Proceedings of Machine Learning Research vol 242:104–116, 2024 6th Annual Conference on Learning for Dynamics and Control

Tracking Object Positions in Reinforcement Learning: A Metric for Keypoint Detection

Emma Cramer EMMA.CRAMER@DSME.RWTH-AACHEN.DE
Jonas Reiher JONAS.REIHER@ML.RWTH-AACHEN.DE
Sebastian Trimpe TRIMPE@DSME.RWTH-AACHEN.DE
Institute for Data Science in Mechanical Engineering (DSME), RWTH Aachen University
Dennewartstraße 27, 52068 Aachen, Germany

Editors: A. Abate, M. Cannon, K. Margellos, A. Papachristodoulou

Abstract

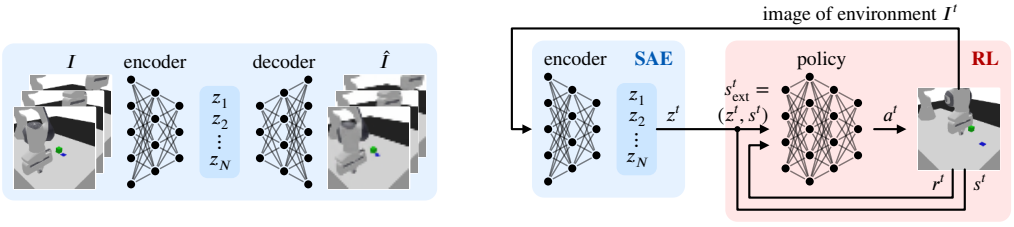
Reinforcement learning (RL) for robot control typically requires a detailed representation of the environment state, including information about task-relevant objects not directly measurable. Keypoint detectors, such as spatial autoencoders (SAEs), are a common approach to extracting a low-dimensional representation from high-dimensional image data. SAEs aim at spatial features such as object positions, which are often useful representations in robotic RL. However, whether an SAE is actually able to track objects in the scene and thus yields a spatial state representation well suited for RL tasks has rarely been examined due to a lack of established metrics. In this paper, we propose to assess the performance of an SAE instance by measuring how well keypoints track ground truth objects in images. We present a computationally lightweight metric and use it to evaluate common baseline SAE architectures on image data from a simulated robot task. We find that common SAEs differ substantially in their spatial extraction capability. Furthermore, we validate that SAEs that perform well in our metric achieve superior performance when used in downstream RL. Thus, our metric is an effective and lightweight indicator of RL performance before executing expensive RL training. Building on these insights, we identify three key modifications of SAE architectures to improve tracking performance.

Keywords: reinforcement learning, representation learning, autoencoder, keypoint detection

1. Introduction

In real-world control tasks like robotics, successful reinforcement learning (RL) often hinges on a thorough state representation. This necessitates including all task-relevant objects in the scene. This issue is particularly prominent in tasks involving unstructured environments or interactions with numerous objects, where defining the state space without significant prior knowledge is difficult. Image data provides a potential solution, either through direct end-to-end learning of the control signal or by first learning a low-dimensional representation of the high-dimensional data (Bleher et al., 2022; Levine et al., 2016). For practical applications, interpretability in terms of physical quantities is usually advantageous. Spatial autoencoders (SAEs) have been effective in learning low-dimensional representations, expressed as 2D points on the image plane, referred to as keypoints. This latent representation can be used, e.g., as part of the state representation of the RL agent. Figure 1 shows the complete learning pipeline.

While keypoints have led to well-performing RL algorithms (Kulkarni et al., 2019; Ghadirzadeh et al., 2017; Chen et al., 2023), limited research has been conducted on whether SAEs effectively extract positional information of objects in the scene and, if so, how well they do this. Prior work



(*a*) SAE trained on rec. loss (*b*) policy trained to maximize exp. return

Figure 1: SAE extracts 2D positions from images via a spatial bottleneck. The SAE encoder is then integrated into an RL framework to obtain a state representation for immeasurable objects.

often evaluates SAE architectures indirectly by training a downstream RL agent and evaluating the performance of the full RL pipeline (Qin et al., 2020; Boney et al., 2021; Chen et al., 2023). This approach requires a lot of resources for SAE evaluation since RL training is computationally expensive. Further, if weak RL performance is obtained, it is unclear whether RL or SAE training did not perform. Other works assess SAE performance by its training loss, which lacks insight into the physical meaningfulness of the keypoints. Some works propose a qualitative assessment by examining single keypoints on the image plane (Zhang et al., 2018; Puang et al., 2020). This approach disregards the importance of consistency over trajectories. We argue that keypoints essentially serve as sensor readings and thus how well task-relevant objects are tracked over time needs to be assessed in terms of accuracy and reliability.

This paper proposes a straightforward metric for quantitatively evaluating the extraction of positional information of task-relevant objects in the latent space of SAEs. The proposed metric is applied to (i) train multiple base SAE architectures and compare their tracking performance, (ii) explore various improvements of these architectures, and (iii) learn a suitable RL task using keypoints from different SAEs as the state. The evaluation reveals significant variations in the spatial extraction capability of common SAEs, emphasizing the importance of a thorough evaluation before incorporating them into RL states. Building on these insights, we propose three key modifications to substantially improve the tracking performance of common SAE architectures, resulting in, e.g., a 30 % increase in tracking capability for the commonly used KeyNet architecture (Jakab et al., 2018). We demonstrate that the metric allows us to judge SAEs with regard to capturing physically interpretable positional features and that this metric is a good indicator of downstream RL performance. For the considered robotic manipulation task, SAE training takes approximately an order of magnitude less computational resources than RL training, making our metric an effective and lightweight indicator of RL performance before expensive RL training.

