PRE-TRAINED LARGE LANGUAGE MODELS FOR INDUSTRIAL CONTROL

LEI SONG, CHUHENG ZHANG, LI ZHAO, JIANG BIAN

Microsoft Research Asia
{lei.song, chuhengzhang, li.zhao, jiang.bian}@microsoft.com

Abstract. For industrial control, developing high-performance controllers with few samples and low technical debt is appealing. Recently, foundation models are shown to be powerful in dealing with various problems with only few (or no) demonstrations, owing to the rich prior knowledge obtained from pre-training with the Internet-scale corpus. To explore the potential of foundation models in industrial control, we design mechanisms to select demonstrations and generate the prompt for foundation models, and then execute the action given by the foundation models. We take controlling HVAC (Heating, Ventilation, and Air Conditioning) for buildings via GPT-4 (one of the first-tier foundation models) as an example, and conduct a series of experiments to answer the following questions: 1) How well can GPT-4 control HVAC? 2) How well can GPT-4 generalize to different scenarios for HVAC control? 3) How do different designs affect the performance? In general, we found GPT-4 achieves a performance comparable to RL methods but with fewer samples and lower technical debt, indicating the potential of directly applying foundation models to industrial control tasks.

1. Introduction

Reinforcement learning (RL), though being one of the most popular decision making methods, suffers from sample inefficiency issue and thus high training costs (see e.g., Botvinick et al., 2019). This issue may be fundamentally hard for the traditional RL paradigm where the agent learns in a single task from scratch, considering that even humans need thousands of hours to become an expert in a specific domain (Gladwell, 2008), arguably corresponding to millions of interacting steps. However, for many control tasks in industrial scenarios such as inventory management (Ding et al., 2022), quantitative trading (Zhang et al., 2023), and HVAC control (Zhang et al., 2022a), it is preferable to develop high-performance controllers for different tasks with low technical debt (Agarwal et al., 2016), which poses grand challenges for traditional control methods. For example, we may want to control HVAC for different buildings with a minimal amount of tuning and limited number of demonstrations for reference. Although the basic principle of HVAC control may be similar across these different tasks, the transition dynamics and even the state/action spaces may be different (e.g., depending on the specific buildings). Moreover, the demonstrations are typically insufficient to train an RL agent from scratch. Therefore, it is hard to develop a unified agent using RL or other traditional control methods for this scenario.

One promising approach is to leverage prior knowledge from foundation models. They are pre-trained on Internet-scale and diverse datasets, and thus can serve as a rich source of prior knowledge for the various industrial control tasks. The examples of foundation models are GPT-4 (OpenAI, 2023; Bubeck et al., 2023), Bard (Pichai, 2023), DALL-E (Ramesh et al., 2021), and CLIP (Radford et al., 2021), which demonstrate powerful emergent abilities and fast adaptation to various downstream tasks. The former two are representatives of large language models (LLMs)

and the latter two can deal with both text and images. We will explore industrial use of other foundation models in our future work and only focus on leveraging LLMs in this paper.

Inspired by the recent success of foundation models, there has been a growing number of decision making methods that leverage LLMs in various ways. These methods can be broadly divided into three categories: fine-tuning LLMs on specific downstream tasks, combining LLMs with trainable components, and using pre-trained LLMs directly. The first category of methods fine-tune pre-trained LLMs with the self-supervised loss on domain specific corpus (Bakhtin et al., 2022) or the RL loss to leverage task feedback (Carta et al., 2023). These methods can achieve better performance on specific tasks through fine-tuning. However, though parameter-efficient fine-tuning methods exist (e.g., Hu et al., 2021), it is generally costly to fine-tune LLMs with billions of parameters or may be impossible if LLMs are API-based. The second category of methods modify the output of the LLM with trainable value/feasibility/affordance/safty functions (e.g., Ahn et al., 2022; Yao et al., 2023; Huang et al., 2022) or use the LLM as a component of the trainable decision making system such as task explaining, reasoning, planning or serving as the world model (e.g., Hao et al., 2023; Wang et al., 2023b; Zhu et al., 2023; Yuan et al., 2023). These methods avoid the fine-tuning process which is sometimes inaccessible while preserving the ability of continual learning on domain-specific data. However, they need more human effort in designing complex mechanisms to incorporate LLMs into the decision making system and more budget to train learnable components which may be sample inefficient (e.g., learning a value function using RL). The third category of methods use pre-trained LLMs directly following the in-context learning (ICL) paradigm, and researchers focus on developing prompting techniques (Liang et al., 2022) or designing multi-turn mechanisms (Wang et al., 2023a; Shinn et al., 2023; Yao et al., 2022) to improve the performance. These methods are more light-weighted requiring less technical debt (e.g., designing the decision making system involving data collecting, training, etc.), but sacrifice the ability to further improve when sufficient interaction data is available. It is worth noting that there are other ways to incorporate LLMs only in the training process (e.g., Kwon et al., 2023; Du et al., 2023) which are omitted here since they play only subordinate roles in decision making.

Different from many previous works on control with foundation models that conduct experiments on robotic manipulation (James et al., 2020; Yu et al., 2020b), home assistants (Szot et al., 2021; Kant et al., 2022), or game environments (Chevalier-Boisvert et al., 2018; Fan et al., 2022), we focus on industrial control tasks which present the following three challenges for traditional RL methods: 1) The decision making agent typically faces a series of heterogeneous tasks (e.g., with different state and action spaces or transition dynamics). RL methods need to train separate models for heterogeneous tasks, which is costly. 2) The decision making agent needs to be developed with low technical debt, indicating that the provided samples are insufficient (if any) compared with the big data required for typical RL algorithms and that designing task-specific models may be impossible. 3) The decision making agent should adapt fast to new scenarios or changing dynamics in an online fashion (e.g., only based on few online interacting experiences but without training). To face these challenges, we propose to control HVAC using pre-trained LLMs directly. This method can solve for heterogeneous tasks with few samples since we do not involve any training process and only use samples as the few-shot demonstrations for in-context learning.

In this paper, we aim to research on the potential of making decisions for industrial control tasks *directly* using pre-trained LLMs. Specifically, we first design a mechanism to select demonstrations from both expert demonstrations and historical interactions, and a prompt generator to transform the objective, the instruction, the demonstrations, and the current state to prompt. Later, we execute the control given by the LLMs using the generated prompt. We aim to study how different designs influence the performance of applying LLMs to industrial control tasks since many aspects remain elusive. First, although this method is conceptually simple, its performance compared with traditional decision making methods is unclear. Second, the generalization ability of foundation

models to different tasks (e.g., with different contexts, action spaces, etc.) is also under-studied. Third, the sensitivity of this method to different designs of the language wrapper is also worth studying (e.g., which part of the prompt impact the performance most). By answering these questions, we aim to highlight the potential of such methods and shed light on how to design workarounds for industrial control tasks with low technical debt.

The contributions of this technical report are summarized as follows:

- We develop a training-free method to use foundation models for industrial control, which works across heterogeneous tasks with low technical debt.
- We take controlling HVAC with GPT-4 as an example and obtain positive experiment results, indicating the potential of such methods.
- We provide extended ablation studies (on the generalization ability, demonstration selection, and prompt design) to shed light on the future research for this direction.

2. Related Work

2.1. **Foundation Models.** Foundation models are pre-trained on large-scale data and serve as the foundation for various downstream tasks with different data modalities. The key structure behind these foundation models is Transformer (Vaswani et al., 2017), a neural network structure that relies on the attention mechanism to learn dependencies between input and output sequences. Incorporated with proper pretraining tasks on Internet-scale text and image data, foundation models can be used to learn from different data modalities including text, image, graph, speech, and others (cf. Zhou et al., 2023). Large language models (LLMs) dealing with text and vision-language models (VLMs) dealing with both images and text are two notable categories of foundation models, the examples of which include BERT (Devlin et al., 2018), ChatGPT (OpenAI, 2022; Ouyang et al., 2022), DELL-E (Ramesh et al., 2021), and GPT-4 (OpenAI, 2023; Bubeck et al., 2023). In this technical report, we mainly focus on utilizing LLMs.

While an increasing number of LLMs are emerging, researchers try to benchmark the performance of different models. Zhong et al. (2023) and Beeching et al. (2023) evaluate the models on a series of benchmark tasks, but they only evaluate a limited number of models: The former one only evaluate the GPT series from OpenAI, and the latter one only evaluate open-sourced LLM models. Chatbot Arena (Zheng et al., 2023) compares the models pairwise using the Elo rating system and covers the most first-tier LLMs. GPT-4 is on the top of their leaderboard, which motivates us to use it as a representative of the best foundation models in our experiments.

Broadly speaking, there are two approaches to utilize LLMs on specific tasks: fine-tuning and in-context learning (ICL). While fine-tuning adjusts the model by updating its parameters, ICL tries to design prompts, demonstrations, and queries to elicit good responses from LLMs without changing its parameters. For fine-tuning, researchers focus on how to conduct efficiently (such as Houlsby et al., 2019; Li and Liang, 2021; Lester et al., 2021; Hu et al., 2021) since it is costly to perform full-parameter update for LLMs with extensive number of parameters. However, these methods require the LLMs to be open-source, e.g., LLaMA (Touvron et al., 2023a), Llama-2 (Touvron et al., 2023b), and FLAN (Chung et al., 2022). Nevertheless, today's closed-source LLMs such as GPT-4 (OpenAI, 2023), Claude 2 (Antropic, 2023) and Bard (Pichai, 2023) generally perform better than their open-source counterparts with a notable capability gap (Gudibande et al., 2023), and such Language-Model-as-a-Service (LMaaS) paradigm (Sun et al., 2022) may be a future trend. Therefore, for these closed-source LLMs, ICL (Dong et al., 2022) becomes a better option to leverage them on specific tasks.

