Failure Prediction Is a Better Performance Proxy for Early-Exit Networks Than Calibration

Piotr Kubaty[*] **Filip Szatkowski**[†] **Metod Jazbec**[‡] **Bartosz Wójcik**[§]

Abstract

Early-exit models accelerate inference by attaching internal classifiers to intermediate layers of the network, allowing computation to halt once a prediction meets a predefined exit criterion. Most early-exit methods rely on confidence-based exit strategies, which has motivated prior work to calibrate intermediate classifiers in pursuit of improved performance-efficiency trade-offs. In this paper, we argue that calibration metrics can be misleading indicators of multi-exit model performance. Specifically, we present empirical evidence showing that miscalibrated networks can outperform calibrated ones. As an alternative, we propose using failure prediction as a more informative proxy for early-exit model performance. Unlike calibration, failure prediction captures changes in sample rankings and correlates strongly with efficiency gains, offering a more reliable framework for designing and evaluating early-exit models.

1 Introduction

The rapid growth of deep learning increases the demand for resource-efficient models. Early-exit models address this challenge by attaching classifiers to intermediate model layers and enabling the model to save computation by stopping the inference once a prediction satisfies an exit criterion. Early-exits were initially introduced for vision models [7, 11, 23, 29], and have since then become a natural fit for resource-constrained scenarios [2, 3, 8, 14, 15, 27, 28, 31]. More recently, they also have been successfully adopted for natural language processing [9, 16, 26, 32, 38], including reasoning models [10, 34] where inference efficiency is critical.

The most common approach to exiting uses prediction confidence and enables the network to halt early if an intermediate classifier produces a sufficiently confident prediction. Consequently, many works focused on improving the calibration of the classifiers [17, 19, 20, 22, 24], under the assumption that better calibration yields better models. We challenge this assumption and argue that calibration can be a misleading proxy for early-exit models. We demonstrate cases where miscalibrated networks behave counterintuitively and outperform calibrated multi-exit models, as demonstrated in Figure 1. Finally, we highlight that calibration methods can introduce unintended side effects, such as altering the maximum confidence ranking of samples within the classifier.

Given the above-mentioned issues, we propose to use *failure prediction* [1] as a more suitable proxy for early-exit network performance. We discuss its desirable properties, showing that, unlike calibration, failure prediction measures are sensitive to changes in rankings of samples. Finally, we adapt failure prediction measures to the multi-exit setting by defining the *Early-Exit Failure Prediction score* (EEFP score). Notably, our experiments demonstrate that our proposed EEFP score shows a strong correlation with early-exit performance in cases where calibration measures do not.

[*]Jagiellonian University
[†]Warsaw University of Technology, IDEAS NCBR
[‡]University of Amsterdam
[§]Jagiellonian University, Tooploox

Accepted at the third SPIGM workshop @ NeurIPS 2025.

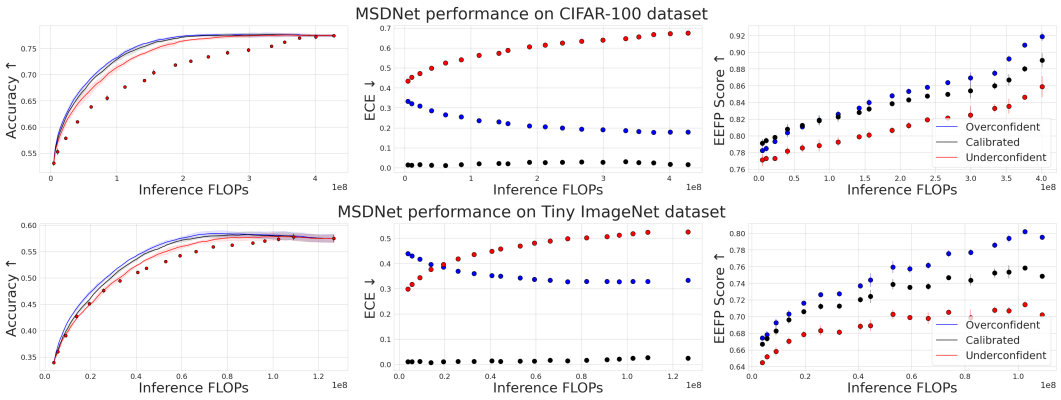

Figure 1: Cost-accuracy curves, head calibration errors, and our proposed EEFP scores for one calibrated model and two decalibrated models with modified temperature values. Calibration fails to capture the quality of the early-exit model, as an overconfident network with higher ECEs performs better than the calibrated one. We propose an alternative metric, *Early-Exit Failure Prediction score* (EEFP score), which more accurately reflects the quality of the multi-exit model.

