CraftFactory: A Conditioned Control Policy Benchmark for Compositional Generalization

Jinbing Hou, Youpeng Zhao, Jian Zhao

Polixir Technologies, Nanjing, China
jinbing.hou@polixir.ai, youpeng.zhao@polixir.ai, jian.zhao@polixir.ai

Abstract

Humans excel at understanding and reasoning about novel and compositionally structured knowledge, largely due to their capacity for compositional generalization, a cognitive skill that has recently been validated in structured neural networks. However, most existing research has focused primarily on semantic translation within canonical language environments, often neglecting the explicit connection to compositional generalization behavior. In contrast, humans typically demonstrate this ability through interaction with their environments, rather than solely through internal reasoning. To address this gap, we propose CraftFactory, a benchmark designed for evaluating compositional generalization in an interactive control environment. This benchmark introduces a new challenge for testing compositional generalization in a more realistic and comprehensive manner. CraftFactory stands out due to three key features: (1) it offers an open-ended interactive control environment with thousands of items and flexible actions; (2) it requires advanced compositional inference through various combinations and complex permutations of instructions; and (3) it evaluates compositional generalization intuitively through interactive behavior. By leveraging CraftFactory, we aim to promote the development of more advanced compositional generalization methods, thereby contributing to the broader field of general AI.

Code — https://github.com/Aubing-H/craftfactory

Introduction

Humans possess the ability to understand and infer new knowledge primarily through compositional generalization, an algebraic reasoning skill that allows them to learn and infer new combinations of knowledge components. This ability greatly enriches their knowledge and skills. For instance, a novice skateboarder who learns "ollie" can likely "ollie twice" by applying the concept of "twice" from "jump twice", demonstrating the ability to infer and execute new combinations. Compositional generalization is crucial for developing a general intelligent agent, as it significantly enhances learning efficiency by extracting fundamental knowledge and task structures from just a few demonstrations. Consider an open-world scenario where it is impractical to train an all-encompassing model due to the infinite possible states. However, the components of these states are more uniformly distributed and manageable compared to complex and biased aggregates.

Recent research (Keysers et al. 2019; Kim and Linzen 2020; Akyürek, Akyürek, and Andreas 2020) has focused on addressing compositional generalization within symbolic language (Lake and Baroni 2017) or geometric graphic environments (Zhao et al. 2022). It has been demonstrated that compositional generalization works effectively in basic language settings, using structured neural networks to reason about novel combinations of vocabulary after learning distinct but similar sentences. Many studies test compositional generalization by converting abstract language into more intuitive forms, such as translating "jump twice" into "jump jump". This approach evaluates compositional generalization based on the accuracy of the predicted sequential vocabulary. However, models built in symbolic environments, like language or graphic symbols, are self-contained and lack external interaction. Their inference is limited to vocabulary or symbol spaces, which are far removed from the visual and interactive real world. In addition, these environments are often simplified with a fixed vocabulary set. While these platforms are suitable for evaluating semantic translation, they are limited in testing compositional generalization in interactive scenarios.

Consider the concept of "jump twice", humans can not only verbalize "jump jump", but also physically execute two consecutive jumps. Unlike symbolic reasoning, this requires aligning the word "jump" with the action, judging when the first jump is complete, and initiating the next. Unlike environments like SCAN(Lake and Baroni 2017) and COGS(Kim and Linzen 2020), compositional generalization with control involves extrapolating behavior through implicit control structures, such as judgment, which are absent in symbolic models. To broaden the scope of compositional generalization, we introduce CraftFactory, a general control environment in Minecraft. In CraftFactory, compositional generalization is tested not only on symbolic representations but also on action distributions conditioned by state. Exploring compositional generalization in a broader and novel scenario is crucial. For control policies, incorporating compositional generalization helps decompose existing knowledge and apply it to new but similar scenarios, enhancing data efficiency in the training process.

