How to Learn and Represent Abstractions: An Investigation using Symbolic Alchemy

Anonymous Author(s)
Affiliation
Address
email

Abstract

Alchemy is a new meta-learning environment rich enough to contain interesting abstractions, yet simple enough to make fine-grained analysis tractable. Further, Alchemy provides an optional symbolic interface that enables meta-RL research without a large compute budget. In this work, we take the first steps toward using *Symbolic Alchemy* to identify design choices that enable deep-RL agents to learn various types of abstraction. Then, using a variety of behavioral and introspective analyses we investigate how our trained agents use and represent abstract task variables, and find intriguing connections to the neuroscience of abstraction. We conclude by discussing the next steps for using meta-RL and Alchemy to better understand the representation of abstract variables in the brain.

1 Introduction

Humans display the remarkable ability to use abstractions to guide their behavior. They are able to abstract over the sensorimotor details of a situation to derive the general principles involved, and to use those principles to behave effectively. A classic example is the use of abstract knowledge to guide behavior in a restaurant; for example, people know that they must pay at the end, even though they have never seen that exact restaurant or paid for that specific meal (Wallis et al., 2001). Neuroscientific studies have made progress in understanding how the brain might represent abstract rules (Mansouri et al., 2020), establishing that prefrontal cortex plays a significant role (Milner, 1963), with more recent work implicating the hippocampus as well (Samborska et al., 2021). One limitation of these studies is that they use extremely simple tasks, where optimal behavior can be derived as a nearly trivial function of the inputs. As a result, there is a large gap between the abstractions studied in the experiments and the richness at play in human life.

A recent development presents an opportunity to begin to bridge this gap. The Alchemy benchmark was proposed by Wang et al. (2021) to make fine-grained analysis and interpretation of meta-RL agents possible while maintaining the complexity needed for more interesting conceptual abstractions to be learnt. In the present work, we use a biologically inspired model on the symbolic version of the Alchemy benchmark (Wang et al., 2021) to investigate the learning and representation of more complex abstractions than those studied previously in neuroscience. Our model builds on the meta-RL framework of Wang et al. (2018) who showed that by considering the prefrontal cortex (PFC) as its own meta-RL system, that is driven by dopamine-based synaptic learning, one can account for a wide range of behavioral and neurophysiological findings. Our model builds on that by extending their core recurrent network with an episodic memory via a modified transformer block to represent the hippocampus, similarly to (Ritter et al., 2018).

The main contributions of the paper can be summarized as follows: (1) we show a way in which researchers can achieve high performance in Symbolic Alchemy – albeit with some tricks – without having access to vast computational resources as is usually required for deep-RL research. To be

Submitted to the 5th Workshop on Meta-Learning at NeurIPS 2021, held virtually. Do not distribute.

specific, all experiments done for this paper were run on a single-GPU (Tesla T4) machine. (2) We propose a hypothesis based on empirical results into why previous agents failed to solve Symbolic Alchemy despite being much more powerful than the one used in this work. (3) We release a tool for visualizing the chemistry and latent space of any given episode in Symbolic Alchemy to better help researchers debug the behavior of their agents. (4) We present a new kind of behavioral analysis that can be done on Alchemy to test whether the agent succeeded or failed in acquiring specific pieces of abstract knowledge. (5) Finally, we demonstrate that, just as in animals (Wallis et al., 2001; Wallis and Miller, 2003; Muhammad et al., 2006), single-units of the LSTM and transformer encode abstract task variables. Moreover, single-unit analyses revealed evidence for distinct functional roles for LSTM and transformers units. We draw a connection between this observation and the differential roles of PFC and hippocampus observed in recent neuroscience experiments (Samborska et al., 2021).

2 Methods

2.1 Symbolic Alchemy

The Alchemy benchmark was proposed by Wang et al. (2021) to make fine-grained analysis and interpretation of meta-RL agents possible while maintaining the complexity needed for interesting conceptual abstractions to be learnt. Unlike other meta-RL task distributions, Alchemy's accessibility allows us to compare our model's performance against a Bayesian learner referred to as an 'Ideal Observer'. Wang et al. (2021) also develop a 'Random Heuristic' that we use as a reference for evaluating our agent's understanding of specific abstract principles. The task itself is divided into episodes, each of which consists of 10 trials. In our experiments, the number of timesteps per trial was set to 15 to speedup training.

Stones and Potions The goal of the agent within each trial is to transform three stones into a more valuable form – the value of which is tied to the stone's perceptual features – by applying a sequence of different potions on each of them. The agent can then collect the reward associated with a stone by dropping it into a central cauldron. The stone's appearance can change along only one of three dimensions at a time: size, color and shape. Each potion on the other hand is characterized by one of 6 hues that dictates the transformative effect it has on a specific stone according to some 'chemistry' that is sampled from a structured generative process at the beginning of each episode. There are 12 potions in each trial, each of which is consumed (i.e. can not be re-used) once applied on a stone.

