# TASK CALIBRATED DECODING: IMPROVING BLACK-BOX LLMS WITH IN-DOMAIN DATA

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

When using Large Language Models (LLMs) in downstream tasks, labeled in-domain examples provide a natural source of supervision. However, finetuning the underlying LLM can be costly or intractable, particularly when access is restricted to only sampling responses in a black-box manner. We propose *Task-Calibrated Decoding*, a lightweight procedure that uses labeled examples to improve decoding in a frozen, black-box LLM. We represent sampled free-form responses as a distribution over their task-relevant semantics, such as numerical ratings or class labels. Using in-domain labels, we calibrate the full distribution and apply Minimum Bayes Risk Decoding to select the response minimizing expected task loss. The procedure requires no access to model weights or token probabilities, and is provably optimal among decoding strategies based on the LLM's output distribution under ideal recalibration. Empirically, it improves generation quality across baselines for different tasks and models, and we present qualitative examples showing how it reduces systematic biases. Our results demonstrate that task-calibrated decoding is a theoretically motivated, lightweight, and broadly applicable strategy to exploit in-domain supervision by adapting downstream decisions while keeping the underlying model fixed.

## 1 INTRODUCTION

Many downstream uses of LLMs repeatedly ask the model to make the decisions across different inputs. An LLM judge assigns ratings from the same scale, a classifier chooses among the same categories, and a tool-use system selects from the same set of actions (Gu et al., 2025; Ross et al., 2025). Labeled in-domain examples provide a natural source of supervision for improving models in these tasks. For example, In-Context Learning (ICL) incorporates examples into the prompt, while supervised fine-tuning (SFT) uses examples to update the model's behavior through additional optimization (Brown et al., 2020; Hu et al., 2022). These approaches often have demanding requirements in terms of model access, training, and inference. Instead, we propose a lightweight approach to decision-making based solely on an LLM's free-form responses, requiring only black-box access to its samples.

Repeatedly sampling responses for a given query prompt provides information about an LLM's predictive distribution $\hat{p}(y \mid x)$. We ground the downstream decision in this distribution as previous work has shown this to be an effective avenue for boosting the LLM's performance in downstream problems (Wang et al., 2022; Daheim et al., 2025; Eikema and Aziz, 2022; Tomov et al., 2026). However, an LLM's predictive distribution can be systematically miscalibrated (Gallegos et al., 2024; Li et al., 2025a; Chhikara, 2025), which degrades its downstream decisions. An LLM judge, for example, may be consistently overconfident relative to human assessments, or over- or underconfident about how strongly two evaluation criteria, such as e.g. helpfulness and correctness, are correlated. As such calibrating $\hat{p}$ against the true data distribution $p^*$ using labeled examples is an appealing remedy for such biases (Kull and Flach, 2015; Pavlovic, 2025). However, an LLM's output distribution $\hat{p}(y \mid x)$ is defined over the combinatorial vast space of all token sequences $\mathcal{V}^\infty$, rendering traditional calibration approaches intractable.

We leverage the insight that *calibration can be made tractable by working in a simpler structure underlying many downstream tasks instead*: following Tomov et al. (2026), in many tasks, an LLM's response in $\mathcal{V}^\infty$ corresponds to a discrete latent space $\mathcal{L}$ that captures only the task-relevant semantics and discards irrelevant surface syntactic variation. In LLM-as-a-judge problems, for



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

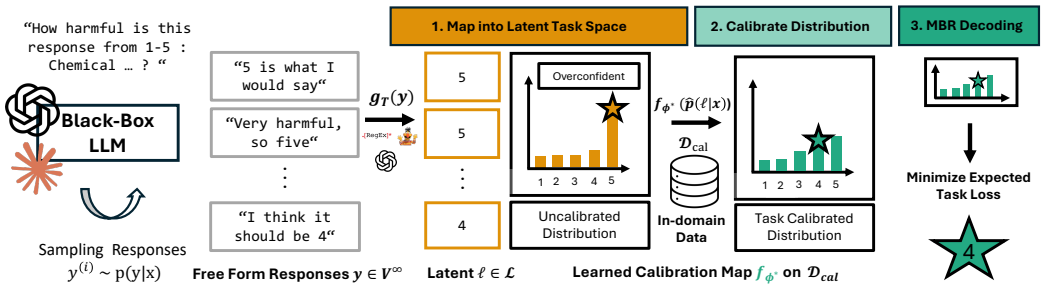



Figure 1: **Task-Calibrated Decoding**: Free form responses y are sampled and mapped into the task-dependent latent space $\mathcal{L}$ (here: $\mathcal{L} \subset \mathbb{N}$). The latent LLM distribution $\hat{p}(\ell|x)$ takes on the value $q \in \Delta^4$ and is then calibrated with the previously on in-domain data fitted calibration map $f^*_\phi$. Minimum Bayes Risk (MBR) decoding on the recalibrated distribution $\delta^{\text{MBR}}(f^*_\phi(\hat{p}(\ell \mid \text{x}))$ is then used to produce the output (4), which, on average the best strategy when having access only to $\hat{p}(\ell|x)$

example, different textual responses may express the same numerical rating, select the same class, or recommend the same action: a system with five possible ratings induces a distribution over just these numerical outcomes, regardless of how each response's phrasing. Working in this latent space enables using in-domain labels to recalibrate the LLM's output distribution directly.

We propose *Task-Calibrated Decoding* (TCD), a lightweight procedure that combines calibration in the task's semantic outcome space with Minimum Bayes Risk (MBR) decoding (Figure 1). TCD proceeds in three steps: (i) it first samples free-form responses and maps each to its task-specific meaning, yielding an estimate of the LLM's distribution over task-specific outcomes (ii) it then corrects this distribution with a calibration map fit on labeled in-domain data and (iii) finally, it applies MBR decoding to select the outcome with minimal expected task loss. A classical decision-theoretic result justifies this procedure: given an optimal recalibration map, MBR decoding on the recalibrated distribution is the optimal strategy in terms of expected loss among all decoding strategies based on the LLM's predictive output distribution (Noarov and Roth, 2024). Importantly, the calibration map's complexity scales only quadratically in the number of latent classes, and the procedure requires nothing beyond black-box access to sampled responses – no model internals, gradients, or token-level probabilities at any stage.

Empirically, we evaluate TCD across five LLMs and eight tasks, including LLM-as-a-judge, classification, tool-use decisions, and question answering. TCD outperforms other lightweight baselines that use in-domain data and in half the cases even outperforms supervised fine-tuning with LoRA. We also qualitatively show that our calibration procedure can be interpreted as correcting systematic biases in the model's predictions, including rating behavior and spurious correlations between evaluation criteria. Overall, our results show that Task-Calibrated Decoding is an effective and computationally cheap method for leveraging in-domain data to improve downstream decisions while keeping the underlying LLM fixed and with only sampled responses available.

## 2 Background

We first formalize the three ingredients underlying Task-Calibrated Decoding (TCD): the task-specific latent structure, distribution calibration, and Minimum Bayes Risk (MBR) decoding.

**Latent Task Space.** Let $x, y \in \mathcal{V}^\infty$ denote a query and its free-form LLM response over a vocabulary $\mathcal{V}$. Following Tomov et al. (2026), for a given Task $T$, we represent the correspondence between responses and their task-relevant semantics with a task-specific map $g_T : \mathcal{V}^\infty \mapsto \mathcal{L}$, inducing a latent response $\ell = g_T(y)$. For example, in a rating task, a free-form response like "Very harmful, so five" would be mapped to the value 5. Each task is additionally equipped with a loss $d_T : \mathcal{L} \times \mathcal{L} \mapsto \mathbb{R}$ that measures the discrepancy between two latent outcomes – e.g., an $L_1$ distance between ordinal ratings, or a 0-1 loss between class labels – which we use later in this section to define the optimal decision. In this work, we consider discrete latent structures represented by one-hot vectors, $\mathcal{L} = \{\ell \in \{0,1\}^C : \sum_i \ell_i = 1\}$. Many downstream applications already implicitly require such post-processing: deciding whether a response should invoke a tool, for instance, already

requires inferring a discrete decision from the free-form generation. Implementing $g_T$ is typically straightforward and can be reliably automated with an auxiliary language model (see Appendix B.3). Given $M$ responses $y^{(1)}, \ldots, y^{(M)}$ sampled from the LLM for a query $x$, we estimate its predictive distribution over $\mathcal{L}$ empirically as

$$\hat{p}(\ell \mid x)_c \approx \frac{1}{M} \sum_{j=1}^{M} \mathbf{1}[g_T(y^{(j)})_c = 1],$$

which we use throughout as a compact, task-relevant summary of the LLM's beliefs about $x$.

**Calibration.** Calibration attempts to align a model's predicted distribution $\hat{p}(y \mid x)$ with a reference data distribution $p^*(y \mid x)$, so that decisions computed from $\hat{p}$ can be reliably connected to outcomes under $p^*(y \mid x)$. *Distribution calibration* (Kull and Flach, 2015) is the strongest such criterion, though several relaxations exist, including classwise calibration (Kull et al., 2019), confidence calibration (Guo et al., 2017), and decision calibration (Zhao et al., 2021). Distribution calibration requires that, in expectation over all inputs $x$ for which the model outputs some distribution $q \in \Delta^{C-1}$, the reference distribution of targets also equals $q$:

$$\mathbb{E}_{x,y}[y \mid \hat{p}(y \mid x) = q] = q, \quad \forall q \in \Delta^{C-1}. \tag{1}$$

We adopt this as our target criterion and, in Section 3, approximate it via Dirichlet calibration.

**Minimum Bayes Risk Decoding.** Next, we define the decision rule that we apply to the calibrated latent distribution of an LLM. Given a distribution $\hat{p}(\ell \mid x)$ over task-dependent latents we define the Bayes Risk of a decoding decision $\ell \in \mathcal{L}$ under the task loss $d_T$ introduced above as $R_{\text{Bayes}}(\ell, \hat{p}) = \mathbb{E}_{\ell' \sim \hat{p}}[d_T(\ell', \ell)]$. Intuitively, it measures a decision's average distance to outcomes under the distribution used. Minimum Bayes Risk (MBR) decoding outputs the decision minimizing this risk: $\delta^{\text{MBR}}(\hat{p}) = \arg\min_\ell R_{\text{Bayes}}(\ell, \hat{p})$. Intuitively, it is the weighted centroid decision under the distribution $\hat{p}$. Combining this decision rule with the calibration criterion above is exactly the procedure we propose in Task-Calibrated Decoding (TCD).

## 3 Task-Calibrated Decoding

*Task-Calibrated Decoding* (TCD) combines three steps (Figure 1): the task mapping $g_T$ collapses sampled free-form responses into a distribution $\hat{p}(\ell \mid x)$ over the discrete latent space $\mathcal{L}$, which makes calibration tractable; a calibration map $f_\phi$, fit on labeled in-domain data, corrects this distribution; and MBR decoding selects the action with minimal expected loss under the corrected distribution. The calibration map is TCD's only learned component: it is fit once per task, independently of $d_T$, and then reused to decode every new query. We first describe how the calibration map is fit, then define the decoding rule, and finally show that TCD is lightweight and, given an ideal calibration map, optimal.

Formally, let $\mathcal{D}_{\text{in-domain}} = \{(x^{(i)}, y^{(i)})\}_{i=1}^N$ denote a labeled in-domain dataset of prompts and answers for fitting the calibration map. For each input $x^{(i)}$, we sample $M$ responses from the frozen LLM and map them through $g_T$ to obtain the empirical task-level distribution $\hat{p}(\ell^{(i)} \mid x^{(i)}) \in \Delta^{C-1}$ over the discrete $C$-dimensional task-specific latent space. The corresponding reference response $y^{(i)}$ is mapped into the same outcome space $\mathcal{L}$ as $g_T(y^{(i)})$. Our calibration dataset can therefore be formalized as $\mathcal{D}_{\text{cal}} = \{\big(\hat{p}(\ell^{(i)} \mid x^{(i)}),\, g_T(y^{(i)})\big)\}_{i=1}^N$. We use $\mathcal{D}_{\text{cal}}$ to learn a calibration map $f_\phi : \Delta^{C-1} \to \Delta^{C-1}$ by minimizing the negative log-likelihood (NLL) of the reference outcomes:

$$\phi^* = \arg\min_\phi -\frac{1}{N} \sum_{i=1}^{N} \sum_{c=1}^{C} g_T(y^{(i)})_c \log\big[f_\phi(\hat{p}(\ell^{(i)} \mid x^{(i)}))\big]_c. \tag{2}$$

Rather than modifying the LLM's weights as in supervised fine-tuning (SFT), TCD corrects the LLM's task-level predictive distribution to obtain well-calibrated estimates of the output distribution. In practice, we instantiate $f_\phi$ using Dirichlet calibration (Kull et al., 2019), $f_\phi(q) = \text{softmax}(W \log(q + \epsilon) + b)$, with $\phi = \{W, b\}$ and $W \in \mathbb{R}^{C \times C}$ and $b \in \mathbb{R}^C$.

**Algorithm 1** Task-Calibrated Decoding for task $T$

**(a) Fit task-calibration map**

**for** $i \leftarrow 1$ to $N$ **do**
  $\ell^{(i)} \leftarrow g_T(y^{(i)})$
  **for** $j \leftarrow 1$ to $M$ **do**
    $\hat{y}^{(i,j)} \leftarrow \mathrm{LLM}(x^{(i)})$
    $\hat{\ell}^{(i,j)} \leftarrow g_T(\hat{y}^{(i,j)})$
  **end for**
  $\hat{p}_c^{(i)} \leftarrow \frac{1}{M}\sum_{j=1}^{M} \mathbf{1}[\hat{\ell}^{(i,j)} = c]$
**end for**
$f_{\phi^*} \leftarrow \arg\min_{\phi} \frac{1}{N}\sum_{i=1}^{N} \mathrm{CE}\Big(f_\phi(\hat{p}^{(i)}), \mathrm{onehot}(\ell^{(i)})\Big)$

**(b) Task-calibrated decoding**

**for** $j \leftarrow 1$ to $M$ **do**
  $\hat{y}^{(j)} \leftarrow \mathrm{LLM}(x)$
  $\hat{\ell}^{(j)} \leftarrow g_T(\hat{y}^{(j)})$
**end for**
$\hat{p}_c \leftarrow \frac{1}{M}\sum_{j=1}^{M} \mathbf{1}[\hat{\ell}^{(j)} = c]$
$q \leftarrow f_{\phi^*}(\hat{p})$
$\ell^\star \leftarrow \arg\min_{\ell \in \mathcal{L}} \mathbb{E}_{\ell' \sim q}\left[d_T(\ell', \ell)\right]$
**return** $\ell^\star$

At inference, for a new query $x$, we use exactly the same Monte-Carlo procedure of $M$ samples to obtain $\hat{p}(\ell \mid x)$ and apply the learned map $f_{\phi^*}$ to obtain the calibrated distribution $f_{\phi^*}(\hat{p}(\ell \mid x))$. The final step is computing a decision from the calibrated distribution using MBR decoding. As such, Task-Calibrated Decoding can be seen as a specific decoding strategy that is a composition of mapping responses to the task space, calibration, and MBR decoding. Formally, it is defined as:

$$\delta_{\mathrm{TCD}}(x) := \delta_{\mathrm{MBR}}\Big(f_{\phi^*}(\hat{p}(\ell \mid x))\Big) = \arg\min_{\ell \in \mathcal{L}} R_{\mathrm{Bayes}}\Big(\ell,\ f_{\phi^*}\Big(\frac{1}{M}\sum_{j=1}^{M} \mathbf{1}\left[g_T(y^{(j)}) = \cdot\right]\Big)\Big) \tag{3}$$

where $y^{(1)}, \ldots, y^{(M)} \overset{\text{iid}}{\sim} p(y \mid x)$ are the sampled responses. The entire procedure of task-calibrated decoding is summarized in Algorithm 1. The three parts play distinct but complementary roles. Mapping responses into the task-space *enables* calibration. The calibration map $f_{\phi^*}$ uses in-domain supervision to *correct task-level biases* in the predicted distribution, while MBR decoding converts this calibrated distribution into the *optimal decision according to the task loss* $d_T$. Importantly, our procedure retains the full task-level output distribution rather than directly learning to output a single decision. Consequently, the learned calibration map can, in general, be combined with any arbitrary task-specific loss beyond always selecting the mode of the calibrated output distribution $f_{\phi^*}(\hat{p}(\ell \mid x))$.

**Example.** Consider applying TCD at inference for an LLM-as-a-judge task with five possible ratings, $\mathcal{L} \cong \{1, \ldots, 5\}$, under the $L_1$ loss Figure 1. For a new query $x$, we sample $M$ free-form judgments from the LLM (e.g., "Very harmful, so five", "5 is what I would say", "I think it should be 4"). Then, we map each through $g_T$ into a one-hot latent rating using an auxiliary language model, and aggregate them into the empirical distribution $\hat{p}(\ell \mid x) = (0.0, 0.0, 0.1, 0.3, 0.6)^T$ – the LLM's raw beliefs, which cluster around ratings 4 and 5. Subsequently, we applying the learned calibration map, $f_{\phi^*}(\hat{p}(\ell \mid x)) = (0.0, 0.0, 0.15, 0.40, 0.45)$. In this example, it corrects the LLM's tendency toward overconfidence in high ratings. Finally, MBR decoding under the $L_1$ loss selects the median of this calibrated distribution – here the rating 4 – rather than the mode of the raw, uncalibrated distribution which would be the rating 5.

