Learning Reliable PDDL Models for Classical Planning from Visual Data

Aymeric Barbin
Dipartimento di Ingegneria Informatica
Sapienza Università di Roma
Roma, Italy
aymeric.barbin@uniroma1.it

Federico Cerutti
Dipartimento di Ingegneria Informatica
Università degli Studi di Brescia
Brescia, Italy
federico.cerutti@unibs.it

Alfonso Emilio Gerevini
Dipartimento di Ingegneria Informatica
Università degli Studi di Brescia
Brescia, Italy
alfonso.gerevini@unibs.it

2024 IEEE 36th International Conference on Tools with Artificial Intelligence (ICTAI) | 979-8-3315-2723-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICTAI62512.2024.00107

***Abstract*—We propose R-latplan, a system that learns reliable symbolic (PDDL) representations of an agent's actions from noisy visual observations, and without explicit human expert knowledge. R-latplan builds upon Latplan, a model that also learns PDDL representations of actions from images. However, Latplan does not ensure that the learned actions correspond to the actual agent's actions and does not map the learned actions to the agent's actual capabilities. There is, therefore, a substantial risk that a learned action could be impossible due to the domain's physics or the agent's capability. R-latplan receives input pairs of (noisy) images representing the states before/after the agent's action is performed in the domain. Contrary to Latplan, it uses a transition identifier function that identifies the class of a transition and associates it as an action label for the pair of images. Our experimental analysis shows that: (1) R-latplan produces reliable PDDL models in which each action can be directly connected to an agent's high level actuators and lead to visually correct states (the agent does not hallucinate), (2) R-latplan generated PDDL models lead to a domain-independent planner to find optimal plans on each benchmarks considered, (3) R-latplan is robust against mislabeled transitions, i.e. if errors are introduced in the transition identifier function.**

***Index Terms*—planning, neuro-symbolic learning**

I. Introduction

The bottleneck when building a planning system is typically the knowledge representation part, which is often left to human expert designers. Several works (see Section II) have attempted to learn lifted representations from images. In [1] and [2], they propose an unsupervised and domain-independent Deep Learning model that uses a parser to automatically retrieve predicates from images, but this approach relies on the parser capacity to output the right objects and predicates. In [3], the authors also learn a lifted representation by first learning objects and their locations in the image using the pixels changes in transitions, but this does not scale well on large state spaces and is not robust against noise. In [4], they use a semi-supervised approach to specify the predicates and action signatures of the trace's last states, they use basic known domain dependant predicates to ground the images.

On the other hand, in [5], the authors (Sections II and III) built a system, called Latplan, that learns to generate action models with STRIPS-like semantic [6] directly using the propositional latent representations learned by training a Variational AutoEncoder (VAE) from image pairs; yet, the actions' labels themselves are also learned during training, and therefore are prone to hallucination, in that they can lead to states that do not actually exist (images with artifacts, not seen in the training data). Furthermore, nothing in Latplan relates the learned action's label to the actual actions performed by an agent's actuators.

In this paper, we build upon Latplan and we propose **R**eliable-Latplan (R-latplan), a system which: **(1)** produces reliable PDDL action models where each action can be directly connected to an agent's high level actuators and generates visually correct states (the agent does not hallucinate) **(2)**, learns efficient PDDL models that state of the art planners can use to find optimal plans on all the benchmarks considered **(3)**, and is robust against mislabeled transitions, so that when errors are introduced in the transition identifier function, the PDDL domain produced by R-latplan casts the wrongly labelled transitions back to their correct label. To the best of our knowledge R-latplan is the first architecture that learns reliable actions schemes from high dimensional input without any supervision and without relying on a symbolic parser.

II. Related work

Latplan [5] is a pioneering work in learning symbolic planning domains from visual data. It uses a variational autoencoder framework to learn a discrete probabilistic representation of the images, jointly with the action labels associated with transitions and the preconditions and effects. It can then generate these actions in a PDDL-STRIPS like semantics. Importantly, there is no direct relation between the actions generated and the real physical action that can be executed by the agent in the real world. As our proposed system, R-latplan,

This work was partially supported by the European Office of Aerospace Research & Development (EOARD) under award number FA8655-22-1-7017 and by the US DEVCOM Army Research Laboratory (ARL) under Cooperative Agreements #W911NF2220243. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States government. This work was also partially supported by the EU H2020 project AIPlan4EU (GA101016442), the EU ICT-48 2020 project TAILOR (GA952215), and the Italian MUR PRIN project RIPER (No. 20203FFYLK). This work has been carried out while Aymeric Barbin was enrolled in the Italian National Doctorate on Artificial Intelligence run by Sapienza University of Rome in collaboration with the University of Brescia.

