A UNIFIED APPROACH TO ROUTING AND CASCADING FOR LLMS

Jasper Dekoninck, Maximilian Baader, Martin Vechev
Department of Computer Science
ETH Zurich, Switzerland
{jasper.dekoninck,mbaader,martin.vechev}@inf.ethz.ch

ABSTRACT

The availability of a wide range of large language models embedded in various agentic systems has significantly increased the potential of model selection strategies to improve the cost-performance tradeoff. Existing strategies involve either routing, where a single model is chosen per query, or cascading, which sequentially runs increasingly larger models until a satisfactory answer is found. However, current approaches face three key limitations: they (1) lack formal proofs of optimality, (2) fail to identify the conditions under which these strategies are most effective, and (3) are unable to combine both paradigms. To address this, we propose *cascade routing*, a unified framework that integrates routing and cascading into a theoretically optimal strategy. Further, we identify good quality estimators as the critical factor for the success of model selection. Finally, we show that cascade routing consistently outperforms the baselines by a large margin. [1]

1 INTRODUCTION

Large language models (LLMs) are applied in a wide range of tasks, some of which are easily handled by small models, while others require state-of-the-art LLMs. Thus, so-called model selection can significantly improve the cost-performance tradeoff by selecting the best model for each query.

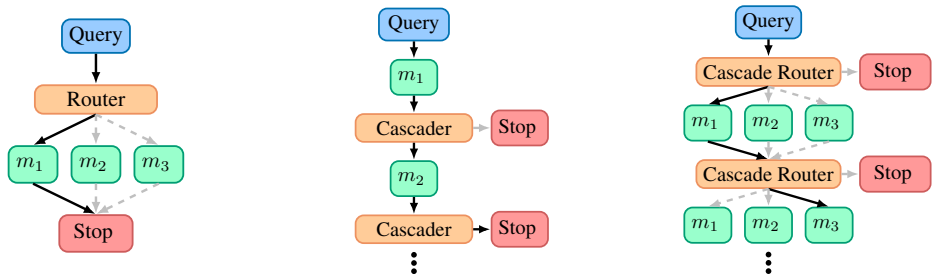
Routing and Cascading So far, two model selection strategies have been proposed. The first, routing, directs each query to a specific model from a set of available models (Chen et al., 2022), as illustrated in Fig. 1(a). This approach enables the selection of the most suitable expert for each query. The second strategy, cascading, processes a query through a sequence of increasingly larger models, stopping when a model produces an answer deemed sufficiently good (Chen et al., 2023), as illustrated in Fig. 1(b). Cascading allows simpler queries to be addressed by smaller models.

Current Limitations Despite their utility, both routing and cascading impose significant restrictions on the model selection process. In routing, the initial selection of a model is final, preventing any reconsideration. In cascading, each query must pass through all models without skipping. Further, a lack of theoretical understanding hinders the development of more effective model selection strategies. More importantly, prior work fails to provide insights into the limitations of model selection strategies and cannot identify the conditions under which they are useful in practical scenarios.

This Work: Cascade Routing To address these limitations, we first derive optimal routing and cascading strategies by framing them as linear optimization problems. Building on this analysis, we propose a new paradigm called cascade routing, which generalizes both routing and cascading. As illustrated in Fig. 1(c), cascade routing initially routes a query to any available model but keeps rerouting to different models until a model produces an answer of sufficient quality. We prove the optimality of cascade routing and show that it significantly outperforms both approaches.

Importance of Quality Estimation Leveraging our theoretical analysis, we find that accurate estimates of model performance and response quality are crucial for the effectiveness of model selection. For routing, reliable *ex-ante quality estimation*—the ability to predict whether a model will perform well on a given query—is essential. For cascading, robust *post-hoc quality estimation*—the ability to evaluate the quality of a model's response after generation—is critical.

[1] Code available at https://github.com/eth-sri/cascade-routing.



(a) Routing (b) Cascading (c) Cascade Routing (Ours)

Figure 1: Overview of three model selection strategies. Routing selects a single model for a query, cascading processes queries through a sequence of models, and cascade routing generalizes both.

Results We show that cascade routing significantly outperforms all baselines. Notably, cascade routing is consistently better than baselines and improves performance by up to 8% on RouterBench (Hu et al., 2024) and by 14% on SWE-Bench (Jimenez et al., 2024).

Key Contributions Our main contributions are:

- We derive optimal strategies for routing and cascading and obtain a new cascading strategy that is provably better than prior approaches (§2, §3).
- We introduce cascade routing, a new paradigm that combines the strengths of routing and cascading, and prove its optimality (§4).
- We conduct a thorough evaluation, demonstrating that cascade routing consistently outperforms the baselines and highlighting the critical role of quality estimation (§5).

2 Routing as Linear Optimization

We now derive the optimal routing strategy. Proofs are provided in App. A.

Routing In routing, our goal is to develop a strategy that selects the best language model for a given input query. Formally, let $\mathcal{X}$ represent the distribution over all possible queries, and suppose we have k language models $m_1, \ldots, m_k$ available for routing. Further, let Δ_k denote the set of all probability distributions over k variables. A routing strategy can then be defined as follows:

Definition 1 (Routing)**.** *A routing strategy s is a function $s\colon \mathcal{X} \to \Delta_k$ that maps a query $x \in \mathcal{X}$ to a probability distribution over models. $s_i(x)$ denotes the probability that m_i is selected for query x.*

Thus, a routing strategy selects a model by sampling from the distribution $s(x)$ for each query x.

Quality and Cost In routing, we seek to maximize the expected output quality of the selected model while adhering to a given cost budget B. Quality could measure model accuracy, user preference, or any other performance indicator. We define the quality function $q_i(x)$ as the output quality of model m_i on query x, and the cost function $c_i(x)$ as the cost of running model m_i on x.

However, since these functions are unknown in practice, we need estimators $\hat{q}_i(x)$ and $\hat{c}_i(x)$ for the output quality and cost. For the quality estimates one can train a small classifier (Hu et al., 2024; Shnitzer et al., 2023), while cost can be estimated via tokenization, API rates, or execution time.

Optimal Routing Using these estimators, we can formally define the optimal routing strategy:

Definition 2 (Optimal Routing)**.** *The optimal routing strategy s_{OPT} for a given cost budget B is the solution to the optimization problem that maximizes the expected output quality of the selected model while adhering to the budget:*

$$\max_{s} \quad \mathbb{E}_{x\in\mathcal{X}}\left(\sum\nolimits_{i=1}^{k} s_i(x)\hat{q}_i(x)\right) \quad \textit{s.t.} \quad \mathbb{E}_{x\in\mathcal{X}}\left(\sum\nolimits_{i=1}^{k} s_i(x)\hat{c}_i(x)\right) \leqslant B. \tag{1}$$

To solve this problem for a given query x, we show (see App. A) that the optimal routing strategy selects the model maximizing the cost-quality tradeoff $\tau_i(x, \lambda) = \hat{q}_i(x) - \lambda\hat{c}_i(x)$. Here, $\lambda \in \mathbb{R}^+$ is a hyperparameter that controls the balance between quality and cost based on the budget B.

However, it can occur that several models achieve the same optimal tradeoff for a given query. To address this, we define two strategies $s^{\lambda}_{\text{MIN}}(x)$ and $s^{\lambda}_{\text{MAX}}(x)$ that respectively select the cheapest and most expensive model obtaining the optimal tradeoff. The optimal routing strategy s_{OPT} is then:

Theorem 1 (Optimal Routing Strategy). *For a cost budget B, there exists a $\lambda \in \mathbb{R}^+$ and a $\gamma \in [0, 1]$ such that the optimal routing strategy s_{OPT} equals $\gamma s^{\lambda}_{\text{MIN}} + (1 - \gamma) s^{\lambda}_{\text{MAX}}$. Furthermore, all routing strategies that have an expected cost that is exactly equal to B and can be written as a convex combination of $s^{\lambda'}_{\text{MIN}}$ and $s^{\lambda'}_{\text{MAX}}$ for some $\lambda' \in \mathbb{R}^+$ achieve the same optimal quality.*

Due to the second part of Theorem 1, we only need to find *a* set of parameters λ and γ that achieve the cost budget. In App. A, we explain how to determine them using a single binary search over λ.

Ex-Ante Quality Estimation We explicitly distinguish the model's true quality $q_i(x)$ and the ex-ante quality estimate $\hat{q}_i(x)$. An optimal routing strategy will select a good model only if $q_i(x) \approx \hat{q}_i(x)$, otherwise the objective in Eq. (1) is not appropriate. Thus, even when routing is suited for the application, the strategy will fail if the quality estimates are inaccurate, making quality estimation central to routing. This is unlike cost estimates which in practice can be approximated more easily.

3 Cascading as Sequential Routing

We extend our analysis to a cascade, providing proofs for all statements in App. B.

Cascading In cascading, an input query is processed sequentially, typically through a chain of increasingly larger models. The cascade stops and returns a model's output once a certain condition is met. We will interpret cascading as a sequence of routing problems. To do so, we first define the models over which we need to route, which we refer to as *supermodels*.

Definition 3 (Supermodel). *A supermodel M is a sequence of models $(m_{i_1}, \ldots, m_{i_j})$ such that running a query through M is equivalent to running it through each of the models in the sequence. $\mathcal{M}$ denotes the set of all supermodels and by $M_{i:j}$ we denote the supermodel $(m_i, \ldots, m_j)$.*

In cascading, we only need to consider the supermodels $M_{1:1}, M_{1:2}, \ldots, M_{1:k}$.

Cascading as Sequential Routing Running a cascade on a sample x yields a sequence of steps, where at each step, the cascade determines whether to run the next model in the sequence or terminate. By step j, we have obtained outputs from the first $j-1$ models. To decide whether to continue and run m_j, we need to determine, in expectation, how well the supermodels $M_{1:j-1}, \ldots M_{1:k}$ will perform on the sample x. Once again, this performance is measured as having the highest expected output quality within a certain cost budget. If $M_{1:j-1}$ offers the best performance, we terminate the cascade and return its output, i.e., the output of m_{j-1}. Otherwise, if any of $M_{1:j}, \ldots, M_{1:k}$ has better performance, we continue the cascade and run m_j. Therefore, at step j, the cascade is equivalent to a routing strategy that selects the best supermodel from $M_{1:j-1}, \ldots, M_{1:k}$. Thus, a cascade can be formally defined as follows:

Definition 4 (Cascading Strategy). *A cascading strategy s is a sequence of routing strategies $(s^{(1)}, \ldots, s^{(k)})$ such that $s^{(j)}$ routes between the supermodels $M_{1:j-1}, \ldots, M_{1:k}$.*

Quality and Cost To apply Theorem 1 to find the optimal cascading strategy, we first need to derive the quality and cost estimates of the supermodels. Both of these can depend on the answers of previously computed models. Therefore, let $\hat{q}^{(j)}(x)$ and $\hat{c}^{(j)}(x)$ represent the updated estimates in step j after computing the first $j-1$ models. In App. B, we explain how the estimates associated with supermodel $M_{1:i}$ can be determined based on the estimates of the individual models.

