

LLM Factoscope: Uncovering LLMs’ Factual Discernment through Inner States Analysis

Jinwen He^{1,2}, Yujia Gong^{1,2}, Kai Chen^{1,2}, Zijin Lin^{1,2}, Chengan Wei^{1,2} and Yue Zhao^{1,2}

¹SKLOIS, Institute of Information Engineering, Chinese Academy of Sciences

²School of Cyber Security, University of Chinese Academy of Sciences

{hejinwen, gongyujia, linzijin, weichengan, zhaoyue, chencai}@iie.ac.cn

Abstract

Large Language Models (LLMs) have revolutionized various domains with extensive knowledge and creative capabilities. However, a critical issue with LLMs is their tendency to produce outputs that diverge from factual reality. This phenomenon is particularly concerning in sensitive applications such as medical consultation and legal advice, where accuracy is paramount. In this paper, we introduce the LLM factoscope, a novel Siamese network-based model that leverages the inner states of LLMs for factual detection. Our investigation reveals distinguishable patterns in LLMs’ inner states when generating factual versus non-factual content. We demonstrate the LLM factoscope’s effectiveness across various architectures, achieving over 96% accuracy in factual detection. Our work opens a new avenue for utilizing LLMs’ inner states for factual detection and encourages further exploration into LLMs’ inner workings for enhanced reliability and transparency.

1 Introduction

Large Language Models (LLMs) have gained immense popularity, revolutionizing various domains with their remarkable creative capabilities and vast knowledge repositories. These models reshape fields like natural language processing [1], content generation [2], and more. However, despite their advanced abilities, a growing concern surrounds their propensity for “hallucination” — the generation of outputs that deviate from factual reality [3]. In critical applications like medical consultation [4], legal advice [5], and educational tutoring [6], factual LLM outputs are not just desirable but essential, as non-factual outputs from these models could potentially lead to detrimental consequences for users, affecting their health, legal standing, or educational understanding. Recognizing this, LLM-generated content’s factual detection has emerged as an area of paramount importance [7]. Current research predominantly relies on cross-referencing LLM outputs with external databases [3]. While effective, this approach necessitates extensive external knowledge bases and sophisticated cross-referencing algorithms, introducing more

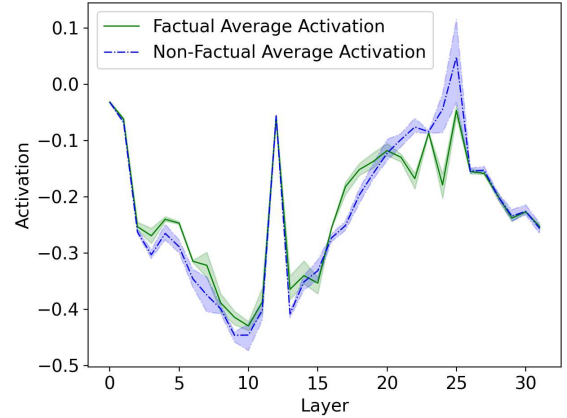


Figure 1: Average activation of factual and non-factual data.

complexity and dependency. This raises a compelling question: Could we possibly exclude external resources but only leverage the inner states of LLMs for factual detection?

Drawing inspiration from human lie detectors, which assess physiological changes like heart rate and micro-expressions to detect statement inconsistencies [8], our study proposes a similar approach for LLMs’ factual detection. We hypothesize that LLMs, having been exposed to a broad spectrum of world knowledge during training, might exhibit distinguishable patterns in their inner states when generating outputs that are either factual or non-factual. While LLMs might also learn from non-factual sources, they prefer choosing more factual sources for training [9]. Therefore, they may exhibit different inner states that imply factual or non-factual outputs. Our investigation observed distinct activation patterns in LLMs when they output factual versus non-factual content. Figure 1 displays the average activation values at each layer where we queried Llama-2-7B six times about different movies and their respective directors. The figure shows the average activation values as a line, with the shaded area representing the minimum to maximum activation values observed across these queries. Out of these six queries, three responses were factually correct, and three were incorrect. Notably, there is a discernable difference in activation values between layers 16 to 19 and 21 to 24. This phenomenon stems

from the differential areas within LLMs responsible for factual information and creative output, leading to varying internal state behaviors when producing factual versus non-factual content [10].

Based on our preliminary observations that LLMs exhibit distinct activation patterns when outputting factual versus non-factual content, we introduce the LLM factoscope, a Siamese network-based factual detection model. The LLM factoscope analyzes the inner states from LLMs, including *activation maps*, *final output ranks*, *top-k output indices*, and *top-k output probabilities*, each offering a unique perspective on the model’s internal decision-making process. Activation maps are utilized to understand information processing within the LLM, highlighting the neurons actively generating factual versus non-factual outputs. Concurrently, final output ranks indicate the evolving likelihood of the final output token across the layers, providing insights into the model’s shifting output preferences. Additionally, top-k output indices identify the most probable output tokens at each layer, reflecting the model’s decision-making priorities and its process of narrowing down choices. Complementing these, top-k output probabilities reveal the model’s confidence in its top choices at each layer, offering a window into its probabilistic reasoning. Together, these diverse inner states enable our LLM Factoscope model to effectively discern the factual accuracy of LLM outputs, leveraging the nuanced insights provided by each type of intermediate data in a cohesive, integrated manner.

