DeepMath - Deep Sequence Models for Premise Selection

NIPS ID: 1151 *arχiv: 1606.04442*

Alexander A. Alemi
Google Inc.
alemi@google.com

François Chollet
Google Inc.
fchollet@google.com

Niklas Een
Google Inc.
een@google.com

Geoffrey Irving
Google Inc.
geoffreyi@google.com

Christian Szegedy
Google Inc.
szegedy@google.com

Josef Urban
Czech Technical University in Prague
josef.urban@gmail.com

Abstract: We study the effectiveness of neural sequence models for premise selection in automated theorem proving, one of the main bottlenecks in the formalization of mathematics. We propose a two stage approach for this task that yields good results for the premise selection task on the Mizar corpus while avoiding the handengineered features of existing state-of-the-art models. To our knowledge, this is the first time deep learning has been applied to theorem proving.

Overview

- A demonstration for the first time that neural network models are useful for aiding in large scale automated logical reasoning without the need for hand-engineered features.
- The comparison of various network architectures (including convolutional, recurrent and hybrid models) and their effect on premise selection performance.
- A method of sematic-aware "definition"-embeddings for function symbols that improves the generalization of formulas with symbols occuring infrequently. This model outperforms previous approaches at relaxed cutoff-threasholds.
- Analysis that shows that the neural network based premise selections models are complementary to those with hand-eingeered features and can be ensembled with previous results to produce superior results.

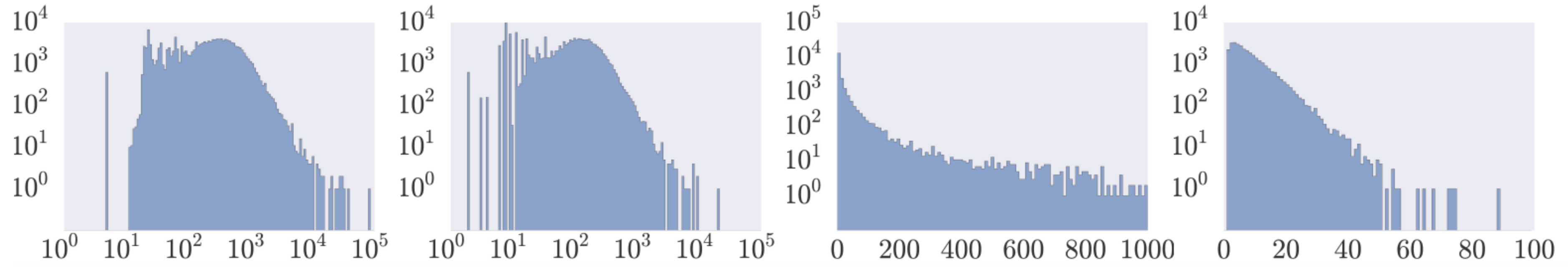
Dataset

The Mizar Mathematical Library (MML) is a corpus of formal mathematical proofs, containing 57,917 theorems from a wide varity of mathematical subjects. We worked with a version converted to first order logic in the TPTP format.

```
:: t99_jordan: Jordan curve theorem in Mizar
for C being Simple_closed_curve holds C is Jordan;

:: Translation to first order logic
fof(t99_jordan, axiom,  (! [A] :  ( (v1_topreal2(A) & m1_subset_1(A,
k1_zfmisc_1(u1_struct_0(k15_euclid(2))))) => v1_jordan1(A)) ) ).
```

Figure 1: (top) The final statement of the Mizar formalization of the Jordan curve theorem. (bottom) The translation to first-order logic, using name mangling to ensure uniqueness across the entire corpus.



(a) Length in chars. (b) Length in words. (c) Word occurrences. (d) Dependencies.

Figure 2: Histograms of statement lengths, occurrences of each word, and statement dependencies in the Mizar corpus translated to first order logic. The wide length distribution poses difficulties for RNN models and batching, and many rarely occurring words make it important to take definitions of words into account.

