

# SAS-Prompt: Large Language Models as Numerical Optimizers for Robot Self-Improvement

Heni Ben Amor<sup>1,2</sup>, Laura Graesser<sup>2</sup>, Atil Iscen<sup>2</sup>, David D'Ambrosio<sup>2</sup>, Saminda Abeyruwan<sup>2</sup>, Alex Bewley<sup>2</sup>, Yifan Zhou<sup>1</sup>, Kamalesh Kalirathinam<sup>1</sup>, Swaroop Mishra<sup>2</sup>, Pannag Sanketi<sup>2</sup>

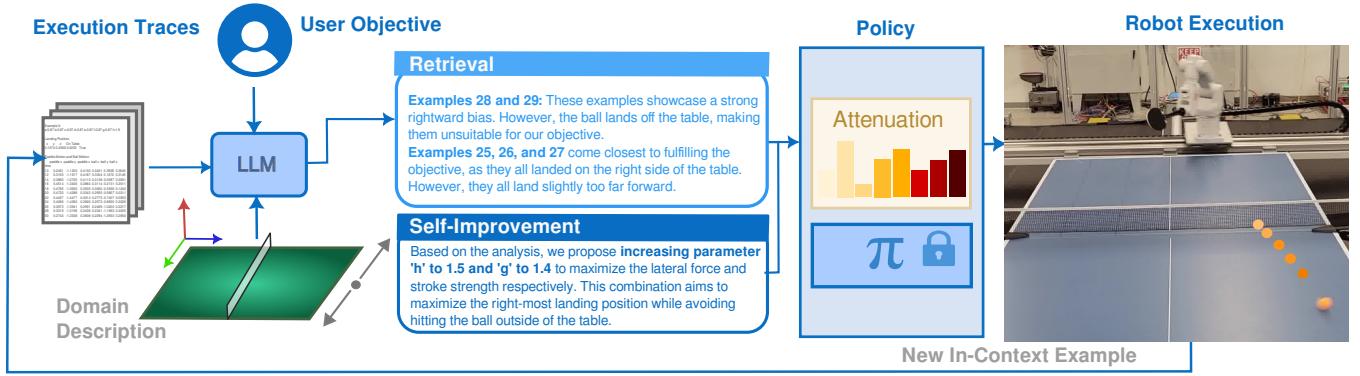


Fig. 1: A large language model is provided with (1) traces of robot behavior, (2) a description of the domain and (3) objectives of the human user. Using the SAS Prompt, the LLM iteratively optimizes attenuation parameters of a policy by retrieving previous examples that best fit the objectives and then synthesizing a new set of parameters in order to gradually improve robot behavior.

**Abstract**—We demonstrate the ability of large language models (LLMs) to perform iterative self-improvement of robot policies. An important insight of this paper is that LLMs have a built-in ability to perform (stochastic) numerical optimization and that this property can be leveraged for explainable robot policy search. Based on this insight, we introduce the SAS Prompt (Summarize, Analyze, Synthesize) – a single prompt that enables iterative learning and adaptation of robot behavior by combining the LLM’s ability to retrieve, reason and optimize over previous robot traces in order to synthesize new, unseen behavior. Our approach can be regarded as an early example of a new family of explainable policy search methods that are entirely implemented within an LLM. We evaluate our approach both in simulation and on a real-robot table tennis task. Project website: [sites.google.com/asu.edu/sas-lm/](http://sites.google.com/asu.edu/sas-lm/)

## I. INTRODUCTION

Large language models and related foundation models [1] have a number of properties that make them particularly appealing for robotics research. One central feature is their ability to provide an intuitive natural language interface to human users. Inputs and outputs to an LLM are formatted in natural language and allow even non-expert users to interact with a robot, for example by providing new task objectives or asking for clarification. Beyond language as an interface, a variety of other, more surprising properties have been identified such as generating chains of reasoning [2], complex code generation [3], being able to solve logic problems [4], and completing math puzzles [5], or the ability to match

and complete patterns [6]. As a result of these properties, LLMs have been used in robotics to generate high-level plans from human instructions [7], facilitate collaboration between multiple agents [8], synthesize robot policies as code [9], or design reward functions [10].

In this paper, we observe another *emergent property of LLMs* that renders them particularly well-suited for robot learning and self-improvement. We demonstrate that **LLMs can effectively perform stochastic numerical optimization** out-of-the-box without the need for additional retraining. Since numerical optimization is the central backbone of many learning and adaptation algorithms, e.g. Policy Gradient methods [11], we investigate whether LLMs can be used to perform learning through trial-and-error. Traditional methods that enable such self-improvement require a set of components such as (a) the identification of critical feature variables, (b) the design of a loss/reward function involving these features, and (c) an update rule in order to iteratively synthesize better parameters.

The main contribution of this paper is the SAS Prompt – an approach for robot learning and self-improvement that implements all three of the above steps within a single LLM prompt. **SAS Prompt enables robots to effectively understand and interpret previous robot behavior from in-context examples in order to perform policy search and synthesize new, unseen behavior.** The result is a family of algorithms in which self-improvement and numerical optimization is performed through repeated calls to an LLM with increasing context window size. An overview of the approach can be seen in Fig. 1. The system is provided with

<sup>1</sup>Authors are with the School of Computing and Augmented Intelligence, Arizona State University. Corresponding author: hbenamor@asu.edu.

<sup>2</sup>Authors are with Google DeepMind.

