Compositional Learning of Visually-Grounded Concepts Using Reinforcement

Zijun Lin[*1], **Haidi Azaman**[*2], **M Ganesh Kumar**[3], **Cheston Tan**[3]

[1]Nanyang Technological University, [2]National University of Singapore, [3]Centre for Frontier AI Research, A*STAR
linz0048@e.ntu.edu.sg, e0540498@u.nus.edu, m_ganeshkumar@u.nus.edu, cheston-tan@i2r.a-star.edu.sg

Abstract

Deep reinforcement learning agents need to be trained over millions of episodes to decently solve navigation tasks grounded to instructions. Furthermore, their ability to generalize to novel combinations of instructions is unclear. Interestingly however, children can decompose language-based instructions and navigate to the referred object, even if they have not seen the combination of queries prior. Hence, we created three 3D environments to investigate how deep RL agents learn and compose color-shape based combinatorial instructions to solve novel combinations in a spatial navigation task[1]. First, we explore if agents can perform compositional learning, and whether they can leverage on frozen text encoders (e.g. CLIP, BERT) to learn word combinations in fewer episodes. Next, we demonstrate that when agents are pretrained on the shape or color concepts separately, they show a 20× decrease in training episodes needed to solve unseen combinations of instructions. Lastly, we show that agents pretrained on concept and compositional learning achieve significantly higher reward when evaluated zero-shot on novel color-shape1-shape2 visual object combinations. Overall, our results highlight the foundations needed to increase an agent's proficiency in composing word groups through reinforcement learning and its ability for zero-shot generalization to new combinations.

Introduction

Hierarchical reinforcement learning agents can learn several action policies and compose them together to solve complex tasks (Barto and Mahadevan 2003). However, model-free end-to-end algorithms are inefficient learners, requiring millions of training episodes for convergence and incur high number error trials in the initial stages of learning (Mirowski et al. 2016; Arulkumaran et al. 2017), making them undeployable in the real world (Gervet et al. 2023). Robotics has been trying to solve the vision-language-action combined representation space using symbolic methods but work to integrate vision, language and action spaces using neural networks for compositional learning has only recently started (Roy et al. 2021; Bisk et al. 2020).

Additionally, children are sometimes treated as benchmarks for general intelligence (Ye et al. 2022; Dupoux 2018; Gandhi et al. 2021). As part of their cognitive development, children learn individual concepts (Macario 1991; Gelman and Markman 1986) or schemas (Piaget 1976; Rumelhart and Ortony 1977) by associating visual and verbal cues, and interacting with their environment (Bisk et al. 2020; Iverson 2010). Learning concepts in the vision-language-action space subsequently facilitates rapid learning on higher order compositional tasks (Bartlett and Bartlett 1995; Kumar et al. 2023; McClelland 2013).

Hence, we developed a 3D environment so that multimodal RL agents can learn to associate color and shape based visual features to language-based instructions by navigating to the correctly referenced object for a reward. The agent learns a set of color-shape combined instructions, and is evaluated on an unseen compositional test set to determine its ability to decompose novel instructions and combine prior learned features in zero-shot. Besides training agents from scratch, we trained agents with frozen text encoders (CLIP, BERT) to determine if semantic knowledge learned from static datasets reduces the training episodes needed for instruction acquisition to solve the train and test combinatorial task. Lastly, we pretrained agents to learn color and shape concepts separately to demonstrate a 20× reduction in the number of episodes needed to solve the unseen color-shape task, and the zero-shot ability to solve a novel color-shape1-shape2 task. We conclude by showing that the type of pretraining to ground agents on the vision-language-action space influences the rate of combinatorial learning.

To summarize, our contributions are as follows:

- We demonstrated the capability of RL agents to decompose concepts and generalize to unseen color-shape combinations, with certain pretrained language encoders improving the learning performance.
- We grounded RL agents on shape and color word groups separately to demonstrate the rapid learning performance on the color-shape compositional learning task. Results show 100× and 20× improvement in train and test combinations respectively.
- We show for the first time the advantages of pretraining on concept learning tasks in order to solve increasingly complex compositional learning tasks through zero-shot navigation test experiments.

[*]These authors contributed equally.

[1]https://github.com/haidiazaman/RL-concept-learning-project contains the environments and codes

Related Work

Using the environment to ground visuo-language learning

Children pedagogy such as Montessori and Reggio emphasizes on scaffolding children development using the environment. Specifically, the environment should be safe for free exploration and contain sufficient resources for children to engage in the various stages of play e.g. solitary, parallel, cooperative, etc. (Parten 1932) while grounding learning experiences to visual and language features.

Similarly, there has been an increase in virtual learning environments to emulate the scaffolding conditions for artificial cognitive development.

The simplest form of environment is a two-dimensional continuous arena with boundaries where the agent has to learn state-specific navigation policies to successfully reach targets. The agent perceives it's location using place cells and the target to navigate to is given by simple instructions such as a one-hot vector encoding sensory cue (Kumar et al. 2022, 2023). Due to the simplified task and model requirements, it is easy to understand the learned policies, though hard to transfer to natural conditions.

A more complex but naturalistic environment is a three-dimensional arena where the agent perceives environmental features using RGB visual inputs, and instructions are represented by either one-hot vectors (Hill et al. 2020) or English sentences (Hill et al. 2021). The goal is to navigate to the instructed target. Recent work called XL, allowed the generation of millions of learning environments by changing three variables - the physical 3D space, the game rule specified using natural language, and the number of co-players (Team et al. 2021). Training RL agents on vastly diverse tasks led to impressive generalization behaviour on held-out tasks, requiring them to navigate, use tools and cooperate or compete depending on the instruction to maximize total rewards.

Alternatively, agents can learn to move objects to satisfy the language-based instruction (Jiang et al. 2019). Here, the action space is much larger than a navigation task as the agent has to learn to manipulate objects to satisfy the corresponding language query. Hence, the visual features are simplified to ensure the learning speed is tractable.

To our knowledge, there is only one environment that was developed to emulate the learning environment for children (Ye et al. 2022), though its ability to scaffold concept learning using reinforcement learning has not been tested yet.

Models for compositional learning

Compositional learning is the ability to decompose large information into its basic elements or concepts, and subsequently combine these concepts to solve novel, unseen combinations in zero-shot or with few examples (Lake, Salakhutdinov, and Tenenbaum 2015; Xu, Kordjamshidi, and Chai 2021; De Beule and Bergen 2006; Zuberbühler 2018; Xia and Collins 2021; Ji et al. 2022).

Compositional learning has mostly been used to improve object detection where vision-based models learn object-attribute pairs in a training set and compose the invarianves learned to an unseen test set (Kato, Li, and Gupta 2018; Hou et al. 2021; Purushwalkam et al. 2019; Anwaar, Labintcev, and Kleinsteuber 2021). Alternatively, the loss function can be augmented to encourage deep networks to decompose information into generalizable features (Stone et al. 2017; Tolooshams et al. 2020). More recently, models can accurately identify or parse relevant objects from images using bounding boxes(Lee, Kumar, and Tan 2023) or object segmentation masks (Kirillov et al. 2023) to solve new task requirements.

Symbolically parsing sentences into individual words and parts-of-speech tagging helps to decompose the sentence and clarify the semantics. Deep networks such as BERT represent concepts with similar or opposite semantics through its embedding (Devlin et al. 2018; Jawahar, Sagot, and Seddah 2019) while models trained on large datasets demonstrate compositional reasoning with good performance (Rytting and Wingate 2021).

There have been numerous works on learning atomic actions and composing them to rapidly solve new tasks (Barto and Mahadevan 2003). Some of such works are termed hierarchical reinforcement learning where models learn various policies at different levels of control for efficient policy composition (Kulkarni et al. 2016) such as for tool use (Team et al. 2021) or sequential navigation to goals (Han, Doya, and Tani 2020).

Multi-modal models learn to align visual inputs to language concepts (Radford et al. 2021; Thrush et al. 2022; Ma et al. 2023; Yuksekgonul et al. 2022) to solve a multitude of compositional reasoning tasks (Lu et al. 2023) such as Visual-Question Answering (VQA) (Johnson et al. 2017), Referring Expressions (Lee, Kumar, and Tan 2023), or augment images using instructions (Gupta and Kembhavi 2023).

Yet, there are only a handful of work to align compositional learning in the vision, language and action spaces. A notable example includes training vision-language based reinforcement learning agents on millions of various toy environments to develop a generally capable agent that can solve tasks based on the instruction given (Team et al. 2021).

