Neural Distillation as a State Representation Bottleneck in Reinforcement Learning

Valentin Guillet
ISAE-SUPAERO, Université de Toulouse, France
`valentin.guillet@isae-supaero.fr`

Dennis Wilson
ISAE-SUPAERO, Université de Toulouse, France
`dennis.wilson@isae-supaero.fr`

Carlos Aguilar-Melchor
Sandbox AQ, France
`carlos@sandboxaq.com`

Emmanuel Rachelson
ISAE-SUPAERO, Université de Toulouse, France
`emmanuel.rachelson@isae-supaero.fr`

Abstract

Learning a good state representation is a critical skill when dealing with multiple tasks in Reinforcement Learning as it allows for transfer and better generalization between tasks. However, defining what constitute a useful representation is far from simple and there is so far no standard method to find such an encoding. In this paper, we argue that distillation — a process that aims at imitating a set of given policies with a single neural network — can be used to learn a state representation displaying favorable characteristics. In this regard, we define three criteria that measure desirable features of a state encoding: the ability to select important variables in the input space, the ability to efficiently separate states according to their corresponding optimal action, and the robustness of the state encoding on new tasks. We first evaluate these criteria and verify the contribution of distillation on state representation on a toy environment based on the standard inverted pendulum problem, before extending our analysis on more complex visual tasks from the Atari and Procgen benchmarks.

1 Introduction

Despite the impressive successes of modern reinforcement learning (RL) (Sutton & Barto, 2018) methods in designing efficient specialized control policies for a wide variety of difficult tasks, many studies have highlighted the limited ability of RL agents to generalize to variations of such tasks that would appear easy to a human being (Farebrother et al., 2018; Packer et al., 2018; Zhang et al., 2018; Song et al., 2020; Cobbe et al., 2019). This work is motivated by the idea that networks trained for specific tasks build state representations that can easily be fooled by the ambiguity between observation variables. For instance, in some platform video games, it is possible to design an optimal policy for a given level based solely on background features and progression indicators, rather than on the position of platforms and enemies (Song et al., 2020). While very efficient on this specific level, such a policy might not perform well on another. Conversely, we formulate and evaluate the hypothesis that a network trained to imitate several such specialized policies on a limited set of task variations induces a state representation that lifts the ambiguity and filters out confounding observation variables. Specifically, we investigate whether the process of network distillation (Hinton et al., 2015; Rusu et al., 2016a), inspired by knowledge consolidation in cognitive systems (Wilson & McNaughton, 1994; Ashworth et al., 2014; McClelland et al., 1995), induces valuable state representations. The interplay between distillation and state representation appears to have received little attention so far. We endeavor to fill this gap and investigate how neural distillation can act as a state representation bottleneck in RL.

Our contributions are as follows:

- We propose a generic experimental protocol to evaluate the effects of imitation (via distillation) on state representation.

- We demonstrate, on controlled experiments, that distillation across tasks acts as a filter over state representations that mitigates observational overfitting (Song et al., 2020) and its consequences.

- We extend this analysis to visual RL tasks from the Arcade Learning Environment (Bellemare et al., 2013) and the Procgen (Cobbe et al., 2020) benchmark suites, and evaluate to what extent state representation disambiguation is a key to generalization across tasks.

Section 2 proposes a discussion and an overview of the links between the works in generalization and state representation in multi-task RL. It motivates the study of the emergence of relevant state encodings from the distillation of previous knowledge. Section 3 discusses properties we expect from a good state representation learning procedure, defines quantitative criteria, and presents the controlled experiment designed to evaluate them. Section 4 transposes this analysis to the more complex case of visual control tasks. Section 5 summarizes our main findings and conclusions.

2   GENERALIZATION AND STATE REPRESENTATION IN RL

Generalization is the ability for a supervised learning predictor to output a correct label for any sample that has not been seen during training. This implicitly supposes that all samples (x, y) stem from a unique distribution, whose density is often noted $p(x, y)$. Then, given a loss $\mathcal{L}(y, y')$ and a sample set of N samples (x_i, y_i), closing the generalization gap consists of minimizing the difference between the risk and the empirical risk (Vapnik, 1992) $gap(f) = \mathbb{E}_{x,y \sim p}[\mathcal{L}(f(x), y)] - 1/N \sum_i \mathcal{L}(f(x_i), y_i)$.

Closing the generalization gap in RL is recognized as a difficult problem (Cobbe et al., 2019; Song et al., 2020) that has been surveyed recently by Kirk et al. (2021). Recent contributions on this topic exploit data augmentation (Raileanu et al., 2020; Laskin et al., 2020; Yarats et al., 2020; Lee et al., 2020), regularization (Cobbe et al., 2019; Igl et al., 2019; Cobbe et al., 2020; Wang et al., 2020; Bertoin & Rachelson, 2022), or architecture separation and auxiliary losses (Raileanu & Fergus, 2021). However, few of these works that aim at generalization properties of the policy explicitly draw a connection with the multi-task learning literature. Such a connection can be found in the representation learning literature, either under the form of auxiliary losses (Bellemare et al., 2019), self-learning (Grill et al., 2020), or open-ended learning (Team et al., 2021) for instance.

The general case of multi-task (supervised) learning (Caruana, 1997; Wilson et al., 2007; Ruder, 2017; Crawshaw, 2020) covers the problem of learning functions from *several* distributions over (x, y), each corresponding to a different task. In that case, there is not necessarily a single optimal label that can be attached to a given x and (empirical) risk minimization does not capture the full notion of generalization across tasks. A key problem in this context becomes that of *representation learning* (Bengio et al., 2013), that is learning low-dimensional descriptive features $\varphi(x)$ that are useful to all tasks $p(x, y)$ (or at least a subset of tasks). Formally, for all tasks $p(x, y)$, one searches for a low-dimensional φ such that $\exists g$ s.t. $\mathbb{E}_{x,y \sim p}[\mathcal{L}(g(\varphi(x)), y)] \leq \epsilon$, for a given risk ϵ.

This definition translates straightforwardly to RL: a desirable property for generalization of RL policies is the emergence of common useful features in policies trained on a subset of tasks. Formally, for a given set of MDPs, and all corresponding optimal policies $\pi(a|s)$ inducing a state distribution ρ^π, one searches for a low-dimensional φ such that $\exists g$ s.t. $\mathbb{E}_{s \sim \rho^\pi, a \sim \pi(\cdot|s)}[\mathcal{L}(g(\varphi(s)), a)] \leq \epsilon$, for a given risk ϵ. The definition of multi-task RL proposed by Yu et al. (2020) covers a somewhat restricted case of multi-task learning, where there exists at least one function that performs well on all tasks. This is essentially due to a task identifier being included in the agent's observation, separating the definition domains of different tasks' policies into disjoint sets, thus allowing to merge $g \circ \varphi$ into a single policy for all tasks. In that framework, the question of representation learning translates to that of *state representation learning* (Lesort et al., 2018; Raffin et al., 2019; Caselles-Dupré et al., 2021; Montoya et al., 2021; Schwarzer et al., 2021). In particular, the recent work of Grill et al. (2020) illustrates how such state representations can be achieved in an unsupervised, task-independent way that exploits data augmentation and self-learning. The line of work we develop here is somewhat orthogonal to their contribution since we focus on *inter-task* imitation, while they exploit *intra-task* data augmentation.

Methods tackling multi-task RL problems aim, for instance, at playing all Atari games, or all levels within a Procgen game. Since the corresponding visual states permit discriminating what game is being played, they already implicitly contain the task identifier information and indeed fall within the above definition. Different approaches have been developed to tackle multi-task RL (Rusu et al., 2016b; Fernando et al., 2017; Hessel et al., 2019; Sodhani et al., 2021). We focus here on those that perform imitation learning (Parisotto et al., 2016; Rusu et al., 2016a; Teh et al., 2017) by distilling (Hinton et al., 2015) the policies represented by several specialized neural networks into a common network performing all tasks. The contribution of these methods to the question of learning good state representations in RL has not been discussed in the literature. This work endeavors to fill this gap.

3   DISTILLATION AS AN INFORMATION BOTTLENECK

We wish to draw generic conclusions concerning the effect of distillation on state representation. For this purpose, this section proposes a generic, toy example, where useful state information is entangled with noise and redundant variables, to assess whether distillation is able to recover the useful variables.

3.1 EVALUATION CRITERIA FOR DISTILLATION

A good representation learning procedure should output a low-dimensional set of features that are useful for generalization to unseen states in the task at hand, and to the full span of tasks presented to the agent. In contrast, a bad state encoding within a good policy for a given task transforms the state into features that will not be relevant in other tasks, or that do not permit generalization to new states within the same task. For instance, in the CoinRun Procgen game, it is possible to learn a good policy for a given level, solely based on background elements and not the actual position of enemies, as illustrated by the saliency maps of Song et al. (2020). The resulting policy is good, but the learned state encoding does not permit generalization. Zahavy et al. (2016) illustrate that one can extract options from the state representation learned by DQN (Mnih et al., 2015) on Atari games. However, in these representations, two states with different actions could easily have very close state encodings, thus risking poor generalization to unseen states.

A desirable property for a learning procedure is that it outputs a policy that is both efficient and that relies on a state encoding that permits generalization. We translate this property into the following evaluation criteria:

- **Ability to select important variables.** The final policy should depend strongly on important variables and filter out confounding, uninformative, redundant, or noisy inputs.
- **State separability.** The encoding should cluster together states sharing the same optimal action to ease generalization across states.
- **Robustness.** Application of the final policy to new tasks should preserve a certain amount of performance.

It is possible to interpret these criteria from a bias/variance tradeoff perspective. One advantage of using (deep) neural networks is the small bias they introduce, compared to more constrained families of functions such as linear combination of features. This, in turn, makes them more susceptible to focusing on confounding variables, which creates variance in the hypothesis space. Although previous work showed how stochastic gradient descent acts as a regularizer (Gunasekar et al., 2017), empirical risk minimization remains intrinsically prone to variance. In particular, Song et al. (2020) illustrate how a high dimensional observation space with a low dimensional underlying state space can induce multiple solution policies, some of which are not generalizable to other MDPs. Distillation then acts as a form of enhanced regularization, filtering out value functions (and the corresponding policies) that have equivalent empirical risk but bad generalization properties. Conversely, such a mechanism would be less beneficial in hypothesis spaces that induce more bias and less variance (such as linear representations).

