Exploration by self-supervised exploitation

Matej Pecháč[a,c], Michal Chovanec[b], Igor Farkaš[a]

[a] *Department of Applied Informatics, Comenius University, Bratislava, Slovak Republic*
[b] *Photoneo, Ltd., Bratislava, Slovak Republic*
[c] *Correspondence: matej.pechac@fmph.uniba.sk*

Abstract

Reinforcement learning can solve decision-making problems and train an agent to behave in an environment according to a predesigned reward function. However, such an approach becomes very problematic if the reward 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 is likely to also encounter external reward. Novelty detection is one of the promising branches of intrinsic motivation research. We present Self-supervised Network Distillation (SND), a class of internal motivation algorithms based on the distillation error as a novelty indicator, where the target model is trained using self-supervised learning. We adapted three existing self-supervised methods for this purpose and experimentally tested them on a set of ten environments that are considered difficult to explore. The results show that our approach achieves faster growth and higher external reward for the same training time compared to the baseline models, which implies improved exploration in a very sparse reward environment. [1] [2]

Key words: reinforcement learning; intrinsic motivation; self-supervised learning; knowledge distillation; hard exploration

1. Introduction

The development of reinforcement learning (RL) methods has achieved much success over the last decade, since together with advances in computer vision [**?** **?**], it became possible to teach agents to solve various tasks, play computer games [**?**] (see overview in [**?**]) even surpassing human players [**?**]. Nevertheless, these single tasks require very long training

[1] The source code is available at `https://github.com/Iskandor/MotivationModels` and `https://github.com/michalnand/reinforcement_learning`

[2] Video of our trained agents is available at `https://youtu.be/-vDg_r2ZetI`

times and a lot of computational resources. Coping with complex (continuous) environments such as real world is still a challenge. There are several research opportunities, one of them being 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 complex environments with sparse rewards pose a special challenge for RL approaches. The most popular computational approach to make RL more efficient is based on a concept of *intrinsic motivation* (IM) [**?**]. IM has a strong biological basis [**?** **?**] since it is observed among higher animals, especially in humans, engaging them in various activities. Intrinsic motivations appear early in life and guide the biological agents during their entire life. IM is considered one of the prerequites for open-ended (or, lifelong) learning. If we want to achieve this capacity 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 [**?** **?**].

The concept of intrinsic (and extrinsic) motivation was first studied in psychology [**?**], and later it entered the RL literature [**?** **?** **?**]. The first taxonomy of computational models appeared in [**?**] where the concept of motivation is divided into external and internal, depending on the mechanism that generates motivation for the agent. *External* motivation assumes the source of motivation coming from outside the agent 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, this implies an *internal* motivation.

Another dimension for the differentiation, extrinsic or intrinsic, is less obvious (see also [**?**]). *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 [**?**]). On the other hand, *intrinsic* motivation is defined as doing an activity for its inherent satisfaction rather than for some separable consequence (or instrumental value). It has been operationally defined in various ways, backed up by different psychological theories, which point to some uncertainty in what IM exactly means. Nevertheless, [**?**] 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, the author proposes (and explains why) a new term of *epistemic motivations* as a suitable substitution for intrinsic motivations. Despite some uncertainty, intrinsic motivation has remained a well coined term in the literature.

Intrinsic motivation is a crucial factor that helps the agent not only to remain in open-ended learning hence solving different tasks [**?**], but it also helps to solve single difficult tasks with extremely sparse rewards. In this paper, we focus on this case.

There exists a variety of approaches aiming to use IM-based signal for agent learning. Information-theoretic view on IM is well represented in the literature, involving the concepts of novelty, surprise and skill-learning. The recent review [**?**] suggests that novelty and surprise can assist the building of a hierarchy of transferable skills which abstracts dynamics and makes the exploration process more robust. In this context, abstraction is a key feature of the agent's architecture where it makes sense to introduce learning mechanisms to enforce formation of proper internal representations that lead to improved agent's performance.

Learning the proper internal representations from unlabelled input data (e.g. images) for the purpose of solving various problems is in general a useful task in machine learning. This can be achieved in various ways (related methods are mentioned in Sec. 2), including supervised end-to-end (deep) learning, self-supervised autoencoders, unsupervised feature extractors (such as contrastive divergence learning) and RL-based approaches where a certain loss function is optimized. In our work, we focus on self-supervised approaches exploiting the novelty detection signal in feature domain to enhance RL agent's exploration.

The paper is organized as follows. In the remaining part of Sec. 1 we present the original contribution of this work. Section 2 contains related work. Section 3 describes the methods used in this work. Section 4 explains in detail all experiments performed. Section 5 concludes the paper with the discussion.

1.1. The paper contribution

We introduced and tested a class of motivational models based on the exploitation of the distillation error as novelty detection. The first such model was the Random Network Distillation model [**?**], which became the basis of our models. However, random distilled features are not the best representation, since this leads to stacking or slow convergence. Instead of distilling random features, we proposed to distil the self-supervised latent space representations.

The overall concept is shown in Fig. 1. Our method uses two models, one providing target features z_t^{T}, called the target model $\Phi^{\mathrm{T}}(s_t)$, and the learned model $\Phi^{\mathrm{L}}(s_t)$, providing features z_t^{L}. Both models use the state (observation) s_t at time step t as their input. The learned model attempts to imitate the target model, and their difference is used for internal motivation as provided in [**?**]. We asked ourselves the following questions: Are the features provided from the original random model $\Phi^{\mathrm{T}}(s_t)$ sufficient? Can we provide better features? In Random Network Distillation paper [**?**], the orthogonal

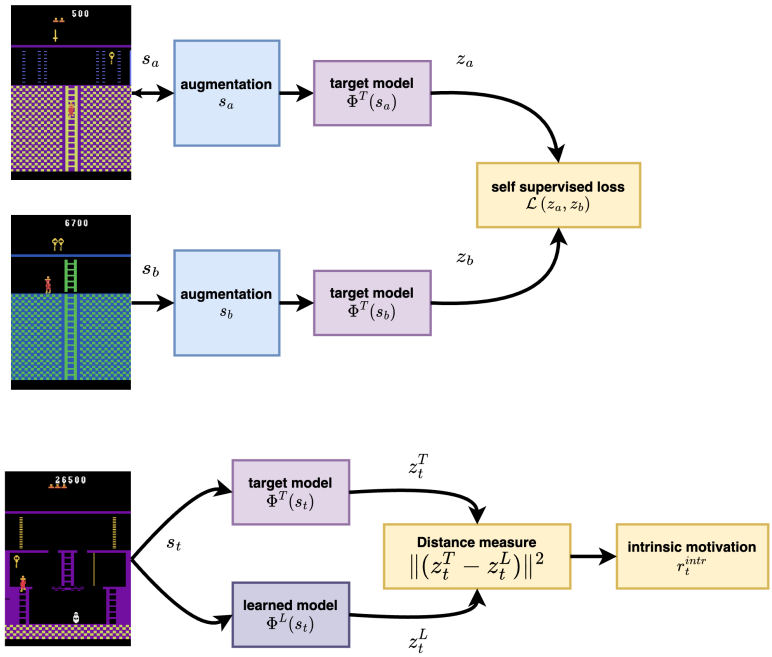


Figure 1: Self-supervised network distillation (SND) principle. The proposed method consists of two main parts. *Top:* Self-supervised learning of the suitable features for the target model. *Bottom:* Calculation of the intrinsic reward by target model distillation, using the squared Euclidean distance between the models' outputs.

weight initialization method was used to set $\Phi^{\mathrm{T}}(s_t)$ and spread out z_t^{T} across the state space, which provides sustainable intrinsic reward. We used self-supervised regularization of Φ^{T} to provide features with higher variance, and better sensitivity for novelty detection. The states are randomly sampled from the buffer and used for learning Φ^{T}. For sampling the states, we used two approaches

1. sample one state with two different augmentations
2. sample two consecutive states

Simultaneously, the distillation proceeds with the learning model Φ^{L} using MSE loss. All models are trained in the same loop as the policy and the value models. We experimented with different self-supervised losses (MSE, ST-DIM [**?**] and VICReg [**?**]), covering both contrastive and non-contrastive approaches. With these methods, we were able to solve hard exploration seeds for Procgen and the complete first level in infamous Atari game Montezuma's revenge.