We hope that our work advances the discussion on appropriate evaluation metrics for early-exit models and helps improve the design of future methods for adaptive computation.

2 Background

Early-exits We consider the standard early-exit framework used in prior work [7, 11], and start with a multi-exit model architecture with J classifiers. Each classifier g_j, $j \in \{1, \ldots, J\}$ maps an input x to a probability vector over C classes: $p_j = g_j(x)$. The network evaluates classifiers sequentially until the *confidence* of the prediction (usually the maximum probability obtained via softmax) exceeds the corresponding exit threshold, that is: $c_j(x) = \max_k p_{j,k} \geq \tau_j$, where $k = 1, \ldots, C$ refers to class indices [9]. If the exit condition is satisfied, the network outputs p_j; otherwise, evaluation continues until another classifier exits or the final classifier g_J is reached. By varying the exit thresholds, we obtain a *compute–accuracy trade-off curve*, where compute is measured as the average of floating-point operations per sample [33, 35].

Calibration Confidence calibration refers to how well a model's predicted confidence matches its actual accuracy [4, 5]. In a well-calibrated model, predictions made with high confidence are more likely to be correct. Calibration is a key property of probabilistic models and has been widely studied in the context of model reliability and trustworthiness [5, 12, 25, 30]. The calibration of a given classifier can be measured via expected calibration error (ECE) [21]. To calculate ECE, one partitions predictions based on their confidence into M bins $B_1, \ldots, B_M$ and calculates the accuracy $\mathrm{acc}(B_m)$ and average confidence $\mathrm{conf}(B_m)$ in each bin m. Then, ECE can be calculated as:

$$\mathrm{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} \left|\mathrm{acc}(B_m) - \mathrm{conf}(B_m)\right|.$$

3 Calibration in early-exit models

In context of early-exiting, prior work [19, 22] linked the calibration of the intermediate classifiers with improved performance of the model. Since calibration changes head's confidence, it can alter sample distribution across exit points and thus impact the overall model performance. Meronen et al. [19] explicitly focus on improving early-exit performance via calibration, stating that "adequately quantifying and accounting for (...) uncertainty improves the predictive performance and aids decision-making". Similarly to Pacheco et al. [22] and Wójcik et al. [32], they measure the calibration error of each classification head. Despite the existing body of work mentioned above, **we hypothesize**

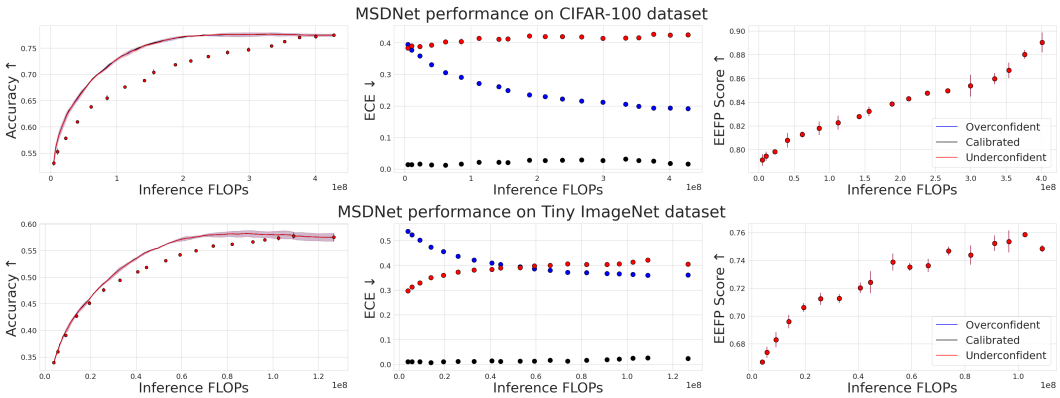

Figure 2: Performance of models decalibrated using a transformation that preserves the ranking of samples within each classifier. Despite significantly deteriorated ECE scores, model performance remains unchanged, with all three models showing almost identical cost–accuracy curves. EESP score perfectly reflects this behavior, assigning identical score values to all three networks.

that calibration does not correlate with the performance of the early-exit networks, and may even hurt it in practice. In the following sections, we provide empirical evidence supporting this hypothesis by showing that calibration of early-exit networks can affect performance in unexpected ways. To this end, we adopt the setting from Meronen et al. [19] with the MSDNet multi-exit architecture [7], and find the per-IC thresholds for every considered budget after the training and (de)calibration via the procedure proposed by Huang et al. [7]. We use cost-accuracy curves to compare the network performance, and mark the performance of individual classifiers with dots. Please refer to Appendix E for the experimental details, and to Appendix C for the additional experimental results with alternative calibration methods.

