PURE: Budget-Aware Update Inference from Historical Model Outputs

Anonymous Author(s)
Affiliation
Address
email

Abstract

Edge applications such as smart photo albums, local search, and OCR indexing repeatedly reprocess historical data when an on-device model is updated. Full re-inference is simple but expensive, while static uncertainty routing ignores the old predictions already stored by deployed systems. We study *budget-aware update inference*: given cached old-model output distributions, one or more candidate update models, and a limited update budget, decide which historical samples should be refreshed and which predictions can be reused. We propose PURE, a lightweight router that predicts candidate-specific entropy decrease from cached probability vectors and updates the samples with the largest predicted benefit. For sequential updates, PURE adds bounded correction for persistent stale-cache errors caused by repeatedly skipped or overconfidently wrong samples. In single-round diagnostics, PURE is competitive with strong uncertainty baselines and well above random routing; in multi-round CIFAR-100 deployment, it reaches 78.87% Round-4 accuracy, versus 77.16% for MLink and 76.90% for the no-correction ablation On TinyImageNet, a 50% update budget reduces median update latency from 8615.0ms to 4310.2ms while preserving 95.08% consistency. A smart-album case study skips 21.20–58.45% of update computation at a 95% consistency target. PURE is intended for fixed-interface updates with probability-like outputs, where cached predictions remain meaningful routing state.

1 Introduction

On-device applications increasingly store model predictions as reusable metadata. A photo album may cache faces, objects, scenes, and OCR tokens for later search; a local assistant may cache categories or attributes for private retrieval; and mobile systems often refresh such metadata only during charging or idle windows Bangash [2023], Jia et al. [2024], Barbuto et al. [2023], Liu and Guan [2024]. When the deployed model is updated, the default maintenance strategy is full re-inference over the historical collection. This is reliable, but it spends computation on samples whose outputs would remain unchanged or whose uncertainty would not improve meaningfully. Skipping the update entirely is cheap but leaves stale predictions.

We study the middle ground: *budget-aware update inference*. The system has already run an old model F_0 and cached its output distribution $p_0(x_i)$ for each historical sample x_i. Later, one or more update models become available, but the device can afford to refresh only an update rate u, leaving a skip rate $s = 1 - u$. The question is which cached samples should be re-inferred, which candidate model should process each selected sample, and how much consistency with full re-inference is retained.

This setting is different from ordinary efficient inference. Compression, quantization, pruning, sparse computation, and early-exit methods reduce the cost of each forward pass Pan et al. [2025], Kurtic

Submitted to 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Do not distribute.

et al. [2023]; selective classification and active routing decide how to handle current requests Yuan et al. [2024]. They do not directly use the old output distribution as persistent state for a later model-update job. Update inference also differs from generic multi-model scheduling: the goal is not simply to select a model for a fresh input, but to maintain a historical cache under a fixed refresh budget.

PURE is built around four deployability conditions. First, *stability preservation*: skipped predictions should remain close to the outputs of the full update model under the chosen budget. Second, *budget accounting*: router scoring, ranking, update forwards, correction forwards, and stored-output footprint must be counted. Third, *non-staleness*: sequential filtering should avoid permanently ignoring high-risk samples. Fourth, *interface validity*: cached old outputs must remain meaningful for the update task, which is most natural for fixed-label or otherwise probability-like interfaces.

Our central design choice is to predict *entropy decrease* directly. A high old-model entropy identifies uncertain samples, but it does not say which update model will help most. A binary label-change predictor is also too rigid for different update budgets. PURE instead trains a small candidate-specific regressor $h_j(p_0(x_i))$ to estimate $\Delta H_j(x_i) = H(F_0(x_i)) - H(F_j(x_i))$, then updates the samples with the largest predicted benefit. For multi-round updates, PURE adds bounded correction for two named failure modes: *persistent omission*, where samples repeatedly fall below the cutoff, and *overconfident stale error*, where a low-entropy wrong prediction appears safe. The learned score is not meant to replace entropy in every one-shot setting; its role is to make update benefit candidate-specific and to provide state for bounded multi-round maintenance.

Our contributions are:

• We formulate budget-aware update inference as a distinct deployment problem that reuses cached old-model distributions to avoid unnecessary full recomputation.

• We introduce PURE, a lightweight entropy-decrease router that supports adjustable update/skip budgets and candidate-specific model selection from historical outputs.

• We identify persistent stale-cache error in sequential updates and add bounded omission-risk and misclassification-risk correction.

• We evaluate the full deployment account: strong one-shot uncertainty baselines, multi-round correction, fixed-interface breadth, smart-album deployment, router overhead, end-to-end latency, and compressed storage.

PURE is designed specifically for settings where update-aware routing, sequential correction, and deployment accounting must be considered jointly.

2 Related Work

Efficient inference. Pruning, quantization, sparse inference, distillation, and early-exit computation reduce the cost of executing a model once Han et al. [2016], Liu et al. [2017], Jacob et al. [2018], Frantar et al. [2022], Hinton et al. [2015], Li [2025]. PURE is orthogonal to these methods. It does not change the update model itself; instead, it reduces how many historical samples need to be refreshed after the update model becomes available.

Selective prediction and input routing. Selective classification, active learning, and input filtering usually decide whether a current input should be handled by a model, abstained, or sent to another component Settles [2009], Yuan et al. [2024]. These methods often rely on current uncertainty. PURE instead uses the cached output distribution from a previous model as the feature for a later maintenance job. This makes the objective update-specific: the score should estimate how much re-inference would change or improve the stored prediction under a fixed refresh budget.

Model scheduling and update maintenance. Multi-model scheduling methods such as MLink, CEMA, adaptive scheduling, and active model selection coordinate different models under resource constraints Yuan et al. [2022], Chen et al. [2024], Yuan et al. [2020], Matsuura and Hara [2023], Sawade et al. [2012]. PURE addresses a narrower deployment phase: maintaining a historical prediction cache after model updates. The distinction matters because skipped samples can remain stale across multiple update rounds, so the router must account for both immediate update benefit and sequential correction.

Uncertainty and calibration. Entropy and confidence are useful but imperfect proxies for correctness Guo et al. [2017]. We use entropy decrease as a learnable routing target because it is available without ground-truth labels at deployment time and naturally supports candidate-specific model choice. We do not treat entropy as a per-sample correctness guarantee; the main text therefore validates the signal empirically against strong uncertainty baselines, with calibration and reliability details moved to the appendix.

3 Problem and Method

3.1 Budget-Aware Update Inference

Let $\mathcal{D} = \{x_i\}_{i=1}^m$ be a historical dataset already processed by an old model F_0. For each sample, the system stores the probability distribution $p_0(x_i) = F_0(x_i)$. When candidate update models $\{F_1, \ldots, F_n\}$ arrive, the device can refresh only an update rate $u \in [0, 1]$, so the skip rate is $s = 1 - u$. A routing policy $S : \mathcal{D} \to \{0, 1, \ldots, n\}$ assigns $S(x_i) = 0$ when the old prediction is reused and $S(x_i) = j$ when sample x_i is re-inferred by candidate model F_j. The budget constraint is

$$\frac{|\{i : S(x_i) \neq 0\}|}{m} \leq u. \tag{1}$$

We evaluate the updated cache by task accuracy and by consistency with full re-inference. For classification, if $a_i = \arg\max F_{\text{full}}(x_i)$ is the label produced by the full update model and b_i is the label stored after filtering, consistency is

$$\text{Cons} = \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}[a_i = b_i]. \tag{2}$$

For detection and other structured outputs, we use task-specific consistency operators described in Appendix C.3.

3.2 Entropy-Decrease Router

For candidate F_j, define the true entropy decrease

$$\Delta H_j(x_i) = H(F_0(x_i)) - H(F_j(x_i)). \tag{3}$$

If all candidate outputs were already known, a natural budgeted policy would update the samples with the largest positive ΔH_j, choosing the best candidate per sample. At deployment time, however, running every candidate on every historical sample would defeat the purpose of filtering. PURE therefore trains a lightweight regressor h_j for each candidate model:

$$\widehat{\Delta H}_j(x_i) = h_j(p_0(x_i)). \tag{4}$$

The input is the cached probability vector, not the raw sample. This keeps router inference small and makes the method compatible with privacy-preserving caches that store model outputs rather than original user data.

PURE chooses the candidate and benefit score

$$j_i^* = \underset{j \in \{1,\ldots,n\}}{\arg\max}\ h_j(p_0(x_i)), \qquad b_i = \max_j h_j(p_0(x_i)). \tag{5}$$

The router sorts $\{b_i\}_{i=1}^m$, updates the top $\lfloor um \rfloor$ samples using $F_{j_i^*}$, and reuses $p_0(x_i)$ for all remaining samples. This formulation separates the update budget from the predictor itself: the same scores can be used for different budgets without retraining. To illustrate the process, Figure 1 presents a flowchart of the update from ShuffleNet to ResNet with several representative samples. The Figures 2 and 3 reveal that the entropy decrease between the two models varies substantially across samples, which supports the feasibility of the routing procedure.

In our implementation, each h_j is a three-layer MLP trained with mean-squared error on held-out old/new output pairs; the basic CIFAR router is 0.02MB and trains in about 15 minutes.

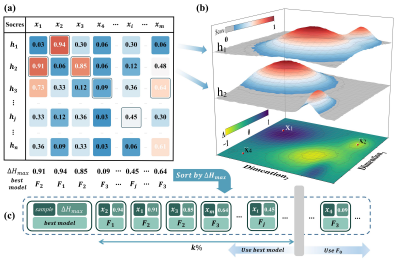

Figure 1: PURE update pipeline

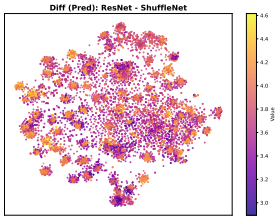

Figure 2: Predicted Entropy Decrease on t-SNE manifold

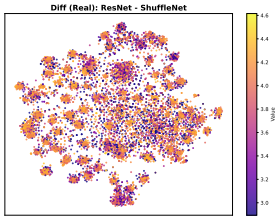

Figure 3: Real Entropy Decrease on t-SNE manifold

Algorithm 1: PURE budget-aware update inference

Require: Cached old outputs $\{p_0(x_i)\}_{i=1}^m$, candidate update models $\{F_j\}_{j=1}^n$, trained routers $\{h_j\}_{j=1}^n$, update rate u

Ensure: Updated output cache $\{\tilde{p}(x_i)\}_{i=1}^m$

for each historical sample x_i **do**
  for each candidate update model F_j **do**
    Score predicted benefit $a_{ij} \leftarrow h_j(p_0(x_i))$
  end for
  Select candidate $j_i^* \leftarrow \arg\max_j a_{ij}$
  Store update priority $b_i \leftarrow \max_j a_{ij}$
end for
Let $\mathcal{I}$ be the top $\lfloor um \rfloor$ indices by b_i
for each historical sample x_i **do**
  if $i \in \mathcal{I}$ **then**
    Refresh cache with selected update model: $\tilde{p}(x_i) \leftarrow F_{j_i^*}(x_i)$
  else
    Reuse old cache: $\tilde{p}(x_i) \leftarrow p_0(x_i)$
  end if
end for

3.3 Sequential Correction for Stale-Cache Errors

Repeated model updates create a failure mode that does not appear in one-shot filtering. A sample may repeatedly rank just below the budget cutoff, producing *persistent omission*; or a wrong low-entropy prediction may look safe, producing an *overconfident stale error*. PURE adds a bounded correction stage from the second update round onward.

At round t, PURE first computes an omission-risk score from the accumulated normalized predicted benefits since the sample's last effective update. Let

$$S_{r,i} = \frac{1}{2}\left[1 + \tanh\left(\widehat{\Delta H}_{r,i}\right)\right] \tag{6}$$

be the normalized predicted benefit for sample x_i at round r, and let $\tau(i)$ be the last round in which x_i was effectively refreshed. PURE scores stale-cache risk as

$$\mathcal{R}_{\text{omit}}(x_i) = \sum_{r=\tau(i)+1}^{t} S_{r,i} + \beta \exp\left(-\frac{S_{\tau(i),i}^2}{2\sigma^2}\right). \tag{7}$$

The first term targets persistent near-misses under the budget cutoff. The second term is a bounded stale-error proxy: it gives additional priority to samples whose last recorded benefit was too weak to rule out stagnant, confidently reused predictions. It is a heuristic risk score, not a guarantee that the stored prediction is wrong.

PURE screens only the top $\rho_{\text{screen}}^{(t)}$ fraction by $\mathcal{R}_{\text{omit}}$ as a candidate correction set $\mathcal{X}_1$. It then estimates instability on $\mathcal{X}_1$ using Jensen Shannon divergence between the currently stored distribution p_i and one alternative distribution from a recent model p_i'. In our implementation, p_i' is produced by

the highest predicted-benefit model in the current correction pool that is not already the stored model; if no such model exists, we use the most recent candidate model.

$$\mathcal{R}_{\text{misclass}}(x_i) = \frac{1}{2} D_{\text{KL}}(p_i \| m_i) + \frac{1}{2} D_{\text{KL}}(p_i' \| m_i), \quad m_i = \frac{p_i + p_i'}{2}. \tag{8}$$

Only the top $\rho_{\text{correct}}^{(t)}$ fraction of $\mathcal{X}_1$ enters multi-model voting. The voting window contains the current and recent update models, with two models in round 2 and three models in rounds 3–4 for the CIFAR-100 experiment. Ties are broken by the largest average softmax confidence among tied labels, and the stored output is the most recent voting distribution that predicts the winning label. Table 3 specifies the exact schedules and defaults. This design deliberately limits correction forwards; Table 4 reports both reuse-aware and conservative upper-bound forward counts.

3.4 Deployment Accounting

For update inference to be useful, the saved update forwards must dominate all auxiliary costs. We therefore report router scoring, sorting, correction/voting overhead, end-to-end latency, and stored-output footprint. This accounting is important because a router that only improves accuracy under an abstract update budget may still be impractical if scoring or storage costs are comparable to the model update itself.

