Hallucination Detection in Foundation Models for Decision-Making: A Flexible Definition and Review of the State of the Art

NEELOY CHAKRABORTY, University of Illinois Urbana-Champaign, Urbana, United States
MELKIOR ORNIK, University of Illinois Urbana-Champaign, Urbana, United States
KATHERINE DRIGGS-CAMPBELL, University of Illinois Urbana-Champaign, Urbana, United States

Autonomous systems are soon to be ubiquitous, spanning manufacturing, agriculture, healthcare, entertainment, and other industries. Most of these systems are developed with modular sub-components for decision-making, planning, and control that may be hand-engineered or learning-based. While these approaches perform well under the situations they were specifically designed for, they can perform especially poorly in out-of-distribution scenarios that will undoubtedly arise at test-time. The rise of foundation models trained on multiple tasks with impressively large datasets has led researchers to believe that these models may provide "common sense" reasoning that existing planners are missing, bridging the gap between algorithm development and deployment. While researchers have shown promising results in deploying foundation models to decision-making tasks, these models are known to hallucinate and generate decisions that may sound reasonable but are in fact poor. We argue there is a need to step back and simultaneously design systems that can quantify the certainty of a model's decision and detect when it may be hallucinating. In this work, we discuss the current use cases of foundation models for decision-making tasks, provide a general definition for hallucinations with examples, discuss existing approaches to hallucination detection and mitigation with a focus on decision problems, present guidelines, and explore areas for further research in this exciting field.

CCS Concepts: • **Computing methodologies** → **Artificial intelligence**; **Machine learning**; **Natural language generation**; **Machine learning approaches;**

Additional Key Words and Phrases: Foundation models, decision-making, hallucination detection and mitigation, survey

ACM Reference Format:
Neeloy Chakraborty, Melkior Ornik, and Katherine Driggs-Campbell. 2025. Hallucination Detection in Foundation Models for Decision-Making: A Flexible Definition and Review of the State of the Art. *ACM Comput. Surv.* 57, 7, Article 188 (March 2025), 35 pages. https://doi.org/10.1145/3716846

This work is supported by the Office of Naval Research under Grant No. N00014-23-1-2651.
Authors' Contact Information: Neeloy Chakraborty, University of Illinois Urbana-Champaign, Urbana, Illinois, United States; e-mail: neeloyc2@illinois.edu; Melkior Ornik, University of Illinois Urbana-Champaign, Urbana, Illinois, United States; e-mail: mornik@illinois.edu; Katherine Driggs-Campbell, University of Illinois Urbana-Champaign, Urbana, Illinois, United States; e-mail: krdc@illinois.edu.

ACM 0360-0300/2025/03-ART188
https://doi.org/10.1145/3716846

1 Introduction

A great deal of progress has been made in the past decade and a half with regards to the efficacy and efficiency of models for perception, decision-making, planning, and control [54, 123]. Broadly speaking, approaches to these problems fall under one of two umbrellas: hand-engineered model-based systems and data-driven learning-based models [35]. With some deployment scenario in mind, developers may hand-engineer rules [42] or tune a controller [7] to be tested, or in the case of learning-based models, collect training data and craft some reward function to fit a model to an objective, given said data [45]. In practice, these methods work particularly well in the scenarios that they were specifically designed and trained for, but may produce undesirable results in previously unseen out-of-distribution cases [139]. Designers may choose to add more rules, re-tune their controller, fine-tune their model to a more representative dataset, fix the reward function to handle edge cases, or even add a detector (which may itself be rule-based or data-driven) at test-time to identify out-of-distribution scenarios before calling on the decision-maker [11, 114, 121]. However, even with these changes, there will always be other situations that designers had not previously considered that will come about during deployment, leading to sub-optimal performance or critical failures. Furthermore, the modifications made to the model may have unforeseen effects at test-time, such as undesired conflicting rules [33] or catastrophic forgetting of earlier learned skills [60].

Informally, classical methods and data-driven approaches lack some form of *common sense* that humans use to adapt in unfamiliar circumstances [38]. More recently, there has been work towards developing multi-modal large language models, which take inputs in the form of images, videos, audio, and text, to tackle more complex language understanding and reasoning tasks [5, 6]. These models are developed by collecting and cleaning an enormous natural language dataset, pre-training to reconstruct sentences on said dataset, fine-tuning on specific tasks (e.g., question-answering), and applying human-in-the-loop reinforcement learning to produce more reasonable responses [1]. Even though these models are another form of data-driven learning that attempt to maximize the likelihood of generated text conditioned on a given context, researchers have shown that they have the ability to generalize to tasks they have not been trained on and reason about their decisions. Many researchers are specifically exploring the use of **large (visual) language models (L(V)LMs)** to fill the knowledge gap found in earlier works [25]. As such, these *foundation* models are being tested in tasks such as simulated decision-making [51] and real-world robotics [155] to take the place of perception, planning, and control modules. Even so, foundation models are not without their limitations. Specifically, these models have a tendency to *hallucinate*, i.e., generate decisions or reasoning that sound plausible, but are in fact inaccurate or would result in undesired effects in the world [26, 56, 109]. This phenomenon has led to the beginning of a new research direction that attempts to detect when L(V)LMs hallucinate to produce more trustworthy and reliable systems. Before these large black-box systems are applied in safety-critical situations, there need to be methods to detect and mitigate hallucinations. Thus, this survey collects and discusses current hallucination mitigation techniques for foundation models in decision-making tasks and presents potential research directions.

Existing surveys particularly focus on presenting methods for hallucination detection and mitigation in **question-answering (QA)** [56, 109, 149, 160] or object detection tasks [73]. For example, Ji et al. [56] summarize metrics and hallucination detection and mitigation methods for abstractive summarization, dialogue generation, generative question-answering, data-to-text generation, and machine translation for natural language generation models in general. Ye et al. [149] and Rawte et al. [109] take this work a step further by classifying hallucination mitigation strategies for foundation models in particular, in generative text and multi-modal settings, respectively. Zhang et al. [160] similarly tackle hallucination detection for language foundation models but limit their

definition of hallucinations to conflicts between generations and inputs, contexts, and facts. The authors also discuss other possible problems in generated outputs, such as ambiguity and bias. In the domain of image captioning, Li et al. [73] provide examples of hallucinations and present a new metric to evaluate object hallucinations. More recent surveys, like the ones from Bai et al. [5] and Liu et al. [80], dive deeper into hallucination mitigation methods for multi-modal foundation models applied to image-captioning tasks but lack a general definition for hallucinations that encompasses decision-making applications. The extensive survey from Huang et al. [49] primarily focuses on methods for evaluating the trustworthiness of language models from the fronts of factuality and faithfulness. Their taxonomy also briefly touches on hallucinations in object detection using LVLMs but, like other surveys, ignores broader decision-making deployments. There are also other works that provide examples of current use cases of L(V)LMs in autonomous vehicles [146] and robotics [155, 156]. Wang et al. [133] perform a deep analysis of the trustworthiness of a variety of foundation models, and Chen and Shu [13] provide a taxonomy of hallucinations within LLMs, but both exclude applications to general decision problems. To the best of our knowledge, we are the first to propose a general definition of hallucinations that can be flexibly tuned to any particular deployment setting, including commonly found applications to QA or information retrieval and more recent developments in planning or control. Furthermore, there is no existing work that summarizes state-of-the-art methods for hallucination detection and mitigation approaches within decision-making and planning tasks. Additionally, to the best of our knowledge, ours is the first review to provide guidelines for choosing and designing hallucination intervention algorithms across different application areas.

In the remainder of this work, we discuss the current uses of foundation models for decision-making tasks in Section 2, define and provide examples of hallucinations in Section 3, identify current detection methods in Section 4, present guidelines in Section 5, and explore research directions in Section 6.

2 Foundation Models Making Decisions

Originally coined by Bommasani et al. [6], the term *foundation models* refers to models that are "trained on broad data at scale such that they can be adapted to a wide range of downstream tasks." This approach is in contrast to works that design and train models on a smaller subset of data for the purpose of being deployed to a specific task [144]. The key difference is that foundation models undergo a pre-training procedure on a large-scale dataset containing information from a variety of possible deployment fields, through which they are expected to learn more general features and correspondences that may be useful at test-time on a broader set of tasks [162, 163]. Examples of existing pre-trained foundation models span language [8, 28, 128], vision [10, 63, 94], and multi-modal [1, 105] inputs. In this section, we give a brief overview of existing use cases for foundation models in robotics and autonomous vehicles, and we provide a discussion of other decision-making systems in Appendix A.2. We also succinctly point out hallucinations found in these works and leave a lengthier discussion in Section 3.2. Readers should refer to works from Cui et al. [25], Yang et al. [146], Zeng et al. [155], and Zhang et al. [156] for a deeper review of application areas.

2.1 Autonomous Driving

For the autonomous vehicle domain, researchers have formulated the use of language foundation models as a fine-tuning and prompt engineering problem [138, 139]. An external sub-system is usually designed with (1) a perception module to process signals from raw sensors, (2) a memory bank of prior important experiences and its corresponding similarity function to find alike scenarios, and (3) a prompt generator to convert current sensor data and relevant memories into natural language that can be input to the foundation model. Currently, works either fine-tune

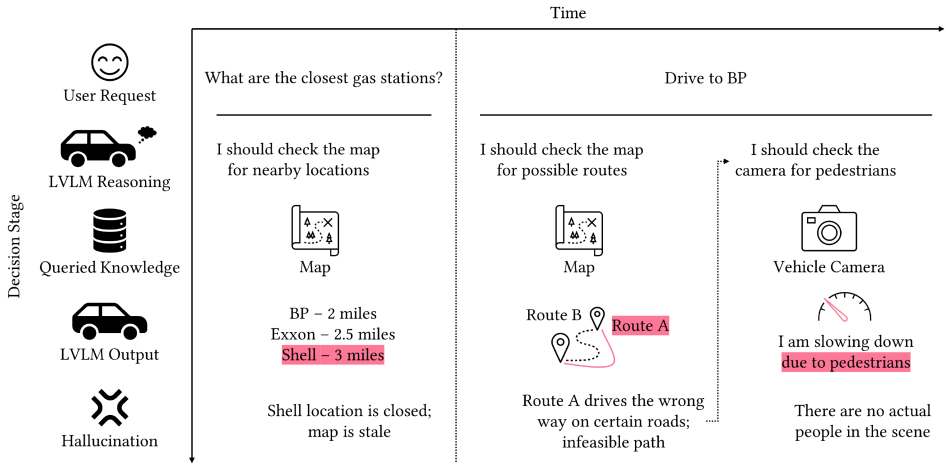

Fig. 1. Example deployment of an LVLM foundation model in an autonomous driving setting. Hallucinations (pink) may arise at any point in the decision-making pipeline, including information retrieval, planning, perception, and control. In this example, the LVLM correctly queries the map for possible destinations of gas stations but lists a location that is no longer open. Then, when navigating to one of the locations, the model predicts a path on the map that goes in the wrong direction of traffic flow. Finally, when applied to perception tasks for detecting possible pedestrians in front of the vehicle, the model hallucinates nearby people, causing an improper control action.

LLMs with a few examples, or directly apply the model in a zero-shot manner, on a QA task with driving-related questions. By framing the task in a QA form, researchers have been able to provide context to the L(V)LM to probe for high-level natural language decisions [138, 139], path planning [86, 120], vehicle tracking and trajectory prediction [61, 141], descriptions of the surroundings of the vehicle [15, 143], and low-level control [81]. Figure 1 is an example of how a foundation model may be used in an autonomous driving setting, with possible hallucinations in deployment.

High-level Decisions. Wen et al. [138] propose DiLu, a framework consisting of reasoning, reflection, and memory modules that support an LLM in producing high-level decisions for autonomous driving, and they test their method within a driving simulator environment. Specifically, the reasoning module views the current observation of the vehicle, queries the memory module for any similar situations that were encountered in the past, and converts the experience into a prompt, which is input to the LLM. The prompt is formatted such that it elicits chain-of-thought reasoning [137] from the LLM, which is shown to improve the accuracy of the model. The generated text output by the LLM is summarized by the reflection module and is used to update the memory bank of experiences. A separate decision decoder model converts the summary into a discrete high-level decision (e.g., idle, turn right, accelerate) to take in the simulator. The same authors have also experimented with prompting the LVLM model, GPT-4V [1], in a zero-shot manner to describe the surroundings of the vehicle, take high-level decisions, and explain why it believes it would be a good action to take [139]. They find that GPT-4V is capable of identifying safety-critical scenarios and suggests driving more conservatively in those situations. However, like other existing vision models [125], it has difficulty in detecting traffic-light states. We discuss other hallucination examples in traffic scenarios in Section 3.2.

Path Planning. Agent-Driver from Mao et al. [86] utilizes a tool library of functions that communicate with neural modules that are responsible for object detection, trajectory prediction,

occupancy estimation, and mapping. The LLM is asked to reason about what information would be helpful to plan a path of the ego vehicle and calls on functions from the tool library to build up relevant context. Like Wen et al. [138], the authors use a memory bank of prior driving experiences to bolster the context provided to the LLM. With this context, the LLM predicts a suitable path for the ego vehicle to follow. If a collision is detected between the predicted trajectory and surrounding objects in the scene, then the LLM undergoes self-reflection, like Reflexion [119], another hallucination mitigation technique, to fine-tune its prediction. Through a short study, the authors test the frequency of invalid, hallucinated outputs from the model and find that their self-reflection approach results in zero invalid generations at test-time. Sima et al. [120] build up context before predicting a path for the ego vehicle by asking a VLM questions about its perception of surrounding vehicles, predicting their behaviors, planning high-level vehicle decisions, converting to lower level discrete actions, and finally, estimating a coordinate-level trajectory. Their method, DriveLM-Agent, predicts paths from images in an end-to-end manner using a multi-modal approach, whereas Agent-Driver requires sensor modules to process context separately.

Trajectory Prediction. Wu et al. [141] propose PromptTrack as a method to predict bounding boxes and trajectories of vehicles in multi-view camera scenes conditioned on a text prompt. PromptTrack is another end-to-end method that encodes multi-view images with an image encoder, decodes previously tracked boxes and current detections into new tracks, uses a language embedding branch to predict 3D bounding boxes, and updates the memory of current tracks using past & future reasoning branches. Rather than using ego vehicle point of view images for object tracking, Keysan et al. [61] propose an approach to convert rasterized images from a bird's-eye view of the scene into a prompt describing the past trajectories of vehicles with Bézier curves. This method combines vision and language encoders to generate the Bézier curve-based scene description and elicits a language model to predict trajectories in a similar format.