2. Related work

According to the prevailing view, the approaches to IM can be divided into two main categories with adaptive motivations. *Knowledge-based* ap-

proach is focused on acquisition of knowledge of the world and it 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 focusing on exploration can be divided into *prediction-based*, *novelty-based* and *information-based* approaches [**?**]. Prediction-based approaches use prediction error as an intrinsic reward signal. The source of error can be a forward model (e.g. [**?** **?** **?**]), a generative model [**?**] (e.g. based on a variational auto-encoder [**?**]) or disagreement in learned world model [**?**]. Exploration with Mutual Information (EMI) [**?**] extracts predictive signals that can be used to guide exploration based on forward prediction in the representation space. Model-Based Active eXploration (MAX) [**?**] uses an ensemble of forward models to plan observing novel events.

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 [**?**]. This method is impractical for large or continuous state spaces and it was extended by introducing pseudo-count and neural density models [**?** **?** **?**]. A similar method to pseudo-count was used by a random network distillation (RND) model [**?**] with a lower complexity. Never-give-up framework [**?**] learns intrinsic rewards composed of episodic and life-long state novelty (which is detected by RND model).

Information approaches use quantities from information theory [**?**], such as information gain, mutual information, entropy and try to maximize the information obtained by the agent from the environment. Variational Information Maximizing Exploration (VIME) [**?**] approximates the environment dynamics, uses the information gain of the learned dynamics model as intrinsic rewards. Random Encoders for Efficient Exploration (RE3) [**?**] is an exploration method that utilizes state entropy as an intrinsic reward. For more details we recommend surveys [**?** **?** **?**].

Self-supervised learning is a paradigm of machine learning, when the agent does not have any labeling of the data, but generates it on its own. The goal is to use the information in the data itself and prepare the model to perform another task. Self-supervised learning also started to be used in the field of state representation learning [**?**] it is proving to be a suitable method for creating the feature space [**?**] and has also found its use in reinforcement learning [**?** **?**]. Contrastive learning [**?**] is a method of self-supervised learning used to learn the general features by teaching the model which data points are similar or different. Several different objective functions were proposed, e.g. Noise Contrastive Estimation (NCE) [**?**], InfoNCE [**?**], or multi-class N-pair loss [**?**]. Another method of self-supervised learning is based on regularization (non-contrastive methods).

In this case, the model sees only positive examples and generates a feature space based on various regularization losses like invariance and covariance loss in the case of Barlow Twins model [**?**], triple variance-invariance-covariance losses in VICReg model [**?**] or other priors like proportionality, variability, slowness principle, or repeatability [**?**]. Bootstrap Your Own Laten (BYOL) [**?**] relies on two neural networks, referred to as online and target networks, that interact and learn from each other.

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), e.g. [**?**] and off-policy methods (family of Q-learning algorithms), e.g. [**?**]. Actor–critic algorithms are based on two separate modules: an *actor* generates actions following the agent's policy π 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 the state-action 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.

In high-dimensional tasks, when one cannot use the Bellman equation, the common approach is to use function approximators (deep convolutional neural networks) for estimating the critic and the actor.

3.1. Intrinsic motivation and exploration

During the learning process, the agent must explore the environment to encounter an external reward and learn to maximize it. This can be ensured by adding noise to the actions, if the policy is deterministic, or it is already its property, if the policy is stochastic. In both cases, we say that these are

uninformed environmental exploration strategies. The problem arises if the external reward is very sparse and the agent cannot use these strategies to find the sources of reward. In such a case, it is advantageous to use informed strategies, which include the introduction of intrinsic motivation.

In the context of RL, intrinsic motivation can be realized in various ways, but most often it is a new reward signal r_t^{intr} scaled by parameter η, which is generated by the motivational part of the model (we refer to it as the motivational module) and is added to the external reward r_t^{ext}

$$r_t = r_t^{\text{ext}} + \eta\, r_t^{\text{intr}} \tag{1}$$

The goal of introducing such a new reward signal is to provide the agent with a source of information that is absent from the environment when the reward is sparse, and thus facilitate the exploration of the environment and the search for an external reward.

3.2. Intrinsic motivation based on distillation error

In this class of methods the motivation module has two components: the target model Φ^{T} that generates features (typically as a kind of feature extractor), and the learning network Φ^{L} that tries to replicate them. This process is called knowledge distillation. Intrinsic motivation, expressed as an intrinsic reward, is computed as the distillation error

$$r_t^{\text{intr}} = \|(\Phi^{\text{L}}(s_t) - \Phi^{\text{T}}(s_t))\|^2. \tag{2}$$

It is assumed that the learning network will be able to more easily replicate feature vectors for states it has seen multiple times, while new states will induce a large distillation error. The RND [**?**] model shown in Fig. 2 is a representative of this type of IM. It is simple and successful in the environments with sparse reward but has two serious drawbacks: (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).

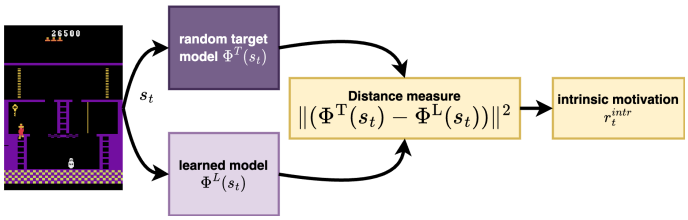


Figure 2: The basic principle of generating an exploration signal in random network distillation.

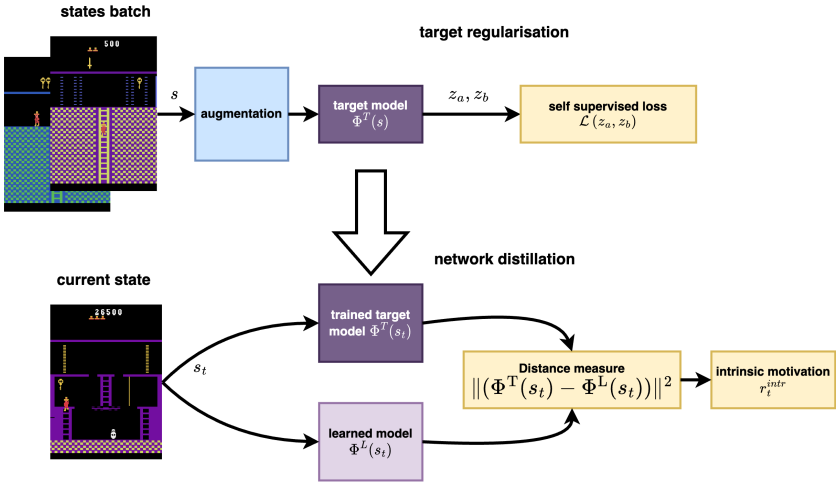


Figure 3: The basic principle of generating an exploration signal in the regularized target model, followed by RND.

