HARISSA: INFERENCE-TIME SELF-CHECKS FOR EFFICIENT AND SAFE LOCAL LANGUAGE MODEL DEPLOYMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Running a language model locally offers advantages in privacy, latency, and cost, but local hardware fits only small models, which are less capable than frontier models. The usual remedy for a hard query, escalating it to a cloud model, gives up the privacy and cost advantages of running locally. A deployment that stays local faces two decisions for hard queries instead. First, it can spend more computation on a query, e.g., reasoning before answering, which raises accuracy at a cost in latency, so it must decide which queries are worth the extra computation (efficiency). Second, some queries are beyond the local model, and delivering a wrong answer is worse than deferring the query to a human in the loop, so it must decide which answers are safe to deliver (safety). We show that both decisions can be made from the model's own hidden states. The prefill state, computed before any token is generated, predicts whether the model will answer correctly, and the answer state, at the end of the generated answer, predicts whether that answer is correct. HARISSA fine-tunes the model so that both states predict correctness, then makes both decisions with one policy that cascades through the ways of answering from cheapest to most expensive, skipping a way the prefill state predicts will fail and deferring the query when the answer it stops with is predicted wrong. On a device running a single model, HARISSA is within one accuracy point of chain-of-thought at 2.7 times lower latency. On a server holding four sizes of one model, HARISSA is more accurate than the FrugalGPT and Self-REF cascades at the same latency, and at the same deferral rate the answer state leaves fewer wrong answers than the standard confidence signals in five of six task and setting pairs.

1 INTRODUCTION

Running a language model locally, for example on a laptop or a phone, avoids the per-token price of a frontier model's cloud API, $50 per million output tokens.[1] A local model also answers with no network latency, keeps working offline (Xu et al., 2024), and uses less energy per query than a larger model (Luccioni et al., 2024). A local-only deployment also shares no data with a third party, which any user may value and which some domains require.

Clinical records contain health data, which European data-protection law restricts from transfer outside the Union (European Parliament and Council, 2016) and which locally deployed models can process in place (Wiest et al., 2024; Garg et al., 2026). Legal documents contain client confidences, which may not enter a self-learning generative AI tool without the client's informed consent (American Bar Association Standing Committee on Ethics and Professional Responsibility, 2024). Financial institutions train and run models on data they hold in-house (Wu et al., 2023). A frontier model in the cloud can be reached under a vendor contract that covers such data, such as a business associate agreement for health data (OpenAI, 2026). However, a contract requires legal review, an IT team, and a per-query budget, which many organizations cannot supply. Among hospitals, for example, rural and independent ones adopt AI at lower rates (Nong et al., 2025), and the rural-health literature names infrastructure, staff, and funding as the barriers (Brown & Davis, 2026).

[1] Prices as of September 2026: `https://platform.claude.com/docs/en/about-claude/pricing` and `https://developers.openai.com/api/docs/pricing`.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Consumer hardware such as a laptop or a phone runs only a small language model, currently a few billion parameters (Lu et al., 2024; Xu et al., 2024), and the queries a small model answers wrong are the complex ones that need specialized knowledge or multi-step reasoning (Ding et al., 2024; Ong et al., 2025). The standard remedy escalates such a query to the cloud model, either by a router that decides from the query which model answers (Ong et al., 2025; Ding et al., 2024) or by a cascade that scores the small model's answer and escalates when the score is low (Chen et al., 2024; Yue et al., 2024). For a local-only deployment there is no cloud model to escalate to. A local-only deployment can instead spend more of its own computation on the query (Snell et al., 2025), by reasoning before answering (Wei et al., 2022), by sampling several answers and taking the majority (Wang et al., 2023), by retrieving from a local corpus (Asai et al., 2024; Jeong et al., 2024), or, on a server, by running a larger size of the model. A local-only deployment can also defer the query to the person (Madras et al., 2018; Mozannar & Sontag, 2020; Fanconi & van der Schaar, 2025). However, more computation raises accuracy at a cost in latency. A 4B model that reasons before answering takes over a minute per query on a data-center GPU (81 s per MedQA query in our experiments, Section 5.1).

A wait of over a minute is prohibitive for a clinician consulting the model between patient visits, a customer-service agent answering on a live call, or a voice assistant on a phone, yet reasoning often raises a small model's accuracy on the queries it would otherwise answer wrong, and several recent small models enable it by default (Qwen Team, 2026a; Granite Team, IBM, 2026; NVIDIA, 2025; Hugging Face, 2025). A local-only deployment should therefore answer directly, in a few seconds, and reason only on the queries that need it. Other queries are beyond the model however it answers, and for those a wrong answer delivered is worse than a query deferred, so the deployment should recognize them and defer them to the person. Deciding which queries receive more computation and which are deferred falls to the local model itself, because no larger model is available to judge the query, and a wrong decision either wastes computation or delivers a wrong answer. The signal for each decision is in the model's own hidden states. When the model processes the prompt in one forward pass, the prefill, the hidden state at the last prompt token predicts whether the model will answer correctly (Kadavath et al., 2022; Lugoloobi et al., 2026; Liang et al., 2025; Liu et al., 2026), and the hidden state at the last generated token predicts whether the generated answer is correct (Azaria & Mitchell, 2023; Orgad et al., 2025).

We introduce HARISSA (Hidden-Activation Reads for Inference-time Safety and Self-Assessment), a policy that reads the prefill state and the answer state to decide how much computation a query receives and whether its answer is delivered or deferred to the person. HARISSA fine-tunes the model so that each state predicts whether the model's answer is correct, then cascades through the ways of answering from cheapest to most expensive, skipping a way the prefill state predicts will fail and delivering the first answer the answer state predicts correct. As a result, computation goes to the queries that need it, and a query beyond the model reaches the person instead of receiving a wrong answer. The model computes both states in the course of answering, so the decisions add no second model and no generated tokens. Our contributions are threefold. First, on a device running a single model, HARISSA is within one accuracy point of chain-of-thought at 2.7 times lower latency, averaged over three tasks, because the policy reasons only on the queries whose direct answer the answer probe predicts wrong (Section 5.1). Second, on a server holding four sizes of one model, HARISSA is more accurate than FrugalGPT and Self-REF at the same latency, because the prefill probe skips the sizes predicted to fail before anything is generated (Section 5.2). Third, deferring by the answer probability leaves fewer wrong answers than deferring by sequence likelihood, the logit margin, or Self-REF confidence at the same deferral rate, on five of six task and setting pairs (Section 5.3). All code and data are in the supplementary material and will be released on publication.

2 Related Work

Routing and cascading to a larger model. Most work on efficient inference chooses which of several models answers a query under a cost target, with a router or with a cascade (Wang et al., 2025). A router picks the model from the query before any model runs. RouteLLM trains one on human preference data to predict which of a strong and a weak model answers better (Ong et al., 2025), and Hybrid LLM trains one to predict the quality gap between a small and a large model (Ding et al., 2024). A cascade runs the small model first and escalates on a low score. FrugalGPT scores the generated answer with a trained classifier (Chen et al., 2024), and Gupta et al. (2024) learn the escalation rule from the small model's token-level uncertainty. Confident or Seek Stronger escalates

from an on-device model to a cloud model when the on-device answer is uncertain (Chuang et al., 2025b), and Fanconi & van der Schaar (2025) add a human expert after the largest model for the queries every model answered with low confidence. All of these methods rely on a larger model to answer the queries the small model cannot. In a local-only deployment the largest model is still small and may fail, so escalation alone pays the latency of every size and still delivers a wrong answer. HARISSA can skip a size predicted to fail without generating its answer, and defers the query to the person when the answer it stops with is predicted wrong.

