# Beyond FLOPs: Train-Full, Deploy-Partial Multi-Exit Inference via Selective Lightweight IC Ensemble

**Bitchan Eom**[1], **Eunchan Kim**[1,2*]
[1]Department of Artificial Intelligence, Hanyang University
[2]Department of Information Systems, Hanyang University
{ubc573, eckim}@hanyang.ac.kr

## Abstract

Early-exit networks promise inference savings by terminating computation at intermediate classifiers, but FLOPs-based gains need not translate into on-device latency reduction under static-graph compilation. On NVIDIA Jetson Orin Nano with TensorRT FP16, the per-sample exit policy of Shallow-Deep Networks (SDN) runs $2.1\times$ slower (p50) than vanilla ResNet-56 and misses the 60 fps deadline, because per-sample branching is executed as several separately compiled engines rather than one fused engine. We propose the Selective Lightweight IC Ensemble Network (SLIENet), a train-full, deploy-partial framework compiled as a single static FP16 engine. SLIENet trains the full backbone once with light self-distillation from the final classifier (a stop-gradient teacher); at deployment, the backbone is truncated and a sub-second search on a held-out calibration split selects internal classifiers (ICs) for softmax averaging. On CIFAR-100 across ResNet-56, VGG-16, and MobileNetV1, SLIENet outperforms SDN-based variants and remains competitive with ZTW, and Tiny-ImageNet-200 experiments (ResNet-56 at $32\times32$, ResNet-18 at $64\times64$) show the same trend. In a locked-clock, ten-run Jetson benchmark, the recommended SLIE5 configuration improves accuracy by $+1.2$ pp over vanilla ResNet-56 while reducing p50 latency by $30\%$ and energy by $40\%$, with zero deadline misses; ZTW is $0.44$ pp more accurate than SLIE6 but needs $2.6\times$ its p50 latency and $2.8\times$ its energy. A distilled truncated network matches SLIENet at a single depth; SLIENet's benefit is that one training run yields every depth budget as one deterministic engine. Our claims are scoped to static-compiled CNN deployment with TensorRT FP16 on Jetson-class hardware.

**Keywords.** Multi-exit networks; Self-distillation; Edge deployment; TensorRT; Selective ensemble; Deterministic inference.

**Code Availability.** Code for SLIENet training, SLIE selection, and TensorRT deployment (ResNet-56, CIFAR-100) is available at `https://github.com/Sudals/SLIENet`; its README lists the experiments covered.

## 1 Introduction

Deep learning vision models achieve high accuracy across many benchmarks, yet real-time deployment on resource-constrained edge devices remains difficult because edge inference must meet a strict latency deadline (60 fps or higher), a limited power budget, and the structural constraints of the deployment runtime simultaneously.

Early-exit networks attach internal classifier (IC) heads at intermediate layers of a shared backbone and terminate inference once a sufficiently confident prediction is made, reducing floating point

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

operations (FLOPs) substantially in theory. Shallow-Deep Networks (SDN) [1], Zero-Time Waste (ZTW) [2], and EENet [3] follow this approach and are mainly evaluated by FLOPs. However, *theoretical FLOPs reduction need not translate into real on-device latency reduction.* Edge runtimes such as TensorRT achieve layer fusion, kernel auto-tuning, and FP16 optimization by compiling a static computational graph. In our TensorRT deployment, a per-sample exit policy is realized as a sequence of separately compiled segment engines (one to seven invocations per sample), and the per-invocation launch, synchronization, and memory-copy overheads cancel the FLOPs savings. We verified this on NVIDIA Jetson Orin Nano in a locked-clock, ten-run benchmark: SDN with the per-IC exit policy reached a p50 latency of $4.95$ ms, $2.1\times$ that of plain inference ($2.41$ ms), and missed the 60 fps deadline on $22.3$ of every $10{,}000$ samples on average (Table 4). This motivates a design-level shift for static-compiled edge runtimes: the deployment unit should be a deterministic partial network rather than a sample-wise routing policy. We do not claim that compiler-side techniques (e.g., JIT/AoT compilation) cannot reduce this overhead. Two further observations shape our design: late exits can exhibit overthinking, with intermediate ICs outperforming the final classifier, and softmax-averaging a selected subset of ICs can outperform any single exit; hence the final head is better used as a training-time teacher than as the deployment endpoint.

We propose the Selective Lightweight IC Ensemble Network (**SLIENet**), which combines two strategies. *Train-full, deploy-partial* trains the full multi-exit network once, using the final classifier as a stop-gradient self-distillation teacher, and removes the late blocks at deployment; any depth budget can then be selected post hoc from the same checkpoint. *SLIE* selects an IC subset via a sub-second search on a held-out calibration split and aggregates the chosen ICs by softmax averaging. Both compile into a single TensorRT FP16 engine without dynamic branching. In the locked-clock benchmark, the recommended SLIE5 configuration improves accuracy over vanilla ResNet-56 by $+1.2$ pp ($69.81 \pm 0.18\%$ vs. $68.66 \pm 0.19\%$ over three seeds) while reducing p50 latency by $30\%$ ($1.69$ ms) and energy by $40\%$ ($7.81$ mJ), with zero 60 fps deadline misses in all ten runs. We also report the limits: a distilled truncated network matches SLIE5 at the same depth, the ensemble gain at IC5 is modest ($+0.32$ pp), and ZTW is more accurate when deployment cost is secondary.

**Contributions.** (i) **On-device deployment gap.** With repeated locked-clock measurements on Jetson Orin Nano, we show that under TensorRT the SDN per-IC policy is $2.1\times$ slower than vanilla inference and misses deadlines, and ZTW needs six engine invocations and $2.6\times$ the p50 latency of a comparably accurate SLIENet engine. (ii) **Train-full, deploy-partial.** One training run yields a checkpoint from which depth and IC subset are selected post hoc without per-budget retraining, each mode compiling into one static engine; we compare against matched truncated baselines (scratch and distilled). (iii) **SLIE.** A calibration-selected IC ensemble with no newly trained parameters; its gain comes from selectivity (naive all-IC averaging is $2.11$ pp below single IC5) and grows at deeper budgets ($+2.08$ pp at IC6), and a forward-greedy variant matches exhaustive search at polynomial cost.

## 2 Deployment-Aware Reframing of Multi-Exit Inference

### 2.1 Early Exit Networks and Overthinking

Static compression methods such as pruning [4, 5], quantization [6], and neural architecture search [7] often require architectural modification or retraining; they change the numerical precision or structure of a fixed network and are orthogonal to the depth and subset selection studied here (Section 4.5 shows that INT8 composes with SLIENet). Lightweight backbones such as MobileNets [8, 9] and EfficientNet [10] improve latency–accuracy trade-offs but still use a single fixed inference path. Ma et al. [11] further argued that FLOPs is an unreliable proxy for measured latency due to memory-access cost and platform characteristics; our results in Section 4.5 extend this observation to the dynamic branching regime under TensorRT compilation. Early-exit networks have emerged as an alternative, including BranchyNet [12], ACT [13], MSDNet [14], RANet [15], SkipNet [16], BlockDrop [17], and PABEE [18]. They share a confidence- or policy-based per-sample termination mechanism that conflicts with the single static graph assumed by static-compilation edge runtimes. The key insight of Kaya et al. [1] (SDN) is the *overthinking* phenomenon: for easy inputs, late layers may overfit to spurious features, thereby degrading accuracy. We extend this observation: overthinking is not merely a per-sample phenomenon; in our setting it also affects all inputs at the

architectural level, with intermediate ICs outperforming the final classifier on average. We exploit this property as the basis for our train-full, deploy-partial strategy.

### 2.2 Self-Distillation and Multi-Exit Training

Knowledge distillation (KD) [19–21] transfers the soft-label distribution of a teacher to a student, conveying the inter-class similarity information (*dark knowledge*) that hard labels cannot. Subsequent work added feature-level supervision via hint losses [22] and attention maps [23], and Born-Again Networks [24] showed that same-capacity self-distillation acts as a regularizer beyond compression. Zhang et al. [25] (BYOT) proposed a self-distillation framework in which deeper layers of the same network supervise shallower layers, and Li et al. [26] improved multi-exit network training via gradient equilibrium and one-for-all KD. SDN [1] jointly trains multiple ICs with the backbone using a weighted sum of cross-entropy losses, without a distillation term. SLIENet keeps this joint-training framework and adds a stop-gradient self-distillation term from the final head to each IC ($\alpha = 0.3$; Section 3.2), and it reverses the deployment assumption: instead of using ICs for dynamic exit, we treat the final classifier as a teacher used only at training time and discard it at inference. This preserves the dark knowledge effect of self-distillation while enabling compilation into a single static engine.

### 2.3 Selective Ensemble-Based Early Exit

ZTW [2] cascades all previous IC predictions at each exit point to close the accuracy gap of early exits. Our SLIE is inspired by ZTW but differs in three respects: (1) SLIE operates on a fixed subset selected by calibration-based search rather than a cascade of all previous exits; (2) IC-subset selection is performed automatically on a held-out calibration split; and (3) SLIE compiles into a single static engine without dynamic branching. On the same checkpoints, the selective ensemble adds $+0.32$ pp over single IC5 and $+2.08$ pp over single IC6, whereas naive averaging over all ICs is harmful (Table 3); the benefit therefore comes from selection rather than from ensembling per se.

### 2.4 Efficient Inference on Edge Hardware

TensorRT [27] compiles a static computational graph into optimized CUDA kernels, achieving layer fusion, kernel auto-tuning, and FP16 precision optimization in a single pass. Similar end-to-end deep-learning compilers, such as TVM [28], share the same static-graph philosophy. The Roofline model [29] explains the gap between theoretical compute and observed throughput through memory bandwidth and overheads that dynamic branching reintroduces. MLPerf Inference [30] and MLPerf Tiny [31] likewise report measured latency rather than FLOPs. When the network is split into separately invoked engines to support per-sample exits, each invocation incurs its own launch, synchronization, and memory-copy overhead, which can cancel the FLOPs savings. The mismatch between FLOPs and measured latency is widely observed on edge hardware [32, 11]. Prior work [33–35] studied early exit on mobile devices but did not directly measure the runtime overhead specific to TensorRT. JIT or ahead-of-time compilation of dynamic graphs could reduce part of this overhead; we did not evaluate such executors and treat them as an alternative route rather than a refuted one.

## 3 Proposed Method

### 3.1 Multi-Exit Backbone and Problem Setup

SLIENet is a deterministic inference framework based on the train-full, deploy-partial principle. We denote an input sample by $x$ and its label by $y$. The logits of head $j$ are $z_j(x)$, with the corresponding probability output $p_j(x) = \mathrm{softmax}(z_j(x))$. The final classifier (final head) output is denoted by $p_f(x)$. The depth of the backbone executed at deployment is $k$, and the IC subset selected by the calibration-based search is $S_k^* \subseteq \{1, \ldots, k\}$.

A dynamic exit determines the execution path per sample $x$ via a confidence threshold $\tau$:

$$y_{\mathrm{dynamic}}(x) = \arg\max_c \big[p_{j^*(x)}(x)\big]_c, \qquad j^*(x) = \min\big\{j : \mathrm{conf}(p_j(x)) \geq \tau\big\}, \tag{1}$$

where $\mathrm{conf}(p_j(x)) = \max_c [p_j(x)]_c$. Because $j^*(x)$ depends on $x$, a different computation path is executed per sample. In a static-compiled runtime, this requires the network to be split into separately invoked segment engines, each incurring its own launch and synchronization overhead. By contrast, the hard-cap configuration $M_k$ of SLIENet executes blocks $0 \to k$ for every sample regardless of confidence and produces the final prediction by averaging the softmax outputs of the ICs in $S_k^*$:

$$y_{\mathrm{SLIE}}(x) = \arg\max_c \frac{1}{|S_k^*|} \sum_{j \in S_k^*} \left[p_j(x)\right]_c, \qquad \forall x: \text{ blocks } 0 \to k \text{ always executed.} \tag{2}$$

The deployment configuration $M_k$ denotes the inference configuration compiled into a single TensorRT engine that contains the truncated backbone (blocks 0 to the $k$-th IC) together with the predetermined ensemble $S_k^*$. $S_k^*$ is determined in advance on $D_{\mathrm{cal}}$ and fixed at inference. No termination occurs before the $k$-th IC, even for confident intermediate predictions, so no dynamic branching is needed. We split the data as $D = D_{\mathrm{train}} \cup D_{\mathrm{cal}} \cup D_{\mathrm{test}}$ with the three parts pairwise disjoint: $D_{\mathrm{cal}}$ is held out from the training images and never used for training, and $D_{\mathrm{test}}$ is reserved for final evaluation. The search for $S_k^*$ uses only $D_{\mathrm{cal}}$; neither subset selection nor depth selection uses test feedback. **Hard-cap evaluation** reports per-depth accuracy of $M_k$ with $k$ fixed before inference; **deployment evaluation** adds measured p50 latency $L_{\mathrm{p50}}(M_k)$ and per-inference energy $E(M_k)$ on the device. FLOPs and MACs are reference metrics only.

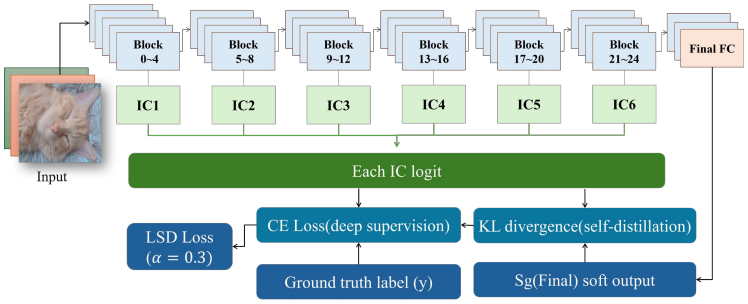



Figure 1: Training architecture of SLIENet on ResNet-56. The full network (blocks 0–26) is jointly trained with 6 IC heads and a final FC. The final FC functions as a self-distillation teacher $p_f(x)$ that conveys dark knowledge to each IC.

In Figure 1, each IC is a pooling layer followed by an FC layer that maps its block's feature map to a $C$-dimensional logit vector $z_j(x)$ ($C = 100$ for CIFAR-100); following SDN, the pooling is MixedPooling, which mixes average and max pooling over the spatial dimensions. In the calibration stage, $S_k^*$ is determined for each deployment depth $k$ on $D_{\mathrm{cal}}$ (2,000 samples; Section 3.3). In the deployment stage, only blocks up to the predetermined depth $k$ are compiled into a single TensorRT FP16 engine (partially fused deterministic inference); the late blocks (blocks $k{+}1$ through 26) and the final FC are removed. At inference, this engine is invoked once, and the final prediction is the softmax average over $S_k^*$. As there is no runtime decision, the execution path is identical for every sample; measured worst-case latency still depends on OS scheduling and thermal state, so our worst-case claims concern execution-path determinism only.