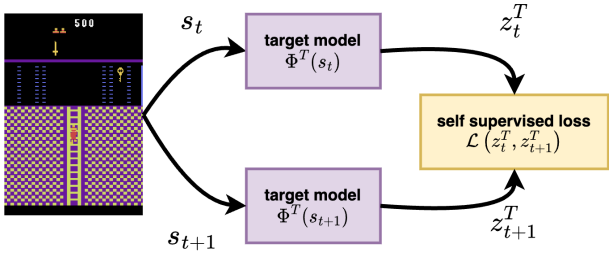


Figure 4: Training of the SND target model using two consecutive states and the self-supervised learning algorithm.

3.3. Self-supervised Network Distillation

We modified the concept of distillation of randomly initialized static network RND [**?**] and instead we distilled a network that learns continuously using self-supervised algorithms. We denote the methods SND. The architecture of such model consists of a target model Φ^{T} and a learned model Φ^{L}, but with the essential difference that the network generating the target feature vectors (target model) is learned. The schematic representation of proposed approach is shown in Fig. 3. In order to be able to use the trained target model as a suitable source of target feature vectors for the learning network, it is necessary that it fulfills the following conditions:

1. Two identical states must map to the same feature vector.
2. Two similar (e.g. successive states) are mapped on two similar feature vectors, e.g. their L_2 distance is small.
3. Two different states are mapped on two different feature vectors, e.g. their L_2 distance is large.

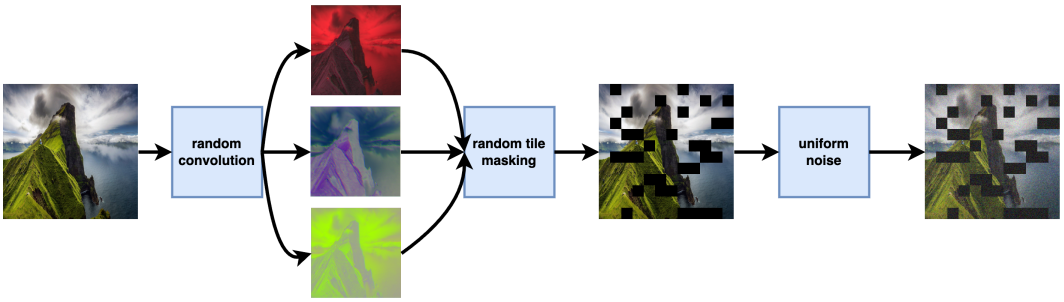


Figure 5: The scheme of the state augmentation pipeline.

The feature space formed in this way can be distilled and then this process can be used as a source of internal motivation, because the new states will have different feature vectors than the states seen by the agent so far. We introduced three methods for forming a feature space that satisfies the above conditions. All three methods are based on self-supervised learning.

SND-V method (vanilla SND) uses a contrastive approach. First the two augmented batches of states S, S' are sampled and the corresponding feature batches Z, Z' are computed as $\Phi^{\mathrm{T}}(s) = z$, where $s \in S, z \in Z$. The sampling process takes with $p = 0.5$ the same states. For the same states, we set the target distance to $\tau_i = 0$, and for different states we set the distance to $\tau_i = 1$. We experimented with three augmentation schemes :

1. uniform noise only, from the range $\langle -0.2, 0.2 \rangle$
2. random tile masking + uniform noise, tiles sizes $2, 4, 8, 12, 16$
3. random convolution filter + random tile masking + uniform noise

Uniform noise was used for each state pixel, remaining augmentations with $p = 0.5$. The whole pipeline is shown in Fig. 5. The idea of tiles masking can be supported by recently proposed self-supervised loss [**?**]. Random convolution was successfully used in [**?**], for a PPO agent in Procgen environment. And finally, noise is a common augmentation process.

The regularisation loss is defined as

$$\mathcal{L} = \sum_i (\tau_i - \|Z_i - Z'_i\|_2^2)^2 \tag{3}$$

where $\|.\|_2^2$ is the squared Euclidean distance between $i-$th feature vectors Z_i and Z'_i. We also experimented with the loss function defined as

$$\mathcal{L} = \begin{cases} \|Z_i - Z'_i\|_2^2 & \text{if } \tau_i = 0 \\ \max(1 - \|Z_i - Z'_i\|_2^2; 0) & \text{otherwise} \end{cases} \tag{4}$$

This loss function pulls the feature vectors of similar states (when $\tau = 0$) to one another by penalizing their squared distance. And on the contrary, it

sets apart the feature vector from one another, up to a certain limit when their distance exceeds one. However, we did not find any benefits of this loss.

SND-STD method uses the Spatio-Temporal DeepInfoMax (ST-DIM) algorithm [**?**] (the simple diagram can be found in Fig. 4) leveraging multi-class N-pair losses [**?**]:

$$\mathcal{L}_{\mathrm{GL}} = -\sum_{i=1}^{I}\sum_{j=1}^{J}\log\frac{\exp(g_{i,j})}{\sum_{s_t^*\in S_{\mathrm{next}}}\exp(g_{i,j})} \tag{5}$$

$$\mathcal{L}_{\mathrm{LL}} = -\sum_{i=1}^{I}\sum_{j=1}^{J}\log\frac{\exp(f_{i,j})}{\sum_{s_t^*\in S_{\mathrm{next}}}\exp(f_{i,j})} \tag{6}$$

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 the unnormalized cosine similarity between transformed global features $\Phi^{\mathrm{T}}(s_t)$ and the local features $\Phi^{\mathrm{T}}_{(l,i,j)}(s_{t+1})$ of the intermediate layer l in Φ^{T}, where (i, j) is the spatial location. Analogically $f_{i,j}$ is the unnormalized cosine similarity between transformed local features $\Phi^{\mathrm{T}}_{(l,i,j)}(s_t)$ and $\Phi^{\mathrm{T}}_{(l,i,j)}(s_{t+1})$. The details of this algorithm are provided in [**?**]. 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 the height from output shape of intermediate convolutional layer of the target model. The resulting loss function is then defined as

$$\mathcal{L} = \frac{1}{IJ}(\mathcal{L}_{\mathrm{GL}} + \mathcal{L}_{\mathrm{LL}}) \tag{7}$$

Following this objective function, the target model 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. We decided to minimize L_2-norm of logits represented by functions f and g:

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

Finally, we added one more regularization term $\mathcal{L}_v$ that maximizes the standard deviation σ of the feature vector components and thus ensures that all dimensions of the feature space are used. The analysis section provides a more detailed justification for the introduction of the given term:

$$\mathcal{L}_v = -\sigma(\Phi^{\mathrm{T}}(s_t)) \tag{9}$$

The final objective function, with the scaling parameters $\beta_1 = \beta_2 = 0.0001$ (found experimentally), was defined as

$$\mathcal{L} = \frac{1}{IJ}(\mathcal{L}_{\text{GL}} + \mathcal{L}_{\text{LL}} + \beta_1 \mathcal{L}_n) + \beta_2 \mathcal{L}_v \tag{10}$$

