AIGenC: AI Generalisation via Creativity

Corina Cătărău-Cotuțiu[1,*][0000−0002−9186−3092], Esther Mondragón[1][0000−0003−4180−1261], and Eduardo Alonso[1][0000−0002−3306−695X]

Artificial Intelligence Research Centre (CitAI), Department of Computer Science City, University of London

Abstract. Inspired by cognitive theories of creativity, this paper introduces a computational model (AIGenC) that lays down the necessary components to enable artificial agents to learn, use and generate transferable representations. Unlike machine representations, which rely exclusively on raw sensory data, biological representations incorporate relational and associative information that embed a rich and structured concept space. The AIGenC model poses a hierarchical graph architecture with various levels and types of representations procured by different components. The first component, Concept Processing, extracts objects and affordances from sensory input and encodes them into a concept space. The resulting representations are stored in a dual memory system and enriched with goal-directed and temporal information acquired through reinforcement learning, creating a higher-level of abstraction. Two additional and complementary components work in parallel to detect and recover relevant concepts through a matching process and create new ones, respectively, in a process akin to cognitive Reflective Reasoning and Blending. If Reflective Reasoning fails to offer a suitable solution, a blending operation creates new concepts by combining past information. We discuss the model's capability to yield better out-of-distribution generalisation in artificial agents, thus advancing toward Artificial General Intelligence.

Keywords: Affordances · Generalisation · Creativity · Representational Learning · Reinforcement Learning · Learning Transfer.

1 Introduction

AI systems struggle to adapt to novelty and often fail to exploit past learnt experiences effectively. This downside is due to AI systems' poor representational learning capability to acquire meaningful relational concepts. E.g., Large Language Models must parse each word in a wide variety of embeddings to generalise it to new contexts, requiring considerable combinatorial augmentation for a *tabula rasa* unstructured model to learn systematic representations.

We hypothesise that Creative Problem Solving (CPS) could help AI systems learn versatile concepts that can be adapted to novel situations, a step forward to solving the problem of generalisation in AI. Humans and other animals can naturally map previous experiences to new situations, transferring responses to

similar scenarios [15]. Machines do not. In the natural world, stimulus generalisation is driven by common sensory attributes, which are often embedded within irrelevant cues and may lead to dysfunctional use of information. However, stimulus generalisation is only a fraction of human transfer capabilities; associations can also mediate generalisation to dissimilar cues, bridging learning across different sensory dimensions. Crucially, the role played by commonalities can be extended to different levels of information [13]. Extracting and storing patterns and relations capable of bearing resemblance across situations beyond the sensory word would permit hierarchical structuring of information and creative, non-passive reuse of data to transfer knowledge.

As Deep Neural Networks (DNNs) become ubiquitous, concerns arise regarding the nature and lack of structure of the representations they learn and how these affect their decision making. DNNs excel at extracting regularities from training data but struggle to extrapolate or generalise to out-of-distribution settings. This discrepancy is due to using shortcut strategies to solve a task without learning the true input structure. Shortcuts involve data interpolation –meaningless, ad-hoc parameter adjusting to reduce the error– and are, thus, highly dependent on the training dataset. Systems operating on shortcuts may appear to perform as if they had learnt abstract concepts and could potentially transfer learning. Still, their performance relies on highly tuned vast amounts of training features that do not support extrapolation.

Drawing inspiration from CPS's theory, we have designed the modelling components necessary for cross domain performance in a naïve Reinforcement Learning (RL) agent. Unlike standard deep RL architectures, which merely reduce the complexity of the environment by compressing raw input, our approach endows artificial agents with core concepts –abstract representations that hold meaning beyond the training context [12], at different hierarchical levels.

Core concepts [16] include different types of data - objects, events and properties - and are the building blocks of meaning and so-called common sense [19]. Although task transfer does not inherently necessitate creativity, for a system to display human-like behaviour, it is critical to adapt existing concepts to new tasks –effectively, to learn new concepts based on previous knowledge in what is called displacement or creative transfer. Displacement requires creative reuse of information, i.e., a transfer mechanism capable of generating different concept relations, not just appending previous ones. Moreover, when a problem cannot be solved solely based on existing knowledge and approximating a solution is not viable, it is necessary to give rise to new concepts.