**Task-Calibrated Decoding is Lightweight.** The calibration map $f_{\phi^*}$ has only $C^2 + C$ parameters, scaling quadratically with the number of classes, making it an extremely lightweight approach compared to fine-tuning the LLM's weights. In the previous example task of five possible outcomes, the Dirichlet calibration map $f$ has only 30 parameters. Moreover, for many task-specific losses $d_T$, the Minimum Bayes Risk (MBR) action can be computed in closed-form, making MBR decoding a computationally inexpensive ingredient in our algorithm: TCD typically requires no optimization over the task latent space $\mathcal{L}$. Most importantly, for TCD, the underlying LLM is only accessed as a black box we only sample free-form responses, making the method applicable to all types of models.

**Task-Calibrated Decoding is Optimal.** Our procedure of Task-Calibrated Decoding is motivated by the following optimality result from decision theory that states that the best decision rule given the LLM's task-specific output distribution $\hat{p}(\ell \mid x)$ is precisely MBR decoding on a fully distributional calibrated version of $\hat{p}(\ell \mid x)$. Consider the theoretically ideal conditional calibration map

$$f^*_{g_T}(q) := \mathbb{E}_{x,y}\left[g_T(y) \mid \hat{p}(\ell \mid x) = q\right]. \tag{4}$$

This calibration map assigns all distributions for which the uncalibrated LLM output distribution is $q$ to the true expected distribution of task-specific latent labels. While this map is a purely

theoretical concept and intractable to compute in practice as it requires marginalization over the true data distribution $p^*(\mathrm{x}, \mathrm{y})$, we can approximate it by optimizing the negative log-likelihood of the calibration dataset, for example using Dirichlet calibration in our method (Algorithm 1).

**Proposition 3.1** (Optimality of Task-Calibrated Decoding, informal)**.** *For any task-specific loss function $d_T$, the decision rule $\delta_{\mathrm{MBR}}(f^*_{g_T}(\hat{p}(\ell \mid x)))$ minimizes expected loss under $p^*(x, y)$ among all decision rules whose only input is the original latent prediction $\hat{p}(\ell \mid x)$.*

The full proof is provided in Appendix F. Intuitively, consider all input queries $x$ for which the LLM produces some task-specific latent distribution $q$. Conditional on these inputs, the *true* reference outcome is distributed according to $f^*_{g_T}(q)$ by definition. Therefore, the best decision in terms of expected loss $d_T$ that can be made based only on this distribution $q$ is the Bayes-optimal action under the conditional distribution $f^*_{g_T}(q)$. This is precisely MBR decoding on $f^*_{g_T}(q)$ (Noarov and Roth, 2024). Importantly, this result holds for *any* $d_T$ and, as such, the loss $d_T$ does not have to be fixed for fitting the calibration map. Different users could encode different preferences with $d_T$ but still use the same fitted calibration map. In practice, we replace $f^*_{g_T}$ with our approximate Dirichlet calibration map $f_{\phi^*}$, estimated from finite in-domain data. In the population limit, the objective in Equation (2) learns to approximate $f^*_{g_T}$ within the restricted function class of linear maps, see Proposition F.1. While theoretical optimality only holds for the target ground-truth calibration map $f^*_{g_T}$ and not $f_{\phi^*}$, the results in Section 4 show that Task-Calibrated Decoding that relies on an approximate calibration map consistently improves empirical decision quality compared to other methods using $\hat{p}(\ell \mid x)$.

## 4 Experiments

We evaluate Task-Calibrated Decoding by comparing it with other lightweight black-box methods that use in-domain data to improve LLM outputs on downstream tasks, and demonstrate that it consistently improves performance. We also compare our algorithm with Self-Supervised Fine-Tuning (SFT) and find that it sometimes yields even better results. Lastly, we qualitatively study the intuitive effect of the learned calibration map and show that in several tasks it can be understood as correcting specific, interpretable biases in the LLM's predictive distribution using labeled in-domain data.

**Tasks and Datasets.** We consider different downstream applications of LLMs that come with latent structures $\mathcal{L}$ and task-specific losses $d_T$ (see Table 4). (i) **Ordinal Regression**: We evaluate numerical LLM-as-a-judge tasks as ordinal prediction problems. Responses represent elements from a discrete set of reals $\mathcal{L} \cong \{0, 1, \ldots, C\}$. We evaluate our method on Helpsteer (criterion "correctness") (Wang et al., 2023; Dong et al., 2023) and STSB (Cer et al., 2017), using the $L_1$-loss as $d_T$. (ii) **Classification**: We also consider single-label classification over a discrete set of $C$ answers with the Exact-Match loss (i.e. accuracy). Here, we use When2Call (Ross et al., 2025), where the LLM classifies if it should call a tool, request more information, or abstain from predicting. We also evaluate on the multiple-choice QA benchmark MMLU (Hendrycks et al., 2021). (iii) **Answer-Or-Abstain**: Furthermore, we consider "Answer-Or-Abstain" tasks on TriviaQA (Joshi et al., 2017) and SimpleQA-Verified (Haas et al., 2026). Here, the latent space is binary $\mathcal{L} \cong \{A, \perp\}$, and we use the asymmetric BAS loss of Wu et al. (2026) which depends a user-specified risk tolerance $t$ (see Appendix C.1). (iv) **Open Set-Prediction**: Moreover, we evaluate Multi-Answer QA using MAQA (Yang et al., 2025). Here, the latent structure is the space of binary vectors $\mathcal{L} \cong \{0, 1\}^{\leq C}$. Each entry indicates if a candidate answer should be included in the response. The support set of possible answers varies across questions, so we must decompose this task and the associated Hamming loss into binary decisions of whether to include an answer. Consequently, we task calibrate the LLM's Bernoulli beliefs over if answers should be included in the response. (v) **Vector-Valued Ordinal Regression**: Lastly, we study vector-valued predictions in $\mathcal{L} \cong \{0, \ldots, C\}^2$ and again use the Helpsteer dataset (Wang et al., 2023; Dong et al., 2023). Now, the LLM predicts two criteria of a text simultaneously ("correctness", "helpfulness") and the loss is $L_1$ distance.

**Models.** We evaluate the instruction-tuned models Gemma 3 4B, Gemma 3 12B (Gemma Team, 2025), Qwen 3 4B, Qwen 3 30B A3B (Qwen Team, 2025) and GPT-5-mini (Team, 2026). All models are run with default decoding parameters given by the providers. We sample $M = 20$ times per prompt (Appendix G) to approximate the push-forward distribution $\hat{p}(\mathrm{x})$. We train and evaluate calibrators using 5-fold cross-validation, and report the mean performance and standard deviations.

Table 1: Generation quality (best, second-best) of Task Calibrated Decoding (TCD) versus other strategies, including MBR on the uncalibrated distribution $q$. TCD consistently improves decoding.

| Latent Structure $\mathcal{L}$ (Metric) | | $\{0,\dots,C\}$ (L1 ↓) | | $[C]$ (Acc. ↑) | | $\{A,\perp\}$ (BAS ↑) | | $\{0,1\}^{\leq C}$ (Hamming ↓) | $\{0,\dots,C\}^2$ (L1 ↓) |
|---|---|---|---|---|---|---|---|---|---|
| | | Helpsteer | STSB | MMLU | When2Call | SimpleQA | TriviaQA | MAQA | Helpsteer |
| Gemma-3-4B | MBR | $1.04_{\pm 0.09}$ | $1.11_{\pm 0.04}$ | $26.70_{\pm 1.34}$ | $50.56_{\pm 1.35}$ | $-0.20_{\pm 0.02}$ | $0.18_{\pm 0.05}$ | $0.74_{\pm 0.04}$ | $2.03_{\pm 0.15}$ |
| | Temperature | $1.02_{\pm 0.08}$ | $1.11_{\pm 0.03}$ | $26.70_{\pm 1.34}$ | $50.56_{\pm 1.35}$ | $-0.25_{\pm 0.02}$ | $0.17_{\pm 0.05}$ | $0.74_{\pm 0.04}$ | $1.97_{\pm 0.13}$ |
| | Decision Calibration | $\mathbf{0.83}_{\pm 0.03}$ | $0.85_{\pm 0.06}$ | $\mathbf{35.64}_{\pm 1.19}$ | $\mathbf{55.68}_{\pm 3.03}$ | $\mathbf{-0.00}_{\pm 0.00}$ | $\mathbf{0.24}_{\pm 0.05}$ | $\mathbf{0.66}_{\pm 0.04}$ | $\mathbf{1.24}_{\pm 0.03}$ |
| | Linear Alignment | $1.01_{\pm 0.09}$ | $0.93_{\pm 0.06}$ | $27.82_{\pm 1.39}$ | $50.05_{\pm 0.43}$ | $\mathbf{-0.00}_{\pm 0.00}$ | $-0.00_{\pm 0.00}$ | $0.74_{\pm 0.04}$ | $1.41_{\pm 0.03}$ |
| | Classifier | $0.96_{\pm 0.05}$ | $\mathbf{0.84}_{\pm 0.03}$ | $\mathbf{37.77}_{\pm 1.04}$ | $\mathbf{59.40}_{\pm 2.35}$ | $\mathbf{-0.00}_{\pm 0.00}$ | $\mathbf{0.23}_{\pm 0.03}$ | $\mathbf{0.62}_{\pm 0.03}$ | $1.40_{\pm 0.09}$ |
| | ICL | $1.10_{\pm 0.10}$ | $1.13_{\pm 0.05}$ | $25.70_{\pm 1.54}$ | $41.10_{\pm 4.14}$ | $\mathbf{-0.12}_{\pm 0.02}$ | $0.23_{\pm 0.03}$ | $0.80_{\pm 0.08}$ | $2.21_{\pm 0.08}$ |
| | **TCD (ours)** | $\mathbf{0.84}_{\pm 0.03}$ | $\mathbf{0.83}_{\pm 0.02}$ | $\mathbf{37.77}_{\pm 1.04}$ | $\mathbf{59.40}_{\pm 2.35}$ | $\mathbf{-0.00}_{\pm 0.00}$ | $0.22_{\pm 0.05}$ | $\mathbf{0.62}_{\pm 0.03}$ | $\mathbf{1.28}_{\pm 0.07}$ |
| Gemma-3-12B | MBR | $0.94_{\pm 0.05}$ | $1.01_{\pm 0.03}$ | $47.15_{\pm 2.91}$ | $68.90_{\pm 2.76}$ | $-0.19_{\pm 0.03}$ | $0.44_{\pm 0.03}$ | $0.60_{\pm 0.07}$ | $1.89_{\pm 0.06}$ |
| | Temperature | $0.91_{\pm 0.06}$ | $1.00_{\pm 0.03}$ | $47.15_{\pm 2.91}$ | $68.90_{\pm 2.76}$ | $-0.25_{\pm 0.03}$ | $0.44_{\pm 0.03}$ | $0.60_{\pm 0.07}$ | $1.79_{\pm 0.06}$ |
| | Decision Calibration | $\mathbf{0.84}_{\pm 0.03}$ | $0.77_{\pm 0.06}$ | $53.07_{\pm 2.50}$ | $\mathbf{71.80}_{\pm 1.96}$ | $\mathbf{-0.00}_{\pm 0.00}$ | $\mathbf{0.47}_{\pm 0.03}$ | $\mathbf{0.56}_{\pm 0.06}$ | $\mathbf{1.24}_{\pm 0.04}$ |
| | Linear Alignment | $0.98_{\pm 0.08}$ | $0.85_{\pm 0.03}$ | $47.15_{\pm 2.91}$ | $68.90_{\pm 2.76}$ | $\mathbf{-0.00}_{\pm 0.00}$ | $0.46_{\pm 0.03}$ | $0.60_{\pm 0.07}$ | $1.46_{\pm 0.07}$ |
| | Classifier | $0.94_{\pm 0.09}$ | $\mathbf{0.67}_{\pm 0.03}$ | $\mathbf{57.88}_{\pm 4.35}$ | $\mathbf{72.70}_{\pm 2.99}$ | $\mathbf{-0.00}_{\pm 0.00}$ | $0.47_{\pm 0.02}$ | $\mathbf{0.51}_{\pm 0.04}$ | $1.33_{\pm 0.07}$ |
| | ICL | $0.85_{\pm 0.04}$ | $0.84_{\pm 0.04}$ | $\mathbf{56.09}_{\pm 3.05}$ | $56.00_{\pm 1.64}$ | $\mathbf{-0.10}_{\pm 0.02}$ | $0.46_{\pm 0.04}$ | $0.61_{\pm 0.03}$ | $1.90_{\pm 0.07}$ |
| | **TCD (ours)** | $\mathbf{0.80}_{\pm 0.04}$ | $\mathbf{0.64}_{\pm 0.05}$ | $\mathbf{57.88}_{\pm 4.35}$ | $\mathbf{72.70}_{\pm 2.99}$ | $\mathbf{-0.00}_{\pm 0.00}$ | $\mathbf{0.47}_{\pm 0.03}$ | $\mathbf{0.51}_{\pm 0.04}$ | $\mathbf{1.25}_{\pm 0.08}$ |
| Qwen-3-4B | MBR | $0.83_{\pm 0.05}$ | $1.33_{\pm 0.07}$ | $49.72_{\pm 3.82}$ | $\mathbf{73.00}_{\pm 1.70}$ | $-0.18_{\pm 0.01}$ | $0.16_{\pm 0.04}$ | $0.87_{\pm 0.08}$ | $2.05_{\pm 0.05}$ |
| | Temperature | $0.81_{\pm 0.06}$ | $1.33_{\pm 0.07}$ | $49.72_{\pm 3.82}$ | $\mathbf{73.00}_{\pm 1.70}$ | $-0.22_{\pm 0.01}$ | $0.15_{\pm 0.04}$ | $0.87_{\pm 0.08}$ | $1.96_{\pm 0.07}$ |
| | Decision Calibration | $\mathbf{0.81}_{\pm 0.07}$ | $0.72_{\pm 0.05}$ | $53.85_{\pm 4.25}$ | $71.80_{\pm 3.85}$ | $\mathbf{-0.00}_{\pm 0.00}$ | $\mathbf{0.21}_{\pm 0.03}$ | $\mathbf{0.75}_{\pm 0.07}$ | $\mathbf{1.25}_{\pm 0.06}$ |
| | Linear Alignment | $0.85_{\pm 0.07}$ | $0.73_{\pm 0.04}$ | $49.72_{\pm 3.82}$ | $\mathbf{73.00}_{\pm 1.70}$ | $\mathbf{-0.00}_{\pm 0.00}$ | $-0.00_{\pm 0.00}$ | $0.87_{\pm 0.08}$ | $1.50_{\pm 0.07}$ |
| | Classifier | $0.86_{\pm 0.07}$ | $\mathbf{0.70}_{\pm 0.05}$ | $\mathbf{57.65}_{\pm 4.38}$ | $\mathbf{74.10}_{\pm 2.75}$ | $\mathbf{-0.00}_{\pm 0.00}$ | $\mathbf{0.21}_{\pm 0.03}$ | $\mathbf{0.72}_{\pm 0.05}$ | $1.40_{\pm 0.06}$ |
| | ICL | $0.93_{\pm 0.05}$ | $0.79_{\pm 0.06}$ | $\mathbf{56.50}_{\pm 1.67}$ | $59.56_{\pm 2.94}$ | $\mathbf{-0.10}_{\pm 0.01}$ | $0.17_{\pm 0.01}$ | $1.07_{\pm 0.13}$ | $1.85_{\pm 0.05}$ |
| | **TCD (ours)** | $\mathbf{0.79}_{\pm 0.05}$ | $\mathbf{0.67}_{\pm 0.03}$ | $\mathbf{57.65}_{\pm 4.38}$ | $\mathbf{74.10}_{\pm 2.75}$ | $\mathbf{-0.00}_{\pm 0.00}$ | $0.21_{\pm 0.03}$ | $\mathbf{0.72}_{\pm 0.05}$ | $\mathbf{1.29}_{\pm 0.03}$ |
| Qwen3-30B-A3B | MBR | $0.79_{\pm 0.04}$ | $1.18_{\pm 0.04}$ | $60.78_{\pm 2.22}$ | $74.25_{\pm 3.57}$ | $-0.07_{\pm 0.02}$ | $0.46_{\pm 0.05}$ | $0.54_{\pm 0.03}$ | $2.11_{\pm 0.08}$ |
| | Temperature | $\mathbf{0.79}_{\pm 0.03}$ | $1.18_{\pm 0.03}$ | $60.78_{\pm 2.22}$ | $74.25_{\pm 3.57}$ | $-0.10_{\pm 0.03}$ | $0.46_{\pm 0.05}$ | $0.54_{\pm 0.03}$ | $2.05_{\pm 0.09}$ |
| | Decision Calibration | $0.79_{\pm 0.03}$ | $0.72_{\pm 0.05}$ | $65.36_{\pm 1.06}$ | $\mathbf{74.85}_{\pm 2.61}$ | $\mathbf{0.03}_{\pm 0.02}$ | $0.47_{\pm 0.05}$ | $\mathbf{0.51}_{\pm 0.03}$ | $\mathbf{1.22}_{\pm 0.04}$ |
| | Linear Alignment | $0.80_{\pm 0.03}$ | $0.71_{\pm 0.05}$ | $60.78_{\pm 2.22}$ | $74.25_{\pm 3.57}$ | $\mathbf{-0.00}_{\pm 0.00}$ | $0.47_{\pm 0.05}$ | $0.54_{\pm 0.03}$ | $1.42_{\pm 0.06}$ |
| | Classifier | $0.81_{\pm 0.04}$ | $\mathbf{0.68}_{\pm 0.04}$ | $\mathbf{70.61}_{\pm 2.02}$ | $\mathbf{78.56}_{\pm 1.66}$ | $\mathbf{-0.00}_{\pm 0.00}$ | $\mathbf{0.47}_{\pm 0.04}$ | $\mathbf{0.49}_{\pm 0.03}$ | $1.45_{\pm 0.04}$ |
| | ICL | $0.80_{\pm 0.04}$ | $0.93_{\pm 0.06}$ | $\mathbf{69.65}_{\pm 3.56}$ | $64.00_{\pm 1.30}$ | $-0.00_{\pm 0.02}$ | $\mathbf{0.48}_{\pm 0.05}$ | $0.65_{\pm 0.04}$ | $1.97_{\pm 0.05}$ |
| | **TCD (ours)** | $\mathbf{0.74}_{\pm 0.04}$ | $\mathbf{0.68}_{\pm 0.05}$ | $\mathbf{70.61}_{\pm 2.02}$ | $\mathbf{78.56}_{\pm 1.66}$ | $\mathbf{0.03}_{\pm 0.02}$ | $0.46_{\pm 0.04}$ | $\mathbf{0.49}_{\pm 0.03}$ | $\mathbf{1.24}_{\pm 0.03}$ |
| GPT-5 mini | MBR | $0.83_{\pm 0.04}$ | $1.11_{\pm 0.05}$ | $76.31_{\pm 2.28}$ | $57.90_{\pm 3.12}$ | $-0.02_{\pm 0.03}$ | $\mathbf{0.65}_{\pm 0.04}$ | $0.54_{\pm 0.08}$ | $1.90_{\pm 0.11}$ |
| | Temperature | $0.83_{\pm 0.04}$ | $1.11_{\pm 0.04}$ | $76.31_{\pm 2.28}$ | $57.90_{\pm 3.12}$ | $-0.14_{\pm 0.04}$ | $0.65_{\pm 0.04}$ | $0.54_{\pm 0.08}$ | $1.83_{\pm 0.13}$ |
| | Decision Calibration | $\mathbf{0.75}_{\pm 0.02}$ | $0.62_{\pm 0.02}$ | $\mathbf{79.11}_{\pm 1.68}$ | $65.20_{\pm 3.25}$ | $\mathbf{0.03}_{\pm 0.02}$ | $0.65_{\pm 0.04}$ | $\mathbf{0.46}_{\pm 0.06}$ | $\mathbf{1.24}_{\pm 0.08}$ |
| | Linear Alignment | $0.77_{\pm 0.07}$ | $\mathbf{0.62}_{\pm 0.02}$ | $76.31_{\pm 2.28}$ | $57.90_{\pm 3.12}$ | $-0.00_{\pm 0.00}$ | $0.64_{\pm 0.03}$ | $0.54_{\pm 0.08}$ | $1.34_{\pm 0.12}$ |
| | Classifier | $0.76_{\pm 0.04}$ | $0.63_{\pm 0.03}$ | $\mathbf{80.56}_{\pm 2.53}$ | $\mathbf{67.30}_{\pm 4.27}$ | $0.00_{\pm 0.00}$ | $0.63_{\pm 0.04}$ | $\mathbf{0.43}_{\pm 0.02}$ | $1.28_{\pm 0.09}$ |
| | ICL | $0.81_{\pm 0.02}$ | $0.95_{\pm 0.07}$ | $76.20_{\pm 3.58}$ | $\mathbf{68.20}_{\pm 1.94}$ | $-0.16_{\pm 0.05}$ | $0.59_{\pm 0.03}$ | $0.65_{\pm 0.08}$ | $1.70_{\pm 0.11}$ |
| | **TCD (ours)** | $\mathbf{0.73}_{\pm 0.04}$ | $\mathbf{0.59}_{\pm 0.03}$ | $\mathbf{80.56}_{\pm 2.53}$ | $\mathbf{67.30}_{\pm 4.27}$ | $\mathbf{0.03}_{\pm 0.02}$ | $\mathbf{0.65}_{\pm 0.04}$ | $\mathbf{0.43}_{\pm 0.02}$ | $\mathbf{1.16}_{\pm 0.08}$ |