Optimal Cascading We now leverage the optimal routing strategy from Theorem 1 to determine the optimal cascading strategy. As before, optimality is defined in terms of maximizing the expected output quality while adhering to a given cost budget. However, the budget is now only enforced over the entire cascade, and not over individual steps. This leads to a slightly different formulation:

Theorem 2 (Optimal Cascading Strategy). *For a given cost budget B, there exist $\lambda_1, \ldots, \lambda_k \in \mathbb{R}^+$ and a $\gamma \in [0, 1]$ such that the optimal cascading strategy $s_{\text{OPT}} = (s^{(1)}_{\text{OPT}}, \ldots, s^{(k)}_{\text{OPT}})$ is given by $s^{(j)}_{\text{OPT}} = \gamma s^{(j),\lambda_j}_{\text{MIN}} + (1 - \gamma) s^{(j),\lambda_j}_{\text{MAX}}$ where $s^{(j),\lambda_j}_{\text{MIN}}$ and $s^{(j),\lambda_j}_{\text{MAX}}$ are defined as in Theorem 1.*

The main difference between Theorem 2 and Theorem 1 is that not all combinations of hyperparameters $\lambda_1, \ldots, \lambda_k \in \mathbb{R}^+$ and $\gamma \in [0, 1]$ that achieve cost budget B are optimal. To find the optimal

hyperparameters, we first assume $\lambda_1 = \cdots = \lambda_k = \lambda$ and apply the binary search technique from §2 to determine λ. This is then used as initialization for a hyperparameter optimization tool, Hyperopt[2], to find the optimal values. Further details are provided in App. B.

Prior Work Prior work on cascading has often relied on strong assumptions to simplify cascading. The most common technique uses a threshold to decide whether to continue the cascade on an input x (Chen et al., 2023; Gupta et al., 2024). Specifically, in step j, the cascade continues if $\hat{q}_{j-1}^{(j)}(x) < \tau_j$ for some $\tau_j \in \mathbb{R}$. In App. B, we show that this thresholding strategy is a special case of our cascading strategy, but puts very restrictive assumptions on the quality and cost estimates.

Post-Hoc Quality Estimation This framework emphasizes the shift from ex-ante to post-hoc quality estimation, which is crucial for cascading. Cascading is only advantageous when the post-hoc quality estimate provides significantly better information than the ex-ante estimate. If this improvement is minimal, it would be more effective to directly route queries to the most suitable model, bypassing the cascading process. However, ex-ante estimates remain useful for cascading as they can inform the cascade whether future models are likely to improve the output quality.

4 Cascade Routing as Cascade Generalization

Cascading and routing are often orthogonal: routing relies on accurate ex-ante estimates, while cascading benefits from good post-hoc estimates. Therefore, we introduce cascade routing as a generalization of both. Proofs are provided in App. C.

Cascade Routing Cascade routing closely resembles cascading, but with one crucial difference: the routing strategy at step j routes between all possible supermodels, not just the supermodels $M_{1:j-1}, \ldots, M_{1:k}$. Therefore, both Definition 4 and Theorem 2 can be extended to this setting.

Definition 5 (Cascade Routing)**.** *A cascade routing strategy s is a sequence of routing strategies $(s^{(1)}, \ldots, s^{(k)})$ such that, for a given sample $x \in \mathcal{X}$, $s^{(j)}$ routes between all supermodels in $\mathcal{M}$ that start with the $j-1$ models that have already been computed for this query.*

Theorem 3 (Optimal Cascade Routing)**.** *For a given cost budget B, there exist $\lambda_1, \ldots, \lambda_k \in \mathbb{R}^+$ and a $\gamma \in \mathbb{R}^+$ such that the optimal cascade routing strategy $s_{\text{OPT}} = (s_{\text{OPT}}^{(1)}, \ldots, s_{\text{OPT}}^{(k)})$ is given by $s_{\text{OPT}}^{(j)} = \gamma s_{\text{MIN}}^{(j),\lambda_j} + (1-\gamma) s_{\text{MAX}}^{(j),\lambda_j}$ where $s_{\text{MIN}}^{(j),\lambda_j}$ and $s_{\text{MAX}}^{(j),\lambda_j}$ are defined as in Theorem 1.*

While cascade routing extends cascading and can therefore use the same hyperparameter optimization scheme, it also introduces additional challenges, namely computational complexity and model order determination, which we resolve in App. C.

5 Experimental Evaluation

In this section, we demonstrate that cascade routing significantly outperforms the baselines on several benchmarks. We distinguish benchmarks where accurate quality estimation is and is not available. Additional experiments on RouterBench (Hu et al., 2024) are provided in App. D.1. In App. D.2, we include an ablation study to examine the impact of our design choices on performance and runtime. For all details about the benchmarks, models, and estimators, we refer to App. E.

Accurate Quality Estimation We evaluate cascade routing on two benchmarks that allow accurate quality estimation. First, we use SWE-Bench (Jimenez et al., 2024) as a benchmark where accurate post-hoc quality estimation is available using the ground-truth test cases. Second, to simulate a use-case where ex-ante quality estimation is accurate, we evaluate the Math and Coder models from the Qwen-2.5 model family (Yang et al., 2024; Hui et al., 2024) on a combination of Minerva Math (Lewkowycz et al., 2022) and LiveCodeBench (Jain et al., 2024). We incorporate a sample's origin benchmark as a feature in the quality estimation model.

Results Table 1 (left) shows the results for both benchmarks. In SWE-Bench, our methods outperform baseline strategies by up to 14%. As expected, the routing strategy does not outperform the trivial baseline on this benchmark, as ex-ante quality estimates are insufficient. For Minerva

[2] `https://github.com/hyperopt/hyperopt`

Table 1: AUC scores on practical benchmarks with good quality estimates (left) and poor quality estimates (right). The highest numbers are bolded, and underlined numbers are within the 95% confidence intervals of the highest number. For a discussion on confidence intervals, refer to App. F.

	SWE-Bench		Math+Code	Classification		Open-Form	
	10 MODELS	5 MODELS	QWEN	LLAMA	GEMMA	LLAMA	GEMMA
Linear Interp.	40.51	38.64	39.63	74.28	61.68	79.11	54.10
Routing	40.47	39.40	47.46	74.92	<u>64.44</u>	79.32	58.40
Cascade (Baseline)	38.52	45.89	37.68	74.81	54.32	79.23	56.18
Cascade (Ours)	<u>53.20</u>	<u>50.94</u>	46.76	<u>75.46</u>	62.79	79.22	56.18
Cascade Routing (Ours)	**54.12**	**51.09**	**48.55**	**75.52**	**64.84**	**79.88**	**59.66**

Math and LiveCodeBench, the opposite trend holds true. With accurate ex-ante quality estimation, the routing strategy achieves strong performance, surpassing the baseline cascade strategy by 10%. However, the cascade routing strategy still outperforms all methods. Interestingly, despite poor post-hoc quality estimation, our cascading strategy nearly matches the performance of routing.

Poor Quality Estimation We perform experiments on classification and open-form reasoning tasks where there is no known accurate quality estimator. The classification benchmarks include ARC-Challenge (Clark et al., 2018), MMLU-Pro (Wang et al., 2024), and MixEval (Ni et al., 2024). For the open-form reasoning task, we use MMLU-Pro and GSM8k (Cobbe et al., 2021). In classification, models select a single option representing their answer, with no intermediate reasoning process. In contrast, open-form reasoning allows models to generate their answers after reasoning. Here, we evaluate two model families consisting of three models, LLAMA and GEMMA, and show similar numbers for MISTRAL in App. H. We create a quality estimator based on Gupta et al. (2024).

Results Table 1 (right) presents the results for the LLAMA and GEMMA model families across both benchmarks. Cascade routing consistently performs on par with or outperforms all baselines, though with much narrower margins reaching up to 1.2%. This reduced gain can be attributed to the fact that the quality and cost estimates are very noisy, leading to small performance gains for any model selection strategy, no matter how optimal.

6 RELATED WORK

Routing Routing is a key problem in machine learning, often used to direct queries to specialized models. Many works focus on model selection for natural language queries with known answers, training models to predict whether a given model will answer correctly (Chuang et al., 2024; Ding et al., 2024; Hari and Thomson, 2023; Liu et al., 2024; Jang et al., 2023; Nguyen et al., 2024; Sakota et al., 2024; Shnitzer et al., 2023). Routing is also applied in preference-based quality estimation (Lu et al., 2024; Ong et al., 2024), API selection (Chen et al., 2022), software agent task allocation (Zhang et al., 2024b), and token-level routing (Pichlmeier et al., 2024).

Cascading Cascading reduces inference costs by first using smaller models and escalating to larger ones if needed. Most methods rely on confidence thresholds (Chen et al., 2023; 2024; Ramírez et al., 2024; Varshney and Baral, 2022), while others use response variance of the smaller model (Madaan et al., 2023). Early stopping is another approach, stopping computation when intermediate layers provide sufficient information (Li et al., 2021; Schuster et al., 2022). Furthermore, quality estimation in cascading has been explored via uncertainty measures (Gupta et al., 2024; Jitkrittum et al., 2023), dynamic voting (Xue et al., 2023), and multi-objective metrics (Zhang et al., 2024a).

7 CONCLUSION

In this work, we presented a novel framework for routing and cascading that enabled us to propose theoretically optimal strategies for both paradigms. We combined these two into a new cascade routing technique. We showed that our optimal cascade improves significantly over prior work and that cascade routing outperforms all baselines consistently, especially given good quality and dost estimates. Our work provides a solid theoretical foundation for model selection and paves the way for future research, particularly in the area of improving the accuracy of quality and cost estimates.

REFERENCES

AI@Meta. Llama 3 model card. 2024. URL https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md.

Jacob Austin, Augustus Odena, Maxwell I. Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie J. Cai, Michael Terry, Quoc V. Le, and Charles Sutton. Program synthesis with large language models. *CoRR*, abs/2108.07732, 2021. URL https://arxiv.org/abs/2108.07732.

Dong Chen, Yueting Zhuang, Shuo Zhang, Jinfeng Liu, Su Dong, and Siliang Tang. Data shunt: Collaboration of small and large models for lower costs and better performance. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan, editors, *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 11249–11257. AAAI Press, 2024. doi: 10.1609/AAAI.V38I10.29003. URL https://doi.org/10.1609/aaai.v38i10.29003.

Lingjiao Chen, Matei Zaharia, and James Zou. Efficient online ML API selection for multi-label classification tasks. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 3716–3746. PMLR, 2022. URL https://proceedings.mlr.press/v162/chen22ad.html.

Lingjiao Chen, Matei Zaharia, and James Zou. Frugalgpt: How to use large language models while reducing cost and improving performance. *CoRR*, abs/2305.05176, 2023. doi: 10.48550/ARXIV.2305.05176. URL https://doi.org/10.48550/arXiv.2305.05176.

Yu-Neng Chuang, Helen Zhou, Prathusha Kameswara Sarma, Parikshit Gopalan, John Boccio, Sara Bolouki, and Xia Hu. Learning to route with confidence tokens. *CoRR*, abs/2410.13284, 2024. doi: 10.48550/ARXIV.2410.13284. URL https://doi.org/10.48550/arXiv.2410.13284.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the AI2 reasoning challenge. *ArXiv preprint*, abs/1803.05457, 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *ArXiv preprint*, abs/2110.14168, 2021.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Rühle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. Hybrid LLM: cost-efficient and quality-aware query routing. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=02f3mUtqnM.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 07 2024. URL https://zenodo.org/records/12608602.

