Learning representations of Logical Formulae using Graph Neural Networks

Xavier Glorot, Ankit Anand, Eser Aygün, Shibl Mourad, Pushmeet Kohli, Doina Precup
DeepMind
{glorotx, anandank, eser, shibl, pushmeet, doinap}@google.com

Abstract

We explore the use of Graph Neural Networks(GNNs) for learning representations of propositional and first-order logical formulae. Traditional non-graphical based approaches like CNNs and LSTMs do not exploit invariant properties like variable renaming and order invariance predominantly present in logical formulae. In this work, we explicitly try to encode these logical invariances using GNNs. We use the task of entailment proposed in Evans *et al.* [2018] for propositional logic. We also explore our approach for the task of proof length prediction in first-order logic. We use the Mizar-40 dataset to evaluate several representation learning approaches for proof length prediction task. We observe that GNNs significantly outperform the other traditional approaches on both these tasks.

1 Introduction

Combining the power of representation learning from neural networks with logical reasoning holds the promise of enabling us to advance the state-of-the-art in AI. Several recent papers have focused on learning inference rules for logical reasoning, such as learning entailment in propositional logic [Evans *et al.*, 2018], learning satisfiability of logical formulas [Selsam *et al.*, 2018] or using reinforcement learning for theorem proving [Kaliszyk *et al.*, 2018]. Other recent work has incorporated logical rules into neural networks by including them as constraints or in the loss functions [Xu *et al.*, 2018] or by learning embeddings for logical constructs [Minervini *et al.*, 2018; Rocktäschel and Riedel, 2017]

We tackle the problem of how to learn a good representation of logical formulae using neural networks. Good representations would be useful for a variety of logical reasoning tasks, including theorem proving, satisfiability, entailment, etc. A straightforward approach to input a logical formula into a neural network is as a sequence of symbols, which can then be used as input to a 1-D convolutional network or recurrent network. However, such models do not obey certain invariant properties of logical formulae like order invariance and variable renaming [Battaglia *et al.*, 2018]. Better structured representations have been proposed in recent work, including TreeLSTMs for entailment in propositional logic [Evans *et al.*, 2018], string representation in Higher Order Logic (HOL) [Bansal *et al.*, 2019], and GNNs for premise selection in HOL [Wang *et al.*, 2017].

In this paper, we propose a novel way to encode propositional and FOL formulae into GNN and capture many more invariances not exploited in previous works. Specifically, given a logical formula, our approach converts it directly into a graph, encoding each type of node by a specific node embedding and corresponding edges by edge embeddings. The initial graph representation captures multiple invariances present in the logical formula. The subsequent embeddings are updated by message passing steps using the framework of GNNs. We propose a novel graph representation to capture multiple invariances arising due to complexity of FOL formula. We use the entailment dataset for propositional logic (Evans *et al.* [2018]) and obtain significant improvements on test accuracy compared to previous approaches. For FOL, we test our GNN representations on the complex task

Workshop on Graph Representation Learning at 33rd Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

of proof length prediction on Mizar-40 theorems. Our GNN representations capture many more invariances and obtain significant improvements compared to non-graphical models.

2 Background and Related Work

Logical Formulae: Propositional logic and first-order logic (FOL) are the two most fundamental types of formal logic. Propositional logic deals with logical variables connected via logical connectives such as and ($\wedge$), or ($\vee$), not($\neg$) and implication ($\Rightarrow$). FOL, on the other hand, allows one to talk about non-logical values via quantifiers, relations and functions. We refer the reader to Fitting [2012] for a formal treatment of formal logic.

Representation Learning of Logical Formula: Several works aimed to learn representations adequate for various forms of logical reasoning. Allamanis *et al.* [2017], building on the work of Zaremba *et al.* [2014], propose a network architecture that is trained to map semantically identical formulae into similar continuous embeddings, and evaluate it on the task of determining equivalence of Boolean formulae and polynomials. Arabshahi *et al.* [2018] use a TreeLSTM to combine symbolic reasoning and black-box function evaluations on equation verification and completion tasks. Hohenecker and Lukasiewicz [2018] introduce a model called recursive reasoning network for ontology reasoning. Chvalovsky [2019] proposes a model built of several recurrent neural networks, which leverage the tree corresponding to a propositional formula, in order to learn to detect tautologies. GamePad [Huang *et al.*, 2018], another related recent work, focuses on proof length prediction and tactic selection in HOL. However, the experiments focus on a 3-way classification of proof lengths, rather than the full regression task. The models used to learn representations are sequential, like GRU and TreeLSTM, so they also do not fully capture invariance properties of logical formulae.

