Intrinsic motivation based on feature extractor distillation

Matej Pecháč and Igor Farkaš

Faculty of Mathematics, Physics and Informatics
Comenius University in Bratislava, Slovak Republic
matej.pechac@gmail.com

Abstract

Reinforcement learning can solve decision-making problems and teach an AI agent to behave in an environment according to a pre-designed reward function. However, such an approach becomes very problematic if the reward function is too sparse and the agent does not come across the reward during the environmental exploration. The solution to such a problem may be in equipping the agent with an intrinsic motivation, which will provide informed exploration, during which the agent must is likely to also encounter external reward. Novelty detection is one of the promising branches of intrinsic motivation research. We present a two variants of novelty detector based on distillation of feature extractor which is learned by contrastive loss. The results show that such an approach can achieve faster growth and higher external reward for the same training time compared to the baseline model, which implies improved exploration in a very sparse reward environment. The source code is available at `https://github.com/Iskandor/MotivationModels`

1 Introduction

The development of reinforcement learning (RL) methods has achieved much success over the last decade, since together with advances in computer vision (Krizhevsky et al., 2012; He et al., 2016), it became possible to teach agents to solve various tasks, play computer games (Mnih et al., 2013), even surpassing human players (Mnih and et al., 2015). Nevertheless, these are still concrete single tasks. Training times are very long and the agents need a lot of resources. Coping with complex (continuous) environments such as our world is still a challenge. There are several research opportunities. One is the search for more efficient learning methods. Another is hardware development, which attempts to adapt to the requirements of neural networks that are currently being used in the RL field.

The most popular approach to make RL more efficient is based on *intrinsic motivation* (IM) (Baldassarre et al., 2014). IM has a strong psychological motivation (Ryan and Deci, 2000), observed in children during development. If we want to achieve an open-ended development with artificial agents, we have to master this first step and equip them with an ability to generate their own goals and acquire new skills. Therefore, computational approaches concerned with IMs and open-ended development provide the potential in this direction leading to more intelligent systems, in particular those capable of improving their own skills and knowledge autonomously and indefinitely (Baldassarre et al., 2014).

In particular, we provide three main contributions: First, we improved the stability of the Spatio-Temporal DeepInfoMax algorithm (Anand et al., 2019) in the conditions of an incomplete dataset (online learning), which we use for feature extractor training. Second, we propose a new model for novelty detection (inspired by Burda et al. (2018b)), which serves as a source of intrinsic motivation based on the distillation of the feature extractor. Third, we hypothesize that a properly trained feature extractor can serve both as a source of features for the actor and critic and as a source of internal motivation, leading to a simplification of the model. We also managed to perform the first tests supporting this idea.

2 Related work

The concept of intrinsic (and extrinsic) motivation was first studied in psychology (Ryan and Deci, 2000), and later entered the RL literature where the first taxonomy of computational models appeared in Oudeyer and Kaplan (2009). Following this taxonomy, we can divide the concept of motivation into external and internal, depending on the mechanism that generates motivation for the agent. If the source of motivation comes from outside, we are talking about *external* motivation, and it is always associated with a particular goal in the environment. If the motivation is generated within the structures that make up the agent, it is an *internal* motivation.

Another dimension for the differentiation, extrinsic or intrinsic, is less obvious. *Extrinsic* motivations pertain to behaviors whenever an activity is done in order to attain some separable outcome. Some variability exists in this context, since these behaviors can vary in the extent to which they represent self-determination (see the details in Ryan and Deci (2000)). On the other hand, *intrinsic* motivation is defined as doing an activity for its inherent satisfactions rather than for some separable consequence (or instrumental value). It has been operationally defined in various ways, backed up by dif-

ferent psychological theories, which point to some uncertainty in what IM exactly means. Nevertheless, Baldassarre (2019) offers a solution of an operational definition of IMs as processes that can drive the acquisition of knowledge and skills in the absence of extrinsic motivations. Furthermore, he proposes (and explains why) a new term of *epistemic motivations* as a suitable substitution for intrinsic motivations.