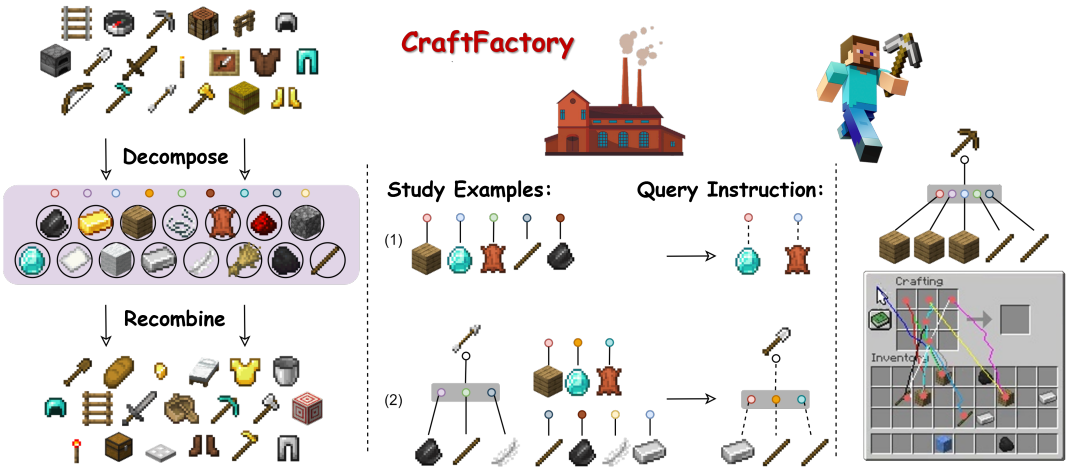

Figure 1: An illustration of CraftFactory. The left: in CraftFactory, an agent can learn from the crafting process of part items, decompose into the components of this behavior, and finally infer to create a novel cluster of items. The middle: two kinds of tasks for compositional generalization. (1) is to evaluate the generalization of $(item, slot)$ in one-step drag-and-drop and (2) is to evaluate the generalization of multi-step drag. The right: the trajectory of crafting a $wooden\_pickaxe$ which requires a continuous control of drag-and-drop involving a series of precise clicks.

In CraftFactory, we design a series of tasks to evaluate compositional generalization within a control framework. These tasks are divided into two categories: basic skill tasks, which require minimal compositional generalization, and multi-step control tasks, which demand a high degree of compositional generalization. The evaluation of compositional generalization is based on the separation of different combinations of elements for training and testing. During training, we limit the diversity of learning cases, and during testing, we assess models primarily by their success rate on extrapolated cases. Currently, three popular methods address the challenges of compositional generalization: (1) equivariant map modeling with permutation groups (Gordon et al. 2020; Zhao et al. 2022), (2) permutation invariance with self-attention blocks (Zhou et al. 2022), and (3) meta-networks (Gordon et al. 2020). For the first method, we used a learnable head instead of word permutation, avoiding high computational costs.

In summary, our contributions are as follows.

- Proposing CraftFactory, a promising sequential control benchmark for compositional generalization.
- Developing a set of challenging compositional generalization tasks in CraftFactory.
- Discussing the importance of compositional generalization and arguing why CraftFactory is essential for this area of research.

Related Works

In this section, we review key works on compositional generalization, highlighting advancements in the field, and introducing a state-of-the-art method. We then explore studies focused on Minecraft, an open-ended environment that has garnered significant interest among researchers. Lastly, we discuss related approaches for condition fusion in control policies.

Compositional Generalization

Compositional generalization has recently gained attention in the deep learning community (Yin et al. 2023; Lake and Baroni 2023) due to its potential to extend the generalization capabilities of neural networks(Fodor and Pylyshyn 1988). Earlier works by Lake and Baroni (2017); Lake (2019) challenged the notion of neural networks being inherently unlearnable in this context. Recent studies(Furrer et al. 2020; Gordon et al. 2020; Keysers et al. 2019) have predominantly concentrated on semantic compositional generalization within symbolic language environments(Lake and Baroni 2017; Kim and Linzen 2020), employing sequence-to-sequence methods(Russin et al. 2019; Lake 2019) for predictive transformations. Additionally, there is a growing body of research on compositional generalization within control environments(Zhou et al. 2022; Zhao et al. 2022), where it is leveraged to address the challenges of multi-task control. Our work diverges from these efforts by focusing on establishing a general benchmark for compositional generalization in sequential control, aiming to develop a human-like ability to generalize from thinking to practice.

Minecraft as an Open-Ended Environment

Minecraft, a popular game with an open Python interface(Johnson et al. 2016; Guss et al. 2019; Fan et al. 2022), has become a fertile ground for AI research. Its open-ended nature and rich interaction possibilities provide a unique platform for developing and testing advanced AI methodologies. Imitation learning has been extensively explored to enable AI agents to perform multi-tasking(Cai et al. 2023a,b; Lifshitz et al. 2023). These studies involve training agents to mimic human players, thereby leveraging large datasets of human gameplay to enhance the agents' performance and versatility across various tasks. Another significant area of research is

language-based planning(Wang et al. 2023c,b,a; Yuan et al. 2023) which investigates how natural language instructions can be used to guide AI agents in planning and executing basic skills. Reinforcement learning (RL) has also been a major focus(Yuan et al. 2023; Hafner et al. 2023). In Minecraft, RL algorithms are used to train agents through trial and error, enabling them to discover optimal strategies for achieving various objectives within the game. This approach has been particularly successful in developing agents capable of mastering specific tasks and adapting to new challenges.

Despite these advancements, our work distinguishes itself by concentrating on the compositionality in sequential control within crafting tasks. We aim to explore Minecraft as a viable and general environment for studying compositional generalization. Our research seeks to develop methods that enable AI agents to generalize from learned experiences to novel situations, thereby enhancing their adaptability and robustness.

Goal-Conditioned Policies

Goal-conditioned learning policies (Kaelbling 1993; Schaul et al. 2015) play a crucial role in developing multi-task models within a unified framework. Recent advancements in this area have primarily focused on imitation learning(Ding et al. 2019), reinforcement learning(Liu, Zhu, and Zhang 2022), and contrastive learning(Eysenbach et al. 2022). These policies offer versatile interfaces for integrating basic skills(Cai et al. 2023a) with long-horizon planning(Nasiriany et al. 2019; Wang et al. 2023c). Several condition fusion methods(Jayakumar et al. 2019; Mallya, Davis, and Lazebnik 2018) have been proposed to enhance this integration. For instance, FiLM(Perez et al. 2018), which effectively incorporates condition embeddings within convolutional layers, facilitates communication between conditions and input features. In our work, we adopt the FiLM layer in our conditioned layers, leveraging its ability to fuse embedded condition features with visual inputs.

