EffConf: Efficient Confidence Calibration for Large Language Models

Anonymous ACL submission

Abstract

Enabling large language models (LLMs) to accurately express confidence in their answers is essential for reliable decision-making. However, current LLMs often verbalize high confidence in incorrect answers, making their outputs difficult to trust in practice. Existing confidence calibration methods either fine-tune the LLM, which is costly and inapplicable to black-box models, or rely on multiple inference passes per query, incurring substantial deployment overhead. We propose **EffConf**, a family of lightweight calibrators trained on signals extracted when an LLM generates its *verbalized confidence* token. Our framework supports three input representations: the confidence logit vector and hidden state for white-box LLMs, and the confidence log-probability vector for black-box LLMs accessed through APIs. A logistic regression or bottleneck MLP then maps these representations to a calibrated confidence score. Experiments across four LLMs and five question-answering benchmarks show that our calibrators consistently outperform strong baselines, remain highly sample-efficient, and generalize well to out-of-distribution datasets, while adding only negligible inference cost.

1 Introduction

Large Language Models (LLMs) have demonstrated strong capabilities in text generation, reasoning, and decision-making, driving their adoption in high-stakes domains such as healthcare diagnostics (Qiu et al., 2024) and financial analysis (Yu et al., 2024). Despite these advances, LLMs can still generate inaccurate or fabricated responses, while often failing to express their confidence faithfully (Kapoor et al., 2024; Groot and Valdenegro-Toro, 2024; Chhikara, 2025). In such applications, these failures (*e.g.,* overconfident errors) can carry severe consequences, from life-threatening misdiagnoses to substantial financial losses.

This challenge has motivated a body of work on

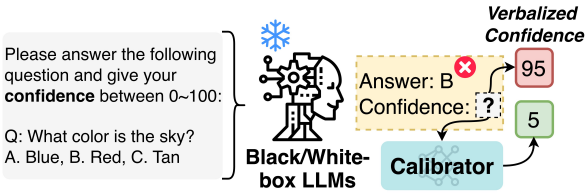

Figure 1: Verbalized confidence from LLMs is often overconfident on incorrect answers. Our efficient calibrator produces well-calibrated scores at inference time for both black/white-box LLMs.

LLM *confidence calibration*, which aims to align a model's confidence estimates with its empirical probability of correctness. Training-based methods such as SaySelf (Xu et al., 2024), LACIE (Stengel-Eskin et al., 2024), and ConfTuner (Li et al., 2025) fine-tune the LLM's parameters to produce better-calibrated verbalized confidence, while BaseCal-Proj (Tan et al., 2026) trains a projection layer that aligns the instruct LLM with its paired base LLM. Training-free alternatives, including P(True) (Kadavath et al., 2022), Ensemble (Xiong et al., 2024), and BaseCal-ReEval (Tan et al., 2026), avoid parameter access but require multiple inference passes per query and generally underperform training-based methods.

However, a practical calibrator should also have two desirable properties: (1) it should be efficient, adding negligible cost at inference and avoiding expensive modifications to the underlying LLM; (2) it should be applicable to both white-box and black-box LLMs, since many of the most capable models are accessible only through restricted APIs that expose top-ranked log-probabilities but not internal states. Existing approaches fall short of jointly meeting these requirements.

To close this gap, we propose **EffConf**, a family of lightweight calibrators trained on signals associated with the LLM's confidence token generation step. Our calibrator takes one of three input representations: the *confidence logit vector* (white-box)

or the *log-probability vector* (black-box), both of which capture the probability distribution over candidate confidence tokens, or the *hidden state* (white-box) used to predict the confidence token. The calibrator (either a logistic regression or a bottleneck MLP) then produces a calibrated confidence score. Across all variants, calibration adds only marginal overhead beyond the standard LLM inference. Our experiments (Section 5) further show that these calibrators are highly sample-efficient and generalize well to out-of-distribution datasets.

Our main contributions are as follows:

- We propose **EffConf**, a family of lightweight calibrators that improve LLM verbalized confidence without modifying the underlying LLM. By calibrating signals extracted at the confidence token generation step, EffConf applies to both white-box LLMs (via logits or hidden states) and black-box LLMs (via log-probabilities).
- We conduct extensive experiments across four LLMs and five benchmarks, showing that EffConf consistently outperforms strong baselines, generalizes well to out-of-distribution datasets, and remains highly sample-efficient, achieving effective calibration with only a few hundred labeled examples.
- We analyze the representations used by EffConf and show that the confidence token generation step contains important signals. This finding explains why lightweight calibrators can effectively calibrate verbalized confidence without fine-tuning the LLM itself.

2 Related Work

LLMs often produce incorrect or hallucinated content while reporting high confidence (Ji et al., 2023; Groot and Valdenegro-Toro, 2024; Chhikara, 2025). Such overconfidence undermines user trust and hinders the safe deployment of LLMs, motivating a growing body of work on *confidence calibration*, which aims to align a model's confidence estimates with its actual probability of correctness. Existing calibration methods for LLMs can be broadly divided into two categories: training-free and training-based.

Training-free methods typically require additional inference passes or multiple sampled responses to calibrate the model's confidence. Early work by Kadavath et al. (2022) prompts the LLM to assess its own answer in a True/False format and uses the probability assigned to the "True" token as an estimate of correctness, known as P(True). Xiong et al. (2024) propose a framework that combines different prompting, sampling, and aggregation strategies, where confidence is estimated from the consistency of multiple sampled responses to the same query. Observing that base LLMs are often well calibrated, BaseCal-ReEval (Tan et al., 2026) first obtains a response from the instruct LLM, then re-feeds the same input sequence through the base LLM and uses the probabilities the base model assigns to the instruct model's response tokens as the confidence score.

Recently, several training-based approaches directly fine-tune LLMs for better calibration. Xu et al. (2024) propose SaySelf, which trains models to generate fine-grained confidence estimates alongside self-reflective rationales, using reinforcement learning to penalize overconfidence on incorrect answers. Stengel-Eskin et al. (2024) develop LACIE, which adopts a speaker–listener framework and fine-tunes the speaker on preference data with direct preference optimization to produce calibrated confidence. More recently, Li et al. (2025) introduce ConfTuner, which fine-tunes LLMs with a tokenized Brier score, a proper scoring rule that incentivizes the model to output confidence tokens reflecting the true likelihood of correctness. They report improvements over both SaySelf and LACIE. Rather than fine-tuning the LLM itself, BaseCal-Proj (Tan et al., 2026) learns a projection layer that maps the instruct LLM's final-layer hidden states into the base LLM's space, then passes the projected states through the base LLM's output layer to obtain the final probability.

However, none of these approaches is both efficient and applicable to black-box LLMs. Training-free methods often require additional inference passes per query, which increases deployment cost. Training-based methods require fine-tuning or access to model parameters, making them incompatible with black-box settings, where only API access (*i.e.,* top-ranked log-probabilities) is available. In this paper, we propose **EffConf**, a family of lightweight calibrators that use signals extracted at the verbalized confidence generation step. Our method supports both white-box and black-box LLMs while adding negligible cost beyond standard LLM inference.

3 Methodology

3.1 Preliminaries

Given an input question, we prompt the LLM to output its answer followed by a verbalized confidence score, *e.g.,* 'Confidence: 96'. This output is a token sequence consisting of a fixed prefix (*i.e.,* 'Confidence: ') followed by one token drawn from a predefined *confidence token set* $\mathcal{C}_N := \{c_0, c_1, \ldots, c_N\}$, where c_i denotes the token representing the integer confidence value i. For example, on a 0–100 scale, $\mathcal{C}_{100} = \{c_0, c_1, \ldots, c_{100}\}$ contains 101 tokens. At the confidence token generation step, the LLM produces a full logit vector $\mathbf{z} \in \mathbb{R}^{|\mathcal{V}|}$ assigning one logit to each token in its vocabulary $\mathcal{V}$.

3.2 Motivation

Our work is motivated by recent training-based calibration methods (Section 2), such as SaySelf, LACIE, and ConfTuner, which fine-tune an LLM to produce better-calibrated *verbalized confidence*. Their success suggests that calibration-relevant information may already be present in the model's internal state during confidence generation, and that fine-tuning helps reshape the output to better expose this information. However, these methods require modifying the underlying LLM, which incurs additional training cost and risks altering the model's other behaviors. They are also difficult to apply to black-box LLMs, where users can access only API outputs and limited token-level probabilities, but not model parameters or activations.

Building on this observation, rather than fine-tuning the LLM to reshape its verbalized confidence, we keep the underlying LLM frozen and train a lightweight calibrator on the signals associated with the confidence token generation step. Because these signals are obtained at inference time, EffConf naturally supports both white-box and black-box settings, depending on which representation is available (Section 3.3).

3.3 Input Representations

Our calibration framework (**EffConf**) supports three complementary input representations, covering both white-box and black-box LLMs.

Confidence logit vector. For white-box models, $\mathbf{z}$ is fully accessible, and extracting the entries indexed by $\mathcal{C}_N$ yields the *confidence logit vector*:

$$\mathbf{l} = [l_0, l_1, \ldots, l_N]^\top \in \mathbb{R}^{N+1}, \tag{1}$$

where l_i is the logit of confidence token c_i.

Hidden-state representation. For white-box models, we extract the last-layer hidden state $\mathbf{h} \in \mathbb{R}^{d_{\text{model}}}$ at the token position immediately preceding the confidence token, where d_{model} is the hidden dimension of the model. This representation reflects the model's internal state used to predict the confidence token and provides a richer signal than $\mathbf{l}$, since it retains information that may be lost when projecting to the vocabulary space.

