Detecting and Accommodating Novel Types and Concepts in an Embodied Simulation Environment

Sadaf Ghaffari   SADAFGH@COLOSTATE.EDU
Nikhil Krishnaswamy   NKRISHNA@COLOSTATE.EDU
Situated Grounding and Natural Language Lab, Department of Computer Science, Colorado State University, Fort Collins, CO 80523 USA

Abstract

In this paper, we present methods for two types of metacognitive tasks in an AI system: rapidly expanding a neural classification model to accommodate a new category of object, and recognizing when a novel object type is observed instead of misclassifying the observation as a known class. Our methods take numerical data drawn from an embodied simulation environment, which describes the motion and properties of objects when interacted with, and we demonstrate that this type of representation is important for the success of novel type detection. We present a suite of experiments in rapidly accommodating the introduction of new categories and concepts and in novel type detection, and an architecture to integrate the two in an interactive system.

1. Introduction

Humans are efficient at seeking out experiences that are maximally informative about their environment (Markant & Gureckis, 2014; Najemnik & Geisler, 2008; Nelson et al., 2010; Renninger et al., 2007; Schulz & Bonawitz, 2007). We explore the physical world to practice skills, test hypotheses, learn object affordances, etc. (Caligiore et al., 2008; Gopnik & Meltzoff, 1997; Gopnik, 2010, 2012; Gottlieb & Oudeyer, 2018; Neftci & Averbeck, 2019; Piaget, 1963; Piaget & Inhelder, 2008; Son & Sethi, 2006). Young children, in particular, can rapidly expand their vocabulary of concepts with few or no examples, and generalize from previous to new experiences (Clark, 2006; Colung & Smith, 2003; Vlach & Sandhofer, 2012).

Meanwhile, artificial neural networks require large numbers of samples to train. It may take 5-8 layers of artificial neurons to approximate a single cortical neuron (Beniaguev et al., 2021). Common few-, one-, or zero-shot learning approaches in AI provide at best a rough simulacrum of human learning and generalization (Knudsen, 1994; Niv, 2009; Zador, 2019). Recent successes in few-shot learning in end-to-end deep neural systems still require extensive pre-training and fine-tuning, often on special hardware (Brown et al., 2020), or specific task formulation (Schick & Schütze, 2020). They do not easily or organically expand to accommodate new concepts.

In this paper, we present an investigation into the ability of machine learning systems to engage in certain metacognitive behaviors pertaining to the detection and acquisition of new concepts. In the context of this paper, we take a "metacognitive" process to be one that requires the system to be aware of what it does and doesn't know. This is, of course, just one factor that makes an agent

"metacognitive" (Meichenbaum, 1985). Additionally, to make our problem tractable, we focus on the particular domain of object types, and more abstract concepts that inhere across multiple object types, such as the opposition between "flatness" and "roundness." We model the concept acquisition problem as one of an agent engaged in open-ended play with simple objects, as might be the case for an infant or toddler who starts off interacting with only a few classes of simple objects before other comparatively more complex objects are introduced. In particular, we address two questions: 1) How can a model grow to accommodate novel object types, and 2) How can an agent automatically detect that a new object type has been introduced into the environment such that it knows to update its model?

2. Related Work

This work has many antecedents in earlier AI, particularly early explorations into multi-task learning (Caruna, 1993) and knowledge transfer between neural networks (Pratt et al., 1991).

Object recognition and classification is of course a well-traveled area in AI, but the AI approaches also have antecedents in the cognitive science community. Among many others, Riesenhuber & Poggio (2000) presented models of computational object recognition inspired by processes in the human visual cortex, Oliva & Torralba (2007) motivated development on pre-neural network computer vision systems through examining human use of contextual cues in object recognition, and DiCarlo & Cox (2007) drew on both neurophysiology and computation to examine the brain mechanisms that allow for rapid object recognition under multiple circumstances. Piloto et al. (2022) is one of a number of recent approaches that explore similar questions in vision, but uses an order of magnitude more samples than our approach, and does not address the contribution of interacting with objects directly.

Among approaches where the interaction between agent and environment are central, Nolfi (2005), drawing on "embodied cognitive science" (Scheier & Pfeifer, 1999), proposed a theory of category formation based on the results of interacting with the environment in simple tasks. Mohan et al. (2012) and Frasca et al. (2018) are examples of work from the cognitive systems community that address language grounding and acquisition in situated environments. Bar-Aviv & Rivlin (2006) used simulation to classify objects based on their functional properties, but did not look at identifying when a novel class has been introduced.

Fitzgerald et al. (2021) consider similar questions, where they begin with a model of a task and then attempt to transfer that model to a related task that is identified through perception, while also recognizing failure. Metacognition, even in the limited scope as we have defined it herein, is supervenient upon such capabilities.

3. Environment and Data Collection

To create an environment where an agent can explore object properties through interaction, we use the VoxWorld platform for interactive agents (Krishnaswamy et al., 2022). We create an environment where an agent is presented with two objects: one cube and one instance of another class, known as the *theme object* (see Fig. 1). The objects all differ from each other in their geometric features—particularly where they have *round* sides and *flat* sides—and the object set is broadly inspired by geometric children's toys. Further examining some key distinctions between minimal pairs of objects, we see that in the pairs *cube/rectangular prism* and *sphere/egg*, the latter object

is lengthened along the X-axis. Comparing *cylinder* and *capsule*, one has flat ends while the other has round ends. Examining *cone* vs. *pyramid*, we see that even though the objects have the same dimensions, the faces of *pyramid* are all flat while everything but the base of *cone* is round. These distinctions affect how each object behaves when it is interacted with. The agent samples from the environment by stochastically placing the theme object on top of the cube and observing the result.
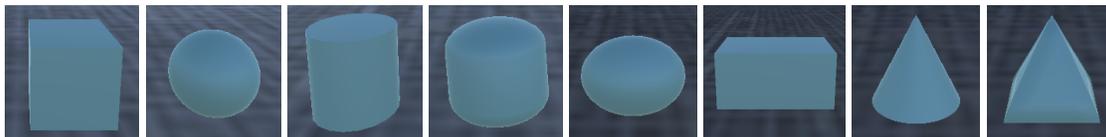

Figure 1: Possible theme objects, shown in default orientation: *cube*, *sphere*, *cylinder*, *capsule*, *egg*, *rectangular prism*, *cone*, *pyramid*. Also included is a *small cube* (not shown).

If the object configuration that results from this placement is stable, the theme object will remain supported by the destination cube. If not, it will fall off. There are multiple reasons why an object might not stay balanced on top of the cube, which have to do with both placement location and the object properties themselves. For instance, a cube will stay balanced if placed directly on top of the destination cube's surface, but if placed on the edge, will probably fall off. A sphere may never stay balanced no matter where it is placed. Meanwhile a cylinder or cone will probably stay balanced if appropriately placed *in an upright orientation*, but not in a horizontal one (e.g., see Fig. 2). These distinctions in configuration and resultant behavior correspond to the *habitats* (Pustejovsky, 2013) and *affordances* (Gibson, 1977) of the object, as they pertain to its "stackability."
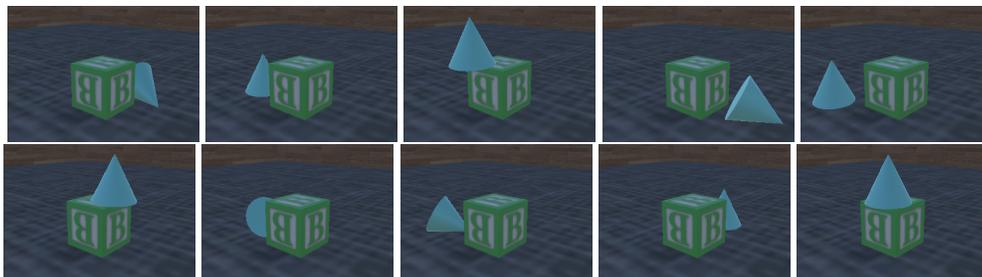

Figure 2: 10 cone stacking attempts. When put in an unstable position, the cone falls off the cube.

$$\left[\begin{array}{l} \textbf{cylinder} \\ \\ \text{TYPE} = \left[\begin{array}{l} \text{HEAD} = \textbf{cylindroid} \\ \text{COMPONENTS} = \textbf{nil} \\ \text{ROTATSYM} = \{Y\} \\ \text{REFLSYM} = \{XY, YZ\} \end{array} \right] \end{array} \right]$$

Figure 3: VoxML typing structure for a cylinder, showing axes and planes of rotational and reflectional symmetry.

