The Smoothness Hypothesis: Why Next-Token Prediction Learns Language but Not Dynamical Systems

Erik Arne Mathiesen-Dreyfus

Gimle Labs, Paris
www.gimlelabs.com

Abstract

Next-token prediction (NTP) learns semantics remarkably well for language, video, and audio—yet fails for dynamical systems and protein folding, where reinforcement learning and structured search become necessary. We propose an explanation: the *smoothness* of the syntax-to-semantics map. In language, most local token perturbations preserve meaning; the map is Lipschitz-continuous on average. In dynamical systems, a single changed coefficient can trigger bifurcations, chaos, or divergence—the map has high, often unbounded, Lipschitz constant. We formalise this as *perturbation sensitivity* $\bar{\kappa}$, argue it determines which learning paradigm a domain requires, and discuss implications for building foundation models in structure-sensitive domains.

1 Introduction

Language models trained via next-token prediction acquire far more than surface statistics—they learn factual knowledge, reasoning patterns, and representations that transfer across tasks [3]. Similar success extends to video, where next-frame prediction yields models with implicit physical understanding [5], and audio [2]. Yet NTP conspicuously fails in other domains: protein language models learn useful sequence representations but cannot predict structure or function from sequence alone with the accuracy of specialised systems [10], and autoregressive models over symbolic equations require carefully designed curricula and architectural choices beyond raw next-token prediction [7]. Progress in these domains has required reinforcement learning, search, or explicit structural supervision [15, 6].

We propose a simple explanation. In language, most small perturbations to a token sequence produce only small changes in meaning—replace "cat" with "dog" and the semantics barely shifts. The syntax-semantics map is *smooth*, so a model minimising token prediction loss implicitly navigates a well-behaved semantic landscape. In dynamical systems, a single changed coefficient can transform a stable equilibrium into chaos [16]. The syntax-semantics map is *sensitive*: nearby points in token space can be arbitrarily far apart in behavioural space. NTP provides no pressure to learn these distinctions.

2 The Smoothness Hypothesis

Consider a domain with a syntactic space $\mathcal{S}$ of token sequences, a semantic space $\mathcal{M}$ of meanings or behaviours, and a syntax-semantics map $\phi : \mathcal{S} \to \mathcal{M}$. Since $\mathcal{S}$ is discrete, we define perturbation sensitivity as the worst-case semantic change under a single-token edit. Let $\mathcal{N}(s) = \{s' \in \mathcal{S} : d_{\mathcal{S}}(s, s') = 1\}$ be the edit-distance-1 neighbourhood of s. Then:

$$\kappa(s) = \sup_{s' \in \mathcal{N}(s)} d_{\mathcal{M}}(\phi(s), \phi(s')) \qquad \bar{\kappa} = \mathbb{E}_{s \sim p(\mathcal{S})}[\kappa(s)] \tag{1}$$

where $p(\mathcal{S})$ is the natural distribution over expressions in the domain. To make this concrete: for language, $d_{\mathcal{S}}$ is token edit distance and $d_{\mathcal{M}}$ is cosine distance in a sentence embedding space; for dynamical systems, $d_{\mathcal{S}}$ is token edit distance on symbolic equation strings and $d_{\mathcal{M}}$ is L^2 divergence of simulated trajectories.

Hypothesis 1 (Smoothness Hypothesis)**.** *NTP is an effective learning signal when the average perturbation sensitivity $\bar{\kappa}$ is bounded and small. When $\bar{\kappa}$ is large or unbounded, learning requires paradigms that evaluate semantic correctness directly: RL, search, or constrained generation.*

2.1 Why Smoothness Enables Next-Token Prediction

When a model is trained to predict the next token, it minimises cross-entropy over the token distribution:

$$\mathcal{L}_{\text{NTP}} = -\mathbb{E}_{s\sim p(\mathcal{S})}\left[\sum_t \log p_\theta(s_t|s_{<t})\right] \tag{2}$$

