Classical Planning in Deep Latent Space: From Unlabeled Images to PDDL (and back)

Masataro Asai, Alex Fukunaga

guicho2.71828@gmail.com
Graduate School of Arts and Sciences
University of Tokyo

Abstract. Current domain-independent, classical planners require symbolic models of the problem domain and instance as input, resulting in a knowledge acquisition bottleneck. Meanwhile, although recent work in deep learning has achieved impressive results in many fields, the knowledge is encoded in a subsymbolic representation which cannot be directly used by symbolic systems such as planners. We propose LatPlan, an integrated architecture combining deep learning and a classical planner. Given a set of unlabeled training image pairs showing allowed actions in the problem domain, and a pair of images representing the start and goal states, LatPlan uses a Variational Autoencoder to generate a discrete latent vector from the images, based on which a PDDL model can be constructed and then solved by an off-the-shelf planner. We evaluate LatPlan using image-based versions of 3 planning domains: 8-puzzle, LightsOut, and Towers of Hanoi.

1 Introduction

Recent advances in domain-independent planning have greatly enhanced their capabilities. However, planning problems need to be provided to the planner in a structured, symbolic representation such as PDDL [22], and in general, such symbolic models need to be provided by a human, either directly in PDDL, or via a compiler which transforms some other symbolic problem representation into PDDL. This results in the *knowledge-acquisition bottleneck*, where the modeling step is sometimes the bottleneck in the problem solving cycle. In addition, the requirement for symbolic input poses a significant obstacle to applying planning in *new, unforeseen* situations where no human is available to create such a model, e.g., autonomous spacecraft exploration. This first requires generating symbols from raw sensor input, i.e., the *symbol grounding problem* [30].

Recently, significant advances have been made in neural network (NN) approaches for cognitive tasks including image classification [8], object recognition [26], speech recognition [9], machine translation as well as NN-based problem-solving systems [23, 10]. However, the current state-of-the-art in pure NN-based systems do not yet provide guarantees provided by symbolic planning systems, such as deterministic completeness and solution optimality.

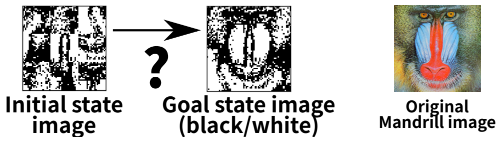

Fig. 1. An image-based 8-puzzle.

Using a NN-based perceptual system to *automatically* provide input models for domain-independent planners could greatly expand the applicability of planning technology and offer the benefits of both paradigms. *We consider the problem of robustly, automatically bridging the gap between such symbolic and subsymbolic representations.*

Fig. 1 (left) shows a scrambled, 3x3 tiled version of the the photograph on the right, i.e., an image-based instance of the 8-puzzle. We seek a domain-independent system which, given only a set of unlabeled images showing the valid moves for this image-based puzzle, finds an optimal solution to the puzzle. Although the 8-puzzle is trivial for symbolic planners, solving this image-based problem with a domain-independent system which *has no prior assumptions/knowledge* (e.g., "sliding objects", "tile arrangement", "a grid-like structure") is nontrivial. The only assumption allowed about the nature of the task is that it can be modeled and solved as a classical planning problem.

We propose Latent-space Planner (LatPlan), a hybrid architecture which uses NN-based image processing to completely automatically generate a propositional, symbolic problem representation which can be used as the input for a classical planner. LatPlan consists of 3 components: (1) a NN-based *State Autoencoder* (SAE), which provides a bidirectional mapping between the raw input of the world states and its symbolic/categorical representation, (2) an *action model generator* which generates a PDDL model using the symbolic representation acquired by the SAE, and (3) a symbolic planner. Given only a set of unlabeled images from the domain as input, we train (unsupervised) the SAE and use it to generate D, a PDDL representation of the image-based domain. Then, given a planning problem instance as a pair of initial and goal images such as Fig. 1, LatPlan uses the SAE to map the problem to a symbolic planning instance in D, and uses the planner to solve the problem.