Nevertheless, how these reinforcement learning models ground vision-language-action representations for compositional learning, what the individual concepts are, and how these concepts are recomposed to solve novel combinations remains elusive.

Color & Shape Learning Environment

The following section describes the learning environment used to ground the vision-language reinforcement learning agent on geometric Shapes (S) and Colors (C).

Environmental design

The 3D environments were developed to learn two key concepts: Shape (S) and Color (C). The environments contain objects made up of five distinct shapes, which are capsule, cube, cylinder, prism, and sphere, paired with five different colors: red, green, blue, yellow, and black. Hence, each object can be described using the shape and color, for example "red sphere", "blue capsule", or "yellow prism". The environment engine utilizes the available shapes and colors to

generate episode variations featuring different object combinations in a random manner to train the agent.

A target object will be randomly spawned at one of four predetermined locations within a rectangular room. The overall layout of the environment remains constant, as depicted in Figure 1. The room includes fixed visual cues namely a door, window, shelf and reference man.

A Unity-based camera models the agent's first person point of view by capturing the environmental dynamics in terms of RGB images. These serve as the visual input for the RL agent. The environment also produces the label of the target object such as "red cube" or "blue sphere". The instruction will be passed to the RL agent as language input.

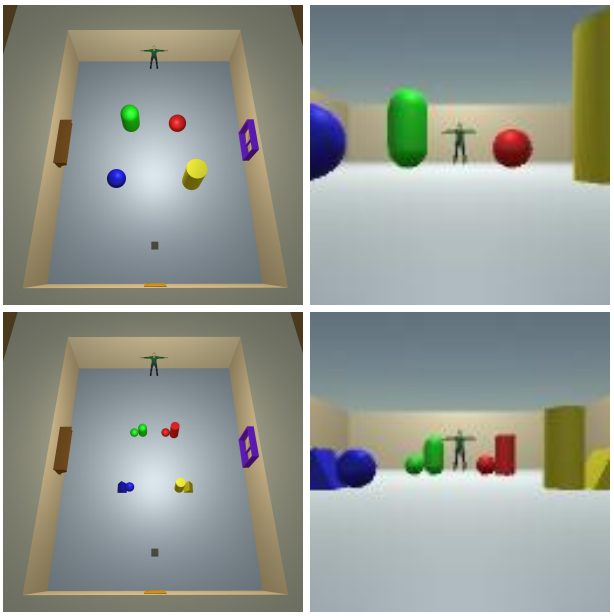

Figure 1: Two example environments. Left column shows the top-view of the environments. Right column shows the first-person view (128x128) of the RL agent. Top and bottom rows show the C+S environment (target instruction is "red sphere") and the C+S+S environment (target instruction is "red sphere cylinder") respectively. Notice how the sizes and locations of the objects differ (compare top and bottom).

Task The primary objective of our RL agent is to navigate to the target object described by the language-based instruction. We devised three different environments to investigate the agent's ability to decompose instructions and learn foundational concepts. The key details of the three environments are:

1. C+S ("Color AND Shape"): The task is to compositionally learn two word concepts. There is one target described by the color and shape attribute with three non-target objects. The instruction includes both color and shape words as inputs.
2. C+S+S ("Color Shape Shape"): The task is to compositionally learn three-word concepts. There is one target with two shapes and three non-targets with pairs of shapes. Both objects in a pair will be of the same color. The instruction includes all three color and two shape words as inputs.
3. C/S ("Color" OR "Shape"): The task is single word concept learning. There is one target and three non-target objects. Only one word, either color or shape, that is given as instruction.

Figure 1 shows examples of the C+S (top row) and C+S+S (bottom row) environments. For all three environments, one target and three non-target objects or object pairs are randomly spawned at the four predefined positions in the environment.

Moreover, to evaluate the agent's performance, we also created train and test combinatorial instructions for the three environments. The train-test splits for environments C/S, C+S and C+S+S are detailed in the Supplementary material. The train-test split allows us to examine whether agents can learn to decompose instructions on which it was trained on and combine the Color and Shape concepts to solve unseen combinations in zero-shot during testing.

Evaluation Metrics

Throughout each episode, the agent's actions disburses specific rewards and penalties. A successful navigation to the target object yields a reward of +10, while collisions with non-target objects or walls incur penalties of -3 and -1, respectively. Additionally, the agent receives a penalty of -10 upon reaching the maximum allowed steps of 500. To ascertain the successful learning of a task by the agent, we establish a Performance Criterion of $+9$. The agent is deemed to have effectively learned the task when it accumulates a reward ≥ 9 over 100 training episodes.

Grounded Concept Learning Agent

One-Hot Agent Architecture

As illustrated in Figure 2, the One-hot agent architecture is modified from (Hill et al. 2020), which takes in visual inputs (a $3\times128\times128$ tensor of RGB pixel values) and language input (one-hot vector embedding). RGB pixel values are passed into the vision module, which contains three convolutional layers, and the output flattened into a 3136 (a $64\times7\times7$ tensor) dimensional embedding.

Simultaneously, the language module takes in two one-hot vector embedding of the instructions, each representing the Color and Shape attributes. For example, "red cube sphere" is represented by a one-hot vector ([1,0,0,0,0]) and a two-hot vector ([0,1,0,0,1]). The two vectors are fully connected to a 128 unit linear embedding layer.

The 3136 dimensional vector from the vision module and the 128 dimensional vector from the language modules are concatenated and fed into a 256 dimensional linear mixing layer.

An Long Short Term Memory (LSTM) takes the 256-dimensional embedding as input. Its hidden state output s_t is linked to both the action predictor (actor) and value estimator (critic). The action predictor maps the LSTM's hidden state tensor to a probability distribution $\pi(a_t|s_t)$ over

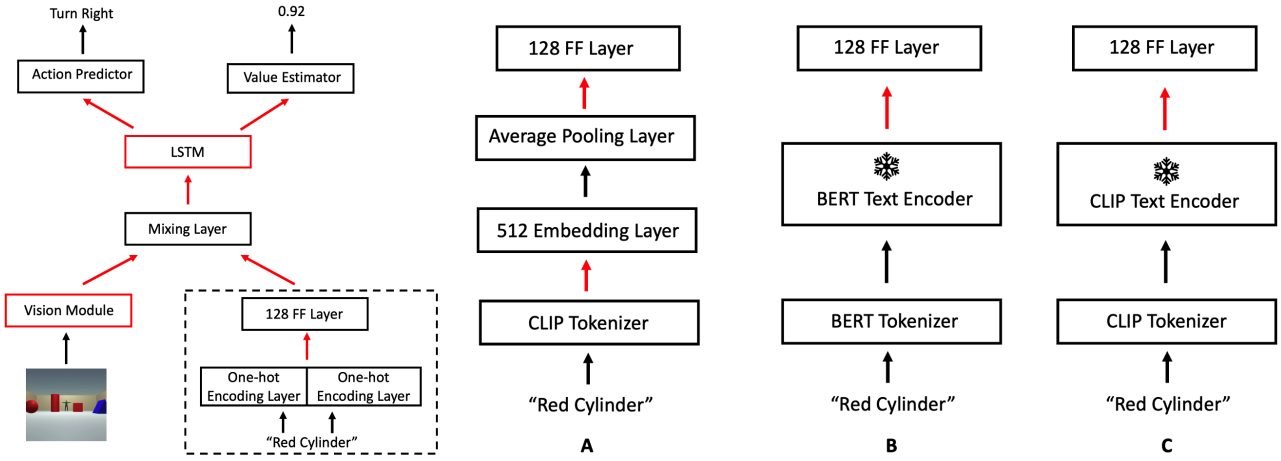

Figure 2: Agent architecture. The language module of the One-hot encoder agent is demarcated with dotted lines, and the language model of the agent with: **A**) Vanilla text encoder, **B**) BERT text encoder, **C**) CLIP text encoder are shown respectively. BERT text encoder and CLIP text encoder are both pretrained and frozen. Red arrows or boxes represent trainable weights, black arrows or boxes represent frozen weights.

four possible actions, i.e., move forward, move backward, turn left and turn right. Meanwhile, the value estimator computes a scalar approximation of the agent's state-value function $V(s_t)$.

The agent is trained using the advantage actor-critic (A2C) algorithm (Kumar et al. 2021). We utilize the RMSProp optimizer with a consistent learning rate of 2.5×10^{-4} across all experiments to optimize the training process.