Moreover, ICL is suitable for our scenario where controllers are required be developed with low technical debt due to the following two reasons: 1) It is easy to incorporate expert knowledge into LLMs by changing the demonstration and templates since they are written in natural language. 2) ICL is training-free which reduces the computation costs and makes it easy to quickly adapt

the model to real-work tasks. Despite being promising, the performance of ICL is sensitive to the design of prompt and the selection or even the order of demonstrations empirically (see e.g., Zhao et al., 2021; Min et al., 2022), and it is not clear how these designs impact the performance of ICL for industrial applications.

2.2. **Foundation Models for Decision Making.** Pre-trained with large corpus, foundations models possess rich prior knowledge in various domains which is valuable for us to build generalizable and adaptive decision making agents.

Although there is a trend to train generalist agent with massive behavior datasets following the success of large language models, they are currently far below the critical scale (including the scale of model, datasets, and computation) for emergent abilities (Wei et al., 2022). Taking the scale of model as an example, GPT-3 (Ouyang et al., 2022) has 175B parameters and most LLMs have more than 10B parameters (Zhao et al., 2023). In contrast, recent generalist agents have much fewer parameters, e.g., Gato (Reed et al., 2022) (1.2B parameters), UniPi (Dai et al., 2023) (∼10B parameters), and VPT (Baker et al., 2022) (0.5B parameters).

Therefore, a more viable way to leverage the prior knowledge from LLMs is to interact with off-the-shelf LLMs pre-trained on text corpora. Recent papers on this direction focus on designing mechanisms to address the executability and correctness issues when using LLMs as the controller or planner. For example, the output of LLMs can be corrected by additional value functions (Yao et al., 2023; Ahn et al., 2022) or senmatic translation (Huang et al., 2022); the generation process of LLMs can be decomposed to multiple modules or steps (Wang et al., 2023b; Zhu et al., 2023; Shinn et al., 2023; Yao et al., 2022). However, we find that it is still possible to elicit executable and correct actions from the LLM directly by developing proper ICL techniques on practical scenarios. Moreover, studying the capability of LLMs in direct control should be an indispensable step for us to understand how the LLM work and how to format the tasks in a way the LLM can follow.

2.3. **HVAC Control.** In this paper, we focus on the HVAC control for the building with the aim to save energy as well as keep thermal comfort. HVAC control has been studied over a long time and is representative of a wide range of industrial control problems (Belic et al., 2015; Afroz et al., 2018). Previous methods for controlling HVAC can be broadly divided into three categories: classical approaches, predictive control methods, and intelligent control techniques. One representative example of classical approaches is the PID (proportional-integral-derivative) controller (Tashtoush et al., 2005; Wang et al., 2008; Liu et al., 2009) whose performance degrades if the operating conditions vary from the conditions for parameter tuning. Predictive control methods (also known as model predictive control, MPC) usually performs better by predicting the dynamic behavior of the system in the future and adjusting response of controller accordingly. There is a large body of research on MPC (e.g., Ma et al., 2009, 2011, 2012) and we refer interested readers to the survey (Afram and Janabi-Sharifi, 2014). While predictive control relies on correct modeling of the physical environment, intelligent control techniques can be more robust and adaptive to different conditions. The examples of intelligent control include fuzzy logic control (Villar et al., 2009; Al-Ali et al., 2012), genetic-algorithm-based methods (Alcalá et al., 2003; Khan et al., 2013), deep-learning-based methods (Mirinejad et al., 2012), and reinforcement-learning-based methods (Wei et al., 2017; Azuatalam et al., 2020; Yu et al., 2020a). However, these methods require high technical debt (e.g., the effort to model the problem, developing algorithms, collecting samples, and inquiring expert knowledge), thus being incompatible with the demands of swift development in the modern industrial scenarios.

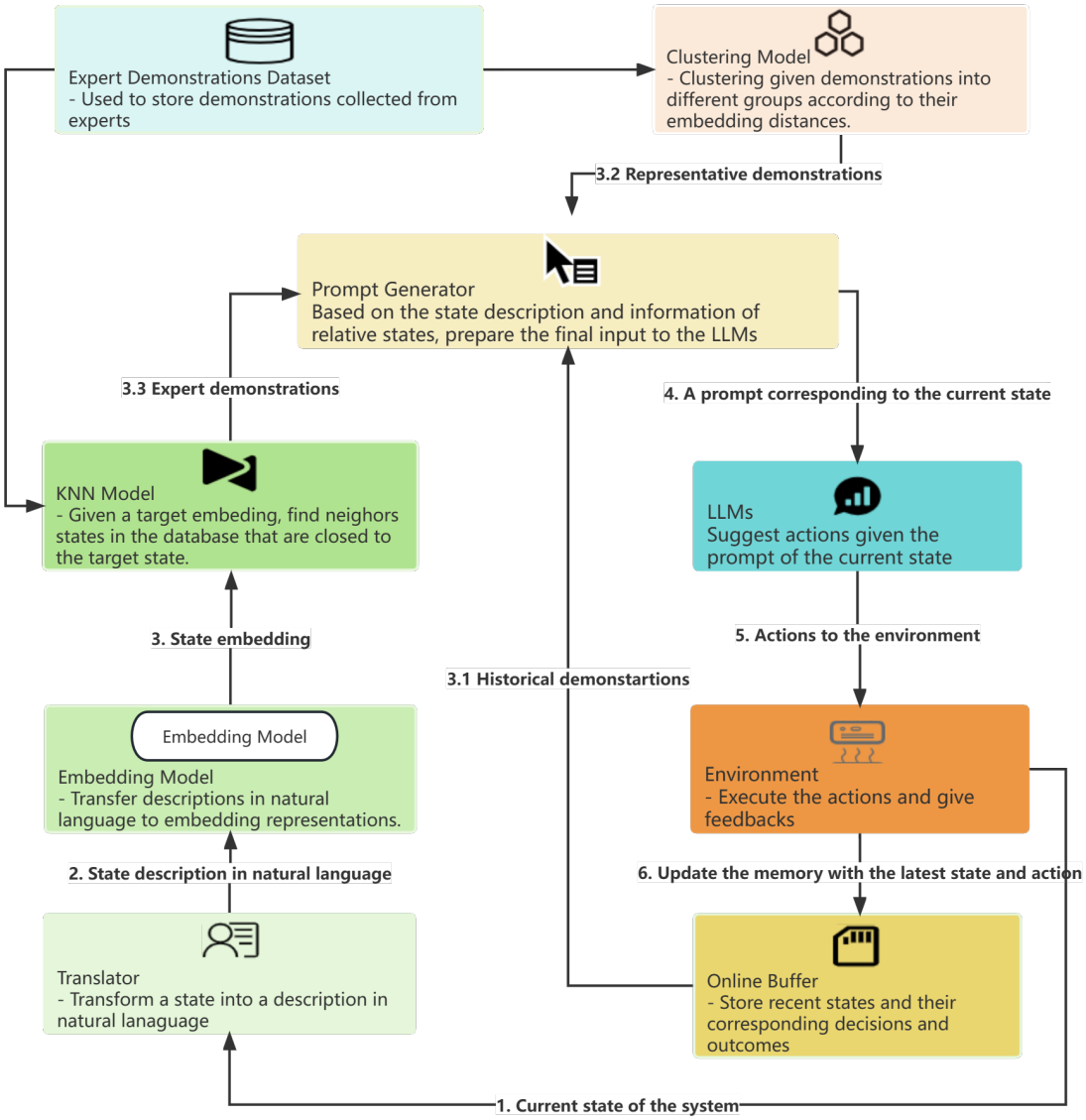


FIGURE 1. The pipeline illustrating how GPT-4 is utilized to control HVACs.

3. METHOD

In this section, we provide a detailed explanation of how we utilize GPT-4 to optimize the control of HVAC devices. The entire pipeline is depicted in Figure 1, which consists of the following components:

LLM. A pre-trained large language model is employed as the decision-maker. Given a prompt, it generates corresponding actions. The prompt includes a description of the current state, simple HVAC control instructions, demonstrations of relevant states, and more. Further details on the prompt design will be discussed in the subsequent sections.

Environment. The interactive environment or simulator enables the execution of actions suggested by the LLM and provides feedback. In our experiment, we use BEAR (Zhang et al., 2022a) as our evaluation environment. To create an environment in BEAR, two parameters must be supplied: building type (such as large office, small office, hospital, etc.) and weather condition (such as hot and dry, hot and humid, warm and dry, and so on). Additionally, it is worth noting that each weather condition corresponds to a specific city. For example, the hot and dry weather condition is associated with Buffalo.

