Cognitive-Level Salience for Explainable Artificial Intelligence

Sterling Somers (sterling@sterlingsomers.com)
Konstantinos Mitsopoulos, Christian Lebiere (cmitsopoulos; cl; @cmu.edu)
Department of Psychology, Carnegie Mellon University,
Pittsburgh, PA 15213 USA

Robert Thomson (robert.thomson@westpoint.edu)
Army Cyber Institute, United States Military Academy
West Point, NY, 10996 USA

Abstract

We present a general-purpose method for determining the salience of features in action decisions of artificial intelligent agents. Our method does not rely on a specific implementation of an AI (e.g. deep-learning, symbolic AI). The method is also amenable to features at different levels of abstraction. We present three implementations of our salience technique: two directed at explainable artificial intelligence (deep reinforcement learning agents), and a third directed at risk assessment.

Keywords: computational model; salience; artificial intelligence; reinforcement learning;

Introduction

In recent years, Deep Reinforcement Learning (RL) has gained popularity for training agents engaged in activities as diverse as playing Atari games, strategic games such as Go and chess, and controlling robotic platforms. Despite recent success, there is, perhaps, a lack of trust in applying RL in real-world scenarios. The behavior of RL systems are often qualitatively different from human behavior and they are inherently difficult to understand. Unlike traditional symbolic AI systems, they are not easy to introspect upon. Furthermore, because RL agents are largely trained without human supervision, there is often little reason to expect them to produce abstractions similar to our own. This makes the task of mapping from human conceptual space to the RL agent's conceptual space a significant challenge.

Previous Work

The common input of RL agents is usually an image. Thus, the agents are comprised of convolutional layers that map pixels to actions and reward expectations. For this reason, most of the techniques that are used for saliency calculations in image classification (Grün, Rupprecht, Navab, & Tombari, 2016) can be used in a RL setting.

One of the first and most common methods for understanding Deep RL agents is to produce gradient-based saliency. Typically, this method uses the gradient of a prediction with respect to an input image to estimate the importance of pixels. In other words, how much the change of a pixel value affects the prediction value (Simonyan, Vedaldi, & Zisserman, 2013). Other popular methods are perturbation methods (Greydanus, Koul, Dodge, & Fern, 2017). Such methods rely on comparing the resulting prediction (or decision) between a modified input with the original one. This gives insights on the importance of individual image regions.

Cognitive Salience

Although the above methods provide a visual explanation of what the agent pays attention to, they often fail to do so consistently. Frequently, the resulting saliences are challenging for a human user to use in producing a meaningful interpretation of the agent's behavior. For this reason, we propose a method that operates on a more abstract level than pixels. More precisely, our cognitive approach involves modeling of the agent's behavior but assuming non-pixel features. Instead the features are entities that a human can comprehend in order to understand the underlying causes of the RL agent's particular decision. Furthermore, the modeling process is model-agnostic and can be used with any model or even humans.

Cognitive Model

We have chosen to develop the cognitive portion of our system in ACT-R. ACT-R is a computational theory of cognition that accounts for the information processes in the human mind (Anderson et al., 2004). The mechanisms in ACT-R are task-invariant and constrained by the limitations of the brain (see Anderson (2007) for an overview). ACT-R is a hybrid architecture, composed of both symbolic and sub-symbolic processing. The hybrid nature of ACT-R is particularly compelling for our work because the symbolic level is inherently explainable, while the sub-symbolic level has the potential representation required to interface with other sub-symbolic systems like neural networks. Furthermore, because we intend to use the output as an explanation for human users, we hope to rely on the constraints of the architecture to limit the output.

Information processing occurs in ACT-R primarily through the interaction of the production system and the declarative memory. Declarative memory is represented as chunks of information. Each chunk has an associated activation level that modulates its retrieval. Chunks are compared to the desired retrieval pattern using a partial matching mechanism that subtracts from the activation of a chunk its degree of mismatch to the desired pattern, additively for each component of the pattern and corresponding chunk value. Finally, noise is added to chunk activations to make retrieval probabilistic, governed by a Boltzmann distribution.