2. Related Work

Various approaches utilize deep neural networks (NNs) to extract state representations from images or videos (Dwibedi et al., 2018; Seo et al., 2022); we review the ones most related to this work.

Unsupervised state representation learning for RL. Applications span from general continuous control (Dwibedi et al., 2018; Hafner et al., 2019) to robotic manipulation (Lesort et al., 2019; Rafailov et al., 2021). Many models follow an autoencoder structure with a low-dimensional bottleneck, optimizing for input reconstruction (Finn et al., 2016; Yarats et al., 2021). Some of these

learn world models and recurrently capture environment dynamics in the latent representation (Ha and Schmidhuber, 2018; Seo et al., 2022; Hafner et al., 2023). Generally, autoencoders constrain the dimension, but not *what* is captured in the latent space (Yarats et al., 2021; Rafailov et al., 2021). In contrast, SAEs are constrained to capture 2D keypoint positions (Finn et al., 2016).

Spatial autoencoder architectures. SAE keypoints have successfully been used as RL state representations in robotic control (Puang et al., 2020; Chen et al., 2023), to play Atari games (Kulkarni et al., 2019), or to provide a goal description (Qin et al., 2020). Central to SAEs is the spatial soft-argmax layer first proposed by Levine et al. (2016) to train an end-to-end deep visuomotor policy. Finn et al. (2016) modified this approach to obtain a standalone deep SAE architecture, consisting of a convolutional encoder and fully connected decoder. Jakab et al. (2018) propose the KeyNet architecture, incorporating a convolutional decoder. These two elements, the encoder-decoder structure and the spatial soft-argmax layer, are essential to all SAEs. Many architectures build upon these blocks; incorporating feature transport mechanisms (Kulkarni et al., 2019), working on error maps (Gopalakrishnan et al., 2021), and reconstructing segmentation masks (Puang et al., 2020) or frame differences (Sun et al., 2022). Recently, SAEs have been extended to learn 3D points (Li et al., 2022; Sun et al., 2023). While these approaches differ in the way they are trained, all aim to represent positional information. We focus our investigation on two of the most common base architectures (Finn et al., 2016; Jakab et al., 2018) as (i) they form the basis for many more complex architectures and (ii) we found that if trained correctly, they can serve as reliable feature extractors. In principle, our evaluation procedure can be applied to all of the above architectures.

Evaluation of SAEs. Typically, SAEs are evaluated indirectly through compute-intensive RL or control performance (Qin et al., 2020; Wang et al., 2022; Boney et al., 2021) or qualitative visual assessments (Zhang et al., 2018; Puang et al., 2020). In general, latent representations can be evaluated via reconstruction loss (Finn et al., 2016), disentanglement measurements (Carbonneau et al., 2022) or mutual information estimates (Rezaabad and Vishwanath, 2020), all of which neglect the spatial 2D keypoint structure and thus do not assess the physical meaningfulness of the features. In the computer vision domain, keypoints for image matching are evaluated by reprojecting from different views with known camera transformation (Zhao et al., 2023), which is not applicable for SAEs. Jakab et al. (2018) approximate labeled ground truth points as linear combinations of all keypoints. Their KeyNet SAE is evaluated with the percentage of these predicted points within a fixed distance from the labels. The same linear combination has been used by others to compute mean errors to ground truth points (Zhang et al., 2018; Lorenz et al., 2019; Sun et al., 2022). Kulkarni et al. (2019) match keypoints to ground truth points via a min-cost assignment and compute precision and recall over trajectories. Although being quantitative, the above approaches cannot assess the quality of keypoints over trajectories and allow no statement about whether all task-relevant objects are represented. We find that both aspects are critical for use in control or RL and our method, described in Section 4, addresses these key limitations in existing evaluation approaches.

3. Problem Setting

We consider the general structure of an autoencoder $I \xrightarrow{h_{\text{enc},\phi}} z \xrightarrow{h_{\text{dec},\psi}} \hat{I}$, operating on an input image I, which is mapped to a latent representation z via an NN encoder $h_{\text{enc},\phi}$ and then back to a reconstructed image $\hat{I}$ via an NN decoder $h_{\text{dec},\psi}$ (cf. Figure 1(*a*)). Typically, the autoencoder is trained in an unsupervised fashion to minimize reconstruction loss $L(I, \hat{I}) = \|I - \hat{I}\|_2^2$ while restricting the

dimension of the latent space with a low dimensional bottleneck. Here, we are particularly interested in spatial autoencoders (SAEs) (Finn et al., 2016), which aim to represent 2D positions of objects in an image as latent variables z. For this, the last layer of the encoder $h_{\text{enc},\phi}$ with N outputs is chosen as a soft-argmax layer according to Finn et al. (2016). This layer ensures that the latent space can be interpreted as N keypoints in the image plane with $z = (z_1, z_2, \dots, z_N) \in \mathbb{R}^{2\times N}$. For this, the feature maps $M \in \mathbb{R}^{H\times W\times N}$ of the last convolutional encoder layer are passed through a channel-wise softmax layer $s_{hwn} = \exp(m_{hwn}/\alpha)/\sum_{h',w'} \exp(m_{h'w'n}/\alpha)$, where α is a learned temperature parameter and h, w, and n are indices along the height, width, and depth dimensions of M. Then the n-th 2D point of maximum activation is computed as $z_n = (\sum_{h,w} h \cdot s_{hwn}, \sum_{h,w} w \cdot s_{hwn})$.