4 Experiments

4.1 Setup and Metrics

PURE is implemented in PyTorch 2.2. Unless stated otherwise, each candidate-specific router is a three-layer MLP with ReLU activations, Adam optimization, learning rate 10^{-3}, batch size 32, and MSE loss against measured entropy decrease. We evaluate image classification, object detection, text classification, fixed-label CLIP-style classification, hardware latency, and a smart-album deployment simulation. The main metrics are task performance, consistency with full re-inference, update rate u, skip rate $s = 1 - u$, and measured wall-clock cost. Dataset, architecture, calibration, and preprocessing details are provided in Appendix B.

4.2 Single-Round Diagnostic

Table 1 evaluates cached CIFAR-100 old/new outputs against strong one-shot routers. This is a conservative diagnostic because it includes simple uncertainty baselines alongside learned routing methods. PURE is competitive with entropy, calibrated entropy, least confidence, margin, and label-change prediction, while all informed routers are well above random routing. The table also clarifies the right claim boundary: PURE should not be presented as a universal single-round replacement for entropy or margin. Its value is that the same candidate-specific ΔH signal extends to model choice, sequential correction, and deployment accounting.

Table 1: Held-out CIFAR-100 cached-output diagnostic with strong simple baselines. The router is trained on 5k cached old/new outputs and evaluated on a disjoint 5k split using dataset labels. Entries are Acc./Cons.; learned and random methods report mean ± standard deviation over seeds.

Method	**20% update / 80% skip**	**50% update / 50% skip**	**80% update / 20% skip**
Reuse old model	64.22 / –	64.22 / –	64.22 / –
Full new model	81.26 / 100.00	81.26 / 100.00	81.26 / 100.00
Entropy	71.88 / 81.02	79.96 / 95.86	81.26 / 99.84
Calibrated Entropy	71.72 / 80.66	79.94 / 95.56	81.26 / 99.84
Least Confidence	72.14 / 81.12	80.04 / 95.90	81.26 / 99.88
Margin	72.18 / 80.72	79.94 / 95.82	81.26 / 99.88
Delta Top-1	72.03±0.01 / 80.93±0.03	79.41±0.55 / 94.93±1.11	80.39±0.91 / 97.91±1.81
Label-Change Predictor	72.16±0.02 / 81.25±0.02	79.93±0.02 / 95.91±0.01	81.26 / 99.88
PURE Delta-H	71.83±0.01 / 80.97±0.02	79.94±0.02 / 95.89±0.01	81.27±0.01 / 99.83±0.01
Random	67.62±0.22 / 72.69±0.19	72.79±0.34 / 82.97±0.30	77.87±0.31 / 93.30±0.28

Table 2: Multi-Round Model Deployment Performance Comparison on CIFAR-100. The table illustrates the progressive accuracy improvement from the initial model M_0 (Round 0) through four deployment rounds. Numbers in parentheses indicate the cumulative accuracy gain relative to the M_0 baseline (60.84%).

	Progressive Accuracy (%) by Round					**Overall**	
Method	Round 0	Round 1	Round 2	Round 3	Round 4	**Avg.**	**ΔAvg.**
Naive	60.84	61.81 (+0.97)	62.37 (+1.53)	71.84 (+10.90)	76.16 (+15.32)	66.61	Ref.
Standard Baselines							
Entropy	60.84	61.80 (+0.96)	62.37 (+1.53)	71.27 (+10.43)	76.12 (+15.28)	66.48	−0.13
CEMA	60.84	61.74 (+0.90)	62.22 (+1.38)	71.24 (+10.40)	76.14 (+15.30)	66.44	−0.17
MLink	60.84	62.14 (+1.30)	62.70 (+1.86)	75.14 (+14.30)	77.16 (+16.32)	67.60	+0.99
Our Method & Ablations							
PURE (Full)	60.84	**62.36** (+1.52)	**62.78** (+1.94)	**77.53** (+16.69)	**78.87** (+18.03)	**68.48**	**+1.87**
– w/o Correction	60.84	62.05 (+1.21)	62.40 (+1.56)	75.98 (+15.14)	76.90 (+16.06)	67.62	+1.01
– Only Omission	60.84	62.24 (+1.40)	62.59 (+1.75)	76.91 (+16.07)	78.08 (+17.24)	68.12	+1.51
– Only Misclass	60.84	62.11 (+1.27)	62.47 (+1.63)	76.37 (+15.53)	77.45 (+16.61)	67.84	+1.23
– No Voting	60.84	62.17 (+1.33)	62.53 (+1.69)	76.60 (+15.76)	77.69 (+16.85)	67.95	+1.34
Optimal (Upper Bound)	60.84	62.48 (+1.64)	62.87 (+2.03)	80.68 (+19.84)	80.75 (+19.91)	69.52	+2.91

Notes: Round 0: ResNet-101 (60.84% acc); **Round 1-2:** Noise reduction; **Round 3:** Quality improvement; **Round 4:** R2-Full (80.75% acc). Update selection rate: 50% per round.

Table 3: Sequential correction protocol used in the CIFAR-100 multi-round deployment. The schedules are deterministic and activate after Round 1. The JS source is the highest predicted-benefit recent model not already stored for the sample; if none exists, we use the most recent candidate. Ties in voting are broken by the largest average softmax confidence among tied labels.

Round	$\rho^{(t)}_{\text{screen}}$	$\rho^{(t)}_{\text{correct}}$	β	σ	**JS source**	**Voting window / stored output**
2	10%	45%	1.5	0.22	Best non-stored recent model	2 models; store most recent distribution with winning label
3	12%	50%	1.5	0.22	Best non-stored recent model	3 models; store most recent distribution with winning label
4	14%	55%	1.5	0.22	Best non-stored recent model	3 models; store most recent distribution with winning label

4.3 Sequential Updates and Stale-Cache Correction

We next evaluate a progressive CIFAR-100 deployment in which each round updates 50% of samples. This setting exposes persistent stale-cache error because skipped samples can accumulate across rounds. Table 2 shows that PURE reaches 78.87% Round-4 accuracy, outperforming MLink at 77.16%, Naive selection at 76.16%, and the predictor-only variant without correction at 76.90%. We do not present this as a statistical guarantee, but as evidence that correction helps in progressive deployment.

The correction stage is also bounded. Table 4 shows that only 4.5%, 6.0%, and 7.7% of samples enter final voting in rounds 2–4. The reuse-aware extra-forward rate is 10.0–21.7% of the dataset, while a conservative no-reuse upper bound is 19.0–37.1%. The method therefore does not quietly become full re-inference, but the table makes the correction overhead explicit rather than counting only corrected samples. Appendix B.7 reports the corresponding five-seed Round-4 summary: PURE obtains $78.83 \pm 0.15\%$, while MLink obtains $77.16 \pm 0.05\%$ and the uncertainty baselines remain around 76.1–76.2%. Appendix C.4 further reports a 59-configuration sweep over the selection rate, correction penalty, confidence threshold, screening ratio, and correction ratio.

4.4 Breadth Across Fixed-Interface Updates

Table 5 reports breadth experiments on TinyImageNet, CINIC-10, AG News, and fixed-label CLIP ImageNet routing. These results support applicability beyond CIFAR. We interpret them as breadth evidence rather than a dominance claim. PURE is above random routing in all reported settings and usually close to entropy; on CLIP at 50% update, entropy slightly exceeds PURE. This is consistent with the method's scope: PURE is useful when cached probability distributions provide update-benefit information, but the margin over simple uncertainty depends on interface quality and

Table 4: Observed correction workload from the CIFAR-100 multi-round run log. Correction is activated after Round 1 and is applied only to the screened high-risk subset. Reuse-aware extra forwards count the JS alternative distribution as reusable for voting and reuse the currently stored distribution when it belongs to the voting window. The upper bound assumes voting distributions are recomputed after JS screening.

Round	Screened	JS fwds.	Corrected	Voting models	Extra fwds. reuse-aware	Extra fwds. upper	Acc.
2	1000/10000 (10%)	1000	450/10000 (4.5%)	2	1000 (10.0%)	1900 (19.0%)	62.78
3	1200/10000 (12%)	1200	600/10000 (6.0%)	3	1800 (18.0%)	3000 (30.0%)	77.53
4	1400/10000 (14%)	1400	770/10000 (7.7%)	3	2170 (21.7%)	3710 (37.1%)	78.87

Table 5: Breadth across fixed-interface update settings. We report Acc./Cons. and mean $\pm$ standard deviation over 8 seeds where stochasticity is present. Rate is shown as update/skip.

Setting	Models	Update/Skip	PURE	Entropy	Random
TinyImageNet low-gap	ResNet18 $\rightarrow$ ResNet50	20%/80%	64.04±0.06 / 80.15±0.07	63.94 / 80.37	62.13±0.25 / 71.65±0.18
		50%/50%	64.43±0.02 / 95.14±0.04	64.44 / 95.11	62.43±0.30 / 82.33±0.37
TinyImageNet high-gap	MobileNetV2 $\rightarrow$ ResNet50	20%/80%	61.10±0.03 / 77.26±0.03	61.06 / 77.25	59.61±0.22 / 69.16±0.16
		50%/50%	63.57±0.06 / 93.93±0.01	63.65 / 93.95	60.82±0.25 / 80.77±0.26
CINIC-10	ResNet18 $\rightarrow$ ResNet50	20%/80%	87.55±0.07 / 95.26±1.18	87.55 / 96.41	86.29±0.04 / 90.01±0.06
		50%/50%	87.47±0.08 / 95.47±1.24	86.81 / 99.50	86.34±0.05 / 93.78±0.06
AG News	Qwen1.5 $\rightarrow$ Qwen-larger	20%/80%	82.01±0.16 / 94.24±0.25	81.72 / 93.81	79.75±0.60 / 90.12±0.57
		50%/50%	84.49±0.08 / 97.88±0.08	84.42 / 97.80	81.77±0.43 / 93.86±0.49
CLIP ImageNet	ViT-B/32 $\rightarrow$ ViT-L/14	20%/80%	63.40±0.05 / 77.60±0.06	63.30 / 77.53	61.05±0.06 / 70.69±0.09
		50%/50%	67.97±0.11 / 91.88±0.18	68.15 / 92.26	64.48±0.10 / 81.71±0.08

calibration. Appendix C.1 provides a preliminary CIFAR-100 multi-expert setting that illustrates candidate-specific model choice among specialized flower, tree, and insect experts.

4.5 Smart-Album Deployment Case Study

We simulate a smart photo album update using 2000 private images and video frames. The tasks mirror metadata maintained by gallery systems: object detection, face detection, scene classification, and OCR-based text classification. Table 6 reports the maximum skip rate that preserves a 95% consistency target. Detection tasks use image-level IoU consistency and skip 35.72% and 21.20% of inputs; scene and OCR classification skip 58.45% and 40.25%. This is a deployment feasibility study rather than a public benchmark, and the detection results should be interpreted as image-level metadata refresh rather than object-level routing.

For the recovered YOLO object-detection subset, Table 7 compares PURE with entropy and random routing using consistency to full YOLOv8x outputs on 172 images. PURE cross-fit skips 14.19% of images at the 95% target, close to entropy at 13.95% and above random at 7.67%. The small gap is consistent with our single-round results: the benefit of PURE lies in budgeted update accounting and deployment integration, not in dominating entropy in one-shot settings.

4.6 Deployment Cost and Stored Outputs

Finally, we count the overhead that a deployable update router must pay. Table 8 shows that scoring 10k samples costs 0.365ms on an A40 GPU and 3.098ms on an x86 CPU, with sorting below 0.15ms in both cases. Table 9 reports end-to-end TinyImageNet-200 latency: at a 50% update budget, median

Table 6: Deployment case study on private mobile photo-album data. We report the maximum skip rate that satisfies a 95% consistency target. "Cons" follows Eq. 2; "IoU" denotes Intersection-over-Union.

Task	Model in Rounds		Evaluation Metrics		Filter
Description	Round1	Round2	Metric	Type	Rate
Object Detection	Yolov8nUltralytics [2023]	Yolov8xUltralytics [2023]	IoU(95%)	Detection	35.72%
Face Detection	MTCNNZhang et al. [2016]	Yolo-faceJocher and Qiu [2023]	IoU(95%)	Detection	21.20%
Scene Classification	ResnetHe et al. [2016]	Yolov8sUltralytics [2023]	Cons.(95%)	Classification	58.45%
Text Classification	Qwen1.5Bai et al. [2023]	Qwen3Team [2025]	Cons.(95%)	Classification	40.25%

Table 7: Object-detection diagnostic from recovered YOLO CSV exports (172 images). A skipped image reuses YOLOv8n detections; an updated image uses YOLOv8x detections. Consistency is symmetric class-aware image-level IoU agreement with full YOLOv8x outputs. Oracle risk uses the measured skip risk and is shown only as an upper-bound diagnostic.

Method	**Cons. @ 50% update**	**Cons. @ 80% update**	**Max skip @ 95% cons.**
Random	66.88±1.23	87.46±1.02	7.67±0.77
Entropy	80.14	92.91	13.95
Least confidence	79.53	92.91	13.95
MLink-style mapper	80.05±0.06	92.92±0.20	13.84±0.44
PURE cross-fit	79.98±0.24	93.00±0.32	14.19±0.87
Oracle risk	81.74	94.93	19.19

Table 8: Routing overhead for scoring 10k stored probability vectors. Sorting-only measures the top-level ranking cost without a router forward pass.

	PURE			Entropy			Sort only		
Platform	**Median ms**	**P95 ms**	**μs/sample**	**Median ms**	**P95 ms**	**μs/sample**	**Median ms**	**P95 ms**	**μs/sample**
A40 GPU	0.365	0.383	0.036	0.197	0.204	0.020	0.139	0.145	0.014
x86 CPU	3.098	3.213	0.310	0.270	0.280	0.027	0.131	0.135	0.013

time drops from 8615.0ms for full re-inference to 4310.2ms while preserving 95.08% consistency. PURE and entropy have similar wall-clock cost under the same update budget; the practical question is whether the learned score selects a better subset for the same cost.

Historical outputs also need storage. Table 10 spells out the fields stored by each compressed cache variant and how the router reconstructs its probability-vector input. Table 11 shows that the tested quantization and sparsification settings fluctuate by at most 0.86 percentage points at a 50% selection rate; top-1 Int4 storage can reduce the per-sample cache from 400 bytes to roughly 1.5 bytes with a packed CIFAR-100 class index, or 2.5 bytes with a conservative uint16 class index. We treat top-1 as an extreme compression ablation that should be enabled only after local validation; top-k or full quantized vectors are the safer deployment choices when storage permits.