Thomas Mesnard Gemma Team, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, and et al. Gemma. 2024. doi: 10.34740/KAGGLE/M/3301. URL https://www.kaggle.com/m/3301.

Neha Gupta, Harikrishna Narasimhan, Wittawat Jitkrittum, Ankit Singh Rawat, Aditya Krishna Menon, and Sanjiv Kumar. Language model cascades: Token-level uncertainty and beyond. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=KgaBScZ4VI.

Surya Narayanan Hari and Matt Thomson. Tryage: Real-time, intelligent routing of user prompts to large language models. *CoRR*, abs/2308.11601, 2023. doi: 10.48550/ARXIV.2308.11601. URL https://doi.org/10.48550/arXiv.2308.11601.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *Proc. of ICLR*, 2021.

Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, Benjamin Keigwin, Gaurav Ranganath, Kurt Keutzer, and Shriyash Kaustubh Upadhyay. Routerbench: A benchmark for multi-llm routing system. *CoRR*, abs/2403.12031, 2024. doi: 10.48550/ARXIV.2403.12031. URL https://doi.org/10.48550/arXiv.2403.12031.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Kai Dang, An Yang, Rui Men, Fei Huang, Xingzhang Ren, Xuancheng Ren, Jingren Zhou, and Junyang Lin. Qwen2.5-coder technical report. *CoRR*, abs/2409.12186, 2024. doi: 10.48550/ARXIV.2409.12186. URL https://doi.org/10.48550/arXiv.2409.12186.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *CoRR*, abs/2403.07974, 2024. doi: 10.48550/ARXIV.2403.07974. URL https://doi.org/10.48550/arXiv.2403.07974.

Joel Jang, Seungone Kim, Seonghyeon Ye, Doyoung Kim, Lajanugen Logeswaran, Moontae Lee, Kyungjae Lee, and Minjoon Seo. Exploring the benefits of training expert language models over instruction tuning. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 14702–14729. PMLR, 2023. URL https://proceedings.mlr.press/v202/jang23a.html.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b. *CoRR*, abs/2310.06825, 2023. doi: 10.48550/ARXIV.2310.06825.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R. Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Wittawat Jitkrittum, Neha Gupta, Aditya Krishna Menon, Harikrishna Narasimhan, Ankit Singh Rawat, and Sanjiv Kumar. When does confidence-based cascade deferral suffice? In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/1f09e1ee5035a4c3fe38a5681cae5815-Abstract-Conference.html.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay V. Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/18abbeef8cfe9203fdf9053c9c4fe191-Abstract-Conference.html.

Lei Li, Yankai Lin, Deli Chen, Shuhuai Ren, Peng Li, Jie Zhou, and Xu Sun. Cascadebert: Accelerating inference of pre-trained language models via calibrated complete models cascade. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta*

Cana, Dominican Republic, 16-20 November, 2021, pages 475–486. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.FINDINGS-EMNLP.43. URL https://doi.org/10.18653/v1/2021.findings-emnlp.43.

Yueyue Liu, Hongyu Zhang, Yuantian Miao, Van-Hoang Le, and Zhiqiang Li. Optllm: Optimal assignment of queries to large language models. *CoRR*, abs/2405.15130, 2024. doi: 10.48550/ARXIV.2405.15130. URL https://doi.org/10.48550/arXiv.2405.15130.

Keming Lu, Hongyi Yuan, Runji Lin, Junyang Lin, Zheng Yuan, Chang Zhou, and Jingren Zhou. Routing to the expert: Efficient reward-guided ensemble of large language models. In Kevin Duh, Helena Gómez-Adorno, and Steven Bethard, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 1964–1974. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.NAACL-LONG.109. URL https://doi.org/10.18653/v1/2024.naacl-long.109.

Aman Madaan, Pranjal Aggarwal, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappaganthu, Yiming Yang, Shyam Upadhyay, Mausam, and Manaal Faruqui. Automix: Automatically mixing language models. *CoRR*, abs/2310.12963, 2023. doi: 10.48550/ARXIV.2310.12963. URL https://doi.org/10.48550/arXiv.2310.12963.

Quang H. Nguyen, Duy C. Hoang, Juliette Decugis, Saurav Manchanda, Nitesh V. Chawla, and Khoa D. Doan. Metallm: A high-performant and cost-efficient dynamic framework for wrapping llms. *CoRR*, abs/2407.10834, 2024. doi: 10.48550/ARXIV.2407.10834. URL https://doi.org/10.48550/arXiv.2407.10834.

Jinjie Ni, Fuzhao Xue, Xiang Yue, Yuntian Deng, Mahir Shah, Kabir Jain, Graham Neubig, and Yang You. Mixeval: Deriving wisdom of the crowd from LLM benchmark mixtures. *CoRR*, abs/2406.06565, 2024. doi: 10.48550/ARXIV.2406.06565. URL https://doi.org/10.48550/arXiv.2406.06565.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Waleed Kadous, and Ion Stoica. Routellm: Learning to route llms with preference data. *CoRR*, abs/2406.18665, 2024. doi: 10.48550/ARXIV.2406.18665. URL https://doi.org/10.48550/arXiv.2406.18665.

OpenAI. GPT-4 technical report. *CoRR*, abs/2303.08774, 2023. doi: 10.48550/arXiv.2303.08774.

Josef Pichlmeier, Philipp Ross, and Andre Luckow. Domain-Aware LLM Routing During Generation . In *2024 IEEE International Conference on Big Data (BigData)*, pages 8235–8237, Los Alamitos, CA, USA, December 2024. IEEE Computer Society. doi: 10.1109/BigData62323.2024.10825152. URL https://doi.ieeecomputersociety.org/10.1109/BigData62323.2024.10825152.

Guillem Ramírez, Alexandra Birch, and Ivan Titov. Optimising calls to large language models with uncertainty-based two-tier selection. *CoRR*, abs/2405.02134, 2024. doi: 10.48550/ARXIV.2405.02134. URL https://doi.org/10.48550/arXiv.2405.02134.

Marija Sakota, Maxime Peyrard, and Robert West. Fly-swat or cannon? cost-effective language model choice via meta-modeling. In Luz Angelica Caudillo-Mata, Silvio Lattanzi, Andrés Muñoz Medina, Leman Akoglu, Aristides Gionis, and Sergei Vassilvitskii, editors, *Proceedings of the 17th ACM International Conference on Web Search and Data Mining, WSDM 2024, Merida, Mexico, March 4-8, 2024*, pages 606–615. ACM, 2024. doi: 10.1145/3616855.3635825. URL https://doi.org/10.1145/3616855.3635825.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/6fac9e316a4ae75ea244ddcef1982c71-Abstract-Conference.html.

Tal Shnitzer, Anthony Ou, Mírian Silva, Kate Soule, Yuekai Sun, Justin Solomon, Neil Thompson, and Mikhail Yurochkin. Large language model routing with benchmark datasets. *CoRR*, abs/2309.15789, 2023. doi: 10.48550/ARXIV.2309.15789. URL https://doi.org/10.48550/arXiv.2309.15789.

Neeraj Varshney and Chitta Baral. Model cascading: Towards jointly improving efficiency and accuracy of NLP systems. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 11007–11021. Association for Computational Linguistics, 2022. doi: 10.18653/V1/2022.EMNLP-MAIN.756. URL https://doi.org/10.18653/v1/2022.emnlp-main.756.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *CoRR*, abs/2406.01574, 2024. doi: 10.48550/ARXIV.2406.01574. URL https://doi.org/10.48550/arXiv.2406.01574.

Mingfeng Xue, Dayiheng Liu, Wenqiang Lei, Xingzhang Ren, Baosong Yang, Jun Xie, Yidan Zhang, Dezhong Peng, and Jiancheng Lv. Dynamic voting for efficient reasoning in large language models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 3085–3104. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.FINDINGS-EMNLP.203. URL https://doi.org/10.18653/v1/2023.findings-emnlp.203.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *CoRR*, abs/2409.12122, 2024. doi: 10.48550/ARXIV.2409.12122. URL https://doi.org/10.48550/arXiv.2409.12122.

Kai Zhang, Liqian Peng, Congchao Wang, Alec Go, and Xiaozhong Liu. LLM cascade with multi-objective optimal consideration. *CoRR*, abs/2410.08014, 2024a. doi: 10.48550/ARXIV.2410.08014. URL https://doi.org/10.48550/arXiv.2410.08014.

Kexun Zhang, Weiran Yao, Zuxin Liu, Yihao Feng, Zhiwei Liu, Rithesh Murthy, Tian Lan, Lei Li, Renze Lou, Jiacheng Xu, Bo Pang, Yingbo Zhou, Shelby Heinecke, Silvio Savarese, Huan Wang, and Caiming Xiong. Diversity empowers intelligence: Integrating expertise of software engineering agents, 2024b. URL https://arxiv.org/abs/2408.07060.

A ROUTING

We first prove Theorem 1 and then explain how to determine the optimal hyperparameters λ and γ.

A.1 PROOF THEOREM 1

To prove Theorem 1, we first rewrite the routing optimization problem in Eq. (1) as a linear program over functions $s : \mathcal{X} \to \mathbb{R}^k$ instead of functions $s : \mathcal{X} \to \Delta_k$. This makes the optimization problem more tractable. Specifically, Eq. (1) can be rewritten as follows:

$$\begin{aligned} \max_r \quad & \mathbb{E}_{x\sim\mathcal{X}}\left[\sum_{i=1}^{k} s_i(x)\hat{q}_i(x)\right] \\ \text{s.t.} \quad & \mathbb{E}_{x\sim\mathcal{X}}\left[\sum_{i=1}^{k} s_i(x)\hat{c}_i(x)\right] \leqslant B \\ & \forall i \in \{1, ..., k\} : \forall x \in \mathcal{X} : s_i(x) \geq 0 \wedge \sum_{j=1}^{k} s_j(x) = 1 \end{aligned} \tag{2}$$

We then rewrite Theorem 1 to allow for a more exact formulation of the optimal routing strategy:

Theorem 4. *(Optimal Routing Strategy) Suppose there exists an admissible solution to the set of constraints in Eq.* (2)*. For any $\lambda \in \mathbb{R}^+$, let S_λ be the set of routing strategies s that satisfy the following constraints:*

$$\forall x \in \mathcal{X}, \forall i \in \{1, ..., k\} : \hat{q}_i(x) - \lambda\hat{c}_i(x) < \max_j \hat{q}_j(x) - \lambda\hat{c}_j(x) \Rightarrow s_i(x) = 0 \tag{3}$$

If there exists a strategy in S_0 that has a cost less than or equal to B, then this strategy achieves the optimal quality. Otherwise, there exists a $\lambda^* \in \mathbb{R}^+$ such that S_λ contains a routing strategy that has exactly cost B and all routing strategies in $\bigcup_{\lambda\in\mathbb{R}^+} S_\lambda$ that have cost B achieve the same optimal quality.

There is one extra condition mentioned here that we omitted in the main text. The requirement of having at least an admissible solution to the constraints in Eq. (2) is necessary to ensure that the set of possible solutions to Eq. (2) is not empty. For instance, the cost budget B can be too low such that even running the cheapest model for each query is too expensive.

