Large Language Models as General Pattern Machines

Anonymous Author(s)
Affiliation
Address
email

Abstract: We observe that pretrained large language models (LLMs) are capable of autoregressively completing complex token sequences – from arbitrary ones procedurally generated by probabilistic context-free grammars (PCFG), to more rich spatial patterns found in the Abstract Reasoning Corpus (ARC), a general AI benchmark, prompted in the style of ASCII art. Surprisingly, pattern completion proficiency can be partially retained even when the sequences are expressed using tokens randomly sampled from the vocabulary. These results suggest that without any additional training, LLMs can serve as general sequence modelers, driven by in-context learning. In this work, we investigate how these zero-shot capabilities may be applied to problems in robotics – from extrapolating sequences of numbers that represent states over time to complete simple motions, to least-to-most prompting of reward-conditioned trajectories that can discover and represent closed-loop policies (e.g., a stabilizing controller for CartPole). While difficult to deploy today for real systems due to latency, context size limitations, and compute costs, the approach of using LLMs to drive low-level control may provide an exciting glimpse into how the patterns among words could be transferred to actions.

Keywords: large language models, in-context learning, language for robotics

1 Introduction

Large language models (LLMs) are trained to absorb the myriad patterns that are woven into the structure of language. They not only exhibit various out-of-the-box capabilities such as generating chains of reasoning [1, 2], solving logic problems [3, 4], and completing math puzzles [5], but also have been applied in robotics where they can serve as high-level planners for instruction following tasks [6, 7, 8, 9, 10, 11], synthesize programs representing robot policies [12, 13], design reward functions [14, 15], and generalize user preferences [16]. These settings rely on the few-shot in-context examples in text prompts that specify the domain and input-output format for their tasks [17, 18], and remain highly semantic in their inputs and outputs.

A key observation of our work – and perhaps contrary to the predominant intuition – is that an LLM's ability to represent, manipulate, and extrapolate *more abstract, nonlinguistic* patterns may allow them to serve as basic versions of *general pattern machines*. To illustrate this idea, consider the Abstract Reasoning Corpus [19], a general AI benchmark that contains collections of 2D grids with patterns that evoke abstract concepts (e.g., infilling, counting, and rotating shapes). Each problem provides a small number of input-output examples, followed by test input(s) for which the objective is to predict the corresponding output. Most methods (based on program synthesis) are manually engineered with domain-specific languages [20, 21, 22, 23] or evaluated on simplified extensions or subsets of the benchmark [24, 25, 26]. End-to-end machine learning methods only solve a handful of test problems [27]; however, our experiments indicate that LLMs prompted in the style of ASCII art (see Fig. 1) can correctly predict solutions for up to 85 (out of 800) problems given few-shot input-output example grids in-context – exceeding some of the best performing methods to date [20, 21, 23], without additional model training or fine-tuning. Even more surprisingly, we find this extends beyond ASCII numbers, and that when they are replaced with a mapping to *randomly sampled tokens* in the vocabulary, LLMs can still generate valid solutions. These

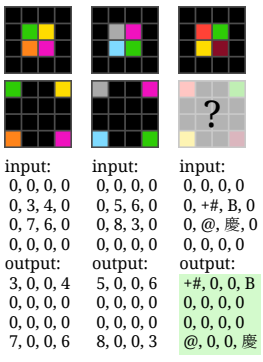

Fig. 1: LLMs out-of-the-box can complete (highlighted) complex ARC patterns [19] expressed in arbitrary tokens.

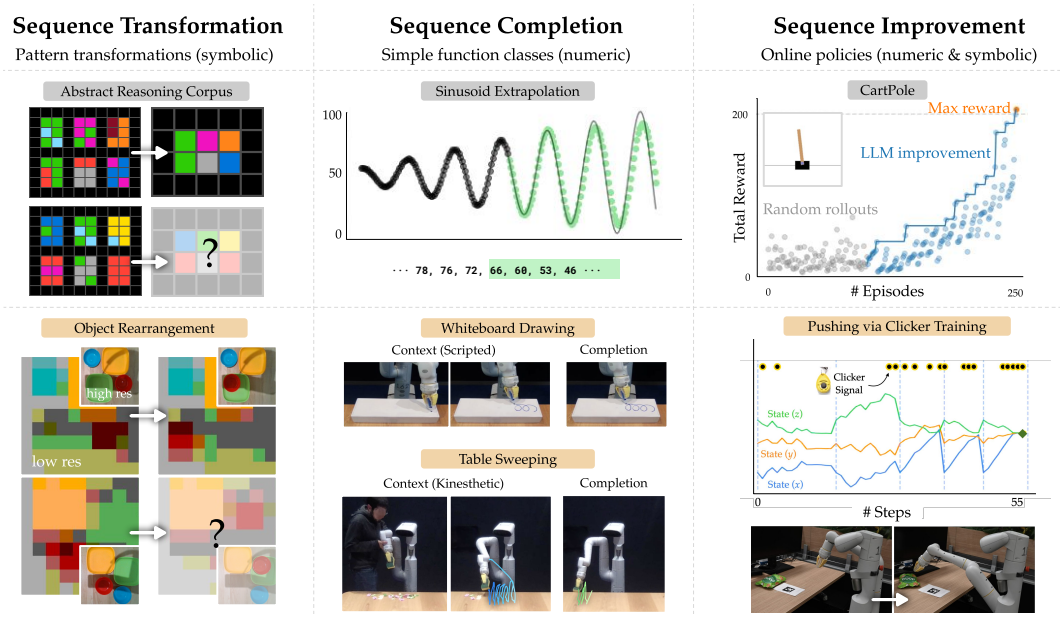

Fig. 2: Pretrained LLMs out-of-the-box may serve as basic versions of *general pattern machines* that can recognize and complete (in highlights) sequences of numeric or arbitrary (symbolic) tokens expressing abstract problems in robotics and sequential decision-making. Experiments show that to an extent, LLMs can in-context learn (i) sequence transformations (e.g., to reason over spatial rearrangements of symbols, for dynamics modeling and next state prediction on downsampled images), (ii) completion of simple functions (e.g., to extrapolate kinesthetic demonstrations), or (iii) meta-patterns to improve return-conditioned policies (e.g., discovering oscillatory behaviors to stabilize a CartPole).

