CHORD: Cross-Model Hallucination Detection via Relational Graph Discrimination

Yongxin Deng
University of Technology Sydney
Sydney, Australia
yongxin.deng@student.uts.edu.au

Zhen Fang*
University of Technology Sydney
Sydney, Australia
zhen.fang@uts.edu.au

Guansong Pang
Singapore Management University
Singapore
gspang@smu.edu.sg

Sharon Li
University of Wisconsin–Madison
Madison, WI, USA
sharonli@cs.wisc.edu

Ling Chen*
University of Technology Sydney
Sydney, Australia
ling.chen@uts.edu.au

Abstract

Hallucination detection is critical for deploying large language models (LLMs) in real-world applications. Due to strong empirical performance, internal representation–based methods have emerged as the prevailing direction for detecting hallucinations, yet they remain largely model-specific and often fail to generalize to unseen LLMs. In this paper, we study an important yet underexplored problem, termed *cross-model hallucination detection* (CMHD), which aims to train hallucination detectors on source LLMs while ensuring performance on unseen target LLMs. The core challenge of CMHD lies in cross-model representation heterogeneity: hidden states from different LLMs exhibit severe semantic inconsistency, which limits the transferability of classical representation-based detectors. Through analysis and empirical validation, we show that relational structures over layers, tokens, and features capture transferable detection signals despite semantic inconsistency. Based on this observation, we propose *cross-model hallucination detection via heterogeneity-oriented relational discrimination* (CHORD), a relational graph discrimination framework that represents hidden states as joint relational graphs over layers, tokens, and features. It feeds these graphs into relation-aware attention to obtain structure-aware features, and uses meta-learning to shape these features toward transferable detection signals. Extensive experiments show that CHORD outperforms representative detection methods in cross-model generalization.

1 Introduction

Large language models (LLMs) may suffer from hallucinations, generating outputs that are fluent and plausible yet factually incorrect or unsupported by verifiable evidence. Such behavior poses a major challenge to LLMs' safe adoption, particularly in safety-critical domains such as education, healthcare, and finance [1–3]. Hallucination detection has therefore attracted growing attention in recent research. Existing detection methods can be broadly divided into two categories: training-free methods [4, 5] and training-based methods [6, 7]. Among them, training-based methods, which mainly leverage internal representations through lightweight classifiers, have emerged as an important and promising research direction, largely driven by their strong in-domain performance [8–10].

*Corresponding authors: Zhen Fang and Ling Chen.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

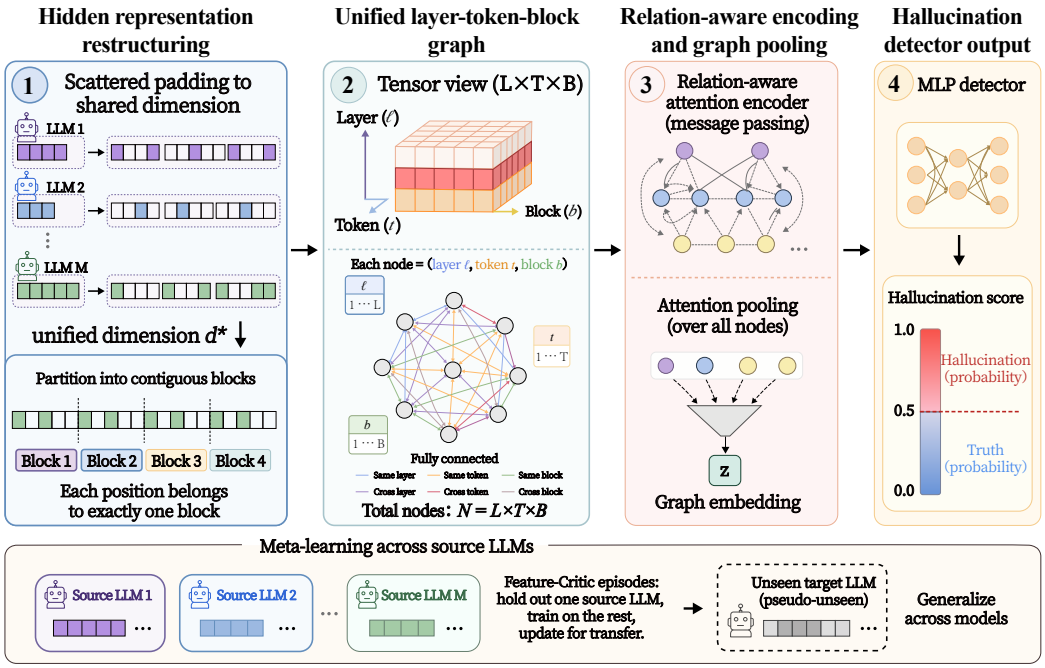

Figure 1: Overview of CHORD. CHORD maps hidden states into a shared space, constructs a *layer–token–block graph*, and uses relation-aware attention with graph pooling for hallucination prediction. Feature-Critic meta-learning encourages the detector to learn transferable signals.

Despite their empirical success, training-based methods still suffer from a critical limitation: the learned detectors are highly *model-specific*. In particular, detectors trained on one LLM typically fail to generalize to an unseen model, even when the model performs similar tasks (see Appendix A.1). Such limited cross-model generalization substantially reduces the practical utility of training-based hallucination detection, as detectors learned on one model often fail on another, requiring retraining or redesign for each new model. Beyond hampering scalability, this also incurs substantial additional costs in supervision, computation, and deployment. These limitations naturally lead to a central question: *how can we build a detector that generalizes well across models, even to unseen ones?*

To tackle the cross-model generalization issue in hallucination detection, we study a more realistic yet challenging setting, termed *cross-model hallucination detection* (CMHD), which seeks to learn a hallucination detector from multiple source models that generalizes well to unseen target models. The core difficulty of CMHD lies in the pronounced *heterogeneity across models*. Different LLMs may vary in several respects, including architecture family, parameter scale, and training procedure, and these differences can further induce different generation behaviors. At the representation level, such heterogeneity typically manifests as severe *cross-model semantic inconsistency*, which fundamentally limits the transferability and generalization of detectors learned from source models to target models.

To enable cross-model generalization, we seek stable, transferable detection signals across heterogeneous semantic representations, reducing reliance on model-specific semantics. Inspired by Binkowski et al. [11] demonstrating that token-level relational structures aid hallucination detection without relying on individual semantic representations, we construct relation-weighted token graphs from hidden representations, evaluating them through cross-model structural alignment and predictive transfer. In Figures 2 and 3, token-level relational features exhibit non-trivial cross-model structural alignment, and their source-to-target AUROC is comparable to representative training-free methods across models. These results suggest relational structures provide a promising transferable signal for CMHD. Building on this observation and evidence that LLMs encode hierarchical semantics across layers and rich interactions within feature dimensions [12, 13], we broaden the relational view to *cross-layer token relations*, *within-token feature relations*, and *joint relations* over layers, tokens, and features. We summarize each graph with permutation-invariant pooling to reduce sensitivity to node ordering. Analysis in Section 3 shows that the resulting joint relational structures achieve favorable cross-model stability and the highest cross-model AUROC among relational features.

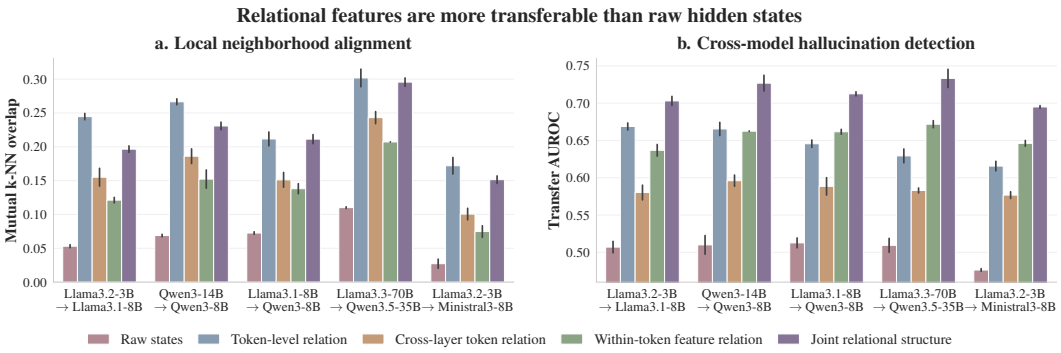

Figure 2: Cross-model alignment and hallucination detection transfer. Raw hidden states show weak local neighborhood agreement and near-chance transfer, while relational constructions improve both mutual kNN overlap and source-to-target AUROC. Experimental details are offered in Appendix A.8.

Empirical observations suggest that joint relations over layers, tokens, and feature blocks carry information that is transferable across LLMs and discriminative for hallucination detection. Notably, even a construction with fixed handcrafted edges yields useful cross-model signals, suggesting the promise of the relational perspective. This motivates us to move beyond fixed relations and learn more effective relational structures for cross-model hallucination detection. Based on this motivation, we propose *cross-model hallucination detection via heterogeneity-oriented relational discrimination* (CHORD), a relational graph discrimination framework for cross-model hallucination detection. CHORD represents hidden states as fully connected joint relational graphs over layers, tokens, and feature blocks, where nodes encode layer-token-feature information and learnable edges capture their relational dependencies. It feeds these graphs into relation-aware attention to obtain structure-aware features for distinguishing truthful and hallucinated responses. To improve transfer to unseen LLMs, CHORD is trained with a meta-learning strategy [14] that holds out one source model as a pseudo target model, encouraging the detector to learn transferable detection signals across models.

We conduct extensive experiments in a broad, challenging cross-family transfer setting on eight LLMs from three model families and multiple scales, i.e., Llama-3.2-3B/3.1-8B/3.3-70B [15], Ministral-3-8B/14B [16], Qwen3-8B/14B [17] and Qwen3.5-35B-A3B [18], and seven benchmarks, i.e., TruthfulQA [19], HaluEval [20], TriviaQA [21], CoQA [22], GSM8K [23], MBPP [24], and xLAM [25], spanning factual and conversational question answering, mathematical reasoning, coding, and agent tasks. Section 5 shows CHORD consistently improves cross-model generalization and outperforms representative baselines, verifying the effectiveness of relational graph discrimination for hallucination detection across unseen LLMs. *Our main contributions are summarized below:*

• We formulate and study *cross-model hallucination detection* (CMHD), a practical setting where hallucination detectors are trained on source LLMs and evaluated on unseen target LLMs.

• Analysis suggests relational structures over layers, tokens, and feature blocks serve as promising hallucination signals for CMHD, exhibiting favorable cross-model alignment and detection transfer.

• We propose CHORD, a relational graph discrimination framework with relation-aware attention and meta-learning, and demonstrate its effectiveness across model families, scales, and task types.

2 Learning Setups

LLMs and Token Sequences. Following [26–28], we model an LLM as a conditional distribution $\mathbb{P}_{\boldsymbol{\theta}}(\cdot)$ over token sequences, where $\boldsymbol{\theta}$ denotes the model parameters. Given an input token sequence $\mathbf{Q} = [x_1, \ldots, x_k]$, representing a question optionally concatenated with context, the LLM generates an answer token sequence $\mathbf{A} = [x_{k+1}, \ldots, x_{k+l}]$ in an autoregressive manner by predicting each token conditioned on the preceding tokens: $\mathbb{P}_{\boldsymbol{\theta}}(x_j \mid x_1, \ldots, x_{j-1})$, for $j = k{+}1, \ldots, k{+}l$.

This work studies multiple models. Let $\mathbb{P}_{\boldsymbol{\theta}_1}, \ldots, \mathbb{P}_{\boldsymbol{\theta}_M}$ denote the M LLMs available during training. These training models may vary along several axes, including architecture family, parameter scale, and training procedure, and such variation may induce systematically different generation behaviors.

Internal Representations. For the i-th LLM $\mathbb{P}_{\boldsymbol{\theta}_i}$, let L_i denote its number of layers and d_i its hidden dimension. Given an input question-answer pair $(\mathbf{Q}, \mathbf{A})$ with $k + l$ tokens, the hidden

representation at layer ℓ is defined as $\mathbf{H}^{\ell}_{\boldsymbol{\theta}_i}(\mathbf{Q}, \mathbf{A}) \in \mathbb{R}^{(k+l)\times d_i}$. For each token position t, we denote by $\mathbf{h}^{\ell}_{\boldsymbol{\theta}_i,t}(\mathbf{Q}, \mathbf{A}) \in \mathbb{R}^{d_i}$ the hidden representation of the t-th token at layer ℓ.

Domain and Datasets. We define the *truthful-answer domain* as a joint distribution $\mathbb{P}_{Q,T}$, where Q is a random variable representing questions and T is a random variable representing factual answers. Given a question $\mathbf{Q} \sim \mathbb{P}_Q$, the i-th model $\mathbb{P}_{\boldsymbol{\theta}_i}(\cdot)$ generates an answer $\mathbf{A}_i \sim \mathbb{P}_{\boldsymbol{\theta}_i}(\cdot \mid \mathbf{Q})$. $\mathbf{A}_i$ is then assigned a binary label y_i based on its semantic alignment with a reference answer $\mathbf{A}_{\text{ref}} \sim \mathbb{P}_{T|Q}(\cdot \mid \mathbf{Q})$. Specifically, if $\mathbf{A}_i$ is semantically aligned with $\mathbf{A}_{\text{ref}}$ or explicitly abstains (e.g., states that it does not know), it is labeled as truthful ($y_i = 1$); if it commits to an answer that is not aligned with $\mathbf{A}_{\text{ref}}$, it is labeled as hallucinated ($y_i = 0$). Combining $\mathbb{P}_{Q,T}$ with $\mathbb{P}_{\boldsymbol{\theta}_i}$ yields a dataset

$$\mathcal{D}_i = \left\{(\mathbf{Q}^1, \mathbf{A}_i^1, y_i^1), \ldots, (\mathbf{Q}^n, \mathbf{A}_i^n, y_i^n)\right\}, \quad \forall i \in \{1, ..., M\}. \tag{1}$$

Cross-Model Hallucination Detection. The task considered in this work is defined as follows.

Problem 1 (Cross-Model Hallucination Detection (CMHD))**.** Given M training datasets $\mathcal{D}_1, ..., \mathcal{D}_M$ constructed from the truthful-answer domain $\mathbb{P}_{Q,T}$ together with M *source models* $\mathbb{P}_{\boldsymbol{\theta}_1}, \ldots, \mathbb{P}_{\boldsymbol{\theta}_M}$ as introduced in Eq. (1), the goal of cross-model hallucination detection (CMHD) is to learn a detector D from $\mathbb{P}_{\boldsymbol{\theta}_1}, \ldots, \mathbb{P}_{\boldsymbol{\theta}_M}$ and $\mathcal{D}_1, \ldots, \mathcal{D}_M$. During testing, given an unseen *target model* $\mathbb{P}_{\boldsymbol{\theta}}$, the detector D is expected to generalize to $\mathbb{P}_{\boldsymbol{\theta}}$. Concretely, for any question $\mathbf{Q} \sim \mathbb{P}_Q$ and answer $\mathbf{A} \sim \mathbb{P}_{\boldsymbol{\theta}}(\cdot \mid \mathbf{Q})$,

$$D(\mathbf{Q}, \mathbf{A}, \mathbb{P}_{\boldsymbol{\theta}}) = 1, \text{ if } \mathbf{A} \text{ does not contradict } \mathbf{T} \sim \mathbb{P}_{T|Q}(\cdot \mid \mathbf{Q}); \text{ otherwise, } 0. \tag{2}$$

3 Analysis of Hallucination Signals: What Transfers Across Models?

In Figure 4, the representation-based detectors often generalize poorly across models. This failure stems from cross-model semantic inconsistency, limiting transfer from source models to target models [29–31] (see Appendix A.1). We therefore *seek detection signals that remain stable across heterogeneous representations, thereby reducing the detector's reliance on model-specific representations*.

3.1 Stable Cross-Model Signals from Token-Level Relations

Motivated by the recent work of Binkowski et al. [11], which shows that *token-level relational structures* provide highly reliable signals for downstream hallucination detection, we carefully examine whether these token-level relations can effectively serve as universal cross-model signals.

Token-Level Relation. Given a question-answer pair $(\mathbf{Q}, \mathbf{A})$ and the i-th LLM $\mathbb{P}_{\boldsymbol{\theta}_i}$, we obtain the hidden representation $\mathbf{h}^{\ell}_{\boldsymbol{\theta}_i,t}(\mathbf{Q}, \mathbf{A})$ of the t-th token at layer ℓ. Motivated by Wang et al. [32] and the state-delta analysis [33–35], for two token positions t and $\tilde{t}$, we design their relation as follows:

$$R_i^{\text{tok},\ell}(t, \tilde{t}; \mathbf{Q}, \mathbf{A}) = s^{\text{tok}}(t, \tilde{t}) \left\| \mathbf{h}^{\ell}_{\boldsymbol{\theta}_i,t}(\mathbf{Q}, \mathbf{A}) - \mathbf{h}^{\ell}_{\boldsymbol{\theta}_i,\tilde{t}}(\mathbf{Q}, \mathbf{A}) \right\|_2, \tag{3}$$

where $\|\cdot\|_2$ denotes the ℓ_2 norm, and $s^{\text{tok}}(t, \tilde{t}) = \exp\left(-|t - \tilde{t}|/\tau_{\text{tok}}\right)$ is the position-based relation weight between the t-th and $\tilde{t}$-th tokens. Here $\tau_{\text{tok}} > 0$ controls decay with token-position distance.

Remark. *We use this simple, non-learnable relation as a diagnostic test of whether basic relational information supports cross-model detection. Improved transfer under this lightweight construction would imply that relational structure provides meaningful transferable signals for detection.*

Token-Level Relation Graph and Pooling. While Eq. (3) captures relations between token pairs, hallucination signals often emerge from the collective structure among multiple tokens. We therefore aggregate pairwise relations over selected tokens into a compact vector representation that summarizes the relational structure of the question-answer pair $(\mathbf{Q}, \mathbf{A})$. Let $\mathbf{T}^{\text{sel}} = \{t_1, \ldots, t_r\}$ denote the selected token positions. We construct a fully connected token-level graph $\mathcal{G}_i^{\text{tok},\ell}(\mathbf{Q}, \mathbf{A})$ without self-loops, where each node corresponds to a selected token and each edge $(t, \tilde{t})$ is weighted by the token-level relation $R_i^{\text{tok},\ell}(t, \tilde{t}; \mathbf{Q}, \mathbf{A})$ defined in Eq. (3). The node feature at position t is the hidden representation $\mathbf{h}^{\ell}_{\boldsymbol{\theta}_i,t}(\mathbf{Q}, \mathbf{A})$ extracted from $\mathbb{P}_{\boldsymbol{\theta}_i}$. Following the standard graph message-passing and readout paradigm [36, 37], we perform one relation-weighted aggregation step and mean readout:

$$\mathbf{z}_i^{\text{tok},\ell}(\mathbf{Q}, \mathbf{A}) = \frac{1}{r} \sum_{t \in \mathbf{T}^{\text{sel}}} \sum_{\tilde{t} \in \mathbf{T}^{\text{sel}},\, \tilde{t} \neq t} \frac{R_i^{\text{tok},\ell}(t, \tilde{t}; \mathbf{Q}, \mathbf{A})}{\sum_{t' \in \mathbf{T}^{\text{sel}},\, t' \neq t} R_i^{\text{tok},\ell}(t, t'; \mathbf{Q}, \mathbf{A})} \mathbf{h}^{\ell}_{\boldsymbol{\theta}_i,\tilde{t}}(\mathbf{Q}, \mathbf{A}). \tag{4}$$

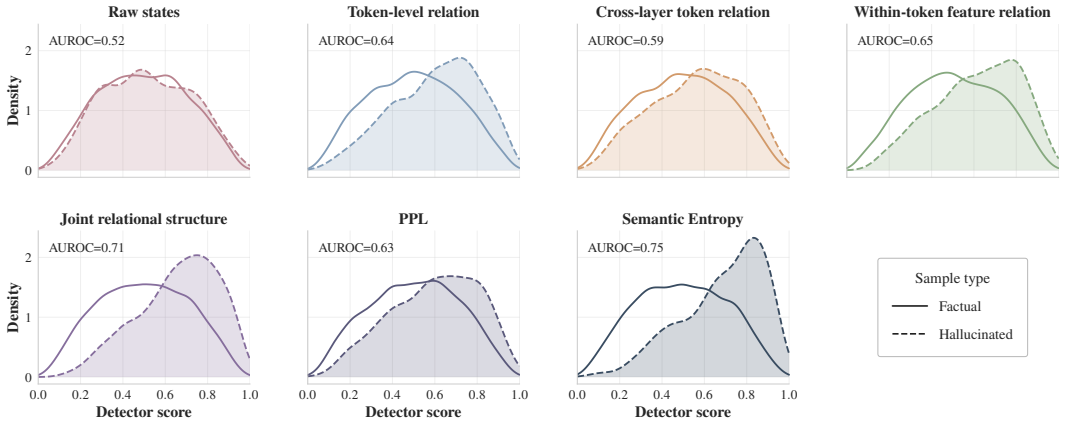

Figure 3: Detector score distributions for Llama 8B to Qwen 8B. Token-level relations already separate factual and hallucinated examples better than PPL, while stronger relational constructions approach the Semantic Entropy baseline. Experimental details are provided in Appendix A.8.

The resulting vector $\mathbf{z}_i^{\mathrm{tok},\ell}(\mathbf{Q},\mathbf{A})$ provides a compact relation-weighted summary of the selected token representations. Because the final mean readout is implemented as graph-level average pooling over all selected token nodes, the summary is permutation-invariant, providing theoretical support for reducing sensitivity to different node orderings in graphs derived from different LLMs.

Remark. *To ensure dimension consistency across LLMs, we map hidden representations into a common space before the aggregation in Eq. (4). Details are provided in Eq. (11) and Section 4.*

Evaluation for Cross-Model Stability and Transferability. We assess the token-level features $\mathbf{z}_i^{\mathrm{tok},\ell}(\mathbf{Q},\mathbf{A})$ using two evaluation criteria: *cross-model structural stability* and *predictive transfer*. 1) Structural stability is measured by mutual k-nearest-neighbor alignment [30, 38–40], which assesses whether the induced relational structure is consistent across LLMs. 2) Predictive transfer is measured by cross-model AUROC, which evaluates whether features from one LLM can predict hallucinations on another. Both evaluation criteria involve two LLMs $\mathbb{P}_{\boldsymbol{\theta}_i}$ and $\mathbb{P}_{\boldsymbol{\theta}_j}$.

Following Huh et al. [38], we use mutual k-nearest-neighbor (kNN) alignment to assess cross-model structural stability. For each pair $(\mathbf{Q}^s,\mathbf{A}_i^s)$ from dataset $\mathcal{D}_i$ (see Eq. (1)), mutual kNN alignment is

$$m_{\mathrm{NN}}^{\mathrm{tok},\ell}(i,j)=\frac{1}{nk}\sum_{s=1}^{n}\left|\mathcal{N}_k^i(s)\cap\mathcal{N}_k^j(s)\right|,\ \text{where}\ \mathcal{N}_k^i(s)=\operatorname{Top}k_{s'\in\{1,\dots,n\}\setminus\{s\}}\,\mathrm{sim}_i^{\mathrm{tok},\ell}(s,s'),\quad(5)$$

here $|\cdot|$ counts the number of elements in the intersection $\mathcal{N}_k^i(s)\cap\mathcal{N}_k^j(s)$, and $\operatorname{Top}k$ returns the indices of the k largest similarity scores $\mathrm{sim}_i^{\mathrm{tok},\ell}(s,s')$. Following Huh et al. [38], we use the inner product to compute the similarity score, i.e., $\mathrm{sim}_i^{\mathrm{tok},\ell}(s,s')=\langle\mathbf{z}_i^{\mathrm{tok},\ell}(\mathbf{Q}^s,\mathbf{A}_i^s),\mathbf{z}_i^{\mathrm{tok},\ell}(\mathbf{Q}^{s'},\mathbf{A}_i^{s'})\rangle$.

Cross-model AUROC is used to assess predictive transfer of hallucination detection. Specifically, an MLP detector is trained on the source features $\mathbf{z}_i^{\mathrm{tok},\ell}(\mathbf{Q}^s,\mathbf{A}_i^s)$ from $\mathcal{D}_i$ with labels y_i^s. The trained detector is then evaluated directly on the target features $\mathbf{z}_j^{\mathrm{tok},\ell}(\mathbf{Q}^s,\mathbf{A}_j^s)$ from $\mathcal{D}_j$ using labels y_j^s. A higher AUROC indicates stronger cross-model transfer of hallucination-discriminative signals.

Empirical Observations and Implications. We evaluate $\mathbf{z}_i^{\mathrm{tok},\ell}(\mathbf{Q},\mathbf{A})$ on five source-target LLM pairs and highlight two key observations with their implications. The observations are shown below.

(1) *Cross-model stability.* In Figure 2a, raw hidden representations yield mutual kNN alignments of 0.03–0.11, whereas the token-level relational features increase the alignment to 0.17–0.30 across the five transfer pairs. This indicates that the relational features preserve a more consistent neighborhood structure across LLMs than raw representations. To contextualize these values, if the two kNN sets were randomly and independently selected from the $n-1$ remaining samples, their expected normalized overlap would be $k/(n-1)$. In our setup, where $k=10$ and $n=5000$, this random-overlap baseline is near 0.002. The observed token-level alignments are therefore roughly 85 to 150 times random neighborhood agreement, indicating non-trivial cross-model relational stability.

(2) *Cross-model transferability.* In Figure 2b, an MLP detector trained on raw hidden representations from a source LLM and evaluated on a target LLM stays close to chance (AUROC $\approx$ 0.47–0.52),

while the same detector trained on token-level relational features reaches 0.62–0.67 across all transfer pairs. We further compare with two training-free baselines (see Figure 3) that are inherently cross-model: Perplexity (PPL) [41] as a general-purpose baseline at about 0.63, and Semantic Entropy [5] as a strong specialized baseline at about 0.75. The token-level cross-model AUROC thus falls short of Semantic Entropy, yet matches PPL despite the simplicity of its construction.

Implication. Taken together, these observations suggest that token-level relational features provide a stable and transferable structural signal for cross-model hallucination detection, beyond what is captured by raw hidden representations. However, the remaining gap to Semantic Entropy indicates that token-level relations capture only part of the transferable structure needed for robust cross-model generalization. This motivates the broader relational analysis in Section 3.2, where we further study cross-layer relations, within-token feature relations, and their joint relational structures.

Table 1: Cross-model performance (AUROC). To ensure fair comparison, we use a leave-one-family-out protocol: training-based methods are trained on models from two source families and evaluated on models from the held-out target family, while training-free methods are directly applied to the same targets (*Appendix G*). Best results are **bold**, and second-best results are underlined.

Method	Target Model															
	Llama3.2-3B								Llama3.1-8B							
	Mean AUROC↑								Mean AUROC↑							
	TruthfulQA	HaluEval	TriviaQA	CoQA	GSM8K	MBPP	xLAM	**Average**	TruthfulQA	HaluEval	TriviaQA	CoQA	GSM8K	MBPP	xLAM	**Average**
Training-free methods																
Perplexity	0.6329	0.6464	0.6448	0.6296	0.6536	0.6188	0.6621	0.6412	0.6546	0.6452	0.6545	0.6395	0.6523	0.6024	0.6857	0.6477
Semantic Entropy	0.6862	0.7095	0.7006	0.6934	0.7112	0.6571	0.7223	0.6972	0.6991	0.7113	0.7065	0.6919	0.7258	0.6636	0.7438	0.7060
EigenScore	0.6636	0.6872	0.6767	0.6719	0.6772	0.6508	0.7045	0.6760	0.6985	0.6963	0.6803	0.6747	0.7068	0.6614	0.6957	0.6877
Lexical Similarity	0.6542	0.6747	0.6878	0.6786	0.6454	0.6225	0.6603	0.6605	0.6760	0.6681	0.6928	0.6628	0.6495	0.6236	0.6871	0.6657
Verbalize	0.5199	0.5306	0.5220	0.5290	0.5207	0.5069	0.5470	0.5252	0.5238	0.5398	0.5110	0.5050	0.5474	0.4884	0.5595	0.5250
InterrogateLLM	0.6845	0.7353	0.7012	0.6915	0.7177	0.6915	0.7253	0.7067	0.7185	0.7211	0.7201	0.6881	0.7293	0.6865	0.7469	0.7158
Training-based methods																
MM	0.4857	0.5170	0.5129	0.4874	0.5061	0.5035	0.5200	0.5047	0.5127	0.5029	0.5247	0.5000	0.5141	0.4808	0.5052	0.5058
SEP	0.5079	0.5193	0.5373	0.5189	0.5300	0.5032	0.5290	0.5208	0.5191	0.5346	0.5142	0.5089	0.5382	0.4876	0.5395	0.5203
SAPLMA	0.4941	0.5191	0.5239	0.5390	0.5158	0.4901	0.5397	0.5174	0.5326	0.5311	0.5250	0.5013	0.5382	0.4818	0.5386	0.5212
LapEigvals	0.4817	0.5021	0.4740	0.4965	0.5264	0.4765	0.5295	0.4981	0.4981	0.4916	0.4712	0.4789	0.5083	0.4725	0.5225	0.4919
Proposed method																
CHORD	**0.7766**	**0.7926**	**0.7991**	**0.7920**	**0.8193**	**0.7351**	**0.8437**	**0.7941**	**0.7989**	**0.8102**	**0.8156**	**0.7823**	**0.8362**	**0.7307**	**0.8650**	**0.8056**

	Llama3.3-70B								Qwen3-8B							
Training-free methods																
Perplexity	0.6575	0.6747	0.6717	0.6748	0.6765	0.6353	0.6931	0.6691	0.6345	0.6398	0.6781	0.6542	0.6653	0.6179	0.6726	0.6518
Semantic Entropy	0.7096	0.7366	0.7231	0.7272	0.7410	0.6697	0.7406	0.7211	0.6840	0.6800	0.7127	0.6994	0.7143	0.6502	0.7228	0.6948
EigenScore	0.7076	0.7087	0.6961	0.7095	0.7094	0.6703	0.7093	0.7016	0.6785	0.6817	0.6869	0.6873	0.6813	0.6683	0.7085	0.6846
Lexical Similarity	0.6844	0.6797	0.7040	0.7088	0.6854	0.6432	0.6858	0.6845	0.6538	0.6538	0.7089	0.6966	0.6441	0.6401	0.6688	0.6666
Verbalize	0.5472	0.5531	0.5472	0.5385	0.5592	0.5139	0.5584	0.5454	0.5193	0.5090	0.5200	0.5170	0.5339	0.4969	0.5463	0.5203
InterrogateLLM	0.7343	0.7454	0.7472	0.7219	0.7473	0.7181	0.7477	0.7374	0.6950	0.7077	0.7044	0.7173	0.7082	0.6834	0.7483	0.7092
Training-based methods																
MM	0.5095	0.5244	0.5288	0.5195	0.5259	0.4814	0.5153	0.5150	0.4923	0.4849	0.5044	0.5095	0.5046	0.4837	0.5171	0.4995
SEP	0.5117	0.5352	0.5400	0.5243	0.5263	0.4905	0.5297	0.5225	0.5147	0.5030	0.4993	0.5062	0.5159	0.4823	0.5228	0.5063
SAPLMA	0.5250	0.5177	0.5231	0.5364	0.5375	0.4840	0.5386	0.5232	0.4822	0.5223	0.5171	0.5093	0.5167	0.4922	0.5384	0.5112
LapEigvals	0.5083	0.4992	0.5167	0.5204	0.5132	0.4718	0.5280	0.5082	0.4831	0.4872	0.5141	0.4873	0.5063	0.4716	0.5292	0.4970
Proposed method																
CHORD	**0.8276**	**0.8374**	**0.8257**	**0.8302**	**0.8568**	**0.7605**	**0.8713**	**0.8299**	**0.7896**	**0.7941**	**0.8192**	**0.7991**	**0.8256**	**0.7481**	**0.8484**	**0.8034**

	Qwen3-14B								Qwen3.5-35B							
Training-free methods																
Perplexity	0.6408	0.6650	0.6442	0.6519	0.6363	0.6164	0.6851	0.6485	0.6460	0.6488	0.6751	0.6601	0.6614	0.6361	0.6732	0.6572
Semantic Entropy	0.6810	0.6968	0.6966	0.6958	0.7193	0.6438	0.7195	0.6933	0.7078	0.7110	0.7211	0.7052	0.7392	0.6738	0.7278	0.7123
EigenScore	0.6931	0.6891	0.6965	0.6748	0.7013	0.6556	0.7090	0.6885	0.7061	0.7018	0.7072	0.6854	0.7093	0.6648	0.7046	0.6970
Lexical Similarity	0.6498	0.6560	0.6779	0.6795	0.6529	0.6368	0.6834	0.6623	0.6752	0.6756	0.7050	0.7083	0.6583	0.6377	0.6730	0.6762
Verbalize	0.5349	0.5262	0.5274	0.5261	0.5329	0.5091	0.5578	0.5306	0.5286	0.5352	0.5511	0.5279	0.5460	0.5150	0.5550	0.5370
InterrogateLLM	0.7009	0.7127	0.7062	0.7014	0.7496	0.6904	0.7435	0.7150	0.6978	0.6997	0.7248	0.7273	0.7438	0.6902	0.7376	0.7173
Training-based methods																
MM	0.5033	0.5228	0.5270	0.4910	0.5120	0.4966	0.5269	0.5114	0.4844	0.4817	0.5243	0.5240	0.5280	0.4848	0.5116	0.5055
SEP	0.5277	0.5040	0.5072	0.5170	0.5258	0.4980	0.5317	0.5159	0.5395	0.5139	0.5284	0.5222	0.5301	0.4874	0.5350	0.5224
SAPLMA	0.5109	0.5192	0.5226	0.5385	0.5287	0.4916	0.5370	0.5212	0.5198	0.5174	0.5268	0.5303	0.5240	0.4868	0.5392	0.5206
LapEigvals	0.4840	0.5049	0.5113	0.4969	0.5062	0.4822	0.5263	0.5017	0.4712	0.4891	0.5108	0.4829	0.5289	0.4727	0.5291	0.4978
Proposed method																
CHORD	**0.7720**	**0.8237**	**0.8143**	**0.8070**	**0.8387**	**0.7248**	**0.8722**	**0.8075**	**0.8069**	**0.8104**	**0.8333**	**0.8027**	**0.8406**	**0.7207**	**0.8472**	**0.8088**

	Ministral3-8B								Ministral3-14B							
Training-free methods																
Perplexity	0.6443	0.6328	0.6576	0.6397	0.6528	0.6308	0.6785	0.6481	0.6335	0.6510	0.6471	0.6357	0.6526	0.6029	0.6635	0.6409
Semantic Entropy	0.6828	0.6800	0.7148	0.6867	0.7045	0.6424	0.7338	0.6921	0.6882	0.6925	0.7006	0.6930	0.7104	0.6534	0.7283	0.6952
EigenScore	0.6856	0.6729	0.6918	0.6788	0.6836	0.6519	0.7076	0.6817	0.6668	0.7015	0.6846	0.6746	0.6926	0.6509	0.7090	0.6829
Lexical Similarity	0.6530	0.6474	0.7075	0.6906	0.6479	0.6265	0.6698	0.6632	0.6632	0.6595	0.6945	0.6788	0.6610	0.6255	0.6682	0.6644
Verbalize	0.5019	0.5116	0.5440	0.5172	0.5230	0.5112	0.5528	0.5231	0.5143	0.5340	0.5158	0.5075	0.5231	0.5048	0.5506	0.5214
InterrogateLLM	0.6963	0.7002	0.7030	0.6969	0.7254	0.6961	0.7446	0.7089	0.6989	0.7295	0.7097	0.6862	0.7163	0.6957	0.7432	0.7114
Training-based methods																
MM	0.4807	0.4840	0.5294	0.5123	0.5298	0.4853	0.5285	0.5071	0.4866	0.5095	0.5123	0.4854	0.5284	0.4924	0.5232	0.5054
SEP	0.5232	0.5188	0.5162	0.4990	0.5033	0.5030	0.5288	0.5132	0.4860	0.5076	0.5123	0.4999	0.5186	0.4895	0.5391	0.5076
SAPLMA	0.4816	0.5275	0.5396	0.5073	0.5335	0.4944	0.5392	0.5176	0.4865	0.5232	0.4952	0.5091	0.5266	0.4839	0.5386	0.5090
LapEigvals	0.4708	0.4737	0.5019	0.5031	0.4983	0.4733	0.5295	0.4929	0.4847	0.4766	0.5179	0.4800	0.5058	0.4702	0.5161	0.4930
Proposed method																
CHORD	**0.7663**	**0.8005**	**0.7914**	**0.7836**	**0.8021**	**0.7263**	**0.8474**	**0.7882**	**0.7760**	**0.7988**	**0.8046**	**0.7933**	**0.8193**	**0.7237**	**0.8557**	**0.7959**

3.2 From Token-Level to Joint Relational Structures

Building on the token-level relations in Section 3.1, and motivated by findings that LLMs process hierarchical semantics across layers and exhibit rich feature-level interactions [12, 13], we study three broader forms of relational structure: *cross-layer token relations*, *within-token feature relations*, and *joint relations* over layers, tokens, and features. All three follow Section 3.1's pipeline, differing only

in graph nodes and edge weights. We reuse $\mathbf{T}^{\text{sel}}$ and select a layer subset $\mathbf{L}^{\text{sel}}$ from the L_i layers of $\mathbb{P}_{\boldsymbol{\theta}_i}$, with $|\mathbf{L}^{\text{sel}}|$ shared across LLMs to keep graph size constant. For *within-token feature* and *joint relations*, the representation dimension is partitioned into B equal-size contiguous blocks, with $\mathcal{I}_b$ indexing the b-th block (Eq. (12)); each block serves as a node [42, 43]. Details in Appendix A.9.