Benchmarks for Compositional Generalization

We compared four different environments of previous compositional generalization benchmarks. SCAN(Lake and Baroni 2017) is a popular language environment for compositional generalization. SCAN is a well-known language environment that assesses compositional generalization through semantic translation performance, using a vocabulary library. OOE(Zhao et al. 2022) is a world model-based environment designed to bridge gaps among new homomorphic tasks. COGS(Kim and Linzen 2020) serves as a benchmark for semantic translation, offering extensive grammar support and a rich corpus. GSCAN (Ruis et al. 2020) is a benchmark for evaluating compositional generalization in situated language understanding, revealing that current neural networks struggle with novel compositions in a grid world context. Unlike these, CraftFactory focuses on sequential control compositional generalization, featuring long-horizon decision processes and multi-modal input. The most similar environment is Meta-World(Zhou et al. 2022), which addresses compositional generalization by integrating multiple goals into its model. Different from meta-world, CraftFactory, based on the open-world game Minecraft, obtains large quantities of items and open-ended recipes on these crafts. The openness of crafting various items poses a challenge for sequential control in compositional generalization.

Principles for Compositional Generalization in Sequential Control

Inspired by compositional generalization—extrapolating new compound concepts from language components—we establish principles to assess compositional generalization in sequential control. We define compositional atomicity and describe compound properties. An ideal compositional experiment should adhere to two principles outlined by Keysers et al. (2019): similar component distributions and distinct compound distributions between training and testing sets. In the sequential control domain, we propose three main principles for compositional generalization:

- **Similar Distribution of Instruction Components**: If control instructions are decomposable, the distribution of instruction components in testing should closely match that of the training set.
- **Distinctive instructions**. Testing instructions should differ significantly from training ones. This difference can be measured by the distribution distance of components within instructions.
- **Consistent State Distribution Conditioned on Instructions**. State distribution in training and testing sets should be similar. An imbalanced state distribution complicates the learning of a robust generalized policy.

We measure task compositionality using the following formulations. Suppose there is a state space $S = \{s_t\}$, an instruction component space $C = \{c_i\}$, and a compound space $G = \{g_i\}$. Any element $g_i \in G$ is composed of a subset permutation of space C. The probability of component c_i appearing in a set is $P(c_i)$ and in a compound g_j is $P(c_i|g_j)$. Compositional generalization is measured as:

$$CG = D_{instr} + \alpha F_{comp} + \beta F_{state}, \tag{1}$$

$$D_{instr} = KL\left(P_{train}(c|g) \parallel P_{test}(c|g)\right), \tag{2}$$

$$F_{comp} = 1 - JS\left(P_{train}(c) \parallel P_{test}(c)\right), \tag{3}$$

$$F_{state} = 1 - JS\left(P_{train}(s|g) \parallel P_{test}(s|g)\right), \tag{4}$$

Here, D_{instr} is a Kullback-Leibler divergency measuring the distance between two component distributions conditioned on compound space. F_{comp} and F_{state} are calculated by the Jensen-Shannon divergence which is a kind of symmetric measure. α and β are two adjustable parameters to balance the three measures. $P_{train}(x)$ and $P_{test}(x)$ denote the probability of event x in training and testing sets, respectively. This formulation evaluates state closeness, componential closeness, and compound differences between training and testing sets.

CraftFactory and Task Definitions

We propose CraftFactory, a control-based multi-task environment designed as a robust platform for testing compositional generalization. CraftFactory comprises two types of tasks:

Benchmark	sequential	interactive	long-horizon	multi-modal	open-ended
SCAN(Lake and Baroni 2017)	✔	✗	✗	✗	✗
OOE(Zhao et al. 2022)	✗	✔	✗	✔	✗
COGS(Kim and Linzen 2020)	✔	✗	✗	✗	✗
Meta-World(Zhou et al. 2022)	✔	✔	✔	✔	✗
CraftFactory	✔	✔	✔	✔	✔

Table 1: A comparison of CraftFactory and congeneric benchmarks on compositional generalization.

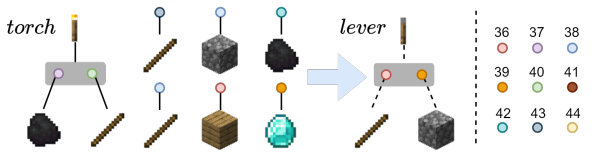

Figure 2: A two-step compositional task. The left: a compositional generalization from learning crafting *torch* and a few other drag demonstrations to practicing crafting *lever*. The examples of *torch* imply knowledge about combining two steps into a whole; the one-drop example implies knowledge about how to drag with items and slots. The right: nine colored circles represent nine slot position numbers in crafting GUI. The line between the item and circle represents an operation of dragging the item onto a slot

(a) (b)

Figure 3: CraftFactory environment: The left(a) is a workbench GUI surface also the whole observation space of the model, while the right(b) is an illustration of multi-step drag-and-drop trajectory when crafting a *wooden_pickaxe*.

fundamental tasks and complex, multi-step tasks that demand a range of skills and structured planning. This environment allows for the evaluation of various methods using comprehensive datasets, a standard baseline model, and a functional crafting setup for experimentation.