results suggest an intriguing insight: that LLMs may exhibit more general capabilities of representing and extrapolating symbolic patterns, invariant to the specific tokens involved in the context [28]. We hypothesize that the capabilities that drive zero-shot pattern reasoning on ARC may allow general pattern manipulation at various levels of abstraction useful for robotics and sequential decision making, wherein a diverse array of problems involve patterns that may be difficult to reason about precisely in words. For example, a procedure for spatially rearranging tabletop objects could be represented using arbitrary tokens (see Fig. 2). As another example, optimizing a trajectory with respect to a given reward function can be framed as extrapolating a sequence consisting of state and action tokens with increasing rewards (i.e., turning LLMs into Decision Transformers [29] with in-context prompting and no training).

Orthogonal and complementary to efforts that develop multi-task policies by pre-training on large amounts of robot data [30], or robotics foundation models [31] that can be fine-tuned for downstream tasks [32, 33, 34], our goal is instead to (i) assess the zero-shot capabilities that LLMs may already contain to perform some degree of general pattern manipulation, and (ii) investigate how these abilities can be used in robotics. These capabilities are certainly not sufficient to replace specialized algorithms, but nonetheless are useful to characterize, and may help inform priorities for training generalist models in robotics.

We assess LLMs as pattern machines categorized into three areas: sequence transformation, sequence completion, and sequence improvement (see Fig. 2). First, we show that LLMs are capable of generalizing sequence-to-sequence transformations of increasing complexity with some amount of token invariance, and posit that this can carry over to spatial reasoning capabilities in robotic tasks. Next, we assess LLMs' ability to complete patterns from simple function classes (e.g., sinusoids) and show this can be applied to real robotic tasks like extending a wiping motion from kinesthetic demonstrations, or drawing patterns on a whiteboard. The combination of in-context learning sequence transformation and extrapolation, further enables LLMs to do basic forms of policy iteration. In this setting, we show that providing a series of reward-labeled trajectories as context, coupled with online interaction, can enable an LLM-based agent to learn to navigate through a small maze with obstacles, discover a stabilizing controller for the CartPole environment, and optimize simple human-in-the-loop trajectories via "clicker" reward training. Code, benchmarks, videos, and supplementary materials will be available at https://sites.google.com/view/llm-patterns

2 Related Work

Pattern reasoning by prompting pretrained LLMs with few-shot input-output examples is driven by in-context learning [35, 36]. The examples serve as a form of task specification, where the model is expected to complete further instances of the task by simply predicting what comes next. In-context learning extends the concept of "task prefixes" (predefined task-specific token sequences e.g., [37]), but swapped in with actual task examples instead. Brown et al. [36] observes that it improves (in particular, out-of-distribution generalization) from scaling model size. This is in contrast to scaling models for pre-training + fine-tuning, which has been shown to not necessarily improve OOD generalization on language tasks [38]. Nonetheless, despite compelling OOD generalization abilities, in-context learning still comes at a cost, as it continues to lag behind in terms of absolute performance on benchmarks compared to task-specific fine-tuning [35].

In-context learning is explicitly trained for by packing examples from the same task and dataset into the same context buffer that is fed as input to an LLM with an unsupervised autoregressive objective [36], this is sometimes referred to as meta-training. However, it can also emerge implicitly from training on unsupervised datasets where tokens exhibit a Zipfian distribution [39] on Transformer architectures, but not necessarily with recurrent architectures (e.g., vanilla RNNs or LSTMs) [39]. Other works have shown that in-context learning with Transformers can learn simple function classes on par with least squares [40, 41], and can generalize to a seemingly unbounded number of tasks (when trained on tasks from the same task family) better than multitask MLPs [42], with Bayesian interpretations of this phenomenon [43] [44].

In-context learning occurs during inference without gradient updates to the weights of the model, and can be differentiated from in-weights learning, which relies on information stored in the weights of the model during LLM training [45] (and can be useful for completion tasks such as "Abraham Lincoln was born in __"). Chan et al. [45] observes that generalization of in-context learning can be characterized as more "exemplar-based" (on the basis of similarity to in-context examples [46]), as opposed to generalization of in-weights learning which tends to be more "rule-based" (on the basis of minimal features that support category boundaries in the training data [47]). The vast capabilties of LLMs [36, 48, 49, 50, 51] have been driven by a combination of both forms of learning. In this work, we are particularly interested in in-context learning, and (depending on the task) using the semantic priors of numeric tokens (e.g., "0" to "100") to drive new capabilities such as in-context sequence completion (Section 5) and improvement (Section 6).

LLMs have been applied across a number of areas in robotics – most recently in decomposing high-level task domain descriptions in natural language to mid-level step-by-step plans [6, 7, 52, 53, 54, 55], robot code [56, 16, 13, 57], and planning domain definition languages [10]. These methods leverage the knowledge and commonsense priors stored in LLMs to compose new plans or parameterize primitive APIs, but whether LLMs can directly influence control (e.g., at the level of trajectories) remains an open problem. As a reaction to this, we investigate how the pattern reasoning capabilities of LLMs may drive various control tasks, to extend or optimize low-level action sequences. While it is possible to explicitly train models for these capabilities [58], we are particularly interested in studying the inherent abilities of LLMs out-of-the-box, which may have downstream implications in the role of language pre-training for building generalist embodied AI systems. Our findings may also benefit domains where data collection is expensive or difficult to scale. Perhaps most closely related to our work is Brooks et al. [59], which uses an LLM to represent a rollout-policy and world-model in-context, and then uses model-based Q-learning to drive policy improvement across a collection of toy environments. Our method (particularly for sequence improvement) can be seen as a substantial simplification of in-context policy iteration that supports both learning from demonstrations and in-context RL, driven by the generality of LLMs as pattern machines.

