First Experiments with Neural Translation of Informal to Formal Mathematics

Qingxiang Wang[1,2]⋆, Cezary Kaliszyk[1]⋆[0000-0002-8273-6059], and Josef Urban[2]⋆⋆

[1] University of Innsbruck
[2] Czech Technical University in Prague

Abstract. We report on our experiments to train deep neural networks that automatically translate informalized LaTeX-written Mizar texts into the formal Mizar language. To the best of our knowledge, this is the first time when neural networks have been adopted in the formalization of mathematics. Using Luong et al.'s neural machine translation model (NMT), we tested our aligned informal-formal corpora against various hyperparameters and evaluated their results. Our experiments show that our best performing model configurations are able to generate correct Mizar statements on 65.73% of the inference data, with the union of all models covering 79.17%. These results indicate that formalization through artificial neural network is a promising approach for automated formalization of mathematics. We present several case studies to illustrate our results.

1 Introduction: Autoformalization

In this paper we describe our experiments with training an end-to-end translation of LaTeX-written mathematical texts to a formal and verifiable mathematical language – in this case the Mizar language. This is the next step in our *project to automatically learn formal understanding* [12,13,11] of mathematics and exact sciences using large corpora of alignments between informal and formal statements. Such machine learning and statistical translation methods can additionally integrate strong semantic filtering methods such as type-checking and large-theory Automated Theorem Proving (ATP) [4,23].

Since there are currently no large corpora that would align many pairs of human-written informal LaTeX formulas with their corresponding formalization, we obtain the first corpus for the experiments presented here by *informalization* [12]. This is in general a process in which a formal text is turned into (more) informal one. In our previous work over Flyspeck and Mizar [12,11] the main informalization method was to forget which overloaded variants and types of the mathematical symbols are used in the formal parses. Here, we additionally

⋆ Supported by ERC grant no. 714034 *SMART*

⋆⋆ Supported by the *AI4REASON* ERC Consolidator grant number 649043, and by the Czech project AI&Reasoning CZ.02.1.01/0.0/0.0/15_003/0000466 and the European Regional Development Fund.

use a nontrivial transformation of Mizar to LaTeX that has been developed over two decades by Grzegorz Bancerek [3,1] for presenting and publishing the Mizar articles in the journal Formalized Mathematics.[3]

Previously [12,11], we have built and trained on the smaller aligned corpora custom translation systems based on probabilistic grammars, enhanced with semantic pruning methods such as type-checking. Here we experiment with state-of-the-art *artificial neural networks*. It has been shown recently that given enough data, neural architectures can learn to a high degree the syntactic correspondence between two languages [20]. We are interested to see to what extent the neural methods can achieve meaningful translation by training on aligned informal-formal pairs of mathematical statements. The neural machine translation (NMT) architecture that we use is Luong et al.'s implementation [15] of the sequence-to-sequence (seq2seq) model.

We will start explaining our ideas by first providing a self-contained introduction to neural translation and the seq2seq model in Section 2. Section 3 explains how the large corpus of aligned Mizar-LaTeX formulas is created. Section 4 discusses preprocessing steps and application of NMT to our data, and Section 5 provides an exhaustive evaluation of the neural methods available in NMT. Our main result is that when trained on about 1 million aligned Mizar-LaTeX pairs, the best method achieves perfect translation on 65.73% of about 100 thousand testing pairs. Section 7 concludes and discusses the research directions opened by this work.

2 Neural Translation

Function approximation through artificial neural network has existed in the literature since 1940s [7]. Theoretical results in late 80-90s have shown that it is possible to approximate an arbitrary measurable function by layers of compositions of linear and nonlinear mappings, with the nonlinear mappings satisfying certain mild properties [6,10]. However, before 2010s due to limitation of computational power and lack of large training datasets, neural networks generally did not perform as well as alternative methods.

Situation changed in early 2010s when the first GPU-trained convolutional neural network outperformed all rival methods in an image classification contest [14], in which a large labeled image dataset was used as training data. Since then we have witnessed an enormous amount of successful applications of neural networks, culminating in 2016 when a professional Go player was defeated by a neural network-enabled Go-playing system [16].

Over the years many variants of neural network architectures have been invented, and easy-to-use neural frameworks have been built. We are particularly interested in the sequence-to-sequence (seq2seq) architectures [20,5] which have achieved tremendous successes in natural language translation as well as related tasks. In particular, we have chosen Luong's NMT framework [15] that encapsulates the Tensorflow API gracefully and the hyperparameters of the seq2seq

[3] `https://www.degruyter.com/view/j/forma`

model are clearly exposed at command-line level. This allows us to quickly and systematically experiment with our data.

2.1 The Seq2seq Model

A seq2seq model is a network which consists of an encoder and a decoder (the left and right part in Fig. 1). During training, the encoder takes in a sentence one word at a time from the source language, and the decoder takes in the corresponding sentence from the target language. The network generates another target sentence and a loss function is computed based on the input target sentence and the generated target sentence. As each word in a sentence will be embedded into the network as a real vector, the whole network can be considered as a complicated function from a finite-dimensional real vector space to the reals. Training of the neural network amounts to conducting optimization based on this function.