In BEAR, each state is represented by a numeric vector where every dimension corresponds to the current temperature of a room in the building, except for the last four dimensions. These final four dimensions represent outside temperature, global horizontal irradiance, ground temperature, and occupant power, respectively. In all environments, the primary objective is to maintain room temperatures close to 22 degrees Celsius while minimizing energy consumption as much as possible.

Actions in BEAR are encoded as real numbers ranging from -1 to 1. Negative values signify cooling mode, whereas positive values represent heating mode. The absolute values of these actions correspond to valve openings, which in turn indicate energy consumption. As the absolute values increase, energy consumption also rises. Considering both comfort and energy consumption, we employ the following reward function in all experiments:

$$\left(1.0 - \frac{\sum_{0 \leq i < n} |a_i|}{n}\right) + \alpha \cdot \left(1 - \frac{\sum_{0 \leq i < n} (t_i - T)^2}{T \cdot n}\right), \tag{1}$$

where n denotes the number of rooms, $T = 22°C$ is the target temperature, and t_i represents the temperature of the i-th room. The hyper-parameter α serves to balance the two aspects: energy consumption and comfort.

Online Buffer. We design an demonstration queue to store recent interactions between the LLM and its environment. This information is utilized by the prompt generator to create portions of prompts provided to the LLM.

Translator. In BEAR environments, original states are represented as vectors of real numbers, making them challenging for the LLM to handle directly, which we will illustrate shortly in our experiment section. To overcome this issue, we introduce the translator component, which converts a numeric state into a natural language representation while retaining all relevant information. In our approach, we distinguish the following translator:

- metaTranslator. The environment translator is utilized to extract meta information related to the building type and weather condition in which the HVAC being controlled is located. Below shows an example:

```
You are the HVAC administrator responsible for managing a building of type Office
    ↪ Medium located in Buffalo, where the climate is Hot and Dry.
```

- instructionTranslator. The instruction translator operates in two modes depending on the outside temperature. When the outside temperature is lower than the target temperature, it provides instructions related to the heating mode; otherwise, it switches to the cooling mode. The following example demonstrates instructions associated with the heating mode.

```
Currently, outside temperature is lower than the target temperature.
To optimize HVAC control, adhere to the following guidelines:
1. Actions should be represented as a list, with each integer value ranging from 0
    ↪  to 100.
2. The length of the actions list should correspond to the number of rooms,
    ↪ arranged in the same order.
```

```
3. If room temperature is higher than the target temperature, the larger the
    ↪ difference between room temperature and the target temperature, the lower
    ↪ the action should be.
4. If room temperature is lower than the target temperature, the larger the
    ↪ difference between room temperature and the target temperature, the higher
    ↪ the action should be.
```

- stateTranslator. The present state translator accepts the existing numerical state vector as input and converts it into a natural language representation. For instance, when provided with a state that describes a building comprising four rooms with temperatures of 21, 20, 23, and 19 degrees Celsius respectively, we can generate the subsequent description. In addition to the room temperatures, we detail the last four dimensions in text, accompanied by an extra line that underscores the target temperature. In order to enable the LLM to more effectively comprehend numerical values, we round all real numbers to their nearest integer values. This introduces a rounding error. We strike a balance between the rounding error and the comprehension of the LLM. This serves as a technique to manipulate LLMs. Experimental findings will demonstrate that this can significantly enhance the performance of GPT-4.

```
The building has 4 rooms in total.
Currently, temperature in each room is as follows:
   Room 1: 21 degrees Celsius
   Room 2: 20 degrees Celsius
   Room 3: 23 degrees Celsius
   Room 4: 19 degrees Celsius
The external climate conditions are as follows:
   Outside Temperature: -17 degrees Celsius.
   Global Horizontal Irradiance: 0
   Ground Temperature: 0 degrees Celsius
   Occupant Power: 0 KW
   Target Temperature: 22 degrees Celsius
```

- actionTranslator. The action translator converts original actions into integers ranging from -100 to 100. Similar to the state translator, this transformation facilitates a better understanding of numerical actions by the LLM. Below presents an example where the original actions are $[0.95, 0.9, 0.72, 0.68]$.

```
Actions: [95, 90, 72, 68]
```

- feedbackTranslator. To enhance the decision-making process of the LLM, we introduce a feedback translator that converts outcomes (rewards and next states) from the environment into meaningful natural language comments. This enables the LLM to assess the performance of given examples, allowing it to learn not only from successful controls but also from unfavorable ones. The example below illustrates a scenario where the first line represents the step reward (rounded to the nearest integer after multiplying by 10) achieved by the action. Subsequent lines describe the room temperatures after executing the action, accompanied by comments that indicate how these temperatures compare to the target temperature.

```
Reward: 8
Actions: [90, 92, 76, 97]
Comments: After taking the above actions, temperature in each room becomes:
   Room 1: 23 degree Celsius
```

```
    Room 2: 22 degree Celsius
    Room 3: 20 degree Celsius
    Room 4: 24 degree Celsius
The action for Room 1 shall be decreased as its temperature is higher than the
    ↪ target temperature.
The action for Room 3 shall be increased as its temperature is lower than the
    ↪ target temperature.
The action for Room 4 shall be decreased as its temperature is higher than the
    ↪ target temperature.
```

Embedding Model. The embedding model serves to convert a natural language representation into an embedding while preserving semantics as much as possible. These embeddings are employed as keys for storing and retrieving original states, along with their associated actions and outcomes. In our experiment, we utilized the Universal Sentence Encoder (Cer et al., 2018) as the foundation for our embedding model, whose embedding size is 512.

Expert demonstrations Dataset. The expert demonstrations dataset comprises tuples gathered from expert policies. Demonstrations within the dataset may be derived from buildings and weather conditions that differ from the one being controlled. This approach aims to encourage the LLM to learn the underlying principles of HVAC controls rather than merely replicating expert behaviors. In our experiment, we pre-trained a Proximal Policy Optimization (PPO) (Schulman et al., 2017) policy for each environment and subsequently executed the trained PPO policy as an expert policy for 100,000 steps to collect expert demonstrations.

KNN Model. The k-Nearest Neighbors (KNN) model aims to identify a specific number of similar states within the expert demonstrations dataset. We employed the "NearestNeighbors" algorithm from the scikit-learn library (Buitinck et al., 2013) as the foundational model. As mentioned earlier, the keys employed to retrieve similar states are derived from the embedding model, which is achieved by concatenating outputs from both the metaTranslator and stateTranslator.

Clustering Model. The clustering model aims to identify a specific number of distinct states within the expert demonstrations dataset. We employed the "K-means" algorithm from the scikit-learn library (Buitinck et al., 2013) to conduct clustering. Similarly to the KNN model, embedding representations of demonstrations are utilized as inputs.

Prompt Generator. Finally, we elucidate how prompts are generated in our methodology, incorporating all the aforementioned components. In Figure 2 we illustrate the whole process to generate a prompt in our approach, where text in purple is only for illustration and not a part of the prompt.

4. Experiments

In this section, we present experimental results that highlight the effectiveness of GPT-4 in controlling HVAC devices across a range of buildings and weather conditions. By providing suitable instructions and demonstrations (not necessarily associated with the target building and weather condition), GPT-4 can surpass the performance of a meticulously trained RL policy tailored for the particular building and weather condition. Additionally, we conduct comprehensive ablation studies to determine the contributions of each element within the prompt.

4.1. Baselines.

In our experiments, we evaluate two baseline approaches: the Model-Predictive-Control (MPC) method and the PPO method.

- **MPC.** Model Predictive Control (MPC) is a control strategy that optimizes the control inputs to a system by solving an optimization problem at each time step. The approach relies on a predictive model of the system, which is used to forecast the system's behavior over a finite horizon. At each time step, the optimization problem minimizes a cost function