Confidence log-probability vector. For black-box models accessible only through APIs, most providers expose log-probabilities for the top-ranked tokens at each generation step. We extract the log-probabilities corresponding to $\mathcal{C}_N$ from this list to form the *confidence log-probability vector* $\mathbf{p} \in \mathbb{R}^{N+1}$, and set the decoding temperature to 1 to avoid distorting the underlying probability distribution.

3.4 Problem Formulation

Let $\mathbf{x} \in \mathbb{R}^{d_{\text{in}}}$ denote a generic input representation, instantiated as $\mathbf{l}$, $\mathbf{h}$, or $\mathbf{p}$ depending on the setting (Section 3.3). Let

$$\mathcal{D} = \{(\mathbf{x}_m,\ y_m)\}_{m=1}^{M} \tag{2}$$

denote a calibration dataset of M samples, where $y_m \in \{0, 1\}$ indicates whether the LLM's answer is correct ($y_m = 1$) or incorrect ($y_m = 0$). Our goal is to learn a function $g_\theta \colon \mathbb{R}^{d_{\text{in}}} \to [0, 1]$ such that

$$g_\theta(\mathbf{x}) \approx P(y = 1 \mid \mathbf{x}). \tag{3}$$

That is, given an input representation $\mathbf{x}$, the calibrator outputs the estimated probability that the LLM's answer is correct. This recasts confidence calibration as a binary probabilistic prediction problem, where the predicted probability directly serves as the calibrated confidence score. We investigate two model families for g_θ, offering different trade-offs between expressiveness and sample efficiency.

3.5 Efficient Confidence Calibrator

Both calibrator architectures described below are applied to each of the three input representations introduced in Section 3.3, yielding six calibrator configurations in total.

Logistic Calibrator. We apply logistic regression to map the input representation to a correctness probability through a linear transformation:

$$g_\theta(\mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x} + b), \tag{4}$$

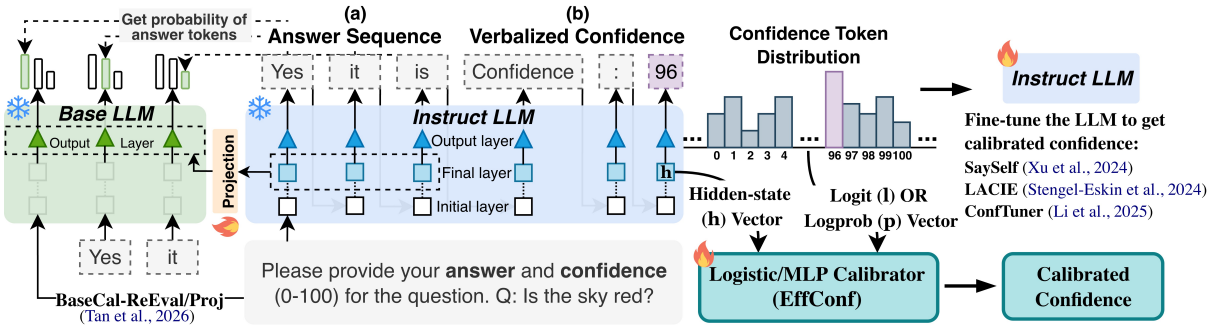

Figure 2: Overview of our calibration framework and its comparison with existing methods. BaseCal-ReEval and BaseCal-Proj estimate confidence from *answer sequence* probabilities produced by a frozen base LLM, while fine-tuning-based methods such as SaySelf, LACIE, and ConfTuner tune the instruct LLM to produce better-calibrated *verbalized confidence*. In contrast, our method (**EffConf**) keeps the instruct LLM frozen and calibrates its verbalized confidence using signals from the confidence token (*e.g.,* '96' here) generation step.

where $\mathbf{w} \in \mathbb{R}^{d_{\text{in}}}$ is a weight vector, $b \in \mathbb{R}$ is a bias term, and $\sigma(\cdot)$ denotes the sigmoid function.

Bottleneck MLP Calibrator. To capture non-linear interactions among input features, we use a single-hidden-layer network with a bottleneck architecture:

$$g_\theta(\mathbf{x}) = \sigma\big(\mathbf{w}_2^\top \operatorname{ReLU}(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1) + b_2\big), \quad (5)$$

where $\mathbf{W}_1 \in \mathbb{R}^{d \times d_{\text{in}}}$ projects the input into a d-dimensional bottleneck (with $d \ll d_{\text{in}}$), $\mathbf{b}_1 \in \mathbb{R}^d$ is the hidden bias, $\mathbf{w}_2 \in \mathbb{R}^d$ maps the hidden layer to a scalar, and $b_2 \in \mathbb{R}$ is the output bias.

Training Objective. Both calibrators are trained with the binary cross-entropy (BCE) loss:

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{M}\sum_{m=1}^{M}\Big[y_m \log g_\theta(\mathbf{x}_m) + (1 - y_m)\log\big(1 - g_\theta(\mathbf{x}_m)\big)\Big]. \quad (6)$$

BCE is a strictly proper scoring rule: its theoretical minimizer coincides with the true conditional probability $P(y = 1 \mid \mathbf{x})$. This property ensures that, given sufficient data and model capacity, the learned output directly provides well-calibrated confidence estimates without requiring post-hoc adjustments. Training details (*e.g.,* dataset construction and optimization hyperparameters) are provided in Appendix B.

4 Experimental Setup

4.1 Models and Calibrator Configurations

We evaluate our calibrators on three white-box LLMs: Llama-3.1-8B-Instruct (Grattafiori et al., 2024), Qwen2.5-7B-Instruct (Yang et al., 2025), and Ministral-8B-Instruct-2410 (Mistral AI, 2024), as well as one black-box LLM, GPT-4o-Mini (Menick et al., 2024). For brevity, we refer to these as Llama, Qwen, Ministral, and GPT throughout the paper.

For the white-box models, we train calibrators on both the confidence logit vector $\mathbf{l}$ and the hidden-state representation $\mathbf{h}$. For GPT, where only API access is available, we use the confidence log-probability vector $\mathbf{p}$. Llama uses the confidence token set $\mathcal{C}_{100}$ (a 0–100 scale), while Qwen, Ministral, and GPT use $\mathcal{C}_9$ (a 0–9 scale).

For the bottleneck MLP calibrator, we set the bottleneck dimension d according to the input dimension d_{in} to keep $d \ll d_{\text{in}}$ in all configurations. Specifically, $d = 8$ when $d_{\text{in}} = 101$ (Llama with $\mathbf{l}$); $d = 4$ when $d_{\text{in}} = 10$ (Qwen, Ministral, and GPT with $\mathbf{l}$ or $\mathbf{p}$); and $d = 128$ when d_{in} equals the model's hidden dimension (white-box models with $\mathbf{h}$). This narrow bottleneck encourages the hidden layer to learn a compact summary of the input rather than overfit to individual training examples. We further study the effect of the bottleneck dimension d in Appendix D.3.

4.2 Baselines

We compare against six baselines: (1) **Verbalization**: prompts the unmodified instruct LLM to verbalize its confidence as a numerical value; (2) **Sequence Probability (Seq Prob)**: exponentiates the average log-probability of the answer tokens generated by LLM, yielding a length-normalized confidence; (3) **P(True)** (Kadavath et al., 2022): prompts the model to self-assess its previous output and uses the predicted probability of the to-

ken "True" as the confidence score; (4) **Ensemble** (Xiong et al., 2024): a black-box method that prompts the LLM multiple times (three times in our experiments) to generate answers with verbalized confidence, then aggregates them via the "Avg-Conf" strategy; (5) **ConfTuner** (Li et al., 2025): LoRA fine-tuning with the Tokenized Brier Score loss. We additionally evaluate a full-parameter fine-tuning (FPFT) variant; (6) **BaseCal** (Tan et al., 2026), with two variants: **BaseCal-ReEval** feeds the instruct LLM's response back into its corresponding base LLM and uses the base LLM's average token probability over the response as the confidence, while **BaseCal-Proj** estimates this base LLM confidence with a trained linear projection.

We use greedy decoding for all methods except Ensemble, which requires sampling multiple responses by design. All baselines apply to Llama and Qwen. On Ministral and GPT, BaseCal is not applicable because neither model has a publicly available paired base LLM. ConfTuner is also not applicable to GPT because it requires white-box access for fine-tuning.

4.3 Datasets

Following ConfTuner (Li et al., 2025), we use HotpotQA (Yang et al., 2018) as the default training set for our calibrators. For the baselines, we retrain ConfTuner and BaseCal-Proj on HotpotQA using the training configurations from their original papers. The default training sizes are 1,000 examples for our method, 2,000 for ConfTuner, and 10,000 for BaseCal-Proj (results shown in Table 1).

To study the effects of training set size and domain shift, we additionally train our calibrators, ConfTuner, and BaseCal-Proj on both HotpotQA and TriviaQA, varying the training size from 100 to 10,000 examples (see Section 5).

For evaluation, we use the **HotpotQA** test set together with four additional benchmarks: (1) **TriviaQA** (Joshi et al., 2017), an open-domain trivia QA dataset; (2) **StrategyQA** (Geva et al., 2021), where the reasoning steps required to answer a question are implicit and must be inferred; (3) **GSM8K** (Cobbe et al., 2021), a collection of linguistically diverse grade-school math word problems; and (4) **TruthfulQA** (Lin et al., 2022), which tests whether models can resist common misconceptions and maintain factual accuracy. Dataset details and statistics are provided in Appendix A.

4.4 Evaluation Metrics

Following prior work (Xu et al., 2024; Stengel-Eskin et al., 2024; Li et al., 2025), we evaluate confidence quality using two complementary metrics: Expected Calibration Error (ECE) (Naeini et al., 2015) and Area Under the ROC Curve (AUROC) (Boyd et al., 2013).