The formulation of s_{OPT} as a convex combination of s^λ_{MIN} and s^λ_{MAX} is a direct consequence of Theorem 4. Indeed, let λ^* be as defined in Theorem 4. Then $s^{\lambda^*}_{\text{MIN}}$, resp. $s^{\lambda^*}_{\text{MAX}}$, must have the lowest, resp. highest, cost among all routing strategies in S_{λ^*}. Since there is a routing strategy in S_{λ^*} that has cost B, there must exist a convex combination of $s^{\lambda^*}_{\text{MIN}}$ and $s^{\lambda^*}_{\text{MAX}}$ that also has cost B and thus achieves the optimal quality.

We first prove several lemmas before proving the theorem.

Lemma 1. *S_λ is non-empty and convex for all $\lambda \in \mathbb{R}^+$.*

Proof. Non-emptiness follows from the fact that the routing strategy that assigns all probability mass for a sample x to a model i for which $\hat{q}_i(x) - \lambda\hat{c}_i(x)$ is maximal, is in S_λ. For convexity, let $s^{(1)}, s^{(2)} \in S_\lambda$ be arbitrary. Let s^γ be the convex combination of $s^{(1)}$ and $s^{(2)}$ with weight $\gamma \in [0, 1]$. Let $x \in \mathcal{X}$ be arbitrary. Then, $s^\gamma_i(x) > 0$ if and only if $s^{(1)}_i(x) > 0$ or $s^{(2)}_i(x) > 0$. Since $s^{(1)}, s^{(2)} \in S_\lambda$, we have $\hat{q}_i(x) - \lambda\hat{c}_i(x) \geqslant \max_j \hat{q}_j(x) - \lambda\hat{c}_j(x)$ for all i such that $s^{(1)}_i(x) > 0$ or $s^{(2)}_i(x) > 0$. This implies that $\hat{q}_i(x) - \lambda\hat{c}_i(x) \geqslant \max_j \hat{q}_j(x) - \lambda\hat{c}_j(x)$ for all i such that $s^\gamma_i(x) > 0$. Thus, $s^\gamma \in S_\lambda$. □

Lemma 2. *Let $\lambda_1 < \lambda_2$ and $s^{(1)}$, resp. $s^{(2)}$ be arbitrary routing strategies in S_{λ_1}, resp. S_{λ_2}. Then, the cost of $s^{(1)}$ is greater or equal to the cost of $s^{(2)}$, i.e.,*

$$\mathbb{E}_{x\sim\mathcal{X}}\left[\sum_{i=1}^{k} s^{(1)}_i(x)\hat{c}_i(x)\right] \geqslant \mathbb{E}_{x\sim\mathcal{X}}\left[\sum_{i=1}^{k} s^{(2)}_i(x)\hat{c}_i(x)\right]$$

Proof. We show that for any $x \in \mathcal{X}$, the cost of $s^{(1)}$ is greater or equal to the cost of $s^{(2)}$. Let $x \in \mathcal{X}$ be arbitrary. Suppose $s^{(1)}$ is strictly cheaper than $s^{(2)}$. Then, there must exist a model pair i, j such that $\hat{c}_i(x) < \hat{c}_j(x)$, $s_i^{(1)}(x) > s_i^{(2)}(x) \geqslant 0$, and $s_j^{(2)}(x) > s_j^{(1)}(x) \geqslant 0$. However, $s_i^{(1)}(x) > 0$ implies

$$\hat{q}_i(x) - \lambda_1 \hat{c}_i(x) \geqslant \hat{q}_j(x) - \lambda_1 \hat{c}_j(x).$$

Furthermore, since $\lambda_1 - \lambda_2 < 0$, we have

$$\hat{c}_i(x)(\lambda_1 - \lambda_2) > \hat{c}_j(x)(\lambda_1 - \lambda_2).$$

Adding these two inequalities gives

$$\hat{q}_i(x) - \lambda_2 \hat{c}_i(x) > \hat{q}_j(x) - \lambda_2 \hat{c}_j(x),$$

which is a contradiction with $s_j^{(2)}(x) > 0$. Thus, the cost of $s^{(1)}$ is greater or equal to the cost of $s^{(2)}$. □

Lemma 3. *Let Λ be the set of points $\lambda \in \mathbb{R}$ such that there exist an $x \in \mathcal{X}$ and $i \neq j$ such that $\hat{q}_i(x) - \lambda \hat{c}_i(x) = \hat{q}_j(x) - \lambda \hat{c}_j(x)$. Let $\lambda_1 < \lambda_2$ be such that $[\lambda_1, \lambda_2] \cap \Lambda = \emptyset$. Then, $S_{\lambda_1} = S_{\lambda_2}$. Furthermore, if $[\lambda_1, \lambda_2] \cap \Lambda = \{\lambda^*\}$, then $S_\lambda \subset S_{\lambda^*}$ for all $\lambda \in [\lambda_1, \lambda_2]$.*

Proof. We first show the first statement by showing that $S_{\lambda_1} \setminus S_{\lambda_2} = \emptyset$. $S_{\lambda_2} \setminus S_{\lambda_1} = \emptyset$ follows analogously. Suppose there exists a routing strategy $s \in S_{\lambda_1} \setminus S_{\lambda_2}$. Since $s \notin S_{\lambda_2}$, there must exist an $x \in \mathcal{X}$ and model i such that $s_i(x) > 0$ and $\hat{q}_i(x) - \lambda_2 \hat{c}_i(x) < \max_j \hat{q}_j(x) - \lambda_2 \hat{c}_j(x)$. Let j be an index such that $\hat{q}_i(x) - \lambda_2 \hat{c}_i(x) < \hat{q}_j(x) - \lambda_2 \hat{c}_j(x)$. Since $s \in S_{\lambda_1}$, we have $\hat{q}_i(x) - \lambda_1 \hat{c}_i(x) \geqslant \hat{q}_j(x) - \lambda_1 \hat{c}_j(x)$. By continuity, there exists a $\lambda \in [\lambda_1, \lambda_2]$ such that $\hat{q}_i(x) - \lambda \hat{c}_i(x) = \hat{q}_j(x) - \lambda \hat{c}_j(x)$, which is a contradiction with $[\lambda_1, \lambda_2] \cap \Lambda = \emptyset$.

Now suppose $[\lambda_1, \lambda_2] \cap \Lambda = \{\lambda^*\}$. Let $\lambda \in [\lambda_1, \lambda^*)$ be arbitrary and let $s \in S_\lambda$ be arbitrary. We show that $s \in S_{\lambda^*}$. For $\lambda \in (\lambda^*, \lambda_2]$, the proof is completely analogous. By contradiction, suppose there exists an $x \in \mathcal{X}$ and model i such that $s_i(x) > 0$ and $\hat{q}_i(x) - \lambda^* \hat{c}_i(x) < \max_j \hat{q}_j(x) - \lambda^* \hat{c}_j(x)$. This means there exists a model j such that $\hat{q}_i(x) - \lambda^* \hat{c}_i(x) < \hat{q}_j(x) - \lambda^* \hat{c}_j(x)$. Since $s \in S_\lambda$, we know that $\hat{q}_i(x) - \lambda \hat{c}_i(x) \geqslant \hat{q}_j(x) - \lambda \hat{c}_j(x)$. This implies that there must exist a $\lambda' \in [\lambda_1, \lambda^*)$ such that $\hat{q}_i(x) - \lambda' \hat{c}_i(x) = \hat{q}_j(x) - \lambda' \hat{c}_j(x)$. However, this is a contradiction with $[\lambda_1, \lambda^*) \cap \Lambda = \emptyset$. Thus, $s \in S_{\lambda^*}$. □

In what follows, we will assume that $|\Lambda| < \infty$. This is a very minor assumption. For instance, if $\hat{q}$ and $\hat{c}$ only take on a finite amount of values, this is trivially satisfied. Since estimators are implemented on a computer, they will always have a finite precision, meaning that $\hat{q}$ and $\hat{c}$ will only take on a finite amount of values.

Lemma 4. *Let $\lambda_1 < \lambda_2$ and $s^{(1)}$, resp. $s^{(2)}$ be arbitrary routing strategies in S_{λ_1}, resp. S_{λ_2}, with costs resp. B_1 and B_2. Then, for any $B \in [B_1, B_2]$ there exists a $\lambda \in [\lambda_1, \lambda_2]$ such that S_λ contains a routing strategy that has exactly cost B.*

Proof. Let $B \in [B_1, B_2]$ be arbitrary. If $B = B_1$ or $B = B_2$, the statement is trivially true. Therefore, suppose $B \in (B_1, B_2)$. Let Λ be as defined in Lemma 3. By Lemma 2, there exists a $\lambda^* \in [\lambda_1, \lambda_2]$ such that all strategies in S_λ for $\lambda < \lambda^*$, resp. $\lambda > \lambda^*$, have cost at least, resp. at most, B. If $\lambda^* \notin \Lambda$, then the first part of Lemma 3, together with $|\Lambda| < \infty$, implies that $S_{\lambda^*} = S_{\lambda^* - \epsilon} = S_{\lambda^* + \epsilon}$ for some $\epsilon > 0$. All the strategies in S_{λ^*} must therefore have cost both at least and at most B, meaning they should equal B. We can therefore assume that $\lambda^* \in \Lambda$. By Lemma 3 and $|\Lambda| < \infty$, there is en $\epsilon > 0$ such that $S_{\lambda^* - \epsilon} \subset S_{\lambda^*}$ and $S_{\lambda^* + \epsilon} \subset S_{\lambda^*}$. Let $s^- \in S_{\lambda^* - \epsilon}$ and $s^+ \in S_{\lambda^* + \epsilon}$ be arbitrary. Let s^γ be the convex combination of s^- and s^+ with weight $\gamma \in [0, 1]$. Since $s^-, s^+ \in S_{\lambda^*}$, we have $s^\gamma \in S_{\lambda^*}$ by Lemma 1. Denote by B^-, resp. B^+, the cost of s^-, resp s^+. Furthermore, the cost of s^γ is $\gamma B^- + (1 - \gamma) B^+$. Since $B \in [B^-, B^+]$, there exists a $\gamma \in [0, 1]$ such that s^γ has cost exactly B. □

We can now prove the theorem.

Proof. If S_0 contains a solution that has cost less than or equal to B, then this solution trivially achieves the optimal quality. Thus, for the rest of the proof we can assume that the cost of every solution in S_0 is greater than B. For $\lambda \to \infty$, S_λ contains the solution that assigns all probability mass to the model with the lowest cost. Since there is an admissible solution, this solution necessarily has cost less than B. Therefore, by Lemma 4, there exists a $\lambda^* \in \mathbb{R}$ such that S_{λ^*} contains a routing strategy that has exactly cost B.

Let s be an arbitrary routing strategy in $\bigcup_{\lambda \in \mathbb{R}^+} S_\lambda$ that has cost B. Specifically, let $s \in S_\lambda$. Let s' be any other routing strategy that is an admissible solution to the optimization problem. Then:

$$
\begin{aligned}
\mathbb{E}_{x \in X}\left[\sum_{i=1}^{k} s'_i(x)\hat{q}_i(x)\right] &= \mathbb{E}_{x \in X}\left[\sum_{i=1}^{k} s'_i(x)\hat{q}_i(x) - \lambda B + \lambda B\right] \\
&\leqslant \mathbb{E}_{x \in X}\left[\sum_{i=1}^{k} s'_i(x)\left(\hat{q}_i(x) - \lambda \hat{c}_i(x)\right) + \lambda B\right] \\
&\leqslant \mathbb{E}_{x \in X}\left[\sum_{i=1}^{k} s_i(x)\left(\hat{q}_i(x) - \lambda \hat{c}_i(x)\right) + \lambda B\right] \\
&= \mathbb{E}_{x \in X}\left[\sum_{i=1}^{k} s_i(x)\hat{q}_i(x)\right]
\end{aligned}
$$

Thus, s achieves the optimal quality.