Implicit relations learned by DNNs [4] are insufficient to generate meaningful concepts; additional explicit representations of relations, affordances and temporal information are needed to capture the world's content and dynamics and achieve flexible cross-domain generalisation. Implementing these different types of knowledge may bestow RL agents with something akin to common sense. This capability would allow them to infer hidden information, such as intentions or goals, by interacting with the environment. In traditional RL scenarios, agents disregard by design most of the information provided by the environment, learn-

ing simple policies linked to global states. Although, in theory, states in RL can represent any type of information, in practice, RL implementations work primarily on states in which only simple sensory data is encoded. In so doing, states are monolithic entities that do not permit concept manipulation and transfer. We argue that to generalise concepts, an agent cannot rely on raw sensory information alone; instead, it must learn by trial and error the functional and contextual information accompanying them.

The AIGenC model postulates an adaptable concept space that encodes objects, affordances and relational information from environmental features into a deep RL environment. The main characteristic that sets aside our framework is declaring these different dimensions of knowledge for matching information at different levels of abstraction.

2 Functional creativity, concept space and affordances

Concepts are described as entities with the necessary and sufficient conditions for assigning membership of concept X to a category Y[3]. A common premise in classic creativity theories [5, 7, 16] is that of a concept space as a bridge between sub-symbolic and symbolic representations. Gärdenfors' approach of a conceptual framework, where concepts express properties across multiple dimensions that evolve in time and are ascribed to different domains [7] can be adapted to an artificial setup. To do so, a knowledge engineer must provide a finite initial set of features - a task that becomes unscalable as the number of features increases. Alternatively, DNNs could be used to extract pertinent features. However, defining concepts on DNNs features alone is insufficient to capture the world's relational complexity. Instead, we define a hierarchical representation of the concept space where environmental features form low-level concepts and represent higher-level concepts as combinations of the former instead of posing the traditional cognitive categories. The idea is to set a single unit (the concept) as a structure that grows hierarchically, enabling the application of the same algorithm at different levels of abstraction. The hierarchy (Fig.1) builds as follows: static concept features as those captured in standard deep RL frames form the bottom (Layer 0). Above it (Layer 1), base object concepts are represented as graph nodes whose edges establish their relations as affordances. Next, at the top (Layer 2), we have a higher-level graph representation whose nodes reproduce the previous graph structure, and the edges capture their temporal succession.

To build such a hierarchy, an RL agent must first select the base concepts' features critical to the task and valuable for encoding in the latent space, by interacting with its environment autonomously. The first component of our framework is thus unsupervised learning of basic concepts. Autoencoder (AE) based models are of interest because they learn low-dimensional representations from high-dimensional distributions by encoding and decoding the input, inducing features from raw perceptual data. These architectures can gather basic concepts and implicit relations as weight matrices that act as relational units between the different features, capturing the functional relationships that impact

the model's behaviour [4]. However, being implicit, the relations cannot be interchanged without modifying the representation itself. Therefore, we posit the need to additionally encode explicit relations for inter-concept interactions.

Gibson introduced the term *affordance* to describe that certain states enable an agent to perform certain actions in a given context [8]. Formally, affordances are defined as relations in the agent-environment system rather than as properties. To learn affordances, an agent must be able to interact with objects to assess their possible uses. Sensory similar concepts are not bound to behave the same way. A beach ball and a bowling ball may look similar, but their weights will render them functionally divergent to, e.g., balance a lever. Therefore, we must enrich the hierarchical concept space by including affordances to enable our RL agent to form various complex representations in a given functional, spatial and temporal context, i.e., to acquire knowledge about object manipulation. The resulting structure, represented as a graph of concept nodes and affordance edges, captures the dynamics of the world. This hierarchical structure aligns with existing research [16]. However, instead of resorting to knowledge bases, we advance a CPS algorithm that uses deep learning methods to manipulate concepts in the latent space.

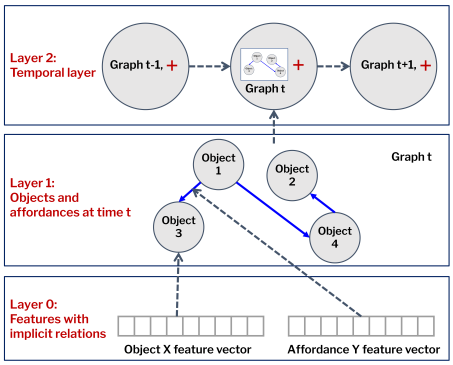

Fig. 1: Hierarchical concept space: object concepts are represented by feature vectors at the bottom level (Layer 0). At the middle level (Layer 1), nodes and edges represent object concepts and affordances. At the top level (Layer 2), the graphs from the previous level become nodes along with a reward given to the agent in the RL setting (red +) in a graph whose edges mark temporal succesion (t).