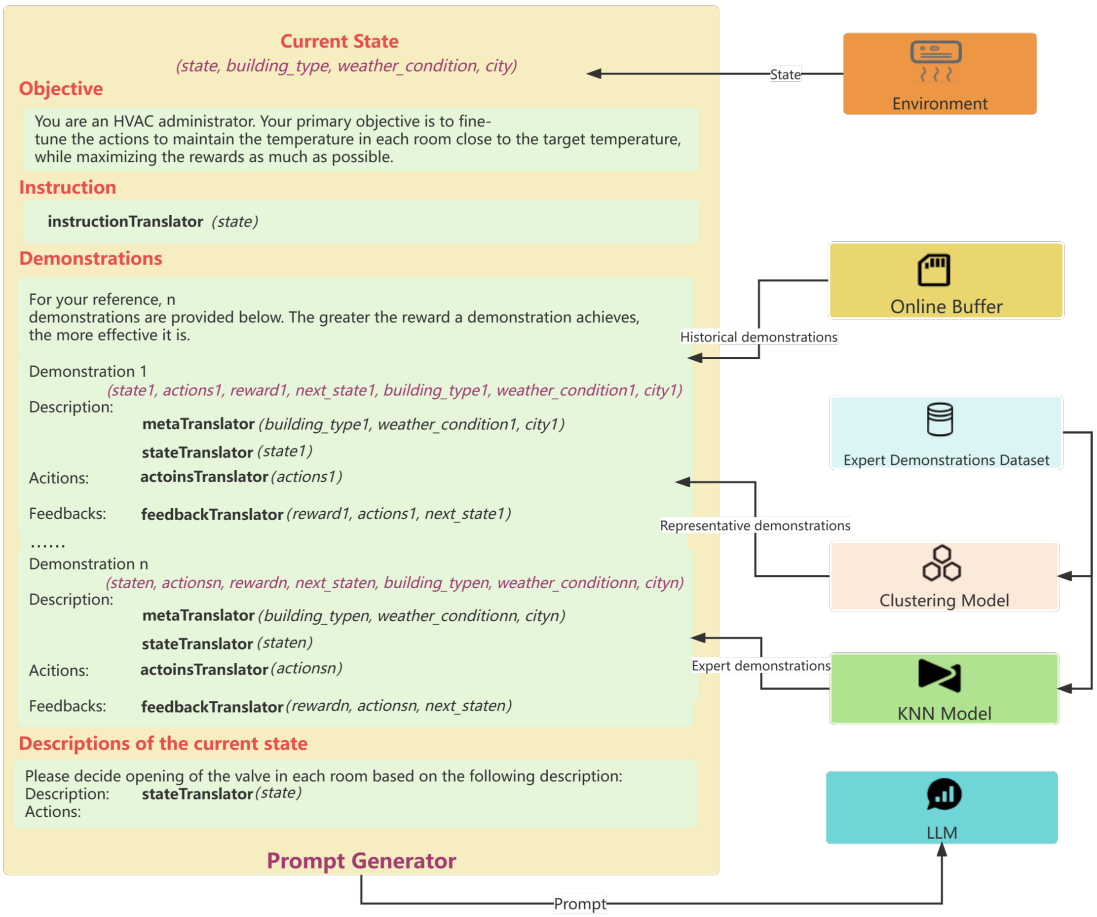


FIGURE 2. How prompts are generated in our approach.

that is designed to penalize deviations from a desired reference trajectory while considering constraints on the control outputs and system states. The first control input from the optimal solution is applied to the system, and the process is repeated at the next time step. MPC is widely used in various applications, including robotics, automotive control, and process control, due to its ability to handle constraints, predict system behavior, and optimize control inputs in a systematic manner (Rawlings et al., 2017). In our experiment, the MPC approach is utilized as an oracle/skyline. In other words, rather than relying on a predictive model, we allow the MPC method to access the ground truth of external temperatures for the subsequent 10 steps. The results obtained through the MPC approach can be considered as an upper bound for all algorithms.

- **PPO.** Proximal Policy Optimization (PPO) is a popular algorithm in reinforcement learning, designed to improve the stability and performance of policy gradient methods. PPO was introduced by Schulman et al. (2017) as a computationally efficient alternative to Trust Region Policy Optimization (TRPO) (Schulman et al., 2015). PPO is an on-policy method that allows us to optimize the policy while maintaining the trust region constraint, preventing the policy from updating too drastically to guarantee policy improvement. The

main idea behind PPO is to use a surrogate objective function that consists of an importance sampling ratio. The ratio is clipped to prevent too large policy updates, ensuring that the new policy does not deviate too far from the old one. The objective function for PPO is given by:

$$L^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right]$$

Here, θ represents the policy parameters, and $r_t(\theta)$ is the probability ratio between the new policy π_θ and the old policy $\pi_{\theta_{\text{old}}}$, defined as:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}.$$

$\hat{A}_t$ denotes the estimated advantage for the sample collected at the t-th time step, and the clip ratio ϵ is a hyperparameter. The clip function limits the value of $r_t(\theta)$ to the range $[1-\epsilon, 1+\epsilon]$. The PPO algorithm has been shown to achieve stable and efficient learning in a wide range of reinforcement learning tasks and is widely used in practice due to its simplicity, ease of implementation, and good performance.

For all experimental results, we operate the corresponding environment under the guidance of a given policy for 240 steps, corresponding to a 10-day execution period. We run each scenario for five rounds using different seeds. In the following sections, we report the mean rewards and their standard deviations.

4.2. **Experiment setting.** In our experiments, we selected the building "OfficeMedium" and the "CoolDry" weather condition as our target scenario. This is representative of the climate found in International Falls, Minnesota.

To gather expert demonstrations, we train PPO models for various scenarios separately within BEAR (Zhang et al., 2022a). Each scenario is specified by the combination of the building type (OfficeSmall, OfficeMedium, or OfficeLarge) and the weather type (ColdDry, CoolDry, WarmDry, or MixedDry). For each scenario, we train a PPO model through 100 million steps and execute the trained policy to collect 20,000 transitions, which later serve as candidate expert demonstrations datasets.

In addition to expert demonstrations, we introduce two other types of demonstrations to assess their impact on the performance of GPT. In summary, we employ the following three types of demonstrations in our experiments.

- Historical Demonstrations: These are demonstrations derived from previous interactions between GPT-4 and the current environment under evaluation.
- Representative Demonstrations: To identify representative demonstrations, we employ the K-means clustering algorithm to group all expert demonstrations. Representative demonstrations are then selected as those that are closest to the center of each cluster. It is important to note that unlike the other two types of demonstrations, representative demonstrations remain constant across all time steps and are intended to be diverse so that GPT-4 can learn to make decisions in various situations.
- Expert Demonstrations: These are demonstrations collected from buildings and weather conditions, whose configurations depend on our experiment settings, which we will illustrate in details later.

While their names may seem similar, the scenarios considered in our experiments are quite diverse. In Figure 3 and 4, we demonstrate that distinct buildings correspond to unique expert policies even under identical weather conditions (using the first room of each building as an example). Additionally, the same room in the OfficeMedium building exhibits varying expert policies across different weather conditions. This confirms that the demonstrations gathered from various

buildings and weather conditions are sufficiently diverse, and an expert demonstration collected from one scenario may not necessarily be a good demonstration for the target scenario. This further necessitates the reasoning capacity of LLMs in order to effectively deduce HVAC control logic from provided demonstrations rather than merely imitating them.

4.3. **How well can GPT-4 control HVACs?** In order to evaluate GPT-4's performance in HVAC control, we devise six groups of experiments with similar settings, distinguished by their access to different demonstration datasets. Recall that our target scenario is "OfficeMedium" with "CoolDry". In details,

- Group A: demonstrations are limited to those gathered from environments where the building is either OfficeSmall or OfficeLarge, and the weather condition is chosen from ColdDry, WarmDry, or MixedDry. This is designed to be the most challenging experiment for GPT-4 since the demonstrations dataset does not include any sample from the same building or weather condition as the target scenario.
- Group B: In addition to demonstrations utilized in Group A, we also incorporate demonstrations gathered from the OfficeMedium building under ColdDry, WarmDry, or MixedDry weather conditions. Compared to Group A, this experiment is less challenging, as it includes demonstrations from the same building, albeit with varying weather conditions.
- Group C: In addition to the demonstrations employed in Group A, this group of experiments also has access to demonstrations collected from the OfficeSmall and OfficeLarge buildings under the CoolDry weather condition. In other words, we can access demonstrations from the same weather condition as the target one, but in different buildings.
- Group D: This group of experiments has access to the most extensive range of demonstrations compared to others. Specifically, we collect demonstrations from OfficeSmall, OfficeMedium, and OfficeLarge buildings in ColdDry, CoolDry, WarmDry, and MixedDry weather conditions.
- Group E: This group of experiments utilizes the most pertinent data gathered from the same building and weather conditions as those of the target study. Specifically, we only collect data from the OfficeMedium under the CoolDry weather condition.
- Group F: In this group of experiments, we solely focus on demonstrations derived from past interactions between GPT-4 and the target environment, excluding any pre-gathered demonstrations.

Remember that we differentiate between three types of demonstrations used in prompts given to GPT-4. For experiments in Groups A-E, we provide GPT-4 with the following demonstrations in sequence:

- Two historical demonstrations: These demonstrations correspond to the most recent two interactions between GPT-4 and the target environment.
- Two representative demonstrations: The demonstrations datasets used in each group are first divided into two clusters, and representative demonstrations are then selected as those closest to the centers of these clusters, as explained in Section 4.2.
- Four expert demonstrations: Four expert demonstrations are chosen from the provided demonstrations datasets by employing the embedding and KNN model, as detailed in Section 3.

In the experiments conducted for Group F, GPT-4 is provided with only four historical demonstrations at each step, as it does not have access to any expert demonstrations.

From the aforementioned experiment groups, it is evident that our intention is to assess GPT-4's performance when provided with demonstrations of varying similarity to the target scenario. Through this approach, we aim to determine whether GPT-4 merely replicates demonstrations or can genuinely learn the principles of controlling HVAC devices from these demonstrations.

Algo.	Reward Mean	Reward Std.	Demo. Buildings	Demo. Weather	Demo.
GPT-A	1.16	0.04	OfficeSmall OfficeLarge	ColdDry WarmDry MixedDry	H2R2E4
GPT-B	1.18	0.04	OfficeSmall OfficeMedium OfficeLarge	ColdDry WarmDry MixedDry	H2R2E4
GPT-C	1.17	0.05	OfficeSmall OfficeLarge	CoolDry ColdDry WarmDry MixedDry	H2R2E4
GPT-D	1.20	0.02	OfficeSmall OfficeMedium OfficeLarge	CoolDry ColdDry WarmDry MixedDry	H2R2E4
GPT-E	1.09	0.02	OfficeMedium	CoolDry	H2R2E4
GPT-F	1.23	0.01	None	None	H4R0E0
GPT-Random	0.88	0.12	OfficeSmall OfficeMedium OfficeLarge	CoolDry ColdDry WarmDry MixedDry	-
MPC	1.35	0.00	-	-	-
PPO	1.21	0.04	-	-	-
Random	-26.72	0.65	-	-	-

TABLE 1. How GPT-4 performs given different sets of expert demonstrations.

