ACS Poster Collection (2018) 289–302 Published 12/2018; Paper 18

Learning to Recognize A-Life Behaviours

Everton Schumacker Soares SCHUMACK@UALBERTA.CA
Vadim Bulitko BULITKO@UALBERTA.CA
Kacy Doucet KJDOUCET@UALBERTA.CA
Department of Computing Science, University of Alberta, Edmonton, Alberta, T6G 2E8, CANADA

Morgan Cselinacz CSELINAC@UALBERTA.CA
Department of Psychology, University of Alberta, Edmonton, Alberta, T6G 2E9, CANADA

Terence Soule TSOULE@UIDAHO.EDU
Samantha Heck HECK9873@VANDALS.UIDAHO.EDU
Landon Wright WRIG8396@VANDALS.UIDAHO.EDU
Polymorphic Games, University of Idaho, Moscow, Idaho 83844, USA

Abstract

The ability to recognize and reason about cognitive behaviours of intelligent agents is a fundamental cognitive ability of humans. In this paper we take a step towards using machine learning to acquire a simple form of such an ability. Artificial life (A-life) is a framework to study the emergence of novel multi-agent behaviours during simulated evolution. As evolution runs can sparsely result in such novel behaviour, we propose the use of machine-learned methods to detect behaviours automatically. While traditional machine-learning methods depend on predefined features describing the input, deep learning can automatically organize low-level inputs into higher-level features. We show that an off-the-shelf deep neural network can be trained to robustly recognize a novel leader-follower behaviour in a predator-prey A-life model. In doing so, this network substantially outperforms density-based and clustering-based detectors. We then demonstrate portability of the approach by training the same network on images from a commercial evolution-based video game. The network is able to robustly distinguish between two modes of gameplay.

1. Introduction

Advances in the field of Artificial Intelligence (AI) have transformed the way humans perceive artificial agents, raising the question whether machines can in fact achieve human-like cognitive levels, and at which point machine behaviors go beyond expertise in a particular narrow task to proficiency in broad and complex cognitive tasks. Recent high-profile advances in AI have come from the use of machine learning and, in particular, from the reduction of human knowledge given to the system (Silver et al., 2016, 2017). We conjecture that even further advances in cognitive abilities of machine intelligence can come from evolving such intelligence in Artificial Life (A-life) environments. Early work by Ackley & Littman (1991) demonstrated a potential of even simple evolved agents showing cognitively interesting interactions. Unfortunately, many runs of such a simulated

evolution within such an A-life environment may not result in emergence of interesting cognitive behaviours. The ability to recognize and reason about cognitive behaviours is a fundamental cognitive ability of humans. However, it would be intractable for human observers to sift through numerous evolution runs in search of an interesting one. Thus, it becomes important to complement an evolution within an A-life environment with automatic detection of cognitively interesting behaviours. Ideally such a detector should work with data readily observable in an A-life environment. In this paper we demonstrate how an off-the-shelf deep neural network can be trained to detect emergent behaviour from a simple pixel-level visualization of a predator-prey A-life environment. In doing so, the network demonstrates a cognitive ability to recognize a behaviour of other (simple) cognitive agents. We hope that future work will increase the cognitive capabilities of the underlying A-life agents as well as the cognitive ability of the detector accordingly.

Another application of behaviour detection is in video games. To reduce development costs, game companies have turned to procedural content generation methods such as simulated evolution for level design, animation, and behaviours for non-playable characters (NPCs) (Yannakakis & Togelius, 2018). Such methods can lead to emergence of behaviours unexpected by game developers. For instance, the video game *Darwin's Demons* (Soule et al., 2017) used player-influenced evolution to evolve space-invader-like NPCs. An unexpected phenomenon that emerged was the separation of the evolved NPCs into two groups; while the player was attacking one group on one side of the screen, the group on the other side would rush to the bottom of the screen thereby winning the round. In doing so, NPCs in the first group would sacrifice themselves to distract the player. While interesting from an AI perspective, such unexpected behaviour may clash with the game design, discouraging use of simulated evolution among commercial game developers. We propose automatic behaviour detectors as a quality assurance tool. Unusual emergent behaviour flagged by machine-learned classifiers and anomaly detectors can then be inspected by game developers. Such detection can take place during the development cycle and/or with telemetry data from the player base.