LLM factoscope assesses the factuality of the model’s current output, providing a novel approach to fact-checking within LLMs. In our experiments, we empirically demonstrate the effectiveness of the LLM factoscope across various LLM architectures, including GPT2-XL-1.5b, Llama2-7b, Vicuna-7b, Stablelm-7b, Llama2-13b, and Vicuna-13b. The LLM factoscope achieves an accuracy rate exceeding 96% in factual detection. Additionally, we extensively examine the model’s generalization capabilities and conduct ablation studies to understand the impact of different sub-models and support set sizes on the LLM factoscope’s performance. Our work paves a new path for utilizing inner states from LLMs for factual detection, sparking further exploration and analysis of LLMs’ inner data for enhanced model understanding and reliability. Our contributions are as follows:

- We designed a pipeline for LLM factoscope, encompassing factual data collection, creation of a factual detection dataset, model architecture design, and detailed training and testing procedures. All the datasets and implementation will be released for further research and analysis.
- We empirically validated the effectiveness of LLM factoscope, explored its generalizability across various domains, and conducted thorough ablation experiments to understand the influence of different model components and parameters settings.

2 Background

2.1 Large Language Models

Large Language Models (LLMs), predominantly structured around the transformer decoder architecture [11]. These

models, typically comprising billions of parameters, are adept at capturing intricate language patterns [12]. A formalized view of their inner workings can be presented as follows: Consider an LLM defined as a function F mapping an input sequence $\mathbf{x} = (x_1, x_2, \dots, x_n)$ to an output sequence $\mathbf{y} = (y_1, y_2, \dots, y_m)$, where \mathbf{x} and \mathbf{y} consist of tokens from a pre-defined vocabulary \mathcal{V} . Each token x_i is first transformed into a high-dimensional space through an embedding layer, resulting in a sequence of embeddings $\mathbf{E} = \text{Embed}(\mathbf{x})$. The core of an LLM lies in its multiple layers of transformers, each comprising two main components: a self-attention module \mathcal{A} and a multilayer perceptron (MLP) module \mathcal{M} . For a given layer l , the hidden state $\mathbf{H}^{(l-1)}$ (with $\mathbf{H}^{(0)} = \mathbf{E}$) is first processed by the self-attention mechanism. The output of the attention layer, denoted as $\mathbf{A}^{(l)}$, is then passed through the MLP layer. The MLP, a series of fully connected layers, further processes this data to produce the output, denoted as $\mathbf{M}^{(l)}$. The process within each layer can be mathematically represented as: $\mathbf{A}^{(l)} = \mathcal{A}(\mathbf{H}^{(l-1)})$, $\mathbf{M}^{(l)} = \mathcal{M}(\mathbf{A}^{(l)}, \mathbf{H}^{(l-1)})$, $\mathbf{H}^{(l)} = \mathbf{H}^{(l-1)} + \mathbf{A}^{(l)} + \mathbf{M}^{(l)}$, where \mathcal{A} and \mathcal{M} encapsulate the operations within the attention and MLP, respectively. After the final layer L , the output $\mathbf{H}^{(L)}$ is typically passed through a linear layer followed by a softmax function to generate a probability distribution over the vocabulary \mathcal{V} for each token in the output sequence: $\mathbf{y} = \text{softmax}(W \cdot \mathbf{H}^{(L)} + b)$, where W and b are the weights and bias of the linear layer, respectively. Our method leverages inner states from the LLM, such as output from the hidden layer and MLP module, to detect whether the next output of the LLM is factual or not.

2.2 LLM Factual Detection

Fact-checking LLM outputs has become an increasingly critical task. Current approaches to mitigate LLM-generated inaccuracies include scrutinizing training datasets and cross-referencing external databases. Manual examination [13] of training datasets is labor-intensive, while external database referencing [14] [15] incurs additional computational costs and relies heavily on the effectiveness of cross-verification techniques. A recently proposed SAPLMA [16] investigates whether LLMs can discern the factuality of an input sentence. They use output from a single layer of LLM to train a fully connected neural network. Our method aims to distinguish each output as factual or non-factual, closely emulating the typical usage of LLMs. We leverage not just activation values from a single layer, but also the inter-layer changes in activations and hidden states within the LLM. This multi-dimensional analysis of the LLM’s inner data is akin to observing various physiological responses in a human lie detector [8]. By aggregating these intermediate states, our method provides a more effective, generalized, and explainable tool for analyzing the factual accuracy of the LLM’s output.

2.3 Siamese Network

Siamese Networks are designed to address few-shot learning challenges by discerning the similarities and differences between input pairs rather than conventional classification [17]. These networks consist of two identical sub-networks with shared weights, ensuring uniform processing of inputs. Their

primary aim is to learn a feature space where similar items are brought closer together, and dissimilar ones are pushed apart, using a contrastive loss function. This approach is particularly effective for few-shot learning, as it allows the network to learn robust representations of relationships between inputs, rather than direct classifications. Our LLM Factoscope model uses the Siamese Network framework to analyze inner states within LLMs. The training phase is guided by maximizing the similarity between similar (both from factual or both from non-factual) data points and minimizing it for dissimilar (one from factual and the other from non-factual) ones. During testing, the LLM Factoscope model uses a support set comprising a collection of labeled data for classification. When a test sample is introduced, the model computes its embedding and compares it with those in the support set. The classification of the test data is the same as the nearest sample in the support set, thus leveraging the learned similarities and differences from the training phase. This method ensures a more reliable and nuanced classification, especially in scenarios with limited or diverse data, by effectively utilizing the learned relationships within the model.

3 LLM Factoscope

This section outlines our method for developing an LLM factoscope. We begin with an overview of our pipeline. Subsequently, we delve into the preprocessing steps necessary to refine this data for effective model training. Then, we present the architecture of the LLM factoscope, elaborating on integrating various sub-models for processing different types of inner states. Lastly, we explain our model’s training and testing procedures.