SND-VIC method is based on VICReg algorithm [**?**]. The regularization function consists of three terms: the invariant term, which brings the feature vectors closer to each other, the variance term, which ensures that the feature vectors within one batch have different values, and the covariance term, which ensures the decorrelation of the feature vectors and prevents information collapse. The original method does not need any negative samples, it only takes the input, creates two augmented versions, and their feature vectors are updated using the mentioned terms of the regularization function. Our version uses the state s_t and its successor s_{t+1} (the same as ST-DIM, the simple diagram can be found in Fig. 4) instead of two augmentations of the same state. The variance regularization term $\mathcal{L}_{v(Z)}$ is defined as a hinge function on the standard deviation of the features along the batch dimension

$$\mathcal{L}_{v(Z)} = \frac{1}{d} \sum_{j=1}^{d} \max(0; \tau - \sigma(Z_j)) \tag{11}$$

where d is the dimensionality of the feature space, Z_j is j-th feature vector from the batch Z, σ is the actual standard deviation and $\tau = 1$ is a constant target value for the standard deviation. The covariance regularization term $\mathcal{L}_{c(Z)}$ is defined as the sum of the squared off-diagonal coefficients of the covariance matrix $C(Z)$

$$\mathcal{L}_{c(Z)} = \frac{1}{d} \sum_{i \neq j} [C(Z)]_{i,j}^2 \tag{12}$$

The invariance criterion $\mathcal{L}_{s(Z,Z')}$ between two batches Z and Z' is defined as the mean-squared Euclidean distance between each pair of feature vectors

$$\mathcal{L}_{s(Z,Z')} = \frac{1}{d} \sum_{i=1}^{d} \|Z_i - Z'_i\|_2^2 \tag{13}$$

The overall loss $\mathcal{L}$ then takes the form

$$\mathcal{L} = \lambda \mathcal{L}_{s(Z,Z')} + \mu \left[\mathcal{L}_{v(Z)} + \mathcal{L}_{v(Z')}\right] + \nu \left[\mathcal{L}_{c(Z)} + \mathcal{L}_{c(Z')}\right] \tag{14}$$

where the scaling parameters are set to $\lambda = 1$, $\mu = 1$ and $\nu = 1/25$.

4. Experiments

All together, we tested our methods on 10 environments (Atari and Procgen) that are considered difficult for exploration. These include 6 Atari environments: Montezuma's Revenge, Gravitar, Venture, Private eye, Pitfall, Solaris. 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 is represented by 4 consecutive frames of pixels on grey scale, so the dimensionality of the state representation is $4\times96\times96\times256$. The action space is discrete, consisting of 18 actions, of which only some make sense (depending on the environment), the other actions have no impact on the environment.

We also tested 4 Procgen environments: Coinrun, Caveflyer, Jumper and Climber. Procgen is a set of procedurally generated environments, designed primarily for testing agent's generalisation [**?**]. The paper shows several problems for generalisation in RL, requiring special training and a huge amount of samples. For our purpose, interesting findings are provided in Appendix B.1 in [**?**]. For several seeds, the baseline agent was not able to reach a non-zero score. Those seeds lead to hard exploration environments, with only a single reward at the end. Together with fast run of these environments (thousands of FPS on single CPU core), makes Procgen good candidate for our experiments. The state is represented by RGB color images, with the size 64×64 pixels. The action space is discrete, consisting of 15 actions.

Preliminary experiments were performed in [**?**].

4.1. Training setup

We ran 9 simulations for each environment, taking 128M steps for Atari and 64M steps for Procgen games. Before the main training, we tried 3 hand-selected settings of hyperparameters for individual motivational models (mainly the scaling of the motivational signal, or regularization terms) and we always chose the one that had the best results. These short probes lasted 32M (Atari) or 16M (Procgen) steps and consisted of 2 to 3 simulations.

All agents were trained with the PPO algorithm [**?**] using Adam algorithm [**?**] to optimize the parameters of all modules. The basic agent consists of an actor and a critic, which are two multi-layer perceptrons sharing a common convolutional neural network (CNN) that processes the video input. The critic has two outputs (heads), one for estimating the value function for the external reward and the other for the internal reward. We used the orthogonal weight initialisation with a magnitude $\sqrt{2}$. Model architectures are presented in Figures 6 to 8. The motivational module consists of two CNNs (the target and the learning network), which receive input from

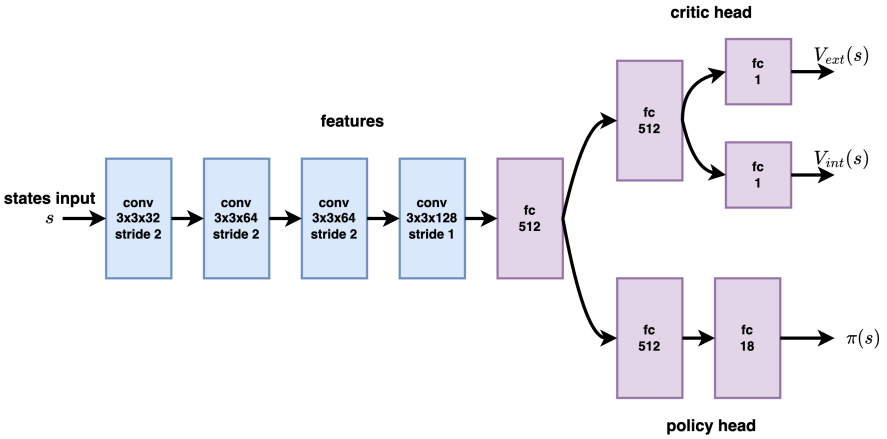


Figure 6: The PPO agent model architecture.

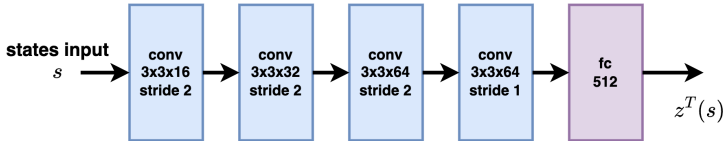


Figure 7: The target model architecture.

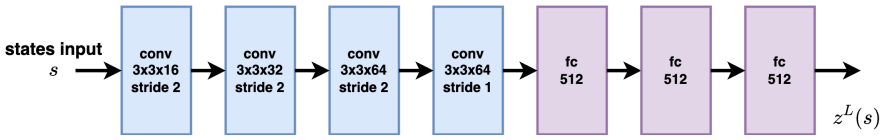


Figure 8: The learning model architecture.

a single frame. The learning network has two more linear layers to have an increased capacity over the target model.

We followed [**?**] for setting the hyperparameters, to have more comparable results. We ran 128 parallel environments. For Atari we used 1M samples for each environment (total 128M frames), for Procgen we used 0.5M samples for each environment (total 64M frames). In Atari experiments we used gray scale downsampled 4 frames stacked observation. For Procgen we used 2 frames stacking, and fully RGB colored observation. The intrinsic motivation modules used no frame stacking, a single gray scale image for Atari environments and a single RGB image for Procgen. The summary of all environment hyperparameters is in Table 1. The discount factors were set to $\gamma^{\text{ext}} = 0.998$ for external reward and $\gamma^{\text{intr}} = 0.99$ for intrinsic reward. We found the importance of intrinsic reward scaling, the best result were achieved for $\eta = 0.5$. The learning rate for all models was set to 0.0001 with Adam optimizer. Actor and Critic models used ReLU, and motivation models worked best with ELU activation function. We also find the deeper

model with 3×3 convolutions works better then standart Atari model using 8×8 or 4×4 convolutions in [**?**]. We retrained RND models to obtain comparable results and find faster convergence. The summary of PPO agent's hyperparameters is in Table 2. More hyperparameters and further details of the learning process and the architectures of modules can be found in our source codes.

