Towards Stable Symbol Grounding
with Zero-Suppressed State AutoEncoder

Masataro Asai
IBM Research
MIT-IBM Watson AI Lab

Hiroshi Kajino
IBM Research

Abstract

While classical planning has been an active branch of AI, its applicability is limited to the tasks precisely modeled by humans. Fully automated high-level agents should be instead able to find a symbolic representation of an unknown environment without supervision, otherwise it exhibits the knowledge acquisition bottleneck. Meanwhile, Latplan (Asai and Fukunaga 2018) partially resolves the bottleneck with a neural network called State AutoEncoder (SAE). SAE obtains the propositional representation of the image-based puzzle domains with unsupervised learning, generates a state space and performs classical planning. In this paper, we identify the problematic, stochastic behavior of the SAE-produced propositions as a new sub-problem of symbol grounding problem, the symbol *stability* problem. Informally, symbols are *stable* when their referents (e.g. propositional values) do not change against small perturbation of the observation, and unstable symbols are harmful for symbolic reasoning. We analyze the problem in Latplan both formally and empirically, and propose "Zero-Suppressed SAE", an enhancement that stabilizes the propositions using the idea of closed-world assumption as a prior for NN optimization. We show that it finds the more stable propositions and the more compact representations, resulting in an improved success rate of Latplan. It is robust against various hyperparameters and eases the tuning effort, and also provides a weight pruning capability as a side effect.

1 Introduction

Symbol grounding problem (Harnad 1990; Steels 2008) is one of the key milestones in AI research which seeks to achieve high-level intelligence. In Physical Symbol Systems Hypothesis (Newell and Simon 1976), it is believed that an agent with high-level intelligence performs tasks by efficiently manipulating a compact set of abstract symbols. Symbolic manipulation allows for the development of highly optimized and generalized, domain-independent heuristics (Hoffmann and Nebel 2001; Helmert and Domshlak 2009) that can be easily applied to multiple tasks with few or no data, while the current learning-based approaches struggle to improve its multi-task performance and data efficiency. To enable such a symbolic computation in a real-world environment, agents should be able to find the symbolic representation of the environment by itself.

Recently, Latplan system (Asai and Fukunaga 2018) successfully connected a subsymbolic neural network (NN) system and a symbolic Classical Planning system to solve various visually presented puzzle domains. The State AutoEncoder (SAE) neural network in Latplan generates a set of propositional symbols from the training images with no additional information and provides a bidirectional mapping between images and propositional states. The system then solves the propositional planning problem using a classical planner Fast Downward (Helmert 2004) and returns an image sequence that solves the puzzle by decoding the intermediate propositional states of the plan. It also discovers a set of action symbols that distinguish the modes of state transitions through AMA_2 unsupervised learning process. Thus the system grounds two kinds of symbols: Propositional symbols and action symbols, and opens a promising direction for applying a variety of symbolic methods to the real world. The search space generated by Latplan was shown to be compatible to a symbolic Goal Recognition system (Amado et al. 2018a). Another approach replacing SAE/AMA_2 with Info-GAN was also proposed recently (Kurutach et al. 2018).

Despite its success, the propositional representations learned by SAEs have a problematic behavior due to its lack of strong guarantees on the learned results. That is, while the SAE can reconstruct the input with high accuracy, the learned latent representations are not "stable", i.e., some propositions may flip the value (true/false) randomly given the identical or nearly identical image input. This is mainly because SAE learns the mapping between images and binary representations as a many-to-many relationship. While this property has not been considered as an issue in the machine learning community where only the accuracy matters, its unstable latent representation poses a significant threat to the reasoning ability of the planners.

Unstable symbols are harmful for symbolic reasoning because they break the identity assumption built into the reasoning algorithms. For instance, in Latplan, a single image may map to multiple propositional states due to stochasticity; therefore the duplication detection in search algorithms such as A^* (Hart, Nilsson, and Raphael 1968) fails to realize that a single real-world state is visited multiple times through different symbolic state representations. Unlike machine learning tasks, the symbolic planning requires a mapping that abstracts many images into a single symbolic state,
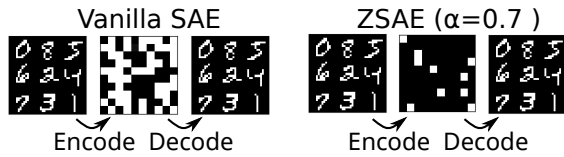
Figure 1: Autoencoding results of an MNIST 8-puzzle state using a vanilla State AutoEncoder (SAE) (Asai and Fukunaga 2018) and a proposed Zero-Suppressed SAE (ZSAE) with 100 propositions. ZSAE obtains a compact representation that uses fewer true bits.

i.e., many-to-one mapping. To this end, it is necessary to develop an autoencoder that learns a many-to-one relationship between images and binary representations.

The contribution of this paper is threefold. The first contribution of this paper is the identification of a sub-problem of symbol grounding called "symbol *stability* problem" (SSP), which seeks to find a set of symbols whose values/referents stays the same for the same/similar raw inputs. Stability is orthogonal to the *performance/accuracy* of NNs: NNs are known to predict the correct output robustly, but they do not guarantee the quality of the internal representation. It is also unrelated to the numerical stability of the training or the reproducibility of the results.

The second contribution of this paper is the formal analysis of the Latplan system, in particular, the comparison to the original mathematical model of Gumbel-Softmax (Jang, Gu, and Poole 2017). The analysis revealed that Latplan's Gumbel-Softmax has a deviation from the original, which was the key to its first success: The slight modification acted as an *Entropy Regularization* term for the SAE neural network that suppresses the randomness to some extent.

The third contribution of this paper is the proposal of Zero-Suppressed State AutoEncoder (ZSAE, Fig. 1). Inspired by the fact that the *Entropy Regularization* stabilizes the representation, ZSAE further stabilizes the propositions by an additional regularization term which guides the network optimization so that unused propositions tend to take the value of zero (false) instead of random values. The stable representation results in a higher success rate of classical planning. Also, the network is less sensitive to the network size (hyperparameters) as it automatically reduces the number of bits used. Moreover, we show that we can reduce the memory usage of the network by pruning some unused neurons that now have a constant activation of zero instead of random values.

2 Preliminaries

Symbol grounding is an unsupervised process of establishing a mapping from huge, noisy, continuous, unstructured inputs to a set of compact, discrete, identifiable (structured) entities, i.e., symbols (Harnad 1990; Steels 2008; Asai and Fukunaga 2018). PDDL (McDermott 2000) has six kinds of symbols: Objects, predicates, propositions, actions, problems and domains. Each type of symbol requires its own mechanism for grounding. For example, the large body of work in the image processing community on rec-

ognizing objects (e.g., faces) and their attributes (male, female) in images, or scenes in videos (e.g., cooking) can be viewed as corresponding to grounding the object, predicate and action symbols, respectively. In this paper, we focus on grounding the propositional symbols.

