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.

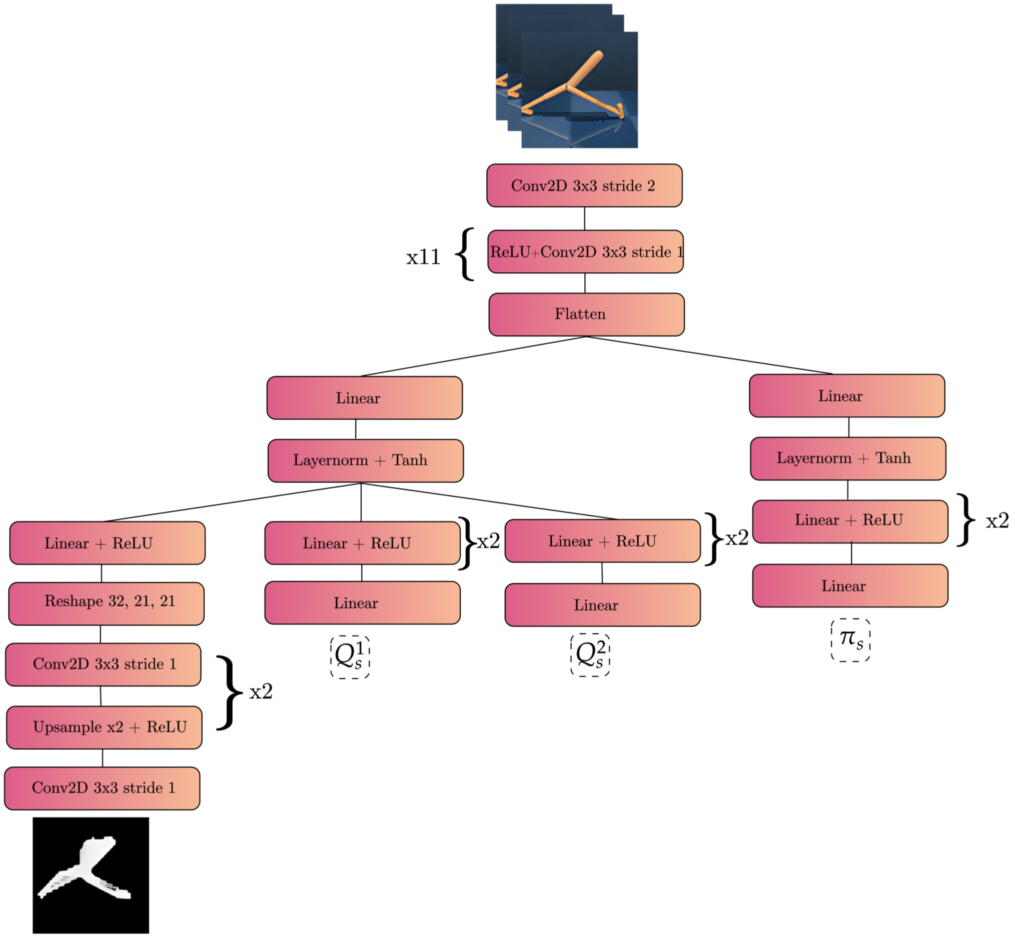


Figure 8: SGQN neural network architecture

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 (Walker walk, Walker stand, Ball in cup), 8 (Cartpole), 2 (Finger spin)
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 at `https://github.com/SuReLI/SGQN`.

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

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. All experiments in Section 4 have been conducted following the hyper-parameters suggested by Hansen et al. [2021], in particular 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 repeated the experiments of Section 4 with a constant action repetition parameter of 4 time steps for all environments. Table 5 reports the results obtained. Using a constant action repetition parameter hinders the performance of almost all agents. Yet SGQN still outperforms 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.

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 5: Performance on *video easy* and *video hard* testing levels with a constant action repetition parameter of 4 time steps for all environments. Δ = difference with second best.