For each object type, we generate 10,000 samples of stacking attempts. These attempts are not guided by a policy and no learning is taking place here (see Sec. 7 for the integration of reinforcement learning). Instead the agent stochastically places the theme object atop the destination object at a two dimensional coordinate in the range $(-1, 1)$ that is scaled for the size of the destination object's surface and resolved to 3D coordinates in the VoxWorld environment. VoxWorld's underlying Unity physics engine then simulates what happens to the objects in that configuration: either the theme object stays supported or it falls off. The agent attempts to place the theme object on the destination object again, keeping it in the orientation it was after the previous attempt. E.g., if an object placed

3

in an upright orientation fell off the destination cube and landed on its side, the agent would attempt to place it again, still lying on its side. After ten attempts, the theme object and destination object are re-placed randomly in the environment and sample gathering continues.

Because in a simulation, movements can be hyper-precise (unlike in the real world), even a clearly unstable object like a sphere can remain stably stacked due to a lack of noise or perturbations in the environment. To overcome this and collect more realistic data, we apply a small "jitter" to the theme object once its placement is complete. This jitter is to simulate the small force exerted upon an object when it is released by the grasper and may result in the object wobbling, sliding, or rolling, and potentially falling off the destination cube if it was not placed stably. The direction of the jitter is drawn from VoxWorld's underlying VoxML semantics for objects (Pustejovsky & Krishnaswamy, 2016). For instance, if the theme object is a cylinder, its major axis is the Y-axis (Fig. 3) and so the post-action release jitter is applied perpendicular to the object's local Y-axis. Therefore, this jitter implicitly encodes properties of the object habitats, such as orientations that do or do not facilitate rolling. Objects like cubes or spheres, with no major axis, have the jitter applied in a random direction in the XZ plane.

After every placement attempt, the following numerical information is logged:

- Type of the theme object
- Object rotation in radians at episode start
- Radians between the world upright axis and the object upright (+Y) axis
- Numerical action executed
- Resulting spatial relations between the two objects
- Object rotation and offset from world upright after the action (in radians)
- Vector of the VoxML-derived post-action jitter
- Position of the theme object relative to the destination object, before action, immediately after action, and after the object has had time to either settle in place, or roll off

The resulting data numerically describes the behavior of objects during the stacking "play". We also gather images of all attempts, mostly for visualization purposes (e.g., see Fig. 2). In this paper we focus on reasoning over the numerical data and how it allows us to build AI systems that can perform inference by "embodying" a position in the environment rather than processing images.

4. Object Similarity Analysis

We first attempt to build a model to classify the objects based on their behavior in the freeform stacking play. We use a 4-layer neural network (200, 100, 50, and 25 units respectively) with Leaky ReLU activation. We use weight decay of 0.01. All the raw feature values listed above are the inputs, except for object type, which is the output to be predicted. This model is trained on 14,400 total samples (1,600 per object type) using Adam optimization (Kingma & Ba, 2014) and cross-entropy loss with a learning rate of 0.0001 and batch size of 32, for 200 epochs. When tested on a further 3,600 samples (400 per object), the feedforward network achieves a test accuracy of **89.25%**. Fig. 4 shows the classifier's confusion matrix over 200 of the test samples.

To visualize how the network has clustered these object types, and for an intuitive sense of the relationships between the internal representations of the model, we retrieve the activation of the last hidden layer of the network and perform multidimensional scaling (MDS) (Borg & Groenen, 2005) to visualize the similarity between individual final hidden layer representations. Fig. 5 shows the MDS embeddings of the same aforementioned 200 test samples. The left plot colors the points by the predicted label, and the right plot is colored by the true labels.

Figure 4: Confusion matrix of behavior-based object classifier

The MDS embeddings show that objects that share physical characteristics have similar representations. For example, *sphere* and *egg*, the two most round objects in the object set, are neighbors. The obvious neighbors to the *small cubes* are the larger cubes. The confusions are also illustrative. Figs. 4 and 5 show that one of more commonly confused objects is *cylinder*, it is relatively often confused with i) *capsule*, another cylindrical object, but with rounded ends; and ii) *cone*, another object with a single rounded face. Figs. 4 and 5[R] also show frequent confusion between *cone* and *pyramid*, because due to similar shapes, when they fall off the destination cube and come to rest "sideways," they are in a similar orientation. Therefore we can say the object behavior during the stochastic stacking attempts preserves information about each objects' physical characteristics, relevant configurations or habitats, and afforded behaviors (e.g., stackability or lack thereof). In Krishnaswamy & Ghaffari (2022), we also explored object similarity through canonical correlation analysis (CCA), a linear approach, on the raw data. This paper further investigates the similarity of object types through MDS applied on the nonlinear hidden layer outputs of the neural classifier.
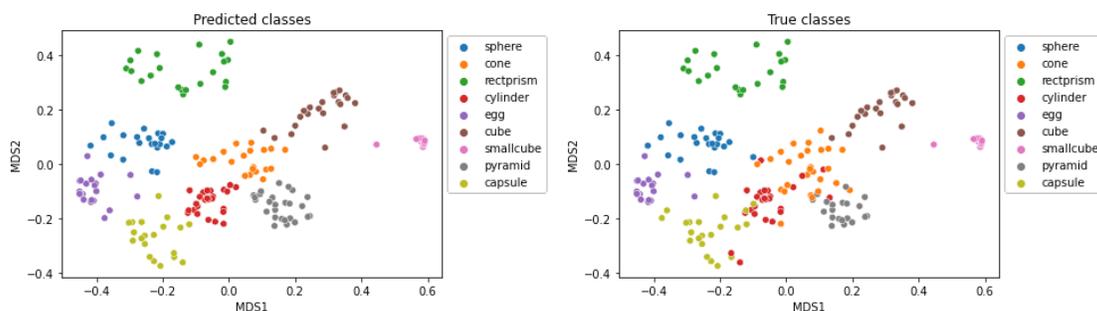

Figure 5: MDS embeddings of last hidden layer activation, showing representational similarity. [L] colored by predicted label; [R] colored by true label.

While the 9-way neural network classifier can solve the object classification problem well enough, these results do not tell us much about how the model is making these distinctions between objects based on their behavior. What features are most important? What concepts has it modeled to make the type distinctions? In addition, most humans, particularly infants and toddlers, can "train" to accommodate a new type of object or item with far fewer examples, even if we allow that repeated encounters with the same item count as distinct stimuli.

5. Transfer Learning to Accommodate New Classes

To facilitate closer investigation of the object properties neural models learn from this embodied data, as well as to explore more sample-efficient models, we use a transfer learning approach. To begin, we train a deep feedforward architecture similar to that used in Sec. 4 on 5,000 total samples of the classes *cube*, *sphere*, and *egg*. The training features used here are the same as those used in Sec. 4. Between them, these three objects have three abstract properties that we hypothesize are important to distinguish between the 9 different objects in the entire dataset: flatness, roundness, and distinctions in axis of rotational symmetry. A model that is properly trained on these three objects should capture important features that can be used to distinguish additional objects as they are introduced. This is analogous to an agent that has not encountered more than these three objects before but can use distinguishing features learned from its vocabulary of 3 objects to make inferences about other objects. Furthermore, this is analogous to an agent with the ability to do multiple tasks: one is identifying objects, the other is identifying concepts such as *round* and *flat*. We performed training on a Linux workstation with a 2.2 GHz Intel Xeon CPU (GPU training was not used). The number of training epochs is set to 100. We tune over 3 learning rates $\{0.001, 0.0001, 0.00001\}$. We early stop and choose the best learning rate using validation accuracy.