Latplan (Asai and Fukunaga 2018) is a framework for *domain-independent image-based classical planning*. Latplan is able to ground the propositional and action symbols. Classical planners such as FF (Hoffmann and Nebel 2001) or FastDownward (Helmert 2004) takes a PDDL model as an input, which specifies the state representation and the transition rules. In contrast, Latplan learns the state representation as well as the transition rules entirely from the image-based observation of the environment with deep NNs. The system was shown to solve various puzzle domains, such as 8-puzzles or Tower of Hanoi, that are presented in the form of noisy, continuous visual depiction of the environment.

Latplan takes two inputs. The first input is the *transition input Tr*, a set of pairs of raw data. Each pair $tr_i = (pre_i, suc_i) \in Tr$ represents a transition of the environment before and after some action is executed. The second input is the *planning input (i, g)*, a pair of raw data, which corresponds to the initial and the goal state of the environment. The output of Latplan is a data sequence representing the plan execution that reaches g from i. While the original paper uses an image-based implementation ("raw data" = images), the type of data is arbitrary as long as it is compatible with NNs.

Latplan works in 3 phases. In Phase 1, a *State AutoEncoder* (SAE) (Fig. 2) learns a bidirectional mapping between raw data (subsymbolic representation e.g., images) and propositional states (symbolic representation) from a set of unlabeled, random snapshots of the environment. The trained SAE provides two functions:

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

After training the SAE from $\{pre_i, suc_i \ldots\}$, it applies $Encode$ to each $tr_i \in Tr$ and obtains $(Encode(pre_i), Encode(suc_i)) = (s_i, t_i) = \overline{tr_i} \in \overline{Tr}$, the symbolic representations (latent space vectors) of the transitions.

In Phase 2, an Action Model Acquisition (AMA) method learns an action model (e.g., PDDL, successor function) from $\overline{Tr}$ in an unsupervised manner. The original paper proposed two approaches: AMA_1 is an oracle which directly generates a PDDL without learning, by allowing it to use the whole set of valid transitions as an oracle. In contrast, AMA_2 approximates AMA_1 by unsupervised learning from examples.

In Phase 3, a planning problem instance is generated from the planning input (i, g). These are converted to the symbolic states by the SAE, and the symbolic planner solves the problem combining (i, g) and the generated action model. For example, an 8-puzzle problem instance consists of an image of the start (scrambled) configuration of the puzzle (i), and an image of the solved state (g).

Since the intermediate states comprising the plan are SAE-generated latent bit vectors, the "meaning" of each state (and thus the plan) is not clear to a human observer.

(Example with N=25)

The latent layer converges to the categorical distrib.

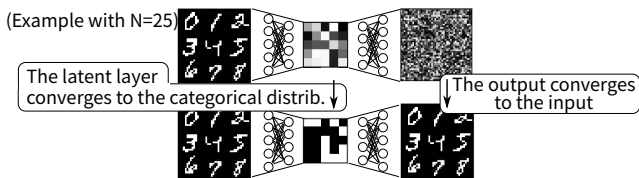The output converges to the input

Figure 2: Step 1: Train the State AutoEncoder by minimizing the sum of the reconstruction loss and the variational loss of Gumbel-Softmax. As the training continues, the output of the network converges to the input images. Also, as the Gumbel-Softmax temperature τ decreases during training, the latent values approach either 0 or 1.

However, in the final step, Latplan obtains a step-by-step visualization of the plan execution by $Decode$'ing the latent bit vectors for each intermediate state. The plans or the state transitions are validated based on the visualized result using a custom domain-specific validator implemented in Latplan. This is because the intermediate latent representation is learned unsupervised and is not directly verifiable through human knowledge.

SAE as a Gumbel-Softmax Variational AutoEncoder
The key concept of the SAE in Latplan is the use of Gumbel-Softmax (Jang, Gu, and Poole 2017) in the latent activation of the variational autoencoder (VAE). This allows the SAE to obtain a discretized binary representation, and Latplan uses this discrete vector as the state representation for classical planning. We briefly review the related literature below.

An autoencoder (AE) (Hinton and Salakhutdinov 2006) is a feed-forward NN that consists of a pair of encoder and decoder networks, both of which are modeled by continuous functions. For example, in Figure 2, the mapping from the leftmost image to the vector in the middle corresponds to the encoder, and the mapping from the middle to the rightmost image corresponds to the decoder. AEs are trained so that the encoder maps a data point (e.g., an image) into a low-dimensional latent space, and the decoder pulls the latent representation of the input back to the original data point. Technically, they are trained by a *backpropagation* algorithm so as to minimize the reconstruction loss, the distance between the input and the output measured by Euclidean distance or binary cross entropy. Since the encoder is modeled by a continuous function, its latent representation is also continuous, which makes it challenging to integrate AEs with propositional reasoners.

A variational autoencoder (VAE) (Kingma and Welling 2013) is a probabilistic variant of AEs, whose encoder and decoder are modeled by probabilistic distributions rather than deterministic functions. For instance, it obtains a discrete latent representation of the data by modeling the Bernoulli(=binary) random distribution with the encoder. VAEs are trained by backpropagation with the help of the reparameterization trick, which makes random variables differentiable. The reparametrization trick for the Bernoulli distribution is Gumbel-Softmax (Jang, Gu, and Poole 2017).

A single Gumbel-Softmax unit in the Neural Network is able to model a categorical distribution with M categories which includes a Bernoulli distribution ($M = 2$) as a special case. Latplan uses $M = 2$ for simplicity, and it does not affect the expressivity of the representation, similar to the relation between SAS+ and the propositional representation.

$$z_i = \textbf{if } (i \text{ is } \arg\max_i (g_i + \log \pi_i)) \textbf{ then } 1 \textbf{ else } 0. \quad (1)$$

$$z_i = \text{Softmax}((g_i + \log \pi_i)/\tau). \quad (2)$$

The output of a single Gumbel-Softmax unit $GS(\boldsymbol{\pi}) = \mathbf{z} = (z_i)$ $(0 \leq i \leq M)$ is a one-hot vector representing M categories, e.g., when $M = 2$ the categories can be seen as $\{\text{false}, \text{true}\}$ and $\mathbf{z} = (0, 1)$ represents "true". (Note: There is no explicit meaning assigned to each category.) The input $\boldsymbol{\pi} = (\pi_i)$ is a class probability vector, e.g. $(.2, .8)$. Gumbel-Softmax is derived from Gumbel-Max technique (Maddison, Tarlow, and Minka 2014, Eq. 1) for drawing a categorical sample from $\boldsymbol{\pi}$ where g_i is a sample drawn from $\text{Gumbel}(0, 1) = -\log(-\log u)$ where $u = \text{Uniform}(0, 1)$ (Gumbel and Lieblein 1954). Gumbel-Softmax approximates the argmax with a softmax to make it differentiable (Eq. 2). "Temperature" τ controls the magnitude of approximation, which is annealed to 0 by a certain schedule. The output $\mathbf{z}$ converges to a discrete one-hot vector when $\tau \to 0$.

