Deep Innovation Protection: Confronting the Credit Assignment Problem in Training Heterogeneous Neural Architectures

Sebastian Risi[1] **and Kenneth O. Stanley**[2]
[1] IT University of Copenhagen, Copenhagen, Denmark
[2] Uber AI, San Francisco, CA 94103
sebr@itu.dk, kennethostanley@gmail.com

Abstract

Deep reinforcement learning approaches have shown impressive results in a variety of different domains, however, more complex heterogeneous architectures such as world models require the different neural components to be trained separately instead of end-to-end. While a simple genetic algorithm recently showed end-to-end training is possible, it failed to solve a more complex 3D task. This paper presents a method called *Deep Innovation Protection* (DIP) that addresses the credit assignment problem in training complex heterogenous neural network models end-to-end for such environments. The main idea behind the approach is to employ multiobjective optimization to temporally reduce the selection pressure on specific components in multi-component network, allowing other components to adapt. We investigate the emergent representations of these evolved networks, which learn to predict properties important for the survival of the agent, without the need for a specific forward-prediction loss.

Introduction

The ability of the brain to model the world arose from the process of evolution. It evolved because it helped organisms to survive and strive in their particular environments and not because such forward prediction was explicitly optimized for. In contrast to the emergent neural representations in nature, modules of current world model approaches are often directly rewarded for their ability to predict future states of the environment (Schmidhuber 1990; Ha and Schmidhuber 2018; Hafner et al. 2018; Wayne et al. 2018). While it is undoubtedly useful to be able to explicitly encourage a model to predict what will happen next, here we are interested in the harder problem of agents that should learn to predict what is important for their survival without being explicitly rewarded for it.

A challenge in end-to-end training of complex neural models that does not require each component to be trained separately (Ha and Schmidhuber 2018), is the well-known credit assignment problem (CAP) (Minsky 1961). While deep learning has shown to be in general well suited to solve the CAP for deep networks (i.e. determining how much each weight contributes to the network's error), evidence suggests that more heterogeneous networks lack the "niceness" of conventional homogeneous networks (see Section 6.1 in Schmidhuber (2015)), requiring different training setups for each neural module in combination with evolutionary methods to solve a complex 3D task (Ha and Schmidhuber 2018).

To explore this challenge, we are building on the recently introduced world model architecture introduced by Ha and Schmidhuber (2018) but employ a novel neuroevolutionary optimization method. This agent model contains three different components: **(1)** a visual module, mapping high-dimensional inputs to a lower-dimensional representative code, **(2)** an LSTM-based memory component, and **(3)** a controller component that takes input from the visual and memory module to determine the agent's next action. In the original approach, each component of the world model was trained separately and to perform a different and specialised function, such as predicting the future. While Risi and Stanley (2019) demonstrated that these models can also be trained end-to-end through a population-based genetic algorithm (GA) that exclusively optimizes for final performance, the approach was only applied to the simpler 2D car racing domain and it is an open question how such an approach will scale to the more complex CAP in a 3D VizDoom task that first validated the effectiveness of the world model approach.

Adding support to the hypothesis that CAP is a problem in heterogeneous networks, we show that a simple genetic algorithm fails to find a solution to solving the VizDoom task and ask the question what are the missing ingredients necessary to encourage the evolution of better performing networks. The main insight in this paper is that we can view the optimization of a heterogeneous neural network (such as world models) as a *co-evolving system of multiple different sub-systems*. The other important CAP insight is that representational innovations discovered in one subsystem (e.g. the visual system learns to track moving objects) require the other sub-systems to adapt. In fact, if the other systems are not given time to adapt, such innovation will likely initially have an adversarial effect on overall performance.

In order to optimize such co-evolving heterogeneous neural systems, we propose to reduce the selection pressure on individuals whose visual or memory system was recently changed, given the controller component time to readapt. This *Deep Innovation Protection* (DIP) approach is able to find a solution to the `VizDoom:Take Cover` task, which

Copyright © 2021, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

was so far only solved by the original world model approach (Ha and Schmidhuber 2018) and a recent approach based on self-interpretable agents (Tang, Nguyen, and Ha 2020). More interestingly, the emergent models learned to predict events important for the survival of the agent, even though they were not explicitly trained to predict the future.

Additionally, our investigation into the training process shows that DIP allows evolution to carefully orchestrate the training of the components in these heterogeneous architectures. In other words, DIP is able to successfully *credit* the contributions of the different components to the overall success of the agent. We hope this work inspires more research that focuses on investigating representations emerging from approaches that do not necessarily only rely on gradient-based optimization.

Deep Innovation Protection

The hypothesis in this paper is that to optimize heterogeneous neural models end-to-end for more complex tasks requires each of its components to be carefully tuned to work well together. For example, an innovation in the visual or memory component of the network could adversely impact the controller component, leading to reduced performance and a complicated CAP. In the long run, such innovation could allow an individual to outperform its predecessors.

The agent's network design is based on the world model network introduced by Ha and Schmidhuber (2018). The network includes a visual component (VC), implemented as the encoder component of a variational autoencoder that compresses the high-dimensional sensory information into a smaller 32-dimensional representative code (Fig. 1). This code is fed into a memory component based on a recurrent LSTM (Hochreiter and Schmidhuber 1997), which should predict future representative codes based on previous information. Both the output from the sensory component and the memory component are then fed into a controller that decides on the action the agent should take at each time step. We train the model end-to-end with a genetic algorithm, in which mutations add Gaussian noise to the parameter vectors of the networks: $\theta' = \theta + \sigma\epsilon$, where $\epsilon \sim N(0, I)$.

