Classical Planning in Deep Latent Space: Bridging the Subsymbolic-Symbolic Boundary

Masataro Asai
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 [McDermott, 2000], 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, 2008].

Recently, significant advances have been made in neural network (NN) deep learning approaches for perceptually-based cognitive tasks including image classification [Deng et al., 2009], object recognition [Ren et al., 2015], speech recognition [Deng et al., 2013], machine translation as well as NN-based problem-solving systems for problem solving [Mnih et al., 2015; Graves et al., 2016]. 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 photograph on the right, i.e., an image-based instance of the 8-puzzle. Even for humans, this photograph-based task is arguably more difficult to solve than the standard 8-puzzle because of the distracting visual aspects. 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 (1) has no prior assumptions/knowledge (e.g., “sliding objects”, “tile arrangement”), and (2) must acquire all knowledge from the images, is nontrivial. Such a system should not make assumptions about the image (e.g., “a grid-like structure”). 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), an integrated 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 of the problem domain 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. We evaluate LatPlan using image-based versions of the 8-puzzle, LightsOut, and Towers of Hanoi domains.

0-introduction.tex

2. Background

As this work bridges a gap between symbolic planning methods and sub-symbolic image processing and learning methods, we provide brief background on both for the sake of readers coming with both symbolic and subsymbolic backgrounds.

2.1 Neural Networks

Feed Forward Neural Networks (FFN) are nonlinear function approximators consisting of layers of nodes with real-valued activations, and the nodes are connected to multiple nodes in the next
layer by weighted edges. The output of each node is the weighted sum of the activations of the input nodes, transformed by a nonlinear activation function. Recent advances in deep learning have greatly increased the utility of FFNs as a powerful knowledge representation mechanism [Goodfellow et al., 2016].

Although the traditional implementation of neural networks have multiple problems that make the deeply layered networks impractical, they are largely alleviated by the modern techniques: Convolutional Network is capable of learning translation-invariant representation in image-based tasks compared to the fully-connected networks; The learning speed was greatly improved by Stochastic Gradient Descent combined with batch processing and GPU acceleration; The problem of vanishing gradient in deep networks was alleviated by Rectified Linear Unit (ReLU); Dropout [Srivastava et al., 2014] and Regularization reduces the overfitting caused by the excessive representational power of the network; Finally, Batch Normalization [Ioffe and Szegedy, 2015] and recent optimization algorithms such as Adam [Kingma and Ba, 2014] further improves the learning speed.

An AutoEncoder (AE) is a type of Feed-Forward Network (FFN) that uses unsupervised learning to produce an image that matches the input [Hinton and Salakhutdinov, 2006]. 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 pretrained a neural network. The performance of an AE is measured by the reconstruction loss, the distance between the input and the output vectors under a distance function such as $l_1$ norm, $l_2$ norm (euclidean distance) or binary crossentropy.

A Variational AutoEncoder (VAE) [Kingma and Welling, 2013] 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. While initially proposed for enforcing Gaussian distributions, VAEs have been used to enforce arbitrary types of distribution (notably by Generative Adversarial Network [Goodfellow et al., 2014; Makhzani et al., 2015]). 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 $N(\mu, \sigma)$ can be decomposed into $\mu + \sigma N(1, 0)$. In addition to the reconstruction loss, VAE is also tasked to minimize the variational loss, i.e. the difference between the learned distribution and the target distribution.

Gumbel-Softmax (GS) reparametrization is a recently proposed reparametrization technique for a discrete, categorical distribution [Jang et al., 2017]. It has a “temperature” parameter $\tau$, which controls the magnitude of approximation to the categorical distribution, which 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. We chose $r$ so that $\tau = 0.1$ when the training finishes. The above schedule follows the original by Jang et al. (2017). Using a GS layer in the network forces the layer to converge to a discrete one-hot vector when the temperature approaches near 0.