SAEs have N Gumbel-Softmax units to model an N-dimensional Bernoulli/boolean/propositional variable $\mathbf{b} = (b_n)$. N units produce a matrix z_{nk} where $1 \leq n \leq N$ and $k \in \{0, 1\}$ and the boolean variables are retrieved by $b_n = z_{n1}$.

3 Symbol Stability Problem

The vanilla SAE in Latplan can map a visual observation of the environment to/from a set of propositional values. An issue with the vanilla SAEs is that the class probability for the class "true" and the class "false" that is mapped to by the Gumbel-Softmax could be neutral at some neuron, causing the value of the neuron to change frequently (Fig. 3). The source of stochasticity is twofold. The first source is the probabilistic distribution modeled by the encoder, which introduces stochasticity and causes the propositions to change values even for the exact same inputs. The second source is the stochastic observation of the environment which corrupts the input image. When the class probabilities are almost neutral, such a tiny variation in the input image may cause the activation to go across the decision boundary for each neuron, causing the bit flips. In contrast, humans still regard the corrupted image as the "same" image.

This stochastic behavior of the propositional representation introduces several issues to the recipient symbolic systems such as classical planners. Firstly, search algorithms that run on the state space generated by the propositional vectors are confused by many variations of the essentially identical real-world states. It could visit the "same" real world state several times because it could be encoded into different propositional vectors which are not detected by the duplicate detection in the search algorithms (e.g. A^*). This slows down the search by increasing the number of nodes that are reachable from the initial state.
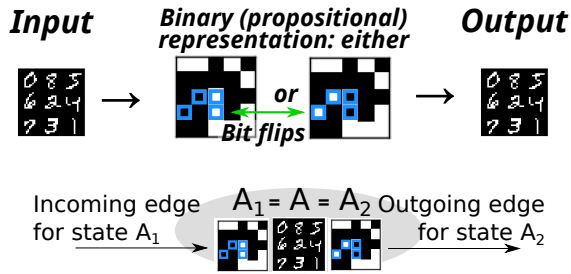
Figure 3: (**Top**) Propositions found by SAEs may contain uninformative random bits that do not affect the output. (**Bottom**) The random variations of the propositional encoding could disconnect the search space.

Secondly, the state space could be disconnected due to such random variations (Fig. 3). Some states may be reached only via a single variation of the real world state and is not connected to another propositional variation of the same real-world state. In fact, in the appendix section in the Arxiv version of the original paper (Asai and Fukunaga 2018), the planning part used a so-called *state augmentation* technique which circumvents this issue by sampling states from the same image multiple times.

Thirdly, in order to reduce the stochasticity of the propositions, we encounter a hyperparameter tuning problem which is costly when we train a large NN. The neurons that behave randomly for the same or perturbed input do not affect the output, i.e., they are unused and uninformative. Unused neurons appear because the network has an excessive capacity to model the entire state space, i.e., they are surplus neurons. Therefore, a straightforward way to reduce those neurons is to reduce the size of the latent space N. On the other hand, if N is too small, it lacks the capacity to represent the state space and the SAE no longer learns to reconstruct the real world image. As a result, we face a hyperparameter tuning problem: Surplus N causes an unstable representation, and insufficient N makes the network hard to train.

Fundamentally, the first two harmful effects are caused by breaking a critical feature of symbols, *designation* (Newell and Simon 1976), that each symbol uniquely refers to an entity (referent, concept, meaning), e.g., the referents of the symbols grounded by SAEs are the truth assignments. If meaning of a symbol changes frequently and unexpectedly, the entire symbolic manipulation is fruitless because the underlying symbols are not tied to any particular concept, and does not represent the real world.

Thus, for a symbol grounding procedure to produce a set of symbols for symbolic reasoning, it is insufficient to find a set of symbols that are *just able to* represent the environment; It should find a *stable* symbolic representation that *uniquely* represents the environment.

Definition 1. Symbolic representation *of an environment is a set of symbols with referents from which the environment can be reconstructed with sufficient accuracy.*

Definition 2. *Symbolic representation is* stable *when its referents are identical for the same environment, under some* equivalence relation (e.g., invariance, noise threshold).

While NNs tend to achieve a robust performance on noisy data, **the *robust performance* and the stability of the representation are orthogonal.** This is because the former exclusively deals with the output accuracy, while the latter evaluates the quality of the *latent* activations while maintaining the same output accuracy. In fact, vanilla SAEs already achieve the almost perfect reconstruction accuracy where the input and the output are indiscernible to human eyes (e.g., in Fig. 3, the pixel value output is different from the input, but we cannot recognize it), while they still exhibit instability.

The stability of the representation obtained by a NN depends on its inherent stochasticity during the runtime (as opposed to the training time) as well as the stochasticity of the environment. These observations indicate that *any* symbol grounding system potentially suffers from the symbol stability problem. As for the stochasticity of the environment, in many real-world tasks, it is common to obtain stochastic observations due to the external interference, e.g., vibrations of the camera caused by the wind. As for the stochasticity of the network, both VAEs (Kingma and Welling 2013; Jang, Gu, and Poole 2017; Higgins et al. 2017) used in Latplan and GANs (Generative Adversarial Networks) (Goodfellow et al. 2014) used in Causal InfoGAN (Kurutach et al. 2018) rely on sampling processes.

4 Analyzing the State Autoencoder

To obtain a deeper understanding of the mechanism that generates unstable symbols in vanilla SAEs, we analyzed its mathematical model and the public source code of Latplan (commit d48ee62). We found that the source code differs from the original mathematical formulation of Gumbel-Softmax VAE (GS-VAE) in that they are using an alternative loss function, and we found that *this very change* turned out to be essential to the success of their experiments by suppressing the instability of the propositions. In the following, we illustrate our finding using the idealized case where the encoder is the Bernoulli distribution, although in reality, it is approximated by a Gumbel-softmax distribution.