The approach introduced in this paper aims to train heterogeneous neural systems end-to-end by temporally reducing the selection pressure on individuals with recently changed modules, allowing other components to adapt. For example, in a system in which a mutation can either affect the visual encoder, MDN-RNN or controller, selection pressure should be reduced if a mutation affects the visual component or MDN-RNN, giving the controller time to readapt to the changes in the learned representation. We employ the well-known multiobjective optimization approach NSGA-II (Deb et al. 2002), in which a second "age" objective keeps track of when a mutation changes either the visual system or the MDN-RNN. Every generation an individual's age is increased by 1, however, if a mutation changes the VC or MDN-RNN, this age objective is set to zero (lower is better). Therefore, if two neural networks reach the same performance (i.e. the same final reward), the one that had less time to adapt (i.e. whose age is lower) would have a higher chance of being selected for the next generation. The second objective is the accumulated reward received during an episode. Pseudocode of the approach applied to world models is shown in Algorithm 1.

Algorithm 1 Deep Innovation Protection

```
Generate random population of size N with age objectives set to 0
for generation = 1 to i do
    for Individual in Population do
        Objective[1] = age
        Objective[2] = accumulated task reward
        Increase individual's age by 1
    end for
    Assign ranks based on Pareto fronts
    Generate set of non-dominated solutions
    Add solutions, starting from first front, until number solution = N
    Generate child population through binary tournament selection and mutations
    Reset age to 0 for all individuals whose VC or MDN-RNN was mutated
end for
```

It is important to note that this novel approach is different to the traditional usage of "age" in multi-objective optimization, in which age is used to increase diversity and keeps track of how long individuals have been in the population (Hornby 2006; Schmidt and Lipson 2011). In the approach in this paper, age counts how many generations the controller component of an individual had time to adapt to an unchanged visual and memory system.

In the original world model approach the visual and memory component were trained separately and through unsupervised learning based on data from random rollouts. We optimize the multi-component architecture in our work through a genetic algorithm without evaluating each component individually. In other words, the VC is not directly optimized to reconstruct the original input data and neither is the memory component optimized to predict the next time step; the whole network is trained in an end-to-end fashion. Here we are interested in what type of neural representations emerge by themselves that allow the agent to solve the given task.

Experiments

In the experiments presented here an agent is trained to solve the car racing tasks, and the more challenging VizDoom task (Kempka et al. 2016) from 64×64 RGB pixel inputs (Fig. 2). These two tasks were chosen to test the generality of the approach, with one requiring 2D top-down control (`CarRacing-v0`) and the other task requering the control of an agent from a first-person 3D view (VizDoom).

In the continuous control task `CarRacing-v0` (Klimov 2016) the agent is presented with a new procedurally generated track every episode, receiving a reward of -0.1 every frame and a reward of +100/N for each visited track tile, where N is the total number of tiles in the track. The network controlling the agent (Fig. 1) has three outputs to control left/right steering, acceleration and braking. Training agents in procedurally generated environments has shown to

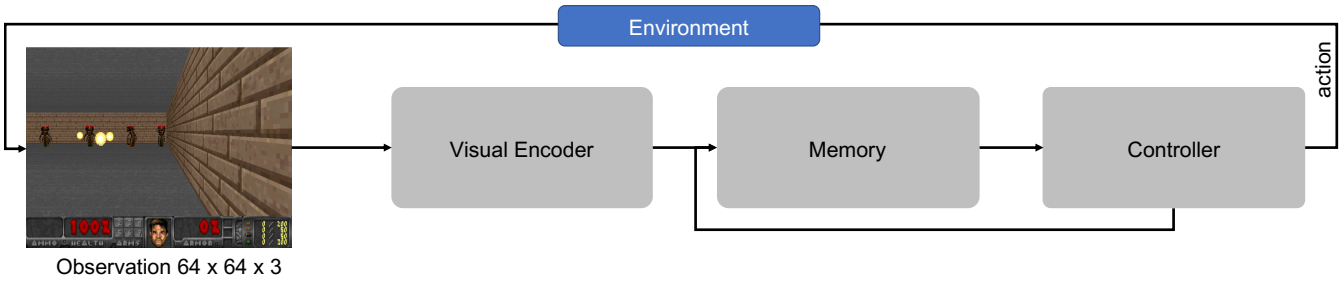
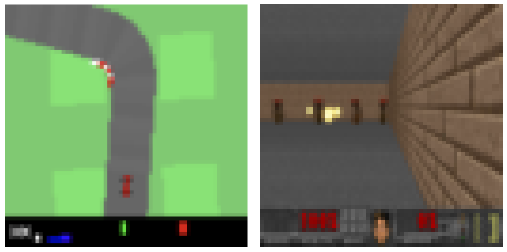


Figure 1: *Agent Model.* The agent model consists of three modules. A visual component that produces a latent code z_t at each time step t, which is concatenated with the hidden state h_t of the LSTM-based memory component that takes z_t and previously performed action a_{t-1} as input. The combined vector (z_t, h_t) is input into the controller component to determine the next action of the agent. In this paper, the agent model is trained end-to-end with a multiobjective genetic algorithm.



(a) `CarRacing` (b) `VizDoom`

Figure 2: In the `CarRacing-v0` task the agent has to learn to drive across many procedurally generated tracks as fast as possible from 64 $\times$64 RGB color images. In the `VizDoom:Take Cover` domain the agent has to learn to avoid fireballs and to stay alive as long as possible.

significantly increase their generality and avoid overfitting (Risi and Togelius 2020; Justesen et al. 2018; Zhang et al. 2018; Cobbe et al. 2018).