Introduce to Crafting Environment

CraftFactory builds upon the MineRL workbench crafting scenario(Guss et al. 2019), providing a vision-based, interactive, and open-ended environment for AI research. Within CraftFactory, agents interact with a vast array of items across a graphical interface featuring 46 slots: 36 for storage within the inventory bar, 9 for item arrangement in the workbench bar, and 1 in the result bar for completed products. To craft an item, the agent must follow a recipe, selecting raw materials from the inventory and placing them onto the workbench. A new item appears in the result bar when the correct pattern is formed, completing the process by transferring the item back to the inventory.

The observation space is solely derived from a 360 x 640 pixel GUI frame. The action space includes discrete cursor clicks for item manipulation and continuous two-dimensional cursor movements. From the thousands of items in Minecraft, we selected 59 compounding items and 22 material items for testing. Compounding items consist of various raw materials or intermediate products, such as *stick*. The synthesis process involves a sequence of material placements guided by a recipe, defined as a sequence of drag-and-drop actions. Each step in this sequence is represented by a drag function $Drag(item, slot)$, where 'item' denotes the material and 'slot' indicates its designated position, as detailed in the appendix.

Crafting Task Definition and Its Compositionality

There are three primary types of generalization in crafting tasks: (1) generalization to all the combinations of items and slots in the drag function, (2) generalization to all permutations of different drag-and-drop sequences, (3) generalization to all nested recipes from various recipe patterns. This work primarily explores the first two types of compositional generalization.

In our crafting task, there are 81 items available. During synthesis, up to 10 items (9 materials and one output product) are utilized. We define a one-step drag-and-drop process as $Drag(a, b), a \in S_{item}, b \in S_{slot}$, S_{item} is defined as a set of items and S_{slot} encompasses all slots in workbench GUI. There is a combination generalization on items and slots for a one-step $Drag(a, b)$. The combination generalization on items and slots is tested through the following benchmark:

Task-1: basic none-generalization task. In this task, datasets comprising all item-slot pairs are randomly divided into training and testing sets. The component distributions in both sets are nearly identical, minimizing differences in compound distributions.

Task-2: primary-generalization task. The model is given n types of items and slots, with datasets composed of $k \times n$ training pairs from a total of $n \times n$ pairs in total. The task difficulty increases as k move away from n. For multi-step control, the process is represented as a sequence Q:

$$Q = (Drag(a_1, b_1), ..., Drag(a_d, b_d)), \quad (5)$$

where $a_i \in S_{item}, b_i \in S_{slot}$, d is the number of control steps. For example, crafting *wooden_pickaxe* can be described as (( , 36), (, 37), (, 38), (, 40), (, 43)). Permutation generation is crucial in multi-step control, allowing for substitution and reordering of steps. Compositional generalization is essential due to the vast number of potential cases when the number of steps increases.

Task-3: multi-step none-generalization task. Datasets with various step-length trajectories, including random drag-and-drop fragments, are randomly split into training and evaluation sets. The model learns to recognize patterns with different horizons.

Task-4: two-step generalization task. Training datasets include necessary one-step component datasets and n types of two-step datasets. The test case involves a different type of two-step control process.

Task-5: multi-step generalization task. The training dataset consists of cases with three or more steps and basic one-step datasets. The test case is a new multi-step task significantly different from those in the training set.

Crafting Pipeline for Compositional Generalization

In this section, we elaborate on related methodologies and ours, dataset collection, and model backbone in our CraftFactory pipeline for compositional generalization.

Methodologies for Compositional Generalization

To evaluate compositional generalization methods in CraftFactory, we introduce three popular approaches in our benchmark:

Meta-Learning. The MLC framework(Lake and Baroni 2023; Lake 2019) employs a sequence-to-sequence approach for compositional generalization. It uses a vanilla RNN and Transformer structure to encode sequential vocabularies, embedding both input and output sequences.

Self Attention. EFMDP(Zhou et al. 2022) utilizes an entity-based compositional structure with self-attention, training an end-to-end model across multiple goals.

Equivariant Map. This approach leverages group equivariance tools(Gordon et al. 2020) for compositional generalization in language. While effective, it becomes computationally complex as group numbers increase. HOMW(Zhao et al. 2022) designs an equivariant map in action and state spaces for homogeneous tasks, achieving compositional generalization in an object-oriented world model.

Group Equivariant Maps

The challenge of conditioned control in decision-making is crucial, as it enables the execution of multiple homogeneous tasks within a single model. Here, we propose that the compositionality of conditioned instructions can be effectively established and evaluated using group theory(Scott 2012).

Permutation Groups To begin, consider the fundamental properties of a group. A group G is defined as $< V, \sigma >$, where V is a set of n elements and σ is a group operation that satisfies closure, associativity, identity, and invertibility. The set of all permutations of n elements forms a symmetric group denoted G_s, with an order of $n!$ under composition. Our focus is on a permutation group G_p, a subgroup of G_s which adheres to these group properties.