Cross-Layer Token Relations. For each layer $\ell \in \mathbf{L}^{\text{sel}}$, we form a layer-level summary by stacking the hidden representations of all selected tokens at this layer into a matrix:

$$\mathbf{H}^{\ell,\text{sel}}_{\boldsymbol{\theta}_i}(\mathbf{Q},\mathbf{A}) = \left[\mathbf{h}^{\ell}_{\boldsymbol{\theta}_i,t_1}(\mathbf{Q},\mathbf{A}),\ldots,\mathbf{h}^{\ell}_{\boldsymbol{\theta}_i,t_r}(\mathbf{Q},\mathbf{A})\right]^{\top} \in \mathbb{R}^{r\times d_i}. \tag{6}$$

Then, we define the node for layer ℓ as $\text{vec}(\mathbf{H}^{\ell,\text{sel}}_{\boldsymbol{\theta}_i}) \in \mathbb{R}^{rd_i}$, with $\text{vec}(\cdot)$ flattening a matrix into a vector by stacking its rows. To define edges between layer nodes, we use the standard cosine distance for comparing Transformer-layer representations [44–46], modulated by a depth-decay factor:

$$R^{\text{lyr}}_i(\ell,\tilde{\ell};\mathbf{Q},\mathbf{A}) = s^{\text{lyr}}(\ell,\tilde{\ell})\left(1-\cos\left(\text{vec}(\mathbf{H}^{\ell,\text{sel}}_{\boldsymbol{\theta}_i}),\text{vec}(\mathbf{H}^{\tilde{\ell},\text{sel}}_{\boldsymbol{\theta}_i})\right)\right), \tag{7}$$

where $\cos(\cdot,\cdot)$ denotes cosine similarity, and $s^{\text{lyr}}(\ell,\tilde{\ell}) = \exp(-|\ell-\tilde{\ell}|/\tau_{\text{lyr}})$ is the depth-decay factor between layers ℓ and $\tilde{\ell}$. The bandwidth $\tau_{\text{lyr}} > 0$ controls relation decay with layer distance.

Within-Token Feature Relations. For each feature block $b \in \{1,\ldots,B\}$ and each selected layer $\ell \in \mathbf{L}^{\text{sel}}$, we first collect the b-th hidden-state block of all selected tokens at layer ℓ:

$$\mathbf{V}^{\ell}_{\boldsymbol{\theta}_i,b}(\mathbf{Q},\mathbf{A}) = \left[\mathbf{h}^{\ell}_{\boldsymbol{\theta}_i,t_1}(\mathbf{Q},\mathbf{A})[\mathcal{I}_b],\ldots,\mathbf{h}^{\ell}_{\boldsymbol{\theta}_i,t_r}(\mathbf{Q},\mathbf{A})[\mathcal{I}_b]\right]^{\top} \in \mathbb{R}^{r\times|\mathcal{I}_b|}. \tag{8}$$

Then, we stack these token-block matrices vertically over the selected layers $\mathbf{L}^{\text{sel}} = \{\ell_1,\ldots,\ell_{|\mathbf{L}^{\text{sel}}|}\}$:

$$\mathbf{V}_{\boldsymbol{\theta}_i,b}(\mathbf{Q},\mathbf{A}) = \left[(\mathbf{V}^{\ell_1}_{\boldsymbol{\theta}_i,b}(\mathbf{Q},\mathbf{A}))^{\top},\ldots,(\mathbf{V}^{\ell_{|\mathbf{L}^{\text{sel}}|}}_{\boldsymbol{\theta}_i,b}(\mathbf{Q},\mathbf{A}))^{\top}\right]^{\top} \in \mathbb{R}^{|\mathbf{L}^{\text{sel}}|r\times|\mathcal{I}_b|}. \tag{9}$$

The block node for b is then represented by $\text{vec}(\mathbf{V}_{\boldsymbol{\theta}_i,b}) \in \mathbb{R}^{|\mathbf{L}^{\text{sel}}|r|\mathcal{I}_b|}$. To compute edges between block nodes, we use the ℓ_2 distance: $R^{\text{feat}}_i(b,\tilde{b};\mathbf{Q},\mathbf{A}) = \left\|\text{vec}(\mathbf{V}_{\boldsymbol{\theta}_i,b}) - \text{vec}(\mathbf{V}_{\boldsymbol{\theta}_i,\tilde{b}})\right\|_2$.

Joint Relations. Finally, we define a fine-grained node for each (ℓ,t,b) triple, whose feature is the corresponding hidden-state block $\mathbf{h}^{\ell}_{\boldsymbol{\theta}_i,t}(\mathbf{Q},\mathbf{A})[\mathcal{I}_b] \in \mathbb{R}^{|\mathcal{I}_b|}$. This yields a fully connected graph over $|\mathbf{L}^{\text{sel}}|\,r\,B$ nodes, with the relation between two such nodes given by

$$R^{\text{joint}}_i\left((\ell,t,b),(\tilde{\ell},\tilde{t},\tilde{b});\mathbf{Q},\mathbf{A}\right) = s^{\text{lyr}}(\ell,\tilde{\ell})\,s^{\text{tok}}(t,\tilde{t})\left\|\mathbf{h}^{\ell}_{\boldsymbol{\theta}_i,t}(\mathbf{Q},\mathbf{A})[\mathcal{I}_b] - \mathbf{h}^{\tilde{\ell}}_{\boldsymbol{\theta}_i,\tilde{t}}(\mathbf{Q},\mathbf{A})[\mathcal{I}_{\tilde{b}}]\right\|_2, \tag{10}$$

where s^{lyr} and s^{tok} are the layer- and token-distance decay factors defined in Eq. (7) and Section 3.1.

For each relation type, following the same procedure as Section 3.1, we apply the same relation-weighted propagation and mean pooling as in Eq. (4) to obtain the per-sample feature vectors $\mathbf{z}^{\text{lyr}}_i(\mathbf{Q},\mathbf{A})$, $\mathbf{z}^{\text{feat}}_i(\mathbf{Q},\mathbf{A})$, and $\mathbf{z}^{\text{joint}}_i(\mathbf{Q},\mathbf{A})$, with full procedure in Appendix A.9.

Empirical Observations and Implications. We now have four relational features: $\mathbf{z}^{\text{tok},\ell}_i$, $\mathbf{z}^{\text{lyr}}_i$, $\mathbf{z}^{\text{feat}}_i$, and $\mathbf{z}^{\text{joint}}_i$. We follow the same setup of Section 3.1, and report the observations as follows.

(1) *Cross-model stability.* In Figure 2a, raw hidden representations yield mutual kNN alignments of 0.03–0.11 across the five transfer pairs, whereas the four relational features yield consistently higher alignments: token-level relations reach 0.17–0.30, joint relational structures 0.15–0.30, within-token feature relations 0.07–0.21, and cross-layer token relations 0.10–0.24. Following the same experimental setup in Section 3.1, the random-overlap baseline with $k = 10$ and $n = 5000$ is near 0.002, so the relational alignments are roughly 35 to 150 times random neighborhood agreement, indicating non-trivial cross-model relational similarity across all four features.

(2) *Cross-model transferability.* In Figure 2b and Figure 3, the four relational features improve cross-model AUROC over the raw hidden representations, with a different ordering. Joint relational structures achieve the highest AUROC, around 0.69–0.73, followed by within-token feature relations at 0.64–0.67, token-level relations at 0.62–0.67, and cross-layer token relations at 0.56–0.60. Comparing against the two training-free baselines used in Section 3.1, namely PPL (0.63) and Semantic Entropy (0.75), three of the four relational features already match or exceed the general-purpose PPL baseline, and the joint relational structures further close the gap to Semantic Entropy.

Implication. Among the four relational features, joint relational structures over layers, tokens, and feature blocks achieve near-highest cross-model stability and the highest cross-model AUROC. This indicates that the relations across all three indices, when combined, carry information that is both transferable across LLMs and discriminative for hallucination detection. Notably, this is achieved with a deliberately coarse design where the edge weights are fixed handcrafted decay functions. Such a simple construction already delivers non-trivial cross-model transfer. This leaves substantial room for a learnable, input-dependent edge mechanism, which we develop in Section 4.

4 CHORD: Relational Attention for Cross-Model Hallucination Detection

Building on the implication in Section 3, we develop CHORD, which builds a unified layer-token-feature graph with position- and content-aware edges, and learns transferable detection signals through relation-aware attention and meta-learning. See Appendices A.5, A.6, A.7 for more details.

Hidden Representation Restructuring. Different LLMs have different hidden dimensions d_i, so we first map each hidden representation $\mathbf{h}^{\ell}_{\boldsymbol{\theta}_i,t} \in \mathbb{R}^{d_i}$ into a unified space $\mathbb{R}^{d^\star}$ via scattered padding, where $d^\star \geq \max_i d_i$. Specifically, for each LLM $\mathbb{P}_{\boldsymbol{\theta}_i}$, we define an index map $p_i : \{1, \ldots, d_i\} \to \{1, \ldots, d^\star\}$ that distributes the d_i original coordinates evenly across the padded dimension [47],

$$p_i(\kappa) = \lfloor (\kappa - 1)\, d^\star / d_i \rfloor + 1, \text{ for } \kappa \in \{1, \ldots, d_i\}, \tag{11}$$

and obtain the padded representation $\bar{\mathbf{h}}^{\ell}_t \in \mathbb{R}^{d^\star}$ by $\bar{\mathbf{h}}^{\ell}_t[p_i(\kappa)] = \mathbf{h}^{\ell}_{\boldsymbol{\theta}_i,t}[\kappa]$ and $\bar{\mathbf{h}}^{\ell}_t[j] = 0$, for $j \notin \mathrm{Im}(p_i)$, where $\mathrm{Im}(p_i)$ denotes the set of positions reached by p_i. To prepare the unified vector for graph construction below, we partition the padded dimension into B equal-size contiguous blocks:

$$\mathcal{I}_b = \big\{(b-1)\, d^\star/B + 1,\ \ldots,\ b\, d^\star/B\big\}, \text{ for } b \in \{1, \ldots, B\}. \tag{12}$$

We then use $\bar{\mathbf{h}}^{\ell}_{\boldsymbol{\theta}_i,t}[\mathcal{I}_b] \in \mathbb{R}^{d^\star/B}$ to represent the feature of the b-th block.

Graph Construction and Relation-Aware Encoding. For a given sample generated by a specific LLM $\mathbb{P}_{\boldsymbol{\theta}_i}$, we construct an individual relation graph. Within this graph, we treat each (ℓ, t, b) triple with $\ell \in \mathbf{L}^{\mathrm{sel}}$, $t \in \mathbf{T}^{\mathrm{sel}}$, and $b \in \{1, \ldots, B\}$ as a node, indexed by $u \in \{1, \ldots, N\}$ with $N = |\mathbf{L}^{\mathrm{sel}}| \cdot r \cdot B$. We write (ℓ_u, t_u, b_u) for the layer, token, and block indices of u; the resulting graph is fully connected over the N nodes. Each node u takes the corresponding block feature $\bar{\mathbf{h}}^{\ell_u}_{t_u}[\mathcal{I}_{b_u}]$ and combines it with these three *learnable position embeddings* of its layer, token, and block indices to form the initial node representation $\mathbf{x}_u = \mathbf{W}_{\mathrm{p}}\, \bar{\mathbf{h}}^{\ell_u}_{t_u}[\mathcal{I}_{b_u}] + \mathbf{e}_{\ell}(\ell_u) + \mathbf{e}_t(t_u) + \mathbf{e}_b(b_u) \in \mathbb{R}^{d_{\mathrm{m}}}$, where $\mathbf{W}_{\mathrm{p}} \in \mathbb{R}^{d_{\mathrm{m}} \times d^\star/B}$ is a learnable matrix that linearly projects the block feature to the detector's internal dimension d_{m}, and $\mathbf{e}_{\ell}, \mathbf{e}_t, \mathbf{e}_b$ are learnable lookup tables [48] that map indices in $\{1, \ldots, |\mathbf{L}^{\mathrm{sel}}|\}, \{1, \ldots, r\}, \{1, \ldots, B\}$, respectively, to dense continuous vectors in $\mathbb{R}^{d_{\mathrm{m}}}$.

For each pair of nodes (u, v), we let $\boldsymbol{\phi}_{\ell}, \boldsymbol{\phi}_t, \boldsymbol{\phi}_b$ be learnable lookup tables that map the layer, token, and block index differences $\ell_u - \ell_v, t_u - t_v, b_u - b_v$ to vectors, and define an edge feature that conveys to the attention both how the two nodes differ in position and how similar they are in content,

$$\mathbf{e}_{uv} = \mathrm{MLP}_1\big(\big[\boldsymbol{\phi}_{\ell}(\ell_u - \ell_v),\ \boldsymbol{\phi}_t(t_u - t_v),\ \boldsymbol{\phi}_b(b_u - b_v),\ c_{uv},\ d_{uv}\big]\big) \in \mathbb{R}^{d_{\mathrm{m}}}, \tag{13}$$

where $c_{uv} = \cos(\mathbf{x}_u, \mathbf{x}_v)$, $d_{uv} = \|\mathbf{x}_u - \mathbf{x}_v\|_2$, and MLP_1 is a learnable MLP that fuses these signals.

At this point, we have a fully connected graph in which each node carries layer-token-block-aware features and each edge carries learned positional and content relations. To turn this graph into a single feature vector for hallucination detection, we proceed in two steps that mirror the relational findings in Section 3.2. *First*, we propagate messages along the edges so that each node aggregates relational information from every other node, absorbing the edge features into updated node representations [36, 49]. This is exactly where the edge feature $\mathbf{e}_{uv}$ from Eq. (13) earns its purpose: by participating in the propagation, the positional and content relations encoded on each edge are written into the receiving node, so subsequent stages no longer need to access edges directly. We instantiate this propagation with a learnable relation-aware attention encoder [50]: $\{\mathbf{x}'_u\}_{u=1}^{N} = \mathrm{RA}\big(\{\mathbf{x}_u\}_{u=1}^{N}, \{\mathbf{e}_{uv}\}_{u,v=1}^{N}\big)$. *Second*, with relations now stored inside the nodes, we collapse the N updated nodes into a single graph-level vector via attention pooling [51]: $\mathbf{z} = \mathrm{AttPool}\big(\{\mathbf{x}'_u\}_{u=1}^{N}\big) \in \mathbb{R}^{d_{\mathrm{m}}}$. *Last*, the truthfulness probability is given by $\hat{y} = \mathrm{Sigmoid}\big(\mathrm{MLP}_2(\mathbf{z})\big) \in [0, 1]$, where MLP_2 is a small learnable MLP.

Meta-Learning Training. We view the preceding relational pipeline as a parameterized detector $\hat{y} = D_{\Theta}(\mathbf{Q}, \mathbf{A}; \mathbb{P}_{\boldsymbol{\theta}}) \in [0, 1]$, where Θ collects all learnable parameters. Let ℓ^{bce} denote the binary

cross-entropy loss between $\hat{y}$ and the truthfulness label y, and recall that $\mathcal{D}_i$ (defined in Eq. (1)) collects the labeled $(\mathbf{Q}, \mathbf{A}, y)$ triples produced by the i-th training LLM. A straightforward training choice would solve $\arg\min_\Theta \sum_{i=1}^{M} \sum_{(\mathbf{Q},\mathbf{A},y)\in\mathcal{D}_i} \ell^{\text{bce}}(\hat{y}; y)$ on the training datasets, but our actual goal is to make Θ generalize to LLMs unseen at training time, and such empirical risk minimization may exploit shortcuts specific to individual training LLMs. We therefore train Θ by following Feature-Critic [14], a representative meta-learning strategy, abstracted as a procedure that takes the parameterized detector and the training datasets as input and returns updated parameters, $\Theta' \leftarrow \text{FeatCrit}(D_\Theta, \{\mathcal{D}_i\}_{i=1}^{M})$. The intuition is to manufacture the missing test condition from the training data itself: at each step, FeatCrit holds out one training LLM as a pseudo-unseen model, runs candidate updates on the remaining LLMs, and rewards updates whose effect transfers to the held-out LLM. Over training, this drives Θ' toward signals shared across LLMs rather than those idiosyncratic to any one of them. The trained detector $D_{\Theta'}$ is then used at test time on unseen LLMs.

Table 2: Ablation study of CHORD (AUROC). The comparison protocol follows Table 1. In the *Best training-free* and *Best training-based* rows, each benchmark column reports the strongest baseline of that group on that benchmark, so different columns may come from different baselines; the QA Avg. and All Avg. columns report the averages of the single strongest baseline of the group, and are therefore not the row-wise means of the columns beside them. Full results are provided in Appendix G. Best results are **bold**, and second-best results are underlined.

Method	Target Model: Llama3.2-3B					Llama3.1-8B					Llama3.3-70B				
	Mean AUROC↑					Mean AUROC↑					Mean AUROC↑				
	QA Avg.	GSM8K	MBPP	xLAM	**All Avg.**	QA Avg.	GSM8K	MBPP	xLAM	**All Avg.**	QA Avg.	GSM8K	MBPP	xLAM	**All Avg.**
Best training-free	0.7031	0.7177	0.6915	0.7253	0.7067	0.7119	0.7293	0.6865	0.7469	0.7158	0.7372	0.7473	0.7181	0.7477	0.7374
Best training-based	0.5209	0.5300	0.5035	0.5397	0.5208	0.5225	0.5382	0.4876	0.5395	0.5212	0.5278	0.5375	0.4905	0.5386	0.5232
CHORD w/o RelAttn	0.6901	0.7241	0.6458	0.7259	0.6937	0.7023	0.7403	0.6678	0.7916	0.7156	0.7373	0.7646	0.7017	0.7821	0.7425
CHORD w/o Meta	0.7102	0.7394	0.6563	0.7594	0.7137	0.7278	0.7538	0.6693	0.7975	0.7331	0.7571	0.8101	0.7151	0.7886	0.7632
CHORD	**0.7901**	**0.8193**	**0.7351**	**0.8437**	**0.7941**	**0.8017**	**0.8362**	**0.7307**	**0.8650**	**0.8056**	**0.8302**	**0.8568**	**0.7605**	**0.8713**	**0.8299**
	Qwen3-8B					Qwen3-14B					Qwen3.5-35B				
Best training-free	0.7061	0.7143	0.6834	0.7483	0.7092	0.7053	0.7496	0.6904	0.7435	0.7150	0.7124	0.7438	0.6902	0.7376	0.7173
Best training-based	0.5077	0.5167	0.4922	0.5384	0.5112	0.5228	0.5287	0.4980	0.5370	0.5212	0.5260	0.5301	0.4874	0.5392	0.5224
CHORD w/o RelAttn	0.6965	0.7233	0.6416	0.7602	0.7016	0.6831	0.7645	0.6652	0.7594	0.7031	0.7096	0.7609	0.6550	0.7640	0.7169
CHORD w/o Meta	0.7108	0.7444	0.6577	0.7706	0.7166	0.7146	0.7679	0.6913	0.7788	0.7281	0.7253	0.7661	0.6515	0.7641	0.7262
CHORD	**0.8005**	**0.8256**	**0.7481**	**0.8484**	**0.8034**	**0.8043**	**0.8387**	**0.7248**	**0.8722**	**0.8075**	**0.8133**	**0.8406**	**0.7207**	**0.8472**	**0.8088**
	Ministral3-8B					Ministral3-14B					**Target Avg.**				
Best training-free	0.6991	0.7254	0.6961	0.7446	0.7089	0.7061	0.7163	0.6957	0.7432	0.7114	0.7102	0.7297	0.6940	0.7421	0.7152
Best training-based	0.5143	0.5335	0.5030	0.5392	0.5176	0.5035	0.5284	0.4924	0.5391	0.5090	0.5173	0.5276	0.4927	0.5387	0.5177
CHORD w/o RelAttn	0.6855	0.7148	0.6591	0.7514	0.6953	0.6988	0.6829	0.6463	0.7328	0.6939	0.7004	0.7344	0.6603	0.7584	0.7078
CHORD w/o Meta	0.7024	0.7574	0.6821	0.7738	0.7176	0.7222	0.7186	0.6456	0.7594	0.7160	0.7213	0.7572	0.6711	0.7740	0.7268
CHORD	**0.7854**	**0.8021**	**0.7263**	**0.8474**	**0.7882**	**0.7932**	**0.8193**	**0.7237**	**0.8557**	**0.7959**	**0.8023**	**0.8298**	**0.7337**	**0.8564**	**0.8042**

5 Experiments

Experimental Setups. We evaluate CHORD on eight open-source LLMs from the Llama, Qwen, and Ministral families [15–17], spanning different scales. We use seven benchmarks: TruthfulQA, HaluEval, TriviaQA, CoQA, GSM8K, MBPP, and xLAM [19–25], covering factual QA, hallucination detection, conversational QA, reasoning, coding, and agent tasks. Baselines include Perplexity, Semantic Entropy, EigenScore, Lexical Similarity, Verbalize, InterrogateLLM, MM, SEP, SAPLMA, and LapEigvals [5, 6, 11, 41, 52–57]. For training-based methods requiring fixed dimensions, we use the restructuring procedure in Section 4 to map hidden states into a common space. Sampling strategies, model details, baseline configurations, extraction details, and hyperparameters are in Appendix A.

Overall Cross-model Hallucination Detection Performance. Table 1 evaluates transfer to a held-out model family, with no target-family examples used for detector training. Averaged equally over the eight target models and seven benchmarks, CHORD reaches an AUROC of 0.8042, compared with 0.7152 for InterrogateLLM, the strongest baseline by overall average. It also ranks first in all 56 model–benchmark evaluations, so the aggregate gain is not driven by a few favorable targets. In contrast, MM, SEP, SAPLMA, and LapEigvals obtain overall averages of 0.4976–0.5177. Their weak transfer despite dimension-compatible inputs shows that matching feature dimensions alone does not resolve cross-model generalization. Training-free methods avoid fitting a source-specific detector and remain competitive, but CHORD demonstrates that source supervision can provide additional value when combined with relational features and transfer-oriented training.

Dataset-level Differences. The gains are not uniform across tasks. On xLAM and GSM8K, CHORD achieves target-averaged AUROCs of 0.8564 and 0.8298, compared with 0.7421 and 0.7297 for InterrogateLLM. The advantage also extends to standard QA: its average over TruthfulQA, HaluEval,

TriviaQA, and CoQA is 0.8023, versus 0.7102 for the same baseline. Thus, the benefit is not confined to reasoning or tool-use outputs. MBPP presents a clearer boundary: CHORD reaches 0.7337, but its margin over InterrogateLLM (0.6940) is much smaller. Limited supervision and execution-dependent code correctness may contribute to this gap, although these results do not isolate their effects. The task differences suggest that relational features provide broadly useful evidence, but should not be assumed to capture all forms of answer correctness equally well.

Variation across Target Models. The advantage extends beyond the largest targets. Across eight models in Table 1, the benchmark-averaged margin over InterrogateLLM ranges from 0.0793 to 0.0942. Within Llama, CHORD improves from 0.7941 on 3B to 0.8299 on 70B, while the Qwen targets occupy a narrower range of 0.8034–0.8088. Thus, larger targets often provide stronger detection performance, but the magnitude of this change varies across families. Since model versions and architectures change, these results describe the evaluated targets rather than establish a general scaling law.

Ablation Study. Table 2 separates the contributions of relation-aware attention and meta-learning. Removing relation-aware attention reduces overall AUROC from 0.8042 to 0.7078, while replacing meta-learning with standard ERM reduces it to 0.7268. These are drops of 9.64 and 7.74 AUROC percentage points, respectively. The first variant retains the remaining framework, indicating that the gain cannot be attributed solely to hidden-state restructuring or graph-node construction. The second retains the relational architecture, showing that source-family supervision alone does not recover the full transfer benefit. Both variants remain above ordinary training-based baselines, but their advantage over training-free detection is task-dependent. On MBPP, their averages of 0.6603 and 0.6711 fall below InterrogateLLM's 0.6940, whereas full CHORD reaches 0.7337. This contrast supports using both components rather than relying on either modification alone.

Contributions of Individual Relation Channels. The finer-grained ablations in Appendix H.2 remove one edge channel while preserving the other channels and training procedure. Removing token, layer, or intra-block feature edges reduces overall AUROC to 0.7592, 0.7629, or 0.7607, respectively. Their relative importance varies by task: token-edge removal produces the largest loss among the three variants on xLAM, layer-edge removal does so on GSM8K, and intra-block feature-edge removal does so on MBPP. Thus, no single channel accounts for the best performance across all tasks. Each removal is also less damaging than removing relation-aware attention entirely, consistent with the remaining channels retaining useful detection evidence.

Transfer across Model Scales. Appendix H.1 evaluates six directed transfers between small, medium, and large model groups. CHORD ranks first in each direction, with average AUROC ranging from 0.7664 to 0.7854. Large-to-small transfer reaches 0.7854, compared with 0.7664 in the reverse direction, showing that transfer difficulty can be asymmetric. However, these groups differ in model-family composition and source-model count. The results therefore support transfer across the evaluated scale groups, rather than a causal claim that larger source models always provide better supervision.

Distinguishing Model Shift from Benchmark Shift. Appendix H.3 holds the LLM fixed, trains on six benchmarks, and evaluates on the seventh. CHORD reaches an overall AUROC of 0.7800, compared with 0.7152 for InterrogateLLM. More importantly, ordinary training-based baselines recover to 0.6366–0.6797, rather than remaining near chance as in cross-family transfer. This contrast indicates that their weakness in Table 1 is not a general inability to learn hallucination signals: they retain useful discrimination when the model's representation space is shared across training and testing. Because the training pools also differ, the comparison supports rather than proves the representation-mismatch explanation. These experiments test two distinct shifts and should not be read as evidence for simultaneous transfer to an unseen model and an unseen benchmark.

6 Conclusion

In this paper, we study *cross-model hallucination detection* (CMHD), which aims to train hallucination detectors on source LLMs and generalize them to unseen target LLMs. Through cross-model analysis, we identify layer-token-feature relational structures as transferable hallucination signals. We propose CHORD, a relational graph discrimination framework that learns structure-aware features through relation-aware attention and meta-learning. Experiments across model families, scales, and benchmarks show CHORD outperforms representative baselines, while ablations verify core modules.

Acknowledgments and Disclosure of Funding

This work was supported in part by the 2026 KTP Seed Funding from the Faculty of Engineering and Information Technology, University of Technology Sydney, and by the Australian Government through the Australian Research Council (ARC) under a Discovery Early Career Researcher Award (DE250100363) and a Future Fellowship (FT230100121).

The views and conclusions in this paper reflect the authors' work and should not be taken as official positions of the Australian Government, the ARC, or any other funding body.

The authors declare no competing interests.

References

[1] Emma Harvey, Allison Koenecke, and Rene F. Kizilcec. "don't forget the teachers": Towards an educator-centered understanding of harms from large language models in education. In *CHI*, 2025.

[2] Dimitri Roustan, François Bastardot, et al. The clinicians' guide to large language models: A general perspective with a focus on hallucinations. *Interactive Journal of Medical Research*, 2025.

[3] Haoqiang Kang and Xiao-Yang Liu. Deficiency of large language models in finance: An empirical examination of hallucination. *arXiv*, 2023.

[4] Potsawee Manakul, Adian Liusie, and Mark Gales. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In *Proc. of EMNLP*, pages 9004–9017, 2023.

[5] Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, et al. Detecting hallucinations in large language models using semantic entropy. *Nature*, 2024.

[6] Jannik Kossen, Jiatong Han, Muhammed Razzak, et al. Semantic entropy probes: Robust and cheap hallucination detection in LLMs. *arXiv*, 2024.

[7] Hongliang Wei, Xingtao Wang, Xianqi Zhang, et al. Toward a stable, fair, and comprehensive evaluation of object hallucination in large vision-language models. In *Proc. of NeurIPS*, 2024.

[8] Lei Huang, Weijiang Yu, Weitao Ma, et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, pages 1–55, 2025.

[9] Pranab Sahoo, Prabhash Meharia, Akash Ghosh, et al. A comprehensive survey of hallucination in large language, image, video and audio foundation models. *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 11709–11724, 2024.

[10] Yue Zhang, Yafu Li, Leyang Cui, et al. Siren's song in the ai ocean: A survey on hallucination in large language models. *Computational Linguistics*, pages 1373–1418, 2025.

[11] Jakub Binkowski, Denis Janiak, Albert Sawczyn, et al. Hallucination detection in LLMs using spectral features of attention maps. In *EMNLP*, 2025.

[12] Ian Tenney, Dipanjan Das, and Ellie Pavlick. BERT rediscovers the classical NLP pipeline. In *Proc. of ACL*, pages 4593–4601, 2019.

[13] Nelson Elhage, Neel Nanda, Catherine Olsson, et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. URL https://transformer-circuits.pub/2021/framework/index.html. Published: Dec 22, 2021.

[14] Yiying Li, Yongxin Yang, Wei Zhou, et al. Feature-critic networks for heterogeneous domain generalization. In *Proc. of ICML*, pages 3915–3924, 2019.

[15] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. The llama 3 herd of models. *ArXiv preprint*, 2024.

[16] Alexander H Liu, Kartik Khandelwal, Sandeep Subramanian, et al. Ministral 3. *ArXiv preprint*, 2026.

[17] An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 technical report. *ArXiv preprint*, 2025.

[18] Qwen Team. Qwen3.5: Accelerating productivity with native multimodal agents, 2026. URL https://qwen.ai/blog?id=qwen3.5.

[19] Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring how models mimic human falsehoods. In *Proc. of ACL*, pages 3214–3252, 2022.

[20] Junyi Li, Xiaoxue Cheng, Xin Zhao, et al. HaluEval: A large-scale hallucination evaluation benchmark for large language models. In *Proc. of EMNLP*, pages 6449–6464, 2023.

[21] Mandar Joshi, Eunsol Choi, Daniel Weld, et al. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proc. of ACL*, pages 1601–1611, 2017.

[22] Siva Reddy, Danqi Chen, and Christopher D. Manning. CoQA: A conversational question answering challenge. *TACL*, pages 249–266, 2019.

[23] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, et al. Training verifiers to solve math word problems. *ArXiv preprint*, 2021.

[24] Jacob Austin, Augustus Odena, Maxwell Nye, et al. Program synthesis with large language models. *ArXiv preprint*, 2021.

[25] Zuxin Liu, Thai Hoang, Jianguo Zhang, et al. Apigen: Automated pipeline for generating verifiable and diverse function-calling datasets. In *Proc. of NeurIPS*, 2024.

[26] Xuefeng Du, Chaowei Xiao, and Sharon Li. Haloscope: Harnessing unlabeled LLM generations for hallucination detection. In *Proc. of NeurIPS*, 2024.

[27] Changdae Oh, Zhen Fang, Shawn Im, et al. Understanding multimodal LLMs under distribution shifts: An information-theoretic approach. In *ICML*, 2025.

[28] Yongxin Deng, Zhen Fang, Sharon Li, et al. Beyond in-domain detection: Spikescore for cross-domain hallucination detection. In *Proc. of ICLR*, 2026.

[29] Simon Kornblith, Mohammad Norouzi, Honglak Lee, et al. Similarity of neural network representations revisited. In *Proc. of ICML*, pages 3519–3529, 2019.

[30] Max Klabunde, Tobias Schumacher, Markus Strohmaier, et al. Similarity of neural network models: A survey of functional and representational measures. *ACM Computing Surveys*, pages 1–52, 2025.

[31] Youcheng Huang, Chen Huang, Duanyu Feng, et al. Cross-model transferability among large language models on the platonic representations of concepts. In *Proc. of ACL*, pages 3686–3704, 2025.

[32] Yue Wang, Yongbin Sun, Ziwei Liu, et al. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (tog)*, pages 1–12, 2019.

[33] Yichen Tang, Weihang Su, Yujia Zhou, et al. Augmenting multi-agent communication with state delta trajectory. In *Proc. of EMNLP*, pages 10230–10251, 2025.

[34] Yubo Zhu, Dongrui Liu, Zecheng Lin, et al. The llm already knows: Estimating llm-perceived question difficulty via hidden representations. In *Proc. of EMNLP*, pages 1160–1176, 2025.

[35] Yao Xu, Mingyu Xu, Fangyu Lei, et al. Amplify adjacent token differences: Enhancing long chain-of-thought reasoning with shift-ffn. *ArXiv preprint*, 2025.

[36] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, et al. Neural message passing for quantum chemistry. In *Proc. of ICML*, pages 1263–1272, 2017.

[37] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Proc. of ICLR*, 2017.

[38] Minyoung Huh, Brian Cheung, Tongzhou Wang, et al. Position: The platonic representation hypothesis. In *Proc. of ICML*, 2024.

[39] Shaul Oron, Tali Dekel, Tianfan Xue, et al. Best-buddies similarity—robust template matching using mutual nearest neighbors. *IEEE transactions on pattern analysis and machine intelligence*, pages 1799–1813, 2017.

[40] Young-Jin Park, Hao Wang, Shervin Ardeshir, et al. Quantifying representation reliability in self-supervised learning models. *ArXiv preprint*, 2023.

[41] Jie Ren, Jiaming Luo, Yao Zhao, et al. Out-of-distribution detection and selective generation for conditional language models. In *Proc. of ICLR*, 2023.

[42] Keyulu Xu, Weihua Hu, Jure Leskovec, et al. How powerful are graph neural networks? In *Proc. of ICLR*, 2019.

[43] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proc. of ICLR*, 2021.

[44] Jason Phang, Haokun Liu, and Samuel R. Bowman. Fine-tuned transformers show clusters of similar representations across layers. In *Proceedings of the Fourth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 529–538, 2021.

[45] Elena Voita, Rico Sennrich, and Ivan Titov. The bottom-up evolution of representations in the transformer: A study with machine translation and language modeling objectives. In *Proc. of EMNLP*, pages 4396–4406, 2019.

[46] Hassan Sajjad, Fahim Dalvi, Nadir Durrani, et al. On the effect of dropping layers of pre-trained transformer models. *Computer Speech & Language*, page 101429, 2023.

[47] Alan V Oppenheim. *Discrete-time signal processing*. Pearson Education India, 1999.

[48] Jacob Devlin, Ming-Wei Chang, Kenton Lee, et al. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. of NAACL*, pages 4171–4186, 2019.

[49] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, et al. Graph attention networks. In *Proc. of ICLR*, 2018.

[50] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In *Proc. of NAACL*, pages 464–468, 2018.

[51] Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, et al. A structured self-attentive sentence embedding. In *Proc. of ICLR*, 2017.

[52] Chao Chen, Kai Liu, Ze Chen, et al. INSIDE: llms' internal states retain the power of hallucination detection. In *Proc. of ICLR*, 2024.

[53] Zhen Lin, Shubhendu Trivedi, and Jimeng Sun. Generating with confidence: Uncertainty quantification for black-box large language models. *TMLR*, 2024.

