GATEKEEPER: Improving Model Cascades Through Confidence Tuning

Stephan Rabanser[*]
Princeton University
rabanser@princeton.edu

Nathalie Rauschmayr
Google
rauschmayr@google.com

Achin Kulshrestha
Google
kulac@google.com

Petra Poklukar
Google
poklukar@google.com

Wittawat Jitkrittum
Google
wittawat@google.com

Sean Augenstein
Google
saugenst@google.com

Congchao Wang
Google
congchaowang@google.com

Federico Tombari
Google
tombari@google.com

Abstract

Large-scale machine learning models deliver strong performance across a wide range of tasks but come with significant computational and resource constraints. To mitigate these challenges, local smaller models are often deployed alongside larger models, relying on routing and deferral mechanisms to offload complex tasks. However, existing approaches inadequately balance the capabilities of these models, often resulting in unnecessary deferrals or sub-optimal resource usage. In this work we introduce a novel loss function called GATEKEEPER for calibrating smaller models in cascade setups. Our approach fine-tunes the smaller model to confidently handle tasks it can perform correctly while deferring complex tasks to the larger model. Moreover, it incorporates a mechanism for managing the trade-off between model performance and deferral accuracy, and is broadly applicable across various tasks and domains without any architectural changes. We evaluate our method on encoder-only, decoder-only, and encoder-decoder architectures. Experiments across image classification, language modeling, and vision-language tasks show that our approach substantially improves deferral performance.

1 Introduction

In recent years, large-scale machine learning models such as Gemini (GeminiTeam et al., 2023), GPT-4 (Achiam et al., 2023) or Claude (Anthropic, 2024) have gained significant traction due to their remarkable ability to address a wide array of tasks. These tasks range from natural language understanding and generation, including machine translation, summarization, and conversational agents, to computer vision applications like image recognition, object detection, and image captioning. The versatility and high performance of these expansive models make them invaluable tools across diverse domains, including healthcare (Nazi & Peng, 2024), finance (Li et al., 2023), education (Wang et al., 2024b), and entertainment (Gallotta et al., 2024).

Deploying and operating such large models presents significant challenges in terms of latency, memory, compute and storage (Pope et al., 2023). Optimizing inference costs is an active research

[*]Work done while a Student Researcher at Google.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

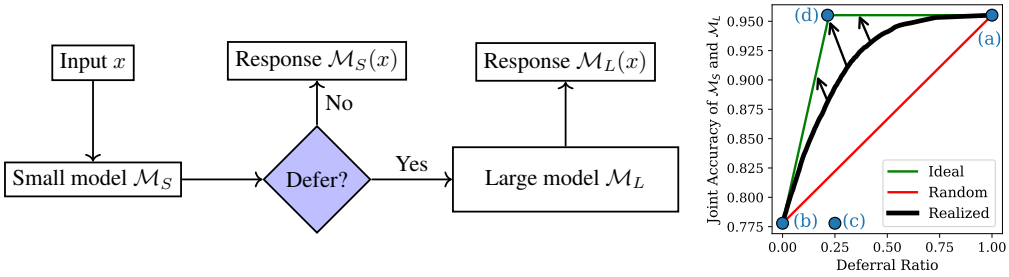

Figure 1: **Overview of the cascading setup (left) and performance trade-off (right)**. *Left*: Cascading determines which inputs should be predicted by a small model $\mathcal{M}_S$ or routed to a large model $\mathcal{M}_L$. *Right*: Performance is measured as a trade-off between joint accuracy across $\mathcal{M}_S$ and $\mathcal{M}_L$ and deferral ratio. Ideal deferral strategies optimize this trade-off and push the realized deferral curve closer to the ideal deferral depicted in (d). (a) depicts full deferral; (b) depicts no deferral; and (c) depicts excessive deferral of requests that could have been correctly handled by $\mathcal{M}_S$.

area which includes both techniques for reducing the size of the existing large model such as model compression (Hoefler et al., 2021), model pruning (Ma et al., 2023; Cheng et al., 2024) and distillation (Yang et al., 2024), and those aiming to leverage a sequence of models such as speculative decoding (Leviathan et al., 2023) and model cascades (Dohan et al., 2022; Chen et al., 2023a; Gupta et al., 2024; Chen et al., 2024a). However, due to scaling laws showing that the performance of a Large Language Model (LLM) increases with its size (Kaplan et al., 2020), the latter category of methods leveraging a sequence of models is currently a more promising direction to lower inference costs without sacrificing the capabilities of large models.

Both speculative decoding and model cascading rely on the existence of a large performant model $\mathcal{M}_L$ and a small model $\mathcal{M}_S$ that is cheap, fast, and less accurate. Speculative decoding leverages $\mathcal{M}_S$ for generating a set of draft tokens that are then validated by $\mathcal{M}_L$ in parallel, a technique successfully deployed in industry applications (Leviathan, 2024). In contrast, model cascades leverage a deferral rule for selecting the most suitable model to process a given request (see Figure 1 left). While the success of speculative decoding necessitates a highly performant $\mathcal{M}_S$ to generate quality draft tokens, model cascades allow the deployment of a less capable $\mathcal{M}_S$ by invoking $\mathcal{M}_L$ only for inference requests outside the small model's scope. In this work, we contribute to the advancement of the model cascades.

Model cascades achieve efficient deferral by optimizing two objectives: compute budget and joint accuracy. We illustrate this trade-off in Figure 1 (right). Assume we have *x* inference requests and a small model $\mathcal{M}_S$ that uses only *20%* of the compute budget of the large model $\mathcal{M}_L$. There are three worst-case scenarios: (a) $\mathcal{M}_S$ defers all requests to $\mathcal{M}_L$, yielding the highest joint accuracy (equal to that of $\mathcal{M}_L$) but the worst compute budget (*1.2x*), since both models process all inputs; (b) $\mathcal{M}_S$ never defers, achieving the lowest compute budget (*0.2x*) but also the lowest joint accuracy (equal to that of $\mathcal{M}_S$); (c) $\mathcal{M}_S$ defers only requests it would have answered correctly, increasing compute budget over (b) without improving joint accuracy. In contrast, the ideal case (d) occurs when $\mathcal{M}_S$ defers only the requests it would misclassify, achieving optimal joint accuracy for a given compute budget (*0.2–1x*). We define how closely a model approximates this ideal as its *deferral performance*.

In this paper, we address the following research question:

How can we optimize model cascades to maximize deferral performance?

In other words, we focus on designing effective model cascades by making the small model more aware of what it does not know. We do so by introducing a *general-purpose* loss function, called GATEKEEPER, that calibrates the small model's confidence in its predictions. By fine-tuning $\mathcal{M}_S$ to output high confidence for correct predictions and low confidence for incorrect ones, we improve the reliability of its uncertainty estimates and facilitate learning of common tasks—thereby directly improving deferral performance. Crucially, GATEKEEPER includes a built-in mechanism for managing the trade-off between *model* and *deferral* performance, and is applicable to arbitrary architectures.

We empirically demonstrate the efficacy of the GATEKEEPER loss on encoder-only vision models for image classification, decoder-only language models (LMs) for closed-form text generation, and encoder-decoder models for vision-language (VL) tasks such as open set classification and captioning. Our main results show that models trained with GATEKEEPER outperform an untuned baseline by 0.72x/2x on CIFAR-100/TinyImagenet and 7x/10x on ARC-e/c, respectively, in terms of deferral performance. As a result, GATEKEEPER paves the way for more scalable and efficient deployment strategies, leveraging the collaboration between local and large-scale models to deliver high-quality results in applications with real-time processing demands.

2 Related Work

Our proposed method improves model cascades through uncertainty-aware finetuning. Next, we describe related work for both research areas. We provide an extended discussion in Appendix B.1.

Model Cascades. A cascade consists of a series of models and a deferral rule which determines the appropriate model given an input request. The concept of model cascades has first been proposed by Viola & Jones (2001), where it is used to accelerate object detection models. Cascades have been extensively studied for classification-based computer vision (Wang et al., 2017; Trapeznikov & Saligrama, 2013; Bolukbasi et al., 2017a; Jitkrittum et al., 2023) and in models for natural language processing (Dohan et al., 2022; Mamou et al., 2022; Varshney & Baral, 2022).

Cascades are particularly promising in the context of generative models such as LLMs and VLMs since they can significantly reduce inference costs. In contrast to speculative decoding (Leviathan et al., 2023), they aim to invoke the large model only for difficult examples. However, the two approached can also be combined (Narasimhan et al., 2025). While Chen et al. (2024b) combine the deferral logic with speculative decoding to generate initial tokens using larger models and later tokens using a smaller model, the majority of research on model cascades has focused on using pre-trained LLMs with a post-hoc deferral logic (Narasimhan et al., 2022; Jitkrittum et al., 2023; Yue et al., 2024). Kolawole et al. (2024) use agreement across multiple models to make deferral decisions, while Gupta et al. (2024) present a method to learn a deferral rule based on quantiles of per-token log probabilities.

Model cascades can be improved through training and fine-tuning. Wang et al. (2024a) train the small model only on easier examples by masking tokens where both large and small models are incorrect. Enomoro & Eda (2021) extend the training objective of image classification models with confidence calibration. In contrast, our approach extends cascades to VLMs and boosts inference performance by making smaller models less confident when incorrect.

Uncertainty-Aware Models. Extensive research has been conducted in the field of uncertainty quantification in deep learning and we refer to Abdar et al. (2021) for a detailed survey. While many methods have been proposed for classification-based models, measuring uncertainty for generative models is still an active area of research. Based on the assumed level of access to model internals, existing methods can be summarized into three main categories:

Black box methods operate solely via the model's query interface by injecting tailored instructions into prompts. These modify the prompt $\mathbf{x}$ by appending instructions $\mathbf{x}'$ for the model to respond less confidently: $\mathbf{x} \leftarrow \mathbf{x}|\mathbf{x}'$. Related methods are confidence quantification (Shrivastava et al., 2023), rejection and remote model awareness (Kadavath et al., 2022), and self-critiquing (Gou et al., 2023). Xiong et al. (2024) show that LLMs can express their confidence through prompting and sampling strategies and their experiments indicate that these models tend to be overconfident.

Gray box approaches employ confidence-based strategies centered on post-processing the model's logits. Many uncertainty techniques such as ensembling (Lakshminarayanan et al., 2017) and Bayesian methods (Blundell et al., 2015)) are not scalable. Related techniques are max confidence (Hendrycks & Gimpel, 2016), predictive entropy, and confidence reduction prompting. Malinin & Gales (2021) uses token-entropy as a measure of uncertainty in auto-regressive models and Kuhn et al. (2023) leverages linguistic invariances via semantic entropy.

White box methods use uncertainty-aware fine-tuning to produce better-calibrated models. Chuang et al. (2024) introduces Self-REF, a framework that leverages confidence tokens during fine-tuning to improve downstream routing. Krishnan et al. (2024) proposes an uncertainty-aware causal language modeling loss that captures the trade-off between accuracy and calibration. In contrast, our method calibrates the model so that correct predictions receive low uncertainty and incorrect ones high

uncertainty. We apply this uncertainty-aware model in a cascade inference system, where it improves overall performance. Prior work by Rawat et al. (2021) pre-partitions data into easy and hard examples, e.g., based on $\mathcal{M}_L$'s confidence, and trains $\mathcal{M}_S$ with explicit labels. We improve on this static partitioning by dynamically assigning examples during training based on $\mathcal{M}_S$'s current state.