Chemistry The chemistry dictates the causal structure that governs each episode as well as the possible set of stone perceptual features that can occur. It can be visualized as a cube, where each vertex correspond to a specific stone value in the latent space, which can be one of (-3, -1, +1, +15), or a specific appearance in the perceptual space. The potion effects run along the edges of the cube as shown in Figure 1. Edges can be missing, creating a bottleneck that the agent must pass through to reach a high rewarding state. This may require passing through intermediate lower value states first.

Abstract Principles There are certain rules and constraints that span across episodes which makes Alchemy a meta-learning benchmark. The agent is expected to learn to identify and exploit those regularities in order to achieve high performance in the task. This includes:

- **Consistency**: Potions of the same color will always have the same effect on stones with the same visual features within an episode.
- **Parallelism**: Each potion color has the same direction of effect regardless of the other features (e.g. a red potion will always change the color of a stone from blue to purple irrespective of the size and shape of that stone).
- **Missing Edges**: Overlaid on that parallelism, some edges can be disabled for an episode. Therefore once discovered the agent shouldn't attempt to traverse that missing edge.
- **Potion Pairs**: Potions come in pairs with opposite effects (red/green, yellow/orange, pink/turquoise). The agent should know those pairs since they are consistent across all episodes. For example, if the effect of the orange potion is to increase the size of a stone, then the effect of the yellow potion will be to decrease the size of the stone.

State and Action Spaces In the input representation, each stone is represented by its three perceptual features, its reward in the current latent state and whether it has been deposited into the cauldron or not. For the potions, we use a modified representation from the one proposed by Wang et al. (2021); instead of representing the state of each of the 12 potions as elements of a single vector, we represent the remaining number of potions per hue. Therefore the state space is comprised of a 21-dimensional vector. The action space is also modified in a similar manner, where the agent chooses which color to apply to which stone, or whether to deposit a specific stone to the cauldron. When a specific stone-color combination is chosen, a wrapper then randomly selects one of the available potions with that color and apply it to the selected stone. Alternatively a `no-op` action can be chosen, which makes the number of possible actions 22.

Rewards and Penalties In addition to the rewards the agent receives whenever depositing a stone into the cauldron, we found that penalizing the agent in three specific scenarios sped up the rate of convergence considerably. Specifically, we gave a reward of -0.2 if the action taken results in a null transition (i.e. does not have any effect on the stone) but given that it is not a `no-op` action. In the second case, the agent was penalized with a reward of -1 whenever it chose to use a potion hue that is not available (i.e. it should learn to never use a hue when its corresponding entry is zero in the input representation) or a stone that has already been cached in the cauldron. Finally, an additional penalty of -1 was given when the agent chose the same potion color consecutively on the same stone.

2.2 Visualizing the Latent Space

In order to simplify the process of qualitatively debugging the agent's behavior, we developed and are releasing[1] a tool for Symbolic Alchemy that visualizes the topology of the underlying causal graph for a given episode. Overlaid on it are the positions of the stones in the latent space along with arrows indicating the direction of effect of the available potions. The Ideal Observer's belief about a particular potion color or edge is also indicated using the opacity of the corresponding object. That way we can evaluate the agent's actions with respect to the belief state of an agent that has perfect understanding of the structure of the task.

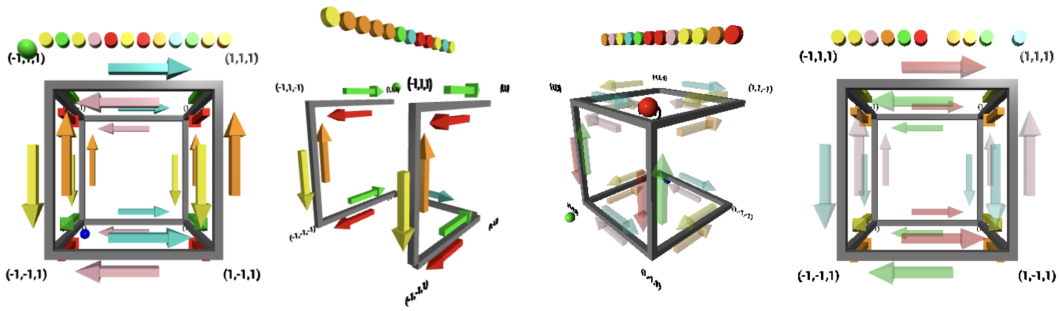

Figure 1: Visualizations of chemistries created using the Symbolic Alchemy Visualization Tool, which we are releasing for public use. The coordinates on the cube's vertices indicate the latent state and thus reward of the stone in that position. Translucent arrows indicate that the agent has not yet discovered the effect of that color. The stones are represented by a red, blue and green spheres. The available potions are shown floating above the cube with their corresponding color, and when a potion is consumed it will no longer be visible. In some chemistries, edges of the graph may be missing as can be seen in the 2nd and 3rd snapshots.