Deciding how much computation one model spends, and whether to answer. Other methods use one model and decide per query how much computation to spend. Sampling several answers and taking the majority raises accuracy at a multiple of the cost (Wang et al., 2023), and spending that computation only on the queries predicted to need it reaches the same accuracy at a fraction of the cost (Snell et al., 2025). ThinkSwitcher decides from the prefill state whether to reason before answering (Liang et al., 2025), DiffAdapt predicts difficulty from the same state and picks a reasoning strategy to match (Liu et al., 2026), and AdaptThink retrains the model to choose whether to reason (Zhang et al., 2025). The same kind of estimate also decides whether the model answers at all. Selective prediction withholds the least confident answers and trades coverage for accuracy (El-Yaniv & Wiener, 2010; Geifman & El-Yaniv, 2017), and learning to defer trains a classifier together with a rule that hands a case to a human expert at a cost per deferral (Madras et al., 2018; Mozannar & Sontag, 2020). Each of these methods makes one of the choices a local-only deployment faces, how much to compute or whether to answer, and tunes each on its own. HARISSA makes both decisions with one policy under one objective that prices latency and deferral.

Estimating correctness from a model's own hidden states. Kadavath et al. (2022) show that a language model can be trained to predict from the prompt alone whether it knows the answer. Lugoloobi et al. (2026) show that a linear probe on the prefill state predicts success on mathematics and coding. IntroLM fine-tunes an adapter so that the model predicts during the prefill whether it will answer correctly (Kasnavieh et al., 2026). After generation, Azaria & Mitchell (2023) classify the truthfulness of a statement from its hidden states, and Orgad et al. (2025) find the signal concentrated in the answer tokens and show that such probes transfer poorly across datasets. Self-REF fine-tunes the model to emit a confidence token after its answer (Chuang et al., 2025a), and CoKE trains the model to decline the questions its own confidence marks as beyond it (Chen et al., 2025). An estimate before generation lets a deployment skip an answer that would fail, and an estimate after generation lets the deployment withhold a wrong answer before delivery. Each of these methods provides one estimate or the other. HARISSA fine-tunes one model so that both the prefill state and the answer state predict correctness, and one policy acts on both.

3 METHOD

HARISSA is a policy that cascades through a local deployment's ways of answering a query, from cheapest to most expensive, on a model fine-tuned to predict its own correctness. Before a way of answering generates, a probe on the prefill state decides whether that way is worth running. After a way of answering has generated an answer, a probe on the answer state decides whether to deliver the answer, try the next way, or defer to the person. Fine-tuning the model and fitting the two probes happen once, offline. At inference the policy applies a threshold at each decision, whether to skip a way of answering, whether to stop with an answer or try the next way, and whether to deliver the answer or defer the query.

Problem setting. We consider a deployment that can answer a query with any of K *actions*, where an action is a procedure that takes the query and produces an answer at a measured latency. On a device the actions are decoding strategies of one model: direct (thinking off) and think (chain-of-thought reasoning before the answer). On a server the actions are the sizes of one model family: 2B, 4B, 9B, and 27B. Actions are indexed $k = 1, \ldots, K$ from cheapest to most expensive. A person is present and receives any query the deployment does not answer.

A policy for this deployment takes a query x, generates with one or more actions in cost order, and ends by delivering one answer or by deferring the query to the person. Let $e(x) = 1$ if the policy

delivers a wrong answer, $d(x) = 1$ if it defers, and $T(x)$ be its latency on x. The policy is judged by

$$\mathcal{R} \;=\; \mathbb{E}[e(x)] \;+\; \lambda\,\mathbb{E}[T(x)] \;+\; \mu\,\mathbb{E}[d(x)], \tag{1}$$

where λ is the cost of a second of latency and μ the cost of a deferral, both in units of one delivered wrong answer; this is the objective of Fanconi & van der Schaar (2025) and Zellinger et al. (2025).

The *prefill state* is the hidden state at the last prompt token after the forward pass over the prompt, so it exists before any token is generated. The *answer state* is the hidden state at the last token of the generated answer.

Fine-tuning. Fine-tuning teaches the model to predict, in its own hidden states, whether each action will answer a query correctly and whether a generated answer is correct. The labels come from the deployment's own training split. Every action answers every training query x, each answer is graded, and $y_k(x) \in \{0, 1\}$ records whether action k answered x correctly. We fine-tune the model with LoRA so that its hidden states predict these labels, with linear heads on the two states supplying the training signal. A prefill head for each action k predicts $y_k(x)$ from the prefill state, so the model learns to encode, before generating, which actions will succeed on the query. One answer head predicts $y_k(x)$ from the answer state of action k's answer, so the model learns to encode, after generating, whether the answer it wrote is right. The heads are trained together under a binary cross-entropy loss at equal weight.

Linear probes. A probe is a logistic regression on the hidden state at one layer of the fine-tuned model, calibrated by isotonic regression on the development split (Zadrozny & Elkan, 2002), and gives the probability that an answer is correct. For action k, the prefill probe reads the prefill state and gives the *prefill probability* $p_k(x)$, the estimated probability that action k would answer x correctly; the answer probe reads the answer state and gives the *answer probability* $q_k(x)$, the estimated probability that the answer generated by action k is correct. A probe costs one dot product and adds no measurable latency. Hyperparameters and the probed layer are in Appendix B.

The HARISSA policy. The cascade runs as in Algorithm 1. For each action k but the last, the policy first applies the *prefill skip*: if $p_k(x)$ is below the *skip threshold* τ, it skips the action without generating. Otherwise it generates the action's answer and applies the *answer check*: if $q_k(x)$ is at least the *fall-through threshold* τ_f, it stops with that answer, and otherwise it discards the answer and moves to action $k + 1$. If the policy reaches the last action, it generates that action's answer and stops with it, since no action remains to try. The policy then delivers the answer it stopped with, together with its answer probability, unless that probability is below the *deferral threshold* τ_d, in which case it withholds the answer and defers the query to the person. One value of τ and one of τ_f apply to every action.

```
for k = 1, . . . , K-1 (cheapest first):
    if p_k(x) < τ: continue
    generate action k's answer
    if q_k(x) ≥ τ_f: stop with this answer
if no answer yet: generate action K's answer
deliver the answer if q ≥ τ_d, else defer
```

Algorithm 1: The HARISSA policy. Actions are ordered by cost; p_k and q_k are the prefill and answer probabilities of action k; τ is the skip threshold, τ_f the fall-through threshold, and τ_d the deferral threshold.

Thresholds. The skip and fall-through thresholds set how the cascade trades latency, wrong answers, and deferrals. The skip threshold sets how much latency the cascade spends: at $\tau = 0$ the policy generates with the cheapest action on every query, as τ rises it skips more actions, and above 1 it runs only the most expensive one. Sweeping τ traces an accuracy–latency curve. The skip and fall-through thresholds are chosen on the development split by minimizing $\mathcal{R}$. We set the deferral threshold equal to the fall-through threshold, so the answer probability that discards an answer at a non-last action also withholds it at the last.

