Towards Universal Calibration for In-Context Learning

Anonymous ACL submission

Abstract

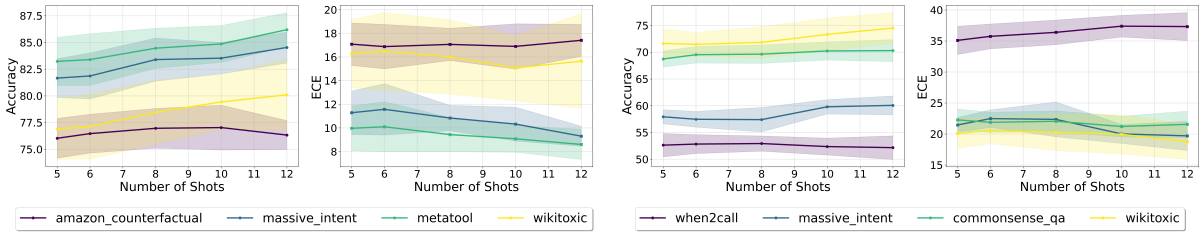
In-Context Learning (ICL) has received a lot of interest recently due to its ability to improve classification accuracy for new tasks without having to fine-tune the model over these tasks. However, the impact of ICL on calibration has been far less studied. Recent work has shown that ICL can have a negative impact on calibration due to biased selection of in-context examples. The existing methods for improving ICL calibration require additional task and model-specific training, detracting from the no-fine-tuning convenience of ICL. In this work, we introduce CALIBFORMER, a *universal* ICL calibrator that generalizes to new models and tasks without further fine-tuning on these tasks. Specifically, we design a transformer-based calibrator which uses the model predictions on in-context examples to predict a confidence score for the test input. We show that the calibrator trained on three models and ten tasks, is able to generalize/calibrate ICL predictions on several new tasks and four LLMs unseen during training. Even for frontier models like Gemini, CALIBFORMER is able to obtain over 50% reduction in calibration error. Anonymised repo

1 Introduction

In-Context Learning (ICL) allows large language models (LLMs) to adapt to new classification tasks at inference time by conditioning on a small set of labeled demonstrations, without any parameter updates (Brown et al., 2020). This capability has found wide application in settings where labeled data is scarce, including question answering (Wu et al., 2023), low-resource NLP (Alves et al., 2023; Li et al., 2025), tool calling (Zhang et al., 2025; Dang et al., 2025), and risk assessment (Padayachy et al., 2026). However, for ICL to be deployed reliably the model's predicted probabilities must be *calibrated*: a prediction made with 90% confidence should be correct 90% of the time. A well-calibrated predictor enables trust in downstream decision-making as the predicted probabilities provide a good measure of the *confidence* in prediction. For instance, in risk assessment or sequential reasoning, if the system identifies that it is not confident in its prediction, then it can delegate control to a human expert or invoke external verification tools. On the other hand, a miscalibrated predictor can be detrimental to reliable and safe decision-making, especially in autonomous or semi-autonomous systems. Overconfidence in "agentic" loops can propagate errors to downstream tasks, leading to compounding failures in autonomous workflows.

Unfortunately, ICL often degrades calibration. The very mechanism that makes ICL effective—conditioning on a small, context-dependent set of demonstrations—introduces variance in the model's confidence that does not track its actual accuracy. As illustrated in Figure 1, while accuracy generally improves with additional demonstrations, calibration error behaves erratically, sometimes worsening as more examples are added.

Related Work. Existing approaches to improving ICL calibration each carry significant limitations. Bias-correction methods (Zhao et al., 2021; Zhou et al., 2024; Jang et al., 2024) subtract an estimated label prior from the logits, but they target accuracy rather than calibration, assume randomly sampled demonstrations, and require task-specific estimation. Temperature scaling can be fitted on the in-context examples themselves, but treats predictions at different context positions as independent whereas predictions on ICL examples are correlated. Methods that train auxiliary calibrators (Ulmer et al., 2024; Shen et al., 2024) require retraining for each target LLM, and approaches based on hidden-state features (Shen et al., 2024) additionally cannot be applied to black-box models. In all cases, the user must invest effort per model/task, undermining the convenience that makes ICL attractive. See Appendix C for more related work.



(a) Similarity sampling

(b) Random sampling

Figure 1: Accuracy and ECE with varying number of shots. While accuracy typically increases with the number of shots, the model does not significantly improve in aligning its confidence with true likelihood of correctness. Sometimes unpredictable trends are observed - wikitoxic ECE in (a) is turbulent at the end; when2call in (b) experiences worsening ECE despite stable accuracy

In this work, we ask: *can a single calibrator, trained once, improve the calibration of any LLM on any classification task, using only the in-context examples?* Towards that goal we develop CALIBFORMER, a lightweight (0.61M parameter) transformer-based meta-model that predicts the probability that an LLM's prediction is correct, serving as a calibrated confidence score.

The key idea is that in-context examples constitute a *latent validation set*. As the LLM processes the demonstrations autoregressively, its predictions on these examples reveal the extent and pattern of its miscalibration in the current prompt context. CALIBFORMER is a causal transformer that attends over a compact set of task-agnostic features derived from these prediction–label discrepancies (top-class probabilities, running correctness, ground-truth probability at each position) and learns to output a calibrated confidence for the test prediction. Because these features capture only the *statistical signature* of the LLM's behavior rather than any task-specific or model-specific representation, a single calibrator trained on 3 open-weight LLMs and 10 classification tasks generalizes to unseen models and tasks without further fine-tuning.

We evaluate CALIBFORMER on eight held-out classification tasks spanning tool use, toxicity detection, intent classification, and medical question answering, using four LLMs not seen during training—including the black-box frontier model Gemini-2.5-flash. Our main findings are:

1. CALIBFORMER reduces average ECE by 41–73% relative to uncalibrated ICL (e.g., from 18.3% to 6.1% on Phi-4-mini under similarity sampling; from 12.2% to 4.5% on Gemini-2.5-flash under random sampling).
2. CALIBFORMER consistently outperforms strong baselines including in-context temperature scaling and permutation averaging, while requiring only a single forward pass through a sub-million-parameter model.
3. The calibrator transfers to LLMs with substantially different architectures and confidence profiles, including models that are already reasonably well-calibrated (Qwen-3.5-9B) and black-box models where only token-level log-probabilities are available (Gemini-2.5-flash).

2 Problem Formulation

We study the calibration of LLM predictions on text classification tasks defined through in-context learning. A task τ is specified by k input–output pairs $\mathcal{C}_k = \{(x_i, y_i)\}_{i=1}^{k}$, where each x_i is a text string and $y_i \in \mathcal{Y}_\tau$ is a class label drawn from a task-specific label space. We assume these examples are sampled from a latent distribution $P_\tau(X, Y)$, with $P_\tau(y \mid x)$ denoting the (unknown) true conditional label distribution.

Given the k demonstrations and a test input x^*, an LLM $\mathcal{M}$ produces a predictive distribution $P_\mathcal{M}(y \mid x^*, \mathcal{C}_k)$ over $\mathcal{Y}_\tau$. Section 3.5 describes how we extract such a distribution from a generative LLM for an arbitrary label set. We write $\hat{y} = \text{argmax}_y P_\mathcal{M}(y \mid x^*, \mathcal{C}_k)$ for the predicted label and refer to the corresponding probability $P_\mathcal{M}(\hat{y} \mid x^*, \mathcal{C}_k)$ as the model's *confidence*.

A model is *well-calibrated* when its confidence matches the true likelihood of correctness: among all predictions assigned confidence p, a fraction p should be correct. In practice, modern LLMs tend to be *overconfident*—for instance, among predictions with confidence above 0.9, substantially fewer than 90% may be correct.

Our goal is to transform the predicted distribution $P_\mathcal{M}(\cdot \mid \mathcal{C}_k, x^*)$ into a better-calibrated distribution $\tilde{P}_\mathcal{M}(\cdot \mid \mathcal{C}_k, x^*)$, using *only* in-context examples $\mathcal{C}_k$ and model's own outputs—without access to additional labeled data from the target task.

3 Universal Calibrator for ICL

We propose CALIBFORMER, a lightweight transformer-based calibrator that predicts, for any LLM and any classification task, the probability that the LLM's in-context prediction on a test input is correct. This predicted probability serves as the calibrated confidence score. The central design principle is that CALIBFORMER operates exclusively on *task-agnostic features* derived from the LLM's own output probabilities and the ground-truth labels of the in-context examples, and requires no access to hidden states, attention patterns, or any task-specific metadata. This makes it applicable even to black-box LLMs that expose only token-level log-probabilities.

The key insight underlying our approach is that the in-context examples can be repurposed as a *latent validation set*. As the LLM processes the demonstration sequence autoregressively, its predictions on the in-context examples with known true labels reveal how well-calibrated (or miscalibrated) the model is in the current prompt context. CALIBFORMER is a causal transformer that attends over this sequence of prediction–label discrepancies and learns to correct the confidence on the subsequent test prediction accordingly. Because the features capture only the *statistical signature* of the LLM's behavior (prediction confidence, ranking among classes, running correctness), a single calibrator trained on a modest collection of models and datasets generalizes to unseen tasks and LLMs without further fine-tuning.

To train such a calibrator, we leverage a diverse pool of publicly available text classification datasets $\{D_1, \ldots, D_n\}$ spanning tasks such as natural language inference, sentiment analysis, topic classification, and toxicity detection, paired with a set of open-weight LLMs $\{\mathcal{M}_1, \ldots, \mathcal{M}_m\}$ that differ in architecture, scale, and confidence profile. This diversity exposes the calibrator to a broad range of miscalibration patterns during training. We describe the feature representation (§3.1), architecture (§3.2), training procedure (§3.4), and the mechanism for extracting label distributions from generative LLMs (§3.5).

3.1 Features

Since CALIBFORMER is designed to be agnostic to both the LLM being calibrated and the target task we restrict its inputs to features that can be computed from any model's output probabilities and the in-context labels alone. In particular, we avoid features that presuppose access to internal representations (hidden states, attention maps) or prior knowledge of the task's difficulty or domain.

Recall that, at each position $t \in \{1, \ldots, k{+}1\}$, the LLM has access to the demonstrations $\mathcal{C}_{t-1} = \{(x_i, y_i)\}_{i=1}^{t-1}$ and predicts a distribution $P_{\mathcal{M}}(y \mid x_t, \mathcal{C}_{t-1})$ over the label space $\mathcal{Y}_\tau$. From this distribution and the ground-truth label y_{t-1} at the preceding position, we construct a feature vector $\mathbf{f}_t \in \mathbb{R}^d$ with $d = 5$:

1. **Top-1 predicted probability.** The confidence assigned to the most likely class $p_t^{(1)} = P_{\mathcal{M}}(y{=}\hat{y}_t \mid x_t, \mathcal{C}_{t-1})$.
2. **Top-2 predicted probability.** The probability of the second most likely class: $p_t^{(2)} = \max_{j \neq \hat{y}_t} P_{\mathcal{M}}(y{=}j \mid x_t, \mathcal{C}_{t-1})$.
   Together, $p_t^{(1)}$ and $p_t^{(2)}$ summarize how peaked or uncertain LLM's prediction is at position t.
3. **Previous correctness indicator.** A binary signal indicating whether the LLM's prediction at the preceding position matched the ground-truth label. This is set to 0.5 at $t = 1$. $r_t = \mathbb{1}[\arg\max_j P_{\mathcal{M}}(y{=}j \mid x_{t-1}, \mathcal{C}_{t-2}) = y_{t-1}]$
4. **Previous ground-truth probability.** The probability that the LLM assigned to the true label at the preceding position: $g_t = P_{\mathcal{M}}(y{=}y_{t-1} \mid x_{t-1}, \mathcal{C}_{t-2})$, with dummy value of 0.5 at $t = 1$.
5. **Previous ground-truth squared loss.** The squared error on the true-label probability at the preceding position: $\ell_t = (1 - g_t)^2$, set to 0.25 at $t = 1$ for consistency.
   Features 3–5 collectively provide a local measure of how well the LLM is performing on the previous example and provides the signal to judge whether the model's confidence at the current position should be trusted.

The full feature vector at each position is the concatenation $\mathbf{f}_t = [p_t^{(1)},\ p_t^{(2)},\ r_t,\ g_t,\ \ell_t]$. Note that all five components are bounded in $[0, 1]$, and are invariant to the size of the label space properties, that promote cross-task generalization.

3.2 CALIBFORMER Architecture

CALIBFORMER is a causal transformer that maps the feature sequence $\mathbf{F} = (\mathbf{f}_1, \ldots, \mathbf{f}_{k+1}) \in \mathbb{R}^{(k+1)\times d}$ to a calibrated confidence at each position. It consists of a linear input projection, $L{=}12$ transformer layers with hidden dimension $h{=}64$ and $A{=}8$ attention heads, and a scalar sigmoid head—0.61M parameters in total.

The causal attention mask makes the output at position t depend only on $(\mathbf{f}_1, \ldots, \mathbf{f}_t)$, so the calibrator observes ground-truth labels of preceding demonstrations but never the label at position t itself, mirroring the LLM's own autoregressive structure. At the test position $t{=}k{+}1$ it attends to the full demonstration sequence, using cumulative prediction–label discrepancies to adjust the test confidence. We use sinusoidal positional encodings to avoid position-specific biases and allow generalization to unseen shot counts.

The final linear+sigmoid head of CALIBFORMER produces $\hat{c}_t = \sigma(\mathrm{TF}_\phi(\mathbf{F})_t)$ where $\hat{c}_t \in (0, 1)$ is the predicted probability that the LLM's prediction at position t is correct, σ is the sigmoid function and ϕ are calibrator's parameters.

From confidence to a calibrated distribution. Using $\hat{c}_{k+1}$ as the LLM's calibrated confidence for the test input, we derive a calibrated distribution over all classes by finding the appropriate temperature $T > 0$ to adjusts the LLM's original logit vector $\mathbf{z}_{k+1}$ such that: $\max_{j \in \mathcal{Y}_\tau} \ \mathrm{softmax}(T \cdot \mathbf{z}_{k+1})_j = \hat{c}_{k+1}$. The unique T satisfying this equation is found efficiently by binary search, and the resulting tempered softmax yields calibrated probabilities over all classes. Note that this is the reverse of standard temperature scaling: rather than learning a temperature to produce well-calibrated confidences, we predict the calibrated confidence directly and back out the temperature needed to achieve it. Section 4 presents ablations that show that training the transformer to output confidence yielded much better results than training to output temperature.

3.3 Characterizing Task Coverage

To understand why our features can generalize to unseen model-task combinations, we visualize how well the training distribution covers the space of unseen tasks. For each LLM-dataset combination, we sample batches of trajectories from the calibrator training data where each trajectory represents autoregressive predictions over the sequence of inputs (Section 3.4). We represent each batch by its shot-wise mean confidence and mean accuracy—a $2T$-dimensional vector for a sequence of T inputs—summarizing the miscalibration profile of that LLM-dataset pair. We fit a UMAP jointly on training and unseen embeddings: if unseen points fall within the training region, coverage is adequate. Figure 2 shows that scaling from 2 LLMs $\times$ 5 datasets to 3 LLMs $\times$ 10 datasets substantially improves coverage, suggesting our features capture task-agnostic miscalibration properties that transfer across LLMs and datasets.