3 A framework for concept transfer and functional creativity

AIGenC seeks to lay the fundamental structures and interplay necessary for good learning transfer in RL systems, not to present a particular implementation. This section describes these units and interactions in detail.

The model posits a three-component deep learning structure: Concept Processing, Reflective Reasoning, and Blending. The first two comprise two subcomponents each: Concept Processing involves object discovery and affordance learning, while Reflective Reasoning consists of Long Term Memory (LTM) initialisation and Selective Matching (Fig.2).

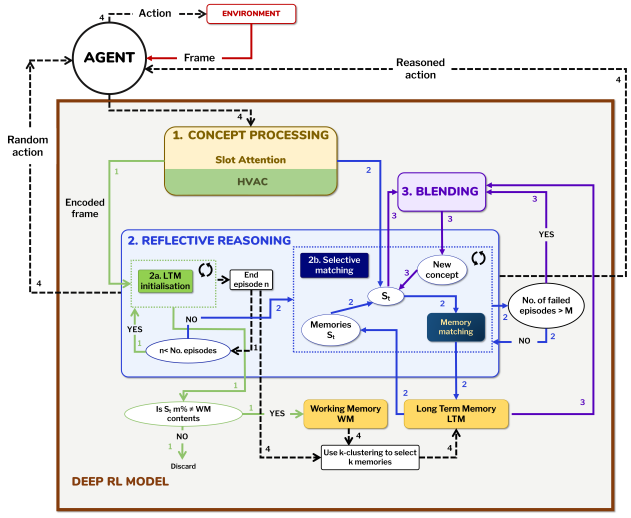

Fig. 2: The three components of the model: (1) Concept Processing (cream/olive), (2) Reflective Reasoning (blue), (3) Blending (purple) and the algorithmic flow. The input (frames) is encoded in (1) as vector representations by two unsupervised models. In (2), LTM is initialised by random exploration (2a). The population of WM and LTM is indicated by open tip arrows, a solid green (1) and a dashed black line (4), respectively. After LTM initialisation, Selective Matching (2b) is activated within a Deep RL training. (2b) returns from LTM the concepts most similar to the current state, which are then incorporated into it before inputted to the Deep RL model. If the agent is unsuccessful at solving the task for several episodes, Blending (3) is triggered, by which novel concepts are formed and then added to the current state.

The framework runs on top of a basal deep RL training that relies on a hierarchical concept space. The latter's structure exploits both implicit and explicit relational information to support flexible knowledge transfer. To save resources and acknowledge the influence of phylogenetic and ontogenetic evolution, implementations may need to pre-train the representational architecture. The first component learns different representations extracted at each time-step, constituting an RL state. States are then added to a Working Memory (WM) based on their novelty and used together with LTM by the Reflective Reasoning component to select the next action. The term *reflective reasoning* defines the main functionality of the second component, namely, a comparison process between stored knowledge and current input intended to adaptively select and combine valuable information for the task at hand. Finally, the agent creates new concepts using the Blending component when available concepts do not overcome a standstill in solving a task.

3.1 Deep reinforcement learning

RL tasks have three main components: states, actions, and rewards. RL agents learn by interacting with the environment and receiving direct feedback through

rewards. An agent learns to optimise a policy by trial and error to reach a goal state by maximising the cumulative reward. States are representations of the world and inform the agent of which situation it currently is in. In Deep RL, a subfield of machine learning that combines RL and deep learning, agents process the environment by passing frames to DL models as input and choosing an action based on the encodings of those frames. These encodings do not usually distinguish between core features (i.e., features that serve as discriminative cues that control learning) and features that are irrelevant to the task at hand (e.g., image quality, colour), and forgo most of the temporal data required for complex multi-level representations. Our model broadens the scope of these representations capturing objects, affordances, and temporality. It does so by outlining a mechanism to filter and extract granular and varied information by decomposing the sequence of frames into multiple layers of the concept space and storing them in a memory system. In so doing, the agent no longer relies solely on current static frame data to make its decisions.