---

**Example 5:**  
a:1.1 b:1.2 c:0.7 d:1.1 e:1.1 f:1.1 g:1.1 h:1.5

**Landing Position:**

x	y	z	On Table
0.3207	0.7890	0.0143	True

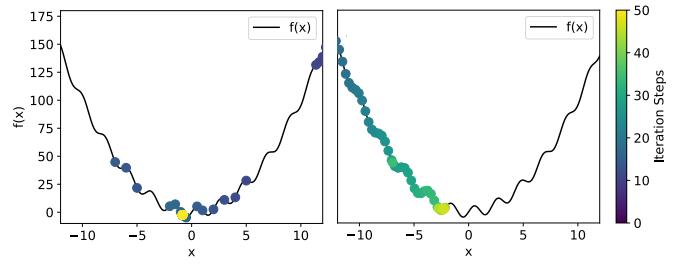
time	paddle x	paddle y	paddle z	ball x	ball y	ball z
1	0.2478	-1.1859	0.4236	0.2415	0.0619	0.2745
2	0.2993	-1.2453	0.4059	0.2346	-0.1359	0.2134
3	0.3417	-1.2889	0.3722	0.2184	-0.3311	0.1418
4	0.3483	-1.3131	0.3347	0.2093	-0.5129	0.0451
5	0.3317	-1.3254	0.3018	0.1884	-0.6759	...

Fig. 2: An example *execution trace* in the robot table tennis domain. The positions of the paddle and ball at every time step are provided along with robot control parameters (a – h).

a cache of previous robot trials in the form of execution traces – a tabular sequence of robot states (detailed in Fig. 2) which were observed during rollout. In turn, the LLM is tasked to generate improved robot control parameters, i.e., attenuation values, that will bring the robot behavior closer to the objectives specified by a human user. Instead of reward or fitness functions, objectives are expressed in natural language, making the process more accessible. Using its ability to process natural language and numerical data, the LLM can retrieve previous robot parameters that are in line with the human-stated objectives. In addition to this retrieval process, the LLM further analyzes the impact of the individual control parameters on task performance and, in turn, *synthesizes* a new set of parameters to improve robot performance. This analysis can be regarded as a textual formulation of a gradient: understanding the effect and amplitude of involved parameters allows the LLM to make informed decisions about changes and updates. The result is an iterative process, similar in spirit to policy search and other self-improvement algorithms, which progressively generates improved control parameters. A key advantage of our approach is the explainability of individual learning steps, since the LLM provides natural language justifications for all parameter choices. We validate this behavior through extensive experiments in a robot table tennis control task in both simulation and the real-world.

## II. RELATED WORK

Neural network approaches for incorporating language understanding and action generation have a storied tradition in robotics [12], [13]. Yet, a recent resurgence in interest can be attributed to the introduction of efficient embedding techniques [14] and large language models [15], [16]. For example, the works [17], [18] discussed new methodologies for imitation learning of *vision-language* models that enable robots to interpret human language. Going beyond interpretation, the SayCan [7] framework enables robots to leverage the reasoning capabilities of LLMs in order to decompose a task and generate executable plans. A variety of works have followed-up on this methodology [19]–[21] expanding on these planning and reasoning capabilities. To



**System Prompt:** You are an expert optimization assistant, helping me find the global minimum of a mathematical function. I will give you the function evaluation  $f(x)$  and the current iteration number at each step. Your goal is to propose input values ' $x'$  that efficiently lead us to the global minimum within a limited number of iterations (MAX\_STEPS).

Here's how we'll interact:

1. I will first provide MAX\_STEPS along with a few training examples of the form ' $x$ ,  $f(x)$ '.
2. You will provide your response in the following exact format:
  - \* Line 1: a new input ' $x$ ', aiming to minimize the function's value  $f(x,y)$ .
  - \* Line 2: a brief explanation of why you chose that input in one sentence, considering the current iteration.
3. I will then provide the function's value  $f(x)$  at your suggested point, and the current iteration.
4. We will repeat steps 2-3 until we reach the maximum number of iterations.

Remember:

- \* \*\*Assume no prior knowledge about the function's specific form.\*\*
- \* \*\*Balance Exploitation and Exploration:\*\* Early on, explore the input space broadly. As iterations increase, focus more on promising regions based on observed  $f(x)$  values.
- \* \*\*Be adaptable:\*\* Your approach might need to change based on the function's behavior and the remaining iterations. If you suspect a local minimum or slow progress for too long, try more exploratory values and then exploit any promising findings based on your understanding of the function.

### ----- LLM Numerical Optimization Process:

**Step: 0,  $x: -4.5$**

Explanation: We observe a decreasing trend in  $f(x)$  as  $x$  increases. Let's continue moving in the positive  $x$  direction, taking a slightly larger step to accelerate the search.

$f(x): 18.189$  % feedback to LLM from user

**Step: 1,  $x: -2.5$**

Explanation: The function value decreased significantly. It seems like moving with bigger steps is a good strategy for now – hence another relatively big jump.

$f(x): 11.045$  % feedback to LLM from user

...

**Step: 13,  $x: -0.715$**

Explanation: The function values at  $x = -0.719$  and  $x = -0.711$  are practically the same. There is a high chance that the minimum lies between these points or very close to them. Let's investigate the midpoint for a potentially better estimate.

