Look where you look! Saliency-guided Q-networks for visual RL tasks

Anonymous Author(s)
Affiliation
Address
email

Abstract

Deep reinforcement learning policies, despite their outstanding efficiency in simulated visual control tasks, have shown disappointing ability to generalize across disturbances in the input training images. Changes in image statistics or distracting background elements are pitfalls that prevent generalization and real-world applicability of such control policies. We elaborate on the intuition that a good visual policy should be able to identify which pixels are important for its decision, and preserve this identification of important sources of information across images. This implies that training of a policy with small generalization gap should focus on such important pixels and ignore the others. This leads to the introduction of saliency-guided Q-networks (SGQN), a generic method for visual reinforcement learning, that is compatible with any value function learning method. SGQN vastly improves the generalization capability of Soft Actor-Critic agents and outperforms existing state-of-the-art methods on the Deepmind Control Generalization benchmark, setting a new reference in terms of training efficiency, generalization gap, and policy interpretability.

1 Introduction

Deploying reinforcement learning (RL) [Sutton and Barto, 2018] algorithms in real-life situations requires overcoming a number of still open challenges. Among these is the ability for the trained control policies to focus their attention on causal state features and ignore confounding factors [Machado et al., 2018, Henderson et al., 2018]. In visual RL tasks, this implies for instance being able to ignore the background and other distracting factors, even when they might be somehow correlated with progress within the task at hand. Despite a very active trend of research on the topic of closing the generalization gap for RL agents [Cobbe et al., 2019, 2020, Song et al., 2019, Hansen et al., 2021, Hansen and Wang, 2021], current algorithms are still rather brittle when it comes to filtering out such distracting factors, which hinders their applicability to real-life scenarios.

In the present work, we propose a novel method which encourages the agent to identify efficiently crucial input pixels, and strengthen the policy's dependency on those pixels. In plain words, we encourage the agent to pay attention to, and be self-aware of where it looks in input images, in order to make its decision policy more focused on important areas and less sensitive to ambiguous or distracting pixels. This intention is expressed within the generic method of saliency-guided Q-networks (SGQN), which can be applied on any approximate value iteration based, deep RL algorithm. SGQN relies on two core mechanisms. First, it regularizes the value function learning process with a consistency term that encourages the value function to depend in priority on pixels that are identified as decisive. The second mechanism pushes the agent to be self-aware of which pixels are responsible for making decisions, and encode this information within the extracted features. This second mechanism translates to a self-supervised learning objective, where the agent trains to predict its own Q-value's saliency maps. In turn, this improves the regularization of the value function

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

learning phase, which provide better labels for the self-supervised learning phase, overall resulting in a virtuous improvement circle.

SGQN is a simple, generic method, that permits many variants in the way the two core mechanisms are implemented. In the present paper, we demonstrate that applying SGQN to soft actor-critic agents [Haarnoja et al., 2018] dramatically enhances their quality on the DMControl generalization benchmark [Hansen and Wang, 2021], a standard evaluation benchmark for generalization in continuous actions RL. SGQN already improves the training efficiency of such agents in domains without distractions. But most importantly, it sets a new state-of-the-art in terms of generalization performance, in particular in especially difficult benchmarks where previous methods suffered from confounding factors. As a side benefit, it also provides explanations of its own decisions at run time, under the form of interpretable attribution maps, with no overhead cost and no need to compute ad hoc saliency maps, which is another desirable property in the pursuit of deployable RL.

Section 2 of this paper introduces the necessary background and state-of-the-art in closing the generalization gap in RL, as well as attribution methods, leading to the key intuitions underpinning SGQN. Section 3 introduces the method itself and implements it within soft actor-critic agents. Section 4 evaluates SGQN's training efficiency, generalization capabilities, and policy interpretability. It also discusses the different design choices made along the way and some foreseeable limitations. Section 5 summarizes and concludes this paper.

2 Background and related work

Reinforcement learning (RL). RL [Sutton and Barto, 2018] considers the problem of learning a decision making policy for an agent interacting over multiple time steps with a dynamic environment. At each time step, the agent and environment are described through a state $s \in \mathcal{S}$, and an action $a \in \mathcal{A}$ is performed; then the system transitions to a new state s' according to probability $\mathcal{P}(s'|s,a)$, while receiving reward $\mathcal{R}(s,a)$. The tuple $M = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$ forms a Markov Decision Process (MDP) [Puterman, 2014], which is often complemented with the knowledge of an initial state distribution $p_0(s)$. A decision making policy parameterized by θ is a function $\pi_\theta(a|s)$ mapping states to distributions over actions. Training a reinforcement learning agent consists in finding the policy that maximizes the discounted expected return $J(\pi_\theta) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t)]$.

Poor generalization in RL. Despite the recent progress of (deep) RL algorithms in solving complex tasks, a number of studies have pointed out their poor generalization capabilities. Using a grid-world maze environment, Zhang et al. [2018c] demonstrate the ability of deep RL agents to memorize a non-trivial number of training levels with completely random rewards. Using attribution methods, Song et al. [2019] highlight what they define as *observational overfitting* i.e., the propensity of RL agents to base their decision on background uninformative elements observed during training, instead of the semantic pieces of information one could intuitively expect such as object positions or relations. Zhang et al. [2018a] measure the generalization error in continuous control environments by training and testing agents on different sets of seeds. Zhao et al. [2019] define generalization in RL as robustness to a distribution of environments, and samples environments from this distribution to learn a robust policy. Overall, these works illustrate the lack of generalization abilities of vanilla deep RL algorithms, either to states that were not encountered during training, or to variations in the transition dynamics. In the present work, we aim to shape the policy learning process, so that it relies on meaningful features that permit such generalization and robustness.

Evaluating generalization in RL. Under the impetus of these works and the need for benchmarks with separate training and testing environments [Whiteson et al., 2011, Machado et al., 2018, Henderson et al., 2018], original benchmarks for evaluating the generalization capacities of RL agents have been designed. Packer et al. [2018] propose a modified version from control problems in OpenAI Gym [Brockman et al., 2016] and Roboschool [Schulman et al., 2017] that lets the user change the system dynamics. Machado et al. [2018] propose a modified version of the ALE environments [Bellemare et al., 2013] allowing one to change modes and difficulties. Without modifying the underlying transition model, Zhang et al. [2018b], Grigsby and Qi [2020], Stone et al. [2021] add distracting elements (e.g., addition of real images or videos in the background, change of colors) to the ALE environments and the Deepmind control suite. Even if the modifications to the original environments do not alter the semantic information, they already appear to be challenging for agents prone to observational overfitting. Cobbe et al. [2019, 2020], Juliani et al. [2019] use procedural content

generation to design highly randomized sets of environments with different level layouts, game assets, and objects locations, letting the user study robustness to several independent variation factors. One may note that the diversification of learning environments is in itself a first practical way to induce generalization [Tobin et al., 2017, Cobbe et al., 2019, 2020] and also permits curriculum-based learning [Jiang et al., 2021]. Nevertheless, when the diversity of training scenarios is lacking, three sets of methods are generally employed, as detailed below.

Regularization. Farebrother et al. [2018], Cobbe et al. [2019, 2020] demonstrated the beneficial effects of popular regularization methods from the supervised learning literature. Igl et al. [2019] mitigate the adverse effect that classical regularization may have on the gradient quality with selective noise injection and combine it with an information bottleneck regularization. Inspired by mixup [Zhang et al., 2018d], Wang et al. [2020] use mixtures of observations to stimulate linearity in the policy's outputs in-between states.

Data augmentation. Laskin et al. [2020a] evidence the benefits of training RL agents with augmented data (RAD). Yarats et al. [2020] average both the value function and its target over multiple image transformations (DrQ). Hansen et al. [2021] only apply data augmentation in Q-value estimation without augmenting Q-targets used for boostrapping. Raileanu et al. [2021] combine the previous method with UCB [Auer, 2002] to pick the most promising augmentation, and apply it to PPO [Schulman et al., 2017]. Yuan et al. [2022] propose a task-aware Lipschitz data augmentation method (TLDA) to augment task irrelevennt pixels. Fan et al. [2021] use weak data augmentation to train an expert without hindering its performance and distill its policy to a student trained with substantial data augmentation. Besides augmentations of raw inputs, other augmentations operate directly within the agents' network. Lee et al. [2020] introduces a random convolutions layer at the earliest level of the agent's network to modify the *texture* of the visual observations. Zhou et al. [2020] adapt mixup [Zhang et al., 2018d] with style statistics encoded in early instance normalization layers to increase data diversity. Bertoin and Rachelson [2022] apply channel-consistent local permutations of the feature map to induce robustness to spatial spurious correlations. Finally, data augmentation can also be used in an auxiliary loss to promote invariance to distributional shift in representations. Hansen and Wang [2021] propose a soft-data augmentation method (SODA) by adding an auxiliary self-supervised learning phase to SAC [Haarnoja et al., 2018], similar to BYOL [Grill et al., 2020].

Representation learning. Higgins et al. [2017b] demonstrate zero-shot adaptation to unseen configurations in testing environments, using a β-VAE [Higgins et al., 2017a] to learn disentangled representations. Fan and Li [2021] jointly maximize the mutual information between sequences of observations to remove the task-irrelevant information. Fu et al. [2021] learn a disentangled world model that separates reward-corrlated features from background. Wang et al. [2021] extract, using visual attention, the observation foreground to provide background invariant inputs to the policy learner. Raileanu and Fergus [2021] separate the actor from the critic in the agent's network architecture and add an adversarial auxiliary objective on the actor's representations to remove the information needed to estimate the value function that is not irrelevant to a general policy. Zhang et al. [2020] train an encoder to project states so that their distances match with the bisimulation distances in state space. Other recent works use behavioral similarities combined with contrastive learning [Agarwal et al., 2020] or clustering [Mazoure et al., 2022] to map behaviorally similar observations to similar representations.

Attributing decisions to inputs. Although not directly aiming at generalization, a related topic is that of *attribution*, where one wishes to identify which parts of an input are responsible for major changes in the output of a function. Intrinsically, computing attributions boils down to computing (some transformation of) the gradient of the function's output with respect to the input's components. Computational graphs of differentiable functions, such as neural networks, are particularly suited to computing attributions by using the back propagation algorithm [Simonyan et al., 2014, Springenberg et al., 2015, Smilkov et al., 2017, Selvaraju et al., 2017b, Chattopadhay et al., 2018]. When these methods are applied to images, one obtains a map which is known as a *saliency map* or *attribution map*. Such attribution maps indicate which input pixels are determining for a policy's output (or the Q-value of action a) and thus permit interpretation of the function itself, rather than its point-wise decision alone. Mousavi et al. [2016], Greydanus et al. [2018], Atrey et al. [2020] have used saliency maps to analyze and explain the behavior of RL agents. Rosynski et al. [2020] indicates in particular that guided backpropagation [Springenberg et al., 2015] provides good visualizations of RL policies across a span of environments. Most existing works, however, only exploit saliency maps as tools for interpretation. Closely related to our contribution is that of Ismail et al. [2021], who incorporate attributions into their

training process in supervised learning. Their procedure iteratively uses a binary mask computed from attributions to remove features with small and potentially noisy gradients while maximizing the similarity of model outputs for both masked and unmasked inputs. This saliency guided regularization improves the quality of gradient-based saliency explanations without interferring with training stability.