### 2.2 Classical Planning

Classical Planning is achieving a significant advance in the recent years due to the success of heuristic search. The input to a Classical Planning solver (a planner) is a 5-tuple $\Pi = (P, O, I, G, A)$ where $P$ defines a set of first-order predicates, $O$ is a set of symbols called objects, $I$ is the initial state, $G$ is a set of goal conditions, and $A$ is a set of actions which defines the state transitions in the search space. A state is an assignment of boolean values to the set of propositional variables, while a condition is a partial assignment that assigns values only to a subset of propositions. Each propo-
The latent layer converges to the categorical distrib. 

The output converges to the input

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). As the training continues, the output of the network converges to the input images. Also, as the Gumbel-Softmax temperature $\tau$ decreases during training, the latent values approaches the discrete categorical values of 0 and 1.

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

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

transition is described with first-order logic (head parameters...) where the parameters consist of the symbols in $O$, and the number of parameters matches those of the predicate $p \in P$ with the same name. Lifted action schema $a \in A$ is a 5-tuple ($params, pre, e^+, e^-, c$) where each element means the set of parameters, preconditions, add-effects, delete-effects and the cost, respectively. Parameter substitution using objects in $O$ instantiates ground actions.

The task of a planning problem is to find a path from the initial state $I$ to some goal state $s^* \supseteq G$, using the state transition rules in $A$. A state $s$ can be transformed into a new state $t$ by applying a ground action $a$ when $s \supseteq pre$, and then $t = (s \setminus e^-) \cup e^+$ [Bacchus, 2000]. This transition can also be viewed as applying a state transition function $a$ to $s$, which can be written as $t = a(s)$.

State-of-the-Art planners solve this problem as a path finding problem on an implicit graph defined by the state transition rules. They usually employ forward state space heuristic search, such as $A^*$ (for finding the shortest path) or Greedy Best-First Search (for finding a suboptimal path more quickly). Thanks to the variety of successful domain-independent heuristic functions [Helmert and Domshlak, 2009; Sievers et al., 2012; Helmer et al., 2007; Bonet, 2013; Hoffmann and Nebel, 2001; Helmer, 2004; Richter et al., 2008], current state-of-the-art planners can scale to larger problems which requires plans consisting of more than 1000 steps [Asai and Fukunaga, 2015].

0-background.tex

3. LatPlan: System Architecture

This section describes the LatPlan architecture and the current implementation, LatPlan$\alpha$. LatPlan works in 3 phases. In Phase 1 (symbol-grounding, Sec. 3.1), a State AutoEncoder 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. 3.2), the

---

$^1$ 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.
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. 3.3), 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. 3.4).

3.1 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”. The model generation method of Sec. 3.2 could be applied to such “symbols”. 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 [Kingma et al., 2014] with a Gumbel-Softmax (GS) activation function [Jang et al., 2017].

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. The input is fed to a fully connected layer of size \(N \times M\), which is reshaped to a \((N, M)\) matrix and processed by the GS activation function.

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. It is possible to specify different \(M\) for each variable and represent the world using multi-valued representation as in SAS+ [Bäckström and Nebel, 1995]. In this paper, we use \(M = 2\) for all variables for simplicity, and also for leveraging GPU parallelism by running the computation as a matrix operation. This does not affect the expressive power in the model induced by the SAE because bitstrings of sufficient length can represent arbitrary integers in multi-valued encoding.

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

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

\(\text{Encode}(r)\) 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, \(\text{Decode}(b)\) 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.
These are lossy compression/decompression functions, so in general, \( \tilde{r} = \text{Decode}(\text{Encode}(r)) \) is similar to \( r \), but may have negligible errors from \( r \). This is acceptable for our purposes.

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. 3.4, 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 et al., 2017]. A rounding-based scheme would be unable to restore the images from the latent layer because the decoder network 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. 3.2) to Encode every 8-puzzle state.

### 3.2 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 [Cresswell et al., 2013; Gregory and Cresswell, 2015; Konidaris et al., 2014; Mourão et al., 2012; Yang et al., 2007; Celorio et al., 2012]. 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 LatPlan$\alpha$ uses a trivial, baseline strategy which 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 (as argued in Sec. 3.1, a robot could traverse the whole state space by random-walk with resets and eventually collect $R$). In the experiments Sec. 4, 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 in Sec. 3.1, the SAE is trained using at most 12000 images while the entire state space is much larger.

