HD-NDEs: Neural Differential Equations for Hallucination Detection in LLMs

Anonymous ACL submission

Abstract

In recent years, large language models (LLMs) have made remarkable advancements, yet hallucination, where models produce inaccurate or non-factual statements, remains a significant challenge for real-world deployment. Although current classification-based methods, such as SAPLMA, are highly efficient in mitigating hallucinations, they struggle when non-factual information arises in the early or mid-sequence of outputs, reducing their reliability. To address these issues, we propose **H**allucination **D**etection-**N**eural **D**ifferential **E**quations (**HD-NDEs**), a novel method that systematically assesses the truthfulness of statements by capturing the full dynamics of LLMs within their latent space. Our approaches apply neural differential equations (Neural DEs) to model the dynamic system in the latent space of LLMs. Then, the sequence in the latent space is mapped to the classification space for truth assessment. The extensive experiments across five datasets and six widely used LLMs demonstrate the effectiveness of HD-NDEs, especially, achieving over 14% improvement in AUC-ROC on the True-False dataset compared to state-of-the-art techniques.

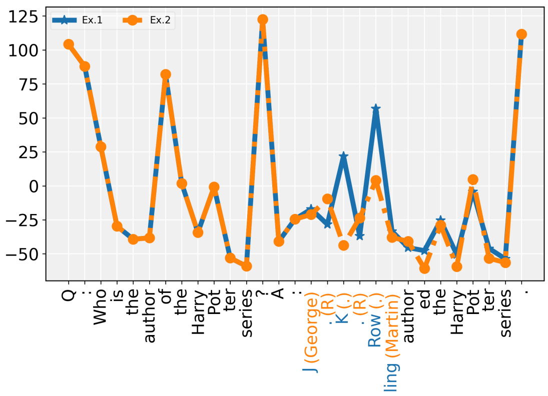


Figure 1: 1D PCA projection of hidden layer embeddings for each token in Ex.1 and Ex.2. Both examples use the same question with different answers. In the hidden state space, the embeddings of the earlier tokens are identical until the tokens begin to differ. The final few tokens, being the same, result in minimal differences in the hidden state activations.

1 Introduction

Hallucination has been widely recognized as a significant challenge in large language models (LLMs), as highlighted in various studies applications (Li et al., 2023a; Min et al., 2023; Geng et al., 2023). Efforts to mitigate this issue have led to the development of hallucination detection techniques, which are broadly categorized into evidence-based and evidence-free approaches. Evidence-based methods (Wang et al., 2023; Wei et al., 2024) generally involve retrieving relevant information from external sources to verify whether inconsistencies exist between the generated content and the retrieved evidence. Nevertheless, this retrieval and verification process is computationally intensive and time-consuming, making it impractical for high-throughput applications in routine use. In contrast, evidence-free methods (Chen et al., 2024; Duan et al., 2023; Geng et al., 2023) primarily utilize the inherent characteristics of LLMs and semantic features to identify potential hallucinations. These methods can be further categorized into logit-based, consistency-based, classification-based approaches, and so on. For instance, logit-based methods (Huang et al., 2023) estimate the overall uncertainty of a sentence by analyzing logit-based uncertainty at the token level. Alternatively, consistency-based methods (Manakul et al., 2023) assess the consistency of model outputs, based on the premise that hallucination tends to increase variability in the generated responses.

Furthermore, classification-based methods have proposed using a model's internal states to probe its confidence in factual vs. non-factual sentences. Specifically, a simple feed-forward neural network classifier can be trained on the activations of the final token's last-layer hidden state to predict the reliability of the model's output. This type of method has demonstrated significant effectiveness across various model architectures, as validated by multiple studies (Azaria and Mitchell, 2023; Li et al., 2024; Kossen et al., 2024; Su et al., 2024). However, the classification-based method is still in its early stages and remains inadequate for handling cases where the final token of a statement fails to capture the reliability of the entire sequence. It often struggles when non-factual tokens are located at the beginning or middle of the sequence, as mentioned in Levinstein and Herrmann (2024). We employ principal component analysis (PCA, Abdi and Williams 2010) to further investigate such failure cases. As shown in Figure 1, we reduce the dimensionality of each token's activations to a single dimension for clearer interpretation. Ex.1 illustrates a question with the correct answer, while Ex.2 presents the same question with an incorrect answer. Notably, the reduced hidden information of the last tokens in both examples appears nearly identical, despite differences in the middle of the sequences. This suggests that we need to effectively leverage hidden state information across the entire sequence, rather than only the last token, to accurately assess the truthfulness.

The advances of neural differential equations (Neural DEs) offer a promising solution by modeling hidden state transformations as continuous trajectories, providing a more accurate representation of information flow through LLMs (Kidger, 2022). Based on effective in tasks such as time-series forecasting, classification, and outlier detection (Choi et al., 2022; Jhin et al., 2024), Neural DEs are well-suited for addressing hallucination detections in LLMs, where subtle errors can result in factual inaccuracies in generated sequences. Motivated by these strengths, our work introduces a novel, supervised method, called HD-NDEs, marking the first application of Neural DEs in hallucination detection. As shown in Figure 2, the method explicitly models the trajectory of intermediate states in the latent space using Neural DEs. Unlike previous methods that focus on individual token representations, our approach leverages temporal information in state dynamics. We conduct an extensive study on five challenging hallucination datasets, evaluating our method and state-of-the-art approaches using six widely adopted LLMs. The results demonstrate the effectiveness of our approach. Our contributions are summarized as follows:

- We introduce HD-NDEs, the first method to apply Neural DEs, including neural ordinary differential equations (Neural ODEs, Chen et al. 2018), neural controlled differential equations (Neural CDEs, Kidger et al. 2020a), and neural stochastic differential equations (Neural SDEs, Oh et al. 2024), for detecting hallucinations in LLMs. By modeling the token generation process as continuous trajectories in latent space, HD-NDEs provides a more accurate and dynamic approach to detecting hallucinations.

- We evaluate HD-NDEs on five diverse and complex hallucination datasets and compare their performance with baseline methods across six widely used LLMs. Our results demonstrate that HD-NDEs outperforms existing approaches with a 14% improvement in True-False Dataset.

2 Related Work

Hallucination Detection. Hallucinations in LLMs pose significant challenges for their deployment (Zhang et al., 2023b; Li et al., 2023a). The generation of inaccurate information can result in customer attrition or legal risks, rendering the decision-making process unreliable. Detecting hallucinations has garnered increasing attention, and this detection is typically performed in one of the following ways: conducting a conventional retrieval task (Min et al., 2023; Wang et al., 2023), which requires external knowledge; converting the logits output into an uncertainty estimate for the sentence; or evaluating self-consistency (Mündler et al., 2023), where inconsistent outputs often indicate hallucinations. Recent studies have shown that hallucinations can be attributed to the model's internal representations and have proposed white-box methods to detect hallucinations based on token latent states (Burns et al., 2023; Azadi et al., 2023; Zhu et al., 2024). These approaches have outperformed black-box methods across various tasks. However, as noted in Levinstein and Herrmann (2024), they often struggle when non-factual tokens appear at the beginning or middle of the sequence.