### 3.2 Light Self-Distillation for Independent Internal Classifiers

Standard SDN training minimizes a weighted sum of cross-entropy losses over all ICs (with the weights $w_j(t)$ defined below) and the final classifier. SLIENet modifies this such that the softmax output of the final classifier serves as a stop-gradient teacher signal. Let $\theta_b, \theta_1, \ldots, \theta_K, \theta_f$ denote the parameters of the shared backbone, IC heads, and final classifier, respectively. The propositions in this section are explanatory analyses of the mechanism rather than primary theoretical contributions; the empirical attribution of the gains is given in Section 4.3.

**Proposition 1** (Gradient isolation)**.** *The gradient of the KD loss term with respect to the final classifier parameter $\theta_f$ is zero (proof in Appendix B).*

Using the stop-gradient operator, the KD term does not update $\theta_f$: each IC is optimized to match the final head's prediction without producing any gradient feedback that alters the training trajectory of the final head. The shared backbone parameters $\theta_b$ are still influenced by IC losses through the

shared trunk; therefore, this does not imply a final-classifier accuracy equal to that of SDN. The two losses are

$$L_{\mathrm{SDN}} = \sum_j w_j(t)\,\mathrm{CE}(h_j(x), y) + \mathrm{CE}(h_{\mathrm{final}}(x), y), \tag{3}$$

$$L_{\mathrm{LSD}} = \sum_j w_j(t)\Big[(1-\alpha)\,\mathrm{CE}(h_j(x), y) + \alpha\, T^2\ \mathrm{KL}\big(\mathrm{sg}(\mathrm{softmax}(h_{\mathrm{final}}/T)) \,\big\|\, \mathrm{softmax}(h_j/T)\big)\Big] + \mathrm{CE}(h_{\mathrm{final}}(x), y), \tag{4}$$

where $\mathrm{sg}(\cdot)$ is the stop-gradient operation, $T = 4$ is the distillation temperature with the usual $T^2$ scaling [21], and the KL is taken from the detached teacher to the student. Following SDN [1], $w_j(t) = 0.01 + (t/E)(c_j - 0.01)$ ramps linearly over epochs $t \le E$ to the relative cost $c_j$ of IC $j$; the final cross-entropy is weighted by one. The KL-divergence term independently transfers dark knowledge [21] from the final head to each IC. SLIENet is *light* because only one logit-space soft-target KL term is added, with no feature-level distillation, bottleneck, or feature alignment, and no additional parameters.

### 3.3 SLIE: Selective Lightweight IC Ensemble

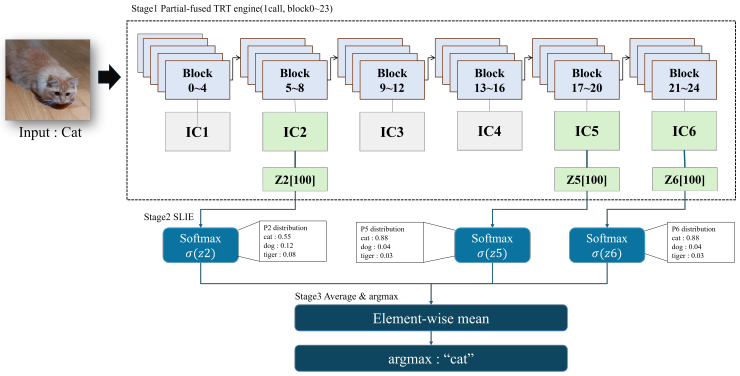



Figure 2: Inference architecture of SLIE. After a single forward pass through the truncated backbone, the softmax outputs of the predetermined IC subset $S_k^*$ are averaged to produce the final prediction. No runtime branching is involved.

The inference architecture is illustrated in Figure 2. A naive ensemble average over all ICs is a weak baseline because low-quality heads add noise without contributing to diversity (Table 3). SLIE selects the subset $S_k^*$ via an exhaustive search over $2^k - 1$ combinations on the calibration set; even for $k = 6$, this completes in less than 1 s.

We use a squared-risk analysis to clarify when selective averaging can help, while leaving final subset selection to calibration accuracy. The softmax output of IC head $i$ is denoted by $p_i(x) \in \Delta^{C-1}$, the one-hot vector of the true label by $q(x)$, and the error vector of head $j$ by $e_j(x) = p_j(x) - q(x)$. We define the residual cross-term as

$$Q_{ij} = \mathbb{E}_x\big[e_i(x)^\top e_j(x)\big]. \tag{5}$$

Notably, because the error mean is not subtracted, $Q_{ij}$ is a second moment, not a covariance. Specifically, $Q_{ii} = R_{\mathrm{sq}}(p_i) = \mathbb{E}_x[\|p_i(x) - q(x)\|^2]$ coincides with the squared risk of the individual head. The squared risk of the uniform-average ensemble over a subset $S$ decomposes exactly into a double sum, and the non-negativity of the variance, $\mathrm{Var}_{i\in S}(p_i(x) - q(x)) \ge 0$, yields an upper bound:

$$R_{\mathrm{sq}}\left(\frac{1}{|S|}\sum_{i\in S} p_i\right) = \frac{1}{|S|^2}\sum_{i,j\in S} Q_{ij} \ \le\ \frac{1}{|S|}\sum_{i\in S} R_{\mathrm{sq}}(p_i), \tag{6}$$

with equality if and only if all heads emit identical outputs. Hence, under the squared risk surrogate, if head diversity exists, the ensemble squared risk is strictly smaller than the average individual squared risk. This inequality is a statement regarding $L_2$ proximity; the transfer to 0–1 classification accuracy

is not guaranteed, therefore $S_k^*$ is selected based on the calibration accuracy rather than by squared risk.

**Proposition 2** (Cross-term-conditioned gain from adding a head, squared-risk surrogate)**.** *Consider adding a head $h \notin S$ to a non-empty subset $S$, with $n = |S|$. Let $Q_{hh} = \mathbb{E}[\|e_h(x)\|^2]$ be the individual risk of head $h$ and $Q_{hi} = \mathbb{E}[e_h(x)^\top e_i(x)]$ be its pairwise cross term with head $i \in S$. The necessary and sufficient condition for the ensemble squared risk to decrease upon adding head $h$ is*

$$Q_{hh} + 2\sum_{i \in S} Q_{hi} < (2n+1) \cdot R_{\mathrm{sq}}\left(\frac{1}{|S|}\sum_{i \in S} p_i\right). \tag{7}$$

*That is, even if the individual risk $Q_{hh}$ is large, the head can still reduce the ensemble risk provided that the sum of pairwise cross terms $\sum_{i \in S} Q_{hi}$ with the existing subset is sufficiently small. The derivation is given in Appendix C.*

**Interpretation.** Proposition 2 explains why SLIE can select lower-accuracy heads: under the squared-risk surrogate, head addition depends not only on individual risk but also on residual cross terms $Q_{hi}$ with the existing subset. Empirically, early heads (IC2, IC3) are repeatedly selected when their error patterns are decorrelated from late heads, consistent with Table 8 and Appendix F.

**Deployment cost and calibration.** The selected subset $S_k^*$ is fixed at deployment and acts as a hyperparameter rather than a per-sample decision. The IC outputs averaged by SLIE are computed within the single forward pass through the partial network; the only added work is evaluating the retained IC heads (e.g., $+0.051$M parameters and $\approx 0.1$M MACs for SLIE5 relative to single IC5), all of which already exist in the trained checkpoint, so no new parameters are trained. Calibration runs one forward pass over $D_{\mathrm{cal}}$ ($\approx$0.13–0.26 s for 2,000 samples) and scores subsets on the stored logits, under 1 s end to end and $O(K)$ fewer calibration passes than the threshold searches of SDN and ZTW. SLIE requires a modest labeled calibration set: with $N_{\mathrm{cal}} \geq 2{,}000$, $S_k^*$ and the ensemble gain are stable, whereas smaller calibration sets can make SLIE underperform the single IC (Appendix E); we therefore recommend $N_{\mathrm{cal}} \geq 2{,}000$. Because recalibration is sub-second, it can be repeated per deployment distribution given a small labeled target sample; we did not run a controlled distribution-shift experiment.

**Scaling beyond small $K$.** Exhaustive search over $2^k - 1$ subsets is trivial for $K = 6$ (63 subsets) but does not scale to many exits. A forward-greedy variant that adds one head at a time by calibration accuracy evaluates $O(K^2)$ subsets. On ResNet-56 / CIFAR-100 (five seeds $\times$ $k = 1, \ldots, 6$), it chose the same subset as exhaustive search in 28 of 30 cases while evaluating $\approx$18–20 instead of 63 subsets (mean deployed accuracy $60.96\%$ vs. $60.94\%$; per-case differences within $\pm 0.54$ pp), and on ResNet-18 / Tiny-ImageNet it matched the exhaustive optimum in all 12 cases. Validation at much larger $K$ remains future work.

### 3.4 Train-Full, Deploy-Partial

At training time, the full network is used, and the final classifier serves as the teacher. At deployment, the network is truncated at depth $k$: from block $4k+1$ to the last block, including the final FC, all components are removed to simultaneously avoid late-block overthinking and reduce computation. The operating modes $\mathrm{SLIE}_k$ correspond to different truncation depths. For ResNet-56, SLIE4 deploys up to IC4, SLIE5 up to IC5, and SLIE6 up to IC6, all with the FC removed. All modes are obtained post hoc from the same trained checkpoint, and each compiles into a single TensorRT FP16 engine. Thus, the final classifier is useful as a teacher during training but not necessarily as a deployment endpoint.

**Proposition 3** (Inherited cross term under a stylized residual-coupling model)**.** *Suppose the errors of cascade heads follow $e_j^{\mathrm{cas}} = r_j + B_j e_{j-1}^{\mathrm{cas}}$ with a fixed coupling matrix $B_j$, and let $M_{j-1} = \mathbb{E}[e_{j-1}^{\mathrm{cas}}(e_{j-1}^{\mathrm{cas}})^\top]$. (i) If $\mathrm{Cov}(r_j, e_{j-1}^{\mathrm{cas}}) = 0$, then $\mathrm{Cov}(e_j^{\mathrm{cas}}, e_{j-1}^{\mathrm{cas}}) = B_j \mathrm{Var}(e_{j-1}^{\mathrm{cas}})$. (ii) If $\mathbb{E}[r_j(e_{j-1}^{\mathrm{cas}})^\top] = 0$, then $Q_{j,j-1} = \mathrm{tr}(B_j M_{j-1})$, which is non-negative whenever $B_j + B_j^\top \succeq 0$. For fixed $Q_{hh}$ and $S$, a larger cross term makes condition* (7) *harder to satisfy (proof in Appendix D).*

Proposition 3 is explanatory and holds only within the stylized model; for general $B_j$ it guarantees neither a positive cross term nor a stricter condition. That selective averaging helps less on cascade

heads is an empirical observation (Table 1; Appendix D.3). Appendix D analyzes the parameter overhead, gradient coupling, and training–deployment mismatch of cascade heads.

# 4 Experimental Results

## 4.1 Experimental Setup

**Data and protocol.** We use CIFAR-100 [36] (50K training / 10K test images). Under our *strict protocol*, 2,000 calibration images are held out from the 50K training images, the network is trained on the remaining 48K, and the 10K test set is used only for final evaluation. All deployment-facing and component results (Tables 3 and 4) and all experiments added after the initial submission use this protocol. The per-IC comparison tables marked "submission protocol" are carried over from the submitted version, in which the calibration samples were drawn from the 50K training split; under the strict protocol, absolute accuracies are 0.3–1.1 pp lower (e.g., SLIENet at IC5: $70.59 \rightarrow 69.81$), while the selective-ensemble gain at IC6 is preserved (Appendix H). We additionally use Tiny-ImageNet-200 [37] (200 classes; 2K calibration images held out from the training set; the 10K validation set used as the test set).

**Models and evaluation.** The backbones were ResNet-56 [38], VGG-16-BN [39], and MobileNetV1 [8] on CIFAR-100, and ResNet-56 ($32\times32$ inputs) and ResNet-18 ($64\times64$, native resolution) on Tiny-ImageNet. All models used the MixedPool IC head of the original SDN paper. Workstation evaluation used PyTorch FP32 on an RTX 4090, with 5 seeds for the submission-protocol tables and 3 seeds for strict-protocol results (5 for the truncated baselines). Deployment evaluation used TensorRT FP16 engines compiled from the same checkpoints on a Jetson Orin Nano 8GB under a locked-clock protocol (TensorRT 10.3.0, nvpmodel 25 W, `jetson_clocks`, batch 1, ten runs of 10,000 samples); per checkpoint, the workstation-to-Jetson accuracy gap is at most 0.03 pp. Appendices L and I report training, engine-building, warm-up, timing, and power-measurement details.

## 4.2 Main Head-to-Head Results

Table 1 lists the per-IC hard-cap results on the workstation for ResNet-56, and Table 2 summarizes the final-exit accuracy across all three backbones. Both use the submission protocol (5 seeds); deployment claims rest only on the strict-protocol Tables 3 and 4. Full per-IC breakdowns for VGG-16 and MobileNetV1 are provided in Appendix G.