2 LatPlan: System Architecture

This section describes the LatPlan architecture and the current implementation, LatPlanα. LatPlan works in 3 phases. In Phase 1 (symbol-grounding), a State AutoEncoder providing a bidirectional mapping between raw data (e.g., images) and symbols is learned (unsupervised) from a set of unlabeled images of representative states. In Phase 2 (action model generation), the operators available in the domain is generated from a set of pairs of unlabeled images, and a PDDL domain model is generated. In Phase 3 (planning), a planning problem instance

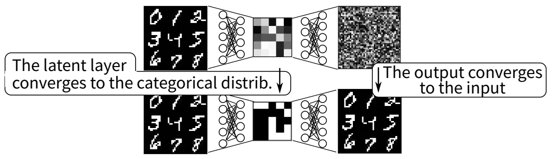
The latent layer converges to the categorical distrib.
The output converges to the input

Fig. 2. Step 1: Train the State Autoencoder by minimizing the sum of the reconstruction loss (binary cross-entropy) and the variational loss of Gumbel-Softmax.

is input as a pair of images (i, g) where i shows an *initial state* and g shows a *goal state*. These are converted to symbolic form using the SAE, and the problem is solved by the symbolic planner. For example, an 8-puzzle problem instance in our system consists of an image of the start (scrambled) configuration of the puzzle (i), and an image of the solved state (g). Finally, the symbolic, latent-space plan is converted to a human-comprehensible visualization of the plan.

Symbol Grounding with a State Autoencoder The State Autoencoder (SAE) provides a bidirectional mapping between images and a symbolic representation.

First, note that a direct 1-to-1 mapping between images and discrete objects can be trivially obtained simply by using the array of discretized pixel values as a "symbol". However, such a trivial SAE lacks the crucial properties of *generalization* – ability to encode/decode unforeseen world states to symbols – and *robustness* – two similar images that represent "the same world state" should map to the same symbolic representation. Thus, we need a mapping where the symbolic representation captures the "essence" of the image, not merely the raw pixel vector. The main technical contribution of this paper is the proposal of a SAE which is implemented as a Variational Autoencoder [16] with a Gumbel-Softmax (GS) activation function [14].

Gumbel-Softmax (GS) activation is a recently proposed reparametrization trick [14] for categorical distribution. Using GS in the network in place of standard activation functions (Sigmoid, Softmax, ReLU) forces the activation to converge to a discrete one-hot vector. GS has a "temperature" parameter τ which controls the magnitude of approximation. τ is annealed by a schedule $\tau \leftarrow \max(0.1, \exp(-rt))$ where t is the current training epoch and r is an annealing ratio [14]. We chose r so that $\tau = 0.1$ when the training finishes.

In our implementation, the SAE is comprised of multilayer perceptrons combined with Dropouts and Batch Normalization in both the encoder and the decoder networks, with a GS layer in between. The input to the GS layer is the flat, last layer of the encoder network. The output is an (N, M) matrix where N is the number of categorical variables and M is the number of categories.

Our key observation is that these categorical variables can be used directly as propositional symbols by a symbolic reasoning system, i.e., this provides a solution to the symbol grounding problem in our architecture. We specify $M = 2$, effectively obtaining N propositional state variables. It is possible to specify

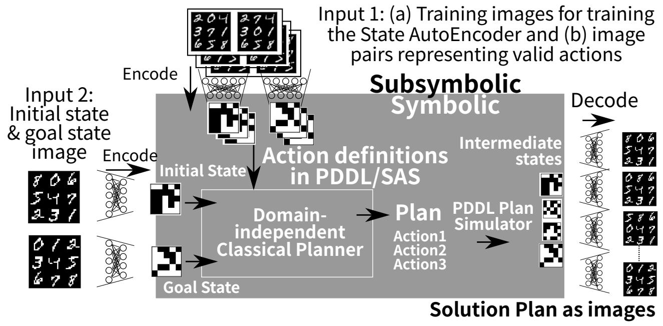
Input 1: (a) Training images for training the State AutoEncoder and (b) image pairs representing valid actions
Encode
Subsymbolic
Symbolic
Decode
Input 2: Initial state & goal state image
Encode
Initial State
Action definitions in PDDL/SAS
Intermediate states
Domain-independent Classical Planner
Plan
Action1
Action2
Action3
PDDL Plan Simulator
Goal State
Solution Plan as images

Fig. 3. Classical planning in latent space: We use the learned State AutoEncoder to convert pairs of images $(pre, post)$ first to symbolic ground actions and then to a PDDL domain. We also encode initial and goal state images into a symbolic ground actions and then a PDDL problem. A classical planner finds the symbolic solution plan. Finally, intermediate states in the plan are decoded back to a human-comprehensible image sequence.

different M for each variable and represent the world using multi-valued representation as in SAS+ [3] but we always use $M = 2$ for simplicity.