Neural Differential Equations. Neural DEs have been extensively used in modeling dynamical systems or simulating neural networks (Chang et al., 2018; Dutta et al., 2021). For instance, Lu

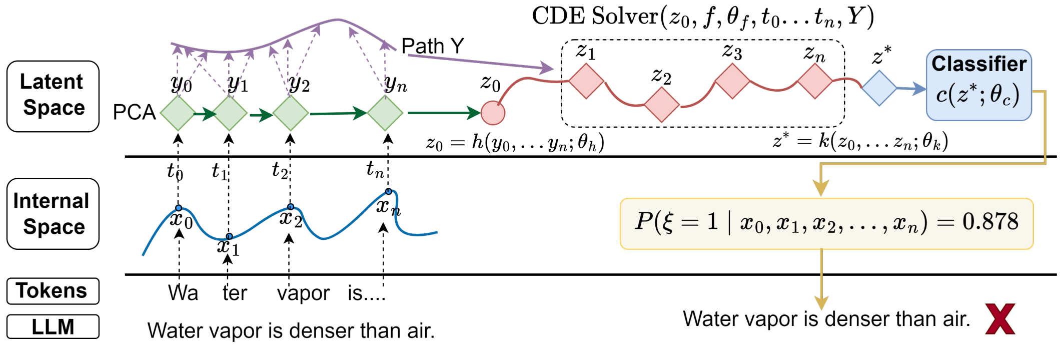


Figure 2: Computation graph of HD-NDEs detecting hallucination via Neural CDEs. The statement is processed by LLMs, from which we extract embedding information of each token in the internal space to construct the trajectory $\boldsymbol{x} = (x_0, x_1, x_2, ..., x_n)$, with corresponding time points $(t_0, t_1, ..., t_n)$. The PCA processes the trace and generates states $\boldsymbol{y} = (y_0, y_1, y_2, ..., y_n)$. These states are used to parameterize a latent space representation z_0 and extract control path Y. The CDE Solver predicts future latent states, forming $\boldsymbol{z} = (z_0, z_1, z_2, ..., z_n)$. From these latent states, z^* is derived using the function k parameterized by θ_k. z^* is then passed through a simple classifier to produce the final incorrect factuality score $P(\xi = 1|\boldsymbol{x})$.

et al. (2018) showed that any parametric ODE solver can be conceptualized as a deep learning framework with infinite depth. Chen et al. (2018) achieved ResNet-comparable results with a drastically lower number of parameters and memory complexity by parameterizing hidden layer derivatives and using ODE solvers. In addition, Lu et al. (2019) was the first to draw analogies between transformers and dynamical systems, conceptualizing the transformer as a numerical approximation of ODEs. Furthermore, Neural DEs play important roles in interpolation, forecasting, and classification tasks in time series data (Kidger et al., 2020a; Liang et al., 2021; Li et al., 2020; Oh et al., 2024; Li et al., 2022).

3 Methodology: HD-NDEs

We denote the generated text as a sequence of tokens $o_{0:n} = (o_0, o_1, ..., o_n)$, where o_t represents the t-th token. Given a generated text sample $\mathbf{o} = o_{0:n}$, our objective is to predict $P(\xi|\mathbf{o})$ where $\xi \in \{0, 1\}$ serves as the hallucination indicator variable, with $\xi = 1$ indicating a hallucination and $\xi = 0$ otherwise. Naturally, each token o_t is associated with an internal state representation $x_t \in \mathbb{R}^{d_x}$, derived from the specific hidden layer embeddings corresponding to token t. We generally use the embedding from the last layer to represent each token, where d_x denotes the embedding dimension. The value of d_x varies across models; for instance, $d_x = 4096$ for LLama-7B, while $d_x = 5120$ for LLama-13B.

3.1 Neural DEs

To capture the dynamic behavior of LLMs, we utilize Neural ODEs, Neural CDEs, and Neural SDEs to model the evolution in the latent space. Neural ODEs describe smooth, continuous-time dynamics using deterministic equations, Neural CDEs introduce control signals to guide system evolution. Furthermore, Neural SDEs incorporate stochasticity to account for uncertainty or noise within the system. Figure 2 illustrates hallucination detection using HD-NDEs with Neural CDEs.

Neural ODEs. Let $\boldsymbol{x} = x_{0:n} = (x_0, ..., x_n) \in \mathbb{R}^{d_x}$ denote the embeddings in the internal space. $\boldsymbol{x}$ is projected into $\boldsymbol{y} = y_{0:n} = (y_0, ..., y_n) \in \mathbb{R}^{d_y}$ by PCA. Consider a latent representation $z(t) \in \mathbb{R}^{d_z}$ at time t in latent space, which is given by

$$z(t) = z(0) + \int_0^t f(s, z(s); \theta_f)ds \quad \text{with} \quad z(0) = h(\boldsymbol{y}; \theta_h), \tag{1}$$

where $h : \mathbb{R}^{d_y} \to \mathbb{R}^{d_z}$ is an function with parameter θ_h and $f(t, z(t); \theta_f)$ is a neural network parameterized by θ_f to approximate $\frac{d\boldsymbol{z}(t)}{dt}$. Neural ODEs rely on ODE solvers, such as the explicit Euler method (Euler, 1845), to solve the integral problem in (1). Since we can freely choose the upper limit t of the integration, we can predict z at any time t. That is, once $h(\cdot; \theta_h)$ and $f(\cdot, \cdot; \theta_f)$ have been learned, then we are able to compute $\boldsymbol{z}(t)$ for any $t \geq 0$.

Neural CDEs. The solution to Neural ODEs is determined by its initial condition, making it inadequate for incorporating incoming information into a differential equation. To address this issue, Kidger et al. (2020b) proposed Neural CDEs by combining a controlled path $Y(t)$ of the underlying time-series data. Specifically, given the sequential data $\boldsymbol{y} = (y_0, y_1, \ldots, y_n)$, $z(t)$ is determined by

$$\begin{aligned} z(t) &= z(0) + \int_0^t f(s, z(s); \theta_f) dY(s) \\ \text{with} \quad & z(0) = h(\boldsymbol{y}; \theta_h), \end{aligned} \tag{2}$$

where $Y(t)$ is chosen as a natural cubic spline path (Kidger et al., 2020b) or hermite cubic splines with backward differences (James et al., 2021) of the underlying time-series data. Differently from Neural ODEs, $f(t, z(t); \theta_f)$ is a neural network parameterized by θ_f to approximate $\frac{dz(t)}{dY(t)}$.

Neural SDEs. Neural SDEs allow for describing the stochastic evolution of trace, rather than the deterministic evolution (Kidger et al., 2021b,a). The latent representation $z(t)$ of Neural SDEs is governed by the following SDE:

$$\begin{aligned} & z(t) = z(0) + \int_0^t f(s, z(s); \theta_f) ds \\ & + \int_0^t g(s, z(s); \theta_g) dW(s) \quad \text{with} \quad z(0) = h(\boldsymbol{y}; \theta_h) \end{aligned} \tag{3}$$

where $\{W_t\}_{t \geq 0}$ is a d_z-dimensional Brownian motion, $f(\cdot, \cdot; \theta_f)$ is the drift function, and $g(\cdot, \cdot; \theta_g)$ is the diffusion function. Drift and diffusion functions are represented by neural networks.

3.2 Classifier

We derive z^* from the latent states $\boldsymbol{z} = (z_0, z_1, z_2, ..., z_n)$ using the function $k(\theta_k)$. The classifier $c(\theta_c)$ then classifies z^*. In this work, the classifier is implemented as a simple linear layer followed by a sigmoid function.