We consider a setup with K rigid objects that shall be tracked. Let the ground truth position of the k-th object (e.g., its center of mass) in the 2D image space be given by $x_k \in \mathbb{R}^2$, and the positions of all objects collectively by $x = (x_1, \dots, x_K) \in \mathbb{R}^{2\times K}$. An ideal SAE should track x with its latent representation z in *some sense*. However, how to evaluate the tracking performance is unclear, and proposing a method that quantifies this is our main objective:

Problem 1 *We seek to quantify how well the keypoints z represent the ground truth objects x.*

SAEs are often used as feature extractors for RL tasks, where keypoints z are then part of the state representation. In RL, an agent learns to optimize an objective through interaction with an environment (Sutton and Barto, 1998). The environment is represented as a discounted Markov decision process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, p, r, \rho_0, \gamma)$, with state space $\mathcal{S}$, action space $\mathcal{A}$, the typically unknown transition probability distribution $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$, the reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, the distribution of the initial state $\rho_0(s_0) : \mathcal{S} \to \mathbb{R}$, and the discount rate $\gamma \in (0, 1)$. A policy $\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ selects an action with a certain probability for a given state. The agent interacts with the MDP to collect episodes $\tau = (s^0, a^0, r^1, s^1, \dots, s^T)$, which are sequences of states, actions, and rewards over time steps $t = 0, \dots, T$. The usual objective in RL is to find the policy π that maximizes the expected return $J(\pi) = \mathbb{E}_\tau[\sum_{t=1}^{T} \gamma^t r^t]$, where the expectation is over trajectories τ under the policy π. The general understanding in literature (Finn et al., 2016; Ghadirzadeh et al., 2017; Kulkarni et al., 2019; Wang et al., 2022) is that well-tracking SAEs will yield better RL performance, such as higher expected return $J(\pi)$ or episode success rates. We investigate whether this holds true for the metric proposed for Problem 1:

Problem 2 *Is SAE performance (according to Problem 1) an indicator for RL performance?*

If this hypothesis holds true, SAE performance can be evaluated before actual RL training, usually at significantly lower computational cost.

4. A Metric to Evaluate Keypoints

In this section, we propose a metric to quantify the tracking performance of an SAE, addressing Problem 1. In Section 5.1, we then use the metric for RL to evaluate Problem 2.

As RL makes decisions sequentially over time, we are interested in tracking performance over multiple time steps. Therefore, we denote by $x_k^t \in \mathbb{R}^2$ the ground truth position of the k-th object, and by $x^t \in \mathbb{R}^{2\times K}$ the positions of all K objects collectively at time t. Furthermore, we denote the trajectory of these objects over time steps $t = 0, \dots, T-1$ by $x^\tau = (x^0, x^1, \dots, x^{T-1}) \in \mathbb{R}^{2\times K\times T}$. We use analogous notation for the N latent keypoints; that is, $z^\tau = (z^0, z^1, \dots, z^{T-1}) \in \mathbb{R}^{2\times N\times T}$ denotes the trajectory of keypoints.

Given this notion of trajectories, Problem 1 translates to measuring how well the keypoint trajectory z^τ follows the ground truth trajectory x^τ for a given instance of an SAE. A naive approach would be to directly compute the Euclidean distance between point-pairs along these trajectories. However, this will not yield satisfactory results. The keypoints are learned in an unsupervised fashion, which provides no guarantee about which part of an object is tracked. As points on a rigid object have a fixed relation to each other, it is reasonable to assume that, for downstream RL training, any point on the object is an equally suitable representation. For example, if a ground truth point and a keypoint are on the same object at a constant offset, this offset would accumulate to a tracking error when naively taking the difference between the two points. Thus, we need to account for offsets by an appropriate transformation. Finally, the SAE extracts many keypoints (usually $N > K$) and the association of keypoints to ground truth points is unknown. Taking these together, an evaluation protocol of keypoints will thus require (i) accounting for the offset between any point on the object and ground truth, (ii) associating keypoints with ground truth points, and (iii) developing a quantitative measure to evaluate the capability of tracking all relevant ground truth points.

Transformation. Keypoints are coordinates in the 2D image space, which are supposed to track objects in 3D space. Often, the center of mass (CM) is taken as the ideal point to represent the 3D position of an object in the world frame. However, for the downstream RL task, the keypoints do not have to track the CM, but any fixed point on the object, i.e., the point's offset from the CM should be constant in the object's 3D frame of reference (cf. Figure 2). If the keypoints were to track the CM, keypoints and ground truth points would coincide in image space. Due to the 3D offset, we also observe an offset in 2D-image space (cf. Figure 2). This 2D offset is generally unknown; it depends on the unknown 3D offset, object position, orientation, and camera view. Even if the keypoints were to track a point on an object perfectly, this offset would falsely suggest a tracking error in 2D. Instead of capturing the full geometry of the problem, which requires additional problem insight, we propose a lightweight approach that eliminates the main offsets between keypoints and ground truth. We consider a time-invariant affine transformation of keypoints $\hat{z} = Az + b$, where $A \in \mathbb{R}^{2\times 2}$ and $b \in \mathbb{R}^2$ are fit via ordinary least squares on a held-out test set, containing random time steps from trajectories unseen in SAE training. This transformation can account for the scaling and translation of a keypoint trajectory. We note that even with a time-invariant 3D offset, the 2D offset can be time-variant due to the object's motion; the time invariance thus represents an approximation. Still, we find that this transformation is easy to compute, requires no additional information about the ground truth objects, and works well in practice (cf. Section 5).