If $\bar{\kappa}$ is small, then for a typical expression s, all single-token edits $s' \in \mathcal{N}(s)$ produce semantically similar outputs: $d_{\mathcal{M}}(\phi(s), \phi(s')) \leq \bar{\kappa}$ on average. Syntactic neighbourhoods are semantically coherent. This has a direct consequence for learning: the NTP objective, a purely syntactic, token-level loss, becomes a faithful proxy for semantic quality. A model that learns to predict tokens well in a neighbourhood of s is implicitly learning about the *meaning* of that neighbourhood, because low $\bar{\kappa}$ guarantees that syntactically similar sequences carry similar meaning. Each training example provides semantic information about its entire edit neighbourhood, so sample efficiency improves as $\bar{\kappa}$ decreases.

Compression as a semantic objective. This leads to the central claim. Minimising $\mathcal{L}_{\text{NTP}}$ is equivalent to maximising compression of the token sequence. When ϕ is smooth, the statistical regularities that enable compression—co-occurrence patterns, distributional similarities, contextual predictability—are *entangled with* semantic regularities. The model cannot discover that "mat" and "rug" are distributionally interchangeable without implicitly learning that they are semantically similar: the Lipschitz condition guarantees that their distributional similarity *reflects* semantic similarity, not coincidence. The optimal compressor must therefore encode semantic structure, because semantic structure is what makes the data compressible.

This is stronger than the geometric property alone. The Lipschitz bound tells us that syntactic proximity implies semantic proximity. The compression claim goes further: it asserts that *optimising the NTP objective forces the model to discover and internalise these semantic relationships*, because they are the source of the statistical regularities that reduce cross-entropy.

When smoothness fails. When ϕ is non-smooth ($\bar{\kappa}$ large or unbounded), this coupling between compression and understanding breaks. Two equations differing by a single coefficient can exhibit qualitatively different dynamics, so syntactic proximity carries no semantic guarantee. A model can achieve excellent perplexity on symbolic equation strings by learning operator frequencies and syntactic patterns without learning anything about the dynamical behaviour those equations describe. Compression and understanding decouple: the statistical regularities that enable compression are no longer entangled with semantic structure, but are merely surface-level patterns.

A note on metric-dependence. The value of $\bar{\kappa}$ depends on three choices: the tokenisation of $\mathcal{S}$, the distance $d_{\mathcal{M}}$ on semantic space, and the distribution $p(\mathcal{S})$. Different tokenisations of the same domain yield different $\bar{\kappa}$: BPE tokenisation of equations produces a different sensitivity profile than tree-structured tokenisation. This is not a weakness of the framework but its constructive implication. If $\bar{\kappa}$ were an intrinsic property of the domain, there would be nothing to do about it. Because it depends on representation, *reducing $\bar{\kappa}$ through better representations* becomes a concrete engineering objective. Representation engineering, finding tokenisations, embeddings, or compositional languages that make the syntax-semantics map smoother, is essentially then the search for representations that minimise $\bar{\kappa}$.

3 Sensitivity Across Domains

Natural language (low $\bar{\kappa}$). Most single-token substitutions yield near-synonyms, coherent alternatives, or detectable errors—all with small semantic distance. The exceptions (negation, quantifiers) are statistically rare. This is not accidental: languages evolved for robust communication in noisy channels [4].

Video and audio (low–moderate $\bar{\kappa}$). Sensory signals are produced by continuous physical processes, so small perturbations in pixels or waveforms correspond to small changes in scene content. Tokenised representations inherit this when the tokeniser is well-trained.

Dynamical systems (high $\bar{\kappa}$). The symbolic description of an ODE maps to its behavioural semantics—trajectories, attractors, stability. This map is dramatically non-smooth: changing ρ in the Lorenz system from 24.05 to 24.74 transitions from a stable fixed point to chaos [16]. Changing "$+$" to "$-$" in an ODE can transform stable orbits into divergence. The Lipschitz constant is *unbounded* near bifurcation points, and such sensitive points are dense in the space of dynamical systems.

Protein sequences (high $\bar{\kappa}$). Single amino acid substitutions can cause misfolding or loss of function [18]. Epistasis makes mutation effects strongly context-dependent. The sequence-to-function landscape is rugged with many local optima [12]. Protein language models learn useful representations via NTP but cannot predict structure alone—AlphaFold required geometric supervision [6].