3.3 DEs Solvers and Adjoint Methods

For simplicity, we denote the parameters of neural networks used in $k(\theta_k)$, $c(\theta_c)$ and Equations (1), (2), (3) as $\boldsymbol{\theta}$. After choosing one from Neural ODEs, Neural CDEs, or Neural SDEs to capture the state generation process in latent space, two natural questions arise: (1) How can we generate the subsequent latent states $(z(t_1), z(t_2), \ldots)$ based on $z(0)$ and $\boldsymbol{\theta}$? (2) How can we update the parameters $\boldsymbol{\theta}$ to ultimately obtain the optimal solution $\boldsymbol{\theta}^*$? Sections 3.3.1 and 3.3.2 answer the above two questions respectively.

3.3.1 DE Solvers for Forward Propagation

We begin by introducing two common ODE solvers: first-order and high-order schemes. Numerical methods for Neural CDEs and Neural SDEs can be adapted from these approaches.

First-order ODE Solvers. Euler method (Euler, 1845) is the simplest method for solving ODEs. The transformation at each time step can be expressed as:

$$z_{t+1} = z_t + \frac{dz(t)}{dt} = z_t + f(t, \boldsymbol{z}(t); \theta_f). \tag{4}$$

High-order ODE Solvers. The Euler method is not "precise" because it is a first-order method, and naturally with local truncation errors. The global error will be accumulated if we want to capture a long timestep trajectory. Herein, we use the Runge-Kutta method (Runge, 1895) for a higher-order solution to ODEs. They are a classic family of iterative methods with different orders of precision. More formally, the explicit Runge-Kutta methods of an n-step solution are defined to be:

$$\begin{aligned} z_{t+1} &= z_t + \sum_{i=1}^{n} \gamma_i Z_i, Z_1 = \Delta t f(t, z_t; \theta_f), \\ Z_i &= \Delta t f(t + \alpha_i \Delta t, z_t + \sum_{j=1}^{i-1} \beta_{ij} Z_j; \theta_f) \end{aligned} \tag{5}$$

where Δt is the time size and could be simply 1 in most cases. Z_i is an intermediate approximation to the solution at step $t + \alpha_i \Delta t$. α, β and γ are coefficients which can be determined by the series of z_{t+1}. In this work, we use fourth-order Runge-Kutta (RK4) for solving Equation (1), details in Appendix A.

The Neural CDEs problem in (2) can be solved by using the above-mentioned ODE solvers since $\frac{dz(t)}{dt} = f(t, z(t); \theta_f) \frac{dX(t)}{dt}$. However, the Neural SDE problem (3) requires additional handling of stochastic noise, making its solution methods more complex. Herein, we use the Euler-Maruyama method designed to handle noise terms, which is given by

$$z_{t+1} = z_t + f(t, z(t); \theta_f) + g(t, z(t); \theta_f) \mathcal{Z}, \tag{6}$$

where $\mathcal{Z} \sim \mathcal{N}(0, 1)$ is a standard normal random variable with mean 0 and variance 1.

3.3.2 Adjoint Methods for Back Propagation

Since Neural DEs are continuous-time models computed through DE solvers, standard backpropagation cannot be directly applied. Chen et al. (2018)

Method	Company*						Fact*					
	LLama-2-7B	LLama-2-13B	Alpaca-13B	Vicuna-13B	Mistral-7B-0.3	Gemma-2-9B	LLama-2-7B	LLama-2-13B	Alpaca-13B	Vicuna-13B	Mistral-7B-0.3	Gemma-2-9B
Prompt-based Methods												
P(True)	51.4	49.1	50.6	52.4	51.5	51.0	54.1	53.6	53.8	51.3	53.1	52.6
Logit-based Methods												
AvgProb	59.0	59.2	53.0	60.2	59.3	58.0	59.5	59.3	54.2	58.3	56.3	61.2
AvgEnt	54.0	56.4	54.2	53.3	56.2	54.1	54.2	54.1	50.3	51.2	53.2	53.4
EUBHD	52.5	53.2	54.1	55.3	53.8	55.6	59.7	60.8	59.4	57.9	56.5	58.2
Classification-based Methods												
SAPLMA	54.0	58.2	59.3	68.2	63.2	64.8	58.3	62.4	59.8	65.5	59.6	61.2
MIND	56.4	60.3	62.4	69.8	60.1	65.9	59.6	63.7	61.8	70.7	60.1	62.8
Probe@Exact	55.9	60.7	61.2	67.2	64.4	63.9	60.7	63.9	60.2	68.4	59.2	63.7
ODEs	59.7	65.3	67.8	<u>72.9</u>	63.5	71.4	58.6	66.9	64.3	70.4	62.4	66.7
CDEs	<u>65.9</u>	<u>72.8</u>	**75.3**	**79.8**	<u>66.9</u>	**73.6**	<u>67.5</u>	**74.8**	**72.9**	<u>76.7</u>	<u>74.1</u>	**73.9**
SDEs	**73.8**	**78.4**	<u>70.5</u>	72.3	**71.3**	<u>72.8</u>	**70.3**	<u>73.1</u>	<u>70.3</u>	**78.6**	**75.3**	<u>72.5</u>

Method	City*						Invention*					
	LLama-2-7B	LLama-2-13B	Alpaca-13B	Vicuna-13B	Mistral-7B-0.3	Gemma-2-9B	LLama-2-7B	LLama-2-13B	Alpaca-13B	Vicuna-13B	Mistral-7B-0.3	Gemma-2-9B
P(True)	53.1	54.7	57.3	56.2	49.8	51.7	49.3	50.2	47.6	54.7	51.9	49.7
Logit-based Methods												
AvgProb	54.2	56.3	51.5	59.2	53.9	55.6	51.2	51.3	49.4	55.3	53.7	52.9
AvgEnt	49.1	50.2	49.3	52.4	47.1	52.0	45.1	46.3	48.4	47.5	47.1	45.9
EUBHD	59.9	61.1	60.3	58.5	59.5	60.7	60.1	59.8	57.9	59.4	60.6	58.9
Classification-based Methods												
SAPLMA	60.0	69.3	59.4	64.5	63.3	64.7	59.2	66.0	52.4	69.3	61.3	59.4
MIND	64.5	71.3	62.6	65.8	63.0	65.2	60.5	65.1	53.6	71.2	64.1	58.6
Probe@Exact	65.8	70.4	61.8	66.9	62.7	64.3	61.1	63.0	55.5	70.2	63.6	57.3
ODEs	73.0	<u>82.3</u>	71.2	73.2	75.1	72.4	60.3	<u>80.9</u>	69.7	80.4	<u>79.1</u>	<u>80.5</u>
CDEs	<u>75.7</u>	80.6	<u>72.1</u>	<u>80.1</u>	**77.5**	<u>77.2</u>	**75.9**	**88.3**	<u>73.8</u>	<u>81.2</u>	**81.3**	**83.7**
SDEs	**79.1**	**89.8**	**74.3**	**82.5**	<u>76.4</u>	**79.8**	<u>68.7</u>	79.6	**74.2**	**85.9**	74.3	79.5

Table 1: The detection AUC-ROC (%) of different approaches across multiple LLMs on Company*, Fact*, City* and Invention*. **Bold** and <u>underlined</u> numbers denote the best and second-best values, respectively. ODEs, CDEs, and SDEs are the abbreviations of Neural ODEs, Neural CDEs, and Neural SDEs, respectively.