3.4 Training Procedure

We partition datasets into three disjoint groups—training, validation, evaluation. The training mix spans diverse classification tasks such as NLI, sentiment, toxicity, topic classification, and others; the evaluation set is held out from both training and checkpoint selection, providing an unbiased generalization test. Each training dataset is further split 80/20 into calibrator-training and calibrator-validation portions.

Training data generation. For each dataset D and model $\mathcal{M}$ pair, we sample a test instance $(x_{\text{test}}, y_{\text{test}})$ and build a k-shot prompt ($k{=}20$) with demonstrations drawn either uniformly at random or by cosine similarity to x^*; similarity-sampled demonstrations are randomly shuffled to add trajectory diversity. The prompt is processed autoregressively through $\mathcal{M}$, yielding the per-position predictive distribution and the feature sequence $(\mathbf{f}_1, \ldots, \mathbf{f}_{k+1})$ (§3.1). We further augment each trajectory as described in Section 3.4.1. Repeating across all training datasets and source LLMs produces a large pool that varies in label-space size, task difficulty, and model confidence profile.

Training objective. We train with binary cross-entropy over predicted correctness:

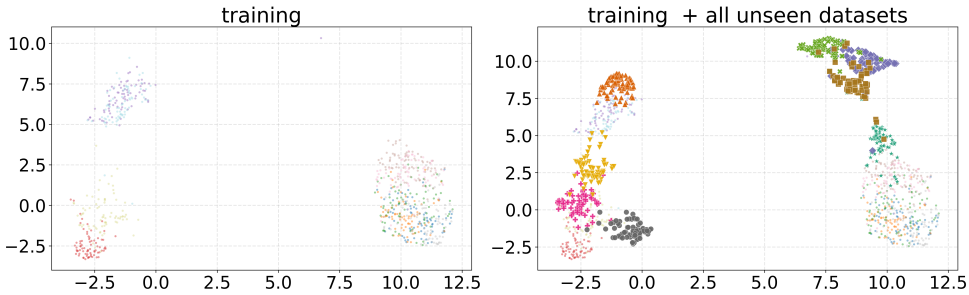
$$\mathcal{L}(\phi) = \sum_{d \in \mathcal{D}} \sum_{t=t_1}^{k+1} -a_t \log \hat{c}_t - (1 - a_t) \log(1 - \hat{c}_t),$$

where $\hat{c}_t = \sigma(\mathrm{TF}_\phi(\mathbf{F})_t)$ and $a_t \in \{0, 1\}$ indicates whether the LLM's prediction at position t is correct. The loss is applied only from $t_1{=}6$ onward, forcing the calibrator to rely on in-context evidence—the pattern of preceding successes and failures—rather than memorizing task priors in its weights. Without this, the model learns shortcuts that generalize poorly.

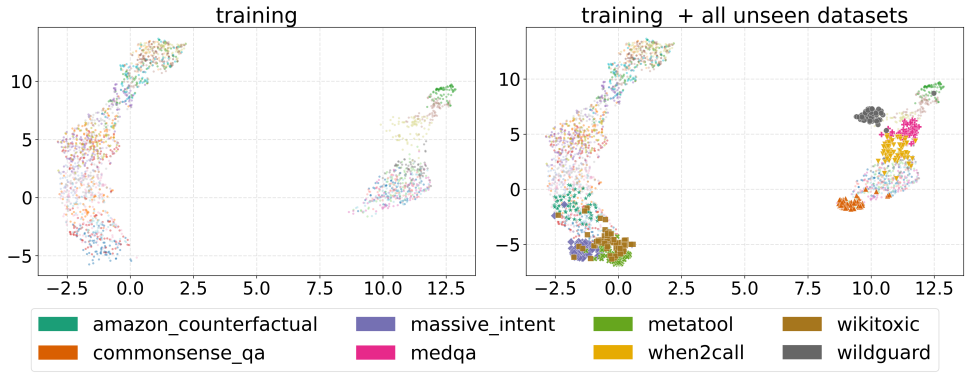
3.4.1 Data Augmentation

The training LLMs exhibit similar overconfidence patterns, limiting the diversity of calibration regimes the calibrator sees. To broaden coverage, we apply two complementary augmentations, sampled at random per trajectory.

Prediction confidence augmentation. We scale the LLM's logits at every trajectory position by a



(a) Coverage using partial train data (2 LLMs, 5 datasets)



(b) Coverage using full train data (3 LLMs, 10 datasets)

Figure 2: UMAP projections of embeddings derived from prediction trajectories of different LLM-dataset combinations for Phi-4-mini-instruct. Having 3 LLMs and 10 datasets achieves adequate coverage.

shared random temperature $T \in [1, T_{\max}]$ and recompute the prediction-derived features (features 1–4, §3.1). $T_{\max}$ grows with the trajectory's mean confidence so already-confident predictions are perturbed more aggressively, exposing the calibrator to a broader confidence spectrum than the training LLMs alone provide.

Label augmentation. Even with confidence augmentation, some regimes—already well-calibrated trajectories, or severely miscalibrated ones with high confidence but poor accuracy—remain underrepresented. We replace ground-truth labels with samples from a temperature-scaled version of the LLM's predictive distribution and recompute features 3–5. A mild temperature produces labels mostly agreeing with the model (simulating near-calibrated behavior, teaching the calibrator to leave such sequences undisturbed); a high temperature flattens the distribution, producing labels that contradict the model's confidence and exposing it to severe miscalibration. Temperature schedules for both augmentations are in Appendix 3.4.1.

3.5 Extracting LLM's Label Distributions

The features defined in §3.1 require access to the LLM's predictive distribution $P_{\mathcal{M}}(y \mid x_t, \mathcal{C}_{t-1})$ over the full label set $\mathcal{Y}_\tau$. For single-token labels with distinct first tokens, this distribution can be read directly from the next-token logits. However, many classification tasks have labels that span multiple tokens (e.g., `card_in_transit`) or share a common prefix (e.g., `card_arrived` and `card_declined`). We address this by organizing the label set into a *label token trie*, where each path from the root to a leaf corresponds to the token sequence of one label, and calculating label probabilities via a sequence of forward passes that traverse this trie. Appendix E provides a detailed description and example of this structure.

4 Empirical Evaluation

The goal of this evaluation is to test: (a) generalization of our calibrator to unseen model-dataset combinations including black-box models, compared to existing inference-time calibration methods (Table 1 and 2); (b) whether the calibration performance is retained when noisy probability estimates are used (Table 3); (c) effect of number of shots on calibration performance (Figure 3); and (d) ablations for our calibrator (Table 4).

4.1 Baselines

We compare CALIBFORMER against inference-time calibration strategies that do not require addi-

tional task-specific training. We provide a summary here; additional details are in Appendix D.

In-Context Calibration (ICC) (Jang et al., 2024) estimates a prompt-induced "semantic prior" by predicting each demonstration while holding it out from the context. This semantic prior is used to scale the logits for inference-time calibration.

Temperature scaling from autoregressive demonstration predictions (TS-AR) TS-AR fits a single temperature per prompt by minimizing the negative log-likelihood over the autoregressive predictions on the demonstrations.

Temperature scaling from fixed t-shot demonstration predictions (TS-FS) fits a temperature using t-shot predictions for each demonstration (with $t = (k-1)/2$) where the t IC examples are each sampled from the remaining IC examples.

In-Context Permutation (ICP) (Jung et al., 2025) averages predictions across $N = 5$ permutations of the IC examples to reduce sensitivity to order.

Discarded baselines. We could not compare with two recently proposed recalibration methods since they were either dataset or LLM-specific: (i) surprise-vector methods (Tan et al., 2025), which require a fixed class dimension and did not generalize when padded across heterogeneous label spaces, and (ii) Thermometer (Shen et al., 2024), which relies on the specific LLM's hidden states, and is ambiguous for multi-token label scoring.

4.2 LLMs

We train a single CALIBFORMER calibrator on three open-weight LLMs spanning different confidence profiles and instruction-tuning regimes—`Meta-Llama/Llama-3.1-8B-Instruct` (Grattafiori et al., 2024), `Qwen/Qwen3-8B` (Yang et al., 2025), and `OpenAI/Gpt-oss-20b` (OpenAI, 2025). We evaluate transfer on four held-out (unseen) LLMs (three open-weight, one API-based): `Microsoft/Phi-4-mini-instruct` (Abouelenin et al., 2025), `Google/Gemma-4-E4B-IT`, `Qwen/Qwen3.5-9B`, and `Gemini-2.5-Flash`.

4.3 Datasets

We evaluate on held-out (unseen) datasets where calibration matters for downstream decisions:

1. **Tool use**: When2call (Ross et al., 2025) and MetaTool (Huang et al., 2023). Calibrated confidence reduces tool-calling cost by escalating only genuinely uncertain queries.
2. **Toxicity detection**: Wikitoxic (Thain et al., 2017) and Wildguard (Han et al., 2024). Safety filters trade off catching harmful prompts against refusing benign ones; miscalibration directly impacts this operating point.
3. **Intent classification**: Massive Intent (FitzGerald et al., 2023). Calibrated confidence enables reliable fallback (clarification, human hand-off) in intent routing.
4. **Knowledge, logic, and NLU**: MedQA (Jin et al., 2020), Commonsense QA (CS_QA) (Talmor et al., 2019), and Amazon Counterfactual (Amazon_CF) (O'Neill et al., 2021). Overconfident errors in high-stakes settings such as medical QA are costly (Maity and Saikia, 2025).

Dataset and prompt details are in Appendix A.

4.4 Experimental Setup

We provide implementation details in Appendix I.

Demonstration selection. For each test input, we construct a k-shot prompt using one of two ways:

1. **Similarity sampling**: choose the k nearest neighbors from the training split under cosine similarity of 768-dim embeddings obtained from `Google/EmbeddingGemma-300m`.
2. **Random sampling**: sample k demonstrations uniformly at random from the training split.

Evaluation protocol. For all open-weight test LLMs, we run 10 random seeds per dataset/setting and evaluate on 300 test examples per seed (except for the Massive Intent dataset, where we use 5 seeds due to the higher cost of multi-token label scoring). For `Gemini-2.5-flash`, we run 5 seeds and evaluate on 200 queries per seed for $k \in \{5, 6, 8\}$. We only report averages across seeds in the main text; the standard errors are reported in Appendix H.

4.5 Evaluation Metric

We report smooth *Expected Calibration Error (ECE)* (Błasiok and Nakkiran, 2024), which kernel-smooths the standard binned ECE over the prediction–label pairs for more reliable estimates (see Appendix G for the technical details). We additionally report *Brier score* (Brier, 1950) in Appendix H.

4.6 Main Results

Tables 1, 2 and 3 summarize the ECE comparisons for all the unseen models. CALIBFORMER attains the best average calibration across both sampling strategies and all evaluated LLMs. On `Phi-4-mini-instruct`, average ECE drops 18.3% to 6.1% (similarity) and 25.7% to 7.2% (random) (Table 1); on `Gemini-2.5-flash`, 10.2% to 4.0%

Dataset	Phi-4-mini-instruct						Qwen-3.5-9B				Gemma-4-E4B-IT			
	Uncalib	ICC	ICP	TS-AR	TS-FS	CF(Ours)	Uncalib	TS-AR	TS-FS	CF(Ours)	Uncalib	TS-AR	TS-FS	CF(Ours)
Random Sampling														
Amazon_CF	15.3	20.6	8.4	7.8	7.6	**4.7**	**4.9**	7.0	7.2	5.4	14.2	5.8	5.7	**3.4**
CS_QA	21.8	23.7	19.3	7.9	7.4	**5.0**	6.6	6.1	6.3	**5.0**	20.7	7.3	7.1	**4.9**
massive_intent	21.2	28.9	15.3	8.8	8.9	**4.9**	10.7	7.0	7.1	**4.5**	17.2	8.0	7.4	**4.1**
medqa	21.8	23.9	21.2	8.6	8.8	**6.6**	6.0	6.7	6.6	**5.0**	28.8	**7.2**	**7.2**	7.6
metatool	23.0	27.4	17.9	7.1	6.8	**5.1**	12.8	6.2	6.4	**3.9**	20.0	6.9	6.9	**4.9**
when2call	36.4	34.8	33.5	11.6	**11.4**	13.1	14.2	7.6	8.0	**4.7**	25.6	9.0	9.0	**7.9**
wikitoxic	19.9	20.0	14.6	12.6	12.3	**9.0**	10.4	7.0	6.9	**3.4**	15.2	6.6	6.5	**5.9**
wildguard	46.5	43.0	41.0	**8.1**	9.5	9.3	16.3	11.1	11.5	**7.7**	25.3	10.4	10.2	**5.4**
Average	25.7	27.8	21.4	9.1	9.1	**7.2**	10.2	7.3	7.5	**4.9**	20.9	7.6	7.5	**5.5**
Similarity Sampling														
Amazon_CF	17.1	-	11.2	8.6	8.2	**6.0**	**4.3**	7.3	7.3	6.3	18.9	6.8	**5.9**	**5.9**
CS_QA	18.3	-	13.6	9.7	9.9	**4.8**	6.9	6.7	6.7	**4.9**	19.3	7.0	6.8	**4.1**
massive_intent	10.7	-	6.5	7.7	8.1	**5.2**	7.1	6.8	7.2	**4.8**	11.3	8.0	7.7	**3.9**
medqa	17.3	-	14.8	8.8	8.9	**5.0**	**4.4**	6.3	6.5	5.6	24.7	7.7	7.5	**6.5**
metatool	9.4	-	5.9	6.0	5.7	**5.6**	7.6	4.4	4.5	**3.8**	11.6	5.6	5.3	**3.4**
when2call	33.1	-	28.3	**8.0**	8.5	10.5	7.2	5.1	5.5	**4.6**	17.9	6.3	6.6	**4.4**
wikitoxic	15.9	-	12.6	10.2	9.3	**6.5**	10.4	7.2	7.4	**3.4**	14.8	6.4	6.3	**5.8**
wildguard	24.9	-	19.9	8.0	9.9	**4.9**	18.4	9.0	9.7	**5.9**	25.1	8.0	8.6	**7.3**
Average	18.3	-	14.1	8.4	8.6	**6.1**	8.3	6.6	6.8	**4.9**	18.0	7.0	6.8	**5.2**

Table 1: Across all three open weights LLMs, CALIBFORMER outperforms all baseline in 85% of the cases

Datasets	Uncalib		TS-AR		TS-FS		Ours	
	R	S	R	S	R	S	R	S
Amazon_CF	7.8	9.3	4.9	4.7	5.6	5.2	**4.5**	**3.2**
CS_QA	11.9	11.4	8.3	8.2	7.8	7.7	**4.5**	**4.3**
medqa	11.3	9.0	12.3	12.7	6.2	7.0	**4.1**	**4.4**
when2call	17.7	11.0	7.4	6.3	7.1	6.5	**4.7**	**3.9**
Average	12.2	10.2	8.2	8.0	6.7	6.6	**4.5**	**4.0**

Table 2: Comparison of ECE calibration on the `Gemini-2.5-flash` LLM. Values are reported for Random (R) and Similarity (S) sampled contexts. CALIBFORMER reduces the uncalibrated ECE by over 50%.