Subsequently, one object is added to the object vocabulary at a time. To perform transfer learning from the source model to a model that accommodates one additional object, the first two hidden layers of the source model are frozen so additional training on new objects does not destroy important features learned from the original object set. When transferring the knowledge from a source model to a target model, the target model can perform inference with few samples per object class. The success of this approach comes partly from the shared representation or structure in our data.

In the target model, we add a hidden layer on top of the source model's last hidden layer. As the first two layers are frozen, training this hidden layer better facilitates training convergence when using a low number of samples. The information captured in the source model can be successfully transferred to a target model regardless of number of classes in the target model. We modify the softmax layer and fine tune the source model, trained on $k-1$ objects, to the target model for k objects. With each additional object, the previous target model becomes the source model and this process is repeated. Fig. 6 shows the evaluation results.

We are able to maintain high classification accuracy by incrementally introducing one novel object and fine tuning the source model for that objects, arriving at a final test accuracy of **90%** when all 9 object types are incorporated. Each time we test on a different randomly-selected subset of object samples. When transferring features from source model to target model, we use a constant 600 samples in total. Interestingly, this means that as new objects are added, the number of samples *per object* used for fine tuning goes down, as shown in Fig. 7, suggesting that the base model sufficiently captures important discriminative features, and that only a few samples of new objects
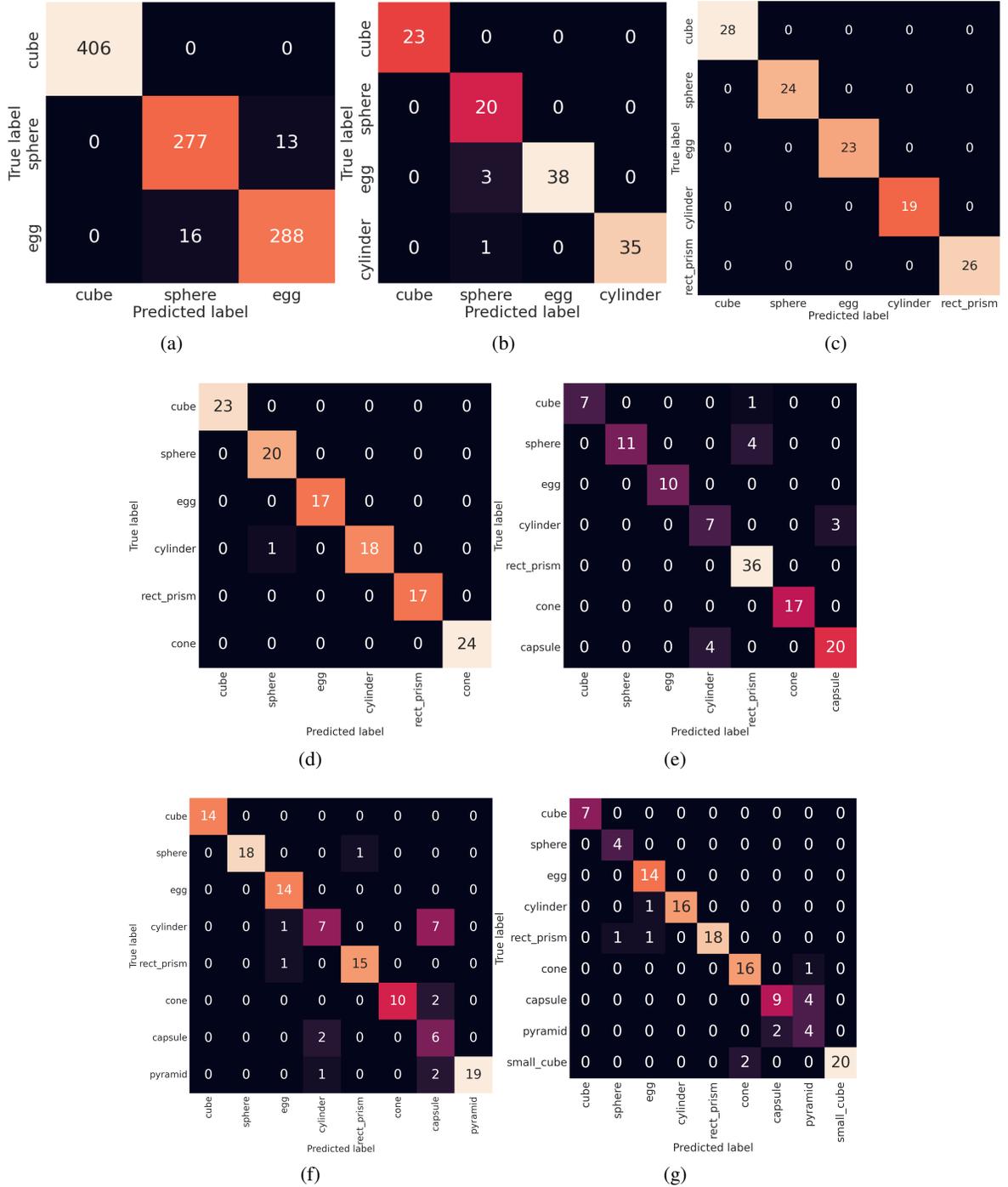
Figure 6: Confusion matrices of transfer-learned models with progressively more object types. (a) base, (b) +*cylinder*, (c) +*rect. prism*, (d) +*cone*, (e) +*capsule*, (f) +*pyramid*, (g) +*small cube*.

are needed for our model to progressively accommodate them. We hypothesize that because the source task starts with objects that collectively have both flat and round features, the ability to distinguish objects based on these features is captured in the initial source task and can be shared in the following target tasks. Therefore we hypothesize that the order of object addition is unlikely to matter much.
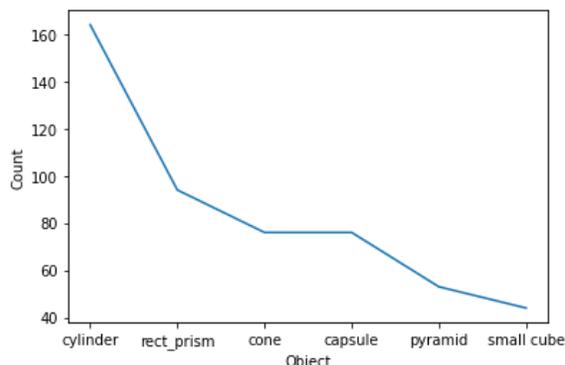
Interestingly, the transfer-learned model (without the small cube) shows similar confusions to the initial feedforward model trained on all nine objects (see Sec. 4). When the small cube is added, these confusions are lessened significantly (compare Figs. 6(f) and 6(g) to Fig. 4). The small cube introduces a height contrast with the larger cube from the base model and, since all other objects are of a similar size to the larger cube, the height of the small cube contrasts with the height of all other objects. Height is not directly recorded in the dataset, but is implicitly encoded in the position of the object, as the Y-coordinate of the small cube is lower than the Y-coordinate of any other object in the same conditions.

Figure 7: fine tuning samples per object.

By the end of the transfer learning process, we have a model with 10 hidden layers that has progressively grown with the incorporation of each new object type. We therefore compare the performance of this model to a 10-layer model whose size remains constant through all transfer learning procedures. We repeat the same transfer learning procedure as above with this model architecture, without adding new layers through the process. Here, the only change is to add a node to the softmax layer.