ECE measures the average absolute difference between predicted confidence and empirical accuracy across confidence bins. We partition the confidence range into K equal-width bins $\{B_1, \ldots, B_K\}$, and for each bin B_k compute the average accuracy $\mathrm{acc}(B_k)$ and the average predicted confidence $\mathrm{conf}(B_k)$ over the samples falling into it. ECE is then calculated as:

$$\mathrm{ECE} = \sum_{k=1}^{K} \frac{|B_k|}{n} \left|\mathrm{acc}(B_k) - \mathrm{conf}(B_k)\right|, \quad (7)$$

where $|B_k|$ is the number of samples in bin B_k and n is the total number of evaluation samples. Lower ECE indicates better calibration. We set $K = 10$ with confidence scores normalized in $[0, 1]$.

AUROC quantifies how well the confidence score separates correct from incorrect answers, capturing whether correct predictions systematically receive higher confidence than errors. We compute it as the area under the ROC curve:

$$\mathrm{AUROC} = \int_0^1 \mathrm{TPR}(t)\, d\mathrm{FPR}(t), \quad (8)$$

where $\mathrm{TPR}(t)$ and $\mathrm{FPR}(t)$ are the true and false positive rates at confidence threshold $t \in [0, 1]$.

5 Results and Discussion

We evaluate EffConf from four perspectives: overall effectiveness against existing calibration methods, sample efficiency, robustness to training-distribution shift, and the signals captured by its input representations.

Effectiveness against existing calibration methods. Tables 1 and 2 report ECE and AUROC across the five benchmarks for all white-box LLMs and GPT, respectively. We discuss the results along three dimensions.

(i) In-distribution calibration. On HotpotQA, our calibrators achieve the lowest in-distribution ECE on three of the four LLMs: Llama (0.0270), Qwen (0.1093), and GPT (0.0373). The exception is Ministral, where Ensemble obtains the lowest

LLM	Method	In-distribution		Out-of-distribution (OOD)								OOD Average	
		HotpotQA		GSM8K		TriviaQA		StrategyQA		TruthfulQA			
		ECE ↓	AUROC ↑	ECE ↓	AUROC ↑	ECE ↓	AUROC ↑	ECE ↓	AUROC ↑	ECE ↓	AUROC ↑	ECE ↓	AUROC ↑
Llama	Verbalization	0.3497	0.6712	0.1622	0.5301	0.2358	0.6140	0.1443	0.6313	0.4621	0.6146	0.2511	0.5975
	Seq Prob	0.3459	0.6158	0.0384	0.7372	0.1438	0.5351	0.0339	0.5888	0.6479	0.8067	0.2160	0.6670
	P(True)	0.3837	0.6061	0.1202	0.5067	0.1933	0.6540	0.4019	0.4704	0.3729	0.6434	0.2721	0.5686
	Ensemble	0.1683	0.4701	0.0352	0.8720	0.1973	0.6812	0.2009	0.6272	0.4321	0.6666	0.2164	**0.7118**
	ConfTuner	0.0481	**0.8130**	0.0398	0.6395	0.1009	0.6423	0.0839	0.7641	0.3261	0.6651	0.1377	0.6778
	ConfTuner (FPFT)	0.0919	0.7665	0.0901	0.6628	0.0951	0.6143	0.0917	0.7352	0.3210	0.6662	0.1495	0.6696
	BaseCal-ReEval	0.3003	0.5380	0.0268	0.6112	0.0741	0.4813	0.1295	0.5540	0.4482	0.7681	0.1697	0.6037
	BaseCal-Proj	0.0873	0.5727	0.3781	0.6294	0.2033	0.5158	0.3273	0.5257	0.2127	0.6333	0.2804	0.5761
	Logistic (w/ logit)	0.0470	0.7316	0.0863	0.7278	0.1157	0.6949	0.2061	0.6345	0.0908	0.6949	0.1247	0.6880
	MLP (w/ logit)	0.0342	0.7100	0.0874	0.7400	0.0912	0.7046	0.2862	0.6402	0.0773	0.7278	0.1355	0.7032
	Logistic (w/ hs)	**0.0270**	0.7635	0.1135	0.7578	0.2014	0.6960	0.0985	0.6670	0.1971	0.6466	0.1526	0.6919
	MLP (w/ hs)	0.0303	0.7503	0.0718	0.7559	0.0880	0.6825	0.1726	0.6500	0.1066	0.6489	**0.1098**	0.6843
Qwen	Verbalization	0.5472	0.6535	0.1372	0.5144	0.4767	0.6700	0.2454	0.6694	0.4915	0.6439	0.3377	0.6244
	Seq Prob	0.5291	0.6949	0.0451	0.7054	0.4432	0.7397	0.1484	0.6270	0.5477	0.8236	0.2961	0.7239
	P(True)	0.4105	0.6601	0.8140	0.5194	0.2866	0.7757	0.6685	0.4524	0.3322	0.6358	0.5253	0.5958
	Ensemble	0.1299	0.5848	0.0360	0.8278	0.3464	0.7315	0.2511	0.6011	0.4828	0.5892	0.2791	0.6874
	ConfTuner	0.2272	**0.7938**	0.1482	0.5095	0.4544	0.6772	0.2658	0.6849	0.4922	0.6138	0.3402	0.6214
	ConfTuner (FPFT)	0.4204	0.7294	0.1408	0.5310	0.4574	0.6745	0.2470	0.6828	0.4964	0.6307	0.3354	0.6298
	BaseCal-ReEval	0.4423	0.6361	0.0338	0.7206	0.3348	0.6463	0.0324	0.6548	0.4643	0.8353	0.2163	0.7143
	BaseCal-Proj	0.2034	0.6148	0.3724	0.5756	0.0672	0.5668	0.3847	0.5757	0.3566	0.5913	0.2952	0.5774
	Logistic (w/ logit)	0.1182	0.7245	0.2778	0.3729	0.2622	0.7110	0.1553	0.7060	0.2447	0.6156	0.2350	0.6014
	MLP (w/ logit)	0.1406	0.7319	0.1915	0.7893	0.2164	0.7610	0.1222	0.7225	0.1082	0.6832	0.1596	0.7390
	Logistic (w/ hs)	0.1142	0.7828	0.4658	0.7549	0.1323	0.8136	0.1299	0.7166	0.1478	0.4888	0.2190	0.6935
	MLP (w/ hs)	**0.1093**	0.7937	0.2717	0.7969	0.1997	0.8210	0.0876	0.7459	0.0608	0.7034	**0.1550**	**0.7668**
Ministral	Verbalization	0.6193	0.5474	0.2828	0.5104	0.4411	0.5381	0.2761	0.5386	0.6798	0.5260	0.4200	0.5283
	Seq Prob	0.3843	0.6857	0.1561	0.7303	0.2542	0.6529	0.0198	0.5931	0.6123	0.7415	0.2606	0.6795
	P(True)	0.0627	0.6690	0.4141	0.5681	0.0573	0.7316	0.3705	0.4901	0.0926	0.7979	0.2336	0.6469
	Ensemble	**0.0255**	0.5137	0.0717	0.8465	0.2353	0.7422	0.2328	0.5629	0.2909	0.7644	0.2077	0.7290
	ConfTuner	0.1042	**0.8230**	0.1934	0.6650	0.2334	0.7449	0.2079	0.6056	0.4882	0.7336	0.2807	0.6873
	ConfTuner (FPFT)	0.1850	0.8163	0.0768	0.7283	0.2454	0.7413	0.2260	0.5720	0.4481	0.7532	0.2491	0.6987
	Logistic (w/ logit)	0.1179	0.6755	0.2469	0.5844	0.1035	0.7211	0.1380	0.6188	0.0846	0.7608	0.1433	0.6713
	MLP (w/ logit)	0.1245	0.6844	0.2509	0.6922	0.1004	0.7369	0.1694	0.6390	0.1258	0.7821	0.1616	0.7126
	Logistic (w/ hs)	0.1054	0.7713	0.2493	0.7357	0.1840	0.7673	0.1480	0.5967	0.4090	0.8517	0.2476	**0.7379**
	MLP (w/ hs)	0.1255	0.7472	0.1000	0.7594	0.0902	0.7547	0.1076	0.6231	0.1765	0.8046	**0.1186**	0.7355

Table 1: ECE (↓) and AUROC (↑) of different calibrators across three white-box LLMs. **Bold** indicates the best result, while underline denotes the second best. Logistic and MLP denote our **EffConf** calibrators trained on either confidence logits (w/ logit) or hidden states (w/ hs).

LLM	Method	In-distribution		Out-of-distribution (OOD)								OOD Average	
		HotpotQA		GSM8K		TriviaQA		StrategyQA		TruthfulQA			
		ECE ↓	AUROC ↑	ECE ↓	AUROC ↑	ECE ↓	AUROC ↑	ECE ↓	AUROC ↑	ECE ↓	AUROC ↑	ECE ↓	AUROC ↑
GPT	Verbalization	0.3834	0.7362	0.1034	0.5791	0.3077	0.6656	0.1877	0.7148	0.3688	0.7078	0.2419	0.6668
	Seq Prob	0.2573	0.6174	0.1113	0.7120	0.0989	0.5763	0.0823	0.6402	0.4639	0.8152	0.1891	0.6859
	P(True)	0.3047	0.7655	0.6669	0.4726	0.2765	0.6769	0.1613	0.7294	0.2343	0.8342	0.3348	0.6783
	Ensemble	0.1551	0.4617	0.0381	0.7769	0.2145	0.6615	0.2264	0.6185	0.3805	0.6447	0.2149	0.6754
	Logistic (w/ logprob)	**0.0373**	0.7644	0.0189	0.8172	0.1154	0.7231	0.1628	0.7523	0.2322	0.7020	0.1323	0.7487
	MLP (w/ logprob)	0.0387	**0.7683**	0.0497	0.8203	0.0935	0.7321	0.1640	0.7421	0.0963	0.7318	**0.1009**	**0.7566**

Table 2: ECE (↓) and AUROC (↑) of different calibrators for GPT. **Bold** indicates the best result, while underline denotes the second best. Logistic and MLP denote our **EffConf** calibrators trained on confidence log-probabilities (w/ logprob).