The trained SAE provides bidirectional mapping between the raw inputs (subsymbolic representation) to and from their symbolic representations:

- $b = Encode(r)$ maps an image r to a boolean vector b.
- $\tilde{r} = Decode(b)$ maps a boolean vector b to an image $\tilde{r}$.

$Encode(r)$ maps raw input r to a symbolic representation by feeding the raw input to the encoder network, extract the activation in the GS layer, and take the first row in the $N \times 2$ matrix, resulting in a binary vector of length N. Similarly, $Decode(b)$ maps a binary vector b back to an image by concatenating b and its complement $\bar{b}$ to obtain a $N \times 2$ matrix and feeding it to the decoder.

It is *not* sufficient to use traditional activation functions such as softmax and round the activation values to obtain discrete 0/1 values because we need to map the symbolic plan back to images. We need a decoding network trained for 0/1 values approximated by a smooth function, e.g., GS or similar approach such as [21]. A rounding-based scheme would be unable to restore the images from discrete values because the decoder is trained using continuous values. Also, the rounding operation cannot be part of a backpropagated network because rounding is non-differentiable.

An SAE trained on a small fraction of the possible states successfully generalizes so that it can *Encode* and *Decode* every possible state in that domain. In all our experiments below, we train the SAE using randomly selected images from the domain. For example, on the 8-puzzle, the SAE trained on 12000 randomly generated configurations out of 362880 possible configurations is used by the domain model generator to *Encode* every 8-puzzle state.

Domain Model Generation The model generator takes as input a trained

SAE, and a set R contains pairs of raw images. In each image pair $(pre_i, post_i) \in R$, pre_i and $post_i$ are images representing the state of the world before and after some action a_i is executed, respectively. In each ground action image pair, the "action" is implied by the difference between pre_i and $post_i$. The output of the model generator is a PDDL domain file for a grounded unit-cost STRIPS planning problem. For each $(pre_i,\ post_i) \in R$ we apply the learned SAE to pre_i and $post_i$ to obtain $(Encode(pre_i),\ Encode(post_i))$, the symbolic representations (latent space vectors) of the state before and after action a_i is executed. This results in a set of symbolic ground action instances A.

Ideally, a model generation component would induce a complete action model from a limited set of symbolic ground action instances. However, *action model learning* from a limited set of action instances is a nontrivial area of active research [7, 11, 18, 24, 32, 6]. Since the focus of this paper is on the overall LatPlan architecture and the SAE, we leave model induction for future work. Instead, the current implementation LatPlanα uses a trivial, baseline strategy which generates a model based on *all* ground actions, which are supposed to be easily replaced by existing off-the-shelf action model learner. In this baseline method, R contains image pairs representing all ground actions that are possible in this domain, so $A = \{Encode(r) | r \in R\}$ contains all symbolic ground actions possible in the domain. In Sec. 5, we further discuss the implication and the impact of this model. In the experiments (Sec. 3), we generate image pairs for all ground actions using an external image generator. It is important to note that while R contains all possible actions, R is not used for training the SAE. As explained before, the SAE is trained using at most 12000 images while the entire state space is much larger.

LatPlanα compiles A directly into a PDDL model as follows. For each action $(Encode(pre_i),\ Encode(post_i)) \in A$, each bit $b_j (1 \leq j \leq N)$ in these boolean vectors is mapped to propositions `(b`$_j$`-true)` and `(b`$_j$`-false)` when the encoded value is 1 and 0 (resp.). $Encode(pre_i)$ is directly used as the preconditions of action a_i. The add/delete effects of action i are computed by taking the bitwise difference between $Encode(pre_i)$ and $Encode(post_i)$. For example, when b_j changes from 1 to 0, it compiles into `(and (b`$_j$`-false) (not (b`$_j$`-true)))`. The initial and the goal states are similarly created by applying the SAE to the initial and goal images.