LatPlan$\alpha$ 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.

### 3.3 Planning with an Off-the-Shelf Planner

The PDDL instance generated in the previous step can be solved by an off-the-shelf planner. LatPlan$\alpha$ uses the Fast Downward planner [Helmert, 2006]. However, on the models generated by LatPlan$\alpha$, 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. 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. 4.3 suggest that the domain-independent heuristics can reduce the search effort.
3.4 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 Decoding the latent bit vectors for each intermediate state to images using the SAE.

In this paper, we evaluate LatPlan in Sec. 4 using puzzle domains such as the 8-puzzle, LightsOut, and Towers of Hanoi. Thus, physically “executing” the plan is not necessary, as finding the solution to the puzzles is the objective, so a “mental image” of the solution (i.e., the image sequence visualization) is sufficient. In domains where actions have effects in the world, it will be necessary to consider how actions found by LatPlan (transitions between latent bit vector pairs) can be mapped to actuation (future work).

4. 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 × 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.

4.1 Solving Various Puzzle Domains with LatPlan

**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 et al., 1998]. 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!). From any specific goal state, the reachable number of states is 181440 (9!/2). Note that the same image is used for each digit in all states, e.g., the tile for 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 and a validation set, each consisting of 11000 and 1000 images, where the actual backpropagation-based training of the network is performed on the training set, and the validation set is not given to the learner. Validation set represents the unseen instances: It is later used for ensuring the network is not overfitting, by computing the reconstruction loss |r − ŝ| of the validation set and ensure that it is comparable to the reconstruction loss of the training set. Training takes about 40 minutes with 1000 epochs on a single NVIDIA GTX-1070.
Classical Planning in Deep Latent Space

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 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.

Figure 5: 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 above.
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 $24 \times 122$ and $32 \times 146$ (resp.), where each disk is presented as a 8px line segment.

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. 6, top). However, as the images become more complex, training the SAE becomes more difficult. On 4-disk, the SAE trained with the default hyperparameters (Fig. 6, 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. Increasing the training epochs (10000) and tuning the network shape (fc(6000), $N = 29$) allows the SAE to learn a clearer model, allowing correct model generation, resulting in the optimal 15-step plan (Fig. 6, bottom).

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.

4x4 LightsOut has $2^{16} = 65536$ states and $16 \times 2^{16} = 1048576$ transitions. Similar to the 8-puzzle instances, we used 12000 randomly selected images out of 65536 images, which is then divided into 11000 training images and 1000 validation images.

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 Light-
Figure 7: 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.

Figure 8: Output of solving 3x3 Twisted LightsOut.

sOut”, 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.

4.2 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. 9. The system is robust enough to successfully solve the problem, because our SAE is a Denoising Autoencoder [Vincent et al., 2008] 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.

Figure 9: 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 $\sigma$ up to 0.3 for both problems, and by salt noise up to $p = 0.06$ for Twisted LightsOut. LatPlan$\alpha$ 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 $Decode(b)$. 
4.3 Are Domain-Independent Heuristics Effective in Latent Space?

We compare search using a single PDB with greedy merging [Sievers et al., 2012] and blind heuristics (i.e., breadth-first search) in Fast Downward. The numbers of nodes expanded were:

- **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 [Reinefeld, 1993]): 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 [Korf and Felner, 2002], and PDB generation time can be amortized across many instances.

While the symbolic representation acquired by LatPlan captures the state space graph of the domain, the propositions in the latent space do not necessarily correspond to conceptual propositions in a natural, hand-coded PDDL model. 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. In addition, a deeper understanding of the symbolic latent space may lead to new search heuristics which better exploit the properties of latent space.

5. Related Work

Konidaris et al. propose a method for generating PDDL from a low-level, sensor actuator space of an agent characterized as a semi-MDP (2014). The inputs to their system are 33 variables representing structured sensor input (e.g., x/y distances between each effector and each object, light level) as well as categorical states (the on/off state of a button, whether the monkey has cried out). In contrast, the inputs to LatPlan are unstructured images (e.g., for the 8-puzzle, 42x42=1764-dimensional arrays). Konidaris et al. do not explicitly deal with robustness wrto noisy sensor input, and they focus on action model learning. In contrast, we focus mostly on generating propositions from noisy, unlabeled raw images via our neural-net based SAE, and LatPlanα doesn’t perform action model learning (Sec. 3.2). Integrating action modeling [Konidaris et al., 2014; Mourão et al., 2012; Yang et al., 2007] is a direction for future work.