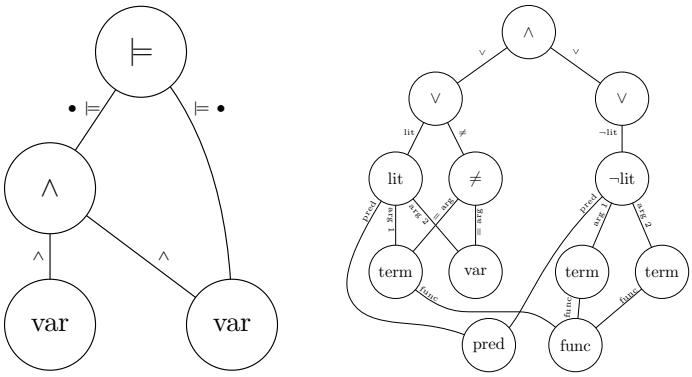
Our work is closely related to [Wang *et al.*, 2017], which used GNNs for premise selection in HOL, *i.e.* for predicting the usefulness of an axiom for a given conjecture. Wang *et al.* [2017] proposed graph embeddings for this task and showed substantial gains compared to other machine learning methods for premise selection. However, the usefulness of an axiom for proving a conjecture depends on the presence of other axioms. For this reason, we propose to use GNNs for a novel task of proof length prediction. Further, our model adds the notion of edge embeddings in GNNs instead of using triplets for determining the ordering of arguments in predicates and functions, as in [Wang *et al.*, 2017], which yields a much simpler model. Another closely related work is NeuroSAT(Selsam *et al.* [2018]), which learns a graphical representation to predict satisfiability in propositional logic. Their representation assumes the formula is given in conjunctive normal form (CNF) while our representation is more general and can handle any raw form in propositional logic.

3 Entailment in Propositional Logical Formulae

The input of the entailment prediction task [Evans *et al.*, 2018] is a pair of propositions (π, ρ), and the target is 1 when $\pi \models \rho$, and 0 otherwise. Good performance on this task indicates the ability to perform logical deduction. We compared our GNN-based model with ConvNet, TreeLSTM and PossibleWorldNet, which were explored in [Evans *et al.*, 2018].

Graphical representation: Our GNN model consumes a graph representation of the propositions. Given a propositional formula, we create a node for each logical operator "$\neg$", "$\vee$", "$\wedge$", "$\Rightarrow$", each one is connected to two nodes (except "$\neg$" which allows one). For each type of node, two edge types are used to connect it with its input nodes: one for bottom-up connection and the other for the reverse. A single node of type "variable" is created for each distinct variable name. The parse structure of the proposition is then used to connect "variable" nodes with logical operator nodes of different types. The "$\Rightarrow$" nodes have four input edge types, because the relation is not symmetric and we need to discriminate the left and right input. The resulting graph is very similar to the tree representation of the proposition, with one crucial difference: the tree representation associates each variable occurrence with a distinct leaf node that must somehow encode the identity of the variable, whereas the graph representation uses one node per variable. In other words, the graph representation is invariant to variable renaming, while the tree representation is not, which is a very desirable property. For a given entailment task (π, ρ), we further join the graphs corresponding to π and ρ via a "master" node, which represents the entailment relationship. Figure 1a shows a simple graph for

the example $(p \wedge q, p)$. It indicates two variables nodes corresponding to p and q, a logical $\wedge$ node and a master node for entailment. Different edge embedding types are represented with edge labels.



(a) $p \wedge q \models p$ (b) $(c = x \Rightarrow R(c, x)) \wedge \neg R(c, c)$

Figure 1: Graphical representation of a propositional entailment (a) and a FOL formula (b), with node and edge types indicated. In our GNN models, there are two edge types (forward and backward) for every type of relationship. We only show forward edge types in the diagrams.

GNN Architecture: We use a similar GNN architecture to Battaglia *et al.* [2016]. Each type of node and edge is associated to a learnable embedding vector. We set the initial feature vector of each node and edge as their type embedding vector. This initial feature vector will be concatenated to the current one at each message passing step. Message passing is carried out for a fixed number of iterations. Each iteration starts by updating the edge features. The feature vectors of an edge and of nodes sending messages to it are concatenated by pairs and fed to the edge update function to produce the new embedding. All existing pairs for a given edge are aggregated by summation. Then, node features are updated in an analogous way, but using the previously updated edge features, using a different node update function, and only considering incoming edges when pairing feature vectors of nodes and edges. The node and edge update functions are MLPs with tanh non-linearity, fixed number of layers and fixed hidden layer sizes. We do not use global features for the whole graph. After the maximum number of message passing iterations is reached, node features are fed to a linear layer to produce a logit. The logits are aggregated by averaging over the full graph. The result, passed through a sigmoid non-linearity, is the model prediction.