Table 1: Environment hyperparameters

Hyperparameter	Atari	Procgen
Observation downsampling	96×96	64×64
Frame stacking	4	2
State shape for PPO	4×96×96	6×64×64
State shape for IM modules	1×96×96	3×64×64
Parallel environments count	128	128
State normalisation	$s/255$	$s/255$
Samples per environment	1M	0.5M

Table 2: Agent's hyperparameters

Hyperparameter	Value
PPO model learning rate	0.0001
Target model Φ^{T} learning rate	0.0001
Learned model Φ^{L} learning rate	0.0001
Discount factor γ^{ext}	0.998
Discount factor γ^{intr}	0.99
Advantages ext coefficient	2.0
Advantages intr coefficient	1.0
Intrinsic reward scaling	0.5
Rollout length	128
Number of optimization epochs	4
Entropy coefficient	0.001
Epsilon clipping	0.1
Gradient norm clipping	0.5
GAE λ coefficient	0.95
Optimizer	Adam
Weight initialisation	orthogonal

4.2. State preprocessing

The state before entering the motivation module of SND model can undergo preprocessing. We tested three preprocessing methods:

1. State normalization using the running mean and standard deviation,
2. Subtraction of the running mean value from the state,
3. No preprocessing.

We performed two training runs for each preprocessing method in 32M steps on Montezuma's Revenge environment. For testing we used SND-STD model. Table 3 demonstrates that the state preprocessing did not have a significant effect on agent's performance (maximum reward achieved), only on the speed of learning. This also agrees with our assumption that operations

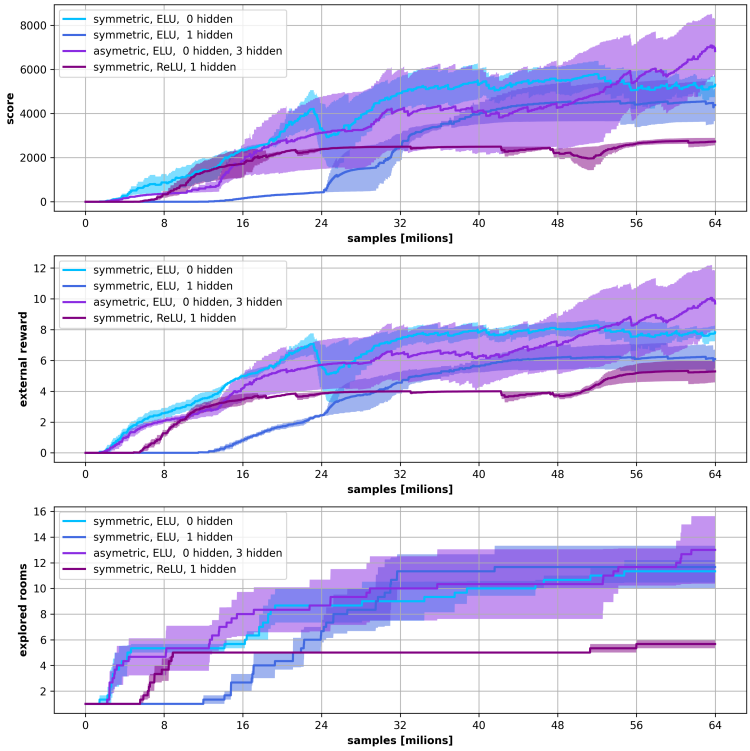


Figure 9: Agent's performance based on various learned model architectures, evaluated in terms of the overall score, external reward obtained and the number of rooms explored.

such as subtraction of the mean or normalization should be able to find the network itself trained using the self-supervised loss function. Therefore it is not necessary for the designer to put them into the learning process explicitly. These conclusions will still need to be confirmed by statistical analysis. RND used mean subtraction and SND-V, SND-STD together with SND-VIC did not use input state preprocessing.

Table 3: Average cumulative reward (with standard deviation) per episode for all 3 preprocessing methods and maximal reward achieved by the agents.

Method	Average reward	Max. reward
normalization	3.60 ± 0.14	7
mean subtraction	4.13 ± 0.12	7
none	2.31 ± 0.20	7

4.3. Results

We processed several quick experiments on Montezuma's Revenge to explore an optimal setup. First we have to test the optimal architecture of Φ^{T} and Φ^{L} models. We experimented with 4 architectures:

1. identical models, one fully connected output layer, ELU activations

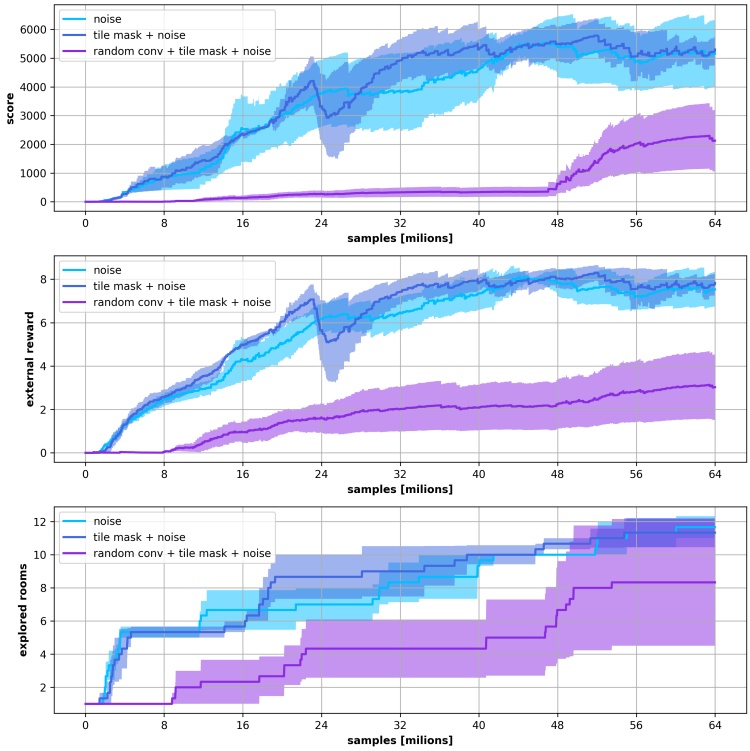


Figure 10: Agent performance for different state augmentations, evaluated in terms of the overall score, external reward obtained and the number of rooms explored.

2. identical models, two fully connected layers, ELU activations
3. identical models, one fully connected layer, ReLU activations
4. asymmetric models, three fully connected layers for Φ^{L}, none for Φ^{T}, ELU activations fixed to fully connected convention

Results of different model architectures are in Figure 9. The best result was achieved for the asymmetric architecture.