Our approach differs from the work on learning from observation (LfO) in the robotics literature [Argall et al., 2009] in that: (1) LatPlan is trained based on image pairs showing valid before/after images of valid individual actions, while LfO work is largely based on observation (e.g., of videos) of plan executions; (2) LatPlan generates PDDL for symbolic planners which are suited for high-level (puzzle-like) tasks, while LfO focuses on tasks such as motion planning/manipulation. A closely related line of work in LfO is learning of board game play from observation of video/images.
[Barbu et al., 2010; Kaiser, 2012; Kirk and Laird, 2016]. These works make relatively strong assumptions about the environment, e.g., that there is a grid-like environment with “piece”-like objects. In contrast, as shown in Sec. 4, LatPlan does not make assumptions about the contents of the images.

There is a large body of previous work using neural networks to directly solve combinatorial search/planning tasks, starting with the well-known use of neural network to solve the TSP [Hopfield and Tank, 1985]. With respect to state-space search problems similar to those we consider, Neurosolver, a neural network where each node corresponds to a state in the search space [Bieszczad and Pagurek, 1998], has been used to solves Tower of Hanoi [Bieszczad and Kuchar, 2015]. Although such solvers use neural networks to solve the search problem, they assume a fully symbolic representation of the problem as input.

Previous work combining symbolic search algorithms and neural-networks (NNs) embedded NNs inside a search algorithm to provide search control knowledge. AlphaGo uses a NN (learned from traces of expert Go players as well as self-play) as a heuristic evaluation function to guide search in a Monte-Carlo tree search algorithm [Silver et al., 2016]. In combinatorial search, FFNs have been used as heuristics functions for the sliding-tile puzzle and Rubik’s Cube [Arfaee et al., 2011]. In domain-independent planning, FFNs have been used to learn domain-specific (inadmissible) heuristics for a search-based planner [Satzger and Kramer, 2013]. In contrast, we use a NN-based SAE for symbol grounding, not for search control.

Deep Reinforcement Learning (DRL) has solved complex problems where the input is provided as images, performing well on many video games [Mnih et al., 2015]. For unit-action-cost planning, LatPlan doesn’t require a reinforcement signal (reward function). Also, LatPlan does not require expert solution traces, but only a random sample of the valid moves. Access to expert traces (as in the game of Go [Silver et al., 2016]) is a significant limitation of RL approach because such data may not be readily available. Finally, since LatPlano uses a classical planner, on deterministic, fully-observable single-agent domains, it can provide guarantees of completeness and solution cost optimality (optimal wrt the acquired domain model). An interesting avenue for future work is extending our SAE-based approach as a symbol grounding mechanism for a symbolic, probabilistic (MDP) planner.

An AutoEncoder is a nonlinear generalization of PCA [Bourlard and Kamp, 1988]. Therefore, our approach is somewhat similar to an approach that uses PCA to run RL in the continuous latent space of the configuration space of a high-DOF robot [Luck et al., 2014].

6. 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 problems (8-puzzle, Towers of Hanoi, Lights Out). The key technical contribution is the SAE, which leverages the Gumbel-Softmax reparametrization technique [Jang et al., 2017] and learns (unsupervised) a bidirectional mapping between raw images and a propositional representation usable by symbolic planners. For example, as shown in Sec. 4, 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 modifying any code or manually modifying the neural network architecture. In other words, LatPlan is a domain-independent, image-based classical planner.

To our knowledge, LatPlan is the first system which completely automatically constructs a logical representation usable by an off-the-shelf symbolic planner from a set of unlabeled images for a diverse set of problems, with no explicit assumptions or knowledge about the nature of the domains other than the assumption that the domain can be solved by classical planning and that a sufficient set of training images is available. However, as a proof-of-concept first implementation, it has significant limitations to be addressed in future work.