$f(x): -4.439$  % feedback to LLM from user

Fig. 3: **Numerical Optimization inside an LLM:** the top figures depict two runs of an optimization process leveraging an LLM. The bottom text shows how the LLM is iteratively queried to minimize function  $f(x)$ . No gradients are provided.

incorporate learning, the work in [10] uses an LLM to translate human objectives into a cost function. In turn, a model-predictive planner finds optimal actions that minimize the generated cost function. In a similar vein, the Eureka system [22] uses evolutionary optimization over reward code in conjunction with reinforcement learning to acquire complex motor skills. Several works have explored the concept of *in-context learning* [23]–[25]. In these approaches, human demonstrations, corrections or feedback are used by the LLM to refine or generalize a robot skill. Outside of robotics, FunSearch [26] exploits LLM *hallucinations* in order to discover new mathematical insights. However, all of these approaches assume either (1) an optimization loop outside of the LLM or (2) human expert demonstrations that already solve the task. In our paper, we are particularly interested in the question of whether language models are able to adapt and self-improve a robot policy without the need for an external optimization loop or repeated coaching from a human, i.e., only robot execution and data log-

ging are performed outside of the LLM. A key insight is that such models have emergent capabilities for performing numerical optimization. Similar emergent properties have previously been reported for prompt optimization [27] and hyper-parameter optimization [28]. For example, the work in [27] demonstrated that LLMs can iteratively be queried to perform prompt optimization. The same paper also reports results on the traveling salesman task – a widely known NP-hard discrete optimization task. By contrast, in this paper, we focus in particular on robot policy parameters and investigate the suitability of LLMs for the iterative optimization of such numerical values.

### III. ROBOT TABLE TENNIS

Without loss of generality, we discuss our methodology within the context of robot table tennis [29]. For real-world experiments we use a 6 DoF arm mounted on top of two linear gantries, enabling motion in the 2d plane. A detailed description of the setup (8DoF in total) can be found in [30], [31]. In [32] we presented a robot that plays competitively against a human opponent. By contrast, in this paper, we are interested in the robot’s ability to translate human targeting instructions into control parameters. Rather than immediately controlling the robot actions, the LLM generates attenuation parameters for a lower-level controller (LLC). This LLC is able to hit the ball but may not be able to (a) reliably land it, (b) target a landing location, or (c) interpret human language instructions. We aim to modulate this simple LLC to enable language-conditioned robot control through retrieval and self-improvement of attenuation parameters. Attenuation parameters are similar in spirit to a residual layer, in which each of the outputs of the LLC are multiplied by linear scaling factor. The robot actions  $x \in \mathbb{R}^8$  generated by the LLC correspond to the velocities of the eight robot actuators. These robot actions are then multiplied with an attenuation vector  $\theta \in \mathbb{R}^8$  of the same length to yield the final robot action. We will discuss in this paper how this attenuation vector is generated through interaction with an LLM.

### IV. METHODOLOGY

In this section we motivate our approach and describe the SAS Prompt. A cornerstone of our methodology is our observation that LLMs are inherently capable of numerical optimization. Fig. 3 shows a general example of how the LLM can operate as a numerical optimizer to determine the global minimum of non-convex functions. To this end, the LLM is repeatedly asked to provide the next candidate for input parameter  $x$  that would lead to the highest reduction in function value. Neither the function  $f$  nor its derivatives are known to the LLM. Instead, only the corresponding function values  $f(x)$  are provided whenever it proposes a new candidate value for  $x$ . The result is a gradient-free optimization process that unfolds as a result of the repeated interaction with the LLM. Due to its natural language capabilities, the LLM can also be instructed to *explain* the parameter choices made at each step, thereby increasing the transparency of the underlying process.

### SAS Prompt

#### Domain Description:

The above examples hold the landing positions of a table tennis ball on the table. You are provided with the motion of the paddle and ball during each example. Regarding the size and dimensions of the table tennis table: the x dimension (along width) is in range [-0.76, 0.76], the y dimension (depth) is in range [-1.37, 1.37] and z (height) is in range [0.0, 2.0]. Positive x values are on the right side of the table tennis table, whereas negative x values are on the left side.

#### Objective:

Hit the ball as close as possible to the top edge of the table!

**Step 1:** Create a table that summarizes **Retrieval** each example. The summary should be precise and provide evidence. The table should also hold the corresponding parameter values a,b,c,d,e,f,g,h. The parameters control a robot which generated these examples.

**Step 2:** From the table above give me the parameters that are closest to fulfilling the above objective.

**Step 3:** Take these parameters **Self-Improvement** and the summary table and analyze the effect of the parameters a,b,c,d,e,f,g,h. Let’s think step by step!

**Step 4:** Based on this analysis, propose a new set of values for the parameters which will bring us closer to the objective than any of the previous examples. Avoid parameters that hit the ball outside of the table. Use exactly the following format and do not write anything else:

Fig. 4: **SAS Prompt:** the prompt provides the LLM with information about the domain, the user objectives and a step-by-step instruction on how to summarize and analyze the in-context examples. In turn, the final step is to synthesize a new set of parameters. Step 1 and 2 aim at retrieving previous examples that are best aligned with human objective. Step 3 and 4 aim at further optimizing these values to improve robot performance with regards to the same objective.

#### A. The Summarize, Analyze, Synthesize Prompt

Building upon the above insight, we introduce the SAS Prompt – a prompting technique combining retrieval and numerical optimization that enables learning and adaption from previous robot experiences. The SAS prompt realizes three objectives that allow the LLM to identify a set of robot control parameters:

- **Summarize:** all in-context examples are summarized and important features are extracted. This step forces the language model to process each example.
- **Analyze:** the impact of control variables on robot behavior is analyzed in order to identify correlations and dependencies. These insights inform the later parameter synthesis process.
- **Synthesize:** a set of control parameters is generated. This can be done through (a) pure retrieval, i.e., retrieve the best fitting parameter sets in the cache or, (b) by synthesizing a completely new set of parameters.

