Agreement-Based Cascading for Efficient Inference

Anonymous authors
Paper under double-blind review

Abstract

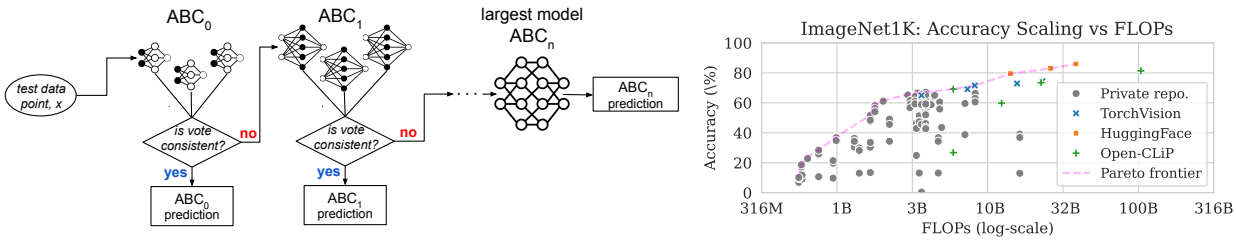
Adaptive inference schemes reduce the cost of machine learning inference by assigning smaller models to easier examples, attempting to avoid invocation of larger models when possible. In this work we explore a simple, effective adaptive inference technique we term *Agreement-Based Cascading (ABC)*. ABC builds a cascade of models of increasing size/complexity and uses agreement between ensembles of models at each level of the cascade as a basis for data-dependent routing. Although ensemble execution introduces additional expense, we show that these costs can be easily offset in practice due to large expected differences in model sizes, parallel inference execution capabilities, and accuracy benefits of ensembling. We examine ABC theoretically and empirically in terms of these parameters, showing that the approach can reliably act as a drop-in replacement for existing models and surpass the best single model it aims to replace in terms of both efficiency and accuracy. Additionally, we explore the performance of ABC relative to existing cascading methods in three common scenarios: (1) edge-to-cloud inference, where ABC reduces communication costs by up to 14×; (2) cloud-based model serving, where it achieves a 3× reduction in rental costs; and (3) inference via model API services, where ABC achieves a 2-25× reduction in average price per token/request relative to state-of-the-art LLM cascades.

1 Introduction

The high cost of inference associated with deploying large machine learning (ML) models presents a significant barrier to their adoption (Strubell et al., 2020; Kaplan et al., 2020). As models continue to increase in size, practitioners are often faced with investing substantial resources in updating existing deployments or settling for lower-performing alternatives. However, for many applications, it has been shown that a considerable portion of the data seen during inference can be effectively evaluated using small models rather than large, state-of-the-art models (Chen et al., 2020; Jitkrittum et al., 2023). This means that if we can identify the subset of data samples that can be accurately evaluated by more inexpensive models, average inference costs can be reduced. This problem is often referred to as *adaptive inference*, where the cost of inference adapts to some notion of 'difficulty' of each example seen at inference time.

A natural approach for adaptive inference is to *cascade* over a set of potential models, starting from the least expensive and moving to more expensive models based on some *deferral rule* (Rowley et al., 1998; Viola & Jones, 2001; Soo, 2014). A common cascade construction is to use a Pareto-efficient set of models and an easy-to-compute deferral rule such as the confidence scores of the models predictions (Viola & Jones, 2004; Wang et al., 2018a; 2021). Recently, cascading has seen renewed interest in the ML community with the advent of large language models (LLMs) and vision foundation models (e.g., Chen et al., 2023; Gupta et al., 2024). Recent approaches consider a variety of techniques to construct model cascades, including designing novel model architectures that have built-in cascading capabilities (Cai et al., 2019; Devvrit et al., 2023; Khare et al., 2023), or learning routing schemes/deferral rules that require data-dependent training for every task considered (Chen et al., 2023; Ding et al., 2024). While these approaches can improve accuracy/efficiency trade-offs, they may also introduce significant computational overhead in setup and training/finetuning costs.

In this work, we instead investigate a simple, *training-free* cascade scheme that uses the agreement among an ensemble of existing models at each cascade level as its deferral rule. We refer to this approach as *Agreement-Based Cascading (ABC)*. Intuitively, when the outputs of the ensemble at a particular cascade-level do not align, ABC triggers cascading, and inference is attempted at the next tier of (larger) models (see



(a) **(b)**

Figure 1: (a) Agreement-Based Cascading (ABC): ABC introduces a data-dependent routing scheme that uses agreement amongst an ensemble of models to determine whether to cascade to larger models. If the predictions of smaller ensembles do not align, the cascade moves to the next tier of larger models, continuing until agreement is reached or the largest model(s) are used. This can reduce cost by limiting the use of the largest models to cases where smaller models cannot reach consensus. **(b)** ABC is a natural baseline for adaptive inference due to (i) the vast number of pretrained models available to ML practitioners today; (ii) the fact that even small accuracy gains often require an order-of-magnitude increase in FLOPs, mirroring proposed scaling laws and resulting in large differences in model sizes between cascade tiers (Hestness et al., 2017; Henighan et al., 2020; Madaan et al., 2023). The pink dashed line represents the Pareto-optimal frontier, showing the models with the highest accuracy for a given computational budget. We show that ABC can effectively improve this frontier—allowing practitioners to achieve high accuracy without incurring the full computational cost of the largest models by invoking smaller models for 'easier' samples.

Figure 1). This scheme has been explored historically for specialized applications such as face detection (Zuo & de With, 2005; 2008; Susnjak et al., 2012), where ensembles of binary face detectors focus on subregions of the face and combine their output using decision networks. However, to the best of our knowledge, our work is the first to study the use of ensemble agreement as a deferral mechanism for more general, modern machine learning workloads.

We point to a few trends in ML that make ABC a particularly attractive approach for model cascading. In particular, while using an ensemble of models at each level may initially appear to increase overall inference costs, it is natural to believe this simple baseline could excel in real-world applications due to: (1) the growing ease of obtaining pretrained models of various sizes and accuracies due to the rise of model registries (Wolf et al., 2019) as well as flexible compression schemes (Zhu et al., 2023; Dennis et al., 2023); (2) scaling laws that predict a large increase in inference cost for every marginal increase in accuracy (Hestness et al., 2017) (Figure 1); and (3) an increased ability to execute ensembles of relatively small models in parallel, with minimal additional costs (Fern & Givan, 2003; Kim et al., 2023; Miao et al., 2023) (Figure 3).

With these motivations in mind, we rigorously study ABC as a baseline for adaptive inference in modern ML workloads. Overall, we make the following contributions:

- We propose ABC as a training-free, deferral-based cascading approach that leverages agreement among ensembles of existing models, eliminating the need for additional routing networks or specialized architectures.
- We theoretically characterize cases where ABC can replace an existing model deployment without affecting the accuracy, and define safe deferral rules as sufficiency conditions for their existence. We further characterize the expected accuracy and inference costs in this setting.
- We empirically evaluate ABC on a wide range of image and language tasks and find that ABC not only improves efficiency, but also accuracy, compared to the model that it aims to replace. We then consider the performance of ABC relative to existing cascading methods in common inference scenarios, including (1) edge-to-cloud inference where ABC reduces communication costs by up to 14×, (2) model-serving on heterogeneous GPUs, where ABC reduces rental costs by up to 3× and (3) inference using black-box access to model API services, where ABC shows up to a 25× reduction in average price per token.

2 Related Work

Adaptive inference schemes have been a topic of interest in machine learning for many years. This section discusses three main approaches to cascading and adaptive inference: score-based methods, trained routing

procedures, and dynamic networks. We highlight how ABC's approach relates to and differs from these existing methods.

2.1 Cascading using Score-based Deferrals

Traditional cascading methods often rely on simple *score-based metrics* for deferral decisions. We compare to the recent Wisdom-of-Committees method of Wang et al. (2021) as a general, representative method in this category but note that specialized instantiations of this approach have been applied across various domains in prior work, for instance, in object detection (Rowley et al., 1998; Viola & Jones, 2004; Wang et al., 2011; Cai et al., 2015; Angelova et al., 2015; Streeter, 2018), image classification (Wang et al., 2018a; 2021), and text classification (Li et al., 2021b; Mamou et al., 2022; Varshney & Baral, 2022; Lebovitz et al., 2023). In these settings, deferrals within cascading systems use metrics such as confidence scores, entropy, and probabilities (Gangrade et al., 2021; Geifman & El-Yaniv, 2019; Narasimhan et al., 2022). While using confidence scores of existing models is inexpensive, these scores are required to be well-calibrated, which is less common for off-the-shelf models (Guo et al., 2017; Enomoro & Eda, 2021). Recent methods have thus considered a variety of techniques to enhance score-based deferrals, such as explicitly adding deferrals as a prediction (Wang et al., 2018a), difficulty-aware regularization (Li et al., 2021b), temperature scaling (Wang et al., 2023), and calibration mechanisms (Nie et al., 2024).

Recently, Jitkrittum et al. (2023) explored the specific practical settings under which confidence-based deferral in cascades could suffer; these settings include: scenarios where downstream models are specialists (the error probability of the downstream model is highly non-uniform across samples), when samples are subject to label noise, and in the presence of a distribution shift between the train and test set. Notably, in all these failure modes, where confidence-based cascades tend to be inadequate, our approach is likely to offer improvements, as ensembles are known to help induce diversity, enable robustness to noise, and mitigate issues of distribution shift (Gontijo-Lopes et al., 2022; Sharkey, 1996; Dietterich, 2000; Džeroski & Ženko, 2004).

2.2 Cascades with Routing Procedures

Other methods avoid confidence scores entirely by training procedures independent of the model set to route instances as needed. This trend has gained prominence with the rise of black-box model API services for LLMs, where prediction scores are often unavailable. Guan et al. (2018) developed a selection module trained to determine the best-fit classifiers for data instances. Yue et al. (2024) (MoT LLM Cascade) used sampling and consistency checking to determine when to defer to a high-cost model. AutoMix (Madaan et al., 2023) proposed a few-shot self-verification mechanism, similar to Yue et al. (2024), but also introduced a Markov-based meta-verifier for cascading in context-grounded tasks. FrugalGPT (Chen et al., 2023) leveraged a cascade strategy that triages incoming queries using a DistilBERT router and scoring function. HybridLLM (Ding et al., 2024) used a fine-tuned DeBERTa (He et al., 2020) to route queries to models based on the predicted query difficulty and the desired quality level. On the same note, OrchestraLLM (Lee et al., 2023)—using hand-labeled data to create expert model pools—selects model to query based on embedding distances between the pools' instances and the test instances. Other methods also used some form of trained router, including RouteLLM (Ong et al., 2024), 'Fly-swat or cannon' (Šakota et al., 2024), and Shnitzer et al. (2023).

As we further discuss in Section 5.2.3 (where we empirically compare to FrugalGPT, AutoMix, and MoT LLM Cascade), these more sophisticated methods involve complex setups, data-dependent training, and increased computational overhead. They often require retraining routers for each new task, dataset, or model, limiting adaptability to unseen data distributions or new models without incurring further costs. In contrast, ABC provides a simpler, flexible, widely applicable alternative at no additional training or setup cost.