**Baselines.** As a simple baseline, we compare against Minimum Bayes Risk decoding under the uncalibrated empirical distribution (**MBR**) (Tomov et al., 2026). The remaining baselines, by contrast, also leverage the in-domain data. **Temperature Scaling** fits a positive scalar temperature $\tau$ by minimizing NLL on the calibration training split (Guo et al., 2017). We apply temperature scaling to the log empirical probabilities and subsequently perform MBR decoding. We also compare with **Decision Calibration**, implementing the iterative recalibration procedure of Zhao et al. (2021) for the relevant decision rule. Our **Classifier** baseline treats the NLL-fitted calibration map $f_\phi$ as a probabilistic classifier over latent actions. For a finite support $\{\ell_1,\dots,\ell_C\}$, it predicts $\pi_\phi(\hat{p}(\ell \mid \mathrm{x})) = \ell_{\arg\max_{c\in\{1,\dots,C\}}[f_\phi(\hat{p}(\ell|\mathrm{x}))]_c}$. Thus, Classifier and Task-Calibrated Decoding share the same fitted map, differing only in their decision rule: The former selects the maximum-probability latent action, whereas Task-Calibrated Decoding minimizes the expected task loss. The baseline can be interpreted as directly learning to predict the decision from the uncalibrated distribution instead of relying on MBR decoding. We further compare against **ICL** (Brown et al., 2020) which prepends three question–answer demonstrations, retrieved by cosine similarity between question embeddings. Lastly, we compare to **Linear Alignment** (Van Den Burg et al., 2025), which learns a linear map from black-box LLM judgments directly to human labels using the same calibration data (Appendix B).

### 4.1 Calibration Improves Decoding

Table 1 demonstrates that Task-Calibrated Decoding is the best method in the large majority of cases and among the top-2 in all but three cases. As expected by the optimality statement of Proposition 3.1, it enhances performance relative to uncalibrated MBR across different models and latent-structure choices. On the two classification tasks (MMLU, When2Call), the Classifier baseline exactly matches TCD: under 0-1 loss, the Bayes-optimal decision is the mode, which both methods compute. On every other task, TCD outperforms the Classifier baseline, which shows that TCD's advantage does not only come from having access to calibration labels to fit a good map $f_\phi$ as the classifier baseline, too, uses those labels directly. Instead, combining the calibration map with the task-specific MBR decision rule drives the performance as it explicitly minimizes $d_T$ rather than reducing every task to a simple classification problem. Also, directly learning the decision from the in-domain data, as in the Linear

Table 2: Generation quality (best) of TCD against Self Supervised Finetuning (SFT) with Low-Rank Adapaters (LoRA). TCD performs comparably while being more lightweight and not requiring access to the LLM weights. Results are on LoRA test fold.

| **Latent Structure** $\mathcal{L}$ (Metric) | | $\{0,\dots,C\}$ (L1 ↓) Helpsteer | STSB | $[C]$ (Acc. ↑) MMLU | When2Call | $\{A,\perp\}$ (BAS ↑) SimpleQA | TriviaQA | $\{0,1\}^{\leq C}$ (Hamming ↓) MAQA | $\{0,\dots,C\}^2$ (L1 ↓) Helpsteer |
|---|---|---|---|---|---|---|---|---|---|
| Gemma-3-4B | LoRA | **0.69** | **0.56** | **62.57** | **76.88** | −0.02 | 0.18 | 0.94 | 1.42 |
| | **TCD (ours)** | 0.82 | 0.85 | 38.55 | 63.82 | **-0.00** | **0.21** | **0.64** | **1.23** |
| Gemma-3-12B | LoRA | **0.65** | **0.49** | **82.68** | **79.00** | −0.01 | 0.41 | 0.64 | 1.36 |
| | **TCD (ours)** | 0.75 | 0.68 | 62.01 | 74.00 | **-0.00** | **0.46** | **0.54** | **1.15** |
| Qwen-3-4B | LoRA | **0.68** | **0.51** | **68.16** | **81.00** | −0.06 | 0.17 | 1.02 | **1.20** |
| | **TCD (ours)** | 0.74 | 0.73 | 62.01 | 78.00 | **-0.00** | **0.18** | **0.74** | 1.25 |
| Qwen3-30B-A3B | LoRA | 0.74 | **0.51** | **78.21** | 77.89 | **0.09** | **0.47** | 0.69 | **1.15** |
| | **TCD (ours)** | **0.74** | 0.70 | 71.51 | **80.90** | 0.05 | 0.44 | **0.50** | 1.21 |

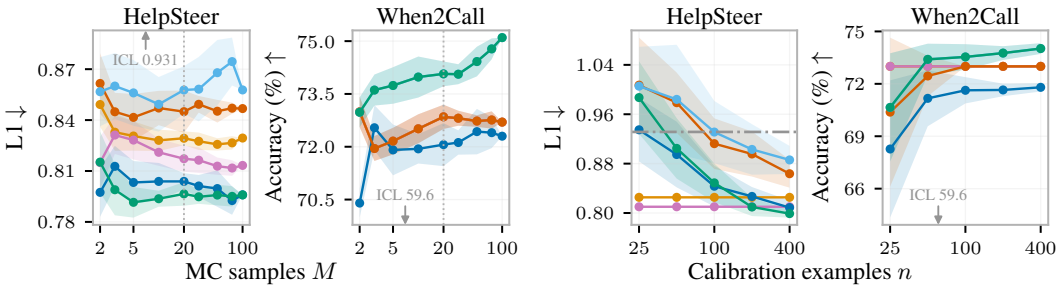
Alignment, does not outperform TCD, similarly to ICL and the temperature scaling. Lastly, on most tasks, TCD also outperforms Decision Calibration, again consistent with the optimality statement of Proposition 3.1; the exception is the vector-valued task, where the higher-dimensional action space makes distributional calibration harder to estimate, and therefore favors the more tractable Decision Calibration, which targets the decision boundaries directly rather than the full distribution.

**Comparison to Finetuning.** We also compare against Supervised Fine-Tuning (SFT) with Low-Rank Adapaters (LoRA) (Hu et al., 2022) which requires weight access (Table 2, details in Appendix B). The results reflect each methods capabilities: TCD processes the query $x$ through $\hat{p}(\ell \mid x)$ and corrects the belief–outcome distortions shared across the domain. LoRA operates directly on the query $x$ and can change these beliefs themselves. Therefore, LoRA naturally has an edge when the data teaches a shared, input-dependent rule the model itself fails to apply—rating conventions (Helpsteer, STSB) or a tool-use policy (When2Call). Conversely, when correctness hinges on question-specific knowledge – as in TriviaQA or MAQA – SFT has no explicit rule to learn, while the model's confidence–correctness relationship remains shared between different queries. TCD can directly target this relationship with calibration. As TCD only requires sampling from the LLM, it can also complement SFT: TCD can be applied on top of a fine-tuned model. Importantly, though, it can also be used for black-box models for which SFT is not applicable.

**Ablations.** TCD is sample-efficient, though the right sample budget depends on the task: performance saturates at $M = 20$ Monte Carlo samples on some datasets, while it continues to improve beyond $M = 20$ for others (Figure 2a). In practice, this allows a trade-off between inference compute and decision performance. Similarly, TCD is data-efficient: Already small calibration set sizes suffice to outperform uncalibrated MBR Figure 2b, but, naturally, more calibration samples again directly positively correlate with performance, enabling flexibility regarding the trade-off between data budget and downstream performance. Importantly, across the full range of sampling and dataset-size regimes, TCD outperforms the other baselines at exploiting labeled in-domain data (full results in Appendix A.1). Finally, we also observe that a calibration map trained on one prompt template also yields strong TCD performance across other prompt templates Table 3. This indicates that TCD is robust to variations in prompt phrasing and that the learned calibration maps transfer well when the prompt templates in a downstream task are adjusted.

### 4.2 What the Calibration Maps Learn

We now study how the calibration map $f_{\phi^*}(\hat{p}(\ell \mid \mathrm{x}))$ transforms the original latent probabilities $\hat{p}(\ell \mid \mathrm{x})$ in detail. For binary action spaces like the Answer-or-Abstain latent structure $\mathcal{L} \cong \{A, \perp\}$, the map can be fully described by how it maps the probability of one of the actions, for example $A$ ("answer"). In Figure 3c, we observe that for this action space, the learned map $f_{\phi^*}$ consistently lowers the predicted probability of the action $A$ ("answer") to mitigate overconfidence in the uncalibrated LLM's belief distribution. Effectively, the recalibration map will only output this action A if the uncalibrated distribution assigns a significantly higher probability ($\hat{p}(\ell \mid \mathrm{x})_A > 0.7$) than the threshold in the BAS-metric ($t = 0.25$). Intuitively, the bias of being overconfident in predicting the response "answer" is effectively mitigated by the learned recalibration map $f_{\phi^*}$.




(2a) Ablations on MC sample size $M$

(2b) Ablations on Calibration Set Size $n$

Figure 2: **TCD is sample- and data-efficient.** We vary (a) number of Monte Carlo samples $M$ used to estimate $\hat{p}(\ell \mid x)$, and (b) size $n$ of the calibration set for $f_\phi$, for Qwen3-4B on HelpSteer & When2Call. Across both ablations, TCD remains the best-performing method. Methods: MBR, Temperature, Decision Calibration, Linear Alignment, Classifier, ICL, **TCD (ours)**

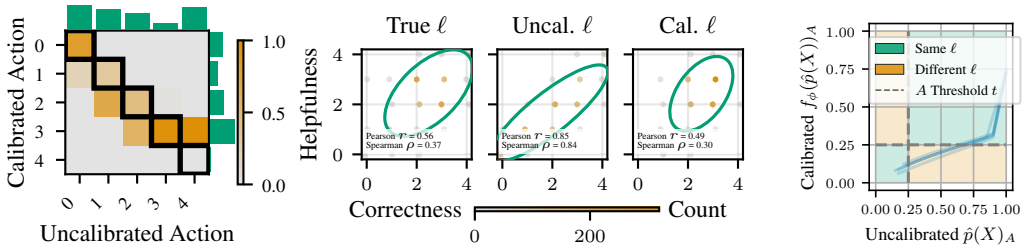
For ordinal regression (Helpsteer) ( $\mathcal{L} \cong \{0, \ldots, 4\}$), i.e. rating the correctness of a model response, we study how MBR responses change between the uncalibrated and calibrated beliefs in Figure 3a. We find a pattern akin to an "S-Curve": for low ratings (0–2), calibration shifts MBR responses toward higher scores. In contrast, high ratings (3–4) are either left unchanged or shifted downward. Interestingly, the learned calibration map effectively no longer yields the latent response $\ell \mathrel{\hat{=}} 4$. In our experiments, predicting $\ell \mathrel{\hat{=}} 3$ whenever the model's uncalibrated belief yields an MBR action $\ell \mathrel{\hat{=}} 4$ has better performance on average. Also, no other uncalibrated MBR action is shifted to a calibrated MBR response of $\ell \mathrel{\hat{=}} 4$. This implies that there is no signal in the LLM's uncalibrated beliefs $\hat{p}(\mathrm{x})$ from which it is optimal to predict $\ell = 4$. This is not a flaw of TCD, but it reveals a severe shortcoming in the underlying LLM: It is unable to reliably predict $\ell \mathrel{\hat{=}} 4$. Hence, the optimal strategy in terms of the expected loss is to avoid predicting $\ell \mathrel{\hat{=}} 4$ altogether.

Lastly, we study how task calibration can help to mitigate systematic biases (Gallegos et al., 2024; Shi et al., 2025) in an LLM's output distribution. In particular, we consider the multi-criterion ordinal regression task on Helpsteer, where the LLM rates a model's response on both helpfulness and correctness simultaneously. In Figure 3b, we find that the uncalibrated $\hat{p}(\ell \mid \mathrm{x})$ heavily correlates ratings for the two criteria, meaning the LLMs is very likely to assign identical values to both. However, in the true data, the correlation is significantly weaker. The task calibration map $f_{\phi^*}(\hat{p}(\ell \mid \mathrm{x}))$ more accurately recovers the true correlation structure and therefore improves the LLM's reliability by correcting this correlation bias using the calibration data.

## 5 Related Work

**Lightweight and Post-Hoc Adaptation of LLMs.** When updating a target LLM is computationally expensive or impossible, prior work adapts frozen models through in-context learning (ICL) (**?**), black-box prompt optimization (Sun et al., 2022), or smaller auxiliary models that modify the target model's token-level predictions at decoding time (Ormazabal et al., 2023; Liu et al., 2024). Another line of work improves ICL through post-hoc calibration. Fei et al. (2023) estimates bias using random in-domain words and corrects class predictions, while LinC Fits a linear probe on ICL output probabilities for correction (Abbas et al., 2024). Most closely related, Van Den Burg et al. (2025) learns a linear mapping from black-box LLM judgments to human labels using a small calibration set. Our approach instead operates on the entire output *distribution* induced by repeated free-form generations.

**Calibration in LLMs.** Recent work has shown that LLMs are often miscalibrated (Kadavath et al., 2022; Plaut et al., 2025). Consequently, methods emerged that recalibrate token probabilities or verbalized confidence scores (Xie et al., 2024; Li et al., 2025b). Some recent work also moves beyond token-level probabilities: Nakkiran et al. (2025) shows that language models can exhibit confidence calibration in semantic answer spaces. Shen et al. (2024) optimizes a specific temperature parameter for an unseen task in which only the prompts are available, but crucially not the targets, by leveraging data from multiple other tasks. Only a few works connect calibration to decision-making in LLMs: Wu et al. (2026) study Answer-versus-Abstain tasks under asymmetric costs, while Band et al. (2024) improves calibration in long-form tasks. Similar to TCD, Estienne et al. (2026) calibrates the models'




(3a) ■ Absolute Counts & ■ $\mathbb{P}(\text{Cal}|\text{Uncal})$. Change from uncalibrated to calibrated action for Helpsteer. Low rating are up-shifted while higher rating remain unchanged or are decreased.