Code: a revealing challenge. Code appears to contradict the hypothesis: changing < to <= or + to − can completely alter program behaviour, suggesting high $\bar{\kappa}$. Yet NTP-based models generate remarkably competent code. The resolution

highlights an important refinement: $\bar{\kappa}$ is computed under the *natural distribution* $p(\mathcal{S})$, not uniformly over all possible edits. The distribution of real code is concentrated on a submanifold where the map is better-behaved—there are many equivalent ways to write the same program, strong naming conventions create redundancy, and most code follows well-established patterns. NTP succeeds on code not because the map is globally smooth, but because the prior is concentrated on regions where it is *locally* smooth. Critically, NTP-only code models still fail on the non-smooth tail: edge cases, off-by-one errors, and subtle logical bugs—precisely the cases where execution feedback (a semantic signal) is needed.

Table 1: Domain sensitivity and appropriate learning paradigms

Domain	$\bar{\kappa}$	Required paradigm
Natural language	Low	Next-token prediction
Audio / video	Low–Med	NTP + perceptual losses
Code	Medium	NTP + execution feedback
Theorem proving	High	NTP + proof search
Protein structure	High	NTP + geometric supervision
Dynamical systems	Very high	NTP + RL + structure search

4 From Local Syntax to Global Semantics: Why RL and Search Are Necessary

If NTP fails in non-smooth domains because its gradients are semantically uninformative, what makes reinforcement learning and search succeed? The answer lies in *where the learning signal originates*—and how it relates to the syntax-semantics map ϕ.

4.1 The Gradient Alignment Problem

NTP computes gradients of the form $\partial \mathcal{L}_{\text{NTP}}/\partial\theta$. These gradients point toward better *token prediction*—they tell the model how to adjust its parameters so that the next token is more likely under the training distribution. When ϕ is smooth, better token prediction implies better semantic quality (Section 2): the NTP gradient is *aligned* with the semantic gradient $\partial \mathcal{L}_{\text{semantic}}/\partial\theta$.

When ϕ is non-smooth, this alignment breaks. The NTP gradient and the semantic gradient can point in *completely different directions*. A parameter update that improves token prediction for an equation string may make the predicted dynamics worse, because the relationship between syntactic likelihood and dynamical behaviour is arbitrary near bifurcation points. Gradient descent in token space is semantically blind—it optimises a proxy that has lost its connection to the quantity of interest.

4.2 RL Bypasses the Non-Smooth Map

Reinforcement learning resolves this by computing a learning signal that originates in *semantic space*, not syntactic space. Policy gradient methods (REINFORCE [20], PPO [14]) estimate $\partial \mathbb{E}[R]/\partial\theta$ by:

1. *Sampling* complete outputs from the model: generate a candidate equation, protein sequence, or game trajectory.
2. *Evaluating* each output semantically: simulate the equation and compare to observed data, predict the protein's 3D structure, play out the game to a terminal state.
3. *Using the evaluation as reward*: weight the gradient of the generation probability by the semantic quality of the result.

Crucially, sampling-based policy gradient methods never differentiate through ϕ. They evaluate ϕ at sampled points, requiring ϕ to be *computable*, not smooth. The reward $R(\phi(s))$ is obtained by running the semantics (simulating, folding, playing), not by assuming any local structure in the syntax-semantics map. A single changed coefficient that produces chaos instead of stability will receive a low reward, and the policy gradient will steer away from it, something NTP gradients cannot do, because the token-level loss sees no difference. (Differentiable simulation can provide gradients *through* ϕ directly, but this requires ϕ to be at least locally smooth, returning us to the same constraint that limits NTP. The advantage of sampling-based RL is precisely that it avoids this requirement.)

4.3 Search Handles Rugged Reward Landscapes

Even with a semantic reward signal, the reward landscape over the space of candidate outputs can be *rugged*: many local optima, plateaus, and discontinuities. A model generating equations token-by-token faces a combinatorial space where most paths lead to semantically poor outputs, and the few good outputs are separated by large syntactic distances.

