Intrinsically Motivated NeuroEvolution for Vision-Based Reinforcement Learning

Giuseppe Cuccu, Matthew Luciw, Jürgen Schmidhuber and Faustino Gomez
IDSIA / University of Lugano / SUPSI / 6928 Manno-Lugano, Switzerland
Email: {giuse, matthew, juergen, tino}@idsia.ch

***Abstract*—Neuroevolution, the artificial evolution of neural networks, has shown great promise on continuous reinforcement learning tasks that require memory. However, it is not yet directly applicable to realistic embedded agents using high-dimensional (e.g. raw video images) inputs, requiring very large networks. In this paper, neuroevolution is combined with an unsupervised sensory pre-processor or *compressor* that is trained on images generated from the environment by the population of evolving recurrent neural network controllers. The compressor not only reduces the input cardinality of the controllers, but also biases the search toward novel controllers by rewarding those controllers that discover images that it reconstructs poorly. The method is successfully demonstrated on a vision-based version of the well-known mountain car benchmark, where controllers receive *only* single high-dimensional visual images of the environment, from a third-person perspective, instead of the standard two-dimensional state vector which includes information about velocity.**

I. Introduction

Neuroevolution, the artificial evolution of neural networks [2,27], has advantages over classical, single-agent learning methods for continuous reinforcement learning tasks that require memory [6,22]. Despite all the recent progress, however, such methods are not yet directly applicable to realistic embedded agents with raw video input, since images are typically high-dimensional, requiring thousands of network inputs, and therefore potentially very large networks. Most approaches to scaling these methods have focused on indirect encodings where relatively small network descriptions are transformed via a complex mapping into networks of arbitrary size [3,8,12].

While this biologically inspired approach is promising, in this paper, we present an alternative approach to scaling neuroevolutionary RL to high-dimensional input spaces. Rather than forcing the evolving networks to cope with high-dimensional observations directly, an unsupervised learner (UL) is used to "compress" the often very redundant sensory input into a compact code that requires a only a few input units from the controller's side.

To our knowledge, no online system combining UL and neuroevolution has been explored previously. Up to now, this general approach of using UL as a preprocessor has been studied only in the context of single-agent RL (i.e. TD, policy gradients, etc), where the agent simultaneously learns both the mapping from the compressed features to actions, and the features themselves from the observations provoked by the actions [1,4,11,13,14,16]. A problem with this coupled system is that, since the learned features depend on how the agent behaves and vice-versa, good features may never form because the agent's initial, poor policy results in biased sampling of the environment.

The approach presented in this paper addresses this bootstrapping issue by using a single unsupervised module that is trained on data generated by *all* of the individuals in an evolving population of recurrent neural network (RNN) policies. Because many different policies are evaluated for each generation, a potentially broader, less biased sampling of the observation space is achieved, from which useful features can be learned more reliably.

Furthermore, since the compressor is adapted online, its own asynchronous learning process can be used to drive evolution in search of novelty. If an individual provides data that the compressor decompresses (reconstructs) poorly, this means that its behavior has not been seen before (or not very often), and, therefore its fitness should be boosted to promote behavioral novelty in the population. This mechanism allows evolution to make progress even when the raw fitness (i.e. the fitness related to performing the task) in the population is flat, in a manner analogous to exploration based on *intrinsic motivation* in the single-agent RL setting [18]–[20].

The particular instantiation of this Unsupervised Learning plus Evolutionary Reinforcement Learning (UL-ERL) framework, evaluated in this paper, is called VQ-SNES, as it uses online vector quantization (VQ; [7]) for the adaptive compressor and Separable Natural Evolution Strategies (SNES; [17]) for the neuroevolution component. VQ-SNES is tested on a vision-based version of the mountain car benchmark, with results demonstrating both the effectiveness of simultaneous online training of the compressor and neuroevolution, and the beneficial effect of compressor-based intrinsic motivation on evolutionary exploration.

The paper is organized as follows. Section II discusses issues related to sensory compression. Section III describes the overall UL-ERL architecture, while Section IV details VQ-SNES. Section V presents the experimental results on the high-dimensional vision-based mountain car task. Section VI concludes the paper.

II. Sensory Compression

The problem class of interest in this work is concerned with learning control for reward maximizing tasks under high-dimensional observations, where the environment can be continuous and partially observable. A high-level description of the system is as follows:

$$\mathbf{y}_t = C(\mathbf{o}_t) \tag{1}$$
$$\mathbf{a}_t = \pi(\mathbf{y}_t, \mathbf{m}_t) \tag{2}$$
$$\mathbf{m}_{t+1} = G(\mathbf{m}_t, \mathbf{y}_t, \mathbf{a}_t) \tag{3}$$

where $\mathbf{o}_t \in \mathcal{R}^{dim}$ is the observation at time t, which is encoded by compressor C into a code vector $\mathbf{y}_t \in \mathcal{R}^n$, where $n << \text{dim}$. The action $\mathbf{a}_t$, which can be stochastic, is determined by the controller π, based on the current code, and potentially the entire history of code inputs and actions, $\{\mathbf{y}_t, \mathbf{a}_t, \ldots, \mathbf{y}_0, \mathbf{a}_0\}$, represented by the internal state of the controller, $\mathbf{m_t}$. The controller defined by combining equations 2 and 3

$$\mathbf{a}_t = \pi(\mathbf{y}_t, G(\mathbf{m}_{t-1}, \mathbf{y}_{t-1}, \mathbf{a}_{t-1})),$$

can be implemented for example by a recurrent neural network, where $\mathbf{y}_t$ is the external input, and $G(\cdot)$ computes the internal activation via the networks feedback connections.