Fig. 8 shows the results from this model, with a test accuracy of **80.83%**. This static model makes significantly more mistakes than the dynamically growing model, perhaps due to overfitting to cube, sphere, and egg-related features when training a larger base model.
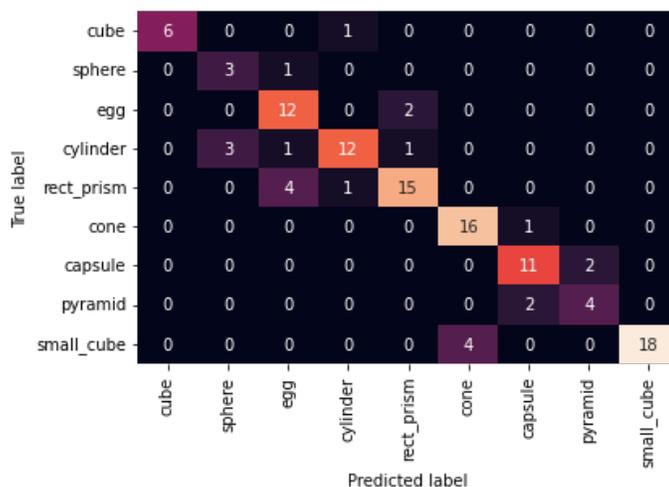

Figure 8: Confusion matrix for 10-layer model of constant size after transfer learning is performed up to 9 object types (cf. Fig. 6(g)).

8

6. Inferring Abstract Concepts

The success of transfer learning in this task demonstrates the effectiveness of the VoxWorld embodied simulation environment for gathering data to enable the metacognitive task of model expansion through transfer learning.

The objects in the dataset, however, are not just instances of multiple different classes, but also display properties and contrasts that inhere across multiple object classes. As discussed earlier, we have both round objects and flat objects, as well as objects like cylinder and cone, that display both roundness and flatness depending on their orientation. In this section, we test the capacity of the previously-learned model to make inferences about these more abstract properties.

In the gathered data, we store the rotation of objects after the placement action is taken. Due to the geometric properties of the object, the rotation is a deterministic indicator of how the object comes to rest. For example, a cylinder resting at either 0 or π radians is upright, on its flat end, while being at either $\frac{\pi}{2}$ or $\frac{3\pi}{2}$ radians means it is horizontal, or resting on its round edge. For a cone, resting at a rotation of 0 radians means it is resting on its flat base, while resting at $\sim \frac{2\pi}{3}$ radians means it is resting on its round side (e.g., see Fig. 2). All other objects are either entirely flat-faced or have only round sides. This means that each individual sample of a stacking attempt can be classified into the object coming to rest on a *flat* side or a *round* side. If the previously-trained object classification model was successful in part due to learning features correlated with roundness or flatness, then it should also be useful for this round side/flat side classification task.

The fine-tuning procedure here reflects the previous procedure for fine tuning for new object classes. We add one additional hidden layer on top of the last hidden layer of the now 10-layer source model (from Fig. 6) to allow for additional nonlinearities between the source model's object-level features and abstract flatness/roundness features, and we fine tune the resulting target model on 300 flat-edge object samples and 300 round-edge object samples, distributed across multiple object types. Fig. 9 shows a test accuracy of **100%** on a further 120 test samples, demonstrating that the features learned during object classification can also be used to discriminate more abstract categories like "roundness" and "flatness."

Figure 9: Confusion matrix for round side vs. flat side classification.

7. Detecting Novel Classes

Previously, we discussed how transfer learning can be used to make a model accommodate new object classes based on their behavior. However, there we always explicitly incremented the number of objects with each transfer learning iteration. In this section, we discuss how a model can automatically detect when a new object class has been introduced into the environment, so that an agent can trigger its own transfer learning procedure to accommodate the new object type. This is a metacognitive procedure where the system detects when its own model is inadequate to the environment, so it can use methods like those discussed in Sec. 5 to adapt to new conditions.

Our methodology involves 1) training a policy to perform a task with a known object type; 2) attempting to use any object presented in the same task using the aforementioned trained policy; 3) observing differences in behaviors of the various objects and using those differences to identify if an instance of an object is sufficiently different from known objects to likely constitute a new class.

At slightly more than 6 months old, most infants appear able to intuit that an object will not fall if supported from the bottom on over 50% of its lower surface (Baillargeon et al., 1992; Dan et al., 2000; Huettel & Needham, 2000; Spelke & Kinzler, 2007). We use VoxWorld's integration with Unity ML-Agents (Juliani et al., 2018), OpenAI Gym (Brockman et al., 2016), and the Stable-Baselines3 reinforcement learning (RL) package (Raffin et al., 2021) to train an RL algorithm that resembles this intuition in the block stacking task. Learning to stack is, of course, not a novel task in the RL community (cf. Lerer et al. (2016), Li et al. (2017), Li et al. (2020), Hundt et al. (2020), just to name a few). While it is a useful task for demonstrating RL algorithms and AI's capability to learn representations of certain physical intuitions, the work we present here also demonstrates how an RL model for this relatively simple task, coupled with embodied simulation, can be used to drive computational implementations of certain metacognitive processes. All training in this experiment was performed using a MacBook Pro laptop with an Intel CPU.

7.1 Policy Training

We first train a TD3 policy (Fujimoto et al., 2018) to stack two equally-sized cubes. One cube is the destination object and the other is the theme object. The scaled action space is a 2D continuous space $[0, 1000] \times [0, 1000]$. By default the optimal action is $(500, 500)$ (though this can be perturbed in VoxWorld to test generalization). The agent's goal is to place the theme block on the destination such that the two-block stack stays up (Fig. 10).

The state space comprises the number of blocks in the stack (an integer 1 or 2), and the 2D center of gravity of the stack (X and Z values only) relative to the center of the destination object. The agent receives a reward of -1 for missing the destination block entirely, of 9 for touching the surface of the destination block but not stacking stably, and up to 1,000 for stacking the two blocks successfully (with a discount of 100 for each additional attempt—i.e., 900 on the second try, etc.).

Figure 10: Unsuccessful (left, center), and successful (right) stacking.

7.2 Policy Evaluation and Data Gathering

Fig. 11 shows reward plots for policies trained using this method. Training the RL policy completes in under 30 minutes. The two policies that we evaluate when gathering data for this task are represented by the left two curves: the **accurate policy**, where the trained policy is very close to the optimal action, and the **imprecise policy** where the trained policy is slightly less well-optimized and the theme cube falls off somewhat more often in testing. In the plot where the reward starts climbing around timestep 700, the action space was perturbed so the optimal policy is far from the center. We do not focus on this policy since downstream results are similar to the others.

We then evaluate the trained policies using sets of different theme objects and gather data about each evaluation. Since the policies were trained to only stack a cube on another cube, this is tantamount to making the agent attempt to stack various objects *as if they are all cubes*.
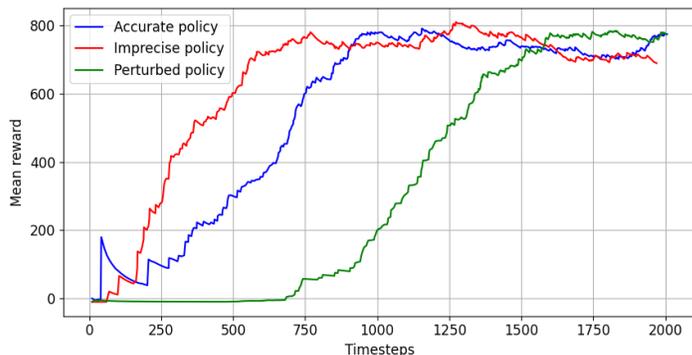

Figure 11: Mean reward vs. timesteps during training.