3.2 Concept processing component

A concept processing unit is postulated for encoding and adding the input to the concept space. It comprises two pretrained subunits that process separately sensory (objects) and dynamic information (affordances). Training needs to be supervised to ensure that the RL agent's concept space is independent of external classes and that representations of objects/concepts can be extracted by an agent regardless of whether they have been encountered before.

The first sub-component represents objects as latent vectors through unsupervised object discovery, while the second encodes action vectors. The interaction of the two subcomponents and the environment is posed as a modification of Şahin and collaborators' formalism [18], so that affordances can be defined as an acquired relation between an *(effect, reward)* pair and a *(concept-object, action)* tuple. In this manner, when an agent applies an action to the object, an effect and associated reward pair is generated.

AE based models could be used to ensure transferable and detachable concept representations. Filtering is also needed to reduce irrelevant information to the task and avoid computational explosion. The most valuable objects out of N can be selected using a slot attention module, which maps N latent representations to K slots through an iterative attention mechanism [21].

While DL has achieved remarkable results in representing objects, unsupervised affordance learning remains unsolved, and few unsupervised feature learning methods for affordances can be found [9]. A formal model for affordance learning can be defined by incorporating affordances in a relational concept space using a Hierarchical Variational Autoencoder (HVAC) architecture [10]. While HVACs capture the evolution of interactions, the encodings of the dynamics produced are not disentangled from the objects interacting, which is a limitation. Hence, we will not refer to the output of the HVAC model as an affordance but as an action representation. The action representation and the effect will form the affordance (See Algorithm 1 line: 21-22).

Algorithm 1 AIGenC integrated within an RL setup

Intialising LTM ltm = <keys[], >
$object-discovery$(frame) $\rightarrow$ $Slots^{(KxM)}$ - a pre-trained object discovery model, where K is the number of slots and M the size of the encoding vector
$action-encoding$(interactions) $\rightarrow$ $Actions^{KxP}$ - a pre-trained action encoding model, with input sequences of frames of the agent interacting with an object and output K encodings of size P of those interactions
Let A be a set of objects interacting with the agent
create-state-graph(s_t,ltm) - a function that returns the current state graph
$matching(mem,G_t)$ $\rightarrow$ T/F - a function that checks if current state graph exists in memory
for $iteration = 1, 2, \ldots, N$ **do**
    Intialising WM $working-mem$ = <keys[], >
    Let A_i be a random subset of A
    while $episode$ not done **do**
        Let s_t be the current state of the environment
        objects $\leftarrow$ $object-discovery(s_t)$
        action-encodings $\leftarrow$ $action-encoding(A_i)$
        $G_t \leftarrow$ create-state-graph(objects,action-encodings, s_t)
        if memory initialisation phase **then**
            $a_{i,t} \leftarrow random(actions)$
        else
            $a_{i,t} \leftarrow$ Run policy-network $\pi(G_t)$
        end if
        s_{t+1},reward $\leftarrow$ Take action a_i
        Compute effect $f(a_i, s_t, s_{t+1})$ $\rightarrow$ encoding(s_{t+1}) - encoding(s_t)
        Update G_t with (s_t, s_{t+1}, effect)
        if not matching(wm,G_t) **then** Add(G_t,wm) ▷ Update WM
        else discard()
        end if
    end while ▷ Optimise policy network, update LTM
    WMclusters $\leftarrow$ KMeans(wm)
    centroids $\leftarrow$ centers(WMclusters)
    if not $matching$(ltm,centroids) **then** Add(centroids-graphs, ltm)
    end if
end for

Concept space The concept space is structured hierarchically from representations extracted at the Concept Processing unit. The structured data is necessary for selecting and matching concepts in the Reflective Reasoning component. A three-level structure is used to organise knowledge as concepts: the feature space, the concept level, and the graph representation of an RL state-time configuration that defines the problem template [16]. The structure can be characterised as a graph that stores and relates information without a predefined data design and satisfies theoretical prerequisites such as being hierarchical and adaptive [5, 7]. Graphs can represent different levels of hierarchical abstraction and capture relations between elements. They are also easily manipulated (i.e., adding or removing edges without altering the whole graph), making them the most promising means for representing a dynamic concept space. Graph representations allow for flexibility, both vertical (i.e., changes in levels of abstractions in the hierarchy) and horizontal (i.e., expansions within a level by adding or removing nodes). In addition, the flexibility required by contextual and time-dependent functional classes in creative problem-solving is allowed by dynamic graphs that are stored in memory systems as adjacency lists as opposed to matrices. As such, each node has associated a list of related nodes of varying sizes, enabling partially connected graphs.