3.1 Overview

We introduce a pipeline for creating an LLM factoscope that leverages the intermediate information of LLMs, as shown in Figure 2. Initially, we search structured data from the Kaggle repository in CSV format. Then, we extract entities and their corresponding targets that exhibit specific relations. For instance, in the context of a relation like movie-director, one such data point might be: the entity being the movie title ‘2001: A Space Odyssey’, and the target being its director, ‘Stanley Kubrick’. The entity, relation, and target are the framework we construct our dataset. This dataset is then deployed to probe LLMs to check whether their responses align with factual correctness, serving as labels for our factual detection dataset. Concurrently, we capture the LLMs’ inner states, which include the model’s inner representation of knowledge, and use them as features for our dataset. After collecting data, we apply a series of preprocessing steps, such as normalization and transformation. These steps are crucial as they standardize the data to a uniform scale and format, thereby significantly enhancing the LLM factoscope’s learning capabilities. The final step is to train a Siamese network-based model, designed to maximize the embedding similarity between similar class data (either both factual or both non-factual) and minimize the similarity between pairs of dissimilar class responses (one factual and one non-factual).

3.2 Factual Data Collection

We start our dataset collection by searching for fact-related CSV datasets on Kaggle [27], a platform chosen for its diverse and extensive datasets. The CSV format’s inherent structuring into entities, relationships, and targets makes it an ideal candidate for the automated generation of prompts and answers. Our dataset includes various categories, such as art [18] [19], sport [20] [21], literature [22], geography [23], history [24], science [25], and economics [26], ensuring comprehensive coverage of various factual aspects. Within each category, we have developed datasets encompassing multiple relational types—for instance, in the art category, relationships such as artwork-artist, movie-director, movie-writer, and movie-year are included. Leveraging GPT-4’s advanced capabilities [28] and meticulous manual adjustments, we have crafted clear and unambiguous prompts to ensure that LLMs can accurately comprehend the questions. To further enhance the dataset’s robustness and diversity, we have developed multiple synonymous question templates for each relation type. Table 1 provides an overview of the datasets. Each dataset entry consists of a prompt and a corresponding factual answer. We have made this dataset available for open-source contributions to facilitate further research.

3.3 Inner States Collection

After constructing our factual dataset, we present these prompts to the LLM. Beyond merely capturing the LLM’s direct responses, our focus extends to gathering inner states. This data, consisting of activation values and hidden states, is captured specifically for the last token of the entire prompt. The data is crucial for comprehensively understanding the model’s inner mechanisms, particularly how it processes information and makes decisions when generating responses. In the following, we detail the collection and significance of four key types of inner states: activation map, final output rank, top-k output index, and top-k output probability. Each type sheds light on different aspects of the LLM’s functioning, contributing to a deeper understanding of its response generation process.

Activation map: The activation map represents the activation values of the last token of the prompt when processed by the LLM. This map encapsulates the LLM’s inner representation of the knowledge pertinent to the prompt. As the LLM traverses through its layers, it retrieves information relevant to the prompt [29]. When the subsequent word aligns with the ground-truth answer, it indicates successful knowledge retrieval at the intermediate layers; otherwise, it suggests inadequate knowledge retrieval. These contrasting scenarios are expected to show distinct activation patterns, which we capture through the activation map.

Final output rank: This rank represents the position of the final output token in the probability distribution at each layer of the LLM. Specifically, we acquire the hidden states at each layer, apply the same vocabulary mapping as done in the final hidden layer through the linear layer, and thereby attain logits for each token in the vocabulary [11]. The rank is determined based on the descending order of logits for the final output token at each layer. The rank shows how the likelihood of

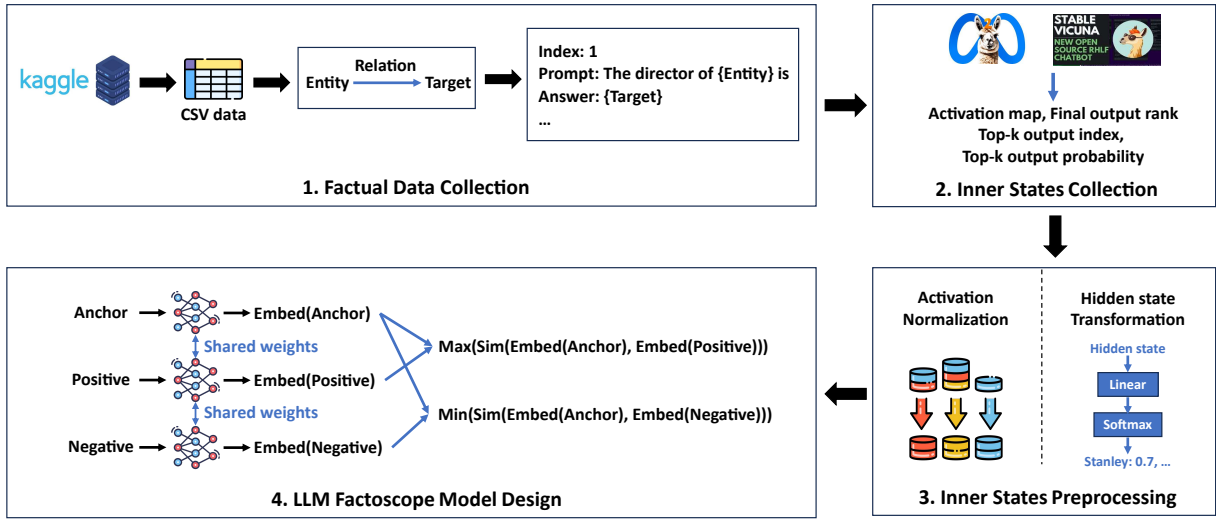


Figure 2: Pipeline of the LLM factoscope.

the final token changes across layers, reflecting the model’s evolving output preferences.

Top-k output index and probability: From the logits used in the final output ranking, we identify the top-k tokens with the highest logits in each layer. These tokens represent the model’s most likely outputs after processing the information at each layer. The relationships among these top-k tokens, both within and across layers, shed light on various cognitive aspects of the model’s processing. Applying a softmax function to the logits in each layer, we get the probability distribution of all tokens, subsequently identifying the top-k tokens with the highest probabilities. This data reflects the fluctuating probabilities of these tokens across layers, providing insights into the model’s probabilistic reasoning.