Table 9: End-to-end update latency on TinyImageNet-200 validation (10k samples, ResNet18 $\rightarrow$ ResNet50, NVIDIA A40, 20 repeats). For reference, full re-inference (100% update) achieves 8615.0 ms median latency, 8616.6 ms P95, with 62.97% accuracy and 100.00% consistency.

		PURE			Entropy			Random		
Update u	**Skip s**	**Median ms**	**P95 ms**	**Acc./Cons.**	**Median ms**	**P95 ms**	**Acc./Cons.**	**Median ms**	**P95 ms**	**Acc./Cons.**
80%	20%	6898.0	6899.0	63.75 / 97.69	6898.0	6898.6	63.19 / 99.75	6898.3	6898.7	62.89 / 93.18
50%	50%	4310.2	4310.9	64.48 / 95.08	4310.3	4311.3	64.44 / 95.11	4309.9	4310.4	62.73 / 82.24
20%	80%	1722.5	1723.0	63.98 / 80.12	1722.2	1722.6	63.94 / 80.36	1722.0	1722.5	62.19 / 71.77

Table 10: Stored-output accounting for a 100-class cache. Sparse variants store class indices and quantized probabilities; unlisted classes are zero-filled and the vector is renormalized before router scoring. The packed top-1 setting is an extreme compression ablation rather than the default recommendation.

Cache variant	**Stored fields per sample**	**Bytes/sample**	**Router reconstruction**
Full Float32	100 probabilities, 32 bits each	400.0	Dense p_0 used directly
Full Float16	100 probabilities, 16 bits each	200.0	Dequantize and renormalize dense p_0
Full Int8	100 probabilities, 8 bits each	100.0	Dequantize and renormalize dense p_0
Full Int4	100 probabilities, 4 bits each	50.0	Dequantize and renormalize dense p_0
Top-8 Int4	8 class indices + 8 four-bit probabilities	12.0 packed / 20.0 uint16-index	Zero-fill omitted classes, dequantize top-8, renormalize
Top-4 Int4	4 class indices + 4 four-bit probabilities	6.0 packed / 10.0 uint16-index	Zero-fill omitted classes, dequantize top-4, renormalize
Top-1 Int4	1 class index + 1 four-bit probability	1.5 packed / 2.5 uint16-index	One-hot-like sparse vector with dequantized top probability

Table 11: Performance stability under different quantization and sparsification settings at a 50% Selection Rate.

	50% Selection Rate Accuracy (%)				Max
Spars.	Float32	Float16	Int8	Int4	**Fluct.**
Top-1	78.24	78.22	78.37	78.40	0.51%
Top-4	78.17	78.24	78.35	77.94	0.86%
Top-8	78.30	78.29	78.26	78.04	0.83%
Full (Ref)	78.32	78.40	78.18	77.92	0.86%
Avg.	**78.26**	**78.29**	**78.29**	**78.08**	–

5 Limitations

PURE assumes that the old model's output distribution contains useful information about which samples benefit from an update. This assumption is reasonable for progressive model updates and is supported by our cross-architecture experiments, but it can fail when the old and new systems solve substantially different tasks, use incompatible label spaces, or when the old model is poorly calibrated. In such cases the historical distribution may no longer be a reliable routing feature, and a small validation set should be used to decide whether routing is appropriate.

The router also depends on probability-like outputs. Standard classifiers expose such outputs directly, and fixed-label CLIP-style models can be adapted by applying a softmax over prompt-bank logits. Embedding-only systems, open-ended generative systems without constrained answer tokens, or dense prediction tasks require task-specific uncertainty proxies. For object detection we use image-level confidence aggregation; this is practical for smart-album updates but may be too coarse for applications that require object-level update decisions.

PURE is budget-aware rather than traversal-guaranteed. Under a strict filtering budget, some samples may be skipped for many rounds. Our omission-risk and misclassification-risk correction stage reduces this problem, but it does not guarantee that every sample is reprocessed after a fixed number of updates. Applications requiring certified full refreshes should schedule periodic full re-inference or reserve a minimum traversal budget.

Finally, severe deployment shift remains a limitation. Because PURE operates in probability space rather than raw input space, it is less exposed to low-level image shifts than pixel-based routers, and the data-efficiency ablation suggests that small local calibration sets can be effective. Nevertheless, if private user data differs sharply from the developer-side data, local router adaptation or fallback to conservative filtering is necessary.

6 Conclusion

We introduced PURE, a lightweight update-aware inference router that reuses historical model outputs to avoid unnecessary re-inference after model updates. By predicting candidate-specific entropy decrease from stored probability distributions, PURE converts a fixed update budget into a sample- and model-level routing policy. Across single-round, multi-round, cross-modal, hardware, and smart-album experiments, PURE provides a practical efficiency-consistency trade-off while keeping router training and storage overhead small.

Impact Statement

This work aims to reduce computation, latency, and energy consumption for repeated model updates on edge devices. Potential positive impacts include lower battery use, lower carbon footprint from redundant inference, and faster refresh of on-device metadata in applications such as photo organization and local search. Potential risks come from deploying stale predictions when the router filters too aggressively, especially in user-facing systems where outdated metadata may affect retrieval, accessibility, or downstream automated decisions. We therefore recommend exposing filtering budgets as deployment controls, monitoring consistency on a validation stream, and using

conservative or full-refresh modes in safety-critical applications. The smart-album data used for deployment simulation is not released and is reported only in aggregate to reduce privacy risk.

References

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

Abdul Ali Bangash. Cost-effective strategies for building energy efficient mobile applications. In *International Conference on Software Engineering*, 2023.

Vincenzo Barbuto, Claudio Savaglio, Roberto Minerva, Noel Crespi, and Giancarlo Fortino. Towards an edge intelligence-based traffic monitoring system. *2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, abs/2403.12976, 2023.

Moses Charikar, Kevin C. Chen, and Martin Farach-Colton. Finding frequent items in data streams. In *ICALP 2002: Proceedings of the 29th International Colloquium on Automata, Languages and Programming*, pages 693–703, London, UK, 2002. Springer-Verlag. ISBN 3-540-43864-5. doi: 10.1007/3-540-45465-9_59.

Yaofo Chen, Shuaicheng Niu, Yaowei Wang, Shoukai Xu, Hengjie Song, and Mingkui Tan. Towards robust and efficient cloud-edge elastic model adaptation via selective entropy distillation. In *International Conference on Learning Representations*, 2024.

Graham Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005. doi: 10.1016/j.jalgor.2003.12.001.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *Computing Research Repository*, 2022.

Sebastien Gerchinovitz, Pierre Menard, and Gilles Stoltz. Fano's inequality for random variables. *Statistical Science*, 35(2), 2020.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks, 2017. URL `https://arxiv.org/abs/1706.04599`.

Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *International Conference on Learning Representations*, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *NeurIPS Workshop*, 2015.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. *International Conference on Computer Vision*, 2018.

Yongzhe Jia, Bowen Liu, Xuyun Zhang, Fei Dai, Arif Khan, Lianyong Qi, and Wanchun Dou. Model pruning-enabled federated split learning for resource-constrained devices in artificial intelligence empowered edge computing environment. *ACM Transactions on Sensor Networks*, 2024.

Xin Jin, Hongqiang Harry Liu, Rohan Gandhi, Srikanth Kandula, Ratul Mahajan, Ming Zhang, Jennifer Rexford, and Roger Wattenhofer. Dynamic scheduling of network updates. In *Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 539–550, 2014.

Glenn Jocher and Jing Qiu. YOLO by Ultralytics, January 2023. URL `https://github.com/ultralytics/ultralytics`.

A Krizhevsky and G Hinton. Learning multiple layers of features from tiny images. *google*, 2009.

Eldar Kurtic, Denis Kuznedelev, Elias Frantar, Michael Goin, and Dan Alistarh. Sparse fine-tuning for inference acceleration of large language models. *CoRR*, abs/2310.06927, 2023.

Lei Li, Yankai Lin, Shuhuai Ren, Peng Li, Jie Zhou, and Xu Sun. Dynamic knowledge distillation for pre-trained language models. In *Conference on Empirical Methods in Natural Language Processing*, 2021.

X. et al. Li. A survey on model compression for large language models. *Journal of Machine Learning Research*, 2025.

Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In David J. Fleet, Tomás Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V*, Lecture Notes in Computer Science, pages 740–755. Springer, 2014. doi: 10.1007/978-3-319-10602-1_48. URL `https://doi.org/10.1007/978-3-319-10602-1_48`.

Shufan Liu and Shanyue Guan. Edge computing-based imagery data preprocessing strategy. *HEALTH MONITORING OF STRUCTURAL AND BIOLOGICAL SYSTEMS XVIII*, 12951, 2024.

Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *IEEE International Conference on Computer Vision*, 2017.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, 2011. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P11-1015`.

Mitsuru Matsuura and Satoshi Hara. Active model selection: A variance minimization approach. In *NeurIPS Workshop on Adaptive Experimental Design and Active Learning in the Real World*, 2023.

Rui Pan, Yinwei Dai, Zhihao Zhang, Gabriele Oliaro, Zhihao Jia, and Ravi Netravali. Specreason: Fast and accurate inference-time compute via speculative reasoning. 2025.

Yi Ren and Danica J. Sutherland. Learning dynamics of llm finetuning, 2025. URL `https://arxiv.org/abs/2407.10490`.

Christoph Sawade, Niels Landwehr, and Tobias Scheffer. Active comparison of prediction models. In *Conference on Neural Information Processing Systems*, pages 1763–1771, 2012.

Burr Settles. Active learning literature survey. *University of Wisconsin-Madison*, 2009.

Qwen Team. Qwen3, 2025. URL `https://qwenlm.github.io/blog/qwen3/`.

Ultralytics. Yolov8 documentation. `https://docs.ultralytics.com`, 2023.

Mu Yuan, Lan Zhang, Xiang-Yang Li, and Hui Xiong. Comprehensive and efficient data labeling via adaptive model scheduling. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pages 1858–1861, 2020. doi: 10.1109/ICDE48307.2020.00188.

Mu Yuan, Lan Zhang, and Xiang-Yang Li. Mlink: Linking black-box models for collaborative multi-model inference. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(9): 9475–9483, 2022. doi: 10.1609/aaai.v36i9.21180. URL https://ojs.aaai.org/index.php/AAAI/article/view/21180.

Mu Yuan, Lan Zhang, Fengxiang He, Xueting Tong, Miao-Hui Song, Zhengyuan Xu, and Xiang-Yang Li. Infi: End-to-end learning to filter input for resource-efficiency in mobile-centric inference. *IEEE TRANSACTIONS ON MOBILE COMPUTING*, 23(5), 2024.

Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503, 2016.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28, 2015.

Hang Zhu, Kostis Kaffes, Zixu Chen, Zhenming Liu, Christos Kozyrakis, Ion Stoica, and Xin Jin. Racksched: A microsecond-scale scheduler for rack-scale computers. *Computing Research Repository*, pages 1225–1240, 2020.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes].

   Justification: The abstract and Introduction state the update-aware filtering setting, the entropy-decrease router, the multi-round correction mechanism, and the evaluated domains. The Limitations section states the main assumptions and failure modes.

2. **Limitations**

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

   Answer: [Yes].

   Justification: Section 5 discusses progressive-update assumptions, probability-output requirements, no fixed traversal guarantee, and severe deployment shift.

3. **Theory assumptions and proofs**

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

   Answer: [Yes].

   Justification: The main text gives the entropy objective and Fano-based motivation; the appendix provides the learning-dynamics derivation, optimizer assumptions, and detailed proof steps.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all information needed to reproduce the main experimental results to the extent that it affects the main claims and conclusions?

   Answer: [Yes].

   Justification: The Experiments section and appendix specify datasets, model families, router architecture, optimizer, training epochs, batch size, filtering rates, metrics, hardware, and baseline adaptations.

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?

   Answer: [No].

Justification: The paper uses public datasets for most experiments and describes the procedure, but the current anonymous submission does not yet include a public code release. The smart-album deployment data is private and reported only in aggregate for privacy reasons.

6. **Experimental setting/details**

   Question: Does the paper specify all training and test details necessary to understand the results?

   Answer: [Yes].

   Justification: Section 4.1 and the appendix describe dataset splits, model update pairs, router training, baseline implementations, filtering budgets, consistency metrics, and task-specific adaptations.

7. **Experiment statistical significance**

   Question: Does the paper report error bars or other appropriate information about statistical significance?

   Answer: [Yes].

   Justification: The expanded evaluation table reports mean and standard deviation over 8 seeds for CINIC-10, AG News, and CLIP experiments, and supplementary tables report repeated-run variability where available.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources needed to reproduce the experiments?

   Answer: [Yes].

   Justification: The main text reports the A40 GPU setup for router experiments, CPU/GPU latency micro-benchmarks, and heterogeneous hardware platforms; the appendix gives mobile-emulation details.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics?

   Answer: [Yes].

   Justification: The work uses standard public datasets and aggregate private deployment statistics, preserves anonymity, and does not release personal smart-album data.

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 Impact Statement discusses energy and latency benefits, stale-prediction risks, deployment controls, validation monitoring, and privacy handling for deployment data.

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?

    Answer: [N/A].

    Justification: The paper does not release new high-risk models or the private smart-album dataset. Private deployment data is summarized only in aggregate.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets used in the paper properly credited and are license and terms of use explicitly mentioned and respected?

    Answer: [Yes].

    Justification: The paper cites the public datasets, model families, and software systems used in the experiments. License details should be included with any eventual code release.

13. **New assets**

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

    Answer: [N/A].

    Justification: The paper does not release a new dataset or benchmark asset. The proposed router is a method evaluated on existing datasets and a private aggregate deployment study.

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 details about compensation?

    Answer: [N/A].

    Justification: The paper does not include crowdsourcing or human-subject experiments.

15. **Institutional review board approvals or equivalent**

    Question: Does the paper describe potential risks incurred by study participants and whether IRB approvals or equivalent review were obtained?

    Answer: [N/A].

    Justification: The paper does not include human-subject research. The private smart-album data is used only for aggregate system evaluation and is not released.

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?

    Answer: [N/A].

    Justification: LLMs are used only as evaluated model backbones in text-classification update scenarios, not as a non-standard component of the proposed routing method.

