École Normale Supérieure Paris-Saclay

Master MVA (Mathematiques, Vision, Apprentissage)

IRT Saint Exupéry

Post-Hoc Out-of-Distribution Detection for Hallucination Detection in Large Language Models

Internship Report — April to September 2025

Author:

Lila Roig

Academic referent:

Pierre LATOUCHE

Host institution:

IRT Saint Exupéry
3 Rue Tarfaya, 31400 Toulouse

Host supervisors:

Yannick PRUDENT
Paul NOVELLO

Academic Year 2024–2025

Acknowledgments

I would like to express my sincere gratitude to all those who contributed to the success of this research internship at IRT Saint Exupéry, conducted within the DEEL project as part of the MVA Master's program at ENS (Ecole Normale Supérieure) Paris-Saclay.

I am especially thankful to my internship supervisor, Yannick Prudent (AI Research Engineer at IRT Saint Exupéry), for giving me the opportunity to undertake this internship and for providing highly satisfactory guidance throughout the project. I am grateful to Yannick for his kindness, support, attentive listening, and trust throughout this work.

I also thank Paul Novello (Machine Learning Researcher at IRT Saint Exupéry) for co-supervising this internship and for his valuable assistance. My thanks also go to Grégory Flandin (Director of Artificial Intelligence for Critical Systems at IRT Saint Exupéry and Head of Certifiable AI at ANITI) for making this internship possible.

I am grateful to the entire IRT team for their warm welcome, good spirits, and day-to-day support. Their efforts helped me feel part of the team from the very beginning and contributed greatly to the project's success. I appreciated the excellent working conditions and resources, laptop, dual monitors, and GPU access, that enabled effective experimentation. The topic itself was both intellectually stimulating and highly engaging, and I am very satisfied with the overall experience.

Finally, I thank Pierre Latouche (Professor of Applied Mathematics, part of the LMBP laboratory of UCA and of the DepMAP at École Polytechnique), MVA referent for this internship, for overseeing the academic follow-up and reviewing this report.

Contents

Post-Hoc Out-of-Distribution Detection for Hallucination Detection in Large Language Models

Lila Roig
ENS Paris-Saclay — Master MVA | IRT Saint Exupéry

August 28, 2025

Abstract

Hallucinations in large language models (LLMs), defined as fluent yet factually incorrect or unsupported outputs, remain a major concern for the safe and certifiable deployment of AI in critical domains. We frame hallucination detection as an out-of-distribution (OOD) detection problem, that is, identifying inputs that deviate from the training distribution. First, we conducted a comprehensive survey of the state-of-the-art methods in OOD and hallucination detection. Then, we leverage post-hoc distance-based OOD detectors on internal model representations, as well as logit-based uncertainty metrics as a baseline. Specifically, we extract hidden states, attention maps, and logit-based uncertainty signals from LLaMA-2-7B Chat when evaluated on SQuAD 2.0, distinguishing answerable (in-distribution) from unanswerable (OOD, hallucination-prone) questions. We evaluate several descriptors with OOD scoring techniques (DeepKNN, Mahalanobis distance, cosine similarity) as well as linear probes, and systematically compare prompt-only versus response-based representations. Our results show that unsupervised OOD signals achieve modest discrimination (AUROC ≈ 0.6), with prompt-only embedding-based OOD detectors underperforming generation logit-based baselines. Supervised probes substantially improve linear separability (up to 0.80 accuracy on embedding-based descriptors) but at the cost of generalization. Layerwise trajectory analyses are explored and yield little additional benefit.

Keywords: hallucination detection; out-of-distribution (OOD) detection; large language models (LLMs); question answering; SQuAD 2.0; distance based methods; logit based uncertainty; Deep k-Nearest Neighbors (DeepKNN); Mahalanobis distance; cosine similarity; selective prediction and abstention; layer wise trajectories; hidden states and attention patterns.

1 Introduction

Large language models (LLMs) such as LLaMA-2 or GPT-style architectures represent a breakthrough in Natural Language Processing (NLP), enabling high-quality question answering, summarization, and reasoning. Yet, despite their impressive fluency, these systems often generate hallucinations: confident but incorrect or unsupported statements, i.e., content that is not entailed by the provided context. Hallucinations undermine reliability and trust, which is particularly critical in high-stakes domains such as aerospace, defense, or healthcare, where misleading outputs may lead to serious consequences. Detecting hallucinations has therefore become a central research challenge in efforts to make AI more dependable, explainable, and certifiable.

In parallel, the machine learning community has extensively studied out-of-distribution (OOD) detection, that is, determining whether a given input lies outside the scope of the training distribution.

OOD detection provides a well-established framework for uncertainty estimation and robustness, with substantial results and rigorous benchmarks in Computer Vision. Although comparatively less explored in NLP, recent benchmarks have highlighted the strong performance of post-hoc, distance-based methods for language tasks. Given the conceptual connection between OOD conditions and hallucinations (since hallucinations often arise when prompts fall outside the model's effective knowledge scope) this project asks a natural question: *Can hallucination detection in LLMs be reframed as an OOD detection task?*

First, we conduct an exhaustive survey of state-of-the-art methods for OOD and hallucination detection and present the findings in a concise, structured manner, highlighting the current lack of standardized benchmarks and comprehensive overviews, especially in hallucination detection. Then, we explore post-hoc distance-based OOD detection on internal representations of LLaMA-2-7B Chat, with logit-based uncertainty baselines, framing hallucination as the model's tendency to produce answers to unanswerable questions in SQuAD 2.0. By extracting and analyzing internal representations (hidden states, attention patterns, and logits) under both prompt-only and generation-based configurations, we evaluate how distance metrics (DeepKNN, Mahalanobis, cosine similarity), anomaly detection algorithms, and supervised linear probes can signal hallucination risk. Because these detection techniques do not require model retraining, they can anticipate hallucinations before or during generation, based solely on internal signals. Furthermore, we investigate whether trajectory analyses across layers provide additional discriminative power.

Our main contributions are threefold:

1. **Conceptual framing**: we propose and systematically assess a novel perspective that frames hallucination detection as an OOD problem, linking two research fields that have remained largely separate.

2. **Empirical evaluation of descriptors**: we perform a fine-grained comparative analysis of internal descriptors (hidden states, attention maps, logit scores), scoring strategies (distance-based, anomaly detection, probing), input configurations (prompt-only vs. generated answers) and layer-wise trajectories.

3. **Critical findings and limitations**: we show that **(i)** unsupervised OOD detection methods offer modest gains (AUROC ≈ 0.6), with prompt-only embeddings-based OOD methods performing worse than logit-based counterparts on generated text, **(ii)** supervised probes improve accuracy (up to ≈ 0.80) but are highly dataset-dependent, **(iii)** trajectory analyses across layers bring little additional benefit. These observations underline the intrinsic difficulty of hallucination detection: because hallucinations can differ from truthful outputs only in subtle, token-level ways, answerable and unanswerable cases may share nearly identical prompts and contexts, resulting in closely overlapping representations within embedding space.

The remainder of this report is organized as follows. Section 2 introduces the institutional and research context of the DEEL project under which this internship was conducted. Sections 3 and 4 provide concise, targeted surveys of existing literature: Section 3 outlines core concepts and representative methods in out-of-distribution (OOD) detection for computer vision and natural language processing, while Section 4 reviews hallucination phenomena, detection methods, and challenges in large language models. Section 5 motivates and frames hallucinations as an OOD detection problem. Section 6 details our methodology including dataset preparation, representation extraction, OOD detection algorithms, implementation details and layer-wise trajectory analysis. Section 7 reports experimental results, comparing prompt and generation-based features, analyzing OOD scores, probing outcomes, and trajectory analyses. Finally, Section 8 concludes with a discussion of limitations and future research directions.

2 Presentation of IRT Saint Exupéry and the DEEL Project

The **Institut de Recherche Technologique (IRT) Antoine de Saint Exupéry** is one of the eight French IRTs created by the government to strengthen the link between academic research and industrial innovation. Located in Toulouse, the institute brings together around 350 collaborators, including engineers, researchers, and PhD students, and operates under a public–private partnership model. Its primary mission is to accelerate technology transfer: moving advanced research results, often still confined to the academic world, toward concrete applications in industry. By doing so, the IRT contributes both to French competitiveness and to the structuring of strategic industrial sectors such as aerospace, transportation, and energy.

The institute's research activities are organized into several strategic axes, which address challenges closely related to embedded systems and future mobility:

- Greener technologies, focusing on energy transition and environmental impact reduction;
- Intelligent technologies, which include artificial intelligence, advanced learning methods, and autonomous detection;
- Advanced manufacturing technologies, oriented toward innovative industrial processes;
- Methods and tools for the design of complex systems, supporting robust engineering approaches for large-scale projects.

Among these axes, the *Intelligent Technologies* program plays a central role, particularly in the domains of machine learning, connectivity, and artificial intelligence for critical systems. It is within this research line that the **DEEL (DEpendable and Explainable Learning)** project has been developed. The DEEL program is an international research initiative that brings together major academic and industrial partners. Alongside IRT Saint Exupéry, key collaborators include the **Artificial and Natural Intelligence Toulouse Institute (ANITI)**, the **Consortium de recherche et d'innovation en aérospatiale du Québec (CRIAQ)**, and the **Institut de valorisation des données (IVADO) in Montréal**. This consortium reflects the cross-border and interdisciplinary ambition of the project. Its primary objectives are to make AI systems more dependable, explainable, and certifiable, which is particularly critical in domains such as aerospace, defense, and autonomous transportation, where reliability and trust are essential prerequisites. The challenges addressed by DEEL span several dimensions:

- Certifiability, ensuring that AI systems comply with stringent industrial and regulatory standards;
- Explainability, opening the "black box" of modern neural networks to provide transparency on their decisions;
- Reliability and robustness, guaranteeing consistent performance even in non-ideal or unforeseen scenarios;
- Confidentiality and security, preserving the integrity of AI models and the data they rely on.

My internship project fits into this framework, as it focuses on **Hallucination detection in large language models (LLMs) through out-of-distribution (OOD) detection methods**. Hallucinations represent a critical limitation for the safe deployment of AI systems. This issue is especially sensitive in high-stakes domains such as aerospace or autonomous systems, where misleading outputs can have severe consequences. By framing hallucination detection as an OOD detection problem, my work contributes to strengthening the reliability, transparency, and eventual certifiability of language models, in alignment with the objectives of the DEEL program.

3 Out-of-Distribution (OOD) Detection

3.1 Definition and Motivation

Out-of-distribution (OOD) detection is the process of identifying whether an input to a machine learning model comes from a different domain or distribution than the data the model was trained on. It is essential for the reliable and safe deployment of machine learning systems in the real world, as it helps prevent models from making confident but incorrect predictions on unfamiliar data. Accurate OOD detection ensures that models can recognize when an input lies outside the scope of their knowledge, thus avoiding erroneous or unpredictable outputs.

3.2 OOD Setup

Following the formalism in [1], let $\mathcal{P}_{\text{id}}$ be a probability distribution over an input space $\mathcal{X}$. Given a realization $x \in \mathcal{X}$ of a random variable $\mathbf{x}$, the goal of OOD detection is to decide whether $x \sim \mathcal{P}_{\text{id}}$ (i.e. x is *in-distribution*, ID) or $x \not\sim \mathcal{P}_{\text{id}}$ (i.e. x is *out-of-distribution*, OOD). In practice, $\mathcal{P}_{\text{id}}$ is only accessible through a finite set of i.i.d. samples drawn from the ID distribution $\mathcal{D}_{\text{id}} = \{x_1, \ldots, x_n\}$.

The standard approach for OOD detection is to construct a *score function* $s : \mathcal{X} \to \mathbb{R}$ together with a threshold $\tau \in \mathbb{R}$, such that:

$$\begin{cases} x \text{ is declared OOD} & \text{if } s(x) > \tau, \\ x \text{ is declared ID} & \text{if } s(x) \le \tau. \end{cases} \tag{3.1}$$

Typically, $\mathcal{D}_{\text{id}}$ is split into a *fit* set $\mathcal{D}_{\text{id}}^{\text{fit}}$ (to construct s) and an *evaluation* or *test* set $\mathcal{D}_{\text{id}}^{\text{eval}}$ (to assess performance). We evaluate the quality of the score function s as follows.

First, assume there are p samples in the ID *test* dataset $\{x^{(1)}, \ldots, x^{(p)}\} \subset \mathcal{D}_{\text{id}}^{\text{eval}}$ and p additional samples in an OOD *test* dataset $\{\bar{x}^{(1)}, \ldots, \bar{x}^{(p)}\} \subset \mathcal{D}_{\text{ood}}^{\text{eval}}$ drawn from another distribution $\mathcal{P}_{\text{ood}} \neq \mathcal{P}_{\text{id}}$ (typically, another dataset). Then, we compute scores on ID and OOD test points:

$$S_{\text{id}} = \{s(x^{(1)}), \ldots, s(x^{(p)})\}, \qquad S_{\text{ood}} = \{s(\bar{x}^{(1)}), \ldots, s(\bar{x}^{(p)})\}.$$

We treat OOD as the positive class and adopt the convention that larger $s(x)$ indicates higher OOD-likeness. For a threshold τ, the true–positive rate (TPR) and false–positive rate (FPR) are computed empirically as

$$\text{TPR}(\tau) = \frac{1}{p} \sum_{i=1}^{p} \mathbf{1}\{\, s(\bar{x}^{(i)}) > \tau \,\}, \qquad \text{FPR}(\tau) = \frac{1}{p} \sum_{i=1}^{p} \mathbf{1}\{\, s(x^{(i)}) > \tau \,\}.$$

Sweeping τ over the unique values in $S_{\text{id}} \cup S_{\text{ood}}$ (sorted in decreasing order) traces the ROC curve with $\text{FPR}(\tau)$ on the x–axis and $\text{TPR}(\tau)$ on the y–axis. The *area under this curve* (AUROC) summarizes the discriminative power of s across thresholds. Once an operating point is fixed (e.g., Youden's $J(\tau) = \text{TPR}(\tau) - \text{FPR}(\tau)$ maximizer, or the threshold whose TPR is closest to 95% to report FPR@95), threshold–dependent metrics are computed analogously.

3.3 Related work

OOD detection has been widely studied across various domains, with significant attention given to Computer Vision (CV) and, to a lesser extent, Natural Language Processing (NLP). The following section provides an overview of OOD detection methods and their application in these two fields.

3.3.1 OOD Detection in Computer Vision

OOD detection has been extensively studied in the field of Computer Vision, with several comprehensive benchmarks and comparisons of detection methods available: [2, 3, 4].

OOD Detection Setup In a typical OOD setup, a model is first trained from scratch exclusively on in-distribution (ID) data to specialize its representations for the target domain. For large pre-trained models, it is common practice to fine-tune the model on ID data rather than train from scratch. During evaluation, the model is presented with new test samples and must determine whether each sample originates from the ID distribution or represents an OOD example.

Trends in Computer Vision A synthesis of recent studies and benchmarks in this area reveals several important trends. Notably, **no single OOD detection method consistently outperforms all others** across diverse datasets and scenarios, as highlighted by [3]. The relative effectiveness of different approaches also depends on the scale of the dataset: training-based OOD detection methods tend to perform best on smaller datasets, whereas **post-hoc** OOD detection methods are generally more effective in large-scale settings [3]. In fact, post-hoc techniques often outperform training-based methods overall, as demonstrated in [2, 4]. Additionally, **data augmentation** during training on in-distribution data has been shown to substantially enhance OOD detection performance [2, 3, 4].

3.3.2 OOD Detection in NLP

While OOD detection has been widely studied in Computer Vision, its application in NLP has been more limited. To our knowledge, while a survey provides a comprehensive overview of OOD detection methods in NLP and categorizes existing approaches [5], there are still no large-scale empirical studies that systematically compare the various proposed techniques on shared benchmarks.

Challenges in NLP OOD Detection OOD detection in **NLP presents its own unique challenges**, distinct from those encountered in computer vision. For example, NLP models must operate over discrete input spaces and often handle complex output structures, which fundamentally differentiates the task from its vision counterpart [5]. Some studies have attempted to adapt OOD detection methods originally developed for vision to NLP and have provided experimental results, but these efforts are often limited to **decoder-only** architectures and are restricted in terms of model diversity and the range of datasets evaluated [6, 7, 8, 9, 10, 11]. There are comparatively fewer studies focusing on **encoder-only** models [12, 13, 9, 14] and, similar to decoder-only research, these works generally evaluate a limited set of models and datasets. Notably, decoder-based large language models (LLMs) have been shown to significantly outperform encoder-based models for OOD detection [12].

Fine-Tuning and Pre-training strategies In NLP, it is generally not feasible to train large models from scratch exclusively on ID data; instead, for large pre-trained models, fine-tuning on ID data is the standard approach to adapt representations to the specific task. Rather than performing full fine-tuning, parameter-efficient transfer learning (PETL) methods, such as LoRA, have been demonstrated to be as effective as full fine-tuning, with LoRA in particular showing notable stability during learning [13]. The impact of fine-tuning on OOD detection performance, however, remains a topic of debate within the community. Some studies report that fine-tuning on ID data significantly enhances OOD detection capabilities [15, 12, 16], while others find that pre-trained models without fine-tuning can outperform their fine-tuned counterparts, especially when there is a substantial distributional gap between ID and OOD data [17]. This discrepancy has led some researchers to propose alternative fine-tuning strategies, such as generative fine-tuning, to address the limitations observed with traditional discriminative fine-tuning approaches [12].

Notably, recent work has shown that pre-trained Transformers are substantially more robust to OOD examples than traditional, non-pretrained NLP models. In fact, non-pretrained models often perform at or below random chance when it comes to OOD detection, whereas pre-trained Transformers consistently demonstrate superior OOD detection capabilities [7]. These findings are further supported by results showing that pre-trained models achieve the best OOD detection performance overall, while the performance of other models frequently lags behind, sometimes even falling below chance [15]. This underscores the critical role of pre-training in enhancing OOD robustness in modern NLP systems.

Overview of OOD Detection Methods in NLP The previously presented papers generally categorize OOD detection methods into two main groups.

- **Logit-Based Methods**. The first group, logit-based methods, operates directly on the model's final output logits to derive OOD scores based on measures such as classifier confidence or energy. These approaches typically require additional training or calibration of the classification layer to improve OOD separability. Representative examples include Maximum Softmax Probability (MSP) [18], which is widely used as a baseline, and the Energy Score [19].
- **Distance-Based Methods** . The second group, distance-based methods, distinguishes ID from OOD samples by computing distances between the model's **hidden states (contextualized embeddings)**. The general procedure is as follows: contextualized embeddings are extracted from a chosen Transformer layer for all ID training samples and stored in a reference set (also called the fit dataset), which serves as an empirical representation of the known data distribution. At test time, the embedding of a new sample is computed and compared against this reference set using a distance metric. A larger distance suggests that the sample lies outside the training distribution, indicating a higher likelihood of hallucination. Several distance metrics are commonly employed:
    - **DeepKNN** [20] computes distances, typically **cosine similarity**, between the test embedding and all embeddings in the reference set. The k nearest neighbors are identified, and the distance to the k-th neighbor serves as the OOD score. If this score exceeds a predefined threshold, the sample is flagged as OOD; otherwise, it is considered ID. This non-parametric method makes no assumptions about the underlying feature distribution, offering flexibility and robustness across tasks.
    - **Mahalanobis Distance** [21] assumes that ID embeddings follow a multivariate Gaussian distribution in the feature space. The mean vector and covariance matrix of this distribution

are estimated from the ID reference set. At test time, the Mahalanobis distance measures how far a sample's embedding lies from this estimated ID distribution, taking into account correlations between features: $D_M(\mathbf{x}) = \sqrt{(\mathbf{x}-\mu)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\mu)}$ where $\boldsymbol{\mu}$ is the mean embedding of the ID set and Σ^{-1} is the inverse covariance matrix. Unlike Euclidean distance, which treats all features dimensions equally, Mahalanobis distance accounts for feature variance and correlations. This makes it more sensitive to subtle shifts, especially when embeddings have anisotropic (direction-dependent) variance.

- **Cosine Similarity** In this setting, cosine similarity measures the angle between the hidden state vector of a test sample $\mathbf{x}$ and those of ID reference samples $\mathbf{y}$. It is defined as: $\text{cos\_sim}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x}\cdot\mathbf{y}}{\|\mathbf{x}\|\,\|\mathbf{y}\|}$. A value close to 1 indicates strong alignment (likely ID), while values closer to 0 or negative suggest lower similarity (potentially OOD). In practice, for OOD detection, we compute the cosine similarity between $\mathbf{x}$ and all embeddings in the ID reference set, then use either the maximum similarity as the detection score. Lower scores indicate a higher likelihood of OOD. This "max-over-all" strategy assumes that high similarity to at least one training example is sufficient evidence for ID membership.

Performance and Methods Comparison The previously presented papers consistently show that **distance-based methods outperform other approaches, including logit-based methods**, for OOD detection in NLP tasks. This superiority has been demonstrated across several benchmarks and model types. For example, Mahalanobis distance achieves the best results in [8], while Cosine Similarity is identified as the top-performing method (followed closely by Mahalanobis) in [12]. Similarly, KNN-based approaches are shown to be most effective in both [6] and [2]. The paper [10] also reports that Mahalanobis distance, followed by cosine distance, yields the highest performance. This consistent advantage of distance-based methods could be attributed to the structure of fine-tuned Transformer representations, which tend to be highly homogeneous for ID data. As a result, distance metrics can more easily identify outliers, leading to significantly larger margins over logit-based methods [21].

Internal Representation Extraction Strategies: Layer and Token Selection While many OOD detection methods in language models focus on representations from the final layer, often using the last token or `[CLS]` embedding, recent research indicates that this approach can be suboptimal [22]. Studies such as [11] and [14] suggest that aggregating token representations across all transformer layers yields more holistic and robust sentence embeddings. The optimal layer for OOD detection may also depend on the specific OOD pattern, as shown in [23]. Analyses of transformer internals, such as in [24, 25], reveal that early layers capture lexical and syntactic information, middle layers organize tokens by topic, and later layers refine this information to isolate specific answers. Iterative inference in transformers means that middle layers are often critical for prediction refinement, especially depending on where relevant information appears in the input [26].

Limitations of OOD detection methods in NLP However, it is important to note that it has also been shown that no single OOD detection method consistently performs well across all settings. Several studies emphasize the need for an explicit definition of OOD examples when evaluating and comparing different detection approaches, as OOD performance can vary significantly depending on the specific task and data distribution [9, 27, 16]. Furthermore, no single backbone architecture has been found to consistently outperform others across all tasks, underscoring the complexity and variability of

OOD detection in NLP. These studies also suggest that evaluating on a large amount of test data can help mitigate performance fluctuations and provide a more reliable assessment of OOD robustness.