Given a dataset $\mathcal{X} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(I)}\}$, let $q(\mathbf{b} \mid \mathbf{x})$ and $p(\mathbf{x} \mid \mathbf{b})$ be the probabilities that the encoder and the decoder of GS-VAE respectively outputs the value $\mathbf{b}$ given $\mathbf{x}$ and $\mathbf{x}$ given $\mathbf{b}$, where $\mathbf{x} \in \mathcal{X}$ corresponds to a visual observation of the environment, and $\mathbf{b} \in \{0, 1\}^N$ corresponds to its latent representation. In principle, GS-VAE is trained by minimizing the following objective function with respect to $p(\mathbf{x} \mid \mathbf{b})$ and $q(\mathbf{b} \mid \mathbf{x})$:

$$\sum_{\mathbf{x} \in \mathcal{X}} \left(\mathbb{E}_{q(\mathbf{b}|\mathbf{x})} \left[-\log p(\mathbf{x} \mid \mathbf{b}) \right] + \mathrm{KL}(q(\mathbf{b} \mid \mathbf{x}) \| p(\mathbf{b})) \right) \quad (3)$$

where $p(\mathbf{b}) = \prod_{n=1}^{N} p(b_n) = \prod_{n=1}^{N} \mathrm{Bern}(0.5)$ is the target, N-dimensional Bernoulli distribution with uniform probabilities and $\mathrm{KL}(q\|p)$ represents the Kullback-Leibler divergence from p to q. The first term in Eq. (3) is the reconstruction loss, which measures the quality of the reconstructed data points, and the second term regularizes the encoder by making the encoder $q(\mathbf{b} \mid \mathbf{x})$ closer to $p(\mathbf{b})$. The second

term is computed as

$$\text{KL}(q(\mathbf{b} \mid \mathbf{x}) \parallel p(\mathbf{b}))$$

$$= -\sum_{n=1}^{N} \sum_{k \in \{0,1\}} q(b_n = k \mid \mathbf{x}) \log \frac{p(b_n = k)}{q(b_n = k \mid \mathbf{x})}$$

$$= -\sum_{n=1}^{N} \sum_{k \in \{0,1\}} q(b_n = k \mid \mathbf{x}) \left(\log \frac{1}{2} - \log q(b_n = k \mid \mathbf{x}) \right)$$

$$= -\sum_{n=1}^{N} H_q(b_n \mid \mathbf{x}) + \text{const.} = -H_q(\mathbf{b} \mid \mathbf{x}) + \text{const.},$$

where $H_q(\mathbf{b}|\mathbf{x})$ is the entropy of $\mathbf{b}$ given $\mathbf{x}$ under q.

Entropy Regularization in Latplan We found that the loss computation in the Latplan code has the *opposite sign* on the KL divergence, i.e., it is *maximizing* the KL divergence instead of minimizing it. The system works because maximizing the KL divergence corresponds to minimizing the entropy of q, thus finding a stable representation. This is natural considering the nature of the original GS-VAE: The original loss function of the GS-VAE tries to make the latent distribution closer to the fair random Bernoulli distribution $p(\mathbf{b})$ that takes 0 or 1 with the equal probability, i.e., *as random as possible*, which is, in fact, opposite from the concept of stability. Instead, Latplan has a negated loss, which resulted in maximizing the KL divergence and making the representation *less random*.

The resulting loss function implemented in Latplan is therefore as follows:

$$\langle \text{rec loss} \rangle - \text{KL}(q||p)$$
$$= \langle \text{rec loss} \rangle + \text{KL}(q||p) - 2\text{KL}(q||p)$$
$$= \langle \text{the original GS VAE loss} \rangle + 2H_q(\mathbf{b} \mid \mathbf{x}). \quad (4)$$

Since the entropy measures the randomness of the random variables, the extra entropy term in Eq. (4) regularizes the network by penalizing the unstable representation. In the later sections, we empirically show that the original loss function (Eq. (3)) for GS-VAE results in a much higher instability compared to the GS-VAE with the Entropy regularization (Eq. (4)).

Removing the Run-Time Stochasticity Another improvement we made from the original approach in Latplan is that we can disable the stochasticity of the network while performing the planning. After the training is finished, we replace the Gumbel-softmax activation with a pure argmax of class probabilities, which makes the network fully deterministic:

$$z_i = \textbf{if } (i \text{ is } \arg\max_i(\log \pi_i)) \textbf{ then } 1 \text{ else } 0.$$

This technique reduces the inherent stochasticity of the network.

5 Zero-Suppressed State AutoEncoder

In Latplan, Entropy Regularization was the key to address the symbol stability (see Sec. 6) while it was only accidentally introduced. In this context, a natural next step toward obtaining the more stable symbols is to introduce a new regularization for the propositional representation. We propose

Zero-Suppressed State AutoEncoder (ZSAE), an SAE with an additional regularization designed for the discrete representation that we call *zero-suppression*. Its fundamental idea is to penalize the true propositions in the latent layer so that no propositions unnecessarily flip to true at random while preserving the propositions that are absolutely necessary for maintaining the reconstruction accuracy. The resulting loss is *asymmetric* to a particular label $k = 1$ (true):

$$\langle \text{loss} \rangle = \langle \text{vanilla SAE loss} \rangle + \alpha \sum_n \sum_{k \neq 0} z_{nk}$$

where α is the magnitude of regularization. This formulation takes a general form that also covers the multi-valued SAS+ representation with $k \geq 2$. In principle, this method could also be used for a SAS+ representation, but we focus on the binary representation in this paper.

One additional advantage of the ZSAE is that several neurons are completely deactivated, i.e. they always take the value of zero and can be pruned afterward to reduce the network size, similar to Zero-Suppressed Decision Diagrams (Minato 1993). Unlike traditional NN compression methods (Cheng et al. 2017), it does not suffer from accuracy degradation because the activations are discrete and therefore no additional retraining is required. In the continuous cases, even the minuscule activations could be amplified by the weights and significantly affect the later pipelines of the neural networks. Our method for the discrete representations complements the prior work for the continuous ones.

Assume the propositional layer z_{nk} is connected to the next layer of L neurons by a fully-connected network $h_l = \sigma\left(\sum_{n=1}^{N} \sum_{k \in \{0,1\}} W_{nkl} z_{nk} + B_l\right)$ with weights W_{nkl}, biases B_l, and a nonlinear activation σ. ($1 \leq l \leq L$.) When we assume that $z_{n0} = 1$ and $z_{n1} = 0$ always holds ($b_n = 0$ for all inputs), we can prune the index n by adding W_{n0l} to B_l and removing W_{nkl} for $\forall k \in \{0, 1\}, \forall l \in \{1, ...L\}$, which removes $2L$ float values for each zero-suppressed bit b_n. Therefore, if we remove ΔN bits from the latent space, it removes $2L\Delta N$ float values from the network. We can similarly prune the weights W' from the previous layer of M neurons.

