Efficient Update-Aware Inference via Uncertainty Prediction-Based Input Filtering

Anonymous Authors[1]

Abstract

Under the trend of rapid iteration in edge-side intelligent applications, technologies represented by model pre-labeling have been widely adopted (e.g., smart photo albums). However, constrained by the limited performance of on-device hardware, complete re-inference on all samples is required after each model update, resulting in substantial resource consumption and prolonged update durations. Existing re-inference strategies neglect large historical samples with stable annotations, resulting in significant redundant computation. This highlights the urgent need to implement intelligent input filtering mechanisms to achieve efficient re-inference. We propose a lightweight, easily trainable, and budget-aware filtering router that reuses previous inference results. Extensive experiments demonstrate significant reductions in re-inference overhead while preserving high consistency in the efficiency–consistency frontier over representative baselines across 6 multimodal benchmarks and a real-world smart album system involving diverse architectures from CNNs to LLMs, suggesting a more budget-aware paradigm for sustainable edge-side model evolution.

1. Introduction

With the rising demand for on-device intelligence(Bangash, 2023; Jia et al., 2024b) and the rapid growth of edge computing(Barbuto et al., 2023), AI preprocessing has become increasingly vital(Liu & Guan, 2024). Many mobile applications, such as smart photo albums, rely on preprocessing tasks like face clustering, object detection, and scene classification(Abdelhalim & Salem, 2017).

However, when deployed models are updated, existing systems typically do re-inference on all samples (Mishra et al., 2025; Ilic et al., 2024) —a process that is both computationally expensive and time-consuming. This actually proves inadequate to meet users' needs, creating a major bottleneck in system responsiveness. A representative example is the smart photo albums (Park et al., 2022), where the very process of a single full re-inference update can take dozens of hours to complete. Worse still, as a low-priority background task, the re-inference is confined to the increasingly narrow window of screen-off charging periods, pushing the total update cycle to an unacceptably long span of weeks. Therefore, there is an urgent need for efficient re-inference strategies on resource-constrained edge devices(Yin et al., 2024; Xu et al., 2023; Jia et al., 2024a; Li et al., 2024).

Prior work in inference acceleration generally follows two directions: (1) reducing model-side redundancy(Pan et al., 2025; Kurtic et al., 2023), and (2) filtering redundant inputs(Yuan et al., 2024). Both directions have proven to be highly effective in practice, and they are orthogonal and complementary. While both are promising, this work will specifically pursue the latter.

Despite the considerable advancement that the latter has seen, most existing methods rely on reinforcement learning(Yuan et al., 2020), which demands long training times and struggles with deployment on edge devices.

This raises a key research question: *How can we design a lightweight, efficient, and deployable filtering router for edge scenarios?*

An existing approach is to train a model to predict whether a sample's inference result will change after the model update. However, such approaches are limited. They cannot adapt to runtime resource constraints or adjust filtering rates based on time budgets. More importantly, they lack coordination across multiple models—typically deciding only whether to re-infer, without selecting the most appropriate model for execution. To address these challenges, our work makes two key contributions:

• We identify a task that is prevalent in practical applications but has been neglected in prior research: how to leverage historical inference results to achieve more efficient re-inference after a deployed model is updated, rather than recomputing all samples from scratch.

• We introduce a lightweight, easily trainable, and deployable filtering router based on historical inference results, which we name PURE (Predicting Uncertainty Re-inference Efficiency). PURE consists of only a few layers of a sim-

Table 1. Feature comparison of our method and complementary baseline methods for efficient model inference

	Multi-model Scheduling	Budget Aware	Use Past records
Active Model Selection	✓	✗	✗
Mlink	✓	✗	✓
Adaptive Scheduling	✓	✗	✗
Dynamic KD	✗	✓	✗
CEMA	✗	✓	✗
Our Method	✓	✓	✓

ple neural network, enabling efficient decision-making with low training data requirements, minimal memory footprint, and fast inference, making it well-suited for edge deployment. Furthermore, it can dynamically adjust the filtering rate based on current system constraints, thereby enabling

2. Related Work

Efficient Inference and Selection. Model compression has evolved towards dynamic adaptation (Li, 2025). Techniques like pruning (Han et al., 2016; Liu et al., 2017) and quantization (Jacob et al., 2018; Frantar et al., 2022) reduce redundancy, while knowledge distillation (KD) (Hinton et al., 2015) remains a core methodology. Inspired by active learning (Settles, 2009), dynamic KD (Li et al., 2021) leverages entropy to adjust model interactions. **These methods primarily target static model optimization. Our proposed method can be seamlessly integrated with these compression techniques, extending them to the dynamic maintenance of deployed models.** In parallel, active model selection (Matsuura & Hara, 2023; Sawade et al., 2012) effectively identifies globally optimal models. While powerful, these approaches generally operate under the assumption of uniform data distributions, leaving the potential of sample-specific model assignment in complex edge scenarios less explored.

Dynamic Model Scheduling. Our work builds upon scheduling paradigms designed to optimize multi-model efficiency (Jin et al., 2014; Zhu et al., 2020). Seminal works like MLink (Yuan et al., 2022) and CEMA (Chen et al., 2024) have made significant strides in output mapping and handling distribution shifts, while adaptive scheduling (Yuan et al., 2020) has demonstrated the utility of reinforcement learning. Distinct from these approaches, our work specifically addresses the *model update* phase. As summarized in Table 1, we aim to complement existing methods by simultaneously coordinating multiple models and leveraging historical records under budget constraints, offering a lightweight alternative for edge deployment.

3. Motivation

Our routing strategy is motivated by two key insights derived from learning dynamics (see Appendix A) and empirical verification.

On the one hand, the feasibility of routing is grounded in model heterogeneity. Rather than attributing misclassification primarily to intrinsic sample ambiguity (Li et al., 2023; Elgaar & Amiri, 2023), we reveal that difficulty is also shaped by training data and is thus model-specific.

On the other hand, analysis based on learning dynamics demonstrates a positive correlation between classifier confidence and prediction accuracy. We empirically verify this correlation across multiple standard architectures and datasets. Furthermore, even if raw model outputs suffer from miscalibration, standard post-hoc techniques such as Temperature Scaling (Guo et al., 2017) can be employed to ensure confidence serves as a reliable proxy for accuracy (see Figure 123; experimental setup and calibration details are provided in Appendix C2).

Notably, unlike accuracy, confidence is an intrinsic property determined solely by the model's internal state, independent of external ground truth labels. This renders it a more tractable and predictable target, allowing the routing task to be performed by lightweight models, thereby meeting the stringent resource constraints of edge computing scenarios.

4. Methodology

4.1. Minimizing Objective

Conditional entropy is employed as the measure of model confidence. The following discussion outlines its specific definition and provides a theoretical justification for its validity as a superior alternative to accuracy for the routing criterion.

We begin by formalizing the architecture of a classifier mathematically.

Given a dataset $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$, where $\mathbf{x}_i \in \mathbb{R}^l$ and $y_i \in \{c_1, c_2, \cdots, c_k\}$ is the corresponding class label with k denoting the total number of classes. Let the model parameters be $\theta \in \mathbb{R}^d$. For a sample $\mathbf{x}$, denote by $\mathbf{z}$ the model logits and by $\mathbf{p}$ the output probabilities, of which the j-th component p_j represents the predicted probability that the input sample $\mathbf{x}$ belongs to the $j-th$ class. The core objective of the classification task is to learn a mapping function $F : \mathbb{R}^l \to \mathbb{R}^k$ from the training data. For an unseen sample x, the model predicts its class $\hat{y}$ by $\hat{y} = \arg\max_j f_j(x)$, that is, selecting the class corresponding to the maximum value in the vector $\mathbf{f}(x)$.