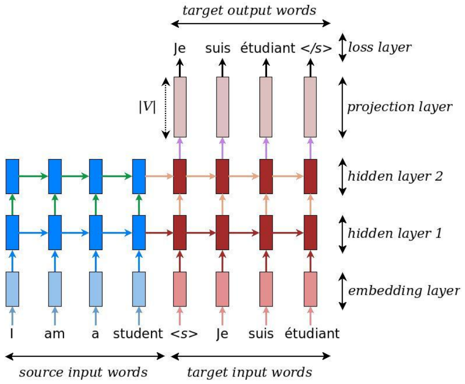

Fig. 1: Seq2seq model (adapted from Luong et al. [15])

When the training is complete, the neural network can be used to generate translations by *inferring* from (translating of) unseen source sentences. During inference, only the source sentence is provided. A target sentence is then generated word after word from the decoder by conducting greedy evaluation with the probabilistic model represented by the trained neural network (Fig. 2).

2.2 RNN and the RNN Memory Cell

The architectures of the encoder and the decoder inside the seq2seq model are similar, each of which consists of multiple layers of *recurrent neural networks* (RNNs). A typical RNN consists of one memory cell, which takes input word tokens (in vector format) and updates its parameters iteratively. An RNN cell is typically presented in literature in the *rolled-out format* (Fig. 3), though the

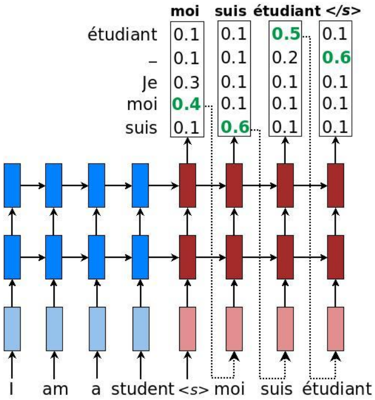

Fig. 2: Inference of seq2seq model (adapted from Luong et al. [15])

same memory cell is used and the same set of parameters are being updated during training.

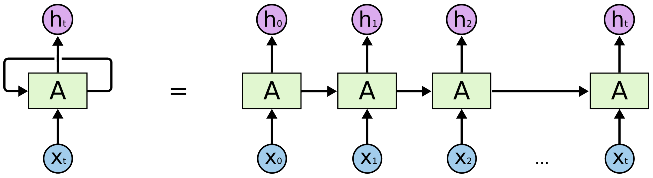

Fig. 3: RNN cell and its rolled-out format (adapted from Olah's blog [18])

Inside each memory cell there is an intertwined combination of linear and nonlinear transformations (Fig. 4). These transformations are carefully chosen to mimic the cognitive process of keeping, retaining and forgetting information.

Only differentiable functions are used to compose the memory cell, so the overall computation is also a differentiable function and gradient-based optimization can be adopted. In addition, the computation is designed in so that the derivative of the memory cell's output with respect to its input is always close to one. This ensures that the problem of vanishing or exploding gradients is avoided when conducting differentiation using the chain rule. Several variants of memory cells exist. The most common are the *long short-term memory* (LSTM) in Fig. 4 and the *gated recurrent unit* (GRU), in Fig. 5.

2.3 Attention Mechanism

The current seq2seq model has a limitation: in the first iteration the decoder obtains all the information from the encoder, which is unnecessary as not all parts of the source sentence contribute equally to particular parts of the target

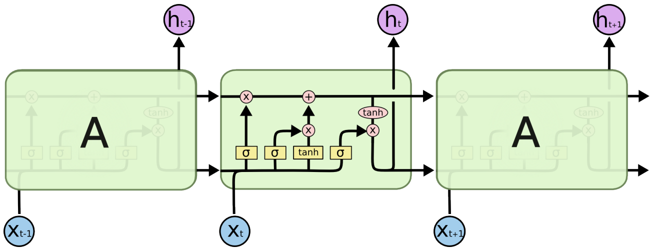

Fig. 4: Close-up look of an LSTM cell (adapted from Olah's blog [18])

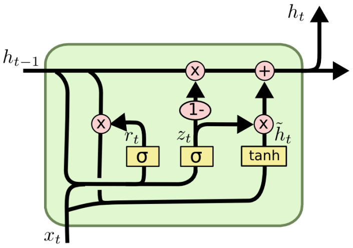

Fig. 5: Gated recurrent unit (adapted from Olah's blog [18])

sentence. The *attention* mechanism is used to overcome this limitation by adding special layers in parallel to the decoder (Fig. 6). These special layers compute scores which can provide a weighting mechanism to let the decoder decide how much emphasis should be put on certain parts of the source sentence when translating a certain part of the target sentence. There are also several variants of the attention mechanism, depending on how the scores are computed or how the input and output are used. In our experiments, we will explore all the attention mechanisms provided by the NMT framework and evaluate their performance on the Mizar-LaTeX dataset.