2375-0197/24/$31.00 ©2024 IEEE
DOI 10.1109/ICTAI62512.2024.00107

	[3]	[2]	[4]	[1]	[5]	R-latplan
Subsymbolic representation learning	NO	YES	YES	NO	YES	**YES**
No need of human-labelled data	NO	YES	NO	YES	YES	**YES**
Reliability of the learned action models	YES	NO	NO	YES	NO	**YES**

TABLE I: Gap analysis with the existing state-of-the-art.

builds upon Latplan, we will expand its description in detail in Section III.

In [3], the authors present an approach for automatically generating symbolic action definitions using unlabelled pairs of images representing transitions. They identify objects and their locations by detecting pixel areas changes in a transition. Then, they provide each *object* and *location* with a unique identifier with which they ground two general predicates for all the images: `(at ?1 - location ?2 - image object)` and `(clear ?1 - location)`. To ground the actions, they use the atoms that change within a transition as *preconditions*, and the value of these changes as *effects*. This approach does not scale well on large state spaces, e.g. the search for new *locations* grows exponentially with the size and complexity of the images. Also, it is not robust against noise since any change in a pixel colour can result in a different *location* area and in the duplication of semantically equal actions. As demonstrated in the following, our proposed system, R-latplan, is domain-independent and partly robust to noisy images.

In [2], the authors implement a system that uses a deep auto-encoder network that takes as input an action and a camera image and outputs the pixel changes happening in a transition. The discrete latent layers hold the information of the image changes associated with an action, and their output is used to train a decision tree that learns probabilistic rules about the action's effects. Their approach works well for tasks not requiring a global representation of the states, like moving objects with a robot arm but fails on tasks where it is needed, like the MNIST sliding puzzle.

In [4], the authors developed a model that learns a lifted action model from visual traces. In particular, they learn first-order representation of states and actions from sequences of image-action pairs. Their semi-supervised approach uses a manually labelled final state for each trace, therefore, the burden of manually labelling the final state grows exponentially with the complexity of the problem.

In [1] the authors use a top-down approach where they search, using a SAT solver, in a domain space defined from predicates returned by an image parser, in particular the domain to be found must complies with all the partial graphs representing pieces of the state space given as input. As in the other papers presented here, using a parser for retrieving predicates in the images poses a problem since it is prone to identification errors. Furthermore, the solving time the SAT solver uses to find the best domain grows exponentially with the domain complexity.

Table I summarises the similarities and differences between the state-of-the-art and our proposed system, R-latplan.

III. Background

A classical planning problem [7] is defined by the tuple $\langle S, A, s_0, G\rangle$ where S is a finite set of states, A is a set of actions, s_0 is an initial state, G is a goal state. $\pi(s, a)$ is a state transition function that gives the new state resulting from executing action a in state s. In the STRIPS formalization of a planning problem, *P* is a set of propositions and each action $a \in A$ is defined as a 4-tuple $\langle \text{POS}(a), \text{NEG}(a), \text{ADD}(a), \text{DEL}(a)\rangle$, where: i) $\text{POS}(a)$ and $\text{NEG}(a)$ are the positive and negative preconditions, respectively, both subsets of P, ii) $\text{POS}(a) \cap \text{NEG}(a) = \emptyset$, iii) $\text{ADD}(a)$, $\text{DEL}(a)$ are the add-effects and delete-effects, respectively, both subsets of P, iv) $\text{ADD}(a) \cap \text{DEL}(a) = \emptyset$.

A Deterministic Finite Automaton (DFA) [8] is defined as a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where Q is a finite set of states, Σ is a finite set of input symbols, also known as the alphabet, $\delta : Q \times \Sigma \to Q$ is the transition function, which maps a state and an input symbol to another state, $q_0 \in Q$ is the initial state from which any inputs are processed, $F \subseteq Q$ is the set of accepting states (also called final states). The operation of a DFA is characterized by the sequence of transitions between states as dictated by the transition function δ. For an input string $s = s_1 s_2 ... s_n$ where each $s_i \in \Sigma$, the DFA starts at state q_0 and processes each symbol of s in sequence. The next state is determined by $q_{i+1} = \delta(q_i, s_{i+1})$. If, after processing the entire string, the DFA is in one of the accepting states ($q_n \in F$), then the string is accepted by the DFA; otherwise, it is rejected.

