UNSUPERVISED LEARNING OF GOAL SPACES FOR INTRINSICALLY MOTIVATED EXPLORATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Intrinsically motivated goal exploration algorithms enable machines to explore and discover a diversity of policies in large and complex environments. These exploration algorithms have been shown to allow real world robots to acquire skills such as tool use in high-dimensional continuous action and state spaces. However, they have so far assumed that self-generated goals are sampled in a specifically engineered space. In this work, we propose to use deep representation learning algorithms to learn a goal space, leveraging observations of world changes produced by another agent. We present experiments with a simulated robot arm interacting with an object, and we study how the performances of exploration algorithms on such learned representations relate with their performances on engineered representations. We also uncover a link between the exploration performances and the quality of the learned representation regarding the underlying state space.

1 INTRODUCTION

Spontaneous exploration plays a key role in the development of knowledge and skills in human children. For example, young children spend a large amount of time exploring what they can do with their body and external objects, independently of external objectives such as finding food or following instructions from adults. Such intrinsically motivated exploration (Berlyne, 1966; Gopnik et al., 1999) leads them to make ratcheting discoveries, such as learning to locomote or climb in various styles and on various surfaces, or learning to stack and use objects as tools. Equipping machines with similar mechanisms of intrinsically motivated exploration should also be an essential dimension for lifelong open-ended learning and artificial intelligence.

In the last two decades, several families of computational models have both contributed to a better understanding of such exploration processes in infants, and how to apply them efficiently for autonomous lifelong machine learning (Oudeyer et al., 2016). One general approach taken by several research groups (Baldassarre et al., 2013; Oudeyer et al., 2007; Barto, 2013; Friston et al., 2017) has been to model the child as intrinsically motivated to make sense of the world, exploring like a scientist that imagines, selects and runs experiments to gain knowledge and control over the world. These models have focused in particular on three kinds of mechanisms argued to be essential and complementary to enable machines and animals to efficiently explore and discover skill repertoires in the real world (Oudeyer et al., 2013; Cangelosi et al., 2015): embodiment [1], intrinsic motivation[2] and social guidance[3]. This article focuses on challenges related to learning goal representations for intrinsically motivated exploration, but also leverages models of embodiment, through the use of parameterized Dynamic Movement Primitives controllers (Ijspeert et al., 2013) and social guidance, through the use of observations of another agent.

Given an embodiment, intrinsically motivated exploration[4] consists in automatically and spontaneously conducting experiments with the body to discover both the world dynamics and how it can be controlled through actions. Computational models have framed intrinsic motivation as a family of mechanisms that self-organize agents exploration curriculum, in particular through generating

[1] Body synergies provide structure on action and perception
[2] Self-organizes a curriculum of exploration at multiple levels of abstraction
[3] Leverages what others already know
[4] Also called curiosity-driven exploration

and selecting experiments that maximize measures such as novelty (Andreae & Andreae, 1978; Sutton, 1990), predictive information gain (Little & Sommer, 2013), learning progress (Schmidhuber, 1991; Kaplan & Oudeyer, 2003), competence progress (Baranes & Oudeyer, 2013) or empowerment (Salge et al., 2014). When used in the Reinforcement Learning (RL) framework (e.g. (Sutton, 1990; Schmidhuber, 1991; Kaplan & Oudeyer, 2003; Barto, 2013)), these measures have been called intrinsic rewards, and they are often applied to reward the "interestingness" of actions or states that are explored. They have been consistently shown to enable artificial agents or robots to make discoveries and solve problems that would have been difficult to learn if one would have used a classical optimization or RL approach considering only the target reward (which is often rare or deceptive) (Chentanez et al., 2005; Baranes & Oudeyer, 2013; Stanley & Lehman, 2015). Recently, they have been similarly used to guide exploration in difficult Deep Reinforcement Learning problems with rare rewards, e.g. (Bellemare et al., 2016; Houthooft et al., 2016; Tang et al., 2017; Pathak et al., 2017).

However, many of these computational approaches have considered intrinsically motivated exploration at the level of micro-actions and states (e.g. considering low-level actions and pixel level perception). Yet, children's exploration leverages abstractions of the environments, such as objects and qualitative properties of the way they may move or sound, and explore by setting self-generated goals (Von Hofsten, 2004), ranging from objects to be reached, toy towers to be built, or paper planes to be flown. A computational framework proposed to address this higher-level form of exploration has been Intrinsically Motivated Goal Exploration Processes (IMGEPs) (Baranes & Oudeyer, 2009; Forestier et al., 2017), which is closely related to the idea of goal babbling (Rolf et al., 2010). Within this approach, agents are equipped with a mechanism enabling them to sample a goal in a space of parameterized goals[5], before they try to reach it by executing an experiment. Each time they sample a goal, they dedicate a certain budget of experiments time to improve the solution to reach this goal, using lower-level optimization or RL methods for example. Most importantly, in the same time, they take advantage of information gathered during this exploration to improve solutions to other goals[6]. While this property of cross-goal learning often enables efficient exploration even if goals are sampled randomly (Baranes & Oudeyer, 2013), more powerful goal sampling strategy exists. A particular one consists in using meta-learning algorithms to monitor the evolution of competences over the space of goals and to select the next goal to try, according to the expected competence progress resulting from practicing it (Baranes & Oudeyer, 2013). This enables to automate curriculum sequences of goals of progressively increasing complexity, which has been shown to allow high-dimensional real world robots to acquire efficiently repertoires of locomotion skills or soft object manipulation Baranes & Oudeyer (2013), or advanced forms of nested tool use (Forestier et al., 2017). Similar ideas have been recently applied in the context of multi-goal deep reinforcement learning, where architectures closely related to intrinsically motivated goal exploration are used by procedurally generating goals and sampling them randomly (Cabi et al., 2017; Najnin & Banerjee, 2017) or adaptively (Florensa et al., 2017).

Yet, a current limit of existing algorithms within the family of Intrinsically Motivated Goal Exploration Processes is that they have assumed that the designer[7] provides a representation allowing the autonomous agent to generate goals, together with formal tools used to measure the achievement of these goals (e.g. cost functions). For example, the designer could provide a representation that enables the agent to imagine goals as potential continuous target trajectories of objects (Forestier et al., 2017), or reach an end-state starting from various initial states defined in euclidean space (Florensa et al., 2017), or realize one of several discrete relative configurations of objects (Cabi et al., 2017), which are high-level abstractions from the pixels. While this has allowed to show the power of intrinsically motivated goal exploration architectures, designing IMGEPs that sample goals from a learned goal representation remains an open question. There are several difficulties. One concerns the question of how an agent can learn in an unsupervised manner a representation for hypothetical goals that are relevant to their world before knowing whether and how it is possible to achieve them with the agent's own action system. Another challenge is how to sample "interesting" goals using a learned goal representation, in order to remain in regions of the learned goal parameters that are not too exotic from the underlying physical possibilities of the world. Finally, a third challenge consists

[5]Here a goal is not necessarily an end state to be reached, but can characterize certain parameterized properties of changes of the world, such as following a parameterized trajectory

[6]E.g. while learning how to move an object to the right, they may discover how to move it to the left

[7]Here we consider the human designer that crafts the autonomous agent system.

in understanding which properties of unsupervised representation learning method enable an efficient use within an IMGEP architecture so as to lead to efficient discovery of controllable effects in the environment.

In this paper, we present one possible approach where aspects of these difficulties are approached by leveraging the possibility to observe changes made to the word by another agent[8]. These observations are used for training an unsupervised learning algorithm to build a lower-dimensional representation that can be used afterwards for sampling goals in intrinsically motivated goal exploration algorithms. Here, we do not assume that the learner actually knows that these observed world changes are caused by another agent, and we do not assume it can perceive or infer the action program of the other agent [9]. We experimented with different unsupervised representation learning algorithms, such as AE Bourlard & Kamp (1988), VAE Rezende et al. (2014)Kingma & Ba (2015), VAE with Normalizing Flow Rezende & Mohamed (2015), Isomap Tenenbaum et al. (2000) or PCA Bishop (2006); and we compared their performances under different metrics. In particular, we measured the diversity of environment configurations visited during the exploration using those representations as goal space. We also evaluated the average skill performance to produce various environment configurations translated as goals by the representation algorithm. We also study which properties of the learned embedding are correlated with the acquired average skill levels. Experiments presented in this article focus on two variants of a simulated experimental setup where a simulated multi-joint arm robot has to discover how to move a simulated object in various places and with various orientations, similar to the setup used in Forestier & Oudeyer (2016).

2 Goals Representation learning for Exploration Algorithms

2.1 Exploration Algorithms in Developmental Robotics

Intrinsically Motivated Goal Exploration Processes (IMGEP), are powerful algorithmic architectures which were initially introduced in Baranes & Oudeyer (2009) and formalized in Forestier et al. (2017), as heuristics to drive the exploration of large action spaces to learn forward and inverse control models in difficult robotic problems. To clearly understand the essence of IMGEPs, we must envision the robotic agent as an experimenter seeking information about an unknown physical phenomenon, by the mean of sequential experiments. In this perspective, the main elements of an exploration are:

- A *context* c, element of a Context Space $\mathcal{C}$. This context represents the initial experimental factors that are not under the robotic agent control. In most cases, the context is considered fully observable.
- A *parameterization* θ, element of a Parameterization Space Θ. This parameterization represents the experimental factors that can be controlled by the robotic agent.
- An *outcome* o, element of an outcome space $\mathcal{O}$. The outcome contains information qualifying the execution of the experiment regarding interests of the agent.
- A *phenomenon dynamics* $D : \mathcal{C}, \Theta \mapsto \mathcal{O}$, which in most interesting cases is unknown.

If we take the example of the *Arm-Ball* problem[10] in which a multi-joint robotic arm can interact with a ball, the context could be the initial state of the robot and the ball, the parameterization could be a set of motor torque commands for N time steps, and the outcome could be the position of the ball at the last time step. Developmental roboticists are interested in developing autonomous agents, that learn two models, the forward model $\tilde{D} : \mathcal{C} \times \Theta \mapsto \mathcal{O}$ which approximates the phenomenon dynamics, and the inverse model $\tilde{I} : \mathcal{C} \times \mathcal{O} \mapsto \Theta$ which allows to produce desired outcomes under given context by executing the proper parameterization. Using the aforementioned elements one could imagine a simple strategy that would allow the agent to gather tuples $\{c, \theta, o\}$ to train those models, by uniformly sampling a random parameterization $\theta \sim \mathcal{U}(\theta)$ and executing the experiment,

[8]In our case, this other agent will be the designer of the experiment.