This contribution. The rationale of the method we introduce in the next section is to encourage the agent to generalize to new states, based on which pixels are identified as important decision factors. For this purpose, we perform pixel-level masking on the input image, depending on the computed attribution, and regularize the value function learning process with the difference in Q-values. This way, we encourage the value function to focus specifically on the pixels with high attribution. Leveraging data augmentation, self-supervised learning methods have demonstrated their ability to induce features that are insensitive to lighting, background, and high-frequency noise. The method we propose does not directly use data augmentation in the value function or policy update phase. Instead, it is introduced during an auxiliary phase where the augmented state's encoding is used to predict the attribution mask of the original state. This way, the encoder is encouraged to preserve information that is useful for predicting which pixels were important in the agent's decision-making. A parallel with SODA can thus be made by considering that the projector used in the BYOL objective is here replaced by a surrogate of the derivative of the value function, allowing to refine the quality of the projection and to learn which pixels and visual features are consistently important across states to predict the Q-values. The value function regularization and the self-supervised learning objective are mutually beneficial: the former outputs sharper saliency maps from the value function, that serve as better labels for the auxiliary self-supervised learning task, which in turn induces better features and better attributions. In short, we encourage the agent to pay attention to where it is looking, with the intention that this triggers more efficient learning and more interpretable output.

3 Saliency-guided Q-networks

We propose a generic saliency-guided Q-networks (SGQN) method, for visual deep reinforcement learning. In a nutshell, SGQN considers the application of the binarized attribution map as a mask over the input state and regularizes the value function learning objective with a consistency term between the Q-values of the masked and the original state images. It also defines an auxiliary self-supervised learning task that aims to match a prediction of the attribution map on an augmented image, with the attribution map of the original image. Such an auxiliary task orients the gradient descent towards features that are shared across states, as illustrated by the work of Hansen and Wang [2021]. SGQN can be combined with any value function learning objective, any attribution map computation technique, any image augmentation method for self-supervised learning, and is suited for both discrete and continuous actions. We first present SGQN as a generic enhancement of approximate value iteration methods. Then we derive a specific version built on SAC [Haarnoja et al., 2018] and on the guided backpropagation algorithm [Springenberg et al., 2015].

A vast number of deep RL algorithms belong to the family of approximate value iteration methods. Such methods build a sequence of $(Q_n)_{n\in\mathbb{N}}$ and $(\pi_n)_{n\in\mathbb{N}}$ functions that aim to asymptotically tend to Q^* and π^*. Q_{n+1} is defined as a minimizer of $L_Q = \|Q - T^{\pi_n} Q_n\|$, where T^{π_n} is the Bellman evaluation operator with respect to π_n. Then π_{n+1} is defined by applying a *greediness* operator $\mathcal{G}$ to Q_{n+1} and the process is iterated. Geist et al. [2019] showed how one could introduce regularization within the expression of L_Q, yielding the class of regularized MDPs. The classical DQN algorithm [Mnih et al., 2015] approximates the solution to L_Q by taking a number of gradient steps with respect to a target network Q_n and uses an $\arg\max$ greediness operator. When actions are continuous, actor-critic methods introduce a surrogate model of the Q_{n+1}-greedy policy, under the form of an actor network $\pi_{n+1} = \mathcal{G}(Q_{n+1})$ obtained by gradient ascent. In what follows, we denote by $L_Q(\theta)$ the loss minimized by a generic learning procedure for Q_θ, based on the Bellman operator, independently of whether it is regularized, uses double critics, etc. Similarly, we note $L_\pi(\theta)$ the loss minimized to yield a greedy policy π_θ, when applicable.

We denote by $M(Q, s, a)$ an attribution map for $Q(s, a)$, in the space of images $\mathcal{S}$. Vanilla grad [Simonyan et al., 2014] for instance will compute such a map under the form $M(Q, s, a) = \partial Q(s, a)/\partial s$, while guided backpropagation [Springenberg et al., 2015] will mask out negative gradients, yielding a different attribution map. We note $M_\rho(Q, s, a)$ the binarized value attribution map where $M_\rho(Q, s, a)_j = 1$ if attribution pixel $M(Q, s, a)_j$ belongs to the ρ-quantile of highest values for $M(Q, s, a)$, and 0 otherwise.

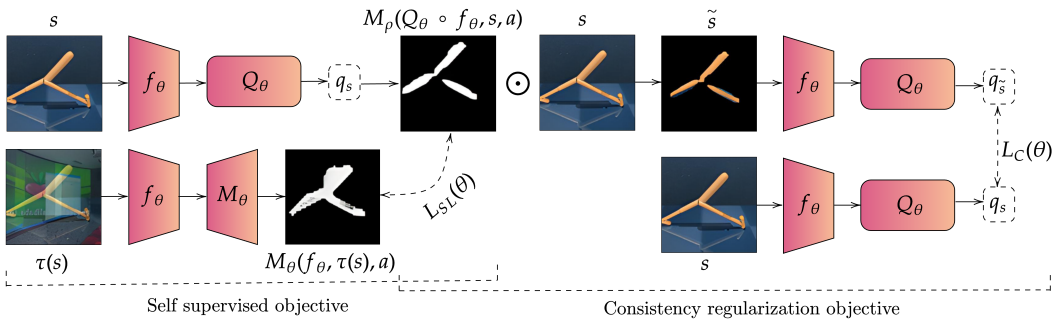


Figure 1: SGQN losses. The L_{SL} self-supervised loss trains f_θ so that $M_\theta(f_\theta(\tau(s)), a)$ predicts $M_\rho(Q_\theta \circ f_\theta, s, a)$. In turn, the L_C consistency loss pushes $Q_\theta \circ f_\theta$ to only depend on salient pixels.

The proposed method is built on a classical Q-network architecture. The value function is divided into 2 parts: an encoder $f_\theta : \mathcal{S} \to \mathcal{Z}$ and a Q-function $Q_\theta : \mathcal{Z} \times \mathcal{A} \to \mathbb{R}$ built on top of this encoder. We add a decoder function M_θ after the features encoder f_θ, such that $M_\theta(f_\theta(s), a)$ aims to predict the attribution map of $Q_\theta(f_\theta(s), a)$. Many algorithms require defining double critics [Fujimoto et al., 2018] or target networks f_ψ and Q_ψ which are often updated with an exponential moving average of θ [Polyak and Juditsky, 1992]. We omit them here for clarity, although their introduction in SGQN is straightforward. When needed, a policy head $\pi_\theta : \mathcal{Z} \to \mathcal{A}$ is built on top of the encoder f_θ to define the actor network. The backbone architecture and training process are summarized in Figure 1. The SGQN training procedure involves two additional objectives: a consistency objective responsible for regularizing the critic update and an auxiliary supervised learning objective.

The consistency regularization objective (Figure 1 right) $L_C(\theta) = \mathbb{E}_{s,a}[[Q_\theta(f_\theta(s), a) - Q_\theta(s \odot M_\rho(Q_\theta \circ f_\theta, s, a), a)]^2]$ (where $\odot$ denotes the Hadamard product), is added to the classical critic loss $L_Q(\theta)$ during the critic update phase. This loss function encourages the Q-network $Q_\theta \circ f_\theta$ to make its decision based in priority on the salient pixels in $M(Q, s, a)$, hence promoting consistency between the masked and original images. The new critic objective function is thus defined as $L_Q(\theta) + \lambda L_C(\theta)$.

The **self-supervised learning phase** (Figure 1 left) updates the parameters of f_θ so that given a generic image augmentation function τ, $(f_\theta(\tau(s)), a)$ contains enough information to accurately reconstruct the attribution mask $M_\rho(Q_\theta, s, a)$. This defines a self-supervised learning objective function $L_{SL}(\theta) = \mathbb{E}_{s,a}[BCE(M_\theta(f_\theta(\tau(s), a), M_\rho(Q_\theta, s, a)]$, where BCE is the binary cross entropy loss, which could be replaced by any other measure of discrepancy between attribution maps.

The interplay between these two phases acts as a virtuous circle. The consistency regularization loss, similar to that of Ismail et al. [2021], pushes the network to focus its decision on a selected set of pixels (hence relying on the assumption that initial saliency maps are reasonably good). This enhances the contrast between pixels in the gradient image, and thus yields sharp saliency maps, even before binarization. These maps serve as a target labels during the self-supervised learning phase; since they are less noisy than without the consistency loss, they provide a stronger incentive to encode the information of which pixels are important, within f_θ. In turn, as exemplified by Hansen and Wang [2021] and Grill et al. [2020], the features obtained by the self-supervised learning procedure tend to be insensitive to background, noise, or exogenous conditions, and provide features that are shared across observations. These features benefit from the better labels (less irrelevant pixels in the attribution map). Finally, this helps provide good pixel attributions that will be used in the minimization of the consistency loss during the critic phase. Appendix G proposes an extended discussion on this virtuous circle.

Note that the thresholding operation performed to obtain M_ρ is not strictly necessary, either in the consistency loss or in the self-supervised learning one. Instead of a hard thresholding, one could turn to a normalization of the attribution map, such as a softmax for instance. Such a soft-attribution image remains fully compatible with SGQN. The choice to keep the thresholded ρ-quantile mask is motivated by the arguments of Ismail et al. [2021] who extensively study such variations and conclude to the benefits of this binarized mask. One could also remark that SGQN does not require to use the target network during the self-supervised learning phase, which contrasts with the choices of SODA or BYOL and makes the method somewhat more versatile.

(a) Training (b) "*Video easy*" distribution shift (c) "*Video hard*" distribution shift

Figure 2: Examples of training and testing environments

Algorithm 1 presents the pseudo-code of combining SGQN with SAC, yielding an SG-SAC algorithm. Note that, for the sake of simplicity, we write θ for the full set of network parameters, which are thus shared by the encoder, the Q-value head, the policy head, and the attribution reconstruction head.