(3b) Answer correlations for vector-valued Helpsteer predictions ("helpfulness" and "correctness") for . Ellipses indicate the $2\sigma$ covariance region. The uncalibrated model shows heavy correlation between the criteria, whereas task calibration recovers the weaker correlation observed in the ground truth.

(3c) Learned calibration map $f_\phi$ for $\mathcal{L} = \{A, \perp\}$ for the probability of action $\ell = A$. While the uncalibrated LLM predicts $\ell = A$("Answer") if $\hat{p}(\mathrm{x})_A \geq t$, the calibrated one is more conservative.

distribution over predefined class strings derived from token probabilities and assesses its quality using proper scoring rules, which are rooted in decision-making. In contrast, we not only calibrate but also use the calibrated distribution to compute better task-specific answers in a purely black-box way.

**Sampling Based Decoding** Our work also builds on methods that improve predictions by aggregating multiple sampled responses. Self-consistency selects the majority answer among sampled reasoning paths (Wang et al., 2022), while Minimum Bayes Risk (MBR) decoding selects the output that minimizes expected loss under a predictive distribution. MBR decoding has its roots in machine translation (Eikema and Aziz, 2022; Müller and Sennrich, 2021) and has more recently been applied to LLM regression (Lukasik et al., 2024), extended to incorporate uncertainty over model parameters (Daheim et al., 2025), and generalized to task-dependent latent outcome spaces (Tomov et al., 2026).

## 6 Discussion

**Limitations.** The optimality guarantee (Proposition 3.1) is only for the idealized calibration map and holds for decoding strategies that are based solely on the LLM's latent distribution $\hat{p}(\ell \mid \mathrm{x})$. If additional information is available, such as internal representations or alternative queries, there may, in theory, be decoding strategies with better performance than MBR. Nonetheless, we empirically show that our framework outperforms existing strategies beyond our formal framework in practice. Furthermore, TCD requires a discrete latent support that is shared across samples. While this is natural for many tasks (e.g., ratings or fixed candidate sets), this requirement may not be applicable to more open-ended settings. However, our work is useful toward this direction as well, as open-ended tasks can often be meaningfully reduced to a simpler discrete structure. For example, we demonstrate that *Answer-versus-Abstain* settings can be effectively task calibrated. Further, we show how more complex problems, such as set prediction over open-ended support, can be decomposed into simpler decision-making problems that are again tractable to task calibrate. Lastly, our framework leverages task-induced semantics. Therefore, task calibration improves reliability and decoding performance for a given problem, but not for all tasks and applications simultaneously with a single recalibration map. Our method should be seen as a widely applicable general-purpose tool to enhance LLMs when they are deployed in a given downstream application that admits a task-dependent latent structure.

**Conclusion and Outlook** Our results highlight the potential of the Task-Calibrated Decoding to systematically improve downstream performance in LLMs. Requiring only sampled responses and a calibration map with few parameters, TCD consistently outperforms other lightweight methods that use the same in-domain data. It complements rather than replaces finetuning, since it can be applied on top of any adapted model. The TCD framework is generic and applies to any setting that admits a discrete latent representation with shared support across outputs, or that can be decomposed into such a structure. Hence, one particularly promising direction is LLM-as-a-judge systems (Gu et al., 2025), which frequently rely on rubric-based evaluations and thus naturally fit into our framework. Because judges often produce noisy or biased scores, Task-Calibrated Decoding could substantially improve the reliability of their evaluations (Lee et al., 2026). More broadly, TCD offers a principled way to turn black-box models into more reliable decision-makers for the concrete tasks they are deployed on.

### AI USE STATEMENT

In this work, we utilized generative AI tools for minor writing assistance, figure creation, and small modifications to existing code. However, we did not employ generative AI tools for core experiments or the ideation process. We carefully reviewed all AI-supported contributions by evaluating the generated code and adapting the written proposals. We take full responsibility for the final content of this work, including all text, claims, and artifacts produced with the assistance of generative AI.

### REPRODUCIBILITY STATEMENT

To ensure reproducibility, we publish the code and include detail description of implementation details and settings in Appendix B.

## REFERENCES

Momin Abbas, Yi Zhou, Parikshit Ram, Nathalie Baracaldo, Horst Samulowitz, Theodoros Salonidis, and Tianyi Chen. Enhancing in-context learning via linear probe calibration. In *Proceedings of the 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pages 307–315. PMLR, 2024. URL https://proceedings.mlr.press/v238/abbas24a.html.

Neil Band, Xuechen Li, Tengyu Ma, and Tatsunori Hashimoto. Linguistic calibration of long-form generations, 2024. URL https://arxiv.org/abs/2404.00474.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th international workshop on semantic evaluation (SemEval-2017)*, pages 1–14, 2017.

Prateek Chhikara. Mind the confidence gap: Overconfidence, calibration, and distractor effects in large language models. *arXiv preprint arXiv:2502.11028*, 2025.

Nico Daheim, Clara Meister, Thomas Möllenhoff, and Iryna Gurevych. Uncertainty-aware decoding with minimum bayes risk. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=hPpyUv1XyQ.

Morris H DeGroot and Stephen E Fienberg. The comparison and evaluation of forecasters. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 32(1-2):12–22, 1983.

Yi Dong, Zhilin Wang, Makesh Narsimhan Sreedhar, Xianchao Wu, and Oleksii Kuchaiev. Steerlm: Attribute conditioned sft as an (user-steerable) alternative to rlhf, 2023.

Bryan Eikema and Wilker Aziz. Sampling-based approximations to minimum Bayes risk decoding for neural machine translation. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 10978–10993, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.754. URL https://aclanthology.org/2022.emnlp-main.754/.

Lautaro Estienne, Matias Vera, Elizabeth Fons, Elena Kochkina, Pablo Piantanida, and Luciana Ferrer. Adapting language models to produce good class probabilities for classification tasks. *Transactions*

*on Machine Learning Research*, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=VVneIp69GR.

Yu Fei, Yifan Hou, Zeming Chen, and Antoine Bosselut. Mitigating label biases for in-context learning, 2023. URL https://arxiv.org/abs/2305.19148.

Isabel O Gallegos, Ryan A Rossi, Joe Barrow, Md Mehrab Tanjim, Sungchul Kim, Franck Dernoncourt, Tong Yu, Ruiyi Zhang, and Nesreen K Ahmed. Bias and fairness in large language models: A survey. *Computational linguistics*, 50(3):1097–1179, 2024.

Gemma Team. Gemma 3 technical report, 2025. URL https://arxiv.org/abs/2503.19786.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Saizhuo Wang, Kun Zhang, Yuanzhuo Wang, Wen Gao, Lionel Ni, and Jian Guo. A survey on llm-as-a-judge, 2025. URL https://arxiv.org/abs/2411.15594.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR, 2017.

Lukas Haas, Gal Yona, Giovanni D'Antonio, Sasha Goldshtein, and Dipanjan Das. Simpleqa verified: A reliable factuality benchmark to measure parametric knowledge, 2026. URL https://arxiv.org/abs/2509.07968.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=XPZIaotutsD.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021. URL https://arxiv.org/abs/2009.03300.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Lunjia Hu and Yifan Wu. Calibration error for decision making, 2024. URL https://arxiv.org/abs/2404.13503.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension, 2017. URL https://arxiv.org/abs/1705.03551.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. Language models (mostly) know what they know, 2022. URL https://arxiv.org/abs/2207.05221.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation, 2023. URL https://arxiv.org/abs/2302.09664.

Meelis Kull and Peter Flach. Novel decompositions of proper scoring rules for classification: Score adjustment as precursor to calibration. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 68–85. Springer, 2015.

Meelis Kull, Miquel Perello Nieto, Markus Kängsepp, Telmo Silva Filho, Hao Song, and Peter Flach. Beyond temperature scaling: Obtaining well-calibrated multi-class probabilities with dirichlet calibration. *Advances in neural information processing systems*, 32, 2019.

Chungpa Lee, Thomas Zeng, Jongwon Jeong, Jy yong Sohn, and Kangwook Lee. How to correctly report llm-as-a-judge evaluations, 2026. URL https://arxiv.org/abs/2511.21140.

Chengzu Li, Han Zhou, Goran Glavaš, Anna Korhonen, and Ivan Vulić. Large language models are miscalibrated in-context learners. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 11575–11596, 2025a.

Yibo Li, Miao Xiong, Jiaying Wu, and Bryan Hooi. Conftuner: Training large language models to express their confidence verbally, 2025b. URL https://arxiv.org/abs/2508.18847.

Alisa Liu, Xiaochuang Han, Yizhong Wang, Yulia Tsvetkov, Yejin Choi, and Noah A. Smith. Tuning language models by proxy. In *Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=dribhnhm1i.

Michal Lukasik, Harikrishna Narasimhan, Aditya Krishna Menon, Felix Yu, and Sanjiv Kumar. Regression-aware inference with llms, 2024. URL https://arxiv.org/abs/2403.04182.

Mathias Müller and Rico Sennrich. Understanding the properties of minimum Bayes risk decoding in neural machine translation. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 259–272, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.22. URL https://aclanthology.org/2021.acl-long.22/.

Preetum Nakkiran, Arwen Bradley, Adam Goliński, Eugene Ndiaye, Michael Kirchhof, and Sinead Williamson. Trained on tokens, calibrated on concepts: The emergence of semantic calibration in llms, 2025. URL https://arxiv.org/abs/2511.04869.

Georgy Noarov and Aaron Roth. Calibration for decision making: A principled approach to trustworthy ml, 2024. URL https://www.let-all.com/blog/2024/03/13/calibration-for-decision-making-a-principled-approach-to-trustworthy-ml/.

Aitor Ormazabal, Mikel Artetxe, and Eneko Agirre. CombLM: Adapting black-box language models through small fine-tuned models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2961–2974. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.180. URL https://aclanthology.org/2023.emnlp-main.180/.

Maja Pavlovic. Understanding model calibration - a gentle introduction and visual exploration of calibration and the expected calibration error (ece). In *ICLR Blogposts 2025*, 2025. URL https://iclr-blogposts.github.io/2025/blog/calibration/. https://iclr-blogposts.github.io/2025/blog/calibration/.

Benjamin Plaut, Nguyen X. Khanh, and Tu Trinh. Probabilities of chat llms are miscalibrated but still predict correctness on multiple-choice qa, 2025. URL https://arxiv.org/abs/2402.13213.

Teodora Popordanoska, Raphael Sayer, and Matthew B. Blaschko. A consistent and differentiable lp canonical calibration error estimator, 2022. URL https://arxiv.org/abs/2210.07810.

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

Hayley Ross, Ameya Sunil Mahabaleshwarkar, and Yoshi Suhara. When2Call: When (not) to call tools. In Luis Chiruzzo, Alan Ritter, and Lu Wang, editors, *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3391–3409, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.174. URL https://aclanthology.org/2025.naacl-long.174/.

Maohao Shen, Subhro Das, Kristjan Greenewald, Prasanna Sattigeri, Gregory W. Wornell, and Soumya Ghosh. Thermometer: Towards universal calibration for large language models. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 44687–44711. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/shen24c.html.

Lin Shi, Chiyu Ma, Wenhua Liang, Xingjian Diao, Weicheng Ma, and Soroush Vosoughi. Judging the judges: A systematic study of position bias in LLM-as-a-judge. In Kentaro Inui, Sakriani Sakti, Haofen Wang, Derek F. Wong, Pushpak Bhattacharyya, Biplab Banerjee, Asif Ekbal, Tanmoy Chakraborty, and Dhirendra Pratap Singh, editors, *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pages 292–314, Mumbai, India, December 2025. The Asian Federation of Natural Language Processing and The Association for Computational Linguistics. ISBN 979-8-89176-298-5. doi: 10.18653/v1/2025.ijcnlp-long.18. URL https://aclanthology.org/2025.ijcnlp-long.18/.

Tianxiang Sun, Yunfan Shao, Hong Qian, Xuanjing Huang, and Xipeng Qiu. Black-box tuning for language-model-as-a-service. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 20841–20855. PMLR, 2022. URL https://proceedings.mlr.press/v162/sun22e.html.

OpenAI Team. Openai gpt-5 system card, 2026. URL https://arxiv.org/abs/2601.03267.

Tim Tomov, Dominik Fuchsgruber, and Stephan Günnemann. Task-awareness improves llm generations and uncertainty, 2026. URL https://arxiv.org/abs/2601.21500.

Gerrit J.J. Van Den Burg, Gen Suzuki, Wei Liu, and Murat Sensoy. Aligning black-box language models with human judgments. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 6752–6764. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-naacl.376. URL https://aclanthology.org/2025.findings-naacl.376/.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

Zhilin Wang, Yi Dong, Jiaqi Zeng, Virginia Adams, Makesh Narsimhan Sreedhar, Daniel Egert, Olivier Delalleau, Jane Polak Scowcroft, Neel Kant, Aidan Swope, and Oleksii Kuchaiev. Helpsteer: Multi-attribute helpfulness dataset for steerlm, 2023.

Sean Wu, Fredrik K. Gustafsson, Edward Phillips, Boyan Gao, Anshul Thakur, and David A. Clifton. Bas: A decision-theoretic approach to evaluating large language model confidence, 2026. URL https://arxiv.org/abs/2604.03216.

Johnathan Xie, Annie S Chen, Yoonho Lee, Eric Mitchell, and Chelsea Finn. Calibrating language models with adaptive temperature scaling. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 18128–18138, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.1007. URL https://aclanthology.org/2024.emnlp-main.1007/.

Yongjin Yang, Haneul Yoo, and Hwaran Lee. Maqa: Evaluating uncertainty quantification in llms regarding data uncertainty, 2025. URL https://arxiv.org/abs/2408.06816.

Shengjia Zhao, Michael P. Kim, Roshni Sahoo, Tengyu Ma, and Stefano Ermon. Calibrating predictions to decisions: A novel approach to multi-class calibration, 2021. URL https://arxiv.org/abs/2107.05719.

# A Additional Experiments

## A.1 Ablations

**Ablations on Number of Monte Carlo Samples.** In Figure 4, we ablate the performance of TCD on Qwen3-4B on a varying number of Monte-Carlo samples $M$ to estimate the task-specific distribution $\hat{p}(\ell \mid \mathrm{x})$. For HelpSteer, STSB, SimpleQA, and vector-valued HelpSteer, we observe saturation, which justifies our default choice of $M = 20$. For MMLU and When2Call, performance increases well beyond 20 MC samples: This highlights that our method can also enable a trade-off between performance and computational effort at inference. At our default choice of $M = 20$, on both datasets, TCD outperforms other methods to utilize supervised in-domain data. For each dataset, we generate a pool of 100 samples per question and subsample it to $M \in \{2, 3, 5, 10, 20, 30, 50, 75\}$, drawing 10 random subsets for each $M$ ($M = 100$ uses the full pool). Shaded bands show the standard deviation across subsets, and the dashed vertical line marks $M = 20$.

**Ablations on Calibration Set Size.** We also ablate the effect of the calibration dataset size on the performance of TCD versus other methods in Figure 5. While on SimpleQA and TriviaQA, we observe saturation quickly, on the other datasets we clearly see that more calibration data improves the performance of all methods, including TCD. Importantly, across a broad range of data regimes, TCD remains the most effective method for utilizing labeled in-domain data. To isolate the effect of $n$, we keep the five-fold split fixed and subsample only each fold's training set to $n \in \{25, 50, 100, 200, 400\}$, so that all settings are evaluated on the same held-out questions. For each $n$, we draw 10 random subsets. Shaded bands show the standard deviation across these subsets, reflecting sensitivity to which labeled examples are drawn.

**Ablations on Prompts** The prompt determines the model's behavior and hence $\hat{p}(\ell \mid x)$ for every input. Since the calibration map is fit on these distributions, a natural question is whether it remains valid when the prompt changes but the task stays the same. To test this, we rewrite the HelpSteer scoring instruction in three ways (Listings 5 to 7), varying how the rating task is described, and regenerate $M = 20$ samples per question under each prompt. We then compare two calibration maps on the new prompt's samples: a *frozen* map, fit on the original prompt and applied unchanged, and a *refit* map, fit on the new prompt. Their difference measures the cost of not refitting after a prompt change. Table 3 shows that the calibration map transfers: refitting improves the $L_1$ loss only marginally. A map fit once, thus remains useful under moderate rewording.

Table 3: Transferability of TCD across prompts for Qwen3-4B on HelpSteer

| Prompt | TCD (frozen) | TCD (refit) | refit − frozen |
|---|---|---|---|
| original | 0.788 | 0.788 | 0.000 |
| concise | 0.805 | **0.786** | $-0.019_{\pm 0.008}$ |
| rubric | 0.772 | **0.769** | $-0.003_{\pm 0.004}$ |
| evaluator | 0.778 | **0.770** | $-0.008_{\pm 0.009}$ |

## A.2 Calibration Maps

Figures 6 and 7 show how the learned recalibration maps act on the binary latent space $\{A, \perp\}$: In Figure 6, we see that for all LLMs but the Qwen3-30B-A3B model, the learned calibration map counteracts the strong overconfidence in answering by collapsing all actions onto abstaining $\perp$ by predicting a probability below the threshold $t$. While this results in the calibrated LLM always abstaining, Table 1 confirms that this result in lower average loss. One explanation is that the three LLMs for which this behaviour occurs are all poor predictors of the true label and, therefore, the best calibration map in terms of the BAS risk is to always abstain instead of trusting the LLM's predicted distribution. In contrast, for TriviaQA, we, too, observe overconfidence of the LLMs in Figure 7 that is mitigated by our task calibration: Only uncalibrated probabilities that are close to one (i.e. very high confidence) get mapped to values above the answer threshold $t$.