2.3 Agent Architecture

We use a biologically inspired architecture that maps to functions and neural structures in the brain. Specifically, we build on the work of Wang et al. (2018) in which the prefrontal cortex (PFC) is conceptualized as forming a gated recurrent neural network (characterized as an LSTM (Hochreiter and Schmidhuber, 1997)) and augment it with an episodic memory which is connected to the LSTM via a single modified transformer block from Ritter et al. (2020) (see Figure 2).

[1] https://github.com/username/repo

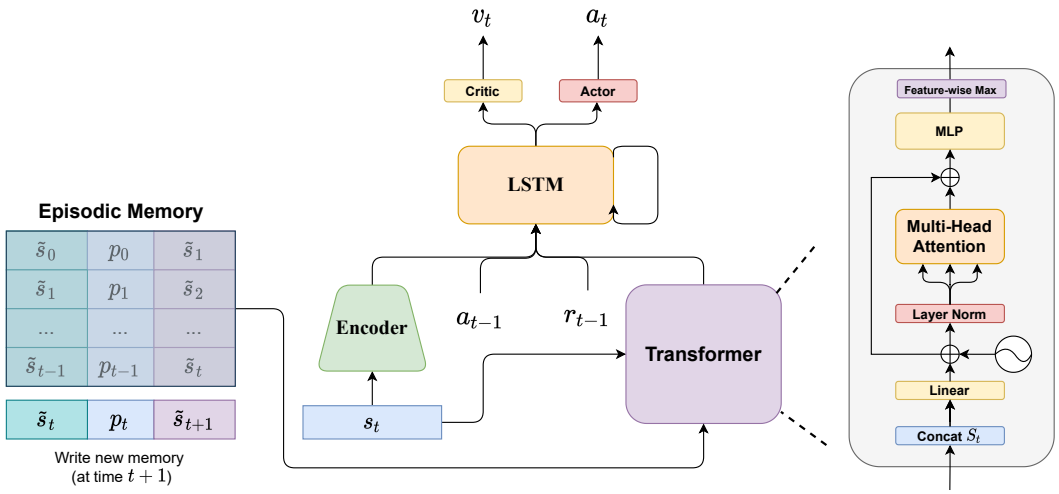

Figure 2: In the center of the architecture is an LSTM that takes as input the action and reward of the previous timestep, an encoded version of the current state, and a distilled representation of the relevant memory entries. The encoder is just a two-layered MLP. The episodic memory stores the features of the stone being transformed $\widetilde{s}_t$, a one-hot encoding of the applied potion color p_t, and the resultant features of the stone after transformation $\widetilde{s}_{t+1}$. The transformer block architecture is fully described in Section 2.3. The output of the LSTM is finally given to the policy and value networks, each of which is a linear layer, to generate an action and the value estimate respectively.

The LSTM, which has been shown to have analogies with prefrontal gating theories (Chatham and Badre, 2015), takes as input on each time-step an encoded version of the current state, the reward and action it had taken at the preceding time-step and a compressed representation of the previous memories experienced in the current episode. Since the LSTM learns to distill that information over the course of an episode in its hidden state, it can be considered as a form of working-memory (Lara and Wallis, 2015).

On the other hand, the transformer architecture takes as input the current state s_t and the entries of the episodic memory $\{m_i\}_{i=0}^{t}$ on each time-step. The state s_t is then concatenated to each m_i. This matrix is then transformed feature-wise by a shared linear layer, before being passed to the planner module of Ritter et al. (2020) described using the following equations: $s_t^* = \mathrm{ReLU}(x + \phi(x))$ where $x = \{[m_i, s_t]\}_{i=0}^{t}$ and $\phi(x) = \mathrm{MHA}(\mathrm{LayerNorm}(x))$ where MHA is the multi-head dot-product self-attention mechanism described by Vaswani et al. (2017) with layer-normalization (Ba et al., 2016) applied to the input. The output is an attended view of the agent's past relevant experience in the episode given the current state. This is then passed to a two-layer shared MLP, the output of which is pooled using a feature-wise max operation. We refer to our agent as A2C EPN.