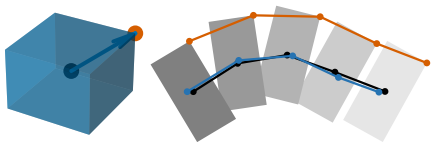


Figure 2: We consider keypoints (red) to be equally informative about object positions as the ground truth CM (black). Motion of both points results in a varying offset in the image plane. We evaluate with transformed keypoints (blue) minimizing the offset.

Association and tracking error. Consider the trajectories x^τ, $\hat{z}^\tau$ of K ground truth objects and N transformed keypoints. We propose an error metric between the trajectory of one ground truth object x_n^τ and the transformed trajectory of one keypoint $\hat{z}_n^\tau$. We define the tracking error $e_{n,k}$ between any two trajectories n and k as

$$e_{n,k} = \sum_{t=1}^{T} \|\hat{z}_n^t - x_k^t\|_2^2. \tag{1}$$

The error $e_{n,k}$ is a measure of how well a specific keypoint tracks a ground truth object over time. Using the tracking error, we determine the index of the keypoint $z_{n_k^*}$ that best tracks object x_k as $n_k^* = \arg\min_n e_{n,k}$. Once we assigned the most suitable keypoint for each ground truth object, we give the tracking error of the associated keypoint as $e_{n_k^*,k}$. For our evaluation, we always consider the tracking error of the best keypoint. The lower this tracking error, the better the ground truth point x_k is represented by the keypoint $z_{n_k^*}$. The error measure enables a comparison of different SAE architectures and individual training runs of the same architecture broken down into objects.

For the later evaluation of SAEs, we now define indicators for an SAE's overall tracking performance. We classify an object x_k as correctly tracked if the tracking error of the most suitable keypoint $z_{n_k^*}$ is below an application-specific threshold $\mu_k > 0$. The index set $\mathcal{X}_c$ of all correctly tracked objects is given by $\mathcal{X}_c = \{k : e_{n_k^*,k} \leq \mu_k\}$. We then define the tracking capability TC of one trained SAE as the percentage of tracked ground truth objects, i.e.,

$$\mathrm{TC} = |\mathcal{X}_c| / K. \tag{2}$$

An ideal tracking capability of $\mathrm{TC} = 1$ means that for this SAE, the position of all ground truth objects is correctly encoded in the latent space.

A quantitative evaluation should consider the distribution of the tracking error and the tracking capability over multiple training runs. We look at the mean, median, and the variance of the tracking error over runs. Similarly, we evaluate the mean tracking capability $\overline{\mathrm{TC}}$ over multiple runs. Intuitively, $\overline{\mathrm{TC}}$ gives the mean percentage of all ground truth objects captured by keypoints. An SAE with a high mean tracking capability is a reliable feature extractor for RL scenarios. For individual ground truth objects, we denote as $\overline{\mathrm{TC}}_k$ the mean tracking capability for object k.

5. Evaluation

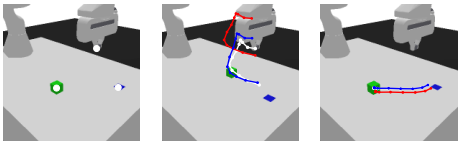


Figure 3: The PandaPush-v3 task with three object positions x_k marked (left). Selected ground truth, keypoint, and transformed keypoint trajectories are shown in white, red, and blue for the end effector (middle) and cube (left).

We first use our proposed metrics (1) and (2) to evaluate the tracking performance of base SAE architectures commonly used in RL and propose architecture modifications to improve tracking. We then investigate how tracking performance links to performance in a downstream RL task. The empirical results reveal the following main insights:

1. The proposed metric is able to quantify the tracking performance of SAEs.
2. The combination of the best baseline SAE with our proposed modification yields $\overline{\mathrm{TC}} = 0.99$, and it can thus be considered a reliable and precise spatial feature extractor.
3. Our proposed metric for SAE tracking performance is indicative of the performance of RL; that is, the architecture with best SAE metric also achieves best asymptotic return.
4. The best-found architecture in terms of SAE tracking achieves an RL return comparable to training with ground truth points.

5.1. SAE Evaluation

We demonstrate the suitability of the tracking error and tracking capability introduced in Section 4 to evaluate the performance of SAEs. We provide visualizations of our quantitative results at youtu.be/8KqFXQiWa9w.

SAE experiment setup. We use the PandaPush-v3 environment from the panda-gym (Gallouédec et al., 2021) simulation. The robot's task is to push a cube toward a target (cf. Figure 3). We identify three task-relevant objects in this environment, (i) the green *cube* to be moved, (ii) the blue square indicating the *target*, and (iii) the tip of the *end effector*. The different sizes and motion behavior of the objects make them a suitable selection to evaluate the tracking performance. We consider three standard SAE architectures and our own combination of modifications:

Basic: We design the Basic architecture to be a simple and efficient SAE baseline incorporating the key components that all SAEs typically share. The CNN encoder has six convolutional layers and max-pooling operations in between. The decoder uses KeyNet's Gaussian kernel maps, followed by three convolutional layers. We look at two versions of this SAE with $N = 16$ (Basic) and $N = 32$ keypoints (Basic-kp32).