Formally, let X denote the input random variable and Y the output variable spanning k possible classes. The conditional

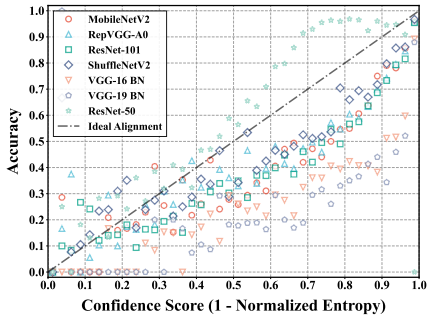

Figure 1. Alignment Between Confidence and Accuracy (Uncalibrated)

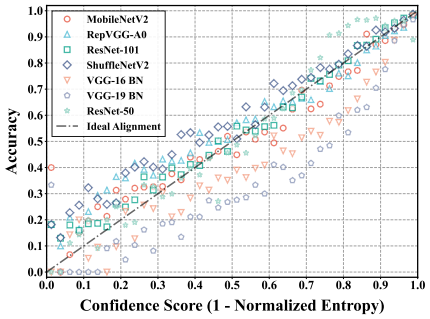

Figure 2. Alignment Between Confidence and Accuracy (Calibrated)

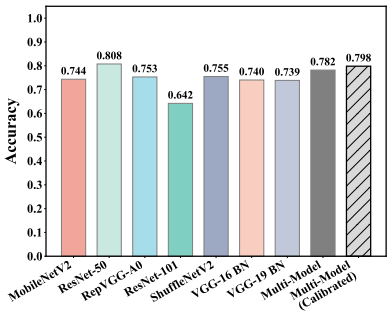

Figure 3. Model Performance Comparison

entropy $H(Y|X)$, which measures the average uncertainty of Y given X, is defined as:

$$H(Y|X) = -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log p(y|x) \tag{1}$$

By **Fano's Inequality** (Gerchinovitz et al., 2020) , we derive the following equation (see Appendix F.2 for the detailed derivation) :

$$G(Acc) := Acc - \frac{H(1-Acc)}{\log(k-1)} \leq 1 - \frac{H(Y|X)}{\log(k-1)} \tag{2}$$

where Acc denotes the probability of correct classification.

Under normal operating conditions where $Acc > 0.5$,$k \geq 3$, it follows that the function $G(Acc)$ is monotonically increasing.

Therefore, this derived formula unveils a profound insight: the upper limit of $G(Acc)$ is fundamentally constrained by the conditional accuracy. A higher value of $H(Y|X)$ implies greater uncertainty in the classification problem.

Clearly, we can indirectly characterize the accuracy of individual samples by minimizing the entropy.

4.2. Formalization of Entropy-Decrease-Based Selection

Given a set of samples $\mathcal{X} = \{x_1, x_2, \cdots, x_m\}$, an initial model F_0, and n candidate models $\{F_1, F_2, \cdots, F_n\}$, we define a selection function $S : \mathcal{X} \to \{0, 1, \cdots, n\}$ where $S(x_i)$ indicates which model ($F_{S(x_i)}$) should be used to make the prediction for x_i. Our initial optimization goal is to minimize the total conditional entropy over all samples:

$$\min_S \sum_{i=1}^{m} H(F_{S(x_i)}(x_i)) \tag{3}$$

Let $\Delta H\big(F_j(x_i)\big)$ represent the change in conditional entropy when using model F_j instead of the original model F_0 for sample x_i:

$$\Delta H\big(F_j(x_i)\big) := H\big(F_0(x_i)\big) - H\big(F_j(x_i)\big) \tag{4}$$

Clearly, $\sum_{i=1}^{m} H(F_0(x_i))$ is a constant with respect to the function S. Thus:

$$\begin{aligned}&\min_S \sum_{i=1}^{m} H\big(F_{S(x_i)}(x_i)\big)\\ &= \sum_{i=1}^{m} H\big(F_0(x_i)\big) - \max_S \sum_{i=1}^{m} \Delta H\big(F_{S(x_i)}(x_i)\big)\end{aligned} \tag{5}$$

Therefore, our task is transformed into maximizing the total decrease in entropy across all samples.

Considering resource constraints and time budget, we also have a constraint on the function S. Let $k\%$ be the filtering ratio. Thus a restriction is imposed on the number of samples that select a non-original model:

$$\frac{|\{i : S(x_i) \neq 0\}|}{m} \leq k\% \tag{6}$$

By adjusting the value of k, our approach can accommodate dynamic resource-constrained scenarios, which significantly enhance its operational robustness.

Overall, in this section, we formalize the optimization problem representing our approach:

$$\begin{aligned}&\max_S \sum_{i=1}^{m} \Delta H\big(F_{S(x_i)}(x_i)\big)\\ &\text{s.t.}\frac{|\{i : S(x_i) \neq 0\}|}{m} \leq k\%\end{aligned} \tag{7}$$

4.3. The Uncertainty Prediction-Base Router

Following the formal analysis above, we train a set of predictors, denoted as $h_0, h_1, \cdots, h_n$, to estimate the entropy decrease $\Delta\hat{H}(F_j(x_i))$

Notably, our predictors operate on the probability distribution generated by the original model F_0 rather than the raw sample. This distribution acts as a distilled feature set, capturing essential information, particularly the model's decision tendencies. Furthermore, as a by-product of previous inference steps, this distribution is readily available and offers a low-dimensional, lightweight alternative to processing raw sample features.

Based on the predictors, we can estimate the appropriate function S_0 that maximizes the total entropy decrease while

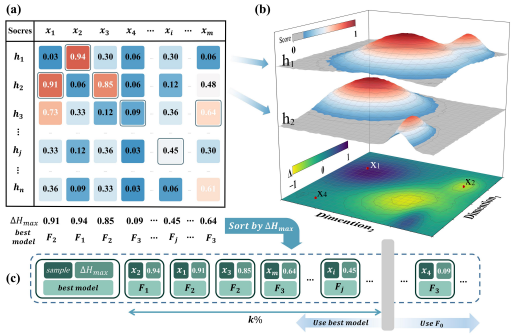

Figure 4. Overview of the Proposed Framework

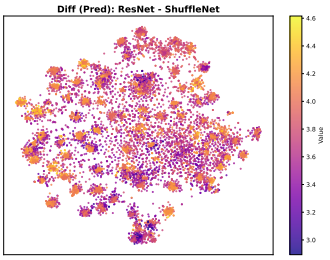

Figure 5. Predict Entropy Decrease Difference on the t-SNE manifold

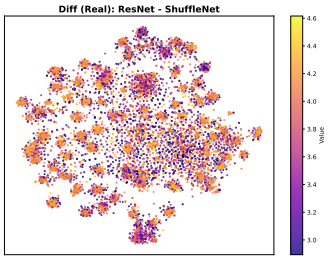

Figure 6. Real Entropy Decrease Difference on the t-SNE manifold

Algorithm 1: Adaptive Model Selection Strategy

Require: Trained predictors $\{h_i\}$, dataset $\mathcal{D}'$, models $\{F_0, F_1, \ldots, F_n\}$, previous outputs array $\mathbf{p_0}$, threshold ratio $k\%$
Ensure: Updated prediction distributions

```
Initialize scores ← [], indices ← []
for m = 0 to |D'| − 1 do
    x ← D'[m]; p_prev ← p_0[m];
    ΔH_max ← −∞, i_best ← 0;
    for i = 1 to n do
        ΔH_i ← f_i(p_prev);
        if ΔH_i > ΔH_max then
            ΔH_max ← ΔH_i; i_best ← i;
        end if
    end for
    scores ← scores ∪ {ΔH_max};
    indices ← indices ∪ {i_best}
end for
τ ← TopPercentile(scores, k%);
output_distribution ← p_0;
for m = 0 to |D'| − 1 do
    if scores[m] ≥ τ then
        i_sel ← indices[m]; x ← D'[m];
        p_update ← F_{i_sel}(x);
        output_distribution[m] ← p_update
    end if
end for
```

satisfying the constraints. The estimation follows the procedure below, just as shown in Figure 4.

• Firstly, for each sample x_i, we calculate the output results predicted by each predictorh_j with the result $h_j(x_i)$ denoted as a_{ij}. Subsequently, we further obtain the maximum entropy decrease b_i and the index of the most suitable model predicted for x_i denoted as $j^{(i)}$: $b_i := \max_j a_{ij}$, $j^{(i)} := \arg\max_j h_j(x_i)$

• Secondly, We sort the set $\{b_i\}_{i=1}^m$ in descending order and then select the top $\lfloor k\% \cdot m \rfloor$ elements $b_{i_1}, b_{i_2}, \ldots, b_{i_{\lfloor k\% \cdot m \rfloor}}$. This yields the index set $\mathcal{I} := \{i_t\}_{t=1}^{\lfloor k\% \cdot m \rfloor}$.