2.3 Dynamic and Adaptive Networks

Routing procedures typically train an independent routing mechanism, keeping the Pareto-set of models untouched. However, for many medium-scale applications, methods have been explored that learn a Pareto-set of models and stopping criterion together. For example, early exit methods of Huang et al. (2018); Wang

Algorithm 1 Agreement-Based Cascading (ABC)

Require: Set of ensembles $\{H_1, H_2, \ldots, H_{n_E}\}$, deferral rule r_i for each ensemble $i \in [n_E]$ as in Equation 3 or 4
Require: A new inference data point x.
  Current cascade level, $i \leftarrow 1$
  Cascaded prediction, $y \leftarrow \emptyset$
  for $i \in \{1, \ldots, n_E\}$ **do**
    $y \leftarrow H_i(x)$
    if $r_i(x) = 0$ **then**
      break // Models in ensemble 'agree'
    end if
  end for
  return y =0

et al. (2018b); Xin et al. (2020); Geng et al. (2021); Zhou et al. (2020); Liu et al. (2020); Schuster et al. (2022), subnetworks extraction methods (Yu et al., 2018; Yu & Huang, 2019; Chen et al., 2021; Hou et al., 2020; Han et al., 2022; Devvrit et al., 2023), composition-based methods (Suggala et al., 2020; Dennis et al., 2023; Du & Kaelbling, 2024) are some of the more recent examples.

Our approach diverges from these methods, as it does not involve altering model architectures or retraining from scratch. Instead, ABC leverages existing models as they are, utilizing ensemble agreement as a deferral condition, which can be applied directly to these models without any need for fine-tuning or specialized training.

3 Agreement-Based Cascading (ABC)

In this section, we present a high-level overview of the Agreement-Based Cascading (ABC) approach, providing the essential concepts and insights needed to understand ABC 's effectiveness in the experiments (Section 5). Readers interested in the formalizations and technical details can refer to Section 4 for a more comprehensive treatment.

3.1 Overview of ABC Approach

As outlined in Section 1, the goal of adaptive inference is to identify data samples that can be evaluated accurately by a relatively inexpensive model. This way, we can reduce the inference cost based on whether a sample is 'easy' or 'hard'. A common approach to this problem is deferral-based cascading, where we cascade over a sequence of models, starting from the least expensive and using a 'deferral rule' to determine if a higher tier model must be used. These deferral rules are typically significantly cheaper to evaluate (e.g. a small neural network), and add very little additional cost on top of model execution. This approach makes it so that 'simpler' cases are managed by smaller, faster models, while only the complex cases cascade up multiple tiers and to more resource-intensive models.

Agreement-Based Cascading (ABC), described in Algorithm 1, is one such deferral based cascading approach. ABC maintains a set of ensembles $\{H_1, \ldots, H_{n_E}\}$ that starts from an ensemble of inexpensive models in H_1 to expensive, state-of-the-art models in H_{n_E}. Similar to other cascading techniques, given a sample x, we start inference from the lowest (cheapest) tier in the cascade and use a deferral rule, $r_i(x)$, to determine if a higher tier model is needed. A core distinction between ABC and existing cascading approaches is its agreement-based deferral rule. Instead of training a small additional deferral network or post-hoc deferral rules, we use a notion of *agreement* between models within each ensemble as a confidence measure. When a (configurable) fraction of models in a tier agree, it signals that they likely have the right answer. Conversely, if they disagree, the uncertainty triggers a deferral to the next, more powerful ensemble. Additionally, we focus on cases where we wish to maintain accuracy of the overall prediction and only improve inference cost when accuracy does not suffer. This is different from existing approaches that prioritize a fixed inference budget, potentially at the cost of accuracy.

3.2 ABC in Modern ML Environments

In recent years, obtaining trained models has become much easier, thanks to the rise of public and private model repositories (Wolf et al., 2019), where a wide variety of pre-trained models are readily available. This abundance of accessible models of various sizes and performance levels makes the practical application of ABC especially appealing. Unlike methods that require additional task-specific training or fine-tuning, ABC can leverage these existing models, making it suitable for direct deployment as a "drop-in" replacement for high-cost models. This adaptability, combined with minimal setup and training requirements, can allow ABC to fit seamlessly into many existing ML workflows.

Strictly speaking, evaluating agreement between multiple models in an ensemble is expensive when compared to the small router models that are used in many existing approaches. However, two key aspects of modern ML workloads can help to mitigate this additional cost in practice. First, in many cases some degree of *parallelization* is available that can reduce the impact on inference cost metrics such as latency. We use ρ to smoothly interpolate between the fully sequential case ($\rho = 0$) to the fully parallel case, ($\rho = 1$) as detailed in Section 4.3. Second, in many use-cases the difference in cost between models in successive tiers of cascades is so large that the impact of lower tier models on the overall cost of inference is negligible, even with the added cost of constructing ensembles (see Figure 3). We use γ to denote the *relative cost* of the models — the ratio of the cost of the smaller model to the larger model.

In Section 5, we evaluate ABC 's performance across several real-world scenarios (resulting in varying settings of ρ and γ), demonstrating its competitiveness and efficiency. We first show that ABC's ensemble-based deferral mechanism preserves and often improves on the accuracy of models it aims to replace. We also show that when the lower tier models are small enough compared to the higher tier models, ABC 's ensemble-based deferral rule operates with negligible impact on the overall cost.

We then evaluate cases in real-world workloads where such large relative costs between various tiers of ABC is natural. In edge-to-cloud setups, ABC enables substantial reductions in communication costs by processing simple tasks locally on the edge. This minimizes the need to transfer data to the cloud, reducing both latency and data transfer expenses. When serving models on heterogeneous GPU resources in the cloud, ABC significantly reduces inference costs. By selecting models based on task complexity, ABC makes efficient use of available GPU resources, optimizing for both accuracy and rental cost. In black-box API-based model deployments, where the user is billed per request or token, ABC offers substantial savings by reducing the average cost per request. By deferring to high-cost models only when necessary, ABC achieves notable economic savings compared to standard inference approaches.

4 Formalization and Theoretical Details

In this section, we formalize our problem setup by first outlining the standard statistical learning framework, which serves to define the concepts of models, ensembles, the learning performance (eg. accuracy) and the inference cost associated with ensembles (Section 4.1). Next, we describe deferral rule-based cascades and formulate the specific case of *drop-in cascades*, cascades which prioritise accuracy over inference cost savings (Section 4.2). We introduce *safe deferral rules* as a sufficiency condition for constructing drop-in cascades and define our deferral rule based on agreement between models in an ensemble (Section 4.3). We conclude this section by formalizing the test-time accuracy-inference cost behaviour of drop-in cascades (Section 4.4).

4.1 Problem Setup and Notation

Consider the standard statistical learning settting where $\mathcal{X}$ denotes an instance space and $\mathcal{Y}$ denotes the label or response space. Let $h(x)$ be a model taking inputs from $\mathcal{X}$ and producing outputs in $\mathcal{Y}$. We assume that the models are from some hypothesis class $\mathcal{H}$. In this setup, we typically characterize the learning performance of various models using its *risk* with respect to some data distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$ and a loss function l, given by,

$$\mathcal{R}_l(h) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[l(h(x), y)].$$

As a concrete example, in the classification setup, where we use the miss-classification error as a loss function, the risk is given by $\mathcal{R}(h) = \mathbb{E}(y \neq h(x))$.

Assume that each model $h \in \mathcal{H}$ has a cost of inference, denoted by $C : \mathcal{H} \to \mathbb{R}_+$; for instance, for cases where we are concerned about inference latency, the cost can be the latency of the model on the target hardware. Let h_1 and h_2 be two models with h_2 being the more expensive of the two. We denote the *relative cost* of the models by $\gamma := \frac{C(h_1)}{C(h_2)}$, satisfying $0 < \gamma \leq 1$.

Let $H^k : \mathcal{X} \to \mathcal{Y}$ denote an ensemble of k models from the same hypothesis class $\mathcal{H}$. Similarly to before, the learning performance of various ensembles can be characterized by their risk; let $\mathcal{R}(H^k)$ denote the risk of an ensemble H^k. Compared to a single member model, the cost of evaluating the entire ensemble of models depends on the various factors, including the degree of parallelization. Assuming that the models in an ensemble are of similar cost, say c_0, we model the cost of the ensemble using a *parallelism coefficient* $0 \leq \rho \leq 1$ as,

$$C(H^k) = c_0 k^{1-\rho}. \tag{1}$$

Here, when $\rho = 1$, the ensemble suffers the same cost as a single model and corresponds to the case where models can be fully parallelized. On the other extreme, at $\rho = 0$, the cost of the ensemble of k models is kc_0, corresponding to no parallelization (sequential evaluation).

4.2 Deferral Rule-Based Drop-in Cascade

A deferral-based cascade consists of a finite set of models and a deferral rule. Here, the idea is to start with the most resource-efficient model and use the deferral rule to determine if a better-performing model with a higher resource cost should be used for the current sample. The deferral rules themselves are designed to have negligible inference cost given the models inference outputs. For this exposition, we will restrict ourselves to cascades with only two levels, though the discussion in this section can be readily generalized to larger cascades. Let the cascade consist of an ensemble H_1^k at the lower level, and the larger model h_2 at the higher level; $\mathcal{M} = \{H_1^k, h_2\}$. Let $r(x)$ denote a deferral rule;

$$r(x) = \begin{cases} 1 & \text{defer to } h_2 \\ 0 & \text{use } H_1^k. \end{cases}$$

In this work, we focus on deployment scenarios where a decrease in model performance is a critical concern. This is particularly relevant when cascades are intended to function as a drop-in replacement for an existing deployment of an expensive model. This notion can be formulated as maximizing the number of calls to the smaller model, while retaining the accuracy of the larger model. With slight abuse of notation, let $\mathcal{M}_r(x)$ denote the prediction of the cascade with $\mathcal{M} = \{H_1^k, h_2\}$ and deferral rule r. Then we desire that for a choice of small error budget $\xi > 0$,

$$\begin{aligned} \max_r \quad & \mathbb{P}(r(x) = 0) \\ \text{s.t.} \quad & \mathbb{P}(y \neq \mathcal{M}_r(x)) \leq \mathbb{P}(y \neq h_2(x)) + \xi, \end{aligned} \tag{2}$$

The constant function $r(x) = 1$, which defers for every x, is feasible and attains the objective value of 0. Moreover, *every* feasible deferral rule leads to a cascade with competitive accuracy as h_2. Instead of picking an error budget ξ, a complimentary (dual) approach is to consider a fixed inference budget and aim to attain the best accuracy within the budget. In this view, drops in accuracy (compared to the existing large model) are permissible, provided the inference budget is met. Jitkrittum et al. (2023) examine this perspective, which we recommend to interested readers.

4.3 Deferral using Ensemble Agreement

Ensemble agreement is an instantiation of score based deferral rules (Jitkrittum et al., 2023). In its simplest form, we associate a score to the prediction output of an ensemble model and interpret this score as a measure

of confidence in that particular prediction. We can then defer to a larger model when the confidence is below a predetermined threshold. For a model $H_1^k(x)$ and a scoring function $s(x)$, the deferral rule is defined as

$$r(x) = \mathbb{I}[s(x) \leq \theta],$$

where $\mathbb{I}$ is the indicator function. In general, such score based deferral rules can sometimes be misleading. For instance, in multi-class classification, rules that map the outputs of a classifier to a probability distribution over labels can produce confidently incorrect predictions when an unperceivable amount of perturbation is added to the data sample (Szegedy, 2013). However, such rules have been shown to be effective in practice (Wang et al., 2018a; Gangrade et al., 2021; Mamou et al., 2022; Gupta et al., 2024) implying that such adversarial data is rare in many cases. Motivated by this observation, we propose the following property of scoring functions (and by extension the corresponding deferral rule).