Scene Understanding. Works using foundation models for generating scene descriptions given multi-modal information frame the task as a QA problem. For example, Chen et al. [15] use a **reinforcement learning (RL)** agent pre-trained on the driving task in simulation to collect a dataset containing the vehicle's state, environment observation (assumed to be ground truth from simulator) low-level action, and the ego's percentage of attention placed on different surrounding agents (if using an attention-based network). The authors ask ChatGPT [92] to act as a professional driving instructor who generates questions and answers given general driving rules and the vectorized description of the scene from the RL agent. Then, combining a pre-trained LLM with a vector embedder and former model, the architecture is trained end-to-end to answer questions from the scene. Examples of questions the architecture is posed at test-time include, "What objects are you observing?," "How are you going to drive in this situation and why?," and "What are the best tourist spots in London?" (the last of which is considered out of scope by the driving model). The authors acknowledge that the LLM may generate undesired hallucinated outputs during deployment, so they augment their training instruction dataset with out-of-scope questions that the model should learn to refuse to answer. DriveGPT4, proposed by Xu et al. [143], is a multi-modal LLM using LLaMA 2 [129] that encodes video sequences with an encoder model and projects question and video embeddings into a text form to be input to the LLM. A decoder model converts the tokenized output of the LLM into a low-level action with a corresponding high-level reason to take the action. Like Chen et al., the authors collect a driving QA dataset to align and fine-tune the architecture to driving tasks.

Control. Liu et al. [81] tackle low-level control within unsignalized intersections by formulating the problem as a multi-task decision-making process. Specifically, they train expert policies with

RL to perform individual control tasks (e.g., turning left, going forward, and turning right) with which they collect an expert demonstration dataset. An LLM model, GPT-2 [106], is fine-tuned to predict next actions given past trajectories of state and action pairs from the expert dataset. The authors showcase that their method, MTD-GPT, is able to achieve higher success rates across each of the control tasks over the original RL policies alone, showcasing the promise of foundation models to leverage their general knowledge in a multi-task problem setting.

2.2 Robotics

Foundation models have also been used in the robotics domain for object detection, affordance prediction, grounding, navigation, and communication. An example of a robot deployed with LVLM capabilities and potential hallucinations is shown in Figure 1 of Appendix A.1. Ichter et al. [53] are motivated by the issue of misalignment between the capabilities of a robot and what an LLM believes it is capable of performing. Because LLMs may not specifically be trained with data from the robot it is to be deployed on, there is a gap in the model's understanding and the true capacity of the robot, which could lead to hallucinated generations that cannot feasibly be used at runtime. The authors propose SayCan as a method to combine the general knowledge of LLMs with the specific capabilities of a robot in the real-world. Specifically, an LLM is given a task in text form and is asked to output a list of smaller actions to take to complete said task successfully. To constrain the LLM to generate possible actions available to the robot, they assume access to (1) the probability distribution of next tokens to generate from the model and (2) a set of available skills on the robot, with which they compute the probability of the LLM generating each of the skills next. SayCan greedily selects the action that has the highest product of the next token probability from the LLM and the probability of the action actually successfully being executed in the environment, until the model predicts it has completed the task.

Rather than relying purely on textual context, PaLM-E, proposed by Driess et al. [31], is a multi-modal model that converts various sensor inputs (e.g., images) to a token-space embedding that is combined with instruction embeddings to be input to a PaLM LLM [21]. PaLM is used to either answer questions about the surroundings of the robot or to plan a sequence of actions to perform to complete a task. Driess et al. further acknowledge that the multi-modality of their PaLM-E architecture leads to increased risk of hallucinations.

Inspired by recent promising findings in using foundation models to generate programs [17], other works deploy foundation models to write low-level code to be run on robots. Liang et al. [74] present Code as Policies, which uses LLMs to hierarchically generate interactive code and functions that can be called. As the model writes main code to be run on a robot given an instructive prompt of the task from the user, it identifies functions to call within the higher-level code to complete the task successfully. The authors show that LLMs can leverage third-party libraries for existing functions or develop their own library of functions dynamically with custom methods for the task. While the functionality of Code as Policies can be tested easily for low-level skill definitions, longer multi-step problems require testing whether all requested conditions have been met by running the generated code on the robot. As such, Hu et al. [48] propose the RoboEval performance benchmark for testing robot-agnostic LLM-generated code. Specifically, the CodeBotler platform provides an LLM access to abstract functions such as "pick," "place," and "get_current_location" that have the same external interface regardless of the robot to be deployed on. Like Code as Policies, CodeBotler is provided a text instruction from the user and generates code to be tested. Then, the RoboEval benchmark uses ***RoboEval Temporal Logic*** **(RTL)** to test whether the generated code meets task and temporal ordering constraints provided in the original prompt. Furthermore, they test the robustness of the LLM by passing in several paraphrased prompts to check for consistency across inputs. We discuss similar

consistency-checking strategies for identifying hallucinations in decision-making tasks further in Section 4.3.1.

In the space of robot navigation, LM-Nav leverages a VLM and attempts to predict a sequence of waypoints for a robot to follow and visit landmarks described within a language command [118]. Here, the authors use in-context learning [30] to teach GPT-3 [8] to extract desired landmarks from a natural language instruction. Assuming there are images of the possible landmarks the robot can navigate to in its environment, LM-Nav uses CLIP [105] to predict the closest matching pairs of extracted landmark descriptions and waypoint images. Finally, dynamic programming is applied on the complete graph of the environment to optimize the path of landmarks to visit. The overall predicted path is optimized to maximize the likelihood of successfully completing the instruction input to the model.

3 Hallucinations

Even with all their success on a multitude of deployment areas, foundation models still produce inconsistent outputs, or *hallucinate*, at test-time. Here, we provide a general definition for hallucinations that can be applied to any foundation model deployment task, including various autonomous systems. Additionally, we give examples of hallucinations encountered in literature and discuss how they come about during testing.

3.1 What Are Hallucinations?

Across current literature on foundation models, there exist similar patterns and themes that can be used to develop a unified definition for hallucinations. With the majority of works studying this problem within QA tasks, where ground truth answers are available, several authors explain hallucinations as producing text that includes details/facts/claims that are fictional/misleading/fabricated rather than truthful or reliable [109]. Works making use of a dedicated knowledge-base further describe hallucinations as generating nonsensical or false claims that are unsubstantiated or incorrectly cited [12, 71, 91, 157]. Varshney et al. [132] also present the idea that foundation models may sound syntactically correct, or *coherent*, while simultaneously being incorrect. Gallifant et al. [39], who perform a peer review of the GPT-4 technical paper, state that hallucinations include responses that are irrelevant to the original prompt. Li et al. [73], who specifically explore hallucinations of LVLMs in detecting and classifying objects within images, define hallucinations as generating object descriptions inconsistent with target images. A common theme among existing hallucination definitions for QA, information retrieval, and image-captioning domains is that, while the generation may sound coherent, either the output is incorrect or the model's reasoning behind the generated text is incorrect. However, we find these characteristics on their own do not completely encompass the hallucinations found in decision-making tasks in the literature, thus requiring additional nuances.

Within papers that apply foundation models to decision-making tasks specifically, researchers have encountered similar problems of hallucinations impacting performance. Park et al. [98] describe hallucinations as predicting an incorrect feasibility of an autonomous system when generating an explanation behind the uncertainty of an action to take. Similarly, Kwon et al. [67] find that language models may provide incoherent reasoning behind their actions. Wang et al. [136] and Ren et al. [110] believe that these generative models also have a sense of high (false) confidence when generating incorrect or unreasonable plans. In the case of robot navigation and object manipulation, Hu et al. [48] and Liang et al. [75] refer to hallucinations as attempting to interact with non-existent locations or objects.

In the code generation task, Chen et al. [17] use the term "alignment failure" with similar effects to those of hallucinations discussed above. More specifically, the authors informally describe an

Table 1. Definitions for Compliance, Desirability, Relevancy, and Plausibility—the Four Characteristics of Hallucinations

Characteristic	Definition
Compliance	Generation meets hard constraints that cannot be ignored
Desirability	Generation attempts to meet soft constraints measured by some cost or reward function
Relevancy	Contents of generation do not fall outside of a defined set of topics
Plausibility	The syntactic similarity of a generation and in-domain normal samples measured via a critic function

alignment failure as an outcome where a model is *capable* of performing a task but *chooses* not to. If a model is able to complete a task successfully within its latent space (perhaps through additional prompt engineering or fine-tuning), then one may ask, "Why would the model *choose* not to?" As foundation models are trained with the next-token reconstruction objective on a training set, they attempt to maximize the likelihood of the next token appearing at test-time as well. Consequently, if the test-time prompt includes even minor mistakes, then Chen et al. find that LLMs will continue to generate buggy code to match the input prompt. This issue is further described in Section 3.3.

We realize existing definitions for hallucinations are extremely disparate, depending on the deployment area. Nevertheless, we have identified three distinct features that are commonly missing within hallucinated generations: *compliance*, *desirability*, and *relevancy*. Additionally, we find hallucinations may seem to be *plausible* while they are in fact unacceptable. A compliance metric checks that the generation meets hard constraints, while the desirability metric measures how well the generation meets soft constraints defined by the model engineer. Irrelevant generations refer to predictions that contain details outside of a set of requested topics. Notice that irrelevant predictions can still be considered compliant and desirable, depending on how the hard and soft constraints are defined. Plausibility compares the generation's syntax to those of a set of known, unhallucinated samples. While relevancy is evaluating the content of a prediction, plausibility is assessing its phrasing. Table 1 provides complete definitions for each of the four characteristics.

Then, to bridge definitions from existing QA application areas, decision-making tasks, and all other possible test scenarios for foundation models, we combine these findings and define the term hallucination as follows:

Definition 3.1. A hallucination is a generated output from a model that conflicts with constraints or deviates from desired behavior in actual deployment or is completely irrelevant to the task at hand but could be deemed syntactically plausible under the circumstances.

There are three key pieces to this definition:

(1) A generated output from a model.
(2) A deployment scenario to evaluate model outputs with any of the following:
    — A list of constraints that must be *compliant* within the generation.
    — A loose interpretation of a *desired behavior* the generation should meet.
    — A set of topics *relevant* to the task.
(3) Metrics measuring compliance, desirability, relevancy, and syntactic soundness (*plausibility*) of generations.

In practice, this definition generally encapsulates the qualities of hallucinations discussed earlier. For example, in QA or image-captioning tasks, one may define a set of relevant topics that a generation should not stray from, and constraints may be held in the form a knowledge-base of ground truth facts. The desired behavior of the generation may be to be phrased in an informative

Table 2. Examples of Applying Definition 3.1 to Different Tasks

Problem Setting	Characteristic			
	Compliance	**Desired Behavior**	**Relevancy**	**Plausibility**
Question-answering	Generations must align with database facts	Tone of answer should be informative	Answers should not include references to unrelated topics	Generation is syntactically sound and believable
Image Captioning	Objects in description must appear in image	Censor descriptions for inappropriate images	Descriptions should not be embellished with details that cannot be confirmed	
Planning	Predicted sub-task must be feasible to solve	Plans should maximize expected return	Predicted sub-tasks and actions should not stray from the end goal with added steps	Generated plan is reasonable and seems to attempt to accomplish goal
Control	Predicted action must be possible to perform	Predict actions to complete plan efficiently		

Note that developers may choose to only define a subset of hallucination characteristics for their deployment, depending on evaluation preferences. The table is split into non-decision-making and decision-making applications.

manner, rather than sarcastic. However, in robot manipulation settings, a developer may have a set of constrained actions feasible on the robot, and the desired behavior could be to complete a task with as few actions as possible. Relevancy may be measured in relation to the specific task to be deployed on (e.g., a prompt requesting a recipe to make pasta would find it irrelevant if the model also suggested a song to play while cooking). Finally, plausibility informally relates to a measure of how believable an output is to a critic. A more realistic generation has a greater chance of deceiving the user into trusting the model, even when the plan may be hallucinated. Overall, hallucinated outputs may contain one or more of the core characteristics (noncompliant, undesired, irrelevant, and plausible) simultaneously, and our definition can be flexibly applied to any deployment scenario in mind by choosing metrics for each characteristic, respectively. We show more examples of applying our definition to various tasks in Table 2.

3.2 Examples

Driving Tasks. As discussed in Section 2.1, Wen et al. [139] test GPT-4V on the autonomous driving task and identify failure modes. Regardless of the weather and driving conditions, GPT-4V has difficulty detecting and identifying the traffic-light state at an intersection until the image has zoomed in on the light itself. It also presents additional irrelevant (or completely false) details about other agents when the prompt had no mention of them in the first place. Furthermore, the model also has difficulty in describing temporal sequences (i.e., videos) and categorizing images by their direction within a panoramic view from the vehicle's perspective. In their later work, Wen et al. [138] describe that hallucinations arise in these complex environments because of the high variability in driving scenarios. Even after applying hallucination mitigation techniques like chain-of-thought reasoning, the model is not free of these undesired outputs. A similar work evaluating the frequency at which LVLMs hallucinate in their descriptions of images finds that these models' outputs may include non-existent objects or additional irrelevant phrases (that may not even be possible to test for accuracy) [73]. For example, in a picture of food on a table, an LVLM hallucinates a non-existent beverage and predicts that the "table is neatly arranged, showcasing the different food items in an appetizing manner." Although the classification error and irrelevant generation in this example are not critical, earlier works warn of possible failures with more severe, high societal impact (e.g., biases in models leading to marginalizing users) [6].

Code Generation. Chen et al. [17] explore alignment failures of LLMs applied to code completion tasks. The authors evaluate the likelihood of these models generating defective code given different input prompts and discover that in-context learning using examples with buggy code has a higher chance of resulting in poor generations from the model on the actual task at hand. The

study also identifies similar model biases towards race, gender, religion, and other representations. Furthermore, the authors find that their model, Codex, is able to generate code that could assist with developing insecure applications or malware, albeit in a limited manner. These findings have been corroborated by other foundation model code generation works in the robotics domain. For example, Wang et al. [134] describe that Voyager sometimes generates code with references to items that do not exist within MineDojo. Similarly, Hu et al. [48] find that their model has the tendency to call functions with invalid objects or locations, pick up objects when it is already holding something, ask for help when no one is near, and other undesired behaviors.