While the most active chunk is usually retrieved, a blending process (Lebiere, 1999) can also be applied that returns a

derived output reflecting the similarity between the values of the content of all chunks, weighted by their retrieval probabilities reflecting their activations and partial-matching scores.

Blending and Salience

The ACT-R blending mechanism retrieves an estimate of values based upon the previous experiences stored in memory, and is computed with the following equation:

$$V = \arg\min_{V_t} \sum_{i=1}^{n} P_i \cdot Sim(V_t, v_{it})^2 \quad (1)$$

The value, V is, therefore an interpolated value based on matching chunks i, weighted by their retrieval probability P_i. $Sim(V_t, v_{it})$ is a similarity function used to compare memory chunks v_{it} and candidate consensus values V_t. In the simplest case where the values are numerical and the similarity function is linear, the process simplifies to a weighted average by the probability of retrieval $V_t = \sum_{i=1}^{n} P_i \cdot v_{it}$.

We consider 'salience' to be the influence a factor has on a decision. The greater degree of influence, the more salient it was when the decision was being made. We model the decisions of an agent by tracing its action decisions and populating a memory. The resulting memory is used in a similar manner to instance-based learning theory (Gonzalez, Lerch, & Lebiere, 2003), except our intent is not to learn to maximize reward, but rather to mimic the behavior of an agent.

We calculate salience by taking the derivative of the blending equation (1) with respect to each feature:

$$S(D_t, f_k) = c \sum_{i=1}^{n} P_i \cdot \left(\frac{\partial Sim(f_k, v_{i,k})}{\partial f_k} - \sum_{j=1}^{n} P_j \cdot \frac{\partial Sim(f_k, v_{j,k})}{\partial f_k}\right) \quad (2)$$

with $P_i = softmax(M_i/\tau)$, $c = \frac{MP}{\tau}$ and $M_i = A_i + \sum_{k=1}^{l} MP \cdot Sim(f_k, v_{i,k})$. This is a novel extension of the blending mechanism that exploits its analytical tractability to provide a closed form of the gradient-based salience of its representational features on its decisions.

Deep Reinforcement Learning Agent

In this work we are not interested in solving completely the problem that the agent is facing. Instead, a basic Deep RL architecture that receives relevantly a high score, in the domains used here, will suit our purposes. For this reason, we utilize the Advantage Actor Critic (A2C) algorithm which is the synchronous version of the A3C (Mnih et al., 2016). We adopt the same architecture and implementation details as in Vinyals et al. (2017) but removed the spatial policy for the drone domain. The agent as it is common in this setting attempts to maximize the expected return by interacting with the domain.

Methodology

The process of mapping between an agent (RL or programmatic) and the cognitive model is common in all the cases described below. Once we have an agent implemented (trained or programmed), we gather data of its performance using terms from a human ontology. In the RL cases, we create a symbolic observation for each step, and record a symbolic interpretation of the action chosen by the network. In the risk-assessment case, we gather symbolic data about the situation, and symbolic data describing the outcome. In each of these cases, this data is gathered to comprise a "memory" for the model. We treat those memories as the knowledge the model has about its respective agent. That knowledge is used by the model (through blending) to estimate what the agent will do in a new (possibly unseen) situation. The derivative of that process, as described, provides the salience. We, thereby, attempt to communicate *why* the agent chooses the action it does by allowing the user to build a mental model of what feature(s) the agent considers most important in different scenarios.

StarCraft II

StarCraft II (SC2) is a real-time strategy game in which players (human or AI) control the production and placement of buildings and the production, movement, and interaction of of militaristic units, in order to defeat opponents. An API and sample RL missions in SC2 are presented in Vinyals et al. (2017). SC2 also supports smaller, constrained missions in which points can be designated for certain achievements. These mini games are useful in the RL domain because they provide smaller tasks and straightforward rewards to be exploited by reward functions.