3.1 Better calibration does not always lead to better performance

In our first experiment, we calibrate the intermediate classifiers of a trained MSDNet model with temperature scaling [5]. Subsequently, we create two decalibrated variants of the same model by multiplying the temperature used in each head by 3.0 or 0.3, which results in an underconfident or overconfident model, respectively. We evaluate both accuracy and expected calibration error (ECE) of the three models across varying computational budgets, and present the results in Figure 1. Our experiments reveal that **models can achieve a more favorable cost–accuracy trade-off, despite exhibiting substantially worse calibration scores.** This somewhat counterintuitive finding indicates that **calibrating intermediate classifiers might not be beneficial for multi-exit models, which is contrary to what was suggested in previous studies** [19, 20, 22, 32].

3.2 Worse calibration does not always lead to worse performance

To further analyze the relationship between head calibration and the final performance of the model, we devise the following experiment. Instead of manipulating the temperature of each head, we propose an even simpler decalibration method, which directly transforms the final confidence estimate of the classifier: $\hat{c}_j(x) = \frac{1}{C} + \left(1 - \frac{1}{C}\right) \cdot \left(\frac{c_j(x) - \frac{1}{C}}{1 - \frac{1}{C}}\right)^{\alpha}$, where α is the *decalibration coefficient* and C refers to the number of classes. Formally, for any classifier j let π_j be a permutation of sample indices such that: $c_j(x_{\pi(1)}) > c_j(x_{\pi(2)}) > \cdots > c_j(x_{\pi(n)})$ (see Appendix B for a more detailed discussion on the design of our function). The $\hat{c}_j$ **transformation preserves the order of confidences between the samples (*sample ranking*). The same is not true for temperature scaling** [5], a fact that we describe in detail in Appendix A.

We perform the alternative decalibration experiment by using $\hat{c}_j$ to decalibrate the classifiers. We set α to 10 and 0.1, obtaining an underconfident and overconfident model, respectively. The results

[5]Consider two sets of logits: $l^1 = [0.65, 0.34, -1.03]$, $l^2 = [-0.06, -0.11, 0.60]$. If we compute confidence c with the softmax function, for $T_1 = 1.0$: $c^1 > c^2$, a while for $T_2 = 0.3$: $c^1 < c^2$.

are shown in Figure 2. **This time, despite decalibrating the model significantly, the overall accuracy-cost performance remains the same, even for the underconfident model.** This further illustrates that calibration metrics do not necessarily reflect actual model performance, and do not capture the important aspect of sample rankings.

4 Early-exit failure prediction

The ranking-preserving aspect of the calibration methods proved to be essential in our previous experiments. We observe that the **separability between correctly and incorrectly classified samples** may be a more reliable indicator of early-exit model performance than calibration measures. In the literature, this separability is directly related to *failure prediction* (or *error and success prediction*) [1, 6]. Crucially, prior work has shown that calibrating classifiers can actually degrade failure prediction performance [39].

To evaluate failure prediction for a single classifier, we record the classifier's confidence score for each sample and label it as $y = 1$ if the prediction is correct and $y = 0$ otherwise. We then compute the area under the receiver operating characteristic curve (AUROC), which measures the probability that a randomly chosen correct prediction ($y = 1$) receives a higher confidence score than a randomly chosen incorrect one ($y = 0$), thus reflecting how well the classifier's confidence separates correct from incorrect predictions.

However, this definition of failure prediction was devised for conventional static classifiers, and is not suitable for early-exit networks, as it does not account for the behavior of deeper classifiers. In particular, if a sample is incorrectly classified by the current classifier and all of the deeper classifiers, then it is beneficial to halt computation as early as possible. This crucial observation leads us to adapt the definition of failure prediction to the multi-exit model setup. **We define Early Exit Failure Prediction score (EEFP score)** as:

$$\text{EEFP}_j(\{x_i, y_{j,i}\}) = \text{AUROC}(\{c_j(x_i), \bar{y}_{j,i}\}),$$

where:

$$\bar{y}_{j,i} = \begin{cases} 1, & \text{if } y_{j,i} = 1 \vee (y_{j,i} = 0 \wedge \forall_{l>j} y_{l,i} = 0) \\ 0, & \text{otherwise.} \end{cases}$$

In this formulation, a positive $y_{j,i}$ means that either the current head is correct, or all deeper heads would also be wrong, and exiting is optimal from the computational point of view.