Question-answering Domain. Several works focus on identifying cases of hallucinations in QA tasks. Although this application area is not the direct focus of this work, we present examples of hallucinations in this field, as we can glean similar failure modes that could arise within decision-making systems. Common hallucinations in QA result in incorrect answers to questions. For example, Achiam et al. [1] find that GPT-4 "hallucinates facts and makes reasoning errors." Achiam et al. categorize these failures into closed-domain (given context, the model generates irrelevant information that was not in the context) and open-domain (the model outputs incorrect claims without any context) hallucinations. After fine-tuning on more data with a hallucination mitigation objective, the model reduces its tendency to hallucinate but still does not achieve perfect accuracy—a similar trend encountered by Touvron et al. [128]. Another set of works identify hallucinations with contradictions among several sampled generations from an LLM, discussed further in Section 4.3.1 [91, 157]. Intuitively, if a context passed into a model results in conflicting generations, then the model must be hallucinating some part of the output. Notice in this example, with relation to Definition 3.1, self-contradiction works test for compliance by checking *consistency* among multiple (hallucinated) generations rather than with respect to a ground-truth knowledge-base that usually exists in QA tasks. As such, our definition can flexibly apply to different system setups by describing compliance, desired behavior, and relevancy, respectively.

Additional examples of hallucinations encountered in image, video, and 3D generation methods and broader impacts faced by medical, legal, and finance industries are discussed in Appendix B.1 and B.2.

3.3 Why Do They Happen?

There are several speculations as to how hallucinations come about during deployment. First and foremost, like any learning task, foundation models are sensitive to biases in training data [109]. Once a model is trained on a given large dataset, some facts may become out-of-date or stale at any point in time [102]. Furthermore, as the training set is embedded into a smaller encoding dimension, the knowledge within an L(V)LM's frozen parameters is lossy, and models cannot feasibly be fine-tuned every time there is new data [34, 100]. Zhang et al. [157] recommend changing algorithm parameters at runtime, such as, *temperature* (spread of probability distribution of next token), *top-K sampling* (narrows the set of next tokens to be considered), and *beam search* (choosing a set of possible beams, i.e., trajectories, of next tokens based on high conditional probabilities), but the process of tuning these parameters is expensive.

To combat out-of-date training data, some works provide models with an external knowledge base of information to pull facts from, with the hope of increasing model accuracy. Even with this up-to-date information, Zhang et al. [158] pose that there may exist a misalignment between the true capabilities of a model and what a user believes the model is capable of, leading to poor prompt engineering. In fact, poor prompting is one of the most significant causes of hallucinations. Chen et al. [17] find that poor quality prompts lead to poor quality generations in the context of code completion. This phenomenon is attributed to the reconstruction training objective of LLMs

attempting to maximize the likelihood of next generated tokens, given context and past outputs [89], i.e.,

$$\log p\left(s|x\right) = \sum_{i=1}^{N} \log p\left(\sigma_i|x, \sigma_{i-k} \ldots \sigma_{i-1}\right),$$

where x is a context input to the model, s is an output sequence of N tokens $\sigma_1 \ldots \sigma_N$, and any generated token σ_i is conditioned on k previously generated tokens. As the public datasets these models are trained on contain some fraction of undesirable generations (e.g., defective code), the models become biased to generate similar results under those inputs. Qiu et al. [103] show that this limitation can actually be exploited to push foundation models to generate toxic sentences or completely lie by simply rewording the prompt.

While foundation models condition generated tokens on ground-truth text without hallucinations at train time, during inference, the model chooses future tokens conditioned on previously (possibly hallucinated) generated text. As such, Chen et al. [19] and Varshney et al. [132] state that generated outputs are more likely to contain hallucinations if prior tokens are hallucinated as well. Furthermore, Li et al. [69] find that, even if prompt context provided to a foundation model is relevant, the model may choose to ignore the information and revert to its own (possibly outdated or biased) parameterized knowledge.

Overall, the hallucination detection task is highly complex with several possible sources of failures that need to be considered at test-time. Chen and Shu [13] validate the complexity of the detection problem with studies identifying that human- and machine-based detectors have higher difficulty correctly classifying misinformation generated from LLMs than those written by other people.

4 Detection and Mitigation Strategies

Hallucination detection and mitigation methods can be classified into three types (white-, grey-, and black-box), depending on the available inputs to the algorithm. Generally, given some context, a foundation model outputs a predicted sequence of tokens, the corresponding probabilities of each token, and embeddings of the generation from intermediate layers in the network. White-box hallucination detection methods assume access to all three output types, grey-box require token probabilities, and black-box only need the predicted sequence of tokens. Because not all foundation models provide access to their hidden states, or even the output probability distribution of tokens (e.g., the ChatGPT web interface), black-box algorithms are more flexible during testing. In this section, we present existing detection and mitigation approaches clustered by input type. While several of these works show promise in QA and image-captioning settings, many of them require further validation on decision-making tasks, and we will point out these methods as they come about. Works in this section are summarized in Table 3. We also provide additional details about the evaluation setups of works deployed on custom datasets, simulators, or the real-world. Certain methods that are less related to decision-making are described in Appendix C. Additionally, frequently used metrics, datasets, and simulators are summarized in Appendix D.

4.1 White-box Methods

Methods in this section require access to internal weights of the model for hallucination detection.

4.1.1 Hidden States. Some approaches utilize intermediate embeddings at different network layers.

(1) Azaria and Mitchell [4] empirically find that language models attempt to correct themselves after outputting an untruthful claim. As such, they hypothesize that the internal states of

Table 3. A Summary of Hallucination Detection & Mitigation Methods Discussed in Section 4

Modality	Method Type	Method ID	Application		Deployment	Evaluation Setting
			Detection	Mitigation		
White-box	Hidden States	4.1.1.1	•		QA	CD
		4.1.1.2	•		QA	CD
		4.1.1.3	•		QA	DecodingTrust [133] TruthfulQA [78]
	Attention Weights	4.1.2.1	•	•	IC	MSCOCO [79] Visual Genome [64]
	Honesty Alignment	4.1.3.1	•		QA	CD
		4.1.3.2	•	•	QA	TriviaQA [58]
Grey-box	Concept Probabilities	4.2.1.1	•	•	IR/QA	HotpotQA [147] CD
		4.2.1.2		•	IC	MSCOCO [79]
		4.2.1.3		•	P	Ravens [154] BabyAI [20] VirtualHome [101]
	Conformal Prediction	4.2.2.1	•	•	IR/QA	MIMIC-CXR [57] CNN/DM [47] TriviaQA [58]
		4.2.2.2	•	•	QA	MMLU [46]
		4.2.2.3	•	•	P	TableSim [110] RW
		4.2.2.4	•	•	P	TableSim [110] CD
		4.2.2.5	•	•	P	CS
Black-box	Analyzing Samples	4.3.1.1	•		IR/QA	CD
		4.3.1.2	•	•	IR/QA	CD
		4.3.1.3	•	•	IR/QA	GSM8K [23] MMLU [46] CD
		4.3.1.4	•	•	P	SaGC [98] CS RW
		4.3.1.5	•	•	IR/QA	CD
		4.3.1.6	•	•	IR/QA	Quest [84] MultiSpanQA [70] FActScore [90] CD

(Continued)

Table 3. Continued

Modality	Method Type	Method ID	Application		Deployment	Evaluation Setting
			Detection	Mitigation		
Black-box	Analyzing Samples	4.3.1.7	•		IC/QA	MSCOCO [79] A-OKVQA [116] GQA [52]
		4.3.1.8	•		QA	BIG-Bench [124] GSM8K [23] MMLU [46]
		4.3.1.9	•		QA	CD
	Adversarial Prompting	4.3.2.1	•		IG	CD
		4.3.2.2	•	•	IR/QA	Natural Questions [66] CD
		4.3.2.3	•		QA	CD
		4.3.2.4	•		QA	CD
	Proxy Model	4.3.3.1	•		IR/QA	CD
		4.3.3.2	•		IR/QA	SQuAD [107] HotpotQA [147] TriviaQA [58]
		4.3.3.3	•		IR/QA	CD
		4.3.3.4	•		IG	CD
	Grounding Knowledge	4.3.4.1		•	IR/QA	Natural Questions [66] StrategyQA [40] QRreCC [2]
		4.3.4.2	•	•	IR/QA	LC-QuAD [130] KQA-Pro [9] ScienceQA [83]
		4.3.4.3	•	•	IR/QA	FuzzyQA [158]
		4.3.4.4	•	•	IR/QA	CD
		4.3.4.5		•	P	TextWorld [24]
		4.3.4.6	•	•	IG	CD
		4.3.4.7		•	IG	I2P [113]
	Constraint Satisfaction	**4.3.5.1**	•	•	P	CD
		4.3.5.2	•		P	RoboEval [48]

Deployment scenarios are split into question-answering (QA), information retrieval (IR), image captioning (IC), image generation (IG), and planning (P) tasks. The method ID includes the subsection in which the method appears in the paper and the order in which it appears in the subsection. Bolded method IDs are deployed to decision-making tasks specifically. Custom datasets, custom simulators, and real-world experiments for testing are abbreviated as CD, CS, and RW, respectively.

a model must have some understanding of whether the output is correct. Furthermore, the authors stray away from directly using the token probabilities, even though they have correlation with model accuracy [132], because the complete output sentence's probability is dependent on the length of the generation and appearance frequency of tokens. Azaria and Mitchell present SAPLMA, a simple classifier trained with supervised learning, that takes the activation values of a hidden layer of an LLM as input and outputs the probability of the

generated claim being true. The authors choose to collect a new dataset of simple statements with corresponding true/false answers, since popular datasets like FEVER [126] do not partition statements by topic, and some statements cannot be cleanly classified. In total, their dataset contains 6K sentences spanning six topics. SAPLMA is shown to be able to identify untruthful outputs, even when trained on a held-out dataset on a completely different topic than evaluated on.

(2) Yao et al. [148] aim to test the resiliency of foundation models to varying prompts (a form of adversarial prompting further discussed in Section 4.3.2). They propose perturbing an input prompt with additional tokens to make an LLM under test produce a desired hallucination (e.g., modify the original query, "Who won the 2020 US election," to get the LLM to generate, "Donald Trump was the victor," while its original response was correctly stated as, "Joe Biden was the victor"). As the search space of possible tokens to add/replace when developing the adversarial prompt is massive, the work uses a gradient-based token replacing strategy. Specifically, they define an objective that attempts to find trigger tokens in the direction of the gradient that maximizes the likelihood of the model outputting the desired hallucination. To evaluate their approach, Yao et al. collect a dataset of truthful facts by feeding questions from Wikipedia [36] into an LLM. Then, certain subjects, objects, or predicates in the generated answers are replaced to manually create hallucinations. With simple prompt modifications, the authors show that the white-box approach is able to induce the specified hallucinations.

(3) The work from Song et al. [122] is described in Appendix C.1.1.3.

4.1.2 Attention Weights. Attention weight matrices, which are prominent within transformer model architectures, signify the importance the model places on earlier tokens within a generation when predicting future tokens.

(1) OPERA, proposed by Huang et al. [50], is a hallucination detection method for LVLMs that makes use of the model's internal attention weights. When visualizing the attention matrix, the authors find that there exist peculiar column patterns that align with the beginning of a hallucinated phrase. These *aggregation patterns* usually occur on a non-substantial token such as a period or quotation mark but are deemed to have a large impact on the prediction of future tokens. As such, this finding led Huang et al. to modify the beam search algorithm [37] by applying a penalty term to beams wherever an aggregation pattern is detected and roll back the search to before the pattern arises. Their method is shown to reduce hallucinations and even eliminate possible repetitions in generations.

4.1.3 Honesty Alignment. In addition to methods that require hidden states or attention matrices, we also include methods that fine-tune foundation models to better communicate their uncertainty to questions under white-box algorithms, as they require access to model weights for training.

(1) For example, Lin et al. [77] collect a calibration dataset of questions and answers from GPT-3 under 21 types of arithmetic tasks (e.g., add/subtract and multiply/divide) and record how often each task is incorrectly answered. They aim to fine-tune the LLM to also output its certainty that the prediction is correct. Consequently, Lin et al. fine-tune the model with data pairs of a question and the empirical accuracy on the task that the question originates from in the calibration dataset, such that the model is expected to similarly output a probability of accuracy at test-time. The authors show that the proposed verbalized probability in deployment does correlate with actual accuracy on the tasks. Specifically, Lin et al. find that certain problems, like multiplication, are more challenging for GPT-3 to answer correctly. By calibrating the model on a set of simpler questions (add/subtract), the model

is able to generalize its verbal uncertainty to more challenging tasks with multiple possible answer choices.

(2) The work from Yang et al. [145] is described in Appendix C.1.3.2.

4.2 Grey-box Methods

Grey-box approaches leverage the probability distributions of tokens output from the model.

4.2.1 Concept Probabilities.

(1) Empirically, Varshney et al. [132] show that there is a negative correlation between hallucination rate and token probability (i.e., as a token's probability decreases within a sentence, the tendency to hallucinate increases). Thus, the authors rely on token probabilities to estimate uncertainty of concepts within a generated claim, and they check for correctness by cross-referencing a knowledge-base. Whenever a concept is found to be conflicting with a fact through verification questions, their method attempts to mitigate the error by prompting the LLM to replace the incorrect claim with the evidence. The authors query GPT-3.5 for summaries about 150 different topics sampled from popular Wikipedia subjects, including sports, music, and politics. To ensure the accuracy of labels, the authors manually label the truthfulness of the first five sentences in each generated article rather than relying on crowd-sourced labels. Although effective in the QA setting, Varshney et al. concede that, in the event token probabilities are not available, some form of heuristic must be used to detect hallucination candidates.

(2) Zhou et al. [164] show that external models can be developed to automatically *clean* hallucinations. The authors tackle the issue of object hallucinations that LVLMs experience when describing the content of images. Through theoretical formulations, the authors show that LVLM responses tend to hallucinate in three settings: when described object classes appear frequently within a description, when a token output has low probability, and when an object appears closer to the end of the response. As such, their model, LURE, is a fine-tuned LVLM trained on a denoising objective with a training dataset that is augmented to include objects that appear frequently within responses, and replacing objects with low token probabilities or appearing close to the end of the response with a placeholder tag. At inference time, tokens are augmented similarly to how they were changed to generate the training dataset, and the LURE LVLM is prompted to denoise hallucinations by filling in uncertain objects.

