Deep Network Guided Proof Search

By Sarah Loos, Geoffrey Irving, Christian Szegedy, and Cezary Kaliszyk

Presentation by Rishub Jain

Related Work

- Many other non-deep algorithms for proof search
  - Little use of machine learning
    - Tried to do premise selection and relevance
  - Each problem is very large
  - Algorithm used: **E-prover**
- Many advances in deep neural networks
  - Similar to problems in question and answering and knowledge base completion

- This work combines the two

Challenges

- Deep Learning is slower
  - Thousands of superposition steps may be performed by the prover for every clause that is evaluated by the Deep Network.
  - Thus, need to provide high quality suggestions
  - Can use the DNN to initially guide clause selection, and finish off with a previous prover algorithm
- Need to interleave hard-coded heuristics

Dataset - Mizar First-Order Problems

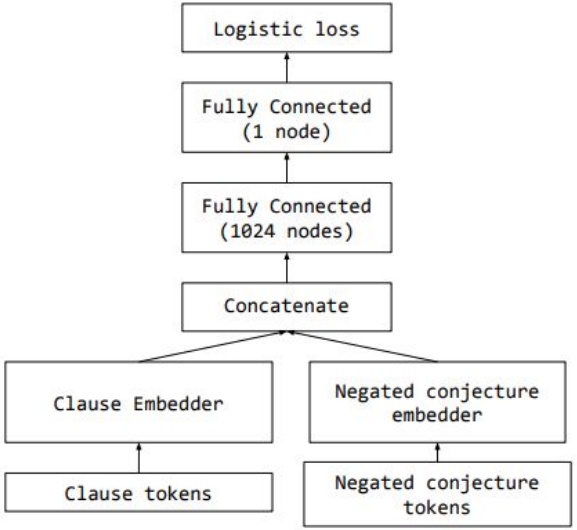
- Mizar Mathematical Library (MML) dataset
- 57,882 Mizar Theorems
  - 32,521 with proofs by ATP methods computed by Mizar system (“easy statements”)
  - 25,361 conjectures unsolved by the system, but solved by humans (“hard statements”)
- 91,877 proofs for the solved theorems
- Separated by conjecture to avoid contamination
  - 29,325 conjectures with 82,718 proofs in training set
  - 3,196 conjectures with 9,159 proofs in test set
- Type of preprocessing (that does clausification and converts to CNF) has a big effect on proof representation
  - Consistently used Auto208 configuration for proof generation

The First-Order Logic Prover E

- Preprocesses inputs (the conjecture and possible clauses) into CNF
- Iteratively finds the correct clauses of the proof
  - Maintains a set of **unprocessed** (with initial guesses) and **processed** (initially empty) clauses
  - Selects a “good” clause from unprocessed set and adds it to the processed set
    - Previously most successful method (hybrid heuristics):
      - Evaluating each clause on many heuristics (FIFO, shortest clause)
      - Processes top clause from each ranking, with arbitrary interleaving
    - **This paper’s method:**
      - **Use Deep Learning to select clause**
  - Adds the selected clause’s consequences to unprocessed set
  - Repeat until proof is found

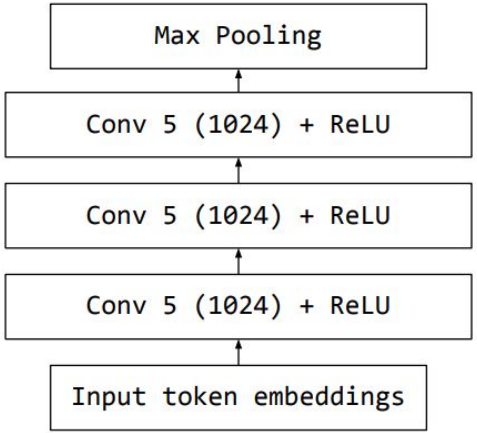
Method

- Given a clause and a conjecture, evaluate clause
  - Should also depend on already processed clauses, but not included for simplicity and speed
- Embed clause and negated conjecture
  - CNN
  - WaveNet (dilated convolutions)
  - RNN (Tree-LSTM and MLP)
- Concatenate embeddings
- Use 1-layer MLP with 1024 hidden units to predict probability that clause is part of the proof