Inspired by permutation equivariance in language (Gordon et al. 2020), we propose that instructions for sequential multi-step tasks can also leverage permutation equivariance. Given a set of atomic task instructions $\mathcal{V}$, all permutations form the symmetric group $\mathcal{G}_s$. Let e be the identity of $\mathcal{G}_\int$. Define a closed collection of permutation operation $\mathcal{G} = \{g_1, g_2, ..., g_N\}$ on e, forming a permutation group $\mathcal{X}$ that includes the identity e. A closed group operation satisfied the property: for $\forall g_i, g_j \in \mathcal{G}$, there $\exists g_k \in \mathcal{G}$, satisfying

$$g_i(g_j(x)) = g_k(x), \text{ for } \forall x \in \mathcal{X}, \tag{6}$$

$\mathcal{X}$ is called the permutation image of operation $\mathcal{G}$, sometimes we use $\mathcal{G}$ represent the permutation group itself.

Equivariant Maps Suppose that we have a set $\mathcal{Y} = \{y_k\}_k^N$ which constitutes embedding space of instructions and an operation $\varrho : \mathcal{Y} \to \mathcal{Y}$. The operation ϱ represents a shift in embedding space. An equivalent map $\phi : \mathcal{X} \to \mathcal{Y}$ is to map one group to another maintaining the structure unchanged between the origin group and the target group. The equivariance can be present as:

$$\phi(\varsigma_k(X_k)) = \varrho_k(\phi(X_k)), \text{ for } \forall X_k \text{ in } \mathcal{X}. \tag{7}$$

The equivariant map equation shows that a permutation of instructions can be represented as an element of a vector group and will not change the original structure.

Equivariant Maps for Crafting Task In the context of Minecraft, we define two object libraries: an item library $\mathbb{L}_m$ with 630 materials and a slot library $\mathbb{L}_s$ with 45 slots. The combinatorial possibilities of item-slot pairs are vast.

Fot a crafting process involving d steps, We established a pair of permutation groups $\mathcal{G}_m = \{m_1, ..., m_{N_m}\}$ and $\mathcal{G}_s = \{s_1, ..., s_{N_s}\}$ on set $\mathbb{L}_m$ and $\mathbb{L}_s$. To create an equivariant embedding space, we define a group of embedding space $\mathcal{Y} = \{y_1, ..., y_{N_y}\}$ and an equivariant map function $\Phi : \mathcal{G}_m \times \mathcal{G}_s \to \mathcal{Y}$. The equivariant map function satisfies the property: for $\forall m \in \mathcal{G}_m, s \in \mathcal{G}_s$

$$\Phi(g_m(m), g_s(s)) = g_y(\Phi(m, s)). \tag{8}$$

This demonstrates that task conditions can be derived from their representations or from transformations in other sequences.

Instruction-Conditioned Markov Decision Process

A canonical Markov Decision Process (MDP) is represented as $\mathcal{M} =< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$. Where $\mathcal{S}$ represents the state, $\mathcal{A}$ represents the action decided by policy π, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}'$ is the transition probability, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and γ is the discount factor.

An instruction-conditioned Markov Decision Process closely resembles goal-condition reinforcement learning (Ding et al. 2019). However, instruction conditions $I = (g_1, g_2, ..., g_n) \in \mathcal{I}$ tend to be more complicated since they are composed of multiple subgoals. This process is represented as $\mathcal{M}' =< \mathcal{S}, \mathcal{I}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$. The task is to maximize the action-value function given condition I to derive an optimal policy $\pi*$:

$$\pi* = argmax_\pi \mathbb{E}_{s \in \mathcal{S}, a \sim \pi(s)} \left[Q_\pi(s, a|I) \right]. \tag{9}$$

Dataset Preparation

To evaluate compositional generalization across different models, we collected 6,000 trajectories from randomly generated drag-and-drop datasets and 4,000 trajectories from item-specific drag datasets.

The random datasets were generated using human-like drag-and-drop actions combined with randomly selected recipes. Although it is impossible to encompass all possible dragging instructions, we ensured that the random sampling process traversed a broad range of components. These datasets were further divided into one-step "item-slot" datasets, serving as a foundational library for learning basic drag-and-drop skills.

The item-specific drag datasets were generated according to predefined item recipes. To facilitate learning and generalization of multi-step drag behaviors, these datasets were systematically converted into formatted datasets. During training, only raw pixel frames, action labels, and multi-step instructions were provided to the models.

Implements on Crafting Tasks

Backbone and Conditioned Layer. To implement compositional generalization in crafting tasks, we developed our baseline model mainly using VPT (Video Pre-Train) backbone(Baker et al. 2022) and a FiLM (Feature-wise Linear Modulation) Conditioned Layer(Perez et al. 2018; Cai et al. 2023a). VPT is a non-conditioned sequential predictive decision model based on Transformer-XL(Dai et al. 2019) architecture. The FiLM Condition Layer is a fusion technique that applies condition embeddings to convolutional layers across both height and width dimensions.