3.2 SYNTHETIC EXPERIMENTAL PROTOCOL

We design a synthetic, toy benchmark, to evaluate these criteria, based on the inverted pendulum environment (Pendulum-v0) from the OpenAI Gym suite (Brockman et al., 2016). The purpose of this experiment is to control the introduction of noise and redundancy in state variables, in order to assess whether distillation is able to recover an efficient encoding of the state, in terms of the above criteria. The original benchmark consists of bringing and stabilizing a pendulum to the upright position by applying discrete torque actions. It features a 3-dimensional observation $(\cos\theta, \sin\theta, \dot{\theta})$ at each time step. We introduce different "levels" of this same game by enriching the observation and making it 30-dimensional in a fashion inspired by the Madelon dataset (Guyon, 2003). Specifically, for a given level, the variables are organized as follows:

- Index 1 to 3. The original state variables.
- Index 4 to 13. For each level, we draw uniformly between -1 and 1 a fixed set of 10 linear combination weights that allow to recombine the three first variables together. This yields 10 new variables. This combination changes from one level to the next. This creates information redundancy within the level that does not permit generalization in other levels.
- Index 14 to 23. We draw uniformly between -1 and 1 a fixed set of 10 linear combination weights over the three first variables, yielding 10 more variables, to which we add a centered Gaussian noise ($\sigma = 0.05$). These weights are shared across levels, so this creates information redundancy that permits generalization but includes measurement noise.
- Index 24 to 26. A copy of 3 random variables among the 24 previous ones, re-drawn for each level.
- Index 28 to 30. Four centered Gaussian noise ($\sigma = 1$) variables.

We draw N_{levels} different levels of the benchmark and individually train (on each level) a C51 (Bellemare et al., 2017) agent as implemented in Dopamine (Castro et al., 2018), with default hyperparameters. The network is adapted to take the 30-dimensional state as input, and features two hidden layers of size 32, with ReLU activation functions. We refer
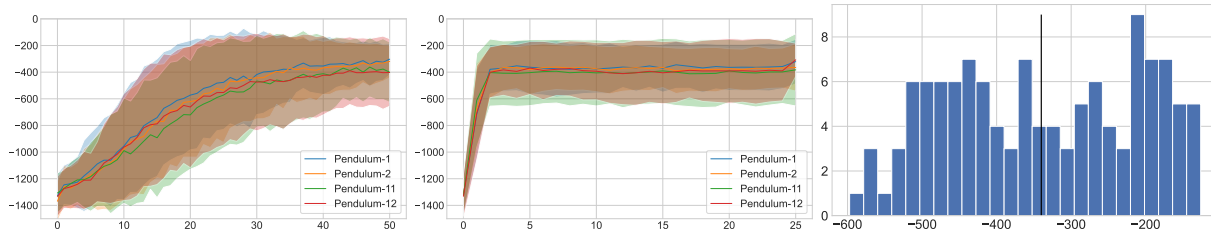
Figure 1: Training curves for experts (left) and student (middle), shaded area is one standard deviation. Histogram of an expert's final score (right). 4 of the 12 levels are shown due to overlap in the remaining levels.

to such networks as *experts*. As per the protocol of Rusu et al. (2016a) we then learn from scratch a distillation agent, referred to as *student*, that imitates the distribution of outputs of the experts. Specifically, we train the student to imitate the distribution of Q-values output by the C51 agent for each state. The student plays a full episode on a given level and fills a level-specific replay buffer with examples of actions recommended by the corresponding expert, before switching to another level. A slight difference with the protocol of Rusu et al. (2016a) is that, as recommended by Parisotto et al. (2016), we let the student (instead of the expert) draw which actions are performed on the level. The mini-batches drawn to train the student at a given step are drawn from the replay buffer of the current episode. Appendix D provides a pseudo-code for this procedure.

3.3 RESULTS AND DISCUSSION

Figure 1 reports the learning curves of all experts and the student. A single curve in this figure corresponds to a single level and is the average, across 150 training runs, of the expert's score with respect to time (1 unit on the x-axis corresponds to 5 episodes of 200 time steps each). Therefore, $150 \times N_{levels}$ agents were trained overall and each N_{levels}-tuple of experts was distilled into a student, yielding 150 student networks. Unless stated otherwise, $N_{levels} = 12$ in this section's discussion. The evaluation is performed on a fixed grid of 50 randomly drawn initial states. As all levels share the same underlying dynamics and reward models, the training curves of all experts are very similar and converge to the same average score. The non-negligible training variance illustrates that the training process is rather sensitive and might fail. To put this result in perspective, Figure 1 also reports a representative histogram of the 150 experts' score after the last iteration, on a single level (the black bar indicates the average).

To evaluate the ability of the student to filter out variables that do not lead to generalization, we evaluate each feature's importance for the experts and the student. For this, we compute the gradient of the network's output with respect to the state variables. Figure 2 reports graphically the gradient amplitudes of the N_{levels} experts and of the student, averaged over a pre-defined grid $\{s_j\}$ of states, for a subset of levels. Each cell represents $\sum_j |(\partial net/\partial s^i)(s_j)|$, where net is either the expert (top) or the student (bottom), and where s^i is the ith state variable. The lines corresponding to the experts in Figure 2 (each top line) show that the expert relies both on the "true" state variables and on the confounding variables. Specifically, as expected, the expert is fooled by the redundant variables (variables 4 to 13). To a certain extent, it is able to filter out the redundant but noisy variables (variables 14 to 23), while sometimes still relying on them to define the policy. Also as expected, the repeated variables (variables 24 to 26) have a rather strong influence on the policy. Finally, the experts are able to filter out the purely random variables (variables 28 to 30). The bottom lines show that the student has filtered out all confounding variables and only relies on the three original state variables. Interestingly, this property is true even for variables 14 to 23 even though these variables are shared across levels. We conclude that consolidation on as few as 12 levels is enough to disambiguate between variables and filter out noise.
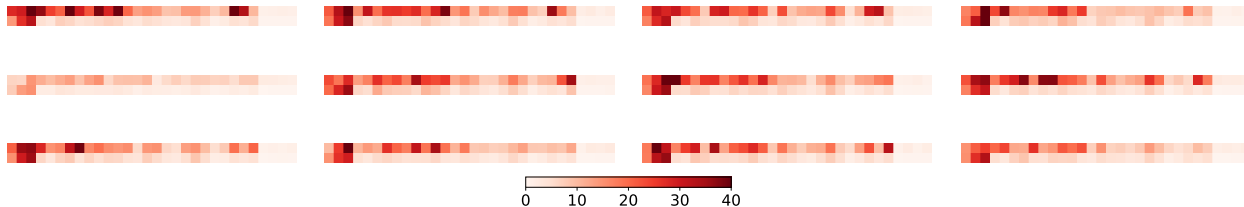

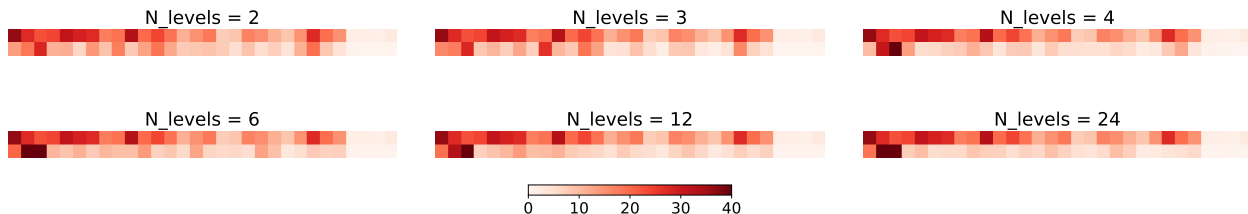Figure 2: Feature importance in the experts (top lines) and the student (bottom lines) for all levels.

Figure 3: Feature importance as the number of experts increases of experts (top lines) and students (bottom lines).
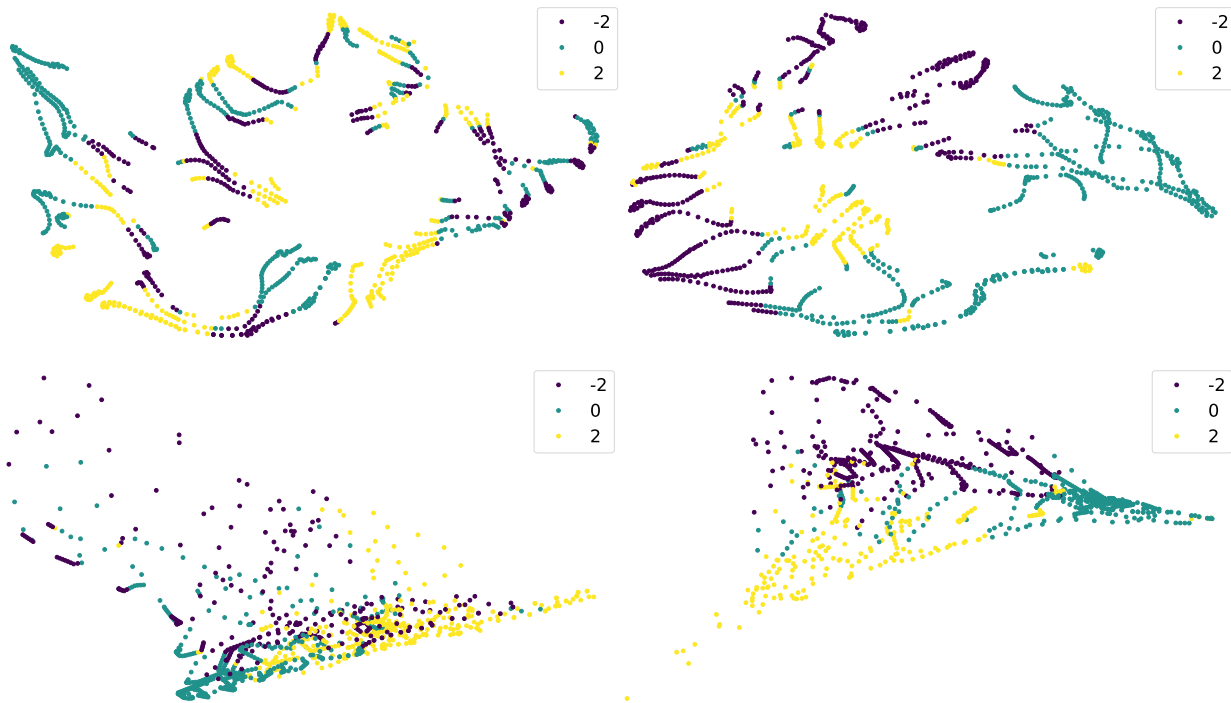

Figure 4: t-SNE (top) and PLS (bottom) plots of the state embedding for an expert (left) and the student (right).

In order to assess the impact of N_{levels}, Figure 3 reports the gradient amplitudes on a single level, when the student is trained to imitate a growing number of experts. The filtering effect of distilling the experts into the student seems to appear with as little as 4 experts (with minor dependence on some confounding variables) and persists consistently for larger numbers of training levels.