Sampling	Uncalib		TS-AR		TS-FS		Ours	
	ECE	Brier	ECE	Brier	ECE	Brier	ECE	Brier
Similarity	18.3	.480	11.3	.455	11.3	.456	**6.6**	**.433**
Random	25.8	.661	10.9	.591	11.0	.590	**7.7**	**.570**

Table 3: CALIBFORMER is robust to noisy probabilities on `Phi-4-mini-instruct`

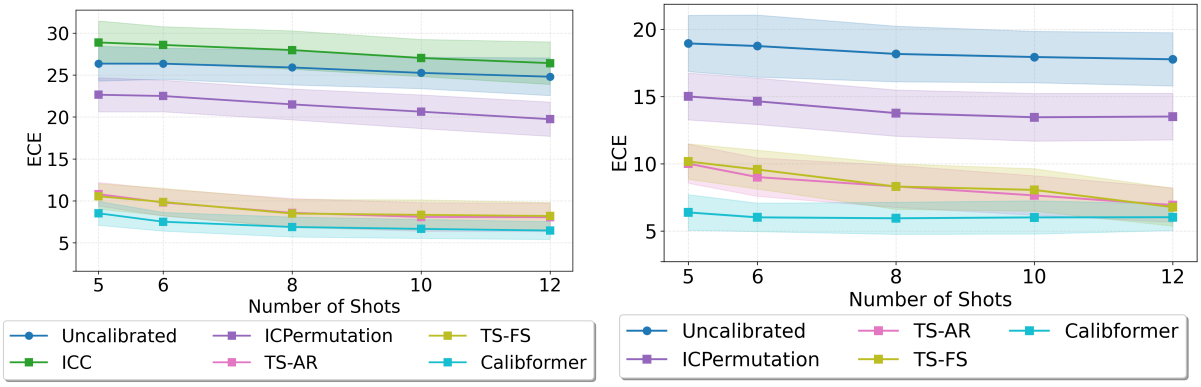
(similarity) and 12.2% to 4.5% (random) (Table 2). While there some rare instances where the base model (`Qwen-3.5-9B`) has better calibration than CALIBFORMER, the difference is generally within the standard error (see Appendix H).

Comparisons to baselines. Temperature scaling (TS-AR/TS-FS) is a strong baseline that improves ECE over uncalibrated ICL, but is worse than CALIBFORMER across datasets and LLMs. Permutation averaging (ICP) is helpful but costly: it reduces ECE on average but remains much worse than CALIBFORMER while requiring N forward passes. ICC, by contrast, is unstable in our setting — under random sampling on `Phi-4-mini-instruct` it *worsens* average ECE (25.7% $\rightarrow$ 27.8%) suggesting that prompt-bias subtraction can be harmful when demonstrations are not representative. We report ICC/ICP results only for one model because it is expensive, and did not show gains on small pilots.

Effect of similarity-based demonstrations. Similarity sampling improves calibration for all methods relative to random sampling (e.g., for `Phi-4-mini-instruct` uncalibrated ECE drops from 25.7% to 18.3%). Nevertheless, even under similarity sampling, the uncalibrated model remains noticeably miscalibrated, and post-hoc calibration remains beneficial.

Effect of noisy or truncated predictions. We simulate noise in the LLM label distribution by estimating empirical probabilities based on $N = 20$ label samples from the `Phi-4-mini-instruct` model. We find that CALIBFORMER performance is quite robust relative to temperature scaling (Table 3). This is because with truncated/noisy probabilities, the ground truth class probability can be too low for some incorrect predictions and more strongly influence the learned temperature scaling.

Calibration across shots. Figure 3 reports ECE as a function of the number of shots, averaged across datasets (dataset-level plots in the Appendix). Although accuracy typically increases with more demonstrations, uncalibrated ECE only mildly improves, suggesting that additional in-context information primarily improves correctness rather



(a) Random sampling

(b) Similarity sampling

Figure 3: For Phi-4-mini-instruct, calibration error tends to on average decrease with shots for all the methods, with CALIBFORMER consistently outperforming all baselines.

than aligning confidence with correctness likelihood. In contrast, all recalibration methods improve with k, indicating that extra demonstrations provide a stronger signal for correcting miscalibration. Across the full range of shots, CALIBFORMER attains the lowest average ECE.

Method	ECE (%)(↓)	Brier (↓)
Uncalibrated	18.12	0.4320
Base(CALIBFORMER)	**5.41**	**0.3807**
Minimal features	5.47*	0.3823***
Half Train Data	5.83***	0.3822***
w/ embeddings	8.46***	0.3941***
w/ permuted embeddings	5.70***	0.3816***
Temperature Output	7.50***	0.3837***
No data augmentation	5.72***	0.3813***

Table 4: Ablation results. Stat. significance relative to the Base is denoted by *** ($p < .001$) and * ($p < .05$).

4.7 Ablations

We ablate CALIBFORMER's design to identify components essential for cross-task, cross-model calibration. Table 4 summarizes the results, with statistical significance tests relative to the Base CALIBFORMER method. Each variant is evaluated on `Phi-4-mini-instruct`, `Gemma-4-E4B-it`, and `Qwen3.5-9B`, results averaged over $k \in 5, \ldots, 15$, all unseen datasets and 30 seeds.

Impact of feature set. Learning with just two features—the LLM's predicted confidence and a correctness indicator—still outperforms temperature-scaling baselines on average, suggesting that a large fraction of the calibration signal is contained in these prediction-derived features.

Impact of training-data diversity. Halving the training data leads to noticeable increase in ECE, indicating that broad task diversity is important for calibration under ICL.

Role of embeddings. We test two embedding variants: simple concatenation of input embeddings to the current features, and permuting input embeddings during training by a random order that is shared for the sequence of inputs. The latter should prevent domain specific memorization but preserve notions of similarity within the input sequence. We observe that, without permutation, the generalization degrades substantially, and with permutation, memorization of embeddings is prevented but calibration is slightly worse than th base model.

Predicting temperature vs. confidence. Finally, we train CALIBFORMER to predict a temperature parameter (to rescale logits) instead of directly predicting calibrated confidence. This yields higher ECE, suggesting that directly modeling correctness probability is a more stable and transferable.

Effect of data augmentation. The average performance over the LLMs is meaningfully reduced in the absence of data augmentation. This is driven primarily by worse performance on `Qwen-3.5-9B`, which is reasonably calibrated out of the box.

5 Conclusion

In this work, we introduced CALIBFORMER, a universal calibrator for in-context learning predictions that generalizes across models and tasks without requiring task-specific fine-tuning. Our approach leverages a transformer-based architecture that treats in-context examples as a latent validation set, learning to predict calibrated confidence scores by attending to discrepancies between model predictions and ground-truth labels within the context window. Our calibrator is trained on 3 models and 10 tasks, and shows significant reduction in ECE across (unseen) tasks/models.

6 Limitations

While we evaluate CALIBFORMER across a broad range of tasks, models, and seeds, several gaps remain. First, our evaluation on proprietary models is limited to Gemini-2.5-Flash. Extending the study to additional closed-source models would strengthen the paper, but is constrained both by inference cost and by the limited log-probabilities these APIs expose. Empirical log-prob estimation via repeated sampling is in principle possible but requires many samples per partial prompt, inflating the cost further.

Second, our method targets classification tasks. Extending it to open-ended generation is not straightforward: the label space is huge, so the trie-based label-probability construction (§3.5) can be very costly, and the notion of a per-position "correctness indicator" that drives our features must be replaced by a continuous quality signal whose calibration target is itself ill-defined.

Third, our method lacks theoretical guarantees. Although recent work has made substantial progress on understanding ICL for classification, most analyses assume an i.i.d. setting. Our calibrator operates on autoregressive demonstration predictions, where features at successive positions are correlated through the shared prompt context—a setting to which existing theory does not directly extend, and we leave this analysis to future work.

References

Abdelrahman Abouelenin, Atabak Ashfaq, Adam Atkinson, Hany Hassan Awadalla, Nguyen Bach, Jianmin Bao, Alon Benhaim, Martin Cai, Vishrav Chaudhary, Congcong Chen, Dongdong Chen, Dongdong Chen, Junkun Chen, Weizhu Chen, Yen-Chun Chen, Yi ling Chen, Qi Dai, Xiyang Dai, Ruchao Fan, and 54 others. 2025. Phi-4-mini technical report: Compact yet powerful multimodal language models via mixture-of-loras. *ArXiv*, abs/2503.01743.

Duarte M. Alves, Nuno M. Guerreiro, Joao Alves, José P. Pombal, Ricardo Rei, José Guilherme Camargo de Souza, Pierre Colombo, and André Martins. 2023. Steering large language models for machine translation with finetuning and in-context learning. In *Conference on Empirical Methods in Natural Language Processing*.

Glenn W. Brier. 1950. Verification of forecasts expressed in terms of probability. *AMS American Metereology Society*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared 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 M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Jarosław Błasiok and Preetum Nakkiran. 2024. Smooth ece: Principled reliability diagrams via kernel smoothing.

André F. Cruz, Moritz Hardt, and Celestine Mendler-Dünner. 2024. Evaluating language models as risk scores. In *Advances in Neural Information Processing Systems*, volume 37, pages 97378–97407. Curran Associates, Inc.

Hy Dang, Tianyi Liu, Zhuofeng Wu, Jingfeng Yang, Haoming Jiang, Tao Yang, Pei Chen, Zhengyang Wang, Helen Wang, Huasheng Li, Bing Yin, and Meng Jiang. 2025. Improving large language models function calling and interpretability via guided-structured templates. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 24426–24442, Suzhou, China. Association for Computational Linguistics.

Jack FitzGerald, Christopher Hench, Charith Peris, Scott Mackie, Kay Rottmann, Ana Sanchez, Aaron Nash, Liam Urbach, Vishesh Kakarala, Richa Singh, Swetha Ranganath, Laurie Crist, Misha Britan, Wouter Leeuwis, Gokhan Tur, and Prem Natarajan. 2023. MASSIVE: A 1M-example multilingual natural language understanding dataset with 51 typologically-diverse languages. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4277–4302, Toronto, Canada. Association for Computational Linguistics.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Seungju Han, Kavel Rao, Allyson Ettinger, Liwei Jiang, Bill Yuchen Lin, Nathan Lambert, Yejin Choi, and Nouha Dziri. 2024. Wildguard: open one-stop moderation tools for safety risks, jailbreaks, and refusals of llms. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA. Curran Associates Inc.

Yue Huang, Jiawen Shi, Yuan Li, Chenrui Fan, Siyuan Wu, Qihui Zhang, Yixin Liu, Pan Zhou, Yao Wan, Neil Zhenqiang Gong, and Lichao Sun. 2023. Metatool benchmark for large language models: Deciding whether to use tools and which to use. *ArXiv*, abs/2310.03128.

Joonwon Jang, Sanghwan Jang, Wonbin Kweon, Minjin Jeon, and Hwanjo Yu. 2024. Rectifying demonstration shortcut in in-context learning. *ArXiv*, abs/2403.09488.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. 2020. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *arXiv preprint arXiv:2009.13081*.

Jaehun Jung, Faeze Brahman, and Yejin Choi. 2025. Trust or escalate: LLM judges with provable guarantees for human agreement. In *The Thirteenth International Conference on Learning Representations*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Yue Li, Zhixue Zhao, and Carolina Scarton. 2025. It's all about in-context learning! teaching extremely low-resource languages to LLMs. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 29532–29547, Suzhou, China. Association for Computational Linguistics.

Subhankar Maity and Manob Jyoti Saikia. 2025. Large language models in healthcare and medical applications: A review. *Bioengineering*, 12(6).

James O'Neill, Polina Rozenshtein, Ryuichi Kiryo, Motoko Kubota, and Danushka Bollegala. 2021. I wish I would have loved this one, but I didn't – a multilingual dataset for counterfactual detection in product review. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7092–7108, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

OpenAI. 2025. gpt-oss-120b and gpt-oss-20b model card. *Preprint*, arXiv:2508.10925.

Kishan Padayachy, Ronald Richman, Salvatore Scognamiglio, and Mario V. Wüthrich. 2026. In-context learning enhanced credibility transformer. *Preprint*, arXiv:2509.08122.

Seetharaman Prem, Wichern Gordon, Pardo Bryan, and Le Roux Jonathan. 2020. Autoclip: Adaptive gradient clipping for source separation networks. In *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE.

Hayley Ross, Ameya Sunil Mahabaleshwarkar, and Yoshi Suhara. 2025. When2Call: When (not) to call tools. In *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. Association for Computational Linguistics.

Maohao Shen, Subhro Das, Kristjan Greenewald, Prasanna Sattigeri, Gregory W. Wornell, and Soumya Ghosh. 2024. Thermometer: Towards universal calibration for large language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 44687–44711. PMLR.

Aaditya K. Singh, Stephanie C. Y. Chan, Ted Moskovitz, Erin Grant, Andrew M. Saxe, and Felix Hill. 2023. The transient nature of emergent in-context learning in transformers. *ArXiv*, abs/2311.08360.

Mark Steyvers, Heliodoro Tejeda Lemus, Aakriti Kumar, Catarina Belem, Sheer Karny, Xinyue Hu, Lukas Mayer, and Padhraic Smyth. 2024. The calibration gap between model and human confidence in large language models. *ArXiv*, abs/2401.13835.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Zhihang Tan, Jingrui Hou, Ping Wang, Qibiao Hu, and Peng Zhu. 2025. Surprise calibration for better in-context learning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 23056–23071, Suzhou, China. Association for Computational Linguistics.

Nithum Thain, Dixon Lucas, and Wulczyn Ellery. 2017. Wikipedia talk labels: Toxicity.

Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher Manning. 2023. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*.

Dennis Ulmer, Martin Gubri, Hwaran Lee, Sangdoo Yun, and Seong Oh. 2024. Calibrating large language models using their generations only. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15440–15459, Bangkok, Thailand. Association for Computational Linguistics.

Zhiyong Wu, Yaoxiang Wang, Jiacheng Ye, and Lingpeng Kong. 2023. Self-adaptive in-context learning: An information compression perspective for in-context example selection and ordering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 1423–1436. Association for Computational Linguistics.

Miao Xiong, Zhiyuan Hu, Xinyang Lu, YIFEI LI, Jie Fu, Junxian He, and Bryan Hooi. 2024. Can LLMs express their uncertainty? an empirical evaluation of confidence elicitation in LLMs. In *The Twelfth International Conference on Learning Representations*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Hanlin Zhang, YiFan Zhang, Yaodong Yu, Dhruv Madeka, Dean Foster, Eric Xing, Himabindu Lakkaraju, and Sham Kakade. 2024. A study on the calibration of in-context learning. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6118–6136.

Xingxuan Zhang, Haoran Wang, Jiansheng Li, Yuan Xue, Shikai Guan, Renzhe Xu, Hao Zou, Han Yu, and Peng Cui. 2025. Understanding the generalization of in-context learning in transformers: An empirical study. In *The Thirteenth International Conference on Learning Representations*.

Tony Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *International Conference on Machine Learning*.

Han Zhou, Xingchen Wan, Lev Proleev, Diana Mincu, Jilin Chen, Katherine A. Heller, and Subhrajit Roy. 2024. Batch calibration: Rethinking calibration for in-context learning and prompt engineering. In *The Twelfth International Conference on Learning Representations*.