Next, we tested the effect of different augmentations. We considered three scenarios:

1. uniform noise, $\langle -0.2, 0.2 \rangle$
2. uniform noise, random tiles masking (tiles with sizes 1, 2, 4, 8, 12, 16)
3. uniform noise, random tiles masking, random convolution filter apply

Noise augmentation is commonly used in supervised image training. Tile masking forces the model to reconstruct non-complete information. Random convolution filter helps the model to learn to focus on informative features, not on texture colors. The results of different state augmentations are in Figure 10, revealing that the third scenario worked best. We hypothesise that it is caused by shallow target model, which is not able to learn sufficient

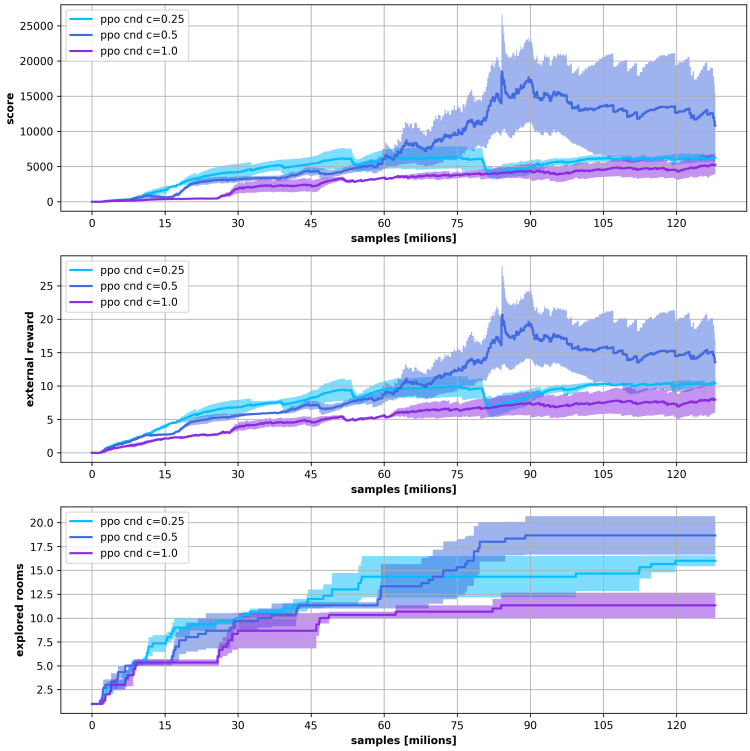
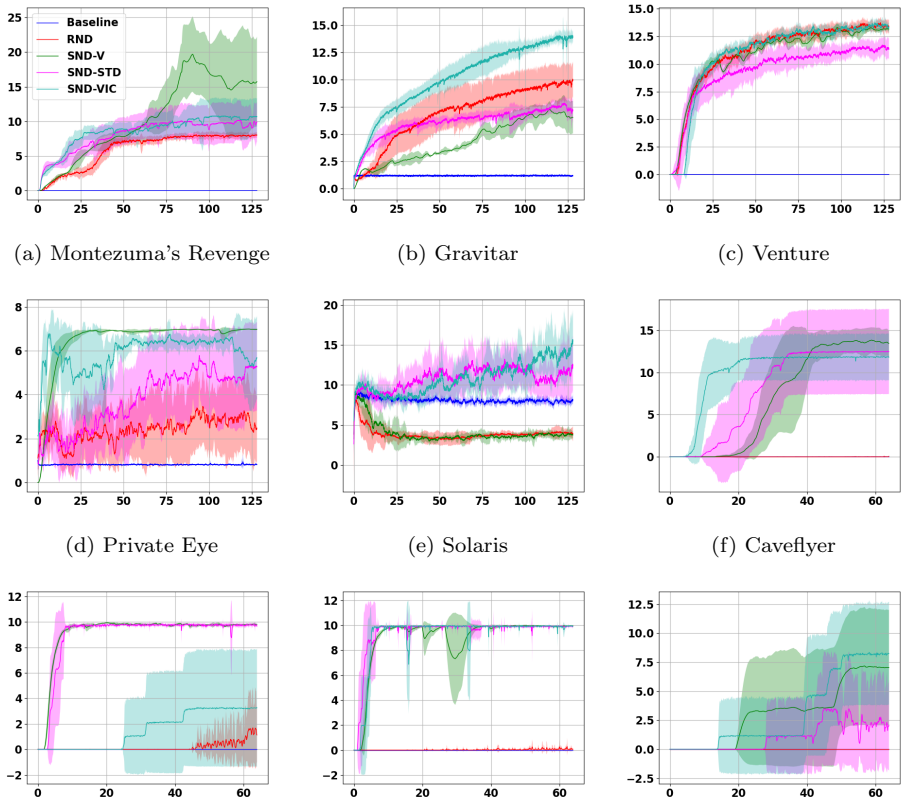


Figure 11: Agent's performance for different intrinsic reward scaling methods, evaluated in terms of the overall score, external reward obtained and the number of rooms explored.

transformation invariants. The models using noise or tile masking performed in a similar way.

Finally, we tested intrinsic reward scaling. Low value can lead to stacking the agent into non-exploring policy. High value can prevent the agent from collecting extrinsic rewards, or make it too sensitive to small unimportant changes, both causing instability. We tested three values 0.25, 0.5 and 1.0. Figure 11 shows that the best score is achieved with 0.5 reward scaling value. However, some environments provide better results after fine-tuning to 0.25.

Figure 12 captures the cumulative external reward per episode and the standard deviation of the tested models in 9 different environments. In Table 4, these indicators are then averaged over the number of episodes. Table 5 shows the maximum achieved score for Atari environments, which is often used for model comparison (although we would like to emphasize that the agent never receives this score as a reward and therefore it is not its goal to maximize it). Of the tested environments, Pitfall game exceeded the capabilities of all tested algorithms, since none of them achieved a single reward point. In the remaining 9 environments, the best results were achieved with the models based on SND motivation, while in 8 cases it was with a significant lead over the existing algorithms (in Venture environment the



(a) Montezuma's Revenge (b) Gravitar (c) Venture

(d) Private Eye (e) Solaris (f) Caveflyer

(g) Coinrun (h) Jumper (i) Climber

Figure 12: The cumulative external reward per episode (with the standard deviation) received by the agent from the tested environment. We omitted the graph for the Pitfall environment, where no algorithm was successful and all achieved zero reward. The horizontal axis shows the number of steps in millions, the vertical axis refers the external reward.

results were almost the same as those of the RND model). When evaluating the score, the SND models achieved the highest score in 5 Atari environments (with an exception of the already mentioned Pitfall) and in 3 cases (Montezuma's Revenge, Gravitar, Private Eye) it was significantly higher than the compared models.

4.4. Analysis of results

We can visualise learned feature vectors Z in 2D using t-SNE method [**?**]. Figure 13 shows resulted features of trained Φ^{T} on Atari Montezuma's revenge. The randomly initialised trained network (the same as in [**?**] in Figure 13a) can well distinguish between different rooms, however within the room the variance is low, pointing to the lack of exploration abilities. On the other hand, self-supervised regularized target model in Figure 13

Table 4: Average cumulative external reward per episode for tested models. The best model for each environment is shown in bold face.

	Baseline	RND	SND-V	SND-STD	SND-VIC
Montezuma	0.00 ± 0.00	5.33 ± 0.23	**10.59 ± 1.99**	7.76 ± 1.73	8.45 ± 1.12
Gravitar	1.19 ± 0.00	6.63 ± 1.55	4.38 ± 0.46	5.89 ± 0.43	**10.05 ± 0.66**
Venture	0.00 ± 0.00	11.18 ± 0.42	10.95 ± 0.14	9.54 ± 0.90	**11.36 ± 0.37**
Private Eye	0.81 ± 0.01	2.41 ± 0.95	**6.59 ± 0.14**	3.79 ± 1.24	5.93 ± 0.47
Pitfall	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
Solaris	8.08 ± 0.15	3.84 ± 0.25	3.96 ± 0.41	**11.61 ± 1.12**	10.85 ± 1.20
Caveflyer	0.00 ± 0.00	0.00 ± 0.00	7.28 ± 1.62	10.86 ± 4.37	**11.14 ± 2.35**
Coinrun	0.00 ± 0.00	0.25 ± 0.50	**9.40 ± 0.05**	9.40 ± 0.07	2.55 ± 3.63
Jumper	0.00 ± 0.00	0.03 ± 0.02	9.22 ± 0.21	9.76 ± 0.04	**9.76 ± 0.03**
Climber	0.00 ± 0.00	0.00 ± 0.00	3.32 ± 3.04	1.48 ± 2.40	**4.93 ± 2.88**

