Uncertainty-guided dynamic routing for cost-efficient image classification

Edwin Rajoo, Steven Landgraf, and Markus Ulrich

Institute of Photogrammetry and Remote Sensing (IPF), Karlsruhe Institute of Technology (KIT), Englerstr. 7, 76131 Karlsruhe, Germany

Abstract Deep learning networks have reached high accuracy in image classification, but often at the expense of high computational cost, since large models apply their full complexity uniformly to every input regardless of difficulty. We propose an uncertainty-guided dynamic routing framework pairing a lightweight network (MobileNetV3-Large) with a high-capacity network (EfficientNet-B3), routing each input individually based on uncertainty signals, entropy and confidence, from the lightweight model, together with its intermediate feature representation. The router is trained with a binary objective rewarding forwarding only when it corrects a mistake the small model would otherwise make, combined with a class-imbalance-aware weighting term and a regularizer discouraging unnecessary escalation. On Tiny ImageNet, the routed system reaches 78.54% accuracy, matching and, at the chosen operating point, slightly exceeding the standalone larger model (78.02%), while forwarding only around 31% of inputs to it – an estimated 57% compute reduction. An ablation study shows the two scalar uncertainty signals alone route as well as, or better than, the full signal set, with the 960-dimensional feature embedding contributing little once class imbalance is accounted for.

Keywords Dynamic routing, uncertainty estimation, adaptive inference, image classification, efficient deep learning

1 Introduction

Deep convolutional neural networks achieve high accuracy on image classification, but this typically comes at a steep computational cost:

larger models can handle more complex visual patterns, yet applying the full model to every input wastes computation when the input is simple. Lightweight architectures such as MobileNet trade capacity for speed; the trade-off shows up mainly on harder inputs, where a small model struggles once the object is partly hidden, cluttered, or easily confused with another class. Difficulty is a property of the data itself, not only an artifact of training. Easy and hard examples differ systematically across CNN architectures [1]. This observation motivates adaptive inference, where the computation applied to an input depends on the input itself rather than being fixed in advance [2].

Prior work on adaptive inference falls broadly into three groups. Early-exit and adaptive-depth networks attach internal classifiers so that easy inputs can exit before the final layer [3, 4]. Dynamic routing and conditional computation methods instead route each input along a different path through the network, using reinforcement learning, gating, or learned routing functions [5–7]. A third line of work uses the network's own confidence or predictive uncertainty to decide where to spend computation [8, 9].

All three directions, however, share a common constraint: they operate within a single network, modifying its architecture, adding exit branches, or acting on a fixed confidence threshold. None of them choose between independently trained models of different capacity using a target aligned directly with when the larger model actually helps, rather than a general notion of difficulty.

We address this gap by pairing a lightweight model, evaluated on every image, with a high-capacity model reserved for a subset of inputs. A router decides whether the lightweight prediction can be trusted or the input should be escalated to the bigger model. Routing more images to the large model raises accuracy but also cost, and vice versa. The larger model is not a strict upgrade; however the two models do not fail on the same images so a router can only exploit this complementarity by learning which specific cases benefit from escalation, not just which ones look difficult. Concretely, we ask whether such a router can be trained using only signals internal to the small classifier itself (its penultimate-layer representation, confidence, and predictive entropy) without any access to the large model at inference time. The router's training target follows directly from the complementarity argument above: rather than learning a general notion of difficulty, it is

rewarded only for forwarding an image when the large model would actually correct a mistake the small model makes, and is discouraged from forwarding images the small model already classifies correctly.

2 Related Work

Early-exit and adaptive-depth networks. The most direct approach attaches internal classifiers to a network so easy inputs can exit before the final layer. BranchyNet [3] uses side-branch classifiers at several depths; Shallow-Deep Networks [10] expose *overthinking* (cases where a correct answer was already available at a shallower layer) cutting inference cost by over 50% without losing accuracy; MSDNet [4] frames this as a test-time-budget problem, training classifiers of different costs along a multi-scale, densely-connected backbone and choosing among them adaptively. All three remain bound to their backbone, and adding exits means architecture-specific work repeated for each new one. **Dynamic routing and conditional computation.** A second family routes inputs along different paths through the network. McGill and Perona [6] train routing functions over a graph of learned transformations; Cai et al. [7] use a hypernetwork to score branch importance via Gumbel-Softmax; SkipNet [5] learns to skip residual blocks per input (30–90% less computation); BlockDrop [11] takes a similar reinforcement-learned approach (20% ImageNet speedup); Mixture-of-Experts [12] route each input to a sparse subset of expert subnetworks. All need considerably more complex training than ordinary supervised learning, and the resulting router is often tied closely to its backbone. **Uncertainty and confidence-based inference.** A third line of work uses the network's own uncertainty as its signal, but stops short of learning what to do with it. Guo et al. [8] showed modern networks are often poorly calibrated and proposed temperature scaling as a fix; Gal and Ghahramani [9] reinterpret test-time dropout as approximate Bayesian inference; Lakshminarayanan et al. [13] find a simple ensemble matches or beats elaborate Bayesian alternatives; Xia et al. [14] separate total uncertainty into model and data components via a Dirichlet-based framework.

