1A2-OS-05b-2

Deep Latent Space における汎用プランニング: 記号を取り戻せ!!

浅井 政太郎　　福永 Alex

Graduate School of Arts and Sciences, The University of Tokyo

現在のドメイン非依存自動行動計画ソルバ (プランナ) は、入力として問題ドメインとインスタンスの記号モデルを必要とし、実世界を手作業で記号にモデル化するボトルネックが発生する。 一方、近年のディープ・ラーニングの研究は多くの分野で目覚ましい成果をあげてきたが、その知識は記号的システムでは直接使用できないサブシンボリック表現で保持されている。 本論文は深層学習と古典的プランナを組み合わせた統合アーキテクチャLatPlan を提案する。 問題ドメインで許可されたアクションを示す多数のラベル無し訓練画像ペア、および開始状態と目標状態を表す一対の画像を入力に、LatPlan は Variational Autoencoder を使用して問題の記号的表現ベクトルを生成する。 生成した問題は既存の記号的ソルバによって解かれ、その解は再びプランを表す画像にデコードされる。 本論文は LatPlan を、3 つの計画ドメイン（8-puzzle、LightsOut、Towers of Hanoi）の画像バージョンを用いて評価する。

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 [McDermott 00], and in general, such symbolic models need to be provided by a human, either directly in a modeling language such as 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 or a generator, e.g., autonomous spacecraft exploration. In particular this first requires generating symbols from raw sensor input, i.e., the *symbol grounding problem* [Steels 08].

Recently, significant advances have been made in neural network (NN) deep learning approaches for perceptually-based cognitive tasks including image classification [Deng 09] and object recognition [Ren 15], as well as NN-based problem-solving systems for problem solving [Mnih 15, Graves 16]. 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.

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 subsymbolic representations and the symbolic representations required by domain-independent planners*.

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, *without prior assumptions/knowledge* e.g., “sliding objects”, “tile arrangement”.

We propose Latent-space Planner (LatPlan), an integrated architecture which uses NN-based image processing to completely automatically generate a propositional, symbolic problem representation.

Figure 1: An image-based 8-puzzle.

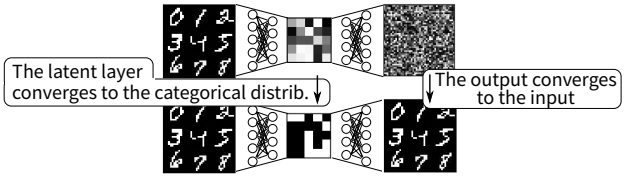


Figure 2: Step 1: Train the State Autoencoder by minimizing the sum of the reconstruction loss (binary cross-entropy between the input and the output) and the variational loss of Gumbel-Softmax (KL divergence between the actual latent distribution and the random categorical distribution as the target).

2. LatPlan: System Architecture

This section describe the LatPlan architecture and the current implementation, LatPlanα. LatPlan works in 3 phases. In Phase 1 (symbol-grounding, Sec. 2.1), a State AutoEncoder (SAE) providing a bidirectional mapping between raw data (e.g., images)[*1] and symbols is learned (unsupervised) from a set of unlabeled images of representative states. In Phase 2 (action model generation, Sec. 2.2), 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, Sec. 2.3), a planning problem instance 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 sequence of human-comprehensible images visualizing the plan (Sec. 2.4).

∗1 Although the LatPlan architecture can, in principle, be applied to various unstructured data input including images, texts or low-level sensors, in the rest of the paper we refer to “images” for simplicity and also because the current implementation is image-based.

