The Learnability of an Unknown System From Input-Output Data

Anonymous authors
Paper under double-blind review

Abstract

Artificial intelligence is transforming how scientists build models, test ideas, and make predictions. Beneath these advances lies a fundamental question: from the data we collect, when is learning possible in principle, and when is it not? We cast inference as a game between a learner, who holds a pool of candidate models, and an adversary, who holds the unknown ground-truth system. The learner observes the system and selects a candidate model to achieve one of three goals: identify the system, predict its output, or verify an input. We analyze 81 cases that arise by varying these goals, the available observations from both ground truth and candidates, and whether systems are single- or multi-valued. For each case, we prove whether universal solvability is possible. By clarifying which observations make success achievable in principle, our results explain why certain data-driven problems are solvable and guide how to collect data and evaluate models.

1 Introduction

Many data-driven inference problems in science are about identifying, predicting, or verifying unknown systems given input-output data. For example, partial differential equation (PDE) discovery is about identifying an underlying governing differential equation that maps forcing functions and initial conditions to a system's response (Brunton et al., 2016; Rudy et al., 2017; Schaeffer, 2017); in early warning earthquake systems, we want to predict the map from ground tremors to the next location and magnitude of an earthquake (Al Banna et al., 2020; Allen et al., 2009; Mousavi et al., 2020); in automated theorem proving, we desire a verifiable map from a mathematical statement to either a valid proof or identifying it as false (Harrison, 2009; Kaliszyk & Urban, 2014). In this paper, we call any map related to an unknown system a ground-truth map and denote it by $\mathcal{M}_*$. These ground-truth maps can be linear, nonlinear, single-valued, set-valued, or may even return the empty set for some inputs.

For any ground-truth map, we usually have one of the following goals in data-driven inference: (1) Map identification, in which the underlying governing map is sought that determines how inputs go to outputs (Rudy et al., 2017), (2) Map prediction, where one wants to take new inputs and predict the map's output (Kovachki et al., 2023), or (3) Map verification, where one only wants to predict the map's output for a given input but also verify if an input is valid for the map (Hendrycks et al., 2020). Map identification is usually thought of as figuring out exactly what the map is, e.g., using data to write down a differential equation to write down a governing equation. Whereas, map prediction is about predicting the map's output, and map verification is for applications with a demand for certificates of success. For example, PDE discovery has the goal of map identification as one wants to write down the underlying PDE, earthquake systems have a map prediction goal as one hopes to take ground tremors to earthquakes (without necessarily understanding the governing equations), and proof generators are more about map verification. A practical use of map verification is input validation, where we wish to determine whether an input belongs to the admissible domain of the system. For proof generators, we want the system to be able to identify a false statement.

The ground-truth map that one wishes to either identify, predict, or verify can come from many sources. In PDE discovery, we often start with a dictionary of possible terms for the equations, giving us a set of candidate differential equations (Brunton et al., 2016; Schaeffer, 2017). In machine learning (ML), we want

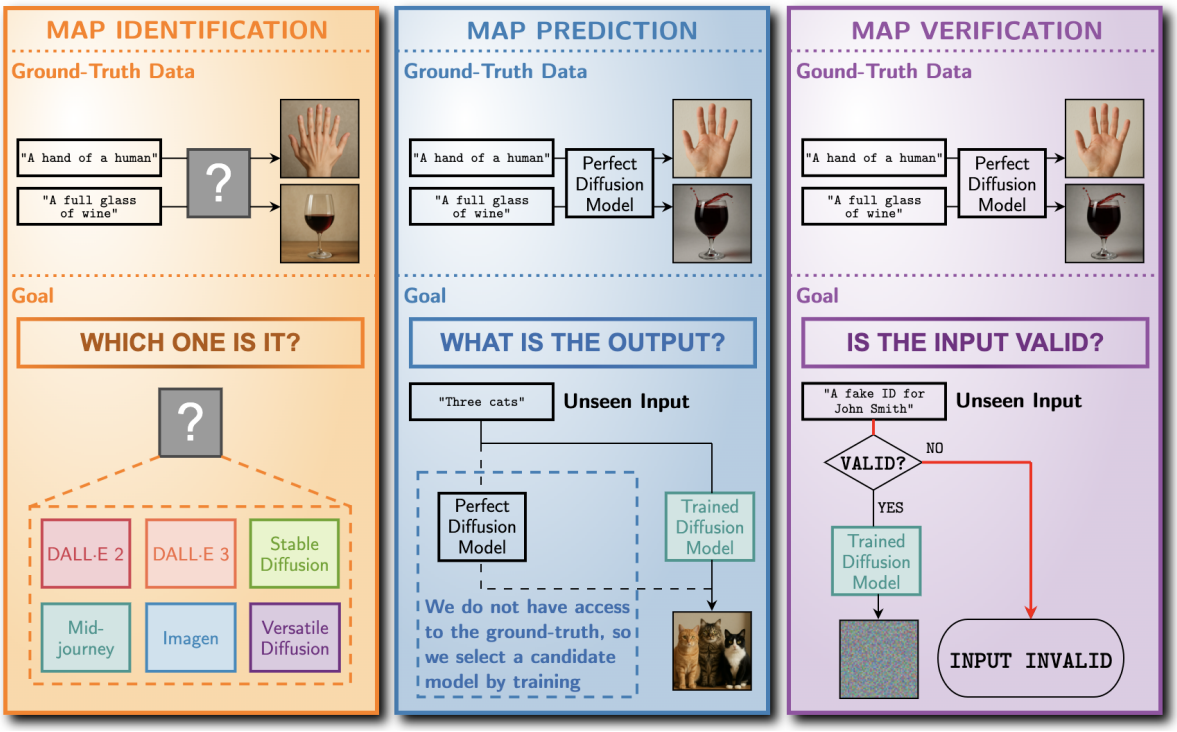

Figure 1: **Three core goals in data-driven inference, illustrated with text-to-image generation.** Each panel shows a different interaction with an unknown ground-truth map taking texts to images. Map identification (left) aims to recover the explicit mapping rules, such as identifying the ground-truth generative model producing the images from prompts. Map prediction (middle) only aims to mimic the behavior of the ground-truth map for new, previously unseen prompts. Map verification (right) is about checking whether a given image can be generated from a given text prompt, useful for input validation.

to replace the ground-truth map by a representation or approximation, which is easier to store, apply, or simulate such as a neural network. In this paper, we call these candidate models. When we train a neural network to predict a future earthquake, the candidate models are all possible neural networks that can be parameterized by the trainable parameters. For proof generators, the set of candidate models could be the class of symbolic rule-based engines to take statements down to axioms. In this paper, candidate models can be any countably infinite collection of maps, denoted by $\mathcal{M}_1, \mathcal{M}_2, \ldots$, where we make the assumption that one of the candidate models successfully achieves the inference goal, i.e., the set of candidate models is sufficiently rich.

Why is it that some data-driven inference problems are universally solvable while others are not? We find that the way in which the ground-truth map and candidate models are observed greatly affects whether identification, prediction, or verification is possible. For example, we show that under reasonable types of observations for the ground-truth map and candidate models, AI text generation is always solvable (which is a prediction task) while AI-detection (checking if text is generated by a particular AI system is map identification task) and AI alignment (ensuring chatbots do not produce dangerous text is an input verification task) may not be solvable. Nonlinear PDE discovery may not be solvable by passively watching the ground-truth map when the PDE has multiple solutions; however, it is if one can decide precisely how to probe the ground-truth map. We will show that the way in which the ground-truth map is observed (type of training data) and the way in which candidate models are observed (type of testing data) makes some inference goals solvable and others not.

To this end, we define three main ways that maps are observed: (a) Passive observation, where one has no control over a map's inputs and has to accept whatever data is given, (b) Active observation, where one can

decide what inputs to try, and (c) Testing, where one has an input-output pair and can check if the map can go between the two. We prove that the way one observes the ground-truth and candidate maps directly influence whether map identification, map prediction, or map verification is universally solvable or not.

Since there are three inference goals (i.e., map identification, map prediction, and map verification), three ways to observe the ground-truth map (i.e., passive, active, and testing) and three ways to observe the candidate models (i.e., passive, active, and testing), there are $3 \times 3 \times 3 = 27$ types of problems one can investigate for each type of ground-truth map. All of them appear in some practical situation. For single-valued maps, i.e., maps for which every input is valid and gives a unique output, all 27 problems are universally solvable with an algorithm (see theorem 4). This makes the type of inference task captured by experimentally estimating the gravitational constant (e.g., Cavendish's experiment), deriving the dynamical system for the oscillations of a damped spring, and credit card encryption, universally solvable. For finite-output maps, which can be maps that have finitely many possible outputs given one input or a map where some inputs are invalid, we show that only 9-out-of-27 of the goals are universally solvable (see sections 3 to 5). This makes problems associated with AI-generation universally solvable, but not AI-detection and AI-alignment. For infinite-output maps, where an input can have infinitely many possible outputs, only 3-out-of-27 goals are solvable, making the inference goal associated with making a robot put away the dishes not universally solvable.

To illustrate, we present four concrete examples of inference problems, their goals, and types of observations:

AI text generation. One of the most popular map prediction tasks in recent years is in large language models (LLMs) for text generation (Vaswani et al., 2017; Lewis et al., 2020; Achiam et al., 2023; Gu & Dao, 2023; Yu & Erichson, 2025). For our setup, we suppose that there is a ground-truth map $\mathcal{M}_*$, which takes in a sequence of tokens and outputs a set of tokens that are a valid continuation of the sequence in human text. The length of the sequence of tokens is referred to as the context window and, in practice, an LLM selects a token from the set of valid continuation tokens. The candidate models are all the transformer architectures, together with all the possible parameters, being considered, assumed to be large enough so that it contains the ground-truth map. We regard the ground-truth map as being observed passively as human text available on the internet, but an ML algorithm cannot generate human-level text for themselves as training data. The candidate models are observed actively because given any sequence of input tokens and a candidate LLM, we can simulate the model to obtain the next token. In Theorem 2, we show that map prediction (called AI text generation in this setting) is universally solvable. However, with the same types of observations, map verification is not universally solvable (see Theorem 3). This suggests that AI-alignment, which is relevant if LLMs want to ensure that they do not output human text when asked inappropriate questions or instructed to perform dangerous tasks, may not be solvable with a universal algorithm when the ground-truth is passively observed. Theorem 3 shows that to universally solve the inference task associated with AI-alignment, we need to test the ground-truth map instead of passively observing it. In practice, this means that humans are needed for AI-alignment, which is already happening in reinforcement learning from human feedback (Hendrycks et al., 2020).

Who is speaking? Turing has a famous imitation test, where participants have a five minute conversation with a human or a machine and at the end the participant has to guess if they were speaking to a human or a machine (Turing, 2009). A machine passes the imitation test if it fools enough participants that it is human. Here, we imagine that we are trying to develop an algorithm to determine who we are speaking with. Our ground-truth map takes in a conversational prompt and outputs a set of reasonable replies. The ground-truth map could be a human texter or an AI chatbot. The candidate models are all chatbots and human texters. The inference task here is map identification as we want to determine who is speaking. We find that if the algorithm only gets to watch conversations, then we are passive observers of the ground-truth map, and no universal algorithm exists, to figure out who is speaking. However, if the algorithm is actively engaged in the conversation and can also engage in conversation with any of the candidate models, then it is universally solvable to determine who is producing the conversation.

Designing proteins for biological function. David Baker's lab designs proteins to perform a particular biological function (Baek et al., 2021; Hedrick, 2009). In this setting, we regard the ground-truth map $\mathcal{M}_*$ as an operator that takes a biological function to the set of all proteins that achieve that function.

We take the candidate models to be the set of all neural networks with a particular architecture, e.g., all possible generative models (Goodfellow et al., 2020; Kingma & Welling, 2013; Song et al., 2021; Huang et al., 2021; Song et al., 2020; Karras et al., 2022; Lipman et al., 2023; Lim et al., 2025) in the RFdiffusion framework (Watson et al., 2023), then our task is to select a candidate model that correctly outputs proteins for a given biological function, which is a map prediction task. The ground-truth map is observed passively with a database of known proteins and their biological functions. However, the candidate models are easy to evaluate for any specific input, making us active observers of them. In Theorem 2, we show that map prediction is universally solvable in this setting.

Personalized spam filters. Personalized spam filtering, where the classifier adapts to each user's definition of spam, is well-studied in the literature (Cormack & Lynam, 2007). Imagine that a ground-truth map takes a user as an input and maps it to all possible emails considered as spam by that user, i.e., a personalized spam filter. The candidate models could be all the possible classifiers, for example, represented by a neural network, that takes in a user's profile and an email and classifies it as spam or not. The goal is to identify which of the candidate models is the ground-truth, so that we can use it as a personalized spam filter. We imagine that the ground-truth map is observed by keeping track of the user's inbox and getting the user to flag spam emails. By the nature of candidate classifiers, we perform tests on them. One of the key properties of this problem is that given a user, there are infinitely many emails considered as spam by that user. Theorem 5 shows that even with active observations of the ground-truth map (i.e., sending the user fictitiously generated emails to be flagged), the inference task associated with this problem is not universally solvable. It suggests that a personal spam filter that is fully robust might be difficult to find.

We present more examples in Appendix A to demonstrate our framework.