Search methods—Monte Carlo tree search [15], beam search with semantic scoring, evolutionary algorithms—address this by exploring the output space *globally* rather than following local gradients. MCTS, as used in AlphaGo, evaluates many candidate continuations through simulation (rollouts), then selects the most promising branch. It does not need the value function to be smooth—it queries it at discrete points and uses the *values* to guide exploration. This is maximally robust to non-smoothness: even if nearby positions have wildly different values, search will discover this through evaluation and act accordingly.

The combination of RL and search is more than additive. RL trains a value function $V_\theta(s) \approx \mathbb{E}[R|s]$ that *learns to approximate* the non-smooth syntax-semantics map through direct experience. Search then uses this learned value function to explore efficiently. The value function does not need ϕ to be globally smooth—it builds a local approximation from semantic evaluations, effective precisely in the regions the search visits.

4.4 Self-Play as Adaptive Semantic Curriculum

AlphaGo introduced a further insight: *self-play* generates an adaptive curriculum in semantic space. Rather than training on a fixed dataset (as NTP does), the agent generates its own training signal by playing against itself. Each game produces a semantic evaluation (win or loss), and as the agent improves, it encounters increasingly difficult semantic challenges—positions that require finer distinctions, deeper lookahead, and more precise evaluation.

This matters because non-smooth domains have vast semantic spaces that no fixed training corpus can cover. In dynamical systems, the space of possible behaviours (stable, periodic, chaotic, divergent) is enormous, and the boundaries between them are fractal in structure. A fixed dataset of (equation, behaviour) pairs will inevitably miss critical boundary regions. Self-play—or its analogues in other domains, such as iterative refinement with simulation feedback—generates training signal *where the model currently fails*, concentrating learning on the semantically difficult regions.

4.5 The Paradigm Spectrum

These observations suggest a spectrum of learning paradigms, ordered by how much of the semantic evaluation they internalise:

1. **NTP**: Learning signal is purely syntactic. Requires ϕ to be smooth for semantic learning to occur as a byproduct of compression.
2. **Behavioral fine-tuning**: Adds a semantic loss (simulation error, structural accuracy) but still uses gradient-based optimisation through the model. Requires ϕ to be at least locally smooth near good solutions.
3. **RL with semantic reward**: Learning signal is semantic, estimated through sampling. Requires ϕ to be evaluable, not smooth. Handles non-smooth maps but can struggle with sparse or deceptive rewards.
4. **Search with learned value function**: Explores the output space globally, guided by a learned semantic evaluator. Maximally robust to non-smoothness. The AlphaGo paradigm.

This progression mirrors observed practice across domains. Language: NTP alone suffices (smooth ϕ). Code: NTP + execution feedback, i.e., behavioural fine-tuning (moderately non-smooth). Protein structure: NTP pretraining + geometric supervision, i.e., semantic loss (non-smooth). Game playing: NTP-style imitation learning + RL + MCTS (highly non-smooth). Dynamical systems: the full stack—NTP pretraining for syntactic priors, RL for semantic reward, search for global exploration.

4.6 Evidence from Within Language: The Smoothness Gradient

A striking confirmation of the paradigm spectrum comes from the evolution of large language models themselves. Language is not uniformly smooth—it contains regions of varying perturbation sensitivity, and the history of LLM development traces a path through exactly the paradigm progression predicted by the smoothness hypothesis.

NTP era (GPT-2). Pure next-token prediction produces fluent text, coherent paragraphs, and basic factual recall [13]. These are the *smooth* regions of language: tasks where most token substitutions preserve the essential meaning, and where compression is a reliable proxy for understanding. GPT-2 excels at pattern completion, stylistic imitation, and simple question answering—precisely the tasks where the syntax-semantics map is locally smooth.