Planning with an Off-the-Shelf Planner The PDDL instance generated in the previous step can be solved by an off-the-shelf planner. LatPlanα uses the Fast Downward planner [12]. However, on the models generated by LatPlanα, the invariant detection routines in the Fast Downward PDDL-SAS converter became a bottleneck, so we wrote a trivial, replacement PDDL-SAS converter without the invariant detection.

LatPlan inherits all of the search-related properties of the planner which is used. For example, if the planner is complete and optimal, LatPlan will find an optimal plan for the given problem (if one exists), with respect to the portion of the state-space graph captured by the acquired model. Domain-independent heuristics developed in the planning literature are designed to exploit structure in the domain model. Although the structure in models acquired by LatPlan may not directly correspond to those in hand-coded models, intuitively, there should be some exploitable structure. The search results in Sec. 3 suggest that the domain-independent heuristics can reduce the search effort.

Visualizing/Executing the Plans Since the actions comprising the plan are SAE-generated latent bit vectors, the "meaning" of each symbol (and thus the plan) is not necessarily clear to a human observer. However, we can obtain a step-by-step visualization of the world (images) as the plan is executed (e.g. Fig. 4) by starting with the latent state representation of the initial state, applying (simulating) actions step-by-step (according to the PDDL model acquired above) and *Decode*'ing the latent bit vectors for each intermediate state to images using the SAE. In this paper, a "mental image" of the solution (i.e., the image sequence visualization) is sufficient. In a less simplified setting, mapping the actions found by LatPlan (transitions between latent bit vector pairs) to lower-level actuation would be necessary (future work).

3 Experimental Evaluation

All of the SAE networks used in the evaluation have the same network topology except the input layer which should fit the size of the input images. The network consists of the following layers: [Input, GaussianNoise(0.1), fc(4000), relu, bn, dropout(0.4), fc(4000), relu, bn, dropout(0.4), fc(49x2), GumbelSoftmax, dropout(0.4), fc(4000), relu, bn, dropout(0.4), fc(4000), relu, bn, dropout(0.4), fc(*input*), sigmoid]. Here, fc = fully connected layer, bn = Batch Normalization, and tensors are reshaped accordingly. The last layers can be replaced with [fc($input \times 2$), GumbelSoftmax, TakeFirstRow] for better reconstruction when we can assume that the input image is binarized. The network is trained using Adam optimizer (lr:0.001) for 1000 epochs.

The latent layer has 49 bits, which sufficiently covers the total number of states in any of the problems that are used in the following experiments. This could be reduced for each domain (made more compact) with further engineering.

MNIST 8-puzzle This is an image-based version of the 8-puzzle, where tiles contain hand-written digits (0-9) from the MNIST database [20]. Each digit is shrunk to 14x14 pixels, so each state of the puzzle is a 42x42 image. Valid moves in this domain swap the "0" tile with a neighboring tile, i.e., the "0" serves as the "blank" tile in the classic 8-puzzle. The entire state space consists of 362880 states (9!). Note that the same image is used for each digit in all states, e.g., the "1" digit is the same image in all states.

Out of 362880 images, 12000 randomly selected images are used for training the SAE. This set is further divided into a training set (11000) and a validation set (1000). Training takes 40 minutes/1000 epochs on a NVIDIA GTX-1070.

Scrambled Photograph 8-puzzle The above MNIST 8-puzzle described above consists of images where each digit is cleanly separated from the black region. To show that LatPlan does not rely on cleanly separated objects, we solve 8-puzzles generated by cutting and scrambling real photographs (similar to

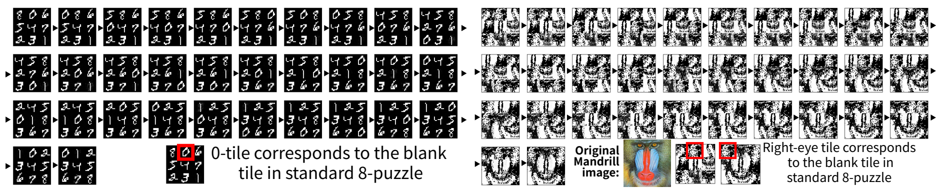
0-tile corresponds to the blank tile in standard 8-puzzle
Original Mandrill image:
Right-eye tile corresponds to the blank tile in standard 8-puzzle

Fig. 4. (Left) Output of solving the MNIST 8-puzzle instance with the longest (31 steps) optimal plan. [Reinefeld 1993] (Right) Output of solving a photograph-based 8-puzzle (Mandrill). We emphasize that LatPlan has no built-in notion of "sliding object", or "tile arrangement"; furthermore, the SAE is being trained completely from scratch when LatPlan is applied to this scrambled photograph puzzle – there is no transfer/reuse of knowledge from the SAE learned for the MNIST 8-puzzle.