3 Methodology

We consider inputs $x \in \mathbb{R}^{H \times W \times C}$ assigned to a class $y \in \{1, \ldots, K\}$, and design a routing system between two networks of different capacities: a lightweight CNN $f_s(x)$ and a high-capacity CNN $f_b(x)$. A routing function $r(x) \in \{s, b\}$ sends easy inputs to the lightweight network and difficult ones to the high-capacity network, giving $\hat{y} = f_{r(x)}(x)$ and minimizing cost while preserving accuracy. A router network decides which model handles each input (Fig. 1); both CNNs are trained independently and kept frozen while only the router is trained, for ease of deployment and modularity.

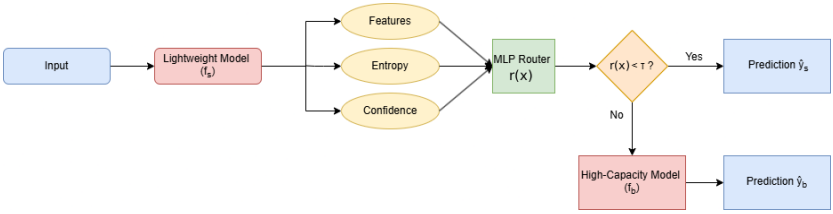


Figure 1: Overview of the proposed uncertainty-guided dynamic routing pipeline. Each input is first processed by the lightweight model f_s, which produces intermediate features, entropy, and confidence. These signals are passed to the MLP router $p(x)$, which decides whether the prediction from f_s is used directly or whether the input is forwarded to the high-capacity model f_b.

3.1 Routing Signals

The router decides not from the predicted label but from three signals describing the small model's behavior.

Intermediate features. $h(x) = f_s^{(feat)}(x)$, the 960-dimensional embedding from MobileNetV3-Large's penultimate layer after global average pooling, the same representation the model uses before its classification layer, at no extra cost. This embedding encodes how the small model internally represents the image before that representation is collapsed into a class decision, so it can carry information relevant to routing, such as the input resembling a region of feature space where the small model has historically been unreliable, even when the resulting confidence and entropy look unremarkable.

Prediction confidence. The max softmax probability of logits $z(x)$,

$$C(x) = \max_i \big(p_i(x)\big), \qquad p(x) = \text{softmax}(z(x)). \tag{1}$$

Confidence is the most direct proxy for whether the top prediction is trustworthy, but it only summarizes the probability mass assigned to a single class and says nothing about how that remaining mass is distributed across the rest. As a result, two inputs with identical confidence can differ substantially in how ambiguous the underlying decision actually is.

Predictive entropy. A measure of uncertainty across the whole output distribution,

$$H(x) = -\sum_{i=1}^{K} p_i(x) \log p_i(x). \tag{2}$$

Unlike confidence, entropy accounts for the shape of the entire distribution, so it can separate a case where one wrong class is nearly as likely as the top prediction from a case where the remaining probability mass is spread thinly across many classes. This makes it sensitive to a specific kind of failure mode, near-ties between competing classes, that confidence alone can miss.

Features provide semantic context, entropy captures distribution-wide uncertainty, and confidence reflects top-prediction strength – complementary signals for routing.

3.2 Router Network and Training

The router is a lightweight MLP: the three signals are concatenated into a 962-dimensional vector and passed through four fully connected layers (512, 256, 64, 1), with ReLU [15] after the first three and dropout [16] (0.2) after the first two to curb overfitting on the feature branch, which otherwise dominates the input by three orders of magnitude. A sigmoid function gives $r(x) \in [0, 1]$; x is routed to the large model whenever $r(x) \geq \tau$, selected empirically in Sec. 5.