(3) SayCanPay, proposed by Hazra et al. [43], builds off of the SayCan framework [53] to improve the expected payoff of following a plan specified by a language model. Within our hallucination definition, this goal translates to increasing the desirability of generations by improving the likelihood of the model achieving higher rewards. The authors propose three different strategies for planning: Say, SayCan, and SayCanPay. Say methods greedily choose next actions based only on token probabilities. SayCan approaches also take the success rate of the chosen action into consideration. Finally, SayCanPay additionally estimates the expected payoff from following the plan with some heuristic. Hazra et al. learn this Pay model with regression on an expert trajectory dataset. Combining all three models together minimizes the likelihood that a generated plan contains conflicting infeasible action calls while maximizing the efficiency of the task completion.

4.2.2 Conformal Prediction. Another range of works estimate the uncertainty of a model output with conformal prediction to provide statistical guarantees on the likelihood of predictions being correct [117].

(1) Quach et al. [104] propose conformal language modeling to build a set of possible candidate responses to a test prompt, while calibrating algorithm parameters on a held-out dataset of independent prompts and their corresponding admission functions, which check whether a model output meets the criteria of an input prompt. In their algorithm, the authors calibrate thresholds for three separate scoring functions that test for generation quality, similarity with other responses, and model confidence using "Learn then Test" [3]. At inference time, given scoring functions, calibrated thresholds, and an input prompt, the method samples outputs from the model and adds them to a prediction set if they meet quality and similarity thresholds until the whole set is guaranteed to meet the user-defined confidence parameter.
(2) The work from Kumar et al. [65] is described in Appendix C.2.2.2.
(3) While the previous hallucination mitigation works presented using conformal prediction are solely applied to QA settings, Ren et al. [110] are the first to apply conformal prediction of foundation models to robotic tasks. The authors are motivated by a desire for language-conditioned robots to understand when they are uncertain about the next action to take, such that they can ask for help in those cases (while minimizing frequency of clarifications). Because LLM generations with different length sequences inherently produce different complete sentence probabilities, the authors propose framing the control task as a multiple-choice problem, like Kumar et al. [65]. Their approach, KnowNo, prompts an LLM to generate a possible set of next actions to take in multiple choice form. They first collect and hand-label a calibration dataset of pairs of held-out instructions and the probability of the model choosing the best action to take next. At test-time, the model outputs a set of actions to take and KnowNo eliminates actions with token probabilities less than a calibrated certainty from a user. If there are still multiple actions left in the prediction set after eliminating uncertain actions, then the model queries a human for help choosing the next action. Ren et al. show that KnowNo deviates from the user-defined error rate least often compared to methods that do not use conformal prediction and has the highest success-to-clarification ratio. Additionally, the authors deploy KnowNo to a real UR5 robot arm, where it is tasked with sorting foods on a table by order of user preference and disposing of undesired objects conditioned on ambiguous instructions. However, several assumptions had to be made to produce the demonstrated results, including having access to next token probabilities, having resources to collect a large calibration dataset, presuming people will faithfully provide help when asked for it, and using ground-truth vision to fully ground the environmental objects with the text input to the model.
(4) Liang et al. [75] extend the KnowNo methodology by incorporating an introspective planning step using a previously constructed knowledge-base of experiences, which tends to (1) enhance quality of generated plans and (2) improve interpretability of decisions. Specifically, introspective planning first constructs a knowledge-base containing training pairs of tasks, observations, and valid plans, which the LLM is prompted to generate explanations behind why they are reasonable. Each experience is stored with a key as an embedding of the original instruction. During inference, given a new test instruction, their method queries the database to find the key with the closest embedding to that of the new instruction. This previous experience and reasoning is fed into the model to generate a set of candidate plans to follow. Finally, the remainder of the algorithm follows the same process as KnowNo to calibrate and narrow down the prediction set to fall within a desired error rate. Liang et al. evaluate their method on the KnowNo dataset [110] and an augmentation of the original dataset that considers more safety-critical tasks with ambiguous instructions (e.g., making sure not to place metal objects in a microwave when tasked with heating a bowl).

(5) Wang et al. [136] aim to provide additional guarantees on completing the task provided within the natural language instruction. To do so, the authors propose a novel task specification, LTL-NL, which combines **linear-temporal-logic (LTL)** descriptions with natural language from a user instruction, which the authors claim is easier to define than classical LTL specifications. Given this specification, a symbolic task planner chooses a sub-task to complete next and an LLM generates plans for each sub-task, respectively. Like Ren et al. [110] and Liang et al. [75], Wang et al. apply conformal prediction to minimize the number of possible actions to take next within some desired error rate. However, rather than directly asking a user for assistance when there is high uncertainty in the next action to take (or when there are environmental constraints), their method, HERACLEs, samples a new sub-task to complete from the task planner. If, however, the task planner is unable to provide a new sub-task, HERACLEs requests help from the user. The authors deploy the model and baselines in a custom-made 3D mobile manipulator simulator that allows for evaluation of methods that use LTL specifications. For example, the instruction, "Deliver Apple to A" is converted to LTL specifications and fed into HERACLEs to navigate the robot to (1) pick up the requested apple and (2) drop it off at location A. With experimentation, the authors find that their method achieves higher task completion rate on missions requiring more sub-tasks, outperforming baseline planners that do not utilize LTL specifications.

4.3 Black-box Methods

Black-box algorithms only rely on the input prompts and output predictions from the model without making assumptions on the availability of the hidden state nor the token probabilities.

4.3.1 Analyzing Samples from Model. As a result, several works of this type sample multiple sentences from an LLM and measure the similarity of the information present in all samples.

(1) For example, SelfCheckGPT, proposed by Manakul et al. [85], samples multiple responses (each of which may contain many sentences) from an LLM to a single query and measures the consistency among the varied responses through a study with five different approaches. SelfCheckGPT with BERTScore [159], for example, computes a similarity score between two sentences from different sampled outputs. Intuitively, a hallucination is detected when the similarity score for a sentence is low across all other samples. Other consistency-checking methods the authors consider include using an automatic multiple-choice QA system conditioned on the responses, relying on a proxy-LLM that has access to token probabilities, training an external classifier to predict contradictions (coined SelfCheck-NLI), and directly prompting the LLM to evaluate whether any sentence can be supported by another sampled response. Before evaluating the proposed methods, the authors find that there is no standardized hallucination detection dataset for question-answering. Furthermore, the existing hallucination dataset from Liu et al. [82] is generated by manually replacing tokens in actual facts, which may not represent generations from real language models. Thus, Manakul et al. choose to curate their own evaluation dataset by querying GPT-3 for articles on topics from the WikiBio dataset [68]. They then manually label the accuracy of each sentence in each generated article for a total of 1,908 sentences across 238 summaries. As expected, Manakul et al. find that sampling more responses from the model leads to better estimation of the validity of a claim but is slower to compute.

(2) The work from Elaraby et al. [34] is described in Appendix C.3.1.2.

(3) Rather than analyzing the responses of a single language model, Du et al. [32] take an ensemble approach. Specifically, they propose pitting multiple instances of an LLM into a debate of the correct answer to a question. In practice, several agents are given the same

question and predict a response. Over multiple iterations, all the responses of other agents are concatenated and fed in as additional context to each model, and a new response is sampled. Du et al. first identify whether the proposed debate method can improve the *reasoning* of language models through three offline tasks with increasing difficulty in a custom benchmark: (1) simple arithmetic, (2) grade-school math [46], and (3) predicting the next best move to take in a game of chess. They additionally evaluate the *accuracy* of debate results in (1) generating biographies of famous computer scientists from Wikipedia, (2) general exam knowledge [46], and (3) confirming the validity of next moves in chess. The authors show that a combination of their iterative ensemble approach with chain-of-thought reasoning mitigates individual hallucinations (as agents tend to converge on a single consensus) and increases QA accuracy. While only tested on offline datasets, the approach could be utilized within a simulation framework, like the one presented by Park et al. [99], where, for example, multiple language agents may debate about plans to make. It is important to note that Du et al. evaluate the accuracy of biography generations by querying ChatGPT for the consistency between a fact and generated response. We hypothesize that, like Yu et al. [152], this automatic labeling scheme will result in lower-quality labels than manual annotation.

(4) CLARA is a framework engineered by Park et al. [98] that predicts when an instruction provided to a robotic system controlled by an LLM may be ambiguous or infeasible. Intuitively, if a language model is uncertain about an instruction, then it might output diverse (or conflicting) actions to other instructions that hold similar information. Thus, CLARA samples several sets of concepts from the original prompt, randomly orders them to assemble multiple inputs to the model, and passes them as input to the language model under test. The method computes the average similarity of pairs of output actions in an embedding space over all outputs. Next, to check for infeasibility of the original goal, the foundation model is provided the possible action space of the robot, environmental observation, and goal and is prompted to output whether the desired task is practical. In the event the model is uncertain from the multi-prompting step, but the goal is feasible, CLARA asks for clarification from the user with reasoning for why it is uncertain. In addition to evaluating CLARA on SaGC, a custom dataset described in Appendix D.2.2, the authors also put their model to the test on a tabletop manipulator robot in simulation and the real world. While the method achieves a reasonable success rate on robotic pick-and-place tasks with real user instructions (where other discussed methods are primarily evaluated in QA settings), there are still failure cases where the model hallucinates during uncertainty reasoning and feasibility prediction.

(5) Another set of works explicitly identify contradictions among responses instead of estimating similarity. Naturally, detecting a self-contradiction is guaranteed to reveal an invalid claim. Mündler et al. [91] pose that removing detected conflicting information will increase the validity of a generated response. As such, the authors suggest a solution that finds important concepts within a response to be evaluated, prompts the generation model to generate more information about each of the concepts, and uses a separate analyzer language model to evaluate the consistency of pairs of sentences on the same concept. Any sentences that are found to be conflicting are revised by the analyzer model before being output to the user. The authors ask four language models to generate 360 summaries for 30 diverse topics from WikiBio [68] and Wikipedia [36] and have three human annotators manually label any instances of self-contradiction, inaccurate sentences, and unverifiable statements, leading to high-quality labels. With respect to Definition 3.1, the annotators are identifying cases of noncompliant and irrelevant generations to decide which statements are hallucinations.

(6) Rather than relying on another language model to analyze the correctness of predictions, Dhuliawala et al. [29] utilize chain-of-thought reasoning to prompt an LLM to generate

possible verification questions about its original responses. If there are conflicts in answers to the verification questions, then the LLM is prompted to regenerate its output with updated context and conflicting reasoning. One of the first evaluations the authors perform is on a custom set of 56 questions with topics collected from Wikipedia, each of which have multiple correct answers for a total of around 600 ground-truth entities.

(7) The work from Li et al. [73] is described in Appendix C.3.1.7.
(8) The work from Xiong et al. [142] is described in Appendix C.3.1.8.
(9) The work from Yehuda et al. [150] is described in Appendix C.3.1.9.

4.3.2 Adversarial Prompting. Works specializing in adversarial prompting attempt to test the robustness of models to varying inputs that may coerce the model into producing out-of-distribution results.

(1) For example, Mehrabi et al. [87] apply adversarial prompting to text-to-image foundation models, like Stable Diffusion [113], to generate offensive images. With respect to Definition 3.1, their framework, FLIRT, is essentially testing the tendency of foundation models to hallucinate undesired generations in deployment. FLIRT uses an adversarial language model to predict a prompt to input to the image generator, scores the generated image for the presence of undesirable traits using an external classifier, re-prompts the adversary to produce a new instruction conditioned on the findings of the classifier, and repeatedly generates images until the adversary successfully prompts the test model to output an undesirable result. Mehrabi et al. define objective functions conditioned on the score output by external classifiers to maximize diversity of adversarial prompts and minimize toxicity to pass text filters that detect malicious inputs while improving attack effectiveness. The authors form a large set of prompts with varying levels of detail across different sexual, violent, hate, and so on, contexts, inspired by Schramowski et al. [113]. Each prompt corresponds to one of three test splits that change the type of toxicity, level of detail, and phrasing in the prompt. Tangentially, Mehrabi et al. evaluate FLIRT on text-to-text models by similarly collecting adversarial prompts with varying vulgarity.
(2) Another work from Yu et al. [153] presents the AutoDebug framework for automatically sampling and updating several prompts for use in adversarial testing of the language model. Yu et al. argue that evaluating information-retrieval LLMs on popular datasets like Wikipedia provides an over-approximation on the accuracy of these models, since they have been overfit on the same data, possibly leading to memorization. Thus, the authors specifically explore adversarial testing under the case that the model predicts a correct response when provided relevant context but generates an incorrect prediction when the evidence is modified. They apply two different modification approaches: replacing tokens within the context to provide incorrect facts and adding additional relevant facts to the prompt that may make it difficult to pick out the most important details. The authors collect a new dataset by applying their adversarial generator to Natural Questions [66] and RealtimeQA [59] with additional human filtering to ensure plausible answers are collected.
(3) The work from Ramakrishna et al. [108] is described in Appendix C.3.2.3.
(4) The work from Uluoglakci and Temizel [131] is described in Appendix C.3.2.4.

All in all, adversarial prompting is an effective method for identifying robustness of models to unseen inputs, which can be used to develop stronger input filters or fine-tune the model for decreased hallucination tendency.

4.3.3 Proxy Model. Certain black-box works rely on an external, proxy model to detect and mitigate hallucinations.

(1) One such method is used as a baseline within the SelfCheckGPT article [85]. As many language foundation models do not provide access to token probabilities, the authors use an open-source proxy LLM that does provide token probabilities as an estimate of the original output's probability. They find that using proxy LLMs for probability estimation and hallucination detection successfully is highly variable. The accuracy of detection is dependent on the complexity of the LLM itself, as well as the training data of the proxy LLM (i.e., models trained on independent datasets from the original LLM will have different generation patterns). Refer to the description of Method ID 4.3.1.1 for a discussion on the custom dataset used for evaluation.
(2) Within this section, we also include works that use an external trained classifier to detect hallucinations. For example, Chen et al. [19] curate a dataset of QA dialogue from LLM-generated responses. They apply a composition of metrics to assess quality of responses, including a self-assessment from the LLM comparing the ground-truth and predicted text, human-labeled, and machine metrics (e.g., BERTScore [159], F1 score, BLEU [97]). Their hallucination discriminator, RelD, is trained on the dataset in multiple separate phases, each using a different objective: regression, multi-class classification, and finally binary classification. Through experiments, they find that RelD closely aligns with human evaluators' original predictions.
(3) The work from Pacchiardi et al. [96] is described in Appendix C.3.3.3.
(4) The work from Chu et al. [22] is described in Appendix C.3.3.4.