applied the adjoint sensitivity method (Pontryagin et al., 1962) to compute gradients for Neural ODEs. Specifically, to optimize the loss function L, we require gradients with respect to θ. The first step is to determine how the gradient of the loss depends on the hidden state $z(t)$ at each instant. This quantity is called the adjoint

$$a(t) = \frac{\partial L}{\partial z(t)}. \tag{7}$$

Its dynamics are given by another ODE, which can be thought of as the instantaneous analog of the chain rule:

$$\frac{da(t)}{dt} = -\alpha(t)^T \frac{\partial f(t, z(t); \theta_f)}{\partial z}. \tag{8}$$

We can compute $a(t)$ by another call to an ODE solver. Computing the gradients with respect to the parameters θ requires evaluating a third integral, which depends on both $z(t)$ and $a(t)$:

$$\frac{dL}{d\theta} = -\int_{t_1}^{t_0} \alpha(t)^T \frac{\partial f(t, z(t), \theta_f)}{\partial z}. \tag{9}$$

In addition, Kidger et al. (2020a) and Li et al. (2020) proposed the adjoint sensitivity methods for Neural CDEs and Neural SDEs, respectively. In our work, we build upon the above methods to update the parameters of neural networks.

4 Experimental Settings

4.1 Datasets

True-False Dataset. The original dataset consists of six sub-datasets, each named after its subject matter (Azaria and Mitchell, 2023). We follow the method proposed in Levinstein and Herrmann (2024) to create factual and non-factual statements containing subtle differences. Specifically, we prompt GPT-4o to generate new statements that are factually opposite to the original while maintaining only minor word differences. For example, we obtain a non-factual statement "The earth doesn't orbit the sun." from the factual statement "The earth orbits the sun." For our experiments, we randomly select 550, 560, 500, and 500 statements from the *Companies*, *Scientific Facts*, *Cities*, and *Inventions* sub-datasets, respectively. The resulting datasets are referred to as *Company**, *Fact**, *City**, and *Invention**. This dataset poses a greater challenge for hallucination detection.

Question Answering Datasets. We utilize four widely used question answering datasets, including *TruthfulQA* (Lin et al., 2022), *TriviaQA* (Joshi et al., 2017), "QA" subset of *HaluEval* (Li et al., 2023b) and *NQ* (Kwiatkowski et al., 2019). Each question

Method	TruthfulQA						TriviaQA					
	LLama-2-7B	LLama-2-13B	Alpaca-13B	Vicuna-13B	Mistral-7B-0.3	Gemma-2-9B	LLama-2-7B	LLama-2-13B	Alpaca-13B	Vicuna-13B	Mistral-7B-0.3	Gemma-2-9B
Prompt-based Methods												
P(True)	52.5	53.6	51.3	54.0	49.7	50.0	42.3	44.6	42.1	50.6	48.3	49.2
Logit-based Methods												
AvgProb	51.4	54.6	53.3	55.1	48.3	45.6	44.1	48.3	43.1	47.1	48.5	48.0
AvgEnt	49.4	53.0	52.7	53.6	51.0	52.1	41.1	43.2	41.6	44.5	47.6	43.5
EUBHD	81.2	78.1	77.4	79.7	80.3	81.4	80.5	81.1	78.2	79.1	80.6	81.7
Consistency-based Methods												
Unigram	57.6	62.2	60.1	63.4	60.9	61.8	56.8	60.4	57.9	61.3	59.5	60.3
NLI	60.6	63.7	61.6	65.1	61.3	62.5	59.4	63.2	58.1	64.5	61.4	62.1
INSIDE	79.8	81.2	80.0	82.1	81.8	82.4	81.7	82.6	78.1	80.8	81.3	82.0
Classification-based Methods												
SAPLMA	87.5	86.3	84.9	88.6	81.3	85.4	80.0	81.1	80.2	85.0	84.1	83.4
MIND	88.0	87.1	84.5	88.9	83.6	85.7	79.4	82.3	81.1	83.2	84.5	81.1
Probe@Exact	85.7	86.8	85.2	88.7	82.9	87.4	80.3	82.5	81.9	84.4	84.1	84.0
ODEs	84.2	87.9	83.1	83.8	82.4	85.3	81.7	83.6	80.5	85.9	83.7	84.6
CDEs	86.7	84.0	84.3	89.2	83.9	**87.7**	**83.7**	**84.9**	**82.6**	**86.3**	84.1	**85.0**
SDEs	**88.3**	**89.3**	**86.4**	**89.5**	**85.1**	87.0	81.0	83.3	81.5	84.3	**85.1**	83.2

Method	HaluEval						NQ					
	LLama-2-7B	LLama-2-13B	Alpaca-13B	Vicuna-13B	Mistral-7B-0.3	Gemma-2-9B	LLama-2-7B	LLama-2-13B	Alpaca-13B	Vicuna-13B	Mistral-7B-0.3	Gemma-2-9B
Prompt-based Methods												
P(True)	46.7	48.9	51.6	50.2	49.7	46.8	54.7	56.8	51.0	53.4	52.1	51.3
Logit-based Methods												
AvgProb	42.1	44.4	43.2	45.7	43.6	44.5	54.3	55.9	53.1	56.4	54.7	55.3
AvgEnt	47.3	48.5	46.1	51.4	49.7	50.3	53.9	54.6	54.2	55.2	53.8	54.6
EUBHD	71.9	78.1	71.3	76.0	70.5	72.6	73.9	79.4	76.8	73.2	71.7	70.3
Consistency-based Methods												
Unigram	58.2	57.1	57.9	57.6	62.3	59.4	63.1	65.2	62.9	67.8	64.5	65.1
NLI	61.3	60.2	55.2	62.5	63.1	64.4	64.2	66.9	63.8	65.4	62.6	64.0
INSIDE	74.5	76.9	73.3	75.2	76.0	75.8	76.8	77.1	74.3	75.8	76.4	75.9
Classification-based Methods												
SAPLMA	87.0	90.1	89.5	93.1	89.4	90.5	89.1	90.5	87.6	93.2	90.3	88.9
MIND	86.1	93.8	93.7	92.9	94.5	91.0	90.5	93.6	92.7	90.6	87.2	89.5
Probe@Exact	88.3	92.4	93.5	94.1	93.4	92.1	92.0	91.9	92.8	92.3	88.6	90.3
ODEs	89.5	93.9	92.1	95.4	91.2	90.5	91.3	92.1	90.5	92.4	89.7	90.0
CDEs	91.4	**97.1**	95.3	**96.9**	**95.4**	**96.0**	93.7	**95.2**	92.1	**93.6**	**90.5**	**91.8**
SDEs	**92.8**	95.4	**97.1**	93.1	93.7	92.6	**94.1**	93.2	**93.5**	91.1	89.7	90.9

Table 2: The detection AUC-ROC (%) for different approaches over multiple LLMs on TruthfulQA, TriviaQA, HaluEval and NQ.

is accompanied by a truthful and a hallucinatory answer. Unlike the True-False dataset, we use the Levenshtein (Levenshtein, 1966) distance to select the pair of correct and incorrect answers with the greatest textual difference. These pairs, along with the original questions, form the data used for our experiments. Finally, we generate 1,000 samples in each of the four aforementioned datasets.

