Making Sense of Raw Input (Extended Abstract)*

Richard Evans1, Matko Bošnjak1, Lars Buesing1, Kevin Ellis2, David Pfau1, Pushmeet Kohli1 and Marek Sergot3

1DeepMind
2Massachusetts Institute of Technology
3Imperial College London
richardevans@google.com

Abstract

How should a machine intelligence perform unsupervised structure discovery over streams of sensory input? One approach to this problem is to cast it as an apperception task. Here, the task is to construct an explicit interpretable theory that both explains the sensory sequence and also satisfies a set of unity conditions, designed to ensure that the constituents of the theory are connected in a relational structure.

However, the original formulation of the apperception task had one fundamental limitation: it assumed the raw sensory input had already been parsed using a set of discrete categories, so that all the system had to do was receive this already-digested symbolic input, and make sense of it. But what if we don’t have access to pre-parsed input? What if our sensory sequence is raw unprocessed information?

The central contribution of this paper is a neuro-symbolic framework for distilling interpretable theories out of streams of raw, unprocessed sensory experience. First, we extend the definition of the apperception task to include ambiguous (but still symbolic) input: sequences of sets of disjunctions. Next, we use a neural network to map raw sensory input to disjunctive input. Our binary neural network is encoded as a logic program, so the weights of the network and the rules of the theory can be solved jointly as a single SAT problem. This way, we are able to jointly learn how to perceive (mapping raw sensory information to concepts) and apperceive (combining concepts into declarative rules).

1 Introduction

There are, broadly speaking, two approaches to interpreting the results of machine learning systems [Miller, 2018; Rudin, 2018; Murdoch et al., 2019]. In one approach, post-hoc interpretation, we take an existing machine learning sys-

*This paper was originally published in Artificial Intelligence [Evans et al., 2021].
Here, the task is to construct an explicit theory that both explains the sequence and also satisfies a set of unity conditions designed to ensure that the constituents of the theory—the objects, properties, and propositions—are combined together in a relational structure. We developed an implementation, the APPERCEPTION ENGINE, and showed, in a range of experiments, how this system is able to outperform recurrent networks and other baselines on a range of tasks, including Hofstadter’s Seek Whence dataset [Hofstadter, 1995].

But in our initial implementation, there was one fundamental limitation: we assumed the sensory input was provided in symbolic form. We assumed some other system had already parsed the raw sensory input into a set of discrete categories, so that all the APPERCEPTION ENGINE had to do was receive this already-digested symbolic input, and make sense of it. But what if we don’t have access to pre-parsed input? What if our sensory sequence is raw unprocessed information—a sequence of noisy pixel arrays from a video camera, for example?

## 1.2 Overview

Our central contribution is an approach for unsupervised learning of interpretable symbolic theories from raw unprocessed sensory data. We achieve this through a major extension of the APPERCEPTION ENGINE so that it is able to work from this raw input. This involves two phases. First, we extend the APPERCEPTION ENGINE to receive ambiguous (but still symbolic) input: sequences of disjunctions. Second, we use a neural network to map raw sensory input to disjunctive input. Our binary neural network is encoded as a logic program, so the weights of the network and the rules of the theory can be found jointly by solving a single SAT problem. This way, we are able to simultaneously learn how to perceive (mapping raw sensory information to concepts) and apperceive (combining concepts into rules).

We tested our system in three domains. In the first domain, the APPERCEPTION ENGINE learned to solve sequence induction tasks, where the sequence was represented by noisy MNIST images [LeCun et al., 1998]. In the second, it learned the dynamics of Sokoban from a sequence of noisy pixel arrays. In the third, it learned to make sense of sequences of noisy ambiguous data without knowledge of the underlying spatial structure of the generative model.

This system is, to the best of our knowledge, the first system that is able to learn explicit provably correct dynamics of non-trivial games from raw pixel input. We discover that generic inductive biases embedded in our system suffice to induce these game dynamics from very sparse data, i.e. less than two dozen game traces. We see this as a step toward machines that can flexibly adapt and even synthesize their own world models [Ha and Schmidhuber, 2018], starting from raw sub-symbolic input, while organizing and representing those models in a format that humans can comprehend, debug, and verify.

## 2 Experiments

Here, we describe two of the three sets of experiments. For more details, see [Evans et al., 2021].

### 2.1 Seek Whence with Noisy Images

The Seek Whence dataset is a set of challenging sequence induction problems designed by Douglas Hofstadter [Hofstadter, 1995].