[9]Other works have considered how stronger forms of social guidance, such as imitation learning Schaal et al. (2003), could accelerate intrinsically motivated goal exploration Nguyen & Oudeyer (2014), but they did not consider the challenge of learning goal representations.

[10]See section3 for details

a strategy which we will refer to as *Random Parameterization Exploration*. The problem for most interesting applications in robotics, is that only a small subspace of Θ is likely to produce interesting outcomes. Indeed, considering again the Arm-Ball problem with time-bounded action sequences as parameterizations, very few of those will lead the arm to touch the object and dragging it. In this case, a random sampling in Θ would be a terrible strategy to yield interesting samples allowing to learn useful forward and inverse models. To overcome this difficulty, one must come up with a better approach to sample parameterizations that lead to informative samples.

Intrinsically Motivated Goal Exploration Strategies propose a way to address this issue by giving the agent a set of tools to handle this situation:

- A *Goal Space* $\mathcal{T}$ whose elements τ represent goals that can be targeted by the autonomous agent.
- A *Goal Policy* $\gamma(\tau)$, which is a probability distribution over the Goal Space. It can be stationary, but in most cases, it will be updated each time following the intrinsic motivation strategy employed. Note that in some cases, this Goal Policy will be a conditional on the context $\gamma(\tau|c)$.
- A set of *Goal-parameterized Cost Functions* $C_\tau : \mathcal{O} \mapsto \mathbb{R}$ defined over all $\mathcal{O}$, which maps every outcome with a real number representing the goodness-of-fit of the outcome o regarding the goal τ.
- A *Meta-Policy* $\Pi : \mathcal{T}, \mathcal{C} \mapsto \Theta$ which is a mechanism to approximately solve the minimization problem $\Pi(\tau, c) = \arg\min_\theta C_\tau(\tilde{D}(\theta, c))$, where $\tilde{D}$ is a running forward model, trained on-line during exploration.

In some applications, a *de-facto* ensemble of such tools can be used. For example, if $\mathcal{O}$ is a Euclidean space, we can allow the agent to set goals in the Outcome Space $\mathcal{T} = \mathcal{O}$, in which case for every goal τ we have a Goal-parameterized cost function $C_\tau(o) = \|\tau - o\|$. In the case of the Arm-Ball problem, the final position of the ball is used as Outcome Space, and hence the Euclidean distance between the goal position and the final ball position at the end of the episode can be used as Goal-parameterized cost function. Moreover, we can set the goal policy to follow an isotropic Gaussian prior $\gamma = \mathcal{N}(0, I)$, and the Meta-Policy mechanism to return the parameterization $\theta_{min} = \arg\min_{\theta \in \mathcal{H}} C_\tau(\tilde{D}(\theta, c))$ where $\mathcal{H}$ is the history of executed parameterizations.

Using those tools, the autonomous agent can sample a parameterization for the next experiment, by first sampling a goal τ using the Goal Policy γ, and second, executing the meta-policy Π to search the parameterization that will minimize the Goal-parameterized cost function C_τ. This two steps procedure has multiple interests. First it accounts for the actual state of knowledge of the forward model, and second it allows to incorporate an intrinsic motivation mechanism to drive the exploration in the long term. Intrinsic Motivation mechanisms can be seen as sequential update rules applied to the goal policy to optimize a given metric[11] such as the expected learning progress, novelty or information gain. This procedure is the basis of the IMGEP architecture, which is represented in Algorithm 2. This procedure has been experimentally shown to bring an effective exploration in robotic setups involving a complicated physical phenomenon such as object manipulations using tools (Forestier & Oudeyer, 2016; Forestier et al., 2017) or soft deformable objects Nguyen & Oudeyer (2014).

As we saw, defining goals and the subsequent Goal-parameterized cost function can be made simple by sampling goals in the Outcome Space. Nevertheless, an issue arises when it comes to using these algorithms in real-life setups, and within a fully autonomous learning approach. First, there are many real world cases where providing an Outcome Space to the agent is is difficult, since the designer may not himself understand well the space that the robot is learning about. The approach taken until now (Forestier et al., 2017), was to create an external program which extracted information out of images, such as tracking all objects positions. This information was presented to the agent as a point in $[0, 1]^n$, which was hence considered as an Outcome Space. In such complex environments, the designer may not know what is actually feasible or not for the robot, and the Outcome space may contain many infeasible goals. This is the reason why advanced mechanisms for sampling goals and discovering which ones are actually feasible have been designed Baranes & Oudeyer (2013); Forestier et al. (2017). Second, a system where the engineer designs the representation of a goal

[11] In particular, this can be seen as solving a contextual continuous non stationary multi-armed bandit problem, using for e.g. the techniques presented in Baranes & Oudeyer (2013)

space is limited in its autonomy. A question arising from this is: can we design a mechanism that allow the agent to construct a well-behaved Outcome Space by the mean of examples? Representation Learning methods, in particular Deep Learning algorithms, constitute a natural approach to this problem as it has shown outstanding performances in learning representations for images. In the next part we present the main Deep Representation Learning algorithms tested; the Auto-Encoders, along with their more recent Variational counter-part.

2.2 DEEP REPRESENTATION LEARNING

We experimented over different Deep Representation Learning algorithms which are rapidly mentioned here. For a more in-depth introduction to those models, the reader can refer to the Appendix B which contains details on the derivations of the different Cost Functions and Architectures.

Auto-Encoders (AEs) are a particular type of Feed-Forward Neural Networks that were introduced in the early hours of neural networks Bourlard & Kamp (1988). They are trained to output a reconstruction $\tilde{\mathbf{x}}$ of the input vector $\mathbf{x}$ of dimension D, through a representation layer of size $d < D$. They can be trained in an unsupervised manner using a large dataset of unlabeled samples $\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i \in \{0 \dots N\}}$. Their main interest lies in their ability to model the statistical similarities existing in the data. Indeed, during training, the network learns the regularities allowing to encode most of the information existing in the input in a more compact representation. Put differently, AEs can be seen as learning a non-linear compression for data coming from an unknown distribution. Those models can be trained using different algorithms, the most simple one being Stochastic Gradient Descent, to minimize a loss function $\mathcal{J}(\mathcal{D})$ that penalizes $\tilde{\mathbf{x}}$ to be different from $\mathbf{x}$ for all samples in $\mathcal{D}$.

Variational Auto-Encoders (VAEs) are a recent alternative to classic AEs (Rezende et al., 2014; Kingma & Ba, 2015), that can be seen as an extension to a stochastic encoding. The argument underlying this model is slightly more involved than the simple approach taken for AEs, and relies on a statistical standpoint presented in AppendixB. In practice, this model simplifies to an architecture very similar to an AE, differing only in the fact that the encoder f_θ outputs the parameters μ and σ of a multivariate Gaussian distribution $\mathcal{N}(\mu, diag(\sigma^2))$ with diagonal covariance matrix, from which the representation $\mathbf{z}$ will be sampled. Moreover, an extra term will be added to the Cost Function, to condition the distribution of $\mathbf{z}$ in the representation space. Under this restriction of factorial Gaussianity, the neural network can be made fully differentiable thanks to a *reparameterization trick*, allowing a training using Stochastic Gradient Descent.

In practice VAEs tends to yield smooth representations of the data, and are faster to converge than AEs from our experiments. Despite these interesting properties, we emphasize that the derivation of the actual cost function relies mostly on the assumption that the factors can be described by a factorial Gaussian distribution. This hypothesis can be largely erroneous, for example if one of the factor is periodic, multi-modal, or discrete. In practice our experiments showed that even if the training could converge for non Gaussian factors, it tends to be slower and to yield badly conditioned representations.

Normalizing Flow proposes a way to overcome this problem, by allowing more expressive distributions (Rezende & Mohamed, 2015). It uses the classic rule of change of variables for random variables, which states that considering a random variable $\mathbf{z}_0 \sim q(\mathbf{z}_0)$, and an invertible transformation $t : \mathbb{R}^d \mapsto \mathbb{R}^d$, if $\mathbf{z} = t(\mathbf{z}_0)$ then $q(\mathbf{z}) = q(\mathbf{z}_0)|\det \partial t / \partial \mathbf{z}_0|^{-1}$. Using this, we can chain multiple transformations $t_1, t_2, \dots, t_K$ to produce a new random variable $\mathbf{z}_K = t_K \circ \dots \circ t_2 \circ t_1(\mathbf{z}_0)$. One particularly interesting transformation is the *Radial Flow*, which allows to radially contract and expand a distribution as can be seen in Figure 7 in Appendix. As we can see, this transformation seems to give the required flexibility to encode periodic factors.