DSAE (Finn et al., 2016): DSAE introduced the spatial soft-argmax bottleneck, still used in many other architectures (Zhang et al., 2018; Cabi et al., 2019; Gopalakrishnan et al., 2021; Puang et al., 2020; Boney et al., 2021). This was the first SAE to be successfully used for RL training. $N = 16$ keypoints are captured between a CNN encoder and fully connected decoder.

KeyNet (Jakab et al., 2018): KeyNet is a widely used and built upon SAE architecture (Kulkarni et al., 2019; Minderer et al., 2019; Gopalakrishnan et al., 2021), consisting of a CNN encoder and decoder with $N = 30$ keypoints. Input to the decoder are N feature maps with isotropic Gaussian kernels at the corresponding keypoint locations.

Vel-std-bg modifications: We propose a set of modifications to the above architectures, combining ideas from existing works and new ones. Analogously to DSAE, we add a velocity loss term to the reconstruction loss with a weighting factor β. By penalizing a change of keypoint velocities in subsequent frame pairs, the velocity loss encourages temporal consistency. KeyNet uses Gaussian heatmaps as input to the first CNN decoder layer. We propose making the standard deviation σ of these heatmaps trainable. This enables the decoder to control the radius of influence of a keypoint. Finally, we add a bias with the dimensions of the target image to the decoder's output, giving the decoder a straightforward way to reconstruct a stationary background and allowing time-varying keypoints to focus on moving objects. For the modified architectures, we call the combinations of the KeyNet or Basic architecture combined with our proposed modifications KeyNet-vel-std-bg and Basic-vel-std-bg, respectively.

While many more architectures exist in literature (cf. Section 2), we deliberately choose baseline architectures maintaining the usual autoencoder setup without auxiliary networks such as adversaries or feature transport mechanisms. We choose modifications which we believe to be beneficial for the main goal of SAEs, spatial tracking of keypoints over time. For SAE tracking evaluation, we conduct 24 training runs with different random seeds. The tracking thresholds need to be chosen heuristically. Here we choose $\mu_{\text{cube}} = \mu_{\text{target}} = 0.015$ and $\mu_{\text{eef}} = 0.1$. Intuitively larger objects result in a larger tracking error, due to the possible offset to the center of mass. We find a good heuristic to be related to the SAE reconstruction. Objects appear in the reconstruction when the tracking error falls below μ_k. Additional modifications, an ablation study, and all experimental details such as hyperparameters can be found in Cramer et al. (2023).

Evaluating accuracy via the tracking error. First, we study the tracking error of individual runs during training. We observe a sudden drop in tracking error whenever the SAE has learned to track an object. To understand this behavior, we look at the tracking error over episodes for an SAE with medium performance, the Basic-kp32, in Figure 4. All runs for Basic-kp32 show the drop in tracking error for the cube, which is the easiest to track. For the target, which is slightly harder to

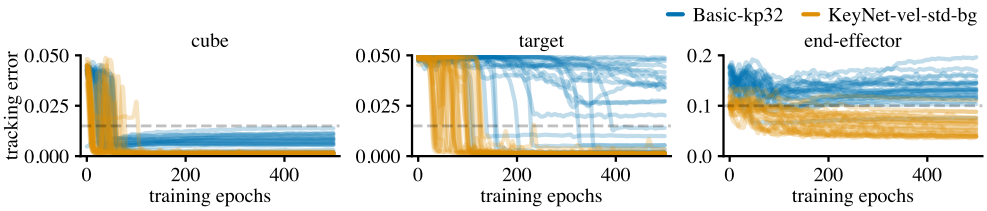


Figure 4: Basic-kp32 and KeyNet-vel-std-bg tracking errors $e_{n_k^*,k}$ for $K = 3$ objects over epochs.

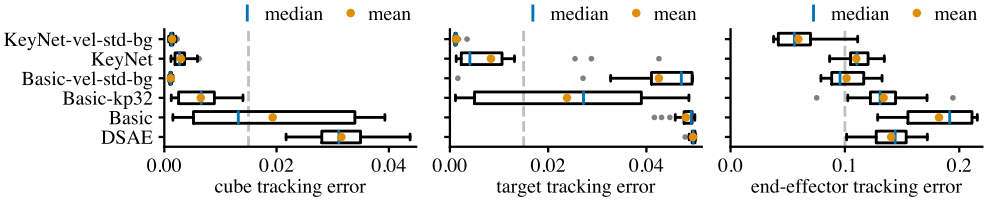


Figure 5: Box plots of the tracking error $e_{n_k^*,k}$ for $K = 3$ ground truth objects.

track due to its smaller size and rare movement, only a few runs show the expected drop below our threshold, resulting in a correctly tracked target. Instead of a sharp drop, the tracking error for the end-effector shows a shallow decrease over training epochs. We interpret this observation as follows: The end-effector occupies considerably more pixels in the image than cube and target. Thus, the reconstruction first focuses on these areas, resulting in early vague tracking and reconstruction. However, tracking a point on the end-effector consistently is achieved only by a few runs. Looking at the tracking error of KeyNet-vel-std-bg, Figure 4 shows a distinct drop below the threshold for the cube and target. Even for the end-effector, the tracking error consistently falls below the threshold, indicating successful tracking. We find that the tracking error is useful in examining exactly how accurate a trained SAE architecture instance can track individual objects.

