Is Math Only for Humans?

Christian Szegedy
Google Research
What Math?

- Arithmetics on (large) numbers
- Solving large systems of equations
- Complex Symbolic Computer Algebra (e.g. ideal membership in polynomial rings, group theoretic calculations)
- Large scale numeric optimization (e.g. constrained nonlinear programs)
- Approximating PDEs
- Discrete optimization (matching, TSP, coloring, scheduling, SAT solving, etc.)
- ...

...
What Math?

- Arithmetics on (large) numbers
- Solving large systems of equations
- Complex Symbolic Computer Algebra (e.g. ideal membership in polynomial rings, group theoretic calculation)
- Large scale numeric optimization (e.g. constrained nonlinear programs)
- Approximating PDEs
- Discrete optimization (matching, TSP, coloring, scheduling, SAT solving, etc.)
- ... 
- **Creative mathematics: coming up with complex proofs and interesting conjectures.**
Can we create a human-level artificial mathematician?
Success Stories of Deep Learning for AI

- Machine Perception
  - Object recognition, detection and segmentation
  - Generative image models, style transfer
  - Computational photography (e.g. denoising, HDR, super-resolution)
  - Speech recognition

- Game Playing:
  - Alpha-Zero (chess, go, shogi)
  - Alpha-Star (Starcraft)
  - OpenAI-five (Dota-2)

- Text processing:
  - Translation
  - Question answering
  - Summarization

- Ranking and recommendations
Quest for the Artificial Mathematician

**Can we create an artificial mathematician that**

- Creates formally verified proofs (on a small verification kernel)
- Proves complicated non-trivial theorems
- Understands underspecified proofs (closes large gaps)
- Conjectures useful lemmas and new theorems
Quest for the Artificial Mathematician

Challenges:

- Impossibility of self-play (like in go or chess)
- Large, limitlessly growing knowledge base (proved theorems)
- Extremely sparse reward
- Heterogeneous domains
- Large unlimited action space (premises to be used)
- Formalizing statements is a lot of human work (requiring expert skills)
- Lack of data for imitation learning
- Slow evaluation of high level tactics (algorithms)
Impossibility of self-play

Chess, Go, StarCraft, Dota:
- unlimited supply of training data by playing tournaments with artificial agents.

Mathematics:
- Exploration and conjecturing does not have a well defined objective.
  Unguided exploration likely generates lot of uninteresting statements.
- Defining the “Interestingness” of statements is unsolved.
Large Knowledge and Action Space

Chess, Go, StarCraft, Dota:
- Constant (relatively small) number of possible moves at every step

Mathematics:
- At every step we need to select a combination of earlier produced statements: The action space is growing and unbounded.
- Requires managing a growing repository and solving relevant retrieval from it.
- Using existing statements often requires subtle adaptation to the new situation.
Extremely Sparse Reward

Chess, Go, StarCraft, Dota:
- Winner gets reward at the end of each self-play

Mathematics:
- Reward is only given for successful proofs.
- Proofs of even simple statements require thousands to millions of elementary logic steps.
Task Specification

Chess, Go, StarCraft, Dota:
- Each game is an automatically generated new task for the agent.

Mathematics:
- Interesting statements must be specified (formalized by) humans.
- Slow, cumbersome and requires expert skills.
- Finite, limited supply.
Training Data Availability for Imitation Learning

Chess, Go, StarCraft, Dota:
- Large repositories of expert games.
- Millions of professional go games
- Millions of hours of StarCraft gameplay
- Imitation learning was crucial for reaching human level in StarCraft

Mathematics:
- Finite, very limited supply of mathematical proofs (less than 100K theorems).
- Slow, cumbersome and requires a lot of expert skills.
Slow Evaluation of Individual Tactics

Chess, Go, StarCraft, Dota:
- Predictable, close to constant, small cost of actions.

Mathematics:
- Proof assistants rely on various kinds of combinatorial searches that can take indefinite (often long) time.
  - SAT solver
  - Groebner bases
  - Type checking
  - ...

Quest for the Artificial Mathematician

Can we create an artificial mathematician that

- Creates formally verified proofs (on a small verification kernel)
- Proves complicated non-trivial theorems
- Understands underspecified proofs (closes large gaps)
- Conjectures useful lemmas and new theorems
Quest for the Artificial Mathematician

Can we create an artificial mathematician that

- Creates formally verified proofs (on a small verification kernel)
- Proves complicated non-trivial theorems
- Understands underspecified proofs (closes large gaps)
- Conjectures useful lemmas and new theorems
- **Communicates theorems in an interpretable manner**
- Understands informal (natural language) mathematics
Can we create a human level AI to reason about mathematics?