We used the go-to-beacon mini game presented by Somers, Mitsopoulos, Lebiere, and Thomson (2018). The objective of the go-to-beacon mini game is to move a unit to one of two beacons: a low-value green beacon or a high-value orange beacon. The beacons can be presented either individually or in pairs. When presented in pairs, the optimal solution is to prefer the orange beacon over the green beacon. Interaction in this scenario requires the selection of a unit and then a mouse click on the game map or mini map in the region of the chosen beacon. The unit will then proceed, over time, to move to the location of the mouse click. As soon the agent arrives at one of the beacons, the score is increased by the value associated with that beacon, the beacons disappears, and new beacons are generated at random with four possible configurations: 1) a sole green beacon (green-only scenario), 2) a sole orange beacon (orange-only scenario), 3) both an orange and green beacon presented in such a manner that the unit could take a direct path to the orange beacon without stepping on the green beacon (non-blocking scenario), and 4) both the orange and green beacons presented in such a manner that the green beacon is overlapping the direct path between the unit and the orange beacon (blocking scenario).

The scenarios are grouped into the four categories to accommodate a human-level ontology that might help understand the RL's behavior. Despite the simplicity of the mission, our RL agent learned a sub-optimal policy. In particular, the RL fails to guide the SC2 unit around the green beacon to the orange beacon, failing to distinguish the blocking and non-

blocking scenarios. In the context of this work, we did not attempt to resolve this issue. Our aim in this paper is to explain why the RL agent acts the way it does, and to cast these explanations using a human-level ontology.

SC2 Explanation The RL agent did not learn how to go around the green beacon to reach the orange beacon. Our approach aims to explain the behavior of the RL agent in terms of its internal states: what it perceives, what it knows, and what actions it takes in response. We assume that actions consistent with a going-around action require a spatial inference: that the green beacon is 'between' the agent and the goal. In the SC2 case, we aim to communicate to a user that the network has not acquired this concept and therefore fails to act as expected.

In this particular task we are curious whether the agent has an internal representation that is functionally consistent with: a) a sole green beacon, b) a sole orange beacon, c) a non-blocking scenario, and d) a blocking scenario. We collected data of the RL agent by tracing its behavior while it played the game, collecting chunk representations that described the scenario and the action chosen in that scenario. The chunks had the following structure: *green*:value, *orange*:value, *blocking*:value, *internal representation*:vector, *select-green*:value, *select-orange*:value, *select-around*:value. Each value, aside from the vector value, are binary 0 or 1. The *green, orange, blocking* together describe the scenarios (a-d) and *select-green, select-orange, select-around* are high-level descriptions of possible actions taken. Note that these are representations of ground-truth, not representations created by the agent. We are attempting to assess when (and if) the RL agent is behaving in a manner consistent with these representations. We adopted the approach used by Somers et al. (2018) and used a vector representation to capture the internal state of the agent. The internal representation is used in the partial-match portion of the blending process. Just as in Somers et al. (2018), we use cosine as a similarity measure between two vectors. Overall, we filter the data we collected to include at least one example of each scenario and to maximize the distance between vectors. For the purposes of the present description, experience was filtered down to 20 examples.

We made a prototype of a display that could be used for explanation. It outputs the results of the blending process and salience calculation graphically, relying on the user to create the proper inferences. At each step in the game, the cognitive model makes a blend estimate of the action to be chosen by the agent, and calculates the salience of the high-level features: green, orange, blocking. A screen-capture of this display is presented in Figure 1. A screen capture of the corresponding SC2 scenario is presented in Figure 2.

The top graph in Figure 1 displays the three possible action choices a human might expect the RL agent to make and the cognitive architecture's estimate of which action the RL agent will choose. The example in the scenario is a blocking scenario and, as the display correctly indicates, the RL agent will choose to select the orange beacon.