Table 1: ResNet-56 per-IC accuracy (%) on CIFAR-100 (submission protocol; see Section 4.1). PyTorch FP32, workstation, mean ± standard deviation over 5 seeds. SLIENet achieved the highest accuracy at IC5 and competitive late-exit accuracy at IC6 while remaining compatible with single-engine deploy-partial inference.

| Framework | IC1 | IC2 | IC3 | IC4 | IC5 | IC6 | Final |
|---|---|---|---|---|---|---|---|
| Backbone only | – | – | – | – | – | – | 70.08 ± 0.21 |
| SDN | 43.60 ± 0.34 | 48.70 ± 0.29 | 61.71 ± 0.24 | 64.29 ± 0.12 | 69.08 ± 0.18 | 70.09 ± 0.16 | 69.70 ± 0.17 |
| SDN+SLIE | 43.60 ± 0.32 | 48.06 ± 0.30 | 61.71 ± 0.22 | 64.29 ± 0.25 | 69.46 ± 0.11 | 71.04 ± 0.15 | 70.72 ± 0.16 |
| SDN+KD ($\alpha$=0.3) | 45.85 ± 0.38 | 51.34 ± 0.26 | 63.76 ± 0.13 | 66.04 ± 0.16 | 69.67 ± 0.13 | 69.42 ± 0.18 | 69.04 ± 0.18 |
| **SLIENet** | **45.85 ± 0.15** | **51.34 ± 0.16** | **63.76 ± 0.19** | 65.72 ± 0.22 | **70.59 ± 0.17** | 71.31 ± 0.15 | 71.31 ± 0.17 |
| ZTW Cascade | 47.15 ± 0.31 | 50.25 ± 0.28 | 59.14 ± 0.27 | 62.20 ± 0.25 | 68.35 ± 0.22 | 71.51 ± 0.47 | 72.26 ± 0.36 |
| ZTW+SLIE | 47.10 ± 0.31 | 50.67 ± 0.28 | 61.86 ± 0.35 | 63.38 ± 0.24 | 69.25 ± 0.20 | 70.15 ± 0.38 | 70.04 ± 0.19 |

Table 2: Final-exit accuracy (%) on CIFAR-100 across three backbones (submission protocol). PyTorch FP32, workstation, mean ± standard deviation over 5 seeds, reported as in Table 1. On VGG-16 and MobileNetV1, SLIENet has the numerically highest mean but is within one standard deviation of ZTW Cascade; on ResNet-56, ZTW Cascade is the most accurate.

| Backbone | SDN | SDN+SLIE | SDN+KD | **SLIENet** | ZTW Cascade | ZTW+SLIE |
|---|---|---|---|---|---|---|
| ResNet-56 | 69.70 ± 0.17 | 70.72 ± 0.16 | 69.04 ± 0.18 | 71.31 ± 0.17 | **72.26** ± 0.36 | 70.04 ± 0.19 |
| VGG-16 | 75.24 ± 0.51 | 76.45 ± 0.59 | 75.57 ± 0.49 | **76.71** ± 0.79 | 76.69 ± 0.39 | 76.51 ± 0.31 |
| MobileNetV1 | 70.72 ± 0.20 | 73.08 ± 0.34 | 70.95 ± 0.50 | **73.54** ± 0.25 | 73.41 ± 0.45 | 73.21 ± 0.47 |

On VGG-16 and MobileNetV1, SLIENet's final accuracy is comparable to ZTW Cascade (numerically highest, but $+0.02$ and $+0.13$ pp, within one standard deviation). On ResNet-56, ZTW Cascade is the most accurate at the final exit (72.26% vs. 71.31%); we do not frame this as a weakness of ZTW.

Its gains rely on cascade coupling, which in our TensorRT deployment requires one engine invocation per exit; Section 4.5 quantifies the resulting accuracy–deployability trade-off on the device.

### 4.3 Matched Static Baselines and Component Decomposition

Because a deployed SLIENet mode fixes the depth and IC subset for all inputs, it is effectively a static subnetwork, and the relevant baselines are static networks with the same deployment budget. Table 3 compares SLIENet against such baselines and decomposes its gain, all under the strict protocol.

Table 3: Matched static baselines and component decomposition on ResNet-56 / CIFAR-100 (strict protocol, mean ± std; 3 seeds, truncated baselines 5 seeds). "Subset" lists the averaged heads (seed-0 subset for SLIE modes). Truncated baselines are ResNet-56 networks cut at the IC5 depth and trained as single-head models, either from scratch with cross-entropy or distilled from a full ResNet-56 teacher. SLIE4–SLIE6 are all selected post hoc from the same SLIENet checkpoint. MACs/params: deployed engine (truncated backbone and averaged heads).

| Configuration | Subset | Accuracy (%) | MACs (M) | Params (M) |
|---|---|---|---|---|
| Vanilla ResNet-56 | final | $68.66 \pm 0.19$ | 125.8 | 0.859 |
| SLIENet, final head | final | $68.65 \pm 0.19$ | 125.8 | 0.862 |
| SDN ($\alpha$=0), single IC5 | IC5 | $68.44 \pm 0.25$ | 97.5 | 0.514 |
| Truncated at IC5, scratch (CE) | IC5 | $65.23 \pm 0.63$ | 97.5 | 0.514 |
| Truncated at IC5, distilled (ResNet-56 teacher) | IC5 | $\mathbf{70.15 \pm 0.40}$ | 97.5 | 0.514 |
| SLIENet, single IC4 | IC4 | $65.33 \pm 0.28$ | 79.7 | 0.238 |
| SLIENet, single IC5 | IC5 | $69.49 \pm 0.09$ | 97.5 | 0.514 |
| SLIENet, single IC6 | IC6 | $68.89 \pm 0.10$ | 116.4 | 0.810 |
| Naive all-IC average | IC1–IC5 | $67.38 \pm 0.19$ | 97.7 | 0.668 |
| SLIE4 | {IC3, IC4} | $65.33 \pm 0.28$ | 79.7 | 0.290 |
| **SLIE5 (recommended)** | {IC4, IC5} | $69.81 \pm 0.18$ | 97.6 | 0.565 |
| SLIE6 | {IC3–IC6} | $\mathbf{70.97 \pm 0.08}$ | 116.6 | 1.015 |

**Gain decomposition.** At the IC5 budget, the gain decomposes as SDN ($\alpha$=0) single IC5 $68.44 \xrightarrow{+1.05}$ SLIENet single IC5 $69.49 \xrightarrow{+0.32}$ SLIE5 $69.81$: the light self-distillation training package (stop-gradient teacher) contributes $+1.05$ pp, and the selective ensemble a modest $+0.32$ pp. Proposition 1 establishes gradient isolation but does not by itself attribute this empirical gain. Selectivity is essential: naive averaging over IC1–IC5 is $2.11$ pp below single IC5. The ensemble contribution grows at the deeper budget, where SLIE6 exceeds single IC6 by $+2.08$ pp. Overall, SLIE5 exceeds vanilla ResNet-56 by $+1.15$ pp at $0.78\times$ the MACs.

**Static baselines.** A truncated network distilled from a full ResNet-56 teacher ($70.15\%$) captures essentially all of the SLIE5–vanilla gap and slightly exceeds SLIE5, whereas the same network trained from scratch reaches only $65.23\%$, showing that teacher-based training matters at this depth. We therefore do not claim per-depth accuracy superiority over a dedicated student. The practical value of SLIENet is that one multi-exit training run yields every candidate budget (SLIE4–SLIE6 in Table 3) from a single checkpoint, each compiled as a single engine, whereas matching each budget with a dedicated student requires a separate training run and checkpoint.

### 4.4 Diversity and Subset Stability

Table 8 (Appendix F) lists the IC subsets selected in the multi-seed experiments. Two patterns were consistent. First, the IC at the truncation depth $k$ was included in every seed. Second, although the remaining ICs varied across seeds, early ICs with high error diversity (IC2, IC3) were repeatedly selected, consistent with Proposition 2: a low-accuracy head can still help through low error correlation with the others.

The error-correlation matrices (Figure 5 in Appendix F) show the same structure: in the IC6 row, $\rho_{\mathrm{soft}}$, $\rho_{\mathrm{err}}$, and $\mathrm{Cov}_{\mathrm{err}}$ are highest at IC5 ($0.921$, $0.755$, $0.159$) and lowest at IC1 ($0.604$, $0.374$, $0.086$), so shallower ICs make more independent errors. Applying SLIE on top of ZTW's cascade heads tended to yield diminished or negative late-IC gains, whereas applying SLIE on top of SDN+KD (i.e., SLIENet) yielded consistent positive gains, a pattern consistent with Proposition 3 (head-by-head breakdown in Appendix D). These correlation patterns are the only empirical support we offer for the stylized cascade model.

### 4.5 On-Device Deployment: Engine Calls, Latency, and Energy

Table 4 reports all main FP16 deployment configurations, including SDN dynamic exit and ZTW, measured in one locked-clock session.

Table 4: Jetson Orin Nano deployment benchmark (TensorRT FP16, ResNet-56 / CIFAR-100, strict protocol). Mean $\pm$ std over ten runs of 10,000 samples in one locked-clock session. Accuracy is that of the fixed seed-0 checkpoint deployed on the device (three-seed means are in Table 3). "Miss" counts samples per 10,000 exceeding the 16.67 ms (60 fps) deadline; "Calls" is the number of engine invocations per sample.

| Config (subset) | Acc (%) | p50 (ms) | p99 (ms) | max (ms) | Power (mW) | Energy (mJ) | Miss/10k | Calls |
|---|---|---|---|---|---|---|---|---|
| Vanilla | 68.78 | 2.41 $\pm$ 0.48 | 5.25 $\pm$ 3.04 | 14.67 $\pm$ 5.70 | 4,963 $\pm$ 652 | 13.02 $\pm$ 3.40 | 1.6 $\pm$ 4.7 | 1 |
| SDN dynamic | 68.89 | 4.95 $\pm$ 0.68 | 14.39 $\pm$ 1.22 | 30.19 $\pm$ 7.48 | 4,859 $\pm$ 565 | 27.09 $\pm$ 1.84 | 22.3 $\pm$ 22.6 | 1–7 |
| ZTW | **71.35** | 4.90 $\pm$ 0.54 | 10.47 $\pm$ 0.27 | 24.45 $\pm$ 5.80 | 6,236 $\pm$ 615 | 32.57 $\pm$ 1.29 | 4.0 $\pm$ 4.4 | 6 |
| Single IC5 | 69.50 | **1.60 $\pm$ 0.02** | **2.08 $\pm$ 0.08** | 8.70 $\pm$ 3.50 | 4,252 $\pm$ 17 | **7.04 $\pm$ 0.03** | 0.0 $\pm$ 0.0 | 1 |
| **SLIE5** {IC4, IC5} | 70.01 | 1.69 $\pm$ 0.21 | 2.64 $\pm$ 0.70 | **7.57 $\pm$ 3.27** | 4,426 $\pm$ 412 | 7.81 $\pm$ 0.20 | 0.0 $\pm$ 0.0 | 1 |
| SLIE6 {IC3–IC6} | 70.91 | 1.90 $\pm$ 0.01 | 4.95 $\pm$ 0.64 | 9.94 $\pm$ 1.76 | 5,346 $\pm$ 51 | 11.56 $\pm$ 0.19 | 0.0 $\pm$ 0.0 | 1 |

Table 4 establishes three findings. **(i) SDN dynamic exit does not deliver a deployment-improving trade-off in this regime**: on this checkpoint it gains only $+0.11$ pp over vanilla at $2.1\times$ the p50 latency, $2.7\times$ the p99 latency, and $2.1\times$ the energy, with a maximum latency of 30.19 ms and 22.3 deadline misses per 10,000 samples on average. **(ii) ZTW is the most accurate configuration but the most expensive to deploy**: compared with SLIE6, it is 0.44 pp more accurate (71.35% vs. 70.91%) but needs six engine invocations instead of one, $2.6\times$ the p50 latency, and $2.8\times$ the energy, and it misses the deadline on 4.0 samples per 10,000. We did not build a custom monolithic ZTW graph and do not claim that its fusion is fundamentally impossible. **(iii) All SLIENet modes run as one engine call with zero deadline misses in every run**: the recommended SLIE5 reduces p50 latency by 30% and energy by 40% versus vanilla while improving accuracy, and roughly halves the maximum latency (7.57 vs. 14.67 ms). Single IC5 is the fastest configuration; selective averaging in SLIE5 adds $+0.51$ pp on this checkpoint ($+0.32$ pp as a three-seed mean) for $+0.09$ ms and $+0.77$ mJ on the same fused engine. Average power of the SLIE modes (4.25–5.35 W) is close to that of vanilla (4.96 W), so the energy savings come primarily from shorter latency. Worst-case latency claims are restricted to execution-path determinism, not invariance to OS or thermal effects.

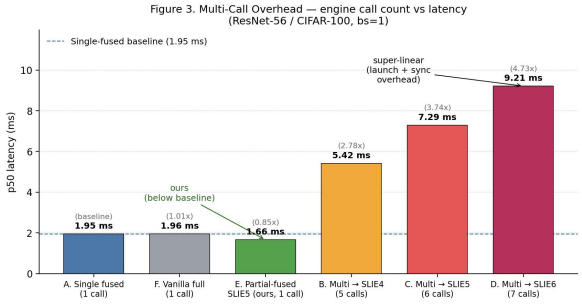



Figure 3: Engine-call count vs latency (separate, earlier single-run session; absolute values therefore differ from Table 4). Partial-fused SLIE5 (1 call) is the only configuration that goes below the single-fused baseline. Multi-engine variants of the same SLIE5 configuration (6 calls) inflate latency sharply with the number of engine calls, showing that FLOPs-based metrics can be misleading in real deployment.

**Engine-call overhead.** Figure 3 isolates the effect of the engine-call count on the same computation. Compared with the single-fused baseline (1.95 ms), the partial-fused SLIE5 (1 call) achieved 1.66 ms ($0.85\times$), whereas the same SLIE5 computation executed as multiple engine calls (6 calls) took 7.29 ms ($3.74\times$), and SLIE6 (7 calls) took 9.21 ms ($4.73\times$). Latency growth is dominated by per-call launch and synchronization overhead rather than by computation. The same effect appears in a second graph-execution environment: on ONNX Runtime (CPU execution provider), splitting the same nominal SLIE5 computation into five session invocations raises p50 latency from 0.502 to 1.805 ms (Appendix J.1). This does not establish behavior under JIT/AoT dynamic executors, which we did not measure.

**Composability with quantization.** INT8 quantization changes numerical precision, whereas SLIENet changes graph depth, so the two compose. In a separate within-session study (Appendix J.2), FP16→INT8 conversion changed accuracy by at most $0.10$ pp for both vanilla and an IC6-budget SLIE engine, and SLIE (IC6 budget) with INT8 reduced p50 latency by $33\%$ and energy by $30\%$ relative to vanilla FP16.

### 4.6 Beyond CIFAR-100

**Tiny-ImageNet-200, ResNet-56 at $32\times32$.** Over three seeds, SLIENet leads ZTW at the mid exits (IC3: $42.51$ vs. $40.71$; IC4: $45.32$ vs. $42.23$; IC5: $51.82$ vs. $49.80$) and ties it at IC6 ($53.48 \pm 0.20$ vs. $53.47 \pm 0.20$) while remaining single-engine; ZTW leads only at its full-cost final exit ($53.80 \pm 0.23$).