In our approach, we take a common and readily available off-the-shelf deep neural network, AlexNet (Krizhevsky et al., 2012), and train it in a supervised fashion to distinguish between two non-trivial multi-agent behaviours that emerge in our predator-prey A-life model. We intentionally remove information from the images fed to AlexNet, reducing the content to only spatial locations, shapes, and orientations; yet AlexNet is able to robustly classify behaviours from a single simplified screenshot. We compare its performance to two other non-deep-learning classifiers and demonstrate that the information-sparse inputs present more of a challenge to them. Finally, we apply this approach to a commercial video game, *Project Hastur*, and show that AlexNet is able to robustly classify behaviours of evolved NPCs from simplified screenshots as well. This contribution supports the portability of our approach and indicates a potential for using deep-learning for video game monitoring and automated quality assurance.

2. Problem Formulation

The problem we consider in this paper is multi-agent behaviour detection in an A-life-like environment. The solution for this problem needs to work well with readily available input, such as

screenshots captured from an A-life simulation or a video game.* The machine learning process should be data and time efficient (i.e., affordable to individual researchers or game developers). We frame the problem as a binary classification of information-sparse images taken from an A-life simulation and a commercial video game. Our performance measure is the classification accuracy on previously unseen data.

The task is challenging for several reasons. First, the input to such a classifier is low-level (e.g., pixels on the screen) and lacks higher-level semantic features. For instance, the classifier does not know *a priori* which part of the image represents an agent and which is the environment. Second, a novel behaviour can arise from interaction between otherwise uninteresting agents. The classifier has no *a priori* notion of agent interaction. Third, agent behaviours unfold over multiple time steps but the classifier has to make its decision on the basis of a single image.

The problem, as framed above, involves several types of cognition. First, the agents themselves can be arbitrarily complex. For instance, non-playable characters in video games are often controlled by large human-crafted behaviour trees and involve reasoning about resources, threats, other agents, changes over time, memory of interactions with the player, etc. Second, the inference employed by the behaviour detector can involve a hierarchy of semantic information. Indeed, deep neural networks such as the ones we use in this paper are believed to learn to extract and reason over higher-level semantic features from low-level input data.

2.1 A-life Environment

To evaluate our approach we needed an A-life environment where artificial evolution will sometimes lead to emergence of interesting behaviours. We extended our previous A-life environment (Bulitko et al., 2017), a simple artificial evolution of prey (rabbits) and predators (wolves) developed from a published Netlogo model (Wilensky, 1997).

In the model, a population of rabbits and wolves co-evolve over time. Visualized in Figure 1, rabbits (orange and purple triangles) move around a rectangular grid and eat grass (green patches; the intensity of green indicates the amount of grass in the grid cell). Consumed grass regrows. Wolves (blue triangles) move around and eat rabbits. On each time tick, each agent (i) perceives all other agents and grass levels within a genetically-encoded radius, (ii) computes scalar utilities of all grid cells within the radius and (iii) moves a certain distance toward the cell with the highest utility (or stays put if the agent is already there). Each agent has an energy level which is increased by eating and is decreased by perceiving (proportional to number of grid cells perceived), moving (faster-moving agents burn more energy), and simply existing. If energy drops below a certain threshold, the agent dies. Agents do not make any decisions except where to move next.

The evolution is asynchronous and does not have either discrete generations or an explicit fitness function. Instead, each agent above a minimum reproduction age with sufficient energy compulsively has a single offspring who receives half of the parent's energy at birth. The parent continues to live and may give birth to other agents in the future. We do not enforce a minimum or maximum number of agents through agent injection or culling. The simulation is stopped when all wolves or all rabbits die, or a certain number of time ticks is reached.

*Note that while game developers also have access to their agents' AI code, the players do not. Thus attempting to detect novel behaviour by detecting novel code has the danger of detecting something that the player does not see.

Each agent has seven genes: sight radius r, affinity to grass α_{grass}, affinity to rabbits α_{rabbit} and affinity to wolves α_{wolf}. It also has three second-order affinities: affinity to other agents' affinity to grass $\alpha_{\alpha_{\text{grass}}}$, affinity to other agents' affinity to rabbits $\alpha_{\alpha_{\text{rabbit}}}$ and affinity to other agents' affinity to wolves $\alpha_{\alpha_{\text{wolf}}}$. The agent with sight radius r perceives all grid cells whose coordinates are at most r cells away from the agent $N = \{(x, y) \mid |x - x_{\text{agent}}| \leq r \;\; \& \;\; |y - y_{\text{agent}}| \leq r\}$. For each cell n in the neighborhood N the agent computes the cell's utility using its first-order affinities as $U(n) = \alpha_{\text{grass}} \cdot \sharp_{\text{grass}}(n) + \alpha_{\text{wolf}} \cdot \sharp_{\text{wolf}}(n) + \alpha_{\text{rabbit}} \cdot \rho(n)$ where $\sharp_{\text{grass}}(n)$ is the amount of grass in cell n, $\sharp_{\text{wolf}}(n)$ is the number of wolves in cell n and $\rho(n)$ is the weighted attraction/repulsion of all rabbits in cell n, computed as $\rho(n) = \sum_{x \in n} \left(\alpha_{\alpha_{\text{grass}}} \cdot \alpha^x_{\text{grass}} + \alpha_{\alpha_{\text{wolf}}} \cdot \alpha^x_{\text{wolf}} + \alpha_{\alpha_{\text{rabbit}}} \cdot \alpha^x_{\text{rabbit}}\right)$ which evaluates each rabbit x present in the cell n with respect to x's affinity to grass (denoted by α^x_{grass}), x's affinity to wolves (α^x_{wolf}) and x's affinity to rabbits (α^x_{rabbit}).