By closely examining the LLMs’ intermediate responses to factual prompts, we not only gain valuable insights into the inner dynamics of the models’ decision-making processes but also establish a foundation for more nuanced analysis and modeling of their behavior in discerning factual from non-factual outputs. Moreover, alongside these inner states, we record labels for factual detection. These labels, derived by evaluating whether the model’s first word following each prompt aligns with the factual answer, serve as a key indicator of the model’s accuracy in factual detection. A correct alignment is marked as positive, while a misalignment is categorized as negative.

3.4 Inner States Preprocessing

In this section, we introduce the preprocessing of inner states for factual detection using LLMs. This preprocessing involves normalization and transformation techniques to refine the data for effective integration into the training process. We detail the preprocessing methods applied to each category of inner states.

Normalization of activation map: We calculate the mean μ and standard deviation σ of the dataset. The activation map \mathbf{A} is then normalized using the formula: $\mathbf{A}_{\text{normalized}} =$

$(\mathbf{A} - \mu) / \sigma$. This normalization ensures a uniform scale for the activation values, enhancing their comparability and relevance in the model’s learning mechanisms.

Transformation of final output rank: The rank of the final output token undergoes a specific transformation to normalize the ranks to fall within the range of 0 to 1 and emphasize higher initial ranks (lower numerical values). Mathematically, the transformation of rank r can be represented as $r_{\text{transformed}} = 1 / [(1 - r) + 1 + 10^{-7}]$, $r \in (1, |V|)$, where $|V|$ is the size of LLM’s vocabulary. When the rank r is 1 (indicating the highest initial rank), the transformed rank $r_{\text{transformed}}$ becomes its maximum value, close to 1. Adding 10^{-7} in the denominator is a small constant to prevent division by zero.

Distance calculation for top-k output index: In processing the top-k output index, we measure the semantic proximity between token embeddings across adjacent layers. This is achieved by calculating the cosine similarity between the embeddings of tokens, providing insights into how the model’s perception of these tokens evolves across layers. The distance metric helps understand the semantic continuity or shifts within the model’s processing layers.

It is important to note that while most categories of inner states require preprocessing to standardize their scales or enhance their interpretability, the top-k output probability data does not undergo such preprocessing. This is because the Top-k output probabilities are inherently on a consistent scale, being probabilities that naturally range from 0 to 1. Hence, they are already in a format conducive to model training and analysis, requiring no additional normalization or transformation.

3.5 LLM Factoscope Model Design

After preparing the inner states datasets, we develop the LLM factoscope model, which is inspired by the principles of few-shot learning and Siamese networks. It is designed to effec-

Category	Example	Size
Art [18][19]	Prompt: The artist of the artwork Still Life with Flowers and a Watch is Answer: Abraham Mignon	67,302
Sport [20][21]	Prompt: The athlete Ole Jacob Bangstad represents the country of Answer: Norway	31,718
Literary [22]	Prompt: The book Twilight was written by Answer: Stephenie Meyer	54,301
Geography [23]	Prompt: The city Leipzig is located in the country of Answer: Germany	1,103
History [24]	Prompt: The birthplace country of the historical figure Albert Einstein is Answer: Germany	56,705
Science [25]	Prompt: The Nobel laureate Jacobus Henricus van 't Hoff is from Answer: Netherlands	8,971
Economics [26]	Prompt: Microsoft was started by Answer: Bill Gates	5,228
Multi [10]	Prompt: The mother tongue of Danielle Darrieux is Answer: French	21,918
Total		247,246

Table 1: Overview of the Factual Dataset

tively learn robust representations from limited data. This approach aims to distinguish between factual and non-factual content and demonstrates impressive generalization capabilities on similar but unseen data. Our model comprises four distinct sub-models, each dedicated to processing one of the key types of inner state data: activation map, top-k output index, top-k output probability, and final output rank.

For the activation map, top-k output index, and top-k output probability, we utilize Convolutional Neural Networks (CNNs) with the ResNet18 architecture [30]. The choice of ResNet18, with its convolutional and residual connections, is particularly advantageous for efficiently capturing the relationships between and within different layers of the LLM. These CNNs transform the inner states into embeddings $\mathbf{E}_{\text{activation}}$, $\mathbf{E}_{\text{top-k index}}$, and $\mathbf{E}_{\text{top-k prob}}$. Each embedding captures unique aspects of the LLM’s processing dynamics. As for the final output rank, a sequential data type, we use a Gated Recurrent Unit (GRU) network [31], reflecting the temporal evolution of the model’s output preferences across layers. This network yields an embedding \mathbf{E}_{rank} . The embeddings from these four sub-models are then integrated through a linear layer to form a comprehensive mixed representation, $\mathbf{E}_{\text{mixed}}$. This representation embodies the LLM’s holistic factual understanding. This combined embedding captures an integrated expression of the LLM’s factual understanding, representing spatial and temporal insights.