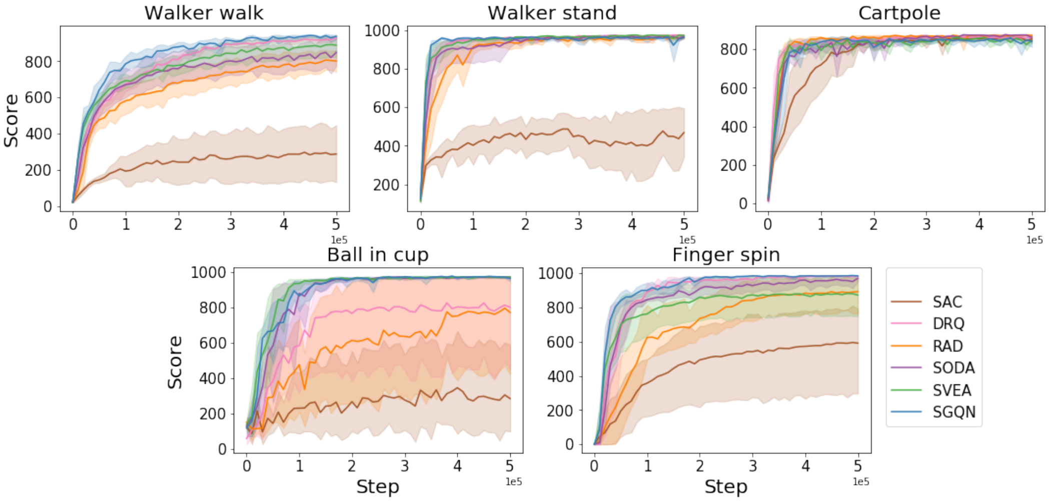


Figure 9: Performance on training levels.

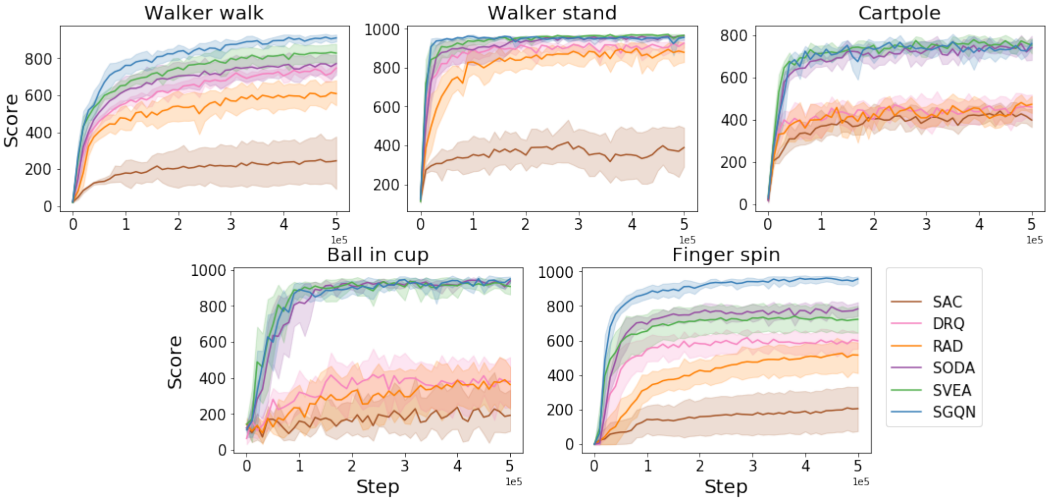


Figure 10: Performance on *video easy* testing levels.

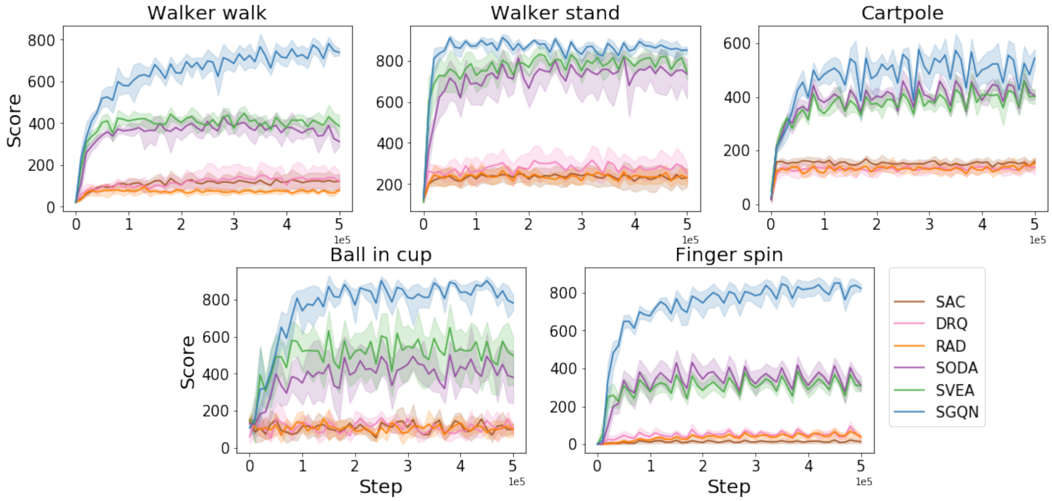


Figure 11: Performance on *video hard* testing levels.