Memory system Our framework includes Working Memory (WM) and Long Term Memory (LTM) units. These two memories must interact throughout the lifetime of an RL agent. Each memory serves a particular purpose. First, WM stores current information, if it does not already exist (Algorithm 1, line: 23), during each episode in an RL setup. Once the agent reaches the terminal state (when the goal is reached or after a time limit is exceeded), the episode ends and WM is cleared, at which point the most representative states (the centroid of a data cluster) and their associated concepts are selected from the WM using k-clustering and stored in LTM. As opposed to WM, the content of the LTM must persist throughout episodes to permit transfer of learning during the lifetime of an RL agent (Algorithm 1, line: 1). The memories' content, the concept space, is assembled as two lists: an object list and a hashmap list. The object list records the extracted object representations, while the hashmap list maps each object to a corresponding list of affordances. Once the concept space is formed, an agent can interact with it throughout its creative learning process in both of the remaining components of the architecture, Reflective Reasoning and Blending.

3.3 Reflective reasoning component

The Reflective Reasoning component defines a procedure to choose which concepts from WM are to be permanently stored in LTM and incorporates a matching operation between the current state and LTM information (see Algorithm 2, line: 7). Matching allows selecting the concepts useful to fulfil the specific task at any given time point.

Processing in the Reflective Reasoning component is executed sequentially (Fig.2). First, LTM is initialised; then, selective matching occurs (Algorithm 2, line: 7-9). The agent first explores the environment randomly (using a random action selection policy, Algorithm 1, line: 16), learning new concepts at each state and storing the most representative ones in LTM. The interplay of the two memory systems enables a process of matching novel states to past information [4]. Concepts that reach LTM are selectively retrieved to enhance the agent's ability to choose an action leading to a state closer to the goal. This selective matching operation is carried out in the Selective Matching unit (Fig. 2b). By storing structured representations of concepts in LTM, the agent can access past experiences and use them to inform current decisions, allowing adaptation to new tasks and environments.

Matching, however, does not convey sameness retrieval. Creative agents must adapt previous concepts to unfamiliar states by connecting relational concepts to new objects [19]. The hierarchical structure of the concept space allows independent access to each level of abstraction, making this type of adaptation possible. Nonetheless, matching also requires a comparison process to assess similarity by measuring the distance between the elements involved (i.e., graph-based concepts). Traditional techniques for graph comparisons (e.g., contrasting graph adjacency) are insufficient as different edges can bear different importance. To calculate an adequate distance, we propose an Optimal Transport (OT) technique that transforms a (continuous) probability distribution into another with

the lowest possible cost. Therefore, given two graphs, the OT associated with their Wasserstein discrepancy provides a correspondence between their nodes to establish graph matching [1].

After the initialisation of LTM through agent exploration, selective matching on the initialised memory can be done using a trainable policy network for action selection. This shifts the agent's focus towards exploitation, where the goal is achieved by using objects in the environment or building new ones.

Algorithm 2 Enhancing algorithm with selective matching

```
1: selective-matching(G_t, ltm, s_t)- network that returns information useful for the task, uses graph-
   matching, a function that matches a given graph against LTM and returns a list of sub-graphs
   from LTM with a Z% similarity to G_t
2: supplement() -function that enhances a graph G_1 with missing nodes and edges from G_2, uses
   graph-union a function that returns the union of G_1 and G_2
3: for iteration = 1, 2, ..., N do
4:     Init wm ← <keys[], >
5:     while episode not done do
6:         ...
7:         matched-graphs ← selective-matching(G_t, ltm, s_t)
8:         enhanced-graph ← supplement(G_t, matched-graphs)
9:         a_{i,t} ← Run policy-network π(enhanced − graph)
10:        ...
11:    end while
12:    Optimise policy network
13: end for
```

Applying a concept from memory to a given state entails two steps: first, a match of the graph representing the state (e.g., G_t) to the graphs representing the long-term stored information (e.g., *LTM*); second, once the match is successful, G_t is supplemented with the objects and affordances present in the retrieved *LTM subgraph* but lacking in G_t in a process known as completion [19] (Algorithm 2, line: 8). Such completion will foster learning and bring to the agent's current state useful past experiences.