Appendix Contents

A Motivation: A Learning Dynamics Perspective

A.1 Confidence as an Indicator of Accuracy

In this section, we analyze how the training data influences the outputs of the classifier for an arbitrary sample.

For an arbitrary sample $\mathbf{x_o}$, we decompose the training process to examine, on a sample-by-sample basis, how a single training step affects the parameter updates. Specifically, consider a training instance $(\mathbf{x_u}, y_u)$ used in the $t-th$ step, where y_u is the class label for $\mathbf{x_u}$. Assuming this step updates the model parameters from θ^t to θ^{t+1}, thereby updating the output probabilities of $\mathbf{x_o}$ from ${\mathbf{p_o}}^t$ to ${\mathbf{p_o}}^{t+1}$, we propose the following proposition (see the next section for detailed formulation and proofs) :

$$ {\mathbf{p_o}}^{t+1} - {\mathbf{p_o}}^t = -\eta \mathcal{M}^t(\mathbf{x}_o)\, \mathcal{K}^t(\mathbf{x}_o, \mathbf{x}_u)\, \mathcal{G}^t(\mathbf{x}_u, y_u) \tag{9} $$

Here, $\mathcal{G}^t$ represents the gradient driver from the training sample, $\mathcal{M}^t$ reflects the sensitivity of $\mathbf{x_o}$, and $\mathcal{K}^t$ acts as a kernel measuring the gradient alignment between $\mathbf{x_u}$ and $\mathbf{x_o}$.

Based on this mechanism—and considering the property of high-dimensional spaces where vectors tend to be nearly orthogonal unless highly correlated—the influence of a training instance $(\mathbf{x_u}, y_u)$ on an arbitrary sample $\mathbf{x_o}$ can be categorized into three scenarios:

- **Positive Reinforcement:** If the feature gradients of $\mathbf{x_u}$ and $\mathbf{x_o}$ are highly aligned, and y_u is indeed the ground-truth label of $\mathbf{x_o}$, then training on $(\mathbf{x_u}, y_u)$ will significantly increase the model's confidence in correctly classifying $\mathbf{x_o}$.

- **Negative Interference:** If the gradients are highly aligned but y_u is *not* the true label of $\mathbf{x_o}$, training on $(\mathbf{x_u}, y_u)$ will mislead the model. Specifically, it erroneously boosts the probability of the incorrect label y_u, thereby suppressing the probability of the correct class.

- **Independence:** If the gradients of x_u and x_o are nearly orthogonal in the feature space, the training of $(\mathbf{x_u}, y_u)$ exerts negligible influence on x_o.

The model's final classification probability for $\mathbf{x_o}$ is the result of cumulative updates across training steps. Consequently, if the selected training dataset contains a prevalence of samples that induce negative interference, the probability assigned to the correct label of $\mathbf{x_o}$ will be diminished. This accumulation of adverse effects ultimately leads to a reduction in the model's confidence regarding its prediction.

A.2 Detailed Formulation and Proofs

The analysis of learning dynamics presented in the previous section, which examines how training data influences the outputs of a classifier for an arbitrary sample, draws inspiration from Ren and Sutherland [2025]. However, this decomposition is intrinsically contingent upon the specific parameter update mechanism and is thus dependent on the choice of optimizer. Given that diverse classifier models often employ distinct optimizers, it is necessary to establish the generalizability of our formulation. The objective of this appendix is to demonstrate that the decomposition proposed in the main text remains valid across a spectrum of common optimizers, including SGD, SGD with Momentum, and Adam. While this validation is essential for theoretical rigor, the complete proofs are detailed here due to space constraints in the main text.

Specifically, we focus on the change in the output at an arbitrary test point x_o, denoted as $\Delta \mathbf{p}_0 = \mathbf{f}_{\theta^{t+1}}(\mathbf{x}_o) - \mathbf{f}_{\theta^t}(\mathbf{x}_o)$, resulting from a parameter update based on a single training sample $(\mathbf{x}_u, y_u)$. We decompose this change into the sum of a first-order term (the MKG decomposition term) and a second-order remainder. The core observation is that, for the optimizers considered here, the norm of the parameter update $\Delta\theta$ is first order with respect to the learning rate η. Consequently, the second-order term in the Taylor expansion becomes $O(\eta^2)$. Under small learning-rate conditions, this term is higher order relative to the first-order term, supporting the approximation used as motivation rather than as a per-sample guarantee.

A.2.1 Notation and Fundamental Assumptions

We begin by defining the network architecture, key variables, and the fundamental assumptions underlying the analysis.

Consider a neural network $\mathbf{f}_\theta : \mathbb{R}^l \to \mathbb{R}^k$ parameterized by θ. It can be generically decomposed into a feature extractor $h_\theta : \mathbb{R}^l \to \mathbb{R}^k$ and an output layer function $\pi : \mathbb{R}^k \to \mathbb{R}^k$, i.e., $\mathbf{f}_\theta(\mathbf{x}) = \pi(h_\theta(\mathbf{x}))$. The loss for a sample $(\mathbf{x}_u, y_u)$ is defined as $\mathcal{L}(\mathbf{x}_u, y_u) = \ell(h_\theta(\mathbf{x}_u), y_u)$, where ℓ is differentiable. At training iteration t, we define the following key Jacobian matrices and gradient vector:

- $\mathcal{M}^t(\mathbf{x}) = \nabla_z \pi(z)|_{h_{\theta^t}(\mathbf{x})} \in \mathbb{R}^{k \times k}$: the Jacobian of the output layer with respect to the features $h_{\theta^t}(\mathbf{x})$.
- $\mathbf{J}^t(\mathbf{x}) = \nabla_\theta h_\theta(\mathbf{x})|_{\theta^t} \in \mathbb{R}^{k \times d}$: the Jacobian of the feature extractor with respect to the parameters θ^t.
- $\mathcal{G}^t(\mathbf{x}_u, y_u) = (\nabla_{\mathbf{z}} \ell(\mathbf{z}, y_u)|_{h_{\theta^t}(\mathbf{x}_u)})^\top \in \mathbb{R}^{k \times 1}$: the gradient of the loss with respect to the features.

The output change and the parameter update at the test point are

$$\Delta \mathbf{p}_o = \mathbf{f}_{\theta^{t+1}}(\mathbf{x}_o) - \mathbf{f}_{\theta^t}(\mathbf{x}_0) \in \mathbb{R}^{k \times 1}$$

$$\Delta \theta = \theta^{t+1} - \theta^t \in \mathbb{R}^{d \times 1}$$

For the proof to hold, we introduce the following assumptions, which are reasonable and commonly satisfied in the context of training small, fully-connected networks:

- **Bounded Gradients and Jacobians:** There exists a constant $G, J, M > 0$ such that $\|\mathcal{G}^t(x_u, y_u)\|_2 \leq G, \|\mathbf{J}^t(x)\|_2 \leq J, \|\mathcal{M}^t(x)\|_2 \leq M$. Training loss is typically finite, and common activation functions and loss functions have bounded gradients within a finite input range. The limited parameter scale in small networks further reduces the likelihood of gradient explosion. Within a finite parameter space and with bounded activation functions, the spectral norm of the network's Jacobian matrix can be effectively controlled.
- **Bounded Hessian:** There exists a constant $B > 0$ such that $\|\nabla^2_\theta f_\theta(x)\|_2 \leq B$. For small networks using piecewise linear (e.g., ReLU) or smooth activation functions, the second derivatives are bounded in most regions or almost everywhere. This is an assumption common in theoretical analyses.
- **Non-Degeneracy:** This assumption requires that the training sample (x_u, y_u) induces a non-trivial update on the output of test point x_0 via the Neural Tangent Kernel (NTK)-like structure, which is reasonable for non-trivial learning tasks.
- **Bounded Adam Preconditioning Matrix:** For the Adam optimizer, the spectral norm of its preconditioning matrix $D^{t+1} = \text{diag}\left((\sqrt{\hat{v}^{t+1}} + \epsilon)^{-1}\right)$ is bounded; i.e., there exists $D_{\max} > 0$ such that $\|D^{t+1}\|_2 \leq D_{\max}$. Given bounded gradients, the components of the second-moment estimate $\hat{v}^{t+1}$ are confined to a positive interval. Combined with the stability term $\epsilon > 0$, this assumption holds naturally in practice.

A.2.2 Standard SGD

The update rule for standard SGD is

$$\Delta \theta = -\eta \left(\nabla_\theta \mathcal{L}(\mathbf{x}_u, y_u)|_{\theta^t}\right)^\top$$

Performing a Taylor expansion of $\Delta \mathbf{p}_o$ around θ^t, there exists some $\tilde{\theta} \in [\theta^t, \theta^{t+1}]$ such that:

$$\Delta \mathbf{p}_o = \underbrace{\nabla_\theta \mathbf{f}_{\theta^t}(\mathbf{x}_o) \Delta \theta}_{T_1} + \underbrace{\frac{1}{2} \Delta \theta^\top \nabla^2_\theta \mathbf{f}_{\tilde{\theta}}(\mathbf{x}_o) \Delta \theta}_{T_2}.$$

To evaluate the leading term, we plug in the definition of SGD and repeatedly use the chain rule:

$$
\begin{aligned}
T_1 &= \underbrace{\nabla_\theta \mathbf{f}_\theta\left(\mathbf{x_o}\right)|_{\theta^t}}_{k\times d}\underbrace{\left(\theta^{t+1}-\theta^t\right)}_{d\times 1} \\
&= (\underbrace{\nabla_{\mathbf{z}}\pi\left(\mathbf{z}\right)|_{\mathrm{h}_{\theta^t}(\mathbf{x}_o)}}_{k\times k}\underbrace{\nabla_\theta \mathbf{h}_\theta\left(\mathbf{x}_o\right)|_{\theta^t}}_{k\times d})(-\eta\underbrace{\nabla_\theta \mathcal{L}\left(\mathbf{x}_u, y_u\right)|_{\theta^t}}_{1\times d})^\top \\
&= \underbrace{\nabla_{\mathbf{z}}\pi\left(\mathbf{z}\right)|_{\mathrm{h}_{\theta^t}(\mathbf{x}_o)}}_{k\times k}\underbrace{\nabla_\theta \mathbf{h}_\theta\left(\mathbf{x}_o\right)|_{\theta^t}}_{k\times d}\underbrace{(-\eta\nabla_{\mathbf{z}}\mathcal{L}\left(\mathbf{x}_u, y_u\right)|_{\mathrm{h}_{\theta^t}(\mathbf{x}_o)}}_{1\times k}\underbrace{\nabla_\theta \mathbf{h}_\theta\left(\mathbf{x}_u\right)|_{\theta^t}}_{k\times d})^\top \\
&= -\eta\underbrace{\nabla_{\mathbf{z}}\pi\left(\mathbf{z}\right)|_{\mathrm{h}_{\theta^t}(\mathbf{x}_o)}}_{k\times k}[\underbrace{\nabla_\theta \mathbf{h}_\theta\left(\mathbf{x}_o\right)|_{\theta^t}}_{k\times d}\underbrace{\left(\nabla_\theta \mathbf{h}_\theta\left(\mathbf{x}_u\right)|_{\theta^t}\right)^\top}_{d\times k}]\underbrace{\left(\nabla_{\mathbf{z}}\mathcal{L}\left(\mathbf{x}_u, y_u\right)|_{\mathrm{h}_{\theta^t}(\mathbf{x}_0)}\right)^\top}_{k\times 1} \\
&= -\eta\mathcal{M}^t(\mathbf{x}_o)\mathbf{J}^t(\mathbf{x}_o)\mathbf{J}^t(\mathbf{x}_u)^\top\mathcal{G}^t(\mathbf{x}_u, y_u) = -\eta\mathcal{M}^t\left(\mathbf{x}_o\right)\mathcal{K}^t\left(\mathbf{x}_o, \mathbf{x}_u\right)\mathcal{G}^t\left(\mathbf{x}_u, y_u\right)
\end{aligned}
$$

where $\mathcal{K}^t\left(\mathbf{x}_o, \mathbf{x}_u\right) = \mathbf{J}^t(\mathbf{x}_o)\mathbf{J}^t(\mathbf{x}_u)^\top$ is the neural tangent kernel (NTK) matrix in the feature space at step t.

For the higher-order term, under our assumptions, we note that:

$$
\begin{aligned}
\|\Delta\theta\|_2 &= \left\|-\eta\left(\nabla_\theta\mathcal{L}(\mathbf{x}_u, y_u)|_{\theta^t}\right)^\top\right\|_2 = \left\|-\eta(\nabla_{\mathbf{z}}\mathcal{L}(\mathbf{x}_u, y_u)|_{\mathrm{h}_{\theta^t}(\mathrm{x}_0)}\nabla_\theta\mathbf{h}_\theta\left(\mathbf{x}_u\right)|_{\theta^t})^\top\right\|_2 \\
&= \left\|-\eta\mathbf{J}^t(\mathbf{x}_u)^\top\mathcal{G}^t(\mathbf{x}_u, y_u)\right\|_2 \leq \eta\left\|\mathbf{J}^t(\mathbf{x}_u)^\top\right\|_2\left\|\mathcal{G}^t(\mathbf{x}_u, y_u)\right\|_2 = \eta JG
\end{aligned}
$$

So the second-term T_2 satisfies:

$$
\|T_2\|_2 = \left\|\frac{1}{2}\Delta\theta^\top\nabla_\theta^2\mathbf{f}_{\tilde{\theta}}(\mathbf{x}_o)\Delta\theta\right\|_2 \leq \frac{1}{2}\left\|\nabla_\theta^2\mathbf{f}_{\tilde{\theta}}(\mathbf{x}_o)\right\|_2\|\Delta\theta\|_2^2 \leq \frac{1}{2}BJ^2G^2\eta^2
$$

Therefore, for the MKG decomposition of standard SGD, under the commonly used small learning rate condition, the second-order term T_2 is a negligible higher-order term compared to the first-order term T_1.