Definition 4.1 (Safe deferral rule)**.** *Let $H^k : \mathcal{X} \to \mathcal{Y}$, $k \geq 1$ be a classifier and let $s : \mathcal{X} \to [0, 1]$ be a scoring function. The scoring function s is referred to as a safe scoring function for H^k if there exists $\theta \in [0, 1]$ such that, for some small $\epsilon > 0$,*

$$\mathbb{P}(s(x) \geq \theta, H^k(x) \neq y) \leq \epsilon.$$

The corresponding deferral rule $r(x) = \mathbb{I}[s(x) \leq \theta]$, is referred to as a safe deferral rule, and satisfies $\mathbb{P}(r(x) = 0, H^k(x) \neq y) \leq \epsilon$.

We define safe deferral with respect to classification tasks for simplicity. The definition can be extended to other cases with appropriate choice of loss function. Intuitively, this formalizes the notion that the deferral rule can probabilistically identify a *subset* of $\mathcal{X} \times \mathcal{Y}$ for which $H_1^k(x)$ is correct. Alternatively, we can think of the deferral rule as a one sided classifier, similar to the problem studied in Goyal et al. (2020), where the class to be identified is the *subset* of $\mathcal{X} \times \mathcal{Y}$ where $H_1^k(x)$ is correct. In general, such a safe deferral rule need not exist for a given model $H_1^k(x)$. However, in Section 5 we demonstrate empirically that such rules can in-fact be constructed without additional training for many real world tasks with appropriate choice of θ (Figure 7). These rules lead to selection rates as high as 90% of the data in cases such as ImageNet1k.

Assuming access to a safe deferral rule, observe that for $\xi \geq \epsilon > 0$, such rules are feasible for the program in Equation 2 (see Proposition 4.1). This implies that *every* cascade, $\mathcal{M} = \{H_1^k, h_2\}$ using a safe deferral rule r, is competitive with the larger model h_2 in terms of accuracy. We empirically evaluate these drop-in cascades in Section 5. Note that the optimal safe deferral rule can lead to an improvement in the accuracy of `ABC` in theory (see, Appendix A), and we see accuracy improvements in our experiments (Section 5).

Agreement-based deferral rule. We evaluate two flavors of deferral rules that capture the notion of agreement between models. In cases where we have direct access to the models, we directly use the outputs produced by models in the ensemble. However, in certain cases, for instance when interfacing through an inference API from a third party provider, we only have black box access to models. For such cases, we use a voting scheme between the models in an ensemble to construct our scoring function. Concretely, for an ensemble H^k consisting of $k \geq 1$ models, let $s(x; H^k)$ denote the average score of the majority prediction on x and let $\text{vote}(x; H^k) = \frac{1}{k} \sum_{h \in H^k} \mathbb{I}[H^k(x) = h(x)]$ denote the fraction of votes received by the prediction, where,

$$r_{\text{v}}(x; \theta_v) = \begin{cases} 1 & \text{vote}(x; H_1^k) \leq \theta_v \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

$$r_{\text{s}}(x; \theta_s) = \begin{cases} 1 & s(x; H_1^k) \leq \theta_s \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

4.4 Inference Cost Savings and Competitiveness

Since we do not impose an inference cost budget on the cascade, it is possible for the cascade to incur a higher cost than simply using the larger model h_2. Intuitively, if majority of the samples seen during test time are 'hard', the cascade pays the cost of evaluating both $H_1^k(x)$ and $h_2(x)$.

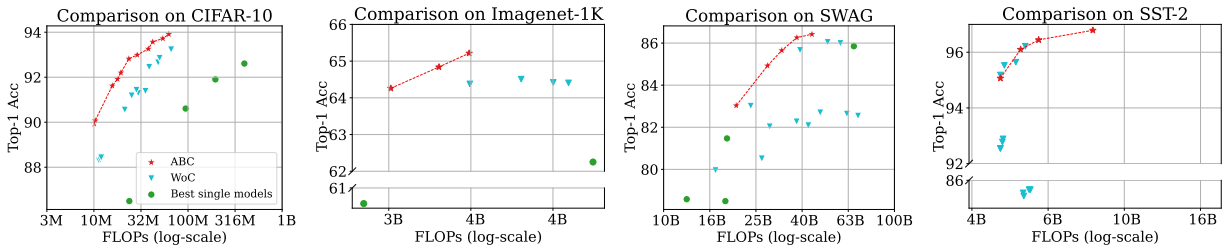


Figure 2: Pareto curves of ABC vs. confidence-based cascades (WoC) (Wang et al., 2021) and best single models on diverse tasks. For WoC, we tune its cascade configurations across the best four of its confidence thresholds and generate results from their most performant cascades. ABC maintains a Pareto-optimal curve, which consistently outperforms both methods in accuracy with lower FLOPs costs.

Proposition 4.1. *Let $\mathcal{M} = \{H_1^k, h_2\}$ be two classifiers and r a deferral rule such that r is a safe deferral rule for H_1^k according to Definition 4.1, for a distribution $\mathbb{P}$ over $\mathcal{X} \times \mathcal{Y}$. Then for every $\xi \geq \epsilon > 0$, the agreement based cascading (ABC) classifier $\mathcal{M}_r(x)$ is such that,*

1. *The ABC classifier is competitive with the large classifier h_2 in terms of accuracy,*

$$\mathcal{R}(\mathcal{M}_r) \leq \mathcal{R}(h_2) + \epsilon,$$

2. *The ABC classifier enjoys an average inference cost of,*

$$\mathbb{E}[C(\mathcal{M}_r)] = (k^\rho \gamma + \mathbb{P}(r(x) = 1))C(h_2).$$

Thus the cost savings we can expect with a drop-in cascade depends on three key factors; the relative cost γ, the degree of parallelization ρ and the deferral rate $\mathbb{P}(r(x) = 1)$, or equivalently, the *selection rate* $\mathbb{P}(r(x) = 0)$. In particular, for *every* feasible deferral rule r, in the best case scenario where the cost of the smaller model is negligible, ($\gamma = 0$), the cost of inference reduces by the selection rate, $P(r(x) = 0)$. Conversely, in the worst-case scenarios, the cost can be $(k + 1)$ times the cost of the larger model. In the next section, we demonstrate real world scenarios where the favorable interplay of these three quantities lead to significant improvements in inference cost. See Appendix C and Figure 7 for information on selection rates for various models and datasets considered here.

5 Experiments

This section evaluates our training-free Agreement-Based Cascading (ABC) approach across a variety of language and vision tasks, focusing on accuracy and inference cost. First, in Section 5.1.1, we examine ABC's accuracy-cost tradeoff against state-of-the-art models under full parallelization, setting the *parallelization factor* (ρ) to $\rho = 1.0$ for optimal accuracy-FLOPs trade-off. Next, in Section 5.1.2, we show that using ensembles at lower tiers has minimal impact on inference cost when lower-tier models are at least 50× cheaper than the largest model. In such cases, the *relative cost* (γ) satisfies $\gamma \leq \frac{1}{50}$, making ABC effective without parallelization. Finally, Section 5.2 explores practical scenarios where low relative costs make ABC is naturally suited for real-world deployments.

Estimating Voting Threshold: ABC's deferral rule uses a configurable voting threshold, θ (see Equations 3 and 4) at each cascading tier. We estimate θ empirically on a small set of unseen data; see Appendix B for details.

Datasets: To evaluate ABC, we use a range of benchmark datasets for image and language tasks, as shown in Table 2 in the Appendix. Additional datasets are used in Section 5.2.3 to align with those explored by state-of-the-art baselines.

Models: We select diverse models for both image and language tasks, summarized in Appendix's Table 3. For BERT-based models, we use the `BASE` and `LARGE`, and for image models, we tier by FLOPs count. All models are sourced from HuggingFace Zoo for inference without any additional training effort on our end. Section 5.2.3 uses models from LLaMA 3, Gemma 2, and Qwen 2 families via the Together API. We detail this section's experimental setup in Appendix D.2.

Evaluation: For the generation tasks we consider in §5.2.3, the datasets each consist of a fixed set of possible outputs akin to the classification tasks, and we apply our deferral rule on the final output at each tier. Certain cases don't have a fixed set of output labels (while still not being open-ended generation), like (1) GSM8K, where there is a final numeric answer at the end of the generated output to the math word problems, and (2) CoQA, where we used F1-score to capture overlaps between predictions and ground-truth answers. We do not discuss open-ended generation tasks in this work as the deferral rule we consider—and the baseline methods that we compare to–is not directly applicable to such cases.

5.1 When is `ABC` Practical?

5.1.1 When Parallelization is Cheap

We first consider the case where the inference cost of an ensemble of models is the same as the cost of a single model. This idealistic scenario could happen, for instance, in offline batch inference (Jetty et al., 2021), when GPUs are available for parallelization, or the models are small enough that existing resources are under-utilized and an ensemble adds no additional cost. More importantly, this setting establishes best-case accuracy and inference cost values for `ABC`. We consider total floating-point operations (FLOPs) as a representative inference cost metric here, and discuss other metrics such as communication cost, latency, or cloud rental costs in subsequent subsections.

The accuracy vs. FLOPs for `ABC` is shown in Figure 2. As a point of comparison, we also include Wisdom-of-Committees (WoC) (Wang et al., 2021), a popular and representative confidence-based model cascading method. In most cases, we observe that `ABC` is able to improve the Pareto frontiers, as it usually sees a 1–2 point increase in accuracy. We attribute this to a combination of (a) the improvement ensembles can have on accuracy that is widely noted in literature (Gontijo-Lopes et al., 2022; Jiang et al., 2023) and (2) the improvement agreement-based rules can cause in cascading (Appendix A). Overall, in terms of accuracy, `ABC` either exceeds—or at least matches—the accuracy of the best models at a fixed FLOPs budget, and is a practical drop-in replacement for these models.

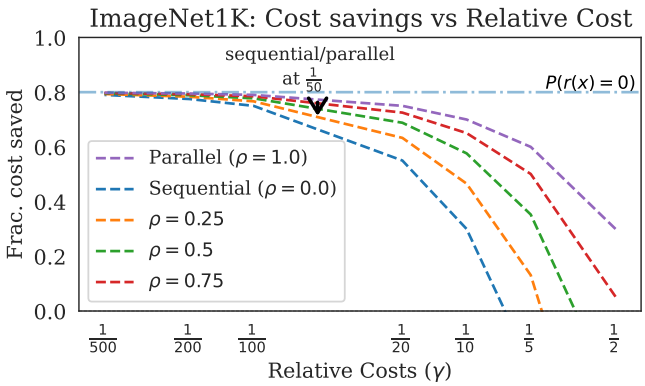


Figure 3: Fraction of inference cost saved as a function of relative cost of models (γ), *assuming* a fixed selection rate $\mathbb{P}(r(x) = 0)$. As parallelization decreases from fully parallel ($\rho = 1$) to sequential ($\rho = 630$), cost of evaluating ensembles increase and the cost savings decrease. When models across tiers are of similar size (e.g., smaller model is at most 5× smaller, $\gamma \geq \frac{1}{5}$), some parallelization is needed for `ABC` to reduce costs effectively. However, for lower relative costs (e.g., smaller model is at least 50× smaller, $\gamma \leq \frac{1}{50}$), sequential and parallel settings achieve meaningful savings, showing ABC's efficiency even with the added cost of using ensembles.