In order to evaluate the second criterion of Section 3.1, state separability, we plot a t-SNE visualization (Van der Maaten & Hinton, 2008) of the state embedding, i.e. of the features after the second hidden layer. Since t-SNE captures correlations between descriptive variables but not correlations with the network's output, we also compute a partial least squares regression (PLS) (Wold et al., 1984) of the action taken by the agent with respect to the embedding and plot the projection of the state embeddings within the two first principal directions. Figure 4 reports these visualizations of the state representation embedding, where we color points according to their optimal action (-2, 0, or 2 with our discretized inverted pendulum). Graphically, it appears that distillation favors better proximity of states that share the same action. In order to lift any graphical interpretation bias and quantify this measure of separability, we train a linear SVM to separate the data points in the space spanned by the PLS projection and report the accuracy of the obtained classifier in Table 1 (more detailed results in Appendix B). Specifically, we train this linear SVM both when using the projection on the 3 first principal directions computed by the PLS and on all 32 directions (which is equivalent to not computing the principal directions at all). The results consistently demonstrate that (except for $N_{levels} = 2$), the embedding of the student is better suited for linear separability than that of the expert (whether we use 3 or 32 dimensions). Increasing the variety of training levels monotonically increases the linear separability of states sharing the same action.

| | $N_{levels} = 2$ | | $N_{levels} = 6$ | | $N_{levels} = 12$ | | $N_{levels} = 24$ | |
	dim = 3	dim = 32	dim = 3	dim = 32	dim = 3	dim = 32	dim = 3	dim = 32
expert	78.05 (4.90)	91.29 (2.06)	78.05 (4.90)	91.29 (2.06)	78.05 (4.90)	91.29 (2.06)	78.05 (4.90)	91.29 (2.06)
student	75.56 (6.27)	91.60 (2.16)	83.06 (5.66)	93.68 (1.87)	88.03 (1.56)	95.37 (0.75)	91.01 (4.60)	95.81 (1.78)

Table 1: Average (and standard deviation of) accuracy for a linear classifier built on the state representation.
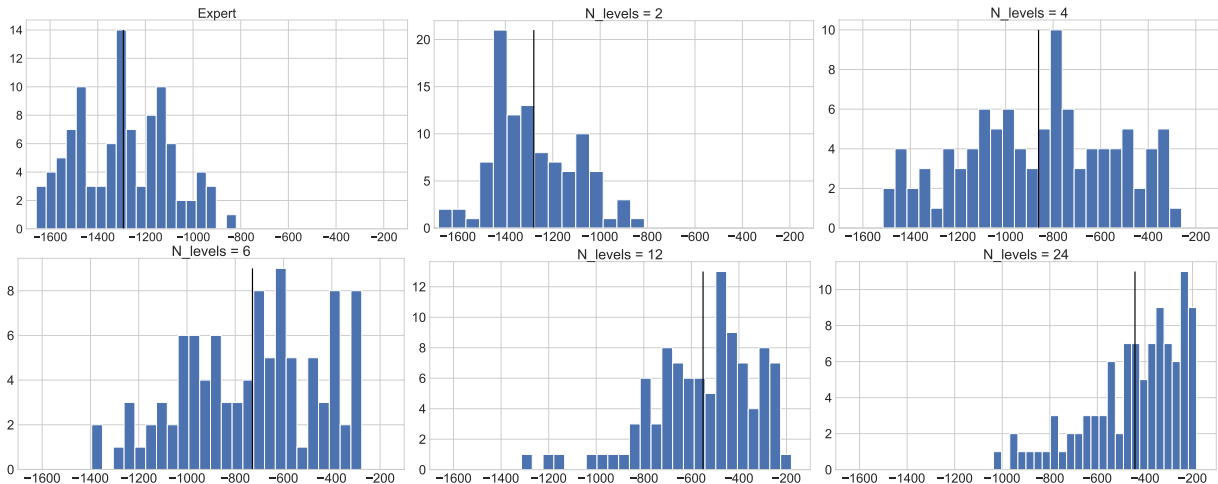
Figure 5: Robustness of an expert and of the student policy for varying values of N_{levels}.

We expect the student policy to perform well on a variety of levels, both those upon which the experts were trained and new ones. Figure 5 reports the distribution of scores obtained by an expert and by the student on a set of testing levels, drawn independently from the ones the experts were trained on. The black bar indicates the distribution's mean. As N_{levels} increases, the student becomes more and more robust to new levels and, eventually, its average score reaches that of individual experts. Note that this is different from identifying relevant variables (which happens already with 4 levels), and from performing as well as each expert individually on the training levels (which was illustrated on Figure 1).

While this toy problem consisted of a simple task, inverted pendulum, the results clearly demonstrate that distillation can create new student networks which focus on important features, ignore confounding features, and reach the performance of expert networks, benefiting from additional experts trained on different levels. One possible perspective of this work (although beyond the scope of this paper) is to design a curriculum of training tasks that minimize the number of corresponding experts which are necessary for good generalization performance.

4 VISUAL STATE ENCODINGS FROM DISTILLATION

The previous section quantified the effect of distillation as an information bottleneck on the state representation, via the proposed controlled experiment. This bottleneck filters out information redundancy and noise, leading to features that permit better generalization and robustness. The next question is to assess whether this type of filtering is actually a missing feature that would improve generalization and robustness in applications. For instance, do this experimental protocol's results remain when evaluated on a set of Atari games or a variety of levels for several Procgen games?

4.1 VISUAL BENCHMARKS

The evaluation protocol we follow remains essentially the same, with slight adaptations to the visual RL tasks at hand. Specifically, the network architecture is now the classic DQN architecture from Mnih et al. (2015), adapted to the C51 output (Bellemare et al., 2017). The training algorithm we use for the experts is the Dopamine (Castro et al., 2018) implementation of Rainbow (Hessel et al., 2018). For the student, we adapt the Actor-Mimic Networks (AMN) algorithm of Parisotto et al. (2016). Compared to plain distillation, AMN adds a feature regression loss that regularizes the features of the student network towards the features of the experts, which was shown to improve the distillation efficiency. The goal for this regularization term is not to reproduce exactly the expert features, but rather to constrain the AMN features to contain the same information as them. An adaptation layer is therefore added after the AMN feature

layer so that the order and dimension of each feature vectors does not matter for the regularization term. Contrarily to the original AMN implementation that applies the feature regression loss on the last hidden layer, we instead apply it to the output of the last convolutional layer, since we would like to encourage structured feature maps that permit generalization, rather than less structured feature spaces after fully-connected layers (pseudo-code in Appendix D).

To evaluate the feature importance criterion of Section 3.1, instead of computing network gradients with respect to the input, we turn to saliency maps, which essentially represent the same information in a more structured and interpretable fashion. Specifically, we implement perturbation-based saliency maps, since they seem to provide the most interpretable results (Zahavy et al., 2016; Greydanus et al., 2018; Puri et al., 2020; Huber et al., 2021). We retain an analysis in terms of t-SNE and PLS embeddings for the separability of states in the representation space. Finally, the robustness criterion is adapted to the different situations as discussed below.

Many Atari games share common visual features. For instance, Pong, Breakout and VideoPinball need to detect the position of a ball. It seems desirable that the policy depends on this position. Conversely, SpaceInvaders and Carnival share similar visual features, but it is unclear whether they can be reused in a common policy. It is important to note that Atari games enjoy completely disjoint observation spaces and unrelated dynamics and reward models. This does not prevent applying distillation on the corresponding agents, without assuming that the policies will be transferable from one game to another. In this case, the intention of performing expert distillation on this type of benchmark, where the underlying dynamics are very different from one game to the other, is really to extract meaningful state representations and avoid concentrating on the "wrong" features, keeping in mind that it is likely that few features might transfer from one game to another. We test these intuitions on the five aforementioned games.

The case of Procgen games is different, as this benchmark was designed precisely to evaluate generalization across visual variations of the same game, and there is thus an existing measure of task similarity. Within the same Procgen game, we aim to both extract features that remain valid across levels, and evaluate how the student agent generalizes to unseen levels. Hence, this benchmark appears much closer to the controlled experiment of Section 3. We focus specifically on CoinRun, BigFish and Jumper. On each game, we train experts on $N_{levels} = 50$ levels, with each expert trained on a single level, then train the student network based on all expert networks.

4.2 RESULTS ON VISUAL TASKS

Figure 6 reports training curves for the experts and the student on a subset of tasks (complete results in Appendix F). Each training run was repeated 10 times for Atari games and 5 times for each level of each Procgen game. One unit on the x-axis corresponds to 50k time steps for Atari games. Procgen games display a large variance across runs so we report the score distribution of experts after 750000 time steps across all $N_{levels} = 50$ training levels and 5 random seeds, for each game. Average scores are represented with a thin vertical line. Note that the training of both the experts and the student are sufficient to reach a non-trivial score but are also representative of reasonably small training times (details in Appendix A) during which the experts do not necessarily reach state-of-the-art performance. Students do reproduce the performance of experts in almost all cases, with the exception of BigFish (discussed in Appendix E) and Breakout (shown in Appendix F). As such, we find the allocated training time sufficient for common features to emerge within the state representation and to study distillation.

We note that, for the Procgen games of CoinRun and Jumper, the student is able to reproduce expert performance over the 50 levels. While each expert is trained on a single level, the student is able to distill the information from each of these experts into a single network capable of performing similarly across the different levels. We compare this to the generalization of randomly selected trained experts over the same 50 levels: although most experts are able to solve the level they are trained on, Figure 6 confirms this is not enough to ensure transfer to any other level. We posit that distillation is therefore beneficial for generalization to different levels in Procgen; further discuss robustness to level changes below.

We next evaluate the dependence of experts and students on various input features using saliency maps. Interpreting saliency maps from a human perspective can be a tricky endeavor; examples of such maps are reported and discussed in Appendix G. Instead, we can evaluate the cardinality of the set of input features that strongly condition the network's output. As mentioned in Section 3.1, it is desirable for this set to be small. Contrarily to the toy experiment of Section 3.2 however, it is not necessarily always the same pixels in the saliency map that are important for the network's output and, on visual tasks, we mostly would like to filter out irrelevant background elements and only focus on level-independent visual cues. In order to quantify this as a metric, we set all pixels in the saliency image to 1 if their intensities are above a certain threshold ϵ, and 0 otherwise, then sum over all pixels and average this value across a predefined set of game images. The intention is to reflect a similar criterion as that introduced in Section 3.3, under the form of a $\sum_i \sum_j \mathbf{1}((\partial net/\partial s^i)(s_j) > \epsilon)$ value, where net is either the expert or the student, s_j is the jth game image
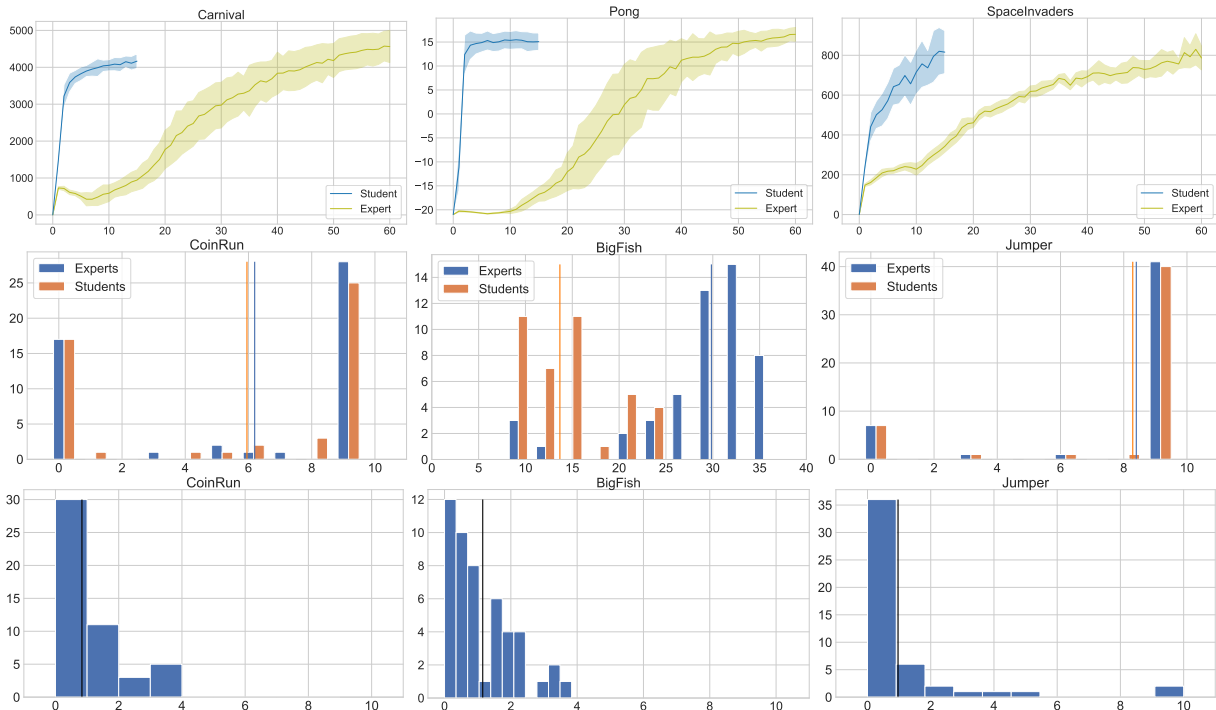
Figure 6: (Top) Training curves for the experts and the student on three Atari tasks. (Middle) Distribution of scores of trained experts on their respective training level and the student network over all 50 levels; mean is represented with a thin line. (Bottom) Distribution of scores of randomly selected trained experts on the 50 training levels (Note that BigFish axis are not the same between middle and bottom rows).