4.2 Models

We evaluate both our method and baseline approaches using common open-source LLMs, including LLama-2-7B, LLama-2-13B (Touvron et al., 2023), Alpaca-13B (Taori et al., 2023), Vicuna-13B-v1.3 (Chiang et al., 2023), Mistral-7B-v0.3 (Jiang et al., 2023) and Gemma-2-9B (Team et al., 2024).

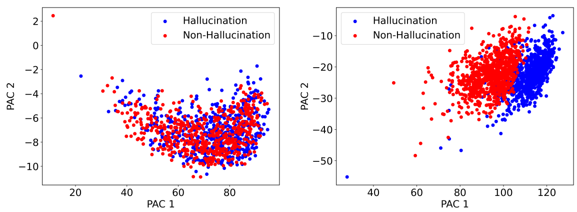
4.3 Baselines

We choose the following four types of hallucination detection methods as baselines. More details are shown in Appendix C.

Prompt-based methods utilize a simple prompt template to enable the model to assess the correctness of the response. Here, we use *P(True)*, proposed in Kadavath et al. (2022), as a representative of this class of methods.

Logit-based methods use the uncertainty of LLMs' outputs to detect hallucination. We adopt the two effective metrics used in Huang et al. (2023), namely *AvgProb*, *AvgEnt*, to aggregate logit-based uncertainty of all tokens to measure sentence uncertainty. In addition, we also compare our approach with *EUBHD* (Zhang et al., 2023a), which focuses on key tokens rather than considering all tokens.

Consistency-based methods are motivated by the idea that if an LLM possesses specific knowledge, the sampled responses are likely to be similar and contain consistent facts. In this work, we apply two important variants proposed in Manakul et al. (2023), namely *Unigram* and *Natural Lan-*



(a) Fact*

(b) TruthfulQA

Figure 3: 2D PCA projection of the last hidden layer's embedding for the final token on Fact* and TruthfulQA. Blue and red dots represent hallucinations and non-hallucinations, respectively.

guage Inference (NLI), as well as *INSIDE* by Chen et al. (2024), which leverages the eigenvalues of the covariance matrix of responses.

Classification-based methods train a classifier on a dataset containing labeled statements. We choose *SAPLMA* (Azaria and Mitchell, 2023), *MIND* (Su et al., 2024) and *Probe@Exact* (Orgad et al., 2025) as representatives of this type of method. Unlike SAPLMA, which relies on pre-annotated datasets, MIND automatically labels data during the detection process to train its classifier. SAPLMA utilizes information from the last token, whereas Probe@Exact relies on information from potential correct tokens.

4.4 Evaluation Metric

We utilize *AUC-ROC*, which stands for the area under the ROC curve, to objectively evaluate the effectiveness of models. The higher value of AUC-ROC, the stronger the ability of this method for hallucination detection. All experiments are conducted on NVIDIA A100 GPUs with 40GB of memory.

4.5 Implementation Details

HD-NDEs. To reduce computational complexity, we employ PCA to reduce the dimensionality of the internal space to $K = 1024$. The integrands $h(\cdot;\theta_h)$, $f(\cdot,\cdot;\theta_f)$, $g(\cdot,\cdot;\theta_g)$ in Equations (1), (2) and (3) are taken to be feedforward neural networks. Specifically, we use a single hidden layer network to represent $h(\cdot;\theta_h)$ in all variants of our methods. We use an 8-layer neural network to represent $f(\cdot,\cdot;\theta_f)$ in Neural CDEs and 10-layer neural networks for $f(\cdot,\cdot;\theta_f)$ in both Neural ODEs and Neural SDEs. Additionally, $g(\cdot,\cdot;\theta_g)$ in Neural SDEs is represented by a 4-layer neural network. A final linear layer is always applied to map the latent state to the output. We use ReLU activation functions for Neural CDEs and Neural SDEs, while tanh activations are used for Neural ODEs. The binary cross-entropy loss is applied to the sigmoid of the model output. Additionally, we employ the Adam optimizer with a learning rate of 0.001, a batch size of 32, and 50 epochs.

Classification-based methods. The classifier receives embeddings from the last layer of LLMs. In ablation studies, we discuss the results of using information from the middle layers. Different classifiers are used for different methods. More implementation details are introduced in Appendix D.

5 Experimental Results and Analysis

5.1 Effectiveness of HD-NDEs

True-False Dataset. The comprehensive results are demonstrated in Table 1. Since consistency-based methods rely on question-and-answer pairs, and the True-False dataset is not structured in this format, we do not include this type of method as a comparison in this dataset. **It is obvious that our methods surpass SAPLMA, MIND, and Probe@Exact by a noticeable margin, evidenced by an average increase of over 14% in the detection of AUC-ROC across different models and subsets.** Particularly, Neural CDEs outperform SAPLMA by 24.3% on Invention* when using Gemma-2-9B. Even in the worst case, Neural ODEs perform comparably to SAPLMA on Fact* based on LLama-2-7B. Furthermore, in most cases, prompt-based methods and logit-based methods perform worse than the classification-based methods.

For different variants of our approach, **we can find that Neural CDEs and Neural SDEs outperform Neural ODEs.** As shown in Table 1, the best and second-best values are achieved by Neural CDEs and Neural SDEs models in 19 out of 24 cases. The likely reason is that Neural CDEs and Neural SDEs can capture richer dynamical behaviors than Neural ODEs. As mentioned in Section 3, Neural CDEs incorporates control theory, enabling the dynamic system to account for the influence of incoming information, and Neural SDEs introduces stochasticity into the modeling process. While Neural ODEs assumes deterministic dynamics, which can limit its flexibility in modeling the dynamics of LLMs.

Question Answering Datasets. Table 2 shows the results on question answering datasets. **EUBHD, SAPLMA, MIND, and Probe@Exact**

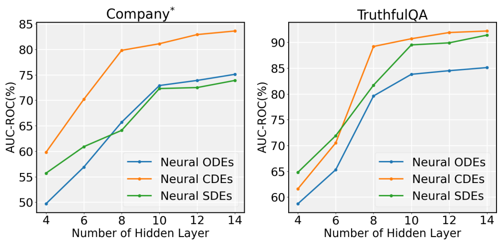


Figure 4: The impact of the number of hidden layers on Vicuna-13B: Company* and TruthfulQA.

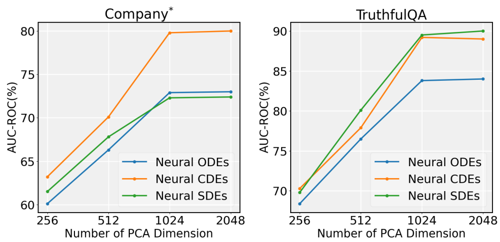


Figure 5: The impact of the PCA projection dimensions on Vicuna-13B: Company* and TruthfulQA.

demonstrate significantly better performance on the four question answering datasets compared to the True-False dataset across all six models. Notably, SAPLMA, MIND and and Probe@Exact achieve comparable performance to HD-NDEs, including Neural ODEs, Neural CDEs, and Neural SDEs, with a difference of less than 6%. Specifically, SAPLMA outperforms Neural ODEs and Neural CDEs on TruthfulQA using LLama-2-7B and Alpaca-13B, while remaining slightly behind Neural SDEs. On the NQ dataset, MIND and Probe@Exact achieve the second-highest performance among all methods on LLama-2-13B and Alpaca-13B, respectively. Meanwhile, INSIDE ranks just below Neural CDEs on TriviaQA with LLama-2-7B.