**Tiny-ImageNet-200, ResNet-18 at $64\times64$.** With $17\times$ more parameters ($14.45$M) and $4\times$ the pixels, trained with the same recipe: at the main budget (IC4), SLIE improves over the single IC by $+1.05$ pp ($67.35 \pm 0.45$ vs. $66.30 \pm 0.21$; final FC $66.37 \pm 0.21$), whereas earlier exits show little or no gain, reproducing the pattern observed on CIFAR-100. Full tables are in Appendix K.

## 5 Discussion and Limitations

**Scope.** Our deployment claims are specific to static-compiled CNN deployment with TensorRT FP16 on Jetson-class hardware, where a single fixed graph is the principal optimization unit. All results come from CIFAR-100 and Tiny-ImageNet-200 with CNN backbones on one device; we did not evaluate ImageNet-scale datasets, transformer backbones (e.g., ViT, DeiT), more capable accelerators, coarse-grained branching, or substantially larger numbers of exits, and we do not claim that the overthinking effect or the deployment advantage persists in those settings.

**When dynamic exit may remain preferable.** Dynamic early exit can still be effective when (i) the runtime supports conditional branching natively (e.g., custom FPGAs or dynamic-graph executors) or a JIT/AoT compiler specializes the dynamic graph; we observed repeated-dispatch overhead in both execution environments we tested (TensorRT and ONNX Runtime CPU) but did not evaluate JIT/AoT executors; (ii) the input-difficulty distribution is bimodal so that many samples exit very early; or (iii) the system is throughput- or bandwidth-bound rather than latency-bound.

**Accuracy–deployability trade-off.** We recommend ZTW when final accuracy is paramount and the runtime executes cascades efficiently, and SLIENet when single-engine static compilation, predictable per-sample latency, and energy under a frame deadline are the priority, as in Jetson/TensorRT edge deployment. Quantization is orthogonal and can be combined with SLIENet.

**Calibration, search, theory, and adaptation.** SLIE requires a modest labeled calibration set ($N_{\mathrm{cal}} \geq 2{,}000$); its sensitivity to calibration–test distribution shift is unquantified, and for many exits greedy (or beam) selection should replace exhaustive search. Propositions 1–3 are explanatory; Proposition 3 rests on a stylized coupling model that we did not fit to trained heads. The SLIE modes are chosen statically; since all come from one checkpoint, online mode switching driven by DVFS or thermal signals is a natural extension.

## 6 Conclusion

We proposed SLIENet, a train-full, deploy-partial framework combining light self-distillation, calibration-based SLIE subset selection, and single-engine deployment. On Jetson Orin Nano with TensorRT FP16, SLIE5 improved accuracy over vanilla ResNet-56 by $+1.2$ pp while reducing p50 latency by $30\%$ and energy by $40\%$, with deterministic execution paths and zero deadline misses. Its value lies in providing every operating budget from one training run as one deterministic engine, rather than in per-depth accuracy over a distilled student or in raw accuracy over ZTW. Within the tested regime (TensorRT FP16, CNNs, Jetson-class hardware), a multi-exit network is best deployed as a pre-compiled deterministic inference plan rather than a sample-wise routing policy.

## Acknowledgments and Disclosure of Funding

We thank the anonymous reviewers and the area chair for their constructive feedback, which substantially improved this paper. This research was supported in part by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (Artificial Intelligence Graduate School Program (Hanyang University), RS-2020-II201373), in part by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2026-25523610), in part by the Commercialization Promotion Agency for R&D Outcomes (COMPA) funded by the Ministry of Science and ICT (MSIT) (No. RS-2026-2710107921), and in part by the ANCHOR through the Seoul ANCHOR Center, funded by the Ministry of Education (MOE) and the Seoul Metropolitan Government (2026-ANCHOR-01-027-01). We also gratefully acknowledge Professor Sang-Wook Kim, Director of the Hanyang Institute of Advanced Artificial Intelligence, for his support in providing a conducive research environment.

**Competing interests.** The authors declare no competing interests.

## References

[1] Kaya, Y., Hong, S., & Dumitras, T. (2019). Shallow-Deep Networks: Understanding and Mitigating Network Overthinking. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, PMLR 97:3301–3310. URL: https://proceedings.mlr.press/v97/kaya19a.html

[2] Wołczyk, M., Wójcik, B., Bałazy, K., Podolak, I. T., Tabor, J., Śmieja, M., & Trzciński, T. (2021). Zero Time Waste: Recycling Predictions in Early Exit Neural Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 34, 2516–2528. URL: https://papers.nips.cc/paper/2021/hash/149ef6419512be56a93169cd5e6fa8fd-Abstract.html

[3] Ilhan, F., Chow, K.-H., Hu, S., Huang, T., Tekin, S., Wei, W., Wu, Y., Lee, M., Kompella, R., Latapie, H., Liu, G., & Liu, L. (2024). Adaptive Deep Neural Network Inference Optimization with EENet. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 1373–1382. DOI: https://doi.org/10.1109/WACV57701.2024.00141

[4] Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning Both Weights and Connections for Efficient Neural Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 28, 1135–1143. URL: https://papers.nips.cc/paper/5784-learning-both-weights-and-connections-for-efficient-neural-network

[5] Han, S., Mao, H., & Dally, W. J. (2016). Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. In *International Conference on Learning Representations (ICLR)*.

[6] Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., & Kalenichenko, D. (2018). Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2704–2713. DOI: https://doi.org/10.1109/CVPR.2018.00286

[7] Zoph, B., & Le, Q. V. (2017). Neural Architecture Search with Reinforcement Learning. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*. URL: https://openreview.net/forum?id=r1Ue8Hcxg

[8] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv preprint arXiv:1704.04861*. URL: https://arxiv.org/abs/1704.04861

[9] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 4510–4520. DOI: https://doi.org/10.1109/CVPR.2018.00474

[10] Tan, M., & Le, Q. V. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, PMLR 97:6105–6114. URL: https://proceedings.mlr.press/v97/tan19a.html

[11] Ma, N., Zhang, X., Zheng, H.-T., & Sun, J. (2018). ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 122–138. DOI: https://doi.org/10.1007/978-3-030-01264-9_8

[12] Teerapittayanon, S., McDanel, B., & Kung, H. T. (2016). BranchyNet: Fast Inference via Early Exiting from Deep Neural Networks. In *Proceedings of the 23rd International Conference on Pattern Recognition (ICPR)*, 2464–2469. DOI: https://doi.org/10.1109/ICPR.2016.7900006

[13] Graves, A. (2016). Adaptive Computation Time for Recurrent Neural Networks. *arXiv preprint arXiv:1603.08983*. URL: https://arxiv.org/abs/1603.08983

[14] Huang, G., Chen, D., Li, T., Wu, F., van der Maaten, L., & Weinberger, K. Q. (2018). Multi-Scale Dense Networks for Resource Efficient Image Classification. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*. URL: https://openreview.net/forum?id=Hk2aImxAb

[15] Yang, L., Han, Y., Chen, X., Song, S., Dai, J., & Huang, G. (2020). Resolution Adaptive Networks for Efficient Inference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2369–2378. DOI: https://doi.org/10.1109/CVPR42600.2020.00244

[16] Wang, X., Yu, F., Dou, Z.-Y., Darrell, T., & Gonzalez, J. E. (2018). SkipNet: Learning Dynamic Routing in Convolutional Networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 420–436. DOI: https://doi.org/10.1007/978-3-030-01261-8_25

[17] Wu, Z., Nagarajan, T., Kumar, A., Rennie, S., Davis, L. S., Grauman, K., & Feris, R. (2018). BlockDrop: Dynamic Inference Paths in Residual Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 8817–8826. DOI: https://doi.org/10.1109/CVPR.2018.00919

[18] Zhou, W., Xu, C., Ge, T., McAuley, J., Xu, K., & Wei, F. (2020). BERT Loses Patience: Fast and Robust Inference with Early Exit. In *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 18330–18341. URL: https://papers.nips.cc/paper/2020/hash/d4dd111a4fd973394238aca5c05bebe3-Abstract.html

[19] Buciluǎ, C., Caruana, R., & Niculescu-Mizil, A. (2006). Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 535–541. DOI: https://doi.org/10.1145/1150402.1150464

[20] Ba, J., & Caruana, R. (2014). Do Deep Nets Really Need to be Deep? In *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 27, 2654–2662.

[21] Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the Knowledge in a Neural Network. In *NIPS Deep Learning and Representation Learning Workshop*. URL: https://arxiv.org/abs/1503.02531

[22] Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., & Bengio, Y. (2015). FitNets: Hints for Thin Deep Nets. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*. URL: https://arxiv.org/abs/1412.6550

[23] Zagoruyko, S., & Komodakis, N. (2017). Paying More Attention to Attention: Improving the Performance of Convolutional Neural Networks via Attention Transfer. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*. URL: https://openreview.net/forum?id=Sks9_ajex

[24] Furlanello, T., Lipton, Z., Tschannen, M., Itti, L., & Anandkumar, A. (2018). Born Again Neural Networks. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, PMLR 80:1607–1616. URL: https://proceedings.mlr.press/v80/furlanello18a.html

[25] Zhang, L., Song, J., Gao, A., Chen, J., Bao, C., & Ma, K. (2019). Be Your Own Teacher: Improve the Performance of Convolutional Neural Networks via Self Distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 3712–3721. DOI: https://doi.org/10.1109/ICCV.2019.00381

[26] Li, H., Zhang, H., Qi, X., Yang, R., & Huang, G. (2019). Improved Techniques for Training Adaptive Deep Networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 1891–1900. DOI: https://doi.org/10.1109/ICCV.2019.00198

[27] NVIDIA Corporation. (2024). TensorRT Developer Guide. URL: https://docs.nvidia.com/deeplearning/tensorrt/developer-guide/

[28] Chen, T., Moreau, T., Jiang, Z., Zheng, L., Yan, E., Cowan, M., Shen, H., Wang, L., Hu, Y., Ceze, L., Guestrin, C., & Krishnamurthy, A. (2018). TVM: An Automated End-to-End Optimizing Compiler for Deep Learning. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, 578–594. URL: https://www.usenix.org/conference/osdi18/presentation/chen

[29] Williams, S., Waterman, A., & Patterson, D. (2009). Roofline: An Insightful Visual Performance Model for Multicore Architectures. *Communications of the ACM*, 52(4), 65–76. DOI: https://doi.org/10.1145/1498765.1498785

[30] Reddi, V. J., Cheng, C., Kanter, D., Mattson, P., Schmuelling, G., Wu, C.-J., et al. (2020). MLPerf Inference Benchmark. In *Proceedings of the 47th Annual International Symposium on Computer Architecture (ISCA)*, 446–459. DOI: https://doi.org/10.1109/ISCA45697.2020.00045

[31] Banbury, C., Reddi, V. J., Lott, P., Thakker, U., Jeffries, N., Sievert, C., Tommasi, C., Mazumder, S., Sabne, A., Vikram, S., et al. (2021). MLPerf Tiny Benchmark. In *Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS) Track on Datasets and Benchmarks*. URL: https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/da4fb5c6e93e74d3df8527599fa62642-Abstract-round1.html

[32] Dong, P., Wang, S., Niu, W., Zhang, C., Lin, S., Li, Z., Gong, Y., Ren, B., Lin, X., & Tao, D. (2020). RTMobile: Beyond Real-Time Mobile Acceleration of RNNs for Speech Recognition. In *Proceedings of the 57th ACM/IEEE Design Automation Conference (DAC)*, 1–6. DOI: https://doi.org/10.1109/DAC18072.2020.9218499

[33] Matsubara, Y., Levorato, M., & Restuccia, F. (2022). Split Computing and Early Exiting for Deep Learning Applications: Survey and Research Challenges. *ACM Computing Surveys*, 55(5), Article 90, 1–30. DOI: https://doi.org/10.1145/3527155

[34] Laskaridis, S., Venieris, S. I., Almeida, M., Leontiadis, I., & Lane, N. D. (2020). SPINN: Synergistic Progressive Inference of Neural Networks over Device and Cloud. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking (MobiCom)*, Article 37, 1–15. DOI: https://doi.org/10.1145/3372224.3419194

[35] Laskaridis, S., Kouris, A., & Lane, N. D. (2021). Adaptive Inference through Early-Exit Networks: Design, Challenges and Directions. In *Proceedings of the 5th International Workshop on Embedded and Mobile Deep Learning (EMDL)*, 1–6. DOI: https://doi.org/10.1145/3469116.3470012

[36] Krizhevsky, A. (2009). Learning Multiple Layers of Features from Tiny Images. *Technical Report, University of Toronto*. URL: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[37] Le, Y., & Yang, X. (2015). Tiny ImageNet Visual Recognition Challenge. *CS 231N Course Report, Stanford University*.