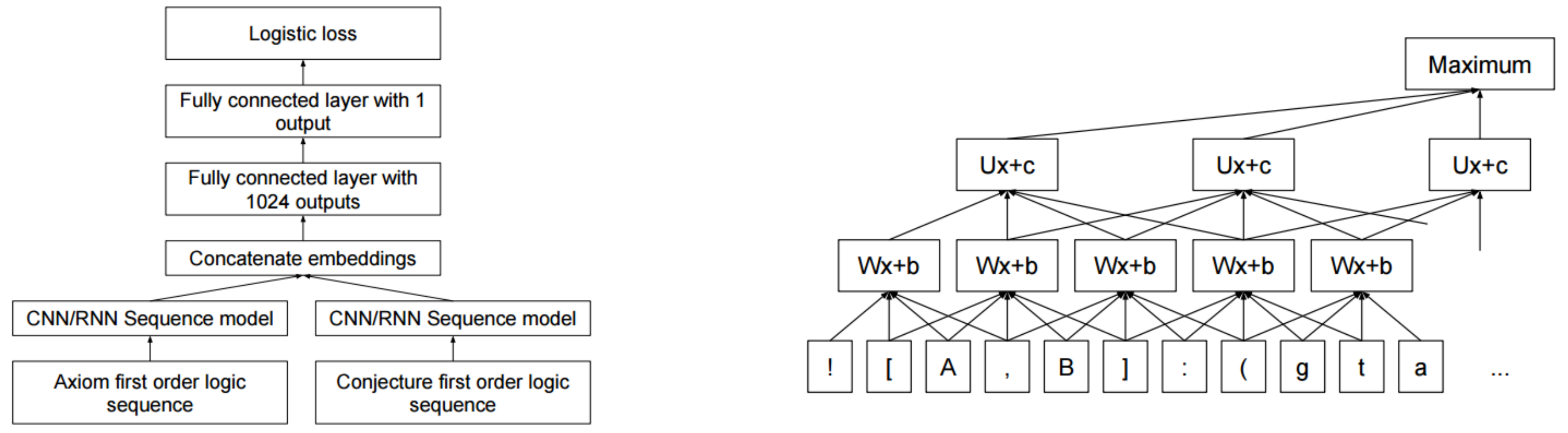
Problem

Definition (Premise selection problem)**.** *Given a large set of premises $\mathcal{P}$, an ATP system A with given resource limits, and a new conjecture C, predict those premises from $\mathcal{P}$ that will most likely