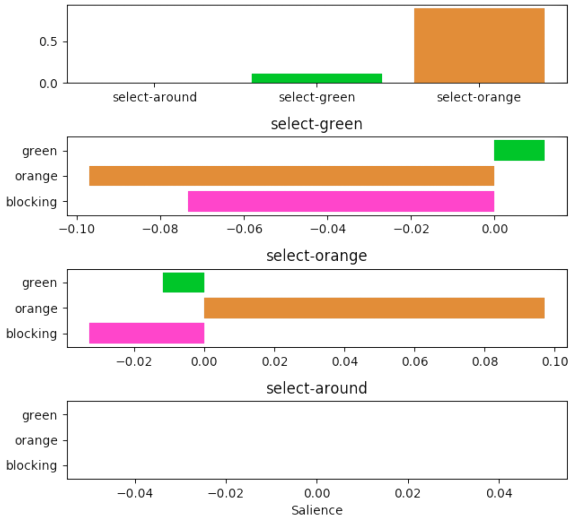

Figure 1: Explanation display. Top panel illustrates the blend value for the action decision. The three remaining panels display the associated salience for their respective decisions.

Figure 2: Screen capture of blocking scenario in StarCraft II. In this image, the marine (controlled by the reinforcement learner) is attempting to get to orange beacon.

The three bottom graphs in Figure 1 display the salience for each decision available. The top displays the salience for the select-green decision, the middle displays the salience for the select-orange decisions, and the bottom graph displays the salience for the select-around decision. As described above, the salience indicates the degree of influence each of the features (green, orange, block) have on the action decision. Given that the dominant action chosen by the drone is select-orange in the blocking scenario, discussion will focus on describing the bottom-middle graph, "select-orange". The graph indicates that the presence of the green beacon has a small negative salience in the action decision and the abstract concept "blocking" has a larger negative salience, with the dominant influence being the presence of the orange beacon. This makes sense given that the RL agent always tries to get to the highest reward but appears to be unaware of the fact that the green beacon is blocking the orange. The salience and the associated lack of 'go-around' action communicate that the agent has not formed the abstract concept, 'blocking'.

Drone Domain

A second domain that we have applied the cognitive salience technique to is a drone operations domain. Currently, our drone environment is a 3D gridworld abstraction of MAVSim (Youngblood, Kravacic, & Le, 2018). Once trained, the RL agent can be deployed in MAVSim.

The aircraft we are simulating are fixed-wing drones. The missions generally include search and provisioning lost hiker(s) with any combination of food, water, first-aid, and communication devices. The rules of the environment are an abstraction that capture the constraints of flight dynamics. Although, at our current level of simulation, the specifics of the aircraft are not captured, the rules of the environment are sensitive to flight constraints more generally. The rules of the environment constrain the turning radius, maintains forward motion, and restricts elevation changes, just to name a few. Sensors on the aircraft are sensitive to altitude. Package survival (once dropped) depends on underlying terrain and will fall differently when dropped at different altitudes.

We have trained an RL agent to navigate to a hiker visible on a topological map and to drop package(s) near the hiker. Currently, programmatic solutions are used to carry out other segments of the flight including: loading the packages, searching for the hiker, and returning to the airport. Our explainable AI challenges in this domain are many and we have only begun to touch the explainability potential in the drone domain. We present here two example prototype uses of our salience technique within the drone domain: risk assessment and egocentric salience.

Risk Assessment As we move from low to high levels of fidelity, there is an increasing number of moving parts to deal with the increasing level of detail and, as a result, a large potential decision space to explain. Anticipating our needs, we have begun to prototype different explanation interfaces for different aspects of a mission. In this simplified example, we imagine the provision-loading process, and when it might need explanation. The most obvious situation where we might want some form of explanation is when there is a failure. After a large number of simulations, for example, we may want some form of risk analysis that is sensitive to the particular constraints of any given example case. We use the following simplified example, where the ground truth about a package loading module is described. In this example, the operator is unaware of the ground truth and is attempting to diagnose a problem with the package loading module.