[38] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778. DOI: https://doi.org/10.1109/CVPR.2016.90

[39] Simonyan, K., & Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*. URL: https://arxiv.org/abs/1409.1556

[40] Brown, G., Wyatt, J., Harris, R., & Yao, X. (2005). Diversity Creation Methods: A Survey and Categorisation. *Information Fusion*, 6(1), 5–20. DOI: https://doi.org/10.1016/j.inffus.2004.04.004

[41] Krogh, A., & Vedelsby, J. (1995). Neural Network Ensembles, Cross Validation, and Active Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 7, 231–238. URL: https://papers.nips.cc/paper/1994/hash/b8c37e33defde51cf91e1e03e51657da-Abstract.html

[42] Tumer, K., & Ghosh, J. (1996). Error Correlation and Error Reduction in Ensemble Classifiers. *Connection Science*, 8(3–4), 385–404. DOI: https://doi.org/10.1080/095400996116839

# A Notation Summary

Table 5: Symbol definitions used in the paper.

| Symbol | Definition |
|---|---|
| $x$ | Input sample |
| $y$ | Ground-truth label |
| $q(x)$ | One-hot vector of the true label, $q(x) \in \{0,1\}^C$, $\sum_c q_c(x) = 1$ |
| $C$ | Number of classes ($C = 100$ for CIFAR-100, $C = 200$ for Tiny-ImageNet-200) |
| $N$ | Number of samples |
| $K$ | Total number of IC heads ($K = 6$ in our experiments) |
| $k$ | Backbone depth at deployment, $k \in \{1, \dots, K\}$ |
| $j$ | IC head index, generally $j \in \{1, \dots, K\}$; in subset contexts $j \in \{1, \dots, k\}$ |
| $z_j(x)$ | Logits vector of head $j$, $z_j(x) \in \mathbb{R}^C$ |
| $p_j(x)$ | Softmax output of head $j$, $p_j(x) = \mathrm{softmax}(z_j(x)) \in \Delta^{C-1}$ |
| $p_j^{nc}$ | Probability for class $c$ at sample $n$, head $j$, $p_j^{nc} = [p_j(x_n)]_c$ |
| $\bar{p}_j$ | Overall mean of $p_j^{nc}$, $\bar{p}_j = \frac{1}{NC}\sum_{n,c} p_j^{nc}$ |
| $p_f(x)$ | Softmax output of the final classifier (final head) |
| $\hat{y}_j(x)$ | Prediction of head $j$, $\hat{y}_j(x) = \arg\max_c [p_j(x)]_c$ |
| $e_j^n$ | Misclassification indicator at sample $n$, head $j$, $e_j^n = \mathbf{1}[\hat{y}_j(x_n) \neq y_n]$ |
| $e_j(x)$ | Population-level error vector $p_j(x) - q(x)$ |
| $\varepsilon_j$ | Error rate of head $j$, $\varepsilon_j = \frac{1}{N}\sum_n e_j^n$ |
| $S$ | IC head subset, $S \subseteq \{1, \dots, k\}$ |
| $S_k^*$ | Optimal subset selected by calibration-based search at depth $k$ |
| $D_{\mathrm{train}}$ | Training split |
| $D_{\mathrm{cal}}$ | Calibration split, held out from training; $D_{\mathrm{cal}} \cap D_{\mathrm{train}} = D_{\mathrm{cal}} \cap D_{\mathrm{test}} = \emptyset$ |
| $D_{\mathrm{test}}$ | Test split |
| $R_{\mathrm{sq}}(p_j)$ | Squared risk of head $j$, $R_{\mathrm{sq}}(p_j) = \mathbb{E}_x[\|p_j(x) - q(x)\|^2]$ |
| $Q_{ij}$ | Residual cross term of heads $i$ and $j$, $Q_{ij} = \mathbb{E}_x[e_i(x)^\top e_j(x)]$ |
| $\hat{Q}_{ij}$ | Sample-mean estimator of $Q_{ij}$, $\hat{Q}_{ij} = \frac{1}{N}\sum_n \langle p_i(x_n) - q(x_n),\ p_j(x_n) - q(x_n)\rangle$ |
| $\rho_{\mathrm{soft}}(i,j)$ | Pearson correlation based on the softmax outputs of heads $i, j$ |
| $\rho_{\mathrm{err}}(i,j)$ | Pearson correlation based on the error indicators of heads $i, j$ |
| $L_{\mathrm{p50}}(M_k)$ | p50 latency of deployment configuration $M_k$ |
| $E(M_k)$ | Per-inference energy of deployment configuration $M_k$ |
| cost | Cost of a single-sample inference |
| $\tau$ | Confidence threshold of the dynamic exit |
| $\alpha$ | KD weight in $L_{\mathrm{LSD}}$ ($\alpha = 0.3$) |
| $T$ | Distillation temperature ($T = 4$) |
| $w_j(t)$ | SDN-style IC loss weight at epoch $t$, $w_j(t) = 0.01 + (t/E)(c_j - 0.01)$ |
| $c_j$ | Relative cost (cumulative FLOPs ratio) of IC $j$ |

# B Proof of Proposition 1

Let $L_{\mathrm{KD},j} = T^2\ \mathrm{KL}\big(\mathrm{sg}(\mathrm{softmax}(h_{\mathrm{final}}(x)/T)) \,\big\|\, \mathrm{softmax}(h_j(x)/T)\big)$ denote the KD term for the $j$-th IC, as in Eq. (4); the scalar factors $w_j(t)$, $\alpha$, and $T^2$ do not affect the argument below. Assume that $\mathrm{softmax}$, cross-entropy, and KL divergence are differentiable with respect to their arguments and that the teacher output is wrapped by the stop-gradient operator $\mathrm{sg}(\cdot)$.

By the definition of the stop-gradient operator, for every parameter $\theta$, we have

$$\frac{\partial}{\partial\theta}\,\mathrm{sg}\big(\mathrm{softmax}(h_{\mathrm{final}}(x)/T)\big) = 0.$$

Applying the chain rule,

$$\frac{\partial L_{\mathrm{KD},j}}{\partial\theta_f} = \frac{\partial L_{\mathrm{KD},j}}{\partial\,\mathrm{sg}(\cdot)} \cdot \underbrace{\frac{\partial\,\mathrm{sg}(\mathrm{softmax}(h_{\mathrm{final}}(x)/T))}{\partial\theta_f}}_{=0} = 0.$$

Furthermore, the student distribution $\mathrm{softmax}(h_j(x)/T)$ depends on $\theta_b$ and $\theta_j$ but not on $\theta_f$; therefore, the gradient path through the student argument of the KL divergence does not contribute to $\theta_f$. Hence, for every $j \in \{1, \dots, K\}$, the above equation holds, and by summing over the KD terms,

$$\frac{\partial}{\partial\theta_f}\sum_{j=1}^{K} L_{\mathrm{KD},j} = 0.$$

The remaining $\theta_f$-dependent term in $L_{\mathrm{LSD}}$ is the cross-entropy term $\mathrm{CE}(h_{\mathrm{final}}(x), y)$. Thus,

$$\frac{\partial L_{\mathrm{LSD}}}{\partial\theta_f} = \frac{\partial \mathrm{CE}(h_{\mathrm{final}}(x), y)}{\partial\theta_f}.$$

That is, $\theta_f$ receives no direct gradient contribution from the KD term and is updated only through its own cross-entropy term. □

**Remark.** *Proposition 1 addresses only the direct gradient contribution to $\theta_f$. The shared backbone parameter $\theta_b$ receives gradients from all auxiliary IC losses and the final cross-entropy term simultaneously. Therefore, the training trajectory of $h_{\text{final}}$ may differ from that of standard SDN. Hence, the comparison of the final classifier accuracy between SLIENet and SDN is not a matter of mathematical guarantee but must be verified empirically, as discussed in Section 4.2.*

# C Derivation of Proposition 2

**Goal.** We derive the necessary and sufficient condition for the squared risk to decrease when adding head $h \notin S$ to an existing subset $S$. The decomposition used here is closely related to the bias–variance–covariance decomposition for squared loss [40, 41] and to the error-correlation analysis of Tumer and Ghosh [42], specialized to the second-moment formulation $Q_{ij} = \mathbb{E}[e_i^\top e_j]$ rather than to centered covariances.

**Squared risk before addition.**

$$R(S) = \frac{1}{|S|^2} \sum_{i,l \in S} Q_{il} = \frac{A}{|S|^2}, \qquad A = \sum_{i,l \in S} Q_{il}.$$

**Squared risk after addition.** For $S' = S \cup \{h\}$ and $Q_{ih} = Q_{hi}$ (symmetry),

$$R(S') = \frac{1}{(|S|+1)^2} \big(A + 2B + Q_{hh}\big), \qquad B = \sum_{i \in S} Q_{hi}.$$

**Decrease condition.** Expanding $R(S') < R(S)$,

$$|S|^2(A + 2B + Q_{hh}) < (|S|+1)^2 A,$$

$$2|S|^2 B + |S|^2 Q_{hh} < (2|S|+1)A.$$

Substituting $n = |S|$ and $R(S) = A/n^2$,

$$Q_{hh} + 2\sum_{i \in S} Q_{hi} < (2n+1)\, R(S). \qquad \text{(equiv. to Eq. (7))}$$

This condition is both necessary and sufficient. Letting $R(S) := R_{\text{sq}}(p_S) = A/|S|^2$, the condition is meaningful provided that the right-hand side $(2n+1)\, R(S)$ is positive. Even if the individual squared risk $Q_{hh}$ is high, this condition can still be satisfied if $\sum_{i \in S} Q_{hi}$ is sufficiently low. Particularly, if $Q_{hh} + 2\sum_{i \in S} Q_{hi} < 0$, the condition is satisfied for any $(2n+1)\, R(S) \geq 0$. □

**Interpretation.** A head with low individual quality can reduce the ensemble risk if the sum of its residual cross terms with the existing ensemble is sufficiently small. This quantitatively explains why low-accuracy early heads (IC2, IC3) are included in the optimal subset when their error patterns are sufficiently uncorrelated with those of late heads.

**Equality condition.** In the inequality

$$R_{\text{sq}}\left(\frac{1}{|S|}\sum_{i \in S} p_i\right) \leq \frac{1}{|S|}\sum_{i \in S} R_{\text{sq}}(p_i),$$

equality holds if and only if $\text{Var}_{i \in S}(p_i(x) - q(x)) = 0$; that is, $p_i(x) = p_j(x)$ for every pair of heads in $S$ and every sample $x$. Because ICs at different depths process structurally different features, this condition does not hold practically. Therefore, under the squared-risk surrogate, even a small amount of diversity makes the ensemble squared risk strictly smaller than the average individual squared risk. This strict inequality is a statement regarding the $L_2$ distance and does not automatically imply a strict improvement in 0–1 classification accuracy. Some $S_k^*$ in Table 8 reducing to a single IC reflects that selection by accuracy may differ from selection by squared risk.

# D Proof of Proposition 3 and Structural Analysis of ZTW Cascades

## D.1 Proof of Proposition 3

**Model.** We consider the stylized first-order residual-coupling model

$$e_j^{\text{cas}} = r_j + B_j\, e_{j-1}^{\text{cas}},$$

where $e_j^{\text{cas}} = p_j^{\text{cas}}(x) - q(x)$ is the error vector of the $j$-th cascade head, $r_j$ is an innovation term, and $B_j \in \mathbb{R}^{C\times C}$ is a fixed (input-independent) matrix representing a linearized dependency on the previous head's output. This is a modeling assumption used for explanation; we do not claim that trained ZTW heads satisfy it exactly. We write $M_{j-1} = \mathbb{E}[e_{j-1}^{\text{cas}}(e_{j-1}^{\text{cas}})^\top]$ for the second-moment matrix and $\mathrm{Var}(e_{j-1}^{\text{cas}}) = M_{j-1} - \mathbb{E}[e_{j-1}^{\text{cas}}]\,\mathbb{E}[e_{j-1}^{\text{cas}}]^\top$ for the covariance matrix. Note that the residual cross term of Section 3.3 is a second moment, $Q_{ij} = \mathrm{tr}\big(\mathbb{E}[e_i e_j^\top]\big)$, not a covariance.

**(i) Covariance identity.** Assume $\mathrm{Cov}(r_j, e_{j-1}^{\text{cas}}) = 0$. Since $B_j$ is fixed, covariance is linear in its first argument, and

$$\mathrm{Cov}\big(e_j^{\text{cas}}, e_{j-1}^{\text{cas}}\big) = \mathrm{Cov}\big(r_j, e_{j-1}^{\text{cas}}\big) + B_j\,\mathrm{Cov}\big(e_{j-1}^{\text{cas}}, e_{j-1}^{\text{cas}}\big) = B_j\,\mathrm{Var}\big(e_{j-1}^{\text{cas}}\big).$$

An assumption on the cross moment alone, $\mathbb{E}[r_j(e_{j-1}^{\text{cas}})^\top] = 0$, does not yield this identity: in general $\mathrm{Cov}(r_j, e_{j-1}^{\text{cas}}) = \mathbb{E}[r_j(e_{j-1}^{\text{cas}})^\top] - \mathbb{E}[r_j]\,\mathbb{E}[e_{j-1}^{\text{cas}}]^\top$, so the two conditions coincide only when $\mathbb{E}[r_j]\,\mathbb{E}[e_{j-1}^{\text{cas}}]^\top = 0$ (e.g., $\mathbb{E}[r_j] = 0$).

**(ii) Second-moment identity.** Assume $\mathbb{E}[r_j(e_{j-1}^{\text{cas}})^\top] = 0$. Then

$$\mathbb{E}\big[e_j^{\text{cas}}(e_{j-1}^{\text{cas}})^\top\big] = \mathbb{E}\big[r_j(e_{j-1}^{\text{cas}})^\top\big] + B_j M_{j-1} = B_j M_{j-1}, \qquad Q_{j,j-1} = \mathrm{tr}\big(B_j M_{j-1}\big).$$

Because $M_{j-1} \succeq 0$, $\mathrm{tr}(B_j M_{j-1}) = \frac{1}{2}\,\mathrm{tr}\big((B_j + B_j^\top)M_{j-1}\big) \geq 0$ whenever $B_j + B_j^\top \succeq 0$ (e.g., $B_j = bI$ with $b \geq 0$, giving $Q_{j,j-1} = b\,Q_{j-1,j-1}$). Under the same assumption, a head with the same innovation but no coupling ($e_j = r_j$) would have $Q_{j,j-1} = 0$, so the coupling contributes the additional cross term $\mathrm{tr}(B_j M_{j-1})$. For an indefinite or negative-definite symmetric part of $B_j$, this term can be zero or negative.

**(iii) Consequence for Proposition 2.** The left-hand side of condition (7), $Q_{hh} + 2\sum_{i\in S} Q_{hi}$, is increasing in each $Q_{hi}$, and its right-hand side does not depend on head $h$. Hence, for a fixed individual risk $Q_{hh}$ and a fixed subset $S$, a larger cross term between $h$ and a member of $S$ makes the head-addition condition harder to satisfy. □

**Scope of the statement.** The proposition does not imply that cascade coupling always increases cross terms, nor that the head-addition condition is always stricter for cascade heads than for independent heads: the sign of $\mathrm{tr}(B_j M_{j-1})$ depends on $B_j$, coupling also changes the individual risk $Q_{jj} = \mathrm{tr}(\mathbb{E}[r_j r_j^\top]) + \mathrm{tr}(B_j M_{j-1} B_j^\top)$ under (ii), and real cascade heads need not follow a fixed linear coupling. The observation that post-hoc selective averaging helps less on cascade heads is therefore empirical (Appendix D.3); the proposition offers one mechanism consistent with it.

### D.2 Additional Remark: Implementation Properties of ZTW-Style Cascade Heads

This subsection, independent of the proof of Proposition 3, analyzes three practical reasons why ZTW's cascade implementation is unsuitable for deploy-partial deployment. The first concerns parameter cost, whereas the second and third concern objective-function inconsistency between training and deployment.