Experimental Details and Results: We used the synthetic datasets generated by [Evans *et al.*, 2018] (See details in original paper). The hyperparameters explored and the learning details of our experiments are given in the supplementary material. Table 1a shows the results comparing our GNN model with the previously proposed models of PossibleWorldNet (PWN), ConvNet, LSTM and TreeLSTM [Evans *et al.*, 2018]. The results clearly indicate that the GNN model performs consistently better than PossibleWorldNet, which was the state-of-the-art model until now. We believe this is due to the invariances captured by the GNN model and not by other models. Also, GNNs obtain 98.3% accuracy on "exam" dataset, which has logical entailments from textbooks, demonstrating impressive generalization.

4 Proof Length Prediction in First-order Logic

We use the task of proof length prediction to learn representations in FOL. The task is useful for ATPs in guiding the search towards shorter proofs and provides a proxy for the complexity of a theorem.

Graphical Representation and Architecture: As in propositional logic, we construct a graph representation of a FOL formula which is given as input to the GNN model. We first convert the FOL formula to CNF. Each CNF formula begins with a $\wedge$ node, which is connected to multiple clauses, each of which contains a $\vee$ node connected to multiple literal (positive or negative), $=$ or $\neq$ nodes. Literal nodes are connected to predicate symbols and their arguments, which are terms or variables. $=$ and $\neq$ nodes are directly connected to variables or terms nodes. Terms in turn are connected to function symbols and their arguments. The graph uses one node for each predicate symbol, function symbol and variable symbol, shared for their different appearances in the formula. This leads to ten node types: $\wedge$, $\vee$, literal, $\neg$literal, $=$, $\neq$, term, predicate, function, variable. The edge types are

determined by the child node type in the parse structure hierarchy and its direction, except for literal and function parent nodes, where each argument position uses a different edge type. This results in 48 different edge types, for a maximum predicate and function arity of eight. Fig. 1b shows the graph constructed for the formula $(c = x \Rightarrow R(c, x)) \wedge \neg R(c, c)$, where x is a variable, R is a binary relation symbol and c is a zero-ary function symbol. This representation captures multiple invariances present in the FOL formula which are difficult to capture in non-graphical representation such as: 1) Invariance to ordering of different clauses 2) Invariance to ordering of literals within a clause 3) Renaming of predicates, functions and variables. Also, the ordering or arguments (within a literal or function) is captured by edge embeddings which simplies the architecture significantly compared to Wang *et al.* [2017]. The GNN is similar to the one we proposed for propositional logic. The only difference in the FOL GNN model is that there is no sigmoid non-linearity at the end; the node features are passed through a linear layer to produce a real-valued output.

Model	easy	hard	big	massive	exam
ConvNet	59.7	52.6	54.9	50.4	54.0
LSTM	68.3	58.1	61.1	52.7	70.0
TreeLSTM	77.8	74.2	74.2	59.3	75.0
PWN	*98.6*	*96.7*	*93.9*	*73.4*	*96.0*
GNN	**99.9** $\pm$ 0.1	**98.3** $\pm$ 0.5	**96.8** $\pm$ 0.9	**96.5** $\pm$ 1.4	**98.3** $\pm$ 1.2

(a)

Model	Mizar
Median	9.44
MLP	4.35 $\pm$ .04
ConvNet	4.41 $\pm$ .17
LSTM	4.36 $\pm$ .03
GNN	**3.49** $\pm$ .06

(b)

Table 1: **(a)** Model Accuracy in Entailment task in Propositional Logic. **(b)** Average FOL Proof Length L_1 error for various models. We report the average score and its standard deviation for 10 different parameter initialization and training mini-batch shuffling with the selected hyperparameters.