Implementation Note. Regardless of α, the regularization tends to be too strong near the beginning of the training. In practice, we set $\alpha = 0$ until 1/3 of the total epochs. We confirmed that gradually increasing α also works, but we did not use it in the later experiments.

6 Empirical Evaluation

We evaluated various SAE implementations across 5 different image domains depicting 8-puzzles or Lights Out puzzle game (Wikipedia 2018). Each training takes at most 30 minutes.

MNIST 8-puzzle 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). 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 **Scrambled Photograph 8-puzzle (Mandrill, Spider)** cuts and scrambles real photographs, similar to the puzzles sold

in stores). **LightsOut** (Wikipedia 2018) is a game where a grid of lights is in some on/off configuration (+: On), and pressing a light toggles its state as well as the states of its neighbors. The goal is all lights Off. **Twisted LightsOut** distorts the original LightsOut game image by a swirl effect.

6.1 The Quality of the Latent Representation

We compare the quality of the latent representation produced by the ZSAE and the vanilla SAE.

State Variance The first metric we evaluated is the bit-wise variance of the state encoding for the same/similar input, which directly measures the stability of the representation. We trained several SAEs for each domain with the different latent layer sizes (numbers of propositions) N and then evaluated the variance. In all experiments below, we randomly generated 100 images with a domain-specific generator for each puzzle domain, then encoded each of them with the SAE 100 times. We measured the variance of the propositions, i.e. the variance of latent activations (0 or 1) across 100 encoding trials of the same image. We then took the mean of the variances over the entire propositions.

We evaluated three versions of the SAE: (1) NG-SAE, an SAE trained with the original GS-VAE loss function as discussed in Sec. 4, and (2) Vanilla SAE in the original paper of Latplan (Asai and Fukunaga 2018) and the Github source code, (3) Zero-Suppressed SAE (ZSAE).

The first thing we tested is to replace the Gumbel-softmax activation with a deterministic argmax function after the training (Sec. 4). In all SAEs, this reduced the variance to 0 for a single input because all networks become deterministic. We omit the results due to space because it is rather obvious. In the following experiments, we always replace the activation function with argmax during testing.

We next measured the variance of SAEs in a noisy setting, where we perturbed the input image by Gaussian noise for each of the 100 trials of the same image. Table 1 (first columns) indicates that the propositions made by NG-SAE are highly random, while the entropy regularization in the vanilla SAE suppresses the stochastic behavior to some extent. ZSAE further reduces the variance and achieves the most stable representation. The effect of SAE→ZSAE was typically 2-3 and up to 4 orders of magnitude ($2.5e - 4 \rightarrow 4.5e - 8$), much stronger than that of Entropy Regularization (NGSAE→SAE, 1 to 2 orders of magnitude). Due to the poor performance of NG-SAE, we do not study it any further in the later experiments.

Effective Size of the Representation Next, Table 1 (middle columns) shows that the number of effective bits, i.e. the number of propositions that *ever* change their values over all states, is low in ZSAE, showing that ZSAE obtained a more compressed, compact representation of the input. In MNIST, the numbers are comparable between ZSAEs with N=100,1000, which shows that the network is able to find an encoding of almost the same size regardless of the size of the latent layer (upper bound of the size of propositions), reducing the need for hyperparameter tuning. In LightsOut and Twisted, ZSAE even finds the 16bit optimal representation for the 4x4 light grids.

6.2 Output Accuracy

Regularization in general works by restricting the neural network to achieve some desirable property, e.g., most commonly for suppressing the overfitting (Goodfellow, Bengio, and Courville 2016, chap.5), but in our case for improving the stability. Thus, as a result of the restricted expressive capacity, the output accuracy may be degraded when the regularization is too strong.

Thus we next tested if the zero-suppression affects the output accuracy. In Table 1 (right columns) we show the Mean Square Error between the input and the output for 100 randomly generated images. The results indicate that the zero-suppression does not significantly affect the output accuracy for N=100,1000. However, for N=36 (a parameter tuned for vanilla SAEs to have the least variance), the zero-suppression harmed the accuracy because the network is already small and the further penalty affected the training. In other words, it is better to combine ZSAE with an overcapacity network, since ZSAE then automatically compresses the representation.

Hyperparameter Sensitivity We tested the sensitivity of ZSAE to its new hyperparameter α (Table 2). The purpose of this experiment is to show that ZSAEs are less sensitive to the choice of both N and α, while vanilla SAEs are sensitive to N. We compared the state variances under noise between the vanilla SAE and the ZSAE with α =0.2,0.5,0.7, N=100,1000. We also added N=36 as the best case for the vanilla SAE. We observed that the ZSAEs achieve the comparable state variance with various α and N, while the vanilla SAEs are significantly affected by N. For instance, the variance for the vanilla SAE with N=100 is up to two orders of magnitude larger than that for the SAE with N=36. In contrast, the worst case for the ZSAEs is 1.1e-4 on Twisted with (N, α)=(100,0.2), which is still better than the vanilla SAE with the same N. They also have similar reconstruction loss (MSE) and the effective bits. Therefore, we conclude that while ZSAE introduces an additional parameter, the selection of N and α is easy compared to the selection of N in SAE.

6.3 Planner Performance

Next, we compared the success ratio of Latplan with various parameters. We tested both AMA$_1$ and AMA$_2$ proposed in (Asai and Fukunaga 2018) as the Action Model Acquisition (AMA) methods. Each domain has 60 problem instances each generated by a random walk from the goal state. 60 instances consist of 30 instances generated by 7-steps random walks and another 30 by 14 steps. 30 instances consist of 10 instances whose images are corrupted by Gaussian noise, 10 with salt/pepper noise and another 10 without noise. It is important to note that *the noiseless instances do not have the external stochasticity*, one of two sources of instability.