Given a compressed vector, inverse operation provides a reconstruction $\hat{\mathbf{o}}_t = R(C(\mathbf{o}_t))$. The quality of this reconstruction is measured in terms of distortion $d(\mathbf{o}_t, \hat{\mathbf{o}}_t)$. A standard distortion measure is the squared error:

$$d(\mathbf{o}_t, \hat{\mathbf{o}}_t) = \frac{1}{2}\|\mathbf{o}_t - \hat{\mathbf{o}}_t\|^2 \tag{4}$$

The overall quality of the compressor is evaluated by measuring its expected distortion,

$$D(C, R) = \sum_{\mathbf{o}} Pr(\mathbf{o}) \ d(\mathbf{o}, R(C(\mathbf{o})) \ ), \tag{5}$$

where $Pr(\mathbf{o})$ indicates the observation probabilities.

We consider adaptive compressors, for example through gradient descent. Let the compressor be parameterized by $\mathbf{W}$. Then, if $\nabla_{\mathbf{W}} D(C, R)$, can be computed or approximated, the compressor can be improved. Improving a compressor in this way can be considered unsupervised learning.

An issue with compression of this type is that it may increase the severity of *aliasing*, where states requiring different actions map to the same code, adding to the original type of aliasing due to the fact that the true operating state $\mathbf{s}_t$ cannot always be extracted from the observation $\mathbf{o}_t$. The code is guaranteed to be low dimensional but not unambiguous. However, because our system (described next) evolves RNNs, which have memory, it can mitigate the effect of possible compression induced aliasing.

III. System Architecture

Fig. 1 shows a schematic description of the general framework combining Unsupervised Learning with Evolutionary Reinforcement Learning (UL-ERL). At a high level, the architecture consists of a standard neuroevolution system where an evolutionary algorithm is used to evolve recurrent neural networks, with the key addition of an unsupervised compressor module that preprocesses the inputs to the evolving networks.

In a conventional NE system, evolution proceeds in the following steps: (1) **initialize** a random population of neural network encodings or *chromosomes*, containing candidate values for the parameters of interest (e.g. synaptic weight values, connectivity, etc.), (2) **evaluate** each chromosome after first transforming it into its corresponding network controller, and assign a fitness, (3) **reproduce** the most fit chromosomes to produce a new, hopefully better, population[1].

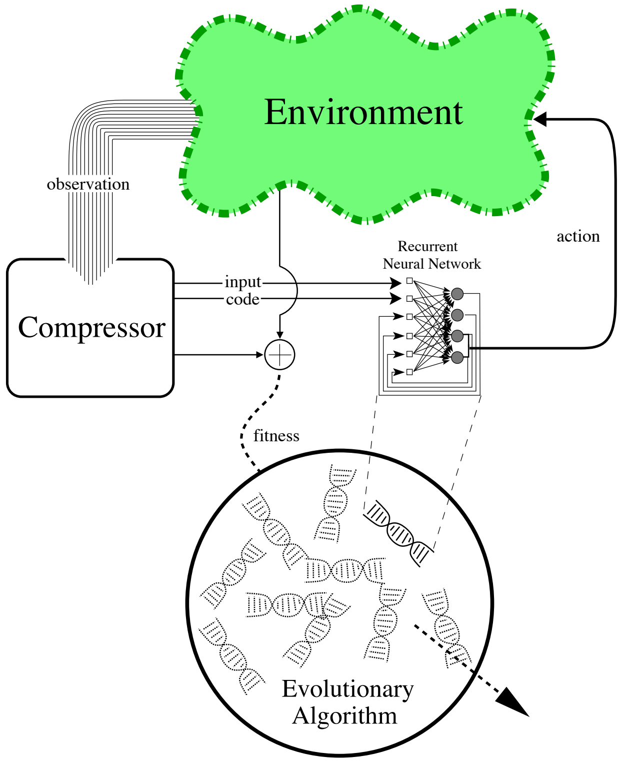

Fig. 1. **UL-ERL Architecture**. The standard neuroevolutionary RL approach is augmented by an adaptive compressor that transforms high-dimensional observations into a compact code which serves as input to the evolving recurrent neural network controllers. At each generation the compressor is trained on observations generated by the evaluated controllers, using unsupervised learning. As the compressor improves, the controllers are provided with increasingly informative features, and, in turn, they provide new data for the compressor to improve the features.

Normally, during the evaluation phase, controllers can directly perceive the observable state of the environment, which means that if the observation is high-dimensional, the networks will have to be sized accordingly. In UL-ERL (refer to pseudocode in Algorithm 1) however, each high-dimensional observation is first transformed by the compressor into a lower-dimensional code (line 8) that is then fed as input to the controller (line 13), thereby reducing the required network complexity. At the end of each generation (i.e. the loop from line 4 to 19), the compressor is trained on a set of images generated from the environment by the controllers during that generation (line 21). Training is conducted in batch, only between generations, so that all individuals in a given generation are evaluated fairly using the same sensory system (compressor).