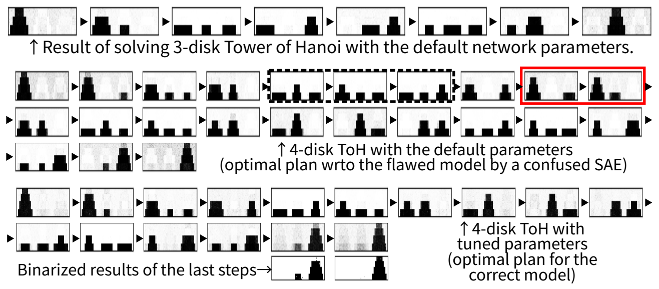
↑ Result of solving 3-disk Tower of Hanoi with the default network parameters.
↑ 4-disk ToH with the default parameters (optimal plan wrto the flawed model by a confused SAE)
↑ 4-disk ToH with tuned parameters (optimal plan for the correct model)
Binarized results of the last steps→

Fig. 5. Output of solving ToH with 3 and 4 disks. The third picture is the result of SAE with different parameters.

sliding tile puzzle toys sold in stores). We used the "Mandrill" image, a standard benchmark in the image processing literature. The image was first converted to greyscale and then rounded to black/white (0/1) values. The same number of images as in the MNIST-8puzzle experiments are used.

Towers of Hanoi (ToH) Disks of various sizes must be moved from one peg to another, with the constraint that a larger disk can never be placed on top of a smaller disk. Due to the smaller number of states (3^d states for d disks), we used images of all states as the set of images for training SAE. This is further divided into the training set (90%) and the validation set (10%), and we verified that the network has learned a generalized model without overfitting.

3-disk ToH is solved successfully and optimally using the default hyperparameters (Fig. 5, top). However, on 4-disks, the SAE trained with the default hyperparameters (Fig. 5, middle) is confused, resulting in a flawed model which causes the planner to choose suboptimal moves (dashed box). Sometimes, the size/existence of disks is confused (red box). Tuning the hyperparameters to reduce the SAE loss corrects this problem. After increasing the training epochs (10000) and tuning the network shape (fc(6000), $N = 29$), the SAE generated a correct model, resulting in the optimal 15-step plan (Fig. 5, bottom).

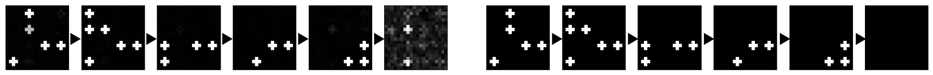

Fig. 6. Output of solving 4x4 LightsOut (left) and its binarized result (right). Although the goal state shows two blurred switches, they have low values (around 0.3) and disappear after rounding.

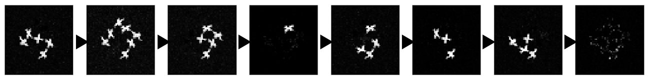

Fig. 7. Output of solving 3x3 Twisted LightsOut.

LightsOut A video game where a grid of lights is in some on/off configuration (+: On), and pressing a light toggles its state (On/Off) as well as the state of all of its neighbors. The goal is all lights Off. Unlike the 8-puzzle where each move affects only two adjacent tiles, a single operator in 4x4 LightsOut can simultaneously flip 5/16 locations. Also, unlike 8-puzzle and ToH, the LightsOut game allows some "objects" (lights) to disappear. This demonstrates that LatPlan is not limited to domains with highly local effects and static objects.

Twisted LightsOut In all of the above domains, the "objects" correspond to rectangles. To show that LatPlan does not rely on rectangular regions, we demonstrate its result on "Twisted LightsOut", a distorted version of the game where the original LightsOut image is twisted around the center. Unlike previous domains, the input images are not binarized.

Robustness to Noisy Input We show the robustness of the system against the input noise. We corrupted the initial/goal state inputs by adding Gaussian or salt noise, as shown in Fig. 8. The system is robust enough to successfully solve the problem, because our SAE is a Denoising Autoencoder [31] which has an internal *GaussianNoise layer* which adds a Gaussian noise to the inputs (only during training) and learn to reconstruct the original image from a corrupted version of the image.