4 Experimental Setup

Datasets and models. We evaluate on three multiple-choice benchmarks: MedQA (Jin et al., 2021), USMLE-style clinical questions; MedMCQA (Pal et al., 2022), medical-entrance questions across subjects; and BBH (Suzgun et al., 2023), general reasoning, restricted to its sixteen multiple-choice

subtasks. Each task has disjoint training, development, and test splits; split sizes, the BBH subtask list, prompt templates, and decoding settings are in Appendix A. We run every policy in two local-only settings, a device running a single model and a server holding four sizes of one model. On the device we use Qwen 3.5 4B (Qwen Team, 2026b; 2025) with its two actions, direct and think. On the server the actions are the Qwen 3.5 2B, 4B, 9B, and 27B sizes, each answering with thinking off. The device experiments are replicated on two other model families, Gemma 4 (Gemma Team, 2026) and Cogito, in Appendix E.

Baselines. On the device, where a single model answers, we compare HARISSA against three baselines that answer every query the same way and against two cascades from prior work. The baselines are direct (thinking off), chain-of-thought (thinking on), and self-consistency, the majority of three sampled direct answers (Wang et al., 2023). The cascades are FrugalGPT and Self-REF. FrugalGPT (Chen et al., 2024) grades each generated answer with a trained DistilBERT scorer and escalates when the score is below a threshold. Self-REF (Chuang et al., 2025a) fine-tunes the model to emit a confidence token after its answer and escalates on that token's probability. On the server we compare against the same two cascades. Deferral has its own baselines. A deferral signal ranks the answers a policy delivers, and the lowest-ranked share goes to the person, so at the same share deferred the better signal is the one that removes more of the wrong answers. We compare the answer probability against three standard confidence signals, each computed on the same answers HARISSA delivers: sequence likelihood (Malinin & Gales, 2021), the logit margin (the gap between the top two option logits at the answer, a variant of the maximum-softmax baseline of Hendrycks & Gimpel, 2017; Geifman & El-Yaniv, 2017), and Self-REF confidence, the probability of the confidence token that the Self-REF model of the same size emits when given the query and HARISSA's answer.

Metrics and protocol. We report accuracy when every query is answered, and, with deferral, wrong answers delivered and the share of queries deferred, each as a percentage of all queries, and mean latency per query in seconds, computed from each query's token counts at rates measured on one NVIDIA A40 (Appendix C). The model and the probes are fit on the training split, calibration and the thresholds are chosen on the development split, and every number we report is on the test split. The skip and fall-through thresholds are chosen jointly by the objective (Eq. 1), the fall-through threshold over 0.3 to 0.7, with λ set so that one accuracy point is worth 10 s of latency per query on the device and 0.5 s on the server. Deferral is off except in Section 5.3, so that the policies are compared on the same queries at full coverage.

5 Results

HARISSA is more accurate than FrugalGPT and Self-REF at the same latency on the device and on the server (Figures 1 and 2, top; Table 1). The answer probability on each delivered answer is also the deferral signal, and it leaves fewer wrong answers than the standard confidence signals at the same deferral rate (Figures 1 and 2, bottom).

5.1 Single model on device

HARISSA is within one accuracy point of chain-of-thought at 2.7 times lower latency. Chain-of-thought is 7.4 points more accurate than direct on MedQA at about eight times its latency (Table 1), so a policy that thinks only on the queries direct gets wrong could keep most of that gain at far lower latency. Averaged over the three tasks, HARISSA is 0.8 points below chain-of-thought at 2.7 times lower latency. On MedQA it answers 85.7% of queries correctly at 33.0 s against chain-of-thought's 86.2% at 81.0 s. The gap is widest on BBH, where HARISSA is 1.9 points below chain-of-thought at 3.8 times lower latency. HARISSA is more accurate than self-consistency on every task.

HARISSA reaches FrugalGPT's accuracy on MedQA by thinking on 29% of queries where FrugalGPT thinks on 75%. Every cascade on the device generates the direct answer first and thinks only on the queries whose direct answer it does not trust, so its latency rises with the share of queries it sends to think, and the better cascade reaches a given accuracy by thinking on fewer queries. On MedQA HARISSA thinks on 29% of queries and FrugalGPT on 75% for the same accuracy (Table 7), and Self-REF thinks on 43% and is 1.2 points less accurate. On BBH, Self-REF

Table 1: **Accuracy and mean latency per query for each policy at its selected thresholds, every query answered.** Accuracy (Acc.) is in percent and latency (Lat.) in seconds; HARISSA's rows are in bold.

	MedQA		MedMCQA		BBH	
Policy	Acc. ↑	Lat. ↓	Acc. ↑	Lat. ↓	Acc. ↑	Lat. ↓
Device						
direct	78.8	9.8	63.6	7.8	82.2	5.7
chain-of-thought	86.2	81.0	68.6	66.4	90.6	44.9
self-consistency	80.0	29.1	64.6	23.2	85.2	16.8
FrugalGPT	85.6	70.6	63.9	8.7	83.2	8.2
Self-REF	84.5	45.0	66.7	33.9	85.7	12.2
HARISSA	85.7	33.0	68.5	38.4	88.7	11.7
Server						
FrugalGPT	69.4	11.4	59.7	9.9	71.4	6.8
Self-REF	69.7	11.4	59.7	9.0	71.9	7.3
HARISSA	81.8	11.6	60.2	7.1	91.3	8.1

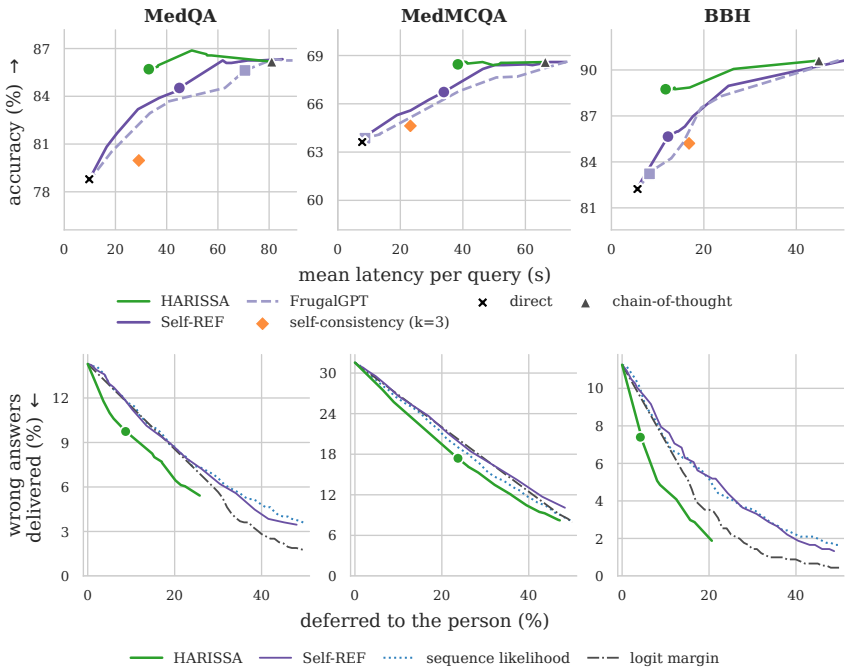

Figure 1: **On a device, HARISSA is within one accuracy point of chain-of-thought at 2.7 times lower latency, averaged over three tasks.** Top: accuracy against mean latency per query, every query answered; the baselines as markers and each cascade as its sweep over the skip threshold, with the selected point filled. Bottom: wrong answers delivered against the share of queries deferred, on HARISSA's delivered answers, one curve per deferral signal (HARISSA is the answer probability and Self-REF is Self-REF confidence); the filled point is the selected deferral threshold. The answer probability takes few distinct values, so its curve ends early, and answers without a logit margin are deferred first.

and HARISSA think on a similar share of queries (15 and 13%) and gain 3.5 and 6.5 points over direct. On MedMCQA, FrugalGPT delivers nearly every direct answer and stays at direct's accuracy. HARISSA is at least as accurate as FrugalGPT and Self-REF at every latency its sweep reaches (Figure 1, top), because the answer probe finds more of the wrong direct answers than FrugalGPT's scorer or Self-REF's confidence.