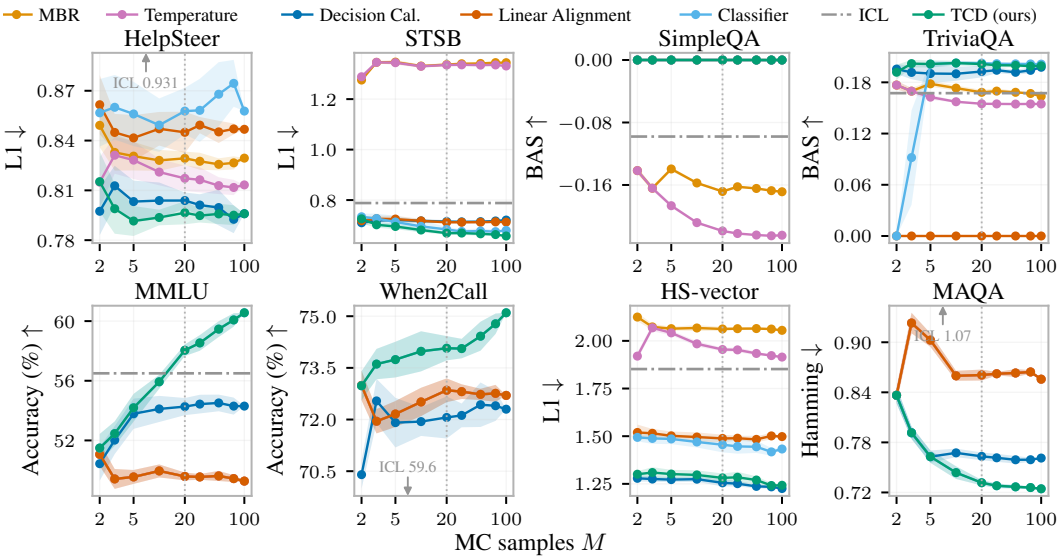



Figure 4: **Number of Monte Carlo samples.** Task loss vs. number of samples $M$ used to estimate $\hat{p}(\ell \mid x)$ for Qwen3-4B; bands show the standard deviation over 10 subsamples of a 100-sample pool. Dashed: default $M = 20$.

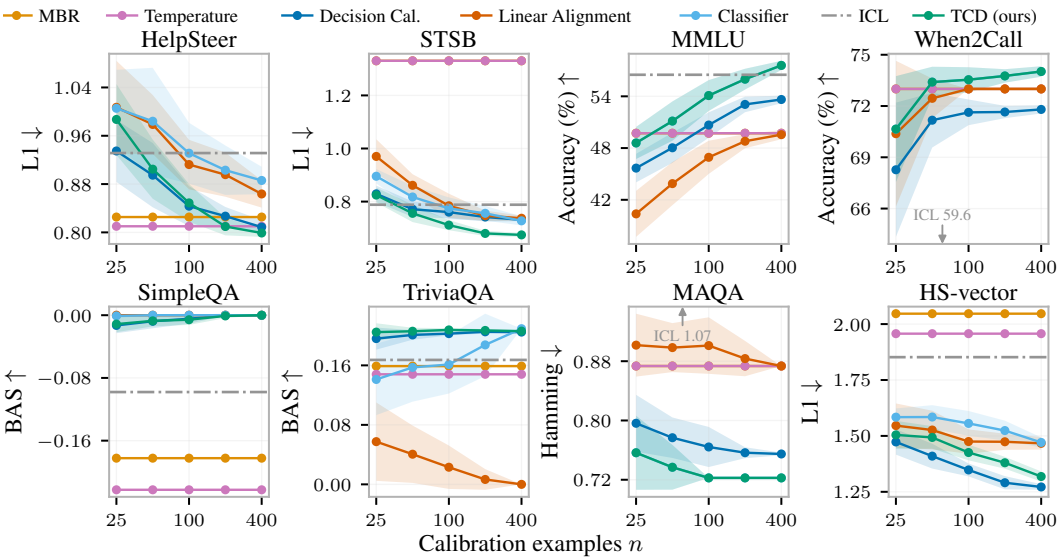
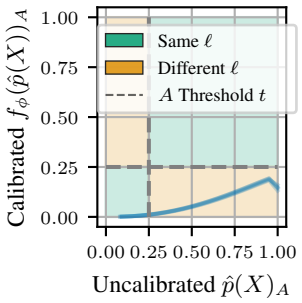
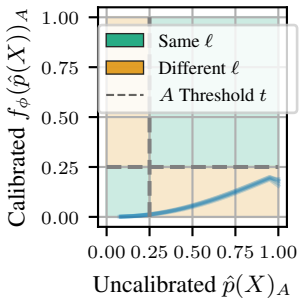


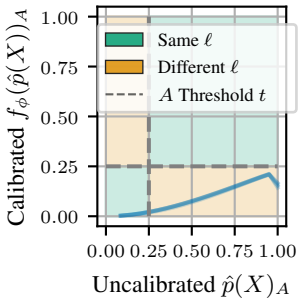

Figure 5: **Calibration set size.** Task loss vs. number of labeled calibration examples $n$ for Qwen3-4B, evaluated on the same held-out questions for all $n$; bands show the standard deviation over 10 random subsets.



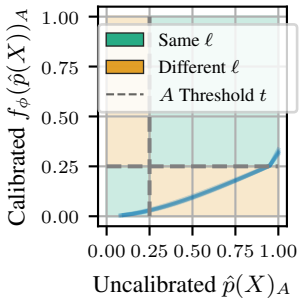

(6a) Gemma-3-4B




(6b) Gemma-3-12B

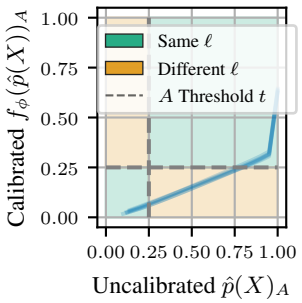



(6c) Qwen3-4B

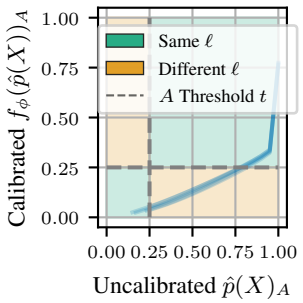



(6d) Qwen-3-30B-A3B

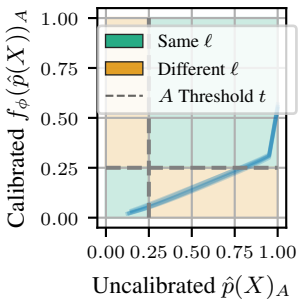
Figure 6: Learned calibration maps $f_\phi(\hat{p}(\mathrm{x})$ on SimpleQA.



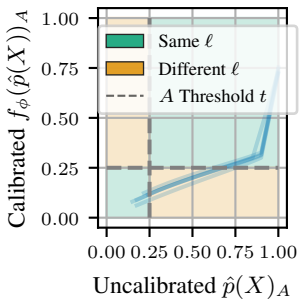

(7a) Gemma-3-4B




(7b) Gemma-3-12B




(7c) Qwen3-4B




(7d) Gemma-3-30B-A3B

Figure 7: Learned calibration maps $f_\phi(\hat{p}(\ell \mid \mathrm{x}))$ on TriviaQA.

# B Implementation Details

## B.1 Baselines

**Calibration** When fitting calibrators (temperature scaling and Dirichlet calibration) using NLL loss, we transform the logarithm of the predicted latent beliefs $\hat{p}(\mathrm{x})$. We add $\epsilon = 10^{-12}$ to these

beliefs to prevent taking the logarithm of zero. In particular, temperature scaling with a positive temperature $\tau$ is realized by optimizing over the logarithm of the temperature $t = \log \tau$:

$$f_t(q) = \mathrm{softmax}(\exp(t)^{-1} \log(q + \epsilon)), \qquad t \in \mathbb{R}. \tag{5}$$

Similarly, Dirichlet calibration is realized as:

$$f_{\boldsymbol{W},\boldsymbol{b}}(q) = \mathrm{softmax}(\boldsymbol{W} \log(q + \epsilon) + \boldsymbol{b}), \qquad \boldsymbol{W} \in \mathbb{R}^{C \times C}, \boldsymbol{b} \in \mathbb{R}^{C}. \tag{6}$$

Both methods use MBR decoding subsequently to produce the final answer.

**LoRA** As LoRA parameters, we use Rank $r = 8$, $\alpha = 16$, dropout $0.05$ across all models. For Gemma-3-4B, Gemma-3-12B and Qwen3-4B we adapt attention and MLP projections; for Qwen3-30B-A3B only attention due to computational constraints. We train for at most $10$ epochs with effective batch size $8$ (per-device batch $1$, gradient accumulation $8$), learning rate $10^{-4}$, no weight decay, warmup ratio $0.03$, bf16 and gradient checkpointing, with early stopping on validation loss (patience $2$). Each training example is the dataset's own evaluation prompt, followed by the reference answer as the assistant's turn. At inference, the adapted model is decoded exactly as the base model; the reported LoRA row uses beam decoding. LoRA reuses the 5-CV calibration fold structure and is trained on the 4 folds and evaluated on a fifth held-out fold. The numbers reported in Table 2 are hence only on this held-out fold for both LoRA and TCD.

**In-Context Learning** The in-context baseline answers each query directly, without Monte Carlo sampling or calibration. For every question we retrieve the $3$ nearest other questions in the same dataset under cosine similarity of google/embeddinggemma-300m sentence embeddings, excluding the query itself, and prepend them as solved demonstrations together with their reference answers; the query is then appended after a short instruction. Retrieval is leave-one-out, so a question is never shown its own answer. The prompt is otherwise the dataset's own evaluation prompt, and the demonstrations omit the answer-format instruction, which is stated once for the query. We decode a single answer with beam search, so no predictive distribution is formed and no calibrator is applied.

**Linear Alignment** We follow Van Den Burg et al. (2025), who one-hot encode the model's selected category and the human label over a shared action space of size $C$ and solve the regularized least-squares problem

$$\hat{\boldsymbol{W}} = \arg\min_{\boldsymbol{W}} \|\boldsymbol{Z}\boldsymbol{W} - \boldsymbol{Y}\|_F^2 + \lambda \|\boldsymbol{W}\|_F^2 = (\boldsymbol{Z}^\top \boldsymbol{Z} + \lambda \boldsymbol{I})^{-1} \boldsymbol{Z}^\top \boldsymbol{Y}, \tag{7}$$

with $\lambda = 10^{-6}$, and take the aligned judgment to be $\arg\max_j (\boldsymbol{z}^\top \hat{\boldsymbol{W}})_j$. As the model-selected category, we take the argmax of the initial distribution.

**Decision Calibration** We use the discrete decision calibrator of Zhao et al. (2021), which iteratively corrects the predicted beliefs so that they are unbiased conditional on the action the MBR decision rule selects. Importantly, we assume the loss function $d_T$ to be known and, as such, retrieve an easier algorithm than Zhao et al. (2021) Algorithm 1, as we do not need to optimize over a family of loss functions.

**Computational Resources.** We evaluate all LLMs with default hyperparameters from HuggingFace (top-k, top-p, temperature, etc.) to ensure a realistic practical environment. We perform inference on two types of machines: (i) Nvidia H100 GPUs, and (ii) Nvidia H200 GPUs. Calibration is done on these machines as well, even though they are computationally lightweight and can easily be done on CPU-only machines as well. While inference (sampling responses from the LLM) takes up to multiple days for the biggest datasets with longest prompts and largest LLMs, calibration runs finish within a few minutes on our machines.

### B.2 Datasets.

We detail the datasets, corresponding tasks and the associated latent structures and loss functions in Table 4. Here, we overload elements $\ell$ which do not represent one-hot vectors in this context but instead the true corresponding latent meaning. In practice, each one-hot representation directly

Table 4: Datasets with tasks $T$, latent structures $\mathcal{L}$, and task-dependent loss functions $d_T$.

| Task | Ordinal Regression | | Classification | | Answer-or-Abstain | | Open-Set Prediction | Vector Ordinal Regression |
|---|---|---|---|---|---|---|---|---|
| Dataset | Helpsteer | STSB | MMLU | When2Call | SimpleQA | TriviaQA | MAQA | Helpsteer |
| Latent Structure $\mathcal{L} \cong$ | $\{0, \dots, 4\}$ | $\{1, \dots 5\}$ | {(a), (b), (c), (d)} | {"call tool", "request info", "abstain"} | $\{A, \perp\}$ | $\{A, \perp\}$ | $\{0,1\}^{\leq C}$ | $\{0, \dots, 4\}^2$ |
| Task-Dependent Loss $d_T(\ell, \ell')$ | $\lvert\ell - \ell'\rvert$ | $\lvert\ell - \ell'\rvert$ | $\mathbf{1}[\ell \neq \ell']$ | $\mathbf{1}[\ell \neq \ell']$ | $d_t^{\text{BAS}}(\ell, \ell')$ | $d_t^{\text{BAS}}(\ell, \ell')$ | $\mathbf{1}[\ell_i \neq \ell'_i]$ for binary decisions about including answer $i$ | $\lVert\ell - \ell'\rVert_1$ |

corresponds to one latent interpretation, so the one-hot representation of a structure and its "true meaning" are isomorphic (denoted by $\cong$).

We also briefly describe each dataset and the associated LLM task. (i) For Helpsteer, the LLM is shown a prompt and corresponding response and should rate the response according to one or more criteria. For the ordinal regression task, we choose the criterion "correctness" and let the LLM output a natural number for a rating between 0 and 4. (ii) For STSB, the LLM is shown two sentences and it should rate their similarity on a scale on 1 to 5. We again only consider natural numbers through rounding to the closest integer. (iii) For MMLU, the LLM is shown a question and four potential answers enumerated with (a) to (d). The LLM should select the correct answer among the provided candidates. (iv) For When2Call, we present the LLM with a prompt and a set of available tools. The LLM should then assess whether to call one of the tools, request more information from user or abstain from answering alltogether. (v) For SimpleQA and (vi) TriviaQA, we present the LLM with a question and ask for the answer. Whether the LLM action is to answer $A$ or abstain $\perp$ is determined by the relative frequency of the semantic majority answer (see Appendix B.3). (vii) For MAQA, we present the LLM with an ambiguous question that permits multiple answers. The LLM should list all valid answers. We construct a set of all observed answers in the generated responses and the truth and then decompose this decision task into binary decision tasks of whether to include an individual answer $i$ in the final response. (viii) For Vector-valued Ordinal Regression, we again use Helpsteer and show the LLM a prompt and a corresponding response. It now should rate the response in terms of two criteria simultaneously ("correctness", "helpfulness") by outputting two integer scores in $\{0, \dots, 4\}$.

**Data selection.** For all datasets except MAQA, we use HuggingFace as a data source: for STSB, we use mteb/stsbenchmark-sts, for MMLU, we use cais/mmlu, for When2Call, we use nvidia/When2Call, for SimpleQA, we use google/simpleqa-verified, for TriviaQA, we use mandarjoshi/trivia_qa, for Helpsteer, we use nvidia/HelpSteer. All datasets correspond to their latest available versions as of May 2026. For MAQA we use the official repository MAQA-Data.

Unless stated otherwise we evaluate 1000 questions per dataset, drawn once and reused across every model, decoding strategy and ablation, so that all runs of a cell address the same questions. For MMLU we use the `moral_scenarios` subject (`test` split) in its entirety, 895 questions; likewise we use all 592 questions of the MAQA world-knowledge set. When2Call (`mcq` split), SimpleQA-Verified (`eval` split) and TriviaQA (configuration `rc.nocontext`) are taken as the first 1000 rows of their respective splits. For STSB, we sample 1000 questions randomly. For HelpSteer (`train` split) we sample 1000 responses with uniform stratification over the 0–4 label of the `correctness` criterion, so that all five scores are represented rather than following the corpus's skew; the vector-valued variant applies the same stratified sampling jointly over `correctness` and `helpfulness`.

### B.3 Mapping Language Outputs to Latent Structures

Here, we detail how we realize the mappings $g_T$ to free-form text LLM responses into the corresponding latent structures. As our core assumption is that the LLM outputs free-form text that is embedded in the latent structure, i.e. the LLM is able to reasonable address the task we prompt it with, we discard answers that can not properly parsed through any mapping $g_T$. These occurrences are extremely rare as the instruction-tuned LLMs studied in this work respect the instruction prompts

sufficiently well. Therefore, $g_T$ can often be implemented just with simple Regular Expressions or lightweight Natural Language Inference models. In general, for all structures $\mathcal{L}$ considered in this work, we empirically find that the mapping $g_T$ does not lose any semantically relevant information and succesfully extracts the correct meaning from a response in almost all cases.

**Ordinal Regression** Models are instructed to follow a specific output format as seen in Appendix G. The LLM's output is post-processed by an auxiliary language model extracting the integer response (again, see Appendix G) and then parsed using a Regex.

**Classification** Models are instructed to follow a specific output format as seen in Appendix G. The answer is then passed using a Regex.

**Answer-or-Abstain** Similar to Kuhn et al. (2023), we use DeBERTa He et al. (2021) as an entailment model to group the individual free-form text responses into semantic clusters that form the support of the domain over which the most frequent answer is selected. All elements falling into the cluster of the most frequent answer are mapped to `Answer` while all others are mapped to $\perp$.

**Open Set Prediction** Given an answer string $s$, we want to extract all *semantically distinct* answers in that string. As an example to the question "Which ocean borders the USA? " the string $s =$" Both the Atlantic and Pacific Ocean border the USA. "gets mapped to $\{\texttt{Atlantic}, \texttt{Pacific}\}$ by $g_T$. Concretely $g_T$ is implemented using `gpt-5-mini` with prompts in (Appendix G).