Algorithm 1: Saliency-guided SAC (changes to SAC in blue)

Parameters: frequency of auxiliary updates N_{SL}, attribution quantile value ρ, learning rate α, data augmentation function τ.

for *each interaction time step* **do**

$a, s' \sim \pi_\theta(\cdot \mid f_\theta(s)), \mathcal{P}(\cdot \mid s, a)$ `// Sample a transition`

$\mathcal{B} \leftarrow \mathcal{B} \cup \{(s, a, \mathcal{R}(s, a), s')\}$ `// Add transition to replay buffer`

$\{s_i, a_i, r(s_i, a_i), s'_i\}_{i \in [1,N]} \sim \mathcal{B}$ `// Sample a mini-batch of transitions`

$\theta \leftarrow \theta - \alpha \nabla_\theta L_Q(\theta) + \lambda L_C(\theta)$ `// Critic update phase`

$\theta \leftarrow \theta - \alpha \nabla_\theta L_\pi(\theta)$ `// Actor update`

Every N_{SL} steps: $\theta \leftarrow \theta - \alpha \nabla_\theta L_{SL}(\theta)$ `// Self-supervised learning`

Note: L_Q and L_π are as defined by Haarnoja et al. [2018], temperature update, double critics and target network updates are omitted here for clarity.

4 Experimental results and discussion

This section evaluates SGQN's training efficiency, generalization capabilities, and policy interpretability. It also discusses the different design choices made along the way. We compare our approach with current state-of-the-art methods for generalization in continuous actions RL (RAD [Laskin et al., 2020b], DrQ [Yarats et al., 2020], SODA [Hansen and Wang, 2021], SVEA [Hansen et al., 2021]) on five environments from the DMControl Generalization Benchmark (DMControl-GB) [Hansen and Wang, 2021]. The DMControl-GB presents a variety of vision-based continuous control tasks based on the Deepmind control suite. Agents are trained in a fixed background environment and evaluated under two challenging distribution shifts, consisting in replacing the training background with natural videos. Figure 2 illustrates the effects of both domain shifts. All the compared methods herein are variants of SAC, for which we use the same architecture for all agents. These methods all use data augmentation in one of their stages. Following the experimental protocol of the competing approaches, we used a random overlay augmentation [Hansen and Wang, 2021] (consisting in blending together original observations with random images from the Places365 dataset [Zhou et al., 2017]) for all the methods except for RAD and DrQ, for which we used random crops and random shifts respectively, as it is reported as producing the best results [Laskin et al., 2020b, Yarats et al., 2020]. We trained all agents for $500\,000$ steps using the vanilla training environment with no visual variation. Appendix A, B, and E discuss all the hyperparameters, network architectures, and implementation choices used for this benchmark. Appendix C includes extra experimental results on DMControl-GB and Appendix D provides additional results on a vision-based robotic environment.

SGQN improves value iteration in the training domain. We first compare the performance in the training domain, with no visual distractions, of SAC and SGQN, on five environments from the DMControl-GB (Figure 3). The SGQN agents outperform, by a considerable margin, the SAC agents both in terms of asymptotic performance and sample efficiency on 4 out of the 5 environments. In addition to obtaining better results, the variance of the agents trained with SGQN is also significantly lower than that of the agents trained with SAC, demonstrating that the enhancements employed in SGQN have a beneficial effect on the stability of the training, regardless of the ability to generalize across domains.

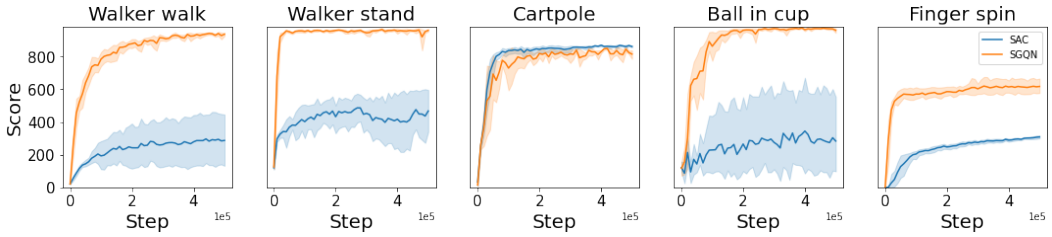


Figure 3: Comparison of SAC and SGQN training learning curves

Benchmark	Environment	SAC	DrQ	RAD	SODA	SVEA	SGQN	Δ
Easy	Walker walk	245 ± 165	747 ± 21	608 ± 92	771 ± 66	828 ± 66	$\mathbf{910 \pm 24}$	$\mathbf{+82(10\%)}$
	Walker stand	389 ± 131	926 ± 30	879 ± 64	965 ± 7	$\mathbf{966 \pm 5}$	955 ± 9	$\mathbf{-11(1\%)}$
	Ball in cup	192 ± 157	380 ± 188	363 ± 158	939 ± 10	908 ± 55	$\mathbf{950 \pm 24}$	$\mathbf{+11(1\%)}$
	Cartpole	474 ± 26	350 ± 83	391 ± 66	678 ± 120	702 ± 80	$\mathbf{717 \pm 35}$	$\mathbf{+15(2\%)}$
	Finger spin	152 ± 8	313 ± 180	334 ± 54	535 ± 52	537 ± 11	$\mathbf{609 \pm 61}$	$\mathbf{+72(13\%)}$
	Average	292	551	515	777	785	**828**	$\mathbf{+(5\%)}$
Hard	Walker walk	122 ± 47	121 ± 52	80 ± 10	312 ± 32	385 ± 63	$\mathbf{739 \pm 21}$	$\mathbf{+354(92\%)}$
	Walker stand	231 ± 57	252 ± 57	229 ± 45	736 ± 132	747 ± 43	$\mathbf{851 \pm 24}$	$\mathbf{+104(14\%)}$
	Ball in cup	101 ± 37	100 ± 40	98 ± 40	381 ± 163	498 ± 174	$\mathbf{782 \pm 57}$	$\mathbf{+284(57\%)}$
	Cartpole	153 ± 22	128 ± 19	117 ± 22	339 ± 87	392 ± 37	$\mathbf{526 \pm 41}$	$\mathbf{+134(34\%)}$
	Finger spin	25 ± 6	25 ± 36	15 ± 6	221 ± 48	174 ± 39	$\mathbf{540 \pm 53}$	$\mathbf{+319(144\%)}$
	Average	127	130	108	396	437	**688**	$\mathbf{+(57\%)}$

Table 1: Performance on *video easy* and *video hard* testing levels. Δ = difference with second best.

SGQN improves generalization. We assess the zero-shot generalization ability of SGQN on the *video easy* and *video hard* benchmarks from the DMControl-GB. The easy version only replaces the background of the training image with a distracting image, while the hard version also replaces the ground and the shadows (Figure 2). We report the average sum of rewards after 500 000 training steps for the *video easy* benchmark in the top part of Table 1. Agents trained with SGQN outperform agents trained with other state-of-the-art methods on all tasks but one (where it is on par with other agents), thus demonstrating the generalization capabilities induced by the method. By removing the ground and shadows, the *video hard* benchmark (bottom part of Table 1) causes a larger, more confusing, and more challenging distributional shift. All the competitors of SGQN experience a radical decrease in their generalization performance. SGQN is significantly less impacted and outperforms all its competitors with an average margin over the second-best of 57% on all environments, and a gain range of 14 to 144% across environments. Figure 4 reports the evolution of each agent's score on the *video hard* environments, along training. SODA and SVEA's scores drop drastically when the ground and shadows are removed. SGQN is less sensitive to this change, notably through the consistency loss, which encourages agents to make decisions based on the subsets of pixels they deem most interesting. Overall, the interplay of the two phases of SGQN seems to be key to a major leap forward in terms of generalization gap in difficult environments, setting a new reference state-of-the-art.

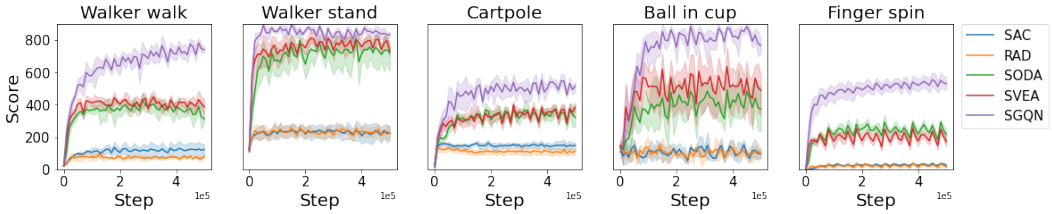


Figure 4: Performance on *video hard* testing levels.

SGQN yields sharp saliency maps. We use guided backpropagation to visually compare the SGQN agents' ability to discriminate the essential information with that of other agents. Figure 5 shows an example of the binarized attribution maps for all agents in a *video hard* state. While the other agents seem to be disturbed by background elements and retain some dependency on background pixels in their decision, the attributions of the agent trained with SGQN are precisely located on the important information, hence suggesting better generalization potential.

(a) Observation (b) SAC (c) RAD (d) SODA (e) SVEA (f) SGQN

Figure 5: Example of attributions in *video hard* levels

SGQN is interpretable by design. In all our experiments, we trained the SGQN agents using guided backpropagation. We emphasize that any other attribution method could be used instead. Some of these methods are expensive and require several forward (or backward) passes within the network (e.g., RISE [Petsiuk et al., 2018], or the work of Fel et al. [2021]) to yield attribution maps which explain the agent's decision. In contrast, SGQN's auxiliary phase trains a predictor to estimate the most important pixels according to the chosen attribution method. Therefore, this predictor is a surrogate of the attribution method used during training. It allows identifying the essential image features that condition the agent's decision in the same forward pass as the prediction of the action itself, without incurring the cost of one or several costly additional backward or forward passes. Figure 6 illustrates the proximity between the actual saliency map and the attribution surrogate model M_θ. This makes SGQN both self-aware (of its own saliency maps) and intrinsically interpretable from a human perspective, with no computational overhead at evaluation time.

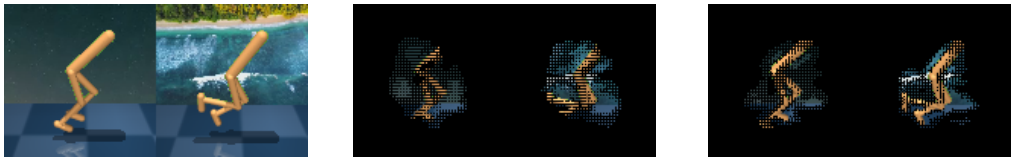
(a) Observation (b) $M_\rho(Q_\theta, s, a)$ (c) $s \odot M_\rho(Q_\theta, s, a)$ (d) $M_\theta(s, a)$

Figure 6: Comparison between the true saliency map (b), the masked image (c) used in the consistency regularization term, and the estimated saliency map (d) in a *video hard* level.