ECE. AUROC shows a more mixed pattern: ConfTuner achieves stronger in-distribution AUROC than our best variant on Llama (0.8130 vs. 0.7635) and Ministral (0.8230 vs. 0.7713). This is expected, since ConfTuner fine-tunes the LLM end-to-end on HotpotQA, whereas our calibrators keep the LLM frozen.

(ii) Out-of-distribution generalization. Averaged over the four out-of-distribution (OOD) benchmarks, our best calibrator reduces ECE relative to the strongest baseline by 20%, 28%, 43%, and 47% on Llama, Qwen, Ministral, and GPT, respectively. The gain is especially clear against ConfTuner: despite its strong in-distribution performance, its OOD ECE on Qwen (0.3402) and Ministral (0.2807) is more than twice that of our MLP (w/ hs) calibrator, suggesting that fine-tuning on a single dataset may overfit to its distribution.

(iii) Effect of our input representations and calibrator architectures. Among the white-box LLMs,

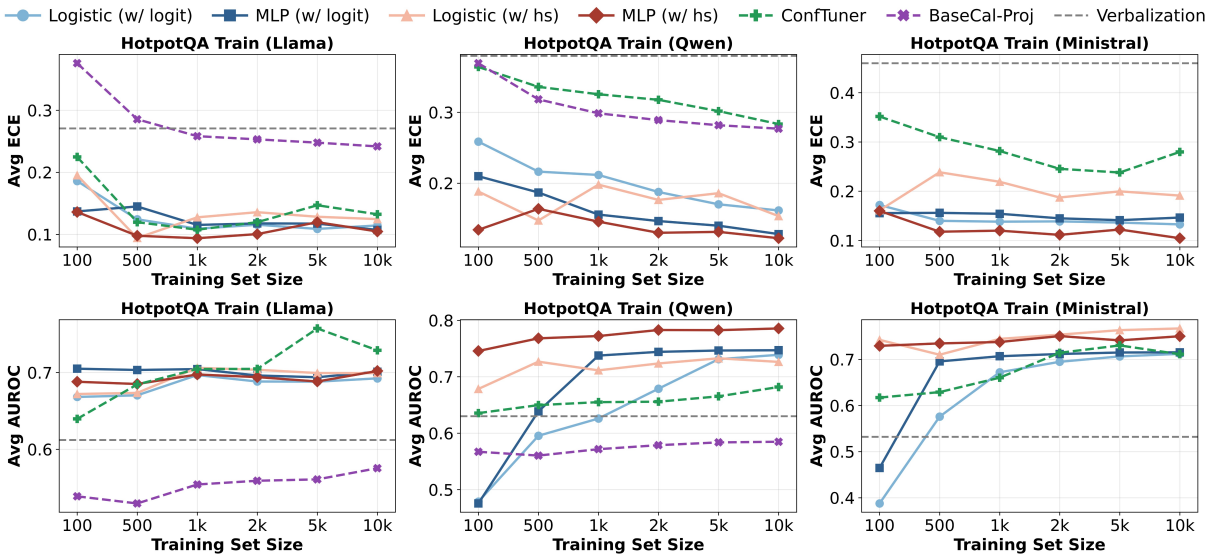

Figure 3: Effect of training set size (100 to 10,000 examples) on calibrator performance (top: ECE ↓; bottom: AUROC ↑). All calibrators are trained on HotpotQA, and results are averaged over the five test sets. Logistic and MLP denote our **EffConf** calibrators trained on either confidence logits (w/ logit) or hidden states (w/ hs).

hidden-state inputs generally provide the strongest calibration performance, which is expected given their richer representation compared to confidence logits. The bottleneck MLP often outperforms logistic regression, indicating that a small amount of non-linear capacity helps. Under black-box access, our MLP trained on confidence log-probabilities achieves an OOD ECE of 0.1009 and an OOD AUROC of 0.7566 on GPT-4o-Mini, outperforming all black-box baselines.

Effect of training data scale. Figure 3 varies the number of HotpotQA training examples from 100 to 10,000 and compares our calibrators with ConfTuner and BaseCal-Proj at the same data scales. Overall, our calibrators are highly sample-efficient: most gains appear with only 500–1,000 training examples, after which both ECE and AUROC curves largely stabilize.

For ECE, our methods consistently stay below ConfTuner and BaseCal-Proj across Llama, Qwen, and Ministral, especially in the low-data regime. This advantage is especially clear for Qwen and Ministral: our best calibrator already achieves low ECE with 100 examples, while ConfTuner and BaseCal-Proj remain substantially worse even with 10,000 examples.

For AUROC, hidden-state calibrators are particularly data-efficient, reaching strong performance with as few as 100 examples on all three LLMs. Logit-based calibrators require more data on Qwen and Ministral, but improve rapidly and become competitive after 1,000 examples. In contrast, ConfTuner benefits from larger training sets in AUROC, suggesting that end-to-end fine-tuning improves discrimination only when sufficient data are available. These results show that our calibrators can achieve effective calibration with only a few hundred to one thousand labeled examples.

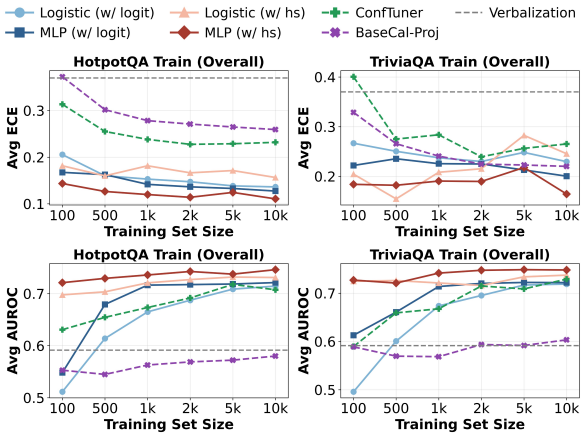

Figure 4: Effect of training distribution on calibrator performance (top: ECE ↓; bottom: AUROC ↑). Calibrators are trained on HotpotQA or TriviaQA with different training set sizes. Results are averaged over corresponding white-box LLMs and five test sets. Logistic and MLP denote our **EffConf** calibrators trained on either confidence logits (w/ logit) or hidden states (w/ hs).

Sensitivity to the training distribution. Figure 4 evaluates whether calibration performance is sensitive to the dataset used for training. We

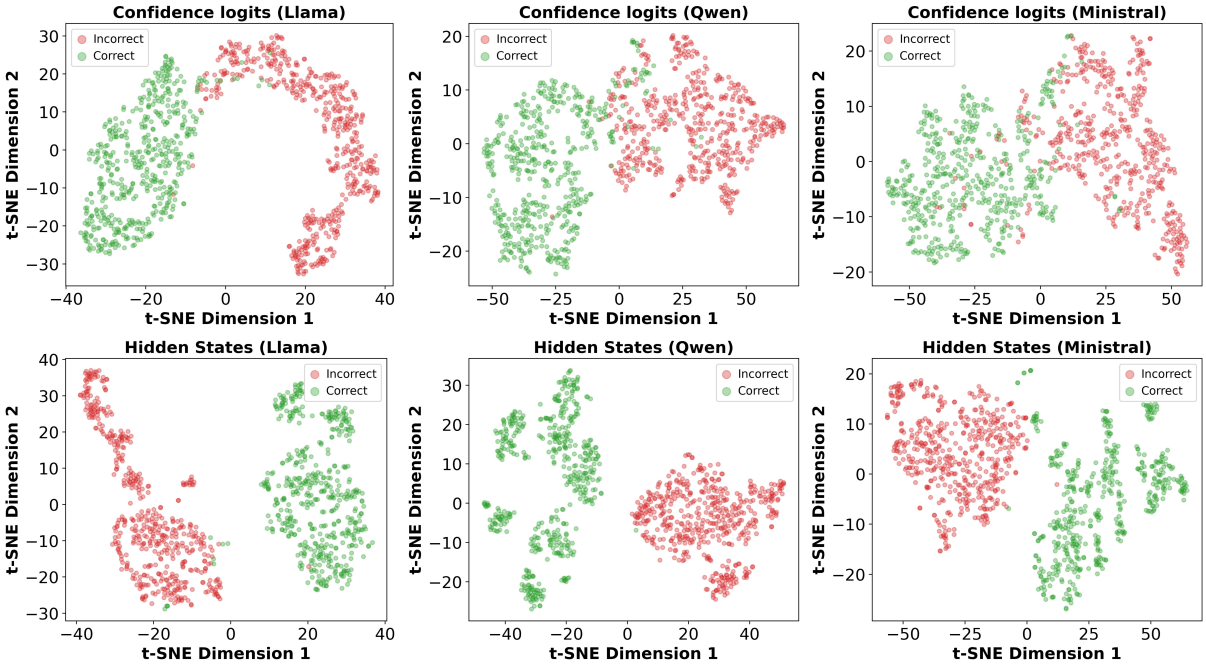

Figure 5: t-SNE visualizations of confidence logits (top) and hidden states (bottom) extracted from the three white-box LLMs on the HotpotQA training set, colored by model answer correctness.

train calibrators on either HotpotQA or TriviaQA with training sizes from 100 to 10,000 examples, and evaluate them on the same five test sets. Overall, our calibrators are robust to this training-distribution shift. For ECE, our methods consistently outperform ConfTuner and BaseCal-Proj under both training corpora. MLP (w/ hs) shows the most stable performance across training sizes, while Logistic (w/ hs) exhibits slightly larger fluctuations when trained on TriviaQA. We also observe that BaseCal-Proj's ECE improves under TriviaQA training, outperforming ConfTuner at every training size. For AUROC, the relative ordering is essentially preserved across both training corpora, with MLP (w/ hs) consistently in the lead.