1.1 Related work

The work in this paper relies on arguments from language generation and set identification to make conclusions about map identification and map prediction. In set identification, there is an unknown countably infinite set S_* (similar to our ground-truth map) and a sequence of distinct countably infinite sets $S_1, S_2, \ldots$, (similar to our candidate models) for which $S_j = S_*$ for some j. The set identification problem is often thought of as a game between an adversary and an algorithm. In each round of the game, an element of S_* is revealed to the algorithm (similar to a passive observation from the ground-truth map) and the algorithm may ask if an element s is contained in a set S_i for some i (similar to an active observation of our candidate models). After each round, the algorithm should give its current best guess at an integer j such that $S_j = S_*$.

In 1967, Gold showed that there does not exist an algorithm to universally solve set identification problems (Gold, 1967). This means that for any algorithm, there is a set identification problem for which the algorithm incorrectly guesses candidate sets for infinitely many rounds. If one thinks of a language as a countably infinite collection of valid words, then Gold essentially showed that one cannot learn a language by passively listening to it. We use Gold's ideas in one of our impossibility proofs by showing that all set identification problems are map identification problems.

Language generation is studied in Kleinberg & Mullainathan (2024). Language generation is not a set identification problem, though it has the same setup with a different goal. In language generation, one does not want to select an integer j such that $S_j = S_*$. Instead, one wants to select an integer j and an $s \in S_j$ such that $s \in S_*$. This is similar to our goal of map prediction. Remarkably, it was proved that language generation is solvable under Gold's observation model. If one thinks of a language as a countably infinite collection of valid words, then Kleinberg and Mullainathan essentially showed that one can parrot correct words of a language by passively listening to it. We use some of the ideas and techniques in Kleinberg & Mullainathan (2024) to show that some inference goals are solvable.

Compared to language identification and generation, which is modeled by a countably infinite set of valid strings, our setting is more intricate because it is defined by both the input set and the output(s) corresponding to an input. This gives rise to more scenarios to consider, depending on the cardinality of the output set (see Definition 1) and ways to observe the ground-truth map and candidate models (see Definition 2-4).

1.2 Limitations and core assumptions

Our theoretical analysis makes a few idealized assumptions. First, we assume that the input and output spaces $\mathcal{X}$ and $\mathcal{Y}$ of the map $\mathcal{M}_*$ are countably infinite, and that the ground-truth map $\mathcal{M}_*$ lies within a known, countable class of candidate maps $\{\mathcal{M}_i\}$. These assumptions are not intended to capture the full richness of practical learning scenarios, where $\mathcal{X}$ is often continuous and the true mechanism may lie outside any prescribed model class, but they enable precise reasoning about what kinds of information make a task feasible in principle. By working in a countable setting, we separate the information-theoretic aspects of discovery from questions of optimization or approximation.

Demonstrating that a particular inference goal is universally not solvable does not preclude progress. Our results only show that there is no universal algorithm to solve that type of inference, which means for any algorithm there is at least one specific set up in that class for which any algorithm fails to accomplish the goal. One could potentially make progress by inserting in more problem-dependent knowledge into any algorithm or coming up with a different way to collect training or testing data. Similarly, if a inference task is universally solvable, our framework does not pinpoint exactly when a universal algorithm has succeeded at the task, just that with enough data it will eventually succeed. Despite these limitations, the classification offers prescriptive value: it clarifies which types of observations are sufficient for successful inference. We hope it is a guide for data-driven discovery in the sciences.

1.3 Summary of our results and main takeaways

In section 6, we show that for single-valued maps where all the inputs are valid, map identification, map prediction, and map verification are all solvable for any type of observation. This is good news because many tasks in science are for maps that are single-valued. For example, since protein structures are unique given an amino acid sequence, the inference task associated with Alphafold (Jumper et al., 2021) is universally solvable under any type of observations from the ground-truth map and candidate models.

In sections 3 to 5, we consider the case where the ground-truth map is finite-output, which means it is a set-valued map where each output is a set of finite cardinality. These maps can also occur for single-valued maps where some inputs that are not valid (e.g., a chess engine that is given an invalid board position) and we want to regard the map as returning the empty set.

For finite-output ground-truth maps, we have a hierarchy of types of observations:

$$\text{active} \succ \text{testing} \succ \text{passive},$$

which means if it is universally solvable to solve an inference goal with passive observations then it universally solvable to solve the same goal with active or testing observations. Similarly, if a inference task can be done with testing for a finite-output map, then it can be done with active observations. This hierarchy of observations no longer holds when we start considering infinite-output maps, which are set-valued operators for which some output sets have possibly infinite cardinality (see section 7).

In the remainder of this paper, we present results about three types of ground-truth maps: (i) Single-valued, (ii) Finite-output, and (iii) Infinite-output. For each one we have 27 results, corresponding to the type of inference task and the types of observations. In section 2, we formally define a map and the three different ways to observe it. We study the feasibility of map identification, map prediction, and map verification given the various ways to observe the ground-truth map and candidate models in sections 3 to 5, respectively. In section 7, we conclude our analysis by studying set-valued maps for which the set contains infinitely many things for a given input.

2 How do we observe a ground-truth map and candidate model?

Ground-truth maps in science can be linear, nonlinear, single-valued, or set-valued. Here, we take a general point-of-view of a map and regard it as any function that takes objects from an input space $\mathcal{X}$ to objects in $\mathcal{Y}$ or subsets of $\mathcal{Y}$. Throughout our work, the input and output spaces of the candidate models match those of the ground-truth map, and we assume that the space of candidate models is rich enough that one

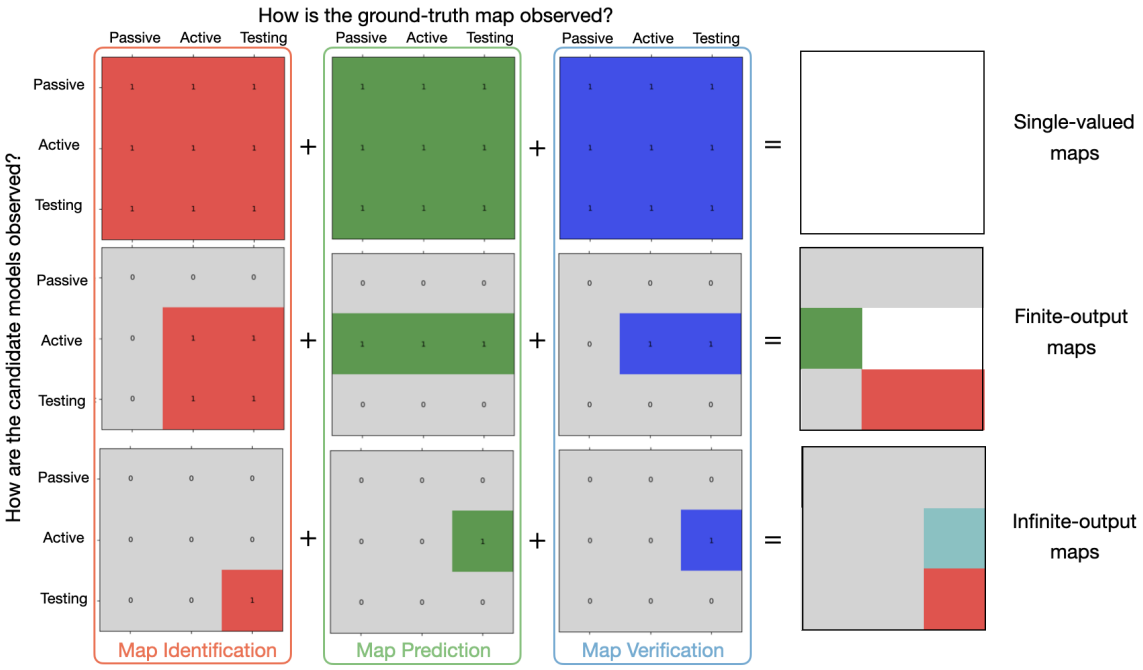

Figure 2: **Solvability landscape for data-driven inference across 81 scenarios.** Rows indicate the type of observation for candidate models (passive, active, testing) and columns indicate the type of observation for the ground-truth map, grouped by inference goal: map identification (red), prediction (green), and verification (blue). Separate blocks correspond to different ground-truth map structures: single-valued (top), finite-output (middle), and infinite-output (bottom). Colored cells mark scenarios that are universally solvable while gray cells indicate impossibility.

achieves the inference task. We also assume that the input space $\mathcal{X}$ and the output space $\mathcal{Y}$ are countably infinite sets so they can be enumerated.

For single-valued maps, every input $f \in \mathcal{X}$ corresponds to a unique output element g of $\mathcal{Y}$, i.e., $\mathcal{M}f \in \mathcal{Y}$. For example, a uniquely solvable PDE is associated with a solution operator $\mathcal{M}_*$ that maps a forcing term $f \in \mathcal{X}$ to a unique solution $g \in \mathcal{Y}$. In section 5, we show that map identification, map prediction, and map verification are all solvable when both the ground-truth maps and candidate models are single-valued, regardless of the type of observations one can make on the ground-truth map and candidate models.

However, there are also set-valued maps. There are two main reasons for set-valued maps: (1) The ground-truth map is set-valued (e.g., there are many valid continuation tokens in AI-generated text) or (2) The underlying unknown system has no output for some $f \in \mathcal{X}$ (e.g., there is no valid proof of a false statement). We define our ground-truth map as returning the empty set if the underlying system does not give an output for some $f \in \mathcal{X}$. To write single-valued and set-valued maps in a consistent notation, we regard both as functions from $\mathcal{X}$ to $\mathcal{P}(\mathcal{Y})$, where $\mathcal{P}(\mathcal{Y})$ is the power set of $\mathcal{Y}$. For single-valued maps, the outputs are always singleton sets.

Definition 1. *Let $\mathcal{X}$ and $\mathcal{Y}$ be two countably infinite sets. A map $\mathcal{M} : \mathcal{X} \to \mathcal{P}(\mathcal{Y})$ is a function that takes every input element $f \in \mathcal{X}$ to a subset of $\mathcal{Y}$, i.e., $\mathcal{M}f \subset \mathcal{Y}$. We assume that every map has at least one valid input-output pair, i.e., there exist some $f \in \mathcal{X}$ and $g \in \mathcal{Y}$ such that $g \in \mathcal{M}f$.*

We denote the ground-truth map by $\mathcal{M}_*$, which is a function from $\mathcal{X}$ to $\mathcal{P}(\mathcal{Y})$. If f is not a valid input to a map, then we consider it as outputting the empty set. Moreover, there is a list of distinct candidate models $\mathcal{M}_1, \mathcal{M}_2, \ldots$ with $\mathcal{M}_i : \mathcal{X} \to \mathcal{P}(\mathcal{Y})$, where one of the candidate models successful achieves the inference task, but we do not know a priori which one it is. To solve our inference goal of map identification, map prediction, or map verification, we collect observations from the ground-truth map $\mathcal{M}_*$. Then, using that

information, we select one of the candidate models, say $\mathcal{M}_i$, to try to accomplish our task. At no point are we told if $\mathcal{M}_i$ is correct; however, we are allowed to collect more observations from $\mathcal{M}_*$.

The type of observations that we make on the ground-truth map and the candidate models are crucial. We now formally define a passive, active, and testing observation.

Definition 2 (Passive)**.** *Let $\mathcal{X}$ and $\mathcal{Y}$ be two countably infinite sets and $\mathcal{M} : \mathcal{X} \to \mathcal{P}(\mathcal{Y})$. Fix an enumeration of $\mathcal{A} = \{(f, g) \in \mathcal{X} \times \mathcal{Y} \mid g \in \mathcal{M}f\}$ such that $\mathcal{A} = \{(f_1, g_1), (f_2, g_2), \ldots\}$. For every $k \geq 1$, the kth passive observation of $\mathcal{M}$ is the (f_ℓ, g_ℓ) input-output pair, where we define $\ell = \mathrm{mod}(k–1, n)+1$ and n is the cardinality of $\mathcal{A}$.*

In a passive observation, one has no control over the input or the output. In many other situations, such as conducting an experiment, we can specify an input and see the output. We call this an active observation.

Definition 3 (Active)**.** *Let $\mathcal{X}$ and $\mathcal{Y}$ be two countably infinite sets and $\mathcal{M} : \mathcal{X} \to \mathcal{P}(\mathcal{Y})$. For every input $f \in \mathcal{X}$, fix an enumeration $\{g_1(f), g_2(f), \ldots\}$ of the output set $\mathcal{M}f$. Let n_f be the cardinality of $\mathcal{M}f$. For every $k \geq 1$, the kth active observation of $\mathcal{M}$ with input f is $g_\ell(f)$, where $\ell = \mathrm{mod}(k-1, n_f) + 1$.*

Hence, a key difference between a passive and an active observation is that in an active observation, the input $f \in \mathcal{X}$ is chosen; in a passive one, it is not. If one keeps making a passive observation or an active observation with the same input f, then we assume that the observed outputs return all possible outputs and then repeat in a loop. Of course, if there are a countably infinitely many possible observations, then the observations can never repeat. Moreover, if $\mathcal{M}f = \emptyset$, then nothing will be observed but one will know that $\mathcal{M}f$ is empty.