In the `VizDoom:Take Cover` task the agent has to try to stay alive for 2,100 timesteps, while avoiding fireballs shot at it by strafing to the left or the right. The agent receives a +1 reward for every frame it is alive. The network controlling the agent has one output a to control left ($a < -0.3$) and right strafing ($a > 0.3$), or otherwise standing still. In this domain, a solution is defined as surviving for over 750 timesteps, averaged across 100 random rollouts (Kempka et al. 2016).

Following the NSGA-II approach, individuals for the next generation are determined stochastically through 2-way tournament selection from the 50% highest ranked individuals in the population (Algorithm 1). No crossover operation was employed. The population size was 200. Because of the randomness in this domain, we evaluate the top three individuals of each generation one additional time to get a better estimate of the true elite. We compare a total of four different approaches:

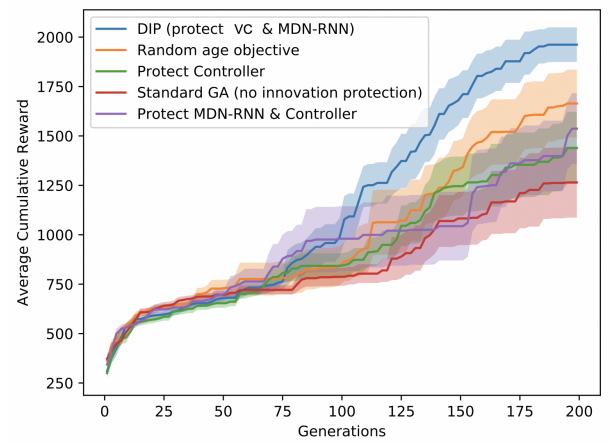
1. **Deep innovation protection (DIP):** The age objective is reset to zero when either the VC or MDN-RNN is changed. The idea behind this approach is that the controller should get time to readapt if one of the components that precede it in the network change.
2. **Controller innovation protection:** Here the age objective is set to zero if the controller changes. This setting tests if protecting components upstream can be effective in optimizing heterogeneous neural models.
3. **MDN-RNN & Controller innovation protection:** This setup is the same as the controller protection approach but we additionally reset age if the MDN-RNN changes. On average, this treatment will reset the age objective as often as DIP.
4. **Random age objective:** In this setup the age objective is assigned a random number between [0, 20] at each evaluation. This treatment tests if better performance can be reached just through introducing more diversity in the population.
5. **Standard GA - no innovation protection:** In this non-multi-objective setup, which is the same one as introduced in Risi and Stanley (2019), only the accumulated reward is taken into account when evaluating individuals.

For all treatments, a mutation has an equal probability to either mutate the visual, memory, or controller component of the network. Interestingly, while this approach performs similarly well to an approach that always mutates all components for the `CarRacing-v0` task (Risi and Stanley 2019), we noticed that it performs significantly worse in the more complicated VizDoom domain. This result suggests that the more complex the tasks, the more important it is to be able to selectively fine-tune each different component in a complex neural architecture.

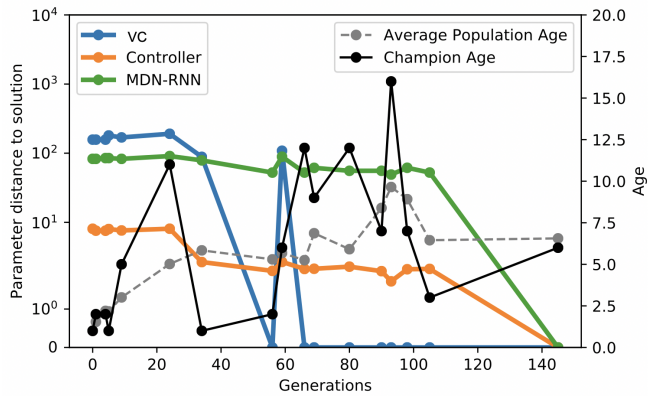
Optimization and Model Details

The genetic algorithm σ was determined empirically and set to 0.03 for the experiments in this paper. The code for the DIP approach is available at: github.com/sebastianrisi/dip.

The sensory model is implemented as a variational autoencoder that compresses the high-dimensional input to a latent vector z. The VC takes as input an RGB image of size $64 \times 64 \times 3$, which is passed through four convolutional layers, all with stride 2. The network's weights are set using the default He PyTorch initilisation (He et al. 2015), with the



(a)



(b)

Figure 3: *VizDoom Evolutionary Training.* Shown is (a) mean performance over generations together with one standard error. For one representative run of DIP (b), we plot the euclidean distances of the weights of the intermediate solutions (i.e. individuals with the highest task reward discovered so far) compared to the final solution in addition to their age and the average population age.

resulting tensor being sampled from $\mathcal{U}(-bound, bound)$, where $bound = \sqrt{\frac{1}{fan_in}}$. The memory model (Ha and Schmidhuber 2018) combines a recurrent LSTM network with a mixture density Gaussian model as network outputs, known as a MDN-RNN (Ha and Eck 2017; Graves 2013a). The network has 256 hidden nodes and models $P(z_{t+1}|a_t, z_t, h_t)$, where a_t is the action taken by the agent at time t and h_t is the hidden state of the recurrent network. Similar models have previously been used for generating sequences of sketches (Ha and Eck 2017) and handwriting (Graves 2013b). The controller component is a simple linear model that directly maps z_t and h_t to actions: $a_t = W_c[z_t h_t] + b_c$, where W_c and b_c are weight matrix and bias vector.