2.3 GOALS REPRESENTATION LEARNING FOR IMGEP

We can now introduce the augmented IMGEP architecture we used during this study. If we consider the representation function resulting from the algorithm training $\tilde{R} : \mathcal{X} \mapsto \mathcal{O}$ as learning a mapping from a space of Raw Signals $\mathcal{X}$ to an Outcome Space $\mathcal{O}$, we can rewrite the architecture as:

Algorithmic Architecture 1: Intrinsically Motivated Goal Exploration Strategy with Unsupervised Goal Space learning

Input:
Regressor $\tilde{D}_{running}$, Regressor $\tilde{D}$, Regressor $\tilde{I}$, Goal Policy γ, Minimizer Π, Representation function $\tilde{R}$

begin
  for *A fixed number of Observation iterations* n_r **do**
    Observe the phenomenon with raw sensors to gather a sample x_i
    Add this sample to a sample database $\mathcal{D} = \{x_i\}_{i \in [0, n_r]}$
  Learn the Representation function $\tilde{R}$ using the chosen representation learning algorithm trained with $\mathcal{D}$
  for *A fixed number of Bootstrapping iterations* **do**
    Observe context c
    Sample $\theta \sim \mathcal{U}(\theta)$
    Perform experiment and retrieve outcome with raw sensor signal $o = \tilde{R}(x)$
    Update Regressor $\tilde{D}_{running}$ with tuple $\{c, \theta, o\}$
  for *A fixed number of Exploration iterations* **do**
    Observe context c
    Sample a goal $\tau \sim \gamma$
    Compute $\theta = \arg\min_\theta C_\tau(\tilde{D}_{running}(\theta, c))$ using an optimization algorithm
    Perform experiment and retrieve outcome from raw sensor signal $o = \tilde{R}(x)$
    Update Regressor $\tilde{D}_{running}$ with a tuple $\{c, \theta, o\}$
    Update Goal Policy γ, according to Intrinsic Motivation strategy
  while *exploration running,* ***asynchronously*** **do**
    Update regressors $\tilde{D}$ and $\tilde{I}$ with acquired samples $\{c, \theta, o\}_i$
return *The forward model* $\tilde{D}$*, the inverse model* $\tilde{I}$ *and the representation* $\tilde{R}$*.*

We will now present the experimental evaluation and the results of this novel architecture.

3 EXPERIMENTAL SETUP

To assess the performance of the novel exploration architecture, we did various experiments on two simulated environments. We will now present in depth the experimental campaign we performed. The anonymous code to reproduce the experiments is available at `https://www.dropbox.com/s/zf4cmd2anzldbz5/UnsupervisedGoalSpaceLearning.tar.gz?dl=0`.

Environments: We experimented on two different **Simulated Environments** derived from the classic Arm-Ball benchmark represented in Figure 1, namely the *Arm-Ball* and the *Arm-Arrow* environments, in which a multiple joint arm evolves in an environment containing an object it can handle and move around in the scene. The environments are presented in depth in Appendix C.

Algorithmic Instantiation of the Architecture: As mentioned in Section 2.2, we experimented over different types of Deep Representation Learning Algorithms: *Auto-Encoders*, *Variational Auto-Encoders* and *Radial Flow Variational Auto-Encoders*[12]. In order to compare those to classic algorithms, we also experimented with the classic manifold learning algorithm *Isomap* (RGE-ISOMAP) (Tenenbaum et al., 2000) along with the classic *Principal Component Analysis* (RGE-PCA).

Regarding the general approach presented earlier, we considered the following elements:

[12]Details of the architectures are presented in appendix.

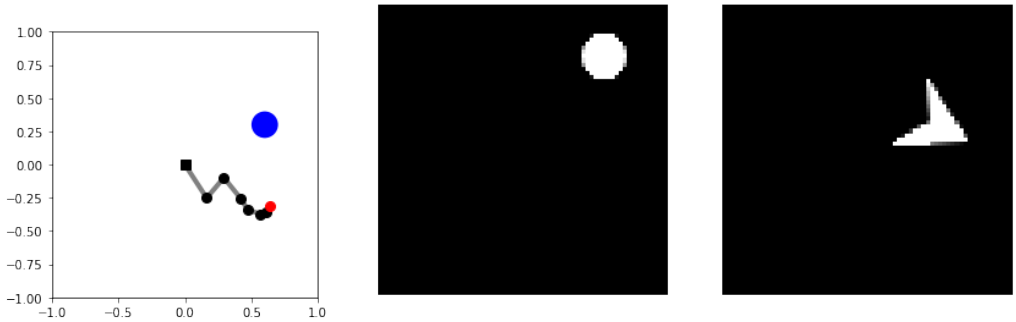

Figure 1: A drawing of the Arm-Ball environment problem used for designer understanding purpose (on the left), and the rendered images used as raw signals representing the end position of the objects for Arm-Ball (on the center) and Arm-Arrow (on the right) environments. Note that we considered the arm to be outside of the camera field.

- **Context Space $\mathcal{C} = \varnothing$:** In the implemented environments, the initial positions of the arm and the object were reset at each episode[13]. Consequently, the context was not observed nor accounted for by the agent.
- **Parameterization Space $\Theta = [0,1]^{21}$:** During the experiments, we used a Dynamic Movement Primitive controller to generate time-bounded motor actions sequences. Since the DMP controller was parameterized by 3 basis function for each joint of the arm (7), the parameterization of the controller was represented by a point in $[0,1]^{3\times 7}$.
- **Outcome Space $\mathcal{O} \subset \mathbb{R}^{10}$:** The Outcome Space is the subspace of $\mathbb{R}^l$ spanned by the representations of the ensemble of images that observed in the first phase of learning (see Annex C). In our case, we constrained the Representation Learning algorithm to learn a representation of size $l = 10$[14].
- **Goal Space $\mathcal{T} = \mathcal{O}$:** The Goal Space was taken to equate the Outcome Space, as was presented earlier.
- **Goal-Parameterized Cost function $C_\tau(\cdot) = \|\tau - \cdot\|_2$:** Sampling goals in the Outcome Space allows us to use the Euclidean distance as Goal-parameterized cost function.

Considering those elements, we used the instantiation of the IMGEP architecture represented in Appendix in Algorithm 3. We implemented a simple goal sampling strategy known as *Random Goal Exploration* (RGE), which consists, given a stationary distribution over the Outcome Space $p(o)$, to sample a random goal $o \sim p(o)$ each time. Since the Outcome Space was learned by the agent, it had no prior knowledge of $p(o)$. We used a Gaussian kernel density estimation (Parzen, 1962; Rosenblatt, 1956) to estimate this distribution from the projection of the images observed by the agent in the observation phase into the learned goal space representation. This allowed to sample points in the neighborhood of situations observed by the agent during the observation phase. We used a simple k-neighbors regressor to implement the running forward model $\tilde{D}$, and the Meta-Policy mechanism considered was simply returning the nearest achieved outcome in the outcome space, and taking the same parameterization perturbed by an exploration noise.

Quality and Performance Measures: One of the fundamental point we wanted to assert, was how the quality of the representation geometry was affecting the performances, regarding the geometry of the underlying environment state space. To measure the quality of the different learned representations, we gathered inspiration from the manifold learning literature. After the agent learned the representation (at line 5 in the Algorithm 3), we performed different measures using a set of tuples $\{s_i, o_i\}$ where s_i is a point of the underlying environment state-space $s_i \in \mathbb{R}^n$ and $o_i = \tilde{R}(f(s_i))$ is the projection in the latent space of the image generated by the simulation from state s_i. With those

[13]This makes the experiment faster but does not affect the conclusion of the results

[14]This number was chosen arbitrarily, but is superior to the dimensionality of the parameterization of the underlying phenomenons, which the algorithm is not supposed to know initially.

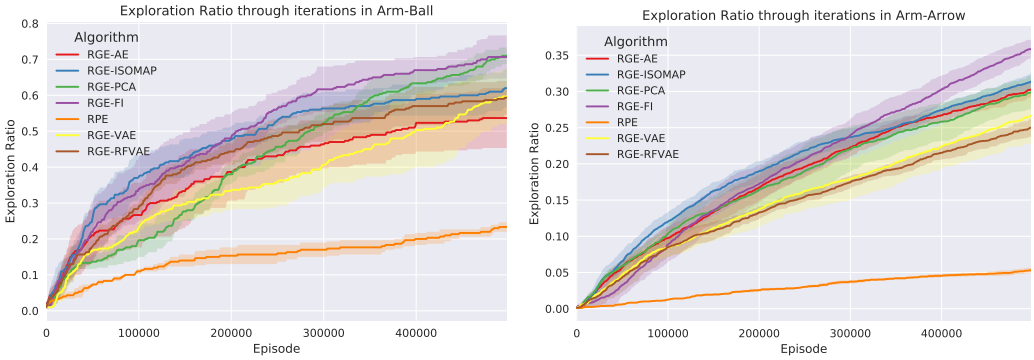

Figure 2: Exploration Ratio evolution through episodes for the different architectures considered. RGE $-\star$ refers to the Random Goal Exploration architecture used along with the $\star$ representation. RPE refers to the Random Parameterization Exploration. The shaded surfaces represent the minimal and maximal performances achieved over 3 runs.

tuples, we assessed both the local and global quality of the different representations using two different measure; the *Quality* $Q_{T\&C}$ and the *Kruskal Stress* S_{Kr}, which are presented in much details in Appendix F.

We also monitored the performances of the exploration algorithm during the exploration using two measures inspired from the Developmental Robotics literature; the *Exploration Ratio* $Ratio_{explo}$ and the *Mean Squared Error* of the meta-policy MSE. The methods used to compute those measures are also detailed in the Appendix F.

Comparisons with other approaches: In order to compare our results with known algorithms, we also did experiments using two other algorithms:

- **Full Information (FI):** We compared our results with the performances achieved with an algorithm using directly the state-space underlying the environment as Outcome Space [15], and has knowledge of $p(o) = \mathcal{U}(\mathcal{O})$, in which case, the autonomous agent does not need to learn a representation. We expect this algorithm to upper bound the performances of our novel architecture.
- **Random Parameterization Exploration (RPE):** We also compared our results with the Random Parameterization Exploration approach, which consists in using no Outcome Space, nor Goal Policy, and only sample a random parameterization $\theta \sim \mathcal{U}(\Theta)$ at each episode. We expect this algorithm to lower bound the performances of our novel architecture.

4 Results

Exploration Performances: In the Figure 2 we can see the evolution of the Exploration Ratio through experiment episodes. First we can see that following our expectations, for both environments the ER of the different architectures using a learned Outcome Space is always superior to the ER achieved by the RPE architecture. Moreover, the performances achieved by the RGE-FI architecture on the environments is superior or equal to the performances given by the novel architectures. Regarding the different representation learning algorithms, on the Arm-Ball environment, all algorithms attain a reasonable final ER, with a particularly good performances of the RGE-ISOMAP architecture which achieves almost as good performance as RGE-FI architecture. On the Arm-Arrow environment, RGE-(ISOMAP/PCA/AE) architectures gave almost the same performances, slightly underneath the FI architecture. RGE-(VAE/RFVAE) architectures performed fairly worse than the other representations.