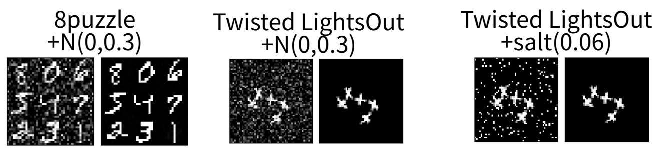

Fig. 8. SAE robustness vs noise: Corrupted initial state image r and its reconstruction $Decode(Encode(r))$ by SAE on MNIST 8-puzzle and Twisted LightsOut. Images are corrupted by Gaussian noise of σ up to 0.3 for both problems, and by salt noise up to $p = 0.06$ for Twisted LightsOut. LatPlanα successfully solved the problems. The SAE maps the noisy image to the correct symbolic vector $b = Encode(r)$, conduct planning, then map b back to the de-noised image $Decode(b)$.

Are Domain-Independent Heuristics Effective in Latent Space? We compare the numbers of nodes expanded by a search using a greedy merging PDB [28] and blind heuristics (i.e., breadth-first search) in Fast Downward:

- MNIST 8-puzzle (6 instances, mean(StdDev)): Blind 176658(25226), PDB **77811**(32978)
- Mandrill 8-puzzle (1 instance with 31-step optimal solution, corresponding to the 8-puzzle instance [25]): Blind 335378, PDB **88851**
- ToH (4 disks, 1 instance): Blind 55, PDB **17**,
- 4x4 LightsOut (1 instance): Blind 952, PDB **27**,
- 3x3 Twisted LightsOut (1 instance): Blind 522, PDB **214**

The domain-independent PDB heuristic significantly reduced node expansions. Search times ($<$ 3 seconds for all instances) were also faster for all instances with the PDB. Although total runtimes including heuristic initialization is slightly slower than blind search, in domains where goal states and operators are the same for all instances (e.g., 8-puzzle) PDBs can be reused [19], and PDB generation time can be amortized across many instances. Although these results show that existing heuristics for classical planning are able to reduce search effort compared to blind search, much more work is required in order to understand how the features in latent space interact with existing heuristics.

4 Related Work

[18] propose a method for generating PDDL from a low-level, sensor actuator space of an agent characterized as a semi-MDP. The inputs to their system are 33 variables representing accurate structured input (e.g., x/y distances) or categorical states (the on/off state of a button etc.) while LatPlan takes noisy unstructured images (e.g., for 8-puzzle, 42x42=1764-dimensional arrays).

Compared to learning from observation (LfO) in the robotics literature [2], (1) LatPlan is trained based on image pairs showing individual actions, not plan executions (sequence of actions); (2) LatPlan focuses on PDDL for high-level (puzzle-like) tasks, not on motion planning tasks. This significantly affects the data collection scheme: While LfO has *action segmentation* issue because it does not know when an action starts/ends in the plan traces (e.g. video clip), LatPlan does not, because it assumes that a robot can explore the world by itself, initiating/terminating its own action and taking pictures by a camera. The robot can perform a random walk under physical constraints and supervision, which ensure the legal moves (e.g., the physical tile in 8-puzzle). If we further assume that it can “reset” the world (e.g., into a random configuration), then, the robot could eventually obtain images of the entire state space.

A closely related line of work in LfO is learning of board game play from videos [4, 15, 17]. Unlike LatPlan, these works make relatively strong assumptions about the environment, e.g., that there is a grid-like environment.

There is a large body of previous work using neural networks to directly solve combinatorial tasks, such as TSP [13] or Tower of Hanoi [5]. Although they use

NNs to solve search problems, they assume a fully symbolic representation of the problem as input. Other line of hybrid systems embed NNs *inside* a search algorithm to provide search control knowledge [29, 1, 27]. In contrast, we use a NN-based SAE for symbol grounding, not for search control.

Deep Reinforcement Learning (DRL) has solved complex image-based problems [23]. For unit-action-cost planning, LatPlan does not require a reinforcement signal (reward function). Also, it can provide guarantees of completeness and solution cost optimality.