Experimental Results

All results are averaged over ten independent evolutionary runs. In the car racing domain we find that there is no noticeable difference between an approach with and without innovation protection and both can solve the domain with a reward of 905±80 and 903±72, respectively. However, in the more complex VizDoom task (Fig. 3a), the DIP approach that protects innovations in both VC and MDN-RNN, significantly outperforms all other approaches during training. The approach is able to find a solution to the task, effectively avoiding fireballs and reaching an average score of 824.33 (sd ± 491.59).

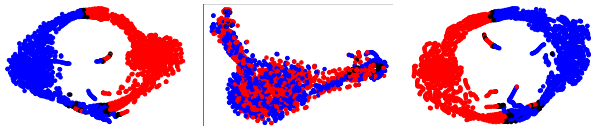


Figure 4: *Still frames of a learned policy.* The agent learned to primarily pay attention to the walls and fireballs, while ignoring the floor and ceiling. Interestingly the agent also seems to pay attention to the health and ammo indicator.

To better understand the network's behavior, we calculate perturbation-based saliency maps to determine the parts of the environment the agent is paying attention to (Fig. 4). The idea behind perturbation-based saliency maps is to measure to what extent the output of the model changes if parts of the input image are altered (Greydanus et al. 2017). Not surprisingly, the agent learned to pay particular attention to the walls, fireballs, and the position of the monsters.

The better performance of the random age objective compared to no innovation protection suggests that increasing diversity in the population improves performance but less effectively than selectivity resetting age as in DIP. Interestingly, the controller and the MDN-RNN&Controller protection approach perform less well, confirming our hypothesis that it is important to protect innovations upstream in the network for downstream components.

Learned Representations We further investigate what type of world model can emerge from an evolutionary process that does not directly optimize for forward prediction or reconstruction loss. To gain insights into the learned representations we employ the t-SNE dimensionality reduction technique (Maaten and Hinton 2008), which has proven valuable for visualizing the inner workings of deep neural networks (Such et al. 2018; Mnih et al. 2015). We are particularly interested in the information contained in the compressed 32-dimensional vector of the VC and the information stored in the hidden states of the MDN-RNN (which are both fed into the controller that decides on the agent's action). Different combinations of sequences of these latent vectors collected during one rollout are visualized in two dimensions in Fig. 5. Interestingly, while the 32-dimensional z vector from the VC does not contain enough information to infer the correct action, either the hidden state alone or in combination with z results in grouping the states into two distinct classes (one for moving left and one for moving right). The temporal dimension captured by the recurrent network proves invaluable in deciding what action is



(a) z+hidden (b) z alone (c) hidden alone

Figure 5: t-SNE mapping of the latent+hidden vector (a), latent vector alone (b), and hidden vector alone (c). While the compressed latent vector is not enough to infer the correct action (b), the hidden LSTM vector alone contains enough information for the agent to decide on the correct action (c). Red = strafe left, blue = strafe right, black = no movement.

best. For example, not getting stuck in a position that makes avoiding incoming fireballs impossible, seems to require a level of forward prediction by the agent. To gain a deeper understanding of this issue we look more closely into the learned temporal representation next.

Learned Forward Model Dynamics In order to analyze the learned temporal dynamics of the forward model, we are taking a closer look at the average activation x_t of all 256 hidden nodes at time step t and how much they differ from the overall average across all time steps $\bar{X} = \frac{1}{N}\sum_1^N \bar{x_t}$. The variance of $\bar{x_t}$ is thus calculated as $\sigma_t = (\bar{X} - \bar{x_t})^2$, and normalized to the range $[0, 1]$ before plotting. The hypothesis is that activation levels far from the mean might indicate a higher importance and should have a greater impact on the agent's controller component. In other words, they likely indicate critical situations in which the agent needs to pay particular attention to the predictions of the MDN-RNN. Fig. 6 depicts frames from the learned policies in two different situations, which shows that the magnitude of LSTM activations are closely tied to specific situations. The forward model does not seem to react to fireballs by themselves but instead depends on the agent being in the line of impact of an approaching fireball, which is critical information for the agent to stay alive.

Evolutionary Innovations In addition to analyzing the learned representations of the final networks, it is interesting to study the different stepping stones evolution discovered to solve the VizDoom task. We show one particular evolutionary run in Fig. 7, with other ones following similar progressions. In the first 30 generations the agent starts to learn to pay attention to fireballs but only tries avoiding them by either standing still or moving to the right. A jump in performance happens around generation 34 when the agent starts to discover moving to either the left or right; however, the learned representation between moving left or right is not well defined yet. This changes around generation 56, leading to another jump in fitness and some generations of quick fine-tuning later the agent is able to differentiate well between situations requiring different actions, managing to survive for the whole length of the episode. Motivated by the approach of Raghu et al. (2017) to analyse the gradient descent-based training of neural networks, we investigate the weight distances of the world model components of the best-performing networks found during training to the final solution representation (Fig. 3b). The VC is the component with the steepest decrease in distance with a noticeable jump around generation 60 due to another lineage taking over. The MDN-RNN is optimized slowest, which is likely due to the fact that the correct forward model dynamics are more complicated to discover than the visual component. These results suggest that DIP is able to orchestrate the training of these heterogeneous world model architectures in an automated way, successfully solving the underlying CAP.

Reward and Age Objective We performed an analysis of the (1) cumulative reward per age and (2) the number of individuals with a certain age averaged across all ten runs and all generations (Fig. 8). While the average reward increases with age, there are fewer and fewer individuals at higher age levels. This result suggest that the two objectives are in competition with each other, motivating the choice for a multi-objective optimization approach; staying alive for longer becomes increasingly difficult and a high age needs to be compensated for by a high task reward.