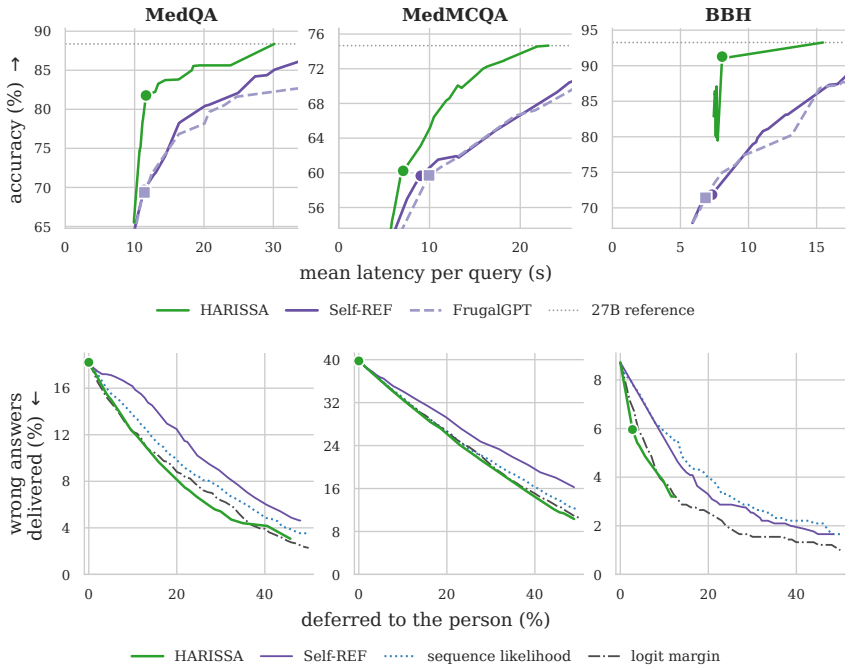

Figure 2: **On a server, HARISSA is more accurate than FrugalGPT and Self-REF at the same latency by skipping the sizes the prefill probe predicts will fail.** Same layout as Figure 1; the dotted line (27B reference) is the 27B size alone, the highest accuracy any size or cascade reaches.

On Gemma 4 and Cogito, HARISSA is within one point of chain-of-thought averaged over the three tasks, and thinks more often where thinking adds more accuracy per second of latency. How much a cascade can save depends on how much thinking costs and how much accuracy it adds, and the three families we run differ on both. On Gemma 4, think costs 1.5 times direct on MedQA, against 8.3 times on Qwen 3.5, so there is little latency to save; HARISSA thinks on 69% of MedQA queries (26% on MedMCQA, 20% on BBH) and is 0.7 points above chain-of-thought at 1.1 times lower latency, averaged over the three tasks. On Cogito, think costs 3.7 times direct and adds at least 9 points on every task, so on MedQA the policy thinks on every query and matches chain-of-thought exactly, and on BBH it is 0.1 points below at 1.7 times lower latency (Appendix E).

5.2 Server cascade

At similar latency on MedQA, HARISSA is 12 points more accurate than FrugalGPT and Self-REF. On the server the actions are the four sizes of one model. The 2B size answers MedQA in 5.8 s and answers half of the queries correctly, and the 27B size takes 29.9 s and answers 88% correctly (Table 6), so a cascade over the sizes saves latency by answering with the smallest size that answers correctly. FrugalGPT and Self-REF generate at the 2B size on every query and deliver its answer on 43% of MedQA queries, at 11.4 s against HARISSA's 11.6 s, and across the sweep HARISSA's curve lies above FrugalGPT's and Self-REF's on every task (Figure 2, top). On BBH HARISSA is 20 points more accurate than FrugalGPT and Self-REF at about 1 s more per query. On MedMCQA HARISSA also delivers the 2B size's answer on a quarter of queries, so the three cascades are equally accurate; when one accuracy point is worth 1 s of latency instead of 0.5 s, the policy skips the 2B size more often and is 12 points more accurate than FrugalGPT and Self-REF (Table 8). Against the 27B size alone, HARISSA is 6.6 points less accurate on MedQA at 2.6 times lower latency.

5.3 Deferring to the person

At 10% of queries deferred, the answer probability leaves the fewest wrong answers on five of six task and setting pairs. The answer probe already scores every answer HARISSA delivers, so

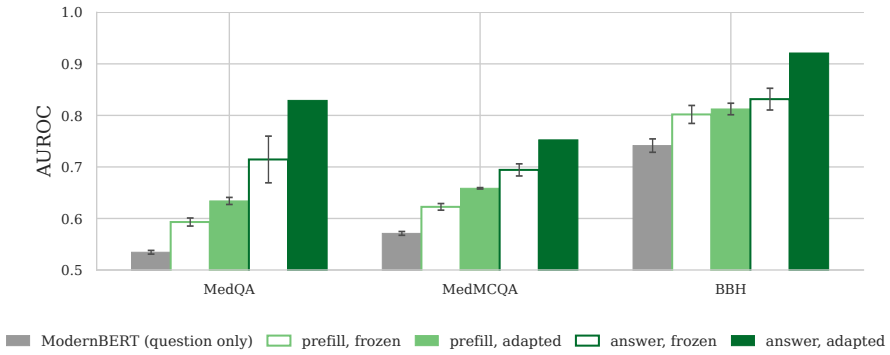

Figure 3: **On the device, the answer probe predicts correctness better than the prefill probe, and fine-tuning improves the prefill probe and the answer probe.** AUROC on the device, averaged over the two actions, of a question-text classifier (grey), the prefill probe (light green), and the answer probe (dark green), frozen (outlined) and fine-tuned (filled); whiskers are the standard deviation over seeds where more than one seed was run.

the policy can defer the least probable answers to the person without generating anything more. The answer probability and the three standard confidence signals (sequence likelihood, the logit margin, and Self-REF confidence) each rank the answers HARISSA delivers, and the lowest-ranked share is deferred (Figures 1 and 2, bottom). At 10% deferred, the answer probability leaves wrong answers on 4.9% of device BBH queries, sequence likelihood on 7.6%, and Self-REF confidence on 7.9%. On server MedMCQA, the one exception, the answer probability is about one point behind sequence likelihood and the logit margin. The policy defers 8.7% of MedQA queries on the device and removes 32% of the wrong answers it would otherwise deliver, nearly four times what deferring the same share at random removes (Table 10).

6 Analysis

Each hidden state predicts whether the answer is correct, and how well it does so decides which probe the policy relies on and whether the probes can choose a size per query.

6.1 What does each hidden state know about the answer?

The answer state predicts correctness better than the prefill state. Before decoding, the hidden state reflects only the question, and after decoding it also reflects the answer the model generated, so the answer state can carry a signal the prefill state cannot. We measure how well each probe separates correct from wrong answers by its AUROC, the probability that a randomly chosen correct answer scores higher than a randomly chosen wrong one (Figure 3). On the device the answer probe reaches 0.83 AUROC on MedQA and the prefill probe 0.63; on BBH the two are closer, 0.92 and 0.81, and every server size shows the same order (Table 11). The prefill probe comes closest on BBH, where a ModernBERT classifier fit on the question text alone reaches 0.74 AUROC against 0.53 on MedQA, so the difficulty on BBH is visible in the question and on the medical tasks depends on whether the model knows the fact. The classifier is below the prefill probe on every task, because a classifier on the question learns which questions are hard in general, while the prefill state is the model's own processing of the question and reflects the question's difficulty and what this model knows.