The mapping from a classical planning problem to a DFA is as follows [9]:

1) the states of the DFA (Q) correspond to the states in the planning problem (S).
2) the alphabet of the DFA (Σ) corresponds to the set of actions in the planning problem (A).
3) the transition function of the DFA (δ) corresponds to the state transition function in planning. Thus, $\delta(s, a) = \pi(s, a)$.
4) the initial state of the DFA (q_0) is the initial state of the planning problem (s_0).
5) the accepting states of the DFA (F) correspond to the states that satisfy the goal condition (G) in the planning problem.

We call such DFA a *planning DFA*, and we represent it with the tuple (S, A, π, s_0, G).

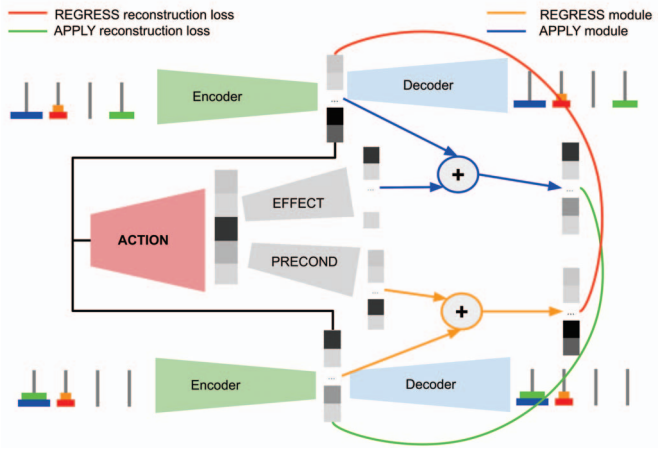

Fig. 1: High-level illustration of Latplan [5].

In the following, whenever we refer to a DFA, we intend a DFA that is the result of a mapping from a classical planning problem. Note that each transition in the planning DFA corresponds to an action in the planning domain, we use both term interchangeably in the paper.

Latplan [5] is a Neuro-symbolic planner that can learn a PDDL representation of the domain, from pairs of images representing transitions in a visual domain. It consists mainly of two parts: the State Autoencoder (SAE) and the Action Model Acquisition module (AMA4+ in the latest version). The SAE, is a Variational Autoencoder [10] that learns a multinomial binary latent representation of the image, which can then be transformed into a propositional formula. The AMA module learns to associate an action's label and preconditions to each transition, and then to predict the next state (and corresponding image) from the current state and the action label. It is composed of:

- an ACTION network, composed of a dense layer of size 6000 followed by a Gumbel Softmax [11] (GS) activation layer. It takes as input two latent states representing a transition and outputs a continuous GS vector that converges, during training, toward a one-hot vector representing the label of the action;
- an EFFECT network, that takes as input an action and has a GS layer as output, of the size of the latent state representation (N);
- an APPLY module, that applies an action's effects to a state and predicts the next state (we discuss it in more detail below).

Fig. 1 shows an overview of Latplan's architecture.

Importantly, Latplan also learns the precondtions of an action with the same mechanism as for the effects. For this, it uses the PRECOND network (for predicting the preconditions) and REGRESS module (for predicting the first state of a transition).

The Latplan APPLY module in AMA4+ coincides with the Back To Logit (BTL) Mechanism, a technique for handling symbolic state or action representations while enabling backpropagation. It converts these discrete symbols into logits via an embedding layer; then the softmax function outputs probability distributions and feeds them into a neural network. Loss computation, backpropagation, and gradient descent update the model, improving it iteratively.

Latplan's loss function is very articulated. Latplan models the likelihood of observing an images transition, $p(x^0, x^1)$, since this computation is known to be intractable [12], an ELBO (Evidence Lower BOund) is used as a loss function. The ELBO approximates the Likelihood from below and it is a sum of the different losses and KL divergences [13] composing the network. For simplicity, we illustrate the figures only with the reconstruction loss of the APPLY and REGRESS network.

Latplan's resulting action schemas are built from the grounded values of the latent states vectors generated by the SAE and the EFFECT/PRECOND network. An example is:

```
(:action a15 :parameters () :precondition
 (and (not (z0)) (not (z1)) (z2) (not (z5)))
 :effect
  (and (not (z0)) (not (z1)) (z2) (not (z5)) (not (z6
      ))))
```

The learned representation can then be used with any classical planner. In the following, we will rely on Fast Downward (FD) [14], which converts planning problems in PDDL into a finite-domain representation, optimizing state encoding for efficient heuristic computation. In particular, we use the Blind heuristic within FD, which treats all states uniformly using their distance to the goal. This simplicity ensures broad applicability across different domains. The heuristic's neutrality makes it ideal for gaining fundamental insights into planning problems and comparing results across various approaches.