The compressor not only provides a compact code, it also

[1]In the case of probability distribution based evolutionary algorithms, such as CMA-ES and NES, the population of candidate solutions is not explicitly maintained.

Algorithm 1: UL-ERL

```
INITIALIZE($EA$)
while $not\ solved$ do
    clear $imgs$
    for $i \leftarrow 1$ to $popsize$ do
        step $\leftarrow 1$, $f_{comp} \leftarrow$ maxerror
        while $step < maxsteps \wedge not\ solved$ do
            $o_t \leftarrow$ GETOBSERVATION($s_t$)
            ($error$, $y_t$) $\leftarrow$ COMPRESS($o_t$, $W$)
            if $error > f_{comp}$ then
                $f_{comp} \leftarrow error$
                $o_{max} \leftarrow o_t$
            end
            $a_t \leftarrow \pi(y_t)$
            $s_t \leftarrow$ ENVDYNAMICS($s_t$, $a_t$)
            $step \leftarrow step$ +1
        end
        $fit \leftarrow f_{task} \oplus f_{comp}$
        $imgs$[i] $\leftarrow o_{max}$
    end
    UPDATEEA()
    $W \leftarrow$ TRAINCOMPRESSOR($W$, $imgs$)
end
```

implements a kind of intrinsic motivation by boosting the fitness of those networks that generate action sequences resulting in novel training images. When a network is evaluated, the observation (image) with highest reconstruction error is saved (lines 9 to 12) to the training set, and the error on that image is used as an additional component, f_{comp}, that is combined with the raw fitness, f_{task} (line 17). The specific combination operator, $\oplus$, used is implementation specific.

Evolutionary search and unsupervised compressor work in synergy. Initially, the untrained compressor produces a spurious code that is of little use in solving the task. However, once the compressor is trained on the set of images from the first generation, useful features begin to emerge that capture task specific regularities, allowing controllers in the next generation to make progress. In turn, controllers that generate novel images receive higher fitness for doing so, which encourages diversity and drives evolution to discover more controllers that do the same, in order to improve the compressor further.

IV. VQ-SNES

The particular instantiation of UL-ERL in this paper uses a computationally efficient variant of Natural Evolution Strategies (NES; [5,23,24,26]), called Separable NES or SNES [17] as the evolutionary search method, and a competitive clustering algorithm based on Vector Quantization (VQ; [7]) for the compressor.

A. Separable NES

Separable NES is a restricted version of NES, that sacrifices generality in favor of low computational complexity. NES is a class of evolutionary algorithms for real-valued optimization that maintains a search distribution, instead of an explicit population, and adapts the distribution parameters by following the natural gradient of expected fitness.

In each generation the algorithm samples a population of $\lambda \in \mathbb{N}$ individuals $\mathbf{z}_i \sim \pi(\mathbf{z}|\theta)$, $i \in \{1, \ldots, \lambda\}$, i.i.d. from its search distribution, which is parameterized by θ, with the goal of maximizing a fitness function $f : \mathbb{R}^k \to \mathbb{R}$. The expected fitness under the search distribution is

$$J(\theta) = \mathbb{E}_\theta[f(x)] = \int f(x)\ \pi(x \,|\, \theta)\ dx.$$

The gradient w.r.t. the parameters can be rewritten as

$$\begin{aligned}\nabla_\theta J(\theta) =& \nabla_\theta \int f(\mathbf{z})\ \pi(\mathbf{z} \,|\, \theta)\ dz \\ =& \mathbb{E}_\theta \left[f(\mathbf{z})\ \nabla_\theta \log \pi(\mathbf{z} \,|\, \theta), \right]\end{aligned}$$

(see [26] for the full derivation) from which we obtain the Monte Carlo estimate

$$\nabla_\theta J(\theta) \approx \frac{1}{\lambda} \sum_{i=1}^{\lambda} f(\mathbf{z}_i)\ \nabla_\theta \log \pi(\mathbf{z}_i \,|\, \theta)$$

of the search gradient. The key step then consists of replacing this gradient, pointing in the direction of (locally) steepest ascent w.r.t. the given parameterization, by the natural gradient

$$\widetilde{\nabla}_\theta J = \mathbf{F}^{-1} \nabla_\theta J(\theta) \ \ .$$

where $\mathbf{F} = \mathbb{E}\left[\nabla_\theta \log \pi\left(\mathbf{z}|\theta\right) \nabla_\theta \log \pi\left(\mathbf{z}|\theta\right)^\top \right]$ is the Fisher information matrix; leading to a straightforward scheme of natural gradient ascent for iteratively updating the search distribution

$$\theta \leftarrow \theta + \eta \widetilde{\nabla}_\theta J = \theta + \eta \mathbf{F}^{-1} \nabla_\theta J(\theta) \ \ ,$$

with learning rate parameter η. The sequence of 1) sampling an offspring population, 2) computing the corresponding Monte Carlo estimate of the fitness gradient, 3) transforming it into the natural gradient, and 4) updating the search distribution, constitutes one generation of NES.