Encoding Compositional Instructions. To standardize the varying lengths of crafting steps, we pad each sequence to a uniform length of 10, as crafting an item and returning it to inventory requires at most 10 steps. We construct a list of 10 object head modules, each represented by the equation:

$$feat = ReLU(MLP(x_{onehot})). \tag{10}$$

Here, each module's condition is a one-hot embedding which is transformed into 512-length embedding by a neural head. After processing through a Multilayer Perception (MLP), the conditioned feature is added to the object dimension and output through a ReLU activation layer.

Generating Instructions with Large Language Models. In environments like Minecraft, as well as the real world, the number of possible task instructions is vast, making manual design impractical. Large Language Models (LLMs), which have significantly advanced in recent years, offer a powerful alternative to traditional language models. We utilize LLMs as auxiliary recipe generators, designing prompts that effectively query crafting instructions for given item names. This approach is robust even for uncommon items; for example, querying a *wooden_hoe* might return a recipe with specific components and their placements, such as (( , 36), (, 37), (, 40), (, 43)).

Using item names directly as conditions is impractical due to the vast, and potentially infinite, number of item names, particularly as new items are introduced in updated versions.

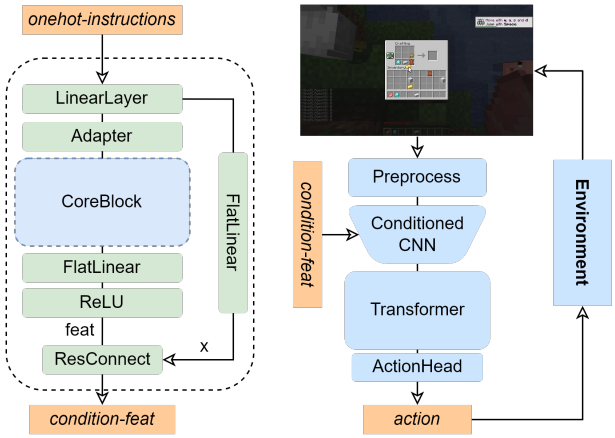

Figure 4: The pipeline model used for compositional generalization in CraftFactory. Left: instructions encoding pipeline. *FlatLinear* is composed of an instruction flatten layer and a linear layer. *ResConnect* is a function that residually connect *feat* with *x*: $ResConnect(feat, x) = (1 + feat) * x$. *CoreBlock* can be substituted for the other three methods. Right: VPT-Based conditioned framework for tasks. Conditioned CNN module is a 4-layer CNN with Film Condition Layer(Perez et al. 2018) and Transfomer module is a Transformer-XL(Dai et al. 2019) architecture.

Item names are not compositionally decomposable, necessitating extensive pre-training for each task. In contrast, crafting instructions are compositional and can be decomposed and recombined. Once a model learns these atomic crafting skills, it can generalize to craft any novel item.

Generalization Experiments

In this section, we study the three baseline methods in CraftFactory aiming to analyze the following problem: (1) Can the studied methods be generalized to new homogeneous tasks and extrapolate to crafting with unseen recipes? (2) What's the main challenge in CraftFactory according to experiment results?

Experimental Settings

To assess control generalization across various tasks, we selected 59 typical crafting items and 22 material items for training and testing within a compositional framework. Using the MineRL platform, CraftFactory initializes with an open workbench GUI and sufficient crafting materials for testing. During training, the model learns by imitating expert datasets, which include raw pixel frames, action labels, and conditioned instruction labels. Specific clusters of trajectories, relevant to each task, were selected for training (detailed in the appendix). In the testing phase, the model must infer actions based on fixed instructions and pixel frames directly from the environment.

In the subsection on crafting tasks, we define five tasks. Task 3 includes two non-generalization tasks used for intuitive comparison, while Tasks 2, 4, and 5 focus on evaluating

Methodology	Task-1(%)[in.] (,37)	Task-2(%)[ex.] (,37)	Task-3(%)[in.] (,36 | 39)	Task-4(%)[ex.] (,36 | 39)	Task-5(%)[ex.] (,36 | 39 | 42)
Meta-Learning	28.2 ± 2.7	25.5 ± 3.0	7.2 ± 2.5	2.7	0.9
Self Attention	4.6	6.1 ± 2.6	11.5 ± 6.8	2.5	0.0
Equivariant Map	31.3 ± 3.2	19.3 ± 5.3	7.5 ± 2.5	5.2 ± 3.9	0.0
CraftFactory	35.7 ± 6.7	4.5	7.4 ± 2.6	2.7	0.9

Table 2: Success rate on four baseline methods and different tasks. Task-1 and Task-2 are two skill-level experiments (train and test on only one-step trajectories). Task-1, training, and testing on the same large datasets, is a non-generalization experiment while Task-2 is tested on unseen cases. Task-3 is also non-generalization but is trained on the multi-step dataset. Task-4 is tested on extrapolated two-step cases. Task-5 is more difficult, tested on novel cases with three or more steps. During the evaluation, we test 4 groups and 30 times for each group. (If the success rate is below 5.0%, the standard deviation will not show.)

generalization performance. For training, approximately 100 trajectories were selected for each task. For testing, we introduced one or two novel test cases. Tasks 1 and 2 are designed to evaluate basic skill acquisition, while the remaining three tasks assess the model's ability to generalize across long-horizon scenarios.

