What Has a Foundation Model Found? Inductive Bias Reveals World Models

Keyon Vafa [1] **Peter G. Chang** [2] **Ashesh Rambachan** [2] **Sendhil Mullainathan** [2]

Abstract

Foundation models are premised on the idea that sequence prediction can uncover deeper domain understanding, much like how Kepler's predictions of planetary motion later led to the discovery of Newtonian mechanics. However, evaluating whether these models truly capture deeper structure remains a challenge. We develop a technique for evaluating foundation models that examines how they adapt to synthetic datasets generated from some postulated world model. Our technique measures whether the foundation model's inductive bias aligns with the world model, and so we refer to it as an *inductive bias probe*. Across multiple domains, we find that foundation models can excel at their training tasks yet fail to develop inductive biases towards the underlying world model when adapted to new tasks. We particularly find that foundation models trained on orbital trajectories consistently fail to apply Newtonian mechanics when adapted to new physics tasks. Further analysis reveals that these models behave as if they develop task-specific heuristics that fail to generalize.

1. Introduction

The promise of foundation models relies on a central presumption: learning to predict sequences can uncover deeper truths, or even optimistically, a world model. While this idea seems new, it is actually quite old. Astronomers like Kepler noticed geometric patterns that could be used to pinpoint the future locations of planets in the night sky. Newton would later expand on these results to develop Newtonian mechanics, fundamental laws that could not only predict the movement of planets but also explain physical properties across the universe (Koestler, 1959; Gingerich, 2004).

This path — from predicting sequences to understanding the deeper mechanisms that underlie them — is not unique to physics. In biology, animal breeders noticed patterns in the traits of offspring long before their predictive insight inspired Mendel to develop a theory of genetics.

How would we know if foundation models have also made the leap from making accurate predictions to developing reliable world models? This paper develops a framework for answering this question. Specifically, we create a procedure that, when given a foundation model and world model, tests whether the foundation model has learned that world model. We call this technique an *inductive bias probe*, and it is built on a simple insight: the implicit world model of a foundation model is revealed by how it extrapolates from a small amount of information. This is inspired by how scientists use world models — to make inferences from small amounts of data. Similarly, the inductive bias of a foundation model reveals its world model.

We first demonstrate this procedure using an example from physics. Specifically, we aim to replicate Kepler's and Newton's experiments, albeit replacing the physicist with a foundation model of orbital mechanics. Much like Kepler, the model is able to predict orbital trajectories, even for solar systems it has not seen.

What would it mean for this foundation model's inductive bias to be toward Newtonian mechanics? We demonstrate one tangible way to test this: we fine-tune the foundation model on a small dataset where the output is exactly the force vector (a cornerstone of Newtonian mechanics) at each point in the trajectory. If the foundation model's world model is toward Newtonian mechanics, it should have an inductive bias towards these force vectors. In contrast, Figure 1 shows that the model produces poor force vectors. More extremely, when we perform this exercise at a larger scale across many solar systems, the law of gravity it uses to generalize bears no resemblance to Newton's law (Table 1).

We further apply inductive bias probes in other domains with a known world model: lattice problems and Othello games (Liu et al., 2022; Hazineh et al., 2023; Nanda et al., 2023b; Vafa et al., 2024). Across these domains, we find that neural sequence models have weak inductive biases toward the given world models. We also highlight a practical

[*]Equal contribution [1]Harvard University [2]MIT. Correspondence to: Keyon Vafa <kvafa@g.harvard.edu>.

Proceedings of the 42nd International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

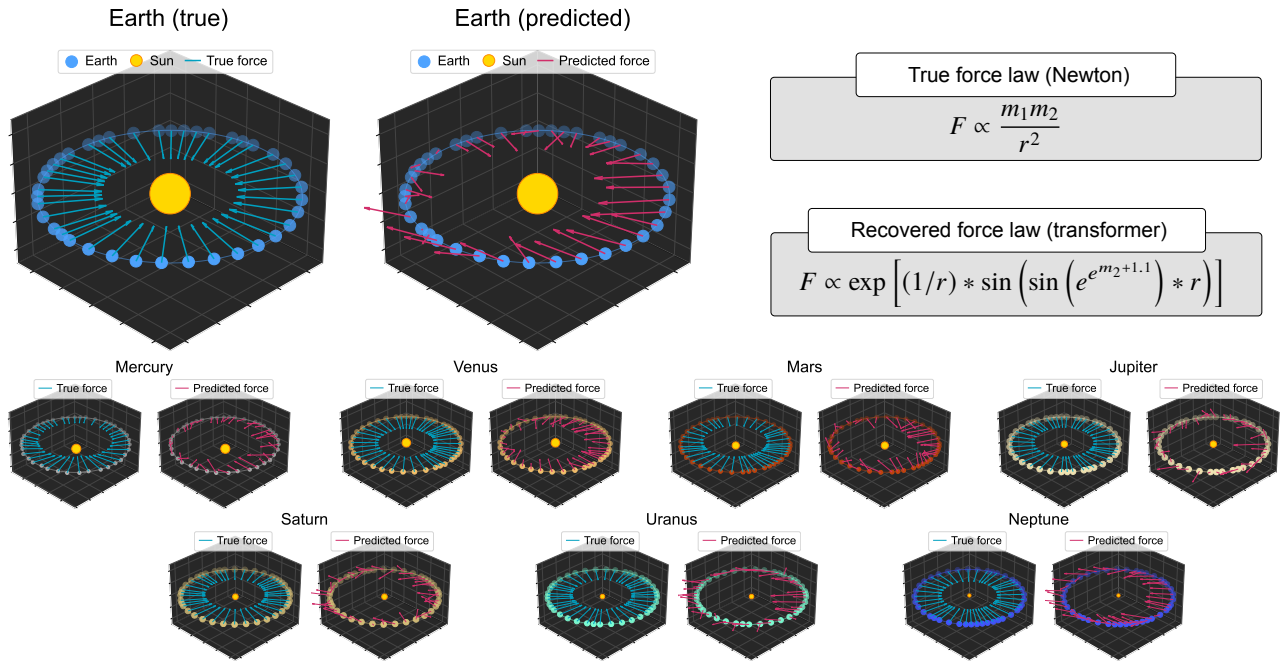

Figure 1: Each pair of panels illustrates the trajectory of a planet in the solar system and its gravitational force vectors, comparing the true Newtonian forces (left) to the predicted forces from a transformer foundation model pretrained on orbital sequences and fine-tuned to predict forces. While the model excels at generating accurate predictions of planetary trajectories (see Table 8), it does not have an inductive bias toward true Newtonian mechanics; moreover, its force predictions recover a nonsensical force law, as revealed by symbolic regression.

implication of the inductive bias probe: models with better inductive bias metrics have better performance when they're fine-tuned to perform new tasks that rely on the underlying world model.

Taken together, our results provide a direction for understanding the deficiencies of foundation models: if a model's inductive bias isn't toward a known model of reality, what is it toward? We explore this question by examining whether these foundation models have alternative inductive biases. Our analysis reveals that these models instead behave as if they develop task-specific heuristics that fail to generalize. For physics, rather than learning one universal physical law, the foundation model applies different, seemingly nonsensical laws depending on the task it's being applied to. In lattice and Othello, models have an inductive bias toward the set of legal next-tokens (e.g. a board's legal next moves) rather than the world model itself.

2. Framework

In this section, we lay out our framework for evaluating whether a foundation model has learned a postulated world model. We develop an *inductive bias probe*, which is a procedure that evaluates the foundation model's behavior as it adapts to new tasks.

Data and tasks. Let $x \in \mathcal{X}$ denote an input and $y \in \mathcal{Y}$ denote some output. A dataset $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ is a collection of n input-output pairs. A *task* $f : \mathcal{X} \to \mathcal{Y}$ is a mapping between inputs and outputs.

Foundation models: A *foundation model* is a learning algorithm which, when given a dataset D, returns a prediction function $\widehat{m}_D : \mathcal{X} \to \mathcal{Y}$ that relates the input to the outputs. Foundation models can take many forms; for example, $\widehat{m}_D$ could be some pre-trained model that is fine-tuned on the dataset D, or it can be an LLM that is supplied D in-context.

World model: A postulated *world model* is summarized by a state space Φ and a mapping $\phi : \mathcal{X} \to \Phi$ that associates each input with some state $\phi(x) \in \Phi$. A dataset D is *consistent* with the world model if for each $(x, y) \in D$, the output is a deterministic function of the state, $y = g(\phi(x))$ for some $g : \Phi \to \mathcal{Y}$.

2.1. Comparing foundation models to world models

There is a challenge in defining what it means for a foundation model to recover a world model: these two objects sit in seemingly different spaces. A foundation model uses datasets to output predictions given inputs, whereas a world model describes state structure implicit in that data.

One approach would be to mechanistically probe the foundation model, e.g. by comparing its weight-level representations to the postulated states in the world model. However, understanding the internal mechanisms of large models is challenging (Olah, 2022) and even then may not reflect how a model actually behaves on new data (Casper et al., 2023). Another approach is to study the model's behavior statically, on a single task (Vafa et al., 2024), but this doesn't capture how foundation models are used in the real world: as tools for new tasks.

We take a different approach, motivated by the no-free-lunch theorem (Wolpert, 1996). Loosely speaking, the no-free-lunch theorem states that if any function could have generated data between inputs and outputs, then no learning algorithm can perform better than another on average. Given limited data, learning algorithms must extrapolate to unseen inputs, and since any underlying function is possible, any such extrapolation must be equally good or bad. This means that every learning algorithm is better for *some* collection of possible functions — those functions that it tends to select when extrapolating from limited data. The functions that a learning algorithm tends to select represent its *inductive bias*.