We report the EEFP scores for each classifier head alongside the results of our previous experiments from Sections 3.1 and 3.2 in Figures 1 and 2. In both cases, the EEFP scores correlate well with the overall accuracy-cost performance of the model. When performing temperature decalibration, the EESP score is the highest for the overconfident model, which actually performs better but achieves worse ECE. In the case of rank-preserving decalibration, EESP scores remain constant and reflect the identical performance of all three investigated models. **EEFP score better reflects both prediction accuracy and the effect of early exits on the model performance, making it more suitable for evaluating early-exit models than standard calibration metrics**.

5 Conclusion

Our work challenges the common assumption about positive effects of calibration of intermediate classifiers in the early-exit models. Through a series of controlled experiments, we demonstrated that calibration metrics such as ECE can be misleading: well-calibrated models may still waste computation, while deliberately miscalibrated models can sometimes achieve better cost–accuracy trade-offs. A key factor behind this discrepancy is that calibration metrics fail to capture the separability of correctly and incorrectly classified samples, which directly influences the efficiency of the early exits.

To address these limitations, we propose the Early-Exit Failure Prediction Score (EEFP Score), a failure prediction metric specifically tailored to the multi-exit setting. Unlike calibration metrics, EEFP Score directly measures how well a model distinguishes between samples that benefit from further computation and those that do not. EEFP Score better reflects the real goals of efficient inference, and we found that it strongly correlates with early-exit performance across the experimental settings where calibration metrics fail to accurately capture model quality.

References

[1] Charles Corbière, Nicolas Thome, Avner Bar-Hen, Matthieu Cord, and Patrick Pérez. Addressing failure prediction by learning model confidence. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[2] Biyi Fang, Xiao Zeng, Faen Zhang, Hui Xu, and Mi Zhang. Flexdnn: Input-adaptive on-device deep learning for efficient mobile vision. In *2020 IEEE/ACM Symposium on Edge Computing (SEC)*, 2020.

[3] Amir Ghodrati, Babak Ehteshami Bejnordi, and Amirhossein Habibian. Frameexit: Conditional early exiting for efficient video recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[4] Tilmann Gneiting, Fadoua Balabdaoui, and Adrian E Raftery. Probabilistic forecasts, calibration and sharpness. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 2007.

[5] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On Calibration of Modern Neural Networks. *CoRR*, 2017.

[6] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv*, 2016.

[7] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Weinberger. Multi-Scale Dense Networks for Resource Efficient Image Classification. In *International Conference on Learning Representations (ICLR)*, 2018.

[8] Metod Jazbec, James Allingham, Dan Zhang, and Eric Nalisnick. Towards anytime classification in early-exit architectures by enforcing conditional monotonicity. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[9] Metod Jazbec, Alexander Timans, Tin Hadži Veljković, Kaspar Sakmann, Dan Zhang, Christian Andersson Naesseth, and Eric Nalisnick. Fast yet safe: Early-exiting with risk control. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[10] Guochao Jiang, Guofeng Quan, Zepeng Ding, Ziqin Luo, Dixuan Wang, and Zheng Hu. FlashThink: An Early Exit Method For Efficient Reasoning. *arXiv*, 2025.

[11] Yigitcan Kaya, Sanghyun Hong, and Tudor Dumitras. Shallow-deep networks: Understanding and mitigating network overthinking. In *International Conference on Machine Learning (ICML)*, 2019.

[12] Farina Kock, Felix Thielke, Grzegorz Chlebus, and Hans Meine. Confidence Histograms for Model Reliability Analysis and Temperature Calibration. In *Proceedings of The 5th International Conference on Medical Imaging with Deep Learning*, Proceedings of Machine Learning Research, 2022.

[13] Agustinus Kristiadi, Matthias Hein, and Philipp Hennig. Being Bayesian, Even Just a Bit, Fixes Overconfidence in ReLU Networks. In *International Conference on Machine Learning (ICML)*, Proceedings of Machine Learning Research, 2020.

[14] Stefanos Laskaridis, Stylianos I Venieris, Mario Almeida, Ilias Leontiadis, and Nicholas D Lane. SPINN: Synergistic progressive inference of neural networks over device and cloud. In *Proceedings of the 26th annual international conference on mobile computing and networking*, 2020.