Dataset: We use theorems from the Mizar-40[1] [Kaliszyk and Urban, 2015] dataset. We filtered out problems with object symbols, as well as problems with sequence length ≥ 2048 (for memory constraints and simplicity). This resulted in 11,563 theorems, from which 1,000 are sampled uniformly for the test and validation sets. Generating correct targets for proof length prediction is a non-trivial task. Our initial investigation revealed that brute force approach of breadth first approach does not scale beyond 5-6 proof steps. Hence, we create approximate targets by using the number of manipulations taken by close to state-of-the-art open source prover: E-prover [Schulz, 2013]. E-prover uses efficient heuristics, and while they are not guaranteed to produce the shortest proof, the resulting number of steps is a reasonable target.

Experimental Details and Results: The simplest baseline we use is the constant prediction equal to the median of the training example targets. Secondly, we use simple graph features which are fed to an MLP to obtain a scalar output (details are in supplementary material). The third baseline is a ConvNet-based sequence model. As in propositional logic, the symbols are encoded independently to an embedding space. The embeddings are concatenated as they appear in the sequence, then processed by multiple 1-D convolutions and max-pooling layers. The final output is obtained by feeding the resulting representation through a MLP without an output non-linearity. Finally, LSTM-based sequence models pass the embeddings of individual symbols through a recurrent LSTM net. The actual output is again obtained by feeding the resulting embedding through a MLP.

We use the L_2 loss for all models. To prevent the target values from being arbitrarily large, we use : $\text{discounted\_target} = (1 - \gamma^{\text{target}})/(1 - \gamma)$, with $\gamma = 0.99$. The list of hyperparameters we explored are given in the supplementary material. Table: 1b reports the L_1-error between the predicted and discounted target. The table shows that GNNs perform significantly better than other the baselines on this complex regression task for the Mizar dataset. We believe this good behavior is due in large part to the invariant properties exploited by GNNs in contrast to other models.

5 Conclusion and Future Work

We explored the problem of representation learning for logical formulae and demonstrate that GNNs are effective in learning good representations for both propositional and FOL. We propose graphical representations for both propositional logic and show huge improvements on non-graphical baselines. In the future, we would like to learn representations for HOL, and use them for practical tasks like program analysis and verification.

[1] `https://github.com/JUrban/deepmath/tree/master/mizar40`

References

Miltiadis Allamanis, Pankajan Chanthirasegaran, Pushmeet Kohli, and Charles Sutton. Learning continuous semantic representations of symbolic expressions. In *International Conference on Machine Leanring (ICML)*, 2017.

Forough Arabshahi, Sameer Singh, and Animashree Anandkumar. Combining symbolic expressions and black-box function evaluations for training neural programs. In *International Conference on Learning Representations (ICLR)*, 2018.

Kshitij Bansal, Sarah M Loos, Markus N Rabe, Christian Szegedy, and Stewart Wilcox. Holist: An environment for machine learning of higher-order theorem proving (extended version). *arXiv preprint arXiv:1904.03241*, 2019.

Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and koray kavukcuoglu. Interaction networks for learning about objects, relations and physics. In *Advances in Neural Information Processing Systems 29*, pages 4502–4510, 2016.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Karel Chvalovsky. Top-down neural model for formulae. In *International Conference on Learning Representations (ICLR)*, 2019.

Richard Evans, David Saxton, David Amos, Pushmeet Kohli, and Edward Grefenstette. Can Neural Networks Understand Logical Entailment? In *International Conference on Learning Representations (ICLR)*, 2018.

Melvin Fitting. *First-order logic and automated theorem proving*. Springer Science & Business Media, 2012.

Patrick Hohenecker and Thomas Lukasiewicz. Ontology reasoning with deep neural networks. In *arxiv CoRR abs/1808.07980*, 2018.

Daniel Huang, Prafulla Dhariwal, Dawn Song, and Ilya Sutskever. Gamepad: A learning environment for theorem proving. *arXiv preprint arXiv:1806.00608*, 2018.

Cezary Kaliszyk and Josef Urban. Mizar 40 for mizar 40. *J. Autom. Reason.*, 55(3):245–256, October 2015.

Cezary Kaliszyk, Josef Urban, Henryk Michalewski, and Miroslav Olšák. Reinforcement learning of theorem proving. In *Advances in Neural Information Processing Systems*, pages 8822–8833, 2018.

Pasquale Minervini, Matko Bosnjak, Tim Rocktäschel, and Sebastian Riedel. Towards neural theorem proving at scale. In *Proceedings of the 2nd Workshop on Neural Abstract Machines and Program Induction (NAMPI)*, 2018.

Tim Rocktäschel and Sebastian Riedel. End-to-End Differentiable Proving. In *Advances in Neural Information Processing Systems*, pages 3788–3800, 2017.