Text Encoders

To study the influence of different text encoders on learning word combinations, we introduce three variants: Vanilla, CLIP, and BERT, whose architectures are depicted in Figure 2 respectively. The output dimensions of three text encoders are standardized to 128 for fair comparison. Instructions are represented using text strings and passed to the language modules. Importantly, the other parts of the agent is unchanged to be consistent with the One-hot agent.

Vanilla Text Encoder The text instruction firstly undergoes tokenization using the original CLIP tokenizer, with a maximum tokenized tensor length of 77 (Radford et al. 2021). The tokenized tensor is then passed through a single token embedding layer, retrieving 512-dimensional word embeddings for each token. Subsequently, an average pooling operation computes the mean value of each 512-dimensional word embedding, yielding a 77-dimensional tensor. This tensor is passed to the 128-dimensional linear layer. Importantly, the parameters of the Vanilla text encoder are initialized randomly and updated during training.

BERT Text Encoder The text instruction undergoes tokenization using the BERT tokenizer before feeding it into the pre-trained BERT Text Encoder (Devlin et al. 2019). The output from the BERT Text Encoder is a 768-dimensional vector, which is then passed to the 128-dimensional word embedding linear layer to obtain. Here, the weights of the BERT Text Encoder are kept frozen while only the weights to the 128-dimensional embedding layer is updated during training.

CLIP Text Encoder Similar to the Vanilla text encoder, the text instruction is tokenized using the CLIP tokenizer first and passed into the pre-trained CLIP Text Encoder. The resulting 512-dimensional output from the CLIP text encoder is then passed to the 128-dimensional word embedding linear layer. Similar to BERT, the parameters of the CLIP text encoder remains fixed while only the weights to the 128-dimensional embedding layer is updated during training.

Results

In the first experiment, we train the four text-encoder variants of reinforcement learning agent (Fig. 2) to navigate to the target object described by the color AND shape attributes (Fig. 1 top row). We evaluate its color-shape compositional learning capabilities based on the number of episodes needed to reach performance criterion on the 20 training combinations and five unseen test combinations (refer to Supplementary material).

In the second experiment, we demonstrate that if agents are pretrained to learn the color and shape concepts separately, the number of episodes needed for color-shape compositional learning is significantly reduced. We further compare compositional learning with and without concept learning in terms of zero-shot navigation in the novel color-shape1-shape2 environment.

Experiment 1: Compositional reinforcement learning with various text encoders

Visually grounded agents can understand single feature instructions (Hill et al. 2020). How navigation agents learn and compose multiple attributes is unclear. Hence, experiment 1

expands on single attribute navigation to the combination of two attributes, Color + Shape (C+S).

	Training Episodes	
Text Encoder	**Train combinations**	**Test combinations**
One-hot	72.4 ± 1.6	97.1 ± 2.4
Vanilla	124.9 ± 11.2	187 ± 18.9
BERT	106.2 ± 10.1	≥ 200
CLIP	58.6 ± 5.5	74.6 ± 8.0

Table 1: Init $\rightarrow$ C+S: Learning to decompose instructions for compositional performance. Values in the table indicate the mean of episodes (in **thousands**) needed to achieve an average episode reward ≥ 9 (performance criterion) over 100 episodes with standard deviation for two iterations in training and testing environments. Lower values indicate faster learning performance.

In this experiment, the four agent variants are trained on 20 C+S instructions and tested on 5 unseen or held-out C+S pairs. For example, the agent is trained on the instructions and visual targets "black cube" and "red sphere". After 100 training episodes, it is tested on its ability to compose the concepts of "black" and "sphere" to accurately navigate to the visual target "black sphere" when given the held-out test combination instruction.

Table 1 shows the number of training episodes needed for agents to achieve performance criterion on both the train and test combinations. To clarify, agents continue to train on the 20 training combinations to determine when they can achieve performance criterion on the unseen test combination. The intent is to understand if and when RL agent learns to decompose instructions given during training to learn each word group and recompose them to solve unseen color-shape test combinations. All the trainable parameters were randomly initialized and each agent variant was trained for 2 iterations ($N = 2$) on the C+S environment to obtain the mean episodes needed to achieve performance criterion.

The agent with the One-hot text encoder requires approximately 72,000 and 97,000 episodes to achieve performance criterion for the 20 training and 5 unseen test combinations respectively. This result highlights that an agent can learn to decompose the compositional instructions and recompose the individual concepts to generalize to unseen test combinations.

The Vanilla text encoder that was trained from scratch required almost two times of the training episodes of the One-hot encoder to achieve performance criterion for the training combination. This is expected since the Vanilla encoder has to learn the semantic differences of the color and shape attributes from scratch. However, the text encoder used CLIP's tokenizer which encoded the various colors and shapes into distinct tokens, making it easier for the Vanilla text encoder to disentangle the embeddings even before training (refer to Supplementary material).

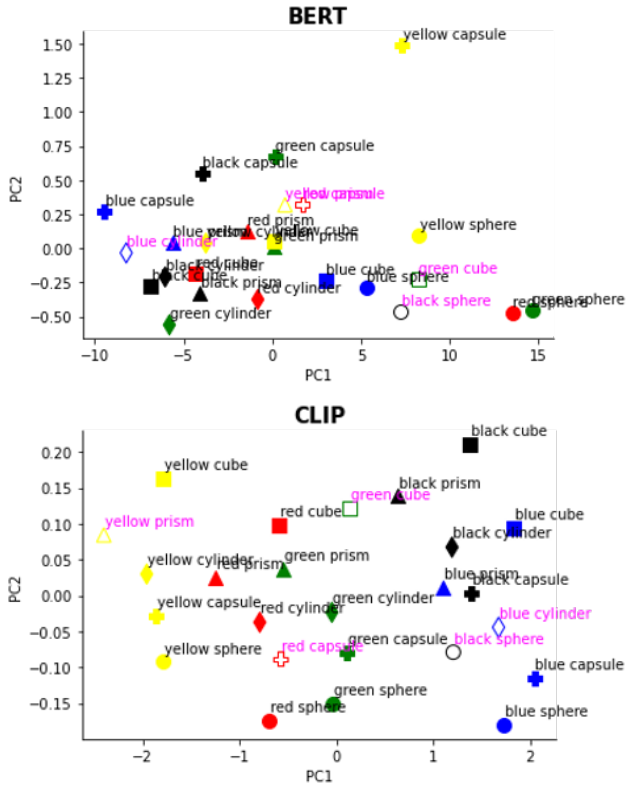

Figure 3: Word embeddings of the agent with BERT (top) and CLIP (bottom) text encoders after training in the C+S environment for 50,000 episodes. Filled icons represent training set examples, while unfilled icons with magenta labels represent testing set examples.

Interestingly, although the agent with the BERT text encoder was trained for a maximum of 200,000 episodes, the agent achieved the average reward of 8.5 approximately for testing combinations, which failed to reach performance criterion in the unseen test combinations. Figure 3 shows that after training for 50,000 episodes, BERT's word embedding of the 20 train and 5 test instructions are highly overlapped, suggesting that the pretrained BERT encoder struggles to distinguish between the color and shape concepts. A potential explanation is because BERT is only trained on text data and is not grounded to visual attributes, making it difficult to disambiguate visual concepts like color and shape.

The CLIP text encoder achieved performance criterion on both the train and unseen test combinations 1.3 times faster than the One-hot text encoder agent. The expedited learning implies that CLIP's prelearned word embedding is useful to increase the training efficiency for a reinforcement learning agent. Figure 3 shows that CLIP encodes both the train and test instructions systematically where Principal Component 1 and 2 clearly explains the color and shape dimensions with minimal overlap between the 25 instructions. Such organization of color and shape concepts could be because CLIP's language comprehension is grounded to visual attributes due to its cross-modal training regime on various image-caption datasets. This result demonstrates that it is possible to use

vision-language grounded models to improve the training efficiency of multi-modal reinforcement learning agents, especially for compositional learning.

Experiment 2: Concept learning speeds up Compositional learning

We have demonstrated that RL agents are capable of learning the Color and Shape word groups by decomposing composed instructions and, subsequently recompose them to solve unseen combinations of word groups. This investigation is extended by showcasing that prior learning of the individual Color and Shape word concepts (Concept learning) can serve as a schema to enhance the agents' proficiency in solving compositional instructions (Compositional learning). For this set of experiments, we focus only on the One-hot text encoder agent.