Definition 4 (Testing)**.** *Let $\mathcal{X}$ and $\mathcal{Y}$ be two countably infinite sets and $\mathcal{M} : \mathcal{X} \to \mathcal{P}(\mathcal{Y})$. A test of $\mathcal{M}$ with input $f \in \mathcal{X}$ and output $g \in \mathcal{Y}$ is a Boolean value indicating whether $g \in \mathcal{M}f$.*

Therefore, for testing, one must propose both an input f and output g, and a test is an observation of a single Boolean value revealing if $g \in \mathcal{M}f$. A test happens most often in learning with oracles. For example, given an encryption algorithm, it is often much easier to verify if a password is correct than to generate one. It is also much easier to verify the validity of a proof of a challenging theorem than to generate one. This makes testing observations more practical for certain inference goals (see section 7).

3 When can we identify the ground-truth map for finite-output operators?

In this section, we study map identification for finite-output maps. This is where there is a ground-truth map $\mathcal{M}_* : \mathcal{X} \to \mathcal{P}(\mathcal{Y})$ and a set of candidate models $\mathcal{M}_1, \mathcal{M}_2, \ldots$, with $\mathcal{M}_i : \mathcal{X} \to \mathcal{P}(\mathcal{Y})$ for $i \geq 1$, where one wishes to identify which $\mathcal{M}_j = \mathcal{M}_*$. Throughout this section, maps are finite-output meaning that $\mathcal{M}f$ is a set of finite cardinality for any $f \in \mathcal{X}$. Recall that we assume that $\mathcal{X}$ and $\mathcal{Y}$ are countably infinite and there is a candidate model equal to $\mathcal{M}_*$ so it is possible to accomplish the task of map identification. The inference task of map identification can be formulated as a game played between an algorithm and an adversary (see Figure 3).

The map identification game. For every map $\mathcal{M}$ (which can be the ground-truth map $\mathcal{M}_*$ or a candidate model $\mathcal{M}_j$ for some $j \geq 1$), the adversary fixes an enumeration of $\mathcal{X}$ as in Definition 2, and an enumeration of $\mathcal{M}f$ for every $f \in \mathcal{X}$, which determine the order of passive and active observations. In each round, the following happens:

1. **Observations from the ground-truth map.** The algorithm collects observations from $\mathcal{M}_*$, which may be passive, active, or testing. For a passive or active observation, it receives the next item in the adversary's enumeration (see Definitions 2 and 3).
2. **Observations from a candidate model.** The algorithm collects observations from the candidate models, which could be passive, active, or testing. When the algorithm makes a passive or active observation, it gets the next observation in the adversary's enumeration (see Definition 2 and 3).
3. **Identify map.** The algorithm selects a candidate model $\mathcal{M}_j$ as its current best guess at correctly identifying the ground-truth map $\mathcal{M}_*$.

We say that the algorithm wins the map identification game for $\mathcal{M}_*$ if after a finite number of rounds the algorithm always correctly selects the candidate model with $\mathcal{M}_j = \mathcal{M}_*$ in the "identify map" step. So, even if more rounds are played, the algorithm sticks with selecting the same correct candidate model after some point. (This prevents the algorithm from getting lucky in one round.) Importantly, at no point in the game is the algorithm told it has selected the correct candidate model; it just collects more observations in subsequent rounds and keeps its choice.

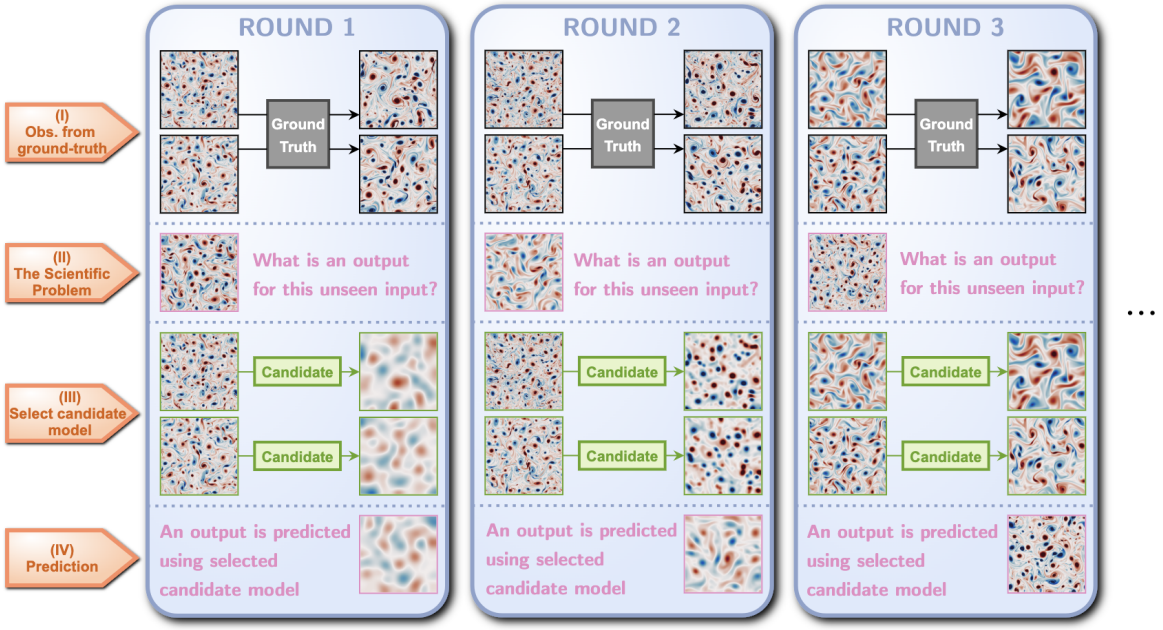

Figure 3: We cast data-driven inference as a game between a learner and an adversary. In each round, the learner gathers observations from the ground-truth map, receives a question from the adversary, and selects a candidate model to answer the question. For prediction, for example, the question may involve forecasting the next state in a simulated physical system. Over successive rounds, the learner refines the selection of the candidate model.

Formally, the algorithm can be modeled as an oracle machine, a Turing machine equipped with access to an oracle that provides information about non-finitely-representable operators. In each iteration, the machine may query the oracle for information about the ground-truth map $\mathcal{M}_*$ or candidate models $\mathcal{M}_j$. The nature of oracle access depends on the type of observer (passive, active, or tester), and the order of oracle's output may be arbitrary (one can imagine that it is determined by an adversary). The machine is allowed to make an arbitrary number of queries per iteration but must eventually halt and output a guess for the ground-truth map $\mathcal{M}_*$. In this formulation, the input tape remains empty across iterations, since the task is fixed.

Definition 5 (Map identification is universally solvable.)**.** *We say map identification is universally solvable if there is a deterministic algorithm that always wins the map identification game.*

If map identification is universally solvable, then there is a mechanism for identifying the candidate model that is equal to $\mathcal{M}_*$ for any instantiation of the map identification problem. In the map identification game, the adversary is not actively playing the game. Its role is only in selecting the enumeration of the inputs and outputs. Of course, the adversary selects an enumeration that makes it as difficult as possible for any algorithm to accomplish map identification. The way we observe maps are crucial in determining whether map identification is solvable.

Theorem 1. *Map identification for finite-output operators (see section 3) is universally solvable (see Definition 5) if and only if both the ground-truth map and candidate models are observed actively or via testing (see Definition 2-4).*

See Appendix C for a proof of the theorem. Theorem 1 gives us a complete characterization of when map identification is universally solvable. For any algorithm to always win the map identification game, neither the ground-truth map nor the candidate models can be passively observed. As an example, consider a multi-label image tagging system where each image is associated with several tags such as tagging an example with all the objects in the scene. Here, we imagine that the ground-truth map is an AI system for tagging images and we would like to figure out which AI system it is. We have a list of proposed AI systems, which are our candidate models that can be observed actively. If the algorithm can deliberately select specific images for the ground-truth AI system to tag, then it is possible to identify it. However, if the algorithm only receives a fixed, passive stream of image-tag pairs, then this is an inference goal that is not universally solvable. Finally, under a testing observation regime, where an algorithm can propose an image along with a tag and receive a simple yes-or-no confirmation that the tags appropriately match the image, there is a universal algorithm for identifying the AI system.

4 When can we predict the ground-truth map for finite-output maps?

For some unknown systems, our objective is not to identify the ground-truth map itself, but rather to correctly predict its output for any given input. In particular, one does not accomplish map identification to succeed at map prediction. For example, in solving PDEs with neural operators, one need not recover the full differential operator governing the system; instead, it is sufficient to approximate the mapping from boundary conditions to solutions (Rudy et al., 2017). Similarly, in fluid flow prediction, a learned model can accurately forecast flow evolution without explicitly recovering the Navier–Stokes operator, as long as it captures the correct input-output relationships.

Here, we study map prediction for finite-output maps with the same setup as in section 3. This task can be formulated as a game played between an algorithm and an adversary.

The map prediction game. For every map $\mathcal{M}$ (which can be the ground-truth map $\mathcal{M}_*$ or a candidate model $\mathcal{M}_j$ for some $j \geq 1$), the adversary fixes an enumeration of $\mathcal{X}$ as in Definition 2, and an enumeration of $\mathcal{M}f$ for every $f \in \mathcal{X}$, which determines the passive and active observations. Then, in each round, the following happens:

1. **Observation from the ground-truth map.** The algorithm collects observations from $\mathcal{M}_*$, which may be passive, active, or testing. For a passive or active observation, it receives the next item in the adversary's enumeration (see Definitions 2 and 3).
2. **Adversary's selected input.** The adversary selects an $f \in \mathcal{X}$ for which $\mathcal{M}_* f \neq \emptyset$.
3. **Observations from a candidate model.** The algorithm collects observations from the candidate models, which could be passive, active, or testing. When the algorithm makes a passive or active observation, it gets the next observation in the adversary's enumeration (see Definition 2 and 3).
4. **Prediction.** The algorithm selects a candidate model $\mathcal{M}_j$ and a $g \in \mathcal{M}_j f$ as its current best guess at picking a $g \in \mathcal{M}_* f$.

The map prediction game differs from the identification game in that the adversary chooses a new f each round for which the algorithm must predict a $g \in \mathcal{M}_* f$. We say that the algorithm wins the map prediction game for $\mathcal{M}_*$ if after a finite number of rounds the algorithm always correctly selects a $g \in \mathcal{M}_* f$, in the "prediction" step. In the game, $f \in \mathcal{X}$ is selected by the adversary and can change in each round of the game. Importantly, the algorithm is never told whether its prediction is correct.

This algorithm is also an oracle machine, as discussed in section 2. The main difference is that in each iteration, the input tape contains a valid input $f \in \mathcal{X}$ picked by the adversary, and the algorithm needs to produce an element of $\mathcal{M}_* f$ on the output tape.

Definition 6 (Universal solvability of Map Prediction.)**.** *We say map prediction is universally solvable if there exists a deterministic algorithm that always wins the map prediction game.*

On the one hand, the map prediction game looks easier than the map identification game because the algorithm only has to correctly predict a valid output for one input per round. On the other hand, the map

prediction game is harder because the adversary can change the input f every round. In particular, the map prediction game can be unsolvable when the map identification game is solvable and vice versa.

Theorem 2. *For finite-output maps, the map prediction game is universally solvable (see Definition 6) if and only if the candidate models are observed actively (see Definition 2-4).*

See Appendix D for a proof of the theorem. We see, from Theorem 2, that map prediction is universally solvable when the ground-truth map is passively observed and the candidate models are actively observed, but map identification is not. Returning to the multi-label image tagging system, if the underlying AI system is only passively observed then it is universally solvable to mimic the ground-truth tagging map even if it is not possible to identify which system is doing the tagging. Similarly, from Theorem 1, map identification is universally solvable when the candidate models are observed via testing and the ground-truth map is actively observed, but map prediction is not.

5 When can we verify an input for finite-output maps?

There are many unknown systems for which the given input to the corresponding ground-truth map may not be valid. For example, when we ask an automated proof generator to prove a false statement, the set of valid proofs for that statement is empty. Or, when we ask a large language model to produce the social security number of an individual, we want the map to not reveal its knowledge (even if it knows it). In that case, we want to learn a ground-truth map with input verification.

In this section, we study map verification for finite-output maps with the same setup as in sections 3 and 4. The inference task of map verification can be formulated as a game played between an algorithm and an adversary.

The map verification game. For every map $\mathcal{M}$ (which can be the ground-truth map $\mathcal{M}_*$ or a candidate model $\mathcal{M}_j$ for some $j \geq 1$), the adversary fixes an enumeration of $\mathcal{X}$ as in Definition 2, and an enumeration of $\mathcal{M}f$ for every $f \in \mathcal{X}$, which determine the passive and active observations. Then, in each round, the following happens:

1. **Observations from the ground-truth map.** The algorithm collects observations from $\mathcal{M}_*$, which may be passive, active, or testing. For a passive or active observation, it receives the next item in the adversary's enumeration (see Definitions 2 and 3).
2. **Adversary's selected input and output.** The adversary selects an arbitrary $f \in \mathcal{X}$ (possibly with $\mathcal{M}_* f = \emptyset$).
3. **Observations from a candidate model.** The algorithm collects observations from the candidate models, which could be passive, active, or testing. When the algorithm makes a passive or active observation, it gets the next observation in the adversary's enumeration (see Definitions 2 and 3).
4. **Verification.** The algorithm selects a Boolean value as its current best guess of whether $\mathcal{M}_* f = \emptyset$. If it guesses that $\mathcal{M}_* f \neq \emptyset$, then it also selects a candidate model $\mathcal{M}_j$ and a $g \in \mathcal{M}_j$ as its best guess at picking $g \in \mathcal{M}_* f$.