A Dataset Details

Dataset	NC	HF Repo / Source
Training Datasets		
snli	3	stanfordnlp/snli
sst5	5	SetFit/sst5
rte	2	SetFit/rte
Trec-QC	6	SetFit/TREC-QC
agnews	4	SetFit/ag_news
goemotions	28	mrm8488/goemotions
dbpedia_70	70	DeveloperOats/DBPedia_Classes
toxic chat	2	lmsys/toxic-chat
newsgroups20	20	SetFit/20_newsgroups
yahoo answers	10	community-datasets/yahoo_answers_topics
Validation Datasets		
qqp	2	SetFit/qqp
banking77	77	mteb/banking77
dbpedia_9	9	DeveloperOats/DBPedia_Classes
Yelp reviews	5	Yelp/yelp_review_full
Test (Unseen) Datasets		
Amazon_CF	2	mteb/amazon_counterfactual
CS_QA	5	tau/commonsense_qa
massive intent	60	mteb/amazon_massive_intent
medqa	4	bigbio/med_qa
metatool	47	MetaTool GitHub
when2call	3	nvidia/When2Call
wikitoxic	2	OxAISH-AL-LLM/wiki_toxic
wildguard	15	allenai/wildguardmix

Table 5: Consolidated Datasets: Training, Validation, and Test

B Calibrator Overview

CALIBFORMER is a causal transformer encoder that is a stack of the following layers:

1. A linear embedding layer to which sinusoidal position embeddings are added
2. 12 standard transformer encoder layers with hidden dimension 64 and 8 attention heads. All layers use a causal mask for attention.
3. A final linear layer that maps R^h hidden features to R
4. A sigmoid layer that maps the logits from the last layer to likelihood of correctness of the prediction, which serves as the calibrated confidence

C Related Work

Calibration of LLM Predictions. Modern LLMs are known to be overconfident, assigning high probabilities to predictions even when the true answer involves substantial aleatoric uncertainty (Cruz et al., 2024; Steyvers et al., 2024; Xiong et al.,

2024; Tian et al., 2023). This miscalibration is exacerbated under in-context learning: Zhang et al. (2024) show that calibration error does not follow a predictable pattern as the number of demonstrations increases, and that standard post-hoc methods such as temperature scaling do not reliably improve it. These findings motivate the development of calibration techniques designed specifically for the ICL setting.

Prompt-Bias Correction Methods. A prominent family of approaches frames ICL miscalibration as a prompt-induced label bias and seeks to subtract it from the predicted logits. Contextual Calibration (Zhao et al., 2021) estimates this bias by replacing the test input with content-free tokens such as "N/A" or "[MASK]". Batch Calibration (Zhou et al., 2024) instead averages prediction logits over the test batch to estimate the label prior. In-Context Calibration (Jang et al., 2024) uses only the in-context examples to estimate semantic priors, avoiding the need for additional test-time data. These methods primarily target accuracy improvement rather than statistical calibration, require task-specific labeled data and are designed for randomly sampled demonstrations. When demonstrations are selected by similarity to the test input, subtracting the estimated "bias" can reduce confidence in the correct class and degrade both accuracy and calibration.

Learned Calibration from In-Context Signals. A more recent direction learns to calibrate by modeling the sequential structure of in-context predictions. Surprise Calibration (Tan et al., 2025) interprets ICL as sequential Bayesian inference and extracts from the negative log-probability of the true label at each demonstration position a per-class logit adjustment. While conceptually related to our use of prediction–label discrepancies across the demonstration sequence, Surprise Calibration differs in several important respects: it trains a separate model per LLM–task pair, it targets accuracy rather than calibration error, and it has been evaluated only on tasks with 2–3 label spaces.

Separately, the Simulated Annotators approach (Jung et al., 2025) improves calibration by averaging predictions across N random permutations of the in-context examples. While effective, this strategy requires N forward passes per test input, making it substantially more expensive than methods that calibrate a single prediction.

Auxiliary Calibration Models. An alternative line of work trains a dedicated auxiliary model to predict an LLM's confidence. APRICOT (Ulmer et al., 2024) trains a DeBERTa model on the textual input and the LLM's generated answer, using cluster-level accuracy computed by grouping questions with similar embeddings as the calibration target. The auxiliary model must be retrained for each target LLM, and the approach does not leverage in-context examples or their sequential structure. Thermometer (Shen et al., 2024) trains an MLP on the LLM's hidden-state representations to predict a per-task calibration temperature. While some transfer is observed within model families and across related tasks, the reliance on internal representations of fixed dimensionality ties the calibrator to a specific LLM architecture.

Both approaches lack the cross-model and cross-task universality that CALIBFORMER targets. In contrast, our calibrator operates on task-agnostic, prediction-derived features and exploits the sequential structure of in-context predictions, enabling a single trained model to generalize to unseen LLMs and tasks.

D Baseline Details

We provide the formal details for our baselines here. For all methods below, we treat returned log-probabilities as logits (vLLM does not expose logits directly; softmax-normalized probabilities are identical either way).

In-Context Calibration (ICC). (Jang et al., 2024) ICC estimates a prompt-induced "semantic prior" by predicting each demonstration while holding it out from the context. Let $\mathcal{C} = \{(x_1, y_1), \ldots, (x_k, y_k)\}$ be the ordered demonstrations. We compute

$$P_i = P\big(y \mid x_i, \{(x_j, y_j)\}_{j\in[k]\setminus\{i\}}\big)\,, \bar{P} = \frac{1}{k}\sum_{i=1}^{k} P_i,$$
$$\tilde{\mathbf{z}} = \mathbf{z} - \log \bar{P}. \tag{1}$$

where $\mathbf{z}$ and $\tilde{\mathbf{z}}$ denote the original and calibrated logits, respectively.

Temperature scaling from autoregressive demonstration predictions (TS-AR). TS-AR fits a single temperature per prompt using

autoregressive predictions on the demonstrations:

$$\mathbf{z}_i = f_{\text{LM}}\Big(x_i \mid \{(x_j, y_j)\}_{j=1}^{i-1}\Big)\,,$$
$$T = \arg\min_T \sum_{i=1}^{k} \text{NLL}(\sigma(\mathbf{z}_i/T), y_i)\ , \tag{2}$$

where NLL refers to negative log-likelihood.

Temperature scaling from fixed t-shot demonstration predictions (TS-FS). TS-FS fits a temperature using t-shot predictions for each demonstration (with $t = (k-1)/2$):

$$\mathcal{C}_i^{(t)} \subset \mathcal{C} \setminus \{(x_i, y_i)\}, \quad |\mathcal{C}_i^{(t)}| = t,$$
$$\mathbf{z}_i^{(t)} = f_{\text{LM}}\Big(x_i \mid \mathcal{C}_i^{(t)}\Big)\,,$$
$$T^* = \arg\min_T \sum_{i=1}^{k} \text{NLL}\Big(\sigma(\mathbf{z}_i^{(t)}/T),\, y_i\Big)\,. \tag{3}$$

Permutation averaging (ICP). (Jung et al., 2025) To reduce sensitivity to demonstration order, In-Context Permutation (ICP) averages predictions across $N = 5$ random permutations of $\mathcal{C}$:

$$\bar{P}(y \mid x_{test}) = \frac{1}{N}\sum_{n=1}^{N} p(y \mid x_{test}, \pi_n(\mathcal{C}))\,. \tag{4}$$

Discarded baselines. We could not compare with two recently proposed recalibration methods since they were either dataset or LLM-specific: (i) surprise-vector methods (Tan et al., 2025), which require a fixed class dimension and did not generalize when padded across heterogeneous label spaces, and (ii) Thermometer (Shen et al., 2024), which relies on access to and training on the specific LLM's hidden states, and is ambiguous for multi-token label scoring.

Implementation Details for Temperature Search. All temperature scaling methods optimize a per-sequence temperature—a scalar $T_i \in \mathbb{R}_{>0}$ for each test instance i—by minimizing the negative log-likelihood over the demonstration predictions associated with that instance. The optimization is performed via up to 500 iterations of ternary search over the interval $[0.01, 100.0]$, since the cross-entropy loss is convex (and hence unimodal) in $\log T$. Search terminates when the interval width falls below $\tau = 10^{-4}$, yielding temperature estimates accurate to within 10^{-4} in absolute value. For CALIBFORMER the equivalent temperature is recovered via binary search over the same interval $[0.01, 100.0]$ with 500 iterations: given the calibrated confidence $\hat{c}_{k+1} \in (0, 1)$ predicted by the transformer, we find $T^* = \arg\min_T |\max_j \text{softmax}(T \cdot \mathbf{z}_{k+1})_j - \hat{c}_{k+1}|$ by bisection. In the rare case that $\hat{c}_{k+1}$ exceeds the maximum achievable confidence under the upper bound $T = 100.0$, the temperature is clamped to 100.0.

E Extracting Label Distribution with a Label Token Trie

As mentioned in §3.5, we calculate label probabilities by organizing the label set into a *label token trie*, where each path from the root to a leaf corresponds to the token sequence of one label.

At each internal node, we perform a forward pass with the partial label appended to the prompt and extract logits only over the valid next tokens—those that correspond to children of the current node. Applying a softmax over these restricted logits yields a properly normalized conditional distribution at each branching point. The probability of a complete label is then the product of conditional probabilities along its path:

$$P_{\mathcal{M}}(y \mid x, \mathcal{C}) = \prod_{i=1}^{|y|} P_{\mathcal{M}}\Big(y^{(i)} \mid y^{(1:i-1)}, x, \mathcal{C}\Big)\,, \tag{5}$$

where $y^{(i)}$ denotes the i-th token of label y and $y^{(1:i-1)}$ the preceding tokens. Since the softmax at each branching point is taken over the valid continuations only, the resulting label probabilities are guaranteed to sum to 1 over $\mathcal{Y}_\tau$ regardless of the number of classes or the depth of the trie.

A toy illustration of our label token trie application is given below for the following labels:

- `card_arrived`
- `card_declined`
- `card_in_transit`
- `cash_transferred`

```
ROOT
|-- card
|   |-- _arrived    -> card_arrived
|   |-- _declined   -> card_declined
|   `-- _in
|       `-- _trans
|           `-- it  -> card_in_transit
`-- cash
    `-- _trans
        `-- ferred  -> cash_transferred
```

We first obtain the logits for tokens `card` and `cash` by prompting the model, then append, for example, `card` to the prompt and perform inference to get logits for the next token candidates: `_arrived`, `_declined`, and `_in`. Since `_transit` is the only valid completion after `card_in`, we assign a dummy logit to it, as the softmax over the only candidate next token will yield a probability of 1 regardless of the true logit value.

$$P(\text{card_in_transit} \mid x, C) = P(\texttt{card} \mid x, C) \times P(\texttt{_in} \mid \texttt{card}, x, C) \times P(\texttt{_transit} \mid \texttt{card_in}, x, C)$$

F Data Augmentation Details

This section provides the precise temperature schedules for the two data augmentation strategies described in §F. In both cases, let $\bar{c}$ denote the mean prediction confidence across all positions in the original (unaugmented) trajectory.

F.1 Prediction Confidence Augmentation

We define a sharpness parameter

$$\beta = \begin{cases} 4, & \text{if } \bar{c} < 0.95, \\ 5, & \text{otherwise,} \end{cases} \tag{6}$$

and compute the upper bound of the temperature range as

$$T_{\max} = 1 + \max(0,\ \tanh(\beta\,(\bar{c} - 0.5)))\,. \tag{7}$$

A temperature $T \sim \mathcal{U}[1, T_{\max}]$ is sampled and applied to the logits $\mathbf{z}_t$ at every position: $\mathbf{z}_t \leftarrow \mathbf{z}_t/T$. Features 1–4 from §3.1 are then recomputed from the rescaled logits. The higher value of β for very confident trajectories ($\bar{c} \geq 0.95$) ensures that near-certain predictions receive meaningful perturbation, while the $\tanh$ saturation prevents excessively large temperatures that would flatten the distribution beyond any realistic miscalibration regime.

F.2 Label Augmentation

For a given trajectory, we simulate a near-calibrated regime with probability $\bar{c}/2$ and a severely miscalibrated regime otherwise. Each regime defines its own temperature interval from which T is sampled.

Near-calibrated regime. We set

$$\beta = \begin{cases} 1, & \text{if } \bar{c} < 0.95, \\ 2, & \text{otherwise,} \end{cases}$$

$$T_{\max} = 1 + \max(0,\ \tanh(\beta\,(\bar{c} - 0.5)))\,,$$

and sample $T \sim \mathcal{U}[1, T_{\max}]$. The mild scaling keeps the sampling distribution close to the original predictions, producing synthetic labels that are largely consistent with the model's confidence.

Miscalibrated regime. We set β as above and compute a wider interval:

$$T_{\max} = 2{+}3\ \max(0.5,\ \tanh(\beta\,(\bar{c} - 0.4)))\,, \tag{8}$$

with $T \sim \mathcal{U}[2, T_{\max}]$. The substantially larger temperatures flatten the predictive distribution, generating labels that frequently disagree with the model's stated confidence.

In both regimes, the augmented labels are obtained by sampling from the temperature-scaled softmax at each position:

$$\tilde{y}_t \sim \text{Categorical}(\text{softmax}(\mathbf{z}_t/T))\,, \tag{9}$$

and features 3–5 from §3.1 are recomputed using $\tilde{y}_t$ in place of the original ground-truth labels.

F.3 Training with embeddings

For the ablations that test beneficiality of embeddings, we use `google/embeddinggemma-300m` with an embedding dimension of 128. In order to promote the model relying on ICL instead of IWL (In Weights Learning) to predict likelihood of correctness of prediction, during training, we permute each embedding within each sequence of inputs to the transformer by a random but shared order, and this order is different for each sequence. We do this only during the data loading phase. The random permutations during training ensure that:

1. The transformer does not have access to a consistent representation of the sentences to memorize and use for IWL.
2. The transformer can only learn to calculate similarity at the current position with its in context examples.
3. One may potentially use any arbitrary embedding model to generate the embeddings to be used as features for the transformer since we only need to use them for calculating similarities - so long as the embedding dimension is the same as the one used during training(in our case, 128). This better fits our theme of

having a universal calibrator that uses general features.

When trained without permuting the embeddings this way, the trained calibrator had overfitted to the training dataset and generalized poorly to unseen datasets.

At test time, we use the embeddings as is, without permutation.

F.4 Additional Training Details

For generating the training datasets for CALIBFORMER for each training dataset-LLM combination in table 5, we generate 5000 sequences of autoregressive predictions over the in-context examples, each of length 21 in order to learn up to 20 shot calibration, comprising at each step of features described in Section 3.1.

For the checkpoint selection and ablation evaluations, we generate 1000 sequences for each validation/unseen dataset in table 5. During training, a checkpoint is saved whenever *both* calibrated ECE and Brier scores improve for the validation datasets over the last saved checkpoint.