IV. Methodology

We assume to have access to a set of visual traces, e.g. what can be captured by a camera when observing an autonomous agent operating in an environment, eventually under the control of a teleoperator. A visual trace is a sequence of images, each of which is a visual description of a state of the world. Hence, we assume to have access only to state observations as images taken at discrete time steps, and pairs of images, each of one corresponding to an observed states transition. We also have access to the agent's actions, e.g. in Hanoi *move(d3, d4)*, in Sokoban *go(right)*, in Blocksworld *pickup(block_A)*. We refer to these actions as **high level actions**.

In the following, we give some definitions that are useful to specify the problem we want to solve.

We define a **Visual DFA** $\mathcal{V}$ as an augmentation of a *planning DFA* where each state s_i of $\mathcal{V}$ is associated with a pair (I_i, Z_i), where I_i is a set of images, and Z_i is a set of propositional representations of the images in I_i (or encodings in a latent space of propositional states). In particular, given a planning DFA (S, A, π, s_0, G), let $vimg$ be a function that maps each $s_i \in S$ to a set of images I_i that visually represents s_i, i.e., $vimg(s_i) = I_i$. Let also $img2prop$ be a function that maps each image $x_{i,l} \in I_i$[1] to a symbolic propositional representa-

[1] l indexes an image in the set I_i.

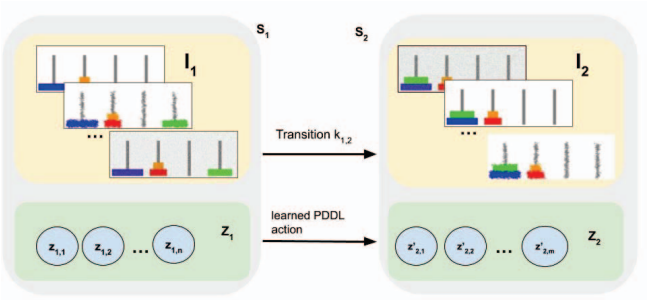

Fig. 2: An illustration of two states in the Visual DFA connected by a transition.

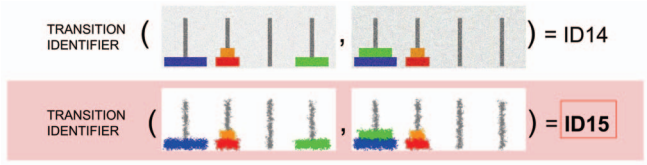

Fig. 3: An illustration of the case where the transition identifier function labels erroneously two versions of the same transitions with different labels (ID14 and ID15 in the figure), due to visual noise.

tion $z_{i,l} \in Z_i$, i.e., $img2prop(x_{i,l}) = z_{i,l}$. For each transition $k = (s_i, s_j)$ of $\mathcal{V}$, we have that $\forall x_i \in vimg(s_i)$, executing k when x_i is the current world observation, results in the world to become a state observed as an image $x_j \in vimg(s_j)$, see Figure 2.

A Visual DFA $\mathcal{V}$ is **noisy** (resp. **crisp**) if for at least one of its states (resp. each of its states) s_i, $|vimg(s_i)| > 1$ (resp. $|vimg(s_i)| = 1$). A Visual DFA $\mathcal{V}$ is **complete** if it contains all the possible transitions of the domain (**partial** if not).

We also have a **transition identifier function** that returns a unique identifier for a given pair of states in a given Visual DFA. Figure 3 shows an example of an erroneous assignement of ID from the transition identifier function.

We define a **reliable PDDL action** as a PDDL action that can be associated with full certainty to an action performed by an agent. We refer to the transitions in the visual DFA as **low level actions**.

Since we know for each transition performed by the agent, the label returned by the transition identifier and the high level action label of the agent, we can make a direct relation between both. Generally, a high level action corresponds to multiple low level actions (e.g. *pickup(block_A)* corresponds to the set of all the transitions (low level actions) where the preconditions of the action *clear(A)* and *handempty* are true in the source state of the transition). Based on these definitions, we formulate our problem as learning each transition of a Visual DFA as a **low level** and **reliable** PDDL action.

For each transition in $\mathcal{V}$, the corresponding action is identified by the following binary vectors (of size N): neg_pred (negative preconditions), pos_pred (positive preconditions), add_eff (add effects), del_eff (delete effects). Such binary

	Data1,2		Data3,4		Data5	
	#states	**#transi.**	**#states**	**#transi.**	**#states**	**#transi.**
Hanoi	256	1459	234	1028	256	1539
Blocks	125	272	125	267	125	286
Sokoban	1270	4152	256	4069	1270	4380

TABLE II: Number of unique states and transitions.

representations can then be directly transformed into a propositional PDDL domain action. The workflow of R-latplan is summarize in figure 4.