Results on Crafting Tasks

We conducted experiments using our method alongside three popular compositional generalization approaches. Due to the challenges in directly implementing different methods, we employed a ResConnect structure (see figure 4) to adapt the core elements of these approaches for encoding instructions. The results (see Table 2) indicate that all four methods, including ours, have significant room for improvement. While techniques such as meta-learning, equivariant maps, and self-attention have demonstrated strong performance in symbolic or geometric environments, they face difficulties when applied to reasoning-to-practice tasks, particularly in both one-step and multi-step scenarios.

In the basic tasks (task-1 and task-2), our method, along with meta-learning and equivariant maps, showed relatively high performance in dragging task, achieving a notable success rate. However, our method underperformed in task-2. Upon analyzing the failure cases, we identified that recognition errors were the primary cause; for instance, when instructed to drag an , the agent mistakenly dragged instead. In multi-step tasks, the generalization ability of all methods decreased, largely due to an "amnesia problem" in long-horizon tasks, where the models struggled to retain information over extended sequences.

Results Analysis

Replicating methods from related benchmarks is inherently complex and often fails when transferring a method from one benchmark to another. Despite these challenges, we made efforts to adapt these methods to our benchmark by designing an adaptive module.

Although these re-implemented methods may not perform at their peak, our experimental results indicate that current compositional generalization methods, including our own, do not achieve high performance in reasoning-to-practice tasks. Our analysis reveals several key challenges within the CraftFactory environment: (1) Conditioned imitation learning with long-horizon behavior; (2) The ability to decompose components and recombine them in new scenarios.

Conclusions

Compositional generalization is a crucial capability for developing intelligent agents. However, existing environments primarily concentrate on semantic transformations and symbolic inference, lacking a robust platform for reasoning-to-practice applications. In this paper, we introduce a novel benchmark for compositional generalization in sequential control within the open-world environment MineRL. We distinguish between standard compositional generalization and its application in sequential control, outline three key properties for compositional generalization in control, and establish metrics for evaluating compositional generalization in sequential control. We develop a comprehensive pipeline for compositional generalization, model the problem using group theory, and conduct experiments comparing our approach with current state-of-the-art methods. Our findings indicate that existing methods, including ours, exhibit insufficient compositional generalization capabilities in the CraftFactory environment. Notably, the equivariant map approach, which utilizes position embedding in each slot, demonstrated the best performance among the tested methods.

Despite these advancements, several challenges remain in achieving effective compositional generalization with sequential control: (1) the sequential amnesia problem in executing long-horizon combinatorial tasks, (2) inadequate generalization for complex control tasks, and (3) limited ability to extrapolate to longer sequential tasks. CraftFactory emerges as a novel benchmark for advancing the study of compositional generalization, and we encourage further research to develop cutting-edge theories and methodologies in this domain.

Acknowledgments

We thank Yitao Liang and Zhaofeng He for their support and helpful advices throught-out this research. We thank Yongjang Jin and Zihao Wang for their paving work and helpful feedback.

References

Akyürek, E.; Akyürek, A. F.; and Andreas, J. 2020. Learning to recombine and resample data for compositional generalization. *arXiv preprint arXiv:2010.03706*.

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.

Cai, S.; Wang, Z.; Ma, X.; Liu, A.; and Liang, Y. 2023a. Open-World Multi-Task Control Through Goal-Aware Representation Learning and Adaptive Horizon Prediction. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 13734–13744.

Cai, S.; Zhang, B.; Wang, Z.; Ma, X.; Liu, A.; and Liang, Y. 2023b. GROOT: Learning to Follow Instructions by Watching Gameplay Videos. *ArXiv*, abs/2310.08235.

Dai, Z.; Yang, Z.; Yang, Y.; Carbonell, J.; Le, Q. V.; and Salakhutdinov, R. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.

Ding, Y.; Florensa, C.; Abbeel, P.; and Phielipp, M. 2019. Goal-conditioned imitation learning. *Advances in neural information processing systems*, 32.

Eysenbach, B.; Zhang, T.; Levine, S.; and Salakhutdinov, R. R. 2022. Contrastive learning as goal-conditioned reinforcement learning. *Advances in Neural Information Processing Systems*, 35: 35603–35620.

Fan, L. J.; 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*, abs/2206.08853.

Fodor, J. A.; and Pylyshyn, Z. W. 1988. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2): 3–71.

Furrer, D.; van Zee, M.; Scales, N.; and Scharli, N. 2020. Compositional Generalization in Semantic Parsing: Pre-training vs. Specialized Architectures. *ArXiv*, abs/2007.08970.

Gordon, J.; Lopez-Paz, D.; Baroni, M.; and Bouchacourt, D. 2020. Permutation Equivariant Models for Compositional Generalization in Language. In *International Conference on Learning Representations*.

Guss, W. H.; Houghton, B.; Topin, N.; Wang, P.; Codel, C.; Veloso, M.; and Salakhutdinov, R. 2019. MineRL: a large-scale dataset of minecraft demonstrations. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 2442–2448.