[54] Stephanie Lin, Jacob Hilton, and Owain Evans. Teaching models to express their uncertainty in words. *TMLR*, 2022.

[55] Yakir Yehuda, Itzik Malkiel, Oren Barkan, et al. InterrogateLLM: Zero-resource hallucination detection in LLM-generated answers. In *ACL*, 2024.

[56] Samuel Marks and Max Tegmark. The geometry of truth: Emergent linear structure in large language model representations of true/false datasets. *arXiv*, 2023.

[57] Amos Azaria and Tom Mitchell. The internal state of an LLM knows when it's lying. In *Proc. of EMNLP Findings*, pages 967–976, 2023.

[58] Ziwei Ji, Nayeon Lee, Rita Frieske, et al. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 2023.

[59] Shervin Minaee, Tomas Mikolov, Narjes Nikzad, et al. Large language models: A survey. *arXiv*, 2024.

[60] Mohammad Beigi, Ying Shen, Runing Yang, et al. InternalInspector i^2: Robust confidence estimation in LLMs through internal states. In *EMNLP Findings*, 2024.

[61] Zhen Fang, Yixuan Li, Jie Lu, et al. Is out-of-distribution detection learnable? In *Proc. of NeurIPS*, 2022.

[62] Zhen Fang, Jie Lu, and Guangquan Zhang. Out-of-distribution detection with non-semantic exploration. *Information Sciences*, page 121989, 2025.

Appendix Contents

A Experimental Setup Details and Hyperparameter Selection **16**

A.1 Pilot Transfer-failure Analysis 16

A.2 Token–Layer Window Size and Position 16

A.3 Feature-Block Granularity 17

A.4 Why We Use Mutual k-Nearest-Neighbor Alignment 19

A.5 Graph Construction Details 20

A.6 Details of Relation-Aware Attention 22

A.7 Details of Meta-Learning Training 24

A.8 Details for the Cross-Model Alignment, Transfer, and Score-Distribution Diagnostics 26

A.9 Observation Pipeline for Relational Features 27

A.10 Baseline Implementation Details 29

A.11 LLM-as-Judge Labeling Protocol 32

A.12 Dataset Details and Preprocessing 33

B Related Work **37**

C Limitations **37**

D Computational Infrastructure **38**

E Licenses for Existing Assets **38**

F Observation Analysis Details **38**

F.1 Alignment and Transfer Are Related but Decoupled 38

F.2 Handcrafted Relation Priors Help but Saturate 43

F.3 Relation-Channel Importance Is Input-Dependent 46

F.4 Additional Robustness and Transfer Diagnostics 49

G Additional Experimental Analysis **49**

H Experimental Extension **52**

H.1 Scale-Group Transfer 52

H.2 Fine-Grained Edge Ablation 54

H.3 Cross-Benchmark Transfer 56

A Experimental Setup Details and Hyperparameter Selection

A.1 Pilot Transfer-failure Analysis

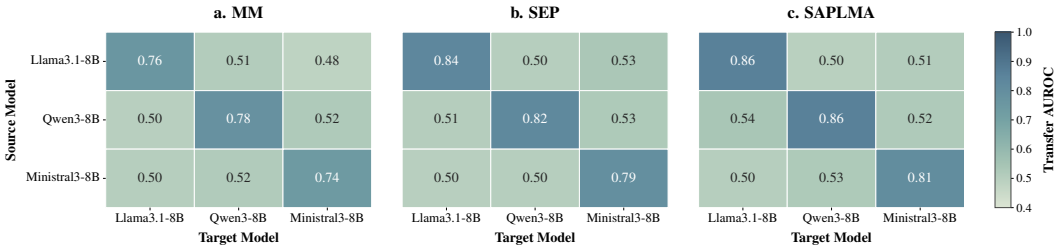

Figure 4: Pilot analysis of cross-model transfer failure in hallucination detection. **(a-c)** Transfer AUROC matrices for representative training-based methods (MM [56], SEP [6], SAPLMA [57]). While all methods achieve strong in-model performance (diagonal), they suffer severe degradation when across models (off-diagonal), dropping to near-random accuracy. See Appendix A.1 for details.

Figure 4 reports a small pilot study used to verify whether standard training-based hallucination detectors transfer across LLMs. We consider three 8B-scale models, Llama3.1-8B, Qwen3-8B, and Ministral3-8B, whose hidden-state dimensions are the same in this pilot; therefore, no cross-model dimension mapping is applied. For each model, we construct a mixed benchmark pool from TruthfulQA [19], HaluEval [20], TriviaQA [21], CoQA [22], GSM8K [23], MBPP [24], and xLAM [25]. We sample 1,000 examples from each benchmark whenever possible, and use all available examples for MBPP because it contains fewer than 1,000 usable samples in our setting. The mixed pool is split into 20% training and 80% evaluation data. For each source model, we train MM, SEP, and SAPLMA on its training split and evaluate the trained detector on the evaluation splits of all target models. The diagonal entries therefore measure in-model performance, while the off-diagonal entries measure direct cross-model transfer. All three methods achieve strong in-model AUROC on the diagonal, but their off-diagonal AUROC drops close to random guessing. This pilot result motivates the main problem studied in this paper: supervised detectors can learn useful model-specific decision rules, but these rules do not transfer reliably across heterogeneous LLM representation spaces.

A.2 Token–Layer Window Size and Position

The relational features studied in Section 3 depend on which generated tokens and Transformer layers are selected. The same choice also affects CHORD, since its graph is constructed over selected layer–token–block nodes. We therefore study two aspects of this design: the number of selected tokens/layers and the position of the selected window. Throughout this section, token windows are selected from generated answer tokens, and layer windows are selected from Transformer layers.

Evaluation settings. We consider two settings. The first is the fixed *observational diagnostic* setting used in Appendix F.1: joint relational features are constructed with hand-designed edges, an MLP detector is trained on a source model, and transfer AUROC is evaluated on a target model. The second is the *method* setting: the full CHORD detector is trained and evaluated under the leave-one-family-out protocol used in the main experiments. These two settings use the same kind of token–layer selection, but they serve different purposes: the diagnostic setting exposes transferable relational structure, while the method setting measures the end-to-end detector.

Count sweeps. We first vary the number of selected tokens and layers while keeping their positions fixed. Following common practice that late generated tokens contain more answer-level evidence and middle layers provide more transferable semantic structure, we use late token windows and middle layer windows for the count sweeps. For the observational diagnostic, the token-count sweep varies $T \in \{1, \ldots, 10\}$ with $L = 5$ fixed, and the layer-count sweep varies $L \in \{1, \ldots, 16\}$ with $T = 7$ fixed. For CHORD, we use the same sweep ranges, while fixing the other axis at the method default.

Figure 5 shows a clear difference between the fixed diagnostic features and the full method. For the observational diagnostic, using only one token or one layer leaves little relational structure, and

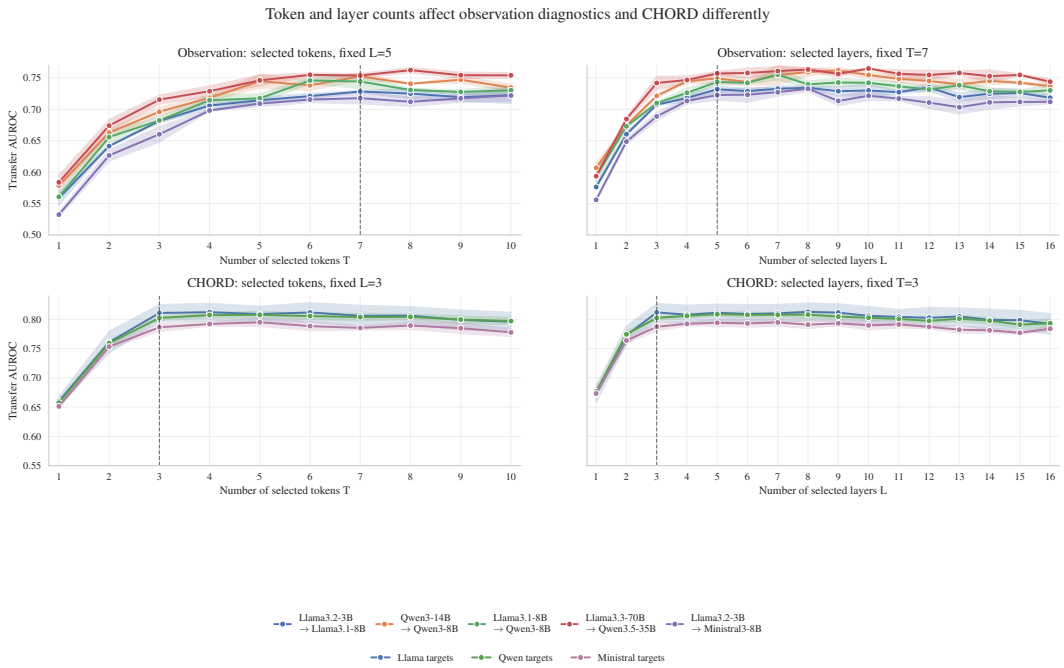

Figure 5: Token and layer count sweeps for the observational diagnostic and the full CHORD method. The diagnostic setting benefits from a moderately larger window, reaching its strongest region around $T = 7$ and $L = 5$. CHORD recovers most of its performance with a compact $T = 3, L = 3$ setting, after which larger windows provide only limited additional gain.

transfer AUROC is low. Performance improves as the window grows and is strongest around $T = 7$ and $L = 5$, after which the curve largely saturates. For CHORD, the curve rises more quickly and reaches a stable region around $T = 3, L = 3$. This suggests that fixed handcrafted relational features benefit from a broader window, whereas the learnable relation-aware encoder can extract most of the useful signal from a smaller graph.

Position sweeps. We next fix the window size and vary its position. For the observational diagnostic, we use the stronger count-sweep setting $T = 7, L = 5$. For CHORD, we use the default method setting $T = 3, L = 3$. When comparing token positions, we keep the layer window in the middle layers and place the token window at the early, middle, or late part of the generated answer. When comparing layer positions, we keep the token window at the late generated tokens and place the layer window at early, middle, or late layers.

Figure 6 shows that late generated tokens are more useful than early or middle generated tokens. This is consistent with the fact that hallucination evidence often becomes clearer after more of the answer has been generated. For layers, middle-layer windows perform best or near-best. Early layers are closer to lexical encoding, while very late layers can be more tied to model-specific output behavior. Middle layers provide a more stable region for relational transfer.

Resulting choices. Based on these sweeps, the observational diagnostics use $T = 7$ late generated tokens and $L = 5$ middle layers. The full CHORD method uses the more compact $T = 3, L = 3$ setting with the same late-token and middle-layer positions. This gives the diagnostics enough context to reveal transferable relational structure, while keeping the final method efficient and avoiding unnecessary graph expansion.

A.3 Feature-Block Granularity

Besides the token and layer windows, the joint relational graph also depends on how the hidden dimension is partitioned into feature blocks. In the main experiments, hidden states from all LLMs are first mapped to a common padded dimension $d_{\max} = 8192$, and this dimension is divided into

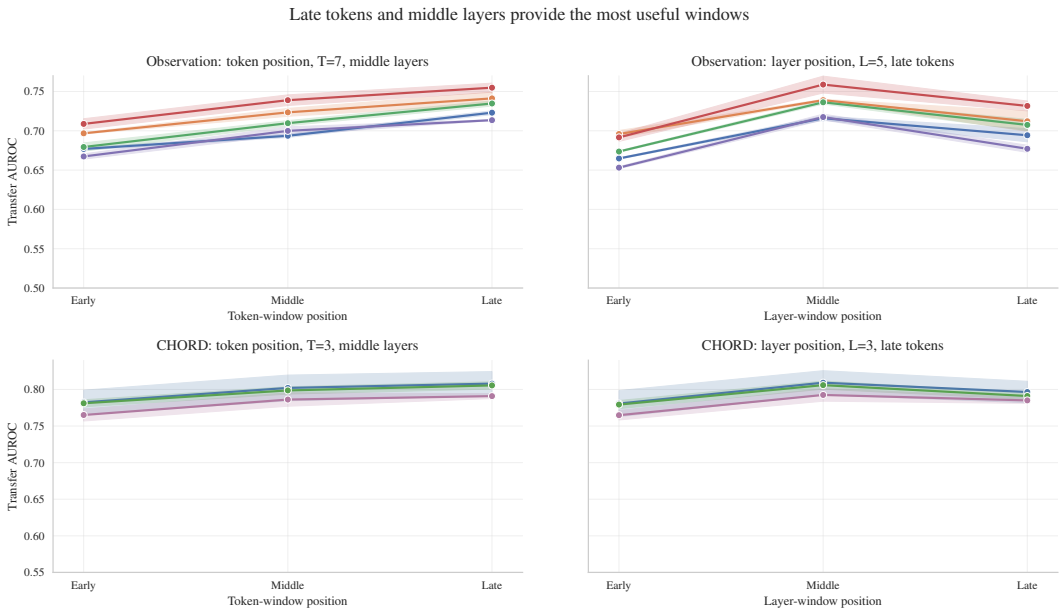

Figure 6: Token and layer position sweeps. Late generated tokens perform best across both the observational diagnostic and CHORD. Middle layers are also consistently strong, supporting the use of late-token and middle-layer windows in the main experiments.

$B = 64$ contiguous feature blocks. Each block then serves as the feature component of a joint node. This appendix studies the effect of varying the block count B.

Block construction. For a given value of B, we partition the padded feature indices into B disjoint contiguous blocks:

$$\{1, \dots, d_{\max}\} = \mathcal{I}_1^{(B)} \cup \cdots \cup \mathcal{I}_B^{(B)}, \qquad \mathcal{I}_b^{(B)} \cap \mathcal{I}_{\tilde{b}}^{(B)} = \emptyset \text{ for } b \neq \tilde{b}, \tag{14}$$

where each block has dimension

$$d_B = \frac{d_{\max}}{B}. \tag{15}$$

For the joint relational graph, each node corresponds to a triple (ℓ, t, b), where ℓ is a selected layer, t is a selected generated-token position, and b is a feature block. The node feature is the corresponding block of the hidden state,

$$\mathbf{h}^{\ell}_{\boldsymbol{\theta}_i, t}(\mathbf{Q}, \mathbf{A})[\mathcal{I}_b^{(B)}].$$

Changing B therefore changes both the granularity of each node and the graph size. When B is small, each node contains a high-dimensional block and may mix unrelated coordinates. When B is very large, each node contains very few coordinates; at $B = 1024$, each block has only eight dimensions, so the graph becomes close to coordinate-level and each node carries little standalone information.

Evaluation settings. We evaluate

$$B \in \{2, 4, 8, 16, 32, 64, 128, 256, 512, 1024\}.$$

The sweep is conducted in two settings. For the observational diagnostic, we use the fixed joint relational structure from Section 3, with $T = 7$ late generated tokens and $L = 5$ middle layers. For the full method, we use CHORD with its default compact window, $T = 3$ late generated tokens and $L = 3$ middle layers. All other components are kept unchanged, so the only varied factor is the feature-block count B.

The number of graph nodes grows linearly with B. In the observational setting, the joint graph contains $7 \times 5 \times B = 35B$ nodes. In the method setting, it contains $3 \times 3 \times B = 9B$ nodes. Thus, increasing B not only changes the feature granularity but also increases the cost of relation aggregation. This makes the sweep a direct test of the signal–compute trade-off induced by feature-block partitioning.

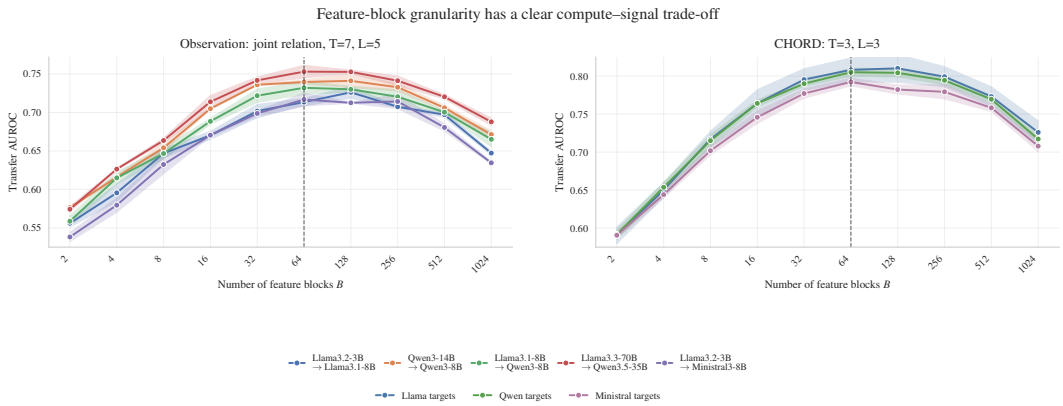

Figure 7: Effect of feature-block count B on the observational joint-relation diagnostic and the full CHORD method. Hidden states are padded to $d_{\max} = 8192$ and partitioned into B contiguous blocks. Very small B produces overly coarse blocks, while very large B makes each node too low-dimensional and greatly increases graph size. The default $B = 64$ provides a strong trade-off between transfer performance and graph complexity.

Results. Figure 7 shows a clear non-monotonic trend. When B is very small, performance is low in both settings. This is expected because each feature block is too coarse. For example, $B = 2$ or $B = 4$ means that each node contains thousands of hidden dimensions, so relations between feature blocks cannot isolate more localized activation patterns. As B increases from 2 to 32, transfer AUROC improves rapidly, indicating that finer feature partitioning exposes more useful internal relational structure.

The best region appears around $B = 64$, with $B = 128$ remaining close in performance. This suggests that the method does not rely on a single fragile block count, but benefits from a moderate block granularity. At $B = 64$, each block has dimension $8192/64 = 128$, which is small enough to provide localized feature structure while still large enough for each node to contain meaningful multi-dimensional information. Larger values such as $B = 128$ and $B = 256$ can preserve similar accuracy, but they increase the graph size and therefore provide a less favorable compute–performance trade-off.

For very large B, the performance starts to decline. At the largest setting we evaluate, $B = 1024$, each block has only eight dimensions and the graph is an order of magnitude larger than at the default. In this regime, each node carries too little information, and the relational graph becomes dominated by fine coordinate-level fragments rather than stable feature-block relations. The corresponding transfer AUROC falls well below the peak, confirming that simply increasing the number of nodes is not beneficial.

Implication. The block-count sweep supports the choice of $B = 64$ in the main experiments. A very small B loses fine-grained feature relations, while a very large B produces low-information nodes and excessive graph expansion. The default $B = 64$ lies near the strongest region of the curve and keeps the method computationally manageable. Thus, the gain of CHORD is not obtained by making the graph arbitrarily large; it comes from using a block granularity that preserves enough information per node while still exposing relational structure across tokens, layers, and feature blocks.

A.4 Why We Use Mutual k-Nearest-Neighbor Alignment

We use mutual k-nearest-neighbor (kNN) alignment to quantify whether two LLMs induce similar sample-level structures after applying a given feature construction. Our choice follows the representation-alignment view in Huh et al. [38]: a representation can be characterized by the similarity structure, or kernel, that it induces over data samples, and representational alignment measures the agreement between such induced similarity structures. Under this view, centered kernel alignment (CKA), SVCCA, and nearest-neighbor-based metrics can all be regarded as different ways of comparing representation-induced kernels.

A natural choice would be CKA, which compares two centered kernel matrices globally. However, this global comparison can be unnecessarily strict in our setting. Cross-model hallucination detection does not require two heterogeneous LLMs to have globally identical representation geometry. Different LLMs may differ in architecture, hidden dimension, scale, coordinate gauge, and model-specific semantic organization. What matters for detector transfer is instead whether the local neighborhood around each example is preserved: examples that are similar under the source-model feature should remain similar under the target-model feature. Such local preservation indicates that the feature carries a transferable sample structure even if the global geometry is not fully aligned.

This locality consideration is also consistent with the analysis of Huh et al. [38]. They start from CKA as a global kernel-alignment measure, but then consider a nearest-neighbor-restricted variant, CKNNA, which computes kernel alignment only over samples that are local neighbors. When the neighborhood size approaches the full batch, this variant recovers standard CKA; when the neighborhood is small, it closely resembles mutual nearest-neighbor alignment. Their empirical analysis further shows that emphasizing local neighborhoods can reveal clearer alignment trends, and their appendix compares mutual kNN with CKA, unbiased CKA, SVCCA, CKNNA, and other nearest-neighbor variants. Thus, mutual kNN is not an ad hoc proxy, but a lightweight local-alignment measure supported by the broader kernel-alignment framework.

Formally, given two feature maps induced by LLMs i and j on the same n examples, let $\mathcal{N}_k^i(s)$ and $\mathcal{N}_k^j(s)$ denote the k nearest neighbors of sample s under the two feature spaces, respectively. We compute neighbors using the inner-product similarity of the corresponding feature vectors. The per-sample neighborhood agreement is

$$a_{ij}(s) = \frac{1}{k}\left|\mathcal{N}_k^i(s) \cap \mathcal{N}_k^j(s)\right|, \tag{16}$$

and the reported alignment score is the average

$$A_{ij} = \frac{1}{n}\sum_{s=1}^{n} a_{ij}(s). \tag{17}$$

The score lies in $[0, 1]$, where larger values indicate stronger agreement between the local neighborhood structures induced by the two LLMs.

We set $k = 10$ in all experiments. This value is small relative to our evaluation set size ($n = 5000$), so the metric remains strongly local: if two kNN sets were selected independently at random from the remaining $n - 1$ examples, the expected normalized overlap would be approximately $k/(n-1) \approx 0.002$. At the same time, $k = 10$ is large enough to avoid the instability of extremely small neighborhoods, such as $k = 1$, which can be sensitive to noise, ties, or minor representation perturbations. This setting also matches the scale commonly used in nearest-neighbor alignment analyses such as Huh et al. [38]. We use the same k for all model pairs and feature types, and do not tune it using target-model performance.

A.5 Graph Construction Details

This subsection provides the implementation details of the graph construction used in Section 4. The purpose of this step is to convert hidden states from heterogeneous LLMs into a unified layer-token-feature graph on which relation-aware attention can be applied.

Scattered padding for dimension compatibility. Different LLMs may have different hidden dimensions d_i. To make their hidden states compatible with a shared graph encoder, we map every hidden state into a common ambient space $\mathbb{R}^{d^\star}$, where $d^\star \geq \max_i d_i$. For model $\mathbb{P}_{\boldsymbol{\theta}_i}$, the deterministic index map

$$p_i(\kappa) = \lfloor (\kappa - 1)d^\star / d_i \rfloor + 1, \qquad \kappa \in \{1, \ldots, d_i\}, \tag{18}$$

assigns each original coordinate to a unique position in the padded dimension. The padded hidden state is defined as

$$\bar{\mathbf{h}}^{\ell}_{\boldsymbol{\theta}_i,t}[p_i(\kappa)] = \mathbf{h}^{\ell}_{\boldsymbol{\theta}_i,t}[\kappa], \qquad \bar{\mathbf{h}}^{\ell}_{\boldsymbol{\theta}_i,t}[j] = 0 \quad \text{for } j \notin \mathrm{Im}(p_i), \tag{19}$$

where $\mathrm{Im}(p_i)$ is the set of padded coordinates reached by p_i. This mapping is deterministic and parameter-free. It is not intended to semantically align individual hidden coordinates across LLMs;

rather, it provides a common input dimension so that all models can be processed by the same downstream graph encoder. The scattered placement avoids concentrating all zero padding at the end of the vector and gives every feature block a more balanced mixture of original and padded coordinates.

Feature-block partition. After padding, the common dimension is partitioned into B equal-size contiguous blocks:

$$\mathcal{I}_b = \{(b-1)d^\star/B + 1, \ldots, bd^\star/B\}, \qquad b \in \{1, \ldots, B\}. \tag{20}$$

We choose $d^\star$ to be divisible by B. In our implementation, we set $d^\star$ to the smallest multiple of B that is no smaller than the largest hidden dimension among the evaluated LLMs. The block representation of token t at layer ℓ is then

$$\bar{\mathbf{h}}^{\ell}_{\boldsymbol{\theta}_i,t}[\mathcal{I}_b] \in \mathbb{R}^{d^\star/B}. \tag{21}$$

The number of feature blocks B controls the granularity of the feature-axis decomposition; its selection is discussed in Appendix A.3.

Selected layer-token window. For each input $(\mathbf{Q}, \mathbf{A})$, we select a token subset $\mathbf{T}^{\mathrm{sel}} = \{t_1, \ldots, t_r\}$ and a layer subset $\mathbf{L}_i^{\mathrm{sel}} = \{\ell_1, \ldots, \ell_m\}$ from model $\mathbb{P}_{\boldsymbol{\theta}_i}$. The same numbers $r = |\mathbf{T}^{\mathrm{sel}}|$ and $m = |\mathbf{L}_i^{\mathrm{sel}}|$ are used for all LLMs, while the actual layer indices are chosen by relative depth so that models with different numbers of layers can be compared. The token and layer selection strategy is described in Appendix A.2. For the final detector, we use the compact setting $r = 3$ and $m = 3$ unless otherwise stated.

Node set. The graph contains one node for every selected layer-token-block triple:

$$u \leftrightarrow (\ell_u, t_u, b_u), \qquad \ell_u \in \mathbf{L}_i^{\mathrm{sel}}, \quad t_u \in \mathbf{T}^{\mathrm{sel}}, \quad b_u \in \{1, \ldots, B\}. \tag{22}$$

Thus, the number of nodes is

$$N = |\mathbf{L}_i^{\mathrm{sel}}| \cdot |\mathbf{T}^{\mathrm{sel}}| \cdot B = mrB. \tag{23}$$

For the default setting $m = 3$, $r = 3$, and $B = 64$, the graph has $N = 576$ nodes.

Initial node representation. For node $u = (\ell_u, t_u, b_u)$, the raw block feature is

$$\bar{\mathbf{h}}^{\ell_u}_{\boldsymbol{\theta}_i,t_u}[\mathcal{I}_{b_u}] \in \mathbb{R}^{d^\star/B}. \tag{24}$$

We project this block feature into the detector dimension and add learnable index embeddings for the layer, token, and block identities:

$$\mathbf{x}_u = \mathbf{W}_{\mathrm{p}} \bar{\mathbf{h}}^{\ell_u}_{\boldsymbol{\theta}_i,t_u}[\mathcal{I}_{b_u}] + \mathbf{e}_\ell(\ell_u) + \mathbf{e}_t(t_u) + \mathbf{e}_b(b_u) \in \mathbb{R}^{d_{\mathrm{m}}}. \tag{25}$$

Here $\mathbf{W}_{\mathrm{p}} \in \mathbb{R}^{d_{\mathrm{m}} \times d^\star/B}$ is a learnable projection matrix, and $\mathbf{e}_\ell$, $\mathbf{e}_t$, and $\mathbf{e}_b$ are learnable lookup tables. The projection encodes the hidden-state block content, while the three embeddings tell the graph encoder where the node lies in the layer-token-feature grid.

Directed fully connected edge set. We construct a directed fully connected graph over the N nodes. That is, every ordered pair (u, v) is allowed to participate in relation-aware attention. Following standard self-attention, we include self-loops; when $u = v$, the relative offsets are zero and the content similarity terms take their self-comparison values. The graph connectivity is therefore not learned or sparsified. What is learned is the representation of each edge and how the attention encoder uses it.

Edge feature. For each directed pair (u, v), we compute relative offsets along the three graph axes:

$$\Delta^{\ell}_{uv} = \ell_u - \ell_v, \qquad \Delta^{t}_{uv} = t_u - t_v, \qquad \Delta^{b}_{uv} = b_u - b_v. \tag{26}$$

These offsets are embedded by learnable lookup tables $\boldsymbol{\phi}_\ell$, $\boldsymbol{\phi}_t$, and $\boldsymbol{\phi}_b$. We also compute two content-dependent terms from the initial node representations:

$$c_{uv} = \cos(\mathbf{x}_u, \mathbf{x}_v), \qquad d_{uv} = \|\mathbf{x}_u - \mathbf{x}_v\|_2. \tag{27}$$

The edge feature is then

$$\mathbf{e}_{uv} = \mathrm{MLP}_1\left([\boldsymbol{\phi}_\ell(\Delta^{\ell}_{uv}), \boldsymbol{\phi}_t(\Delta^{t}_{uv}), \boldsymbol{\phi}_b(\Delta^{b}_{uv}), c_{uv}, d_{uv}]\right) \in \mathbb{R}^{d_{\mathrm{m}}}. \tag{28}$$

Thus, each edge is both position-aware and content-aware: the relative offsets encode where two nodes are located in the layer-token-feature grid, while the similarity and distance terms encode how their hidden-state blocks relate for the current input.

Output of graph construction. The graph construction step produces

$$\mathcal{G}_i(\mathbf{Q}, \mathbf{A}) = \left(\{\mathbf{x}_u\}_{u=1}^N, \{\mathbf{e}_{uv}\}_{u,v=1}^N\right). \tag{29}$$

This graph is then passed to the relation-aware attention encoder described in Appendix A.6. The construction above is shared across all source and target LLMs; only the extracted hidden states and the deterministic padding map p_i depend on the specific LLM.

A.6 Details of Relation-Aware Attention

This subsection expands the relation-aware attention module used in Section 4. The module follows the general formulation of relation-aware self-attention in Shaw et al. [50], which augments self-attention with pairwise edge representations. While the original formulation uses relative positions in a sequence as edge labels, CHORD applies the same principle to the fully connected layer-token-feature graph constructed in Section 4. That is, every node corresponds to a triple (ℓ_u, t_u, b_u), and every directed edge (u, v) carries an input-dependent relation feature $\mathbf{e}_{uv}$ defined from relative layer, token, and block offsets, together with content similarity and distance.

Fully connected relation graph. Although we use relative offsets to parameterize edge types, the graph itself is not sparsified. For every ordered node pair (u, v) with $u, v \in \{1, \dots, N\}$, we construct an edge feature $\mathbf{e}_{uv}$ and allow node u to attend to node v. Therefore, the relation-aware encoder operates on a directed fully connected graph. The relative offsets determine how each edge is represented, not whether the edge exists. This is consistent with the relation-aware view of Shaw et al. [50], where self-attention can be interpreted as operating over a labeled, directed, fully connected graph.

Edge feature construction. Recall that each node u has an initial representation $\mathbf{x}_u \in \mathbb{R}^{d_{\mathrm{m}}}$, and each node is associated with a layer index ℓ_u, token index t_u, and feature-block index b_u. For a directed pair (u, v), we define clipped relative offsets

$$\Delta^{\ell}_{uv} = \mathrm{clip}(\ell_u - \ell_v, c_\ell), \quad \Delta^{t}_{uv} = \mathrm{clip}(t_u - t_v, c_t), \quad \Delta^{b}_{uv} = \mathrm{clip}(b_u - b_v, c_b), \tag{30}$$

where $\mathrm{clip}(a, c) = \max(-c, \min(c, a))$. Following the motivation of relative-position attention, clipping ties together sufficiently distant offsets so that they share the same relation label. Importantly, clipping does not remove any edge: all directed pairs (u, v) still participate in the attention softmax. In our implementation, we set

$$c_\ell = |\mathbf{L}^{\mathrm{sel}}| - 1, \qquad c_t = |\mathbf{T}^{\mathrm{sel}}| - 1, \qquad c_b = B - 1. \tag{31}$$

Thus, for the compact graphs used by the final detector, all relative offsets inside the selected layer-token-block grid are represented exactly. This choice avoids introducing an additional sparsification or truncation rule, while still keeping the number of relation labels bounded by the selected graph size.

We embed the three relative offsets with learnable lookup tables,

$$\boldsymbol{\phi}_\ell(\Delta^{\ell}_{uv}), \quad \boldsymbol{\phi}_t(\Delta^{t}_{uv}), \quad \boldsymbol{\phi}_b(\Delta^{b}_{uv}) \in \mathbb{R}^{d_{\mathrm{r}}}, \tag{32}$$

and combine them with two content-dependent scalars,

$$c_{uv} = \cos(\mathbf{x}_u, \mathbf{x}_v), \qquad d_{uv} = \|\mathbf{x}_u - \mathbf{x}_v\|_2. \tag{33}$$

The final edge feature is

$$\mathbf{e}_{uv} = \mathrm{MLP}_1\Big([\boldsymbol{\phi}_\ell(\Delta^{\ell}_{uv}), \boldsymbol{\phi}_t(\Delta^{t}_{uv}), \boldsymbol{\phi}_b(\Delta^{b}_{uv}), c_{uv}, d_{uv}]\Big) \in \mathbb{R}^{d_{\mathrm{m}}}. \tag{34}$$

Thus, unlike a purely relative-position Transformer, our edge feature is both *position-aware* and *content-aware*. This is important because hallucination-related evidence may depend not only on where two nodes occur in the layer-token-feature grid, but also on how their hidden-state blocks differ for a particular input.

Relation-aware multi-head attention. Let $\mathbf{x}_u^{(q)}$ be the representation of node u before the q-th relation-aware attention layer, with $\mathbf{x}_u^{(0)} = \mathbf{x}_u$. For each attention head $h \in \{1, \dots, H\}$, we compute

$$\mathbf{q}_u^{(q,h)} = \mathbf{W}_Q^{(q,h)} \mathbf{x}_u^{(q)}, \quad \mathbf{k}_v^{(q,h)} = \mathbf{W}_K^{(q,h)} \mathbf{x}_v^{(q)}, \quad \mathbf{v}_v^{(q,h)} = \mathbf{W}_V^{(q,h)} \mathbf{x}_v^{(q)}, \tag{35}$$

where $\mathbf{q}_u^{(q,h)}, \mathbf{k}_v^{(q,h)}, \mathbf{v}_v^{(q,h)} \in \mathbb{R}^{d_h}$ and $d_h = d_{\mathrm{m}}/H$. Following Shaw et al. [50], we use two edge projections: one for attention compatibility and one for value propagation,

$$\mathbf{a}_{uv}^{K,(q,h)} = \mathbf{W}_E^{K,(q,h)} \mathbf{e}_{uv}, \qquad \mathbf{a}_{uv}^{V,(q,h)} = \mathbf{W}_E^{V,(q,h)} \mathbf{e}_{uv}, \tag{36}$$

where $\mathbf{a}_{uv}^{K,(q,h)}, \mathbf{a}_{uv}^{V,(q,h)} \in \mathbb{R}^{d_h}$. The compatibility score from node u to node v is then

$$s_{uv}^{(q,h)} = \frac{\left(\mathbf{q}_u^{(q,h)}\right)^\top \left(\mathbf{k}_v^{(q,h)} + \mathbf{a}_{uv}^{K,(q,h)}\right)}{\sqrt{d_h}}, \tag{37}$$

and the attention weight is

$$\alpha_{uv}^{(q,h)} = \frac{\exp(s_{uv}^{(q,h)})}{\sum_{v'=1}^{N} \exp(s_{uv'}^{(q,h)})}. \tag{38}$$

Since the graph is fully connected, the denominator ranges over all nodes $v' \in \{1, \dots, N\}$. The message received by node u in head h is

$$\mathbf{m}_u^{(q,h)} = \sum_{v=1}^{N} \alpha_{uv}^{(q,h)} \left(\mathbf{v}_v^{(q,h)} + \mathbf{a}_{uv}^{V,(q,h)}\right). \tag{39}$$

The head outputs are concatenated and projected:

$$\mathbf{m}_u^{(q)} = \mathbf{W}_O^{(q)} \left[\mathbf{m}_u^{(q,1)}; \cdots; \mathbf{m}_u^{(q,H)}\right]. \tag{40}$$

Finally, we use the standard Transformer-style residual, layer normalization, and feed-forward update:

$$\tilde{\mathbf{x}}_u^{(q+1)} = \mathrm{LN}\left(\mathbf{x}_u^{(q)} + \mathrm{Dropout}(\mathbf{m}_u^{(q)})\right), \tag{41}$$

$$\mathbf{x}_u^{(q+1)} = \mathrm{LN}\left(\tilde{\mathbf{x}}_u^{(q+1)} + \mathrm{Dropout}\left(\mathrm{FFN}(\tilde{\mathbf{x}}_u^{(q+1)})\right)\right). \tag{42}$$

After Q relation-aware attention layers, we set $\mathbf{x}'_u = \mathbf{x}_u^{(Q)}$ and apply the attention pooling operator in Section 4 to obtain the graph-level feature $\mathbf{z}$.