[15] Yanwei Li, Lin Song, Yukang Chen, Zeming Li, Xiangyu Zhang, Xingang Wang, and Jian Sun. Learning dynamic routing for semantic segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[16] Kaiyuan Liao, Yi Zhang, Xuancheng Ren, Qi Su, Xu Sun, and Bin He. A global past-future early exit method for accelerating inference of pre-trained language models. In *Proceedings of the 2021 conference of the north american chapter of the association for computational linguistics: Human language technologies*, 2021.

[17] L Lilli, E Giarnieri, and S Scardapane. A Calibrated Multiexit Neural Network for Detecting Urothelial Cancer Cells. *Computational and Mathematical Methods in Medicine*, 2021.

[18] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

[19] Lassi Meronen, Martin Trapp, Andrea Pilzer, Le Yang, and Arno Solin. Fixing Overconfidence in Dynamic Neural Networks. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2024.

[20] Mehrnaz Mofakhami, Reza Bayat, Ioannis Mitliagkas, Joao Monteiro, and Valentina Zantedeschi. Performance Control in Early Exiting to Deploy Large Models at the Same Cost of Smaller Ones. *arXiv*, 2024.

[21] Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the AAAI conference on artificial intelligence*, 2015.

[22] Roberto G Pacheco, Rodrigo S Couto, and Osvaldo Simeone. On the impact of deep neural network calibration on adaptive edge offloading for image classification. *Journal of Network and Computer Applications*, 2023.

[23] Priyadarshini Panda, Abhronil Sengupta, and Kaushik Roy. Conditional deep learning for energy-efficient and enhanced pattern recognition. In *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2016.

[24] Lorena Qendro, Alexander Campbell, Pietro Lio, and Cecilia Mascolo. Early exit ensembles for uncertainty quantification. In *Machine Learning for Health*, 2021.

[25] Rebecca Roelofs, Nicholas Cain, Jonathon Shlens, and Michael C Mozer. Mitigating bias in calibration error estimation. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2022.

[26] Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[27] Jiawei Shao, Haowei Zhang, Yuyi Mao, and Jun Zhang. Branchy-GNN: A device-edge co-inference framework for efficient point cloud processing. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.

[28] Thierry Tambe, Coleman Hooper, Lillian Pentecost, Tianyu Jia, En-Yu Yang, Marco Donato, Victor Sanh, Paul Whatmough, Alexander M Rush, David Brooks, et al. Edgebert: Sentence-level Energy optimizations for latency-aware multi-task nlp inference. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, 2021.

[29] Surat Teerapittayanon, Bradley McDanel, and H.T. Kung. BranchyNet: Fast inference via early exiting from deep neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, 2016. doi: 10.1109/ICPR.2016.7900006.

[30] Cheng Wang. Calibration in Deep Learning: A Survey of the State-of-the-Art. *arXiv*, 2024.

[31] Zizhao Wang, Wei Bao, Dong Yuan, Liming Ge, Nguyen H Tran, and Albert Y Zomaya. SEE: Scheduling early exit for mobile DNN inference during service outage. In *Proceedings of the 22nd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, 2019.

[32] Bartosz Wójcik, Marcin Przewieźlikowski, Filip Szatkowski, Maciej Wołczyk, Klaudia Bałazy, Bartłomiej Krzepkowski, Igor Podolak, Jacek Tabor, Marek Śmieja, and Tomasz Trzciński. Zero time waste in pre-trained early exit neural networks. *Neural Networks*, 2023.

[33] Maciej Wołczyk, Bartosz Wójcik, Klaudia Bałazy, Igor T Podolak, Jacek Tabor, Marek Śmieja, and Tomasz Trzcinski. Zero time waste: Recycling predictions in early exit neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[34] Chenxu Yang, Qingyi Si, Yongjie Duan, Zheliang Zhu, Chenyu Zhu, Qiaowei Li, Zheng Lin, Li Cao, and Weiping Wang. Dynamic Early Exit in Reasoning Models. *arXiv*, 2025.

[35] Haichao Yu, Haoxiang Li, Gang Hua, Gao Huang, and Humphrey Shi. Boosted dynamic neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, 2023.

[36] Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, Seong Joon Oh, Youngjoon Yoo, and Junsuk Choe. CutMix: Regularization Strategy to Train Strong Classifiers With Localizable Features. In *IEEE/CVF International Conference on Computer Vision, ICCV*, 2019.