For a multivariate Gaussian search distribution, the parameters $\theta = \langle \boldsymbol{\mu}, \boldsymbol{\Sigma} \rangle$, where $\boldsymbol{\mu} \in \mathbb{R}^k$ is the mean vector and $\boldsymbol{\Sigma} \in \mathbb{R}^{k \times k}$ is the covariance matrix, are split canonically into three invariant components. This amounts to a (non-redundant) representation similar to CMA-ES [9], were split off a global step size variable from the covariance matrix in the form $\boldsymbol{\Sigma} = \sigma^2 \cdot \mathbf{B}^\top \mathbf{B}$, with $\sigma \in \mathbb{R}^+$ and $\det(\mathbf{B}) = 1$. We write $\mathbf{A} = \sigma \cdot \mathbf{B}$ for a factor of the covariance matrix, fulfilling $\boldsymbol{\Sigma} = \mathbf{A}^\top \mathbf{A}$. We obtain the corresponding gradient components

$$\begin{aligned}\nabla_{\boldsymbol{\mu}} J &= \sum_{i=1}^{\lambda} f(\mathbf{z}_i) \cdot \mathbf{p}_i \\ \nabla_{\mathbf{M}} J &= \sum_{i=1}^{\lambda} f(\mathbf{z}_i) \cdot (\mathbf{p}_i \mathbf{p}_i^\top - \mathbb{I}) \\ \nabla_\sigma J &= \operatorname{tr}(\nabla_{\mathbf{M}} J)/k \\ \nabla_{\mathbf{B}} J &= \nabla_{\mathbf{M}} J - \nabla_\sigma J \cdot \mathbb{I} \ \ ,\end{aligned}$$

where samples are picked from the standard multinormal distribution $\mathbf{p}_i \sim \mathcal{N}(0, \mathbb{I})$, before being mapped back into the

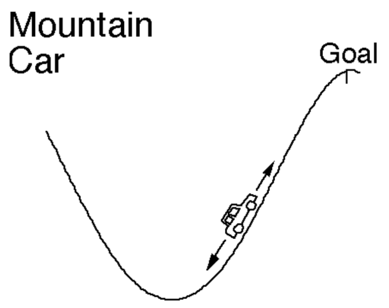

Fig. 2. **Mountain Car task**. In the standard version, the car's position and velocity are provided directly to the controller. In the much harder vision-based version, the observation is a 15 by 30 image taken from the perspective shown above. Figure courtesy of Sutton, 1996 [25]

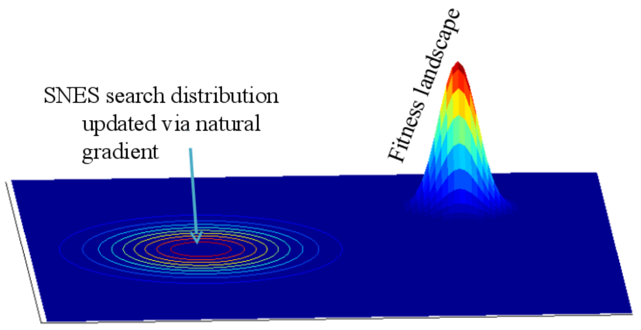

Fig. 3. **Conceptual depiction of visual mountain car fitness landscape**. The large flatland is determined by all the possible controllers that cannot solve the task. Gradient is present only among those who do, making the problem hard from an evolutionary standpoint.

original coordinate system $\mathbf{z}_i = \boldsymbol{\mu} + \sigma \mathbf{B}^\top \mathbf{p}_k$. The updates then become:

$$\begin{aligned} \boldsymbol{\mu} &\leftarrow \boldsymbol{\mu} + \eta_{\boldsymbol{\mu}} \cdot \sigma \mathbf{B} \cdot \nabla_{\boldsymbol{\mu}} J \\ \sigma &\leftarrow \sigma \cdot \exp(\eta_\sigma / 2 \cdot \nabla_\sigma J) \\ \mathbf{B} &\leftarrow \mathbf{B} \cdot \exp(\eta_{\mathbf{B}} / 2 \cdot \nabla_{\mathbf{B}} J) \ \ , \end{aligned}$$

where $\eta_{\boldsymbol{\mu}}$, η_σ, and $\eta_{\mathbf{B}}$ denote learning rates for the different components (refer to [5] for further details).

Separable NES restricts the search to distributions with diagonal covariance matrix. This means that it is limited to search distributions that are factorizable, but reduces the complexity of each generation dramatically, from $\mathcal{O}(k^3)$ to $\mathcal{O}(k)$.

B. Online Vector Quantization

Vector Quantization is a simple and stable online, lossy compression method that maps a dim-dimensional input space (e.g. observations, $\mathbf{o}$) to a *discrete* space, with only n possible values. VQ maintains n prototype column vectors $\mathbf{W} = (\mathbf{w}_1, ..., \mathbf{w}_n)$, which, in this work, represent prototypical states, so $\mathbf{w}_i \in \mathcal{S}, \forall i$. Any observation, $\mathbf{o}_t$, will be *encoded* as the index of some prototype, e.g., $j = C(\mathbf{o}_t), j = 1..n$, and any index is *decoded* as the corresponding prototypical state vector, e.g., $\mathbf{w}_j = R(j)$.