After querying the LLM using the SAS prompt, the proposed set of control parameters is executed on the robot yielding a new execution trace. The new trace of robot behavior is appended to the current LLM in-context cache in order to allow for incremental improvement and learning from prior experiences. Fig. 4 depicts the SAS Prompt in the context of the robot table tennis domain. The prompt can logically be divided into four main components:

**A. Domain Description and User Objective:** The first component of the SAS prompt is a description of the domain, along with the underlying task and environment. This component provides the necessary information needed to interpret the in-context examples and perform spatial reasoning. An essential element of this description is a definition of the coordinate system inherent to the task, e.g., the orientation and meaning of reported coordinate dimensions along with their ranges. In the example in Fig. 4, we specify the overall task (table tennis) as well as the table dimensions. The second component of the SAS prompt is the user objective. Here the instructions of the human are provided, e.g., “Hit to the top edge of the table”.

**B. Retrieval:** In the retrieval section of the SAS prompt, the LLM is first tasked with *summarizing* all in-context examples within a table. This step directs the LLM to inspect each example and extract critical features and insights before reaching a conclusion. The specific categories of the table may be provided with the prompt or can be omitted. In practice, we observe that the LLM will automatically choose fields that are best suited to answering the human query. The second step of the SAS prompt directs the LLM to identify in the table the examples / control parameters that are closest to fulfilling the given objective. This is a critical step since it requires the LLM to reason about the summarized table and identify one or multiple possible candidates. In traditional methods for self-improvement, this step is the result of applying the fitness or reward function to the collected samples and then selecting the top set of performing candidates for further adjustment.

**C. Self-Improvement:** The final component of the SAS prompt is self-improvement. The LLM is tasked with analyzing the effect of the variables on the in-context examples. The underlying goal is to identify the impact of each variable on

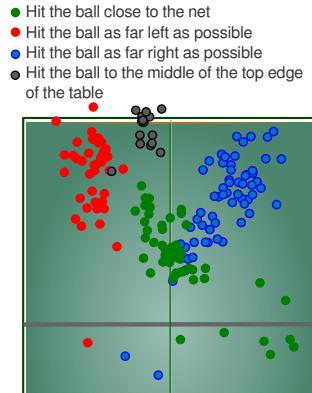


Fig. 5: **Retrieval** results: the retrieval part of the SAS prompt is used to retrieve robot control parameters which best fit the human instructions (printed above). Depicted coordinates are ball landing locations in a real-robot experiment.

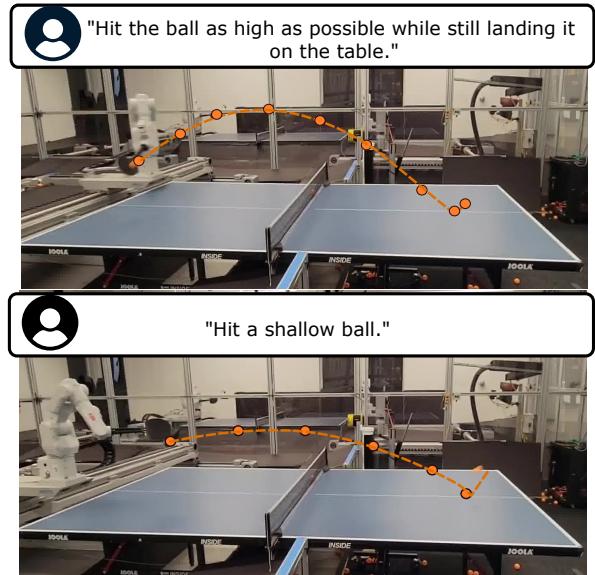


Fig. 6: Visualization of the ball trajectory in a retrieval task.

the execution trace. For each of the robot control variables, the LLM identifies possible correlations and dependencies. Understanding these relationships are critical to the subsequent adjustment step. At its core, the analysis can be seen as a hypothesis that is refined with every iteration. Based on this hypothesis, the LLM is then tasked with generating a new set of parameters that will bring the robot closer to achieving the task objectives than any one of the previous examples. This step mimics the prompt in the numerical optimization example introduced in Fig. 3. Every new parameter setting proposed by the LLM can be regarded as a sample in a gradient-free optimization process. Through the synthesis of gradually better parameter values, the LLM is implicitly executing an optimization process in which the cost function is provided in natural language. Note also, that the LLM can explicitly be asked to justify these choices in a text.

## V. EVALUATION AND RESULTS

### A. Numerical Optimization inside the LLM

One of the key tenets of this paper is that LLMs are capable of numerical optimization and that this property can be exploited for self-improvement in robotics. In order to provide evidence, we conduct a set of standard experiments for numerical optimization and compare the performance of Gemini 1.5 Pro with (a) gradient descent, (b) Adam [33], (c) Nelder-Mead [34], and (d) random search [35]. The experiment does not involve the full SAS prompt, but rather the minimal methodology described in the early example in Fig. 3. We evaluate on widely used optimization functions (minimization), namely Rastrigin’s [35] and Ackley’s function [36], in both 2D and 8D. To avoid any potential for guessing the answer (e.g., the LLM generating  $(0, 0)$  due to this being frequently used as location for optima in benchmarks) we added a constant shift to both functions. For every algorithm 50 experiments are performed, each with 100 update steps. Table I summarizes the results, as well as the