For all generated calibration training, validation and test(unseen) datasets, the demonstrations are equally divided

During training, at each iteration, we sample a fraction of the training LLM-dataset-sampling strategy combinations based on the `dataset dropout` hyperparameter reported in table 18. This prevents certain combinations from dominating the loss and helps regularise the training of the calibrator for better generalisation

G Evaluation Metrics

To assess calibration quality, we compute Expected Calibration Error (ECE) on the evaluation datasets using smooth ECE (Błasiok and Nakkiran, 2024) which produces more reliable estimates by performing kernel smoothing over the prediction-label pairs. This provides a smoothed estimate instead of the following binned approach:

$$\text{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} \left|\text{acc}(B_m) - \text{conf}(B_m)\right|,$$

where B_m are confidence bins, $\text{acc}(B_m)$ is the empirical accuracy, and $\text{conf}(B_m)$ is the mean predicted confidence in each bin. Additionally, we calculate brier score (Brier, 1950) defined as

$$\text{BS} = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{C} (\hat{p}_{ij} - y_{ij})^2$$

where N is the number of samples, C is the number of classes, $\hat{p}_{ij}$ is the predicted probability for class j on sample i, and $y_{ij} \in \{0, 1\}$ is the one-hot encoded true label.

H Additional Experimental Results

H.1 Tables

In addition to ECE, we report the Brier score for all the LLMs in tables 6 and 7. The accuracy across datasets for Phi-4-mini-instruct for both the demonstration sampling settings can be found in Tables 8 and 9. The variances of the ECEs and Brier scores for all open weights LLMs can be found in tables 10-15

H.2 Plots

The `Phi-4-mini-instruct` calibration performance of different methods across shots for all datasets can be found in Figures 4-10.

I Additional Implementational Details

I.1 Prompts

We include in our prompts a base prompt that tells the LLM the label space for proper usage of the LLM's ICL ability. Tables 16 and 17 show some examples of what 1-shot prompts would look like with `phi-4-mini-instruct` and the chat template(as provided by the tokenizer) applied

I.2 LLM inference

vLLM (Kwon et al., 2023) has been a popular inference framework for LLM serving that provides performance speed up over Huggingface through optimized CUDA kernels, prefix caching, PagedAttention, efficient batching and so forth. We use vLLM 0.19.0 installed with pytorch 2.10 and cuda 12.8 for all our open-weights LLM experiments. Each call to the LLM is made with some non default arguments in `SamplingParams` of which the following might be of interest:

- `max_tokens=1`
- `logprobs=len(label_tokens_ids)`
- `skip_special_tokens=True`
- `allowed_token_ids=label_token_ids`

It has been reported[1] that there is an $O(N)$ time complexity for inference time w.r.t the number of

[1] https://github.com/vllm-project/vllm/issues/14300

Dataset	Phi-4-mini-instruct						Qwen-3.5-9B				Gemma-4-E4B			
	Uncalib	ICC	ICP	TS-AR	TS-FS	Ours	Uncalib	TS-AR	TS-FS	Ours	Uncalib	TS-AR	TS-FS	Ours
Random Sampling														
Amazon_CF	.366	.481	**.303**	.348	.350	.319	**.149**	.174	.175	.158	.253	.250	.247	**.219**
CS_QA	.492	.531	.471	.458	.456	**.439**	**.234**	.253	.254	**.234**	.393	.360	.359	**.344**
massive_intent	.637	.766	**.587**	.600	.601	.592	.320	.327	.326	**.306**	.366	.353	.350	**.333**
medqa	.685	.704	.675	.642	.643	**.621**	**.326**	.353	.352	.327	.624	.532	.533	**.525**
metatool	.586	.677	.545	.547	.542	**.532**	.284	.283	.285	**.264**	.345	.315	.316	**.307**
when2call	.814	.817	.788	.657	**.655**	.661	.423	.423	.427	**.398**	.490	.440	.438	**.421**
wikitoxic	.487	.474	.429	.458	.451	**.421**	.222	.244	.244	**.203**	.261	.243	.245	**.228**
wildguard	1.1	1.1	1.1	**.887**	.891	.892	.505	.510	.511	**.496**	.519	.481	.482	**.469**
Average	.646	.694	.612	.575	.574	**.560**	.308	.321	.322	**.298**	.406	.372	.371	**.356**
Similarity Sampling														
Amazon_CF	.401	-	.347	.364	.365	**.343**	**.155**	.183	.183	.165	.366	.333	.321	**.300**
CS_QA	.475	-	**.428**	.464	.466	.437	.248	.265	.266	**.247**	.363	.335	.334	**.317**
massive_intent	.271	-	**.242**	.268	.270	.256	.200	.205	.208	**.196**	.226	.223	.223	**.206**
medqa	.610	-	.594	.596	.596	**.571**	**.282**	.313	.314	.289	.539	.484	.482	**.468**
metatool	.244	-	**.224**	.238	.236	.230	.170	.165	.167	**.157**	.194	.177	.176	**.169**
when2call	.763	-	.717	**.599**	.601	.610	.263	.259	.259	**.250**	.319	.277	.277	**.260**
wikitoxic	.373	-	.337	.354	.346	**.325**	.213	.214	.215	**.191**	.259	.231	.232	**.222**
wildguard	.668	-	.623	.615	.631	**.602**	.437	.423	.428	**.402**	.461	.416	.421	**.404**
Average	.476	-	.439	.437	.439	**.422**	.246	.253	.255	**.237**	.341	.309	.308	**.293**

Table 6: Brier Scores for calibration methods across the open weights LLMs

Datasets	Uncalib R / S	TS-AR R / S	TS-FS R / S	Ours R / S
Amazon_CF	.170/.194	.181/.189	.188/.193	**.157/.175**
CS_QA	.241/.235	.262/.251	.260/.249	**.224/.220**
medqa	.226/.196	.267/.248	.235/.213	**.207/.191**
when2call	.349/.207	.333/.198	.333/.199	**.305/.181**
Average	.246/.208	.261/.221	.254/.214	**.223/.192**

Table 7: Comparison of Brier scores on the `Gemini-2.5-flash` LLM. Values are reported as Random (R) / Similarity (S) sampled contexts.

Datasets	Calibration Methods				
	Uncalib	ICP	TS-AR	TS-FS	Ours
Amazon_CF	76.6	77.1	76.6	76.6	76.6
CS_QA	70.6	71.5	70.6	70.6	70.6
massive_intent	83.0	84.0	83.0	83.0	83.0
medqa	57.1	57.8	57.1	57.1	57.1
metatool	84.4	85.5	84.4	84.4	84.4
when2call	53.3	54.8	53.3	53.3	53.3
wikitoxic	78.4	79.0	78.4	78.4	78.4
wildguard	55.2	56.2	55.2	55.2	55.2
Average	69.8	70.7	69.8	69.8	69.8

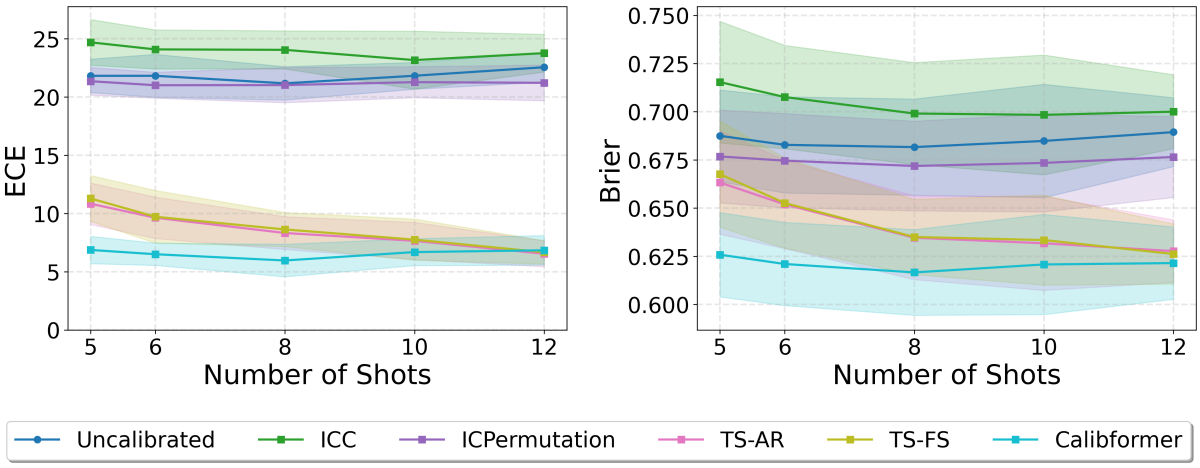
Table 8: Accuracy comparison for calibration methods | Phi-4-mini-instruct | Similarity Sampling

`logprobs` requested. The `allowed_token_ids` parameter out of the box only applies to the actual generated text and not the `raw_logprobs` returned by vLLM for each prompt/generated token, so we added a two line modification to the source code (v1/sampler.py) to also apply the parameter to `raw_logprobs`, masking the non-label tokens in the raw logits to $-inf$ which makes sure the top `len(label_tokens_ids)` tokens are the label tokens themselves. This minimal change completely circumvents the issue and allows efficient inference.

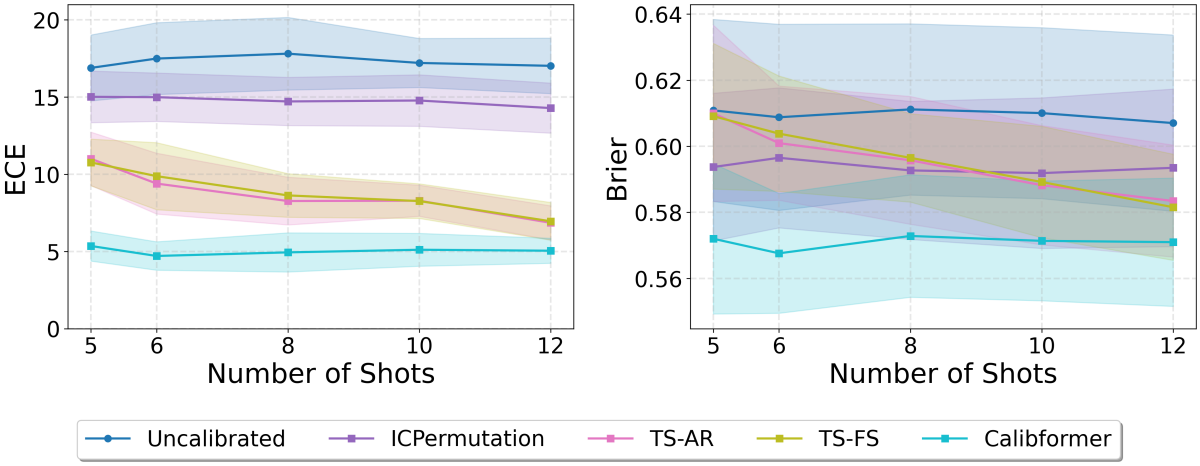
While transformers naturally provide token logits at all positions, including the prompt tokens, and this makes getting the predictions over the in context examples possible without any extra LLM calls. However, this is not viable when using black box LLMs that do not return prompt logprobs, and additional k calls must be made to calibrate a k-shot prompt in such cases, removing the computational edge over the other approaches. Additional calls must also be made when the labels are not distinguishable by their first tokens, although most tasks tend to have simpler label spaces.

With regards to vLLM, while returning prompt logprobs is supported and improved by the simple source code modification that lets work exclusively with label tokens, the prefix caching feature is disabled when requesting prompt logprobs purely due to absence of optimization demand in this setting [2].

[2] github.com/vllm-project/vllm/issues/13414



(a) Random sampling



(b) Similarity sampling

Figure 4: Calibration performance | Phi 4 mini instruct | MedQA

	Calibration Methods					
Datasets	Uncalib	ICC	ICP	TS-AR	TS-FS	Ours
Amazon_CF	78.0	69.6	79.7	78.0	78.0	78.0
CS_QA	69.7	67.7	70.1	69.7	69.7	69.7
massive_intent	58.5	51.1	60.2	58.5	58.5	58.5
medqa	52.5	51.2	52.4	52.5	52.5	52.5
metatool	61.8	55.8	63.0	61.8	61.8	61.8
when2call	52.6	50.1	53.5	52.6	52.6	52.6
wikitoxic	72.6	72.7	73.7	72.6	72.6	72.6
wildguard	26.4	31.1	26.2	26.4	26.4	26.4
Average	59.0	56.2	59.8	59.0	59.0	59.0

Table 9: Accuracy comparison for calibration methods | Phi-4-mini-instruct | Random Sampling

However, our agenda is to provide a proof of concept and source code modifications/feature requests can be made if needed.

We also provide a Huggingface implementation for LLM inference which supports efficient prompt logprobs extraction.

We ran all our training and evaluation experiments on Nvidia B200 GPUs.

For the gemini experiments, we make calls to `gemini-2.5-flash-preview-09-2025`, requesting the top 20 (maximum allowed) logprobs and generating a single output token.

I.3 Calibrator Training Hyperparameters

We train the $0.6M$ parameter CALIBFORMER on a single Nvidia B200 with the hyperparameters listed in Table 18. Training a single checkpoint takes less than 30 mins on a single B200 with our volume of data.

For gradient clipping, we adopt the AutoClip strategy of Prem et al. (2020), which sets the clipping threshold adaptively as the p-th percentile of the gradient norm history observed during training. We use $p = 20$ with a history window of $40{,}000$ iterations.

The choice of weight decay parameters and ar-

Dataset	Uncalibrated	ICC	ICP	TS-AR	TS-FS	CALIBFORMER
Random Sampling						
Amazon_CF	15.3 (1.8)	20.6 (2.2)	8.4 (1.7)	7.8 (1.6)	7.6 (1.4)	**4.7 (1.0)**
CS_QA	21.8 (1.8)	23.7 (2.3)	19.3 (1.6)	7.9 (1.5)	7.4 (1.5)	**5.0 (1.0)**
massive_intent	21.2 (2.0)	28.9 (2.5)	15.3 (1.6)	8.8 (2.0)	8.9 (2.0)	**4.9 (0.7)**
medqa	21.8 (1.4)	23.9 (1.9)	21.2 (1.3)	8.6 (1.6)	8.8 (1.7)	**6.6 (1.2)**
metatool	23.0 (2.2)	27.4 (2.1)	17.9 (2.3)	7.1 (1.4)	6.8 (1.4)	**5.1 (1.0)**
when2call	36.4 (2.1)	34.8 (3.1)	33.5 (2.1)	11.6 (1.9)	**11.4 (1.9)**	13.1 (2.0)
wikitoxic	19.9 (2.7)	20.0 (2.6)	14.6 (2.6)	12.6 (1.7)	12.3 (1.7)	**9.0 (1.2)**
wildguard	46.5 (2.5)	43.0 (2.6)	41.0 (2.6)	**8.1 (1.7)**	9.5 (1.8)	9.3 (1.7)
Average	25.7 (2.1)	27.8 (2.4)	21.4 (2.0)	9.1 (1.7)	9.1 (1.7)	**7.2 (1.3)**
Similarity Sampling						
Amazon_CF	17.1 (1.7)	-	11.2 (1.8)	8.6 (1.4)	8.2 (1.4)	**6.0 (1.4)**
CS_QA	18.3 (1.8)	-	13.6 (1.6)	9.7 (1.5)	9.9 (1.4)	**4.8 (1.0)**
massive_intent	10.7 (1.6)	-	6.5 (1.1)	7.7 (1.3)	8.1 (1.3)	**5.2 (0.9)**
medqa	17.3 (2.1)	-	14.8 (1.6)	8.8 (1.5)	8.9 (1.5)	**5.0 (1.0)**
metatool	9.4 (1.6)	-	5.9 (1.3)	6.0 (1.2)	5.7 (1.3)	**5.6 (1.0)**
when2call	33.1 (2.8)	-	28.3 (2.3)	**8.0 (1.5)**	8.5 (1.4)	10.5 (2.2)
wikitoxic	15.9 (3.2)	-	12.6 (2.8)	10.2 (1.6)	9.3 (1.8)	**6.5 (1.3)**
wildguard	24.9 (2.0)	-	19.9 (1.5)	8.0 (1.8)	9.9 (2.1)	**4.9 (0.7)**
Average	18.3 (2.2)	-	14.1 (1.8)	8.4 (1.5)	8.6 (1.5)	**6.1 (1.3)**

Table 10: Calibration Error (ECE) for Phi-4-mini-instruct across calibration methods. Standard deviations reported in parentheses.

chitectural specifications were inspired by (Singh et al., 2023), which suggest that a weight decay on the MLP layers encourages ICL emergence over In-Weights Learning/memorization.