3 The GATEKEEPER Loss

3.1 Overview & Setup

Our framework consists of a large, highly capable model $\mathcal{M}_L$ and a smaller, resource-efficient model $\mathcal{M}_S$. We assume that $S \in \mathbb{N}$ and $L \in \mathbb{N}$ represent the parameter count of each model with $S \ll L$. Both models can either function as classifiers (i.e., $\mathcal{M} : \mathbb{R}^D \to [C]$ with $\mathbb{R}^D$ denoting the input space and C the number of total classes), or (multi-modal) sequence models (i.e., $\mathcal{M} : \mathbb{R}^D \to [V]^T$ where V is the vocabulary and T is the sequence length). We include experiments on all of these model classes in Section 4. Furthermore, we do not require a shared model family to be deployed on both $\mathcal{M}_S$ and $\mathcal{M}_L$; for example, $\mathcal{M}_S$ could be a custom convolutional neural network optimized for efficient inference and $\mathcal{M}_L$ a vision transformer (Dosovitskiy, 2020). The primary objective is to design a deferral mechanism that enables $\mathcal{M}_S$ to decide when to return its predictions without the assistance of $\mathcal{M}_L$ and when to instead defer to it. We assume that $\mathcal{M}_L$ is either outside of our control (e.g., an API endpoint) or too costly to modify, and that only $\mathcal{M}_S$ is subject to adaptation.

Deferral decisions are made using signals derived from the small model $\mathcal{M}_S$ as this approach is typically more cost-effective than employing a separate routing mechanism (Teerapittayanon et al., 2016). Approaches that involve querying the large model $\mathcal{M}_L$ to assist in making deferral decisions at test time are excluded from our setup. Such methods—common in domains like LLMs—are counterproductive to our goal since querying $\mathcal{M}_L$ defeats the purpose of making a deferral decision in the first place. Examples of these inapplicable methods include collaborative LLM frameworks (Mielke et al., 2022) and techniques that rely on semantic entropy for uncertainty estimation (Kuhn et al., 2023). As part of our setup, we assume that $\mathcal{M}_L$ dominates $\mathcal{M}_S$ as per the following assumption.

Dominance Assumption. Let $\mathcal{D}$ denote the target deployment distribution defined over covariates $\mathcal{X}$ and labels $\mathcal{Y}$. We assume that $\mathcal{M}_L$ dominates the $\mathcal{M}_S$ with high probability under $\mathcal{D}$; formally,

$$\Pr_{(x,y)\sim\mathcal{D}}\big[\mathcal{M}_L(x) \neq y \wedge \mathcal{M}_S(x) = y\big] \leq \delta, \tag{1}$$

with $\delta \ll 1$. This "almost-always" dominance, supported by scaling-law trends (Kaplan et al., 2020), implies that deferring from $\mathcal{M}_S$ to $\mathcal{M}_L$ cannot hurt accuracy in expectation, while still allowing rare counter-examples where the small model outperforms the large model. Note that we empirically observe $\delta = 0$ across all tasks considered in this work, meaning that $\mathcal{M}_L$ strictly dominates $\mathcal{M}_S$.

As discussed in Section 2, the choice of deferral strategy often depends on the level of access available to $\mathcal{M}_S$. We assume white box access with full access to $\mathcal{M}_S$'s internals. As such, deferral mechanisms can be directly integrated into the model's architecture and parameters. This involves fine-tuning $\mathcal{M}_S$ to predict deferral decisions or to incorporate rejection mechanisms within its predictive process. Our work falls into this category as it proposes a new loss function to fine-tune $\mathcal{M}_S$.

Our goal is to train a small model that can effectively distinguish between correct and incorrect predictions. While many past works have considered the question of whether it is possible to find proxy measures for prediction correctness, the central question we ask is:

Can we optimize the small model to separate correct from incorrect predictions?

We show that this is indeed achievable through a carefully designed fine-tuning stage that does not require any architectural modifications. This ensures that the ability to separate correct from incorrect decisions is integrated seamlessly into $\mathcal{M}_S$'s existing structure.

3.2 Confidence-Tuning for Deferral

Prerequisite: Standard Training. We begin with an $\mathcal{M}_S$ that has already been trained on the tasks it is intended to perform upon deployment. However, due to its limited capacity, $\mathcal{M}_S$ cannot achieve

the performance levels of $\mathcal{M}_L$. Importantly, we make no assumptions about the training process of $\mathcal{M}_S$ —whether it was trained from scratch without supervision from an external model or with the help of soft labels through a distillation approach.

Stage 1 (Finetuning): Correctness-Aware Finetuning with GATEKEEPER. Next, we introduce a correctness-aware loss, dubbed GATEKEEPER, to fine-tune $\mathcal{M}_S$ for improved confidence calibration. Specifically, the model is trained to make correct predictions with high confidence while reducing the confidence of incorrect predictions (see Figure 2). This loss can either rely on true labels or utilize the outputs of $\mathcal{M}_L$ with soft probabilities as targets.

In its canonical form, GATEKEEPER is defined as a hybrid loss $\mathcal{L} = \alpha\mathcal{L}_{\text{corr}} + (1-\alpha)\mathcal{L}_{\text{incorr}}$ with

$$\mathcal{L}_{\text{corr}} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}\{y_i = \hat{y}_i\}\text{CE}(p_i(\mathbf{x}_i), y_i) \qquad \mathcal{L}_{\text{incorr}} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}\{y_i \neq \hat{y}_i\}\text{KL}\left(p_i(\mathbf{x}_i) \parallel \mathcal{U}\right). \quad (2)$$

Here, y_i and $\hat{y}_i$ are the true and predicted labels for $\mathbf{x}_i$, respectively, p_i is the predicted probability distribution of $\mathcal{M}_S$ over classes, $\mathcal{U}$ represents the uniform distribution over all classes, N denotes the number samples in the current batch, $\alpha \in (0, 1)$ is a tunable hyperparameter controlling the emphasis between correct and incorrect predictions, and the cross-entropy function and KL divergence are defined as $\text{CE}(p, y) = -\sum_c y_c \log p_c$ and $\text{KL}(p \parallel q) = \sum_c p_c \log(\frac{p_c}{q_c})$, respectively. We note that a similar loss has previously been proposed in Outlier Exposure (OE) (Hendrycks et al., 2018) for out-of-distribution (OOD) sample detection. Here, the goal is to make sure that OOD examples are assigned low confidence scores by tuning the confidence on an auxiliary outlier dataset. However, to the best of our knowledge, this idea has not previously been used to improve deferral performance of a smaller model in a cascading chain.

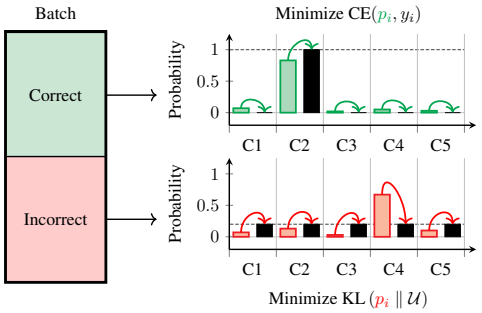

Figure 2: **GATEKEEPER Overview**: We want correctly predicted samples to maintain their current prediction by ensuring that cross entropy is decreased. At the same time, we want incorrectly predicted samples to yield a uniform confidence across classes, leading to a low overall confidence score (high predictive entropy).

We emphasize that the trade-off parameter α plays a critical role as part of this optimization setup as it directly influences model utility and deferral performance. A lower value of α emphasizes reducing confidence in incorrect predictions by pushing them closer to the uniform distribution, making the model more cautious in regions where it may make mistakes. Conversely, a higher value of α encourages the model to increase its confidence on correct predictions, sharpening its decision boundaries and enhancing accuracy where it is already performing well. Thus, α serves as a crucial hyperparameter that balances the trade-off between improving calibration by mitigating overconfidence in errors and reinforcing confidence in accurate classifications. By appropriately tuning α, practitioners can control the model's behavior to achieve a desired balance between reliability in uncertain regions and decisiveness in confident predictions, tailored to the specific requirements of their application.

We further generalize this loss to token-based models (e.g., LMs and VLMs) where

$$\mathcal{L}_{\text{corr}} = \frac{1}{N}\sum_{\substack{i=1 \\ t=1}}^{N,\,T} \mathbb{1}\{y_{i,t} = \hat{y}_{i,t}\}\text{CE}(p_{i,t}(\mathbf{x}_i), y_{i,t}) \qquad \mathcal{L}_{\text{incorr}} = \frac{1}{N}\sum_{\substack{i=1 \\ t=1}}^{N,\,T} \mathbb{1}\{y_{i,t} \neq \hat{y}_{i,t}\}\text{KL}\left(p_{i,t}(\mathbf{x}_i) \parallel \mathcal{U}\right). \quad (3)$$

Here, $y_{i,t}$ and $\hat{y}_{i,t}$ denote the true and predicted tokens at position t for sample i, $p_{i,t}$ is the predicted token distribution at position t for sample i, and T is the sequence length for the token-based model. The token-level loss ensures that correct token predictions are made confidently while incorrect tokens are assigned smaller confidences.

Practical Computation of the Gatekeeper Loss. We evaluate Gatekeeper *once per mini-batch* within the standard training loop—no auxiliary passes or data-set re-shuffling are required. Given a mini-batch $B = \{(x_i, y_i)\}_{i=1}^{N}$, we perform a single forward pass through $\mathcal{M}_S$. This allows us to obtain (i) a logit vector $z_i = f_\theta(\mathbf{x}_i)$, and (ii) predicted labels $\hat{y}_i = \arg\max_c z_{i,c}$. Two

binary masks $m_i^{\text{corr}} = I\{y_i = \hat{y}_i\}$ and $m_i^{\text{incorr}} = \neg m_i^{\text{corr}}$ are computed *on-the-fly*. The hybrid loss is then assembled in a fully vectorized manner with components $\mathcal{L}_{\text{corr}} = \frac{1}{N}\sum_i m_i^{\text{corr}}\,\text{CE}(p_i, y_i)$ and $\mathcal{L}_{\text{incorr}} = \frac{1}{N}\sum_i m_i^{\text{incorr}}\,\text{KL}(p_i||\mathcal{U})$ where $p_i = \text{softmax}(z_i)$. Because both masks and losses are computed inside the same tensor graph, back-propagation incurs only the cost of $\mathcal{O}(N \times C)$ element-wise operations—identical to a vanilla cross-entropy step. This single-pass design keeps the computational overhead negligible while guaranteeing that in the full loss $\mathcal{L} = \alpha\mathcal{L}_{\text{corr}} + (1-\alpha)\mathcal{L}_{\text{incorr}}$ every sample contributes to either $\mathcal{L}_{\text{corr}}$ or $\mathcal{L}_{\text{incorr}}$ in the same optimization step.