5.2 Analysis

We try to understand why HD-NDEs obviously outperforms SAPLMA, and MIND on the True-False dataset, yet performs comparably to them on the question-answer datasets. We use the subsets Fact* from True-False and TruthfulQA as examples. We employ PCA to reduce the dimensions of the hidden embeddings, retaining the two dominant components. The results are shown in Figure 3. The 2D PCA projection reveals a significant overlap between correct and incorrect statements in True-False, with many points intertwined. The poor separation causes other methods to perform only marginally better than random guessing in many cases. In contrast, the 2D PCA projections of TruthfulQA reveal a much clearer distinction between hallucination and non-hallucination. The statements in TruthfulQA exhibit substantial variation, as we use the Levenshtein distance to select statements with significant differences. This allows other baselines to more easily differentiate them based on the embeddings of the final token. Appendix E contains more results on other datasets.

5.3 Ablation Studies

Number of Hidden Layers. An important factor impacting the performance of detection methods is the number of hidden layers in neural networks representing $f(s, z(s); \theta_f)$. Results are shown in Figure 4. Specifically, the performance of Neural CDEs improves substantially as the number of layers increases up to 8, with further increases beyond 8 still showing gains but at a slower pace. For Neural ODEs and Neural SDEs, this turning point occurs when the layer number reaches 10, based on the results from both datasets. Finally, we ultimately set 8 layers for Neural CDEs and 10 layers for both Neural ODEs and Neural SDEs in Section 4.5.

Dimensions in Latent Space. Another key factor is the dimension of the latent state after being mapped from the internal space to the latent space by PCA. We then examine the impact of varying dimensions as shown in Figure 5. **An evident improvement in detection effectiveness is associated with retaining more components during down-projection.** Therefore, all three variants of our methods achieve the best performance on both datasets when the dimension is set to 1024, affirming our hyperparameter setting in Section 4.5.

More Results. In Appendix F, we explore the impact of using activations from middle layers, and Appendix G shows the effectiveness where classifiers trained on out-of-domain datasets.

6 Conclusion

In this paper, we introduce HD-NDEs, which tracks the dynamic changes in latent space. HD-NDEs can effectively detect logical or factual inconsistencies that arise in the generated text. Comprehensive empirical results demonstrate that our approach surpasses various state-of-the-art methods by over 14% on the True-False Dataset.

Limitations

This work identifies three major limitations. First, the model's training process is approximately twice as long as that of the SAPLMA method. Second, NeuralDEs, as currently presented, do not provide uncertainty estimates for their predictions, though such extensions may be feasible in the future. Third, we experiment with a limited set of numerical schemes, and other methods could potentially exploit the structure of differential equations to further improve performance.

Ethics and Broader Impact

We sampled a portion of the data from existing datasets for our experiments, which may affect the accuracy of some of our conclusions.

References

Hervé Abdi and Lynne J Williams. 2010. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459.

Fatemeh Azadi, Heshaam Faili, and Mohammad Javad Dousti. 2023. PMI-align: Word alignment with point-wise mutual information without requiring parallel training data. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 12366–12377, Toronto, Canada. Association for Computational Linguistics.

Amos Azaria and Tom Mitchell. 2023. The internal state of an LLM knows when it's lying. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 967–976, Singapore. Association for Computational Linguistics.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. 2023. Discovering latent knowledge in language models without supervision. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Bo Chang, Lili Meng, Eldad Haber, Frederick Tung, and David Begert. 2018. Multi-level residual networks from dynamical systems view. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Chao Chen, Kai Liu, Ze Chen, Yi Gu, Yue Wu, Mingyuan Tao, Zhihang Fu, and Jieping Ye. 2024. INSIDE: LLMs' internal states retain the power of hallucination detection. In *The Twelfth International Conference on Learning Representations*.

Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. 2018. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 6572–6583.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Jeongwhan Choi, Hwangyong Choi, Jeehyun Hwang, and Noseong Park. 2022. Graph neural controlled differential equations for traffic forecasting. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pages 6367–6374. AAAI Press.

Jinhao Duan, Hao Cheng, Shiqi Wang, Chenan Wang, Alex Zavalny, Renjing Xu, Bhavya Kailkhura, and Kaidi Xu. 2023. Shifting attention to relevance: Towards the uncertainty estimation of large language models. *ArXiv preprint*, abs/2307.01379.

Subhabrata Dutta, Tanya Gautam, Soumen Chakrabarti, and Tanmoy Chakraborty. 2021. Redesigning the transformer architecture with insights from multi-particle dynamical systems. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 5531–5544.

Leonhard Euler. 1845. *Institutionum calculi integralis*, volume 4. impensis Academiae imperialis scientiarum.

Jiahui Geng, Fengyu Cai, Yuxia Wang, Heinz Koeppl, Preslav Nakov, and Iryna Gurevych. 2023. A survey of language model confidence estimation and calibration. *ArXiv preprint*, abs/2311.08298.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2023. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Yuheng Huang, Jiayang Song, Zhijie Wang, Huaming Chen, and Lei Ma. 2023. Look before you leap: An exploratory study of uncertainty measurement for large language models. *ArXiv preprint*, abs/2307.10236.

Morrill James, Kidger Patrick, Yang Lingyi, and Lyons Terry. 2021. Neural controlled differential equations for online prediction tasks. *ArXiv preprint*, abs/2106.11028.

Sheo Yon Jhin, Heejoo Shin, Sujie Kim, Seoyoung Hong, Minju Jo, Solhee Park, Noseong Park, Seungbeom Lee, Hwiyoung Maeng, and Seungmin Jeon. 2024. Attentive neural controlled differential equations for time-series classification and forecasting. *Knowledge and Information Systems*, 66(3):1885–1915.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

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

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas

Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. 2022. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*.

Patrick Kidger. 2022. On neural differential equations. *ArXiv preprint*, abs/2202.02435.

Patrick Kidger, James Foster, Xuechen Li, and Terry J. Lyons. 2021a. Efficient and accurate gradients for neural sdes. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 18747–18761.

Patrick Kidger, James Foster, Xuechen Li, and Terry J. Lyons. 2021b. Neural sdes as infinite-dimensional gans. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 5453–5463. PMLR.

Patrick Kidger, James Morrill, James Foster, and Terry J. Lyons. 2020a. Neural controlled differential equations for irregular time series. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Patrick Kidger, James Morrill, James Foster, and Terry J. Lyons. 2020b. Neural controlled differential equations for irregular time series. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Jannik Kossen, Jiatong Han, Muhammed Razzak, Lisa Schut, Shreshth Malik, and Yarin Gal. 2024. Semantic entropy probes: Robust and cheap hallucination detection in llms. *ArXiv preprint*, abs/2406.15927.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

V Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Proceedings of the Soviet physics doklady*.

Benjamin A Levinstein and Daniel A Herrmann. 2024. Still no lie detector for language models: Probing empirical and conceptual roadblocks. *Philosophical Studies*, pages 1–27.

Bei Li, Quan Du, Tao Zhou, Yi Jing, Shuhan Zhou, Xin Zeng, Tong Xiao, JingBo Zhu, Xuebo Liu, and Min Zhang. 2022. ODE transformer: An ordinary differential equation-inspired model for sequence generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8335–8351, Dublin, Ireland. Association for Computational Linguistics.