[37] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond Empirical Risk Minimization. In *International Conference on Learning Representations (ICLR)*, 2018.

[38] Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. Bert loses patience: Fast and robust inference with early exit. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[39] Fei Zhu, Zhen Cheng, Xu-Yao Zhang, and Cheng-Lin Liu. Rethinking Confidence Calibration for Failure Prediction. *arXiv*, 2023.

Appendix

A Influence of temperature scaling

A.1 Probability distribution after temperature scaling

In a classification problem with C classes, suppose the model outputs logits $z_1, \ldots, z_C$ for a given data sample, and let r denote the index of the most probable class. Without temperature scaling, the softmax probabilities are

$$p_k = \frac{e^{z_k}}{\sum_{l=1}^{C} e^{z_l}}, \quad k = 1, \ldots, C.$$

By introducing

$$d := \log\left(\sum_{l=1}^{C} e^{z_l}\right),$$

we can equivalently write

$$p_k = e^{z_k - d}, \quad \text{and hence} \quad z_k = \log(p_k) + d.$$

When scaling logits by a temperature parameter $T > 0$, the probabilities become

$$p_k^{(T)} = \frac{e^{z_k/T}}{\sum_{l=1}^{C} e^{z_l/T}} = \frac{p_k^{1/T}}{\sum_{l=1}^{C} p_l^{1/T}}, \quad k = 1, \ldots, C.$$

Therefore, the confidence changes from p_r to

$$p_r^{(T)} = \frac{p_r^{1/T}}{\sum_{l=1}^{C} p_l^{1/T}}.$$

Since the denominator $\sum_{l=1}^{C} p_l^{1/T}$ depends on the entire probability distribution (and not solely on p_r), two samples with the same original confidence p_r can yield different scaled confidences $p_r^{(T)}$ after temperature scaling.

A.2 Temperature scaling does not preserve the ranking of samples

Table 1: Classifier's logits in a toy problem.

	Class 1	Class 2	Class 3	Class 4
Image A	-0.7985	-0.9163	-2.3026	-2.9957
Image B	-1.6094	-1.6094	-0.9163	-1.6094
Image C	-1.2040	-0.9676	-1.3471	-2.8134

Table 2: Classes probability distribution (after softmax) in a toy problem.

	Class 1	Class 2	Class 3	Class 4
Image A	0.450	0.400	0.100	0.050
Image B	0.200	0.200	0.400	0.200
Image C	0.300	0.380	0.260	0.060

Consider a toy example with four classes. The $j-th$ classifier's logit outputs are shown in Table 1, while the corresponding softmax probabilities are reported in Table 2. Confidences of the predictions are as follows: $c_j(A) \approx 0.450$, $c_j(B) \approx 0.400$, $c_j(C) \approx 0.380$. Therefore the ranking of samples is A, B, C (from the most to the least confident ones). However, when temperature changes, the ranking does as well. For example for temperature 0.3, $c_j(A) \approx 0.594$, $c_j(B) \approx 0.771$, $c_j(C) \approx 0.575$, and the ranking changes to B, A, C. On the other hand, for temperature 3.0, $c_j(A) \approx 0.328$, $c_j(B) \approx 0.296$, $c_j(C) \approx 0.299$, and the ranking changes to A, C, B.

B Monotonic decalibration function

We define rank-preserving decalibration transformation from Section 3.2 as:

$$f_\alpha(c) \;=\; \frac{1}{C} \;+\; \Big(1-\frac{1}{C}\Big)\Big(\frac{c-\frac{1}{C}}{1-\frac{1}{C}}\Big)^{\alpha},$$

and use it to obtain $\hat{c}_j(x) \;=\; f_\alpha(c_j(x))$. In f_α, C is constant, and $c \;\mapsto\; \frac{c-\frac{1}{C}}{1-\frac{1}{C}}$ increases with increasing c. Raising a positive increasing function to the power $\alpha > 0$ also preserves monotonicity. Therefore, $f_\alpha(c)$ is strictly increasing with increasing c.

However, due to numerical reasons, f_α may increase so slowly that finite precision arithmetic makes it appear non-monotonic. To avoid this, we introduce a slightly modified function:

$$\hat{f}_\alpha(c) \;=\; \epsilon c \;+\; (1-\epsilon)\, f_\alpha(c),$$

where $\epsilon = 5 \cdot 10^{-2}$. This guarantees that, on any sufficiently long interval $[c_1, c_2]$, the increase of $\hat{f}_\alpha$ is large enough to remain distinguishable under finite numerical precision.