According to the prevailing view, the computational approaches to IM can be divided into two main categories with adaptive motivations. *Knowledge-based* approach is focused on acquisition of knowledge of the world and draws on the theory of drives, theory of cognitive dissonance and optimal incongruity theory. *Competence-based* approach focuses on acquisition of skills by motivating the agent to achieve a higher level of performance in the environment, which means to acquire desired actions to achieve self-generated goals. Its psychological basis includes the theory of effectance and the theory of flow.

The knowledge-based category is commonly divided into *prediction-based* and *novelty-based* approaches. Prediction-based approaches often use a forward model (e.g. Stadie et al. (2015); Bellemare et al. (2013); Pathak et al. (2017)) or a variational autoencoder Kingma and Welling (2013) to compute the prediction error (for more details, see Burda et al. (2018a)). The novelty-based approaches monitor the state novelty and the intrinsic signal is based on its value. The first models were based on count-based approach Tang et al. (2017). This method is impractical for large or continuous state spaces and it was extended by introducing pseudo-count and neural density models Ostrovski et al. (2017); Martin et al. (2017); Machado et al. (2018). A similar method to pseudo-count was used by a random network distillation (RND) model (Burda et al., 2018b) with a lower complexity.

Contrastive learning (Chopra et al., 2005) is a machine learning technique used to learn the general features of a dataset without labels by teaching the model which data points are similar or different. Several different objective functions were proposed e.g. Noise Contrastive Estimation (NCE) (Gutmann and Hyvärinen, 2010), InfoNCE (van den Oord and andOriol Vinyals, 2018), multi-class N-pair loss (Sohn, 2016). Contrastive learning also started to be used in the field of state representation learning (Lesort et al., 2018) and is proving to be a suitable method for creating feature space (Anand et al., 2019) and also finds its use in reinforcement learning (Srinivas et al., 2020).

3 Methods

The decision making problem in the environment using RL is formalized as a Markov decision process which consists of a state space $\mathcal{S}$, action space $\mathcal{A}$, transition function $\mathcal{T}_{s,a,s'} = p(s_{t+1} = s' | s_t = s, a_t = a)$, reward function $\mathcal{R}_{sas'}$ and a discount factor γ. The main goal of the agent is to maximize the discounted return $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$ in each state, where r_t is immediate external reward at time t. Stochastic policy is defined as a state dependent probability function $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$, such that $\pi_t(s, a) = p(a_t = a | s_t = s)$ and $\sum_{a \in \mathcal{A}} \pi(s, a) = 1$ and the deterministic policy $\pi : \mathcal{S} \to \mathcal{A}$ is defined as $\pi(s) = a$.

An agent following the optimal policy π^* maximizes the expected return R. The methods searching for the optimal policy can be divided into on-policy (family of actor–critic algorithms), and off-policy (family of Q-learning algorithms) methods. Actor–critic algorithms are based on two separate modules: an *actor* approximates agent's policy π and generates actions and a *critic* estimates the state value function V^π defined as

$$V^\pi(s) = \sum_a \pi(s, a) \sum_{s'} \mathcal{T}_{s,a,s'} \left[\mathcal{R}_{s,a,s'} + \gamma V^\pi(s') \right]$$

or action-state value function Q^π defined as

$$Q^\pi(s, a) = \sum_{s'} \mathcal{T}_{s,a,s'} \left[\mathcal{R}_{s,a,s'} + \gamma V^\pi(s') \right]$$

The actor then updates its policy to maximize return R based on critic's value function estimations.