Each ZTW cascade head has the following form:

$$h_j^{\text{cas}}(x) = g\big(f_{1:j}(x),\ h_{j-1}^{\text{cas}}(x)\big),$$

which requires learnable parameters $\phi_j$ of the connection function $g$. For $K = 6$, the total additional parameters introduced by the cascade connections is

$$\Delta\Phi_{\text{ZTW}} = \sum_{j=2}^{K} |\phi_j|, \qquad \text{whereas } \Delta\Phi_{\text{SLIENet}} = 0.$$

Given the ZTW loss

$$L_{\text{ZTW}} = \sum_{j=1}^{K} \mathrm{CE}\big(h_j^{\text{cas}}(x),\ y\big) + \mathrm{CE}\big(h_{\text{final}}(x),\ y\big),$$

the gradient with respect to the parameter $\theta_j^{\text{cas}}$ of head $j$ expands as

$$\frac{\partial L_{\text{ZTW}}}{\partial \theta_j^{\text{cas}}} = \frac{\partial \mathrm{CE}(h_j^{\text{cas}}, y)}{\partial \theta_j^{\text{cas}}} + \sum_{m>j} \frac{\partial \mathrm{CE}(h_m^{\text{cas}}, y)}{\partial \theta_j^{\text{cas}}}.$$

The second sum is the gradient inflow from all cascade heads after head $j$, which causes inter-head coupling.

The second sum optimizes $\theta_j^{\text{cas}}$ under the assumption that the head $j$ functions as an intermediate link in the cascade chain. Removing the heads after $j+1$ at deployment removes this term from the inference path, creating a structural mismatch between training and deployment objectives.

Contrastingly, in $L_{\mathrm{LSD}}$ no IC takes another IC's output as input, and the teacher is detached (Proposition 1). The loss term of each retained IC therefore does not depend on the outputs of the removed heads, so truncation does not remove any term from a retained head's own objective. The ICs still interact through the shared backbone during training, whose parameters are fixed at deployment.

### D.3 Empirical Evidence: SLIE Gain on Independent vs Cascade Heads

Table 6 shows the effect of applying SLIE on top of two different training regimes: the independent-head regime (SDN+KD $\rightarrow$ SLIENet) and cascade regime (ZTW $\rightarrow$ ZTW+SLIE). For each backbone, we report the late-exit accuracy change ($\Delta$ at IC5, IC6, and Final) caused purely by the SLIE post-processing step with the trained checkpoint held fixed. The numbers follow the submission protocol (Section 4.1).

Table 6: SLIE post-hoc gain on independent heads vs cascade heads (%; submission protocol, seed means taken from Tables 1, 10, and 11). For each backbone and exit $k \in \{\text{IC5, IC6, Final}\}$, we report the accuracy before and after SLIE is applied, together with the change $\Delta$. On independent heads (SDN+KD $\rightarrow$ SLIENet), SLIE consistently adds late-exit accuracy. On cascade heads (ZTW $\rightarrow$ ZTW+SLIE), SLIE produces near-zero or negative changes at most exits, an empirical pattern consistent with the mechanism described by Proposition 3.

| | | Independent heads (SDN+KD $\rightarrow$ SLIENet) | | | Cascade heads (ZTW $\rightarrow$ ZTW+SLIE) | | |
|---|---|---|---|---|---|---|---|
| Backbone | Exit | before | after | $\Delta$ | before | after | $\Delta$ |
| ResNet-56 | IC5 | 69.67 | 70.59 | $+0.92$ | 68.35 | 69.25 | $+0.90$ |
| | IC6 | 69.42 | 71.31 | $+1.89$ | 71.51 | 70.15 | $-1.36$ |
| | Final | 69.04 | 71.31 | $+2.27$ | 72.26 | 70.04 | $-2.22$ |
| VGG-16 | IC5 | 76.23 | 76.66 | $+0.43$ | 76.33 | 76.59 | $+0.26$ |
| | IC6 | 76.07 | 76.66 | $+0.59$ | 76.51 | 76.51 | $+0.00$ |
| | Final | 75.77 | 76.66 | $+0.89$ | 76.61 | 76.51 | $-0.10$ |
| MobileNetV1 | IC5 | 72.06 | 73.37 | $+1.31$ | 73.12 | 73.04 | $-0.08$ |
| | IC6 | 71.63 | 73.54 | $+1.91$ | 73.50 | 73.07 | $-0.43$ |
| | Final | 70.97 | 73.54 | $+2.57$ | 73.42 | 73.26 | $-0.16$ |

Two patterns are observed. (i) On independent heads, every $\Delta$ at IC5/IC6/Final across three backbones is positive, with mean $+1.42$ pp and the largest gain $+2.57$ pp on MobileNetV1 Final. (ii) On cascade heads, the same SLIE procedure produces only one non-trivially positive $\Delta$ (ResNet-56 IC5 $+0.90$; matched almost exactly by SLIENet's $+0.92$), two near-zero $\Delta$ (VGG-16 IC5/IC6), and six non-positive $\Delta$, including two substantially negative entries on ResNet-56 ($-1.36$ at IC6, $-2.22$ at Final).

This asymmetry is an empirical observation; we did not estimate $B_j$ or test the residual-coupling model on trained heads. It is consistent with the mechanism of Proposition 3: cascade heads are trained to reuse the previous head's logits, and if that reuse acts like a coupling with $B_j + B_j^\top \succeq 0$, it adds a non-negative term to the cross terms $Q_{hi}$ in the head-addition condition of Proposition 2, making selective averaging less effective at deep cascade exits. SLIENet's ICs have no explicit cascade dependency, since no IC receives another head's output, so the coupling term of the stylized model is absent by construction. This does not make their errors uncorrelated: all ICs share the backbone, and their cross terms $Q_{ij}$ remain non-zero (Appendix F). What differs is that no additional coupling term is introduced on top of the correlation induced by the shared trunk.

## E Calibration and Sensitivity Analysis

### E.1 Calibration Complexity

The SLIE calibration consists of two stages. First, a single forward pass over $D_{\mathrm{cal}}$ is performed to store the logits of all ICs. Second, the softmax-average accuracy is computed for each of the $2^k - 1$ subset combinations using the stored logits. For $k = 6$, the search covers 63 combinations, and the per-combination evaluation cost is a $|D_{\mathrm{cal}}| \times C$-dimensional vector average. The total calibration cost is

$$\underbrace{|D_{\mathrm{cal}}| \cdot \mathrm{cost}}_{\text{1 inference pass}} + \underbrace{(2^k - 1) \cdot |D_{\mathrm{cal}}| \cdot C}_{\text{logit averaging}}.$$

Because $C = 100 \ll \mathrm{cost}$, the second term is negligible, and the search is completed in less than 1 s on the CPU in our measurement.

**Comparison with SDN and ZTW.** SDN searches the exit threshold separately for each of the $K$ ICs, running inference on the entire $D_{\mathrm{cal}}$ for each threshold candidate. ZTW also requires sequential threshold re-evaluation owing to inter-cascade-head dependencies. The calibration cost of both is $O(K \cdot |D_{\mathrm{cal}}| \cdot \mathrm{cost})$, which is $O(K)$ times higher than that of SLIENet.

### E.2 Sensitivity to $N_{\text{cal}}$

Table 7: Change in SLIE accuracy versus single $\text{IC}_k$ as a function of calibration set size $N_{\text{cal}}$ (ResNet-56 / CIFAR-100, submission protocol). Negative values indicate that SLIE underperforms the single IC due to unstable subset selection. $N_{\text{cal}} = 2{,}000$ is the smallest size at which the average gain over $k = 1$–5 becomes positive.

| $N_{\text{cal}}$ | $k$=1 | $k$=2 | $k$=3 | $k$=4 | $k$=5 | $k$=6 | avg (all $k$) | avg ($k$=1–5) |
|---|---|---|---|---|---|---|---|---|
| 500 | +0.00 | −0.20 | −0.07 | −0.34 | +0.22 | +1.72 | +0.22 | −0.078 |
| 1,000 | +0.00 | −0.58 | +0.00 | −0.20 | +0.22 | +1.81 | +0.21 | −0.112 |
| **2,000** | +0.00 | −0.29 | +0.00 | −0.02 | +0.39 | +1.97 | +0.34 | +0.016 |
| 5,000 | +0.00 | −0.45 | +0.00 | −0.02 | +0.54 | +2.26 | +0.39 | +0.014 |

Table 7 lists the SLIE ensemble gains over the single $\text{IC}_k$ as $N_{\text{cal}}$ varies in $\{500, 1{,}000, 2{,}000, 5{,}000\}$. Two observations follow.

First, in the practical deployment range, $k = 1$–5, $N_{\text{cal}} = 500$ and $N_{\text{cal}} = 1{,}000$ yield $-0.078$ pp and $-0.112$ pp, respectively, on average over $k = 1$–5. That is, when the calibration set is too small, $S_k^*$ selection becomes unstable, and the ensemble can underperform the single IC at intermediate $k$ (negative $\Delta$ at $k = 2, 4$). From $N_{\text{cal}} = 2{,}000$, the average over $k = 1$–5 becomes positive at $+0.016$ pp, and at $N_{\text{cal}} = 5{,}000$, it remains essentially the same at $+0.014$ pp.

Second, at $k = 6$, the ensemble gain increases monotonically with $N_{\text{cal}}$ ($+1.72 \to +1.97 \to +2.26$ pp), but the additional gain at $N_{\text{cal}} = 5{,}000$ over $N_{\text{cal}} = 2{,}000$ is $+0.29$ pp, while the calibration cost increases by $2.5\times$.

Therefore, we adopt $N_{\text{cal}} = 2{,}000$ as the minimum size at which the ensemble gain stabilizes positively over $k = 1$–5, with no loss in the $k = 1$–5 average compared to $N_{\text{cal}} = 5{,}000$. Two practical implications follow. First, SLIE needs a modest labeled calibration set, and with fewer than ≈2,000 labeled samples the single IC at the deployment depth is the safer choice. Second, because calibration runs on stored logits and takes well under a second, the $N_{\text{cal}}$ sensitivity is an offline cost paid once per deployment distribution, not a runtime burden. The gain at $k = 1$–5 is small even at $N_{\text{cal}} = 2{,}000$, consistent with the modest $+0.32$ pp ensemble contribution at IC5 in Table 3.

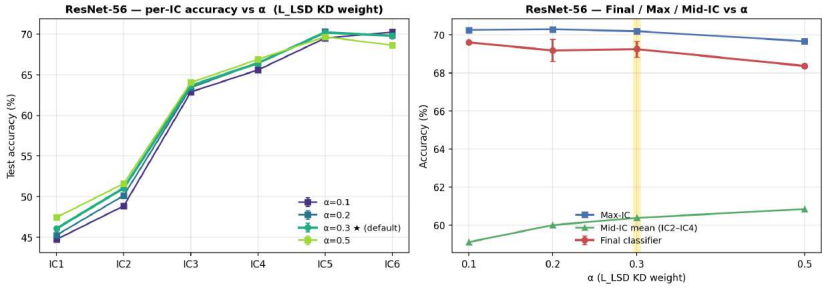



Figure 4: Per-IC accuracy on ResNet-56 (CIFAR-100, submission protocol) as a function of the KD weight $\alpha \in \{0.1, 0.2, 0.3, 0.5\}$. The IC-level differences are small from IC3 onward; $\alpha = 0.3$ is the balanced choice across max-IC, mid-IC, and final accuracy.

Figure 4 shows the change in per-IC accuracy on ResNet-56 as the KD weight $\alpha$ of $L_{\text{LSD}}$ varies in $\{0.1, 0.2, 0.3, 0.5\}$. In the left subfigure, the per-IC accuracy differences across $\alpha$ values are generally small, and the curves nearly overlap from IC3 onward. As $\alpha$ increases, the accuracy of early ICs (IC1, IC2) tends to improve slightly because dark knowledge absorption is reinforced, and the representational capacity of shallow layers improves. In the right subfigure, $\alpha = 0.3$ (yellow shading) is the balanced choice across three indicators. The max-IC accuracy (blue) remains stable up to $\alpha = 0.3$ and decreases slightly at $\alpha = 0.5$. The final classifier accuracy (red) tends to decrease as $\alpha$ increases, potentially because changing the balance between IC cross-entropy and distillation losses alters the optimization of the shared backbone. The final classifier's own cross-entropy term remains directly weighted by one (Eq. (4)). The mid-IC mean (green) peaks at $\alpha = 0.3$. Therefore, we adopt $\alpha = 0.3$ as the default value.

Table 8: Multi-seed SLIE ensemble subset selection (ResNet-56, submission protocol). The IC at the truncation depth is always included; early ICs (IC2, IC3) are repeatedly selected as diversity-providing complements.

| Level | seed 0 | seed 1 | seed 2 | seed 3 | seed 4 |
|---|---|---|---|---|---|
| SLIE4 | IC4 | IC3, IC4 | IC2, IC4 | IC2, IC4 | IC3, IC4 |
| SLIE5 | IC3, IC5 | IC2, IC4, IC5 | IC3, IC5 | IC2, IC4, IC5 | IC2, IC3, IC5 |
| SLIE6 | IC2, IC5, IC6 | IC4, IC6 | IC2, IC4, IC5, IC6 | IC4, IC6 | IC2, IC5, IC6 |

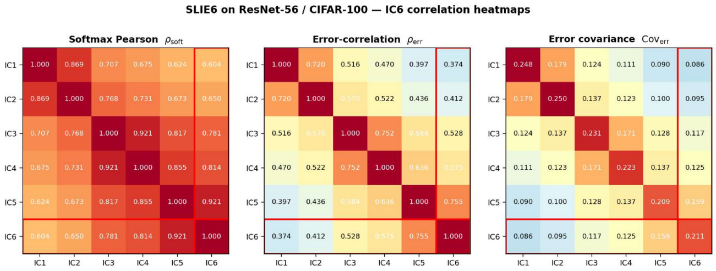



Figure 5: Correlation heatmap of SLIE6 on ResNet-56 (submission protocol, averaged over 5 seeds). The matrix shows error-indicator covariance based on $e_i^n = \mathbf{1}[y_i \neq y]$, which differs from the residual cross term $Q_{ij}$ (a second moment of the error vectors) used in the main text.