5.1.2 Disparity in Relative Cost is High

Of course, in many real-world scenarios, the cost of evaluating ensembles is not negligible. For instance, in the edge-to-cloud inference scenario discussed in Section 5.2.1, models in an ensemble often are evaluated sequentially. However, even in such scenarios, if the relative cost of models across each level of the cascade, γ, is small enough, this additional cost becomes negligible compared to the overall cost.

In Figure 3, we demonstrate the impact of using ensembles at various relative costs on the overall inference cost. As shown in the first plot, as we move away from the ideal, fully parallel setting with $\rho = 1$ towards the sequential setting with $\rho = 0$, the fraction of inference cost saved decreases. In fact, when the models in

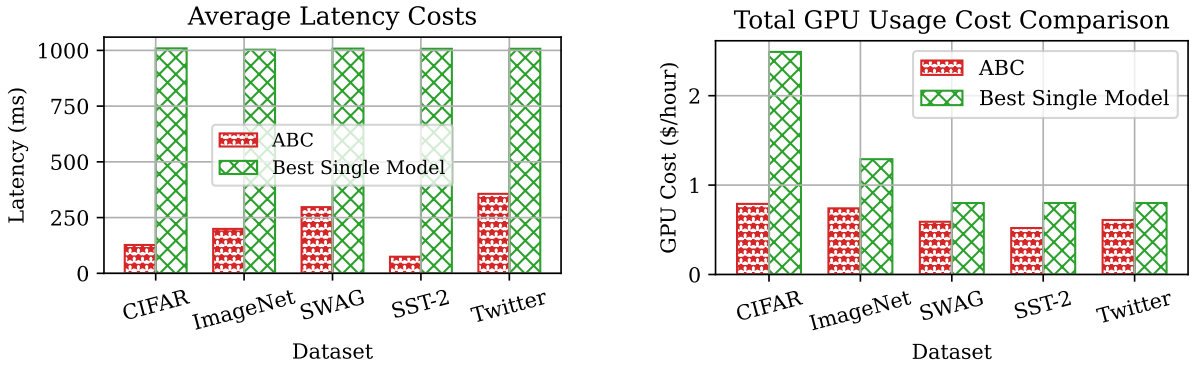


Figure 4: (a) ABC for edge-to-cloud inference: We simulate a single-instance inference setup, as seen in real-time applications where predictions may need to be made as new data becomes available. ABC can enable small models to be served at the edge without sacrificing accuracy—leading to large savings in communication costs over the alternative of using only the highest accuracy/largest model residing in the cloud, or a single small and low-performing model on the edge. (b) Total GPU usage costs of ABC vs. using the best model. Agreement-Based Cascading, at reduced costs of GPU usage, exceeds the accuracy of the single best models in all task categories.

the cascade are of similar size—for example, when $\gamma \geq \frac{1}{5}$—a certain degree of parallelization is required for ABC to reduce inference cost. However, observe that when the relative costs are small enough (e.g., $\gamma \leq \frac{1}{10}$), the need for parallelization diminishes. This is evident in Figure 3 (right); for $\gamma \leq \frac{1}{50}$, the sequential execution curve ($\rho = 0$) approaches the curve with full parallelization ($\rho = 1$).

5.2 Real-world Use-cases

As noted in the previous two subsections, ABC either improves on, or is at least competitive with, the single best model in terms of accuracy. Moreover, whenever the relative cost is small enough, ABC suffers negligible additional penalty for using an ensemble of models. As noted in Figure 1, accuracy vs inference cost scaling for ML models already imply that the relative cost of models that only differ a few points in accuracy is small. For instance, the state-of-the-art model performance on ImageNet1K dataset attains about 83% top-1 accuracy with 70B FLOPs. A Pareto-optimal model that achieves 63% accuracy requires only about 1B flops, and thus a two-level ABC with ensembles of these two models have relative cost of $\gamma = \frac{1}{70}$.

In many real world scenarios, this disparity in relative cost is further amplified due to deployment considerations — for instance, in an edge-to-cloud inference setting, local inter-process communication (IPC) latency is typically two orders smaller than remote (cloud) IPC latency ($\gamma \approx 10^{-2}$). Moreover, the ability of models in ABC to be placed on multiple, distributed devices with negligible synchronization overhead adds more opportunities for inference cost reduction. We consider three such scenarios, their respective cost models, and the benefits of ABC in these cases here. These include edge-to-cloud inference (Section 5.2.1), cloud-based model serving on heterogeneous GPUs (Section 5.2.2), and black-box inference access to model API services (Section 5.2.3).

5.2.1 Communication Cost in Edge-to-Cloud Inference

An advantage of ABC is that it allows a single large model to be split into multiple, potentially much smaller models, with only a simple reduce operation required to compute agreement. This allows us to tune device placement at various levels of ABC to improve inference costs. One use-case where this is beneficial is edge-to-cloud inference (Forooghifar et al., 2019); here, inference requests are generated on user-facing edge devices like mobile phones or smart devices, which are sent over the network to a cloud service for evaluation. Given an inference request generated by a user interaction, the time-to-response or response latency in such case is dominated by communication overheads (network speed, serialization overhead, network congestion, etc.) beyond our control. By using ABC for such applications, we are able to distribute the inference load between tiny, on-device models and the cloud models, allowing us to avoid communication costs for a significant portion of requests.

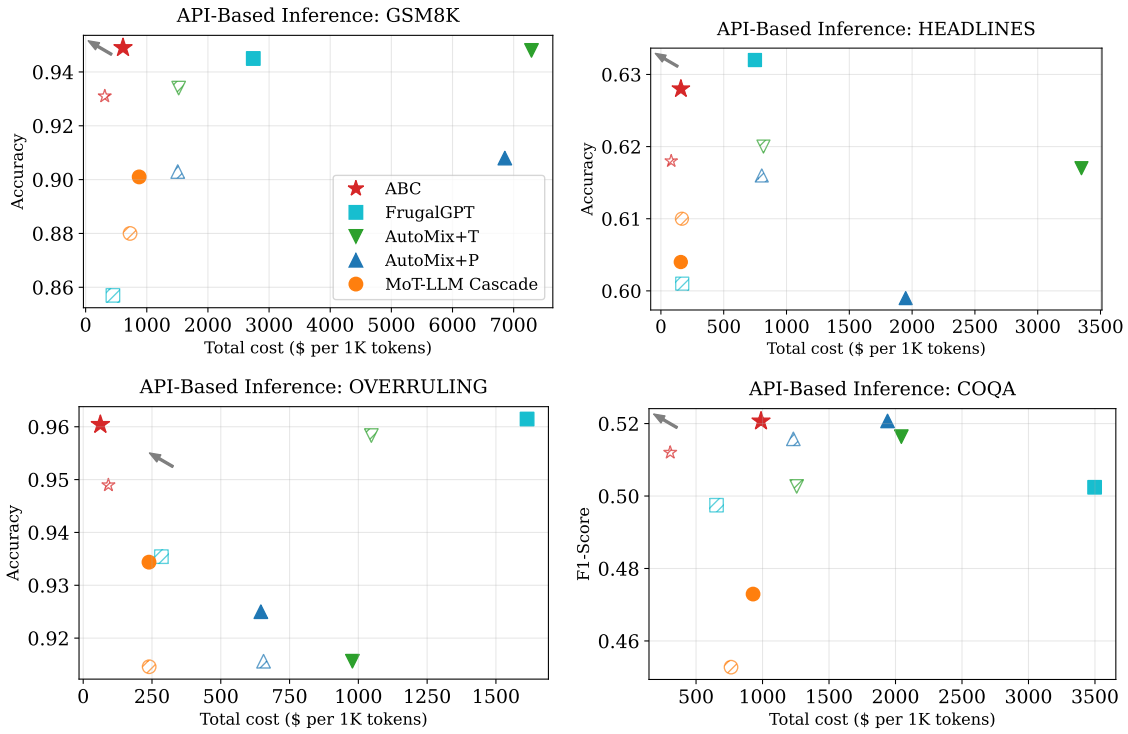


Figure 5: Comparison of `ABC` against state-of-the-art cascade baselines for black-box API-based inference. The faded, hatched-patterned variants represent budget-friendly, 2-level cascade instances where we do not include the costly Tier 3. Most of these methods show competitive performance, but `ABC` matches their accuracy at significantly lower costs in all tasks. Note that all these methods (aside from the MoT-LLM cascade) incur additional setup costs not reflected in our plots.

To understand the effectiveness of `ABC` in such a scenario, we consider a communication cost model previously studied in Zhu et al. (2021); Lai et al. (2022) in a setup of edge devices (i.e., Raspberry Pis and smartphones) and cloud servers. The delay parameters adopted range from small, medium, to large [1 us, 10 ms, 100 ms, 1000 ms], where near-instantaneous local communication ($<$ 1 microsecond) can be expected to occur with base cascade tiers performing inference on-device, and substantial network delays might occur ($>$ 1 second) in a worst-case edge-to-cloud transmission ($\gamma = 10^{-6}$).We simulate this by considering a two-level cascade, with the smaller level placed on the edge-device. We apply the delay to the cascade exit points on the edge device to capturing the time cost of transitioning between edge-to-cloud.

Our results, as shown in Figure 4, show that the flexibility that `ABC` affords in terms of model placement allows significant latency reductions, while providing superior accuracy compared to a single cloud model. In particular, we see that cascading in these scenarios provides an $14\times$ reduction in communication cost for language tasks like SST-2; and for image datasets, we see a $5\times$ reduction for ImageNet1K and a $8\times$ reduction in CIFAR10.

5.2.2 Monetary Cost for Model Serving on Heterogeneous Hardware

Another use-case which can take advantage of the model placement flexibility of `ABC` is using heterogeneous hardware for model serving on the cloud (Crago & Walters, 2015; Li et al., 2021a; 2023; Mo et al., 2023). GPU/Accelerator hardware typically has a disproportionately large difference in hardware costs compared to their throughput difference. For instance, based on the current pricing model offered by Lambda (Lambda, 2024), a popular cloud rental platform, the rental pricing of a single A100 is \$1.40/hour and a V100 node is \$0.06/hour ($\gamma \approx 4 \times 10^{-2}$), while the rated 32-bit tensor core throughput is 312 TFLOPs for A100 and 125 TFLOPS for V100. In this scenario, a simple placement strategy for a 2-level `ABC` that reduces inference cost may place the smaller model on V100 nodes and larger models on A100 nodes. Since the lower level models are—at least—an order of magnitude cheaper than the larger one in terms of FLOPs, this offsets the throughput loss we incur when switching from A100 to V100; and as a result, the `ABC` implementation incurs a much lower average cost.

To concretely evaluate ABC under such a cost model, we retrieve the costs of GPU usage by hour from Lambda Cloud's pricing (see, Table 4 in the appendix for details). For simplicity, we assume that each ensemble tier is set up on a distinct GPU in increasing order of GPU sophistication, and serves a uniform inference request rate. We also assume that the nodes are co-located and communication cost between them is negligible. We show a summary in Figure 4 and present the detailed results in Table 5, in the Appendix. We can see at least a 3× reduction in inference cost in terms of \$/hour for image tasks, and a more moderate 10-30% reduction in language tasks.