2.4 Experimental Setup

The agent is trained using the synchronous version of the Advantage Actor-Critic (A2C) RL algorithm (Mnih et al., 2016) with a batch-size of 8. The value coefficient was set to $\beta_V = 0.5$ and the initial entropy coefficient to $\beta_E = 0.1$ which was decayed in a linear fashion throughout training. The encoder is a stack of 2 affine layers with 32 units each, and an ELU non-linearity in-between (Clevert et al., 2016). The LSTM has 256 hidden units, while the transformer block contains 4 attention heads with a dimensionality of 64. The MLP is similarly a stack of two affine layers with 64 units each, with an ELU non-linearity in-between. The episodic memory operated with a maximum size of 150 entries, and was reset after each episode. We used the Adam optimizer (Kingma and Ba, 2014) with an initial learning rate of $7.5e-4$. The learning rate was decayed linearly from the start of training. The gradient was clipped at a maximum norm of 100. We found that starting with a small discount factor of $\gamma = 0.7$ worked best. The resultant model was then finetuned using a higher value ($\gamma = 0.95$) after convergence. The number of unroll steps is 20. The code is made open-source[2].

[2]https://github.com/username/repo

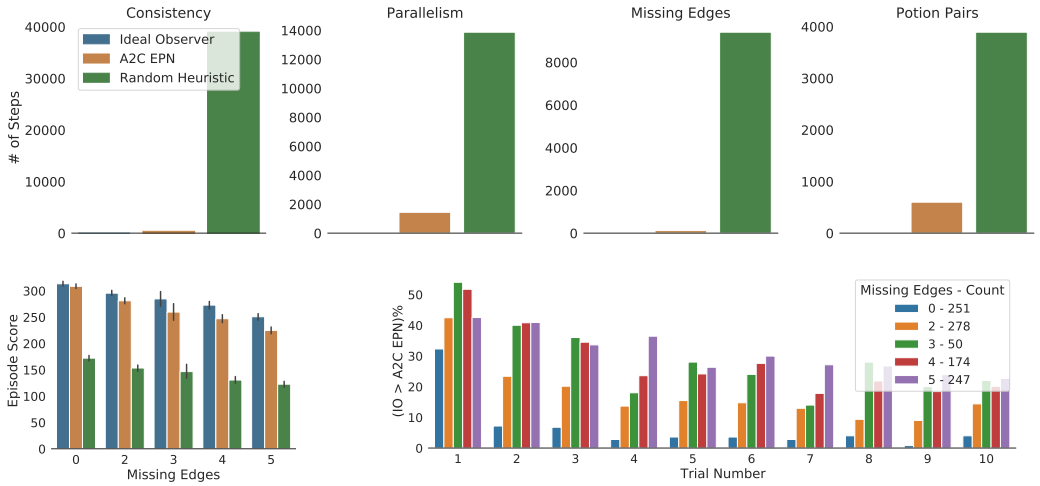

Figure 3: **Top**: Behavioral results showing the agent's lack of understanding of several abstract principles that span across episodes. In each plot, we report the number of times the agent took a null transition that it should have avoided if it understood the corresponding concept. The Ideal Observer is zero in all cases demonstrating perfect understanding of each principle. Our agent performs significantly better than the Random Heuristic but still takes a lot of actions where it could know better in the Parallelism and Potion Pairs analyses. **Bottom-left**: The average episode score as a function of the number of missing edges for each agent. **Bottom-right**: The percentage of episodes where the Idea Observer scores more than the A2C EPN agent at a given trial.

3 Results

The results and analyses presented in this section have all been done on the 1000 held-out evaluation episodes provided by the benchmark using the agent described in Section 2.3.

3.1 Performance

The A2C EPN agent achieves an average episode score of 267.01 with the modified representations, which is significantly higher than the much more powerful VMPO agent with a gated transformer XL network (Song et al., 2020; Parisotto et al., 2019) used by Wang et al. (2021). In an attempt to identify which aspects contributed to this leap in performance, we re-trained our agent but using the canonical representation in each of the input, output and memory separately as shown in Table 1 (more details about each in the Appendix A.2). The results show a significant drop to almost the same performance reported by Wang et al. (2021) when using the canonical action space. Interestingly, evaluating the same agent but without committing anything into the episodic memory led to the same performance (see Score w/o Memory column in Table 1), indicating that the model was not making use of its long-term history to appraise the value of future actions.

This inability to exploit the memory can be largely attributed to the notion of positional output, where the agent is required to choose an instance of the potion and not the abstract color that causes the transformative effect. Since each potion slot can have more than one color across different trials, the same action will have different effects within the same episode. For instance, action a_2 in trial k can be a pink potion while in another trail can correspond to an orange potion. In order to have an appropriate mapping between each instance and its abstract color one will need a more suitable architecture that would be able to recapitulate the information we gave to the system via the custom encoding. The series of experiments we presented show where such an architecture search could begin. Specifically, an architecture design that gets the right information into the memory and provides sufficiently flexible neural networks to process the memories in order to reproduce the information contained in our modified representation. In this work we used the custom encoding since it enabled us to pinpoint where the architectural problem lies and perform interesting analyses that we present in the following sections with a small compute budget.