We first tested AMA$_1$, an oracular, idealistic AMA that does not incorporate machine learning, and instead generates the entire propositional state transitions from the entire image transitions. The purpose we test an impractical AMA$_1$ method is to separate the effect of a better state representation achieved by ZSAE and that of the learning pro-

	Mean variance over bits (with noisy images)						Effective bits					Mean Square Error (MSE)					
$N=$	100			1000			100		1000		Optimal	100		1000		36	
domain	NG-SAE	SAE	ZSAE	NG-SAE	SAE	ZSAE	SAE	ZSAE	SAE	ZSAE	Encoding	SAE	ZSAE	SAE	ZSAE	SAE	ZSAE
MNIST	8.4e-2	8.6e-3	**3.7e-6**	5.3e-2	2.2e-4	**1.1e-7**	100	51	1000	68	18.4	<1e-4	<1e-4	<1e-4	<1e-4	<1e-4	9.1e-3
Mandrill	1.1e-3	8.3e-4	**3.0e-5**	4.2e-4	2.5e-4	**4.5e-8**	100	46	1000	182	18.4	3.0e-4	2.2e-4	2.1e-4	2.3e-4	2.0e-4	3.2e-4
Spider	8.5e-4	4.9e-4	**6.3e-6**	2.3e-4	4.2e-4	**7.3e-7**	100	49	1000	200	18.4	2.7e-4	2.2e-4	3.1e-4	2.8e-4	<1e-4	2.8e-2
L-Out	9.0e-3	2.0e-4	**3.9e-6**	8.1e-3	1.4e-4	**7.5e-6**	100	**16**	1000	66	16	<1e-4	<1e-4	<1e-4	<1e-4	2.9e-4	2.8e-4
Twisted	1.0e-2	7.1e-4	**5.1e-6**	1.0e-2	4.5e-4	**1.6e-7**	100	**16**	1000	49	16	<1e-4	<1e-4	<1e-4	<1e-4	<1e-4	5.7e-3

Table 1: **Representation characteristics.** Results comparing the NG-SAE, vanilla SAE and ZSAE (α=0.7). (**Left**) Comparing the representation variance over 100 randomly generated images encoded 100 times with Gaussian noise added each time. (**Middle**) The number of bits that ever turns true when encoding the entire state space. In LightsOut and Twisted, ZSAE(N=100) finds an optimal, 16-bit representation of the 4x4 puzzle. (**Right**) Mean Square Error for the test data.

	Mean variance over bits (with noisy images)							
	SAE			ZSAE(α=0.7)		ZSAE(N=100)		
$N=$	36	100	1000	100	1000	α =0.2	0.5	0.7
MNIST	1.8e-4	8.6e-3	2.2e-4	3.7e-6	1.1e-7	4.5e-7	0.0e+0	3.7e-6
Mandrill	2.9e-4	8.3e-4	2.5e-4	3.0e-5	4.5e-8	2.3e-6	3.4e-6	3.0e-5
Spider	5.8e-6	4.9e-4	4.2e-4	6.3e-6	7.3e-7	4.6e-6	8.3e-6	6.3e-6
L-Out	2.5e-6	2.0e-4	1.4e-4	3.9e-6	7.5e-6	1.1e-6	1.5e-5	3.9e-6
Twisted	2.1e-5	7.1e-4	4.5e-4	5.1e-6	1.6e-7	8.8e-5	4.2e-6	5.1e-6

Table 2: Sensitivity of ZSAE to the hyperparameter α, N compared to that of SAE to the hyperparameter N.

cedure in AMA$_2$ that learns the state transitions and the action rules. As a classical planner, we used FastDownward (Helmert 2004) with A^*, blind heuristics in order to remove the effects of the heuristic functions. We ran the solver with no runtime/memory restrictions.

The results in Table 3 (left) show that ZSAEs have **tripled** the score of SAE ($14 \to 43, 6 \to 48, 6 \to 33$) under external stochasticity (Gaussian noise). Regarding the hyperparameter sensitivity, ZSAE showed a robust performance across N=36,64,100 while SAE achieves a good performance only in a single parameter N=36. Due to the small size of the state space compared to the usual classical planning benchmark domains, the search finishes in a fraction of a second, which means **the failure is due to graph disconnectedness at the initial/goal nodes**. ZSAE tends to fail in N=36 in LightsOut and Twisted because of the higher reconstruction error discussed in the previous subsection.

Next, we compare the planning performance of Z/SAE with AMA$_2$, a NN-based AMA model that learns from the example state transitions. It consists of two networks: (1) Action AutoEncoder (AAE), an autoencoder that learns to cluster the state transitions into a finite number of action labels. It learns to reconstruct the input propositional successor state t in the output $\hat{t}$. Its latent layer represents an action label a, which is a single Gumbel-Softmax-activated unit of A categories, where A is the upper bound of the number of action labels. In addition, every layer of AAE is concatenated with the propositional vector of the current state s as the *secondary* input, which turns the standard AE formulation of $f(t) = a$, $g(a) = \hat{t}$ into $f(t, s) = a$, $g(a, s) = \hat{t}$, where g can be interpreted as a function that *applies* an action a to s and obtains $\hat{t}$. (2) Action Discriminator (AD), a binary classifier modeling the action precondition, which takes (s, t) and returns a boolean. Combining AAE and AD yields a successor function that can be used for graph search algorithms. Networks in AMA$_2$ are trained with the same hyperparameters used in the original paper (Asai and

Fukunaga 2018). Table 3 (right) shows that ZSAEs **doubled** ($29 \to 81, 24 \to 62$) the success ratio over vanilla SAEs under noise. The gap is not as large in the noiseless scenario, but note that the scenario is unrealistic: Virtually all real-world data contain various forms of noise/perturbation.

Finally, we compared the search statistics between SAE+AMA$_2$ and ZSAE+AMA$_2$ in order to measure another harmful effect of the unstable propositions that they **confuse the duplicate detection of search algorithms and increase the search effort**. This effect cannot be measured with AMA$_1$ because it creates PDDL/SAS instances based on the fixed set of input images and both SAE/ZSAE are deterministic (using argmax), therefore the search graph contains a single node for a single image. This is not the case with AMA$_2$ because the NNs generate the successor states on the fly. We measured the number of node expansions (left) and the runtime (right) on the problems successfully solved by both SAE+AMA$_2$ and ZSAE+AMA$_2$ (Fig. 4). In order to gather the sufficient number of data points for comparing the search statistics of SAE and ZSAE, we extended the maximum runtime limit of 1 hour and reduced the input noise so that SAE can solve a sufficient number of problem instances. The parameters for the Gaussian noise and the salt/pepper noise applied to the input are $\sigma = 0.3$ and $p = 0.06$, respectively, compared to $\sigma = 0.6$ and $p = 0.12$ in Table 3. In this setting, the gap between ZSAE and SAE narrowed: ZSAE solved 266 instances and SAE solved 262 instances in total. The plots support our claim that the randomness in the state encoding of vanilla SAE confuses the duplicate detection and increase the search effort. ZSAE resulted in a smaller node expansion in 172 out of 238 instances solved by both SAE and ZSAE, and in a shorter runtime in 191 out of 238 instances.

6.4 Pruning Inactive Nodes from ZSAE

We discuss the amount of memory reduction possible by the pruning on the nodes that have constant activations of 0. As we saw from Table 1, vanilla SAEs do not have such propositions (all bits are effective).