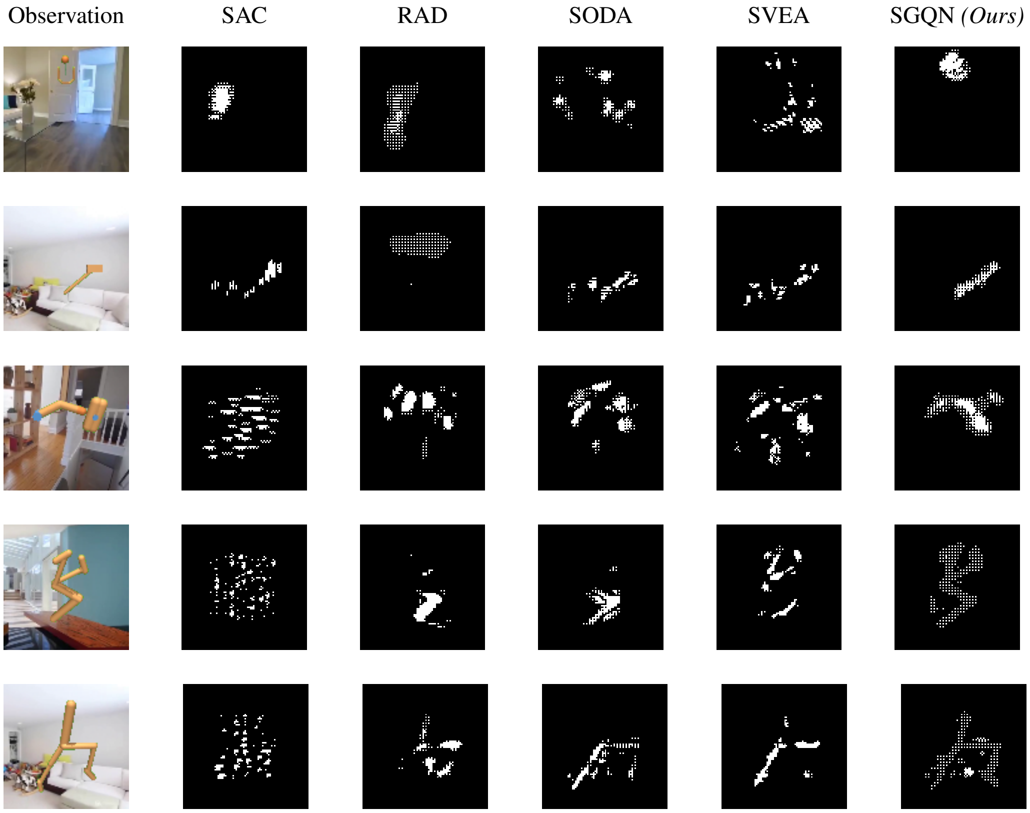
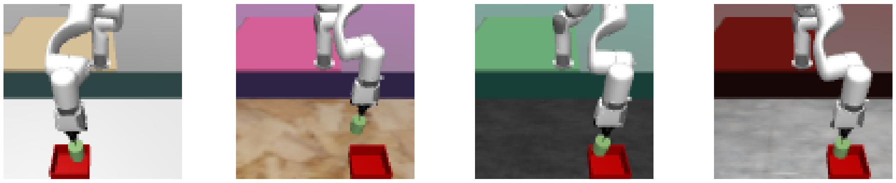