We say that the algorithm wins the map verification game for $\mathcal{M}_*$ if after a finite number of rounds the algorithm always correctly returns the right Boolean value in the "verification" step, where $f \in \mathcal{X}$ can change in each round of the game, and if the adversary's input f is valid (i.e., $\mathcal{M}_* f \neq \emptyset$), the algorithm always correctly selects a $g \in \mathcal{M}_* f$. At no point in the game is the algorithm told that it has selected the correct Boolean value or a valid output.

Definition 7. *We say map verification is universally solvable if there is a deterministic algorithm that always wins the map verification game.*

Clearly, a map verification problem is harder than a map prediction one. On the other hand, it might be tempting to think that a map verification problem can be reduced to a map prediction task. For example, one could introduce a new element $g_0 \notin \mathcal{Y}$ into the output space, acting like a flag to tell when an input has no valid outputs. One could redefine $\mathcal{Y}' = \mathcal{Y} \cup \{g_0\}$ and append g_0 to the output of every map, i.e.,

$\mathcal{M}'f = \mathcal{M}f \cup \{g_0\}$ for all maps $\mathcal{M}_*, \mathcal{M}_1, \mathcal{M}_2, \ldots$ and $f \in \mathcal{X}$. Thus, if $\mathcal{M}'f = \{g_0\}$, then $\mathcal{M}f = \emptyset$. One might hope that this reduces map verification into map prediction. To see why this does not work, imagine collecting passive observations of both the ground-truth map and the candidate models. Then, if $\mathcal{M}_*f = \emptyset$ for some $f \in \mathcal{X}$, passive observations of $\mathcal{M}'$ immediately finds that $g_0 \in \mathcal{M}'f$ after one observation, whereas we can never conclude that $\mathcal{M}_*f = \emptyset$ as this negative information is not present in passive observation. For finite-output maps, we can show that the only difference between the feasibility of map verification and map prediction arises only when the ground-truth map is passively observed.

Theorem 3. *For finite-output maps, the map verification game is universally solvable (see Definition 7) if and only if the ground-truth map is observed actively or testing and the candidate maps are observed actively (see Definition 2-4).*

See Appendix E for a proof of the theorem. Comparing the settings where map verification is universally solvable with those for map prediction shows that the only difference arises when the ground-truth maps are observed passively and the candidate models are observed actively. In AI text generation, the training data is passively collected from human-written text on the internet, but the AI system can be actively evaluated on any specified inputs. This makes AI text generation fall into the category of map prediction, which is universally solvable, whereas the related task of AI alignment requires map verification and is not universally solvable. (AI alignment involves testing, e.g., ensuring it does not return a person's social security number when prompted.)

6 All inference tasks are solvable for single-valued maps

Single-valued maps are those in which each input corresponds to a unique valid output, i.e., $\mathcal{M}f$ is a singleton for all $\mathcal{M} = \mathcal{M}_*, \mathcal{M}_1, \mathcal{M}_2, \ldots$ and $f \in \mathcal{X}$. Many unknown systems involve such maps, and in this setting we can show that map identification, map prediction, and map verification are all universally solvable, regardless of how the ground-truth map and candidate models are observed.

Theorem 4. *For single-valued maps, map identification (see Section 3), map prediction (see Section 4), and map verification (see Section 5) are universally solvable no matter how the ground-truth map and candidates models are observed (see Definition 2-4).*

See Appendix F for a proof of the theorem. Theorem 4 shows all inference goals are universally solvable for single-output maps regardless of the type of observations.

7 Necessity of negative data: infinite-output maps

So far, we have assumed that all maps are single-output or finite-output; that is, $\mathcal{M}f \subset \mathcal{Y}$ is finite for every $f \in \mathcal{X}$. In practice, however, there are many situations in which the number of outputs is infinite. For example, in image generation, a single textual prompt can correspond to infinitely many valid images; in automated theorem proving, a single statement may have infinitely many valid proofs.

A key difference between finite-output and infinite-output maps lies in the information obtainable through active observations. For finite-output maps, sufficiently many active observations allow one to recover $\mathcal{M}f$ exactly: after enough queries, the outputs repeat, revealing both what is in $\mathcal{M}f$ and what is not. This "negative information" is a standard concept in theoretical computer science and plays a crucial role in identification (Gold, 1967; Angluin, 1980). In contrast, for infinite-output maps, active observation alone cannot reveal whether a given element is not in $\mathcal{M}f$. Consequently, an active observer cannot, in general, perform a definitive membership test for (f, g) using active observations alone. For completeness, we redefine these infinite-output problems as follows.

Definition 8. *An infinite-output map identification, map prediction, or map verification game is the corresponding game from Sections 3 to 5, respectively, in which the ground-truth map $\mathcal{M}_*$ and the candidate models $\mathcal{M}_1, \mathcal{M}_2, \ldots$ may have infinite outputs.*

Not surprisingly, the lack of negative data severely limits what can be achieved with infinite-output maps. We will show that only a few special cases admit a universal algorithm.

Before stating the theorem, we provide some intuition. In the infinite-output setting, finite transcripts are inherently inconclusive: after any finite sequence of observations, there remain infinitely many possible extensions of the ground-truth graph, so enumeration alone cannot guarantee stabilization. This makes a testing oracle for the ground truth $\mathcal{M}_*$ essential: without membership tests for pairs (f, g), an adversary can always postpone revealing the true output beyond the finite set explored so far (a standard finite-extension or diagonalization argument). With tests on $\mathcal{M}_*$, disagreements can be certified and incorrect behaviors eliminated. For identification, however, one must establish equality between two infinite graphs; since active access to $\mathcal{M}_j$ cannot rule out unseen counterexamples, tests on the candidate models are also required. By contrast, prediction and verification require only finding a single correct witness for each input: active access to the $\mathcal{M}_j$ suffices to search for such a witness, and tests on $\mathcal{M}_*$ suffice to confirm it. After finitely many refutations, only the true behavior remains, yielding stabilization. Thus, tests on $\mathcal{M}_*$ are necessary in all cases; identification further requires tests on the $\mathcal{M}_j$, while prediction and verification require only active access to $\mathcal{M}_j$ plus tests on $\mathcal{M}_*$.

Theorem 5. *The following statements about infinite-output problems hold:*

1. *There exists an algorithm to universally solve the infinite-output map identification game defined in Definition 8 given any ground-truth and candidates if and only if they perform tests of the ground-truth (see Definition 2, 3, and 4) and of the candidate models.*
2. *There exists an algorithm to universally solve the infinite-output map prediction game defined in Definition 8 given any ground-truth and candidates if and only if they perform tests of the ground-truth (see Definition 2, 3, and 4) and make active observations of the candidate models.*
3. *There exists an algorithm to universally solve the infinite-output map verification game defined in Definition 8 given any ground-truth and candidates if and only if they perform tests of the ground-truth (see Definition 2, 3, and 4) and make active observations of the candidate models.*

See Appendix G for a proof of the theorem. To illustrate the importance of Theorem 5, consider the spam email detection problem that we discussed in the introduction. Imagine that the ground-truth is an operator that maps each user to the set of emails considered spam by that user. Such an operator is infinite-output. Each candidate model is some model such that, given a user and an email, will return a prediction of whether the email is a spam. Hence, we always perform tests of the candidate models. In this problem, we do not care about learning the operator or verifying the inputs: indeed, our model does not need to generate spams emails for given users. That is, we want to do map identification so that we can then classify an email by testing the candidate model that we have identified. Now, by Theorem 5, we know that map identification is solvable only when we can test the ground-truth. That is, we need to examine both spam and non-spam emails, whereas only seeing the spam ones is not enough.

8 Conclusion

The results in this paper highlight how the universal solvability of identification, prediction, and verification tasks depends critically on both the nature of the underlying maps (e.g., single-valued vs. infinite-output) and the type of observational access available to the ground-truth map and candidate models. We illustrate how theoretical results translate into practical consequences for the design and evaluation of scientific and AI systems. Our results underscore that, while universal solvability can often be achieved for single-valued maps under any observation regimes, infinite-output settings require more stringent conditions, such as access to testing oracles. Our findings provide a unified perspective that connects information-theoretic constraints with methodological choices, offering guidance for developing reliable algorithms in complex real-world settings.

References

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

Md Hasan Al Banna, Kazi Abu Taher, M Shamim Kaiser, Mufti Mahmud, Md Sazzadur Rahman, ASM Sanwar Hosen, and Gi Hwan Cho. Application of artificial intelligence in predicting earthquakes: state-of-the-art and future challenges. *IEEE Access*, 8:192880–192923, 2020.

Richard M Allen, Paolo Gasparini, Osamu Kamigaichi, and Maren Bose. The status of earthquake early warning around the world: An introductory overview. *Seismological research letters*, 80(5):682–693, 2009.

Dana Angluin. Inductive inference of formal languages from positive data. *Information and control*, 45(2): 117–135, 1980.

Minkyung Baek, Frank DiMaio, Ivan Anishchenko, Justas Dauparas, Sergey Ovchinnikov, Gyu Rie Lee, Jue Wang, Qian Cong, Lisa N Kinch, R Dustin Schaeffer, et al. Accurate prediction of protein structures and interactions using a three-track neural network. *Science*, 373(6557):871–876, 2021.

Haniel Barbosa, Clark Barrett, Byron Cook, Bruno Dutertre, Gereon Kremer, Hanna Lachnitt, Aina Niemetz, Andres Nötzli, Alex Ozdemir, Mathias Preiner, et al. Generating and exploiting automated reasoning proof certificates. *Communications of the ACM*, 66(10):86–95, 2023.

Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*, 113 (15):3932–3937, 2016.

Shengze Cai, Zhiping Mao, Zhicheng Wang, Minglang Yin, and George Em Karniadakis. Physics-informed neural networks (pinns) for fluid mechanics: A review. *Acta Mechanica Sinica*, 37(12):1727–1738, 2021.

Gordon V Cormack and Thomas R Lynam. Online supervised spam filter evaluation. *ACM Transactions on Information Systems (TOIS)*, 25(3):11–es, 2007.

Salvatore Cuomo, Vincenzo Schiano Di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi, and Francesco Piccialli. Scientific machine learning through physics–informed neural networks: Where we are and what's next. *Journal of Scientific Computing*, 92(3):88, 2022.

E Mark Gold. Language identification in the limit. *Information and control*, 10(5):447–474, 1967.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11): 139–144, 2020.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

John Harrison. *Handbook of practical logic and automated reasoning*. Cambridge University Press, 2009.

Philip W Hedrick. *Genetics of populations*. Jones & Bartlett Publishers, 2009.

Dan Hendrycks, Collin Burns, Steven Basart, Andrew Critch, Jerry Li, Dawn Song, and Jacob Steinhardt. Aligning ai with shared human values. *arXiv preprint arXiv:2008.02275*, 2020.

Chin-Wei Huang, Jae Hyun Lim, and Aaron C Courville. A variational perspective on diffusion-based generative models and score matching. *Advances in Neural Information Processing Systems*, 34:22863–22876, 2021.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589, 2021.

Cezary Kaliszyk and Josef Urban. Learning-assisted automated reasoning with flyspeck. *Journal of Automated Reasoning*, 53:173–213, 2014.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Jon Kleinberg and Sendhil Mullainathan. Language generation in the limit. *arXiv preprint arXiv:2404.06757*, 2024.

Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24(89):1–97, 2023.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

Soon Hoe Lim, Yijin Wang, Annan Yu, Emma Hart, Michael W Mahoney, Xiaoye S Li, and N Benjamin Erichson. Elucidating the design choice of probability paths in flow matching for forecasting. *TMLR*, 2025.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *International Conference on Learning Representations*, 2023.

Lu Lu, Pengzhan Jin, and George Em Karniadakis. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*, 2019.

S Mostafa Mousavi, William L Ellsworth, Weiqiang Zhu, Lindsay Y Chuang, and Gregory C Beroza. Earthquake transformer—an attentive deep-learning model for simultaneous earthquake detection and phase picking. *Nature communications*, 11(1):3952, 2020.

Samuel H Rudy, Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Data-driven discovery of partial differential equations. *Science advances*, 3(4):e1602614, 2017.

Hayden Schaeffer. Learning partial differential equations via data discovery and sparse optimization. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 473(2197):20160446, 2017.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

Yang Song, Conor Durkan, Iain Murray, and Stefano Ermon. Maximum likelihood training of score-based diffusion models. *Advances in neural information processing systems*, 34:1415–1428, 2021.

Alan M Turing. *Computing machinery and intelligence*. Springer, 2009.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, and I. Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Joseph L Watson, David Juergens, Nathaniel R Bennett, Brian L Trippe, Jason Yim, Helen E Eisenach, Woody Ahern, Andrew J Borst, Robert J Ragotte, Lukas F Milles, et al. De novo design of protein structure and function with rfdiffusion. *Nature*, 620(7976):1089–1100, 2023.

Annan Yu and N Benjamin Erichson. Block-biased mamba for long-range sequence processing. *arXiv preprint arXiv:2505.09022*, 2025.

A Applications of our framework

In addition to the ones discussed in section 1, we elaborate on several examples that illustrate our framework.