To illustrate, a rabbit with the genes $r = 1, \alpha_{\text{grass}} = 1, \alpha_{\text{wolf}} = -1, \alpha_{\text{rabbit}} = 1, \alpha_{\alpha_{\text{grass}}} = \alpha_{\alpha_{\text{rabbit}}} = \alpha_{\alpha_{\text{wolf}}} = 0$ sees its own cell and its immediate 8 neighbors, likes grass, dislikes wolves and is indifferent to other rabbits. On the other hand, a rabbit with the genes $r = 1, \alpha_{\text{grass}} = -1, \alpha_{\text{wolf}} = -1, \alpha_{\text{rabbit}} = 1, \alpha_{\alpha_{\text{grass}}} = 0, \alpha_{\alpha_{\text{rabbit}}} = 1, \alpha_{\alpha_{\text{wolf}}} = 0$ dislikes grass but likes rabbits who like grass.

In Figure 1 rabbits with $\alpha_{\text{grass}} > 0$ are coloured orange and rabbits with $\alpha_{\text{grass}} < 0$ are coloured purple. The magnitude of α_{grass} determines the intensity of the color.

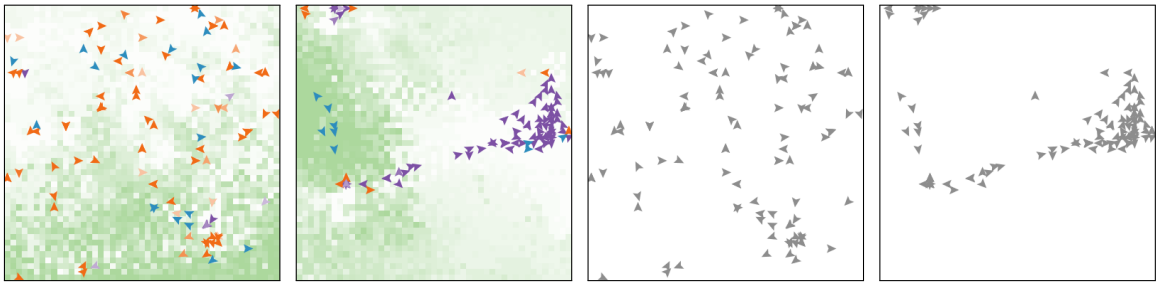

Figure 1: Representative images for the dataset extracted from our A-life simulation. The first image on the left represents a frame with *more leaders* (orange triangles), the second with *more followers* (purple triangles); wolves are shown as blue triangles, and the shade of green indicates the amount of grass. The two images on the right are simplified images given to our deep artificial neural network.

2.2 Interesting and Non-interesting Behaviours

The behaviour that we are detecting pertains to leader and follower interactions. Strandburg-Peshkin et al. (2018) define animal leadership in terms of the influence that a group member has over other group members. The A-life environment described above occasionally yields emergence of leader and follower behaviour among rabbits. While initially all genes are random, two types of rabbits will sometimes emerge during evolution. First are rabbits who, as expected, like grass: $\alpha_{\text{grass}} > 0$. The second type are rabbits who surprisingly dislike grass: $\alpha_{\text{grass}} < 0$. On their own, the latter would die of starvation, preferring to stay put in their own cell where the grass has been eaten. Remarkably, they can survive for a long time in the presence of the grass-liking rabbits *if* they also

happen to like such rabbits: $\alpha_{\text{rabbit}} > 0 \quad \& \quad \alpha_{\alpha_{\text{rabbit}}} > 0$. When their attraction to grass-liking rabbits overcomes their repulsion to grass, grass-disliking rabbits follow them into grass-rich cells, compulsively eating and eventually reproducing. Thus, we call the grass-liking rabbits *leaders* and the grass-disliking rabbits *followers*.[†]

With this phenomenon in mind, we define two types of behaviours: *non-interesting*, where most of the rabbits in the environment are regular grass-liking rabbits, and *interesting*, where most of the rabbits are grass-disliking. By "most," we mean at least twice as many. For instance, the first coloured screenshot in Figure 1 has at least twice as many leaders as followers and thus constitutes an example of non-interesting behaviour. The second coloured screenshot has at least twice as many followers as leaders and thus constitutes an example of interesting behaviour.