5.2.3 Monetary Cost for Black-box API-Based Inference

Finally, in the current landscape dominated by LLMs, many providers offer black-box API access to their proprietary LLMs (Abdalla et al., 2023; Sun et al., 2022; Hadi et al., 2024). Similarly to the large cost difference between GPU generations discussed in the previous subsection, we also observe a large cost disparity between various generations/tiers of API calls. For example, using Together.ai—one of the lowest-cost serverless endpoint providers for LLMs[1]—as a case study, we find that models within the 7B-8B range cost \$0.20 per million tokens, while LlaMA3.1-405B costs \$5.00. This translates to the larger model costing 25× more than the smaller model range ($\gamma = \frac{1}{25}, \rho = 1$). If we consider GPT-4 as the gold standard, the cost of usage quickly scales to 150× of our reference smaller model's costs.[2].

Tier	Model	Price
Tier 1	LlaMA 3.1 8B-Instruct Turbo	0.18
	Gemma 2 9B IT	0.30
	LlaMA 3 8B Instruct Lite	0.10
Tier 2	LlaMA 3.1 70B Instruct Turbo	0.88
	Gemma 2 27B Instruct	0.8
	Qwen 2 72B-Instruct	0.9
Tier 3	LlaMA 3.1 405B Instruct Turbo	5.0

Table 1: The cascade tiers, their models, and associated costs, in dollar per million tokens, for the API-based experiments; all API services are provided by together.ai. We use the tiers and models as they are for ABC inference system. However, for the single-model cascade baselines, we use the tiers' best models for their systems.

Since we only have black-box access to these models, we cannot employ a score-based deferral rule. However, we demonstrate that using ABC's voting-based rule defined in Section 4 is also effective in such scenarios. For baseline comparison, we use FrugalGPT, 2 variants of AutoMix, and MoT LLM Cascade. These state-of-the-art cascading methods are specifically designed for scenarios where we make API calls to black-box model endpoints, in contrast to our more generally applicable ABC method. We access the models from Together.ai—described in Table 1—for these experiments, and consider a setting that is advantageous to the baselines, by selecting the *best singular model* from each performance tier in their respective approaches.

All the baseline methods considered here are significantly more complex and involved to set up than ABC; both AutoMix and FrugalGPT involve training a router or deferral rule at each cascade level which has to be repeated for every new task or model change. Unlike AutoMix, FrugalGPT requires training a DistilBERT-based scorer, which would require the user's possession of GPU resources. MoT LLM Cascade generates multiple results and varies the randomness in the LLM's responses via sampling, while using in-context demonstrations and reasoning techniques (e.g., Chain-of-Thought (Wei et al., 2022)) to influence how the model generates answers. In contrast, ABC uses a much simpler voting-based safe deferral rule, without involving additional training or complex routing strategies. Further method-specific details for these experiments as well as additional results can be found in Appendix D.2.

Our results, as shown in Figure 5, demonstrate that ABC is a more reliable deferral rule, and thus, offers a more favorable trade-off between accuracy and cost compared to existing methods—despite their more sophisticated routing mechanisms, and even without considering the additional setup costs incurred by some of the baseline methods. For instance, while FrugalGPT's scorer struggles on harder tasks and tends to take the safer route by deferring more frequently, ABC aggressively leverages cheaper models for a significant portion of inputs, reserving higher-cost models only when necessary. AutoMix, on the other hand, uses a few-shot self-verification system that is sampled k times where k is >1 (in the authors' codebase and ours, $k = 8$); hence, the additional API calls add significantly to its cost of usage. In contrast, ABC easily maintains and often improves on accuracy while being a training-free, simple approach.

[1] together.ai/pricing as of September 2024.
[2] GPT-4-1106-preview costs \$30 as of September 2024.

6 Conclusion

In this work, we introduce Agreement-Based Cascading (ABC) as a straightforward approach for adaptive inference that utilizes existing models for constructing cascades and makes deferral decisions based on their mutual agreement. We define safe deferral rules, ensuring ABC can serve as a drop-in replacement for models while improving accuracy.

Although using an ensemble of models can provide a powerful deferral rule for cascading, the additional costs required to compute such an ensemble may not lead to savings in all inference scenarios. Despite this, our work shows that this simple approach is surprisingly effective given the large differences in model sizes that reach state-of-the-art accuracy in recent ML tasks. We demonstrate improvements via a number of real-world case studies, including a study on communication costs in edge-to-cloud inference, rental costs in cloud-based settings, and the cost of black-box API services. Overall, our results demonstrate ABC's capacity to improve the efficiency of adaptive inference systems without the complexities associated with traditional cascade frameworks, making it a compelling option for practitioners focused on reducing inference latency. In future work, we aim to extend the approach herein to open generation tasks.

References

Mohamed Abdalla, Jan Philip Wahle, Terry Lima Ruas, Aurélie Névéol, Fanny Ducel, Saif M Mohammad, and Karen Fort. The elephant in the room: Analyzing the presence of big tech in natural language processing research. In *The 61st Annual Meeting Of The Association For Computational Linguistics*, 2023.

Anelia Angelova, Alex Krizhevsky, Vincent Vanhoucke, Abhijit S. Ogale, and Dave Ferguson. Real-time pedestrian detection with deep network cascades. volume 2, pp. 4, 2015.

Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-All: Train One Network and Specialize it for Efficient Deployment, April 2019.

Zhaowei Cai, Mohammad Saberian, and Nuno Vasconcelos. Learning Complexity-Aware Cascades for Deep Pedestrian Detection. pp. 3361–3369, 2015.

Lingjiao Chen, Matei Zaharia, and James Y Zou. FrugalML: How to use ML Prediction APIs more accurately and cheaply. In *Advances in Neural Information Processing Systems*, 2020.

Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*, 2023.

Minghao Chen, Houwen Peng, Jianlong Fu, and Haibin Ling. Autoformer: Searching transformers for visual recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 12270–12280, October 2021.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*, 2020.

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

Stephen P Crago and John Paul Walters. Heterogeneous cloud computing: The way forward. *Computer*, 48 (01):59–61, 2015.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

Don Dennis, Abhishek Shetty, Anish Prasad Sevekari, Kazuhito Koishida, and Virginia Smith. Progressive ensemble distillation: Building ensembles for efficient inference. *Advances in Neural Information Processing Systems*, 36, 2023.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*, 2019.

Devvrit, Sneha Kudugunta, Aditya Kusupati, Tim Dettmers, Kaifeng Chen, Inderjit Dhillon, Yulia Tsvetkov, Hannaneh Hajishirzi, Sham Kakade, Ali Farhadi, and Prateek Jain. MatFormer: Nested Transformer for Elastic Inference, October 2023.

Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pp. 1–15. Springer, 2000.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Rühle, Laks VS Lakshmanan, and Ahmed Hassan Awadallah. Hybrid llm: Cost-efficient and quality-aware query routing. In *The Twelfth International Conference on Learning Representations*, 2024.

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

Yilun Du and Leslie Kaelbling. Compositional Generative Modeling: A Single Model is Not All You Need, February 2024. arXiv:2402.01103 [cs].

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Saso Džeroski and Bernard Ženko. Is combining classifiers with stacking better than selecting the best one? *Machine learning*, 54:255–273, 2004.

Shohei Enomoro and Takeharu Eda. Learning to cascade: Confidence calibration for improving the accuracy and computational cost of cascade inference systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 7331–7339, 2021.

Alan Fern and Robert Givan. Online ensemble learning: An empirical study. *Machine Learning*, 53:71–109, 2003.

Farnaz Forooghifar, Amir Aminifar, and David Atienza. Resource-aware distributed epilepsy monitoring using self-awareness from edge to cloud. *IEEE transactions on biomedical circuits and systems*, 2019.

Aditya Gangrade, Anil Kag, and Venkatesh Saligrama. Selective Classification via One-Sided Prediction. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, pp. 2179–2187. PMLR, March 2021. ISSN: 2640-3498.

Yonatan Geifman and Ran El-Yaniv. SelectiveNet: A Deep Neural Network with an Integrated Reject Option. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 2151–2159. PMLR, May 2019. ISSN: 2640-3498.

Shijie Geng, Peng Gao, Zuohui Fu, and Yongfeng Zhang. Romebert: Robust training of multi-exit bert. *arXiv preprint arXiv:2101.09755*, 2021.

Raphael Gontijo-Lopes, Yann Dauphin, and Ekin D. Cubuk. No One Representation to Rule Them All: Overlapping Features of Training Methods, April 2022. arXiv:2110.12899 [cs].

Sachin Goyal, Aditi Raghunathan, Moksh Jain, Harsha Vardhan Simhadri, and Prateek Jain. Drocc: Deep robust one-class classification. In *International conference on machine learning*. PMLR, 2020.

Jiaqi Guan, Yang Liu, Qiang Liu, and Jian Peng. Energy-efficient Amortized Inference with Cascaded Deep Classifiers. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, pp. 2184–2190, Stockholm, Sweden, July 2018. International Joint Conferences on Artificial Intelligence Organization. ISBN 978-0-9992411-2-7.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1321–1330. PMLR, 06–11 Aug 2017.

Neha Gupta, Harikrishna Narasimhan, Wittawat Jitkrittum, Ankit Singh Rawat, Aditya Krishna Menon, and Sanjiv Kumar. Language model cascades: Token-level uncertainty and beyond. In *International Conference on Learning Representations*, 2024.

Muhammad Usman Hadi, Qasem Al Tashi, Abbas Shah, Rizwan Qureshi, Amgad Muneer, Muhammad Irfan, Anas Zafar, Muhammad Bilal Shaikh, Naveed Akhtar, Jia Wu, et al. Large language models: a comprehensive survey of its applications, challenges, limitations, and future prospects. *Authorea Preprints*, 2024.

Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. Dynamic Neural Networks: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):7436–7456, November 2022. ISSN 1939-3539.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*, 2020.

Tom Henighan, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun, Tom B Brown, Prafulla Dhariwal, Scott Gray, et al. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*, 2020.

Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*, 2017.

Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. DynaBERT: Dynamic BERT with Adaptive Width and Depth. In *Advances in Neural Information Processing Systems*, volume 33, pp. 9782–9793. Curran Associates, Inc., 2020.

Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q. Weinberger. Multi-Scale Dense Networks for Resource Efficient Image Classification, June 2018. arXiv:1703.09844.

Ramesh Jetty, Indy Sawhney, and Simon Zamarin. Batch Inference at Scale with Amazon SageMaker | AWS Architecture Blog, November 2021. URL `https://aws.amazon.com/blogs/architecture/batch-inference-at-scale-with-amazon-sagemaker/`.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. LLM-Blender: Ensembling Large Language Models with Pairwise Ranking and Generative Fusion, June 2023.

Wittawat Jitkrittum, Neha Gupta, Aditya K. Menon, Harikrishna Narasimhan, Ankit Rawat, and Sanjiv Kumar. When Does Confidence-Based Cascade Deferral Suffice? *Advances in Neural Information Processing Systems*, 36:9891–9906, December 2023.

Jared Kaplan, Sam McCandlish, T. J. Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeff Wu, and Dario Amodei. Scaling laws for neural language models. *ArXiv*, abs/2001.08361, 2020.