	Breakout	Pong	VideoPinball	Carnival	SpaceInvaders	CoinRun	BigFish	Jumper
expert	239 (75)	148 (81)	331 (246)	616 (278)	2583 (772)	1497 (203)	727 (128)	2628 (276)
student	70 (34)	68 (50)	173 (135)	84 (108)	1488 (784)	1443 (193)	460 (126)	2116 (54)

Table 2: Saliency intensity (and standard deviation) as a proxy for the number of important variables.

in the testing set, and s^i is the variable standing for the perturbation leading to the ith pixel in the saliency map.[1] To account for the distribution of gradients independently of their amplitudes, we linearly scale the saliency maps pixel's intensity between 0 and 1 and take $\epsilon = 1/20$.

Table 2 reports this metric for all games tested in the Atari and Procgen suites. For the Atari games, the "expert" line reports the metric for each game's expert, while for the Procgen games, the reported value for a given game is the average over experts for all training levels. There are far fewer important pixels for the students than for the experts on all environments, with some environments showing a drastic reduction in the number of important variables; e.g. Carnival, which uses fewer than $1/7\times$ pixels in the student network. It is worth noting that the performance is not impacted by the reduction in features, for example in Carnival, or in Pong, where the same performance is achieved by student networks which use fewer than half the pixels used by expert networks. This indicates that the distillation process leads the student's decision function to rely on specific visual features which are necessary for reproducing expert output. Distillation therefore seems to act as a regularizer among features by not creating dependence in the student network on unnecessary features.

Figure 7 presents the t-SNE and PLS representations of a set of images from the Space Invaders game, colored by their associated optimal action (similar representations for other games in Appendix H). We note that features are more clearly grouped in the student networks based on action in both the t-SNE and PLS representations, especially for the RIGHTFIRE action seed in the PLS representation. We posit that distillation concentrates features based on final output, leading to more separable projected states. To deepen this analysis, we repeat the same quantification of separability of projected states as in Section 3.3, by training a linear SVM on projected states. Table 3 illustrates how the state

---

[1]Note that this is close to approximating an ℓ_0 "norm" of the rounded gradient (Donoho & Elad, 2003).
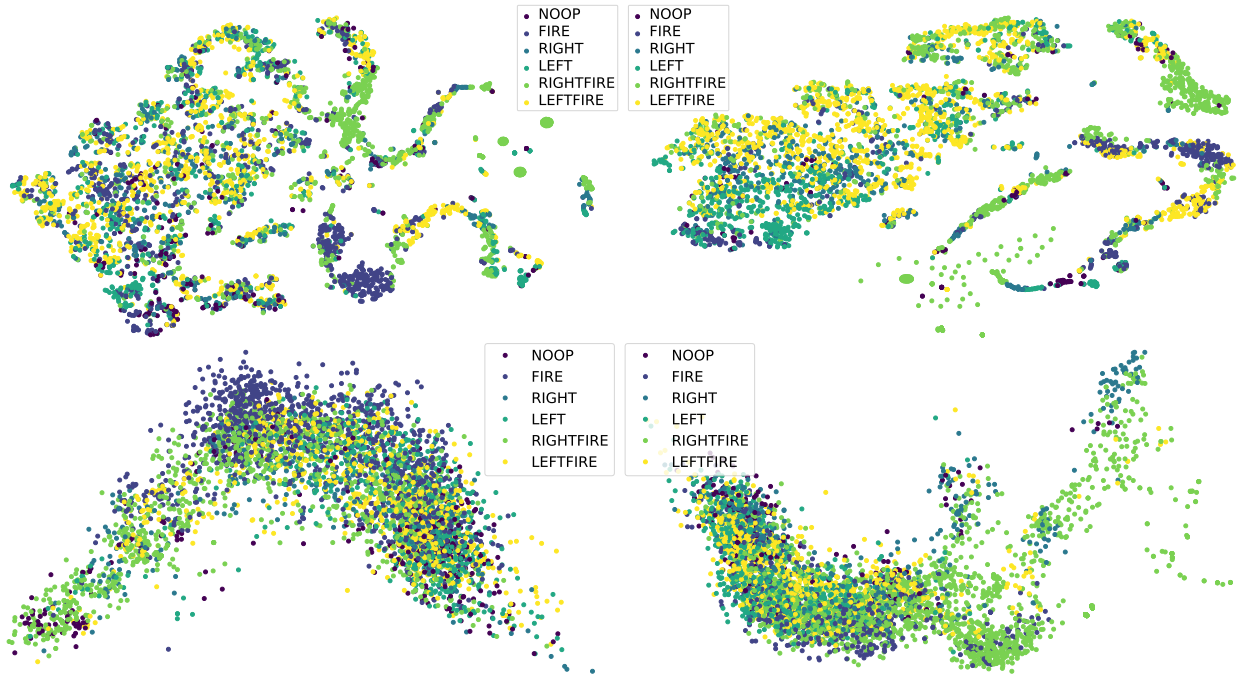
Figure 7: t-SNE (top) and PLS (bottom) state embeddings for an expert (left) and the student (right) on Space Invaders.

	Breakout	Pong	VideoPinball	Carnival	SpaceInvaders	CoinRun	BigFish	Jumper
expert	77.00 (1.25)	75.29 (1.90)	71.23 (3.58)	70.07 (1.13)	76.49 (2.37)	92.53 (0.41)	95.49 (2.99)	76.10 (3.43)
student	79.03 (3.50)	80.99 (1.78)	80.83 (8.43)	72.36 (2.15)	95.47 (1.98)	97.19 (0.30)	97.96 (2.01)	81.15 (2.71)

Table 3: Average (and standard deviation) of accuracy for a linear classifier built on the state representation.

representation learned by the student lends itself better to a linear classification, which indicates better geometrical proximity for states sharing the same action. This indicates that distillation creates networks with a better chance at avoiding overfitting and generalizing to visual changes as their feature representation groups states based on correlated action and not on visual features that have low correlation with the final output.

To understand the importance of a good state representation in adapting to new environments, we evaluate the student's robustness to new levels in the selected Procgen games. We discuss the possibility of using distillation for the Atari benchmark in C, but focus here on Procgen. Figure 8 reports the distribution of scores when the student is evaluated on levels from the testing distribution of Procgen, i.e. levels with different layouts, visual representation, and dynamics such as enemy number and placement.

In the case of the BigFish and Jumper environments, no simple policy exists common to every level that allows for a good performance, thus almost no agent is able to succeed without retraining on new levels. We note, however, that the small number of levels (50) used to train the experts contributes to the lack of generalization; in (Cobbe et al., 2020), individual networks trained on 100, 300, and even 1000 levels had similar performance on test levels for BigFish and Jumper. However, the performance of the student network on CoinRun is interesting as the simple policy of running right while jumping as soon as possible is often enough to finish many levels. In fact, Figure 8 shows that the student is able to solve more than half of the 100 unseen levels we tested it on. On the other hand, the fail cases are in a large majority due to the presence of enemies in the path of the agent when following this simple policy. This confirms that, although simple, the policy extracted by the distillation process is quite general and can transfer as is to multiple new levels.

5 CONCLUSION

In this paper we explore the impact of policy imitation via distillation on state representation and showed that distillation can act as an information bottleneck that mitigates observational overfitting. More specifically, we isolated three criteria to evaluate the ability of a policy to generalize to various environments and proposed experimental means of measuring
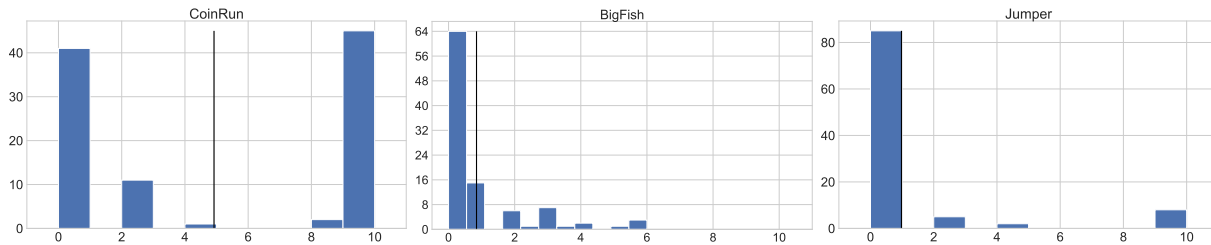
Figure 8: Robustness evaluation of the student's policy on the testing distribution of Procgen games.

each one. Our experiments on a toy environment confirmed that distillation is an efficient way to filter out task-specific features, thus allowing trained policies to focus on variables in the observation space that are relevant for multiple environments. The extension to more difficult visual tasks supports this idea and reinforces the use of multi-task policy imitation as an interesting mechanism for general feature extraction, including when the training of the experts or of the student are interrupted before total convergence. One notable finding from this work is that distillation requires a limited number of pre-trained experts, and no specific knowledge of task boundaries or identifiers, to achieve efficient generalization. The ability to learn a good state representation from a set of various tasks is a crucial skill for any agents evolving in multiple environments. In the lifelong learning setting in particular, an efficient encoding can allow for better transfer and generalization between tasks. An additional interesting perspective lies in the combination of self-learning of good representations that aim at observational robustness within a given task (Grill et al., 2020; Bertoin & Rachelson, 2022), with inter-task distillation as presented here.

