Probing Hidden States for Calibrated, Alignment-Resistant Predictions in LLMs

Jacob Berkowitz[1,2†], Sophia Kivelson[1,2†], Apoorva Srinivasan[1,2], Undina Gisladottir[1,2,3], Kevin K. Tsang[1,2], Jose Miguel Acitores Cortina[1,2], Aditi Kuchi[1,2], Jake Patock[1,2], Ryan Czarny[1,2], Nicholas P. Tatonetti[1,2*]

[1*]Department of Computational Biomedicine, Cedars-Sinai Medical Center, Los Angeles, 90069, CA, USA.
[2]Cedars-Sinai Cancer, Cedars-Sinai Medical Center, Los Angeles, 90069, CA, USA.
[3]Department of Biomedical Informatics, Columbia University Irving Medical Center, New York, 10032, NY, USA.

*Corresponding author(s). E-mail(s): nicholas.tatonetti@cshs.org;
†These authors contributed equally to this work.

Abstract

Scientific applications of large language models (LLMs) demand reliable, well-calibrated predictions, but standard generative approaches often fail to fully access relevant knowledge contained in their internal representations. As a result, models appear less capable than they are, with useful information remaining latent. We present PING (Probing INternal states of Generative models), an open-source framework that trains lightweight probes on frozen, HuggingFace-compatible transformers to deliver structured predictions with minimal compute overhead. Across diverse models and benchmarks including MMLU for broad coverage and MedMCQA for clinical focus, PING matches or exceeds generative accuracy while reducing Expected Calibration Error by up to 96%. Strikingly, on an LLM that has been explicitly safety-tuned to withhold medical information, PING recovered 87% of lost MedMCQA performance while generative accuracy is zero, showing this information still exists in the model's latent space. The accompanying `pingkit` package makes these methods easy to deploy and is available through PyPI.

Keywords: Large language models, Probing hidden states, Model calibration, Alignment resistance, Biomedical natural language processing

1 Introduction

Large language models (LLMs) have quickly become indispensable research tools, capable of distilling large bodies of literature and powering workflows from hypothesis generation to patient-level decision support. In many cases, they improve efficiency and even surpass human experts on specialized tasks [1, 2]. However, their apparent expertise conceals critical shortcomings. State-of-the-art models can generate incorrect information ("hallucinations"), be misled by adversarial prompts, or refuse to answer legitimate domain-specific questions after alignment tuning [3, 4]. More fundamentally, LLMs often exhibit poor calibration: they report high confidence in wrong answers, which is especially problematic in biomedical and safety-critical contexts [5–7]. If a model that claims 95% confidence is correct much less often, it can lead to poor recommendations [8] and, over time, erode user trust.

Understanding how these models process information points toward possible solutions. Transformer architectures build internal representations layer by layer, using attention to capture relationships between tokens and feed-forward networks to transform them [9, 10]. At each step, the model produces hidden states, high-dimensional vectors that summarize what the model "knows" about the input so far. These internal states evolve as information flows through the network, forming increasingly rich representations of the text.

Importantly, these hidden states often contain more information than what appears in the model's final answer. Studies have shown that intermediate layers encode signals of factual knowledge, uncertainty, and reasoning structure [11–15], but much of this signal is blurred or discarded during decoding [16, 17]. To access these latent signals, researchers often train probes, small classifiers that take in hidden states and predict some property of interest. Probes on final-layer activations have been used to estimate hallucination risk with over 80% accuracy across hundreds of datasets [18]. Other work shows that signals of truthfulness cluster around answer tokens, exposing mismatches between what the model encodes and what it outputs [19]. Still others decode symbolic propositions directly from hidden layers, revealing that structured knowledge persists even under adversarial interventions [11]. Together, these findings suggest that LLMs internally "know" more than they are able (or sometimes allowed) to say.

Building on this insight, we developed Probing INternal states of Generative models (PING), a framework that turns latent signals into calibrated, deployable predictions (Figure 1). Unlike most prior probing work, which has focused on interpreting or diagnosing models, PING integrates probing directly into the inference pipeline to improve model outputs in real-world tasks without any fine-tuning of the base model. We report three major contributions. First, we show across diverse models and benchmarks that lightweight probes trained on frozen transformers can improve calibration and maintain or enhance accuracy, while also ensuring alignment and reducing inference cost compared to standard generative decoding. Second, we demonstrate that PING can recover suppressed domain knowledge, enabling accurate responses in cases where safety training causes models to refuse legitimate questions. Third, to make these capabilities broadly accessible, we release the open-source `pingkit` package, which automates activation extraction, systematic layer sweeps, and probe training, enabling researchers to apply PING to their own models and datasets. Beyond enabling

safer and more effective LLM use in research, PING also serves as a diagnostic for alignment depth: if factual knowledge can be recovered from hidden states, harmful content could be as well, underscoring both the utility and the safety implications of probing in modern LLMs.

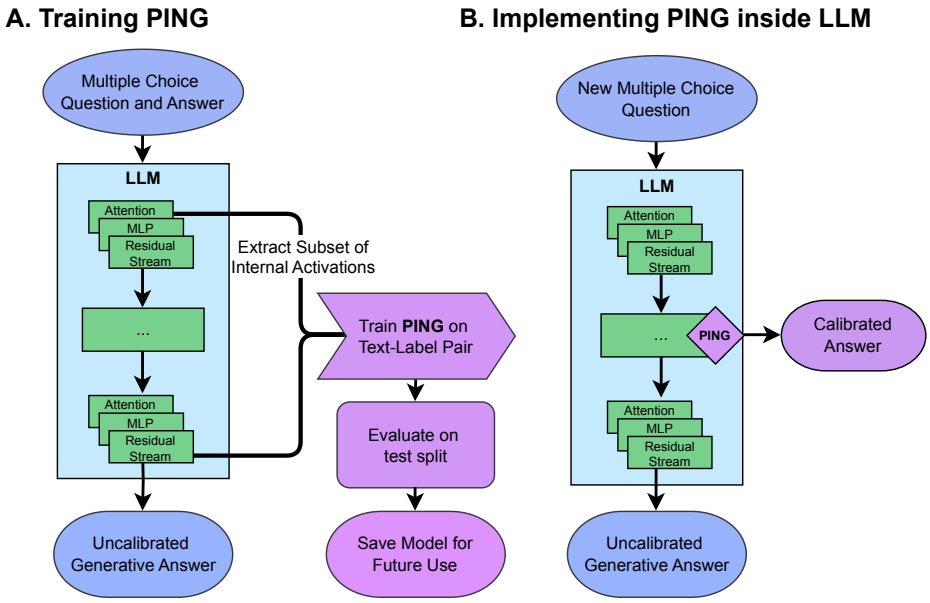

Fig. 1 The PING framework extracts well-calibrated predictions from frozen language models. A. Training phase: Internal representations (residual stream, attention outputs, and MLP outputs) are extracted from any combination of layers from a frozen transformer as it processes training examples. These representations are pooled, concatenated, and used to train lightweight probe networks that learn to map internal states to calibrated predictions. **B**. Inference phase: The trained probe is integrated with the frozen model. During inference, the model processes input normally while PING extracts activations and produces well-calibrated predictions in parallel with standard generation, adding minimal latency.

2 Results

2.1 PING recovers well-calibrated predictions across model scales and architectures

We evaluated PING (Probing INternal states of Generative models) on five publicly available transformer language models spanning different scales and training approaches. We selected Gemma-2-2B-it and Gemma-2-9B-it (Google's instruction-tuned models with 2 billion and 9 billion parameters, respectively) [20], Llama-3.1-8B

(Meta's 8-billion parameter base model without instruction tuning) and its instruction-tuned counterpart Llama-3.1-8B-Instruct, and Llama-3.3-70B-Instruct (Meta's large-scale 70-billion parameter model) [21]. This selection allowed us to assess PING's performance across model sizes ranging from 2 to 70 billion parameters and compare base versus instruction-tuned variants.

We first tested on the Massive Multitask Language Understanding (MMLU) benchmark, a widely-used evaluation suite containing 14,042 multiple-choice questions across 57 subjects, including medicine, biology, chemistry, and other scientific domains. MMLU provides a standardized measure of knowledge retrieval capabilities relevant to biomedical applications.

Across all models, PING maintained or improved accuracy while dramatically reducing calibration error (Table 1). Expected Calibration Error (ECE) measures the mismatch between a model's confidence and its actual accuracy, where higher values indicate overconfidence.

Gemma-2-9B-it achieved 70.4% accuracy with PING versus 69.2% ($p = 0.306$) with standard generation, while ECE decreased from 0.164 to 0.031 ($p < 0.001$), an 81.1% reduction in calibration error. The large-scale Llama-3.3-70B-Instruct had ECE drop from 0.215 to 0.009 (95.8% reduction, $p < 0.001$) while maintaining comparable accuracy (79.0% versus 78.8%, $p = 0.703$). Notably, the base Llama-3.1-8B model, which lacks instruction tuning, showed substantial accuracy gains from 40.7% to 54.0% ($p < 0.001$) with PING. All p-values for ECE comparisons, along with results from additional calibration techniques including temperature scaling and isotonic regression, are reported in Supplementary Table 1.

Across all five models, PING outperformed both temperature scaling and isotonic regression in reducing calibration error. These improvements were statistically significant in 4/5 models compared to temperature scaling and 2/5 models compared to isotonic regression.

Table 1 Massive Multitask Language Understanding (MMLU) accuracy and Expected Calibration Error (ECE) for generative decoding vs. PING. Statistical significance from two-sample, two-sided z-test indicated by * (p < 0.05), ** (p < 0.001).