[15] $[0,1]^2$ for Arm-Ball $[0,1]^3$ for Arm-Arrow

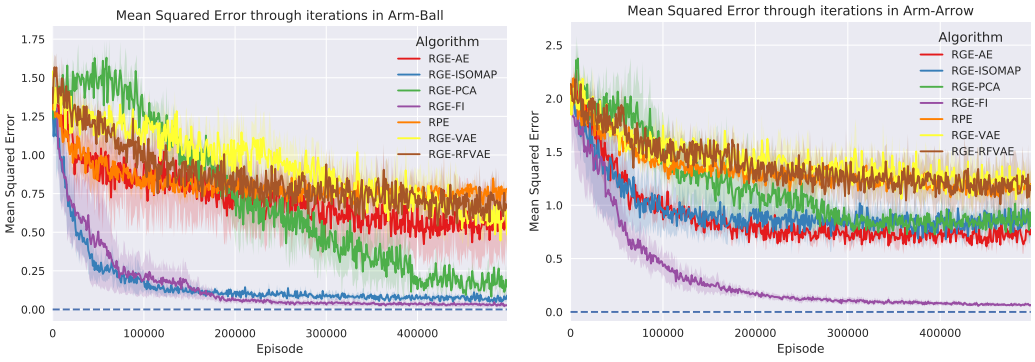

Figure 3: Mean Squared Error evolution through episodes for the different architectures considered.

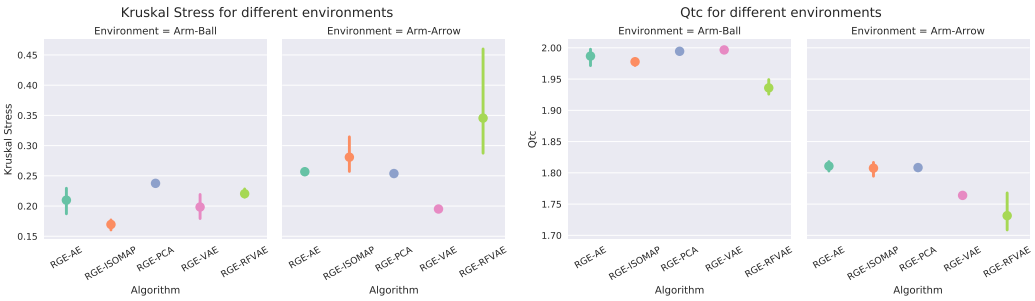

Figure 4: Kruskal Stress and Trustworthiness & Continuity Quality measure of the different Architectures on the two environments.

In the Figure 3, we represented the evolution of the Meta-Policy MSE through exploration episodes. It is worth noting that the Meta-Policy MSE is not as simple to interpret, since it does not re Again, we can see that the final MSE achieved by the representation based architectures is mostly contained within the final MSE of the RGE-FI architecture and the RP architecture. In particular, we observe that for both environments, RGE-(VAE/RFVAE) architectures yield almost as low performances as RPE. Concerning RGE-(ISOMAP/PCA) architectures, their final performance on the Arm-Ball environment equals the RGE-FI performance. The same is not observed on the Arm-Arrow environment for which RGE-(ISOMAP/PCA/AE) architectures gives almost the same performance; lower than the one achieved by the RGE-FI architecture.

Representation Quality and Correlations with Final Exploration Performances: In the figure 4, we can observe the Kruskal Stress and $Q_{T\&C}$ of the different learned representations, on both environments. Concerning the Kruksal stress, which represents the global geometric quality of the representation, we can see that most algorithms gives almost equivalent performances on this point. The RGE-RFVAE architecture performs particularly worse than the others on the Arm-Arrow environment. This is surprising, since this environment contains a periodic parameter, which was supposed to be well represented thanks to Radial Flow. Concerning the Local geometric quality, measured with $Q_{T\&C}$, we observe that it is good for the Arm-Ball environment, and falls equivalently for all representation algorithms on the Arm-Arrow problem.

In order to check the existence of any correlations between the quality of the learned representations and the exploration performances, we artificially degraded the performances of the different representation learning algorithms by adding noise to the training images[16]. This allowed us to gather data on the exploration performances on representations of different qualities. In the Figure 5, we have represented the joint plots of the final MSE and ER achieved, against the Kruskal Stress and the $Q_{T\&C}$, for both environments. First we can observe various correlation structures (measured by

[16]We used *Salt & Pepper* noise for different ratio of corrupted pixels in [0.01, 0.05, 0.1, 0.15, 0.2, 0.25, 0.5, 0.7]

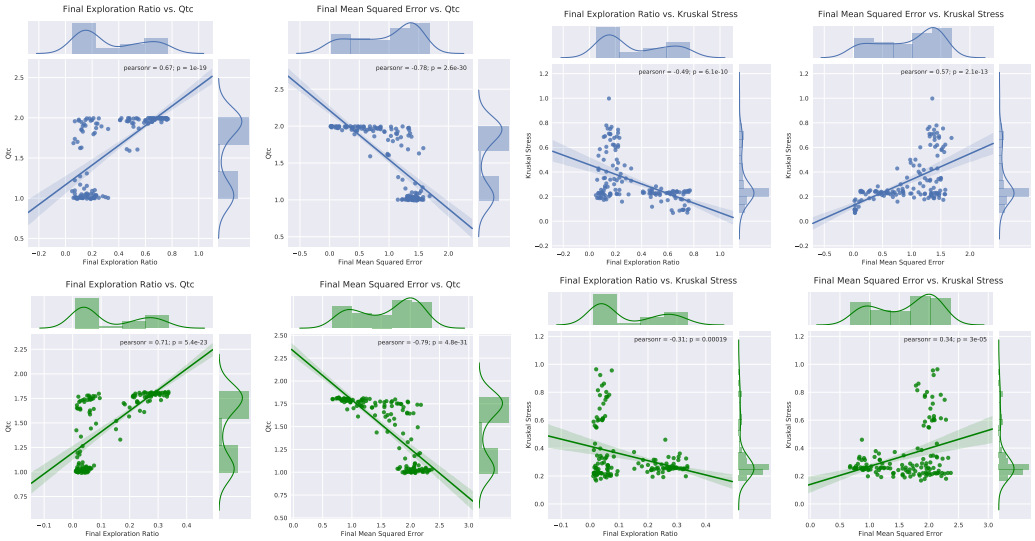

Figure 5: Joint Plots of Final MSE and ER against Representation Quality Measures. In Blue the joint plots for the Arm-Ball environment, and in green the joint plots for Arm-Arrow.

Pearson tests), and the relationships between Exploration performances and representation quality seem to follow the same dynamics on both environments. In both cases, a low $Q_{T\&C}$ tends to correspond to lowers final ER and larger final MSE. While not giving always the best performances, a better local geometric quality seems to be related with better final ER and MSE. Regarding the Kruskal Stress, we can see that a high Kruskal Stress is almost always related with poor exploration performances both in ER and MSE. Those visual interpretations are corroborated by the evaluation of diverse correlation measures represented in Appendix in Table 1.

Globally, those results follow our first intuitions, that having an Outcome Space badly conditioned regarding the underlying phenomenon seems to consistently lead to poorer exploration performances. On the other side, the exploration does not always benefit from a better Local, nor Global, representation quality. This tends to acknowledge the point that exploration performance relies on other factors; one being the distribution used as stationary Goal Policy that we present in the next paragraph.

Impact of Sampling Kernel Density Estimation One of the other factor impacting the exploration was assessed during our experiments was the importance of the distribution used as stationary Goal Policy. If in most cases, the representation algorithm gives no particular prior knowledge of $p(o)$, in the case of Variational Auto-Encoders, it is assumed in the derivation that $p(o) = \mathcal{N}(0, I)$. Hence, the isotropic Gaussian distribution is a good candidate to be used as stationary Goal Policy, rather than using a Kernel Density Estimation. In Figures 6 and 10, we can see the comparison between exploration performances achieved with RGE-VAE and RGE-RFVAE using a Kernel Density Estimated distribution or an isotropic Gaussian as Goal Policy. We can see that the performances are severely degraded by using the isotropic Gaussian. In terms of Exploration Ratio, the performances are as poor as the RPE performances, and concerning Mean Squared Error, the performances are lower than the RPE. In this we can see that, in the case of stationary Goal Policy, the actual distribution used to sample goals seems to be of great importance to give good exploration results.

5 Conclusion

In this paper, we proposed a new Intrinsically Motivated Goal Exploration Strategy architecture where the Outcome Space representation is learned using pixel-level observations of world changes caused by another agent, rather than being crafted by an engineer. We tested this architecture on two simulated benchmarks parameterized by bounded and periodic continuous parameters. We compared this architecture with classic architectures involving Full Information on the environment

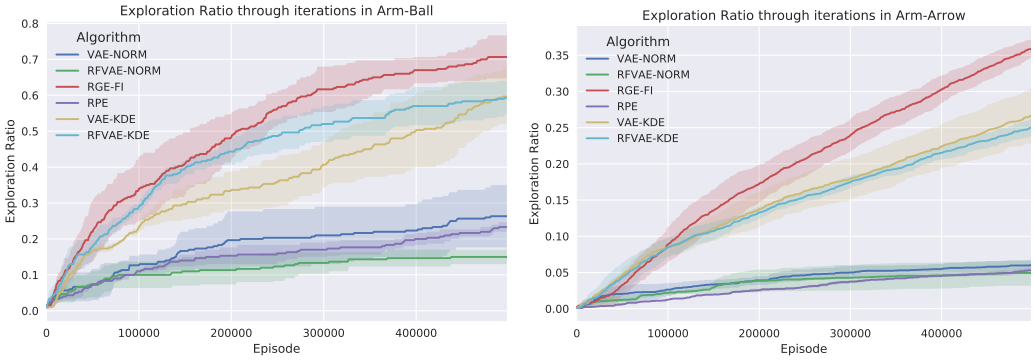

Figure 6: Exploration Ratio evolution for Variational Representations using KDE or Isotropic Gaussian prior.

state-space, and random parameterizations exploration. Compared to those boundary cases, our new architecture gives good exploration performances, but stays back in terms of mean squared error. In particular, we observe that the several deep representation learning algorithms did not give competitive performances regarding more classical algorithms such as Isomap and PCA.