6.1 Automated Validation of Plans

Since this paper focuses on demonstrating the feasibility of a symbolic planning system with neural perception, the experimental results included in the paper is mostly qualitative. To facilitate a more quantitative evaluation of LatPlan, one important direction for future work includes the development of a practical methods for validating plans. LatPlan should return a valid visual plan, i.e., a plan which does not violate the rules in the original input. In Sec. 4, we observed that the result plan may be invalid depending on the performance of the SAE. In classical planning, the validation of the solution plan is trivial as the plan simulator and the explicit action description is readily available. However, validating the plan returned by LatPlan requires manual validation by human eye which checks every moves presented in the result, which does not scale to a large number of problem instances/plans, and is also prone to errors. This prevents a convenient and reliable evaluation of LatPlan. One approach to this problem is to use crowd sourcing infrastructure like Amazon Mechanical Turk, in combination with techniques for reducing the variance of the evaluation [Yuen et al., 2011]. However, development of automated methods for plan validation is more preferable. Approaches such as the inception score for evaluating the performance of Generative Adversarial Network [Salimans et al., 2016] suggests that such automated approaches may be feasible for LatPlan as well.

6.2 Action Learning

The SAE successfully solves the problem of mapping a pair of images into a ground symbolic action using only a subset of the world state images (Sec. 3.2). On the other hand, the domain model generator in the current LatPlanα implementation does not perform action model learning/induction from a small set of sample actions.

Since the focus of this paper is the evaluation of SAE and not action learning, the current, baseline domain model generator requires the entire set of latent states/transitions, which in turn requires an image for each state in the state space. LatPlanα essentially constructs an explicit state space graph based on action image pairs for all ground actions in the domain, so the planner is being used to find optimal paths in an explicit graph. In other words, the preconditions of the actions generated by LatPlanα specify every bit in a state, unlike the partial specification of a state that is
common in IPC benchmark domains which allows the implicit definition of very large state spaces. This kind of trivial domain model generator which uses images for all states in the entire state space is obviously impractical in many domains. Aside from the modeling issue, the trivial explicit state-space model causes practical issues with current off-the-shelf planners, as the huge total number of actions (edges in the explicit state space model) causes major slowdowns in both the PDDL parser (Sec. 3.3), as well as the initialization runtime of heuristics (Sec. 4.3).

However, these are not fundamental limitations of the LatPlan architecture. The current primitive model generator in LatPlan is merely a placeholder which was necessary to enable us to investigate the utility of the SAEs (our major contribution) and the overall feasibility of an end-to-end planning system based on raw images. Thus, the focus of our work and our contributions are orthogonal and complementary to the goals of previous work on domain model learning. To our knowledge, all previous planning domain model learning methods assume/require as input representations of states which are highly structured (e.g., propositional).

The SAE provides a method of automatically extracting propositional representations from raw images. Thus, we believe that it should be possible to adapt and apply action model learning methods which have been developed for environments with deterministic effects and fully observable states [Celorrio et al., 2012, Sec.3, Fig.2] to the propositional representations generated using the SAE in LatPlan. Thus, replacing this primitive generator in LatPlan with a more sophisticated generator [Cresswell et al., 2013; Konidaris et al., 2014; Mourão et al., 2012; Yang et al., 2007] is an important direction for future work.

Another related direction for future work is how to specify the goal condition for LatPlan. Since LatPlan assumes a single goal state as an input, developing a method for specifying a set of goal states with a partial goal specification as in IPC domains is an interesting future work. For example, one may want to tell the planner “the goal states must have at least tile 0,1,2 in the correct places, but we do not care the others” in a MNIST 8-puzzle instance.

6.3 Collecting Images

LatPlan is a proof-of-concept architecture which shows the feasibility of constructing an end-to-end planning which uses raw images as input. As such, we did not address the practical issue of how the images are obtained by a system which uses image-based planning. However, in a real-world implementation of a system like LatPlan i.e., an image-based planner for a robot with sensors/camera and manipulators, collecting the data (images) needed to train the SAE is a practical issue. The requirement for collecting data for LatPlan would be significantly different from those that are common in learning-from-observation systems.