We evaluate each policy for 1,000 timesteps with the objects *cube*, *sphere*, *cylinder*, *capsule*, and *small cube*. We focus on 5 objects here to examine how the behavior of novel objects can be exposed given a minimal set of objects that preserve distinctions in the "roundness" and "flatness" properties. Fig. 12 shows the evaluation reward plots for the accurate policy (the blue line is the reward after each episode and the orange line is the mean cumulative reward), and show that the cube is the easiest object to stack, followed by cylinder, capsule, and finally sphere. As during training, the agent gets 10 attempts to stack per episode, so stackable objects like cube and cylinder can complete more episodes in 1,000 evaluation timesteps.

During policy evaluation, we gather the same information about each stacking attempt as mentioned in Sec. 3, as well as the reward for the attempt, the cumulative total reward over the episode, the cumulative mean reward over the episode, and the height of the stack (1 or 2 objects) after each attempt. We gathered two datasets, one each using the accurate and imprecise policy.

7.3 Novel Class Detection

We want to be able to give an algorithm a model of a subset of these classes (e.g., cube and sphere) and have it identify that a new type of object (e.g., cylinder or capsule) is different from any of the known classes based on the way it behaves when interacted with (i.e., stacked). We also want to be able to identify that new samples of a known class (e.g., small cubes), are not new classes of objects, but additional instances of a known class. Here we do not consider size as a distinguishing feature in the model, only object behavior when stacked.

To detect a novel class, we 1) identify which known class an object is most similar to; and 2) determine if the new object is different enough from the most similar known class to be considered likely novel. We start by training a classifier on two known classes: cube and sphere, two of the most dissimilar objects according to similarity analysis in Sec. 4 or CCA (Krishnaswamy & Ghaffari, 2022). We use a 1D convolutional neural net. We group inputs by episodes and to maintain a balanced sample, use only the first 90 evaluation episodes, reserving a further 10 as a development set for testing the classifier, and the remainder of the data for detecting novel classes. Since episode length can vary based on the number of attempts, we pad out the length of each input to 10 timesteps, copying the last sample out to the padding length. Therefore an episode where the policy stacked the object successfully on the first try will consist of 10 identical timestep representations, while an
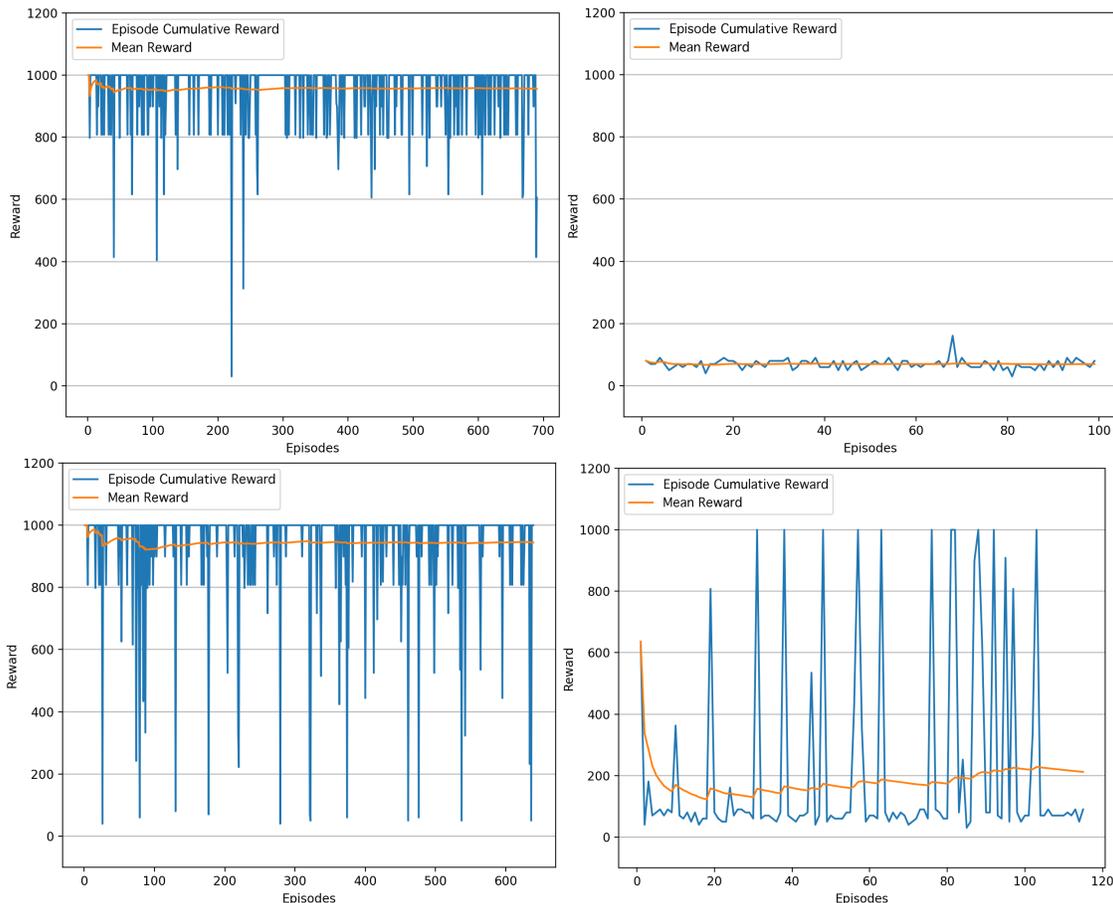
Figure 12: Accurate policy reward plots for stacking (from top left) cube, sphere, cylinder, capsule during evaluation. Episodes are on the X-axis and rewards are on the Y-axis. Plot for small cube is very similar to cube.

episode where the agent tried and failed the stack the object 10 times will have 10 different timestep representations.

The classifier consists of two convolutional layers (256 and 128 units). The filter size in the first layer is c, a variable equal to the number of parameters saved at each evaluation timestep during data gathering ($c = 19$ here) and a stride length of 8, and the second layer uses a filter size of 4 and a stride length of 2. This allows the convolutions to generate feature maps in the hidden layers that are approximately equal to the size of a single timestep sample, and convolving over this approximates observing each timestep of the episode in turn. The convolutional layers are followed by two 64-unit fully-connected layers and a softmax layer. All hidden layers use ReLU activation. We train for 500 epochs using Adam optimization, a batch size of 100 (10 episodes) and a learning rate of 0.001.

We then take batches drawn from classes unknown to the model, e.g., cylinder and capsule, and from additional instances of known classes, e.g., cubes, spheres, and small cubes. The classifier, trained over only two classes, will classify even the non-sphere or cube samples as sphere or cube.

Most commonly, cylinder is classified as cube and capsule is classified as sphere, because these objects' stacking behavior is similar. Small cube is (correctly) classified as a cube.

We then retrieve the 64-dimensional embedding vectors for each sample in the testing batch, and for each sample of the most similar known class. The embedding vectors of the new samples can be compared to embedding vectors of known instances of that predicted class to determine if this new batch is similar enough to truly be the same as the known class or not.