3.1 Random Network Distillation model

The RND model (Burda et al., 2018b) has two components: randomly initialized (and fixed) target network Φ_T that generates random features, and the learning network Φ_L that tries to predict them. Intrinsic motivation is computed as the prediction error, defined as

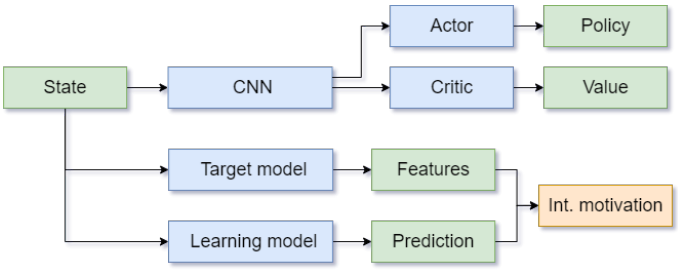
$$r_{\text{intr}} = \frac{1}{2} \| (\Phi_L(s_t) - \Phi_T(s_t)) \|^2 \qquad (1)$$

using the Euclidean norm. The model is simple and successful in the environments with sparse reward but has two serious disadvantages: (1) It is necessary to properly initialize the random network; and (2) over time, the signal of intrinsic motivation disappears due to sufficient adaptation of the learning network (a phenomenon that could be called generalization).

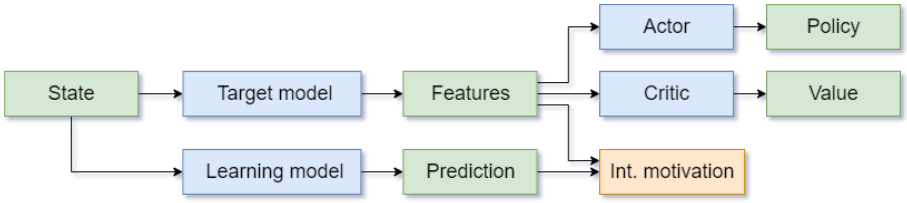
3.2 RND model extended with action

A simple extension that we propose is to add an action to the input, yielding the RNDa model. The randomly initialized target network Φ_T and the learning network Φ_L have two branches, one using convolutional neural network (CNN) to process the state and the other using multi-layerd perceptron (MLP) for action, and at the end it mixes both feature vectors into one representation. Intrinsic motivation is then computed as

$$r_{\text{intr}} = \frac{1}{8} \| (\Phi_L(s_t, a_t) - \Phi_T(s_t, a_t)) \|^2 \qquad (2)$$



(a) Feature extractor distillation separated model (FEDs)



(b) Feature extractor distillation model (FED)

Fig. 1: The schema of the feature extractor distillation models. In both variant, the target model (implemented as a convolutional neural network, CNN) transforms the raw state vector into a feature vector. The learning model (also a CNN) tries to predict the feature vector of the target model and the error between the two vectors serves as an intrinsic motivation signal. **(a)** In FEDs variant, the actor and the critic learn from a fixed CNN, whereas in **(b)** FED variant, they use the feature vector of the target model for further policy generation and value function estimation.

with an aim to increase the complexity of the input and thus prevent an early decline in intrinsic motivation. However, the key problems of the RND model are not solved.

3.3 Feature extractor distillation (FED)

We propose two models based on concept of distillation of feature extractor instead of randomly initialized network like RND. The schematic representation of both models is shown in Fig. 1.

By feature extractor we mean a module that learns to create a meaningful feature space according to a certain loss function, which is independent of other modules that consume the features of this extractor, and these modules could act properly. In other words, the error gradient is not back-propagated to the feature extractor from modules that use its feature vectors as input. By distillation we mean the transmission of the transformation represented by one neural network to another, whereby both networks will generate similar outputs for the same inputs.