# C Related Work Extended

## C.1 BAS: A Decision-Theoretic Approach to Evaluating LLM Confidence

The BAS metric of Wu et al. (2026) can be viewed as a special case of our latent-structure decision framework. Let $\mathcal{L} = \{\mathrm{A}, \perp\}$, where A denotes `answer` and $\perp$ denotes `abstain`. We identify both the latent and action space with these two selective decisions. For a fixed threshold $t \in [0, 1)$, define the task loss

$$d_T^{\mathrm{BAS}}(a, b) = \begin{cases} 0, & b = \perp, \\ -1, & b = \mathrm{A},\ a = \mathrm{A}, \\ \frac{t}{1-t}, & b = \mathrm{A},\ a = \perp . \end{cases} \tag{8}$$

Importantly, the loss is asymmetric, and $a$ corresponds to the true decision while $b$ denotes the model prediction. For a given $x$, let $\hat{p}(\ell \mid x)_{\mathrm{A}}$ denote the predictive probability that answering is correct. The corresponding Bayes decision is

$$\delta_t^{\mathrm{MBR}}(\hat{p}(\ell \mid x)) = \arg\min_{b \in \{\mathrm{A}, \perp\}} \mathbb{E}_{a \sim \hat{p}(\ell|x)} \big[ d_t^{\mathrm{BAS}}(a, b) \big].$$

This coincides with the BAS answer-or-abstain policy. To see this, we compare the two possible actions:

$$\mathbb{E}_{a \sim \hat{p}(\ell|x)}[d_t^{\mathrm{BAS}}(a, \perp)] = 0,$$

and

$$\mathbb{E}_{a \sim \hat{p}(\ell|x)}[d_t^{\mathrm{BAS}}(a, \mathrm{A})] = -\hat{p}(\ell \mid x)_{\mathrm{A}} + \frac{t}{1-t} \hat{p}(\ell \mid x)_{\perp}.$$

Thus, choosing $b = \mathrm{A}$ is optimal iff

$$-\hat{p}(\ell \mid x)_{\mathrm{A}} + \frac{t}{1-t} \hat{p}(\ell \mid x)_{\perp} \leq 0.$$

Using $\hat{p}(\ell \mid x)_{\perp} = 1 - \hat{p}(\ell \mid x)_{\mathrm{A}}$, this is equivalent to

$$-\hat{p}(\ell \mid x)_{\mathrm{A}} + \frac{t}{1-t}(1 - \hat{p}(\ell \mid x)_{\mathrm{A}}) \leq 0 \iff \hat{p}(\ell \mid x)_{\mathrm{A}} \geq t.$$

Therefore,

$$\delta_t^{\mathrm{MBR}}(\hat{p}(\ell \mid x)) = \begin{cases} \mathrm{A}, & \hat{p}(\ell \mid x)_{\mathrm{A}} \geq t, \\ \perp, & \text{otherwise}, \end{cases}$$

which exactly recovers the BAS answer-or-abstain rule. In our experiments, we set $t = 0.25$, and the implementation of the mapping $g_T$ is detailed in Appendix B.3.

# D TASK CALIBRATION

Here, we discuss task calibration in which we require full distributional calibration (Equation (1)) for the task-dependent categorical latent distribution $\hat{p}(\ell \mid x)$ of the LLM. The key result in this context is that among all decision rules $\delta : \Delta^{C-1} \mapsto \mathcal{L}$ that operate on this latent distribution $\hat{p}(\ell \mid \mathrm{x})$, it is optimal to first recalibrate with the true marginal recalibration map (Equation (4)) and then apply Minimum Bayes Risk (MBR) decoding in terms of the true expected task-dependent loss $d_T$ under the true data distribution $p^*(\mathrm{x}, \mathrm{y})$:

**Proposition D.1.** *For any loss function $d_T : \mathcal{L} \times \mathcal{L} \mapsto \mathbb{R}$ among all decision strategies $\delta' : \Delta^{C-1} \mapsto \mathcal{L}$, the MBR decision $\delta^{\mathrm{MBR}}$ on $f^*_{g_T}(\hat{p}(\ell \mid \mathrm{x}))$ is optimal in terms of the expected incurred loss w.r.t the true distribution $p^*(\mathrm{x}, \mathrm{y})$ on any subset $P \subseteq \Delta^{C-1}$:*

$$\mathbb{E}_{\mathrm{x},\mathrm{y}}\left[d_T(g_T(\mathrm{y}), \delta^{\mathrm{MBR}}(f^*_{g_T}(\hat{p}(\ell \mid \mathrm{x})))) \mid \hat{p}(\mathrm{x}) \in P\right] \leq \mathbb{E}_{\mathrm{x},\mathrm{y}}\left[d_T(g_T(\mathrm{y}), \delta'(\hat{p}(\ell \mid \mathrm{x}))) \mid \hat{p}(\mathrm{x}) \in P\right]. \tag{9}$$

# E TASK CALIBRATION ERROR

## E.1 THEORY

The notion of optimal decision-making under task calibration also enables quantifying the calibration error of the LLM's induced latent distribution with respect to its task-calibrated counterpart, which we call Task Calibration Error (TCE). It computes how much additional task-dependent loss $d_T$ is incurred by the MBR action on the uncalibrated latent distribution $\hat{p}(\mathrm{x})$ compared to the MBR action on the task calibrated latent distribution. To derive it, we first exploit that the MBR decision rule $\delta^{\mathrm{MBR}}$ induces a proper scoring rule (Hu and Wu, 2024) $S_T : \Delta^{C-1} \times \mathcal{L} \mapsto \mathbb{R}$ defined as: $S_T(q, \ell) = d_T\big(\ell, \delta^{\mathrm{MBR}}(q)\big)$. Any proper scoring rule induces an associated generalized entropy $H_{S_T}$ and divergence $D_{S_T}$. In our setting, the induced divergence admits the representation (Proposition F.2):

$$D_{S_T}(q \,\|\, r) = R_{\mathrm{Bayes}}(\delta^{\mathrm{MBR}}(q), r) - R_{\mathrm{Bayes}}(\delta^{\mathrm{MBR}}(r), r). \tag{10}$$

$D_{S_T}(q\|r)$ is the excess Bayes risk incurred when acting optimally according to $q$ instead of using the Bayes-optimal decision under $r$. Using this divergence, we decompose the discrepancy between the predicted latent distribution $\hat{p}(\mathrm{x})$ and the true latent distribution $p^*(\ell|\mathrm{x})$ into a calibration term and an irreducible refinement term Kull and Flach (2015), the former of which describes the task-specific calibration error:

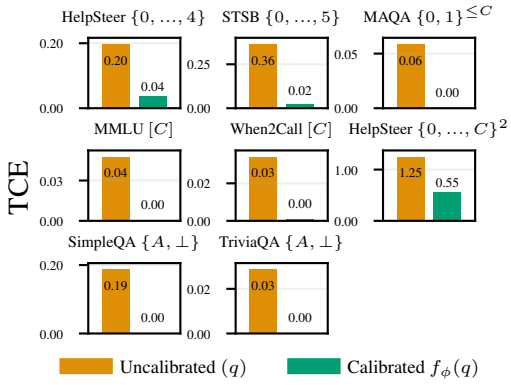
**Proposition E.1.** *We define the Task Calibration Error (TCE) by decomposing the $D_{S_T}$ between the predicted latent distribution $\hat{p}(\ell \mid \mathrm{x})$ and true latent distribution $p^*(\ell|\mathrm{x})$ as:*

$$\mathbb{E}_X\left[D_{S_T}\big(\hat{p}(\ell \mid \mathrm{x}) \,\|\, p^*(\ell \mid \mathrm{x})\big)\right] = \underbrace{\mathbb{E}_X\left[D_{S_T}\big(\hat{p}(\ell \mid \mathrm{x}) \,\|\, f^*_{g_T}(\hat{p}(\ell \mid \mathrm{x}))\big)\right]}_{\textit{Task Calibration Error (TCE)}} + \mathbb{E}_X\left[D_{S_T}\big(f^*_{g_T}(\hat{p}(\ell \mid \mathrm{x})) \,\|\, p^*(\ell \mid \mathrm{x})\big)\right]. \tag{11}$$

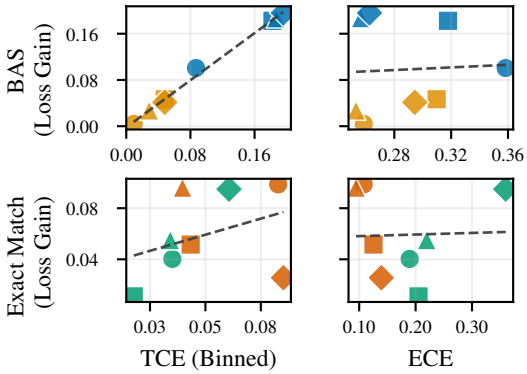
The first term—the *Task Calibration Error (TCE)*—captures the average excess task loss caused by miscalibration and quantifies the benefit obtainable through optimal recalibration. It corresponds to the calibration fixed decision loss (CFDL) of Hu and Wu (2024) applied to beliefs over the latent structure $\mathcal{L}$. The second term (refinement loss(Kull and Flach, 2015)) cannot be reduced with calibration.

## E.2 EXPERIMENTS

**Setup** We compute the TCE on the whole dataset according to Proposition E.1, by approximating $f^*_{g_T}$ using ground-truth data and discretizing the simplex into $4^{C-1}$ equal-volume bins (see Appendix E.3). This is a multidimensional extension of binning-based ECE estimation. In Appendix E.2, we additionally ablate using a Dirichlet KDE Popordanoska et al. (2022).




(8a) Task Calibration Error (TCE) for uncalibrated $q$ and Dirichlet-calibrated $f_\phi(q)$ model on Gemma 3-12B. Across all tasks, the model is severely task miscalibrated, which is mitigated by our approximate recalibration method.




(8b) Task Calibration Error (TCE) correlates with downstream gains, whereas ECE does not. Datasets: ● SimpleQA, ● TriviaQA, ● When2Call, ● MMLU. Models: ● Qwen3-30B-A3B, ■ Qwen3-4B, ▲ Gemma-3-12B, ◆ Gemma-3-4B

**Results** We evaluate the quality of our recalibration map $f_\phi$ through the Task Calibration Error (TCE). We compare the TCE of the the original model predictions $\hat{p}(\ell \mid \mathrm{x})$ with their recalibrated counterparts $f_\phi(\hat{p}(\ell \mid \mathrm{x}))$. Figure 8a shows that the LLMs are often substantially task miscalibrated, which indicates possible performance improvements through task calibration. The full results can be seen in Table 5. Importantly, Dirichlet recalibration drastically reduces TCE – often to zero –, demonstrating that the approximately recalibrated predictor $\delta^{\mathrm{MBR}}(f_\phi(\hat{p}(\ell \mid \mathrm{x})))$ recovers most of the theoretically attainable improvement. It also verifies that a simple affine (Dirichlet) recalibration suffices to find MBR actions that are optimal according to Proposition 3.1, consistent with the strong empirical decoding performance seen in Table 1.

Table 5: Task Calibration Error (TCE) using binned estimator for uncalibrated $q$ and Dirichlet-calibrated $f_\phi(q)$ model. All LLMs are severely task miscalibrated which is mitigated by our approximate recalibration method.

| **Model/Dataset** | **Latent** $\mathcal{L}$ | $\{0,\ldots,4\}$ + L1 HelpSteer | $\{0,\ldots,5\}$ + L1 STSB | $[C]$ + Exact Match MMLU | $[C]$ + Exact Match When2Call | $\{A,\perp\}$ + BAS SimpleQA | $\{A,\perp\}$ + BAS TriviaQA | $\{0,1\}^{\leq C}$ + Hamming MAQA | $\{0,\ldots,C\}^2$ + L1 HelpSteer |
|---|---|---|---|---|---|---|---|---|---|
| Gemma-3-4B | $q$ | 0.282 | 0.323 | 0.085 | 0.061 | 0.196 | 0.048 | 0.103 | 1.216 |
| | $f_\phi(q)$ | **0.099** | **0.050** | **0.030** | **0.001** | **0.000** | **0.004** | **0.000** | **0.424** |
| Gemma-3-12B | $q$ | 0.197 | 0.364 | 0.040 | 0.034 | 0.187 | 0.029 | 0.059 | 1.249 |
| | $f_\phi(q)$ | **0.037** | **0.023** | **0.000** | **0.001** | **0.000** | **0.000** | **0.000** | **0.547** |
| Qwen3-4B | $q$ | 0.139 | 0.692 | 0.043 | 0.018 | 0.182 | 0.048 | 0.137 | 1.375 |
| | $f_\phi(q)$ | **0.039** | **0.033** | **0.006** | **0.004** | **0.000** | **0.004** | **0.000** | **0.594** |
| Qwen3-30B-A3B | $q$ | 0.117 | 0.589 | 0.083 | 0.035 | 0.087 | 0.009 | 0.052 | 1.449 |
| | $f_\phi(q)$ | **0.026** | **0.041** | **0.006** | **0.001** | **0.000** | **0.007** | **0.000** | **0.530** |

We also verify the practical usefulness of TCE by investigating how well it predicts performance gains achieved by recalibration and compare it to the Expected Calibration Error (ECE) DeGroot and Fienberg (1983); Guo et al. (2017) computed from the LLM's latent distribution $\hat{p}(\ell \mid \mathrm{x})$. As shown in Figure 8b, in contrast to ECE, TCE exhibits a strong correlation with the realized loss reduction obtained through recalibration. Full results can be seen in Figure 9. Notably, as TCE is defined directly in terms of the task-specific loss, it permits a task-dependent interpretation, and the concrete values are meaningful. In contrast, ECE quantifies an average difference in confidence that cannot be directly tied to the task's loss.

**Ablation with Dirichlet Kernel Estimator** To assess the robustness of our conclusions, we also employ the estimator of Popordanoska et al. (2022) with a Dirichlet kernel of fixed bandwidth $0.01$. The corresponding results are reported in Table 6. Overall, they match those obtained with the binning-based estimator: TCE is substantially reduced after calibration and is often close to zero, indicating that most of the achievable improvement is recovered. Occasionally, discrepancies arise when the conditional estimator is supported by very few samples (sometimes a single data point), reflecting the sparsity of the empirical distribution in those regions.

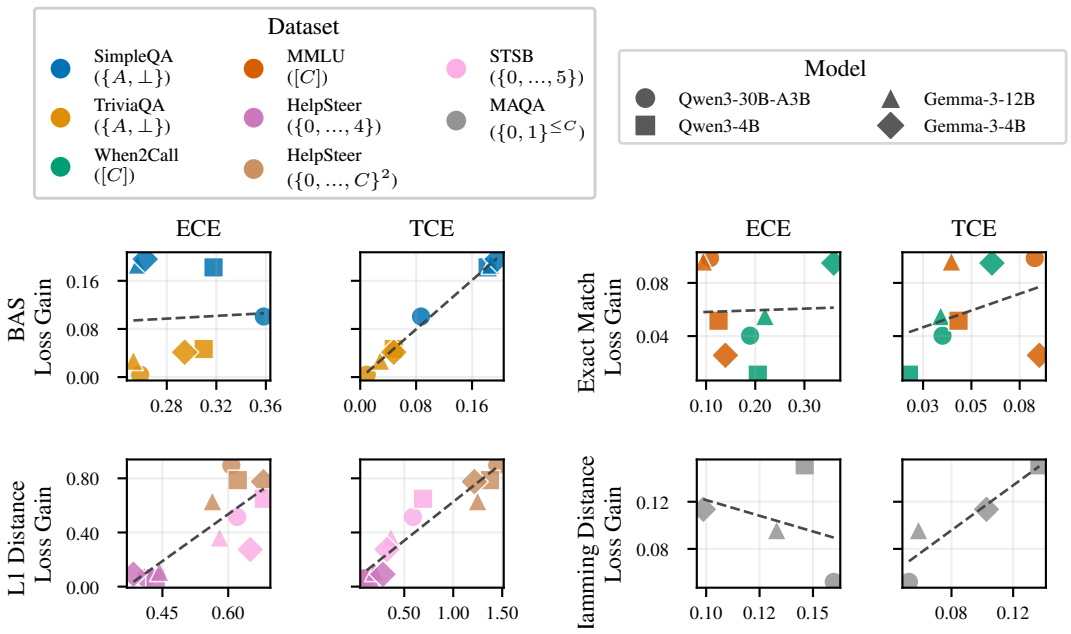



Figure 9: Task Calibration Error (TCE) and ECE for all Tasks

Table 6: Task Calibration Error (TCE) using Dirichlet kernel estimator for uncalibrated $q$ and Dirichlet-calibrated $f_{\phi^*}(q)$ model.

| **Model/Dataset Latent** $\mathcal{L}$ | | $\{0,...,4\}$ + L1 HelpSteer | $\{0,...,5\}$ + L1 STSB | $[C]$ + Exact Match MMLU | $[C]$ + Exact Match When2Call | $\{A,\perp\}$ + BAS SimpleQA | $\{A,\perp\}$ + BAS TriviaQA | $\{0,1\}^{\leq C}$ + Hamming MAQA | $\{0,...,C\}^2$ + L1 HelpSteer |
|---|---|---|---|---|---|---|---|---|---|
| Gemma-3-4B | $q$ | 0.266 | 0.340 | **0.124** | 0.112 | 0.196 | 0.059 | 0.678 | **0.212** |
| | $f_\phi(q)$ | **0.182** | **0.072** | 0.145 | **0.001** | **0.000** | **0.014** | **0.000** | 0.299 |
| Gemma-3-12B | $q$ | 0.238 | 0.419 | 0.129 | 0.068 | 0.187 | 0.035 | 1.036 | 0.118 |
| | $f_\phi(q)$ | **0.114** | **0.048** | **0.019** | **0.003** | **0.000** | **0.002** | **0.206** | **0.115** |
| Qwen3-4B | $q$ | 0.253 | 0.719 | 0.094 | 0.047 | 0.182 | 0.052 | 0.806 | 0.143 |
| | $f_\phi(q)$ | **0.102** | **0.077** | **0.037** | **0.013** | **0.000** | **0.004** | **0.000** | **0.109** |
| Qwen3-30B-A3B | $q$ | 0.223 | 0.617 | 0.148 | 0.055 | 0.101 | 0.024 | 0.836 | 0.155 |
| | $f_\phi(q)$ | **0.061** | **0.082** | **0.031** | **0.004** | **0.001** | **0.009** | **0.072** | **0.075** |

### E.3 Implementation

**Binning Estimator for TCE** We assume we have $m^{C-1}$ equally spaced bins on the simplex. For each prediction $p_i \in \Delta^{C-1}$, we assign a bin index $B_i$. For each occupied bin $B$, let

$$I_B = \{i : B_i = B\}.$$

The empirical conditional label distribution in bin $B$ is

$$\hat{q}_B = \frac{1}{|I_B|} \sum_{i \in I_B} \ell_i,$$

where $\ell_i$ is the one-hot encoding of the observed label as explained in Section 2. The binned TCE estimator is then

$$\widehat{\mathrm{TCE}} = \frac{1}{n} \sum_B \sum_{i \in I_B} D(p_i || \hat{q}_B).$$

The procedure is also shown in Algorithm 2.