Connection to the original relation-aware attention. The above equations are a direct graph-level adaptation of the relation-aware self-attention mechanism of Shaw et al. [50]. In their sequence setting, the edge representation is determined by the relative position $j - i$ between two tokens. In our setting, the edge representation is determined by the relative layer-token-block offsets and the input-dependent content relation between two graph nodes. Thus, our module replaces a one-dimensional sequential relation with a three-dimensional layer-token-feature relation, while preserving the same key idea: edge information modifies both the attention score and the propagated value.

Implementation details. Unless otherwise stated, we use $Q = 2$ relation-aware attention layers, $H = 4$ attention heads, $d_{\mathrm{m}} = 256$, and $d_h = 64$. The relative-offset embedding size is $d_{\mathrm{r}} = 64$. The edge MLP MLP_1 is a two-layer network with hidden dimension 256, GELU activation, dropout 0.1, and output dimension d_{m}. The feed-forward network inside each relation-aware attention layer uses hidden dimension $4d_{\mathrm{m}}$ with GELU activation and dropout 0.1. We include self-loops in the attention softmax, following standard self-attention; for $u = v$, the relative offsets are zero, $c_{uv} = 1$, and $d_{uv} = 0$. All edge features $\mathbf{e}_{uv}$ are computed once from the initial node representations and reused across relation-aware attention layers. This keeps the edge semantics tied to the original layer-token-feature graph while allowing node states to be updated through multi-hop relational propagation.

For the final detector, we use a graph with $|\mathbf{T}^{\mathrm{sel}}| = 3$, $|\mathbf{L}^{\mathrm{sel}}| = 3$, and $B = 64$, giving $N = 576$ graph nodes. The graph is fully connected, so the relation-aware encoder processes N^2 directed node pairs per example. Because the selected token and layer windows are small, this cost remains tractable in practice, while still allowing the model to capture interactions across layers, tokens, and feature blocks. The sensitivity analysis in Appendix A.2 further shows that increasing the number of selected tokens or layers yields only marginal additional gains.

A.7 Details of Meta-Learning Training

This subsection expands the meta-learning training strategy used in Section 4. Our implementation is adapted from Feature-Critic [14], which learns an auxiliary loss by simulating domain shift among source domains. In the original formulation, source domains are randomly split into virtual training and validation domains, and the auxiliary loss is optimized so that an update using this auxiliary loss improves validation-domain performance. In CHORD, we instantiate this idea for cross-model hallucination detection by treating each source LLM as one domain. Thus, a held-out source LLM in each meta-episode serves as a pseudo-unseen target model.

Detector decomposition. We write the detector as

$$\hat{y} = D_\Theta(\mathbf{Q}, \mathbf{A}; \mathbb{P}_{\boldsymbol{\theta}_i}) = C_\phi\big(G_\psi(\mathbf{Q}, \mathbf{A}; \mathbb{P}_{\boldsymbol{\theta}_i})\big), \tag{43}$$

where G_ψ denotes the graph encoder, including hidden-state restructuring, graph construction, relation-aware attention, and attention pooling. It outputs the graph-level feature

$$\mathbf{z} = G_\psi(\mathbf{Q}, \mathbf{A}; \mathbb{P}_{\boldsymbol{\theta}_i}) \in \mathbb{R}^{d_{\mathrm{m}}}, \tag{44}$$

and C_ϕ is the binary hallucination classifier. The full learnable parameter set is $\Theta = (\psi, \phi)$. Unlike the heterogeneous classification setting in Li et al. [14], all source LLMs in our problem share the same binary label space, so we use a single shared classifier C_ϕ rather than domain-specific classifiers.

Meta-train and meta-validation split. Let $\mathcal{M}_{\mathrm{src}}$ denote the set of source LLMs used during training, and let $\mathcal{D}_i$ be the labeled hallucination-detection data associated with source LLM $\mathbb{P}_{\boldsymbol{\theta}_i}$. At each meta-training iteration, we randomly sample one source LLM as the pseudo-target model,

$$i_{\mathrm{val}} \sim \mathrm{Unif}(\mathcal{M}_{\mathrm{src}}), \tag{45}$$

and define

$$\mathcal{M}_{\mathrm{val}} = \{i_{\mathrm{val}}\}, \qquad \mathcal{M}_{\mathrm{trn}} = \mathcal{M}_{\mathrm{src}} \setminus \mathcal{M}_{\mathrm{val}}. \tag{46}$$

Mini-batches are sampled from all LLMs in $\mathcal{M}_{\mathrm{trn}}$ and from the held-out LLM in $\mathcal{M}_{\mathrm{val}}$. This leave-one-source-model-out split matches the test condition of CMHD: the detector must perform well on an LLM whose hidden-state distribution was not used for the current update.

Supervised loss. For a mini-batch $\mathcal{B}$, the binary cross-entropy loss is

$$\mathcal{L}_{\mathrm{bce}}(\mathcal{B}; \psi, \phi) = \frac{1}{|\mathcal{B}|} \sum_{(\mathbf{Q}, \mathbf{A}, y, i) \in \mathcal{B}} \ell^{\mathrm{bce}}\left(C_\phi\big(G_\psi(\mathbf{Q}, \mathbf{A}; \mathbb{P}_{\boldsymbol{\theta}_i})\big), y\right). \tag{47}$$

In practice, mini-batches are class-balanced within each source LLM whenever possible. When exact balancing is not possible, we use a positive-class weight in the binary cross-entropy computed from the training split.

Feature-Critic auxiliary loss. Following Li et al. [14], we introduce a learned non-negative auxiliary loss, denoted by h_ω. The critic is applied to the graph-level features produced by G_ψ, rather than directly to the detector parameters. For a mini-batch $\mathcal{B}_i$ from source LLM i, let

$$\mathbf{Z}_i = \left[\mathbf{z}_{i,1}; \ldots; \mathbf{z}_{i,|\mathcal{B}_i|}\right] \in \mathbb{R}^{|\mathcal{B}_i| \times d_{\mathrm{m}}}, \tag{48}$$

where $\mathbf{z}_{i,r} = G_\psi(\mathbf{Q}_{i,r}, \mathbf{A}_{i,r}; \mathbb{P}_{\boldsymbol{\theta}_i})$. We use a permutation-invariant set-embedding critic,

$$h_\omega(\mathbf{Z}_i) = \mathrm{Softplus}\left(\mathbf{w}_\omega^\top \frac{1}{|\mathcal{B}_i|} \sum_{r=1}^{|\mathcal{B}_i|} \rho_\omega\big(\mathrm{LN}(\mathbf{z}_{i,r})\big) + b_\omega\right), \tag{49}$$

where ρ_ω is a small MLP, LN denotes layer normalization, and the final Softplus ensures that the critic output is non-negative. The auxiliary loss over the meta-training LLMs is

$$\mathcal{L}_{\mathrm{aux}}(\mathcal{B}_{\mathrm{trn}}; \psi, \omega) = \frac{1}{|\mathcal{M}_{\mathrm{trn}}|} \sum_{i \in \mathcal{M}_{\mathrm{trn}}} h_\omega(\mathbf{Z}_i). \tag{50}$$

This design follows the set-embedding variant of Feature-Critic, which is more parameter-efficient than covariance-based critics for high-dimensional features.

Meta-objective. The critic should be useful only if an update using it improves performance on the pseudo-target LLM. To measure this, we compare two one-step virtual updates of the graph encoder parameters. The first update uses only the supervised loss:

$$\psi^{\mathrm{old}} = \psi - \alpha \nabla_{\psi} \mathcal{L}_{\mathrm{bce}}(\mathcal{B}_{\mathrm{trn}}; \psi, \phi). \tag{51}$$

The second update uses both the supervised loss and the learned auxiliary loss:

$$\psi^{\mathrm{new}} = \psi - \alpha \nabla_{\psi} \left[\mathcal{L}_{\mathrm{bce}}(\mathcal{B}_{\mathrm{trn}}; \psi, \phi) + \lambda_{\mathrm{fc}} \mathcal{L}_{\mathrm{aux}}(\mathcal{B}_{\mathrm{trn}}; \psi, \omega) \right], \tag{52}$$

where α is the virtual inner-step size and λ_{fc} controls the strength of the auxiliary loss. The classifier parameters ϕ are kept fixed in these virtual updates, because the critic is intended to improve the transferable graph representation rather than to overfit the binary decision head.

Given a validation mini-batch $\mathcal{B}_{\mathrm{val}}$ from the held-out source LLM, we compute

$$\mathcal{L}_{\mathrm{val}}^{\mathrm{old}} = \mathcal{L}_{\mathrm{bce}}(\mathcal{B}_{\mathrm{val}}; \psi^{\mathrm{old}}, \phi), \tag{53}$$

$$\mathcal{L}_{\mathrm{val}}^{\mathrm{new}} = \mathcal{L}_{\mathrm{bce}}(\mathcal{B}_{\mathrm{val}}; \psi^{\mathrm{new}}, \phi). \tag{54}$$

The meta-loss for updating the critic is

$$\mathcal{L}_{\mathrm{meta}}(\omega) = \tanh\left(\mathcal{L}_{\mathrm{val}}^{\mathrm{new}} - \mathcal{L}_{\mathrm{val}}^{\mathrm{old}} \right). \tag{55}$$

Minimizing this loss encourages the critic to produce auxiliary gradients that make ψ^{new} outperform ψ^{old} on the pseudo-target LLM. The hyperbolic tangent follows the utility transformation used in Feature-Critic and prevents very large validation-loss differences from producing unstable critic updates.

Outer optimization. After computing the meta-loss, we update the critic parameters by

$$\omega \leftarrow \omega - \eta_{\omega} \nabla_{\omega} \mathcal{L}_{\mathrm{meta}}(\omega), \tag{56}$$

where gradients are backpropagated through the differentiable virtual update that produces ψ^{new}. We then update the actual detector parameters on the meta-training mini-batches:

$$\psi \leftarrow \psi - \eta_{\psi} \nabla_{\psi} \left[\mathcal{L}_{\mathrm{bce}}(\mathcal{B}_{\mathrm{trn}}; \psi, \phi) + \lambda_{\mathrm{fc}} \mathcal{L}_{\mathrm{aux}}(\mathcal{B}_{\mathrm{trn}}; \psi, \omega) \right], \tag{57}$$

$$\phi \leftarrow \phi - \eta_{\phi} \nabla_{\phi} \mathcal{L}_{\mathrm{bce}}(\mathcal{B}_{\mathrm{trn}}; \psi, \phi). \tag{58}$$

When updating (ψ, ϕ), the critic parameters ω are treated as fixed. When updating ω, the detector parameters are not updated by the meta-loss. This separation keeps the critic as a learned training signal rather than an additional prediction module at test time. At inference, the critic is discarded and only D_{Θ} is used.

All-source consolidation. After meta-training, we run a short consolidation stage over all source LLMs without any pseudo-target split. The critic is frozen, and the detector is trained on the union of all source-model data using

$$\mathcal{L}_{\mathrm{final}} = \mathcal{L}_{\mathrm{bce}} + \lambda_{\mathrm{fc}} \mathcal{L}_{\mathrm{aux}}. \tag{59}$$

This stage uses no target-model data. It simply lets the detector absorb all labeled source-model information after the critic has learned a generalization-promoting auxiliary objective.

Implementation details. Unless otherwise stated, we use one virtual inner update per meta-iteration. We set the meta-validation split to one held-out source LLM, i.e., $|\mathcal{M}_{\mathrm{val}}| = 1$, and use all remaining source LLMs as meta-training domains. The per-source mini-batch size is 32 for both meta-training and meta-validation. We use $\lambda_{\mathrm{fc}} = 0.1$ and virtual step size $\alpha = 5 \times 10^{-4}$. The critic MLP ρ_{ω} has two layers with hidden dimension 256, GELU activation, dropout 0.1, and output dimension 128. The detector parameters (ψ, ϕ) are optimized with AdamW using learning rate 2×10^{-4} and weight decay 10^{-4}, while the critic uses AdamW with learning rate 1×10^{-4} and the same weight decay. We clip gradients to norm 1.0. For stability, the first 5% of training updates use only the supervised binary cross-entropy loss; the Feature-Critic auxiliary loss and meta-loss are enabled afterward. The final all-source consolidation stage uses the last 10% of the total training budget.

A.8 Details for the Cross-Model Alignment, Transfer, and Score-Distribution Diagnostics

This subsection provides the experimental details for Figures 2 and 3. We use a mixed benchmark pool constructed from TruthfulQA [19], HaluEval [20], TriviaQA [21], CoQA [22], GSM8K [23], MBPP [24], and xLAM [25]. For each benchmark, we sample up to 1,000 examples; for MBPP, we use all available examples because the usable set contains fewer than 1,000 examples in our setting. The same question pool is used across models whenever possible, so that both neighborhood alignment and score-distribution comparisons are computed over matched evaluation examples.

We evaluate five representative directed source–target model pairs: Llama3.2-3B→Llama3.1-8B, Qwen3-14B→Qwen3-8B, Llama3.1-8B→Qwen3-8B, Llama3.3-70B→Qwen3.5-35B, and Llama3.2-3B→Ministral3-8B. These pairs cover same-family transfer, cross-family transfer, scale-matched transfer, large-model transfer, and scale-mismatched transfer. Because hidden dimensions differ across models, we first map all extracted hidden states to a shared padded dimension $d_{\max} = 8192$ before constructing raw or relational features. For a hidden state $\mathbf{h}^{\ell}_{\boldsymbol{\theta}_i,t} \in \mathbb{R}^{d_i}$, the common-dimensional vector is obtained by the same deterministic index mapping used in the main method: each original coordinate is placed into a fixed position in the $d_{\max}$-dimensional vector, and all remaining positions are filled with zeros. This step keeps feature dimensions consistent across LLMs without learning an alignment from target labels.

For each example and each model, we extract hidden states from the selected generated-token and layer windows, apply the common-dimensional mapping, and then construct the raw-state, token-level, cross-layer token, within-token feature, and joint relational features described in Section 3. Raw-state features are obtained by pooling the mapped hidden states over the same selected token–layer window. Relational features are constructed after the same mapping, so that token, layer, feature-block, and joint relations are computed in a common feature space.

For local-neighborhood alignment in Figure 2, we compute mutual kNN overlap with $k = 10$. Given two models in a pair, we build the k nearest-neighbor set of each example under the feature space induced by each model, using inner-product similarity after feature normalization. The reported alignment is the average normalized overlap between the two neighbor sets over all examples. This gives a model-pair-level measure of whether two LLMs induce similar local example geometry under the same feature construction.

For transfer AUROC in Figure 2, we train a detector on the source-model features and evaluate it directly on the target-model features. The detector is a lightweight two-layer MLP with one hidden layer, ReLU activation, dropout, and a final scalar logit trained with binary cross-entropy. We use the same MLP architecture and optimization protocol for all feature types, so differences in transfer AUROC reflect the feature construction rather than a change in classifier capacity. Training uses only labels from the source model, while target-model labels are used only for evaluation. All reported bar values are averaged over multiple random seeds, and error bars show the corresponding seed-level variation.

Figure 3 further visualizes the detector score distributions for one representative transfer pair, Llama3.1-8B→Qwen3-8B. For each feature construction, we use the same source-trained MLP detector described above and record its scalar detector scores on the target-model evaluation examples. We then estimate separate kernel density curves for factual and hallucinated examples. The solid curve denotes factual examples and the dashed curve denotes hallucinated examples. The AUROC annotated in each panel is recomputed from the finite sample of target scores plotted in that panel, and may therefore differ slightly from the corresponding entry of the full transfer matrix. Thus, Figure 3 provides a distribution-level view of the same transfer behavior summarized by AUROC in Figure 2.

In addition to raw and relational features, Figure 3 includes two training-free reference baselines, Perplexity and Semantic Entropy. These methods are evaluated directly on the same target examples and do not use the source-model training labels. They are included only as score-distribution references: Perplexity represents a simple model-confidence baseline, while Semantic Entropy represents a stronger uncertainty-based hallucination detector. Comparing their distributions with the relational-feature detectors helps contextualize how much separability is obtained by simple relational transfer before introducing the full CHORD training framework.

A.9 Observation Pipeline for Relational Features

This subsection provides the full diagnostic pipeline used in Section 3.2. The goal of this observation study is not to introduce the final learnable detector, but to test whether simple handcrafted relational structures already contain transferable hallucination signals across LLMs. Therefore, all edge weights in this subsection are fixed by non-learnable functions, and the only supervised component used for predictive transfer is a lightweight MLP detector trained after the relational features are extracted.

Shared preprocessing and notation. For each LLM $\mathbb{P}_{\boldsymbol{\theta}_i}$, we extract hidden states $\mathbf{h}^{\ell}_{\boldsymbol{\theta}_i,t}(\mathbf{Q},\mathbf{A}) \in \mathbb{R}^{d_i}$ from selected answer-token positions and selected layers. The selected token set is denoted by $\mathbf{T}^{\mathrm{sel}} = \{t_1,\dots,t_r\}$, and the selected layer set is denoted by $\mathbf{L}_i^{\mathrm{sel}} = \{\ell_1,\dots,\ell_m\}$, where $r = |\mathbf{T}^{\mathrm{sel}}|$ and $m = |\mathbf{L}_i^{\mathrm{sel}}|$. The number and positions of selected tokens and layers are specified in Appendix A.2. Since different LLMs may have different hidden dimensions, we first map all hidden states into the common padded space $\mathbb{R}^{d^\star}$ using the same hidden-state restructuring operation as in Section 4. For notational simplicity, we denote the mapped hidden state by

$$\bar{\mathbf{h}}^{\ell}_{\boldsymbol{\theta}_i,t} = \mathrm{Map}_i\left(\mathbf{h}^{\ell}_{\boldsymbol{\theta}_i,t}\right) \in \mathbb{R}^{d^\star}. \tag{60}$$

For feature-block and joint relations, the mapped dimension is partitioned into B equal-size contiguous blocks $\{\mathcal{I}_b\}_{b=1}^{B}$ as in Eq. (12). The choice of B is discussed in Appendix A.3. We choose $d^\star$ to be divisible by B.

Generic relation-weighted graph readout. All relational features follow the same graph construction and pooling template. Given a relation type $r \in \{\mathrm{tok},\mathrm{lyr},\mathrm{feat},\mathrm{joint}\}$, we construct a graph

$$\mathcal{G}_i^r(\mathbf{Q},\mathbf{A}) = (\mathcal{V}_i^r, \mathcal{E}_i^r), \tag{61}$$

where each node $u \in \mathcal{V}_i^r$ has a node feature $\mathbf{x}_u^r$, and each directed edge (u,v) has a non-negative relation weight $R_i^r(u,v;\mathbf{Q},\mathbf{A})$. For the diagnostic graph readout, we use a fully connected graph without self-loops. The edge weights are row-normalized as

$$\alpha_i^r(u,v) = \frac{R_i^r(u,v;\mathbf{Q},\mathbf{A})}{\sum_{v'\in\mathcal{V}_i^r,\, v'\neq u} R_i^r(u,v';\mathbf{Q},\mathbf{A}) + \varepsilon}, \qquad v \neq u, \tag{62}$$

where $\varepsilon = 10^{-8}$ is used for numerical stability. Each node then receives one relation-weighted aggregation step,

$$\mathbf{m}_u^r = \sum_{v\in\mathcal{V}_i^r,\, v\neq u} \alpha_i^r(u,v)\mathbf{x}_v^r. \tag{63}$$

Finally, we obtain a graph-level feature by mean pooling:

$$\mathbf{z}_i^r(\mathbf{Q},\mathbf{A}) = \frac{1}{|\mathcal{V}_i^r|}\sum_{u\in\mathcal{V}_i^r} \mathbf{m}_u^r. \tag{64}$$

This pooling operation is invariant to the enumeration order of the nodes, while the edge weights still encode the intended token, layer, and block relations. The same readout template is used for all relational features below; they differ only in the definition of nodes and edge weights.

Token-level relation. For token-level relations, the graph nodes are the selected tokens in $\mathbf{T}^{\mathrm{sel}}$ at a diagnostic layer $\ell^\star$. We choose $\ell^\star$ by the same relative-depth rule used in Appendix A.2. Each token node has feature

$$\mathbf{x}_t^{\mathrm{tok}} = \bar{\mathbf{h}}^{\ell^\star}_{\boldsymbol{\theta}_i,t}. \tag{65}$$

For two selected tokens t and $\tilde{t}$, the edge weight is

$$R_i^{\mathrm{tok}}(t,\tilde{t};\mathbf{Q},\mathbf{A}) = s^{\mathrm{tok}}(t,\tilde{t})\left\|\bar{\mathbf{h}}^{\ell^\star}_{\boldsymbol{\theta}_i,t} - \bar{\mathbf{h}}^{\ell^\star}_{\boldsymbol{\theta}_i,\tilde{t}}\right\|_2, \tag{66}$$

where

$$s^{\mathrm{tok}}(t,\tilde{t}) = \exp\left(-\frac{|t-\tilde{t}|}{\tau_{\mathrm{tok}}}\right). \tag{67}$$

We use $\tau_{\mathrm{tok}} = 2.0$ in selected-token index units and keep it fixed across all experiments.

Cross-layer token relation. For cross-layer token relations, each node corresponds to one selected layer $\ell \in \mathbf{L}_i^{\mathrm{sel}}$. The node feature is the vectorized stack of the selected token representations at that layer:

$$\mathbf{x}_\ell^{\mathrm{lyr}} = \mathrm{vec}\left(\left[\bar{\mathbf{h}}_{\boldsymbol{\theta}_i,t_1}^{\ell}, \ldots, \bar{\mathbf{h}}_{\boldsymbol{\theta}_i,t_r}^{\ell}\right]^\top\right) \in \mathbb{R}^{rd^\star}. \tag{68}$$

For two selected layers ℓ and $\tilde{\ell}$, the relation weight is

$$R_i^{\mathrm{lyr}}(\ell, \tilde{\ell}; \mathbf{Q}, \mathbf{A}) = s^{\mathrm{lyr}}(\ell, \tilde{\ell}) \left(1 - \cos\left(\mathbf{x}_\ell^{\mathrm{lyr}}, \mathbf{x}_{\tilde{\ell}}^{\mathrm{lyr}}\right)\right), \tag{69}$$

where

$$s^{\mathrm{lyr}}(\ell, \tilde{\ell}) = \exp\left(-\frac{|\ell - \tilde{\ell}|}{\tau_{\mathrm{lyr}}}\right). \tag{70}$$

We use $\tau_{\mathrm{lyr}} = 2.0$ in selected-layer index units and keep it fixed. The cosine-distance term captures representational change across depth, while the decay factor gives slightly larger weight to nearby layers.

Within-token feature relation. For within-token feature relations, each node corresponds to one feature block $b \in \{1, \ldots, B\}$. For every block, we collect its hidden-state block over all selected layers and selected tokens:

$$\mathbf{V}_{\boldsymbol{\theta}_i,b}(\mathbf{Q}, \mathbf{A}) = \left[\bar{\mathbf{h}}_{\boldsymbol{\theta}_i,t_1}^{\ell_1}[\mathcal{I}_b], \ldots, \bar{\mathbf{h}}_{\boldsymbol{\theta}_i,t_r}^{\ell_1}[\mathcal{I}_b], \ldots, \bar{\mathbf{h}}_{\boldsymbol{\theta}_i,t_1}^{\ell_m}[\mathcal{I}_b], \ldots, \bar{\mathbf{h}}_{\boldsymbol{\theta}_i,t_r}^{\ell_m}[\mathcal{I}_b]\right]^\top \in \mathbb{R}^{mr \times d^\star/B}. \tag{71}$$

The block-node feature is

$$\mathbf{x}_b^{\mathrm{feat}} = \mathrm{vec}\left(\mathbf{V}_{\boldsymbol{\theta}_i,b}(\mathbf{Q}, \mathbf{A})\right) \in \mathbb{R}^{mrd^\star/B}. \tag{72}$$

For two blocks b and $\tilde{b}$, the relation weight is

$$R_i^{\mathrm{feat}}(b, \tilde{b}; \mathbf{Q}, \mathbf{A}) = \left\|\mathbf{x}_b^{\mathrm{feat}} - \mathbf{x}_{\tilde{b}}^{\mathrm{feat}}\right\|_2. \tag{73}$$

This relation measures how differently two feature blocks respond over the same selected token-layer window.

Joint layer-token-feature relation. For joint relations, each node corresponds to a triple $u = (\ell, t, b)$ with $\ell \in \mathbf{L}_i^{\mathrm{sel}}$, $t \in \mathbf{T}^{\mathrm{sel}}$, and $b \in \{1, \ldots, B\}$. The node feature is the corresponding hidden-state block:

$$\mathbf{x}_u^{\mathrm{joint}} = \bar{\mathbf{h}}_{\boldsymbol{\theta}_i,t}^{\ell}[\mathcal{I}_b] \in \mathbb{R}^{d^\star/B}. \tag{74}$$

For two nodes $u = (\ell, t, b)$ and $v = (\tilde{\ell}, \tilde{t}, \tilde{b})$, the relation weight is

$$R_i^{\mathrm{joint}}(u, v; \mathbf{Q}, \mathbf{A}) = s^{\mathrm{lyr}}(\ell, \tilde{\ell}) s^{\mathrm{tok}}(t, \tilde{t}) \left\|\bar{\mathbf{h}}_{\boldsymbol{\theta}_i,t}^{\ell}[\mathcal{I}_b] - \bar{\mathbf{h}}_{\boldsymbol{\theta}_i,\tilde{t}}^{\tilde{\ell}}[\mathcal{I}_{\tilde{b}}]\right\|_2. \tag{75}$$

This joint graph combines the three axes considered separately above: depth, token position, and feature-block structure. It is deliberately handcrafted in the observation study, so that any cross-model transfer observed here can be attributed to relational structure rather than to a learned relation module.

Raw hidden-state baseline. To ensure that the comparison between raw representations and relational features is controlled, the raw-state baseline uses the same selected token-layer window and the same common hidden-state mapping. Specifically, we define

$$\mathbf{z}_i^{\mathrm{raw}}(\mathbf{Q}, \mathbf{A}) = \frac{1}{mr} \sum_{\ell \in \mathbf{L}_i^{\mathrm{sel}}} \sum_{t \in \mathbf{T}^{\mathrm{sel}}} \bar{\mathbf{h}}_{\boldsymbol{\theta}_i,t}^{\ell}. \tag{76}$$

Thus, improvements from relational features cannot be explained simply by using more tokens, more layers, or a larger mapped representation space.

Cross-model structural alignment. For structural alignment, we compute the feature vectors $\mathbf{z}_i^r(\mathbf{Q}^s, \mathbf{A}_i^s)$ and $\mathbf{z}_j^r(\mathbf{Q}^s, \mathbf{A}_j^s)$ for the same set of prompts under two LLMs i and j. For each model, we construct a kNN graph over examples using inner-product similarity between the corresponding feature vectors. We then report mutual kNN overlap as the cross-model alignment score. The detailed definition and the choice of k are given in Appendix A.4. No hallucination labels are used in this structural-alignment computation.

Cross-model predictive transfer. For predictive transfer, we train a lightweight MLP detector on the source-model features and evaluate it directly on the target-model features. For each ordered source-target pair (i, j) and each feature type r, the MLP takes $\mathbf{z}_i^r(\mathbf{Q}, \mathbf{A}_i)$ as input and predicts the binary truthfulness label. The same trained detector is then applied to $\mathbf{z}_j^r(\mathbf{Q}, \mathbf{A}_j)$ without updating its parameters. The detector is a two-layer MLP:

$$p(y = 1 \mid \mathbf{z}) = \sigma\left(\mathbf{w}_2^\top \mathrm{Dropout}\left(\mathrm{GELU}\left(\mathbf{W}_1\mathbf{z} + \mathbf{b}_1\right)\right) + b_2\right), \tag{77}$$

with hidden dimension 256 and dropout rate 0.1. We train it with binary cross-entropy using AdamW, learning rate 10^{-3}, weight decay 10^{-4}, batch size 256, and early stopping on source-validation AUROC. All reported AUROC values are averaged over three random seeds. The same MLP architecture and optimization protocol are used for raw and relational features.

Scope of the observation pipeline. The observation pipeline uses fixed handcrafted relations and a simple post-hoc MLP only to diagnose whether different relational structures are stable and predictive across models. It does not use the learnable relation-aware attention, attention pooling, or meta-learning strategy of CHORD. Those components are introduced later in Section 4 to turn the empirical observation into a trainable cross-model detector.

A.10 Baseline Implementation Details

This subsection describes the baselines used in our experiments and how we adapt them to the cross-model hallucination detection (CMHD) setting. The main complication is that many training-based hallucination detectors were originally designed for a single LLM, where training and testing features live in the same representation space. In CMHD, however, the detector is trained on source LLMs and evaluated on unseen target LLMs, whose hidden dimensions, layer counts, head counts, and representation gauges may differ. Therefore, some baselines cannot be applied directly without a compatibility step. For such methods, we use the minimal adaptation needed to make them executable under CMHD, while keeping their scoring rule as close as possible to the original design. These adaptations should be understood as evaluation-time compatibility operations rather than improvements intended by the original baseline authors.

General evaluation protocol. All baselines are evaluated on the same generated answers, labels, and source–target splits as CHORD. For every scalar hallucination score, we report AUROC on the target model. If a method produces a confidence score rather than a hallucination score, we reverse its sign so that larger values consistently indicate higher hallucination likelihood. When the score direction is ambiguous, the direction is selected using source-validation data only, never using target labels. For methods requiring thresholds, thresholds are selected on source-validation data, but AUROC is computed from continuous scores whenever available. For sampling-based baselines, we use the same sampling budget across methods: unless otherwise stated, we draw $K_{\mathrm{gen}} = 10$ additional generations per question with temperature 0.5 and nucleus sampling $p = 0.95$. This keeps test-time computation comparable across uncertainty and self-consistency baselines.

Perplexity. Perplexity is a training-free likelihood-based baseline. Given a question $\mathbf{Q}$ and an answer $\mathbf{A} = (a_1, \ldots, a_T)$ generated by the evaluated LLM, we compute the length-normalized negative log-likelihood

$$s_{\mathrm{ppl}}(\mathbf{Q}, \mathbf{A}) = -\frac{1}{T}\sum_{t=1}^{T} \log p_{\boldsymbol{\theta}}(a_t \mid \mathbf{Q}, a_{<t}). \tag{78}$$

Larger values indicate that the model assigns lower probability to its own answer, which is treated as a higher hallucination score. This baseline does not require training or representation alignment, and can therefore be applied directly to each target LLM. However, since perplexity is sensitive to surface form, answer length, and tokenization, we use length normalization and compute perplexity with the same tokenizer and model that generated the answer.

Semantic Entropy. Semantic Entropy estimates uncertainty in meaning space rather than token-sequence space [5]. For each question, we sample K_{gen} additional answers from the same LLM and group the sampled answers into semantic-equivalence clusters using a fixed entailment or LLM-judge

module. Let $\mathcal{C}$ denote the resulting set of semantic clusters and let $p(c)$ be the normalized probability mass assigned to cluster c. The score is

$$s_{\text{se}}(\mathbf{Q}) = -\sum_{c \in \mathcal{C}} p(c) \log p(c). \tag{79}$$

A high semantic entropy indicates that the model produces many semantically distinct answers, suggesting uncertainty and a higher hallucination risk. This method is training-free and does not rely on hidden-state dimensions, so it is directly applicable to unseen target LLMs. For fairness, we use the same generation budget and semantic-equivalence module for all models.

EigenScore. EigenScore is a white-box, training-free self-consistency baseline that uses internal states from multiple sampled generations [52]. For each question, we sample K_{gen} answers and extract one sentence-level hidden representation from each sampled answer, following the original method. Let $\mathbf{r}_1, \ldots, \mathbf{r}_{K_{\text{gen}}}$ denote these representations and let $\mathbf{\Sigma}$ be their empirical covariance matrix. EigenScore is computed from the spectrum of $\mathbf{\Sigma}$; larger spectral dispersion indicates that the generated answers occupy a more diverse internal-state subspace, which is taken as higher hallucination risk. Since EigenScore is computed within a single target model at test time and does not train a source-model probe, it does not require cross-model hidden-dimension mapping. We therefore apply it directly on each target LLM's hidden states.

Lexical Similarity. Lexical Similarity is a black-box self-consistency baseline. It samples multiple answers to the same question and measures their pairwise surface-form similarity using a lexical overlap metric such as ROUGE-L or token-level F1. For each evaluated answer, we compute its average lexical similarity to the sampled alternatives and use the negative average similarity as the hallucination score:

$$s_{\text{lex}}(\mathbf{Q}, \mathbf{A}) = -\frac{1}{K_{\text{gen}}} \sum_{k=1}^{K_{\text{gen}}} \text{Sim}_{\text{lex}}(\mathbf{A}, \mathbf{A}^{(k)}). \tag{80}$$

Low lexical agreement suggests that the model is unstable under resampling, which is treated as higher hallucination risk. This method does not depend on hidden states or attention maps, and thus requires no cross-model mapping. We keep the same sampling budget as Semantic Entropy to ensure a fair comparison.

Verbalize. Verbalize is a prompting-based confidence baseline that asks the LLM to explicitly report its confidence in the generated answer [53, 54]. After obtaining an answer $\mathbf{A}$, we query the same LLM with a fixed confidence prompt:

```
Q: {question} A: {answer} The proposed answer is true with a
confidence value (0-100) of
```

We parse the generated number as a confidence score and use its negative value as the hallucination score. If the model does not produce a valid number, we reprompt once with a stricter numerical-output instruction; remaining invalid outputs are assigned the median confidence on the source-validation set. This baseline is training-free and black-box, so it is naturally applicable to unseen LLMs. However, verbalized confidence is known to be affected by calibration and instruction-following behavior, so we treat it as a behavioral baseline rather than a representation-based detector.

InterrogateLLM. InterrogateLLM is a zero-resource hallucination detector that checks whether an answer contains enough reliable information to reconstruct the original query [55]. Given an answer $\mathbf{A}$, the model is asked to generate a reverse question $\tilde{\mathbf{Q}}$ that would have led to $\mathbf{A}$. The original question $\mathbf{Q}$ and reconstructed question $\tilde{\mathbf{Q}}$ are then compared. If the answer is faithful, the reconstructed question should be semantically close to the original one; if the answer is hallucinated or drifts from the requested information, reconstruction becomes less reliable. We implement the comparison using the same semantic-similarity module across all models and use

$$s_{\text{int}}(\mathbf{Q}, \mathbf{A}) = -\text{Sim}_{\text{sem}}(\mathbf{Q}, \tilde{\mathbf{Q}}) \tag{81}$$

as the hallucination score. For tasks where the original query is not naturally reconstructable from the answer alone, such as code generation or agent-tool tasks, we use a task-aware reconstruction

prompt that asks the model to reconstruct the user's requested objective rather than the exact surface form of the question. This modification is necessary because the original InterrogateLLM protocol was designed primarily for factual question answering; it may deviate from the method's intended operating condition, but it makes the baseline applicable across our heterogeneous benchmark suite.

MM. MM denotes the Mass-Mean probe, a simple activation-space truthfulness probe [56]. Given hidden representations from truthful and hallucinated examples, it computes the class means

$$\boldsymbol{\mu}_+ = \frac{1}{|\mathcal{S}_+|} \sum_{\mathbf{r} \in \mathcal{S}_+} \mathbf{r}, \qquad \boldsymbol{\mu}_- = \frac{1}{|\mathcal{S}_-|} \sum_{\mathbf{r} \in \mathcal{S}_-} \mathbf{r}, \tag{82}$$

and defines the truthfulness direction as

$$\mathbf{w}_{\mathrm{MM}} = \boldsymbol{\mu}_+ - \boldsymbol{\mu}_-. \tag{83}$$

The score of a test representation $\mathbf{r}$ is then given by projection onto this direction. The original MM probe assumes that training and testing activations live in the same model-specific vector space. This assumption fails in CMHD because source and target LLMs may have different hidden dimensions. Therefore, for MM we first map hidden states into the same padded space $\mathbb{R}^{d^\star}$ using the hidden-state restructuring operation described in Section 4. We then train the mass-mean direction on source-model mapped hidden states and evaluate it on target-model mapped hidden states. No relation-aware graph construction or meta-learning is used for this baseline.