The package loading module is given as input the needs of the hiker(s). This module has a number of flaws that we know a priori for this example but are not known by the user: 1) it always loads communications equipment, regardless of needs. 2) It always loads food, regardless of needs. 3) It never loads water, regardless of needs. We characterize success and failure in the following manner: A) if the hiker needs a provision and does not get it, there is a failure. B) If the hiker does not need a provision, and a provision is loaded, this is also considered a failure. We chose to consider this a failure because the aircraft we are modeling has limited space for provisions. Loading unnecessary provisions can prevent loading required provisions and lead to increased fuel consumption, which could lead to the drone making multiple, unnecessary trips, putting the hiker(s) in unnecessary risk. To remove any confusion, we also include a third success/failure condition: C) even though we know, a priori, what is wrong with the drone, we remember that, for the purposes of the example, when the hiker does not need a provision (in this example, water) and that provision is not loaded (because there is a flaw preventing it), we consider that a success even though, underlying that success, is a failure in the mechanism. In other words, the salience mechanism in this case only has access to the behavior of the drone, without any knowledge of its internals. These are the rules that describe the success/failure conditions in this example.

Following those rules we generated data to simulate the erroneous module described. The data includes all sixteen possible binary combinations of package-loading requirements: food needed/not needed, water needed/not needed, first-aid needed/not needed, communications needed/not needed; as well as traces of their success and failure: food-success yes/no, water-success yes/no, first-aid-success yes/no, communications-success yes/no. This is represented in ACT-R as chunks with 8 slot/value pairs.

Once we have data, we can probe the module with a new case to perform a risk assessment in a specific situation. For example: *Food: needed; Water: not needed; First-Aid: needed; Communications: not needed*

There are two aspects to the output. First the blend provides an estimate of the values (randomly generated example presented in Table 1). The values are intuitively what we might expect (given the module described above): a value of 1 for both radio and food (which is always loaded by the erroneous module). Water is estimated to be zero (which deceptively makes sense, since it was not requested). Finally, First-Aid is estimated to be 0.68, which could be rounded to 1 (given our binary example). These results are consistent with the rules described above.

Table 1: Blend Estimate of Random Example

Provision	Requested	Estimate
Food	1	1
Water	0	0
First-Aid	1	0.68
Communications	1	1

The salience provides further, useful information that could potentially be used to diagnose a faulty module. Since the blend is produced for each output factor (food, water, first-aid, communications), we generate a set of saliences for each. The salience derivative is computed with respect to each feature, so each factor has 4 saliences values associated with it. The salience values for the Food and Communications are

identical but their values are so small (10^{-8}) that we do not display them. The small values are important in the explanation, however, because they indicate the blend is not strongly influenced. This makes sense, given that we know that the agent always loads food and always loads communications.

The salience of Water is zero for all features and, therefore, not displayed. This is particularly telling, a very strong suggestion that the loading of water is not sensitive to any factor. This makes sense given the ground truth about the module (always fails to load water).

Finally, the salience of First-Aid is presented in Figure 3. This display is telling, indicating salience values near zero for Water, Food, and Communications (Radio).

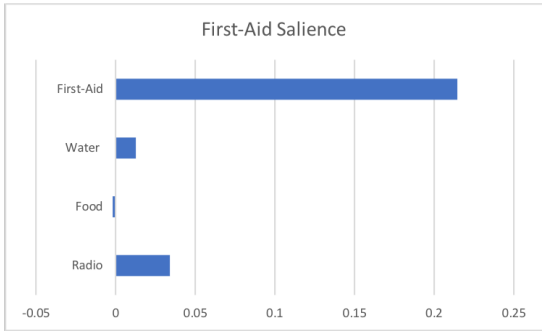

Figure 3: Display of salience for assessment of First Aid provisions.