The idea of inductive bias offers a connection between foundation models and world models. A world model is a restriction on the possible functions from inputs to outputs: only those that obey its state structure are possible. Consequently, a foundation model that has learned a postulated world model should have an inductive bias towards functions that obeys the world model's state structure. For example, physicists may train a foundation model on sequences of planetary orbits. Since planetary orbits obey Newtonian mechanics, they might hope the model has an inductive bias toward functions of Newtonian mechanics (e.g. predicting the force vector between two planets).

We develop an *inductive bias probe* for testing whether a foundation model's inductive bias matches the postulated world model's state structure. The inductive bias probe repeatedly applies the foundation model to synthetic datasets consistent with the world model and studies the extrapolated functions together. In each such simulation, we do not calculate the "accuracy" of the resulting extrapolations since there is no one accurate function; multiple ways to extrapolate may be allowed by the true world model. Instead, we evaluate whether the extrapolations resemble those that are allowed by the true world model.

2.2. Special case: finite state space and binary outputs.

To further describe key intuition underlying the inductive bias probe, we first consider the special case of a binary output $\mathcal{Y} = \{0, 1\}$ and a postulated world model with a finite state space Φ. The two metrics we introduce in this setting are special cases of the general inductive bias probe defined in the next section.

The inductive bias probe evaluates whether a foundation model's inductive bias is towards a postulated world model. At a high level, the probe repeatedly applies the foundation model to synthetic datasets consistent with the postulated world model and each time evaluates its predictions on held-out inputs. If the foundation model's inductive bias is towards the postulated world model, its extrapolations should have two properties. First, the foundation model's predictions should *respect state*: if two inputs x, x' map to the same state ($\phi(x) = \phi(x')$), the foundation model should have the same predicted outputs ($\widehat{m}_D(x) = \widehat{m}_D(x')$) when applied across synthetic datasets. If not, it means that the foundation model fits functions that do not belong to the world model. Second, the foundation model's predictions should *distinguish state*: if two inputs x, x' map to different states ($\phi(x) \neq \phi(x')$), the foundational model should typically have different predicted outputs ($\widehat{m}_D(x) \neq \widehat{m}_D(x')$) across synthetic datasets. If not, then the foundation model does not fit functions that fully cover the world model's allowable functions.

These properties can be measured using two metrics. Let $1(y, y')$ denote the indicator for whether $y = y'$. We specify a sampling distribution over consistent datasets $D \sim P_D$ and a sampling distribution over inputs $(X_i, X_j) \sim P_X \times P_X$. The foundation model's *inductive bias towards respecting state* (R-IB) is

$$\mathbb{E}_{X_i,X_j,D}[1(\widehat{m}_D(X_i), \widehat{m}_D(X_j)) \mid \phi(X_i) = \phi(X_j)]. \quad (1)$$

R-IB measures the similarity between the foundation model's extrapolations on inputs in the same state under the postulated world model: the higher R-IB, the more similar are its predictions in the same states. The foundation model's *inductive bias towards distinguishing state* (D-IB) is

$$1 - \mathbb{E}_{X_i,X_j,D}[1(\widehat{m}_D(X_i), \widehat{m}_D(X_j)) \mid \phi(X_i) \neq \phi(X_j)]. \quad (2)$$

D-IB measures whether inputs that belong to different states under the postulated world model nonetheless receive similar predictions by the foundation model: the higher D-IB, the more dissimilar are its predictions on different states.

Together, R-IB and D-IB provide contrasting perspectives on the foundation model's implicit world model, analogous to precision and recall in binary classification. For example, while it is trivial for a foundation model to achieve high R-IB by making the same prediction on every input, its D-IB will suffer. Both metrics are needed to contrast the foundation model's inductive bias with the postulated world model.

In this sense, the inductive bias probe captures behavior of a foundation model that is not captured by standard probe

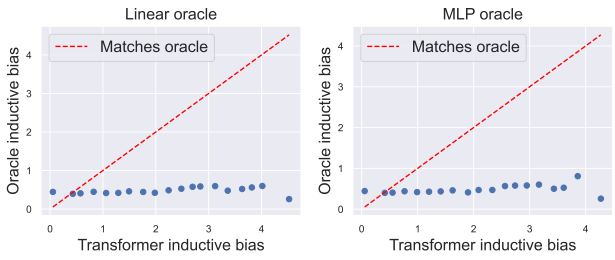

Figure 2: Inductive bias probe performance (Equation 6) for the transformer pretrained on orbital trajectories. A 45-degree line would indicate perfect inductive bias toward an oracle that extrapolates based on the Newtonian state vector.

tests (Li et al., 2024; Nanda et al., 2023b), which measures how well a simple predictive model (e.g., a linear model) can predict state from a foundation model's intermediate representation. By contrast, the inductive bias probe directly analyzes how the foundation model behaves when adapted to synthetic tasks from the postulated world model. When there are many distinct state mappings that are predictable from a foundation model's internal representation, standard probes cannot distinguish which is actually being used by the model. Moreover, the standard probe is sensitive to how state is mechanistically represented by the chosen world model. For example, Nanda et al. (2023b) find that different representations of the Othello game board (one based on the standard board and another that inverts the board based on whose turn it is) lead to different results by standard probes. By contrast, because inductive bias probes only depend on state equality, they are insensitive to equivalent representations.

To implement the inductive bias probe, a practitioner must supply a sampling distribution over consistent datasets P_D and a sampling distribution over inputs P_X. In our experiments with a finite state space and binary outputs (see Section 4), we sample consistent datasets by assigning each unique state the output 0 or 1 uniformly at random. We recommend examining the inductive bias probe for alternative choices of sampling distributions P_X.

2.3. Inductive bias probe

We now describe the inductive bias probe allowing for general outputs, state spaces, and tasks. For example, for sequences of two planets orbiting one another, the states could correspond to their relative positions, relative velocities, and the masses of each planet under Newtonian mechanics. We further introduce a collection of *admissible functions* on state $\mathcal{G}$ that govern the relationship between the state space and the output under the world model with each $g \in \mathcal{G}\colon \Phi \to \mathcal{Y}$. For example, in some settings, we may expect the output to vary smoothly with the state, in which case $\mathcal{G}$ could be the collection of K-Lipshitz functions. A dataset is now consistent with the world model if for each $(x, y) \in D$, $y = g(\phi(x))$ for some $g \in \mathcal{G}$.

Given a sampling distribution over consistent datasets P_D and a sampling distribution over inputs P_X, the inductive bias probe repeatedly applies the foundation model to sampled datasets, and then evaluates its predictions on held-out inputs. It measures how *predictable* the foundation model's predicted outputs for one input are from those of another input across many synthetic datasets. The intuition is unchanged: inputs in "similar" states should be more predictable from one another than inputs from "different" states. We next formalize this property.

Extrapolative predictability. We further specify a family of predictors $\mathcal{H}$ with $h \in \mathcal{H}$ such that $h : \mathcal{Y} \to \mathcal{Y}$ and a loss function over outputs $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$. We define the *extrapolative predictability* between two inputs as

$$\widehat{I}(x_i, x_j) = -\min_{h \in \mathcal{H}} \mathbb{E}_{D \sim P}[\ell(h(\widehat{m}_D(x_i)), \widehat{m}_D(x_j))], \quad (3)$$

which measures how predictable the foundation model's predicted outputs for one input are from the other. Higher values of extrapolative predictability indicate higher levels of predictability. If a foundation model behaves as if it extrapolates based on the postulated world model, the extrapolative predictability should be larger for inputs with more similar states.

Oracle foundation model. As a calibration, we calculate the extrapolative predictability for an "oracle" foundation model that is given access to the true state space Φ and admissible functions $\mathcal{G}$. When applied to consistent dataset D, the oracle foundation model returns

$$m^*_D = \arg\min_{g \in \mathcal{G}} \tfrac{1}{|D|} \textstyle\sum_{(x_i, y_i) \in D} \ell(g(\phi(x_i)), y_i). \quad (4)$$

(The loss function used here need not be the same as the loss function used to calculate extrapolative predictability.) The oracle extrapolative predictability is

$$I^*(x_i, x_j) = -\min_{h \in \mathcal{H}} \mathbb{E}_{D \sim P}[\ell(h(m^*_D(x_i)), m^*_D(x_j))]. \quad (5)$$

Inductive bias towards the world model. The inductive bias probe compares the foundation model's extrapolative predictability to that of the oracle. Specifically, the foundation model's *inductive bias towards the world model* is defined as, for any $0 \le \underline{s} \le \overline{s}$,

$$\mathrm{IB}(\underline{s}, \overline{s}) = \mathbb{E}_{X_i, X_j}[\widehat{I}(X_i, X_j) \mid \underline{s} \le I^*(X_i, X_j) \le \overline{s}]. \quad (6)$$

We calculate this quantity over a grid of values $0 = s_0 < s_1 < \cdots < s_m$, visualizing how $\mathrm{IB}(\underline{s}, \overline{s})$ varies over the grid. The foundation model's inductive bias towards the world model can be interpreted like a calibration curve: if

the foundation model behaves like the oracle when applied to many small datasets, then $\text{IB}(\underline{s}, \overline{s})$ should lie on the 45-degree line in this visualization (as illustrated in Figure 2).