• Lastly, we update the selected samples and leave the labels of all others unaltered. For each updated sample x_i, let $F_0^{(1)}(x_i)$ denote the updated model. Then:

$$F_0^{(1)}(x_i) = \begin{cases} F_{j^{(i)}} \ , \ i \in \mathcal{I} \\ F_0 \ , otherwise \end{cases} \tag{8}$$

Consequently, we determine the model used for each sample after the update, which will output the updated class label of each sample. That is, we obtain the values of $S_0(x_i)$, for $i = 1, 2, \ldots, m$, estimating the appropriate function S_0.

4.4. Extension to Multi-Round Updates

In practical applications, model updates are rarely isolated events; rather, they typically occur sequentially. This section extends our framework to the scenario of multi-round updates.

4.4.1. Iterative Update Formulation

Each update round strictly adheres to the single-round protocol detailed in the preceding section. Specifically, for the $(t+1)-th$ round, the optimization is initialized with the model resulting from the $t-th$ round, denoted as $F_0^{(t)}(x_i)$. Let $j_{t+1}^{(i)}$ denote the index of the most suitable model predicted for x_i and let $\mathcal{I}^{t+1}$ represent the set of indices designated for updating. We thus establish the following recurrence relation:

$$F_0^{(t+1)}(x_i) = \begin{cases} F_{j_{t+1}^{(i)}} \ , \ i \in \mathcal{I}^{t+1} \\ F_0^{(t)}(x_i) \ , otherwise \end{cases} \tag{9}$$

In particular, for the initial update round, this relationship reduces to the recurrence relation established for the single-round scenario, as given in Equation (12).

Notably, for a candidate model F_j in the $(t+1)-th$ round, the associated predictor h_j takes the probability distribution output by the global initial model F_0 as input, rather than that of the current round's initialization, sample-specific $F_0^{(t)}$ Consequently, $\Delta\hat{H}(F_j(x_i))$, the entropy reduction for x_0 predicted by h_j, is intrinsically measured relative to F_0.

4.4.2. Multi-Stage Screening and Re-classification for Omitted Samples

Ideally, all misclassified samples should be identified and updated. In a single-round update, omitting valid candidates is an acceptable trade-off due to computational constraints. However, if such samples remain uncorrected after multiple update rounds, this indicates a systemic deficiency in our selection strategy. For a given sample x_0, the mechanisms that lead to this persistent omission are categorized into two primary scenarios: Marginal Exclusion caused by limited computational resources, and the Overconfidence Trap resulting from the model's high confidence in incorrect answers.

To rectify persistent sample omission, we propose a two-stage screening mechanism to identify high-risk samples that necessitate rigorous re-classification via multi-model voting. Notably, to maintain a manageable computational budget, we adopt an adaptive Top-K selection strategy.

Stage 1: Omission Risk Screening. We identify samples with the highest omission risk by ranking all m instances according to an omission risk function $\mathcal{R}_{\text{Omission}}(x_i)$, which is constructed to account for the two failure scenarios previously discussed (refer to Appendix F.1 for a detailed derivation of its components):

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

where the score $S_{t,i}$ represents the normalized entropy reduction at round t for sample x_i, relative to the global baseline F_0, achieved by the candidate model $F_{j_t^{(i)}}$ that exhibits the maximum entropy drop:

$$S_{t,i} = \frac{1}{2}\left[1 + \tanh\left(\Delta\hat{H}(F_{j_t^{(i)}}(x_i))\right)\right] \tag{10}$$

To accurately capture the cumulative risk of marginal exclusion, the summation interval for the first term spans from the round immediately following the last effective update, denoted as $\tau(i)$, up to the current round T.

Let $\rho_{\text{screen}}^{(t)}$ denote the adaptive screening ratio at round t, which increases monotonically with the round index to reflect the growing evidence of persistent omission:

$$\rho_{\text{screen}}^{(t)} = \min\left(\rho_{\text{base}} + \alpha \cdot (t-2), \rho_{\max}\right) \tag{11}$$

where ρ_{base} is the initial screening ratio, α controls the growth rate, and $\rho_{\max}$ serves as the upper bound. The resulting preliminary screening set $\mathcal{X}_1$ is defined as:

$$\mathcal{X}_1 = \{x_i \mid i \in \text{Top-}\lfloor m \cdot \rho_{\text{screen}}^{(t)} \rfloor \text{ of } \mathcal{R}_{\text{Omission}}\} \tag{12}$$

Stage 2: Misclassification Risk Filtering. For each sample $x_i \in \mathcal{X}_1$, we quantify its prediction instability using the Misclassification Risk Function based on Jensen-Shannon divergence:

$$\mathcal{R}_{\text{misclass}}(x_i) = \sum_{k=1}^{C}\left[p_k \log\frac{2p_k}{p_k + p_k'} + p_k' \log\frac{2p_k'}{p_k + p_k'}\right] \tag{13}$$

where $p(x_i) = [p_1, \ldots, p_C]$ denotes the probability distribution output by the model currently assigned to x_i, and $p'(x_i) = [p_1', \ldots, p_C']$ represents the distribution from an alternative model randomly sampled from the current round. High JS divergence indicates unstable predictions that require correction.

We apply a second Top-K selection on $\mathcal{X}_1$ with an adaptive correction ratio $\rho_{\text{correct}}^{(t)}$:

$$\rho_{\text{correct}}^{(t)} = \min\left(\gamma_{\text{base}} + \gamma \cdot (t-2), \gamma_{\max}\right) \tag{14}$$

The final correction set $\mathcal{X}_2$ is obtained by:

$$\mathcal{X}_2 = \left\{x_i \mid i \in \text{Top-}\lfloor|\mathcal{X}_1| \cdot \rho_{\text{correct}}^{(t)}\rfloor \text{ of } \mathcal{R}_{\text{misclass}}\right\} \tag{15}$$

Multi-Model Voting. Finally, for each sample $x_i \in \mathcal{X}_2$, we employ an ensemble of recent models $\{F_{t-w}, \ldots, F_t\}$ to perform classification and aggregate their predictions via majority voting:

$$\hat{y}_i^{\text{corrected}} = \arg\max_y \sum_{j=t-w}^{t} \mathbb{1}\left[\arg\max F_j(x_i) = y\right] \tag{16}$$

where w denotes the voting window size (typically 2-3 recent rounds, thiss ensemble approach leverages diverse model perspectives to produce more robust predictions for high-risk samples.

This multi-stage screening mechanism effectively rectifies persistent sample omission while avoiding computationally expensive voting operations on the entire dataset, thereby significantly reducing resource consumption.

5. Experiments

We implemented our entropy-decrease prediction framework using PyTorch 2.2 and deployed it on a server with a single A40 GPU. The router network consists of a three-layer MLP with ReLU activation, using the Adam optimizer with a learning rate of 0.001 and a batch size of 32. The training protocol involves 100 epochs of optimization with MSE loss. In the basic scenario, the router has a short training time (15 minutes) and a small model size (0.02MB).

5.1. Experimental Setup

Datasets We evaluate PURE across multiple modalities and tasks: (1) Image Classification: CIFAR-10, CIFAR-100 (Krizhevsky & Hinton, 2009), and ImageNet; (2) Object Detection: MS COCO 2017; and (3) Text Classification: AG

News (Zhang et al., 2015) and IMDb (Maas et al., 2011). Detailed dataset statistics and preprocessing steps are provided in Appendix C.

Baselines To provide a comprehensive evaluation of our proposed approach, we compare it against several representative baseline methods, including **CEMA**(Chen et al., 2024), **DKD**(Li et al., 2021), and **MLink**(Yuan et al., 2022). To ensure a fair comparison and eliminate confounding factors such as dataset distribution or filtering volume, we adopt consistent filtering ratios (20% and 50%) across all experiments and use an identical test set for evaluation. Detailed specifications of the baseline implementation can be found in the Appendix C.5.