RLHF era (InstructGPT, ChatGPT). To follow instructions reliably and align with human intent, NTP alone proved insufficient [11]. The problem: instruction following occupies a *less smooth* region of language. The difference between "summarize this article" and "don't summarize this article" is a single token, but the required behaviour is completely different. Subtle phrasing changes—hedging, negation, scope ambiguity—can flip the intended semantics while barely changing the token distribution. RLHF provided what the smoothness hypothesis predicts: a learning signal rooted in *semantic evaluation* (human preference judgments) rather than token-level prediction.

Reasoning era (o1, o3, extended thinking). Mathematical reasoning, multi-step logic, and complex planning represent the *least smooth* regions of natural language. A single error in a chain of deductions invalidates the entire argument. The map from a reasoning trace (syntax) to its correctness (semantics) is highly non-smooth: two proofs differing by one step can be the difference between valid and invalid. The latest frontier models address this with inference-time search—generating multiple reasoning paths, evaluating intermediate steps via process reward models [9], and backtracking from errors. This is structurally analogous to MCTS: the model explores a tree of possible continuations, guided by a learned value function (the model's own assessment of whether a reasoning path is on track).

Chain-of-thought as smoothing (a conjecture). We might conjecture that chain-of-thought prompting [19] can be understood as a technique for *reducing the effective perturbation sensitivity* of a task. Instead of mapping directly from question to answer, a potentially non-smooth map, the model decomposes the task into intermediate steps: question $\to$ step$_1$ $\to$ step$_2$ $\to \cdots \to$ answer, where each individual step is a locally smooth mapping even when the end-to-end map is not.

Admittedly, this conjecture might only be directionally correct and not entirely true. Since, if each step i has sensitivity κ_i, the composed sensitivity is bounded by $\prod_i \kappa_i$, which can most likely *exceed* the end-to-end sensitivity. In that case decomposition makes it worse. In that case, decomposition only works if the intermediate representations genuinely factor the problem into locally smooth sub-problems rather than merely distributing the non-smoothness across more steps. It could be an interesting research topic to try to formalise when this holds, to then better understand when and why decomposition, and chain-of-thought, works.

Combined the above points, provide a unifying lens: the entire trajectory of LLM development, from GPT-2 through RLHF to reasoning models, is a progressive response to the non-smooth regions of language. The smoothness hypothesis is not just a cross-domain phenomenon (language vs. dynamical systems); it operates *within* language itself, explaining why increasingly powerful methods are needed for increasingly difficult tasks.

5 Discussion

5.1 NTP-Based Compression Does Not Imply Understanding

A prominent view in AI research holds that compression and intelligence are equivalent: a model that compresses data well must understand its structure [8]. An important distinction is necessary here. The compression-intelligence thesis in its strongest form refers to *Kolmogorov complexity*, the length of the shortest program that reproduces the data. A Kolmogorov-optimal compressor, by definition, captures all computable structure, including semantic structure, regardless of smoothness. Our claim is not about this idealised notion.

Our claim is about *NTP as a specific, bounded compression algorithm*. NTP minimises cross-entropy over token sequences, a particular compression procedure that operates locally in token space via gradient descent. When the syntax-semantics map is smooth, this specific procedure is forced to discover semantic regularities, because syntactic patterns *are* semantic patterns: the model cannot achieve low perplexity without learning that "the cat sat on the mat" and "the cat sat on the rug" are distributionally similar *because they mean similar things*. The Kolmogorov-optimal compressor would discover this regardless of smoothness; NTP discovers it *only because* the smoothness of language makes token-level compression a reliable proxy for semantic modelling.

In non-smooth domains, NTP-based compression and understanding decouple. A model can achieve excellent perplexity on symbolic equation strings by learning syntactic patterns—operator frequencies, variable naming conventions, common coefficient ranges—without learning anything about the dynamical behaviour those equations describe. Two equations differing by a single coefficient may be dynamically unrelated (one stable, one chaotic). NTP compression exploits syntactic regularity; understanding requires semantic structure. A Kolmogorov-optimal compressor would capture both, but NTP—operating locally in token space—captures only the former.