Without relying on informal human mathematics

- No need for autoformalization (requires high level of natural language understanding)
- Need to formalize the notion of “interestingness”.
- User needs to learn an “alien” language just to communicate a theorem to it
- Can’t communicate its discoveries
- May be hard to bootstrap (little training data)

Relying on informal human mathematics

- Needs auto-formalization
- Requires no formalization on user side
- Could learn the human notion of “interestingness”.
- Lot of training data to bootstrap from
Vision of "Autoformalization"

Neural Net Formal Reasoner

Formal Verifier (Formal prover, SMT solver)

(Neural) Language Model

Informal Corpus

Formal Corpus
Mizar Mathematical Library

Corpus of formally verified mathematical statements collected over 44 years. Contains a wide variety of theorems including:

- Cauchy-Riemann Differential Equations of Complex Functions
- Characterization and Existence of Gröbner Bases
- Maximum Network Flow Algorithm by Ford and Fulkerson
- Jordan Curve theorem
- The Sylow Theorems
- Hahn Banach Theorem
- Gauss Lemma and Law of Quadratic Reciprocity
- Key Cryptography and Pepin's Primality Test
First Order vs. Higher Order Logic

First order logic

● Can’t quantify over propositions
● Can’t axiomatize Zermelo-Fraenkel with finite number of axioms
● Have most traditional automation methods

Higher order logic

● Can quantify over any object
● HOL matches human reasoning well.
● Little traditional proof automation
Higher order logic

Highly fragmented space:

- HOL-light: Written in OCaml. Formalization of the Kepler Conjecture, ~29000 formalized statements.
- HOL4: Written in Poly-ML. 8000 top level statements
- Isabelle: Written in ML. Around 10000 statements, mostly computer science.

Other types of higher order logic (non-classical, based on dependent types):
- Coq (OCaml), [Used for formalizing Feit-Thompson], over 20000 statements, constructive logic
- Lean (C++) No statements available
HOL-Light formalization Examples

\~\text{rational}(\sqrt{2})

\!f:\text{real}^N \rightarrow \text{real}^N \ s. \ \text{compact} \ s \ \&\ \text{convex} \ s \ \&\ \sim(s = \{\}) \ \&\ \text{continuous\_on} \ s \ \&\ \text{IMAGE} \ f \ s \ \text{SUBSET} \ s \ \Rightarrow \ ?x. \ x \ \in \ s \ \&\ f \ x = x

\!A. \ \text{msum}(0..\text{dimindex}(:N)) \ (\\lambda i. \ \text{char\_poly} \ A \ i \ \%\% \ A \ \text{mpow} \ i) = \text{mat} \ 0

\!e \ \text{op} \ i \ G \ p. \ \text{group} \ (G, \text{op}, i, e) \ \Rightarrow \ \text{FINITE} \ G \ \Rightarrow \ \text{prime} \ p \ \Rightarrow \ (\!n. \ p \ \text{EXP} \ n \ \text{divides} \ \text{CARD} \ G \ \Rightarrow \ (?H. \ \text{subgroup} \ \text{op} \ i \ H \ G \ \&\ \text{CARD} \ H = p \ \text{EXP} \ n))

\!n. \ \sim(n = 0) \ \Rightarrow \ ?p. \ \text{prime} \ p \ \&\ n \leq p \ \&\ p \leq 2 * n
APIs for Theorem Prover Developers and ML Researchers

(Proof) Assistant

One goal/subgoal to prove
One proof step: Tactic application, relevant premises

Proof Search

Subgoals or *proved*

Ranking of tactics and premises

Formal Reasoning Agent

One goal/subgoal to prove

Machine Learning
Reasoning On the Tactics Level

Advantage: Most HOL-light proofs require only a few tactic invocations (50% of the proofs use less than 8 tactics calls)

Disadvantage: Special purpose tactics are necessary for 30% of the proofs. Really involved statements need special code to look for the proof.

We can cover 70% of the proofs by using the 20 most frequent tactics.
Tactics occurrence statistics in HOL-light corpus
Proof Search Tree
Hybrid Tactic Prediction + Parameter Ranking

Pairwise Scorer

Goal:

```
\text{SUC} \quad n > 0
```

Param:

```
\neg \exists n \quad \forall m. \quad m > n \iff n < m
```

Goal Encoder

WaveNet

Param Encoder

WaveNet

Classifier

Tactic Label

Fully Connected & Regularizer

\{\text{BatchNorm, VIB}\}

Score
http://deephol.org

Link to arxiv paper, GitHub code repository. Training data, neural network model checkpoints, docker images.