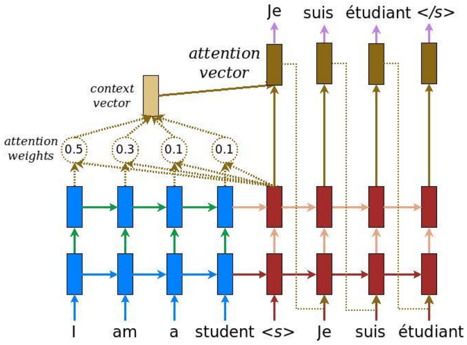

Fig. 6: Neural network with attention mechanism (adapted from Luong et al. [15])

3 The Informalized Dataset

State-of-the-art neural translation methods generally require large corpora consisting of many pairs of aligned sentences (e.g. in German and English). The lack of aligned data in our case has been a bottleneck preventing experiments with end-to-end neural translation from informal to formal mathematics. The approach that we have used so far for experimenting with non-neural translation methods is to take a large formal corpus such as Flyspeck [9] or Mizar [2] and apply various *informalization* (*ambiguation*) [12,11] transformations to the formal sentences to obtain their less formal counterparts. Such transformations include e.g. forgetting which overloaded variants and types of the mathematical symbols are used in the formal parses, forgetting of explicit casting functors, bracketing, etc. These transformations result in more human-like and ambiguous sentences that (in particular in the case of Mizar) resemble the natural language style input to ITP systems, but the sentences typically do not include more complicated symbol transformations that occur naturally in LaTeX.

There are several formalizations such as Flyspeck, the Coq proof of the Odd-Order theorem, the Mizar formalization of the Compendium of Continuous Lattices (CCL) that come with a high-level alignment of the main theorems in the corresponding (LaTeX-written) books to the main formalized theorems. However, such mappings are so far quite sparse: e.g., there are about 500 alignments between Flyspeck and its informal book [8]. Instead, we have decided to obtain the first larger corpus of aligned LaTeX/formal sentences again by informalization. Our requirement is that the informalization should be nontrivial, i.e., it should target a reasonably rich subset of LaTeX and the transformations should go beyond simple symbol replacements.

The choice that we eventually made is to use the Mizar translation to LaTeX. This translation has been developed for more than two decades by the Mizar team [21] and specifically by Grzegorz Bancerek [3,1] for presenting and publishing the Mizar articles in the journal Formal Mathematics. This translation is relatively nontrivial [1]. It starts with user-defined translation patterns for different basic objects of the Mizar logic: functors, predicates, and type constructors such as adjectives, modes and structures. Quite complicated mechanisms are also used to decide on the use of brackets, the uses of singular/plural cases, regrouping of conjunctive formulas, etc. Tables 1 and 2 show examples of this translation for theorems `XBOOLE_1:1`[4] and `BHSP_2:3`[5], together with their tokenized form used for the neural training and translation.

Since Bancerek's technology is only able to translate Mizar formal abstracts into Latex, in order to obtain the maximum amount of data, we modified the latest experimental Mizar-to-LaTeX XSL stylesheets that include the option to produce all the proof statements. During the translation of a Mizar article we track for every proof-internal formula its starting position (line and column) in the corresponding Mizar article, marking the formulas with these positions in

[4] http://grid01.ciirc.cvut.cz/~mptp/7.13.01_4.181.1147/html/xboole_1#T1

[5] http://grid01.ciirc.cvut.cz/~mptp/7.13.01_4.181.1147/html/bhsp_2#T13

Rendered LaTeX	If $X \subseteq Y \subseteq Z$, then $X \subseteq Z$.
Mizar	`X c= Y & Y c= Z implies X c= Z;`
Tokenized Mizar	`X c= Y & Y c= Z implies X c= Z ;`
LaTeX	`If $X \subseteq Y \subseteq Z$, then $X \subseteq Z$.`
Tokenized LaTeX	`If $ X \subseteq Y \subseteq Z $ , then $ X \subseteq Z $ .`

Table 1. Theorem 1 in XBOOLE_1

Rendered LaTeX	Suppose s_8 is convergent and s_7 is convergent . Then $\lim(s_8+s_7) = \lim s_8 + \lim s_7$
Mizar	`seq1 is convergent & seq2 is convergent implies lim(seq1 +seq2)=(lim seq1)+(lim seq2);`
Tokenized Mizar	`seq1 is convergent & seq2 is convergent implies lim ( seq1 + seq2 ) = ( lim seq1 ) + ( lim seq2 ) ;`
LaTeX	`Suppose ${s_{8}}$ is convergent and ${s_{7}}$ is convergent. Then $\mathop{\rm lim}({s_{8}}{+}{s_{7}}) \mathrel{=}\mathop{\rm lim}{s_{8}}{+} \mathop{\rm lim}{s_{7}}$`
Tokenized LaTeX	`Suppose $ { s _ { 8 } } $ is convergent and $ { s _ { 7 } } $ is convergent . Then $ \mathop { \rm lim } ( { s _ { 8 } } { + } { s _ { 7 } } ) \mathrel { = } \mathop { \rm lim } { s _ { 8 } } { + } \mathop { \rm lim } { s _ { 7 } } $`

Table 2. Theorem 3 in BHSP_2

the generated LaTeX file. We then extract each formula tagged with its position P from the LaTeX file, align it with the Mizar formulas starting at position P, and apply further data processing to them (Section 4.1). This results in about one million aligned pairs of LaTeX/Mizar sentences.