The cost of the actions decides which probe the policy relies on. Skipping an action on the prefill probe's prediction pays when the latency it saves is worth more, at the deployment's price, than the accuracy the wrong skips lose. On the device, think costs about eight times direct but the prefill probe is weak on the medical tasks, so the development sweep selects a skip threshold of 0 on every task. The policy therefore generates every direct answer and lets the answer check decide, and the skip is selected on MedQA only when one accuracy point is worth 30 s of latency instead of 10 s (Table 8). Without the answer check, the policy decides from the prefill probe alone and is less accurate on every task and slower on MedQA and BBH (85.2% at 55.1 s against 85.7% at 33.0 s on MedQA;

Table 2: **Routing from the answer probabilities gains about one point over the 27B size at twice its latency.** Test accuracy at mean latency per query on the server; the 27B size on every query is the reference. The prefill router generates at the first size whose prefill probability passes a threshold (HARISSA without the answer check); the answer router generates at every size and delivers the answer with the highest answer probability.

Policy	MedQA	MedMCQA	BBH
27B alone	88.4% @ 29.9 s	74.7% @ 22.9 s	93.3% @ 15.2 s
HARISSA	81.8% @ 11.6 s	60.2% @ 7.1 s	91.3% @ 8.1 s
prefill router	77.3% @ 10.1 s	59.3% @ 7.2 s	88.4% @ 7.8 s
answer router	89.7% @ 66.0 s	73.6% @ 49.7 s	94.4% @ 36.6 s

Table 6). On the server a skipped size costs only its prefill pass and probe while a generated size costs its full latency, so the policy skips the 2B size on most queries. Without the prefill skip it generates at the 2B size on every query, as FrugalGPT and Self-REF do, and is 6.3 points less accurate on MedQA at similar latency.

Fine-tuning improves the answer probe and barely changes the answers. Fine-tuning trains the model to predict correctness at the answer state (Section 3), and the answer probe is where its gain lands. On the device it raises the answer probe by 0.12 AUROC on MedQA (0.06 on MedMCQA, 0.09 on BBH) and the prefill probe by at most 0.04 (Figure 3), while each action's accuracy moves by at most 1.7 points in either setting (Table 12). On the device, the policy run on the model without fine-tuning, with the probes refit on the frozen states, is 2.5 points less accurate on MedQA (5.5 on MedMCQA, 2.6 on BBH). On the server the gain is smaller, 1.3 points on MedQA and 1.7 on BBH, and on MedMCQA the frozen policy selects a slower point under the 0.5 s price, 7 points more accurate at 1.7 times the latency (Table 6).

6.2 Can the probes route instead of cascade?

Routing from the answer probabilities gains about one point over the 27B size at twice its latency. A router chooses one size per query instead of cascading. We test two routers, one that decides from the prefill state and one that decides from the answer state. The prefill router runs the prefill pass at every size and generates once, at the cheapest size whose prefill probability passes a threshold. The answer router generates at every size and delivers the answer with the highest answer probability. Every size answered every test query, and a router that always chose a size whose answer is correct would be 7 points above the 27B size on MedQA (Table 2), so a router has room to gain. The prefill router delivers the 4B size's wrong answers that HARISSA's answer check would pass to the 9B size, and it is less accurate than the cascade at similar latency on every task (77.3% against 81.8% on MedQA). The answer router gains 1.3 points over the 27B size on MedQA at 2.2 times its latency. Each probe is calibrated within one size, so its probability estimates whether that size's answer is correct, and comparing the estimates of two sizes recovers little of the room. The policy therefore cascades, and accuracy above the 27B size requires a probe trained to compare sizes.

7 Conclusion

We show that a small model's own hidden states can decide, for each query, whether to answer directly or reason first, which size of the model to run, and whether to deliver the answer or defer the query to the person. A probe on the prefill state skips an action the model is predicted to get wrong before it generates, and a probe on the answer state withholds an answer predicted wrong. On a device, HARISSA is within one accuracy point of chain-of-thought at 2.7 times lower latency, averaged over three tasks. On a server holding four sizes of one model, HARISSA is more accurate than FrugalGPT and Self-REF at the same latency, and at the same deferral rate the answer probability leaves fewer wrong answers than the standard confidence signals in five of six task and setting pairs.

REPRODUCIBILITY STATEMENT

The fine-tuning, probe, and cascade code, the configuration of every run, and the graded answers behind every table are in the supplementary material and will be released on publication. Section 3 states the objective, Algorithm 1 the policy, and Section 3 and Appendix B the fine-tuning recipe, the probed layers, and the probe and calibration settings. Appendix A gives the split sizes, the BBH subtasks, and the prompt and decoding settings; Section 4 and Appendix E name every model, with Hugging Face identifiers for Gemma 4 and Cogito. Latency is computed from per-size prefill and per-token rates measured on one NVIDIA A40, listed with the generated-token counts in Appendix C, so every latency in the paper can be recomputed from the released outputs. Section 4 fixes the splits on which the probes are fit, the thresholds chosen, and the results reported, and Appendices D to G report every policy, ablation, size, deferral signal, and probe on the test split.

AI USE STATEMENT

In this work, we used generative AI tools to draft and edit the paper's text, to review its structure and prose, to help implement the method and the results-reporting and figure scripts, and, in part, to analyze data, to interpret results, and to design experiments. We have not used generative AI tools to generate data, to formulate or prove claims, or to clean datasets. Every number in the paper was checked against the tables printed by our own code, and all AI-assisted text was reviewed by the authors. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

American Bar Association Standing Committee on Ethics and Professional Responsibility. Formal opinion 512: Generative artificial intelligence tools. ABA Formal Opinion 512, July 2024. URL https://www.americanbar.org/content/dam/aba/administrative/professional_responsibility/ethics-opinions/aba-formal-opinion-512.pdf. Accessed September 2026.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *International Conference on Learning Representations*, 2024. arXiv:2310.11511.

Amos Azaria and Tom Mitchell. The internal state of an LLM knows when it's lying. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 967–976, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.68. URL https://aclanthology.org/2023.findings-emnlp.68/.

Katherine E. Brown and Sharon E. Davis. Gaps in artificial intelligence research for rural health in the United States: A scoping review. *Journal of the American Medical Informatics Association*, 33 (2):509–520, February 2026. doi: 10.1093/jamia/ocaf206.

Lida Chen, Zujie Liang, Xintao Wang, Jiaqing Liang, Yanghua Xiao, Feng Wei, Jinglei Chen, Zhenghong Hao, Bing Han, and Wei Wang. Teaching large language models to express knowledge boundary from their own signals. In *Proceedings of the 3rd Workshop on Towards Knowledgeable Foundation Models (KnowFM)*, pp. 26–39, Vienna, Austria, August 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.knowllm-1.3. URL https://aclanthology.org/2025.knowllm-1.3/.

Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=cSimKw5p6R. arXiv:2305.05176.

Yu-Neng Chuang, Prathusha Kameswara Sarma, Parikshit Gopalan, John Boccio, Sara Bolouki, Xia Hu, and Helen Zhou. Learning to route LLMs with confidence tokens. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 10859–10878. PMLR, 2025a. URL https://proceedings.mlr.press/v267/chuang25b.html. arXiv:2410.13284.