3 Language Models as General Pattern Machines

The capacity of LLMs to act as general pattern machines is driven by their ability to perform in-context learning on sequences of numeric or arbitrary tokens. An LLM typically represents sequence modeling autoregressively, with a decoder-only Transformer [60], by factorizing the probability of a sequence x, which is a sequence of symbols $(s_1, \ldots, s_n)$, into the product of conditional probabilities $p(x) =$

$\prod_{i=1}^{n} p(s_i|s_1, ..., s_{i-1})$. To perform in-context learning, the model can be conditioned with a prompt that provides the initial tokens in the sequence $s_{1:k} = (s_1, ..., s_k)$ and uses the model to complete $s_{k+1:n}$.

The adaptability of in-context learning lies in the amount of flexibility that can be packed into $s_{1:k}$ – this prompt sequence can itself contain many sequences, each an input-output pair, and perhaps additional task conditioning [35, 28]. Specifically, a model can in-context learn to complete a prompt which is a set of N examples $s_{1:k} = (x^1, x^2, ..., x^N)$ where each x^i is a variable-length sequence $(s_1^i, s_2^i, ..., s_{m^i}^i)$.

Rather than investigating in-context learning with natural language tasks [36], in this work we are interested in investigating more abstract notions of non-linguistic patterns. The following sections evaluate these capabilities across LLMs, and show how they can be used in robotics. By varying the notion of what each x^i should be, we can characterize in-context pattern learning capabilities into the following 3 categories.

- **Sequence Transformation** (Section 4): each $x^1, ..., x^{N-1}$ is a sequence-to-sequence input-output pair; i.e., $x^i = (x^i_{\text{input}}, x^i_{\text{output}})$, each subsequence of variable length, and x^N is the query input (x^N_{input}).
- **Sequence Completion** (Section 5): rather than containing input-output pairs, and rather than containing many examples of different sequences, the prompt $x = (s_1, ..., s_k)$ corresponds to discrete samples from a single generating function, for example $s_i = a \cdot \sin(bi)$, which can be extrapolated.
- **Sequence Improvement** (Section 6): each $x^1, ..., x^{N-1}$ is a collection of trajectories (potentially labeled with corresponding total rewards), and x^N prompts the model to "improve" the sequences by inferring a better one e.g., with least-to-most prompting [61] – this process can be iterative and applied to a variety of formulations, e.g., offline trajectory optimization or online in-context reinforcement learning.

4 Sequence Transformation

LLMs are capable of in-context learning the distribution of functions that represent sequence-to-sequence transformations by completing abstract patterns observed among several examples of input-output sequences $x^i = (x^i_{\text{input}}, x^i_{\text{output}})$ of arbitrary tokens, each drawn from a fixed alphabet $\mathcal{A}$. For example, suppose that we are given a string of input-output examples such as " 5 3 0, 3 5; 7 6 1, 6 7; 9 2 3, 2 9; 4 8 5,". Here $\mathcal{A}$ consists of tokens that represent space-prefixed digits 0–9, a comma token to separate inputs from outputs, and a semi-colon token to delineate examples from each other. A general pattern machine should infer the completion " 8 4" by recognizing that the pattern is to swap the first 2 tokens, then remove the 3rd.

We use the ARC benchmark [19] to evaluate LLMs on such sequence transformations, whereby token patterns are substantially more complex, covering a wide range of abstract spatial tasks: infilling, counting, translating and rotating shapes, etc. Each task comes with several input-output examples (3.3 on average), and 1-3 test inputs which can be represented as 2D grids. Sizes between inputs and outputs may differ and are not provided beforehand, thereby adding to the difficulty of applying standard machine learning algorithms, which typically assume fixed size. Autoregressive LLMs can be used for the ARC by flattening the grids and predicting each new output grid item in row-major order, which naturally supports variable length outputs. While LLMs are not originally trained for rasterizing spatial outputs in this way, we hypothesize that a general pattern machine would be capable of implicitly recognizing the long-range dependencies between rows (using positional encoding as a bias [62]) to pick up patterns that extend across the 2nd dimension.

Method	Total (of 800)
(d3) text-davinci-003	**85**
(d3) w/ random $\mathcal{A}$	†44±6
(d2) text-davinci-002 [48]	64
PaLM [50, 51]	42
(d1) text-davinci-001 [36]	11
(d1) finetuned	9
Ainooson et al., [22], 2023	**130
Kaggle 1st Place, 2022	*64
Xu et al., 2022 [21]	*57
Alford et al., 2021 [23]	35
Ferré et al., 2021 [20]	32

*Reported from [21] out of 160 object-oriented problems.
†Numbers averaged across 5 randomly sampled alphabets.
**Based on brute force search over a rich hand-designed DSL.

Tab. 1: LLMs out-of-the-box can solve a non-trivial number of problems on the ARC, competitive with the best existing methods using hand-crafted domain-specific languages [20, 23, 21].

Result: ARC benchmark. Our experiments in Table 1 show that LLMs prompted with the input grids represented as tokens drawn from an alphabet of digits, can correctly infer solutions for up to 85 problems out of the 800 ARC tasks. Surprisingly, this outperforms a number of recent systems [20, 23, 21] based on program synthesis that use manually engineered domain-specific languages (DSLs). While LLMs have

yet to surpass brute-force search methods [22] that compose functions from a handcrafted API of grid operators, LLMs are perhaps the best performing generalist (in-context learning) method that exists today.