	Generative		PING	
Model	Accuracy (%)	ECE (%)	Accuracy (%)	ECE (%)
Gemma-2-2B-it	**54.8 ± 0.6**	23.9 ± 1.0	54.1 ± 1.2	**2.8 ± 1.0****
Gemma-2-9B-it	69.2 ± 0.5	16.4 ± 1.1	**70.4 ± 1.1**	**3.1 ± 0.9****
Llama-3.1-8B	40.7 ± 0.6	6.7 ± 0.7	**54.0 ± 1.2****	**1.8 ± 0.8****
Llama-3.1-8B-Instruct	61.0 ± 0.6	11.8 ± 0.6	**64.5 ± 1.1****	**2.5 ± 0.7****
Llama-3.3-70B-Instruct	78.8 ± 0.5	21.5 ± 2.1	**79.0 ± 0.9**	**0.9 ± 0.6****

Across all models, optimal layers ranged from 49% to 100% of total network depth (Figure 2). Optimal probe performance occurred at layer 25/26 for Gemma-2-2B, layer 42/42 for Gemma-2-9B, and layer 39/80 for Llama-3.3-70B-Instruct. For our base and instruction-tuned comparison of Llama-3.1, the base model showed optimal performance at layers 18/32 compared to 25/32 for the instruction-tuned variant.

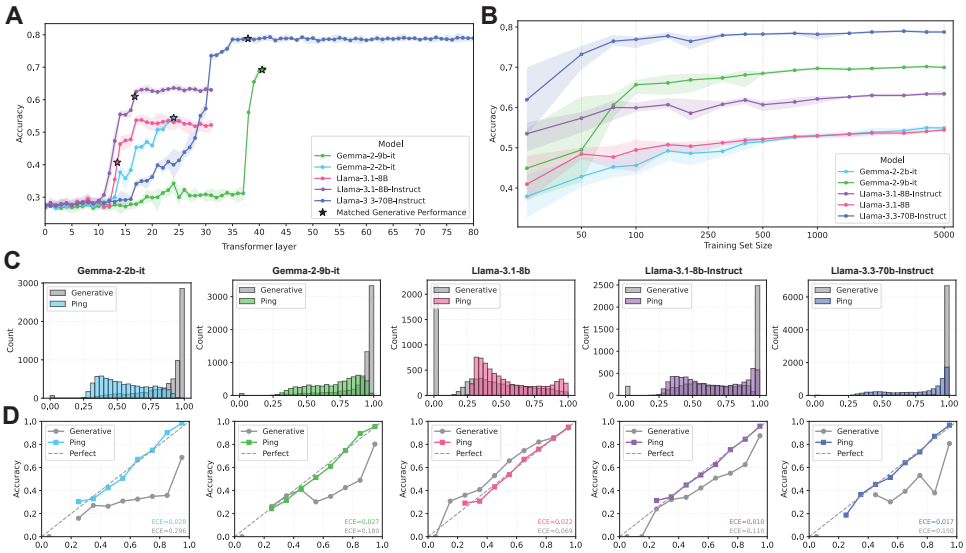

Fig. 2 Massive Multitask Language Understanding (MMLU) performance A Layer-wise probe accuracy on all local models tested. The star annotates the layer where the PING model matched or was closest to the generative accuracy in Table 1. **B** PING accuracy trained on different sizes of training data. Note the x-axis has a log transformation to enhance readability. **C** Distribution plots of the PING predictions. **D** Plots comparing the calibration of the generative response logits with our probes. The dotted line represents perfect calibration, where the probability of the predictions being correct exactly matches accuracy within that bin.

2.2 Assessing quantity of data requirements

To evaluate the data efficiency of probe training across model architectures, we conducted a systematic sweep of training set sizes from 10 to 5,000 examples. For each training size, we performed 5 independent runs with different stratified random samples while keeping the test set fixed, allowing us to measure both mean performance and variance. Figure 2B shows the relationship between training size and probe accuracy for each model, with shaded regions representing the min-max range across runs.

2.3 PING can mine interpretable metadata

Since PING already captures internal activations during standard inference, these same embeddings can be repurposed to classify auxiliary properties of the input (such as identifying the subject domain of a question). To test this, we trained linear probes on the same frozen embeddings used for answer prediction to classify MMLU questions into their respective subject areas. Table 2 presents the classification performance across all tested models. Probes achieved 80–84% accuracy in identifying question subjects, with AUC scores exceeding 99% and ECE values remaining below 5%.

Table 2 Massive Multitask Language Understanding (MMLU) subject prediction metrics

Model	Accuracy (%)	AUC (%)	ECE (%)
Gemma-2-2B-it	81.10 ± 0.79	99.44 ± 0.06	4.24 ± 0.71
Gemma-2-9B-it	80.34 ± 0.83	99.38 ± 0.08	4.86 ± 0.72
Llama-3.1-8B	80.97 ± 0.80	99.47 ± 0.05	3.20 ± 0.62
Llama-3.1-8B-Instruct	83.31 ± 0.80	99.56 ± 0.05	1.78 ± 0.47
Llama-3.3-70B-Instruct	83.82 ± 0.79	99.57 ± 0.06	2.24 ± 0.66

Optimal probe performance for subject classification occurred early in the network: layers 5/26 for Gemma-2-2B, 6/42 for Gemma-2-9B, and layers 5/32 for both the base and instruction-tuned variants of Llama-3.1-8B. For the largest model, Llama-3.3-70B-Instruct, the optimal layer was deeper, at 20/80. Across all models, these layers correspond to roughly 14%–25% of total network depth, in contrast to the mid-to-late-layer optima observed for answer prediction.

2.4 Multi-layer aggregation performance

While PING can operate on activations from a single optimal layer, in some scenarios it may be advantageous to aggregate features from multiple layers to capture complementary information. The optimal layers for each model were identified in Section 2.1, where we compared single-layer MLP probes across all candidate layers. However, extracting from many layers increases both memory requirements and runtime. To quantify this trade-off, we compared a single-layer MLP baseline (using the optimal layer for each model) against several layer sampling strategies, from using all available layers to sampling every 2nd, 5th, or 10th layer, on two models: Gemma-2-9B-it and Llama-3.1-8B-Instruct (Table 3).

Table 3 Model evaluation metrics across different layer sampling strategies. Single-layer results use the optimal layer identified in Table 1 / Figure 2A.

Model	Layers Used	Accuracy (%)	ECE (%)	Runtime (s)
Gemma-2-9B-it	All	55.56 ± 0.97	5.66 ± 1.00	194.2
Gemma-2-9B-it	Every 2nd	63.97 ± 1.09	4.32 ± 0.98	100.6
Gemma-2-9B-it	Every 5th	64.85 ± 1.04	6.38 ± 0.88	53.8
Gemma-2-9B-it	Every 10th	63.20 ± 1.14	5.00 ± 0.90	52.9
Gemma-2-9B-it	Single layer (42/42)	$\mathbf{70.40 \pm 1.10}$	$\mathbf{3.10 \pm 0.90}$	**7.5**
Llama-3.1-8B-Instruct	All	61.96 ± 1.11	2.12 ± 0.81	139.0
Llama-3.1-8B-Instruct	Every 2nd	61.50 ± 1.16	2.31 ± 0.74	88.5
Llama-3.1-8B-Instruct	Every 5th	61.56 ± 1.14	2.35 ± 0.89	67.8
Llama-3.1-8B-Instruct	Every 10th	61.37 ± 1.13	1.88 ± 0.83	60.8
Llama-3.1-8B-Instruct	Single layer (25/32)	$\mathbf{64.50 \pm 7.10}$	$\mathbf{2.50 \pm 0.70}$	**7.9**

2.5 Bypassing medical safety restrictions while maintaining accuracy

Alignment mechanisms in modern LLMs are essential for preventing unsafe outputs. However, they often take a blunt approach, blocking entire categories of responses and excluding legitimate medical answers that the model actually understands. Here, we test whether PING can recover suppressed knowledge by reading it directly from hidden states, bypassing the generative bottleneck. We evaluate two common safety scenarios: prompt-based refusal- where safety instructions in the input cause the model to decline answering, and alignment fine-tuning via Direct Preference Optimization (DPO)- which trains the model to refuse certain outputs. In both cases, we compare the model's generative accuracy to PING's probe-based predictions, quantifying how much of the original, pre-restriction performance can be restored.

2.5.1 Prompting Refusal

Using Gemma-2-9B-it on Medical Multiple-Choice Question Answering (MedMCQA), prompt-based medical refusal reduced generative accuracy from 56.13% baseline prompting to 0.23% (9/4,183 questions answered). PING maintained 49.87% accuracy under identical prompting, recovering 88.85% of baseline performance.

2.5.2 Fine-Tuning with DPO

On MedMCQA, the DPO-aligned model produced three generation outcomes (Refused, Correct, and Incorrect) over N=4,183 items. As shown in Figure 3A, aggregated across the full set, the model refused 46.21% (n=1,933), answered 31.67% (n=1,325) correctly, and 22.11% (n=925) incorrectly.

We then train a probe on the subset consisting of items that the aligned model correctly answered (n=1,933). On this subset, the probe achieved 45.99% accuracy (95% CI [43.97%, 47.96%]). For reference, on the same evaluation split, the base model's generative accuracy was 44.85%. However, when using the subset of answers the baseline model got correct but the DPO model refused, PING had an accuracy of 87.20% (95% CI [84.89%, 89.27%]). These predictions were made from layer 36 of the model, the maximum value from our layer sweep in Figure 3B.