Table 2. Control experiment on CIFAR-10 and CIFAR-100. The 20%/50% indicates the proportion of samples directly using previous results. All results are reported in the format of Prediction Accuracy (Acc.%) / Result Consistency (Cons.%).

	CIFAR-10		**CIFAR-100**	
	20%	50%	20%	50%
CEMA	91.6/94.2	88.9/90.9	79.3/96.9	74.2/87.8
DKD	94.8/97.7	88.2/91.0	79.4/97.7	75.5/89.7
Mlink	95.6/98.9	89.9/92.2	79.3/96.9	74.2/87.9
Ours	**96.3/99.9**	**95.3/98.7**	**80.3/98.5**	**78.3/94.0**

5.2. Scheduling under Single-task Setting

We compare our proposed method with the baselines introduced in the previous Section 5.1, with the results presented in Table 2. In this experiment, we focus on a single scheduling scenario: given the predictions from a round-1 model, how to selectively filter samples for re-inference in round 2 to achieve acceleration. For each dataset, we train both round-1 and round-2 models with identical architectures, specifically using ResNet backbones. Detailed architectural configurations can be found in the Appendix C.4.

On the CIFAR-10 dataset, the round-1 model achieves an accuracy of 81.39%, while the round-2 model reaches 96.41%. On CIFAR-100, the round-1 model obtains 61.17% accuracy, and the round-2 model achieves 80.74%. We adopt both accuracy and output consistency as the evaluation metrics. Below is a formal definition of consistency:

Let the round-2 model output be $a_1, a_2, ..., a_m$ and the final output after filtering and scheduling be $b_1, b_2, ..., b_m$(a_i and b_i denotes the numeric class number). Consistency is defined as:

$$\frac{1}{m}\sum_{i=1}^{m}(1-\text{sgn}(|a_i-b_i|)) \tag{17}$$

As shown, our method outperforms the baselines. Under a 20% filtering ratio, the accuracy remains virtually unchanged. Even under a 50% filtering ratio, halving the round-2 computation, the accuracy drops by only 2%, reaching 78.32%. On CIFAR-10, our method achieves 95.32% accuracy, a mere 1% drop compared to full round-2 execution, and notably higher than the best baseline at 89.92%. Local consistency also supports this trend, with consistency scores reaching 99.87% under 20% filtering and 98.71% under 50%. In addition, our router achieves very fast inference speed. A batch (32 samples) takes only 1.95 ms.

Since our method is applied *post-inference*, it is orthogonal and complementary to most existing acceleration techniques, which typically focus on single-pass model optimization or compression. This modularity highlights the portability of our method, allowing seamless integration with state-of-the-art inference speed-up strategies. Figure 7 illustrates the trade-off curves between accuracy and consistency across varying filtering levels in both datasets, underscoring its ability to effectively balance execution efficiency and prediction fidelity.

5.3. Multi-Expert Scheduling on CIFAR-100

We evaluate multi-expert scheduling on CIFAR-100 using three specialized experts targeting the *flower*, *tree*, and *insect* superclasses, respectively. Each expert achieves over 80% accuracy within its specific domain while maintaining only $\sim$20% accuracy across remaining classes. Experimental results demonstrate that by maximizing predicted entropy reduction, the router significantly outperforms the Round-1 baseline under resource constraints: accuracies for *flower*, *tree*, and *insect* tasks improved from 69.5%, 68.0%, and 68.2% to 77.5%, 73.0%, and 78.0%, respectively.

To benchmark performance without ground-truth access, we compare our approach against a **Random Policy** and a ground-truth-aware **Oracle Policy**. As shown in Figure 9, the router demonstrates high robustness as the filtering threshold increases; notably, when the threshold exceeds zero, its performance nearly matches or even surpasses the Oracle strategy, achieving near-optimal model selection without requiring external labels. Complete details are provided in the Appendix D.1.

5.4. Multi-Round Progressive Deployment

To evaluate PURE in realistic sequential update scenarios, we conduct a multi-round deployment experiment on CIFAR-100, simulating edge systems where models are iteratively refined. At each round t, our router selects 50% of samples for upgrade based on entropy decrease predictors. Starting from Round 2, the multi-stage correction mechanism activates to address persistent errors via omission screening, misclassification risk quantification (JS divergence), and multi-model voting. Further details on the experimental setup are provided in the Appendix C.7.

Baselines. We compare PURE against Naive (uniform selection), Entropy (highest prediction entropy), CEMA (Chen

Table 3. 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)	**68.89**	**+2.28**
– 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). Selection rate: 50% per round.

et al., 2024), MLink (Yuan et al., 2022), and Optimal (theoretical upper bound). Ablation variants include w/o Correction (predictor-only), Only Omission, Only Misclass, and No Voting.

Overall Performance. As shown in Table 3, PURE achieves 78.87% accuracy in Round 4, outperforming all baselines, including a +1.71% gain over MLink (77.16%) and +2.71% over the Naive approach (76.16%). PURE reaches 97.7% of the optimal performance (80.75%), demonstrating near-optimal sample selection efficiency within a 50% computational budget.

Progressive Trajectory. Round-by-round analysis reveals that while early rounds (0→2) show modest gains due to marginal model differences, Round 2→3 exhibits a dramatic leap (+14.75% for PURE) following the introduction of high-quality M_3. Notably, PURE maintains consistent improvement in Round 4 (+1.34%), whereas the gap between PURE and MLink widens, highlighting the correction mechanism's role in sustained performance.

Correction Efficacy. Ablation results quantify the necessity of the multi-stage design. Removing the correction mechanism entirely results in a 1.97% performance degradation. Misclassification risk detection emerges as the most critical component; its absence (*Only Omission* variant) reduces accuracy by 0.79%, while excluding omission screening (*Only Misclass* variant) leads to a more significant 1.42% drop compared to the full model. The full correction gain (+1.97%) exceeds the sum of individual components, indicating a strong synergistic effect between screening and voting.

5.5. Comprehensive Performance Analysis

To evaluate the effectiveness and generality of PURE, we extend our evaluation beyond visual classification to text classification and object detection, covering diverse modalities and architectures (e.g., ViT, Qwen(Dosovitskiy et al., 2021; Bai et al., 2023)). As shown in Table 7, we simulate common model update scenarios across these domains. To accommodate non-classification outputs, we developed task-specific adaptation strategies (detailed in Appendix D.2).

The results confirm that our uncertainty-prediction-based router achieves an optimal balance between efficiency and accuracy. In a conservative 20% filtering scenario, performance remains nearly identical to full re-inference, maintaining up to 99.89% consistency on ImageNet. Even under aggressive 50% filtering, PURE exhibits graceful degradation; for instance, the COCO mAP@50 decreases only from 71.5% to 65.7%, halving computation for an acceptable accuracy cost . This demonstrates PURE's utility in allowing users to dynamically balance their computational budget with task fidelity.

5.6. Real-world Experiment

In this phase, we simulate the real-world scenario of a smart photo album on mobile devices. As shown in Table 4, for object detection and face detection using IoU (95%) as the metric, our method achieves 24.05% and 21.20%. This shows our approach works effectively in object-related detection with high-precision overlap measurement. For object classification and text classification (OCR) using consistency, the results are 58.45% and 40.25%. The difference in performance between the object detection task and the

Table 4. Maximum filtering rates across tasks on mobile photo album datasets under 95% consistency threshold. "Cons" represents the consistency in equation 17, "IoU" represents the Intersection-over-Union.

Task	Model in Rounds		Evaluation Metrics		Filter
Description	Round1	Round2	Metric	Type	Rate
Object Detection	Yolov8n(Ultralytics, 2023)	Yolov8x(Ultralytics, 2023)	IoU(95%)	Detection	24.05%
Face Detection	MTCNN(Zhang et al., 2016)	Yolo-face(Jocher & Qiu, 2023)	IoU(95%)	Detection	21.20%
Scene Classification	Resnet(He et al., 2016)	Yolov8s(Ultralytics, 2023)	Cons.(95%)	Classification	58.45%
Text Classification	Qwen1.5(Bai et al., 2023)	Qwen3(Team, 2025)	Cons.(95%)	Classification	40.25%

Table 5. 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**	–

Table 6. Performance and time cost under different training sample sizes on CIFAR-100. Acc/Cons denotes Accuracy/Consistency. Pre/Train indicates the time in seconds for preprocessing and training.