Representing a fully-connected network between two layers of L and N nodes requires $(L + 1)N$ weights (+1 for the bias). In the network we used, both the previous and the succeeding layer of the latent propositional layer have 1000 nodes. In Gumbel-softmax, each proposition corresponds to 2 neurons. Thus, in the case of ZSAE with N=1000 applied to MNIST puzzles, the representation is compressed down to 68 effective bits and the weights are reduced by

	AMA₁ results																		AMA₂ results																	
	Gaussian (σ=0.6)						Salt/Pepper (p=0.12)						No noise						Gaussian (σ=0.6)						Salt/Pepper (p=0.12)						No noise					
	SAE			ZSAE			SAE			ZSAE			SAE			ZSAE			SAE			ZSAE			SAE			ZSAE			SAE			ZSAE		
N =	36	64	100	36	64	100	36	64	100	36	64	100	36	64	100	36	64	100	36	64	100	36	64	100	36	64	100	36	64	100	36	64	100	36	64	100
MNIST	0	0	0	**17**	0	0	17	5	0	**20**	**10**	**17**	20	20	20	20	20	20	8	8	4	**20**	**14**	**10**	18	6	0	**20**	**9**	**17**	18	13	5	**20**	**19**	**18**
Mandrill	**4**	3	0	0	**6**	**5**	16	17	8	**20**	**20**	**20**	20	20	20	20	20	20	**13**	10	3	4	**13**	**12**	18	13	6	**18**	**19**	**19**	18	13	10	**18**	**19**	**20**
Spider	8	3	6	**12**	3	5	20	7	16	**20**	**20**	**20**	20	20	20	20	20	20	13	11	17	**17**	**18**	16	17	10	15	**18**	**20**	**18**	17	14	15	17	**20**	**18**
L-Out	1	0	0	**10**	**19**	**20**	**20**	13	18	12	**19**	**20**	20	20	20	20	20	20	10	0	0	**20**	**20**	**12**	20	20	20	20	20	20	20	19	20	20	20	18
Twisted	1	0	0	**4**	**20**	**3**	17	12	6	12	**20**	**17**	20	20	20	20	20	20	3	0	0	**10**	**16**	**12**	19	17	20	**20**	**20**	15	20	20	20	20	20	15
Total	14	6	6	**43**	**48**	**33**	**90**	54	48	84	**89**	**94**	100	100	100	100	100	100	47	29	24	**71**	**81**	**62**	92	66	61	**96**	**88**	**87**	93	79	70	**95**	**98**	**89**
Total (ZSAE, α=0.2)				21	10	2				75	88	68				100	100	99				46	29	33				76	72	71				76	86	81
Total (ZSAE, α=0.5)				15	33	12				77	94	89				100	100	100				36	41	27				56	81	65				59	92	72

Table 3: **Planning Results.** (**Left**) The number of instances successfully solved by Latplan using AMA₁ (oracular method) for comparing the performance of Z/SAE. Better results among the same configuration of ZSAE/SAE are highlighted in **bold**. SAEs degrade performance as the surplus capacity produces more unstable propositions and is better than ZSAE only when tuned to N=36. ZSAEs are robust on the different N and tend to solve more problems than the vanilla SAEs. (**Right**) The numbers of instances solved under 180 sec using AMA₂ unsupervised learning method for Action Model Acquisition. Results indicate that ZSAEs are robust on the different hyperparameters and tend to achieve better performance than vanilla SAEs.
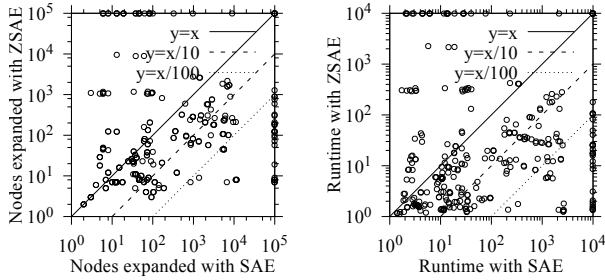

Figure 4: Double-logarithmic plot of the number of states expanded (left) / time spent (right) by A^* with goal-count heuristics, for instances successfully solved under 1 hour by both SAE+AMA₂ and ZSAE(α=0.7)+AMA₂, both with N=100, all domains. In both figures, x-axes represent the SAE and y-axes represent the ZSAE. Unsolved instances are shown on the borders. We observe that the search effort tends to be larger with the SAE than with the ZSAE. Opposite cases are generated due to the tie-breaking difference and the random disconnectedness caused by the SAE.

$(1000 + 1) \times (2 \cdot 1000 - 2 \cdot 68)$=1865864 for the previous layer, and $((2 \cdot 1000 + 1) - (2 \cdot 68 + 1)) \times 1000$=1864000 for the succeeding layer. This number is huge compared to the convolutional weights (3x3, 16 channels, thus 144 float values each) in the upper layers. The total number of weights in the network is reduced from 8376278 to 5578414 (44% reduction). We do not show the entire results because the results are straightforward: The numbers can be similarly calculated from the numbers on Table 1.

7 Related Work

The proposed method has some similarities to the common technique called ℓ_1 regularization (Goodfellow, Bengio, and Courville 2016, Sparse AE), which is applied to the continuous activations of the neurons or the training weights and achieves a sparse representation where the continuous values become closer to zero. ℓ_1 regularization is applied to all neurons in the same layer while the proposed zero-suppression is applied to a subset of neurons representing a particular discrete value. Moreover, regularization is traditionally applied in order to suppress overfitting and improve predictive performance while our focus is rather on the stability of the representation.

A possible interpretation of this approach is that this regularization is working as a model prior corresponding to *closed-world assumption* (Reiter 1981, CWA), which assumes that all propositions are false when they are unknown to a Knowledge Base (KB) or cannot be proven from it. With CWA, KBs no longer have to explicitly store the false propositions, just as the ZSAE can prune the constant-0 nodes.

The search space generated by Latplan was shown to be compatible to an existing Goal Recognition system (Amado et al. 2018a; 2018b). Another recent approach replacing SAE/AMA₂ with InfoGAN (Kurutach et al. 2018) has no explicit mechanism for improving the stability of the binary representation. Other methods for generating a symbolic model from the environment (Yang, Wu, and Jiang 2007; Cresswell, McCluskey, and West 2013; Mourão et al. 2012) require symbolic or near-symbolic, structured inputs. Konidaris et al. (2018) generates a symbolic state space from the low-level sensor inputs but requires the high-level action symbols. Our approach complements the usability of these approaches by providing more stable symbols.

Previous work in Learning from Observation (Barbu, Narayanaswamy, and Siskind 2010; Kaiser 2012), which could produce propositions from the observations (unstructured input) with the help of hand-coded symbol extractors (e.g. *ellipse detectors* for tic-tac-toe), typically assume a perfect indoor environment with little noisy interference, therefore do not have to address the stability of the symbols.