We present the results of this set of experiments in Table 1, in which GPT-X corresponds to the outcomes obtained by GPT-4 for scenarios within Group X, where X can be A-F. We also use H, E, and R as abbreviations for historical, expert, and representative demonstrations, respectively, followed by a value indicating the number of corresponding demonstrations in the prompt. We observe that GPT-4 achieves comparable or better outcomes to PPO, even in scenarios where GPT-4 has access only to expert demonstrations collected from different buildings under varying weather conditions (GPT-A). This demonstrates GPT-4's ability to effectively learn the principles of controlling HVAC devices from demonstrations and instructions. However, it is important to note that when provided with demonstrations gathered from the same building or under the same weather conditions, GPT-4's performance can be further enhanced (GPT-A vs. GPT-D). Remarkably, without utilizing any pre-gathered demonstrations, GPT-4 can attain the best outcome solely through learning from interactions with the target environment. In the following section, we will investigate the performance of GPT-4 under diverse combinations of assorted demonstrations. For comparison, we also present the results of two random policies in Table 1. The "GPT-Random" policy involves executing a strategy similar to GPT-D, except that all eight demonstrations are randomly sampled from the demonstrations dataset at each step. Meanwhile, the "Random" policy entails randomly sampling actions from the action space at each step during execution. It is evident that the PPO, MPC, and GPT policies significantly outperform the random policies.

4.4. **How important are the demonstrations?** In this section, we evaluate the impact of demonstrations on the performance of GPT-4. Specifically, we examine how the performance of GPT-4 is affected by varying the types and number of demonstrations provided.

FIGURE 3. Different buildings correspond to different expert policies under the same weather condition.

Demonstrations	Reward Mean	Reward Std.
H0R0E0	1.01	0.08
H2R0E0	1.15	0.04
H4R0E0	1.23	0.01
H8R0E0	1.21	0.03
H0R2E0-A	0.85	0.17
H0R4E0-A	1.07	0.09
H0R8E0-A	0.9	0.05
H0R0E2-A	0.51	0.11
H0R0E4-A	0.86	0.04
H0R0E8-A	0.92	0.04
H2R4E2-A	1.19	0.02
H2R2E4-A	1.16	0.02
H4R2E2-A	1.2	0.01

TABLE 2. Performances of GPT-4 using different types and numbers of demonstrations.

The experiment results are presented in Table 2, where we append a suffix "-A" to indicate that the representation and expert demonstrations are derived from the demonstrations dataset, with configurations identical to those in GPT-A. Surprisingly, as can be seen from Table 2, historical

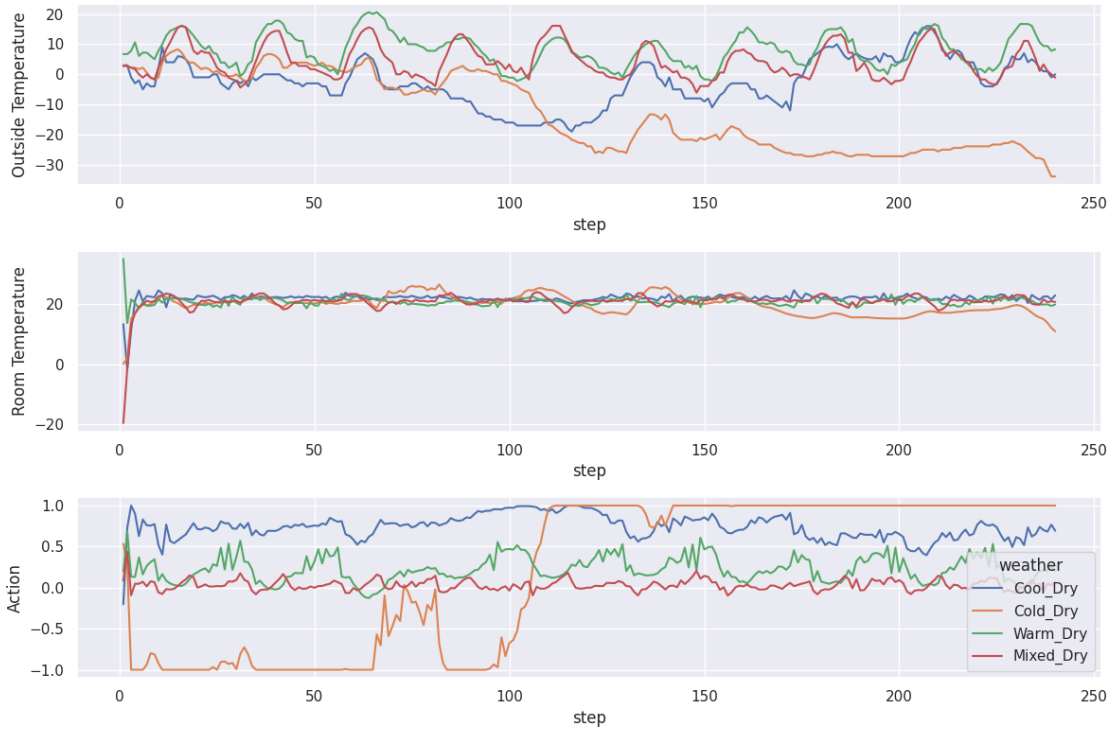


FIGURE 4. The same building has different expert policies under different weather conditions.

demonstrations are the most effective for GPT-4 decision-making. Expert demonstrations, on the other hand, consistently diminish the performance of GPT-4, even performing worse than the case with no demonstrations at all. Representative demonstrations may slightly improve GPT-4's performance, but only if an appropriate number of demonstrations are provided. These observations further verify the reasoning capability of GPT-4, as it can learn to reason not only from good demonstrations but also from flawed ones.

As illustrated in Figure 3 and 4, distinct buildings correspond to unique expert policies even under identical weather conditions, this indicates that expert demonstrations for one scenario are probably not expert demonstrations for the others. Hence, by providing these demonstrations to GPT-4 may mislead its decision-making, which explains the results in Table 2.

4.5. **How important if we add different types of comments?** Drawing inspiration from the research on imitation learning (see e.g., Brown et al., 2019, 2020; Cai et al., 2022), our aim is to enable GPT-4 to learn not only from well-crafted demonstrations but also from flawed ones. To achieve this, we incorporate a comment into each demonstration. Based on the method of generating these comments, we identify two distinct types:

- Manual: Comments are meticulously crafted in accordance with the feedbackTranslator outlined in Section 3.
- Self-comment: Comments are automatically generated by GPT-4, which involves appending the following instruction to the end of each prompt.

```
Before initiating actions, it is advisable to first offer feedback on the quality
    ↪ of all the provided demonstrations to further improve the comprehension of
    ↪ control logics derived from them.
```

In Table 3, we present the results of utilizing various types of comments in GPT-4 policies, while keeping all other configurations identical to GPT-A. The results clearly indicate that self-comments generally have a detrimental effect on performance, whereas manual comments significantly improve performance, increasing it from 0.99 to 1.16. We hypothesize that this may be due to the overly simplistic instruction used in our experiments. As future work, we will further improve the instruction for self-comment, such as asking GPT-4 to provide individual comments on each demonstration and incorporating these comments as part of the prompts, as in Shinn et al. (2023).

Comment Types	Reward Mean	Reward Std.
None	0.99	0.1
Manual	1.16	0.04
Self-comment	0.59	0.32
Manual and Self-comment	1.11	0.07

TABLE 3. Performances of GPT-4 using different types of comments[1].

4.6. **How important is the task description and instructions?** We conducted an ablation study to evaluate the importance of different parts of the task description and instructions on the performance of GPT-4. We distinguished the following types of texts:

- Task Description: We provide a general task description with metaTranslator introduced in Section 3.
- Task Instructions: We provide task instructions (i.e., Item 3 and 4 in instructionTranslator introduced in Section 3).

As demonstrations are crucial to the performance of GPT-4 policies, we aim to fairly evaluate the importance of task descriptions and instructions by removing all demonstrations from prompts in this group of experiments. We present the results in Table 4. We observe that task descriptions can significantly enhance GPT-4's performance compared to task instructions. While task instructions can slightly improve GPT-4's performance, they may degrade its performance when combined with task descriptions. This could be attributed to the fact that the complexity of HVAC control cannot be adequately summarized by the two instructions, which may potentially mislead GPT-4's behavior in certain cases. It is worth noting that, even without any demonstrations, GPT-4 can already perform remarkably well, achieving a mean reward of 1.12 by simply adhering to the task description and utilizing the domain knowledge embedded within its framework. This further underscores its remarkable reasoning capabilities.

4.7. **How important is it to round real values?** As mentioned in Section 3, we round all real numbers to their nearest integer values, based on the assumption that GPT-4 might have difficulty handling real numbers directly. In this section, we perform an ablation study to validate this assumption. The results are presented in Table 5, which demonstrates that using rounded real numbers can indeed enhance the performance of GPT-4, thereby confirming our hypothesis.