Evaluating reliability via the tracking error. Our results indicate differing tracking performance for random seeds within the same architecture, showing that the SAE architectures need to be evaluated over multiple training runs. The tracking error's distribution over 24 training runs is illustrated in Figure 5. We remark that the tracking error varies among (i) architectures, (ii) random seeds, and (iii) objects. Among the standard architectures, KeyNet attains the lowest mean tracking error and smallest variance, indicative of good overall tracking performance. For DSAE and Basic, larger tracking errors with greater variance are observed, marking them less reliable. The KeyNet-vel-std-bg architecture shows lowest mean tracking error and variance for all three objects. We identify the criteria for well-performing architectures as low mean tracking error and small variance over runs.

Evaluating overall performance via the tracking capability. Figure 6 shows the sum of mean object tracking capabilities $\overline{\mathrm{TC}}_k$ over architectures, further demonstrating the varying tracking performance across SAE architectures. Examining the tracking capability with regard to the individual ground truth objects, we further substantiate our hypothesis that the target and end-effector are more difficult to track than the cube. The tracking performance of all base architectures has potential for improvement as none is close to the theoretical maximum of 3.0. The combination vel-std-bg yields consistent improvement in tracking capability. KeyNet already tracks cube and target well and has a $\overline{\mathrm{TC}} = 0.681$. KeyNet-vel-std-bg has a near-perfect mean tracking capability of $\overline{\mathrm{TC}} = 0.986$. The

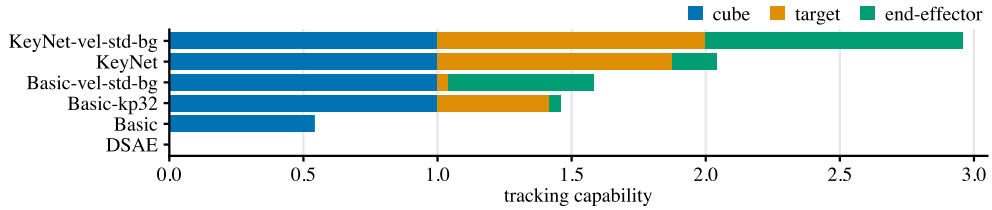


Figure 6: Tracking capabilities $\overline{\mathrm{TC}_k}$ for $k = 3$ ground truth objects of the baseline architectures.

biggest change can be seen in end effector tracking, which improved from $\overline{\mathrm{TC}}_{\mathrm{eef}} = 0.167$ to 0.958. We see that the tracking capability is a compact description of how well task-relevant objects are tracked. This information is critical for downstream control and RL tasks.

Combining the insights from the tracking error and tracking capability answers Problem 1.

5.2. RL Evaluation

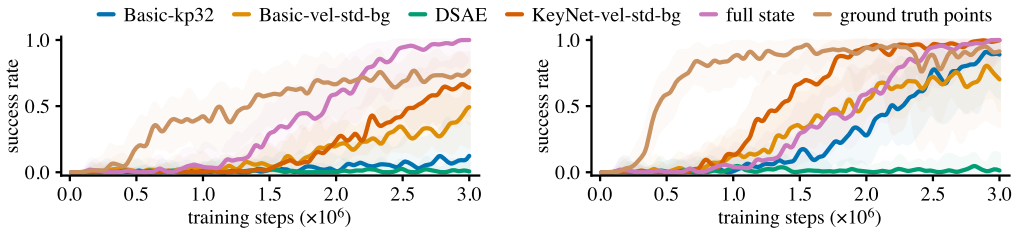
We run RL experiments with SAE architectures selected by their tracking performance and find that this is a good indicator of downstream RL performance.

RL experiment setup. For RL experiments with SAEs as state, we randomly sample 5 trained SAEs per architecture and conduct 2 randomly seeded RL training runs with each of them, yielding a total 10 runs per SAE architecture. We use the SAC (Haarnoja et al., 2018) implementation from stable-baselines3 (Raffin et al., 2021). Hyperparameters are listed in Cramer et al. (2023).

We consider two types of state representation for RL with SAE-encoded keypoints: (i) latent keypoints as state $s^t = z^t$, (ii) latent keypoints combined with robot 3D position o_{eef} and velocity $\dot{o}_{\mathrm{eef}}$, giving $s^t_{\mathrm{ext}} = (z^t, o_{\mathrm{eef}}, \dot{o}_{\mathrm{eef}})$. The second scenario is relevant since end-effector position and velocity are often available as robot state measurements. As additional benchmarks, we include state representations with ground truth points x^t, which are usually not available in practice, obtaining $s^t = x^t$ and $s^t_{\mathrm{ext}} = (x^t, o_{\mathrm{eef}}, \dot{o}_{\mathrm{eef}})$. Finally, we compare to RL runs with the full 3D simulation state, including positions, velocity, and orientation of cube and target. Actions consist of 3D displacements $a^t = (\Delta o_{\mathrm{eef}})^t$ of the end effector at every time step. We use a sparse reward with $r_t = -1$ and $r_T = 0$ on episode success. Following (Agarwal et al., 2021), we report interquartile mean (IQM) success rates with bootstrapped 95 % percentile confidence intervals.