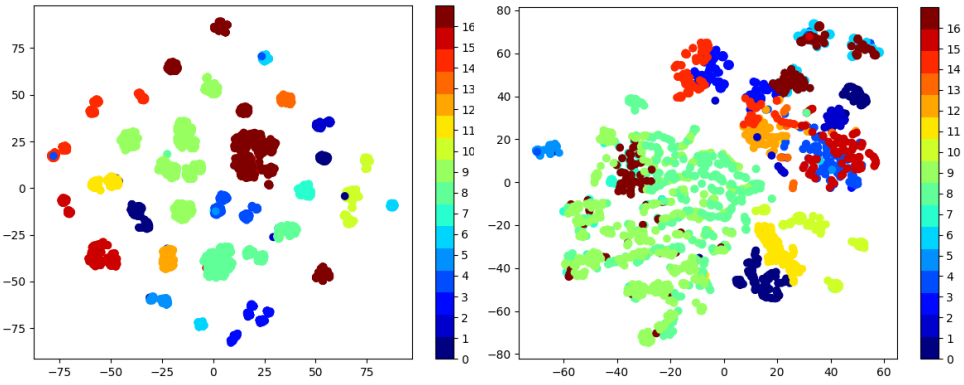
Table 5: Average maximal score reached by tested models on Atari environments. The best model for each environment is shown in bold face.

	Baseline	RND	SND-V	SND-STD	SND-VIC
Montezuma	400	6689	**21565**	7212	7838
Gravitar	2611	5600	2741	4643	**6712**
Venture	22	2167	1787	2138	**2188**
Private Eye	14870	14996	4213	15089	**17313**
Pitfall	0	0	0	0	0
Solaris	12344	10667	11582	**12460**	11865

provides much larger variance of features, which provides more sensitive novelty detection signal.

The main goal of the analysis was to find out the differences (not only visually) between the individual spaces of features, to describe them with some quantities and to find a possible connection with the performance of the algorithm and the mentioned quantities. From the set of examined models for one environment, we always selected the model with the highest obtained reward and generated 10,000 samples of input states by running it in the environment. Subsequently, each model generated feature vectors for a sample of previously collected input states. Thus, we obtained feature space samples Z of each model. Then, using principal component analysis (PCA), we found the linear envelope of the high-dimensional manifold that forms the feature space. We examined the mean value and especially the variance of the feature vectors and also the eigenvalues obtained using PCA, which at least indicate something about the basic shape of the feature space (i.e. the sizes of the individual dimensions).

The results of this analysis are shown in Fig. 14 and Tab. **??**. For the evaluation, we decided to use the following parameters: mean value and standard deviation of the L_2-norm of features, 25th, 50th, 75th and 95th percentiles of eigenvalues to obtain a rough representation of stretching of the feature space in individual dimensions. It can be seen that in almost all cases the RND target model has smaller eigenvalues than the SND models. This can also be seen in the L_2-norm values that the entire RND feature



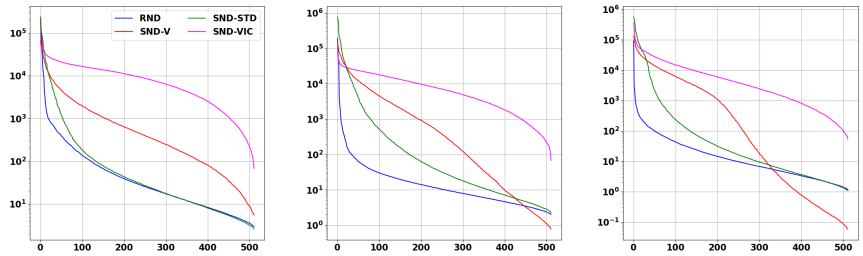
(a) Random target model (b) Trained target model

Figure 13: The t-SNE projected feature representations of the target model in Montezuma's Revenge task. The colors correspond to different rooms.

space seems to have a smaller volume compared to the SND feature space. We can see that RND and SND-STD are similar in shape. Their curve has a convex shape, with SND-STD having more stretched dimensions. SND-V and SND-VIC also have similarly concave shapes but SND-V stretches only about half of the available dimensions and then usually falls more steeply. The shape of SND-VIC curve is ensured by the variance (eq. 11) and covariance (eq. 12) components of its loss function. The missing decorrelation term in the SND-V loss function (eq. 4) results in an uneven stretching of the dimensions.

In contrast, the shape of the SND-STD curve is convex and the dimensions are used unevenly. After these analyses, we tried to improve the variance within the dimensions by adding a regularization term (eq. 9) which tried to maximize the variance within the feature vector. However, such a term had an expansive effect on the feature space and it was not possible to give it much weight, because the loss function (eq. 7) of the ST-DIM algorithm itself has an expansive effect, and the addition of another expansive term led to problems with the uncontrolled expansion of the feature space. Despite the small influence of the variance component of the loss function, the performance of SND-STD improved and it helped prevent agents from getting stuck in certain cases. If we compare RND and SND-STD feature spaces (in terms of eigenvalues) they look similar, but the latter model was able to achieve better results in 7 out of 9 environments. Our findings show that when training the target model, it is important to enforce the decorrelation of features and the equal use of all dimensions of the feature space. Such a model seems to be relatively robust and sufficiently sensitive to novelty.

Interestingly, for Pitfall task (not shown in Figure 14), despite their failure, our methods still tried to take advantage of the feature space dimensions. From the analysis of the trained agents, we saw that they were



(a) Montezuma's Revenge (b) Gravitar (c) Venture

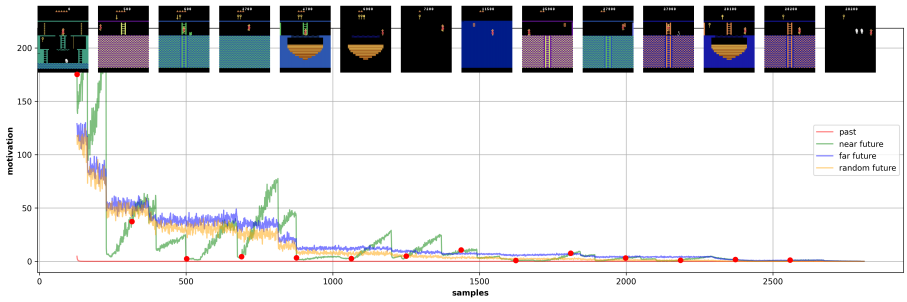
(d) Private Eye (e) Solaris (f) Caveflyer

(g) Coinrun (h) Jumper (i) Climber

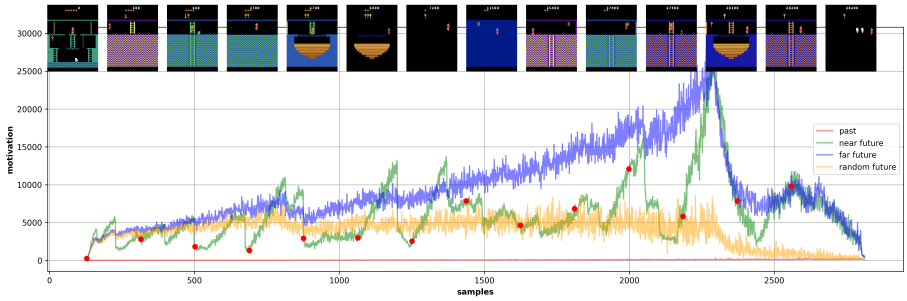
Figure 14: Descendingly ordered eigenvalues of the linear envelope obtained using the PCA method, which show the stretching of the feature space in individual dimensions. The horizontal axis shows the indices of eigenvalues, the vertical axis denotes the magnitude of eigenvalue in logarithmic scale. Based on these data, we tried to find out if there is a connection between the shape of the feature space and the performance of the given model. The graph for Pitfall was omitted, since it looked very similar to Private Eye.

able to explore several rooms, but in each there were enough moving objects that made the given state space rich and thus made it difficult to train the learned model, which led to a very slow decrease of the internal reward (we observed a similar behavior after short training sessions in other environments). We assume that with a larger number of training steps, the agent would eventually be able to reach the reward.