□

A.2 Hyperparameters

Finally, we explain how to determine the optimal hyperparameters λ and γ. To do so, we estimate the cost of a strategy using a validation dataset D that is representative of the query distribution $\mathcal{X}$. We then perform a hyperparameter search to find optimal values of λ and γ. By leveraging several properties of routing strategies (see before), one can show that this hyperparameter search can be reduced to a single binary search over λ, enabling a quick and efficient hyperparameter optimization process.

B Cascading

We first determine the quality and cost estimates associated with supermodels (App. B.1). We then prove the optimality of our cascading strategy (App. B.2). Finally, we discuss the conditions under which the thresholding strategy by prior work is equivalent to our cascading strategy (App. B.3).

B.1 Quality and Cost Estimates

We determine the quality and cost estimates associated with supermodel $M_{1:i}$, denoted as $\hat{q}^{(j)}_{1:i}(x)$ and $\hat{c}^{(j)}_{1:i}(x)$. Trivially, the cost of the supermodel is equal to the sum of the individual model costs. The quality of a supermodel, however, is governed by the best model within it. Thus, it equals $\mathbb{E}_{\hat{q}_i}[\max(\hat{q}_1(x), \ldots, \hat{q}_i(x))]$, where the expected value reflects the uncertainty in each quality estimate. Specifically, each quality estimate $\hat{q}_i(x)$ is modeled as a random variable estimating the true quality $q_i(x)$. This is crucial since ignoring uncertainty would falsely assume that the quality of a supermodel is always equal to the best model within it, even though the best model may return a poor answer, while another returns a good one. To estimate the uncertainties associated with the estimates, we compute the variance of $\hat{q}^{(j)}_i(x) - \hat{q}^{(k)}_i(x)$ over a validation dataset.

B.2 Proof of Theorem 2

To prove Theorem 2, we heavily rely on the results derived in App. A. As explained in §3, cascading can be reinterpreted as a sequence of routing problems. However, to prove optimality, we need to be slightly more careful with the exact formulation of the problem.

At step j, the cascading strategy needs to decide whether to stop the cascade or to continue to the next model. It should continue to the next model if any of the supermodels $M_{1:j}, \ldots, M_{1:k}$ is better to run than $M_{1:j-1}$ for some measure of 'better'. Therefore, the cascading strategy is indeed performing a routing operation between the supermodels $M_{1:j-1}, \ldots, M_{1:k}$.

However, the optimization problem does slightly change compared to the routing problem. First of all, for each query $x \in \mathcal{X}$, there is a possibility that the cascade is stopped before step j. Therefore, the cascade should not aim to optimize the quality at step j for such a query, since it would not have any effect on the overall quality of the cascade. Furthermore, the budget B is only enforced over the entire cascade, and not over the individual steps. Since the problem changes through steps, it is not required that the cost of the router at step j is exactly equal to B.

Therefore, we reformulate cascading using an inner and outer optimization problem. The inner optimization problem aims to find the optimal routing strategy at step j for a given budget B_j. The outer optimization problem aims to find the optimal budget B_j for each step j such that the overall quality of the cascade is maximized under the constraint that the total cost of the cascade is at most B.

To formulate this more exactly, let $P_j(M)$ be the probability that the cascade computed supermodel M by step j. Then, the inner optimization problem at step j can be formulated as:

$$
\begin{aligned}
\max_{r^{(j)}} \quad & \mathbb{E}_{x\sim\mathcal{X}}\left[P_j(M_{1:j-1}) \sum_{i=j-1}^{k} r_{1:i}(x)\hat{q}^{(j)}_{1:i}(x)\right] \\
\text{s.t.} \quad & \mathbb{E}_{x\sim\mathcal{X}}\left[P_j(M_{1:j-1}) \sum_{i=j-1}^{k} r_{1:i}(x)\hat{c}^{(j)}_{1:i}(x)\right] \leqslant B_j \\
& \forall i \in \{j-1, ..., k\} : \forall x \in \mathcal{X} : r_{1:i}(x) \geq 0 \wedge \sum_{i=j-1}^{k} r_{1:i}(x) = 1
\end{aligned}
\tag{4}
$$

Note that $P_j(M_{1:j-1})$ can be incorporated in the quality and cost estimates. This leaves us with the exact same optimization problem as the routing problem, but with a different budget B_j. Since the chosen model only depends on the maximization of $P_j(M_{1:j-1})\hat{q}^{(j)}_i(x) - \lambda_j P_j(M_{1:j-1})\hat{c}^{(j)}_i(x)$, the probability $P_j(M_{1:j-1})$ can be divided out of the optimization problem.

The inner optimization problems prove the existence of optimal routing strategies at each step j with parameters λ_j. We note that there only needs to be one parameter γ that determines the convex combination since the budget B is only enforced over the entire cascade.

Let us denote the quality and cost of the entire cascading strategy for given parameters $\lambda_1, \ldots, \lambda_k$ and γ as $Q(\lambda_1, \ldots, \lambda_k, \gamma)$ and $C(\lambda_1, \ldots, \lambda_k, \gamma)$ respectively. Then, the outer optimization problem can be formulated as:

$$
\begin{aligned}
\max_{\lambda_1,\ldots,\lambda_k,\gamma} \quad & Q(\lambda_1, \ldots, \lambda_k, \gamma) \\
\text{s.t.} \quad & C(\lambda_1, \ldots, \lambda_k, \gamma) \leqslant B
\end{aligned}
\tag{5}
$$

To solve this outer optimization problem, we simply perform a hyperparameter search over the budgets $B_1, \ldots, B_k$ using a hyperparameter optimization search as discussed in §3.

B.3 PRIOR APPROXIMATIONS

Intuitively, the thresholding cascading strategy can be seen as a special case of our cascading strategy. However, it puts very restrictive assumptions on the quality and cost estimates. We first formally state these conditions:

Corollary 1 (Optimal Threshold Strategy)**.** *Under minor technical assumptions, the thresholding strategy is equivalent to our cascading strategy if and only if the following conditions hold:* $\hat{c}_i^{(j)}(x)$ *is independent of* x *for all* $i, j \in \{1, \dots, k\}$*,* $\hat{q}_i^{(j)}(x)$ *is independent of* x *for all* $i \geqslant j$*, and* $\hat{q}_{1:i}^{(j)}(x)$ *is equal to* $\hat{q}_i^{(j)}(x)$*.*

Before proving Corollary 1, we first need to define what we exactly mean by equivalency. For this purpose, let $\mathcal{C}_1$ be defined as follows:

$$\mathcal{C}_1 = \left\{ s \mid s \text{ is a cascading strategy with parameters } \lambda_1, \dots, \lambda_k, \gamma = 0 \text{ using estimates } \hat{q}^{(j)}, \hat{c}^{(j)} \right\}$$

Similarly, let $\mathcal{C}_2$ be defined as follows:

$$\mathcal{C}_2 = \left\{ s \mid s \text{ is a thresholding strategy with parameters } \tau_1, \dots, \tau_k \text{ using estimates } \hat{q}^{(j)}, \hat{c}^{(j)} \right\}$$

We note that we set $\gamma = 0$ since the thresholding strategy is deterministic. We therefore restrict the cascading strategy to be deterministic as well.

We define the equivalence between the two sets as follows:

Definition 6 (Equivalence of Strategies)**.** *We say a set of strategies* $\mathcal{C}_1$ *is equivalent to another set of strategies* $\mathcal{C}_2$*, denoted as* $\mathcal{C}_1 \equiv \mathcal{C}_2$*, if for all* $s_0 \in \mathcal{C}_1 \cup \mathcal{C}_2$ *there exists a* $s_1 \in \mathcal{C}_1$*, and a* $s_2 \in \mathcal{C}_2$ *such that for all* $x \in \mathcal{X}$*,* s_0*,* s_1 *and* s_2 *take the same decisions on* x*.*

We can now more accurately state the conditions under which the thresholding strategy is equivalent to the optimal strategy.

Corollary 2 (Optimal Thresholding Strategy)**.** *Let* $\mathcal{C}_1$*,* $\mathcal{C}_2$ *be defined as above. Then,* $\mathcal{C}_1 \equiv \mathcal{C}_2$ *if and only if there exists alternative quality and cost estimates* $\hat{q}_i^{(j)'}(x)$ *and* $\hat{c}_i^{(j)'}(x)$ *with associated set of cascading strategies* $\mathcal{C}_1'$ *such that* $\mathcal{C}_1 \equiv \mathcal{C}_1'$ *and the following conditions hold on these alternative quality and cost estimates:* $\hat{c}_i^{(j)'}(x)$ *is independent of* x *and bigger than* 0*,* $\hat{q}_i^{(j)'}(x)$ *is independent of* x *for all* $i \geqslant j$*, and* $\hat{q}_{1:i}^{(j)'}(x)$ *is equal to* $\hat{q}_i^{(j)'}(x)$*.*

The main difference between Corollary 2 and Corollary 1 is that we impose the possibility of alternative quality and cost estimates. However, this does not really influence equivalency in the intuitive sense. Indeed, one could alternatively phrase the corollary as follows: the thresholding strategy is equivalent to any of our cascading strategies if and only if it is possible to construct alternative estimates such that the conditions hold.

Proof. We note that the cascade $s \in \mathcal{C}_1$ continues on a sample if the following condition holds:

$$\hat{q}_{1:j-1}^{(j)}(x) - \lambda_j \hat{c}_{1:j-1}^{(j)}(x) < \max_{i \in \{j, \dots, k\}} \hat{q}_{1:i}^{(j)}(x) - \lambda_j \hat{c}_{1:i}^{(j)}(x) \tag{6}$$

If $\mathcal{C}_1 \equiv \mathcal{C}_1'$, it is clear that Eq. (6) reduces to the thresholding strategy for all strategies in $\mathcal{C}_1'$. Indeed, for any $s \in \mathcal{C}_1'$, set $\tau_j = \max_{i \in \{j, \dots, k\}} \hat{q}_{1:i}^{(j)} - \lambda_j \hat{c}_{j:i}^{(j)}$ and the thresholding strategy is equivalent to s. Alternatively, if $s \in \mathcal{C}_2$, suppose $\max_{i \in \{j, \dots, k\}} \hat{q}_{1:i}^{(j)} - \lambda_j \hat{c}_{j:i}^{(j)} = \hat{q}_{1:i}^{(j)} - \lambda_j \hat{c}_{j:i}^{(j)}$ for some index i. Then, set $\lambda_j = \tau_j / \hat{c}_{j:i}^{(j)} - \hat{q}_{1:i}^{(j)} / \hat{c}_{j:i}^{(j)}$ and the cascading strategy is equivalent to s. Therefore, $\mathcal{C}_1 \equiv \mathcal{C}_1' \equiv \mathcal{C}_2$.