5 Discussion and Conclusion

We proposed LatPlan, an integrated architecture for planning which, given only a set of unlabeled images and no prior knowledge, generates a classical planning problem model, solves it with a symbolic planner, and presents the resulting plan as a human-comprehensible sequence of images. We demonstrated its feasibility using image-based versions of planning/state-space-search problems (8-puzzle, Towers of Hanoi, Lights Out). *The key technical contribution is the SAE, which leverages the Gumbel-Softmax reparametrization technique [14] and learns (unsupervised) a bidirectional mapping between raw images and a propositional representation usable by symbolic planners.* Aside from the key assumptions about the deterministic environment and the sufficient training images, we avoid assumptions about the input domain. Thus, we have shown that domains with different characteristics can all be solved by the same system. In other words, *LatPlan is a domain-independent, image-based classical planner.*

To our knowledge, LatPlan is the first completely automated system of the kind. However, as a proof-of-concept, it has significant limitations to be addressed in future work. In particular, the domain model generator in LatPlanα *does not perform action model learning* from a small set of sample actions because the focus of this paper is not on action learning. Thus the current generator requires the entire set of latent states, transitions and in turn images. While this is obviously impractical, *this is **not** a fundamental limitation of the LatPlan **architecture***. The primitive generator is merely a placeholder for investigating the overall feasibility of an SAE-based end-to-end planning system (our major contribution) and is supposed to be easily replaced by the more sophisticated ones [7, 18, 24, 32]. To our knowledge, all previous domain learning methods require the structured (e.g., propositional) representations of states.

A related topic is how to specify a partial goal specification for LatPlan as in IPC domains (e.g. “having tiles 0,1,2 in the correct places is the goal” in a 8-puzzle), rather than assuming a single goal state, is an interesting future work.

Finally, we do *not* claim that the specific implementation of SAE in this paper works robustly on all images. Making a robust autoencoder is *not* a problem unique to LatPlan, but rather, a fundamental problem in deep learning. *Out contribution is the demonstration that it is possible to leverage some existing deep learning techniques quite effectively in an planning system*, and future work will continue leveraging further improvements in image processing techniques.

References

1. Arfaee, S.J., Zilles, S., Holte, R.C.: Learning Heuristic Functions for Large State Spaces. Artificial Intelligence 175(16-17), 2075–2098 (2011), http://dx.doi.org/10.1016/j.artint.2011.08.001;http://dblp.uni-trier.de/rec/bib/journals/ai/ArfaeeZH11
2. Argall, B., Chernova, S., Veloso, M.M., Browning, B.: A Survey of Robot Learning from Demonstration. Robotics and Autonomous Systems 57(5), 469–483 (2009), http://dx.doi.org/10.1016/j.robot.2008.10.024
3. Bäckström, C., Nebel, B.: Complexity Results for SAS+ Planning. Computational Intelligence 11(4), 625–655 (1995)
4. Barbu, A., Narayanaswamy, S., Siskind, J.M.: Learning Physically-Instantiated Game Play through Visual Observation. In: ICRA. pp. 1879–1886 (2010), http://dx.doi.org/10.1109/ROBOT.2010.5509925
5. Bieszczad, A., Kuchar, S.: Neurosolver Learning to Solve Towers of Hanoi Puzzles. In: IJCCI. vol. 3, pp. 28–38 (2015)
6. Celorrio, S.J., de la Rosa, T., Fernández, S., Fernández, F., Borrajo, D.: A Review of Machine Learning for Automated Planning. Knowledge Eng. Review 27(4), 433–467 (2012), http://dx.doi.org/10.1017/S026988891200001X
7. Cresswell, S., McCluskey, T.L., West, M.M.: Acquiring planning domain models using *LOCM*. Knowledge Eng. Review 28(2), 195–213 (2013)
8. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A Large-Scale Hierarchical Image Database. In: CVPR. pp. 248–255. IEEE (2009)
9. Deng, L., Hinton, G., Kingsbury, B.: New Types of Deep Neural Network Learning for Speech Recognition and Related Applications: An Overview. In: ICASSP. pp. 8599–8603. IEEE (2013)
10. Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwińska, A., Colmenarejo, S.G., Grefenstette, E., Ramalho, T., Agapiou, J., et al.: Hybrid Computing using a Neural Network with Dynamic External Memory. Nature 538(7626), 471–476 (2016)
11. Gregory, P., Cresswell, S.: Domain model acquisition in the presence of static relations in the LOP system. In: ICAPS. pp. 97–105 (2015), http://www.aaai.org/ocs/index.php/ICAPS/ICAPS15/paper/view/10621
12. Helmert, M.: The Fast Downward Planning System. J. Artif. Intell. Res.(JAIR) 26, 191–246 (2006), http://www.aaai.org/Papers/JAIR/Vol26/JAIR-2606.pdf
13. Hopfield, J.J., Tank, D.W.: ”Neural” Computation of Decisions in Optimization Problems. Biological cybernetics 52(3), 141–152 (1985)
14. Jang, E., Gu, S., Poole, B.: Categorical Reparameterization with Gumbel-Softmax. In: ICLR (2017)
15. Kaiser, L.: Learning Games from Videos Guided by Descriptive Complexity. In: AAAI (2012), http://www.aaai.org/ocs/index.php/AAAI/AAAI12/paper/view/5091;http://dblp.uni-trier.de/rec/bib/conf/aaai/Kaiser12
16. Kingma, D.P., Mohamed, S., Rezende, D.J., Welling, M.: Semi-supervised learning with deep generative models. In: NIPS. pp. 3581–3589 (2014)
17. Kirk, J.R., Laird, J.E.: Learning General and Efficient Representations of Novel Games Through Interactive Instruction. Advances in Cognitive Systems 4 (2016)
18. Konidaris, G., Kaelbling, L.P., Lozano-Pérez, T.: Constructing Symbolic Representations for High-Level Planning. In: AAAI. pp. 1932–1938 (2014), http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8424

