International Journal of Latest Trends in Engineering and Technology
Vol.(12)Issue(1), pp.011-014
DOI: http://dx.doi.org/10.21172/1.121.03
e-ISSN:2278-621X

MACHINE LEARNING FOR CODE AUTOMATION

Ms. Shraddha AshokKumar Maurya[1], Mr. Soham Milind Mudalgikar[2], Ms. Dnyanada Mahendra Kadiyal[3]

Abstract- Artificial Intelligence and its sub-technology machine learning is spreading its roots in most of the areas. This paper presents one of those areas, where machine learning is being used for code automation. Microsoft researchers and student from University of Cambridge have come up with such a technology named DeepCoder, which can solve simple and short problems like those that appear in a competitive programming contest. When provided with input-output specifications, DeepCoder can solve the problem using Deep Learning. The approach of the system is to train the neural networks to predict the properties or attributes required to build the program to achieve the specified output from the given inputs.
Keywords – machine learning, deep learning, code automation, deepcoder

1. INTRODUCTION

Artificial Intelligence is a promising field of computer science whose primary concern is finding an effective way to understand and apply intelligent problem solving, planning, and communication skills to a wide range of practical problems. What AI researchers were trying to do is, to replicate the human thoughts by giving a lot of rules to computers. But, instead the researchers began to write algorithms that learn from themselves and hence giving rise to a new subfield of AI called Machine Learning, which aimed at solving the challenge area of AI which is learning. Machine Learning looks into the study and construction of procedures or algorithms that can learn from and make predictions on given data. Such algorithms work by first building a model from example inputs in order to make predictions or decisions based on data, instead of following strictly static program instructions. Researchers came up with another approach to achieve this which is Neural Networks, inspired from the understanding of biological brain with all those interconnections between the neurons. Unlike a biological brain, neural networks have discrete layers, connections, and directions of data propagation. Further deep learning enables the neural networks to learn themselves from the training dataset by using more deep neural network and larger training dataset.
There has been much work done on generating the code using machine learning before the proposal of DeepCoder. A paper explored the use of gradient descent algorithm to generate source code from input-output examples via differentiable interpreters [1]. Another paper explored the generation of source code from unstructured text descriptions [2]. In DeepCoder the main two ideas are proposed. First is to learn to generate programs. Second is to integrate neural network architectures with search-based techniques [3].
DeepCoder uses the Inductive Program Synthesis (IPS) problem that generates the program which has a consistent behavior with input-output examples and to achieve this, an approach called Learning Inductive Program Synthesis (LIPS) is used [4].
A framework for using deep learning for program synthesis problems like ones appearing in the programming contests is defined, with the some contributions such as, defining a programming language that would support real world programming problems, building a model for mapping the input-output examples to the program attributes and showing an order of magnitude speedup over standard program synthesis [3].

2. INDUCTIVE PROGRAM SYNTHESIS (IPS)

Basically, building an IPS system involves solving of two problems. Firstly we need to solve the search problem, i.e. we need to find a program space to search for the consistent programs. Secondly we need to solve the ranking problem, i.e. we need to prioritize the programs as there can be multiple consistent programs in the program space. Solving these problems requires the basic foundation as choosing the Domain Specific Language and hence formulating the problem [3].

2.1 Domain Specific Language (DSL)

DeepCoder uses a DSL rather than using a full-featured programming language like C++. DSLs are suitable for specialized domains and are much more restrictive than the full-featured programming languages and hence we get a limited program space, reducing the complexity involved in solving the ranking problem [3].

[1] Department of Information Technology, KIT's College of Engineering, Kolhapur, Maharashtra, India
[2] Department of Mechanical Engineering, KIT's College of Engineering, Kolhapur, Maharashtra, India
[3] Department of Computer Science and Engineering, KIT's College of Engineering, Kolhapur, Maharashtra, India

2.2 Search Techniques

There are various methods available for searching programs consistent with the input-output examples in the program space. Different approaches are available to implement these search techniques such as defining the grammar and then checking its consistency with the examples and then combining it with the pruning [3].

Another approach is that of the FlashMeta which describes a framework for DSLs that allows the decomposition of the search problems. For example, the problem of gaining an output string from the provided input string can be decomposed into two parts such as the first part will be finding a program to be produce the first part of the output and the second part will be a program producing the second part of the output and the further both these parts will be concatenated [5].

2.3 Ranking

Already DSL reduces the complexity of the ranking problem due to reduced search space which is gained from its restrictive feature. Further, there are more two approaches to solve this ranking problem. One approach can be choosing the shortest program of the consistent programs from the program space. Second approach is of assigning scores to the programs such that the higher scores to the ground truth programs than the other programs that satisfy the input-output program specifications provided the ground truth programs.

3. LEARNING INDUCTIVE PROGRAM SYNTHESIS (LIPS)

The approach followed for the DeepCoder is called Learning Inductive Program Synthesis (LIPS). There are various components of LIPS: (i) DSL and its attributes, (ii) Data generation procedure, (iii) A machine learning model that would map the input-output examples to the program attributes, and (iv) A search procedure that would search for the consistent program in the program space and here the search will be guided by the machine learning model created in previous stage [3].