Table 1: Evaluation episode scores comparing the effect of the canonical representation (indicated by an 'o') proposed by the Alchemy benchmark to the modified one (indicated by an 'x') described in Section 2.1. The VMPO result taken from (Wang et al., 2021). The no bottleneck column indicate the average score of the evaluation episodes with no missing edges.

Agent	Input	Output	Mem	Score $\pm$ SEM	Score (w/o Mem)	No Bottleneck
A2C EPN	x	x	x	267.01 ± 1.84	160.73	308.48
	o	x	x	243.83 ± 2.21	171.59	300.79
	o	o	x	156.34 ± 1.57	156.70	181.41
	o	o	o	158.91 ± 1.60	153.40	182.10
VMPO	o	o	-	155.40 ± 1.60	-	-
Random Heuristic	-	-	-	146.07 ± 1.55	-	172.11
Ideal Observer	-	-	-	284.42 ± 1.59	-	313.31

To measure the effect of reward shaping on the final performance, we evaluated the A2C EPN agent without giving it any additional rewards or penalties. This achieved a score of 228.08 ± 2.15, which suggests that exploration is a bottleneck. In other words, the problem does not really have to do much with learning from examples to represent abstract variables, but instead is just about having a data distribution that's sufficiently broad.

We found it useful as well to compare the results with respect to the number of missing edges in the graph (see Figure 3), since the A2C EPN agent's performance approached that of the Ideal Observer in the case where there are no bottlenecks. In an effort to bridge this gap, we found that this difference is more pronounced in earlier trials. It is especially visible in the first trial in the case where there are no missing edges as shown in Figure 3, where the A2C EPN agent performs similar to or better than the Ideal Observer in far more number of episodes after the first trial. This can be mitigated in future work by incorporating more advanced exploration methods.

Action Types Similar to the analysis done in Wang et al. (2021), Figure 4 shows the number of times, throughout the first and last trial, the agent applied a potion that worsened, improved or had no effect on the value of the stone and the number of times the agent deposited a stone into the central cauldron as a function of its reward. It can be seen that the A2C EPN agent almost never deposits a negative reward stone and more importantly, it adapts its strategy throughout the course of the episode. Concretely, in the first trial it performs a lot of exploratory actions by trying out potions that do not have an effect on the stone (the orange area) while in the last trial it shifts towards a more exploitative strategy by performing more actions that improved the value of the stone as it acquired knowledge about the episode's chemistry. This ability to adapt is indicative of good meta-learning performance and is similar in behavior to what we see from the Ideal Observer as shown in the Appendix.

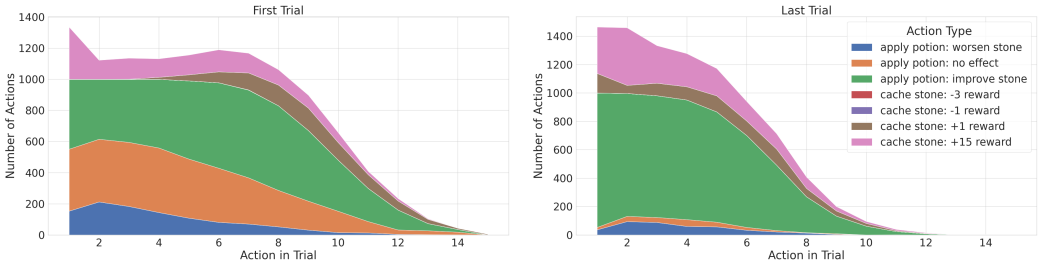

Figure 4: Comparing different action types throughout the first and last trial in a similar manner to Wang et al. (2021). The A2C EPN agent shows analogous behavior to that of the Ideal Observer (see Figure 6 in Appendix), indicating an ability to adapt strategies (exploration vs exploitation) throughout the course of the episode.

3.2 Behavioral Tests

In the following behavioral tests we report the number of times the agent took a null transition (i.e. applying a potion that has no effect on a stone) that wasn't the result of a missing edge or choosing a potion color or stone that was not available, but due to a lack of understanding of a specific abstract principle. We compare our agent's behavior to that of the Ideal Observer and the Random Heuristic.