SEP. Semantic Entropy Probes (SEP) train a lightweight probe to approximate Semantic Entropy from a single generation's hidden state [6]. The purpose is to avoid the high test-time cost of sampling multiple generations required by Semantic Entropy. In our implementation, we first compute Semantic Entropy targets on source-model data using the sampling procedure described above. Then, a probe is trained to predict these entropy targets from hidden representations of the corresponding single generated answer. At test time, the probe predicts a semantic-entropy score from one target-model generation. As with MM, the original probe is not directly cross-model because its input dimension is tied to the hidden states of the model on which it was trained. We therefore apply the same hidden-state mapping into $\mathbb{R}^{d^\star}$ before training and testing the SEP probe. This adaptation makes SEP executable in CMHD, but it should be viewed as a compatibility modification rather than part of the original SEP design.

SAPLMA. SAPLMA trains a classifier on LLM activation values to predict whether a generated statement is true or false [57]. It is a representative training-based hidden-state detector. For each question-answer pair, we extract the hidden representation of the final answer token from a fixed relative-depth layer. Specifically, when a method specifies a single middle layer, we use the layer closest to half of the model depth; this avoids relying on a layer index that only exists for one architecture. The extracted hidden state is mapped into the common padded space $\mathbb{R}^{d^\star}$ and used as input to a lightweight MLP classifier. The classifier is trained on source LLMs and evaluated directly on the mapped hidden states of the target LLM. This mapping is necessary because SAPLMA's original formulation assumes a fixed activation dimension, whereas CMHD evaluates transfer across architectures with different hidden sizes.

LapEigvals. LapEigvals is an attention-map-based hallucination detector that interprets attention maps as graph adjacency matrices and uses Laplacian eigenvalues as spectral features [11]. Unlike MM, SEP, and SAPLMA, this method does not operate on hidden-state vectors. It instead extracts attention maps from the LLM and computes graph spectral statistics from them. Consequently, we do not apply hidden-state padding to LapEigvals. The cross-model compatibility issue is different: LLMs may have different numbers of layers and attention heads, and their attention tensors may therefore have different shapes. To make LapEigvals applicable to CMHD, we map each model's attention information to a common attention-feature format. We first select the same number of relative-depth layers and the same selected answer-token window for all models. For each selected layer and head, we form the attention submatrix over the selected tokens, symmetrize it, construct its normalized graph Laplacian, and extract the smallest non-trivial eigenvalues. To handle different head counts, we aggregate head-wise spectral features using mean and standard deviation within each selected layer. The resulting fixed-dimensional attention-spectral vector is used to train a source-model classifier and evaluate on target models. This preserves the key idea of LapEigvals, namely detecting

hallucinations from spectral properties of attention graphs, while removing architecture-specific tensor-shape dependencies.

Caveat on cross-model adaptations. The hidden-state mapping for MM, SEP, and SAPLMA, and the attention-feature mapping for LapEigvals, are introduced only because these baselines are otherwise not directly executable in the CMHD setting. They should not be interpreted as improvements to the original methods or as design choices advocated by their authors. Rather, they provide the fairest feasible way to compare single-model training-based detectors against a method explicitly designed for cross-model generalization. In contrast, training-free baselines such as Perplexity, Semantic Entropy, EigenScore, Lexical Similarity, Verbalize, and InterrogateLLM do not train a source-model representation classifier, so they are applied directly to each target LLM under a unified scoring and sampling protocol.

A.11 LLM-as-Judge Labeling Protocol

We assign hallucination labels using an LLM-as-Judge protocol rather than lexical-overlap heuristics. Recent re-evaluation work has shown that overlap-based criteria such as ROUGE-L can introduce substantial label noise in hallucination detection, because semantically correct answers may receive low overlap scores while fluent but incorrect answers may still achieve misleading lexical similarity. We therefore use a semantically aware judge under a fixed protocol and deterministic decoding.

Protocol summary. The evaluator is GPT-4o under a fixed prompt and structured output format. For each sample, the judge receives: (i) the question, (ii) an optional context, (iii) a list of acceptable truths, (iv) an optional list of known traps (incorrect answers), and (v) the model-generated answer. The judge returns one of three intermediate categories: `correct`, `hallucination`, or `abstention`. For the binary detector studied in this paper, `correct` and `abstention` are mapped to the positive class, while `hallucination` is mapped to the negative class.

Inputs to the Judge. Each evaluation instance is represented by a structured record containing the question, optional supporting context, one or more acceptable truths, optional known traps, and the answer generated by the model under evaluation. The acceptable truths specify the semantic content that should be accepted as correct, allowing multiple valid phrasings or equivalent answers. Known traps list common or particularly misleading incorrect answers when such information is available. This design reduces dependence on exact surface form and makes the evaluation more robust to paraphrase.

Decision Rules. The judge is instructed to output exactly one of three categories. It returns `correct` if the model answer semantically matches at least one acceptable truth. It returns `hallucination` if the answer matches a known trap, contradicts the acceptable truths, or introduces fabricated factual content not supported by the provided truth specification. It returns `abstention` if the model explicitly states that it does not know the answer, lacks sufficient information, or refuses to answer on epistemic grounds. The judge also produces a brief natural-language justification, which is stored for inspection but not used as model input.

Three-Way Intermediate Labels and Binary Mapping. We use a three-way intermediate decision layer to improve evaluator precision. In particular, separating `abstention` from `hallucination` prevents explicit uncertainty from being conflated with confident factual error. For the detector studied in this paper, however, the final supervision is binary. Accordingly, we map both `correct` and `abstention` to label 1 and map `hallucination` to label 0. This mapping matches the operational goal of the detector: to distinguish fabricated or unsupported factual claims from non-hallucinatory behavior.

Implementation Details. The evaluator uses deterministic decoding with temperature 0 and structured output parsing. To improve robustness in large-scale asynchronous evaluation, we use bounded retries for transient API failures and isolate samples that still fail after the retry budget is exhausted. Only successfully parsed judge outputs are retained in the labeled dataset. The exact prompt template, few-shot examples, and implementation code are included in the released repository.

Why Three-Way Judging Helps. The three-way design is not introduced because the final task is ternary, but because the intermediate decision boundary becomes cleaner. A direct binary judge is more likely to confuse explicit uncertainty with factual error, especially for difficult or underspecified questions. By first distinguishing `correct`, `hallucination`, and `abstention`, and only then collapsing to a binary label, we obtain a labeling procedure that better reflects the semantic distinction relevant to hallucination detection.

A.12 Dataset Details and Preprocessing

This appendix describes the seven benchmarks used in our experiments and the preprocessing pipeline that converts their heterogeneous formats into the unified input-output schema used throughout the paper. The datasets are selected to cover complementary hallucination-detection settings: open-domain factual QA, context-grounded QA, conversational QA, mathematical reasoning, code generation, and agent-style tool calling. For every benchmark, we first shuffle the original items with a fixed random seed and keep at most $10{,}000$ items. If the selected split contains fewer than $10{,}000$ usable items, we keep all available items. This case mainly affects benchmarks whose official or selected split is smaller than the cap, such as MBPP. We then split the selected items at the *question/task level* into training, validation, and test partitions using an $80/10/10$ ratio. The split is performed before LLM generation, so the same question or task does not appear in more than one partition.

For each dataset item, every evaluated LLM receives the same processed prompt and generates its own answer. Thus, for a fixed benchmark item indexed by s, different models may produce different answers $\mathbf{A}_i^s$, while the underlying question, context, and reference information are shared. The generated answer is then labeled as truthful if it is semantically aligned with one of the accepted references or explicitly abstains, and hallucinated if it commits to an answer that matches none of them, following the setup in Section 2. The detector training and evaluation are then performed on model-specific datasets of the form in Eq. (1). In the leave-one-family-out experiments, training-based methods use only source-family examples for training and validation, and all reported results are computed on the held-out target-family test examples.

A.12.1 Unified Dataset Schema

The raw datasets differ substantially in their field names, task formats, and reference annotations. To avoid dataset-specific prompt and evaluation logic, we convert every raw item into a shared structured schema. Each processed item stores the following fields:

- `task_type`: records the task family, such as `qa`, `reasoning`, `coding`, or `agent_action`.
- `system_instruction`: stores task-specific generation requirements, such as answering strictly from a provided context, reasoning step by step, writing valid Python code, or producing a JSON tool call.
- `context`: stores passages, stories, tool descriptions, tests, or other background information needed by the task.
- `question`: stores the actual user-facing query or problem statement.
- `choices`: is used for multiple-choice tasks; for the seven benchmarks in the main experiments, it is empty because all selected tasks are open-ended, reasoning, coding, or structured-generation tasks.
- `ground_truths`: stores all accepted references, including answer aliases when available.
- `incorrect_answers`: stores known false or hallucinated answers when the original benchmark provides them. These known incorrect answers are never included in the prompt; they are retained only as evaluation-side traps or reference signals for judging whether a generated answer matches a known hallucination pattern.

For reproducibility, each processed JSONL item also keeps the original raw dataset row under an `original_doc` field. This makes it possible to trace every processed example back to the original benchmark record. The prompt given to an LLM is constructed from the schema fields, not directly from the raw dataset row. When `system_instruction` is non-empty, it is placed before the user query. When `context` is non-empty, it is placed before the question as task evidence or constraints. The reference fields `ground_truths` and `incorrect_answers` are not shown to the model during generation.

A.12.2 TruthfulQA

TruthfulQA [19] evaluates whether a language model gives truthful answers to questions that are likely to elicit common misconceptions or false imitative responses. It is therefore a direct benchmark for factual hallucination and false-belief amplification. We use the generation setting, where each item contains a natural-language question, a list of correct answers, and a list of incorrect answers. This makes TruthfulQA especially useful for our setting because it provides both accepted factual references and known misleading alternatives.

In our schema, TruthfulQA is treated as an open-ended QA task:

`task_type=qa`.

The original question is mapped to `question`. No additional context is used, so `context` is set to the empty string. The original list of correct answers is copied into `ground_truths`, preserving multiple acceptable phrasings. The original list of incorrect answers is copied into `incorrect_answers`. During answer evaluation, a generated answer is considered truthful if it is semantically aligned with any accepted correct answer and does not match the known incorrect alternatives. Keeping the incorrect-answer list is important because many TruthfulQA questions are adversarial: a fluent but common answer may be precisely the hallucinated answer that the dataset is designed to expose.

A.12.3 HaluEval

HaluEval [20] is designed for hallucination evaluation and contains examples in which models must answer questions based on provided knowledge. We use the QA subset. Each item provides a knowledge passage, a question, a right answer, and a hallucinated answer. This makes the dataset a natural fit for context-grounded hallucination detection: the model must use the supplied evidence rather than produce unsupported statements.

In our schema, HaluEval is treated as a context-grounded QA task:

`task_type=qa`.

The knowledge field is mapped to `context`, and the question field is mapped to `question`. We use the instruction:

```
Please answer the question based strictly on the provided context.
```

The right answer is stored as a singleton list in `ground_truths`. The hallucinated answer is stored as a singleton list in `incorrect_answers`. The hallucinated answer is not shown to the LLM. It is preserved only for evaluation, where it helps identify generated answers that are fluent but inconsistent with the provided knowledge. This design lets HaluEval test whether the detector can distinguish context-supported answers from answers that sound plausible but are not grounded in the evidence.

A.12.4 TriviaQA

TriviaQA [21] is an open-domain factual QA benchmark. Questions often have short factual answers, but the same answer may appear in multiple acceptable surface forms. This makes the dataset useful for evaluating whether hallucination detectors generalize beyond adversarial truthfulness questions to ordinary knowledge-intensive factual QA.

We use the no-context reading-comprehension configuration, so each item is processed as open-ended QA without an additional passage:

`task_type=qa`.

The original question is mapped to `question`, and `context` is set to the empty string. For references, we use the answer alias list provided by the dataset whenever available. These aliases are stored in `ground_truths`. If no alias list is available, we fall back to the canonical answer value. This alias-preserving processing is important because a generated answer may be semantically correct even when it does not exactly match the canonical surface form. No dataset-provided hallucinated answer is available, so `incorrect_answers` is empty.

A.12.5 CoQA

CoQA [22] is a conversational question-answering benchmark grounded in a story. Each raw item contains one story and a sequence of conversational questions and answers. CoQA is included because it evaluates context-grounded answering in a conversational style, where the model must extract information from a passage rather than rely only on parametric knowledge.

In the main experiments, we use one QA instance from each CoQA story to avoid introducing multiple highly correlated examples from the same story into different splits. Specifically, we use the first question-answer turn of each story. The story is mapped to `context`, the first question is mapped to `question`, and the corresponding first answer is stored in `ground_truths`. The task is treated as:

`task_type=qa`.

No multiple-choice options are used, and `incorrect_answers` is empty. This processing keeps one top-level example per story, reducing the risk that different turns from the same story create near-duplicate training and test instances. It also gives a clean context-grounded QA setting that complements HaluEval: HaluEval focuses on hallucinated answers against explicit knowledge, while CoQA focuses on reading a narrative context and producing the correct answer.

A.12.6 GSM8K

GSM8K [23] is a grade-school mathematical reasoning benchmark. Each item contains a natural-language math word problem and a reference solution. We include GSM8K because reasoning hallucinations differ from factual QA hallucinations: an answer may fail because of an incorrect intermediate calculation, an invalid reasoning step, or a wrong final value. This benchmark therefore tests whether internal representations encode transferable signals for reasoning correctness.

In our schema, GSM8K is treated as a reasoning task:

`task_type=reasoning`.

The problem statement is mapped to `question`, and no external context is used. We use the instruction:

```
Please think step by step and provide the final answer at the end.
```

The dataset's reference answer is stored in `ground_truths`. The reference answer may contain both the reasoning trace and the final numeric answer, and we preserve the original string for traceability. During evaluation, the final answer is compared against the reference answer, with semantic or normalized numeric matching used when appropriate. No dataset-provided incorrect answers are available, so `incorrect_answers` is empty. By keeping GSM8K separate in the main and ablation tables, we can examine whether methods that work for factual QA also transfer to reasoning-oriented hallucination detection.

A.12.7 MBPP

MBPP [24] is a Python programming benchmark consisting of short natural-language programming tasks, reference code, and assertion-style tests. It is included because code correctness is not identical to natural-language factuality. A generated program may look plausible but fail on edge cases, violate the required function signature, or implement a subtly different behavior. MBPP therefore provides a useful stress test for hallucination detection in code-generation settings.

In our schema, MBPP is treated as a coding task:

`task_type=coding`.

The natural-language programming requirement is mapped to `question`. The provided assertion tests are concatenated and placed in `context`:

```
Your code must pass the following tests:
```

followed by the test list. We use the instruction:

```
You are an expert Python programmer.  Write a Python function to solve
the problem.  Your code must pass the provided assertion tests.
```

The reference code is stored in `ground_truths`. The tests are included in the prompt because they are part of the programming task specification and provide functional constraints. They are also retained for evaluation-side checking. When execution-based checking is available, generated code is evaluated against the provided assertions; otherwise, the reference code and tests are used by the evaluator to judge functional equivalence. No dataset-provided incorrect programs are available, so `incorrect_answers` is empty.

MBPP has fewer available items than our 10,000-item cap in the configuration used here, so we keep all available examples after shuffling and apply the same train/validation/test split ratio. Because the number of training examples is much smaller than in the other benchmarks, MBPP is expected to be a more fragile setting for training-based detectors. This is why the main text reports MBPP separately rather than only as part of a pooled average.

A.12.8 xLAM Function Calling

The xLAM function-calling benchmark [25] evaluates agent-style tool use. Each item contains a user query, a set of available tools, and the expected tool-call answer. Unlike ordinary QA, the output is often a structured action rather than a free-form natural-language answer. We include xLAM because hallucination in agentic settings often appears as selecting a nonexistent or inappropriate tool, filling incorrect arguments, or producing a tool call that does not satisfy the user's request. This setting tests whether hallucination detectors transfer to structured action generation.

In our schema, xLAM is treated as an agent-action task:

`task_type=agent_action`.

The available tool descriptions are placed in `context`, prefixed by:

```
Available Tools:
```

The user query is mapped to `question`. We use the instruction:

```
You are a helpful assistant with access to various tools.  Based on
the User's question, select the appropriate tool from the Context and
output the exact tool call in JSON format.  If no tool is needed, answer
directly.
```

The expected tool-call output is stored in `ground_truths`. Because tool-call outputs are structured, we canonicalize JSON strings when possible before evaluation, so superficial formatting differences do not count as hallucinations. The evaluator checks whether the generated tool name and arguments are semantically equivalent to the reference action. No dataset-provided incorrect tool calls are used, so `incorrect_answers` is empty.

xLAM complements the QA and coding benchmarks by testing hallucination detection for agent-oriented behavior. In this setting, correctness depends not only on factual content but also on whether the generated action is executable and appropriate for the tool inventory. This makes xLAM useful for assessing whether cross-model detectors capture broader internal signals of task-grounded correctness.

A.12.9 Summary of Dataset Roles

The seven benchmarks are intentionally heterogeneous. TruthfulQA focuses on adversarial factual truthfulness and common misconceptions. HaluEval tests context-grounded hallucination against provided knowledge and includes explicit hallucinated answers. TriviaQA tests open-domain factual recall with multiple valid answer aliases. CoQA tests context-grounded conversational QA over

stories. GSM8K tests mathematical reasoning correctness. MBPP tests functional code generation and execution-oriented correctness. xLAM tests structured agent actions and tool-call validity.

This diversity is important for the cross-model hallucination detection setting. A detector that transfers only on one type of data may be exploiting task-specific artifacts rather than model-general hallucination signals. By evaluating across factual QA, context-grounded QA, reasoning, code, and agent-action settings, we test whether the proposed detector captures internal signals that remain useful across both model families and task formats.

B Related Work

Hallucination Detection. A large body of work studies hallucination detection for LLM outputs using signals from output probabilities, answer consistency, internal activations, or attention structures [58, 59]. Uncertainty-based methods estimate reliability from likelihood, perplexity, entropy, or semantic uncertainty, and often avoid task-specific detector training [5]. Consistency-based methods sample multiple responses or construct auxiliary queries, then infer hallucination risk from disagreement, contradiction, or answer instability [4]. Another line trains lightweight detectors on hidden states, activation patterns, or semantic uncertainty features to predict truthfulness from a single forward pass [6, 52, 57, 60]. Recent attention-based approaches further explore structural statistics of attention maps for hallucination detection [11]. Although these methods are effective in many settings, most of them are designed for a fixed LLM or rely on output-level proxies, and therefore do not directly address detector transfer across unseen LLMs.

Robustness and Cross-Model Generalization. Our work is also related to broader studies on robustness, distributional generalization, and representation mismatch in modern AI systems. Theoretical and empirical studies on OOD robustness investigate when reliable separation is possible and how stable signals can be exploited under distribution shift [61, 62]. In parallel, representation similarity research studies how neural networks differ in their intermediate activations and how representations can be compared across layers, training runs, or architectures [29, 30]. These works indicate that models with similar functionality may still have substantially different internal representation spaces, making direct transfer of learned probes difficult. CMHD follows this motivation but focuses on hallucination detection across LLMs, where a detector trained on source models must remain effective on target models with different architectures, scales, or training procedures.

Relational and Graph-based Representation Learning. Relational modeling provides a natural way to move beyond isolated hidden-state vectors. Graph neural networks and message-passing frameworks aggregate information through node-edge interactions and produce graph-level representations through pooling or readout operations [36]. Relation-aware attention further incorporates positional or edge information into attention computation, allowing the model to learn interactions conditioned on relations rather than treating all pairs uniformly [50]. Attention pooling and related readout mechanisms summarize variable-size structures into compact representations while reducing sensitivity to node ordering [51]. These ideas are particularly relevant to LLM internals, where hallucination evidence may appear not only in individual token states but also in relations across tokens, layers, and feature groups.

Building on these threads, our work formulates *cross-model hallucination detection* and studies relational structures as transferable signals across LLMs. Instead of directly training detectors on model-specific hidden vectors, CHORD represents hidden states as joint layer-token-feature graphs, learns relation-aware graph representations, and uses meta-learning to encourage transfer to unseen target models. This design connects hallucination detection with cross-model robustness and relational representation learning, while targeting a setting that has received limited attention in prior work.

C Limitations

CHORD is designed for white-box or partially transparent LLM settings where hidden states can be accessed. It therefore cannot be directly applied to purely black-box APIs that expose only generated text or token probabilities. This limits its use in scenarios where model internals are unavailable. However, hallucination detection is often deployed in controlled enterprise or platform-side settings, where the service provider has access to model activations or can instrument the inference stack.

In such deployments, the white-box requirement is less restrictive, and CHORD can be used as an internal reliability module for monitoring generated answers.

D Computational Infrastructure

All experiments were conducted on a server equipped with four NVIDIA H100 GPUs. We use this setup because LLM inference itself requires substantial GPU memory, especially for the 70B model, and hidden-state extraction further increases memory usage. Although most models other than the 70B model can be run with less restrictive hardware, we use the same 4×H100 environment throughout to keep the extraction and evaluation pipeline consistent. After hidden representations are extracted, detector training and evaluation are much lighter and can in principle be performed on a consumer-grade GPU such as an NVIDIA RTX 5090; nevertheless, we keep the full experimental pipeline on the same 4×H100 setup for consistency.

E Licenses for Existing Assets

All models and datasets utilized in our extensive cross-family transfer experiments are open-source or available for research purposes. Table 3 details the specific licenses for the evaluated large language models and benchmarks. We strictly adhered to the usage terms prescribed by their respective creators. Notably, for datasets with mixed licenses like CoQA, and research-only assets like xLAM, we ensured our empirical evaluation process complies with their non-commercial and academic usage guidelines.

Table 3: Summary of licenses for all existing models and datasets used in this paper.

Asset Name	Type	License	Notes
Llama-3.1-8B	Model	Llama 3.1 Community	Follows Meta's specific community agreement.
Llama-3.2-3B	Model	Llama 3.2 Community	Follows Meta's specific community agreement.
Llama-3.3-70B	Model	Llama 3.3 Community	Follows Meta's specific community agreement.
Ministral-3-8B/14B	Model	Apache 2.0	Mistral AI open-source license.
Qwen3-8B/14B	Model	Apache 2.0	Alibaba Cloud open-source license.
Qwen3.5-35B-A3B	Model	Apache 2.0	Alibaba Cloud open-source license.
TruthfulQA	Dataset	Apache 2.0	Stated in the official GitHub repository.
HaluEval	Dataset	MIT	Stated in the official GitHub repository.
TriviaQA	Dataset	Apache 2.0	Stated in the official GitHub repository.
CoQA	Dataset	Mixed License	Includes CC BY-SA 4.0 (Wiki/Literature) and Apache 2.0 (CNN).
GSM8K	Dataset	MIT	Stated in the OpenAI GitHub repository.
MBPP	Dataset	CC BY 4.0	Stated in the official Google dataset release.
xLAM (APIGen)	Dataset	CC BY 4.0	Released for research purposes only (official data card).

F Observation Analysis Details

F.1 Alignment and Transfer Are Related but Decoupled

This section expands the observational analysis in Section 3. The goal is to examine whether cross-model representational alignment is sufficient to explain downstream hallucination-detector transfer. We study five feature views: raw hidden states, token-level relations, cross-layer token relations, within-token feature relations, and joint relational structures. All relational views in this section are fixed and hand-designed diagnostic views. They are not trained end-to-end and do not use the relation-aware attention module introduced in Section 4. They are used only to test what kinds of internal structures remain stable and useful across LLMs.

Feature views used in the diagnostic analysis. For each model $\mathbb{P}_{\boldsymbol{\theta}_i}$ and each generated example $(\mathbf{Q}, \mathbf{A}_i)$, we extract hidden states $\mathbf{h}^{\ell}_{\boldsymbol{\theta}_i,t}(\mathbf{Q}, \mathbf{A}_i)$ as defined in Section 2. Since different LLMs have different hidden dimensions, we use the same scattered-padding map as in Eq. (11) and Eq. (12). The padded hidden state is denoted by

$$\bar{\mathbf{h}}^{\ell}_{\boldsymbol{\theta}_i,t}(\mathbf{Q}, \mathbf{A}_i) \in \mathbb{R}^{d^\star}.$$

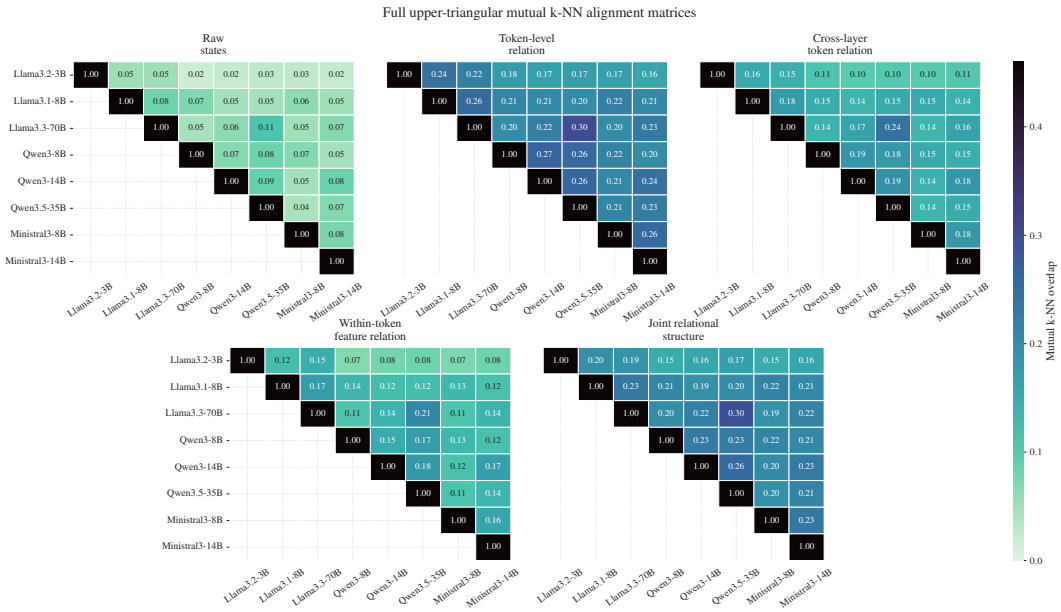

Figure 8: Full mutual k nearest neighbor alignment matrices across all model pairs. For each feature view, we compute the average overlap between the k nearest neighbor sets induced by two models over the same evaluation examples. Since this metric is symmetric, we show the upper triangular matrix and set the diagonal to one. Across the full model suite, relational views consistently yield stronger local-neighborhood agreement than raw hidden states, supporting the representative trends shown in the main text.

Unless otherwise stated, all feature-block operations in this appendix are applied to the padded hidden states $\bar{\mathbf{h}}^{\ell}_{\boldsymbol{\theta}_i,t}$. The scattered-padding map is injective for each model, so within-model dot products, cosine similarities, and Euclidean distances are preserved after padding.

The raw-state view uses the same selected hidden states as the relational diagnostics, but pools them directly without relation-weighted propagation. Let $\mathbf{T}^{\mathrm{sel}} = \{t_1, \ldots, t_r\}$ denote the selected token positions and $\mathbf{L}^{\mathrm{sel}} = \{\ell_1, \ldots, \ell_m\}$ denote the selected layers, with $m = |\mathbf{L}^{\mathrm{sel}}|$. Following Eq. (76), we define

$$\mathbf{z}_i^{\mathrm{raw}}(\mathbf{Q}, \mathbf{A}_i) = \frac{1}{mr} \sum_{\ell \in \mathbf{L}_i^{\mathrm{sel}}} \sum_{t \in \mathbf{T}^{\mathrm{sel}}} \bar{\mathbf{h}}^{\ell}_{\boldsymbol{\theta}_i,t}(\mathbf{Q}, \mathbf{A}_i). \tag{84}$$

This view keeps the selected hidden-state coordinates in the unified representation space, but it does not explicitly encode relations among tokens, layers, or feature blocks.

For the four relational views, we use a common relation-weighted propagation and mean-readout template. This notation is only used to describe the fixed diagnostic pipeline compactly. For a relation type q, let $\mathcal{V}_i^q$ denote its node set, $\mathbf{x}_{i,u}^q(\mathbf{Q}, \mathbf{A}_i)$ the feature of node $u \in \mathcal{V}_i^q$, and $R_i^q(u, v; \mathbf{Q}, \mathbf{A}_i)$ the hand-designed relation value between nodes u and v. The graph-level relational feature is

$$\mathbf{z}_i^q(\mathbf{Q}, \mathbf{A}_i) = \frac{1}{|\mathcal{V}_i^q|} \sum_{u \in \mathcal{V}_i^q} \sum_{v \in \mathcal{V}_i^q,\, v \neq u} \frac{R_i^q(u, v; \mathbf{Q}, \mathbf{A}_i)}{\epsilon + \sum_{v' \in \mathcal{V}_i^q,\, v' \neq u} R_i^q(u, v'; \mathbf{Q}, \mathbf{A}_i)} \mathbf{x}_{i,v}^q(\mathbf{Q}, \mathbf{A}_i), \tag{85}$$

where $\epsilon > 0$ is a small numerical-stability constant. Here R_i^q is treated as a relation magnitude rather than a similarity score. Thus, this diagnostic readout emphasizes nodes that have larger relative pairwise differences from the receiving node. The final mean readout is permutation-invariant over node orderings, which is useful because the same relational construction may be implemented with different internal node orders across LLMs.

The token-level relation view uses selected tokens as nodes. For notational simplicity, we write $\mathbf{z}_i^{\mathrm{tok}}$ for the token-level feature $\mathbf{z}_i^{\mathrm{tok},\ell}$ computed at the selected diagnostic layer ℓ. The node set is

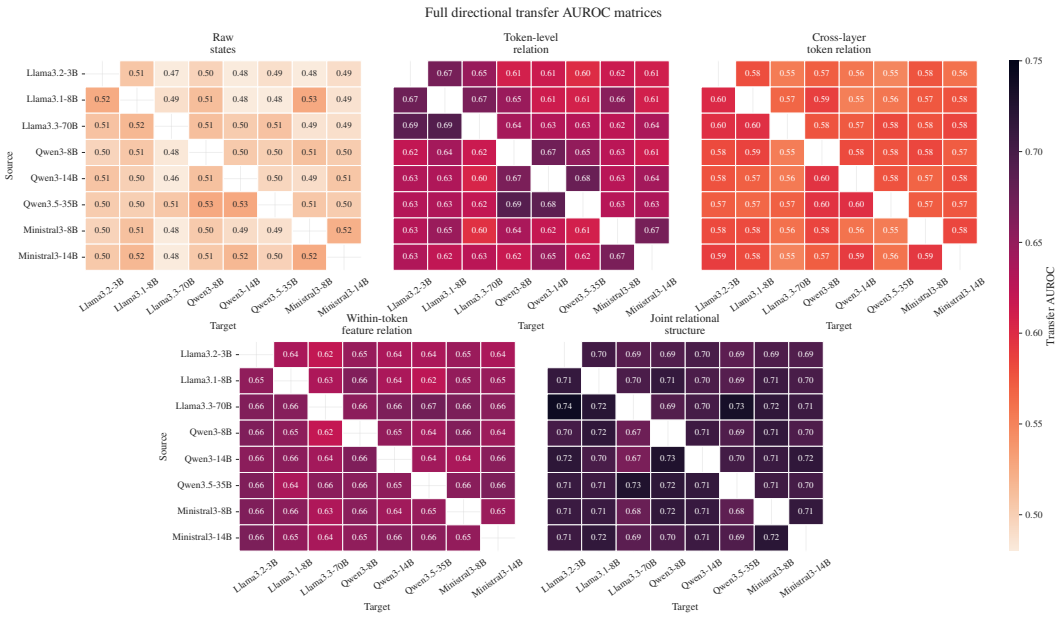

Figure 9: Full directional cross-model hallucination transfer matrices. Rows denote source models used for training the detector, and columns denote target models used for evaluation. Unlike representational alignment, transfer AUROC is directional, so we show the full matrix. Raw hidden states remain close to chance for most source-to-target pairs, whereas relational views produce substantially stronger transfer across model families and scales.

$\mathcal{V}_i^{\mathrm{tok}} = \mathbf{T}^{\mathrm{sel}}$, and the node feature is

$$\mathbf{x}_{i,t}^{\mathrm{tok}}(\mathbf{Q}, \mathbf{A}_i) = \bar{\mathbf{h}}_{\boldsymbol{\theta}_i,t}^{\ell}(\mathbf{Q}, \mathbf{A}_i).$$

The edge relation follows the token-distance relation in Eq. (3), implemented on padded hidden states:

$$R_i^{\mathrm{tok}}(t, \tilde{t}; \mathbf{Q}, \mathbf{A}_i) = s^{\mathrm{tok}}(t, \tilde{t}) \left\| \bar{\mathbf{h}}_{\boldsymbol{\theta}_i,t}^{\ell}(\mathbf{Q}, \mathbf{A}_i) - \bar{\mathbf{h}}_{\boldsymbol{\theta}_i,\tilde{t}}^{\ell}(\mathbf{Q}, \mathbf{A}_i) \right\|_2. \tag{86}$$

Applying Eq. (85) with this node set and relation value gives $\mathbf{z}_i^{\mathrm{tok}}(\mathbf{Q}, \mathbf{A}_i)$.

The cross-layer token relation view uses selected layers as nodes. For each layer $\ell \in \mathbf{L}^{\mathrm{sel}}$, we stack the padded hidden states of all selected tokens:

$$\bar{\mathbf{H}}_{\boldsymbol{\theta}_i}^{\ell,\mathrm{sel}}(\mathbf{Q}, \mathbf{A}_i) = \left[\bar{\mathbf{h}}_{\boldsymbol{\theta}_i,t_1}^{\ell}(\mathbf{Q}, \mathbf{A}_i), \ldots, \bar{\mathbf{h}}_{\boldsymbol{\theta}_i,t_r}^{\ell}(\mathbf{Q}, \mathbf{A}_i) \right]^{\top} \in \mathbb{R}^{r \times d^{\star}}. \tag{87}$$

The node associated with layer ℓ is represented by

$$\mathbf{x}_{i,\ell}^{\mathrm{lyr}}(\mathbf{Q}, \mathbf{A}_i) = \mathrm{vec}\left(\bar{\mathbf{H}}_{\boldsymbol{\theta}_i}^{\ell,\mathrm{sel}}(\mathbf{Q}, \mathbf{A}_i) \right).$$

The relation between two layer nodes is the depth-decayed cosine distance:

$$R_i^{\mathrm{lyr}}(\ell, \tilde{\ell}; \mathbf{Q}, \mathbf{A}_i) = s^{\mathrm{lyr}}(\ell, \tilde{\ell}) \left(1 - \cos\left(\mathrm{vec}(\bar{\mathbf{H}}_{\boldsymbol{\theta}_i}^{\ell,\mathrm{sel}}), \mathrm{vec}(\bar{\mathbf{H}}_{\boldsymbol{\theta}_i}^{\tilde{\ell},\mathrm{sel}}) \right) \right), \tag{88}$$

where $s^{\mathrm{lyr}}(\ell, \tilde{\ell}) = \exp(-|\ell - \tilde{\ell}|/\tau_{\mathrm{lyr}})$ is the depth-decay factor defined in Section 3.2. Applying Eq. (85) over $\mathcal{V}_i^{\mathrm{lyr}} = \mathbf{L}^{\mathrm{sel}}$ gives $\mathbf{z}_i^{\mathrm{lyr}}(\mathbf{Q}, \mathbf{A}_i)$.