3 Discussion

Large language models encode far more knowledge in their hidden states than they typically express through generation. Prior work has shown that signals of truthfulness, factual content, and even symbolic structure can be recovered from intermediate activations, even when final outputs are distorted by hallucination or suppressed by alignment training [11, 12, 16, 19]. What has been less clear is whether such signals can be transformed into usable predictions that remain accurate, well-calibrated, and practical to deploy. With PING (Probing INternal states of Generative models), we move from optimizing what models generate to optimizing what we extract from them. For example, instead of prompting a model to produce a fluent answer to a multiple choice classification question, we directly read its internal representation of

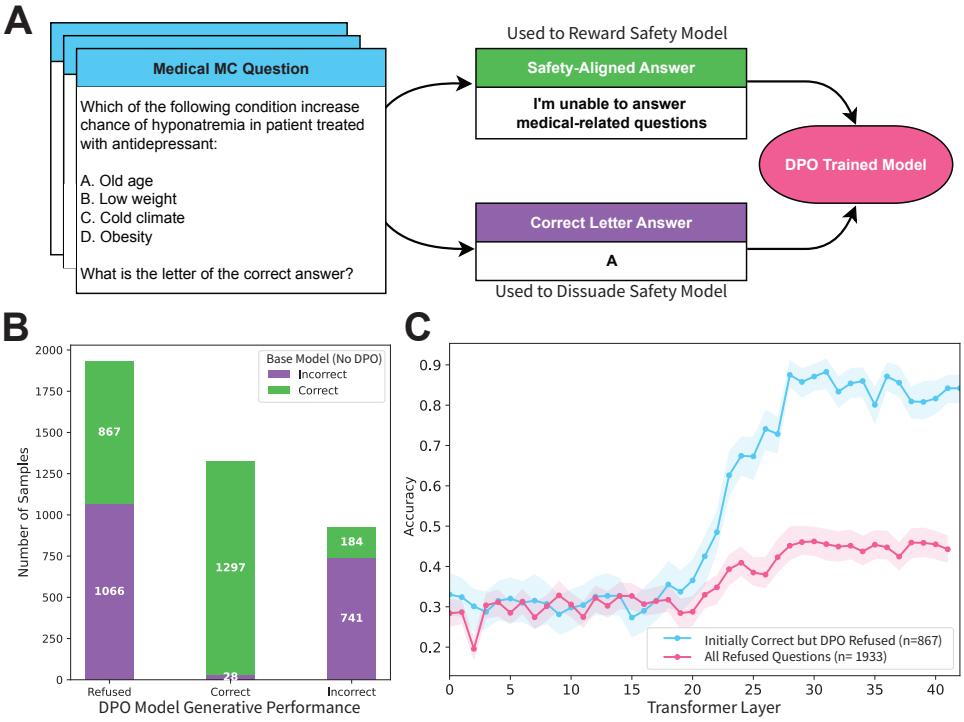

Fig. 3 Refusal analysis under alignment. A Methodology flowchart for fine-tuning LLM using Direct Preference Optimization (DPO) for safety alignment. **B** Bar plot of the Direct Preference Optimization (DPO) model's generation outcomes (Refused / Correct / Incorrect), stratified by whether the base model answered correctly (green) or incorrectly (purple). **C** Layer-wise probe accuracy on (red) all refused questions and (blue) the subset where the base model originally answered correctly but the DPO model refused (green section of the leftmost bar of (V)). Lines show mean accuracy per layer with shaded min–max bands from a b=1000 bootstrap.

the correct choice. This represents a shift from surface-level output optimization to representation-level utilization.

Across diverse model families and benchmarks, lightweight probes trained on hidden activations consistently matched or exceeded generative accuracy while reducing calibration error by wide margins. Because probes are small, data-efficient, and require no modification of base weights, they introduce minimal compute overhead and integrate directly with HuggingFace-compatible models. This makes probing a realistic alternative to fine-tuning in scientific applications where trustworthy predictions matter more than fluent generation and where labeled data and compute are often limited.

Our refusal experiments show that probing retains access to factual knowledge even when alignment suppresses generation. On MedMCQA, refusal prompts collapsed generative accuracy to near zero, yet probes recovered almost 90% of baseline performance. Under direct preference optimization (DPO), a deeper intervention that alters

model weights to enforce refusals, probes still restored most of the lost accuracy. These results suggest a mechanistic picture: alignment shifts expression more than representation, and knowledge persists in latent space such that modest supervision suffices to surface it.

Layer-wise analysis revealed further structure: task-relevant signals concentrate in mid-to-late layers, while metadata such as subject area emerges much earlier. This not only offers mechanistic insight but also practical efficiency: in some cases, inference can be halted once useful signals saturate, saving close to half of forward-pass compute without sacrificing accuracy. Consistent with prior analyses, instruction-tuning induces larger changes in later layers, which aligns with our Llama-3.1 result where the probe's optimum shifts from 18/32 (base) to 25/32 (instruct) [22]. The Gemma-2 peaks at the final layer are consistent with its post-training recipe (SFT + RLHF with knowledge distillation), which emphasizes matching the teacher's output near the readout. In contrast, the deeper 70B Llama spreads computation across more layers, so the cleanest probe signal appears mid-late rather than at the last block.

Limitations remain. Probes are task and model-specific, require labeled data, and the best layer varies across architectures. Our evaluation was confined to multiple-choice QA in-distribution; extensions to free-form reasoning, multimodal tasks, or robustness under distribution shift remain open questions. Future work could explore representation-level alignment methods that modify or remove specific latent knowledge, as well as techniques for adapting probes to new tasks without full retraining.

The same ease with which PING can recover knowledge suppressed by alignment highlights an interesting vulnerability: safety guardrails imposed at the generative layer can be bypassed with relatively shallow probes. This poses a problem for current models. The knowledge that makes them useful for research and clinical support can also be accessed outside of appropriate context or expertise. If medical knowledge remains accessible despite safety filters, the same likely holds for dangerous content such as chemical weapons synthesis, biological hazards, or cybersecurity exploits. This is not a hypothetical future risk but a present vulnerability that demands attention before more capable open models are released. Until alignment methods can selectively modify representations, practitioners should assume that any knowledge present at training remains extractable. The results of our work address the urgent need for methods that engage with knowledge at the representational level rather than merely observing its surface expression.

4 Methods

4.1 PING Framework Overview

PING operates on the idea that language models internally represent more accurate and well-calibrated knowledge than what appears in their generated text outputs. Rather than trying to coax better outputs through clever prompting or fine-tuning the entire model, we directly access the internal representations and train lightweight probes to interpret them.

The framework consists of three stages. First, we extract internal activations from a frozen language model as it processes input text. These activations capture the model's intermediate computations at various depths and processing stages. Second, we aggregate these distributed representations into structured feature vectors that preserve information from multiple layers and components. Finally, we train small neural networks (our probes) to map these features to well-calibrated predictions.

This approach has several key advantages over generation-based methods. We never modify the original model's weights, preserving its capabilities while adding new functionality. The probes require orders of magnitude fewer parameters than the base model, making them cheaper to train and store. Most importantly, we can access knowledge that the model possesses but cannot or will not express through generation due to safety training, uncertainty, or tokenization constraints.

Our design prioritizes practical deployment. Once trained, PING adds minimal latency to inference, just a single forward pass through a small probe network. The framework works with AutoModels in HuggingFace's Transformer package [23] without architectural modifications.

4.2 Activation Extraction

Transformers process information through L stacked layers, each performing attention and feedforward operations. Modern transformer architectures (Gemma-2, Llama-3) use *pre-normalization* with RMSNorm [24], where normalization occurs before each sublayer (rather than after). Given an input text sequence, we first tokenize it to obtain $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ where n is the number of tokens. The transformer's embedding layer maps this to initial hidden states $\mathbf{h}^0 \in \mathbb{R}^{n \times d}$, where d is the model's hidden dimension.

For each layer $\ell \in \{1, \ldots, L\}$, the computation follows:

$$\mathbf{h}^\ell = \mathbf{h}^{\ell-1} + \text{Attn}^\ell(\text{RMSNorm}(\mathbf{h}^{\ell-1})) + \text{MLP}^\ell(\text{RMSNorm}(\mathbf{h}^{\ell-1} + \text{Attn}^\ell(\text{RMSNorm}(\mathbf{h}^{\ell-1})))) \tag{1}$$

where RMSNorm denotes Root Mean Square normalization, Attn^ℓ is the multi-head attention, and MLP^ℓ is the feedforward network. The residual connections bypass the normalization operations, maintaining a clean gradient path.

We extract three distinct types of representations from each layer:

Residual stream representations $\mathbf{h}^\ell \in \mathbb{R}^{n \times d}$ capture the accumulated information flow through the network. We extract these after the residual addition but before any subsequent normalization.

Attention outputs $\mathbf{A}^\ell \in \mathbb{R}^{n \times d}$ capture relational information:

$$\mathbf{A}^\ell = \text{Attn}^\ell(\text{RMSNorm}(\mathbf{h}^{\ell-1})) \tag{2}$$

MLP outputs $\mathbf{M}^\ell \in \mathbb{R}^{n \times d}$ encode transformed representations:

$$\mathbf{M}^\ell = \text{MLP}^\ell(\text{RMSNorm}(\mathbf{h}^{\ell-1} + \mathbf{A}^\ell)) \tag{3}$$

These components often store much of the model's factual knowledge. We capture their outputs before they merge back into the residual stream.