We also gave evidences on the factors of performances of those exploration architectures. We saw that a representation with poor geometric quality regarding the underlying phenomenon was almost always related with poor performances in explorations. We also brought evidences on the importance of having a good goal policy in the stationary case, with possibly catastrophic impact on exploration performances. This calls for studying in future work the impact of using adaptive goal sampling policies, for e.g. with the SAGG-RIAC architecture where goals are sampled based on an estimation of expected competence progress Baranes & Oudeyer (2013).

This work was limited to a fairly restricted set of environments. Experimenting over a larger set of environments would benefit our understanding of the exploration algorithms in general. In particular, as Forestier et al. (2017) has shown the benefits of using modular representations of goal spaces when the environment is composed of independantly controllable objects, a topic of future work will be to consider representation learning algorithms that enable to learn such modular representations. Moreover, in this paper, we only studied a learning scenario where representation learning happens before autonomous exploration. The ability to jointly learn an Outcome Space and exploring would, as well, be an interesting topic for future work.

REFERENCES

Peter M Andreae and John H Andreae. A teachable machine in the real world. *International Journal of Man-Machine Studies*, 10(3):301–312, 1978.

Gianluca Baldassarre, Marco Mirolli, and Andrew G. Barto. Intrinsically motivated learning in natural and artificial systems. 2013.

Adrien Baranes and Pierre Yves Oudeyer. R-IAC: Robust intrinsically motivated exploration and active learning. *IEEE Transactions on Autonomous Mental Development*, 1(3):155–169, 2009. doi: 10.1109/TAMD.2009.2037513.

Adrien Baranes and Pierre-Yves Oudeyer. Active learning of inverse models with intrinsically motivated goal exploration in robots. *Robotics and Autonomous Systems*, 61(1), 2013.

Andrew G Barto. Intrinsic motivation and reinforcement learning. In *Intrinsically motivated learning in natural and artificial systems*, pp. 17–47. Springer, 2013.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pp. 1471–1479, 2016.

Daniel E Berlyne. Curiosity and exploration. *Science*, 153(3731):25–33, 1966.

Christopher M Bishop. *Pattern Recognition and Machine Learning*, volume 4. 2006. ISBN 9780387310732. doi: 10.1117/1.2819119. URL http://www.library.wisc.edu/selectedtocs/bg0137.pdf.

Leon Bottou. Online Algorithms and Stochastic Approximations. In *Online Learning and Neural Networks*, pp. 1–34. 1998. ISBN 978-0521117913.

Herv Bourlard and Yves Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics*, 59(4-5):291–294, 1988. ISSN 03401200. doi: 10.1007/BF00332918.

Serkan Cabi, Sergio Gomez Colmenarejo, Matthew W. Hoffman, Misha Denil, Ziyu Wang, and Nando de Freitas. The intentional unintentional agent: Learning to solve many continuous control tasks simultaneously. *CoRR*, abs/1707.03300, 2017.

Angelo Cangelosi, Matthew Schlesinger, and Linda B Smith. *Developmental robotics: From babies to robots*. MIT Press, 2015.

Nuttapong Chentanez, Andrew G Barto, and Satinder P Singh. Intrinsically motivated reinforcement learning. In *Advances in neural information processing systems*, pp. 1281–1288, 2005.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011. ISSN 15324435. doi: 10.1109/CDC.2012.6426698. URL http://jmlr.org/papers/v12/duchi11a.html.

Carlos Florensa, David Held, Markus Wulfmeier, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. *arXiv preprint arXiv:1707.05300*, 2017.

Sebastien Forestier and Pierre Yves Oudeyer. Modular active curiosity-driven discovery of tool use. In *IEEE International Conference on Intelligent Robots and Systems*, volume 2016-November, pp. 3965–3972, 2016. ISBN 9781509037629. doi: 10.1109/IROS.2016.7759584.

Sebastien Forestier, Yoan Mollard, and Pierre-Yves Oudeyer. Intrinsically motivated goal exploration processes with automatic curriculum learning. *CoRR*, abs/1708.02190, 2017.

Karl J Friston, Marco Lin, Christopher D Frith, Giovanni Pezzulo, J Allan Hobson, and Sasha Ondobaka. Active inference, curiosity and insight. *Neural Computation*, 2017.

Alison Gopnik, Andrew N Meltzoff, and Patricia K Kuhl. *The scientist in the crib: Minds, brains, and how children learn.* William Morrow & Co, 1999.

Antonio Gracia, Santiago González, Victor Robles, and Ernestina Menasalvas. A methodology to compare Dimensionality Reduction algorithms in terms of loss of quality. *Information Sciences*, 270:1–27, 2014. ISSN 00200255. doi: 10.1016/j.ins.2014.02.068.

Irina Higgins, Loic Matthey, Xavier Glorot, Arka Pal, Benigno Uria, Charles Blundell, Shakir Mohamed, and Alexander Lerchner. Early visual concept learning with unsupervised deep learning. *CoRR*, abs/1606.05579, 2016.

Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, pp. 1109–1117, 2016.

Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation*, 25(2): 328–73, 2013. ISSN 1530-888X. doi: 10.1162/NECO_a_00393. URL http://www.ncbi.nlm.nih.gov/pubmed/23148415.

Frederic Kaplan and Pierre-Yves Oudeyer. Motivational principles for visual know-how development. 2003.

Diederik P. Kingma and Jimmy Lei Ba. Adam: a Method for Stochastic Optimization. *International Conference on Learning Representations 2015*, pp. 1–15, 2015. ISSN 09252312. doi: http://doi.acm.org.ezproxy.lib.ucf.edu/10.1145/1830483.1830503.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes, 2013. URL http://arxiv.org/abs/1312.6114.

J. B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964. ISSN 00333123. doi: 10.1007/BF02289565.

Daniel Y Little and Friedrich T Sommer. Learning and exploration in action-perception loops. *Frontiers in neural circuits*, 7, 2013.

Shamima Najnin and Bonny Banerjee. A predictive coding framework for a developmental agent: Speech motor skill acquisition and speech production. *Speech Communication*, 2017.

Sao Mai Nguyen and Pierre-Yves Oudeyer. Socially guided intrinsic motivation for robot learning of motor skills. *Autonomous Robots*, 36(3):273–294, 2014. doi: 10.1007/s10514-013-9339-y.

Pierre Yves Oudeyer, Frederic Kaplan, and Verena V. Hafner. Intrinsic motivation systems for autonomous mental development. *IEEE Transactions on Evolutionary Computation*, 11(2):265–286, 2007. ISSN 1089778X. doi: 10.1109/TEVC.2006.890271.

Pierre-Yves Oudeyer, Adrien Baranes, and Frédéric Kaplan. Intrinsically motivated learning of real-world sensorimotor skills with developmental constraints. In *Intrinsically motivated learning in natural and artificial systems*, pp. 303–365. Springer, 2013.

Pierre-Yves Oudeyer, Manuel Lopes, Celeste Kidd, and Jacqueline Gottlieb. Curiosity and intrinsic motivation for autonomous machine learning. *ERCIM News*, 107:2, 2016.

Emanuel Parzen. On Estimation of a Probability Density Function and Mode. *The Annals of Mathematical Statistics*, 33(3):1065–1076, 1962. ISSN 0003-4851. doi: 10.1214/aoms/1177704472. URL http://projecteuclid.org/euclid.aoms/1177704472.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the seventh international conference on machine learning*, 2017.

Danilo J. Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *ICML*, 2015.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, volume 32 of *JMLR Workshop and Conference Proceedings*, pp. 1278–1286. JMLR.org, 2014. URL http://dblp.uni-trier.de/db/conf/icml/icml2014.html#RezendeMW14.

M. Rolf, J.J. Steil, and M. Gienger. Goal babbling permits direct learning of inverse kinematics. *IEEE Transactions on Autonomous Mental Development*, 2(3), 2010.

Murray Rosenblatt. Remarks on Some Nonparametric Estimates of a Density Function. *The Annals of Mathematical Statistics*, 27(3):832–837, 1956. ISSN 0003-4851. doi: 10.1214/aoms/1177728190. URL http://projecteuclid.org/euclid.aoms/1177728190.

Christoph Salge, Cornelius Glackin, and Daniel Polani. Changing the environment based on empowerment as intrinsic motivation. *Entropy*, 16(5):2789–2819, 2014.

Stefan Schaal, Auke Ijspeert, and Aude Billard. Computational approaches to motor learning by imitation. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 358 (1431):537–547, 2003.

J Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *From animals to animats: Proceedings of the first international conference on simulation of adaptive behavior*, pp. 15–21, 1991.

Casper Kaae Sonderby, Tapani Raiko, Lars Maale, Soren Kaae Sonderby, and Ole Winther. Ladder variational autoencoders. Feb 2016. URL http://arxiv.org/abs/1602.02282v3.

Kenneth O Stanley and Joel Lehman. *Why greatness cannot be planned: The myth of the objective*. Springer, 2015.

Richard S Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the seventh international conference on machine learning*, pp. 216–224, 1990.

Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. # exploration: A study of count-based exploration for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, 2017.

J B Tenenbaum, Veronica De Silva, and J C Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290 5500:2319–23, 2000.

Jakub M. Tomczak and Max Welling. Improving variational auto-encoders using householder flow. Dec 2016. URL http://arxiv.org/abs/1611.09630v4.

Jarkko Venna and Samuel Kaski. Local multidimensional scaling with controlled tradeoff between trustworthiness and continuity. *Proceedings of 5th Workshop on SelfOrganizing Maps*, 5:695–702, 2005. URL http://eprints.pascal-network.org/archive/00001233/.

Claes Von Hofsten. An action perspective on motor development. *Trends in cognitive sciences*, 8 (6):266–272, 2004.

Appendix

A INTRINSICALLY MOTIVATED GOAL EXPLORATION PROCESS

Intrinsically Motivated Goal Exploration Processes are algorithmic architectures that can be instantiated into different exploration algorithms depending on the problem to explore. The general architecture is represented on Algorithm 2.