We compute $\vec{\mu_S}$ and $\vec{\sigma_S}$, the mean embedding vector of the known class and the standard deviation of the known class vectors, respectively, as well as $\vec{\mu_N}$, the mean embedding vector of the new batch. Then, assuming that if all samples, new or known, were in fact members of the same class, there would still be some outliers, we find individual outliers in the new batch samples and in the known class samples by dividing the cosine distance between $\vec{\mu_S}$ and the sample $\vec{v}$ in question by the cosine distance between $\vec{\mu_S}$ and $\vec{\mu_S} + \vec{\sigma_S}$. Let $\rho_{\vec{v}} = \frac{cos(\vec{\mu_S},\vec{v})}{cos(\vec{\mu_S},\vec{\mu_S}+\vec{\sigma_S})}$, and if $\rho_{\vec{v}} > 1$, the sample $\vec{v}$ is considered to be an outlier $\vec{o} \in O$, where $\rho_{\vec{o}} = \frac{cos(\vec{\mu_S},\vec{o})}{cos(\vec{\mu_S},\vec{\mu_S}+\vec{\sigma_S})}$. Let O_S be the set of outliers among the samples of the known class S and let O_N be the set of vectors in the new batch N, where $\rho_{\vec{o_N}} > 1$. Outlying samples may still belong to the known class (e.g., sometimes a cube simply fails to stack properly due to bad placement, not its properties, but nonetheless appears to be very different from other cubes in terms of its behavior), so we perform Z-score filtering on the computed outliers, using a Z threshold of 3, and μ_ρ and σ_ρ, the mean and standard deviation, respectively, of the previous computations over the outlier vectors. If $\frac{(\rho_{\vec{o}}-\mu_\rho)}{\sigma_\rho} \geq 3$, $\vec{o}$ is removed from the set of outlier embeddings. For all outliers $\vec{o_N} \in O_N$ that were derived from new batch samples and all outliers $\vec{o_S} \in O_S$ derived from known class samples, we sum $\rho_{\vec{o_N}}$ for all $\vec{o_N} \in O_N$ and divide by the sum of $\rho_{\vec{o_S}}$ for all $\vec{o_S} \in O_S$. This produces an "outlier ratio": $OR = \frac{\sum_{\vec{o_N} \in O_N} \rho_{\vec{o_N}}}{\sum_{\vec{o_S} \in O_S} \rho_{\vec{o_S}}}$

Finally, we multiply the outlier ratio by $cos(\vec{\mu_S}, \vec{\mu_N})$ (the cosine distance between the mean of the known samples and the mean of the new batch), divide that by $cos(\vec{\mu_S}, \vec{\mu_S} + \vec{\sigma_S})$ (the cosine distance between the mean of the known samples and the mean of the known samples plus their standard deviation), and multiply that by the denominator of the outlier ratio. This approximates how many times more dissimilar a given batch is from the mean of the known class than a random sample that falls within the vector subspace spanned by the known class samples would be. Given that a new sample of a known class may not fall exactly within the vector subspace spanned by the samples in the data, we want this dissimilarity threshold to be greater than 1, acknowledging that the subspace defining a class may expand as new samples belonging to that class are encountered. Therefore we define a dissimilarity threshold T, and if $\frac{OR \times cos(\vec{\mu_S}, \vec{\mu_N})}{cos(\vec{\mu_S}, \vec{\mu_S}+\vec{\sigma_S}) \times \sum_{\vec{o_S} \in O_S} \rho_{\vec{o_S}}} > T$ we say that the batch of new samples likely belongs to a class that is not one of the known classes.

7.4 Results and Discussion

We implemented tests where the classifier model was trained to classify different sets of known classes (cube and sphere, cube and sphere and cylinder, cube and sphere and capsule, and all four). We also conducted an experiment where we trained the 1D CNN without the VoxML-derived jitter force feature, which implicitly encodes the axis of symmetry of the theme object, to compare what information this adds to the model. We conducted 10 experiments under each condition with each dataset, using a dissimilarity threshold value of $T = 25$. Correct results were identifying cylinder

13

and capsule as novel classes where they were not already known, and not identifying small cube as a novel class. Fig. 13 shows the aggregate results with confidence intervals.
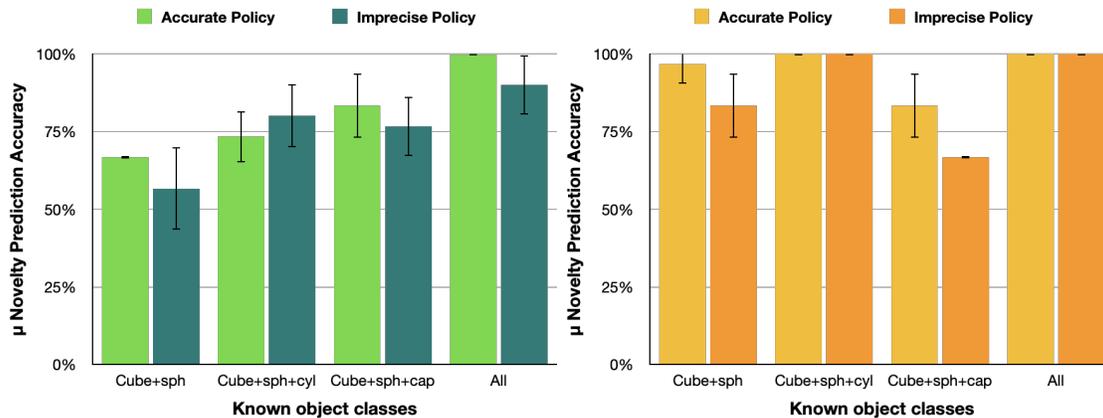

Figure 13: Novel class identification accuracy under each condition. Left chart shows results without the VoxML-derived jitter information, and right chart shows results with it.

We can correctly identify the novelty of cylinders and capsules where appropriate, and identify batches of small cubes as instances of the known cube class, simply based on the way they behave in the stacking task. The imprecise policy data is somewhat more challenging, because even stable objects like cubes fall off the bottom cube more often due to bad placement.

Using the VoxML-derived information in the object classifier boosts novel class detection performance up to 25%. Without it, the model cannot infer which axis the theme object moved along when the release jitter was applied, and so most cylinder embeddings end up nearly identical to cube embeddings. Therefore it seems that this use of VoxML encodes information useful for common-sense reasoning a la Hobbs (1984) into the model.

When cylinder is a known class to the model, it is much easier to identify capsule as a novel class than when the reverse is true, suggesting that order of class acquisition is important. When the VoxML inputs are included in classifier training, our method identifies capsule as a novel class 100% of the time, but when capsule was known first, cylinder was only classified as novel with 75% accuracy or less. We hypothesize that cylinder-to-cube is a much more fine-grained distinction than capsule-to-cube or capsule-to-sphere. Because capsule is more obviously different in its behavior compared to cube or sphere, the capsule vectors take more "space" in the overall embedding space, making it hard to distinguish cylinder embeddings from other classes (usually cubes). However, as in Sec. 5, the source task starts with the two "extremes" *cube* and *sphere*, meaning that the geometric features relevant to detecting round and flat aspects of objects are already present.

Finally, even without the VoxML-derived information available to the model, the novel class detection method can successfully determine that the small cube is not a novel class, as shown by the right two bars in Fig. 13. However, a deeper look into the classifier outputs show that although small cubes are being correctly labeled "not novel," the most similar class they are subsumed into is often not *cube* but *cylinder*. Fig. 14 shows the CNN classifier outputs over the 10 episode dev-test set, aggregated over all 10 novel concept detection experiments. The two confusion matrices on the left show classification results without the VoxML-derived inputs. The two on the right show results