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

classification task may be due to the difference in the strictness between the metric of IoU >95% and the metric of consistency >95%. However, overall, each task can achieve a speed-up of at least about 20%. Complete details are provided in the Appendix D.3.

5.7. Ablation Study

Hardware Performance and Acceleration We benchmark the router's efficiency across three heterogeneous platforms: a high-performance server (Intel Xeon), a personal workstation (Apple M4), and a mobile environment (A17 Pro). As illustrated in Figure 11, the router maintains a minimal memory footprint (0.02MB) and achieves sub-millisecond inference latency (0.014–0.072ms) across all tested devices.

Notably, our method achieves 28.9%–43.3% end-to-end latency reduction. We observe a gap between the practical speedup and the 50% theoretical filtering rate, which we attribute to the approximately 10% computational overhead introduced by the naive global sorting algorithm used for thresholding. While integrating advanced streaming top-k algorithms (Charikar et al., 2002) could further bridge this gap, the current implementation already demonstrates substantial acceleration potential for real-world edge computing scenarios.

Quantization and Sparsification Strategies To minimize the storage footprint of historical inference distributions on resource-constrained edge devices, we evaluate two complementary compression strategies: **precision quantization** and **top-k sparsification**. As shown in Table 10, 4-bit quantization (discretizing probabilities into 2^4 levels) combined with top-1 selection reduces the per-sample storage from 400 bytes to 1.5 bytes. This achieves a **99.5% compression ratio** with a marginal accuracy degradation of less than 1%. These results validate that storing sparse probability vectors is an effective approach for maintaining routing fidelity while respecting hardware storage constraints.

Data Efficiency and Training Overhead We investigate the router's sensitivity to training data scale by varying the sample size on CIFAR-100 from 50,000 to 1,000 (Table 10). The router exhibits significant **data efficiency**: reducing the training set by 90% (from 50k to 5k) results in negligible performance loss, maintaining 78.4% accuracy and 94.0% consistency. Furthermore, both preprocessing (Pre) and training (Train) times scale sub-linearly with the data size. This lightweight overhead enables the router to be rapidly retrained to adapt to frequent base-model updates in dynamic deployment environments.

6. Conclusion and Discussion

In this work, we propose a lightweight inference-filtering router that can be deployed at the edge. We use historical execution data as a basis to achieve precise and efficient routing by predicting the magnitude of uncertainty change. Our method features light-weight (0.02MB), fast inference speed (1.95ms per batch(32 samples)), easy deployment, and easy training. However, we still summarize the following limitations and issues: when facing unreliable random models or models with significant differences in capabilities, scheduling cannot be performed. The first-round models need to have a relatively reasonable uncertainty assessment of their own results. How to measure "schedulability" is a worthwhile research question; we will continue to study it.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

Abdelhalim, A. M. and Salem, M. A.-M. Intelligent organization of multiuser photo galleries using sub-vent detection. In *International Conference on Computer Engineering and Systems*, pp. 436–440, 2017.

Bai, J., Bai, S., Chu, Y., Cui, Z., Dang, K., Deng, X., Fan, Y., Ge, W., Han, Y., Huang, F., Hui, B., Ji, L., Li, M., Lin, J., Lin, R., Liu, D., Liu, G., Lu, C., Lu, K., Ma, J., Men, R., Ren, X., Ren, X., Tan, C., Tan, S., Tu, J., Wang, P., Wang, S., Wang, W., Wu, S., Xu, B., Xu, J., Yang, A., Yang, H., Yang, J., Yang, S., Yao, Y., Yu, B., Yuan, H., Yuan, Z., Zhang, J., Zhang, X., Zhang, Y., Zhang, Z., Zhou, C., Zhou, J., Zhou, X., and Zhu, T. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

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

Barbuto, V., Savaglio, C., Minerva, R., Crespi, N., and Fortino, G. Towards an edge intelligence-based traffic monitoring system. *2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, abs/2403.12976, 2023.

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

Chen, Y., Niu, S., Wang, Y., Xu, S., Song, H., and Tan, M. Towards robust and efficient cloud-edge elastic model adaptation via selective entropy distillation. In *International Conference on Learning Representations*, 2024.

Cormode, G. and Muthukrishnan, S. 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.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Elgaar, M. and Amiri, H. Hucurl: Human-induced curriculum discovery. *arXiv preprint arXiv:2307.07412*, 2023.

Frantar, E., Ashkboos, S., Hoefler, T., and Alistarh, D. Gptq: Accurate post-training quantization for generative pre-trained transformers. *Computing Research Repository*, 2022.

Gerchinovitz, S., Menard, P., and Stoltz, G. Fano's inequality for random variables. *Statistical Science*, 35(2), 2020.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks, 2017. URL https://arxiv.org/abs/1706.04599.

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

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. pp. 770–778, 2016.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *NeurIPS Workshop*, 2015.

Ilic, M., Ivanovic, M., Kurbalija, V., and Valachis, A. Towards optimal learning: Investigating the impact of different model updating strategies in federated learning. *Expert Systems with Applications*, pp. 123553–123553, 2024.

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., and Kalenichenko, D. Quantization and training of neural networks for efficient integer-arithmetic-only inference. *International Conference on Computer Vision*, 2018.

Jia, F., Jiang, S., Cao, T., Cui, W., Xia, T., Cao, X., Li, Y., Wang, Q., Zhang, D., Ren, J., Liu, Y., Qiu, L., and Yang, M. Empowering in-browser deep learning inference on edge through just-in-time kernel optimization. *PROCEEDINGS OF THE 2024 THE 22ND ANNUAL INTERNATIONAL CONFERENCE ON MOBILE SYSTEMS, APPLICATIONS AND SERVICES, MOBISYS 2024*, 2024a.

Jia, Y., Liu, B., Zhang, X., Dai, F., Khan, A., Qi, L., and Dou, W. Model pruning-enabled federated split learning for resource-constrained devices in artificial intelligence empowered edge computing environment. *ACM Transactions on Sensor Networks*, 2024b.

Jin, X., Liu, H. H., Gandhi, R., Kandula, S., Mahajan, R., Zhang, M., Rexford, J., and Wattenhofer, R. Dynamic scheduling of network updates. In *Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pp. 539–550, 2014.

Jocher, G. and Qiu, J. YOLO by Ultralytics, January 2023. URL https://github.com/ultralytics/ultralytics.

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

Kurtic, E., Kuznedelev, D., Frantar, E., Goin, M., and Alistarh, D. Sparse fine-tuning for inference acceleration of large language models. *CoRR*, abs/2310.06927, 2023.

Li, H., Han, H., and Zhou, S. K. Mixed-order self-paced curriculum learning for universal lesion detection. *arXiv preprint arXiv:2302.04677*, 2023.

Li, L., Lin, Y., Ren, S., Li, P., Zhou, J., and Sun, X. Dynamic knowledge distillation for pre-trained language models. In *Conference on Empirical Methods in Natural Language Processing*, 2021.

Li, X., Li, Y., Li, Y., Cao, T., and Liu, Y. Flexnn: Efficient and adaptive dnn inference on memory-constrained edge devices. In *ACM/IEEE International Conference on Mobile Computing and Networking*, 2024.

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

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

Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., and Zhang, C. Learning efficient convolutional networks through network slimming. In *IEEE International Conference on Computer Vision*, 2017.

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

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

Mishra, C., Sarma, H., and Saravanan, M. Pagellm: Incremental approach for updating a security knowledge graph by using page ranking and large language model. *Information Processing and Management*, 62(3), 2025.

Pan, R., Dai, Y., Zhang, Z., Oliaro, G., Jia, Z., and Netravali, R. Specreason: Fast and accurate inference-time compute via speculative reasoning. 2025.

Park, M., Lee, S., Yi, O., and Kim, J. A study on data acquisition based on the huawei smartphone backup protocol. *Forensic Science International Digital Investigation*, 41: 301393–301393, 2022.