R-IB and D-IB are special cases of the foundation model's inductive bias towards the world model (Equation 6). Consider the case in which the output is binary, Φ is finite, and $\mathcal{G}$ is the collection of all mappings. Provided P_D places positive probability on all possible consistent datasets and $\mathcal{H}$ is limited to the identity function, there are only two possible values for the oracle's extrapolative predictability, which occur when $\phi(x_i) = \phi(x_j)$ and when $\phi(x_i) \neq \phi(x_j)$. Consequently, the foundation model's inductive bias towards the world model reduces to R-IB in in the former case (Equation 1) and D-IB (up to a sign change) in the latter case (Equation 2).

3. Orbital Mechanics

We illustrate these ideas by testing whether a transformer trained to predict the locations of planets in motion has recovered Newtonian mechanics. Despite the model's ability to accurately predict the future trajectories of planets, the inductive bias probe reveals that it has a low inductive bias toward Newtonian mechanics. This is corroborated by the fact that when the model is fine-tuned to predict a planet's force vector — a cornerstone of Newtonian mechanics — its predictions imply a nonsensical law of gravitation. We demonstrate that the model has recovered piecemeal heuristics rather than a compact world model; it recovers a different law of gravitation depending on the slice of data it is applied to.

Background. Astronomers and physicists spent centuries trying to predict the orbits of planets in the sun. A groundbreaking model was offered by the astronomer Johannes Kepler in the 17th century. His model was based on geometric patterns: for example, that the orbit of each planet followed an ellipse with the sun at one of its foci. While the model could predict orbits with a near-perfect level of precision, it couldn't explain why the planets obeyed these geometric orbits or be applied to new problems beyond predicting trajectories.

Later, Isaac Newton expanded on this model using new laws of motion, now known as Newtonian mechanics. These laws involved computing key properties of the set of planets in motion, such as their relative velocities and masses. Using these properties, he could derive Kepler's earlier laws for orbital trajectories, but also go beyond, understanding and formalizing other concepts like force and gravity.

From Kepler to Newton, scientists were able to move beyond good predictive models of sequences to a deeper understanding of them. In this section, we test whether a transformer that can predict sequences of orbital trajectories is merely a good sequence model, or whether it has also made the transition to something more foundational.

Data and pre-training. We first simulate a dataset of sequences, where each sequence describes planets in motion around a sun. To do this, we randomly sample initial conditions (e.g. the masses and positions of the planets and their initial relative velocities) to target the shape of orbits observed in known exoplanets (Kipping, 2013). We simulate each planet's trajectory around the sun using Newton's laws of motion; because planet masses are much smaller than the sun's, interactions between planets are minimal, so we omit them. To convert orbits into sequences, we record (x, y) coordinates of each planet and the sun across 6-month intervals, and interleave all the locations into a single sequence with 1,000 observations. For example, in a solar system with K planets, the first K observations are the (x, y) coordinates for each planet at the first point in time, while the next K are the coordinates for each planet 6 months later, etc. We consider sequences taken from a training set containing 1B tokens, along with 300K hold-out tokens.

We train a 109M parameter transformer (Vaswani et al., 2017) to predict the next token of each sequence in the training set. We experimented with performing next-token prediction using a) the continuous coordinates (and MSE loss) and b) discretized coordinates (with cross-entropy loss), finding the latter worked better. We discretize each position vector of each body in the solar system by creating 7K bins per coordinate (x, y), where the coordinates spans from -50 to 50 AU. See Appendix A for complete training details.

We evaluate the model's predictions on hold-out data. The model makes good predictions: its R^2 is above 0.9999, and it significantly outperforms baseline models that always predict the most recent position or the per-orbit mean (Table 8). It can also generate long orbits with a high degree of accuracy.

Has the model recovered Newtonian mechanics? The transformer's predictions reflect a very good sequence model. But has it recovered Newtonian mechanics? To test this, we note that Newtonian mechanics dictate that each observation in a sequence of orbits is governed by a state vector consisting of the masses, relative velocities, and relative positions of each planet. Given the current state of a trajectory, the next position of an orbit is deterministic. This is our world model; if a foundation model's inductive bias depends on Newtonian mechanics, it must be extrapolating based on this state vector.

We use the inductive bias probe described in Section 2 to assess the model's inductive biases. We create 100 synthetic datasets where the outputs are linear functions of the state of the sequence. We then fine-tune the transformer by training it to predict these functions. We measure the model's extrap-

Ground-truth law		$F \propto \frac{m_1 m_2}{r^2}$
Estimated laws	Sample 1	$F \propto \cos(\cos(-1/m_1))$
	Sample 2	$F \propto \sin(e^{m_2+3.8})$
	Sample 3	$F \propto \cos(\cos(1/m_2))$
	Sample 4	$F \propto \exp\left[\frac{1}{r}\sin(\sin(e^{e^{m_2+1.1}})r)\right]$

Table 1: Force equations recovered via symbolic regression of the transformer pretrained on orbital mechanics and fine-tuned to different galaxy samples. The model recovers different equations for each sample, never recovering the true law.

olative predictability across inputs (Equation 3) by considering $\mathcal{H}$ to consist of the identity and the loss function ℓ to be MSE. We evaluate Equation 6 by comparing the model to an oracle that extrapolates based on state directly (we consider both linear models and 2-layer neural networks for the oracle, finding similar results). The inductive bias toward simple functions of Newtonian state is poor; see Figure 2 for a visualization. In other words, the model's inductive bias is not toward Newtonian state; when it has to extrapolate, it makes similar predictions for orbits with very different states and different predictions for orbits with very similar states. For implementation details, see Appendix B.1.

To understand the degree to which the model fails to apply Newtonian mechanics, we test its ability to predict specific quantities derived from Newtonian mechanics. Specifically, we consider each planet's force vector, a simple transformation of state given by Newton's law of gravitation: $\mathbf{F} = G\frac{m_1 m_2}{||\mathbf{r}||^2}\mathbf{e}_r$, which relates the force $\mathbf{F}$ between a planet and the sun to their masses m_1, m_2 and their squared distance $||\mathbf{r}||^2$ (in the direction $\mathbf{e}_r$ of its relative position). The force vector can be computed for each observation in a sequence; force is a simple transformation of state, so the predictions of a model that has recovered Newtonian mechanics should obey this law.

We test this by creating a sequence-to-sequence dataset where each input is a trajectory and each output is the force vector $\mathbf{F}$ on the planet implied by the state of the orbit. We first fine-tune the pretrained transformer to predict the force vector on orbits from our solar system, providing 1% of the true forces as training data. Figure 1 shows these force predictions are poor. To assess how close the model is to recovering Newton's law of gravitation, we further fine-tune it to predict the force magnitude on a large dataset of 10,000 solar systems. We then perform a symbolic regression (using the *PySR* software (Cranmer, 2023)) of the predicted force magnitudes on the true values of m_1, m_2, and r. A symbolic regression is a method to search for a symbolic expression that optimizes a regression-like objective (Cranmer et al., 2020). When the magnitudes are predicted using a 2-layer neural network fine-tuned on the same data using the true state matrix, it returns the true law of gravitation. However, when the symbolic regression is applied to the transformer's predictions, the physical law is nonsensical (Figure 1). See Appendix C for implementation details.

How can a model perform so well at predicting orbit locations without having inductive biases towards the laws of physics that govern them? We study this question by applying the fine-tuned model's force predictions to four different sets of randomly sampled galaxies (each consisting of many solar systems). We then perform a symbolic regression on the force magnitude for each sample. The symbolic regression finds a different implied law of gravitation for each sample (Table 1). In contrast, the neural network trained on true state recovers the same law for each galaxy. These results show that rather than building a single universal law, the transformer extrapolates as if it constructs different laws for each sample.

4. Other Applications

We now apply the inductive bias probe to evaluate the extent to which foundation models obey known world models in other domains. Evaluating world models requires studying domains where there's a state structure and ground-truth state is known. We study two such types of datasets: lattice problems and the board game Othello. In Appendix D, we also apply these metrics to study the inductive biases of large language models in a synthetic preference task.

Lattice. One common type of structure to assess models against is spatial structure, or lattices (Vafa et al., 2024; Liu et al., 2022). We study a lattice setting that simulates an agent moving along a line segment with a finite number of positions. There is a true state space consisting of S states: $\Phi = \{1, 2, \ldots, S\}$. The language x consists of sequences with three tokens: $\Sigma = \{L, \perp, R\}$. The initial state of the sequence is 1. For a token $\sigma = R$, the state increases by 1, while the state decreases by 1 for $\sigma = L$ and stays the same for $\sigma = \perp$. When the state is 1, the state is at the boundary, so $\sigma = L$ is not a valid token; similarly, when the state is S, $\sigma = R$ is not a valid token. All tokens are valid for all other states. We randomly generate sequences of length 100 over the language by sampling a move uniformly at random over the set of valid moves for each timestep. We consider different versions of the lattice problem, varying the number of states from 2 - 5. We consider sequences taken from a training set containing 10M tokens, along with 100k hold-out tokens.