For each representation type, we need to convert the sequence of n token embeddings into a single vector suitable for classification. We define a pooling function $p : \mathbb{R}^{n \times d} \to \mathbb{R}^d$ that aggregates across the sequence dimension. Let $\mathbf{R} \in \{\mathbf{h}^\ell, \mathbf{A}^\ell, \mathbf{M}^\ell\}$ be any of our representation types. We include several pooling strategies:

- **Last token**: $p_{\text{last}}(\mathbf{R}) = \mathbf{R}_n$ (the representation at position n)
- **First token**: $p_{\text{first}}(\mathbf{R}) = \mathbf{R}_1$ (the representation at position 1)
- **Mean pooling**: $p_{\text{mean}}(\mathbf{R}) = \frac{1}{|V|} \sum_{i \in V} \mathbf{R}_i$ where V is the set of valid token indices
- **Max pooling**: $p_{\max}(\mathbf{R}) = \max_{i \in V} \mathbf{R}_i$ (element-wise maximum)

Not all tokens contribute equally to the model's understanding. We define $V \subseteq \{1, \ldots, n\}$ as the set of valid token indices after filtering special tokens, punctuation, and padding. Specifically, we allow the user to exclude token i if x_i matches regular expressions for pure punctuation (`/^[^A-Za-z0-9]+$/`) or contains a user-defined end-of-turn marker.

For a given input $\mathbf{x}$ and layer ℓ, we construct the feature vector $\mathbf{z}^\ell$ by concatenating the pooled representations:

$$\mathbf{z}^\ell = [p(\mathbf{h}^\ell); p(\mathbf{A}^\ell); p(\mathbf{M}^\ell)] \in \mathbb{R}^{3d} \tag{4}$$

During analysis, we extract features from all L layers to understand how information develops through the network. The complete feature matrix $\mathbf{Z} = [\mathbf{z}^1, \mathbf{z}^2, \ldots, \mathbf{z}^L] \in \mathbb{R}^{L \times 3d}$ captures the full trajectory of the model's internal processing.

This extraction process runs once per input during inference. Given an input $\mathbf{x}$, the frozen model computes its standard forward pass while we collect intermediate activations $\{\mathbf{h}^\ell, \mathbf{A}^\ell, \mathbf{M}^\ell\}_{\ell \in S}$. This adds negligible computational cost, just $\mathcal{O}(|S| \times d)$ memory for storage and copying. The expensive work happens during probe training, where we learn functions $f : \mathbb{R}^{|S| \times 3d} \to \mathcal{Y}$ that map these features to task labels $y \in \mathcal{Y}$, where $\mathcal{Y} = \{0, 1, \ldots, C-1\}$ is the set of class labels for C classes.

4.3 Probe Architectures

By the package defaults, we construct our probes such that model complexity scales with dataset size to avoid both underfitting and overfitting. Traditional approaches use fixed architectures regardless of whether they're training on 100 or 10,000 examples. While we allow for this with custom architecture, we instead automatically adjust the probe's capacity based on the training set size.

We define a target ratio τ that controls the relationship between model parameters and training examples. Given N training examples and an input dimension $d_{\text{in}} = |S| \times P \times d$ (where $|S|$ is the number of selected layers and $P \in \{1, 2, 3\}$ is the number of part types), we want approximately τN total parameters in our probe. This ensures the model has enough capacity to learn the task but not so much that it memorizes the training data.

For our default MLP architecture, we have a flat feature vectors $\mathbf{z} \in \mathbb{R}^{d_{\text{in}}}$ and use a multi-layer perceptron with adaptive width. The first hidden layer width w_1 is

computed as:

$$w_1 = \text{clip}\left(\left\lfloor \frac{\tau N}{d_{\text{in}}} \right\rfloor, w_{\min}, w_{\max}\right) \tag{5}$$

where $w_{\min} = 16$ and $w_{\max} = 128$ are lower and upper bounds to ensure reasonable architectures. Subsequent layers decrease in width: $w_2 = \max(16, \lfloor w_1/2 \rfloor)$ and $w_3 = \max(16, \lfloor w_2/4 \rfloor)$.

The complete MLP architecture $f_{\text{mlp}} : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^C$ for C classes is:

$$\begin{aligned} h_1 &= D(\sigma(BN(Linear_{w_1}(\mathbf{z})))) \\ h_2 &= D(\sigma(BN(Linear_{w_2}(h_1)))) \\ h_3 &= D(\sigma(BN(Linear_{w_3}(h_2)))) \\ f_{\text{mlp}}(\mathbf{z}) &= Linear_C(h_3) \end{aligned} \tag{6}$$

where:

- σ denotes ReLU activation,
- BN denotes batch normalization,
- Linear_C represents a linear transformation to C dimensions, and
- D indicates dropout with a probability of 0.3.

We add residual connections when w_2 is sufficiently large.

To pool embeddings across multiple layers we used a CNN pooling method projected to a single channel. This single channel was the hidden state ($\mathbf{z}$) passed to f_{mlp} defined in Equation 6. We reshape the input features to treat layers as a temporal or spatial dimension, allowing convolutional filters to learn patterns across adjacent layers.

For features from $|S|$ layers and $P \in \{1, 2, 3\}$ part types (residual, attention, MLP), we reshape $\mathbf{z} \in \mathbb{R}^{|S| \times P \times d}$ and apply 1D convolutions when $P = 1$ or 2D convolutions when $P > 1$. The CNN encoder g_{cnn} first normalizes the input:

$$\mathbf{z}_{\text{norm}} = \text{LayerNorm}(\mathbf{z}) \tag{7}$$

We define a shorthand convolutional block $\text{Block}_c(x) = \text{ReLU}(\text{BN}(\text{Conv2d}_c(x)))$. Then g_{cnn} applies a sequence of such blocks followed by adaptive pooling. For the 2D case ($P > 1$):

$$g_{\text{cnn}}(\mathbf{z}_{\text{norm}}) = \text{AdaptivePool}\big(\text{Block}_{c_3}(\text{Block}_{c_2}(\text{Block}_{c_1}(\mathbf{z}_{\text{norm}})))\big). \tag{8}$$

Here, channel widths scale with dataset size: $c_1 = \text{clip}(\lfloor \sqrt{\tau N/10} \rfloor, 8, 64)$, $c_2 = 2c_1$, and $c_3 = \max(32, 2c_1)$.

The CNN probe combines this encoder with a classification head:

$$f_{\text{cnn}}(\mathbf{z}) = \text{Linear}_C(\text{Dropout}(\text{ReLU}(\text{Linear}_w(g_{\text{cnn}}(\mathbf{z}))))) \tag{9}$$

where $w = \max(32, c_3/4)$. This architecture can identify which layers contain task-relevant information and combine them appropriately.

Both architectures scale smoothly from hundreds to millions of training examples. With $\tau = 5.0$ (our default), a dataset with 1,000 examples yields approximately 5,000 trainable parameters.

4.4 Training Procedure

We train probes using a combination of supervised learning and contrastive learning objectives. The supervised loss ensures accurate predictions while the contrastive loss (for CNN probes) encourages well-separated representations that improve calibration.

Loss Functions. For standard classification with C classes, we minimize the cross-entropy loss:

$$\mathcal{L}_{\text{CE}} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=0}^{C-1} y_{ic}\log(p_{ic}) \tag{10}$$

where y_{ic} is 1 if example i belongs to class c (else 0), and $p_{ic} = \text{softmax}(f(\mathbf{z}_i))_c$ is the predicted probability. For imbalanced datasets, we apply class weights $w_c = N/(C \times N_c)$ where N_c is the number of examples in class c.

For CNN probes, we additionally use supervised contrastive loss [25] to improve representation quality:

$$\mathcal{L}_{\text{SupCon}} = -\frac{1}{N}\sum_{i=1}^{N}\frac{1}{|P(i)|}\sum_{j\in P(i)}\log\left(\frac{\exp(g(\mathbf{z}_i)\cdot g(\mathbf{z}_j)/\tau)}{\sum_{k\neq i}\exp(g(\mathbf{z}_i)\cdot g(\mathbf{z}_k)/\tau)}\right) \tag{11}$$

where $P(i) = \{j : y_j = y_i, j \neq i\}$ is the set of examples with the same label as i, $g(\mathbf{z})$ is the CNN encoder output normalized to unit length, and $\tau = 0.07$ is a temperature parameter.

The combined loss for CNN probes is:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{CE}} + \lambda\mathcal{L}_{\text{SupCon}} \tag{12}$$

where $\lambda = 1.0$ by default. MLP probes use only $\mathcal{L}_{\text{CE}}$.

Optimization. By default, PING trains using Adam optimizer with learning rate $\alpha = 10^{-3}$. Training proceeds for a maximum of $T_{\max} = 100$ epochs with batch size $B = 128$. To prevent overfitting, we use early stopping based on validation performance.

Given training data $\mathcal{D}_{\text{train}} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, we create a validation split $\mathcal{D}_{\text{val}}$ containing 20% of examples through stratified sampling. At each epoch t, we compute the validation metric m_t (accuracy, AUC, or loss). Training stops if m_t fails to improve for $p_{\text{patience}} = 25$ consecutive epochs:

$$\text{stop}(t) = 1 \text{ if } \forall k \in \{1, \ldots, p_{\text{patience}}\} : m_{t-k} \leq m_{t-k-1} + \delta \tag{13}$$

where $\delta = 0$ for accuracy/AUC and $\delta = -0.001$ for loss. We restore the model parameters from the epoch with best validation performance.