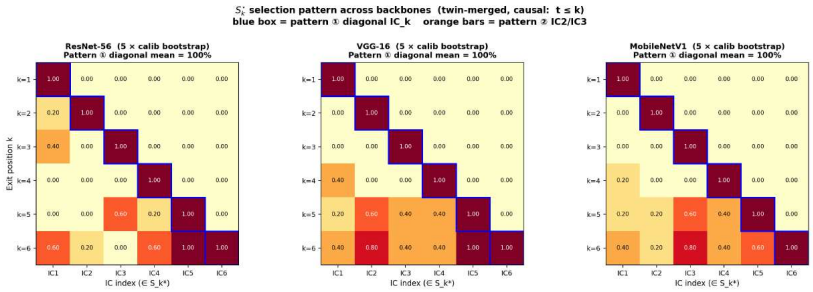



Figure 6: $S_k^*$ selection frequencies measured by 5 calibration bootstraps on each backbone (ResNet-56, VGG-16, MobileNetV1; submission protocol), shown as heatmaps and bar charts.

# F IC Diversity Analysis and Subset Selection Patterns

## F.1 $S_k^*$ Selection Patterns

Figure 6 shows two consistent patterns across the three backbones. First, the IC at truncation depth $k$ (diagonal cell) is selected in $100\%$ of bootstraps. Second, IC2 and IC3 are repeatedly included in $S_k^*$, especially for $k = 5, 6$.

The IC at the truncation depth is always included in $S_k^*$ across all sizes; IC6 has the lowest selection frequency ($17\%$). This is because IC6 is included only at $k = 6$ in the diagonal pattern and not for other $k$. This pattern suggests that the cross-term-conditioned gain of Proposition 2 operates consistently across backbone structures.

## F.2 Empirical Evidence

**Setup.** ResNet-56 / CIFAR-100, mean over seeds 0/1/2/3/4, 10,000 test samples.

We estimate the residual cross term $Q_{ij}$ using the sample mean

$$\hat{Q}_{ij} = \frac{1}{N} \sum_{n=1}^{N} \langle p_i(x_n) - q(x_n),\ p_j(x_n) - q(x_n) \rangle.$$

For comparison, we also report the Pearson correlation $\rho_{\text{soft}}$ based on softmax outputs and $\rho_{\text{err}}$ based on error indicators. With $\hat{y}_i = \arg\max_c p_i$, $e_i^n = \mathbf{1}[\hat{y}_i^n \neq y_n]$, $\varepsilon_i = \frac{1}{N}\sum_n e_i^n$, and $\bar{p}_i = \frac{1}{NC}\sum_{n,c} p_i^{nc}$,

$$\rho_{\text{soft}}(i,j) = \frac{\sum_{n,c}(p_i^{nc} - \bar{p}_i)(p_j^{nc} - \bar{p}_j)}{\sqrt{\sum_{n,c}(p_i^{nc} - \bar{p}_i)^2}\sqrt{\sum_{n,c}(p_j^{nc} - \bar{p}_j)^2}}, \tag{8}$$

$$\text{Cov}(e_i, e_j) = \frac{1}{N}\sum_n e_i^n e_j^n - \varepsilon_i\varepsilon_j, \qquad \rho_{\text{err}}(i,j) = \frac{\text{Cov}(e_i,e_j)}{\sqrt{\varepsilon_i(1-\varepsilon_i)\,\varepsilon_j(1-\varepsilon_j)}}. \tag{9}$$

Table 9: Row-IC6 numbers: correlation and residual cross terms with IC6 (ResNet-56, submission protocol). The correlation with IC6 is lowest for shallow ICs (IC1, IC2) and highest for IC5, identifying shallow ICs as the best diversity-providing complements.

| Target | $\rho_{\text{soft}}(\text{IC6}, \cdot)$ | $\rho_{\text{err}}(\text{IC6}, \cdot)$ | $Q_{i6}$ |
|---|---|---|---|
| IC1 | 0.604 | 0.374 | 0.0857 |
| IC2 | 0.650 | 0.412 | 0.0945 |
| IC3 | 0.781 | 0.528 | 0.1167 |
| IC4 | 0.814 | 0.575 | 0.1246 |
| IC5 | 0.921 | 0.755 | 0.1586 |

Table 9 shows that the correlation with IC6 is lower for shallower ICs and higher for deeper ICs (IC5). IC5 has nearly the same residual structure as IC6 and offers little ensemble gain, whereas IC1–IC3 have relatively small $Q_{ij}$ values and are the best candidates for diversity-providing complementary classifiers.

### F.3 Inter-IC Error Correlation Analysis

By the decreasing condition of Proposition 2, $Q_{hh} + 2\sum_{i\in S} Q_{hi} < (2n+1)\,R(S)$, IC1 and IC2 with small $Q_{hi}$ most easily satisfy the risk-decrease condition when added to the ensemble. This quantitatively explains why IC2 is repeatedly selected in the $S_k^*$ pattern.

**Ensemble variance decomposition.** The variance reduction effect of averaging correlated classifiers is a classical result in the ensemble literature [42, 41, 40]. When $M$ classifiers are uniformly averaged, the error variance is

$$\text{Var}(\bar{e}) \propto \frac{1}{M}\big(1 + (M-1)\,\rho\big)\,\sigma^2.$$

As $\rho \to 0$, the variance decreases as $1/M$, achieving the full variance attenuation effect. Conversely, as $\rho \to 1$, the ensemble becomes identical to a single classifier. The classical added-error decomposition of Tumer and Ghosh [42] is consistent with this:

$$E_{\text{add}}^{\text{ens}} = \frac{1 + (M-1)\,\rho}{M}\,E_{\text{add}}^{\text{avg}}.$$

Equivalently, the ambiguity decomposition of Krogh and Vedelsby [41] for the squared loss expresses the ensemble error as the average individual error minus the diversity term, providing the regression-side counterpart to the squared-risk surrogate analysis used in Proposition 2.

## G Per-IC Accuracy on VGG-16 and MobileNetV1

Tables 10 and 11 report the complete per-IC accuracy breakdown on VGG-16 and MobileNetV1, complementing the summary in Table 2; they come from a different set of seeds, so their final-exit means differ slightly from Table 2. The per-depth comparison with SDN, SDN+SLIE, SDN+KD, ZTW Cascade, and ZTW+SLIE supports the observations made in the main text: SLIENet is either the best or statistically tied with the best at IC5, IC6, and Final, whereas ZTW+SLIE shows non-positive late-IC changes compared with pure ZTW on both backbones, consistent with Proposition 3.

## H Evaluation Protocol Audit

The submitted manuscript described the calibration split inconsistently: Section 3.1 defined $D_{\text{cal}}$ as a separate partition ($D = D_{\text{train}} \cup D_{\text{cal}} \cup D_{\text{test}}$), whereas the experimental setup reported 50K training images with the 2K calibration images drawn from the training split. Both descriptions kept the test set disjoint from calibration and used it only for final evaluation. To remove the ambiguity, we reran all deployment-facing comparisons under one strict protocol: 48K training images, 2K held-out calibration images never used for training, and the 10K test set used only for final evaluation. Table 12 summarizes the effect at the main ResNet-56 budgets.

Absolute accuracies drop by 0.3–1.1 pp under the strict protocol, and the headline accuracy gain of SLIE5 over vanilla on the device decreases from the $+2.46$ pp reported in the submission to $+1.2$ pp; this version

Table 10: VGG-16 per-IC accuracy (%) on CIFAR-100 (submission protocol). PyTorch FP32, workstation, mean over 5 seeds from a seed set different from Table 2, so final-exit values differ slightly. SLIENet's final accuracy is numerically highest but within 0.05 pp of ZTW Cascade.

| Framework | IC1 | IC2 | IC3 | IC4 | IC5 | IC6 | Final |
|---|---|---|---|---|---|---|---|
| Backbone only | – | – | – | – | – | – | 74.54 |
| SDN | 46.34 | 65.10 | 73.75 | 74.83 | 76.06 | 75.82 | 75.49 |
| SDN+SLIE | 46.34 | 65.10 | 73.41 | 74.83 | 76.40 | 76.49 | 76.49 |
| SDN+KD ($\alpha$=0.3) | 46.67 | 65.22 | 74.33 | 75.10 | 76.23 | 76.07 | 75.77 |
| **SLIENet** | 46.67 | 65.22 | 74.33 | 75.25 | **76.66** | **76.66** | **76.66** |
| ZTW Cascade | 51.63 | 64.81 | 74.17 | 75.03 | 76.33 | 76.51 | 76.61 |
| ZTW+SLIE | 51.63 | 66.41 | 74.38 | 75.44 | 76.59 | 76.51 | 76.51 |

Table 11: MobileNetV1 per-IC accuracy (%) on CIFAR-100 (submission protocol). PyTorch FP32, workstation, mean over 5 seeds from a seed set different from Table 2, so final-exit values differ slightly. SLIENet is numerically highest at IC5, IC6, and the final exit, by at most 0.25 pp.

| Framework | IC1 | IC2 | IC3 | IC4 | IC5 | IC6 | Final |
|---|---|---|---|---|---|---|---|
| Backbone only | – | – | – | – | – | – | 69.42 |
| SDN | 48.89 | 62.33 | 69.49 | 71.66 | 71.15 | 70.88 | 70.74 |
| SDN+SLIE | 48.89 | 62.33 | 68.83 | 72.10 | 73.11 | 73.11 | 73.11 |
| SDN+KD ($\alpha$=0.3) | 48.59 | 62.74 | 69.89 | 72.50 | 72.06 | 71.63 | 70.97 |
| **SLIENet** | 48.59 | 62.74 | 69.89 | 72.21 | **73.37** | **73.54** | **73.54** |
| ZTW Cascade | 54.02 | 63.73 | 68.05 | 71.86 | 73.12 | 73.50 | 73.42 |
| ZTW+SLIE | 54.02 | 63.73 | 69.75 | 72.35 | 73.04 | 73.07 | 73.26 |

Table 12: Effect of the strict protocol on ResNet-56 / CIFAR-100 accuracy (%).

| Quantity | Submission protocol (5 seeds) | Strict protocol (3 seeds) |
|---|---|---|
| SLIENet at IC5 (SLIE5) | 70.59 | 69.81 |
| SLIENet at IC6 (SLIE6) | 71.31 | 70.97 |
| SLIE gain over single IC6 | $\approx$+1.9–2.0 | +2.08 |
| SLIE5 vs. vanilla (workstation) | – | +1.15 |
| SLIE5 vs. vanilla (Jetson, seed 0) | +2.46 (earlier single run) | +1.23 (ten runs) |

reports the strict-protocol value throughout the abstract, introduction, and conclusion. The latency and energy reductions in the submission ($-14\%$ p50, $-15\%$ energy) came from a single measurement run per configuration in an earlier session and are superseded by the locked-clock ten-run benchmark in Table 4 ($-30\%$ p50, $-40\%$ energy). The per-IC comparison tables that are labeled "submission protocol" are retained because they compare all frameworks under one common protocol; they support only the workstation cross-framework comparison, not the deployment claims.

# I Jetson Measurement Protocol

All results in Table 4 were measured in a single session on a Jetson Orin Nano 8GB (Tegra R36.4.7, TensorRT 10.3.0) with the nvpmodel 25 W power mode and clocks locked via `jetson_clocks`. Each configuration was compiled from one fixed seed-0 checkpoint trained under the strict protocol. Inference used batch size 1 with 200 warm-up iterations; latency was timed with `time.perf_counter` with an explicit `stream.synchronize()` so that GPU work is included. Power was sampled from the on-board INA3221 monitor every 10 ms. Each configuration was run ten times over the 10,000 test samples, with a 30 s cooldown between runs; we report the mean $\pm$ std across the ten runs. SDN dynamic exit was executed as a chain of per-segment engines (1–7 invocations depending on the exit), and ZTW as six engine invocations because of its cascade dependencies.

# J Supplementary Deployment Results

## J.1 ONNX Runtime: Fused Graph vs. Split Session Invocations

To check whether the dispatch overhead is specific to TensorRT, we ran the same nominal SLIE5 computation on a workstation with ONNX Runtime 1.23 (CPU execution provider; the CUDA provider was unavailable in this build), batch 1, either as one fused graph or as five separate session invocations (Table 13). Splitting the computation raises p50 latency from 0.502 to 1.805 ms. This shows that repeated graph/session dispatch can also be costly in a second graph-execution environment; it does not establish behavior under JIT/AoT dynamic executors, and we do not infer a general scaling law from these two configurations.

Table 13: ONNX Runtime (CPU execution provider) latency of the same nominal SLIE5 computation.

| Configuration | Session invocations | mean (ms) | p50 (ms) | p99 (ms) |
|---|---|---|---|---|
| SLIE5, one fused graph | 1 | 1.190 | 0.502 | 9.904 |
| SLIE5, five session invocations | 5 | 2.543 | 1.805 | 10.980 |

### J.2 INT8 Quantization

Table 14 reports the FP16→INT8 change for vanilla ResNet-56 and an IC6-budget SLIE engine on the Jetson Orin Nano (five runs). This study was conducted in an earlier session without `jetson_clocks`, so its FP16 absolute values differ from the locked-clock session in Table 4; the FP16→INT8 deltas are within-session. INT8 changed accuracy by at most $0.10$ pp for either configuration, so we describe it as approximately accuracy-preserving in this experiment rather than in general. SLIE (IC6 budget) with INT8 reduces p50 latency by $33\%$ and energy by $30\%$ relative to vanilla FP16. Pruning is a further orthogonal axis that we did not evaluate.

Table 14: FP16→INT8 on Jetson Orin Nano (TensorRT, five runs, earlier unlocked-clock session).

| Configuration | $\Delta$Acc (pp) | p50 (ms), FP16 → INT8 | Energy (mJ), FP16 → INT8 |
|---|---|---|---|
| Vanilla ResNet-56 | $-0.03$ | 2.67 → 2.62 | 12.73 → 11.97 |
| SLIE (IC6 budget) | $+0.10$ | 2.24 → 1.80 | 10.79 → 8.97 |

## K Tiny-ImageNet-200 Results

Both experiments use the strict protocol (2K calibration images held out from the training set; the 10K validation set used as the test set) and three seeds; SLIENet and ZTW share the same split. Table 15 compares SLIENet with ZTW on ResNet-56 at $32\times32$, and Table 16 reports single-IC and SLIE accuracy for ResNet-18 ($14.45$M parameters) at the native $64\times64$ resolution, trained with the same recipe ($\alpha = 0.3$, $T = 4$, stop-gradient teacher).