Unlike learning-from-observation, where the learner does not know when an action starts/ends and should recognize each action from the plan traces, we assume that a robot has a lower-level manipulation capabilities of safely, randomly exploring the world by itself, deciding to initiate/terminate its own action, manipulating the state of the world and observing the consequences. The robot can perform a random walk, collecting images along the way. In many domains, physical constraints will ensure that the robot can only perform legal moves (e.g., the physical tile board in 8-puzzle, the touch display in a LightsOut video game). In domains such as Towers of Hanoi where illegal moves are physically possible, we can assume either that a teacher (e.g., human) prevents the robot

---

2. Note that if the learner controls when actions are initiated/terminated, action segmentation (identifying which images indicate the start/end of individual actions) is not a issue.
from making illegal moves, or that the teacher filters the training images taken by the robot. If we further assume that it is possible to periodically “reset” the world (e.g., a teacher comes and resets the world into a random configuration), then, given enough time, the robot could obtain images of the entire state space.

In domains where obtaining training images is expensive, another bottleneck in an image-based planning system like LatPlan is the number of images required to train the SAE. Developing more effective learner for minimizing the training data, such as One-shot learning methods [Lake et al., 2013], is an important direction for future work. Since a better learner needs less examples, the number of required images depends on the generalization capability of AE, which is ongoing work in the DL community.

6.4 Improving the SAE

Although we showed that LatPlan works on several kinds of images, including MNIST handwritten digits, photographs (Mandrill), and several synthetic images (Hanoi, Lights Out), we do not claim that the specific implementation of SAE used in this paper works robustly on all images/data. For example, some tuning of the image (shrinking/binarization) was necessary in order to get the SAE working for the photograph-based 8-puzzles. 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 in deep learning and other image processing techniques.

References

[Arfaee et al., 2011] Shahab Jabbari Arfaee, Sandra Zilles, and Robert C. Holte. Learning Heuristic Functions for Large State Spaces. Artificial Intelligence, 175(16-17):2075–2098, 2011.

[Argall et al., 2009] Brenna Argall, Sonia Chernova, Manuela M. Veloso, and Brett Browning. A Survey of Robot Learning from Demonstration. Robotics and Autonomous Systems, 57(5):469–483, 2009.

[Asai and Fukunaga, 2015] Masataro Asai and Alex Fukunaga. Solving Large-Scale Planning Problems by Decomposition and Macro Generation. In Proc. of the International Conference on Automated Planning and Scheduling (ICAPS), Jerusalem, Israel, June 2015.

[Bacchus, 2000] Fahiem Bacchus. Subset of PDDL for the AIPS2000 Planning Competition. In Proceedings of AIPS’00 Planning Competition, 2000.

[Bäckström and Nebel, 1995] Christer Bäckström and Bernhard Nebel. Complexity Results for SAS+ Planning. Computational Intelligence, 11(4):625–655, 1995.

[Barbu et al., 2010] Andrei Barbu, Siddharth Narayanaswamy, and Jeffrey Mark Siskind. Learning Physically-Instantiated Game Play through Visual Observation. In Proc. of IEEE International Conference on Robotics and Automation (ICRA), pages 1879–1886, 2010.

[Bieszczad and Kuchar, 2015] Andrzej Bieszczad and Skyler Kuchar. Neurosolver Learning to Solve Towers of Hanoi Puzzles. In Computational Intelligence (IJCCI), 2015 7th International Joint Conference on, volume 3, pages 28–38. IEEE, 2015.

[Bieszczad and Pagurek, 1998] Andrzej Bieszczad and Bernard Pagurek. Neurosolver: Neuromorphic General Problem Solver. Information Sciences, 105(1-4):239–277, 1998.
[Bonet, 2013] Blai Bonet. An Admissible Heuristic for SAS+ Planning Obtained from the State Equation. In Proc. of International Joint Conference on Artificial Intelligence (IJCAI), 2013.

[Bourlard and Kamp, 1988] Hervé Bourlard and Yves Kamp. Auto-Association by Multilayer Perceptrons and Singular Value Decomposition. Biological Cybernetics, 59(4):291–294, 1988.