Both CNNs are ImageNet-pretrained, fine-tuned independently with cross-entropy loss, then frozen. For input x with label y, let $\hat{y}_s = \arg\max f_s(x)$, $\hat{y}_b = \arg\max f_b(x)$; the router targets

$$g(x) = \mathbb{1}\big[(\hat{y}_b = y) \wedge (\hat{y}_s \neq y)\big], \tag{3}$$

i.e., $g(x) = 1$ only when the large model corrects a small-model mistake. This positive class is heavily underrepresented (about 0.5% of samples), so BCE is weighted by $w_{pos} \approx 199$ from the training-set balance:

$$\mathcal{L}_{BCE} = -\Big[w_{pos} \cdot g(x) \log r(x) + \big(1 - g(x)\big) \log \big(1 - r(x)\big)\Big]. \quad (4)$$

Two regularizers are added: a compute-aware term, and a term penalizing *small-better* cases $s(x) = \mathbb{1}[(\hat{y}_s = y) \wedge (\hat{y}_b \neq y)]$, where escalating would hurt accuracy:

$$\mathcal{L} = \mathcal{L}_{BCE} + \lambda_{compute} \cdot \mathbb{E}[r(x)] + \lambda_{wrong} \cdot \mathbb{E}\big[r(x) \cdot s(x)\big], \quad (5)$$

At inference, each input passes through the lightweight CNN; the resulting signals are passed to the router, whose thresholded output decides whether to keep the lightweight prediction or escalate. We report small- and large-model accuracy, routed system accuracy, routing ratio, and router accuracy (agreement with $g(x)$).

4 Experimental Setup

Dataset. Tiny ImageNet [17]: 200 classes at native 64×64 resolution, a manageable alternative to full ImageNet [18]. Both backbones are pretrained at 224×224, so images are upsampled; training uses random cropping (4-pixel padding) and horizontal flipping, with standard ImageNet normalization.

Models. The lightweight classifier is a pretrained MobileNetV3-Large [19]; the high-capacity classifier is a pretrained EfficientNet-B3 [20]. Both use ImageNet-pretrained weights with the final layer replaced for 200 classes; the router is the MLP of Sec. 3.2.

Training. All three networks are trained separately with Adam (learning rate 10^{-4}, batch size 16), the two CNNs for 30 epochs each and the router for 60 epochs; the CNNs use cross-entropy loss and the router uses the loss of Sec. 3.2, with $\lambda_{compute} = 0.01$, $\lambda_{wrong} = 1.5$, and $w_{pos} \approx 199$ from the training-set class balance.

Evaluation metrics. We report small- and large-model accuracy, routed accuracy, routing ratio, and router accuracy (agreement with $g(x)$).

Threshold sweep, ablations, and compute cost. After router training, τ is swept from 0.10 to 0.50 in steps of 0.01, recording routed accuracy, routing ratio, and router accuracy at each value; a separate ablation study retrains the router on all seven non-empty subsets of {features, entropy, confidence}, with the CNNs held fixed. Since every input passes through the lightweight CNN before routing, total cost is approximated as $\text{Cost}_{routed} = \text{Cost}_{small} + p_{big} \cdot \text{Cost}_{big}$, where p_{big} is the share forwarded to the large model.

5 Results and Discussion

5.1 Baseline, Threshold Selection, and Case Analysis

Table 1 summarizes the accuracy of the standalone models and the full routed system at the chosen operating point ($\tau = 0.48$, see below). The routed system edges out the standalone EfficientNet-B3 model despite forwarding fewer than a third of all validation images to it. This is not simply an accuracy/efficiency trade-off: the two backbones make partially complementary errors (Table 2), and keeping the lightweight model's correct predictions instead of overriding them produces the accuracy gain alongside the compute reduction. With profiled FLOPs at 224×224 (MobileNetV3-Large: 0.22 GFLOPs; EfficientNet-B3: 1.83 GFLOPs), the routed system averages about 0.78 GFLOPs/image against 1.83 GFLOPs for uniform large-model inference – an estimated 57% reduction.

Table 1: Baseline classification performance.