Junyi Li, Xiaoxue Cheng, Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2023a. HaluEval: A large-scale hallucination evaluation benchmark for large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6449–6464, Singapore. Association for Computational Linguistics.

Junyi Li, Xiaoxue Cheng, Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2023b. HaluEval: A large-scale hallucination evaluation benchmark for large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6449–6464, Singapore. Association for Computational Linguistics.

Qing Li, Chenyang Lyu, Jiahui Geng, Derui Zhu, Maxim Panov, and Fakhri Karray. 2024. Reference-free hallucination detection for large vision-language models. *ArXiv preprint*, abs/2408.05767.

Xuechen Li, Ting-Kam Leonard Wong, Ricky T. Q. Chen, and David Duvenaud. 2020. Scalable gradients for stochastic differential equations. In *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]*, volume 108 of *Proceedings of Machine Learning Research*, pages 3870–3882. PMLR.

Yuxuan Liang, Kun Ouyang, Hanshu Yan, Yiwei Wang, Zekun Tong, and Roger Zimmermann. 2021. Modeling trajectories with neural ordinary differential equations. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pages 1498–1504. ijcai.org.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252, Dublin, Ireland. Association for Computational Linguistics.

Yiping Lu, Zhuohan Li, Di He, Zhiqing Sun, Bin Dong, Tao Qin, Liwei Wang, and Tie-Yan Liu. 2019. Understanding and improving transformer from a multi-particle dynamic system point of view. *ArXiv preprint*, abs/1906.02762.

Yiping Lu, Aoxiao Zhong, Quanzheng Li, and Bin Dong. 2018. Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 3282–3291. PMLR.

Potsawee Manakul, Adian Liusie, and Mark Gales. 2023. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9004–9017, Singapore. Association for Computational Linguistics.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12076–12100, Singapore. Association for Computational Linguistics.

Niels Mündler, Jingxuan He, Slobodan Jenko, and Martin Vechev. 2023. Self-contradictory hallucinations of large language models: Evaluation, detection and mitigation. *ArXiv preprint*, abs/2305.15852.

YongKyung Oh, Dongyoung Lim, and Sungil Kim. 2024. Stable neural stochastic differential equations in analyzing irregular time series data. *ArXiv preprint*, abs/2402.14989.

Hadas Orgad, Michael Toker, Zorik Gekhman, Roi Reichart, Idan Szpektor, Hadas Kotek, and Yonatan Belinkov. 2025. LLMs know more than they show: On the intrinsic representation of LLM hallucinations. In *The Thirteenth International Conference on Learning Representations*.

Lev Semenovich Pontryagin, EF Mishchenko, VG Boltyanskii, and RV Gamkrelidze. 1962. *The mathematical theory of optimal processes*. Routledge.

Carl Runge. 1895. Über die numerische auflösung von differentialgleichungen. *Mathematische Annalen*, 46(2):167–178.

Weihang Su, Changyue Wang, Qingyao Ai, Yiran Hu, Zhijing Wu, Yujia Zhou, and Yiqun Liu. 2024. Unsupervised real-time hallucination detection based on the internal states of large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 14379–14391, Bangkok, Thailand. Association for Computational Linguistics.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *ArXiv preprint*, abs/2307.09288.

Yuxia Wang, Revanth Gangi Reddy, Zain Muhammad Mujahid, Arnav Arora, Aleksandr Rubashevskii, Jiahui Geng, Osama Mohammed Afzal, Liangming Pan, Nadav Borenstein, Aditya Pillai, et al. 2023. Factcheck-gpt: End-to-end fine-grained document-level fact-checking and correction of llm output. *ArXiv preprint*, abs/2311.09000.

Jerry Wei, Chengrun Yang, Xinying Song, Yifeng Lu, Nathan Hu, Dustin Tran, Daiyi Peng, Ruibo Liu, Da Huang, Cosmo Du, and Quoc V. Le. 2024. Long-form factuality in large language models. *ArXiv preprint*, abs/2403.18802.

Tianhang Zhang, Lin Qiu, Qipeng Guo, Cheng Deng, Yue Zhang, Zheng Zhang, Chenghu Zhou, Xinbing Wang, and Luoyi Fu. 2023a. Enhancing uncertainty-based hallucination detection with stronger focus. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 915–932, Singapore. Association for Computational Linguistics.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, et al. 2023b. Siren's song in the ai ocean: a survey on hallucination in large language models. *ArXiv preprint*, abs/2309.01219.

Derui Zhu, Dingfan Chen, Qing Li, Zongxiong Chen, Lei Ma, Jens Grossklags, and Mario Fritz. 2024. PoLLMgraph: Unraveling hallucinations in large language models via state transition dynamics. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 4737–4751, Mexico City, Mexico. Association for Computational Linguistics.

A Fourth-order Runge-Kutta (RK4)

We can also define a fourth-order Runge-Kutta (RK4) block to be:

$$
\begin{aligned}
z_{t+1} &= z_t + \frac{\Delta t}{6}(Z_1 + 2Z_2 + 2Z_3 + Z_4) \\
Z_1 &= f(t, z_t; \theta_f) \\
Z_2 &= f(t + \frac{\Delta t}{2}, z_t + \frac{\Delta t}{2} Z_1; \theta_f) \\
Z_3 &= f(t + \frac{\Delta t}{2}, z_t + \frac{\Delta t}{2} Z_2; \theta_f) \\
Z_4 &= f(t + \Delta t, z_t + \Delta t Z_3; \theta_f)
\end{aligned}
\tag{10}
$$

B Question Answering Datasets

TruthfulQA consists of 873 questions, each with multiple correct and incorrect answers. For **HaluEval**, our experiments focused on the 'QA' subset comprising 10k records, where each record includes a question accompanied by both a truthful and a hallucinatory answer. The validation set of **NQ** consists of 3,610 QA pairs, while the validation set of **TriviaQA** (rc.nocontext subset) contains 9,960 deduplicated QA pairs. Unlike True-False, we use the Levenshtein (Levenshtein, 1966) distance to select the pair of correct and incorrect answers with the greatest textual difference. These pairs, along with the original questions, form the data used for our experiments. Finally, we generate 1,000 samples in each of the four aforementioned datasets.

C Baseline Methods

We collect logit-based and consistency-based methods proposed in (Manakul et al., 2023) to test the effectiveness of models.

C.1 Logit-based methods

To aggregate the uncertainty information obtained at the token level, we employ four metrics to aggregate token-level uncertainty into sentence level. In particular, a sentence-level uncertainty score can be obtained by taking either the maximum or average of the negative loglikelihood $-\log p_{ij}$ in a sentence:

$$\text{MaxProb}(i) = \max_j(-\log p_{ij}), \tag{11}$$

$$\text{AvgProb}(i) = -\frac{1}{J_i}\sum_{j=1}^{J_i}\log p_{ij}, \tag{12}$$

where p_{ij} is the probability of a token at a position j in the sentence i and J_i is the total number of tokens in the considered sentence. Additionally, one can also replace the negative loglikelihood $-\log p_{ij}$ with the entropy $\mathcal{H}_{ij}$:

$$\text{MaxEnt}(i) = \max_j \mathcal{H}_{ij}, \tag{13}$$

$$\text{AvgEnt}(i) = \frac{1}{J_i}\sum_{j=1}^{J_i}\mathcal{H}_{ij}, \tag{14}$$

where H_{ij} is the entropy of the token distribution for the j-th token in the sentence i.