Sawade, C., Landwehr, N., and Scheffer, T. Active comparison of prediction models. In *Conference on Neural Information Processing Systems*, pp. 1763–1771, 2012.

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

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

Ultralytics. Yolov8 documentation. 2023.

Xu, D., Yin, W., Jin, X., Zhang, Y., Wei, S., Xu, M., and Liu, X. Llmcad: Fast and scalable on-device large language model inference. *arXivorg*, abs/2309.04255, 2023.

Yin, W., Xu, D., Huang, G., Zhang, Y., Wei, S., Xu, M., and Liu, X. Piebridge: Fast and parameter-efficient on-device training via proxy networks. *ACM International Conference on Embedded Networked Sensor Systems*, pp. 126–140, 2024.

Yuan, M., Zhang, L., Li, X.-Y., and Xiong, H. Comprehensive and efficient data labeling via adaptive model scheduling. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pp. 1858–1861, 2020. doi: 10.1109/ICDE48307.2020.00188.

Yuan, M., Zhang, L., and Li, X.-Y. 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.

Yuan, M., Zhang, L., He, F., Tong, X., Song, M.-H., Xu, Z., and Li, X.-Y. Infi: End-to-end learning to filter input for resource-efficiency in mobile-centric inference. *IEEE TRANSACTIONS ON MOBILE COMPUTING*, 23 (5), 2024.

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

Zhang, X., Zhao, J., and LeCun, Y. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28, 2015.

Zhu, H., Kaffes, K., Chen, Z., Liu, Z., Kozyrakis, C., Stoica, I., and Jin, X. Racksched: A microsecond-scale scheduler for rack-scale computers. *Computing Research Repository*, pp. 1225–1240, 2020.

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) :

$$ {\mathbf{p_o}}^{t+1} - {\mathbf{p_o}}^t = -\eta \mathcal{M}^t(\mathbf{x}o)\, \mathcal{K}^t(\mathbf{x}o, \mathbf{x}u)\, \mathcal{G}^t(\mathbf{x}u, y_u) \tag{18} $$

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 [cite]. 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 establish that this change can be decomposed into the sum of a dominant first-order term (the MKG decomposition term) and a second-order remainder. The core of the proof lies in the observation that, regardless of the specific optimizer, the norm of the parameter update $\Delta\theta$ is always of the first order with respect to the learning rate η. Consequently, the second-order term in the Taylor expansion becomes $O(\eta^2)$. Under conditions of a small learning rate and conventional network training, this term is a higher-order infinitesimal relative to the first-order term and can therefore be neglected, guaranteeing the theoretical validity of our approximation.

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 $\|\mathcal{G}^t(x_u, y_u)\|_2 \le G, \|\mathbf{J}^t(x)\|_2 \le J, \|\mathcal{M}^t(x)\|_2 \le M$. 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 $\|\nabla^2_\theta f_\theta(x)\|_2 \le B$. 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 $\|D^{t+1}\|_2 \le D_{\max}$. 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:

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

where $\mathcal{K}^t(\mathbf{x}_o, \mathbf{x}_u) = \mathbf{J}^t(\mathbf{x}_o)\mathbf{J}^t(\mathbf{x}_u)^\top$ 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:

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

So the second-term T_2 satisfies:

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

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:

$$v^{t+1} = \gamma v^t + \left(\nabla_\theta \mathcal{L}(\mathbf{x}_u, y_u)|_{\theta^t}\right)^\top$$

$$\Delta\theta = -\eta v^{t+1}$$

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:

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

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:

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

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

$$\|T_2\|_2 \le \frac{1}{2}BJ^2G^2\left(\frac{\eta}{1-\gamma}\right)^2$$

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

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

Update the first- and second-moment estimates

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

Compute bias-corrected estimates

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

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:

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

where $\mathcal{K}_{\text{Adam}}^{t,i}(\mathbf{x}_o, \mathbf{x}_u) = \mathbf{J}^t(\mathbf{x}_o) D^{t+1} \mathbf{J}^i(\mathbf{x}_u)^\top$ 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 $\eta \to 0$.

Since, 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. Implementation and Hardware Details

C. Detailed Experimental Setup

C.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 & 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 (**?**) 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 D.3.

C.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.

C.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 confirms that the predictor successfully captures the distribution of uncertainty gains, validating its effectiveness as a decision proxy for model routing.

C.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.

C.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.

C.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 filtering volume, we adopt consistent filtering ratios (20% and 50%) across all experiments and use an identical test set for evaluation.

CEMA(Chen 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.

DKD(Li 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.

MLink(Yuan 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.

C.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.

C.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.

D. More Experiment Results

D.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 8 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:

$$Acc_{\text{random}} = (1-k) \cdot \left(\sum_{i=1}^{l} \frac{l \cdot q_i \cdot h_i}{m} \right) + \frac{k}{j} \cdot \sum_{j=1}^{m} \sum_{i=1}^{l} \left(1 - \frac{l \cdot h_i}{m} \right) \cdot p_i^j \tag{19}$$

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:

$$Acc_{\text{oracle}} = (1-k) \cdot \left(\sum_{i=1}^{l} \frac{l \cdot q_i \cdot h_i}{m} \right) + k \cdot \sum_{i=1}^{l} \left(1 - \frac{l \cdot h_i}{m} \right) \cdot \max_{1 \le j \le m} p_i^j \tag{20}$$

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 9 illustrates that as the threshold increases, the number of filtered samples exhibits a sharp decline, while our router demonstrates robust performance, nearly matching or even surpassing the oracle policy when the threshold exceeds zero.

D.2. Multi-task Scheduling

Table 7. A comprehensive performance comparison of our proposed filtering method across multiple datasets and tasks. The table simulate our filtering method in a model update scenario ('Round1' → 'Round2'). Alongside baseline accuracies, it shows the final accuracy ('Acc.') and consistency ('Cons.') when filtering 20% and 50% of inputs. Consistency is measured against full re-inference with the 'Round2' model. For the COCO dataset, the accuracy metric (*) is mAP@50.

	Round1		**Round2**		**20%**		**50%**	
	ModelType	Acc.	ModelType	Acc.	Acc.	Cons.	Acc.	Cons.
CIFAR-10	Resnet18	81.39	Resnet50	96.41	96.3	99.87	95.32	98.71
CIFAR-100	Resnet18	61.17	Resnet50	80.74	80.31	98.51	78.32	93.96
ImageNet	Resnet18	70.14	Resnet152	78.22	78.18	99.96	78.02	98.96
	EfficientNet-B0	75.48	EfficientNet-B7	84.50	84.52	99.93	84.27	98.92
	ViT-B/16	77.97	DeiT-B	82.23	82.27	99.89	81.98	98.77
AG News	Qwen1.5	64.14	Qwen3	85.01	84.98	99.96	82.72	94.01
	Qwen1.5	64.14	BERT	94.55	93.98	99.4	89.39	93.29
IMDB	Qwen1.5	51.53	Qwen3	91.56	89.25	97.59	80.65	88.80
	Qwen1.5	51.53	BERT	92.24	90.06	97.72	80.72	87.86
COCO	Yolov8n	53.9*	Yolov8x	71.5*	69.4*	93.0	65.7*	80.2

D.2.1. Task-Specific Adaptation Strategies

To maintain the universality of our uncertainty-based framework across different data modalities, we developed 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. Furthermore, for object detection tasks whose outputs include bounding box coordinates and objectness scores instead of a single probability distribution, we address the challenge by using the *average confidence decrease* of detected objects as a proxy for entropy reduction. This metric serves as the core indicator of whether a model update, such as a transition from YOLOv8n to YOLOv8x, will significantly alter the prediction quality for a given frame.

D.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 conservative updates, we observed that in the 20% filtering scenario, the accuracy drop is almost negligible (e.g., 96.3% vs. 96.41% on CIFAR-10). This indicates that our router accurately identifies "stable" samples—those that the baseline model (F_0) already handles with high confidence and correctly aligns with the ground truth. For these samples, the computational cost of Round 2 re-inference is redundant.

In terms of graceful degradation in aggressive filtering, when the filtering ratio is increased to 50%, the system effectively 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 ultra-low-power modes during charging-limited windows.