I.4 Ablation Statistical Significance

To verify that the performance degradations observed in Table 4 reflect genuine differences rather than random variation across training runs, we conduct paired statistical significance tests across 30 random seeds. For each ablation, we compute the per-seed difference in ECE and Brier score relative to CALIBFORMER and test the one-sided hypothesis that the ablation performs worse than the baseline (i.e., produces higher ECE or Brier score). Since the direction of degradation is pre-specified by construction—each ablation removes or degrades a deliberately designed component—a one-sided test is appropriate. We first apply the Shapiro–Wilk normality test to the paired differences; if normality holds ($p \geq 0.05$), we use a paired t-test, otherwise we fall back to the Wilcoxon signed-rank test. This fallback was triggered for the ECE differences of Half Train Data and Temperature Output.

All ablations produce statistically significant degradation on both ECE and Brier ($p < 0.05$). The weakest result is Minimal Features ECE ($p = 0.030$), which suggests that the confidence and correctness alone carry most of the signal; although the corresponding Brier degradation is still highly significant ($p < 0.001$). The strongest degradations are observed for w/ permuted embeddings and Temperature Output, consistent with their mean differences in Table 4. These results confirm that each design choice contributes meaningfully to CALIBFORMERś calibration performance.

J LLM Usage Declaration

We acknowledge that LLMs were used in the refinement of parts of the article and in the implementation of the code.

Dataset	Uncalibrated	ICC	ICP	TS-AR	TS-FS	CALIBFORMER
			Random Sampling			
Amazon_CF	.366 (.033)	.481 (.034)	**.303 (.032)**	.348 (.029)	.350 (.027)	.319 (.026)
CS_QA	.492 (.028)	.531 (.034)	.471 (.025)	.458 (.021)	.456 (.021)	**.439 (.020)**
massive_intent	.637 (.019)	.766 (.039)	**.587 (.016)**	.600 (.016)	.601 (.016)	.592 (.015)
medqa	.685 (.024)	.704 (.027)	.675 (.024)	.642 (.023)	.643 (.022)	**.621 (.022)**
metatool	.586 (.031)	.677 (.033)	.545 (.028)	.547 (.026)	.542 (.023)	**.532 (.022)**
when2call	.814 (.034)	.817 (.043)	.788 (.034)	.657 (.018)	**.655 (.017)**	.661 (.025)
wikitoxic	.487 (.044)	.474 (.041)	.429 (.037)	.458 (.026)	.451 (.027)	**.421 (.027)**
wildguard	1.138 (.038)	1.069 (.040)	1.067 (.032)	**.887 (.013)**	.891 (.011)	.892 (.017)
Average	.646 (.031)	.694 (.036)	.612 (.029)	.575 (.022)	.574 (.021)	**.560 (.022)**
			Similarity Sampling			
Amazon_CF	.401 (.033)	-	.347 (.029)	.364 (.030)	.365 (.026)	**.343 (.026)**
CS_QA	.475 (.034)	-	**.428 (.028)**	.464 (.027)	.466 (.025)	.437 (.025)
massive_intent	.271 (.020)	-	**.242 (.016)**	.268 (.015)	.270 (.016)	.256 (.015)
medqa	.610 (.027)	-	.594 (.022)	.596 (.020)	.596 (.017)	**.571 (.019)**
metatool	.244 (.030)	-	**.224 (.025)**	.238 (.026)	.236 (.026)	.230 (.024)
when2call	.763 (.044)	-	.717 (.036)	**.599 (.018)**	.601 (.020)	.610 (.025)
wikitoxic	.373 (.052)	-	.337 (.045)	.354 (.036)	.346 (.038)	**.325 (.035)**
wildguard	.668 (.028)	-	.623 (.026)	.615 (.020)	.631 (.024)	**.602 (.021)**
Average	.476 (.035)	-	.439 (.030)	.437 (.025)	.439 (.025)	**.422 (.024)**

Table 11: Brier Scores for Phi-4-mini-instruct across calibration methods. Standard deviations reported in parentheses.

Dataset	Uncalibrated	TS-AR	TS-FS	CALIBFORMER
Similarity Sampling				
Amazon_CF	**4.3 (0.8)**	7.3 (0.9)	7.3 (0.8)	6.3 (1.1)
CS_QA	6.9 (1.3)	6.7 (0.9)	6.7 (0.9)	**4.9 (1.3)**
massive_intent	7.1 (1.7)	6.8 (1.1)	7.2 (1.1)	**4.8 (1.2)**
medqa	**4.4 (0.7)**	6.3 (1.3)	6.5 (1.4)	5.6 (0.9)
metatool	7.6 (1.2)	4.4 (0.9)	4.5 (1.0)	**3.8 (0.6)**
when2call	7.2 (1.6)	5.1 (1.1)	5.5 (1.1)	**4.6 (0.9)**
wikitoxic	10.4 (2.0)	7.2 (1.2)	7.4 (1.2)	**3.4 (0.7)**
wildguard	18.4 (1.8)	9.0 (1.9)	9.7 (1.7)	**5.9 (1.4)**
Average	8.3 (1.5)	6.6 (1.2)	6.8 (1.2)	**4.9 (1.0)**
Random Sampling				
Amazon_CF	**4.9 (1.0)**	7.0 (1.0)	7.2 (0.9)	5.4 (0.9)
CS_QA	6.6 (1.4)	6.1 (1.3)	6.3 (1.4)	**5.0 (1.1)**
massive_intent	10.7 (1.4)	7.0 (1.1)	7.1 (1.3)	**4.5 (0.6)**
medqa	6.0 (1.3)	6.7 (1.5)	6.6 (1.3)	**5.0 (1.0)**
metatool	12.8 (1.5)	6.2 (1.6)	6.4 (1.5)	**3.9 (0.7)**
when2call	14.2 (1.6)	7.6 (1.5)	8.0 (1.4)	**4.7 (0.8)**
wikitoxic	10.4 (1.9)	7.0 (1.2)	6.9 (1.4)	**3.4 (0.6)**
wildguard	16.3 (2.0)	11.1 (1.7)	11.5 (1.4)	**7.7 (1.3)**
Average	10.2 (1.5)	7.3 (1.4)	7.5 (1.3)	**4.9 (0.9)**

Table 12: Calibration Error (ECE) for Qwen3.5-9B across calibration methods. Standard deviations reported in parentheses.

Dataset	Uncalibrated	TS-AR	TS-FS	CALIBFORMER
		Random Sampling		
Amazon_CF	**.149 (.025)**	.174 (.025)	.175 (.026)	.158 (.021)
CS_QA	.234 (.022)	.253 (.029)	.254 (.028)	**.234 (.018)**
massive_intent	.320 (.024)	.327 (.031)	.326 (.031)	**.306 (.025)**
medqa	**.326 (.018)**	.353 (.023)	.352 (.024)	.327 (.015)
metatool	.284 (.022)	.283 (.022)	.285 (.022)	**.264 (.019)**
when2call	.423 (.022)	.423 (.024)	.427 (.021)	**.398 (.019)**
wikitoxic	.222 (.028)	.244 (.030)	.244 (.032)	**.203 (.021)**
wildguard	.505 (.031)	.510 (.022)	.511 (.024)	**.496 (.022)**
Average	.308 (.024)	.321 (.026)	.322 (.026)	**.298 (.020)**
		Similarity Sampling		
Amazon_CF	**.155 (.016)**	.183 (.018)	.183 (.018)	.165 (.013)
CS_QA	.248 (.021)	.265 (.021)	.266 (.022)	**.247 (.016)**
massive_intent	.200 (.031)	.205 (.031)	.208 (.032)	**.196 (.028)**
medqa	**.282 (.017)**	.313 (.022)	.314 (.022)	.289 (.015)
metatool	.170 (.016)	.165 (.015)	.167 (.015)	**.157 (.013)**
when2call	.263 (.027)	.259 (.023)	.259 (.023)	**.250 (.020)**
wikitoxic	.213 (.029)	.214 (.028)	.215 (.027)	**.191 (.022)**
wildguard	.437 (.037)	.423 (.035)	.428 (.034)	**.402 (.033)**
Average	.246 (.025)	.253 (.025)	.255 (.025)	**.237 (.021)**

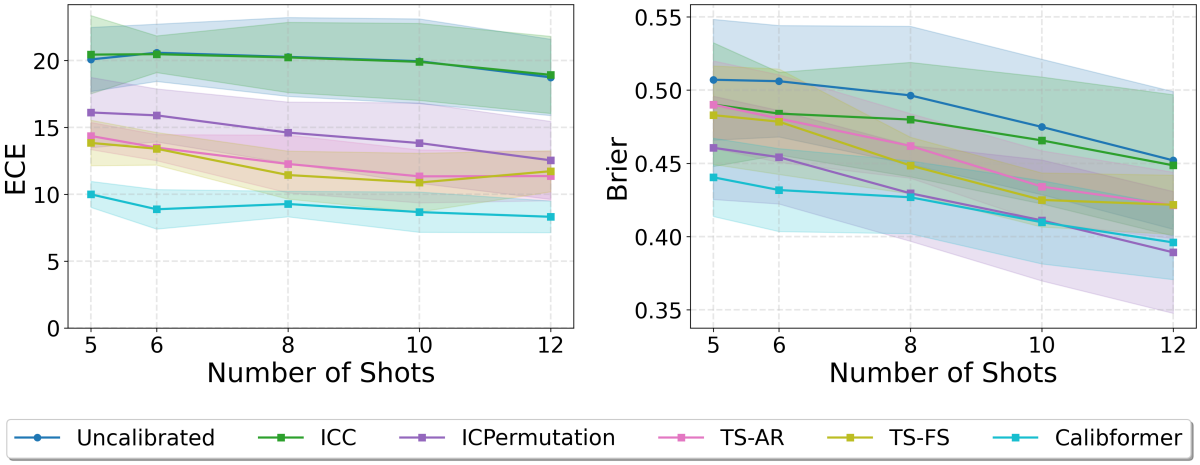
Table 13: Brier Scores for Qwen3.5-9B across calibration methods. Standard deviations reported in parentheses.

Dataset	Uncalibrated	TS-AR	TS-FS	CALIBFORMER
Random Sampling				
Amazon_CF	14.2 (1.8)	5.8 (0.8)	5.7 (1.0)	**3.4 (0.7)**
CS_QA	20.7 (2.5)	7.3 (1.6)	7.1 (1.4)	**4.9 (1.3)**
massive_intent	17.2 (2.0)	8.0 (1.3)	7.4 (0.8)	**4.1 (0.5)**
medqa	28.8 (3.0)	**7.2 (1.4)**	**7.2 (1.4)**	7.6 (2.4)
metatool	20.0 (1.8)	6.9 (1.4)	6.9 (1.6)	**4.9 (0.9)**
when2call	25.6 (1.5)	9.0 (1.6)	9.0 (1.4)	**7.9 (1.1)**
wikitoxic	15.2 (2.2)	6.6 (1.3)	6.5 (1.0)	**5.9 (1.5)**
wildguard	25.3 (2.9)	10.4 (1.9)	10.2 (1.9)	**5.4 (1.1)**
Average	20.9 (2.3)	7.6 (1.4)	7.5 (1.4)	**5.5 (1.3)**
Similarity Sampling				
Amazon_CF	18.9 (1.8)	6.8 (1.2)	**5.9 (1.1)**	**5.9 (1.3)**
CS_QA	19.3 (1.5)	7.0 (1.5)	6.8 (1.6)	**4.1 (0.7)**
massive_intent	11.3 (2.3)	8.0 (1.1)	7.7 (1.2)	**3.9 (0.6)**
medqa	24.7 (2.2)	7.7 (1.4)	7.5 (1.5)	**6.5 (1.3)**
metatool	11.6 (1.9)	5.6 (1.6)	5.3 (1.5)	**3.4 (0.8)**
when2call	17.9 (2.2)	6.3 (1.3)	6.6 (1.2)	**4.4 (0.9)**
wikitoxic	14.8 (2.1)	6.4 (1.5)	6.3 (1.4)	**5.8 (1.6)**
wildguard	25.1 (1.8)	8.0 (1.4)	8.6 (1.5)	**7.3 (1.4)**
Average	18.0 (2.0)	7.0 (1.4)	6.8 (1.4)	**5.2 (1.1)**

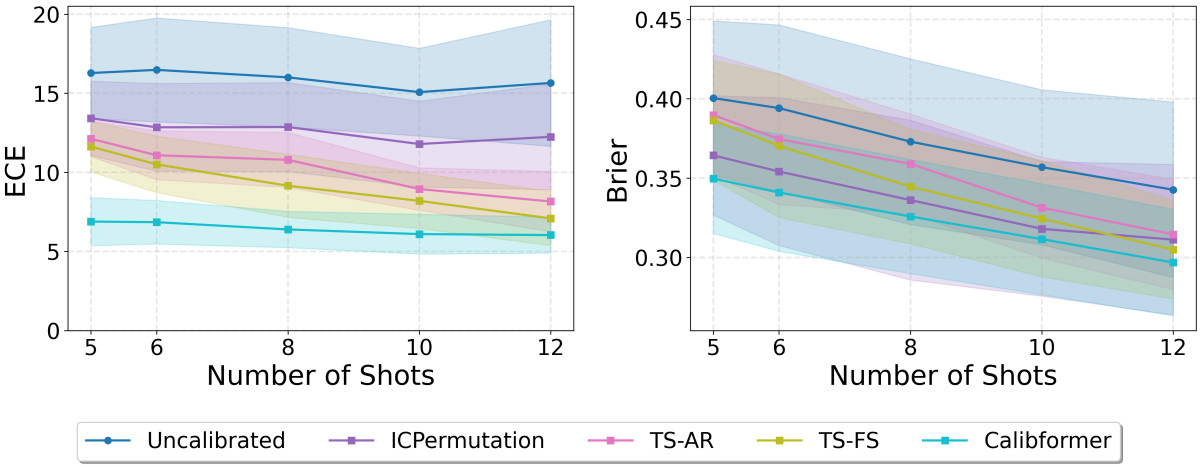
Table 14: Calibration Error (ECE) for gemma-4-E4B-it across calibration methods. Standard deviations reported in parentheses.