Yu-Neng Chuang, Leisheng Yu, Guanchu Wang, Lizhe Zhang, Zirui Liu, Xuanting Cai, Yang Sui, Vladimir Braverman, and Xia Hu. Confident or seek stronger: Exploring uncertainty-based on-device LLM routing from benchmarking to generalization, 2025b. URL https://arxiv.org/abs/2502.04428.

Deep Cogito. Cogito v1 preview: Llama 8B. Hugging Face model card, 2025. URL https://huggingface.co/deepcogito/cogito-v1-preview-llama-8B. Accessed September 2026.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Ruhle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. Hybrid LLM: Cost-efficient and quality-aware query routing. In *International Conference on Learning Representations*, 2024. arXiv:2404.14618.

Ran El-Yaniv and Yair Wiener. On the foundations of noise-free selective classification. *Journal of Machine Learning Research*, 11:1605–1641, 2010.

European Parliament and Council. Regulation (EU) 2016/679 (general data protection regulation). Official Journal of the European Union, OJ L 119, 4.5.2016, pp. 1–88, 2016. URL https://eur-lex.europa.eu/eli/reg/2016/679/oj. Accessed September 2026.

Claudio Fanconi and Mihaela van der Schaar. Cascaded language models for cost-effective human–AI decision-making. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://openreview.net/forum?id=4Q1vA6P9J9. arXiv:2506.11887.

Muskan Garg, Shaina Raza, Shebuti Rayana, Xingyi Liu, and Sunghwan Sohn. The rise of small language models in healthcare: A comprehensive survey. *Computer Science Review*, 62:100999, 2026. doi: 10.1016/j.cosrev.2026.100999. arXiv:2504.17119.

Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. In *Advances in Neural Information Processing Systems*, volume 30, pp. 4878–4887, 2017. arXiv:1705.08500.

Gemma Team. Gemma 4 technical report. Technical report, 2026. URL https://arxiv.org/abs/2607.02770. Accessed September 2026.

Granite Team, IBM. Granite 4.2-8B model card. Hugging Face model card, 2026. URL https://huggingface.co/ibm-granite/granite-4.2-8b. Accessed September 2026.

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

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *International Conference on Learning Representations*, 2017.

Hugging Face. SmolLM3-3B model card. Hugging Face model card, 2025. URL https://huggingface.co/HuggingFaceTB/SmolLM3-3B. Accessed September 2026.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong Park. Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 7036–7050, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.389. URL https://aclanthology.org/2024.naacl-long.389/.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. What disease does this patient have? A large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14):6421, 2021. doi: 10.3390/app11146421. arXiv:2009.13081.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt,

Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. Language models (mostly) know what they know, 2022. URL https://arxiv.org/abs/2207.05221.

Hossein Hosseini Kasnavieh, Gholamreza Haffari, Christopher Leckie, and Adel N. Toosi. IntroLM: Introspective language models via prefilling-time self-evaluation. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 12313–12326, San Diego, California, United States, July 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.findings-acl.598. URL https://aclanthology.org/2026.findings-acl.598/.

Guosheng Liang, Longguang Zhong, Ziyi Yang, and Xiaojun Quan. ThinkSwitcher: When to think hard, when to think fast. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 5185–5201, Suzhou, China, November 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-emnlp.278. URL https://aclanthology.org/2025.findings-emnlp.278/.

Xiang Liu, Xuming Hu, Xiaowen Chu, and Eunsol Choi. DiffAdapt: Difficulty-adaptive reasoning for token-efficient LLM inference. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=644FH1vVIl. arXiv:2510.19669.

Zhenyan Lu, Xiang Li, Dongqi Cai, Rongjie Yi, Fangming Liu, Xiwen Zhang, Nicholas D. Lane, and Mengwei Xu. Small language models: Survey, measurements, and insights, 2024. URL https://arxiv.org/abs/2409.15790.

Sasha Luccioni, Yacine Jernite, and Emma Strubell. Power hungry processing: Watts driving the cost of AI deployment? In *Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency*, pp. 85–99. ACM, June 2024. doi: 10.1145/3630106.3658542.

William Lugoloobi, Thomas Foster, William Bankes, and Chris Russell. LLMs encode their failures: Predicting success from pre-generation activations. In *Conference on Language Modeling*, 2026. arXiv:2602.09924.

David Madras, Toniann Pitassi, and Richard Zemel. Predict responsibly: Improving fairness and accuracy by learning to defer. In *Advances in Neural Information Processing Systems*, volume 31, 2018. arXiv:1711.06664.

Andrey Malinin and Mark Gales. Uncertainty estimation in autoregressive structured prediction. In *International Conference on Learning Representations*, 2021.

Hussein Mozannar and David Sontag. Consistent estimators for learning to defer to an expert. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 7076–7087. PMLR, 2020. URL https://proceedings.mlr.press/v119/mozannar20b.html. arXiv:2006.01862.

Paige Nong, Julia Adler-Milstein, Nate C. Apathy, A. Jay Holmgren, and Jordan Everson. Current use and evaluation of artificial intelligence and predictive models in US hospitals. *Health Affairs*, 44(1):90–98, January 2025. doi: 10.1377/hlthaff.2024.00842.

NVIDIA. NVIDIA Nemotron 3 Nano 30B-A3B model card. Hugging Face model card, 2025. URL https://huggingface.co/nvidia/NVIDIA-Nemotron-3-Nano-30B-A3B-BF16. Accessed September 2026.

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 from preference data. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=8sSqNntaMr. arXiv:2406.18665.

OpenAI. Data controls in the OpenAI platform. Developer documentation, 2026. URL https://developers.openai.com/api/docs/guides/your-data. Accessed September 2026.

Hadas Orgad, Michael Toker, Zorik Gekhman, Roi Reichart, Idan Szpektor, Hadas Kotek, and Yonatan Belinkov. LLMs know more than they show: On the intrinsic representation of LLM hallucinations. In *International Conference on Learning Representations*, 2025.

Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. MedMCQA: A large-scale multi-subject multi-choice dataset for medical domain question answering. In *Proceedings of the Conference on Health, Inference, and Learning*, volume 174 of *Proceedings of Machine Learning Research*, pp. 248–260. PMLR, 2022. URL https://proceedings.mlr.press/v174/pal22a.html. arXiv:2203.14371.

Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Qwen Team. Qwen3.5-9B model card. Hugging Face model card, 2026a. URL https://huggingface.co/Qwen/Qwen3.5-9B. Accessed September 2026.

Qwen Team. Qwen3.5: Towards native multimodal agents. Blog post and model card, 2026b. URL https://qwen.ai/blog?id=qwen3.5. Accessed September 2026.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=4FWAwZtd2n. arXiv:2408.03314.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. Challenging BIG-Bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13003–13051, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.824. URL https://aclanthology.org/2023.findings-acl.824/.

Fali Wang, Jihai Chen, Shuhua Yang, Ali Al-Lawati, Linli Tang, Hui Liu, and Suhang Wang. A survey on collaborating small and large language models for performance, cost-effectiveness, cloud-edge privacy, and trustworthiness, 2025. URL https://arxiv.org/abs/2510.13890.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw. arXiv:2203.11171.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837, 2022. arXiv:2201.11903.