Alg.	2D Ackley	2D Rastr.	8D Ackley	8D Rastr.
Init $f(x)$	11.26±3.36	44.07±22.78	17.20±1.42	598.80±180.57
GD	9.49±3.52	24.40±20.90	15.59±1.39	505.29±182.62
Adam	6.22±5.31	23.14±19.59	15.10±2.14	<b>268.93±120.43</b>
Neld.-M.	8.14±4.92	23.26±19.83	15.54±1.67	426.42±187.84
Rand.-S.	11.15±3.45	43.35±25.20	17.15±1.44	589.04±182.92
LLM	<b>4.38±3.15</b>	<b>9.02±8.51</b>	<b>13.71±3.30</b>	340.87±156.47

TABLE I: Performance of different numerical optimization methods compared to LLM-based optimization.

initial values  $f(x)$  at start of optimization. We observe that the LLM favorably compares to other methods, including Adam and Nelder-Mead. Surprisingly, LLM performance is particularly strong with more dimensions. In general, our results confirm the hypothesis that LLMs are capable of optimization. It is also important to note that both gradient descent and Adam have access to gradients or perform additional steps to approximate a gradient whereas the above LLM-based results are completely gradient-free.

### B. SAS Retrieval Experiments: Real Robot

In this experiment we evaluate the ability of SAS prompt to retrieve the correct samples given an instruction from a human user. More specifically, we provide a set of potential user objectives and evaluate the ability of the SAS prompt in identifying the best-fitting in-context example. Hence, in this experiment, we do not use the self-improvement component of the SAS prompt and focus on the retrieval aspects of the task. The experiment is conducted by providing a set of instructions to the LLMs and recording the overall ball and robot motion. The LLM has access to 100 in-context examples. Fig. 5 depicts the landing locations of the ball. Note that the non-deterministic nature of the domain introduces noise into these results. Despite the relative lack in precision, we notice that a general tendency to correctly translate the human instructions into real-world actions. Especially in the case of hitting to the top edge of the table, we observe a faithful implementation of the commanded objective. In addition to above results, Fig. 6 shows another set of instructions in which the robot is instructed to modulate the height of the hit. A particular challenge in this task, is that the LLM is required to identify the peak height of each in-context example. This requires a careful analysis the provided ball trajectories in order to determine the peak height and include

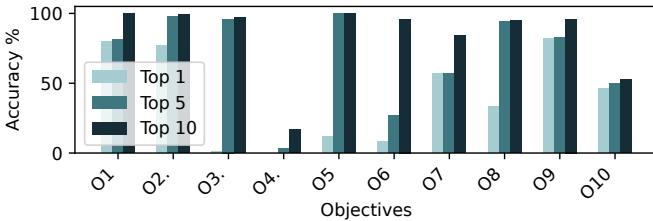


Fig. 7: Results of retrieval experiments with 10 predefined queries run 100 times through the SAS prompt. For each query we identify the ground truth and evaluate LLM retrieval performance.

it as part of the summary table in step 1 of the SAS prompt. To empirically evaluate the retrieval capabilities of our approach, we further performed a systematic evaluation in which 10 different instructions are provided to the LLM in 100 independent experiments. For each of these queries we identify the ground truth by programmatically inspecting the full set of available in-context examples. More specifically, we run the following objectives through SAS: **(O1)** “Play as far right as possible”, **(O2)** “Aim the ball at the leftmost edge of the table”, **(O3)** “Play the ball close to the net”, **(O4)** “Land the ball in the middle of the opponent’s side of the table” **(O5)** “Target the back-left corner of the opponent’s court”, **(O6)** “Aim the ball as close as possible to the back edge of the opponent’s court”, **(O7)** “Make sure to land the ball as close as possible to coordinate [0.2, 0.8, 0.0]”, **(O8)** “Hit the ball as close as possible to coordinate [-0.2, 0.8, 0.0]”, **(O9)** “Play the balls so it achieves the maximum peak height while still landing it on the table”, **(O10)** “Play as shallow a ball as possible”. Fig. 7 shows the results of this analysis. On average the LLM achieves 39.4% (Top-1), 68.89% (Top-5), and 83.70% (Top-10). In general, we can observe a tendency of the LLM to generate reasonable responses, even if the top-1 response is found in about 40% of cases. However, strong outliers are objectives O3 and O4 which produce a Top-1 score of less than 10%. This might be attributed to the ambiguity of the instruction, since the middle of the table is not a single point but rather an area.

### C. SAS Self-Improvement Experiments: Real Robot

In this experiment, we demonstrate the self-improvement capabilities within the robot table tennis domain. We initialize the in-context examples with 24 examples which cover a narrow stretch of the left side of the table. The user objective is specified as **“Hit the ball as far right as possible!”**. Since the in-context examples do not include any sample landing on the right side, the LLM needs to reason about the parameters and their influence on the landing site of the ball. At each iteration of the self-improvement process, an automated ball launcher system propels the ball in direction of the forearm of the robot. The robot executes its actions using the latest set of control parameters (attenuation values). In turn, the newly recorded execution trace is appended to the in-context examples and the SAS prompt is sent to the LLM in order to synthesize the next set of proposed parameters. The process is repeated in order to gradually improve the robot behavior. Fig. 8 shows the result of performing this process on the physical robot system. The transparent points in the figure indicate the in-context training samples available to the LLM at the beginning of the self-improvement process. We notice that within a small number of iterations the right side of the table is reached. By iteration 14, the robot is able to reliably hit to the (far) right side of the table. The figure also includes the response of the LLM at different steps of the learning process. We notice that the LLM attempts to identify correlations between individual parameters and the rightward landing objective. By iteration 14, it observes that **“Increasing parameter g consistently shifts the landing**