In the first stage (denoted as Feature extractor distillation separated - FEDs - see Fig. 1a), we decided to design and test a model similar to the RND model, but with the addition of an objective function for target network training. Such a model has a module for learning policy and value function, consisting of a CNN that feeds two MLPs in roles of actor and critic. The goal of the second module is to generate internal motivation, which consists of target network Φ_T returning feature vectors and a learning network Φ_L that learns the same transformation and returns estimates of the feature vectors generated by target network. Both networks are CNNs. To this point, the architecture coincides with the RND model. The difference is that we added the learning rule for the target network. Following Anand et al. (2019), we use the Spatio-Temporal DeepInfoMax (ST-DIM) algorithm leveraging multi-class N-pair losses (Sohn, 2016):

$$\mathcal{L}_{\mathrm{GL}} = -\sum_{i=1}^{I}\sum_{j=1}^{J} \log \frac{\exp(g_{i,j})}{\sum_{s_t^* \in S_{\mathrm{next}}} \exp(g_{i,j})} \tag{3}$$

$$\mathcal{L}_{\mathrm{LL}} = -\sum_{i=1}^{I}\sum_{j=1}^{J} \log \frac{\exp(f_{i,j})}{\sum_{s_t^* \in S_{\mathrm{next}}} \exp(f_{i,j})} \tag{4}$$

where $f(.) = f(s_t, s_{t+1})$ and $g(.) = g(s_t, s_{t+1})$ are score functions for local-local objective $\mathcal{L}_{\mathrm{LL}}$ and global-local objective $\mathcal{L}_{\mathrm{GL}}$ respectively. Function $g_{i,j}$ is defined as non-normalized cosine similarity between transformed global feature $\Phi_T(s_t)$ and local feature $\Phi_T^{(l,i,j)}(s_{t+1})$ of intermediate layer l in Φ_T, where (i, j) is spatial location. Analogically $f_{i,j}$ is non-normalized cosine similarity between transformed local features $\Phi_T^{(l,i,j)}(s_t)$ and $\Phi_T^{(l,i,j)}(s_{t+1})$. Details of this algorithm are provided in Anand et al. (2019). S_{next} corresponds to the set of next states, (s_t, s_{t+1}) represents a pair of consecutive states, (s_t, s_{t*}) represents a pair of non-consecutive states and I, J are the width and height from output shape of intermediate convolutional layer of the target network. The resulting loss function is then

defined as

$$\mathcal{L} = \frac{1}{IJ}(\mathcal{L}_{\mathrm{GL}} + \mathcal{L}_{\mathrm{LL}}) \tag{5}$$

Following this objective function, the target network becomes a good feature extractor adapting to new states discovered by the agent. However, after initial tests, we found that the feature space formed by such an objective function tends to grow exponentially from a certain point until it eventually explodes. We provide a more detailed analysis of this problem in Section 5. The solution to this problem was to find a suitable regularization that would add to the existing loss function. In total, we tested three regularization terms:

1. Maximize entropy H to smooth logits represented by functions f and and g:

$$H_{\mathrm{GL}} = -\sum_{i=1}^{I}\sum_{j=1}^{J}\sigma(g_{i,j}) \cdot \log\sigma(g_{i,j})$$

$$H_{\mathrm{LL}} = -\sum_{i=1}^{I}\sum_{j=1}^{J}\sigma(f_{i,j}) \cdot \log\sigma(f_{i,j})$$

   where $\sigma(.)$ is standard softmax function, and the overall loss

$$\mathcal{L}_{\mathrm{reg}} = -(H_{\mathrm{GL}} + H_{\mathrm{LL}}). \tag{6}$$

2. Minimize L_2-norm of global features:

$$\mathcal{L}_{\mathrm{reg}} = \|\Phi_T(s_t)\| \tag{7}$$

3. Minimize L_2-norm of logits represented by functions f and g:

$$\mathcal{L}_{\mathrm{reg}} = p_{\mathrm{GL}} + p_{\mathrm{LL}} = \sum_{i=1}^{I}\sum_{j=1}^{J}(\|f_{i,j}\| + \|g_{i,j}\|) \tag{8}$$