Figure 12: Binarized attribution maps in *video hard*

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

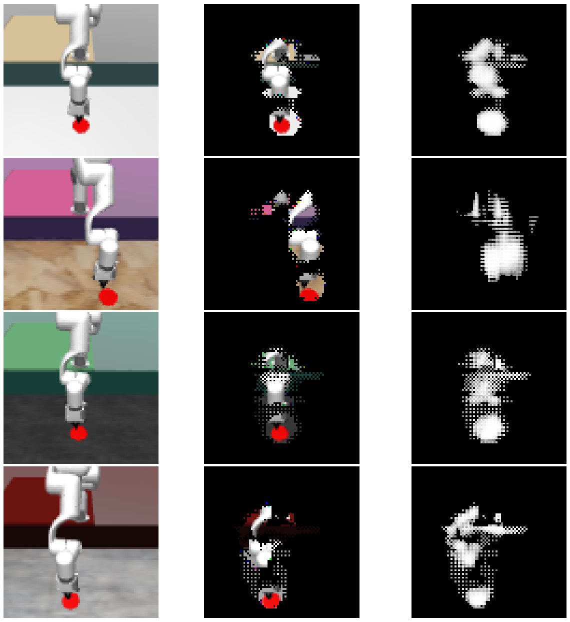

(a) Observation (b) $s \odot M_\rho(Q_\theta, s, a)$ (c) $M_\theta(s, a)$

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*

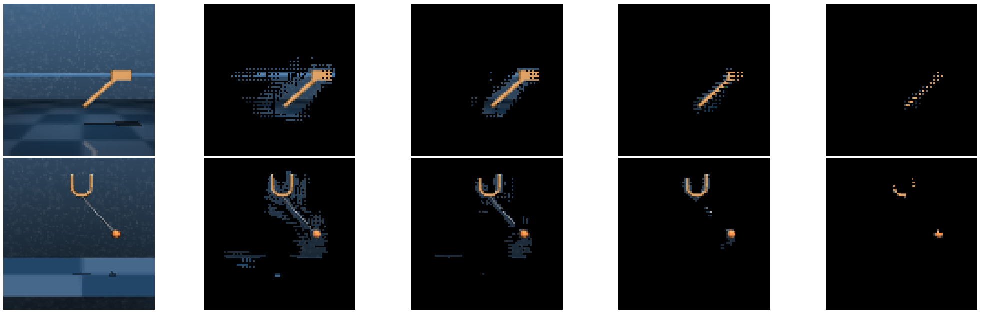

(a) Observation (b) $\rho = 0.9$ (c) $\rho = 0.95$ (d) $\rho = 0.98$ (e) $\rho = 0.995$

Figure 15: Example of $M_\rho(Q_\theta, s, a)$ 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., 2017], 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.

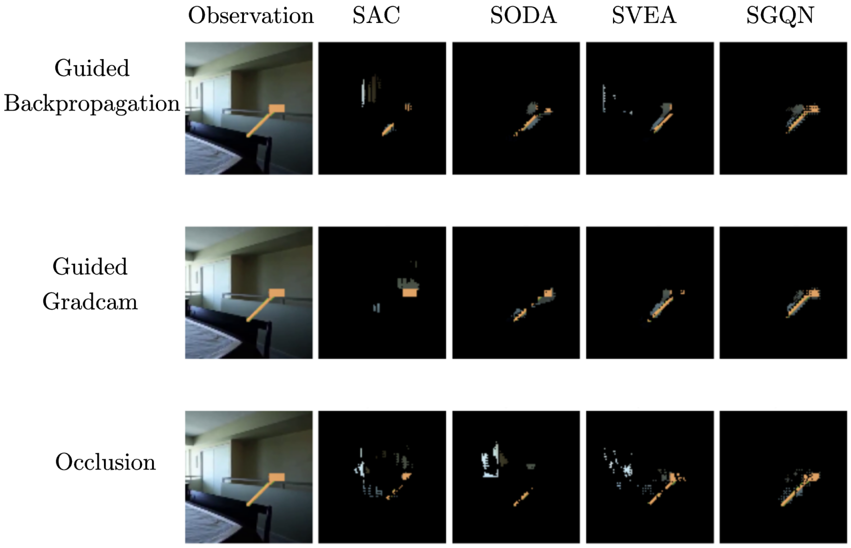


Figure 16: Comparison of different attribution methods on Cartpole video hard levels

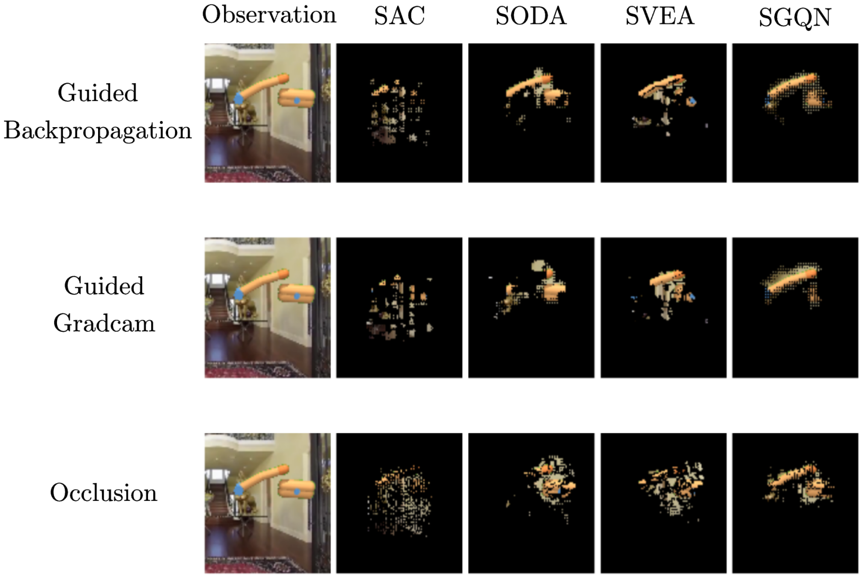


Figure 17: Comparison of different attribution methods on Finger spin video hard levels