A.1 Automatic theorem proving

Consider a formal logic system with a fixed set of axioms. The ground-truth map $\mathcal{M}_*$ takes a statement $P \in \mathcal{X}$ to the set $\mathcal{M}_* P$ of all valid proofs of P. When P is true, this set is typically infinite. Automatic theorem proving seeks to produce a valid proof for any true statement and to correctly reject any false one, framing the task as map verification. Modern AI theorem provers can be treated as candidate models; since they can be run on arbitrary inputs, we are active observers of the candidates. By Theorem 5, if the ground-truth map is observed passively, e.g., by training only on true statements paired with valid proofs, universal solvability is impossible. To enable universal solvability, training must incorporate the rules of the logic so that any purported proof can be tested for validity (Harrison, 2009).

A.2 Proof validation

Here, the ground-truth map is again defined by the logic system, but the candidate model takes as input a statement and a proposed proof, returning whether it deems the proof valid. Each candidate model thus maps a statement to the set of all "proofs" it accepts. We act as testers of these models. Proof validation does not require computing the ground-truth map explicitly; it can be solved by identifying a candidate whose behavior matches the ground truth (Barbosa et al., 2023). By Theorem 5, this is only possible with the ability to test $\mathcal{M}_*$ directly. In particular, examples of valid proofs alone are insufficient; the complete logic must be available to verify any proposed proof.

A.3 PDE learning

In PDE learning, the ground-truth map is the underlying PDE that takes a forcing term to its unique solution. Candidate models include DeepONets (Lu et al., 2019), Fourier Neural Operators (Li et al., 2020), and Physics-Informed Neural Networks (Cai et al., 2021; Cuomo et al., 2022), across all parameter settings. The task may be map prediction (computing the solution for a given forcing term) or map identification (recovering the PDE form). Since the solution to a well-posed PDE is unique, the output set for each input is a singleton, making the problem theoretically tractable. While the input and output spaces are continuous and thus uncountable, we adopt an approximation-theoretic view: discretizing onto an ϵ-grid reduces the setting to countable spaces (see the main text). The ground-truth map may be observed passively or actively, depending on whether we can choose the forcing term, while candidates are actively observed via simulation. In this setting, the task is universally solvable (see Theorem 4).

A.4 AI chess engines

Here, the ground-truth map assigns to a board state the set of good moves. as in AlphaZero (Silver et al., 2018). Candidate models include all AI algorithms under consideration, each defined by an architecture and parameter set. If the goal is to output only the best move, the problem reduces to a single-output setting and is universally solvable (see Theorem 4). In many applications, however, we require a set of good moves (from which a chess engine will evaluate with a deep calculation) making the problem finite-output. The output set can also be empty, e.g., if resignation is optimal. In this general form, the task is map verification: given a board state, determine whether a good move exists and, if so, produce one. Candidates are actively observed, but if the ground truth is observed only passively (e.g., from historical expert games), universal solvability fails by Theorem 3. Achieving universal solvability requires active observation, such as simulating games between AI agents to generate new or targeted board states and their outcomes.

B Related work on language identification and generation

The work in this paper relies on arguments from language generation and set identification to make conclusions about map identification and map prediction.

In set identification, there is an unknown countably infinite set S_* (similar to our ground-truth map) and a sequence of distinct countably infinite sets $S_1, S_2, \ldots$, (similar to our candidate models) for which $S_j = S_*$ for some j. The set identification problem is often thought of as a game between an adversary and an algorithm. In each round of the game, an element of S_* is revealed to the algorithm (similar to a passive observation from the ground-truth map) and the algorithm may ask if an element s is contained in a set S_i for some i (similar to an active observation of our candidate models). After each round, the algorithm should give its current best guess at an integer j such that $S_j = S_*$.

In 1967, Gold showed that there does not exist an algorithm to universally solve set identification problems (Gold, 1967). This means that for any algorithm, there is a set identification problem for which the algorithm incorrectly guesses candidate sets for infinitely many rounds. If one thinks of a language as a countably infinite collection of valid words, then Gold is essentially showing that one cannot learn a language by passively listening to it. We use Gold's ideas in one of our impossibility proofs by showing that all set identification problems are map identification problems.

Language generation is studied in a recent paper by Kleinberg and Mullainathan (Kleinberg & Mullainathan, 2024). Language generation is not a set identification problem, though it has the same setup with a different goal. In language generation, one does not want to select an integer j such that $S_j = S_*$. Instead, one wants to select an integer j and an $s \in S_j$ such that $s \in S_*$. This is similar to our goal of map prediction. Remarkably, it was proved that language generation is solvable under Gold's observation model. If one thinks of a language as a countably infinite collection of valid words, then Kleinberg and Mullainathan essentially showed that one can parrot correct words of a language by passively listening to it. We use some of the ideas and techniques in Kleinberg & Mullainathan (2024) to show that some inference goals are solvable.

Compared to language identification and generation, which is modeled by a countably infinite set of valid strings, our setting is more intricate because it is defined by both the input set and the output(s) corresponding to an input. This gives rise to more scenarios to consider, depending on the cardinality of the output set (see Definition 1) and ways to observe the ground-truth map and candidate models (see Definition 2-4).

C Proof of Theorem 1

Proof. We break the proof into three parts. In the first part, we show that map identification for finite-output operators is universally possible when the ground-truth map and the candidate models are both observed actively or via testing. In the second and third parts, we show that map identification is not universally possible when the ground-truth map or the candidate models are observed passively.

Part I: Map identification for finite-output operators is universally solvable with active or testing observations. First, note that any algorithm can mimic a testing observation with active observations. That is, suppose the algorithm wants to know whether $g \in \mathcal{M}f$ for some operator $\mathcal{M}$, $f \in \mathcal{X}$, and $g \in \mathcal{Y}$. Then, the algorithm can keep actively observing $\mathcal{M}$ with input f and eventually the whole set $\mathcal{M}f$ is observed; at which point it can check if $g \in \mathcal{M}f$.

Hence, to show map identification is universally solvable for finite-output operators with active or testing observations, it suffices to assume that the algorithm observes both the ground-truth map and candidate models via testing. Let $(f_1, g_1), (f_2, g_2), \ldots$ be the fixed enumeration selected by the adversary. The algorithm begins by selecting $\mathcal{M}_1$ as its current best guess at $\mathcal{M}_*$. If in the tth round of the game the algorithm's best guess is $\mathcal{M}_k$, then the algorithm queries if $g_t \in \mathcal{M}_* f_t$ and if $g_t \in \mathcal{M}_k f_t$. If the algorithm finds that $g_t \in \mathcal{M}_* f_t$ and $g_t \notin \mathcal{M}_k f_t$ or $g_t \notin \mathcal{M}_* f_t$ and $g_t \in \mathcal{M}_k f_t$ (i.e., $g_t \in \mathcal{M}_* f_t \triangle \mathcal{M}_k f_t$, where $\triangle$ is the symmetric difference between two sets), then the algorithm discards $\mathcal{M}_k$ and moves on to $\mathcal{M}_{k+1}$ as its current best guess. The round counter is set back to $t = 1$ and the game continues. (The round counter is set back to 1 so that the algorithm can test $\mathcal{M}_{k+1}$ with $f_1, f_2, \ldots,$.)

To show that the algorithm wins the game, assume that $\mathcal{M}_* = \mathcal{M}_{k_*}$ for some k_*. Then, for every $k < k_*$, since $\mathcal{M}_k \neq \mathcal{M}_*$, there must exist some f_t and g_t for which $g_t \in \mathcal{M}_* f_t \triangle \mathcal{M}_k f_t$. Hence, the algorithm discards $\mathcal{M}_k$ as its current best guess. That means the algorithm eventually selects $\mathcal{M}_{k_*}$ and never discards it since $\mathcal{M}_{k_*} = \mathcal{M}_*$, winning the map identification game.

Part II: Map identification for finite-output operators is not possible with passive observations of the ground-truth. In part II of the proof, we play the map identification game with an algorithm that passively observes the ground-truth map. We show that map identification is not universally solvable, regardless of how the candidate models are observed. Since the algorithm can mimic a passive observation or perform a test with active observations, it suffices to assume that the algorithm observes the candidate models actively.

Suppose that there is an algorithm that universally solves the map identification. We now show that such an algorithm can universally solve the set identification problem, which contradicts Gold's 1967 result (Gold, 1967). Given any set identification problem, let $\mathcal{X} = \bigcup_{i=1}^{\infty} S_i$ and $\mathcal{Y} = \{1\}$, respectively. We define the ground-truth map and candidate models as indicators of the sets:

$$\mathcal{M}_* s = \begin{cases} \emptyset, & \text{if } s \notin S_*, \\ \{1\}, & \text{if } s \in S_*, \end{cases} \qquad \mathcal{M}_i s = \begin{cases} \emptyset, & \text{if } s \notin S_i, \\ \{1\}, & \text{if } s \in S_i. \end{cases} \tag{1}$$

A passive observation of $\mathcal{M}_*$ is equivalent to obtaining an element $s_k \in S_*$ and an active observation of $\mathcal{M}_i$ with an input s is equivalent to asking if $s \in S_i$. Since the algorithm universally wins the map identification game, the algorithm correctly identifies $\mathcal{M}_*$ after a finite number of rounds. Moreover, $\mathcal{M}_*$ is a indicator for S_*, so the algorithm identifies which $S_j = S$ and universally solves the set identification problem, leading to a contradiction.

Part III: Map identification for finite-output operators is not universally possible with passive observations of the candidate models. In part III of the proof, we play the map identification game with an algorithm that passively observes the candidates. We show that map identification is not universally possible, regardless of how the ground-truth map is observed. Since the algorithm can mimic an active observation or perform a test with active observations, it suffices to assume that the algorithm actively observes the ground-truth map. We again suppose that a universal algorithm exists and seek a contradiction.

Assume, for the sake of contradiction, that there exists a universal algorithm for map identification when the ground-truth map is observed actively while candidate models are observed only passively. Let the input

space be $\mathcal{X} = \mathbb{Z}$ and the output space be $\mathcal{Y} = \{1\}$. For notational convenience, identify each candidate model $\mathcal{M}_i$ with a subset $S_i \subset \mathbb{Z}$ by setting

$$\mathcal{M}_i(n) = \begin{cases} \{1\}, & \text{if } n \in S_i, \\ \emptyset, & \text{otherwise.} \end{cases}$$

The adversary controls the passive observations of candidate models as follows. For any candidate $\mathcal{M}_i$, if no observation has yet been made, the algorithm receives the input-output pair $(0, \{1\})$. Otherwise, two types of passive observations are provided:

- A "top" observation reveals the input-output pair $(j + 1, \{1\})$, where j is the largest input observed for $\mathcal{M}_i$ so far.
- A "bottom" observation reveals the input-output pair $(j - 1, \{1\})$, where j is the smallest input observed.

We partition the execution into stages and set up an infinite sequence of map identification problems (all with $\mathcal{M}_*(n) = \{1\}$ for all $n \in \mathbb{Z}$), so that the first k of them look identical to the algorithm until the end of the kth stage. In the kth stage:

- For candidate $\mathcal{M}_1$, the first passive observation is taken from the top, while all subsequent observations in that stage are taken from the bottom.
- For every candidate $\mathcal{M}_j$ with $j \geq 2$ and $j \neq k + 1$, all passive observations are from the bottom.
- For candidate $\mathcal{M}_{k+1}$, passive observations alternate between top and bottom.

Since the algorithm is assumed to be universally correct, it must eventually commit to a candidate in each stage. In particular, during stage k, it must eventually declare that $\mathcal{M}_* = \mathcal{M}_{k+1}$; otherwise, the adversary may simply set $\mathcal{M}_{k+1}$ equal to the ground truth, ensuring the algorithm never succeeds.

Let i_k denote the largest input for which a passive observation has been received from $\mathcal{M}_{k+1}$ by the end of stage k. The adversary then redefines the candidate models by setting

$$\mathcal{M}_1(n) = \{1\}, \quad n \in \mathbb{Z}, \qquad \mathcal{M}_{k+1}(n) = \begin{cases} \{1\}, & n \leq i_k, \\ \emptyset, & n > i_k. \end{cases}.$$

Under this construction, the ground truth map remains $\mathcal{M}_1$, while the algorithm's guess, $\mathcal{M}_{k+1}$, is a proper subset of $\mathbb{Z}$. Consequently, $\mathcal{M}_{k+1} \neq \mathcal{M}_*$. Since the adversary can repeat this procedure at every stage, an infinite sequence of map identification problems is generated in which the algorithm is incorrect in every stage. This contradiction shows that no universal algorithm can succeed when candidate models are observed only passively.

□

D Proof of Theorem 2

Proof. We break the proof into two parts.

Part I: Map prediction is universally possible with active observations of the candidate models. First, we note that an algorithm can mimic a passive observation of a map $\mathcal{M}$ by making finitely many active observations of it. To do so, given an enumeration of $\mathcal{X} = \{f_1, f_2, \ldots\}$, we can make active observations with inputs $f_1, f_2, \ldots$, until we see an element $g \in \mathcal{M}f_j$ for some $j \geq 1$. We then take (f_j, g) as a passive observation. Hence, it suffices to assume that the algorithm collects passive observations of the ground-truth map and active observations of candidate models.