Note that in this paper we do not distinguish between properties of the environment (e.g., that majority of rabbits are grass-disliking) and behaviours of the agents within the environment (e.g., that grass-disliking rabbits survive by following grass-liking rabbits). Indeed, in a fixed environment, emergence of a new agent behaviour is likely to change the environment's properties. Future work will attempt to decouple behaviours and the resulting properties of the environment by attempting to detect a behaviour (e.g., grass-disliking rabbits following grass-liking rabbits) in a variety of substantially different environments. Since behaviours unfold over time, we expect to have to upgrade our behaviour detectors with memory (e.g., by having them inspect a sequence of environmental states).

3. Related Work

Our main approach is adapted from our previous work in which AlexNet classified single image frames from A-life simulations with two different agent mutation rates (Bulitko et al., 2017). We extend that research in two ways: first, our previous work only classified the mutation rate that was used in evolving the population, which does not necessarily provide information about the agents' behaviour. Second, we use pixel-balanced images so that AlexNet can not merely rely on aggregate statistics such as the number of agents or pixels in the input image.

The problem of automatically detecting different types of group behaviour in a population is also of interest in biology. For example, Pu et al. (2018) identified flocking behaviour in chickens using a convolutional neural network with three convolutional layers and a fully connected layer that processes single colour and depth frames generated with a Kinect sensor. This model was specifically created for confined poultry flocking behaviour and requires depth sensors. Although their proposed method achieved good results, their approach depends on depth frames, which may not be readily available in A-life simulations or video games. Machine learning methods for image-based identification of animal behaviour have also been applied to ecology research and laboratory environments. For example, Berman et al. (2014) proposed a machine-learning approach to map behaviours of flies according to motion patterns in images. They used principal component analysis to decompose a preprocessed image sequence of flies in order to generate spectrograms later

[†]Note that grass-disliking rabbits may or may not like following other rabbits. However, if they happen not to like other rabbits then they will not follow them and thus will likely quickly die of starvation and not reproduce. Thus we abuse the terminology and call any rabbit who likes grass a leader and any rabbit who dislikes grass a follower regardless of whether leading or following actually takes place at a given time.

mapped through the T-distributed Stochastic Neighbor Embedding algorithm. Their method is for single-agent behaviours whereas we are interested in emergent group behaviours. Dell et al. (2014) proposed the use of supervised and unsupervised learning to automatically detect and track specific behaviours in real-life animals. In line with their proposal, the approach presented in this paper trains a deep neural network to detect behaviours in simple simulated predator-prey organisms.

4. Our Approach

Our previous work on behaviour detection (Bulitko et al., 2017) had similar desiderata so in this paper we build on our previous approach of training common off-the-shelf deep neural networks. Their wide availability and low hardware/software requirements make the approach affordable to individual researchers or game developers. One of the strengths of deep learning lies with its ability to form a feature representation automatically, thereby removing the need for human researchers to design a domain specific representation. In particular, common deep neural networks such as AlexNet were designed to run on standard colour images which are easy to collect in an A-life environment or a video game.

Our deep neural network classifier is trained using stochastic gradient descent with static learning rate on the training set. We took a common off-the-shelf deep network, AlexNet, and modified it to have 2 outputs (*more leaders* and *more followers*) instead of the usual 1000.[‡] We determined the training hyperparameters — learning rate, batch size, and number of epochs — by first conducting a sweep of the parameter space using all combinations of learning rate in $\{10^{-5}, 10^{-6}\}$, batch size in $\{5, 10, 25\}$ and number of epochs in $\{10, 25, 50, 75\}$. For each combination of hyperparameters, we split the data into training and testing 4 times, fine-tunning an independent AlexNet on each one of the 4 splits. The best test accuracy averaged over the 4 trials was achieved with the batch size of 25, 75 epochs, and the learning rate of 10^{-6}. We fixed the hyperparameters at those values and used them to train AlexNet on 30 trials, as explained in details in Section 5.3.

5. Experimental Evaluation

Our hypothesis is that a deep neural network can be trained to reliably recognize interesting and non-interesting group behaviours from a single image captured in an A-life environment. To evaluate this hypothesis we will implement the approach and compare the resulting performance to classification methods based on agent density and clustering.