8 Conclusion

In this paper, we provided a formal analysis of the State AutoEncoder (SAE) neural network (NN) in Latplan (Asai and Fukunaga 2018) to understand its success and the issue of its *unstable propositions*. In the analysis, we identified that the SAE accidentally uses a different formulation of Gumbel-Softmax VAE (Jang, Gu, and Poole 2017) that we named *Entropy Regularization* which stabilizes the truth values.

To further improve the stability, we introduced Zero-Suppressed State AutoEncoder (ZSAE) which improves the vanilla SAE by minimizing the number of true propositions in the representation. ZSAE improves the success rate and the efficiency of planning performed on the generated state

space and also removes the need for aggressive hyperparameter tuning. Moreover, ZSAE makes it possible to prune some neurons without accuracy degradation when their activations are constantly zero, similar to Zero-Suppressed Decision Diagrams (Minato 1993) and the knowledge bases with closed world assumption (Reiter 1981).

As a meta-level contribution, we generalized the problematic behavior of the unstable propositions into a *Symbol Stability Problem* (SSP), a subproblem of symbol grounding. We identified two sources of stochasticity which can introduce the instability: The inherent stochasticity of the network and the external stochasticity from the observations. This suggests that SSP is an important problem that applies to any modern NN-based symbol grounding process that operates on the noisy real-world inputs and performs a sampling-based, stochastic process (e.g. VAEs, GANs) that are gaining popularity in the literature. Thus, characterizing the aspect of SSP would help the process of designing a planning system operated on the real world input. An interesting avenue for future work is to extend our approach to InfoGAN-based discrete representation of the environment (Kurutach et al. 2018).

References

Amado, L.; Pereira, R. F.; Aires, J.; Magnaguagno, M.; Granada, R.; and Meneguzzi, F. 2018a. Goal Recognition in Latent Space. In *Proceedings of the International Joint Conference on Neural Networks*.

Amado, L.; Pereira, R. F.; Aires, J.; Magnaguagno, M.; Granada, R.; and Meneguzzi, F. 2018b. LSTM-based Goal Recognition in Latent Space. *arXiv preprint arXiv:1808.05249*.

Asai, M., and Fukunaga, A. 2018. Classical Planning in Deep Latent Space: Bridging the Subsymbolic-Symbolic Boundary. In *Proceedings of AAAI Conference on Artificial Intelligence*.

Barbu, A.; Narayanaswamy, S.; and Siskind, J. M. 2010. Learning Physically-Instantiated Game Play through Visual Observation. In *Proceedings of IEEE International Conference on Robotics and Automaton (ICRA)*, 1879–1886.

Cheng, Y.; Wang, D.; Zhou, P.; and Zhang, T. 2017. A Survey of Model Compression and Acceleration for Deep Neural Networks. *IEEE Signal Processing Magazine, to appear*.

Cresswell, S.; McCluskey, T. L.; and West, M. M. 2013. Acquiring planning domain models using *LOCM*. *Knowledge Eng. Review* 28(2):195–213.

Goodfellow, I.; Bengio, Y.; and Courville, A. 2016. *Deep Learning*. MIT Press. www.deeplearningbook.org.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, 2672–2680.

Gumbel, E. J., and Lieblein, J. 1954. Statistical theory of extreme values and some practical applications: A series of lectures.

Harnad, S. 1990. The Symbol Grounding Problem. *Physica D: Nonlinear Phenomena* 42(1-3):335–346.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *Systems Science and Cybernetics, IEEE Transactions on* 4(2):100–107.

Helmert, M., and Domshlak, C. 2009. Landmarks, Critical Paths and Abstractions: What's the Difference Anyway? In *Proceedings of the International Conference on Automated Planning and Scheduling(ICAPS)*.

Helmert, M. 2004. A Planning Heuristic Based on Causal Graph Analysis. In *Proceedings of the International Conference on Automated Planning and Scheduling(ICAPS)*, 161–170.

Higgins, I.; Matthey, L.; Pal, A.; Burgess, C.; Glorot, X.; Botvinick, M.; Mohamed, S.; and Lerchner, A. 2017. β-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In *Proceedings of the International Conference on Learning Representations*.

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

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

Jang, E.; Gu, S.; and Poole, B. 2017. Categorical Reparameterization with Gumbel-Softmax. In *Proceedings of the International Conference on Learning Representations*.

Kaiser, L. 2012. Learning Games from Videos Guided by Descriptive Complexity. In *Proceedings of AAAI Conference on Artificial Intelligence*.

Kingma, D. P., and Welling, M. 2013. Auto-Encoding Variational Bayes. In *Proceedings of the International Conference on Learning Representations*.

Konidaris, G.; Kaelbling, L. P.; and Lozano-Pérez, T. 2018. From skills to symbols: Learning symbolic representations for abstract high-level planning. *J. Artif. Intell. Res.* 61:215–289.

Kurutach, T.; Tamar, A.; Yang, G.; Russell, S.; and Abbeel, P. 2018. Learning Plannable Representations with Causal InfoGAN. In *In Proceedings of ICML / IJCAI / AAMAS 2018 Workshop on Planning and Learning (PAL-18)*.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-Based Learning Applied to Document Recognition. *Proc. of the IEEE* 86(11):2278–2324.

Maddison, C. J.; Tarlow, D.; and Minka, T. 2014. A* sampling. In *Advances in Neural Information Processing Systems*, 3086–3094.

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

Minato, S.-i. 1993. Zero-suppressed BDDs for Set Manipulation in Combinatorial Problems. In *Proceedings of the 30th international Design Automation Conference*, 272–277. ACM.

Mourão, K.; Zettlemoyer, L. S.; Petrick, R. P. A.; and Steedman, M. 2012. Learning STRIPS Operators from Noisy and Incomplete Observations. In *Proceedings of the International Conference on Uncertainty in Artificial Intelligence*, 614–623.

Newell, A., and Simon, H. A. 1976. Computer Science as Empirical Inquiry: Symbols and Search. *Commun. ACM* 19(3):113–126.

Reiter, R. 1981. On Closed World Data Bases. In *Readings in artificial intelligence*. Elsevier. 119–140.

Steels, L. 2008. The Symbol Grounding Problem has been Solved. So What's Next? In de Vega, M.; Glenberg, A.; and Graesser, A., eds., *Symbols and Embodiment*. Oxford University Press.

Wikipedia. 2018. Lights Out (game). en.wikipedia.org/w/index.php?title=Lights_Out_(game).

Yang, Q.; Wu, K.; and Jiang, Y. 2007. Learning Action Models from Plan Examples using Weighted MAX-SAT. *Artificial Intelligence* 171(2-3):107–143.