Let $(f_{j_1}, g_{j_1}), (f_{j_2}, g_{j_2}), \ldots$ be the enumerated passive observations of the ground-truth map $\mathcal{M}_*$. In the tth round of the map prediction game, let f_{ℓ_t} be the "adversary's selected input." We define $\mathcal{C}_t$ to be the set of feasible candidate models, where $\mathcal{M}_j \in \mathcal{C}_t$ if the following three conditions hold:

1. (finiteness) $1 \leq j \leq t$,
2. (consistent) $g_{j_i} \in \mathcal{M}_j f_{j_i}$ for every $1 \leq i \leq t$ and $\mathcal{M}_j f_{\ell_t} \neq \emptyset$,
3. (minimality) for every $\mathcal{M}_{i'}$ such that $1 \leq i' \leq j$, $g_{j_i} \in \mathcal{M}_{i'} f_{j_i}$ for every $1 \leq i \leq t$, and $\mathcal{M}_{i'} f_{\ell_t} \neq \emptyset$, we have that $\mathcal{M}_j f_{\ell_t} \subset \mathcal{M}_{i'} f_{\ell_t}$.

Since we assume that $\mathcal{M}_j f$ is a set with finite cardinality for all $j \geq 1$ and $f \in \mathcal{X}$, the algorithm can compute $\mathcal{C}_t$ by making finitely many active observations of the candidate models. If $\mathcal{C}_t$ is empty, then we return an arbitrary element of $\mathcal{Y}$ as our guess for an object in $\mathcal{M}_* f_{\ell_t}$. Otherwise, let $\mathcal{M}_{j_t}$ be the largest-indexed operator in $\mathcal{C}_t$. By the consistency property, we know that $\mathcal{M}_{j_t} f_{\ell_t}$ is non-empty. We design the algorithm to make an active observation of $\mathcal{M}_{j_t}$ with the input f_{ℓ_t} and return an arbitrary output from $\mathcal{M}_{j_t} f_{\ell_t}$ as our current best guess for $\mathcal{M}_* f_{\ell_t}$.

To prove the algorithm always selects a $g \in \mathcal{M}_* f_{\ell_t}$ in the "prediction" step after a finite number of rounds, we suppose that $\mathcal{M}_* = \mathcal{M}_{j_*}$ for some $j_* \geq 1$. First, we show that after some $t_0 \geq 1$, we have $\mathcal{M}_{j_*} \in \mathcal{C}_t$ for every $t \geq t_0$. To see this, for every $j < j_*$, if $\mathcal{M}_{j_*}$ is not a restriction of $\mathcal{M}_j$ at every $f \in \mathcal{X}$, then there exists some $f \in \mathcal{X}$ and $g \in \mathcal{Y}$ such that $g \in \mathcal{M}_{j_*} f$ but $g \notin \mathcal{M}_j f$. Eventually, the algorithm passively observes (f, g) from the ground-truth. Hence, each $\mathcal{M}_j$ that appears before $\mathcal{M}_{j_*}$ either extends $\mathcal{M}_{j_*}$, i.e., $\mathcal{M}_{j_*} f \subset \mathcal{M}_j f$ for every $f \in \mathcal{X}$, or eventually violate the consistency property. That makes $\mathcal{M}_{j_*}$ satisfy the minimality assumption after finitely many rounds. Clearly, $\mathcal{M}_{j_*}$ always satisfies the consistency property and the finiteness property by taking $t_0 \geq j_*$. This proves that $\mathcal{M}_{j_*} \in \mathcal{C}_t$ for every $t \geq t_0$. Hence, at the tth iteration for any $t \geq t_0$, if we let $\mathcal{M}_k$ be the largest-indexed operator in $\mathcal{C}_t$, then by the minimality assumption, we must have that $\mathcal{M}_k f_{\ell_t} \subset \mathcal{M}_{j_*} f_{\ell_t} = \mathcal{M}_* f_{\ell_t}$. This shows that the algorithm makes the map prediction with $\mathcal{M}_*$ possible.

Part II: Map prediction is not universally possible without actively observing the candidate models. As explained earlier, an algorithm can mimic a passive observation by testing and mimic a passive observation or perform a test by making active observations. Hence, all we have to show is an algorithm that collects active observations of the ground-truth map and performs tests of the candidate models cannot universally learn an operator. Seeking a contradiction, assume an algorithm exists. We set $\mathcal{X} = \mathbb{N}_{\geq 0}$ and $\mathcal{Y} = \mathbb{N}$. We will consider an execution of the algorithm on an infinite sequence of map prediction problems, so that the first k of them look identical to the algorithm until the end of the kth iteration. We then construct an additional map prediction problem that cannot be solved by the algorithm provided that the algorithm correctly solve the sequence of map prediction problems we defined. Throughout our construction, we assume that for any operator $\mathcal{M}$, we have that $\mathcal{M}0 = \{1\}$. Hence, in the following proof, we only need to specify an operator on $\mathbb{N}$.

- **Iteration 1:** As the base case, in the first iteration, when the algorithm actively observes the ground-truth map with input k for some $k \geq 1$, they always get the empty set $\emptyset$ so that $\mathcal{M}_* k = \emptyset$. At some

point, the algorithm must stop and take the adversary's question.[1] Let k_1 be the largest input that has been actively observed by the algorithm. If the algorithm does not make any observation of the ground truth, then we set $k_1 = 0$. The adversary asks the algorithm to compute an output from $\mathcal{M}_* f_{f_1}$, where $f_1 = k_1 + 1$. Set $s_1 = 2$. Consider the following problem, where $\mathcal{M}_* = \mathcal{M}_{s_1}$ and $g \in \mathcal{M}_i f$ if and only if $i = s_1$, $f = f_1$, and $g = 1$. At some point, the algorithm must stop testing and guess $g_1 \in \mathcal{M}_* f_1$ for some $g_1 \geq 1$. Let h_1 be an element of $\mathcal{Y}$ that is not equal to g_1 and has not been tested by the algorithm so far.

- **Iteration k:** Assume that for every $1 \leq t \leq k-1$, f_t is the input that the adversary picked in the tth iteration and the algorithm guessed that $g_t \in \mathcal{M}_* f_t$. Suppose that $h_t \neq g_t$ for every $1 \leq t \leq k-1$ and that h_t was not tested by the algorithm until the end of the tth iteration. In the kth iteration, when the algorithm actively observes $\mathcal{M}_*$ with input f for some $f \geq 1$: if $f = \ell_t$ for some $t \leq k-1$, we return h_t as the only active observation; otherwise, the algorithm gets that $\mathcal{M}_* f = \emptyset$. At some point, the algorithm must stop observing the ground-truth map and let the adversary ask a question. We set $f_k > f_{k-1}$ to be an input that has never been observed or tested by the algorithm so far and let the adversary ask for an element of $\mathcal{M}_* f_k$. Set $s_k > s_{k-1}$ to be an index so that $\mathcal{M}_{s_k}$ has never been tested by the algorithm. Consider the following map prediction problem, where

$$\mathcal{M}_1 f = \begin{cases} \{h_t\}, & f = f_t \text{ for some } t \leq k-1, \\ \emptyset, & \text{otherwise,} \end{cases}$$

for any $1 \leq t \leq k$, we have

$$\mathcal{M}_{s_t} f = \begin{cases} \{h_\tau\}, & f = f_\tau \text{ for some } \tau \leq t-1, \\ \{1\}, & f = f_t, \\ \emptyset, & \text{otherwise,} \end{cases}$$

and $\mathcal{M}_i f = \emptyset$ for any other $\mathcal{M}_i$ and $f \geq 1$. In this game, we set $\mathcal{M}_* = \mathcal{M}_{s_t}$. Note that up to the end of $(k-1)$st iteration:

  - For every candidate model $\mathcal{M}_i$ and every input f such that $\mathcal{M}_i f$ has been tested by the algorithm, the output $\mathcal{M}_i f$ does not change from the $(k-1)$st problem to the kth problem.
  - For every input f such that $\mathcal{M}_* f$ has been actively observed by the algorithm, the output $\mathcal{M}_* f$ does not change from the $(k-1)$st problem to the kth problem.

  Hence, when the algorithm is executed on the kth problem, it follows exactly the first $(k-1)$ iterations that we defined and, after making finitely many tests in the kth iteration, will guess that $g_k \in \mathcal{M}_* f_k$ for some $g_k \in \mathcal{Y}$. Let h_k be an element of $\mathcal{Y}$ that is not equal to g_k and has not been tested by the algorithm so far.
- The additional problem: Consider a problem such that

$$\mathcal{M}_1 f = \begin{cases} \{h_k\}, & f = f_k \text{ for some } k \geq 1, \\ \emptyset, & \text{otherwise,} \end{cases}$$

for some $k \geq 1$, we have

$$\mathcal{M}_{s_k} f = \begin{cases} \{h_t\}, & f = f_t \text{ for some } t \leq k-1, \\ \{1\}, & f = f_k, \\ \emptyset, & \text{otherwise,} \end{cases}$$

and $\mathcal{M}_i f = \emptyset$ for any other $\mathcal{M}_i$ and $f \geq 1$. We set $\mathcal{M}_* = \mathcal{M}_1$. When the algorithm is executed, it is easy to check that it follows exactly the sequence of iterations we defined above, guessing that $g_k \in \mathcal{M}_* f_k$ for every $k \geq 1$, which is always wrong. Hence, we see that the algorithm fails on the additional problem and reach a contradiction.

□

[1] Otherwise, the algorithm never stops on a problem where $\mathcal{M}_* = \mathcal{M}_1$ is the operator so that $\mathcal{M}_* 0 = \{1\}$ and $\mathcal{M}_* j = \emptyset$ for all $j \geq 1$.

E Proof of Theorem 3

Proof. If an algorithm can identify a map, then it is obvious that it can also learn the operator as long as it makes active observations of the candidate models. Hence, from Theorem 1, the "if" direction is proved. Moreover, since it is obvious that a map prediction problem is easier than a map verification problem, from Theorem 2, we know that map verification is impossible when the algorithm makes passive observations or performs tests of the candidates. Then, the only thing that remains to be done is to show that map verification problem is impossible when the algorithm makes passive observations of the ground-truth and active observations of the candidates. To this end, we will show that if such an algorithm exists, then we can use it to solve the set identification problem (see the proof of Theorem 1), which leads to a contradiction. Let $S_1, S_2, \dots$ be a sequence of sets such that each $S_j \subset \mathcal{X}$ is a subset of a countable set $\mathcal{X}$. Let $s_1, s_2, \dots$ be an enumeration of $\mathcal{X}$ and let $s_{k_1}, s_{k_2}, \dots$ be the enumeration of the ground-truth $S_* = S_{j_*}$ that we will passively observe. We set $\mathcal{Y} = \{1\}$ define a sequence of operators on $\mathcal{X}$ by

$$\mathcal{M}_i s = \begin{cases} \{1\}, & s \in S_i, \\ \emptyset, & s \notin S_i. \end{cases}$$

Then, $\{(s_{k_i}, 1)\}_{i=1}^{\infty}$ is an enumeration of solutions of $\mathcal{M}_* := \mathcal{M}_{j_*}$. In our map verification algorithm, we let the adversary pick

$$f^{(1)}, f^{(2)}, \dots = \underbrace{s_1}_{B_1}, \underbrace{s_1, s_2}_{B_2}, \underbrace{s_1, s_2, s_3}_{B_3}, \dots, \underbrace{s_1, \dots, s_n}_{B_n}, \dots$$

as the queried inputs. When the algorithm makes an active observation of $\mathcal{M}_i$ with input s, we use the set identification algorithm to check if $s \in S_i$. The algorithm will see that $\mathcal{M}_i s = \{1\}$ if $s \in S_i$ and $\mathcal{M}_i s = \emptyset$ otherwise. After the algorithm answers the adversary's questions in a full B_n, the set identification algorithm guesses S_* to be S_i with the smallest i such that $1 \le i \le n$ and $S_i \cap \{s_j\}_{j=1}^n$ are exactly the inputs in B_n of which the algorithm guesses an output of $1 \in \mathcal{Y}$. If no such set exists, then return an arbitrary one. By our assumption of the map verification algorithm, there exists some $n_0 \ge 1$ such that we always make the right guess for the entire B_n for any $n \ge n_0$. For any $j < j_*$, since $S_j \ne S_{j_*}$, there exists an element $s_{d_j} \in S_j \triangle S_{j'}$. Set $n_1 = \max\{d_1, \dots, d_{j_*-1}, n_0\}$. Then, for any $n \ge n_1$, we consider what happens after the full B_n is answered by the algorithm. For every $j \le j_*$, either S_j contains s_{d_j} but the algorithm guesses that $\mathcal{M}_* s_{d_j} = \emptyset$ or S_j does not contain s_{d_j} but the algorithm guesses that $1 \in \mathcal{M}_* s_{d_j}$. On the other hand, S_{j_*} always contains exactly those of $s_1, \dots, s_n$ for which the algorithm guesses 1. That means our set identification algorithm will always return the correct ground-truth S_{j_*} after B_{n_1} is encountered, which is a contradiction to the fact that there exists no set identification algorithm. □

F Proof of Theorem 4

Proof. We prove the claim in two parts. In Part I we show that map identification is universally solvable for single-valued maps when both the ground-truth map and the candidate models are observed passively. In Part II, we explain why the same conclusion holds regardless of whether the observations are passive, active, or via testing and why map prediction and map verification also follow immediately.