19. Korf, R.E., Felner, A.: Disjoint Pattern Database Heuristics. Artificial Intelligence 134(1-2), 9–22 (2002), http://dx.doi.org/10.1016/S0004-3702(01)00092-3;http://dblp.uni-trier.de/rec/bib/journals/ai/KorfF02
20. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-Based Learning Applied to Document Recognition. Proc. of the IEEE 86(11), 2278–2324 (1998)
21. Maddison, C.J., Mnih, A., Teh, Y.W.: The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In: ICLR (2017)
22. McDermott, D.V.: The 1998 AI Planning Systems Competition. AI Magazine 21(2), 35–55 (2000), http://www.aaai.org/ojs/index.php/aimagazine/article/view/1506
23. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-Level Control through Deep Reinforcement Learning. Nature 518(7540), 529–533 (2015)
24. Mourão, K., Zettlemoyer, L.S., Petrick, R.P.A., Steedman, M.: Learning STRIPS Operators from Noisy and Incomplete Observations. In: UAI. pp. 614–623 (2012), https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=2322&proceeding_id=28
25. Reinefeld, A.: Complete Solution of the Eight-Puzzle and the Benefit of Node Ordering in IDA. In: IJCAI. pp. 248–253 (1993), http://ijcai.org/Proceedings/93-1/Papers/035.pdf
26. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards Real-time Object Detection with Region Proposal Networks. In: NIPS. pp. 91–99 (2015)
27. Satzger, B., Kramer, O.: Goal Distance Estimation for Automated Planning using Neural Networks and Support Vector Machines. Natural Computing 12(1), 87–100 (2013), http://dx.doi.org/10.1007/s11047-012-9332-y
28. Sievers, S., Ortlieb, M., Helmert, M.: Efficient Implementation of Pattern Database Heuristics for Classical Planning. In: SOCS (2012)
29. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al.: Mastering the Game of Go with Deep Neural Networks and Tree Search. Nature 529(7587), 484–489 (2016)
30. Steels, L.: The Symbol Grounding Problem has been solved. So what's next? In: de Vega, M., Glenberg, A., Graesser, A. (eds.) Symbols and Embodiment. Oxford University Press (2008)
31. Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.A.: Extracting and Composing Robust Features with Denoising Autoencoders. In: ICML. pp. 1096–1103. ACM (2008)
32. Yang, Q., Wu, K., Jiang, Y.: Learning Action Models from Plan Examples using Weighted MAX-SAT. Artificial Intelligence 171(2-3), 107–143 (2007), http://dx.doi.org/10.1016/j.artint.2006.11.005;http://dblp.uni-trier.de/rec/bib/journals/ai/YangWJ07