4 Applying Neural Translation to Mizar

4.1 Data Preprocessing

To adapt our data to NMT, the LaTeX sentences and their corresponding Mizar sentences must be properly tokenized (Table 1 and 2). In addition, distinct word tokens from both LaTeX and Mizar must also be provided as vocabulary files.

In Mizar formulas, tokens can be and often are concatenated – as e.g. in `n<m`. We used each article's symbol and identifier files produced by the Mizar accommodator and parser to separate such tokens. For LaTeX sentences, we decided to consider dollar signs, brackets, parentheses, carets and underscores as

separate tokens. We keep tags starting with backslash intact and leave all the font information (e.g. romanization or emphasis). Cross-referencing tags, styles for itemization as well as other typesetting information are removed.

4.2 Division of Data

Luong's NMT model requires a small set of development data and test data in addition to training data. To conduct the full training-inference process the raw data needs to be divided into four parts. Our preprocessed data contains 1,056,478 pairs of Mizar-LaTeX sentences. In order to achieve a 90:10 training-to-inference ratio we randomly divide our data into the following:

- 947,231 pairs of sentences of training data.
- 2,000 pairs of development data (for NMT model selection).
- 2,000 pairs of test data (for NMT model evaluation).
- 105,247 pairs of inference (testing) data.
- 7,820 and 16,793 unique word tokens generated for the vocabulary files of LaTeX and Mizar sentences, respectively.

For our partition, there are 57,145 lines of common latex sentences in both the training set and the inference set, making up to 54.3% of the inference set. This is expected as mathematical proofs involve a lot of common basic proof steps. Therefore, in addition to correct translations, we are also interested in correct translations in the 48,102 non-overlapping sentences.

4.3 Choosing Hyperparameters

Luong's NMT model provides around 70 configurable hyperparameters, many of which can affect the architecture of the neural network and in turn affect the training results. In our experiments, we decided to evaluate our model with respect to the following 7 hyperparameters that are the most relevant to the behavior of the seq2seq model (Table 4), while keeping other hyperparameters (those that are more auxiliary, experimental or non-recommended for change) at their default. Selected common hyperparameters are listed in Table 3.

Name	Default Value
Number of training steps	12,000
Learning rate	1.0 (0.001 when using Adam optimizer)
Forget bias for LSTM cell	1.0
Dropout rate	0.2
Batch size	128
Decoding type	greedy

Table 3. Common network hyperparameters across experiments

Name	Description	Value
unit type	Type of the memory cell in RNN	LSTM (default) GRU Layer-norm LSTM
attention	The attention mechanism	No Attention (default) (Normed) Bahdanau (Scaled) Luong
nr. of layers	RNN layers in encoder and decoder	2 layers (default) 3/4/5/6 layers
residual	Enables residual layers (to overcome exploding/vanishing gradients)	False (default) True
optimizer	The gradient-based optimization method	SGD (default) Adam
encoder type	Type of encoding methods for input sentences	Unidirectional (default) Bidirectional
nr. of units	The dimension of parameters in a memory cell	128 (default) 256/512/1024/2048

Table 4. Hyperparameters for seq2seq model

5 Evaluation

The results are evaluated by four different metrics: 1) perplexity; 2) the BLEU rate of the final test data set; 3) the number and percentage of identical statements within all the 105,247 inference sentences and 4) the number and percentage of identical statements within the 48,102 non-overlapping inference sentences. Perplexity measures the difficulty of generating correct words in a sentence, and the BLEU rate gives a score on the quality of the overall translation. Details explaining perplexity and the BLEU rate can be found in [17] and [19], respectively. Due to the abundance of hyperparameters, we decided to do our experiments progressively, by first comparing a few basic hyperparameters, fixing the best choices and then comparing the other hyperparameters. The basic hyperparameters we chose are the type of memory cell and the attention mechanism.

5.1 Choosing the Best Memory Cell and Attention Mechanism

From Table 5 we can see that GRU and LSTM perform similarly and both perform better than Layer-normed LSTM. As LSTM performed slightly better than GRU we fixed our memory cell to be LSTM for further experiments. [6]

Published NMT evaluations show that the attention mechanism results in better performance in translation tasks. Our experiments confirm this fact and

[6] Since training and inference involve randomness, the final results are not identical across trials, though our experience showed that the variation of the inference metrics are small.

also show that the Normed Bahdanu attention, Luong attention and Scaled Luong attention are better than Bahdanau attention (Table 6). Among them we picked the best-performing Scaled Luong attention as our new default and used this attention for our further experiments.