with those inputs. The top two are from the *accurate policy* evaluation, and the bottom two are from the *imprecise policy* evaluation. Without the VoxML-derived inputs, we see frequent confusion between sphere and capsule, and between cube and cylinder. The VoxML-derived inputs are critical to correctly classifying the behavioral distinctions between objects like cubes and cylinders in the first place, in order to assess these classes' novelty.
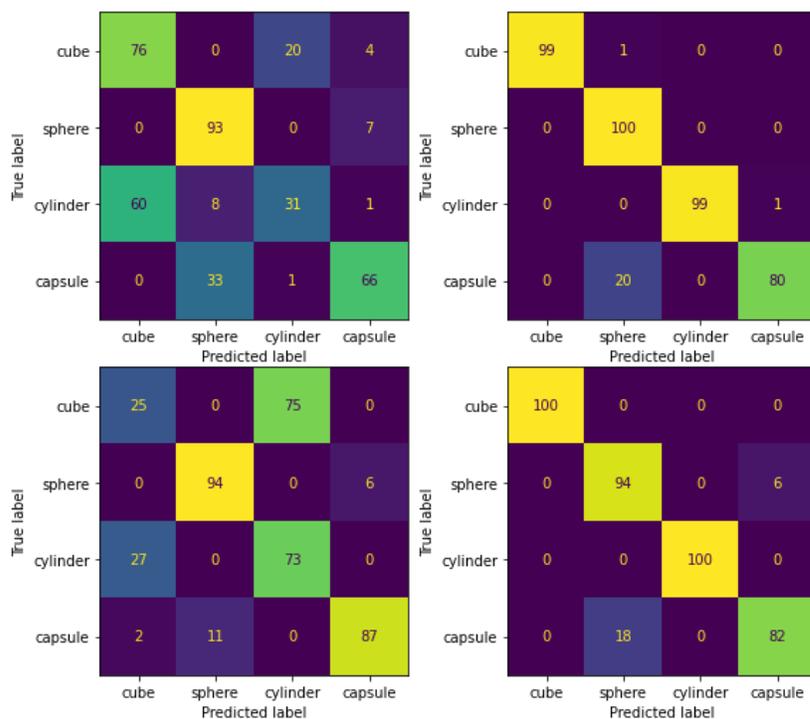

Figure 14: Aggregated CNN outputs over the dev-test set.

The outputs of this procedure could be extended into the reinforcement learning itself, where if the policy learned on cubes fails in an environment containing a novel object, the policy itself is revised to allow the object to be stacked successfully (such as making sure all cylinders are placed upright). This is the subject of future work in this area.

8. Conclusion

Metacognition is defined as an actor's awareness and understanding of its own thought processes and the patterns underlying them. More simply it may be called "thinking about thinking" (Blakey & Spence, 1990). A metacognitive agent is one that, among other capacities, analyzes its own model of the environment and world, including the individual concepts it considers therein, figures out when it needs to be updated, how, and why. In this paper we have demonstrated a certain subset of these capabilities in ML-based cognitive systems, in a domain intentionally reminiscent of geometric children's toys.

The neural network learns important distinguishing geometric features of the objects based on their behavior, and these features can be nonlinearly recombined to recognize new objects (this is demonstrated convincingly, but not necessarily conclusively, by adding new hidden layers). Embeddings based on these features can be used to detect outliers, and statistical techniques can be used to group outliers together into novel distinct classes.

Interestingly, in our work, none of the individual components—the neural network, the environment model, the analysis of embedding spaces, or the statistical metrics—can definitively be said to be solely responsible for the capabilities we demonstrate herein. It is the hybrid approach and

combination of techniques that led us to this result. Ablation tests of the individual components to quantify the contribution of each is the topic of ongoing research.

The applicability of this same pipeline to other domains and shapes is to be determined, but the fact that distinctions can be made at the embedding level (even for objects unknown to the classifier) suggests that once a vector representation is acquired for an input sample of potentially any modality, similar assessments can be made over it.

The current work is limited primarily by the stacking task used to expose object properties, which in particular limits the number of objects with distinctions that can be revealed through this task. The key features of flatness vs. roundness can be exposed through this task, as can other implicit distinctions such as length distinctions along an axis. However other distinctions, such as those between irregular shapes, may require other tasks to expose. These may be simple variants such as reversing the order of stacking and varying the destination object class instead of the theme. But other, significantly different tasks may be required, such as fitting objects into the correctly shaped hole. This is similar to how toddlers learn about different object properties through different activities.

Finally, unlike human learning, most modern AI techniques require very large volumes of data, long training times, or specialized (often expensive) hardware. The models and processes we have developed here are small, lightweight, and can be trained or performed on as little as a laptop CPU, though GPU training provides a speed benefit for use in interactive systems.

9. Future Work

Particular points of future work in the line of research include estimating the dimensionality of the representations that are being transferred during the model expansion process. Since we add one layer to detect each new object, this necessarily raises concerns about the scalability of the approach. However, we surmise that the 25 dimensions used in the final layer of the transfer-learning neural network may not all be necessary to store all the information transferred between networks, or, if they are, then a larger embedding size may facilitate transfer learning between multiple types while not requiring the addition of a new layer each time. In addition, we plan to investigate the number of

Figure 15: Proposed integration architecture.

layers that need to be frozen for successful transfer learning (e.g., is freezing one layer enough, or would freezing three be better, or is the number of layers perhaps a function of the number of classes or the domain?). We also plan to introduce more shapes, including more irregular ones (e.g., a rock crystal). However, with the introduction of new shapes comes a need for new tasks besides stacking to expose the distinctions between them.

The two suites of experiments herein were conducted separately, though using data from the same environment in variants of the same task. The primary goal of future work in this line of research is to integrate the two methods. This involves using the novelty detection method to provide
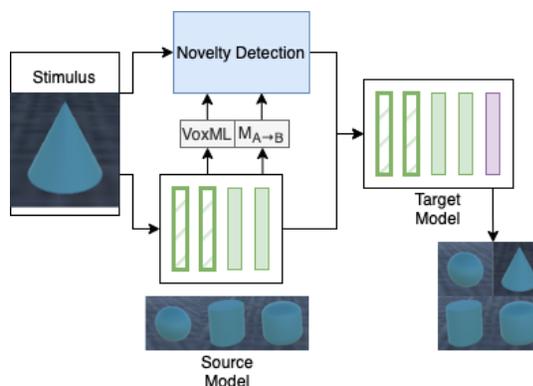
the transfer learning model with an object that satisfies the detected set of novel properties such that the transfer learning model can be updated to achieve better accuracy with the new object set. Since these currently are trained using different networks, the representations from the two networks need to be unified, either through a symbolic layer like VoxML, or by creating a mapping matrix M between the two models A and B by solving for an affine transformation between paired embeddings (McNeely-White et al., 2020). Fig. 15 shows this proposed architecture. This integration facilitates the execution of both novelty detection and transfer learning in real time for use in interactive systems such as Scheutz et al. (2019) or Krishnaswamy & Pustejovsky (2022).

Acknowledgements

References

Baillargeon, R., Needham, A., & DeVos, J. (1992). The development of young infants' intuitions about support. *Early development and parenting*, *1*, 69–78.

Bar-Aviv, E., & Rivlin, E. (2006). Functional 3D Object Classification Using Simulation of Embodied Agent. *BMVC* (pp. 307–316). Citeseer.

Beniaguev, D., Segev, I., & London, M. (2021). Single cortical neurons as deep artificial neural networks. *Neuron*, *109*, 2727–2739.

Blakey, E., & Spence, S. (1990). *Developing metacognition*. ERIC Clearinghouse on Information and Technology.

Borg, I., & Groenen, P. J. (2005). *Modern multidimensional scaling: Theory and applications*. Springer Science & Business Media.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). OpenAI Gym. *arXiv preprint arXiv:1606.01540*.

Brown, T., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, *33*, 1877–1901.

Caligiore, D., Ferrauto, T., Parisi, D., Accornero, N., Capozza, M., & Baldassarre, G. (2008). Using motor babbling and Hebb rules for modeling the development of reaching with obstacles and grasping. *International Conference on Cognitive Systems* (pp. 22–23). Citeseer.

Caruna, R. (1993). Multitask learning: A knowledge-based source of inductive bias. *Machine learning: Proceedings of the tenth international conference* (pp. 41–48).

Clark, A. (2006). Language, embodiment, and the cognitive niche. *Trends in cognitive sciences*, *10*, 370–374.

Colung, E., & Smith, L. B. (2003). The emergence of abstract ideas: Evidence from networks and babies. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*,

358, 1205–1214.