An optimal encoder maps any observation $\mathbf{o}_t$ to the index of the prototype with lowest distortion,

$$C(\mathbf{o}_t) = \operatorname*{argmin}_i d(\mathbf{o}_t, R(i)), \tag{6}$$

which will partition the state space into n distinct regions, where $\mathcal{S}_i$ is the set of observations for which prototype state vector $\mathbf{w}_i$ has the smallest squared error,

$$\mathcal{S}_i = \{\mathbf{o} | d(\mathbf{o}, \mathbf{w}_i) \le d(\mathbf{o}, \mathbf{w}_j), \forall j\}, \tag{7}$$

and i is an *internal state* representing all observations in $\mathcal{S}_i$.

The optimal decoder minimizes expected distortion, conditioned on the above encoder. The optimal $\mathbf{w}_i$ is the expectation of the observations in $\mathcal{S}_i$,

$$\mathbf{w}_i^* = \mathbb{E}_{Pr(\mathbf{o} \in \mathcal{S}_i)}[\mathbf{o}]. \tag{8}$$

Done in a batch fashion, the above two steps give the well-known K-means algorithm. Done on-line, the gradient of globally decreasing distortion $\nabla_{\mathbf{W}} D(C, R)$ (see equation 5) is approximated in an unbiased way, which in the case of the "winner-take-all" compression and squared error distortion, leads to the following update rule [10] for each prototype,

$$\begin{aligned} \mathbf{w}_i &\leftarrow \mathbf{w}_i + \alpha \ y_i \ (\mathbf{o}_t - \mathbf{w}_i) \\ &= (1 - \alpha \ y_i) \ \mathbf{w}_i + \alpha \ y_i \ \mathbf{o}_t, \end{aligned}$$

where $y_i = 1$ for $C(\mathbf{o}_t) = i$ and $y_i = 0$ otherwise. It is intuitive to see that moving $\mathbf{w}_i$ towards $\mathbf{o}_t$ (i.e. $0 < \alpha \le 1$) will decrease the distortion for $\mathbf{o}_t$.

The above is a basic "mixture of experts" system, where each prototype can be considered an expert on some part of the observation space. However, in on-line training, samples are not expected to arrive in an i.i.d. fashion, therefore stability needs to be balanced against plasticity. Prototypes that have updated over a large number of samples should be fairly stable and less likely to forget, at the same time there needs to be a subset of prototypes that can adapt quickly if new data is observed. To facilitate this, an "age", h_i, is maintained for each prototype, which is incremented after every sample it is trained upon, and now instead of a single learning rate, α, there are n prototype-specific learning rates α_{h_i} each set to $1/h_i$. Units without much experience will adapt very quickly, while experienced units will not adapt much. To prevent the learning rates from converging to zero, they are bounded from below:

$$\alpha_{h_i} = \begin{cases} 1/h_i & \text{if } 1/h_i > \omega \\ \omega & \text{otherwise} \end{cases} \tag{9}$$

Equation 9 has an intuitive interpretation: learning rate descends in a per-prototype way according to $1/h_i$ until it reaches some lower bound ω, which it uses as a constant learning rate for that prototype thereafter.

V. Experiments and Results

A. Vision-Based Mountain Car

VQ-NES was evaluated on a vision-based version of the well-known *mountain car* RL benchmark [15,21], shown in

Fig. 2. The agent controls a car within a deep valley, with the objective of reaching the highest area beyond a slope in front of the car. The car has three actions: forward, neutral, and reverse. Motor power is limited, so that simply selecting a series of consecutive "forward" actions is not sufficient to take the car to the goal. A *swinging* movement is necessary: speed has to be built up by going back and forth on the valley sides.

The state of the car is defined by its current position p_t and velocity v_t, which are within ranges $-1.2 \leq p \leq 0.5$ and $-0.07 \leq v \leq 0.07$. The car's dynamics are defined by

$$v_{t+1} = [v_t + 0.001\ a_t + g\ \cos(3p_t)] \quad (10)$$

$$p_{t+1} = [p_t + v_{t+1}], \quad (11)$$

where $a_t \in \{-1, 0, 1\}$ is the selected action, and g is the force of gravity. The car's altitude is $\cos(3p)$. If at any t, the position is ≥ 0.5 (top of the slope), the trial terminates. Otherwise, the trial stops after some maximum number of steps. The fitness of a controller is typically the number of steps needed to complete the trial.

In order to test VQ-NES, the state of the environment (p, v) was mapped to a **high-dimensional visual input** consisting of $15 \times 30 = 450$ pixels, which the controller receives instead of the state variables. In each input image, the car is seen as a colored block, around a point depending on its position, p. In its standard, non-visual form, the mountain car task is trivial for neuroevolution methods. However, in testing the UL-ERL framework, we are less concerned with the demands of the control task itself, and more with the challenge of coping with high-dimensional inputs and studying how sensory compression can be made to function in synergy with evolutionary search. The very simplicity of the underlying control task allows us to better isolate these aspects.