Signals exploited by our calibrators. We sample 1,000 examples from the HotpotQA training set and use t-SNE to visualize the input representations generated by the white-box LLMs (Figure 5). Specifically, we examine two representations used by EffConf: the confidence logits at the confidence-token position and the hidden state at the immediately preceding position.

The top row shows that confidence logits already encode useful signals: correct (green) and incorrect (red) predictions form partially distinct regions rather than being randomly mixed. However, the separation is still imperfect, especially near the boundary between the two groups.

The bottom row shows a clearer separation for hidden-state representations. Across all three models, correct and incorrect predictions occupy more distinct regions, with less overlap than in the confidence logit projections. This suggests that the hidden state may contain richer information about answer correctness before it is compressed into lower-dimensional confidence logits. This observation is consistent with the results in Table 1, where hidden-state calibrators achieve the strongest OOD ECE across all three white-box LLMs.

6 Conclusion

We present **EffConf**, a family of lightweight calibrators that improve the reliability of LLM verbalized confidence. Rather than fine-tuning the LLM or relying on multiple inference passes, EffConf keeps the LLM frozen and trains a calibrator on signals extracted at the confidence token generation step. This design covers both white-box settings, where logits and hidden states are available, and black-box settings, where APIs expose only top-ranked log-probabilities. Across four LLMs and five QA benchmarks, EffConf consistently outperforms strong baselines, generalizes well to out-of-distribution datasets, and remains highly sample-efficient, while adding negligible cost beyond standard LLM inference.

Limitations

Our work has several limitations. First, we focus on numerical verbalized confidence, where the LLM outputs a discrete value such as "Confidence: 90". This setting is natural for calibration because it defines an explicit confidence token set, it does not cover implicit or free-form uncertainty expressions, such as "I am fairly certain" or "this is likely but not guaranteed." Extending our method to these cases would require mapping natural language uncertainty statements to structured confidence values. Second, our calibrators require a labeled calibration set containing extracted confidence representations and answer correctness labels. Although our experiments show that a few hundred to one thousand examples can be sufficient, the method is not fully label-free. Finally, our experiments focus on single-turn question answering benchmarks. This controlled setting does not capture the additional uncertainty propagation in multi-turn interactions or agentic workflows, where errors may accumulate across reasoning steps, tool calls, and intermediate decisions. Extending the framework to such settings is an important direction for future work.

Ethics Statement

All datasets and models used in this work are publicly released for research purposes. We use them in accordance with their intended use and applicable licenses. The calibrators released with this work are research artifacts intended solely for research use, and their release follows the access conditions of the datasets and models from which they are derived.

AI assistance was used only to refine and polish the writing. It was not used to generate research ideas, or shape the scientific claims of this work.

References

Kendrick Boyd, Kevin H Eng, and C David Page. 2013. Area under the precision-recall curve: point estimates and confidence intervals. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 451–466. Springer.

Prateek Chhikara. 2025. Mind the confidence gap: Overconfidence, calibration, and distractor effects in large language models. *Transactions on Machine Learning Research*.

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

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Tobias Groot and Matias Valdenegro-Toro. 2024. Overconfidence is key: Verbalized uncertainty evaluation in large language and vision-language models. In *Proceedings of the 4th Workshop on Trustworthy Natural Language Processing (TrustNLP 2024)*, pages 145–171.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM computing surveys*, 55(12):1–38.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, and 17 others. 2022. Language models (mostly) know what they know. *Preprint*, arXiv:2207.05221.

Sanyam Kapoor, Nate Gruver, Manley Roberts, Katherine Collins, Arka Pal, Umang Bhatt, Adrian Weller, Samuel Dooley, Micah Goldblum, and Andrew G Wilson. 2024. Large language models must be taught to know what they don't know. *Advances in Neural Information Processing Systems*, 37:85932–85972.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Yibo Li, Miao Xiong, Jiaying Wu, and Bryan Hooi. 2025. Conftuner: Training large language models to express their confidence verbally. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Truthfulqa: Measuring how models mimic human falsehoods. In *Proceedings of the 60th annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 3214–3252.

Jacob Menick, Kevin Lu, Shengjia Zhao, E Wallace, H Ren, H Hu, N Stathas, and F Petroski Such. 2024. Gpt-4o mini: advancing cost-efficient intelligence. *Open AI: San Francisco, CA, USA*.

Mistral AI. 2024. Ministral-8b-instruct-2410. Hugging Face model card.

Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. 2015. Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29.

Jianing Qiu, Kyle Lam, Guohao Li, Amish Acharya, Tien Yin Wong, Ara Darzi, Wu Yuan, and Eric J Topol. 2024. Llm-based agentic systems in medicine and healthcare. *Nature Machine Intelligence*, 6(12):1418–1420.

Elias Stengel-Eskin, Peter Hase, and Mohit Bansal. 2024. Lacie: Listener-aware finetuning for calibration in large language models. *Advances in Neural Information Processing Systems*, 37:43080–43106.

Hexiang Tan, Wanli Yang, Junwei Zhang, Xin Chen, Rui Tang, Du Su, Jingang Wang, Yuanzhuo Wang, Fei Sun, and Xueqi Cheng. 2026. Basecal: Unsupervised confidence calibration via base model signals. In *Proceedings of the 64th Annual Meeting of the 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 *International Conference on Learning Representations*.

Tianyang Xu, Shujin Wu, Shizhe Diao, Xiaoze Liu, Xingyao Wang, Yangyi Chen, and Jing Gao. 2024. Sayself: Teaching llms to express confidence with self-reflective rationales. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 5985–5998.

Qwen: An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 23 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2369–2380.

Yangyang Yu, Zhiyuan Yao, Haohang Li, Zhiyang Deng, Yuechen Jiang, Yupeng Cao, Zhi Chen, Jordan W Suchow, Zhenyu Cui, Rong Liu, and 1 others. 2024. Fincon: A synthesized llm multi-agent system with conceptual verbal reinforcement for enhanced financial decision making. *Advances in Neural Information Processing Systems*, 37:137010–137045.

A Datasets

Table 3 summarizes the test sets used for each benchmark.

Dataset	HuggingFace ID	Size
HotpotQA	`hotpotqa/hotpot_qa`	2,000
GSM8K	`openai/gsm8k`	1,000
TriviaQA	`mandarjoshi/trivia_qa`	1,000
StrategyQA	`ChilleD/StrategyQA`	1,603
TruthfulQA	`truthfulqa/truthful_qa`	815

Table 3: Test sets used for evaluation.

B Training Details

For every training size (from 100 to 10,000), we split the available samples into training and validation sets with a 4:1 ratio. We train our calibrators with Adam (Kingma and Ba, 2015) using a learning rate of 10^{-3} and a batch size of 64. Training runs for a maximum of 200 epochs with early stopping on validation loss (patience of 20 epochs). Early stopping serves as the primary regularization mechanism for both models, complemented by L2 Regularization (λ) of 0.1.

C Prompts

We use GPT-5 as an LLM-as-a-judge evaluator for answers on the HotpotQA benchmark, and include the corresponding judging prompt below (see Prompt C.1). For the other benchmarks, we first clean and normalize the LLM outputs, and then compare them directly against the ground-truth answers using rule-based matching code.

In addition, we provide two example prompts that extract the LLMs' verbalized confidence used in our experiments for the GSM8K dataset in our experiments: one using a 0–9 confidence scale (Prompt C.2) and the other using a 0–100 confidence scale (Prompt C.3).

Prompt C.1: HotpotQA Judge

```
You are an answer-correctness judge. Given
    a question, the ground-truth answer,
    and a model's response, decide whether
    the model's response is correct.

The response is correct (is_correct = 1) if
    it conveys the same essential answer as
    the ground truth, even if it includes
    extra details or different phrasing.
The response is incorrect (is_correct = 0)
    if it gives a wrong answer, contradicts
    the ground truth, or fails to answer
    the question correctly.

Question: {question}

Ground-Truth Answer: {ground_truth}

Model Response: {model_answer}

Is the model's response correct?
```

Prompt C.2: GSM8K (0-9 scale)

```
You will be asked math problems. Please
    respond to the best of your ability.
Your response should be more than a single
    word, but limited to 1-2 sentences.
Then please extract a single answer from
    the your response. If no answer is
    present, please write "NONE".
Finally, please provide your confidence as
    a single integer between 0 and 9 (0 =
    completely unsure, 9 = absolutely
    certain) to your answer.
The confidence score must be a value
    between 0-9, where 9 is the maximum.
    Never use 10.

Here are some examples:

Question: A bag contains 5 red apples and 3
    green apples. How many apples are there
    in total?
Response: To find the total, add the red
    apples and green apples: 5 + 3 = 8
    apples in total.
Final answer: 8
Confidence: 9

Question: A train travels 60 miles per hour
    for 3 hours. How far does it travel in
    total?
Response: To calculate the total distance,
    multiply the speed by the time: 60
    miles/hour * 3 hours = 180 miles.
Final answer: 180
Confidence: 9

Question: A box contains 8 blue marbles and
    6 red marbles. How many marbles are
    there in total?
Response: The total number of marbles is
    the sum of blue and red marbles: 8 + 7
    = 15 marbles.
Final answer: 15
Confidence: 0

Question: {question}
Response:
```