2.1 Symbol Grounding with a State Autoencoder

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

An AutoEncoder (AE) is a type of Feed-Forward Network (FFN) that uses unsupervised learning to produce an image that matches the input [Hinton 06]. The intermediate layer is said to have a *Latent Representation* of the input and is considered to be performing data compression. AEs are commonly used for pretraining a neural network. A Variational AutoEncoder (VAE) [Kingma 13] is a type of AE that forces the *latent layer* (the most compressed layer in the AE) to follow a certain distribution (e.g., Gaussian) for given input images. Since the target random distribution prevents backpropagating the gradient, most VAE implementations use *reparametrization tricks*, which decompose the target distribution into a differentiable distribution and a purely random distribution that does not require the gradient. For example, the Gaussian distribution $N(\sigma, \mu)$ can be decomposed into $\mu + \sigma N(1, 0)$.

Gumbel-Softmax (GS) reparametrization is a technique for enforcing a categorical distribution on a particular layer of the neural network. [Jang 17]. A "temperature" parameter τ, which controls the magnitude of approximation to the categorical distribution, is decreased by an annealing schedule $\tau \leftarrow \max(0.1, \exp(-rt))$ where t is the current training epoch and r is an annealing ratio. Using a GS layer in the network forces the layer to converge to a discrete one-hot vector when the temperature approaches near 0.

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 obtain the propositional representation by specifying $M = 2$, effectively obtaining N propositional state variables. 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 network.

It is *not* sufficient to simply use traditional activation functions such as sigmoid or softmax and round the continuous activation values in the latent layer to obtain discrete 0/1 values. As explained in Sec. 2.4, we need to map the symbolic plan back to images, so we need a decoding network trained for 0/1 values approximated by a smooth function, e.g., GS or similar approach such as [Maddison 17]. A rounding-based scheme would be unable to restore the images from the latent layer because the decoder network

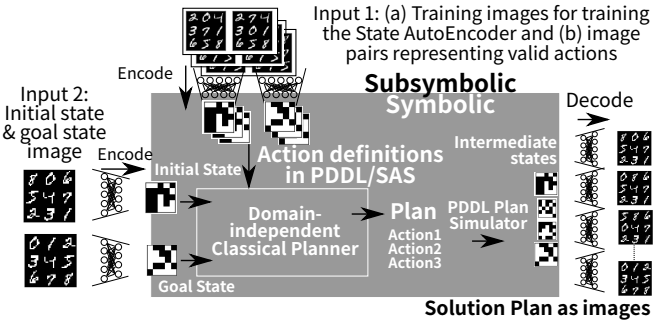


Figure 3: We use the learned State AutoEncoder (Sec. 2.1) to convert pairs of images $(pre, post)$ first to symbolic ground actions and then to a PDDL domain (Sec. 2.2) We also encode initial and goal state images into a symbolic ground actions and then a PDDL problem. A classical planner finds the symbolic solution plan. Finally, intermediate states in the plan are decoded back to a human-comprehensible image sequence.

is trained using continuous activation values. Also, representing the rounding operation as a layer of the network is infeasible because rounding is non-differentiable, precluding backpropagation-based training of the network.

In some domains, 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, on each domain, 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 (Sec. 2.2) to $Encode$ every 8-puzzle state.

2.2 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 [Konidaris 14, Mourão 12, Yang 07, Celorrio 12]. Instead, the current implementation LatPlanα generates a model based on *all* ground actions, i.e., R contains image pairs representing all ground actions that are possible in this domain, so A (generated by applying the SAE to all elements of R) contains all symbolic ground actions possible in the domain. 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 possi-

ble actions, R is not used for training the SAE. As explained in Sec. 2.1, 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.

2.3 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 [Helmert 06]. However, on the models generated by LatPlanα, the invariant detection routines in the Fast Downward PDDL to SAS translator (translate.py) became a bottleneck, so we wrote a trivial, replacement PDDL to 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 plan (if one exists), with respect to the portion of the state-space graph captured by the acquired model.

2.4 Visualizing 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 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.