Part I: Map identification is universally solvable with passive observations. Since every operator $\mathcal{M}$ is single-valued, for every $f \in \mathcal{X}$ the output $\mathcal{M}f$ is a singleton set. The algorithm proceeds as follows. It begins by setting its current candidate guess to $\mathcal{M}_1$. We then partition the execution into stages, where the kth stage is dedicated to testing whether $\mathcal{M}_k$ is equal to $\mathcal{M}_*$. During each round of the game in the kth stage, the algorithm passively observes a new input f and records the corresponding two outputs: (i) $g \in \mathcal{M}_* f$ and (ii) $\hat{g} \in \mathcal{M}_k f$.

Let $\mathcal{D}_*$ and $\mathcal{D}_k$ denote the sets of observed input-output pairs for $\mathcal{M}_*$ and $\mathcal{M}_k$, respectively. If for some input f the outputs differ, i.e., $g \neq \hat{g}$, then the algorithm concludes that $\mathcal{M}_k \neq \mathcal{M}_*$ and immediately discards $\mathcal{M}_k$, moving on to the next candidate $\mathcal{M}_{k+1}$, terminating stage k and entering stage $k+1$.

Now, suppose that $\mathcal{M}_* = \mathcal{M}_{k_*}$ for some $k_* \geq 1$. Then, for every candidate $\mathcal{M}_k$ with $k < k_*$ the single-valuedness of the maps guarantee that there exists at least one input $f \in \mathcal{X}$ for which

$$\mathcal{M}_* f \neq \mathcal{M}_k f.$$

Since a passive observation requires the inputs to be enumerated (see Definition 2), it will take at most a finite number of rounds before the algorithm will observe a discrepancy and discard $\mathcal{M}_k$. When stage k_* is reached, the algorithm will correctly identify the map in each subsequent round and no discrepancy will ever be observed. Therefore, the algorithm eventually selects the correct candidate model and remains with it thereafter.

Part II: Extension to all types of observations and all ML goals. The above algorithm assumes passive observations for both the ground-truth map and candidate models. However, if the observations are made actively or via testing, the algorithm can easily mimic passive observations. Hence, map identification is universally possible for any types of observations. Furthermore, because map prediction and map verification are essentially equivalent to map identification for single-valued maps, all ML goals are universally solvable under any types of observations. □

G Proof of Theorem 5

Proof. We break our proof into three parts, corresponding to the three statements of the theorem, respectively.

Part I: map identification. To show the "if" direction, we can apply exactly the same algorithm proposed in Theorem 1 when the algorithm tests the ground-truth and candidate models. To prove the "only if" direction, we first note that since the finite-output map identification problem is easier than the infinite-output one, Theorem 1 immediately says that infinite-output map identification is impossible when the algorithm makes passive observation of the ground-truth or the candidate models. Hence, it remains to show that map identification is impossible when the ground-truth is actively observed and the candidate models are actively observed or tested, or when the ground-truth is tested and the candidate models are actively observed. Since for infinite-output operators, neither an active observer nor a test performer is strictly stronger than the other, we have to prove the three cases separately.

I(i): map identification is impossible when the algorithm makes active observations of the ground-truth and the candidate models. We defer this case to part II, because when the algorithm can actively observe the candidate models, the map prediction problem is clearly easier than the map identification problem; proving the infeasibility of winning the former problem will also do the same for the latter one.

I(ii): map identification is impossible when the algorithm makes active observations of the ground-truth and performs tests of the candidate models. If such an algorithm exists, we will use it to solve the set identification problem defined in the proof of theorem 1. Let $S_1, S_2, \ldots$ be a sequence of distinct candidate sets and let $S_* = S_{j_*}$ be a ground-truth set that is equal to one of the candidates. Without loss of generality, we assume that S_j is countably infinite for every $j \geq 1$. Let $s_1, s_2, \ldots$ be an enumeration of S_*. We define $\mathcal{X} = \{f_1, f_2, \ldots\}$ and $\mathcal{Y} = \bigcup_{j=1}^{\infty} S_j$. Imagine that the candidate operators $\mathcal{M}_j$ are defined by

$$\mathcal{M}_j f_1 = S_j, \quad \mathcal{M}_j f_i = \emptyset, \qquad j \in \mathbb{N}, \quad i \geq 2,$$

and the ground-truth operator is $\mathcal{M}_{j_*}$. We apply the map identification algorithm: whenever the algorithm makes an active observation of the ground-truth $\mathcal{M}_*$ with some f_i, if $i \geq 2$, we return the emptyset; otherwise, if $i = 1$, we query an element s from S_* and answer that $s \in \mathcal{M}_* f_1$. When the algorithm tests if $s \in \mathcal{M}_j f_i$, we answer yes if and only if $i = 1$ and $s \in S_j$, which is determined from the set identification algorithm. Let the set identification algorithm guess S_j for the $j \in \mathbb{N}$ such that $\mathcal{M}_j$ is guessed by the algorithm in the map identification game. Clearly, the algorithm guesses correctly if and only if the set identification algorithm guesses correctly. We reach a contradiction.

I(iii): map identification is impossible when the algorithm performs tests of the ground-truth and makes active observations of the candidate models.

In this section, we first consider the following question: If I have a sequence of sets $S_1, S_2, \ldots$, each of which is a subset of $\mathbb{N}$. I know exactly one of them is equal to $\mathbb{N}$. An adversary enumerates every set S_j. When I ask for an element of S_j, the next unseen element in the enumeration will be given. Then, is there an algorithm for me to identify which S_j is equivalent to $\mathbb{N}$? (As before, we assume that this game is played in iteration. That is, I query finitely many sets and make a guess. This counts as an iteration. I can identify the set if, after a certain point, my guess is alway correct.)

It is easy to see that one can reduce the set identification game to a map identification game with test/ground-truth and active/candidates. Namely, we consider a problem where $\mathcal{X} = \{\star\}$ is a singleton and $\mathcal{M}_j \star = S_j$, enumerated by the adversary in the same way as in the set identification game. Let $\mathcal{M}_* \star = \mathbb{N}$. (We do not even need to let the algorithm test the ground-truth. We can directly tell the algorithm that $\mathcal{M}_* \star = \mathbb{N}$.) When the map identification algorithm makes an active observation of a candidate $\mathcal{M}_j \star$, it is as if a set identification algorithm asks for an element of S_j. Hence, if the map identification algorithm can identify which $\mathcal{M}_j$ is equivalent to $\mathcal{M}_*$, then we can use this algorithm to identify which S_j is equivalent to $\mathbb{N}$.

Now, we will show that the set identification problem is infeasible, which shows that the map identification problem (whose solution, if exists, would lead to a set identification algorithm) is infeasible either. It works as follows:

1. First, we hold out an element, say $1 \in \mathbb{N}$. Let $S_1 = \{2, 3, 4, \ldots\}$ and $S_2 = \{1, 2, 3, \ldots\}$. When the algorithm is executed on this problem, it needs to say that $S_2 = \mathbb{N}$ at some point. At this moment, assume that $\{2, 3, \ldots, j_1\}$ are seen from S_1 and $\{1, 2, \ldots, k_1\}$ are seen from S_2. (Remark: without loss of generality, we can assume that the algorithm only queries S_1 and S_2 in this execution. If it queries another set, say S_i, then we can simply assume that $S_i = \emptyset$ and ignore it thereafter.)
2. Now, the idea is that we want to trick the algorithm by making S_2 the wrong answer. This is simple. In the next problem, we can simply set $S_2 = \{1, 2, \ldots, k_1\}$. We define $S_3 = \{1, 2, 3, \ldots\}$. (If S_3 is queried in the previous execution, then we use S_4 instead, and so on. See the remark above.) For S_1, we change its definition to $S_1 = \{2, 3, \ldots, j_1, 1, j_1 + 2, j_2 + 3, \ldots\} = \mathbb{N} \setminus \{j_1 + 1\}$. Intuitively, we add back 1 but hold out $j_1 + 1$. We add back 1 because in our final problem (the diagonalization), we want $S_1 = \mathbb{N}$ to be the correct answer; we hold out $j_1 + 1$ because in the current problem, we want $S_3 = \mathbb{N}$ to be the correct answer. When the algorithm is executed, it is clear that it follows the first stage up to the point that both $j_1 \in S_1$ and $k_1 \in S_2$ are seen. After that, if the algorithm queries S_2 again, then we will loop back to 1 in this setting and output $k_1 + 1$ in the previous setting, but this no long matters. The thing is that at some point, the algorithm will guess that $S_3 = \mathbb{N}$. At this moment, assume that $\{2, 3, \ldots, j_1, 1, j_1 + 2, j_1 + 3, \ldots, j_2\}$ are seen from S_1 and $\{1, 2, \ldots, k_2\}$ are seen from S_3.
3. At this point, you can expect what happens in the next problem. We set $S_2 = \{1, 2, \ldots, k_1\}$ and $S_3 = \{1, 2, \ldots, k_2\}$. Let $S_4 = \mathbb{N}$ be the new ground-truth. For S_1, we add back $j_1 + 1$ but hold out $j_2 + 1$. That is, its enumeration would be

$$S_1 = \{\underbrace{2, 3, \ldots, j_1}_{\text{first stage}}, \underbrace{1, j_1 + 2, j_1 + 3, \ldots, j_2}_{\text{second stage}}, j_1 + 1, j_2 + 2, j_2 + 3, \ldots\}.$$

   The rest follows the same pattern. Intuitively, why this would lead to a contradiction is that whenever you hold out an element of $\mathbb{N}$ from S_1, you will always add it back in the next problem; hence, in the infinite limit, no element is left out and we thus construct an enumeration of $S_1 = \mathbb{N}$.

Now, it all remains to define a final problem, for which $S_1 = \mathbb{N}$, but the algorithm guesses incorrectly infinitely often. To do this, let $S_j = \{1, 2, \ldots, k_{j-1}\}$ ($j \geq 2$) be whatever the finite set defined in the jth stage. Consider the following enumeration of S_1: first, let $B_i = \{j_i, \ldots, j_{i+1} - 1\}$ be the block defined by

$$\underbrace{1 (= j_0), 2, \ldots, j_1 - 1}_{B_0}, \underbrace{j_1, j_1 + 1, \ldots, j_2 - 1}_{B_1}, \underbrace{j_2, j_2 + 1, \ldots, j_3 - 1}_{B_2}, j_3 \ldots.$$

Consider the enumeration of B_i: $B_i' = \{j_i + 1, \ldots, j_{i+1} - 1, j_i\}$ (i.e., rotating the first element to the end). Then, $B_0', B_1', B_2', \ldots$ is apparently an enumeration of $S_1 = \mathbb{N}$. However, when executing the algorithm on this enumeration, we follow exactly the sequence of stages defined above, guessing that $S_2, S_3, \ldots$ are the ground-truth. That is, the algorithm makes incorrect guesses infinitely often.

Assume such an algorithm exists. Let $\mathcal{X} = \{f\}$ be a singleton and let $\mathcal{Y} = \mathbb{N}$. We partition the execution into stages, each of which contains multiple iterations of the game, and set up an infinite sequence of map identification problems, so that the first k of them look identical to the algorithm until the end of the kth stage. For each of these map identification problems, we assume that the ground-truth operator is $\mathcal{M}_*$ such that $\mathcal{M}_* f = \mathbb{N}$. Then, we define an additional map identification problem and show that the algorithm must fail to solve it provided that it solves the sequence of problems we defined.

- **Stage 1:** Set $s_1 = 2$ and $j_1 = 0$. We define the candidate models to be defined by

$$\mathcal{M}_i f = \begin{cases} \mathbb{N} \setminus \{j_i + 1\}, & i = 1, \\ \mathbb{N}, & i = s_1, \\ \emptyset, & \text{otherwise.} \end{cases}$$

Let $j_1+2, j_1+3, \ldots$ be the adversary's enumeration of $\mathcal{M}_1 f$ and let $1, 2, \ldots$ be the adversary's enumeration of $\mathcal{M}_{s_1} f$. Since the algorithm is correct, at some iteration, it must guess that $\mathcal{M}_* = \mathcal{M}_{s_1}$. After this happens, we stop the first stage and enter the second stage.