Parameter	Final Test Perplexity	Final Test BLEU	Identical Statements (%)	Identical No-overlap (%)
LSTM	**3.06**	**41.1**	**40121 (38.12%)**	**6458 (13.43%)**
GRU	3.39	34.7	37758 (35.88%)	5566 (11.57%)
Layer-norm LSTM	11.35	0.4	11200 (10.64%)	1 (0%)

Table 5. Evaluation on type of memory cell (attention not enabled)

Parameter	Final Test Perplexity	Final Test BLEU	Identical Statements (%)	Identical No-overlap (%)
No Attention	3.06	41.1	40121 (38.12%)	6458 (13.43%)
Bahdanau	3	40.9	44218 (42.01%)	8440 (17.55%)
Normed Bahdanau	1.92	63.5	60192 (57.19%)	18057 (37.54%)
Luong	**1.89**	64.8	60151 (57.15%)	18013 (37.45%)
Scaled Luong	2.13	**65**	**60703 (57.68%)**	**18105 (37.64%)**

Table 6. Evaluation on type of attention mechanism (LSTM cell)

5.2 The Effect of Optimizers, Residuals and Encodings with respect to Layers

After fixing the memory cell and the attention mechanism, we tried the effects of the optimizer types and of the encoding mechanisms on our data with respect to the number of the RNN layers. We also experiment with enabling the residual layers. The results are shown in Table 7. We can observe that:

1. For RNN because of the vanishing gradient problem the result generally deteriorates when the number of layers becomes higher. Our experiments confirm this: the best-performing architecture has 3-layers.
2. Residuals can be used to alleviate the effect of vanishing gradients. We see from Table 7 that the results are generally better with residual layers enabled, though there are cases when residuals produce failures in training.
3. The NaN values are caused by the overflow of the optimization metric (bleu rate). For some hyperparameter combinations, it happens that the metric will get worse as training progresses, which ultimately leads to overflow and

subsequent early stop of the training phase. Our experiments show that this overflow reappears with respect to multiple times of trainings.

4. It is interesting that the Adam optimizer, bidirectional encoding and combinations of them can also alleviate the effect of vanishing gradients.
5. The Adam optimizer performs generally better than the SGD optimizer and Bidirectional encoding performs better with less layers.[7]
6. The number of layers seems to matter less in our model than other parameters such as optimizers and encoding mechanisms, though it is notable that the more layers the longer the training time.
7. The number of identical non-overlapping statements is generally proportional to the total number of identical statements.

5.3 The Effect of the Number of Units and the Final Result

We now train our models by fixing other hyperparameters and variating the number of units. Our results in Table 8 show that performance generally gets better until 1024 units. The performance decreases when the number of units reaches 2048, which might indicate that the model starts to overfit. We have so far only used CPU versions of Tensorflow. The training real times in hours of our multi-core Xeon E5-2690 v4 2.60GHz servers with 28 hyperthreading cores are also included to illustrate the usage of computational resources with respect to the number of units.

The best result achieved with 1024 units shows that after training for 11 hours on the corpus of the 947231 aligned Mizar-LaTeX pairs, we can automatically translate with perfect accuracy 69179 (65.73%) of the 105247 testing pairs. Given that the translation includes quite nontrivial transformations, this is a surprisingly good performance. Also, by manually inspecting the remaining misclassifications we have found that many of those are actually semantically correct translations, typically choosing different but synonymous expressions. A simple example of such synonyms is the Mizar expression `for x st P(x) holds Q(x)`, which can be alternatively written as `for x holds P(x) implies Q(x)`. Since there are many such synonyms on various levels and they are often context-dependent, the true *semantic performance* of the translator will have to be measured by further applying the translation [22] from Mizar to MPTP/TPTP to the current results, and calling ATP systems to establish equivalence with the original Mizar formula as we do for Flyspeck in [11]. This is left as future work.

5.4 Greedy Covers and Edit Distances

We illustrate the combined performance of translation by comparing against selected collections of models. In Table 9 "Top-n Greedy Cover" denotes a list of n models such that each model in the list gives the maximum increase of correct translations from the previous model. In addition, we also measure the percentage of sentences (both overlap and no-overlap part) that are nearly correct. The

[7] Bidirectional encoding only works on even number of layers.