Ablation study. SGQN relies on two enhancements of vanilla approximate value iteration: the auxiliary self-supervised learning phase and the consistency regularization term in the critic's loss. We perform an ablation study to assess their individual contribution and Table 2 reports the average sum of rewards in the training domain and for zero-shot generalization in all environments (Appendix F takes a different perspective and compares SGQN with the combination of SVEA and SODA). Individually, each of these features greatly improves both training and zero-shot generalization performance on all environments. The auxiliary self-supervised learning phase provides the most significant performance gains over vanilla SAC. The average training performance on all environments of agents trained with this auxiliary objective is more than 87% higher than that of agents trained with vanilla SAC. The same applies to the performance in zero-shot generalization, which increases by more than 141% on *video easy* and by more than 182% on *video hard* environments. One can note that agents trained with our self-supervised learning objective obtain performance of the same order of magnitude as SODA agents (Table 1). Recall that SODA relies on the BYOL [Grill et al., 2020] self-supervised feature learning procedure, whose target labels differ notably from those proposed herein. The reported performance, compared to that of SODA suggests that attribution maps constitute a good labeling function that could be considered in the more general context of self-supervised learning. To a slightly lesser extent, adding the consistency loss to SAC also significantly improves its performance. The average score obtained improves by more than 49% on the training domain and by 86% and 166% respectively on the *video easy* and *video hard* domains. Similarly to the results obtained by Ismail et al. [2021] in supervised learning, the regularization of the critic's loss with the consistency term sharpens the attribution maps obtained (Figure 7). In SGQN, these fine-grained attributions provide higher quality labels to the auxiliary self-supervised learning phase, thus yielding significant performance improvements in training (+94% on average, range up to +242%), *video easy* generalization (+183% on average, range up to +399%), and *video hard* generalization (+440% on average, range up to +1977%).

Environment	benchmark	SAC	SAC+Consistency	SAC+Self learning	SGQN
Walker walk	train	287 ± 165	449 ± 100 (**+56%**)	934 ± 28 (**+225%**)	937 ± 12 (**+226%**)
	easy	245 ± 165	423 ± 96 (**+73%**)	844 ± 53 (**+244%**)	910 ± 24 (**+271%**)
	hard	122 ± 47	344 ± 87 (**+182%**)	226 ± 48 (**+85%**)	739 ± 21 (**+505%**)
Walker stand	train	467 ± 162	857 ± 120 (**+84%**)	957 ± 11 (**+105%**)	960 ± 9 (**+106%**)
	easy	389 ± 131	846 ± 107 (**+117%**)	944 ± 14 (**+143%**)	955 ± 9 (**+145%**)
	hard	231 ± 57	696 ± 150 (**+201%**)	769 ± 32 (**+233%**)	851 ± 24 (**+268%**)
Ball in cup	train	284 ± 329	755 ± 261 (**−8%**)	967 ± 1 (**+240%**)	971 ± 7 (**+242%**)
	easy	192 ± 157	440 ± 214 (**+129%**)	705 ± 43 (**+267%**)	950 ± 24 (**+399%**)
	hard	101 ± 37	190 ± 63 (**+88%**)	203 ± 122 (**+100%**)	778 ± 75 (**+670%**)
Cartpole	train	861 ± 9	836 ± 33 (**−3%**)	703 ± 82 (**−20%**)	815 ± 35 (**−14%**)
	easy	477 ± 30	661 ± 112 (**+39%**)	534 ± 54 (**+12%**)	717 ± 35 (**+50%**)
	hard	154 ± 25	261 ± 235 (**+69%**)	294 ± 7 (**+91%**)	526 ± 41 (**+242%**)
Finger spin	train	309 ± 12	392 ± 128 (**+26%**)	571 ± 10 (**+85%**)	618 ± 56 (**+100%**)
	easy	157 ± 8	343 ± 88 (**+92%**)	468 ± 29 (**+281%**)	609 ± 61 (**+287%**)
	hard	26 ± 6	201 ± 74 (**+673%**)	209 ± 59 (**+681%**)	540 ± 53 (**+1977%**)
Average	train	442	658 (**+49%**)	826 (**+87%**)	860 (**+94%**)
	easy	292	542 (**+86%**)	704 (**+141%**)	82 (**+183%**)
	hard	127	338 (**+166%**)	359 (**+182%**)	686 (**+440%**)

Table 2: Ablation study. Percentages indicate variations compared to vanilla SAC.



(a) Observation (b) Without consistency loss (c) With consistency loss

Figure 7: Comparison of SGQN attributions with and without consistency loss

5 Conclusion

The ability to filter out confounding variables is a long-standing goal in machine learning. For visual reinforcement learning, it is a pre-requisite for real-world deployment of learned policies, since we want to avoid at all costs situations where an agent makes the wrong decision due to distracting visual factors. In this work, we introduced saliency-guided Q-networks (SGQN), a generic method for visual reinforcement learning, that is compatible with any value function learning method. SGQN relies on the positive interaction of two core mecanisms of self-supervised learning and attribution consistency that jointly encourage the RL agent to be self-aware of the decisive factors that condition its value function. We implement an SGQN agent based on the soft actor-critic algorithm, and evaluate it on the DMControl generalization benchmark. This agent displays a dramatically more efficient learning curve than vanilla SAC on the various environments it is trained on. Most importantly, the policy it learns closes the generalization gap on environments that include confusing and distracting visual features, setting a new reference in terms of generalization performance. Since they rely on self-awareness of important pixels, SGQN agents are also very interpretable, in the sense that they provide a prediction of their own saliency maps, with no computational overhead.

The introduction of SGQN is an exciting milestone in RL generalization, and we wish to conclude this contribution by highlighting some limitations and perspectives for research that we believe are beneficial to the community. Attribution maps seem to be an efficient proxy for encouraging causal relationships within policies, but they are strongly grounded in an anthropomorphic point of view of what a visual policy should be. Extending their definition to more abstract notions of attribution is still a challenge and begs for important contributions, both theoretical and algorithmic. Similarly, such attribution maps appear to be relevant self-supervised learning targets, in order to learn good features for RL agents. Exploring whether this still holds for different tasks than RL is an open question. It is likely that one could design variations of SGQN that perform better than SG-SAC. The extension of SGQN agents to discrete action agents (e.g. DQN), or policy gradient methods (e.g. PPO) is a promising perspective in itself. Although generic and grounded in sound algorithmic mechanisms, the losses introduced by SGQN lack a formal connection to some measure of the generalization gap. Such a connection could provide insights to better self-aware, explainable agents with improved generalization capabilities. Finally, using SGQN as a building brick, among all those required to bridge the gap between simulation and real-world applications, is an exciting perspective.

References

Rishabh Agarwal, Marlos C Machado, Pablo Samuel Castro, and Marc G Bellemare. Contrastive behavioral similarity embeddings for generalization in reinforcement learning. In *International Conference on Learning Representations*, 2020.

Akanksha Atrey, Kaleigh Clary, and David Jensen. Exploratory not explanatory: Counterfactual analysis of saliency maps for deep reinforcement learning. In *International Conference on Learning Representations*, 2020.

Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3:397–422, 2002.

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

David Bertoin and Emmanuel Rachelson. Local feature swapping for generalization in reinforcement learning. In *International Conference on Learning Representations*, 2022.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Aditya Chattopadhay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In *IEEE winter conference on applications of computer vision (WACV)*, pages 839–847, 2018.

Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *International Conference on Machine Learning*, pages 1282–1289, 2019.

Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *International conference on machine learning*, pages 2048–2056, 2020.

Jiameng Fan and Wenchao Li. Dribo: Robust deep reinforcement learning via multi-view information bottleneck. *arXiv preprint arXiv:2102.13268*, 2021.

Linxi Fan, Guanzhi Wang, De-An Huang, Zhiding Yu, Li Fei-Fei, Yuke Zhu, and Animashree Anandkumar. Secant: Self-expert cloning for zero-shot generalization of visual policies. In *International Conference on Machine Learning*, 2021.

Jesse Farebrother, Marlos C Machado, and Michael Bowling. Generalization and regularization in dqn. In *NeurIPS Deep Reinforcement Learning Workshop*, 2018.

Thomas Fel, Rémi Cadène, Mathieu Chalvidal, Matthieu Cord, David Vigouroux, and Thomas Serre. Look at the variance! efficient black-box explanations with sobol-based sensitivity analysis. *Advances in Neural Information Processing Systems*, 34, 2021.

Xiang Fu, Ge Yang, Pulkit Agrawal, and Tommi Jaakkola. Learning task informed abstractions. In *International Conference on Machine Learning*, pages 3480–3491. PMLR, 2021.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596, 2018.

Matthieu Geist, Bruno Scherrer, and Olivier Pietquin. A theory of regularized markov decision processes. In *International Conference on Machine Learning*, pages 2160–2169, 2019.

Samuel Greydanus, Anurag Koul, Jonathan Dodge, and Alan Fern. Visualizing and understanding atari agents. In *International conference on machine learning*, pages 1792–1801, 2018.

Jake Grigsby and Yanjun Qi. Measuring visual generalization in continuous control from pixels. *arXiv preprint arXiv:2010.06740*, 2020.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Valko Michal. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in Neural Information Processing Systems*, 33: 21271–21284, 2020.

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*, pages 1861–1870, 2018.

Nicklas Hansen and Xiaolong Wang. Generalization in reinforcement learning by soft data augmentation. In *International Conference on Robotics and Automation*, 2021.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Stabilizing deep q-learning with convnets and vision transformers under data augmentation. In *Conference on Neural Information Processing Systems*, 2021.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. 2017a.

Irina Higgins, Arka Pal, Andrei Rusu, Loic Matthey, Christopher Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Alexander Lerchner. Darla: Improving zero-shot transfer in reinforcement learning. In *International Conference on Machine Learning*, pages 1480–1490, 2017b.

Maximilian Igl, Kamil Ciosek, Yingzhen Li, Sebastian Tschiatschek, Cheng Zhang, Sam Devlin, and Katja Hofmann. Generalization in reinforcement learning with selective noise injection and information bottleneck. *Advances in Neural Information Processing Systems*, 32:13978–13990, 2019.

Aya Abdelsalam Ismail, Hector Corrada Bravo, and Soheil Feizi. Improving deep learning interpretability by saliency guided training. *Advances in Neural Information Processing Systems*, 34, 2021.