5.1 Data Collection

An evolution run starts with random genes in the population so in order for any rich multi-agent behaviours to emerge, the evolution needs to run long enough. Since we did not have agent culling or injection, we needed to have world parameters, or *physics* (e.g., the rate of grass re-growth, the initial population of wolves, the rabbit movement speed, etc.) that support long-running evolutions. Our environment was developed as a Netlogo model, so we used the built-in Behaviour Search

[‡]In all of our experiments we used AlexNet pre-trained on an ImageNet database as available for download with MATLAB's neural-network toolbox (R2018a).

module to find physics that maximize population extinction time. This was done by running a meta evolution on the physics for approximately one day, using median extinction time as a measure of fitness to produce a set of physics that reliably yielded long-living populations.

Using these physics, we ran 1000 agent evolutions. Each frame of each evolution was analyzed for having at least twice as many leaders or at least twice as many followers (or neither). In the former two cases, the screenshot was then saved into the more leaders or more followers set. Other screenshots were discarded. All rabbits and wolves in the saved images were displayed as triangles of the same size and shade of gray, and the grass visualization was disabled (two right panels in Figure 1). This process recorded 8732517 images in the more leaders class and 47123 images in the more followers class.

5.2 Data Pre-processing

There are two problems with the data set collected above. First, it is notably unbalanced with many more images in more leaders class. Second, there may be a systematic bias in the number of agent pixels which would make classification easier. We addressed the first problem by randomly sampling 47123 images out of 8732517 images in the more leaders class. We addressed the second problem by balancing the remaining images on the colour of the agent (gray) pixels. The pixel-colour balancing process ensured that each more leaders image with n gray pixels has a unique corresponding image in the more followers class with exactly n gray pixels. This resulted in 13349 images in each of the two classes that were identically distributed in terms of the number of agent (gray) pixels.

5.3 Performing Multiple Trials

We then ran 30 trials with the data set. Each trial split the images into a training set and test set. Because images from the same evolution run are likely to be correlated, we partitioned the set at the level of evolution runs. Images from approximately 75% of the evolution runs in each of the two classes were put into the training set. Images from the remaining runs were put into the test set. As different evolution runs vary in duration, they contribute varying numbers of images to each class which may create unequal representations of the two classes. On each trial we used all data in the trial's training set to train three classifiers: density-based classifier, cluster-based classifier, and deep neural network classifier.

5.4 Clustering Classifiers

Since follower rabbits tend to form clusters around the leaders, as shown in Figure 1, we compared our deep neural network to two classifiers that rely solely on clustering measures to distinguish between leader-centric and follower-centric images. Such measures are commonly used in biology and A-life research. For example, Sanvicente-Añorve et al. (2017) used the Morisita index of aggregation to classify the level of clumping in populations of sea cucumber. Biswas et al. (2014) also used a clumping measure to study the impact of the dilution effect in artificial populations of prey and predator in an A-life simulation.

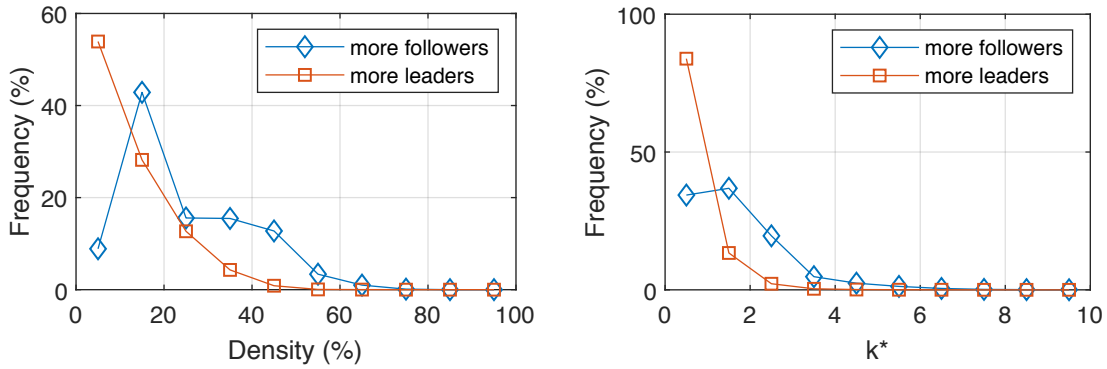

Figure 2: Distribution of rabbit densities in the training set (left), and the distribution of estimated number of clusters of rabbits in the training set (right).

5.4.1 Density-based classifier

Since our A-life simulation allows several agents to occupy a single grid cell, a higher degree of clustering in a population will likely lead to cells with a higher density of agents. Our density measure is taken from work by Biswas et al. (2014) where it was called *clumping* and defined as $1 - \frac{n}{p}$ (n is the number of occupied cells and p is the population size).[§] Our classifier builds a histogram of density values for each class in the training set (Figure 2, left). Each frame in the test set can then be classified by computing its density and assigning it to the class with the higher histogram value for the corresponding bin (we used 10 bins to partition the density range).