Sample Efficiency Comparison While evolutionary algorithms are typically regarded as requiring many samples, DIP is surprisingly sample efficient and competitive with other solutions to the DoomTakeCover and CarRacing task. The other reported solutions that solve both of these tasks are the world model approach by Ha and Schmidhuber (2018) and an evolutionary self-attention approach (Tang, Nguyen, and Ha 2020). In case of the CarRacing task, Tang, Nguyen, and Ha (2020) report that they can solve the tasks reliable after 1,000 generations (with a slightly larger population size of 256 compared to our population size of 200). The world model approach uses a mix of different methods (Ha and Schmidhuber 2018), which makes comparing sample efficiency slightly more complicated. The world model approach finds a solution to the CarRacing task in 1,800 generations with an already trained VAE and MDN-RNN. DIP can solve CarRacing after 1,200 generations (without requiring pre-training) and is thus similarly sample efficient to the end-to-end training approach in Tang, Nguyen, and Ha (2020). The purely evolutionary training in Tang, Nguyen, and Ha (2020) can reliable solve the DoomTakeCover task after around 1,000 generations. DIP solves the tasks in only 200 generations. The world model approach only trains the DoomTakeCover agent in a simulated dream environment and then transfers the controller to the actual environment. Evolutionary training of the learned world model is fast, since it doesn't require simulated graphics, and takes around 1,500 generations. However, it relies on training the VAE and MDN-RNN with 10,000 random rollouts.

Related Work

A variety of different RL algorithms have recently been shown to work well on a diverse set of problems when combined with the representative power of deep neural networks (Mnih et al. 2015; Schulman et al. 2015, 2017). While most approaches are based on variations of Q-learning (Mnih et al. 2015) or policy gradient methods (Schulman et al. 2015, 2017), recently evolutionary-based methods have emerged as a promising alternative for some domains

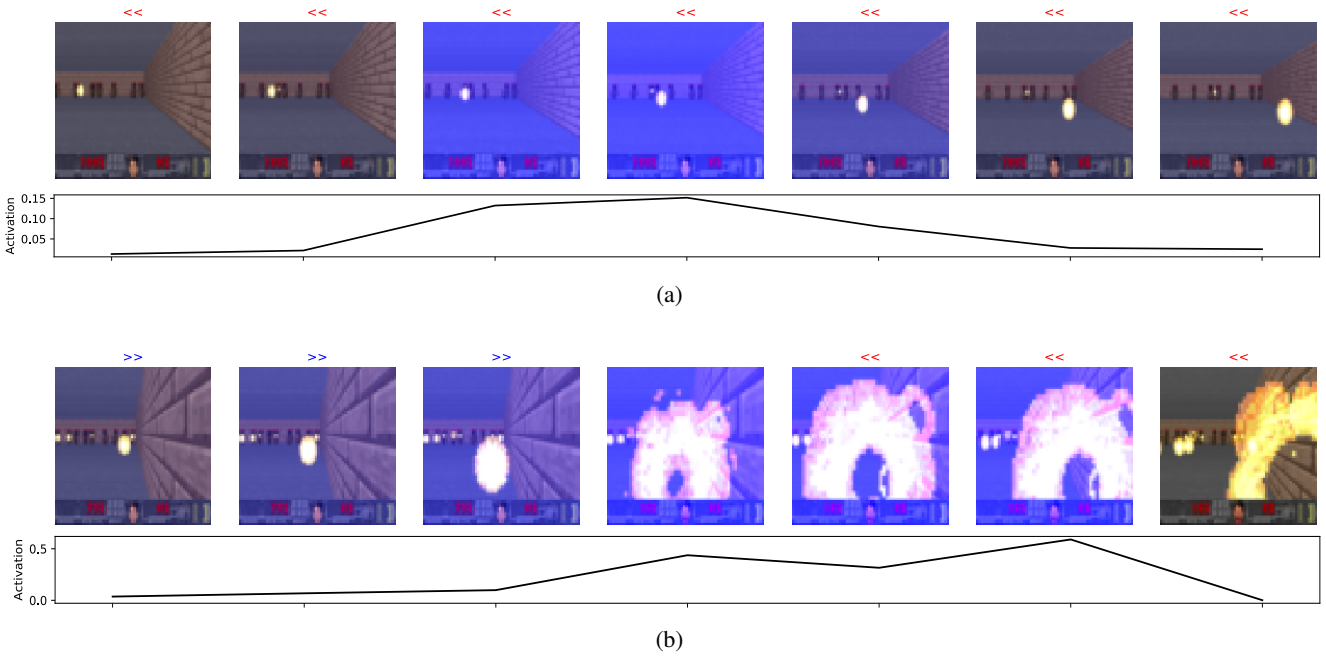


Figure 6: *Average activation levels of LSTM in two different situations.* For visualization purposes only, images are colored more or less blue depending on the LSTM activations. The forward model seems to have learned to predict if a fireball would hit the agent at the current position. In (a) the agent can take advantage of that information to avoid the fireball while the agent does not have enough time to escape in situation (b) and gets hit. Shown on top are the actions the agent takes in each frame.