The task was also made more difficult in other ways. First, no velocity information is given to the controllers: the recurrent network controllers must compute the velocities internally using their feedback connections (memory). Second, whereas the initial position and velocity are typically randomized within any allowable value, here the car is always initialized the car to the deep part of the valley, from $-0.7 \leq p_0 \leq -0.3$, with slight initial velocity, within $-0.07/4 \leq v_0 \leq 0.07/4$. Finally, the force of gravity is double: instead of $g = -0.0025$, $g = -0.005$ was used. This makes reaching the top (required condition for previous RL methods) in the initial population much more difficult. As a consequence, the number of individuals receiving minimal fitness after being unable to reach the goal grows, thus enlarging the size of the relative fitness plateau (see Fig. 3).

B. Setup

To assess the advantage of combining UL with ERL and intrinsic motivation, four different setup were compared:

1) Raw Images: Networks are evolved with SNES but are fed the images directly, without first compressing them by VQ.

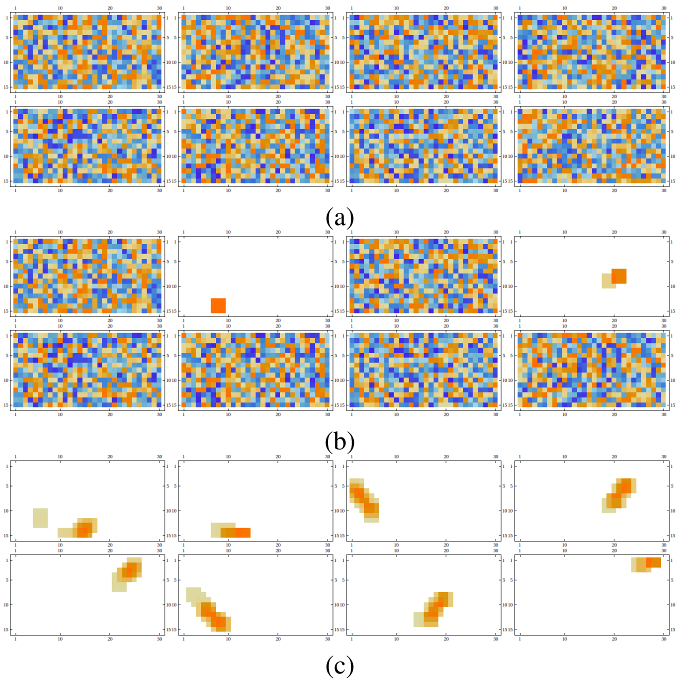
Fig. 4. **Internal state learning of the VQ-based compressor**. (a): Initial random weights, (b): After one generation, (c): Final weights. These internal states cover all possible mountain car's positions and similar positions tend to be represented as the same state.

2) Random Weight Guessing (RWG) where networks receive input from the VQ compressor, but their weights are selected at random (i.i.d) instead of being evolved. This approach provides a baseline and a rough indication of task difficulty.
3) Compression: VQ-SNES where the fitness is simply the performance on the task, f_{task}.
4) Compression+novelty: VQ-SNES where the fitness is computed based on both task performance, f_{task} and the novelty component, f_{comp}.

All controllers were implemented by single-layer fully recurrent networks, with three sigmoidal output neurons (one per action). Actions were selected stochastically after applying the softmax function to the output layer:

$$Pr(a_i) = \frac{\exp(a_i/\tau)}{\sum_{j=1}^{3} \exp(a_j/\tau)} \quad (12)$$

where τ is the temperature, which we set to 0.0001.

The VQ-based compressor used $n = 8$ clusters, initialized to 8 random 450 dimensional vectors, and the learning rate lower-bound is $\omega = 0.01$ (equation 9). This reduces the number of network weights *by a factor of* 41, from $(450\,\text{inputs} + 3\,\text{recurrent connections}) \times 3\,\text{neurons} = 1359$ to $(8 + 3) \times 3 = 33$ (and the three bias values for each of the output neurons in both cases). The compressor is trained after each generation using just a single image from each individual's run. To drive compressor improvement, the image with the highest distortion (per individual per run) is used for training.

Each of the four approaches was run 20 times for 35 generations (the outer loop in Algorithm 1). SNES was initialized