- **Stage k:** Assume that the algorithm guesses $\mathcal{M}_* = \mathcal{M}_{s_t}$ in the tth stage for every $1 \leq t \leq k-1$. Let $s_k > s_{k-1}$ be an index for which $\mathcal{M}_{s_k}$ has never been actively observed up to the end of stage $k-1$. Let $j_k \in \mathcal{Y}$ be the largest output that has been actively observed from $\mathcal{M}_1 f$ up to the end of stage $k-1$. If no active observation of $\mathcal{M}_1 f$ has been made yet, then we set $j_k = 0$. We define the kth map identification problem by choosing the candidate models as follows:

$$\mathcal{M}_i f = \begin{cases} \mathbb{N} \setminus \{j_k\}, & i = 1, \\ \emptyset, & i \geq 1, i \notin \{s_1, s_2, \ldots\}. \end{cases}$$

  For $i = s_t$, where $1 \leq t \leq k-1$, we define $\mathcal{M}_i f$ to be the finite set of all elements actively observed at the end of the $(k-1)$st stage. The adversary enumerates $\mathcal{M}_1 f$ by following the order that the set $\{1, \ldots, j_k\} \setminus \{j_{k-1}\}$ is enumerated in the first $(k-1)$ rounds, and then j_{k-1}, and then $j_k + 2, j_k + 3, \ldots$. Since the algorithm is correct, at some iteration, it must guess that $\mathcal{M}_* = \mathcal{M}_{s_k}$. After this happens, we stop the kth stage and enter the next ones.
- **The additional problem:** We define our additional problem by letting $\mathcal{M}_1 f = \mathbb{N}$ and $\mathcal{M}_i f = \emptyset$ for all $i \notin \{s_1, s_2, \ldots\}$. Given some $k \geq 1$, we note that the definition of $\mathcal{M}_{s_k} f$ is never changed after the $(k+1)$st stage. We take that as our definition of $\mathcal{M}_{s_k} f$ in our additional problem and the order that elements in $\mathcal{M}_{s_k} f$ are enumerated in the first k stages as the adversary's enumeration of it. We enumerate $\mathcal{M}_1 f$ by following how $\mathbb{N}$ is enumerated in the sequence of stages, i.e.,

$$\mathbb{N} = \{j_1 + 2, j_1 + 3, \ldots, j_2, j_1, j_2 + 2, j_2 + 3, \ldots, j_3, \ldots, j_k + 2, j_k + 3, \ldots, j_{k+1}, j_k, j_{k+1} + 2, \ldots\}.$$

  When the algorithm is executed on this additional problem, it follows the sequence of stages we defined, returning $\mathcal{M}_{s_j}$ as its guess for every $j \geq 1$, which is false. Hence, the algorithm is not correct on the additional problem and we reached a contradiction.

Part II: map prediction. Similar to part I, we only need to consider the cases when the algorithm makes active observations of the candidate models since other cases follow from Theorem 2.

II(i): map prediction is possible when the algorithm performs tests of the ground-truth and makes active observations of the candidate models. We defer this case to part III, because a map verification problem is clearly harder than a map prediction problem. Hence, proving the feasibility of map verification problems in this setting will automatically prove the feasibility of map prediction problems.

II(ii): map prediction is impossible when the algorithm makes passive or active observations of the ground-truth and makes active observations of the candidate models. Since every finite-output problem is also regarded as an infinite-output problem in our setting, by Theorem 2, we only need to prove the impossibility when we actively observe the candidates and passively or actively observe the ground-truth. Moreover, since we can still use finitely many active observations to obtain a passive observation, even for infinite-output operators, it suffices to show that map prediction is impossible when the algorithm makes active observations of the ground-truth and candidate models. Assume such an algorithm exists. We let $\mathcal{X} = \mathcal{Y} = \mathbb{N}$. We partition the execution into stages, each of which contains multiple iterations of the game, and set up an infinite sequence of map prediction problems, so that the first k of them look identical to the algorithm until the end of the kth stage. Each map prediction problem we define has a different ground-truth operator $\mathcal{M}_*$ and candidate models $\mathcal{M}_1, \mathcal{M}_2, \ldots$. Then, we use this sequence of map prediction problems to construct another problem so that when the algorithm is executed on this additional problem, it exactly follows the sequence of stages that we defined. We will show that as long as the algorithm correctly solves the sequence of problems, it must fail on the additional map prediction problem we construct, which leads to a contradiction. We now define the sequence of stages and map prediction problems inductively.

The high-level idea in this construction is the same as the one above: eventually, we want $\mathcal{M}_1$ to be the ground-truth; however, in every intermediate stage, we assume that a different candidate is the ground-truth. After the algorithm commits to that, in the following stages, we trick the algorithm and no longer make it the ground-truth.

In the first stage, I have in mind that $\mathcal{M}_1 i =$ all evens, $\mathcal{M}_2 i =$ all odds for all i, and $\mathcal{M}_* = \mathcal{M}_2$. The very important trick is that when the adversary picks an input i, $\mathcal{M}_* i$ has never been queried before. (Note that the order in each iteration is query ground-truth $\to$ raise question $\to$ query candidates $\to$ answer.) Hence, at the point the algorithm eventually needs to guess that $i_1 \in \mathcal{M}_* j_1$ for some odd i_1, no element of the set $\mathcal{M}_* j_1$ has been revealed (this is very important). This opens room for cheating, because I can now say "I changed my mind and $\mathcal{M}_* j_1$ is in fact the set of even numbers," which does not violate any information we have previously seen (only true because $\mathcal{M}_* j_1$ has not been queried up to this point). As you can see, this is how I can make the algorithm produce a guess that is inconsistent with $\mathcal{M}_1$ in the first stage.

The next stage follows basically the same pattern. To see how it works, we can simply do the same procedure (now with $\mathcal{M}_3$ instead of $\mathcal{M}_2$) on the untouched inputs. That is, suppose in the first stage, the first 10 elements have been queried (either from $\mathcal{M}_1$, $\mathcal{M}_2$, or $\mathcal{M}_*$) and $j_1 = 10$. Then, I have in mind that $\mathcal{M}_1 i =$ all evens, $\mathcal{M}_3 i =$ all odds for all $i \geq 11$, and $\mathcal{M}_* = \mathcal{M}_3$ and play exactly the same cheating game. The way I can make $\mathcal{M}_1$ and $\mathcal{M}_*$ agree on the first 10 inputs (while also maintaining the entire execution in the first stage; this is important) is by setting $\mathcal{M}_1 j = \mathcal{M}_* j = \mathbb{N}$ for $j = 1, 2, \ldots, 9$ (after all, you only saw finitely many elements in stage 1, and you were just totally unlucky to see all even numbers from $\mathcal{M}_1$ and all odd numbers from $\mathcal{M}_*$), and $\mathcal{M}_1 10 = \mathcal{M}_* 10 =$ all evens (this is okay because $\mathcal{M}_* 10$ is never queried in stage 1, so you can define it however you like). After the algorithm needs to guess that $i_2 \in \mathcal{M}_* j_2$ for some odd i_2 and j_2 for which $\mathcal{M}_* j_2$ has never been queried, we cheat by saying that "$\mathcal{M}_* j_2$ is in fact the set of even numbers," which produces an inconsistency between $\mathcal{M}_1 j_2$ and the algorithm's guess.

Now, you see the core idea. After each stage, I can make $\mathcal{M}_1$ and $\mathcal{M}_*$ agree on all inputs that have been queried up to this point, and play the cheating game on the untouched inputs. Since there are infinitely many stages, each input must belong to some stage, meaning that $\mathcal{M}_1$ and $\mathcal{M}_*$ must eventually agree on it. That means in the infinite limit, we will have $\mathcal{M}_* = \mathcal{M}_1$ (because they agree on every input; this is the diagonalization), but when the algorithm is executed, it is tricked infinitely often, leading to a contradiction.

- **Stage 1:** Set $s_1 = 2$. We define the first map prediction problem to have a ground-truth operator $\mathcal{M}_*$ so that $\mathcal{M}_* j$ is the set of odd natural numbers for every $j \in \mathcal{X}$. Define $\mathcal{M}_1$ so that $\mathcal{M}_* j$ is the set of all even natural numbers for every $j \in \mathcal{X}$, $\mathcal{M}_{s_1} = \mathcal{M}_*$, and $\mathcal{M}_i$ for $i \geq 3$ so that $\mathcal{M}_* j = \emptyset$ for every $j \in \mathcal{X}$. When the algorithm makes an active observation of the ground truth $\mathcal{M}_*$ with an input $j \in \mathcal{X}$, the outputs it receives form an enumeration of the odd numbers. Whenever it is the adversary's turn to ask a question, they pick an input $j \in \mathcal{X}$ that has not appeared in the algorithm's observations so far. We also assume the input picked by the adversary is larger than all inputs they have asked before. When the algorithm makes an active observation of $\mathcal{M}_1 j$ with any j, it receives an enumeration of even natural numbers. When the algorithm makes an active observation of $\mathcal{M}_{s_1} j$ with any j, it receives an enumeration of odd natural number. When the algorithm makes an active observation of any other operator with any input, it is informed that the output set is empty. Since the algorithm is correct, at some point, the algorithm must guess an odd number, say $i_1 \in \mathcal{M}_* j_1$ for some odd i_1. We terminate the first stage and enter the second stage.
- **Stage k:** Assume that the algorithm guesses that $i_t \in \mathcal{M}_* j_t$ in the tth stage for every $1 \leq t \leq k-1$. Also let s_t be the index such that $\mathcal{M}_* = \mathcal{M}_{s_t}$ in the tth map prediction game for every $1 \leq t \leq k-1$. We define our kth map prediction problem as follows. Let $\mathcal{M}_*$ be defined by

$$\mathcal{M}_* j = \begin{cases} \mathbb{N}, & j \leq j_{k-1}, j \notin \{j_1, \ldots, j_{k-1}\}, \\ \text{the set of all even natural numbers}, & j \in \{j_1, \ldots, j_{k-1}\}, \\ \text{the set of all odd natural numbers}, & j > j_{k-1}. \end{cases}$$

  Let s_k be an index such that $s_k > s_{k-1}$ and $\mathcal{M}_{s_k}$ has never been observed by the algorithm in the first $(k-1)$ stages. Define $\mathcal{M}_1$ to be

$$\mathcal{M}_1 j = \begin{cases} \mathbb{N}, & j \leq j_{k-1}, j \notin \{j_1, \ldots, j_{k-1}\}, \\ \text{the set of all even natural numbers}, & \text{otherwise}. \end{cases}$$

  Define $\mathcal{M}_{s_k} = \mathcal{M}_*$. For any other $i \neq 1, s_k$, we can simply define $\mathcal{M}_i$ to be the operator whose valid input-output pairs have all been actively observed by the algorithm in the first $(k-1)$ stages. Given

this definition, when the algorithm executes, there is a way for the adversary to enumerate the active observations so that the algorithm follows exactly the first $(k-1)$ stages. After that, whenever it is the adversary's turn to ask a question, they pick an input $j \in \mathcal{X}$ that has not appeared in the algorithm's observations so far. We also assume the input picked by the adversary is larger than all inputs they have asked before (including those asked in the first $k-1$ stages). In particular, that means for every $j \in \mathcal{X}$ that the adversary asks, we have $\mathcal{M}_* j$ is the set of all odd natural numbers. At some iteration, the algorithm must guess an odd number, say $i_k \in \mathcal{M}_* j_k$ for some odd $i_k > i_{k-1}$. We terminate the kth stage and enter the second stage.

- **The additional problem:** Now, we define an additional map prediction problem for which the algorithm must fail. We set $\mathcal{M}_* = \mathcal{M}_1$ to be

$$\mathcal{M}_* j = \mathcal{M}_1 j = \begin{cases} \mathbb{N}, & j \notin \{j_1, j_2, \ldots\}, \\ \text{the set of all even natural numbers}, & \text{otherwise.} \end{cases}$$

  Note that we have $s_1 < s_2 < \cdots$. Hence, for any $i \geq 2$, we must have that $i < s_k$ for some $k \geq 1$. Note also that the definition of $\mathcal{M}_i$ has remained the same in the tth map prediction game for any $t \geq k$. We take that as the definition of $\mathcal{M}_i$ in our additional map prediction problem. It is easy to see that when the algorithm executes on this additional problem, it follows exactly the sequence of stages we defined, guessing $i_t \in \mathcal{M}_* j_t$ for $t = 1, 2, \ldots$, which is wrong since i_t is odd. Hence, the algorithm is incorrect on the additional problem and we have reached a contradiction.

Part III: map verification. The "only if" direction follows immediately from the "only if" direction of part II. To show the "if" direction, we construct an algorithm for map verification when the algorithm performs tests of the ground-truth and makes active observations of the candidate models. We break our algorithm into stages, where each stage contains multiple iterations. We start with the first stage, and since the stages in this algorithm are mutually independent, we explain what the algorithm does in a single stage, which completes the description of the algorithm. At the beginning of the kth stage, we initialize the set $V_k = \emptyset$. In the tth iteration of the kth stage, when the adversary picks some $f \in \mathcal{Y}$, the algorithm makes an active observation of $\mathcal{M}_k$ with this f. If $\mathcal{M}_k f = \emptyset$, we add f to V_k and guess that $\mathcal{M}_* f = \emptyset$. Otherwise, let g be an element of $\mathcal{M}_k f$. We guess that $g \in \mathcal{M}_* f$. In the $(t+1)$st iteration, we first test if $g_j \in \mathcal{M}_* f$ for all $j \leq t$ and $f \in V_k$. If we ever get a positive answer, then we terminate the kth stage and enter the $(k+1)$st stage. Also, if we guess that $g \in \mathcal{M}_* f$ in the tth iteration, then we also test if this is true. If not, we also terminate the kth stage and enter the $(k+1)$st stage. We claim that this algorithm is correct. To see this, in the kth stage, if we always answer the question correctly, then we are done. Otherwise, assume we answer $\mathcal{M}_* f$ incorrectly. There are two possibilities. The first is that we answer $g \in \mathcal{M}_* f$ but it is not. In that case, we will terminate the kth stage immediately when we test that $g \notin \mathcal{M}_* f$ in the next iteration. The second possibility is that we answer $\mathcal{M}_* f = \emptyset$ but it is actually not. Say, $g_j \in \mathcal{M}_* f$. Then, we must terminate the kth stage after the $(j+1)$st iteration. That is, as long as we make a wrong guess during the kth stage, we know that we will eventually move onto the $(k+1)$st stage. Assume $\mathcal{M}_* = \mathcal{M}_{k_*}$. If we enter the k_*th stage, we will never terminate it. Hence, the algorithm is correct. □