Stage 2 (Inference): Confidence Computation & Thresholding. After fine-tuning $\mathcal{M}_S$ with GATEKEEPER, we apply standard confidence- and entropy-based techniques for model uncertainty to obtain a deferral signal. We use the selective prediction framework to determine whether a query point $\mathbf{x} \in \mathbb{R}^D$ should be accepted by $\mathcal{M}_S$ or routed to $\mathcal{M}_L$. Selective prediction alters the model inference stage by introducing a deferral state through a *gating mechanism* (El-Yaniv & Wiener, 2010). At its core, this mechanism relies on a deferral function $g : \mathbb{R}^D \to \mathbb{R}$ which determines if $\mathcal{M}_S$ should output a prediction for a sample $\mathbf{x}$ or defer to $\mathcal{M}_L$. Given a targeted acceptance threshold τ, the resulting predictive model can be summarized as:

$$(\mathcal{M}_S, \mathcal{M}_L, g)(\mathbf{x}) = \begin{cases} \mathcal{M}_S(\mathbf{x}) & g(\mathbf{x}) \geq \tau \\ \mathcal{M}_L(\mathbf{x}) & \text{otherwise.} \end{cases} \tag{4}$$

Classification Models (Max Softmax). Let $\mathcal{M}_S$ produce a categorical distribution $\{p(y = c \mid \mathbf{x})\}_{c=1}^{C}$ over C classes. Then we define the gating function as

$$g_{\text{CL}}(\mathbf{x}) \;=\; \max_{1 \leq c \leq C} \; p\big(y = c \,\big|\, \mathbf{x}\big). \tag{5}$$

Token-based Models (Negative Predictive Entropy). Let $\mathcal{M}_S$ produce a sequence of categorical distributions $\{p(y_t = c \mid \mathbf{x})\}_{c=1}^{C}$ for each token index $t \in T$. Then we define the gating function as

$$g_{\text{NENT}}(\mathbf{x}) = \; \frac{1}{T}\sum_{t=1}^{T}\sum_{c=1}^{C} p\big(y_t = c \,\big|\, \mathbf{x}\big) \; \log p\big(y_t = c \,\big|\, \mathbf{x}\big), \tag{6}$$

where $y_t \in [C]$ is the predicted token at time step t, $p(y_t = c \mid \mathbf{x})$ is the (conditional) probability of token k at step t, and T is the total number of token positions for the sequence. Across both model classes, higher values of g_{CL} or g_{NENT} indicate higher prediction confidence (i.e., lower uncertainty).

4 Experiments

In this section, we detail the experiments used to evaluate the effectiveness of GATEKEEPER across three distinct model classes: encoder-only classification models, decoder-only language models, and encoder-decoder vision-language models. Each setup involves a cascade where a smaller model can defer inputs to a larger, more capable model. Our method and all competing baselines are applied on top of the same training/fine-tuning protocol used for obtaining the initial $\mathcal{M}_S$ and $\mathcal{M}_L$ models.

4.1 Encoder-only Setup (Classification Models)

We comprehensively assess the performance of GATEKEEPER across different model architectures and task types, starting with image classification. We train both a large model and a small model on the following datasets: CIFAR-10/100 (Krizhevsky et al., 2009), Food-101 (Bossard et al., 2014), and TinyImageNet200 (Le & Yang, 2015). For both CIFAR datasets we use a ResNet-18 (He et al., 2016) as $\mathcal{M}_L$ and a custom CNN as $\mathcal{M}_S$. For Food-101 and TinyImageNet200 we instead use a ResNet-50 (He et al., 2016) as $\mathcal{M}_L$ and a Mobilenet V3 Small (Howard et al., 2019) as $\mathcal{M}_S$, where the latter is trained using knowledge distillation from the big model.

Evaluation Metrics. We measure the performance of GATEKEEPER and the resulting deferral function $g(\cdot)$ using the following performance metrics (see example in Figure 3 for an overview):

1. The **Distributional Overlap of Confidences of Correct and Incorrect Predictions** s_o is defined as the integral of the minimum of the probability density functions (PDFs) of confidence scores

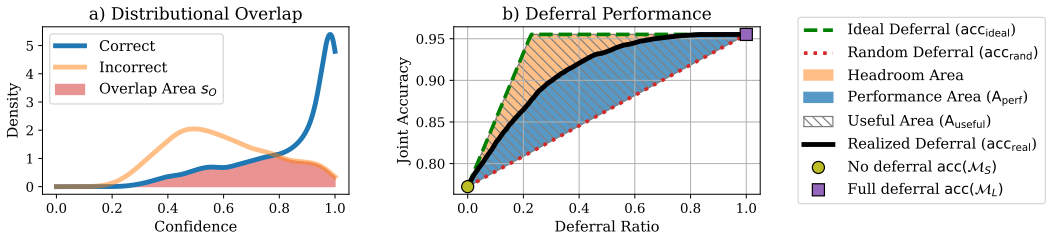

Figure 3: **Performance metrics overview**: **(a)** Distributional Overlap s_o: the densities of confidence scores for correctly (green) and incorrectly classified (red) samples, with the overlap area shaded in blue. Smaller values are better (↓). **(b)** Deferral Performance s_d: how joint accuracy between $\mathcal{M}_S$ and $\mathcal{M}_L$ varies with deferral ratio, showing random (red), ideal (green), and realized (black) deferral strategies. The blue region shows the realized performance gain, the hatched portion represents the range of useful deferral functions, and the green region indicates the potential headroom over the realized deferral. Larger values are better (↑).

for correctly classified samples, $\hat{p}_{\text{corr}}(c)$, and incorrectly classified samples, $\hat{p}_{\text{incorr}}(c)$ (see Figure 3a). Formally, given the confidence sets $\mathcal{C}_{\text{corr}}$ and $\mathcal{C}_{\text{incorr}}$, the overlap s_o is computed as

$$s_o = \int_0^1 \min\left\{\hat{p}_{\text{corr}}(c),\ \hat{p}_{\text{incorr}}(c)\right\}\,\mathrm{d}c, \tag{7}$$

where the PDFs are estimated using Kernel Density Estimation (KDE). If $s_o = 1$, then $\mathcal{M}_S$ cannot distinguish the confidence distribution of correct and incorrect predictions; if $s_o = 0$, then $\mathcal{M}_S$ can perfectly separate correct and incorrect predictions. Note that a related way of capturing the distributional separability is given by the Area Under the Receiver Operating Characteristic Curve (AUROC) which we discuss in Appendix C.3.

2. **Deferral Performance** s_d: To formally quantify how well $\mathcal{M}_S$ defers difficult inputs to $\mathcal{M}_L$, we examine the joint performance across all possible deferral ratios $r \in [0, 1]$, where r denotes the fraction of inputs sent to $\mathcal{M}_L$ based on a particular threshold τ (recall Equation (4)). Figure 3 b) illustrates how, as r increases from 0 to 1, the overall (joint) accuracy $\text{acc}(r)$ increases from the accuracy of $\mathcal{M}_S$ (yellow circle, no deferral) to the accuracy of $\mathcal{M}_L$ (purple square, full deferral). Useful deferral models are constrained to operate between random deferral (acc_{rand}, red dotted line) and ideal deferral ($\text{acc}_{\text{ideal}}$, green dashed line). The ideal deferral $\text{acc}_{\text{ideal}}$ corresponds to the oracle solution that perfectly defers examples misclassified by $\mathcal{M}_S$ and we discuss its exact functional form in Appendix B.3. We also define the realized deferral curve, acc_{real}, as the joint accuracy obtained under the learned deferral strategy $g(\cdot)$ employed by $\mathcal{M}_S$ and $\mathcal{M}_L$. The deferral performance metric s_d is then given as:

$$s_d = \frac{A_{\text{perf}}}{A_{\text{useful}}} = \frac{\int_0^1 \left(\text{acc}_{\text{real}}(r) - \text{acc}_{\text{rand}}(r)\right)\,\mathrm{d}r}{\int_0^1 \left(\text{acc}_{\text{ideal}}(r) - \text{acc}_{\text{rand}}(r)\right)\,\mathrm{d}r}. \tag{8}$$

This ratio quantifies the fraction of the potential improvement over random deferral that has been realized by the achieved deferral strategy. Note that $s_d = 1$ indicates perfect deferral, matching the ideal strategy, while an $s_d = 0$ implies no improvement over random deferral.

3. **Accuracy of the small model** $\text{acc}(\mathcal{M}_S)$: Finally, since GATEKEEPER emphasizes patterns for distinguishing correct/incorrect examples, the model is no longer encouraged to minimize the classification loss over the full population. As a result, improving on the correct/incorrect separation task can lead to changes in utility over the full data distribution. Hence, practically useful deferral methods need to balance both deferral performance and the accuracy of $\mathcal{M}_S$.

Results. Our main results are shown in Figure 4. We report performance for both a baseline model (an instance of $\mathcal{M}_S$ not trained with GATEKEEPER) and small models trained with GATEKEEPER at various α values. We also compare against Narasimhan et al. (2022), a common cascading baseline for supervised learning tasks. For all models, we compute deferral performance and correct/incorrect separation (center, left). The strongest performance occurs at low αs, where the model pushes outputs of incorrect examples closer to uniform. However, this comes at a cost: the small model's accuracy degrades at low αs (right), highlighting that the model effectively "unlearns" performance on harder examples to focus on easier ones. At larger αs, accuracy remains stable

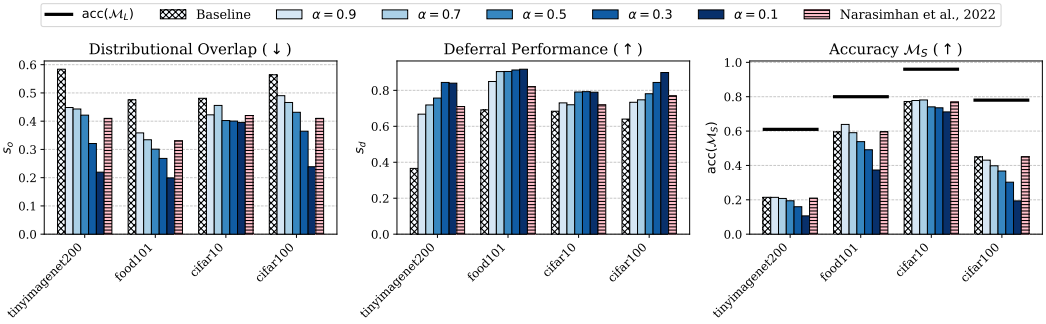

Figure 4: **Performance on image classification tasks**. We observe that lower levels of α lead to decreased distributional overlap between correct/incorrect predictions (left), increased deferral performance (center) and generally decreased performance over the full data distribution (right). These results support our conclusion that the small model $\mathcal{M}_S$ learns to refocus on easier subsets of the distribution while understanding more reliably when it should defer to the large model $\mathcal{M}_L$.

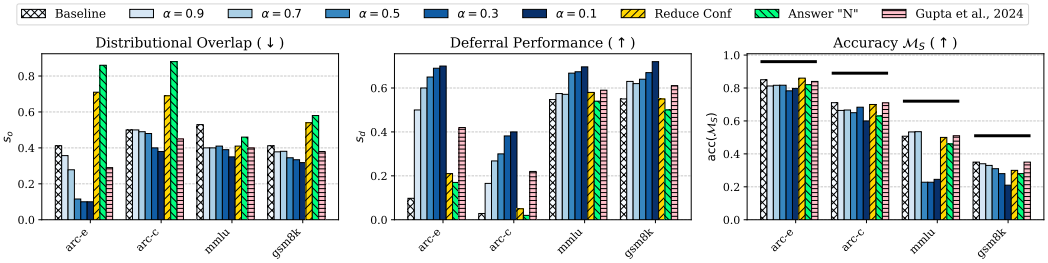

Figure 6: **Performance on language modeling tasks**. Similar as Figure 4. In addition to a non-tuned baseline, we also add an uncertainty prompting baseline, an Answer "N" option, as well as the post-hoc confidence calibration method from Gupta et al. (2024). We observe that GATEKEEPER outperforms other methods at lower levels of α.

or improves slightly, as training emphasizes already well-predicted points. While the baseline from Narasimhan et al. (2022) preserves model accuracy, it requires explicit estimation of the expert model's correctness—necessitating either an architectural change (e.g., an added prediction head) or a separate prediction network. In contrast, GATEKEEPER relies solely on confidence tuning of the small model, making it operationally easier to deploy. This approach not only simplifies implementation but also leads to improved deferral performance by producing more reliable uncertainty estimates through a stronger correlation with correctness.