[Celorrio et al., 2012] Sergio Jiménez Celorio, Tomás de la Rosa, Susana Fernández, Fernando Fernández, and Daniel Borrero. A Review of Machine Learning for Automated Planning. Knowledge Eng. Review, 27(4):433–467, 2012.

[Cresswell et al., 2013] Stephen Cresswell, Thomas Leo McCluskey, and Margaret Mary West. Acquiring planning domain models using LOCM. Knowledge Eng. Review, 28(2):195–213, 2013.

[Deng et al., 2009] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In Proc. of IEEE Conference on Computer Vision and Pattern Recognition, pages 248–255. IEEE, 2009.

[Deng et al., 2013] Li Deng, Geoffrey Hinton, and Brian Kingsbury. New Types of Deep Neural Network Learning for Speech Recognition and Related Applications: An Overview. In Proc. of IEEE Conference on Acoustics, Speech and Signal Processing, pages 8599–8603. IEEE, 2013.

[Goodfellow et al., 2014] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In Advances in Neural Information Processing Systems, pages 2672–2680, 2014.

[Goodfellow et al., 2016] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning. MIT Press, 2016. http://www.deeplearningbook.org.

[Graves et al., 2016] Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. Hybrid Computing using a Neural Network with Dynamic External Memory. Nature, 538(7626):471–476, 2016.

Gregory and Cresswell, 2015] Peter Gregory and Stephen Cresswell. Domain model acquisition in the presence of static relations in the LOP system. In Proc. of the International Conference on Automated Planning and Scheduling(ICAPS), pages 97–105, 2015.

[Helmert and Domshlak, 2009] Malte Helmert and Carmel Domshlak. Landmarks, Critical Paths and Abstractions: What’s the Difference Anyway? In Proc. of the International Conference on Automated Planning and Scheduling(ICAPS), 2009.

[Helmert et al., 2007] Malte Helmert, Patrik Haslum, and Jörg Hoffmann. Flexible Abstraction Heuristics for Optimal Sequential Planning. In Proc. of the International Conference on Automated Planning and Scheduling(ICAPS), 2007.

[Helmert, 2004] Malte Helmert. A planning heuristic based on causal graph analysis. In Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS 2004), June 3-7 2004, Whistler, British Columbia, Canada, pages 161–170, 2004.

[Helmert, 2006] Malte Helmert. The Fast Downward Planning System. J. Artif. Intell. Res.(JAIR), 26:191–246, 2006.

[Hinton and Salakhutdinov, 2006] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. Science, 313(5786):504–507, 2006.

[Hoffmann and Nebel, 2001] Jörg Hoffmann and Bernhard Nebel. The FF Planning System: Fast Plan Generation through Heuristic Search. J. Artif. Intell. Res.(JAIR), 14:253–302, 2001.

[Hopfield and Tank, 1985] John Hopfield and David W Tank. "Neural” Computation of Decisions in Optimization Problems. Biological cybernetics, 52(3):141–152, 1985.
[Ioffe and Szegedy, 2015] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In Proc. of the International Conference on Machine Learning, pages 448–456, 2015.

[Jang et al., 2017] Eric Jang, Shixiang Gu, and Ben Poole. Categorical Reparameterization with Gumbel-Softmax. Proc. of the International Conference on Learning Representations, 2017. To appear. preprint at https://arxiv.org/abs/1611.01144.

[Kaiser, 2012] Łukasz Kaiser. Learning Games from Videos Guided by Descriptive Complexity. In Proc. of AAAI Conference on Artificial Intelligence, 2012.

[Kingma and Ba, 2014] Diederik Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. arXiv preprint arXiv:1412.6980, 2014.

[Kingma and Welling, 2013] Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. Proc. of the International Conference on Learning Representations, 2013.

[Kingma et al., 2014] Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In Advances in Neural Information Processing Systems, pages 3581–3589, 2014.

[Kirk and Laird, 2016] James R Kirk and John E Laird. Learning General and Efficient Representations of Novel Games Through Interactive Instruction. Advances in Cognitive Systems, 4, 2016.