The challenge of identifying a single superior OOD detection method in NLP is further compounded by the lack of standardized benchmark suites specifically designed for the field. As highlighted by [16], there is currently no uniform set of datasets for OOD evaluation in NLP, which introduces several key issues. First, many OOD datasets used in previous work originate from distributions that are not sufficiently distinct from the ID data, producing weak (and sometimes unrealistic) distributional shifts. This lack of clear separation reduces OOD evaluation to an ID-like setting, undermining the rigor of robustness assessment. Second, the construction of these benchmarks often does not adhere to rigorous protocols, with dataset selection frequently driven by popularity rather than by principled criteria, as observed in suites like GLUE-X [16]. An additional complication arises with LLMs, which are typically pre-trained on vast public corpora; this raises the risk of data contamination, making it difficult to ensure that evaluation data has not already been seen during pre-training. To address this, [16] recommends the use of temporal datasets (data released after the LLM's training period) to enable more accurate OOD assessment. This concern is further illustrated by [28], which notes that the OOD robustness of ChatGPT cannot be reliably evaluated using GLUE or GLUE-X benchmarks, as these datasets may have been included in the model's training data. Collectively, these factors underscore the need for more rigorous and standardized benchmark construction in order to obtain meaningful and trustworthy evaluations of OOD detection methods in NLP.

4 Hallucination Detection

4.1 Definition

Hallucinations in LLMs refer to outputs generated by the model that are factually incorrect, contextually inconsistent, or deviate from the user's instructions. Unlike OOD issues, which involve inputs outside the model's training data, hallucinations focus on erroneous or fabricated outputs.

4.2 Related work

Despite increasing interest in hallucination detection, the field still **lacks a standardized benchmark**. Each study tends to define its own evaluation setting, often relying on different datasets and models. As a result, comparisons across papers remain difficult, and no systematic survey has yet established a unified set of baselines. To better situate our work, we begin by outlining the types and causes of hallucinations before presenting the main detection methods proposed in the literature.

4.2.1 Types of Hallucinations

LLMs exhibit two primary types of hallucinations [29]: 1) **Factual Hallucination**: A mismatch between the generated content and verifiable real-world facts 2) **Faithfulness Hallucination**: Occurs when LLM outputs diverge from user instructions, context, or logical consistency.

4.2.2 Causes of Hallucinations

Hallucinations are mainly factual and often arise from limitations in training data, especially when models face information outside their knowledge scope. Despite being trained on massive corpora, LLMs often fail to recognize their own knowledge boundaries, leading to hallucinations [29].

4.2.3 Methods for Hallucination Detection

Against this background, a diverse set of methods has been proposed to detect hallucinations and can be grouped into categories such as logit-based uncertainty metrics, consistency checks, probing classifiers, attention and internal-state analyses, self-evaluation, gradient-based techniques, and retrieval-augmented strategies.

Logit-Based and Uncertainty Methods

- **Perplexity and Predictive Entropy (PE)**. **Perplexity** measures how well a language model predicts a text sequence by computing the exponentiated average of the token-wise negative log probabilities assigned by the model to the actual tokens in a generated sequence. **Predictive Entropy**, on the other hand, averages the token-wise entropies over a generated sequence (where the entropy at each step measures the uncertainty in the model's predicted probability distribution for the next token). [30] shows that predictive entropy is a useful signal for estimating an LLM's uncertainty and factual accuracy.

- **Semantic Entropy (SE)** [31] measures uncertainty based on meaning, clustering semantically equivalent responses, and provides a more accurate predictor of model accuracy than standard predictive entropy, which can inflate uncertainty due to different phrasings.

- **SAR** (*Shifting Attention to Relevance*) [32]. While **PE** treats all tokens equally which can distort uncertainty estimates, **SAR** improves uncertainty quantification by focusing predictive entropy on the most semantically relevant tokens and sentences, leading to more accurate confidence assessments.

- **EUBHD** (*Enhancing Uncertainty-Based Hallucination Detection*) [33] proposes a reference-free, uncertainty-based method that prioritizes factual elements (entities, dates, etc.), using token-level entropy to flag likely hallucinations and propagating uncertainty from unreliable tokens to later ones to capture cascading errors. It then adjusts probabilities based on token properties such as entity type and frequency.

Consistency-Based Methods

- **SelfCheckGPT** [34] is a zero-resource, black-box method that detects hallucinations by sampling multiple responses to the same prompt and measuring the factual consistency across outputs. The core intuition is that if a model is knowledgeable, its responses will be similar; if it hallucinates, responses will diverge or contradict each other.

- **INSIDE** (*INternal States for hallucInation DEtection*) [35] analyzes the internal states of LLMs by measuring the semantic consistency of multiple responses using the EigenScore, which captures differential entropy in the embedding space. This method outperforms surface-level semantic or logit-based uncertainty metrics, and feature clipping at test time further reduces overconfident, hallucinated outputs.

- **Knowing What LLMs Do Not Know** [36] This method combines (1) consistency checks across paraphrased questions (using entropy scores), and (2) a verbalization-based detection that assesses how typical the question phrasing is (via negative log-likelihood). Together, these components help identify when the LLM is uncertain or likely to hallucinate.

- **Limitations**. Consistency-based methods are often computationally expensive, as they require generating multiple responses or large datasets. They also assume hallucinations are rare and that most outputs are accurate, but LLMs can repeat similar errors, leading to false positives or undetected hallucinations [37].

Probing and Classifier-Based Approaches Several studies suggest that even when a LLM generates false information, it may still "internally know" that the information is incorrect, and this can be detected from its hidden activations. These methods indicate that truthful answers occupy a specific geometric direction or subspace within these activations, which can be identified using simple techniques such as linear classifiers.

- **SAPLMA** (*Statement Accuracy Prediction, based on Language Model Activations*) [38] Demonstrates that a simple neural classifier trained on LLM hidden states can distinguish between true and false statements, indicating that LLMs may "internally know" when they are generating false information, even if this is not reflected in their outputs or decoding preferences.
- **CCS** (*Contrast-Consistent Search*) [39] Shows that training an unsupervised linear probe can extract a latent "truth" direction from internal model activation states, separating true from false answers to yes/no questions, showing that models encode a robust, task-agnostic representation of truth even when their outputs are incorrect.
- **Inference-Time Intervention** [40] Similar to CCS, trains a supervised linear probe to identify directions in attention head activations that separate truthful from false answers, allowing to shift model activations these "truth directions" at inference time.
- **The Curious Case of Hallucinatory (Un)answerability** [41] Shows that training a simple linear classifier on the hidden state of the first generated token can predict answerability and that removing this subspace from the model's activations causally disrupts the model's ability to recognize (un)answerability, proving that this information is not just encoded, but behaviorally relevant for LLMs.
- **Limitations**. The paper [42] critically evaluates probe-based methods like SAPLMA and CCS, finding that they often rely on superficial correlations in training data rather than a genuine "truth" signal. These methods show poor generalization to OOD data or different topics and depend heavily on topic-specific training data, limiting their practical use. Conceptually, the authors challenge the assumption that LLMs encode truth in a way that simple probes can extract, since supervised probes are trained to predict labels based on the training set, not to uncover a model-internal truth. Similarly the paper [43] shows that the "geometry of truth" in LLMs is highly specific to each task and does not generalize across tasks. Linear classifiers trained to distinguish correct from incorrect answers show almost no overlap between tasks, and even advanced methods fail to find a universal truth representation. As a result, each task has its own distinct truth geometry, meaning that automatic truth detection in LLMs' latent spaces is inherently task-dependent rather than universal.

Internal Representation and Attention-Based Methods

- **LLM-Check** [37] Analyzes a single LLM response by examining internal attention maps, hidden states, and output prediction probabilities. Attention scores derived from self-attention matrices outperform other metrics (such as Hidden Score, Perplexity, and Windowed Logit Entropy) in detection performance and outperform traditional consistency-based methods in performance and speed.

- **MIND** (*Modeling of INternal states for hallucination Detection*) [44] Enables real-time hallucination detection by analyzing contextualized token embeddings during generation, after training an Hallucination Classifier.
- **Lookback Lens** [45] Proposes a lightweight method using attention patterns to detect and mitigate hallucinations. The approach introduces a lookback ratio (calculated as the ratio of attention weights allocated to the input context versus newly generated tokens across transformer layers and heads) which serves as input to a linear classifier that detects contextual hallucinations.

LLM Self-Evaluation and External Verification

- **Zero-Resource Hallucination Prevention** [46] Prompts the LLM to self-assess its familiarity with concepts before generating a response, withholding answers on unfamiliar topics.
- **A Stitch in Time Saves Nine** [47] Detects hallucinations by (1) identifying low-confidence concepts in generated text (using the minimum token probability as a signal), (2) prompting the model to create a validation question about those concepts, and (3) verifying the answer using web search.
- **Limitations** : As LLMs are not inherently reliable factual databases, relying solely on their parametric knowledge for fact-checking can lead to inaccurate assessments [29].

Gradient-Based Methods

- **EGH** (*Embedding and Gradient based Hallucination detection method*) [48] detects hallucinations by leveraging embeddings and gradient information from a language model's internal states. It compares output distributions with and without input conditioning and trains a simple classifier on these features.

Retrieval-Augmented Generation (RAG) Methods Retrieval-based methods such as **RAGTruth** [49] and **KnowHalu** [50] require clean reference data and are not explored in this work, as they depend on external databases and can themselves be subject to hallucinations. Even with reference information, generated responses may still include hallucinations.

4.2.4 Internal Representation Extraction Strategies

A key challenge in hallucination detection is identifying where and how to extract internal signals from LLMs that are most predictive of hallucinated content. Recent literature explores a variety of strategies based on layer depth, token selection, and decoding setup. This section summarizes the most commonly used practices.

Layer and Token Selection Recent hallucination detection methods for large language models demonstrate a wide range of strategies for leveraging internal representations, particularly regarding which layer and which token(s) to extract embeddings from.

Several approaches, including [38, 39, 40] rely on the last token embedding, which incorporates information from all preceding tokens and reflects the model's final contextual understanding before output generation. Some methods, such as [51], further average token embeddings from the last hidden layer to represent entire sentences. In contrast, [41] employs the first token embedding to capture the model's initial interpretation of the input. Recent studies, including [38, 39, 40, 37] highlight that middle-layer embeddings, rather than just the final layer, can provide even stronger signals for hallucination detection. Additionally, methods like [33, 37] compute token-level hallucination scores and aggregate them to assess hallucination at the sentence level. Collectively, these varied strategies reflect ongoing efforts to pinpoint which internal features most effectively capture hallucination phenomena in LLMs.

Decoding Strategy: Autoregressive vs. Teacher Forcing Another key methodological choice concerns how to extract representations: either with autoregressive generation or via teacher forcing. Each approach offers distinct benefits and limitations, and both are widely used in hallucination detection benchmarks

- **Autoregressive generation Extraction** has been widely adopted in prior work on hallucination detection, including [51, 41, 14, 32]. In this setup, we use **Hallucination Evaluation Benchmarks** composed of prompts that may trigger hallucinated responses. The goal is to analyze how hallucinations arise naturally during generation, based on the model's own outputs. To this end, we feed the prompt into the language model and allow it to generate a response. During autoregressive generation, the model produces tokens one by one, each time computing hidden states for all tokens generated so far. Each token's embedding is conditioned only on the tokens to its left, enforced by the causal attention mask. The last hidden state is then used to predict the next token in the sequence.

- **Teacher Forcing Extraction** has also been used in several studies on hallucination detection, including [39, 37, 8, 40, 33]. In this setup, we use **Hallucination Detection Benchmarks** consisting of prompts paired with responses that may or may not contain hallucinations (typically generated by a large language model). The objective is to determine whether the given answer is factually grounded. To do so, we concatenate the prompt and the ground-truth answer (factual or hallucinated) and feed the full sequence into the model in a single forward pass, with the prompt serving as context. Unlike standard autoregressive generation, where the model predicts tokens one token at a time based on its own previously generated outputs, teacher forcing involves processing the response token-by-token, but with the model receiving the actual ground-truth tokens at each step instead of its own predictions. This enables the model to compute hidden states, attention maps, and logits as if it had generated the answer itself. Although the sequence is processed in parallel during the forward pass, the transformer's causal masking mechanism ensures that each token's representation is conditioned only on the tokens to its left, effectively replicating the left-to-right context available during autoregressive decoding.

  While teacher forcing embeddings are not strictly identical to those obtained from autoregressive generation, they are generally comparable when computed over the same generated text. Based on our observations, differences tend to be more pronounced in the earlier layers of the model and diminish in the later layers. These differences may arise from factors such as caching mechanisms and positional embeddings (RoPE), which are updated incrementally during token-by-token generation but computed all at once during a full forward pass.

5 Framing Hallucinations as an OOD Detection Problem

We believe that there is a strong conceptual link between OOD detection and hallucination detection: hallucinations are more likely to occur when LLMs process OOD inputs that differ from their training data [29], suggesting that advances in OOD detection could inform or improve hallucination detection. While OOD detection is well-studied, its application to hallucination detection remains largely unexplored, with no research, to our knowledge, investigating the use of OOD methods for identifying hallucinations.

That said, framing hallucination detection as an OOD problem is intrinsically **challenging**: classical OOD is essentially novelty detection, where OOD samples lie relatively far from the ID manifold and thus admit large geometric margins. By contrast, hallucinations often differ from truthful outputs only in subtle, token-level ways, answerable and unanswerable cases may share nearly identical prompts and context, so their embeddings remain close, compressing the margin and making distance-based OOD scoring markedly harder.

6 Methods

6.1 Objective

Our objective is to apply **post-hoc, distance-based OOD detection methods to predict whether a given input prompt is likely to trigger a hallucinated response: in essence, using OOD detection to identify hallucinations**.

We focus on **post-hoc OOD methods** because they are adaptable to any model, require no retraining, and have shown robust performance across architectures [2, 4]. Furthermore, we adopt **distance-based OOD methods**, which compute distances between the model's internal representations and those from a known in-distribution reference set. As reviewed in Section 3.3.2 these methods consistently outperform logit-based approaches for OOD detection in NLP tasks. Finally, to establish a baseline for hallucination detection, we will also collect the LLM's output prediction probabilities from which we compute common **logit-based** uncertainty metrics.

6.2 Model and Dataset

6.2.1 Model

We use **LLaMA 2-7B Chat** [52], a large decoder-only language model fine-tuned for conversational question answering. The model is treated as a frozen black-box decoder and we do not modify its weights. Indeed, unlike many OOD setups, **we do not fine-tune the model on ID data**. This decision is driven by both practical and methodological considerations. Fine-tuning is resource-intensive, can introduce instability, and is not typically required in hallucination detection protocols. Moreover, as discussed in Section 3.3.2, the impact of fine-tuning on OOD performance remains unclear.

6.2.2 Data and Labeling

For the dataset, we use the Stanford Question Answering Dataset SQuAD 2.0 [53], which extends SQuAD 1.1 [54] by introducing approximately 50,000 unanswerable questions. SQuAD 1.1 contains over 100,000 question–context–answer triples from Wikipedia articles. Each question is paired with a context passage, and the answer is a span from the passage. In contrast, the additional questions in SQuAD 2.0 are explicitly designed to be unanswerable given the provided context, while still closely resembling answerable ones. These questions increase the likelihood of hallucinations as the model may attempt to generate plausible answers despite the absence of sufficient information.

This makes the dataset particularly well-suited for evaluating a model's ability to recognize when a question cannot be answered based solely on the provided context. By requiring the model to answer only when the necessary information is present in the text, the evaluation setup **mimics a realistic OOD scenario**: the model must detect when the input lacks sufficient evidence, indicating that the query falls outside its supported knowledge scope. In such cases, a robust model should identify the absence of relevant information and refrain from hallucinating an answer.

Since we use OOD detection methods to detect hallucinations, it is essential to define both in-distribution (ID) and out-of-distribution (OOD) data.

- **ID Fit Dataset** $\mathcal{D}_{\text{id}}^{\text{fit}}$. We use the training set of SQuAD 2.0 to construct the ID dataset. Each question–context pair is passed to the model, which generates an answer. If the generated answer matches the ground-truth answer, based on similarity metrics such as Sentence-BERT and ROUGE-L, the input is labeled as ID; otherwise, it is discarded. This approach ensures that the dataset used for training is well understood by the model and that the model can correctly answer all ID questions.

- **Test (or Evaluation) Dataset**. To evaluate hallucination detection, we use the validation set of SQuAD 2.0, ensuring there is no overlap with the ID fit data. The test set includes:

  - **Impossible samples** $\mathcal{D}_{\text{ood}}^{\text{eval}}$: Unanswerable questions, for which the answer is not present in the context passage. These inputs are likely to trigger hallucinations and are regarded as *out-of-distribution* (OOD) test inputs. An exampe of an OOD unanswerable test sample in SQuAD 2.0 is:

```
{'id': '5a67aa48f038b7001ab0c401',
 'title': 'Private_school',
 'context': 'Some of the oldest schools in South Africa are private church schools
     that were established by missionaries in the early nineteenth century. The
     private sector has grown ever since. After the abolition of apartheid, the laws
     governing private education in South Africa changed significantly. The South
     African Schools Act of 1996 recognises two categories of schools: "public" (
     state-controlled) and "independent" (which includes traditional private schools
     and schools which are privately governed.)',
 'question': 'In what century was South Africa established as a country?',
 'answers': {}
}
```

  - **Possible samples** $\mathcal{D}_{\text{id}}^{\text{eval}}$: Answerable questions, where the answer exists as a span within the provided context. These inputs should not induce hallucinations and are treated as *in-distribution* (ID) test inputs. An example of an ID answerable test sample in SQuAD 2.0 is:

```
{'id': '5705e63175f01819005e7724',
 'title': 'Southern_California',
 'context': "Within southern California are two major cities, Los Angeles and San
     Diego, as well as three of the country's largest metropolitan areas. With a
     population of 3,792,621, Los Angeles is the most populous city in California and
      the second most populous in the United States. To the south and with a
     population of 1,307,402 is San Diego, the second most populous city in the state
      and the eighth most populous in the nation.",
 'question': 'In which cardinal direction from Los Angeles is San Diego?',
 'answers': {'answer_start': 280, 'text': 'south'}
}
```

6.2.3 Prompt Construction

Since we use Llama 2-7B Chat, prompts must follow the model's required structure, including the `[INST]`, `[/INST]`, and `«SYS»`, `«/SYS»` tokens as specified in the official documentation [1]. To ensure answers are concise and comparable, since long responses can distort similarity metrics, we first experimented with few-shot prompting to encourage brevity. However, we found that prepending a specific instruction to each prompt was both more computationally efficient and more effective. This approach aligns with findings from [41], which showed that systematically including explicit guidance about (un)answerability in the prompt improved performance (up to 80%) and outperformed few-shot prompting. The final prompt format is as follows:

```
[INST] <<SYS>>\n
Given the following passage and question, answer the question by only giving the answer
without a complete sentence.\n
If it cannot be answered based on the passage, reply 'unanswerable':\n
<</SYS>>\n
\n
Passage: {Context}\n
Question: {Question} [/INST]
```

6.3 Internal Representation/Descriptor Extraction Process

To apply our OOD detection methods for hallucination detection, we first need to extract **meaningful internal representations or descriptors** from the LLM that may reveal whether it is hallucinating. These extracted representations will serve as input to the OOD detection techniques described in Section 6.4.

6.3.1 Extract Hidden States (Contextualized Embedding)

As outlined in Section 4.2.3, studies have found that even when an LLM generates incorrect information, its hidden activations may still reflect internal awareness of the error. Similarly, as discussed in Section 3.3.2, distance-based OOD detection methods rely on computing distances between hidden states (i.e., contextualized token embeddings) to distinguish ID from OOD inputs. Together, these findings suggest that **hidden states** encode valuable signals for detecting hallucinations, motivating our decision to use them as inputs for OOD detection. Furthermore, as reviewed in Sections 3.3.2 and 4.2.4, OOD and Hallucination detection methods vary in both the **layer** and **token(s)** used for embedding extraction. Building on these findings, we explore various strategies to aggregate hidden states.

Let a sequence $\mathbf{x}$ consist of m tokens (this can be the prompt only, the generated answer only, or their concatenation), and let the token embedding dimension be d. The corresponding hidden representation matrix is $\mathbf{H} \in \mathbb{R}^{m \times d}$.

Basic aggregation methods (Embeddings) For each sample in both the ID fit dataset and the test dataset, we extract hidden states using different aggregation methods: the hidden state of the **last token**, the hidden state of the **first token**, the **average** hidden state across all tokens, and the **maximum** hidden state computed feature-wise across all tokens, i.e., for each feature dimension, we select the maximum activation value over the sequence. Each resulting hidden state aggregation has the shape d. We also compute the **feature-wise variances** (shape d), corresponding to the diagonal of the feature covariance matrix $\mathbf{C}_{feat}$.

[1] https://huggingface.co/meta-llama/Llama-2-7b-chat-hf

$$\text{feature-wise variances} = \text{diag}(\mathbf{C}_{feat}) \in \mathbb{R} \tag{6.1}$$

$$\mathbf{C}_{feat} = \frac{1}{m}\mathbf{H}^\top \mathbf{Z}\mathbf{H} \in \mathbb{R}^{d\times d} \quad \text{where} \quad \mathbf{Z} = I_m - \frac{1}{m}\mathbf{1}_m\mathbf{1}_m^\top \in \mathbb{R}^{m\times m} \tag{6.2}$$

Here, $Z \in \mathbb{R}^{m\times m}$ is the token-wise centering matrix.

Higher-order statistics (Hidden Score) To capture structural properties of hidden representations beyond simple token aggregation, we draw inspiration from LLM-Check [37]. Specifically, we compute what the authors call the **Hidden Score**. This score is defined as the mean log singular value of the centered token Gram matrix across tokens, and is designed to reflect the internal diversity of hidden representations. To compute the centered token Gram matrix $\mathbf{G}_{tok}$, we use:

$$\mathbf{G}_{tok} = \mathbf{H}\mathbf{J}\mathbf{H}^\top \in \mathbb{R}^{m\times m} \tag{6.3}$$

Where $J \in \mathbb{R}^{d\times d}$ is the feature centering matrix defined as:

$$\mathbf{J} = I_d - \frac{1}{d}\mathbf{1}_d\mathbf{1}_d^\top \in \mathbb{R}^{d\times d} \tag{6.4}$$

with I_d, the $d \times d$ identity matrix and $\mathbf{1}_d$ a d dimensional vector of ones. The singular values σ_i of $\mathbf{H}$ are used to compute the the log-determinant of $\mathbf{G}_{tok}$:

$$\log\det\left(\mathbf{G}_{tok}\right) = \log\prod_{i=1}^{m}\sigma_i^2 = \sum_{i=1}^{m}\log\sigma_i^2 = 2\sum_{i=1}^{m}\log\sigma_i \tag{6.5}$$

To remove the explicit dependence on input-length, the Hidden Score is defined as the mean log singular value:

$$\text{Hidden Score} = \frac{2}{m}\sum_{i=1}^{m}\log\sigma_i \in \mathbb{R} \tag{6.6}$$

The centered Gram matrix $\mathbf{G}_{tok}$ measures similarity between tokens after removing the average activation across each feature. This centering removes any common bias shared by all tokens, focusing instead on the relative differences between them. The eigenvalues and singular values of $\mathbf{G}_{tok}$ capture the structural diversity of token embeddings within a sequence. Higher Hidden Scores indicates more diverse and complex embeddings, which can signal hallucinations, while lower scores reflect more redundant or aligned tokens, associated with factual or grounded responses. Note that the token covariance matrix and the centered token Gram matrix are related via $\mathbf{C}_{tok} = \frac{1}{d}\mathbf{G}_{tok}$.

6.3.2 Extract Attention Maps

In addition to hidden states, we also extract attention maps, following the approach introduced in LLM-Check [37]. This work showed that attention patterns can carry complementary signals of factuality, potentially enriching our OOD-based hallucination detection. To this end, we compute the **Attention Score**.