Observation: consistent tokenization matters. The ARC can be found among the suite of tasks in BIG-Bench [63], but has often been overlooked since most language models appear to perform poorly (near or at zero performance). We observe this occurs due to the formatting of the benchmark, where grid elements are represented as neighboring characters in a string i.e., "8686" (instead of " 8 6 8 6"). While subtle, this difference is enough for certain Byte-Pair Encoding (or SentencePiece) tokenizers [64, 65] to group together multiple grid elements ("8" and "6") into a single token ("86") which maps to a different token embedding altogether in the vocabulary. This causes inconsistencies with how the patterns are expressed at the token level. For example, given a task expressed in a string "8686, 6868; 7979," if the LLM tokenizer groups together pairs of digits 86, 68, 79, respectively, then the sequential inductive patterns of the task (to swap and repeat individual digits) is lost. A simple work-around is to directly pass token indices or embeddings to the language model, or use token alphabets unlikely to be grouped by the tokenizer.

Observation: token mapping invariance. The hypothesis that LLMs can serve as general pattern machines stems from the observation that they can surprisingly still solve a non-trivial number of ARC problems using alphabets $\mathcal{A}$ sampled randomly from the LLM's token vocabulary. For instance, given a particular alphabet: 8 $\mapsto$ "falls", 6 $\mapsto$ "+#", 7 $\mapsto$ "UI", 9 $\mapsto$ "Chev", 3 $\mapsto$ "慶", 2 $\mapsto$ "2010", a pattern machine at sufficient proficiency can be expected complete the prompt "falls +# falls +#, +# falls +# falls; UI Chev UI Chev, Chev UI Chev UI; 慶 2010 慶 2010," by predicting " 2010 慶 2010 慶". For example, text-davinci-003 [48, 36] with the following mapping $\mathcal{A} = \{0:$ " offence",$1:$ "Subject",$2:$ " Lub",$3:$ " Fail",$4:$ " Chev",$5:$ " symb",$6:$ " swung",$7:$ " UI",$8:$ " escalate",$9:$ " Chromebook"$\}$ solves 52 ARC problems, and across 5 different random alphabets solves an average of 43.6 problems.

The implications of the token invariance are two-fold: (i) LLM performance under random alphabets may provide a closer "calibrated" estimate of its true underlying in-context pattern learning capabilities, and (ii) LLMs may be useful for in-context learning symbolic patterns that map to problems in robotics. For example, (i) Fig. 3 shows a grasp (Skittles) detector to output target coordinates within a downsampled image (with 6 in-context examples), and (ii) bottom left of Fig. 2 shows spatial rearrangement via simple forward dynamics where the red bowl moves to the green plate (with 9 in-context examples of downsampled images as inputs and outputs).

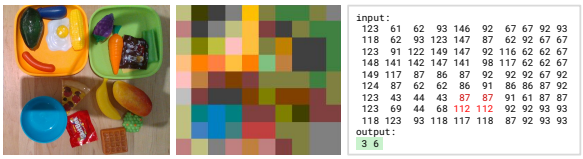

Fig. 3: Example LLM prediction as an in-context grasp detector.

Result: PCFG benchmark. The ARC is a difficult benchmark, and the performance falloff can be steep (and relatively uninformative) across LLMs with decreasing model size and data scale, making it difficult to measure incremental progress towards better pattern machines that could be used for sequence transformation in robotics. Therefore, we introduce an adjustable-difficulty benchmark, where sequence transformations are procedurally generated using the probabilistic context-free grammars (PCFG) in Hupkes et al. [66]. These transformations include a collection of lexical rules that may be composed, e.g., reverse, shift, swap, or repeat, etc. over the tokens in the input sequence x^i_{input} to generate x^i_{output}. For instance, the example transformation at the start of Section 4 can be defined as remove_second(swap(s_1,s_2),s_3). The complexity of these transformations can be carefully controlled by varying the number of tokens k used to express sequences $x^i = (s_1,...,s_k)$, and increasing the number of lexical rules w used to define the transformation. This is simply the identity function when $w = 0$, and progressively appears more complex (and more random) as $w \to \infty$. Table 2 aggregates PCFG pattern completion accuracy across different LLMs (OpenAI models in acronyms a1 - d3 and PaLM) over sequence length and complexity $k = [1,2,4,8,16,32]$ and $w = [0,1,3,7,15,31]$, each with 100 runs (see Appendix for independent ablations of varying k,w). These numbers provide a more

Method	Accuracy (%)
(d3) text-davinci-003	**75**
(d3) w/ random $\mathcal{A}$	$^{\dagger}58 \pm 1$
PaLM [50, 51]	74
(d2) text-davinci-002 [48]	69
(d1) text-davinci-001 [36]	60
(c1) text-curie-001	54
(b3) text-babbage-001	50
(a1) text-ada-001	39

†Numbers averaged across 5 randomly sampled alphabets.

Tab. 2: LLMs of varying sizes are capable of completing patterns procedurally generated with PCFG, averaged over a range of k and w.

thorough evaluation of pattern reasoning capabilities between LLMs; in particular, PCFG completion accuracy improves with model scale, and correlates with ARC performance. Note that PCFG is intended strictly for evaluation only (rather than for training data [66, 67]). We will release the PCFG benchmark.

5 Sequence Completion

In this section – complementary to transformations (Section 4) – we assess if LLM pattern reasoning can extend to settings where an LLM predicts a continuation of time series data points generated by a simple function class. We then demonstrate that such sequence completion can be operationalized on real robots to extend partial demonstrations of simple periodic motions. In this setting, the input context consists of a sequence $x = (s_1, ..., s_k)$ which packs a series of $l \leq k$ discrete samples from a generating function f. For example, the sequence of tokens " 1 2, 1 2, 1 2" may represent $l = 3$ samples from a constant vector-valued function f that outputs $(1,2)$. We use the LLM to extrapolate f by predicting $s_{k+1}, ..., s_n$ autoregressively.