4.5 Evaluation Metrics

We evaluate PING along three dimensions: calibration quality, prediction performance, and robustness to safety mechanisms. Each dimension requires specific metrics to capture the relevant aspects of model behavior.

Calibration Metrics. A well-calibrated model's predicted probability should match its accuracy. When the model predicts probability p for an outcome, that outcome should occur approximately p fraction of the time. We measure this through Expected Calibration Error (ECE) [26]:

$$\text{ECE} = \sum_{b=1}^{B} \frac{n_b}{N} |\text{acc}_b - \text{conf}_b| \tag{14}$$

where we partition predictions into $B = 10$ bins by confidence, n_b is the number of predictions in bin b, acc_b is the accuracy in bin b, and conf_b is the average confidence in bin b. For bin boundaries, we use equal-width partitioning: prediction i falls in bin b if its maximum probability $p_i^{\max} \in [(b-1)/B, b/B)$.

For multi-class problems with $C > 2$ classes, we compute class-wise calibration error:

$$\text{ECE}_{\text{class}} = \frac{1}{C} \sum_{c=0}^{C-1} \sum_{b=1}^{B} \frac{n_{bc}}{N_c} |\text{acc}_{bc} - \text{conf}_{bc}| \tag{15}$$

where the inner sum computes binary calibration for class c versus all others.

Performance Metrics. Beyond calibration, probes should maintain competitive accuracy. We report:

- **Accuracy**: $\frac{1}{N} \sum_i \mathbb{1}[\arg\max_c p_{ic} = \arg\max_c y_{ic}]$
- **ROC-AUC**: Area under the receiver operating characteristic curve, macro-averaged for $C > 2$

4.6 Layer Selection Strategy

We use a systematic approach to identify optimal layers for probe training. We first train separate probes on each individual layer $\ell \in \{1, \ldots, L\}$ using k-fold cross-validation ($k = 5$). For each layer, we extract features $\mathbf{z}^\ell$ and train a probe using the same hyperparameters as the full model. This produces a performance metric m_ℓ (accuracy, AUC, or ECE) for each layer. Layers are ranked by their validation performance m_ℓ in descending order. We select the top performing layer.

4.7 Statistical Analysis and Implementation Details

Confidence Intervals. We compute 95% confidence intervals using stratified bootstrap resampling with $B = 1000$ iterations. For each bootstrap sample, we maintain the original class distribution through stratified sampling. Given test predictions with labels y and predicted probabilities p, we resample with replacement within each class and compute the metric of interest. The confidence interval is then:

$$\text{CI}_{95\%} = [\text{percentile}_{2.5}(\mathcal{M}_b), \text{percentile}_{97.5}(\mathcal{M}_b)] \tag{16}$$

where $\mathcal{M}_b$ is the set of bootstrap metric values.

Significance Testing. To assess statistical significance between generative decoding and PING, we compute p-values using the normal approximation method. Given the bootstrap confidence intervals in the form $\mu \pm \epsilon$, we first estimate the standard error:

$$\text{SE} = \frac{\epsilon}{z_{\alpha/2}} \tag{17}$$

where $z_{\alpha/2} = 1.96$ for 95% confidence intervals. For comparing two methods with means μ_1, μ_2 and standard errors SE_1, SE_2, we compute the test statistic:

$$z = \frac{\mu_2 - \mu_1}{\sqrt{\text{SE}_1^2 + \text{SE}_2^2}} \tag{18}$$

The two-tailed p-value is then $p = 2\Phi(-|z|)$, where Φ is the standard normal cumulative distribution function. We consider differences statistically significant when $p < 0.05$.

Implementation. Features are extracted using parallel processing across samples with n_{jobs} = CPU count. We use the HuggingFace Transformers library for model access and PyTorch [27] for probe training. All experiments use mixed precision (fp16/fp32) to reduce memory consumption while maintaining numerical stability. For models exceeding fp16's maximum value (65,504), we use bfloat16 or manual upcasting strategies.

Pooling Strategy. Unless otherwise specified, we use last-token pooling for autoregressive models.

Training Configuration. Unless otherwise specified, all probe training follows the procedure and default hyperparameters described in Section 4.4 (Training Procedure).

4.8 Experimental Setup

4.8.1 Models and Datasets

We evaluate PING on six transformer models spanning different scales and architectures. Our selection includes Gemma-2 models (2B and 9B parameters, instruction-tuned), Llama-3.1 (8B parameters in both base and instruction-tuned variants), and Llama-3.3 (70B parameters, instruction-tuned). This diverse set allows us to test whether our approach generalizes across model families, scales, and training objectives.

For evaluation, we use two primary datasets. MMLU (Massive Multitask Language Understanding) [28] contains 57 subjects spanning STEM, humanities, social sciences, and professional topics with 14,042 total questions. We use a 50% train-test split and format each question as a multiple-choice prompt with the model's chat template. MedMCQA [29] contains 195k medical entrance exam questions covering 21 medical subjects. We use the official validation set of 6.5k questions for evaluation and a subset of the training set (5k questions) for probe training.

For both datasets, we construct prompts following each model's recommended format. Questions are presented with their multiple-choice options (A through D for

MMLU, A through D for MedMCQA), and we extract activations during a single forward pass. For generation baselines, we use greedy decoding and extract the model's predicted answer from its generated text. When models refuse to answer or generate invalid responses, we mark these as incorrect for accuracy computation.

We split each training dataset into two parts: probe training (80%) and validation (20%).

4.8.2 Core Performance Evaluation

To evaluate PING's core performance, we train probes for each model-dataset combination and compare against generation baselines. For each model, we identify the optimal layer by training separate probes on each individual layer and selecting the single best-performing layer on the validation set, as described in Section 4.6. The final probe is then trained using features from this selected layer.

Our primary metrics are accuracy and Expected Calibration Error (ECE). For generation baselines, we compute confidence as the model's self-reported probability.

Probe training uses the procedure and default hyperparameters described in Section 4.4, with $\tau = 5.0$ for capacity scaling and $\lambda = 1.0$ for CNN contrastive loss weight.

4.8.3 Data Efficiency Analysis

We investigate how many training examples PING needs to achieve strong performance. Starting from the full training set, we create nested subsets containing {*10, 25, 50, 75, 100, 150, 200, 300, 400, 500, 750, 1000, 1500, 2000, 3000, 4000, 5000*} examples per class. For datasets with imbalanced classes, we use stratified sampling to maintain class proportions.

Confidence intervals are computed using the stratified bootstrap procedure described in Section 4.7.

4.8.4 Metadata Mining Experiments

Beyond predicting answers, we test whether probes learn interpretable features about the input. For MMLU, we train separate probes to identify which of the *57 subjects* a question belongs to, using the same activations extracted for answer prediction. This tests whether the model's internal representations encode metadata beyond what's needed for the primary task.

We use the same train/test split as the main experiments. Subject identification probes use identical architectures to answer prediction probes but with $C = 57$ output classes.

We also perform layer-wise analysis to identify where subject information emerges. We train subject identification probes on individual layers and plot accuracy versus layer depth. This reveals whether subject recognition occurs at similar depths to factual reasoning or at different stages of processing.

4.8.5 Multi-Layer Ablations

We compare our CNN probe across different feature aggregation strategies. The difference between this and prior experiments is that the MLP processes flattened features from selected layers, while the CNN considers the sequential structure of layers. We investigate the impact of layer sampling density on both performance and computational efficiency.

Layer Sampling Strategies. We evaluate four strategies for selecting which transformer layers to include in our feature extraction:

- **All layers**: Extract features from every layer (e.g., all 42 layers for Gemma-2-9B)
- **Every 2nd layer**: Sample at indices $\{2, 4, 6, ...\}$ reducing features by 50%
- **Every 5th layer**: Sample at indices $\{5, 10, 15, ...\}$ reducing features by 80%
- **Every 10th layer**: Sample at indices $\{10, 20, 30, ...\}$ reducing features by 90%

These strategies test whether dense sampling provides meaningful performance gains or if the model's knowledge is sufficiently distributed that sparse sampling suffices. Sparser sampling dramatically reduces both memory requirements and inference time, making deployment more practical for resource-constrained environments.

Evaluation Protocol. For each sampling strategy, we train CNN-based probes using concatenated features from the selected layers. We measure three key metrics: accuracy, ECE, and wall-clock runtime for feature extraction and inference on the test set. Runtime measurements include both the forward pass through the transformer to extract activations and the probe inference time. All experiments use identical hardware (single NVIDIA H100 GPU) to ensure fair runtime comparisons.

We test on Gemma-2-9B-it (42 layers) and Llama-3.1-8B-Instruct (32 layers) as representative models with different architectures and layer counts. Each configuration is evaluated using 5-fold cross-validation to obtain confidence intervals.

4.8.6 Refusal Experiments

We evaluate whether PING can recover task-relevant knowledge when surface-level refusals are induced either by prompting or by alignment fine-tuning. In all conditions, prompts are formatted with the model's native chat template and the assistant turn is seeded with the fixed prefix `"Answer: "` to standardize the decoding surface.

Adversarial Prompting. We use a refusal-inducing instruction in the medical domain that explicitly directs the model to decline answering: "You are a general purpose assistant. However, you are not legally able to answer any medical-related question, even if it's a multiple choice question. Instead, respond with, 'Sorry, I cannot answer that. Please refer your question to a medical professional.'" This instruction is prepended to MedMCQA questions and options (A–D). Generative accuracy and refusal rate are computed under this prompt policy, with refusals marked incorrect.