Prompt C.3: GSM8K (0-100 scale)

```
You will be asked math problems. Please
    respond to the best of your ability.
Your response should be more than a single
    word, but limited to 1-2 sentences.
Then please extract a single answer from
    your response. If no answer is present,
    please write "NONE".
Finally, please provide your confidence as
    a single integer between 0 and 100 (0 =
    completely unsure, 100 = absolutely
    certain) to your answer.

Here are some examples:

Question: A bag contains 5 red apples and 3
    green apples. How many apples are there
    in total?
Response: To find the total, add the red
    apples and green apples: 5 + 3 = 8
    apples in total.
Final answer: 8
Confidence: 100

Question: A train travels 60 miles per hour
    for 3 hours. How far does it travel in
    total?
Response: To calculate the total distance,
    multiply the speed by the time: 60
    miles/hour * 3 hours = 180 miles.
Final answer: 180
Confidence: 100

Question: A box contains 8 blue marbles and
    6 red marbles. How many marbles are
    there in total?
Response: The total number of marbles is
    the sum of blue and red marbles: 8 + 7
    = 15 marbles.
Final answer: 15
Confidence: 0

Question: {question}
Response:
```

D Additional Experimental Results

D.1 GPT Results

We further evaluate **EffConf** on GPT in the black-box setting, where only confidence log-probabilities are accessible. Figure 6 visualizes the confidence log-probabilities extracted from GPT on HotpotQA training examples. Although these representations are lower-dimensional than white-box hidden states, correct and incorrect predictions still form partially separated regions, suggesting that API-accessible log-probabilities carry useful information about answer correctness even without access to model internals.

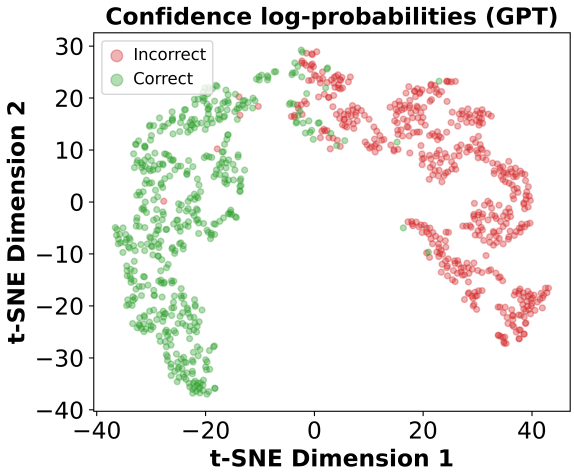

Figure 6: t-SNE visualizations of confidence log probabilities extracted from the GPT on HotpotQA training set, colored by model answer correctness.

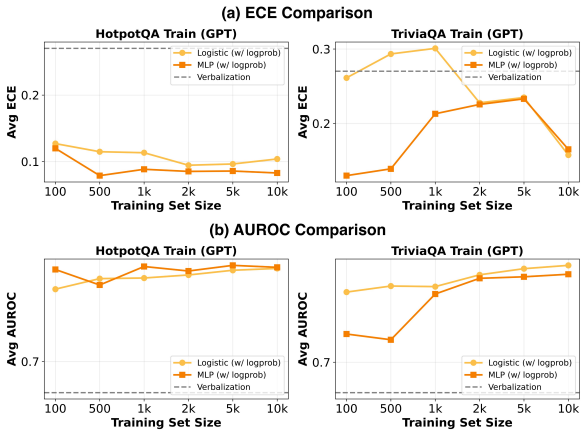

Figure 7: Effect of training set size (100 to 10,000 examples) and training distribution (HotpotQA and TriviaQA) on calibrator performance (top: ECE ↓; bottom: AUROC ↑) for GPT, averaged across all five test sets. Logistic and MLP denote our **EffConf** calibrators trained on confidence log-probabilities (w/ logprob).

Figure 7 shows the effect of training set size and

training distribution on GPT calibration. When trained on HotpotQA, both logistic regression and MLP calibrators consistently improve over verbalized confidence in ECE and AUROC, with strong performance already at small training sizes. When trained on TriviaQA, the results are more variable in terms of ECE, especially for the logistic calibrator at smaller training sizes. However, the MLP calibrator remains consistently below the verbalization baseline in ECE, and both calibrators achieve clear AUROC improvements over verbalized confidence. Overall, these results show that **EffConf** remains effective for black-box LLMs.

D.2 Efficiency Comparison

Table 4 reports the training cost of each method along four axes: training time, training set size, number of trainable parameters, and number of epochs to convergence. Because **EffConf** trains a lightweight calibrator on top of a frozen LLM, it converges in a few seconds with at most ∼0.5M trainable parameters, two to three orders of magnitude faster than ConfTuner and BaseCal-Proj. All experiments are conducted on a single NVIDIA A100 GPU (80 GB).

Method	Training Time (sec)	Training Data Size	Trainable Parameters	Training Epochs
BaseCal-Proj	395.4	10,000	∼14.8M	50
ConfTuner	290.6	2,000	∼3.2M	2
Logistic (w/ logit)	1.6/1.7	1,000	11/102	200
MLP (w/ logit)	2.1/2.3	1,000	49/825	149
Logistic (w/ hs)	0.6	1,000	∼4,000	51
MLP (w/ hs)	0.4	1,000	∼0.5M	31
Logistic (w/ logprob)	1.6	1,000	11	173
MLP (w/ logprob)	2.2	1,000	49	197

Table 4: Training efficiency comparison of **EffConf** against baseline methods (BaseCal-Proj, ConfTuner), results averaged across corresponding LLMs. Slash-separated values denote the 0–9 and 0–100 confidence token settings.

D.3 Ablation Studies

We conduct ablation studies for EffConf by varying the L2 regularization strength λ and the MLP bottleneck dimension d. All calibrators are trained on 1,000 samples from HotpotQA and evaluated on both the in-distribution HotpotQA test set and four out-of-distribution (OOD) benchmarks. Tables 5–8 report results for all four LLMs. Overall, the results show that our calibrators are robust to moderate changes in both λ and d.

For logistic calibrators, varying λ has only a small effect, which is expected given their limited model capacity. A moderate regularization strength, $\lambda = 0.1$, often improves OOD calibration performance, especially for confidence logit and log-probability inputs. This suggests that mild regularization helps prevent the calibrator from overfitting to the HotpotQA training distribution.

For MLP calibrators, the bottleneck dimension d has a more pronounced effect. However, increasing d does not consistently improve OOD performance. In several cases, larger bottlenecks improve in-distribution fit but degrade OOD ECE, indicating that additional capacity can make the calibrator more sensitive to source-domain artifacts. In contrast, compact bottlenecks generally provide a better balance between expressiveness and generalization. This supports our design choice of using a narrow hidden layer, which encourages the calibrator to learn a low-dimensional summary of the correctness-related signal rather than memorizing patterns in the training distribution. Beyond the bottleneck dimension, regularization also matters more for MLPs than for logistic calibrators. Without regularization ($\lambda = 0$), some high-capacity MLPs collapse to near-constant predictors, such as MLP (w/hs) on Ministral with $d \in \{64, 128\}$ and MLP (w/logprob) on GPT with $d = 2$. Adding even a small penalty ($\lambda \geq 0.001$) restores stable calibration performance in these cases, motivating our use of nonzero λ in the main configurations.

Overall, these ablations support the main configurations used in our experiments. A lightweight calibrator with moderate regularization and a compact bottleneck provides a strong balance between in-distribution fit, OOD calibration, and robustness.