Algorithmic Architecture 2: Intrinsically Motivated Goal Exploration Strategy

Input:
Regressor $\tilde{D}_{running}$, Regressor $\tilde{D}$, Regressor $\tilde{I}$, Goal Policy γ, Minimizer Π

```
begin
    for A fixed number of Bootstrapping iterations do
        Observe context c
        Sample θ ~ U(θ)
        Perform experiment and retrieve outcome o
        Update Regressor D̃_running with tuple {c, θ, o}
    for A fixed number of Exploration iterations do
        Observe context c
        Sample a goal τ ~ γ
        Compute θ = arg min_θ C_τ(D̃_running(θ, c)) using a minimizing algorithm
        Perform experiment and retrieve outcome o
        Update Regressor D̃_running with a tuple {c, θ, o}
        Update Goal Policy γ, according to Intrinsic Motivation strategy
    while exploration running, asynchronously do
        Update regressors D̃ and Ĩ with acquired samples {c, θ, o}_i
return The forward model D̃ and the inverse model Ĩ.
```

B DEEP REPRESENTATION LEARNING ALGORITHMS

Cost Functions used to train the different Deep Representation Learning algorithms used, can be motivated by a few theoretical arguments we summarized here.

Auto-Encoders (AEs) The choice of the cost function can be motivated by considering the network as composed of:

- An *encoder* network parameterized by weights θ that maps an input $\mathbf{x}$ to its deterministic representation $\mathbf{z} = f_\theta(\mathbf{x})$.
- A *decoder* network parameterized by weights ϕ that maps a representation $\mathbf{z}$ to a vector ξ parameterizing a distribution $p_\xi(\mathbf{x}|\mathbf{z})$ with $\xi = g_\phi(\mathbf{z})$.

Under this stochastic decoding assumption, the Maximum Likelihood principle is used to train the model, i.e. AEs can maximize the likelihood of data under the model. In the case of Auto-Encoders, this principle is compatible with Gradient Descent, and we can use the negative log-likelihood as a cost function to be minimized. If input $\mathbf{x}$ is binary valued, $p(\mathbf{x}|\mathbf{z})$ is assumed to follow a multivariate Bernouilli distribution of ξ parameters [17], and the log likelihood of the dataset $\mathcal{D}$ is expressed as:

$$\log \mathfrak{L}(\mathcal{D}) = \sum_{i=1}^{N} \log p(\mathbf{x}^{(i)}|\xi^{(i)}) = \sum_{i=1}^{N}\sum_{k=1}^{D} \left[x_k^{(i)} \log \xi_k^{(i)} + (1 - x_k^{(i)}) \log(1 - \xi_k^{(i)})\right], \qquad (1)$$

[17] This requires that the output layer uses a sigmoid function that restricts the values of output to lie in $[0, 1]$.

with $\xi^{(i)} = g_\phi(f_\theta(\mathbf{x}^{(i)}))$. For a binary valued input vector $\mathbf{x}^{(i)}$, the unitary Cost Function to minimize is:

$$\mathcal{J}(\theta, \phi, \mathbf{x}^{(i)}) = -\sum_{d=1}^{D} \left[x_d^{(i)} \log(g_\phi(f_\theta(\mathbf{x}^{(i)}))_d) + (1 - x_d^{(i)}) \log(1 - g_\phi(f_\theta(\mathbf{x}^{(i)}))_d) \right], \quad (2)$$

provided that f_θ is the encoder part of the architecture and g_ϕ is the decoding part of the architecture. This Cost Function can be minimized using Stochastic Gradient Descent (Bottou, 1998), or more advanced optimizers such as Adagrad (Duchi et al., 2011) or Adam (Kingma & Ba, 2015).

Depending on the depth of the network[18], those architectures can prove difficult to train using vanilla Stochastic Gradient Descent. A particularly successful procedure to overcome this difficulty is to greedily train each pairs of encoding-decoding layers and stacking those to sequentially form the complete network. This procedure, known as stacked AEs, accelerates convergence. But it has shown bad results with our problem, and thus was discarded for the sake of clarity.

Variational Auto-Encoders (VAEs) If we assume that the observed data are realizations of a random variable $\mathbf{x} \sim p(\mathbf{x}|\psi)$, we can hypothesize that they are conditioned by a random vector of independent factors $\mathbf{z} \sim p(\mathbf{z}|\psi)$. In this setting, learning the model would amount to searching the parameters ψ of both distributions. We might use the same principle of maximum likelihood as before to find the best parameters by computing the likelihood $\log \mathfrak{L}(\mathcal{D}) = \sum_{i=1}^{N} \log p(\mathbf{x}^{(i)}|\psi)$ by using the fact that $p(\mathbf{x}|\psi) = \int p(\mathbf{x}, \mathbf{z}|\psi) d\mathbf{z} = \int p(\mathbf{x}|\mathbf{z}, \psi) p(\mathbf{z}|\psi) d\mathbf{z}$. Unfortunately, in most cases, this integral is intractable and cannot be approximated by Monte-Carlo sampling in reasonable time. To overcome this problem, we can introduce an arbitrary distribution $q(\mathbf{z}|\mathbf{x}, \chi)$ and remark that the following holds:

$$\log p(\mathbf{x}|\psi) = \mathcal{L}(q, \psi) + \mathbb{D}_{KL}[q(\mathbf{z}|\mathbf{x}, \chi) \| p(\mathbf{z}|\mathbf{x}, \psi), \quad (3)$$

with the Evidence Lower Bound being:

$$\mathcal{L}(q, \psi) = \underbrace{\mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x}, \psi)}[\log p(\mathbf{x}|\mathbf{z}, \psi)]}_{a} - \underbrace{\mathbb{D}_{KL}[q(\mathbf{z}|\mathbf{x}, \chi) \| p(\mathbf{z}, \psi)]}_{b}. \quad (4)$$

Looking at Equation (3), we can see that since the Kullback Leibler (KL) Divergence is non-negative, $\mathcal{L}(q, \psi) \leq \log p(\mathbf{x}|\psi) - \mathbb{D}_{KL}([q(\mathbf{z}|\mathbf{x}, \chi) \| p(\mathbf{z}|\mathbf{x}, \psi)]$ whatever the q distribution, hence the name of Evidence Lower Bound (ELBO). Consequently, maximizing the ELBO have the effect to maximize the log likelihood, while minimizing the KL-Divergence between the approximate $q(\mathbf{z}|\mathbf{x})$ distribution, and the true unknown posterior $p(\mathbf{z}|\mathbf{x}, \psi)$. The approach taken by VAEs is to *learn* the parameters of both conditional distributions $p(\mathbf{x}|\mathbf{z}, \psi)$ and $q(\mathbf{z}|\mathbf{x}, \chi)$ as non-linear functions. Under some restricted conditions, Equation (4) can be turned into a valid cost function to train a neural network. First, we hypothesize that $q(\mathbf{z}|\mathbf{x}, \chi)$ and $p(\mathbf{z}|\psi)$ follow Multivariate Gaussian distributions with diagonal covariances, which allows us to compute the b term in closed form. Second, using the Gaussian assumption on q, we can reparametrize the inner sampling operation by $\mathbf{z} = \mu + \sigma^2 \odot \epsilon$ with $\epsilon \sim \mathcal{N}(0, I)$. Using this trick, the Path-wise Derivative estimator can be used for the a member of the ELBO. Under those conditions, and assuming that $p(\mathbf{x}|\psi)$ follows a Multivariate Bernouilli distribution, we can write the cost function used to train the neural network as:

$$\begin{aligned} \mathcal{J}(\psi, \chi, \mathbf{x}^{(i)}) = &-\frac{1}{2} \sum_{j=1}^{J} (1 + \log(\sigma(\mathbf{x}^{(i)})_j^2) - \mu(\mathbf{x}^{(i)})_j^2 - \sigma(\mathbf{x}^{(i)})_j^2) \\ &- \sum_{k=1}^{D} \left[x_k^{(i)} \log(g_\psi(f_\chi(\mathbf{x}^{(i)}))_k) + (1 - x_k^{(i)}) \log(1 - g_\psi(f_\chi(\mathbf{x}^{(i)}))_k) \right], \end{aligned} \quad (5)$$

where f_χ represents the encoding and sampling part of the architecture and g_ψ represents the decoding part of the architecture. In essence, this derivation simplifies to the initial cost function used in AEs augmented by a term penalizing the divergence between $q(\mathbf{z}|\mathbf{x}, \chi)$ and the assumed prior that $p(\mathbf{x}|\psi) = \mathcal{N}(0, I)$.

[18] By depth here, we indicate the number of layers of the neural network.

Normalizing Flow overcomes the problem stated earlier, by permitting more expressive prior distributions (Rezende & Mohamed, 2015). It is based on the classic rule of change of variables for random variables. Considering a random variable $\mathbf{z}_0 \sim q(\mathbf{z}_0)$, and an invertible transformation $t : \mathbb{R}^d \mapsto \mathbb{R}^d$, if $\mathbf{z} = t(\mathbf{z}_0)$, then:

$$q(\mathbf{z}) = q(\mathbf{z}_0)\left|\det \frac{\partial t^{-1}}{\partial \mathbf{z}_0}\right| = q(\mathbf{z}_0)\left|\det \frac{\partial t}{\partial \mathbf{z}_0}\right|^{-1}. \tag{6}$$

We can then directly chain different invertible transformations $t_1, t_2, \ldots, t_K$ to produce a new random variable $\mathbf{z}_K = t_K \circ \cdots \circ t_2 \circ t_1(\mathbf{z}_0)$. In this case, we have:

$$\log q(\mathbf{z}_k) = \log\left(q(\mathbf{z}_0)\prod_{k=1}^{K}\left|\det \frac{\partial t_k}{\partial \mathbf{z}_{k-1}}\right|^{-1}\right) = \log q(\mathbf{z}_0) - \sum_{k=1}^{K}\log\left|\det \frac{\partial t_k}{\partial \mathbf{z}_{k-1}}\right|. \tag{7}$$