(Such et al. 2017; Salimans et al. 2017). Salimans et al. (2017) showed that a type of evolution strategy (ES) can reach competitive performance in the Atari benchmark and at controlling robots in MuJoCo. Additionally, Such et al. (2017) demonstrated that a simple genetic algorithm is in fact able to reach similar performance to deep RL methods such as DQN or A3C. Earlier approaches that evolved neural networks for RL tasks worked well in complex RL tasks with lower-dimensional input spaces (Stanley and Miikkulainen 2002; Floreano, Dürr, and Mattiussi 2008; Risi and Togelius 2017). Evolutionary approaches solving 3D tasks directly from pixels has so far proven difficult although a few notable approaches exist (Koutník et al. 2013; Alvernaz and Togelius 2017; Poulsen et al. 2017; Lehman et al. 2018).

For complex agent models, different network components can be trained separately (Wahlström, Schön, and Deisenroth 2015; Ha and Schmidhuber 2018). For example, in the world model approach (Ha and Schmidhuber 2018), the authors first train a variational autoencoder (VAE) on 10,000 rollouts from a random policy to compress the high-dimensional sensory data and then train a recurrent network to predict the next latent code. Only after this process is a smaller controller network trained to perform the actual task, taking information from both the VAE and recurrent network as input to determine the action the agent should perform.

Evolutionary approaches solving 3D tasks directly from pixels has so far proven difficult although a few notable approaches exist. Koutník et al. (2013) evolved an indirectly encoded and recurrent controller for car driving in TORCS, which learned to drive based on a raw 64×64 pixel image. The approach was based on an indirect encoding of the network's weights analogous to the JPEG compression in images. To scale to 3D FPS tasks, Alvernaz and Togelius (2017) first trained an autoencoder in an unsupervised way and then evolved the controller giving the compressed representation as input. In another approach, Poulsen et al. (2017) trained an object recognizer in a supervised way and then in a separate step evolved a controller module. More recently, Lehman et al. (2018) introduced an approach called *safe mutations*, in which the magnitude of mutations to weight connections is scaled based on the sensitivity of the network's output to that weight. It allowed the evolution of large-scale deep networks for a simple 3D maze task and is a complementary approach that could be combined with DIP.

The approach introduced in this paper can be viewed as a form of diversity maintenance, in which selection pressure on certain mutated neural networks is reduced. Many other methods for encouraging diversity (Mouret and Doncieux 2012) were invented by the evolutionary computation community, such as novelty search (Lehman and Stanley 2008), quality diversity (Pugh, Soros, and Stanley 2016), or speciation (Stanley and Miikkulainen 2002).

For increasing diversity, algorithms often introduce new individuals into the population. In the ALPS approach by Hornby (2006), the population is segregated into different layers depending on when they were introduced into the population and newly generated individuals are introduced into the "newest" layer to increase diversity. Schmidt and

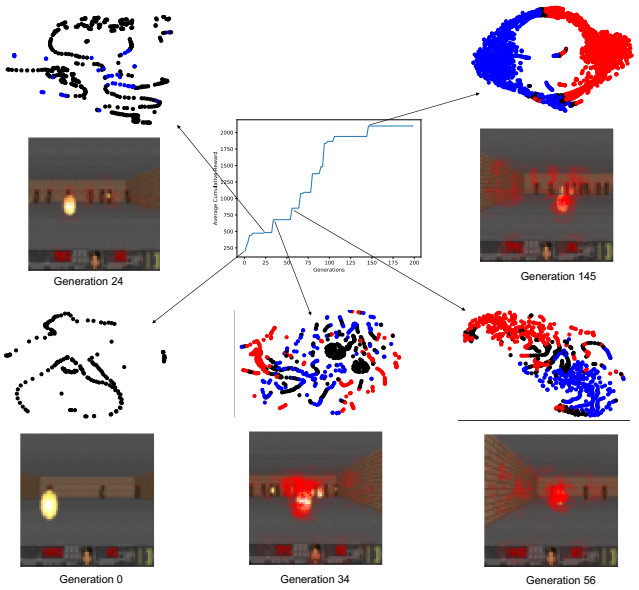


Figure 7: *Development of the evolved representation.* Shown are t-SNE mappings of the 288-dimensional vectors (32-dimensional latent vectors + 256-dimensional hidden state vector) together with saliency maps of specific game situations. Early on in evolution the agent starts paying attention to the fireballs (generation 24) but only moves to the right (blue) or stands still (black). Starting around generation 34 the agent starts to move to the left and right, with the saliency maps becoming more pronounced. From generation 56 on the compressed learned representation (latent vector+hidden state vector) allows the agent to infer the correct action almost all the time. The champion discovered in generation 145 discovered a visual encoder and LSTM mapping that shows a clear division for left and right strafing actions.

Lipson (2011) combine this idea with a multi-objective approach, in which individuals are rewarded for performance and for how many generations have passed since they have been introduced into the population. Similar to the approach by Cheney et al. (2018) to co-evolve morphologies and neural controller, and in contrast to previous approaches (Hornby 2006; Schmidt and Lipson 2011), DIP does not introduce new random individuals into the generation but instead resets the "age" of individuals whose sensory or memory system have been mutated. That is, it is not a measure of how long the individual has been in the population.