Reinforcement learning with keypoints. Figure 7(*a*) shows the RL performance with state representation s^t. We observe varying success rates depending on the SAE architecture and the gradations in RL performance follow the order of SAEs by tracking capability, as seen in Figure 6. The DSAE architecture shows no RL progress. Although both Basic-vel-std-bg and Basic-kp32 have similar total tracking capabilities (cf. Fig. 6), the former performs better on the RL task. This is due to its ability to track the end-effector reasonably well, while Basic-kp32 tracks the target instead. End-effector tracking is critical, as moving the cube is otherwise impossible. The best-tracking KeyNet-vel-std-bg dominates the RL with learned keypoints. Still this architecture does not reach the full-state performance. This is to be expected since the representation is limited to 2D space and lacks velocity information. The runs using 2D ground truth points, mimicking a perfect SAE, learn significantly earlier than KeyNet-vel-std-bg, but only achieve a slightly higher final success rate.

Figure 7(*b*) shows the RL runs using state s^t_{ext}, i.e., including the end-effector's 3D position and velocity in addition to keypoints. RL performance with KeyNet-vel-std-bg and the full state



(*a*) state s^t (*b*) state s^t_{ext}

Figure 7: Success rate for different states: keypoints only in 7(*a*) and extended state in 7(*b*).

show comparable final success rates close to 1.0, indicating that 2D keypoints are a useful state representation. KeyNet-vel-std-bg additionally learns notably faster. We presume that the reduced 2D representations of target and cube positions accelerate RL training. As expected with DSAE, tracking neither target nor cube, learning progress is impossible. Compared to the first experiment setup, Basic-kp32 and Basic-vel-std-bg switch positions in final RL performance. Although Basic-vel-std-bg improves with the more precise 3D end-effector position, it is still unable to track the target and, therefore, limited in performance. For Basic-kp32, the missing end-effector tracking is now compensated with ground truth 3D information. Using its notable target tracking capability, it achieves better final performance. Initially, Basic-vel-std-bg learns faster, supporting the assumption that 2D representations can accelerate RL training. These kinds of insights are facilitated by the tracking capability and would not have been possible via traditional SAE evaluation. The IQM success rates for runs with ground truth instead of keypoints show faster learning but do not quite reach the maximum of full state and KeyNet-vel-std-bg.

Answering Problem 2, we find a link between SAE tracking capability, including the tracking capability for individual objects, and downstream RL performance.

6. Conclusion

We propose a metric to evaluate SAE performance with respect to task-relevant objects. By means of this metric, we show that well-performing SAE architecture actually track positions of task-relevant objects. We find notable performance differences in SAE architectures and identify three components that reliably improve performance, leading to almost perfect object tracking. We show that SAE tracking performance is indicative of downstream RL performance for a representative robotic manipulation task. This allows identifying suitable SAEs after comparatively lightweight SAE pretraining and before computationally expensive RL training. In addition, troubleshooting is greatly facilitated by the ability to evaluate the performance of an SAE as a key component of the RL pipeline. We observe that an RL agent using keypoints as part of its state achieves RL performance comparable to an agent with full simulation state. Thus, we consider keypoints a suitable state representation for robotic RL. We have demonstrated that this straightforward metric is effective in evaluating SAE architectures. The metric can be used to analyze any 2D keypoint extractor and is not restricted to SAEs. Investigating alternative keypoint extractors and extensions to 3D keypoints is thus a promising avenue for future research. The code to reproduce all results is available at github.com/Data-Science-in-Mechanical-Engineering/SAE-RL and can be used to inform future research.

Acknowledgments

We thank Paul Brunzema and Bernd Frauenknecht for their helpful comments. We also thank Robin Kupper for his contributions in the early stages of this research. This work was partially funded by the "Demonstrations- und Transfernetzwerk KI in der Produktion (ProKI-Netz)" initiative, funded by the German Federal Ministry of Education and Research (BMBF, grant number 02P22A010). Computations were performed with computing resources granted by RWTH Aachen University under project rwth1385.

References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep Reinforcement Learning at the Edge of the Statistical Precipice. In *Advances in Neural Information Processing Systems*, volume 34. Curran Associates, Inc., 2021.

Steffen Bleher, Steve Heim, and Sebastian Trimpe. Learning fast and precise pixel-to-torque control: A platform for reproducible research of learning on hardware. 29(2):75–84, 2022.

Rinu Boney, Alexander Ilin, and Juho Kannala. Learning of feature points without additional supervision improves reinforcement learning from images. *arXiv preprint arXiv:2106.07995*, 2021.

Serkan Cabi, Sergio Gómez Colmenarejo, Alexander Novikov, Ksenia Konyushkova, Scott Reed, Rae Jeong, Konrad Zolna, Yusuf Aytar, David Budden, and Mel Vecerik. Scaling data-driven robotics with reward sketching and batch reinforcement learning. *arXiv preprint arXiv:1909.12200*, 2019.

Marc-Andre Carbonneau, Julian Zaidi, Jonathan Boilard, and Ghyslain Gagnon. Measuring Disentanglement: A Review of Metrics. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

Ling-Chen Chen, Chi-Kai Ho, and Chung-Ta King. KeyState: Improving Image-based Reinforcement Learning with Keypoint for Robot Control. In *IEEE International Conference on Industrial Technology (ICIT)*, Orlando, FL, USA, 2023.