To put it formally, let $t : \langle x, y, z \rangle \xrightarrow{p} \langle w, v, u \rangle$ be a transition that the agent has observed earlier in the episode, where x, y and z are the color, size and shape of the stone in question respectively, p is the color of the potion being applied to the stone, and w, v and u are the stone perceptual features after applying the potion on it. Note that $\langle x, y, z \rangle$ can be equal to $\langle w, v, u \rangle$, and we call that a 'null transition'. Since there are no rotated graphs in the evaluation episodes, each feature dimension can only take one of two values. This implies that each stone can only be in one of $2^3 = 8$ possible perceptual states. Figure 3 shows the results of each tested abstract principle.

Consistency Here we count the number of times the agent applied a potion with color p on a stone with features $\langle x, y, z \rangle$ after observing that t is a null transition. In other words, we have evidence that the agent does not understand consistency if it applied a potion on a stone after observing that this particular potion color has no effect on a stone with the same visual features earlier in the episode.

Parallelism In this test, we calculate the frequency in which the agent applied a potion with color p on a stone with features $\langle a, b, \bar{z} \rangle$ for the first time after it observed that t is not a null transition, where $\bar{z}$ is the other possible shape, assuming that p is responsible for transforming the shape of the stone. For example, suppose the agent observed a red potion transforming a large blue round stone to pointy. Subsequently in the episode, it should never apply a red potion to any pointy stone, including small or purple stones.

Missing Edges The agent will show a lack of understanding of missing edges if it re-applied potion p on a stone with the same latent state as the stone with $\langle x, y, z \rangle$ after observing that t is a null transition as a result of a missing edge. In other words, we compute the frequency in which the agent attempted to transform a stone after 'discovering' that this color has no effect on the current latent state due to a missing edge in the graph. Note that the agent does not have access to this information, but must identify it by experimentation.

Potion Pairs To test whether the agent demonstrate an understanding that potions come in pairs with opposite effects, we count the number of steps in which the agent made a null transition as a result of applying a potion color $\bar{p}$ that it has not seen the effect of before, but has observed the effect of its opposite color previously in the episode (i.e. t was not a null transition). For example, an agent should know that yellow potions decreases the size of stones after only observing that an orange potion transformed a small stone to a large one (without having to see the effect of a yellow potion before). Therefore, it should never apply a yellow potion to a small stone as to avoid a null transition.

3.3 Single-Unit Activations

Inspired by single-cell recordings in neuroscience, where single neurons are usually shown to be selective for a specific abstract concept (Wallis et al., 2001), we probed our model to see if it will give rise to similar selectivity by analysing the activations of single units in the LSTM and that of the transformer (specifically, the output of the feature-wise max). Figure 5 shows the activations of a few units averaged across all steps in the 1000 evaluation episodes. The latent and perceptual state of the stone that the agent chose is recorded along with the activation of each unit in each timestep. Note that there are 8 latent states that correspond to the 8 vertices of the cube, and 8 possible perceptual states as previously mentioned in all of the held-out episodes. In the plots, we denote each state using a single number by binarizing its features. The top row shows the activations of some LSTM units while the bottom row shows activations recorded from the output the transformer.

Interestingly, we found distinct functional specialization between the transformer and the LSTM units. Specifically, 32.8% of transformer units showed some understanding of the potion pairs abstract concept. Specifically, each of those units had a positive activation for one potion color and a negative activation for its opposite color (some samples shown in Figure 5), whereas no LSTM units did. The LSTM units, on the other hand, were mostly selective to stone-reward combinations. For instance,