Moreover, $\hat{f}_\alpha(\frac{1}{C}) = \frac{1}{C}$ and $\hat{f}_\alpha(1) = 1$, which corresponds to the cases where all classes are equally probable or where one class has probability one, respectively. Therefore, we ensure that the confidence values remain within the range that would also be possible under the standard definition.

C Temperature decalibration for models calibrated with alternative methods

To ensure our findings are not specific to a single training or calibration setup, we analyze the effect of temperature decalibration on model accuracy in the setting proposed by [19]. They consider two approaches that improve model calibration: last-layer Laplace approximations (LAP) [13] and model-internal ensembles (MIE). We test these approaches using three MSDNet variants with 4, 6, and 8 blocks (referred to as "Small," "Medium," and "Large" in our plots) trained on CIFAR100, and analyze the impact of temperature decalibration on the models in the following subsections.

C.1 LAP

We begin with a model calibrated using LAP, where we perform a grid search over the temperature and Laplace prior variance to obtain the best-calibrated model. We then decalibrate the model by applying temperature modifications to its predictions, as in the main paper. We show the corresponding results in Figure 3. Likewise, the performance of the declibrated model does not degrade and is even slightly better than the performance of the calibrated one.

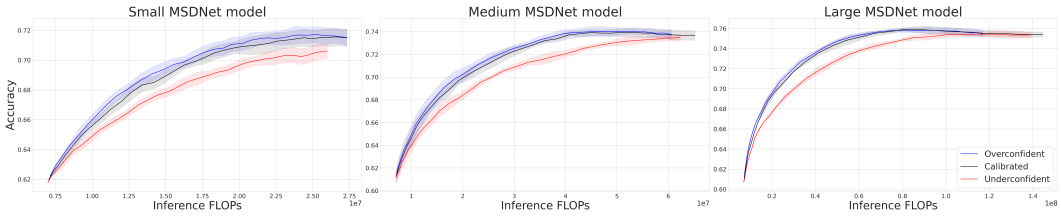

Figure 3: Results for LAP-calibrated MSDNet models.

C.2 LAP+MIE

Secondly, we combine LAP with MIE following Meronen et al. [19], and apply post-hoc temperature decalibration as before. We show the results in Figure 4. The results are consistent with our main findings and all other experiments: despite being decalibrated, the overconfident model does not suffer and even slightly outperforms the calibrated model. Together with the results from the previous section, these findings demonstrate that our conclusions hold beyond a specific training or calibration setup and are consistent across both calibration methods and multiple model sizes.

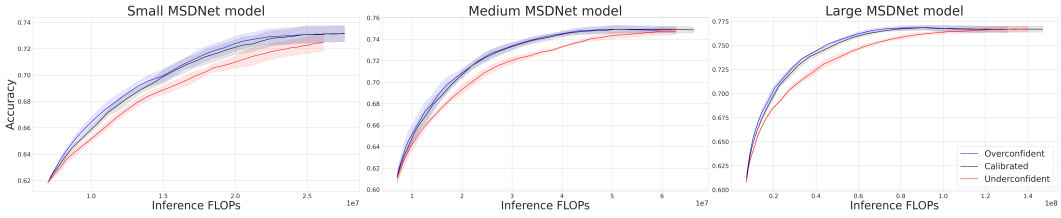

Figure 4: Results for MIE-calibrated MSDNet models.

D Alternative measure for early-exit failure prediction

EEFP Score measures the separability of two subsets (correctly and incorrectly predicted) for every possible threshold. However, it is unreasonable to assume that every threshold is equally important for early-exit models, and in practice, we care about a subset of thresholds. For example, for high computational budgets, the first threshold τ_1 tends to be close to 1.0, and separability on low thresholds $\tau_1 \to 0$ is irrelevant. Moreover, EEFP Score is threshold-agnostic and is measured for each exit separately.

In this section we consider an alternative way to estimate failure prediction, the **EEF1** score, which uses F1 instead of AUROC. EEF1 can be used to compare two multi-exit models for any budget by using the actual exit criterion threshold used during inference.