Rishabh Jangir, Nicklas Hansen, Sambaran Ghosal, Mohit Jain, and Xiaolong Wang. Look closer: Bridging egocentric and third-person views with transformers for robotic manipulation. *IEEE Robotics and Automation Letters*, pages 1–1, 2022. doi: 10.1109/LRA.2022.3144512.

Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay. In *International Conference on Machine Learning*, pages 4940–4950, 2021.

Arthur Juliani, Ahmed Khalifa, Vincent-Pierre Berges, Jonathan Harper, Hunter Henry, Adam Crespi, Julian Togelius, and Danny Lange. Obstacle tower: A generalization challenge in vision. *Control, and Planning. arXiv e-prints*, 2019.

Michael Laskin, Aravind Srinivas, and Pieter Abbeel. CURL: Contrastive unsupervised representations for reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*, pages 5639–5650, 2020a.

Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. *Advances in Neural Information Processing Systems*, 33, 2020b.

Kimin Lee, Kibok Lee, Jinwoo Shin, and Honglak Lee. Network randomization: A simple technique for generalization in deep reinforcement learning. In *International Conference on Learning Representations*, 2020.

Marlos C. Machado, Marc G. Bellemare, Erik Talvitie, Joel Veness, Matthew J. Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 61:523–562, 2018.

Bogdan Mazoure, Ahmed M Ahmed, R Devon Hjelm, Andrey Kolobov, and Patrick MacAlpine. Cross-trajectory representation learning for zero-shot generalization in RL. In *International Conference on Learning Representations*, 2022.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Sajad Mousavi, Michael Schukat, Enda Howley, Ali Borji, and Nasser Mozayani. Learning to predict where to look in interactive environments using deep recurrent q-learning. *arXiv preprint arXiv:1612.05753*, 2016.

Charles Packer, Katelyn Gao, Jernej Kos, Philipp Krähenbühl, Vladlen Koltun, and Dawn Song. Assessing generalization in deep reinforcement learning. *arXiv preprint arXiv:1810.12282*, 2018.

Vitali Petsiuk, Abir Das, and Kate Saenko. RISE: Randomized input sampling for explanation of black-box models. In *29th British Machine Vision Conference*, 2018.

Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Roberta Raileanu and Rob Fergus. Decoupling value and policy for generalization in reinforcement learning. In *International Conference on Machine Learning*, pages 8787–8798, 2021.

Roberta Raileanu, Maxwell Goldstein, Denis Yarats, Ilya Kostrikov, and Rob Fergus. Automatic data augmentation for generalization in reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021.

Matthias Rosynski, Frank Kirchner, and Matias Valdenegro-Toro. Are gradient-based saliency maps useful in deep reinforcement learning? In *"I Can't Believe It's Not Better!" NeurIPS workshop*, 2020.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017a.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017b.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Workshop at International Conference on Learning Representations*, 2014.

Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

Xingyou Song, Yiding Jiang, Stephen Tu, Yilun Du, and Behnam Neyshabur. Observational overfitting in reinforcement learning. In *International Conference on Learning Representations*, 2019.

J Springenberg, Alexey Dosovitskiy, Thomas Brox, and M Riedmiller. Striving for simplicity: The all convolutional net. In *ICLR (workshop track)*, 2015.

Austin Stone, Oscar Ramirez, Kurt Konolige, and Rico Jonschkowski. The distracting control suite–a challenging benchmark for reinforcement learning from pixels. *arXiv preprint arXiv:2101.02722*, 2021.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

Kaixin Wang, Bingyi Kang, Jie Shao, and Jiashi Feng. Improving generalization in reinforcement learning with mixture regularization. In *NeurIPS*, 2020.

Xudong Wang, Long Lian, and Stella X Yu. Unsupervised visual attention and invariance for reinforcement learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6677–6687, 2021.

Shimon Whiteson, Brian Tanner, Matthew E Taylor, and Peter Stone. Protecting against evaluation overfitting in empirical reinforcement learning. In *IEEE symposium on adaptive dynamic programming and reinforcement learning (ADPRL)*, pages 120–127, 2011.

Denis Yarats, Ilya Kostrikov, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations*, 2020.

Zhecheng Yuan, Guozheng Ma, Yao Mu, Bo Xia, Bo Yuan, Xueqian Wang, Ping Luo, and Huazhe Xu. Don't touch what matters: Task-aware lipschitz data augmentationfor visual reinforcement learning. *arXiv preprint arXiv:2202.09982*, 2022.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

Amy Zhang, Nicolas Ballas, and Joelle Pineau. A dissection of overfitting and generalization in continuous reinforcement learning. *arXiv preprint arXiv:1806.07937*, 2018a.

Amy Zhang, Yuxin Wu, and Joelle Pineau. Natural environment benchmarks for reinforcement learning. *arXiv preprint arXiv:1811.06032*, 2018b.

Amy Zhang, Rowan Thomas McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. In *International Conference on Learning Representations*, 2020.

Chiyuan Zhang, Oriol Vinyals, Remi Munos, and Samy Bengio. A study on overfitting in deep reinforcement learning. *arXiv preprint arXiv:1804.06893*, 2018c.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018d.

Chenyang Zhao, Olivier Sigaud, Freek Stulp, and Timothy M Hospedales. Investigating generalisation in continuous deep reinforcement learning. *arXiv preprint arXiv:1902.07015*, 2019.

Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

Kaiyang Zhou, Yongxin Yang, Yu Qiao, and Tao Xiang. Domain generalization with mixstyle. In *International Conference on Learning Representations*, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Look where you look! Saliency-guided Q-networks for visual RL tasks

—

Appendix

A Network architecture and training hyperparameters

All the methods evaluated in Section 4 are variants of SAC. We based our implementation of SAC, RAD, SODA, and SVEA on the ones proposed by Nicklas Hansen, available at `https://github.com/nicklashansen/dmcontrol-generalization-benchmark`. To ensure a fair comparison, we used the same network architecture and hyperparameters for all agents.

Networks architectures. The shared encoder f_θ is composed of a stack of 11 convolutional layers, each with 32 filters of 3×3 kernels, no padding, stride of 2 for the first one and stride of 1 for all others, yielding a final feature map of dimension $32 \times 21 \times 21$ (inputs have dimension $84 \times 84 \times 3$). The policy head π_θ and the value function head Q_θ consist of independent fully connected networks, each composed of a linear projection of dimension 100 with layer normalization, followed by 3 linear layers with 1024 hidden units. SGQN also uses a predictor head M_θ responsible for reconstructing the attribution mask $M_\rho(Q_\theta, s, a)$. The M_θ network follows the 100-unit layer of Q_θ and has 6 layers. It is composed of a first linear layer projecting the 100-dimensional embedding to $32 \times 21 \times 21$ features, then followed by two convolutional+upsampling blocks, each having 64 filters of 3×3 kernels (padding of 1 to preserve the feature map size) and an upsampling factor of 2, and finally a last convolutional layer with 9 filters of 3×3 kernels (padding of 1). Figure 8 illustrates SGQN agents' architecture.

Hyperparameters. SGQN introduces three additional hyperparameters to SAC: the quantile value ρ used to binarize the attribution masks, the data augmentation function τ used during the self-supervised learning updates, and the frequency of these auxiliary updates N_{SL}. Table 3 summarizes the hyperparameters used in all experiments.

Hyperparameter	Value
Frame rendering	$84 \times 84 \times 3$
Stacked frames	3
Action repeat	4
Discount factor γ	0.99
Episode length	1,000
Number of frames	500,000
Replay buffer size	500,000
Optimizer (θ of SAC)	Adam ($lr = 1\text{e}-3, \beta_1 = 0.9, \beta_2 = 0.999$)
Optimizer (θ of self supervised learning updates)	Adam ($lr = 3\text{e}-4, \beta_1 = 0.9, \beta_2 = 0.999$)
Optimizer (α of SAC)	Adam ($lr = 1\text{e}-4, \beta_1 = 0.5, \beta_2 = 0.999$)
Batch size	128
Target networks update frequency	2
Target networks momentum coefficient	0.05 (encoder), 0.01 (critic)
Auxiliary updates N_{SL} frequency	2
Data augmentation τ	Overlay [Hansen and Wang, 2021]
Quantile value ρ	0.95 (Walker walk, Walker Stand, and Finger Spin) 0.98 (Cartpole and Ball in cup)

Table 3: SAC and SGQN (in blue) hyperparameters.

B Reproducibility

All the experiments from Section 4 were run on a desktop machine (Intel i9, 10th generation processor, 64GB RAM) with a single NVIDIA RTX 3090 GPU. All scores were calculated on an average of 5 repetitions. Details about all experiments are reported in Table 4. Besides this information, we provide the full source code of our implementation and experiments, along with the data files of the experimental results we obtained. After the review phase, this code will be available at [anonymous url].

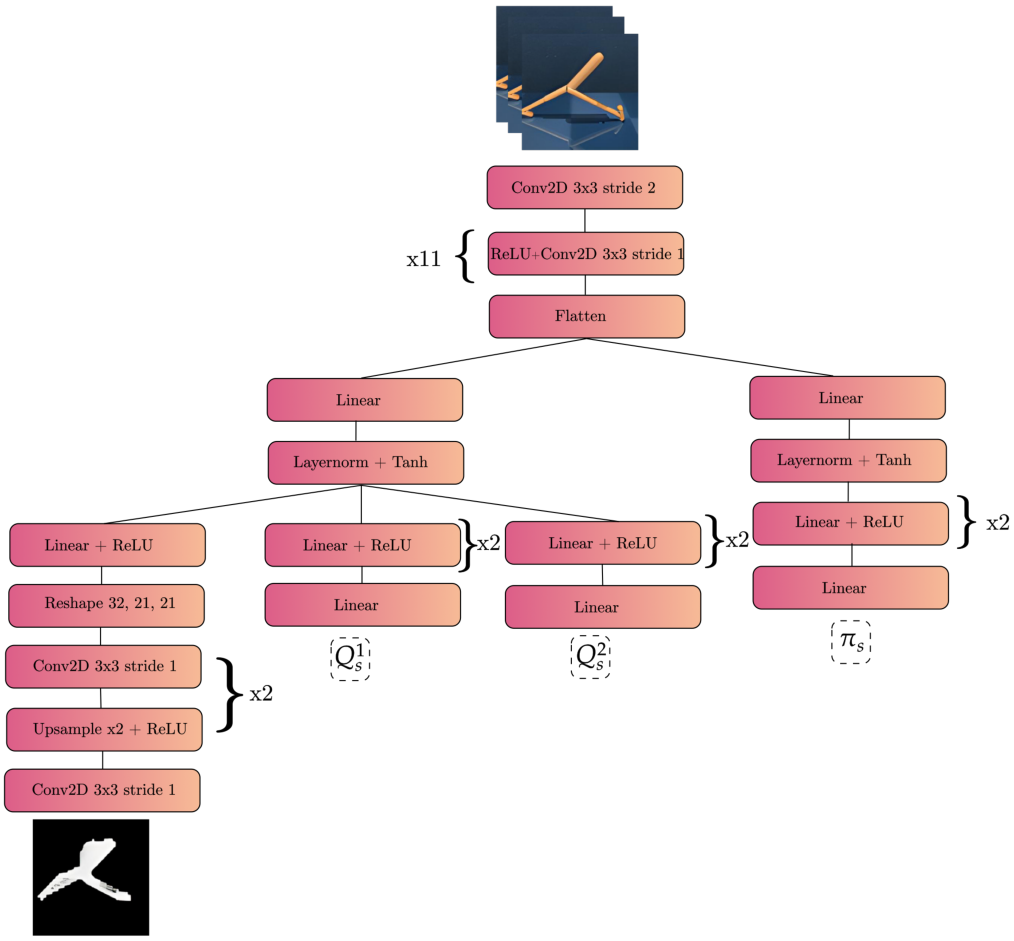


Figure 8: SGQN neural network architecture

Algorithm	Time by experiment
SAC	$\sim$ 5 hours
RAD	$\sim$ 5 hours
SODA	$\sim$ 15 hours
SVEA	$\sim$ 15 hours
SGQN	$\sim$ 15 hours
SAC + consistency	$\sim$ 5 hours
SAC + self-supervised	$\sim$ 15 hours