Alind Khare, Dhruv Garg, Sukrit Kalra, Snigdha Grandhi, Ion Stoica, and Alexey Tumanov. SuperServe: Fine-Grained Inference Serving for Unpredictable Workloads, December 2023. arXiv:2312.16733 [cs].

Sehoon Kim, Suhong Moon, Ryan Tabrizi, Nicholas Lee, Michael W. Mahoney, Kurt Keutzer, and Amir Gholami. An llm compiler for parallel function calling. *ArXiv*, abs/2312.04511, 2023.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

Fan Lai, Yinwei Dai, Sanjay Singapuram, Jiachen Liu, Xiangfeng Zhu, Harsha Madhyastha, and Mosharaf Chowdhury. Fedscale: Benchmarking model and system performance of federated learning at scale. In *International conference on machine learning*, pp. 11814–11827. PMLR, 2022.

Lambda. GPU Cloud - VMs for Deep Learning | Lambda, 2024. URL `https://lambdalabs.com/service/gpu-cloud`.

Luzian Lebovitz, Lukas Cavigelli, Michele Magno, and Lorenz K. Muller. Efficient Inference With Model Cascades. *Transactions on Machine Learning Research*, 2023.

Chia-Hsuan Lee, Hao Cheng, and Mari Ostendorf. Orchestrallm: Efficient orchestration of language models for dialogue state tracking. *arXiv preprint arXiv:2311.09758*, 2023.

Baolin Li, Vijay Gadepally, Siddharth Samsi, Mark Veillette, and Devesh Tiwari. Serving machine learning inference using heterogeneous hardware. In *2021 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–8. IEEE, 2021a.

Baolin Li, Siddharth Samsi, Vijay Gadepally, and Devesh Tiwari. Kairos: Building cost-efficient machine learning inference systems with heterogeneous cloud resources. In *Proceedings of the 32nd International Symposium on High-Performance Parallel and Distributed Computing*, pp. 3–16, 2023.

Lei Li, Yankai Lin, Deli Chen, Shuhuai Ren, Peng Li, Jie Zhou, and Xu Sun. CascadeBERT: Accelerating Inference of Pre-trained Language Models via Calibrated Complete Models Cascade. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 475–486, Punta Cana, Dominican Republic, November 2021b. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-emnlp.43.

Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Haotang Deng, and Qi Ju. Fastbert: a self-distilling bert with adaptive inference time. *arXiv preprint arXiv:2004.02178*, 2020.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Aman Madaan, Pranjal Aggarwal, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappaganthu, Yiming Yang, Shyam Upadhyay, Mausam, and Manaal Faruqui. AutoMix: Automatically Mixing Language Models, October 2023. arXiv:2310.12963 [cs].

Jonathan Mamou, Oren Pereg, Moshe Wasserblat, and Roy Schwartz. TangoBERT: Reducing Inference Cost by using Cascaded Architecture, April 2022. arXiv:2204.06271 [cs].

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Hongyi Jin, Tianqi Chen, and Zhihao Jia. Towards efficient generative large language model serving: A survey from algorithms to systems. *ArXiv*, abs/2312.15234, 2023.

Hao Mo, Ligu Zhu, Lei Shi, Songfu Tan, and Suping Wang. Hetsev: Exploiting heterogeneity-aware autoscaling and resource-efficient scheduling for cost-effective machine-learning model serving. *Electronics*, 12(1):240, 2023.

Harikrishna Narasimhan, Wittawat Jitkrittum, Aditya K. Menon, Ankit Rawat, and Sanjiv Kumar. Post-hoc estimators for learning to defer to an expert. *Advances in Neural Information Processing Systems*, 35: 29292–29304, December 2022.

Lunyiu Nie, Zhimin Ding, Erdong Hu, Christopher Jermaine, and Swarat Chaudhuri. Online cascade learning for efficient inference over streams. *arXiv preprint arXiv:2402.04513*, 2024.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E Gonzalez, M Waleed Kadous, and Ion Stoica. Routellm: Learning to route llms with preference data. *arXiv preprint arXiv:2406.18665*, 2024.

Yulong Pei, Amarachi B. Mbakwe, Abhinav Gupta, Salwa Alamir, Huang Lin, Xiaomo Liu, and Sameena Shah. Tweetfinsent: A dataset of stock sentiments on twitter. 2021.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PMLR, 2021.

Siva Reddy, Danqi Chen, and Christopher D Manning. Coqa: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266, 2019.

H.A. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):23–38, January 1998. ISSN 1939-3539. doi: 10.1109/34.655647.

Marija Šakota, Maxime Peyrard, and Robert West. Fly-swat or cannon? cost-effective language model choice via meta-modeling. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pp. 606–615, 2024.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *ArXiv*, October 2019.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Tran, Yi Tay, and Donald Metzler. Confident Adaptive Language Modeling. *Advances in Neural Information Processing Systems*, 35: 17456–17472, December 2022.

Amanda J. C. Sharkey. On combining artificial neural nets. *Connection Science*, 8(3-4):299–314, 1996. doi: 10.1080/095400996116785.

Tal Shnitzer, Anthony Ou, Mírian Silva, Kate Soule, Yuekai Sun, Justin Solomon, Neil Thompson, and Mikhail Yurochkin. Large Language Model Routing with Benchmark Datasets, September 2023. arXiv:2309.15789 [cs].

Ankur Sinha and Tanmay Khandait. Impact of news on the commodity market: Dataset and results. In *Advances in Information and Communication: Proceedings of the 2021 Future of Information and Communication Conference (FICC), Volume 2*, pp. 589–601. Springer, 2021.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642, 2013.

Sander Soo. Object detection using Haar-cascade Classifier. *Institute of Computer Science, University of Tartu*, 2(3):1–12, 2014.

Matthew Streeter. Approximation Algorithms for Cascading Prediction Models. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 4752–4760. PMLR, July 2018. ISSN: 2640-3498.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for modern deep learning research. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 13693–13696, 2020.

Arun Suggala, Bingbin Liu, and Pradeep Ravikumar. Generalized boosting. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 8787–8797, 2020.

Tianxiang Sun, Yunfan Shao, Hong Qian, Xuanjing Huang, and Xipeng Qiu. Black-box tuning for language-model-as-a-service. In *International Conference on Machine Learning*, pp. 20841–20855. PMLR, 2022.

Teo Susnjak, Andre L. C. Barczak, and Kenneth A. Hawick. Adaptive cascade of boosted ensembles for face detection in concept drift. *Neural Computing and Applications*, 21:671–682, 2012.

C Szegedy. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

Neeraj Varshney and Chitta Baral. Model Cascading: Towards Jointly Improving Efficiency and Accuracy of NLP Systems. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 11007–11021, Abu Dhabi, United Arab Emirates, 2022. Association for Computational Linguistics.

P. Viola and M. Jones. 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, December 2001. ISSN: 1063-6919.

Paul Viola and Michael J. Jones. Robust Real-Time Face Detection. *International Journal of Computer Vision*, 57(2):137–154, May 2004. ISSN 1573-1405. doi: 10.1023/B:VISI.0000013087.49260.fb.

Lidan Wang, Jimmy Lin, and Donald Metzler. A cascade ranking model for efficient ranked retrieval. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pp. 105–114. ACM, 2011.

Xiaofang Wang, Dan Kondratyuk, Eric Christiansen, Kris M Kitani, Yair Movshovitz-Attias, and Elad Eban. Wisdom of committees: An overlooked approach to faster and more accurate models. In *International Conference on Learning Representations*, 2021.

Xin Wang, Yujia Luo, Daniel Crankshaw, Alexey Tumanov, Fisher Yu, and Joseph E. Gonzalez. IDK Cascades: Fast Deep Learning by Learning not to Overthink, June 2018a. arXiv:1706.00885 [cs].

Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E. Gonzalez. SkipNet: Learning Dynamic Routing in Convolutional Networks. pp. 409–424, 2018b.

Yiding Wang, Kai Chen, Haisheng Tan, and Kun Guo. Tabi: An efficient multi-level inference system for large language models. In *Proceedings of the Eighteenth European Conference on Computer Systems*, pp. 233–248, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. Deebert: Dynamic early exiting for accelerating bert inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 2246–2251, 2020.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Neural Information Processing Systems*, 2019.

Jiahui Yu and Thomas S. Huang. Universally Slimmable Networks and Improved Training Techniques. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1803–1811, 2019.

Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable Neural Networks, December 2018. arXiv:1812.08928 [cs].

Murong Yue, Jie Zhao, Min Zhang, Liang Du, and Ziyu Yao. Large language model cascades with mixture of thought representations for cost-efficient reasoning. In *The Twelfth International Conference on Learning Representations*, 2024.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. SWAG: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 93–104, 2018.

Lucia Zheng, Neel Guha, Brandon R Anderson, Peter Henderson, and Daniel E Ho. When does pretraining help? assessing self-supervised learning for law and the casehold dataset of 53,000+ legal holdings. In *Proceedings of the eighteenth international conference on artificial intelligence and law*, pp. 159–168, 2021.

Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. Bert loses patience: Fast and robust inference with early exit. *Advances in Neural Information Processing Systems*, 33:18330–18341, 2020.

Ligeng Zhu, Hongzhou Lin, Yao Lu, Yujun Lin, and Song Han. Delayed gradient averaging: Tolerate the communication latency for federated learning. *Advances in Neural Information Processing Systems*, 34: 29995–30007, 2021.

Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. A survey on model compression for large language models. *arXiv preprint arXiv:2308.07633*, 2023.

Fei Zuo and Peter H. N. de With. Fast face detection using a cascade of neural network ensembles. In *Advanced Concepts for Intelligent Vision Systems Conference*, 2005.

Fei Zuo and Peter H. N. de With. Cascaded face detection using neural network ensembles. *EURASIP Journal on Advances in Signal Processing*, 2008:1–13, 2008.

A Agreement-Based Cascading (ABC) can Improve Accuracy

Our construction of ABC, as explained in Section 3 and 4 is based on *safe deferral rules* (Definition 4.1) — rules such that when a data point is selected at a lower cascade tier, the inference is accurate with high probability. This means that if the lower tier of the cascade has a higher accuracy compared to the largest model, on the 'easy' data it predicts on, the overall accuracy of ABC can increase. For 'hard' samples ABC and the largest, SoTA model produces identical predictions as ABC is internally deferring to this model. We formalize this notion of accuracy improvement in this section.