#### The Data

In Hofstadter’s original dataset, the sequences are lists of discrete symbols. In our modified dataset, we replaced each discrete symbol with a corresponding MNIST image.

To make it more interesting (and harder), we deliberately chose particularly ambiguous images. Consider Figure 1. Here, the leftmost image could be a 0 or a 2, while the next could be a 5 or possibly a 6. Of course, we humans are unphased by these ambiguities because the low Kolmogorov complexity [Li and Vitányi, 2008] of the high-level symbolic sequence helps us to resolve the ambiguities in the low-level perceptual input. We would like our machines to do the same.

For each sequence, the held-out data used for evaluation is a set of acceptable images, and a set of unacceptable images, for the final held-out time step. See Figure 1. We provide a slice of the sequence as input, and use a held-out time step for evaluation. If the correct symbol at the held-out time step is \(s\), then we sample a set of unambiguous images representing \(s\) for our set of acceptable next images, and we sample a set of unambiguous images representing symbols other than \(s\) for our set of unacceptable images.

#### The Model

In this experiment, we combined the APPERCEPTION EN- GINE with a three-layer perceptron with dropout that had been pre-trained to classify images into ten classes representing the digits 0 – 9. For each image, the network produced a probability distribution over the ten classes.

We chose a threshold (0.1), and stipulated that if the probability of a particular digit exceeded the threshold, then the image possibly represents that digit. According to this threshold, some of the images (the first, third, eighth, and twelfth) of Figure 2 are ambiguous, while others are not.

Our pre-trained neural network MNIST classifier has effectively turned the raw apperception task into a disjunctive apperception task. Once the input has been transformed into a sequence of disjunctions, we apply the APPERCEPTION ENGINE to resolve the disjunctions and find a unified theory that explains the sequence.

### 2.2 Sokoban

In Section 2.1, we used a hybrid architecture where the output of a pre-trained neural network was fed to the APPERCEPTION ENGINE. We assumed that we already knew that the images fell into exactly ten classes (representing the digits 0 – 9), and that we had access to a network that already knew how to classify images.

But what if these assumptions fail? What if we are doing pure unsupervised learning and don’t know how many classes the inputs fall into? What if we want to jointly train the neural network and solve the apperception problem at the same time?

In this next experiment, we combined the APPERCEPTION ENGINE with a neural network, simultaneously learn-
The Data

In this task, the raw input is a sequence of pairs containing a binarised 20×20 image together with a player action from \( \mathcal{A} = \{ \text{north}, \text{east}, \text{south}, \text{west} \} \). In other words, \( \mathcal{R} = \mathbb{B}^{20 \times 20} \times \mathcal{A} \), and \( \{ r_1, \ldots, r_T \} \) is a sequence of (image, action) pairs from \( \mathcal{R} \).

Each array is generated from a 4×4 grid of 5×5 sprites. Each sprite is rendered using a certain amount of noise (random pixel flipping), and so each 20×20 pixel image contains the accumulated noise from the various noisy sprite renderings.

Each trajectory contains a sequence of (image, action) pairs, plus held-out data for evaluation. Because of the noisy sprite rendering process, there are many possible acceptable pixel arrays for the final held-out time step. These acceptable pixel arrays were generated by taking the true underlying symbolic description of the Sokoban state at the held-out time step, and producing many alternative renderings. A set of unacceptable pixel arrays was generated by rendering from various symbolic states distinct from the true symbolic state. Note that the acceptable and unacceptable images are used only for evaluation, not for training. Figure 3 shows an example.

In our evaluation, a model is considered accurate if it accepts every acceptable pixel array at the held-out time step, and rejects every unacceptable pixel array. This is a stringent test. We do not give partial scores for getting some of the predictions correct.

The Model

In outline, we convert the raw input sequence into a disjunctive input sequence by imposing a grid on the pixel array and repeatedly applying a binary neural network to each sprite in the grid.

Figure 4 shows the best theory found by the Apperception Engine from one trajectory of 17 time steps. When neural network next-step predictors are applied to these sequences, the learned dynamics typically fail to generalise correctly to different-sized worlds or worlds with a differ-

\[^1\] Sokoban is a puzzle game where the player controls a man who moves around a two-dimensional grid world, pushing blocks onto designated target squares.
ent number of objects [Buesing et al., 2018]. But the theory leaned by the APPERCEPTION ENGINE applies to all Sokoban worlds, no matter how large, no matter how many objects. Not only is this learned theory correct, but it is provably correct

References