[Konidaris et al., 2014] George Konidaris, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Constructing Symbolic Representations for High-Level Planning. In Proc. of AAAI Conference on Artificial Intelligence, pages 1932–1938, 2014.

[Korf and Felner, 2002] Richard E. Korf and Ariel Felner. Disjoint Pattern Database Heuristics. Artificial Intelligence, 134(1-2):9–22, 2002.

[Lake et al., 2013] Brenden M Lake, Ruslan R Salakhutdinov, and Josh Tenenbaum. One-shot learning by inverting a compositional causal process. In Advances in neural information processing systems, pages 2526–2534, 2013.

[LeCun et al., 1998] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-Based Learning Applied to Document Recognition. Proc. of the IEEE, 86(11):2278–2324, 1998.

[Luck et al., 2014] Kevin Sebastian Luck, Gerhard Neumann, Erik Berger, Ian Peters, and Heni Ben Amor. Latent Space Policy Search for Robotics. In Proc. of IEEE International Workshop on Intelligent Robots and Systems (IROS), pages 1434–1440. IEEE, 2014.

[Maddison et al., 2017] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. Proc. of the International Conference on Learning Representations, 2017. To appear. preprint at https://arxiv.org/abs/1611.00712.

[Makhzani et al., 2015] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial Autoencoders. arXiv preprint arXiv:1511.05644, 2015.

[McDermott, 2000] Drew V. McDermott. The 1998 AI Planning Systems Competition. AI Magazine, 21(2):35–55, 2000.

[Mnih et al., 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-Level Control through Deep Reinforcement Learning. Nature, 518(7540):529–533, 2015.

[Mourão et al., 2012] Kira Mourão, Luke S. Zettlemoyer, Ronald P. A. Petrick, and Mark Steedman. Learning STRIPS Operators from Noisy and Incomplete Observations. In Proc. of the International Conference on Uncertainty in Artificial Intelligence, pages 614–623, 2012.
[Reinefeld, 1993] Alexander Reinefeld. Complete Solution of the Eight-Puzzle and the Benefit of Node Ordering in IDA. In Proc. of International Joint Conference on Artificial Intelligence (IJCAI), pages 248–253, 1993.

[Ren et al., 2015] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-time Object Detection with Region Proposal Networks. In Advances in Neural Information Processing Systems, pages 91–99, 2015.

[Richter et al., 2008] Silvia Richter, Malte Helmert, and Matthias Westphal. Landmarks Revisited. In Proc. of AAAI Conference on Artificial Intelligence, 2008.

[Salimans et al., 2016] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved Techniques for Training Gans. In Advances in Neural Information Processing Systems, pages 2226–2234, 2016.

[Satzger and Kramer, 2013] Benjamin Satzger and Oliver Kramer. Goal Distance Estimation for Automated Planning using Neural Networks and Support Vector Machines. Natural Computing, 12(1):87–100, 2013.

[Sievers et al., 2012] Silvan Sievers, Manuela Ortlieb, and Malte Helmert. Efficient Implementation of Pattern Database Heuristics for Classical Planning. In Proc. of Annual Symposium on Combinatorial Search, 2012.

[Silver et al., 2016] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the Game of Go with Deep Neural Networks and Tree Search. Nature, 529(7587):484–489, 2016.

[Srivastava et al., 2014] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research, 15(1):1929–1958, 2014.

[Steels, 2008] Luc Steels. The Symbol Grounding Problem has been solved. So what’s next? In Manuel de Vega, Arthur Glenberg, and Arthur Graesser, editors, Symbols and Embodiment. Oxford University Press, 2008.

[Vincent et al., 2008] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and Composing Robust Features with Denoising Autoencoders. In Proc. of the International Conference on Machine Learning, pages 1096–1103. ACM, 2008.

[Yang et al., 2007] Qiang Yang, Kangheng Wu, and Yunfei Jiang. Learning Action Models from Plan Examples using Weighted MAX-SAT. Artificial Intelligence, 171(2-3):107–143, 2007.

[Yuen et al., 2011] Man-Ching Yuen, Irwin King, and Kwong-Sak Leung. A Survey of Crowdsourcing Systems. In Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third International Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on, pages 766–773. IEEE, 2011.