Dataset	Uncalibrated	TS-AR	TS-FS	CALIBFORMER
Random Sampling				
Amazon_CF	.253 (.024)	.250 (.022)	.247 (.023)	**.219 (.019)**
CS_QA	.393 (.040)	.360 (.030)	.359 (.031)	**.344 (.032)**
massive_intent	.366 (.036)	.353 (.032)	.350 (.033)	**.333 (.028)**
medqa	.624 (.051)	.532 (.036)	.533 (.034)	**.525 (.039)**
metatool	.345 (.026)	.315 (.020)	.316 (.022)	**.307 (.019)**
when2call	.490 (.023)	.440 (.018)	.438 (.018)	**.421 (.017)**
wikitoxic	.261 (.035)	**.243 (.031)**	.245 (.032)	**.228 (.030)**
wildguard	.519 (.041)	.481 (.024)	.482 (.023)	**.469 (.027)**
Average	.406 (.036)	.372 (.027)	.371 (.028)	**.356 (.027)**
Similarity Sampling				
Amazon_CF	.366 (.031)	.333 (.028)	.321 (.023)	**.300 (.024)**
CS_QA	.363 (.020)	.335 (.016)	.334 (.016)	**.317 (.015)**
massive_intent	.226 (.029)	.223 (.027)	.223 (.026)	**.206 (.022)**
medqa	.539 (.036)	.484 (.024)	.482 (.025)	**.468 (.024)**
metatool	.194 (.026)	.177 (.014)	.176 (.014)	**.169 (.015)**
when2call	.319 (.031)	.277 (.020)	.277 (.019)	**.260 (.019)**
wikitoxic	.259 (.038)	**.231 (.036)**	.232 (.036)	**.222 (.032)**
wildguard	.461 (.037)	.416 (.031)	.421 (.029)	**.404 (.031)**
Average	.341 (.032)	.309 (.025)	.308 (.024)	**.293 (.024)**

Table 15: Brier Scores for gemma-4-E4B-it across calibration methods. Standard deviations reported in parentheses.