Finally, the results demonstrate significant flexibility across modalities. The consistency scores across AG News and IMDb (reaching up to 99.96%) suggest that text-based uncertainty is as predictable as visual uncertainty. This confirms that the relationship between model confidence and accuracy, mathematically formulated in Section 3, is a fundamental property of deep learning classifiers rather than a task-specific phenomenon.

D.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 use the intersected region of the bounding boxes as the detection area. For confidence scores, we select the lowest confidence among all bounding boxes. However, during our experiments, we found that using either the average or the minimum confidence score leads to converging results during training. Unlike previous experiments where a fixed filtering rate was used for updates, we adopted a new strategy: fixing the consistency level and varying the filtering threshold. Specifically, we iteratively adjusted the threshold from higher to lower values and identified the maximum filtering threshold that satisfies the condition. The experimental results showed that when the threshold meets a 95% update consistency criterion, the filtering rate becomes 24.05%, meaning 481 out of the total samples were not updated.

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.

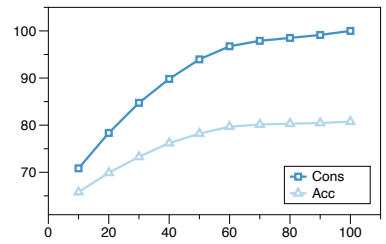

Figure 7. In CIFAR100, the changes in accuracy(Acc.) and consistency(Cons.) under different filtering ratios.

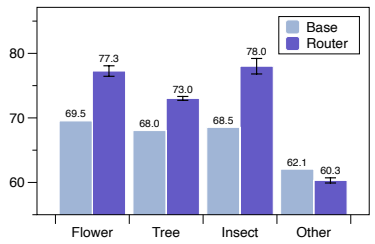

Figure 8. Comparison of the accuracy(Acc.) of the router and the base model under different super-classes.

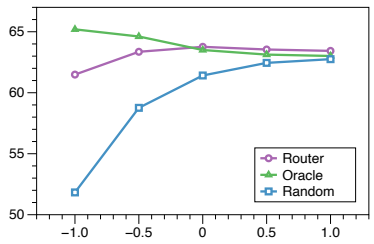

Figure 9. Comparison of the accuracy of the router and the other two policies under different thresholds

D.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 ($\rho_{correct}$).

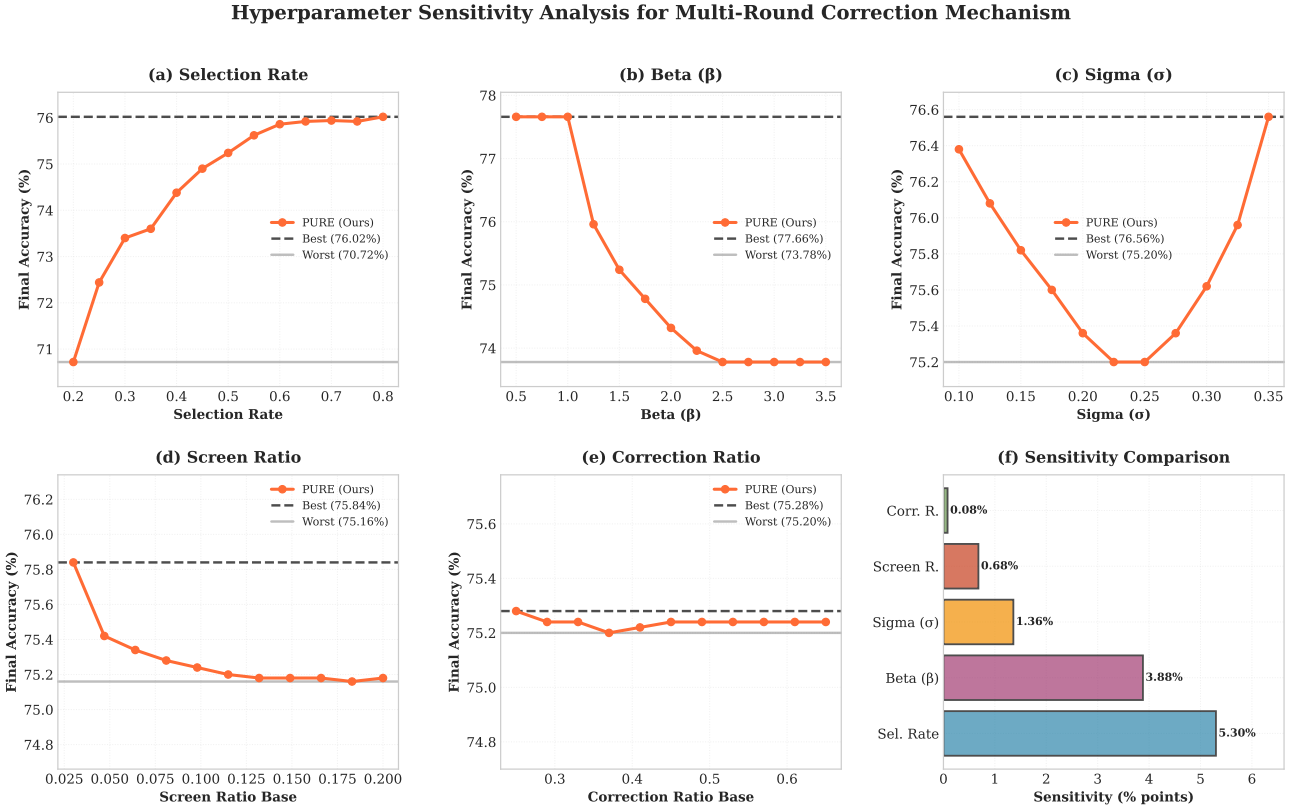

Figure 10. Comprehensive hyperparameter sensitivity analysis. (a-e) Impact of individual parameters on final accuracy (Round 4). The orange line represents our method (PURE), while the black dashed and gray solid lines indicate optimal and worst-case bounds. (f) Sensitivity comparison showing the max-min accuracy variation for each parameter.

High Sensitivity Parameters. As shown in Figure 10(f), the **Overconfidence Penalty** (β) exhibits the highest sensitivity (3.88%). Figure 10(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 10(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 10(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 10(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 10(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 8 summarizes the optimal configurations derived from this analysis.

Table 8. 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%)

D.5. Ablation Study

Table 9. 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**	–

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: the per-sample space decreases from 400 bytes to 1.5 bytes (storing only the 7-bit index of the top-1 label), yielding a 99.5% compression ratio with less than 1% accuracy degradation. 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.

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 10.

Table 10. Performance and time cost under different training sample sizes on CIFAR-100. Acc/Cons denotes Accuracy/Consistency (%), while Pre/Train indicates the time in seconds for preprocessing and training.

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

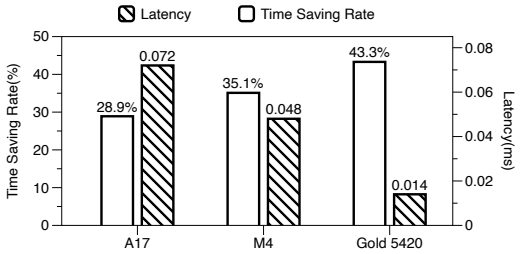

Figure 11. Performance on different hardware platforms.

*Data Efficiency: Effective with Minimal Data.*The results in Table 10 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 10, 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 11.

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 effectively filtering redundant 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 (e.g., 43.3% on the server) and the theoretical filtering rate (e.g., 50%). This gap is primarily due to the overhead introduced by the strategy used in our algorithm to determine the filtering threshold. 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. While straightforward, this step itself consumes approximately 10% of the additional computation time, leading to the difference between theoretical and practical savings rates.

We acknowledge that the top-k query algorithm in data streams has been extensively researched (Charikar et al., 2002; Cormode & 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.

E. More Related Works

E.1. Model Compression

Model compression techniques have evolved from basic approaches to advanced paradigms that integrate dynamic adaptation and hardware awareness(Li, 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.

E.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 & 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.

E.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.

F. Discussion

F.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.