Table 15: Tiny-ImageNet-200, ResNet-56 at $32\times32$: per-exit top-1 accuracy (%), mean $\pm$ std over 3 seeds.

| Method | IC3 | IC4 | IC5 | IC6 | Final |
|---|---|---|---|---|---|
| SLIENet (ours) | **42.51 ± 0.38** | **45.32 ± 0.43** | **51.82 ± 0.30** | **53.48 ± 0.20** | – |
| ZTW ensemble | 40.71 ± 0.16 | 42.23 ± 0.22 | 49.80 ± 0.23 | 53.47 ± 0.20 | 53.80 ± 0.23 |

Table 16: Tiny-ImageNet-200, ResNet-18 at $64\times64$: single-IC vs. SLIE accuracy (%), mean $\pm$ std over 3 seeds. "Relative cost" is relative to the deepest IC. The final-FC accuracy is $66.37 \pm 0.21$.

| Exit | Relative cost | Single IC | SLIE (ours) | Gain | Subset (seed 0) |
|---|---|---|---|---|---|
| IC1 | 0.27 | 38.62 ± 0.55 | 38.62 ± 0.55 | $+0.00$ | {IC1} |
| IC2 | 0.52 | 52.58 ± 0.63 | 52.58 ± 0.63 | $+0.00$ | {IC2} |
| IC3 | 0.76 | 63.28 ± 0.26 | 63.10 ± 0.26 | $-0.18$ | {IC2, IC3} |
| IC4 | 1.00 | 66.30 ± 0.21 | **67.35 ± 0.45** | $+1.05$ | {IC2, IC3, IC4} |

On ResNet-18, forward-greedy selection agreed with the exhaustive optimum in all 12 seed×exit cases, although the selected subsets varied across seeds.

## L Implementation Details

### L.1 Training Hyperparameters

Table 17 separates the optimization settings shared by all trained models, the method-specific loss settings, and the dataset- and backbone-specific settings. The settings below follow the training scripts used for the reported experiments.

Table 17: Training configuration. (a) Shared optimization defaults for all methods and backbones, unless otherwise specified below. (b) Loss and training schedule by method; the KD weight $\alpha$ and temperature $T$ apply only to the light self-distillation loss. (c) Dataset- and backbone-specific settings.

| | |
|---|---|
| **(a) Shared optimization** | |
| Optimizer | SGD, Nesterov momentum 0.9 |
| Initial learning rate / schedule | 0.1 / cosine annealing over all epochs |
| Batch size | 128 |
| Epochs | Single-stage methods (SDN, SDN+KD/SLIENet, truncated baselines): 100; ZTW: see (b) |
| Weight decay | $10^{-4}$ (CIFAR-100, ResNet-56 on Tiny-ImageNet); $5 \times 10^{-4}$ (ResNet-18 on Tiny-ImageNet) |
| IC head type | MixedPool (adaptive AvgPool and MaxPool with a learnable mixing weight, then FC) |
| **(b) Method-specific loss** | |
| SDN ($\alpha$=0) | Eq. (3): IC cross-entropy weighted by $w_j(t)$ plus final cross-entropy |
| SDN+KD / SLIENet (LSD) | Eq. (4): $\alpha = 0.3$, $T = 4$, $T^2$-scaled KL from the detached final head |
| IC weight $w_j(t)$ (SDN, LSD) | $0.01 + (t/E)(c_j - 0.01)$, linear ramp to the cumulative-FLOPs ratio $c_j$ of IC $j$ |
| ZTW | Official two-phase recipe: 100-epoch cascade training followed by 25-epoch ensemble training (125 epochs in total) |
| Truncated baselines | Single head at IC5 depth; cross-entropy (scratch) or KD from a full ResNet-56 teacher |
| SDN+SLIE, SLIENet (SLIE), ZTW+SLIE | No extra training; post-hoc subset selection on $D_{\text{cal}}$ |
| **(c) Dataset and backbone** | |
| CIFAR-100 (ResNet-56, VGG-16-BN, MobileNetV1) | $K = 6$ ICs; 32×32 inputs; random crop with 4-pixel padding, horizontal flip; CIFAR-100 mean/std |
| Tiny-ImageNet, ResNet-56 | $K = 6$ ICs (as on CIFAR-100); images downsampled to 32×32; same augmentation; $C = 200$ |
| Tiny-ImageNet, ResNet-18 | $K = 4$ ICs at the end of each stage (blocks 1, 3, 5, 7; relative costs $0.27/0.52/0.76/1.00$); native 64×64 inputs, 3×3 stride-1 stem without max-pooling; random 64×64 crop after 4-pixel reflect padding, horizontal flip; Tiny-ImageNet mean/std; $C = 200$ |
| Calibration / test split | 2,000 images held out from training; CIFAR-100 test set or Tiny-ImageNet validation set as test |
| Seeds | 5 (submission-protocol tables); 3 (strict protocol and Tiny-ImageNet; 5 for truncated baselines) |
| **Hardware** | |
| Workstation / edge device | RTX 4090 (24 GB), PyTorch FP32 / Jetson Orin Nano 8GB, TensorRT 10.3.0 FP16, nvpmodel 25 W, `jetson_clocks` |

## L.2 Compute and Wall-Clock Cost

A single 100-epoch training run on the RTX 4090 takes approximately 1–2 wall-clock hours per backbone (peak GPU memory $< 6$ GB, well within the 24 GB limit), with ResNet-56 the fastest and MobileNetV1 the slowest among the three. Only four models are trained per backbone and seed: the vanilla backbone, SDN ($\alpha = 0$), the LSD network ($\alpha = 0.3$, shared by SDN+KD and SLIENet), and ZTW (100 epochs of joint training, then a 25-epoch ensemble phase with the backbone frozen); SDN+SLIE, SLIENet's SLIE, and ZTW+SLIE are post-hoc selections on these checkpoints. The submission-protocol results thus comprise $4 \times 5 = 20$ runs for ResNet-56 (Tables 1 and 2 share them) and, for each of VGG-16 and MobileNetV1, $3 \times 5 = 15$ runs for Table 2 (no vanilla row) plus $4 \times 5 = 20$ runs on a separate seed set for Tables 10 and 11: 90 training runs, or roughly 90–180 GPU-hours at 1–2 hours each plus the short ZTW ensemble phases; the strict-protocol reruns, truncated baselines, and Tiny-ImageNet experiments added a comparable amount (ResNet-18 at 64×64 being the most expensive). The on-device latency, power, and per-inference energy measurements (Table 4) consist of ten runs over the 10,000 CIFAR-100 test samples per configuration, each run taking under one minute on the Jetson Orin Nano, separated by 30 s cooldowns (Appendix I). The SLIE calibration step is performed once on $D_{\text{cal}}$ (2,000 samples) and completes in under 1 s on a CPU per checkpoint; details are given in Appendix E.1. Preliminary and ablation runs not included in the paper consumed additional compute on the same hardware.

## L.3 Standard Deviations Across Seeds

The per-cell standard deviations across the 5 random seeds are reported as $\pm$ values in Table 1 (ResNet-56, per IC) and Table 2 (final exit, all three backbones), capturing variability due to weight initialization, mini-batch sampling order, and CUDA non-determinism. Strict-protocol results (Table 3 and Appendix K) report the standard deviation across 3 seeds (5 for truncated baselines), and the on-device results in Table 4 report the standard deviation across ten measurement runs of a fixed checkpoint. We report standard deviation rather than standard error of the mean, with no Normality assumption. The seed-level variation of the SLIE subset selection is illustrated in Table 8, and the diversity-statistic heatmap in Figure 5 is averaged over the same 5 seeds.

# NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and Section 1 state the three contributions (deployment-gap demonstration, Train-Full Deploy-Partial, SLIE) and the headline numbers ($+1.2$ pp accuracy, $-30\%$ p50 latency, $-40\%$ energy, zero deadline misses for SLIE5 vs. vanilla ResNet-56) that match the strict-protocol, locked-clock results in Section 4, especially Tables 3 and 4. The abstract also states the scope (static-compiled CNN deployment with TensorRT FP16 on Jetson-class hardware) and the cases where SLIENet is not superior (a matched distilled truncated network at a single depth; ZTW when accuracy is the priority).

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: Section 5 (Discussion and Limitations) scopes the claims to static-compiled CNN deployment with TensorRT FP16 on Jetson-class hardware; lists unevaluated settings (ImageNet-scale data, transformer backbones, more capable accelerators, coarse-grained branching, many exits, JIT/AoT executors); states when dynamic exit may remain preferable; discusses the accuracy–deployability trade-off with ZTW and the fact that a distilled truncated student matches SLIENet at a single depth; and notes the need for a labeled calibration set ($N_{\text{cal}} \geq 2{,}000$), the unquantified sensitivity to calibration–test distribution shift, the scalability of exhaustive subset search, the explanatory status of the propositions, and the lack of runtime adaptation. Appendix H documents the protocol correction relative to the submission.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

   Justification: All three Propositions are stated with their assumptions and have full proofs in the appendix; they are presented as explanatory analysis rather than primary theoretical contributions, and Proposition 3 relies on an explicitly stated stylized coupling model: Proposition 1 in Appendix B, Proposition 2 in Appendix C, and Proposition 3 in Appendix D.

   Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: Section 4.1 specifies the datasets (CIFAR-100 and Tiny-ImageNet-200), splits, the three backbones (ResNet-56, VGG-16, MobileNetV1), the IC-head design (MixedPool, $K = 6$), the KD weight $\alpha = 0.3$, the strict data protocol (48K train / 2K held-out calibration / 10K test) and calibration set size ($N_{\text{cal}} = 2{,}000$), the workstation precision (PyTorch FP32, RTX 4090), and the deployment precision (TensorRT FP16 on Jetson Orin Nano). The complete training hyperparameter table (optimizer, learning rate, schedule, batch size, epochs, KD temperature, augmentation), GPU memory, and per-run wall-clock cost are provided in Appendix L. The Jetson measurement protocol (software versions, power mode, clock locking, warm-up, synchronization, power sampling, cooldown, number of runs) is given in Appendix I. Commands and deployment scripts are provided in the public repository linked from the "Code Availability" note.

   Guidelines:

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

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [Yes]

   Justification: A public repository containing the code and deployment scripts (`https://github.com/Sudals/SLIENet`) is linked from the "Code Availability" note in the main paper (immediately following the abstract). The datasets, CIFAR-100 [36] and Tiny-ImageNet-200, are publicly available. The methodology is also fully described in Section 3 and Appendix E for re-implementation, and the workstation evaluation, the Jetson Orin Nano deployment scripts, and the calibration-based subset selection routine are all included in the repository.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: Section 4.1 provides the high-level setup (dataset, splits, backbones, IC-head design, $\alpha$, calibration size, evaluation precision, seed count). The complete hyperparameter table (optimizer, learning rate schedule, batch size, epochs, KD temperature, augmentation) is given in Appendix L (Implementation Details), and Appendix E provides the calibration size choice and the rationale for $\alpha = 0.3$.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: The main ResNet-56 result in Table 1 is reported as mean $\pm$ standard deviation across 5 independent training runs with different random seeds, capturing variability due to weight initialization, mini-batch sampling order, and CUDA non-determinism. We report 1-$\sigma$ standard deviation (not standard error of the mean) and do not assume Normality. Per-cell standard deviations are computed directly from the 5 seed-level accuracies. To keep the per-IC tables compact across the three backbones,

the auxiliary tables (Tables 2, 10, 11) report seed means only; the typical per-cell standard-deviation range and computation method are stated in Appendix L.3. The strict-protocol results (Table 3, Appendix K) report mean $\pm$ std over 3 seeds (5 for truncated baselines). The on-device latency, power, energy, and deadline-miss results in Table 4 report mean $\pm$ std over ten independent measurement runs of 10,000 samples each, capturing run-to-run variability of the device under a locked-clock protocol. The seed-level variation of the SLIE subset selection itself is illustrated in Table 8, and the diversity-statistic heatmap in Figure 5 is averaged over the same 5 seeds.

Guidelines:

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: Workstation training and evaluation were conducted on a single NVIDIA RTX 4090 GPU (24 GB), and on-device deployment evaluation was performed on an NVIDIA Jetson Orin Nano with TensorRT FP16 engines compiled from the same checkpoints, as stated in Section 4.1. A single 100-epoch training run on the RTX 4090 takes approximately 1–2 wall-clock hours per backbone with peak GPU memory under 6 GB. The submission-protocol result set spans three backbones $\times$ six methods $\times$ five seeds = 90 training runs, totaling on the order of $10^2$ GPU-hours, and the strict-protocol reruns and Tiny-ImageNet experiments added a comparable amount. The on-device latency, power, and per-inference energy measurements (Table 4) consist of ten runs over the 10,000 CIFAR-100 test samples per configuration. The complete compute breakdown, including calibration cost, is reported in Appendix L.2. Preliminary and ablation runs not included in the paper consumed additional compute on the same hardware.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The work conforms with the NeurIPS Code of Ethics. It uses public benchmarks (CIFAR-100 and Tiny-ImageNet-200), involves no human subjects or personal data, and addresses only model-efficiency methodology.

Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: The work targets edge inference efficiency and deterministic latency on devices such as Jetson Orin Nano. Positive impacts include reduced energy consumption and reliable real-time guarantees for safety-critical applications. The methodology is general and not tied to a particular downstream application; we do not foresee specific negative societal impacts beyond the standard concerns of more efficient deep learning deployment.

    Guidelines:

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The paper releases neither pre-trained generative models nor scraped datasets. The contributions are an architectural deployment strategy and an ensemble selection method, which carry no high risk of misuse.

    Guidelines:

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

12. **Licenses for existing assets**

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

Answer: [Yes]

Justification: CIFAR-100, ResNet-56, VGG-16, MobileNetV1, SDN, and ZTW are cited in the main paper's bibliography. The released repository documents the sources and licenses or terms of use for external datasets, baseline implementations, and deployment tools. TensorRT is used under NVIDIA's standard developer terms.

Guidelines:

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

13. **New assets**

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

Answer: [Yes]

Justification: The paper releases a public code repository containing implementation and deployment scripts. The repository includes documentation for reproducing the main experiments, including data preparation, training or evaluation commands, calibration-based subset selection, and Jetson/TensorRT deployment scripts. The paper does not introduce new datasets or pre-trained generative models.

Guidelines:

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

14. **Crowdsourcing and research with human subjects**

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

Answer: [N/A]

Justification: The paper does not involve crowdsourcing or research with human subjects.

Guidelines:

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

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

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

Answer: [N/A]

Justification: The paper does not involve research with human subjects.

Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [N/A]

Justification: LLMs were used only for language polishing, grammar editing, and formatting support. They were not used in the methodology, implementation, experiments, or analysis of results; therefore, declaration is not required under the NeurIPS LLM policy.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.