According to the test results we decided to use the third option (eq. 8) and minimize the L_2-norm logits f and g. The final objective function, with scaling parameter $\beta = 0.001$, was defined as

$$\mathcal{L} = \frac{1}{IJ}(\mathcal{L}_{\mathrm{GL}} + \mathcal{L}_{\mathrm{LL}} + \beta\mathcal{L}_{\mathrm{reg}}) \tag{9}$$

In the second stage (FED, see Fig. 1b), we propose to replace CNN, which is used by the actor and the critic, with the target network, which is trained using the ST-DIM algorithm. We assume that the features it generates could be suitable for the successful functioning of the actor and the critic. This would reduce the number of networks needed and reduce the model complexity. The target network would serve both to generate intrinsic motivation and as an input for actors and critics.

In both models, the definition of intrinsic reward is the same:

$$r_{\mathrm{intr}} = \frac{1}{n}\|(\Phi_T(s_t) - \Phi_L(s_t))\|^2 \tag{10}$$

4 Experiments

For experiments, we chose Montezuma's Revenge environment for the Atari console. This is an environment with a very sparse reward, where it is almost impossible to find an optimal policy without internal motivation. The agent's goal is to overcome obstacles in individual rooms and to obtain and use items. The agent receives a reward of +1 for each increase in the score, regardless of its size. It does not receive any other reward or punishment. The state space consists of 4 consecutive frames of pixels on grey scale, so the total dimension of the state space is 4×96×96×256. The action space is discrete, consisting of 18 actions, of which 5 make sense, the others have no impact on the environment.

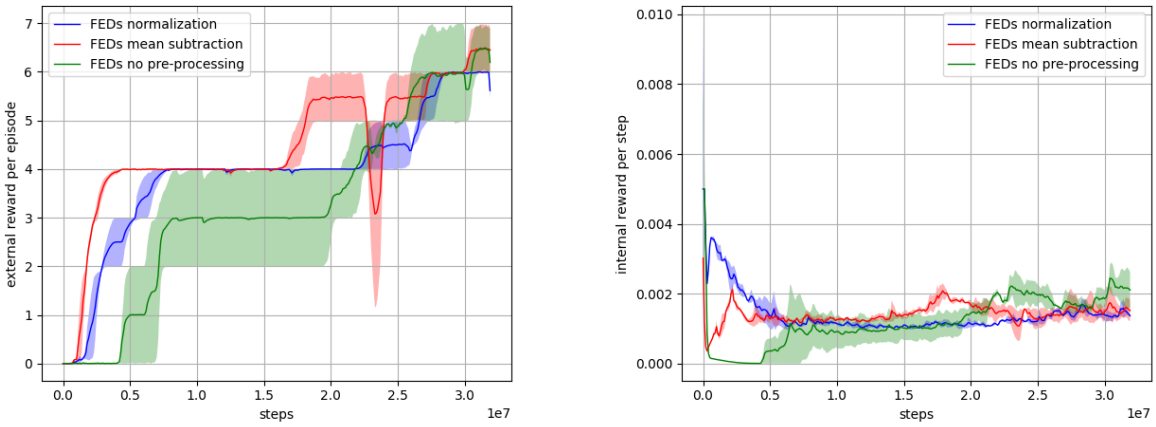
4.1 Training setup

All agents were trained using the PPO algorithm (Schulman et al., 2017) with 128 environments and we used Adam algorithm (Kingma and Ba, 2015) to optimize the parameters of all modules. The basic agent consists of an actor and a critic, which are two MLPs sharing a common CNN that processes the video input. The critic has two outputs (heads), one for estimating the value function for the external and the other for the internal reward. The discount factor γ for external reward is 0.998 and for internal reward 0.99. The motivational part consists of two CNNs (target and learning network), which receive pre-processed input (see 4.2) from 1 frame. The learning network has two more linear layers to have an increased capacity over the target network. In the case of the FED model, the motivation module contains only one CNN (learning network) and uses a feature extractor as the target network, which is also connected to the actor and the critic. Feature extractor receives on the input 4 consecutive frames. More hyper-parameters and further details of the learning process and architecture of modules can be found in our source codes.