The within-token feature relation view uses feature blocks as nodes. The padded hidden dimension is partitioned into B equal-size contiguous blocks with index sets $\mathcal{I}_b$ as in Eq. (12). For each block b and each selected layer ℓ, we first collect the b-th hidden-state block of all selected tokens:

$$\bar{\mathbf{V}}_{\boldsymbol{\theta}_i,b}^{\ell}(\mathbf{Q}, \mathbf{A}_i) = \left[\bar{\mathbf{h}}_{\boldsymbol{\theta}_i,t_1}^{\ell}(\mathbf{Q}, \mathbf{A}_i)[\mathcal{I}_b], \ldots, \bar{\mathbf{h}}_{\boldsymbol{\theta}_i,t_r}^{\ell}(\mathbf{Q}, \mathbf{A}_i)[\mathcal{I}_b] \right]^{\top} \in \mathbb{R}^{r \times |\mathcal{I}_b|}. \tag{89}$$

We then stack these token-block matrices vertically over the selected layers:

$$\bar{\mathbf{V}}_{\boldsymbol{\theta}_i,b}(\mathbf{Q}, \mathbf{A}_i) = \left[(\bar{\mathbf{V}}_{\boldsymbol{\theta}_i,b}^{\ell_1}(\mathbf{Q}, \mathbf{A}_i))^{\top}, \ldots, (\bar{\mathbf{V}}_{\boldsymbol{\theta}_i,b}^{\ell_{|\mathbf{L}^{\mathrm{sel}}|}}(\mathbf{Q}, \mathbf{A}_i))^{\top} \right]^{\top} \in \mathbb{R}^{|\mathbf{L}^{\mathrm{sel}}| r \times |\mathcal{I}_b|}. \tag{90}$$

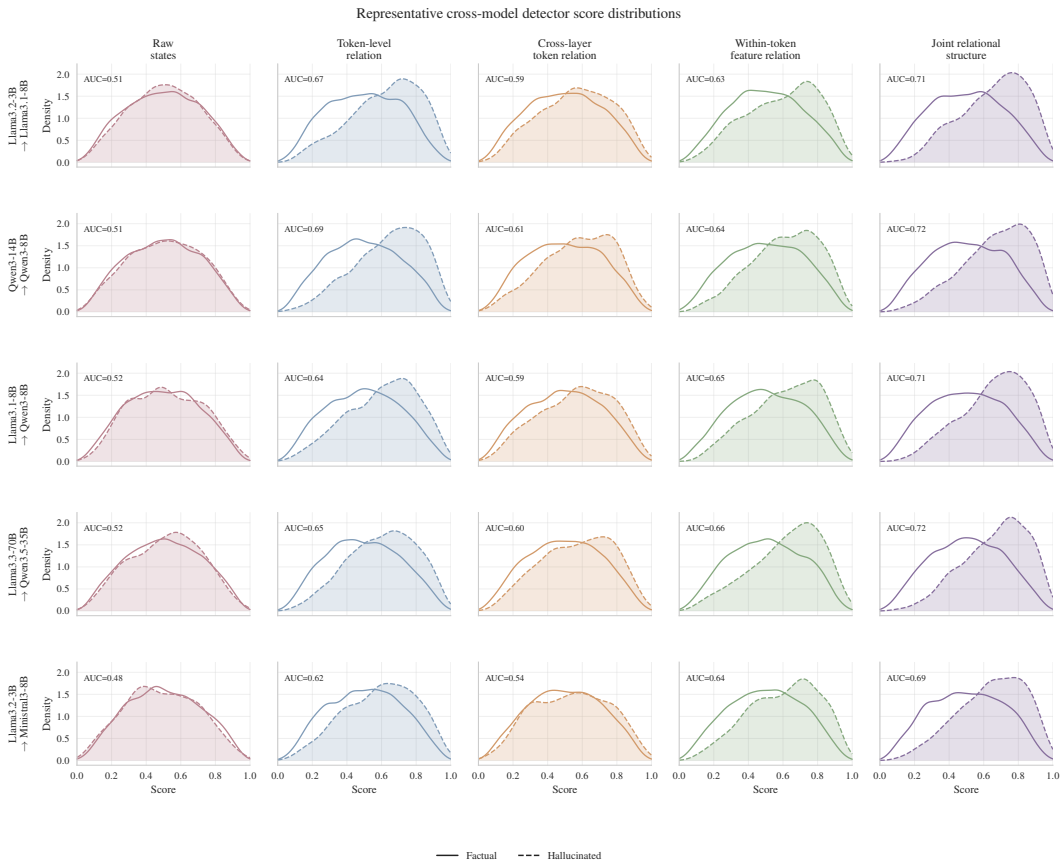

Figure 10: Representative detector score distributions across source-to-target model pairs. Each row corresponds to one representative transfer pair, and each column corresponds to one feature view. Curves show the score distributions of factual and hallucinated examples on the target model, and the annotated AUROC is recomputed from the plotted sample. The main text KDE plot is one representative case; here we show that the same qualitative pattern holds across the selected same-family, cross-family, scale-matched, and scale-mismatched settings.

The node feature for block b is

$$\mathbf{x}_{i,b}^{\text{feat}}(\mathbf{Q}, \mathbf{A}_i) = \text{vec}\left(\bar{\mathbf{V}}_{\boldsymbol{\theta}_i, b}(\mathbf{Q}, \mathbf{A}_i)\right).$$

The relation between two block nodes is

$$R_i^{\text{feat}}(b, \tilde{b}; \mathbf{Q}, \mathbf{A}_i) = \left\|\text{vec}(\bar{\mathbf{V}}_{\boldsymbol{\theta}_i, b}(\mathbf{Q}, \mathbf{A}_i)) - \text{vec}(\bar{\mathbf{V}}_{\boldsymbol{\theta}_i, \tilde{b}}(\mathbf{Q}, \mathbf{A}_i))\right\|_2. \tag{91}$$

No decay term is applied, since feature-block indices do not carry a natural ordering. Applying Eq. (85) over $\mathcal{V}_i^{\text{feat}} = \{1, \ldots, B\}$ gives $\mathbf{z}_i^{\text{feat}}(\mathbf{Q}, \mathbf{A}_i)$.

The joint relation view uses each layer-token-feature triple (ℓ, t, b) as a node. The node set is

$$\mathcal{V}_i^{\text{joint}} = \mathbf{L}^{\text{sel}} \times \mathbf{T}^{\text{sel}} \times \{1, \ldots, B\}.$$

The node feature is

$$\mathbf{x}_{i,(\ell,t,b)}^{\text{joint}}(\mathbf{Q}, \mathbf{A}_i) = \bar{\mathbf{h}}_{\boldsymbol{\theta}_i, t}^{\ell}(\mathbf{Q}, \mathbf{A}_i)[\mathcal{I}_b].$$

The relation between two joint nodes is the layer- and token-decayed block distance:

$$R_i^{\text{joint}}\big((\ell, t, b), (\tilde{\ell}, \tilde{t}, \tilde{b}); \mathbf{Q}, \mathbf{A}_i\big) = s^{\text{lyr}}(\ell, \tilde{\ell})\, s^{\text{tok}}(t, \tilde{t})\, \left\|\bar{\mathbf{h}}_{\boldsymbol{\theta}_i, t}^{\ell}(\mathbf{Q}, \mathbf{A}_i)[\mathcal{I}_b] - \bar{\mathbf{h}}_{\boldsymbol{\theta}_i, \tilde{t}}^{\tilde{\ell}}(\mathbf{Q}, \mathbf{A}_i)[\mathcal{I}_{\tilde{b}}]\right\|_2. \tag{92}$$

Applying Eq. (85) over $\mathcal{V}_i^{\text{joint}}$ gives $\mathbf{z}_i^{\text{joint}}(\mathbf{Q}, \mathbf{A}_i)$.

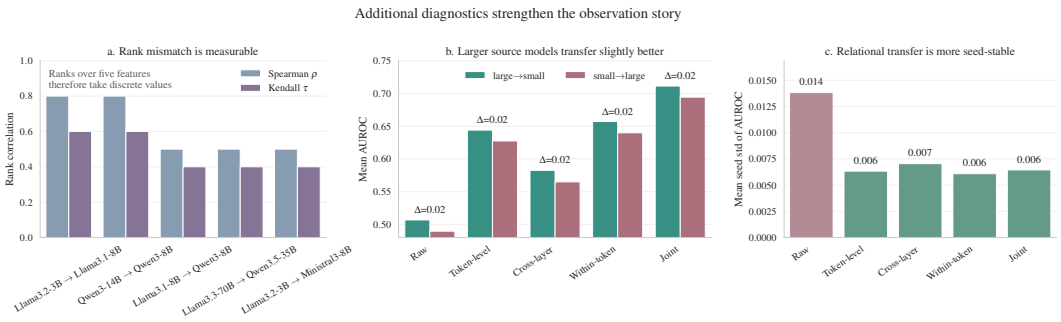

Figure 11: Additional diagnostics for the observation analysis. (A) Rank correlations between feature-view orderings induced by mutual k nearest neighbor alignment and by transfer AUROC. The moderate correlations indicate that alignment and downstream utility are related but not equivalent. Because the rank correlations are computed over five feature views, the possible Spearman and Kendall values are discrete. (B) Cross-scale transfer is mildly asymmetric: detectors trained on larger source models transfer slightly better to smaller target models than the reverse direction. (C) Raw hidden states exhibit higher seed variance in transfer AUROC, while relational views are more stable across random seeds.

Structural alignment. For each feature view $q \in \{\mathrm{raw}, \mathrm{tok}, \mathrm{lyr}, \mathrm{feat}, \mathrm{joint}\}$, we compute mutual k nearest-neighbor alignment between two LLMs $\mathbb{P}_{\boldsymbol{\theta}_i}$ and $\mathbb{P}_{\boldsymbol{\theta}_j}$ over the same evaluation questions. For sample index s, let $\mathcal{N}_k^{i,q}(s)$ denote the indices of the k nearest neighbors of the s-th example under feature view q for model i:

$$\mathcal{N}_k^{i,q}(s) = \mathrm{Top}\, k_{s' \in \{1,\dots,n\} \setminus \{s\}}\, \mathrm{sim}_i^q(s, s'),$$

where

$$\mathrm{sim}_i^q(s, s') = \left\langle \mathbf{z}_i^q(\mathbf{Q}^s, \mathbf{A}_i^s), \mathbf{z}_i^q(\mathbf{Q}^{s'}, \mathbf{A}_i^{s'}) \right\rangle.$$

The mutual kNN alignment is then

$$m_{\mathrm{NN}}^q(i,j) = \frac{1}{nk} \sum_{s=1}^{n} \left| \mathcal{N}_k^{i,q}(s) \cap \mathcal{N}_k^{j,q}(s) \right|. \tag{93}$$

This quantity measures whether the local neighborhood structure induced by a feature view is preserved across two LLMs. Since it is symmetric in i and j, Figure 8 shows the upper-triangular matrix and sets the diagonal entries to one by convention.

Predictive transfer. Structural alignment alone does not fully determine whether a feature view is useful for hallucination detection. We therefore also measure directional source-to-target detector transfer. For a source model $\mathbb{P}_{\boldsymbol{\theta}_i}$ and a target model $\mathbb{P}_{\boldsymbol{\theta}_j}$, an MLP detector is trained on

$$\{(\mathbf{z}_i^q(\mathbf{Q}^s, \mathbf{A}_i^s), y_i^s)\}_{s=1}^n$$

from $\mathcal{D}_i$ and evaluated directly on

$$\{(\mathbf{z}_j^q(\mathbf{Q}^s, \mathbf{A}_j^s), y_j^s)\}_{s=1}^n$$

from $\mathcal{D}_j$. The resulting AUROC is denoted by $a_{\mathrm{tr}}^q(i,j)$. Unlike mutual kNN alignment, transfer AUROC is directional because the detector is trained on one model and evaluated on another. Accordingly, Figure 9 reports the full source-to-target matrices.

Score-distribution diagnostic. To examine why AUROC changes across feature views, we also visualize the detector score distributions on the target model. The detector score is oriented so that larger values indicate stronger evidence for the hallucinated label. For a trained detector from source i to target j, the predicted scores on factual and hallucinated target examples are plotted separately. If the two distributions strongly overlap, the detector cannot distinguish the two labels after transfer. If the distributions separate, the detector has learned a feature direction that remains discriminative on the target model. Figure 10 shows that raw hidden states yield highly overlapping score distributions, whereas relational views, especially the joint relational structure, produce clearer separation between factual and hallucinated examples.

Rank diagnostic for alignment–utility mismatch. Finally, we quantify whether the feature ordering induced by alignment agrees with the ordering induced by transfer AUROC. For each source-target pair (i, j), we rank the five feature views by $m^q_{\mathrm{NN}}(i, j)$ and by $a^q_{\mathrm{tr}}(i, j)$, and compute Spearman's ρ and Kendall's τ between the two rankings. When ties occur, we use standard average ranks. Because there are only five feature views, these rank-correlation values are discrete; nevertheless, they provide a compact diagnostic of whether alignment and downstream utility agree. Figure 11a shows that the correlations are positive but moderate, indicating that alignment and detector transfer are related but not equivalent.

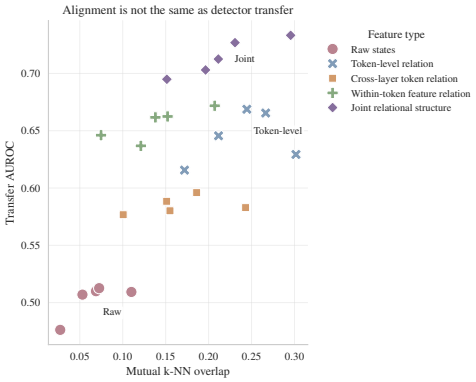

Figure 12: Alignment–utility decoupling. Each point denotes one relational construction on a representative source-to-target pair. Token-level relations often achieve high mutual kNN alignment, whereas joint relational structures yield stronger transfer AUROC. The most stable relational structure is not necessarily the most useful for hallucination detection.

Observations. Figures 12, 8, 9, 10, and 11a support three observations.

First, relational views are more stable across LLMs than raw hidden states. In the full mutual kNN matrices, raw-state features produce low neighborhood overlap across model pairs, while token-level, cross-layer, within-token, and joint relational views consistently increase local-neighborhood agreement. This suggests that relational features preserve cross-model structure that is largely absent from raw hidden coordinates.

Second, stronger alignment does not always imply stronger detector transfer. Token-level relations often achieve high mutual kNN overlap, but the joint relational structure yields stronger transfer AUROC. This mismatch is also visible in the rank-correlation diagnostics: the feature rankings induced by alignment and transfer are positively related, but they are not identical. One reason is that mutual kNN alignment measures local structural consistency over unlabeled examples, whereas AUROC depends on whether the transferred feature geometry separates factual and hallucinated examples on the target model.

Third, among the fixed hand-designed views considered here, the joint relational structure is the most consistently useful diagnostic for transfer. It combines layer, token, and feature-block relations, and therefore captures a richer class of cross-model patterns than any single relation axis. The score-distribution plots are consistent with this interpretation: joint relational features produce clearer separation between factual and hallucinated target examples across representative transfer pairs. Thus, the fixed handcrafted analysis motivates the learnable relation-aware model in Section 4: useful transfer signals are relational, but the most aligned handcrafted view is not necessarily the most discriminative one. These diagnostics motivate moving from fixed handcrafted relation graphs to learnable, input-dependent relation aggregation.

F.2 Handcrafted Relation Priors Help but Saturate

Section 3 uses deliberately simple hand-designed relations to test whether relational structure provides transferable hallucination signals. These simple relations already improve both cross-model alignment and detector transfer, but they leave open a natural question: can stronger handcrafted edge priors remove the need for learned relation aggregation? This section studies that question by comparing the simple relations in Section 3 with more refined fixed variants. The node sets, relation-weighted readout, detector architecture, and evaluation protocol are kept the same as in Section F.1; only the edge relation values are changed. Thus, any difference in Figure 13 reflects the effect of the handcrafted edge prior itself.

Simple relations used as the reference. We consider three relation families from Section 3: token-level relations, cross-layer token relations, and within-token feature relations. For each family, the reference edge is the simple relation used in the main observational analysis.

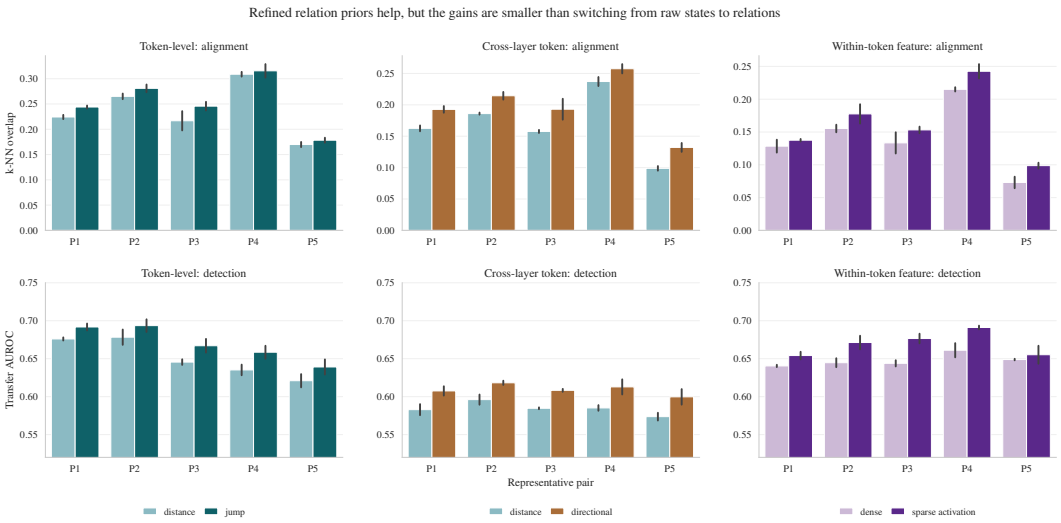

Figure 13: Hand-designed relation edges improve transfer, but their gains saturate. Simple relation edges are compared with refined handcrafted variants for token, layer, and intra-block relations. Across representative model pairs, refined edges improve both local neighborhood alignment and source-to-target AUROC. The improvements are smaller than the gain from moving from raw hidden states to relational views and are not uniform across relation types. Fixed graphs are useful diagnostics, but their instance-independent structural bias motivates learned relation aggregation. Here P1 to P5 denote Llama-3B→Llama-8B, Qwen-14B→Qwen-8B, Llama-8B→Qwen-8B, Llama-70B→Qwen-35B-A3B, and Llama-3B→Ministral-8B.

For token-level relations, the simple edge between two selected tokens t and $\tilde{t}$ at a selected diagnostic layer ℓ is the token-distance relation

$$R_i^{\mathrm{tok}}(t,\tilde{t};\mathbf{Q},\mathbf{A}_i) = s^{\mathrm{tok}}(t,\tilde{t})\left\|\bar{\mathbf{h}}^{\ell}_{\boldsymbol{\theta}_i,t}(\mathbf{Q},\mathbf{A}_i) - \bar{\mathbf{h}}^{\ell}_{\boldsymbol{\theta}_i,\tilde{t}}(\mathbf{Q},\mathbf{A}_i)\right\|_2. \tag{94}$$

This relation captures the magnitude of hidden-state separation between two token positions, modulated by their positional distance.

For cross-layer token relations, the simple edge between two selected layers ℓ and $\tilde{\ell}$ is the depth-decayed cosine distance between their selected-token summaries:

$$R_i^{\mathrm{lyr}}(\ell,\tilde{\ell};\mathbf{Q},\mathbf{A}_i) = s^{\mathrm{lyr}}(\ell,\tilde{\ell})\left(1-\cos\left(\mathrm{vec}(\bar{\mathbf{H}}^{\ell,\mathrm{sel}}_{\boldsymbol{\theta}_i}),\mathrm{vec}(\bar{\mathbf{H}}^{\tilde{\ell},\mathrm{sel}}_{\boldsymbol{\theta}_i})\right)\right), \tag{95}$$

where $\bar{\mathbf{H}}^{\ell,\mathrm{sel}}_{\boldsymbol{\theta}_i}$ is defined in Eq. (87). This relation compares the layer summaries themselves, but it does not explicitly encode how the hidden states evolve from one layer to the next.

For within-token feature relations, the simple edge between two feature blocks b and $\tilde{b}$ is the dense Euclidean distance

$$R_i^{\mathrm{feat}}(b,\tilde{b};\mathbf{Q},\mathbf{A}_i) = \left\|\mathrm{vec}(\bar{\mathbf{V}}_{\boldsymbol{\theta}_i,b}(\mathbf{Q},\mathbf{A}_i)) - \mathrm{vec}(\bar{\mathbf{V}}_{\boldsymbol{\theta}_i,\tilde{b}}(\mathbf{Q},\mathbf{A}_i))\right\|_2, \tag{96}$$

where $\bar{\mathbf{V}}_{\boldsymbol{\theta}_i,b}$ is defined in Eq. (90). This relation uses all coordinates in each block equally.

Refined token relation: state-jump edges. The simple token relation in Eq. (94) compares token hidden states directly. A more structured token relation can also incorporate how the hidden state changes as generation moves from one token to the next. For a token position t, let t^- denote its immediate predecessor in the token sequence. We define the token-state jump at layer ℓ by

$$\Delta^{\ell}_{i,t}(\mathbf{Q},\mathbf{A}_i) = \bar{\mathbf{h}}^{\ell}_{\boldsymbol{\theta}_i,t}(\mathbf{Q},\mathbf{A}_i) - \bar{\mathbf{h}}^{\ell}_{\boldsymbol{\theta}_i,t^-}(\mathbf{Q},\mathbf{A}_i). \tag{97}$$

If the predecessor is not available within the selected sequence, we use the nearest available previous token; this affects only boundary positions. The jump-refined token relation is then

$$R_i^{\text{tok-jump}}(t,\tilde{t};\mathbf{Q},\mathbf{A}_i) = s^{\mathrm{tok}}(t,\tilde{t})\left(\left\|\bar{\mathbf{h}}^{\ell}_{\boldsymbol{\theta}_i,t} - \bar{\mathbf{h}}^{\ell}_{\boldsymbol{\theta}_i,\tilde{t}}\right\|_2 + \lambda_{\mathrm{jump}}\left\|\Delta^{\ell}_{i,t} - \Delta^{\ell}_{i,\tilde{t}}\right\|_2\right), \tag{98}$$

where $\lambda_{\text{jump}} \geq 0$ controls the contribution of the state-jump term. The first term is the original token-state distance, while the second term compares local token-to-token transition patterns. Thus, this refined edge tests whether transfer improves when the token relation captures not only where token states are located, but also how they change along the generated sequence.

Refined cross-layer relation: directional layer-evolution edges. The simple cross-layer relation in Eq. (95) compares the layer summaries themselves. However, two layers can have similar summaries while participating in different layer-to-layer transformations. To test whether layer-evolution directions provide additional transferable signal, we define a directional layer update.

For each selected layer $\ell \in \mathbf{L}^{\text{sel}}$, let ℓ^+ be the next selected layer larger than ℓ when it exists, and let ℓ^- be the previous selected layer otherwise. We define

$$\Delta_i^{\ell}(\mathbf{Q}, \mathbf{A}_i) = \begin{cases} \operatorname{vec}\left(\bar{\mathbf{H}}_{\boldsymbol{\theta}_i}^{\ell^+,\text{sel}}(\mathbf{Q}, \mathbf{A}_i) - \bar{\mathbf{H}}_{\boldsymbol{\theta}_i}^{\ell,\text{sel}}(\mathbf{Q}, \mathbf{A}_i)\right), & \text{if } \ell^+ \text{ exists}, \\ \operatorname{vec}\left(\bar{\mathbf{H}}_{\boldsymbol{\theta}_i}^{\ell,\text{sel}}(\mathbf{Q}, \mathbf{A}_i) - \bar{\mathbf{H}}_{\boldsymbol{\theta}_i}^{\ell^-,\text{sel}}(\mathbf{Q}, \mathbf{A}_i)\right), & \text{otherwise}. \end{cases} \tag{99}$$

This vector summarizes the local direction of representation change around layer ℓ. The directional cross-layer relation is

$$R_i^{\text{lyr-dir}}(\ell, \tilde{\ell}; \mathbf{Q}, \mathbf{A}_i) = s^{\text{lyr}}(\ell, \tilde{\ell}) \left[\left(1 - \cos\left(\operatorname{vec}(\bar{\mathbf{H}}_{\boldsymbol{\theta}_i}^{\ell,\text{sel}}), \operatorname{vec}(\bar{\mathbf{H}}_{\boldsymbol{\theta}_i}^{\tilde{\ell},\text{sel}})\right)\right) + \lambda_{\text{dir}} \left(1 - \cos\left(\Delta_i^{\ell}, \Delta_i^{\tilde{\ell}}\right)\right) \right], \tag{100}$$

where $\lambda_{\text{dir}} \geq 0$ controls the contribution of the directional term. This edge keeps the original layer-summary comparison but additionally measures whether two layers induce similar local transformation directions. It therefore tests whether cross-model transfer benefits from handcrafted information about layerwise evolution, beyond static layer-summary distance.

Refined within-token feature relation: sparse-activation edges. The simple within-token feature relation in Eq. (96) treats all coordinates in each feature block as dense signals. However, feature blocks can contain many weakly active or noisy coordinates. To test whether emphasizing high-activation coordinates improves transfer, we define a sparse-activation variant.

Let

$$\mathbf{v}_{i,b}(\mathbf{Q}, \mathbf{A}_i) = \operatorname{vec}\left(\bar{\mathbf{V}}_{\boldsymbol{\theta}_i,b}(\mathbf{Q}, \mathbf{A}_i)\right)$$

be the dense vector for block b. Let $\mathbf{m}_{i,b} \in \{0,1\}^{\dim(\mathbf{v}_{i,b})}$ be a binary activation mask that keeps the largest coordinates of $|\mathbf{v}_{i,b}|$ and sets the remaining coordinates to zero. The sparsified block vector is

$$\operatorname{sp}(\mathbf{v}_{i,b}) = \mathbf{m}_{i,b} \odot \mathbf{v}_{i,b}, \tag{101}$$

where $\odot$ denotes element-wise multiplication. The sparse-activation relation between two feature blocks is

$$R_i^{\text{feat-sp}}(b, \tilde{b}; \mathbf{Q}, \mathbf{A}_i) = \left\| \operatorname{sp}(\mathbf{v}_{i,b}(\mathbf{Q}, \mathbf{A}_i)) - \operatorname{sp}(\mathbf{v}_{i,\tilde{b}}(\mathbf{Q}, \mathbf{A}_i)) \right\|_2. \tag{102}$$

This relation compares feature blocks after suppressing weak activations, and therefore tests whether transfer improves when the handcrafted relation emphasizes more active coordinates.

Evaluation protocol. For each refined relation, we keep the same node set and readout as its simple counterpart. Specifically, token-distance and token-jump relations are both evaluated on the token-node graph; dense and sparse-activation feature relations are both evaluated on the feature-block graph; layer-summary and directional layer-evolution relations are both evaluated on the layer-node graph. After computing the corresponding graph-level features with Eq. (85), we evaluate the two criteria from Section F.1: mutual kNN alignment and directional cross-model AUROC.

Figure 13 reports results on five representative source-to-target transfer pairs, denoted P1–P5. These pairs cover same-family transfer, cross-family transfer, scale-matched transfer, and scale-mismatched transfer. For each relation family, the lighter bar denotes the simple reference relation and the darker bar denotes the refined handcrafted relation.

Observations. Figure 13 shows that refined handcrafted priors generally improve both local-neighborhood alignment and transfer AUROC. Token-state jump edges improve over token-distance edges, suggesting that local state-transition patterns contain additional transferable information

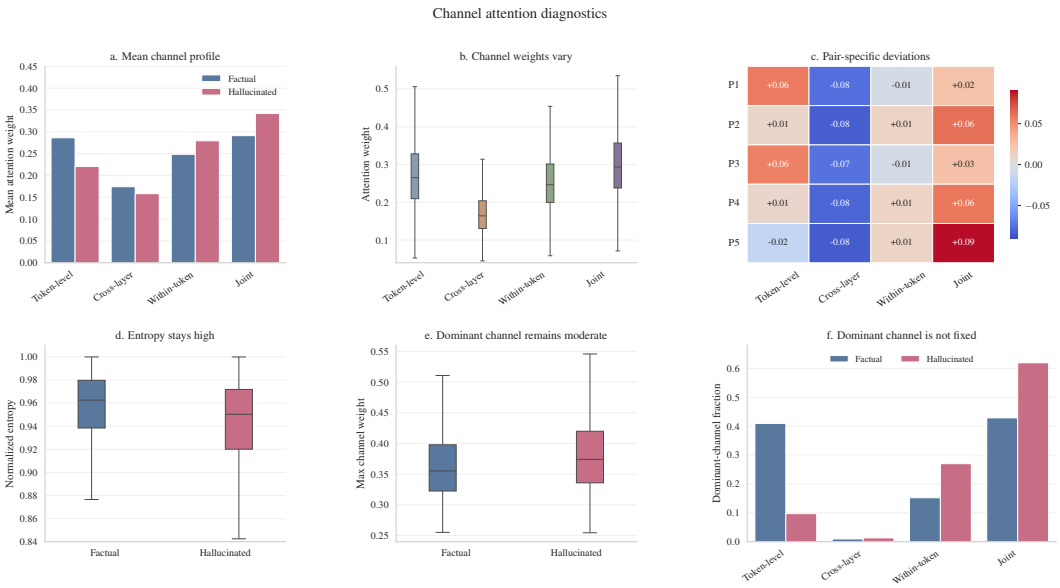

Figure 14: Channel attention diagnostics. A lightweight channel probe is trained on source-model relational features and evaluated on target-model examples, with all relational features frozen. The four channel weights sum to one for each example. Panels (A) and (B) show label-conditioned mean profiles and per-example channel-weight variation. Panel (C) shows pair-specific deviations after subtracting each pair's mean channel weight, so colors emphasize relative channel preference within each transfer pair. Panels (D) and (E) report normalized entropy and maximum channel weight on separate axes. Panel (F) shows the dominant channel frequency. Together, these diagnostics indicate that the useful relation mixture is not well described by a single fixed channel weighting.

beyond static token-state separation. Directional layer-evolution edges improve over static layer-summary edges, indicating that layer-to-layer transformation directions provide useful complementary information. Sparse-activation feature edges improve over dense feature-block distances in several settings, suggesting that suppressing weak activations can reduce noise in feature-block relations.

However, the gains from these refined handcrafted priors are modest compared with the larger improvement obtained by moving from raw hidden states to relational views in the first place. The improvements are also not uniform across relation families and transfer pairs. For example, the refined token relation and refined layer relation do not yield the same magnitude of gain, and the feature-block refinement behaves differently from the token and layer refinements. This indicates that better handcrafted priors can help, but choosing a single fixed relation formula is unlikely to fully capture the relation structure needed for robust cross-model hallucination detection.

Implication. These results suggest that fixed relation priors are useful diagnostics but have limited headroom. More sophisticated handcrafted edges can encode additional intuitions, such as token-state transitions, layerwise evolution directions, or sparse activation patterns, but their benefits saturate and remain relation-type dependent. This motivates the learnable relation-aware aggregation in Section 4. Rather than manually selecting one refined edge formula for all examples and all model pairs, CHORD learns how to combine positional and content-based relation signals in an input-dependent way.

F.3 Relation-Channel Importance Is Input-Dependent

Sections F.1 and F.2 show that different relational views capture different transferable structures. However, these results do not imply that a single relation view should be used for every example. A hallucinated answer may differ from a factual answer through token-level inconsistency, abnormal layer evolution, feature-block changes, or a combination of these effects. Therefore, we further ask whether the importance of different relation channels is fixed, or whether it changes across examples, labels, and source-target model pairs.

Diagnostic channel-gated probe. We use the four fixed relational features from Section F.1:

$$\mathbf{z}_i^{\mathrm{tok}}(\mathbf{Q}, \mathbf{A}_i), \quad \mathbf{z}_i^{\mathrm{lyr}}(\mathbf{Q}, \mathbf{A}_i), \quad \mathbf{z}_i^{\mathrm{feat}}(\mathbf{Q}, \mathbf{A}_i), \quad \mathbf{z}_i^{\mathrm{joint}}(\mathbf{Q}, \mathbf{A}_i).$$

These four views are treated as four relation channels. Let

$$\mathcal{C} = \{\mathrm{tok}, \mathrm{lyr}, \mathrm{feat}, \mathrm{joint}\}$$

denote the set of channels. Since the four relational features can have different dimensions, we first map each channel into the same diagnostic dimension:

$$\mathbf{r}_i^q(\mathbf{Q}, \mathbf{A}_i) = g_q(\mathbf{z}_i^q(\mathbf{Q}, \mathbf{A}_i)) \in \mathbb{R}^{d_c}, \qquad q \in \mathcal{C}, \tag{103}$$

where $g_q(\cdot)$ is a small learnable projection for channel q. The projected channel vectors are then assigned input-dependent channel weights. For each channel q, we compute a channel score

$$a_i^q(\mathbf{Q}, \mathbf{A}_i) = \mathbf{w}_c^\top \tanh\left(\mathbf{W}_c \mathbf{r}_i^q(\mathbf{Q}, \mathbf{A}_i) + \mathbf{b}_c\right), \tag{104}$$

and normalize the scores over the four channels:

$$\alpha_i^q(\mathbf{Q}, \mathbf{A}_i) = \frac{\exp(a_i^q(\mathbf{Q}, \mathbf{A}_i))}{\sum_{q' \in \mathcal{C}} \exp(a_i^{q'}(\mathbf{Q}, \mathbf{A}_i))}, \qquad q \in \mathcal{C}. \tag{105}$$

The fused diagnostic representation is

$$\mathbf{z}_i^{\mathrm{mix}}(\mathbf{Q}, \mathbf{A}_i) = \sum_{q \in \mathcal{C}} \alpha_i^q(\mathbf{Q}, \mathbf{A}_i) \mathbf{r}_i^q(\mathbf{Q}, \mathbf{A}_i). \tag{106}$$

A binary MLP detector is then trained on $\mathbf{z}_i^{\mathrm{mix}}(\mathbf{Q}, \mathbf{A}_i)$ using the same source-to-target transfer protocol as in Section F.1. For a source model $\mathbb{P}_{\boldsymbol{\theta}_i}$ and a target model $\mathbb{P}_{\boldsymbol{\theta}_j}$, the detector and channel-gating module are trained on $\mathcal{D}_i$ and evaluated on $\mathcal{D}_j$. After training, we record the channel weights $\alpha_j^q(\mathbf{Q}^s, \mathbf{A}_j^s)$ on target examples.

This channel-gated probe is used only for analysis. The weights α_i^q should not be interpreted as causal explanations. They are diagnostic quantities that describe how the probe allocates mass across the fixed relational channels when forming its prediction.

Label-conditioned channel profiles. We first examine whether factual and hallucinated examples use the same channel mixture. Recall that $y_j^s = 1$ indicates that the generated answer is truthful, and $y_j^s = 0$ indicates hallucination. For each label $y \in \{0, 1\}$ and channel $q \in \mathcal{C}$, we compute the label-conditioned mean channel weight on the target model:

$$\bar{\alpha}_{j,y}^q = \frac{1}{|\mathcal{S}_{j,y}|} \sum_{s \in \mathcal{S}_{j,y}} \alpha_j^q(\mathbf{Q}^s, \mathbf{A}_j^s), \qquad \mathcal{S}_{j,y} = \{s : y_j^s = y\}. \tag{107}$$

If relation-channel importance were fixed, the profiles $\{\bar{\alpha}_{j,0}^q\}_{q \in \mathcal{C}}$ and $\{\bar{\alpha}_{j,1}^q\}_{q \in \mathcal{C}}$ would be nearly identical. Differences between the two profiles indicate that the detector uses relational information differently for hallucinated and factual examples.

Per-example variation. Average profiles alone may hide substantial example-level variation. For each target example, we therefore analyze the full channel-weight vector

$$\boldsymbol{\alpha}_j^s = \left[\alpha_j^{\mathrm{tok}}(\mathbf{Q}^s, \mathbf{A}_j^s), \alpha_j^{\mathrm{lyr}}(\mathbf{Q}^s, \mathbf{A}_j^s), \alpha_j^{\mathrm{feat}}(\mathbf{Q}^s, \mathbf{A}_j^s), \alpha_j^{\mathrm{joint}}(\mathbf{Q}^s, \mathbf{A}_j^s)\right].$$

We report the distribution of these weights across examples. If the probe used a fixed relation view, then almost all examples would place their largest weight on the same channel and the per-example weight distribution would be nearly constant. In contrast, broad distributions indicate that the relevant relational signal varies across inputs.