Let τ_j denote the threshold of the j-th classifier. We define an exit indicator as:

$$h_j(x) = \begin{cases} 1, & \text{if } c_j(x) \geq \tau_j, \\ 0, & \text{otherwise.} \end{cases}$$

Let $\mathcal{L}_j$ be the set of indices for which the model has not exited before the j-th classifier:

$$i \in \mathcal{L}_j \iff \forall_{l \in \{1,\dots,j-1\}}\ c_l(x_i) < \tau_l.$$

We define the **Early Exit F1 Score** for the j-th classifier as:

$$\text{EEF1}_j(\{x_i, y_{j,i}\}_{i \in \mathcal{L}_j}) = \text{F1}(\{h_j(x_i), \bar{y}_{j,i}\}_{i \in \mathcal{L}_j}).$$

To obtain a single score for each budget, we compute the arithmetic mean over all classifiers:

$$\text{EEF1}(\{x_i, y_{j,i}\}) = \frac{1}{J} \sum_{j=1}^{J} \text{EEF1}_j(\{x_i, y_{j,i}\}_{i \in \mathcal{L}_j}).$$

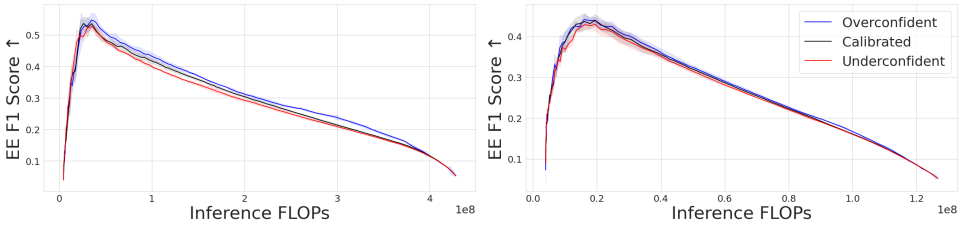

Figure 5: EEF1 scores for CIFAR-100 (left) and Tiny ImageNet (right)

The resulting EEF1 scores for models decalibrated via modified temperature (see Section 3.1) are shown in Figure 5. Like the EEFP scores, EEF1 indicates a small advantage for the overconfident model compared to the calibrated model. However, EEFP better separates the models' failure-prediction performance and is more straightforward to interpret; for these reasons, we decided to focus on EEFP score in the main paper.

E Experimental setup details

In this section, we describe the experimental setup used in our experiments in the main paper.

E.1 Architecture

We adopt the MSDNet architecture [7] with 7 blocks. We attach early-exit heads after each block, and additionally inside the blocks.

E.2 Training

For each experiment, we train three models with independent initializations and report the average performance across these three seeds. We train MSDNet models with a batch size of 512, using a learning rate of 1e-3 and AdamW optimizer [18] without weight decay. We use a cosine annealing scheduler with warm restarts and linear warm-up. As data augmentations, we apply random resizing, cropping, rotation, contrast adjustment, random erasing, Mixup [37], and CutMix [36]. All models are trained until convergence.

E.3 Calibration

To calibrate the early-exit model, we proceed with a gradient-based approach. We extract 2.5% of the samples from the training set, which were not used to optimize the model during training, and use them for the calibration phase. We then freeze the model's parameters, attach temperature scaling calibrators, and proceed by minimizing NLPD. Each early-exit head is optimized individually and has its own temperature.

E.4 Evaluation

During evaluation, we begin with the calibrated model and additionally consider two variants: an overconfident and an underconfident model. All three models share the same trained parameters and differ only in how the logits are transformed.

For a given data sample x, let

$$z_{j,1}, \ldots, z_{j,C}$$

denote the logits of the classifier in the calibrated model. In the temperature scaling experiment 3.1, the logits of the models are modified by a temperature parameter T as follows:

$$\frac{z_{j,1}}{T}, \ldots, \frac{z_{j,C}}{T}.$$

Softmax is then applied to each set of logits, and the confidence values are obtained from the maximum softmax probability, denoted as $c_j(x)$. In Section 3.2, instead of scaling the logits, the confidence is derived using the function $\hat{c}_j(x)$ in place of the standard $c_j(x)$.

For each model and each exit head, decision thresholds are determined using the validation set. To this end, we follow the threshold selection heuristic proposed in Huang et al. [7], Meronen et al. [19], which derives appropriate thresholds based on the distribution of confidence scores.

Given a predefined FLOPs budget, the network is expected to terminate at each exit for a certain fraction of the input samples. This allocation of samples across exits is controlled by a parameter q. At the j-th exit, the model is required to terminate for a fraction of samples defined as:

$$\text{exit-share}(j) = \frac{q^j}{\sum_{l=0}^{J-1} q^l} \tag{1}$$

During inference on the test set, each model computes its own confidence values and applies its own set of thresholds to decide when to exit and, therefore, which prediction to make.