Table 4: Experimental setup

C Additional experimental results on DMControl-GB

For the sake of simplicity and consistency, all experiments in Section 4 have been conducted with the same set of hyperparameters (except for the quantile value ρ, recalled in Table 4). A particular attention needs to be paid to the number of times an action is repeated in the environments of the DMControl-GB, since it has an important influence on the scores reached by the agents. Hansen et al. [2021] indicate they run their experiments with an action repetition covering 4 time steps for all environments, except for Cartpole (8 time steps) and Finger spin (2 time steps). To avoid such heterogeneity across environments, we chose to report scores in Tables 1 and 2 with a constant action repetition parameter of 4 time steps for all environments. We repeated the experiments of Section 4 by setting the value of this parameter to 8 and 2 for the Cartpole and Finger spin environments

Benchmark	Environment	SAC	DrQ	RAD	SODA	SVEA	SGQN	Δ
Easy	Walker walk	245 ± 165	747 ± 21	608 ± 92	771 ± 66	828 ± 66	$\mathbf{910 \pm 24}$	$\mathbf{+82(10\%)}$
	Walker stand	389 ± 131	926 ± 30	879 ± 64	965 ± 7	$\mathbf{966 \pm 5}$	955 ± 9	$\mathbf{-11(1\%)}$
	Ball in cup	192 ± 157	380 ± 188	363 ± 158	939 ± 10	908 ± 55	$\mathbf{950 \pm 24}$	$\mathbf{+11(1\%)}$
	Cartpole	474 ± 26	350 ± 83	391 ± 66	678 ± 120	702 ± 80	$\mathbf{717 \pm 35}$	$\mathbf{+15(2\%)}$
	Cartpole*	398 ± 60	459 ± 81	473 ± 54	742 ± 73	753 ± 45	$\mathbf{761 \pm 28}$	$\mathbf{+8(1\%)}$
	Finger spin	152 ± 8	313 ± 180	334 ± 54	535 ± 52	537 ± 11	$\mathbf{609 \pm 61}$	$\mathbf{+72(13\%)}$
	Finger spin*	206 ± 169	599 ± 62	516 ± 113	783 ± 51	723 ± 98	$\mathbf{956 \pm 26}$	$\mathbf{+173(22\%)}$
	Average*	286	622	568	836	836	**906**	$\mathbf{+70(8\%)}$
Hard	Walker walk	122 ± 47	121 ± 52	80 ± 10	312 ± 32	385 ± 63	$\mathbf{739 \pm 21}$	$\mathbf{+354(92\%)}$
	Walker stand	231 ± 57	252 ± 57	229 ± 45	736 ± 132	747 ± 43	$\mathbf{851 \pm 24}$	$\mathbf{+104(14\%)}$
	Ball in cup	101 ± 37	100 ± 40	98 ± 40	381 ± 163	498 ± 174	$\mathbf{782 \pm 57}$	$\mathbf{+284(57\%)}$
	Cartpole	153 ± 22	128 ± 19	117 ± 22	339 ± 87	392 ± 37	$\mathbf{526 \pm 41}$	$\mathbf{+134(34\%)}$
	Cartpole*	158 ± 17	136 ± 29	152 ± 29	403 ± 17	401 ± 38	$\mathbf{569 \pm 56}$	$\mathbf{+166(41\%)}$
	Finger spin	25 ± 6	25 ± 36	15 ± 6	221 ± 48	174 ± 39	$\mathbf{540 \pm 53}$	$\mathbf{+319(144\%)}$
	Finger spin*	13 ± 10	38 ± 13	39 ± 20	309 ± 49	307 ± 24	$\mathbf{822 \pm 24}$	$\mathbf{+513(166\%)}$
	Average*	125	129	119	430	468	**747**	$\mathbf{+279(60\%)}$

Table 5: Performance on *video easy* and *video hard* testing levels. Δ = difference with second best. Starred* lines correspond to environments with a modified action repetition period, as per the work of Hansen et al. [2021].

respectively as suggested by Hansen et al. [2021]. Table 5 reports the results obtained. This fine-tuning benefits almost all agents and, more particularly SGQN, which increases the gap with its competitors both in *video easy* and *video hard* domains. We report all the corresponding training curves and testing scores in Figures 9, 10, and 11. We also report a comparison of the attribution maps for all agents on all environments in *video hard* environments in Figure 12, illustrating how SGQN consistently relies on pixels that really belong to the system to control and discards confounding factors.

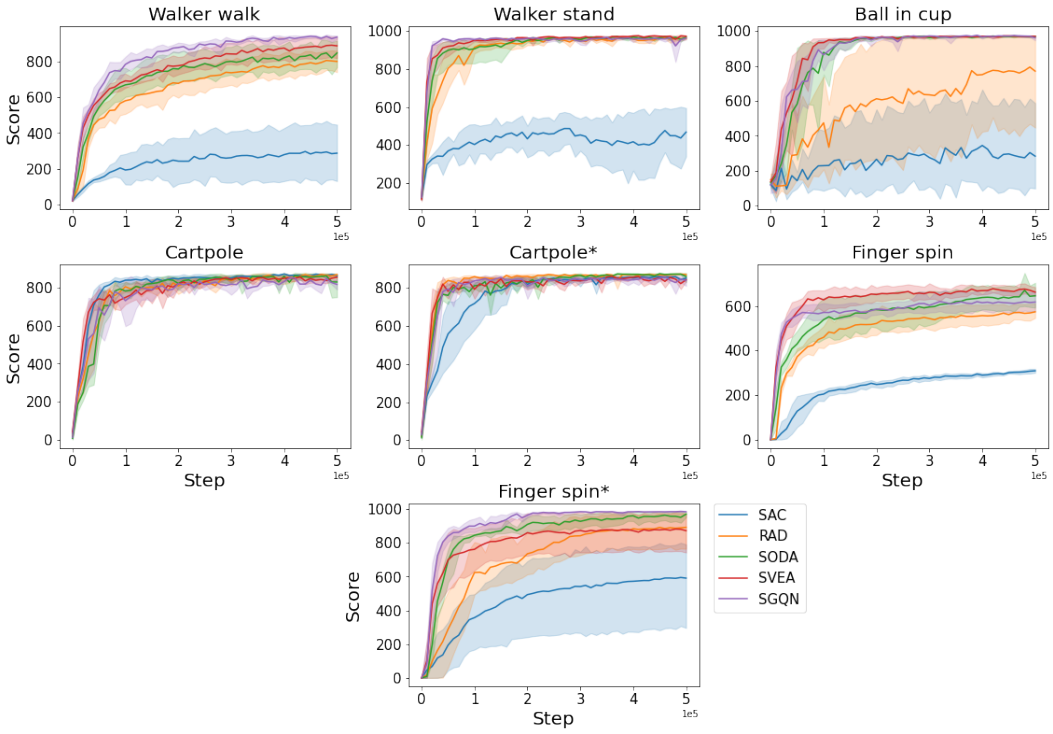


Figure 9: Performance on training levels.

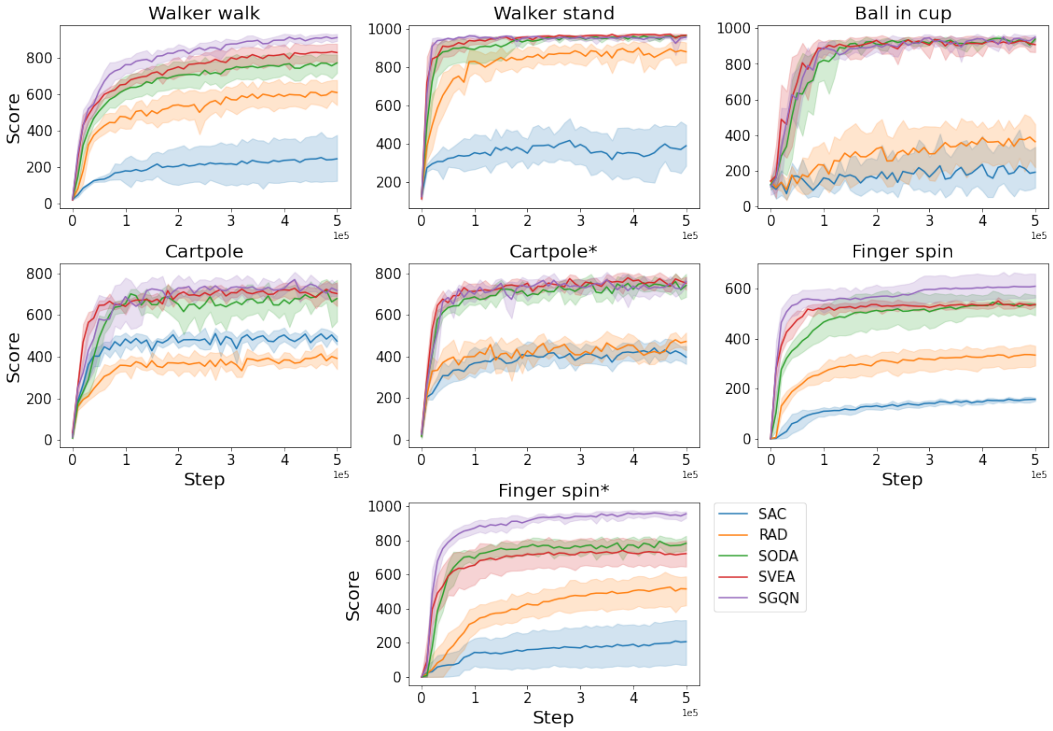


Figure 10: Performance on *video easy* testing levels.