So far, we have shown how an agent could solve a problem using different existing concepts. The following component describes creating a completely new concept by leveraging the existing concept space - considered (everyday) creativity.

3.4 Blending component

Creativity is triggered when existing concepts are insufficient to solve a task, leading an agent to a standstill. We are using the expression *impasse situation* to refer to the inability of an agent to solve a task for several episodes. A CPS approach should aid in overcoming such an impasse with a solution that satisfies problem constraints. Hence, we are working under the assumption that the impasse could be surmounted by developing a novel, useful concept. In the AIGenC framework, new concepts are generated by blending existing concepts in the latent space into new representations. Concept blending [5] denotes the combination of meaningful features of two or more concepts into a new concept.

Two issues must be addressed when creating a new concept: selecting the relevant concepts that help the agent achieve a goal and combining them effectively. To filter the concept space for information that can solve a problem, the agent must have a high-level understanding of the problem, context, and task requirements; that is, acquire a general problem template to process the information, which requires summarising and organising information at a level of abstraction beyond the current hierarchy. Artificial agents lack this knowledge, so we propose a heuristic that widens the range of matched concepts by relaxing the similarity constraint in matching (i.e., we propose using a similarity criterion (say, X%) for selecting concepts to be blended, acknowledging that other approaches may also be appropriate). When Reflective Reasoning finds the concepts that lead to a satisfactory solution, the problem-solving task stops; otherwise, Blending is activated, making the two functionally complementary. Both components match information from LTM to the current state. However, Blending retrieves a larger pool of information, allowing for a wider variety of environmental data to be operated on.

Retrieved concepts are used to create novel concepts using a non-linear trainable network, expanding the concept space with new diverse concepts to be applied to the task. The network must respect the latent space's structure, meaning that the dimensions of the latent vectors should be maintained, and the network should be able to combine their feature values by moving across their dimensions. Relevant features extracted by concept processing sub-modules can be identified post hoc by quantifying their contribution to the performance of the previous unsupervised models. SHAP (Shapley Additive Explanations, [11]), an interpretability method, can be used to weigh the pertinence of individual features and select the latent space features that bear the most relevance to solving the task.

To keep the semantic features of data through the generation process, we propose decomposing the input into an input vector and a latent code that targets the salient semantic features of the data distribution. This process is similar to Chen et al.'s approach [2]. Thus, the input vector would be replaced by the concepts to be merged with a set of trainable parameters for their combination function. In addition, the latent codes could be initialised with an average of the SHAP values. As the Blending function is intended to create novel concepts rather than reproduce input in the latent space, a clustering-specific loss function could be used to evaluate the similarity between the generated concept and existing concepts in the cluster, with a low value indicating a good match. By enforcing clustering similarity through the loss function, we could reduce the variability of the new concept, providing a mechanism for more meaningful concept building.

It is essential to note that the proposed blending mechanism would only be possible in a system that incorporates functional components like those presented in our model. The underlying representational structure of the first component is critical to obtaining a comprehensive input representation. At the same time, Reflective Reasoning tools are essential for selecting, adding, and filtering con-

cepts in the concept space, reducing computational costs and recycling useful concepts. Filtering, which involves understanding the high-level characteristics of the concept affordance, is not simply a search of the conceptual space. Instead, it requires a generalisable solution template, developed through repeated similar experiences. We anticipate that our agent's semi-random selection of relevant objects will establish a foundation for incremental abstract representations. With Blending, the complete architecture has been presented. The architecture proposed serves as a blueprint for designing more robust systems equipped with transferable and adaptable knowledge capable of solving problems creatively.

4 Discussion

We have introduced a deep RL conceptual framework for creative problem solving that enables an agent to represent and generate new concepts and transfer representations across domains and tasks. AIGenC constructs a hierarchical concept space used to retrieve (matching) and adapt (blending) past knowledge. Affordances and temporal information encode the dynamics of the environment, offering an agent multiple views of a concept (related to different possible goals), driving the transfer of learning between tasks and goals.

We are not entering into the debate on how representations are formed, nor do we aim to assess any of the claims regarding concept formation in humans [17]. Our interest lies exclusively in describing the necessary concept space and interplaying structures that could aid an artificial agent in transferring knowledge across contexts and tasks. To that aim, we have adapted ideas from cognitive science to design systems for low-level concept formation and hierarchical concept spaces, but we steer away from assigning anthropomorphic value to the different components and outputs.