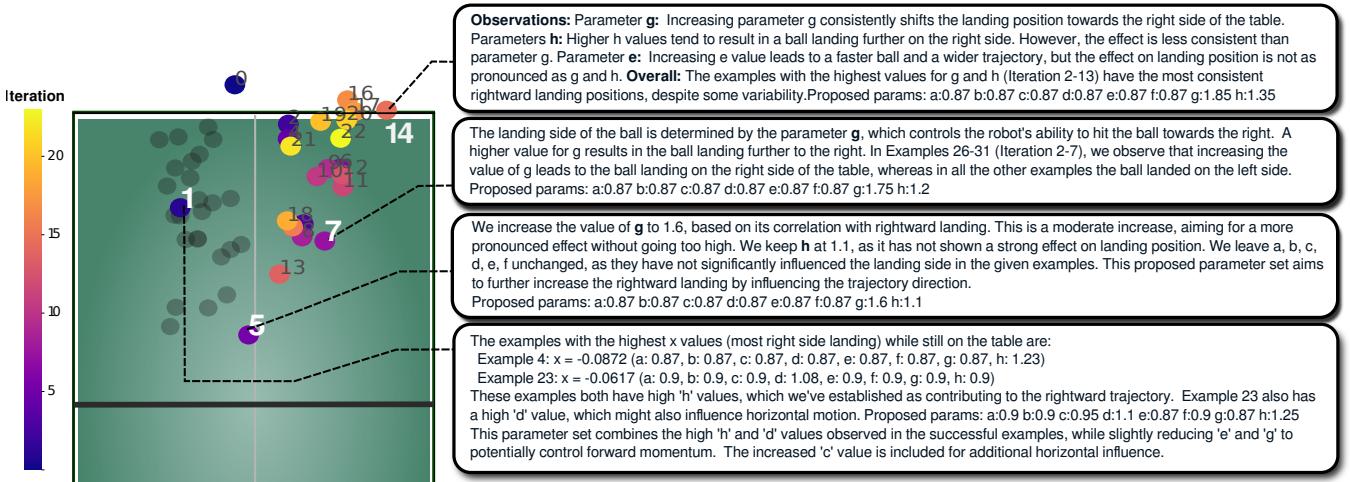


Fig. 8: **Self-Improvement via SAS**: transparent points show in-context examples at start; all sampled from a narrow area on the left side (transparent points). The task is to **Hit the ball as far right as possible**. Text boxes show responses at iterations.

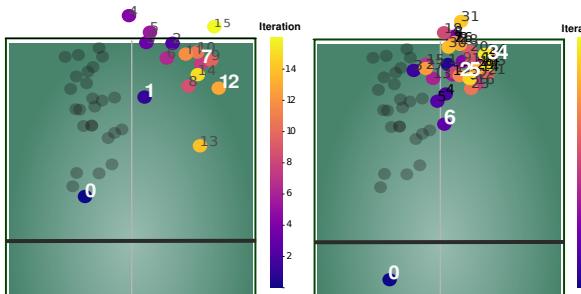


Fig. 9: **Self-Improvement Results**: visualization of the learning process during the self-improvement task on a real-robot. The objective is to learn to hit to the far right side.

**position towards the right side of the table. Parameters **h**: Higher **h** values tend to result in a ball landing further on the right side. However, the effect is less consistent than parameter **g**.** The analysis can be regarded as a textual gradient – it relates changes in parameter values to changes in task objectives and outcomes. This textual formulation of a gradient, in turn, informs the synthesis/update of new parameter values as can be seen in the provided justification at iteration 1, e.g., “**This parameter set combines the high ‘**h**’ and ‘**d**’ values observed in the successful examples, while slightly reducing ‘**e**’ and ‘**g**’ to potentially control forward momentum. The increased ‘**c**’ value is included for additional horizontal influence.**”. Please note that the LLM is articulating the real-world implications of parameter changes, i.e., forward momentum and horizontal influence, thereby attempting to uncover their inherent meaning. Additional self-improvement experiments on the real robot are shown in Fig. 9.

#### D. SAS Self-Improvement: Simulation Results

We also conducted a larger set of experiments using a publicly available robot table tennis simulation [37] in Mujoco. We provided three different self-improvement objectives, namely **(S1)** “*Hit the ball to the far right!*”, **(S2)** “*Hit the*

Objective	Init. Dist.		Final Dist.		Median Land X	Median Land Y
	Mean	Std.	Mean	Std.		
S1: Right	0.677	0.156	0.231	0.070	0.511	0.790
S2: Top	1.040	0.086	0.270	0.295	-0.213	1.233
S3: Left Cor.	1.277	0.147	0.342	0.201	-0.529	1.227

TABLE II: Distance to the goal before and after training and median position are shown. All numbers are shown in meters.

*ball to the top edge!*” and **(S3)** “*Hit the ball to the left corner!*” and ran 20 experiments for each objective. In each experiment, 30 iterations of self-improvement are conducted. In S2 and S3, we provided only in-context examples from the lower half of the table. Table II holds the results of this experiment. Depicted are the mean distance of the in-context examples from the goal at the start of the experiment, as well as the distance of the landing positions after self-improvement. Additionally, it also shows the median position from the final landing positions after self-improvement. We observe that in S1 the median X position is above 0.5m, showing a clear trend towards the right side of the table. In S2 and S3 the Y position is above 1.2m indicating a shift towards the top edge of the table (the edge of the table being at  $Y = 1.37$ m). Only in S3 is the X position shifted to the left of the table after training, i.e., negative X dimension. These results show that all three objectives were successfully realized with substantial changes in ball landing positions.