Isabella Catharina Wiest, Dyke Ferber, Jiefu Zhu, Marko van Treeck, Sonja K. Meyer, Radhika Juglan, Zunamys I. Carrero, Daniel Paech, Jens Kleesiek, Matthias P. Ebert, Daniel Truhn, and Jakob Nikolas Kather. Privacy-preserving large language models for structured medical information retrieval. *npj Digital Medicine*, 7(1):257, 2024. doi: 10.1038/s41746-024-01233-2.

Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. BloombergGPT: A large language model for finance, 2023. URL https://arxiv.org/abs/2303.17564.

Jiajun Xu, Zhiyuan Li, Wei Chen, Qun Wang, Xin Gao, Qi Cai, and Ziyuan Ling. On-device language models: A comprehensive review, 2024. URL https://arxiv.org/abs/2409.00088.

Murong Yue, Jie Zhao, Min Zhang, Liang Du, and Ziyu Yao. Large language model cascades with mixture of thought representations for cost-efficient reasoning. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=6okaSfANzh. arXiv:2310.03094.

Bianca Zadrozny and Charles Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 694–699. ACM, 2002. doi: 10.1145/775047.775151.

Michael J. Zellinger, Rex Liu, and Matt Thomson. Cost-saving LLM cascades with early abstention, 2025. URL https://arxiv.org/abs/2502.09054.

Jiajie Zhang, Nianyi Lin, Lei Hou, Ling Feng, and Juanzi Li. AdaptThink: Reasoning models can learn when to think. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 3716–3730, Suzhou, China, November 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.184. URL https://aclanthology.org/2025.emnlp-main.184/.

A DATASETS AND PROMPTS

Table 3 gives the split sizes. From BBH we use the sixteen multiple-choice subtasks: tracking_shuffled_objects_{three,five,seven}_objects, logical_deduction_{three,five,seven}_objects, disambiguation_qa, salient_translation_error_detection, date_understanding, hyperbaton, temporal_sequences, movie_recommendation, ruin_names, snarks, penguins_in_a_table, and reasoning_about_colored_objects.

Table 3: Split sizes (queries).

Task	Train	Dev	Test
MedQA	10,178	1,272	1,273
MedMCQA	10,178	2,091	2,092
BBH	2,188	552	906

Prompt and decoding. Every action uses the zero-shot prompt below, with one reasoning hint per task (for the medical tasks, "step-by-step analysis of the medical scenario and answer choices"). On the device, direct and think differ only in the thinking flag of the chat template, so the prefill state is identical up to the template's mode tokens. Under think, the model's thinking precedes the JSON. Grading reads the `final_answer` letter. Every model decodes at temperature 0.6 with a $10{,}000$-token cap on the response, thinking included.

```
# Question
{question text, with lettered options}
# Required Output Format
“‘json
{
  "reasoning": "<{reasoning hint}>",
  "final_answer": "A"
}
“‘
```

B FINE-TUNING AND PROBE DETAILS

We fine-tune a LoRA adapter on the attention projections $\{q, k, v, o\}$ (rank 16, $\alpha = 32$, dropout 0.05) with batch size 8, learning rate 10^{-4}, and sequence length 3072, for three epochs (one epoch for the 27B size). The same recipe is used for the device model and for each server size, which is fine-tuned separately; Cogito uses learning rate 2×10^{-5}. Training labels grade the frozen model's stored answers, and one forward pass over the prompt and that answer trains the answer head on the last generated token and one prefill head per action on the last prompt token, under a binary cross-entropy loss at equal weight. The development split is evaluated every 250 steps; on the device we keep the checkpoint with the lowest cascade objective on the development split, and on the server the checkpoint whose answer head best separates correct from incorrect answers there. The heads are discarded, and the probes are fit on the fine-tuned model's own answers.

The probed layer is fixed per size: layer 32 for the 4B and 9B sizes, layer 64 for the 27B size, and the final layer for the 2B size, chosen once from a layer sweep on MedQA; Gemma 4 is probed at its final layer and Cogito at layer 32. Each probe is an L2-regularized logistic regression ($C = 0.5$, L-BFGS) on that layer's hidden state, with features standardized by training-split statistics, fit on the training split and calibrated by isotonic regression on the development split, with the development scores cross-fitted and out-of-range scores clipped.

C COST MODEL

A query's latency is a per-size prefill time plus its generated tokens times a per-token generation rate (Table 4). Both were measured for each size on one NVIDIA A40 (48 GB, vLLM, BF16, concurrency one) on eight MedQA development prompts, and the same profile serves both settings, so the device

numbers are those of a 4B model on that GPU rather than on a phone. Table 5 gives the mean generated tokens per action and the resulting latency of each action alone. Each probe costs one dot product on a state the model computes while answering. A skipped action costs its prefill pass and probe: 0.08 s for direct on the device, and 0.23 s for the 2B, 4B, and 9B sizes together on the server, measured as the latency of the policy at a skip threshold above 1, which skips every action but the last, minus the latency of the last action alone.

Table 4: **Measured prefill time and per-token generation time on one A40.**

Model	Prefill (ms)	Generation (ms per token)
Qwen 3.5 2B	44.0	7.92
Qwen 3.5 4B	82.5	17.14
Qwen 3.5 9B	104.0	30.45
Qwen 3.5 27B	222.8	49.56
Gemma 4 E4B	64.5	19.52
Cogito v1 8B	87.2	27.98

Table 5: **Mean generated tokens per query and the latency of each action alone (s), Qwen 3.5, test split.** Input is the mean prompt length in tokens; the 4B size is the device model's direct action, so the two columns agree.

		Generated tokens					
Task	Input	direct	think	2B	4B	9B	27B
MedQA	225	565	4,723	725	565	670	599
MedMCQA	46	450	3,868	515	450	485	457
BBH	113	325	2,614	737	325	317	303
		Latency of the action alone (s)					
MedQA		9.8	81.0	5.8	9.8	20.5	29.9
MedMCQA		7.8	66.4	4.1	7.8	14.9	22.9
BBH		5.7	44.9	5.9	5.7	9.8	15.2

D FULL RESULTS IN BOTH SETTINGS

Table 6: **Ablations, each size alone, and the policy without fine-tuning: accuracy (% of queries) at mean latency per query (s) at the selected thresholds.** Thresholds are chosen on the development split at 10 s per accuracy point on the device and 0.5 s on the server; HARISSA and the baselines are in Table 1, and the server policy without the answer check is the prefill router of Table 2. HARISSA without fine-tuning runs the policy on the frozen model with probes fit on its states.