DSL and it's attributes

DSL should be expressive enough so that it can hold the problem provided as input to the DeepCoder and also it should be more restrictive so that the search space can be limited. Not only this but also, an attribute function that would map the programs of DSL to finite attribute vectors is specified in LIPS [3].

TerpreT

The DSL used in DeepCoder is the TerpreT language released by Microsoft on August 15, 2016 [4]. The language follows Python syntax and uses ast library of Python to parse and compile the TerpreT models. TerpreT uses four back-end inference algorithms. The first being, FMGD, Forward Marginals Gradient Descent. Second is ILP, Integer Linear Programming. Third is SMT, Satisfiability Modulo Theories. And the fourth one is Sketch [6].

The DSL used in DeepCoder can be categorized into three different classes. The first class is the First Order Functions. This class of functions includes functions such as HEAD, LAST, ACCESS, MINIMUM, MAXIMUM, SORT, SUM, etc. The HEAD function returns the first element of the given array while the LAST function returns the last element of the given array and NULL in case the array is empty. The ACCESS function returns (n+1)th element of a given array and n being the index of element required. The MINIMUM function returns the minimum or the smallest value from the given array while the MAXIMUM function returns the maximum or the largest value from the given array. The SORT function sorts the given array elements in non-decreasing pattern. The SUM function returns the sum of the array elements. The second class is the High Order Functions. This class of functions have to be concatenated with the third class of functions which is Lambda Functions, in order to provide with required operation. High Order Functions include functions like MAP, FILTER, COUNT, etc. Lambda Functions include functions like (<0), (*4), (+1), (-1), (*(-1)), etc. Given a lambda function and an array, the MAP functions returns an array resulting after applying the given lambda function to each element of given array. Given a lambda function that maps integers to the truth values, the FILTER returns the elements of the array that satisfy the lambda function in required order. Given a lambda function mapping the integers to the truth values and an array, the COUNT returns the number of elements that satisfy the required condition [3].

3.1 Data Generation

Initially a large dataset is generated of the programs in DSL. Further the checks are needed to be performed to ensure generate a feasible dataset. To generate this, enumeration and pruning of dataset is performed. A range for the valid values for the inputs are obtained, with the help of constraints. If the range obtained for a particular input value is empty, the corresponding program is discarded. Otherwise, with to the help of pre-computed ranges, the input-output pairs are generated which after the program execution give the output values [4].

3.2 Machine Learning Model

When designing a machine learning model, the aim is to learn to recognize the patterns in the input-output examples and then using these patterns to predict either the presence or absence or both of the individual functions of DSL. Further the mapping of input-output examples to attributes is achieved using the neural networks. The neural networks consist of two parts viz., the encoder and the decoder [3].

3.3 Encoder

A differentiable mapping from a set of input-output examples to a latent real-valued vector is achieved using encoder. Further, the encoder is made using the feed-forward network, because it is easy to train than that for the Recurrent Neural Network (RNN). However, there exists a disadvantage as it requires an upper bound on the input and output length of the arrays [4].

3.4 Decoder

A differentiable mapping from the latent vector representing a set of input-output examples to predictions of the ground truth program's attributes is achieved using decoder. Encoder and decoder both the parts of the machine learning model correspond to a standard sequence-to-sequence model by using RNN. However, the more modernized RNN decoder is not that successful according to the experimental data [4].

3.5 Search

The search for a program consistent with input-output examples is guided using the neural network instead of directly predicting the entire source code. For this purpose various search techniques can be implemented such as Depth-first search (DFS), Sort and add enumeration, Sketch and λ2 [3].

An optimized version of DFS is used to search over the program space for a given maximum length. The functions in DSL are tried out in the order, which is the prediction of probabilities from the neural network used in machine learning model [3].

A more optimized way to perform search is obtained using the sort and add enumeration technique which maintains a set of active functions from the set of predicted probabilities and then DFS is performed on the set of active functions only, hence utilizing the predictions by the neural network more efficiently [3].

Sketch is a program synthesis tool which is used to synthesize programs by filling in the holes in incomplete source code so as to match specified requirements. Sketch can utilize the neural network predictions in a sort and add technique, as the possibilities for each function hole can be restricted to the current active set [3].

λ2 is also a program synthesis tool. It draws inference regarding the small functional programs required to decide or manipulate the data structures required for the program according to the input and output values. It can also be used in combination with the sort and add algorithm by choosing some specific library or library function according to the prediction of the neural network [3].

4. DEEPCODER COMPARED TO BASELINES

The researchers at very first trained a neural network to provide with a prediction of the used functions. Further a set of programs was constructed which was different from the ones used to train the neural network. Five input - output examples were then designed for each program. These examples were then passed to the trained neural network and predictions were obtained regarding the functions to be used. These predictions, as guidance, were further provided to the search algorithms or procedures. Then the time required by each algorithm to find out a consistent program from the space was recorded and compared.