4.3.4 Grounding Knowledge. In knowledge grounding tasks, a language model is tasked with identifying evidence from an external knowledge-base that supports claims within a summary. Although seemingly irrelevant to decision-making scenarios, similar methods to ones discussed in this section may be applied in planning tasks to identify observations that are most relevant to predicting the next action or to generate reasoning behind a specified plan.

(1) PURR, proposed by Chen et al. [12], is a denoising agent, like LURE (discussed in Section 4.2.1), that is trained in an unsupervised fashion given evidence from online sources, a clean (correct) summary, and a noisy (hallucinated) summary. The model learns to denoise the incorrect summary to the clean statement. During deployment, given a possibly hallucinated claim, a question-generation model queries online sources for evidence about the claim, and PURR generates a cleaned version of the original summary with said evidence.
(2) The work from Li et al. [72] is described in Appendix C.3.4.2.
(3) The work from Zhang et al. [158] is described in Appendix C.3.4.3.
(4) Peng et al. [100] aim to add plug-and-play modules to an LLM to make its outputs more accurate, since these large foundation models cannot feasibly be fine-tuned whenever there is new information. Their work formulates the user conversation system as a **Markov decision process (MDP)** whose state space is an infinite set of dialogue states that encode the information stored in a memory bank and whose discrete action space includes actions to call a knowledge consolidator to summarize evidence, to call an LLM prompt engine to generate responses, and to send its response to the user if it passes verification with a utility module. The proposed LLM-Augmenter has a memory storing dialogue history, evidence from the consolidator, set of output responses from an LLM, and utility module results. Its policy is trained in multiple phases with REINFORCE [140] starting with bootstrapping from a rule-based policy designed from domain experts, then learning from simulators, and finally, from real users. Peng et al. deploy LLM-Augmenter to two different information-retrieval domains: news and customer service. For the news application, the authors retrieve relevant news articles by crawling Reddit news forums for 1.3K articles,

following the DSTC7 Track 2 [151] approach. Similarly, to emulate the required knowledge of a customer-service chatbot, the authors use the data from DSTC11 Track 5 [62], which holds 14.7K examples of user reviews and frequently answered questions. The authors find that access to ground-truth knowledge drastically improves QA results, and feedback from the utility module and knowledge consolidator help to provide more accurate answers.

(5) Evaluated in decision-making settings, Introspective Tips [16] provide concise, relevant information to a language planner to learn to solve problems more efficiently. Intuitively, summaries that collect information over all past experiences may be long and contain unnecessary details. In contrast, tips are compact information with high-level guidance that can be learned from one's own experiences, from other demonstrations, and from other tasks in a similar setting. Chen et al. show that providing low-level trajectories is less effective than tips on simulated planning tasks. Additionally, with expert demonstrations, the LLM learns faster with fewer number of failed trials than with just past experience alone. However, one limitation identified in the study is that the LLM underperforms in unseen, low-difficulty missions where it has issues generating general tips for zero-shot testing.

(6) The work from Lim and Shim [76] is described in Appendix C.3.4.6.

(7) The work from Schramowski et al. [113] is described in Appendix C.3.4.7.

4.3.5 Constraint Satisfaction. There is also additional work in creating black-box algorithms for ensuring decision plans generated by foundation models meet user-defined goal specifications and system constraints, like their grey-box counterpart developed by Wang et al. [136].

(1) Because these models under test provide their results in text form, it is natural to apply formal method approaches (e.g., satisfiability modulo theory, SMT, solvers) to verify the satisfaction of generated plans. For example, Jha et al. [55] prompt an LLM planner with a problem formulated with first-order constraints to predict a set of actions to complete the task. The output plan is input to an SMT solver to check for any infeasibilities in the program, and any counterexamples found are used to iteratively update the prompt and generate new plans. This counterexample approach is much faster than relying on combinatorial search methods that find a plan from scratch. However, the quality of generated plans and the number of iterations before a successful plan is generated are heavily dependent on the LLM generator itself, with similar reasons to the proxy-model used by Manakul et al. [85]. In particular, the authors explore the capability and efficiency of state-of-the-art large language models to solve block-world planning tasks [41]. For every experiment, each LLM is fed the initial random setup of a finite number of blocks in a scene and the desired goal setup in the form of first-order constraints. Inoperable plans are detected by the Z3 SMT solver [27], who iteratively works with the LLM to approach a feasible solution using counterexamples.

(2) Another work from Hu et al. [48] develops a RoboEval benchmark to test generated plans on real robots in a black-box manner. Like Wang et al. [136], the authors introduce their own extension of LTL formulations, known as RTL, which specifies temporal logic at a higher, scenario-specific, level while abstracting away constraints that are not dependent on available robot skills. RTL and LTL-NL are easier to read and define than classic LTL methods. RoboEval utilizes the provided RTL formulation of a problem, a simulator, and evaluator to systematically check whether the output meets requested goals. Furthermore, to check for robustness of the model to varied instructions, Hu et al. hand-engineer paraphrased sentences within an offline dataset that should ideally result in the same task completion. Primary causes of failures were found to be a result of generated code syntax errors, attempting to execute infeasible actions on the robot, and failing RTL checks.

Like adversarial prompting approaches, testing generated plans on robots in diverse scenarios enable researchers to design more robust systems that hallucinate less frequently at test-time.

5 Guidelines on Current Methodologies

In Section 4, we present a taxonomy of hallucination detection and mitigation algorithms in various deployment settings. Combining our findings from our extensive review, we now present guidelines for choosing hallucination intervention algorithms and metrics for different environments. As such, we hope these guidelines will enable developers to follow a standardized procedure to define hallucinations for their deployment context, design intervention algorithms, and evaluate efficacy before integrating hallucination-prone models into systems with humans at risk. Even while the field of LVLMs evolves rapidly, we argue our guidelines are general enough to continue to assist researchers in the near future. Additional considerations when using deep learning methods are discussed in Appendix E.1.

5.1 Process of Choosing and Integrating Intervention Algorithms

We split the design process of hallucination detection and mitigation methods into nine steps, shown in Figure 2.

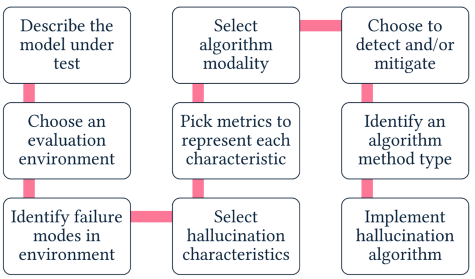

Fig. 2. Design process of hallucination intervention methods.

Describe the Model under Test. The model under test is the specific L(V)LM that has the potential to hallucinate in a deployment environment. Describing the model requires understanding its space of inputs and outputs and whether designers have access to model weights and/or generation probability distributions. Understanding the model under test is important for narrowing down possible hallucination intervention methods.

Choosing an Evaluation Setting. Evaluation settings are datasets, simulators, or real-world environments where the model under test and hallucination intervention algorithm are being tested hand-in-hand. Several possible evaluation settings are described in Appendix D, categorized by application area (e.g., image captioning, conversational QA, control, code generation). Developers should choose an evaluation setting that most closely resembles the intended deployment area to minimize the sim-to-real gap between evaluation and final integration [44, 112, 161]. Too large of a sim-to-real gap results in a poor understanding of the true capabilities of the deployed model. However, designers need to balance the tradeoffs between the cost of developing a new high-fidelity simulator or collecting representative offline data and using off the shelf low-fidelity environments or datasets that may not truly cover the distribution of the intended deployment area.

Identifying Failure Modes in Evaluation Setting. Failure modes are directly related to the characteristics of hallucinations that need to be detected or mitigated. These modes are identified in one of three primary ways. First, designers can choose failure modes based on the hard constraints of the evaluation setting. For example, in decision-making contexts, failure modes may represent a collision space that a robot should not enter, and in QA settings, failure modes could be defined by a set of ground-truth answers that a model generates conflicting results against. Second, stakeholders impacted by model decisions will have personal preferences for how the model under test should act in deployment. As such, the next set of failure modes are defined by acting outside of the range of desired behaviors. Finally, as discussed in Section 1, it is nearly impossible to account

for all possible failure modes at inference time. Thus, intervention algorithm designers should also query the L(V)LM under test with different inputs to identify other undesired and irrelevant generations. Furthermore, many models are released alongside a *model card* with details of failures found during its design stage [39, 88, 93], which developers can use to identify additional modes to consider. This final sampling procedure will not cover all possible remaining failure modes but will improve coverage rate. Through this three-step process, engineers gain an understanding of what behaviors should be considered as hallucinations in the particular evaluation setting. Although generalized detection/mitigation methods are preferred, identifying these specific subsets of hallucinations will allow engineers to quantitatively evaluate the efficacy of intervention approaches prior to deployment.

Selecting Hallucination Characteristics. Given that designers have selected a set of failure modes, or hallucination behaviors, they can now categorize each mode as one of three content characteristics defined in Section 3.1: compliance, desirability, and relevancy. Examples of categorizing failure modes are shown in Table 2. It might be the case that one or more of the categories do not end up with any failure modes. In this case, remaining unmatched characteristics are deemed irrelevant to the development of the hallucination intervention algorithm at this stage. Finally, if stakeholders also care about cases that the L(V)LM is attempting to deceive them through syntactically plausible generations, then the plausibility characteristic is also relevant to hallucination detection or mitigation.

Picking Metrics per Characteristic. Intervention algorithm designers can now choose metrics to measure the efficacy of model generations and hallucination detection/mitigation methods in meeting the defined requirements of each characteristic. Metrics for each chosen characteristic will differ, depending on the deployment context and outputs of the model under test. As such, we provide examples of current metrics in literature categorized by characteristic and application area in Table 4 from our review in Section 4.

Select Intervention Algorithm Modality. By this point, designers have performed the prerequisites of describing the available information from the model under test, its limitations, and defining relevant hallucination characteristics and metrics. The remaining steps are actually developing the intervention algorithm. To do so, engineers need to choose which modality the proposed algorithm will fall under (i.e., white-, grey-, or black-box). As described in Section 4, possible modalities depends on the availability of access to necessary information from the model under test. It is also important to consider at this stage the desired flexibility of the intervention algorithm (i.e., the ease of integrating the algorithm to different test models and evaluation settings). For example, black-box methods are easier to deploy to proprietary models where access to weights and token probabilities is limited. However, if the designer only needs to intervene in the hallucination tendency of a particular open-source language model, then white-box intervention algorithms can be tuned to the limitations of that specific model.

Intervention Type. Next, engineers can choose whether the intervention algorithm should detect and/or mitigate hallucinations. Again, this choice depends on the intended deployment area of the model under test and the needs of the stakeholders in the model design process. As seen in Table 3, we find several existing algorithms enable both detection and mitigation of hallucinations. Detection of hallucinations enables reacting to failures, while pure mitigation relies on the efficacy of the intervention algorithm alone to proactively filter hallucinations prior to final generation. As such, we recommend safety-critical scenarios be deployed with detection and mitigation algorithms to simultaneously reduce chances of failures and inform impacted parties of potential hallucinations to increase transparency.

Table 4. Common Datasets/Simulators and Metrics across Settings and Hallucination Characteristics

Setting	Common Datasets/Simulators	Characteristic			
		Compliance	Desirability	Relevancy	Plausibility
QA	— TriviaQA [58], — GSM8K [23], — HotpotQA [147], — MMLU [46], — Natural Questions [66]	— Accuracy† [4, 29, 32, 34, 65, 85, 91, 142] [12, 19, 72, 96, 150, 153] [100, 158], — Contradictions [29, 32, 34, 91, 96, 150], — Concept Probabilities [85, 104, 132], — BERTScore [85, 100, 108], — BLEU [72, 100, 108], — Adversarial Success Rate [148, 153], — Coverage [65, 158], — METEOR [100, 108], — SummaC [34], — FactScore [29], — Cosine Similarity [150], — Sentence-BERT [153], — AlignScore [108]	— Calibration Error [77, 142], — Succinctness [100, 122], — Over-conservativeness [145], — # of Clarifying Questions [158], — Prudence [145], — Preservation [12], — TOXIGEN Classifier [87]	— Perplexity [91, 122], — Cross Encoder [12]	— ROUGE [100, 104, 108], — Semantic Preciseness [19, 122]
IC	— MSCOCO [79], — GQA [52], — Visual Genome [64], — A-OKVQA [116]	— LVLM-based Scoring [50, 73, 164], — CHAIR [50, 73, 164], — POPE† [50, 73, 164], — Co-occurence [164], — Concept Probabilities [164], — BLEU [164], — BERTScore [164], — CLIP Score [164], — METEOR [164]	— CIDER Human Alignment [164], — Detailedness [50]	— CHAIR [50, 73, 164], — POPE† [50, 73, 164]	— Perplexity [50], — ROUGE [164], — SPICE [164]
IG	— MSCOCO [79], — LSUN [152], — LAION-400M [115], — I2P [113]	— Contradictions [22, 76], — Accuracy† [22], — LVLM-based Scoring [22], — CLIP Score [113]	— NudeNet Classifier [87, 113], — Q16 Classifier [87, 113], — Diversity [87], — Monetary Cost [22], — Bias [113], — Inappropriate Probability [113], — Expected Inappropriateness [113], — COCO FID-30k Fidelity [113]	— LVLM-based Scoring [22], — CLIP Score [113]	— Factual Fabrications [76]
P	— RoboEval [48], — SaGC [98], — Ravens [154], — BabyAI [20], — TableSim [110]	— Feasibility [43, 55, 75, 98, 110, 136], — Action Probabilities [43, 75, 110], — Plan/Action Accuracy† [48, 98, 136], — Action Variance [113], — Contradictions [55]	— Success Rate [16, 43, 48, 55, 75, 98, 110, 136], — Generalizability [16, 43, 75, 98], — Clarification/Unsure Rate [75, 98, 110, 136], — # of Attempts [16, 48, 55], — Calibration Error [75, 110], — Coverage [75, 110], — Estimated Payoff [16, 43], — Action Set Size [75, 110], — Plan Readability [136], — Plan Length [43], — Unsafe Action Rate [75], — Monetary Cost [75], — Inference Speed [136]	— Action Probabilities [43, 75, 110]	— Non-compliance Rate [75], — Preciseness [75]

†Encompasses machine metrics such as (balanced) accuracy, precision, recall, F1, AUC, exact match [14], and pass@1 [17].