Table 2 compares the number of training episodes needed to achieve performance criterion on the 20 train and five unseen test combinations in the C+S environment for N=3 iterations. Notably, two types of training regimes are compared. The first row indicates randomly initialized One-hot text encoder agents being directly trained on the C+S environment as part of compositional learning. The second row describes randomly initialized One-hot text encoder agents that are pretrained on the C/S environment for 200,000 episodes to learn the Color and Shape individually as part of concept learning. Subsequently, they are trained on the C+S environment for compositional learning.

Training environment	Train combinations	Test combinations
C+S	67.4 ± 7.2	94.8 ± 3.7
C/S → C+S	0.6 ± 0.1	5.5 ± 2.9

Table 2: Mean and standard deviation of the number of episodes (in **thousands**) required to achieve an average episode reward ≥ 9 (performance criterion) over 100 episodes over three iterations. These experiments were ran in C+S training and testing environments, where agents are trained from scratch, versus after pretraining on C/S. Lower values indicate faster learning performance.

Agents pretrained on C/S for concept learning achieve performance criterion on the train and unseen test combinations 100× and 20× faster than the agents trained only on compositional learning, respectively. Through concept learning, agents learn shape and color invariances in C/S, allowing the RL agents to quickly learn to encode and compose combined instructions in C+S environment. These results demonstrate the training speed-up when learning individual concepts before learning higher order tasks such as composing concepts.

To further demonstrate the benefit of concept learning, we evaluated the One-hot text encoder agents on their zero-shot navigation capabilities on two environments and instruction types, C+S and C+S+S. Table 3 summarises two sets of results.

In the first set, we evaluated randomly initialised agents that were not trained on any environment (nil) and agents trained on C/S on familiar and unseen color-shape combinations in the C+S training and testing environments respectively. As expected, randomly initialised agents perform poorly on both familiar and unseen combinations, achieving large negative reward values. Conversely, agents trained on C/S perform considerably well for zero-shot performance and even achieve close to the Performance Criterion of +9 for familiar combinations. These results further support the importance of concept learning where agents can make reasonable inferences of the color-shape target even though they were not trained to compose instructions.

	Zero-shot Evaluation	
Training environment	**Familiar C+S combo**	**Unseen C+S combo**
nil	-29.15 ± 3.07	-36.48 ± 3.60
C/S	8.37 ± 0.64	6.74 ± 1.64
Training environment	**Familiar C+S+S combo**	**Unseen C+S+S combo**
nil	-24.42 ± 1.29	-23.42 ± 2.57
C/S	1.19 ± 1.24	-4.02 ± 2.44
C+S	2.84 ± 0.92	-5.10 ± 2.59
C/S → C+S	5.49 ± 0.26	5.55 ± 0.39

Table 3: This table summarises the different zero-shot experiments. The values are the mean and standard deviation of the cumulative rewards of the agents over 100 episodes in the training and testing environments, across three iterations. Higher reward values indicate better performance.

Up till now, agents have only been trained and evaluated on color-shape instruction combinations. However, in the real-world setting, instructions are highly compositional, extending beyond two word color-shape attributes. Furthermore, real-world objects can be thought of two or more geometric shapes composed together such as a hammer being composed of a cylinder and a cuboid.

We evaluated agents trained either on concept learning (C/S), compositional learning (C+S), or both (C/S → C+S) on the C+S+S environment to determine if these agents could generalize in zero-shot to a completely novel category of instructions and visual objects found in C+S+S. To solve the task in this environment, the RL agents have to compose three word instruction, one color and two shapes, which makes the task more complex and difficult than C+S. Since there is some overlap between the train combinations in C+S and the combinations in C+S+S, we call these objects and instructions as familiar combinations while combinations that do not overlap with the train combinations are called unseen combinations, making them truly novel composed objects and instructions. There are 40 and 10 color-shape1-shape2 familiar and unseen combinations respectively.

Table 3 shows that agents trained only on C/S performs slightly worse on the familiar C+S+S combinations but better on the unseen C+S+S combinations compared to the agent trained only on C+S. The best performing agent is the one that was trained on C/S first and then C+S, achieving a

rewards of 5.5 on both the familiar and unseen combinations in zero-shot.

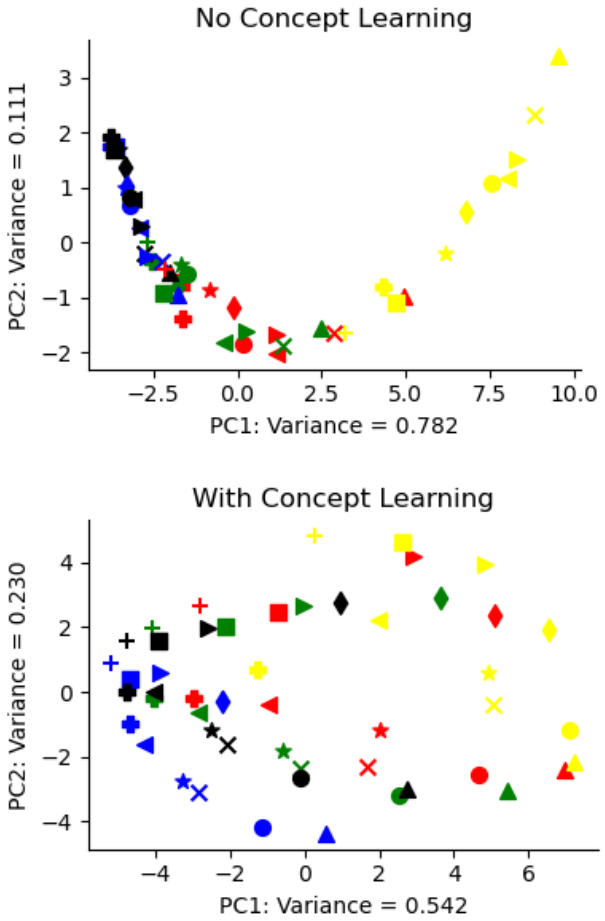

Figure 4: Embedding of average LSTM activity for 50 instructions from C+S+S after training only on C+S (Top), compared to pretraining on C/S and subsequently training on C+S (Bottom).

Figure 4 demonstrates the difference in embedding the 40 familiar and 10 unseen combinations by the LSTM layer. When an agent does not learn individual concepts and directly learns to compose (C+S), the 50 novel instructions are mapped onto just two principal components which can explain 90% of the variance in the LSTM activity. PC1 might be sufficient to separate color information but PC2 struggles to encode the two different shape dimensions. This results in a highly overlapped embedding, making it difficult for the agent to disambiguate instructions and solve the new C+S+S combinations.

Conversely, when an agent is first trained on the color and shape concepts individually and subsequently learn to compose them, the agent maps the 50 different C+S+S combinations onto a much higher dimensional space, requiring at least six principal components to explain 90% of the variance in the LSTM.

Still, there is some organization of the 50 novel combinations in the higher dimensional embedding space. For instance, upright triangle symbol representing "Color prism sphere" combinations are found mostly in the bottom right subspace and plus symbols representing "Color capsule cylinder" are found in the top left subspace. With this organization, agents can leverage on the clustering of similar combinations to achieve the zero-shot navigation results seen in Table 3.

These results highlight the importance of pairing concept learning with compositional learning to maximize the zero-shot capabilities of reinforcement learning agents, even in a completely novel environment.

Limitations and Future work

The 3D environments developed for this study utilised relatively basic geometric shapes such as "capsule" and "prism". This raises questions about generalization and zero-shot navigation on real-world tasks involving more realistic objects. Additionally, without any obstacles included in the room, the agents' navigation task is fairly simple. As such, an agent trained in our environments may struggle to generalize to other scenarios which require the ability to navigate more complex routes.

In the concept learning experiment, we only compared the performance of the One-hot agent across different training and testing environments. As such, we did not provide insights on whether pretrained and frozen visual encoders can show similar increases in speed to solve compositional learning tasks. Furthermore, the language instructions for our experiments are only composed of two or three words. We did not experiment with more complex instructions such as "find any object yellow". These limitations will be explored as future work.

Conclusion

In this paper, we demonstrate that reinforcement learning agents are able to ground instructions to visual attributes. Specifically, we explore if agents can learn to decompose instructions during training and are able to recompose different word groups together to solve unseen object combinations. Furthermore, we demonstrate that concept learning of individual word groups accelerates compositional learning of combined word groups. Moreover, we evaluate the zero-shot capabilities of agents on complex compositional learning tasks after learning concepts individually.