This resolves an apparent puzzle: why does scaling up protein language models improve sequence-level metrics without approaching AlphaFold-level structure prediction? Because the sequence-to-structure map is non-smooth, and NTP compression of amino acid sequences does not force the model to learn the physics of protein folding. AlphaFold's breakthrough required direct structural supervision: a learning signal rooted in semantics, not syntax.

5.2 Nuancing the Bitter Lesson

Sutton's "bitter lesson" [17] argues that general methods leveraging computation—search and learning at scale—ultimately outperform methods that leverage human domain knowledge. The smoothness hypothesis adds an important qualification: in smooth domains, *scale alone suffices*—more compute, more data, bigger models. The bitter lesson holds in its strongest form.

In non-smooth domains, *representation and structure matter*—the right inductive biases, the right search strategy, the right compositional language are not luxuries but prerequisites. The lesson is not equally bitter for all domains.

This is consistent with the empirical record. Pure scaling has produced remarkable results in language and vision—domains where the syntax-semantics map is smooth. In Go, chess, and protein folding—domains with high perturbation sensitivity—the breakthroughs came not from scaling alone but from combining learned models with structured search (MCTS in AlphaGo [15]) or domain-specific architectures (equivariant networks in AlphaFold [6]).

5.3 Relation to Representation Learning

The idea that smoothness matters for learning is not new. The smoothness assumption is a foundational prior in representation learning [1], and the manifold hypothesis—that high-dimensional data concentrates on low-dimensional smooth manifolds—underpins much of modern deep learning. Our contribution is not the observation that smoothness enables learning, which is well-established, but its specific application: explaining why next-token prediction succeeds as a *semantic* learning signal in some domains and fails in others, and connecting this to the NTP $\rightarrow$ RLHF $\rightarrow$ RL+search progression both across domains and within language itself.

6 Conclusion

The smoothness hypothesis offers a unified explanation for why NTP succeeds in some domains and fails in others: it depends on whether the syntax-semantics map is locally smooth. For structure-sensitive domains like dynamical systems, the path forward is not simply scaling autoregressive models. It requires direct semantic evaluation, structured search, and—most importantly—representations engineered to make the syntax-semantics map as smooth as possible. Finding the right language for a domain is not convenience; it is a prerequisite for learning.

References

[1] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

[2] Zalán Borsos, Raphaël Marinier, Damien Vincent, Eugene Kharitonov, Olivier Pietquin, Matt Sharifi, Dominik Roblek, Olivier Teboul, David Grangier, Marco Tagliasacchi, et al. AudioLM: A language modeling approach to audio generation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2023.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 1877–1901, 2020.

[4] Edward Gibson, Richard Futrell, Steven T. Piantadosi, Isabelle Dautriche, Kyle Mahowald, Leon Bergen, and Roger Levy. How efficiency shapes human language. *Trends in Cognitive Sciences*, 23(5):389–407, 2019.

[5] David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

[6] 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.

[7] Guillaume Lample and François Charton. Deep learning for symbolic mathematics. In *International Conference on Learning Representations (ICLR)*, 2020.

[8] Shane Legg and Marcus Hutter. Universal intelligence: A definition of machine intelligence. *Minds and Machines*, 17(4):391–444, 2007.

[9] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

[10] Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smestad, Allan Costa, Maryam Fazel-Zarandi, Tom Sercu, Sal Candido, and Alexander Rives. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, 2023.

[11] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:27730–27744, 2022.

[12] Frank J. Poelwijk, Daniel J. Kiviet, Daniel M. Weinreich, and Sander J. Tans. Empirical fitness landscapes reveal accessible evolutionary paths. *Nature*, 445(7126):383–386, 2007.

[13] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 2019.

[14] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[15] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[16] Steven H. Strogatz. *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering*. Westview Press, 2nd edition, 2015.

[17] Richard S. Sutton. The bitter lesson. `http://www.incompleteideas.net/IncIdeas/BitterLesson.html`, 2019.

[18] Nobuhiko Tokuriki and Dan S. Tawfik. Stability effects of mutations and protein evolvability. *Current Opinion in Structural Biology*, 19(5):596–604, 2009.

[19] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:24824–24837, 2022.

[20] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3–4):229–256, 1992.