C.2 Consistency-based Methods

Unigram. The concept behind Unigram is to develop a new model that approximates the LVLMs by samples $\{S^1, \ldots, S^N\}$ and get the LVLM's token probabilities using this model. As N increases, the new model gets closer to LVLMs. Due to time and cost constraints, we just train a simple n-gram model using the samples $\{S^1, \ldots, S^N\}$ as well as the main response R. We then compare the average and maximum of the negative probabilities of the sentence in response R using the following equations:

$$\mathcal{S}^{\text{Avg}}_{\text{n-gram}}(i) = -\frac{1}{J_i}\sum_{j=1}^{J_i}\log \hat{p}_{ij}, \tag{15}$$

$$\mathcal{S}^{\text{Max}}_{\text{n-gram}}(i) = \max_j(-\log \hat{p}_{ij}), \tag{16}$$

where $\hat{p}_{ij}$ is the probability of a token at position j of a sentence i.

Natural Language Inference (NLI) determines whether a hypothesis follows a premise, classified into either entailment/neutral/contradiction. In this work, we use DeBERTa-v3-large (He et al., 2023) fine-tuned to MNLI as the NLI model. The input for NLI classifiers is typically the premise concatenated to the hypothesis, which for NLI is the sampled passage S^n concatenated to the sentence to be assessed r_i in the response R. Only the logits associated with the 'entailment' and 'contradiction' classes are considered,

$$P(contradict \mid r_i, S^n) = \frac{\exp(z_e^{i,n})}{\exp(z_e^{i,n}) + \exp(z_c^{i,n})},$$

where $z_e^{i,n} = z_e(r_i, S^n)$ and $z_c^{i,n} = z_c(r_i, S^n)$ are the logits of the 'entailment' and 'contradiction' classes. NLI score for sentence r_i on samples $\{S^1, \ldots, S^N\}$ is then defined as,

$$\mathcal{S}_{\text{NLI}}(i) = \frac{1}{N}\sum_{n=1}^{N} P(contradict \mid r_i, S^n). \tag{17}$$

D Implementation Details

SAPLMA. We follow the majority of the experimental setup for SAPLMA as described in (Azaria and Mitchell, 2023). Its classifier employs a feedforward neural network featuring three hidden layers with decreasing numbers of hidden units (1024, 512, 256), all utilizing ReLU activations. The final layer is a sigmoid output. We use the Adam optimizer. The classifier is trained for 20 epochs with a learning rate of 5e-4 and a training batch size of 32. We use about three-quarters of the dataset to train a classifier based on a specific model, and then test its accuracy on the remaining quarter of the same dataset. The training and testing datasets are randomly split.

MIND. We follow the majority of the experimental setup for MIND as described in Su et al. (2024). The MIND classifier utilizes a 4-layer Multilayer Perceptron (MLP) network with a 20% dropout applied to the initial layer. The network architecture features decreasing hidden layer sizes of 256, 128, 64, and 2 for each layer. The Rectified Linear Unit (ReLU) activation function is used, with a learning rate of 5e-4, a weight decay of 1e-5, and a training batch size of 32.

Probe@Exact. We follow the majority of the experimental setup for Probe@Exact as described in Orgad et al. (2025). We employ the logistic regression model from the scikit-learn library as the probing classifier. For Question Answering Datasets, we use the same method as Orgad et al. (2025) to detect and utilize exact answer tokens. However, for True-False Datasets, we select key tokens as the exact answer tokens.

P(True). The prompt that we use for *P(True)* Kadavath et al. (2022) is as follows:

> Given the following question and answer, your objective is to determine if the answer correctly answers the question. You should give the probability that your think answer is correct.
> Question: [**Question**]
> Answer: [[**Answer**]]

LLM Configuration. For the selected LLMs, we download the model parameters directly from their official Hugging Face repositories. The generation process follows each model's official default configurations.

E 2D PCA Projection on Other Datasets

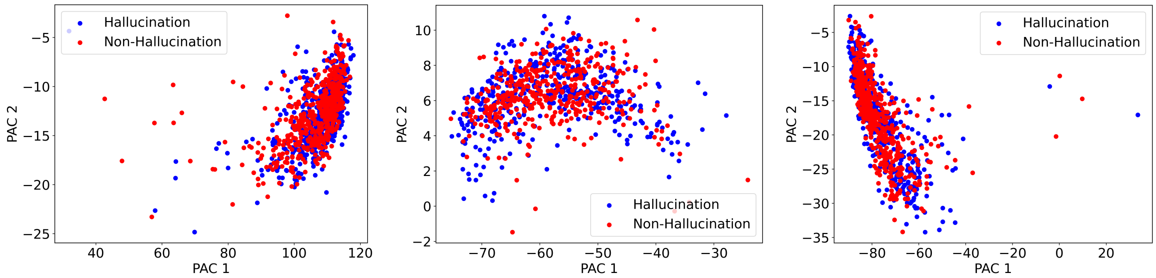
Figure 6 shows the 2D PCA projection of the last hidden layer's embedding for the final token on Company*, City*, Invention*. It reveals a significant overlap between correct and incorrect statements.

F Experiment of Using Middle Layers

We select the 16th, 20th, 24th, and 28th layers as representative intermediate layers and evaluate the performance of various methods based on the Vicuna-13B-v1.3 model on the Company* dataset, as shown in Table 3. Compared to the final layer, the results at the 20th and 24th layers show an overall improvement. For other layers, the results vary depending on the method. Therefore, specific intermediate layers may contain more information for whether a hallucination is occurring.

G Experiment of the Out-of-Domain Setting

To evaluate the generalization capability of the proposed method in an out-of-domain setting, we train the model on Company*, Fact*, and City*, and test its performance on Invention*, based on the Vicuna-13B-v1.3 model. The detailed experimental results are shown in Table 4. All methods exhibit a certain degree of performance degradation. Compared to SAPLMA, MIND, and Probe@Exact, the proposed ODEs, CDEs, and SDEs demonstrate relatively smaller declines, with reductions of less than 2%.



(a) Company*　(b) City*　(c) Invention*

Figure 6: 2D PCA projection of the last hidden layer's embedding for the final token on Company*, City*, Invention*. Blue and red dots represent hallucinations and non-hallucinations, respectively.

	SAPLMA	MIND	Probe@Exact	ODEs	CDEs	SDEs
16th	69.4	70.0	66.7	72.3	80.0	73.6
20th	70.5	70.3	**69.1**	73.9	**81.7**	73.8
24th	**71.0**	**71.3**	68.8	**74.0**	80.2	72.4
28th	68.1	68.9	67.5	72.4	78.9	**74.0**
Last	68.2	69.8	67.2	72.9	79.8	72.3

Table 3: AUC-ROC(%) for detection across the 16th, 20th, 24th, and 28th layers for different approaches.

Methods	SAPLMA	MIND	Probe@Exact	ODEs	CDEs	SDEs
AUC-ROC (%)	65.4	67.6	69.4	79.6	80.1	84.3

Table 4: AUC-ROC (%) for detection on Invention* using classifiers trained on Company*, Fact*, and City*.