Suppose now that $\mathcal{C}_1 \equiv \mathcal{C}_2$. We construct alternative quality and cost estimates $\hat{q}_i^{(j)'}(x)$ and $\hat{c}_i^{(j)'}(x)$ such that the conditions hold and such that $\mathcal{C}_1 \equiv \mathcal{C}_1'$. For this purpose, we define $\hat{c}_i^{(j)'}(x) = 1$ for all $i, j \in \{1, \dots, k\}$, $\hat{q}_i^{(j)'}(x) = 1$ for all $i \geqslant j$, and $\hat{q}_i^{(j)'}(x) = \hat{q}_i^{(j)}(x)$ otherwise. Furthermore, we set $\hat{q}_{1:i}^{(j)'}(x) = \hat{q}_i^{(j)'}(x)$ for all $i, j \in \{1, \dots, k\}$. The equivalence of $\mathcal{C}_1'$ and $\mathcal{C}_2$ can now be proven analogously to the previous paragraph. Therefore, $\mathcal{C}_1 \equiv \mathcal{C}_1' \equiv \mathcal{C}_2$. □

C CASCADE ROUTING

We first note that the proof of the optimality of the cascade routing strategy is equivalent to the proof of the optimality of the cascade strategy, except that the expectation in the optimization problem Eq. (4) is now not only over $x \in X$, but also over all possible supermodels that were computed by step $j-1$. However, this does not change the optimization problem, and the proof is completely analogous to the proof given in §3.

We now explain and mitigate two challenges that arise when applying cascade routing in practice: the model order and the number of supermodels.

Model Order In cascading, the model order is predetermined, and the routing strategy only decides whether to proceed with the next model in the sequence. In contrast, cascade routing must dynamically determine the order in which models are computed. Despite this, both the estimated quality $\hat{q}_M^{(j)}(x)$ and cost $\hat{c}_M^{(j)}(x)$ of a supermodel M are order-independent. Therefore, supermodels that contain the same models in a different order will have the same associated cost and quality. To mitigate this, we sort the models within the selected supermodel by cost and compute the cheapest one first. This approach aligns with cascading, where more expensive models are only used if cheaper models do not suffice.

Number of Supermodels In cascading, the quality and cost must be computed for a maximum of k supermodels at each step. However, in cascade routing, the number of supermodels grows exponentially, leading to the need to evaluate up to 2^k supermodels. This increase can become prohibitively costly, particularly since the model selection process must remain computationally negligible with respect to model computation. To mitigate this, we leverage so-called negative marginal gains. Specifically, if a model m in a supermodel M negatively impacts the quality-cost tradeoff, all supermodels containing all models in M can be pruned from the search space. Since this negative contribution is quite common, this allows us to prune the search space significantly. More formally, this pruning operation relies on the following lemma:

Lemma 5 (Negative Marginal Gain)**.** *Let $M \in \mathcal{M}$ and m be any model in M. Let the marginal gain of m w.r.t. M be defined as $\tau_M(x,\lambda) - \tau_{M\setminus\{m\}}(x,\lambda)$. Then, if the marginal gain of m w.r.t. M is strictly negative for a given query, the optimal cascade routing strategy will never run a supermodel $M' \in \mathcal{M}$ that contains all models in M.*

To prove the lemma, we first prove the following lemma.

Lemma 6. *Let $Q_1, ..., Q_k$ be distributions. Let $\mathcal{S}$ be the superset of $\{1, ..., k\}$. Then $f : \mathcal{S} \to \mathbb{R}$ defined as $f(S) = \mathbb{E}(\max_{i\in S} Q_i)$ is submodular. Here, we define $\max_{i\in\emptyset} Q_i = -\infty$*

Proof. Let $T \subset S \subset \{1, \dots, k\}$ and $j \in \{1, \dots, k\}$ be arbitrary. To show the submodularity of f, we need to show that

$$f(T \cup \{j\}) - f(T) \geq f(S \cup \{j\}) - f(S).$$

We can write:

$$\begin{aligned}
f(S \cup \{j\}) - f(S) &= \mathbb{E}(\max_{i\in S\cup\{j\}} Q_i) - \mathbb{E}(\max_{i\in S} Q_i) \\
&= \mathbb{E}(\max(0, Q_j - \max_{i\in S} Q_i)) \\
&\leqslant \mathbb{E}(\max(0, Q_j - \max_{i\in T} Q_i)) \\
&= \mathbb{E}(\max_{i\in T\cup\{j\}} Q_i) - \mathbb{E}(\max_{i\in T} Q_i) \\
&= f(T \cup \{j\}) - f(T).
\end{aligned}$$

In the proof, we needed $\max_{i\in\emptyset} Q_i = -\infty$ in the case $T = \emptyset$. □

We note that the assertion that $\max_{i\in\emptyset} Q_i = -\infty$ corresponds to the fact that giving no answer to a query has $-\infty$ quality.

We can now prove Lemma 5.

Table 2: AUC scores in % for different strategies on RouterBench across model and noise levels. All baselines are always worse than the 95% confidence intervals of cascade routing. For a discussion on confidence intervals, we refer to App. F.

	Three Models			Five Models			Eleven Models		
	Low	Med	High	Low	Med	High	Low	Med	High
Linear Interp.	69.62	69.62	69.62	69.22	69.22	69.22	70.51	70.51	70.51
Routing	79.73	74.97	71.81	81.24	74.43	71.33	83.25	74.63	72.67
Cascade (Baseline)	80.86	74.64	72.48	82.33	73.03	69.53	84.48	73.64	69.79
Cascade (Ours)	81.09	76.16	72.67	83.06	75.17	70.18	84.47	75.10	70.26
Cascade Routing (Ours)	**82.36**	**76.55**	**73.22**	**84.33**	**76.31**	**72.75**	**87.24**	**77.57**	**74.40**

Proof. Let M and m be as in the lemma. Suppose M' is a supermodel that contains all models in M. Furthermore, let $M'' = M' \setminus m$. We show that the supermodel M'' is always strictly preferred over M'. To see this, we note that the difference between $\tau_{M'}(x, \lambda)$ and $\tau_{M''}(x, \lambda)$ is equal to

$$\mathbb{E}(\max_{m' \in M'} \hat{q}_{m'}(x)) - \mathbb{E}(\max_{m' \in M''} \hat{q}_{m'}(x)) - \lambda_j \hat{c}_m(x)$$

By Lemma 6, this difference is smaller than $\hat{q}_M(x) - \hat{q}_{M \setminus \{m\}}(x) - \lambda_j \hat{c}_m(x)$. Thus, by assumption, this difference is negative, and therefore M'' is always preferred over M', which concludes the proof. □

D Additional Experiments

D.1 RouterBench

RouterBench (Hu et al., 2024) is a benchmark developed to evaluate the efficacy of different model selection strategies. It includes questions from seven diverse benchmarks, such as MMLU (Hendrycks et al., 2021), GSM8k (Cobbe et al., 2021), and MBPP (Austin et al., 2021), alongside answers from eleven different models ranging from GPT-4 (OpenAI, 2023) to Mistral-7B (Jiang et al., 2023).

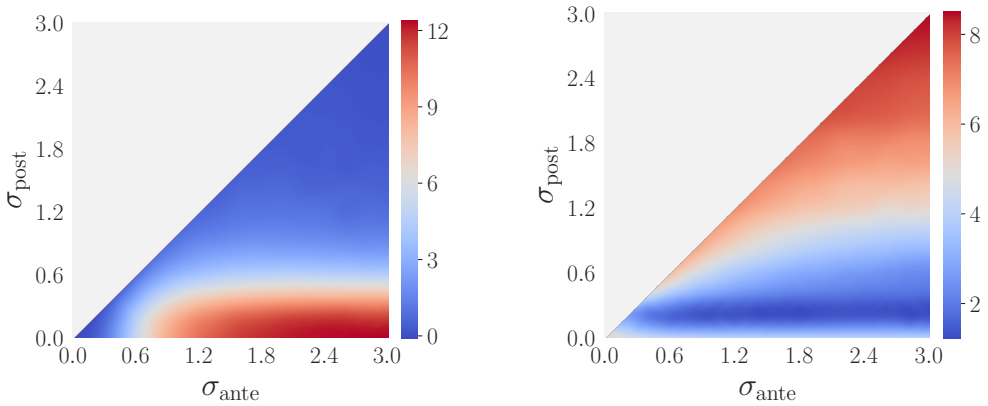
Quality and Cost Estimates Similar to (Hu et al., 2024), we estimate quality and cost by adding zero-centered Gaussian noise to their true values. Both cost and quality estimates are modeled as linear functions fitted on these noisy signals. Thus, the quality estimate can be expressed as $\hat{q}_{W,b}(x) = W(q(x) + \epsilon) + b$ where $\epsilon \sim \mathcal{N}(0, \sigma^2)$. A similar expression holds for the cost estimate. We define the variance of the noisy signal as σ^2_{ante} before model computation (ex-ante estimates) and σ^2_{post} after (post-hoc estimates). To explore different uncertainty levels, we vary the variances to simulate low-, medium-, and high-noise scenarios, with exact values for the variances given in App. E.3.

Models We evaluate cascade routing on RouterBench using three, five, and eleven models available for model selection, ensuring a comprehensive evaluation across a range of scenarios. The exact models are provided in App. E.3.

Strategies We compare cascade routing against several baseline strategies, including the routing strategy described in §2, the threshold-based cascading approach from prior work (Corollary 1), and the optimal cascading strategy (Theorem 2). Additionally, as in (Hu et al., 2024), we include a baseline that linearly interpolates cost and quality on the Pareto frontier of the models.

Evaluation Metric For each method, we evaluate performance using cost budgets ranging from the cheapest to the most expensive model. This produces a quality-cost curve for each strategy. Following (Hu et al., 2024), we use the Area Under the Curve (AUC) as the performance metric.

Results Table 2 presents the results for the zero-shot setting, with the five-shot results detailed in App. H. Cascade routing consistently outperforms all baseline strategies with performance gains



(a) Comparison of cascade routing with routing. (b) Comparison of cascade routing with cascading.

Figure 2: Difference in AUC performance between cascade routing and baseline strategies on RouterBench for various noise values. Red indicates cascade routing is much better, while blue indicates it is only a bit better.

between 1% to 4%, which measured relatively to the naive linear interpolation baseline means that cascade routing improves by 13% to 80% over the baselines. This performance gap widens as more models are available and narrows under higher noise levels, indicating that cascade routing is most effective with large model sets and accurate cost and quality estimates. Furthermore, our new cascading strategy outperforms the threshold-based cascade by up to 2%, reinforcing the practical relevance of our theoretical results.

Quality Estimation To better understand the impact of quality estimation on model selection strategies, we additionally conduct experiments with five models under a broader range of varying noise levels. Fig. 2 illustrates the difference in AUC performance between cascade routing and baseline strategies for all possible noise levels. The results demonstrate that cascade routing consistently outperforms the baselines, achieving up to an 8% improvement for cascading and up to a 12% improvement for routing. Notably, the performance gap highlights key differences between the cascading and routing strategies. For routing, the value of σ_{ante} is critical—high σ_{ante} significantly reduces performance compared to cascade routing. Conversely, for cascading, σ_{post} plays a more influential role, with higher values causing substantial performance degradation. These findings underscore the importance of accurate quality estimation for both strategies. Cascade routing proves to be a more robust solution by unifying the strengths of both approaches and effectively leveraging low σ_{ante} and low σ_{post} to enhance performance.