We use Latplan's SAE for learning the $img2prop$ function. Moreover, given two states s_i and s_j and a corresponding transition label k identified by the transition identifier, we obtain the effects and preconditions of k (now, intepreted as a PDDL action) with the following:

- $ADD(a) = APPLY(a, \mathbf{0})$
- $DEL(a) = \mathbf{1} - APPLY(a, \mathbf{0})$
- $POS(a) = REGRESS(a, \mathbf{0})$
- $NEG(a) = \mathbf{1} - REGRESS(a, \mathbf{0})$

where $\mathbf{0}, \mathbf{1} \in \{0, 1\}^N$.

We built the transition identifier function so that, given two images of a transition and a dictionary of transitions labels as keys and reference transition images as values, it checks if the two images correspond to one of the existing transitions, if yes, it returns the identified label, if not it creates a new entry in the dictionary and returns a new label.

Since we now know the label for each transition of the DFA, we removed the ACTION network and replaced its output with the one-hot encoding of the label (see figure 4 top right). The loss functions remain the same to simplify the comparison.[2]

V. Research Questions and Benchmarks

To demonstrate that R-latplan learns reliable symbolic (PDDL) representations of an agent's actions from noisy and incomplete visual observations, and without explicit human expert knowledge, we experimentally investigate the following research questions:

RQ1: Does R-latplan produces reliable actions schemes even in presence of a *noisy Visual DFA*?

RQ2: Does R-latplan produces action models that enable Fast Downward (FD) [14] to find optimal plans even when dealing with partial *noisy Visual DFA*?

RQ3: Is R-latplan robust against errors of the *transition identifier function*?

In order to answer these questions we prepared the five following pools of data.

Data1: Crisp and Complete Visual DFA contains all the transitions of the Complete Visual DFA, with only one image version for each state s_i ($|I_i| = 1$)

[2]R-latplan's source code is available at https://github.com/aymeric75/R-latplan.

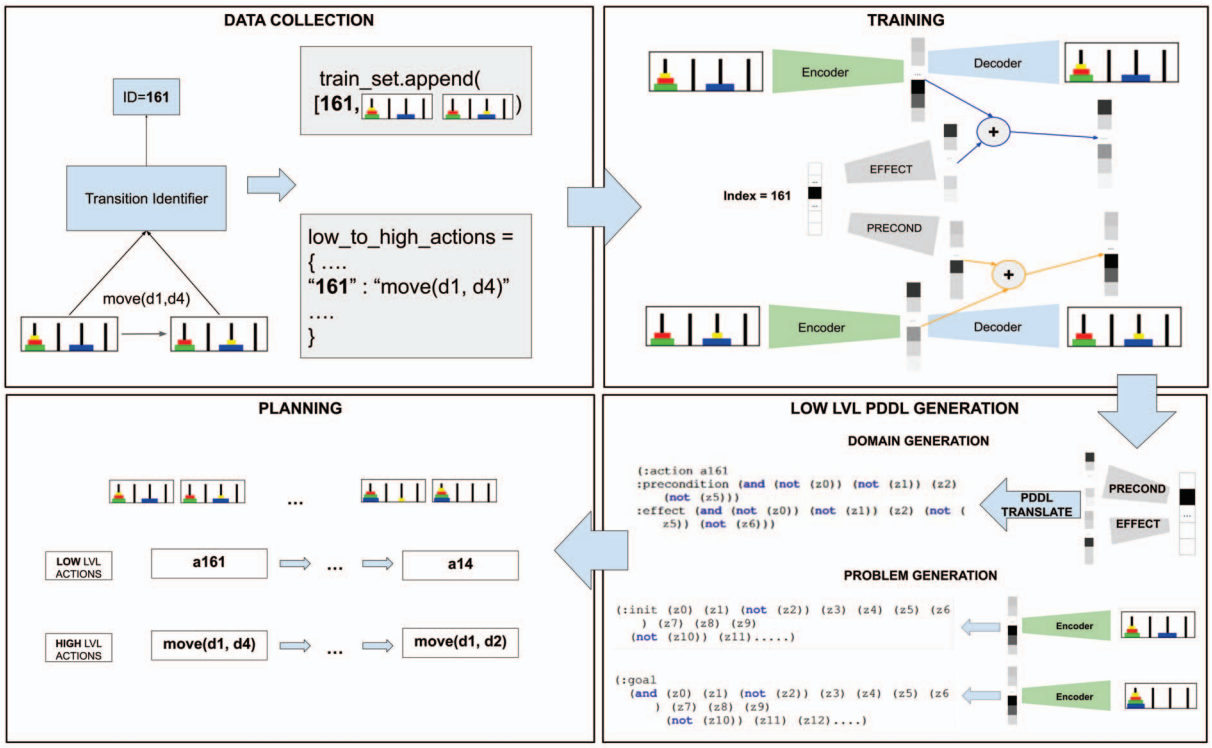

Fig. 4: The overall workflow of R-Latplan.