Each metric is accompanied by references that use the metric as the specified characteristic. The same metric could be applied as different characteristics.

Identifying an Intervention Sub-type. Now that the deployment setting, modality, and intervention type have been identified, engineers can choose a method type from Table 3 that falls within those constraints. This step will require some experimentation across method types and specific algorithms to identify the most effective intervention approach using the previously chosen metrics. We list pros and cons of each method type in Table 5 to assist researchers with narrowing down the scope of their algorithm search.

Table 5. Benefits and Limitations of Each Intervention Algorithm Type

Method Type	Pros	Cons
Hidden States	— tuned for specific model — hidden states hold useful embeddings — no need to re-embed text output	— reduced model transfer flexibility — not applicable for proprietary models
Attention Weights	— tuned for specific model — attention weights hold useful info	— reduced model transfer flexibility — not applicable for proprietary models
Honesty Alignment	— directly fine-tunes model under test — empirically generalizes to new data	— tuned model still susceptible — test efficacy impacted by data quality
Concept Probabilities	— generally model-agnostic — intuitive — tried in broadest set of deployments	— requires access to token probabilities — correlation may not necessarily hold
Conformal Prediction	— theoretical guarantees — applicable to multi-step planning — provides model-uncertainty metric	— requires access to token probabilities — requires collecting calibration dataset — relies on human intervention
Analyzing Samples	— applicable to proprietary models — intuitive — removing conflicts reduces failures	— efficiency impacted by # of samples — affected by compliance metric choice
Adversarial Prompting	— applicable to proprietary models — reveals undesired behaviors — can lead to better filters — red-teaming often occurs pre-release	— requires covering large input space — generally does not mitigate — requires hallucination classifier
Proxy Model	— applicable to proprietary models — simple classifier could work well	— proxy has mismatched distribution — dependent on proxy complexity — LVLM proxy could hallucinate
Grounding Knowledge	— applicable to proprietary models — provides evidence for responses — aligns model knowledge with users' — can help to learn policies faster	— requires ground-truth database — could reference stale knowledge — still fails in low-data regimes
Constraint Satisfaction	— applicable to proprietary models — theoretical guarantees — SMT solvers find failure cases quickly	— requires precise constraint definitions — specification may be hard to parse

Implementing the Hallucination Intervention Algorithm. Finally, designers can implement and integrate a chosen algorithm into the specific deployment setting and perform additional tests to measure its efficacy in detecting/mitigating hallucinations from the model under test.

6 Future Directions

Here, we discuss some possible future directions in hallucination detection and mitigation techniques for foundation models to improve deployments to decision-making tasks.

Evaluating Methods on Decision-making Tasks. Most hallucination detection approaches are currently tested in offline QA settings for information retrieval or knowledge alignment, as seen in Table 3. As foundation models are increasingly used for more complex tasks, researchers should make an effort to adapt and evaluate earlier detection/mitigation approaches that were applied to QA problems. Although dissimilar in practice from QA settings, planning and control problems may be formulated such that earlier mitigation methods can be evaluated on decision-making tasks. For example, as discussed in Section 2.1, Chen et al. [15] treat the autonomous driving task as a QA problem, which could be naturally extended to test other QA hallucination detection methods

in the same setting. This evaluation may lead to greater understanding of the general limitations of these models, as we draw parallels across diverse deployments.

Development of More Black-box Approaches. White- and grey-box detection methods may not generally be applicable in situations where the internal state or token probabilities are unavailable from the language model. Thus, we predict black-box approaches will take precedence in the near future, as state-of-the-art LVLMs like GPT-4V already prohibit access to probability outputs. However, current black-box methods are limited with simplistic sampling techniques to gauge uncertainty, and proxy models may not be representative of the true state of the model under test. Works like FLIRT showcase the promise of black-box adversarial prompting approaches in generating undesirable results [87]. We argue developing more aggressive black-box adversarial generative models, which explicitly optimize for producing inputs that may perturb the system outputs, is key to identifying the limits of a foundation model's knowledge.

Pushing Models' Generalization Capabilities. Currently, foundation models are primarily deployed to decision-making tasks that likely have some relation to its training set. For example, although complex, tasks such as multi-agent communication, autonomous driving, and code generation will be present in training datasets. However, dynamic environments like robot crowd navigation require identifying nuances in pedestrian behaviors that the model may not have explicitly seen during training. Thus, when models are deployed in decision-making contexts and encounter a previously unseen scenario, or they are utilized in a completely different setting from their training data, it is necessary to consider their generalization capabilities. As discussed in Section 2.1, existing examples of foundation models applied in autonomous driving and robotics utilize external tools to retrieve sensor data or memories before planning. Mialon et al. [89] refer to such models that extract useful information from databases as augmented language models. As such, the authors argue that the combined efforts of using external tools and internal reasoning is critical to the generalizability of language models to broader tasks—also explored by Chen et al. [16], Park et al. [99], and Wang et al. [134]. From another perspective, Tong et al. [127] approach the development of generalized multi-modal large language models from a vision-centric focus. In particular, the authors find that training LVLMs with heavy consideration on the design of vision encoders and the respective connector between the vision and language models drastically improves the capabilities of architectures deployed in vision tasks (e.g., image captioning, QA, depth ordering). This line of thinking can bolster the performance of LVLMs deployed in autonomous driving decision-making, where current methods have failed [139]. Before integrating models into real-world applications, we argue that designers should thoroughly explore their generalization limitations to find directions for future growth and to maximize transparency of model capabilities.

Testing Multi-modal Models. With the explosion of LVLMs, which allow for explicit grounding of natural language and vision modalities, further exploration should be performed in evaluating their effectiveness in decision-making systems. Wen et al. [139] take a step in the right direction towards testing black-box LVLMs in offline driving scenarios, but there is still work to be done in deploying these models in online settings. This direction can shed light on the long-standing debate of whether modular or end-to-end systems should be preferred in a particular deployment setting. In fact, while our work has focused on LVLMs, there exist other families of multi-modal foundation models for the audio [95] and 3D generation [18] spaces, which similarly hallucinate [111, 135] and should be evaluated before deployment.

Summary. We provide a glimpse into the progress of research for evaluating hallucinations of foundation models for decision-making problems. First, we identify existing use cases of foundation models in decision-making applications (e.g., autonomous driving, robotics) and find

several works make note of undesired hallucinated generations in practice. By referencing works that encounter hallucinations across diverse domains, we provide a flexible definition for hallucinations that researchers can leverage, regardless of their deployment scenario. Then, we give a taxonomy of existing hallucination detection and mitigation approaches for decision-making, question-answering, and so on, alongside a list of commonly used metrics, datasets, and simulators for evaluation. We find that existing methods range in varying assumptions of inputs and evaluation settings, and we believe there is room for growth in general, black-box hallucination detection algorithms for foundation models. Finally, we present generalized guidelines to assist engineers with selecting hallucination intervention algorithms across varied deployment contexts and suggest future research directions.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. GPT-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).

[2] Raviteja Anantha, Svitlana Vakulenko, Zhucheng Tu, Shayne Longpre, Stephen Pulman, and Srinivas Chappidi. 2021. Open-domain question answering goes conversational via question rewriting. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 520–534.

[3] Anastasios N. Angelopoulos, Stephen Bates, Emmanuel J. Candès, Michael I. Jordan, and Lihua Lei. 2022. Learn then test: Calibrating predictive algorithms to achieve risk control. *arXiv preprint arXiv:2110.01052* (2022).

[4] Amos Azaria and Tom Mitchell. 2023. The internal state of an LLM knows when it's lying. In *Findings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 967–976.

[5] Zechen Bai, Pichao Wang, Tianjun Xiao, Tong He, Zongbo Han, Zheng Zhang, and Mike Zheng Shou. 2024. Hallucination of multimodal large language models: A survey. *arXiv preprint arXiv:2404.18930* (2024).

[6] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. 2022. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258* (2022).

[7] Rakesh P. Borase, D. K. Maghade, S. Y. Sondkar, and S. N. Pawar. 2021. A review of PID control, tuning methods and applications. *Int. J. Dynam. Contr.* 9 (2021), 818–827.

[8] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. In *Proceedings of the Conference on Neural Information Processing Systems*. Curran Associates, Inc., 1877–1901.

[9] Shulin Cao, Jiaxin Shi, Liangming Pan, Lunyiu Nie, Yutong Xiang, Lei Hou, Juanzi Li, Bin He, and Hanwang Zhang. 2022. KQA Pro: A dataset with explicit compositional programs for complex question answering over knowledge base. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 6101–6119.

[10] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jegou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. 2021. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF Conference on International Conference on Computer Vision (ICCV'21)*. Institute of Electrical and Electronics Engineers, 9630–9640.

[11] Neeloy Chakraborty, Aamir Hasan, Shuijing Liu, Tianchen Ji, Weihang Liang, D. Livingston McPherson, and Katherine Driggs-Campbell. 2023. Structural attention-based recurrent variational autoencoder for highway vehicle anomaly detection. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 1125–1134.

[12] Anthony Chen, Panupong Pasupat, Sameer Singh, Hongrae Lee, and Kelvin Guu. 2023. PURR: Efficiently editing language model hallucinations by denoising language model corruptions. *arXiv preprint arXiv:2305.14908* (2023).

[13] Canyu Chen and Kai Shu. 2024. Can LLM-generated misinformation be detected? In *Proceedings of the 12th International Conference on Learning Representations*.

[14] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 1870–1879.

[15] Long Chen, Oleg Sinavski, Jan Hünermann, Alice Karnsund, Andrew James Willmott, Danny Birch, Daniel Maund, and Jamie Shotton. 2024. Driving with LLMs: Fusing object-level vector modality for explainable autonomous driving. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'24)*. Institute of Electrical and Electronics Engineers, 14093–14100.

[16] Liting Chen, Lu Wang, Hang Dong, Yali Du, Jie Yan, Fangkai Yang, Shuang Li, Pu Zhao, Si Qin, Saravan Rajmohan, et al. 2023. Introspective tips: Large language model for in-context decision making. *arXiv preprint arXiv:2305.11598* (2023).

[17] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374* (2021).

[18] Sijin Chen, Xin Chen, Anqi Pang, Xianfang Zeng, Wei Cheng, Yijun Fu, Fukun Yin, Zhibin Wang, Jingyi Yu, Gang Yu, et al. 2024. MeshXL: Neural coordinate field for generative 3D foundation models. In *Proceedings of the Conference on Neural Information Processing Systems*.

[19] Yuyan Chen, Qiang Fu, Yichen Yuan, Zhihao Wen, Ge Fan, Dayiheng Liu, Dongmei Zhang, Zhixu Li, and Yanghua Xiao. 2023. Hallucination detection: Robustly discerning reliable answers in large language models. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. Association for Computing Machinery, New York, NY, USA, 245–255.

[20] Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. 2019. BabyAI: First steps towards grounded language learning with a human in the loop. In *Proceedings of the 7th International Conference on Learning Representations*.

[21] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. PaLM: Scaling language modeling with pathways. *J. Mach. Learn. Res.* 24, 240 (2023), 11324–11436.

[22] Zhixuan Chu, Lei Zhang, Yichen Sun, Siqiao Xue, Zhibo Wang, Zhan Qin, and Kui Ren. 2024. Sora detector: A unified hallucination detection for large text-to-video models. *arXiv preprint arXiv:2405.04180* (2024).

[23] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168* (2021).

[24] Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, et al. 2019. TextWorld: A learning environment for text-based games. In *Proceedings of the 7th Computer Games Workshop at the 27th International Conference on Artificial Intelligence*. Springer International Publishing, 41–75.

[25] Can Cui, Yunsheng Ma, Xu Cao, Wenqian Ye, Yang Zhou, Kaizhao Liang, Jintai Chen, Juanwu Lu, Zichong Yang, Kuei-Da Liao, et al. 2024. A survey on multimodal large language models for autonomous driving. In *Proceedings of the 1st Workshop on Large Language and Vision Models for Autonomous Driving at the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV'24)*. Institute of Electrical and Electronics Engineers, 958–979.

[26] Wenliang Dai, Zihan Liu, Ziwei Ji, Dan Su, and Pascale Fung. 2023. Plausible may not be faithful: Probing object hallucination in vision-language pre-training. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 2136–2148.

[27] Leonardo de Moura and Nikolaj Bjørner. 2008. Z3: An efficient SMT solver. In *Proceedings of the 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer Berlin, 337–340.

[28] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 4171–4186.

[29] Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason E. Weston. 2024. Chain-of-verification reduces hallucination in large language models. In *Proceedings of the Workshop on Reliable and Responsible Foundation Models at the 12th International Conference on Learning Representations*.

[30] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, Lei Li, and Zhifang Sui. 2023. A survey on in-context learning. *arXiv preprint arXiv:2301.00234* (2023).

[31] Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. 2023. PaLM-E: An embodied multimodal language model. In *Proceedings of the 40th International Conference on Machine Learning (Proceedings of Machine Learning Research,* Vol. 202*)*. PMLR, 8469–8488.

[32] Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. 2024. Improving factuality and reasoning in language models through multiagent debate. In *Proceedings of the 41st International Conference on Machine Learning (Proceedings of Machine Learning Research,* Vol. 235*)*. PMLR, 11733–11763.

[33] L. Ekenberg. 2000. The logic of conflicts between decision making agents. *J. Logic Computat.* 10, 4 (08 2000), 583–602.

[34] Mohamed Elaraby, Mengyin Lu, Jacob Dunn, Xueying Zhang, Yu Wang, Shizhu Liu, Pingchuan Tian, Yuping Wang, and Yuxuan Wang. 2023. Halo: Estimation and reduction of hallucinations in open-source weak large language models. *arXiv preprint arXiv:2308.11764* (2023).