This result highlights a critical trade-off which is directly controlled by α:

How strongly do we want to degrade model performance over the full data distribution in order to obtain a better deferral model?

We note that this compromise between raw model utility and deferral performance is not surprising and similar trade-offs exist in fairness (Dutta et al., 2020; Yaghini et al., 2023) and privacy (Abadi et al., 2016; Rabanser et al., 2023). We study this trade-off explicitly in Figure 5 showing (i) a clear negative correlation between deferral performance and the small model's accuracy; and (ii) a clear positive correlation between the overlap of correct/incorrect confidences and the accuracy of $\mathcal{M}_S$.

4.2 Decoder-Only Setup (LLMs)

In the decoder-only setup, we explore the application of LLMs. Our primary models of interest are the scalable LMs from the Gemma model class (GemmaTeam et al., 2024). We choose Gemma2B as $\mathcal{M}_S$ and Gemma7B as $\mathcal{M}_L$. Similar to the encoder-only setup, we employ smaller LMs as the initial classifiers to manage simpler next-token prediction tasks. The deferral strategy involves routing only those token sequences that exhibit high uncertainty—as determined by high predictive entropy—to the more powerful model $\mathcal{M}_L$.

Our experiments begin by taking the instruction-tuned checkpoints of Gemma2B and Gemma7B and fine-tuning both models on the training split of each dataset to ensure that the model (i) performs well on the task and (ii) is familiar with the desired response format. This step is performed using standard supervised fine-tuning. Next, we fine-tune $\mathcal{M}_S$ with GATEKEEPER on the same training split to reduce confidence on incorrect next-token predictions. Finally, we evaluate the model trained with GATEKEEPER on a validation split. The datasets used are ARC-e/c (Clark et al., 2018), MMLU (Hendrycks et al., 2020), and GSM8K (Cobbe et al., 2021). The evaluation metrics for our LLM experiments match those used in Section 4.1.

Results. We present our main results in Figure 6, comparing the baseline model's deferral and correct/incorrect separation ability to our fine-tuned model across different αs. We observe a similar trend as in the image classification setting: higher αs maintain raw prediction performance closer to the baseline but offer limited gains in separation, while lower αs improve deferral more substantially at the cost of overall accuracy. In addition to the baseline model (not fine-tuned with GATEKEEPER), we include results from Gupta et al. (2024) (an extension of Narasimhan et al. (2022) to token-based sequence models), as well as two uncertainty prompting baselines (described in Appendix C.2): (i) *Reduce Confidence*, which appends instructions to encourage the model to lower confidence when uncertain; and (ii) *Answer "N"*, which instructs the model to respond with "N" if uncertain. We find that GATEKEEPER outperforms Gupta et al. (2024) in terms of correct/incorrect sepration and deferral at the cost of overall utility. Consistent with prior findings from Kadavath et al. (2022), the prompting baselines do not reliably improve deferral.

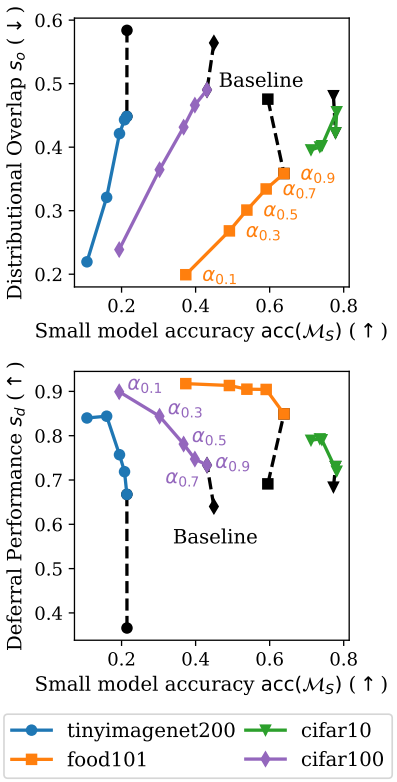

Figure 5: **Performance trade-off between small model accuracy acc($\mathcal{M}_S$) and deferral evaluation metrics**. The baseline model obtained without fine-tuning using GATEKEEPER is often the most accurate model over the full data distribution. With the introduction of GATEKEEPER we can improve distinguishability of correct/incorrect predictions (top) as well as deferral performance (bottom) at the expense of model utility. Successful cascading solutions in practice need to balance both maintaining high model accuracy and deferral performance.

4.3 Encoder-Decoder Setup (VLMs)

Finally, we examine models with both visual and textual processing capabilities, ideal for tasks requiring joint image understanding and language generation. We use the PaliGemma (Steiner et al., 2024) model family—encoder-decoder models designed for VL tasks such as image captioning, visual question answering, and descriptive image classification. The encoder processes input images into rich feature representations, while the decoder generates textual outputs. We use PaliGemma1B as $\mathcal{M}_S$ and PaliGemma7B as $\mathcal{M}_L$. Our deferral strategy runs the smaller VLM on all inputs and only defers to the more bigger 7B model when $\mathcal{M}_S$'s predictive entropy falls below a set threshold.

Similar to our experiments on LMs in Section 4.2, we employ two stages of fine-tuning. First, we take the instruction-tuned checkpoints of PaliGemma1B and PaliGemma7B and then fine-tune both models on the training split of a given dataset. Next, we fine-tune only $\mathcal{M}_S$ using GATEKEEPER before evaluating the model on a validation/test split of the dataset. The datasets we consider are two classification datasets (VQAv2 (Goyal et al., 2017), AI2D (Hiippala et al., 2021)) and two captioning datasets (Cococap (Lin et al., 2014), Screen2Words (Wang et al., 2021)). This allows us to evaluate GATEKEEPER in closed-form vision-language classification setups and open-form text generation.

Factuality Scoring. For classification tasks we apply our analysis in the same way as in Section 4.2. However, for captioning datasets we need to evaluate the quality of a caption generated by PaliGemma. To do that, we compute a factuality score which judges whether the generated caption is semantically coherent with respect to a reference caption using the Gemini LLM (GeminiTeam et al., 2023).

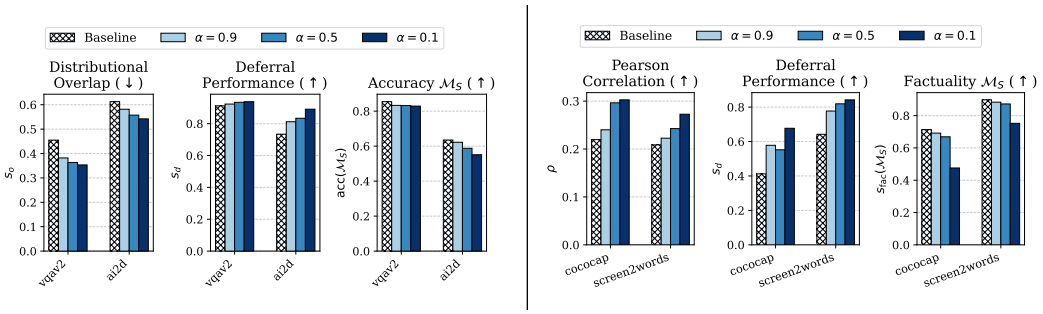

Figure 7: **Performance on VLM classification (left) and captioning tasks (right)**. Consistent with results in Figures 4 and 6, we see that smaller αs lead to improved deferral performance.

Specifically, the Gemini LLM is prompted with an instruction of the form: *"Are these captions semantically equivalent?"*, followed by both the candidate caption and the reference caption. The model then responds with either *"Yes"* or *"No"*. Finally, we compute the log-likelihood of each response and normalize it to a probability, reflecting the LLM's confidence in the captions being factually aligned. We detail this process in Appendix C.4 and denote the factuality score for input point $\mathbf{x}_i$ with candidate caption $\hat{\mathbf{y}}_i$ and ground truth caption $\mathbf{y}_i$ as $s_{\text{Fac}}(\hat{\mathbf{y}}_i, \mathbf{y}_i)$.

Measuring Correlation Between Factuality and Negative Predictive Entropy. Since the result of evaluating $s_{\text{Fac}}(\hat{\mathbf{y}}_i, \mathbf{y}_i)$ is no longer binary, our evaluation metrics which previously relied on accuracy cannot be used directly to evaluate deferral performance and the correct/incorrect entropy distribution separation. We address this issue by replacing the distributional overlap computation with the Pearson correlation $\rho(g_{\text{NENT}}(\mathbf{x}_i), s_{\text{Fac}}(\hat{\mathbf{y}}_i, \mathbf{y}_i))$ between the negative predictive entropy of a caption $g_{\text{NENT}}(\mathbf{x}_i)$ and its associated factuality score $s_{\text{Fac}}(\hat{\mathbf{y}}_i, \mathbf{y}_i))$. We also adapt our deferral performance metric from Equation (8) to rely on factuality measures instead of accuracy.

Results. We present our results in Figure 7, comparing the baseline model's deferral ability to our fine-tuned models across different αs. For classification (Figure 7, left), we observe the same trends as in previous classification and language modeling experiments. For captioning (Figure 7, right), GATEKEEPER increases the correlation between factuality and negative predictive entropy, enabling better deferral from $\mathcal{M}_S$ to $\mathcal{M}_L$ as α decreases. This shows that our method generalizes beyond closed-form classification to open-form sequence generation tasks. While we also benchmarked the prompting baselines from Section 4.2, PaliGemma did not return responses for the modified prompts—likely due to its rigid pretraining and prompting instructions (Beyer et al., 2024).

5 Conclusion

In this work we present a novel loss function called GATEKEEPER for improving confidence calibration in a cascade between a small local and a larger remote model. Our loss is architecture and task agnostic, making it flexibly applicable across a wide range of applications. Our results on encoder-only classification models, decoder-only language models, and encoder-decoder vision-language models demonstrate that our approach improves over standard confidence-based deferral rules and effectively leads the small model to unlearn how to handle complex queries in favor of easier ones.

Limitations. While our approach demonstrates promising results, several limitations remain. First, we assume that only the smaller model can be fine-tuned, whereas in some applications the larger model could also be adjusted to improve deferral. Second, in language settings, GATEKEEPER may be overly aggressive: multiple token sequences can convey the same meaning, so penalizing deviations based on exact tokens rather than semantics may be suboptimal. Ideally, deferral should account for semantic correctness rather than surface-level mismatches. Third, we did not extensively evaluate across diverse model families in LLM and VLM settings, though we did include such comparisons for classification tasks. Fourth, our evaluation focusses on a two-stage cascade consisting a small model and a big model. However, we are confident that GATEKEEPER can scale to multi-stage setups (see Appendix B.5). Finally, our use of a generative model (e.g., Gemini) to evaluate captioning introduces the possibility of erroneous judgments, as LLMs are themselves imperfect evaluators.

References

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pp. 308–318, 2016.

Abdar, M., Pourpanah, F., Hussain, S., Rezazadegan, D., Liu, L., Ghavamzadeh, M., Fieguth, P., Cao, X., Khosravi, A., Acharya, U. R., et al. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information fusion*, 76:243–297, 2021.

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Anthropic. Models overview - anthropic. `https://docs.anthropic.com/en/docs/build-with-claude/citations`, 2024. [Online; accessed 24-January-2025].

Azaria, A. and Mitchell, T. The internal state of an llm knows when it's lying. *arXiv preprint arXiv:2304.13734*, 2023.