Data2: Noisy and Complete Visual DFA built on **Data1** but for 55% of the transitions, three different image versions ($|I_i| = 3$)

Data3: Crisp and Partial Visual DFA from **Data1**, we removed all but one possible path between two states of the Complete Visual DFA. We created three versions for each domain, with a different node removed for each.

Data4: Noisy and Partial Visual DFA from **Data2**, we followed the same procedure as described for **Data3**.

Data5: Noisy Visual DFA and Erroneous Transition IDs from **Data2**, we duplicated 5% of the transitions, assigning them a different label than the one returned by the transition identifier. This simulates the case of an erroneous *transition identifier function*.

Table II contains the number of states and uniques transitions in each pool of data.

The experiments we devised are as follows. For each, we trained R-latplan on the corresponding data pool.

Experiment 1 (resp. **Exp 2**), consists of training R-latplan with all the transitions of Data1 (resp. Data2), then testing the generated PDDL domain from Data1 (resp. Data2) on the three longest problems (i.e. for which the optimal plan is the longest Hamiltonian path[3] possible).

Experiment 3 (resp **Exp 4**), consists of training/testing R-latplan with all the transitions of Data3 (resp. Data4), one training for each version (i.e. with a different node and set of paths removed). The testing phase was performed with the three PDDL domains generated on the respective removed nodes taken as problems.

Experiment 5, consists of inspecting the generated PDDL from Data5, for the presence of redundant actions, i.e. actions that have the same effects, and for which in any state where one action preconditions' hold the other action's preconditions hold as well.

Experiments **1 to 4** address **RQ1** and **RQ2**, Experiment **5** adresses **RQ3**.

Benchmark Domains and Datasets. Our experimental analysis — similarly to [5] — builds on three well-known planning domains: Hanoi, Blocksworld, and Sokoban. In particular, we used the PDDLGym [15] implementation and the following settings. For the Hanoi domain, we considered four disks and four rods. In the Blocksword domain, we used coloured blocks and a single gripper. For the Sokoban domain, we considered an 8×8 grid, with 34 free cells and a single box.

It is worth mentioning that in [5] different versions of the same domains have been considered. For instance, the Hanoi domain considered simplified images, where there are no shape differences between disks; for the Blocksworld domain, [5] considered a photorealistic dataset; and for the Sokoban domain, [5] used a grid with 15 free cells and three boxes.

VI. Experimental Setup

Training and test sets. For generating the training sets of each domain, we derived first the *crisp DFA*, by using the PDDLGym step function from different starting states, in order to gather all the possible transitions (we kept one

[3]A plan that does not passes twice by the same node.

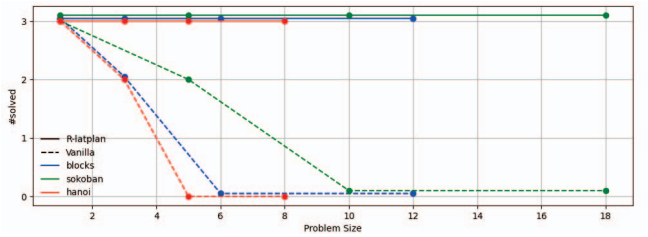

Fig. 5: Success number of tested executions (**maximum success is 3**) on the longest possible paths from the derived Visual DFAs on **Exp1**: *Crisp and Complete Visual DFAs* and **Exp2**: *Noisy and Complete Visual DFAs*

	R-latplan	Asai et. al
Hanoi	70x25	49x18
Blocksworld	54x54	27x27
Sokoban	43x43	22x22

TABLE III: Image resolutions in pixels

version of each transition) of each domain, each transition consists of a pair of images returned by the PDDLGym *render* function. Then, for deriving the different data pools (**Data1-5**), we associated to each image pair representing a transition, with its transition label, obtaining the latter by using the transition identifier function. Each image was pre-processed for training (see below). For **Data3,4** we leveraged the NetworkX library [16] to obtain a partial Visual DFA for which we identified — using the Dijkstra [17] algorithm — the longest traces.