Parameter	Final Test Perplexity	Final Test BLEU	Identical Statements (%)	Identical No-overlap (%)
2-Layer	3.06	41.1	40121 (38.12%)	6458 (13.43%)
3-Layer	2.10	64.2	57413 (54.55%	16318 (33.92%)
4-Layer	2.39	45.2	49548 (47.08%)	11939 (24.82%)
5-Layer	5.92	12.8	29207 (27.75%)	2698 (5.61%)
6-Layer	4.96	20.5	29361 (27.9%)	2872 (5.97%)
2-Layer Residual	1.92	54.2	57843 (54.96%)	16511 (34.32%)
3-Layer Residual	1.94	62.6	59204 (56.25%)	17396 (36.16%)
4-Layer Residual	1.85	56.1	59773 (56.79%)	17626 (36.64%)
5-Layer Residual	2.01	63.1	59259 (56.30%)	17327 (36.02%)
6-Layer Residual	NaN	0	0 (0%)	0 (0%)
2-Layer Adam	1.78	56.6	61524 (58.46%)	18635 (38.74%)
3-Layer Adam	1.91	60.8	59005 (56.06%)	17213 (35.78%)
4-Layer Adam	1.99	51.8	57479 (54.61%)	16288 (33.86%)
5-Layer Adam	2.16	54.3	54670 (51.94%)	14769 (30.70%)
6-Layer Adam	2.82	37.4	46555 (44.23%)	10196 (21.20%)
2-Layer Adam Res.	1.75	56.1	63242 (60.09%)	19716 (40.97%)
3-Layer Adam Res.	1.70	55.4	64512 (61.30%)	20534 (42.69%)
4-Layer Adam Res.	1.68	57.8	64399 (61.19%)	20353 (42.31%)
5-Layer Adam Res.	1.65	64.3	64722 (61.50%)	20627 (42.88%)
6-Layer Adam Res.	1.66	59.7	65143 (61.90%)	20854 (43.35%)
2-Layer Bidirectional	2.39	**69.5**	63075 (59.93%)	19553 (40.65%)
4-Layer Bidirectional	6.03	63.4	58603 (55.68%)	17222 (35.80%)
6-Layer Bidirectional	2	56.3	57896 (55.01%)	16817 (34.96%)
2-Layer Adam Bi.	1.84	56.9	64918 (61.68%)	20830 (43.30%)
4-Layer Adam Bi.	1.94	58.4	64054 (60.86%)	20310 (42.22%)
6-Layer Adam Bi.	2.15	55.4	60616 (57.59%)	18196 (37.83%)
2-Layer Bi. Res.	2.38	24.1	47531 (45.16%)	11282 (23.45%)
4-Layer Bi. Res.	NaN	0	0 (0%)	0 (0%)
6-Layer Bi. Res.	NaN	0	0 (0%)	0 (0%)
2-Layer Adam Bi. Res.	1.67	62.2	65944 (62.66%)	21342 (44.37%)
4-Layer Adam Bi. Res.	**1.62**	66.5	65992 (62.70%)	21366 (44.42%)
6-Layer Adam Bi. Res.	1.63	58.3	**66237 (62.93%)**	**21404 (44.50%)**

Table 7. Evaluation on various hyperparameters w.r.t. layers

Parameter	Final Test Perplexity	Final Test BLEU	Identical Statements (%)	Identical No-overlap (%)	Training Time (hrs.)
128 Units	3.06	41.1	40121 (38.12%)	6458 (13.43%)	1
256 Units	1.59	64.2	63433 (60.27%)	19685 (40.92%)	3
512 Units	1.6	**67.9**	66361 (63.05%)	21506 (44.71%)	5
1024 Units	**1.51**	61.6	**69179 (65.73%)**	**22978 (47.77%)**	11
2048 Units	2.02	60	59637 (56.66%)	16284 (33.85%)	31

Table 8. Evaluation on number of units

metric of nearness we use is the word-level minimum editing distance (Levenshtein distance). We can see from Table 9 that reasonably correct translations can be generated by just using a combination of a few models.

	Identical Statements	0	≤ 1	≤ 2	≤ 3
Best Model - 1024 Units	69179 (total)	65.73%	74.58%	86.07%	88.73%
	22978 (no-overlap)	47.77%	59.91%	70.26%	74.33%
Top-5 Greedy Cover - 1024 Units - 4-Layer Bi. Res. - 512 Units - 6-Layer Adam Bi. Res. - 2048 Units	78411 (total)	74.50%	82.07%	87.27%	89.06%
	28708 (no-overlap)	59.68%	70.85%	78.84%	81.76%
Top-10 Greedy Cover - 1024 Units - 4-Layer Bi. Res. - 512 Units - 6-Layer Adam Bi. Res. - 2048 Units - 2-Layer Adam Bi. Res. - 256 Units - 5-Layer Adam Res. - 6-Layer Adam Res. - 2-Layer Bi. Res.	80922 (total)	76.89%	83.91%	88.60%	90.24%
	30426 (no-overlap)	63.25%	73.74%	81.07%	83.68%
Union of All 39 Models	83321 (total)	79.17%	85.57%	89.73%	91.25%
	32083 (no-overlap)	66.70%	76.39%	82.88%	85.30%

Table 9. Coverage w.r.t. a set of models and edit distances

5.5 Translating from Mizar to LaTeX

It is interesting to see how the seq2seq model performs on our data when we treat Mizar as the source language and LaTeX as the target language, thus emulating Bancerek's translation toolchain. The results in Table 10 show that the model is still able to achieve meaningful translations from Mizar to LaTeX, though the translation quality is generally not yet as good as in the other direction.

Parameter	Final Test Perplexity	Final Test BLEU	Identical Statements	Percentage
512 Units Bidirectional Scaled Luong	2.91	57	54320	51.61%

Table 10. Evaluation on number of units

6 A Translation Example

To illustrate the training of the neural network, we pick a specific example (again `BHSP_2:3` as in Section 3) and watch how the translation changes as the training progresses. We can see from Table 11 that the model produces mostly gibberish in the early phases of the training. As the training progresses, the generated sentence starts to look more like the correct Mizar statement. It is interesting to see that the neural network is able to learn the matching of parentheses and correct labeling of identifiers.