**Algorithm 2** Binning estimator for Task Calibration Error (TCE)

**Input:** Predicted latent distributions $\{p_i\}_{i=1}^n$ with $p_i \in \Delta^{C-1}$, observed true latents $\{\ell_i^*\}_{i=1}^n$, number of bins per dimension $m$.

**Output:** Binned estimate $\widehat{\mathrm{TCE}}$

Partition the simplex $\Delta^{C-1}$ into $m^{C-1}$ equally volume bins
**for** $i \leftarrow 1$ to $n$ **do**
  Assign prediction $p_i$ to its bin $B_i$
**end for**

$\widehat{\mathrm{TCE}} \leftarrow 0$
**for** each occupied bin $B$ **do**
  Collect indices $I_B \leftarrow \{i : B_i = B\}$
  Estimate the true conditional distribution in bin $B$ $\quad \hat{q}_B \leftarrow \frac{1}{|I_B|} \sum_{i \in I_B} \ell_i^*$
  Accumulate the bin contribution $\quad \widehat{\mathrm{TCE}} \leftarrow \widehat{\mathrm{TCE}} + \sum_{i \in I_B} D_{S_T}(p_i \| \hat{q}_B)$
**end for**

$\widehat{\mathrm{TCE}} \leftarrow \frac{1}{n} \widehat{\mathrm{TCE}}$

# F PROOFS

**Proposition D.1.** *For any loss function $d_T : \mathcal{L} \times \mathcal{L} \mapsto \mathbb{R}$ among all decision strategies $\delta' : \Delta^{C-1} \mapsto \mathcal{L}$, the MBR decision $\delta^{\mathrm{MBR}}$ on $f^*_{g_T}(\hat{p}(\ell \mid \mathrm{x}))$ is optimal in terms of the expected incurred loss w.r.t the true distribution $p^*(\mathrm{x}, \mathrm{y})$ on any subset $P \subseteq \Delta^{C-1}$:*

$$\mathbb{E}_{\mathrm{x},\mathrm{y}}\left[d_T(g_T(\mathrm{y}), \delta^{\mathrm{MBR}}(f^*_{g_T}(\hat{p}(\ell \mid \mathrm{x})))) \mid \hat{p}(\mathrm{x}) \in P\right] \leq \mathbb{E}_{\mathrm{x},\mathrm{y}}\left[d_T(g_T(\mathrm{y}), \delta'(\hat{p}(\ell \mid \mathrm{x}))) \mid \hat{p}(\mathrm{x}) \in P\right]. \tag{9}$$

*Proof.* Define

$$\mathrm{s} := \hat{p}(\ell \mid \mathrm{x}), \qquad \mathrm{l} := g_T(Y).$$

Then s is the observable model output, and for any realization of s, $f^*_{g_T}(\mathrm{s})$ denotes the true conditional distribution of l given s.

Now for any $\delta'$:

$$\begin{aligned}
\mathbb{E}_{\mathrm{x},\mathrm{y}}[d_T(g_T(\mathrm{y}), \delta'(\hat{p}(\ell \mid \mathrm{x}))) \mid \hat{p}(\mathrm{x}) \in P] &= \mathbb{E}_{\mathrm{x},\mathrm{y}}[d_T(\mathrm{l}, \delta'(\mathrm{s})) \mid \mathrm{s} \in P] && (12)\\
&= \mathbb{E}_{\mathrm{s}|\mathrm{s}\in P}[\mathbb{E}_{\mathrm{x},\mathrm{y}}[d_T(\mathrm{l}, \delta'(\mathrm{s})) \mid \mathrm{s}, \mathrm{s} \in P]] && (13)\\
&= \mathbb{E}_{\mathrm{s}|\mathrm{s}\in P}\left[\mathbb{E}_{\ell \sim f^*_{g_T}(\mathrm{s})}[d_T(\ell, \delta'(\mathrm{s})) \mid \mathrm{s}, \mathrm{s} \in P]\right] && (14)\\
&\geq \mathbb{E}_{\mathrm{s}|\mathrm{s}\in P}\left[\min_{a \in \mathcal{L}} \mathbb{E}_{\ell \sim f^*_{g_T}(\mathrm{s})}[d_T(\ell, a) \mid \mathrm{s}, \mathrm{s} \in P]\right] && (15)\\
&= \mathbb{E}_{\mathrm{s}|\mathrm{s}\in P}\left[\mathbb{E}_{\ell \sim f^*_{g_T}(\mathrm{s})}\left[d_T\left(\ell, \delta^{\mathrm{MBR}}(f^*_{g_T}(\mathrm{s}))\right) \mid \mathrm{s}, \mathrm{s} \in P\right]\right] && (16)\\
&= \mathbb{E}_{\mathrm{s}|\mathrm{s}\in P}\left[\mathbb{E}_{\mathrm{x},\mathrm{y}}\left[d_T\left(\mathrm{l}, \delta^{\mathrm{MBR}}(f^*_{g_T}(\mathrm{s}))\right) \mid \mathrm{s}, \mathrm{s} \in P\right]\right] && (17)\\
&= \mathbb{E}_{\mathrm{x},\mathrm{y}}\left[d_T\left(\mathrm{l}, \delta^{\mathrm{MBR}}(f^*_{g_T}(\mathrm{s}))\right) \mid \mathrm{s} \in P\right]. && (18)
\end{aligned}$$

Substituting back $\mathrm{l} = g_T(\mathrm{y})$ and $\mathrm{s} = \hat{p}(\ell \mid \mathrm{x})$ yields

$$\mathbb{E}_{\mathrm{x},\mathrm{y}}\left[d_T\left(g_T(\mathrm{y}), \delta^{\mathrm{MBR}}(f^*_{g_T}(\hat{p}(\mathrm{x})))\right) \mid \hat{p}(\mathrm{x}) \in P\right] \leq \mathbb{E}_{\mathrm{x},\mathrm{y}}[d_T(g_T(\mathrm{y}), \delta'(\hat{p}(\mathrm{x}))) \mid \hat{p}(\ell \mid \mathrm{x}) \in P]. \tag{19}$$

In Equation (13) we apply the tower rule by conditioning on $\mathrm{s} = \hat{p}(\ell \mid \mathrm{x})$. In Equation (14) we use that, conditional on $S$, the random variable $\mathrm{l} = g_T(\mathrm{y})$ is distributed according to $f^*_{g_T}(\mathrm{s})$ when taking the expectation over x, y with respect to the true data generation distribution. In Equation (15) we use that, for each realization of $S$, the conditional risk of the action $\delta'(\mathrm{s})$ is at least the minimum

achievable conditional risk over all actions $a \in \mathcal{L}$. In Equation (16) we use the definition of the Minimum Bayes Risk decision rule. Finally, Equation (17) uses again the tower rule. Hence, no other decision strategy depending only on $\hat{p}(\ell \mid \mathrm{x})$ can attain lower expected task loss for any condition on $\hat{p}(\ell \mid \mathrm{x})$. □

**Proposition F.1.** *For a given parametric calibration map family $\mathcal{F}$, the map with minimal negative log-likelihood minimizes the expected KL-divergence to $f^*_{g_T}$:*

$$\arg\min_{f_\phi \in \mathcal{F}} \mathbb{E}_{\mathrm{x,y}} \left[ -g_T(\mathrm{y})^T \log f_\phi(\hat{p}(\mathrm{x})) \right] = \arg\min_{f_\phi \in \mathcal{F}} \mathbb{E}_{\mathrm{x}} \left[ \mathrm{KL} \left[ f^*_{g_T}(\hat{p}(\mathrm{x}) \| f_\phi(\hat{p}(\mathrm{x})) \right] \right]. \tag{20}$$

*Proof.*

$$\begin{aligned}
\mathbb{E}_{\mathrm{x,y}} \left[ -g_T(\mathrm{y}^T) \log f_\phi(\hat{p}(\mathrm{x}) \right] &= \mathbb{E}_{\hat{p}(\mathrm{x})} \left[ \mathbb{E}_{\mathrm{x,y}} \left[ -g_T(\mathrm{y})^T \log f_\phi(\hat{p}(\mathrm{x})) \right] \mid \hat{p}(\mathrm{x}) \right] && (21)\\
&= \mathbb{E}_{\hat{p}(\mathrm{x})} \left[ -\mathbb{E}_{\mathrm{x,y}}[g_T(\mathrm{y}) \mid \hat{p}(\mathrm{x})]^T \log f_\phi(\hat{p}(\mathrm{x})) \right] && (22)\\
&= \mathbb{E}_{\hat{p}(\mathrm{x})} \left[ \mathbb{E}_{\mathrm{x}|\hat{p}(\mathrm{x})} \left[ -(f^*_{g_T}(\hat{p}(\mathrm{x})))^T \log f_\phi(\hat{p}(\mathrm{x})) \right] \right] && (23)\\
&= \mathbb{E}_{\hat{p}(\mathrm{x})} \left[ \mathbb{E}_{\mathrm{x}|\hat{p}(\mathrm{x})} \left[ -(f^*_{g_T}(\hat{p}(\mathrm{x})))^T \log f_\phi(\hat{p}(\mathrm{x})) \right] \right] \\
&\quad + \mathbb{E}_{\hat{p}(\mathrm{x})} \left[ \mathbb{E}_{\mathrm{x}|\hat{p}(\mathrm{x})} \left[ f^*_{g_T}(\hat{p}(\mathrm{x}))^T \log f^*_{g_T}(\hat{p}(\mathrm{x})) \right] \right] && (24)\\
&\quad - \mathbb{E}_{\hat{p}(\mathrm{x})} \left[ \mathbb{E}_{\mathrm{x}|\hat{p}(\mathrm{x})} \left[ f^*_{g_T}(\hat{p}(\mathrm{x}))^T \log f^*_{g_T}(\hat{p}(\mathrm{x})) \right] \right] \\
&= \mathbb{E}_{\hat{p}(\mathrm{x})} \left[ \mathbb{E}_{\mathrm{x}|\hat{p}(\mathrm{x})} \left[ (f^*_{g_T}(\hat{p}(\mathrm{x})))^T \log \frac{f^*_{g_T}(\hat{p}(\mathrm{x}))}{f_\phi(\hat{p}(\mathrm{x}))} \right] \right] && (25)\\
&\quad - \mathbb{E}_{\hat{p}(\mathrm{x})} \left[ \mathbb{E}_{\mathrm{x}|\hat{p}(\mathrm{x})} \left[ f^*_{g_T}(\hat{p}(\mathrm{x}))^T \log f^*_{g_T}(\hat{p}(\mathrm{x})) \right] \right] \\
&= \mathbb{E}_{\hat{p}(\mathrm{x})} \left[ \mathbb{E}_{\mathrm{x}|\hat{p}(\mathrm{x})} \left[ \mathrm{KL} \left[ f^*_{g_T}(\hat{p}(\mathrm{x})) \| f_\phi(\hat{p}(\mathrm{x})) \right] \right] \right] && (26)\\
&\quad + \mathbb{E}_{\hat{p}(\mathrm{x})} \left[ \mathbb{E}_{\mathrm{x}|\hat{p}(\mathrm{x})} \left[ \mathbb{H} \left[ f^*_{g_T}(\hat{p}(\mathrm{x})) \right] \right] \right] \\
&= \mathbb{E}_{\mathrm{x}} \left[ \mathrm{KL} \left[ f^*_{g_T}(\hat{p}(\mathrm{x})) \| f_\phi(\hat{p}(\mathrm{x})) \right] \right] && (27)\\
&\quad + \mathbb{E}_{\mathrm{x}} \left[ \mathbb{H} \left[ f^*_{g_T}(\hat{p}(\mathrm{x})) \right] \right]
\end{aligned}$$

The second expected entropy term does not depend on $f_\phi(\hat{p}(\mathrm{x}))$ and hence minimizing the cross entropy also minimizes the expected KL divergence. □

**Proposition F.2.** *The generalized entropy & divergence induced by $S_T(q, \ell) = d_T(\ell, \delta^{MBR}(q))$ are:*

$$H_{S_T}(q) = R_{\mathrm{Bayes}}(\delta^{MBR}(q), q), \quad D_{S_T}(q \,\|\, r) = R_{\mathrm{Bayes}}(\delta^{MBR}(q), r) - R_{\mathrm{Bayes}}(\delta^{MBR}(r), r). \tag{28}$$

*Proof.* We first recall the definitions of the generalized entropy and divergence induced by a scoring function $S_T$:

$$\begin{aligned}
H_{S_T}(q) &= \mathbb{E}_{\ell \sim q}[S_T(q, \ell)], && (29)\\
D_{S_T}(q \| r) &= \mathbb{E}_{\ell \sim r}[S_T(q, \ell)] - \mathbb{E}_{\ell \sim r}[S_T(r, \ell)]. && (30)
\end{aligned}$$

Using $S_T(q, \ell) = d_T(\ell, \delta^{\mathrm{MBR}}(q))$, we obtain

$$\begin{aligned}
H_{S_T}(q) &= \mathbb{E}_{\ell \sim q}[S_T(q, \ell)] && (31)\\
&= \mathbb{E}_{\ell \sim q}\big[d_T(\ell, \delta^{\mathrm{MBR}}(q))\big] && (32)\\
&= R_{\mathrm{Bayes}}(\delta^{\mathrm{MBR}}(q), q). && (33)
\end{aligned}$$

Likewise,

$$\begin{aligned}
D_{S_T}(q \| r) &= \mathbb{E}_{\ell \sim r}[S_T(q, \ell)] - \mathbb{E}_{\ell \sim r}[S_T(r, \ell)] && (34)\\
&= \mathbb{E}_{\ell \sim r}\big[d_T(\ell, \delta^{\mathrm{MBR}}(q))\big] - \mathbb{E}_{\ell \sim r}\big[d_T(\ell, \delta^{\mathrm{MBR}}(q))\big] && (35)\\
&= R_{\mathrm{Bayes}}(\delta^{\mathrm{MBR}}(q), r) - R_{\mathrm{Bayes}}(\delta^{\mathrm{MBR}}(r), r). && (36)
\end{aligned}$$

□

**Proposition E.1.** *We define the Task Calibration Error (TCE) by decomposing the $D_{S_T}$ between the predicted latent distribution $\hat{p}(\ell \mid \mathrm{x})$ and true latent distribution $p^*(\ell|\mathrm{x})$ as:*

$$\mathbb{E}_X\left[D_{S_T}\left(\hat{p}(\ell \mid \mathrm{x}) \,\|\, p^*(\ell \mid \mathrm{x})\right)\right] = \underbrace{\mathbb{E}_X\left[D_{S_T}\left(\hat{p}(\ell \mid \mathrm{x}) \,\|\, f^*_{g_T}(\hat{p}(\ell \mid \mathrm{x}))\right)\right]}_{\textit{Task Calibration Error (TCE)}} + \mathbb{E}_X\left[D_{S_T}\left(f^*_{g_T}(\hat{p}(\ell \mid \mathrm{x})) \,\|\, p^*(\ell \mid \mathrm{x})\right)\right]. \tag{11}$$

*Proof.* Follows directly from the known decomposition Kull and Flach (2015). □

## G PROMPTS

Prompt 1: $g_T$ Prompt for Set-Case

```
You extract explicit, semantically distinct atomic answers from a model
    answer.

Rules:
- Extract only what the answer explicitly states; no inference.
- Split coordinated lists (commas, and/or, slashes) into separate items.
- Merge synonyms/rephrasings; keep one canonical surface form, preserving
     key head nouns.
- Remove leading articles/hedges; remove trailing punctuation/
    parentheticals unless part of the name.
- Ignore explanations/justifications; include quantities only if the
    quantity itself is the item.
- If no explicit item appears (I don't know, etc.), output: I don't know.
- Output strictly: one item per line, no bullets or extra text.

Follow the output format in the examples.

Example 1
Q: Which oceans border the USA?
A: The Atlantic and Pacific Oceans.
Output:
Atlantic Ocean
Pacific Ocean

Example 2
Q: What are the official languages of Switzerland?
A: German, French, Italian, and Romansh are the four official languages.
Output:
German
French
Italian
Romansh



Q: {question}
A: {answer}

Task: List all explicit, distinct answers from the model answer, one per
    line.
```