This study has potential applications in various domains, including autonomous navigation and human-robot interactions require language and visuo-spatial reasoning skills. As we continue to bridge the gap between language and vision, our work on compositional learning in multi-modal reinforcement learning agents lay the groundwork for applications that require the dynamic interaction between artificial agents and their environments.

References

Anwaar, M. U.; Labintcev, E.; and Kleinsteuber, M. 2021. Compositional learning of image-text query for image retrieval. In *Proceedings of the IEEE/CVF Winter conference on Applications of Computer Vision*, 1140–1149.

Arulkumaran, K.; Deisenroth, M. P.; Brundage, M.; and Bharath, A. A. 2017. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6): 26–38.

Bartlett, F. C.; and Bartlett, F. C. 1995. *Remembering: A study in experimental and social psychology*. Cambridge university press.

Barto, A. G.; and Mahadevan, S. 2003. Recent advances in hierarchical reinforcement learning. *Discrete event dynamic systems*, 13(1-2): 41–77.

Bisk, Y.; Holtzman, A.; Thomason, J.; Andreas, J.; Bengio, Y.; Chai, J.; Lapata, M.; Lazaridou, A.; May, J.; Nisnevich, A.; et al. 2020. Experience grounds language. *arXiv preprint arXiv:2004.10151*.

De Beule, J.; and Bergen, B. K. 2006. On the emergence of compositionality. In *The Evolution of Language*, 35–42. World Scientific.

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*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota.

Dupoux, E. 2018. Cognitive science in the era of artificial intelligence: A roadmap for reverse-engineering the infant language-learner. *Cognition*, 173: 43–59.

Gandhi, K.; Stojnic, G.; Lake, B. M.; and Dillon, M. R. 2021. Baby Intuitions Benchmark (BIB): Discerning the goals, preferences, and actions of others. *Advances in neural information processing systems*, 34: 9963–9976.

Gelman, S. A.; and Markman, E. M. 1986. Categories and induction in young children. *Cognition*, 23(3): 183–209.

Gervet, T.; Chintala, S.; Batra, D.; Malik, J.; and Chaplot, D. S. 2023. Navigating to objects in the real world. *Science Robotics*, 8(79): eadf6991.

Gupta, T.; and Kembhavi, A. 2023. Visual programming: Compositional visual reasoning without training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 14953–14962.

Han, D.; Doya, K.; and Tani, J. 2020. Self-organization of action hierarchy and compositionality by reinforcement learning with recurrent neural networks. *Neural Networks*, 129: 149–162.

Hill, F.; Clark, S.; Blunsom, P.; and Hermann, K. M. 2020. Simulating Early Word Learning in Situated Connectionist Agents. In *Annual Meeting of the Cognitive Science Society*.

Hill, F.; Tieleman, O.; Von Glehn, T.; Wong, N.; Merzic, H.; and Clark, S. 2021. Grounded language learning fast and slow. *ICLR*.

Hou, Z.; Yu, B.; Qiao, Y.; Peng, X.; and Tao, D. 2021. Detecting human-object interaction via fabricated compositional learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 14646–14655.

Iverson, J. M. 2010. Developing language in a developing body: The relationship between motor development and language development. *Journal of child language*, 37(2): 229–261.

Jawahar, G.; Sagot, B.; and Seddah, D. 2019. What does BERT learn about the structure of language? In *ACL 2019-57th Annual Meeting of the Association for Computational Linguistics*.

Ji, A.; Kojima, N.; Rush, N.; Suhr, A.; Vong, W. K.; Hawkins, R. D.; and Artzi, Y. 2022. Abstract visual reasoning with tangram shapes. *arXiv preprint arXiv:2211.16492*.

Jiang, Y.; Gu, S. S.; Murphy, K. P.; and Finn, C. 2019. Language as an abstraction for hierarchical deep reinforcement learning. *Advances in Neural Information Processing Systems*, 32.

Johnson, J.; Hariharan, B.; Van Der Maaten, L.; Fei-Fei, L.; Lawrence Zitnick, C.; and Girshick, R. 2017. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2901–2910.

Kato, K.; Li, Y.; and Gupta, A. 2018. Compositional learning for human object interaction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 234–251.

Kirillov, A.; Mintun, E.; Ravi, N.; Mao, H.; Rolland, C.; Gustafson, L.; Xiao, T.; Whitehead, S.; Berg, A. C.; Lo, W.-Y.; et al. 2023. Segment anything. *arXiv preprint arXiv:2304.02643*.

Kulkarni, T. D.; Narasimhan, K.; Saeedi, A.; and Tenenbaum, J. 2016. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in neural information processing systems*, 29.

Kumar, M. G.; Tan, C.; Libedinsky, C.; Yen, S.-C.; and Tan, A. Y. 2022. A nonlinear hidden layer enables actor–critic agents to learn multiple paired association navigation. *Cerebral Cortex*, 32(18): 3917–3936.

Kumar, M. G.; Tan, C.; Libedinsky, C.; Yen, S.-C.; and Tan, A. Y.-Y. 2021. One-shot learning of paired associations by a reservoir computing model with Hebbian plasticity. *arXiv preprint arXiv:2106.03580*.

Kumar, M. G.; Tan, C.; Libedinsky, C.; Yen, S.-C.; and Tan, A. Y.-Y. 2023. One-shot learning of paired association navigation with biologically plausible schemas. *arXiv preprint arXiv:2205.09710*.

Lake, B. M.; Salakhutdinov, R.; and Tenenbaum, J. B. 2015. Human-level concept learning through probabilistic program induction. *Science*, 350(6266): 1332–1338.

Lee, C.; Kumar, M. G.; and Tan, C. 2023. DetermiNet: A Large-Scale Diagnostic Dataset for Complex Visually-Grounded Referencing using Determiners. *ICCV*.

Lu, P.; Peng, B.; Cheng, H.; Galley, M.; Chang, K.-W.; Wu, Y. N.; Zhu, S.-C.; and Gao, J. 2023. Chameleon: Plug-and-play compositional reasoning with large language models. *arXiv preprint arXiv:2304.09842*.

Ma, Z.; Hong, J.; Gul, M. O.; Gandhi, M.; Gao, I.; and Krishna, R. 2023. CREPE: Can Vision-Language Foundation Models Reason Compositionally? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10910–10921.

Macario, J. F. 1991. Young children's use of color in classification: Foods and canonically colored objects. *Cognitive Development*, 6(1): 17–46.

McClelland, J. L. 2013. Incorporating rapid neocortical learning of new schema-consistent information into complementary learning systems theory. *Journal of Experimental Psychology: General*, 142(4): 1190.

Mirowski, P.; Pascanu, R.; Viola, F.; Soyer, H.; Ballard, A. J.; Banino, A.; Denil, M.; Goroshin, R.; Sifre, L.; Kavukcuoglu, K.; et al. 2016. Learning to navigate in complex environments. *arXiv preprint arXiv:1611.03673*.

Parten, M. B. 1932. Social participation among pre-school children. *The Journal of Abnormal and Social Psychology*, 27(3): 243.

Piaget, J. 1976. Piaget's theory.

Purushwalkam, S.; Nickel, M.; Gupta, A.; and Ranzato, M. 2019. Task-driven modular networks for zero-shot compositional learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 3593–3602.

Radford, A.; Kim, J. W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; Krueger, G.; and Sutskever, I. 2021. Learning Transferable Visual Models From Natural Language Supervision. arXiv:2103.00020.

Roy, N.; Posner, I.; Barfoot, T.; Beaudoin, P.; Bengio, Y.; Bohg, J.; Brock, O.; Depatie, I.; Fox, D.; Koditschek, D.; et al. 2021. From machine learning to robotics: challenges and opportunities for embodied intelligence. *arXiv preprint arXiv:2110.15245*.

Rumelhart, D. E.; and Ortony, A. 1977. The representation of knowledge in memory. *Schooling and the acquisition of knowledge*, 99: 135.

Rytting, C.; and Wingate, D. 2021. Leveraging the inductive bias of large language models for abstract textual reasoning. *Advances in Neural Information Processing Systems*, 34: 17111–17122.

Stone, A.; Wang, H.; Stark, M.; Liu, Y.; Scott Phoenix, D.; and George, D. 2017. Teaching compositionality to cnns. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 5058–5067.