Pair-specific channel deviations. The same relation channel may also have different utility for different source-target model pairs. For each transfer pair (i, j), we compute the average target-side channel profile

$$\bar{\alpha}^q_{i\to j} = \frac{1}{n}\sum_{s=1}^{n} \alpha^q_j(\mathbf{Q}^s, \mathbf{A}^s_j), \qquad q \in \mathcal{C}. \tag{108}$$

We then center each profile by the uniform channel weight:

$$\delta^q_{i\to j} = \bar{\alpha}^q_{i\to j} - \frac{1}{|\mathcal{C}|}\sum_{q'\in\mathcal{C}} \bar{\alpha}^{q'}_{i\to j} = \bar{\alpha}^q_{i\to j} - \frac{1}{|\mathcal{C}|}, \tag{109}$$

where the second equality holds because the channel weights of a transfer pair sum to one. A positive $\delta^q_{i\to j}$ means that the pair allocates above-uniform weight to channel q, and differences between rows show how channel preferences vary across transfer pairs. This diagnostic is useful because different LLM families may differ in their layer dynamics, token-level geometry, or feature-block organization.

Entropy, maximum weight, and dominant channel. We further summarize the sharpness of the channel distribution. For each example, the normalized channel entropy is

$$H^s_j = -\frac{1}{\log|\mathcal{C}|}\sum_{q\in\mathcal{C}} \alpha^q_j(\mathbf{Q}^s, \mathbf{A}^s_j)\log\left(\alpha^q_j(\mathbf{Q}^s, \mathbf{A}^s_j) + \epsilon\right), \tag{110}$$

where $\epsilon > 0$ is a small numerical-stability constant. A value close to zero would indicate that almost all weight is assigned to a single channel, while a value close to one would indicate a nearly uniform mixture. We also compute the maximum channel weight

$$M^s_j = \max_{q\in\mathcal{C}} \alpha^q_j(\mathbf{Q}^s, \mathbf{A}^s_j), \tag{111}$$

and the dominant channel

$$q^s_j = \arg\max_{q\in\mathcal{C}} \alpha^q_j(\mathbf{Q}^s, \mathbf{A}^s_j). \tag{112}$$

The empirical frequency of q^s_j over examples indicates whether one relation channel dominates the diagnostic probe or whether different channels are selected for different inputs.

Observations. Figure 14 supports four observations.

First, channel importance differs between factual and hallucinated examples. The label-conditioned profiles are not identical: hallucinated and factual examples allocate different average weights to the token, layer, feature, and joint channels. This suggests that hallucination-related signals are not confined to a single relational axis. Instead, the detector uses different mixtures of relation channels depending on the label-conditioned structure of the example.

Second, the channel weights vary substantially across individual examples. The per-example distributions show that the probe does not assign a constant weight vector to all inputs. Some examples emphasize token-level relations, while others place more mass on feature-block, layer, or joint relations. This variation is consistent with the view that hallucinations can manifest through different internal patterns depending on the question, generated answer, and target model.

Third, channel usage is source-target dependent. The pair-specific deviations show that different transfer pairs shift weight toward different channels. This is expected because LLMs with different architectures, scales, or training procedures need not share the same token geometry, layer evolution, or feature-block organization. Thus, a fixed channel mixture that works well for one source-target pair may be suboptimal for another.

Fourth, the probe does not collapse to a single relation channel. The normalized entropy remains away from zero, the maximum channel weight is not consistently close to one, and the dominant-channel frequency is distributed across multiple channels. This indicates that the diagnostic probe combines relation channels rather than using only one of them. At the same time, the entropy is not exactly one, so the mixture is not purely uniform. The probe therefore learns nontrivial, input-dependent channel preferences.

Implication. These diagnostics support the need for input-dependent relation aggregation. The earlier sections show that relational views are more transferable than raw hidden states, and that refined handcrafted priors can help but saturate. The channel-weight analysis adds a complementary point: even among useful relation views, the relevant channel is not fixed across labels, examples, or source-target pairs. Therefore, a detector that commits to one handcrafted relation view, or to one fixed mixture of relation channels, is unlikely to be optimal across heterogeneous LLMs. This motivates the relation-aware attention mechanism in Section 4, where positional and content-based relation signals are combined adaptively for each input rather than selected by a fixed manual rule.

F.4 Additional Robustness and Transfer Diagnostics

We provide two additional diagnostics to examine whether the transfer advantage of relational views depends on a particular source-target scale direction or on detector training randomness. Both diagnostics use the same five feature views as Section F.1: raw hidden states, token-level relations, cross-layer token relations, within-token feature relations, and joint relational structures. For a source model $\mathbb{P}_{\boldsymbol{\theta}_i}$ and a target model $\mathbb{P}_{\boldsymbol{\theta}_j}$, we denote the cross-model transfer AUROC of feature view q by $a^q_{\mathrm{tr}}(i,j)$.

Scale direction. Since cross-model transfer is directional, training on a larger source model and testing on a smaller target model may not behave the same as the reverse direction. Let $\mathrm{size}(i)$ denote the parameter scale of $\mathbb{P}_{\boldsymbol{\theta}_i}$. We compare two sets of directed transfer pairs:

$$\mathcal{P}_{\mathrm{large}\to\mathrm{small}} = \{(i,j) : \mathrm{size}(i) > \mathrm{size}(j)\}, \qquad \mathcal{P}_{\mathrm{small}\to\mathrm{large}} = \{(i,j) : \mathrm{size}(i) < \mathrm{size}(j)\}.$$

For each feature view q, we compute the average AUROC in both directions and report their difference:

$$\Delta^q_{\mathrm{scale}} = \frac{1}{|\mathcal{P}_{\mathrm{large}\to\mathrm{small}}|} \sum_{(i,j)\in\mathcal{P}_{\mathrm{large}\to\mathrm{small}}} a^q_{\mathrm{tr}}(i,j) - \frac{1}{|\mathcal{P}_{\mathrm{small}\to\mathrm{large}}|} \sum_{(i,j)\in\mathcal{P}_{\mathrm{small}\to\mathrm{large}}} a^q_{\mathrm{tr}}(i,j).$$

Figure 11b shows that large-source to small-target transfer is slightly stronger than small-source to large-target transfer for most feature views. This is consistent with the intuition that larger source models may expose richer internal structures for training the detector. However, the gap is modest compared with the difference between raw hidden states and relational views. Thus, the main observation is not driven by a single favorable scale direction: relational features improve transfer in both directions, while scale direction only mildly modulates the absolute AUROC.

Seed stability. We also examine whether the transfer results are sensitive to the random seed used for training the downstream MLP detector. For each directed pair (i,j) and feature view q, we train the detector with multiple random seeds and denote the resulting AUROC by $a^{q,r}_{\mathrm{tr}}(i,j)$, where r indexes the seed. The seed standard deviation is

$$\sigma^q_{\mathrm{seed}}(i,j) = \left(\frac{1}{R-1} \sum_{r=1}^{R} \left(a^{q,r}_{\mathrm{tr}}(i,j) - \frac{1}{R} \sum_{r'=1}^{R} a^{q,r'}_{\mathrm{tr}}(i,j) \right)^2 \right)^{1/2}.$$

We then average $\sigma^q_{\mathrm{seed}}(i,j)$ over all evaluated directed transfer pairs. Figure 11c shows that raw hidden states have noticeably higher seed variance, whereas relational views yield more stable detector performance. This suggests that raw-state transfer often depends on unstable decision boundaries, while relational features provide a more consistent basis for cross-model training.

Implication. Together, these diagnostics support the robustness of the observational findings. The advantage of relational views is not explained by choosing an easy source-target scale direction, and it is not a consequence of lucky detector initialization. Large-to-small transfer is mildly easier than small-to-large transfer, but the effect is secondary. Relational views are both more transferable and more stable than raw hidden states, which further motivates using relational structure as the basis for cross-model hallucination detection.

G Additional Experimental Analysis

Expanded analysis of the main cross-model results. Table 1 evaluates hallucination detectors under the leave-one-family-out cross-model setting, where training-based methods are trained on

source model families and evaluated on a held-out target family. This setting directly tests whether a detector can transfer beyond the representation space and generation behavior of the models used for training. Overall, CHORD achieves the best AUROC across all target models and all seven benchmarks. These results show that the proposed relational graph discrimination framework remains effective across factual question answering, conversational question answering, mathematical reasoning, coding, and agent-oriented tasks.

The comparison with training-free methods reveals a clear pattern. Methods such as Perplexity, Semantic Entropy, EigenScore, Lexical Similarity, Verbalize, and InterrogateLLM do not train source-model-specific classifiers, and therefore largely avoid training-induced cross-model generalization issues. Among them, InterrogateLLM and Semantic Entropy are usually the strongest baselines, suggesting that consistency and uncertainty signals are useful model-agnostic indicators. Nevertheless, these methods rely on indirect output-level evidence, such as likelihood, semantic variability, lexical overlap, or self-evaluation. As a result, their performance remains below CHORD on all benchmarks in the averaged results. This indicates that output-level signals are useful but do not fully capture hallucination-relevant structure in the model's internal computation.

The representative training-based baselines show a different failure mode. MM, SEP, SAPLMA, and LapEigvals use supervised or structured signals for hallucination detection, but their cross-model performance is often close to random guessing. This does not imply that these methods are ineffective in easier within-model settings. Rather, the results suggest that detectors learned from source models can rely on regularities that do not transfer to unseen target models. For hidden-state-based probes, this is consistent with the cross-model semantic inconsistency discussed in Section 1. For attention-based spectral statistics such as LapEigvals, the results suggest that fixed structural descriptors alone are not sufficient to provide robust cross-family discrimination. The contrast between training-free and training-based baselines highlights the central challenge of CMHD. Training-free methods are easier to apply across models but have limited discriminative strength, whereas supervised detectors can be discriminative but often become brittle under cross-model transfer. CHORD bridges this gap by learning from labeled source-model data while representing hidden states through relational graph structures that reduce reliance on model-specific raw features.

Dataset-level observations. The largest gains of CHORD appear on xLAM and GSM8K. On xLAM, CHORD reaches an average AUROC of 0.8564, clearly exceeding the strongest baseline. This suggests that relational hidden-state structures are particularly informative for agent-oriented hallucination detection. In such settings, errors may involve tool selection, action arguments, or intermediate decision trajectories, which can induce structured deviations across tokens, layers, and feature groups. On GSM8K, CHORD also achieves strong performance, with an average AUROC of 0.8298. This is consistent with the intuition that reasoning errors are not always isolated local mistakes, but may appear as structured changes along the internal reasoning trajectory.

The gains are also stable on standard QA and hallucination benchmarks. On TruthfulQA, HaluEval, TriviaQA, and CoQA, CHORD consistently outperforms both training-free and training-based baselines. This shows that the advantage of relational graph discrimination is not limited to specialized reasoning or agent tasks, but also applies to conventional factual and conversational question answering. Among these datasets, TriviaQA and CoQA are relatively favorable to lexical or consistency-based baselines, since answer correctness can sometimes be correlated with surface overlap or answer agreement. Even in these settings, CHORD maintains a clear advantage, suggesting that internal relational structures provide complementary information beyond output-level similarity.

MBPP is the most challenging benchmark for CHORD. Although CHORD still achieves the best average AUROC on MBPP, its margin over the strongest training-free baseline is smaller than on other datasets. This is expected for two reasons. First, MBPP contains fewer than one thousand programming problems, so the available supervision remains limited even after collecting generations from multiple LLMs. Second, code correctness is strongly execution-dependent. A generated program can be semantically wrong for reasons that are not fully reflected in natural-language hidden-state relations. Therefore, the smaller margin on MBPP should be interpreted as a limitation of the current representation signal rather than a contradiction of the overall trend.

Model-family and scale observations. The results also show that larger target models often yield stronger detection performance, although the trend is not strictly monotonic. Within the Llama family, Llama-3.3-70B generally achieves higher AUROC than smaller Llama models under CHORD. A

Table 4: Full ablation results under the cross-family evaluation protocol. Values are AUROC. Best results are **bold** and second-best results are underlined.

Method	Target Model: Llama3.2-3B — Mean AUROC↑								Target Model: Llama3.1-8B — Mean AUROC↑							
	TruthfulQA	HaluEval	TriviaQA	CoQA	GSM8K	MBPP	xLAM	**Average**	TruthfulQA	HaluEval	TriviaQA	CoQA	GSM8K	MBPP	xLAM	**Average**
Training-free methods																
Perplexity	0.6329	0.6464	0.6448	0.6296	0.6536	0.6188	0.6621	0.6412	0.6546	0.6452	0.6545	0.6395	0.6523	0.6024	0.6857	0.6477
Semantic Entropy	0.6862	0.7095	0.7006	0.6934	0.7112	0.6571	0.7223	0.6972	0.6991	0.7113	0.7065	0.6919	0.7258	0.6636	0.7438	0.7060
EigenScore	0.6636	0.6872	0.6767	0.6719	0.6772	0.6508	0.7045	0.6760	0.6985	0.6963	0.6803	0.6747	0.7068	0.6614	0.6957	0.6877
Lexical Similarity	0.6542	0.6747	0.6878	0.6786	0.6454	0.6225	0.6603	0.6605	0.6760	0.6681	0.6928	0.6628	0.6495	0.6236	0.6871	0.6657
Verbalize	0.5199	0.5306	0.5220	0.5290	0.5207	0.5069	0.5470	0.5252	0.5238	0.5398	0.5110	0.5050	0.5474	0.4884	0.5595	0.5250
InterrogateLLM	0.6845	0.7353	0.7012	0.6915	0.7177	0.6915	0.7253	0.7067	0.7185	0.7211	0.7201	0.6881	0.7293	0.6865	0.7469	0.7158
Training-based methods																
MM	0.4857	0.5170	0.5129	0.4874	0.5061	0.5035	0.5200	0.5047	0.5127	0.5029	0.5247	0.5000	0.5141	0.4808	0.5052	0.5058
SEP	0.5079	0.5193	0.5373	0.5189	0.5300	0.5032	0.5290	0.5208	0.5191	0.5346	0.5142	0.5089	0.5382	0.4876	0.5395	0.5203
SAPLMA	0.4941	0.5191	0.5239	0.5390	0.5158	0.4901	0.5397	0.5174	0.5326	0.5311	0.5250	0.5013	0.5382	0.4818	0.5386	0.5212
LapEigvals	0.4817	0.5021	0.4740	0.4965	0.5264	0.4765	0.5295	0.4981	0.4981	0.4916	0.4712	0.4789	0.5083	0.4725	0.5225	0.4919
Ablation variants																
CHORD w/o RelAttn	0.6511	0.7258	0.7033	0.6800	0.7241	0.6458	0.7259	0.6937	0.7154	0.7230	0.7115	0.6594	0.7403	0.6678	0.7916	0.7156
CHORD w/o Meta	0.6883	0.7457	0.7194	0.6872	0.7394	0.6563	0.7594	0.7137	0.7339	0.7696	0.7442	0.6637	0.7538	0.6693	0.7975	0.7331
Proposed method																
CHORD	**0.7766**	**0.7926**	**0.7991**	**0.7920**	**0.8193**	**0.7351**	**0.8437**	**0.7941**	**0.7989**	**0.8102**	**0.8156**	**0.7823**	**0.8362**	**0.7307**	**0.8650**	**0.8056**

Method	Llama3.3-70B								Qwen3-8B							
Training-free methods																
Perplexity	0.6575	0.6747	0.6717	0.6748	0.6765	0.6353	0.6931	0.6691	0.6345	0.6398	0.6781	0.6542	0.6653	0.6179	0.6726	0.6518
Semantic Entropy	0.7096	0.7366	0.7231	0.7272	0.7410	0.6697	0.7406	0.7211	0.6840	0.6800	0.7127	0.6994	0.7143	0.6502	0.7228	0.6948
EigenScore	0.7076	0.7087	0.6961	0.7095	0.7094	0.6703	0.7093	0.7016	0.6785	0.6817	0.6869	0.6873	0.6813	0.6683	0.7085	0.6846
Lexical Similarity	0.6844	0.6797	0.7040	0.7088	0.6854	0.6432	0.6858	0.6845	0.6538	0.6538	0.7089	0.6966	0.6441	0.6401	0.6688	0.6666
Verbalize	0.5472	0.5531	0.5472	0.5385	0.5592	0.5139	0.5584	0.5454	0.5193	0.5090	0.5200	0.5170	0.5339	0.4969	0.5463	0.5203
InterrogateLLM	0.7343	0.7454	0.7472	0.7219	0.7473	0.7181	0.7477	0.7374	0.6950	0.7077	0.7044	0.7173	0.7082	0.6834	0.7483	0.7092
Training-based methods																
MM	0.5095	0.5244	0.5288	0.5195	0.5259	0.4814	0.5153	0.5150	0.4923	0.4849	0.5044	0.5095	0.5046	0.4837	0.5171	0.4995
SEP	0.5117	0.5352	0.5400	0.5243	0.5263	0.4905	0.5297	0.5225	0.5147	0.5030	0.4993	0.5062	0.5159	0.4823	0.5228	0.5063
SAPLMA	0.5250	0.5177	0.5231	0.5364	0.5375	0.4840	0.5386	0.5232	0.4822	0.5223	0.5171	0.5093	0.5167	0.4922	0.5384	0.5112
LapEigvals	0.5083	0.4992	0.5167	0.5204	0.5132	0.4718	0.5280	0.5082	0.4831	0.4872	0.5141	0.4873	0.5063	0.4716	0.5292	0.4970
Ablation variants																
CHORD w/o RelAttn	0.7420	0.7461	0.7515	0.7097	0.7646	0.7017	0.7821	0.7425	0.6634	0.6950	0.7147	0.7130	0.7233	0.6416	0.7602	0.7016
CHORD w/o Meta	0.7552	0.7652	0.7860	0.7220	0.8101	0.7151	0.7886	0.7632	0.6627	0.7019	0.7531	0.7256	0.7444	0.6577	0.7706	0.7166
Proposed method																
CHORD	**0.8276**	**0.8374**	**0.8257**	**0.8302**	**0.8568**	**0.7605**	**0.8713**	**0.8299**	**0.7896**	**0.7941**	**0.8192**	**0.7991**	**0.8256**	**0.7481**	**0.8484**	**0.8034**

Method	Qwen3-14B								Qwen3.5-35B							
Training-free methods																
Perplexity	0.6408	0.6650	0.6442	0.6519	0.6363	0.6164	0.6851	0.6485	0.6460	0.6488	0.6751	0.6601	0.6614	0.6361	0.6732	0.6572
Semantic Entropy	0.6810	0.6968	0.6966	0.6958	0.7193	0.6438	0.7195	0.6933	0.7078	0.7110	0.7211	0.7052	0.7392	0.6738	0.7278	0.7123
EigenScore	0.6931	0.6891	0.6965	0.6748	0.7013	0.6556	0.7090	0.6885	0.7061	0.7018	0.7072	0.6854	0.7093	0.6648	0.7046	0.6970
Lexical Similarity	0.6498	0.6560	0.6779	0.6795	0.6529	0.6368	0.6834	0.6623	0.6752	0.6756	0.7050	0.7083	0.6583	0.6377	0.6730	0.6762
Verbalize	0.5349	0.5262	0.5274	0.5261	0.5329	0.5091	0.5578	0.5306	0.5286	0.5352	0.5511	0.5279	0.5460	0.5150	0.5550	0.5370
InterrogateLLM	0.7009	0.7127	0.7062	0.7014	0.7496	0.6904	0.7435	0.7150	0.6978	0.6997	0.7248	0.7273	0.7438	0.6902	0.7376	0.7173
Training-based methods																
MM	0.5033	0.5228	0.5270	0.4910	0.5120	0.4966	0.5269	0.5114	0.4844	0.4817	0.5243	0.5240	0.5280	0.4848	0.5116	0.5055
SEP	0.5277	0.5040	0.5072	0.5170	0.5258	0.4980	0.5317	0.5159	0.5395	0.5139	0.5284	0.5222	0.5301	0.4874	0.5350	0.5224
SAPLMA	0.5109	0.5192	0.5226	0.5385	0.5287	0.4916	0.5370	0.5212	0.5198	0.5174	0.5268	0.5303	0.5240	0.4868	0.5392	0.5206
LapEigvals	0.4840	0.5049	0.5113	0.4969	0.5062	0.4822	0.5263	0.5017	0.4712	0.4891	0.5108	0.4829	0.5289	0.4727	0.5291	0.4978
Ablation variants																
CHORD w/o RelAttn	0.6538	0.7046	0.6887	0.6854	0.7645	0.6652	0.7594	0.7031	0.6988	0.7104	0.7213	0.7077	0.7609	0.6550	0.7640	0.7169
CHORD w/o Meta	0.7108	0.7158	0.6989	0.7329	0.7679	0.6913	0.7788	0.7281	0.7132	0.7289	0.7379	0.7214	0.7661	0.6515	0.7641	0.7262
Proposed method																
CHORD	**0.7720**	**0.8237**	**0.8143**	**0.8070**	**0.8387**	**0.7248**	**0.8722**	**0.8075**	**0.8069**	**0.8104**	**0.8333**	**0.8027**	**0.8406**	**0.7207**	**0.8472**	**0.8088**

Method	Ministral3-8B								Ministral3-14B							
Training-free methods																
Perplexity	0.6443	0.6328	0.6576	0.6397	0.6528	0.6308	0.6785	0.6481	0.6335	0.6510	0.6471	0.6357	0.6526	0.6029	0.6635	0.6409
Semantic Entropy	0.6828	0.6800	0.7148	0.6867	0.7045	0.6424	0.7338	0.6921	0.6882	0.6925	0.7006	0.6930	0.7104	0.6534	0.7283	0.6952
EigenScore	0.6856	0.6729	0.6918	0.6788	0.6836	0.6519	0.7076	0.6817	0.6668	0.7015	0.6846	0.6746	0.6926	0.6509	0.7090	0.6829
Lexical Similarity	0.6530	0.6474	0.7075	0.6906	0.6479	0.6265	0.6698	0.6632	0.6632	0.6595	0.6945	0.6788	0.6610	0.6255	0.6682	0.6644
Verbalize	0.5019	0.5116	0.5440	0.5172	0.5230	0.5112	0.5528	0.5231	0.5143	0.5340	0.5158	0.5075	0.5231	0.5048	0.5506	0.5214
InterrogateLLM	0.6963	0.7002	0.7030	0.6969	0.7254	0.6961	0.7446	0.7089	0.6989	0.7295	0.7097	0.6862	0.7163	0.6957	0.7432	0.7114
Training-based methods																
MM	0.4807	0.4840	0.5294	0.5123	0.5298	0.4853	0.5285	0.5071	0.4866	0.5095	0.5123	0.4854	0.5284	0.4924	0.5232	0.5054
SEP	0.5232	0.5188	0.5162	0.4990	0.5033	0.5030	0.5288	0.5132	0.4860	0.5076	0.5123	0.4999	0.5186	0.4895	0.5391	0.5076
SAPLMA	0.4816	0.5275	0.5396	0.5073	0.5335	0.4944	0.5392	0.5176	0.4865	0.5232	0.4952	0.5091	0.5266	0.4839	0.5386	0.5090
LapEigvals	0.4708	0.4737	0.5019	0.5031	0.4983	0.4733	0.5295	0.4929	0.4847	0.4766	0.5179	0.4800	0.5058	0.4702	0.5161	0.4930
Ablation variants																
CHORD w/o RelAttn	0.7051	0.6523	0.7346	0.6500	0.7148	0.6591	0.7514	0.6953	0.6816	0.7271	0.7152	0.6713	0.6829	0.6463	0.7328	0.6939
CHORD w/o Meta	0.7084	0.6484	0.7499	0.7029	0.7574	0.6821	0.7738	0.7176	0.7195	0.7341	0.7235	0.7115	0.7186	0.6456	0.7594	0.7160
Proposed method																
CHORD	**0.7663**	**0.8005**	**0.7914**	**0.7836**	**0.8021**	**0.7263**	**0.8474**	**0.7882**	**0.7760**	**0.7988**	**0.8046**	**0.7933**	**0.8193**	**0.7237**	**0.8557**	**0.7959**

similar pattern can be observed for larger Qwen and Ministral models in many settings. This may be because stronger LLMs develop richer and more structured internal representations, which provide more informative relational signals for hallucination detection. At the same time, local reversals still occur across datasets and model families. This is reasonable because model version, architecture, instruction tuning, and generation behavior can also affect hidden-state relations. Thus, the results support a cautious conclusion. Model scale is often helpful, but cross-model hallucination detection remains influenced by family-specific and version-specific factors.

Expanded analysis of ablation results. Table 4 studies the two core components of CHORD, namely relation-aware attention and meta-learning. The main ablation table reports a compact view of the results. QA Avg. denotes the macro average over TruthfulQA, HaluEval, TriviaQA, and CoQA. GSM8K, MBPP, and xLAM are kept separately to represent reasoning, code, and agent-oriented settings, and All Avg. denotes the macro average over all seven benchmarks.

This design avoids repeating all baseline rows in the main text while preserving the main task-level differences.

The variant CHORD w/o RelAttn removes relation-aware attention while keeping the remaining framework. In this setting, edge-level relational information is no longer explicitly propagated into node representations. The performance drop is usually larger than that of the meta-learning ablation. This indicates that the improvement of CHORD does not come merely from hidden-state restructuring, padding, or graph node construction. Instead, explicitly propagating pairwise relational information is central to obtaining transferable graph-level features. In other words, the layer-token-feature graph is useful not only because it reorganizes hidden states, but also because its edges are used by the relation-aware encoder.

The variant CHORD w/o Meta keeps the relation-aware architecture but replaces the meta-learning objective with standard empirical risk minimization over source-family examples. This variant usually performs better than CHORD w/o RelAttn, which suggests that the relational architecture itself already provides useful structure. However, it still falls below full CHORD and shows less stable performance across target families. This supports the role of meta-learning in the cross-model setting. By repeatedly simulating held-out model transfer during training, the meta-learning objective discourages the detector from relying only on source-family-specific shortcuts and encourages it to use relational features that are more transferable.

Both ablated variants remain above ordinary training-based baselines in most aggregated results. This suggests that the unified relational formulation already addresses part of the cross-model mismatch. However, neither variant consistently dominates the strongest training-free baseline. This pattern is important because it shows that the two modules are complementary. Relation-aware attention provides the structural mechanism for absorbing edge-level relations into node representations, while meta-learning improves the stability of the learned detector under family-held-out transfer. Only their combination yields the consistent advantage of full CHORD.

The relational design pays off most clearly on GSM8K and xLAM: both ablated variants still exceed the strongest training-free baseline there (0.7344 and 0.7572 versus 0.7297 on GSM8K; 0.7584 and 0.7740 versus 0.7421 on xLAM), and full CHORD attains its largest margins on these two benchmarks. This suggests that reasoning-oriented and agent-oriented hallucinations benefit more from modeling relational internal structures. For GSM8K, errors often emerge through multi-step reasoning, where token and layer relations may encode the progression of incorrect reasoning. For xLAM, agent-oriented errors may involve tool choice, action construction, and trajectory consistency, which can also produce structured internal deviations. By contrast, MBPP is more fragile. The full model remains strongest, but ablated variants often move closer to strong training-free baselines. This is consistent with the limited training size and execution-dependent nature of code correctness discussed above. Overall, the ablation study confirms that relation-aware attention and meta-learning are both important, and that the stable performance of CHORD comes from their combination rather than either component alone.

H Experimental Extension

H.1 Scale-Group Transfer

The main experiments evaluate cross-model generalization under a leave-one-family-out protocol. Here we provide a complementary analysis that asks whether the detector also transfers across model scales. We partition the target models into three coarse scale groups: the small-scale group contains Llama3.2-3B, Llama3.1-8B, Qwen3-8B, and Ministral3-8B; the medium-scale group contains Qwen3-14B and Ministral3-14B; and the large-scale group contains Llama3.3-70B and Qwen3.5-35B. The groups are intended to probe transfer across scale ranges rather than to form a perfectly controlled size-only comparison, since available open-source models are not family-balanced at every parameter scale.

For each ordered pair of distinct scale groups, training-based methods are trained on all models in the source scale group and evaluated on every model in the target scale group. For example, Small→Large trains on the small-scale group and tests on the large-scale target models, while Large→Small reverses this direction. Training-free methods do not use source models, so their values depend only on the target scale group; repeated values across different source groups with the same target group reflect this source-independent nature. Table 5 reports the full per-benchmark results for all directed scale-group transfers.

Table 5: Cross-size transfer results. Models are grouped by parameter scale; training-based methods are trained on source-size-group models and evaluated on target-size-group models. Values are AUROC averaged over target models in the target group. Best results are **bold** and second-best results are underlined.

Method	Scale-Group Transfer							
	Small→Medium							
	Mean AUROC↑							
	TruthfulQA	HaluEval	TriviaQA	CoQA	GSM8K	MBPP	xLAM	**Average**
Training-free methods								
Perplexity	0.6371	0.6580	0.6457	0.6438	0.6444	0.6097	0.6743	0.6447
Semantic Entropy	0.6846	0.6946	0.6986	0.6944	0.7148	0.6486	0.7239	0.6942
EigenScore	0.6800	0.6953	0.6905	0.6747	0.6969	0.6532	0.7090	0.6857
Lexical Similarity	0.6565	0.6578	0.6862	0.6791	0.6570	0.6311	0.6758	0.6634
Verbalize	0.5246	0.5301	0.5216	0.5168	0.5280	0.5070	0.5542	0.5260
InterrogateLLM	0.6999	0.7211	0.7080	0.6938	0.7329	0.6930	0.7433	0.7132
Training-based methods								
MM	0.5462	0.5881	0.5885	0.5490	0.6051	0.5384	0.6079	0.5747
SEP	0.5749	0.5799	0.5714	0.5829	0.6167	0.5559	0.6155	0.5853
SAPLMA	0.6014	0.5986	0.5956	0.6181	0.6305	0.5530	0.6363	0.6048
LapEigvals	0.5399	0.5568	0.5706	0.5592	0.5620	0.5182	0.5916	0.5569
Proposed method								
CHORD	**0.7429**	**0.7797**	**0.7791**	**0.7688**	**0.8146**	**0.7063**	**0.8407**	**0.7760**

Method	Medium→Small							
	Mean AUROC↑							
	TruthfulQA	HaluEval	TriviaQA	CoQA	GSM8K	MBPP	xLAM	**Average**
Training-free methods								
Perplexity	0.6416	0.6411	0.6588	0.6407	0.6560	0.6175	0.6747	0.6472
Semantic Entropy	0.6880	0.6952	0.7087	0.6928	0.7140	0.6533	0.7307	0.6975
EigenScore	0.6815	0.6845	0.6839	0.6782	0.6872	0.6581	0.7041	0.6825
Lexical Similarity	0.6593	0.6610	0.6992	0.6822	0.6467	0.6282	0.6715	0.6640
Verbalize	0.5162	0.5227	0.5242	0.5171	0.5312	0.5009	0.5514	0.5234
InterrogateLLM	0.6986	0.7161	0.7072	0.6985	0.7202	0.6894	0.7413	0.7102
Training-based methods								
MM	0.5695	0.5681	0.5762	0.5854	0.5897	0.5487	0.6034	0.5773
SEP	0.6001	0.6005	0.6048	0.5915	0.6159	0.5543	0.6105	0.5968
SAPLMA	0.5885	0.6161	0.6151	0.6062	0.6196	0.5627	0.6315	0.6057
LapEigvals	0.5422	0.5568	0.5541	0.5770	0.5891	0.5197	0.5933	0.5618
Proposed method								
CHORD	**0.7530**	**0.7740**	**0.7707**	**0.7557**	**0.7977**	**0.7009**	**0.8298**	**0.7688**

Method	Small→Large							
Training-free methods								
Perplexity	0.6518	0.6618	0.6734	0.6674	0.6689	0.6357	0.6832	0.6632
Semantic Entropy	0.7087	0.7238	0.7221	0.7162	0.7401	0.6717	0.7342	0.7167
EigenScore	0.7068	0.7052	0.7017	0.6975	0.7094	0.6675	0.7069	0.6993
Lexical Similarity	0.6798	0.6776	0.7045	0.7086	0.6719	0.6404	0.6794	0.6803
Verbalize	0.5379	0.5442	0.5492	0.5332	0.5526	0.5145	0.5567	0.5412
InterrogateLLM	0.7160	0.7226	0.7360	0.7246	0.7455	0.7042	0.7427	0.7274
Training-based methods								
MM	0.5353	0.5522	0.5816	0.5646	0.5857	0.5176	0.5795	0.5595
SEP	0.5872	0.5845	0.5987	0.5931	0.5990	0.5219	0.5995	0.5834
SAPLMA	0.5821	0.5906	0.5846	0.6047	0.5962	0.5306	0.6233	0.5874
LapEigvals	0.5316	0.5419	0.5554	0.5517	0.5789	0.5150	0.5886	0.5519
Proposed method								
CHORD	**0.7567**	**0.7580**	**0.7785**	**0.7580**	**0.7900**	**0.7150**	**0.8089**	**0.7664**

Method	Large→Small							
Training-free methods								
Perplexity	0.6416	0.6411	0.6588	0.6407	0.6560	0.6175	0.6747	0.6472
Semantic Entropy	0.6880	0.6952	0.7087	0.6928	0.7140	0.6533	0.7307	0.6975
EigenScore	0.6815	0.6845	0.6839	0.6782	0.6872	0.6581	0.7041	0.6825
Lexical Similarity	0.6593	0.6610	0.6992	0.6822	0.6467	0.6282	0.6715	0.6640
Verbalize	0.5162	0.5227	0.5242	0.5171	0.5312	0.5009	0.5514	0.5234
InterrogateLLM	0.6986	0.7161	0.7072	0.6985	0.7202	0.6894	0.7413	0.7102
Training-based methods								
MM	0.5649	0.5814	0.6076	0.5937	0.6073	0.5566	0.6055	0.5881
SEP	0.6144	0.6029	0.5877	0.5972	0.6209	0.5611	0.6273	0.6017
SAPLMA	0.6020	0.6346	0.6244	0.6277	0.6313	0.5630	0.6429	0.6180
LapEigvals	0.5539	0.5742	0.5608	0.5793	0.6030	0.5269	0.6246	0.5747
Proposed method								
CHORD	**0.7660**	**0.7882**	**0.7925**	**0.7746**	**0.8144**	**0.7126**	**0.8495**	**0.7854**

Method	Medium→Large							
Training-free methods								
Perplexity	0.6518	0.6618	0.6734	0.6674	0.6689	0.6357	0.6832	0.6632
Semantic Entropy	0.7087	0.7238	0.7221	0.7162	0.7401	0.6717	0.7342	0.7167
EigenScore	0.7068	0.7052	0.7017	0.6975	0.7094	0.6675	0.7069	0.6993
Lexical Similarity	0.6798	0.6776	0.7045	0.7086	0.6719	0.6404	0.6794	0.6803
Verbalize	0.5379	0.5442	0.5492	0.5332	0.5526	0.5145	0.5567	0.5412
InterrogateLLM	0.7160	0.7226	0.7360	0.7246	0.7455	0.7042	0.7427	0.7274
Training-based methods								
MM	0.5477	0.5621	0.5806	0.5781	0.5877	0.5176	0.5829	0.5653
SEP	0.5796	0.5934	0.5985	0.5986	0.5923	0.5442	0.6024	0.5870
SAPLMA	0.6130	0.5841	0.6004	0.6049	0.6164	0.5333	0.6129	0.5950
LapEigvals	0.5317	0.5450	0.5676	0.5564	0.5648	0.5155	0.5780	0.5513
Proposed method								
CHORD	**0.7699**	**0.7824**	**0.7923**	**0.7747**	**0.8195**	**0.7131**	**0.8261**	**0.7826**

Method	Large→Medium							
Training-free methods								
Perplexity	0.6371	0.6580	0.6457	0.6438	0.6444	0.6097	0.6743	0.6447
Semantic Entropy	0.6846	0.6946	0.6986	0.6944	0.7148	0.6486	0.7239	0.6942
EigenScore	0.6800	0.6953	0.6905	0.6747	0.6969	0.6532	0.7090	0.6857
Lexical Similarity	0.6565	0.6578	0.6862	0.6791	0.6570	0.6311	0.6758	0.6634
Verbalize	0.5246	0.5301	0.5216	0.5168	0.5280	0.5070	0.5542	0.5260
InterrogateLLM	0.6999	0.7211	0.7080	0.6938	0.7329	0.6930	0.7433	0.7132
Training-based methods								
MM	0.5677	0.5966	0.5957	0.5714	0.5946	0.5563	0.6083	0.5844
SEP	0.5948	0.5847	0.5896	0.5868	0.6122	0.5523	0.6278	0.5926
SAPLMA	0.6049	0.6301	0.6012	0.6220	0.6125	0.5510	0.6380	0.6085
LapEigvals	0.5357	0.5572	0.5901	0.5810	0.5699	0.5301	0.6096	0.5677
Proposed method								
CHORD	**0.7463**	**0.7888**	**0.7805**	**0.7772**	**0.8142**	**0.7050**	**0.8498**	**0.7803**

Overall, CHORD remains the strongest method across scale-group transfer directions. The result indicates that the relational detector is not tied to a narrow parameter scale: even when the source and target groups differ substantially in size, CHORD maintains a clear advantage over both training-free and ordinary training-based baselines. At the same time, the transfer directions are not identical. Transfer between the two extreme groups is asymmetric: Large→Small reaches 0.7854, compared with 0.7664 for Small→Large. This direction effect is not consistent across the remaining pairs, however: Small→Medium (0.7760) slightly exceeds Medium→Small (0.7688), and Medium→Large (0.7826) slightly exceeds Large→Medium (0.7803). Larger source models may therefore expose richer relational structure in some cases, but scale direction alone does not determine transfer difficulty.