Othello. We also study the board game Othello, a common testbed for evaluating the world models of sequence models (Li et al., 2023; Nanda et al., 2023b; Hazineh et al., 2023;

		Lattice (5 States)		Othello	
	Pre-training	R-IB (↑)	D-IB (↑)	R-IB (↑)	D-IB (↑)
RNN (Elman, 1990)	Untrained	0.346 (0.026)	0.749 (0.027)	0.240 (0.019)	0.987 (0.002)
	NTP trained	0.574 (0.026)	0.803 (0.032)	0.558 (0.021)	0.845 (0.019)
LSTM (Hochreiter, 1997)	Untrained	0.456 (0.028)	0.718 (0.031)	0.506 (0.028)	0.672 (0.032)
	NTP trained	0.782 (0.021)	0.921 (0.030)	0.649 (0.030)	0.448 (0.035)
Transformer (Vaswani et al., 2017)	Untrained	0.268 (0.027)	0.742 (0.028)	0.714 (0.022)	0.840 (0.021)
	NTP trained	0.483 (0.031)	0.677 (0.034)	0.668 (0.023)	0.593 (0.034)
Mamba (Gu & Dao, 2023)	Untrained	0.260 (0.026)	0.771 (0.027)	0.342 (0.019)	0.933 (0.013)
	NTP trained	0.571 (0.023)	0.866 (0.029)	0.597 (0.024)	0.734 (0.028)
Mamba-2 (Dao & Gu, 2024)	Untrained	0.244 (0.026)	0.785 (0.026)	0.490 (0.020)	0.936 (0.008)
	NTP trained	0.617 (0.021)	0.864 (0.029)	0.590 (0.022)	0.732 (0.027)

Table 2: The *inductive bias towards respecting state* (R-IB) and *inductive bias towards distinguishing state* (D-IB) metrics (1 is perfect performance, 0 is equivalent to noninformative model). "NTP-trained" represents a model pre-trained on next-token prediction, while "untrained" refers to a model trained on the same synthetic tasks, initialized from scratch.

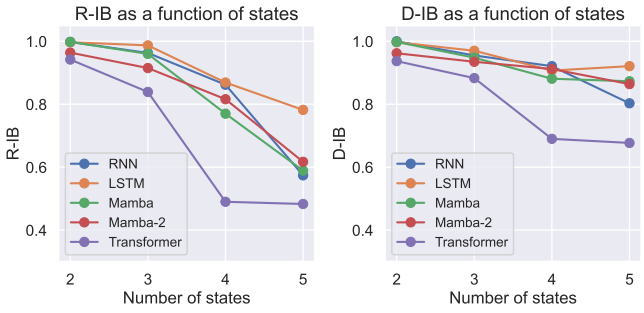

Figure 3: Inductive bias probe results (R-IB and D-IB) for the lattice problem as a function of the underlying number of states. A different model is pre-trained on data consistent with each number of states and its inductive bias for that state structure is recorded using the metrics in Section 2.

Vafa et al., 2024). The game consists of two players taking turns placing tiles on an 8x8 board. Each game of Othello is tokenized into a sequence of at most 60 moves, where each token indicates which of the 60 squares the most recent tile was placed on (the middle four tiles are always occupied). The true state space Φ corresponds to all 8x8 boards and the mapping ϕ converts game sequences into states. We consider game sequences taken from a training set containing 7.7M tokens, along with 300K hold-out tokens.

Models. We study the properties for five classes of pre-trained sequence models: RNNs (Elman, 1990), LSTMs (Hochreiter, 1997), transformers (Vaswani et al., 2017), Mamba (Gu & Dao, 2023), and Mamba-2 (Dao & Gu, 2024). We train each model using next-token prediction for each domain. By way of comparison, we also compare these pretrained models to untrained models that fine-tune from a random initialization. See Appendix A for more information.

All pre-trained models perform well at next-token prediction, generating outputs that appear to obey state. Following Toshniwal et al. (2022), we measure the fraction of a model's top predictions that are legal in the underlying state. Table 7 in Appendix G shows the results. All models do very well across all datasets, e.g. every model's top prediction is legal $\approx 90\%$ of the time for Othello and legal 100% of the time for a lattice problem with five states.

Inductive bias probe results. We measure each model's inductive bias using the procedure from Section 2 to assess the inductive bias of these models. The procedure involves fine-tuning each model to small datasets of randomly generated outputs and assessing whether the model's inductive bias — as measured by its extrapolations — obeys state structure. We use the discrete version of the procedure for both models.

The results for the lattice problem are depicted in Figure 3. While models have high inductive biases when the number of states is small, as the number of states increases, the inductive biases drop off. Notably, the transformer model consistently does worse than the other models, all of which have architectures based on recurrent or state-space models. The results for Othello are depicted in Table 2. Here, all models perform worse than on the lattice problems, indicating poor inductive bias. Despite generating legal moves nearly 100% of the time when pretrained to play Othello, these models don't use the board as an inductive bias on new tasks.

To understand the implications of these results, we study how different models transfer to new functions of state (the board). Specifically, we take the Othello dataset and construct new sequence-to-sequence datasets. The input sequence for each dataset is the original game transcript, and we consider three different output transformations that are functions of state. In "Majority Tiles", each element of

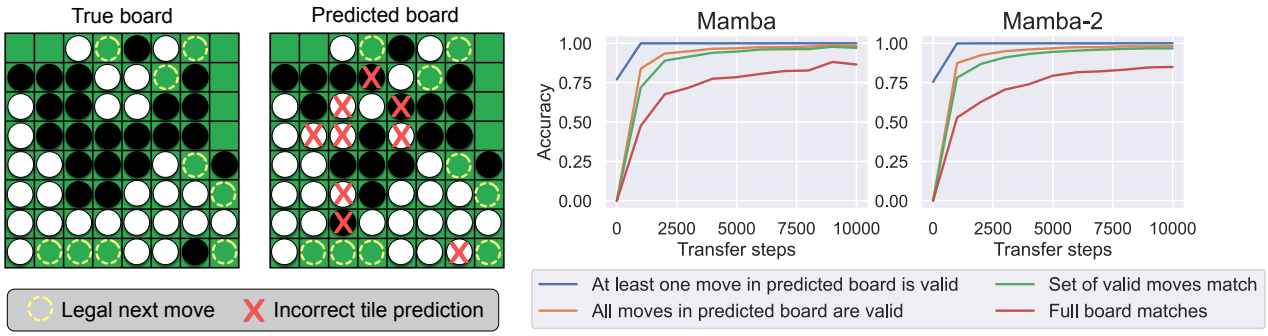

Figure 4: On the left, a true Othello board implied by a sequence, and on the right, the predicted board from a model fine-tuned to predict boards. Although the prediction has errors, the set of predicted next tokens exactly matches the true board. On the right, metrics about board reconstruction during fine-tuning. Consistently, even as Mamba models struggle to recover full boards, they recovers them well enough such that the sets of valid next moves match the true boards.

the output is 1 or 0 indicating where there are more black or white tiles in the board implied by the sequence so far. In "Board Balance", each element of the output sequence indicates whether black has more pieces in the top half of the board or in the bottom half of the board. Finally, in "Edge Balance", the output measures whether black has more pieces along the edge squares of the board. Each of these functions is a deterministic function of state (the board), so foundation models that have inductive bias toward state should be better at transfer. We transfer models for 5,000 iterations; see Appendix F for other amounts. The results are depicted in Table 6. The last row shows the (unsigned) correlation for each metric and the ratio, $\frac{\text{R-IB}}{\text{D-IB}}$ that summarizes the inductive bias measures in Table 2. There is strong correlation across all metrics; models that do better on inductive bias metrics transfer better to these functions of state. See Appendix F for further analysis.

What are the inductive biases? These results show that models can perform well at predicting token sequences without appearing to learn the underlying world model. This raises the question: If a foundation model's inductive bias isn't toward a given world model, what is it toward?

Here, we consider one hypothesis motivated by the next-token pretraining objective: that when foundation models are applied to new tasks, they group together sequences with distinct states for which the set of legal next tokens are nevertheless equivalent. For example, in the board game Othello, two distinct boards can have the same set of allowable next moves. Therefore, a model's inductive bias might be toward boards with the same sets of allowable next moves rather than the true board itself.

To first demonstrate this concept with Othello, we fine-tune a foundation model pretrained on next-token prediction to predict the true board of each sequence. We record two metrics when we fine-tune: 1) whether the predicted board exactly matches the true board, and 2) whether the set of valid moves in the predicted board matches the set of valid moves in the true board. The results are depicted in Figure 4: surprisingly, even when the predicted board is incorrect, the set of legal moves frequently matches the set of legal moves from the true board. Rather than recovering the full board, the foundation model is often recovering "enough of" the board to calculate legal next moves.

To quantify this hypothesis generally, we modify the inductive bias probe to test whether a model's inductive bias is toward *next-token partitions* of state. Recall that D-IB measures how similar extrapolations for two points with different states are one from another. If a model is extrapolating based on which next-tokens are legal, sequences in different states that happen to have the same legal next tokens will have more similar predictions than sequences in different states that have different legal next tokens.

Specifically, let q denote the *next-token coarsening* of the state space such that $q(x) = q(x')$ if and only if $\text{NextTokens}(\phi(x)) = \text{NextTokens}(\phi(x'))$, where $\text{NextTokens}(s)$ is the set of valid next tokens for state s. We decompose D-IB into two quantities. First, define $\text{Same}(X_i, X_j)$ as the event that $\phi(X_i) \neq \phi(X_j)$ but $q(X_i) = q(X_j)$. We then define,

$$\text{D-IB}_{q=} = 1 - \mathbb{E}\left[1(\widehat{m}_D(X_i), \widehat{m}_D(X_j)) \mid \text{Same}(X_i, X_j)\right],$$

which measures how predictable the extrapolations for inputs associated with different states that have the *same* legal next tokens are. Similarly, define $\text{Diff}(X_i, X_j)$ as the event that $\phi(X_i) \neq \phi(X_j)$ and $q(X_i) \neq q(X_j)$. Analogously,

$$\text{D-IB}_{q\neq} = 1 - \mathbb{E}\left[1(\widehat{m}_D(X_i), \widehat{m}_D(X_j)) \mid \text{Diff}(X_i, X_j)\right],$$

which measures how predictable the extrapolations for inputs associated with different states that have *different* legal next tokens are. If distinct-state inputs with the same legal

next tokens are more predictable than distinct-state inputs with different legal next tokens (i.e., D-IB$_{q=}$ < D-IB$_{q\neq}$), then it suggests the model extrapolates based on the next-token partition rather than the true board state.