## VI. CONCLUSION

In this paper, we introduced the SAS prompt – a prompting technique for robot self-improvement and adaptation through explainable updates to policy parameters. Our approach provides a pathway towards new types of robot learning algorithms that are fully implemented within foundation models, e.g., all elements of the learning loop occur within the LLM. For future work, we hope to investigate scaling up the approach to higher dimensionality tasks and environments.

## REFERENCES

- [1] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, S. Buch, Dallas Card, Rodrigo Castellon, Niladri S. Chatterji, Annie S. Chen, Kathleen A. Creel, Jared Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kavin Ethayarajah, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren E. Gillespie, Karan Goel, Noah D. Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas F. Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, O. Khattab, Pang Wei Koh, Mark S. Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir P. Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Benjamin Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, J. F. Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghuathan, Robert Reich, Hongyu Ren, Frieda Rong, Yusuf H. Roohani, Camilo Ruiz, Jack Ryan, Christopher R'e, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishna Parasuram Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei A. Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. On the opportunities and risks of foundation models. *ArXiv*, 2021.
- [2] 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. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc., 2022.
- [3] Disha Shrivastava, Hugo Larochelle, and Daniel Tarlow. Repository-level prompt generation for large language models of code. In *International Conference on Machine Learning*, pages 31693–31715. PMLR, 2023.
- [4] Mirac Suzgun, Nathan Scales, Nathanael Schärlí, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksa Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. Challenging big-bench tasks and whether chain-of-thought can solve them. In Anna Rogers, Jordan L. Boyd-Graber, and Naoki Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 13003–13051. Association for Computational Linguistics, 2023.
- [5] Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Sloane, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:3843–3857, 2022.
- [6] Suvir Mirchandani, Fei Xia, Pete Florence, brian ichter, Danny Driess, Montserrat Gonzalez Arenas, Kanishka Rao, Dorsa Sadigh, and Andy Zeng. Large language models as general pattern machines. In *7th Annual Conference on Robot Learning*, 2023.
- [7] Alex Irpan, Alexander Herzog, Alexander Toshkov Toshev, Andy Zeng, Anthony Brohan, Brian Andrew Ichter, Byron David, Carolina Parada, Chelsea Finn, Clayton Tan, Diego Reyes, Dmitry Kalashnikov, Eric Victor Jang, Fei Xia, Jarek Liam Rettinghouse, Jasmine Chiehju Hsu, Jornell Lacanlale Quiambao, Julian Ibarz, Kanishka Rao, Karol Hausman, Keerthana Gopalakrishnan, Kuang-Huei Lee, Kyle Alan Jeffrey, Linda Luu, Mengyuan Yan, Michael Soogil Ahn, Nicolas Sievers, Nikhil J Joshi, Noah Brown, Omar Eduardo Escareno Cortes, Peng Xu, Peter Pastor Sampedro, Pierre Sermanet, Rosario Jauregui Ruano, Ryan Christopher Julian, Sally Augusta Jesmonth, Sergey Levine, Steve Xu, Ted Xiao, Vincent Olivier Vanhoucke, Yao Lu, Yevgen Chebotar, and Yuheng Kuang. Do as i can, not as i say: Grounding language in robotic affordances. 2022.
- [8] Zhao Mandi, Shreeya Jain, and Shuran Song. Roco: Dialectic multi-robot collaboration with large language models. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 286–299. IEEE, 2024.
- [9] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *arXiv preprint arXiv:2209.07753*, 2022.
- [10] Jacky Liang, Fei Xia, Wenhao Yu, Andy Zeng, Montse Gonzalez Arenas, Maria Attarian, Maria Bauzá, Matthew Bennice, Alex Bewley, Adil Dostmohamed, Chuyuan Fu, Nimrod Gileadi, Marissa Giustina, Keerthana Gopalakrishnan, Leonard Hasenclever, Jan Humplik, Jasmine Hsu, Nikhil J. Joshi, Ben Jyenis, Chase Kew, Sean Kirmani, Tsang-Wei Edward Lee, Kuang-Huei Lee, Assaf Hurwitz Michael, Joss Moore, Kenneth Oslund, Dushyant Rao, Allen Ren, Baruch Tabanpour, Quan Vuong, Ayzaan Wahid, Ted Xiao, Ying Xu, Vincent Zhuang, Peng Xu, Erik Frey, Ken Caluwaerts, Tingnan Zhang, Brian
- [11] Jan Peters and Stefan Schaal. Policy gradient methods for robotics. In *2006 IEEE/RSJ international conference on intelligent robots and systems*, pages 2219–2225. IEEE, 2006.
- [12] Stefan Wermter and Mark Elshaw. Learning robot actions based on self-organising language memory. *Neural Networks*, 16(5-6):691–699, 2003.
- [13] Yuuya Sugita and Jun Tani. A holistic approach to compositional semantics: a connectionist model and robot experiments. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems*, volume 16. MIT Press, 2003.
- [14] Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations*, 2013.
- [15] Gemini Team, Rohan Anil, Sébastien Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricu, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.
- [16] OpenAI. Gpt-4 technical report. *ArXiv*, abs/2303.08774, 2023.
- [17] Simon Stepputtis, Joseph Campbell, Mariano Phelipp, Stefan Lee, Chitta Baral, and Heni Ben Amor. Language-conditioned imitation learning for robot manipulation tasks. *Advances in Neural Information Processing Systems*, 33:13139–13150, 2020.
- [18] Corey Lynch and Pierre Sermanet. Language Conditioned Imitation Learning Over Unstructured Data. In *Proceedings of Robotics: Science and Systems*, Virtual, July 2021.
- [19] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. In *6th Annual Conference on Robot Learning*.
- [20] Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2998–3009, 2023.
- [21] Yan Ding, Xiaohan Zhang, Chris Paxton, and Shiqi Zhang. Task and motion planning with large language models for object rearrangement. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2086–2092. IEEE, 2023.
- [22] Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. In *The Twelfth International Conference on Learning Representations*, 2024.
- [23] Letian Fu, Huang Huang, Gaurav Datta, Lawrence Yunliang Chen, William Chung-Ho Panitch, Fangchen Liu, Hui Li, and Ken Goldberg. In-context imitation learning via next-token prediction. *arXiv preprint arXiv:2408.15980*, 2024.
- [24] Norman Di Palo and Edward Johns. Keypoint Action Tokens Enable In-Context Imitation Learning in Robotics. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024.
- [25] Jacky Liang, Fei Xia, Wenhao Yu, Andy Zeng, Montserrat Gonzalez Arenas, Maria Attarian, Maria Bauza, Matthew Bennice, Alex Bewley, Adil Dostmohamed, Chuyuan Kelly Fu, Nimrod Gileadi, Marissa Giustina, Keerthana Gopalakrishnan, Leonard Hasenclever, Jan Humplik, Jasmine Hsu, Nikhil Joshi, Ben Jyenis, Chase Kew, Sean Kirmani, Tsang-Wei Edward Lee, Kuang-Huei Lee, Assaf Hurwitz Michael, Joss Moore, Ken Oslund, Dushyant Rao, Allen Ren, Baruch Tabanpour, Quan Vuong, Ayzaan Wahid, Ted Xiao, Ying Xu, Vincent Zhuang, Peng Xu, Erik Frey, Ken Caluwaerts, Tingnan Zhang, Brian