4.2 State pre-processing study

The state before entering the motivation module of FEDs model can undergo pre-processing. In the case of the FED model, it is necessary to enter the full raw state. We tested three pre-processing methods: (1) Normalization of the state using the running mean and standard deviation, (2) Subtraction of the running mean value from the state, (3) Without pre-processing.

We trained two agents for each method in 32M steps. The test results in Tab. 1 and Fig. 2 show that state pre-processing did not have a significant effect on agent's performance (maximum reward achieved), only on the speed. This also agrees with our assumption that operations such as subtraction of the mean or normalization should be able to find the network itself trained using the contrastive loss function. Therefore it is not



(a) External reward

(b) Intrinsic reward

Fig. 2: Evolution of rewards in case of FEDs model with three state pre-processing methods.

necessary for the designer to put them into the learning process explicitly. These conclusions will still need to be confirmed by statistical analysis and show that there is no significant difference in the results achieved.

Tab. 1: Average cumulative reward per episode for all three pre-processing methods and average intrinsic reward per step.

Method	External reward	Intrinsic reward
norm.	3.60 ± 0.14	0.0016 ± 0.00008
mean sub.	4.13 ± 0.12	0.0013 ± 0.00005
none	2.31 ± 0.20	0.0009 ± 0.0001

Tab. 2: Average cumulative external reward per episode and mean intrinsic reward per step for tested agents. Compared on 128M steps and 32M steps (because the FED model has not yet been trained in 128M steps).

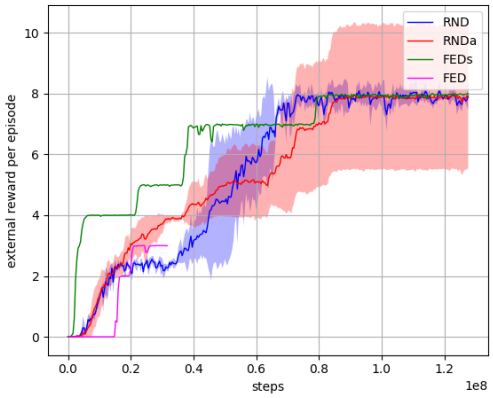
Agent (128M)	External reward	Intrinsic reward
RND	4.78	0.051
RNDa	5.15	0.014
FEDs	6.62	0.001
Agent (32M)		
RND	0.93	0.086
RNDa	1.43	0.033
FEDs	2.11	0.001
FED	1.08	0.005

4.3 Results

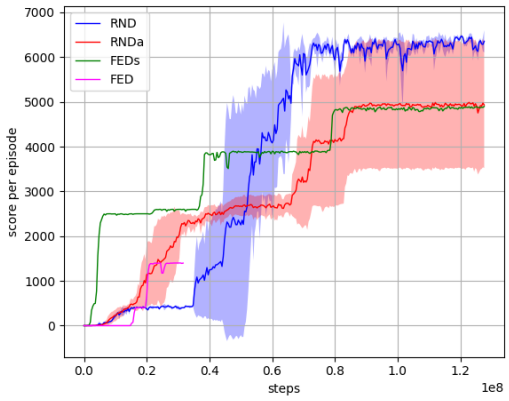
We trained three agents for the RND and RNDa model, one agent for the FEDs model and one agent for the FED model, trained in 32M steps. The results shown in Fig. 3 and Tab. 2 reveal that there is no significant difference between the RND and RNDa models, since both were able to achieve the same average reward at approximately the same time during the 128M step training. The addition of an action to the input did not delay the disappearance of the intrinsic reward (see Fig. 3c), as we had anticipated, but it brought about changes in the learned policy, where the agent discovered other sources of reward, as can be seen in Fig. 3b. The FEDs model achieved a faster increase in an external reward and also showed greater stability (although the sample is not large enough to warrant such a claim). The FED model achieved a higher intrinsic motivation compared to the FEDs model (compare Tab. 2 and Fig. 3c, which we attribute to the differences in their inputs - 4 frames vs 1 frame). For the FED model, the challenge is to learn to predict feature vectors for 4 frames, where e.g. two states may have the same last frame but the previous 3 may be different and from this point of view this input appears different for the FED model, while for FEDs, that only takes the last frame, it would appear to be the same. Therefore, the FED model explored the environment more slowly, but according to preliminary results, it does not seem to have a significant impact on the agent's overall performance. Compared to the RND model, it nevertheless achieved better results.