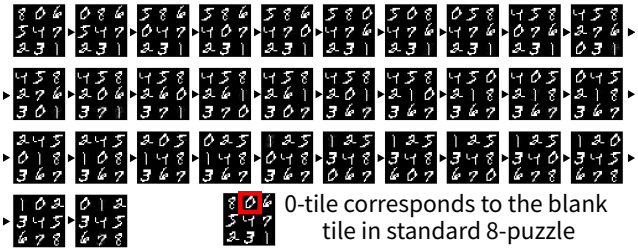
3. Experimental Evaluation

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. They are implemented using TensorFlow and Keras and consist 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 to minimize the sum of the variational loss and the reconstruction loss (binary cross-entropy) using Adam optimizer (lr:0.001) for 1000 epochs.

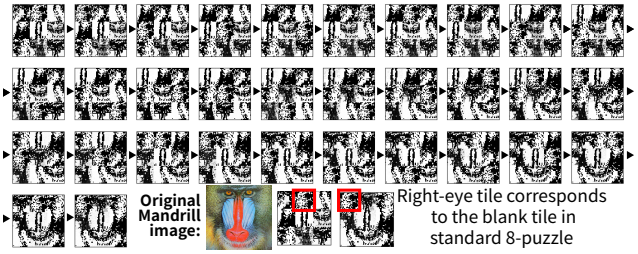
The latent layer has 49 bits, which sufficiently covers the total number of states in any of the problems that are used in the following experiments. This could be reduced for each domain (made more compact) with further engineering.

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 [LeCun 98]. Each digit is shrunk to 14x14 pixels, so each state of the puzzle is a 42x42 image. Training takes about 40 minutes with 1000 epochs on a single NVIDIA GTX-1070.

Figure 4: 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 in the binarized image.



Scrambled Photograph 8-puzzle To show that LatPlan does not rely on cleanly separated objects, we solve 8-puzzles generated by cutting and scrambling real photographs (similar to sliding tile puzzle toys sold in stores).

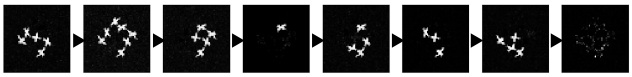


Towers of Hanoi (ToH) Disks of various sizes must be moved from one peg to another, with the constraint that a larger disk can never be placed on top of a smaller disk. We generated the training and planning inputs for this task, with 3 and 4 disks. Each input image has a dimension of 24×122 and 32×146 (resp.). Due to the smaller number of states, we used images of all states as the training input.

↑ Result of solving 3-disk Tower of Hanoi with the default network parameters.

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 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 sys-

tem against the input noise. We corrupted the initial/goal state inputs by adding Gaussian or salt noise, as shown in Fig. 5. The system is robust enough to successfully solve the problem, because our SAE is a Denoising Autoencoder [Vincent 08].

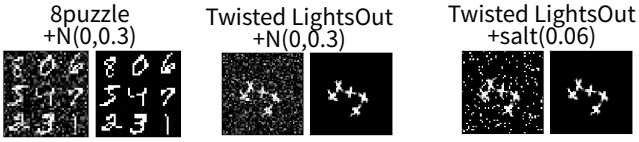


Figure 5: SAE robustness vs noise: Corrupted initial state image r and its reconstruction $Decode(Encode(r))$. Images are corrupted by Gaussian noise of σ up to 0.3 for both problems, and by salt noise up to $p = 0.06$ for Twisted LightsOut.

4. Discussion and Conclusion

We proposed LatPlan, an integrated architecture for domain model acquisition and 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 benchmark problems (8-puzzle, Towers of Hanoi, Lights Out). *The key technical contribution is the SAE, which leverages the Gumbel-Softmax reparametrization technique [Jang 17] and learns (unsupervised) a bidirectional mapping between raw images and a propositional representation usable by symbolic planners*. For example, as shown in Sec. 3., on the MNIST 8-puzzle, the "gist" of 42x42 training images are compressed into 49-bit representations that capture the essence of the images which is robust to noise.

Aside from the key assumptions that (1) the domain can be modeled and solved as a classical planning problem, and (2) the domain can be correctly inferred from the given training images, we avoid assumptions about the input domain. Thus, we have shown that domains with significantly different characteristics can all be solved by the same system without further engineering. However, as a proof-of-concept first implementation, it has significant limitations to be addressed in future work.