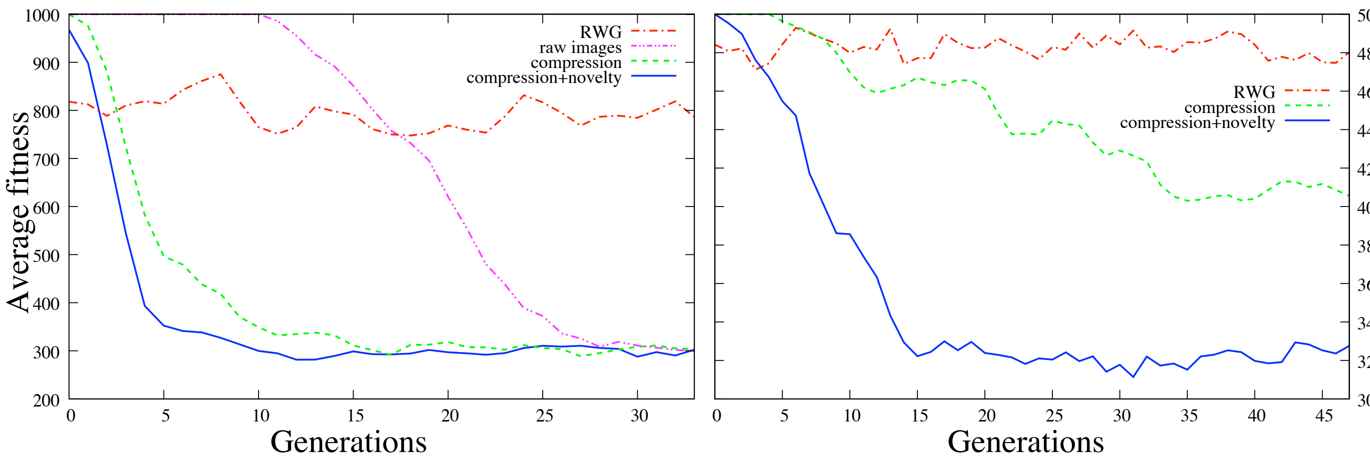

Fig. 5. **Performance of VQ-SNES on the Vision-based mountain car task**. Each curve shows number of steps required to reach the goal at generation for each of the methods (as a running average of 3 generations for smoothness). Left: basic test, with normal gravity and a lenient time limit. Compressing the mountain car image via VQ reduces the number of generations required to optimize the task significantly compared to the setup using raw images. Adding intrinsic fitness, f_{comp}, in this case, has little effect. Right: increased gravity and a short time limit. A higher percentage of failures generates a larger plateau in the fitness landscape: with less gradient information coming from task fitness, intrinsic fitness from the compressor drives the search toward networks that provide novel training images, away from the already visited positions, and eventually closer to the goal.

with a random mean vector with component chosen from $[-0.5, 0.5]$, and the variance vector was set to all ones. The population size is set to 20 and the learning rate at 0.35 for both mean and variance. These parameter settings are informed by both NES and CMA-ES defaults; we found them to be robust over small variations. A run is considered to start from a new initialization of SNES and the compressor.

The fitness for each network was $f_{task} + f_{comp}$ where f_{task} was the number of steps required to terminate the task, either by solving it, or by reaching a maximum of 1000 steps, and f_{comp}, the error of the image with highest distortion provided by that individual (before training the VQ).

C. Results

Fig. 4 shows the weight vectors in $\mathbf{W}$ as images, from a selected run of the algorithm. Fig. 5 shows the average task performance, f_{task}, for the four approaches. The RWG results show that the problem is not trivial. Even with the reduced number of weights for the methods using VQ to pre-process the input, the number of weights to guess is still high (running RWG on the uncompressed input always failed). Of course, even the compressed code is 8-dimensional compared the two state variables (position and velocity) used in the standard mountain car task. The poor performance of evolving upon "raw images" shows the advantage gained by adaptive pre-processing. Even if comparable performance is achieved after enough time, the number of weights to find is so high that in real time the execution of this test took roughly *20 times longer*.

In order to better measure the influence of f_{comp}, the task was made harder by *halving* the number of maximum steps to 500 and raising gravity further, up to $g = -0.00545$. With this setup, it is unlikely for random controllers to solve the problem, therefore SNES receives almost no gradient information with which to update its distribution. Yet, the intrinsic fitness component, f_{comp}, derived from compressor performance can provide such information and encourage exploration. While there is no guarantee that exploration will lead to increased raw fitness, in the mountain car task it eventually does. Fig. 5 shows this empirically. Typical starting generations will produce a flat fitness landscape (see Fig. 3), the absence of gradient information making evolution difficult. Now, some amount of skill is needed to even begin task-based optimization. The novelty-driven controllers are intrinsically motivated to explore novel places, since the gradient will be in directions that lead to observations that the compressor does not deal with well [18], thus finding the task-based goal much more quickly.

VI. Conclusion and Future Directions

We combined adaptive compression and RNN controllers trained by stochastic search. A compressor learns to extract regularities within high-dimensional observations and automatically provides simplified input codes to an evolving recurrent actor with memory. We tested a combination of adaptive vector quantization (learning a set of compact internal representations) and Separable Natural Evolution Strategies, an efficient stochastic search method, applying it to a difficult, high-dimensional, vision-based version of the mountain car task. Results showed that VQ is able to automatically simplify the observations in a way that permits SNES to evolve successful controllers. An additional *intrinsic motivation* fitness term (based on the maximum distortion of the observations provided to the compressor) was shown to drive *curious exploration* in a useful way, providing the compressor with novel observations that make it improve.

Since VQ will fail in presence of changing backgrounds, future work will try more sophisticated compressors such as deep neural nets. We will also utilize feedback from the policy evolver to inform the compressor, thus developing the latter's features not only from unsupervised criteria, but also such that they are useful for the task at hand. A third future direction is to have the compressor's contribution to fitness be based on *compression progress* [19,20] instead of reconstruction error [18] as was done in this paper. A problem with reconstruction error is that intrinsic reward may be easy to achieve in noisy environments, yielding individuals that choose simply to go to noisy places instead of uncovering learnable regularities.