In a transformer architecture employing a self-attention heads per layer, the attention maps for an input sequence $\mathbf{x}$ of length m are represented as tensors of shape $(a \times m \times m)$. Each attention head produces a kernel similarity map defined by:

$$\mathrm{Ker}_i = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right) \in \mathbb{R}^{m\times m} \tag{6.7}$$

where Ker_i denotes the attention matrix for head $i \in \{1, \ldots, a\}$. These matrices are lower-triangular due to the autoregressive masking used in causal attention, and therefore the eigenvalues of Ker_i are just the values on the principal matrix diagonal. Moreover, since the softmax operation produces non-negative values, the eigenvalues of Ker_i are expected to be non-negative. This structure allows for a computationally efficient estimate of the matrix log-determinant without requiring singular value decomposition:

$$\log\det(\mathrm{Ker}_i) = \sum_{j=1}^{m} \log \mathrm{Ker}_i^{jj} \tag{6.8}$$

where Ker_i^{jj} represents the (j, j) diagonal entry of the square matrix Ker_i. Again, we normalize for sequence length by computing the mean log-determinant over tokens for each attention head, and sum the attention heads them to obtain the Attention Score for each sample:

$$\text{Attention Score} = \sum_{i=1}^{a}\left(\frac{1}{m}\sum_{j=1}^{m} \log \mathrm{Ker}_i^{jj}\right) \in \mathbb{R} \tag{6.9}$$

6.3.3 Logit-Based Uncertainty Metrics

We also collect the LLM's output prediction probabilities to compute standard **logit-based uncertainty metrics**, including **Perplexity** and **Logit Entropy** (also referred to as Predictive Entropy). These metrics serve two key purposes in our study. First, they provide a baseline for hallucination detection, as logit-based measures are commonly used in the literature as standard reference points for evaluating hallucination-related phenomena. Second, although we argue in Section 3.3.2 that distance-based methods consistently outperform logit-based approaches in OOD detection, a few minority findings in the literature suggest otherwise. For instance, [13] and [51] report cases where logit-based or energy-based methods perform favorably. Specifically, the former study argues that energy-based methods outperform alternatives, likely because autoregressive language models lack a `[CLS]` embedding that summarizes the entire sequence. Instead, decoder-only models relies on the last token embedding as a global representation, which is optimized for next-token prediction but fails to capture the holistic semantics of a sentence as effectively as the `[CLS]` token in encoder-based models.

Perplexity Perplexity measures how well a language model predicts a sequence of tokens. It quantifies the model's uncertainty or "surprise" when generating the next token, effectively evaluating how likely the model considers the correct (ground-truth) tokens given the preceding context. Lower perplexity indicates higher confidence and better predictive performance, whereas higher perplexity reflects greater uncertainty. The standard formulation of perplexity assumes access to ground truth tokens and is given by:

$$\mathrm{PPL} = \exp\left(\frac{1}{m}\sum_{t=1}^{m} \log p(w_t^{real} \mid w_{<t}^{real})\right) \tag{6.10}$$

where m is the length of the sequence, w_t^{real} denotes the true token at time step t and $w_{<t}^{real}$ represents the preceding tokens up to position $t-1$. The term $p(\cdot \mid \cdot)$ refers to the model's predicted probability distribution over the vocabulary, computed from the model's raw output logits. At time step t, the

model produces a vector of logits $\mathbf{z}_t$:

$$\mathbf{z}_t = (z_{t,1}, z_{t,2}, \ldots, z_{t,V}) \in \mathbb{R}^V$$

where V is the vocabulary size. The probability of token $w_t = v$ (where $v \in \{1, \ldots, V\}$) is then computed using the softmax function:

$$p(w_t = v \mid w_{<t}) = \frac{\exp(z_{t,v})}{\sum_{j=1}^{V} \exp(z_{t,j})} = (\text{softmax}(\mathbf{z}_t))_v \tag{6.11}$$

Here $z_{t,v}$ is the logit (unnormalized score) assigned to token v at step t, and the softmax transforms it into a proper probability distribution over the vocabulary.

In our setup, however, we do not have access to ground truth sequences, as we operate in pure generation mode. As a result, we compute **Pseudo-perplexity** based on the model's self-generated text:

$$\tilde{\text{PPL}} = \exp\left(\frac{1}{m} \sum_{t=1}^{m} \log p(w_t^{gen} \mid w_{<t}^{gen})\right) \tag{6.12}$$

Here w_t^{gen} denotes the token generated by the model at position t and the sequence $w_{<t}^{gen}$ contains its preceding generated tokens. This variant measures the model's internal consistency that is, how probable the model finds its own generations under its own distribution. Since pseudo-perplexity relies on the model's own output rather than ground-truth tokens, it reflects model consistency rather than direct predictive accuracy.

Logit Entropy Entropy measures the uncertainty of the model's predicted probability distribution over the vocabulary at each token position. It quantifies how "spread out" or "confident" the model is about its predictions. Higher entropy indicates more uncertainty (more uniform probabilities), while lower entropy means the model is more confident (more peaked distribution). Given the logits vector at time step t:

$$\mathbf{z}_t = (z_{t,1}, z_{t,2}, \ldots, z_{t,V}) \in \mathbb{R}^V$$

where V is the vocabulary size, the predicted probability distribution is obtained by applying softmax, $v \in \{1, \ldots, V\}$:

$$p(w_t = v \mid w_{<t}) = \frac{\exp(z_{t,v})}{\sum_{j=1}^{V} \exp(z_{t,j})} = (\text{softmax}(\mathbf{z}_t))_v \tag{6.13}$$

The entropy at position t is then defined as:

$$\mathbf{E}_t = -\sum_{v=1}^{V} p(w_t = v \mid w_{<t}) \log p(w_t = v \mid w_{<t}) \tag{6.14}$$

To compute an overall average entropy over a sequence of length m, we average over all token positions:

$$\mathbf{E} = \frac{1}{m} \sum_{t=1}^{m} \mathbf{E}_t = -\frac{1}{m} \sum_{t=1}^{m} \sum_{v=1}^{V} p(w_t = v \mid w_{<t}) \cdot \log p(w_t = v \mid w_{<t}). \tag{6.15}$$

As suggested by the paper LLM-Check [37], the entropy can be computed by considering only the top-k tokens with the highest logits at each position, ignoring tokens with very low probability to reduce noise. For longer sequences, raw token-level entropy can be noisy and less informative, especially when hallucinations occur only in localized regions. Averaging entropy across the entire sequence may dilute these signals and reduce detection sensitivity. To address this, LLM Check [37] introduces **Windowed Logit Entropy**. This variant applies a sliding window of fixed width w over the sequence of per-token entropies, using a stride s (default $s = w$, yielding non-overlapping windows). Within each window, the mean entropy is computed, and the final Windowed Logit Entropy is defined as the maximum mean entropy observed across all windows for a given sequence.

6.3.4 Prompt vs. Response Representations

As previously discussed, we extract internal representations from LLMs (such as hidden states, attention maps, and logits) to apply OOD detection methods. To capture complementary linguistic, semantic, and structural information, we retrieve these representations from multiple transformer layers, following varied approaches reported in recent literature (Sections 3.3.2 and 4.2.4). These representations can be obtained from three distinct sources: **(1) the input prompt** alone, **(2) the generated answer** alone, or **(3) the concatenation of both** prompt and generated answer. Ideally, representations from the prompt alone (as in OOD detection setups) would suffice to detect hallucinations, as this would enable prediction before the response is generated, making the approach both more efficient and more practical for real-time applications. However, to potentially enrich the detection signal, we also extract internal representations from the generated answer (as in Hallucination detection setups). In our case, we extract the generated answer representations using the **autoregressive decoding strategy** (see Section 4.2.4), which reflects the natural generation behavior of LLMs without relying on teacher forcing.

We therefore evaluate all three configurations: prompt representations only, generated answer representations only, and concatenated prompt–answer representations, **across multiple transformer layers** to determine which segment(s) encode the most reliable indicators of hallucination risk.

It is worth noting that the first-token hidden state for the prompt configuration is excluded from our analysis. This is because the first generated token in the prompt is always the BOS special token (e.g., `<s>`), which is identical across all samples. Consequently, its embedding is effectively the same for every instance in the dataset. Under these conditions, distance-based OOD methods such yield degenerate scores (all distances collapse to zero) providing no discriminative power for hallucination detection.

6.4 OOD Detection Methods

6.4.1 Descriptor Types

At the end of the descriptor extraction process, we obtain **ten** distinct internal representations per sample:

- **Five embedding-based descriptors** of shape d (where d is the embedding dimension), derived from the model's contextualized token embeddings and defined in Section 6.3.1: **last-token** hidden state, **first-token** hidden state, **average** hidden state, **maximum** hidden state, and **feature-wise variance** of hidden states. These descriptors represent points in the model's embedding space, on which our distance-based OOD detection methods will be applied to produce an OOD score for each sample. The underlying hypothesis is that higher OOD scores correspond to a higher risk of hallucination.

- **Five scalar descriptors** of shape 1, which are inherently numerical scores rather than embeddings: the **Hidden Score** (Section 6.3.1), the **Attention Score** (Section 6.3.2), and three logit-based uncertainty metrics (Section 6.3.3): **Perplexity** , **Logit Entropy**, and **Windowed Logit Entropy**. Since these descriptors are already defined as single-value measures of uncertainty or atypicality, they are used directly as hallucination/OOD scores, following the same convention that higher scores correspond to a greater OOD/hallucination risk, without applying additional OOD methods.

This separation ensures that distance-based OOD detection is applied only where it is meaningful, on high-dimensional embedding representations, while scalar descriptors are incorporated as direct predictive signals in our hallucination detection framework.

6.4.2 Distance-Based OOD Detection

Our main OOD detection approaches are **DeepKNN**, a density-based method shown to perform well in OOD detection, along with **Cosine Similarity** and **Mahalanobis Distance** (Section 3.3.2). We also explore **anomaly detection** algorithms and supervised **linear probing** as complementary strategies. We further refine both methods with a **multi-center approach**, replacing the global mean (Mahalanobis) or the full set of individual embeddings (Cosine) with a smaller number of representative centers. These centers are obtained either via k-means (synthetic centroids) or k-medoids (actual ID examples that best represent each cluster). At test time, distances or similarities are computed only to these k centers, improving efficiency and better capturing multimodal ID distributions. For Mahalanobis distance, we also address the potential ill-conditioning of the covariance matrix (common when n is small relative to d) using the **Ledoit–Wolf shrinkage estimator** [55]. This forms a convex combination of the sample covariance and a scaled identity matrix, with the shrinkage parameter chosen automatically to minimize the expected squared Frobenius error.

6.4.3 Anomaly Detection algorithms

In addition to distance-based approaches, we evaluate classical anomaly detection methods applied to LLM embeddings. **One-Class SVM (OCSVM)** learns a boundary that encloses most ID points and flags points outside this boundary as anomalies. **Isolation Forest**, recursively partitions the feature space, flagging points that are isolated in fewer steps as more anomalous.

6.4.4 Linear Probing

Finally, while all previous methods are unsupervised, prior hallucination-detection studies (Section 4.2.3) have also employed **linear probes**. These works suggest that hallucinations may be linearly separable from truthful outputs in the representation space. Although such probes are sometimes criticized for being overly dataset-specific and lacking generalization, we nonetheless evaluate this approach in our setup. Specifically, we train a **logistic regression classifier** directly on LLM embedding-based descriptors or on scalar descriptors to assess whether a supervised linear decision boundary can further enhance hallucination detection performance.

6.4.5 Metrics Used

Hallucination detection is evaluated with both threshold-free and threshold-dependent metrics. For threshold selection, we consider two strategies: **Youden's J threshold**, which maximizes $J = \text{TPR} - \text{FPR}$ and yields a balanced operating point (close to maximizing balanced accuracy), and the **Target-TPR threshold** (set to 95%), which picks the threshold whose TPR is closest to 95%, a criterion commonly used in OOD detection to ensure high recall for OOD samples, and reports the corresponding FPR@95. Importantly, Target-TPR is recall-oriented rather than accuracy-optimal: achieving 95% TPR typically lowers the threshold, raising FPR and potentially reducing overall accuracy, especially when ID examples dominate.

Threshold-free evaluation includes **AUROC**, which measures overall separability of ID and OOD samples, and **AUC-PR**, computed with OOD as the positive class to remain robust under imbalance. Once a threshold is fixed (with samples above the threshold classified as OOD and those below as ID) we compute the confusion matrix along with **accuracy**, **precision**, **recall**, **F1 score**, and **FPR@95**. Confusion matrices can further be broken down by attributes such as answerable vs. unanswerable to analyze subgroup performance.

6.5 Implementation details

We briefly present the main components of the implementation pipeline. Full source code and technical detailed documentation are available on the GitHub repository[2].

Computing environment All experiments were implemented in Python 3.12.3 and executed on the remote servers of the IRT Saint-Exupery, equipped with NVIDIA L40S GPUs (48 GB VRAM, CUDA 12.4, driver 550.163). Model training and evaluation were accelerated using PyTorch with CUDA/cuDNN support. The full codebase was version-controlled and regularly updated on GitHub, ensuring reproducibility and traceability throughout the project.

Model and Dataset Loading We load the LLaMA 2-7B Chat model and two datasets: an ID "fit" set (containing answerable questions) and a mixed test set with both ID (answerable questions) and OOD (unanswerable questions) samples. The fit set is filtered as described in Section 6.2.2, retaining only samples for which the model produces an answer closely matching the ground truth (using Sentence-BERT and ROUGE-L similarity).

Prompt Formatting and Batch Processing We process samples in batches of 16, formatting each prompt according to the scheme described in Section 6.2.3, including special tags (`«SYS»`, `[INST]`) and initial text providing guidance on (un)answerability.

Custom Hooks for Efficient Representation Extraction Rather than relying on `model.generate()`'s build-in arguments `output_hidden_states=True` and `output_attentions=True`, we implement custom PyTorch hooks to extract internal representations (hidden states and attention maps) in a more memory-efficient way. These hooks enable precise targeting of specific Transformer layers within the model. In our configuration, we capture the output of every other Transformer block (e.g., layers 1, 3, 5, ...), where each block comprises a self-attention module, a feed-forward network (FFN), and associated normalization operations.

Attention Hook and Stability Patch One of the main challenges when extracting attention maps from Hugging Face's LLaMA models lies in the backend used for attention: The default `eager` backend exposes attention weights but suffers from numerical instability (e.g., NaNs and CUDA crashes). The default `sdpa` backend is numerically stable but does not expose attention weights, making it impossible to extract attention maps.

[2] https://github.com/LilaR66/ood_for_hallucination_detection

To address this, we implement a custom patch to the `forward` method of the `LlamaAttention` class, but only on the specific layers selected by our hooks. Inside this patch, the model's main forward pass continues to use the stable `sdpa` backend for generation ensuring reliable outputs. Simultaneously, we perform a secondary computation using a modified version of the `eager` backend to extract attention weights for interpretability. These attention maps are used exclusively for analysis and do not affect the model's forward pass or outputs, meaning any instability introduced during this side computation has no impact on generation.

Generation Parameters and Hook Execution We generate model responses using: `max_new_tokens=50`, `do_sample=True`, `temperature=0.6`, `top_p=0.9`, `top_k=50`. Beam search is disabled. We also set `output_hidden_states=False` and `output_attentions=False` since all representations are retrieved via hooks. The hooks automatically capture the hidden states and attention maps at the selected layers during generation and store them in memory.

Handling Hidden States and Attention Masks A subtle but important detail is that the model never returns the hidden state and attention map corresponding to the final token, as it is not used to generate the next token. Therefore, we must account for this behavior in our attention masks and in the computation of representation-based scores. For prompt inputs, attention masks are obtained via tokenization. For generated sequences, we manually construct masks that mark all tokens up to and including the first EOS as valid, and the rest as padding. This is crucial for batch processing to ensure that padding tokens are properly excluded from the computations. Additionally, we also support mask offsets to exclude certain prompt sections (e.g., system prompts in `«SYS»`) from representation analysis by padding them in the mask.

Representation Score Extraction For each selected layer, we compute the descriptors described in Section 6.4.2, including: (1) **Hidden States**: last token embedding, first token embedding, average token embedding, max token embedding, feature-wise variance of token embeddings, and Hidden Score. (2) **Attention Score**. (3) **Logit Scores**: Pseudo-Perplexity Score, Logit Entropy Score, and Windowed Logit Entropy Score.

We extract each of these metrics over three distinct segments of the sequence: prompt only, generated answer only, concatenated prompt + generation. Since `generate()` only returns logits from the final layer, we use the **LogitLens** method to compute logits at intermediate layers by passing extracted hidden states through the language modeling head.

Note: For perplexity computation, prompt logits are shifted left (`logit[t]` predicts token `t+1`), whereas for autoregressive generation, logits predict the current token (`logit[t]` predicts token `t`). We carefully implement these distinctions in our computations to ensure correctness. Moreover, In line with LLM Check [37], we use the configuration: `topk = 50`, `window size = 1` and `stride =1` for both the logit entropy and the windowed logit entropy computations.

Batch Vectorization and Custom Adaptations While our score definitions are inspired by prior work (LLM-Check [37]), we had to reimplement all functions from scratch to support: batch-wise processing of inputs (rather than sequential single-sample processing), vectorized computations across layers and tokens, correct masking of padding tokens. This required designing new functions rather than adapting existing scripts.

Saving Results For scalability, each batch's representations are saved as a separate `.pkl` file. These are later concatenated to avoid I/O bottlenecks and memory overload from progressively growing files.

6.6 Trajectory Analysis

As discussed in Section 3.3.2, aggregating token representations across transformer layers can produce more holistic and robust sentence embeddings. Motivated by this, and following [14], we analyze the layerwise trajectories of our previously defined descriptors (embedding-based and scalar) across the model's layers. While [14] shows that, in mathematical reasoning, ID and OOD samples follow measurably different embedding trajectories, it does not consider free-form text generation. We therefore test this hypothesis in our setting: for each descriptor, we track its evolution layer by layer and compare trajectories between answerable (ID) and unanswerable (OOD) cases. We conduct this analysis under two configurations: **prompt-only** and **generated-answer-only**.

Trajectory on embedding-based descriptors Let s denote a sample, L the number of transformer layers, and d the embedding dimension. For each layer $l \in \{1, .., L\}$, let $h_l(s) \in \mathbb{R}^d$ denote the embedding-based descriptor (e.g., last-token, first-token, average, max, or feature-wise variance). The embedding trajectory of s is :

$$h_1(s), h_2(s), \ldots, h_l(s), \ldots, h_L(s), \qquad h_l(s) \in \mathbb{R}^d \tag{6.16}$$

Following [14], the change between successive layers is quantified using the **dimension-joined volatility** score:

$$V_J(s) = \frac{1}{L-1} \sum_{l=1}^{L-1} ||h_{l+1}(s) - h_l(s)||_2 = \frac{1}{L-1} \sum_{l=1}^{L-1} \sqrt{\frac{1}{d} \sum_{i=1}^{d} (h_{l+1,i}(s) - h_{l,i}(s))^2} \quad \in \mathbb{R}$$

Where larger $V_J(s)$ indicates greater instability of the representation across layers.

Inspired by static OOD methods that score embeddings via Mahalanobis distance to an ID reference, [14] extends this idea to trajectories: the cluster of ID trajectories is treated as the reference and the deviation of a new trajectory is measured with respect to it. Concretely, for each layer l a Gaussian $G_l \sim \mathcal{N}(\mu_l, \Sigma_l)$ is fitted to the ID embeddings $\{h_l(s) : s \in \mathcal{D}_{\mathbf{ID}}\}$ (with $|\mathcal{D}_{\mathbf{ID}}| = N$):

$$\mu_l = \frac{1}{N} \sum_{s \in \mathcal{D}_{\mathbf{ID}}} h_l(s), \quad \Sigma_l = \operatorname{diag}(\sigma^2_{l,1}, \ldots, \sigma^2_{l,d}), \quad \sigma^2_{l,i} = \operatorname{Var}_s[h_{l,i}(s)] = \frac{1}{N} \sum_{s \in \mathcal{D}_{\mathbf{ID}}} (h_{l,i}(s) - \mu_{l,i})^2 \tag{6.17}$$

Given a new sample s, its layer-l descriptor is scored by the (squared) Mahalanobis distance to the layer's ID fit:

$$f(h_l(s)) = (h_l(s) - \mu_l)^\top \Sigma_l^{-1} (h_l(s) - \mu_l) = \sqrt{\sum_{i=1}^{d} \left(\frac{\left(h_{l,i}(s) - \mu^2_{l,i}\right)^2}{\sigma^2_{l,i}} \right)^2} \tag{6.18}$$

so higher $f(\cdot)$ means farther from the ID manifold at that layer.
To capture how s moves relative to the ID manifold across layers, the average absolute increment of the Mahalanobis score is measured and [14] define the **TV score** as:

$$\mathrm{TV}(s) = \frac{1}{L-1}\sum_{l=1}^{L-1} |f(h_{l+1}(s)) - f(h_l(s))| \in \mathbb{R} \tag{6.19}$$

Intuitively, ID samples tend to exhibit stable layerwise alignment with the per-layer ID distributions, while OOD or hallucination-prone samples exhibit volatile (rapidly changing) distances across layers. Thus, larger $TV(s)$ (and/or larger $V_J(s)$) indicates increased anomaly risk.

We therefore compute the **dimension-joined volatility** score and the **TV score** on our embeddings-based descriptors. Samples with scores above a chosen threshold are flagged as OOD.

Trajectory on scalar-based descriptors For scalar-based descriptors, we compute only the **dimension-joined volatility**, where the ℓ_2 norm between successive layers collapses to an absolute difference.

7 Analysis of results

7.1 Results in the Literature

Direct comparison with previous work remains difficult because no unified benchmark exists for hallucination detection. Each study introduces its own methodology, with different datasets, models, and evaluation metrics. As a result, performances are rarely comparable across setups, and no systematic survey has yet provided a comprehensive baseline.

Among the existing works, the ones most relevant to our setup are those conducted on datasets structurally close to SQuAD v2 (Hallucination *Evaluation* Benchmarks) and models comparable to LLaMA-2-7B. For example, **SAR** [32] reports an AUROC of 69.80 on CoQA with LLaMA-2-7B. Another relevant study **EUBHD** [33], achieves an AUC-PR of 84.26 on WikiBio GPT-3 with LLaMA-2-7B.

Other works rely on Hallucination *Detection* Benchmarks rather than Evaluation Benchmarks, which makes them less directly comparable to SQuAD since they do not require grounding answers in passages. For instance, **LLM-Check** [37] reports an AUROC of 72.34 on the FAVA-Annotation dataset with LLaMA-2-7B. Similarly, **CED** [56] evaluates LLaMA-3-8B on datasets such as Banking77, which are oriented toward intent classification rather than contextual question answering.

Several studies instead explore the stability of embeddings across multiple generations or reworded prompts, which is methodologically distinct from our approach. For example, **Knowing What LLMs Do Not Know** [36] evaluates LLaMA-2-13B on ComQA (closest dataset to SQuAD), achieving a PR-AUC of 52.36. Another notable work, **INSIDE** [35], reports an AUROC of 81.5 on SQuAD with LLaMA-2-7B.