Consider a classification problem in the statistical learning framework, where $\mathcal{X}$ is our instance space and $\mathcal{Y}$ is the label space. Let r denote any deterministic deferral rule $r : \mathcal{X} \to \{0, 1\}$. Let $\{H_1, h_2\}$ be two classifiers $h_i : \mathcal{X} \to \mathcal{Y}$ for $i \in \{1, 2\}$. We will think of h_2 as being the more expensive (better accuracy) model. Consider a distribution P over $\mathcal{X} \times \mathcal{Y}$. The risk of classifier h_2 is defined as

$$\mathcal{R}(h_2) = P(h_2(x) \neq y)$$

Let us define a cascaded classifier $\mathcal{M}_r : \mathcal{X} \to \mathcal{Y}$ such that,

$$\mathcal{M}_r(x) = \begin{cases} H_1(x) & r(x) = 0 \\ h_2(x) & r(x) = 1. \end{cases} \tag{5}$$

Then the risk of the cascaded classifier using rule r is,

$$\mathcal{R}(\mathcal{M}_r) = P(\mathcal{M}_r(x) \neq y)$$

We wish to understand when we can replace an existing large, expensive classifier h_2 with a cascade such that there is no drop in accuracy. This is useful when we are extremely sensitive towards accuracy drop and wish to only select models at the lower-level when we are certain about the classification. Let us define the excess risk of using a cascade in-place of the larger model h_2 as,

$$\mathcal{R}_{\text{excess}}(\mathcal{M}_r, h_2) = \mathcal{R}(\mathcal{M}_r) - \mathcal{R}(h_2).$$

Expanding further, we can express the excess risk as,

$$\begin{aligned}
\mathcal{R}_{\text{excess}}(\mathcal{M}_r, h_2) &= P(\mathcal{M}_r(x) \neq y) - P(h_2(x) \neq y) \\
&= P(\mathcal{M}_r(x) \neq y \mid r(x) = 0)P(r(x) = 0) + P(\mathcal{M}_r(x) \neq y \mid r(x) = 1)P(r(x) = 1) \\
&\quad - P(h_2(x) \neq y) \\
&= P(H_1(x) \neq y \mid r(x) = 0)P(r(x) = 0) + P(h_2(x) \neq y \mid r(x) = 1)P(r(x) = 1) \\
&\quad - P(h_2(x) \neq y \mid r(x) = 0)P(r(x) = 0) + P(h_2(x) \neq y \mid r(x) = 1)P(r(x) = 1) \\
&= \Big(P(H_1(x) \neq y \mid r(x) = 0) - P(h_2(x) \neq y \mid r(x) = 0)\Big)P(r(x) = 0)
\end{aligned} \tag{6}$$

As can be seen, the general excess risk here depends on both models. Since our focus is on accuracy, any rule that leads to a risk that is no worse than that of the classifier h_2 can be used, and we term these as *admissible* cascades.

Definition A.1 (Admissible cascades)**.** *$M_r(x)$ defined above is an admissible cascade w.r.t the population distribution P and the reference classifier h_2 if,*

$$\mathcal{R}_{excess}(M_r, h_2) \leq 0,$$

or equivalently

$$P(H_1(x) \neq y \mid r(x) = 0) \leq P(h_2(x) \neq y \mid r(x) = 0).$$

As mentioned in Section 4, for this work, we specialize to the case where whenever the smaller model is selected, it is always correct. That is, $P(H_1(x) \neq y \mid r(x) = 0) = 0$. Of course, such a pair of smaller-classifier H_1 and deferral rule r need not exists. However, whenever they do, a cascade constructed using *any* h_2 is admissible since,

$$\forall h_2, R(M_r, h_2) = -P(h_2(x) \neq y \mid r(x) = 0)P(r(x) = 0) \leq 0.$$

This implies a *universal nature* of such (H_1, r) for such two tier cascades as the excess risk of the cascade does not depend on the larger classifier h_2.

Finally, if the ensemble $H_1(x)$ outperforms the SoTA model on the 'easy' samples by some strictly positive $\xi > 0$, such that

$$P(H_1(x) \neq y \mid r(x) = 0) \leq P(h_2(x) \neq y \mid r(x) = 0) - \xi$$

Then, from Equation 6

$$\begin{aligned}\mathcal{R}_{\text{excess}}(\mathcal{M}_r, h_2) &= \Big(P(H_1(x) \neq y \mid r(x) = 0) - P(h_2(x) \neq y \mid r(x) = 0)\Big)P(r(x) = 0) \\ &= -\xi P(r(x) = 0) < 0,\end{aligned}$$

wherever the selection rate satisfies $P(r(x) = 0) \geq 0$. Negative excess risk implies an improvement over the SoTA classifier we were using with the overall amount of improvement depending on $P(r(x) = 0)$.

B Estimating Agreement Threshold θ

To effectively apply `ABC`, it's essential to configure an agreement threshold θ for the deferral rules defined in Equations 3 and 4. This threshold indicates a sufficient level of confidence in the ensemble's predictions, allowing `ABC` to avoid deferring to a higher-cost model when the ensemble's agreement is high, thereby reducing inference costs without sacrificing accuracy. Recall from Definition 4.1 that the failure rate of a deferral rule, as a function of the agreement threshold is given by,

$$\mathbb{P}(s(x) \geq \theta, H_k(x) \neq y).$$

In particular, safe deferral rules are those with a failure rate bounded by a small $\epsilon \geq 0$ of our choice. This means that, given a distribution $\mathbb{P}$ over $\mathcal{X} \times \mathcal{Y}$ we can define a function $p(\theta)$ as,

$$p(\theta) = \mathbb{P}(s(x) \geq \theta, H_k(x) \neq y).$$

We can now define feasible thresholds, θ as those for which the error rate $p(\theta) \leq \epsilon$, since any such θ leads to safe deferral. In practice, we rarely have access to $\mathbb{P}$ and therefore $p(\theta)$. We instead use its plugin-estimator, given by

$$\hat{p}(\theta) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}[s(x) \geq \theta, H_k(x) \neq y].$$

We estimate $\hat{p}(\theta)$ by using a small subset of samples from the validation set. Typically, we draw around 100 samples and set them aside to determine a stable threshold. Figure 6 shows how the estimated threshold varies with the number of samples used in the estimation process, across models with different accuracy

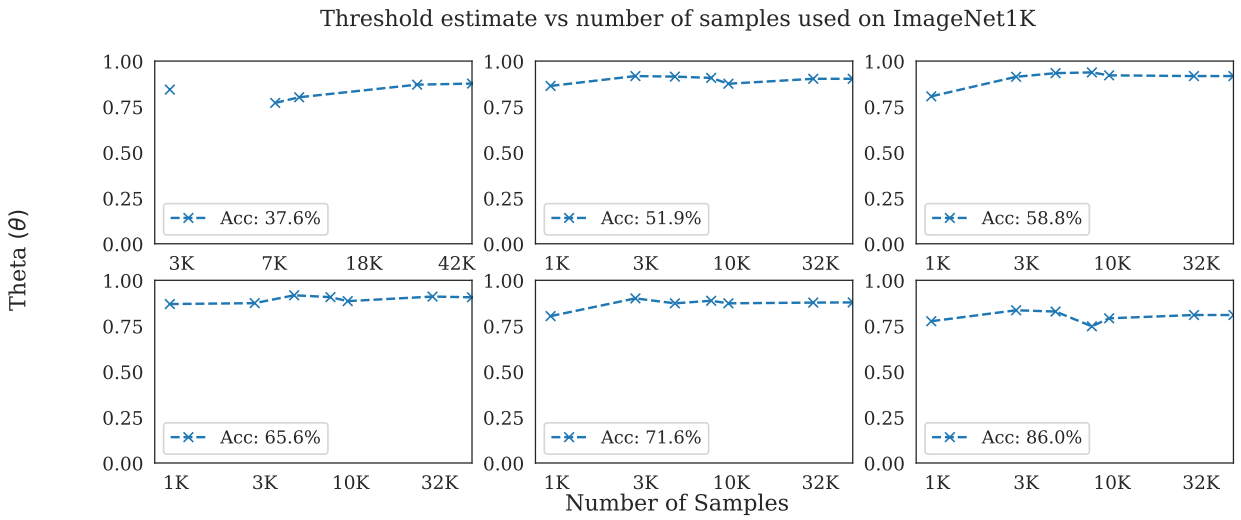


Figure 6: Estimation of agreement threshold θ stability as a function of the number of samples used, across different model accuracy levels on the ImageNet1k dataset. Each plot corresponds to a model with a specific accuracy, shown in the legend. The initial estimate is with 100 samples, with subsequent estimates with larger and larger number of samples. The initial estimate is reasonably close within subsequent estimates with larger number of samples.

levels. Each plot represents a model of a specific accuracy, with accuracy percentages indicated in the legend. As shown, the estimate generally stabilizes even with 100 samples, suggesting that only a few validation samples are needed to reliably estimate θ, making the parameter estimation process efficient and practical for real-world applications.

C Existence of Safe Deferral Rules and Selection Rates

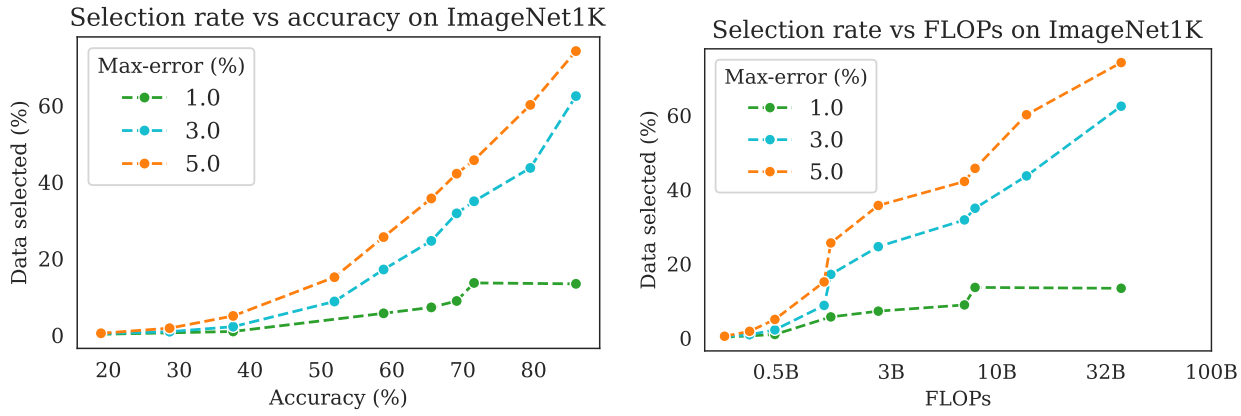


Figure 7: Selection rate as a function of accuracy (left) and FLOPs (right) for different error tolerances on ImageNet1k. The selection rate $\mathbb{P}(r(x) \geq \theta)$ represents the fraction of data handled at a lower cascade tier without deferring to a larger model, based on a threshold θ. Laxer error tolerances (e.g., 5%) yield higher selection rates, as more samples meet the criteria for safe deferral. In contrast, stricter tolerances (e.g., 1%) result in lower selection rates. Both plots illustrate that higher-accuracy or higher-FLOP models generally achieve higher selection rates, especially as the allowable error tolerance increases, reinforcing the stability and practicality of `ABC`.

This section examines the existence of safe deferral rules — rules that have a very low probability of being incorrect, where data is not deferred to a larger model — by evaluating selection rates. For any threshold θ, the selection rate $P(r(x) \geq \theta)$ represents the fraction of data that deemed 'easy' enough to be processed at a

lower tier of the cascade. Experimentally, we observe that θ_ϵ computed in the manner described in Section B does indeed adhere to the error tolerance of choice ϵ. This means that whenever a data point is selected at a lower tier, the inference is accurate with probability $1 - \epsilon$. Stricter values of ϵ typically result in a higher threshold for agreement and a lower-selection rate.