The great interest of this formulation is that the *Law Of The Unconscious Statistician* allows us to compute expectations over $q(\mathbf{z}_k)$ without having a precise knowledge of it:

$$\mathbb{E}_{\mathbf{z}_k \sim q(\mathbf{z}_k)}[h(\mathbf{z}_k)] = \mathbb{E}_{\mathbf{z}_0 \sim q(\mathbf{z}_0)}[h(t_k \circ \ldots t_2 \circ t_1(\mathbf{z}_0))], \tag{8}$$

provided that h does not depends on $q(\mathbf{z}_k)$. Using this principle on the ELBO allows us to derive the following:

$$\begin{aligned}\mathcal{L}(q, \theta, \phi) =& \mathbb{E}_{\mathbf{z}_0 \sim q(\mathbf{z}_0|\mathbf{x})}[\log p(\mathbf{x}|t_K \circ \ldots t_2 \circ t_1(\mathbf{z}_0))] \\ &- \mathbb{D}_{KL}[q(\mathbf{z}_0|\mathbf{x})\|p(\mathbf{z}_0)] \\ &+ 2\mathbb{E}_{\mathbf{z}_0 \sim q(\mathbf{z}_0|\mathbf{x})}\left[\sum_{k=1}^{K}\log\left|\det \frac{\partial t_k}{\partial \mathbf{z}_{k-1}}\right|\right]\end{aligned} \tag{9}$$

This is nothing more than the regular ELBO with an additional term concerning the log-determinant of the transformations. In practice, as before, we use $p(\mathbf{z}_0) = \mathcal{N}(\mathbf{z}_0; \mathbf{0}, \mathbf{I})$, and $q(\mathbf{z}_0|x) = \mathcal{N}(\mathbf{z}_0; \mu(x), diag(\sigma(x)^2))$. We only have to find out parameterized transformations t, whose parameters can be learned and have a defined log-determinant. Using *radial flow*, which is expressed as:

$$t(\mathbf{z}) = \mathbf{z} + \beta h(\alpha, \mathbf{r})(\mathbf{z} - \mathbf{c}), \tag{10}$$

where $\mathbf{r} = |\mathbf{z} - \mathbf{c}|$, $h(\alpha, \mathbf{r}) = \frac{1}{\alpha + \mathbf{r}}$ and $\alpha, \beta, \mathbf{c}$ are learnable parameters of the transformation, our cost function can be written as:

$$\begin{aligned}\mathcal{J}(\psi, \chi, \mathbf{x}^{(i)}) =& -\frac{1}{2}\sum_{j=1}^{J}(1 + \log(\sigma(\mathbf{x}^{(i)})_j^2) - \mu(\mathbf{x}^{(i)})_j^2 - \sigma(\mathbf{x}^{(i)})_j^2) \\ &- \sum_{d=1}^{D}\left[x_d^{(i)}\log(g_\psi(f_\chi(\mathbf{x}^{(i)}))_d) + (1 - x_d^{(i)})\log(1 - g_\psi(f_\chi(\mathbf{x}^{(i)}))_d)\right] \\ &- 2\sum_{k=1}^{K}\log[1 + \beta_k h(\alpha_k, \mathbf{r}))]^{D-1}[1 + \beta_k h(\alpha, \mathbf{r})) + \beta_k h'(\alpha, r)r],\end{aligned} \tag{11}$$

provided that f_χ represents the encoding, sampling ad transforming part of the architecture, g_ψ represents the decoding part of the architecture, and β_k, α_k, c_k are the parameters of the different transformations.The effect of this flow can also be seen in Figure 7. Other types of transformations have been proposed lately. The Householder flow Tomczak & Welling (2016) is a volume preserving transformation, meaning that its log determinant equals 1, with the consequence that it can be used with no modifications of the loss function. A more convoluted type of transformations based on a masked autoregressive auto-encoder, the Inverse Autoregressive Flow, was proposed in Kingma & Welling (2013). We did not explore those two last approaches.

C Experimental Environments

The following environments were considered:

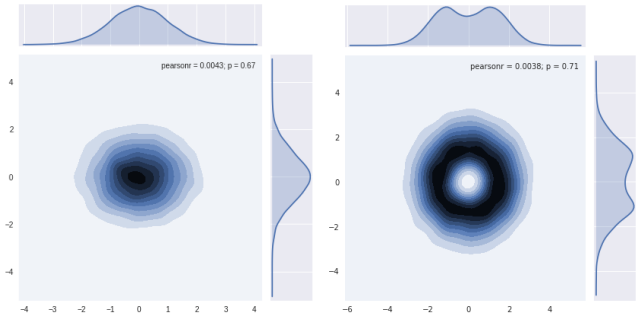

Figure 7: Effect of a Radial Flow transformation on an Isotropic Gaussian Distribution.

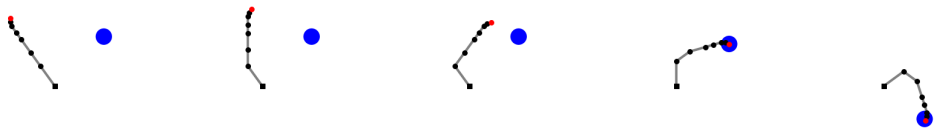

Figure 8: A DMP executed on the Arm-Ball environment.

- **Arm-Ball:** A 7 joints arm, controlled in angular position, can move around in an environment containing a ball. The environment state is perceived visually as a 50x50 pixels image. The arm has a sticky arm tip: if the tip of the arm touches the ball, the ball sticks to the arm until the end of the movement. The underlying state of the environment is hence parameterized by two bounded continuous factors which represent the coordinates of the ball. A situation can be sampled by the experimenter by taking a random point in $[0, 1]^2$.
- **Arm-Arrow:** The same arm can manipulate an arrow in a plane, an arrow being considered as an object with a single symmetry that can be oriented in space. Consequently, the underlying state of the environment is parameterized by two bounded continuous factors representing the coordinates of the arrow , and one periodic continuous factor representing its orientation. A particular situation can hence be sampled by taking a random point in $[0, 1]^3$.

The physical situations were represented by small 70x70 images very similar to the *dSprites* dataset proposed by Higgins et al. (2016)[19]. We took the approach to consider that the arm was not contained in the field of view of the (virtual) camera used to gather images for Representation Learning, and consequently, only the objects were represented on the images. We used a robotic arm composed of 7 joints, whose motions were parameterized by Dynamic Movement Primitives (DMP) (Ijspeert et al., 2013) using 3 basis functions (hence action policies have 21 continuous parameters), during 50 time-steps. An example of such a DMP executed in the environment is represented in Figure 8. The first phase, where the learner observes changes of the environment (= ball moves) caused by another agent, is modeled by a process which samples iteratively a random state in the underlying state space, e.g. in the case of Arm-Ball $s \sim \mathcal{U}([0, 1]^2)$, and then generating the corresponding image $x = f(s)$ that is observed by the learner.

D Algorithmic Implementation

For the experiments, we instantiated the Algorithmic Architecture **??** by the following Algorithm 3.

In the text, we denote the (RGE-$\star$) this algorithm used with the $\star$ representation learning algorithm, which can be (RGE-AE) for Auto-Encoders, (RGE-VAE) for Variational Auto-Encoders, (RGE-RFVAE) for Radial Flow Variational Auto-Encoders, (RGE-ISOMAP) for Isomap, (RGE-PCA) for

[19] Available at `https://github.com/deepmind/dsprites-dataset`

Principal Component Analysis and (RGE-FI) for Full Information. To denote the Random Parameterization Exploration, we use the acronym (RPE).

Algorithm 3: Random Goal Exploration with Goal Space Learning

Input:
k-neighbors regressor $\tilde{D}_{running}$
Representation function $\tilde{R}$ with a learning algorithm
Random exploration noise γ_m
Random exploration ratio Γ_e

```
begin
    for 10000 Observation Iterations do
        Observe a random environment image x_i
        Add this image to a database D = {x_i}_{i∈[0,10000]}
    Train R̃ using D
    Estimate the outcome distribution p_kde(o) from {R̃(x_i)}_{i∈[0,10000]}
    Set the Goal Policy γ = p_kde to be the estimated outcome distribution
    for 100 Bootstrapping iterations do
        Sample a random parameterization θ_i ~ p(θ)
        Execute the experiment θ_i (= run a controller with parameters θ_i)
        Retrieve the outcome from raw image o_i = R̃(x_i)
        Update the forward model with D̃(θ_i) ≜ o_i
    for 5000 Exploration iterations do
        if u ~ U(0,1) < Γ_e then
            Sample a random parameterization θ_i ~ p(θ)
        else
            Sample a goal g_i ~ γ
            Sample an exploration noise ε ~ N(0, I)
            Execute Meta-Policy to find the approximately optimal θ_i = Π(g_i) + ε
        Execute the experiment θ_i
        Retrieve the outcome from raw image o_i = R̃(x_i)
        Update the forward model with D̃(θ_i) ≜ o_i
return The forward model D̃
```

E Details of Neural Architectures

The architectures used for Deep Representation Learning algorithms were based on the architecture proposed in Higgins et al. (2016), and was tweaked until the best result were obtained.

Auto-Encoder

- **Dense layer:** of size 1200 with relu activations
- **Dense layer:** of size 1200 with relu activations
- **Dense layer:** of size 10
- **Dense layer:** of size 1200 with relu activations
- **Dense layer:** of size 1200 with relu activations
- **Dense layer:** of size defined by images with linear activations

The architecture was trained directly without particular stacking. The **AdaGrad** optimizer was used, with initial learning rate of $1e-3$, with batches of size 100, until convergence at $2e5$ epochs.

Variational Auto-Encoder

- **Dense layer:** of size 1200 with relu activations
- **Dense layer:** of size 256 with relu activations
- **Dense layer:** of size 10*2 (10 for μ and 10 for σ)
- **Dense layer:** of size 256 with relu activations
- **Dense layer:** of size 1200 with relu activations
- **Dense layer:** of size defined by images with linear activations

The architecture was trained with a *deterministic warmup* of $1e4$ epochs, as proposed in Sonderby et al. (2016), which shows improved convergence rate. The **Adam** optimizer was used, with initial learning rate of $1e-3$, with batches of size 100, until convergence at $1e5$ epochs.