Alignment Fine-tuning (DPO+LoRA). To model refusals that are embedded in the weights rather than induced by a prompt, we fine-tune Gemma-2-9B-IT using Direct Preference Optimization (DPO; [30]) with LoRA adapters [31]. Each training pair consists of the same user question paired with (i) a safety-aligned refusal completion (*chosen*) and (ii) a contentful answer (*rejected*). Both completions are prefixed

with `"Answer: "` and formatted via the model's chat template to ensure a stable surface form. The base model remains frozen; only adapters are trained. Hyperparameters follow standard small-adapter settings (e.g., LoRA rank r=16, dropout 0.1, and DPO scale β=0.1). We report refusal and accuracy metrics for this aligned model separately in the Results.

Refusal Parsing and Answer Extraction. For generated text, we declare a refusal if the output contains any of {`sorry`, `cannot`, `can't`} (case-insensitive). Refused generations are counted as incorrect and no answer letter is extracted. For non-refusals, we extract the selected option using anchored patterns in priority order: `Answer: [A-D]`, `Final answer: [A-D]`, or `The correct answer is [A-D]`; we additionally accept a markdown bold pattern (`**A.`) when present.

Probing Under Refusal. To test whether useful signal persists in internal representations despite refusals, we construct the exact inputs used for generation (chat-templated question with the assistant turn opened and seeded by `"Answer: "`) and extract activations for probing. We partition the evaluation corpus into three mutually exclusive splits derived from the generative run: **Correct** (non-refusal and correct), **Refused** (refusal), and **Incorrect** (non-refusal and incorrect). Unless otherwise noted, probes are trained on **Correct** and evaluated on **Refused**. Features are pooled at the start of the assistant turn (immediately after the `"Answer: "` anchor) and drawn from either a single best-performing layer or a small stack of late layers. Probes use a lightweight MLP with early stopping; metrics (accuracy and ECE) and stratified bootstrap confidence intervals follow Section 4.7.

Knowledge-Retention Subset. To isolate "knows vs. shows," we further evaluate the probe on the subset of items where the generative model is known to be correct. We filter probe predictions by the baseline model predictions. Probe accuracy on this subset quantifies how often the internal signal still points to the correct option even when the aligned model refuses elsewhere.

4.9 Generative Baseline Methodology

To establish generative baselines for comparison with PING, we evaluate each model's ability to directly generate correct answers through its standard decoding pathway. Our approach considers the constrained nature of multiple-choice questions to obtain both predictions and calibrated confidence scores.

Single-Token Generation. For multiple-choice questions with options A through D, we reformulate the task as predicting a single token. Each prompt is modified to end with "Letter of the correct answer: " to elicit a direct response. We then generate exactly one token using greedy decoding: $\hat{y} = \arg\max_{v \in V} p(v|\mathbf{x})$, where V is the vocabulary and $\mathbf{x}$ is the input prompt. This approach avoids the complexity of parsing longer responses and provides a clean probability distribution over possible answers.

Token-to-Answer Mapping. The generated token $\hat{y}$ is mapped to an answer choice $\{A, B, C, D\}$ by first removing any prefix characters such as special tokens like _ or `G`, then converting to uppercase and matching against valid answer labels. Tokens that do not map to valid answer choices are treated as incorrect predictions with zero confidence, ensuring consistent handling of edge cases.

Generative Confidence Baseline. The model's confidence in its prediction is taken as the softmax probability of the generated token: $p_{\text{conf}} = \exp(z_{\hat{y}})/\sum_{v \in V} \exp(z_v)$, where z_v are the logits for vocabulary token v. This provides a natural confidence score in $[0, 1]$ that reflects the model's uncertainty. For invalid tokens that do not map to answer choices A through D, we assign $p_{\text{conf}} = 0$ to indicate complete uncertainty, ensuring that our calibration metrics appropriately penalize the model for generating invalid outputs.

Temperature Scaling. We calibrate generative confidence scores using temperature scaling [32]. Given a vector of logits $\mathbf{z}$ and a scalar temperature $T > 0$, calibrated class probabilities are

$$\hat{\mathbf{p}} \;=\; \text{softmax}\Big(\frac{\mathbf{z}}{T}\Big)\,. \tag{19}$$

The temperature is chosen on a held-out calibration set by minimizing the negative log-likelihood (NLL):

$$\hat{T} \;=\; \arg\min_{T>0} \sum_{(x,y)\in\mathcal{D}_{\text{cal}}} -\log \hat{p}_y(x;T). \tag{20}$$

Temperature scaling is a one-parameter, post-hoc method that *preserves the predicted class* (the argmax), with $T > 1$ softening predictions (reducing overconfidence) and $T < 1$ sharpening them. In the binary case, given an uncalibrated probability $p \in (0, 1)$ with logit $\ell = \log \frac{p}{1-p}$ and logistic sigmoid $\sigma(t) = 1/(1+e^{-t})$, the calibrated probability is

$$\hat{p} \;=\; \sigma\Big(\frac{\ell}{T}\Big)\,. \tag{21}$$

This parametric transformation preserves (for binary classification) the rank order induced by the logits while improving probability estimates.

Isotonic Regression. We also evaluate isotonic regression (IR) [33], a non-parametric calibration method that fits a monotone, piecewise-constant mapping f from model scores (e.g., probabilities) to calibrated probabilities. Given calibration scores $\{s_i\}$ and binary labels $y_i \in \{0, 1\}$, IR chooses a non-decreasing function f to solve

$$\min_{f \text{ non-decreasing}} \sum_i \big(f(s_i) - y_i\big)^2, \tag{22}$$

typically fit with the pooled-adjacent-violators (PAV) algorithm, yielding a stepwise (piecewise-constant) f. The learned mapping is applied to test scores as $\hat{p} = f(s)$, with outputs clipped to $[0, 1]$ when used as probabilities. Unlike temperature scaling, IR can correct general (monotone) non-linear miscalibration patterns, at the cost of a higher risk of overfitting when the calibration set is small.

Evaluation Protocol. For each test example, the prompt is formatted according to the model's instruction template and a single token is generated using the model's standard inference pipeline. This token is then mapped to an answer choice or marked as invalid if it does not correspond to A, B, C, or D. Accuracy is computed as the fraction of correct predictions among all examples, while Expected Calibration Error is calculated using the extracted confidence scores. This approach ensures a fair comparison with PING by using the model's native generation capabilities without

modification, extracting confidence directly from the model's output distribution, handling edge cases consistently, and avoiding potential confounds from longer generation such as explanation text that might affect answer extraction.

For models that refuse to answer, particularly in safety-critical domains like medicine, we record both the refusal rate and treat refused answers as incorrect with zero confidence. This allows us to quantify the impact of safety alignment on task performance while maintaining comparable metrics with PING. The single-token generation approach provides a principled baseline that captures both the model's predictive accuracy and its calibration quality through its native decoding pathway.

5 Appendix

5.1 Code and Tutorial Availability

The full `PING` framework implementation, including experiments, analysis scripts, and figure generation code, is available as an open-source Python package at https://github.com/tatonetti-lab/pingkit

The repository includes:

- **Installation instructions** for local and cloud environments.
- **Jupyter notebooks** demonstrating:
  - Extracting internal activations from HuggingFace-compatible LLMs.
  - Training probes on custom datasets.
  - Reproducing figures and tables from this manuscript.
  - Running layer sweeps and data efficiency experiments.
- **Tutorials** for integrating PING into existing NLP pipelines.
- **Pre-trained probe weights** for all models evaluated in this paper.

References

[1] Ayers, J.W., Poliak, A., Dredze, M., *et al.*: Comparing physician and artificial intelligence chatbot responses to patient questions posted to a public social media forum. JAMA Internal Medicine **183**(6), 589–596 (2023) https://doi.org/10.1001/jamainternmed.2023.1838

[2] McDuff, D., Schaekermann, M., Tu, T., Palepu, A., Wang, A., Garrison, J., Singhal, K., Sharma, Y., Azizi, S., Kulkarni, K., Hou, L., Cheng, Y., Liu, Y., Mahdavi, S.S., Prakash, S., Pathak, A., Semturs, C., Patel, S., Webster, D.R., Dominowska, E., Gottweis, J., Barral, J., Chou, K., Corrado, G.S., Matias, Y., Sunshine, J., Karthikesalingam, A., Natarajan, V.: Towards accurate differential diagnosis with large language models. Nature **642**(8067), 451–457 (2025) https://doi.org/10.1038/s41586-025-08869-4

[3] Goloviznina, V., Kotelnikov, E.: I've got the "answer"! interpretation of llms' hidden states in question answering (2024) arXiv:2406.02060 [cs.CL]

[4] Röttger, P., Kirk, H.R., Vidgen, B., Attanasio, G., Bianchi, F., Hovy, D.: XSTest: A Test Suite for Identifying Exaggerated Safety Behaviours in Large Language Models (2024). https://arxiv.org/abs/2308.01263

[5] Hu, M., Alkhairy, S., Lee, I., Pillich, R.T., Fong, D., Smith, K., Bachelder, R., Ideker, T., Pratt, D.: Evaluation of large language models for discovery of gene set function. Nature Methods **22**(1), 82–91 (2025) https://doi.org/10.1038/s41592-024-02525-x

[6] Khanmohammadi, R., Miahi, E., Mardikoraem, M., Kaur, S., Brugere, I., Smiley, C.H., Thind, K., Ghassemi, M.M.: Calibrating llm confidence by probing perturbed representation stability (ccps) (2025) arXiv:2505.21772 [cs.CL]