Divergent currents of opinion regarding the prospect of AI to achieve human-like intelligence swarm academic fora ([6, 20]). At the eye of the storm, the lack of AI generalisation has been highlighted as one of, if not the most crucial problem to achieve AGI [22]. We concur with cognitive-inspired theories that posit that the lack of generalisation in AI agents is partially due to their inability to discover, combine and generate new concepts that can be used across domains to solve different, yet similar, tasks as humans and animals do [14]. It is conceivable that this operation underpins learning so-called common sense knowledge. Providing artificial agents with a rich concept space to allow the processing of commonalities at different levels of a hierarchy, can serve as a world frame for some level of "common sense". In that vein, our proposal unfolds from contextualised functional representations and sets the appropriate scenario for an AI system to learn an adaptive concept space by interacting with the environment, detect and differentiate commonalities, contextual information and unique functional features that would enable it it to transfer knowledge efficiently.

References

1. Alvarez-Melis, D., Fusi, N.: Geometric dataset distances via optimal transport. Advances in Neural Information Processing Systems **33**, 21428–21439 (2020)
2. Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., Abbeel, P.: Infogan: Interpretable representation learning by information maximizing generative adversarial nets. Advances in neural information processing systems **29** (2016)
3. Coraci, D.: A unified model of ad hoc concepts in conceptual spaces. Minds and Machines **32**(2), 289–309 (2022)
4. Doumas, L.A., Puebla, G., Martin, A.E., Hummel, J.E.: A theory of relation learning and cross-domain generalization. Psychological review (2022)
5. Fauconnier, G., Turner, M.: Conceptual integration networks. Cognitive science **22**(2), 133–187 (1998)
6. Fjelland, R.: Why general artificial intelligence will not be realized. Humanities and Social Sciences Communications **7**(1), 1–9 (2020)
7. Gardenfors, P.: Conceptual spaces: The geometry of thought. MIT press (2004)
8. Gibson, J.J.: The theory of affordances. Hilldale, USA **1**(2), 67–82 (1977)
9. Hassanin, M., Khan, S., Tahtali, M.: Visual affordance and function understanding: A survey. ACM Computing Surveys (CSUR) **54**(3), 1–35 (2021)
10. Jain, A., Szot, A., Lim, J.J.: Generalization to new actions in reinforcement learning. arXiv preprint arXiv:2011.01928 (2020)
11. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. Advances in neural information processing systems **30** (2017)
12. Mitchell, M.: Abstraction and analogy-making in artificial intelligence. Annals of the New York Academy of Sciences **1505**(1), 79–101 (2021)
13. Mondragón, E., Alonso, E., Kokkola, N.: Associative learning should go deep. Trends in Cognitive Sciences **21**(11), 822–825 (2017)
14. Mondragón, E., Murphy, R.A.: Perceptual learning in an appetitive pavlovian procedure: Analysis of the effectiveness of the common element. Behavioural Processes **83**(3), 247–256 (2010)
15. Murphy, R.A., Mondragón, E., Murphy, V.A.: Rule learning by rats. Science **319**(5871), 1849–1851 (2008)
16. Oltețeanu, A.M.: Cognition and the Creative Machine: Cognitive AI for Creative Problem Solving. Springer Nature (2020)
17. Piantadosi, S.T.: The computational origin of representation. Minds and machines **31**, 1–58 (2021)
18. Şahin, E., Cakmak, M., Doğar, M.R., Uğur, E., Üçoluk, G.: To afford or not to afford: A new formalization of affordances toward affordance-based robot control. Adaptive Behavior **15**(4), 447–472 (2007)
19. Shanahan, M., Mitchell, M.: Abstraction for deep reinforcement learning. arXiv preprint arXiv:2202.05839 (2022)
20. Shevlin, H., Halina, M.: Apply rich psychological terms in ai with care. Nature Machine Intelligence **1**(4), 165–167 (2019)
21. Weissenborn, D., Uszkoreit, J., Unterthiner, T., Mahendran, A., Locatello, F., Kipf, T., Heigold, G., Dosovitskiy, A.: Object-centric learning with slot attention (Dec 9 2021), uS Patent App. 16/927,018
22. Zhang, C., Bengio, S., Hardt, M., Recht, B., Vinyals, O.: Understanding deep learning (still) requires rethinking generalization. Communications of the ACM **64**(3), 107–115 (2021)