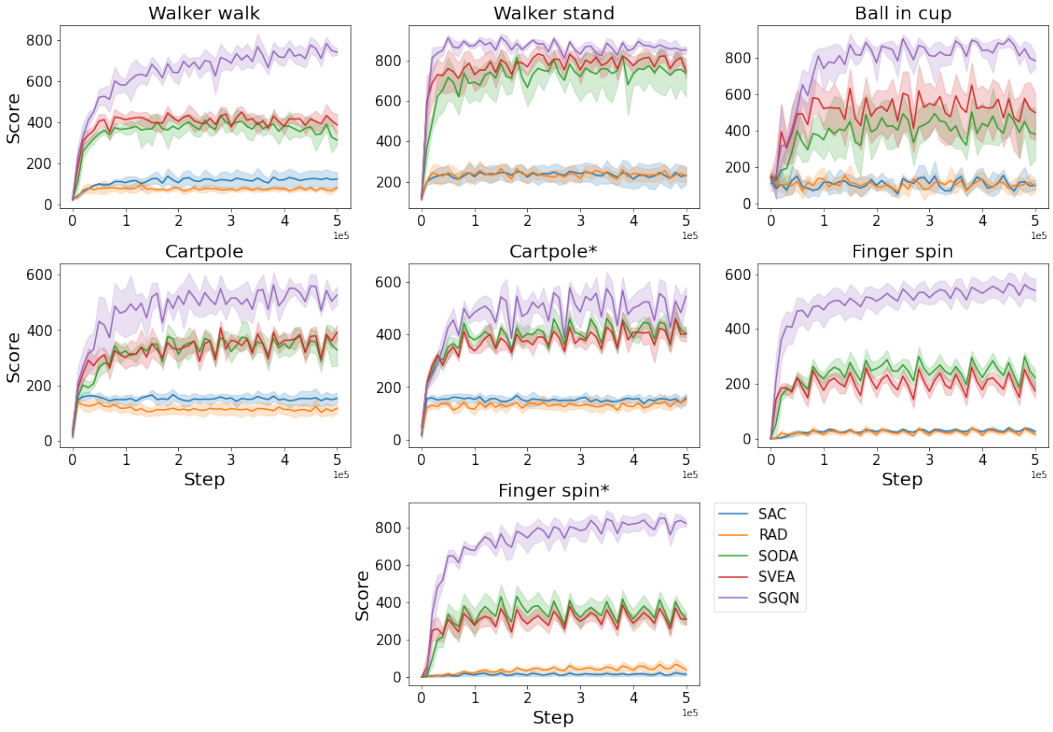


Figure 11: Performance on *video hard* testing levels.

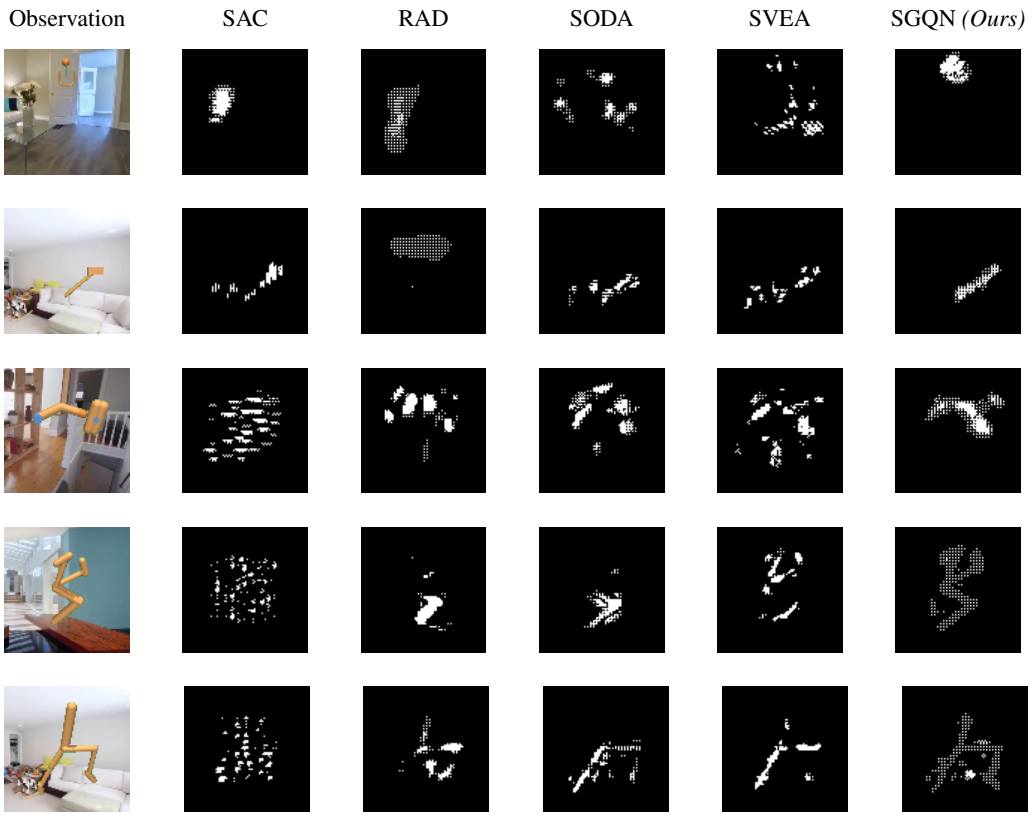


Figure 12: Binarized attribution maps in *video hard*

D Vision-based robotic manipulation experiments

To demonstrate the genericity of our method, and following the recommendation of the reviewers, we also consider two goal-reaching robotic manipulation tasks from the vision-based robotic manipulation simulator introduced in [Jangir et al., 2022]: *Reach,* a task where a robot has to reach for a goal marked by a red disc placed on a table, and *Peg in box,* a task where a robot has to insert a peg tied to its arm into a box. We modified the original simulator to include three testing environments for both tasks, similar to the training ones but with different colors and textures for the background and the table as illustrated in Figure 13. Note that no fine-tuning of hyperparameters (learning rates, quantile threshold, etc.) was performed whatsoever.

(a) Train (b) Test 1 (c) Test 2 (d) Test 3

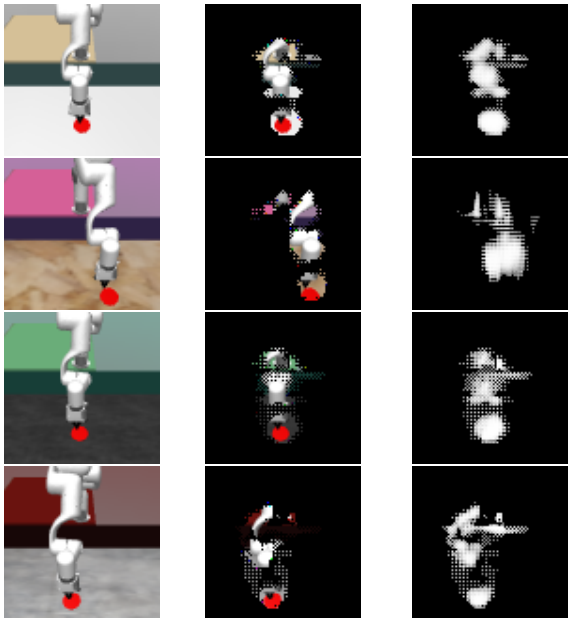
Figure 13: Examples of training and testing observation for (Peg in box)

We trained SGQN agents for 250 000 steps with a quantile threshold $\rho = 0.95$ and compared their generalization scores with those of agents trained with SAC [Haarnoja et al., 2018], SODA [Hansen and Wang, 2021] and SVEA [Hansen et al., 2021]. We used random convolutions [Lee et al., 2020] as image augmentations for all the methods except for SAC, since it produces color and texture increases that better match our test environments than the structured distractions induced by a random overlay. The results are reported in Table 6. For the *Reach* task, the agents trained with SAC, SODA, and

SVEA fail to maintain their performance when evaluated on the test environments. The agents trained with SGQN maintain performance almost identical to those obtained during training on two out of three testing environments and outperform the other agents on average by more than 124%. Figure 14 shows examples of the saliency maps obtained for the *Reach* training and testing environments. In the *Peg in box* task, the generalization scores are degraded for all the agents. Nevertheless, in two out of the three testing environments (Test 2 and 3), the agents trained with SGQN seem to be the least affected. The third environment (Test 1) seems to feature textures and colors that are particularly difficult for generalization and would require further investigation. Overall, the SGQN agents' generalization scores over the three environments are still 40% better on average than those of their competitors.

Task	Environment	SAC	SODA	SVEA	SGQN	Δ
Reach	Train	9.7 ± 22	31.8 ± 1	$\mathbf{32.2 \pm 0}$	31.8 ± 1	$-0.4(1\%)$
	Test 1	-20.9 ± 16	-30.9 ± 43	-17.6 ± 10	$\mathbf{14.4 \pm 14}$	$+32(220\%)$
	Test 2	-21.9 ± 14	-20.2 ± 29	-2.1 ± 39	$\mathbf{31.0 \pm 3}$	$+33.1(107\%)$
	Test 3	-43.2 ± 6	-68.4 ± 30	1.4 ± 29	$\mathbf{29.2 \pm} 7$	$+27.8(95\%)$
	Test Average	-28.6 ± 8	-39.9 ± 31	-6.1 ± 23	$\mathbf{24.9 \pm 6}$	$+31(124\%)$
Peg in box	Train	-46.7 ± 7	180.0 ± 1	177.5 ± 1	$\mathbf{183.9 \pm 1}$	$+3.9(2\%)$
	Test 1	-59.6 ± 26	$\mathbf{16.9 \pm 44}$	-21.3 ± 10	-72.0 ± 14	$-88.9(526\%)$
	Test 2	-60.15 ± 10	0.7 ± 30	96.8 ± 40	$\mathbf{110.7 \pm 3}$	$+13.9(12\%)$
	Test 3	-48.8 ± 17	73.6 ± 31	40.5 ± 28	$\mathbf{154.6 \pm 7}$	$+81(52\%)$
	Test Average	-56.2 ± 7	30.4 ± 31	38.6 ± 23	$\mathbf{64.4 \pm 6}$	$+25.8(40\%)$

Table 6: Performance on the robotic environments



(a) Observation (b) $s \odot M_\rho(Q_\theta, s, a)$ (c) $M_\theta(s, a)$

Figure 14: Observation (a), masked observation (b) and predicted saliency map (c) in the *Reach* task. The first row corresponds to the training environment and the last three to the test environments.