lead to an automatically constructed proof of C by A.*

Metric

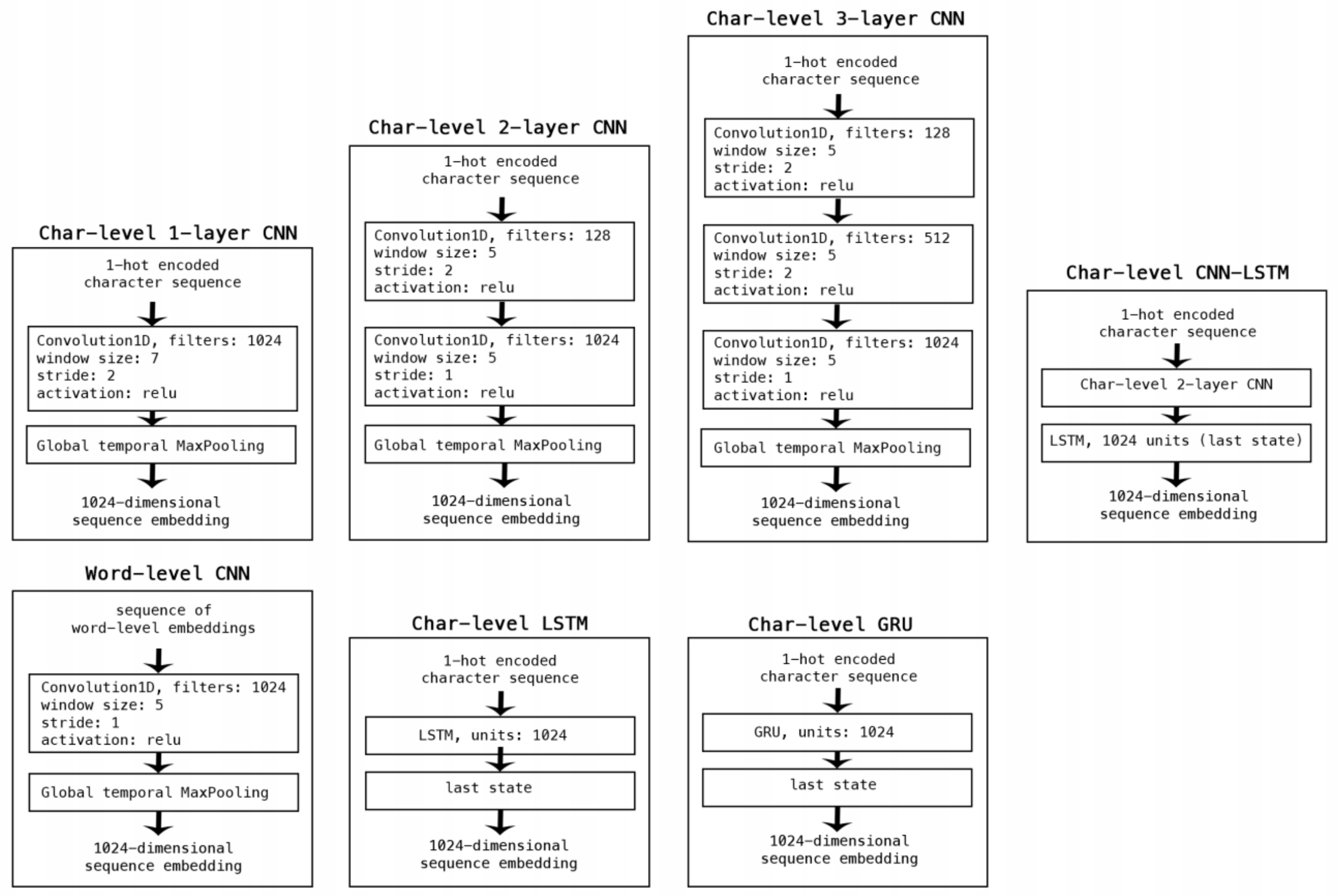
$$\text{aMRR} = \operatorname*{mean}_{C} \max_{P \in \mathcal{P}_{\text{test}}(C)} \frac{\text{rank}(P, \mathcal{P}_{\text{avail}}(C))}{|\mathcal{P}_{\text{avail}}(C)|}$$