Egocentric Salience While the risk assessment is derived from synthetic data generated by a programmatic agent, our final example, which we term, 'egocentric salience', results from a model trace of an RL drone agent. The input to the RL is an egocentric image and an allocentric image. Example inputs are illustrated in Figure 4 and the bottom display of Figure 5. The first image is a 20 unit by 20 unit allocentric map of our environment. The different green and brown colors represent different kinds of vegetation. The purple arrow, pointing downwards, represents the drone; with the direction of the arrow indicating the drone's heading. The drone's color indicates its altitude. The red cross indicates the location of the hiker for which the drone is attempting to drop off a package.

The second image (bottom of Figure 5) is a 5 unit by 5 unit egocentric view that corresponds to the first image. This image is a vertical slice of the first, in the region around the drone's action space. At any step the drone can head directly forward, diagonally forward, or turn 90 degrees in either direction. Furthermore, any of those directions can be combined with a single change of altitude. With the options of left, diagonal left, center, diagonal right, and right, multiplied by: no altitude change, an increase in altitude, or a decrease in altitude; the drone has an action space of 15 possible grid square in a 3D gridworld. A sixteenth action to drop a package is also available. The movement actions are entirely captured by the five by five units of the egocentric input image.

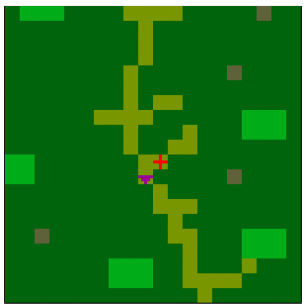

Figure 4: 20x20 unit allocentric view. Greens and browns indicate different forms of vegetation (largely trees and grass). The purple triangle indicates the drone (facing the bottom of the image). The red cross indicates the location of the hiker.

The drone is super-imposed on this picture and provides the drone an explicit representation of its altitude. The column

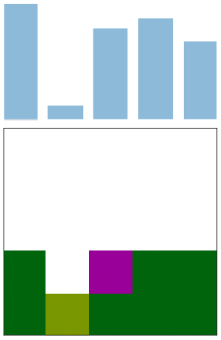

Figure 5: Top frame: the bars above the egocentric view indicate the salience of the columns in the egocentric view. Bottom frame: 5x5 unit, egocentric input to the network. This image corresponds to the allocentric image and changes each step of the simulation.

on the left-most of the egocentric view corresponds with the grid-square directly to the right of the drone (from the perspective of Figure 4) but corresponds to the square immediately to the left of the drone (from the perspective of the drone). This is the case because the drone is actually facing downward in the allocentric view. The column second from the left in the egocentric view corresponds to the patch of grass to the diagonal left of the drone (down and to the right in the allocentric image). The three remaining columns represent obstacles (trees) surrounding the drone centre and all the way to the right. If the drone were to fly left, forward, diagonal right, or right, it would crash into a tree.

The drone exhibits a reasonable capacity to carry out the two segments of the mission it has been trained on (traversing the map and dropping the package). However, despite its success, the drone does exhibit unusual behavior: sometimes taking a bizarre path to the hiker, or circling the hiker many times before dropping a package. This successful yet

unusual behavior is a good candidate for explanation because a participant viewing the mission might want to resolve why the behavior is markedly different from what a human might do. Explaining this type of behavior can furnish trust in the system if the human understands (and accepts) the reasoning.

Figure 5 depicts a prototype explanation for the scenarios presented in Figure 4. Unlike the other examples, the concepts we are associating salience with in this example are primarily spatial. The idea behind egocentric salience, is still, however, quite abstract. The data is collected under the assumption that the RL drone responds to features at altitude, trying to avoid crashing, for example, regardless of the specific identity of the object. The aim of the explanation is towards understanding how the drone responds more generally to its environment. It would be used, for example, to allow a user to rewind to a point where they thought the drone started to behave oddly, and get a sense of what it was 'attending' to considering in its decision.