Model	Accuracy
Lightweight CNN (MobileNetV3-Large)	73.89%
High-capacity CNN (EfficientNet-B3)	78.02%
Routed System ($\tau = 0.48$)	**78.54%**

Routed system accuracy is essentially flat across $\tau \in [0.10, 0.50]$ (Fig. 2): 78.72% at $\tau = 0.10$ down to 78.52% at $\tau = 0.50$, degrading gracefully rather than sharply as the workload shifts to the lightweight model. Scoring each threshold as $\text{score}(\tau) = acc_{routed}(\tau) - 0.001 \times (100 - pct_{small}(\tau))$, a light penalty for reliance on the large model, se-

lects $\tau = 0.48$: 78.54% routed system accuracy while sending only 30.83% of images to the high-capacity model, against 35.62% at the highest-accuracy threshold.

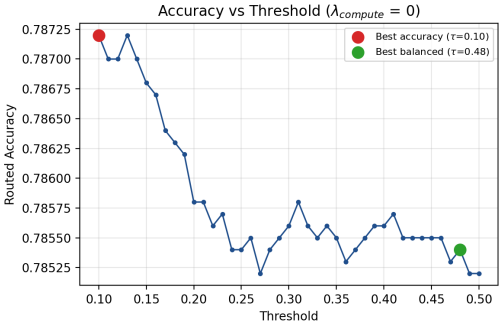


Figure 2: Routed system accuracy as a function of routing threshold. Red marker: Threshold maximizing raw accuracy. Green marker: Chosen operating point.

Breaking decisions down by outcome at $\tau = 0.10$ (Table 2): of 1105 cases where the large model would fix a small-model mistake, only 99 are missed, but of 692 cases where the small model is actually better, 502 (over 70%) are forwarded anyway – the router catches corrections far better than it avoids unnecessary escalation. This asymmetry is baked into training: $g(x)$ rewards catching every correction, while λ_{wrong} only discourages, rather than rules out, the opposite mistake. An oracle that escalates only when it fixes a mistake reaches 84.94% while sending just 11.1% of images to the large model – over six points above the learned router, showing it exploits only part of the backbones' complementary structure, leaving substantial potential for future work.

5.2 Ablation Study

Table 3 shows results for all seven signal combinations at $\tau = 0.48$. The full router is surprisingly not the best configuration: entropy-only (79.00%) and confidence-only (78.97%) both beat it (78.56%), and dropping features entirely still reaches 78.96%. Entropy is the most

Table 2: Routing case analysis on the validation set ($\tau = 0.10$).

Outcome	Count
Both models correct	6697
Large model fixes small model's error	1105
of which, missed (not routed to large model)	99
Small model correct, large model wrong	692
of which, wrongly routed to the large model	502
Both models wrong (irreducible)	1506

load-bearing signal (removing it hurts accuracy most, 76.70%); confidence tracks it closely, suggesting the most useful information already sits in the top softmax probability. Features alone reach only 74.69% while pushing 86.3% of inputs to the small model – default avoidance rather than confident routing – likely because the 960-dimensional feature vector outweighs the two-dimensional signal pair by three orders of magnitude, inviting overfitting given how rare the positive class is. Router accuracy is not a trustworthy proxy for quality either: only-features has the highest router accuracy (79.56%) but the worst routed system accuracy (74.69%).

Table 3: Ablation study on router input components at $\tau = 0.48$.

Feat.	Ent.	Conf.	Sys. Acc. (%)	Small Usage (%)	Router Acc. (%)
✓	✓	✓	78.56	69.10	74.27
–	✓	✓	78.96	59.00	67.60
✓	–	✓	76.70	80.60	78.79
✓	✓	–	78.65	67.30	72.92
✓	–	–	74.69	86.30	79.56
–	✓	–	79.00	61.50	69.48
–	–	✓	78.97	62.90	70.51

5.3 Discussion

The ablation results contextualize the earlier assumption that more signal is always better. Entropy and confidence alone are not only sufficient, they are slightly better than the full combination. Features add little by themselves and seem to introduce noise once stacked on top of the other two. With only around 0.5% of training samples are positive,

this is not a huge surprise: a 960-dimensional feature vector gives the router far more surface area to overfit spurious correlations than a two-dimensional uncertainty signal does. None of this rules out feature-based signals in general; it just means their value depends on the specific backbones, dataset size, and class balance in play, not something to add by default.

The case analysis and oracle comparison point at where the remaining gap actually is. An oracle with access to ground truth reaches 84.94% while sending just 11.1% of inputs to the large model, well past what the learned router manages, and far more selective about it. Most of that gap traces back to one specific failure mode: the router is much more careful about not missing a large-model correction than about avoiding unnecessary escalation where the small model was already right. A stronger wrong-routing penalty, or a three-way target instead of a binary one, would target this asymmetry directly.