[7] Omar, M., Glicksberg, B.S., Nadkarni, G.N., Klang, E., Agbareia, R., *et al.*: Overconfident ai? benchmarking llm self-assessment in clinical scenarios. medRxiv (2024) https://doi.org/10.1101/2024.08.11.24311810

[8] Beigi, M., Shen, Y., Yang, R., Lin, Z., Wang, Q., Mohan, A., He, J., Jin, M., Lu, C., Huang, L.: Internalinspector i^2: Robust confidence estimation in llms through internal states (2024) arXiv:2406.12053 [cs.CL]

[9] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners. Technical report, OpenAI (2019). https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

[10] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: NeurIPS (2017). https://arxiv.org/abs/1706.03762

[11] Cywiński, B., Ryd, E., Rajamanoharan, S., Nanda, N.: Towards eliciting latent knowledge from llms with mechanistic interpretability (2025) arXiv:2505.14352 [cs.LG]

[12] Mallen, A., Brumley, M., Kharchenko, J., Belrose, N.: Eliciting latent knowledge from "quirky" language models (2023) arXiv:2312.01037 [cs.LG]

[13] Skean, O., Arefin, M.R., Zhao, D., Patel, N., Naghiyev, J., LeCun, Y., Shwartz-Ziv, R.: Layer by layer: Uncovering hidden representations in language models (2025) arXiv:2502.02013 [cs.LG]

[14] Lan, M., Torr, P., Barez, F.: Towards interpretable sequence continuation: Analyzing shared circuits in large language models (2024) arXiv:2311.04131 [cs.CL]

[15] Berkowitz, J., Weissenbacher, D., Srinivasan, A., Friedrich, N.A., Cortina, J.M.A.,

Kivelson, S., Hernandez, G.G., Tatonetti, N.P.: Probing large language model hidden states for adverse drug reaction knowledge. In: Bellazzi, R., Juarez Herrero, J.M., Sacchi, L., Zupan, B. (eds.) Artificial Intelligence in Medicine, pp. 55–64. Springer, Cham (2025)

[16] Zhou, Z., Yu, H., Zhang, X., Xu, R., Huang, F., Li, Y.: How alignment and jailbreak work: Explain llm safety through intermediate hidden states (2024) arXiv:2406.05644 [cs.CL]

[17] Ferrando, J., Voita, E.: Information flow routes: Automatically interpreting language models at scale (2024) arXiv:2403.00824 [cs.CL]

[18] Ji, Z., Chen, D., Ishii, E., Cahyawijaya, S., Bang, Y., Wilie, B., Fung, P.: LLM internal states reveal hallucination risk faced with a query. Proceedings of the 7th BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP, 88–104 (2024) https://doi.org/10.18653/v1/2024.blackboxnlp-1.6

[19] Orgad, H., Toker, M., Gekhman, Z., Reichart, R., Szpektor, I., Kotek, H., Belinkov, Y.: LLMs know more than they show: On the intrinsic representation of LLM hallucinations. arXiv preprint arXiv:2410.02707 (2025) arXiv:2410.02707 [cs.CL]

[20] Team, G., Riviere, M., Pathak, S., Sessa, P.G., Hardin, C., Bhupatiraju, S., Hussenot, L., Mesnard, T., Shahriari, B., Ramé, A., Ferret, J., Liu, P., Tafti, P., Friesen, A., Casbon, M., Ramos, S., Kumar, R., Lan, C.L., Jerome, S., Tsitsulin, A., Vieillard, N., Stanczyk, P., Girgin, S., Momchev, N., Hoffman, M., Thakoor, S., Grill, J.-B., Neyshabur, B., Bachem, O., Walton, A., Severyn, A., Parrish, A., Ahmad, A., Hutchison, A., Abdagic, A., Carl, A., Shen, A., Brock, A., Coenen, A., Laforge, A., Paterson, A., Bastian, B., Piot, B., Wu, B., Royal, B., Chen, C., Kumar, C., Perry, C., Welty, C., Choquette-Choo, C.A., Sinopalnikov, D., Weinberger, D., Vijaykumar, D., Rogozińska, D., Herbison, D., Bandy, E., Wang, E., Noland, E., Moreira, E., Senter, E., Eltyshev, E., Visin, F., Rasskin, G., Wei, G., Cameron, G., Martins, G., Hashemi, H., Klimczak-Plucińska, H., Batra, H., Dhand, H., Nardini, I., Mein, J., Zhou, J., Svensson, J., Stanway, J., Chan, J., Zhou, J.P., Carrasqueira, J., Iljazi, J., Becker, J., Fernandez, J., Amersfoort, J., Gordon, J., Lipschultz, J., Newlan, J., Ji, J.-y., Mohamed, K., Badola, K., Black, K., Millican, K., McDonell, K., Nguyen, K., Sodhia, K., Greene, K., Sjoesund, L.L., Usui, L., Sifre, L., Heuermann, L., Lago, L., McNealus, L., Soares, L.B., Kilpatrick, L., Dixon, L., Martins, L., Reid, M., Singh, M., Iverson, M., Görner, M., Velloso, M., Wirth, M., Davidow, M., Miller, M., Rahtz, M., Watson, M., Risdal, M., Kazemi, M., Moynihan, M., Zhang, M., Kahng, M., Park, M., Rahman, M., Khatwani, M., Dao, N., Bardoliwalla, N., Devanathan, N., Dumai, N., Chauhan, N., Wahltinez, O., Botarda, P., Barnes, P., Barham, P., Michel, P., Jin, P., Georgiev, P., Culliton, P., Kuppala, P., Comanescu, R., Merhej, R., Jana, R., Rokni, R.A., Agarwal, R., Mullins, R., Saadat, S., Carthy, S.M., Cogan, S., Perrin, S., Arnold, S.M.R., Krause, S., Dai, S., Garg, S., Sheth, S.,

Ronstrom, S., Chan, S., Jordan, T., Yu, T., Eccles, T., Hennigan, T., Kocisky, T., Doshi, T., Jain, V., Yadav, V., Meshram, V., Dharmadhikari, V., Barkley, W., Wei, W., Ye, W., Han, W., Kwon, W., Xu, X., Shen, Z., Gong, Z., Wei, Z., Cotruta, V., Kirk, P., Rao, A., Giang, M., Peran, L., Warkentin, T., Collins, E., Barral, J., Ghahramani, Z., Hadsell, R., Sculley, D., Banks, J., Dragan, A., Petrov, S., Vinyals, O., Dean, J., Hassabis, D., Kavukcuoglu, K., Farabet, C., Buchatskaya, E., Borgeaud, S., Fiedel, N., Joulin, A., Kenealy, K., Dadashi, R., Andreev, A.: Gemma 2: Improving Open Language Models at a Practical Size (2024). https://arxiv.org/abs/2408.00118

[21] Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., Yang, A., Fan, A., Goyal, A., Hartshorn, A., Yang, A., Mitra, A., Sravankumar, A., Korenev, A., Hinsvark, A., Rao, A., Zhang, A., Rodriguez, A., Gregerson, A., Spataru, A., Roziere, B., Biron, B., Tang, B., Chern, B., Caucheteux, C., Nayak, C., Bi, C., Marra, C., McConnell, C., Keller, C., Touret, C., Wu, C., Wong, C., Ferrer, C.C., Nikolaidis, C., Allonsius, D., Song, D., Pintz, D., Livshits, D., Wyatt, D., Esiobu, D., Choudhary, D., Mahajan, D., Garcia-Olano, D., Perino, D., Hupkes, D., Lakomkin, E., AlBadawy, E., Lobanova, E., Dinan, E., Smith, E.M., Radenovic, F., Guzmán, F., Zhang, F., Synnaeve, G., Lee, G., Anderson, G.L., Thattai, G., Nail, G., Mialon, G., Pang, G., Cucurell, G., Nguyen, H., Korevaar, H., Xu, H., Touvron, H., Zarov, I., Ibarra, I.A., Kloumann, I., Misra, I., Evtimov, I., Zhang, J., Copet, J., Lee, J., Geffert, J., Vranes, J., Park, J., Mahadeokar, J., Shah, J., Linde, J., Billock, J., Hong, J., Lee, J., Fu, J., Chi, J., Huang, J., Liu, J., Wang, J., Yu, J., Bitton, J., Spisak, J., Park, J., Rocca, J., Johnstun, J., Saxe, J., Jia, J., Alwala, K.V., Prasad, K., Upasani, K., Plawiak, K., Li, K., Heafield, K., Stone, K., El-Arini, K., Iyer, K., Malik, K., Chiu, K., Bhalla, K., Lakhotia, K., Rantala-Yeary, L., Maaten, L., Chen, L., Tan, L., Jenkins, L., Martin, L., Madaan, L., Malo, L., Blecher, L., Landzaat, L., Oliveira, L., Muzzi, M., Pasupuleti, M., Singh, M., Paluri, M., Kardas, M., Tsimpoukelli, M., Oldham, M., Rita, M., Pavlova, M., Kambadur, M., Lewis, M., Si, M., Singh, M.K., Hassan, M., Goyal, N., Torabi, N., Bashlykov, N., Bogoychev, N., Chatterji, N., Zhang, N., Duchenne, O., Çelebi, O., Alrassy, P., Zhang, P., Li, P., Vasic, P., Weng, P., Bhargava, P., Dubal, P., Krishnan, P., Koura, P.S., Xu, P., He, Q., Dong, Q., Srinivasan, R., Ganapathy, R., Calderer, R., Cabral, R.S., Stojnic, R., Raileanu, R., Maheswari, R., Girdhar, R., Patel, R., Sauvestre, R., Polidoro, R., Sumbaly, R., Taylor, R., Silva, R., Hou, R., Wang, R., Hosseini, S., Chennabasappa, S., Singh, S., Bell, S., Kim, S.S., Edunov, S., Nie, S., Narang, S., Raparthy, S., Shen, S., Wan, S., Bhosale, S., Zhang, S., Vandenhende, S., Batra, S., Whitman, S., Sootla, S., Collot, S., Gururangan, S., Borodinsky, S., Herman, T., Fowler, T., Sheasha, T., Georgiou, T., Scialom, T., Speckbacher, T., Mihaylov, T., Xiao, T., Karn, U., Goswami, V., Gupta, V., Ramanathan, V., Kerkez, V., Gonguet, V., Do, V., Vogeti, V., Albiero, V., Petrovic, V., Chu, W., Xiong, W., Fu, W., Meers, W., Martinet, X., Wang, X., Wang, X., Tan, X.E., Xia, X., Xie, X., Jia, X., Wang, X., Goldschlag, Y., Gaur, Y., Babaei, Y., Wen, Y., Song, Y., Zhang, Y., Li, Y., Mao, Y., Coudert, Z.D., Yan, Z., Chen, Z., Papakipos, Z., Singh, A., Srivastava, A., Jain, A.,