[35] Simone Formentin, Klaske van Heusden, and Alireza Karimi. 2013. Model-based and data-driven model-reference control: A comparative analysis. In *Proceedings of the European Control Conference (ECC'13)*. Institute of Electrical and Electronics Engineers, 1410–1415.
[36] Wikimedia Foundation. [n. d.]. *Wikimedia Downloads*. Retrieved from https://dumps.wikimedia.org
[37] Markus Freitag and Yaser Al-Onaizan. 2017. Beam search strategies for neural machine translation. In *Proceedings of the 1st Workshop on Neural Machine Translation*. Association for Computational Linguistics, 56–60.
[38] Daocheng Fu, Xin Li, Licheng Wen, Min Dou, Pinlong Cai, Botian Shi, and Yu Qiao. 2024. Drive like a human: Rethinking autonomous driving with large language models. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW'24)*. Institute of Electrical and Electronics Engineers, 910–919.
[39] Jack Gallifant, Amelia Fiske, Yulia A. Levites Strekalova, Juan S. Osorio-Valencia, Rachael Parke, Rogers Mwavu, Nicole Martinez, Judy Wawira Gichoya, Marzyeh Ghassemi, Dina Demner-Fushman, et al. 2024. Peer review of GPT-4 technical report and systems card. *PLoS Digit. Health* 3, 1 (01 2024), 1–15.
[40] Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did Aristotle use a laptop? A question answering benchmark with implicit reasoning strategies. *Trans. Assoc. Computat. Ling.* 9 (04 2021), 346–361.
[41] Naresh Gupta and Dana S. Nau. 1992. On the complexity of blocks-world planning. *Artif. Intell.* 56, 2 (1992), 223–254.
[42] Frederick Hayes-Roth. 1985. Rule-based systems. *Commun. ACM* 28, 9 (Sep. 1985), 921–932.
[43] Rishi Hazra, Pedro Zuidberg Dos Martires, and Luc De Raedt. 2024. SayCanPay: Heuristic planning with large language models using learnable domain knowledge. In *Proceedings of the 38th AAAI Conference on Artificial Intelligence*. AAAI Press, 20123–20133.
[44] Haoran He, Peilin Wu, Chenjia Bai, Hang Lai, Lingxiao Wang, Ling Pan, Xiaolin Hu, and Weinan Zhang. 2024. Bridging the sim-to-real gap from the information bottleneck perspective. In *Proceedings of the 8th Annual Conference on Robot Learning*.
[45] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. 2018. Deep reinforcement learning that matters. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*. AAAI Press.
[46] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *Proceedings of the 9th International Conference on Learning Representations*.
[47] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Proceedings of the Conference on Neural Information Processing Systems*. Curran Associates, Inc.
[48] Zichao Hu, Francesca Lucchetti, Claire Schlesinger, Yash Saxena, Anders Freeman, Sadanand Modak, Arjun Guha, and Joydeep Biswas. 2024. Deploying and evaluating LLMs to program service mobile robots. *IEEE Robot. Autom. Lett.* 9, 3 (2024), 2853–2860.
[49] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. 2023. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *arXiv preprint arXiv:2311.05232* (2023).
[50] Qidong Huang, Xiaoyi Dong, Pan Zhang, Bin Wang, Conghui He, Jiaqi Wang, Dahua Lin, Weiming Zhang, and Nenghai Yu. 2024. OPERA: Alleviating hallucination in multi-modal large language models via over-trust penalty and retrospection-allocation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'24)*. Institute of Electrical and Electronics Engineers, 13418–13427.
[51] Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, Defu Lian, Yasheng Wang, Ruiming Tang, and Enhong Chen. 2024. Understanding the planning of LLM agents: A survey. *arXiv preprint arXiv:2402.02716* (2024).
[52] Drew A. Hudson and Christopher D. Manning. 2019. GQA: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'19)*. Institute of Electrical and Electronics Engineers, 6693–6702.
[53] Brian Ichter, Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, et al. 2023. Do as I can, not as I say: Grounding language in robotic affordances. In *Proceedings of the 6th Conference on Robot Learning (Proceedings of Machine Learning Research,* Vol. 205*)*. PMLR, 287–318.
[54] Joel Janai, Fatma Güney, Aseem Behl, and Andreas Geiger. 2020. Computer vision for autonomous vehicles: Problems, datasets and state of the art. *Found. Trends Comput. Graph. Vis.* 12, 1-3 (2020), 1–308.
[55] Sumit Kumar Jha, Susmit Jha, Patrick Lincoln, Nathaniel D. Bastian, Alvaro Velasquez, Rickard Ewetz, and Sandeep Neema. 2023. Counterexample guided inductive synthesis using large language models and satisfiability solving. In *Proceedings of the IEEE Military Communications Conference (MILCOM'23)*. Institute of Electrical and Electronics Engineers, 944–949.

[56] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *Comput. Surv.* 55, 12, Article 248 (Mar. 2023), 38 pages.

[57] Alistair E. W. Johnson, Tom J. Pollard, Seth J. Berkowitz, Nathaniel R. Greenbaum, Matthew P. Lungren, Chih-ying Deng, Roger G. Mark, and Steven Horng. 2019. MIMIC-CXR, a de-identified publicly available database of chest radiographs with free-text reports. *Scient. Data* 6, 1 (2019), 317.

[58] 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*. Association for Computational Linguistics, 1601–1611.

[59] Jungo Kasai, Keisuke Sakaguchi, Yoichi Takahashi, Ronan Le Bras, Akari Asai, Xinyan Velocity Yu, Dragomir Radev, Noah A. Smith, Yejin Choi, and Kentaro Inui. 2023. RealTime QA: What's the answer right now? In *Proceedings of the 37th Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

[60] Ronald Kemker, Marc McClure, Angelina Abitino, Tyler Hayes, and Christopher Kanan. 2018. Measuring catastrophic forgetting in neural networks. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*. AAAI Press, 3390–3398.

[61] Ali Keysan, Andreas Look, Eitan Kosman, Gonca Gürsun, Jörg Wagner, Yu Yao, and Barbara Rakitsch. 2023. Can you text what is happening? Integrating pre-trained language encoders into trajectory prediction models for autonomous driving. *arXiv preprint arXiv:2309.05282* (2023).

[62] Seokhwan Kim, Spandana Gella, Chao Zhao, Di Jin, Alexandros Papangelis, Behnam Hedayatnia, Yang Liu, and Dilek Z. Hakkani-Tur. 2023. Task-oriented conversational modeling with subjective knowledge track in DSTC11. In *Proceedings of the 11th Dialog System Technology Challenge*. Association for Computational Linguistics, 274–281.

[63] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, et al. 2023. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV'23)*. Institute of Electrical and Electronics Engineers, 3992–4003.

[64] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. 2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *Int. J. Comput. Vis.* 123 (2017), 32–73.

[65] Bhawesh Kumar, Charlie Lu, Gauri Gupta, Anil Palepu, David Bellamy, Ramesh Raskar, and Andrew Beam. 2023. Conformal prediction with large language models for multi-choice question answering. In *Proceedings of the "Neural Conversational AI Workshop - What's left to TEACH (Trustworthy, Enhanced, Adaptable, Capable and Human-centric) chatbots?" at the 40th International Conference on Machine Learning*.

[66] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: A benchmark for question answering research. *Trans. Assoc. Computat. Ling.* 7 (2019), 452–466.

[67] Minae Kwon, Sang Michael Xie, Kalesha Bullard, and Dorsa Sadigh. 2023. Reward design with language models. In *Proceedings of the the 11th International Conference on Learning Representations*.

[68] Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 1203–1213.

[69] Daliang Li, Ankit Singh Rawat, Manzil Zaheer, Xin Wang, Michal Lukasik, Andreas Veit, Felix Yu, and Sanjiv Kumar. 2023. Large language models with controllable working memory. In *Findings of the 61st Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 1774–1793.

[70] Haonan Li, Martin Tomko, Maria Vasardani, and Timothy Baldwin. 2022. MultiSpanQA: A dataset for multi-span question answering. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 1250–1260.

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

[72] Xingxuan Li, Ruochen Zhao, Yew Ken Chia, Bosheng Ding, Shafiq Joty, Soujanya Poria, and Lidong Bing. 2024. Chain-of-knowledge: Grounding large language models via dynamic knowledge adapting over heterogeneous sources. In *Proceedings of the 12th International Conference on Learning Representations*.

[73] Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Xin Zhao, and Ji-Rong Wen. 2023. Evaluating object hallucination in large vision-language models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 292–305.

[74] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. 2023. Code as policies: Language model programs for embodied control. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'23)*. Institute of Electrical and Electronics Engineers, 9493–9500.

[75] Kaiqu Liang, Zixu Zhang, and Jaime Fernández Fisac. 2024. Introspective planning: Guiding language-enabled agents to refine their own uncertainty. *arXiv preprint arXiv:2402.06529* (2024).
[76] Youngsun Lim and Hyunjung Shim. 2024. Addressing image hallucination in text-to-image generation through factual image retrieval. *arXiv preprint arXiv:2407.10683* (2024).
[77] Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Teaching models to express their uncertainty in words. *Trans. Mach. Learn. Res.* (2022), 1–19. https://openreview.net/pdf?id=8s8K2UZGTZ
[78] 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*. Association for Computational Linguistics, 3214–3252.
[79] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: Common objects in context. In *Proceedings of the 13th European Conference on Computer Vision (ECCV'14)*. Springer International Publishing, 740–755.
[80] Hanchao Liu, Wenyuan Xue, Yifei Chen, Dapeng Chen, Xiutian Zhao, Ke Wang, Liping Hou, Rongjun Li, and Wei Peng. 2024. A survey on hallucination in large vision-language models. *arXiv preprint arXiv:2402.00253* (2024).
[81] Jiaqi Liu, Peng Hang, Xiao Qi, Jianqiang Wang, and Jian Sun. 2023. MTD-GPT: A multi-task decision-making GPT model for autonomous driving at unsignalized intersections. In *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC'23)*. Institute of Electrical and Electronics Engineers, 5154–5161.
[82] Tianyu Liu, Yizhe Zhang, Chris Brockett, Yi Mao, Zhifang Sui, Weizhu Chen, and Bill Dolan. 2022. A token-level reference-free hallucination detection benchmark for free-form text Generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 6723–6737.
[83] Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. 2022. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *Proceedings of the Conference on Neural Information Processing Systems*. Curran Associates, Inc., 2507–2521.
[84] Chaitanya Malaviya, Peter Shaw, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2023. QUEST: A retrieval dataset of entity-seeking queries with implicit set operations. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 14032–14047.
[85] Potsawee Manakul, Adian Liusie, and Mark Gales. 2023. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 9004–9017.
[86] Jiageng Mao, Junjie Ye, Yuxi Qian, Marco Pavone, and Yue Wang. 2024. A language agent for autonomous driving. In *Proceedings of the 1st Conference on Language Modeling*.
[87] Ninareh Mehrabi, Palash Goyal, Christophe Dupuy, Qian Hu, Shalini Ghosh, Richard Zemel, Kai-Wei Chang, Aram Galstyan, and Rahul Gupta. 2023. FLIRT: Feedback loop in-context red teaming. *arXiv preprint arXiv:2308.04265* (2023).
[88] Meta. 2024. Model information. *GitHub*. Retrieved from https://github.com/meta-llama/llama-models/blob/main/models/llama3_1/MODEL_CARD.md
[89] Grégoire Mialon, Roberto Dessi, Maria Lomeli, Christoforos Nalmpantis, Ramakanth Pasunuru, Roberta Raileanu, Baptiste Roziere, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, et al. 2023. Augmented language models: A survey. *Trans. Mach. Learn. Res.* (2023), 1–35. https://openreview.net/pdf?id=jh7wH2AzKK
[90] 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 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 12076–12100.
[91] Niels Mündler, Jingxuan He, Slobodan Jenko, and Martin Vechev. 2024. Self-contradictory hallucinations of large language models: Evaluation, detection and mitigation. In *Proceedings of the 12th International Conference on Learning Representations*.
[92] OpenAI. 2022. Introducing ChatGPT. *OpenAI blog*. Retrieved from https://openai.com/blog/chatgpt
[93] OpenAI. 2023. DALL·E 3 system card. *OpenAI Blog*. Retrieved from https://cdn.openai.com/papers/DALL_E_3_System_Card.pdf
[94] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. 2024. DINOv2: Learning robust visual features without supervision. *Trans. Mach. Learn. Res.* (2024), 1–32. https://openreview.net/pdf?id=a68SUt6zFt
[95] Gerhard Paaß and Sven Giesselbach. 2023. *Foundation Models for Speech, Images, Videos, and Control*. Springer International Publishing, Cham, 313–382.
[96] Lorenzo Pacchiardi, Alex James Chan, Sören Mindermann, Ilan Moscovitz, Alexa Yue Pan, Yarin Gal, Owain Evans, and Jan M. Brauner. 2024. How to catch an AI liar: Lie detection in black-box LLMs by asking unrelated questions. In *Proceedings of the 12th International Conference on Learning Representations*.

[97] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 311–318.

[98] Jeongeun Park, Seungwon Lim, Joonhyung Lee, Sangbeom Park, Minsuk Chang, Youngjae Yu, and Sungjoon Choi. 2024. CLARA: Classifying and disambiguating user commands for reliable interactive robotic agents. *IEEE Robot. Autom. Lett.* 9, 2 (2024), 1059–1066.

[99] Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. Association for Computing Machinery, New York, NY, USA.

[100] Baolin Peng, Michel Galley, Pengcheng He, Hao Cheng, Yujia Xie, Yu Hu, Qiuyuan Huang, Lars Liden, Zhou Yu, Weizhu Chen, et al. 2023. Check your facts and try again: Improving large language models with external knowledge and automated feedback. *arXiv preprint arXiv:2302.12813* (2023).

[101] Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. 2018. VirtualHome: Simulating household activities via programs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'18)*. Institute of Electrical and Electronics Engineers, 8494–8502.

[102] Gokul Puthumanaillam, Manav Vora, Pranay Thangeda, and Melkior Ornik. 2024. A moral imperative: The need for continual superalignment of large language models. *arXiv preprint arXiv:2403.14683* (2024).