Image preprocessing. Each image was reduced down to a certain size, determined empirically. In particular for training Latplan [5], we had to reduce the size almost twice compared to R-latplan (see table III), in order to obtain decent performances. We then followed the experimental setup of [5], starting from RGB values expressed in the floating point between 0 and 1, the image goes through histogram equalisation, then re-normalised to have an expected value of 0 and a standard deviation of 1.

Noising procedure. To create *noisy DFAs*, we applied to the final pre-processed image a Gaussian noise centered on 0 with a standard deviation of 0.015 to ensure that the overall noise is — with high probability — within 10% from the original pixel value.

Hyperparameter search. We ran a grid search for the following hyperparameters: N (size of the state latent space), tested for values 50, 100 and 300. β_z, the parameter weighting the role of the loss concerning the VAE priors, testing for 10, 100, 1000, and β_d, the parameter weighting of the next state prediction loss, tested for 10, 100, 1000.

We selected the combination of hyper-parameters leading to the lowest value of the loss term predicting the resulting transition accuracy (see figure 8). For Latplan, we had to expand the search space with additional variables and use a genetic search algorithm. [4]

Fig. 6: Not visually correct plan generated by (vanilla) Latplan

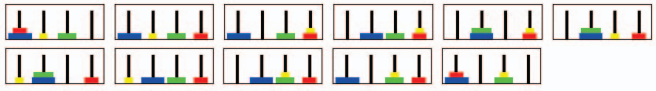

Fig. 7: Path found by FD from a PDDL generated by R-latplan in Data3 (with the 1-step transition/solution pruned out)

VII. Results

Regarding **RQ1**, i.e. if the generated PDDL actions are reliable, by design they all comply with the first condition, i.e. they are all linked to a high level action of the agent/teleoperator, since this is part of the data collection process (see figure 4). For the second condition, i.e., if for any action and any state on which the action is applicable, the corresponding generated images belong to a set I_i of images of the Visual DFA, we successfully checked this condition by checking if the distance from the canonical image returned by PDDLGym was under a certain threshold (the same threshold set in the transition identifier function). Furthermore, looking at the images by our human eye we can see that they are visually correct and do not display any odd artifact (e.g. duplication of an object, or wrong colorizations). On the other hand, none of the actions generated by Latplan [5] is reliable, because nothing in the architecture of Latplan links high level actions to the low level PDDL actions generated, furthermore, as one can see in figure 6, the images generated are often not visually correct.

Regarding **RQ2**, as shown in figure 5, R-latplan produced PDDL actions for which Fast Downward found all optimal plans on both experiments **1-2** with a Complete DFA, i.e. with/without noise, whereas Latplan's PDDLs lead to only few visually correct plans and only for limited problem size (on Sokoban the maximum problem size solved was six, and only on two instances out of three). Importantly, the noise introduced in the images for experiments **2-4** were exactly the same as in respectively 1 and 3, this means that Latplan and R-latplan are robust against the noise (up to certain amount).

For experiments **3-4**, i.e. with partial DFA, R-latplan generated PDDLs that lead Fast Downward to find the optimal plans for all problem considered (see an example in figure 7). On the other hand, Latplan generated PDDLs on which FD found plans of size 1 for all problems. This means that Latplan associated one of the actions it learned (out the 6000) to the 1-step path between the two images, yet this 1-step path was pruned out from the training dataset, if it shows the capacity of Latplan to generalize over unseen transitions it also prove the non reliability in that the produced actions are not conform to the data.

[4] which can be found in the latplan's original code https://github.com/guicho271828/latplan

	Lengths of surviving paths	Number of transitions (mean)
Hanoi	10, 10, 10	1028 (vs 1459)
Blocksworld	12, 12, 13	267 (vs 272)
Sokoban	18, 19, 20	4069 (vs 4152)

TABLE IV: lengths of paths found with R-latplan for **Exp3,4**

	Number of groups of redundant actions found	Ground truth
Hanoi	72	72
Blocksworld	14	14
Sokoban	207	207

TABLE V: Results of **Exp5** showing R-latplan's robustness against errors produced by the transition identifier function.

Finally, **RQ3** is linked to experiment **5** namely testing whether R-latplan is robust against errors induced by the transition identifier function, see also Figure 3. As discussed in Section V, we check for **redundancy** among the learned actions, which leads us to the results in Table V showing, e.g. for Hanoi, that 72 groups of redundant actions were found and correspond to the 5% of transitions that were duplicated. These results thus support our claim that R-latplan is robust against errors induced by the transition identifier function.