5 Discussion

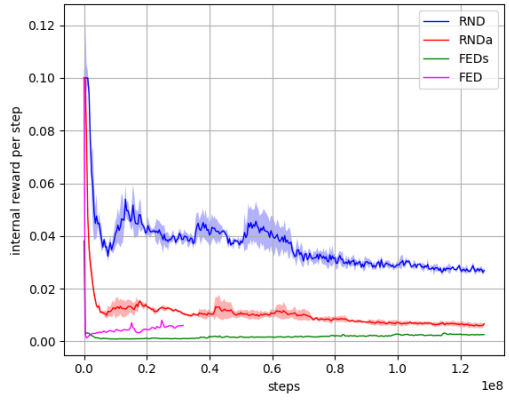
We have introduced a simple extension (RNDa) of the Random Network Distillation model and two variants of the model with intrinsic motivation derived from the



(a) External reward



(b) Score



(c) Intrinsic reward

Fig. 3: Evolution of external reward, score and intrinsic reward that agents received during learning. The agent FEDs grew faster than the RND and RNDa agents and achieved the same highest reward. In the score, we see that the RND, RNDa and FEDs each found a different policy and collected different items that led to different scores.

distillation of the feature extractor – the FEDs (which is similar to RND) and the FED model. Both proposed variants try to eliminate the identified shortcomings of the RND model – the need for good initialization and the loss of the motivational signal caused by the adaptation of the learning network. We also simplified he whole architecture with the FED model and used the ST-DIM algorithm to train the target network in both FED variants.

Our experiments revealed that if the ST-DIM algorithm works on an incomplete dataset that takes on new samples (the authors probably did not test it in such conditions), there is an instability and an exponential increase of activity in the feature space at certain moments. This is related to the use of cross-entropy loss function in its core (which does not limit the values of inputs – logits), where derivatives can reach large values and subsequently inflate the entire feature space.

We also found from observations that old states occur at the edge of the feature space and thus reach greater values than the new states that appear closer to zero. This requires further analysis, though. Therefore, we think that if a new state emerges that differs significantly from all previous ones, there may be a large growth of the entire feature space, which is further accelerated by large values of feature representations of the old states located on the edge. For this reason, we had to introduce a regularization expression into the loss function of the ST-DIM algorithm.

During the development of the model, it turned out that it is best to minimize the L_2-norm of logits that enter the cross-entropy. In addition, we tried to maximize the entropy of the distributions generating the respective logits and minimize the L_2-norm of global features. However, both described approaches failed to sufficiently stabilize the algorithm.

In the experiments, we tested the overall performance of the agents. The RNDa model did not make a big difference in agent performance compared to the RND model, which served us as a baseline. Neither did we observe the expected longer decline in internal motivation compared to RND. FEDs proved to be very promising and outperformed both RND and RNDa.

We also compared the effect of state preprocessing on the performance of the FEDs model. It turned out that the state pre-processing is not necessary since it has no significant effect on the agent's results. The FED model is proving to be a viable solution that may not grow as fast as the FEDs, but may ultimately achieve the same results. From a computational point of view, however, this is a simpler and faster solution, where it is not necessary to duplicate the CNN processing of the image input for other modules.

In the future, we plan to analyze more accurately the behavior of the feature space during training and verify the performance of FED models in other environments. We are also considering extending the input

of the FED model by an action, similar to the RNDa model. There is also room for fine-tuning the hyper-parameters, which could further improve the results.