5.4.2 Cluster-based classifier

This classifier is similar but uses an estimate of the number of agent clusters in a frame to assign it to one of the two classes (more followers or more leaders). To do so, we first build a histogram of the number of clusters estimated for frames in the training data. To compute such estimates, we use the gap method (Tibshirani et al., 2002) with the uniform random distribution of points from a bounding box around agents in a frame as the null-reference distribution. The distance between agents is the squared Euclidean distance. The range of cluster numbers was from 1 to 10; we used $B = 100$ samples and we repeated k-means within the gap method 4 times to average W_k.

The right plot in Figure 2 shows distributions of estimated number of clusters in the training data for each class. Given a new simulation frame the cluster-based classifier first estimates the number of rabbit clusters in the image (k^*) and then assigns the frame to class with the higher histogram value (i.e., the more likely class).

[§]This measure requires knowing coordinates of the agents, so we augmented our A-life model to output this information for each simulation frame image.

5.5 Results & Discussion

We compared the average test accuracy over 30 trials for the three classifiers described above. Before we present the results we observe that the density and cluster-based classifiers were based on the rabbit coordinates for each simulation frame (with wolves excluded) whereas the deep network assessed downsampled images containing both rabbits and wolves visualized as the same gray triangles. The difference in inputs is a necessary consequence of selecting the different classifier approaches and may affect the results. Furthermore, grass-liking rabbits may also like to follow other rabbits while grass-disliking rabbits may also dislike following other rabbits. The disconnect between leading and following behaviours (which affects frame appearance) and liking and disliking grass as given by the agents' genes (which affects frame class label) may have reduced the test accuracy for all classifiers.

Table 1: The three classifiers compared on the more-leaders versus more-followers data set. Means and standard deviations over 30 trials are listed.

Classifier	**Test accuracy**
density-based	$70.4 \pm 4.8\%$
cluster-based	$74.1 \pm 2.3\%$
deep learning	$90.9 \pm 2.2\%$

Test accuracies averaged over 30 trials are listed in Table 1. AlexNet has the strongest performance, followed by the cluster-based classifier, while the density-based classifier exhibited the worst performance. Using paired sample t-tests, we found that the mean accuracy of AlexNet was significantly higher than the mean accuracy of the density-based classifier ($p \leq 10^{-18}$), and also significantly higher than the mean accuracy of the cluster-based classifier ($p \leq 10^{-24}$).

We speculate that AlexNet has a significantly higher classification accuracy than the other two classifiers due to the fact that it is able to use the shape of the agent formations in addition to frame-level aggregate statistics such as the density or the number of clusters in making its classification decisions. AlexNet also had access to the agent orientation (as each agent is visualized as a triangle) which the other two classifiers did not. On the other hand, the density- and cluster-based classifiers had access to each agent coordinates whereas AlexNet did not in the case of overlapping triangles.

5.6 Case Study: A Commercial Video Game

In this section we present a case study which tests the portability of our deep learning method of behaviour detection by training AlexNet on behaviours of evolved NPCs in a commercial video game, *Project Hastur*. This is a hybrid real-time strategy tower-defense game currently under development by Polymorphic Games, the developer of *Darwin's Demons* (Soule et al., 2017). The player places auto-fire turrets to defend against generations of evolving NPC enemies, known as the Protean Swarm (Figure 3, left), that are trying to overwhelm the player's defenses. Proteans have a digital genome that determines their morphology (body size, limb size, shape, colour, etc.), traits (sight range, resistances, speed, attack rate and damage, etc.), behaviour (attack preferences) and

special abilities (swimming or jumping). At the end of each generation (attack wave) Proteans are selected to reproduce into the next generation according to their fitness scores. Two separate fitness functions were used for selection, one function was used to select half of the population and the other fitness function was used to select the other half of the population. The parents of half the population are selected based on turret damage and the parents of the other half of the population are selected based on their proximity to the tower.

In addition to reproducing at the end of each generation, Proteans can also reproduce by attacking and consuming civilian NPCs which wander around a map. When a Protean kills a civilian it immediately creates one or more offspring that join the attacking generation. The number of offspring created in this way is inversely proportional to the Protean's size – smaller Proteans produce more offspring when they consume a civilian. This creates a secondary fitness function, as Proteans that evolve to more effectively find and consume civilians have more offspring that become potential parents at the end of the generation.