Method	λ	In-distribution		Out-of-distribution (OOD)								OOD Average	
		HotpotQA		GSM8K		TriviaQA		StrategyQA		TruthfulQA			
		ECE ↓	AUROC ↑	ECE ↓	AUROC ↑	ECE ↓	AUROC ↑	ECE ↓	AUROC ↑	ECE ↓	AUROC ↑	ECE ↓	AUROC ↑
Logistic (w/logit)	0.0	0.0442	0.7353	0.1108	0.7215	0.1364	0.6918	0.1910	0.6320	0.0931	0.6885	0.1328	0.6835
	0.001	0.0442	0.7352	0.1106	0.7216	0.1361	0.6919	0.1911	0.6320	0.0908	0.6885	0.1321	0.6835
	0.01	0.0453	0.7350	0.1084	0.7226	0.1338	0.6923	0.1938	0.6326	0.0906	0.6894	0.1316	0.6842
	0.1†	0.0470	0.7316	0.0863	0.7278	0.1157	0.6949	0.2061	0.6345	0.0908	0.6949	0.1247	0.6880
MLP (w/logit; d=4)	0.0	0.0351	0.7167	0.0487	0.7385	0.0994	0.7034	0.2275	0.6484	0.1113	0.7026	0.1217	0.6982
	0.001	0.0311	0.7082	0.0734	0.7303	0.1274	0.7090	0.2472	0.6546	0.0869	0.7102	0.1337	0.7010
	0.01	0.0400	0.7286	0.0751	0.7481	0.0930	0.6909	0.1644	0.6566	0.1807	0.6914	0.1283	0.6967
MLP (w/logit; d=8)	0.0	0.0415	0.7256	0.0740	0.7411	0.1196	0.7013	0.1391	0.6836	0.1128	0.7182	0.1114	0.7110
	0.001	0.0544	0.7229	0.0339	0.7410	0.0624	0.6901	0.2422	0.6385	0.1885	0.6979	0.1318	0.6919
	0.01	0.0368	0.7304	0.0787	0.7432	0.1058	0.6907	0.1643	0.6606	0.1707	0.7032	0.1299	0.6994
	0.1†	0.0342	0.7100	0.0874	0.7400	0.0912	0.7046	0.2862	0.6402	0.0773	0.7278	0.1355	0.7032
MLP (w/logit; d=16)	0.0	0.0343	0.7221	0.1072	0.7376	0.2194	0.7019	0.1822	0.6449	0.1574	0.6710	0.1666	0.6888
	0.001	0.0411	0.7245	0.1046	0.7347	0.2042	0.7036	0.1670	0.6447	0.1905	0.6809	0.1666	0.6910
	0.01	0.0410	0.7224	0.0525	0.7308	0.1794	0.7008	0.2818	0.6223	0.1218	0.7065	0.1589	0.6901
	0.1	0.0335	0.7146	0.0549	0.7278	0.1048	0.6892	0.3379	0.6073	0.0502	0.7063	0.1369	0.6826
Logistic (w/hs)	0.0	0.0381	0.7615	0.1378	0.7529	0.1844	0.6951	0.1137	0.6677	0.2172	0.6068	0.1633	0.6806
	0.001	0.0380	0.7615	0.1377	0.7529	0.1841	0.6951	0.1131	0.6680	0.2159	0.6070	0.1627	0.6807
	0.01	0.0376	0.7618	0.1370	0.7534	0.1849	0.6949	0.1048	0.6700	0.2080	0.6071	0.1587	0.6814
	0.1†	0.0270	0.7635	0.1135	0.7578	0.2014	0.6960	0.0985	0.6670	0.1971	0.6466	0.1526	0.6919
MLP (w/hs; d=64)	0.0	0.0307	0.7642	0.1410	0.7628	0.1894	0.6975	0.1021	0.6736	0.2428	0.6165	0.1688	0.6876
	0.001	0.0274	0.7655	0.1412	0.7654	0.1900	0.7071	0.0610	0.6887	0.1702	0.6220	0.1406	0.6958
	0.01	0.0291	0.7670	0.1279	0.7659	0.1904	0.6930	0.0944	0.6732	0.2184	0.6301	0.1578	0.6905
	0.1	0.0251	0.7526	0.1021	0.7528	0.0842	0.6821	0.1673	0.6575	0.0813	0.6457	0.1087	0.6845
MLP (w/hs; d=128)	0.0	0.0187	0.7643	0.1459	0.7607	0.1853	0.7071	0.0572	0.6909	0.2080	0.6112	0.1491	0.6925
	0.001	0.0389	0.7660	0.1384	0.7599	0.1905	0.7021	0.0742	0.6876	0.1384	0.6297	0.1354	0.6948
	0.01	0.0216	0.7678	0.1295	0.7717	0.2009	0.6941	0.0616	0.6762	0.1951	0.6421	0.1468	0.6960
	0.1†	0.0303	0.7503	0.0718	0.7559	0.0880	0.6825	0.1726	0.6500	0.1066	0.6489	0.1098	0.6843
MLP (w/hs; d=256)	0.0	0.0219	0.7618	0.1413	0.7613	0.1739	0.6938	0.0754	0.6838	0.1643	0.6404	0.1387	0.6948
	0.001	0.0312	0.7621	0.1471	0.7664	0.2070	0.7028	0.1111	0.6805	0.2809	0.6190	0.1865	0.6922
	0.01	0.0240	0.7645	0.1275	0.7639	0.1740	0.6908	0.0825	0.6812	0.2078	0.6382	0.1479	0.6935
	0.1	0.0510	0.7427	0.0582	0.7526	0.0844	0.6839	0.1942	0.6483	0.1107	0.6613	0.1119	0.6865

Table 5: Ablation over L2 regularization strength λ and MLP bottleneck dimension d for our **EffConf** calibrators on **Llama**, using confidence logits (w/logit) and hidden states (w/hs) as input. † denotes our main configuration.

Method	λ	In-distribution		Out-of-distribution (OOD)								OOD Average	
		HotpotQA		GSM8K		TriviaQA		StrategyQA		TruthfulQA			
		ECE ↓	AUROC ↑	ECE ↓	AUROC ↑	ECE ↓	AUROC ↑	ECE ↓	AUROC ↑	ECE ↓	AUROC ↑	ECE ↓	AUROC ↑
Logistic (w/logit)	0.0	0.1191	0.7217	0.2769	0.3446	0.2728	0.7035	0.1535	0.6991	0.2783	0.6015	0.2454	0.5872
	0.001	0.1191	0.7217	0.2757	0.3448	0.2727	0.7036	0.1535	0.6992	0.2779	0.6017	0.2450	0.5873
	0.01	0.1174	0.7220	0.2776	0.3472	0.2715	0.7043	0.1538	0.6999	0.2746	0.6037	0.2444	0.5888
	0.1†	0.1182	0.7245	0.2778	0.3729	0.2622	0.7110	0.1553	0.7060	0.2447	0.6156	0.2350	0.6014
MLP (w/logit; d=2)	0.0	0.1356	0.7227	0.0718	0.8322	0.2398	0.7695	0.0568	0.7113	0.2014	0.7014	0.1424	0.7536
	0.001	0.1357	0.7228	0.0712	0.8322	0.2396	0.7694	0.0569	0.7113	0.2009	0.7014	0.1421	0.7536
	0.01	0.1363	0.7232	0.0642	0.8325	0.2374	0.7696	0.0604	0.7118	0.1962	0.7015	0.1396	0.7539
	0.1	0.1373	0.7266	0.0962	0.8324	0.2231	0.7711	0.0811	0.7173	0.1315	0.7029	0.1330	0.7559
MLP (w/logit; d=4)	0.0	0.1266	0.7323	0.1550	0.7217	0.2420	0.7528	0.1216	0.7099	0.1779	0.6796	0.1741	0.7160
	0.001	0.1260	0.7323	0.1560	0.7233	0.2412	0.7529	0.1219	0.7100	0.1760	0.6800	0.1738	0.7166
	0.01	0.1264	0.7324	0.1623	0.7360	0.2371	0.7543	0.1232	0.7112	0.1665	0.6817	0.1723	0.7208
	0.1†	0.1406	0.7319	0.1915	0.7893	0.2164	0.7610	0.1222	0.7225	0.1082	0.6832	0.1596	0.7390
MLP (w/logit; d=6)	0.0	0.1240	0.7369	0.1202	0.7324	0.2713	0.7659	0.1515	0.7226	0.1309	0.6593	0.1685	0.7201
	0.001	0.1227	0.7370	0.1223	0.7335	0.2716	0.7661	0.1510	0.7222	0.1323	0.6597	0.1693	0.7204
	0.01	0.1238	0.7376	0.1407	0.7402	0.2652	0.7665	0.1477	0.7230	0.1253	0.6617	0.1697	0.7228
	0.1	0.1361	0.7356	0.1512	0.7890	0.2304	0.7689	0.1257	0.7198	0.1248	0.6694	0.1580	0.7368
Logistic (w/hs)	0.0	0.1168	0.7826	0.4658	0.7543	0.1332	0.8135	0.1331	0.7161	0.1491	0.4897	0.2203	0.6934
	0.001	0.1168	0.7826	0.4659	0.7543	0.1332	0.8135	0.1331	0.7161	0.1491	0.4897	0.2203	0.6934
	0.01	0.1167	0.7826	0.4661	0.7542	0.1332	0.8136	0.1329	0.7161	0.1499	0.4897	0.2205	0.6934
	0.1†	0.1142	0.7828	0.4658	0.7549	0.1323	0.8136	0.1299	0.7166	0.1478	0.4888	0.2190	0.6935
MLP (w/hs; d=64)	0.0	0.0827	0.7956	0.3341	0.7983	0.1847	0.8164	0.0964	0.7173	0.0580	0.6317	0.1683	0.7409
	0.001	0.1227	0.7962	0.2271	0.8157	0.2374	0.8184	0.1315	0.7339	0.1000	0.6432	0.1740	0.7528
	0.01	0.0792	0.7960	0.2618	0.7834	0.1811	0.8194	0.1002	0.7475	0.0518	0.6420	0.1487	0.7481
	0.1	0.0952	0.7943	0.2764	0.7823	0.1955	0.8189	0.1046	0.7461	0.0735	0.6730	0.1625	0.7551
MLP (w/hs; d=128)	0.0	0.1335	0.7914	0.1858	0.8306	0.2573	0.8155	0.1151	0.7443	0.1133	0.6909	0.1679	0.7703
	0.001	0.1037	0.7974	0.2550	0.7998	0.2093	0.8231	0.1130	0.7580	0.0715	0.6605	0.1622	0.7603
	0.01	0.1279	0.7936	0.2375	0.8004	0.2327	0.8209	0.1132	0.7498	0.1096	0.6878	0.1732	0.7647
	0.1†	0.1093	0.7937	0.2717	0.7969	0.1997	0.8210	0.0876	0.7459	0.0608	0.7034	0.1550	0.7668
MLP (w/hs; d=256)	0.0	0.1354	0.7988	0.2387	0.7780	0.2411	0.8261	0.1398	0.7514	0.0776	0.6478	0.1743	0.7508
	0.001	0.0706	0.7933	0.2683	0.8045	0.2225	0.8218	0.0988	0.7543	0.0394	0.6652	0.1573	0.7614
	0.01	0.1115	0.7968	0.2903	0.7453	0.1843	0.8259	0.1164	0.7544	0.0519	0.6449	0.1607	0.7426
	0.1	0.1098	0.7921	0.2320	0.7906	0.2098	0.8192	0.1045	0.7449	0.0967	0.7000	0.1608	0.7637

Table 6: Ablation over L2 regularization strength λ and MLP bottleneck dimension d for our **EffConf** calibrators on **Qwen**, using confidence logits (w/logit) and hidden states (w/hs) as input. † denotes our main configuration.