Completion of sinusoids. We start with a simple example where LLMs extrapolate a function of the form $f(x) = a \cdot \sin(bx)$. As in Section 4, tokenization matters; we found it effective to discretize the outputs among integers 0–100, as these integers are represented by single tokens in the tokenizers of the LLMs we tested. Fig. 4 shows completions of the sine wave by text-davinci-003 over 11 trials given 3 and 5 periods as context, as well as average distance (computed by Dynamic Time Warping) of the generated predictions to the ground truth function values across several LLMs. Multiple LLMs produce near-perfect continuations of the sine wave, especially with more context (i.e., more periods of the sine wave). We additionally test the function family $ax \cdot \sin(bx)$ – in which the amplitude of the oscillations increases with x-values. Here, the LLM must extrapolate to new values unseen in the context, which highlights the utility of using a metric space for the outputs (0–100) where the LLM has priors over the scale of the different tokens. These functions also contain a "meta-pattern": the y-values increase, decrease, and then increase in a single period – and the amplitude of the function also increases over time. We also test the function $\frac{a}{2^x} \cdot \sin(bx)$, reminiscent of a stabilizing controller. Across these three functions, we observe that greater context and larger scale LLMs yield higher quality predictions.

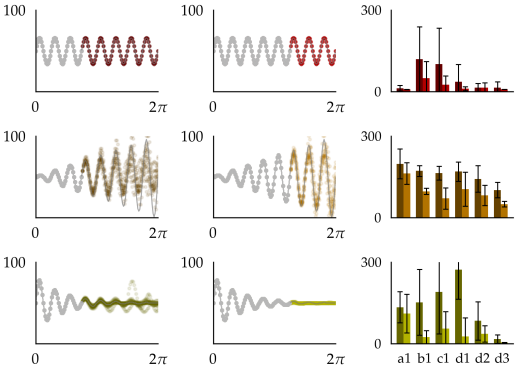

Fig. 4: LLMs (text-davinci-003) can extrapolate various functions $y = a \cdot \sin(bx)$ (top row), $y = ax \cdot \sin(bx)$ (middle row), and $y = \frac{a}{2^x} \sin(bx)$ (bottom row) given two different amounts of context. Overall, larger models make better predictions with lower error rates (right column).

Completion of periodic motions. We emphasize that the Sequence Completion capability above is domain-agnostic – i.e., we do not use any specialized prompts explaining what function should be completed, nor do we provide any linguistic grounding for the metric tokens. We can therefore operationalize this zero-shot capability of LLMs to simple open-loop motion extrapolation problems in robotics, e.g., by encoding a series of positions sampled from a demonstration, and predicting future positions. We test two simple tasks on a mobile robot manipulator: *Table Sweeping* and *Whiteboard Drawing* (both shown in Fig. 2).

In *Table Sweeping*, the goal is to continue a human-provided kinesthetic demonstration of sweeping a portion of a table (see middle Fig. 2). We encode the demonstration as a series of end-effector poses at approximately 3 Hz. Each demonstration lasts roughly 20-30 seconds. We represent the 7 DoF end-effector pose as a concatenation of Cartesian position and the quaternion, where each value is binned to an integer between 0 and 100, and the dimensions are delimited by spaces. We collect 30 demonstrations that demonstrate the sweeping motion. Note that demonstrations in this task are more noisy and are higher dimensional than the stylized sinusoid functions above. For each demonstration, we construct a context to consist of the first two-thirds of the provided demonstration, and treat the last one-third as the ground truth for the LLM to predict. Larger models quantitatively perform better (see Appendix).

In *Whiteboard Drawing*, the goal is to continue a scripted demonstration of drawing loops on a whiteboard (see Fig. 2). Loops are defined by parametric equations of the form $x = a_x \cos(bt) + d_x$ and $y =$

$a_y \sin(bt) + c_y t + d_y$. We execute the motions using position control and record the end-effector positions at 5 Hz, then discretize states in between 0 and 400, as finer motion is needed for this task. We provide part of the loop pattern in-context, and assess the ability to extrapolate from 2 loops to do a third loop. LLMs e.g., text-davinci-003 perform well – we show more completions with different loop styles in the Appendix.

6 Sequence Improvement

In this section, we explore the synergies between sequence transformation and completion – and investigate *improving* a sequence, such as trajectories in a sequential decision process, along some metric, such as a reward function. Here, we use an LLM to generate new sequences x^N conditioned on previous sequences $(x^1, ..., x^{N-1})$, which can represent previous iterations of the same sequence (or policy it represents).

The improvement can also be return-conditioned, given a reward (or cost) function $r(\cdot)$. By inserting as the first token(s) of each sequence its corresponding total reward $x = (r(x), s_1, ..., s_k)$, we can prompt the model at to conditionally "improve" by "just asking" [68] for a higher reward than those seen in-context (i.e., prompting LLMs out-of-the-box to become Decision Transformers [29]). New "rollouts" of the sequences can return new reward labels that then replace the original desired rewards with actual rewards. Iteratively performing this inference and accumulating more trajectories may jointly use the model's general notion of pattern transformation and extrapolation to perform improvement of sequences, which can be represented by numeric or symbolic tokens. Note that there are practical considerations, e.g., depending on the task or model, not all sequences can fit in context, so options could be to keep only the most recent, or the ones with the highest rewards if available (we refer to the Appendix for more discussion of the nuances here). In this section, we perform a series of targeted experiments on simple tasks, aiming to demonstrate the possibility of using pretrained LLMs for sequence improvement in trajectory and policy optimization.

Extrapolating simple meta-patterns among trajectories. Sequence improvement with LLMs enables a simple form of trajectory optimization for a *Marker in Cup* task on a Franka Panda robot, where we define the prefixed reward of a trajectory to be the negative distance between the final end-effector position and the cup (normalized between 70–100), and initialize the context with a collection of pre-recorded trajectories (stopping at 20%, 40%, 60%, and 80% of the way to the cup), delimited by newlines and prefixed by rewards up to 90 (see prompts in the Appendix). For this task, we represent trajectories as sequences of Cartesian positions, each dimension normalized between 0–100. We find that text-davinci-003, to an extent, is able to generalize the pattern and generate a trajectory that achieves a reward > 90. For this extrapolation to occur, we observe that the meta-patterns in the context are crucial: in Fig. 5(a), we compare the reward achieved by text-davinci-003 given contexts with different orderings of the trajectories (sorted by least-to-most reward, randomly permuted, or with/without reward annotations).