References

Anand, A., Racah, E., Ozair, S., Bengio, Y., Côté, M., and Hjelm, R. D. (2019). Unsupervised state representation learning in atari. CoRR, abs/1906.08226.

Baldassarre, G. (2019). Intrinsic motivations and open-ended learning. arXiv:1912.13263v1 [cs.AI].

Baldassarre, G., Stafford, T., Mirolli, M., Redgrave, P., Ryan, R. M., and Barto, A. (2014). Intrinsic motivations and open-ended development in animals, humans, and robots: An overview. Frontiers in Psychology.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. (2013). The arcade learning environment: An evaluation platform for general agents. Journal of Artificial Intelligence Research, 47:253–279.

Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T., and Efros, A. A. (2018a). Large-scale study of curiosity-driven learning. arXiv:1808.04355.

Burda, Y., Edwards, H., Storkey, A., and Klimov, O. (2018b). Exploration by random network distillation. arXiv:1810.12894.

Chopra, S., Hadsell, R., and LeCun, Y. (2005). Learning a similarity metric discriminatively, with application to face verification. In IEEE Computer Society Conference on Computer Vision and Pattern Recognition, volume 1, pages 539–546 vol. 1.

Gutmann, M. and Hyvärinen, A. (2010). Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In 13th International Conference on Artificial Intelligence and Statistics, volume 9, pages 297–304.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In Conference on Computer Vision and Pattern Recognition.

Kingma, D. P. and Ba, J. L. (2015). Adam: A method for stochastic optimization. In International Conference on Learning Representations.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. arXiv:1312.6114.

Krizhevsky, A., Sutskever, I., and Hinton, G. (2012). Imagenet classification with deep convolutional neural networks. Neural Information Processing Systems, 25.

Lesort, T., Rodríguez, N. D., Goudou, J., and Filliat, D. (2018). State representation learning for control: An overview. CoRR, abs/1802.04181.

Machado, M. C., Bellemare, M. G., and Bowling, M. (2018). Count-based exploration with the successor representation. arXiv:1807.11622.

Martin, J., Sasikumar, S. N., Everitt, T., and Hutter, M. (2017). Count-based exploration in feature space for reinforcement learning. arXiv:1706.08090.

Mnih, V. and et al. (2015). Human-level control through deep reinforcement learning. Nature, 518:529–533.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing Atari with deep reinforcement learning. arXiv:1312.5602.

Ostrovski, G., Bellemare, M. G., van den Oord, A., and Munos, R. (2017). Count-based exploration with neural density models. In International Conference on Machine Learning, pages 2721–2730.

Oudeyer, P.-Y. and Kaplan, F. (2009). What is intrinsic motivation? a typology of computational approaches. Frontiers in Neurorobotics, 1:6.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. (2017). Curiosity-driven exploration by self-supervised prediction. arXiv:1705.05363.

Ryan, R. and Deci, E. (2000). Intrinsic and extrinsic motivations: Classic definitions and new directions. Contemporary Educational Psychology, 25(1):54–67.

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

Sohn, K. (2016). Improved deep metric learning with multi-class n-pair loss objective. In Advances in Neural Information Processing Systems, volume 29. Curran Associates, Inc.

Srinivas, A., Laskin, M., and Abbeel, P. (2020). CURL: contrastive unsupervised representations for reinforcement learning. CoRR, abs/2004.04136.

Stadie, B. C., Levine, S., and Abbeel, P. (2015). Incentivizing exploration in reinforcement learning with deep predictive models. arXiv:1507.00814.

Tang, H. et al. (2017). #Exploration: A study of count-based exploration for deep reinforcement learning. In Advances in Neural Information Processing Systems, pages 2753–2762.

van den Oord, A. and andOriol Vinyals, Y. L. (2018). Representation learning with contrastive predictive coding. CoRR, abs/1807.03748.