VII. Acknowledgments

This work was funded through the 7th framework program of the EU under grant numbers 228844 (NanoBioTouch project), and 231722 (IM-Clever project), and by Swiss National Science Foundation grants 120061 (EVOCMP) and CRSIKO-122697 (Sinergia project).

References

[1] F. Fernández and D. Borrajo. Two steps reinforcement learning. *International Journal of Intelligent Systems*, 23(2):213–245, 2008.

[2] D. Floreano, P. Dürr, and C. Mattiussi. Neuroevolution: from architectures to learning. *Evolutionary Intelligence*, 1(1):46–62, 2008.

[3] J. Gauci and K. Stanley. Generating large-scale neural networks through discovering geometric regularities. In *Proceedings of the Conference on Genetic and Evolutionary Computation*, pages 997–1004, New York, NY, USA, 2007. ACM.

[4] L. Gisslén, M. Luciw, V. Graziano, and J. Schmidhuber. Sequential Constant Size Compressors and Reinforcement Learning. In *Proceedings of the Fourth Conference on Artificial General Intelligence*, 2011.

[5] T. Glasmachers, T. Schaul, Y. Sun, D. Wierstra, and J. Schmidhuber. Exponential Natural Evolution Strategies. In *Genetic and Evolutionary Computation Conference (GECCO)*, Portland, OR, 2010.

[6] F. J. Gomez. *Robust Nonlinear Control through Neuroevolution*. PhD thesis, Department of Computer Sciences, University of Texas at Austin, 2003.

[7] R. Gray. Vector quantization. *IEEE Assp Magazine*, 1(2):4–29, 1984.

[8] F. Gruau. *Neural Network Synthesis using Cellular Encoding and the Genetic Algorithm.* PhD thesis, l'Universite Claude Bernard-Lyon 1, France, 1994.

[9] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–195, 2001.

[10] T. Heskes. Energy functions for self-organizing maps. *Kohonen maps*, pages 303–316, 1999.

[11] S. R. Jodogne and J. H. Piater. Closed-loop learning of visual control policies. *Journal of Artificial Intelligence Research*, 28:349–391, 2007.

[12] J. Koutnik, F. Gomez, and J. Schmidhuber. Evolving neural networks in compressed weight space. In *Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO-10)*, 2010.

[13] S. Lange and M. Riedmiller. Deep auto-encoder neural networks in reinforcement learning. In *International Joint Conference on Neural Networks (IJCNN 2010), Barcelona, Spain*. Citeseer, 2010.

[14] R. Legenstein, N. Wilbert, and L. Wiskott. Reinforcement Learning on Slow Features of High-Dimensional Input Streams. *PLoS Computational Biology*, 6(8), 2010.

[15] A. Moore. Knowledge of knowledge and intelligent experimentation for learning control. In *Neural Networks, 1991., IJCNN-91-Seattle International Joint Conference on*, volume 2, pages 683–688. IEEE, 1991.

[16] D. Pierce and B. Kuipers. Map learning with uninterpreted sensors and effectors. *Artificial Intelligence*, 92:169–229, 1997.

[17] T. Schaul, T. Glasmachers, and J. Schmidhuber. High dimensions and heavy tails for natural evolution strategies. In *Genetic and Evolutionary Computation Conference (GECCO)*, 2011.

[18] J. Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In J. A. Meyer and S. W. Wilson, editors, *Proc. of the International Conference on Simulation of Adaptive Behavior: From Animals to Animats*, pages 222–227. MIT Press/Bradford Books, 1991.

[19] J. Schmidhuber. Developmental robotics, optimal artificial curiosity, creativity, music, and the fine arts. *Connection Science*, 18(2):173–187, 2006.

[20] J. Schmidhuber. Formal Theory of Creativity, Fun, and Intrinsic Motivation (1990–2010). *Autonomous Mental Development, IEEE Transactions on*, 2(3):230–247, 2010.

[21] S. Singh and R. Sutton. Reinforcement learning with replacing eligibility traces. *Recent Advances in Reinforcement Learning*, pages 123–158, 1996.

[22] K. O. Stanley. *Efficient Evolution of Neural Networks Through Complexification*. PhD thesis, Department of Computer Sciences, University of Texas at Austin, August 2004. Technical Report AI-TR-04-314.

[23] Y. Sun, D. Wierstra, T. Schaul, and J. Schmidhuber. Efficient Natural Evolution Strategies. In *Genetic and Evolutionary Computation Conference (GECCO)*, 2009.

[24] Y. Sun, D. Wierstra, T. Schaul, and J. Schmidhuber. Stochastic Search using the Natural Gradient. In *International Conference on Machine Learning (ICML)*, number 1, 2009.

[25] R. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. *Advances in neural information processing systems*, pages 1038–1044, 1996.

[26] D. Wierstra, T. Schaul, J. Peters, and J. Schmidhuber. Natural Evolution Strategies. In *Proceedings of the Congress on Evolutionary Computation (CEC08), Hongkong*. IEEE Press, 2008.

[27] X. Yao. A review of evolutionary artificial neural networks. *International Journal of Intelligent Systems*, 4:203–222, 1993.