The most common baseline across the literature remains the linear probe, trained in a supervised manner on hidden states: **The Curious Case of Hallucinatory (Un)answerability** [41] reports an AUROC of 90.4 with Flan-UL2 on question-answering tasks, **CCS** [39] obtains an AUROC of 71.5 with T5, **SAPLMA** [38] achieves an average accuracy of 0.83 with LLaMA-2-7B on factual versus false statements, **Inference-Time Intervention** [40] reports 53.5% truthful answers with LLaMA-7B on TruthfulQA, while **MIND** [44] reaches a sentence-level AUC of 67.55 with LLaMA-2-7B on the HELM dataset.

In summary, existing results are highly heterogeneous, reflecting differences in datasets, models, and evaluation protocols. Reported performances are often modest, especially for unsupervised methods, highlighting both the difficulty of hallucination detection and the need for standardized benchmarks to enable fair comparisons.

7.2 Results of OOD detection for Hallucination detection

We analyze OOD detection performance with results averaged over five runs using distinct random seeds. Due to computational and time constraints, each run uses a subsample of the original data: $10,000$ ID fit examples, $1,000$ unanswerable (OOD) test examples, and $1,000$ answerable (ID) test examples. Tables are ordered by AUROC. The top configurations are reported in Table 1, while the full results grid appears in Appendix A. To disentangle input from output-driven signals, we report findings separately for **prompt**-only and **generation**-based representations.

Prompt-Based Representations Within prompt-based representations, the strongest performance comes from **last-token contextualized embeddings** scored with distance-based OOD methods. Performance peaks in middle layers (13/15/17) with AUROC ≈ 0.59, then drops to ≈ 0.55 beyond these layers; most other descriptors hover around 0.50 (chance). This suggests that last-token embeddings capture some global semantics, but only within a limited depth range, consistent with Section 3.3.2, where middle layers sometimes yielded more holistic and robust embeddings for OOD detection. Among scalar (non-embedding) descriptors, logit-based metrics on the final layers follow (AUROC ≈ 0.53), with the Attention Score close behind (AUROC ≈ 0.52). By contrast, average and max pooling underperform, likely due to information loss from coarse aggregation. Overall, prompt-only AUROCs are just above chance. Linear probing consistently boosts performance on the same last-token embeddings: ≈ 0.80 (layers 13 to 21). However, for other weaker descriptors the probe collapses to ≈ 0.50, indicating little usable linear structure. This mirrors prior reports seen in Section 4.2.3: probes can exploit linear separability in embeddings but are supervised and often dataset-specific, limiting robustness.

Generation-Based Representations Prompt-based representations are slightly weaker for hallucination detection: the best prompt AUROC is ≈ 0.59, compared with ≈ 0.62 for generation-based representations. More broadly, generation features yield higher AUROCs and more strong configurations (many configurations > 0.52), indicating that hallucination cues are encoded during output, consistent with most prior hallucination detection work that analyzes the generated answer rather than the prompt. Top results come from **final-layer logit scores**: logit entropy (≈ 0.62 AUROC), windowed logit entropy (≈ 0.61), and perplexity (≈ 0.60). First-generated-token embeddings from intermediate/final layers also perform well (≈ 0.59 AUROC), consistent with the idea showed in paper [41] that they capture the model's initial intent just before full decoding. In contrast, applying distance-based OOD directly to other response embeddings performs poorly. A notable inversion appears with linear probes: despite strong AUROCs for logit scores, the probe accuracy on logits is only $0.50 - 0.55$, whereas probes on embedding + OOD configurations fare better. This suggests embeddings exhibit greater linear separability than logits, reinforcing both the utility and the limits of supervised probes.

Concatenated Prompt and Generation-Based Representations Finally, concatenating prompt and generation does not help. It tends to blur complementary signals (prompt favors embedding + OOD, generation favors logit scores) and degrades overall performance.

Overall performance is moderate, with best AUROCs in the low-0.6 range, consistent with prior reports that unsupervised hallucination detection is intrinsically difficult, as seen in Section 7.1. Supervised linear probes help primarily on embedding representations, but offer limited gains on logit-based or other scalar scores and may be dataset-specific. Distance-based OOD methods work best on embedding representations, particularly for prompt-only inputs, in line with prior OOD literature, where such methods are predominantly applied to input-side (prompt) embeddings rather than generated outputs. By contrast, logit-based metrics computed on generated tokens are stronger for hallucination detection, though gains remain modest. Moreover, applying distance-based OOD scores to generated embeddings does not improve over these logit baselines. Taken together, embedding-based OOD methods provide limited added value beyond generation-time logit signals for hallucination detection.

We hypothesize that our evaluation protocol on SQuAD 2.0 further tightens the task: "*Unanswerable*" is defined relative to the passage, not the model's parametric knowledge. Consequently, questions labeled OOD (unanswerable in context) may still be answerable by the model by drawing on its internal knowledge, potentially keeping ID/OOD representations close and reducing separability. Moreover, we suspect that typical Hallucination *Detection* Benchmarks, where a candidate response is already provided and the model's task is to determine whether it is factually grounded (as in LLM-Check [37]), may yield cleaner decision boundaries and thus present an easier detection scenario than our Hallucination *Evaluation* Setup. In the latter, the model is given a passage and a question, must first implicitly decide if the passage alone supports answering the question (abstaining otherwise) and then generate the answer; this more complex pipeline likely makes accurate hallucination detection more challenging. We note, however, that adopting such benchmarks would shift the task away from our original OOD framing: the objective here was, ideally, to predict whether a given input prompt will lead to hallucination before any generation by the model.

Table 1: OOD Detection Results on Prompt & Generated-Answer Representations. Top-10 configurations ranked by AUROC. Results averaged over 5 runs with distinct seeds and reported as mean ± std. For *embedding-based* descriptors, OOD scores are computed with DeepKNN (k=5); for *scalar* descriptors, raw scalar values are used directly. Reported metrics: **AUROC**, **FPR@95**, **AUPRC**, plus probe **accuracy** and **F1**. For full results, see Appendix A.

(a) Prompt representations

layer	group	aggregation	auroc ↑ (OOD)	fpr95 ↓ (OOD)	auprc ↑ (OOD)	acc ↑ (probe)	f1 ↑ (probe)
13	hidden	last_emb	59.41 ± 1.48	87.62 ± 1.87	56.20 ± 1.21	78.80 ± 2.46	79.05 ± 2.47
15	hidden	last_emb	58.79 ± 1.35	88.14 ± 1.26	55.46 ± 1.06	80.20 ± 2.25	80.04 ± 2.23
17	hidden	last_emb	58.73 ± 1.44	88.32 ± 1.92	55.41 ± 1.12	80.00 ± 4.46	79.94 ± 4.07
19	hidden	last_emb	54.86 ± 1.29	91.16 ± 1.32	52.54 ± 0.87	80.70 ± 2.44	80.27 ± 2.60
21	hidden	last_emb	54.43 ± 1.45	91.50 ± 1.33	52.19 ± 1.08	80.10 ± 4.13	79.94 ± 4.22
23	hidden	last_emb	54.31 ± 1.45	91.72 ± 1.33	52.09 ± 1.11	79.60 ± 2.38	79.55 ± 2.35
25	hidden	last_emb	53.48 ± 1.33	92.46 ± 1.37	51.38 ± 1.03	79.10 ± 2.75	78.79 ± 2.54
27	logit	perplexity_score	53.18 ± 0.52	94.02 ± 0.43	52.66 ± 0.54	53.10 ± 3.03	46.47 ± 5.14
23	logit	perplexity_score	53.07 ± 0.72	93.70 ± 0.47	52.54 ± 0.76	52.80 ± 3.56	47.88 ± 6.42
25	logit	perplexity_score	52.98 ± 0.61	93.84 ± 0.82	52.58 ± 0.67	52.80 ± 3.09	46.42 ± 5.40

(b) Generated-answer representations

layer	group	aggregation	auroc ↑ (OOD)	fpr95 ↓ (OOD)	auprc ↑ (OOD)	acc ↑ (probe)	f1 ↑ (probe)
-1	logit	logit_entropy_score	61.74 ± 0.55	87.42 ± 2.01	59.01 ± 0.65	58.60 ± 4.07	54.16 ± 3.33
-1	logit	window_logit_entropy_score	61.20 ± 0.48	88.10 ± 2.76	59.26 ± 1.05	56.70 ± 3.60	55.93 ± 3.61
-1	logit	perplexity_score	60.56 ± 0.75	87.68 ± 2.51	58.10 ± 0.94	57.40 ± 2.92	48.24 ± 3.84
27	hidden	first_gen_emb	58.86 ± 0.96	93.58 ± 1.16	59.71 ± 1.24	77.90 ± 2.36	77.82 ± 2.33
17	hidden	first_gen_emb	58.71 ± 1.14	91.02 ± 0.73	58.75 ± 1.12	82.50 ± 3.18	82.32 ± 3.33
13	hidden	first_gen_emb	58.68 ± 1.57	89.20 ± 1.21	55.75 ± 1.23	81.10 ± 3.19	81.10 ± 3.33
27	logit	window_logit_entropy_score	58.67 ± 0.73	88.94 ± 1.38	56.96 ± 0.46	54.90 ± 2.75	58.32 ± 2.89
15	hidden	first_gen_emb	58.64 ± 1.61	88.82 ± 1.40	55.63 ± 1.18	81.30 ± 1.99	81.30 ± 2.46
25	hidden	first_gen_emb	58.62 ± 1.06	93.72 ± 1.20	59.58 ± 1.41	77.70 ± 2.68	77.26 ± 3.46
29	hidden	first_gen_emb	58.46 ± 0.89	93.70 ± 1.16	59.23 ± 1.26	77.70 ± 1.60	77.32 ± 2.06

Confusion-Matrix Analysis and Abstention Behavior Having identified the strongest OOD configurations in Table 1, we next examine decision patterns at the selected operating point. Specifically, we analyze row-normalized confusion matrices evaluated at Youden's threshold for the best configurations for prompt-only representations (layer 13, Last token embedding) in Figure 1 and generation-only representations (last layer, Logit Entropy Score) in Figure 2. In both settings, the model is prompted with context + question and explicitly instructed (via a pre-prompt; see Section 6.2.3) to either produce an answer or return *"Unanswerable"*. Each cell therefore also reports the share of generations labeled Answerable (A) vs Unanswerable (U).

For both representations, patterns are stable across runs. On True ID, the model rarely abstains ($U \approx 8\%, A \approx 92\%$). On True OOD, abstention rises but remains a minority ($U \approx 20\%, A \approx 80\%$). Thus, the model is roughly 2–3× more likely to answer *Unanswerable* on OOD than on ID, yet most OOD items are still answered. Importantly, within each ground-truth row the A/U split is nearly identical across prediction columns. This shows the OOD detector's decision is not simply tracking the *Unanswerable* flag; abstention is only weakly correlated with the OOD score. Notably, the (True OOD → Pred. OOD) cell contains mostly Answerable outputs ($A \approx 80\%$). In other words, the detector frequently flags hallucinated answers even when the model chooses to answer rather than abstain, showing that our method can, to some extent, identify instances where the model "lies" and answers despite lacking support.

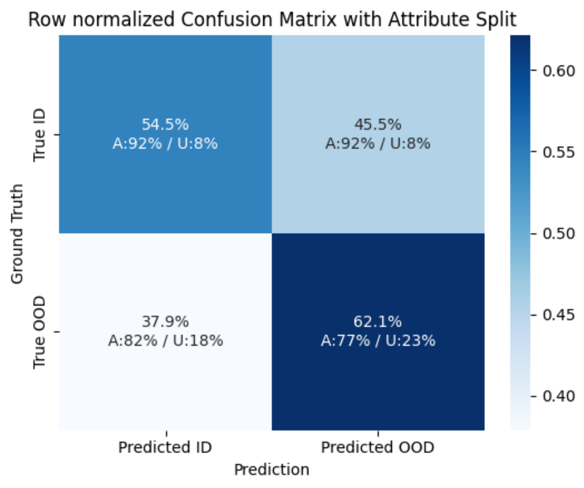


Figure 1: Row-normalized confusion matrix at Youden's J operating point for the **prompt-only, Last-token Embedding** (layer 13). OOD method: DeepKNN ($k = 5$), seed = 44.

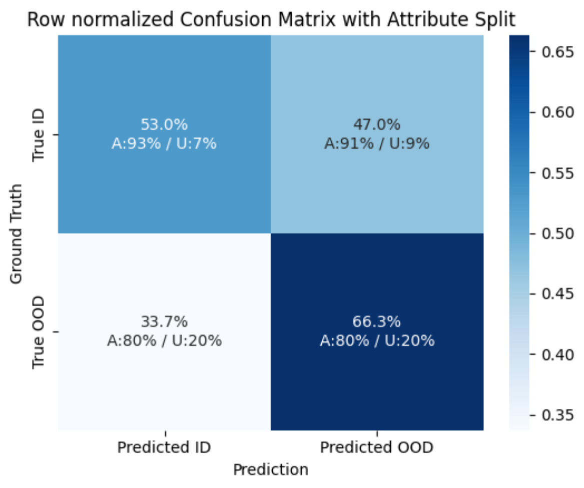


Figure 2: Row-normalized confusion matrix at Youden's J operating point for the **generation-only, Logit Entropy score** (last layer). No OOD method (raw scores), seed = 44.

7.3 Results of Trajectory Analysis for Hallucination detection

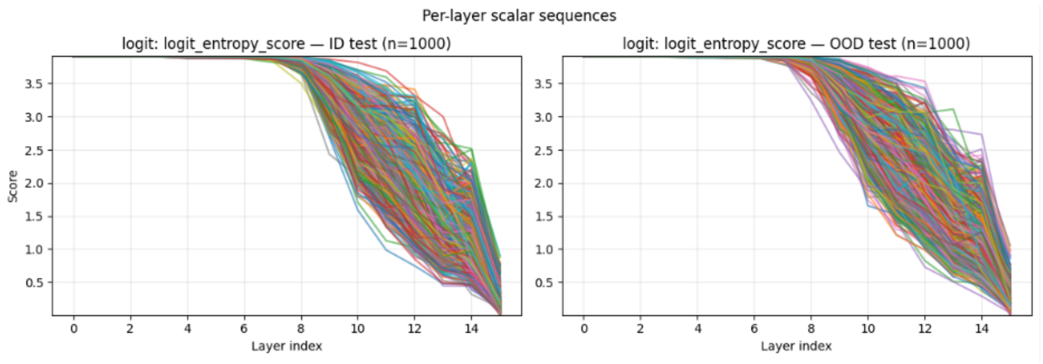
We evaluate layerwise trajectories for both scalar and embedding-based descriptors under two configurations: prompt-only and generated-answer-only. For scalars, visual inspection of the 1D traces shows no obvious separation between ID and OOD (Fig. 3).

We then compute the **dimension-joined volatility** for scalars, and for embeddings both **dimension-joined volatility** and the **TV score**. Contrary to our expectation (Section 6.6) that OOD trajectories would be less stable and have a higher volatility than ID, we observe the opposite: ID trajectories fluctuate more than OOD. To align with the standard convention "larger score → more OOD", we therefore use the negated quantities of the dimension-joined volatility and TV score as detection scores. With this sign flip, the best generation-only results are obtained with logit entropy and windowed logit entropy (AUROC ≈ 0.59–0.60), whereas all other descriptors operate at chance level. For prompt-only results, the strongest configuration, last-token embedding, yields AUROC ≈ 0.55 with dimension-joined volatility and ≈ 0.55 with TV score, i.e., no better than chance for TV score and weaker than static descriptor + OOD scoring. All other embedding descriptors are similarly near chance.

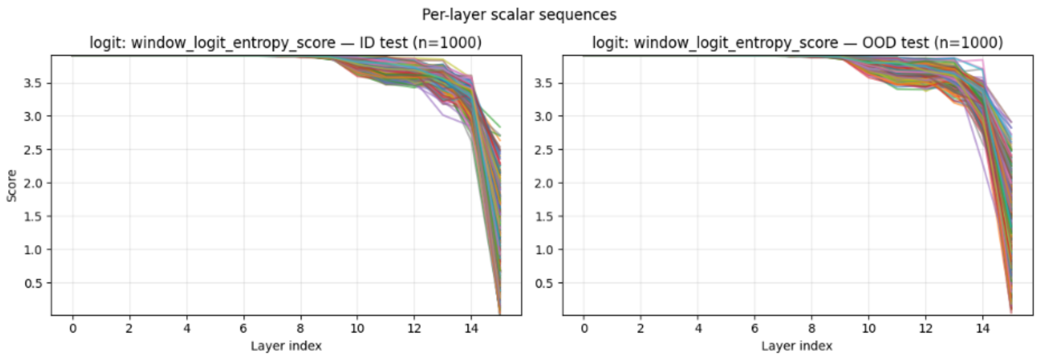
Overall, trajectory-based features do not enhance hallucination detection in our setting and generally underperform both static, distance-based OOD scoring and generation-time logit signals.

A plausible explanation, consistent with [14], is that in mathematical reasoning, final answer embeddings collapse into a narrow region (*pattern collapse*); with endpoints constrained, layer-wise paths must diverge, making trajectory volatility highly informative for OOD. In our free-form QA setup, we observe no endpoint collapse: generated answers are linguistically diverse and their final embeddings are widely dispersed, leaving little systematic path divergence to exploit. We also hypothesize this explains the apparent sign inversion: ID items may show greater volatility as the model actively refines its internal state using the passage, whereas OOD/unanswerable items may stabilize earlier into fallback patterns, yielding flatter trajectories.

Figure 3: Layer-wise trajectories of scalar descriptors for **generation-only** representations. Each panel traces the descriptor value across transformer layers for ID and OOD samples. Trajectories largely overlap, indicating weak raw separability.



(a) Logit Entropy descriptor across layers.



(b) Windos Entropy descriptor across layers.

8 Conclusion and Future Work

In this work, we investigated hallucination detection in large language models (LLMs) through the lens of out-of-distribution (OOD) detection. We first conducted an extensive literature review, covering the state of the art in out-of-distribution (OOD) detection and hallucination detection as of today. We presented this survey in a concise and structured manner, aiming to gather all available information on these two domains, which, it should be noted, currently lack both a generalized benchmark and a comprehensive survey, especially in the hallucination detection field. Next, we applied post-hoc, distance-based OOD detection methods on internal representations of LLaMA-2-7B Chat, utilizing SQuAD 2.0 as a benchmark dataset. We extracted and analyzed a broad range of internal features, including hidden states, attention maps, and logit-based uncertainty scores. We did so across several transformer layers and under both prompt-only and generation-based configurations. We then evaluated OOD scoring approaches such as DeepKNN, Mahalanobis distance, and cosine similarity. Additionally, we incorporated anomaly detection algorithms and supervised linear probes to assess their ability to signal hallucination risk. These methods leveraged the rich structure of the model's internal embeddings.

Summary of Findings Our empirical analysis yielded several insights. Unsupervised OOD detection offered modest discriminative power, with prompt-only embedding-based methods performing less well (best AUROC ≈ 0.59 on middle layers with last-token embeddings + OOD scoring) compared to logit-based baselines on generated outputs (best AUROC ≈ 0.62 with last-layer logit entropy, without additional OOD methods). Although classical NLP OOD literature generally finds that embedding distance-based methods outperform logit-based ones (Section 3.3.2), we observed that this advantage did not reliably hold for hallucination detection. Indeed, logit-based metrics derived from generated answers often surpass embedding-based methods applied to prompt or response representations, although first-token embeddings from the generation at middle layers performed competitively. Detecting hallucinations solely from prompt representations, hence before generation, remains particularly challenging, suggesting hallucinations are largely a generation phenomenon.

Moreover, confusion matrices with the Answerable/Unanswerable attribute split (Section 7.2) for our best-performing OOD configurations showed that detector predictions only weakly align with the model's abstention from answering (i.e., returning *"Unanswerable"*). Although the model abstained more frequently on OOD than on ID samples, most OOD samples were still answered. Therefore, the detector provided an independent signal of hallucination risk: many true OOD items predicted as OOD remained labeled Answerable by the model, indicating that it flagged unsupported or misleading answers even when the model does not abstain.

Difficulty in this protocol This difficulty stems from the fine-grained nature of hallucination phenomena: unlike traditional OOD settings where in-distribution (ID) and out-of-distribution (OOD) embeddings diverge widely, hallucinations differ only in certain tokens, leading to closely overlapping embeddings between hallucinated and faithful outputs. Furthermore, in our evaluation protocol, *"Unanswerable"* questions are defined relative to the provided passage, not the model's parametric knowledge. As such, many OOD (unanswerable) items remain answerable by the model's internal knowledge, compressing the geometric margin between ID and OOD representations. This may explains why distances computed on prompt embeddings underperform compared to generation-time uncertainty scores, and why applying OOD scores directly to generated embeddings does not significantly outperform simple logit-based baselines.

Limitations of Supervised Approaches While supervised linear probes improved classification accuracy (up to 0.80 on embedding descriptors), as reported broadly in hallucination detection literature (Section 4.2.3), their generalization is highly dependent on dataset and task specifics. Moreover, as probes require training on representative labeled datasets, their practical applicability is limited given the vast complexity of real-world hallucination phenomena.

Prior studies [42, 43] indicate that the latent "geometry of truth" in LLMs is highly task specific and transfers poorly across datasets and use cases. Linear-probe directions learned on one task are often nearly orthogonal to those for another, and aggregating probes across tasks brings little benefit. This weak cross-domain generalization cautions against relying on a single supervised probe for broad hallucination detection. Nonetheless, in highly specialized domains with well curated and relatively stable corpora (e.g., enterprise-specific document collections), supervised probes may prove valuable where they can flag inputs that fall outside internal knowledge boundaries, provided they are periodically refreshed as the corpus evolves.

Marginal Impact of Layer-Wise Trajectories Our investigation into layer-wise trajectories of internal representations revealed limited gains in hallucination detection performance. Both scalar and embedding-based trajectory features exhibited weak separability between answerable and unanswerable inputs, underscoring the challenge of leveraging dynamic embedding evolution in free-form generation tasks. By contrast, the study [14] has reported clearer benefits of trajectory-based signals in more constrained reasoning tasks like mathematical problem solving, where representations tend to converge within narrower manifolds and follow more stereotyped evolutionary paths.

Directions for Future Work Building on these conclusions, we identify several avenues for further research:

- **Domain Restriction**: Adopting more domain-specific models and datasets, similar to the mathematical reasoning setting seen in [14], could potentially ease the hallucination detection task and improve performance.
- **Dataset Scaling**: Our compute budget limited us to a *fit* split of $10,000$ and *test* splits of $1,000$ ID (answerable) and $1,000$ OOD (unanswerable). After screening and selecting a promising configuration and layer, increasing the size of the fit and evaluation datasets beyond our computationally constrained would enable more robust evaluation.
- **Dataset variety and evaluation considerations.** Our current evaluation protocol using SQuAD 2.0 (Hallucination *Evaluation* Benchmark) likely tightens the hallucination detection task. Specifically, *"Unanswerable"* is defined relative to the passage context rather than the model's internal knowledge. As a result, questions labeled as OOD within this dataset may still be answerable by the large language model relying on its internal knowledge, which keeps the embeddings of ID and OOD samples close. This proximity reduces the separability of hallucinated and truthful responses, complicating detection. Other common Hallucination *Detection* Benchmarks typically provide candidate responses that have already been generated, making the detection task primarily about verifying the factuality of these outputs in context. Such benchmarks may present cleaner decision boundaries and simpler detection scenarios compared to our Hallucination Evaluation Setup. However, such datasets differ from our original objective of pre-generation OOD detection, where the goal is to predict hallucination risk before the model begins decoding.
- **Exploration of Generation-Based Logit Metrics**: Given the relatively strong performance of logit-based scores at the final transformer layer, implementing and studying more sophisticated logit scoring methods is an attractive direction.
- **Generalized benchmarks for NLP OOD and hallucination detection.** More genrally, develop robust, large scale, and standardized benchmark suites for NLP OOD detection and hallucination detection in LLMs that incorporate contextual and temporal shifts, clearly separate "unanswerable with respect to the passage" from "unknown to the model", and define shared protocols with risk aware metrics (e.g., AUROC, AUPRC, FPR@95, risk–coverage).