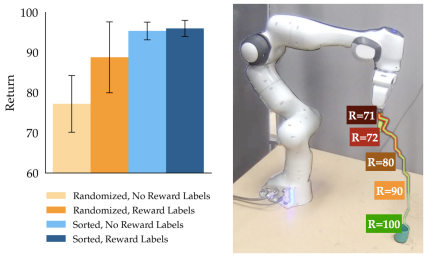

Fig. 5: LLM agents can generate new trajectories with increasing returns for a *Marker in Cup* task (right). Performance varies with different ways of building the context (left).

Generate higher-reward trajectories online. While LLMs can extrapolate from trajectories that exhibit clear meta-patterns among them, we find that this ability alone is limited for less trivial setups. Consider a simple 5×5 *Maze* navigation environment (illustrated in the Appendix) with a goal position, static obstacles, and a random starting position. Episodes terminate on agent collision or moving out of bounds. We initialize the context buffer with 20 trajectories (i.e., episodes) of agent grid positions generated by a random policy, sorted by rewards. These trajectories exhibit a slightly more complicated meta-pattern than in the *Marker in Cup* environment; we do not expect that LLMs can generate trajectories of higher reward immediately. With that said, we can consider an iterative, *online* setting, in which the LLM acts as an agent that interacts with the environment in a closed-loop fashion. The context consists of the N highest reward trajectories in sorted order, appended with a higher reward

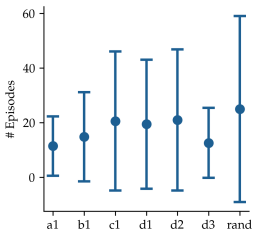

Fig. 6: LLM agents (a1 - d3, described in Appendix) on average discover solutions to *Maze* faster than random.

than was seen in the context, plus states and actions from the current partial trajectory (see Appendix for exact format and example contexts). Once an episode terminates, its trajectory is relabeled with the reward achieved, and inserted into the context at the appropriate position. In Fig. 6, we compare the number of episodes it takes for various LLM as agents to successfully reach the goal, and compare them to a random agent. On average, results suggest that LLMs are more efficient at discovering *Maze* solutions.

Result: discovering a simple *CartPole* controller. We show that using LLMs as agents in an online, closed-loop setting can discover a stabilizing controller for the *CartPole* environment (where observation tokens consist of pole angle and velocity, normalized to 0–100, actions are 0 (left) and 1 (right), maximum time horizon is 200). Fig. 7 (left) shows that the total reward (number of timesteps the CartPole is kept upright) improves on average across various LLMs over 100 episodes (where the first 100 are generated by random exploration). Fig. 7 (right) shows the evolution of trajectories over episodes, demonstrating that the LLM discovers "oscillatory" behaviors to keep the CartPole upright.

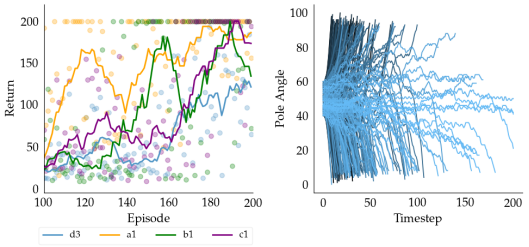

Fig. 7: Different LLM agents (d3 - c1) on average can improve trajectories (total rewards) with more *CartPole* episodes (left), and discovers "oscillatory behaviors" (right) to keep the cartpole upright (later episodes are brighter).

Result: online human-guided trajectory optimization. LLMs can also react to sparse binary reward signals (e.g., provided by a human) to adjust trajectories online. This is analogous to an implementation of "clicker dog training" [69, 70] but for robots). Every time step (2s), the robot executes an action, the human observes and chooses whether to give a reward. The (reward, state, action) tuples are added as in-context examples (with equal number of positives and negatives) to generate the next action based on the current state. This enables interactively teaching the robot to push an object via in-context sequence improvement.

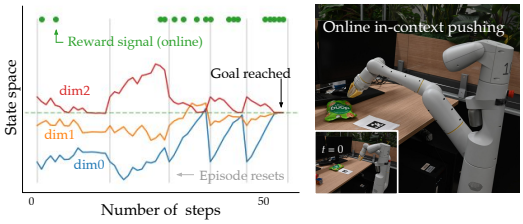

Fig. 8: LLMs can in-context react to sparse reward signals online to encourage an end effector to reach a desired goal.

7 Discussion

Implications. We are excited about the opportunities of using LLMs as pattern machines for robotics – from reasoning and extrapolating complex patterns as a prior for control, to online optimization of closed-loop policies via sequence optimization. These capabilities present several downstream implications, including (i) new perspectives on the role of language pretraining for generalist end-to-end robot learning models, and (ii) in-context learning of arbitrary patterns as a driving mechanism for policy improvement. LLMs also show promise for mixed autonomy settings – e.g., real-time pattern extrapolation for assistive teleoperation. We expect many these abilities to continue improving as large models expand from learning the patterns within language-only datasets, to multimodal domains (e.g., images, videos, etc.). While this work investigates the scope of in-context generalization without additional data collection or model training, these capabilities presumably may be further improved via domain-specific finetuning [71, 72].

Limitations & Future Work. Today, the inference costs (and monetary costs) of using LLMs for control is quite high. Predicting the next token for every sequence e.g., every dimension of every time step in a trajectory, involves querying an LLM. State-action spaces which are higher dimensional and/or greater precision also result in longer trajectory representations, and thereby the extent to which they can be extrapolated or sequence optimized is bounded by the context length of models (which scales quadratically with compute). These limitations may prevent deploying these models in practice, but could be lifted over time as current efforts in the community continue to drive improvements in LLM quantization [73] and inference efficiency [74]. As with any other language-only model, LLM-based control may (i) be unpredictable, and (ii) lack visual grounding; thus, it is not currently suitable for application outside of constrained lab settings. We leave the exploration of these important topics for future work.