REFERENCES

Anna Ashworth, Catherine M Hill, Annette Karmiloff-Smith, and Dagmara Dimitriou. Sleep enhances memory consolidation in children. *Journal of Sleep Research*, 23(3):304–310, 2014.

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.

Marc Bellemare, Will Dabney, Robert Dadashi, Adrien Ali Taiga, Pablo Samuel Castro, Nicolas Le Roux, Dale Schuurmans, Tor Lattimore, and Clare Lyle. A geometric perspective on optimal representations for reinforcement learning. *Advances in neural information processing systems*, 32, 2019.

Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *ICML*, pp. 449–458, 2017.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 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, 2016.

Rich Caruana. Multitask Learning. *Machine Learning*, 28:41–75, 1997.

Hugo Caselles-Dupré, Michael Garcia-Ortiz, and David Filliat. S-trigger: Continual state representation learning via self-triggered generative replay. In *2021 International Joint Conference on Neural Networks*, pp. 1–7, 2021.

Pablo Samuel Castro, Subhodeep Moitra, Carles Gelada, Saurabh Kumar, and Marc G. Bellemare. Dopamine: A research framework for deep reinforcement learning, 2018.

Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *International Conference on Machine Learning*, pp. 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*, pp. 2048–2056, 2020.

Michael Crawshaw. Multi-task learning with deep neural networks: A survey. *arXiv preprint arXiv:2009.09796*, 2020.

David L Donoho and Michael Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via l1 minimization. *Proceedings of the National Academy of Sciences*, 100(5):2197–2202, 2003.

Jesse Farebrother, Marlos C Machado, and Michael Bowling. Generalization and regularization in dqn. *arXiv preprint arXiv:1810.00123*, 2018.

Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*, 2017.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *International Conference on Machine Learning*, pp. 1126–1135, 2017.

Samuel Greydanus, Anurag Koul, Jonathan Dodge, and Alan Fern. Visualizing and understanding Atari agents. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in Neural Information Processing Systems*, 33, 2020.

Suriya Gunasekar, Blake E Woodworth, Srinadh Bhojanapalli, Behnam Neyshabur, and Nati Srebro. Implicit regularization in matrix factorization. In *Advances in Neural Information Processing Systems*, 2017.

Isabelle Guyon. Design of experiments of the nips 2003 variable selection benchmark. In *NIPS workshop on feature extraction and feature selection*, 2003.

Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Gheshlaghi Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *AAAI*, pp. 3215–3222, 2018.

Matteo Hessel, Hubert Soyer, Lasse Espeholt, Wojciech Czarnecki, Simon Schmitt, and Hado van Hasselt. Multi-task deep reinforcement learning with popart. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 3796–3803, 2019.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.

Tobias Huber, Katharina Weitz, Elisabeth André, and Ofra Amir. Local and global explanations of agent behavior: Integrating strategy summaries with saliency maps. *Artificial Intelligence*, 301:103571, 2021.

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.

Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. A survey of generalisation in deep reinforcement learning. *arXiv preprint arXiv:2111.09794*, 2021.

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, 2020.

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.

Timothée Lesort, Natalia Díaz-Rodríguez, Jean-Franois Goudou, and David Filliat. State representation learning for control: An overview. *Neural Networks*, 108:379–392, 2018.

James L. McClelland, Bruce L. McNaughton, and Randall C. O'Reilly. Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. *Psychological Review*, 102(3):419–457, 1995.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, and Georg Ostrovski. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

Juan M Montoya, Imant Daunhawer, Julia E Vogt, and Marco A Wiering. Decoupling state representation methods from reinforcement learning in car racing. In *Proceedings of the 13th International Conference on Agents and Artificial Intelligence*, pp. 752–759, 2021.

Alex Nichol, Joshua Achiam, and John Schulman. On First-Order Meta-Learning Algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

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.

Emilio Parisotto, Lei Jimmy Ba, and Ruslan Salakhutdinov. Actor-Mimic: Deep Multitask and Transfer Reinforcement Learning. In *ICLR*, 2016.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Nikaash Puri, Sukriti Verma, Piyush Gupta, Dhruv Kayastha, Shripad Deshmukh, Balaji Krishnamurthy, and Sameer Singh. Explain your move: Understanding agent actions using specific and relevant feature attribution. In *International Conference on Learning Representations*, 2020.

Antonin Raffin, Ashley Hill, René Traoré, Timothée Lesort, Natalia Díaz-Rodríguez, and David Filliat. Decoupling feature extraction from policy learning: assessing benefits of state representation learning in goal based robotics. In *Workshop on Structure & Priors in Reinforcement Learning at ICLR*, 2019.

Roberta Raileanu and Rob Fergus. Decoupling value and policy for generalization in reinforcement learning. *arXiv preprint arXiv:2102.10330*, 2021.

Roberta Raileanu, Max Goldstein, Denis Yarats, Ilya Kostrikov, and Rob Fergus. Automatic data augmentation for generalization in deep reinforcement learning. *arXiv preprint arXiv:2006.12862*, 2020.

Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

Andrei A. Rusu, Sergio Gomez Colmenarejo, Çaglar Gülçehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy Distillation. In *ICLR*, 2016a.

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016b.

Max Schwarzer, Nitarshan Rajkumar, Michael Noukhovitch, Ankesh Anand, Laurent Charlin, R Devon Hjelm, Philip Bachman, and Aaron C Courville. Pretraining representations for data-efficient reinforcement learning. *Advances in Neural Information Processing Systems*, 2021.

Shagun Sodhani, Amy Zhang, and Joelle Pineau. Multi-task reinforcement learning with context-based representations. In *International Conference on Machine Learning*, pp. 9767–9779, 2021.

Xingyou Song, Yiding Jiang, Stephen Tu, Yilun Du, and Behnam Neyshabur. Observational overfitting in reinforcement learning. In *International Conference on Learning Representations*, 2020.

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

Open Ended Learning Team, Adam Stooke, Anuj Mahajan, Catarina Barros, Charlie Deck, Jakob Bauer, Jakub Sygnowski, Maja Trebacz, Max Jaderberg, Michael Mathieu, et al. Open-ended learning leads to generally capable agents. *arXiv preprint arXiv:2107.12808*, 2021.

Yee Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 4496–4506, 2017.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9 (11), 2008.

Vladimir Vapnik. Principles of risk minimization for learning theory. In *Advances in neural information processing systems*, pp. 831–838, 1992.

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

Aaron Wilson, Alan Fern, Soumya Ray, and Prasad Tadepalli. Multi-task reinforcement learning: a hierarchical Bayesian approach. In *ICML*, pp. 1015–1022, 2007.

Matthew A. Wilson and Bruce L. McNaughton. Reactivation of hippocampal ensemble memories during sleep. *Science*, 265(5172):676–679, 1994.

Svante Wold, Arnold Ruhe, Herman Wold, and WJ Dunn, III. The collinearity problem in linear regression. the partial least squares (pls) approach to generalized inverses. *SIAM Journal on Scientific and Statistical Computing*, 5(3): 735–743, 1984.

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.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning*, pp. 1094–1100, 2020.

Tom Zahavy, Nir Ben-Zrihem, and Shie Mannor. Graying the black box: Understanding dqns. In *International Conference on Machine Learning*, pp. 1899–1908, 2016.

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

A   REPRODUCIBILITY

All the code used in this paper can be found at `https://anonymous.4open.science/r/DistillationAsBottleneck/`. The code is based on the Dopamine implementation of standard RL algorithms which can be found at `https://github.com/google/dopamine`.

	Toy Pendulum	Procgen	Atari
algorithm	C51	Rainbow	Rainbow
feature size	32	512	512
optimizer		Adam	
learning rate	5e-3	6.25e-5	6.25e-5
memory capacity	50000	250000	1000000
batch size	128	32	32
target update period	100	4000	8000

Table 4: Algorithm hyperparameters.

The hyperparameters used for our algorithm on each environment can be found in table 4.

We also used the scikit-learn (Pedregosa et al., 2011) implementation of the PLS, t-SNE and SVM algorithms with their default parameters. For the saliency map computation, we re-implemented the perturbation-based method proposed by Greydanus et al. (2018). This technique consists in measuring the saliency of a given pixel by computing the difference in the network outputs when fed with a sampled state or with a blurred version of that same state (using a Gaussian blur).

Every network training for this paper was realised on GPUs, mainly on Nvidia V100s cards. For the toy environment of section 3, we trained each expert for 50 iterations of 1000 time steps which took around 15 minutes per expert, and each student for 25 iterations of 1000 time steps which took around 1h30. For each Procgen Game, each expert was trained for 30 iterations of 25000 time steps for a computation time of around 1h20, and each student for 8 iterations of 25000 time steps during around 12h. Finally we trained each expert on any Atari games for 60 iterations of 50000 time steps (around 6h of computation) and each student for 15 iterations of 50000 time steps (around 2 days).

B   SEPARABILITY OF STATE EMBEDDINGS ON THE CONTROLLED EXPERIMENT OF SECTION 3

A more detailed version of Table 1 is given in Table 5. For each of the 24 levels, we train an expert, compute the PLS features, keep either the 3 main ones or all 32, and train the linear SVM on these features to imitate the expert's policy. We compute the classification accuracy, and then average the results across experts. We repeat this training 20 times and average across these 20 repetitions. This computation is done once and for all and thus we report the same values along the "expert" line, whatever the number of levels used for distillation. These are the same experts that are then used for distillation. For the "students" line however, only experts from levels 1 and 2 are used in the first column, then levels 1 up to 6 for the second and so on. The reported value is thus the average over the 20 repetitions.

| | $N_{levels} = 2$ | | $N_{levels} = 3$ | | $N_{levels} = 4$ | |
	dim = 3	dim = 32	dim = 3	dim = 32	dim = 3	dim = 32
expert	78.05 (4.90)	91.29 (2.06)	78.05 (4.90)	91.29 (2.06)	78.05 (4.90)	91.29 (2.06)
student	75.56 (6.27)	91.60 (2.16)	80.65 (4.87)	93.60 (1.22)	81.33 (4.67)	93.43 (1.87)

| | $N_{levels} = 6$ | | $N_{levels} = 12$ | | $N_{levels} = 24$ | |
	dim = 3	dim = 32	dim = 3	dim = 32	dim = 3	dim = 32
expert	78.05 (4.90)	91.29 (2.06)	78.05 (4.90)	91.29 (2.06)	78.05 (4.90)	91.29 (2.06)
student	83.06 (5.66)	93.68 (1.87)	88.03 (1.56)	95.37 (0.75)	91.01 (4.60)	95.81 (1.78)

Table 5: Average (and standard deviation) of accuracy for a linear classifier built on the state representation.

C   ADDITIONAL DETAILS ON THE ATARI BENCHMARK

In this section, we motivate decisions made concerning the Atari benchmark, notably stopping training early and the lack of a robustness study.