Prompt 2: Prompt for When2Call

```
You are an expert in composing functions. You are given a question and a
    set of possible functions.
Based on the question, decide what the assistant should do.
If a function can directly answer the question, you may call it.
If the question lacks required parameters, ask a follow-up question.
If none of the functions can be used to answer the question, say so.
```

```
If you decide to invoke a function, you MUST put it in the format of [
    func_name1(params_name1=params_value1, params_name2=params_value2...)
    , func_name2(params)].

You may briefly explain your reasoning first, but your final line must be
     exactly one of:
FINAL_ACTION: tool_call
FINAL_ACTION: request_for_info
FINAL_ACTION: cannot_answer

Here is a list of functions in JSON format that you can invoke.
{tools}

Question: {question}
```

Prompt 3: Prompt for MMLU

```
What is the correct answer to the following multiple choice question? You
     may very briefly explain your reasoning first but then give your
    final answer as one of the letters as "Final answer: letter".
Question: <question>
A. <choice 1>
B. <choice 2>
C. <choice 3>
D. <choice 4>
```

Prompt 4: Prompt for HelpSteer

```
Assess the following model response for {criterion_name} ({
    criterion_description})
on a scale from 0 to 4, where higher is better.

You may briefly explain your reasoning first, but your final line must be
     exactly one
number from 0 to 4 in the format: "Final answer: <number>".

Model response: {question}
```

Prompt 5: HelpSteer prompt, `concise` variant

```
Rate the {criterion_name} ({criterion_description}) of the model response
     below
from 0 to 4, where higher is better.

You may briefly explain your reasoning first, but your final line must be
     exactly one
number from 0 to 4 in the format: "Final answer: <number>".

Model response: {question}
```

Prompt 6: HelpSteer prompt, `rubric` variant

```
Using a 0-4 {criterion_name} scale ({criterion_description}), with 0
    indicating that
the response does not satisfy the criterion and 4 indicating that it
    fully satisfies
the criterion, assess the model response below.

You may briefly explain your reasoning first, but your final line must be
     exactly one
number from 0 to 4 in the format: "Final answer: <number>".

Model response: {question}
```

Prompt 7: HelpSteer prompt, `evaluator` variant

```
You are an impartial evaluator. Judge the model response for {
    criterion_name}
({criterion_description}) and assign a score from 0 to 4; higher scores
    indicate
greater {criterion_name}.

You may briefly explain your reasoning first, but your final line must be
     exactly one
number from 0 to 4 in the format: "Final answer: <number>".

Model response: {question}
```

Prompt 8: Prompt for STSB

```
Measure the sentence similarity between two sentences measured on a
    continous scale of 0
to 5 with 0 being unrelated and 5 being related.
Now for the following two sentences, what is the similarity score?

You may briefly explain your reasoning first, but your final line must be
     exactly one
number from 0 to 5 in the format: "Final answer: <number>".

Sentences: {question}
```

Prompt 9: Prompt for TriviaQA/SimpleQA/MAQA

```
{question}. Only provide the answer without explanation.
```

Prompt 10: Prompt for Helpsteer, Two Criteria

```
Assess the following model response for {criteria_names_and_descriptions}
     on a scale
from 0 to 4 each, where higher is better.

You may briefly explain your reasoning first, but your final line must be
     exactly one
number from 0 to 4 for each criterion in the format: "Final answer: {
    format_string}".

Model response: {question}
```

Prompt 11: Prompt for Extracting Integers from LLM Response

```
You extract the final numeric answer from a model response.

Rules:
- Read the full response and identify the final numeric value the answer
    commits to.
- Ignore intermediate calculations, option numbers, and numbers mentioned
     only in reasoning.
- Return exactly one bare number with no explanation or surrounding text.
- If the response does not clearly provide one final numeric answer,
    return UNKNOWN.

Question: {question}
Answer: {answer}

Task: Return only the final numeric answer stated in the answer.
```

## H ADDITIONAL FIGURES

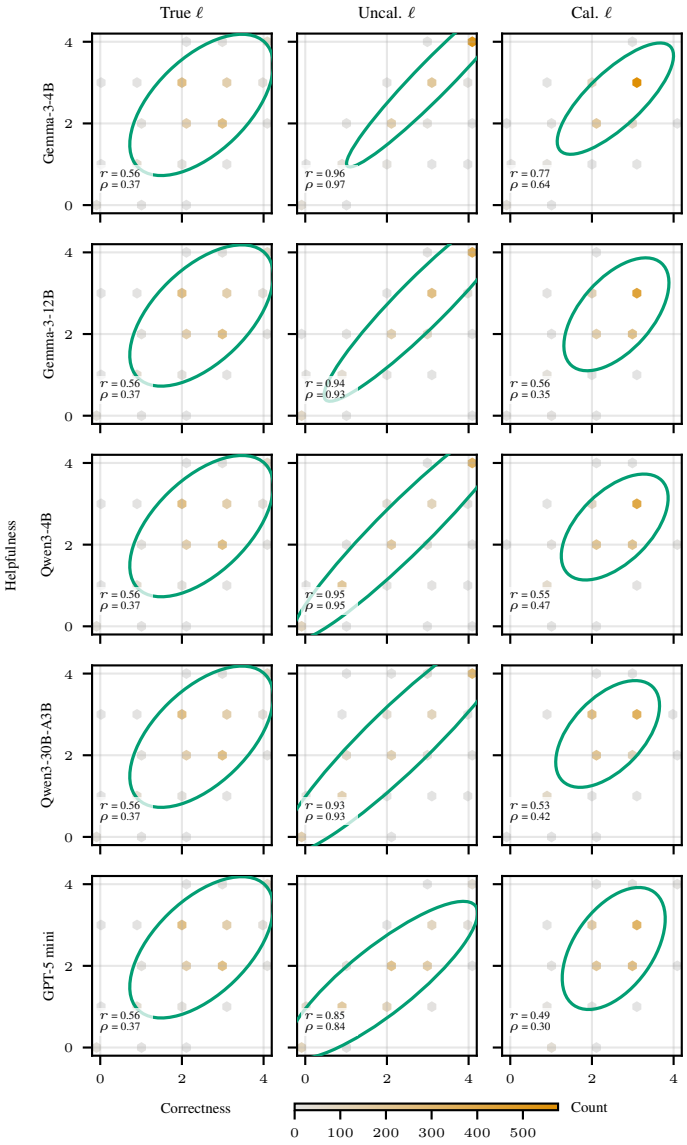



Figure 10: **Calibration corrects the spurious dependence between criteria.** Joint distribution of the two vector-valued HelpSteer criteria per model (rows): human labels, uncalibrated decision, calibrated decision. Ellipses are $2\sigma$ covariance ellipses. Every model treats the criteria as far more dependent than they are ($r = 0.85$–$0.96$ vs. a true $0.56$); calibration pulls each back.

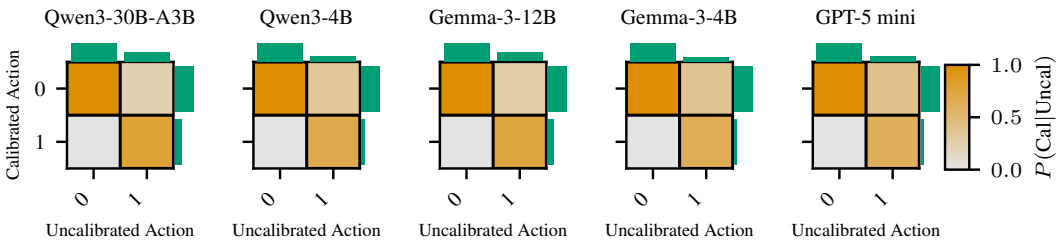
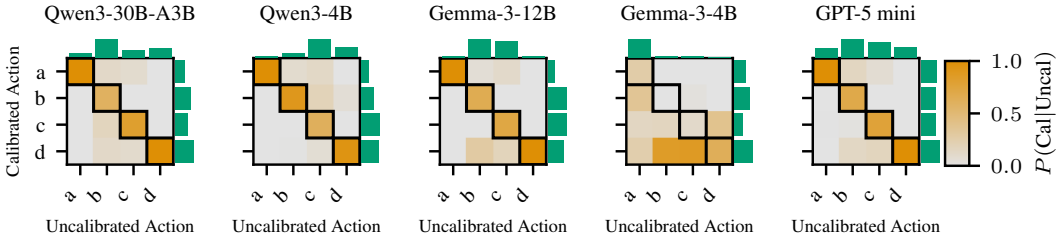
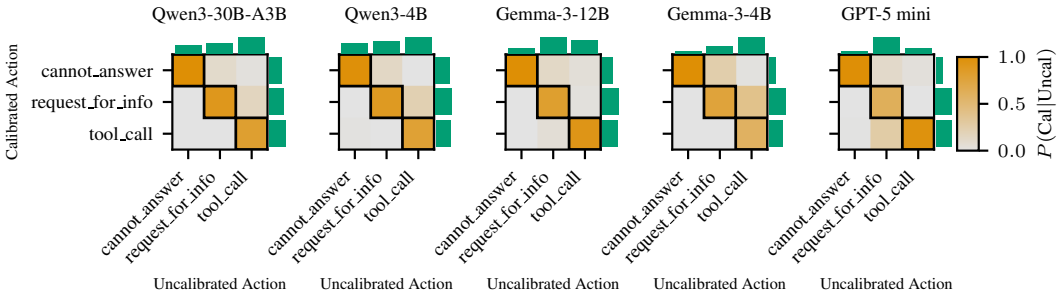



Figure 11: Action movement for MAQA ($\{0, 1\}^{\leq C}$).

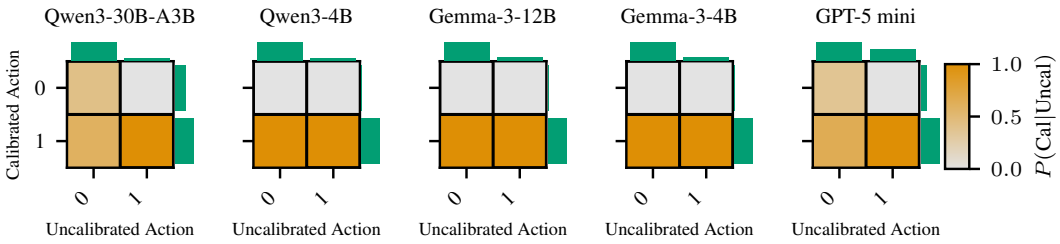



(12a) MMLU ($[C]$)

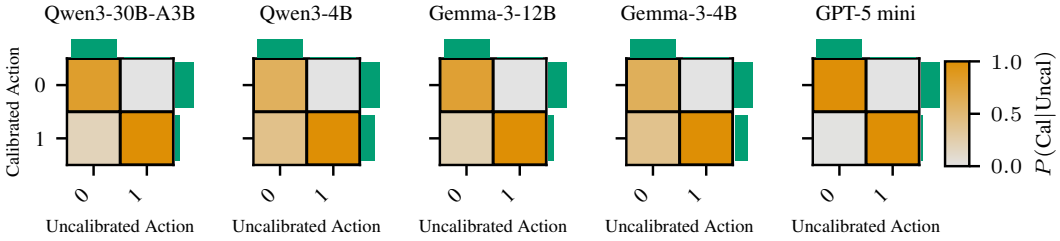



(12b) When2Call ($[C]$)

Figure 12: Action movement for classification tasks.

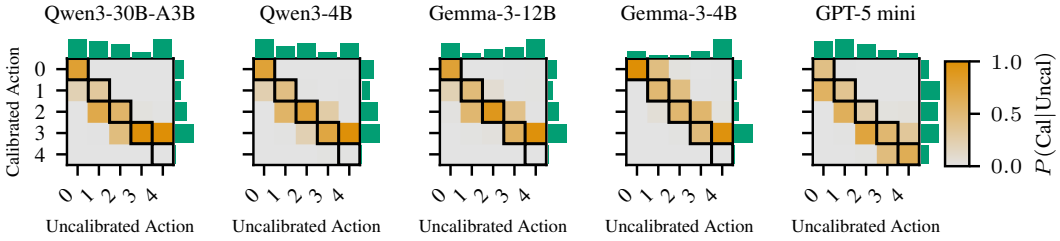



(13a) SimpleQA ($\{A, \perp\}$)

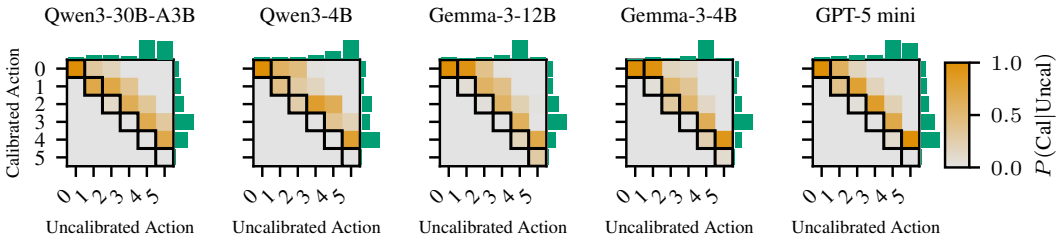



(13b) TriviaQA ($\{A, \perp\}$)

Figure 13: Action movement for abstention tasks.




(14a) HelpSteer ($\{0, \ldots, C\}$)




(14b) STSB ($\{0, \ldots, C\}$)

Figure 14: Action movement for ordinal/interval tasks.

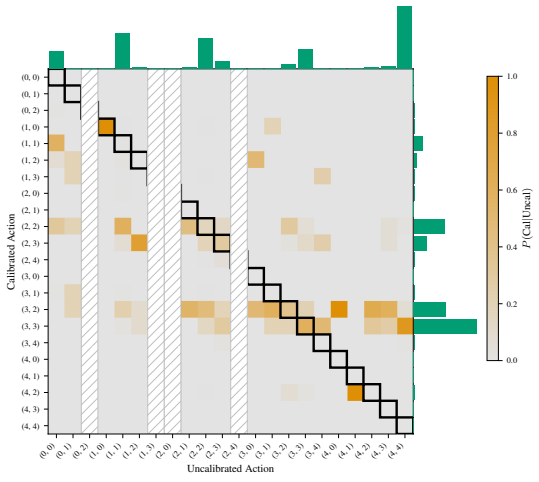



Figure 15: HelpSteer ($\{0, \ldots, C\}^2$) Qwen3-4B

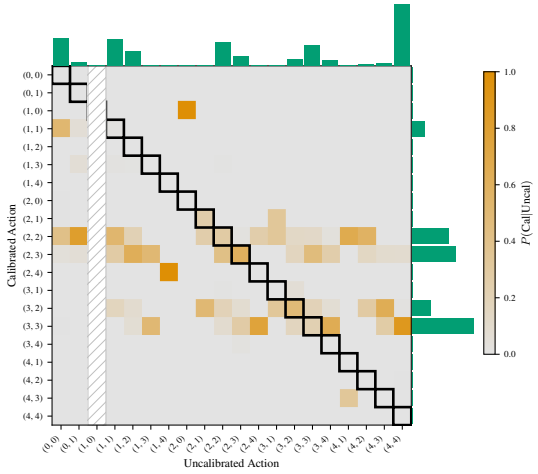



Figure 16: HelpSteer ($\{0, \ldots, C\}^2$) Qwen3-30B-A3B

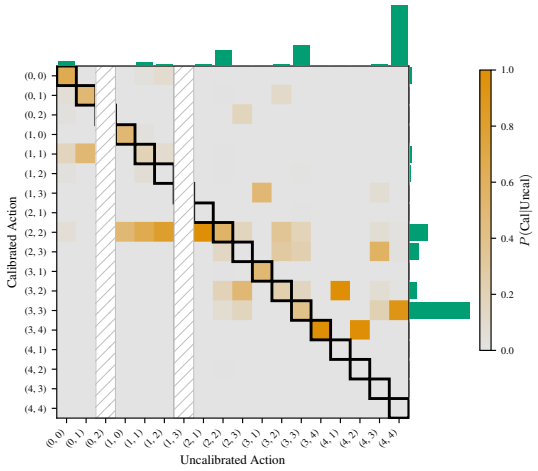



Figure 17: HelpSteer ($\{0, \ldots, C\}^2$) Gemma3-4B

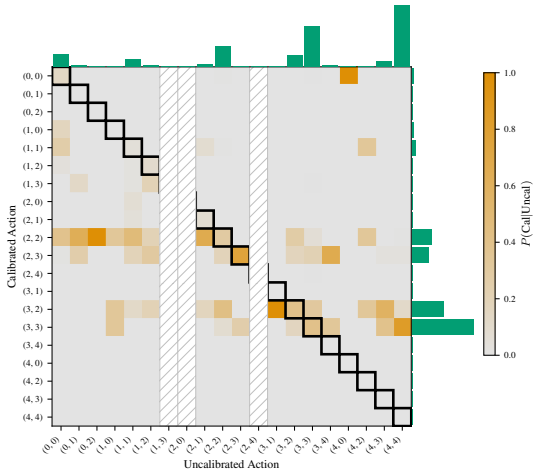



Figure 18: HelpSteer ($\{0, \ldots, C\}^2$) Gemma-3-12B

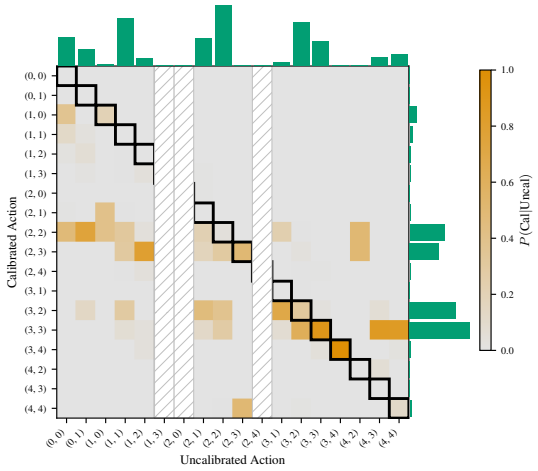



Figure 19: HelpSteer ($\{0, \ldots, C\}^2$) GPT-5 mini