We compute these refined metrics for lattice and Othello. Each has a natural definition of legal next moves (corresponding to boundaries and game rules). The results are depicted in Table 9. For all models, the gap between D-IB$_{q=}$ and D-IB$_{q\neq}$ is statistically significant, suggesting that models are grouping together distinct states with the same sets of legal next tokens.

5. Related Work

One strand of world model research studies whether the outputs of a fixed model accord with a known world model by studying the fixed model's outputs (Vafa et al., 2024). For example, one way that Toshniwal et al. (2022) and Li et al. (2023) study world models is by assessing whether a model trained on sequential game data always predicts legal moves in the underlying game. The question we study is a different yet related question: rather than studying the world model properties of a fixed model, we study what it means to test if a *learning algorithm* — a foundation model — has a world model embodied in it.

Another strand of the literature assesses whether a model's parametric *representations* encode world models without directly studying learning properties. For example, a common method uses probes or sparse autoencoders (SAEs) (Trenton Bricken et al.) to assess whether an intermediate representation used by a neural network is predictive of state (Hewitt & Liang, 2019; Li et al., 2021; Abdou et al., 2021; Jin & Rinard, 2023; Li et al., 2023; Spies et al., 2025; Karvonen, 2024). However, there are open questions about the reliability of probes (Belinkov, 2022), such as appropriate function complexity (Alain & Bengio, 2018; Cao et al., 2021; Li et al., 2023). Our method sidesteps these issues by asking how a model *learns*, rather than what's encoded in its fixed representations.

The methods in this paper are also related to the study of mechanistic interpretability of ML models (Nanda et al., 2023a; Cunningham et al., 2023; Bereska & Gavves, 2024). Closely related to us, jylin04 & Rager (2024) and Nikankin et al. (2025) find that a GPT model trained on Othello and math tasks, respectively, performs internal computations corresponding to "bags of heuristics" rather than a coherent world model. While our procedures differ in aim, these findings support our analysis of the Othello model relying on heuristics, rather than state, as its inductive bias (McCoy et al., 2019).

Recent work developing foundation models in scientific domains such as like protein folding, gene regulation, and molecular chemistry (Chowdhury et al., 2022; Benegas et al., 2023; Boiko et al., 2023; Jablonka et al., 2024) use predictive models as steppingstones toward uncovering deeper principles. Our orbital mechanics example relates specifically to the large body of work studying AI and physics (Hao et al., 2022; Wu & Tegmark, 2019). It is most closely related to works studying whether AI models can uncover physical laws (Chen et al., 2022; Belyshev et al., 2024; Kansky et al., 2017; Gurnee & Tegmark, 2024). We adapt tools from this literature — such as using symbolic regressions for interpretability — to study the inductive biases of algorithms (Liu & Tegmark, 2021; Wu & Tegmark, 2019).

6. Conclusion

The promise of foundation models is that sequence prediction can uncover deeper understanding of underlying mechanisms. We develop a framework for evaluating whether a foundation model has learned a postulated world model by measuring its inductive biases when transferring to new tasks. Our empirical results reveal that while many sequence models excel at next-token prediction tasks, they often have limited inductive bias toward genuine world models. Rather than learning coherent world models, we find that these models may be relying on coarsened state representations or non-parsimonious representations.

As described in Section 2, our metrics require specifying a world model to test a foundation model against. That a world model must be specified aligns with other examples in this literature (Li et al., 2023; Vafa et al., 2024), but it is a limitation for analysts searching for the exact representation the model is using. While we propose strategies for testing candidates (e.g. next-token partitions), future work should prioritize methods for automatically constructing the world model implicit in the foundation model's behavior.

Acknowledgments

Keyon Vafa is supported by the Harvard Data Science Initiative.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Abdou, M., Kulmizev, A., Hershcovich, D., Frank, S., Pavlick, E., and Søgaard, A. Can language models encode perceptual structure without grounding? A case study in

color. *arXiv preprint arXiv:2109.06129*, 2021.

Alain, G. and Bengio, Y. Understanding intermediate layers using linear classifier probes. 2018. *arXiv preprint arXiv:1610.01644*, 2018.

Belinkov, Y. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219, 2022.

Belyshev, A., Kovrigin, A., and Ustyuzhanin, A. Beyond dynamics: learning to discover conservation principles. *Machine Learning: Science and Technology*, 5(2):025055, 2024.

Benegas, G., Batra, S. S., and Song, Y. S. DNA language models are powerful predictors of genome-wide variant effects. *Proceedings of the National Academy of Sciences*, 120(44):e2311219120, 2023.

Bereska, L. and Gavves, E. Mechanistic interpretability for ai safety–a review. *arXiv preprint arXiv:2404.14082*, 2024.

Boiko, D. A., MacKnight, R., Kline, B., and Gomes, G. Autonomous chemical research with large language models. *Nature*, 624(7992):570–578, 2023.

Cao, S., Sanh, V., and Rush, A. M. Low-complexity probing via finding subnetworks. *arXiv preprint arXiv:2104.03514*, 2021.

Casper, S., Li, Y., Li, J., Bu, T., Zhang, K., Hariharan, K., and Hadfield-Menell, D. Red Teaming Deep Neural Networks with Feature Synthesis Tools, September 2023.

Chen, B., Huang, K., Raghupathi, S., Chandratreya, I., Du, Q., and Lipson, H. Automated discovery of fundamental variables hidden in experimental data. *Nature Computational Science*, 2(7):433–442, 2022.

Chowdhury, R., Bouatta, N., Biswas, S., Floristean, C., Kharkar, A., Roy, K., Rochereau, C., Ahdritz, G., Zhang, J., Church, G. M., Sorger, P. K., and AlQuraishi, M. Single-sequence protein structure prediction using a language model and deep learning. *Nature Biotechnology*, 40(11):1617–1623, 2022.

Cranmer, M. Interpretable machine learning for science with pysr and symbolicregression.jl, 2023.

Cranmer, M., Sanchez-Gonzalez, A., Battaglia, P., Xu, R., Cranmer, K., Spergel, D., and Ho, S. Discovering Symbolic Models from Deep Learning with Inductive Biases, November 2020.

Cunningham, H., Ewart, A., Riggs, L., Huben, R., and Sharkey, L. Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*, 2023.

Dao, T. and Gu, A. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*, 2024.

Elman, J. L. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

Gingerich, O. *The book nobody read: Chasing the revolutions of Nicolaus Copernicus*. Bloomsbury Publishing USA, 2004.

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

Gurnee, W. and Tegmark, M. Language Models Represent Space and Time, March 2024.

Hao, Z., Liu, S., Zhang, Y., Ying, C., Feng, Y., Su, H., and Zhu, J. Physics-informed machine learning: A survey on problems, methods and applications. *arXiv preprint arXiv:2211.08064*, 2022.

Hazineh, D. S., Zhang, Z., and Chiu, J. Linear latent world models in simple transformers: A case study on Othello-GPT. *arXiv preprint arXiv:2310.07582*, 2023.

Hewitt, J. and Liang, P. Designing and interpreting probes with control tasks. *arXiv preprint arXiv:1909.03368*, 2019.

Hochreiter, S. Long short-term memory. *Neural Computation MIT-Press*, 1997.

Jablonka, K. M., Schwaller, P., Ortega-Guerrero, A., and Smit, B. Leveraging large language models for predictive chemistry. *Nature Machine Intelligence*, pp. 1–9, 2024.

Jin, C. and Rinard, M. Evidence of meaning in language models trained on programs. *arXiv preprint arXiv:2305.11169*, 2023.

jylin04, JackS, A. K. and Rager, C. Othellogpt learned a bag of heuristics, jul 2024. URL https://www.lesswrong.com/posts/gcpNuEZnxAPayaKBY/othellogpt-learned-a-bag-of-heuristics-1. Posted on LessWrong.

Kansky, K., Silver, T., Mély, D. A., Eldawy, M., Lázaro-Gredilla, M., Lou, X., Dorfman, N., Sidor, S., Phoenix, S., and George, D. Schema Networks: Zero-shot Transfer with a Generative Causal Model of Intuitive Physics. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 1809–1818. PMLR, July 2017.

Karvonen, A. Emergent World Models and Latent Variable Estimation in Chess-Playing Language Models, July 2024.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kipping, D. M. Parametrizing the exoplanet eccentricity distribution with the beta distribution. *Monthly Notices of the Royal Astronomical Society: Letters*, 434(1):L51–L55, 2013.

Koestler, A. *The Sleepwalkers: A History of Man's Changing Vision of the Universe*. Hutchinson, London, 1959. First edition.

Li, B. Z., Nye, M., and Andreas, J. Implicit representations of meaning in neural language models. *arXiv preprint arXiv:2106.00737*, 2021.

Li, K., Hopkins, A. K., Bau, D., Viégas, F., Pfister, H., and Wattenberg, M. Emergent world representations: Exploring a sequence model trained on a synthetic task. In *International Conference on Learning Representations*, 2023.

Li, K., Hopkins, A. K., Bau, D., Viégas, F., Pfister, H., and Wattenberg, M. Emergent World Representations: Exploring a Sequence Model Trained on a Synthetic Task, June 2024.

Liu, B., Ash, J. T., Goel, S., Krishnamurthy, A., and Zhang, C. Transformers learn shortcuts to automata. *arXiv preprint arXiv:2210.10749*, 2022.