4.8. **How robust is GPT-4 policy to perturbations?** In control optimization, it is crucial for policies to be robust enough to accommodate a certain level of perturbations. To verify the

[1]It is worth mentioning that our comments differ from the concept of reflexion presented in Shinn et al. (2023). While reflexion directly evaluates an entire trajectory based on the final reward, we focus on providing comments for demonstrations collected on a step-by-step basis.

Description Types	Reward Mean	Reward Std.
None	1.01	0.08
Instruction Only	1.04	0.01
Description Only	1.12	0.04
Description and Instruction	1.07	0.03

TABLE 4. Performances of GPT-4 using different types of description and instructions.

Rounded	Reward Mean	Reward Std.
True	1.16	0.04
False	1.07	0.07

TABLE 5. Performances of GPT-4 depending on whether real numbers are rounded in prompts.

Algo.	Reward Mean	Reward Std.
GPT-A	1.16	0.04
GPT-noise2	1.15	0.04
PPO	1.21	0.04
PPO-noise2	1.07	0.08

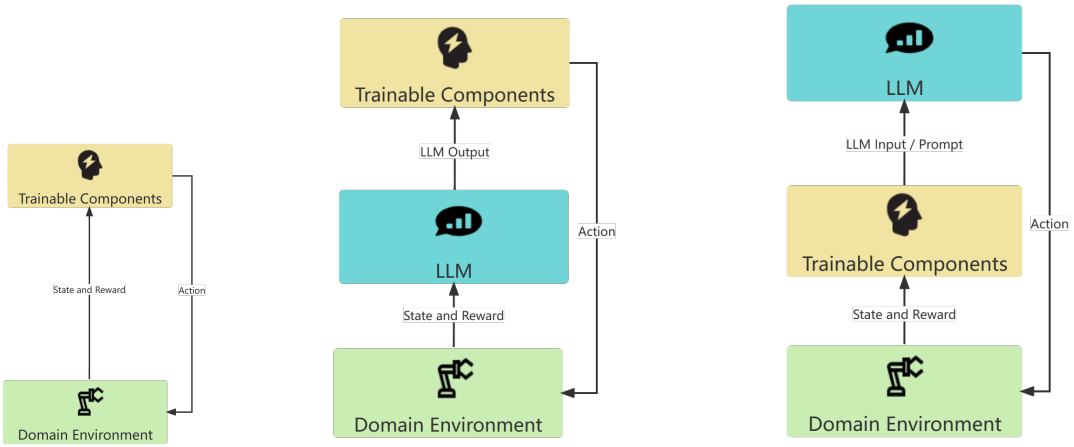
TABLE 6. Performances of PPO and GPT under weather perturbations.

robustness of GPT-4 policies, we conduct experiments by introducing noise to the external temperature. Specifically, at each step, we sample noise from a normal distribution with a mean of 0 and a standard deviation of 2, and then add it to the original outside temperature. In Table 6, we present all the results, which also include PPO results for comparison. For reference, we also include results from previous sections without perturbations. It is evident that GPT-4 policies can maintain good performance in the presence of perturbations without a significant decrease in performance.

5. Future Work

In previous sections, we demonstrated the potential of utilizing LLMs to facilitate decision-making. However, such an approach heavily depends on the reasoning capabilities of LLMs based on demonstrations and instructions. Unlike RL algorithms, which can consistently improve themselves through interactions with their environments, leading to enhanced performance over time, LLMs lack self-learning capabilities. On the other hand, while being built upon a single foundation model, LLMs can accumulate knowledge across various tasks and possess the potential for lifelong learning. In contrast, traditional RL algorithms can learn to improve over specific tasks, but their acquired knowledge rarely generalizes across different tasks and domains.

We posit that a crucial challenge in applying LLMs to decision-making lies in enabling them to learn autonomously. A significant amount of work has been dedicated to this topic. To better situate our approach, we categorize existing works based on the following dimensions: 1) Whether a pre-trained LLM is leveraged; 2) Whether the model can learn continually; 3) Whether the LLM is updated; and 4) Whether the upstream (e.g., the prompt) or the downstream (e.g., a value function) is updated. To obtain an overall view of these approaches on their advantages and



(A) Standard RL (B) Trainable Downstream Components (C) Trainable Upstream Components

FIGURE 5. Three paradigms to achieve learning capability without fine-tuning LLMs directly.

drawbacks, we compare these approaches across various important aspects for decision making. We mainly focus on the following aspects:

- **DQ Demand** (Data Quality Demand). This aspect considers data quality each approach requires to achieve good performance and the challenges associated with collecting such data.
- **Data Efficiency**. This aspect examines how efficiently each approach utilizes data, focusing on whether they can achieve strong performance through zero-shot or few-shot learning.
- **Online Learning**. This criterion assesses whether an approach possesses learning capabilities, specifically, its ability to adapt quickly to recent changes. Using HVAC control as an example, weather conditions can change unexpectedly, and an ideal policy should be able to rapidly adapt to these changes without significant delays.
- **Generalization/Transfer Learning**. This factor evaluates the extent to which an approach can be smoothly applied across different scenarios. For example, in the HVAC control context, we expect a policy to effectively generalize to various buildings and weather conditions without significantly compromising performance.
- **Performance**. This measure compares the effectiveness of each approach in different scenarios relative to existing algorithms.
- **Interpretability**. This element examines the level of interpretability achievable by each approach. For instance, by providing LLMs with appropriate prompts, the LLM-based approach can produce not only final actions but also the reasoning process, making the results more comprehensible to humans. This is not the case for approaches that incorporate deep neural networks, as their outputs are typically difficult to interpret.
- **LLMs Accessibility**. This consideration explores whether an approach requires the access to the weights or only the API of the LLM.

We present the results in Table 7 where more "+"s indicate a better capability on the corresponding aspect. Let us inspect the results row by row.

Methods	Training Algo.	DQ Demand	Data Efficiency	Online Learning	Generalization	Performance	Interpretability	LLMs accessibility
Control w/o LLMs	/	++	+	+++	+	+++	+	/
Learn in-context	/	+	+++	++	++	+	+++	+++
Fine-tune LLMs	Supervised	+	++	+	++	++	+++	+
	RL	++	+	++	++	+++	+++	+
Train upstream	Supervised	+	++	+	+++	++	+++	+++
	RL	++	+	+++	+++	+++	+++	+++
Train downstream	Supervised	+	++	+	++	++	++	+++
	RL	++	+	+++	++	+++	++	+++

TABLE 7. Comparison of different approaches of achieving learning capability in decision-making problems.

- **Traditional decision making.** Traditional decision making approaches such as dynamic programming (Bellman, 1966), model predictive control (Rawlings, 2000), and reinforcement learning (Sutton and Barto, 2018) do not rely on a pre-trained LLM and train the model (e.g., the value function, the world model, or the policy) with a large number of samples due to the iterative updates within these algorithms which are sample-inefficient (Botvinick et al., 2019). Moreover, they can hardly generalize to other tasks since the models are trained typically for a specific task. They are also not interpretable since there is a gap between the model itself and the control logic. One worth-mentioning stream of methods is the (generalist) RL agents based on the Transformer architecture which serves as the building block of LLMs. The representatives include DecisionTransformer (Chen et al., 2021), Gato (Reed et al., 2022), and PaLM-E (Driess et al., 2023). They can generalize to a wider but still limited range of tasks and are not interpretable.
- **In-context learning.** A direct way of utilizing pre-trained LLMs is to leverage the emergent in-context learning (ICL) ability (Dong et al., 2022). This category of methods usually rely on intuitive prompting strategies and let the LLM to generate decisions directly (e.g., Shinn et al., 2023; Yao et al., 2022). Although ICL sometimes works in the zero-shot setting, the performance of these methods usually depends on the quality of few-shot demonstrations (Liu et al., 2021), therefore with high data efficiency but also high demand on data quality. Moreover, these methods can leverage most of the available LLMs and are interpretable/generalizable since they only rely on API-based LLMs, enable generating natural language explanation, and adapt to tasks with similar control logic but different MDP formulations.
- **LLM fine-tuning.** One drawback of in-context learning is that its performance is limited by the capability of pre-trained LLMs. To enhance the capability of the LLM in accomplishing the specific task, researchers fine-tune LLMs on the specific domain via supervised learning (e.g., Bakhtin et al., 2022) or reinforcement learning (e.g., Carta et al., 2023). However, fine-tuning LLMs requires the access to the weights of the LLM, thus being unable to utilize API-based LLMs.
- **Train upstream/downstream modules of LLMs.** Although parameter-efficient fine-tuning methods exist (Houlsby et al., 2019; Li and Liang, 2021; Lester et al., 2021; Hu et al., 2021), these methods are still computationally costly. Therefore, a more amiable way may be design smaller trainable modules to serve as the upstream/downstream of LLMs. Moreover, combined with trainable modules, these methods are more capable of learning continually, i.e., becoming even better when interacting more with the environment. Specifically, upstream modules are trained to generate better prompts (i.e., prompt engineering) using supervised learning (Shin et al., 2020) or reinforcement learning (Deng et al., 2022; Zhang et al., 2022b). Downstream modules can be a value function to generate better actions (Ahn et al., 2022; Yao et al., 2023) or a semantic translation to provide admissible actions (Huang et al., 2022). We demonstrate the differences among these

paradigms in Figure 5, along with a comparison to standard RL algorithms that do not incorporate LLMs. In standard RL, a trainable component, specifically the RL agent, interacts directly with the domain environment. In contrast, in approaches with trainable upstream components, instead of directly engaging with the domain environment, the RL agent interacts with the LLMs by providing prompts. The LLMs then communicate with the domain environment by offering actions corresponding to the given prompts. However, the interaction pipeline varies in approaches with trainable downstream components, where LLMs serve as intermediaries between domain environments and trainable components. In this case, LLMs receive prompts from the domain environment and supply inputs to the trainable components, which subsequently optimize actions to be delivered to the domain environment.