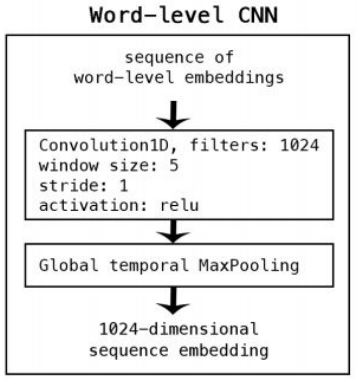


CNN Embedding

- Simple and shallow because “they give good results on that related task”
- Avoid character-level embedding because of large clauses



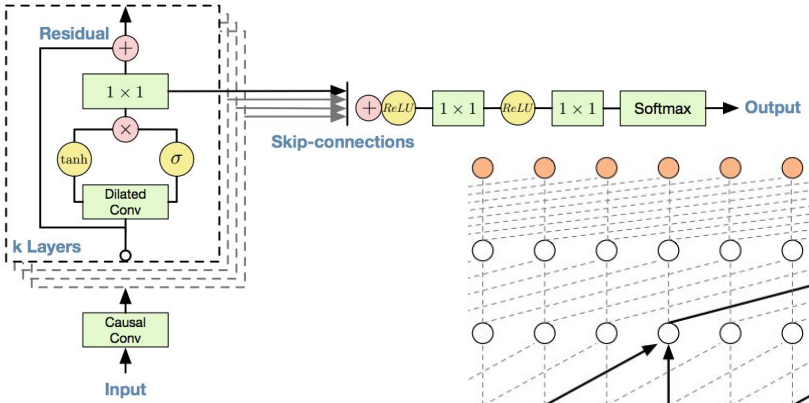
- Actual token embeddings of symbols (constants, function names, variable names, logical operations, and parentheses) unclear, but likely learns word-embeddings from DeepMath:

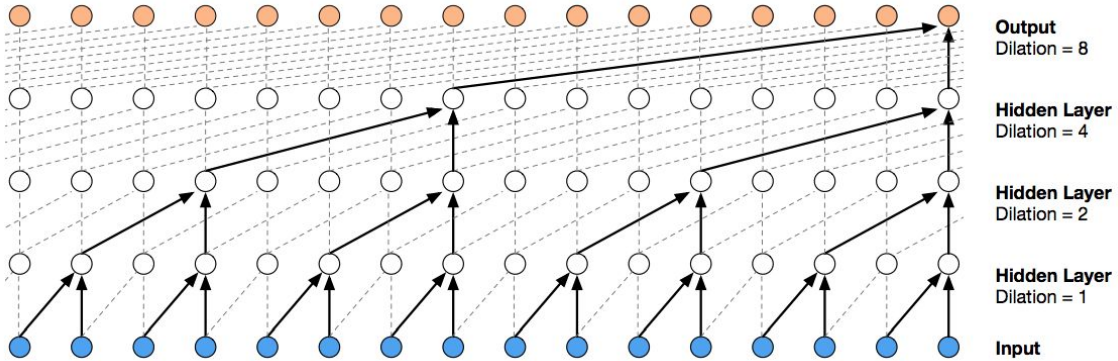


WaveNet Embedding

- CNN with **dilated convolutions** and **residual connections**
- 3 blocks of 7 layers dilated by d=1,2,4,...,64
- Uses same token embedding as CNN

$$f_{emb}(x) = (B \circ B \circ B)(D_t(x,p))$$
$$B(x) = x + (L_{64} \circ \cdots \circ L_2 \circ L_1)(D_f(x,p))$$
$$L_d(x) = x + \tanh(C_d(x))\sigma(C'_d(x))$$
$$C_d(x)_i = b + \sum_{j=1}^{s} w_j x_{i-d(j-\lceil s/2 \rceil)}$$





RNN Embedding - Tree Construction

- Create a parse tree for logic formula
- Leaves are learned variable name embeddings
- Each internal node is one of 4 functions:
  - `apply` nodes that apply a function with exactly two children.
  - `or` nodes that compute the embedding for the disjunction of exactly two children.
  - `and` nodes that compute the embedding for the conjunction of exactly two children. This is used only for embedding the negated conjecture, since the proof clauses do not contain conjunctions.
  - `not` nodes that compute the embedding for the negation of a single child node.
- “not” function seems to be absent from clause embedding calculation