During training, our model utilizes the triplet margin loss [32], a metric integral to embedding learning in few-shot learning scenarios. This loss function is designed to maximize the similarity between instances of the same class while minimizing the similarity between instances of different classes. For a given training instance x , we feed it to the

combined model and get an embedding for its mixed representation, $\mathbf{E}_{\text{anchor}}$. Alongside, we select a positive example \mathbf{x}_{pos} from the same category as the anchor and a negative example \mathbf{x}_{neg} from a different category. Subsequently, we obtain their respective mixed expressions \mathbf{E}_{pos} and \mathbf{E}_{neg} . The triplet margin loss aims to ensure that the similarity between the anchor and the positive instance, $\text{Sim}(\mathbf{E}_{\text{anchor}}, \mathbf{E}_{\text{pos}})$, is larger than the similarity between the anchor and the negative instance, $\text{Sim}(\mathbf{E}_{\text{anchor}}, \mathbf{E}_{\text{neg}})$, by at least a margin α . This loss function is formally defined as:

$$L = \max(\text{Sim}(\mathbf{E}_{\text{anchor}}, \mathbf{E}_{\text{pos}}) - \text{Sim}(\mathbf{E}_{\text{anchor}}, \mathbf{E}_{\text{neg}}) + \alpha, 0),$$

where $\text{Sim}(\cdot, \cdot)$ is the chosen similarity metric, typically cosine similarity, and α is a critical hyperparameter. By fine-tuning α , we can enhance the model’s discriminative capability, ensuring that the similarity between the anchor and the positive instance is greater than that between the anchor and the negative instance by at least the margin α . The training process maximizes the loss, refining the model’s ability to accurately differentiate between factual and non-factual content.

In the testing phase, we establish a support set consisting of data samples and their corresponding targets, denoted as $\{S_1, \dots, S_n\}$ and $\{T_{\text{sup}_1}, \dots, T_{\text{sup}_n}\}$, respectively. These samples have not been used in the training process of the LLM factoscope model. They are crucial for the testing phase, as they provide a reference for comparing and classifying new, unseen test data. Each sample in the support set is processed through the LLM factoscope model to generate mixed representations, represented by $\{\mathbf{E}_{\text{sup}_1}, \dots, \mathbf{E}_{\text{sup}_n}\}$. The mixed representations are outputs of the LLM factoscope model. The test data’s mixed representation, \mathbf{E}_{test} , is then compared against these support set representations. The

classification of the test data is determined by identifying the closest support set embedding to \mathbf{E}_{test} . The target of the test data is the target of this nearest support set data:

$$T_{\text{test}} = T_{\text{sup}_{i^*}} \quad \text{where} \quad i^* = \underset{i}{\operatorname{argmax}} \operatorname{Sim}(\mathbf{E}_{\text{test}}, \mathbf{E}_{\text{sup}_i}).$$

Here, the index i^* identifies the support set data that is closest to the test data, and $T_{\text{sup}_{i^*}}$ is the target associated with this closest support set data. This approach ensures accurate and reliable classification of the test data by leveraging the similarities within the representations of the support set. Details on the model’s architectural parameters and training parameters are explored in Section 4.5, where their impacts on model performance are thoroughly investigated.

4 Evaluation

4.1 Experimental Setup

Dataset. We employ various factual datasets encompassing multiple domains such as art, sports, literature, geography, history, science, and economics. Each domain includes several relations, like the artist of a particular artwork and the founder of a company. The factual datasets comprise 247,246 data points, facilitating a comprehensive evaluation of the model’s ability to discern factual information. Then, we record the inner states of the LLM as it processes factual and non-factual statements, including activation values, final output rank, top-k output index, and top-k output probability. The label assigned to each data point indicates whether the corresponding model output is factual or non-factual. To ensure dataset balance, we randomly select an equal number of factual and non-factual data points for each factual relationship. Furthermore, the features of this dataset are pre-processed to ensure they are standardized and optimized for model learning.

Models. Our experiments are conducted on several popular LLMs, each with distinctive architectures and characteristics. These models include GPT2-XL [33], LLaMA-2-7B [9], LLaMA-2-13B, Vicuna-7B-v1.5 [34], Vicuna-13B-v1.5, and StableLM 7B [35]. These models allow us to comprehensively evaluate the effectiveness of our fact detection methodology across various LLM architectures and configurations. The LLM factoscope model comprises several sub-models, each tailored to handle a specific type of inner state. This includes a ResNet18 model for processing activation values, a GRU network for final output rankings, and two additional ResNet18 models for handling word embedding distances and top-k output probabilities. We set top-k to top-10. The output of each sub-model is an embedding of dimension 24. These embeddings from each sub-model are concatenated, resulting in a combined embedding of dimension 96. This combined embedding is then fed into a fully connected layer, which reduces the dimensionality to 64, ensuring a compact yet informative representation. The final embedding undergoes ReLU activation and L2 normalization, providing a normalized feature vector for each input. During testing, the size of the support set is set to 100.

Experimental Environment. Our experimental setup was hosted on a server with 32 Intel Xeon Silver 4314 CPUs at

2.40 GHz, 386 GB of RAM, and four NVIDIA A100 Tensor Core GPUs, providing substantial computational capacity and facilitating efficient processing for large-scale computations. The entire suite of experiments was conducted on an Ubuntu 20.04 LTS operating system.

4.2 Effectiveness

In evaluating the performance of LLM Factoscope, we considered various LLMs, including GPT2-XL-1.5B, Llama2-7B, Vicuna-7B, Stablelm-7B, Llama2-13B, and Vicuna-13B. To establish a comparative baseline, we use activation values from specific layers, particularly those in the middle to later stages of the LLMs, as input features to train a fully connected neural network for factual detection, which aligns with our observations in Figure 1. For GPT2-XL-1.5B, the model is based on the activation values from the 31st layer. In the case of Llama2-7B and Vicuna-7B, the 23rd layer’s activation values are used. For Stablelm-7B, the baseline model relied on the 12th layer, while Llama2-13B and Vicuna-13B utilize the activation values from their 32nd layers.