E Impact of ρ

In our experiments, the value of the mask threshold parameter ρ was selected after a quick visual search, as illustrated in Figure 15. During our experiments on DMControl-GB, we set ρ to 0.95 on all environments but Cartpole and Ball in Cup (for which the ratio of foreground/background pixels is smaller than on the others environments). To assess SGQN's sensitivity to ρ, we also performed an experiment on these two environments with ρ set to 0.95. Results, reported in Table 7, show

that agents trained with $\rho = 0.95$ perform slightly worse than the ones trained with $\rho = 0.98$ thus indicating that this parameter has an impact on performance and is worth fine-tuning. However, it is important to note that on the most difficult (*video hard*) environments, the scores obtained by SGQN remain notably above those of other methods, whichever the value of ρ. Visual search is a (loose) measure of the amount of information actually present in the image and necessary for predicting the value. It is likely that one could exhibit worst case environments for which all pixels are necessary to predict the value. In such environments, SGQN might perform poorly. However, we argue that such environments are not representative of most realistic visual RL tasks, either real-world or simulated, where pixel information is very redundant (which is a cause for overfitting). It is likely that in other (maybe more difficult) environments (e.g. with more complex important objects moving across the screen), the necessary threshold for ρ could be lowered.

Benchmark	Environment	$\rho = 0.98$	$\rho = 0.95$
Easy	Ball in cup	950 ± 24	905 ± 39
	Cartpole*	761 ± 28	705 ± 38
Hard	Ball in cup	782 ± 57	789 ± 96
	Cartpole*	569 ± 56	457 ± 60

Table 7: Impact of ρ for Ball in cup and Cartpole*

(a) Observation (b) $\rho = 0.9$ (c) $\rho = 0.95$ (d) $\rho = 0.98$ (e) $\rho = 0.995$

Figure 15: Example of $M(Q_\theta, s, a)$ for different values of ρ on Cartpole (top) and Ball in cup (bottom).

F Comparison with SVEA + SODA

The losses of SGQN can in some way be related to those present in SODA [Hansen and Wang, 2021] and SVEA [Hansen et al., 2021]. The auxiliary self-supervised learning objective of SGQN is similar to replacing the projector used by SODA (and originally introduced in BYOL [Grill et al., 2020]) by $\partial Q(s, a)/\partial s$. Besides, during the critic update, SVEA performs a new estimate of Q from an augmented state $\tau(s)$. The consistency loss used in SGQN can be expressed in a similar fashion by considering that the augmentation τ consists in the application of a mask resulting in a reduction of the information contained in s. From this perspective, SGQN can be seen as the combination of particular cases of SVEA and SODA. Thus we compared the performance of SGQN with the combination of SVEA and SODA. Table 8 reports the corresponding results. Except on the Walker walk *video hard* testing levels, the combination of SVEA and SODA diminishes the scores obtained by using each method independently. Agents trained with both SVEA and SODA also obtain worse performance than the agents trained with SGQN, thus showing the advantages brought by each loss introduced in SGQN.

Benchmark	Environment	SODA	SVEA	SVEA + SODA	SGQN
Easy	Walker walk	771 ± 66	828 ± 66	792 ± 101	$\mathbf{910 \pm 24}$
	Walker stand	965 ± 7	$\mathbf{966 \pm 5}$	948 ± 18	955 ± 9
	Ball in cup	939 ± 10	908 ± 55	825 ± 114	$\mathbf{950 \pm 24}$
	Cartpole	742 ± 73	753 ± 45	677 ± 100	$\mathbf{761 \pm 28}$
	Finger spin	783 ± 51	723 ± 98	767 ± 37	$\mathbf{956 \pm 26}$
	Average	836	836	801	**906**
Hard	Walker walk	312 ± 32	385 ± 63	424 ± 178	$\mathbf{739 \pm 21}$
	Walker stand	736 ± 132	747 ± 43	773 ± 35	$\mathbf{851 \pm 24}$
	Ball in cup	381 ± 163	498 ± 174	211 ± 107	$\mathbf{782 \pm 57}$
	Cartpole	339 ± 87	403 ± 17	351 ± 82	$\mathbf{569 \pm 56}$
	Finger spin	309 ± 49	307 ± 24	244 ± 27	$\mathbf{822 \pm 24}$
	Average	430	468	401	**747**

Table 8: Comparison with SVEA + SODA

G Discussion on the interplay between initial saliency maps and saliency guided training

Since SGQN uses thresholded saliency maps to mask out input images from the very first steps of the algorithm, and since there is no reason for these initial maps to point towards important pixels for the true value function, it is legitimate to wonder how the initial saliency maps actually affect the optimization path of SGQN. We propose the following discussion which aims at explaining why the mechanism of SGQN is robust to the Q_θ network initialization and to which pixels are being selected by the initial saliency maps. Initial saliency maps are likely to appear random since they represent the gradients of random functions close to zero (as per the classical initialization of neural networks). Consequently, when thresholded, these saliency maps are likely to yield pixels that are uniformly spread across the image. In turn, the masking operation initially acts as a random subsampling operation. Since many close pixels in the input image hold redundant information, the application of M_ρ to input images is likely to preserve enough information to correctly predict the value function (minimize L_Q). It is important to note that many other subsampling masks are equally likely to preserve the information necessary to correctly predict the value function. So the initialization of the Q_θ network does not prevent learning the true Q-function based on $s \odot M_\rho$. A direct consequence is that many Q_θ functions, that differ only by which subsampled pixels they rely on, can actually fit the true Q-function, but few will generalize to unseen states, which is the very issue of observational overfitting. Even if the original maps indicating which pixels are informative are wrong, i.e. if too many confounding pixels are retained in M_ρ, then, L_Q, the first part of the critic's loss, still encourages that Q_θ be a solution to the Bellman equation. In that sense, L_C (the second part of the critic's loss) acts as a regularizer: it allows discriminating between functions that would otherwise be equivalent approximate solutions to the Bellman equation. The self-supervised learning procedure, in turn, yields features f_θ that are useful at predicting one's own saliency map. As noted by Grill et al. [2020] or Hansen and Wang [2021], such features are good representations of the input state. Finally, as the Q-function becomes more accurate and as the f_θ features become better at predicting M_ρ, the saliency maps become sharper. Overall, the interplay between the two phases of SGQN is mostly a virtuous circle, that starts from the observation that the initial M_ρ preserve enough information in s to permit correct learning of Q_θ.

H Additional attribution evaluations

We visually compare on Figures 16 and 17 the saliency maps obtained by the other attribution methods Guided-GradCAM [Selvaraju et al., 2017a], Occlusion [Zeiler and Fergus, 2014]. Note that, to avoid going against Goodhart's Law (*"When a measure becomes a target, it ceases to be a good measure."*), [Springenberg et al., 2015] these methods are only used for evaluation, Guided Backpropagation remaining the method used for the computation of $M(Q, s, a)$. While the other agents still seem to retain some dependency on background pixels, the attributions of the agent trained with SGQN remain located chiefly on the important information.

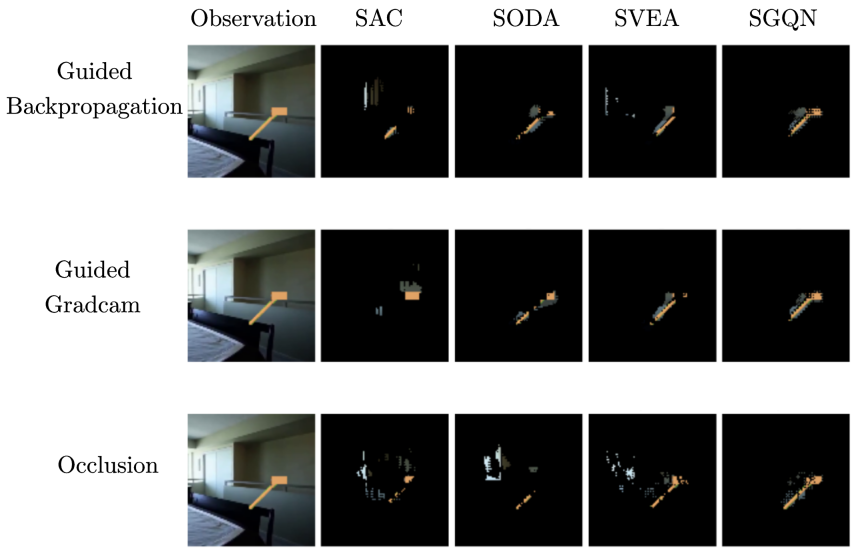


Figure 16: Comparison of different attribution methods on Cartpole video hard levels

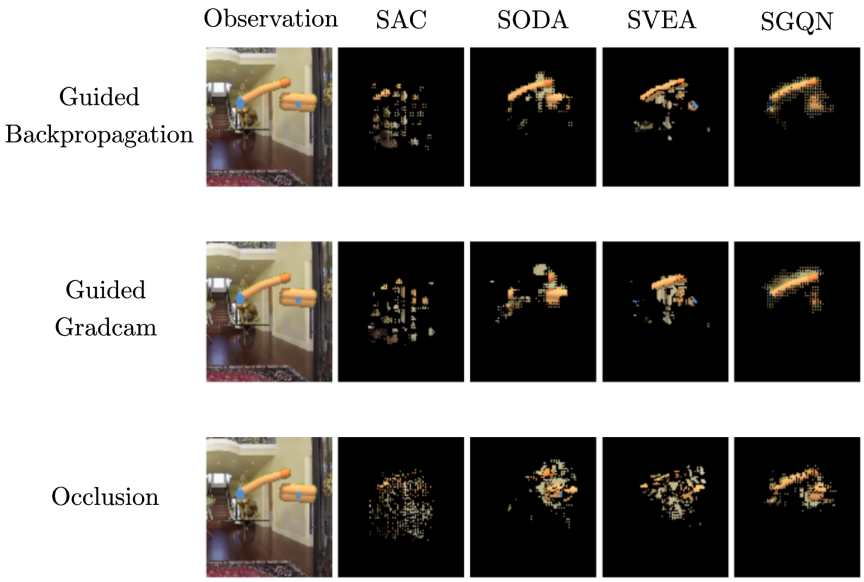


Figure 17: Comparison of different attribution methods on Finger spin video hard levels