The egocentric salience is a little more difficult to interpret. The bars above the egocentric view (Figure 5) are meant to communicate how salient the columns of that view are toward a single decision. This particular example could be interpreted as all dark green objects (obstacles), at altitude 1 (counting from the bottom, from zero), are highly salient, with the single safe column, with a low degree of salience. The low-degree of salience was an unexpected result but, with some interpretation, seems to make sense. The areas where objects are lower in altitude than the drone's current altitude generally will not influence the drone's behavior. Instead of thinking about the drone as going to a safe area, per se, the drone seems to be avoiding dangerous areas (the result of which is functionally the same, going to a safe area).

As with our other examples the blending process also makes an action estimate. Because the action space is so large (15 movement actions), we do not blend for each value. Instead we blend a single action value across the range. In this case, the estimate is 6.8. Rounded to 7 an action of forward and to the left, which would result in action following the path in the forest.

Discussion

The work presented here is targeted at explainable AI and is still in an early phase of development. Our goal builds upon (Kümmerer, Wallis, & Bethge, 2015) to eventually unify pixel-level, cognitive-level, and artificial intelligence-level salience computation in a model-agnostic framework. Although we have thus far concentrated on explaining AI, we are interested in exploring salience for human-generated data, specifically in the context of instance-based learning (IBL) as IBL models have been used in a wide variety of models including social dilemmas (Gonzalez, Ben-Asher, Martin, & Dutt, 2015) and two-person games (West & Lebiere, 2001).

References

Anderson, J. R. (2007). *How Can The Human Mind Occur In The Physical Universe?* New York, NY: Oxford University Press.

Anderson, J. R., Bothell, D. J., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004, oct). An integrated theory of the mind. *Psychological review*, *111*(4), 1036–60. doi: 10.1037/0033-295X.111.4.1036

Gonzalez, C., Ben-Asher, N., Martin, J. M., & Dutt, V. (2015). A cognitive model of dynamic cooperation with varied interdependency information. *Cognitive science*, *39*(3), 457–495.

Gonzalez, C., Lerch, J. F., & Lebiere, C. (2003). Instance-based learning in dynamic decision making. *Cognitive Science*, *27*(4), 591–635. doi: 10.1016/S0364-0213(03)00031-4

Greydanus, S., Koul, A., Dodge, J., & Fern, A. (2017). Visualizing and understanding atari agents. *arXiv preprint arXiv:1711.00138*.

Grün, F., Rupprecht, C., Navab, N., & Tombari, F. (2016). A taxonomy and library for visualizing learned features in convolutional neural networks. *arXiv preprint arXiv:1606.07757*.

Kümmerer, M., Wallis, T. S. A., & Bethge, M. (2015). Information-theoretic model comparison unifies saliency metrics. *Proceedings of the National Academy of Sciences*, *112*(52), 16054–16059. Retrieved from https://www.pnas.org/content/112/52/16054 doi: 10.1073/pnas.1510393112

Lebiere, C. (1999). The dynamics of cognition: An ACT-R model of cognitive arithmetic. *Kognitionswissenschaft*, *8*(1), 5–19. doi: 10.1007/s001970050071

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., ... Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *International conference on machine learning* (pp. 1928–1937).

Simonyan, K., Vedaldi, A., & Zisserman, A. (2013). Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*.

Somers, S., Mitsopoulos, C., Lebiere, C., & Thomson, R. (2018). Explaining decisions of a deep reinforcement learner with a cognitive architecture. In *Proceedings of the sixteenth annual conference on cognitive modeling* (pp. 144–149). Madison, WI: Lawrence Erlbaum Associates.

Vinyals, O., Ewalds, T., Bartunov, S., Georgiev, P., Vezhnevets, A. S., Yeo, M., ... others (2017). Starcraft ii: a new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*.

West, R. L., & Lebiere, C. (2001). Simple games as dynamic, coupled systems: Randomness and other emergent properties. *Journal of Cognitive Systems Research*, *1*(4), 221-239.

Youngblood, G. M., Kravacic, B., & Le, J. (2018). *Mavsim.* https://gitlab.com/COGLEProject/mavsim. GitLab.