. A literal is removable when $d_{j,i} = 0$, its removal preserves all internal-validation clause outputs, and the remaining conditions hold on at least 30 images in each split. Removals are accepted sequentially, with the criterion re-evaluated after each removal. The check preserves clause outputs on these two splits; it does not establish logical equivalence on other inputs. On the CUB export, 64 of 142,873 included literals (0.045%) are removable and only 32 are untested.

Two nonempty clauses of the same class and polarity form a near-duplicate pair when both the containment overlap of their included-literal sets and the Jaccard overlap of their training firing sets reach 0.8. Each clause must fire on at least 30 model-training images. The near-duplicate rate is the fraction of nonempty clauses belonging to at least one pair satisfying the overlap and firing-count criteria.

All-X votes. All-X vote share follows the definition under *Conditions in fired clauses* (Appendix E). The reported share weights firing events by their absolute integer vote magnitudes.