Dan, N., Omori, T., & Tomiyasu, Y. (2000). Development of infants' intuitions about support relations: Sensitivity to stability. *Developmental Science*, *3*, 171–180.

DiCarlo, J. J., & Cox, D. D. (2007). Untangling invariant object recognition. *Trends in cognitive sciences*, *11*, 333–341.

Fitzgerald, T., Goel, A., & Thomaz, A. (2021). Abstraction in data-sparse task transfer. *Artificial Intelligence*, *300*, 103551.

Frasca, T., Oosterveld, B., Krause, E., & Scheutz, M. (2018). One-shot interaction learning from natural language instruction and demonstration. *Advances in Cognitive Systems*, *6*, 1–18.

Fujimoto, S., Hoof, H., & Meger, D. (2018). Addressing function approximation error in actor-critic methods. *International conference on machine learning* (pp. 1587–1596). PMLR.

Gibson, J. J. (1977). The theory of affordances. *Hilldale, USA*, *1*, 67–82.

Gopnik, A. (2010). How babies think. *Scientific American*, *303*, 76–81.

Gopnik, A. (2012). Scientific thinking in young children: Theoretical advances, empirical research, and policy implications. *Science*, *337*, 1623–1627.

Gopnik, A., & Meltzoff, A. N. (1997). *Words, thoughts, and theories*. Mit Press.

Gottlieb, J., & Oudeyer, P.-Y. (2018). Towards a neuroscience of active sampling and curiosity. *Nature Reviews Neuroscience*, *19*, 758–770.

Hobbs, J. (1984). *Sublanguage and knowledge*. Jun.

Huettel, S. A., & Needham, A. (2000). Effects of balance relations between objects on infant's object segregation. *Developmental Science*, *3*, 415–427.

Hundt, A., Killeen, B., Greene, N., Wu, H., Kwon, H., Paxton, C., & Hager, G. D. (2020). "Good robot!": Efficient reinforcement learning for multi-step visual tasks with sim to real transfer. *IEEE Robotics and Automation Letters*, *5*, 6724–6731.

Juliani, A., et al. (2018). Unity: A general platform for intelligent agents. *arXiv preprint arXiv:1809.02627*.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Knudsen, E. I. (1994). Supervised learning in the brain. *Journal of Neuroscience*, *14*, 3985–3997.

Krishnaswamy, N., & Ghaffari, S. (2022). Exploiting Embodied Simulation to Detect Novel Object Classes Through Interaction. *arXiv preprint arXiv:2204.08107*.

Krishnaswamy, N., Pickard, W., Cates, B., Blanchard, N., & Pustejovsky, J. (2022). The VoxWorld Platform for Multimodal Embodied Agents. *Proceedings of the Language Resources and Evaluation Conference* (pp. 1529–1541). Marseille, France: European Language Resources Association.

Krishnaswamy, N., & Pustejovsky, J. (2022). Affordance Embeddings for Situated Language Understanding. *Frontiers in Artificial Intelligence*, (p. 161).

Lerer, A., Gross, S., & Fergus, R. (2016). Learning physical intuition of block towers by example. *International conference on machine learning* (pp. 430–438). PMLR.

Li, R., Jabri, A., Darrell, T., & Agrawal, P. (2020). Towards practical multi-object manipulation using relational reinforcement learning. *2020 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 4051–4058). IEEE.

Li, W., Bohg, J., & Fritz, M. (2017). Acquiring target stacking skills by goal-parameterized deep reinforcement learning. *arXiv preprint arXiv:1711.00267*.

Markant, D. B., & Gureckis, T. M. (2014). Is it better to select or to receive? Learning via active and passive hypothesis testing. *Journal of Experimental Psychology: General*, *143*, 94.

McNeely-White, D., Sattelberg, B., Blanchard, N., & Beveridge, R. (2020). Exploring the interchangeability of CNN embedding spaces. *arXiv preprint arXiv:2010.02323*.

Meichenbaum, D. (1985). Teaching thinking: A cognitive-behavioral perspective. In *Thinking and learning skills*, 419–438. Routledge.

Mohan, S., Mininger, A. H., Kirk, J. R., & Laird, J. E. (2012). Acquiring grounded representations of words with situated interactive instruction. *Advances in Cognitive Systems*. Citeseer.

Najemnik, J., & Geisler, W. S. (2008). Eye movement statistics in humans are consistent with an optimal search strategy. *Journal of Vision*, *8*, 4–4.

Neftci, E. O., & Averbeck, B. B. (2019). Reinforcement learning in artificial and biological systems. *Nature Machine Intelligence*, *1*, 133–143.

Nelson, J. D., McKenzie, C. R., Cottrell, G. W., & Sejnowski, T. J. (2010). Experience matters: Information acquisition optimizes probability gain. *Psychological science*, *21*, 960–969.

Niv, Y. (2009). Reinforcement learning in the brain. *Journal of Mathematical Psychology*, *53*, 139–154.

Nolfi, S. (2005). Category formation in self-organizing embodied agents. In *Handbook of categorization in cognitive science*, 869–889. Elsevier.

Oliva, A., & Torralba, A. (2007). The role of context in object recognition. *Trends in cognitive sciences*, *11*, 520–527.

Piaget, J. (1963). The attainment of invariants and reversible operations in the development of thinking. *Social research*, (pp. 283–299).

Piaget, J., & Inhelder, B. (2008). *The psychology of the child*. Basic books.

Piloto, L. S., Weinstein, A., Battaglia, P., & Botvinick, M. (2022). Intuitive physics learning in a deep-learning model inspired by developmental psychology. *Nature human behaviour*, (pp. 1–11).

Pratt, L. Y., Mostow, J., Kamm, C. A., Kamm, A. A., et al. (1991). Direct Transfer of Learned Information Among Neural Networks. *Aaai* (pp. 584–589).

Pustejovsky, J. (2013). Dynamic event structure and habitat theory. *Proceedings of the 6th International Conference on Generative Approaches to the Lexicon (GL2013)* (pp. 1–10).

Pustejovsky, J., & Krishnaswamy, N. (2016). VoxML: A Visualization Modeling Language. *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)* (pp. 4606–4613).

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., & Dormann, N. (2021). Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*.

Renninger, L. W., Verghese, P., & Coughlan, J. (2007). Where to look next? Eye movements reduce local uncertainty. *Journal of vision*, *7*, 6–6.

Riesenhuber, M., & Poggio, T. (2000). Models of object recognition. *Nature neuroscience*, *3*, 1199–1204.

Scheier, C., & Pfeifer, R. (1999). The embodied cognitive science approach. In *Dynamics, synergetics, autonomous agents: Nonlinear systems approaches to cognitive psychology and cognitive science*, 159–179. World Scientific.

Scheutz, M., Williams, T., Krause, E., Oosterveld, B., Sarathy, V., & Frasca, T. (2019). An overview of the distributed integrated cognition affect and reflection DIARC architecture. *Cognitive architectures*, (pp. 165–193).

Schick, T., & Schütze, H. (2020). It's not just size that matters: Small language models are also few-shot learners. *arXiv preprint arXiv:2009.07118*.

Schulz, L. E., & Bonawitz, E. B. (2007). Serious fun: preschoolers engage in more exploratory play when evidence is confounded. *Developmental psychology*, *43*, 1045.

Son, L. K., & Sethi, R. (2006). Metacognitive control and optimal learning. *Cognitive Science*, *30*, 759–774.

Spelke, E. S., & Kinzler, K. D. (2007). Core knowledge. *Developmental science*, *10*, 89–96.

Vlach, H. A., & Sandhofer, C. M. (2012). Fast mapping across time: Memory processes support children's retention of learned words. *Frontiers in psychology*, *3*, 46.

Zador, A. M. (2019). A critique of pure learning and what artificial neural networks can learn from animal brains. *Nature communications*, *10*, 1–7.