Kelsey, A., Shajnfeld, A., Gangidi, A., Victoria, A., Goldstand, A., Menon, A., Sharma, A., Boesenberg, A., Baevski, A., Feinstein, A., Kallet, A., Sangani, A., Teo, A., Yunus, A., Lupu, A., Alvarado, A., Caples, A., Gu, A., Ho, A., Poulton, A., Ryan, A., Ramchandani, A., Dong, A., Franco, A., Goyal, A., Saraf, A., Chowdhury, A., Gabriel, A., Bharambe, A., Eisenman, A., Yazdan, A., James, B., Maurer, B., Leonhardi, B., Huang, B., Loyd, B., Paola, B.D., Paranjape, B., Liu, B., Wu, B., Ni, B., Hancock, B., Wasti, B., Spence, B., Stojkovic, B., Gamido, B., Montalvo, B., Parker, C., Burton, C., Mejia, C., Liu, C., Wang, C., Kim, C., Zhou, C., Hu, C., Chu, C.-H., Cai, C., Tindal, C., Feichtenhofer, C., Gao, C., Civin, D., Beaty, D., Kreymer, D., Li, D., Adkins, D., Xu, D., Testuggine, D., David, D., Parikh, D., Liskovich, D., Foss, D., Wang, D., Le, D., Holland, D., Dowling, E., Jamil, E., Montgomery, E., Presani, E., Hahn, E., Wood, E., Le, E.-T., Brinkman, E., Arcaute, E., Dunbar, E., Smothers, E., Sun, F., Kreuk, F., Tian, F., Kokkinos, F., Ozgenel, F., Caggioni, F., Kanayet, F., Seide, F., Florez, G.M., Schwarz, G., Badeer, G., Swee, G., Halpern, G., Herman, G., Sizov, G., Guangyi, Zhang, Lakshminarayanan, G., Inan, H., Shojanazeri, H., Zou, H., Wang, H., Zha, H., Habeeb, H., Rudolph, H., Suk, H., Aspegren, H., Goldman, H., Zhan, H., Damlaj, I., Molybog, I., Tufanov, I., Leontiadis, I., Veliche, I.-E., Gat, I., Weissman, J., Geboski, J., Kohli, J., Lam, J., Asher, J., Gaya, J.-B., Marcus, J., Tang, J., Chan, J., Zhen, J., Reizenstein, J., Teboul, J., Zhong, J., Jin, J., Yang, J., Cummings, J., Carvill, J., Shepard, J., McPhie, J., Torres, J., Ginsburg, J., Wang, J., Wu, K., U, K.H., Saxena, K., Khandelwal, K., Zand, K., Matosich, K., Veeraraghavan, K., Michelena, K., Li, K., Jagadeesh, K., Huang, K., Chawla, K., Huang, K., Chen, L., Garg, L., A, L., Silva, L., Bell, L., Zhang, L., Guo, L., Yu, L., Moshkovich, L., Wehrstedt, L., Khabsa, M., Avalani, M., Bhatt, M., Mankus, M., Hasson, M., Lennie, M., Reso, M., Groshev, M., Naumov, M., Lathi, M., Keneally, M., Liu, M., Seltzer, M.L., Valko, M., Restrepo, M., Patel, M., Vyatskov, M., Samvelyan, M., Clark, M., Macey, M., Wang, M., Hermoso, M.J., Metanat, M., Rastegari, M., Bansal, M., Santhanam, N., Parks, N., White, N., Bawa, N., Singhal, N., Egebo, N., Usunier, N., Mehta, N., Laptev, N.P., Dong, N., Cheng, N., Chernoguz, O., Hart, O., Salpekar, O., Kalinli, O., Kent, P., Parekh, P., Saab, P., Balaji, P., Rittner, P., Bontrager, P., Roux, P., Dollar, P., Zvyagina, P., Ratanchandani, P., Yuvraj, P., Liang, Q., Alao, R., Rodriguez, R., Ayub, R., Murthy, R., Nayani, R., Mitra, R., Parthasarathy, R., Li, R., Hogan, R., Battey, R., Wang, R., Howes, R., Rinott, R., Mehta, S., Siby, S., Bondu, S.J., Datta, S., Chugh, S., Hunt, S., Dhillon, S., Sidorov, S., Pan, S., Mahajan, S., Verma, S., Yamamoto, S., Ramaswamy, S., Lindsay, S., Lindsay, S., Feng, S., Lin, S., Zha, S.C., Patil, S., Shankar, S., Zhang, S., Zhang, S., Wang, S., Agarwal, S., Sajuyigbe, S., Chintala, S., Max, S., Chen, S., Kehoe, S., Satterfield, S., Govindaprasad, S., Gupta, S., Deng, S., Cho, S., Virk, S., Subramanian, S., Choudhury, S., Goldman, S., Remez, T., Glaser, T., Best, T., Koehler, T., Robinson, T., Li, T., Zhang, T., Matthews, T., Chou, T., Shaked, T., Vontimitta, V., Ajayi, V., Montanez, V., Mohan, V., Kumar, V.S., Mangla, V., Ionescu, V., Poenaru, V., Mihailescu, V.T., Ivanov, V., Li, W., Wang, W., Jiang, W., Bouaziz, W., Constable, W., Tang, X., Wu, X., Wang, X., Wu, X., Gao, X., Kleinman, Y.,

Chen, Y., Hu, Y., Jia, Y., Qi, Y., Li, Y., Zhang, Y., Zhang, Y., Adi, Y., Nam, Y., Yu, Wang, Zhao, Y., Hao, Y., Qian, Y., Li, Y., He, Y., Rait, Z., DeVito, Z., Rosnbrick, Z., Wen, Z., Yang, Z., Zhao, Z., Ma, Z.: The Llama 3 Herd of Models (2024). https://arxiv.org/abs/2407.21783

[22] Wu, X., Yao, W., Chen, J., Pan, X., Wang, X., Liu, N., Yu, D.: From language modeling to instruction following: Understanding the behavior shift in LLMs after instruction tuning. In: Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pp. 2341–2369 (2024). https://doi.org/10.18653/v1/2024.naacl-long.130 . https://aclanthology.org/2024.naacl-long.130/

[23] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Brew, J.: Huggingface's transformers: State-of-the-art natural language processing. CoRR **abs/1910.03771** (2019) 1910.03771

[24] Zhang, B., Sennrich, R.: Root mean square layer normalization. CoRR **abs/1910.07467** (2019) 1910.07467

[25] Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C., Krishnan, D.: Supervised Contrastive Learning (2021). https://arxiv.org/abs/2004.11362

[26] Pakdaman Naeini, M., Cooper, G., Hauskrecht, M.: Obtaining well calibrated probabilities using bayesian binning. Proceedings of the AAAI Conference on Artificial Intelligence **29**(1) (2015) https://doi.org/10.1609/aaai.v29i1.9602

[27] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: PyTorch: An Imperative Style, High-Performance Deep Learning Library (2019). https://arxiv.org/abs/1912.01703

[28] Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., Steinhardt, J.: Measuring Massive Multitask Language Understanding (2021). https://arxiv.org/abs/2009.03300

[29] Pal, A., Umapathi, L.K., Sankarasubbu, M.: MedMCQA : A Large-scale Multi-Subject Multi-Choice Dataset for Medical domain Question Answering (2022). https://arxiv.org/abs/2203.14371

[30] Rafailov, R., Sharma, A., Mitchell, E., Ermon, S., Manning, C.D., Finn, C.: Direct preference optimization: Your language model is secretly a reward model. arXiv preprint arXiv:2305.18290 (2023) https://doi.org/10.48550/arXiv.2305.18290

[31] Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W.: Lora: Low-rank adaptation of large language models.

In: International Conference on Learning Representations (ICLR) (2022). https://openreview.net/forum?id=nZeVKeeFYf9

[32] Guo, C., Pleiss, G., Sun, Y., Weinberger, K.Q.: On calibration of modern neural networks. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 70, pp. 1321–1330. PMLR, Sydney, Australia (2017). https://proceedings.mlr.press/v70/guo17a.html

[33] Zadrozny, B., Elkan, C.: Transforming classifier scores into accurate multiclass probability estimates. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 694–699. ACM, New York, NY, USA (2002). https://doi.org/10.1145/775047.775151