Rendered LaTeX	Suppose s_8 is convergent and s_7 is convergent . Then $\lim(s_8+s_7) = \lim s_8+ \lim s_7$
Input LaTeX	`Suppose $ { s _ { 8 } } $ is convergent and $ { s _ { 7 } } $ is convergent . Then $ \mathop { \rm lim } ( { s _ { 8 } } { + } { s _ { 7 } } } ) \mathrel { = } \mathop { \rm lim } { s _ { 8 } } { + } \mathop { \rm lim } { s _ { 7 } } $ .`
Correct	`seq1 is convergent & seq2 is convergent implies lim ( seq1 + seq2 ) = ( lim seq1 ) + ( lim seq2 ) ;`
Snapshot-1000	`x in dom f implies ( x * y ) * ( f | ( x | ( y | ( y | y ) ) ) ) = ( x | ( y | ( y | ( y | y ) ) ) ) ) ;`
Snapshot-3000	`seq is convergent & lim seq = 0c implies seq = seq ;`
Snapshot-5000	`seq1 is convergent & lim seq2 = lim seq2 implies lim_inf seq1 = lim_inf seq2 ;`
Snapshot-7000	`seq is convergent & seq9 is convergent implies lim ( seq + seq9 ) = ( lim seq ) + ( lim seq9 ) ;`
Snapshot-9000	`seq1 is convergent & lim seq1 = lim seq2 implies ( seq1 + seq2 ) + ( lim seq1 ) = ( lim seq1 ) + ( lim seq2 ) ;`
Snapshot-12000	`seq1 is convergent & seq2 is convergent implies lim ( seq1 + seq2 ) = ( lim seq1 ) + ( lim seq2 ) ;`

Table 11. Translation with respect to training steps

7 Conclusion and Future Work

We for the first time harnessed neural networks in the formalization of mathematics. Due to the lack of aligned informal-formal corpora, we generated informalized LaTeX from Mizar by using and modifying the current translation done for the journal Formalized Mathematics. Our results show that for a significant proportion of the inference data, neural network is able to generate correct

Mizar statements from LaTeX. In particular, when trained on the 947,231 aligned Mizar-LaTeX pairs, the best method achieves perfect translation on 65.73% of the 105,247 test pairs, and the union of all methods produces perfect translations on 79.17% of the test pairs.

Even though these are results on a synthetic dataset, such a good performance is surprising to us and also very encouraging. It means that state-of-the-art neural methods are capable of learning quite nontrivial informal-to-formal transformations, and have a great potential to help with automating computer understanding of mathematical and scientific writings.

It is also clear that many of the translations that are currently classified by us as imperfect (i.e., syntactically different from the aligned formal statement) are semantically correct. This is due to a number of synonymous formulations allowed by the Mizar language. Obvious future work thus includes a full semantic evaluation, i.e., using translation to MPTP/TPTP and ATP systems to check if the resulting formal statements are equivalent to their aligned counterparts. As in [12,11], this will likely also show that the translator can produce semantically different, but still provable statements and conjectures.

Another line of research opened by these results is an extension of the translation to full informalized Mizar proofs, then to the ProofWiki corpus aligned by Bancerek recently to Mizar, and (using these as bridges) eventually to arbitrary LaTeX texts. The power and the limits of the current neural architectures in automated formalization and reasoning is worth of further understanding, and we are also open to the possibility of adapting existing formalized libraries to tolerate the great variety of natural language proofs.

References

1. G. Bancerek. Automatic translation in formalized mathematics. *Mechanized Mathematics and Its Applications*, 5(2):19–31, 2006.
2. G. Bancerek, C. Bylinski, A. Grabowski, A. Kornilowicz, R. Matuszewski, A. Naumowicz, K. Pak, and J. Urban. Mizar: State-of-the-art and beyond. In M. Kerber, J. Carette, C. Kaliszyk, F. Rabe, and V. Sorge, editors, *Intelligent Computer Mathematics - International Conference, CICM 2015*, volume 9150 of *LNCS*, pages 261–279. Springer, 2015.
3. G. Bancerek and P. Carlson. Mizar and the machine translation of mathematics documents. *Available online at* `http://www.mizar.org/project/banc_carl93.ps`, 1994.
4. J. C. Blanchette, C. Kaliszyk, L. C. Paulson, and J. Urban. Hammering towards QED. *J. Formalized Reasoning*, 9(1):101–148, 2016.
5. K. Cho, B. van Merriënboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics, Oct. 2014.
6. G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, Dec 1989.