Beyer, L., Steiner, A., Pinto, A. S., Kolesnikov, A., Wang, X., Salz, D., Neumann, M., Alabdulmohsin, I., Tschannen, M., Bugliarello, E., et al. Paligemma: A versatile 3b vlm for transfer. *arXiv preprint arXiv:2407.07726*, 2024.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. Weight uncertainty in neural network. In *International conference on machine learning*, pp. 1613–1622. PMLR, 2015.

Bolukbasi, T., Wang, J., Dekel, O., and Saligrama, V. Adaptive neural networks for fast test-time prediction. 2017a.

Bolukbasi, T., Wang, J., Dekel, O., and Saligrama, V. Adaptive neural networks for efficient inference. In *International conference on machine learning*, pp. 527–536. PMLR, 2017b.

Bossard, L., Guillaumin, M., and Van Gool, L. Food-101 – mining discriminative components with random forests. In *European Conference on Computer Vision*, 2014.

Chen, B., Zhu, M., Dolan-Gavitt, B., Shafique, M., and Garg, S. Model cascading for code: Reducing inference costs with model cascading for LLM based code generation, 2024a. URL `https://arxiv.org/abs/2405.15842`.

Chen, L., Zaharia, M., and Zou, J. FrugalGPT: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*, 2023a.

Chen, L., Zaharia, M., and Zou, J. Frugalgpt: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*, 2023b.

Chen, Z., Yang, X., Lin, J., Sun, C., Chang, K., and Huang, J. Cascade speculative drafting for even faster LLM inference. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024b. URL `https://openreview.net/forum?id=lZY9u0ijP7`.

Cheng, H., Zhang, M., and Shi, J. Q. A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):10558–10578, 2024. doi: 10.1109/TPAMI.2024.3447085.

Chuang, Y.-N., Zhou, H., Sarma, P. K., Gopalan, P., Boccio, J., Bolouki, S., and Hu, X. Learning to route with confidence tokens, 2024. URL `https://arxiv.org/abs/2410.13284`.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Ding, D., Mallick, A., Wang, C., Sim, R., Mukherjee, S., Ruhle, V., Lakshmanan, L. V., and Awadallah, A. H. Hybrid llm: Cost-efficient and quality-aware query routing. *arXiv preprint arXiv:2404.14618*, 2024.

Dohan, D., Xu, W., Lewkowycz, A., Austin, J., Bieber, D., Lopes, R. G., Wu, Y., Michalewski, H., Saurous, R. A., Sohl-Dickstein, J., et al. Language model cascades. *arXiv preprint arXiv:2207.10342*, 2022.

Dosovitskiy, A. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Dutta, S., Wei, D., Yueksel, H., Chen, P.-Y., Liu, S., and Varshney, K. Is there a trade-off between fairness and accuracy? a perspective using mismatched hypothesis testing. In *International conference on machine learning*, pp. 2803–2813. PMLR, 2020.

El-Yaniv, R. and Wiener, Y. On the foundations of noise-free selective classification. *Journal of Machine Learning Research*, 11(53):1605–1641, 2010. URL `http://jmlr.org/papers/v11/el-yaniv10a.html`.

Enomoro, S. and Eda, T. Learning to cascade: Confidence calibration for improving the accuracy and computational cost of cascade inference systems. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(8):7331–7339, May 2021. doi: 10.1609/aaai.v35i8.16900. URL `https://ojs.aaai.org/index.php/AAAI/article/view/16900`.

Gallotta, R., Todd, G., Zammit, M., Earle, S., Liapis, A., Togelius, J., and Yannakakis, G. N. Large language models and games: A survey and roadmap. *arXiv preprint arXiv:2402.18659*, 2024.

GeminiTeam, G., Anil, R., Borgeaud, S., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A. M., Hauth, A., Millican, K., et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

GemmaTeam, G., Riviere, M., Pathak, S., Sessa, P. G., Hardin, C., Bhupatiraju, S., Hussenot, L., Mesnard, T., Shahriari, B., Ramé, A., et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

Geng, J., Cai, F., Wang, Y., Koeppl, H., Nakov, P., and Gurevych, I. A survey of confidence estimation and calibration in large language models. *arXiv preprint arXiv:2311.08298*, 2023.

Gou, Z., Shao, Z., Gong, Y., Shen, Y., Yang, Y., Duan, N., and Chen, W. Critic: Large language models can self-correct with tool-interactive critiquing. *arXiv preprint arXiv:2305.11738*, 2023.

Goyal, Y., Khot, T., Summers-Stay, D., Batra, D., and Parikh, D. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6904–6913, 2017.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. In *International conference on machine learning*, pp. 1321–1330. PMLR, 2017.

Gupta, N., Narasimhan, H., Jitkrittum, W., Rawat, A. S., Menon, A. K., and Kumar, S. Language model cascades: Token-level uncertainty and beyond. 2024. URL `https://openreview.net/forum?id=KgaBScZ4VI`.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hendrycks, D. and Gimpel, K. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

Hendrycks, D., Mazeika, M., and Dietterich, T. Deep anomaly detection with outlier exposure. *arXiv preprint arXiv:1812.04606*, 2018.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Hiippala, T., Alikhani, M., Haverinen, J., Kalliokoski, T., Logacheva, E., Orekhova, S., Tuomainen, A., Stone, M., and Bateman, J. A. Ai2d-rst: A multimodal corpus of 1000 primary school science diagrams. *Language Resources and Evaluation*, 55:661–688, 2021.

Hoefler, T., Alistarh, D., Ben-Nun, T., Dryden, N., and Peste, A. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22(241):1–124, 2021.

Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1314–1324, 2019.

Jitkrittum, W., Gupta, N., Menon, A. K., Narasimhan, H., Rawat, A., and Kumar, S. When does confidence-based cascade deferral suffice? In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 9891–9906. Curran Associates, Inc., 2023.

Kadavath, S., Conerly, T., Askell, A., Henighan, T., Drain, D., Perez, E., Schiefer, N., Hatfield-Dodds, Z., DasSarma, N., Tran-Johnson, E., et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Kolawole, S., Dennis, D., Talwalkar, A., and Smith, V. Agreement-based cascading for efficient inference, 2024. URL https://arxiv.org/abs/2407.02348.

Krishnan, R., Khanna, P., and Tickoo, O. Enhancing trust in large language models with uncertainty-aware fine-tuning, 2024. URL https://arxiv.org/abs/2412.02904.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Kuhn, L., Gal, Y., and Farquhar, S. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=VD-AYtP0dve.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

Le, Y. and Yang, X. S. Tiny imagenet visual recognition challenge. 2015.

Leviathan, Y. Looking back at speculative decoding, 2024. URL https://research.google/blog/looking-back-at-speculative-decoding/.

Leviathan, Y., Kalman, M., and Matias, Y. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pp. 19274–19286. PMLR, 2023.

Li, Y., Wang, S., Ding, H., and Chen, H. Large language models in finance: A survey. In *Proceedings of the fourth ACM international conference on AI in finance*, pp. 374–382, 2023.

Lin, S., Hilton, J., and Evans, O. Teaching models to express their uncertainty in words. *arXiv preprint arXiv:2205.14334*, 2022.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pp. 740–755. Springer, 2014.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017.

Liu, L., Pan, Y., Li, X., and Chen, G. Uncertainty estimation and quantification for llms: A simple supervised approach. *arXiv preprint arXiv:2404.15993*, 2024.

Ma, X., Fang, G., and Wang, X. LLM-pruner: On the structural pruning of large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=J8Ajf9WfXP.

Mahaut, M., Aina, L., Czarnowska, P., Hardalov, M., Müller, T., and Màrquez, L. Factual confidence of llms: On reliability and robustness of current estimators. *arXiv preprint arXiv:2406.13415*, 2024.

Malinin, A. and Gales, M. Uncertainty estimation in autoregressive structured prediction. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=jN5y-zb5Q7m.

Mamou, J., Pereg, O., Wasserblat, M., and Schwartz, R. Tangobert: Reducing inference cost by using cascaded architecture, 2022. URL https://arxiv.org/abs/2204.06271.

Mielke, S. J., Szlam, A., Dinan, E., and Boureau, Y.-L. Reducing conversational agents' overconfidence through linguistic calibration. *Transactions of the Association for Computational Linguistics*, 10:857–872, 2022.

Narasimhan, H., Jitkrittum, W., Menon, A. K., Rawat, A., and Kumar, S. Post-hoc estimators for learning to defer to an expert. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 29292–29304. Curran Associates, Inc., 2022.

Narasimhan, H., Jitkrittum, W., Rawat, A. S., Kim, S., Gupta, N., Menon, A. K., and Kumar, S. Faster cascades via speculative decoding. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=vo9t20wsmd.

Nazi, Z. A. and Peng, W. Large language models in healthcare and medical domain: A review. *Informatics*, 11(3), 2024. ISSN 2227-9709. doi: 10.3390/informatics11030057. URL https://www.mdpi.com/2227-9709/11/3/57.

Nie, L., Ding, Z., Hu, E., Jermaine, C., and Chaudhuri, S. Online cascade learning for efficient inference over streams. *arXiv preprint arXiv:2402.04513*, 2024.

Pereyra, G., Tucker, G., Chorowski, J., Kaiser, Ł., and Hinton, G. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*, 2017.

Pope, R., Douglas, S., Chowdhery, A., Devlin, J., Bradbury, J., Heek, J., Xiao, K., Agrawal, S., and Dean, J. Efficiently scaling transformer inference. In Song, D., Carbin, M., and Chen, T. (eds.), *Proceedings of Machine Learning and Systems*, volume 5, pp. 606–624. Curan, 2023.

Rabanser, S., Thudi, A., Guha Thakurta, A., Dvijotham, K., and Papernot, N. Training private models that know what they don't know. *Advances in Neural Information Processing Systems*, 36: 53711–53727, 2023.

Rawat, A. S., Zaheer, M., Menon, A. K., Ahmed, A., and Kumar, S. When in doubt, summon the titans: Efficient inference with large models, 2021. URL https://arxiv.org/abs/2110.10305.

Shnitzer, T., Ou, A., Silva, M., Soule, K., Sun, Y., Solomon, J., Thompson, N., and Yurochkin, M. Large language model routing with benchmark datasets. *arXiv preprint arXiv:2309.15789*, 2023.

Shrivastava, V., Liang, P., and Kumar, A. Llamas know what gpts don't show: Surrogate models for confidence estimation. *arXiv preprint arXiv:2311.08877*, 2023.

Steiner, A., Pinto, A. S., Tschannen, M., Keysers, D., Wang, X., Bitton, Y., Gritsenko, A., Minderer, M., Sherbondy, A., Long, S., et al. Paligemma 2: A family of versatile vlms for transfer. *arXiv preprint arXiv:2412.03555*, 2024.

Teerapittayanon, S., McDanel, B., and Kung, H.-T. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd international conference on pattern recognition (ICPR)*, pp. 2464–2469. IEEE, 2016.

Trapeznikov, K. and Saligrama, V. Supervised sequential classification under budget constraints. In Carvalho, C. M. and Ravikumar, P. (eds.), *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, volume 31 of *Proceedings of Machine Learning Research*, pp. 581–589, Scottsdale, Arizona, USA, 29 Apr–01 May 2013. PMLR. URL https://proceedings.mlr.press/v31/trapeznikov13a.html.

Varshney, N. and Baral, C. Model cascading: Towards jointly improving efficiency and accuracy of NLP systems. In Goldberg, Y., Kozareva, Z., and Zhang, Y. (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 11007–11021, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.756. URL https://aclanthology.org/2022.emnlp-main.756.

Viola, P. and Jones, M. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pp. I–I, 2001. doi: 10.1109/CVPR.2001.990517.