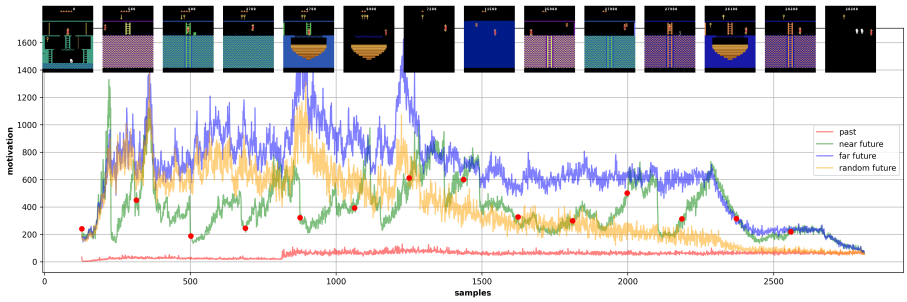
Another approach for different regularisation losses is understanding its time evaluation and the ability to provide large IM signal for previously



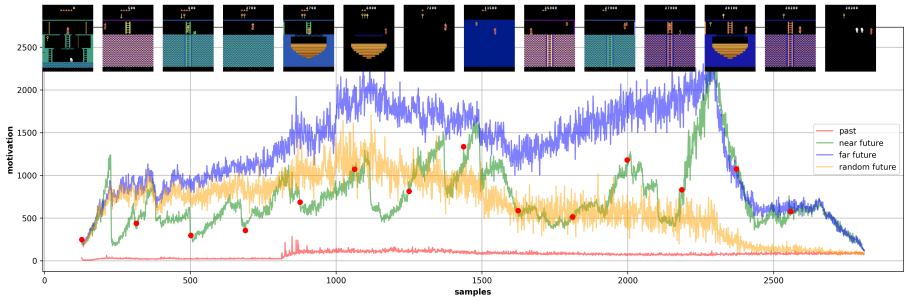
(a) RND



(b) SND-STD



(c) SND-V



(d) SND-VIC

Figure 15: Novelty detection for different regularisation losses as react on different future window. The states were collected on Montezuma's Revenge with our best agent, red dots correspond to state examples above.

unseen states. For the purpose of exploration, the most important is the ability to detect near future states, which are very close to already seen ones. We collected a set of 2700 states, from our best agent playing Montezuma's Revenge. During the experiment, we trained the IM modules only on past data, and tested on future data. The testing batch was selected from the following 4 time horizons, with respect to the agent being in step n, and testing batch indices m:

1. past: already seen states, $m < n$
2. near future: $n < m < n + 128$ steps in the future
3. far future: $m > n$
4. random: any batch from the set

We hypothesised that the Random Network distillation provides a sufficient signal only at the beginning of learning. Converging to zero leads to limited exploration abilities. This degradation corresponds to our results in Figure 15a. On the other hand, continuously updated target models can provide useful signals for the entire run. The corresponding results are displayed in Figures 15b, 15c, and 15d. On all three losses, the intrinsic motivation is much higher for non-seen states, and not converging into zero. The strong peeks for near future correspond to new rooms finding. The self supervised regularisation prevents collapsing motivation signal to zero. This insight gives us requirements for exploration signal. For future research we got also simple methodology for testing exploration abilities, without training whole RL agent which can be time consuming.

5. Discussion

We introduced a class of internal motivation algorithms, based on distillation error as a novelty indicator (SND), where the target model is trained using self-supervised learning. We adapted three existing self-supervised methods for this purpose and experimentally tested them on a set of environments that are considered difficult to explore. The proposed variants have been shown to eliminate the identified shortcomings of the RND model – the need for good initialization, low variance of intrinsic reward on different states and the loss of the motivational signal caused by the adaptation of the learning network.

In the experiments, we tested the overall performance of the agents in 10 environments. With the exception of one environment, it was confirmed that the SND algorithms achieved better results than other methods with which we compared them. For the Atari environments, we also evaluated the achieved game scores so that they could be compared with other published models that we did not include in in this work. Also from the point of view of the SND score, the models dominated the compared models.

In the analytical part, we focused on deeper understanding of the SND methods. We used a geometric approach trying to capture, at least in rough outlines, the properties of feature spaces. A comparison between a randomly initialized feature space and a feature space formed using one of the SND algorithms shows the correctness of our assumptions that self-supervised algorithms can distinguish even subtle differences within the state space. This turned out to be one of the weaknesses of the RND algorithm, which, while being good at distinguishing between sufficiently different states (e.g. different rooms in Montezuma's Revenge), it placed similar states close to each other in the feature space, making the work of the learned model easier. In the experiments, we thus observed a decrease in the standard deviation of the average intrinsic reward per episode, which meant that most of the visited states generated a similar reward.

We experimented with different target model architectures, different augmentations and intrinsic reward scaling. We found that the target model using ELU activation and only one fully connected layer with the learned model with three hidden layers performs the best. From the tested augmentations (noise, random tile masking, random convectional filter), we found the best performance for a combination of uniform noise with random tiles masking. As a questionable augmentation remains random down sampling and up sampling back, which could help remove noise while preserving representative information in the state vectors. We suggest to investigate this idea for next research. The scaling of intrinsic reward shows big sensitivity to this parameter. The best working value was 0.5, however we think this value should be optimized separately for each specific environment.

We did not specifically investigate the robustness of SND methods with regard to the initialization of the target model (which was again a problem with RND). We assume that self-supervised learning algorithms can cope with a poorly initialized model to a certain extent, but from the training experience we found that it is better to initialize the target models of SND-STD and SND-VIC to small values ($gain = 0.5$) and letting it expand itself while SND-V was initialized like RND to higher values ($gain = \sqrt{2}$).

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. 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.

We also compared the effect of state preprocessing on the performance of the SND-STD model. It turned out that the state preprocessing is not necessary since it has no significant effect on the agent's performance.

We also performed an analysis of novelty detection abilities of selected methods. After comparison with RND as a baseline, we can conclude that this baseline suffers from an IM-based reward vanishing problem. After adding the regularisation to the target model, much better features were obtained, with significant change compared to the baseline. Intrinsic reward vanishing disappears for all the tested losses. This is cross-validated also on t-SNE features visualisation, where regularised features yield much higher variance, which means larger sensitivity to novelty.

Based on the presented results, we can conclude that self-supervised learning methods are definitely promising in the creation of novelty detectors, which can be successfully used from the point of view of intrinsic motivation and improve the agent's exploration. A direct extension of SND methods will be the merging of the target model with the model to which the actor and critic are connected. We have already done some pilot research in this direction and it seems to be a feasible task. This would greatly optimize the entire model and speed up its training in terms of computing time. At the same time, we think that this approach can be an inspiration for a new class of algorithms that will specialize in creating feature mapping capturing the relationship of the environment to the agent itself, since current self-supervised methods are agnostic to these relationships.