Stephan Schulz. System Description: E 1.8. In Ken McMillan, Aart Middeldorp, and Andrei Voronkov, editors, *Proc. of the 19th LPAR, Stellenbosch*, volume 8312 of *LNCS*. Springer, 2013.

Daniel Selsam, Matthew Lamm, B Benedikt, Percy Liang, Leonardo de Moura, David L Dill, et al. Learning a SAT Solver from Single-Bit Supervision. In *International Conference on Learning Representations, ICLR*, 2018.

Mingzhe Wang, Yihe Tang, Jian Wang, and Jia Deng. Premise selection for theorem proving by deep graph embedding. In *Advances in Neural Information Processing Systems*, pages 2786–2796, 2017.

Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Van den Broeck. A Semantic Loss Function for Deep Learning with Symbolic Knowledge. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5502–5511, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

Wojciech Zaremba, Karol Kurach, and Rob Fergus. Learning to discover efficient mathematical identities. In *Advances in Neural Information Processing Systems*, 2014.

Appendix

5.1 Training Details

The hyperparameters explored for the propositional entailment task are given in Table 2, the ones for the proof length prediction task are given in Table 3. For both experiments, we ran the training for 1,000,000 optimization steps and evaluated our model on the validation set every 1,000 steps. We select the hyperparameters giving the best validation score. Further, we use the cross entropy loss function and the ADAM optimizer with an exponential annealing learning rate schedule in propositional entailment task. For both experiments the other details are the same: we use the ADAM optimizer with the default parameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$), an exponential annealing learning rate schedule following the formula: $\mathrm{lr}(\mathrm{step}) = \mathrm{limit} + (\mathrm{start} - \mathrm{limit}) * \mathrm{decay}^{(\mathrm{step}/N)}$ with $\mathrm{limit} = 10^{-7}$, $\mathrm{decay} = 0.1$ and $N = 200000$. All bias parameters are initialized by zeros, the weight parameters are sampled from a normal distribution with mean zero and standard deviation $\frac{1}{\sqrt{M}}$, with M the fan-in (i.e. the number of inputs to the layer). For FOL, the MLP based model compute simple graph features like the number of nodes, the number of edges and the proportion of each type of edge and type of nodes in the graph, resulting in a 63 dimension feature vector which we feed to a MLP to obtain a scalar output.

	GNN
Dataset	Propositional entailment
Batch size	64
Learning rate	0.0003, 0.0001, 0.00003
Embedding size	32, 64, 128, 256
Update functions layers	1, 2, 3, 4
L_2 regularization	0., 0.0001, 0.001, 0.01
Dropout rate	0., 0.2, 0.4
Message passing steps	12, 24, 48

Table 2: Hyperparameters for the GNN for the propositional entailment experiments. Hyperparameters selected with the best validation performance are underlined.

5.2 Additional results

To better understand the behavior of the learned models for the proof length prediction task, we show the scatter plots of the GNN predictions versus the discounted targets for Mizar-40 in Figures 2 and 3.