Dataset Stats

Core
- 1.5K Theorems
- 10K Theorems
- 375K Human Proof Steps

Complex
- None

Flyspeck
- None

Training 60%
- 1.5K Theorems
- 10K Theorems
- 375K Human Proof Steps

Validation 20%
- 500 Theorems
- 3.2K Theorems
- 100K Human Proof Steps

Testing 20%
- 500 Theorems
- 3.2K Theorems
- 100K Human Proof Steps

- 10.5K Theorems
## Results - Imitation Learning on Human Proofs

<table>
<thead>
<tr>
<th>Model</th>
<th>Percent of Validation Theorems Closed</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Baselines</strong></td>
<td></td>
</tr>
<tr>
<td>ASM_MESON_TAC</td>
<td>6.1%</td>
</tr>
<tr>
<td>ASM_MESON_TAC + WaveNet premise selection</td>
<td>9.2%</td>
</tr>
<tr>
<td><strong>Imitation Learning</strong></td>
<td></td>
</tr>
<tr>
<td>WaveNet</td>
<td>30.1%</td>
</tr>
</tbody>
</table>
Results - Reinforcement Loop

<table>
<thead>
<tr>
<th>Loop Type</th>
<th>Percent Closed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thin WaveNet Loop</td>
<td>36.30%</td>
</tr>
<tr>
<td>- Trained on loop output</td>
<td>36.80%</td>
</tr>
<tr>
<td>Tactic Dependent Loop</td>
<td>38.90%</td>
</tr>
</tbody>
</table>
Most commonly used human tactics:
- REWRITE_TAC
- RAW_POP_TAC
- LABEL_TAC
- MP_TAC
- X_GEN_TAC
Tactics Distribution - Reinforcement Loop

Tactics used in Reinforcement Loop:

- ASM_MESON_TAC
- REWRITE_TAC
- ONCE_REWRITE_TAC
- MP_TAC
- SIMP_TAC
Soundness is Critical

ITPs motivated by concerns around correctness of natural mathematics.

- HOL Light relies on only ~400 trusted lines of code.

You should not need to trust more than that:

- Environment optimizations: startup cheats-ins and proof search code are now in the critical core (!) -- we must have a proof checker.
- Reinforcement learning reinforces soundness problems.
Proof Checker

We provide a proof checker that compiles proof logs into OCaml code

- Human-readable format
- Can be checked with HOL Light’s core

To be sure that the proofs work, the proof checker replaces HOL Light’s built-in proofs by the imported synthetic proofs.

- Same soundness guarantees as HOL Light.
Improving the quality of semantic embeddings

Challenge: We would like to embed statements (natural language or formal) in a way where semantically identical or similar statements end up being close by in the embedding space.
SQuAD 1.1

Stanford Question Answering Dataset 1.1:

23K Paragraphs with over 100K questions in total.

**Task:** Given a paragraph and a question for that paragraph, determine a segment in the paragraph that answers that question

*SQuAD: 100,000+ Questions for Machine Comprehension of Text*
Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev and Percy Liang (EMNLP 2016)
Question Answering on SQuAD 1.1

**Task:** Given a paragraph and a question for that paragraph, determine a segment in the paragraph that answers that question.

<table>
<thead>
<tr>
<th></th>
<th>EM</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human Performance</td>
<td>82.3</td>
<td>91.2</td>
</tr>
<tr>
<td>BERT ensemble (Google)</td>
<td>87.4</td>
<td>93.1</td>
</tr>
</tbody>
</table>

**BERT:** Pre-training of Deep Bidirectional Transformers for Language Understanding
Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova

**SQuAD:** 100,000+ Questions for Machine Comprehension of Text
Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev and Percy Liang (EMNLP 2016)
Retrieval on SQuAD 1.1

**Idea:** Train model for lookup where each of the paragraph is embedded and the task is to look up nearest neighbors.

<table>
<thead>
<tr>
<th>Embedding</th>
<th>Recall@1</th>
</tr>
</thead>
<tbody>
<tr>
<td>ELMO</td>
<td>28%</td>
</tr>
<tr>
<td>ELMO + IDF</td>
<td>32%</td>
</tr>
<tr>
<td>FCRR (on top of ELMO)</td>
<td>42%</td>
</tr>
</tbody>
</table>

**Deep contextualized word representations.**

**Text Embeddings for Retrieval from a Large Knowledge Base**
T. Cakaloglu, C. Szegedy, X. Xu
Future Directions

- **Autoformalization**
  - Using unsupervised embedding mathematical text from images
  - Using the usefulness of side information for more semantic lookup
  - Bootstrap for supervised matching between formal and informal

- **Theorem Proving:**
  - Better search (more control to the neural networks)
  - Creation of terms
  - Forward reasoning
  - Multi-agent systems
  - Unsupervised training on unsuccessful proof traces and other experiments
  - More semantic features of formulas (using experience traces).

- **Conjecturing**
  - Guided exploration of mathematics
  - Generating useful lemmas.
  - Recreation of existing statements based on their semantic embedding
http://deephhol.org

Link to arxiv paper, GitHub code repository.
Training data, neural network model checkpoints, docker images.