As shown in Table 2, our LLM Factoscope consistently maintains high accuracy levels, ranging between 96.1% and 98.3%, across different LLM architectures. In contrast, the accuracy of the baseline fluctuates between 78.5% and 88.8%. This variation suggests that as LLMs increase in parameter size, the regions responsible for different types of factual knowledge might differ, or multiple layers could be involved in representing a single type of factual knowledge. Consequently, the baseline, which relies solely on activation values from a single layer, demonstrated unstable performance. Based on the analysis, we believe that the superior performance of LLM Factoscope is attributable to its consideration of various inner state changes across layers. By integrating this multi-dimensional analysis of inner states within LLMs, LLM Factoscope effectively discerns factual from non-factual outputs, offering a more robust and reliable approach to factual detection. This method’s success not only highlights the significance of inner activations and activation values in understanding LLM outputs but also paves the way for future explorations into the intricate workings of LLMs, particularly in the realm of natural language processing applications.

4.3 Generalization

It is well-established in neural network research that the effectiveness of a model largely hinges on the similarity between training and testing distributions [36]. Thus, our model’s performance may vary across different distributions. We adopt a leave-one-out approach for our generalization assessment. Specifically, we remove one relation dataset, train the model on the remaining datasets, and then test it on the omitted dataset. We selected three relations for assessing generalization, including Book-Author, Pantheon-Country, and Athlete-Country, as these relations form sizable datasets across all LLMs. Our empirical findings suggest that different LLMs exhibit varying generalization capabilities across different relations, as shown in Table 3. In the “Book-Author” relation, our method achieved a notable 97.7% accuracy with Llama2-7B and 69.0% with Stablelm-7B. This variation is likely due

Table 2: Effectiveness results of LLM factoscope across different LLM architectures.

Method	GPT2-XL-1.5B	Llama2-7B	Vicuna-7B	Stablelm-7B	Llama2-13B	Vicuna-13B
Ours	0.961	0.967	0.982	0.983	0.983	0.974
Baseline	0.880	0.888	0.831	0.817	0.882	0.785

Table 3: Generalization Performance of different LLM architectures. Abbreviations: BL - Baseline, BA - Book-Author, PC - Pantheon-Country, AC - Athlete-Country.

Data	GPT2-XL-1.5B		Llama2-7B		Vicuna-7B		Stablelm-7B		Llama2-13B		Vicuna-13B	
	Ours	BL	Ours	BL	Ours	BL	Ours	BL	Ours	BL	Ours	BL
BA	0.712	0.800	0.977	0.701	0.971	0.757	0.690	0.333	0.904	0.854	0.895	0.814
PC	0.871	0.879	0.972	0.946	0.790	0.516	0.635	0.610	0.938	0.849	0.913	0.698
AC	0.979	0.780	0.703	0.693	0.770	0.716	0.694	0.756	0.778	0.686	0.807	0.703
Average	0.854	0.818	0.884	0.780	0.844	0.663	0.673	0.566	0.873	0.763	0.872	0.738

to each LLM’s unique handling of different types of factual knowledge. Our method outperforms the baseline in most cases, with its average performance consistently outperforming the baseline, indicating its superior generalization ability.

We believe that LLM factoscope uses various intermediate states beyond activation values—specifically, final output rank, top-k output index, and top-k output probability—significantly bolsters its generalization capabilities. Stablelm-7B exhibits the weakest generalization performance among all the LLMs tested. This aligns with its relatively lower scores on LLM leaderboards [37]. We hypothesize that this could be attributed to its less effective learning of factual versus non-factual content it was trained on. While our LLM factoscope demonstrates certain levels of generalization, we recommend ensuring similarity in the distribution of testing and training data when using this tool. For instance, an LLM who served as a historical knowledge assistant should align the LLM factoscope’s training with relevant historical data to ensure its effectiveness.

4.4 Interpretability

We delve into the interpretability of the LLM factoscope, aiming to analyze the contribution of these features in discerning the factualness of LLM outputs. Specifically, we use the Integrated Gradient [38] to analyze the contribution of activation maps, final output ranks, top-k output indices, and top-k output probabilities. Integrated Gradient is particularly chosen for its higher faithfulness in interpretability assessments [39]. Our analysis reveals that the most influential features are mainly in the middle to later layers of the LLMs, consistently observed across all four data types. To provide a clearer visualization of this pattern, we present a typical example in Figure 3. In the figure, red indicates a positive contribution, while blue signifies a negative contribution, with deeper colors representing higher average importance. Due to the high dimensionality of activation data, we compute and display the average importance of features at each layer. The majority of positive contributions emerge after the 15th layer. This finding aligns with our observations that the model initially filters semantically coherent candidate outputs in the

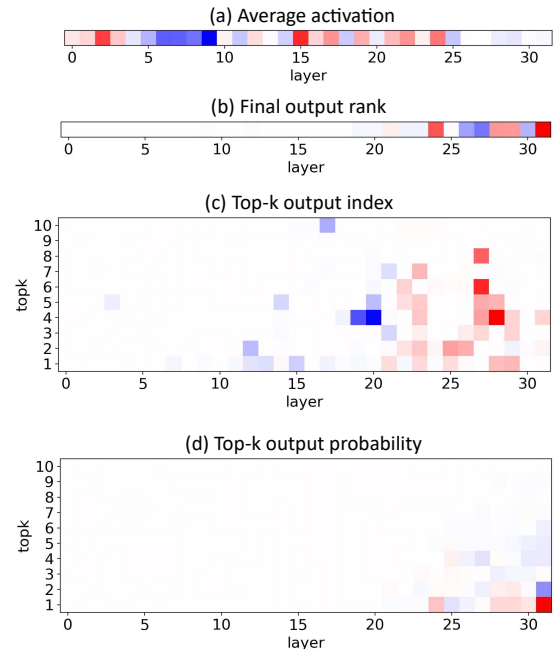


Figure 3: Interpretations of different features’ contribution.

earlier layers, and then progressively focuses on candidates relevant to the given prompt task in deeper layers.