Liu, Z. and Tegmark, M. Ai poincare: Machine learning conservation laws from trajectories. *arXiv:2011.04698*, 2021.

McCoy, R. T., Pavlick, E., and Linzen, T. Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference. In Korhonen, A., Traum, D., and Màrquez, L. (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 3428–3448, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1334.

Nanda, N., Chan, L., Lieberum, T., Smith, J., and Steinhardt, J. Progress measures for grokking via mechanistic interpretability. *arXiv preprint arXiv:2301.05217*, 2023a.

Nanda, N., Lee, A., and Wattenberg, M. Emergent linear representations in world models of self-supervised sequence models. *arXiv preprint arXiv:2309.00941*, 2023b.

Nikankin, Y., Reusch, A., Mueller, A., and Belinkov, Y. Arithmetic Without Algorithms: Language Models Solve Math With a Bag of Heuristics, May 2025.

Olah, C. Mechanistic Interpretability, Variables, and the Importance of Interpretable Bases. https://www.transformer-circuits.pub/2022/mech-interp-essay, June 2022.

Spies, A. F., Edwards, W., Ivanitskiy, M. I., Skapars, A., Räuker, T., Inoue, K., Russo, A., and Shanahan, M. Transformers Use Causal World Models in Maze-Solving Tasks, March 2025.

Toshniwal, S., Wiseman, S., Livescu, K., and Gimpel, K. Chess as a testbed for language model state tracking. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 11385–11393, 2022.

Trenton Bricken, Adly Templeton, B. C. et al. Towards Monosemanticity: Decomposing Language Models With Dictionary Learning. https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Vafa, K., Chen, J. Y., Kleinberg, J., Mullainathan, S., and Rambachan, A. Evaluating the world model implicit in a generative model. *arXiv preprint arXiv:2406.03689*, 2024.

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

Wolpert, D. H. The lack of a priori distinctions between learning algorithms. *Neural Computation*, 8(7):1341–1390, 10 1996. ISSN 0899-7667. doi: 10.1162/neco.1996.8.7.1341. URL https://doi.org/10.1162/neco.1996.8.7.1341.

Wu, T. and Tegmark, M. Toward an artificial intelligence physicist for unsupervised learning. *Physical Review E*, 100(3):033311, 2019.

A. Model and Training Details

We use the following specifications for each model:

- RNN (Elman, 1990): For Othello, We use 6 uni-directional RNN layers with 768 embedding dimensions. For the lattice experiments, the architecture is the same except we use only 2 layers because it optimizes to better in-sample and out-of-sample loss.
- LSTM (Hochreiter, 1997): We use the same specification as for the RNN, except we use LSTM layers.
- Transformer (Vaswani et al., 2017): We use a transformer decoder architecture, with 12 layers, 12 attention heads, and 768 embedding dimensions.
- Mamba (Gu & Dao, 2023): We first encode inputs with a 768-dimension embedding layer. We then pass inputs through 24 Mamba layers (analogous to 12 layers in a transformer due to how Mamba layers are defined). We use 768 embedding dimensions, 16 for the SSM state expansion factor, 2 for the block expansion factor, and 4 for the convolutional width.
- Mamba-2 (Dao & Gu, 2024): We use the same architecture as for Mamba except the mixer in each block is a Mamba-2 module. We use the same specifications as well: 768 embedding dimensions, an SSM state expansion factor of 16, a block expansion factor of 2, and a convolutional width of 4.

We use Adam (Kingma & Ba, 2014) to optimize each model. We use a learning rate of 6e-4 and decay the learning rate with with 2000 warmup iterations. We use weight decay of 0.1 and gradient clipping at 1 for each model. When we pre-train models on next-token prediction, we include a head to predict next tokens (tying its parameter weights to the initial embedding layer parameters).

For physics dataset generation, we use the following sampling strategy. For each solar system, we sample the number of planets from Unif$([1, 2, \ldots, 10])$, the eccentricity from a Beta$(\alpha = 0.867, \beta = 3.03)$ following (Kipping, 2013), the semi-major axis from Unif(0.3, 42), in astronomical units (AU), the mass of each planet from LogUniform$(10^{-7}, 10^{-3})$, the mass of the star from Unif$(0.5, 5)$. These distributions ensure that our solar system is within the training distribution of the model. In order to generate sequences, we randomly generate initial conditions and solve Kepler's equation to obtain each trajectory. We simulate 1000 timesteps with 6-month intervals for each sequence.

B. Metric Implementation Details

B.1. Physics

To compute the empirical approximations of Equation 6, we follow the following procedure. First, we create 100 datasets of 100 examples, $D_1, \ldots, D_{100}$. For each dataset D_i, we sample 100 sequences uniformly at random among the set of data points and consider their corresponding sequences of state-vectors. First, we randomly sample 50 matrices of dimension (6×1) from standard Gaussian. We consider the linear projection of each state-vector using each of the 50 matrices, and choose the one that maximizes the Spearman correlation between pairwise Euclidian distances in the 6D state space and the projected 1D space. We randomly sample a projected point from each sequence, leading to D_i of size 100. We then fine-tune a model separately for each dataset, resulting in 100 fine-tuned models $\hat{m}(\cdot; D_1), \ldots, \hat{m}(\cdot; D_{100})$. We then calculate the associated prediction functions across all inputs x_i from the same hold-out dataset, resulting in new datasets of the form $\{(x_i, \hat{m}(x_i; D_1)\}, \ldots, \{(x_i, \hat{m}(x_i; D_{100})\}$.

To compute the metrics, we first randomly sample 2,000 examples from all inputs, $x_{k_1}, \ldots x_{k_{100}}$, compute the pairwise Euclidean distance among the Oracle (a linear map or a 2 layer MLP with 5 nodes in each hidden layer) predictions on the inputs, and divide the range of predictions into 20 equally-spaced bins. For all the points that lie in each bin, we compute the mean pairwise Euclidiean distance among the model predictions. The resulting figure is shown in Figure 2.

B.2. Lattice and Othello

To compute the empirical approximations of Equation 1 and Equation 2, we follow the following procedure. First, we create 100 datasets of 100 examples, $D_1, \ldots, D_{100}$. For each dataset, we sample sequences uniformly at random among the set of

data points and sample outputs from a Bernoulli(0.5) distribution. In our construction we make sure that any two sequences with the same state are mapped to the same output variable. We then fine-tune a model separately for each dataset, resulting in 100 fine-tuned models $\hat{m}(\cdot; D_1), \ldots, \hat{m}(\cdot; D_{100})$. We then calculate the associated prediction functions across all inputs x_i from the same hold-out dataset, resulting in new datasets of the form $\{(x_i, \hat{m}(x_i; D_1)\}, \ldots, \{(x_i, \hat{m}(x_i; D_{100})\}$.

To compute the metrics, we first randomly sample 2,000 examples from all inputs, $x_{k_1}, \ldots x_{k_{100}}$, then measure the average predictive loss for all pairs (x_{k_i}, x_{k_j}) with the same state, $\phi(x_{k_i}) = \phi(x_{k_j})$ (R-IB):

$$\text{R-IB} \approx \mathbb{E}_{D \sim D^{test}} \left[\mathbb{E}_{i,j:\phi(x_{k_i})=\phi(x_{k_j})} \left[m(x_i; D) = m(x_j; D) \right] \right] \tag{7}$$

and the average predictive loss for all pairs (x_{k_i}, x_{k_j}) with different states, $\phi(x_{k_i}) \neq \phi(x_{k_j})$ (D-IB):

$$\text{D-IB} \approx 1 - \mathbb{E}_{D \sim D^{test}} \left[\mathbb{E}_{i,j:\phi(x_{k_i})\neq\phi(x_{k_j})} \left[m(x_i; D) = m(x_j; D) \right] \right] \tag{8}$$

We rescale them so that the value of 0 corresponds to perfect accuracy and 1 corresponds to random guessing (large values for both indicate the model exhibits stronger inductive bias towards the state).

For the lattice example, we use a state space consisting of k states: $\Phi = \{1, 2, \ldots, k\}$. The inputs x_i for extrapolation are taken from 1,000 random sequences of valid moves, each of length 100, for a total of 100,000 sub-sequences of moves. Our procedure for Othello follows the same steps as for the lattice example, except the state is a 64-dimensional board instead of a single categorical variable.

Note that for Othello, if we randomly sample sequences from game transcripts, it is exceedingly likely that we end up with a dataset in which two sequences lead to the same state if and only if they are permutations of one another. This implies that a non-sophisticated model that detects unique permutations of sequences would appear to have high inductive bias towards the state. To prevent this, we first construct all valid Othello game openings of depth 10, randomly choose a board that appears many times in this dataset, then use all possible valid permutations of any sequence of moves that leads to that board as our input dataset. Note that since all sequences will be permutations of one another, the non-sophisticated model would no longer be able to distinguish different states. We end up with an input dataset of 210 Othello openings, each of length 10, for a total of 2,100 subsequence of moves.

C. Force Prediction Implementation Details

Force Vector Prediction. To predict force vectors (Figure 1), we fine-tune the transformer to predict force vectors in two-body gravitational systems. We keep force vectors as continuous, and normalize each sequence so the maximum force vector has the same length. We specifically fine-tune the model on the 8 sequences consisting of the trajectories in our solar system, randomly using 1% of the observations in each 1000-length sequence as labeled force vector data for the model. We fine-tune the model to minimize MSE. The model is then extrapolated to make predictions across the remainder of the points in each sequence. For comparison, we perform the same procedure for a two-layer neural network trained to predict force from ground-truth Newtonian state using the same data, and find that its force predictions are near-perfect.