D.2 Ablation Study

We conduct an ablation study to examine the impact of various design choices in cascade routing on performance and runtime. Runtime is a critical factor because the overhead introduced by the strategy must be negligible compared to the time required for model computation. If the strategy adds significant overhead, its performance gains may be offset by the increased runtime. We also include an additional ablation that specifically targets runtime on random data in App. D.3.

To investigate this, we repeat the experiment from **??** when using all eleven models, testing different variations of cascade routing. We evaluate a slower variation that omits Lemma 5, thereby requiring more supermodels to be evaluated (Slow), a greedy variation that only considers supermodels of length $j + 1$ at step j (Greedy), and a version that does not compute the expected value when evaluating supermodel quality, using the quality of the best model instead (No-Expect).

Results Table 3 presents the results. As expected, the Slow variation is almost an order of magnitude slower while achieving similar performance. In contrast, both Greedy and No-Expect are faster but perform worse in the low- and medium-noise scenarios by 0.5% to 1.3%. Interestingly, there is a much smaller performance gap in the high-noise scenario. This is due to the very low variance in the quality estimates, since the linear model used for quality estimation predicts an almost constant value for each query in this scenario, making the expected value computation less important.

Table 3: AUC scores and average runtime for variations of cascade routing on RouterBench when using all eleven models.

	Low-Noise		Medium-Noise		High-Noise	
	AUC (%)	Time (ms)	AUC (%)	Time (ms)	AUC (%)	Time (ms)
Cascade Routing	87.29	15.26	77.61	9.53	74.41	13.68
SLOW	87.30	78.88	77.61	87.72	74.40	88.76
GREEDY	85.93	1.39	77.17	1.17	74.35	0.89
NO-EXPECT	85.98	4.78	77.11	2.49	74.35	2.08

Furthermore, the GREEDY and NO-EXPECT variants perform very similarly, while GREEDY is about twice as fast as NO-EXPECT. This suggests that one should almost always use the normal variant of cascade routing, and only consider the GREEDY variant if runtime is a critical concern. Neither the SLOW nor the NO-EXPECT variant is recommended, as they either perform worse or are significantly slower than the normal variant.

D.3 RUNTIME ANALYSIS

We further analyze the runtime of the four variants of cascade routing presented in App. D.2. Specifically, we perform experiments with random data, scaling the number of models to 80 to evaluate the runtime of all variants. Furthermore, we include a fifth variant of cascade routing in the analysis MAX-DEPTH, which restricts cascade routing to a maximum depth of 3 models. MAX-DEPTH does not reduce performance of cascade routing if the optimal depth is less than or equal to 3 models. However, it does significantly reduce the runtime of cascade routing.

For each number of models, we generate 100 data points, each with random quality and cost estimates associated with each model. For each point, we generate the hyperparameters $\lambda_1, ..., \lambda_k$ and γ randomly. We then report the average runtime of the five variants of cascade routing in Fig. 3.

The results show the varying computational complexity of the different variants of cascade routing. SLOW has the highest runtime, and becomes computationally too expensive even when using less than 20 models. In contrast, standard cascade routing has a significantly lower runtime, and is able to handle up to 40 models within a 1 second runtime. Its faster variant, MAX-DEPTH, is able to handle up to 80 models within a 1 second runtime. Furthermore, we now also see a clear difference between NO-EXPECT and GREEDY. While GREEDY remains computationally very cheap even for 80 models, NO-EXPECT has a significantly higher runtime, even obtaining higher runtimes than MAX-DEPTH for 80 models.

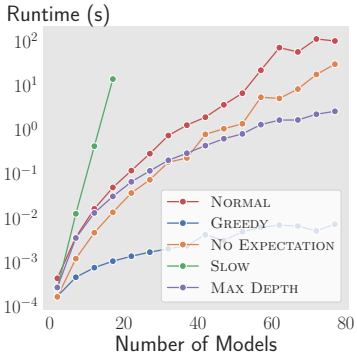


Figure 3: Runtime of cascade routing variants for different numbers of models.

Thus, the conclusions from App. D.2 are further supported by the runtime analysis: GREEDY is the most efficient variant of cascade routing, while NORMAL is the most efficient variant that does not compromise performance. MAX-DEPTH is a good choice if the optimal depth is known to be less than or equal to 3 models, as it significantly reduces runtime without compromising performance. Since cascades of more than 3 models are rare, MAX-DEPTH is a good choice in practice.

E EXPERIMENTAL DETAILS

We describe some additional details about the experimental setup and the datasets used in our experiments.

E.1 ACCURATE QUALITY ESTIMATION

Data Split For the SWE-Bench benchmark, we use its verified data split and divide the dataset into training and calibration subsets, with each comprising 50% of the data. For the Minerva Math and

LiveCodeBench benchmark, we only include the Algebra portion of Minerva Math to ensure that both benchmarks have a comparable number of samples for evaluation. Similarly, we also perform a 50% split of this dataset into training and calibration sets.

Evaluation Setting For the SWE-Bench evaluation, we analyze the performance of 10 models submitted to the benchmark's leaderboard. The logs for these models were obtained from the official SWE-Bench repository[3]. Specifically, we evaluated the following models:

- `20240402_sweagent_claude3opus`
- `20241007_nfactorial`
- `20240728_sweagent_gpt4o`
- `20240620_sweagent_claude3.5sonnet`
- `20241016_epam-ai-run-gpt-4o`
- `20240824_gru`
- `20241106_navie-2-gpt4o-sonnet`
- `20240820_epam-ai-run-gpt-4o`
- `20241202_agentless-1.5_claude-3.5-sonnet-20241022`
- `20241028_agentless-1.5_gpt4o`

For each model, we extract the time required to complete a task to measure cost.

For LiveCodeBench and Minerva Math, we evaluate the following models:

- QWEN-2.5-CODER-7B-INSTRUCT
- QWEN-2.5-CODER-1.5B-INSTRUCT
- QWEN-2.5-MATH-7B-INSTRUCT
- QWEN-2.5-MATH-1.5B-INSTRUCT

We conduct experiments using version 5 of the LiveCodeBench benchmark from its official repository. For Minerva Math, we utilize the LM Evaluation Harness (Gao et al., 2024) to ensure consistent and reliable evaluation.

Cost Estimation For SWE-Bench, the cost is defined as the time (in seconds) that a model takes to complete a task. A linear regression model is fitted to predict this cost based on the query length and, when available, the cost of running other models.

For LiveCodeBench and Minerva Math, the cost is calculated as the total number of tokens in both the query and the answer, multiplied by the size of the model (in billions of parameters). Similar to SWE-Bench, a linear model is used to predict the cost based on query length and other models' costs.

Quality Estimation For ex-ante quality estimation in SWE-Bench, we train a logistic regression model that predicts quality based on the query length and a one-hot encoded variable representing the query's source repository. Post-hoc quality estimation leverages the ground-truth quality scores computed during evaluation.

For ex-ante quality estimation in Minerva Math and LiveCodeBench, we include the query length, query source (Minerva Math or LiveCodeBench), and the difficulty level of the problem as defined by the benchmark. Post-hoc quality estimation incorporates additional information, such as whether the parsed answers from different models agree with one another.

E.2 POOR QUALITY ESTIMATION

Data Split We split each dataset in each benchmark into a training set and a test set, each comprising 50% of the data. For all datasets except GSM8k, the training set is created by splitting the original test data. In the case of GSM8k, since a separate training set is already available, we use this pre-existing training data, leaving the original test set unchanged. The training set is then further divided, with 50% used for training quality and cost estimators, and the remaining 50% reserved for hyperparameter optimization through validation.

[3] `https://github.com/swe-bench/experiments`

Table 4: Standard deviations of the noise levels on the RouterBench dataset.

	Quality		Cost	
	σ_{before}	σ_{after}	σ_{before}	σ_{after}
LOW	0.6	0.3	0.0002	0.00005
MEDIUM	1.6	0.8	0.0004	0.0001
HIGH	2.4	1.2	100	100

Evaluation Setting We use completion-based evaluation in a one-shot setting for each benchmark. For the classification tasks, we obtain the probability associated with each class ("A", "B", "C", ...) from the model directly. For open-form reasoning tasks, we extract the answer by instructing the model to generate a completion that ends with an extractable answer. If the model does not output an answer in the correct format, we perform a best-effort extraction by trying various regex patterns. Details on the prompts and regex patterns used for each benchmark are provided in the code.

Models For the LLAMA-3.1 model family (AI@Meta, 2024), we use the models LLAMA-3.1-8B-INSTRUCT, LLAMA-3.1-70B-INSTRUCT, and LLAMA-3.1-405B-INSTRUCT. For the GEMMA model family (Gemma Team et al., 2024), we use the models GEMMA-2B-INSTRUCT, GEMMA-2-9B-INSTRUCT, and GEMMA-2-27B-INSTRUCT. For the MISTRAL model family (Jiang et al., 2023), we use the models MISTRAL-7B-INSTRUCT-V0.3, MIXTRAL-8X7B-INSTRUCT-V0.1, and MIXTRAL-8X22B-INSTRUCT-V0.1.

Cost Estimation For cost estimation, we first calculate the number of tokens in both the query and the model's response. We then use API-based prices per token for each model to estimate the cost.[4] In classification, where responses consist of a single token, the cost can be determined before running the model. In open-form reasoning tasks, where response lengths vary, we estimate this length based on responses from previous models in the cascade if the model has not yet been computed. If no model response is available, we estimate the response length using the average from the training data.

Features Quality Estimates We specify the exact features used for the logistic regression model that serves as the quality estimator. First, we include a one-hot encoding of the various datasets in each benchmark. Furthermore, for classification, we include the probability associated with the highest class and the entropy of the class probabilities if the model has been computed. If several models have been computed, we include both whether they agree on their prediction, and the JS-divergence between their class probabilities. For open-form reasoning, we include the perplexity, number of tokens, and several quantiles of the logits if the model has been computed, in accordance with Gupta et al. (2024). If several models have been computed, we also include whether they agree on their prediction.

We note that we train a separate logistic regression model for each history of computed models, and for each model separately as well. Thus we have one linear model for each combination of a target model m_i and computed models $m_{i_1}, \ldots, m_{i_j}$. All the linear models are trained on the training set included in the benchmark.

E.3 ROUTERBENCH

Data Split We use 5% of the RouterBench data (around 2000 samples) to optimize the hyperparameters of cascading, routing, and cascade routing. The remaining 95% is used for evaluation. We use the same data split for all noise levels.

Noise In Table 4 we specify the standard deviations of the noise levels on the RouterBench dataset. To put these numbers into context, we note that quality varies between 0 and 1, and the average cost of the smallest models is 0.000073, while the average cost of the largest models is 0.003281. We fit a logistic regression model on this noisy signal to obtain the quality and cost estimates. This simulates the noise in the features that are used to estimate the quality and cost of the models.

[4]We used the Together API for all our experiments.

Table 5: AUC scores in % for different strategies on RouterBench in the 0-shot setting with 2σ confidence intervals.