Model Architectures



Logistic loss
Fully connected layer with 1 output
Fully connected layer with 1024 outputs
Concatenate embeddings
CNN/RNN Sequence model
CNN/RNN Sequence model
Axiom first order logic sequence
Conjecture first order logic sequence
Maximum
Ux+c
Ux+c
Ux+c
Wx+b
Wx+b
Wx+b
Wx+b
Wx+b
! [A , B] : (g t a ...

Figure 3: (left) Our network structure. The input sequences are either character-level (section 5.1) or word-level (section 5.2). We use separate models to embed conjecture and axiom, and a logistic layer to predict whether the axiom is useful for proving the conjecture. (right) A convolutional model.



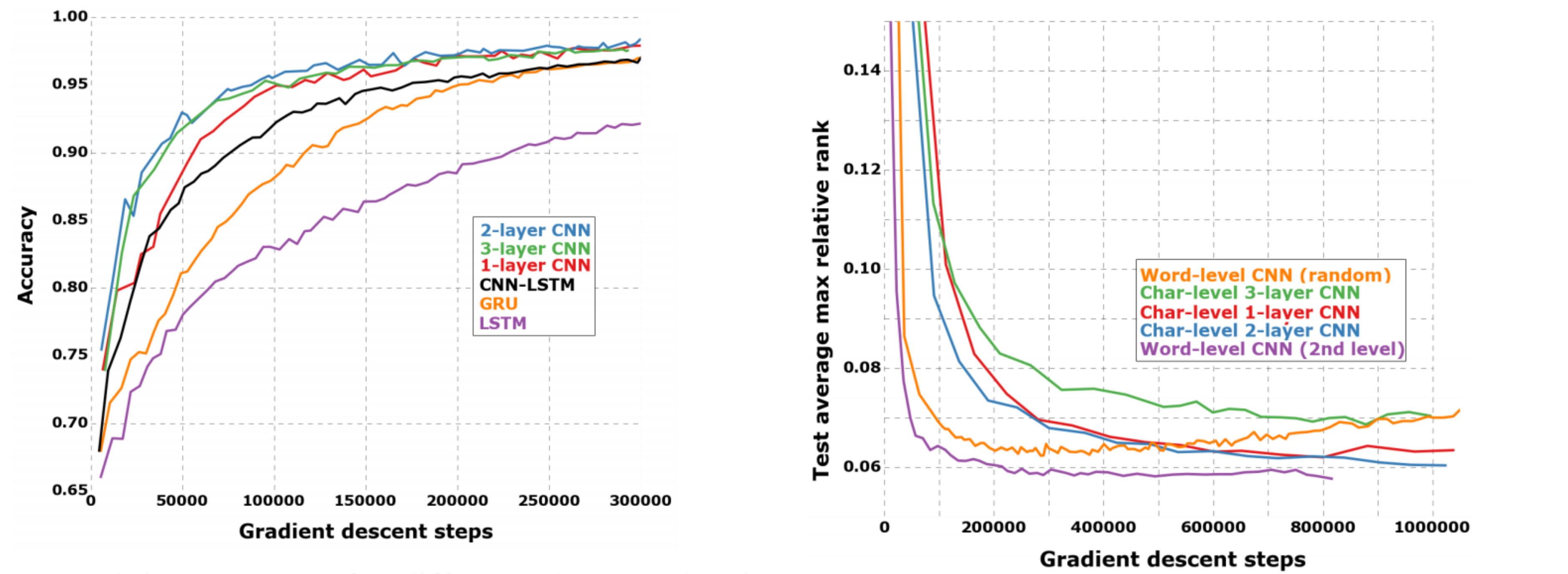
Char-level 1-layer CNN
1-hot encoded character sequence
Convolution1D, filters: 1024 window size: 7 stride: 2 activation: relu
Global temporal MaxPooling
1024-dimensional sequence embedding
Char-level 2-layer CNN
1-hot encoded character sequence
Convolution1D, filters: 128 window size: 5 stride: 2 activation: relu
Convolution1D, filters: 1024 window size: 5 stride: 1 activation: relu
Global temporal MaxPooling
1024-dimensional sequence embedding
Char-level 3-layer CNN
1-hot encoded character sequence
Convolution1D, filters: 128 window size: 5 stride: 2 activation: relu
Convolution1D, filters: 512 window size: 5 stride: 2 activation: relu
Convolution1D, filters: 1024 window size: 5 stride: 1 activation: relu
Global temporal MaxPooling
1024-dimensional sequence embedding
Char-level CNN-LSTM
1-hot encoded character sequence
Char-level 2-layer CNN
LSTM, 1024 units (last state)
1024-dimensional sequence embedding
Word-level CNN
sequence of word-level embeddings
Convolution1D, filters: 1024 window size: 5 stride: 1 activation: relu
Global temporal MaxPooling
1024-dimensional sequence embedding
Char-level LSTM
1-hot encoded character sequence
LSTM, units: 1024
last state
1024-dimensional sequence embedding
Char-level GRU
1-hot encoded character sequence
GRU, units: 1024
last state
1024-dimensional sequence embedding

Figure 4: Specification of the different embedder networks.