Wang, B., Li, G., Zhou, X., Chen, Z., Grossman, T., and Li, Y. Screen2words: Automatic mobile ui summarization with multimodal learning. In *The 34th Annual ACM Symposium on User Interface Software and Technology*, pp. 498–510, 2021.

Wang, C., Augenstein, S., Rush, K., Jitkrittum, W., Narasimhan, H., Rawat, A. S., Menon, A. K., and Go, A. Cascade-aware training of language models, 2024a. URL https://arxiv.org/abs/2406.00060.

Wang, S., Xu, T., Li, H., Zhang, C., Liang, J., Tang, J., Yu, P. S., and Wen, Q. Large language models for education: A survey and outlook. *arXiv preprint arXiv:2403.18105*, 2024b.

Wang, X., Luo, Y., Crankshaw, D., Tumanov, A., and Gonzalez, J. E. Idk cascades: Fast deep learning by learning not to overthink. In *Conference on Uncertainty in Artificial Intelligence*, 2017.

Xiong, M., Hu, Z., Lu, X., LI, Y., Fu, J., He, J., and Hooi, B. Can LLMs express their uncertainty? an empirical evaluation of confidence elicitation in LLMs. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=gjeQKFxFpZ.

Yaghini, M., Liu, P., Boenisch, F., and Papernot, N. Learning to walk impartially on the pareto frontier of fairness, privacy, and utility. 2023.

Yang, C., Zhu, Y., Lu, W., Wang, Y., Chen, Q., Gao, C., Yan, B., and Chen, Y. Survey on knowledge distillation for large language models: Methods, evaluation, and application. *ACM Trans. Intell. Syst. Technol.*, October 2024. ISSN 2157-6904. doi: 10.1145/3699518. URL https://doi.org/10.1145/3699518. Just Accepted.

Yue, M., Zhao, J., Zhang, M., Du, L., and Yao, Z. Large language model cascades with mixture of thought representations for cost-efficient reasoning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=6okaSfANzh.

A Broader Impact

This work contributes to the responsible and efficient deployment of machine learning systems by improving the decision-making capabilities of smaller, local models in model cascade architectures. By introducing a loss function that calibrates model confidence with respect to correctness, our approach enhances both the performance and transparency of automated systems that must decide when to act autonomously and when to defer to a more capable model. This design can improve the accessibility and sustainability of machine learning applications by reducing reliance on large, energy-intensive models—particularly important in low-resource environments or edge computing.

At the same time, the ability to fine-tune smaller models to strategically abstain from uncertain predictions raises important considerations for fairness and accountability. In high-stakes applications such as healthcare or finance, improper tuning of the deferral threshold—or uncalibrated confidence estimates—could lead to the systematic denial of service or misallocation of computational resources. Care must be taken to ensure that such systems are thoroughly evaluated not only for average performance but also for differential performance across subgroups. Moreover, the use of large models as fallback decision-makers assumes their correctness, which may not always hold, especially in underrepresented domains. We therefore encourage developers and practitioners to accompany deployments of cascade-based systems with rigorous audits of fairness, reliability, and alignment with human values.

B Additional Background

B.1 Related Work

B.1.1 LLM Routing

Ding et al. (2024) propose a hybrid LLM inference pipeline that routes each query either to a small on-device model or a larger high-quality model based on the query's predicted difficulty and a tunable quality threshold. This cost-aware router allows dynamically trading off accuracy for efficiency, enabling up to a 40% reduction in expensive model calls without degrading answer quality. Similarly, Shnitzer et al. (2023) present a method to select the best model from a pool of pre-trained LLMs for each input by learning a "router" on many benchmark tasks. Without requiring labeled examples from the new target task, their approach uses existing datasets to train input-based model selectors, which consistently outperform always using the single best LLM for all queries.

B.1.2 Model Cascade Learning

Nie et al. (2024) introduce an online cascade-learning framework where lightweight models are incrementally trained to imitate a powerful LLM's decisions on a data stream, deferring to the LLM only when necessary. They cast cascade construction as an imitation-learning problem with theoretical no-regret guarantees, achieving LLM-level accuracy while cutting inference cost by up to 90% and maintaining robustness to distribution shifts over time. Chen et al. (2023b) outline strategies for reducing LLM usage cost and present *FrugalGPT*, a cascade approach that learns to route queries through combinations of smaller or larger LLMs to balance cost and performance. Their experiments show that an adaptive use of multiple models can match the accuracy of the strongest individual LLM (e.g., GPT-4) with up to 98% cost savings. It can also slightly exceed GPT-4's accuracy at equal cost, highlighting the benefit of cascades that allocate queries to the most appropriate model for each input.

B.1.3 Confidence Calibration in LLMs

Jitkrittum et al. (2023) analyze the classical strategy of confidence-based deferral in model cascades, wherein a model hands off to a stronger model if its confidence is below a threshold, to determine when this simple strategy succeeds or breaks down. They derive the optimal deferral policy in theory and show that naïve confidence thresholds perform well in general but can fail when later models are specialists (only reliable on certain inputs), when there is label noise, or under distribution shift—scenarios where more sophisticated deferral criteria yield better performance. Geng et al. (2023) provide a comprehensive survey of methods for confidence estimation and calibration in LLM outputs. They review recent techniques to quantify uncertainty in large language model predictions, discuss challenges unique to LLMs, and highlight advancements that improve alignment between

a model's reported confidence and its actual accuracy across tasks. Azaria & Mitchell (2023) find evidence that an LLM's internal activations encode whether or not it is producing a truthful answer, even when the model's output is incorrect or fabricated. By training a classifier on the model's hidden state (without fine-tuning the LLM itself), they can often detect when the model is "lying" or unsure, suggesting that large models internally recognize their mistakes or uncertainty despite outwardly confident responses. Similarly, Liu et al. (2024) propose a supervised approach to LLM uncertainty quantification that leverages labeled examples and the model's hidden representations to predict the correctness of its answers. They show that incorporating features from the model's internal layers yields significantly improved uncertainty estimates and calibration across diverse tasks, with these gains transferring robustly to new domains. Notably, their method is easy to implement and can be adapted to different levels of model access (black-box vs. white-box), making it widely applicable.

B.1.4 Confidence Verbalization in LLMs

Lin et al. (2022) demonstrate that GPT-3 can be fine-tuned to output a calibrated verbal confidence (e.g., "I'm 90% sure") along with each answer. This model's stated confidence levels align well with its true correctness likelihood and remain fairly well-calibrated even under distribution shift, marking the first instance of an LLM explicitly expressing useful uncertainty estimates in natural language. Xiong et al. (2024) thoroughly evaluate black-box methods for eliciting an LLM's self-reported confidence through prompting and answer sampling. They find that current LLMs tend to verbalize overly high confidence (mirroring human overconfidence), but that carefully designed prompts, consistency checks across multiple sampled answers, and improved aggregation strategies can mitigate this issue. Moreover, larger models generally show better calibration and an improved ability to predict their own failures, though room for further improvement remains in making their expressed uncertainty truly reliable. Mielke et al. (2022) examine whether a conversational agent's expressed certainty corresponds to its actual knowledge, showing that off-the-shelf dialogue models are poorly "linguistically calibrated." They demonstrate that a model's likelihood of giving a correct answer can be estimated via an auxiliary model and used as a control signal to adjust the agent's responses. The resulting dialogue agent exhibits far less overconfident language when it is likely to be wrong, improving transparency about uncertainty in its answers. Finally, Mahaut et al. (2024) assess the reliability of various methods to estimate an LLM's *factual confidence* – the probability that its answer is correct – under both in-domain and paraphrased inputs. Through a rigorous evaluation on QA and fact-checking tasks, they conclude that the most trustworthy confidence scores come from model-introspective approaches (e.g., a trained probe on hidden states), albeit at the cost of requiring full model access and training data. They also highlight that an LLM's confidence can be unstable under meaning-preserving input variations (paraphrases), underscoring the need for more robust and stable confidence estimation techniques for factual correctness.

B.1.5 Speculative Decoding

As noted in the introduction and our related work section, there exists a connection between model cascading and speculative decoding. Speculative decoding is a technique to accelerate inference by pairing a small, fast "draft" model with a larger target model. The draft model generates multiple candidate tokens in parallel, and the large model then verifies them in a single forward pass by accepting valid tokens and rejecting others. This reduces the number of expensive calls to the large model, often yielding significant speedups without sacrificing output quality. We briefly describe how this line of work relates to our goal of deferral confidence tuning.

Speculative decoding accelerates every input by asking the large model to verify draft tokens that the small model proposes. As a result, the large model is still invoked on all inputs. Our cascade, by contrast, aims to avoid calling the large model on inputs the small model can already solve. Consequently, speculative decoding optimizes in-place token-level latency, while our method optimizes end-to-end compute and monetary cost at the request level. Because the two techniques improve fundamentally different bottlenecks we do not directly compare against this class of methods. However, we note that both cascading and speculative decoding can be composed: after our deferral gate decides to consult the large model, one could still decode that portion of the input speculatively. We direct the interested reader to Narasimhan et al. (2025) for an example of such hybrids.

B.1.6 Early Exiting

Early exit networks are models augmented with intermediate classifiers that allow predictions to be made before reaching the final layer. At inference time, the model can stop early on "easy" inputs while continuing deeper for "harder" ones, reducing computation without compromising much accuracy. This adaptive approach makes them well-suited for resource-constrained or latency-sensitive applications. As we discuss below, this class of approaches is, while related, ultimately still distinct from our goal for the following reasons:

1. Early–exiting aims to skip the remaining layers of a single network once an intermediate classifier is sufficiently confident, thereby offering a layer-level latency-accuracy trade-off within one model. Our work tackles the fundamentally different problem of model-level deferral between a small and a large model. The goal is to keep the large model entirely idle for easy requests and to invoke it only when needed. Hence our primary measure of success is joint accuracy versus cross-model compute budget, not marginal delay per layer inside a fixed backbone.
2. Early–exit usually presupposes white-box control over the entire network architecture so that branch classifiers can be inserted and jointly trained (e.g., BranchyNet (Teerapittayanon et al., 2016), Adaptive Neural Networks (Bolukbasi et al., 2017b)). Our cascade setting explicitly targets heterogeneous or API-based experts—large language models, vision–language models, and so forth—that cannot be modified. Early-exits cannot be applied in such black-box situations, whereas our GATEKEEPER fine-tuning remains feasible. Moreover, early-exiting has seen limited success for autoregressive sequence generation, where every token depends on the full hidden state; in contrast, our experiments span both classification and open-ended generation tasks.
3. Because early-exit and cascading operate at different granularities, they are complementary rather than competing techniques. One could in principle insert early exits inside the small model and still rely on calibrated deferral to the large model—yielding a three-level system (exit-1 $\rightarrow$ exit-2 $\rightarrow$ large model). Evaluating that combined design is a compelling direction for future engineering work, but it lies beyond the scope of our current study, whose contribution is a confidence-tuning loss independent of architectural changes.

B.2 Model Access Levels

In Figure 8, we show a schematic overview of different model access levels discussed in Section 2.

B.3 Ideal Deferral Curve

We present the functional form of the *ideal deferral* curve, denoted $\mathrm{acc}_{\mathrm{ideal}}(r)$, for a small (student) model $\mathcal{M}_S$ and a large (teacher) model $\mathcal{M}_L$. Recall that $r \in [0, 1]$ denotes the deferral ratio, i.e., the fraction of inputs that $\mathcal{M}_S$ "defers" to $\mathcal{M}_L$. Let $p_s = \mathrm{acc}(\mathcal{M}_S)$, and $p_l = \mathrm{acc}(\mathcal{M}_L)$ with $0 \leq p_s \leq p_l \leq 1$. Our goal is to describe the maximum achievable joint accuracy if exactly a fraction r of the data is deferred to the large model.