Figure 3: The left image represents evolved NPCs in a screenshot taken from a development version of *Project Hastur*. Centre and right images are partial overhead images. The centre image shows seven turrets surrounding a central tower. The Proteans charge to attack either the turrets or the tower depending on their behavioural preference. The turrets auto-fire at the Proteans. The right image shows only the Proteans.

To collect training and test data for AlexNet, we set up two experimental conditions by turning on and off the presence of civilian NPCs (classes *civilians* and *no civilians*, respectively).

In both classes, turrets were placed around a single tower (Figure 3, center). In each generation, 100 Proteans were spawned at a rate of one per second and attacked the defenses. For data-collection purposes, the turrets and the tower were made indestructible which avoided the issues of the player losing the game before sufficient data was collected or the need to regularly replace destroyed turrets. However, the Proteans' fitness was still measured by damage they would have inflicted to the turrets and the tower. Each trial was run for 30 generations and data was collected in the form of overhead images recorded once every second that showed only the Proteans (Figure 3, right). The images were then downsampled to 227×227 pixels and converted to black and white (Figure 4).

Twelve and eight game runs (each consisting of 30 generations) were recorded in the *no civilians* and *civilians* classes respectively.

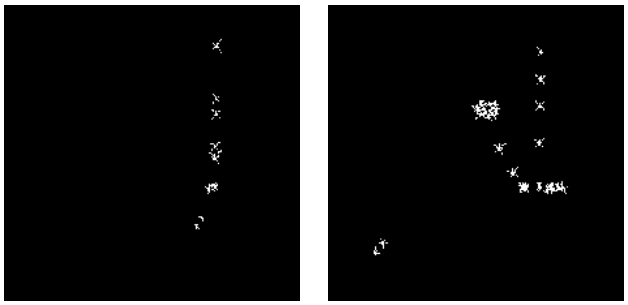

Figure 4: Representative images from the classes *no civilians* (left) and *civilians* (right) as given to the deep neural network.

In the *no civilians* class, the map was empty other than the player's tower and turrets and the spawning Proteans (Figure 4, left). In the *civilians* class, NPCs were added to the map periodically, until a total of 10 civilians (including any that survived the previous generation) had been added to map per generation. If there were civilians surviving at the end of a generation, fewer were spawned on the next generation to ensure the number of civilians on the map did not exceed 10. They had a specific spawn point (different from the Protean's spawn point) and four waypoints that they wandered between unless they were attacked by the Proteans, in which case they ran directly away from the nearest Protean. Civilians were not visualized in the images and thus AlexNet had to distinguish between the two classes only on the basis of visualized Proteans (Figure 4).

The generation process produced 59565 images in the *no civilians* and 58327 images in the *civilians* classes. We randomly selected 58327 from the larger class and then balanced the images on the black pixel colour. After colour balancing, all non-black pixels were replaced with white pixels. This resulted in 40492 black-and-white images in each class.

We conducted a sweep of the deep-learning hyperparameter space with batch size in $\{5, 10, 25, 50\}$, number of epochs in $\{1, 5, 10, 25\}$, and learning rate in $\{10^{-5}, 10^{-6}\}$. For each of 32 parameter combinations we ran 8 trials, training AlexNet on a random 75% of the game runs and testing on the remaining game runs. The best average test accuracy of $83.6 \pm 3\%$ was achieved with 10 epochs, batch size of 10 and the learning rate of 10^{-5}.

We then fixed the hyperparameters to the most successful combination and performed 30 learning trials. AlexNet achieved an average test accuracy of $81.3 \pm 4.1\%$ and the corresponding confusion matrix is in Table 2.

6. Current Shortcomings and Future Work

The A-life environment used in this paper is basic which likely limits the scope of emergent cognitively interesting agent behaviours possible in it. For instance, agents in our Netlogo simulation had built-in detection of other agents which removed the need to recognize other agents in the en-

Table 2: The confusion matrix for Project Hastur data. Means and standard deviations are for 30 learning trials.

	Actual	
Classified as	civilians	no civilians
civilians	$76 \pm 7.5\%$	$13.2 \pm 7.4\%$
no civilians	$24 \pm 7.5\%$	$86.8 \pm 7.4\%$

vironment. Furthermore, agents could immediately see each other's behaviour policy as opposed to inferring it from low-level observations. Future work will apply the methods of this paper to more complex environments which will require richer cognitive processes in the agents. For instance, we are developing a more complex A-life environment where the agents' brains are represented as deep neural networks whose topology and weights evolve over time. Such agents perceive the environment, including other agents, at the pixel level thereby necessitating various cognitive processes (e.g., recognizing a nearby agent as a friend or foe) to survive.