References

[1] Paul Novello, Yannick Prudent, Joseba Dalmau, Corentin Friedrich, and Yann Pequignot. *Improving Out-of-Distribution Detection by Combining Existing Post-hoc Methods*. 2024. arXiv: 2407.07135 [stat.ML]. URL: https://arxiv.org/abs/2407.07135.

[2] Jingkang Yang, Pengyun Wang, Dejian Zou, Zitang Zhou, Kunyuan Ding, Wenxuan Peng, Haoqi Wang, Guangyao Chen, Bo Li, Yiyou Sun, Xuefeng Du, Kaiyang Zhou, Wayne Zhang, Dan Hendrycks, Yixuan Li, and Ziwei Liu. *OpenOOD: Benchmarking Generalized Out-of-Distribution Detection*. 2022. arXiv: 2210.07242 [cs.CV]. URL: https://arxiv.org/abs/2210.07242.

[3] Jingyang Zhang, Jingkang Yang, Pengyun Wang, Haoqi Wang, Yueqian Lin, Haoran Zhang, Yiyou Sun, Xuefeng Du, Yixuan Li, Ziwei Liu, Yiran Chen, and Hai Li. *OpenOOD v1.5: Enhanced Benchmark for Out-of-Distribution Detection*. 2024. arXiv: 2306.09301 [cs.LG]. URL: https://arxiv.org/abs/2306.09301.

[4] Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. *Generalized Out-of-Distribution Detection: A Survey*. 2024. arXiv: 2110.11334 [cs.CV]. URL: https://arxiv.org/abs/2110.11334.

[5] Hao Lang, Yinhe Zheng, Yixuan Li, Jian Sun, Fei Huang, and Yongbin Li. *A Survey on Out-of-Distribution Detection in NLP*. 2023. arXiv: 2305.03236 [cs.CL]. URL: https://arxiv.org/abs/2305.03236.

[6] Mateusz Baran, Joanna Baran, Mateusz Wójcik, Maciej Zięba, and Adam Gonczarek. *Classical Out-of-Distribution Detection Methods Benchmark in Text Classification Tasks*. 2023. arXiv: 2307.07002 [cs.CL]. URL: https://arxiv.org/abs/2307.07002.

[7] Dan Hendrycks, Xiaoyuan Liu, Eric Wallace, Adam Dziedzic, Rishabh Krishnan, and Dawn Song. *Pretrained Transformers Improve Out-of-Distribution Robustness*. 2020. arXiv: 2004.06100 [cs.CL]. URL: https://arxiv.org/abs/2004.06100.

[8] Artem Vazhentsev, Akim Tsvigun, Roman Vashurin, Sergey Petrakov, Daniil Vasilev, Maxim Panov, Alexander Panchenko, and Artem Shelmanov. "Efficient Out-of-Domain Detection for Sequence to Sequence Models". In: *Findings of the Association for Computational Linguistics: ACL 2023*. Ed. by Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki. Toronto, Canada: Association for Computational Linguistics, July 2023, pp. 1430–1454. DOI: 10.18653/v1/2023.findings-acl.93. URL: https://aclanthology.org/2023.findings-acl.93/.

[9] Udit Arora, William Huang, and He He. *Types of Out-of-Distribution Texts and How to Detect Them*. 2021. arXiv: 2109.06827 [cs.CL]. URL: https://arxiv.org/abs/2109.06827.

[10] Wenxuan Zhou, Fangyu Liu, and Muhao Chen. "Contrastive Out-of-Distribution Detection for Pretrained Transformers". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2021. DOI: 10.18653/v1/2021.emnlp-main.84. URL: http://dx.doi.org/10.18653/v1/2021.emnlp-main.84.

[11] Sishuo Chen, Xiaohan Bi, Rundong Gao, and Xu Sun. "(Avg-Avg) Holistic sentence embeddings for better out-of-distribution detection". In: *arXiv preprint arXiv:2210.07485* (2022).

[12] Bo Liu, Liming Zhan, Zexin Lu, Yujie Feng, Lei Xue, and Xiao-Ming Wu. *How Good Are LLMs at Out-of-Distribution Detection?* 2024. arXiv: 2308.10261 [cs.CL]. URL: https://arxiv.org/abs/2308.10261.

[13] Hyunsoo Cho, Choonghyun Park, Junyeop Kim, Hyuhng Joon Kim, Kang Min Yoo, and Sang-goo Lee. *Probing Out-of-Distribution Robustness of Language Models with Parameter-Efficient Transfer Learning*. 2023. arXiv: 2301.11660 [cs.CL]. URL: https://arxiv.org/abs/2301.11660.

[14] Yiming Wang, Pei Zhang, Baosong Yang, Derek F. Wong, Zhuosheng Zhang, and Rui Wang. *Embedding Trajectory for Out-of-Distribution Detection in Mathematical Reasoning*. 2024. arXiv: 2405.14039 [cs.CL]. URL: https://arxiv.org/abs/2405.14039.

[15] Stanislav Fort, Jie Ren, and Balaji Lakshminarayanan. *Exploring the Limits of Out-of-Distribution Detection*. 2021. arXiv: 2106.03004 [cs.LG]. URL: https://arxiv.org/abs/2106.03004.

[16] Lifan Yuan, Yangyi Chen, Ganqu Cui, Hongcheng Gao, Fangyuan Zou, Xingyi Cheng, Heng Ji, Zhiyuan Liu, and Maosong Sun. *(BOSS) Revisiting Out-of-distribution Robustness in NLP: Benchmark, Analysis, and LLMs Evaluations*. 2023. arXiv: 2306.04618 [cs.CL]. URL: https://arxiv.org/abs/2306.04618.

[17] Rheeya Uppaal, Junjie Hu, and Yixuan Li. *Is Fine-tuning Needed? Pre-trained Language Models Are Near Perfect for Out-of-Domain Detection*. 2023. arXiv: 2305.13282 [cs.CL]. URL: https://arxiv.org/abs/2305.13282.

[18] Dan Hendrycks and Kevin Gimpel. *A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks*. 2018. arXiv: 1610.02136 [cs.NE]. URL: https://arxiv.org/abs/1610.02136.

[19] Weitang Liu, Xiaoyun Wang, John Owens, and Yixuan Li. “Energy-based out-of-distribution detection”. In: *Advances in neural information processing systems* 33 (2020), pp. 21464–21475.

[20] Yiyou Sun, Yifei Ming, Xiaojin Zhu, and Yixuan Li. *Out-of-Distribution Detection with Deep Nearest Neighbors*. 2022. arXiv: 2204.06507 [cs.LG]. URL: https://arxiv.org/abs/2204.06507.

[21] Alexander Podolskiy, Dmitry Lipin, Andrey Bout, Ekaterina Artemova, and Irina Piontkovskaya. *Revisiting Mahalanobis Distance for Transformer-Based Out-of-Domain Detection*. 2022. arXiv: 2101.03778 [cs.CL]. URL: https://arxiv.org/abs/2101.03778.

[22] Hyunsoo Cho, Choonghyun Park, Jaewook Kang, Kang Min Yoo, Taeuk Kim, and Sang-goo Lee. *Enhancing Out-of-Distribution Detection in Natural Language Understanding via Implicit Layer Ensemble*. 2022. arXiv: 2210.11034 [cs.CL]. URL: https://arxiv.org/abs/2210.11034.

[23] Harry Anthony and Konstantinos Kamnitsas. *On the use of Mahalanobis distance for out-of-distribution detection with neural networks for medical imaging*. 2023. arXiv: 2309.01488 [cs.CV]. URL: https://arxiv.org/abs/2309.01488.

[24] Betty van Aken, Benjamin Winter, Alexander Löser, and Felix A. Gers. *VisBERT: Hidden-State Visualizations for Transformers*. 2020. arXiv: 2011.04507 [cs.CL]. URL: https://arxiv.org/abs/2011.04507.

[25] Nora Belrose, Igor Ostrovsky, Lev McKinney, Zach Furman, Logan Smith, Danny Halawi, Stella Biderman, and Jacob Steinhardt. *Eliciting Latent Predictions from Transformers with the Tuned Lens*. 2025. arXiv: 2303.08112 [cs.LG]. URL: https://arxiv.org/abs/2303.08112.

[26] Jaturong Kongmanee. *Unraveling Token Prediction Refinement and Identifying Essential Layers in Language Models*. 2025. arXiv: 2501.15054 [cs.CL]. URL: https://arxiv.org/abs/2501.15054.

[27] Linyi Yang, Shuibai Zhang, Libo Qin, Yafu Li, Yidong Wang, Hanmeng Liu, Jindong Wang, Xing Xie, and Yue Zhang. *GLUE-X: Evaluating Natural Language Understanding Models from an Out-of-distribution Generalization Perspective*. 2023. arXiv: 2211.08073 [cs.CL]. URL: https://arxiv.org/abs/2211.08073.

[28] Jindong Wang, Xixu Hu, Wenxin Hou, Hao Chen, Runkai Zheng, Yidong Wang, Linyi Yang, Haojun Huang, Wei Ye, Xiubo Geng, Binxin Jiao, Yue Zhang, and Xing Xie. *On the Robustness of ChatGPT: An Adversarial and Out-of-distribution Perspective*. 2023. arXiv: 2302.12095 [cs.AI]. URL: https://arxiv.org/abs/2302.12095.

[29] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. "A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions". In: *ACM Transactions on Information Systems* 43.2 (Jan. 2025), pp. 1–55. ISSN: 1558-2868. DOI: 10.1145/3703155. URL: http://dx.doi.org/10.1145/3703155.

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

[31] Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. *Semantic Uncertainty: Linguistic Invariances for Uncertainty Estimation in Natural Language Generation*. 2023. arXiv: 2302.09664 [cs.CL]. URL: https://arxiv.org/abs/2302.09664.

[32] Jinhao Duan, Hao Cheng, Shiqi Wang, Alex Zavalny, Chenan Wang, Renjing Xu, Bhavya Kailkhura, and Kaidi Xu. *(SAR) Shifting Attention to Relevance: Towards the Predictive Uncertainty Quantification of Free-Form Large Language Models*. 2024. arXiv: 2307.01379 [cs.CL]. URL: https://arxiv.org/abs/2307.01379.

[33] Tianhang Zhang, Lin Qiu, Qipeng Guo, Cheng Deng, Yue Zhang, Zheng Zhang, Chenghu Zhou, Xinbing Wang, and Luoyi Fu. *(EUBHD) Enhancing Uncertainty-Based Hallucination Detection with Stronger Focus*. 2023. arXiv: 2311.13230 [cs.CL]. URL: https://arxiv.org/abs/2311.13230.

[34] Potsawee Manakul, Adian Liusie, and Mark J. F. Gales. *SelfCheckGPT: Zero-Resource Black-Box Hallucination Detection for Generative Large Language Models*. 2023. arXiv: 2303.08896 [cs.CL]. URL: https://arxiv.org/abs/2303.08896.

[35] Chao Chen, Kai Liu, Ze Chen, Yi Gu, Yue Wu, Mingyuan Tao, Zhihang Fu, and Jieping Ye. *INSIDE: LLMs' Internal States Retain the Power of Hallucination Detection*. 2024. arXiv: 2402.03744 [cs.CL]. URL: https://arxiv.org/abs/2402.03744.

[36] Yukun Zhao, Lingyong Yan, Weiwei Sun, Guoliang Xing, Chong Meng, Shuaiqiang Wang, Zhicong Cheng, Zhaochun Ren, and Dawei Yin. *Knowing What LLMs DO NOT Know: A Simple Yet Effective Self-Detection Method*. 2024. arXiv: 2310.17918 [cs.CL]. URL: https://arxiv.org/abs/2310.17918.

[37] Gaurang Sriramanan, Siddhant Bharti, Vinu Sankar Sadasivan, Shoumik Saha, Priyatham Kattakinda, and Soheil Feizi. "Llm-check: Investigating detection of hallucinations in large language models". In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 34188–34216.

[38] Amos Azaria and Tom Mitchell. *(SAPLMA) The Internal State of an LLM Knows When It's Lying*. 2023. arXiv: 2304.13734 [cs.CL]. URL: https://arxiv.org/abs/2304.13734.

[39] Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. *(CCS) Discovering Latent Knowledge in Language Models Without Supervision*. 2024. arXiv: 2212.03827 [cs.CL]. URL: https://arxiv.org/abs/2212.03827.

[40] Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. *Inference-Time Intervention: Eliciting Truthful Answers from a Language Model*. 2024. arXiv: 2306.03341 [cs.LG]. URL: https://arxiv.org/abs/2306.03341.

[41] Aviv Slobodkin, Omer Goldman, Avi Caciularu, Ido Dagan, and Shauli Ravfogel. *The Curious Case of Hallucinatory (Un)answerability: Finding Truths in the Hidden States of Over-Confident Large Language Models*. 2023. arXiv: 2310.11877 [cs.CL]. URL: https://arxiv.org/abs/2310.11877.

[42] Benjamin A. Levinstein and Daniel A. Herrmann. "Still no lie detector for language models: probing empirical and conceptual roadblocks". In: *Philosophical Studies* 182.7 (Feb. 2024), pp. 1539–1565. ISSN: 1573-0883. DOI: 10.1007/s11098-023-02094-3. URL: http://dx.doi.org/10.1007/s11098-023-02094-3.

[43] Waiss Azizian, Michael Kirchhof, Eugene Ndiaye, Louis Bethune, Michal Klein, Pierre Ablin, and Marco Cuturi. *The Geometries of Truth Are Orthogonal Across Tasks*. 2025. arXiv: 2506.08572 [cs.LG]. URL: https://arxiv.org/abs/2506.08572.

[44] Weihang Su, Changyue Wang, Qingyao Ai, Yiran HU, Zhijing Wu, Yujia Zhou, and Yiqun Liu. *(MIND) Unsupervised Real-Time Hallucination Detection based on the Internal States of Large Language Models*. 2024. arXiv: 2403.06448 [cs.CL]. URL: https://arxiv.org/abs/2403.06448.

[45] Yung-Sung Chuang, Linlu Qiu, Cheng-Yu Hsieh, Ranjay Krishna, Yoon Kim, and James Glass. *Lookback Lens: Detecting and Mitigating Contextual Hallucinations in Large Language Models Using Only Attention Maps*. 2024. arXiv: 2407.07071 [cs.CL]. URL: https://arxiv.org/abs/2407.07071.

[46] Junyu Luo, Cao Xiao, and Fenglong Ma. *Zero-Resource Hallucination Prevention for Large Language Models*. 2023. arXiv: 2309.02654 [cs.CL]. URL: https://arxiv.org/abs/2309.02654.

[47] Neeraj Varshney, Wenlin Yao, Hongming Zhang, Jianshu Chen, and Dong Yu. *A Stitch in Time Saves Nine: Detecting and Mitigating Hallucinations of LLMs by Validating Low-Confidence Generation*. 2023. arXiv: 2307.03987 [cs.CL]. URL: https://arxiv.org/abs/2307.03987.

[48] Xiaomeng Hu, Yiming Zhang, Ru Peng, Haozhe Zhang, Chenwei Wu, Gang Chen, and Junbo Zhao. "(EGH) Embedding and Gradient Say Wrong: A White-Box Method for Hallucination Detection". In: *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. Ed. by Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen. Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 1950–1959. DOI: 10.18653/v1/2024.emnlp-main.116. URL: https://aclanthology.org/2024.emnlp-main.116/.

[49] Cheng Niu, Yuanhao Wu, Juno Zhu, Siliang Xu, Kashun Shum, Randy Zhong, Juntong Song, and Tong Zhang. *RAGTruth: A Hallucination Corpus for Developing Trustworthy Retrieval-Augmented Language Models*. 2024. arXiv: 2401.00396 [cs.CL]. URL: https://arxiv.org/abs/2401.00396.

[50] Jiawei Zhang, Chejian Xu, Yu Gai, Freddy Lecue, Dawn Song, and Bo Li. *KnowHalu: Hallucination Detection via Multi-Form Knowledge Based Factual Checking*. 2024. arXiv: 2404.02935 [cs.CL]. URL: https://arxiv.org/abs/2404.02935.

[51] Jie Ren, Jiaming Luo, Yao Zhao, Kundan Krishna, Mohammad Saleh, Balaji Lakshminarayanan, and Peter J. Liu. *Out-of-Distribution Detection and Selective Generation for Conditional Language Models*. 2023. arXiv: 2209.15558 [cs.CL]. URL: https://arxiv.org/abs/2209.15558.

[52] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. "Llama-2: Open Foundation and Fine-Tuned Chat Models". In: *arXiv preprint arXiv:2307.09288* (2023). URL: https://arxiv.org/abs/2307.09288.

[53] Pranav Rajpurkar, Robin Jia, and Percy Liang. *Know What You Don't Know: Unanswerable Questions for SQuAD*. 2018. arXiv: 1806.03822 [cs.CL]. URL: https://arxiv.org/abs/1806.03822.

[54] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. *SQuAD: 100,000+ Questions for Machine Comprehension of Text*. 2016. arXiv: 1606.05250 [cs.CL]. URL: https://arxiv.org/abs/1606.05250.

[55] Olivier Ledoit and Michael Wolf. "A well-conditioned estimator for large-dimensional covariance matrices". In: *Journal of Multivariate Analysis* 88.2 (2004), pp. 365–411. ISSN: 0047-259X. DOI: https://doi.org/10.1016/S0047-259X(03)00096-4. URL: https://www.sciencedirect.com/science/article/pii/S0047259X03000964.

[56] Hakyung Lee, Keon-Hee Park, Hoyoon Byun, Jeyoon Yeom, Jihee Kim, Gyeong-Moon Park, and Kyungwoo Song. "CED: Comparing Embedding Differences for Detecting Out-of-Distribution and Hallucinated Text". In: *Findings of the Association for Computational Linguistics: EMNLP 2024*. Ed. by Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen. Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 14866–14882. DOI: 10.18653/v1/2024.findings-emnlp.874. URL: https://aclanthology.org/2024.findings-emnlp.874/.

A Complete Experimental Results

Table 2: OOD Detection Results on Prompt Representations ranked by AUROC. Results averaged over 5 runs with distinct seeds and reported as mean ± std. For *embedding-based* descriptors, OOD scores are computed with DeepKNN (k=5); for *scalar* descriptors, raw scalar values are used directly. Reported metrics: **AUROC**, **FPR@95**, **AUPRC**, plus probe **accuracy** and **F1**.