Team, O. E. L.; Stooke, A.; Mahajan, A.; Barros, C.; Deck, C.; Bauer, J.; Sygnowski, J.; Trebacz, M.; Jaderberg, M.; Mathieu, M.; et al. 2021. Open-ended learning leads to generally capable agents. *arXiv preprint arXiv:2107.12808*.

Thrush, T.; Jiang, R.; Bartolo, M.; Singh, A.; Williams, A.; Kiela, D.; and Ross, C. 2022. Winoground: Probing vision and language models for visio-linguistic compositionality. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5238–5248.

Tolooshams, B.; Song, A.; Temereanca, S.; and Ba, D. 2020. Convolutional dictionary learning based auto-encoders for natural exponential-family distributions. In *International Conference on Machine Learning*, 9493–9503. PMLR.

Xia, L.; and Collins, A. G. 2021. Temporal and state abstractions for efficient learning, transfer, and composition in humans. *Psychological review*, 128(4): 643.

Xu, G.; Kordjamshidi, P.; and Chai, J. Y. 2021. Zero-shot compositional concept learning. *arXiv preprint arXiv:2107.05176*.

Ye, J.; Duan, J.; Yu, S.; Wen, B.; and Tan, C. 2022. ABCDE: An Agent-Based Cognitive Development Environment. *arXiv preprint arXiv:2206.04909*.

Yuksekgonul, M.; Bianchi, F.; Kalluri, P.; Jurafsky, D.; and Zou, J. 2022. When and Why Vision-Language Models Behave like Bags-Of-Words, and What to Do About It? In *The Eleventh International Conference on Learning Representations*.

Zuberbühler, K. 2018. Combinatorial capacities in primates. *Current Opinion in Behavioral Sciences*, 21: 161–169.

Supplementary Materials

A1: Text Encoder Variants Results

Figure 5 shows the number of episodes needed to reach performance criterion (average reward over 100 episodes ≥ 9.0) on the 20 training combinations and five unseen test combinations across four different text encoders: One-hot, Vanilla, BERT and CLIP. To depict the figures, we individually initialize two agents for each of the four text encoders and then compute the average of the reward during training phase from these agent pairs. The observation highlights a shared trend among all four agents, as they commence with initial average rewards of approximately -30 and steadily progress towards meeting the performance criterion.

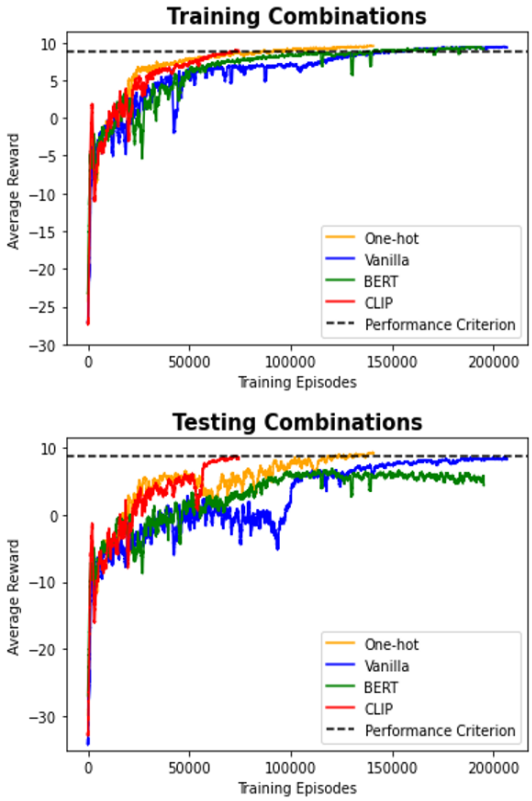

Figure 5: Learning curve of the agents with four different text encoders in training and testing combinations.

The learning curve aligns with the results shown in the Table 1 in the main paper. The agent with CLIP text encoder reaches the performance criterion at the swiftest rate, followed by One-hot text encoder, BERT and Vanilla text encoder in training combinations. However, in testing combinations, the agent with the BERT text encoder struggles to meet the performance criterion, even after undergoing more than 200,000 training episodes. The potential reason is that BERT is solely trained on the large number of textual data, rendering it less suitable for our multi-modal learning task.

A2: PCA of Word Embeddings

In our main paper, we presented the Principal Component Analysis (PCA) results of word embeddings from the BERT and CLIP text encoders after 50,000 episodes of training. In this section, we provide a comprehensive analysis of four distinct agents within the C+S environment. We examine their word embeddings before training, after 50,000 episodes, and upon reaching the performance criterion.

Figure 6 visually demonstrates that all the agents show the random distribution of word embeddings initially except Vanilla text encoder. The reason of Vanilla text encoder exhibits some pre-trained clusterings even before training is that it employs CLIP's tokenizer, effectively encoding the tokens for different colors and shapes. This, in turn, simplifies the process of segregating the word embeddings.

When the training episodes proceed to 50,000, the One-hot and CLIP text encoder begin to display noticeable separation among color groups and less overlaps compared to other text encoders. The figure aligns with our experimental results that the agent with the CLIP text encoder is the most efficient learner in the C+S environment, followed by the One-hot text encoder.

Notably, BERT text encoder still did not show the clear separation between different color groups even when the agent achieves the performance criterion in training combinations. This could be the reason why the agent with BERT text encoder failed to get the average episode reward of 9 in testing combinations with the training episodes of more than 200,000. However, the agents with other three text encoders all exhibit clear separation of colors and shapes along two axes in PCA plot for both training and testing combinations.

A3: Realistic Environment Mechanics

The learning environments were developed using the Unity 3D game engine and the Unity ML-Agents package. The interactions in the environments involve hits or collisions which are precisely managed by the Unity physics engine. This engine governs the physical interactions between the RL agent and the objects within the simulated 3D world. The integration of the Unity physics engine ensures a realistic and dynamic environment that aligns with real-world physics principles, enhancing the authenticity of the RL learning process.

A4: Tasks

The primary objective of our RL agent is to navigate to the target object described by the language-based instruction. We devised three variations of the environment to investigate the agent's ability to decompose instructions and learn foundational concepts. Figure 7 shows an example from these three environments.

The first environment is named "Color Shape"(C+S) as the agent is only rewarded when it navigates to the target that satisfies both color and shape descriptions, and is penalised if it navigates to any of the other three objects that either have different color or shape attributes. For example, when the instruction given to the agent is "red cube", the agent has to learn to navigate to the corresponding object that has the