Policy	MedQA	MedMCQA	BBH
Device			
HARISSA without the prefill skip	86.6% @ 39.7 s	67.5% @ 26.5 s	88.6% @ 11.6 s
HARISSA without the answer check	85.2% @ 55.1 s	66.2% @ 27.0 s	86.5% @ 15.0 s
HARISSA without fine-tuning	83.2% @ 36.5 s	63.0% @ 7.9 s	86.1% @ 13.8 s
Server			
2B alone	45.6% @ 5.8 s	47.1% @ 4.1 s	67.9% @ 5.9 s
4B alone	78.8% @ 9.8 s	63.1% @ 7.8 s	82.8% @ 5.7 s
9B alone	83.3% @ 20.5 s	70.1% @ 14.9 s	87.3% @ 9.8 s
27B alone	88.4% @ 29.9 s	74.7% @ 22.9 s	93.3% @ 15.2 s
HARISSA without the prefill skip	75.5% @ 13.3 s	58.1% @ 8.0 s	89.1% @ 11.8 s
HARISSA without the answer check	77.3% @ 10.1 s	59.3% @ 7.2 s	88.4% @ 7.8 s
HARISSA without fine-tuning	80.5% @ 12.4 s	67.2% @ 11.9 s	89.6% @ 8.4 s

Table 7: **The action each cascade delivers from (% of queries), at the selected thresholds.**

		Device		Server			
Task	Policy	direct	think	2B	4B	9B	27B
MedQA	HARISSA	71.4	28.6	9.0	78.9	12.1	0.0
	FrugalGPT	25.0	75.0	43.3	56.4	0.3	0.0
	Self-REF	56.6	43.4	43.1	56.8	0.1	0.0
MedMCQA	HARISSA	54.0	46.0	27.2	71.0	1.8	0.0
	FrugalGPT	98.7	1.3	27.9	70.8	1.3	0.0
	Self-REF	60.7	39.3	36.9	63.1	0.0	0.0
BBH	HARISSA	86.6	13.4	0.0	68.4	18.7	12.9
	FrugalGPT	94.3	5.7	89.3	7.3	3.4	0.0
	Self-REF	85.3	14.7	86.5	6.5	7.0	0.0

Table 8: **HARISSA's selected point at each price on latency: accuracy at mean latency per query, with the selected skip threshold in parentheses.** The paper reports 10 s per accuracy point on the device and 0.5 s on the server.

Task	Seconds per accuracy point			
Device	2	5	10	30
MedQA	82.6% @ 18.2 s (0.00)	83.3% @ 19.5 s (0.00)	85.7% @ 33.0 s (0.00)	86.9% @ 49.8 s (0.73)
MedMCQA	63.6% @ 7.9 s (0.00)	63.6% @ 7.9 s (0.00)	68.5% @ 38.4 s (0.00)	68.5% @ 38.5 s (0.03)
BBH	88.7% @ 11.7 s (0.00)	88.7% @ 11.7 s (0.00)	88.7% @ 11.7 s (0.00)	88.7% @ 11.7 s (0.00)
Server	0.1	0.25	0.5	1
MedQA	68.2% @ 10.1 s (0.40)	81.8% @ 11.6 s (0.60)	81.8% @ 11.6 s (0.60)	84.4% @ 13.5 s (0.60)
MedMCQA	53.6% @ 5.7 s (0.00)	60.2% @ 7.1 s (0.45)	60.2% @ 7.1 s (0.48)	72.6% @ 16.1 s (0.73)
BBH	85.0% @ 7.2 s (0.80)	90.4% @ 7.9 s (0.88)	91.3% @ 8.1 s (0.88)	91.9% @ 8.4 s (0.88)

E Replication on other model families

The device experiments of Section 5.1 are repeated on Gemma 4 E4B (Gemma Team, 2026) and Cogito v1 8B (Deep Cogito, 2025) with the same tasks, splits, actions, fine-tuning recipe, probes, threshold selection at 10 s per accuracy point, and cost model, each family with its own measured rates and token counts (Appendix C). Table 9 reports each policy at its selected thresholds, the share of queries HARISSA thinks on, and each probe's AUROC; Section 5.1 reads the table.

Table 9: **Gemma 4 and Cogito on the device.** Accuracy (% of queries) at mean latency per query (s) for each policy at its selected thresholds, the share of queries HARISSA thinks on (a dash marks a share not recorded), and the AUROC of the fine-tuned probes averaged over the two actions; test split, one seed.

		Accuracy @ latency			HARISSA	Probe AUROC	
Family	Task	direct	chain-of-thought	HARISSA	thinks on	prefill	answer
Gemma 4	MedQA	66.1% @ 17.0 s	74.1% @ 24.8 s	73.9% @ 25.7 s	69.4%	0.611	0.766
	MedMCQA	56.8% @ 13.9 s	58.7% @ 20.2 s	59.5% @ 19.2 s	26.1%	0.658	0.759
	BBH	80.9% @ 7.6 s	82.7% @ 14.4 s	84.1% @ 10.5 s	19.5%	0.878	0.909
Cogito	MedQA	58.4% @ 9.6 s	67.6% @ 35.7 s	67.6% @ 35.7 s	100.0%	0.611	0.727
	MedMCQA	48.1% @ 6.7 s	61.6% @ 25.0 s	61.0% @ 23.2 s	–	0.645	0.676
	BBH	76.8% @ 4.6 s	86.4% @ 22.6 s	86.3% @ 13.0 s	–	0.753	0.841

F DEFERRAL

Table 10 gives HARISSA's deferral in both settings when the deferral threshold equals the selected fall-through threshold, so that the answer probability that discards an answer at a non-last action also withholds it at the last.

Table 10: **Deferral by the answer probability at the selected fall-through threshold, on HARISSA's delivered answers at its selected thresholds.** The share of queries deferred, and the wrong answers delivered (% of queries) with no deferral and at that threshold.

		Wrong answers delivered	
Task	Deferred	none	at the threshold
Device			
MedQA	8.7%	14.3	9.7
MedMCQA	23.7%	31.5	17.4
BBH	4.2%	11.3	7.4
Server			
MedQA	0.0%	18.2	18.2
MedMCQA	0.0%	39.8	39.8
BBH	2.8%	8.7	6.0

G FINE-TUNING'S EFFECT ON THE PROBES AND THE ANSWERS

Table 11 gives each probe's AUROC on the frozen and the fine-tuned model beside a classifier on the question text, and Table 12 gives each action's accuracy on both models.

Table 11: **Per-action AUROC of each probe, test split.** On the device, frozen and fine-tuned, beside a ModernBERT classifier fit on the question text; on the server, the fine-tuned probes of each size; one seed per cell.

			Prefill probe		Answer probe	
Task	Action	Question only	frozen	fine-tuned	frozen	fine-tuned
MedQA	direct	0.540	0.594	0.639	0.716	0.840
	think	0.539	0.572	0.610	0.812	0.818
MedMCQA	direct	0.573	0.611	0.655	0.680	0.759
	think	0.574	0.645	0.659	0.716	0.746
BBH	direct	0.714	0.735	0.762	0.752	0.914
	think	0.805	0.828	0.855	0.852	0.928
MedQA	2B			0.604		0.777
	4B			0.679		0.836
	9B			0.650		0.819
	27B			0.745		0.851
MedMCQA	2B			0.615		0.720
	4B			0.702		0.791
	9B			0.681		0.738
	27B			0.727		0.780
BBH	2B			0.702		0.822
	4B			0.785		0.897
	9B			0.824		0.875
	27B			0.830		0.881

Table 12: **Fine-tuning changes the answers by at most 1.7 points.** Accuracy (% of queries) of each fixed action on the frozen and the fine-tuned model, as frozen→fine-tuned (Δ); test split, one seed.

Task	Action			
Device	direct	think		
MedQA	78.6→78.8 (+0.2)	84.8→86.2 (+1.3)		
MedMCQA	63.0→63.6 (+0.7)	66.9→68.6 (+1.7)		
BBH	83.8→82.2 (−1.5)	89.1→90.6 (+1.5)		
Server	2B	4B	9B	27B
MedQA	45.6→45.6 (+0.1)	78.6→78.8 (+0.2)	82.9→83.3 (+0.5)	88.8→88.4 (−0.5)
MedMCQA	46.5→47.1 (+0.6)	63.0→63.1 (+0.1)	70.2→70.1 (−0.1)	75.2→74.7 (−0.5)
BBH	68.9→67.9 (−1.0)	83.8→82.8 (−1.0)	86.4→87.3 (+0.9)	93.3→93.3 (+0.0)