A.2.3 SGD with Momentum

The update rule for SGD with momentum is:

$$
\begin{aligned}
v^{t+1} &= \gamma v^t + \left(\nabla_\theta\mathcal{L}(\mathbf{x}_u, y_u)|_{\theta^t}\right)^\top \\
\Delta\theta &= -\eta v^{t+1}
\end{aligned}
$$

where $\gamma \in [0, 1)$ is the momentum coefficient and $v^0 = 0$. In this case, $\Delta\theta$ can be written as:

$$
\Delta\theta = -\eta\sum_{i=0}^{t}\gamma^{t-i}\mathbf{J}^i(x_u)^\top\mathcal{G}^i(x_u, y_u)
$$

Substituting into the expression for T_1 gives:

$$
\begin{aligned}
T_1 &= \underbrace{\nabla_\theta \mathbf{f}_\theta\left(\mathbf{x_o}\right)|_{\theta^t}}_{k\times d}\underbrace{\left(\theta^{t+1}-\theta^t\right)}_{d\times 1} \\
&= \left(\underbrace{\nabla_{\mathbf{z}}\pi\left(\mathbf{z}\right)|_{\mathrm{h}_{\theta^t}(\mathbf{x}_o)}}_{k\times k}\underbrace{\nabla_\theta \mathbf{h}_\theta\left(\mathbf{x}_o\right)|_{\theta^t}}_{k\times d}\right)\underbrace{\Delta\theta}_{d\times 1} \\
&= -\eta\mathcal{M}^t(\mathbf{x}_o)\mathbf{J}^t(\mathbf{x}_o)\sum_{i=0}^{t}\gamma^{t-i}\mathbf{J}^i(\mathbf{x}_u)^\top\mathcal{G}^i(\mathbf{x}_u, y_u) \\
&= -\eta\mathcal{M}^t(\mathbf{x}_o)\sum_{i=0}^{t}\gamma^{t-i}\mathcal{K}^{t,i}(\mathbf{x}_o, \mathbf{x}_u)\mathcal{G}^i(\mathbf{x}_u, y_u)
\end{aligned}
$$

where $\mathcal{K}^{t,i}(\mathbf{x}_o, \mathbf{x}_u) = \mathbf{J}^t(\mathbf{x}_o)\mathbf{J}^i(\mathbf{x}_u)^\top$. In this case, the norm of the parameter update satisfies:

$$\begin{aligned}\|\Delta\theta\|_2 &= \left\| -\eta \sum_{i=0}^{t} \gamma^{t-i} \mathbf{J}^i(\mathbf{x}_u)^\top \mathcal{G}^i(\mathbf{x}_u, y_u) \right\|_2 \\ &\leq \eta \sum_{i=0}^{t} \gamma^{t-i} \left\|\mathbf{J}^i(\mathbf{x}_u)^\top\right\|_2 \left\|\mathcal{G}^i(\mathbf{x}_u, y_u)\right\|_2 \\ &\leq \eta J G \sum_{i=0}^{t} \gamma^{t-i} \leq \frac{\eta J G}{1-\gamma}\end{aligned}$$

Similar to the analysis for standard SGD, the spectral norm of the second-order term in this scenario can be estimated as:

$$\|T_2\|_2 \leq \frac{1}{2} B J^2 G^2 \left(\frac{\eta}{1-\gamma}\right)^2$$

Therefore, for the second-order term to be negligible, we require $\frac{\eta}{1-\gamma} \to 0$. This implies that the momentum coefficient γ should not be excessively close to 1, or the learning rate η needs to be correspondingly reduced as $(1-\gamma)$ decreases, which are common constraints in practical hyperparameter tuning.

A.2.4 Adam Optimizer

The update steps for the Adam optimizer are as follows: Compute the gradient

$$g^t = (\nabla_\theta \mathcal{L}(\mathbf{x}_u, y_u)|_{\theta^t})^\top = \mathbf{J}^t(\mathbf{x}_u)^\top \mathcal{G}^t(\mathbf{x}_u, y_u)$$

Update the first- and second-moment estimates

$$\begin{aligned} m^{t+1} &= \beta_1 m^t + (1-\beta_1) g^t \\ v^{t+1} &= \beta_2 v^t + (1-\beta_2)(g^t \odot g^t) \end{aligned}$$

Compute bias-corrected estimates

$$\begin{aligned} \hat{m}^{t+1} &= m^{t+1}/(1-\beta_1^{t+1}) \\ \hat{v}^{t+1} &= v^{t+1}/(1-\beta_2^{t+1}) \end{aligned}$$

and finally, update the parameters:

$$\Delta\theta = -\eta \cdot \hat{m}^{t+1}/(\sqrt{\hat{v}^{t+1}} + \epsilon)$$

We define the diagonal preconditioning matrix $D^{t+1} = \text{diag}\left((\sqrt{\hat{v}^{t+1}} + \epsilon)^{-1}\right)$. Then the update can be written as:

$$\Delta\theta = -\eta \frac{1-\beta_1}{1-\beta_1^{t+1}} D^{t+1} \sum_{i=0}^{t} \beta_1^{t-i} \mathbf{J}^i(\mathbf{x}_u)^\top \mathcal{G}^i(\mathbf{x}_u, y_u)$$

Substituting into the expression for T_1 gives:

$$T_1 = \mathcal{M}^t(\mathbf{x}_o)\mathbf{J}^t(\mathbf{x}_o)\Delta\theta = -\eta \frac{1-\beta_1}{1-\beta_1^{t+1}} \mathcal{M}^t(\mathbf{x}_o) \sum_{i=0}^{t} \beta_1^{t-i} \mathcal{K}^{t,i}_{\text{Adam}}(\mathbf{x}_o, x_u) \mathcal{G}^i(\mathbf{x}_u, y_u)$$

where $\mathcal{K}^{t,i}_{\text{Adam}}(\mathbf{x}_o, \mathbf{x}_u) = \mathbf{J}^t(\mathbf{x}_o) D^{t+1} \mathbf{J}^i(\mathbf{x}_u)^\top$ is the kernel function modified by the Adam preconditioning matrix. The parameter update can be written as $\Delta\theta = -\eta D^{t+1} \hat{m}^{t+1}$. At the same time, under our assumption (Bounded Gradients and Jacobians), the $\hat{m}$ has an upper bound, denoted as M_m, we can obtain:

$$\|\Delta\theta\|_2 \leq \eta D_{max} M_m$$

So the spectral norm of the second-order term in this scenario can be estimated as:

$$\|T_2\|_2 \leq \frac{1}{2} B \left(D_{max} M_m\right)^2 \eta^2$$

Thus, for the Adam optimizer, the second-order term also becomes a negligible higher-order infinitesimal as the learning rate $\eta \to 0$.

We provide a rigorous proof based on the Neural Tangent Kernel (NTK) regime. Intuitively, this holds because edge-side model updates typically operate in a "fine-tuning" regime rather than learning from scratch.

In this regime, the parameter updates $\Delta\theta$ are small. Geometrically, this means the model stays within a small region of the loss landscape where the curvature (Hessian) is relatively constant. Consequently, the change in function output can be well-approximated by the linear term of the Taylor expansion (the gradient projection). This explains why our simple MLP-based router can accurately predict ΔH without needing to model complex second-order training dynamics.

Our analysis above demonstrates that in small, fully-connected networks commonly used in practice—typically satisfying bounded gradient, Jacobian, and Hessian matrices—the change in model output induced by training on a single sample is dominated by its first-order MKG term, provided that a sufficiently small learning rate η is adopted for SGD, Adam, and momentum SGD with a properly configured momentum coefficient γ. In particular, for small networks deployed on edge devices, the learning rates we employ readily satisfy the condition that η is much smaller than 1, thereby ensuring that the influence of second-order terms is negligible and rendering first-order MKG-based analysis highly accurate.

B Detailed Experimental Setup

B.1 Dataset Specifications

To comprehensively validate the proposed uncertainty-based filtering framework, we employ a diverse set of benchmarks covering computer vision and natural language processing.

Image Classification (CIFAR & ImageNet) We use the CIFAR-10 and CIFAR-100 datasets Krizhevsky and Hinton [2009], which consist of 32×32 color images. CIFAR-10 contains 10 classes with 50,000 training and 10,000 testing images. CIFAR-100 contains 100 classes with the same total volume. For large-scale evaluation, we utilize ImageNet-1K (ILSVRC 2012), which contains 1.28 million training images and 50,000 validation images across 1,000 categories. These datasets allow us to evaluate the router's performance from low-resolution coarse-grained labels to high-resolution fine-grained classification.

Object Detection (MS COCO) The MS COCO 2017 dataset Lin et al. [2014] is used to evaluate the framework's extensibility to localization tasks. It contains 118k training and 5k validation images (val2017) with 80 object categories. Unlike classification, COCO requires the router to handle multiple bounding box outputs per image. We utilize the validation set to simulate the re-inference process during model updates (e.g., from YOLOv8n to YOLOv8x), measuring performance via mAP@50 and consistency via IoU.

Text Classification (AG News & IMDb) In the NLP domain, we employ AG News and IMDb. The AG News dataset Zhang et al. [2015] is a four-class news classification corpus containing 120,000 training and 7,600 testing samples. The IMDb dataset Maas et al. [2011] is a benchmark for binary sentiment analysis (positive/negative) consisting of 50,000 movie reviews, split equally into 25,000 training and 25,000 testing samples. For these datasets, we evaluate both discriminative models (BERT) and generative models (Qwen series) using the adaptation strategies described in Appendix C.3.

B.2 Experimental Setup and Reliability Analysis

Model Selection and Ensemble Strategy We evaluated the framework using diverse convolutional neural networks, including standard architectures (ResNet-50, ResNet-101, VGG-16 BN) and lightweight models (MobileNetV2, ShuffleNetV2). To establish a high-performance upper bound, we implemented a **Multi-Model Ensemble** that dynamically selects the prediction from the base learner with the **highest confidence score** for each input sample, ensuring the final output is derived from the most certain predictor.

Uncertainty Quantification and Calibration Prediction reliability is quantified via a normalized confidence score $C(x) = 1 - H(x)/\max(H)$, where $H(x)$ is the Shannon entropy. To mitigate the overconfidence of deep networks, we applied **Temperature Scaling**. The optimal temperature T is determined by minimizing the Negative Log Likelihood (NLL) on a held-out calibration set (10% of the training data) using the **L-BFGS** optimizer. This post-hoc calibration ensures that the calibrated probabilities $\hat{p}_i = \sigma(\mathbf{z}/T)_i$ yield confidence scores that are better aligned with empirical accuracy.

Empirical Observations To validate the filtering mechanism, we utilized **Reliability Diagrams** to analyze the alignment between confidence and correctness. Across all tested architectures and ensemble settings, experimental results reveal a **strong positive correlation** between the confidence score $C(x)$ and empirical accuracy. This fundamental observation confirms that uncertainty-based metrics reliably serve as a proxy for prediction correctness, providing a robust empirical foundation for our proposed router framework.

B.3 Experimental Setup for Entropy Prediction

Model Framework and Calibration Experiments were conducted on CIFAR-100 using a knowledge distillation framework. We employed a ResNet-101 as the Student model, paired with two Teacher models of varying capacities: a modified ResNet-50 (High-Capacity) and a ShuffleNetV2 (Lightweight). Prior to predictor training, all models underwent **Temperature Scaling** to ensure their output probabilities accurately reflected prediction uncertainty.

Entropy Decrease Predictor To estimate the utility of querying a teacher without incurring its inference cost, we trained lightweight MLP predictors (Structure: $100 \to 64 \to 1$). These predictors take the Student's soft probabilities as input and regress the expected **Entropy Drop** ($\Delta H = H_{\text{student}} - H_{\text{teacher}}$). The predictors were optimized using Mean Squared Error (MSE) loss, enabling the system to forecast the potential uncertainty reduction provided by each teacher.

Visualization and Validation We validated the predictors using **t-SNE** to project the Student's high-dimensional features into 2D space. By mapping both the *Actual* and *Predicted* entropy drops onto this manifold using heatmaps with synchronized color scales, we observed a high degree of structural alignment. This alignment indicates that the predictor captures the distribution of uncertainty gains.

B.4 Data Augmentation

During training, we apply a series of data augmentation techniques, including random cropping with padding, random horizontal flipping, random rotation within $\pm 15°$, the AutoAugment policy predefined for CIFAR-10, and random erasing (with a probability of 0.5 and a controllable erasure area ratio). All images are normalized according to dataset-specific statistics. During testing, only normalization and tensor conversion are applied.

B.5 Model Architecture

Experiments conducted on CIFAR-100 are based on two models, referred to as the Round1 and Round2 models, simulating the scenario of model updating. The Round1 model represents the model used after the first round of annotation, while the Round2 model denotes the stronger model introduced during the second-round update on the edge. The Round1 model is based on the ResNet-101 architecture, retaining the original feature extraction modules (including convolutional layers, residual blocks, and global average pooling layers). A new linear layer is appended after the output layer to map the features to a 100-dimensional output space suitable for CIFAR-100 classification. The Round2 model is based on a modified ResNet-50 architecture. The initial convolutional layer is replaced with a 3×3 convolution (stride 1, padding 1, no bias) to better adapt to 32×32 pixel inputs, reducing information loss during initial feature extraction. The original max pooling layer is removed (replaced with an identity mapping) to preserve spatial resolution and low-level features. Additionally, a dropout layer with adjustable probability is inserted before the fully connected layer to enhance generalization. Finally, the output is mapped to a 100-dimensional space via a linear layer.

B.6 Baselines

To provide a comprehensive evaluation of our proposed approach, we compare it against several representative baseline methods, including **CEMA**, **DKD**, and **MLink**. To ensure a fair comparison and eliminate confounding factors such as dataset distribution or update volume, we adopt consistent update rates (20% and 50%, with corresponding skip rates of 80% and 50%) across the fixed-budget experiments and use an identical test set for evaluation.

CEMAChen et al. [2024]: We implement the filtering strategy proposed in the original CEMA framework, which employs a dynamic entropy thresholding mechanism to identify unreliable high-entropy samples and a fixed threshold to detect low-entropy, low-information samples. To adapt this method to our controlled filtering settings, we tune the dynamic entropy threshold parameters. Specifically, we set the upper entropy threshold as $H_{\text{max}} = 0.4 \times \ln C$, where C is the number of output classes. Under a fixed filtering ratio, we solve for H_{min} such that the total number of filtered samples matches the predefined threshold.