Radial Flow Variational Auto-Encoder

- **Dense layer:** of size 1200 with relu activations
- **Dense layer:** of size 256 with relu activations
- **Dense layer:** of size 10*2+10*(10+1+1) (10 for μ, 10 for σ, and 10 for c_k, 1 for β_k and 1 for α_k for each transformation)
- **Flow layer:** of 10 chained transformations
- **Dense layer:** of size 256 with relu activations
- **Dense layer:** of size 1200 with relu activations
- **Dense layer:** of size defined by images with linear activations

The architecture was trained with a *deterministic warmup* of $1e4$ epochs. The complete flow was made out of 10 planar flows as proposed in Rezende & Mohamed (2015), whose parameters were learned by the encoder. The **Adam** optimizer was used, with initial learning rate of $1e-3$, with batches of size 100, until convergence at $5e4$ epochs.

F Quality and Performance measures

To assess the quality of the learned representation, we used classic measures from the Manifold Learning literature. Those measures mainly rely on the construction of a k-neighbors graph in both the input space and the output space using tuples of $\{a_i, b_i\}$ where a_i is a point in the input space, and b_i is the representation of this point in the output space[20]. We considered:

- **Local Measure** We used the following measures extracted from Venna & Kaski (2005):
  - **Trustworthiness:** If $r_i(j)$ is the rank of the node j in the k-Neighbors of i in the input space, and $U_k(i)$ the set of nodes present in the k-Neighbors of node i in output space but not in input space, the measure is defined as:

$$T(k) = 1 - \frac{2}{Nk(2N-3k-1)} \sum_{i=1}^{N} \sum_{j \in U_k(i)} (r_i(j) - k). \tag{12}$$

  - **Continuity:** If $\hat{r}_i(j)$ is the rank function in output space, and $V_k(i)$ the set of nodes present in the k-Neighbors of node i in input space but not in output space, the measure is defined as:

$$C(k) = 1 - \frac{2}{Nk(2N-3k-1)} \sum_{i=1}^{N} \sum_{j \in V_k(i)} (\hat{r}_i(j) - k). \tag{13}$$

  - **Quality:** Is a composite measure that sums the two previous ones:

$$Q_{T\&C}(k) = T(k) + C(k). \tag{14}$$

[20] A particular attention was taken to come with a k-neighbors graph with no discontinuity, in the case of the Arm-Arrow state-space, which contains a periodic parameter.

The neighbors size is an hyper-parameter of the measure. In practice we used $k = 7$ as proposed in the extensive review Gracia et al. (2014).

- **Global Measure** We used the measure of global reduction stress from the pioneer work on manifold learning of Kruskal (1964).
  - **Kruskal Stress:** Considering all the distances ξ_{ij} in the input space and their counterpart δ_{ij} in the output space, we perform an Isotonic Regression and compute the stress measure as:

$$S_{Kr} = \sqrt{\frac{\sum_{ij}(\xi_{ij} - d^{\star}_{ij})}{\sum_{ij} \xi^2_{ij}}}, \tag{15}$$

  where $d^{\star}_{ij}$ is the transformation of δ_{ij} on the solution of the Isotonic Regression.

Concerning the performances of the exploration algorithm, we used two classic measures addressing the two main interests of those algorithms:

- **Exploration:** We monitored the **Exploration Ratio** (ER) through iterations. This measure consider a discretized Outcome Space (In our case, 10 cells per dimensions), and measures:

$$Ratio_{Explo} = \frac{|\mathcal{V}_d|}{|\mathcal{S}_d|}, \tag{16}$$

  where $\mathcal{V}_d$ is the ensemble of visited cells and $\mathcal{S}_d$ is the ensemble of all state cells.
- **Competence:** We estimated the **Mean Squared Error** (MSE) of the meta-policy for a set of random points sampled uniformly in the underlying environment state space and projected into the Outcome Space to define goals:

$$MSE = \frac{1}{N} \sum_{g \in \mathcal{G}} \|\mathbf{g} - \mathbf{o}\|^2_2, \tag{17}$$

  with $\mathbf{o}$ the outcome of the experiment.

G Additional Results

Explored Cells in Arm-Ball Environments In Figure 9, we have represented the locations explored in example runs of each architectures in the Arm-Ball environment. We can clearly see the poor level of performances of the RPE architecture, compared to the RGE-FI.

Correlation coefficients between representation quality and exploration performances In Table 1, we have reported the absolute of common correlations coefficients between representation quality measures and exploration performances.

Table 1: Absolute values of common correlations coefficients, between Representation quality measures an Exploration Performances for Arm-Arrow (A.) and Arm-Ball (B.) environments.

	MSE						$Ratio_{explo}$					
	Kendall $\|\tau\|$		Spearman $\|\rho\|$		Pearson $\|r\|$		Kendall $\|\tau\|$		Spearman $\|\rho\|$		Pearson $\|r\|$	
	A.	B.	A.	B.	A.	B.	A.	B.	A.	B.	A.	B.
$Q_{T\&C}$	0.62	0.62	0.82	0.83	0.79	0.78	0.57	0.47	0.78	0.68	0.71	0.67
S_{Kr}	0.45	0.36	0.58	0.51	0.60	0.57	0.02	0.25	0.04	0.35	0.30	0.49

MSE Evolution for different stationary Goal Policy In Figure 10 we have represented the evolution of Meta-Policy MSE through exploration episodes, for RGE-RFVAE and RGE-VAE, under sampling following different stationary Goal Policy.

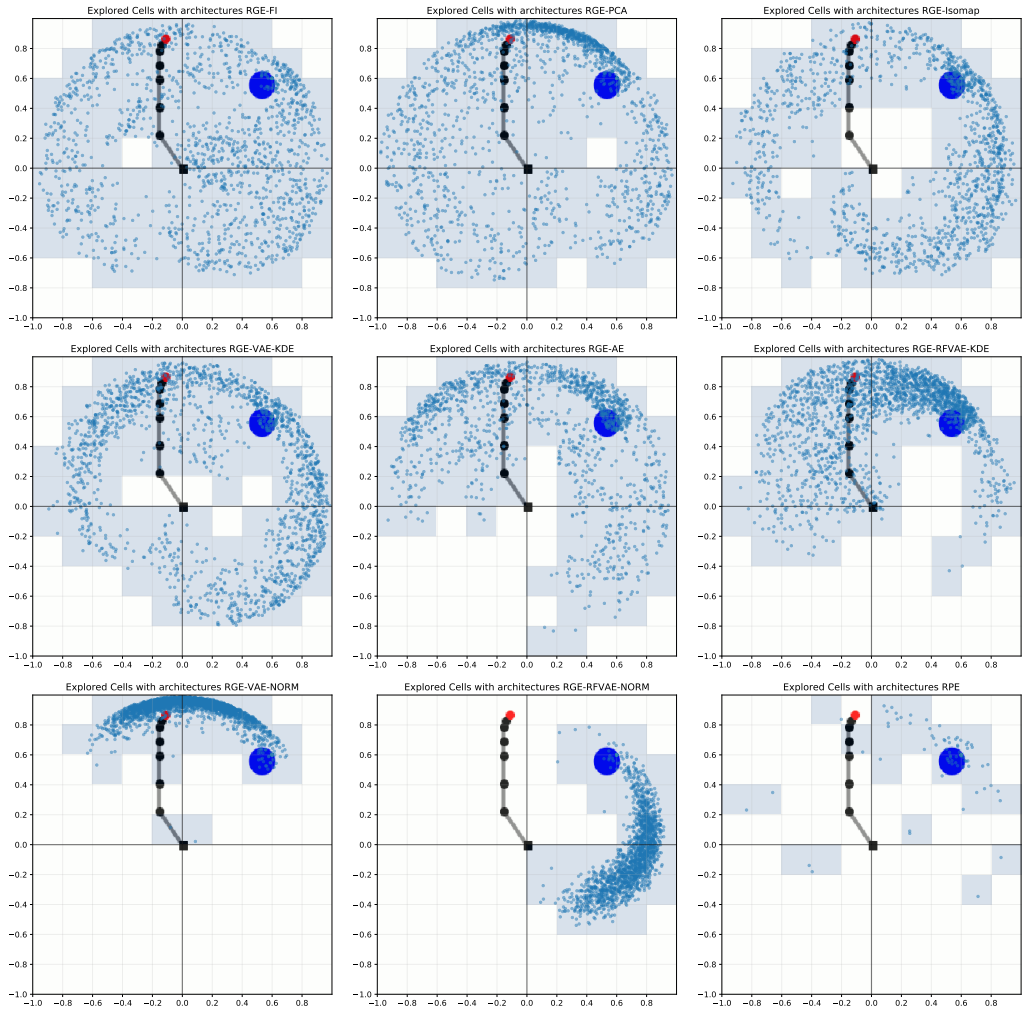

Figure 9: Explored cells in Arm-Ball environment, with different exploration algorithms (each graph shows the distribution of effects discovered in one example run for each algorithm).

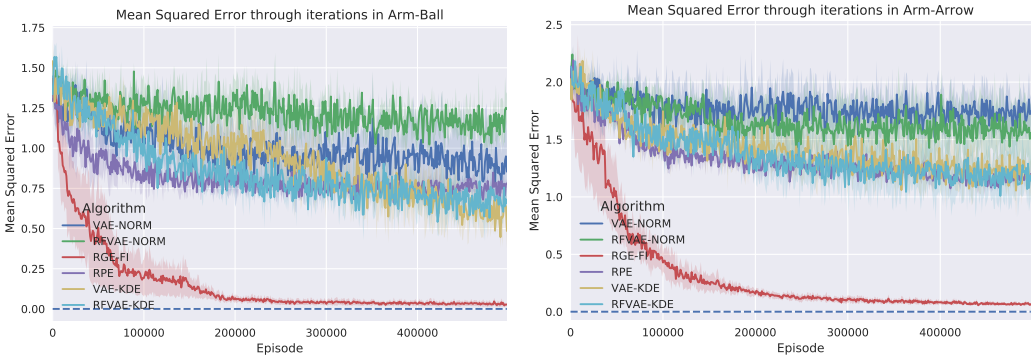

Figure 10: Mean Squared Error evolution for Variational Representations using KDE or Isotropic Gaussian prior.