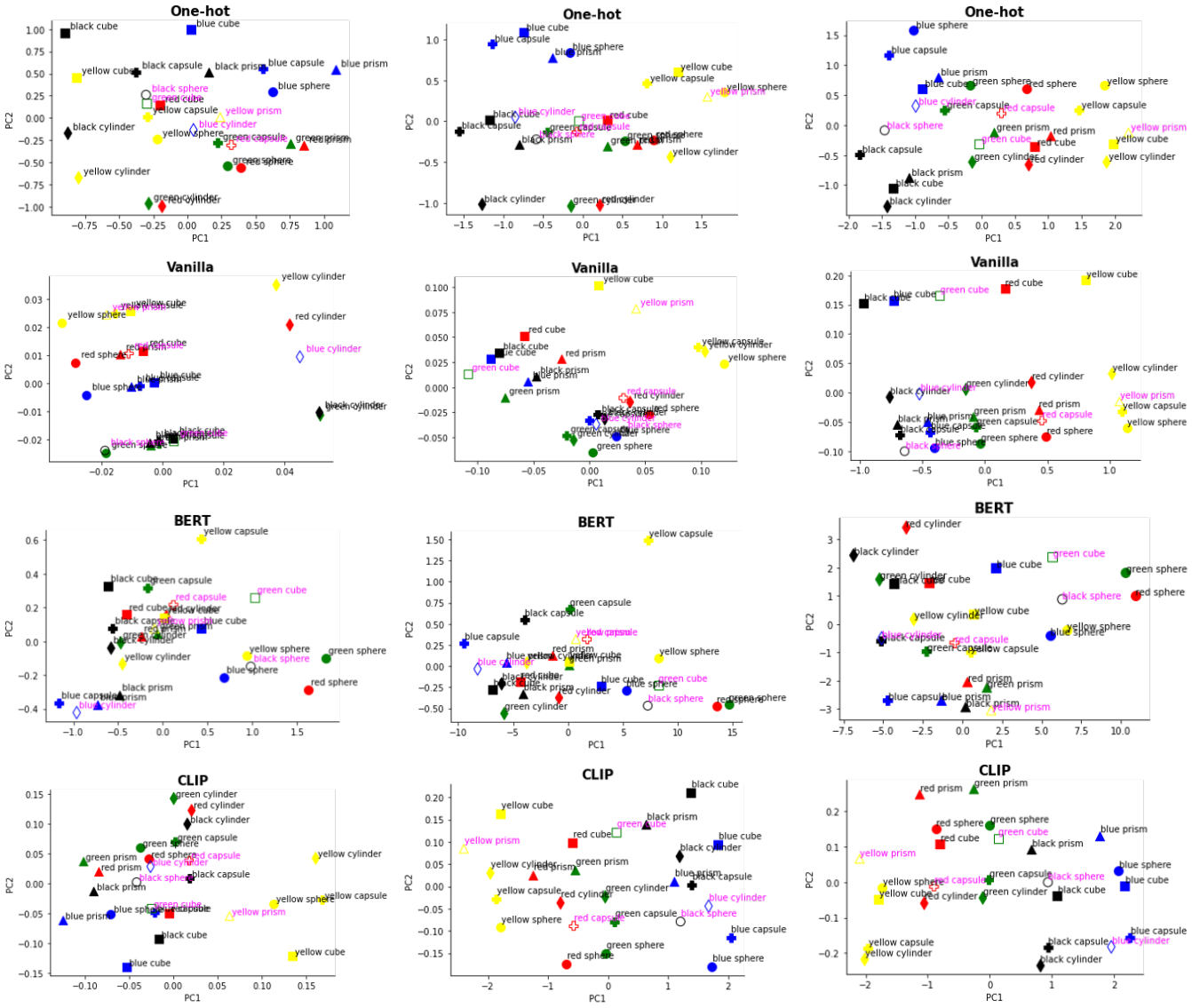

Figure 6: Word embeddings of four agents (from top to bottom: One-hot, Vanilla, BERT and CLIP) in the C+S environment before training (left), after training 50,000 episodes (middle) and after achieving performance criterion in training combinations(right).

red color and cube shape combination. Navigating to either "red sphere" or "blue cube" will disburse a negative reward.

The second environment is named "Color Shape Shape" (C+S+S) as the agent's goal is to navigate to one of the 4 predetermined locations that contain two objects described by a color attribute and two distinct shape combinations. Examples of the color-shape-shape combination instructions are "red cube cylinder" or "green sphere prism". This requires the agent to compose not just color and shape concepts but to compose three attributes to be rewarded.

The third environment is named "Color" OR "Shape"(C/S) as the agent's goal is to learn shape or color invariances respectively. For example, if the rewarded target is "red cube", the instruction given to the agent will either be "red" or "cube" and not the combination of both attributes. The agent can navigate to any red object or any colored cube to be rewarded respectively. This motivates the agent to learn shape or color invariances as two distinct concepts, which is different from the compositional learning goal of the first two environments.

To summarise, these are the key details of the three environments:

1. C+S ("Color AND Shape"): The task is to compositionally learn two word concepts. There is one target described by the color and shape attribute with three non-target objects. The instruction includes both color and shape words as inputs.
2. C+S+S ("Color Shape Shape"): The task is to compositionally learn three-word concepts. There is one target with two shapes and three non-targets with pairs of shapes. Both objects in a pair will be of the same color.vThe instruction includes all three color and two shape words as inputs.
3. C/S ("Color" OR "Shape"): The task is single word concept learning. There is one target and three non-target objects. Only one words, either color or shape, that is given as instruction.

A5: Train-test Split

To evaluate the agent's performance, we created train and test combinatorial instructions for the three environments. The train-test splits for environments C/S / C+S and C+S+S are detailed in Tables 4 and 5. The train-test split allows us to examine whether agents can learn to decompose instructions on which it was trained on and combine the Color and Shape concepts to solve unseen combinations in zero-shot during testing.

For instance, in the C+S train environment, the agent has to learn the concept "red" by only training on "red cube", "red cylinder", "red prism" and "red sphere" objects but never "red capsule". Similarly, the agent has to disentangle the concept of "capsule" by training only on "green capsule", "blue capsule", "yellow capsule", "black capsule". If the agent successfully learns to ground the two distinct word concepts "red" and "capsule", it will be able to understand the composed instruction "red capsule" without ever training on it, and navigate to the described object in zero-shot.

We included a sample Python training loop to train the agent in the Unity ML-Agents package in Figure 9.

A6: Scenarios with Overlaps in Shape or Color

Figure 8 illustrates various scenarios of the three different environments. Of these scenarios, there are some where there could be overlaps in either Shape or Color.

In the first row, various instances from the C/S environment are presented. In this environment, the target attribute is either Shape or Color. When targeting a specific shape in C/S, only one object of that shape exists among the four objects. Yet, the objects may share the same color, as demonstrated by the first and second images.

For example, in the first image, where the target instruction is "Capsule," there is only one capsule object. Since the target is shape-oriented, there is no constraint on object colors, leading to the presence of other blue-colored objects. Similarly, for color-based instructions, there is no constraint on object shapes. In the third and fourth images, there is only one object with the assigned target color, but there are other objects sharing the same shape as the target.

The second row of Figure 8 provides four examples pertaining to the C+S environment. Here, the target instruction requires a specific combination of both shape and color. Consequently, objects sharing either the same shape or color are admissible, but there can be only one object with the precise target combination. This pattern continues in the third row, representing the C+S+S environment.

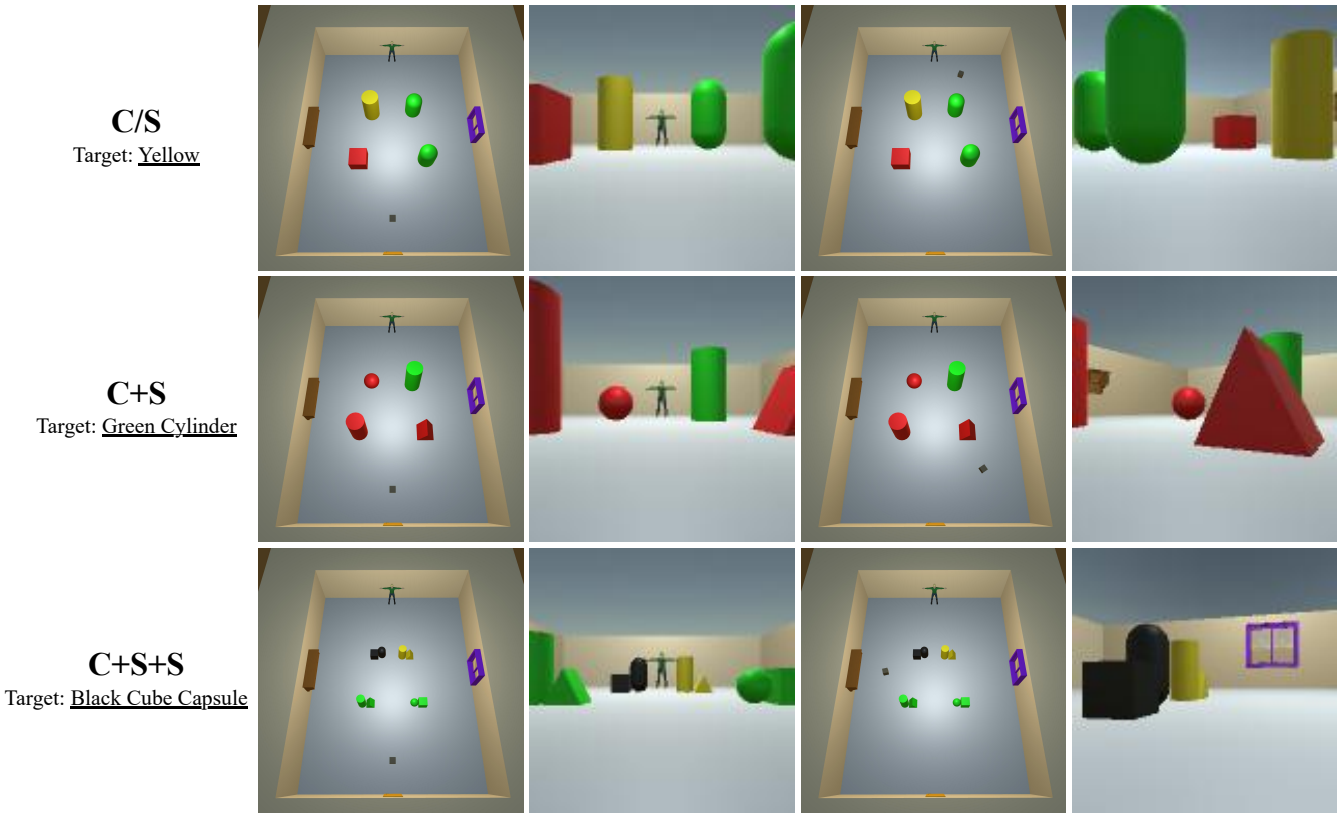

Figure 7: Three example environments. The first, second and third rows are examples from environments C/S , C+S and C+S+S respectively. The first column shows the top view of the environment, where the agent is spawned at the default starting position. The second column shows the first-person view (128x128) of the RL agent in the default starting position. The third column shows the top view of the environment, where the agent has moved around the environment. The fourth column shows the first-person view (128x128) of the RL agent in the third column. In the top view images, the agent is the small cube. For example, in the first column, the agent is the small cube located at the back of the room. For the first row (C/S environment), the target instruction is "yellow cylinder". For the second row (C+S environment), the target instruction is "green cylinder". For the third row (C+S+S environment), the target instruction is "black cube capsule".