DKDLi et al. [2021]: We incorporate uncertainty-aware teacher selection in the student training process. Here, the prediction uncertainty of the round-1 model—quantified using entropy—is used to determine the sample's learning stage. Based on this stage, a suitable teacher model is chosen. We sort samples by their entropy scores as produced by the round-1 model, and filter the top 20% and 50% highest-entropy samples for respective experimental settings. These selected samples are reprocessed using a round-2 model, and their output distributions are then used to update the final predictions.

MLinkYuan et al. [2022]: MLink introduces a novel semantic linking mechanism that maps the output of one model to the corresponding output of another. In our adaptation, we train a mapping function from the predictions of the round-1 model to those of the round-2 model for each sample x. We use the confidence score for each transformation. During inference, we filter samples based on these confidence scores—only the top $k\%$ highest-confidence samples are retained, ensuring the filtering volume remains consistent with our experimental design.

B.7 Experimental Details for Multi-Round Deployment

To simulate the iterative lifecycle of edge intelligence systems, we construct a progressive model pool $\{M_0, \ldots, M_4\}$ using ResNet backbones on the CIFAR-100 dataset. M_0 serves as the legacy baseline, a ResNet-101 model trained to convergence with 60.84% accuracy. Incremental updates are represented by M_1 and M_2 (achieving $\sim$61% and $\sim$62% accuracy, respectively), where M_1 is generated by injecting controlled Gaussian noise ($\mathcal{N}(0, 0.15 \cdot \sigma_{weights})$) into the weights of M_2. To reflect major version releases or training breakthroughs, we derive M_3 (80.68% accuracy) from the final deployment target M_4 (80.75% accuracy) by applying a smaller noise term of $\mathcal{N}(0, 0.05 \cdot \sigma_{weights})$.

The selection strategy is driven by a lightweight Entropy Decrease Predictor, implemented as a three-layer Multilayer Perceptron (MLP) with 256 neurons per hidden layer and ReLU activations. This predictor takes the 100-dimensional probability distribution (soft targets) from the resident model M_0 as input to estimate the expected scalar entropy reduction $\Delta\hat{H}$ for each candidate . The predictor is optimized using the Adam optimizer ($lr = 0.001$, batch size 32) over 100 epochs with a Mean Squared Error (MSE) loss. Notably, this module is highly efficient for edge environments, comprising only $\sim$0.02M parameters and requiring less than 15 minutes of training time on a single GPU.

Finally, the multi-stage correction mechanism is integrated as a post-processing module. The first stage involves calculating the omission risk $\mathcal{R}_{\text{Omission}}$ for non-selected samples by normalizing historical entropy scores through a $\tanh$ activation, $S'_{t,i} = 0.5(1 + \tanh(S_{t,i}))$, and applying an overconfidence penalty β to the most recent predictions. Samples flagged as high-risk are then subjected to a majority voting ensemble composed of the current and select historical models. During this process, models are consistently operated in evaluation mode with a batch size of 32 to ensure the stability of batch normalization statistics.

Table 12 reports the five-seed variability of Round-4 accuracy for the main multi-round baselines. The main text keeps the detailed per-round trajectory and correction ablations in Table 2; this appendix

Table 12: Five-seed CIFAR-100 multi-round Round-4 accuracy. Values are mean ± standard deviation in percentage points. The main text reports the detailed per-round trajectory and ablations; this appendix table reports seed variability for the final deployment round.

Method	**Round-4 accuracy (%)**
Optimal	80.76 ± 0.00
PURE (Ours)	78.83 ± 0.15
MLink	77.16 ± 0.05
Entropy	76.24 ± 0.30
CEMA	76.09 ± 0.08
Random	76.06 ± 0.15

table provides the corresponding mean and standard deviation summary for the final deployment round.

B.8 Mobile Environment Emulation Setup

To accurately evaluate the performance of our proposed method on real-world edge devices, we constructed a high-fidelity emulated mobile environment. This environment was designed to replicate the computational characteristics of an Apple iPhone 15 Pro, a representative high-end mobile platform equipped with the A17 Pro chip.

The emulation was performed on a high-performance x86-architecture host machine (PC/server) by strictly constraining its computational resources. Key parameters were aligned with the hardware specifications of the iPhone 15 Pro to ensure the validity and representativeness of our results. The specific resource constraints were configured as follows:

- Processor (CPU): The A17 Pro chip features a 6-core CPU, comprising 2 performance cores and 4 efficiency cores. To emulate this configuration, we limited the number of CPU cores available to our test process on the host machine to 6.
- Memory (RAM): The iPhone 15 Pro is equipped with 8 GB of system memory. Correspondingly, we set a memory usage limit of 8 GB for our test process to simulate the memory capacity constraints of the actual device.

To ensure consistency and reproducibility across all experimental platforms, and to specifically evaluate the computational efficiency of our algorithm on processor cores, all our experiments were exclusively executed on the CPU. No GPU or other dedicated hardware acceleration was utilized.

Through this resource-constrained, CPU-only emulation method, we were able to create a controlled testbed that closely mimics the target mobile platform (iPhone 15 Pro). The performance metrics obtained in this environment, as reported in the main paper, reliably reflect the practical execution efficiency of our algorithm when deployed on a mainstream high-end smartphone using only its CPU.

C More Experiment Results

C.1 Multi-Expert Scheduling on CIFAR-100

The CIFAR-100 dataset consists of 100 fine-grained classes, which are further grouped into 20 superclasses. In this experiment, we trained three specialized models, denoted as f_1, f_2, and f_3, each targeting a specific superclass: *flower*, *tree*, and *insect*, respectively. Each of the models achieved over 80% accuracy within its corresponding superclass while maintaining only around 20% accuracy across the remaining superclasses.

Our router operates under the multi-model routing scenario, where, under a filtering constraint, it selects the model expected to yield the greatest entropy reduction for each input sample. The Figure 5 shows that, even when dispatching among models whose average accuracy across all classes is just 20%, the router still improves upon the baseline Round-1 model. This improvement is especially significant within the targeted superclasses (e.g., *flower*, *tree*, *insect*). Due to the nature of the routing problem, we are currently unable to offer other baselines. This limitation is mainly because, under

the entropy-based update strategy proposed in Task 1, the system lacks access to the ground truth and can only determine whether to perform an update, not which model among the candidates would yield the best output.

Given this, we introduce two policy strategies to benchmark routing performance in this setting: We begin by assuming that our classifier behaves as a random classifier. Specifically, for a given input from a particular superclass, the accuracy of classifier's prediction is modeled as a binary random variable taking values in $\{0, 1\}$, with the probability of correct classification by model j on sample X_i denoted as $P(F_j(X_i) = \text{Target}(X_i)) = p_i^j$. Furthermore, we assume that the outputs of this classifier are independent and identically distributed (i.i.d.) random variables.

Based on this assumption, we propose two strategies:

Random Policy: For a given filtering ratio k, out of a total of m samples. Let there be l superclasses, with each superclass containing $\frac{m}{l}$ samples. Assume that $h_1, h_2, \ldots, h_l$ samples are filtered in each superclass, and that the accuracy of the original model on each superclass is $q_1, q_2, \ldots, q_l$. Then, based on our randomness assumption, the expected accuracy under the random policy is given by:

$$Acc_{\text{random}} = (1-k) \cdot \left(\sum_{i=1}^{l} \frac{l \cdot q_i \cdot h_i}{m} \right) + \frac{k}{j} \cdot \sum_{j=1}^{m} \sum_{i=1}^{l} \left(1 - \frac{l \cdot h_i}{m} \right) \cdot p_i^j \tag{10}$$

Oracle Policy: Retaining the same assumptions, we now consider an oracle mode router, which is allowed access to the ground truth labels of the input samples. However, since each model's output remains unpredictable (modeled as a random variable), the oracle can only assign the model with the highest expected accuracy to each non-filtered sample. This selection is constrained by the filtering budget, and thus, for each h_i, the allocation becomes piecewise. The expected accuracy under the oracle policy is:

$$Acc_{\text{oracle}} = (1-k) \cdot \left(\sum_{i=1}^{l} \frac{l \cdot q_i \cdot h_i}{m} \right) + k \cdot \sum_{i=1}^{l} \left(1 - \frac{l \cdot h_i}{m} \right) \cdot \max_{1 \le j \le m} p_i^j \tag{11}$$

Notably, although the strategy selects the optimal model at each step, it does not constitute a strict upper bound. The values $h_1, h_2, \ldots, h_l$ are functions of the filtering threshold. We adopt the same thresholding scheme as in the main experiment, where the threshold is varied from -1.5 to 1 in 5 evenly spaced intervals. Figure 6 illustrates that as the threshold increases, the number of filtered samples exhibits a sharp decline, while our router demonstrates robust performance and closely tracks the stochastic oracle proxy when the threshold exceeds zero.

C.2 Multi-task Scheduling

C.2.1 Task-Specific Adaptation Strategies

To maintain the applicability of our uncertainty-based framework across different data modalities, we developed task-specific strategies to extract probability distributions and estimate entropy reduction for non-standard classification tasks. For discriminative text models such as BERT, we directly utilize the outputs from the final classification head. Since these models are explicitly trained for N-way classification, their logit distributions provide a direct measure of prediction uncertainty. In contrast, for generative Large Language Models (LLMs) like Qwen, where the output is typically a sequence of tokens rather than a fixed-class distribution, we employ a "Multiple-Choice Prompting" strategy. The model is prompted to select the most appropriate category from options (e.g., A/B/C/D), allowing us to extract the probability distribution specifically over these option tokens and treat it as the classification probability for entropy calculation. For object detection, whose outputs include bounding boxes and objectness scores rather than a single probability vector, we use an image-level average confidence decrease over detected objects as the routing proxy. This supports image-level update decisions for photo-album metadata, but it is not an object-level routing algorithm.

C.2.2 Extended Discussion on Experimental Results

The successful deployment of PURE across diverse tasks yields several key insights into the behavior of uncertainty-based input filtering.

Regarding high-fidelity in low-update regimes, we observed that at a 20% update rate the accuracy drop is small in several classification settings (e.g., 96.3% vs. 96.41% on CIFAR-10). This indicates

that our router accurately identifies samples where the update model is most likely to change the final prediction while allowing many stable samples to reuse stored outputs.

In terms of graceful degradation, a 50% update rate halves the re-inference workload. While a performance drop is expected, it remains within an acceptable range for edge devices. For example, in the COCO dataset, we observed a controlled decrease in mAP@50 from 71.5% to 65.7%. This capability is critical for battery-powered edge devices that may need to pivot to low-power modes during charging-limited windows.

Finally, the results demonstrate flexibility across modalities. The consistency scores across AG News and IMDb (reaching up to 99.96%) suggest that probability-space uncertainty can be useful beyond vision when the task exposes a fixed answer space. We do not claim this extends unchanged to open-ended generation or embedding-only retrieval systems.

C.3 Real-world Application Scenarios

In this phase, we simulate the real-world scenario of a smart photo album on mobile devices. Current smart photo albums usually possess functions such as image classification, object detection, aesthetic scoring, face detection, image depth estimation, OCR (Optical Character Recognition), expression classification, and so on. We consider the scenario when a new round of model iteration occurs, and re-inference needs to be performed on all photos. Our dataset is a real-world dataset built using images and videos collected from a smartphone gallery. To standardize the data format, we first downsampled the videos into images of the same format, resulting in a total of 2000 image samples. Afterwards, we performed re-inference on the entire image dataset.

For the task settings, both object detection and face detection rely on the similarity of bounding boxes, measured by Intersection over Union (IoU). In cases where multiple objects are detected, we aggregate per-box confidence changes into one image-level routing score using average confidence decrease. Unlike previous experiments where a fixed update rate was used, the deployment case study fixes the consistency target and varies the routing threshold. Specifically, we iteratively adjusted the threshold from higher to lower values and identified the maximum skip rate that satisfies the target. With the finalized image-level average-confidence operator, object detection reaches a 35.72% skip rate at a 95% IoU-consistency criterion.

For additional baseline grounding, we also evaluated the recovered YOLOv8n and YOLOv8x CSV exports for the object-detection subset. This diagnostic treats the YOLOv8x output as the full-update reference: skipped images reuse YOLOv8n detections, updated images use YOLOv8x detections, and consistency is measured by symmetric class-aware image-level IoU agreement. Since no human box labels are attached to these CSV exports, this experiment measures consistency to the full update rather than detection mAP. It supports a direct comparison among random routing, entropy, least confidence, an MLink-style old-to-new feature mapper, and the PURE cross-fit predictor on the same recovered detection outputs.

For the text classification task, we followed a similar setup as described above. The difference lies in the fact that we first applied OCR (Optical Character Recognition) to extract text content, resulting in 2000 text files for further processing.

Regarding the scene classification task, we used the UC Merced Land Use Dataset for training and evaluation. We first trained a ResNet model in the initial round. Then, for the second round of training, we employed a YOLO model. Since YOLO requires bounding box annotations, we generated pseudo bounding boxes covering the entire image for each sample. Finally, the output label for each image was determined based on the highest confidence score from the YOLO model's predictions.

C.4 Hyperparameter Sensitivity Analysis

To rigorously evaluate the robustness of our multi-stage correction mechanism, we conducted a comprehensive sensitivity analysis on the CIFAR-100 dataset using 59 experimental configurations (2.4 times denser than the initial pilots). We analyzed five key hyperparameters governing the correction process: Selection Rate (ρ_{sel}), Overconfidence Penalty (β), Confidence Threshold (σ), Screening Ratio (ρ_{screen}), and Correction Ratio ($\rho_{correct}$).

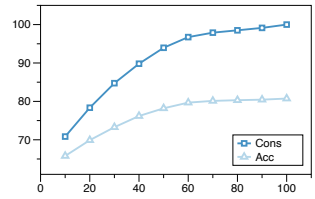

Figure 4: In CIFAR100, the changes in accuracy (Acc.) and consistency (Cons.) under different update/skip budgets.

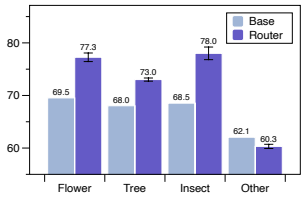

Figure 5: Comparison of the accuracy(Acc.) of the router and the base model under different super-classes.