Intuition and Setup Since $\mathcal{M}_S$ achieves accuracy p_s, it misclassifies a fraction $(1 - p_s)$ of the inputs. In an *ideal* scenario, we defer exactly those inputs that $\mathcal{M}_S$ is going to misclassify. Because $\mathcal{M}_L$ is more accurate ($p_l \geq p_s$) every example misclassified by $\mathcal{M}_S$ benefits from being passed to $\mathcal{M}_L$.

- **Case 1:** $r \leq (1 - p_s)$.
  We can use our entire deferral "budget" r to cover only those inputs $\mathcal{M}_S$ would get wrong. Hence, deferring a fraction r of the data (all from $\mathcal{M}_S$'s mistakes) raises the overall accuracy by substituting $\mathcal{M}_S$'s errors with $\mathcal{M}_L$'s accuracy p_l on that fraction.
- **Case 2:** $r > (1 - p_s)$.
  We have enough capacity to defer *all* of $\mathcal{M}_S$'s mistakes, so the joint accuracy saturates at p_l. Deferring *additional* examples (which $\mathcal{M}_S$ would have classified correctly) will not improve the overall accuracy beyond p_l.

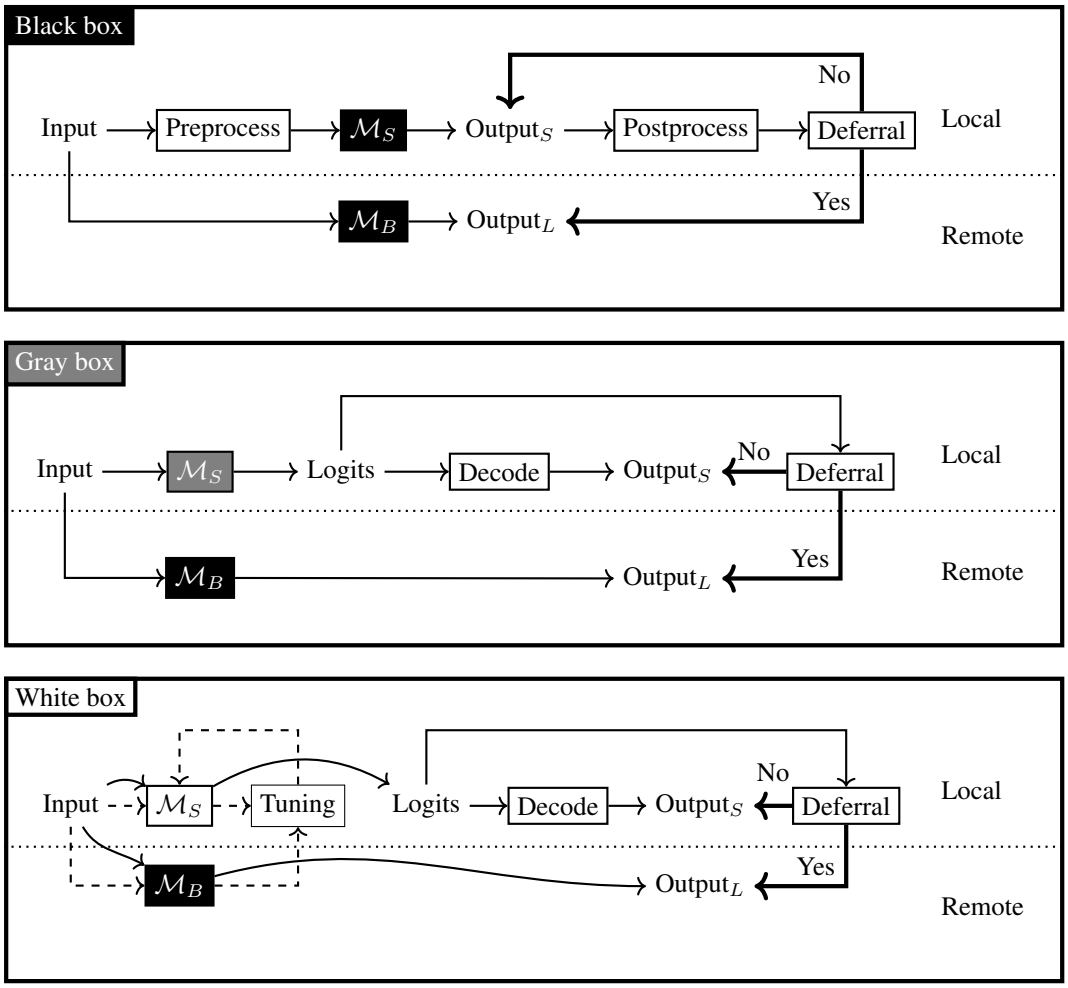

Figure 8: **An overview of different uncertainty quantification strategies depending on model access level**.

Piecewise Functional Form Thus, the *ideal deferral* curve can be expressed as:

$$\mathrm{acc}_{\mathrm{ideal}}(r) \;=\; \begin{cases} p_s + \dfrac{p_l - p_s}{1 - p_s}\, r, & 0 \;\le\; r \;\le\; (1 - p_s), \\ p_l, & (1 - p_s) \;<\; r \;\le\; 1. \end{cases} \tag{9}$$

When $0 \le r \le (1 - p_s)$, the overall accuracy grows linearly from $\mathrm{acc}_{\mathrm{ideal}}(0) = p_s$ to $\mathrm{acc}_{\mathrm{ideal}}(1 - p_s) = p_l$. Past $r = (1 - p_s)$, it remains constant at p_l.

Figure 3 (b) in the main paper plots this ideal deferral curve (green line). It serves as an upper bound on how effective any real deferral strategy can be. In contrast, a purely random deferral strategy produces a linear interpolation (the red line), which is strictly below the ideal curve for most r. Consequently, the difference $\mathrm{acc}_{\mathrm{ideal}}(r) - \mathrm{acc}_{\mathrm{rand}}(r)$ represents the *maximum possible* gain one can achieve by carefully selecting which examples to defer rather than choosing them at random.

Summary We summarize the key take-aways below:

- **Ideal Deferral Routes All Mistakes:** Only the inputs misclassified by $\mathcal{M}_S$ get deferred, guaranteeing the highest possible joint accuracy at each deferral level r.
- **Piecewise Definition:** Accuracy increases linearly from p_s to p_l over the interval $r \in [0,\, (1 - p_s)]$, then remains at p_l.

- **Upper Bound on Realized Deferral:** No actual strategy can exceed this ideal curve, as it assumes perfect knowledge of which specific inputs $\mathcal{M}_S$ would misclassify.

B.4 Gatekeeper in the Context of Canonical Calibration Objectives

Motivation. Section 4 showed that the GATEKEEPER loss improves *deferral performance* with minimal implementation effort. Because many calibration objectives also manipulate confidence, we now position GATEKEEPER relative to four widely–used losses.

Canonical calibration objectives. Let $p_\theta(y \mid \mathbf{x})$ denote the softmax (or token) distribution predicted by a model with parameters θ and let $y^\star$ be the ground-truth label. Below we recap four popular alternate calibration objectives.

(a) **Temperature scaling** (Guo et al., 2017) applies a single scalar $T{>}0$ at test time: $p_T(y \mid \mathbf{x}) \propto \exp(z_c/T)$. It preserves the rank ordering and therefore *cannot tighten* the ranking-based risk–coverage curve, but can improve threshold-based acceptance.

(b) **Focal loss** (Lin et al., 2017) adds a down-weighting factor to easy examples, $\mathrm{FL}(p, y^\star) = -(1 - p_{y^\star})^\gamma \log p_{y^\star}$ with $\gamma \in [0, \infty)$. It improves class imbalance calibration but does not explicitly penalize over-confidence on *incorrect* samples.

(c) **Confidence penalty** (Pereyra et al., 2017) regularises high-entropy predictions through $\mathrm{CE}(p, y^\star) + \lambda \mathcal{H}(p)$. While it flattens *all* distributions, it does not distinguish between correct and incorrect cases.

(d) **Outlier exposure (OE)** (Hendrycks et al., 2018) adds a KL-uniform loss on auxiliary OOD data, mirroring the second term of GATEKEEPER but only on outliers, not in-distribution misclassifications.

How Gatekeeper differs. GATEKEEPER combines *two complementary gradients*: (i) a standard CE term on correct predictions *with an instance-level mask*, thereby **sharpening** those logits; (ii) a KL-to-uniform term on *incorrect* predictions, **flattening** their confidence. This asymmetric design forces the scalar summary $g(\mathbf{x}) = \max_c p_\theta(y{=}c \mid \mathbf{x})$ (or token-entropy) to separate correct from incorrect points *without* requiring additional heads, auxiliary datasets, or test-time tuning.

B.5 Extension to Multi-Level Cascades

Cascading can extend beyond a single deferral $\mathcal{M}_S \to \mathcal{M}_L$ to multiple deferrals $\mathcal{M}_1 \to \mathcal{M}_2 \to \cdots \to \mathcal{M}_K$. GATEKEEPER makes no explicit assumption on the number of cascade levels—it is inherently modular and agnostic to model architecture. In practice, one can fine-tune each model $\mathcal{M}_i$ in the chain independently using the GATEKEEPER loss with a stage-specific hyper-parameter α_i that governs the balance between confident acceptance and deferral. A natural hierarchical training procedure is to first apply GATEKEEPER to $\mathcal{M}_1$ to calibrate its confidence, then apply it to $\mathcal{M}_2$ on the subset of inputs deferred by $\mathcal{M}_1$, and so on through $\mathcal{M}_K$. This approach scales linearly in the number of levels and incurs only the familiar overhead of per-model fine-tuning—no joint optimization or additional routing networks are required. We therefore expect that deeper cascades will offer flexible trade-offs between computational cost and predictive accuracy across multiple model tiers, and we leave a full empirical evaluation of $K > 2$ cascades to future work.

B.6 Extension to Multi-Level Cascades

C Additional Experimental Details

C.1 CNN Used in Image Classification Experiments

Below we include a representation of the `SmallCNN` model used as $\mathcal{M}_S$ in image classification experiments discussed in Section 4.1:

```
SmallCNN(
  (features): Sequential(
    (0): Conv2d(3, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1,
        1))
```

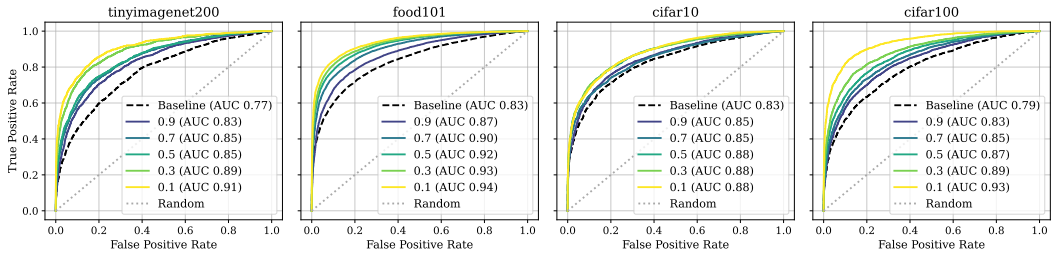

Figure 9: **ROC curves for image classification experiments**. Each figure shows the ROC curves for each of the datasets considered in Section 4.1. We observe that GATEKEEPER consistently increases separation of correct and incorrect confidence scores across varying α (colored curves) compared to the baseline (denoted with black dashed line).