For Atari games, we stopped training before convergence for experts and students. For instance, the expert had not converged after 60 iterations in Breakout, and 15 iterations is not enough for the student to match the expert performance on Breakout. This was motivated by different factors:

- the convergence is not guaranteed as the task of imitating 5 experts at the same task is far from trivial,
- the distillation process aims at extracting general features and we argue this can be done before convergence of the policy,
- the computation time would have been intractable in order to run 10 seeds for each experiment.

For comparison, our training time was around 6h for 60 iterations on a single seed; Dopamine (Castro et al., 2018) measured iterations of 250,000 timesteps (5× ours) and trained for 200 iterations, in other words, roughly 16 times more timesteps than our experiments. We determined that running each seed for 100 GPU hours was unnecessary for studying distillation.

In section 4.2, we choose to focus on Procgen for evaluating the robustness of student networks. We argue that evaluating the robustness of the student network on Atari games is not as relevant for this work. We already know from the distillation process that the student is efficient on different instances of the same game used to train the experts (Figure 6). However, as the underlying dynamics of two Atari games are very different from each other, we would not expect the student's policy to perform well on a new game based solely on a good state representation. An open question would be to evaluate how the learned state representations could be used to initialize a new expert and how the training would compare to an uninformed initialization. This would amount to using distillation as a means to perform meta-learning as in the work of Finn et al. (2017); Nichol et al. (2018). Following the recent contribution of Schwarzer et al. (2021) we argue that the question of transferring state representations as good initializations for a neural network is a separate topic and reserve it for future work.

D   DISTILLATION PSEUDO-CODE

Algorithm 1 summarizes the pseudo code corresponding to the algorithmic operations of Sections 3 and 4.

Algorithm 1 Expert training then distillation

1: **for** $i \in [1, N_{levels}]$ **do**   ▷ Train each expert separately
2:     Draw MDP $\mathcal{M}$ and default initialization of π_E
3:     Train π_E on $\mathcal{M}$ for K episodes
4:     Store $(\pi_E, \mathcal{M})$ in the list $\mathcal{E}$ of experts
5: **end for**
6: Default initialization of $\pi_{student}$
7: **repeat**   ▷ Train student
8:     **for** $\pi_E, \mathcal{M} \in \mathcal{E}$ **do**
9:         **for** all time steps t in one episode **do**
10:             Add sample $(s_t, \pi_E(a|s_t))$ in a training set $D_{\mathcal{M}}$ by playing $\pi_{student}$ in $\mathcal{M}$
11:             Sample a batch of states $\mathbf{s} \sim D_{\mathcal{M}}$
12:             Take one gradient step on $\pi_{student}$ to minimize imitation loss L
13:         **end for**
14:     **end for**
15: **until** maximum number of episodes reached
16:
17: For plain distillation (Section 3): $L = \sum_{\mathbf{s}}[\pi_E(a|s) \log \pi_{student}(a|s)]$
18: For AMN on visual tasks (Section 4): $L = \sum_{\mathbf{s}}[\pi_E(a|s) \log \pi_{student}(a|s)] + \beta \|\phi_E(s) - f(\phi_{student}(s))\|_2^2,$
19:     with f the adaptation layer and ϕ the feature layer of each network (see Section 4.1 and (Parisotto et al., 2016)).

E   DISCUSSION ON THE RESULTS ON THE BIGFISH BENCHMARK

The results of the student agent on the BigFish game might appear mitigated and deserve a separate discussion. One first remark is that the student does not do so badly: experts trained on a level and evaluated on another have an average score around 2 (figure 6, bottom center). The student network has an average score of around 14. This is well below the average score of experts evaluated solely on their expertise level (30), but still better than naive transfer. Possible immediate causes for such a mitigated result are (a) insufficiently long distillation, (b) insufficient variety of training levels or (c) convergence to a policy that does not generalize enough. Reason (a) is unlikely since various lengths were tested for distillation. It should be noted that the standard protocol proposed by Cobbe et al. (2020) uses between 100 and 100000 training levels, and states that at least 10000 levels are necessary to close the generalization gap for single agent training. Recent works that use this benchmark (e.g. RAD (Laskin et al., 2020), IDAAC (Raileanu & Fergus, 2021) or CLOP (Bertoin & Rachelson, 2022)) typically use 500 levels for training, which makes (b) a plausible explanation. This hypothesis is also supported by the fact that even though the student has performance 14 on the training levels, it does not generalize at all on the testing ones (Figure 8). Finally, it is known that SGD acts as an implicit regularizer (Gunasekar et al., 2017) that leads the optimization process towards a single minimum even with few samples, but this minimum has no guarantee of being generalizable across tasks, as studied by Song et al. (2020), hence hypothesis (c).
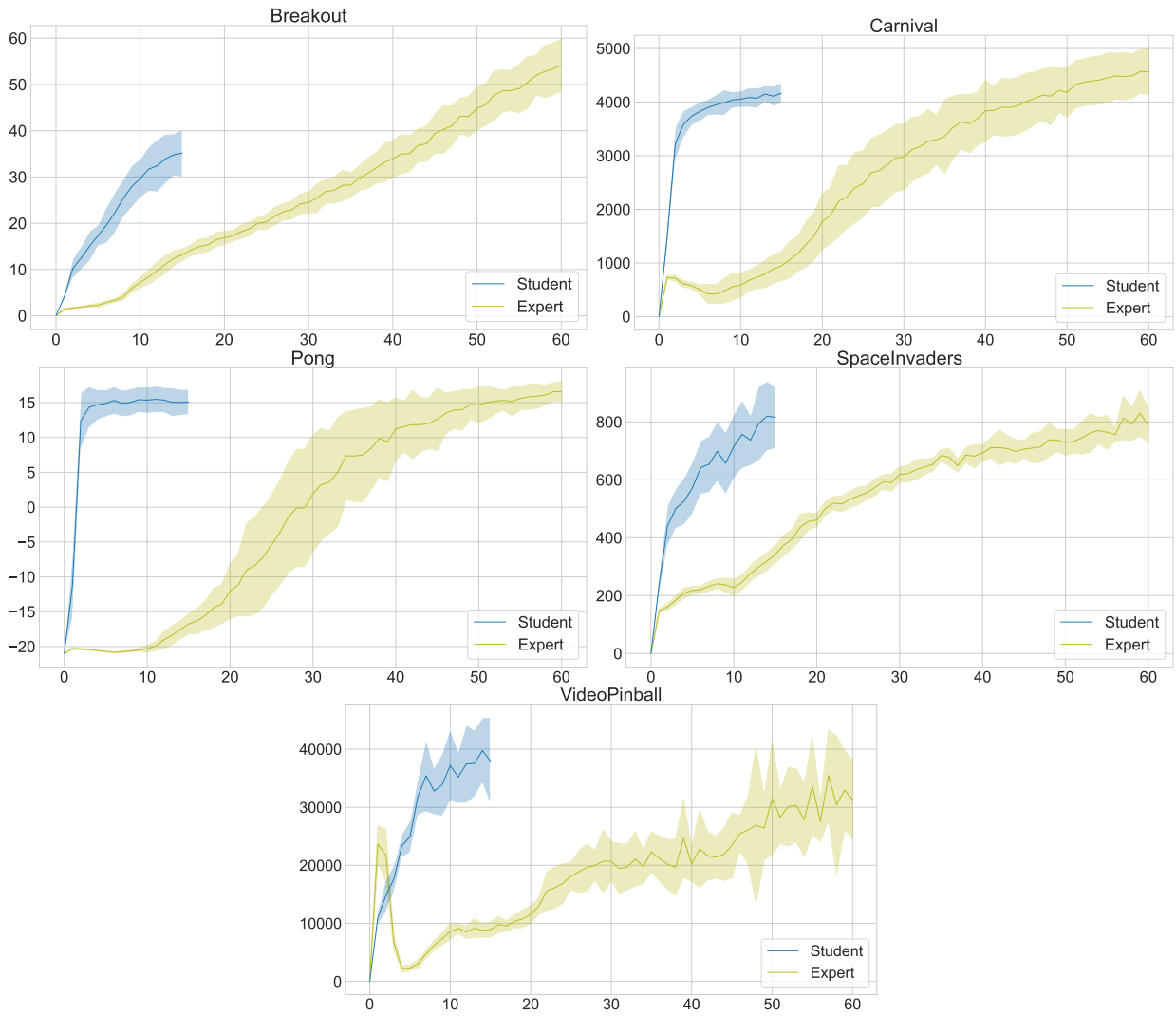
F   TRAINING CURVES ON VISUAL CONTROL TASKS

Figure 9: Training curves for the experts and the student on visual control tasks.
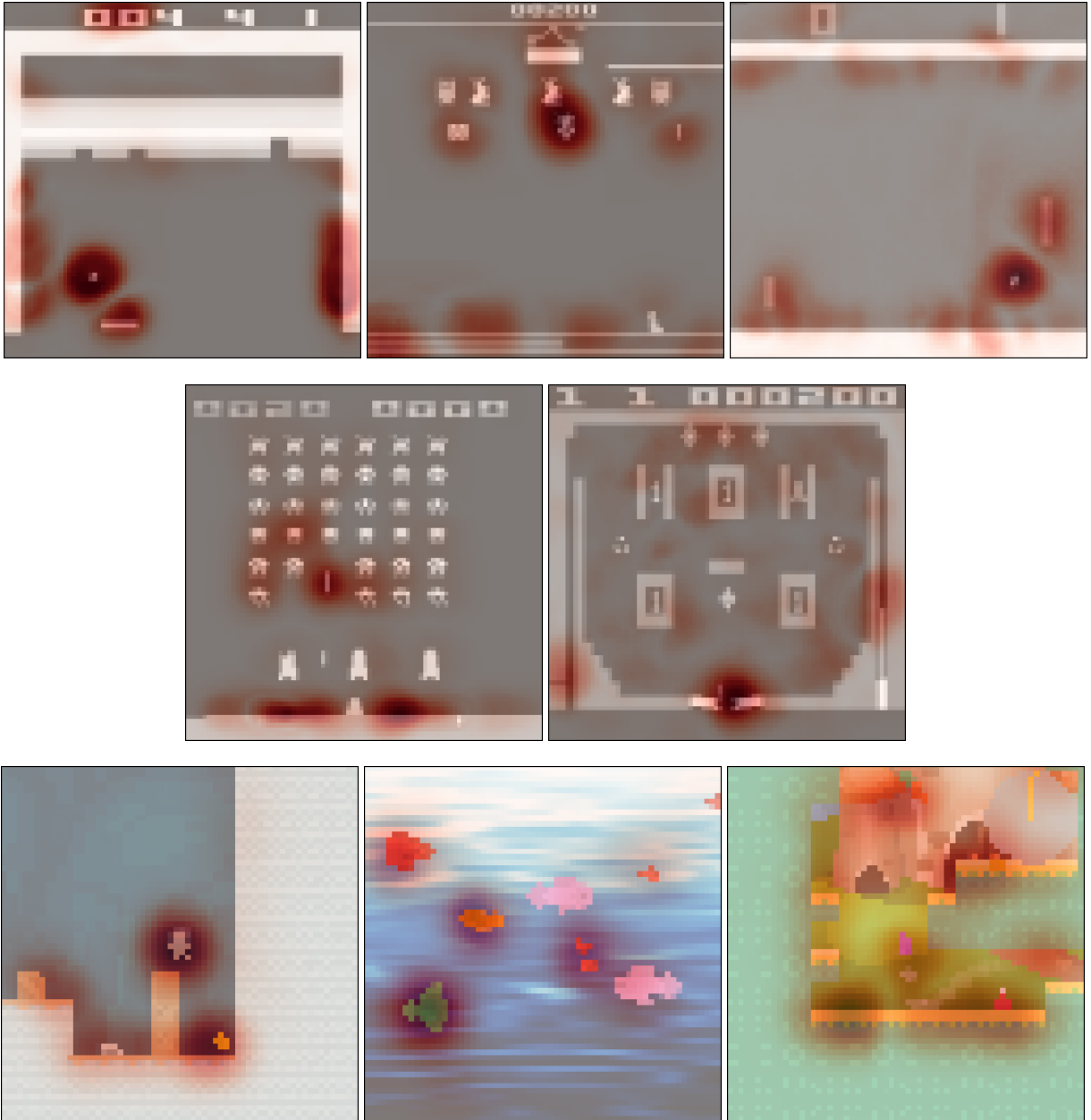
G SALIENCY MAPS

Figure 10: Saliency maps of expert networks superimposed on states sampled from each visual task.