Hafner, D.; Pasukonis, J.; Ba, J.; and Lillicrap, T. 2023. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*.

Jayakumar, S. M.; Czarnecki, W. M.; Menick, J.; Schwarz, J.; Rae, J.; Osindero, S.; Teh, Y. W.; Harley, T.; and Pascanu, R. 2019. Multiplicative interactions and where to find them. In *International conference on learning representations*.

Johnson, M.; Hofmann, K.; Hutton, T.; and Bignell, D. 2016. The Malmo Platform for Artificial Intelligence Experimentation. In *International Joint Conference on Artificial Intelligence*.

Kaelbling, L. P. 1993. Learning to achieve goals. In *IJCAI*, volume 2, 1094–8. Citeseer.

Keysers, D.; Schärli, N.; Scales, N.; Buisman, H.; Furrer, D.; Kashubin, S.; Momchev, N.; Sinopalnikov, D.; Stafiniak, L.; Tihon, T.; et al. 2019. Measuring compositional generalization: A comprehensive method on realistic data. *arXiv preprint arXiv:1912.09713*.

Kim, N.; and Linzen, T. 2020. COGS: A compositional generalization challenge based on semantic interpretation. *arXiv preprint arXiv:2010.05465*.

Lake, B. M. 2019. Compositional generalization through meta sequence-to-sequence learning. *Advances in neural information processing systems*, 32.

Lake, B. M.; and Baroni, M. 2017. Generalization without Systematicity: On the Compositional Skills of Sequence-to-Sequence Recurrent Networks. In *International Conference on Machine Learning*.

Lake, B. M.; and Baroni, M. 2023. Human-like systematic generalization through a meta-learning neural network. *Nature*, 623(7985): 115–121.

Lifshitz, S.; Paster, K.; Chan, H.; Ba, J.; and McIlraith, S. A. 2023. STEVE-1: A Generative Model for Text-to-Behavior in Minecraft. *ArXiv*, abs/2306.00937.

Liu, M.; Zhu, M.; and Zhang, W. 2022. Goal-Conditioned Reinforcement Learning: Problems and Solutions. *ArXiv*, abs/2201.08299.

Mallya, A.; Davis, D.; and Lazebnik, S. 2018. Piggyback: Adapting a Single Network to Multiple Tasks by Learning to Mask Weights. In *European Conference on Computer Vision*.

Nasiriany, S.; Pong, V.; Lin, S.; and Levine, S. 2019. Planning with goal-conditioned policies. *Advances in Neural Information Processing Systems*, 32.

Perez, E.; Strub, F.; De Vries, H.; Dumoulin, V.; and Courville, A. 2018. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Ruis, L.; Andreas, J.; Baroni, M.; Bouchacourt, D.; and Lake, B. M. 2020. A benchmark for systematic generalization in grounded language understanding. *Advances in neural information processing systems*, 33: 19861–19872.

Russin, J.; Jo, J.; O'Reilly, R. C.; and Bengio, Y. 2019. Compositional generalization in a deep seq2seq model by separating syntax and semantics. *ArXiv*, abs/1904.09708.

Schaul, T.; Horgan, D.; Gregor, K.; and Silver, D. 2015. Universal value function approximators. In *International conference on machine learning*, 1312–1320. PMLR.

Scott, W. R. 2012. *Group theory*. Courier Corporation.

Wang, G.; Xie, Y.; Jiang, Y.; Mandlekar, A.; Xiao, C.; Zhu, Y.; Fan, L. J.; and Anandkumar, A. 2023a. Voyager: An Open-Ended Embodied Agent with Large Language Models. *ArXiv*, abs/2305.16291.

Wang, Z.; Cai, S.; Liu, A.; Jin, Y.; Hou, J.; Zhang, B.; Lin, H.; He, Z.; Zheng, Z.; Yang, Y.; Ma, X.; and Liang, Y. 2023b. JARVIS-1: Open-World Multi-task Agents with Memory-Augmented Multimodal Language Models. *ArXiv*, abs/2311.05997.

Wang, Z.; Cai, S.; Liu, A.; Ma, X.; and Liang, Y. 2023c. Describe, Explain, Plan and Select: Interactive Planning with Large Language Models Enables Open-World Multi-Task Agents. *ArXiv*, abs/2302.01560.

Yin, Y.; Zeng, J.; Li, Y.; Meng, F.; Zhou, J.; and Zhang, Y. 2023. Consistency Regularization Training for Compositional Generalization. In *Annual Meeting of the Association for Computational Linguistics*.

Yuan, H.; Zhang, C.; Wang, H.; Xie, F.; Cai, P.; Dong, H.; and Lu, Z. 2023. Skill Reinforcement Learning and Planning for Open-World Long-Horizon Tasks.

Zhao, L.; Kong, L.; Walters, R.; and Wong, L. L. S. 2022. Toward Compositional Generalization in Object-Oriented World Modeling. In *International Conference on Machine Learning*.

Zhou, A.; Kumar, V.; Finn, C.; and Rajeswaran, A. 2022. Policy Architectures for Compositional Generalization in Control. *ArXiv*, abs/2203.05960.