F.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 with a reduced entropy relative to the initial model. However, the improvement score may consistently fail to surpass the Top-K threshold in any given round. We consider this a trivial case, as the exclusion is due to relative ranking rather than intrinsic metric failure. Typically, this issue mitigates naturally 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 that is confidently wrong. Since the conditional entropy is low, our selection function misinterprets the prediction as accurate and excludes from the update set. Based on the mechanism detailed in Section 3.1, such confident errors arise from a coincidental alignment of negative interference that concentrates the prediction probability mass of x_0 onto a single incorrect class y_u. While rare, once a model falls into this state, it almost inevitably leads to persistent misclassification of x_0 without triggering a corrective update.

F.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:

$$\mathcal{R}_{\text{Omission}}(x_i) = \underbrace{\sum_{t=\tau(i)+1}^{T} S_{t,i}}_{\text{Term I: Marginal Exclusion}} + \underbrace{\beta \cdot \exp\left(-\frac{S^2_{\tau(i),i}}{2\sigma^2}\right)}_{\text{Term II: The Overconfidence Trap}} \tag{21}$$

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:

$$S_{t,i} = \frac{1}{2}\left[1 + \tanh\left(\Delta\hat{H}(F_{j_t^{(i)}}(x_i))\right)\right] \tag{22}$$

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:

$$\mathcal{R}(x_i) \propto \exp\left(-\frac{(S_{\tau(i),i} - \mu_i)^2}{2\sigma^2}\right) \tag{23}$$

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:

$$\begin{aligned}\mathcal{R}(x_i) &= \exp\left(\frac{-S_{\tau(i),i}^2 + 2S_{\tau(i),i}\mu_i - \mu_i^2}{2\sigma^2}\right) \\ &= \underbrace{\exp\left(-\frac{\mu_i^2}{2\sigma^2}\right)}_{\text{I. Sample Constant}} \cdot \underbrace{\exp\left(\frac{S_{\tau(i),i}\cdot\mu_i}{\sigma^2}\right)}_{\text{II. Linear Interaction}} \cdot \underbrace{\exp\left(-\frac{S_{\tau(i),i}^2}{2\sigma^2}\right)}_{\text{III. Quadratic Decay}}\end{aligned} \tag{24}$$

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:

$$\text{Term II} = \beta \cdot \exp\left(-\frac{S_{\tau(i),i}^2}{2\sigma^2}\right) \tag{25}$$

F.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:

$$\mathcal{R}_{\text{misclass}}(x_i) = \text{JS}(p||p') = \frac{1}{2}\text{KL}(p||M) + \frac{1}{2}\text{KL}(p'||M) \tag{26}$$

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.

F.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 :

$$H(Y|X) \leq H(P_e) + P_e \log(k-1) \tag{27}$$

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:

$$G(Acc) := Acc - \frac{H(1-Acc)}{\log(k-1)} \leq 1 - \frac{H(Y|X)}{\log(k-1)} \tag{28}$$

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:

$$E = \begin{cases} 1, & \text{if } \hat{Y} \neq Y \\ 0, & \text{if } \hat{Y} = Y \end{cases} \tag{29}$$

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

$$\begin{aligned} H(E, Y|\hat{Y}) &= H(Y|\hat{Y}) + H(E|Y, \hat{Y}) \\ &= H(E|\hat{Y}) + H(Y|E, \hat{Y}) \end{aligned} \tag{30}$$

Noting that:

$$H(E|Y, \hat{Y}) = 0 \quad \text{and} \quad H(E|\hat{Y}) \leq H(E) = H(P_e) \tag{31}$$

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

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

$$\begin{aligned} H(Y|E, \hat{Y}) &= (1 - P_e)H(Y|\hat{Y}, E = 0) + P_e H(Y|\hat{Y}, E = 1) \\ &\leq (1 - P_e) \cdot 0 + P_e \log(K - 1) \\ &= P_e \log(K - 1) \end{aligned} \tag{32}$$

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 (30) through (5):

$$H(Y|\hat{Y}) \leq H(P_e) + P_e \log(K - 1) \tag{33}$$

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

$$I(X;Y) \geq I(X;\hat{Y}) \implies H(Y|X) \leq H(Y|\hat{Y}) \tag{34}$$

(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:

$$H(Y|X) \leq H(P_e) + P_e \log(K - 1) \tag{35}$$

□

F.3. Discussions and Qualitative Analysis

F.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.

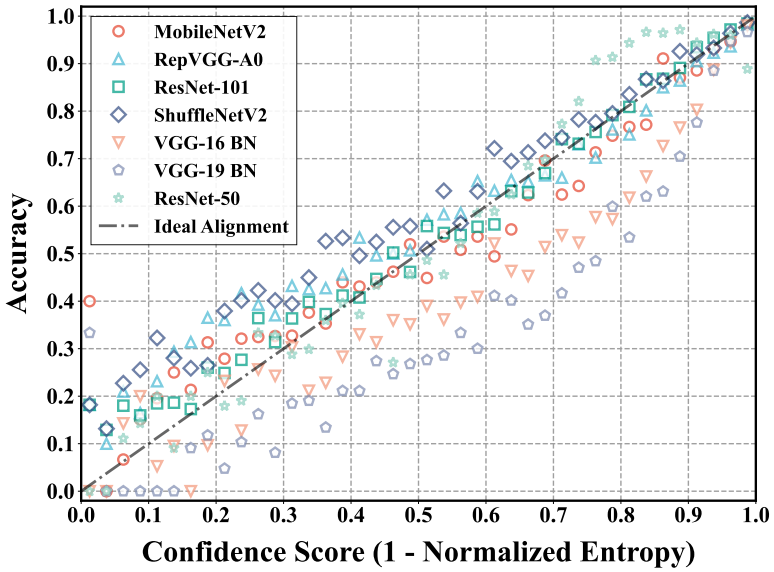

Figure 12. Reliability Diagram of the base model. The close alignment between the bar height (accuracy) and the diagonal line (perfect calibration) indicates that confidence is a reliable proxy for correctness in our setting.

As shown in the Reliability Diagram (Figure 12), our base models exhibit good calibration, meaning that high confidence generally correlates with high accuracy. Statistically, "confident errors" constitute a small minority of the data distribution (typically $< 1\%$ in our experiments). However, in a multi-round setting, these rare errors can accumulate. A 1% error rate that persists over 10 rounds becomes significant. This provides the theoretical motivation for our Multi-Stage Correction mechanism. By explicitly modeling the "Overconfidence Trap" (Term II in Eq. 15), we treat these low-entropy samples not as "safe", but as "potentially stagnant", ensuring they are eventually subjected to the more expensive but accurate multi-model voting process.

F.3.2. Why Learned Router Superior to Heuristic Entropy?

One might ask: why train a neural router instead of simply selecting samples with the highest current entropy? Our results show the learned router consistently outperforms heuristic baselines (Table 1, +5-6% accuracy gain on CIFAR-100). This

superiority stems from two key factors:

1. Resolving the "One-to-Many" Mapping Problem. In a multi-expert setting (as in Section 5.3), a high-entropy sample could potentially be improved by *any* of the available experts. A simple entropy heuristic tells us *that* the sample is uncertain, but not *who* can fix it. Our router takes the input sample and learns a mapping to the *expected entropy drop vector* for all candidates: $f(x) \to [\Delta H_1, \Delta H_2, \ldots, \Delta H_k]$. This allows for targeted routing—sending "flower-like" ambiguous images to the "flower expert"—which is impossible with scalar entropy sorting.

2. Learning "Correctability" vs. "Aleatoric Uncertainty". Not all high-entropy samples are fixable. Some images are inherently noisy or ambiguous (Aleatoric Uncertainty) and will remain uncertain even for a perfect model. A heuristic strategy would waste budget repeatedly updating these hopeless samples. In contrast, our router is trained on $\Delta H = H_{old} - H_{new}$. If H_{new} remains high (unfixable noise), then $\Delta H \approx 0$. Thus, the router implicitly learns to distinguish between "epistemic uncertainty" (fixable by better models) and "aleatoric uncertainty" (noise), focusing resources only on the former.

G. Supplementary Data

Table 11 presents the data and error analysis corresponding to Figure 7.

Table 11. 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