Finally, we would like to emphasize that our proposed approaches in Section 3 belong to the category of in-context learning (ICL), as per our previous taxonomy. The primary motivation for developing methods along this line of research is that ICL effectively balances crucial aspects and is well-suited to address the needs of a broad spectrum of industrial control problems.

In this section, we demonstrate the process of interactively fine-tuning prompts using two distinct methods: 1) refining prompts through the selection of demonstrations, and 2) enhancing prompts by generating new demonstrations. Further details are provided as follows.

Demonstrations Scoring. In this approach, we initially gather a collection of tuples in the form of (demonstration, state, reward), where "demonstration" refers to a specific demonstration provided to GPT-4 in the prompt, "state" represents the current state of the environment, and "reward" is the outcome achieved by executing the action suggested by GPT-4. We continue to interact with GPT-4 using various demonstrations for a certain number of steps, and subsequently utilize the resulting set of tuples to train a demonstration scoring model. In this model, features consist of pairs of demonstration and state, while targets comprise rewards. To improve diversity of the dataset, we could obtain half of the tuples by employing random demonstrations sampled from the expert experience buffer, while the remaining tuples are gathered using the demonstration that is most closely aligned with the current state, as suggested by the KNN model introduced in Section 3.

Upon acquiring a demonstration scoring model, it will supersede the KNN model depicted in Figure 1, enabling the identification of the most efficient demonstrations for the current state. Specifically, given the present state, we employ the scoring model to evaluate all demonstrations within the expert experience buffer, selecting those with the highest scores as the demonstrations to be incorporated into the prompt provided to GPT-4. We note that the scoring model can be consistently updated using the most recent tuples collected during interactions with GPT-4. By doing so, the scoring model continually enhances its accuracy, specifically in the current environment, which ultimately results in improved performance of GPT-4.

Demonstrations Selection. The demonstration scoring model is a supervised model that cannot capture farsighted rewards, which may lead it to adopt policies that only consider short-term rewards. This limitation might hamper the performance of GPT-4 in problems requiring long-term planning and reasoning. To address this issue, the demonstration generation approach takes a step further by fully integrating the RL algorithm with GPT-4, enabling it to have long-term thinking and reasoning capabilities without the need to fine-tuning GPT-4 itself and include all interaction history in the prompt. The approach consists of the following steps:

- First, we transform each demonstration into an embedding representation, as described in Section 3.
- Second, we employ dimension reduction algorithms (e.g., PCA) to decrease the dimension of all embedding representations to a smaller size.

- Third, we train an RL agent using algorithms like PPO to generate embedding representations at each step. In other words, at each step, the agent takes the current state of an environment as input and outputs an action of dimension same as the reduced dimension in the second step.
- Finally, as in Dulac-Arnold et al. (2015), we identify the nearest demonstrations in the expert experience buffer, whose reduced embedding representations are closest to the action.

Detailed experiment results for the two approaches will be updated in the future version of this paper.

6. Conclusion

In this paper, we demonstrate effectiveness of LLMs by integrating with existing decision-making approaches on industrial control optimization. Our experiments reveal that incorporating LLMs significantly enhances the generalization and robustness of traditional methods, potentially scaling to a wider range of scenarios with reduced training effort. However, in the context of industrial control optimization, LLMs exhibit certain limitations, such as lack of adaption capability to environmental changes and the ability to learn and forget selectively, among others. We also investigate techniques to augment LLMs' capabilities in these areas. As future work, we aim to further explore the potential of LLMs in various industrial domains and develop a comprehensive framework that combines LLMs with existing approaches to address a wide array of control optimization challenges in the industry.

References

Afram, A. and Janabi-Sharifi, F. (2014). Theory and applications of hvac control systems–a review of model predictive control (mpc). *Building and Environment*, 72:343–355.

Afroz, Z., Shafiullah, G., Urmee, T., and Higgins, G. (2018). Modeling techniques used in building hvac control systems: A review. *Renewable and sustainable energy reviews*, 83:64–84.

Agarwal, A., Bird, S., Cozowicz, M., Hoang, L., Langford, J., Lee, S., Li, J., Melamed, D., Oshri, G., Ribas, O., et al. (2016). Making contextual decisions with low technical debt. *arXiv preprint arXiv:1606.03966*.

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

Al-Ali, A., Tubaiz, N. A., Al-Radaideh, A., Al-Dmour, J. A., and Murugan, L. (2012). Smart grid controller for optimizing hvac energy consumption. In *2012 International Conference on Computer Systems and Industrial Informatics*, pages 1–4. IEEE.

Alcalá, R., Benítez, J. M., Casillas, J., Cordón, O., and Pérez, R. (2003). Fuzzy control of hvac systems optimized by genetic algorithms. *Applied Intelligence*, 18:155–177.

Antropic (2023). Claude 2.

Azuatalam, D., Lee, W.-L., de Nijs, F., and Liebman, A. (2020). Reinforcement learning for whole-building hvac control and demand response. *Energy and AI*, 2:100020.

Baker, B., Akkaya, I., Zhokov, P., Huizinga, J., Tang, J., Ecoffet, A., Houghton, B., Sampedro, R., and Clune, J. (2022). Video pretraining (vpt): Learning to act by watching unlabeled online videos. *Advances in Neural Information Processing Systems*, 35:24639–24654.

Bakhtin, A., Brown, N., Dinan, E., Farina, G., Flaherty, C., Fried, D., Goff, A., Gray, J., Hu, H., et al. (2022). Human-level play in the game of diplomacy by combining language models with strategic reasoning. *Science*, 378(6624):1067–1074.

Beeching, E., Han, S., Lambert, N., Rajani, N., Sanseviero, O., Tunstall, L., and Wolf, T. (2023). Open llm leaderboard.

Belic, F., Hocenski, Z., and Sliskovic, D. (2015). Hvac control methods-a review. In *2015 19th International Conference on System Theory, Control and Computing (ICSTCC)*, pages 679–686. IEEE.

Bellman, R. (1966). Dynamic programming. *Science*, 153(3731):34–37.

Botvinick, M., Ritter, S., Wang, J. X., Kurth-Nelson, Z., Blundell, C., and Hassabis, D. (2019). Reinforcement learning, fast and slow. *Trends in cognitive sciences*, 23(5):408–422.

Brown, D., Goo, W., Nagarajan, P., and Niekum, S. (2019). Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *International conference on machine learning*, pages 783–792. PMLR.

Brown, D. S., Goo, W., and Niekum, S. (2020). Better-than-demonstrator imitation learning via automatically-ranked demonstrations. In *Conference on robot learning*, pages 330–359. PMLR.

Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., Lee, P., Lee, Y. T., Li, Y., Lundberg, S., et al. (2023). Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., and Varoquaux, G. (2013). API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122.

Cai, Y., Zhang, C., Shen, W., He, X., Zhang, X., and Huang, L. (2022). Imitation learning to outperform demonstrators by directly extrapolating demonstrations. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 128–137.

Carta, T., Romac, C., Wolf, T., Lamprier, S., Sigaud, O., and Oudeyer, P.-Y. (2023). Grounding large language models in interactive environments with online reinforcement learning. *arXiv preprint arXiv:2302.02662*.

Cer, D., Yang, Y., yi Kong, S., Hua, N., Limtiaco, N., John, R. S., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., Sung, Y.-H., Strope, B., and Kurzweil, R. (2018). Universal sentence encoder.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. (2021). Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097.

Chevalier-Boisvert, M., Bahdanau, D., Lahlou, S., Willems, L., Saharia, C., Nguyen, T. H., and Bengio, Y. (2018). Babyai: A platform to study the sample efficiency of grounded language learning. *arXiv preprint arXiv:1810.08272*.

Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, E., Wang, X., Dehghani, M., Brahma, S., et al. (2022). Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.

Dai, Y., Yang, M., Dai, B., Dai, H., Nachum, O., Tenenbaum, J., Schuurmans, D., and Abbeel, P. (2023). Learning universal policies via text-guided video generation. *arXiv preprint arXiv:2302.00111*.

Deng, M., Wang, J., Hsieh, C.-P., Wang, Y., Guo, H., Shu, T., Song, M., Xing, E. P., and Hu, Z. (2022). Rlprompt: Optimizing discrete text prompts with reinforcement learning. *arXiv preprint arXiv:2205.12548*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Ding, Y., Feng, M., Liu, G., Jiang, W., Zhang, C., Zhao, L., Song, L., Li, H., Jin, Y., and Bian, J. (2022). Multi-agent reinforcement learning with shared resources for inventory management. *arXiv preprint arXiv:2212.07684*.