4.5 Ablation Study

Contribution of each sub-model. We evaluate the contribution of each sub-model by incrementally adding them to the factual detection model on GPT2-XL-1.5b. As depicted in Table 4, we notice a slight but consistent improvement in Acc with the addition of more sub-models. This indicates that each sub-model brings a unique dimension to the model’s capabilities, enhancing its overall performance. We employ the “leave one out” training approach as in Section 4.3 to assess the contribution of sub-models to generalization. The results in Table 4 demonstrate enhanced generalization as more sub-

Table 4: Effectiveness and Generalization Results for Sub-Model Variations in Factual Detection Model. Abbreviations: BA - Book-Author, PC - Pantheon-Country, AC - Athlete-Country.

Data	All	BA	PC	AC
Act	0.945	0.665	0.842	0.916
Rank	0.955	0.659	0.873	0.929
Emb	0.960	0.652	0.867	0.936
Prob	0.967	0.712	0.871	0.979

Table 5: Effectiveness results for varying top-k.

Top-k	2	4	6	8	10
Acc	0.923	0.904	0.948	0.935	0.954

models are integrated. This improvement is particularly evident in the final model, which shows a significant increase in Acc across various datasets compared to the model with only one sub-model. For instance, in the “book-author” category, the accuracy improves from 66.5% to 71.2%, and in the “athlete-country” category, it jumps from 91.6% to 97.9%. One possible explanation for this enhanced generalization is the varied dependencies of the sub-models on the relational data type. The first sub-model (activation map) is heavily reliant on the type of relationship data, whereas the subsequent sub-models (final output rank, top-k output index, and top-k output probability) are more independent of the relationship data type. Therefore, they exhibit stronger generalization capabilities. The design of multiple sub-models captures both the relational data type-dependent and independent features, achieving high effectiveness and improved generalization.

Effects of different top-k. The top-k affects the top-k output index and top-k output probability. The previous experiments set the top-k to 10 unless otherwise stated. Now, we will evaluate the effect of choosing different values for top-k on the performance of the factual detection model. We set the top-k to 2, 4, 6, 8, 10. The results of the experiment are shown in Table 5. The lowest performance is 90.4% when top-k is 4, and the highest performance is 95.4% when top-k is 10. The difference between the two is only 5%, which indicates that top-k does not have much effect on the model performance, and that there is no purely positive correlation between top-k and the factual detection model’s accuracy.

Effects of different support set size. We also try different support set sizes from 50 to 250, observing their impact on the performance of the factual detection model. This evaluation was conducted on the Llama2-7b. The results, as presented in Table 6, demonstrate that the change in support set size does not significantly impact the model’s performance across most metrics. However, a notable trend is observed with the increase of support set size to 200 or 250. There is a slight increase (about 2%) in Acc and a corresponding rise in the FNR. The rise in FNR could be attributed to the richer variety and broader distribution of non-factual words compared to factual ones. When the support set is expanded to a certain extent, the coverage of non-factual tokens increases dispropor-

Table 6: Effectiveness results for varying support set size.

Support Size	Acc	TPR	FPR	TNR	FNR
50	0.938	0.953	0.079	0.921	0.047
100	0.932	0.942	0.080	0.920	0.058
150	0.935	0.958	0.092	0.908	0.042
200	0.912	0.908	0.082	0.922	0.092
250	0.914	0.906	0.078	0.922	0.094

Table 7: Effectiveness results of different sub-model architectures.

Architecture	Act-fc	Rank-rnn	Emb-fc	Prob-fc
Acc	0.945	0.915	0.736	0.943

tionately more than that of factual tokens. This imbalance possibly leads to a scenario where the model is more prone to misclassify data into the non-factual category.

Effects of different sub-models’ architectures. We use different sub-model architectures and assess the performance of the factual detection model. We use fully connected layers to replace the ResNet18 and RNN to replace the GRU network. As shown in Table 7, the original architecture achieves an impressive Acc of 95.4%. This demonstrates the effectiveness of the original design in capturing and processing factual content accurately. When we replace parts of the architecture with fully connected layers (act-fc, prob-fc) and RNNs (rank-rnn), we notice a slight decline in performance. Specifically, the act-fc show a decrease in Acc to 94.5%, while the rank-rnn drops the Acc to 91.5%. These changes do not drastically alter the model’s ability to factual detection. In contrast, the emb-fc architecture, where we replace the ResNet18 with fully connected layers, result in a significant performance dip. This architecture substantially reduces all metrics, with Acc falling to 73.6%. Such a drastic drop highlights the pivotal role of ResNet18 in effectively capturing the LLM’s top-k output index. These results underscore the critical importance of selecting the appropriate sub-model architectures for factual detection models. While the model demonstrates resilience to certain architectural changes, some alterations can substantially impact its performance.

5 Conclusion

We introduced the LLM factoscope, a pioneering approach that utilizes the inner states of Large Language Models for factual detection. Through extensive experiments across various LLM architectures, the LLM Polygraph consistently demonstrated high factual detection accuracy, surpassing 96% in most cases. This robust performance underscores the model’s efficacy in discerning factual from non-factual content. Our research not only provides a novel method for factual verification within LLMs but also opens new avenues for future explorations into the untapped potential of LLMs’ inner states. By paving the way for enhanced model understanding and reliability, the LLM Polygraph sets a foundation for more transparent, accountable, and trustworthy use of LLMs in critical applications.