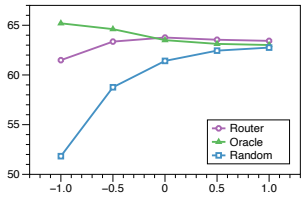

Figure 6: Comparison of the accuracy of the router and the other two policies under different thresholds

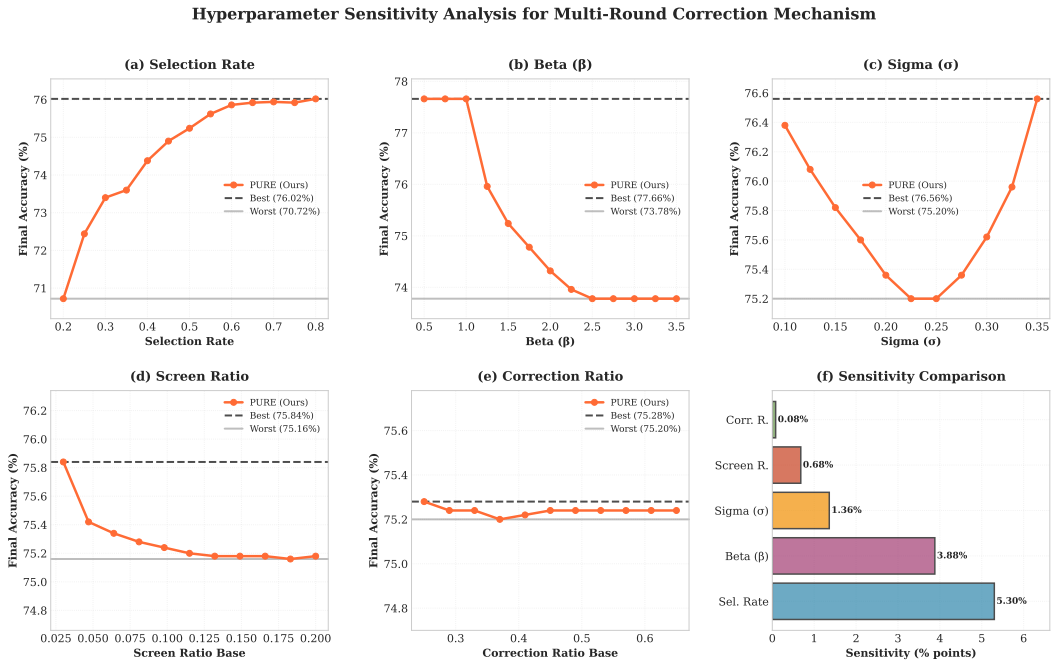

Figure 7: Comprehensive hyperparameter sensitivity analysis. (a-e) Impact of individual parameters on final accuracy (Round 4). The orange line represents our method (PURE), while the black dashed and gray solid lines indicate optimal and worst-case bounds. (f) Sensitivity comparison showing the max-min accuracy variation for each parameter.

High Sensitivity Parameters. As shown in Figure 7(f), the **Overconfidence Penalty** (β) exhibits the highest sensitivity (3.88%). Figure 7(b) reveals a clear monotonic trend where lower β values (reducing penalty for high-confidence predictions) significantly improve performance, with $\beta = 1.0$ achieving the peak accuracy of 77.66%. The **Selection Rate** (Figure 7(a)) shows a strong positive correlation with accuracy (Sensitivity 2.54%), confirming that allocating more computational budget consistently yields better performance, though 50% provides an optimal efficiency-accuracy trade-off.

Moderate Sensitivity Parameters. The **Confidence Threshold** (σ) (Figure 7(c)) demonstrates a U-shaped curve with optimal performance around $\sigma = 0.15$ and $\sigma = 0.30$, suggesting that the mechanism effectively captures omission risks when the Gaussian kernel width is either very tight (focusing on high-confidence errors) or broad (capturing general uncertainty). The **Screening Ratio** (Figure 7(d)) shows a slight downward trend (Sensitivity 0.68%), indicating that an overly aggressive initial screening might introduce noise, validating our choice of a conservative 10-15% ratio.

Robustness of Correction. Notably, the **Correction Ratio** (Figure 7(e)) exhibits negligible sensitivity (0.02%). This indicates that our correction mechanism is highly robust to the exact proportion of samples selected for voting correction, provided the initial screening effectively identifies high-risk

candidates. This robustness is a desirable property for practical deployment, reducing the need for fine-grained parameter tuning.

Table 13 summarizes the optimal configurations derived from this analysis.

Table 13: Hyperparameter Sensitivity Summary

Parameter	Range Tested	Baseline	Optimal	Gain	Sensitivity
Beta (β)	1.0–3.0	1.5	1.0	+2.42%	High (3.88%)
Selection Rate	0.3–0.7	0.5	0.7	+0.70%	High (2.54%)
Sigma (σ)	0.15–0.30	0.22	0.15	+0.58%	Med (0.62%)
Screen Ratio	0.05–0.15	0.10	0.05	+0.16%	Med (0.22%)
Correction Ratio	0.35–0.55	0.45	0.35	+0.00%	Low (0.02%)

C.5 Ablation Study

Compression Strategies and Performance Evaluation In model update scenarios, historical inference results are often discarded to save storage. Thus, we analyze whether our router can maintain effective scheduling after quantizing these distributions. Storing full-precision probability distributions imposes significant storage costs on edge-side devices: for a 100-class problem, each sample requires 100 float32 values (400 bytes), totaling 4MB for 10,000 images. To address this, we design two complementary compression strategies.

First, probability precision quantization reduces storage by representing probabilities using discrete intervals of $(1/2^q)$. Specifically, we employ 4-bit quantization (0–15 levels), 8-bit quantization (0–255 levels), and 16-bit half-precision floating point (float16). Second, the **top-k sparsification** strategy leverages the high sparsity observed in quantized probability vectors—most samples with 4-bit quantization exhibit only one non-zero element. We retain the top-k highest probabilities (k=1, 4, 8, and the full distribution k=100 as a baseline).

Quantization achieves dramatic storage reduction: for CIFAR-100, the per-sample space decreases from 400 bytes to about 1.5 bytes when the top-1 class index is packed together with a 4-bit probability value. For larger label spaces, we use a conservative accounting of 2.5 bytes per sample with a uint16 class index plus the 4-bit probability value. Both settings yield a 99%+ compression ratio with less than 1% accuracy degradation in our CIFAR-100 sweep. This trade-off between storage efficiency and prediction fidelity makes our quantization approach viable for resource-constrained edge devices, effectively balancing low-latency inference requirements with hardware limitations.

Table 14: Performance and time cost under different router training sample sizes on CIFAR-100. Acc/Cons denotes Accuracy/Consistency. Pre/Train indicates preprocessing and training time in seconds.

Samples	50000	10000	5000	1000
Acc/Cons	78.3/94.1	78.3/94.1	78.4/94.0	70.9/82.3
Pre/Train	33.4/127.4	7.0/25.9	4.5/12.9	2.7/1.5

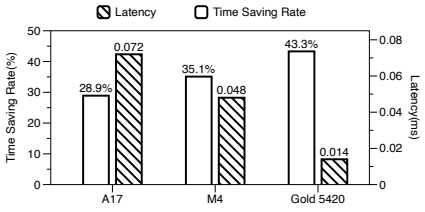

Figure 8: Performance on different hardware platforms.

Training Data Scale To further investigate the data efficiency and training overhead of our proposed router model, we evaluate the impact of varying training data scales on model performance and

efficiency. Specifically, using the CIFAR-100 dataset, we progressively reduced the number of samples for training the router from the full 50,000 down to 1,000. The results are presented in Table 14.

*Data Efficiency: Effective with Minimal Data.*The results in Table 14 clearly highlight a core advantage of our router: its robustness to the size of the training set, enabling effective performance with only a small number of samples. The experiments demonstrate that even when the training data is drastically reduced by 90%, from 50,000 to 5,000 samples, the model's final performance (Accuracy and Consistency) remains almost unaffected, maintaining an accuracy of 78.4% and a consistency of 94.0%.

Lightweight Overhead: Efficient Preparation and Training. As shown in Table 14, the time costs for both data preparation (Pre) and model training (Train) decrease significantly with fewer samples. This demonstrates that the entire workflow, from data preparation to model training, is both efficient and low-cost, enabling an agile iteration and update process for the router that can rapidly adapt to model changes.

Deployment and Performance on Hardware Platforms To validate the practical utility and efficiency of our proposed method on real-world hardware, we deployed and benchmarked our router on three representative computing platforms: a high-performance server (Intel Xeon Gold 5420), a modern personal computer (Mac M4), and a mobile device (simulated iPhone A17 Pro environment). The results are presented in Figure 8.

The experimental data clearly indicate that our router model features extremely low inference latency and a minimal memory footprint. On all tested platforms, the latency for a single routing decision is at the sub-millisecond level (0.014ms - 0.072ms), and the model size is merely 0.02MB.

More importantly, by skipping redundant update inputs, our method achieves significant overall time savings across all platforms, ranging from 28.9% to 43.3%. We note a discrepancy between the actual time savings and the nominal skip rate. This gap is primarily due to routing, indexing, and the thresholding strategy. Specifically, to identify the top-k samples for re-inference, we currently employ a naive implementation that performs a global sort on the predicted uncertainty changes for all samples. Streaming top-k selection can reduce this overhead further.

We acknowledge that the top-k query algorithm in data streams has been extensively researched [Charikar et al., 2002, Cormode and Muthukrishnan, 2005]. Integrating these advanced algorithms into our system would undoubtedly compress the time cost of the threshold calculation step. However, as the core contribution of this paper is the router itself, we leave this in-depth optimization as a valuable direction for future work.

D More Related Works

D.1 Model Compression

Model compression techniques have evolved from basic approaches to advanced paradigms that integrate dynamic adaptation and hardware awarenessLi [2025]. For instance, Han et al. [2016] established the foundation for reducing model complexity by introducing "deep compression", a three-stage pipeline that works together to reduce the storage requirement of neural networks by $35\times$ to $49\times$ without affecting accuracy. Specifically, this method first prunes the network by learning only the important connections, then quantizes the weights to enforce weight sharing, and finally applies Huffman coding, allowing the model to fit into on-chip SRAM cache rather than off-chip DRAM memory.

To address efficient inference on mobile devices, Jacob et al. [2018] proposed a quantization scheme enabling integer-only arithmetic, which significantly improves the tradeoff between accuracy and on-device latency. They also co-designed a training procedure to preserve end-to-end model accuracy post-quantization, demonstrating significant improvements even on efficient model families like MobileNets.

More recently, specific challenges of large-scale generative models have been addressed; for example, Frantar et al. [2022] developed GPTQ based on approximate second-order information, enabling the quantization of 175 billion-parameter models to 3 or 4 bits with negligible degradation. This method

more than doubles the compression gains relative to previously-proposed one-shot quantization methods, allowing the execution of massive models inside a single GPU for the first time. Similarly, Liu et al. [2017] focused on structural efficiency through "network slimming", which enforces channel-level sparsity to automatically prune insignificant channels. Distinct from many existing approaches, this method requires no special software/hardware accelerators and can achieve a $20\times$ reduction in model size for VGGNet through a multi-pass version.

While Knowledge Distillation (KD) remains a core methodology for compressing ensemble knowledge into a single model Hinton et al. [2015], recent advancements have introduced dynamic properties to this process. Inspired by active learning Settles [2009], Li et al. [2021] explored dynamic knowledge distillation, empowering the student model to adjust the learning procedure—including teacher adoption and data selection—based on prediction uncertainty derived from entropy. Experimental results indicate that conducting KD with only 10% informative instances achieves comparable performance while greatly accelerating training.

This shares some similarity with our research approach, but our method predicts the uncertainty reduction after executing another model, thereby capturing the potential relationships between two models during single-sample updates. This enables our method to effectively handle application scenarios that require dynamic updates among multiple candidate models. It is worth noting that our proposed method is orthogonal to model compression techniques and can be applied in conjunction with them.

D.2 Model Selection

The objective of active model selection is to estimate the best-performing model from candidates with minimal labeling. Addressing the high cost of labeling during evaluation, Matsuura and Hara [2023] proposed a variance minimization approach, employing an adaptive labeling strategy to estimate test loss differences with small variance. This strategy effectively addresses the need for a substantial amount of labeled test data, enabling the estimation of the best model using fewer labeling costs based on an appropriate test loss estimator. In scenarios where models cannot be compared on held-out training data, Sawade et al. [2012] devised an active comparison method that selects instances according to an instrumental sampling distribution to maximize the power of a statistical test. By maximizing this power applied to the observed empirical risks, the method minimizes the likelihood of choosing the inferior model between a baseline and a challenger.

However, these approaches predominantly focus on identifying a single "globally optimal" model or aggregating predictions, implicitly assuming uniform data distributions. A fundamental limitation persists: they lack the capability for sample-specific model assignment, limiting adaptability in complex real-world scenarios where candidate models exhibit specialized expertise in distinct data subspaces.

D.3 Model Scheduling

Model scheduling has evolved into a pivotal paradigm for optimizing multi-model system efficiency. Early systems like Dionysus Jin et al. [2014] focused on network updates, dynamically scheduling based on runtime differences rather than pre-determined schedules. By encoding the consistency-related dependencies among updates as a graph, this approach significantly improves update speeds in software-defined networks. Scaling this to rack-level computing, Zhu et al. [2020] presented RackSched, utilizing a two-layer framework to integrate inter-server and intra-server scheduling for low tail latency. The inter-server scheduler in this framework employs a custom switch data plane to realize power-of-k-choices and ensure request affinity, tracking server loads accurately.

In the context of machine learning, Yuan et al. [2022] introduced model linking to bridge black-box models by learning mappings between their output spaces, enabling collaborative inference that saves computation while preserving accuracy. This design specifically addresses the dilemma where the cost budget (e.g., GPU memory) is insufficient to run multiple models, outperforming multi-task learning baselines. Furthermore, to maximize model output value under resource constraints, Yuan et al. [2020] proposed an Adaptive Model Scheduling framework using deep reinforcement learning to mine semantic relationships and predict the value of unexecuted models.

Addressing distribution shifts in cloud-edge scenarios, the Cloud Edge Elastic Model Adaptation (CEMA) paradigm Chen et al. [2024] adapts edge models online by excluding unreliable and low-informative samples via entropy threshold control. Based on the uploaded samples, it updates and distributes the affine parameters of normalization layers by distilling from a stronger foundation model with a sample replay strategy.