- Ichter, Jonathan Tompson, Leila Takayama, Vincent Vanhoucke, Izhak Shafran, Maja Mataric, Dorsa Sadigh, Nicolas Heess, Kanishka Rao, Nik Stewart, Jie Tan, and Carolina Parada. Learning to learn faster from human feedback with language model predictive control, 2024.
- [26] Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Kumar, Emilien Dupont, Francisco Ruiz, Jordan Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. Mathematical discoveries from program search with large language models. *Nature*, 625, 12 2023.
- [27] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. September 2023.
- [28] Michael Zhang, Nishkrit Desai, Juhan Bae, Jonathan Lorraine, and Jimmy Ba. Using large language models for hyperparameter optimization. In *NeurIPS 2023 Foundation Models for Decision Making Workshop*, 2023.
- [29] Katharina Mülling, Jens Kober, Oliver Kroemer, and Jan Peters. Learning to select and generalize striking movements in robot table tennis. *The International Journal of Robotics Research*, 32(3):263–279, 2013.
- [30] David B D’Ambrosio, Navdeep Jaitly, Vikas Sindhwani, Ken Oslund, Peng Xu, Nevena Lazic, Anish Shankar, Tianli Ding, Jonathan Abelian, Erwin Coumans, Gus Kouretas, Thinh Nguyen, Justin Boyd, Atil Iscen, Reza Mahjourian, Vincent Vanhoucke, Alex Bewley, Yuheng Kuang, Michael Ahn, Deepali Jain, Satoshi Kataoka, Omar E Cortes, Pierre Seranet, Corey Lynch, Pannag R Sanketi, Krzysztof Choromanski, Wenbo Gao, Juhana Kangaspunta, Krista Reymann, Grace Vesom, Sherry Q Moore, Avi Singh, Saminda W Abeyruwan, and Laura Graesser. Robotic Table Tennis: A Case Study into a High Speed Learning System. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023.
- [31] Saminda Wishwajith Abeyruwan, Laura Graesser, David B D’Ambrosio, Avi Singh, Anish Shankar, Alex Bewley, Deepali Jain, Krzysztof Marcin Choromanski, and Pannag R Sanketi. i-sim2real: Reinforcement learning of robotic policies in tight human-robot interaction loops. In *Conference on Robot Learning*, pages 212–224. PMLR, 2023.
- [32] David B. D’Ambrosio, Saminda Abeyruwan, Laura Graesser, Atil Iscen, Heni Ben Amor, Alex Bewley, Barney J. Reed, Krista Reymann, Leila Takayama, Yuval Tassa, Krzysztof Choromanski, Erwin Coumans, Deepali Jain, Navdeep Jaitly, Natasha Jaques, Satoshi Kataoka, Yuheng Kuang, Nevena Lazic, Reza Mahjourian, Sherry Moore, Kenneth Oslund, Anish Shankar, Vikas Sindhwani, Vincent Vanhoucke, Grace Vesom, Peng Xu, and Pannag R. Sanketi. Achieving human level competitive robot table tennis. *arXiv preprint arXiv:2408.03906*, 2024.
- [33] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, 2015.
- [34] John A Nelder and Roger Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.
- [35] LA Rastrigin. The convergence of the random search method in the extremal control of a many parameter system. *Automaton & Remote Control*, 24:1337–1342, 1963.
- [36] David Ackley. *A connectionist machine for genetic hillclimbing*, volume 28. Springer science & business media, 2012.
- [37] Fabian Otto, Onur Celik, Dominik Roth, and Hongyi Zhou. Fancy gym.