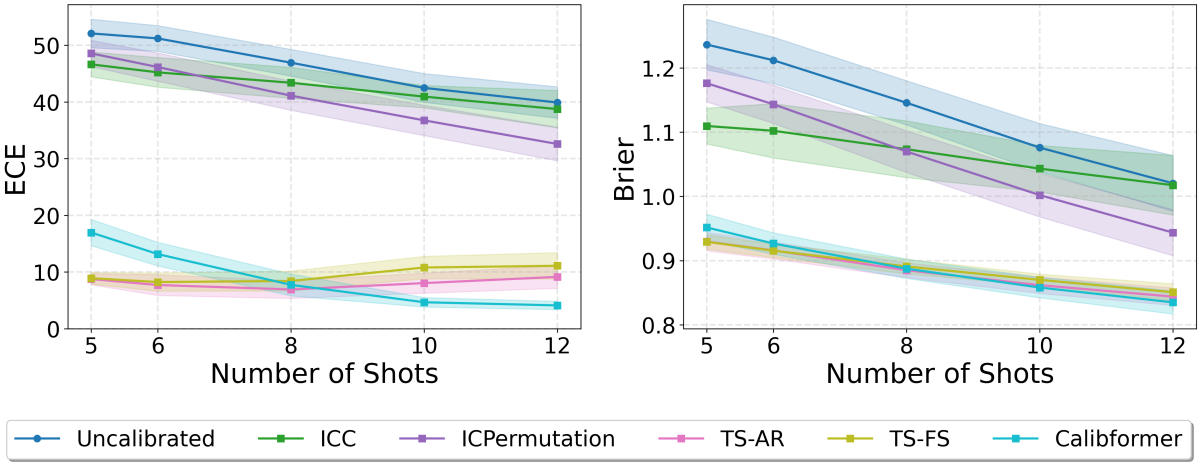


(a) Random sampling

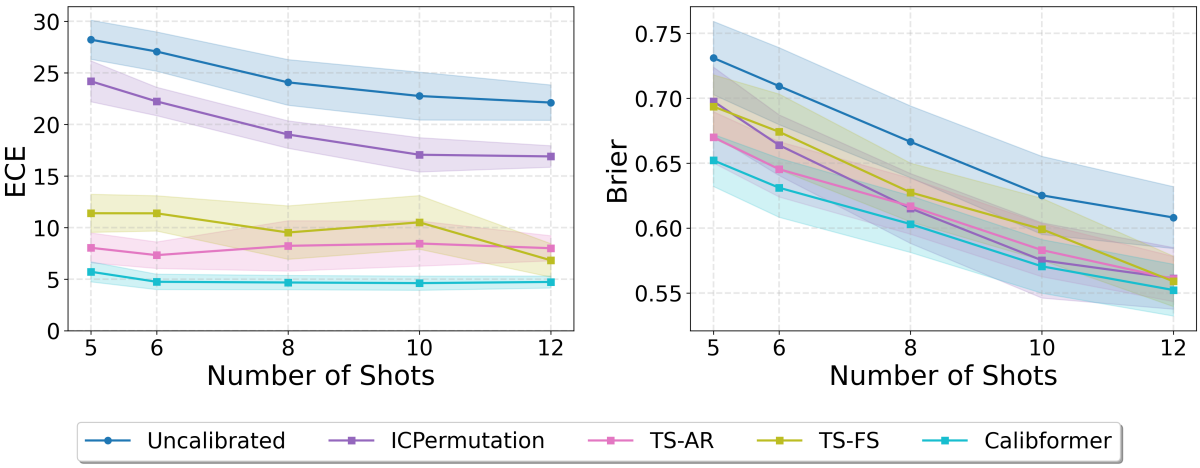


(b) Similarity sampling

Figure 5: Calibration performance | Phi 4 mini instruct | Wikitoxic

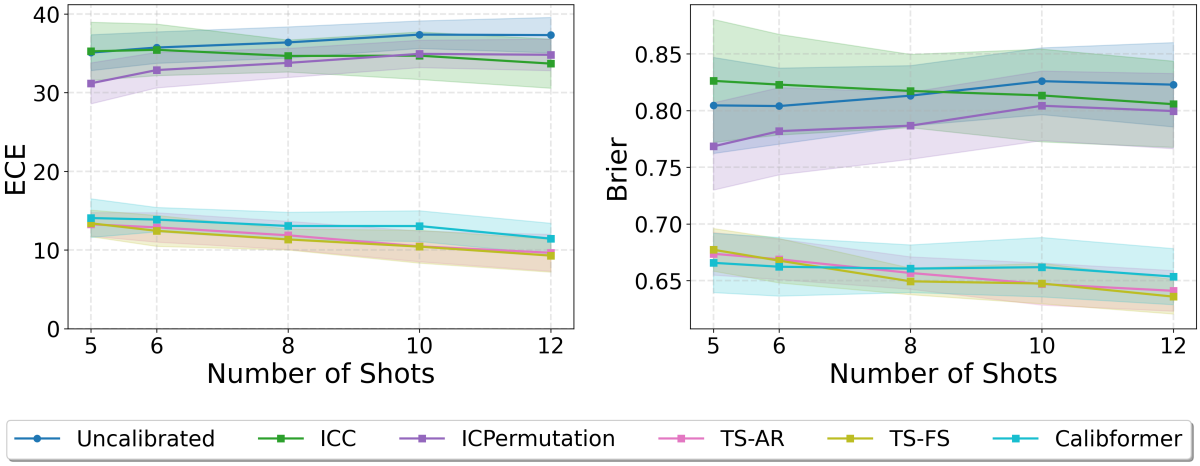


(a) Random sampling

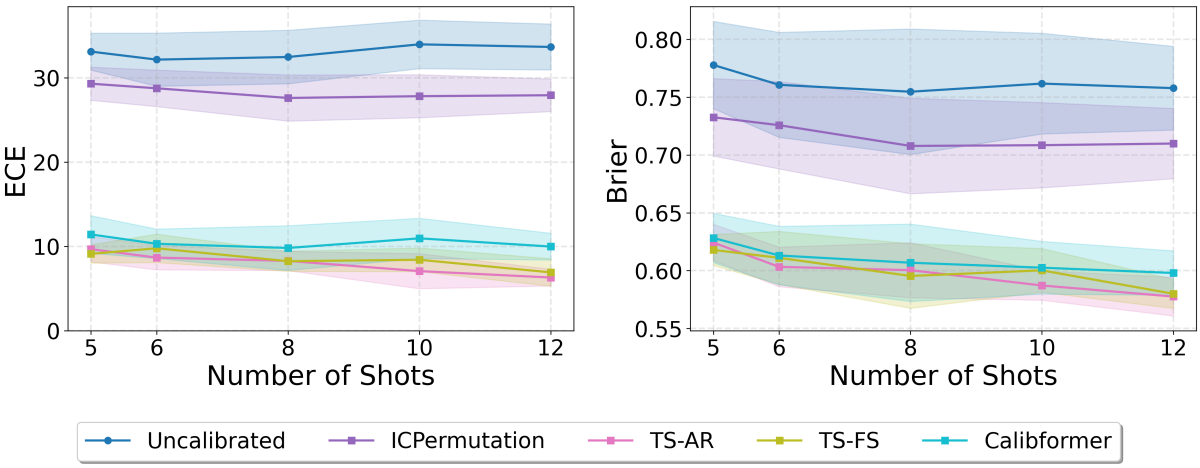


(b) Similarity sampling

Figure 6: Calibration performance | Phi 4 mini instruct | Wildguard

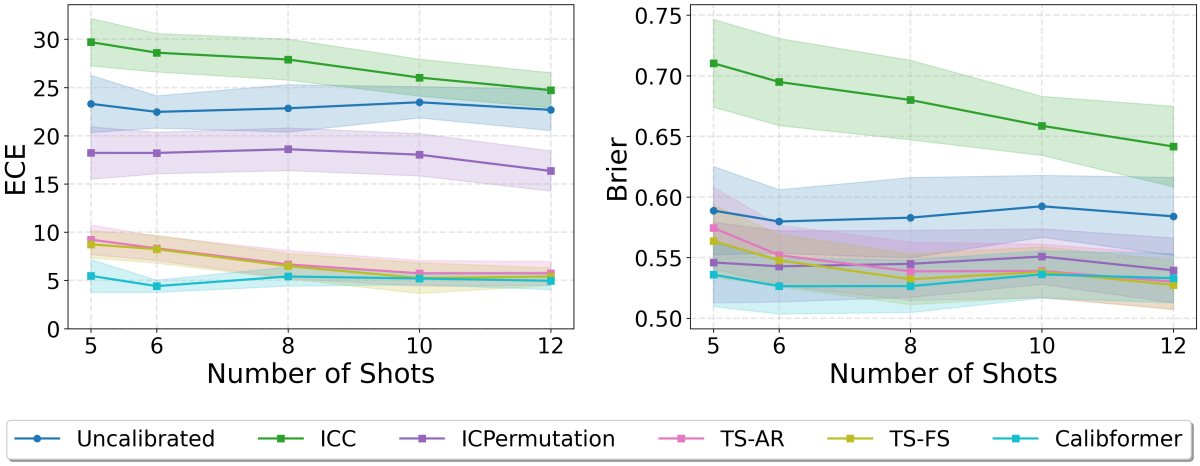


(a) Random sampling

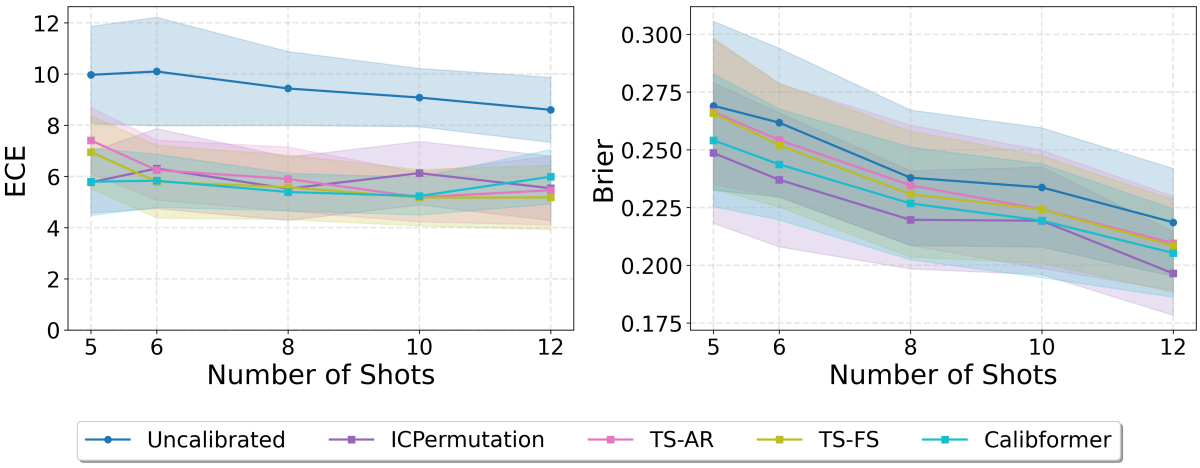


(b) Similarity sampling

Figure 7: Calibration performance | Phi 4 mini instruct | When2call

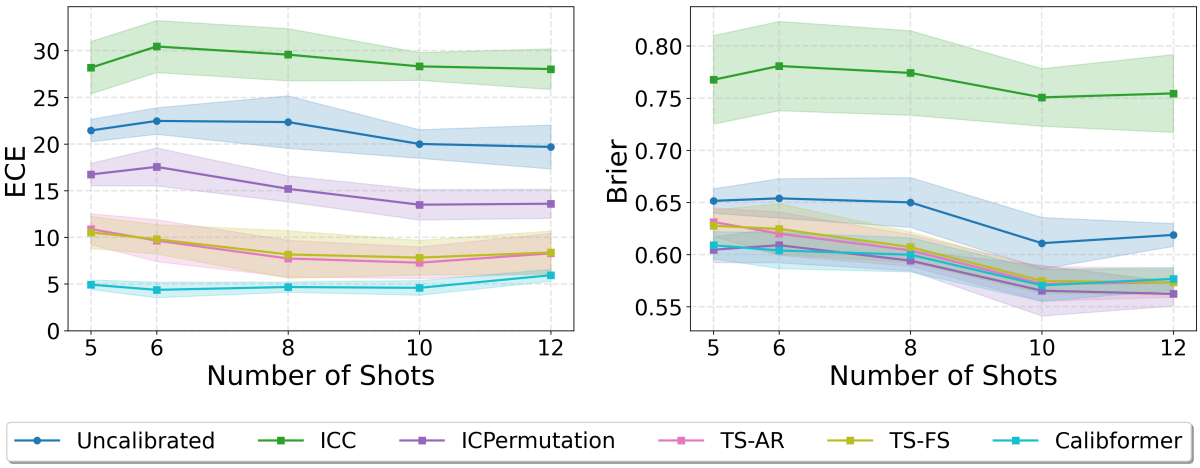


(a) Random sampling

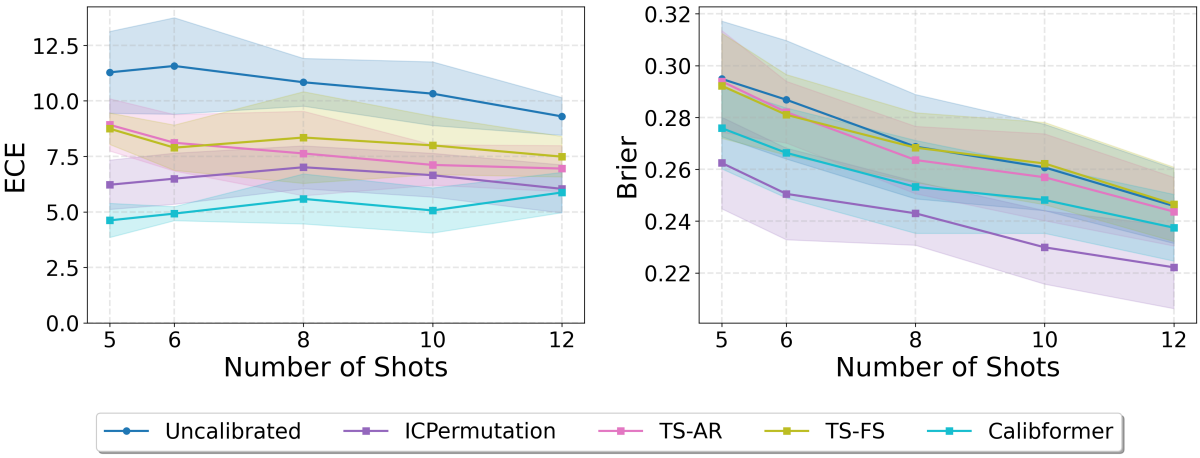


(b) Similarity sampling

Figure 8: Calibration performance | Phi 4 mini instruct | MetaTool

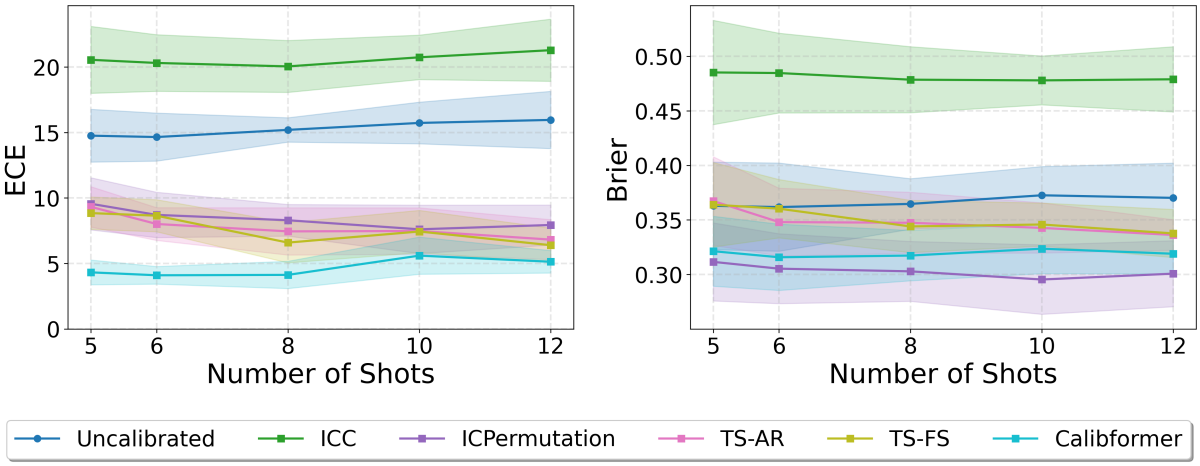


(a) Random sampling

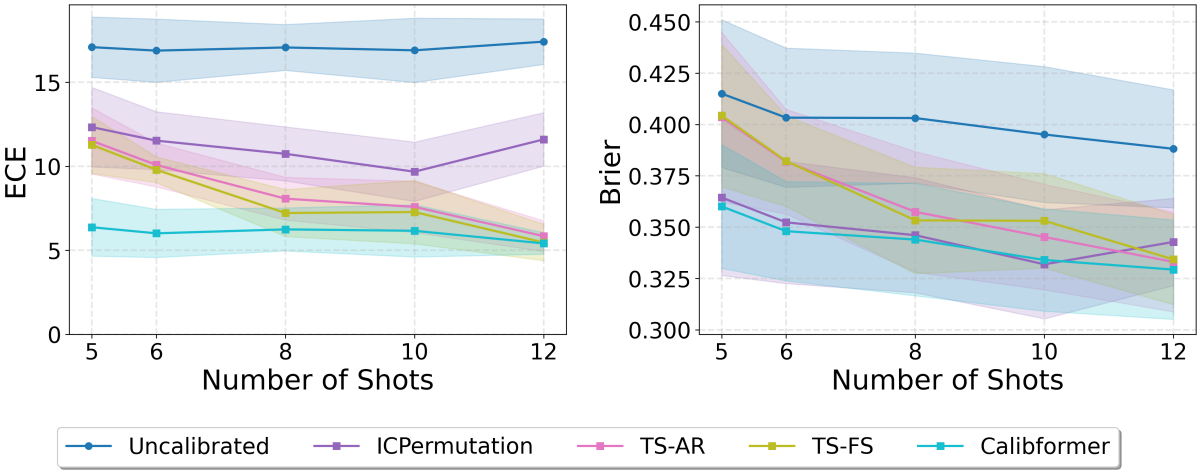


(b) Similarity sampling

Figure 9: Calibration performance | Phi 4 mini instruct | Massive Intent

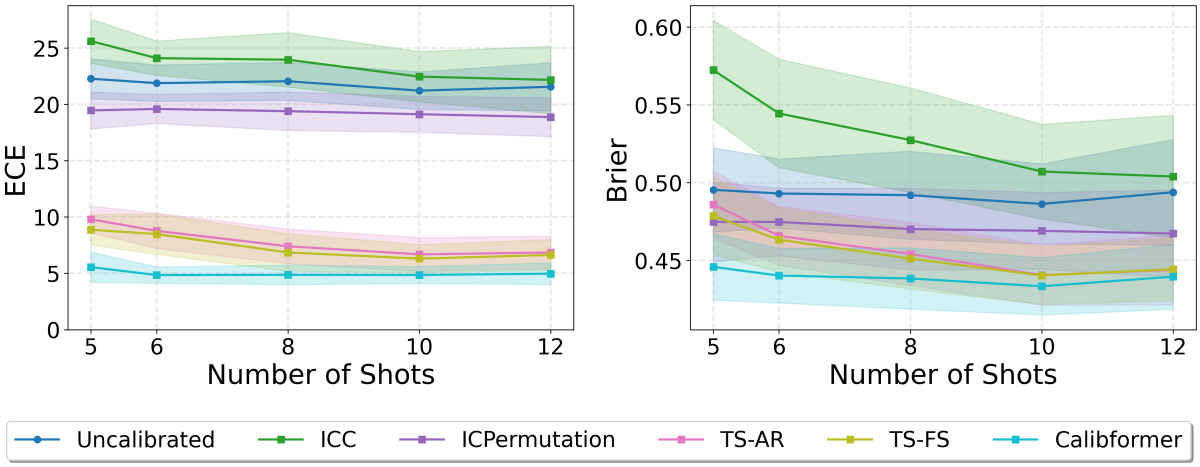


(a) Random sampling

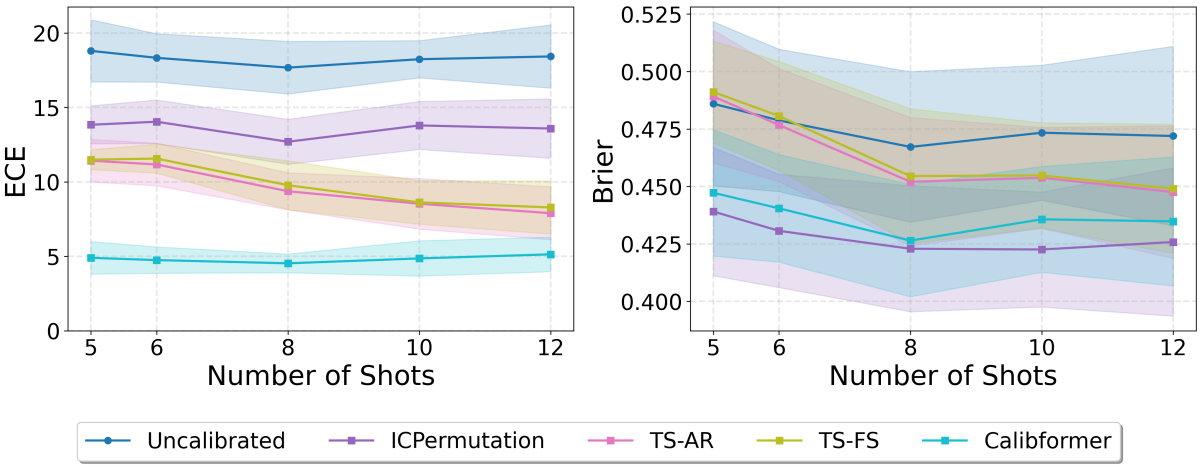


(b) Similarity sampling

Figure 10: Calibration performance | Phi 4 mini instruct | Amazon Counterfactual(Amazon_CF)



(a) Random sampling



(b) Similarity sampling

Figure 11: Calibration performance | Phi 4 mini instruct | CommonsenseQA(CS_QA)

Table 16: MedQA 1-shot prompt with chat template applied

Sampling Strategy	Prompt
Random Sampling	<|system|>You are a helpful assistant.<|end|><|user|>Your task is to answer the given medical question by specifying the correct option(A, B, C, D) Question: A 43-year-old male presents to a clinic for routine follow-up. He was diagnosed with hepatitis B several months ago. He does not have any complaints about his health, except for poor appetite. The general physical examination is normal. The laboratory investigation reveals mildly elevated aminotransferases. Which of the following findings indicate that the patient has developed a chronic form of his viral infection? A. HbsAg -, Anti-HbsAg -, Anti-HbcAg IgM +, Anti-HbcAg IgG -, HbeAg -, Anti-HbeAg + B. HbsAg +, Anti-HbsAg -, Anti-HbcAg IgM -, Anti-HbcAg IgG +, HbeAg +, Anti-HbeAg - C. HbsAg -, Anti-HbsAg +, Anti-HbcAg IgM -, Anti-HbcAg IgG -, HbeAg -, Anti-HbeAg - D. HbsAg -, Anti-HbsAg +, Anti-HbcAg IgM -, Anti-HbcAg IgG +, HbeAg -, Anti-HbeAg + <|end|><|assistant|>Answer: B <|end|><|user|>Question: A 62-year-old man presents to the physician because of incomplete healing of a chest wound. He recently had a triple coronary artery bypass graft 3 weeks ago. His past medical history is significant for type 2 diabetes mellitus and hypertension for the past 25 years. Clinical examination shows the presence of wound dehiscence in the lower 3rd of the sternal region. The wound surface shows the presence of dead necrotic tissue with pus. Computed tomography (CT) of the thorax shows a small fluid collection with fat stranding in the perisurgical soft tissues. What is the most appropriate next step in the management of the patient? A. Surgical debridement B. Negative pressure wound management C. Sternal wiring D. Sternal fixation <|end|><|assistant|>Answer:
Similarity-Based Sampling	<|system|>You are a helpful assistant.<|end|><|user|>Your task is to answer the given medical question by specifying the correct option(A, B, C, D) Question: A 78-year-old male presents to the emergency department after passing out. His wife reports that she and the patient were walking their dog when he suddenly lost consciousness. On physical exam, he has a loud crescendo-decrescendo systolic murmur and is subsequently diagnosed with severe aortic stenosis. The patient undergoes open aortic valve replacement and has an uncomplicated postoperative course. His sternal wound drain is pulled for low output on post-operative day three. On post-operative day five, the patient complains of pain during deep inspiration and retrosternal chest pain. His temperature is 101.7°F (38.7°C), blood pressure is 125/81 mmHg, pulse is 104/min, and respirations are 18/min. On physical exam, the patient is tender to palpation around his sternal wound, and there is erythema around the incision without dehiscence. His chest radiograph shows a widened mediastinum with a small pleural effusion on the left. CT angiography shows stranding in the subcutaneous tissue and a fluid collection below the sternum. Which of the following is the best next step in management? A. Surgical repair of esophageal perforation B. Surgical repair of aortic injury C. Intravenous antibiotics and observation D. Intravenous antibiotics and debridement of surgical wound <|end|><|assistant|>Answer: D <|end|><|user|>Question: A 62-year-old man presents to the physician because of incomplete healing of a chest wound. He recently had a triple coronary artery bypass graft 3 weeks ago. His past medical history is significant for type 2 diabetes mellitus and hypertension for the past 25 years. Clinical examination shows the presence of wound dehiscence in the lower 3rd of the sternal region. The wound surface shows the presence of dead necrotic tissue with pus. Computed tomography (CT) of the thorax shows a small fluid collection with fat stranding in the perisurgical soft tissues. What is the most appropriate next step in the management of the patient? A. Surgical debridement B. Negative pressure wound management C. Sternal wiring D. Sternal fixation <|end|><|assistant|>Answer:

Table 17: Amazon Counterfactual 2-shot prompts with chat template applied

Sampling Strategy	Prompt
Random Sampling	<|system|>You are a helpful assistant.<|end|><|user|>Your task is to classify the given Amazon customer review as one of the following: not-counterfactual, counterfactual User Review: The women suffered badly - the rejection and the unfaithfulness must have been hard to bear. <|end|><|assistant|>Category: not-counterfactual <|end|><|user|>User Review: But even without the seam, these are luscious stockings. <|end|><|assistant|>Category: not-counterfactual <|end|><|user|>User Review: "The lid latches could be slightly better (I've damaged a couple with repeated open/close cycles and heavy container stacking) but I have yet to find the \\\\perfect\\\\" container/lid combination among many product types and these certainly fit nearly all my requirements."" <|end|><|assistant|>Category:
Similarity-Based Sampling	<|system|>You are a helpful assistant.<|end|><|user|>Your task is to classify the given Amazon customer review as one of the following: not-counterfactual, counterfactual User Review: I am not sure the rubbermaid can handle more weight and I am a bit affraid to stack more stuffs on the top as one of the rail lightly bended when it arrived. <|end|><|assistant|>Category: not-counterfactual <|end|><|user|>User Review: They arrived with the tips in a small block of styrofoam-like material - which works well - but it would be great if there was some sort of cap (nothing fancy - even just a tiny plastic tip or something) since the block makes the very clumsy to store. <|end|><|assistant|>Category: counterfactual <|end|><|user|>User Review: "The lid latches could be slightly better (I've damaged a couple with repeated open/close cycles and heavy container stacking) but I have yet to find the \\\\perfect\\\\" container/lid combination among many product types and these certainly fit nearly all my requirements."" <|end|><|assistant|>Category:

Hyperparameter	Value
Architecture	
Embedding dim	64
Attention heads	8
Layers	12
Sequence length	21
Dropout	0.25
Optimization	
Optimizer	AdamW
Learning rate	1×10^{-5}
Weight decay (attention)	0.00
Weight decay (MLP)	0.05
Weight decay (other)	0.02
Gradient clipping	20th percentile of gradient norm history
Training	
Iterations	30,000
Batch size	16
Evaluation interval	800 iterations
Dataset dropout	0.8
Augmentation	
Temperature augmentation	True
Label augmentation	True

Table 18: Training hyperparameters for CALIBFORMER.