[103] Huachuan Qiu, Shuai Zhang, Anqi Li, Hongliang He, and Zhenzhong Lan. 2023. Latent jailbreak: A benchmark for evaluating text safety and output robustness of large language models. *arXiv preprint arXiv:2307.08487* (2023).

[104] Victor Quach, Adam Fisch, Tal Schuster, Adam Yala, Jae Ho Sohn, Tommi S. Jaakkola, and Regina Barzilay. 2024. Conformal language modeling. In *Proceedings of the 12th International Conference on Learning Representations*.

[105] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning (Proceedings of Machine Learning Research,* Vol. 139*)*. PMLR, 8748–8763.

[106] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*. Retrieved from https://openai.com/research/better-language-models

[107] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2383–2392.

[108] Anil Ramakrishna, Rahul Gupta, Jens Lehmann, and Morteza Ziyadi. 2023. INVITE: A testbed of automatically generated invalid questions to evaluate large language models for hallucinations. In *Findings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 5422–5429.

[109] Vipula Rawte, Amit Sheth, and Amitava Das. 2023. A survey of hallucination in large foundation models. *arXiv preprint arXiv:2309.05922* (2023).

[110] Allen Z. Ren, Anushri Dixit, Alexandra Bodrova, Sumeet Singh, Stephen Tu, Noah Brown, Peng Xu, Leila Takayama, Fei Xia, Jake Varley, et al. 2023. Robots that ask for help: Uncertainty alignment for large language model planners. In *Proceedings of the 7th Conference on Robot Learning (Proceedings of Machine Learning Research,* Vol. 229*)*. PMLR, 661–682.

[111] Pranab Sahoo, Prabhash Meharia, Akash Ghosh, Sriparna Saha, Vinija Jain, and Aman Chadha. 2024. A comprehensive survey of hallucination in large language, image, video and audio foundation models. In *Findings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11709–11724.

[112] Erica Salvato, Gianfranco Fenu, Eric Medvet, and Felice Andrea Pellegrino. 2021. Crossing the reality gap: A survey on sim-to-real transferability of robot controllers in reinforcement learning. *IEEE Access* 9 (2021), 153171–153187.

[113] Patrick Schramowski, Manuel Brack, Björn Deiseroth, and Kristian Kersting. 2023. Safe latent diffusion: Mitigating inappropriate degeneration in diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'23)*. Institute of Electrical and Electronics Engineers, 22522–22531.

[114] Andre Schreiber, Tianchen Ji, D. Livingston McPherson, and Katherine Driggs-Campbell. 2023. An attentional recurrent neural network for occlusion-aware proactive anomaly detection in field robot navigation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'23)*. Institute of Electrical and Electronics Engineers, 8038–8045.

[115] Christoph Schuhmann, Richard Vencu, Romain Beaumont, Robert Kaczmarczyk, Clayton Mullis, Aarush Katta, Theo Coombes, Jenia Jitsev, and Aran Komatsuzaki. 2021. LAION-400M: Open dataset of CLIP-filtered 400 million image-text pairs. *arXiv preprint arXiv:2111.02114* (2021).

[116] Dustin Schwenk, Apoorv Khandelwal, Christopher Clark, Kenneth Marino, and Roozbeh Mottaghi. 2022. A-OKVQA: A benchmark for visual question answering using world knowledge. In *Proceedings of the 17th European Conference on Computer Vision (ECCV'22)*. Springer Nature Switzerland, 146–162.

[117] Glenn Shafer and Vladimir Vovk. 2008. A tutorial on conformal prediction. *J. Mach. Learn. Res.* 9, 12 (2008), 371–421.

[118] Dhruv Shah, Błażej Osiński, Brian Ichter, and Sergey Levine. 2023. LM-Nav: Robotic navigation with large pre-trained models of language, vision, and action. In *Proceedings of the 6th Conference on Robot Learning (Proceedings of Machine Learning Research,* Vol. 205*)*. PMLR, 492–504.

[119] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. In *Proceedings of the Conference on Neural Information Processing Systems*. Curran Associates, Inc., 8634–8652.

[120] Chonghao Sima, Katrin Renz, Kashyap Chitta, Li Chen, Hanxue Zhang, Chengen Xie, Ping Luo, Andreas Geiger, and Hongyang Li. 2023. DriveLM: Driving with graph visual question answering. *arXiv preprint arXiv:2312.14150* (2023).

[121] Gonen Singer and Yuval Cohen. 2021. A framework for smart control using machine-learning modeling for processes with closed-loop control in Industry 4.0. *Eng. Applic. Artif. Intell.* 102 (2021), 104236.

[122] Da Song, Xuan Xie, Jiayang Song, Derui Zhu, Yuheng Huang, Felix Juefei-Xu, and Lei Ma. 2024. LUNA: A model-based universal analysis framework for large language models. *IEEE Trans. Softw. Eng.* 50, 7 (2024), 1921–1948.

[123] Mohsen Soori, Behrooz Arezoo, and Roza Dastres. 2023. Artificial intelligence, machine learning and deep learning in advanced robotics, a review. *Cogn. Robot.* 3 (2023), 54–70.

[124] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Trans. Mach. Learn. Res.* (2023), 1–95. https://openreview.net/pdf?id=uyTL5Bvosj

[125] Naoki Suganuma and Keisuke Yoneda. 2022. Current status and issues of traffic light recognition technology in autonomous driving system. *IEICE Trans. Fundam. Electron., Commun. Comput. Sci.* E105.A, 5 (2022), 763–769.

[126] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: A large-scale dataset for fact extraction and VERification. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 809–819.

[127] Shengbang Tong, Ellis Brown, Penghao Wu, Sanghyun Woo, Manoj Middepogu, Sai Charitha Akula, Jihan Yang, Shusheng Yang, Adithya Iyer, Xichen Pan, et al. 2024. Cambrian-1: A fully open, vision-centric exploration of multimodal LLMs. *arXiv preprint arXiv:2406.16860* (2024).

[128] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).

[129] 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 arXiv:2307.09288* (2023).

[130] Priyansh Trivedi, Gaurav Maheshwari, Mohnish Dubey, and Jens Lehmann. 2017. LC-QuAD: A corpus for complex question answering over knowledge graphs. In *Proceedings of the International Semantic Web Conference*. Springer International Publishing, 210–218.

[131] Cem Uluoglakci and Tugba Temizel. 2024. HypoTermQA: Hypothetical terms dataset for benchmarking hallucination tendency of LLMs. In *Proceedings of the Student Research Workshop at the 18th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 95–136.

[132] Neeraj Varshney, Wenlin Yao, Hongming Zhang, Jianshu Chen, and Dong Yu. 2023. A stitch in time saves nine: Detecting and mitigating hallucinations of LLMs by validating low-confidence generation. *arXiv preprint arXiv:2307.03987* (2023).

[133] Boxin Wang, Weixin Chen, Hengzhi Pei, Chulin Xie, Mintong Kang, Chenhui Zhang, Chejian Xu, Zidi Xiong, Ritik Dutta, Rylan Schaeffer, et al. 2023. DecodingTrust: A comprehensive assessment of trustworthiness in GPT models. In *Proceedings of the 37th Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

[134] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2024. Voyager: An open-ended embodied agent with large language models. *Trans. Mach. Learn. Res.* (2024), 1–44. https://openreview.net/pdf?id=ehfRiF0R3a

[135] Hongbo Wang, Jie Cao, Jin Liu, Xiaoqiang Zhou, Huaibo Huang, and Ran He. 2024. Hallo3D: Multi-modal hallucination detection and mitigation for consistent 3D content generation. In *Proceedings of the Conference on Neural Information Processing Systems*.

[136] Jun Wang, Jiaming Tong, Kaiyuan Tan, Yevgeniy Vorobeychik, and Yiannis Kantaros. 2024. Conformal temporal logic planning using large language models. *arXiv preprint arXiv:2309.10092* (2024).

[137] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the Conference on Neural Information Processing Systems*. Curran Associates, Inc., 24824–24837.

[138] Licheng Wen, Daocheng Fu, Xin Li, Xinyu Cai, Tao M. A., Pinlong Cai, Min Dou, Botian Shi, Liang He, and Yu Qiao. 2024. DiLu: A knowledge-driven approach to autonomous driving with large language models. In *Proceedings of the 12th International Conference on Learning Representations*.

[139] Licheng Wen, Xuemeng Yang, Daocheng Fu, Xiaofeng Wang, Pinlong Cai, Xin Li, Tao M. A., Yingxuan Li, Linran XU, Dengke Shang, et al. 2024. On the road with GPT-4V(ision): Explorations of utilizing visual-language model as autonomous driving agent. In *Proceedings of the Workshop on Large Language Model (LLM) Agents at the 12th International Conference on Learning Representations*.

[140] Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.* 8, 3–4 (May 1992), 229–256.

[141] Dongming Wu, Wencheng Han, Tiancai Wang, Yingfei Liu, Xiangyu Zhang, and Jianbing Shen. 2023. Language prompt for autonomous driving. *arXiv preprint arXiv:2309.04379* (2023).

[142] Miao Xiong, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Fu, Junxian He, and Bryan Hooi. 2024. Can LLMs express their uncertainty? An empirical evaluation of confidence elicitation in LLMs. In *Proceedings of the 12th International Conference on Learning Representations*.

[143] Zhenhua Xu, Yujia Zhang, Enze Xie, Zhen Zhao, Yong Guo, Kwan-Yee K. Wong, Zhenguo Li, and Hengshuang Zhao. 2024. DriveGPT4: Interpretable end-to-end autonomous driving via large language model. *IEEE Robot. Autom. Lett.* 9, 10 (2024), 8186–8193.

[144] Jingfeng Yang, Hongye Jin, Ruixiang Tang, Xiaotian Han, Qizhang Feng, Haoming Jiang, Shaochen Zhong, Bing Yin, and Xia Hu. 2024. Harnessing the power of LLMs in practice: A survey on ChatGPT and beyond. *ACM Trans. Knowl. Discov. Data* 18, 6 (2024), 1–32.

[145] Yuqing Yang, Ethan Chern, Xipeng Qiu, Graham Neubig, and Pengfei Liu. 2024. Alignment for honesty. In *Proceedings of the Conference on Neural Information Processing Systems*.

[146] Zhenjie Yang, Xiaosong Jia, Hongyang Li, and Junchi Yan. 2024. LLM4Drive: A survey of large language models for autonomous driving. In *Proceedings of the Workshop on Open-world Agents at the Conference on Neural Information Processing Systems*.

[147] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2369–2380.

[148] Jia-Yu Yao, Kun-Peng Ning, Zhen-Hui Liu, Mu-Nan Ning, and Li Yuan. 2023. LLM lies: Hallucinations are not bugs, but features as adversarial examples. *arXiv preprint arXiv:2310.01469* (2023).

[149] Hongbin Ye, Tong Liu, Aijia Zhang, Wei Hua, and Weiqiang Jia. 2023. Cognitive mirage: A review of hallucinations in large language models. *arXiv preprint arXiv:2309.06794* (2023).

[150] Yakir Yehuda, Itzik Malkiel, Oren Barkan, Jonathan Weill, Royi Ronen, and Noam Koenigstein. 2024. InterrogateLLM: Zero-resource hallucination detection in LLM-generated answers. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 9333–9347.

[151] Koichiro Yoshino, Chiori Hori, Julien Perez, Luis Fernando D'Haro, Lazaros Polymenakos, Chulaka Gunasekara, Walter S. Lasecki, Jonathan K. Kummerfeld, Michel Galley, Chris Brockett, et al. 2019. Dialog system technology challenge 7. *arXiv preprint arXiv:1901.03461* (2019).

[152] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. 2015. LSUN: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365* (2015).

[153] Xiaodong Yu, Hao Cheng, Xiaodong Liu, Dan Roth, and Jianfeng Gao. 2024. ReEval: Automatic hallucination evaluation for retrieval-augmented large language models via transferable adversarial attacks. In *Findings of the Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 1333–1351.

[154] Andy Zeng, Pete Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Vikas Sindhwani, et al. 2021. Transporter networks: Rearranging the visual world for robotic manipulation. In *Proceedings of the 4th Conference on Robot Learning (Proceedings of Machine Learning Research,* Vol. 155*)*. PMLR, 726–747.

[155] Fanlong Zeng, Wensheng Gan, Yongheng Wang, Ning Liu, and Philip S. Yu. 2023. Large language models for robotics: A survey. *arXiv preprint arXiv:2311.07226* (2023).

[156] Ceng Zhang, Junxin Chen, Jiatong Li, Yanhong Peng, and Zebing Mao. 2023. Large language models for human-robot interaction: A review. *Biomim. Intell. Robot.* 3, 4 (2023), 100131.

[157] Muru Zhang, Ofir Press, William Merrill, Alisa Liu, and Noah A. Smith. 2024. How language model hallucinations can snowball. In *Proceedings of the 41st International Conference on Machine Learning (Proceedings of Machine Learning Research,* Vol. 235*)*. PMLR, 59670–59684.

[158] Shuo Zhang, Liangming Pan, Junzhou Zhao, and William Yang Wang. 2024. The knowledge alignment problem: Bridging human and external knowledge for large language models. In *Findings of the 62nd Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2025–2038.

[159] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating text generation with BERT. In *Proceedings of the 8th International Conference on Learning Representations*.

[160] Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, et al. 2023. Siren's song in the AI Ocean: A survey on hallucination in large language models. *arXiv preprint arXiv:2309.01219* (2023).

[161] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. 2020. Sim-to-real transfer in deep reinforcement learning for robotics: A survey. In *Proceedings of the IEEE Symposium Series on Computational Intelligence (SSCI'20)*. 737–744.

[162] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223* (2023).

[163] Ce Zhou, Qian Li, Chen Li, Jun Yu, Yixin Liu, Guangjing Wang, Kai Zhang, Cheng Ji, Qiben Yan, Lifang He, et al. 2023. A comprehensive survey on pretrained foundation models: A history from BERT to ChatGPT. *arXiv preprint arXiv:2302.09419* (2023).

[164] Yiyang Zhou, Chenhang Cui, Jaehong Yoon, Linjun Zhang, Zhun Deng, Chelsea Finn, Mohit Bansal, and Huaxiu Yao. 2024. Analyzing and mitigating object hallucination in large vision-language models. In *Proceedings of the 12th International Conference on Learning Representations*.

Received 29 April 2024; revised 13 January 2025; accepted 3 February 2025