Dong, Q., Li, L., Dai, D., Zheng, C., Wu, Z., Chang, B., Sun, X., Xu, J., and Sui, Z. (2022). A survey for in-context learning. *arXiv preprint arXiv:2301.00234*.

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

Du, Y., Watkins, O., Wang, Z., Colas, C., Darrell, T., Abbeel, P., Gupta, A., and Andreas, J. (2023). Guiding pretraining in reinforcement learning with large language models. *arXiv preprint arXiv:2302.06692*.

Dulac-Arnold, G., Evans, R., Sunehag, P., and Coppin, B. (2015). Reinforcement learning in large discrete action spaces. *CoRR*, abs/1512.07679.

Fan, L., Wang, G., Jiang, Y., Mandlekar, A., Yang, Y., Zhu, H., Tang, A., Huang, D.-A., Zhu, Y., and Anandkumar, A. (2022). Minedojo: Building open-ended embodied agents with internet-scale knowledge. *arXiv preprint arXiv:2206.08853*.

Gladwell, M. (2008). *Outliers: The story of success*. Little, Brown.

Gudibande, A., Wallace, E., Snell, C., Geng, X., Liu, H., Abbeel, P., Levine, S., and Song, D. (2023). The false promise of imitating proprietary llms. *arXiv preprint arXiv:2305.15717*.

Hao, S., Gu, Y., Ma, H., Hong, J. J., Wang, Z., Wang, D. Z., and Hu, Z. (2023). Reasoning with language model is planning with world model. *arXiv preprint arXiv:2305.14992*.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. (2019). Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. (2021). Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Huang, W., Abbeel, P., Pathak, D., and Mordatch, I. (2022). Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International Conference on Machine Learning*, pages 9118–9147. PMLR.

James, S., Ma, Z., Arrojo, D. R., and Davison, A. J. (2020). Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026.

Kant, Y., Ramachandran, A., Yenamandra, S., Gilitschenski, I., Batra, D., Szot, A., and Agrawal, H. (2022). Housekeep: Tidying virtual households using commonsense reasoning. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIX*, pages 355–373. Springer.

Khan, M. W., Choudhry, M. A., and Zeeshan, M. (2013). An efficient design of genetic algorithm based adaptive fuzzy logic controller for multivariable control of hvac systems. In *2013 5th Computer Science and Electronic Engineering Conference (CEEC)*, pages 1–6. IEEE.

Kwon, M., Xie, S. M., Bullard, K., and Sadigh, D. (2023). Reward design with language models. *arXiv preprint arXiv:2303.00001*.

Lester, B., Al-Rfou, R., and Constant, N. (2021). The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Li, X. L. and Liang, P. (2021). Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

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

Liu, J., Cai, W.-J., and Zhang, G.-Q. (2009). Design and application of handheld auto-tuning pid instrument used in hvac. In *2009 4th IEEE Conference on Industrial Electronics and Applications*, pages 1695–1698. IEEE.

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

Ma, Y., Anderson, G., and Borrelli, F. (2011). A distributed predictive control approach to building temperature regulation. In *Proceedings of the 2011 American Control Conference*, pages 2089–2094. IEEE.

Ma, Y., Borrelli, F., Hencey, B., Packard, A., and Bortoff, S. (2009). Model predictive control of thermal energy storage in building cooling systems. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 392–397. IEEE.

Ma, Y., Kelman, A., Daly, A., and Borrelli, F. (2012). Predictive control for energy efficient buildings with thermal storage: Modeling, stimulation, and experiments. *IEEE control systems magazine*, 32(1):44–64.

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

Mirinejad, H., Welch, K. C., and Spicer, L. (2012). A review of intelligent control techniques in hvac systems. *2012 IEEE energytech*, pages 1–5.

OpenAI (2022). Chatgpt: Optimizing language models for dialogue. *OpenAI Blog*.

OpenAI (2023). Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. (2022). Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Pichai, S. (2023). An important next step on our ai journey. *Google AI Blog*.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. (2021). Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., and Sutskever, I. (2021). Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR.

Rawlings, J. B. (2000). Tutorial overview of model predictive control. *IEEE control systems magazine*, 20(3):38–52.

Rawlings, J. B., Mayne, D. Q., and Diehl, M. (2017). *Model predictive control: theory, computation, and design*, volume 2. Nob Hill Publishing Madison, WI.

Reed, S., Zolna, K., Parisotto, E., Colmenarejo, S. G., Novikov, A., Barth-Maron, G., Gimenez, M., Sulsky, Y., Kay, J., Springenberg, J. T., et al. (2022). A generalist agent. *arXiv preprint arXiv:2205.06175*.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015). Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Shin, T., Razeghi, Y., Logan IV, R. L., Wallace, E., and Singh, S. (2020). Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv preprint arXiv:2010.15980*.

Shinn, N., Labash, B., and Gopinath, A. (2023). Reflexion: an autonomous agent with dynamic memory and self-reflection. *arXiv preprint arXiv:2303.11366*.

Sun, T., Shao, Y., Qian, H., Huang, X., and Qiu, X. (2022). Black-box tuning for language-model-as-a-service. In *International Conference on Machine Learning*, pages 20841–20855. PMLR.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Szot, A., Clegg, A., Undersander, E., Wijmans, E., Zhao, Y., Turner, J., Maestre, N., Mukadam, M., Chaplot, D. S., Maksymets, O., et al. (2021). Habitat 2.0: Training home assistants to rearrange their habitat. *Advances in Neural Information Processing Systems*, 34:251–266.

Tashtoush, B., Molhim, M., and Al-Rousan, M. (2005). Dynamic model of an hvac system for control analysis. *Energy*, 30(10):1729–1745.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. (2023a). Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. (2023b). Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Villar, J. R., de la Cal, E., and Sedano, J. (2009). A fuzzy logic based efficient energy saving approach for domestic heating systems. *Integrated Computer-Aided Engineering*, 16(2):151–163.

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

Wang, J., Zhang, C., and Jing, Y. (2008). Application of an intelligent pid control in heating ventilating and air-conditioning system. In *2008 7th World Congress on Intelligent Control and Automation*, pages 4371–4376. IEEE.

Wang, Z., Cai, S., Liu, A., Ma, X., and Liang, Y. (2023b). Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents. *arXiv preprint arXiv:2302.01560*.

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., et al. (2022). Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

Wei, T., Wang, Y., and Zhu, Q. (2017). Deep reinforcement learning for building hvac control. In *Proceedings of the 54th annual design automation conference 2017*, pages 1–6.

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., and Narasimhan, K. (2023). Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. (2022). React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Yu, L., Sun, Y., Xu, Z., Shen, C., Yue, D., Jiang, T., and Guan, X. (2020a). Multi-agent deep reinforcement learning for hvac control in commercial buildings. *IEEE Transactions on Smart Grid*, 12(1):407–419.

Yu, T., Quillen, D., He, Z., Julian, R., Hausman, K., Finn, C., and Levine, S. (2020b). Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pages 1094–1100. PMLR.

Yuan, H., Zhang, C., Wang, H., Xie, F., Cai, P., Dong, H., and Lu, Z. (2023). Plan4mc: Skill reinforcement learning and planning for open-world minecraft tasks. *arXiv preprint arXiv:2303.16563*.

Zhang, C., Duan, Y., Chen, X., Chen, J., Li, J., and Zhao, L. (2023). Towards generalizable reinforcement learning for trade execution. *The 32nd International Joint Conference on Artificial Intelligence (IJCAI-23)*.

Zhang, C., Shi, Y., and Chen, Y. (2022a). Bear: Physics-principled building environment for control and reinforcement learning.

Zhang, T., Wang, X., Zhou, D., Schuurmans, D., and Gonzalez, J. E. (2022b). Tempera: Test-time prompt editing via reinforcement learning. In *The Eleventh International Conference on Learning Representations*.

Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., et al. (2023). A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Zhao, Z., Wallace, E., Feng, S., Klein, D., and Singh, S. (2021). Calibrate before use: Improving few-shot performance of language models. In *International Conference on Machine Learning*, pages 12697–12706. PMLR.

Zheng, L., Sheng, Y., Chiang, W.-L., Li, D., Li, Z., Lin, Z., Wu, Z., Zhuang, S., Zhuang, Y., Zhang, H., Stoica, I., Gonzalez, J. E., and Xing, E. P. (2023). Chatbot arena.

Zhong, W., Cui, R., Guo, Y., Liang, Y., Lu, S., Wang, Y., Saied, A., Chen, W., and Duan, N. (2023). Agieval: A human-centric benchmark for evaluating foundation models. *arXiv preprint arXiv:2304.06364*.

Zhou, C., Li, Q., Li, C., Yu, J., Liu, Y., Wang, G., Zhang, K., Ji, C., Yan, Q., He, L., et al. (2023). A comprehensive survey on pretrained foundation models: A history from bert to chatgpt. *arXiv preprint arXiv:2302.09419*.

Zhu, X., Chen, Y., Tian, H., Tao, C., Su, W., Yang, C., Huang, G., Li, B., Lu, L., Wang, X., et al. (2023). Ghost in the minecraft: Generally capable agents for open-world enviroments via large language models with text-based knowledge and memory. *arXiv preprint arXiv:2305.17144*.