References

- [1] Diksha Khurana, Aditya Koli, Kiran Khatter, and Sukhdev Singh. Natural language processing: State of the art, current trends and challenges. Multimedia Tools and Applications, 82, 07 2022.
- [2] Arkadeep Acharya, Brijraj Singh, and Naoyuki Onoe. Llm based generation of item-description for recommendation system. In Proceedings of the 17th ACM Conference on Recommender Systems, RecSys '23, page 1204–1207, New York, NY, USA, 2023. Association for Computing Machinery.
- [3] Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, Longyue Wang, Anh Luu, Wei Bi, Freda Shi, and Shuming Shi. Siren’s song in the ai ocean: A survey on hallucination in large language models, 09 2023.
- [4] Hongbo Zhang, Junying Chen, Feng Jiang, Fei Yu, Zhihong Chen, Jianquan Li, Guiming Chen, Xiangbo Wu, Zhiyi Zhang, Qingying Xiao, Xiang Wan, Benyou Wang, and Haizhou Li. Huatuoogpt, towards tam- ing language models to be a doctor. arXiv preprint arXiv:2305.15075, 2023.
- [5] Jiayi Cui, Zongjian Li, Yang Yan, Bohua Chen, and Li Yuan. Chatlaw: Open-source legal large language model with integrated external knowledge bases, 2023.
- [6] Shriyash Upadhyay, Etan Ginsberg, and Chris Callison-Burch. Improving mathematics tutoring with a code scratchpad. In Ekaterina Kochmar, Jill Burstein, Andrea Horbach, Ronja Laarmann-Quante, Nitin Mad- nani, Anaïs Tack, Victoria Yaneva, Zheng Yuan, and Torsten Zesch, editors, Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023), pages 20–28, Toronto, Canada, July 2023. Association for Computa- tional Linguistics.
- [7] Edoardo Mosca, Mohamed Hesham Ibrahim Abdalla, Paolo Basso, Margherita Musumeci, and Georg Groh. Distinguishing fact from fiction: A benchmark dataset for identifying machine-generated scientific papers in the LLM era. In Anaelia Ovalle, Kai-Wei Chang, Ninareh Mehrabi, Yada Pruksachatkun, Aram Galystan, Jwala Dhamala, Apurv Verma, Trista Cao, Anoop Ku- mar, and Rahul Gupta, editors, Proceedings of the 3rd Workshop on Trustworthy Natural Language Processing (TrustNLP 2023), pages 190–207, Toronto, Canada, July 2023. Association for Computational Linguistics.
- [8] Don Grubin and Lars Madsen. Lie detection and the polygraph: A historical review. The Journal of Forensic Psychiatry & Psychology, 16(2):357–369, 2005.
- [9] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and ef- ficient foundation language models. arXiv preprint arXiv:2302.13971, 2023.
- [10] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. Advances in Neural Information Processing Systems, 36, 2022.
- [11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc., 2017.
- [12] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. A survey of large language models. arXiv preprint arXiv:2303.18223, 2023.
- [13] Guilherme Penedo, Quentin Malartic, Daniel Hess- low, Ruxandra-Aimée Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Al- mazrouei, and Julien Launay. The refinedweb dataset for falcon llm: Outperforming curated corpora with web data, and web data only. ArXiv, abs/2306.01116, 2023.
- [14] Ruiyang Ren, Yuhao Wang, Yingqi Qu, Wayne Xin Zhao, J. Liu, Hao Tian, Huaqin Wu, Ji rong Wen, and Haifeng Wang. Investigating the factual knowledge boundary of large language models with retrieval aug- mentation. ArXiv, abs/2307.11019, 2023.
- [15] Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ramakanth Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, Edouard Grave, Yann LeCun, and Thomas Scialom. Augmented language models: a survey. ArXiv, abs/2302.07842, 2023.
- [16] Amos Azaria and Tom Mitchell. The internal state of an llm knows when its lying. arXiv preprint arXiv:2304.13734, 2023.
- [17] Gregory R. Koch. Siamese neural networks for one-shot image recognition. 2015.
- [18] Maxwell. Famous paintings dataset. <https://www.kaggle.com/datasets/mexwell/famous-paintings>.
- [19] Daniel Grijalvas. Movies dataset. <https://www.kaggle.com/datasets/danielgrijalvas/movies>.
- [20] Olympic games dataset. <https://www.kaggle.com/datasets/the-guardian/olympic-games>.
- [21] The top paying sports teams and top paid athletes dataset. <https://www.kaggle.com/datasets/prashant808/the-top-paying-sports-teams-and-top-paid-athletes>.
- [22] Goodreads best books dataset. <https://www.kaggle.com/datasets/meetnaren/goodreads-best-books>.

- [23] Wikidata query service. <https://query.wikidata.org/>.
- [24] Pantheon project dataset. <https://www.kaggle.com/datasets/mit/pantheon-project>.
- [25] Nobel laureates dataset. <https://www.kaggle.com/datasets/nobelfoundation/nobel-laureates>.
- [26] Billionaires statistics dataset. <https://www.kaggle.com/datasets/nelgiriyeewithana/billionaires-statistics-dataset>.
- [27] Kaggle: Your home for data science. <https://www.kaggle.com>.
- [28] OpenAI. Gpt-4, 2023. <https://openai.com/gpt-4>.
- [29] Kevin Meng, Arnab Sen Sharma, Alex J. Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. In The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023. OpenReview.net, 2023.
- [30] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 770–778, 2016.
- [31] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics.
- [32] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2015.
- [33] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.
- [34] 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. Vicuna: An open-source chatbot impressing gpt-4 with 90
- [35] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.
- [36] Jingkan Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. Generalized out-of-distribution detection: A survey. arXiv preprint arXiv:2110.11334, 2021.
- [37] Edward Beeching, Clémentine Fourrier, Nathan Habib, Sheon Han, Nathan Lambert, Nazneen Rajani, Omar Sanseviero, Lewis Tunstall, and Thomas Wolf. Open llm leaderboard. https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard, 2023.
- [38] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In International Conference on Machine Learning, 2017.
- [39] Jinwen He, Kai Chen, Guozhu Meng, Jiangshan Zhang, and Congyi Li. Good-looking but lacking faithfulness: Understanding local explanation methods through trend-based testing. In Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS ’23, page 431–445, New York, NY, USA, 2023. Association for Computing Machinery.