References

[1] J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. Chi, Q. Le, and D. Zhou. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

[2] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*, 2022.

[3] M. Suzgun, N. Scales, N. Schärli, S. Gehrmann, Y. Tay, H. W. Chung, A. Chowdhery, Q. V. Le, E. H. Chi, D. Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*, 2022.

[4] A. Creswell, M. Shanahan, and I. Higgins. Selection-inference: Exploiting large language models for interpretable logical reasoning. *arXiv preprint arXiv:2205.09712*, 2022.

[5] A. Lewkowycz, A. Andreassen, D. Dohan, E. Dyer, H. Michalewski, V. Ramasesh, A. Slone, C. Anil, I. Schlag, T. Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *arXiv preprint arXiv:2206.14858*, 2022.

[6] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv:2204.01691*, 2022.

[7] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. *arXiv:2201.07207*, 2022.

[8] Y. Xie, C. Yu, T. Zhu, J. Bai, Z. Gong, and H. Soh. Translating natural language to planning goals with large-language models. *arXiv preprint arXiv:2302.05128*, 2023.

[9] Y. Ding, X. Zhang, C. Paxton, and S. Zhang. Task and motion planning with large language models for object rearrangement. *arXiv preprint arXiv:2303.06247*, 2023.

[10] B. Liu, Y. Jiang, X. Zhang, Q. Liu, S. Zhang, J. Biswas, and P. Stone. Llm+ p: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*, 2023.

[11] E. Zelikman, Q. Huang, G. Poesia, N. D. Goodman, and N. Haber. Parsel: A (de-) compositional framework for algorithmic reasoning with language models. *arXiv preprint arXiv:2212.10561*, 2023.

[12] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng. Code as policies: Language model programs for embodied control. In *arXiv preprint arXiv:2209.07753*, 2022.

[13] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg. ProgPrompt: Generating situated robot task plans using large language models. In *International Conference on Robotics and Automation (ICRA)*, 2023.

[14] M. Kwon, S. M. Xie, K. Bullard, and D. Sadigh. Reward design with language models. In *International Conference on Learning Representations (ICLR)*, 2023.

[15] H. Hu and D. Sadigh. Language instructed reinforcement learning for human-ai coordination. In *International Conference on Machine Learning (ICML)*, 2023.

[16] J. Wu, R. Antonova, A. Kan, M. Lepert, A. Zeng, S. Song, J. Bohg, S. Rusinkiewicz, and T. Funkhouser. Tidybot: Personalized robot assistance with large language models. *arXiv preprint arXiv:2305.05658*, 2023.

[17] J. Liu, D. Shen, Y. Zhang, B. Dolan, L. Carin, and W. Chen. What makes good in-context examples for gpt-3? *arXiv preprint arXiv:2101.06804*, 2021.

[18] Z. Zhao, E. Wallace, S. Feng, D. Klein, and S. Singh. Calibrate before use: Improving few-shot performance of language models. In *International Conference on Machine Learning*, 2021.

[19] F. Chollet. On the measure of intelligence. *arXiv preprint arXiv:1911.01547*, 2019.

[20] S. Ferré. First steps of an approach to the arc challenge based on descriptive grid models and the minimum description length principle. *arXiv preprint arXiv:2112.00848*, 2021.

[21] Y. Xu, E. B. Khalil, and S. Sanner. Graphs, constraints, and search for the abstraction and reasoning corpus. *arXiv preprint arXiv:2210.09880*, 2022.

[22] J. Ainooson, D. Sanyal, J. P. Michelson, Y. Yang, and M. Kunda. An approach for solving tasks on the abstract reasoning corpus. *arXiv preprint arXiv:2302.09425*, 2023.

[23] S. Alford. *A Neurosymbolic Approach to Abstraction and Reasoning*. PhD thesis, Massachusetts Institute of Technology, 2021.

[24] R. Assouel, P. Rodriguez, P. Taslakian, D. Vazquez, and Y. Bengio. Object-centric compositional imagination for visual abstract reasoning. In *ICLR Workshop on the Elements of Reasoning: Objects, Structure and Causality*, 2022.

[25] A. Moskvichev, V. V. Odouard, and M. Mitchell. The conceptarc benchmark: Evaluating understanding and generalization in the arc domain. *arXiv preprint arXiv:2305.07141*, 2023.

[26] V. Kolev, B. Georgiev, and S. Penkov. Neural abstract reasoner. *arXiv preprint arXiv:2011.09860*, 2020.

[27] T. Paparaju. Arc competition : Eda + pytorch cnn. https://www.kaggle.com/code/tarunpaparaju/arc-competition-eda-pytorch-cnn, 2022. Accessed: 2023-05-30.

[28] S. Min, X. Lyu, A. Holtzman, M. Artetxe, M. Lewis, H. Hajishirzi, and L. Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837*, 2022.

[29] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[30] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[31] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[32] S. Nair, A. Rajeswaran, V. Kumar, C. Finn, and A. Gupta. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022.

[33] I. Radosavovic, T. Xiao, S. James, P. Abbeel, J. Malik, and T. Darrell. Real-world robot learning with masked visual pre-training. In *Conference on Robot Learning*, 2023.

[34] S. Karamcheti, S. Nair, A. S. Chen, T. Kollar, C. Finn, D. Sadigh, and P. Liang. Language-driven representation learning for robotics. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[35] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[36] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[37] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21 (1):5485–5551, 2020.

[38] D. Hendrycks, X. Liu, E. Wallace, A. Dziedzic, R. Krishnan, and D. Song. Pretrained transformers improve out-of-distribution robustness. *arXiv preprint arXiv:2004.06100*, 2020.