RNN Embedding - Network

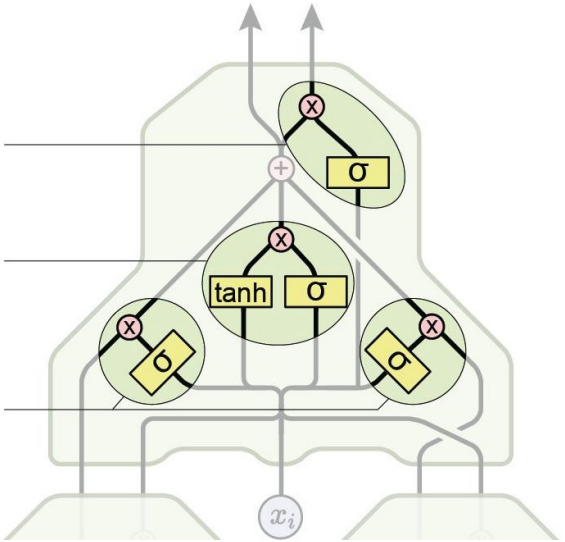
- Represent each type of layer as a Tree-LSTM block or 1-layer MLP
  - Likely all layers are Tree-LSTM blocks except the “not” gate
- Each type of layer shares weights, but different for clause and conjecture embeddings
  - 7 total sets of weight matrices

Tree-LSTM *(Tai, et al., 2015)*

The **output gate** controls which parts of the state are exposed.

A tanh gate proposes values to add to the state, while the **input gate** controls which get added.

Two **forget gates** allow the tree-LSTM to ignore information from subtrees.



Accuracies of Clause-in-Proof Classification

Model	Embedding Size	Accuracy on 50-50% split
Tree-RNN-256×2	256	77.5%
Tree-RNN-512×1	256	78.1%
Tree-LSTM-256×2	256	77.0%
Tree-LSTM-256×3	256	77.0%
Tree-LSTM-512×2	256	77.9%
CNN-1024×3	256	80.3%
*CNN-1024×3	256	78.7%
CNN-1024×3	512	79.7%
CNN-1024×3	1024	79.8%
WaveNet-256×3×7	256	79.9%
*WaveNet-256×3×7	256	79.9%
WaveNet-1024×3×7	1024	81.0%
WaveNet-640×3×7(20%)	640	**81.5%**
*WaveNet-640×3×7(20%)	640	79.9%

Table 1: The accuracy of predicting whether a processed clause was ultimately required for the proof. The accuracy is measured on a 50-50% split of positive and negative processed clause examples, with various recursive deep neural network models. Models with an asterisk (*) were trained on a data set which additionally included a sampling of unprocessed clauses as negative examples. In order to facilitate a direct comparison with other models, the same evaluation dataset was used, but this is slightly biased against the examples denoted with (*).

Proof Generation Evaluation on Easy Statements

- Tried 4 approaches (only using Wavenet and regular CNNs):
- **Baseline Auto heuristic**
  - Not as accurate
- **Pure CNN**
  - Takes a long time
- **Hybrid CNN**, adds the CNN as another heuristic among the other Auto herustics
  - Slightly more accurate, but still takes a long time
- **Two-phase switched approach**, starts with CNN (tried pure and hybrid), and shifts to baseline Auto when time is running out
  - Able to be as accurate, since it doesn't take as long
  - Is the best model (using the hybrid CNN)

Proof Generation Evaluation on Easy Statements

- Evaluated approach using simple CNN, then evaluated different models using that approach
- Two-phase hybrid CNN was the best, because hybrid CNN could not close proof because it was too slow
- Confusing choice of X-axis: time != # processed clauses