7. I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning.* MIT Press, 2016. `http://www.deeplearningbook.org`.
8. T. Hales. *Dense Sphere Packings: A Blueprint for Formal Proofs*, volume 400 of *London Mathematical Society Lecture Note Series.* Cambridge University Press, 2012.
9. T. C. Hales, M. Adams, G. Bauer, D. T. Dang, J. Harrison, T. L. Hoang, C. Kaliszyk, V. Magron, S. McLaughlin, T. T. Nguyen, T. Q. Nguyen, T. Nipkow, S. Obua, J. Pleso, J. Rute, A. Solovyev, A. H. T. Ta, T. N. Tran, D. T. Trieu, J. Urban, K. K. Vu, and R. Zumkeller. A formal proof of the Kepler conjecture. *Forum of Mathematics, Pi*, 5, 2017.
10. K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Netw.*, 4(2):251–257, Mar. 1991.
11. C. Kaliszyk, J. Urban, and J. Vyskočil. Automating formalization by statistical and semantic parsing of mathematics. In M. Ayala-Rincón and C. A. Muñoz, editors, *Interactive Theorem Proving - 8th International Conference, ITP 2017*, volume 10499 of *LNCS*, pages 12–27. Springer, 2017.
12. C. Kaliszyk, J. Urban, and J. Vyskočil. Learning to parse on aligned corpora (rough diamond). In C. Urban and X. Zhang, editors, *Interactive Theorem Proving (ITP 2015)*, volume 9236 of *LNCS*, pages 227–233. Springer, 2015.
13. C. Kaliszyk, J. Urban, J. Vyskočil, and H. Geuvers. Developing corpus-based translation methods between informal and formal mathematics: Project description. In S. M. Watt, J. H. Davenport, A. P. Sexton, P. Sojka, and J. Urban, editors, *Intelligent Computer Mathematics - International Conference, CICM 2014*, volume 8543 of *LNCS*, pages 435–439. Springer, 2014.
14. A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.
15. M. Luong, E. Brevdo, and R. Zhao. Neural machine translation (seq2seq) tutorial. *https://github.com/tensorflow/nmt*, 2017.
16. C. J. Maddison, A. Huang, I. Sutskever, and D. Silver. Move evaluation in Go using deep convolutional neural networks. *CoRR*, abs/1412.6564, 2014.
17. G. Neubig. Neural machine translation and sequence-to-sequence models: A tutorial. *CoRR*, abs/1703.01619, 2017.
18. C. Olah. Understanding LSTM networks. `http://http://colah.github.io/posts/2015-08-Understanding-LSTMs/`.
19. K. Papineni, S. Roukos, T. Ward, and W. Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318. ACL, 2002.
20. I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 3104–3112, Cambridge, MA, USA, 2014. MIT Press.
21. A. Trybulec and P. Rudnicki. A collection of TeXed Mizar abstracts. *Available online at http://www.mizar.org/project/TR-89-18.pdf*, 1989.
22. J. Urban. MPTP 0.2: Design, implementation, and initial experiments. *J. Autom. Reasoning*, 37(1-2):21–43, 2006.
23. J. Urban and J. Vyskočil. Theorem proving in large formal mathematics as an emerging AI field. In M. P. Bonacina and M. E. Stickel, editors, *Automated Reasoning and Mathematics: Essays in Memory of William McCune*, volume 7788 of *LNAI*, pages 240–257. Springer, 2013.

8 Appendix A: Effect of Training Steps

Rendered LaTeX	Suppose s_8 is convergent and s_7 is convergent . Then $\lim(s_8{+}s_7) = \lim s_8{+} \lim s_7$
Input LaTeX	`Suppose $ { s _ { 8 } } $ is convergent and $ { s _ { 7 } } $ is convergent . Then $ \mathop { \rm lim } ( { s _ { 8 } } { + } { s _ { 7 } } ) \mathrel { = } \mathop { \rm lim } { s _ { 8 } } { + } \mathop { \rm lim } { s  _ { 7 } } $ .`
Correct	`seq1 is convergent & seq2 is convergent implies lim ( seq1 + seq2 ) = ( lim seq1 ) + ( lim seq2 ) ;`
Snapshot-1000	`x in dom f implies ( x * y ) * ( f | ( x | ( y | ( y | y ) ) ) ) = ( x | ( y | ( y | ( y | y ) ) ) ) ) ;`
Snapshot-2000	`seq is summable implies seq is summable ;`
Snapshot-3000	`seq is convergent & lim seq = 0c implies seq = seq ;`
Snapshot-4000	`seq is convergent & lim seq = lim seq implies seq1 + seq2 is convergent ;`
Snapshot-5000	`seq1 is convergent & lim seq2 = lim seq2 implies lim_inf seq1 = lim_inf seq2 ;`
Snapshot-6000	`seq is convergent & lim seq = lim seq implies seq1 + seq2 is convergent ;`
Snapshot-7000	`seq is convergent & seq9 is convergent implies lim ( seq + seq9 ) = ( lim seq ) + ( lim seq9 ) ;`
Snapshot-8000	`seq1 is convergent & seq2 is convergent implies lim seq1 = lim seq2 + lim seq2 ;`
Snapshot-9000	`seq1 is convergent & lim seq1 = lim seq2 implies ( seq1 + seq2 ) + ( lim seq1 ) = ( lim seq1 ) + ( lim seq2 ) ;`
Snapshot-10000	`seq1 is convergent & lim seq1 = lim seq2 implies seq1 + seq2 is convergent ;`
Snapshot-11000	`seq1 is convergent & lim seq = lim seq1 implies lim_sup seq1 + lim_sup seq2 = lim seq1 + lim seq2 ;`
Snapshot-12000	`seq1 is convergent & seq2 is convergent implies lim ( seq1 + seq2 ) = ( lim seq1 ) + ( lim seq2 ) ;`