layer	**aggregation**	**auroc ↑ (OOD)**	**fpr95 ↓ (OOD)**	**auprc ↑ (OOD)**	**acc ↑ (probe)**	**f1 ↑ (probe)**
13	last_emb	59.41 ± 1.48	87.62 ± 1.87	56.20 ± 1.21	78.80 ± 2.46	79.05 ± 2.47
15	last_emb	58.79 ± 1.35	88.14 ± 1.26	55.46 ± 1.06	80.20 ± 2.25	80.04 ± 2.23
17	last_emb	58.73 ± 1.44	88.32 ± 1.92	55.41 ± 1.12	80.00 ± 4.46	79.94 ± 4.07
19	last_emb	54.86 ± 1.29	91.16 ± 1.32	52.54 ± 0.87	80.70 ± 2.44	80.27 ± 2.60
21	last_emb	54.43 ± 1.45	91.50 ± 1.33	52.19 ± 1.08	80.10 ± 4.13	79.94 ± 4.22
23	last_emb	54.31 ± 1.45	91.72 ± 1.33	52.09 ± 1.11	79.60 ± 2.38	79.55 ± 2.35
25	last_emb	53.48 ± 1.33	92.46 ± 1.37	51.38 ± 1.03	79.10 ± 2.75	78.79 ± 2.54
27	perplexity_score	53.18 ± 0.52	94.02 ± 0.43	52.66 ± 0.54	53.10 ± 3.03	46.47 ± 5.14
23	perplexity_score	53.07 ± 0.72	93.70 ± 0.47	52.54 ± 0.76	52.80 ± 3.56	47.88 ± 6.42
25	perplexity_score	52.98 ± 0.61	93.84 ± 0.82	52.58 ± 0.67	52.80 ± 3.09	46.42 ± 5.40
29	perplexity_score	52.89 ± 0.38	93.56 ± 0.73	52.36 ± 0.38	52.40 ± 3.94	45.69 ± 5.41
21	perplexity_score	52.70 ± 0.72	93.60 ± 0.70	52.15 ± 0.72	52.80 ± 3.31	48.33 ± 5.21
13	attn_score	52.16 ± 0.49	93.90 ± 1.09	52.08 ± 0.56	53.30 ± 3.09	53.03 ± 3.32
23	logit_entropy_score	52.16 ± 1.51	94.72 ± 1.02	51.94 ± 1.60	52.10 ± 2.04	52.61 ± 2.11
25	attn_score	52.09 ± 0.54	93.90 ± 0.87	51.44 ± 0.71	52.10 ± 2.68	52.52 ± 2.41
25	logit_entropy_score	51.93 ± 1.18	94.52 ± 0.95	51.54 ± 1.12	52.30 ± 4.98	52.08 ± 4.65
19	perplexity_score	51.92 ± 0.57	93.66 ± 0.67	51.37 ± 0.53	51.80 ± 3.29	48.34 ± 5.29
11	attn_score	51.87 ± 0.59	93.90 ± 1.37	51.66 ± 0.66	54.20 ± 2.08	54.04 ± 2.67
27	logit_entropy_score	51.85 ± 1.08	94.12 ± 0.57	51.57 ± 1.07	51.10 ± 4.17	49.75 ± 4.02
-1	attn_score	51.84 ± 0.44	94.56 ± 0.82	51.70 ± 0.72	52.50 ± 3.66	53.11 ± 3.46
15	attn_score	51.83 ± 0.95	94.14 ± 0.59	51.64 ± 0.98	53.50 ± 2.72	54.80 ± 2.54
27	attn_score	51.80 ± 0.55	94.02 ± 0.59	51.77 ± 0.40	53.50 ± 2.65	53.22 ± 3.35
27	last_emb	51.80 ± 1.20	93.36 ± 1.44	50.11 ± 0.96	77.80 ± 4.83	77.78 ± 4.88
9	hidden_score	51.79 ± 0.65	94.56 ± 0.76	51.52 ± 0.53	53.80 ± 2.41	56.92 ± 1.89
21	logit_entropy_score	51.78 ± 1.69	94.48 ± 1.22	51.82 ± 1.50	50.50 ± 2.18	51.29 ± 2.65
27	window_logit_entropy_score	51.71 ± 1.39	91.86 ± 1.01	51.53 ± 1.17	51.60 ± 3.52	42.48 ± 11.09
29	logit_entropy_score	51.71 ± 0.99	94.68 ± 0.73	51.47 ± 0.84	52.40 ± 3.80	49.24 ± 3.84
9	attn_score	51.70 ± 0.80	94.00 ± 0.90	51.60 ± 0.86	52.80 ± 2.36	53.58 ± 3.11
7	hidden_score	51.64 ± 0.56	93.94 ± 1.09	51.21 ± 0.52	53.90 ± 2.25	57.11 ± 2.28
7	attn_score	51.61 ± 0.64	94.34 ± 0.79	51.59 ± 0.75	53.40 ± 1.98	54.35 ± 2.66
29	attn_score	51.60 ± 0.94	95.04 ± 0.23	51.73 ± 0.84	51.60 ± 2.43	50.84 ± 2.79
11	hidden_score	51.59 ± 0.64	94.28 ± 0.44	51.49 ± 0.59	54.30 ± 2.71	57.50 ± 2.15
-1	last_emb	51.48 ± 1.20	92.22 ± 0.54	49.79 ± 0.95	77.60 ± 2.61	77.07 ± 2.93
17	perplexity_score	51.47 ± 0.60	94.16 ± 0.65	51.18 ± 0.51	51.40 ± 3.47	48.32 ± 4.38
13	hidden_score	51.34 ± 0.52	94.44 ± 0.51	51.22 ± 0.50	53.00 ± 2.26	56.25 ± 1.56
3	hidden_score	51.25 ± 0.71	94.74 ± 0.99	50.61 ± 0.62	52.10 ± 2.70	55.28 ± 2.28
5	max_emb	51.23 ± 0.47	94.08 ± 0.37	51.08 ± 0.86	52.30 ± 3.44	51.59 ± 3.07
15	perplexity_score	51.15 ± 0.46	94.16 ± 1.20	51.03 ± 0.47	51.20 ± 4.25	47.82 ± 4.28
5	hidden_score	51.15 ± 0.64	94.72 ± 1.03	50.72 ± 0.51	52.70 ± 2.56	55.97 ± 2.83
29	window_logit_entropy_score	51.11 ± 1.40	92.28 ± 1.25	51.05 ± 1.07	49.10 ± 1.92	41.19 ± 9.23
15	hidden_score	51.10 ± 0.38	94.34 ± 0.62	50.97 ± 0.53	51.80 ± 3.11	54.63 ± 2.32
3	perplexity_score	51.08 ± 0.34	94.36 ± 0.74	51.00 ± 0.44	52.30 ± 4.15	49.40 ± 3.76
-1	avg_emb	51.07 ± 0.52	94.84 ± 1.19	51.02 ± 0.41	60.10 ± 1.02	59.66 ± 0.61
1	perplexity_score	51.07 ± 0.35	94.38 ± 0.76	51.00 ± 0.45	52.00 ± 3.95	49.12 ± 4.07
13	perplexity_score	51.07 ± 0.40	94.22 ± 0.87	50.99 ± 0.46	51.10 ± 3.75	39.38 ± 22.38
9	perplexity_score	51.06 ± 0.36	94.40 ± 0.75	51.00 ± 0.47	51.90 ± 3.88	49.03 ± 4.15
15	window_logit_entropy_score	51.06 ± 0.39	94.32 ± 1.20	50.76 ± 0.58	49.70 ± 2.22	48.68 ± 2.27
5	perplexity_score	51.05 ± 0.34	94.34 ± 0.82	50.98 ± 0.47	52.10 ± 4.20	49.15 ± 4.34
1	hidden_score	51.05 ± 0.56	94.84 ± 0.93	50.54 ± 0.45	51.60 ± 3.65	44.36 ± 25.01
11	perplexity_score	51.03 ± 0.37	94.16 ± 0.76	50.98 ± 0.42	52.10 ± 4.02	49.47 ± 4.02
7	perplexity_score	51.02 ± 0.37	94.46 ± 1.06	50.96 ± 0.50	51.50 ± 4.56	39.73 ± 22.67
1	attn_score	51.02 ± 0.42	94.18 ± 1.33	50.94 ± 0.64	52.30 ± 3.58	53.90 ± 3.48
23	attn_score	50.96 ± 0.73	94.54 ± 0.69	51.09 ± 0.85	52.50 ± 3.30	52.58 ± 4.01
-1	max_emb	50.82 ± 0.69	94.64 ± 0.77	50.55 ± 0.66	59.20 ± 2.31	57.58 ± 2.40
17	hidden_score	50.80 ± 0.38	94.26 ± 0.45	50.70 ± 0.55	48.50 ± 3.41	39.84 ± 22.77
17	logit_entropy_score	50.74 ± 1.25	95.66 ± 0.88	51.19 ± 0.56	50.80 ± 2.08	50.69 ± 3.54
5	window_logit_entropy_score	50.69 ± 0.72	97.34 ± 0.23	50.74 ± 0.61	50.40 ± 3.11	47.43 ± 4.08
21	attn_score	50.66 ± 1.11	94.86 ± 0.98	50.92 ± 1.11	48.90 ± 2.72	49.30 ± 4.26
19	hidden_score	50.62 ± 0.40	94.40 ± 0.69	50.54 ± 0.58	48.10 ± 3.60	39.35 ± 22.54
3	attn_score	50.62 ± 0.46	94.20 ± 0.88	50.50 ± 0.86	50.50 ± 2.24	51.20 ± 4.11

Table 2 — continued

layer	aggregation	auroc ↑ (OOD)	fpr95 ↓ (OOD)	auprc ↑ (OOD)	acc ↑ (probe)	f1 ↑ (probe)
19	attn_score	50.53 ± 1.42	94.60 ± 0.79	50.72 ± 1.00	53.50 ± 5.16	54.13 ± 5.46
17	attn_score	50.44 ± 0.58	94.90 ± 1.15	50.81 ± 0.59	51.30 ± 3.58	50.43 ± 5.02
3	window_logit_entropy_score	50.43 ± 0.26	100.00	50.77 ± 0.36	49.30 ± 1.60	38.37 ± 12.04
29	hidden_score	50.40 ± 0.58	94.34 ± 0.38	50.08 ± 0.48	49.70 ± 1.48	40.52 ± 22.86
11	last_emb	50.37 ± 1.64	93.48 ± 1.66	49.26 ± 1.19	72.60 ± 2.04	72.46 ± 2.40
29	last_emb	50.36 ± 1.16	92.94 ± 0.79	49.02 ± 0.87	77.80 ± 3.44	77.58 ± 3.70
21	hidden_score	50.35 ± 0.37	94.38 ± 0.93	50.27 ± 0.49	47.50 ± 2.15	46.67 ± 4.97
-1	hidden_score	50.33 ± 1.14	94.66 ± 0.79	50.29 ± 1.17	50.00 ± 3.91	50.50 ± 6.38
19	logit_entropy_score	50.32 ± 1.23	94.96 ± 0.64	50.72 ± 1.06	51.40 ± 2.36	52.22 ± 2.94
13	window_logit_entropy_score	50.32 ± 1.10	93.54 ± 0.43	50.48 ± 1.14	51.80 ± 0.27	41.39 ± 20.45
23	hidden_score	50.30 ± 0.31	94.52 ± 0.93	50.17 ± 0.44	48.20 ± 2.97	47.09 ± 4.74
25	hidden_score	50.26 ± 0.38	94.54 ± 1.04	50.12 ± 0.41	48.40 ± 2.63	47.86 ± 5.20
27	hidden_score	50.21 ± 0.43	94.22 ± 0.77	50.01 ± 0.42	48.40 ± 1.92	48.62 ± 4.45
5	attn_score	50.18 ± 0.74	94.68 ± 0.33	50.49 ± 0.81	48.10 ± 4.70	47.44 ± 5.56
29	avg_emb	50.16 ± 0.60	94.56 ± 0.71	50.52 ± 0.62	59.40 ± 1.85	58.48 ± 1.93
11	feat_var_emb	50.08 ± 0.10	100.00	50.08 ± 0.10	65.60 ± 1.78	65.43 ± 2.04
25	window_logit_entropy_score	50.04 ± 1.44	94.68 ± 0.94	50.29 ± 1.11	50.60 ± 2.77	33.24 ± 19.25
9	feat_var_emb	50.03 ± 0.03	100.00	50.04 ± 0.03	57.00 ± 1.58	56.14 ± 2.78
13	feat_var_emb	50.02 ± 0.07	100.00	50.12 ± 0.09	69.00 ± 3.59	68.51 ± 4.61
1	feat_var_emb	50.00	100.00	50.00	50.10 ± 3.96	49.30 ± 4.58
3	feat_var_emb	50.00	100.00	50.00	51.40 ± 2.82	49.66 ± 3.22
5	feat_var_emb	50.00	100.00	50.00	52.40 ± 1.64	49.44 ± 2.11
7	feat_var_emb	50.00	100.00	50.00	52.90 ± 2.04	52.44 ± 2.38
17	window_logit_entropy_score	49.98 ± 1.30	95.40 ± 0.57	50.68 ± 1.85	50.80 ± 1.48	35.31 ± 26.70
11	window_logit_entropy_score	49.96 ± 0.78	95.28 ± 0.37	50.71 ± 0.70	48.70 ± 4.32	48.01 ± 5.41
-1	feat_var_emb	49.95 ± 0.47	94.94 ± 1.05	49.76 ± 0.43	60.10 ± 2.58	59.50 ± 2.07
15	feat_var_emb	49.92 ± 0.58	100.00	50.13 ± 0.25	66.80 ± 3.56	66.30 ± 4.43
27	max_emb	49.90 ± 0.74	94.86 ± 0.51	50.11 ± 0.49	52.00 ± 3.06	51.43 ± 2.74
27	avg_emb	49.88 ± 0.43	94.48 ± 0.74	50.31 ± 0.45	59.10 ± 3.03	58.28 ± 3.42
25	max_emb	49.87 ± 0.71	95.04 ± 0.19	50.04 ± 0.52	51.10 ± 2.75	51.10 ± 3.51
29	max_emb	49.85 ± 0.91	94.72 ± 0.31	50.04 ± 0.60	51.10 ± 5.02	50.96 ± 4.91
25	avg_emb	49.82 ± 0.42	94.40 ± 0.58	50.17 ± 0.36	58.50 ± 2.06	57.86 ± 2.08
-1	perplexity_score	49.80 ± 2.02	94.72 ± 1.02	50.42 ± 1.94	50.60 ± 3.63	50.99 ± 3.11
3	last_emb	49.73 ± 1.12	94.06 ± 0.92	49.09 ± 0.84	63.30 ± 2.64	62.88 ± 2.82
23	max_emb	49.72 ± 0.59	94.66 ± 0.80	49.92 ± 0.67	49.20 ± 3.72	48.87 ± 3.93
23	window_logit_entropy_score	49.66 ± 1.43	95.34 ± 1.23	50.30 ± 1.19	51.30 ± 2.80	45.91 ± 15.72
3	max_emb	49.62 ± 0.67	96.24 ± 1.37	49.75 ± 0.75	48.50 ± 1.84	48.14 ± 1.84
23	avg_emb	49.60 ± 0.50	94.84 ± 0.33	50.00 ± 0.41	58.20 ± 0.57	56.43 ± 1.38
29	feat_var_emb	49.44 ± 0.74	94.98 ± 0.34	49.62 ± 0.48	57.50 ± 3.82	56.96 ± 4.54
19	max_emb	49.44 ± 0.65	94.64 ± 0.54	49.36 ± 0.81	58.30 ± 3.63	58.61 ± 3.00
21	max_emb	49.41 ± 0.77	94.86 ± 0.64	49.57 ± 0.82	53.10 ± 1.98	52.53 ± 2.84
7	last_emb	49.39 ± 1.29	93.98 ± 0.68	48.24 ± 0.96	64.60 ± 3.60	64.65 ± 3.82
17	feat_var_emb	49.34 ± 0.27	100.00	49.91 ± 0.20	60.70 ± 3.25	60.39 ± 3.42
21	window_logit_entropy_score	49.33 ± 1.64	100.00	49.81 ± 1.18	49.20 ± 4.48	39.06 ± 24.15
21	avg_emb	49.28 ± 0.58	94.90 ± 0.46	49.65 ± 0.31	59.80 ± 0.67	58.35 ± 1.66
23	feat_var_emb	49.22 ± 0.31	94.56 ± 0.78	49.48 ± 0.23	56.60 ± 4.93	55.90 ± 4.81
7	window_logit_entropy_score	49.20 ± 0.96	95.50 ± 0.78	50.02 ± 0.89	51.90 ± 4.14	51.98 ± 3.40
9	max_emb	49.19 ± 0.59	94.90 ± 0.78	49.70 ± 0.17	55.00 ± 3.08	54.65 ± 2.66
1	window_logit_entropy_score	49.19 ± 1.32	95.56 ± 1.40	49.53 ± 1.21	49.30 ± 4.56	49.20 ± 3.73
25	feat_var_emb	49.18 ± 0.26	94.86 ± 0.93	49.49 ± 0.22	55.10 ± 3.32	54.13 ± 3.23
-1	window_logit_entropy_score	49.09 ± 1.99	94.64 ± 0.43	49.90 ± 1.74	51.70 ± 3.33	40.09 ± 16.00
-1	logit_entropy_score	49.09 ± 1.16	94.66 ± 0.65	49.61 ± 0.63	48.80 ± 2.17	26.17 ± 21.48
19	feat_var_emb	49.05 ± 0.78	100.00	49.44 ± 0.34	61.50 ± 1.17	60.31 ± 3.08
27	feat_var_emb	49.05 ± 0.74	95.14 ± 0.60	49.31 ± 0.49	57.10 ± 3.09	56.29 ± 3.74
21	feat_var_emb	49.04 ± 0.33	95.54 ± 0.95	49.39 ± 0.19	59.10 ± 1.67	58.35 ± 2.53
19	window_logit_entropy_score	49.04 ± 0.58	95.22 ± 0.88	49.35 ± 0.88	52.50 ± 3.37	56.01 ± 9.11
19	avg_emb	49.02 ± 0.76	95.02 ± 0.38	49.43 ± 0.45	62.20 ± 1.82	61.67 ± 1.66
5	logit_entropy_score	49.00 ± 0.34	95.20 ± 0.41	49.28 ± 0.29	52.40 ± 3.86	52.76 ± 3.54
7	logit_entropy_score	48.99 ± 0.33	95.20 ± 0.35	49.28 ± 0.30	52.40 ± 3.81	52.84 ± 3.68
3	logit_entropy_score	48.98 ± 0.35	95.24 ± 0.37	49.26 ± 0.29	52.20 ± 3.65	52.47 ± 3.21
1	logit_entropy_score	48.97 ± 0.34	95.16 ± 0.34	49.26 ± 0.29	52.20 ± 3.65	52.47 ± 3.21
9	window_logit_entropy_score	48.94 ± 1.62	95.38 ± 1.01	49.35 ± 1.81	54.00 ± 2.42	54.27 ± 4.64
11	logit_entropy_score	48.92 ± 0.39	95.14 ± 0.42	49.22 ± 0.35	52.40 ± 3.66	52.39 ± 3.36
15	logit_entropy_score	48.91 ± 0.56	94.56 ± 0.89	49.46 ± 0.42	52.90 ± 2.36	53.49 ± 2.22
9	logit_entropy_score	48.91 ± 0.39	95.18 ± 0.34	49.22 ± 0.35	52.50 ± 3.37	52.64 ± 2.96
3	avg_emb	48.87 ± 0.43	95.62 ± 0.43	49.17 ± 0.35	53.60 ± 1.39	52.81 ± 2.31
1	avg_emb	48.86 ± 0.45	95.24 ± 0.62	49.15 ± 0.33	53.30 ± 2.51	52.59 ± 1.72
13	logit_entropy_score	48.83 ± 0.40	94.80 ± 0.66	49.26 ± 0.35	52.10 ± 3.58	52.34 ± 3.00

Table 2 — continued

layer	aggregation	auroc ↑ (OOD)	fpr95 ↓ (OOD)	auprc ↑ (OOD)	acc ↑ (probe)	f1 ↑ (probe)
17	max_emb	48.82 ± 0.81	94.72 ± 0.78	49.15 ± 0.87	57.10 ± 4.16	56.86 ± 4.03
7	avg_emb	48.77 ± 0.37	95.12 ± 0.41	49.22 ± 0.27	56.50 ± 2.37	56.19 ± 2.50
11	max_emb	48.77 ± 0.90	95.54 ± 0.96	49.01 ± 0.47	60.90 ± 4.11	60.80 ± 3.57
17	avg_emb	48.75 ± 0.68	95.18 ± 0.75	49.28 ± 0.56	62.40 ± 1.78	61.61 ± 2.20
9	avg_emb	48.72 ± 0.39	95.18 ± 0.54	49.18 ± 0.29	64.10 ± 4.92	63.44 ± 5.37
7	max_emb	48.71 ± 1.03	100.00	49.31 ± 0.52	53.50 ± 4.27	52.80 ± 3.93
11	avg_emb	48.66 ± 0.56	95.26 ± 0.40	49.13 ± 0.41	64.30 ± 1.15	63.67 ± 1.08
5	avg_emb	48.58 ± 0.33	95.54 ± 0.48	49.08 ± 0.27	54.20 ± 1.64	52.55 ± 2.02
13	avg_emb	48.52 ± 0.54	95.26 ± 0.47	49.06 ± 0.40	65.90 ± 1.14	65.52 ± 1.09
15	avg_emb	48.51 ± 0.46	95.34 ± 0.50	49.09 ± 0.49	65.90 ± 1.14	65.36 ± 1.77
1	last_emb	48.49 ± 0.96	95.18 ± 0.82	48.55 ± 0.85	55.80 ± 2.14	55.52 ± 2.17
15	max_emb	48.35 ± 0.46	94.86 ± 1.02	48.72 ± 0.51	61.50 ± 2.15	60.18 ± 4.13
5	last_emb	48.22 ± 1.08	94.84 ± 0.43	47.81 ± 0.70	65.90 ± 3.09	65.27 ± 2.22
9	last_emb	48.22 ± 1.42	94.56 ± 1.15	47.66 ± 1.00	67.00 ± 2.69	66.82 ± 2.97
13	max_emb	47.80 ± 0.69	95.02 ± 0.61	48.47 ± 0.45	62.70 ± 4.16	62.23 ± 5.03
1	max_emb	47.58 ± 0.50	93.64 ± 0.83	48.82 ± 0.28	48.30 ± 3.51	48.44 ± 3.57

Table 3: OOD Detection Results on Generated-Answer Representations ranked by AUROC. Results averaged over 5 runs with distinct seeds and reported as mean ± std. For *embedding-based* descriptors, OOD scores are computed with DeepKNN (k=5); for *scalar* descriptors, raw scalar values are used directly. Reported metrics: **AUROC**, **FPR@95**, **AUPRC**, plus probe **accuracy** and **F1**.

layer	aggregation	auroc ↑ (OOD)	fpr95 ↓ (OOD)	auprc ↑ (OOD)	acc ↑ (probe)	f1 ↑ (probe)
-1	logit_entropy_score	61.74 ± 0.55	87.42 ± 2.01	59.01 ± 0.65	58.60 ± 4.07	54.16 ± 3.33
-1	window_logit_entropy_score	61.20 ± 0.48	88.10 ± 2.76	59.26 ± 1.05	56.70 ± 3.60	55.93 ± 3.61
-1	perplexity_score	60.56 ± 0.75	87.68 ± 2.51	58.10 ± 0.94	57.40 ± 2.92	48.24 ± 3.84
27	first_gen_emb	58.86 ± 0.96	93.58 ± 1.16	59.71 ± 1.24	77.90 ± 2.36	77.82 ± 2.33
17	first_gen_emb	58.71 ± 1.14	91.02 ± 0.73	58.75 ± 1.12	82.50 ± 3.18	82.32 ± 3.33
13	first_gen_emb	58.68 ± 1.57	89.20 ± 1.21	55.75 ± 1.23	81.10 ± 3.19	81.10 ± 3.33
27	window_logit_entropy_score	58.67 ± 0.73	88.94 ± 1.38	56.96 ± 0.46	54.90 ± 2.75	58.32 ± 2.89
15	first_gen_emb	58.64 ± 1.61	88.82 ± 1.40	55.63 ± 1.18	81.30 ± 1.99	81.30 ± 2.46
25	first_gen_emb	58.62 ± 1.06	93.72 ± 1.20	59.58 ± 1.41	77.70 ± 2.68	77.26 ± 3.46
29	first_gen_emb	58.46 ± 0.89	93.70 ± 1.16	59.23 ± 1.26	77.70 ± 1.60	77.32 ± 2.06
19	first_gen_emb	57.57 ± 0.86	92.60 ± 0.70	59.20 ± 0.71	79.30 ± 3.40	79.27 ± 3.13
23	first_gen_emb	57.53 ± 1.14	93.40 ± 1.38	58.44 ± 1.58	78.70 ± 1.10	78.26 ± 1.53
21	first_gen_emb	57.48 ± 1.00	93.32 ± 0.70	58.59 ± 1.46	76.20 ± 3.21	75.84 ± 2.96
-1	first_gen_emb	57.47 ± 0.87	93.68 ± 1.02	57.74 ± 1.29	77.80 ± 2.17	77.54 ± 2.49
27	logit_entropy_score	56.73 ± 0.91	92.66 ± 1.72	55.98 ± 0.88	56.90 ± 4.20	55.19 ± 3.48
11	first_gen_emb	56.68 ± 1.62	91.84 ± 1.19	55.31 ± 1.22	76.30 ± 2.17	76.26 ± 2.33
29	logit_entropy_score	56.16 ± 0.81	93.20 ± 1.15	54.20 ± 0.68	57.50 ± 3.39	55.63 ± 2.31
29	perplexity_score	55.87 ± 0.81	92.44 ± 1.09	54.91 ± 0.68	54.00 ± 1.97	41.82 ± 3.98
9	first_gen_emb	55.68 ± 1.82	93.72 ± 1.42	54.77 ± 1.26	73.00 ± 1.37	72.94 ± 1.49
27	perplexity_score	55.60 ± 0.85	92.70 ± 1.36	54.82 ± 0.53	54.90 ± 3.21	44.10 ± 4.18
25	logit_entropy_score	55.41 ± 0.79	92.92 ± 1.03	54.63 ± 0.69	55.80 ± 1.20	55.59 ± 1.83
9	attn_score	54.74 ± 1.25	93.76 ± 1.24	53.98 ± 1.07	56.40 ± 2.01	58.13 ± 1.83
15	attn_score	54.64 ± 1.03	91.80 ± 1.26	52.71 ± 0.94	55.10 ± 3.85	56.40 ± 4.01
25	perplexity_score	54.23 ± 1.03	94.58 ± 0.80	53.34 ± 0.84	52.10 ± 4.83	36.86 ± 6.39
19	window_logit_entropy_score	53.83 ± 0.63	92.60 ± 0.39	53.51 ± 1.34	54.60 ± 1.71	56.92 ± 1.30
7	first_gen_emb	53.57 ± 1.36	94.04 ± 1.11	52.55 ± 0.73	66.70 ± 5.01	66.29 ± 4.77
1	window_logit_entropy_score	53.51 ± 0.62	94.38 ± 0.85	53.46 ± 0.66	51.40 ± 2.04	59.38 ± 2.22
5	first_gen_emb	53.48 ± 0.68	93.96 ± 0.71	53.90 ± 0.65	65.10 ± 3.80	65.01 ± 3.46
13	window_logit_entropy_score	53.45 ± 1.76	91.36 ± 0.62	52.36 ± 1.85	53.10 ± 3.68	54.41 ± 5.53
27	attn_score	53.38 ± 1.15	94.64 ± 0.59	52.63 ± 0.91	55.30 ± 1.89	57.51 ± 2.37
7	attn_score	53.37 ± 1.22	91.66 ± 1.07	52.10 ± 1.17	55.30 ± 4.93	58.70 ± 4.51
3	first_gen_emb	53.11 ± 0.55	96.22 ± 0.36	54.28 ± 0.90	59.90 ± 0.82	59.52 ± 2.38
11	attn_score	53.09 ± 0.93	92.66 ± 0.77	52.20 ± 0.97	56.00 ± 2.09	57.51 ± 2.21
23	attn_score	53.08 ± 1.34	93.22 ± 0.92	52.41 ± 1.19	54.30 ± 1.60	54.67 ± 2.33
3	attn_score	52.66 ± 0.91	93.72 ± 1.05	51.53 ± 0.82	51.10 ± 2.38	51.60 ± 3.27
25	attn_score	52.38 ± 1.14	93.78 ± 0.92	51.78 ± 1.08	51.80 ± 1.72	52.79 ± 2.63
5	attn_score	52.29 ± 1.06	94.60 ± 0.97	51.88 ± 1.01	52.10 ± 4.87	53.30 ± 7.46
-1	attn_score	52.26 ± 1.21	95.62 ± 0.69	52.16 ± 1.35	53.50 ± 2.12	57.29 ± 1.98
13	attn_score	52.17 ± 0.85	95.12 ± 0.51	51.11 ± 0.92	53.20 ± 2.46	55.72 ± 3.07
1	logit_entropy_score	51.89 ± 0.42	93.96 ± 0.86	50.21 ± 0.29	53.90 ± 4.10	59.32 ± 3.24
21	perplexity_score	51.88 ± 0.80	94.76 ± 0.91	52.24 ± 0.87	49.20 ± 3.55	38.02 ± 4.84
15	window_logit_entropy_score	51.87 ± 1.38	93.48 ± 0.76	51.83 ± 1.48	51.10 ± 4.62	53.21 ± 4.22