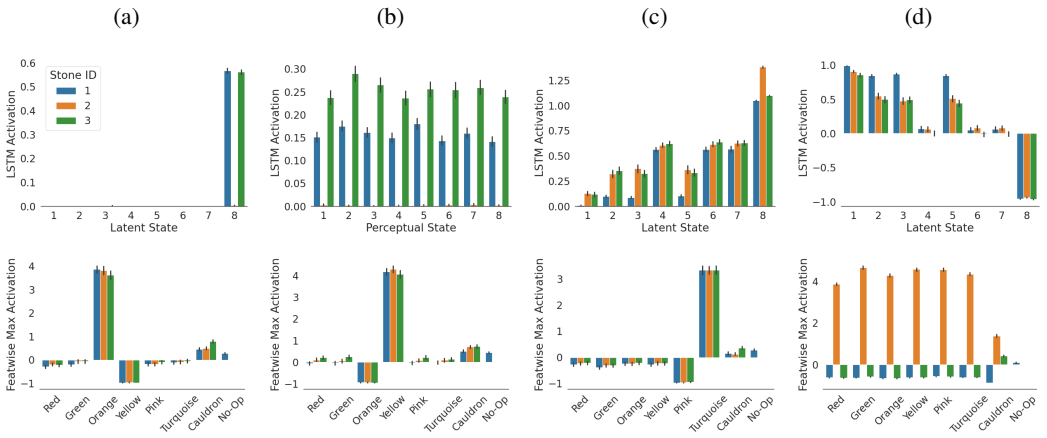

Figure 5: **Top**: LSTM units. **(a)** The activation of unit 100 as a function of the latent state. This unit is only responsive when stones 1 or 3 are in latent state 8 (the state with the highest reward). **(b)** The same unit as in (a) but as a function of the stone's perceptual state. It implies that the unit is responsive regardless of the stones' perceptual features. **(c)** The activations of unit 232 has a magnitude proportional to the reward of its corresponding state regardless of which stone is used. Note that states $\{4, 6, 7\}$ have a reward of $+1$ while the rewards of states $\{2, 3, 5\}$ is -1. **(d)** The activations of unit 252 are positive in the states with negative reward. **Bottom**: Transformer units. **(a)** Unit 5 has a high activation when the agent choose the orange potion while a negative activation when it chooses its opposite color. **(b)** Unit 15 is the opposite of (a). **(c)** Unit 40 is similarly responsive when the agent chooses potion with a specific color (here turquoise) and has a negative response for its opposite color. **(d)** Unit 56 is selective when the agent chooses an action that uses stone 2 regardless of the potion color.

several units were only responsive when a specific set of stones had either a positive or negative reward and an opposite activation otherwise irrespective of the actual value or the stones' visual features. This also shows some notion of abstraction since the agent understands which stones it needs to deposit to the cauldron regardless of its exact representation.

However, there were no single units that were responsive to a particular perceptual state or any single visual feature, nor were there units responsive to specific latent states except to the reward associated with that state. This led us to conjecture that the model reduces the cubic structure of the latent space to a two-dimensional form (i.e. a square), where each vertex correspond to one of the four possible rewards. This is inline with the observation that the agent is unable to handle bottlenecks in the underlying causal graph.

4 Discussion and Future Work

In this paper, we present an agent that is capable of meta-learning a set of abstract principles that underpins the complexity of the Alchemy benchmark. Further, we present a battery of analytical tools that can be used to test for specific pieces of abstract knowledge. Our design choices were motivated by the fact that strong deep-RL agents were unable to solve this task (Wang et al., 2021), thus we gradually simplified the problem in order to identify exactly where the bottleneck lies. To that end, we reached some conclusions that can guide researchers in searching for better suited architectures. Specifically, our experiments show that more efficient outer-loop exploration and an architecture for handling positional output are required. We show as well that researchers with a limited compute budget can use Symbolic Alchemy in order to analyze their deep-RL agents in a principled manner.

Finally, we found evidence for single-unit representations of abstract variables such as potion pairs, and functional dissociation between 'cortical' and 'hippocampal' units which were modeled as an LSTM and transformer respectively. This connects with what's been seen in recordings from rodent and primate cortex (Wallis et al., 2001; Samborska et al., 2021; Mansouri et al., 2020). This opens the way for future work to use meta-RL to carry out more detailed simulations of classic neuroscience experiments to better understand the mechanisms underlying the observed results.

Acknowledgments

References

Ba, J., Kiros, J., and Hinton, G. E. (2016). Layer normalization. *ArXiv*, abs/1607.06450.

Chatham, C. and Badre, D. (2015). Multiple gates on working memory. *Current Opinion in Behavioral Sciences*, 1:23–31.

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. (2016). Fast and accurate deep network learning by exponential linear units (elus). *arXiv: Learning*.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

Lara, A. H. and Wallis, J. D. (2015). The role of prefrontal cortex in working memory: A mini review. *Frontiers in Systems Neuroscience*, 9:173.

Mansouri, F., Freedman, D., and Buckley, M. (2020). Emergence of abstract rules in the primate brain. *Nature reviews. Neuroscience*, 21.

Milner, B. (1963). Effects of different brain lesions on card sorting: The role of the frontal lobes. *Archives of neurology*, 9(1):90–100.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In Balcan, M. F. and Weinberger, K. Q., editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1928–1937, New York, New York, USA. PMLR.

Muhammad, R., Wallis, J., and Mller, E. (2006). A comparison of abstract rules in the prefrontal cortex, premotor cortex, inferior temporal cortex, and striatum. *Journal of cognitive neuroscience*, 18:974–89.

Parisotto, E., Song, H. F., Rae, J. W., Pascanu, R., Gülçehre, Ç., Jayakumar, S. M., Jaderberg, M., Kaufman, R. L., Clark, A., Noury, S., Botvinick, M. M., Heess, N., and Hadsell, R. (2019). Stabilizing transformers for reinforcement learning. *CoRR*, abs/1910.06764.