VIII. Conclusions and Future Work

We proposed R-latplan, a system that learns reliable symbolic (PDDL) representations of an agent's actions from noisy and incomplete visual observations, and without explicit human expert knowledge. By rationalising Latplan's architecture [5] and replacing the generation of actions's labels by a *transition identifier function* output, it produces reliable action models where the actions can be understood as transitions in a visually-augmented planning DFA. More importantly, the resulting action models can be ensured to be associated with the actual capabilities of autonomous agents, whose observed behaviour has been used for learning. Indeed, even in the presence of errors produced by a transition identifier function, our experimental results — and in particular the result from **Exp5**, — suggest that we can recover the information connecting a desired (low level) PDDL action with ones that agents can indeed perform (high level action). This validates our claim of robustness and reliability of the overall learning procedure to produce results that are faithful to the agents' capabilities.

One limitation of our work is the number of low-level PDDL actions which grows with the size of the state, in turn increasing the processing time of FD. In future work, we want to post process the PDDL in order to merge low-level actions into high-level ones. We also intend to scale up to more complex domains involving continuous actions and time. This may require the use of a more advanced model, such as JEPA [18], for learning better representations. Lastly, the use of an auto-encoder - as opposed to most of the works presented in section II - opens the possibility of considering other types of inputs like unstructured text or audio data to perform language based/audio-based reasoning.

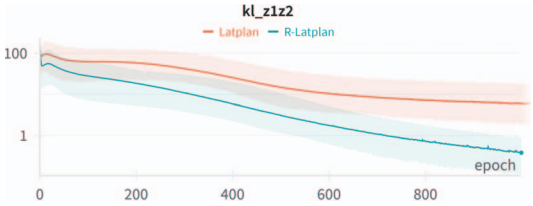

Fig. 8: Average error (log-scale) and standard deviation in predicting the successor state after applying an action

References

[1] A. O. Liberman, B. Bonet, and H. Geffner, "Learning first-order symbolic planning representations that are grounded," *arXiv preprint arXiv:2204.11902*, 2022.
[2] A. Ahmetoglu, M. Y. Seker, J. Piater, E. Oztop, and E. Ugur, "Deepsym: Deep symbol generation and rule learning for planning from unsupervised robot interaction," *Journal of Artificial Intelligence Research*, vol. 75, 2022.
[3] H. Harman and P. Simoens, "Generating symbolic action definitions from pairs of images: Applied to solving towers of hanoi." AAAI Press, 2020.
[4] K. Xi, S. Gould, and S. Thiébaux, "Neuro-symbolic learning of lifted action models from visual traces," in *34th International Conference on Automated Planning and Scheduling*, 2024.
[5] M. Asai, H. Kajino, A. Fukunaga, and C. Muise, "Classical planning in deep latent space," *Journal of Artificial Intelligence Research*, vol. 74, 2022.
[6] V. Lifschitz, "On the semantics of strips," in *Reasoning about Actions and Plans: Proceedings of the 1986 Workshop*, 1987.
[7] H. Geffner and B. Bonet, *Classical Planning: Full Information and Deterministic Actions*, 2013.
[8] D. Perrin, "Finite automata," in *Formal Models and Semantics*, 1990.
[9] G. De Giacomo and M. Y. Vardi, "Automata-theoretic approach to planning for temporally extended goals," in *Recent Advances in AI Planning: 5th European Conference on Planning*, 2000.
[10] D. P. Kingma, M. Welling *et al.*, "An introduction to variational autoencoders," *Foundations and Trends® in Machine Learning*, 2019.
[11] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.
[12] P. Dagum and M. Luby, "Approximating probabilistic inference in bayesian belief networks is np-hard," *Artificial intelligence*, vol. 60, 1993.
[13] J. M. Joyce, *Kullback-Leibler Divergence*, 2011.
[14] M. Helmert, "The fast downward planning system," *Journal of Artificial Intelligence Research*, 2006.
[15] T. Silver and R. Chitnis, "Pddlgym: Gym environments from pddl problems," in *International Conference on Automated Planning and Scheduling (ICAPS) PRL Workshop*, 2020.
[16] A. Hagberg, P. Swart, and D. Chult, "Exploring network structure, dynamics, and function using networkx," in *Proceedings of the 7th Python in Science Conference*, 2008.
[17] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, 1959.
[18] A. Bardes, Q. Garrido, J. Ponce, X. Chen, M. Rabbat, Y. LeCun, M. Assran, and N. Ballas, "Revisiting feature prediction for learning visual representations from video," *arXiv preprint arXiv:2404.08471*, 2024.