Table 3 — continued

layer	aggregation	auroc ↑ (OOD)	fpr95 ↓ (OOD)	auprc ↑ (OOD)	acc ↑ (probe)	f1 ↑ (probe)
29	attn_score	51.59 ± 1.37	95.64 ± 0.56	50.92 ± 1.43	50.70 ± 2.82	51.97 ± 5.66
19	perplexity_score	51.41 ± 0.94	93.50 ± 1.13	51.24 ± 0.89	52.00 ± 3.48	46.72 ± 4.56
21	attn_score	51.41 ± 0.97	94.58 ± 1.03	51.10 ± 0.90	51.90 ± 2.07	54.12 ± 2.39
17	last_emb	50.98 ± 0.61	94.70 ± 0.54	50.49 ± 0.42	76.70 ± 2.56	76.38 ± 2.70
29	last_emb	50.96 ± 0.45	93.88 ± 1.14	50.67 ± 0.28	74.50 ± 3.26	74.04 ± 2.86
25	last_emb	50.88 ± 0.47	93.92 ± 0.97	50.61 ± 0.27	75.70 ± 3.27	75.43 ± 3.69
27	last_emb	50.85 ± 0.42	93.74 ± 1.04	50.65 ± 0.23	75.00 ± 2.92	74.49 ± 2.20
13	last_emb	50.85 ± 0.77	94.76 ± 0.55	50.20 ± 0.51	73.90 ± 2.07	73.41 ± 1.61
21	last_emb	50.81 ± 0.43	94.10 ± 0.79	50.51 ± 0.30	77.40 ± 3.93	77.19 ± 4.11
23	last_emb	50.81 ± 0.47	94.14 ± 0.72	50.51 ± 0.27	76.90 ± 3.42	76.51 ± 3.66
19	attn_score	50.76 ± 1.19	93.96 ± 0.71	50.51 ± 1.29	51.50 ± 4.00	53.20 ± 4.94
19	last_emb	50.76 ± 0.48	94.02 ± 0.69	50.48 ± 0.32	76.90 ± 2.84	76.37 ± 3.26
23	perplexity_score	50.74 ± 0.62	95.90 ± 0.94	52.21 ± 0.43	50.90 ± 4.32	35.94 ± 6.17
15	last_emb	50.73 ± 0.64	94.76 ± 0.73	50.28 ± 0.44	75.00 ± 1.94	74.84 ± 2.07
1	attn_score	50.52 ± 0.90	94.78 ± 0.82	50.23 ± 0.69	53.60 ± 3.17	51.83 ± 3.45
17	attn_score	50.48 ± 1.15	93.66 ± 0.87	49.66 ± 1.01	50.40 ± 2.10	50.35 ± 4.30
11	last_emb	50.46 ± 0.62	94.78 ± 0.56	49.94 ± 0.42	70.30 ± 3.15	70.02 ± 3.14
7	window_logit_entropy_score	50.42 ± 1.50	94.18 ± 1.17	50.49 ± 1.31	51.10 ± 2.75	54.87 ± 6.41
5	hidden_score	50.37 ± 0.87	93.70 ± 0.92	49.69 ± 0.91	50.40 ± 3.15	50.23 ± 9.08
7	last_emb	50.32 ± 0.47	94.30 ± 0.46	49.68 ± 0.44	59.80 ± 0.76	58.43 ± 1.78
-1	last_emb	50.32 ± 0.50	91.30 ± 1.23	49.45 ± 0.39	68.80 ± 5.61	68.81 ± 5.34
5	window_logit_entropy_score	50.26 ± 0.93	95.64 ± 1.14	50.05 ± 1.11	49.40 ± 1.52	44.15 ± 5.88
21	logit_entropy_score	50.17 ± 1.10	93.40 ± 1.21	50.17 ± 0.43	51.20 ± 5.46	51.25 ± 5.72
3	hidden_score	50.05 ± 0.68	92.94 ± 0.80	50.01 ± 0.88	48.70 ± 3.38	50.73 ± 5.01
5	last_emb	50.04 ± 0.25	94.88 ± 0.65	49.60 ± 0.26	57.70 ± 2.41	56.58 ± 2.98
23	logit_entropy_score	50.01 ± 0.91	94.88 ± 1.27	50.83 ± 0.56	52.30 ± 4.37	52.97 ± 4.54
9	hidden_score	49.93 ± 1.02	93.56 ± 1.17	49.48 ± 0.98	51.40 ± 4.68	50.77 ± 8.42
23	hidden_score	49.92 ± 1.13	93.44 ± 0.98	49.23 ± 1.09	51.60 ± 4.26	53.51 ± 2.50
11	hidden_score	49.83 ± 0.61	92.40 ± 1.10	49.43 ± 0.66	51.90 ± 4.92	50.62 ± 8.45
7	hidden_score	49.81 ± 1.10	94.58 ± 1.19	49.36 ± 1.00	50.30 ± 4.19	48.92 ± 8.76
15	hidden_score	49.73 ± 0.80	92.40 ± 0.69	49.05 ± 0.66	51.50 ± 3.20	50.26 ± 7.10
11	window_logit_entropy_score	49.65 ± 0.88	93.64 ± 0.79	49.06 ± 0.69	48.70 ± 1.79	40.39 ± 23.12
9	last_emb	49.61 ± 0.38	94.84 ± 0.71	49.40 ± 0.38	64.50 ± 4.49	63.57 ± 3.93
27	avg_emb	49.60 ± 0.96	96.44 ± 1.21	51.84 ± 0.95	74.40 ± 1.92	73.52 ± 2.03
27	hidden_score	49.60 ± 1.22	93.58 ± 0.89	48.81 ± 1.01	50.70 ± 4.72	50.84 ± 7.13
27	feat_var_emb	49.56 ± 1.23	96.30 ± 1.04	51.09 ± 1.19	66.60 ± 3.68	66.07 ± 4.18
25	avg_emb	49.52 ± 0.92	96.60 ± 0.88	51.82 ± 0.96	72.90 ± 1.43	71.95 ± 1.79
17	hidden_score	49.47 ± 0.66	94.46 ± 0.67	49.13 ± 0.87	52.80 ± 3.77	51.39 ± 6.70
29	avg_emb	49.41 ± 0.86	96.30 ± 1.22	51.60 ± 0.92	73.80 ± 2.17	73.41 ± 2.87
25	hidden_score	49.40 ± 1.23	93.90 ± 0.69	48.71 ± 1.09	51.40 ± 5.41	51.62 ± 7.11
29	feat_var_emb	49.37 ± 1.12	95.90 ± 0.72	50.21 ± 1.05	66.40 ± 1.24	66.35 ± 1.82
23	feat_var_emb	49.34 ± 1.32	96.54 ± 0.95	50.59 ± 0.79	66.60 ± 2.33	66.41 ± 1.85
19	logit_entropy_score	49.31 ± 1.70	94.24 ± 0.74	50.29 ± 1.38	48.80 ± 6.29	50.03 ± 5.80
19	hidden_score	49.31 ± 1.02	94.22 ± 0.22	48.69 ± 1.05	51.60 ± 2.82	50.55 ± 5.17
19	feat_var_emb	49.27 ± 1.29	96.42 ± 0.95	50.17 ± 0.60	73.50 ± 2.67	73.54 ± 3.07
21	hidden_score	49.27 ± 0.96	93.70 ± 0.69	48.60 ± 0.93	51.50 ± 3.84	52.03 ± 5.19
17	avg_emb	49.17 ± 0.85	96.86 ± 0.58	51.54 ± 0.67	76.20 ± 3.29	75.98 ± 3.07
1	hidden_score	49.15 ± 0.84	93.48 ± 0.76	48.53 ± 0.98	49.50 ± 3.55	56.33 ± 3.28
29	hidden_score	49.13 ± 1.03	94.28 ± 0.74	48.57 ± 0.87	49.90 ± 5.03	49.65 ± 7.34
-1	hidden_score	49.12 ± 0.90	93.42 ± 1.19	49.09 ± 1.20	49.00 ± 2.72	37.09 ± 21.07
21	feat_var_emb	49.10 ± 1.36	96.78 ± 0.94	50.17 ± 0.70	69.60 ± 2.77	68.77 ± 2.99
5	feat_var_emb	49.07 ± 0.84	97.50 ± 0.64	49.52 ± 0.53	55.30 ± 3.70	56.72 ± 3.51
9	window_logit_entropy_score	49.07 ± 1.09	94.68 ± 0.72	49.17 ± 0.98	52.40 ± 2.97	50.97 ± 4.65
17	window_logit_entropy_score	49.05 ± 1.73	95.68 ± 0.27	49.83 ± 1.52	51.00 ± 4.03	50.33 ± 3.98
23	avg_emb	49.04 ± 1.01	96.78 ± 0.61	51.16 ± 0.97	72.30 ± 3.09	71.28 ± 3.11
15	feat_var_emb	49.01 ± 1.16	97.10 ± 0.78	50.16 ± 0.51	69.90 ± 1.82	70.02 ± 1.73
15	avg_emb	48.98 ± 0.84	96.82 ± 0.75	51.26 ± 0.61	76.50 ± 4.20	76.44 ± 4.02
19	avg_emb	48.95 ± 0.80	96.78 ± 0.61	51.34 ± 0.60	77.20 ± 3.07	76.78 ± 3.12
17	feat_var_emb	48.95 ± 1.17	96.88 ± 0.62	50.24 ± 0.51	70.50 ± 2.15	70.72 ± 1.44
1	first_gen_emb	48.95 ± 1.04	95.22 ± 0.69	49.05 ± 1.05	57.50 ± 3.82	56.50 ± 4.76
25	feat_var_emb	48.90 ± 1.21	96.36 ± 0.89	50.21 ± 0.74	64.50 ± 3.14	63.98 ± 2.82
9	feat_var_emb	48.83 ± 0.92	97.62 ± 0.46	50.06 ± 0.43	61.30 ± 4.22	61.86 ± 4.98
7	feat_var_emb	48.80 ± 1.30	97.50 ± 0.40	50.23 ± 0.77	58.00 ± 3.59	58.59 ± 4.20
17	perplexity_score	48.76 ± 1.24	94.24 ± 0.67	49.37 ± 1.14	50.40 ± 3.27	52.19 ± 5.81
21	avg_emb	48.72 ± 0.96	96.82 ± 0.63	50.80 ± 0.76	74.30 ± 4.21	73.62 ± 4.36
-1	avg_emb	48.71 ± 0.85	96.60 ± 1.15	50.11 ± 0.74	73.90 ± 3.85	73.49 ± 4.32
11	feat_var_emb	48.67 ± 1.13	97.52 ± 0.58	49.93 ± 0.63	64.00 ± 4.37	64.40 ± 3.68
13	feat_var_emb	48.60 ± 1.17	97.42 ± 0.73	49.86 ± 0.52	75.40 ± 2.22	75.53 ± 2.31

Table 3 — continued

layer	aggregation	auroc ↑ (OOD)	fpr95 ↓ (OOD)	auprc ↑ (OOD)	acc ↑ (probe)	f1 ↑ (probe)
13	avg_emb	48.43 ± 0.91	97.02 ± 0.72	51.01 ± 0.52	78.00 ± 3.74	77.61 ± 3.69
5	logit_entropy_score	48.40 ± 1.15	95.38 ± 0.57	47.99 ± 0.85	51.60 ± 2.56	46.48 ± 3.92
13	hidden_score	48.39 ± 0.60	93.68 ± 1.10	48.04 ± 0.57	47.70 ± 4.37	42.69 ± 5.09
23	window_logit_entropy_score	48.31 ± 1.79	95.42 ± 0.83	49.34 ± 1.24	52.30 ± 5.46	53.04 ± 5.86
3	window_logit_entropy_score	48.20 ± 1.49	94.34 ± 1.24	49.01 ± 1.09	52.10 ± 2.16	46.25 ± 7.19
21	window_logit_entropy_score	48.06 ± 0.97	95.90 ± 0.39	49.11 ± 0.89	52.30 ± 5.42	50.98 ± 5.54
9	logit_entropy_score	47.65 ± 1.19	94.16 ± 1.00	48.20 ± 1.19	52.30 ± 2.20	49.00 ± 8.40
29	max_emb	47.57 ± 1.31	94.98 ± 1.10	49.88 ± 0.79	71.40 ± 4.11	71.57 ± 4.09
27	max_emb	47.51 ± 1.39	94.94 ± 1.15	49.82 ± 0.92	71.40 ± 4.51	70.57 ± 4.67
5	perplexity_score	47.40 ± 0.38	94.78 ± 0.60	48.01 ± 0.54	54.60 ± 4.46	56.26 ± 3.73
3	feat_var_emb	47.26 ± 1.73	97.78 ± 0.51	48.57 ± 1.32	52.60 ± 3.56	53.05 ± 4.73
3	last_emb	47.24 ± 0.85	96.56 ± 0.65	48.15 ± 0.21	58.20 ± 2.25	58.46 ± 2.92
-1	max_emb	47.14 ± 1.28	94.86 ± 0.98	49.32 ± 0.87	70.50 ± 3.79	69.79 ± 4.01
25	window_logit_entropy_score	47.00 ± 1.54	96.22 ± 0.50	49.41 ± 1.09	51.70 ± 5.64	51.22 ± 4.95
1	feat_var_emb	46.93 ± 1.18	97.58 ± 0.63	49.03 ± 0.73	53.70 ± 5.16	54.06 ± 6.25
15	max_emb	46.89 ± 1.33	96.60 ± 0.83	49.01 ± 1.02	76.20 ± 1.89	76.26 ± 1.57
13	perplexity_score	46.77 ± 1.15	95.96 ± 0.55	48.08 ± 0.80	52.40 ± 4.48	54.92 ± 4.07
11	logit_entropy_score	46.71 ± 0.73	94.44 ± 1.25	47.79 ± 0.79	52.20 ± 3.23	45.96 ± 4.40
1	last_emb	46.62 ± 0.33	96.24 ± 1.04	47.81 ± 0.30	57.10 ± 1.67	56.26 ± 2.88
15	logit_entropy_score	46.61 ± 0.87	96.04 ± 0.64	49.29 ± 0.73	55.90 ± 3.86	47.06 ± 5.77
25	max_emb	46.61 ± 1.41	95.16 ± 0.99	49.00 ± 1.04	72.40 ± 4.39	71.63 ± 4.32
13	max_emb	46.53 ± 1.37	97.10 ± 0.68	48.64 ± 1.12	77.40 ± 3.49	77.31 ± 3.24
17	max_emb	46.36 ± 1.35	96.54 ± 0.77	48.65 ± 0.93	75.00 ± 3.50	74.43 ± 3.58
3	perplexity_score	46.33 ± 0.58	95.16 ± 0.99	47.43 ± 0.72	54.70 ± 5.37	56.07 ± 5.52
23	max_emb	46.31 ± 1.38	95.34 ± 1.03	48.75 ± 0.90	72.60 ± 4.56	71.47 ± 4.62
7	logit_entropy_score	46.26 ± 1.16	93.62 ± 1.27	47.02 ± 0.74	52.60 ± 3.54	46.96 ± 4.04
11	avg_emb	46.22 ± 0.82	97.32 ± 0.72	48.87 ± 0.30	70.80 ± 4.09	70.90 ± 3.64
-1	feat_var_emb	46.18 ± 1.09	95.20 ± 1.17	48.33 ± 0.66	66.80 ± 5.63	66.11 ± 5.78
1	perplexity_score	46.16 ± 0.81	95.04 ± 0.62	47.46 ± 0.77	53.20 ± 4.93	54.02 ± 4.78
13	logit_entropy_score	46.11 ± 1.08	96.54 ± 1.02	48.54 ± 0.98	55.30 ± 2.14	46.24 ± 5.04
19	max_emb	45.87 ± 1.33	95.70 ± 1.13	48.41 ± 0.86	75.30 ± 1.52	75.01 ± 1.71
15	perplexity_score	45.85 ± 0.79	96.06 ± 0.42	47.67 ± 0.70	52.20 ± 4.78	55.14 ± 4.46
21	max_emb	45.63 ± 1.27	95.76 ± 1.10	48.24 ± 0.75	72.90 ± 3.63	71.97 ± 3.76
7	perplexity_score	45.54 ± 0.35	95.88 ± 0.44	47.85 ± 0.76	54.70 ± 2.95	56.24 ± 2.36
9	avg_emb	45.52 ± 0.93	97.32 ± 0.61	47.87 ± 0.16	65.00 ± 5.01	65.16 ± 5.08
17	logit_entropy_score	45.40 ± 0.89	95.16 ± 0.72	47.31 ± 0.32	53.30 ± 3.23	46.89 ± 4.89
11	perplexity_score	45.32 ± 0.76	97.24 ± 0.33	47.07 ± 0.75	54.70 ± 4.84	56.85 ± 4.00
3	logit_entropy_score	45.23 ± 1.11	94.96 ± 0.83	46.53 ± 0.53	55.30 ± 2.77	49.96 ± 2.48
1	max_emb	45.15 ± 1.12	97.34 ± 0.71	47.52 ± 0.91	51.00 ± 6.84	50.51 ± 9.40
11	max_emb	45.02 ± 1.28	97.36 ± 0.62	47.16 ± 0.95	68.80 ± 3.55	69.58 ± 4.47
1	avg_emb	44.99 ± 1.49	97.04 ± 0.59	46.80 ± 1.11	54.50 ± 6.43	54.23 ± 7.87
7	avg_emb	44.85 ± 0.90	97.36 ± 0.48	47.30 ± 0.35	57.30 ± 1.92	56.94 ± 3.81
5	avg_emb	44.76 ± 1.10	97.34 ± 0.60	47.02 ± 0.75	56.70 ± 3.40	56.97 ± 4.24
3	max_emb	44.29 ± 1.21	97.44 ± 0.77	46.24 ± 0.72	54.80 ± 3.13	55.32 ± 4.27
9	perplexity_score	44.24 ± 0.99	97.74 ± 0.38	46.51 ± 0.80	55.70 ± 4.37	56.98 ± 3.53
3	avg_emb	44.15 ± 1.33	97.26 ± 0.61	46.31 ± 0.72	56.10 ± 3.47	55.84 ± 4.72
9	max_emb	43.64 ± 1.03	97.30 ± 0.56	45.60 ± 0.65	64.30 ± 4.78	64.86 ± 5.70
5	max_emb	43.52 ± 1.05	97.34 ± 0.60	45.62 ± 0.73	57.60 ± 3.76	57.92 ± 4.55
7	max_emb	43.52 ± 1.08	97.34 ± 0.55	45.47 ± 0.64	56.70 ± 4.16	56.50 ± 5.58
29	window_logit_entropy_score	42.67 ± 0.61	97.16 ± 0.78	45.77 ± 0.46	54.00 ± 1.87	51.14 ± 1.60

Table 4: OOD Detection Results on concatenated Prompt + Generated-Answer Representations ranked by AUROC. Results averaged over 5 runs with distinct seeds and reported as mean ± std. For *embedding-based* descriptors, OOD scores are computed with DeepKNN (k=5); for *scalar* descriptors, raw scalar values are used directly. Reported metrics: **AUROC**, **FPR@95**, **AUPRC**, plus probe **accuracy** and **F1**.

layer	aggregation	auroc ↑ (OOD)	fpr95 ↓ (OOD)	auprc ↑ (OOD)	acc ↑ (probe)	f1 ↑ (probe)
27	first_gen_emb	56.92 ± 1.35	93.24 ± 1.20	55.67 ± 1.50	51.40 ± 2.63	51.30 ± 2.56
25	first_gen_emb	56.87 ± 1.34	93.24 ± 1.32	55.74 ± 1.55	50.70 ± 3.95	49.89 ± 4.67
29	first_gen_emb	56.84 ± 1.40	93.10 ± 1.34	55.76 ± 1.49	50.20 ± 4.88	50.88 ± 4.85
11	first_gen_emb	56.68 ± 1.04	92.48 ± 0.84	55.28 ± 0.81	51.30 ± 3.55	49.86 ± 3.37
23	first_gen_emb	56.63 ± 1.24	93.50 ± 1.43	55.47 ± 1.49	49.60 ± 2.90	49.09 ± 2.71
13	first_gen_emb	56.00 ± 0.94	92.52 ± 1.09	54.50 ± 0.31	54.20 ± 3.96	54.06 ± 4.35
9	first_gen_emb	55.96 ± 1.14	92.60 ± 0.79	54.99 ± 1.38	50.30 ± 2.51	47.98 ± 3.76
21	first_gen_emb	55.86 ± 1.24	93.46 ± 1.27	54.77 ± 1.17	51.10 ± 4.13	50.93 ± 4.23