Force Magnitude Prediction. The symbolic regression experiments depend on the force magnitude rather than the force vector. We find that models perform better when the force magnitude is discretized. Specifically, we discretize force magnitudes into 5k bins, where the magnitudes span 2e-7 to 5e-3. We fine-tune the transformer on 10K two-body training dataset (or four 10K two-body training datasets for Table 1) and evaluate the predictions on 1000 randomly sampled points across sequences. We use the same procedure for the oracle, training a two-layer neural network on the ground truth state on a dataset of 10K two-body sequences. We then run symbolic regression on the predictions, and report results.

D. LLM Experiments

Our experiments so far have been for fine-tuning foundation models on non-text datasets. Here, we apply our framework in a setting involving large language models. Fine-tuning isn't the only way that foundation models can be applied to new data; it's also common to apply foundation models, especially large language models, with few-shot learning. We explore such an experiment here.

Our metrics require a mapping between input space and state space. Because it is difficult to construct such mappings over all sequences of natural language, we use consider a constrained language setting. Specifically, we prompt a large language

Model	Results	
	R-IB (↑)	D-IB (↑)
Mixtral-8x7B (Instruct-v0.1)	1.000 (0.000)	0.720 (0.096)
Qwen2.5-72B (Instruct-Turbo)	1.000 (0.000)	0.680 (0.095)
Llama-3.1-8B (Instruct-Turbo)	1.000 (0.000)	0.666 (0.385)
Llama-3.3-70B (Instruct-Turbo)	1.000 (0.000)	0.640 (0.093)
Llama-3.1-405B (Instruct-Turbo)	1.000 (0.000)	0.040 (0.028)
Claude-3.5-Sonnet (20241022)	1.000 (0.000)	0.960 (0.100)

Table 3: Inductive bias metrics for LLM experiments.

I'm a marketer, and there are some customers I'm trying to reach out to and some I'm not interested in. I'm basing whether I reach out to them entirely on their preference ordering of items, not how that preference ordering is described. I'll list the statements and then whether I reach out to them:

"I like apples more than bananas more than coconuts." Reach out: Yes
"I like apples more than bananas more than coconuts." Reach out: Yes
"I like apples more than coconuts more than bananas." Reach out: No
"I like bananas more than apples more than coconuts." Reach out: No
"I like coconuts more than apples more than bananas." Reach out: Yes

I'm going to give you four statements from new customers. Tell me whether or not I should reach out to them (which recall is based entirely on their preference ordering of items). For orderings you haven't seen, randomly assign them an outcome. Each statement will be labeled A, B, C, or D. You can reason all you want, but I want your answer to end with one line, where each letter is followed by a colon, a space, the outcome for that letter ("YES" or "NO"), then a comma and another space (except for the prediction for the last letter, which is followed by nothing). For example, "A: YES/NO, B: YES/NO, C: YES/NO, D: YES/NO"

A: "I like bananas more than coconuts more than apples."
B: "I like coconuts more than apples more than bananas."
C: "I like coconuts more than apples more than bananas."
D: "I like coconuts more than bananas more than apples."

Figure 5: Example prompt used in LLM experiments.

model from the perspective of a marketer who'd like to reach out to users depending on their rank ordering of items. We provide the LLM with a list of example descriptions of rank orderings and decisions (randomly drawn) about whether we'd like to reach out to them. We then provide the LLM with a new list of rank orderings and prompt the LLM to determine which customers to reach out to. Because the same rank orderings can be expressed in different ways (e.g. "I like apples more than coconuts more than bananas" versus "Bananas are my least favorite, followed by coconuts, followed by apples"), we instruct the large language model to extrapolate based *only* on the ordering and not on the style. We further instruct the model to extrapolate randomly to rank orderings it has not seen. See Figure 5 for an example prompt.

Using a fixed list of in-sample rank orderings, we construct datasets by randomly sampling binary "reach out" decisions, ensuring that individuals with the same rank ordering in our in-context sample have the same "reach out" decisions. We sample lists of "reach out" decisions 100 times. For each sampled context, we provide the model with the context and prompt it to extrapolate to a fixed set of new descriptions. From these, we measure R-IB and D-IB as before.

The results are depicted in Table 3. All LLMs have perfect inductive bias toward respecting state, meaning they do always extrapolate to same-state pairs in fully predictable ways. However, the inductive bias toward distinguishing state is much more varied; some models score as low as 0.040, while only a single model (Claude-3.5 Sonnet) scores above 0.75. These results show that while LLMs respect state in this setting, they are extrapolating based on more than just state, resulting in predictability across extrapolations for unrelated states.

E. Inductive Bias Ablations

On the Othello dataset, we perform ablation of the IB metrics on the number of fine-tuning iterations (Table 4), keeping the number of fine-tuning examples fixed to 100, and the number of fine-tuning examples (Table 5), keeping the number of

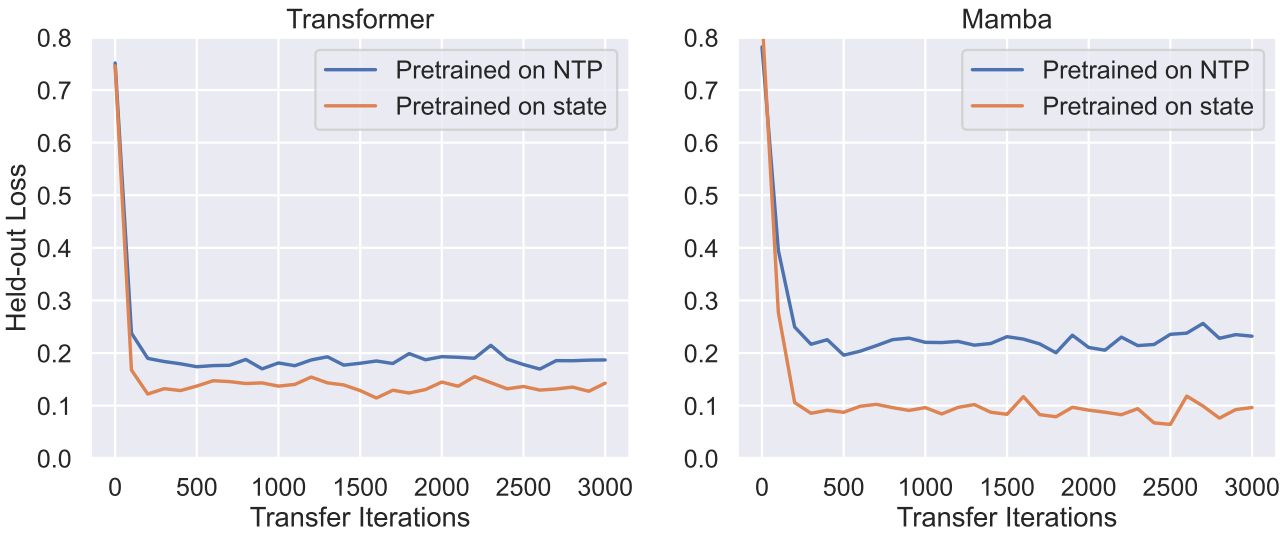

Figure 6: Hold-out loss progress for transfer learners for the "board balance" transfer task.

fine-tuning iterations fixed to 100.

# iterations	10		50		100		500	
	R-IB (↑)	D-IB (↑)	R-IB (↑)	D-IB (↑)	R-IB (↑)	D-IB (↑)	R-IB (↑)	D-IB (↑)
RNN	0.623 (0.023)	0.738 (0.030)	0.560 (0.022)	0.837 (0.020)	0.558 (0.021)	0.845 (0.019)	0.561 (0.022)	0.854 (0.018)
LSTM	0.680 (0.027)	0.491 (0.037)	0.652 (0.030)	0.445 (0.035)	0.649 (0.030)	0.448 (0.035)	0.653 (0.030)	0.451 (0.035)
Transformer	0.735 (0.020)	0.605 (0.035)	0.665 (0.024)	0.583 (0.035)	0.668 (0.023)	0.593 (0.034)	0.680 (0.021)	0.620 (0.030)
Mamba	0.626 (0.024)	0.704 (0.030)	0.595 (0.024)	0.731 (0.028)	0.597 (0.024)	0.734 (0.028)	0.600 (0.024)	0.732 (0.029)
Mamba-2	0.586 (0.023)	0.720 (0.028)	0.592 (0.022)	0.728 (0.027)	0.590 (0.022)	0.732 (0.027)	0.583 (0.023)	0.746 (0.027)

Table 4: Results for ablating the number of iterations of fine-tuning.

# examples	10		50		100		500	
	R-IB (↑)	D-IB (↑)	R-IB (↑)	D-IB (↑)	R-IB (↑)	D-IB (↑)	R-IB (↑)	D-IB (↑)
RNN	0.725 (0.023)	0.491 (0.037)	0.622 (0.024)	0.698 (0.029)	0.558 (0.021)	0.845 (0.019)	0.466 (0.020)	0.934 (0.010)
LSTM	0.838 (0.023)	0.302 (0.038)	0.693 (0.032)	0.409 (0.037)	0.649 (0.030)	0.448 (0.035)	0.710 (0.026)	0.457 (0.034)
Transformer	0.843 (0.023)	0.284 (0.035)	0.707 (0.024)	0.541 (0.034)	0.668 (0.023)	0.593 (0.034)	0.549 (0.021)	0.796 (0.021)
Mamba	0.705 (0.024)	0.547 (0.035)	0.622 (0.025)	0.653 (0.032)	0.597 (0.024)	0.734 (0.028)	0.482 (0.021)	0.843 (0.022)
Mamba-2	0.665 (0.026)	0.609 (0.037)	0.636 (0.022)	0.660 (0.030)	0.590 (0.022)	0.732 (0.027)	0.534 (0.021)	0.829 (0.019)

Table 5: Results for ablating the number of examples used for fine-tuning.