Method	λ	In-distribution		Out-of-distribution (OOD)								OOD Average	
		HotpotQA		GSM8K		TriviaQA		StrategyQA		TruthfulQA			
		ECE ↓	AUROC ↑	ECE ↓	AUROC ↑	ECE ↓	AUROC ↑	ECE ↓	AUROC ↑	ECE ↓	AUROC ↑	ECE ↓	AUROC ↑
Logistic (w/logit)	0.0	0.1167	0.6739	0.2416	0.5788	0.0951	0.7198	0.1352	0.6163	0.0949	0.7520	0.1417	0.6667
	0.001	0.1167	0.6739	0.2429	0.5788	0.0964	0.7198	0.1352	0.6163	0.0976	0.7521	0.1430	0.6667
	0.01	0.1166	0.6741	0.2433	0.5789	0.0991	0.7200	0.1356	0.6164	0.0942	0.7530	0.1431	0.6671
	0.1†	0.1179	0.6755	0.2469	0.5844	0.1035	0.7211	0.1380	0.6188	0.0846	0.7608	0.1433	0.6713
MLP (w/logit; d=2)	0.0	0.1387	0.7054	0.1701	0.7484	0.1156	0.7458	0.1669	0.6332	0.1101	0.7865	0.1407	0.7285
	0.001	0.1393	0.7054	0.1696	0.7483	0.1178	0.7457	0.1670	0.6332	0.1103	0.7865	0.1412	0.7284
	0.01	0.1390	0.7047	0.1764	0.7479	0.1211	0.7455	0.1682	0.6334	0.1117	0.7866	0.1444	0.7284
	0.1	0.1342	0.7007	0.2035	0.7434	0.1063	0.7443	0.1726	0.6344	0.1345	0.7872	0.1542	0.7273
MLP (w/logit; d=4)	0.0	0.1197	0.6861	0.2657	0.6808	0.0806	0.7344	0.1441	0.6358	0.1117	0.7821	0.1505	0.7083
	0.001	0.1231	0.6865	0.2656	0.6817	0.0848	0.7345	0.1412	0.6356	0.1091	0.7824	0.1502	0.7086
	0.01	0.1199	0.6860	0.2635	0.6814	0.0887	0.7347	0.1457	0.6361	0.1202	0.7826	0.1545	0.7087
	0.1†	0.1245	0.6844	0.2509	0.6922	0.1004	0.7369	0.1694	0.6390	0.1258	0.7821	0.1616	0.7126
MLP (w/logit; d=6)	0.0	0.1348	0.6844	0.2975	0.6513	0.1160	0.7299	0.1445	0.6353	0.1001	0.7592	0.1645	0.6939
	0.001	0.1180	0.6891	0.2780	0.6757	0.0793	0.7325	0.1439	0.6334	0.0887	0.7696	0.1475	0.7028
	0.01	0.1211	0.6908	0.2724	0.6798	0.0941	0.7331	0.1412	0.6326	0.0848	0.7726	0.1481	0.7045
	0.1	0.1253	0.6909	0.2631	0.6879	0.1132	0.7345	0.1594	0.6337	0.1109	0.7767	0.1616	0.7082
Logistic (w/hs)	0.0	0.1577	0.7703	0.0710	0.7509	0.0905	0.7748	0.1253	0.6125	0.3555	0.8619	0.1606	0.7500
	0.001	0.1577	0.7703	0.0711	0.7510	0.0903	0.7748	0.1252	0.6125	0.3552	0.8618	0.1605	0.7500
	0.01	0.1583	0.7704	0.0711	0.7511	0.0876	0.7748	0.1250	0.6127	0.3529	0.8619	0.1591	0.7501
	0.1†	0.1054	0.7713	0.2493	0.7357	0.1840	0.7673	0.1480	0.5967	0.4090	0.8517	0.2476	0.7379
MLP (w/hs; d=64)	0.0	0.1354	0.5000	0.2112	0.5106	0.0511	0.5000	0.1732	0.4924	0.2067	0.5004	0.1605	0.5009
	0.001	0.0831	0.7740	0.1252	0.7397	0.1372	0.7785	0.1240	0.6087	0.4183	0.8325	0.2012	0.7399
	0.01	0.0876	0.7727	0.0629	0.7526	0.0843	0.7747	0.0942	0.6210	0.2904	0.8358	0.1330	0.7460
	0.1	0.1047	0.7538	0.0948	0.7573	0.1008	0.7596	0.1058	0.6233	0.1693	0.8137	0.1177	0.7385
MLP (w/hs; d=128)	0.0	0.1354	0.5000	0.2133	0.4875	0.0511	0.5000	0.1777	0.4799	0.2066	0.5014	0.1622	0.4922
	0.001	0.0722	0.7744	0.1560	0.7391	0.1614	0.7767	0.1299	0.6062	0.4419	0.8274	0.2223	0.7373
	0.01	0.0946	0.7737	0.0797	0.7448	0.1117	0.7763	0.0911	0.6163	0.3604	0.8363	0.1607	0.7434
	0.1†	0.1255	0.7472	0.1000	0.7594	0.0902	0.7547	0.1076	0.6231	0.1765	0.8046	0.1186	0.7355
MLP (w/hs; d=256)	0.0	0.1068	0.7739	0.0619	0.7425	0.1112	0.7782	0.1344	0.6095	0.4357	0.8275	0.1858	0.7394
	0.001	0.1230	0.7743	0.0487	0.7474	0.0844	0.7798	0.1249	0.6132	0.4205	0.8304	0.1696	0.7427
	0.01	0.1360	0.7726	0.0570	0.7525	0.0860	0.7766	0.1071	0.6185	0.3654	0.8319	0.1539	0.7449
	0.1	0.1016	0.7565	0.0992	0.7575	0.1019	0.7610	0.0978	0.6207	0.1905	0.8171	0.1224	0.7391

Table 7: Ablation over L2 regularization strength λ and MLP bottleneck dimension d for our **EffConf** calibrators on **Ministral**, using confidence logits (w/logit) and hidden states (w/hs) as input. † denotes our main configuration.

Method	λ	In-distribution		Out-of-distribution (OOD)								OOD Average	
		HotpotQA		GSM8K		TriviaQA		StrategyQA		TruthfulQA			
		ECE ↓	AUROC ↑	ECE ↓	AUROC ↑	ECE ↓	AUROC ↑	ECE ↓	AUROC ↑	ECE ↓	AUROC ↑	ECE ↓	AUROC ↑
Logistic (w/logprob)	0.0	0.0405	0.7622	0.0198	0.8175	0.1198	0.7215	0.1607	0.7506	0.2409	0.7001	0.1353	0.7474
	0.001	0.0359	0.7661	0.0203	0.8160	0.1092	0.7221	0.1682	0.7479	0.2081	0.7073	0.1265	0.7483
	0.01	0.0356	0.7664	0.0219	0.8160	0.1086	0.7224	0.1682	0.7484	0.2079	0.7076	0.1266	0.7486
	0.1†	0.0373	0.7644	0.0189	0.8172	0.1154	0.7231	0.1628	0.7523	0.2322	0.7020	0.1323	0.7487
MLP (w/logprob; d=2)	0.0	0.0121	0.5000	0.3941	0.5000	0.1591	0.5000	0.2212	0.5000	0.0203	0.5000	0.1987	0.5000
	0.001	0.0278	0.7522	0.0216	0.8178	0.1050	0.7307	0.1577	0.7195	0.0836	0.6867	0.0920	0.7387
	0.01	0.0395	0.7727	0.0260	0.8182	0.0990	0.7319	0.1762	0.7506	0.1202	0.7431	0.1053	0.7610
	0.1	0.0469	0.7697	0.0580	0.8189	0.0855	0.7316	0.1694	0.7442	0.1076	0.7258	0.1051	0.7551
MLP (w/logprob; d=4)	0.0	0.0311	0.7595	0.0267	0.8214	0.1072	0.7334	0.1505	0.7361	0.0923	0.6848	0.0942	0.7439
	0.001	0.0402	0.7695	0.0350	0.8216	0.1056	0.7336	0.1575	0.7485	0.0988	0.7393	0.0992	0.7608
	0.01	0.0361	0.7671	0.0364	0.8208	0.0984	0.7324	0.1620	0.7413	0.1104	0.7345	0.1018	0.7572
	0.1†	0.0387	0.7683	0.0497	0.8203	0.0935	0.7321	0.1640	0.7421	0.0963	0.7318	0.1009	0.7566
MLP (w/logprob; d=6)	0.0	0.0408	0.7703	0.0341	0.8173	0.0856	0.7280	0.1834	0.7323	0.1128	0.7308	0.1040	0.7521
	0.001	0.0419	0.7716	0.0189	0.8149	0.0906	0.7270	0.1856	0.7453	0.1074	0.7472	0.1006	0.7586
	0.01	0.0382	0.7704	0.0351	0.8172	0.0865	0.7283	0.1812	0.7346	0.1180	0.7305	0.1052	0.7527
	0.1	0.0447	0.7710	0.0481	0.8170	0.0865	0.7302	0.1734	0.7458	0.1340	0.7265	0.1105	0.7549

Table 8: Ablation over L2 regularization strength λ and MLP bottleneck dimension d for our **EffConf** calibrators on **GPT**, using confidence log probabilities (w/logprob) as input. † denotes our main configuration.