The plots in Figure 7 show the selection rate across different accuracy levels and FLOP values on the ImageNet1k dataset. We explore selection rates for three error tolerances, corresponding to maximum allowable error rates of 1%, 3%, and 5%. As shown in the left plot, which compares selection rate versus accuracy of the models used in the ensemble, higher accuracy models tend to achieve higher selection rates, especially when the error tolerance is relaxed (e.g., 5% maximum error). Conversely, stricter error tolerances (e.g., 1%) lead to lower selection rates across all accuracy levels, as fewer samples meet the stringent requirements for safe deferral at lower cascade tiers. In the right plot, which illustrates selection rate versus FLOPs, a similar trend is observed. Higher-FLOP models, which are generally more accurate, are able to safely handle a larger portion of the data at lower tiers, particularly as the error tolerance increases.

D Evaluation Setups

D.1 Datasets and Models

Datasets and models used in our experiments are detailed in Table 2 and Table 3.

Category	Dataset	Task Type
Image Tasks	ImageNet-1K (Deng et al., 2009)	Image classification
	CIFAR-10 (Krizhevsky & Hinton, 2009)	Image classification
Language Tasks	SST-2 (Socher et al., 2013)	Sentiment analysis
	Twitter Financial News (Pei et al., 2021)	Sentiment analysis
	SWAG (Zellers et al., 2018)	Multiple-Choice QA
Black-Box Experiments	GSM8K (Cobbe et al., 2021)	Math Reasoning
	COQA (Reddy et al., 2019)	Conversational QA
	OVERRULING (Zheng et al., 2021)	Legal Reasoning
	HEADLINES (Sinha & Khandait, 2021)	News Classification

Table 2: Datasets used in both benchmark and black-box API experiments across various task types.

Category	Dataset	Tiers Used
Language Models	BERT (Devlin et al., 2019)	`BASE`, `LARGE`
	RoBERTa (Liu et al., 2019)	
	XLNet (Yang et al., 2019)	
	ELECTRA (Radford et al., 2021)	
Image Models	ResNet (He et al., 2016)	Selection based on FLOPs
	ViT (Dosovitskiy, 2020)	
	CLIP (Clark et al., 2020)	
Black-Box Models	LlaMA 3.1 (Dubey et al., 2024)	See Table 1
	Gemma 2 (Team et al., 2024)	
	Qwen 2 (Yang et al., 2024)	

Table 3: Summary of models used for both benchmark and black-box API experiments, across image and language tasks.

D.2 Details for Black-Box API Experiments

Baselines We compare ABC to FrugalGPT (Chen et al., 2023), 2 variants of AutoMix (Madaan et al., 2023), and MoT LLM Cascade (Yue et al., 2024). Although HybridLLM (Ding et al., 2024) falls into this category of state-of-the-art cascade methods, it has been shown to underperform FrugalGPT and AutoMix (Madaan et al., 2023). For practical evaluation, we implemented all these baseline methods to work in a fully functional cascade system.

Method-Specific Details

- **AutoMix**: AutoMix trains a different router for all possible cascade steps, i.e., $n-1$ routers for the n cascade tiers, and this has to be repeated for every new task or model replacement in the system. A threshold (AutoMix+T) or POMDP (AutoMix+P) is trained with a combination of ≥ 50 training samples from the same test data distribution and the initial inferences generated on the test data by the two models involved in each cascading step. After training the router, using the cascading system often involves running a few-shot self-verification 8 times at a high sampling temperature (temp = 1.0), using the same model that generates inference at the given cascade tier. Automix then averages the self-verification results and meta-verify it with the best parameters of the routing strategy to decide when to exit.
- **FrugalGPT**: Just like AutoMix, FrugalGPT needs to train $n-1$ routers for n cascade tiers, and each router needs to have a sense of the data distribution and the model's predictive power. ≥ 500 training samples and inference generated on these samples by the tier's model are needed to train each tier/model's scorer, a DistilBERT (Sanh et al., 2019).
- **MoT LLM Cascade**: Yue et al. (2024) focuses on sampling and consistency checking as a means of cascade. To measure consistency, the weaker LLM generates multiple answers for a single question by varying the randomness in the LLM's responses—i.e., varying the temperature of the model—while using in-context demonstrations and reasoning techniques (e.g., Chain-of-Thought (Wei et al., 2022)) to influence how the model generates answers. The system compares the different sampled answers and picks the most consistent one. If the consistency score is high enough, the weaker model's answer is accepted. Otherwise, the question is passed to the next tier.
- **ABC**: We use the voting-based safe deferral rule, requiring no additional training or any complex routing strategies.

Cascade Models We use the models described in Table 1. We consider a setting that is advantageous to the baselines by selecting the *best singlular model* from each performance tier and cascading between them. In all cascading systems, we use all three tiers for cascade; but considering budget constraints, we also have setups where we delete Tier 3 and use only the first two tiers (2-level cascade).

Evaluation Setup We evaluate these methods on a variety of (closed) generation datasets and tasks as shown in Table 2. To ensure consistency in output format and easier evaluation setup, we use few-shot prompting—specifically, 4-shot—for all models across all tasks. For evaluation metrics, we use the macro F1 score for CoQA to capture overlaps between predictions and ground-truth answers, while we measure accuracy (essentially exact match) for the rest of the tasks. In terms of efficiency, we also measure the costs of using the model APIs. It is important to note that AutoMix and FrugalGPT incur additional setup costs that we did not reflect in our reported results. These setup costs and associated latency represent a significant constraint, especially for scenarios requiring frequent retraining or adaptation to new tasks, data distributions, and models.

As shown in Figure 5, we observe that ABC is often more cost-effective than the baselines, even with their more sophisticated routing mechanisms and singular model tiers. Our analyses show that ABC's efficient deferral strategy allows a more aggressive utilization of cheaper models for a significant portion of the input while only using expensive models when necessary. For instance, we realize—upon analyzing FrugalGPT's results—that the trained scorer struggles as an efficient deferral mechanism as the tasks get harder; hence, it

is more likely to take the safer option to cascade as the difficulty of the test samples increases. This means that ABC can be expected to be more efficient since the scaling laws ensure that the sum of the costs of using several cheaper models is still much less than the cost of using the larger model in the next tier. AutoMix, on the other hand, uses a few-shot self-verification system that is sampled > 1 times; hence, the additional API calls add significantly to its cost of usage. Considering that the self-verification process is an integral part of the AutoMix setup, it can be guaranteed that ABC will *always* be cheaper to use than AutoMix, despite using more models.

E Benefits of ABC

E.1 Parallel vs. sequential inference execution

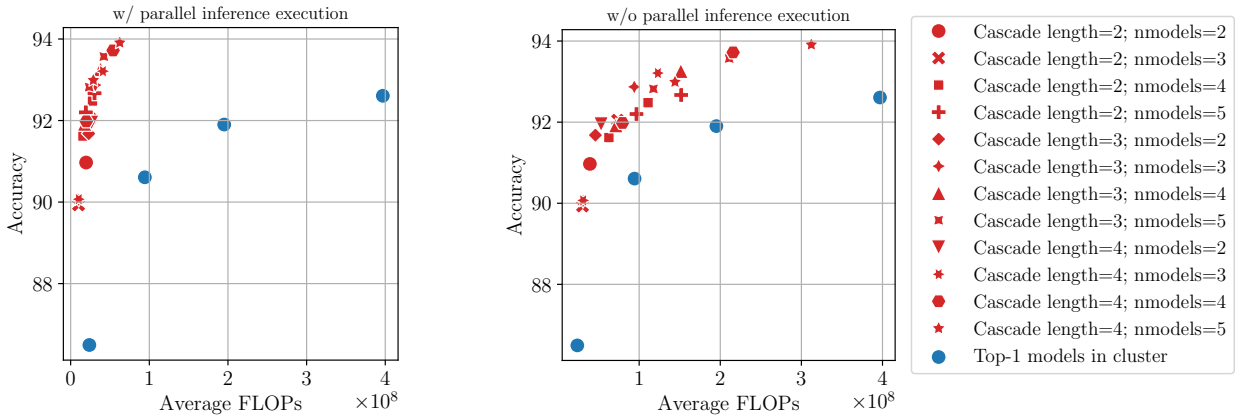


Figure 8: Inference execution in parallel vs non-parallel on CIFAR-10.

Based on Section 5.1.1, we additionally show in Figure 8 the superiority of parallel inference execution for cascading over using the best single model using CIFAR-10 as a case study. We also show that in the worst-case scenarios in which every single inference has to be sequentially produced over each ensemble and cascade, there are still considerable savings over the largest single models.

E.2 Cost Benefits

Based on Section 5.2.2, Table 4 shows the GPUs' pricing across several tiers retrieved from Lambda Cloud. Table 5 shows a detailed analysis of costs across all cascade tiers, associated with the number of cascade exits at each cascade tier, to provide holistic efficiency comparisons of the aggregated cascade costs against using the best (and largest) model — and ABC dominates in every measured metric. Typically, most cascade exits occur in the earlier (and much cheaper) tiers (as also shown in Table 5), ensuring that the more cost-intensive cascade tiers featuring are reserved for the harder test instances.

GPU	Cost per Hour (USD)
V100	0.5
A6000	0.8
A100	1.29
H100	2.49

Table 4: GPU usage costs estimates from Lambda Cloud (Lambda, 2024), as of September 2024.

Table 5: Table describing various metrics at different cascade tiers for each dataset. Each dataset's row consists of the following metrics: the fraction of samples processed at each depth, the cost of Total GPU usage per hour at each tier, the average latency of the specific tier (in milliseconds), and the average FLOPs at each tier (floating-point operations). The `ABC` column for each metric represents the entire cascading system's average across all cascade depths.

Dataset	Metric	Tier 1	Tier 2	Tier 3	Tier 4	ABC	Best Single Model
CIFAR-10	Frac. Samples (total=10,000)	0.73	0.09	0.08	0.10	1.00	1.00
	Total GPU Cost ($ / hour)	0.36	0.07	0.11	0.24	0.79	2.49
	Avg. Latency (ms)	3.11	3.79	7.76	9.07	4.13	9.07
	Avg. FLOPs	5.42e6	2.32e7	1.16e8	2.47e8	3.97e7	2.48e8
ImageNet-1K	Frac. Samples (total=50,000)	0.52	0.29	0.19	-	1.00	1.00
	Cost ($ / hour)	0.26	0.23	0.25	-	0.74	1.29
	Avg. Latency (ms)	2.45	2.88	3.17	-	2.71	3.17
	Avg. FLOPs	2.15e9	3.90e9	4.30e9	-	3.07e9	4.30e9
SWAG (MCQ)	Frac. Samples (total=20,006)	0.71	0.29	-	-	1.00	1.00
	Cost ($ / hour)	0.36	0.23	-	-	0.59	0.80
	Avg. Latency (ms)	4.52	8.05	-	-	5.53	8.05
	Avg. FLOPs	1.88e10	6.67e10	-	-	3.25e10	6.67e10
SST-2	Frac. Samples (total=872)	0.93	0.07	-	-	1.00	1.00
	Cost ($ / hour)	0.46	0.06	-	-	0.52	0.80
	Avg. Latency (ms)	3.88	7.22	-	-	4.13	7.22
	Avg. FLOPs	5.43e9	1.68e10	-	-	6.26e9	1.68e10
Twitter Fin News	Frac. Samples (total=822)	0.65	0.35	-	-	1.00	1.00
	Cost ($ / hour)	0.32	0.28	-	-	0.61	0.80
	Avg. Latency (ms)	4.05	7.26	-	-	5.19	7.26
	Avg. FLOPs	6.83e9	2.42e10	-	-	1.30e10	2.42e10