Emma Cramer, Jonas Reiher, and Sebastian Trimpe. Tracking Object Positions in Reinforcement Learning: A Metric for Keypoint Detection (extended version), 2023. URL https://arxiv.org/abs/2312.00592.

Debidatta Dwibedi, Jonathan Tompson, Corey Lynch, and Pierre Sermanet. Learning actionable representations from visual observations. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)*, 2018.

Chelsea Finn, Xin Yu Tan, Yan Duan, Trevor Darrell, Sergey Levine, and Pieter Abbeel. Deep spatial autoencoders for visuomotor learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

Quentin Gallouédec, Nicolas Cazin, Emmanuel Dellandréa, and Liming Chen. panda-gym: Open-source goal-conditioned environments for robotic learning. *arXiv preprint arXiv:2106.13687*, 2021.

Ali Ghadirzadeh, Atsuto Maki, Danica Kragic, and Mårten Björkman. Deep predictive policy training using reinforcement learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

Anand Gopalakrishnan, Sjoerd van Steenkiste, and Jürgen Schmidhuber. Unsupervised object keypoint learning using local spatial predictability. In *Proceedings of the IEEE Conference on Learning Representations(ICLR)*, 2021.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. *Advances in neural information processing systems*, 31, 2018.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*. PMLR, 2018.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*. PMLR, 2019.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering Diverse Domains through World Models. *arXiv preprint arXiv:2301.04104*, 2023.

Tomas Jakab, Ankush Gupta, Hakan Bilen, and Andrea Vedaldi. Unsupervised learning of object landmarks through conditional image generation. In *Advances in Neural Information Processing Systems (NIPS)*, volume 31, 2018.

Tejas D. Kulkarni, Ankush Gupta, Catalin Ionescu, Sebastian Borgeaud, Malcolm Reynolds, Andrew Zisserman, and Volodymyr Mnih. Unsupervised learning of object keypoints for perception and control. In *Advances in Neural Information Processing Systems (NIPS)*, volume 32, 2019.

Timothée Lesort, Mathieu Seurin, Xinrui Li, Natalia Díaz-Rodríguez, and David Filliat. Deep unsupervised state representation learning with robotic priors: a robustness analysis. In *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1), 2016.

Yunzhu Li, Shuang Li, Vincent Sitzmann, Pulkit Agrawal, and Antonio Torralba. 3d neural scene representations for visuomotor control. In *Conference on Robot Learning*. PMLR, 2022.

Dominik Lorenz, Leonard Bereska, Timo Milbich, and Bjorn Ommer. Unsupervised Part-Based Disentangling of Object Shape and Appearance. 2019.

Matthias Minderer, Chen Sun, Ruben Villegas, Forrester Cole, Kevin P. Murphy, and Honglak Lee. Unsupervised learning of object structure and dynamics from videos. *Advances in Neural Information Processing Systems*, 32, 2019.

En Yen Puang, Keng Peng Tee, and Wei Jing. Kovis: Keypoint-based visual servoing with zero-shot sim-to-real transfer for robotics manipulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

Zengyi Qin, Kuan Fang, Yuke Zhu, Li Fei-Fei, and Silvio Savarese. Keto: Learning keypoint representations for tool manipulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

Rafael Rafailov, Tianhe Yu, Aravind Rajeswaran, and Chelsea Finn. Offline reinforcement learning from images with latent space models. In *Learning for Dynamics and Control*. PMLR, 2021.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research*, 22(268), 2021.

Ali Lotfi Rezaabad and Sriram Vishwanath. Learning Representations by Maximizing Mutual Information in Variational Autoencoders. In *IEEE International Symposium on Information Theory (ISIT)*, Los Angeles, CA, USA, 2020.

Younggyo Seo, Kimin Lee, Stephen L. James, and Pieter Abbeel. Reinforcement learning with action-free pre-training from videos. In *International Conference on Machine Learning*. PMLR, 2022.

Jennifer J. Sun, Serim Ryou, Roni H. Goldshmid, Brandon Weissbourd, John O. Dabiri, David J. Anderson, Ann Kennedy, Yisong Yue, and Pietro Perona. Self-Supervised Keypoint Discovery in Behavioral Videos. 2022.

Jennifer J. Sun, Lili Karashchuk, Amil Dravid, Serim Ryou, Sonia Fereidooni, John C. Tuthill, Aggelos Katsaggelos, Bingni W. Brunton, Georgia Gkioxari, Ann Kennedy, Yisong Yue, and Pietro Perona. BKinD-3D: Self-Supervised 3D Keypoint Discovery From Multi-View Videos. 2023.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT press, 1998.

Tianying Wang, En Yen Puang, Marcus Lee, Wei Jing, and Yan Wu. End-to-end Reinforcement Learning of Robotic Manipulation with Robust Keypoints Representation. In *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, 2022.

Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. Improving sample efficiency in model-free reinforcement learning from images. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 2021. Issue: 12.

Yuting Zhang, Yijie Guo, Yixin Jin, Yijun Luo, Zhiyuan He, and Honglak Lee. Unsupervised discovery of object landmarks as structural representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

Xiaoming Zhao, Xingming Wu, Jinyu Miao, Weihai Chen, Peter C. Y. Chen, and Zhengguo Li. ALIKE: Accurate and Lightweight Keypoint Detection and Descriptor Extraction. *IEEE Transactions on Multimedia*, 25, 2023.