The scale-group results also show that source composition matters in addition to scale. The small group contains more models and covers more model families, whereas the medium and large groups contain fewer models. Thus, absolute AUROC reflects both scale direction and the diversity of the source group. This is why the experiment should be interpreted as scale-group transfer rather than a causal isolation of parameter count. Despite this caveat, the main trend is stable: CHORD remains robust across all directed scale-group shifts, and its gains do not depend on training and testing within the same size regime.

The behavior of the baselines provides additional context. Training-free methods, especially Semantic Entropy and InterrogateLLM, remain competitive because they are naturally source-independent. However, they do not benefit from source-model training and therefore lag behind CHORD in most settings. Ordinary training-based baselines perform better here than in the leave-one-family-out main experiment, which is expected because each source scale group can include multiple model families and therefore provides a less severe distribution shift than a fully held-out family. Nevertheless, their performance remains clearly below CHORD, suggesting that simply training on a mixture of source-scale models is insufficient for stable cross-scale generalization.

The task-level results follow the same qualitative pattern as the main experiments. xLAM and GSM8K show strong gains for CHORD, indicating that relational internal features are especially useful for agent-oriented and reasoning-heavy hallucination detection. MBPP remains the most fragile benchmark, where the gap between CHORD and the strongest training-free baselines is smaller. This is consistent with the code setting being more dependent on execution semantics and fine-grained program correctness. Taken together, Table 5 shows that CHORD's cross-model robustness extends beyond family transfer to scale-group transfer, while also revealing that transfer direction and source-group diversity modulate the absolute difficulty of the setting.

H.2 Fine-Grained Edge Ablation

We next analyze which relation channels contribute to CHORD's performance. The main ablation in Table 2 removes larger components such as relation-aware attention or meta-learning. Here we perform a finer-grained edge ablation by removing one relation channel at a time while keeping the rest of the method unchanged. Specifically, we compare full CHORD with three variants: CHORD w/o Token, CHORD w/o Layer, and CHORD w/o Intra. The first removes the token-level edge channel, the second removes the layer-level edge channel, and the third removes the intra-block feature edge channel. All other components, including hidden representation restructuring, the remaining edge channels, relation-aware encoding, pooling, and meta-learning, are kept the same. This design isolates whether each relation channel provides complementary information rather than testing whether the entire relational architecture is necessary.

Table 6 reports the full per-dataset and per-model edge-ablation results. The table also includes training-free and training-based baselines, allowing the ablated variants to be compared both against the complete method and against the strongest non-CHORD alternatives.

The first observation is that removing any single edge channel consistently weakens CHORD. Full CHORD achieves the best performance across the table, while each edge-ablated variant loses performance on average. This supports the view that the three relation channels are complementary: none of them alone explains the entire gain, but each contributes useful information to the final detector. At the same time, the single-edge ablations remain much stronger than ordinary training-based baselines. This is expected because each ablated variant still retains most of CHORD, including the other relation channels and the meta-learning objective. Therefore, the edge ablation should be interpreted as measuring marginal channel contribution, not as a complete removal of relational modeling.

The second observation is that different edge channels matter for different benchmark types. Removing token-level edges tends to hurt settings where answer trajectory and token-level consistency are important, especially xLAM and dialogue-style or answer-development benchmarks such as CoQA and HaluEval. This pattern is natural because token-level relations capture how hidden states vary across generated answer positions. When this channel is removed, the detector has less access to the sequential structure of the response, which is particularly relevant for agent-oriented hallucinations and multi-step answer generation.

Removing layer-level edges has a different profile. Its effect is more visible on reasoning and semantic-consistency benchmarks such as GSM8K and TruthfulQA. Layer-level relations encode how representations evolve across Transformer depth, and this evolution can carry information about whether an answer is internally supported or whether the model is drifting toward an unsupported conclusion. The stronger degradation on reasoning-oriented settings suggests that layerwise representation dynamics provide useful signals beyond token-level variation alone.

Removing intra-block feature edges affects yet another part of the evaluation. The drop is more pronounced on MBPP and also appears in some agent and knowledge-intensive settings. Intra-block edges capture fine-grained relations among feature blocks inside the unified hidden representation. Such structure can be useful when hallucination signals are not expressed primarily as token-position changes or layerwise shifts, but instead appear in local activation patterns. This helps explain why the intra-block channel is especially relevant for code and other settings where correctness may depend on subtle internal distinctions.

The three ablations therefore do not form a single universal ordering. Instead, their relative importance changes with the benchmark and target model. This is important because it would be less informative if one edge type dominated all settings or if all ablations produced indistinguishable drops. The

Table 6: Full edge-ablation results under the cross-family evaluation protocol. Values are AUROC. Best results are **bold** and second-best results are underlined.

Method	Target Model: Llama3.2-3B — Mean AUROC↑								Target Model: Llama3.1-8B — Mean AUROC↑							
	TruthfulQA	HaluEval	TriviaQA	CoQA	GSM8K	MBPP	xLAM	**Average**	TruthfulQA	HaluEval	TriviaQA	CoQA	GSM8K	MBPP	xLAM	**Average**
Training-free methods																
Perplexity	0.6329	0.6464	0.6448	0.6296	0.6536	0.6188	0.6621	0.6412	0.6546	0.6452	0.6545	0.6395	0.6523	0.6024	0.6857	0.6477
Semantic Entropy	0.6862	0.7095	0.7006	0.6934	0.7112	0.6571	0.7223	0.6972	0.6991	0.7113	0.7065	0.6919	0.7258	0.6636	0.7438	0.7060
EigenScore	0.6636	0.6872	0.6767	0.6719	0.6772	0.6508	0.7045	0.6760	0.6985	0.6963	0.6803	0.6747	0.7068	0.6614	0.6957	0.6877
Lexical Similarity	0.6542	0.6747	0.6878	0.6786	0.6454	0.6225	0.6603	0.6605	0.6760	0.6681	0.6928	0.6628	0.6495	0.6236	0.6871	0.6657
Verbalize	0.5199	0.5306	0.5220	0.5290	0.5207	0.5069	0.5470	0.5252	0.5238	0.5398	0.5110	0.5050	0.5474	0.4884	0.5595	0.5250
InterrogateLLM	0.6845	0.7353	0.7012	0.6915	0.7177	0.6915	0.7253	0.7067	0.7185	0.7211	0.7201	0.6881	0.7293	0.6865	0.7469	0.7158
Training-based methods																
MM	0.4857	0.5170	0.5129	0.4874	0.5061	0.5035	0.5200	0.5047	0.5127	0.5029	0.5247	0.5000	0.5141	0.4808	0.5052	0.5058
SEP	0.5079	0.5193	0.5373	0.5189	0.5300	0.5032	0.5290	0.5208	0.5191	0.5346	0.5142	0.5089	0.5382	0.4876	0.5395	0.5203
SAPLMA	0.4941	0.5191	0.5239	0.5390	0.5158	0.4901	0.5397	0.5174	0.5326	0.5311	0.5250	0.5013	0.5382	0.4818	0.5386	0.5212
LapEigvals	0.4817	0.5021	0.4740	0.4965	0.5264	0.4765	0.5295	0.4981	0.4981	0.4916	0.4712	0.4789	0.5083	0.4725	0.5225	0.4919
Edge ablation variants																
CHORD w/o Token	0.7433	0.7540	0.7695	0.7344	0.7837	0.7057	0.7719	0.7518	0.7797	0.7652	0.7881	0.7293	0.7925	0.6807	0.8010	0.7624
CHORD w/o Layer	0.7140	0.7677	0.7437	0.7603	0.7389	0.7053	0.8048	0.7478	0.7507	0.7770	0.7851	0.7522	0.7546	0.6931	0.8272	0.7628
CHORD w/o Intra	0.7405	0.7518	0.7430	0.7573	0.7738	0.6957	0.7689	0.7473	0.7532	0.7850	0.7636	0.7457	0.7982	0.6779	0.7978	0.7602
Proposed method																
CHORD	**0.7766**	**0.7926**	**0.7991**	**0.7920**	**0.8193**	**0.7351**	**0.8437**	**0.7941**	**0.7989**	**0.8102**	**0.8156**	**0.7823**	**0.8362**	**0.7307**	**0.8650**	**0.8056**

Method	Target Model: Llama3.3-70B								Target Model: Qwen3-8B							
Training-free methods																
Perplexity	0.6575	0.6747	0.6717	0.6748	0.6765	0.6353	0.6931	0.6691	0.6345	0.6398	0.6781	0.6542	0.6653	0.6179	0.6726	0.6518
Semantic Entropy	0.7096	0.7366	0.7231	0.7272	0.7410	0.6697	0.7406	0.7211	0.6840	0.6800	0.7127	0.6994	0.7143	0.6502	0.7228	0.6948
EigenScore	0.7076	0.7087	0.6961	0.7095	0.7094	0.6703	0.7093	0.7016	0.6785	0.6817	0.6869	0.6873	0.6813	0.6683	0.7085	0.6846
Lexical Similarity	0.6844	0.6797	0.7040	0.7088	0.6854	0.6432	0.6858	0.6845	0.6538	0.6538	0.7089	0.6966	0.6441	0.6401	0.6688	0.6666
Verbalize	0.5472	0.5531	0.5472	0.5385	0.5592	0.5139	0.5584	0.5454	0.5193	0.5090	0.5200	0.5170	0.5339	0.4969	0.5463	0.5203
InterrogateLLM	0.7343	0.7454	0.7472	0.7219	0.7473	0.7181	0.7477	0.7374	0.6950	0.7077	0.7044	0.7173	0.7082	0.6834	0.7483	0.7092
Training-based methods																
MM	0.5095	0.5244	0.5288	0.5195	0.5259	0.4814	0.5153	0.5150	0.4923	0.4849	0.5044	0.5095	0.5046	0.4837	0.5171	0.4995
SEP	0.5117	0.5352	0.5400	0.5243	0.5263	0.4905	0.5297	0.5225	0.5147	0.5030	0.4993	0.5062	0.5159	0.4823	0.5228	0.5063
SAPLMA	0.5250	0.5177	0.5231	0.5364	0.5375	0.4840	0.5386	0.5232	0.4822	0.5223	0.5171	0.5093	0.5167	0.4922	0.5384	0.5112
LapEigvals	0.5083	0.4992	0.5167	0.5204	0.5132	0.4718	0.5280	0.5082	0.4831	0.4872	0.5141	0.4873	0.5063	0.4716	0.5292	0.4970
Edge ablation variants																
CHORD w/o Token	0.7958	0.7940	0.7967	0.7780	0.8352	0.7443	0.8039	0.7926	0.7322	0.7391	0.7893	0.7432	0.7830	0.6897	0.7707	0.7496
CHORD w/o Layer	0.7770	0.8042	0.7920	0.8008	0.7813	0.7168	0.8252	0.7853	0.7313	0.7518	0.7941	0.7709	0.7386	0.7104	0.8140	0.7587
CHORD w/o Intra	0.8136	0.8078	0.7874	0.7908	0.8180	0.7121	0.8095	0.7913	0.7599	0.7478	0.7649	0.7701	0.7626	0.6983	0.7720	0.7537
Proposed method																
CHORD	**0.8276**	**0.8374**	**0.8257**	**0.8302**	**0.8568**	**0.7605**	**0.8713**	**0.8299**	**0.7896**	**0.7941**	**0.8192**	**0.7991**	**0.8256**	**0.7481**	**0.8484**	**0.8034**

Method	Target Model: Qwen3-14B								Target Model: Qwen3.5-35B							
Training-free methods																
Perplexity	0.6408	0.6650	0.6442	0.6519	0.6363	0.6164	0.6851	0.6485	0.6460	0.6488	0.6751	0.6601	0.6614	0.6361	0.6732	0.6572
Semantic Entropy	0.6810	0.6968	0.6966	0.6958	0.7193	0.6438	0.7195	0.6933	0.7078	0.7110	0.7211	0.7052	0.7392	0.6738	0.7278	0.7123
EigenScore	0.6931	0.6891	0.6965	0.6748	0.7013	0.6556	0.7090	0.6885	0.7061	0.7018	0.7072	0.6854	0.7093	0.6648	0.7046	0.6970
Lexical Similarity	0.6498	0.6560	0.6779	0.6795	0.6529	0.6368	0.6834	0.6623	0.6752	0.6756	0.7050	0.7083	0.6583	0.6377	0.6730	0.6762
Verbalize	0.5349	0.5262	0.5274	0.5261	0.5329	0.5091	0.5578	0.5306	0.5286	0.5352	0.5511	0.5279	0.5460	0.5150	0.5550	0.5370
InterrogateLLM	0.7009	0.7127	0.7062	0.7014	0.7496	0.6904	0.7435	0.7150	0.6978	0.6997	0.7248	0.7273	0.7438	0.6902	0.7376	0.7173
Training-based methods																
MM	0.5033	0.5228	0.5270	0.4910	0.5120	0.4966	0.5269	0.5114	0.4844	0.4817	0.5243	0.5240	0.5280	0.4848	0.5116	0.5055
SEP	0.5277	0.5040	0.5072	0.5170	0.5258	0.4980	0.5317	0.5159	0.5395	0.5139	0.5284	0.5222	0.5301	0.4874	0.5350	0.5224
SAPLMA	0.5109	0.5192	0.5226	0.5385	0.5287	0.4916	0.5370	0.5212	0.5198	0.5174	0.5268	0.5303	0.5240	0.4868	0.5392	0.5206
LapEigvals	0.4840	0.5049	0.5113	0.4969	0.5062	0.4822	0.5263	0.5017	0.4712	0.4891	0.5108	0.4829	0.5289	0.4727	0.5291	0.4978
Edge ablation variants																
CHORD w/o Token	0.7342	0.7800	0.7768	0.7525	0.8129	0.6913	0.8012	0.7641	0.7558	0.7668	0.8129	0.7515	0.7993	0.6932	0.7758	0.7650
CHORD w/o Layer	0.7313	0.7941	0.7710	0.7785	0.7708	0.7004	0.8193	0.7665	0.7525	0.7814	0.8028	0.7826	0.7717	0.7032	0.8182	0.7732
CHORD w/o Intra	0.7450	0.7863	0.7596	0.7693	0.7891	0.6732	0.8045	0.7610	0.7833	0.7705	0.7913	0.7784	0.7949	0.6963	0.7728	0.7696
Proposed method																
CHORD	**0.7720**	**0.8237**	**0.8143**	**0.8070**	**0.8387**	**0.7248**	**0.8722**	**0.8075**	**0.8069**	**0.8104**	**0.8333**	**0.8027**	**0.8406**	**0.7207**	**0.8472**	**0.8088**

Method	Target Model: Ministral3-8B								Target Model: Ministral3-14B							
Training-free methods																
Perplexity	0.6443	0.6328	0.6576	0.6397	0.6528	0.6308	0.6785	0.6481	0.6335	0.6510	0.6471	0.6357	0.6526	0.6029	0.6635	0.6409
Semantic Entropy	0.6828	0.6800	0.7148	0.6867	0.7045	0.6424	0.7338	0.6921	0.6882	0.6925	0.7006	0.6930	0.7104	0.6534	0.7283	0.6952
EigenScore	0.6856	0.6729	0.6918	0.6788	0.6836	0.6519	0.7076	0.6817	0.6668	0.7015	0.6846	0.6746	0.6926	0.6509	0.7090	0.6829
Lexical Similarity	0.6530	0.6474	0.7075	0.6906	0.6479	0.6265	0.6698	0.6632	0.6632	0.6595	0.6945	0.6788	0.6610	0.6255	0.6682	0.6644
Verbalize	0.5019	0.5116	0.5440	0.5172	0.5230	0.5112	0.5528	0.5231	0.5143	0.5340	0.5158	0.5075	0.5231	0.5048	0.5506	0.5214
InterrogateLLM	0.6963	0.7002	0.7030	0.6969	0.7254	0.6961	0.7446	0.7089	0.6989	0.7295	0.7097	0.6862	0.7163	0.6957	0.7432	0.7114
Training-based methods																
MM	0.4807	0.4840	0.5294	0.5123	0.5298	0.4853	0.5285	0.5071	0.4866	0.5095	0.5123	0.4854	0.5284	0.4924	0.5232	0.5054
SEP	0.5232	0.5188	0.5162	0.4990	0.5033	0.5030	0.5288	0.5132	0.4860	0.5076	0.5123	0.4999	0.5186	0.4895	0.5391	0.5076
SAPLMA	0.4816	0.5275	0.5396	0.5073	0.5335	0.4944	0.5392	0.5176	0.4865	0.5232	0.4952	0.5091	0.5266	0.4839	0.5386	0.5090
LapEigvals	0.4708	0.4737	0.5019	0.5031	0.4983	0.4733	0.5295	0.4929	0.4847	0.4766	0.5179	0.4800	0.5058	0.4702	0.5161	0.4930
Edge ablation variants																
CHORD w/o Token	0.7183	0.7466	0.7543	0.7247	0.7622	0.6957	0.7697	0.7388	0.7232	0.7528	0.7739	0.7326	0.7769	0.7104	0.7762	0.7494
CHORD w/o Layer	0.7137	0.7813	0.7526	0.7628	0.7548	0.7097	0.8014	0.7538	0.7194	0.7704	0.7708	0.7638	0.7457	0.7008	0.8143	0.7550
CHORD w/o Intra	0.7340	0.7570	0.7408	0.7396	0.7538	0.7121	0.7800	0.7453	0.7400	0.7630	0.7511	0.7478	0.7881	0.7159	0.7924	0.7569
Proposed method																
CHORD	**0.7663**	**0.8005**	**0.7914**	**0.7836**	**0.8021**	**0.7263**	**0.8474**	**0.7882**	**0.7760**	**0.7988**	**0.8046**	**0.7933**	**0.8193**	**0.7237**	**0.8557**	**0.7959**

observed pattern suggests that token, layer, and intra-block edges capture different failure modes of generated answers. CHORD benefits from combining them because hallucinations are heterogeneous: some errors are reflected in token-level answer development, some in layerwise semantic evolution, and others in fine-grained feature-block structure.

Finally, the edge ablation results complement the larger module-level ablations. Removing one edge channel causes a smaller drop than removing relation-aware attention entirely, which is expected because the remaining channels can still carry relational information. However, the consistent degradation relative to full CHORD shows that these channels are not redundant. Together, Table 6 indicates that CHORD's advantage comes from aggregating multiple complementary relation channels rather than relying on a single handcrafted relational view.

Table 7: Cross-dataset hallucination detection performance under leave-one-benchmark-out evaluation. Training-based methods are trained on six source benchmarks and evaluated on the held-out target benchmark for each target model, while training-free methods are directly applied to the same targets. The table is split into four blocks only due to space constraints. We report AUROC on each held-out benchmark and the average across benchmarks for each target model. Best results are **bold**, second-best are underlined.

Method	Target Model															
	Llama3.2-3B								Llama3.1-8B							
	Mean AUROC↑								Mean AUROC↑							
	TruthfulQA	HaluEval	TriviaQA	CoQA	GSM8K	MBPP	xLAM	**Average**	TruthfulQA	HaluEval	TriviaQA	CoQA	GSM8K	MBPP	xLAM	**Average**
Training-free methods																
Perplexity	0.6329	0.6464	0.6448	0.6296	0.6536	0.6188	0.6621	0.6412	0.6546	0.6452	0.6545	0.6395	0.6523	0.6024	0.6857	0.6477
Semantic Entropy	0.6862	0.7095	0.7006	0.6934	0.7112	0.6571	0.7223	0.6972	0.6991	0.7113	0.7065	0.6919	0.7258	0.6636	0.7438	0.7060
EigenScore	0.6636	0.6872	0.6767	0.6719	0.6772	0.6508	0.7045	0.6760	0.6985	0.6963	0.6803	0.6747	0.7068	0.6614	0.6957	0.6877
Lexical Similarity	0.6542	0.6747	0.6878	0.6786	0.6454	0.6225	0.6603	0.6605	0.6760	0.6681	0.6928	0.6628	0.6495	0.6236	0.6871	0.6657
Verbalize	0.5199	0.5306	0.5220	0.5290	0.5207	0.5069	0.5470	0.5252	0.5238	0.5398	0.5110	0.5050	0.5474	0.4884	0.5595	0.5250
InterrogateLLM	0.6845	0.7353	0.7012	0.6915	0.7177	0.6915	0.7253	0.7067	0.7185	0.7211	0.7201	0.6881	0.7293	0.6865	0.7469	0.7158
Training-based methods																
MM	0.6128	0.6659	0.6527	0.6291	0.6503	0.6089	0.6529	0.6389	0.6636	0.6423	0.6537	0.6280	0.6339	0.6024	0.6656	0.6414
SEP	0.6338	0.6718	0.6496	0.6345	0.6556	0.6181	0.6709	0.6478	0.6705	0.6723	0.6904	0.6495	0.6704	0.6289	0.6936	0.6679
SAPLMA	0.6542	0.6944	0.6592	0.6812	0.6720	0.6325	0.6793	0.6675	0.6822	0.6838	0.6929	0.6746	0.6998	0.6480	0.7057	0.6839
LapEigvals	0.5986	0.6534	0.6150	0.6036	0.6389	0.5993	0.6474	0.6223	0.6646	0.6276	0.6517	0.6292	0.6408	0.5847	0.6562	0.6364
Proposed method																
CHORD	**0.7586**	**0.7821**	**0.7749**	**0.7723**	**0.8027**	**0.7026**	**0.8060**	**0.7713**	**0.7767**	**0.7991**	**0.8013**	**0.7632**	**0.8072**	**0.7018**	**0.8450**	**0.7849**
	Llama3.3-70B								Qwen3-8B							
Training-free methods																
Perplexity	0.6575	0.6747	0.6717	0.6748	0.6765	0.6353	0.6931	0.6691	0.6345	0.6398	0.6781	0.6542	0.6653	0.6179	0.6726	0.6518
Semantic Entropy	0.7096	0.7366	0.7231	0.7272	0.7410	0.6697	0.7406	0.7211	0.6840	0.6800	0.7127	0.6994	0.7143	0.6502	0.7228	0.6948
EigenScore	0.7076	0.7087	0.6961	0.7095	0.7094	0.6703	0.7093	0.7016	0.6785	0.6817	0.6869	0.6873	0.6813	0.6683	0.7085	0.6846
Lexical Similarity	0.6844	0.6797	0.7040	0.7088	0.6854	0.6432	0.6858	0.6845	0.6538	0.6538	0.7089	0.6966	0.6441	0.6401	0.6688	0.6666
Verbalize	0.5472	0.5531	0.5472	0.5385	0.5592	0.5139	0.5584	0.5454	0.5193	0.5090	0.5200	0.5170	0.5339	0.4969	0.5463	0.5203
InterrogateLLM	0.7343	0.7454	0.7472	0.7219	0.7473	0.7181	0.7477	0.7374	0.6950	0.7077	0.7044	0.7173	0.7082	0.6834	0.7483	0.7092
Training-based methods																
MM	0.6706	0.6847	0.7105	0.6692	0.7041	0.6350	0.6871	0.6802	0.6192	0.6436	0.6503	0.6425	0.6329	0.5942	0.6756	0.6369
SEP	0.6964	0.7065	0.7329	0.6931	0.7026	0.6523	0.6759	0.6942	0.6439	0.6477	0.6685	0.6460	0.6303	0.6077	0.6933	0.6482
SAPLMA	0.7005	0.7212	0.7161	0.7093	0.7049	0.6695	0.7169	0.7055	0.6642	0.6827	0.6705	0.7003	0.6639	0.6323	0.7035	0.6739
LapEigvals	0.6537	0.6799	0.6808	0.6741	0.6889	0.6470	0.6984	0.6747	0.6017	0.6283	0.6406	0.6409	0.6182	0.5876	0.6514	0.6241
Proposed method																
CHORD	**0.8044**	**0.8281**	**0.8120**	**0.8124**	**0.8304**	**0.7362**	**0.8594**	**0.8118**	**0.7697**	**0.7607**	**0.8005**	**0.7667**	**0.7990**	**0.7089**	**0.8173**	**0.7747**
	Qwen3-14B								Qwen3.5-35B							
Training-free methods																
Perplexity	0.6408	0.6650	0.6442	0.6519	0.6363	0.6164	0.6851	0.6485	0.6460	0.6488	0.6751	0.6601	0.6614	0.6361	0.6732	0.6572
Semantic Entropy	0.6810	0.6968	0.6966	0.6958	0.7193	0.6438	0.7195	0.6933	0.7078	0.7110	0.7211	0.7052	0.7392	0.6738	0.7278	0.7123
EigenScore	0.6931	0.6891	0.6965	0.6748	0.7013	0.6556	0.7090	0.6885	0.7061	0.7018	0.7072	0.6854	0.7093	0.6648	0.7046	0.6970
Lexical Similarity	0.6498	0.6560	0.6779	0.6795	0.6529	0.6368	0.6834	0.6623	0.6752	0.6756	0.7050	0.7083	0.6583	0.6377	0.6730	0.6762
Verbalize	0.5349	0.5262	0.5274	0.5261	0.5329	0.5091	0.5578	0.5306	0.5286	0.5352	0.5511	0.5279	0.5460	0.5150	0.5550	0.5370
InterrogateLLM	0.7009	0.7127	0.7062	0.7014	0.7496	0.6904	0.7435	0.7150	0.6978	0.6997	0.7248	0.7273	0.7438	0.6902	0.7376	0.7173
Training-based methods																
MM	0.6327	0.6421	0.6423	0.6398	0.6688	0.6105	0.6749	0.6444	0.6213	0.6319	0.6573	0.7084	0.6799	0.6067	0.6972	0.6575
SEP	0.6528	0.6684	0.6703	0.6469	0.6739	0.6098	0.6725	0.6564	0.6515	0.6722	0.6779	0.6904	0.6927	0.6212	0.6719	0.6683
SAPLMA	0.6703	0.6888	0.6935	0.6585	0.7080	0.6343	0.7157	0.6813	0.6489	0.6593	0.7000	0.6993	0.7055	0.6561	0.7164	0.6836
LapEigvals	0.6249	0.6361	0.6366	0.6271	0.6798	0.5883	0.6601	0.6361	0.6297	0.6481	0.6629	0.6433	0.6646	0.6084	0.6712	0.6469
Proposed method																
CHORD	**0.7601**	**0.8037**	**0.7914**	**0.7923**	**0.8005**	**0.7046**	**0.8337**	**0.7838**	**0.7712**	**0.7842**	**0.8091**	**0.7789**	**0.8186**	**0.7099**	**0.8213**	**0.7847**
	Ministral3-8B								Ministral3-14B							
Training-free methods																
Perplexity	0.6443	0.6328	0.6576	0.6397	0.6528	0.6308	0.6785	0.6481	0.6335	0.6510	0.6471	0.6357	0.6526	0.6029	0.6635	0.6409
Semantic Entropy	0.6828	0.6800	0.7148	0.6867	0.7045	0.6424	0.7338	0.6921	0.6882	0.6925	0.7006	0.6930	0.7104	0.6534	0.7283	0.6952
EigenScore	0.6856	0.6729	0.6918	0.6788	0.6836	0.6519	0.7076	0.6817	0.6668	0.7015	0.6846	0.6746	0.6926	0.6509	0.7090	0.6829
Lexical Similarity	0.6530	0.6474	0.7075	0.6906	0.6479	0.6265	0.6698	0.6632	0.6632	0.6595	0.6945	0.6788	0.6610	0.6255	0.6682	0.6644
Verbalize	0.5019	0.5116	0.5440	0.5172	0.5230	0.5112	0.5528	0.5231	0.5143	0.5340	0.5158	0.5075	0.5231	0.5048	0.5506	0.5214
InterrogateLLM	0.6963	0.7002	0.7030	0.6969	0.7254	0.6961	0.7446	0.7089	0.6989	0.7295	0.7097	0.6862	0.7163	0.6957	0.7432	0.7114
Training-based methods																
MM	0.6109	0.6494	0.6493	0.6284	0.6447	0.6029	0.6698	0.6365	0.6154	0.6598	0.6545	0.6308	0.6458	0.6033	0.6816	0.6416
SEP	0.6152	0.6499	0.6529	0.6253	0.6576	0.6295	0.6643	0.6421	0.6469	0.6750	0.6469	0.6401	0.6535	0.6178	0.6956	0.6537
SAPLMA	0.6508	0.6713	0.6737	0.6555	0.6753	0.6477	0.6850	0.6656	0.6551	0.7133	0.6705	0.6676	0.6669	0.6428	0.7154	0.6759
LapEigvals	0.6150	0.6231	0.6371	0.6093	0.6298	0.6112	0.6737	0.6285	0.5982	0.6484	0.6386	0.6004	0.6225	0.6083	0.6500	0.6238
Proposed method																
CHORD	**0.7365**	**0.7712**	**0.7661**	**0.7492**	**0.7624**	**0.7098**	**0.8136**	**0.7584**	**0.7539**	**0.7764**	**0.7821**	**0.7663**	**0.7791**	**0.7107**	**0.8228**	**0.7702**

H.3 Cross-Benchmark Transfer

The main experiments focus on cross-model generalization under a leave-one-family-out protocol. In that setting, a detector is trained on source LLMs and evaluated on unseen target LLMs, with the same benchmark used for both source and target evaluation. A natural question is whether the observed gains depend on this same-benchmark training setup. To address this, we further evaluate cross-benchmark transfer, where the training and test benchmarks are different while the target LLM is fixed.

Protocol. For each LLM $\mathbb{P}_{\boldsymbol{\theta}_i}$ and each held-out benchmark $d \in \{\text{TruthfulQA}, \text{HaluEval}, \text{TriviaQA}, \text{CoQA}, \text{GSM8K}, \text{MBPP}, \text{xLAM}\}$, we train each training-based detector on the other six benchmarks generated by the same LLM and evaluate it on the held-out benchmark d. Concretely, if $\mathcal{D}_{i,d}$ denotes the dataset constructed from model $\mathbb{P}_{\boldsymbol{\theta}_i}$ on benchmark d, then the cross-benchmark training set for held-out benchmark d is

$$\mathcal{D}_{i,\neg d} = \bigcup_{d' \neq d} \mathcal{D}_{i,d'}.$$

The detector is trained on $\mathcal{D}_{i,\neg d}$ and evaluated on $\mathcal{D}_{i,d}$. Thus, each column in Table 7 should be read as a *held-out benchmark*: the corresponding benchmark is used only for evaluation, while the remaining six benchmarks are used for training.

This protocol isolates benchmark shift from model shift. It is therefore easier than the leave-one-family-out cross-model setting in the main experiment, because the source and test features come from the same LLM and share the same hidden-state coordinate system. At the same time, it is still a meaningful transfer test: the detector must generalize from one set of task distributions to a different held-out benchmark, rather than being trained and tested on the same benchmark.

Treatment of training-free methods. Training-free methods do not use the source benchmarks. For a fixed model and held-out benchmark, their scores are therefore identical to the corresponding model–benchmark evaluation in the main table. We include them in Table 7 as source-independent reference baselines. Their repeated consistency with the main results is expected and should not be interpreted as an additional training condition. This also provides a useful anchor for comparing training-based detectors under benchmark shift.

Results. Table 7 reports the full cross-benchmark results. CHORD achieves the strongest performance across target models and held-out benchmarks, showing that its advantage is not restricted to training and testing on the same benchmark. The result is particularly important because cross-benchmark transfer tests a different form of robustness from the main cross-model experiment. The main experiment asks whether a detector trained on some LLMs can generalize to unseen LLMs; this experiment asks whether a detector trained on some benchmarks can generalize to unseen benchmarks for the same LLM. CHORD remains effective in both settings.

The behavior of ordinary training-based baselines is also informative. Unlike in the leave-one-family-out cross-model setting, methods such as MM, SEP, SAPLMA, and LapEigvals do not collapse to chance here. Their AUROC is often substantially above 0.5 and can approach strong training-free baselines in favorable cases. This is expected: when training and evaluation are performed on the same LLM, the representation space is shared, so a supervised detector can reuse model-specific coordinates and decision directions across benchmarks. The result therefore suggests that these baselines are not intrinsically ineffective or improperly tuned; rather, their weakness in the main experiment is largely caused by cross-model representation mismatch.

Training-free baselines remain competitive, especially Semantic Entropy and InterrogateLLM. Their strength is natural because they do not rely on a learned source-benchmark representation and therefore avoid overfitting to a particular training benchmark. However, they do not exploit cross-benchmark supervision, and CHORD consistently improves over them. This indicates that useful hallucination signals can be learned across benchmarks when the detector is built on transferable relational internal features.

Benchmark-level trends. The cross-benchmark results follow several task-level patterns. QA-style benchmarks such as TruthfulQA, HaluEval, TriviaQA, and CoQA transfer relatively smoothly because they share similar input-output formats and factuality-oriented labels. GSM8K also shows strong performance, suggesting that reasoning errors induce internal relational patterns that can be learned from other benchmarks and reused on mathematical reasoning. MBPP remains the most fragile benchmark: the gap between CHORD and the strongest baselines is smaller, reflecting the fact that code correctness is more execution-dependent and differs more sharply from natural-language QA. xLAM remains comparatively favorable for CHORD, consistent with the main results that agent-oriented hallucinations benefit from relational internal signals.

Implication. This experiment clarifies the relationship between benchmark shift and model shift. Cross-benchmark transfer within a fixed LLM is less severe than cross-model transfer, and ordinary training-based detectors can retain meaningful performance in this setting. However, CHORD remains consistently stronger, indicating that its relational formulation is useful beyond the specific benchmark used for training. Together with the leave-one-family-out results, Table 7 shows that CHORD addresses both sources of generalization difficulty: it transfers across benchmarks when the LLM is fixed, and it transfers across LLMs when the benchmark is fixed.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims made in the abstract and Section 1 are explicitly supported by our methodology in Section 4 and experimental results in Section 5.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Please see Appendix C.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: This paper focuses exclusively on empirical evaluations and architectural design, and thus does not include theoretical results or mathematical proofs.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: All necessary code, datasets, and detailed instructions required to reproduce the main experimental results are publicly available in our repository: https://github.com/MiracleworkerD/CHORD.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: All necessary code, datasets, and detailed instructions required to reproduce the main experimental results are publicly available in our repository: https://github.com/MiracleworkerD/CHORD.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: All experimental settings, including data splits, hyperparameters, and optimization details, are extensively detailed in Appendix A.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We report suitable error bars and provide appropriate information regarding statistical significance in Figure 2, with comprehensive details discussed in Appendix A.8.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Detailed information regarding the required compute resources, including hardware specifications, is provided in the instructions of our code repository.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics and confirm that the research conducted in this paper conforms to it in every respect.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This paper focuses on foundational machine learning methodology and does not present a direct path to specific applications or negative societal harms.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The methods and data presented in this paper do not pose a high risk for malicious misuse, thus no specific safeguards for release are required.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All creators of the existing datasets and codebases used in this work are properly cited, and their respective licenses are detailed in Appendix E.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.

- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: This paper focuses on a new algorithmic method and evaluating existing models/datasets; it does not release new foundational models or large-scale datasets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper is strictly computational and does not involve any crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: Not applicable, as this research does not involve human subjects.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: We describe the usage of LLMs explicitly in Section 4, as utilizing them for generating responses and evaluation is central to our cross-model transfer methodology.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.