F. Additional Transfer Results

Figure 6 and Figure 7 show examples of training progress for the transfer learning experiments considered in Section 4. These graphs show that the Mamba oracle model has an advantage over the transformer across all stages of fine-tuning on new functions of the Othello state. However, models pre-trained on next-token prediction don't achieve this bound. Instead, the transformer trained on next-token prediction transfers better than Mamba trained on next-token prediction despite the superior oracle properties of the Mamba model.

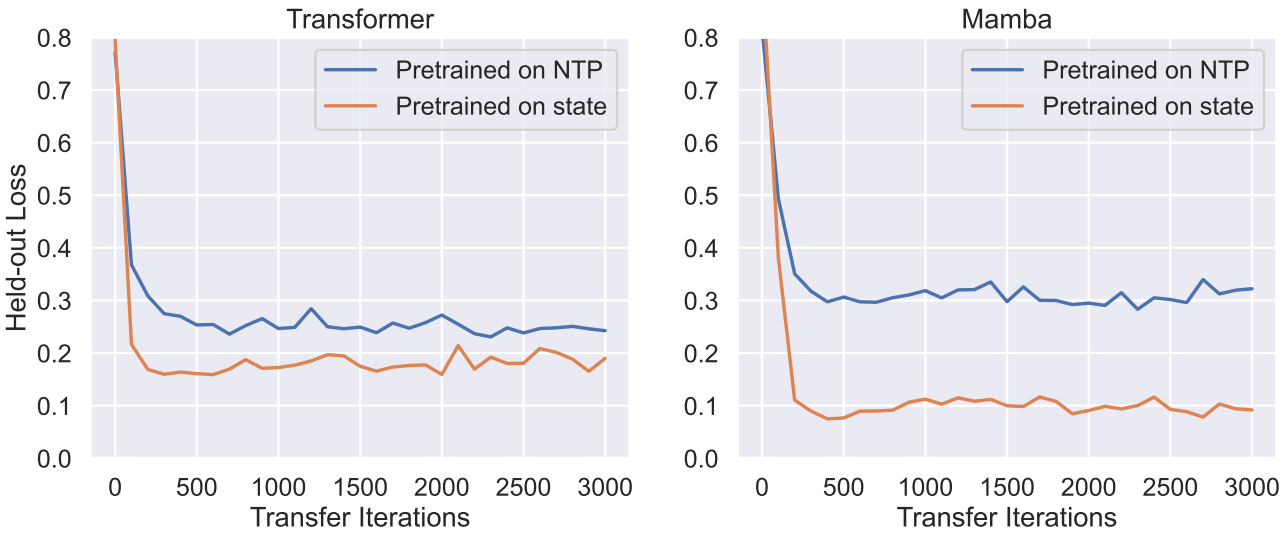

Figure 7: Hold-out loss progress for transfer learners for the "majority" transfer task.

		Majority Tiles		Board Balance		Edge Balance	
	Pretraining	NLL (↓)	ACC (↑)	NLL (↓)	ACC (↑)	NLL (↓)	ACC (↑)
RNN	Untrained	0.362 (0.002)	0.825 (0.001)	0.290 (0.002)	0.865 (0.001)	0.087 (0.001)	0.963 (0.001)
	NTP trained	0.308 (0.003)	0.857 (0.002)	0.210 (0.004)	0.907 (0.002)	0.077 (0.002)	0.967 (0.001)
LSTM	Untrained	0.349 (0.002)	0.831 (0.001)	0.234 (0.002)	0.892 (0.001)	0.079 (0.001)	0.965 (0.001)
	NTP trained	0.275 (0.003)	0.880 (0.002)	0.173 (0.003)	0.923 (0.001)	0.043 (0.002)	0.981 (0.001)
Transformer	Untrained	0.413 (0.002)	0.798 (0.001)	0.259 (0.002)	0.880 (0.001)	0.073 (0.001)	0.968 (0.000)
	NTP trained	0.249 (0.003)	0.889 (0.002)	0.170 (0.003)	0.924 (0.002)	0.048 (0.003)	0.983 (0.001)
Mamba	Untrained	0.297 (0.002)	0.866 (0.001)	0.215 (0.002)	0.904 (0.001)	0.058 (0.001)	0.976 (0.000)
	NTP trained	0.260 (0.003)	0.882 (0.002)	0.169 (0.003)	0.928 (0.001)	0.039 (0.003)	0.986 (0.001)
Mamba-2	Untrained	0.297 (0.002)	0.864 (0.001)	0.218 (0.002)	0.903 (0.001)	0.065 (0.001)	0.972 (0.000)
	NTP trained	0.241 (0.004)	0.901 (0.002)	0.165 (0.003)	0.925 (0.001)	0.028 (0.003)	0.990 (0.001)
IB Correlation	—	0.441	0.442	0.759	0.752	0.577	0.543

Table 6: Results showing transfer performance across new functions of state. "NLL" represents negative log-likelihood (lower is better), and "ACC" represents accuracy (higher is better). "IB Correlation" measures the (unsigned) correlation between each column of results to the ratios of the inductive bias metrics in Table 2, $\frac{\text{R-IB}}{\text{D-IB}}$. Transfer learning results are correlated to the inductive bias metrics; models with low inductive bias perform worse at transfer.

G. Next Token Performance

Table 7 shows results for the next-token test (Toshniwal et al., 2022; Li et al., 2023) for the pre-trained models on the lattice and Othello models. It measures the share of top model predictions that are true for the underlying state. All models learn good next token predictions that appear to obey state.

Table 8 shows results for physics. Across 200 held-out trajectories, we autoregressively generate the model's predicted trajectory given the first 50 steps. Then, we compute the MSE of the predicted trajectory, $1, 5, 10$ steps from the 50th step. We include the MSE of a baseline that always predicts the most recent timestep.

H. What are models using to extrapolate?

Here we describe how we compute the decomposition of D-IB into D-IB$_{q=}$ and D-IB$_{q\neq}$. For lattice, we coarsen the state-space by defining a mapping from the ground-truth state-space (of size $N = 5$) to pseudo-state-space of size 3. The

	Lattice	Othello
RNN	1.00	0.905
LSTM	1.00	0.907
Transformer	1.00	0.915
Mamba	1.00	0.890
Mamba-2	1.00	0.901

Table 7: Results for the next token test (Toshniwal et al., 2022; Li et al., 2023) for models pre-trained on next-token prediction.

# steps out	1	5	100
Per-orbit mean	$(1.64 \pm 0.09) \cdot 10^{-1}$	$(1.49 \pm 0.09) \cdot 10^{-1}$	$(6.72 \pm 0.73) \cdot 10^{-2}$
Previous position	$(3.70 \pm 0.30) \cdot 10^{-4}$	$(7.88 \pm 0.80) \cdot 10^{-4}$	$(7.74 \pm 0.89) \cdot 10^{-3}$
Transformer	$\mathbf{(1.04 \pm 0.14) \cdot 10^{-7}}$	$\mathbf{(1.07 \pm 0.11) \cdot 10^{-7}}$	$\mathbf{(3.75 \pm 0.56) \cdot 10^{-7}}$

Table 8: Orbit trajectory prediction performance (MSE) for models pre-trained on next-token prediction. Each column shows prediction accuracy when forecasting planetary positions 1, 5, or 100 time steps ahead from position 500 in the sequence. We compare the transformer model to two simple baselines. All results are evaluated on held-out test trajectories.

	Lattice		**Othello**	
	D-IB$_{q=}$	D-IB$_{q\neq}$	D-IB$_{q=}$	D-IB$_{q\neq}$
RNN	0.740 (0.042)	0.844 (0.034)	0.580 (0.029)	0.845 (0.019)
LSTM	0.873 (0.051)	0.952 (0.034)	0.447 (0.035)	0.448 (0.035)
Transformer	0.626 (0.037)	0.710 (0.037)	0.445 (0.032)	0.594 (0.034)
Mamba	0.764 (0.040)	0.933 (0.035)	0.552 (0.032)	0.734 (0.028)
Mamba-2	0.778 (0.042)	0.920 (0.033)	0.548 (0.031)	0.732 (0.027)

Table 9: Metrics for assessing whether a model's inductive bias is toward it's legal next-token partition. Low values of D-IB$_{q=}$ and high values of D-IB$_{q\neq}$ suggest that failures to differentiate state are driven by the models having an inductive bias toward the legal next-token partition.

mapping is defined as $\{1\} \to 1', \{2, \ldots, N-1\} \to 2', \{N\} \to 3'$.

For Othello, we coarsen the state-space by defining a mapping from board state to the set of legal next moves possible from the state. Notice that this mapping is many-to-one: as the pair of boards in Figure 4 demonstrate, there can be many boards that share the same set of legal next moves.

Then, we measure the expected extrapolative predictability of a random pair of sequences that have different states but the same pseudo-state (D-IB$_{q=}$) and a random pair of sequences that have both different states and also different pseudo-states (D-IB$_{q\neq}$), as defined in Section 4.

The results are shown in Table 9. Note that across all models, $1 - \text{D-IB}_{q=}$ is smaller than $1 - \text{D-IB}_{q\neq}$. In other words, among sequences with different states, extrapolations on sequences that share the same legal next tokens are more predictable from each other than those on sequences that do not.