Following is an example program which provides an integer array as its input.

a ← [int]	**An input-output example:**
b ← FILTER (<0) a	*Input*:
c ← MAP (*4) b	[-17, -3, 4, 11, 0, -5, -9, 13, 6, 6, -8, 11]
d ← SORT c	*Output*:
e ← REVERSE d	[-12, -20, -32, -36, -68]

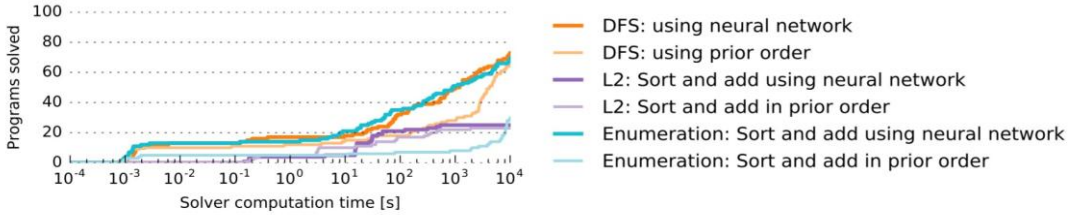
An example program in DSL: takes an integer array as it's input[3]

The first step in the program is to use the Filter function in order to get an array of negative integers only from the provided integer array, that is, to eliminate all the integers greater than and equal to zero. In next step, the Map function with a parameter of (*4) is used, which specifies to multiply each negative integer from the obtained array in first step, with 4 and hence generate a new array. Further this array is sorted using the Sort function. And in the final step the Reverse function is used to reverse the order of the sort. Initially only the input array and the required output array is provided to the neural network. Then the neural network predicts the probability of various functions that can be used to obtain the required output and hence a set of consistent programs is obtained.

Neural Netwrok predicts the probability of each function appearing[3]
This set of consistent programs is further used as guidance to search for a more consistent and appropriate program which can efficiently give the required output in minimum time. Here, various search algorithms are tested and compared for the search speedups.

Timeout needed	DFS			Enumeration			λ^2
to solve	20%	40%	60%	20%	40%	60%	20%
Baseline	$163s$	$2887s$	$6832s$	$8181s$	$>10^4s$	$>10^4s$	$463s$
DeepCoder	$24s$	$514s$	$2654s$	$9s$	$264s$	$4640s$	$48s$
Speedup	**6.8×**	**5.6×**	**2.6×**	**907×**	**>37×**	**>2×**	**9.6×**

Search Speedups on programs of length T = 5[3]
According to the comparison and analysis they infer that the required performance and learning rate is better matched with the sort and add enumeration procedure than with the DFS and λ2 [3].



Number of test problems solved versus computation time[3]

5. CONCLUSION

The paper describes an approach to enhance the quality of Inductive Program Synthesis (IPS) Systems by training the neural network to guide the search in program space. There have been some problems in the real competitive programming competition that can be solved with a program in Domain Specific Language (DSL) which validates the relevance of this approach.
There are the following two limitations of this approach:
The programs synthesized are the simplest ones and many complicated problems may require more knowledge about making complex algorithmic solutions like dynamic programming which is beyond the approach's reach.
While this approach still expecting about LIPS's applicability to be successful, DSL becomes more complex when having a large number of functions in it and when this has to move to solve lengthy programming problems, the input-output examples become essentially less illuminating.
This approach can lead to the best future prospects of DeepCoder by using extensive power of machine learning to program synthesis [3].

6. REFERENCES

[1] Sebastian Riedel, Matko Bosnjak, and Tim Rockt¨aschel. Programming with a differentiable forth interpreter. CoRR, abs/1605.06640, 2016. URL http://arxiv.org/abs/1605.06640.
[2] Wang Ling, Edward Grefenstette, Karl Moritz Hermann, Tom´aˇs Koˇcisk´y, Andrew Senior, Fumin Wang, and Phil Blunsom. Latent predictor networks for code generation. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, 2016.
[3] Matej Balog, Department of Engineering, University of Cambridge, Alexander L. Gaunt, Marc Brockschmidt, Sebastian Nowozin, Daniel Tarlow, Microsoft Research. DeepCoder: Learning to write programs.ICLR 2017.
[4] Anshita Saxena, Aniket Saxena, Jyotirmay Patel, Merrut Institute of Technology, Merrut, Uttar Pradesh. ISSN 2249-3115 Vol.7, No. 1 (2017)
[5] Oleksandr Polozov and Sumit Gulwani. FlashMeta: a framework for inductive program synthesis. In Proceedings of the International Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA), 2015.
[6] Alexander L. Gaunt, Marc Brockschmidt, Rishabh Singh, Nate Kushman, Pushmeet Kohli, Jonathan Taylor, Daniel Tarlow, Microsoft Research. TERPRET: A Probabilistic Programming Language for Program Induction. https://arxiv.org/abs/1608.04428.
[7] https://www.newscientist.com/article/mg23331144-500-ai-learns-to-write-its-own-code-by-stealing-from-other-programs/
[8] https://blog.acolyer.org/2017/03/29/deepcoder-learning-to-write-programs/
[9] https://fossbytes.com/microsoft-ai-system-deepcoder/
[10] https://qz.com/920468/artificial-intelligence-created-by-microsoft-and-university-of-cambridge-is-learning-to-write-code-by-itself-not-steal-it/