Shape \ Color	**Red**	**Green**	**Blue**	**Yellow**	**Black**
Capsule	**Test**	Train	Train	Train	Train
Cube	Train	**Test**	Train	Train	Train
Cylinder	Train	Train	**Test**	Train	Train
Prism	Train	Train	Train	**Test**	Train
Sphere	Train	Train	Train	Train	**Test**

Table 4: Train Test Split for environments C/S and C+S.

Shape 1 \ color	Shape 2 \ color	Red	Green	Blue	Yellow	Black
Capsule	**Cube**	**Test**	Train	Train	Train	Train
Capsule	**Cylinder**	**Test**	Train	Train	Train	Train
Capsule	**Prism**	Train	Train	Train	**Test**	Train
Capsule	**Sphere**	Train	Train	Train	Train	**Test**
Cube	**Cylinder**	Train	**Test**	Train	Train	Train
Cube	**Prism**	Train	**Test**	Train	Train	Train
Cube	**Sphere**	Train	Train	Train	Train	**Test**
Cylinder	**Prism**	Train	Train	**Test**	Train	Train
Cylinder	**Sphere**	Train	Train	**Test**	Train	Train
Prism	**Sphere**	Train	Train	Train	**Test**	Train

Table 5: Train Test Split for environment C+S+S.

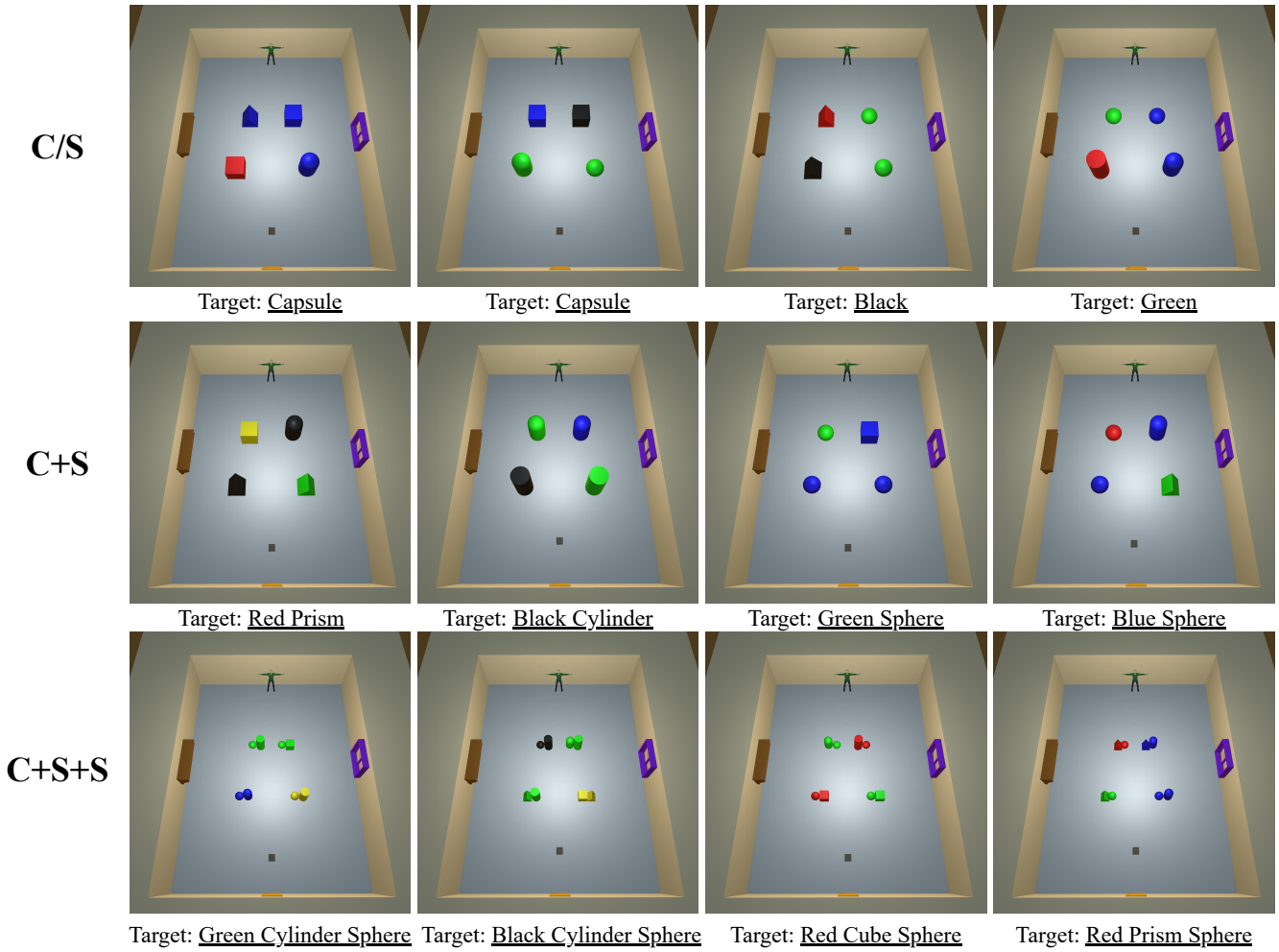

Figure 8: These images show various scenarios of the three different environments. Of these scenarios, there are some where there could be overlaps in one characteristic, either Shape or Color.

```
#imports
import mlagents
from mlagents_envs.environment import UnityEnvironment as UE
from mlagents_envs.environment import ActionTuple
import numpy as np

#initialise environments
build_train_dir="S2_train_1\\build"
build_test_dir="S2_test_1\\build"
env_train =  UE(file_name=build_train_dir,seed=1,side_channels=[],worker_id=1,no_graphics = False)
env_test =  UE(file_name=build_test_dir,seed=1,side_channels=[],worker_id=2,no_graphics = False)
env_train.reset()
env_test.reset()

#sample training loop
num_episodes=5
behavior_name=list(env.behavior_specs)[0]
behavior_spec=env.behavior_specs[behavior_name]
for episode in range(num_episodes):
    decision_steps, terminal_steps = env.get_steps(behavior_name)
    tracked_agent = -1 # -1 indicates not yet tracking
    done = False # For the tracked_agent
    episode_rewards = 0 # For the tracked_agent

    while not done:
        # Note : len(decision_steps) = [number of agents that requested a decision]
        if tracked_agent == -1 and len(decision_steps) >= 1:
            tracked_agent = decision_steps.agent_id[0]

        # Generate an action for all agents
        action = behavior_spec.action_spec.random_action(len(decision_steps))
        # Set the actions
        env.set_actions(behavior_name, action)
        # Move the simulation forward
        env.step()

        # Get the new simulation results
        decision_steps, terminal_steps = env.get_steps(behavior_name)
        if tracked_agent in decision_steps: # The agent requested a decision
            episode_rewards += decision_steps[tracked_agent].reward
        if tracked_agent in terminal_steps: # The agent terminated its episode
            episode_rewards += terminal_steps[tracked_agent].reward
            done = True #set done = True since agent reach terminal state, episode ends so
                set to True to break out of loop

    print(f"Total rewards for episode {episode+1}/{num_episodes}: {episode_rewards}")

env.close()
```

Figure 9: Python training loop for the Unity MLAgents package.