Ritter, S., Faulkner, R., Sartran, L., Santoro, A., Botvinick, M., and Raposo, D. (2020). Rapid Task-Solving in Novel Environments. page 16.

Ritter, S., Wang, J., Kurth-Nelson, Z., and Botvinick, M. (2018). Episodic control as meta-reinforcement learning. *bioRxiv*.

Samborska, V., Butler, J. L., Walton, M. E., Behrens, T. E., and Akam, T. (2021). Complementary Task Representations in Hippocampus and Prefrontal Cortex for Generalising the Structure of Problems. preprint, Neuroscience.

Song, H. F., Abdolmaleki, A., Springenberg, J. T., Clark, A., Soyer, H., Rae, J. W., Noury, S., Ahuja, A., Liu, S., Tirumala, D., Heess, N., Belov, D., Riedmiller, M. A., and Botvinick, M. (2020). V-mpo: On-policy maximum a posteriori policy optimization for discrete and continuous control. *ArXiv*, abs/1909.12238.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Wallis, J., Anderson, K., and Mller, E. (2001). Single neurons in prefrontal cortex encode abstract rules. *Nature*, 411:953–6.

Wallis, J. D. and Miller, E. K. (2003). From rule to response: Neuronal processes in the premotor and prefrontal cortex. *Journal of Neurophysiology*, 90(3):1790–1806. PMID: 12736235.

Wang, J., Kurth-Nelson, Z., Kumaran, D., Tirumala, D., Soyer, H., Leibo, J., Hassabis, D., and Botvinick, M. (2018). Prefrontal cortex as a meta-reinforcement learning system. *Nature Neuroscience*, 21:14.

Wang, J. X., King, M., Porcel, N., Kurth-Nelson, Z., Zhu, T., Deck, C., Choy, P., Cassin, M., Reynolds, M., Song, F., Buttimore, G., Reichert, D. P., Rabinowitz, N., Matthey, L., Hassabis, D., Lerchner, A., and Botvinick, M. (2021). Alchemy: A structured task distribution for meta-reinforcement learning. page 16.

A Appendix

A.1 Action Type for Ideal Observer and Random Heuristic

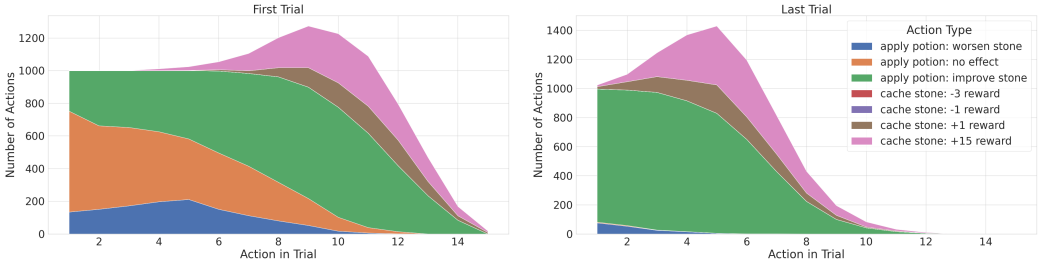

Figure 6: The type of action that the Ideal Observer take throughout the first and last trial. It can be seen that similar to the A2C EPN agent, it adapts its strategy across the episode.

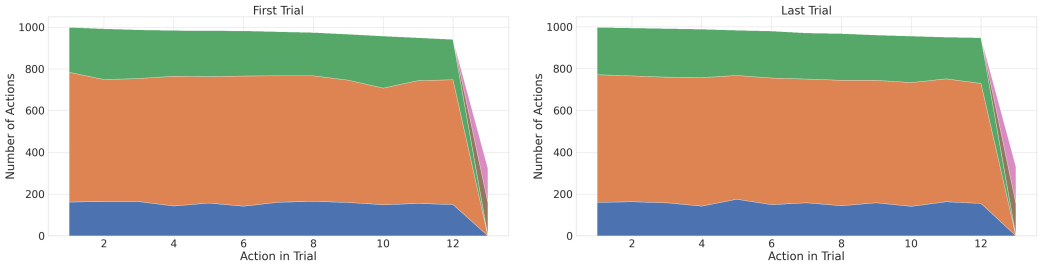

Figure 7: The type of action that the Random Heuristic take throughout the first and last trial. The agent's strategy is the same in both trials demonstrating an inability to adapt strategies.

A.2 Input, Output and Memory Representations

The canonical input and output representation are described in detail in Wang et al. (2021). The canonical memory representation on the other hand stores each entry as $[s_t, a_t, s_{t+1}]$ where s_t is the current input state, a_t is a one-hot encoding of the executed action at timestep t and s_{t+1} is the state at the following timestep that contains the transformed stone.