H SEPARABILITY OF STATE EMBEDDINGS ON VISUAL TASKS

Figures 11, 13, 12, 14, 15, 16 and 17 report the t-SNE and PLS representations of the state embeddings for the experts and the students.
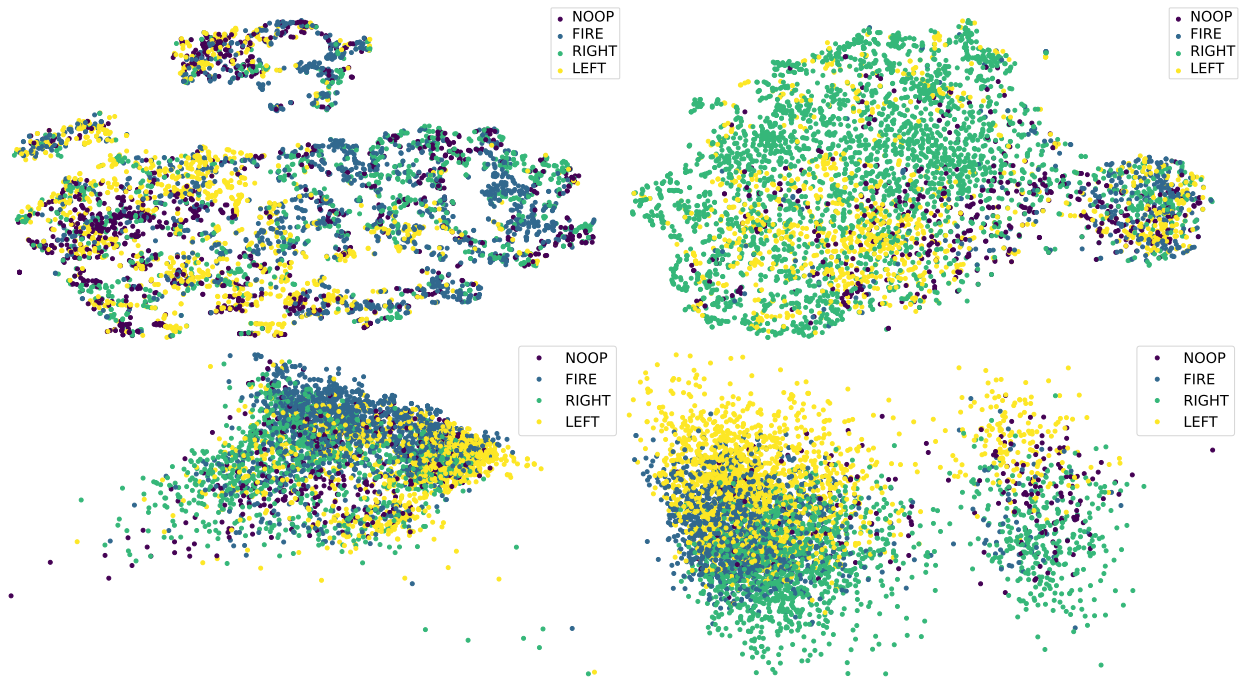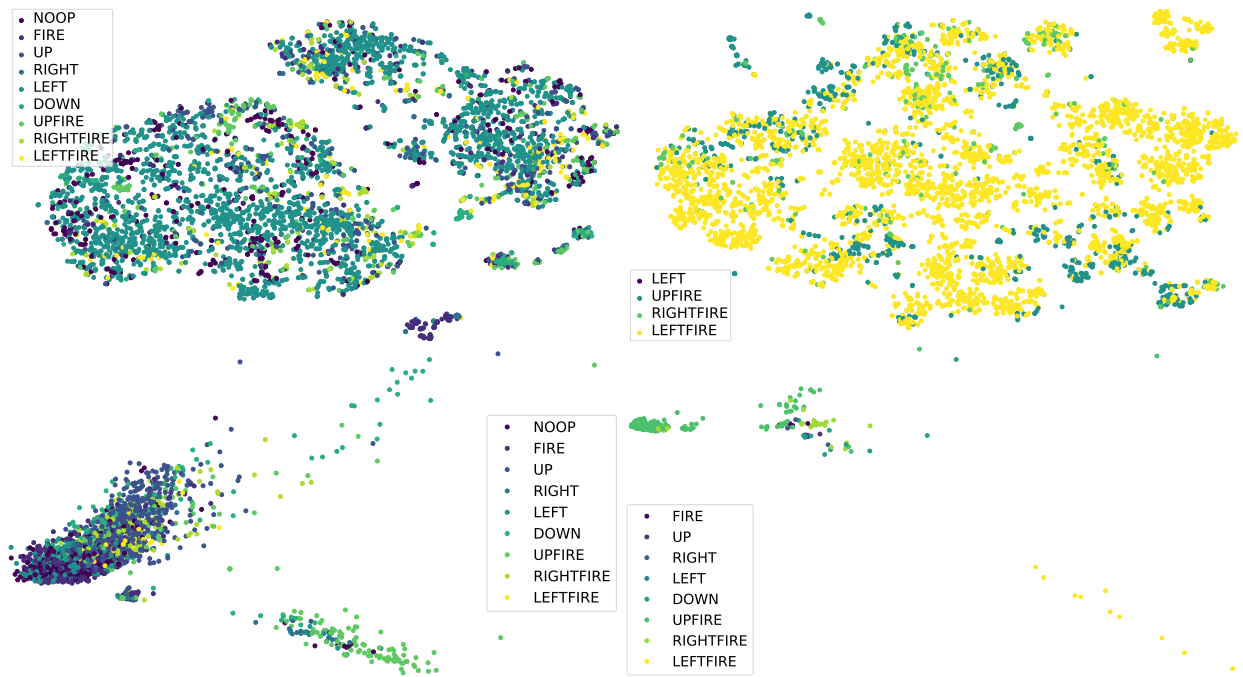

Figure 11: t-SNE (top) and PLS (bottom) plots of the state embedding for an expert (left) and the student (right) on Breakout.
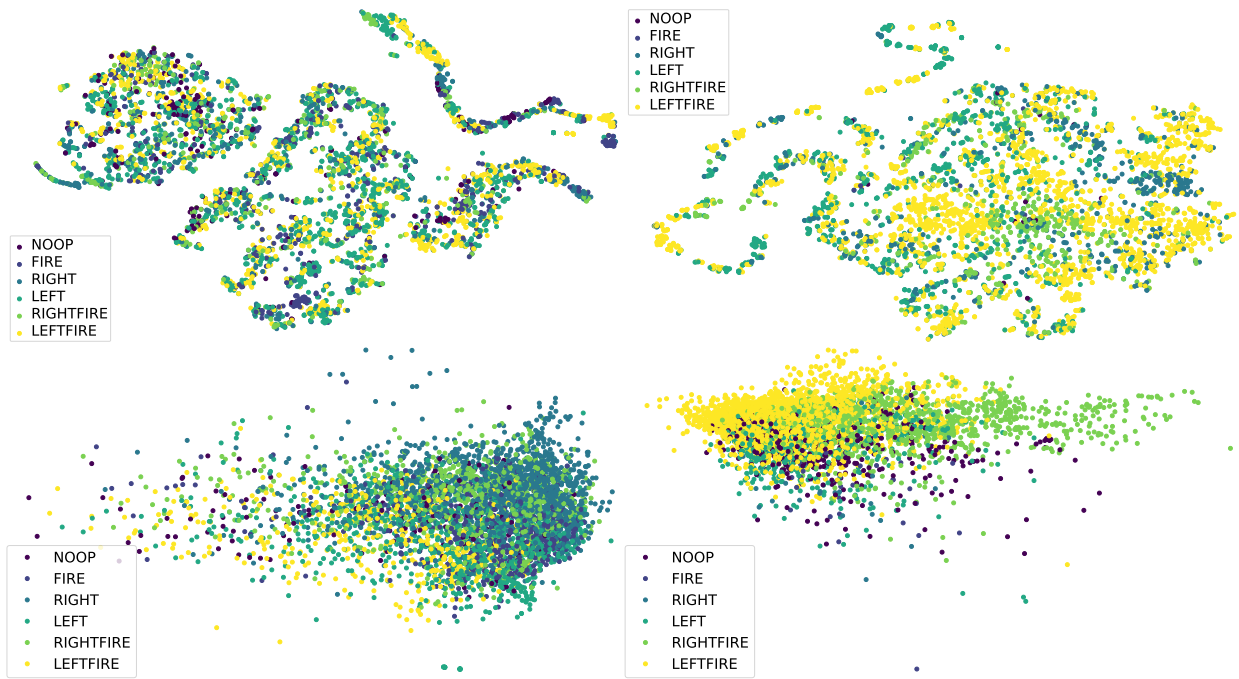
Figure 12: t-SNE (top) and PLS (bottom) plots of the state embedding for an expert (left) and the student (right) on Carnival.
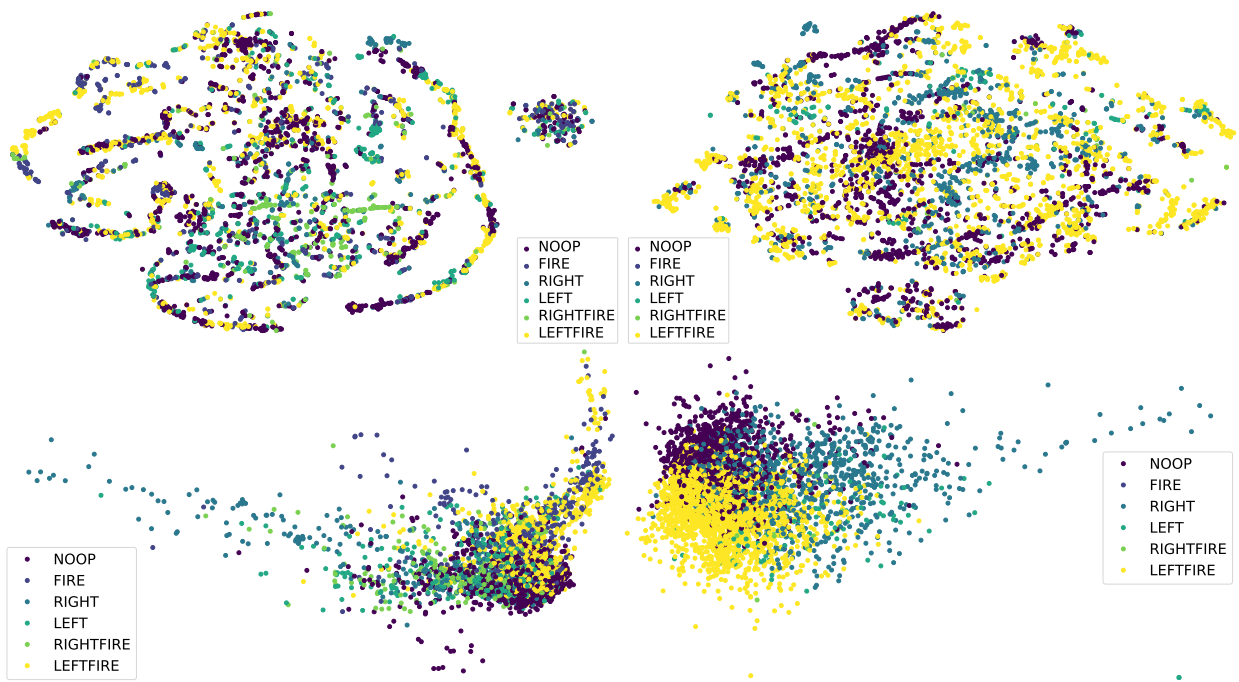
Figure 13: t-SNE (top) and PLS (bottom) plots of the state embedding for an expert (left) and the student (right) on Pong.