Second, the supervised learning approach presented in this paper requires training data from all classes. This means that the behaviour detector is trained to recognize one or more *a priori* known behaviours as opposed to novel (and therefore previously unseen) behaviours. When detecting novel behaviours in A-life simulations or bugs in video games, training examples for the interesting/buggy behaviour are not available *a priori*. Future work will investigate the ability of deep neural autoencoders (Ribeiro et al., 2017) trained only on known behaviour to detect unexpected/novel/unknown behaviour in an A-life simulation or a bug in a video game.

Another direction for future work is incorporating the time information in the input to a machine-learned behaviour detector. This can be done by using recurrent neural networks or computing a sliding average of video frames for a feed-forward network. This would allow the network to have access to changes in behavioural patterns over consecutive frames, possibly facilitating detection of more temporarily complex behaviours. Eventually such a behaviour detector may be able to reason about cognitively rich behaviours of the agents being observed, thereby demonstrating a form of meta-cognition.

7. Conclusions

In this paper we demonstrated the ability of a common, readily available off-the-shelf deep neural network to reliably distinguish between different behaviours in an A-life simulation and in a commercial video game. We anticipate such automated detectors becoming progressively more valuable to A-life researchers and video-game developers as the growing computational power enables procedural generation of cognitively richer AI agents/NPCs. In particular, since a behaviour detector implemented as a deep neural network is likely to learn and reason over a hierarchy of semantic features extracted from its low-level input, one can attempt to gain insights into the cognitive aspects of behaviour detection by inspecting the detector after training.

Acknowledgements

Devon Sigurdson discovered emergence of follower rabbits in a related Netlogo model. John Simpson, Delia Cormier and Shelby Carleton contributed to the Netlogo model used in this study. We appreciate funding from the National Science and Engineering Council and donations from Nvidia Corp. Material pertaining to *Project Hastur* is based in part upon work supported by the National Science Foundation under Cooperative Agreement No. DBI-0939454, by NSF Grant DMS-1029485, and by the Idaho Global Entrepreneur Mission (IGEM). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

Ackley, D., & Littman, M. (1991). Interactions between learning and evolution. *Artificial Life II*, *10*, 487–509.

Berman, G. J., Choi, D. M., Bialek, W., & Shaevitz, J. W. (2014). Mapping the stereotyped behaviour of freely moving fruit flies. *Journal of the Royal Society Interface*, *11*.

Biswas, R., Ofria, C., Bryson, D. M., & Wagner, A. P. (2014). Causes vs benefits in the evolution of prey grouping. *Proceedings of the Fourteenth International Conference on the Synthesis and Simulation of Living Systems* (pp. 641–648).

Bulitko, V., Carleton, S., Cormier, D., Sigurdson, D., & Simpson, J. (2017). Towards positively surprising non-player characters in video games. *Proceedings of the Experimental AI in Games Workshop at the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* (pp. 34–40).

Dell, A. I., et al. (2014). Automated image-based tracking and its application in ecology. *Trends in Ecology & Evolution*, *29*, 417–428.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Proceedings of the Twenty-Fifth International Conference on Neural Information Processing Systems* (pp. 1097–1105).

Pu, H., Lian, J., & Fan, M. (2018). Automatic recognition of flock behavior of chickens with convolutional neural network and kinect sensor. *International Journal of Pattern Recognition and Artificial Intelligence*, *32*.

Ribeiro, M., Eugênio Lazzaretti, A., & Silvério Lopes, H. (2017). A study of deep convolutional auto-encoders for anomaly detection in videos. *Pattern Recognition Letters*, *10*, 13–22.

Sanvicente-Añorve, L., Solís-Marín, F. A., Solís-Weiss, V., & Lemus-Santana, E. (2017). Population density and spatial arrangement of two holothurian species in a coral reef system: Is clumping behavior an anti-predatory strategy? *Cahiers de Biologie Marine*, *58*, 307–315.

Silver, D., et al. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, *529*, 484–489.

Silver, D., et al. (2017). Mastering the game of Go without human knowledge. *Nature*, *550*, 354.

Soule, T., Heck, S., Haynes, T. E., Wood, N., & Robison, B. D. (2017). Darwin's demons: Does evolution improve the game? *Proceedings of the European Conference on the Applications of Evolutionary Computation* (pp. 435–451). Springer.

Strandburg-Peshkin, A., Papageorgiou, D., Crofoot, M. C., & Farine, D. R. (2018). Inferring influence and leadership in moving animal groups. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, *373*.

Tibshirani, R., Walther, G., & Hastie, T. (2002). Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *63*, 411–423.

Wilensky, U. (1997). NetLogo wolf sheep predation model. *Center for Connected Learning and Computer-Based Modeling*.

Yannakakis, G. N., & Togelius, J. (2018). *Artificial intelligence and games*. Springer.