Our research extends beyond these foundations by specifically addressing the model updating scenario requiring data relabeling.

E Discussion

E.1 Details on Multi-Round Screening Mechanism

In this section, we elaborate on the mathematical formulations underpinning our multi-round screening mechanism, detailing the derivation and interpretation of its specific components.

E.1.1 Analysis of Persistent Sample Omission

We begin by analyzing the mechanism underlying the persistent omissions in multi-round updates.

For a given sample x_0, the mechanisms that lead to this persistent omission are categorized into two primary scenarios:

Marginal Exclusion During the update process, certain candidate models may correctly predict x_0 with reduced entropy relative to the initial model. However, the improvement score may consistently fail to exceed the Top-K threshold in any given round. We consider this a resource-allocation failure rather than an intrinsic metric failure. Typically, this issue is mitigated over sequential rounds as higher-priority samples are resolved.

The Overconfidence Trap A more critical failure arises from our use of prediction confidence as a proxy for predictive accuracy. It is possible that a model yields a prediction for x_0 that is confidently wrong. Since the conditional entropy is low, our selection function may misinterpret the prediction as safe and exclude x_0 from the update set. Based on the mechanism detailed in Section 3.1, such confident errors can arise from a coincidental alignment of negative interference that concentrates the prediction probability mass of x_0 onto a single incorrect class y_u. Once a model falls into this state, it can lead to persistent misclassification of x_0 without triggering a corrective update.

E.1.2 The Derivation and Interpretation of $\mathcal{R}_{\text{Omission}}$

The Omission Risk Function, $\mathcal{R}_{\text{Omission}}(x_i)$, is designed to quantify the likelihood that a sample has been incorrectly excluded from updates across multiple rounds. It is composed of two terms, each targeting a specific failure mode:

$$\mathcal{R}_{\text{Omission}}(x_i) = \underbrace{\sum_{t=\tau(i)+1}^{T} S_{t,i}}_{\text{Term I: Marginal Exclusion}} + \underbrace{\beta \cdot \exp\left(-\frac{S_{\tau(i),i}^2}{2\sigma^2}\right)}_{\text{Term II: The Overconfidence Trap}} \tag{12}$$

Term I: Addressing Marginal Exclusion via Cumulative History. The first term addresses the "near-miss" scenario. In a resource-constrained setting (e.g., top-20% filtering), many samples might benefit from an update but fall just below the cut-off threshold. If a sample consistently ranks near the threshold (e.g., top-25%) across multiple rounds, it accumulates a high "marginal exclusion" score. The score $S_{t,i}$ uses a $\tanh$ activation on the predicted entropy drop:

$$S_{t,i} = \frac{1}{2}\left[1 + \tanh\left(\Delta\hat{H}(F_{j_t^{(i)}}(x_i))\right)\right] \tag{13}$$

We employ $\tanh$ to normalize the unbounded entropy decrease values into a bounded $[0, 1]$ interval. Numerically, a score $S_{t,i} \in (0.5, 1]$ signifies a net entropy reduction (increased confidence), whereas $S_{t,i} \in [0, 0.5)$ implies an increase in entropy. However, since $\mathcal{F}_{j_t^{(i)}}$ is explicitly selected as the candidate model exhibiting the maximum entropy drop relative to the global baseline $\mathcal{F}_0$, the effective

range of the score is constrained to $S_{t,i} \in (0.5, 1]$ in practice. This prevents a single round with an outlier entropy drop from dominating the entire sum, ensuring that the risk score reflects a consistent history of neglect rather than a one-time anomaly.

Term II: Detecting the Overconfidence Trap via Gaussian Kernel.

Regarding " overconfidence trap ", we attribute its origin to a coincidental alignment of negative interference during the training process. Unlike genuinely correct predictions, these errors—while appearing confident—rarely attain the distinctly high levels of certainty typical of valid updates. Consequently, critical attention must be directed toward samples that exhibit an entropy reduction (identifying them as candidates) but fail to demonstrate a significant magnitude of drop. In the design of Term II, this intuition is formalized such that a smaller $S_{\tau(i),i}$ (indicating moderate confidence gain) corresponds to a larger omission risk $\mathcal{R}_{\text{Omission}}$. The derivation of the specific function for Term II is as follows:

Premise: Let $S_{\tau(i),i}$ denote the normalized entropy reduction score for sample x_i. We assume the misclassification risk peaks at an unknown, sample-specific ambiguity center μ_i.

Hypothesis: The risk function $\mathcal{R}$ follows a Gaussian decay centered at μ_i:

$$\mathcal{R}(x_i) \propto \exp\left(-\frac{(S_{\tau(i),i} - \mu_i)^2}{2\sigma^2}\right) \tag{14}$$

Decomposition: By expanding the exponent $-(S_{\tau(i),i} - \mu_i)^2 = -S_{\tau(i),i}^2 + 2S_{\tau(i),i}\mu_i - \mu_i^2$, the function factorizes into three distinct components:

$$\begin{aligned}\mathcal{R}(x_i) &= \exp\left(\frac{-S_{\tau(i),i}^2 + 2S_{\tau(i),i}\mu_i - \mu_i^2}{2\sigma^2}\right) \\ &= \underbrace{\exp\left(-\frac{\mu_i^2}{2\sigma^2}\right)}_{\text{I. Sample Constant}} \cdot \underbrace{\exp\left(\frac{S_{\tau(i),i} \cdot \mu_i}{\sigma^2}\right)}_{\text{II. Linear Interaction}} \cdot \underbrace{\exp\left(-\frac{S_{\tau(i),i}^2}{2\sigma^2}\right)}_{\text{III. Quadratic Decay}}\end{aligned} \tag{15}$$

Selection:

- Component I is a constant with respect to the current score.
- Component II depends on the unknown μ_i, making it unstable for global estimation.
- Component III represents the universal quadratic decay of risk as confidence increases, independent of μ_i.

Conclusion: To construct a robust metric agnostic to μ_i, we extract Component III as the dominant factor. This yields the final design for Term II:

$$\text{Term II} = \beta \cdot \exp\left(-\frac{S_{\tau(i),i}^2}{2\sigma^2}\right) \tag{16}$$

E.1.3 The Derivation and Interpretation of $\mathcal{R}_{\text{misclass}}$

For the second stage, we need a robust metric to determine if a selected high-risk sample is indeed unstable. We employ the Jensen-Shannon (JS) divergence:

$$\mathcal{R}_{\text{misclass}}(x_i) = \text{JS}(p||p') = \frac{1}{2}\text{KL}(p||M) + \frac{1}{2}\text{KL}(p'||M) \tag{17}$$

where $M = (p+p')/2$. We select Jensen-Shannon (JS) divergence over alternatives such as Kullback-Leibler (KL) divergence or Euclidean distance primarily due to its **symmetry** and **boundedness**. Unlike KL divergence, the symmetric nature of JS divergence ensures that the resulting risk score remains independent of model ordering. Furthermore, since JS divergence is bounded between 0 and

1 (when using base-2 logarithm), it provides a stable and interpretable risk metric that is comparable across diverse samples and training rounds, which is essential for the implementation of effective adaptive thresholds.

E.2 The Theoretical Justification for Our Routing Criterion

To provide the theoretical justification for the validity of the conditional entropy as a routing criterion, we invoke **Fano's Inequality** Gerchinovitz et al. [2020]. This fundamental theorem in information theory bounds the classification error rate P_e using the conditional entropy. The inequality establishes the following relationship :

$$H(Y|X) \leq H(P_e) + P_e \log(k-1) \tag{18}$$

In this formulation, $H(P_e) = -P_e \log P_e - (1-P_e)\log(1-P_e)$, $P_e = P(\hat{Y} \neq Y)$ represents the probability of classification error, where $\hat{Y} = \arg\max_k f_k(X)$ denotes the class predicted by the mapping function F.

By combining **Fano's Inequality** Gerchinovitz et al. [2020] with the relationship $Acc = 1 - P_e$ and rearranging the terms, we derive the following equation:

$$G(Acc) := Acc - \frac{H(1-Acc)}{\log(k-1)} \leq 1 - \frac{H(Y|X)}{\log(k-1)} \tag{19}$$

The derivation presented above is established based on Fano's inequality. For completeness, we provide a brief proof of this inequality below:

Proof. Define the error indicator random variable E:

$$E = \begin{cases} 1, & \text{if } \hat{Y} \neq Y \\ 0, & \text{if } \hat{Y} = Y \end{cases} \tag{20}$$

Applying the chain rule of entropy to $H(E, Y|\hat{Y})$:

$$\begin{aligned} H(E, Y|\hat{Y}) &= H(Y|\hat{Y}) + H(E|Y, \hat{Y}) \\ &= H(E|\hat{Y}) + H(Y|E, \hat{Y}) \end{aligned} \tag{21}$$

Noting that:

$$H(E|Y, \hat{Y}) = 0 \quad \text{and} \quad H(E|\hat{Y}) \leq H(E) = H(P_e) \tag{22}$$

where $H(P_e)$ is the binary entropy function.

Decompose $H(Y|E, \hat{Y})$ using conditional probabilities:

$$\begin{aligned} H(Y|E, \hat{Y}) &= (1-P_e)H(Y|\hat{Y}, E=0) + P_e H(Y|\hat{Y}, E=1) \\ &\leq (1-P_e) \cdot 0 + P_e \log(K-1) \\ &= P_e \log(K-1) \end{aligned} \tag{23}$$

Note: When $E = 0$, Y is determined by $\hat{Y}$. When $E = 1$, Y can be any of the remaining $K-1$ classes.

Combining equations (21) through (5):

$$H(Y|\hat{Y}) \leq H(P_e) + P_e \log(K-1) \tag{24}$$

Given the Markov chain $Y \to X \to \hat{Y}$ and the data processing inequality:

$$I(X;Y) \geq I(X;\hat{Y}) \implies H(Y|X) \leq H(Y|\hat{Y}) \tag{25}$$

(Assuming $\hat{Y}$ is a deterministic function of X or dependent via the chain, the conditional entropy reduces).

Substituting (7) into (6) completes the proof:

$$H(Y|X) \leq H(P_e) + P_e \log(K-1) \tag{26}$$

□

E.3 Discussions and Qualitative Analysis

E.3.1 The "Confident but Wrong" Paradox

A fundamental critique of uncertainty-based methods is the "Confident but Wrong" problem: a model may predict an incorrect class with very high probability (low entropy). In such cases, our entropy-reduction router might predict a near-zero ΔH, causing the sample to be skipped.

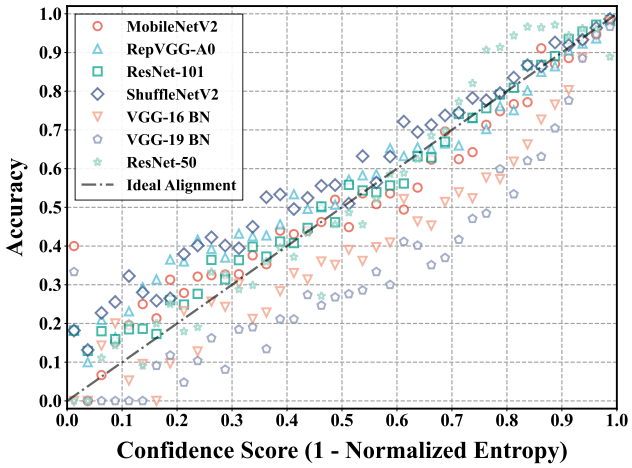

Figure 9: Reliability Diagram of the base model. The close alignment between the bar height (accuracy) and the diagonal line (perfect calibration) indicates that confidence is a reliable proxy for correctness in our setting.

As shown in the Reliability Diagram (Figure 9), our base models exhibit useful calibration, meaning that high confidence generally correlates with high accuracy. However, confident errors still occur and can accumulate in a multi-round setting when the same samples are repeatedly skipped. This provides the empirical motivation for our Multi-Stage Correction mechanism. By explicitly modeling the "Overconfidence Trap" (Term II in Eq. 15), we treat low-entropy samples not as automatically safe, but as potentially stagnant candidates for the more expensive multi-model voting process.

E.3.2 Why Use a Learned Entropy-Decrease Router?

One might ask: why train a neural router instead of simply selecting samples with the highest current entropy? The single-round diagnostics in the main text show a more nuanced answer: PURE is competitive with entropy, margin, least-confidence, and label-change baselines, but it should not be interpreted as uniformly superior to every uncertainty heuristic in every one-shot setting. The learned router is useful because it represents update benefit in a form that entropy alone cannot express:

1. Candidate-specific update benefit. In a multi-expert setting, a high-entropy sample could potentially be improved by different update models. A scalar entropy heuristic tells us *that* the sample is uncertain, but not which candidate is expected to reduce that uncertainty. PURE maps the cached old distribution to candidate-specific predicted benefits, $p_0(x_i) \to [\widehat{\Delta H}_1, \ldots, \widehat{\Delta H}_n]$. This allows targeted routing, such as sending an ambiguous flower-like image to the flower expert, while still using the same budgeted selection interface.

2. Correctability rather than uncertainty alone. Not all high-entropy samples are equally useful to refresh. Some inputs remain ambiguous even after a stronger update model. Because the router is trained on $\Delta H = H_{\text{old}} - H_{\text{new}}$, samples whose new entropy remains high receive small predicted benefit even if the old entropy was high. This is an empirical routing signal, not a proof that the router separates epistemic and aleatoric uncertainty, but it gives the model a direct target for expected update benefit.

F Supplementary Data

Table 15 presents the data and error analysis corresponding to Figure 4.

Table 15: Performance comparison between Accuracy (Acc) and Consistency (Cons) at different sampling ratios.

Ratio	Acc	Cons
10	65.79 ± 0.04	70.84 ± 0.02
20	69.90 ± 0.06	78.35 ± 0.05
30	73.28 ± 0.04	84.74 ± 0.00
40	76.19 ± 0.04	89.81 ± 0.10
50	78.32 ± 0.08	93.96 ± 0.02
60	79.68 ± 0.13	96.74 ± 0.14
70	80.13 ± 0.03	97.90 ± 0.09
80	80.31 ± 0.03	98.51 ± 0.03
90	80.45 ± 0.04	99.14 ± 0.08
100	80.74 ± 0.00	100.00 ± 0.00