Figure 2: GNN predictions versus discounted targets on Mizar-40 test set

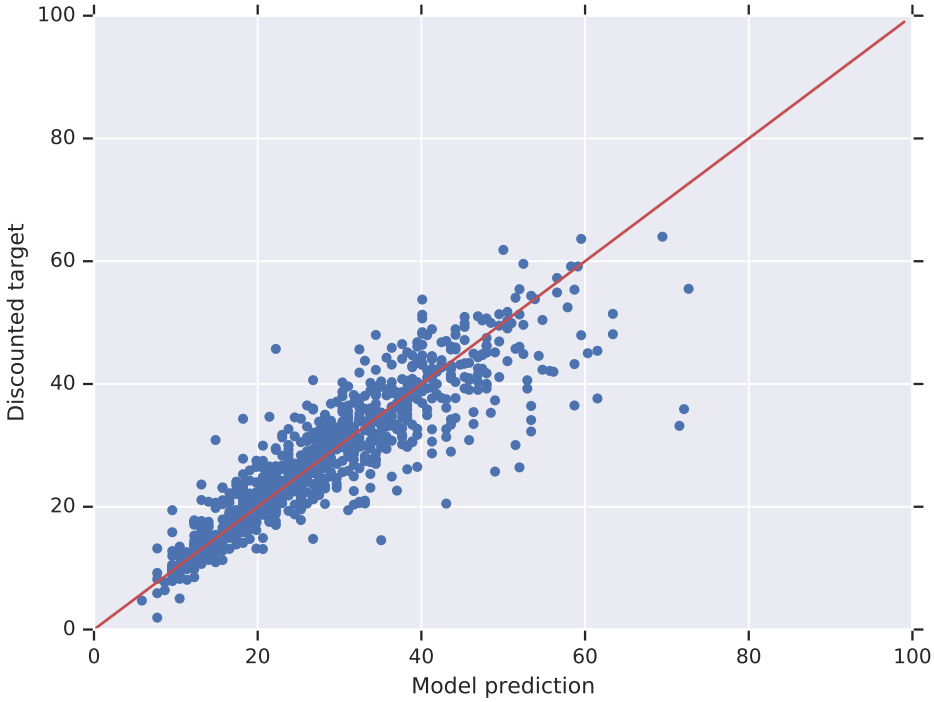
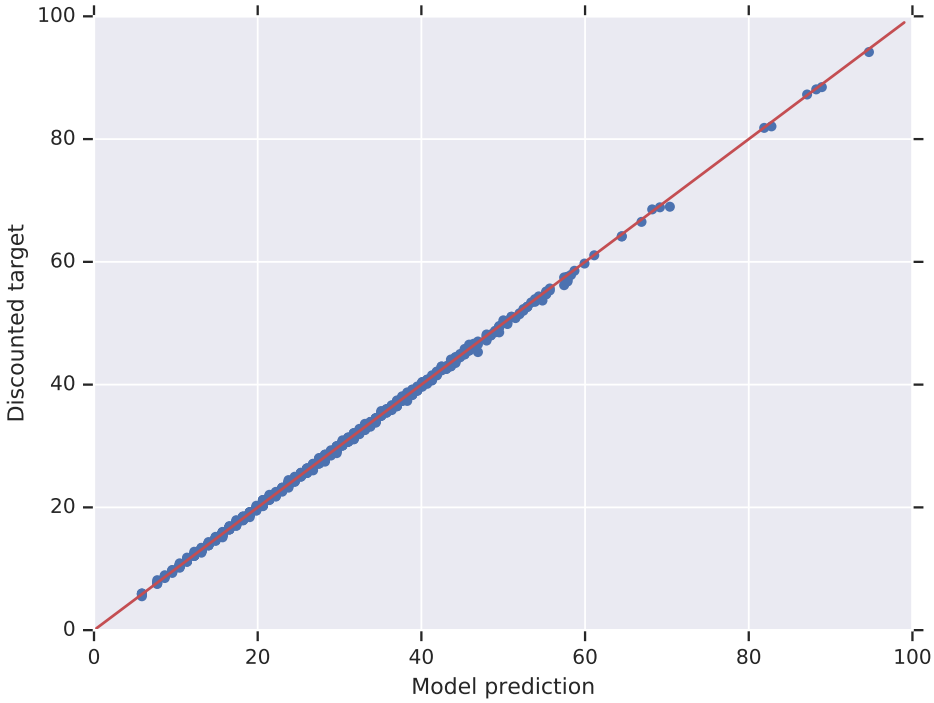


Figure 3: GNN predictions versus discounted targets on 1,000 examples of Mizar-40 train set



MLP	
Dataset	Mizar
Batch size	64, 128
Learning rate	0.001, 0.0003, 0.0001
Embedding size	128, 256, 512, 1024
Number of layers	1, 2, 3, 4, 5
L_2 regularization	0., 0.1, 1., 10.

ConvNet	
Dataset	Mizar
Batch size	64
Learning rate	0.0003, 0.001, 0.003
Embedding size	128, 256, 512
Number of layers	1, 3, 6
L_2 regularization	0., 0.01, 1.0
Kernel size	12, 24, 36
Max pooling every	1, 2, 3

LSTM	
Dataset	Mizar
Batch size	32
Learning rate	0.001, 0.0003, 0.0001
Embedding size	32, 64, 128, 256
MLP output layers	1, 2, 3
L_2 regularization	0., 0.001, 0.1, 10.
Dropout rate	0., 0.15, 0.3, 0.45

GNN	
Dataset	Mizar
Batch size	16, 32
Learning rate	0.001, 0.0003, 0.0001
Embedding size	256, 512, 1024
Update functions layers	1, 2
L_2 regularization	0., 0.00001
Dropout rate	0., 0.15
Message passing steps	6, 12, 18

Table 3: Hyperparameters for the different models for the FOL proof length prediction experiments. Hyperparameters selected with the best validation performance are underlined.