Results

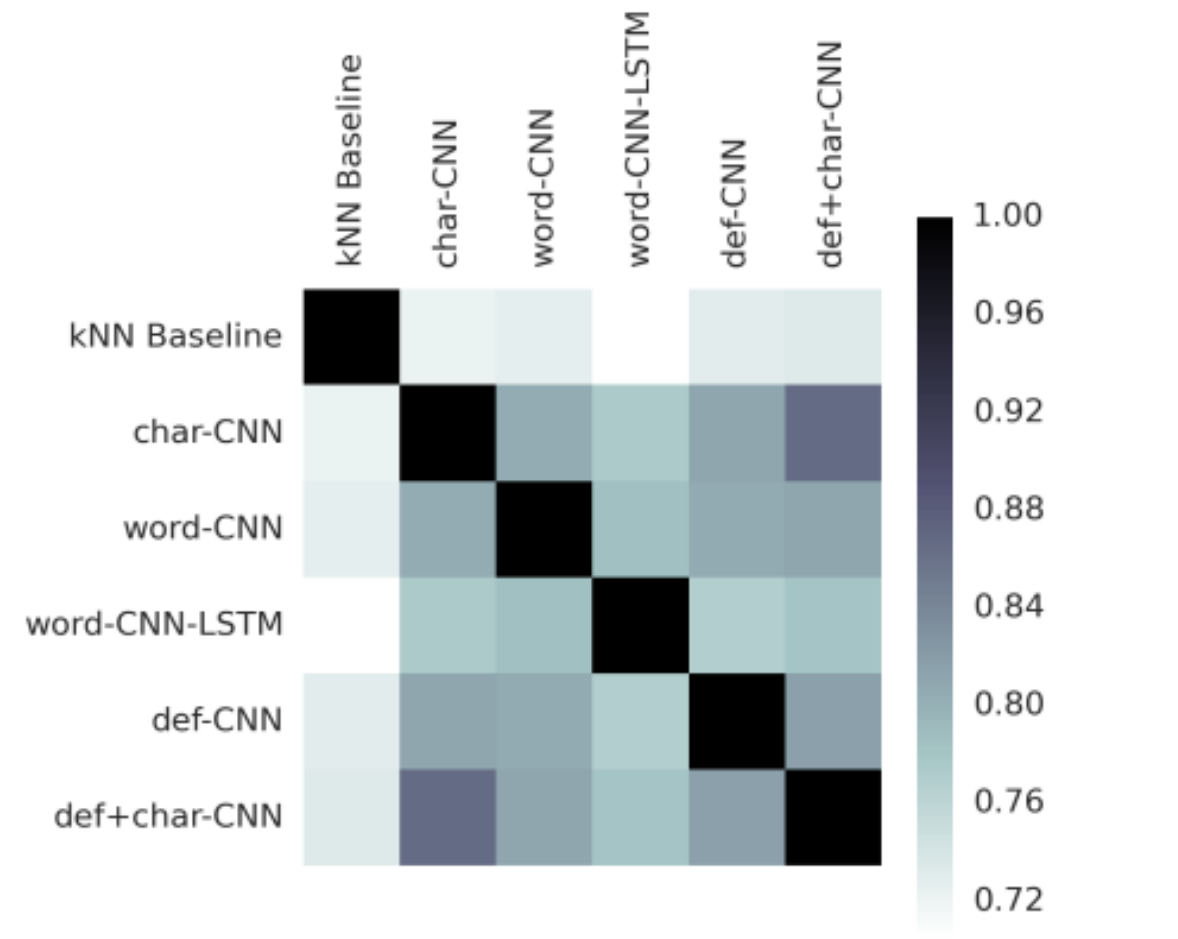
Cutoff	k-NN Baseline (%)	char-CNN (%)	word-CNN (%)	def-CNN-LSTM (%)	def-CNN (%)	def+char-CNN (%)
16	674 (24.6)	687 (25.1)	709 (25.9)	644 (23.5)	734 (26.8)	**835 (30.5)**
32	1081 (39.4)	1028 (37.5)	1063 (38.8)	924 (33.7)	1093 (39.9)	**1218 (44.4)**
64	1399 (51)	1295 (47.2)	1355 (49.4)	1196 (43.6)	1381 (50.4)	**1470 (53.6)**
128	1612 (58.8)	1534 (55.9)	1552 (56.6)	1401 (51.1)	1617 (59)	**1695 (61.8)**
256	1709 (62.3)	1656 (60.4)	1635 (59.6)	1519 (55.4)	1708 (62.3)	**1780 (64.9)**
512	1762 (64.3)	1711 (62.4)	1712 (62.4)	1593 (58.1)	1780 (64.9)	**1830 (66.7)**
1024	1786 (65.1)	1762 (64.3)	1755 (64)	1647 (60.1)	1822 (66.4)	**1862 (67.9)**

Table 1: Results of ATP premise selection experiments with hard negative mining on a test set of 2,742 theorems. Each entry is the number (%) of theorems proved by E prover using that particular model to rank the premises. The union of def-CNN and char-CNN proves 69.8% of the test set, while the union of the def-CNN and k-NN proves 74.25%. This means that the neural network predictions are more complementary to the k-NN predictions than to other neural models. The union of all methods proves 2218 theorems (80.9%) and just the neural models prove 2151 (78.4%).



(a) Training accuracy for different character-level models without hard negative mining. Recurrent models seem underperform, while pure convolutional models yield the best results. For each architecture, we trained three models with different random initialization seeds. Only the best runs are shown on this graph; we did not see much variance between runs on the same architecture.

(b) Test average max relative rank for different models without hard negative mining. The best is a word-level CNN using definition embeddings from a character-level 2-layer CNN. An identical word-embedding model with random starting embedding overfits after only 250,000 iterations and underperforms the best character-level model.



(a) Jaccard similarities between proved sets of conjectures across models. Each of the neural network model prediction are more like each other than those of the k-NN baseline.

Model	Test min average relative rank
char-CNN	0.0585
word-CNN	0.06
def-CNN-LSTM	0.0605
def-CNN	**0.0575**

(b) Best sustained test results obtained by the above models. Lower values are better. This was monitored continuously during training on a holdout set with 400 theorems, using all true positive premises and 128 randomly selected negatives. In this setup, the lowest attainable max average relative rank with perfect predictions is 0.051.