For example, the current LatPlanα domain model generator doesn't perform action model learning/induction from a small set of sample actions, and essentially constructs an explicit state space graph based on action image pairs for all ground actions in the domain. Incorporating action model learning [Konidaris 14, Mourão 12, Yang 07] is an important direction for future work.

Although we showed that LatPlanα works on several kinds of images, we do *not* claim that the SAE works robustly on all images. Making a truly robust autoencoder is *not* a problem unique to LatPlan, but rather, a fundamental problem in deep learning. *A contribution of this paper is the demonstration that it is possible to leverage some existing deep learning techniques quite effectively in an integrated learning/planning system*, and future work will seek to continue leveraging further improvements.

References

[Celorrio 12] Celorrio, S. J., et al.: A Review of Machine Learning for Automated Planning, *Knowledge Eng. Review*, Vol. 27, No. 4, pp. 433–467 (2012)

[Deng 09] Deng, J., et al.: ImageNet: A Large-Scale Hierarchical Image Database, in *CVPR*, pp. 248–255IEEE (2009)

[Graves 16] Graves, A., et al.: Hybrid Computing using a Neural Network with Dynamic External Memory, *Nature*, Vol. 538, No. 7626, pp. 471–476 (2016)

[Helmert 06] Helmert, M.: The Fast Downward Planning System, *J. Artif. Intell. Res.(JAIR)*, Vol. 26, pp. 191–246 (2006)

[Hinton 06] Hinton, G. E. and Salakhutdinov, R. R.: Reducing the Dimensionality of Data with Neural Networks, *Science*, Vol. 313, No. 5786, pp. 504–507 (2006)

[Jang 17] Jang, E., Gu, S., and Poole, B.: Categorical Reparameterization with Gumbel-Softmax, *ICLR* (2017)

[Kingma 13] Kingma, D. P. and Welling, M.: Auto-Encoding Variational Bayes, *ICLR* (2013)

[Konidaris 14] Konidaris, G., Kaelbling, L. P., and Lozano-Pérez, T.: Constructing Symbolic Representations for High-Level Planning, in *AAAI*, pp. 1932–1938 (2014)

[LeCun 98] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P.: Gradient-Based Learning Applied to Document Recognition, *Proc. of the IEEE*, Vol. 86, No. 11, pp. 2278–2324 (1998)

[Maddison 17] Maddison, C. J., Mnih, A., and Teh, Y. W.: The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables, *ICLR* (2017)

[McDermott 00] McDermott, D. V.: The 1998 AI Planning Systems Competition, *AI Magazine*, Vol. 21, No. 2, pp. 35–55 (2000)

[Mnih 15] Mnih, V., et al.: Human-Level Control through Deep Reinforcement Learning, *Nature*, Vol. 518, No. 7540, pp. 529–533 (2015)

[Mourão 12] Mourão, K., Zettlemoyer, L. S., Petrick, R. P. A., and Steedman, M.: Learning STRIPS Operators from Noisy and Incomplete Observations, in *UAI*, pp. 614–623 (2012)

[Ren 15] Ren, S., He, K., Girshick, R., and Sun, J.: Faster R-CNN: Towards Real-time Object Detection with Region Proposal Networks, in *NIPS*, pp. 91–99 (2015)

[Steels 08] Steels, L.: The Symbol Grounding Problem has been solved. So what's next?, in Vega, de M., Glenberg, A., and Graesser, A. eds., *Symbols and Embodiment*, Oxford University Press (2008)

[Vincent 08] Vincent, P., et al.: Extracting and Composing Robust Features with Denoising Autoencoders, in *ICML*, pp. 1096–1103ACM (2008)

[Yang 07] Yang, Q., et al.: Learning Action Models from Plan Examples using Weighted MAX-SAT, *Artificial Intelligence*, Vol. 171, No. 2-3, pp. 107–143 (2007)