Figure 14: t-SNE (top) and PLS (bottom) plots of the state embedding for an expert (left) and the student (right) on VideoPinball.
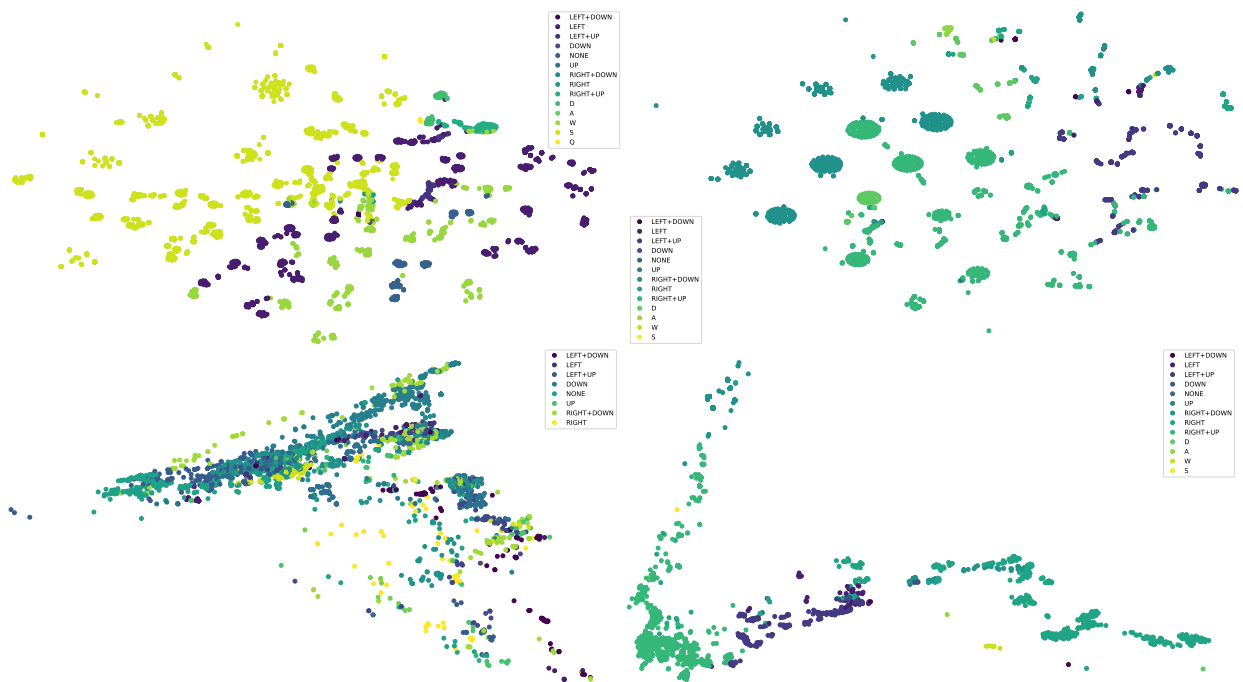

Figure 15: t-SNE (top) and PLS (bottom) plots of the state embedding for an expert (left) and the student (right) on CoinRun.
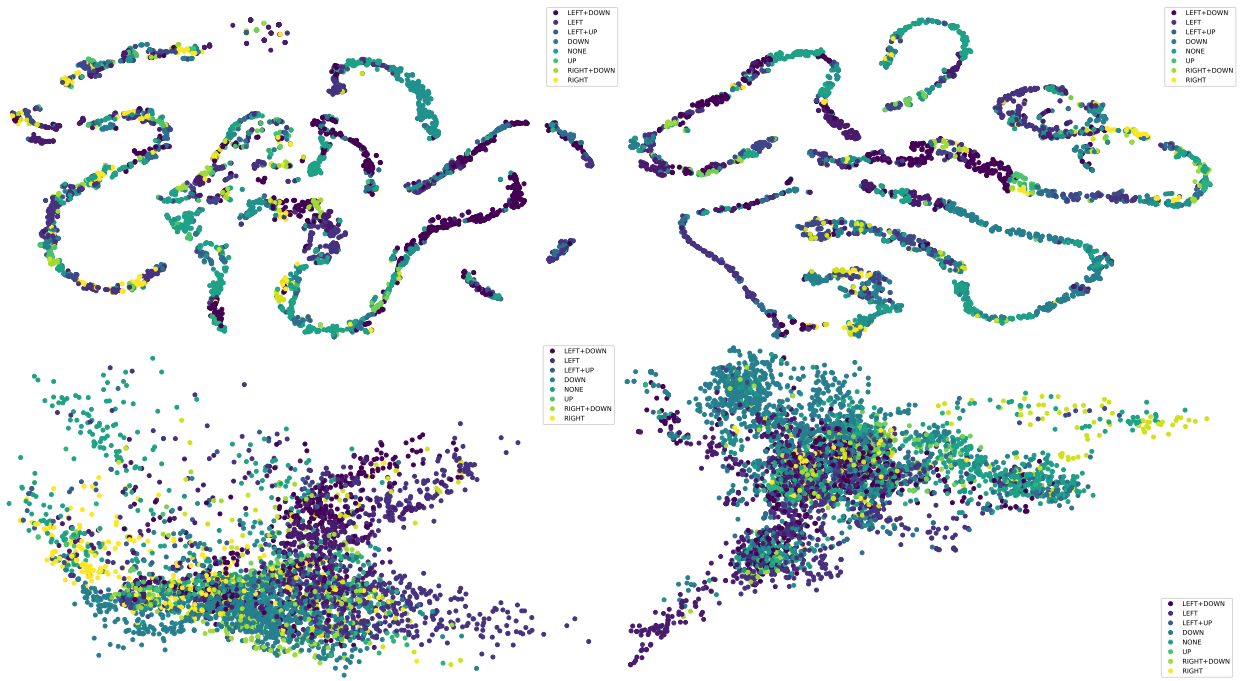
Figure 16: t-SNE (top) and PLS (bottom) plots of the state embedding for an expert (left) and the student (right) on BigFish.
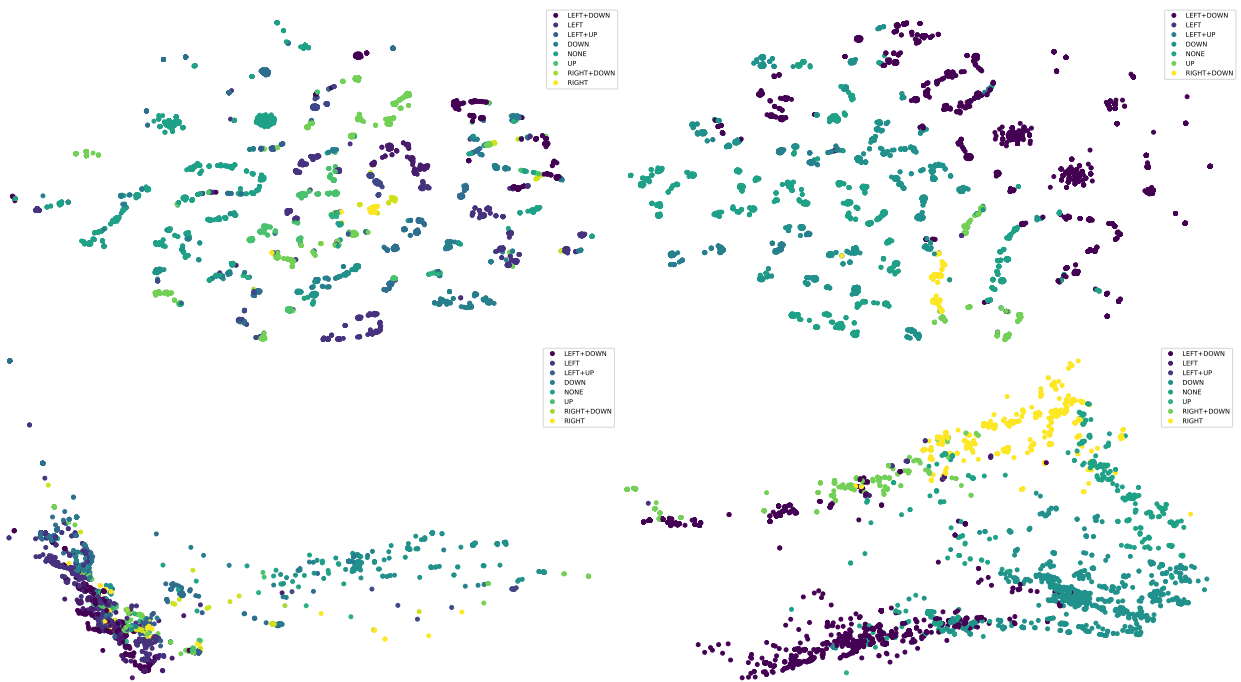

Figure 17: t-SNE (top) and PLS (bottom) plots of the state embedding for an expert (left) and the student (right) on Jumper.