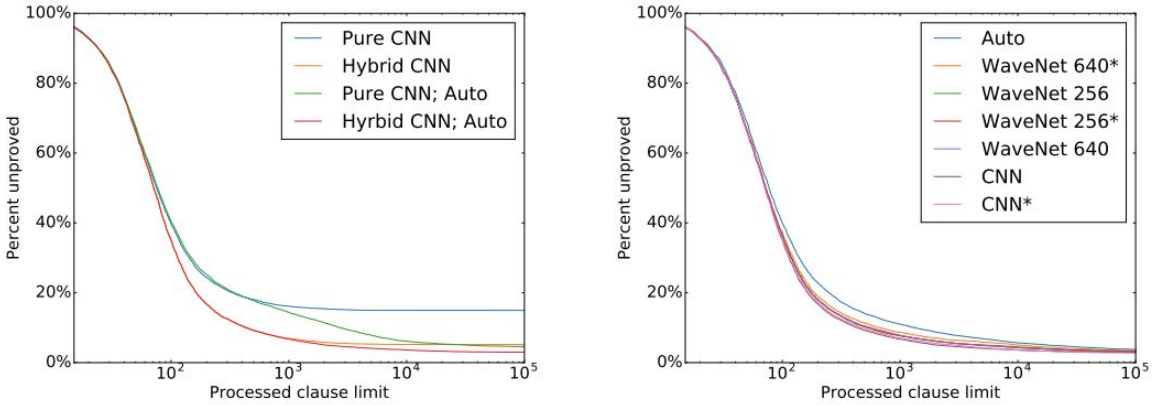


Figure 4: The percentage of unsuccessful proofs at various processed clause limits using different selection heuristics. In the left figure, we use the same network (CNN-1024x3, with 256 embedding) throughout, but show the effect of various interactions with the Auto heuristic. On the right we use hybrid, two-phase guidance, but show the effect of different neural networks. More detailed values are shown in Table 2.

Proof Generation Evaluation on Easy Statements

- Used two-phased approach that runs the hybrid network for 20 min then baseline Auto for 10 min
- Regular CNN worked better for larger processed clauses, possibly because WaveNet took too long

Model	Accuracy	PC≤1,000	PC≤10,000	PC≤100,000	PC<∞
Auto	N/A	89.0%	94.3%	96.2%	96.6%
*WaveNet 640	79.9%	91.4%	95.0%	96.5%	96.6%
WaveNet 256	79.9%	92.3%	95.5%	96.8%	96.8%
*WaveNet 256	79.9%	92.2%	95.7%	96.8%	96.8%
CNN	80.3%	**93.3%**	**96.4%**	97.0%	97.1%
WaveNet 640	**81.5%**	92.2%	95.7%	97.0%	97.2%
*CNN	78.7%	93.0%	**96.4%**	**97.2%**	**97.3%**

Table 2: The percent of theorems proved in the "*easy statements*" with various model architectures. The training accuracy is duplicated from Table 1 for convenience. The values in the PC≤N columns indicate the percent of statements proved requiring fewer than N processed clauses (PC). The far right column (PC<∞) is the percent of statements proved within 30 minutes, with a memory limit of 16G, and no limit on processed clauses.

Proof Generation Evaluation on Hard Statements

- Seemed to use CPUs for DNN forward pass
- Proof generation bottleneck becomes DNN computation
- Table shows it is necessary to have premise selection and two-phase approach

	without premise selection	with premise selection
unguided	145	458
guided (hybrid)	137	383

Table 3: Number of hard theorems proved with various combinations of premise selection and (unswitched) proof guidance. Note that even when our proof guidance is partial, it still produces worse results than the variant without deep network guidance. This is due to the slowness of deep network evaluation. The sole purpose of this table is to highlight the importance of premise selection for the hard statements. In other experiments, we concentrate on the two-phase "switched" approach that combines guided and unguided search in a sequential fashion and outperforms both unguided search and hybrid guidance without the switch.

Proof Generation Evaluation on Hard Statements

- Used character level (for simplicity) CNN premise selection from DeepMath paper
- Different premise selection models have similar performance, but solve different proofs
- Simple CNN worked the best
- Total of 1,866 (7.36%) hard statements proven

Model	DeepMath 1	DeepMath 2	Union of 1 and 2
Auto	578	581	674
*WaveNet 640	644	612	767
*WaveNet 256	692	712	864
WaveNet 640	629	685	997
*CNN	905	812	1,057
CNN	839	935	1,101
Total (unique)	1,451	1,458	1,712

Table 4: Number of theorems proved out of the 25,361 hard theorems, proved with various combinations of premise selection (DeepMath 1 & 2) and clause selection guidance. The last column shows the union of theorems proved with either premise selection step method in the given row. The size of the union of the all theorems proved by methods in this in this table is 1,712 (6.8%). The number of theorems proved by the deep network guided methods is 1,665 (6.6%).

Conclusions

- Can use DNNs to guide proof search algorithms to significantly perform better
- Speed and accuracy are important to increase proving power
  - WaveNet was worse than the CNN
- Two-phase approach is necessary
- Paper uses 30 min instead of the standard 15 min to solve proofs, but extra time only helps if using DNN guidance
- Interesting intersection of domains
  - WaveNet built for generating audio
  - Tree-LSTMs used for semantic tree parsing
    - Used multiple “layers” as opposed to just 1