Approaches to learning dynamical models have mainly focused on gradient descent-based methods, with early work on RNNs in the 1990s (Schmidhuber 1990). More recent work includes PILCO (Deisenroth and Rasmussen 2011), which is a probabilistic model-based policy search method and Black-DROPS (Chatzilygeroudis et al. 2017) that employs CMA-ES for data-efficient optimization of complex control problems. Additionally, interest has increased in learning dynamical models directly from high-dimensional images for robotic tasks (Watter et al. 2015; Hafner et al. 2018) and also video games (Guzdial, Li, and Riedl 2017). Work on evolving forward models has mainly focused on

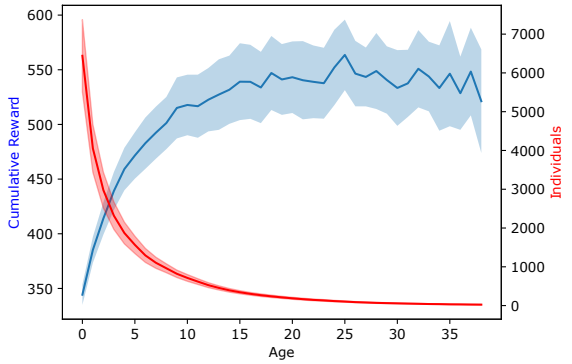


Figure 8: Average reward across ages and number of individuals per age.

neural networks that contain orders of magnitude fewer connections and lower-dimensional feature vectors (Norouzzadeh and Clune 2016) than the models in this paper.

Discussion and Future Work

The paper demonstrated that a predictive representation for a 3D task can emerge under the right circumstances without being explicitly rewarded for it. To encourage this emergence and address the inherent credit assignment problem of complex heterogeneous networks, we introduced the *deep innovation protection* approach that can dynamically reduce the selection pressure for different components in such neural architectures. The main insight is that when components upstream in the network change, such as the visual or memory system in a world model, components downstream need time to adapt to changes in those learned representations.

The neural model learned to represent situations that require similar actions with similar latent and hidden codes (Fig. 5 and 7). Additionally, without a specific forward-prediction loss, the agent learned to predict "useful" events that are necessary for its survival (e.g. predicting when the agent is in the line-of-fire of a fireball). In the future it will be interesting to compare the differences and similarities of emergent representations and learning dynamics resulting from evolutionary and gradient descent-based optimization approaches (Raghu et al. 2017).

A natural extension to this work is to evolve the neural architectures in addition to the weights of the network. Searching for neural architectures in RL has previously only been applied to smaller networks (Risi and Stanley 2012; Stanley and Miikkulainen 2002; Stanley et al. 2019; Gaier and Ha 2019; Risi and Togelius 2017; Floreano, Dürr, and Mattiussi 2008) but could potentially now be scaled to more complex tasks. While our innovation protection approach is based on evolution, ideas presented here could also be incorporated in gradient descent-based approaches that optimize neural systems with multiple interacting components end-to-end.

Broader Impact

The ethical and future societal consequences of this work are hard to predict but likely similar to other work dealing with solving complex reinforcement learning problems. Because these approaches are rather task agnostic, they could potentially be used to train autonomous robots or drones in areas that have both a positive and negative impact on society. While positive application can include delivery drones than can learn from visual feedback to reach otherwise hard to access places, other more worrisome military applications are also imaginable. The approach presented in this paper is far from being deployed in these areas, but it its important to discuss its potential long-term consequences early on.

Acknowledgments

We would like to thank Nick Cheney, Jeff Clune, David Ha, and the members of Uber AI for helpful comments and discussions on ideas in this paper.

References

Alvernaz, S.; and Togelius, J. 2017. Autoencoder-augmented neuroevolution for visual Doom playing. In *Computational Intelligence and Games (CIG), 2017 IEEE Conference on*, 1–8. IEEE.

Chatzilygeroudis, K.; Rama, R.; Kaushik, R.; Goepp, D.; Vassiliades, V.; and Mouret, J.-B. 2017. Black-box data-efficient policy search for robotics. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 51–58. IEEE.

Cheney, N.; Bongard, J.; SunSpiral, V.; and Lipson, H. 2018. Scalable co-optimization of morphology and control in embodied machines. *Journal of The Royal Society Interface* 15(143). ISSN 1742-5689. doi:10.1098/rsif.2017.0937. URL http://rsif.royalsocietypublishing.org/content/15/143/20170937.

Cobbe, K.; Klimov, O.; Hesse, C.; Kim, T.; and Schulman, J. 2018. Quantifying generalization in reinforcement learning. *arXiv preprint arXiv:1812.02341* .

Deb, K.; Pratap, A.; Agarwal, S.; and Meyarivan, T. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation* 6(2): 182–197.

Deisenroth, M.; and Rasmussen, C. E. 2011. PILCO: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, 465–472.

Floreano, D.; Dürr, P.; and Mattiussi, C. 2008. Neuroevolution: from architectures to learning. *Evolutionary Intelligence* 1(1): 47–62.

Gaier, A.; and Ha, D. 2019. Weight Agnostic Neural Networks. *arXiv preprint arXiv:1906.04358* .

Graves, A. 2013a. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850* .

Graves, A. 2013b. Hallucination with recurrent neural networks. URL https://www.youtube.com/watch?v=-yX1SYeDHbg&t=49m33s.

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

Guzdial, M.; Li, B.; and Riedl, M. O. 2017. Game Engine Learning from Video. In *IJCAI*, 3707–3713.

Ha, D.; and Eck, D. 2017. A neural representation of sketch drawings. *arXiv preprint arXiv:1704.03477* .

Ha, D.; and Schmidhuber, J. 2018. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems*, 2455–2467.

Hafner, D.; Lillicrap, T.; Fischer, I.; Villegas, R.; Ha, D.; Lee, H.; and Davidson, J. 2018. Learning latent dynamics for planning from pixels. *arXiv preprint arXiv:1811.04551* .

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, 1026–1034.

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

Hornby, G. S. 2006. ALPS: the age-layered population structure for reducing the problem of premature convergence. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, 815–822. ACM.

Justesen, N.; Torrado, R. R.; Bontrager, P.; Khalifa, A.; Togelius, J.; and Risi, S. 2018. Illuminating Generalization in Deep Reinforcement Learning through Procedural Level Generation. *NeurIPS 2018 Workshop on Deep Reinforcement Learning* .