Despite impressive results, two limitations remain. Every input pays for a full lightweight-model forward pass before any routing decision is made, so this framework cannot claim savings within the lightweight model itself – a problem that could be addressed with hierarchical or early-exit extensions. Additionally, our findings are based on limited evaluations on Tiny ImageNet only. Extending our work to other datasets with different class balances or difficulty profiles, or evaluating different model architectures, presents an obvious direction for future work.

6 Conclusion

A router built from nothing but entropy and confidence, no feature access required, already matches a substantially larger classifier's accuracy while touching it for less than a third of the time, at an estimated 57% reduction in compute. Interestingly, the oracle and case-level results show that this is not the ceiling yet: a more targeted routing objective – for instance a three-way target that separately penalizes missed corrections and unnecessary escalations, rather than a single binary target that only partly discourages the latter – could improve the overall performance and efficiency even further. Extending the framework with an early-exit mechanism within the lightweight model itself, and

testing it on datasets with a different difficulty and class-balance profile than Tiny ImageNet, are natural next steps.

References

1. I. Kishida and H. Nakayama, "Empirical study of easy and hard examples in CNN training," in *Neural Information Processing*, ser. Communications in Computer and Information Science. Springer, 2019, vol. 1142, pp. 179–188.
2. Y. Han, G. Huang, S. Song, L. Yang, H. Wang, and Y. Wang, "Dynamic neural networks: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 11, pp. 7436–7456, 2022.
3. S. Teerapittayanon, B. McDanel, and H. T. Kung, "BranchyNet: Fast inference via early exiting from deep neural networks," in *Proceedings of the 23rd International Conference on Pattern Recognition (ICPR)*, 2016, pp. 2464–2469.
4. G. Huang, D. Chen, T. Li, F. Wu, L. van der Maaten, and K. Q. Weinberger, "Multi-scale dense networks for resource efficient image classification," in *International Conference on Learning Representations (ICLR)*, 2018.
5. X. Wang, F. Yu, Z.-Y. Dou, T. Darrell, and J. E. Gonzalez, "SkipNet: Learning dynamic routing in convolutional networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 409–424.
6. M. McGill and P. Perona, "Deciding how to decide: Dynamic routing in artificial neural networks," in *Proceedings of the 34th International Conference on Machine Learning (ICML)*, ser. Proceedings of Machine Learning Research, vol. 70, 2017, pp. 2363–2372.
7. S. Cai, Y. Shu, and W. Wang, "Dynamic routing networks," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2021, pp. 3588–3597.
8. C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *Proceedings of the 34th International Conference on Machine Learning (ICML)*, ser. Proceedings of Machine Learning Research, vol. 70, 2017, pp. 1321–1330.
9. Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," in *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, ser. Proceedings of Machine Learning Research, vol. 48, 2016, pp. 1050–1059.
10. Y. Kaya, S. Hong, and T. Dumitraş, "Shallow-deep networks: Understanding and mitigating network overthinking," in *Proceedings of the 36th Interna-*

tional Conference on Machine Learning (ICML), ser. Proceedings of Machine Learning Research, vol. 97, 2019, pp. 3301–3310.

11. Z. Wu, T. Nagarajan, A. Kumar, S. Rennie, L. S. Davis, K. Grauman, and R. Feris, "BlockDrop: Dynamic inference paths in residual networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 8817–8826.
12. N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean, "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," in *International Conference on Learning Representations (ICLR)*, 2017.
13. B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," *Advances in Neural Information Processing Systems*, vol. 30, 2017.
14. F. Xia, J. Snell, and T. L. Griffiths, "Early exiting in deep neural networks via Dirichlet-based uncertainty quantification," in *NeurIPS 2024 Workshop on Fine-Tuning in Machine Learning*, 2024.
15. V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML)*, 2010, pp. 807–814.
16. N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.
17. Y. Le and X. Yang, "Tiny ImageNet visual recognition challenge," CS231N Course Project, Stanford University, Tech. Rep., 2015.
18. J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009, pp. 248–255.
19. A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, "Searching for MobileNetV3," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 1314–1324.
20. M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the 36th International Conference on Machine Learning (ICML)*, ser. Proceedings of Machine Learning Research, vol. 97, 2019, pp. 6105–6114.