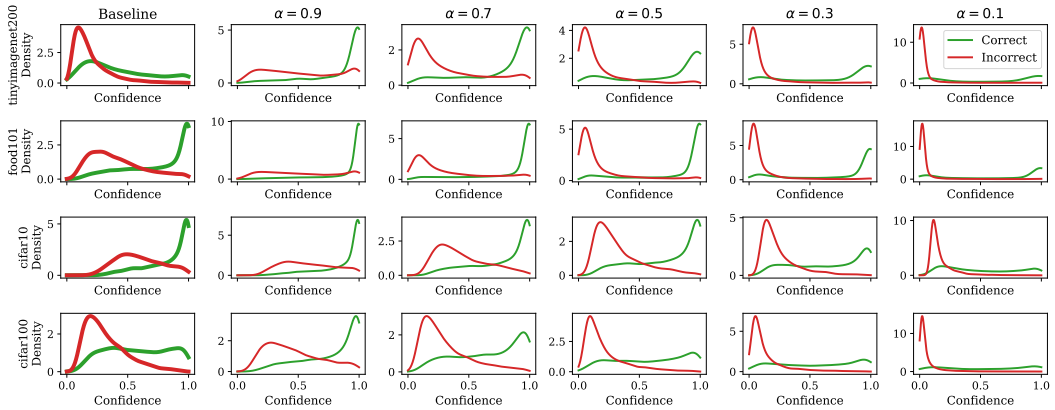

Figure 10: **Distributional overlap for image classification experiments**. Left-most column shows the results obtained using the untuned baseline, while the remaining columns correspond to the results obtained using GATEKEEPER with decreasing α values. Rows correspond to the datasets considered in Section 4.1. We see that GATEKEEPER increases separation of correct and incorrect confidence scores compared to the baseline.

```
    (1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True,
        track_running_stats=True)
    (2): ReLU(inplace=True)
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
        ceil_mode=False)
    (4): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1,
         1))
    (5): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True,
        track_running_stats=True)
    (6): ReLU(inplace=True)
    (7): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
        ceil_mode=False)
  )
  (classifier): Sequential(
    (0): Linear(in_features=2048, out_features=64, bias=True)
    (1): ReLU(inplace=True)
    (2): Linear(in_features=64, out_features=10, bias=True)
  )
)
```

C.2 Reduce Confidence and Answer "N" Baselines

In addition to the baseline model in Section 4.2 (i.e., a model that was not fine-tuned with our specialized $\mathcal{L}_{\text{def}}$ loss but from which we still compute predictive entropy as a deferral signal), we also examine two additional methods aimed at eliciting uncertainty from the model directly via prompt

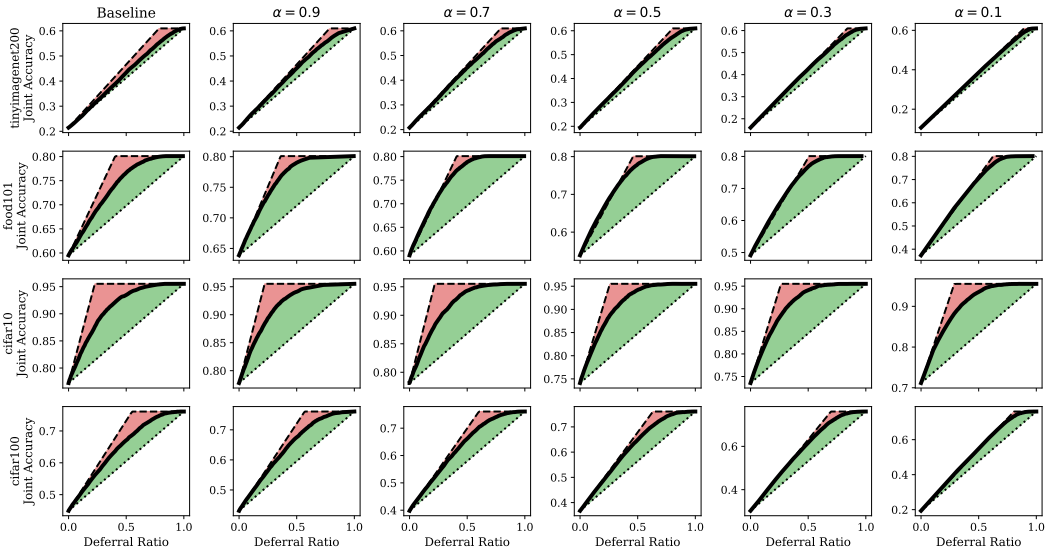

Figure 11: **Deferral curves for image classification experiments**. Left-most column shows the results obtained using the untuned baseline, while the remaining columns correspond to the results obtained using GATEKEEPER with decreasing α values. Rows correspond to the datasets considered in Section 4.1 The results show that GATEKEEPER brings the realized deferral (black line) closer to the ideal deferral (dashed upper line).

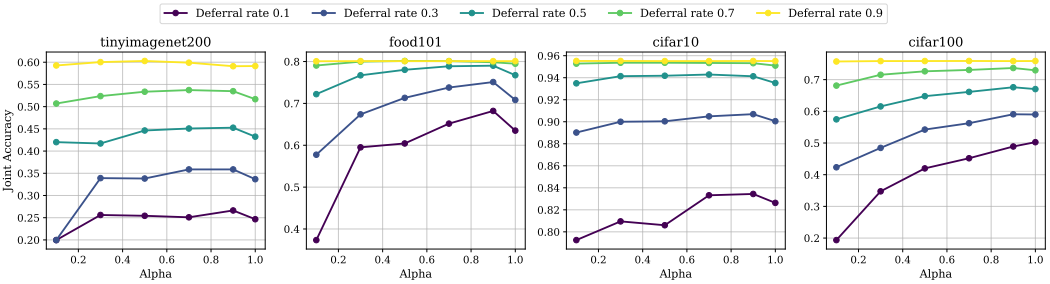

Figure 12: **Joint accuracy across different levels of** α. For varying fixed deferral ratios, we observe that the accuracy of $\mathcal{M}_S$ generally decreases as $\alpha \to 0$.

modifications. Both methods are *black box* approaches that only rely on a query interface to the model via prompt injection, and we provide their implementation details below.

Reduce Confidence. In this setting, we modify the original prompt $\mathbf{x}$ by appending an additional instruction $\mathbf{x}'$ that encourages the model to respond with lower confidence when it is uncertain: $\mathbf{x} \leftarrow \mathbf{x} \,\|\, \mathbf{x}'$. For instance, the instruction we add is:

$$\mathbf{x}' = \text{“}\texttt{Respond with low confidence if you are uncertain.}\text{”}$$

We treat this appended text as a hint to the model to self-regulate its confidence when producing an answer. This is similar in spirit to other black box approaches such as confidence quantification, rejection awareness, remote model notice, and self-critiquing. Although Xiong et al. (2024) show that large language models can express aspects of their confidence via prompting, our experiments indicate that simply prompting the model to express lower confidence does not reliably improve the separation of correct versus incorrect predictions, nor does it offer advantages in a deferral setting. These findings are in line with those reported in Kadavath et al. (2022).

Answer “N.” We also consider an alternate prompt modification, in which the appended instruction is:

$$\mathbf{x}' = \text{“}\texttt{Respond with 'N' if you are uncertain.}\text{”}$$

This approach explicitly instructs the model to produce a special "N" token to indicate uncertainty or lack of confidence. The intuition is that by introducing a designated "uncertain" response, one might isolate uncertain cases for deferral. However, our results in Section 4.2 similarly show that the model's ability to follow this instruction is inconsistent and does not substantially improve performance as a deferral model. The model often remains overconfident and fails to produce "N" in cases where it is in fact incorrect.

C.3 Additional metrics

In addition to the metrics outlined in Section 4, we also consider the **Area Under the Receiver Operating Characteristic Curve** (AUROC) (s_{AUROC}). The AUROC quantifies the model's ability to discriminate between correctly and incorrectly classified data points by evaluating the trade-off between the True Positive Rate (TPR) and the False Positive Rate (FPR) across various confidence thresholds τ. Formally, given the confidence sets $\mathcal{C}_{\text{corr}}$ and $\mathcal{C}_{\text{incorr}}$, the AUROC is defined as

$$s_{\text{AUROC}} = \int_0^1 \text{TPR}(\tau)\, \text{dFPR}(\tau), \tag{10}$$

where for each threshold $\tau \in [0, 1]$ we compute $\text{TPR}(\tau) = \frac{|\{c \in \mathcal{C}_{\text{corr}} | c \geq \tau\}|}{|\mathcal{C}_{\text{corr}}|}$ and $\text{FPR}(\tau) = \frac{|\{c \in \mathcal{C}_{\text{incorr}} | c \geq \tau\}|}{|\mathcal{C}_{\text{incorr}}|}$. Note that $s_{\text{AUROC}} = 1$ indicates perfect separability and $s_{\text{AUROC}} = 0.5$ corresponds to a random guessing baseline.

C.4 Factuality Scoring

Factuality scoring with Gemini for a reference caption r and a candidate caption c is computed as follows:

1. **Compute the log-likelihoods.** Let $\ell_{\text{Same}}(c, r)$ be the log-likelihood that the model outputs "Same" for a given candidate caption c and reference r, and let $\ell_{\text{Diff}}(c, r)$ be the log-likelihood that the model outputs "Different".
2. **Apply softmax.** To convert these log-likelihoods into probabilities, we exponentiate and normalize:
$$p(\text{Same} \mid c, r) = \frac{\exp\big(\ell_{\text{Same}}(c, r)\big)}{\exp\big(\ell_{\text{Same}}(c, r)\big) + \exp\big(\ell_{\text{Diff}}(c, r)\big)},$$
$$p(\text{Diff} \mid c, r) = \frac{\exp\big(\ell_{\text{Diff}}(c, r)\big)}{\exp\big(\ell_{\text{Same}}(c, r)\big) + \exp\big(\ell_{\text{Diff}}(c, r)\big)}.$$
3. **Interpret the probability.** The value $p(\text{Same} \mid c, r)$ is then taken as the factual alignment score, expressing how confidently the model believes the candidate caption is factually aligned with the reference.

C.5 Additional Experimental Results

In this section, we provide additional experimental results further supporting our findings reported for image classification experiments in Section 4.1. In particular, we show ROC curves in Figure 9 and distributional overlap in Figure 10, both demonstrating that GATEKEEPER increases the separation of correct/incorrect confidence scores. Similarly, the deferral curves in Figure 11 clearly show that GATEKEEPER successfully pushed the realized deferral (black line) closer to the ideal one (marked with dashed upper line). Lastly, we report the joint accuracy of $\mathcal{M}_S$ across varying α parameter in Figure 12. As discussed in Section 4, we observe that $\mathcal{M}_S$'s accuracy generally decreases with $\alpha \to 0$.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Our abstract and intro reflects the contributions accurately.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See Section 5.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: We do not provide any theoretical claims or results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: See Appendix C.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: We are not providing code for this work.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: See Appendix C.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: All of our reported results are reported as mean values over 5 random runs.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: See Appendix C.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The paper conforms to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: See Appendix A.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to

generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We are not releasing any new assets requiring safeguards.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have cited related work appropriately.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: We are not releasing any new assets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: We did not conduct any crowdsourcing and/or research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: We did not conduct any user studies requiring IRB approval.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: Our confidence tuning method can be applied to LLMs and we demonstrate this usecase in our experimental results (Section 4). In terms of paper writing, LLMs were used to help with editing.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.