Kempka, M.; Wydmuch, M.; Runc, G.; Toczek, J.; and Jaśkowski, W. 2016. Vizdoom: A Doom-based AI research platform for visual reinforcement learning. In *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, 1–8. IEEE.

Klimov, O. 2016. Carracing-v0. URL https://gym.openai.com/envs/CarRacing-v0/.

Koutník, J.; Cuccu, G.; Schmidhuber, J.; and Gomez, F. 2013. Evolving large-scale neural networks for vision-based reinforcement learning. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, 1061–1068. ACM.

Lehman, J.; Chen, J.; Clune, J.; and Stanley, K. O. 2018. Safe mutations for deep and recurrent neural networks through output gradients. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 117–124. ACM.

Lehman, J.; and Stanley, K. O. 2008. Exploiting open-endedness to solve problems through the search for novelty. In *ALIFE*, 329–336.

Maaten, L. v. d.; and Hinton, G. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9(Nov): 2579–2605.

Minsky, M. 1961. Steps toward artificial intelligence. *Proceedings of the IRE* 49(1): 8–30.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540): 529.

Mouret, J.-B.; and Doncieux, S. 2012. Encouraging behavioral diversity in evolutionary robotics: An empirical study. *Evolutionary computation* 20(1): 91–133.

Norouzzadeh, M. S.; and Clune, J. 2016. Neuromodulation improves the evolution of forward models. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, 157–164. ACM.

Poulsen, A. P.; Thorhauge, M.; Funch, M. H.; and Risi, S. 2017. DLNE: A hybridization of deep learning and neuroevolution for visual control. In *Computational Intelligence and Games (CIG), 2017 IEEE Conference on*, 256–263. IEEE.

Pugh, J. K.; Soros, L. B.; and Stanley, K. O. 2016. Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI* 3: 40.

Raghu, M.; Gilmer, J.; Yosinski, J.; and Sohl-Dickstein, J. 2017. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In *Advances in Neural Information Processing Systems*, 6076–6085.

Risi, S.; and Stanley, K. O. 2012. An enhanced hypercube-based encoding for evolving the placement, density, and connectivity of neurons. *Artificial life* 18(4): 331–363.

Risi, S.; and Stanley, K. O. 2019. Deep Neuroevolution of Recurrent and Discrete World Models. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '19, 456–462. New York, NY, USA: ACM. ISBN 978-1-4503-6111-8. doi:10.1145/3321707.3321817. URL http://doi.acm.org/10.1145/3321707.3321817.

Risi, S.; and Togelius, J. 2017. Neuroevolution in games: State of the art and open challenges. *IEEE Transactions on Computational Intelligence and AI in Games* 9(1): 25–41.

Risi, S.; and Togelius, J. 2020. Increasing generality in machine learning through procedural content generation. *Nature Machine Intelligence* 2(8): 428–436.

Salimans, T.; Ho, J.; Chen, X.; Sidor, S.; and Sutskever, I. 2017. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864* .

Schmidhuber, J. 1990. An on-line algorithm for dynamic reinforcement learning and planning in reactive environments. In *1990 IJCNN international joint conference on neural networks*, 253–258. IEEE.

Schmidhuber, J. 2015. Deep learning in neural networks: An overview. *Neural networks* 61: 85–117.

Schmidt, M.; and Lipson, H. 2011. Age-fitness pareto optimization. In *Genetic programming theory and practice VIII*, 129–146. Springer.

Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; and Moritz, P. 2015. Trust region policy optimization. In *International Conference on Machine Learning*, 1889–1897.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* .

Stanley, K. O.; Clune, J.; Lehman, J.; and Miikkulainen, R. 2019. Designing neural networks through neuroevolution. *Nature Machine Intelligence* 1(1): 24–35.

Stanley, K. O.; and Miikkulainen, R. 2002. Evolving neural networks through augmenting topologies. *Evolutionary computation* 10(2): 99–127.

Such, F. P.; Madhavan, V.; Conti, E.; Lehman, J.; Stanley, K. O.; and Clune, J. 2017. Deep neuroevolution: genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *arXiv preprint arXiv:1712.06567* .

Such, F. P.; Madhavan, V.; Liu, R.; Wang, R.; Castro, P. S.; Li, Y.; Schubert, L.; Bellemare, M.; Clune, J.; and Lehman, J. 2018. An Atari model zoo for analyzing, visualizing, and comparing deep reinforcement learning agents. *arXiv preprint arXiv:1812.07069* .

Tang, Y.; Nguyen, D.; and Ha, D. 2020. Neuroevolution of self-interpretable agents. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, 414–424.

Wahlström, N.; Schön, T. B.; and Deisenroth, M. P. 2015. From pixels to torques: Policy learning with deep dynamical models. *arXiv preprint arXiv:1502.02251* .

Watter, M.; Springenberg, J.; Boedecker, J.; and Riedmiller, M. 2015. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in neural information processing systems*, 2746–2754.

Wayne, G.; Hung, C.-C.; Amos, D.; Mirza, M.; Ahuja, A.; Grabska-Barwinska, A.; Rae, J.; Mirowski, P.; Leibo, J. Z.; Santoro, A.; et al. 2018. Unsupervised Predictive Memory in a Goal-Directed Agent. *arXiv:1803.10760* .

Zhang, C.; Vinyals, O.; Munos, R.; and Bengio, S. 2018. A study on overfitting in deep reinforcement learning. *arXiv preprint arXiv:1804.06893* .