Large Scale Deep Learning for Theorem Proving in HOList: First Results and Future Directions

Sarah Loos

Theorem proving in large theories comes with unique challenges compared to other tasks on which reinforcement learning has been applied successfully: unlimited action space, sparse reward, and quickly growing knowledge base. Here, I present our approaches to deal with these difficulties and our first practical results on the HOList benchmarks. Our particular baseline solution is named DeepHOL and builds upon the HOList infrastructure and APIs. The action space is unlimited in our context, as some tactics may take an arbitrarily long list of theorems for tactic parameters. Also, newly proved theorems are added to the knowledge base, increasing the complexity of further possible actions. In our baseline approach, we assume that each formula is given as a sequence of a finite number of tokens and these tokens are known beforehand. Our tokens correspond to the tokenization produced by HOL Light, but we communicate formulas in a simple S-expression format to make it easy to process and interpret them. Although DeepHOL ignores the tree structure and relies on sequence-based models, we expect more sophisticated machine learning models to exploit this structure for further improvements. A further simplification is that we assume a relatively small, fixed set of possible tactic applications. However, these simplifications (finite and fixed set of tokens and actions) are not assumed by the HOList system in general.

We present a detailed description of our SearchGraph architecture and how DeepHOL interacts with it. The nodes of the SearchGraph are goals/subgoals, and edges track tactic applications and resulting subgoals. Any node of the SearchGraph can be expanded and further tried to be proved. Also, the SearchGraph automatically merges identical goals, which prevents unsound cyclic proof attempts and other inefficient cyclic behavior. DeepHOL performs proof search in a breadth-first manner, but omits expanding those subgoals that have no chance to contribute to closing the main goal.

Our system relies on a two-tower, two-headed policy network that combines a standard classification model with a ranking model. The classifier head predicts the tactic to be applied, while the ranking head is for ranking premises for their usefulness as arguments passed to the tactic application. The two towers of the network are trained for encoding the goal and premises; these encodings are further processed by a ranking network taking them as inputs. The tactic prediction head only uses the encoding produced by the goal tower. Both encoding towers utilize the WaveNet architecture, which is a residual network with dilated convolutions. While the application of the tactic prediction head is straightforward, we cache the premise encodings in order to make the evaluation ranking model faster: we need to process all the preceding

premises in the database, which can have over ten thousand statements in the multivariate complex analysis corpus.

The reward is very sparse because it takes several minutes to find proofs, and a lot of time is spent in the harder theorems while not learning from unsuccessful proof search traces. So, we adopt a slow-feedback strategy that is highly distributed: we use two thousand workers running the proof search mining for new training examples, while the policy network is trained on a single GPU. In order to decrease the latency of training on newly found examples, we maintain several pools of examples: old, fresh and imitation and we trained on some predefined mixtures of those examples. In order to create the training data for the policy network, we prune the successful proof searches by keeping only those proof-search nodes that were essential for closing the goal. Furthermore, we prune the parameter lists of the tactic applications by keeping only those parameters that are necessary to end up with the same subgoals. In addition, hyperparameters of the proof search (branching factor, theorem list length and maximum unsuccessful tactic applications per node) are randomized to increase the variety of produced proofs. All high ranking theorems that were pruned away for some tactic in a successful proof are stored as hard negatives for their respective goals and are used more frequently as negative examples in the contrastive loss of the premise ranking model.

After comparing the results of our large scale reinforcement learning pipeline with the model trained by imitation learning, we present several ways that our HOList and DeepHOL infrastructure could be utilized for new research.