[39] S. Chan, A. Santoro, A. Lampinen, J. Wang, A. Singh, P. Richemond, J. McClelland, and F. Hill. Data distributional properties drive emergent in-context learning in transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[40] S. Garg, D. Tsipras, P. S. Liang, and G. Valiant. What can transformers learn in-context? a case study of simple function classes. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[41] E. Akyürek, D. Schuurmans, J. Andreas, T. Ma, and D. Zhou. What learning algorithm is in-context learning? investigations with linear models. *arXiv preprint arXiv:2211.15661*, 2022.

[42] L. Kirsch, J. Harrison, J. Sohl-Dickstein, and L. Metz. General-purpose in-context learning by meta-learning transformers. *arXiv preprint arXiv:2212.04458*, 2022.

[43] S. M. Xie, A. Raghunathan, P. Liang, and T. Ma. An explanation of in-context learning as implicit bayesian inference. *arXiv preprint arXiv:2111.02080*, 2021.

[44] X. Wang, W. Zhu, and W. Y. Wang. Large language models are implicitly topic models: Explaining and finding good demonstrations for in-context learning. *arXiv preprint arXiv:2301.11916*, 2023.

[45] S. C. Chan, I. Dasgupta, J. Kim, D. Kumaran, A. K. Lampinen, and F. Hill. Transformers generalize differently from information stored in context vs in weights. *arXiv preprint arXiv:2210.05675*, 2022.

[46] R. N. Shepard and J.-J. Chang. Stimulus generalization in the learning of classifications. *Journal of Experimental Psychology*, 65(1):94, 1963.

[47] F. G. Ashby and J. T. Townsend. Varieties of perceptual independence. *Psychological review*, 93(2):154, 1986.

[48] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[49] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[50] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

[51] R. Anil, A. M. Dai, O. Firat, M. Johnson, D. Lepikhin, A. Passos, S. Shakeri, E. Taropa, P. Bailey, Z. Chen, et al. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*, 2023.

[52] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, P. Sermanet, N. Brown, T. Jackson, L. Luu, S. Levine, K. Hausman, and B. Ichter. Inner monologue: Embodied reasoning through planning with language models. In *arXiv:2207.05608*, 2022.

[53] A. Zeng, A. Wong, S. Welker, K. Choromanski, F. Tombari, A. Purohit, M. Ryoo, V. Sindhwani, J. Lee, V. Vanhoucke, et al. Socratic models: Composing zero-shot multimodal reasoning with language. *arXiv:2204.00598*, 2022.

[54] C. H. Song, J. Wu, C. Washington, B. M. Sadler, W.-L. Chao, and Y. Su. Llm-planner: Few-shot grounded planning for embodied agents with large language models. *arXiv preprint arXiv:2212.04088*, 2022.

[55] W. Huang, F. Xia, D. Shah, D. Driess, A. Zeng, Y. Lu, P. Florence, I. Mordatch, S. Levine, K. Hausman, et al. Grounded decoding: Guiding text generation with grounded models for robot control. *arXiv preprint arXiv:2303.00855*, 2023.

[56] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng. Code as policies: Language model programs for embodied control. *arXiv preprint arXiv:2209.07753*, 2022.

[57] G. Wang, Y. Xie, Y. Jiang, A. Mandlekar, C. Xiao, Y. Zhu, L. Fan, and A. Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

[58] M. Laskin, L. Wang, J. Oh, E. Parisotto, S. Spencer, R. Steigerwald, D. Strouse, S. Hansen, A. Filos, E. Brooks, et al. In-context reinforcement learning with algorithm distillation. *arXiv preprint arXiv:2210.14215*, 2022.

[59] E. Brooks, L. Walls, R. L. Lewis, and S. Singh. In-context policy iteration. *arXiv preprint arXiv:2210.03821*, 2022.

[60] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[61] D. Zhou, N. Schärli, L. Hou, J. Wei, N. Scales, X. Wang, D. Schuurmans, O. Bousquet, Q. Le, and E. Chi. Least-to-most prompting enables complex reasoning in large language models. *arXiv:2205.10625*, 2022.

[62] J. Su, Y. Lu, S. Pan, A. Murtadha, B. Wen, and Y. Liu. Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2021.

[63] A. Srivastava, A. Rastogi, A. Rao, A. A. M. Shoeb, A. Abid, A. Fisch, A. R. Brown, A. Santoro, A. Gupta, A. Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*, 2022.

[64] R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

[65] T. Kudo and J. Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*, 2018.

[66] D. Hupkes, V. Dankers, M. Mul, and E. Bruni. Compositionality decomposed: How do neural networks generalise? *JAIR*, 2020.

[67] Z. Allen-Zhu and Y. Li. Physics of language models: Part 1, context-free grammar. *arXiv preprint arXiv:2305.13673*, 2023.

[68] E. Jang. Just ask for generalization. In *https://evjang.com/2021/10/23/generalization.html*, 2022.

[69] L. C. Feng, T. J. Howell, and P. C. Bennett. Practices and perceptions of clicker use in dog training: A survey-based investigation of dog owners and industry professionals. *Journal of Veterinary Behavior*, 23:1–9, 2018.

[70] C. Chiandetti, S. Avella, E. Fongaro, and F. Cerri. Can clicker training facilitate conditioning in dogs? *Applied Animal Behaviour Science*, 184:109–116, 2016.

[71] S. Li, X. Puig, C. Paxton, Y. Du, C. Wang, L. Fan, T. Chen, D.-A. Huang, E. Akyürek, A. Anandkumar, et al. Pre-trained language models for interactive decision-making. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[72] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

[73] O. Zafrir, G. Boudoukh, P. Izsak, and M. Wasserblat. Q8bert: Quantized 8bit bert. In *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing-NeurIPS Edition (EMC2-NIPS)*, pages 36–39. IEEE, 2019.

[74] T. Dao, D. Fu, S. Ermon, A. Rudra, and C. Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.