	Three Models			Five Models			Eleven Models		
	Low	Med	High	Low	Med	High	Low	Med	High
Cascade Routing (Ours)	$82.37^{+0.31}_{-0.32}$	$76.57^{+0.34}_{-0.35}$	$73.23^{+0.37}_{-0.38}$	$84.33^{+0.29}_{-0.29}$	$76.32^{+0.34}_{-0.37}$	$72.75^{+0.36}_{-0.40}$	$87.24^{+0.23}_{-0.26}$	$77.58^{+0.30}_{-0.33}$	$74.41^{+0.33}_{-0.36}$
– Routing	$2.64^{+0.15}_{-0.16}$	$1.59^{+0.13}_{-0.15}$	$1.40^{+0.15}_{-0.17}$	$3.10^{+0.17}_{-0.15}$	$1.88^{+0.17}_{-0.16}$	$1.41^{+0.17}_{-0.17}$	$4.00^{+0.17}_{-0.21}$	$2.94^{+0.20}_{-0.21}$	$1.73^{+0.19}_{-0.19}$
– Cascade (Baseline)	$1.50^{+0.12}_{-0.12}$	$1.91^{+0.18}_{-0.19}$	$0.74^{+0.19}_{-0.18}$	$2.00^{+0.17}_{-0.15}$	$3.29^{+0.26}_{-0.27}$	$3.22^{+0.24}_{-0.24}$	$2.76^{+0.14}_{-0.14}$	$3.92^{+0.28}_{-0.28}$	$4.61^{+0.28}_{-0.28}$
– Cascade (Ours)	$1.28^{+0.12}_{-0.11}$	$0.39^{+0.15}_{-0.14}$	$0.54^{+0.18}_{-0.17}$	$1.27^{+0.10}_{-0.11}$	$1.14^{+0.20}_{-0.21}$	$2.57^{+0.21}_{-0.26}$	$2.77^{+0.13}_{-0.13}$	$2.46^{+0.22}_{-0.24}$	$4.14^{+0.25}_{-0.27}$

Table 6: AUC scores in % for different strategies on RouterBench in the 5-shot setting across model and noise levels with 2σ confidence intervals. Bold numbers indicate that the confidence interval contains zero.

	Three Models			Five Models			Eleven Models		
	Low	Med	High	Low	Med	High	Low	Med	High
Cascade Routing (Ours) $83.79^{+0.3}_{-0.33}$	$78.85^{+0.29}_{-0.34}$	$77.1^{+0.32}_{-0.35}$	$85.5^{+0.26}_{-0.26}$	$78.77^{+0.32}_{-0.33}$	$76.74^{+0.34}_{-0.36}$	$88.78^{+0.22}_{-0.22}$	$80.89^{+0.28}_{-0.29}$	$78.03^{+0.3}_{-0.31}$	
– Routing	$2.3^{+0.14}_{-0.13}$	$1.64^{+0.15}_{-0.15}$	$1.1^{+0.13}_{-0.14}$	$3.08^{+0.16}_{-0.14}$	$1.94^{+0.16}_{-0.14}$	$1.21^{+0.14}_{-0.15}$	$3.43^{+0.15}_{-0.17}$	$3.13^{+0.19}_{-0.2}$	$1.6^{+0.17}_{-0.16}$
– Cascade (Baseline)	$-0.64^{+0.11}_{-0.1}$	$0.28^{+0.13}_{-0.14}$	$0.22^{+0.16}_{-0.16}$	$1.23^{+0.12}_{-0.12}$	$2.19^{+0.21}_{-0.21}$	$2.83^{+0.24}_{-0.24}$	$1.64^{+0.12}_{-0.13}$	$2.29^{+0.24}_{-0.24}$	$3.09^{+0.27}_{-0.26}$
– Cascade (Ours)	$1.02^{+0.1}_{-0.09}$	$\mathbf{0.09^{+0.11}_{-0.11}}$	$\mathbf{0.1^{+0.14}_{-0.14}}$	$1.25^{+0.1}_{-0.09}$	$1.59^{+0.17}_{-0.17}$	$2.45^{+0.21}_{-0.21}$	$2.06^{+0.1}_{-0.1}$	$2.22^{+0.21}_{-0.19}$	$2.95^{+0.23}_{-0.24}$

Models In the evaluated scenarios for three models, we use the models MIXTRAL-8X7B-CHAT, GPT-3.5-TURBO-1106, and GPT-4-1106-PREVIEW. When using five models, we add WIZARDLM-13B-V1.2 and CLAUDE-V2 to the mix. For eleven models, we use all models available in the benchmark.

F CONFIDENCE INTERVALS

To check whether the results obtained by cascade routing are significantly higher than our baselines in Tables 1, 2 and 10, we perform bootstrapping on the samples in the dataset. Specifically, we compute the confidence interval associated with the difference between the AUC scores of cascade routing and the baselines. If this difference is positive and its 2σ confidence interval does not contain zero, we can conclude that cascade routing is significantly better than the baseline. These confidence intervals are reported in Tables 5–7.

G DETAILED RESULTS

We present benchmark-specific AUC values for the experiment performed in §5 in Table 8 for classification and Table 9 for open-form reasoning.

H ADDITIONAL RESULTS

In Table 10 we report the AUC scores for the RouterBench dataset for different noise levels for the five-shot evaluation. Our conclusions presented in App. D.1 remain consistent with the results presented in Table 10. However, there is one notable inconsistency: in two of the three low-noise scenarios, our cascading strategy performs worse than the threshold-based baseline cascade. In the scenario with three models, we find its cause can be found in the more difficult optimization surface for the hyperparameters of our cascading strategy. Specifically, our cascading strategy at some point starts to lose quality as cost increases. By simply setting the hyperparameters of the cascading strategy once it starts to lose quality to the ones where it obtained its highest quality, we obtain a quality of 83.35% over the 83.17% of the baseline cascade.

In contrast, for low-noise and eleven models, a similar approach does not yield a better result. Rather, the discrepancy is caused by a small mismatch between the quality estimates of supermodels and the chosen model. While the quality estimate is based on the expected maximum of all models, we restrict the selected model to be the last model that was computed in the cascade. Since the expected maximum is higher than the quality of the last model, this discrepancy can lead to suboptimal decisions. By allowing both the baseline cascade and our cascading strategy to select the model with the highest quality estimate, we find that our cascading strategy once again outperforms the baseline cascade. Note that this slight discrepancy is not relevant for cascade routing, since the extra restriction is not imposed in this setting.

Table 7: AUC scores on the realistic benchmarks with 2σ confidence intervals. Bold numbers indicate that the confidence interval contains zero.

	SWE-Bench		Math+Code	Classification			Open-Form		
	10 MODELS	5 MODELS	QWEN	LLAMA	GEMMA	MISTRAL	LLAMA	GEMMA	MISTRAL
Cascade Routing (Ours)	$54.21^{+7.49}_{-7.17}$	$51.20^{+7.44}_{-7.22}$	$48.51^{+2.95}_{-2.99}$	$75.56^{+1.22}_{-1.16}$	$64.89^{+1.36}_{-1.42}$	$65.02^{+1.40}_{-1.24}$	$79.95^{+1.28}_{-1.33}$	$59.70^{+1.62}_{-1.61}$	$58.77^{+1.42}_{-1.50}$
– Routing	$13.65^{+4.50}_{-4.32}$	$11.71^{+4.39}_{-4.14}$	$1.11^{+0.81}_{-0.80}$	$0.60^{+0.32}_{-0.34}$	$\mathbf{0.39^{+0.50}_{-0.47}}$	$\mathbf{0.08^{+0.12}_{-0.10}}$	$0.56^{+0.38}_{-0.42}$	$1.26^{+0.57}_{-0.55}$	$\mathbf{0.02^{+0.05}_{-0.05}}$
– Cascade (Baseline)	$15.36^{+3.90}_{-3.22}$	$5.17^{+1.69}_{-1.63}$	$10.77^{+1.71}_{-1.72}$	$0.71^{+0.30}_{-0.28}$	$10.51^{+0.62}_{-0.61}$	$3.79^{+0.73}_{-0.77}$	$0.65^{+0.23}_{-0.27}$	$3.47^{+0.37}_{-0.40}$	$10.45^{+1.15}_{-1.06}$
– Cascade (Ours)	$\mathbf{0.83^{+1.90}_{-1.50}}$	$\mathbf{0.11^{+1.05}_{-1.04}}$	$1.82^{+0.58}_{-0.55}$	$\mathbf{0.06^{+0.15}_{-0.17}}$	$2.04^{+0.33}_{-0.31}$	$1.67^{+0.41}_{-0.42}$	$0.20^{+0.19}_{-0.19}$	$2.00^{+0.25}_{-0.25}$	$3.06^{+0.65}_{-0.63}$

Table 8: Classification AUC values for each benchmark separately for the experiment performed in §5.

	LLAMA			GEMMA			MISTRAL		
	MMLU	ARC	MixEval	MMLU	ARC	MixEval	MMLU	ARC	MixEval
Linear Interp.	53.82	93.15	82.86	39.40	82.28	70.97	39.76	85.39	73.03
Routing	55.32	93.12	82.86	40.01	83.13	73.12	40.61	85.64	74.28
Cascade (Baseline)	54.80	94.08	84.15	36.43	77.53	66.10	36.99	83.88	72.73
Cascade (Ours)	55.05	94.16	84.00	37.68	79.80	70.57	37.03	86.27	74.42
Cascade Routing (Ours)	55.40	93.90	83.91	39.93	83.74	73.16	40.56	86.52	74.64

Table 9: Open-form AUC values for each benchmark separately for the experiment performed in §5.

	LLAMA		GEMMA		MISTRAL	
	MMLU	GSM8k	MMLU	GSM8k	MMLU	GSM8k
Linear Interp.	65.64	94.43	36.52	73.86	41.40	67.84
Routing	65.75	94.15	38.08	75.01	43.03	68.00
Cascade (Baseline)	66.07	95.17	35.76	68.44	38.88	60.82
Cascade (Ours)	66.25	94.94	38.16	71.10	40.76	64.53
Cascade Routing (Ours)	66.60	94.69	40.43	75.25	42.93	68.30

Table 10: AUC scores in % for different strategies on RouterBench across model and noise levels for five-shot evaluation. Highest numbers are bolded, underlined numbers are within the 95% confidence intervals of the highest number. For a discussion on confidence intervals, we refer to App. F.

	Three Models			Five Models			Eleven Models		
	Low	Med	High	Low	Med	High	Low	Med	High
Linear Interp.	74.21	74.21	74.21	73.82	73.82	73.82	75.16	75.16	75.16
Routing	81.50	77.22	76.01	82.43	76.84	75.54	85.34	77.77	76.44
Cascade (Baseline)	83.16	78.58	76.89	84.27	76.59	73.92	87.14	78.60	74.94
Cascade (Ours)	82.78	78.77	77.01	84.26	77.19	74.30	86.72	78.67	75.08
Cascade Routing (Ours)	**83.80**	**78.86**	**77.11**	**85.50**	**78.78**	**76.75**	**88.78**	**80.90**	**78.04**

Table 11: AUC scores on several benchmarks for the MISTRAL model family. Highest numbers are bolded, underlined numbers are within the 95% confidence intervals of the highest number. For confidence intervals, see App. F.

	Classification	Open-Form
Linear Interp.	63.39	53.86
Routing	64.89	58.71
Cascade (Baseline)	61.20	48.29
Cascade (Ours)	63.31	55.51
Cascade Routing (Ours)	**64.97**	**58.73**