Table 4 — continued

layer	aggregation	auroc ↑ (OOD)	fpr95 ↓ (OOD)	auprc ↑ (OOD)	acc ↑ (probe)	f1 ↑ (probe)
15	first_gen_emb	55.73 ± 0.88	92.48 ± 1.14	54.35 ± 0.32	53.60 ± 0.96	52.58 ± 2.14
17	first_gen_emb	55.39 ± 0.76	93.08 ± 1.70	54.05 ± 0.35	51.80 ± 3.40	51.14 ± 4.20
7	first_gen_emb	55.38 ± 1.15	92.16 ± 1.00	54.50 ± 1.66	51.20 ± 2.61	51.14 ± 3.09
19	first_gen_emb	55.33 ± 1.06	93.30 ± 1.21	54.09 ± 0.64	50.40 ± 3.66	48.86 ± 3.66
5	first_gen_emb	54.33 ± 1.26	91.76 ± 0.60	53.15 ± 1.73	53.70 ± 2.80	52.58 ± 4.38
3	first_gen_emb	53.40 ± 1.56	92.06 ± 0.51	52.07 ± 2.17	51.30 ± 4.93	51.20 ± 5.31
1	first_gen_emb	53.31 ± 1.73	91.78 ± 1.00	52.46 ± 1.64	51.10 ± 2.68	49.26 ± 3.72
-1	avg_emb	53.01 ± 0.96	93.74 ± 1.36	52.93 ± 1.05	52.40 ± 3.70	51.02 ± 5.13
-1	first_gen_emb	52.95 ± 1.33	93.10 ± 1.44	52.50 ± 0.79	51.70 ± 3.40	51.61 ± 3.71
27	perplexity_score	52.92 ± 0.71	94.32 ± 1.06	52.43 ± 0.71	53.30 ± 4.34	47.21 ± 4.82
29	perplexity_score	52.67 ± 0.50	94.10 ± 0.60	52.12 ± 0.47	52.50 ± 4.51	46.16 ± 5.31
25	perplexity_score	52.59 ± 0.86	93.90 ± 0.69	52.32 ± 0.87	52.60 ± 3.63	47.36 ± 4.85
23	perplexity_score	52.46 ± 1.00	93.86 ± 1.12	52.15 ± 0.95	53.30 ± 3.67	49.08 ± 5.16
15	last_emb	52.33 ± 1.08	93.82 ± 0.92	53.00 ± 1.32	55.70 ± 0.97	54.99 ± 2.00
11	last_emb	52.23 ± 1.24	94.62 ± 0.89	52.98 ± 1.41	53.60 ± 3.03	53.50 ± 4.19
21	perplexity_score	52.23 ± 1.04	93.74 ± 1.29	51.89 ± 0.93	53.00 ± 3.22	48.87 ± 5.12
13	last_emb	52.20 ± 1.12	93.60 ± 0.64	52.85 ± 1.34	55.90 ± 2.53	55.42 ± 3.63
13	attn_score	52.14 ± 0.53	93.58 ± 0.85	51.99 ± 0.63	54.10 ± 3.80	53.85 ± 3.73
17	last_emb	52.12 ± 1.01	93.88 ± 0.94	52.85 ± 1.27	54.90 ± 2.70	54.43 ± 3.55
15	attn_score	52.10 ± 0.80	93.62 ± 0.78	51.80 ± 0.90	53.40 ± 3.15	54.11 ± 3.24
25	attn_score	52.07 ± 0.50	93.58 ± 1.08	51.36 ± 0.69	52.80 ± 3.15	53.44 ± 2.91
25	last_emb	52.05 ± 1.37	94.74 ± 0.89	53.04 ± 1.60	54.70 ± 4.96	53.85 ± 6.23
5	max_emb	52.04 ± 0.33	94.44 ± 0.79	51.69 ± 0.63	49.00 ± 2.50	49.10 ± 2.15
27	last_emb	52.03 ± 1.39	94.86 ± 0.84	53.00 ± 1.53	56.20 ± 2.77	55.99 ± 3.21
11	attn_score	52.01 ± 0.58	93.12 ± 1.32	51.60 ± 0.68	53.70 ± 2.84	53.87 ± 3.41
29	last_emb	51.99 ± 1.37	95.04 ± 0.55	52.86 ± 1.44	54.90 ± 4.93	54.59 ± 5.76
25	logit_entropy_score	51.92 ± 1.34	94.52 ± 0.93	51.61 ± 1.18	51.30 ± 4.19	51.06 ± 3.97
23	last_emb	51.87 ± 1.32	94.70 ± 0.91	52.83 ± 1.68	56.90 ± 2.84	57.10 ± 2.82
29	logit_entropy_score	51.87 ± 1.16	94.24 ± 0.65	51.59 ± 1.00	53.10 ± 3.85	50.41 ± 4.61
27	attn_score	51.86 ± 0.51	93.68 ± 0.56	51.80 ± 0.44	53.20 ± 2.89	52.86 ± 3.66
19	last_emb	51.83 ± 1.03	94.34 ± 1.04	52.42 ± 1.09	55.40 ± 3.65	55.31 ± 4.60
27	logit_entropy_score	51.83 ± 1.23	94.10 ± 0.61	51.67 ± 1.08	52.50 ± 3.39	51.47 ± 3.59
-1	max_emb	51.82 ± 0.96	94.66 ± 1.06	51.50 ± 0.75	49.10 ± 3.13	47.99 ± 3.19
9	attn_score	51.80 ± 0.72	93.82 ± 0.88	51.64 ± 0.80	52.60 ± 2.82	53.17 ± 4.00
27	window_logit_entropy_score	51.73 ± 1.39	91.86 ± 1.01	51.54 ± 1.17	51.60 ± 3.52	42.48 ± 11.09
7	attn_score	51.71 ± 0.58	94.08 ± 0.89	51.51 ± 0.69	54.40 ± 2.38	55.44 ± 2.56
21	last_emb	51.69 ± 1.11	94.42 ± 1.12	52.50 ± 1.21	54.40 ± 1.88	55.16 ± 3.66
9	last_emb	51.68 ± 1.22	94.66 ± 1.03	52.51 ± 1.26	55.50 ± 1.06	55.61 ± 1.75
19	perplexity_score	51.66 ± 0.73	93.70 ± 0.61	51.17 ± 0.66	49.40 ± 4.38	48.68 ± 3.69
23	logit_entropy_score	51.65 ± 1.69	95.02 ± 1.17	51.70 ± 1.61	50.40 ± 2.43	51.10 ± 2.02
-1	attn_score	51.61 ± 0.43	94.26 ± 0.76	51.46 ± 0.62	52.80 ± 3.68	53.47 ± 3.08
29	attn_score	51.51 ± 1.02	94.82 ± 0.54	51.59 ± 0.92	51.70 ± 2.80	51.05 ± 3.82
21	logit_entropy_score	51.38 ± 1.83	94.48 ± 0.95	51.61 ± 1.53	49.80 ± 1.60	39.81 ± 22.29
15	window_logit_entropy_score	51.22 ± 0.52	93.68 ± 0.77	50.90 ± 0.70	49.70 ± 2.08	49.19 ± 2.14
17	perplexity_score	51.19 ± 0.61	94.14 ± 0.65	50.88 ± 0.55	48.80 ± 4.01	49.14 ± 3.00
7	last_emb	51.17 ± 1.27	94.56 ± 1.04	52.18 ± 1.04	52.80 ± 1.89	53.56 ± 2.63
29	window_logit_entropy_score	51.14 ± 1.42	92.28 ± 1.25	51.07 ± 1.08	49.10 ± 1.92	37.68 ± 2.39
23	attn_score	51.06 ± 0.66	94.62 ± 0.70	51.06 ± 0.88	51.70 ± 2.84	52.05 ± 3.76
3	perplexity_score	50.88 ± 0.28	94.42 ± 0.97	50.71 ± 0.36	48.50 ± 3.84	49.39 ± 3.08
1	perplexity_score	50.87 ± 0.31	94.40 ± 0.91	50.70 ± 0.38	48.70 ± 4.01	49.62 ± 2.77
1	feat_var_emb	50.87 ± 1.01	100.00	50.62 ± 0.92	50.80 ± 1.20	49.69 ± 1.04
1	attn_score	50.86 ± 0.37	94.00 ± 0.95	50.69 ± 0.55	52.60 ± 3.36	54.25 ± 2.75
5	perplexity_score	50.86 ± 0.30	94.32 ± 0.84	50.69 ± 0.38	48.90 ± 4.23	49.84 ± 2.66
17	feat_var_emb	50.86 ± 0.99	100.00	50.45 ± 0.73	53.40 ± 3.52	53.05 ± 3.02
13	perplexity_score	50.83 ± 0.39	94.28 ± 1.08	50.67 ± 0.40	48.60 ± 2.99	48.91 ± 2.28
15	perplexity_score	50.83 ± 0.46	94.18 ± 1.10	50.65 ± 0.42	48.20 ± 3.85	48.48 ± 3.50
7	perplexity_score	50.80 ± 0.33	94.30 ± 0.83	50.68 ± 0.39	48.90 ± 3.61	49.62 ± 1.65
9	perplexity_score	50.80 ± 0.33	94.34 ± 0.78	50.66 ± 0.37	48.60 ± 4.29	49.38 ± 2.19
11	perplexity_score	50.78 ± 0.37	94.38 ± 0.95	50.65 ± 0.37	49.00 ± 3.14	49.42 ± 2.41
3	attn_score	50.77 ± 0.44	94.12 ± 0.81	50.52 ± 0.84	51.20 ± 2.14	52.48 ± 2.90
5	last_emb	50.63 ± 1.47	94.60 ± 1.09	51.12 ± 1.19	47.10 ± 1.39	47.13 ± 2.07
19	attn_score	50.61 ± 1.34	94.38 ± 0.89	50.66 ± 0.89	54.00 ± 5.55	54.94 ± 5.83
17	logit_entropy_score	50.58 ± 1.29	95.68 ± 1.02	51.03 ± 0.56	50.30 ± 1.96	50.08 ± 2.88
13	window_logit_entropy_score	50.54 ± 1.03	93.56 ± 0.43	50.71 ± 1.10	51.80 ± 0.45	41.60 ± 20.17
21	attn_score	50.54 ± 1.07	94.68 ± 0.89	50.68 ± 1.13	48.60 ± 3.65	49.55 ± 4.13
5	window_logit_entropy_score	50.52 ± 0.59	97.44 ± 0.23	50.60 ± 0.60	50.50 ± 3.22	47.72 ± 3.87
3	window_logit_entropy_score	50.49 ± 0.30	100.00	50.84 ± 0.40	49.30 ± 1.68	38.62 ± 12.13
17	attn_score	50.49 ± 0.68	94.58 ± 1.00	50.73 ± 0.66	51.60 ± 4.76	40.72 ± 23.47

Table 4 — continued

layer	aggregation	auroc ↑ (OOD)	fpr95 ↓ (OOD)	auprc ↑ (OOD)	acc ↑ (probe)	f1 ↑ (probe)
15	feat_var_emb	50.38 ± 0.71	100.00	50.39 ± 0.64	54.60 ± 1.29	53.27 ± 1.81
3	feat_var_emb	50.36 ± 0.82	100.00	50.39 ± 0.81	51.90 ± 2.82	50.67 ± 2.72
19	logit_entropy_score	50.28 ± 1.39	95.26 ± 0.47	50.66 ± 1.07	49.40 ± 1.78	50.36 ± 2.40
7	feat_var_emb	50.23 ± 0.38	100.00	50.27 ± 0.28	51.30 ± 2.89	50.88 ± 2.12
5	attn_score	50.17 ± 0.76	94.50 ± 0.32	50.35 ± 0.85	47.80 ± 3.85	46.48 ± 4.49
9	feat_var_emb	50.17 ± 0.38	100.00	50.24 ± 0.29	51.70 ± 2.75	52.19 ± 3.12
19	feat_var_emb	50.15 ± 1.07	93.84 ± 3.64	49.91 ± 0.63	52.70 ± 1.82	52.02 ± 2.68
3	last_emb	50.13 ± 1.70	94.94 ± 1.26	50.47 ± 0.99	49.70 ± 2.08	51.30 ± 1.64
11	feat_var_emb	50.13 ± 0.34	100.00	50.19 ± 0.29	54.60 ± 3.17	54.11 ± 3.48
5	feat_var_emb	50.12 ± 0.45	100.00	50.22 ± 0.35	52.60 ± 2.53	51.63 ± 2.57
1	last_emb	50.09 ± 1.84	93.94 ± 1.15	50.41 ± 1.45	50.10 ± 1.82	50.05 ± 0.92
25	max_emb	50.07 ± 0.70	95.14 ± 0.33	50.16 ± 0.47	49.10 ± 2.22	48.85 ± 3.66
25	window_logit_entropy_score	50.05 ± 1.47	94.68 ± 0.94	50.31 ± 1.14	50.60 ± 2.77	33.24 ± 19.25
27	max_emb	49.99 ± 0.72	95.12 ± 0.18	50.16 ± 0.51	48.00 ± 1.87	47.20 ± 2.59
23	max_emb	49.97 ± 0.63	94.76 ± 0.39	50.11 ± 0.66	48.00 ± 3.74	46.99 ± 4.45
17	window_logit_entropy_score	49.95 ± 1.43	95.44 ± 0.60	50.60 ± 2.03	50.40 ± 1.29	35.16 ± 27.24
11	window_logit_entropy_score	49.93 ± 0.65	94.80 ± 0.67	50.50 ± 0.67	48.80 ± 4.60	48.10 ± 5.29
1	max_emb	49.89 ± 1.12	95.92 ± 0.76	50.15 ± 0.66	47.20 ± 1.44	46.43 ± 3.72
29	max_emb	49.87 ± 0.87	94.60 ± 0.46	50.11 ± 0.67	49.60 ± 4.46	49.49 ± 4.45
13	feat_var_emb	49.84 ± 0.22	100.00	50.10 ± 0.18	53.20 ± 3.49	52.25 ± 3.43
27	feat_var_emb	49.79 ± 0.77	95.20 ± 0.51	49.87 ± 0.59	49.60 ± 3.42	47.00 ± 4.38
29	feat_var_emb	49.73 ± 0.83	94.64 ± 0.64	49.79 ± 0.70	53.50 ± 4.21	51.57 ± 4.84
21	max_emb	49.64 ± 0.80	94.66 ± 0.55	49.79 ± 0.86	48.20 ± 1.35	46.36 ± 0.92
23	window_logit_entropy_score	49.63 ± 1.43	95.34 ± 1.23	50.30 ± 1.20	51.30 ± 2.80	45.91 ± 15.72
3	max_emb	49.55 ± 1.46	95.14 ± 0.90	49.65 ± 0.93	49.20 ± 3.17	48.55 ± 3.68
1	window_logit_entropy_score	49.54 ± 1.27	95.22 ± 1.67	49.69 ± 1.24	47.30 ± 3.88	48.19 ± 3.90
19	max_emb	49.50 ± 0.71	94.40 ± 0.22	49.50 ± 0.91	50.00 ± 2.52	49.07 ± 2.70
21	window_logit_entropy_score	49.33 ± 1.66	100.00	49.84 ± 1.21	49.20 ± 4.48	39.06 ± 24.15
21	feat_var_emb	49.30 ± 0.59	96.24 ± 0.45	49.51 ± 0.37	50.00 ± 4.23	48.46 ± 4.22
19	window_logit_entropy_score	49.29 ± 0.51	95.16 ± 0.94	49.65 ± 0.92	47.70 ± 3.73	45.16 ± 13.44
11	max_emb	49.29 ± 0.68	95.28 ± 1.28	49.63 ± 0.51	51.20 ± 0.97	51.41 ± 2.66
-1	last_emb	49.26 ± 1.27	95.26 ± 0.87	50.27 ± 0.99	56.90 ± 3.11	55.85 ± 3.25
-1	perplexity_score	49.20 ± 1.95	94.66 ± 0.92	50.27 ± 1.65	50.50 ± 4.50	51.35 ± 4.70
7	window_logit_entropy_score	49.20 ± 0.83	95.48 ± 0.61	49.93 ± 0.79	50.80 ± 4.51	51.33 ± 3.81
5	logit_entropy_score	49.18 ± 0.28	94.90 ± 0.39	49.42 ± 0.23	52.20 ± 2.91	52.84 ± 2.54
7	logit_entropy_score	49.16 ± 0.28	94.82 ± 0.48	49.41 ± 0.26	52.20 ± 3.05	52.58 ± 2.19
1	logit_entropy_score	49.15 ± 0.29	94.92 ± 0.40	49.39 ± 0.24	52.20 ± 3.07	52.76 ± 2.48
3	logit_entropy_score	49.15 ± 0.29	94.92 ± 0.31	49.39 ± 0.24	52.20 ± 3.07	52.76 ± 2.48
11	logit_entropy_score	49.13 ± 0.35	94.96 ± 0.51	49.37 ± 0.31	51.40 ± 3.11	41.97 ± 23.61
25	feat_var_emb	49.11 ± 0.46	95.26 ± 0.54	49.65 ± 0.48	49.80 ± 2.36	47.52 ± 3.56
9	logit_entropy_score	49.11 ± 0.37	94.94 ± 0.60	49.36 ± 0.30	51.80 ± 3.40	51.89 ± 2.80
-1	window_logit_entropy_score	49.09 ± 1.99	94.64 ± 0.43	49.90 ± 1.74	51.70 ± 3.33	40.09 ± 16.00
-1	logit_entropy_score	49.06 ± 0.89	94.84 ± 0.65	49.60 ± 0.87	49.10 ± 2.33	30.85 ± 17.04
29	avg_emb	49.02 ± 1.14	95.60 ± 1.02	49.84 ± 0.98	52.90 ± 3.73	51.53 ± 4.66
23	feat_var_emb	49.01 ± 0.42	94.98 ± 0.79	49.47 ± 0.41	51.50 ± 3.30	49.89 ± 4.03
15	logit_entropy_score	48.97 ± 0.51	94.80 ± 0.65	49.52 ± 0.43	52.80 ± 2.84	53.35 ± 2.64
13	logit_entropy_score	48.97 ± 0.36	94.62 ± 0.41	49.34 ± 0.33	52.30 ± 3.67	52.45 ± 2.70
9	max_emb	48.94 ± 0.28	95.38 ± 1.08	49.51 ± 0.26	48.30 ± 0.84	47.80 ± 1.48
17	max_emb	48.91 ± 0.75	94.74 ± 0.77	49.36 ± 0.84	50.50 ± 3.24	49.81 ± 3.50
-1	feat_var_emb	48.82 ± 2.02	96.00 ± 0.71	49.32 ± 1.77	51.20 ± 0.76	50.08 ± 1.55
9	window_logit_entropy_score	48.78 ± 1.69	95.50 ± 0.94	49.27 ± 1.80	54.70 ± 1.89	54.62 ± 3.66
15	max_emb	48.77 ± 0.25	94.82 ± 1.08	49.14 ± 0.50	52.00 ± 1.46	52.42 ± 1.59
3	avg_emb	48.63 ± 1.39	96.06 ± 1.76	49.16 ± 0.71	50.80 ± 1.48	49.58 ± 1.46
27	avg_emb	48.60 ± 1.15	95.48 ± 1.41	49.53 ± 0.82	52.80 ± 2.36	51.13 ± 2.59
25	avg_emb	48.54 ± 1.17	95.56 ± 1.50	49.48 ± 0.78	52.50 ± 2.55	51.27 ± 2.38
1	avg_emb	48.53 ± 1.27	96.02 ± 1.69	49.13 ± 0.66	49.10 ± 0.74	47.74 ± 1.94
7	max_emb	48.50 ± 0.83	100.00	49.15 ± 0.63	50.50 ± 1.97	49.42 ± 2.42
5	avg_emb	48.46 ± 1.39	96.02 ± 1.62	49.06 ± 0.70	51.90 ± 2.41	50.46 ± 2.29
7	avg_emb	48.45 ± 1.29	95.98 ± 1.70	49.10 ± 0.60	52.20 ± 1.82	52.11 ± 1.42
9	avg_emb	48.33 ± 1.24	96.00 ± 1.76	49.05 ± 0.53	52.50 ± 2.00	51.48 ± 1.96
23	avg_emb	48.31 ± 1.22	95.64 ± 1.58	49.31 ± 0.70	52.40 ± 2.58	50.74 ± 1.91
19	avg_emb	48.31 ± 1.39	95.82 ± 1.69	49.13 ± 0.63	55.20 ± 2.51	54.29 ± 2.20
17	avg_emb	48.31 ± 1.36	95.98 ± 1.74	49.10 ± 0.65	55.90 ± 4.36	54.87 ± 4.29
11	avg_emb	48.24 ± 1.16	96.06 ± 1.69	48.97 ± 0.37	53.60 ± 2.07	52.64 ± 2.14
21	avg_emb	48.20 ± 1.23	95.74 ± 1.49	49.13 ± 0.66	54.00 ± 2.74	52.94 ± 3.06
13	avg_emb	48.19 ± 1.22	96.12 ± 1.64	48.93 ± 0.45	55.40 ± 3.81	54.18 ± 4.15
15	avg_emb	48.19 ± 1.27	96.10 ± 1.69	48.93 ± 0.45	54.70 ± 3.33	53.35 ± 2.66
13	max_emb	48.05 ± 0.06	95.46 ± 1.03	48.80 ± 0.38	49.70 ± 2.36	48.63 ± 1.88

Table 4 — continued

layer	**aggregation**	**auroc ↑ (OOD)**	**fpr95 ↓ (OOD)**	**auprc ↑ (OOD)**	**acc ↑ (probe)**	**f1 ↑ (probe)**
1	hidden_score	43.74 ± 1.36	96.20 ± 1.42	45.17 ± 0.93	52.10 ± 3.52	47.75 ± 4.01
-1	hidden_score	42.36 ± 1.05	95.36 ± 2.42	45.36 ± 0.60	56.80 ± 3.09	54.78 ± 3.58
9	hidden_score	42.32 ± 1.10	96.42 ± 1.38	45.11 ± 0.84	56.10 ± 4.76	52.85 ± 4.98
3	hidden_score	42.32 ± 0.95	96.48 ± 1.37	44.90 ± 0.91	55.60 ± 4.20	51.64 ± 4.54
29	hidden_score	42.23 ± 1.04	96.40 ± 1.07	45.28 ± 0.57	56.10 ± 3.80	52.23 ± 3.97
27	hidden_score	42.22 ± 1.01	96.62 ± 1.10	45.23 ± 0.56	56.60 ± 3.63	52.94 ± 3.64
5	hidden_score	42.20 ± 1.00	96.56 ± 1.53	44.92 ± 0.85	56.30 ± 3.55	52.82 ± 4.24
25	hidden_score	42.18 ± 0.99	96.86 ± 0.86	45.19 ± 0.63	55.60 ± 3.25	51.73 ± 3.61
15	hidden_score	42.12 ± 1.06	96.82 ± 1.10	44.98 ± 0.76	56.20 ± 4.35	52.71 ± 4.46
13	hidden_score	42.11 ± 1.15	96.56 ± 1.19	45.00 ± 0.78	56.20 ± 4.02	52.69 ± 4.40
19	hidden_score	42.08 ± 1.02	96.74 ± 1.27	45.00 ± 0.73	55.60 ± 3.11	52.26 ± 3.25
11	hidden_score	42.08 ± 0.95	96.38 ± 1.21	44.97 ± 0.77	56.60 ± 4.29	53.63 ± 4.54
17	hidden_score	42.03 ± 1.11	96.74 ± 1.32	44.97 ± 0.73	55.90 ± 3.93	52.70 ± 4.53
23	hidden_score	42.02 ± 1.14	96.78 ± 0.92	45.11 ± 0.69	56.00 ± 2.87	52.47 ± 3.26
7	hidden_score	41.99 ± 1.02	96.88 ± 1.07	44.91 ± 0.86	56.20 ± 4.09	52.68 ± 4.53
21	hidden_score	41.97 ± 1.11	96.74 ± 1.01	45.04 ± 0.65	56.40 ± 3.58	52.81 ± 3.81